



Eine komplette Anleitung für die Level-Editoren

++ "GtkRadiant" ++ "Q3Radiant" ++

Diese Anleitung soll dir helfen, deine ersten Gehversuche mit dem Editor Radiant zu ermöglichen dir den Umgang mit dem Editor näher zu bringen. Mit dieser Anleitung (Tutorial) will ich versuchen, dir bei dem Weg zum Erstellen einer kompletten Map zu helfen. Deswegen führe ich dich mit jedem Thema intensiver in das komplexe Thema "Mapping". Diese Anleitung ist absichtlich sehr ausführlich geschrieben, um auch auf die kleinsten Tücken und Fallen des Radiant einzugehen. Diese Anleitung soll nicht dazu da sein, um professionell Mappen zu lernen - lediglich soll es dazu da sein, bei den ersten Schritten zu helfen.

Du solltest natürlich schon etwas Vorwissen am PC mitbringen, da ich leider nicht jeden Schritt erklären kann, z.B. wie man einen Ordner erstellt. Wenn du also ein Anfänger am PC bist, solltest du dir das [hier](#) vorher [durchlesen](#).

Anregungen/Kritik/Fehler bitte an: Haradirki@Uni.de

Meine Homepage: <http://www.haradirki.de> oder <http://www.dead-in-bed.net>

Letzte Änderung am: 21.03.2003

Wenn du dir dieses Tutorial herunterladen willst, du findest den Download auf www.haradirki.de - das Tutorial kannst du als *.zip und als *.rar herunterladen. Ebenso findest du auf dieser Seite immer Updates usw. für diesen Tutor.

Gtk - RtCW - Q3A Radiant Tutorial, Copyright (c) 2002 Dirk Marmann, erstellt mit Wordpad

Übersicht über meine Tutorials:

[nach Oben](#) [nach Unten](#)

- [Installation des Gtk-Radiant](#)
- [Einen ersten Raum erstellen](#)
- [2 Räume verbinden](#)
- [Arbeiten mit Texturen](#)
- [Arbeiten mit Brushes](#)
- [Beleuchtung](#)
- [Spezielle Licht-Entities](#)
- [Beleuchtung](#)
- [Lichtkegel](#)
- [zusätzliche Lichteigenschaften](#)
- [Einen Spiegel erstellen](#)
- [Modelle benutzen](#)
- [Flüssigkeiten](#)
- [Türen](#)
- [Fenster](#)
- [Treppen](#)
- [zerstörbare Brushes](#)
- [Leitern](#)
- [Die PK3-Datei](#)
- [Accelerator-Pad, Jumpad](#)
- [Teleporter](#)
- [Feuer](#)
- [Curved Surfaces](#)
- [Die Hint-Textur](#)

- [Die Area-Portal-Textur](#)
- [Die Clip-Texturen](#)
- [Ein Pendel erstellen](#)
- [Ein rotierendes Objekt bauen](#)
- [Überreste nach dem Sprengen](#)
- [Eine drehende Tür bauen](#)
- [Schnee erstellen](#)
- [Rauch erstellen](#)
- [Einen Lift einbauen](#)
- [Globale Einstellungen in der Map \(z.B. Hintergrundmusik, Mapname usw.\)](#)
- [Das Startbild erstellen](#)
- [Eine Skybox erstellen](#)
- [nützliche Konsolen-Kommandos](#)
- [Die Map einbinden](#)
- [Alphatexturen](#)
- [Nebel](#)
- [Team-Startpunkte](#)
- [Kamera](#)
- [Sperrfeuer](#)
- [Shader](#)
- [Multiplayer-Skripting: Deathmatch](#)
- [Türe erstellen, die nur über einen Schalter zu entriegeln ist](#)
- [Türen erstellen, die sich nur über einen Schalter öffnen lassen](#)
- [*.bsp zu *.map konvertieren \(Map-Dateien aus vorhanden Levels erstellen\)](#)

EasyGen - in 13 Schritten zum perfekten Terrain:

[nach Oben](#)

[nach Unten](#)

- [EasyGen - die Installation](#)
- [EasyGen - das Erstellen eines Graustufenbildes](#)
- [EasyGen - eine kleine Einleitung](#)
- [EasyGen - Import des Graustufenbildes](#)
- [EasyGen - Nachträgliche Terrain-Änderungen](#)
- [EasyGen - erstes Texturing](#)
- [EasyGen - Terrains schliessen](#)
- [EasyGen - Export des Terrains](#)
- [EasyGen - Automatische Mehrfachtexturierung](#)
- [EasyGen - Texturen Scalling](#)
- [EasyGen - Manuelle Texturierung](#)
- [EasyGen - Texturfehler](#)
- [EasyGen - Korrekter Export in die Pk3-Datei](#)

Q3Map2 - der Kompiler der zweiten Generation:

[nach Oben](#)

[nach Unten](#)

- [Q3Map2 - Installation](#)
- [Q3Map2 - Terrain Lightmappen](#)
- [Q3Map2 - Bump - Mapping](#)
- [Q3Map2 - Compiler - Parameter](#)
- [Q3Map2 - Entity - Keys](#)

Anhang:

[nach Oben](#)

[nach Unten](#)

- [Allgemeine Fragen und Antworten](#)
- [Links](#)
- [Lexikon \(Wörtererklärung\)](#)

- [Impressum](#)
- [Tastenbelegungen](#)
- [Radiant-Fehlermeldungen - Error Messages](#)
- [Leaks - map is leaked](#)
- [Performance](#)
- [Common-Texturen](#)

- [Terragen](#)
- [Terragen-F.A.Q.](#)
- [Terragen Lexikon](#)

- [Easygen](#)
- [Easygen-F.A.Q.](#)
- [Easygen Lexikon](#)

Die Map-Dateien: (Dies sind Download-Links, d.h. beim Anklicken wird der Download meiner Map-Dateien gestartet. In diesen beiden Dateien findet ihr immer alle Beispielmeps und Ergebnismeps, die ich zum jeweiligen Tutor gemacht habe)

- [Q3-Map-Dateien](#)
- [RtCW-Map-Dateien](#)

Gtk - RtCW - Q3A Radiant Tutorial, Copyright (c) 2002 - 2004 Dirk Marmann, erstellt mit Wordpad

Dieses Tutorial ist grundsätzlich Freeware und darf für den persönlichen Gebrauch weiterkopiert werden. Es darf allerdings NICHT ohne meine Einverständniss auf anderen Homepages als HTML-Datei oder zum Download angeboten werden. Es darf nichts hinzugefügt, verändert oder gelöscht werden. Weiterhin darf es nicht auf digitalen Medien, z.B. CD-Medien veröffentlicht werden, wenn ich nicht ausdrücklich meine Einverständniss dazu gegeben habe. Alle Rechte bleiben beim Autoren. Wird aus diesem Tutorial zitiert, wäre ich sehr dankbar, wenn ihr dabei auf eine meiner beiden Homepages <http://www.haradirki.de> oder <http://www.dead-in-bed.net> hinweist.

183759

<http://www.haradirki.de>

<http://www.dead-in-bed.net>



[.Home](#)
[.Tutorials](#)
[.Downloads](#)
[.Allgemeines](#)
[.Forum](#)
[.Haradirki](#)

Unterlinks

[Arbeiten auf Haradirki.de](#)

[Sommer ist zurück!](#)

[Deaktuell](#)

[Frohe Weihnachten](#)

[haradirki.de / index.htm](http://www.haradirki.de/index.htm)

Herzlich Willkommen auf www.haradirki.de

Arbeiten an Haradirki.de

NEWS
auf www.haradirki.de

Lange war der Tutorial-Bestand auf haradirki.de sehr konstant - zu konstant. Nun hat sich Haradirki etwas neues einfallen lassen - ein zweites Tutorial für Half-Life 2. Diesmal geht es weniger um Tutorials wie man einen Lift, Türen etc. erstellt, sondern wie man eine richtige, voll spielfähige Map erstellt. Dabei wird es unter anderem zunächst um theoretisches wie Map-Planung, Map-Optimierung, Map-Performance usw. gehen. Dass es diesmal aber nicht nur so

theoretisch bleibt wie beim alten Tutorial wird die Planung dann auch in die Tat umgesetzt. So entsteht dann Schritt für Schritt eine Map, die im finalen Tutorial natürlich soweit vollendet wird, dass man sie hinterher auf einem Server laufen lassen kann. Damit soll das Tutorial alle Leser des "alten" Tutorials weiter begeistern, aber auch neue Leser gewonnen werden, denn so will Haradirki es gerade für Clans ermöglichen, dass sie ihre eigenen Trainingsmaps erstellen können. Weiterhin wird gerade im Hintergrund das Half-Life 2 Tutorial überarbeitet, denn Forenmember Alva hat einige Fehler gefunden die ich natürlich korrigieren werde.

Sommer ist zurück!

Wir haben die lange Durststrecke (mit viel Regen) tatsächlich überstanden. Nach dem ja der August etwas verregnet war, haben wir nun im September wieder schöne Temperaturen, die noch einmal Lust auf Freibad, See und natürlich das Meer machen. Dazu habe ich mir gleich überlegt, sollte ja auch die Bekanntschaft und der passende Drink nicht fehlen. Leider kann ich bei der Bekanntschaft nicht helfen, wohl aber bei dem Drink. Daher findet ihr nun 2 neue Drinks unter Allgemeines -> Drinks. Momentan bastel ich weiter an ein paar Themen für das Half-Life 2 Tutorial, wahrscheinlich werde ich aber auch bald ein ganz neues Tutorial veröffentlichen. Doch dazu wird noch nichts verraten, da ich mich da selbst noch etwas einarbeiten muss. Übrigens habe ich mich beschlossen, 1 Jahr nach meinem Besuch bei GIGA TV die GIGA Map zu überarbeiten und zu erweitern. Momentan überlege ich auch, ob ich dazu nicht auch ein gesondertes Tutorial erstellen soll. Man darf also gespannt sein.

Deaktuell!

Tatsächlich war nun auf Haradirki.de über ein halbes Jahr Weihnachten. Aber da ja bekanntlich im Himmel jeden Tag Weihnachten ist (siehe Sinn des Lebens) wird der versierte Besucher der Seite nichts dagegen haben diese Meldung nochmals zu lesen. Seit einer Woche bin ich nun von meiner kleinen USA-Reise zurück und es gibt auch einiges zu berichten, näheres dazu gibt es im Forum zu finden.

Was Spiele angeht, habe ich mich sozusagen "neu" orientiert. Nein, ich bin nicht von 3D-Shootern abgewichen, lediglich das System hat sich geändert. So bin ich nun von der Playstation Portable

äusserst angetan und fiebere nun der PlayStation 3 entgegen. Allerdings ist das für mich noch kein Grund, mit dem Mapping aufzuhören. Jedoch werden die Tutorials jetzt nicht mehr so schnell aufeinander releast werden, aber das seid ihr ja von dem letzten halben Jahr gewöhnt. Im Forum allerdings dürft ihr mit mir täglich aktuell über diverse Themen diskutieren, z.B. auch über die neue Volkssucht World of Warcraft oder sich in deutsche Fahnen zu wickeln. Zwar war mir das etwas befremdlich, jedoch habe ich mich schon gut eingewöhnt und man kann mich jetzt als Mumie in ein Museum stellen.

Frohe Weihnachten!

Haradirki.de wünscht allen Besuchern und besonders auch allen Nutzern des [Forums](#) frohe Weihnachten! Ich hoffe, ihr bekommt viele Geschenke und könnt ein bisschen entspannen und vom nerven-aufreibenden Geschenke-einkaufen erholen. Nach dem es ja lange Zeit ruhig war, release ich jetzt 2 neue Themen für HL2 und eines für Quake4. Darin geht es um 2D Skyboxen, 3D Skyboxen und für Quake4 gibt es einen Artikel über Leaks. Momentan habe ich ein kleines Projekt am laufen, wobei ich mir noch nicht sicher bin, wie weit ich es hier veröffentliche. Auf jeden Fall lohnt ein genauerer Blick ins Forum, dort werde ich demnächst die ersten Kostproben davon veröffentlichen. Bis dahin wünsche ich euch eine erholsame Zeit und viel Spaß allen Mappern, egal ob ihr euch im World Edit für F.E.A.R., im Quake Edit für Quake4, im Hammer Editor für Half-Life2 oder sonstwo herumtreibt. Übrigens lohnt sich auch der ein- oder andere Ausflug zu ein paar Feiern, die gibts ja zu der Zeit an jeder Ecke. Hier aber noch die Links zu den neuen Themen:

- [Eine 2D Skybox erstellen \(HL2\)](#)
- [Eine 3D Skybox erstellen \(HL2\)](#)
- [Leaks \(Quake 4\)](#)

[Über diese Seite...](#) | [Site Map](#) | [Impressum](#) |

© by Haradirki. All rights reserved.
best view with eyes open

183759



zur Verwendung:

Dieses Tutorial habe ich für all die Leute geschrieben, die endlich einmal ihre erste, eigene Map bauen wollen und dies soll eine einfache, leicht zu lesende Einführung zu diesem Thema sein.

Was du zum "Mappen" mitbringen solltest:

- **Du solltest ein wenig Erfahrung mit dem "Handling" von Computern mitbringen:**

D.h. man braucht dafür kein Programmierer zu sein, jedoch solltest du ein Programm installieren und mit einem Textverarbeitungsprogramm umgehen können, sowie Ordner erstellen und Dateien kopieren können. Weisst du all das schon? Wenn ja, dann scheist du ja schonmal gewappnet für das, was auf dich zukommt. Hast du hingegen deinen PC erst ein paar Tage und willst morgen früh schon die erste Mega-Map veröffentlichen, dann könnte es schwierig werden... . Zumindest solltest du dann Rat bei Freunden/Bekannten suchen und dir so unter die Arme greifen lassen.

- **Newbies sollten wirklich ganz von vorn anfangen:**

Newbies sollten auf alle Fälle jeden Tutor durcharbeiten oder zumindest überfliegen, da ich später nichtmehr alles so haarklein erkläre, wie am Anfang, z.B. wie ich alle Seiten eines Würfels einzeln mit Texturen belege. Jedoch biete ich natürlich Links zu den Stellen an, die diese Themen genauer behandeln. Im späteren Verlauf dieses Tutorials wirst du sicher feststellen, dass sich viele Arbeitsschritte wiederholen, daher wirst du mit der Zeit auch immer sicherer werden.

- **alles ganz genau lesen:**

Wenn du etwas durchgelesen hast, und es will bei dir in Radiant nicht klappen, dann erstmal keine Panik, sondern alles ganz genau nochmal lesen. Notfalls Stück für Stück. Ich habe in diesem Tutorial alle Schritte durchprobiert, und mehrmals getestet, dass du das selbe Ergebnis auf dem Bildschirm hast, wie ich. Außerdem habe ich zu jedem Thema eine Beispielpackung dazugepackt, in der du dir alles in Ruhe anschauen kannst. ABER: Auch ich bin nur ein Mensch und mache Fehler, genau wie alle anderen auch - und solche Fehler können sich natürlich auch hier eingeschlichen haben. Sollte dies der Fall sein, schreib' mir eine Mail, und zwar an:

Haradirki@Uni.de

ALLERDINGS: Ich will dann keine Mail erhalten, wie "mhhh.. wie geht denn das jetzt mit...." und ich stelle darauf fest, dass ich alles haarklein im Thema davor behandelt habe und hier nur eine exklusive Erklärung gewollt ist. Ich habe auch nicht soviel Zeit, deshalb habe ich versucht, dieses Tutorial so ausführlich wie möglich zu gestalten und auf mögliche Fragen im Vorraus zu reagieren. Zudem solltet ihr die aktuelle Version der jeweiligen Radianten benutzen (Q3Radiant 202, GtkRadiant 1.2.8 nightly)



Die Version "202" ist die stabilste und aktuellste Version des Radianten, damit wurde dieses Tutorial erstellt.

Die Version "GtkRadiant 1.2.5 nightly" ist ebenfalls die aktuelle Version des Radianten für RtCW. Sie sollten sich gegebenenfalls diese Versionen herunterladen, damit mit der gleichen Version arbeiten, wie ich.



(Zumal ist dann auch die Hilfe von mir effizienter, wenn wir mit der gleichen "Basis" arbeiten).

ACHTUNG: Hier solltest du beachten, dass es ein Update ist. Du brauchst erst die Version 1.2.1. Hier **MUSST** du die datei "q3map.exe" suchen (ist normal im "c:\games\RtCW\Radiant"-Verzeichnis) und diese in einen anderen Ordner kopieren - diese brauchst du als Sicherheitskopie. Anschließend installierst du die Version 1.2.8 und kopierst wieder die Datei "q3map.exe" in das Verzeichnis "Radiant" zurück. Zwar gibt es schon die Version 1.2.9, jedoch habe ich diese nicht installiert.

- **du willst für beide Radianten mappen:**

Im Grunde ist das kein Problem. Ich verwende für Q3 den Q3Radiant - für RtCW den GtkRadiant. So solltest du das auch handhaben, da so Probleme vermieden werden können - schliesslich kommt man so auch nicht durcheinander.

- **Eigeninitiative zeigen...**

Dieses Tutorial soll dir die ersten Schritte im Radianten ermöglichen, in den verschiedenen Themen versuche ich, dir alles so gut wie möglich zu erklären. Jedoch ist dies ein "Allgemeines Tutorial" und ich kann nicht jede Beschreibung liefern, die es gibt, z.B. die x verschiedene Möglichkeiten, einen Lift zu erstellen - das geht natürlich nicht. Daher musst du schon etwas Eigeninitiative zeigen und meine Themen durcharbeiten. Aber ich versuche, euch alles mit auf den Weg zu geben, dass du auch in Zukunft deine eigenen Maps bauen kannst.

- **Es klappt immer noch nicht:**

ja, dann gibt es nur noch folgende Möglichkeiten, entweder du schreibst mir eine Mail (Haradirki@Uni.de) oder du besuchst meine Homepage (<http://www.haradirki.de>).

- **Du willst dieses Tutorial auf einer anderen Homepage zum Download anbieten:**

Nun, um eins gleich klarzustellen...Dieses Tutorial darfst du NICHT auf einer anderen Seite anbieten. Es war eine harte Arbeit, dieses Tutorial zu erstellen und ich will dieses Tutorial nur auf einer Seite sehen, nämlich [meiner](#). Vielleicht wird es hier einige Ausnahmen geben, diese Leute werde ich aber selbst ansprechen.

Was ich dir aber anbieten kann, ist ein Banner, dass du zu deinen Links setzen kannst (sofern dazu Interesse besteht). Dann kannst du entweder das Banner dieses Tutorials benutzen, oder einen reinen Textlink anbringen. Verlinke einfach diese Seite:

<http://www.haradirki.de> oder <http://www.kickme.to/haradirki>

Banner von Haradirki's GtkRadiant & Q3Radiant Tutorial:





WICHTIG: Ich will auch nicht, dass du download-links direkt auf mein Tutorial setzt, sei so ehrlich und benutz' wirklich <http://www.haradirki.de> oder <http://www.kickme.to/haradirki>

- Im Tutorial gibt es aber ein paar dicke Räschtschreibfähler:

Umguck - öhbm... was für Rechtschreibfehler denn ??? *hüstel*. Ich hoffe, du verstehst trotzdem, was ich hier niedergeschrieben habe, und kannst etwas damit anfangen. Was die Rechtschreibfehler angeht:

"Die Rechtschreibfehler auf dieser Seite dienen der allgemeinen Belustigung"

"Wer einen Rechtschreibfehler findet, darf ihn behalten"

- Wieso ist das Tutorial eigentlich so ernst geschrieben???

Nun, das ist eine gute Frage :D... aber man muss dazu sagen, dass ständige Smilies :D :P :) im ganzen Text verteilt nicht :S gerade für ein :-) besseres Verständnis sorgen :D Allerdings ist hier nicht alles so streng gehalten, immerhin soll das Mappen ja Spass machen :-)

[zurück zur Hauptseite](#)

183759



Zur Installation:

Zunächst einmal ist der GtkRadiant ein wirklich mächtiges Werkzeug, um damit Maps, Welten (wie du dazu sagen willst) zu erstellen. Wer einmal mit diesem "Hobby" angefangen hat, der wird sicher auch bald erste Erfolge mit diesem Editor haben. Erst recht wirst du dann auf den Geschmack kommen und vielleicht weitermachen. Natürlich gibt es hin- und wieder auch Engstellen, die es zu überwinden gilt, aber du solltest dich nicht bei jedem Problem entmutigen lassen. Für (fast) jedes Problem hier im Radiant gibt es eine Lösung und so findest du auch sicher schnell deine Lösung.

Solltest du allerdings bei Fragen, die hier nicht behandelt werden, selbst keine Lösung finden, wende dich einfach an die Mapper-Gemeinde. Zum Beispiel die von

<http://www.planetquake.de> - hier solltest du das Editing-Forum aufsuchen und dort um Rat fragen.

<http://pub64.ezboard.com/bthemappingcommunity> - hier ist mein Forum, hier kannst du mich direkt fragen.

In den nachfolgenden Kapiteln benutze ich die aktuelle Version Build 202 (Q3Radiant) und Build 1.2.4 (GtkRadiant). Ihr könnt auch ältere Versionen benutzen, jedoch kann es hier zu Unterschieden kommen. Am besten lädst du dir einfach die neuste Version von <http://www.qeradiant.com> runter.

Beachte bitte auch, das du als Anfänger die Kapitel der Reihe nach durcharbeitet. Wenn du noch nicht mit dem Radiant vertraut seid, macht es keinen Sinn, gleich zu den schweren Themen zu springen, da du hier vielleicht nichtmehr die Schritte verfolgen kannst, da ich mich dort kürzer halte, als am Anfang. Denn so kann leicht Frust aufkommen und das will ich auch verhindern.

Ich benutze den Q3Radiant 202. Es gibt noch eine andere Version und zwar den GTK Radiant, der eine Linux-Like ähnliche Oberfläche aufweist (plattformübergreifend). Dieser ist aber nahezu identisch mit dem normalen Q3Radiant. Es bleibt dir überlassen, was für ein Editor du benutzt. Jedoch solltest du dann nicht stutzig werden, wenn z.B. die Bilder hier von dem Design her etwas abweichen.

Jedoch benutze ich bei RtCW den GtkRadiant, also kannst du dann auch dort sehen, wie es da aussieht. Dazu schaust du dir einfach die Bilder an, die mit dem RtCW-Logo gekennzeichnet sind.



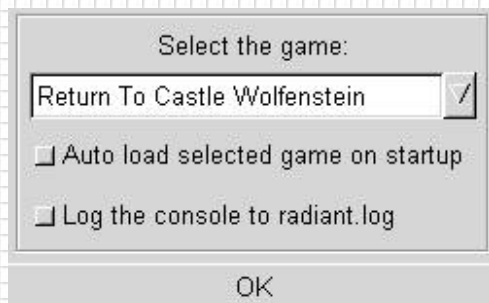
Hier benutze ich den GtkRadiant. Das hat den Vorteil, dass ihr auch den GtkRadiant im Einsatz sehen könnt. Der GtkRadiant ist von der Grafik eher Linux-like aufgebaut, also wundert euch nicht, wenn z.B. der Mausfeil nun etwas anders aussieht, als im Windows.



Hier brauchen die Quake-Mapper von euch garnichtmehr weiterlesen, sofern ihr nicht mit dem Gtk-Radiant arbeitet. Wenn doch, könnt ihr das hier mal überfliegen, vielleicht werden eure Fragen hier auch geklärt.

Nungut, hier werden wir ersteinmal den GtkRadiant installieren. Wenn du den Radiant schon installiert hast, kannst du gleich die Installation starten. Hast du den Radiant aber noch nicht, musst du erst auf <http://www.qeradiant.com> gehen, und dir dort den Radiant runterladen. Nach der Installation präsentiert sich der GtkRadiant mit einem kleinen Fenster, das so aussieht

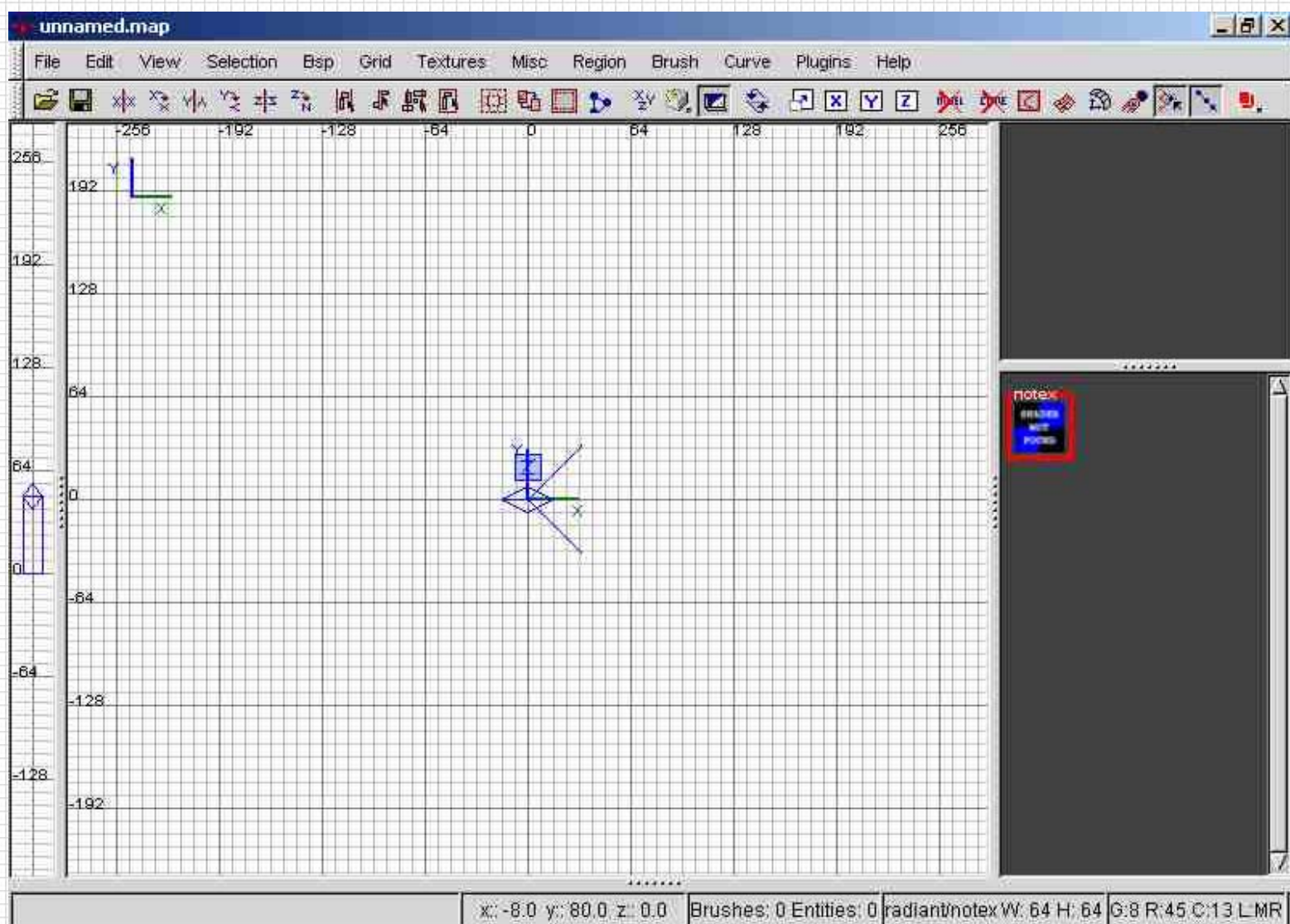
Die Installation:



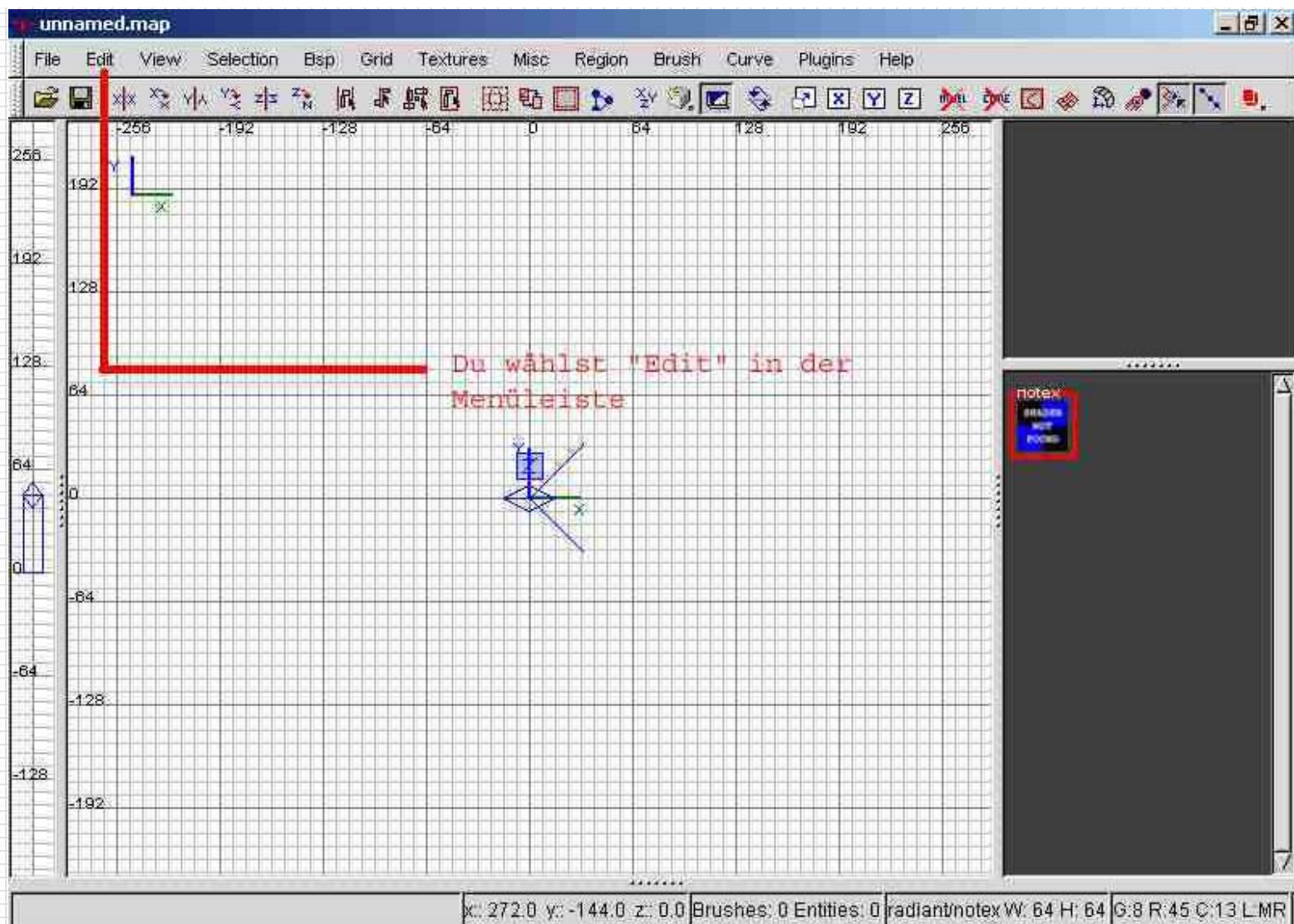
Das haut uns natürlich nicht um, wir wählen natürlich "Return To Castle Wolfenstein" und drücken auf OK.

(Wenn in der weissen Leiste ein anderes Spiel steht, drückt ihr den Pfeil rechts davon und wählt aus dem "Pull-Down-Menü" RtcW aus.)

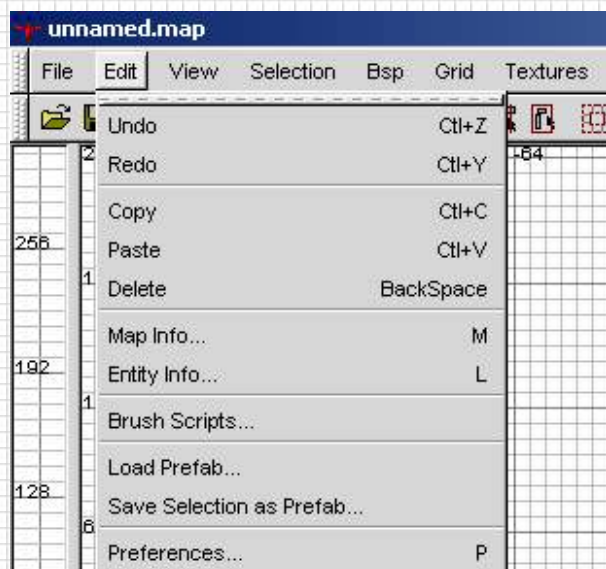
Nun startet der Gtk-Radiant:



Natürlich könnten wir jetzt schon mit dem Mappen anfangen, aber diese Sicht ist sehr sehr ungeschickt. Um diese zu ändern, gehen wir in der Menüleiste (wo File, Edit, View usw. steht) auf "Edit":

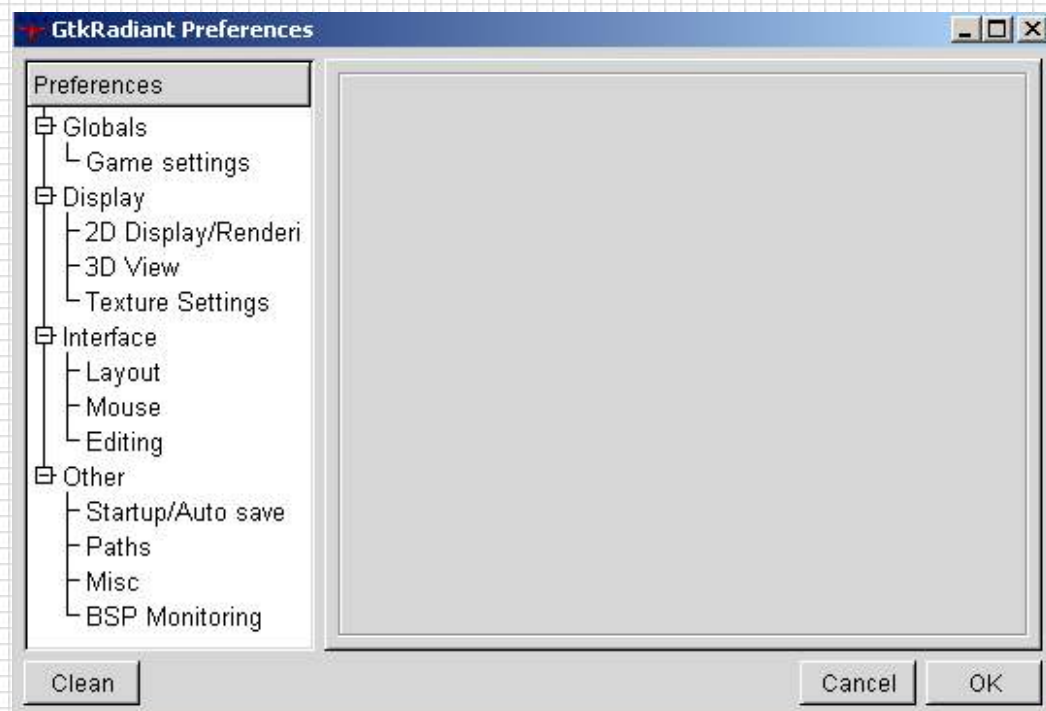


und wählen die letzte Option "Preferences":

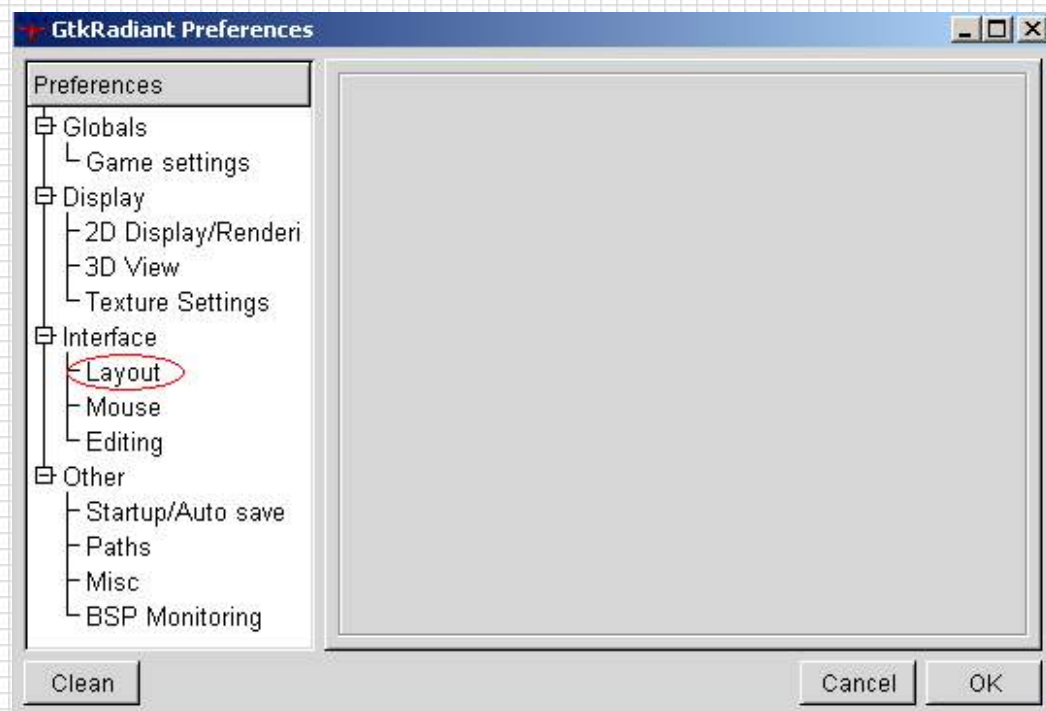


Nun können wir die Voreinstellungen des GtkRadiants verändern. Natürlich werden wir nur die Sicht umändern, der Rest soll so bleiben, wie er ist. Natürlich kannst du später deinen Radiant so einstellen, wie du willst. Du musst nur daran denken, dass du dann mit anderen Einstellungen mappst, wie ich.

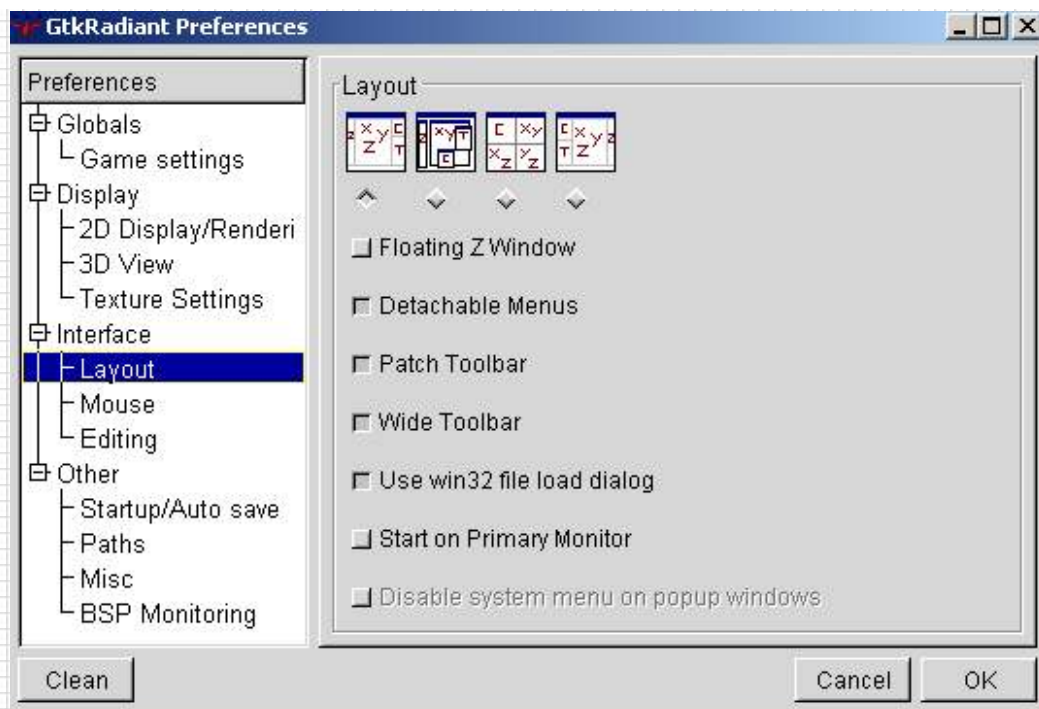
Hier sehen wir nun die Preferences:



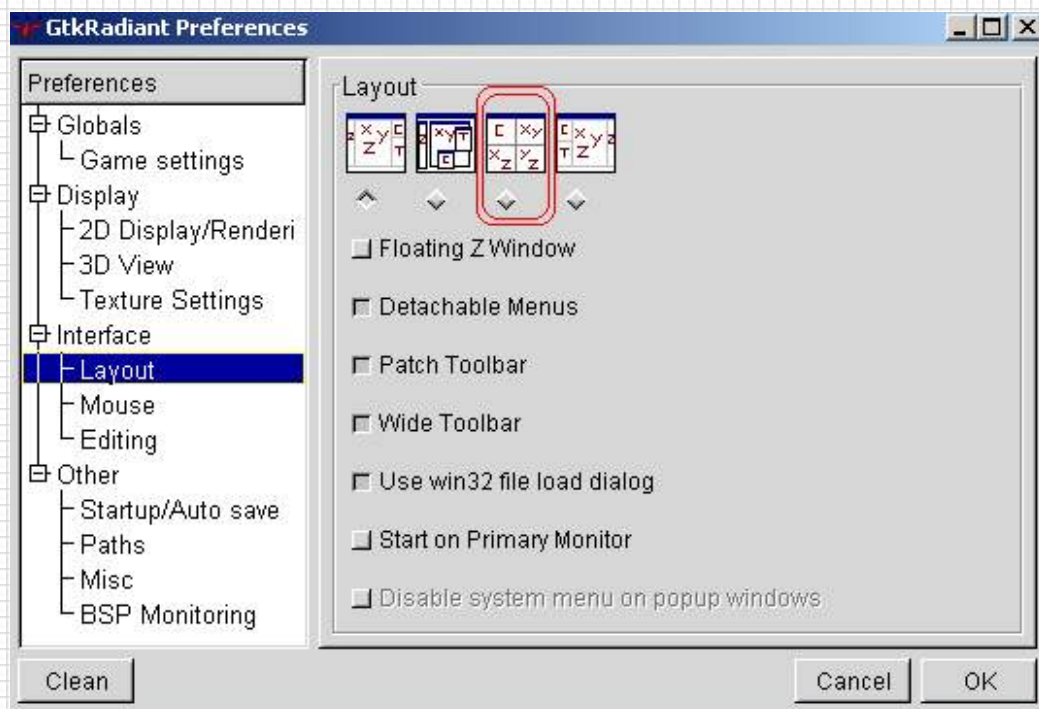
In diesem Menü wählen wir unter "Interface" den Punkt "Layout":



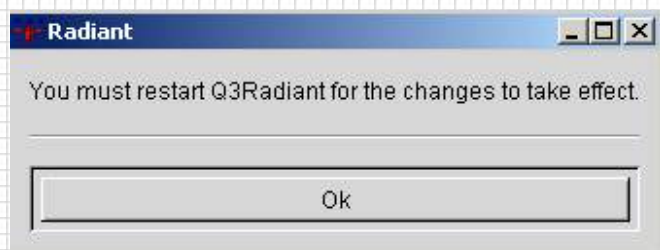
Es erscheint rechts im grauen Kasten ein neues Menü:



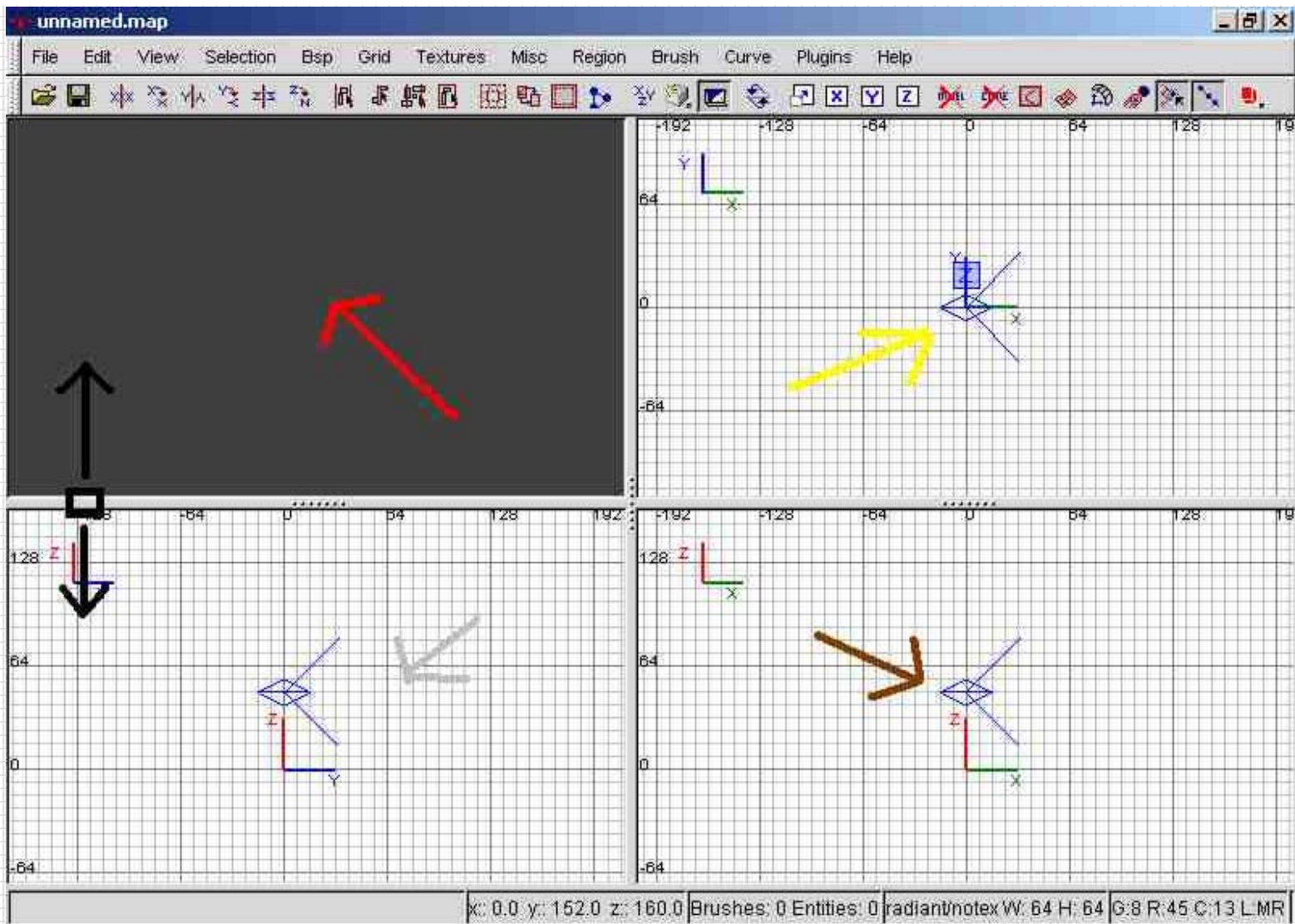
Hier wählen wir eine andere Ansicht in unserem Radiant.



Wir drücken nun auf "OK". Der GtkRadiant will jetzt neustarten, es erscheint diese Meldung:



Also starten wir den Gtk-Radiant neu. Nun sieht das ganze so aus:



Etwas zu diesen Fenstern:

Der gelbe Pfeil kennzeichnet die Draufsicht (Top Fenster)

Der graue Pfeil kennzeichnet die Seitenansicht

Der braune Pfeil die Vorderansicht

Der rote Pfeil kennzeichnet die 3D Ansicht

Wenn du diese Fenster so gross haben willst, wie ich das gemacht habe - kannst du einfach die grauen Ränder verschieben. Dazu klickst du auf eine solche graue Linie (habe ich mit schwarz gekennzeichnet) und ziehst nun den Rand nach oben oder unten.

So, das hätten wir schonmal

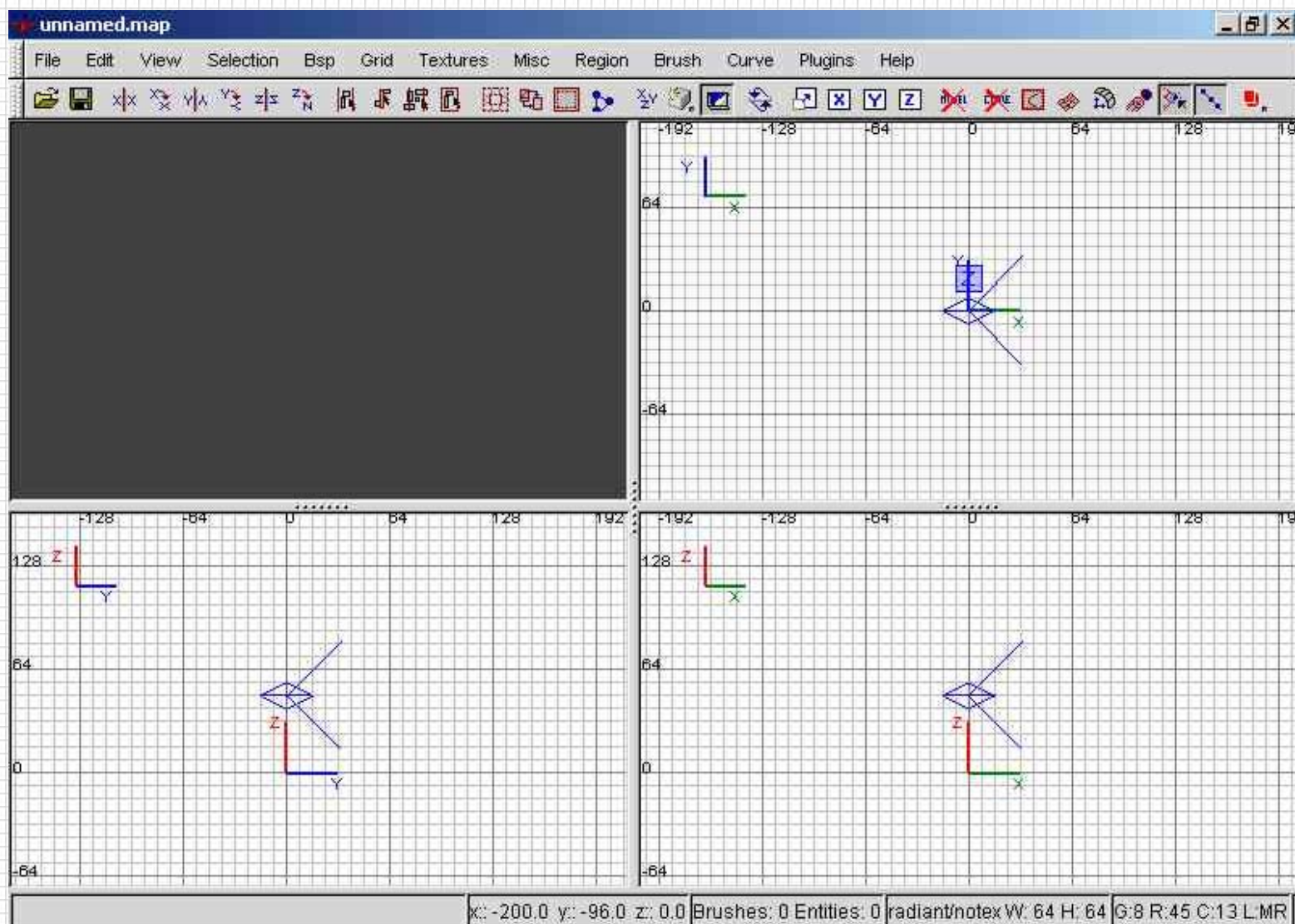
[zurück zur Hauptseite](#)

183759



Einen Raum bauen:

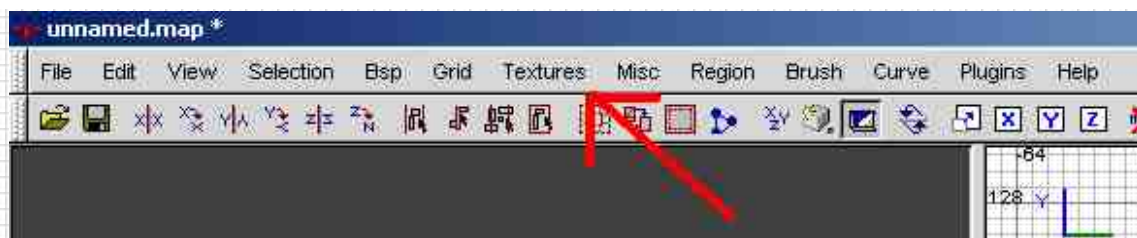
Zunächst einmal müssen wir uns nun im Editor zurecht finden. Bei beiden Editoren (sowohl Q3Radiant, als auch GtkRadiant) sieht nun der Bildschirm so aus:



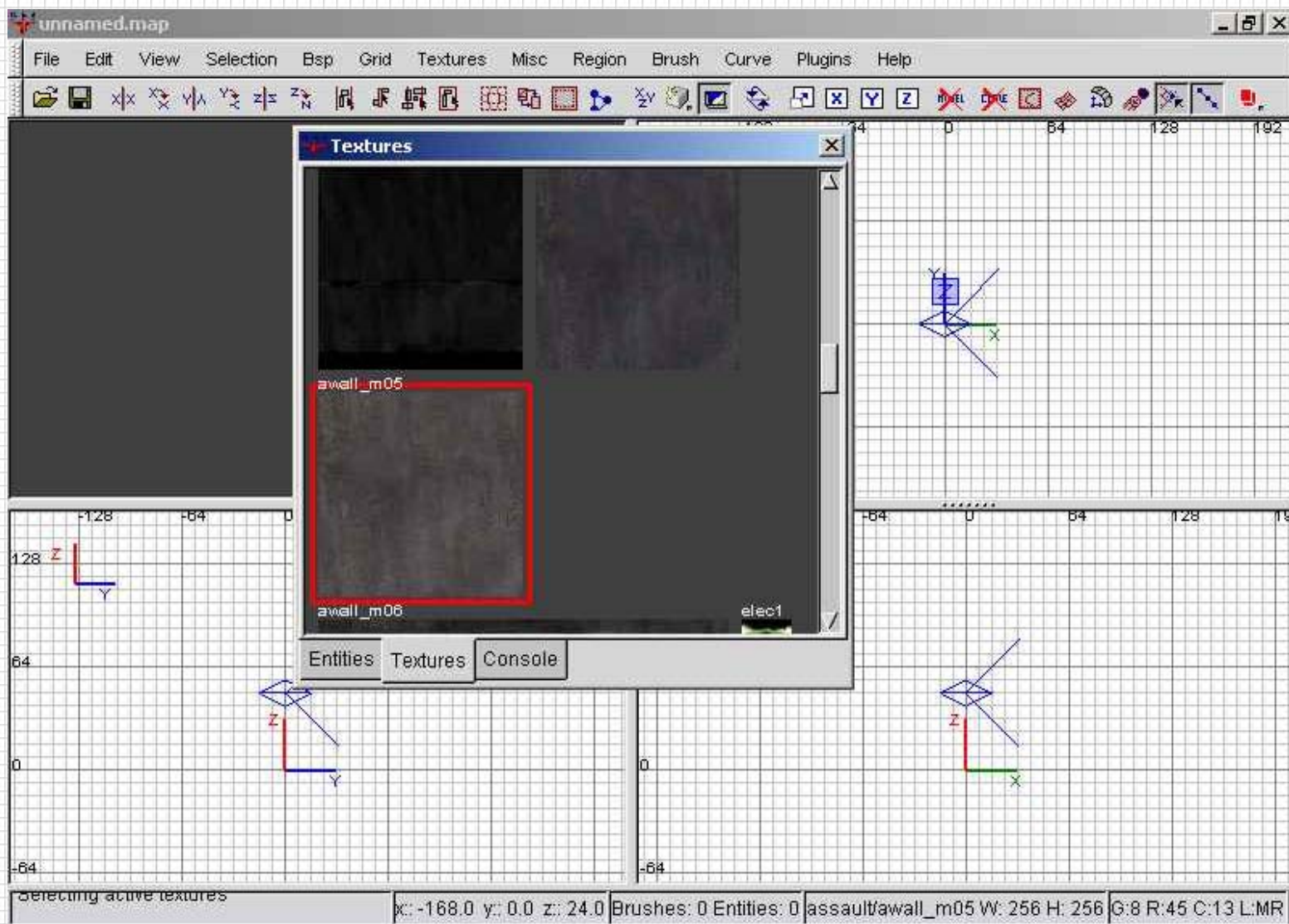
Wenn dein Bildschirm anders aussieht, schau bitte mal [hier](#) vorbei, [hier](#) erkläre ich dir, wie man den Bildschirm einstellt, dass man diese Sicht hat.

Nungut, du siehst jetzt vor die die 4 Fenster, 3 davon im Koordinatenkreuz, eines zeigt uns später die Brushes mit den Texturen an - also unsere Map. In diesen 3 Feldern siehst du nun jeweils ein blaues "Ding", das aussieht, wie ein offener Schnabel. Das ist die Kameraposition. Übrigens, das obere rechte Fenster ist die Draufsicht, unten links die Seitenansicht und unten rechts die Vorderansicht:

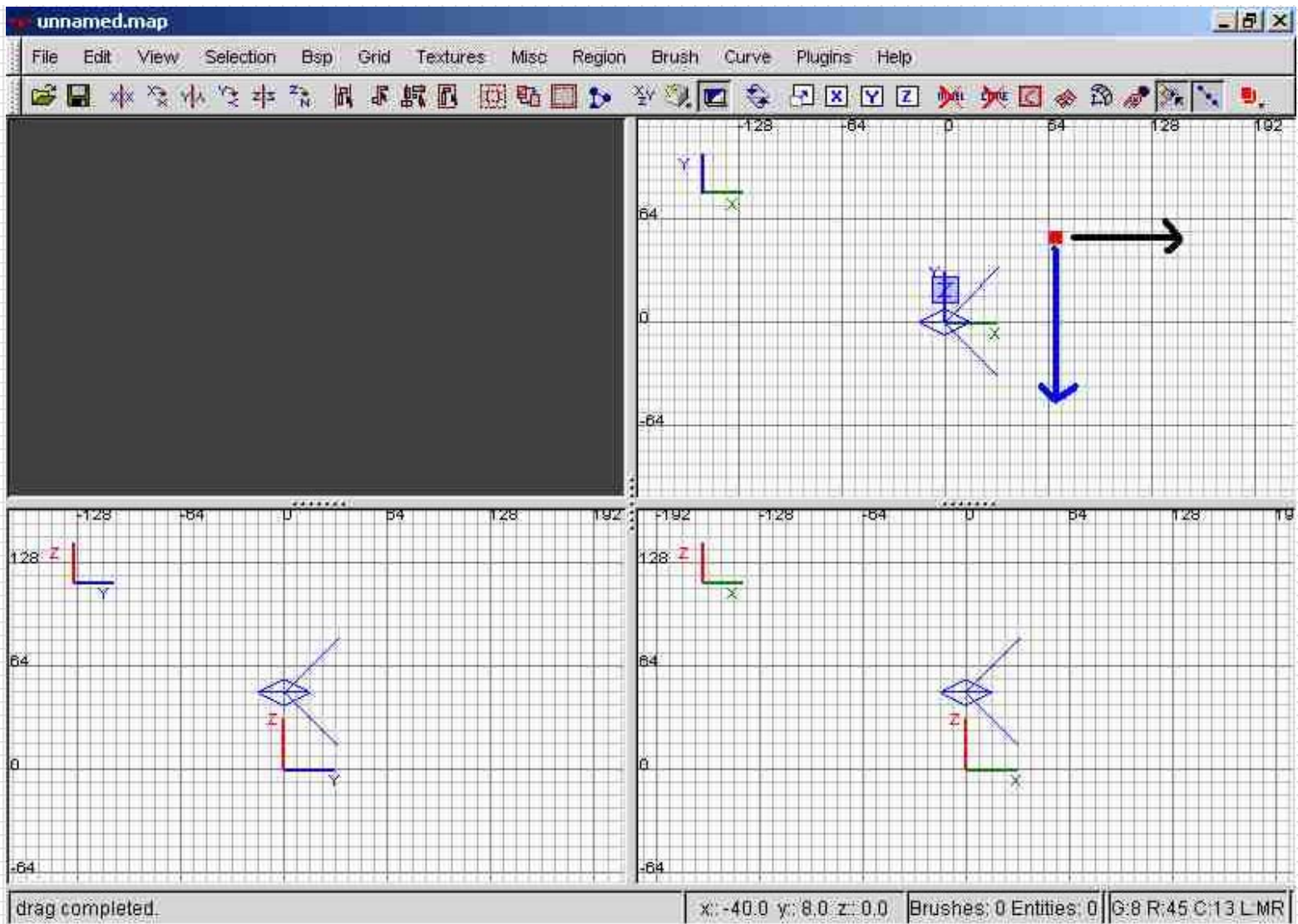
Um unseren ersten Raum zu bauen, benötigen wir ersteinmal eine Textur. Dazu klickst du oben im Menü auf Textures, und wählst eine Kategorie wie z.B.den ersten Raum zu basteln, brauchen wir erst eine Textur. Wir klicken dazu im Menü auf "Textures" und wählen zb. "Assault" (wenn du eine Quake-Map machen willst, z.B. "gothic/block") aus.



Dann dauert es ein wenig, hier werden die Texturen geladen. Dann sollte die Sanduhr bei dir verschwunden sein, und nun drückst du den Buchstaben "T". Nun siehst du die Texturen, die sich in diesem Verzeichnis "Assault" (für die Quake "gothic/block" befinden. Ist dir dieses Texturenfenster zu klein, kannst du es natürlich grösser ziehen. Dazu klickst du mit der Maus auf das untere rechte Eck des Texturenfensters, und nun kannst du das Fenster vergrößern oder verkleinern, wie du es willst. Nun haben wir also erstmal eine Menge an Texturen, wähle jetzt bitte EINE Textur aus - ich wähle z.B. die Textur "assault/awall_m05".

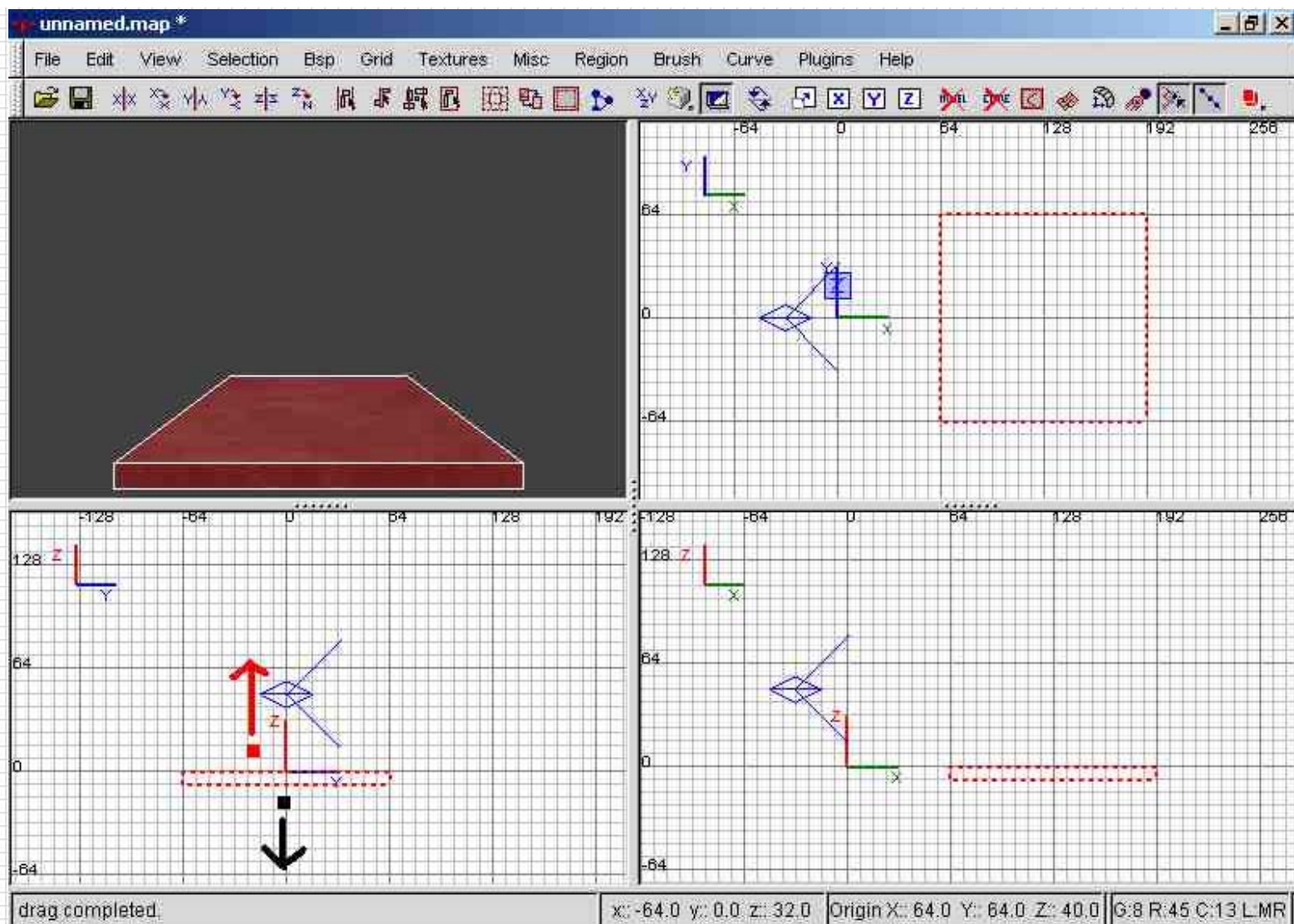


Nun werden wir unseren ersten Raum bauen. Dazu benötigen wir erstmal einen Bodenbrush. Dazu gehst du mit der Maus auf die Top-Ansicht (das Fenster mit dem roten Punkt). Fahr jetzt mit der Maus auf die Stelle, die ich hier mit dem roten Punkt gekennzeichnet habe - dann bewegst du die Maus nach rechts (das habe ich dir mit dem schwarzen Pfeil gekennzeichnet) und anschliessend bewegst du die Maus nach unten (das habe ich dir mit dem blauen Pfeil gekennzeichnet)



Und jetzt ziehst die Maus langsam nach rechts unten. Dabei hältst du natürlich die Maustaste gedrückt. Dabei entsteht ein ein gestrichelter, roter Kasten:

Übrigens, im oberen linken Fenster (das war ja bisher immer leer) siehst du jetzt sehr schön die 3D Ansicht deiner Map (momentan halt nur der Boden, alles weitere machen wir ja noch).

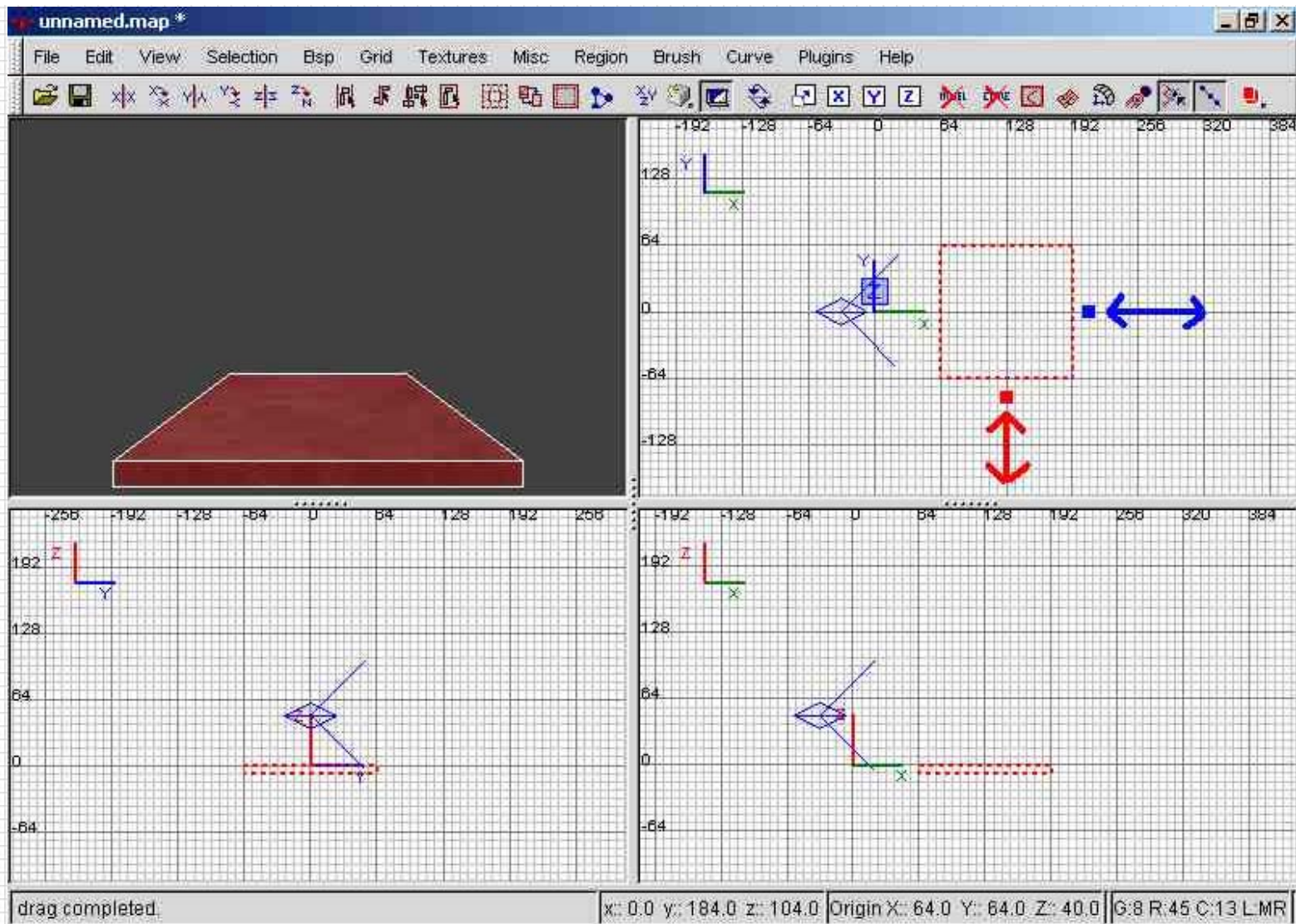


ACHTUNG: Ist dein Brush höher als 8 Units, kannst du ihn nach unten verkleinern/vergrößern, indem du unter den Brush klickst (das habe ich dir mit schwarz gekennzeichnet) und nach oben/unten ziehst.

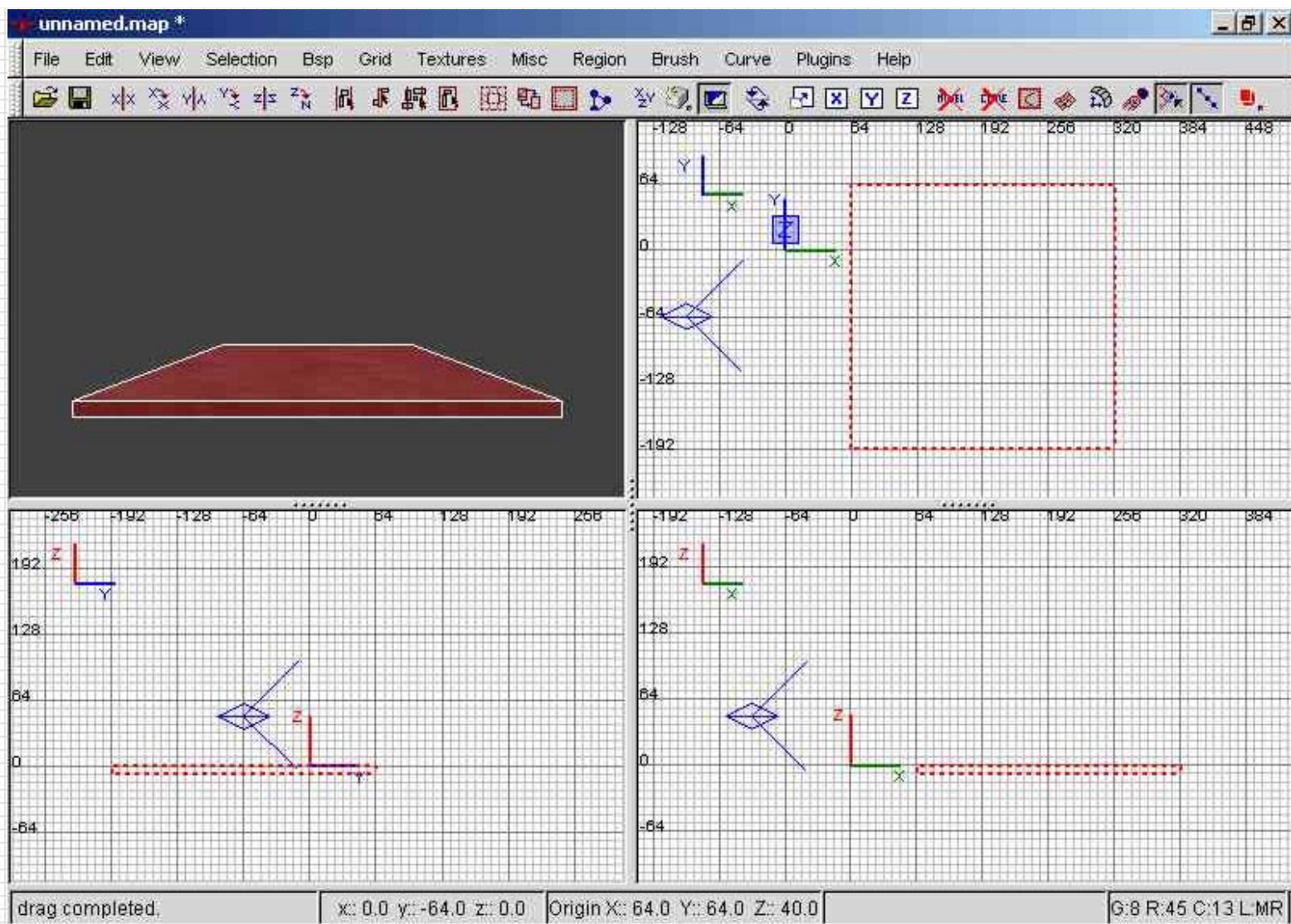
Du kannst ihn auch nach oben verkleinern/vergrößern, indem du über den Brush klickst (das habe ich dir mit rot gekennzeichnet) und nach oben/unten ziehst.

So, damit hast du deinen Boden für deinen ersten Raum gebaut. Natürlich ist diese Fläche etwas klein. Nun kannst du die Fläche nachträglich größer machen, indem du im Topfenster unterhalb deines Brushes klickst, und die Maus nach unten ziehst (das habe ich dir mit rot gekennzeichnet). Damit wird dein Boden-Brush nach unten gezogen -> er wird größer. So kannst du den Boden-Brush auch seitlich verlängern, wenn du das noch tun willst. Dazu klickst du mit der Maus rechts neben den Brush (das natürlich noch im Top-Fenster) und ziehst die Maus nach rechts (das habe ich dir mit blau gekennzeichnet). So wird der Brush nach rechts gezogen -> also nach rechts verlängert. Mit der gleichen Methode kannst du den Brush auch verkleinern, dazu musst du nur die Maus jeweils in die andere Richtung bewegen.

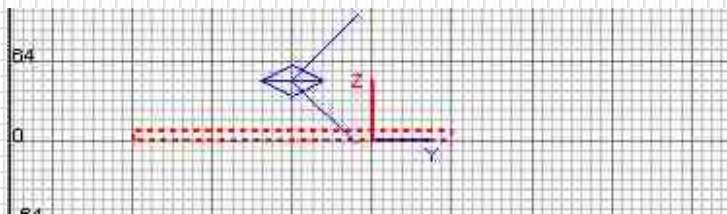
Dazu ein Bild:



So, nun haben wir den Brush groß genug gemacht:



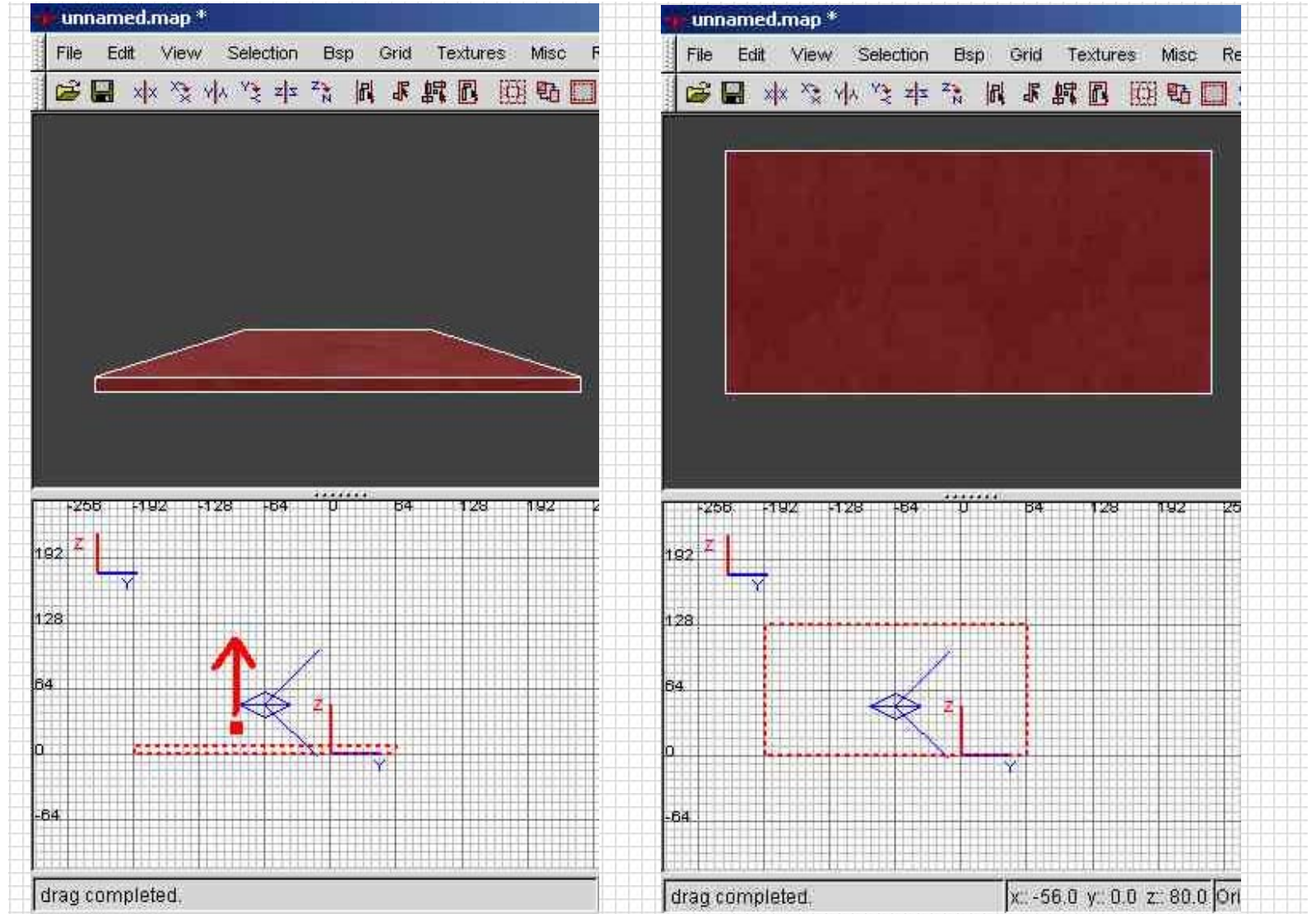
Nun hängt aber unser Brush unter der 0-Linie. Wir wollen ihn allerdings über der 0-Linie haben. Dazu klickst du IN den Brush und bewegst deine Maus nach oben, bis der Brush auf der 0-Linie "liegt":



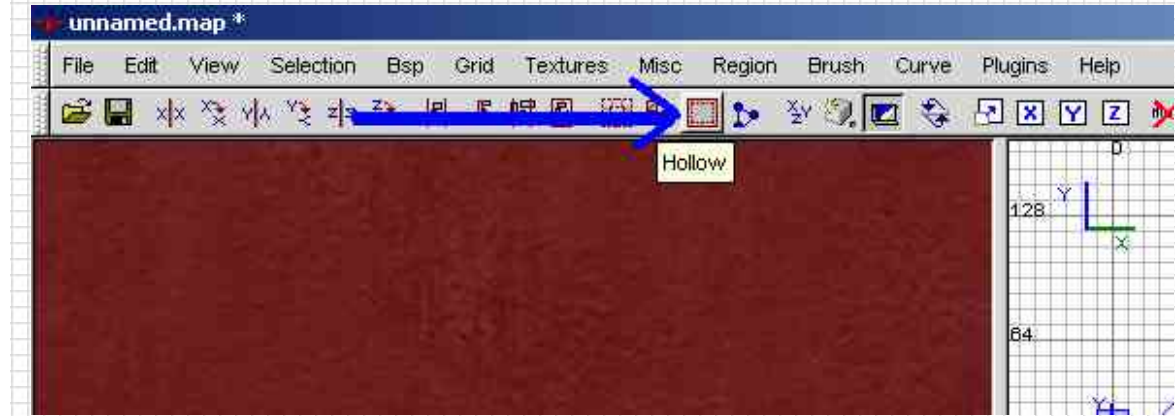
Nun bewegen wir unsere Maus ins linke untere Fenster. Hier klickst du auf den roten Punkt, den ich dir wieder markiert habe, und bewegst die Maus nach oben, und zwar solange, bis der Brush 128 Einheiten (Units) hoch ist: Nun benötigen wir ja noch die Höhe von den Wänden. Im Grunde ist hier die Höhe egal, da wir die Textur notfalls noch nachträglich angleichen können. Im Beispiel wählen wir aber dennoch eine Höhe von 128 Units, da wir mal relativ klein anfangen wollen, und die Übersicht nicht gleich verloren gehen soll.

Vorher:

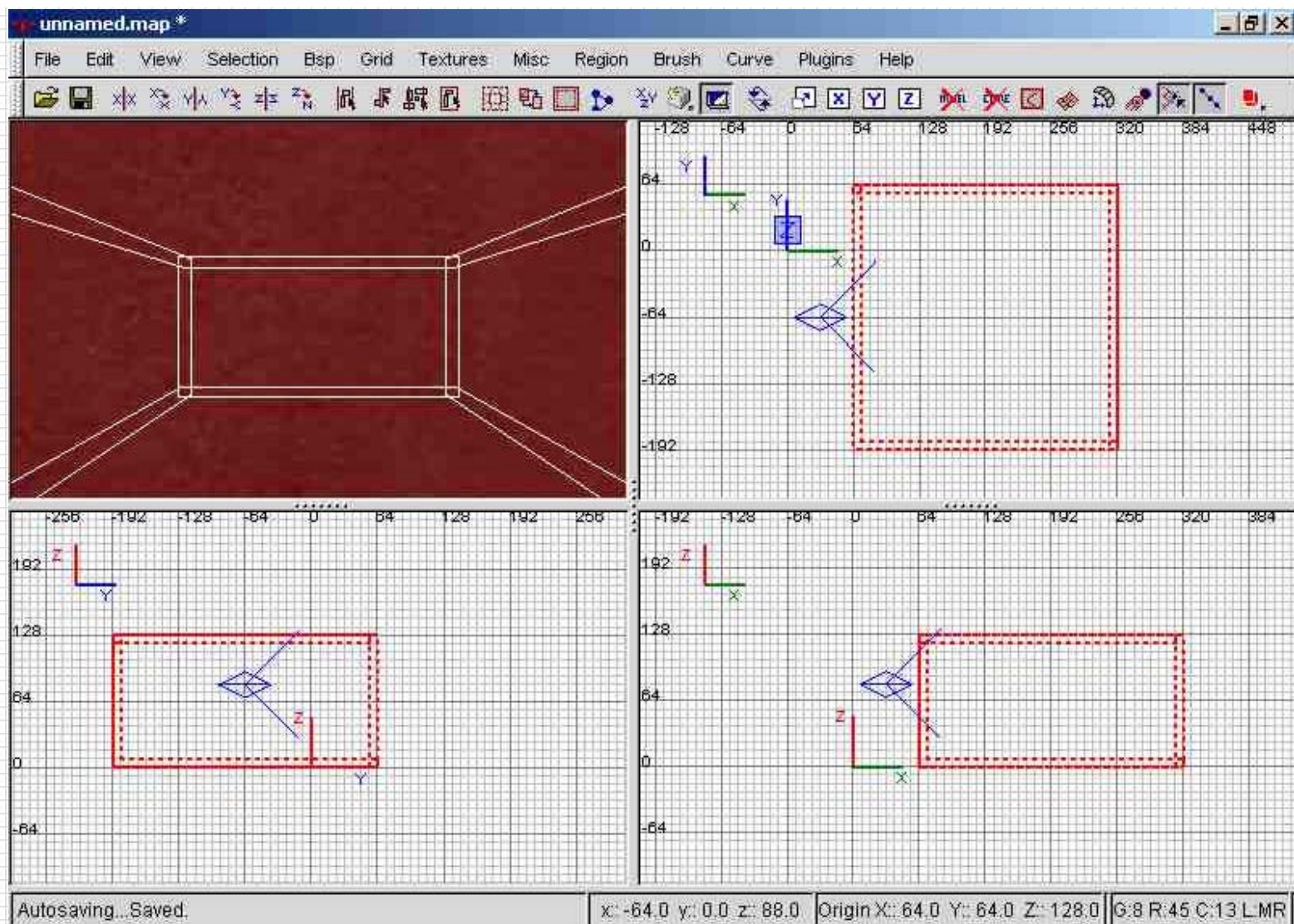
Nachher:



Nun hast du einen hohen Brush erstellt. Wie du nun siehst, ist auf dem "Nachher"-Bild unsere 3D Ansicht verschwunden, da die Kamera jetzt in unserem Brush sitzt und Brushes nunmal solide sind. Jetzt werden wir unseren Brush erstmal in einen Raum verwandeln, dazu klickst du auf das "Hollow"-Symbol unter dem Menü, dazu auch gleich ein Bild:



Das Ergebnis siehst du hier:



ACHTUNG: Mit dem "Hollow"-Symbol hohlen wir unseren Brush also aus.

Wenn du jetzt genau die Linien ansiehst, wirst du zwei Dinge feststellen:

- 1.) Der Raum ist kleiner geworden, und zwar genau an allen Seiten um jeweils 8 Units.
- 2.) Aus deinem Brush sind jetzt mehrere Brushes geworden.

Übrigens, der gesamte Raum ist jetzt selektiert, deshalb leuchten die roten Linien so auf. Jetzt deselektieren wir den gesamten Raum, dazu drücken wir die "ESC"-Taste.

WICHTIG: Mit der "Hollow"-Funktion einen Raum zu erstellen, ist eine Funktion, die man sehr selten, besser jedoch NIE verwenden sollte, da hier Überschneidungen bei den neuen Brushes erstellt werden. Ein richtiger Mapper erstellt seine Räume aus einzelnen Brushes und kann somit diese Überschneidungen verhindern. Ich habe diese Funktion nur benutzt, damit wir etwas schneller voran kommen und verwirren wollte ich euch auch nicht gleich so arg ;)

Dafür wisst ihr jetzt, weshalb man diese Funktion besser nicht verwendet. Jetzt fragst du dich sicher, was so schlimm an diesen Überschneidungen ist:

- Überschneidungen sorgen für Texturfehler im Spiel
- Überschneidungen brauchen bei der Compilierung (Berechnung der Map) deutlich mehr Zeit
- habt ihr zu viele Überschneidungen in der Map, kann es sein, dass sich der Radiant verabschiedet

Nun willst du sicher deinen Raum auch gleich spielen.

Jetzt brauchen wir einen Startpunkt für unseren Player. Dazu drücken wir zweimal (!!) die rechte Maustaste im Top Fenster. Es öffnet sich ein Menü. Daraus wählen wir "info/info_player_deathmatch". Es erscheint wieder ein rot gestricheltes Kästchen, das wir in jedem Fenster anklicken und verschieben können. Wir setzen unsere Playerposition irgendwo hin, sie darf aber nicht zu nahe an einer Wand sitzen (wer will schon beim Erscheinen in der Wand festsitzen) bzw. darf die Startposition auch nicht im Boden sitzen, danach drücken wir ESC und deselektieren hiermit unseren Startpunkt. Der weiße Pfeil zeigt dabei die Richtung, in die man beim Erscheinen schaut.

Spiel hiermit ein wenig, es ist einfach die Position zu ändern. Wenn du mit der Position deines Startpunktes nicht zufrieden ist, kann man das natürlich nachträglich ändern. Hierzu einfach die SHIFT-Taste drücken (und halten) und dann die Startposition mit der linken Maustaste anklicken. Nun ist das ganze wieder selektiert und man kann es verschieben. So einfach läßt sich jeder Brush nachträglich selektieren, was in der 3D Ansicht am allerbesten geht.

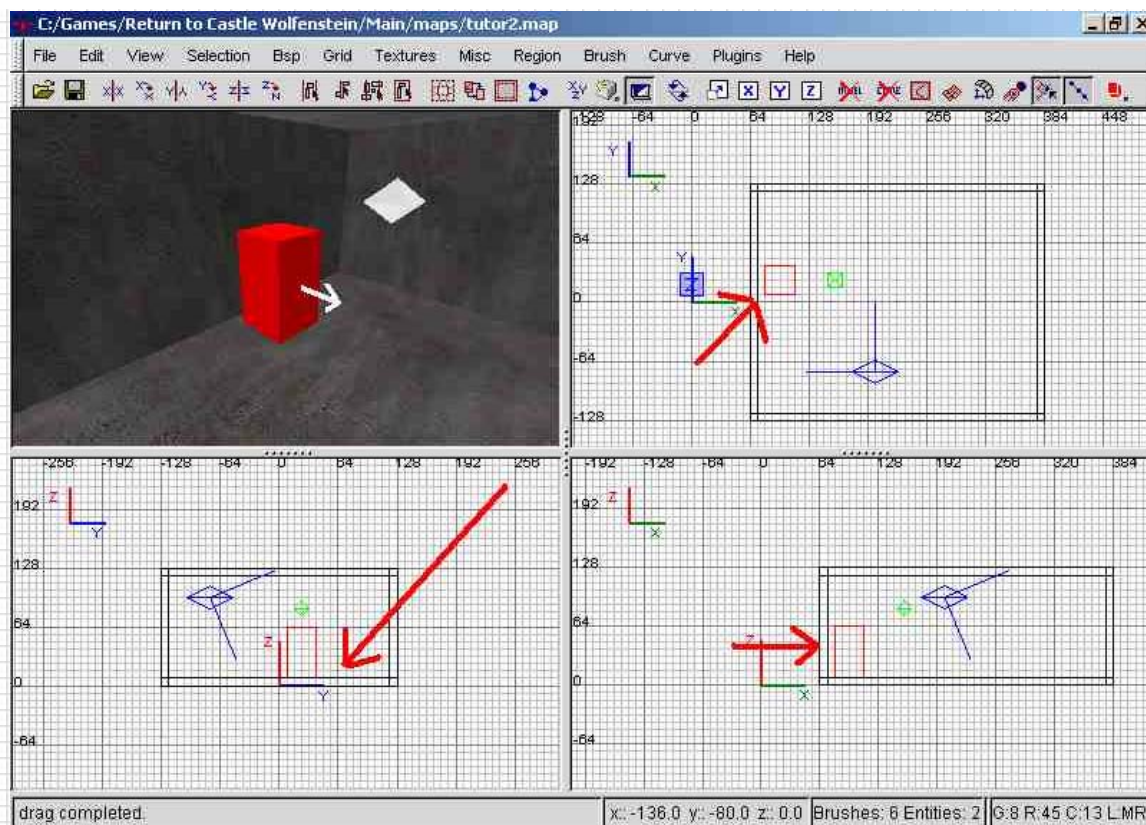
Wie du im Bild unten siehst, habe ich mittlerweile meine Kameraposition verändert. Das geschieht über den Cursorblock hoch, runter, links, rechts. Die Höhe änderst du über die Tasten "D" und "C".

Tastaturbefehle für das "Bewegen" im Radiant:

- Taste "A" : nach oben schauen
- Taste "Z" : nach unten schauen
- Taste "D" : nach oben bewegen
- Taste "C" : nach unten bewegen
- Taste ", " : Sidestep nach links
- Taste ". " : Sidestep nach rechts
- Taste "Einf" : aus dem Koordinatenkreuz herauszoomen
- Taste "Entf" : in das Koordinatensystem hineinzoomen

ACHTUNG: Klickst du mit rechts in die 3D Ansicht, so erscheint ein kleines Fadenkreuz. Nun kannst du mit der Maus nach oben und unten, sowie zur Seite schauen.

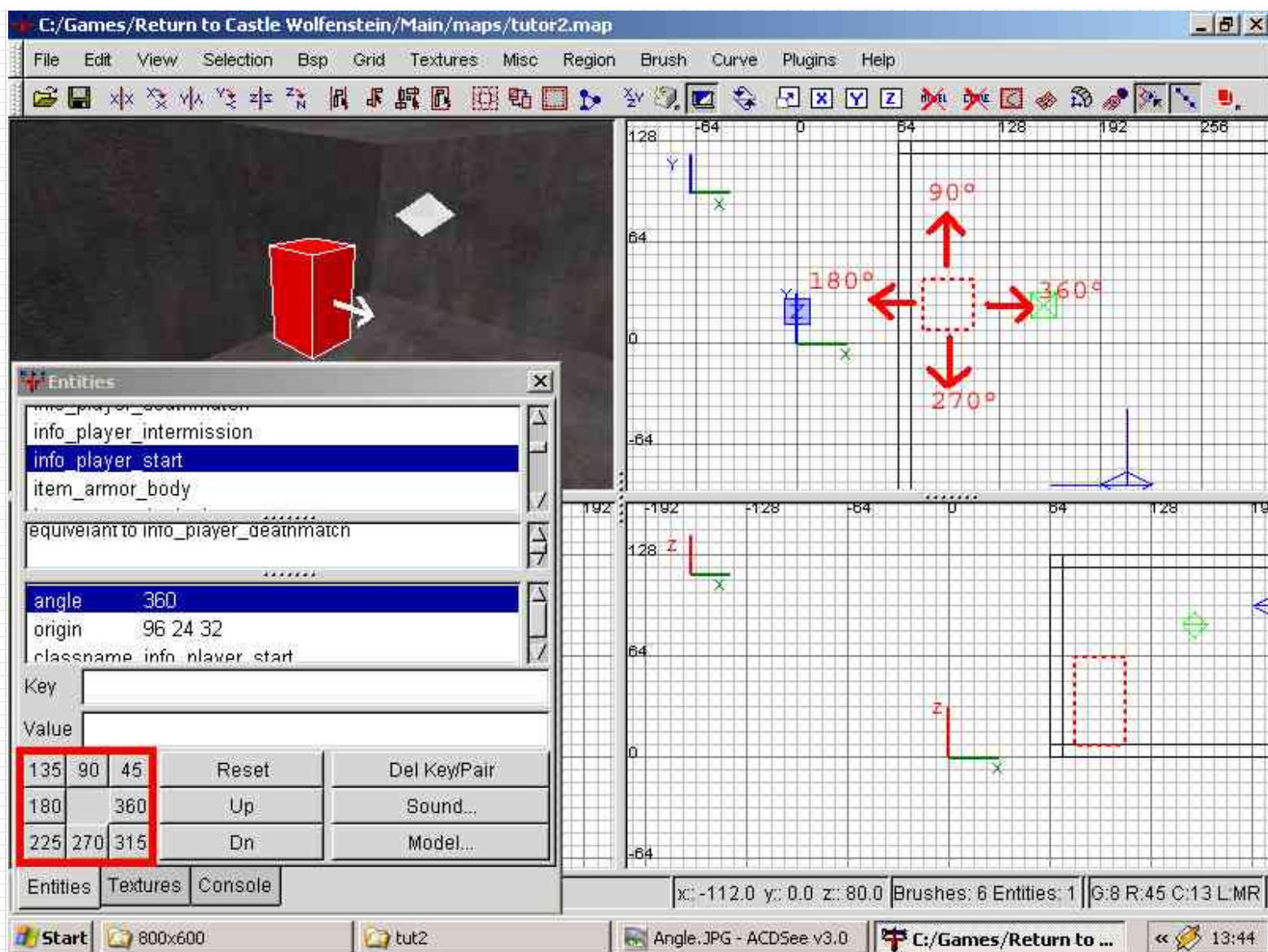
Um dieses Fadenkreuz wieder loszuwerden, musst du einfach wieder mit rechts in die 3D Ansicht klicken.



Du kannst natürlich auch in den 3 Fenstern (Draufsicht, der Seitenansicht und der Top Ansicht) umherschrollen. Dazu klickst du mit der rechten Maustaste in das entsprechende Fenster und bewegst deine Maus hin und her.

Nun bist du vielleicht mit der Richtung unzufrieden, in der dein Startpunkt zeigt (der weiße Pfeil deutet die Richtung an, in der später der Spieler beim Betreten der Map "sieht"). Um dies zu verändern, drückst du die Taste "N". In diesem Menü kannst du die Eigenschaften der Entities ändern. Wir wollen in diesem Fall die Angle ändern (die acht zahlen ganz links unten).

ACHTUNG: Um die Richtung eures Startpunktes zu ändern, muss dieser natürlich selektiert sein

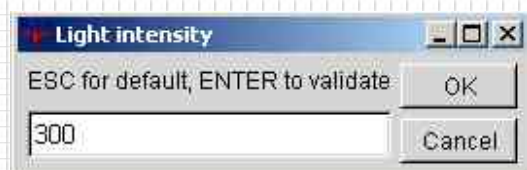


Hier ersteinmal eins vorweg: Diese Anordnung der 8 Zahlen stimmen mit der Topansicht genau überein. Drückst du z.B. den "360"-Button, zeigt nun die Vorderseite des Entities (in dem Fall dein Startpunkt) in der Top-Ansicht nach rechts. Drückst du den "180"-Button, zeigt die Vorderseite des Entities (in dem Fall dein Startpunkt) in der Top-Ansicht natürlich nach links. Drückst du auf den "90"-Button, zeigt die Vorderseite in der Top-Ansicht nach oben - drückst du den "270"-Button, so zeigt die Vorderseite in der Top-Ansicht nach unten.

Jetzt fehlt uns nur noch das Licht:

Doch bevor wir mit dem Licht anfangen, solltest du nochmal "ESC" drücken, um sicherzugehen, dass wirklich alles deselektiert ist. Dann klickst du im Top-Fenster zweimal (!!) mit der rechten Maustaste und wählst "light"

Es erscheint dieses Fenster:



Mit diesem Fenster kannst du die Lichtstärke festlegen. Um diese Zahl zu ändern (in diesem Fall die 300), klickst du in das weiße Kästchen. Dann kannst du die 300 löschen und einen anderen Wert eingeben. Nungut, 300 ist was zu hell, wir wollen den Wert 100. Also gibst du 100 ein. Damit leuchtet nun unser Licht nichtmehr so stark.

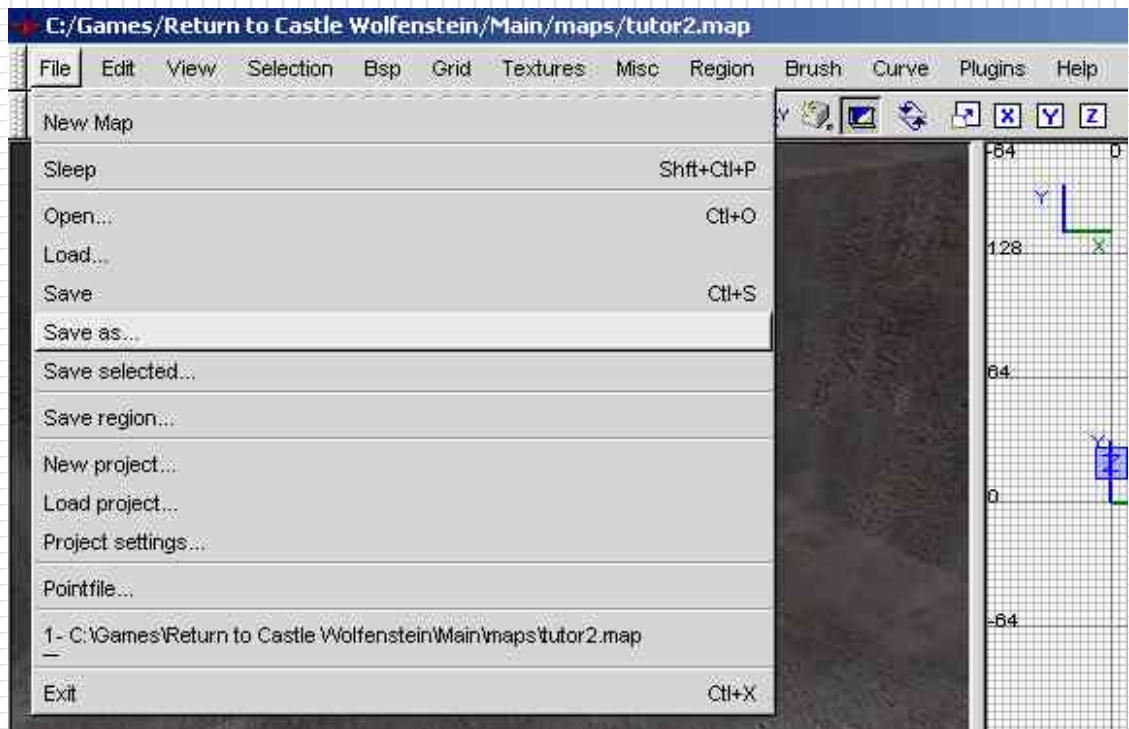
Raum mit Licht



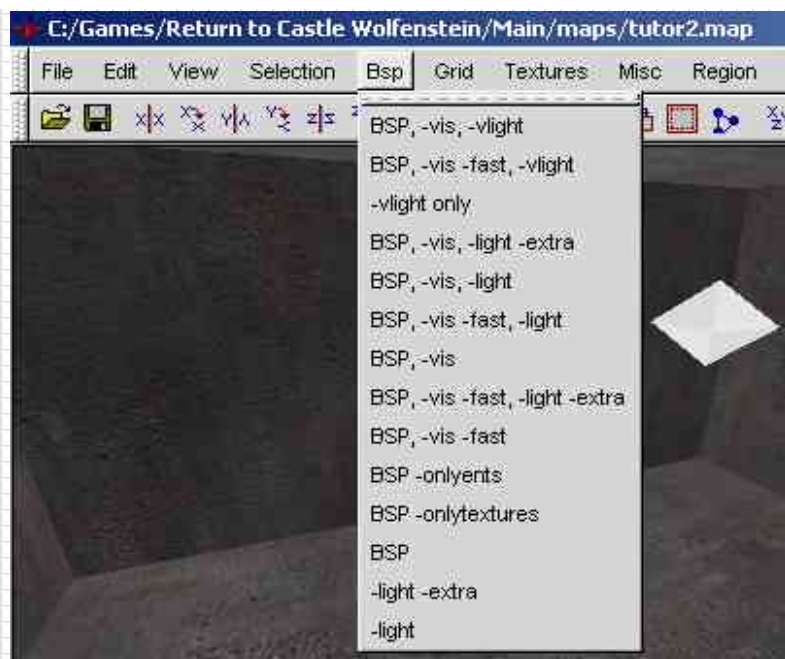
So könnte jetzt dein Raum aussehen. Natürlich wollen wir das Licht mehr an der Decke haben, dass es den Raum möglichst weit erhellen soll. Dazu drückst du "SHIFT" (und hältst die Taste gedrückt) und klickst das Licht an. Nun kannst du es nach oben schieben, dass es ungefähr so wie bei mir aussieht.

Jetzt drückst du nochmal "ESC" um alles zu deselektieren.

Jetzt speicherst du noch deinen Raum im Menü unter File/Save as... ab.: Wir nennen hier die Datei tutor2, damit wir sie auch wiederfinden.



Nun müssen wir noch deinen Raum in ein anderes Format bringen, dass dann unser Spiel auch verstehen kann.



Dazu wählen wir im Menü BSP/BSP. Nun sollte ein DOS-Fenster auftauchen - hier wird eure Map "compiliert", d.h. sie wird nun für das Spiel lesbar gemacht. Ist der Vorgang schliesslich beendet, könnt ihr in eurem Game-Verzeichniss, z.B. c:\RTCW\main\maps\ eure Map als euremap.bsp finden. Natürlich steht "euremap" für den Namen, unter der ihr eure Map abgespeichert habt, in unserem Fall "Tutor2.bsp"

ACHTUNG: Hier gibts es mehrere Möglichkeiten, die Map zu compilieren (zu berechnen), z.B. BSP und nur die Texturen, oder BSP und VIS...

Hast du nun etwas falsch gemacht, und deine Map wird nicht erstellt, musst du mal in mein F.A.Q. sehen, sicher findest du dort eine Lösung für dein Problem.

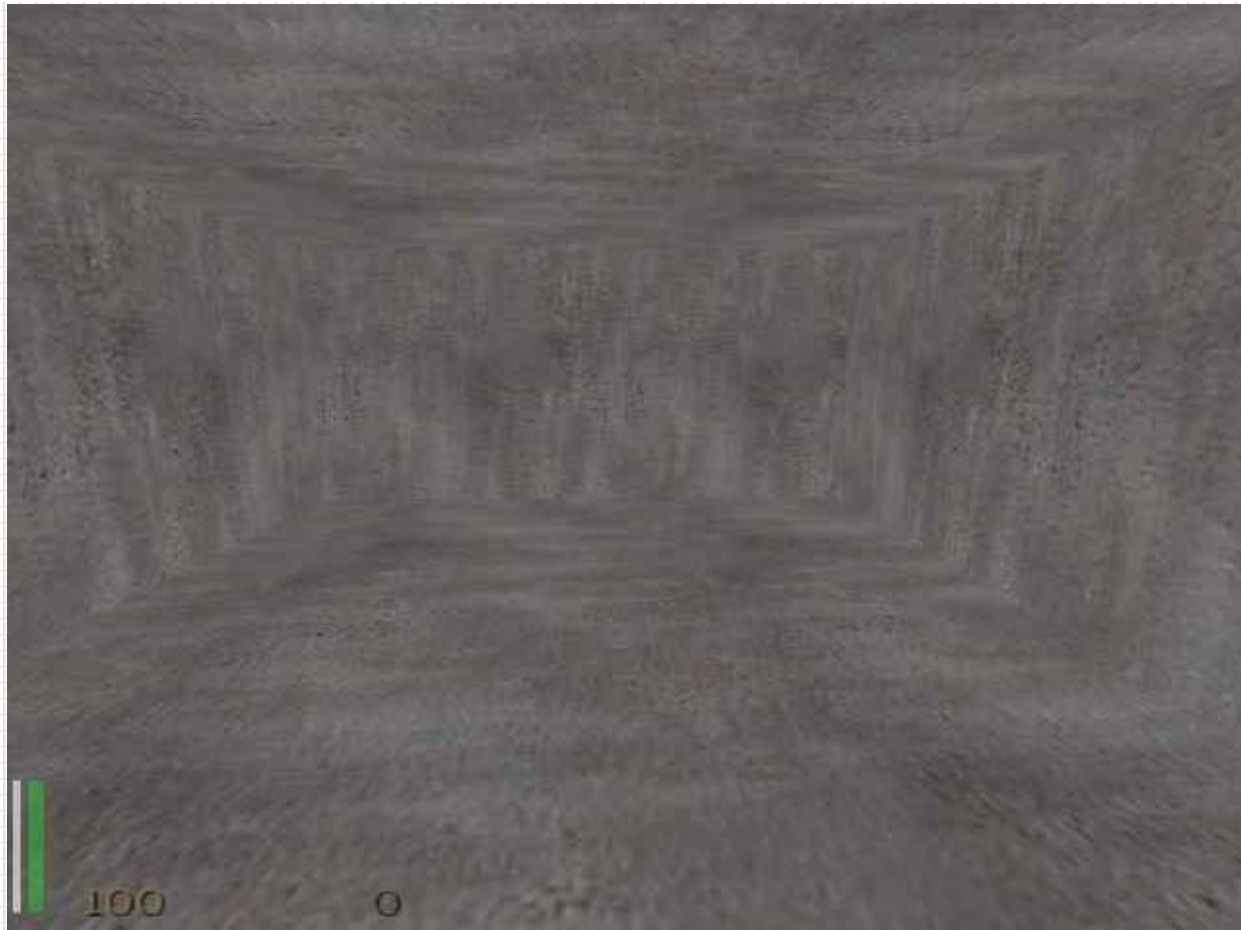
Nun kommt der große Moment, du startest dein Spiel und drückst ^ (für die Cosole) und gibst zuerst "/sv_pure 0" ein und nun "/map tutor2" (beide Befehle OHNE Anführungszeichen!!) ein. Es kann passieren, dass du vorher noch ein paarmal die BACKSPACE-Taste drücken musst, um falsche Zeichen wegzulöschen. Dann gibt man in der leeren Zeile zuerst "/sv_pure 0" und dann "/map tutor2" ein. Hier dürft ihr KEIN .bsp hinten dranhängen. Also, es ist hier etwas chaotisch rübergekommen, aber hier nochmal einzeln die Befehle für die Console:

1. "/sv_pure 0"
2. "/map namedeinermap" bzw. "/devmap namedeinermap"

Nun hat dein PC kurz etwas zu tun.. und...

Glückwunsch!

Wenn alles geklappt hat, kannst du nun deine ersten "vier Wände" bewundern.



[zurück zur Hauptseite](#)

183759



2 Räume verbinden:

Verwendete Map: tutor2.map
Ergebnis: tutor3.map

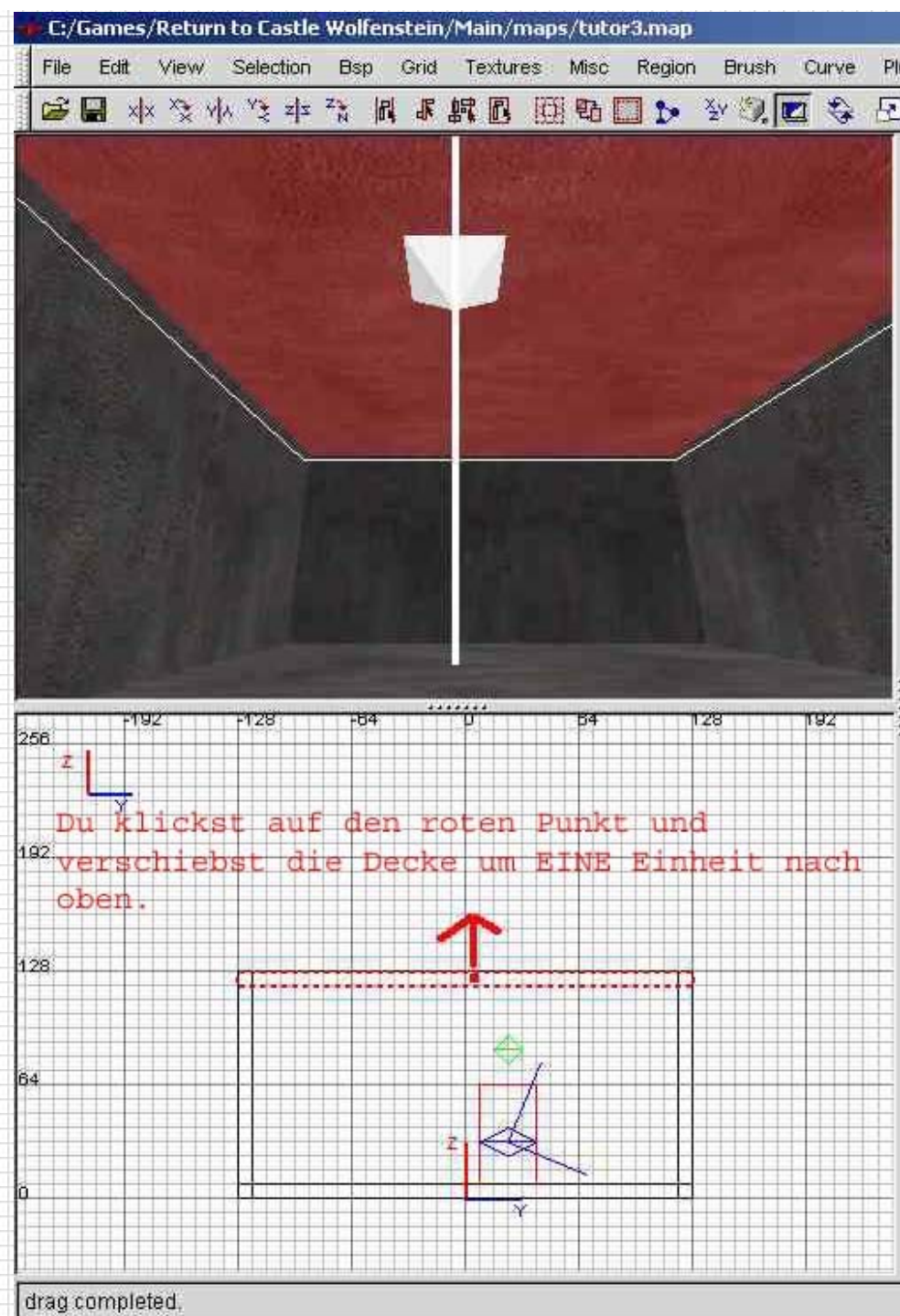
So, zunächst laden wir dazu die Map "tutor2.map", da wir den Raum, den wir im Kapitel vorher erstellt haben, durch einen weiteren Raum und einen Durchgang erweitern wollen

Wer das fertige Ergebnis anschauen möchte, kann auch die "tutor3.map" benutzen. Dies gilt auch für die folgenden Themen/Kapitel. Ihr werdet am Anfang eines Themas stets den Namen der "Ausgangs"-Map, sowie der Ergebnis-Map finden, so wie ganz oben bei diesem Thema. Willst du nun deine Map nach meiner Anleitung nachbauen, benutze einfach die "tutor2.map", also unsere Ausgangsmap.

Zum laden deiner Map benutzt du bitte den Menübefehl "File/Open". Dadurch wird alles, was du vorher im Radiant gemacht hast, GELÖSCHT, also durch die neue Map ÜBERSCHRIEBEN. Der Menübefehl "File/Load" lädt zu einer bestehenden Map noch eine weitere Map hinzu. Dies kann auch sinnvoll sein, aber wir benutzen jetzt die Funktion "File/Open"

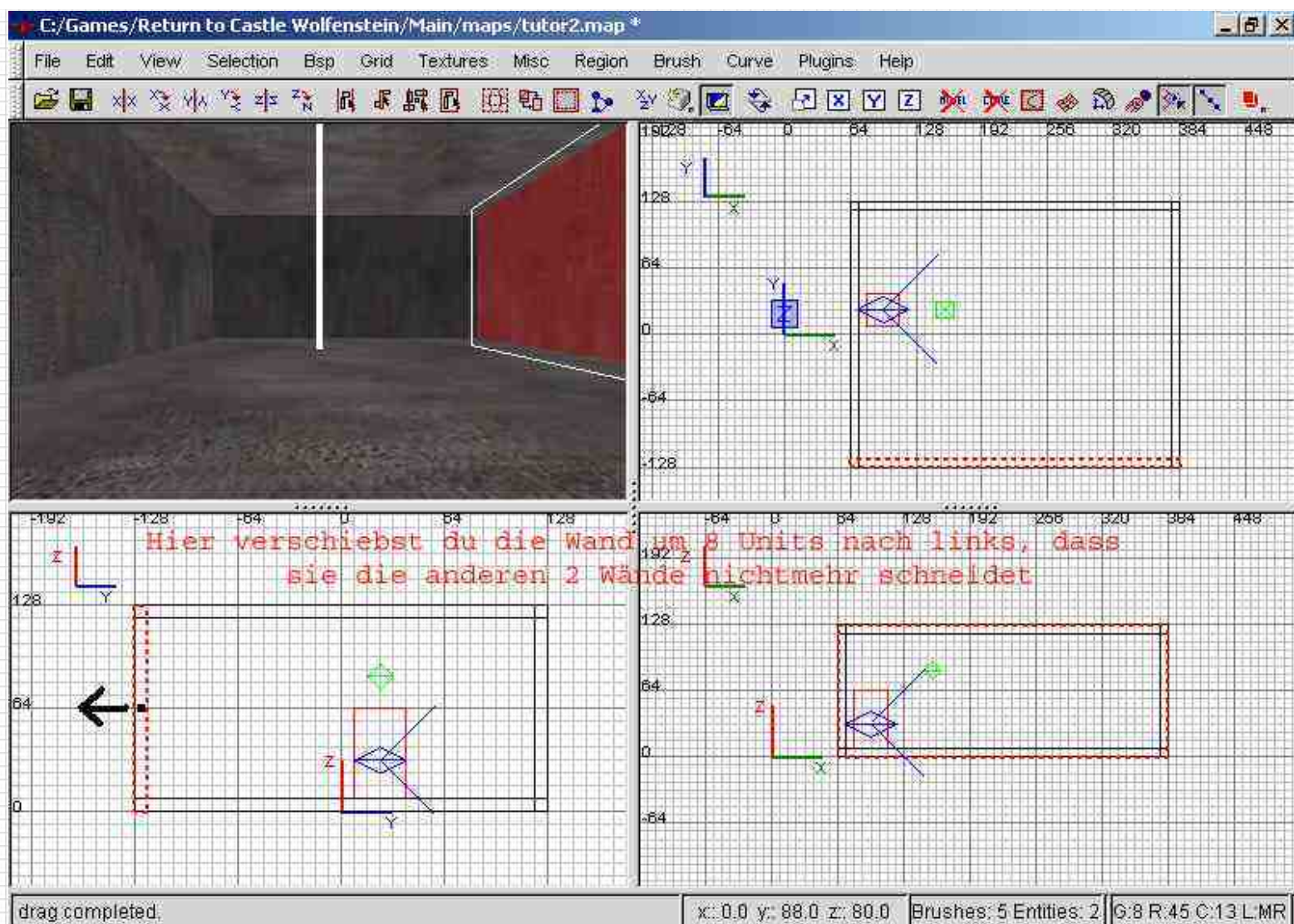
Legen wir los:

Da wir im letzten Kapitel gelernt haben, das Überschneidungen eigentlich nicht so wünschenswert sind, entfernen wir diese auch gleich. Selektiert dazu die Decke des Raumes und zieht diesen Brush um 8 Units nach oben. Dazu müsst ihr lediglich in die Mitte des Brushs klicken (linke Maustaste) und die Maus etwas nach oben bewegen. Klickt auf jeden Fall IN den roten Rahmen und nicht außerhalb. Klickt ihr außerhalb, dann verändert ihr die Größe des Brushs, klickt ihr innerhalb des Brushs, so könnt ihr ihn verschieben.



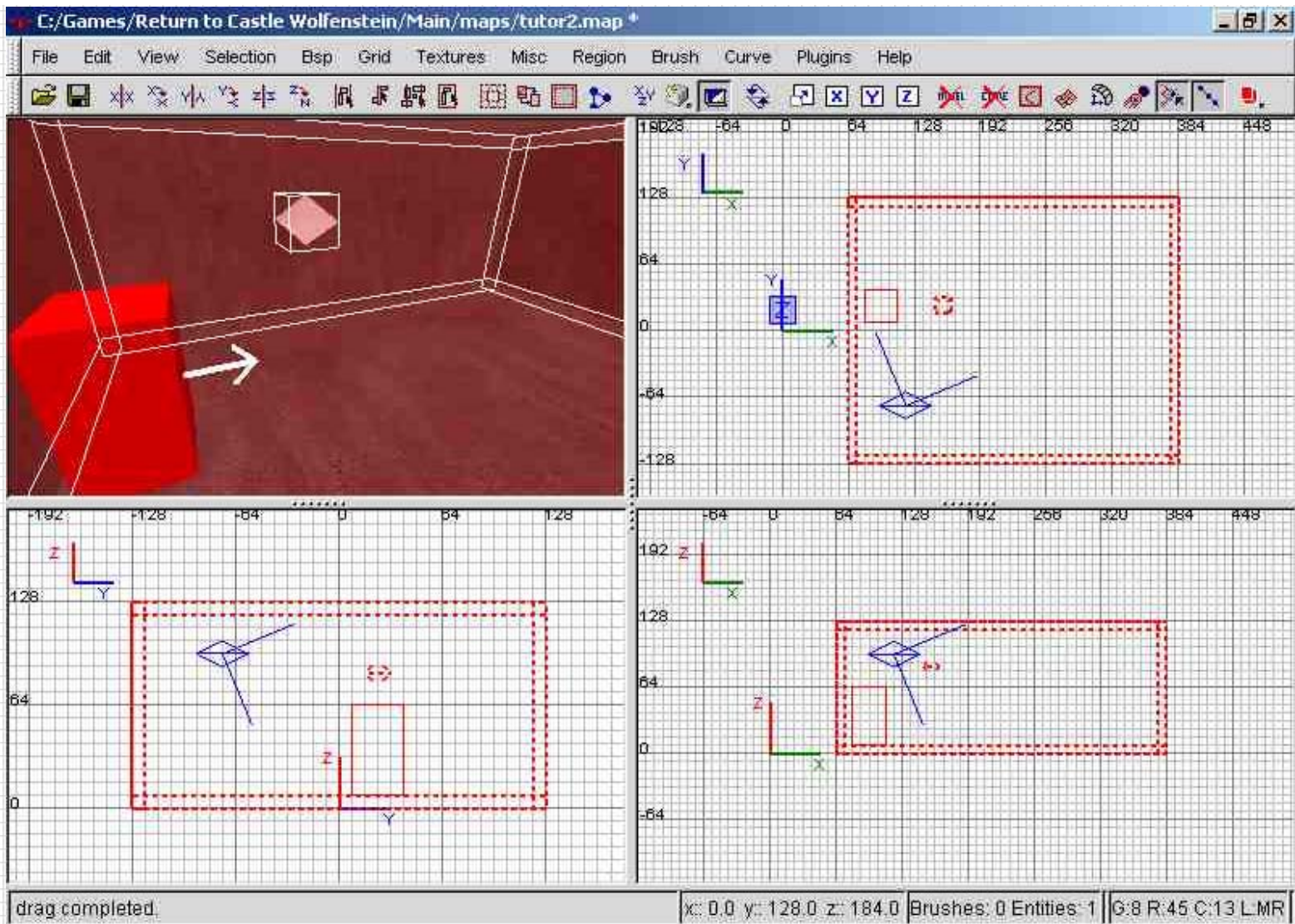
WICHTIG: Allerdings darf es zwischen der Decke und den Wänden keinen Spalt geben, da ihr sonst beim Compilieren ein "LEAK" bekommt. Eure Map MUSS zum Void abgeschlossen sein - es dürfen also keine Lücken vorhanden sein. Ihr dürft auch keine Items, Playerstarts usw. ausserhalb eurer Map bauen. Dies führt zu Fehlern, aber diese kann man ja im Vorraus verhindern.

Nun selektieren wir eine Wand und verschieben diese um 8 Units, so das sie die anderen 2 Wände nicht mehr berührt:

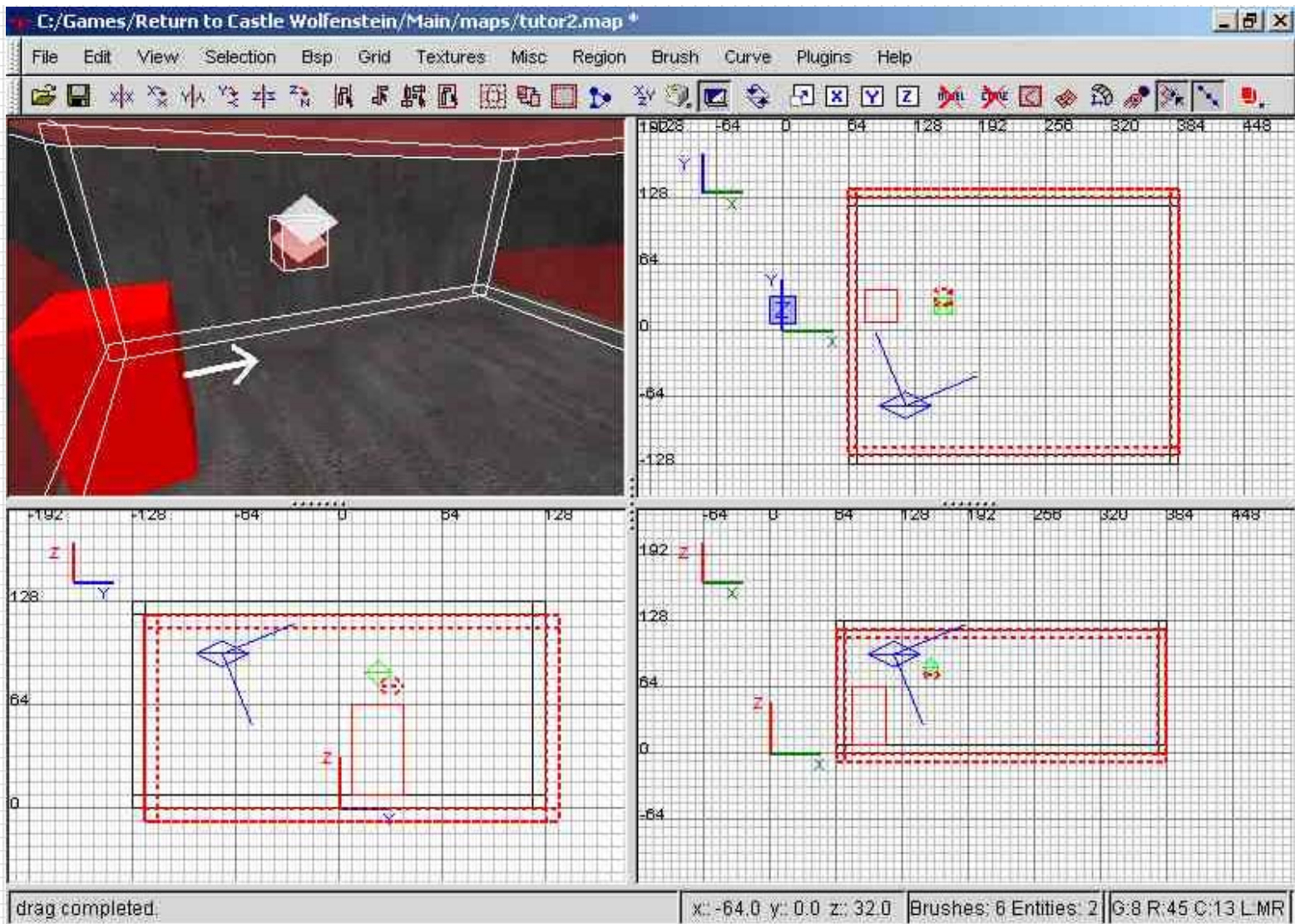


Das gleiche machst du jetzt auch mit den restlichen Wänden - du musst aber darauf achten, dass keine Lücken entstehen. Wenn ihr alles richtig gemacht habt, dann gibt es jetzt keine Lücken mehr und der Boden hat auch keinen Kontakt mehr zu den Wänden.

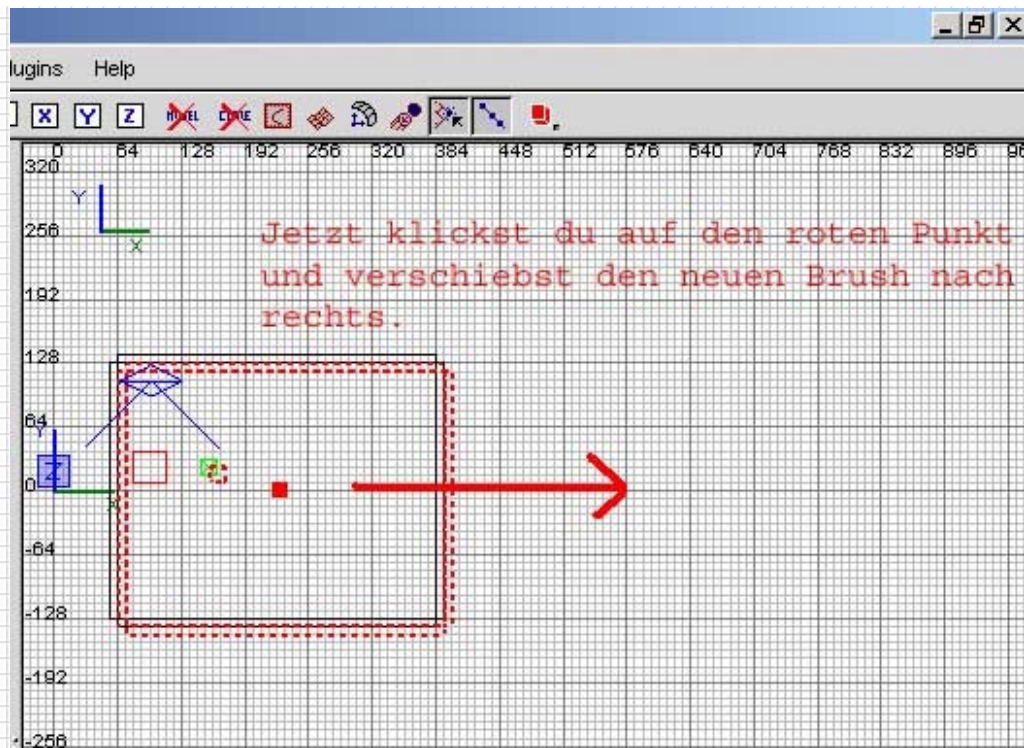
Nun müssen wir natürlich noch einen zweiten Raum neben den ersten bauen, damit wir später auch einen Durchgang bauen können - er soll ja nicht in den Void führen. Wir machen uns aber diese Arbeit leicht. Dabei lernst du auch gleich, wie man mehrere Brushes duplizieren (verdoppeln) kann. Dazu selektierst du alle 4 Wände + die Decke + den Boden + das Licht (mit "SHIFT" + linker Maustaste nacheinander anklicken). Damit das gut klappt, solltet ihr zwischendurch die Kameraposition verändern.



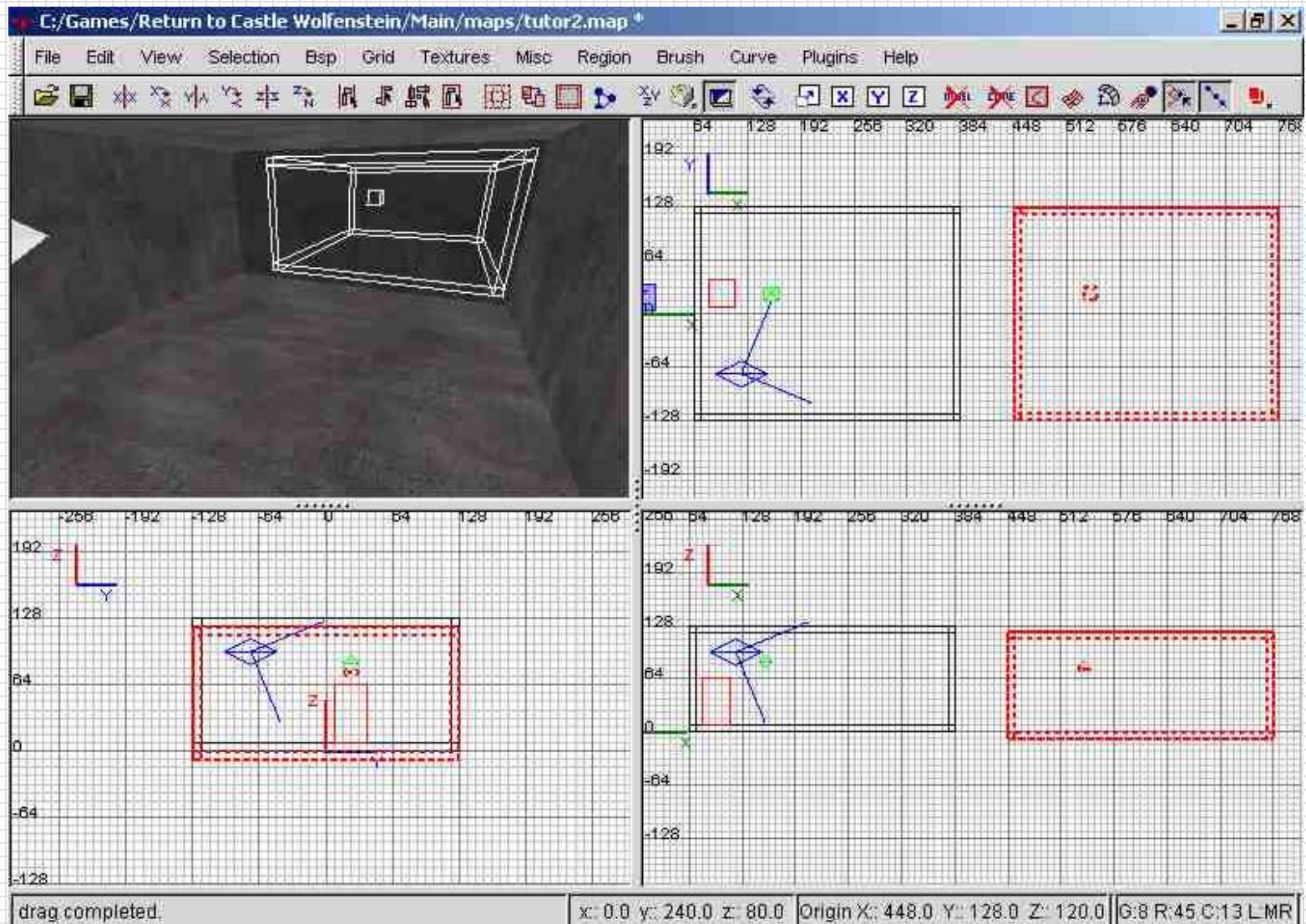
Jetzt drückst du die "SPACE"-Taste (Leertaste). Damit kopierst du alle selektierten Brushes. Dieser neue Raum wird etwas versetzt dargestellt, dass man auch merkt, dass man ihn dupliziert hat. Der neue Raum beinhaltet auch eine Kopie deines Lichts, da wir es ja mitkopiert haben. Ich habe hier zur besseren Ansicht das Top-Fenster schon etwas herausgezoomt.



Jetzt musst du wahrscheinlich deine Fenster zoomen (ich benutze nur das Top-Fenster) und dann setzt du den neuen Raum mit einem kleinen Abstand neben den ersten Raum.



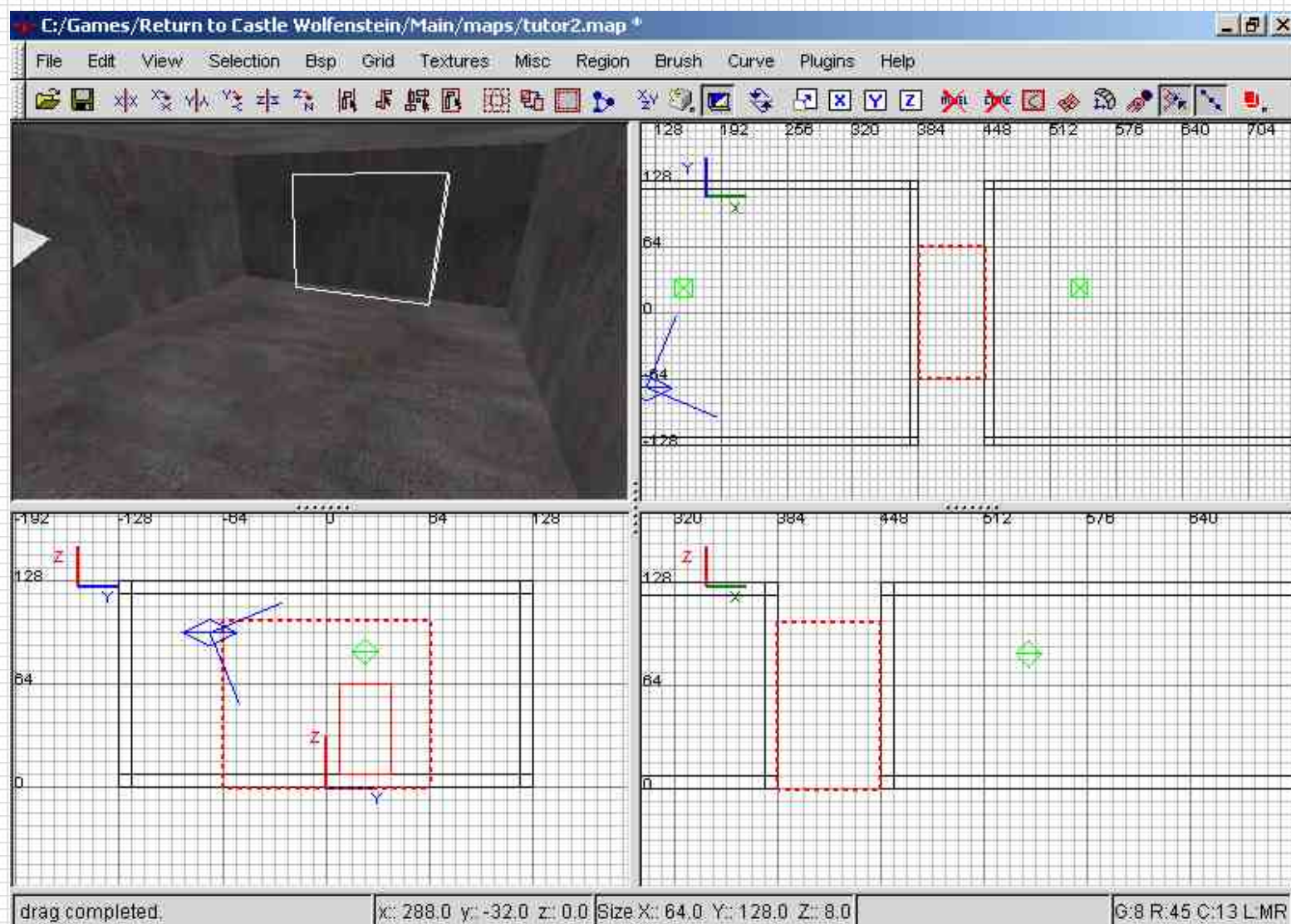
Das Ergebnis könnte ungefähr so aussehen:



WICHTIG: Wenn du einen Fehler gemacht hast oder mit deinem Brush nicht zufrieden bist, kannst du diesen (oder auch eine ganze Gruppe von Brushes) mit der "BACKSPACE"-Taste (über ENTER) löschen. Davor solltest du aber darauf achten, dass du nur die Brushes selektiert hast, die du löschen willst. Also solltest du vielleicht lieber erst alle mit "ESC" deselektieren und dann NUR die Brushes selektieren, die du löschen willst. Zudem kannst du die "UNDO"-Funktion benutzen. Damit kannst du deinen letzten Arbeitsschritt rückgängig machen. Diese Funktion findest du unter "EDIT/UNDO"

Ausserdem ist es sehr wichtig, dass ihr regelmäßige Sicherheitskopien von eurer Map erstellt. Dafür erstellt ihr am besten einen eigenen Ordner, in der ihr eure aktuelle Map abspeichert. Nummeriert diese vielleicht durch (test01.map, test02.map usw.) oder benutzt die Uhrzeit (1200.map, 1230.map usw.). Dazu solltest du deine Map alle 30 min abspeichern. Schmeißt euch jetzt der Radiant oder euer Betriebssystem ab, bist du nicht völlig aufgeschmissen. Diese Dateien nehmen nicht viel Speicherplatz weg, also solltest du dir diese Arbeit UNBEDINGT machen, da du sonst böse Überraschungen erleben kannst.

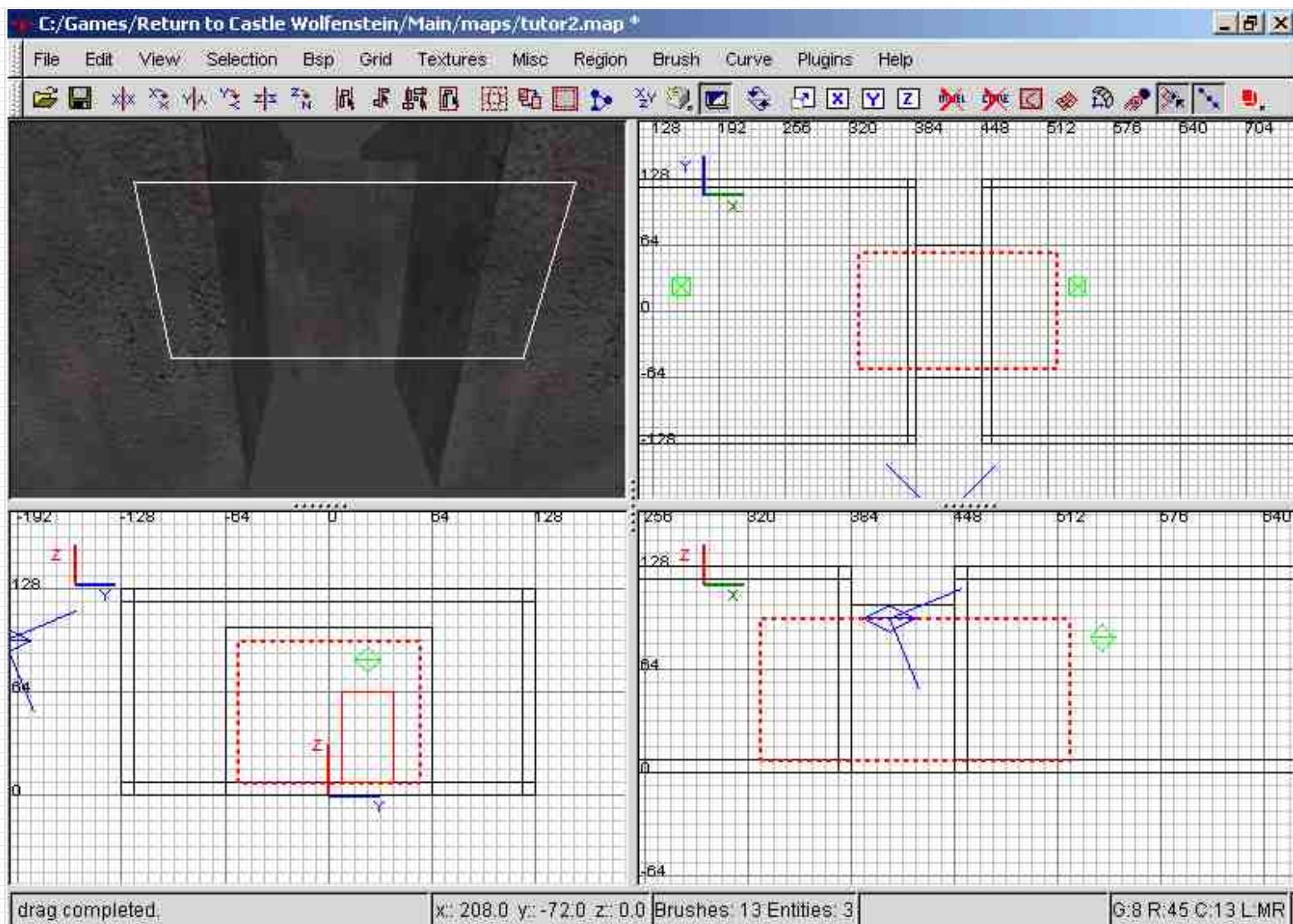
Nun setzen wir zwischen unsere 2 Räume einen Brush, der dann später den Durchgang zwischen den Räumen darstellen soll. Schaut euch dazu folgendes Bild an:



Jetzt kopierst du diesen Brush mit der "SPACE"-Taste (oder du erstellst ihn zur Übung selbst). Dieser neue Brush benutzen wir dazu, ein Loch zwischen den beiden Räumen und dem Durchgangsbrush zu erzeugen. Dazu musst du den neuen Brush aber noch anpassen, also die Größe verändern.

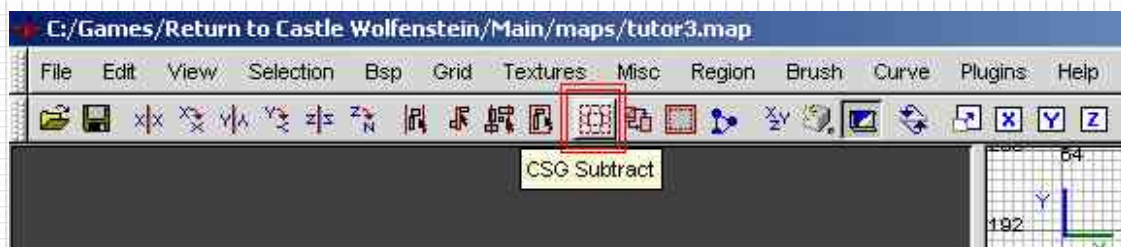
Der neue Brush sollte in beide Räume mit mindestens 8 Units hineinragen, aber in der Breite und in der Höhe um jeweils 8 Units kleiner sein.

Um euch das besser vorzustellen, hier ein Bild:

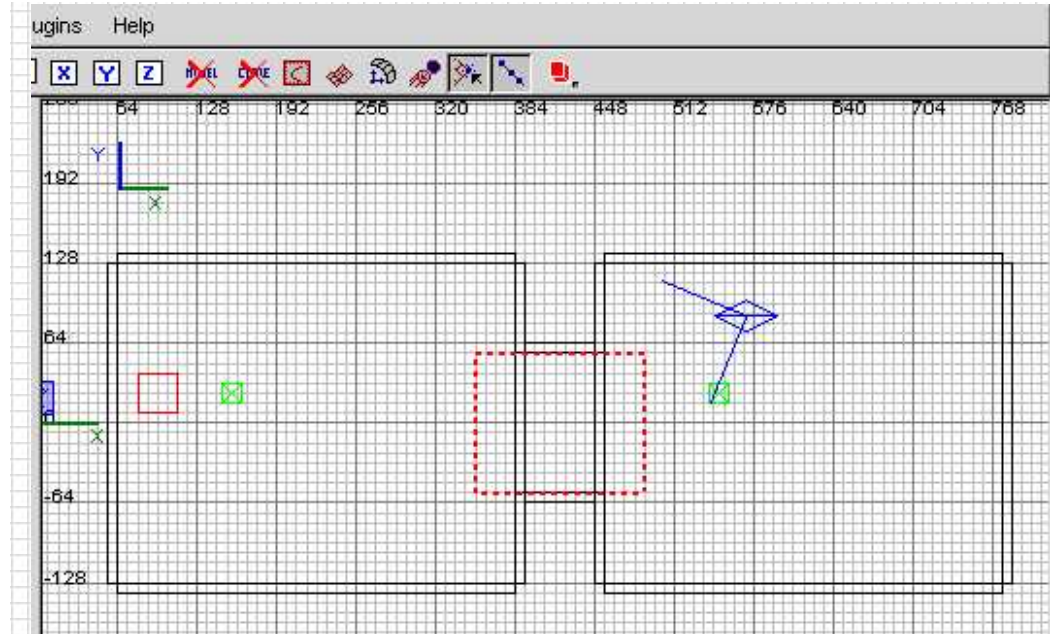


Also, noch einmal schnell zur Erklärung. Unser Schneidebrush (also der selektierte) ist etwas kleiner und schmaler, als der Durchgangsbush, aber er ragt noch in die beiden anderen Räume hinein. Das hat den Sinn, das wir nach dem Schneiden einen Durchgang haben, der auch einen Boden und eine Decke hat. Wäre der Schneidebrush und der Durchgangsbush gleich hoch, hätten wir ja keine Decke und auch keinen Boden.

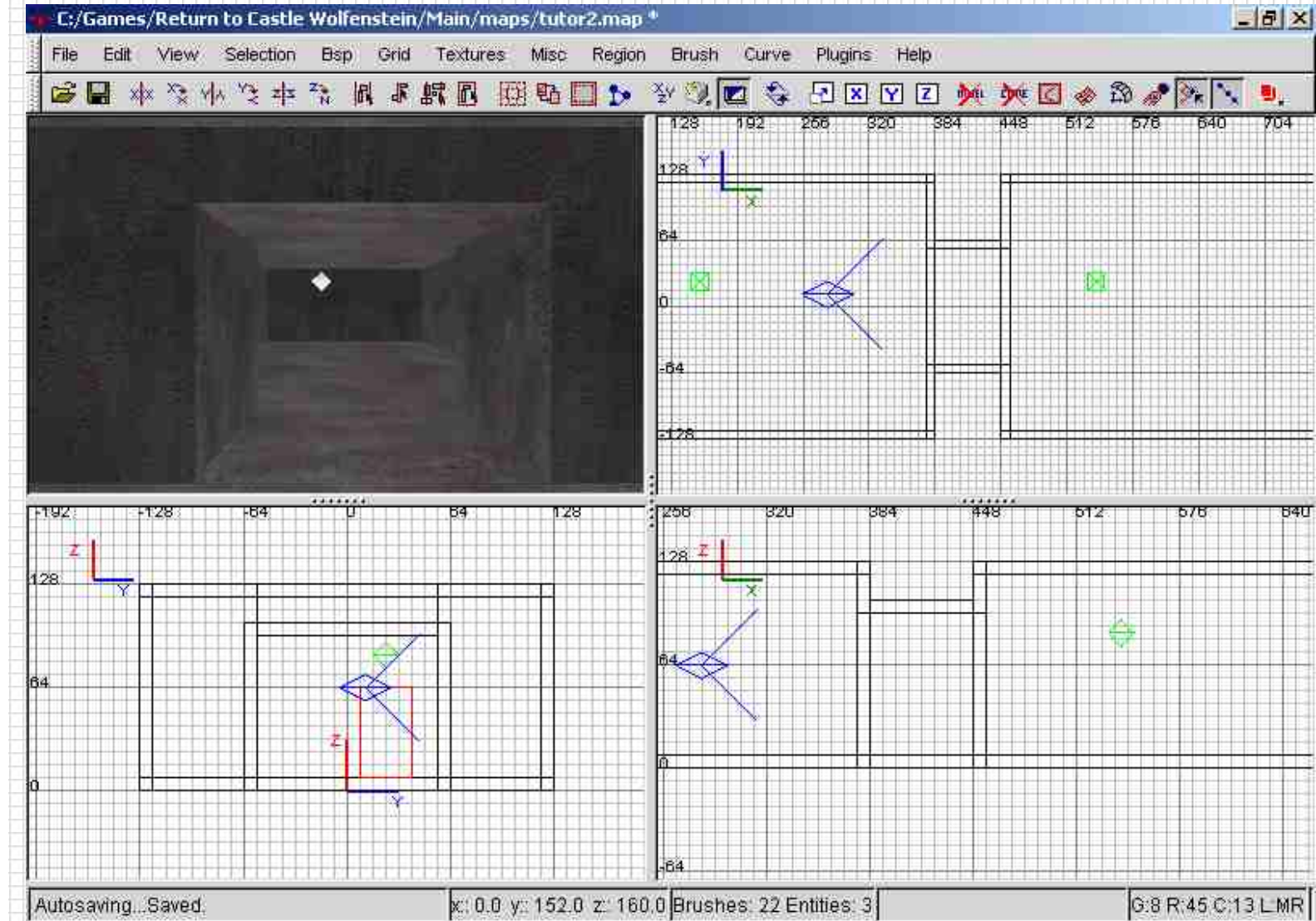
Nun druckst du auf "CSG Subtract" in der Icon-Leiste:



Dann sieht es so aus (keine Panik, es ist normal, das es aussieht, wie vorher)



Nun löschst du allerdings diesen "Schneidebrush" - du brauchst ihn ja nichtmehr - und siehe da, ein schöner Durchgang ist entstanden: Sicher siehst du das Ergebnis nicht so gut, wie hier auf dem Bild, dazu solltest du dann deine Kameraperspektive



Hier muss ich allerdings gleich wieder etwas sagen:

Achtung:

Ein fortgeschrittener Mapper nutzt die CSG Subtract Funktion eigentlich NICHT, da damit nur unnötige und zusätzliche Brushes entstehen, die der Performance unserer Map verschlechtern. Aber als Anfänger brauchst du dir da keine Gedanken machen - du willst ja erst einmal alle Funktionen kennen lernen und dich damit vertraut machen. Für solche einfachen Dinge kann man die CSG Subtract Funktion benutzen. Aber man sollte aus Performance-Gründen die Map Brush für Brush erbauen.

Nun sitzen wir auf diesen zwei Räumen - so toll sieht es ja eigentlich doch nicht aus. Deshalb werden wir die langweilige Decke löschen und einen Himmel einsetzen. Ein Himmel ist in RtCW und in Q3 animiert, wie man unschwer erkennen kann. Diese Animationen werden über Shader-Dateien gesteuert. Diese befinden sich im Verzeichnis "C:\RtCWMain\Scripts\" oder "C:\Q3\baseq3\scripts\". Es sind ganz normale Textdateien, dir ihr euch mit Notepad anschauen könnt. Die Textdateien steuern alles: Durchlässigkeit, Animationen, Flüssigkeiten, Helligkeit... usw..

ABER: Nur Texturen mit weißem Rand haben einen solchen Shader, andere Texturen sind fest, also Strukturen

Dazu kommen wir aber noch später, nun klickst du deine Decken in BEIDEN Räumen an (im 3D Fenster mit der Maus draufzeigen, "SHIFT" + linke Maustaste drücken). Dann wählst du bei den Texturen "skies/" aus und wartest, bis die Texturen geladen sind. Dann wählst du die Textur "skies/mx_assaultsky" aus. Jetzt drückst du ersteinmal "ESC", damit du alles deselektierst.

Und hier das Ergebnis:



So, da wir nun unsere Map mit einem Himmel ausgestattet haben, benötigen wir ja nun die 2 Lichtquellen nichtmehr (Shaders steuern auch die Helligkeit von Texturen, in dem Fall auch vom Himmel). Dieser leuchtet nun unsere beiden Räume aus. Also wählst du nun die Lichtquellen an, und drückst "BACKSPACE" um die Lichtquellen zu löschen.

Nun wollen wir noch den Pepp in die Map bringen - also vielleicht ein paar Waffen und Items. Dazu klickst du wieder im Top Fenster zweimal mit der rechten Maustaste - und wählst im Menü "weapon" und suchst dir eine Waffe aus (ich hab mich für den Flammenwerfer entschieden). Nun musst du natürlich dieses Item noch positionieren und "ESC" drücken. Du musst jedoch darauf achten, dass du diese Waffe nicht in den Boden setzt, da dann die Waffe nicht im Spiel auftaucht. Ich habe die Beispiemap noch mit einem Health-Pack (im Top Fenster 2x rechte Maustaste klicken, "item_health_small" wählen), ein paar Fuel-Pakete (im Top Fenster 2x rechte Maustaste klicken, "ammo_fuel" wählen) und einen zweiten Player-Startpunkt gesetzt (im Top Fenster 2x rechte Maustaste klicken, "info_player_start" wählen)

[zurück zur Hauptseite](#)

<http://www.haradirki.de>



Texturen:

verwendete Map: "tutor4.map"

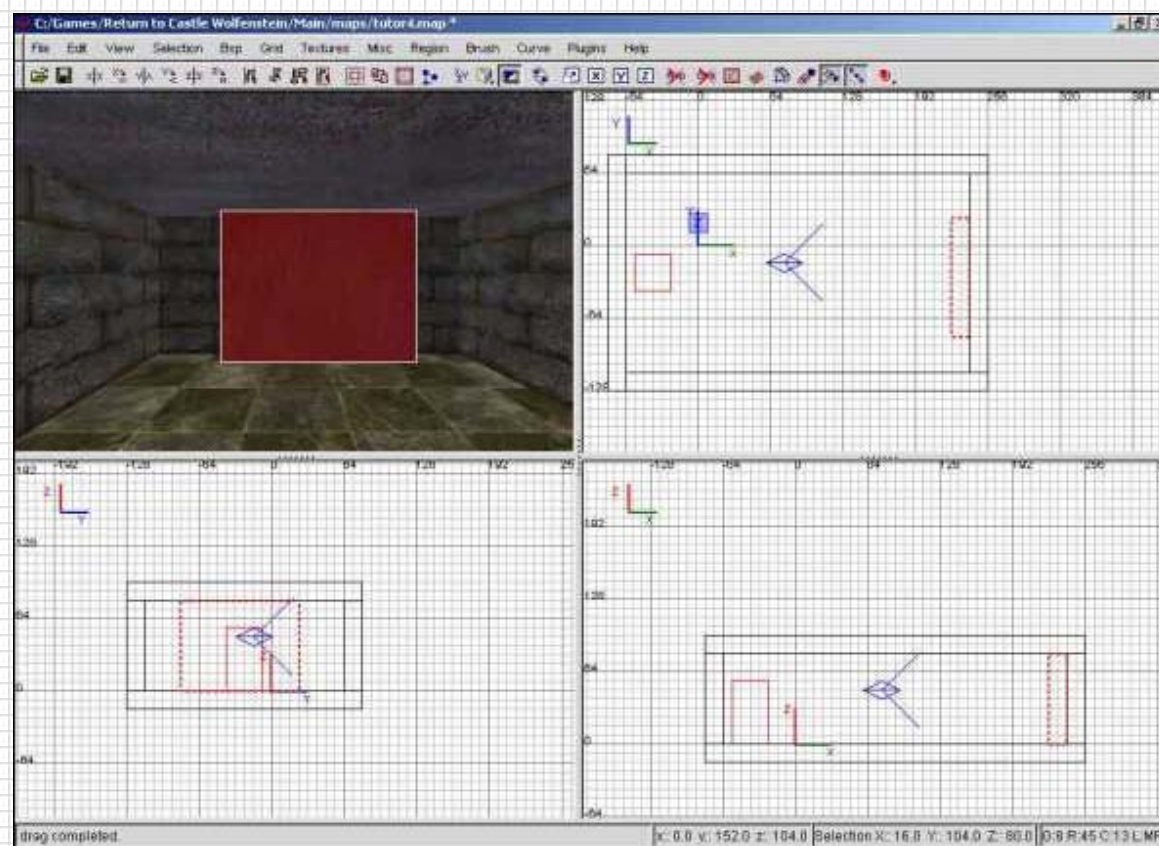
Nun hast du ja eigentlich schon ziemlich alles drauf, um eine recht einfach Map herzustellen. Sicher ist dir vielleicht in der Zwischenzeit auch ein paar Ideen gekommen, die Texturen im GtkRadiant bzw. im Q3Radiant zu nutzen.

Nur noch einmal schnell zur Wiederholung : Klick deinen Brush an, wähle im Menü "Textures" und dort die Art von Textur, die du haben willst, z.B. "village" oder "assault" (gibt es nur bei RtCW) oder aber "gothic/block" oder "base/trim" (gibt es nur in Q3A). Dann wartest du, bis die Texturen geladen sind, und drückst "T" - in dem erscheinenden Texturen-Fenster kannst du dann eine Textur auswählen. Damit belegst du aber den gesamten Brush mit derselben Textur.

Willst du aber hingegen die Textur nur auf einer Seite haben, drückst du in der 3D Ansicht "STRG" + "SHIFT" + linke Maustaste, bei der Fläche, die später deine Textur tragen soll.

So, übertreiben wollen wir mal nicht, und lassen es ruhig angehen:

Ich habe für dieses Thema einen eigenen Raum gebaut, der Boden in dieser Map besteht aus der Textur "chateau/marblefloor_c04", die Wände aus der Textur "crypt/crypt_column_02" und die Decke aus der Textur "assault/awall_m04". Im Grunde ist es egal, was für Texturen du dafür benutzt, es soll ja auch nur als Beispiel dienen.



Dem ganzen neuen Brush habe ich dabei die Textur "assault/awall_m04" zugeordnet. Danach habe ich die Vorderseite unseres "Wandbildes" mit der Kombination STRG + SHIFT + linke Maustaste selektiert:

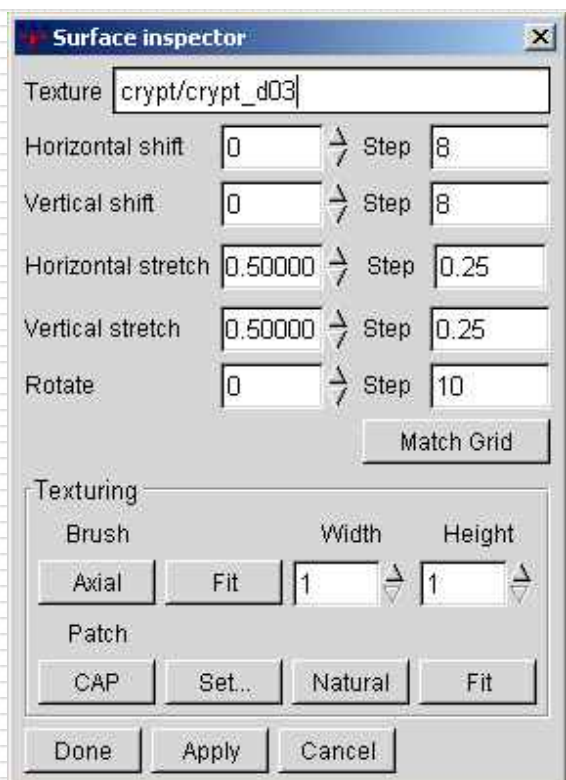


Auf diese Vorderseite werden wir die Textur "crypt/crypt_d03" legen (ein schwarzes Männchen). Dazu gehst du im Menü auf "Textures", und da auf "crypt". Dann drückst du "T" und wählst dir die Textur "crypt/crypt_d03":



Nun, im Grunde sieht das ja nicht gerade berauschend aus, da die Textur doch im Textur-Fenster viel besser aussieht. Dafür gibt es eine sehr nützliche Funktion. Dazu muss die Oberfläche des Brushes nach wie vor selektiert sein. Jetzt drückst du einfach die Taste "S". Es erscheint der "Surface Inspector":

Oberstes Feld "Texture":
Hier siehst du den Namen deiner aktuellen Textur, die



auf einem Brush (bei uns natürlich nur auf der Oberfläche) liegt. Du kannst hier auch "von Hand" eine andere Textur eintippen, und mit "ENTER" bestätigen. Diese neue Textur wird automatisch übernommen und dargestellt.

Horizontal shift/Vertical shift:

Hiermit verschiebst du die Textur, um sie auf den Pixel genau dem Brush (in unserem Fall auf der Oberfläche) anzugleichen.

Horizontal stretch/Vertical stretch:

Damit kannst du eine Textur strecken oder stauchen.

Rotate:

Hier kannst du die Textur im beliebigen Winkel auf deinem Brush (in unserem Fall natürlich die Oberfläche) drehen.

Weiter unten im Feld "Texturing" sehen wir:

Brush Axial:

Hier kannst du eine Textur axial ausrichten

Brush Fit:

eine Textur auf der gesamten Fläche in jede Richtung genau ausrichten

Width/Height:

sehr nützlich, um eine Textur zb. auf einer Wand in einer bestimmten Teilung x-mal darzustellen, sowohl in der Höhe, als auch in der Breite

Mit dem Abschnitt "Patch" beschäftigen wir uns in einem anderen Thema.

Momentan ist der wichtigste Button für uns der "FIT" Button unter "Texturing" sein. Damit setzt sich die Textur automatisch so auf eine Fläche, dass die Textur mittig auf der Oberfläche sitzt. Probier es einfach mal aus, du wirst sehen, unser schwarzes Männchen passt danach prima auf unser "Wandbild"

ACHTUNG: Wenn die Textur spiegelverkehrt ist, solltet ihr in den Feldern Horizontal stretch/Vertical stretch einfach das Minus (-) entfernen, dadurch wird die Texturrichtung umgekehrt bzw. solltet ihr dann ein Minus (-) hinzufügen (je nach dem).

Doch jetzt schauen wir uns mal das Ergebnis an:



So, das schwarze Männchen passt perfekt auf die Oberfläche

Manchmal kann es passieren, dass du die Textur noch "von Hand" ausrichten musst, dazu benutzt du dann "Horizontal shift/Vertical shift". Damit könnt ihr die Textur Pixelgenau verschieben. Und wenn du eine bestimmte Textur auf einem großen Brush x-mal darstellen willst, benutzt du "Width/Height" (danach müsst ihr den "FIT"-Button drücken)

Natürlich lässt sich das ganze auch über die Tastatur steuern:

- SHIFT + Cursor hoch/runter : Textur pixelweise hoch/runter verschieben
- SHIFT + Cursor links/rechts : Textur pixelweise links/rechts verschieben
- STRG + Cursor hoch/runter : Textur hoch/runter stretchen
- STRG + Cursor links/rechts : Textur links/rechts stretchen

Hier noch etwas wichtiges:

Eine Textur schneller von einem anderen Brush kopieren:

Für einen Brush:

Du selektierst den Brush ("SHIFT"-Taste + linke Maustaste). Dann zeigst du mit der Maus auf eine beliebige Textur im 3D Fenster und rückst die mittlere Maustaste. Natürlich kannst du auch auf mehrere Brushes gleichzeitig eine neue Textur kopieren. Hast du nur eine 2-Tasten-Maus, benutz für das Kopieren der Textur die rechte Maustaste.

Für eine einzelne Fläche:

Du selektierst die Fläche, (STRG + SHIFT + linke Maustaste). Nun zeigst du mit der Maus auf eine beliebige Textur im 3D Fenster und drückst die mittlere Maustaste. Du kannst aber auch die Texturfläche, die man weiter kopieren will, selektieren ("STRG" + "SHIFT" + linke Maustaste). Nun klickst du alle Flächen an, die diese Textur erhalten sollen, mit "STRG" und der mittleren Maustaste an. Hast du nur eine 2-Tasten-Maus, benutz für das Kopieren der Textur die rechte Maustaste.

Nun gibt es noch eine "Besonderheit", die ich nicht vorenthalten will. Wenn du einen Brush mit einer Textur verziert hast und diese ganz genau ausgerichtet hast, kann es beim Kopieren des Brushes vorkommen, dass die Textur sich plötzlich total verschiebt. Dann musst du im Menü "Textures" / "Texture Lock" anwählen und jeweils "Moves" und "Rotation" anklicken. Dann bleibt die Textur auf dem Brush ausgerichtet, wenn ihr sie verschiebt.

[zurück zur Hauptseite](#)

183759

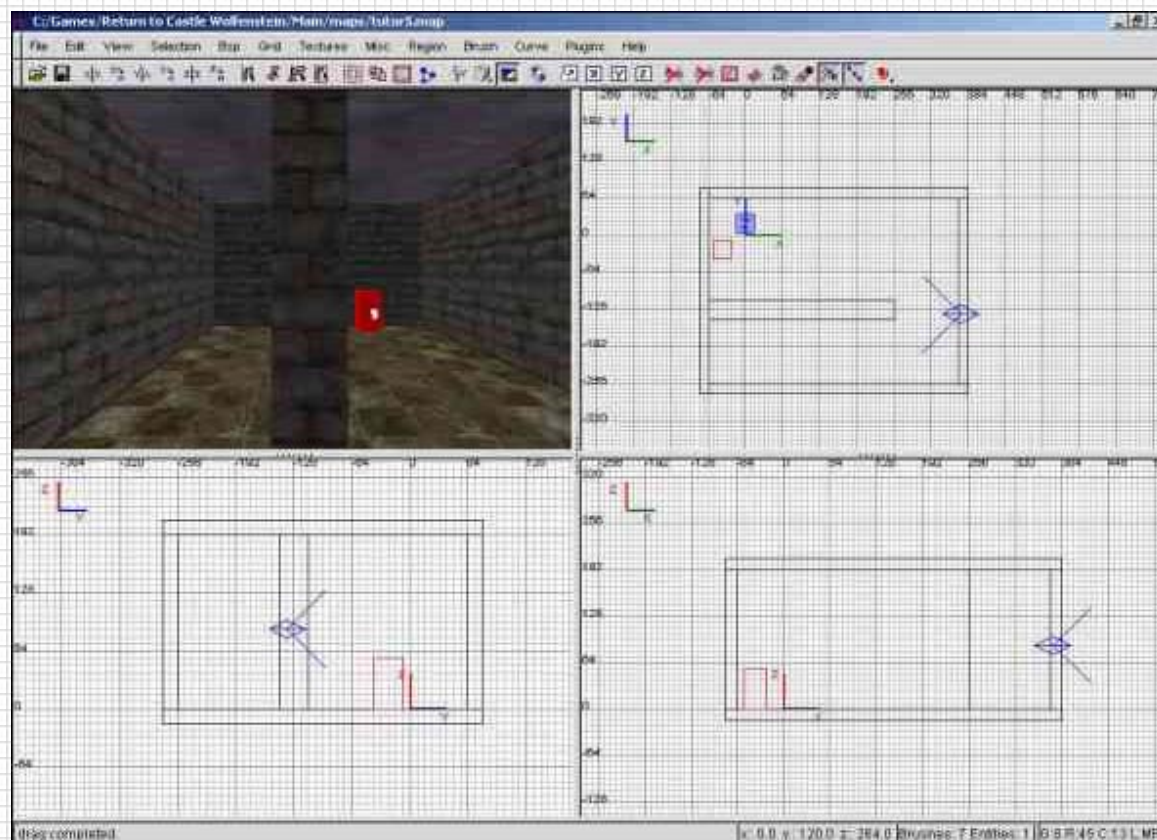


Brushs:

verwendete Map: "tutor5.map"

Ergebnismap: "tutor6.map"

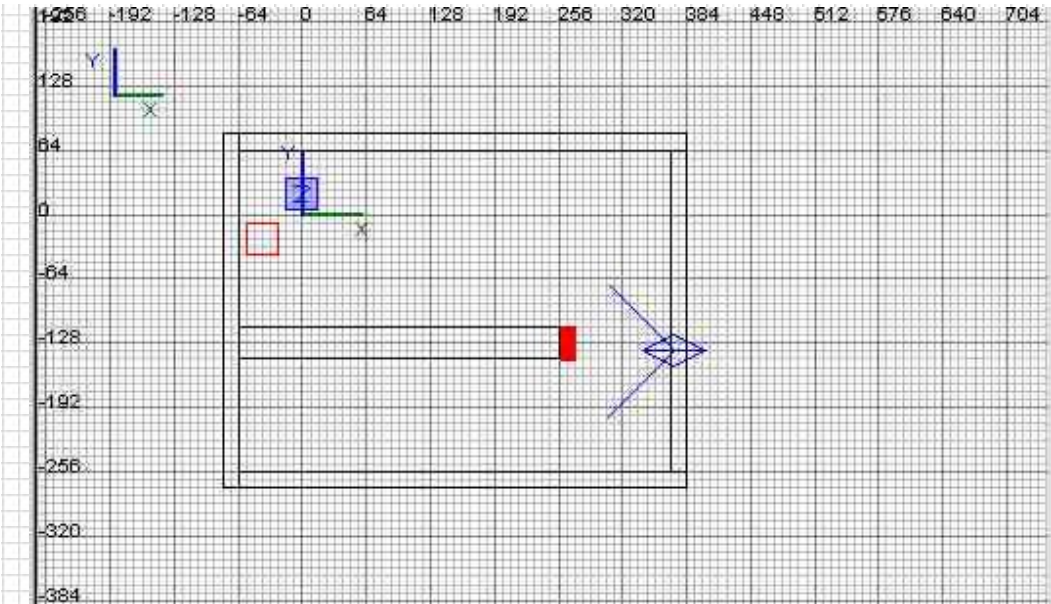
Für diese Map habe ich unsere alte Map "tutor4.map" etwas verändert. Zum einen habe ich die Wände höher gezogen, und so die Decke erhöht. Ausserdem habe ich noch 2 Brushs hinzugefügt, dass ein Raum entstanden ist, der wie ein "U" aussieht:



So, das Ganze kannst du ja jetzt ohne Probleme nachbauen.

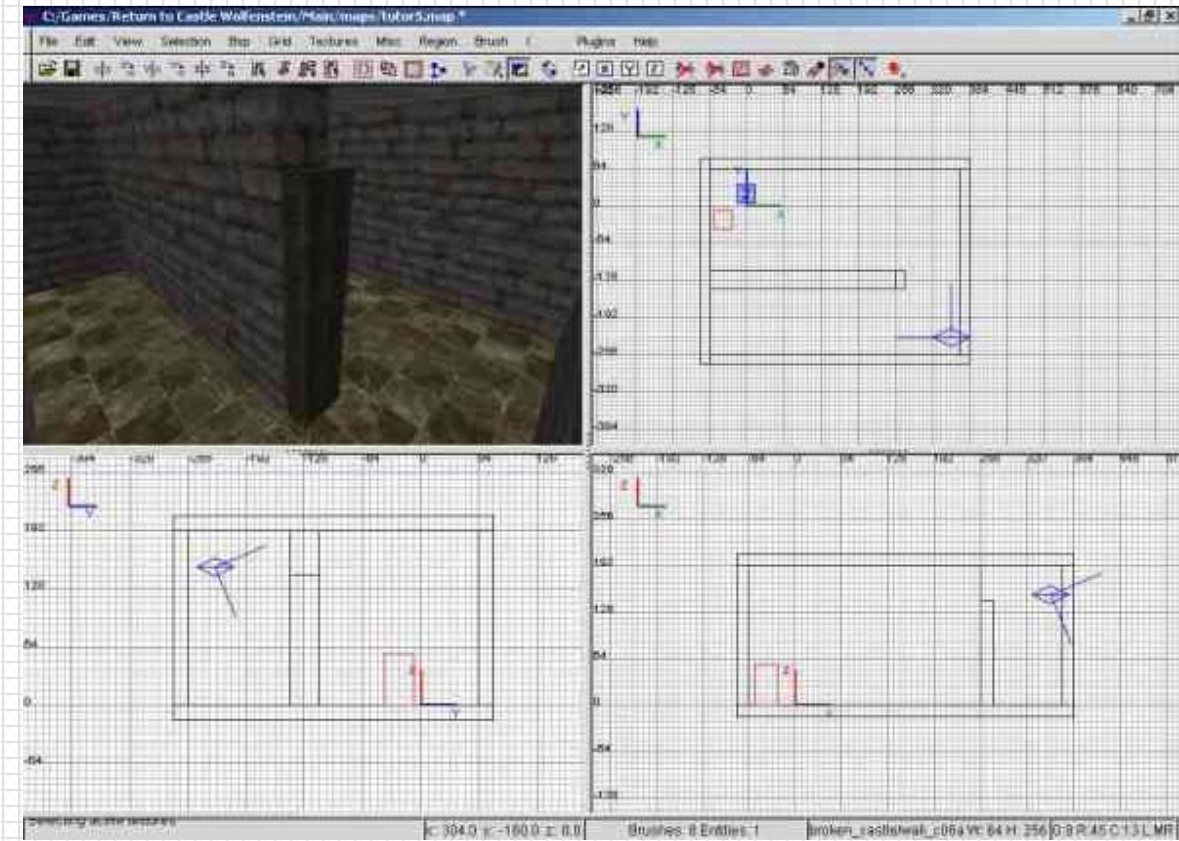
Nun wollen wir ein paar schräge Brushs erstellen, um die langweiligen Wände etwas interessanter zu gestalten:

Ich habe dir hier einen Bereich rot gekennzeichnet, den wir etwas verschönern wollen. Hier erstellst du einen Brush, der etwas kleiner ist, als die Wände. Die Textur des neuen Brush belegst du mit "broken_castle/wall_c06a". Dann drückst du "S", um den

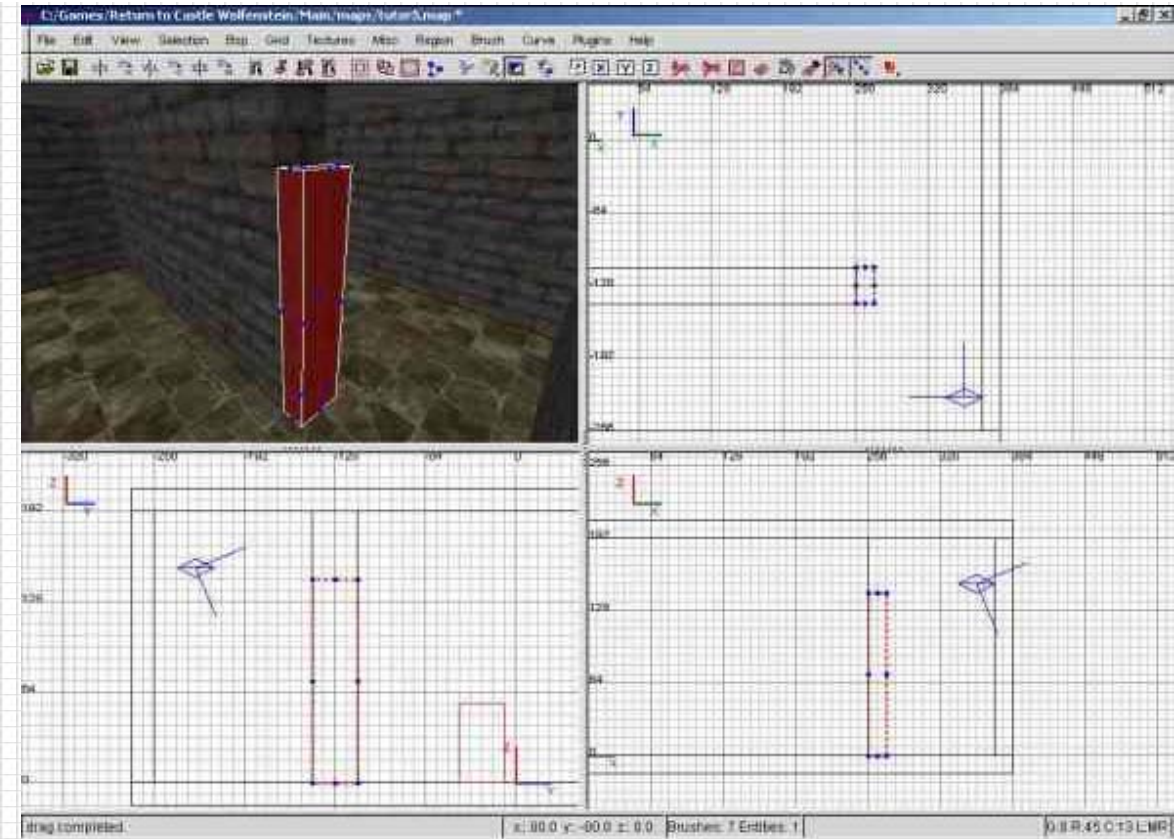


Surface Inspector aufrufen und drückst "FIT" um die Textur dem Brush besser anzupassen.

Bei mir sieht es jetzt so aus:

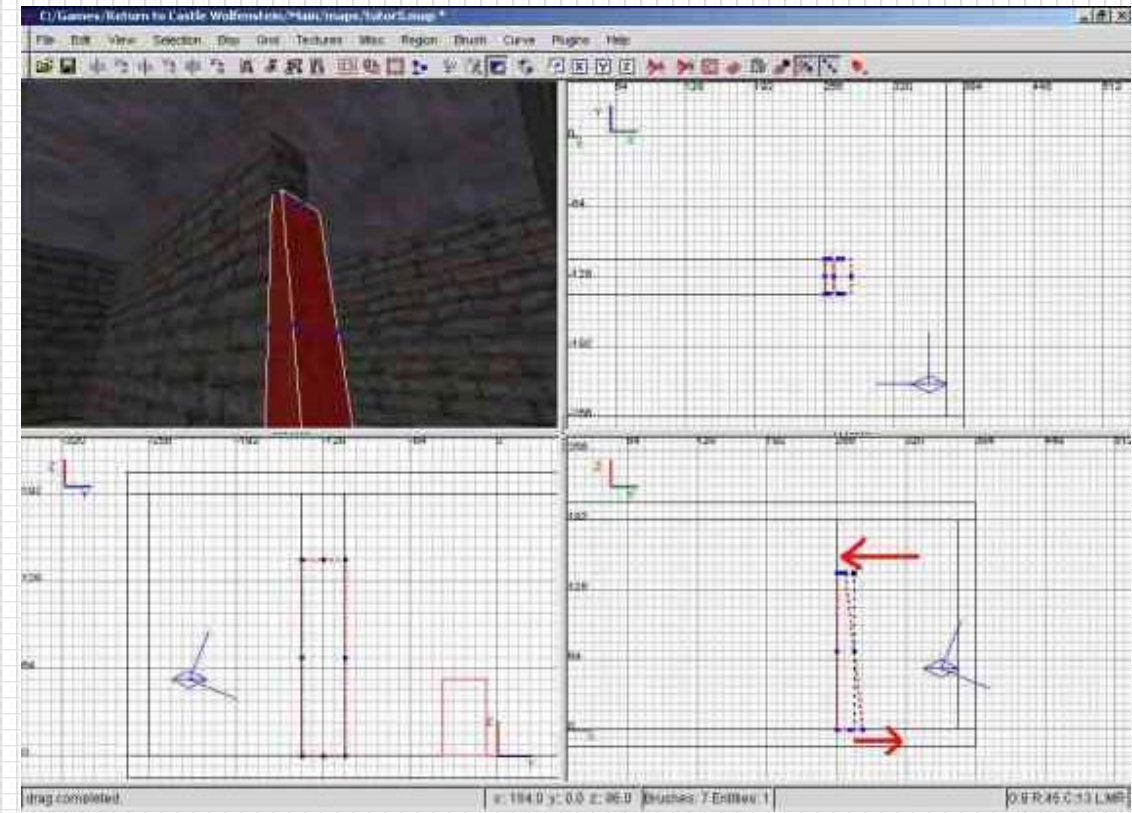


Bestimmt hast du dich schon selbst gefragt, wie man wohl schräge Brushes erstellen kann. Ich werde dies anhand dieses Brushes zeigen. Nun selektierst du ersteinmal deinen Brush, den du abschrägen willst. Nun drückst du die Taste "E" - du aktivierst den "EDGE"-Modus. Das ganze sieht jetzt so aus:



Wie du siehst, sind jetzt an den Ecken dieses Brushes blaue Punkte erschienen. Diese blauen Punkte sind nun an allen Ecken des Brushes erschienen. Übrigens, in der Top Ansicht kannst du den Brush nicht abschrägen, da du ja nur die Umrisse, aber nicht die Ecken sehen kannst.

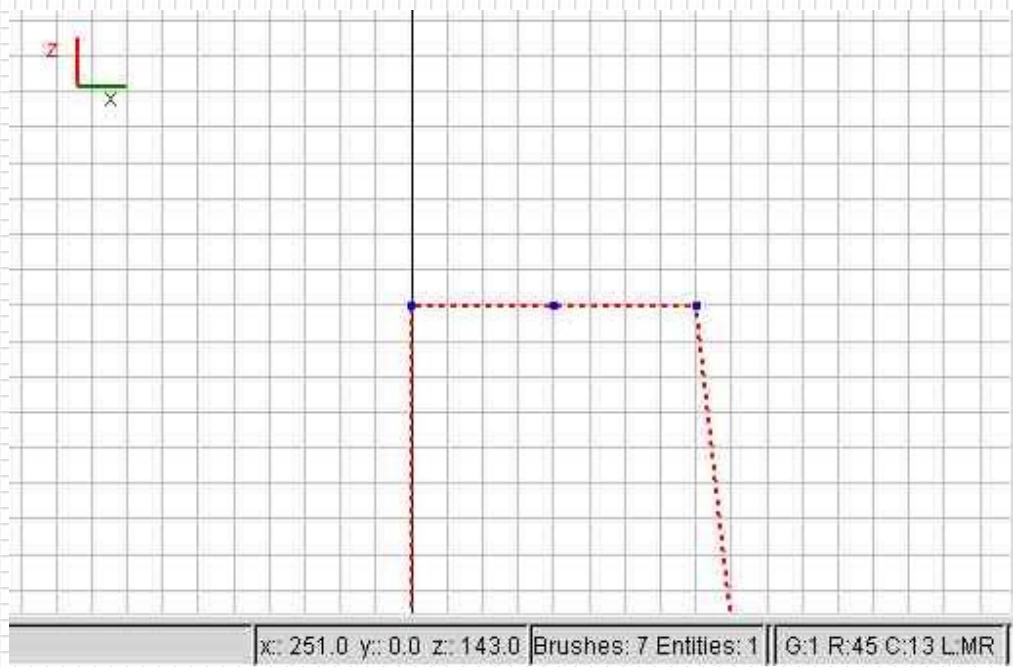
Nun kannst du etwas herumprobieren, dann siehst du gleich, wie das mit den Punkten funktioniert. Dazu musst du jeweils auf einen blauen Punkt klicken und diesen dann in irgendeine Richtung verschieben. In der 3D Ansicht könnt ihr auch gleich euer Ergebnis sehen:



Wie du sehen kannst, habe ich den oberen rechten Punkt um 8 Units nach links verschoben. Danach habe ich den unteren rechten nach rechts verschoben.

So, jetzt siehst du auch, dass an der oberen Kante die blauen Punkte sehr nah aneinander liegen, d.h. die einzelnen Punkte sind schwer zu erkennen. Daher ändern wir jetzt das Grid.

WICHTIG: Das Grid (Gitternetz) ist eigentlich der Anhaltspunkt zum Ausrichten eines Brushes, also kann man sagen, das Grid ist das Koordinatensystem. Wir wollen nun das Grid feiner einstellen, dass wir unsere Punkte auch alle erkennen:



Ich habe hier mal auf "1" gedrückt (die Eins schräg über dem "Q"). Dann musst du in den Ansichten schon etwas hineinzoomen, damit du wieder deinen altbekannten Grid erkennen kannst. Wie du siehst, haben wir die einzelnen Punkte gut im Blick. In der vorherigen Ansicht jedoch war kaum etwas zu erkennen.

Achtung:
Das Grid (oder auch Gitternetz) dient dir als Anhaltspunkt zum Ausrichten der Brushes. Standardmäßig ist das Grid auf 8 Units pro Kästchen eingestellt. Das kannst du aber jederzeit über die Zahlentasten 1 - 7 ändern.

- 1 = Jedes Kästchen beträgt genau 1 Unit (hier musst du schon ziemlich reinzoomen, dass du noch etwas erkennen kannst)
- 2 = Jedes Kästchen beträgt genau 2 Units
- 3 = Jedes Kästchen beträgt genau 4 Units
- 4 = Jedes Kästchen beträgt genau 8 Units (Standardgröße)
- 5 = Jedes Kästchen beträgt genau 16 Units
- 6 = Jedes Kästchen beträgt genau 32 Units
- 7 = Jedes Kästchen beträgt genau 64 Units
- 0 = Damit kann man das Grid ein/ausschalten

Für das ganze gibt es natürlich auch einen Menüpunkt, und zwar unter "GRID" - aber über die Zahlentasten geht das natürlich bedeutend schneller.

Nun hast du also gelernt, wie man die Kanten eines Brushes verändern kann. Aber man kann natürlich auch die Eckpunkte eines Brushes verändern. Dazu drückst du die Taste "V". Du rufst damit den "VERTEX"-Modus auf. Der "VERTEX"-Modus ist im Prinzip das gleiche wie der "EDGE"-Modus, jedoch sind die erscheinenden Punkte grün statt blau. Ausserdem könnt ihr halt die Eckpunkte verschieben, und nicht die Kanten.

WICHTIG: Du musst bei der Erstellung eines schrägen Brushes aufpassen, dass sich nicht ein Punkt mit einem anderen Punkt aufhebt. Das kann recht schnell passieren - und es führt im schlimmsten Fall dazu, dass dein Brush gelöscht wird, und später zu einer fehlerhaften Map führt (unsichtbare Brushes, Duplicate Plane Error, Totalabsturz usw.)
Sollte dir das trotzdem passiert sein, benutz also UNBEDINGT die "UNDO"-Funktion - oder beende den Radiant und starte ihn neu, da so der Fehler nicht gespeichert wird.

Natürlich soll das nicht bedeuten, es ist "gefährlich", einen schrägen Brush einzubauen - lediglich solltest du bei der Bearbeitung etwas sorgfältig sein und nicht wie ein Wilder "drauflos mappen".

Während man den "EDGE-Modus häufig benutzt, z.B. um eine Wand zu verziehen, oder eine Treppe zu erstellen, wird die "VERTEX"-Funktion mehr dazu benutzt, Terrain (also Gelände) darzustellen.

Nun kannst du mit einem Brush noch folgende Dinge anstellen:

- Einen Brush drehen:

Um einen Brush zu drehen, selektierst du diesen und klickst oben in der Icon-Leiste auf x-axis Rotate, y-axis Rotate oder z-axis Rotate. Wenn du auf ein Icon 4x klickst, ist es wieder in der Ausgangsposition, allerdings ist der Brush dann manchmal etwas versetzt und sitzt nichtmehr auf dem Grid - das aber auch nur, wenn der Brush asymmetrisch ist.

- Einen Brush spiegeln:

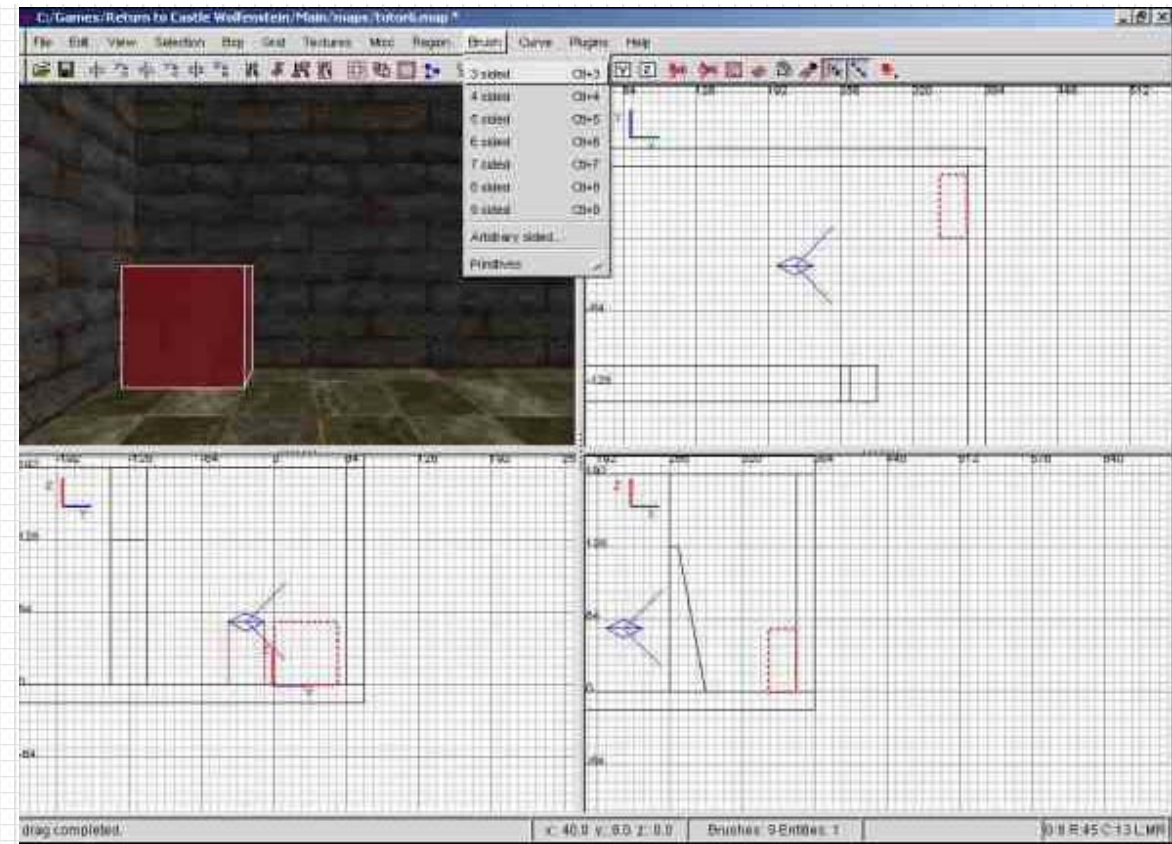
Um einen Brush zu spiegeln, selektierst du wieder deinen Brush und klickst oben in der Iconleiste auf x-axis Flip, y-axis Flip oder z-axis Flip. Du kannst natürlich auch eine ganze Gruppe von Brushes spiegeln. So werden übrigens viele CTF-Maps hergestellt.

- Mehreckige Brushes erstellen:

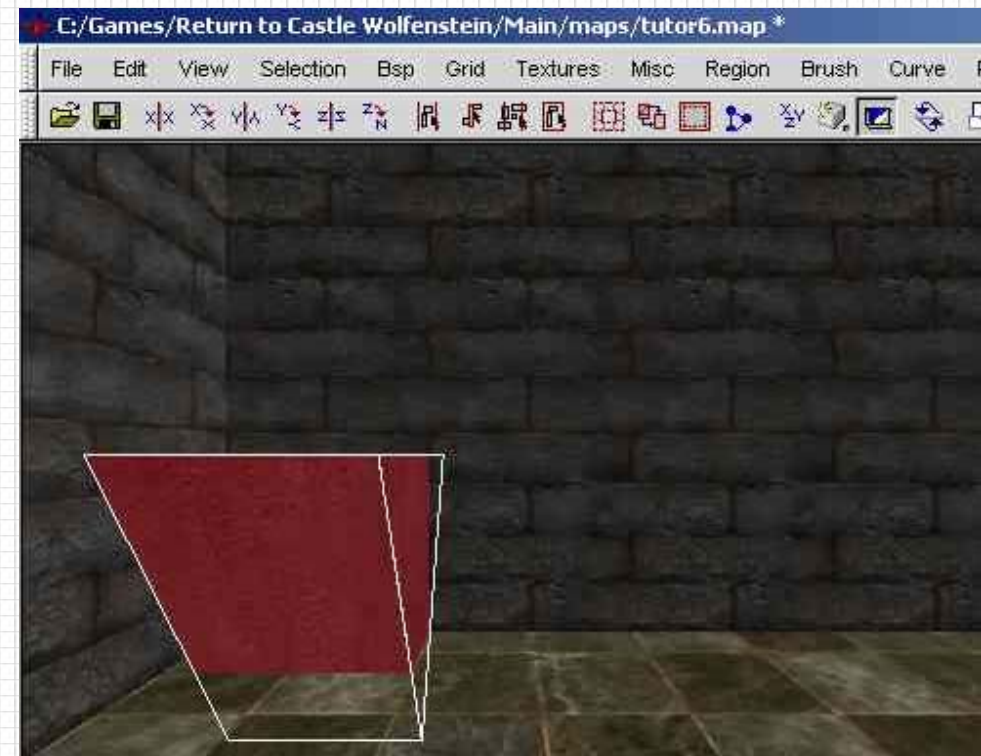
Für diese Funktion erstellst du am besten einen ganz neuen Brush. Dann klickst du im Menü auf "Brush" und dort auf "3 sided", "4 sided" usw. aus. So kannst du schnell beliebige Formen erstellen. Unter dem Punkt "Arbitrary sided" kannst du selbst eine x-beliebige Zahl einsetzen - und es wird ein Brush mit x Ecken erstellt. Achte aber darauf, dass es nicht zuviele Ecken werden (am besten nicht mehr als 15 - 20 Ecken benutzen). Sicher weißt du ja, dass ein Kreis ein n-Eck ist, also müsste man eine möglichst große Zahl eingeben um einen Kreis zu erhalten. Doch dafür gibt es eine Extra-Funktion (das nennt sich dann "Curved Surfaces"), die ich aber noch in einem extra Thema erkläre.

Wenn wir gerade dabei sind: Benutze auch diese Funktion "Arbitrary sided" nur mit Vorsicht (besonders bei ungeraden Zahlen), da sonst die Eckpunkte nichtmehr auf dem Grid landen. Das kann später bei der Compilierung zu Problemen, Fehlermeldungen (oder sogar zu Abstürzen) führen. Dafür gibt es aber auch wieder ein spezielles Tool, das man dafür benutzt. Es nennt sich "Clipper-Tool" und aufrufen kannst du dieses Tool mit der Taste "X". Dieses Tool sorgt dafür, dass alle eure Eckpunkte auf dem Grid sitzen.

Nun wollen wir mal einen 3-Seiten Brush erstellen, der als "Halterung" dienen soll. Ich habe dazu einen vier-Seiten-Brush erstellt, wie ich dir ja schon gesagt habe. Die Textur kannst du dir aussuchen, ich habe allerdings eine Textur gewählt, die wir schon benutzt haben und die sich von der Wand gut abhebt - also gut sichtbar ist. Nun wählst du (wie im Bild unten gezeigt) im Menü "Brush" und da "3 sided" an.



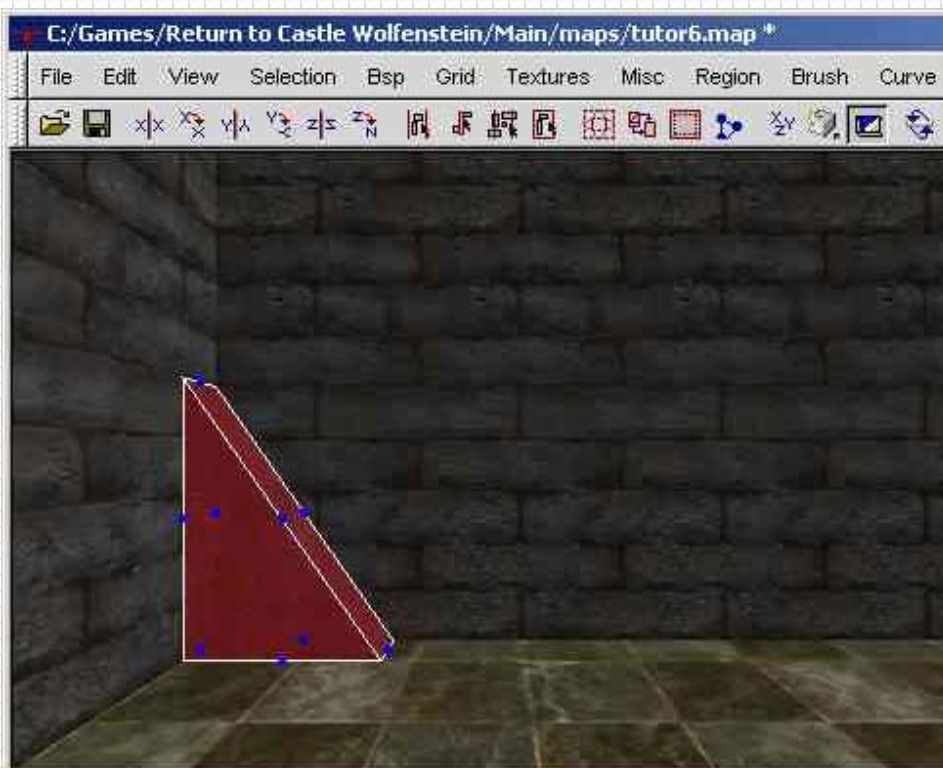
Nun gut.. hier gleich das Ergebnis:



Wie gesagt, so hatten wir uns das ja nicht vorgestellt - aber du hast ja gelernt, wie man einen Brush zurechtdreht und flippt.. also wenden wir das gleich mal an ;)



So, hier habe ich das Dreieck mal deselektiert, dass man es besser sieht - und es passt garnicht. Also drücken wir "E" für die "EDGE"-Funktion und schieben und das Dreieck zurecht.



So, das Ergebnis haben wir. Ein schöner Dreiecksbrush, der unsere Wand stützt.

Einen Brush nachträglich zerschneiden:

So, hier will ich dir den Clipper richtig erklären. Wir sind ja schon vorhin auf die Funktionen dieses Tools eingegangen - aber mehr

auch nicht, genau das folgt jetzt:

Du weisst ja schon, dass man es mit der Taste "X" aufrufen kann - aber es ist auch in der Icon-Leiste aufgeführt und zwar hier:



Hast du das Clipperwerkzeug einmal gestartet, so kannst du nun in einen selektierten Brush klicken. Dabei erscheinen kleine, blaue Zahlen. Mit diesen Zahlen bestimmst du den Schnitt durch deinen Brush. Beim ersten Klick erscheint logischerweise die Zahl "1" für deinen ersten Schnittpunkt. Klickst du dann an eine andere Stelle, so erscheint die zweite Zahl und dein Brush wird "zerschnitten" dargestellt. Die Seite, die wegfällt, wird mit einem gelben Rahmen dargestellt. Über die Tastenkombination "STRG" + "ENTER" kannst du nun die Hälfte auswählen, die du behalten willst. Drückst du nun noch einmal auf die "ENTER"-Taste, so wird diese Hälfte übernommen. Drückst du aber auf "SHIFT" + "ENTER", werden beide Hälften übernommen.

Du kannst auch einen dritten Schnittpunkt auf deinen Brush legen. Dazu legst du nach deinem zweiten Punkt einfach einen dritten Punkt fest, indem du in eines der drei Fenster klickst. Zur Erinnerung, es gibt diese Fenster: XY Top, YZ Side, XZ Front (und das 3D Fenster, es nützt uns hier allerdings nichts). Deshalb kannst du nachträglich jeden blauen Punkt mit der Maus anklicken und dann die Position des Punktes verändern. Sobald du mit dem Mauspfel über dem blauen Punkt bist, verändert sich der Mauszeiger in ein Fadenkreuz und du kannst den Brush verschieben. Dann orientierst du dich am besten an allen drei Fenstern.

WICHTIG:

Willst du einen blauen Punkt nachträglich verschieben, solltest du **UNBEDINGT** darauf achten, dass das Fadenkreuz wirklich sichtbar ist. Klickst du nämlich "daneben" - so startet das Clipper-Tool von vorne und deine bisherigen Punkte sind verloren. Hast du dann deine Bearbeitung abgeschlossen, musst du das Clipper-Tool wieder abschalten (indem du noch einmal die Taste "X" drückst).

Hier nochmal die Shortcuts:

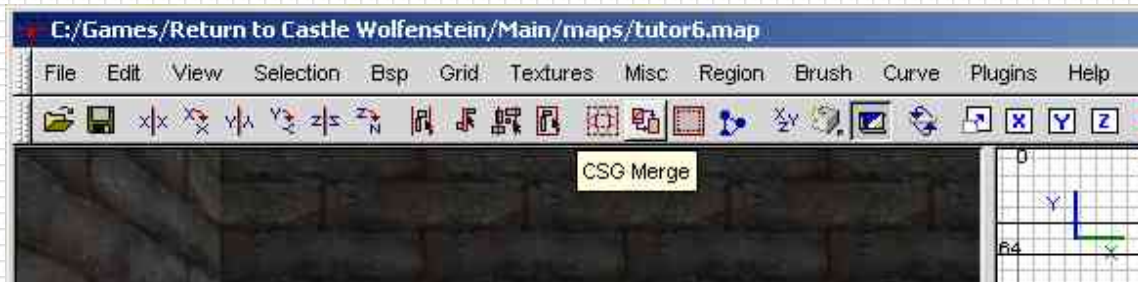
"STRG" + "ENTER" = von einem geteilten Objekt die Seite auswählen

"ENTER" = Geteiltes Objekt übernehmen

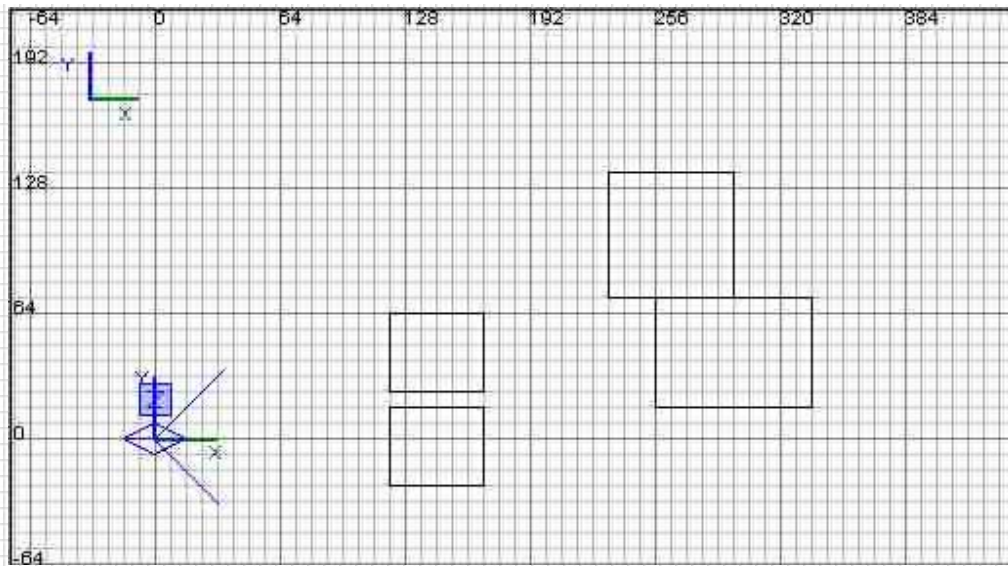
"SHIFT" + "ENTER" = beide zerschnittene Hälften übernehmen

Mehreren Brushes miteinander "verschmelzen":

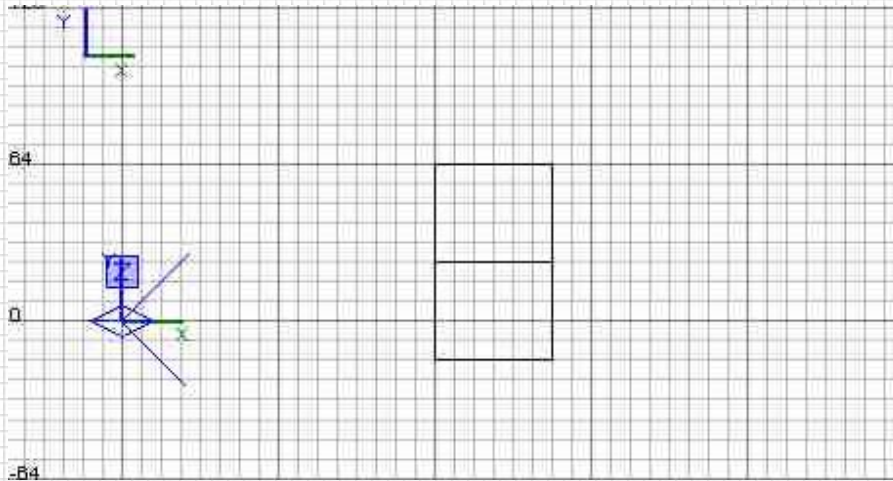
Nun, sogar für diese Funktion hat der Radiant ein Tool parat - dafür benutzt ihr das Icon "CSG Merge" (es ist neben "Hollow" zu finden)



Der Sinn dieses Tools ist es, mehrere Brushes wieder zu einem einzigen Brush zu verbinden. Das funktioniert aber nur, wenn sie die gleiche Größe haben und direkt nebeneinander liegen. Diese Brushes kann man z.B. nicht verbinden:



Diese Brushes kann man verbinden:



Um diese Funktion nun auszuführen, klickst du die beiden Brushes an, die du verbinden willst - und klickst auf das Icon "CSG Merge" - wenn du mit dem Verbinden ein Problem hast, solltest du die Taste "N" drücken (um das Entity-Fenster zu öffnen) und unten auf die Schaltfläche "Console" klicken. Im folgendem Fenster werden alle Meldungen angezeigt, die während dem Mappen anfallen, so auch Fehlermeldungen.

Habt ihr z.B. das vor euch : "Merging...
Cannot add a set of brushes with a concave hull."

dann machst du nichts falsch, sondern diese Funktion kann nicht durchgeführt werden. Kommt hingegen keine Fehlermeldung, hat alles geklappt - und ihr könnt euer Ergebnis in der 3D Ansicht bewundern.

[zurück zur Hauptseite](#)

183759



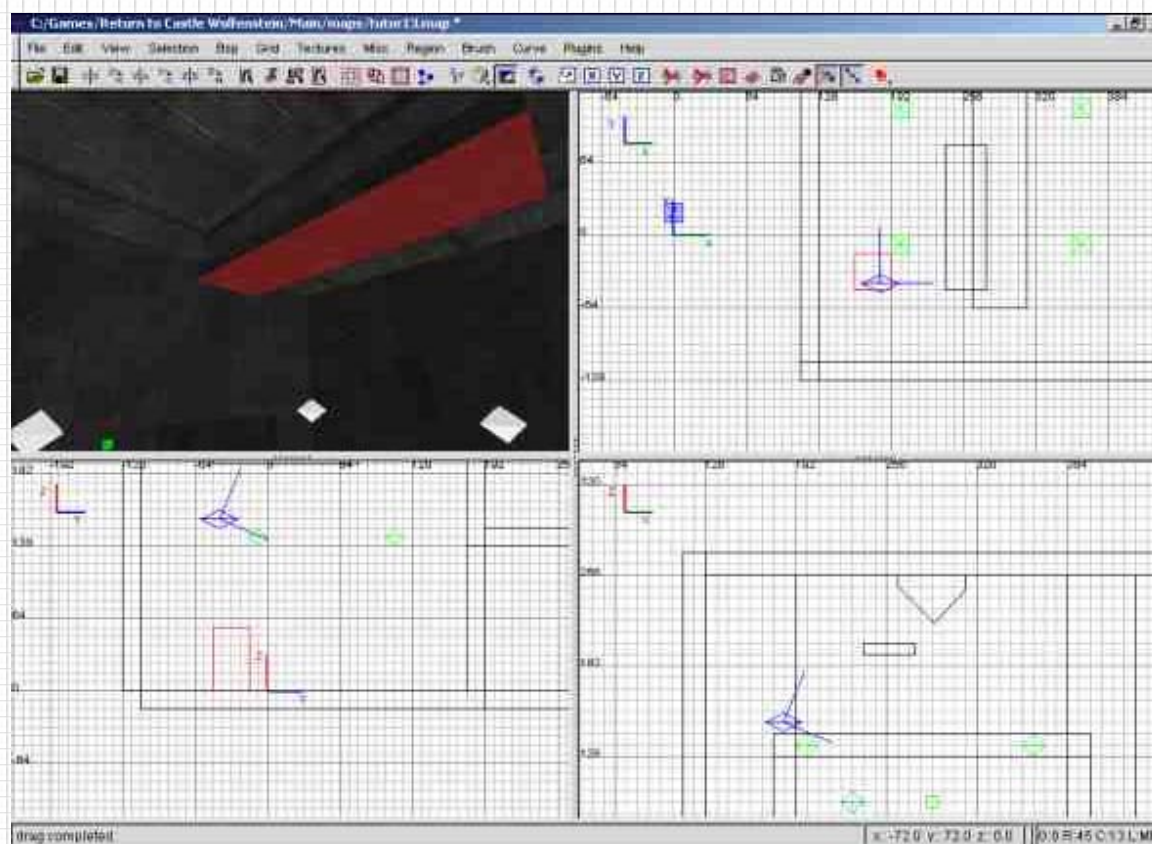
Beleuchtung:

verwendete map: "tutor12.map"

Ergebnismap: "tutor13.map"

Ich habe dir hier mal eine einfache Map gebaut. Nun wollen wir diese beleuchten. Natürlich könnten wir auch einfach Light-Entities setzen, aber das ist uns hier zu einfach. Wir wollen Texturen einsetzen, die von selbst leuchten - die einen Shader haben.

So, dann legen wir mal los - ich habe dir an der Decke einen Brush gesetzt, der als Dekoration für unseren "Licht-Brush" dient. Dazu machs du einen Brush, der 128 Units lang ist - und ca.32 Units breit ist. Nun belegst du diesen Brush mit der Textur "xlab_wall/xconcrete_c54_c". Dann selektierst du die Unterseite des Brushes:



Diese Unterseite belegst du nun mit der Textur "lights/light_xlight_1500". Wenn die Lichttextur bei dir nicht so genau sitzt, kannst du sie ja noch mit dem Surface Inspector zurecht fixen, oder sie per Hand ausrichten. Das kannst du machen, in dem du "SHIFT" und eine der Cursor-Tasten drückst.

"SHIFT" + Cursor nach oben: Textur verschiebt sich nach links

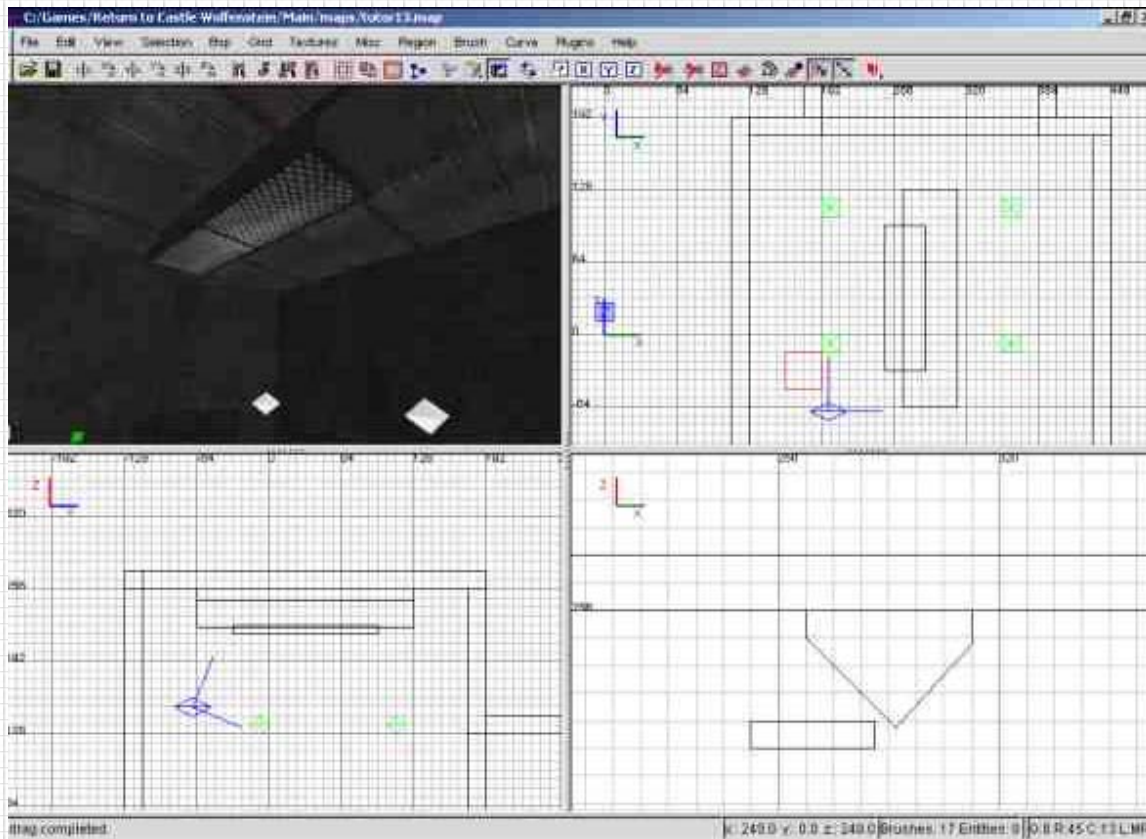
"SHIFT" + Cursor nach unten: Textur verschiebt sich nach rechts

"SHIFT" + Cursor nach rechts: Textur verschiebt sich nach oben

"SHIFT" + Cursor nach links: Textur verschiebt sich nach unten

Wenn du die Textur automatisch auf den Brush anpassen willst, drückst du "S" um den "Surface Inspector" aufzurufen. In dem Menü klickst du auf "FIT".

Jetzt schiebst du diesen Brush ganz nah an unseren Dekorations-Brush, dass es ungefähr so aussieht:

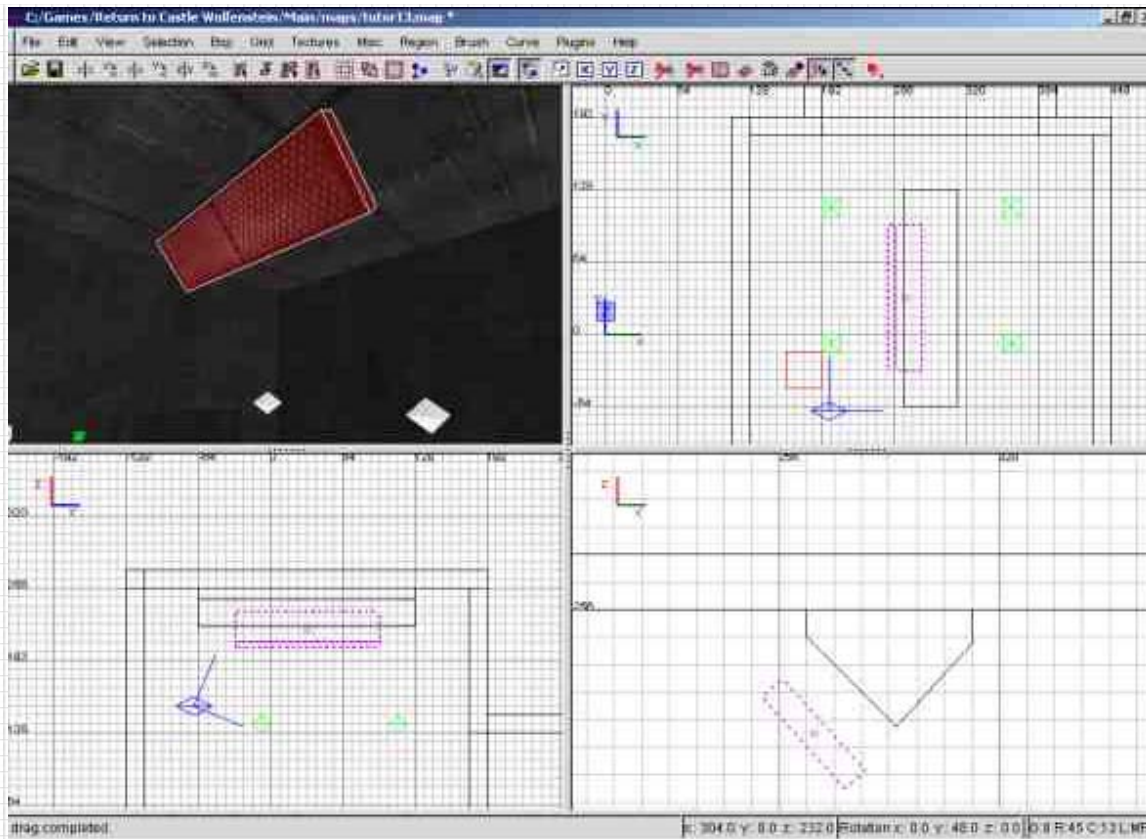


Ich hab hier den Brush deselektiert, dass du ihn besser sehen kannst. Aber du lässt ihn natürlich gleich selektiert. Nun drückst du auf den Button "Free-Rotate" in der Menü-Leiste:



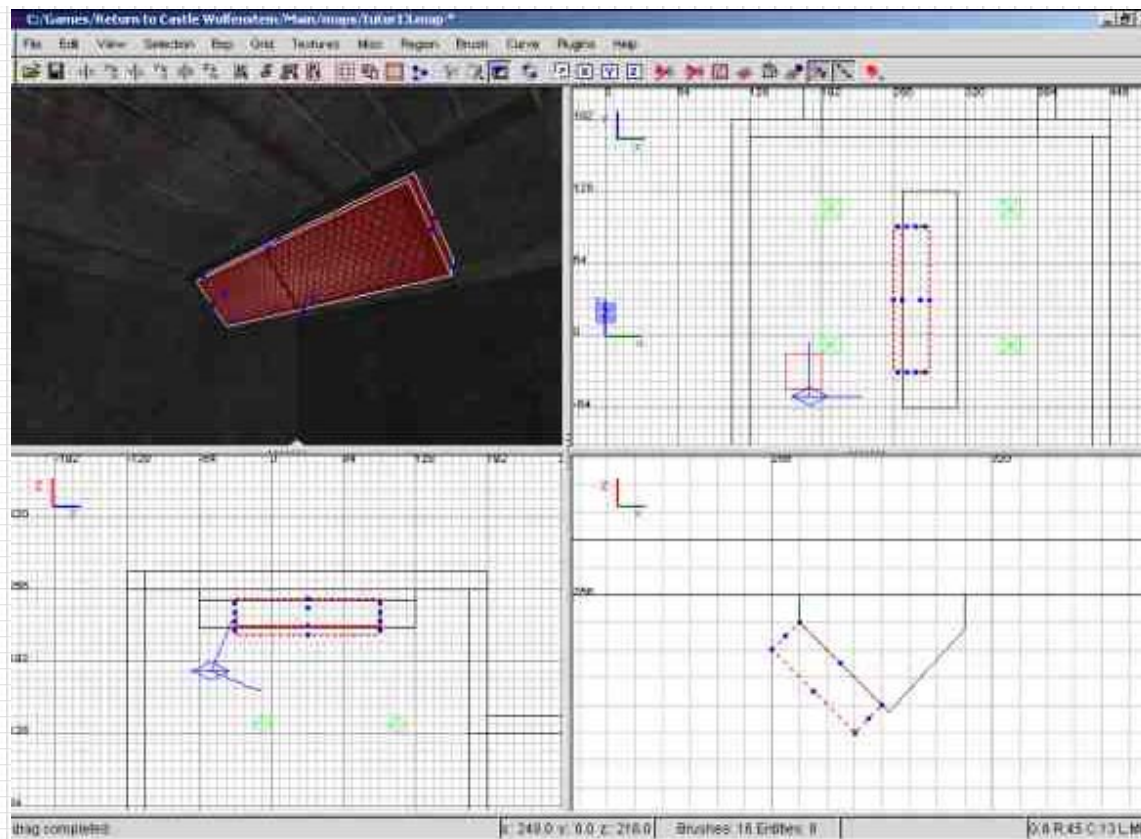
Für diese Sache wechseln wir in die Front-Ansicht (rechts unten). Hier kannst du den Dekobrush und unseren Lampen-Brush von der Seite sehen. Jetzt wirst du bemerken, dass die Umrandung des Lampen-Brushs gar nicht mehr rot, sondern violett dargestellt wird. Ich weiss nicht, ob das ein Bug vom Radi ist - oder ob hiermit angezeigt wird, dass wir uns im "Free-Rotate"-Modus befinden. Aber das nur am Rande.

Jetzt klickst du mit der linken Maustaste in die Front-Ansicht (rechts unten) und bewegst deine Maus langsam ein wenig nach oben und nach unten. Du kannst in der 3D Ansicht sehen, wie sich dein Brush dreht. Hier stellst du auch sicher fest, dass der Brush jetzt nicht mehr in unser Grid passt, d.h. er steht nicht mehr auf den Linien:



Wie du siehst, habe ich den Brush schon so ausgerichtet, dass er ungefähr an unseren Dekorations-Brush passt. Hier solltest du vielleicht ein wenig mit dem "Free-Rotate"-Modus spielen, es kann sein, dass du es nicht so schnell so schön hinbekommst, wie ich. Aber dafür übst du ja auch noch :)

Hast du es geschafft, dass dein Brush jetzt auch so sitzt, wie im Bild oben, kannst du die "ESC"-Taste drücken und damit den "Free-Rotate"-Modus verlassen. Du kannst gleich nochmal die "ESC"-Taste drücken um auch den Brush zu deselektieren. Doch du musst ihn gleich wieder anklicken, schliesslich wollen wir ihn noch an unseren Dekorations-Brush setzen. Dazu schieben wir ihn noch etwas näher an den Deko-Brush heran und drücken die Taste "E". Hier gelangen wir in den "EDGE"-Modus. Nun ziehst du die Ecken ganz nah an den Deko-Brush heran, bis es ungefähr so aussieht:



Nun kann es passiert sein, dass sich die Lampen-Textur verschoben hat. Dann wählst du wieder "S" für den "Surface Inspector" und drückst den "FIT"-Button. Achte aber darauf, dass der Winkel der Textur stimmt, sonst sieht deine Lampe nachher anders aus.

[zurück zur Hauptseite](#)

183759



Spezielle Beleuchtungsarten unter RtCW:

verwendete Map: "tutor10.map"
Ergebnismap: "tutor11.map"

So, in diesem Kapitel bringe ich dir bei, wie man richtige Beleuchtungen einsetzt. Eine (einfache) Lichtquelle kannst du ja schon selber setzen. Dazu öffnest du jetzt die Map "tutor7.map" - das Endergebnis kannst du dir in der Map "tutor8.map" ansehen.

Folgendes hab ich dir gebaut:

Ich habe hier eine Art Raum gebaut, an den eine kleinere Kammer anschliesst. Dieser soll etwas an die "X-Labs" aus dem Original-Spiel erinnern. Dazu habe ich auch "X-Labs"-Texturen verwendet, z.B. besteht der

Boden aus der Textur "xlab_wall/xconcrete_c54_c"
die Wände aus der Textur "xlab_wall/xconcrete_c58m"
die Decke aus der Textur "xlab_wall/xconcrete_c57"

So, nun erzähle ich dir erstmal etwas über die neuen Möglichkeiten der "Lightentities" in RtCW. Wenn du schon für Q3 gemappt hast, kommt für dich auch neue Dinge hinzu. In Q3 gibt es nur einfache Lightentities, aber in RtCW gibt es noch drei neue Light-Entities, nämlich:

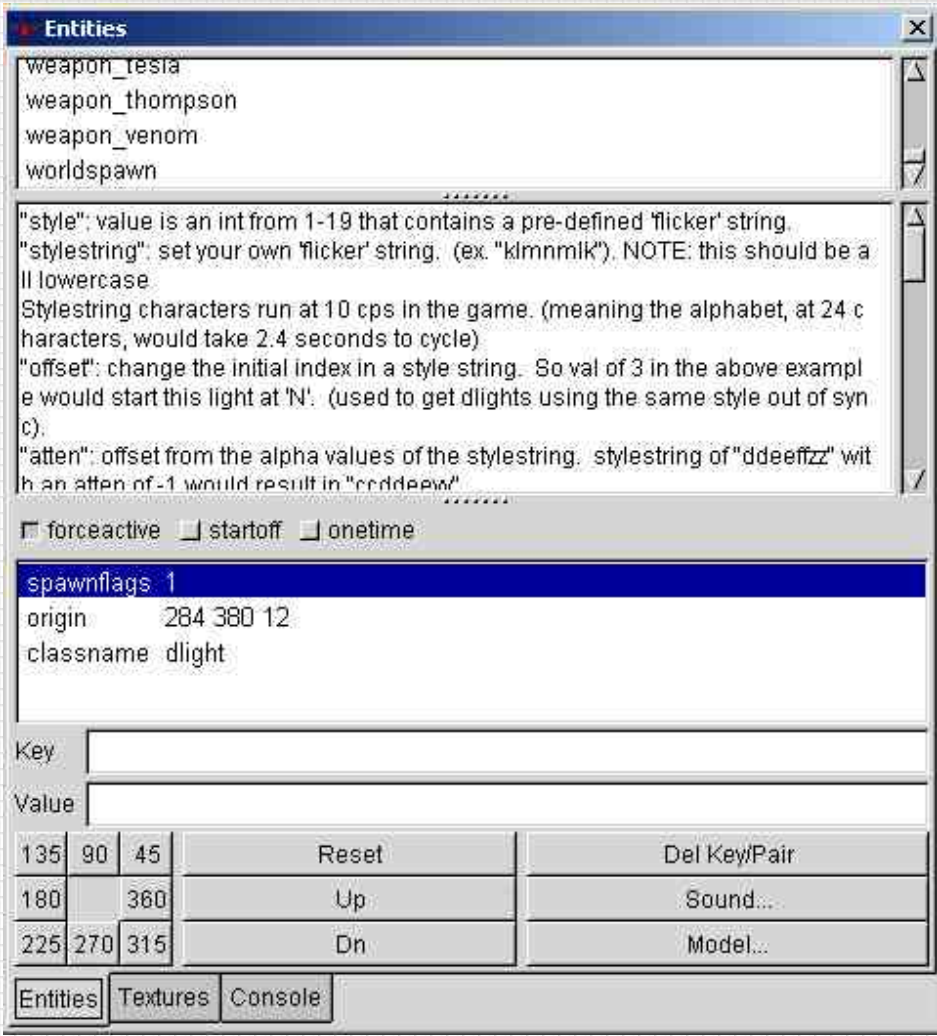
- dlight
- light_junior
- corona

Nun gehe ich mal etwas auf diese drei "Spielzeuge" etwas näher ein.

"dlight"-Entity:

Hiermit kann man jede Lichtquelle flackernd darstellen, was natürlich für Atmopshäre sorgt (dieser Effekt wird oft in der Crypt eingesetzt). Um solch einen Effekt zu haben, klickst du im Top Fenster zweimal mit der rechten Maustaste und wählst "dlight". Dann ziehst du dieses Entity ins Zentrum deiner Lichtquelle.

Wie jedes andere Entity hat auch das "Dlight"-Entity mehrere Keys, die du dir ansehen solltest. Also drückst du "T" um das Texturenfenster zu öffnen, anschliessend drückst du unten auf das "Entity", um in das "Entity"-Fenster zu kommen:



Hier kannst du z.B. folgendes wählen:

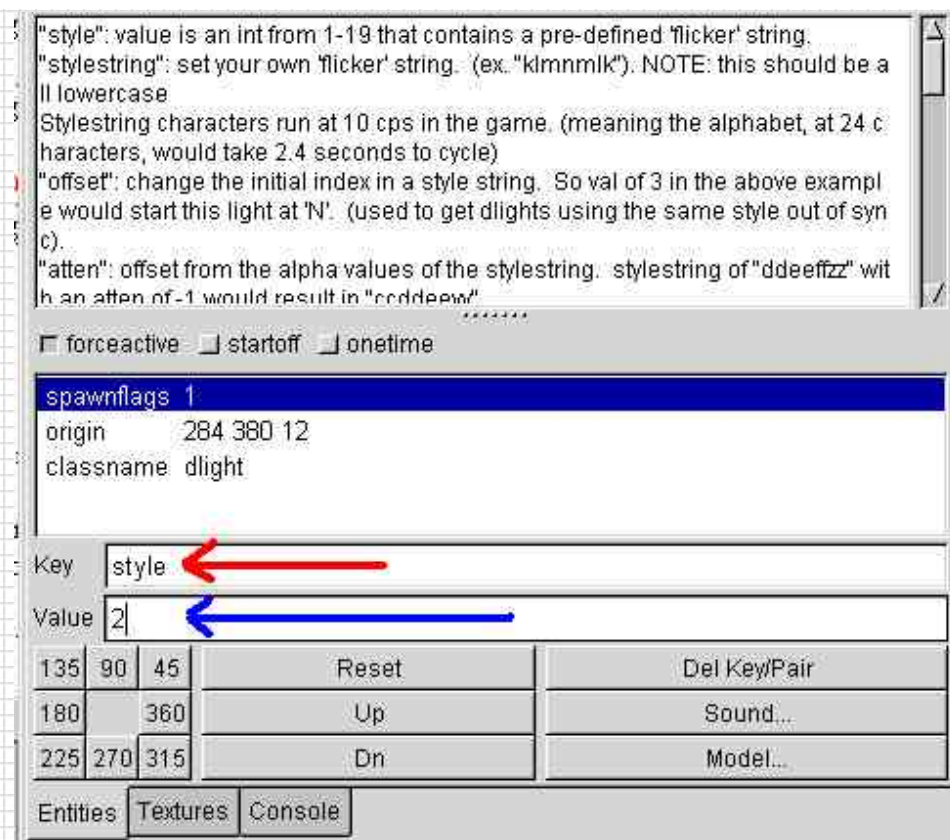
- forceactive: hast du hier ein Häkchen gesetzt, WIRD das Flackern dargestellt, egal ob dieser Effekt vom Spieler aktiviert oder deaktiviert ist.
- startoff: hast du hier ein Häkchen gesetzt, muss der Effekt erst aktiviert werden, d.h. Er ist zu Beginn der Map deaktiviert, also nicht aktiv
(Effekte werden aktiviert, in dem man sie triggert - das erkläre ich aber noch in einem anderen Kapitel)
- onetime: hast du hier ein Häkchen gesetzt, wird das Flackern nur einmal dargestellt

Nun schau wir uns mal die untere Hälfte dieses Fensters genauer an:

Ich habe dir hier das "Key"-Feld und das "Value"-Feld gekennzeichnet.

Das "Key"-Feld bestimmt Keys, die du oben im Textfeld sehen kannst. Ich habe in die Zeile "Style" eingegeben. Oben siehst du, dass man mit diesem Key definieren kann - in diesem Fall die Art, wie unsere Lichtquelle nun flackert.

In das "Value"-Feld gibst du ein, was für eine Eigenschaft zu zuweisen willst. So steht hier "2" für eine der 19 vor-konfigurierten Flackerarten.



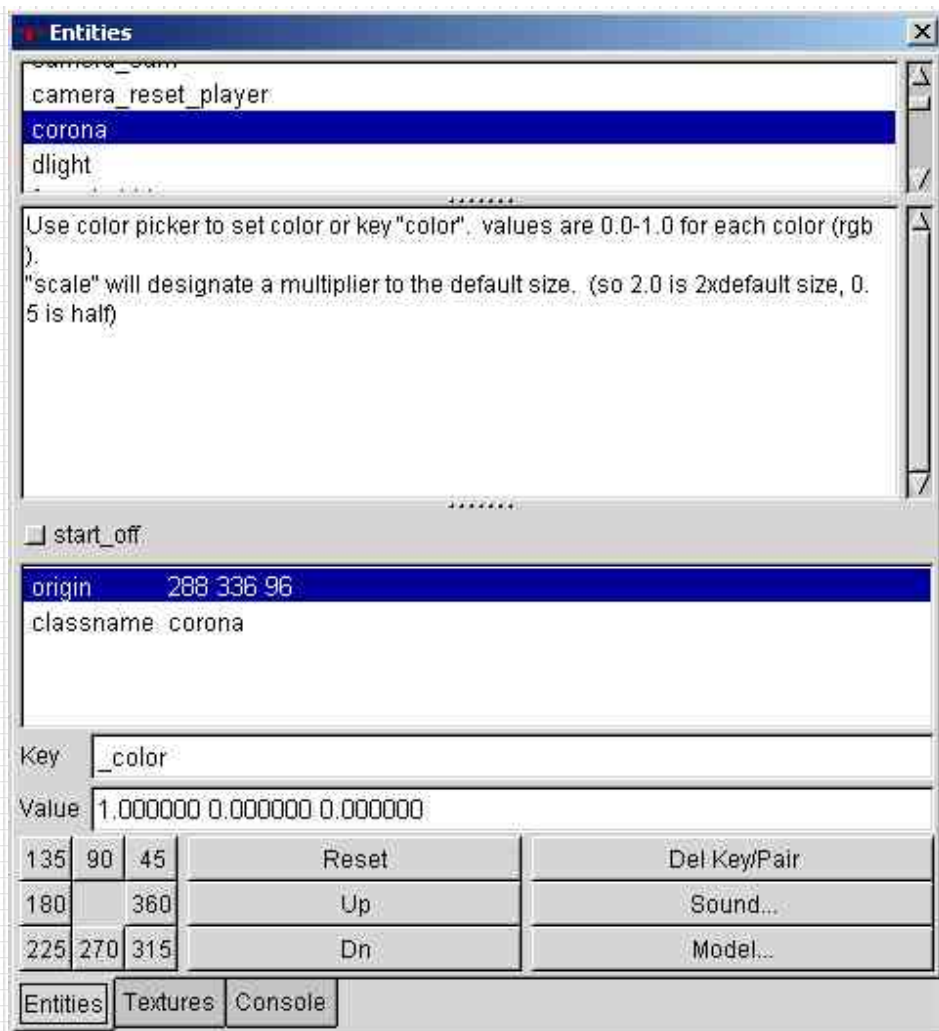
Die anderen "Values" stehen für ein einfaches Blinken, z.B. einer Alarmleuchte. Dazu spielst du am besten ein wenig mit diesen "Values", da es leider noch keine genaue Beschreibung gibt.

WICHTIG: Zum Schluss musst du allerdings "ENTER"drücken, damit diese Eigenschaften auch übernommen werden

Nun kannst du natürlich noch dem Dlight eine Farbe zuweisen - dazu schliesst du einfach das "Entity"-Menü und drückst "K". Damit öffnest du das "Colour-Menü".

Übrigens: Du solltest hier keine "starken" Farben benutzen, da diese dann deine Map zu arg einfärben. Benutze also lieber einen schwachen Farbton.

"Corona"-Entity:

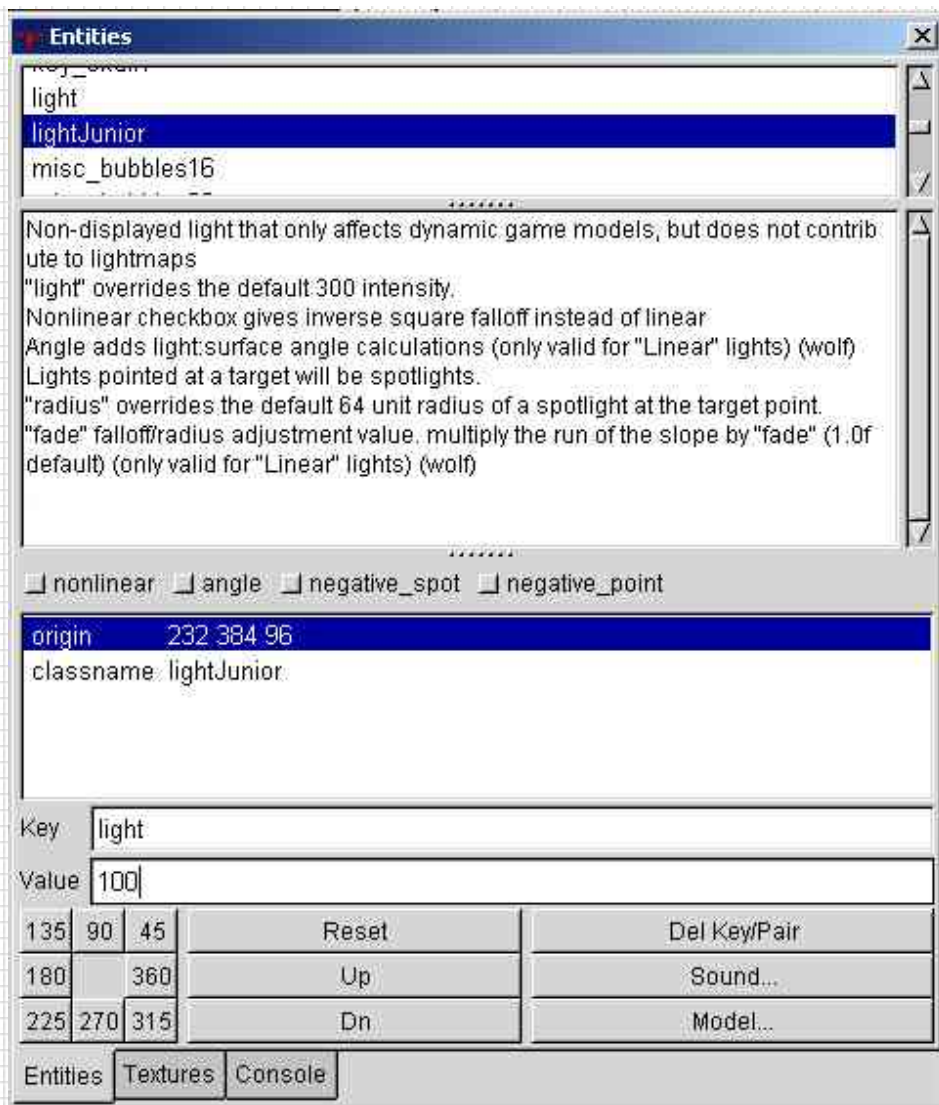


Wie du sehen kannst, ist dieses Entity ein recht einfaches Entity. Man kann hier nicht soviel einstellen, wie oben bei dem Entity "dlight". Dieses Entity erzeugt eine Art Schein - egal, wo ihr dieses Entity hinsetzt.

Ihr könnt hier (wie im Bild zu sehen ist) die Farbe per "Value" eingeben, oder auch die Größe bestimmen. Diese Größe kannst du auch wieder im "Value" eingeben - du musst dafür allerdings als "Key" "Scale" eingeben - sonst weiss der Radiant nicht, welchem Key er nun deine Value zuweisen soll. Oben siehst du, wie sich die Größe berechnet, $2 \times X$ ist die Formel. Also ist 2 der Standardwert, 0.5 ist die Hälfte ($0.5 \times 2 = 1 = \text{Hälfte von } 2$). Eine Corona ist im Kern heller - aber die Helligkeit nimmt zum Rand hin ab - vielleicht habt ihr ja die Sonnenfinsternis gesehen, das war auch eine Corona, was da noch zu sehen war.

Nun kannst du noch die Farbe der Corona festlegen - dazu schliesst du wieder das "Entity"-Fenster und drückst "K" - zwar kannst du die Farbe auch im "Entity"-Fenster festlegen, allerdings ist es hier komfortabler, da du deine eingestellte Farbe auch sehen kannst.

"lightjunior" Entity:



Nun, dieses "lightJunior" ist im Prinzip ein ganz normales Licht. Allerdings erleuchtet es nur Modelle in der Map, also keine Wände. Diese bleiben dunkel. So wird aber z.B. die Waffe angeleuchtet. Wenn du willst, kannst du auf jedes normale Licht auch ein LightJunior setzen, dann werden die Wände und Modelle gleichermaßen beleuchtet (dazu musst du dem "Light-Entity" UND dem "LightJunior" die gleichen Eigenschaften zuteilen.

Im Bild oben habe ich wieder für den "Key" den Wert "light" eingegeben - und als "Value" die Zahl "100". Also leuchtet dieses LightJunior jetzt genauso stark, wie ein Licht, dass den "light"-Wert auf "100" hat.

Nun kannst du das "Entity"-Fenster wieder schliessen. Wenn du willst, kannst du auch hier wieder die Farbe des Lichts ändern, die das "LightJunior" aussendet. Also drückst du wieder "K" - und suchst dir eine Farbe aus. So kannst du z.B. die Waffe blutrot oder giftgrün darstellen.

WICHTIG: Um diese Effekte zu sehen, musst du die Map mit Licht compilieren !!! Also z.B. "BSP, -vis, -light"

[zurück zur Hauptseite](#)

183759



Beleuchtung:

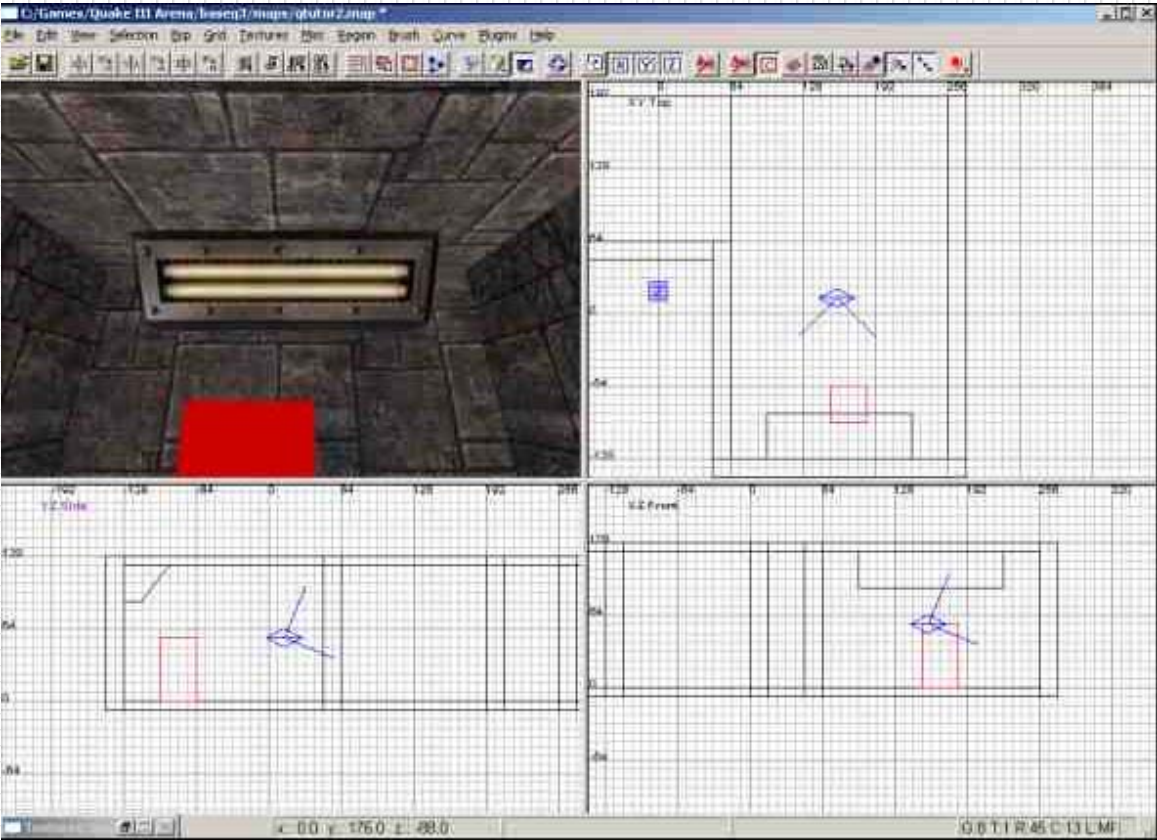
Verwendete Beispielmap: "qtutor1.map"
Ergebnis-Map: "qtutor2.map"

WICHTIG: Diese Maps habe ich mit dem Q3Radiant erstellt. Alle Maps, die ich für den Q3Radiant erstelle, kennzeichne ich mit einem "q" vor dem restlichen Namen. Öffne diese Maps auch nur im Q3Radiant, da wir sonst unterschiedliche Ergebnisse haben.

ACHTUNG: Wenn du für RtcW mappst, kannst du das hier nur überfliegen. Wichtig wird es erst wieder, wenn ich dir erkläre, wie du mit der "free-rotation"-Funktion umgehst.

Ich habe dir hier mal eine einfache Map gebaut. Nun wollen wir diese beleuchten. Natürlich könnten wir auch einfach Light-Entities setzen, aber das ist uns hier zu einfach. Wir wollen Texturen einsetzen, die von selbst leuchten - die einen Shader haben.

So, dann legen wir mal los - ich habe dir im ersten Raum (in dem auch der Playerstart ist) einen Brush an die Decke gesetzt, der mit der Textur "gothic_trim/metalblackwave01" belegt ist. Dieser dient nun als Dekoration für unseren "Licht-Brush". Nun bauen wir uns das Licht. Dazu klickt ihr die schräge Seite des Brushes an (den Brush mit "SHIFT" + "STRG" + linke Maustaste), und belegt diesen mit der Textur "/base_light/baslt4_1_4k":



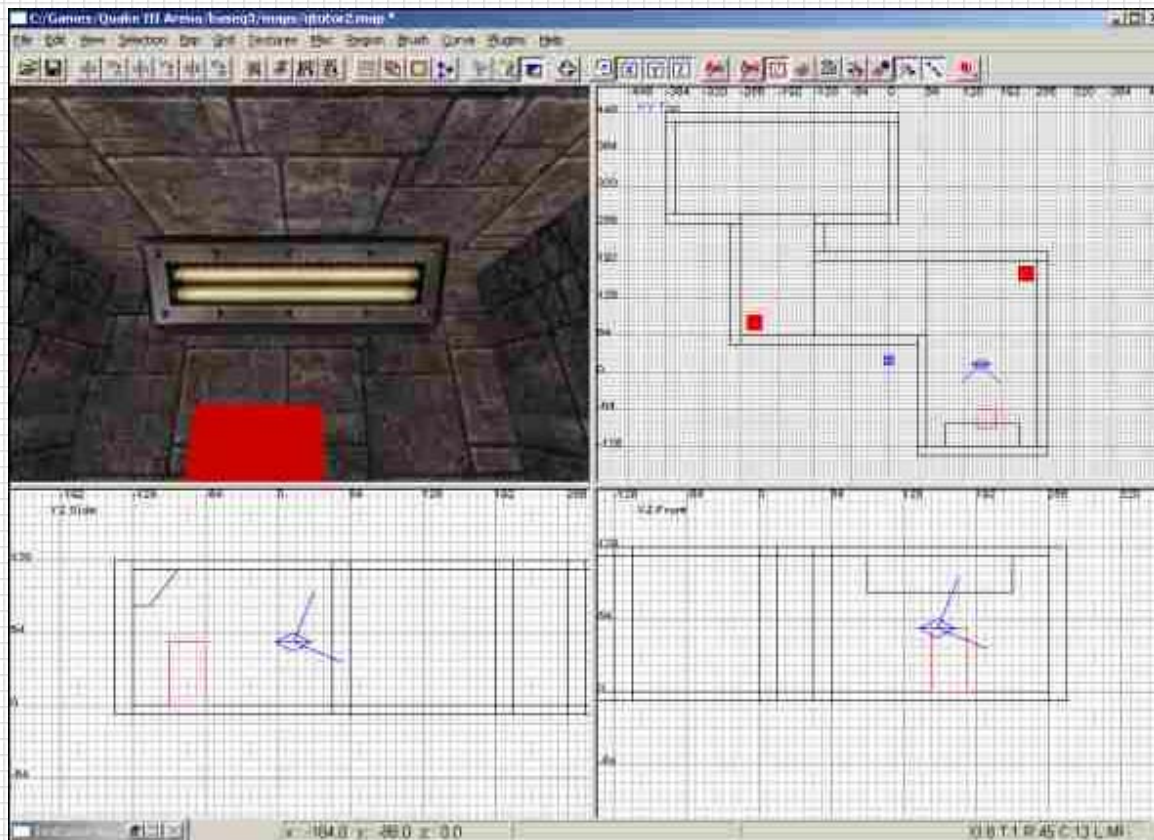
Wenn die Lichttextur bei dir nicht so genau sitzt, kannst du sie ja noch mit dem Surface Inspector zurecht fixen, oder sie per Hand ausrichten. Das kannst du machen, in dem du "SHIFT" und eine der Cursor-Tasten drückst.

- "SHIFT" + Cursor nach oben: Textur verschiebt sich nach links
- "SHIFT" + Cursor nach unten: Textur verschiebt sich nach rechts

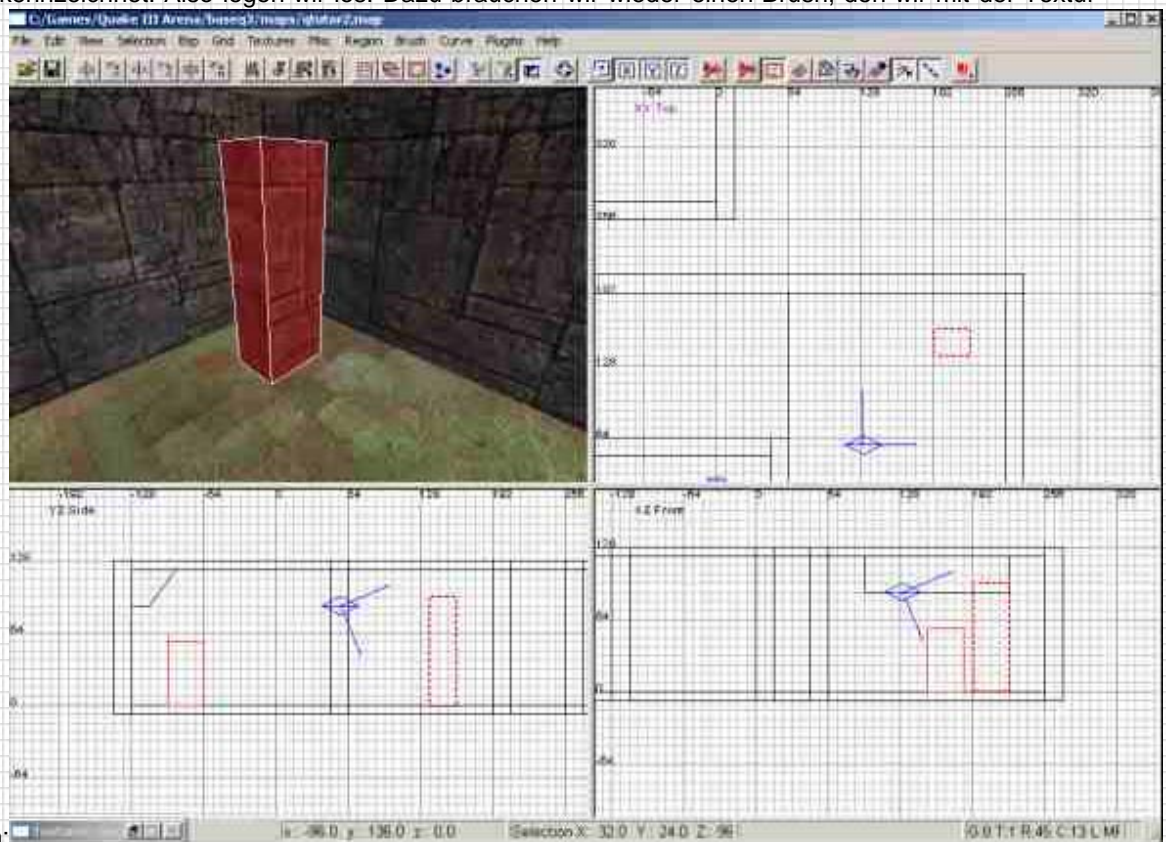
"SHIFT" + Cursor nach rechts: Textur verschiebt sich nach oben

"SHIFT" + Cursor nach links: Textur verschiebt sich nach unten

So, nun wollen wir noch ein paar weitere Lampen bauen, die wir dann jeweils in die Ecken des Ganges setzen wollen:

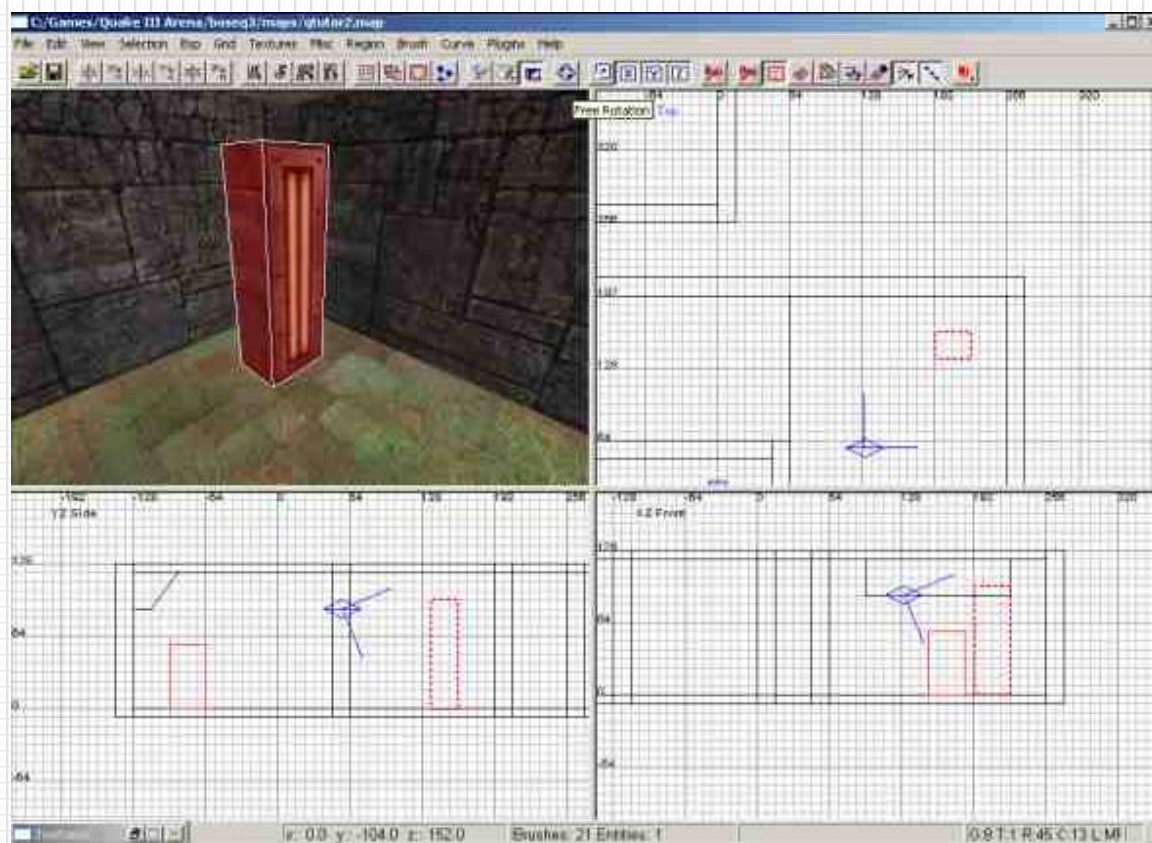


Diese Ecken habe ich dir hier gekennzeichnet. Also legen wir los. Dazu brauchen wir wieder einen Brush, den wir mit der Textur



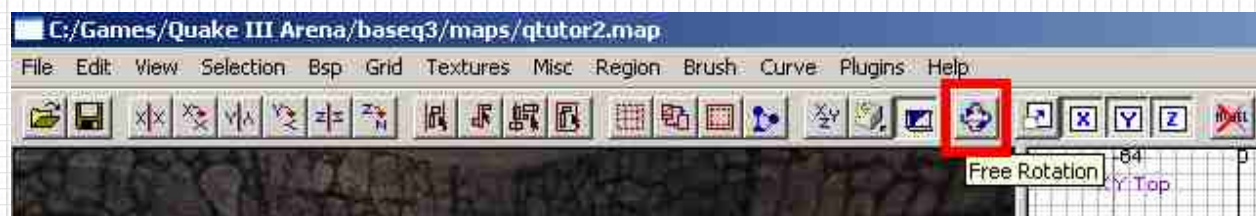
"gothic_block/blocks11b" belegen:

Nun belegen wir die Seite, die dem Player-Start zugewandt ist, mit unserer Lampen-Textur. Nun fällt dir sicher auf, dass der Brush viel zu klein für die Textur ist, also drückst du "S" um den Surface-Inspector zu öffnen, und wählst "FIT". Nun müsste die Textur sehr gut sitzen:



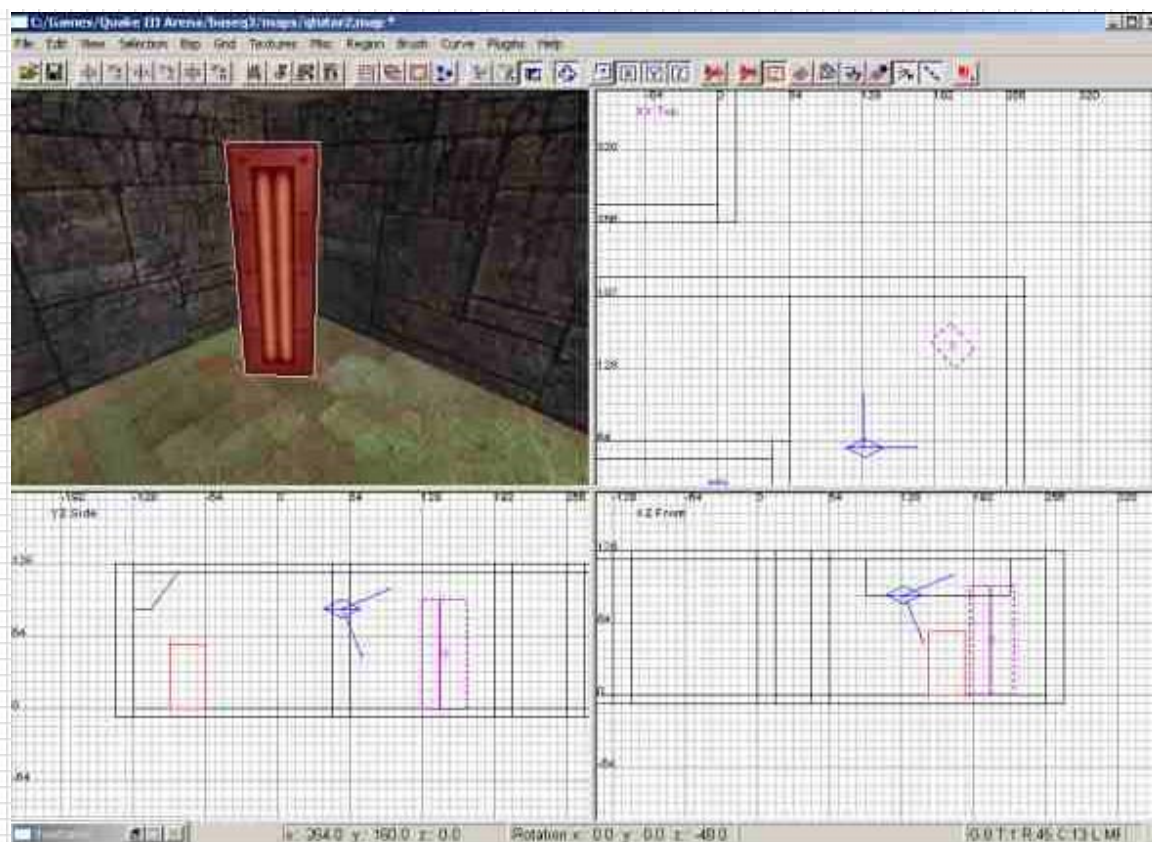
Nunja, eigentlich sind wir ja fertig, aber unsere Lampe steht etwas dumm in der Landschaft, wir sollten sie daher etwas drehen und mehr in die Ecke stellen.

Dazu deselektierst du ersteinmal alles. Dann drückst du in der Icon-Leiste den "free-rotate" Button:



Nun wechselst du in die Top Ansicht, da wir ja den Brush seitlich drehen wollen - und dies geht nunmal nur in dieser Ansicht. Der Brush wird jetzt komplett violett dargestellt - ich weiss nicht, ob damit aufmerksam gemacht werden soll, dass die "Free-Rotation"-Funktion angeschaltet ist, oder ob es nur ein Bug ist. Aber nur am Rande.

Nun klickst du ins Top Fenster und bewegst deine Maus langsam ein wenig nach oben und unten. Wie du siehst, bewegt sich der Brush nun frei, d.h. er passt auch nichtmehr in unser Grid:



Ich habe den Brush so gedreht, dass er nun ziemlich genau im 45°-Winkel zu den beiden Wänden im Hintergrund steht. Nun kannst du wieder die "ESC"-Taste drücken um den "Free-Rotation"-Modus zu verlassen. Jetzt musst du nochmals die "ESC"-Taste drücken um auch den Brush zu deselektieren. Nun musst du den Brush wieder anklicken, da wir ihn noch etwas in die Ecke setzen wollen.

Nun wollen wir ja noch die andere Ecke mit einer solchen Lampe ausrüsten, dazu kopierst du unsere Lampe und setzt sie ins andere Eck, das ich dir gekennzeichnet habe. Dann musst du die Lampe noch entsprechend drehen ;)

WICHTIG:

Du solltest diese Funktion nur für einzelne Brushes verwenden, da mehrere gedrehte Brushes in einer Map zu Darstellungsfehlern führen könnten. Also solltest du für solche Gruppen von Brushes besser gleich die 90° Drehung benutzen, da du somit auch Probleme vermeiden kannst.

[zurück zur Hauptseite](#)

183759



Lichtkegel:

Beispielmap: "qtutor3.map"
 Ergebnismap: "qtutor4.map"

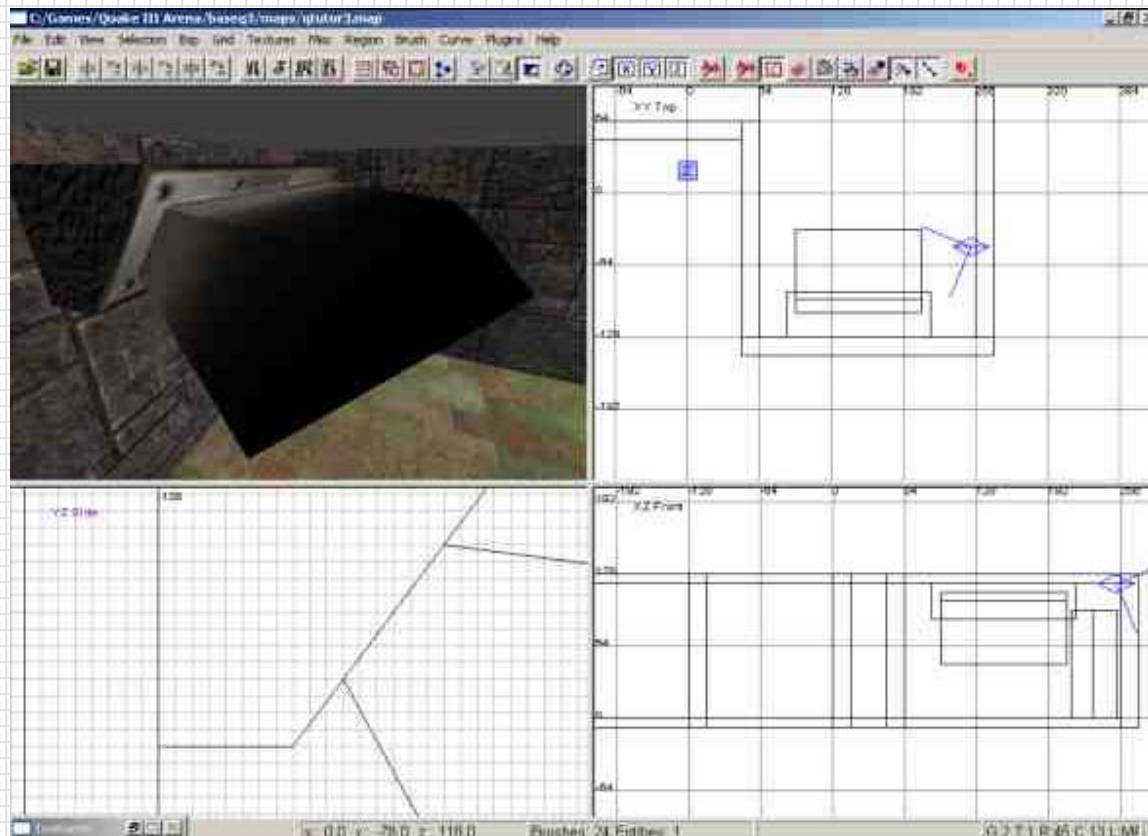
Nun, erstmal wirst du jetzt vielleicht denken, das haben wir doch schon in einem früheren Kapitel gehabt. Da ging es um "Corona" und wie diese eingesetzt wird. Hier will ich dir allerdings erklären, wie man einen eigenen Lichtkegel machen kann. Und das geht ganz leicht - mit einem Brush.

Diesen Tutor habe ich für Q3 geschrieben, da ich finde, dass diese "Lichttechnik" in RtCW eine sehr geringe Rolle spielt und auch nicht oft zum Einsatz kommt. Wenn du das aber trotzdem auch in RtCW haben willst, kannst du das einfach auch machen - allerdings musst du andere Texturen benutzen, die Texturen hier gibt es nicht in RtCW.

Wir wollen erstmal diesen Lichtkegel an der Lampe hinter dem Playerstart setzen. Dazu ist uns natürlich der Playerstart etwas im Weg. Deshalb selektierst du erstmal den Playerstart und drückst "H". Dann verschwindet dein Brush. Das ist die "Hide"-Funktion. Du kannst auch beliebig viele Brushes verstecken, bis du eine gute Sicht hast und gut arbeiten kannst. Um den/die Brushes wieder zu sehen, drückst du "SHIFT" und "H" - jetzt sind auch alle Brushes wieder da, egal wieviele du zuvor versteckt hast.

Nun brauchen wir zunächst wieder eine Textur. Wir benutzen die Textur "sfx/beam". An Beam-Texturen gibts ne ganze Menge, z.B. rote und blaue. Aber wir wollen hier mit dem einfachen "beam" arbeiten.

Dann ziehst du dir deinen Brush, wichtig ist, dass der Lichtkegel später genau so aussieht, wie du ihn jetzt baust. Mein Brush sieht jetzt so aus:



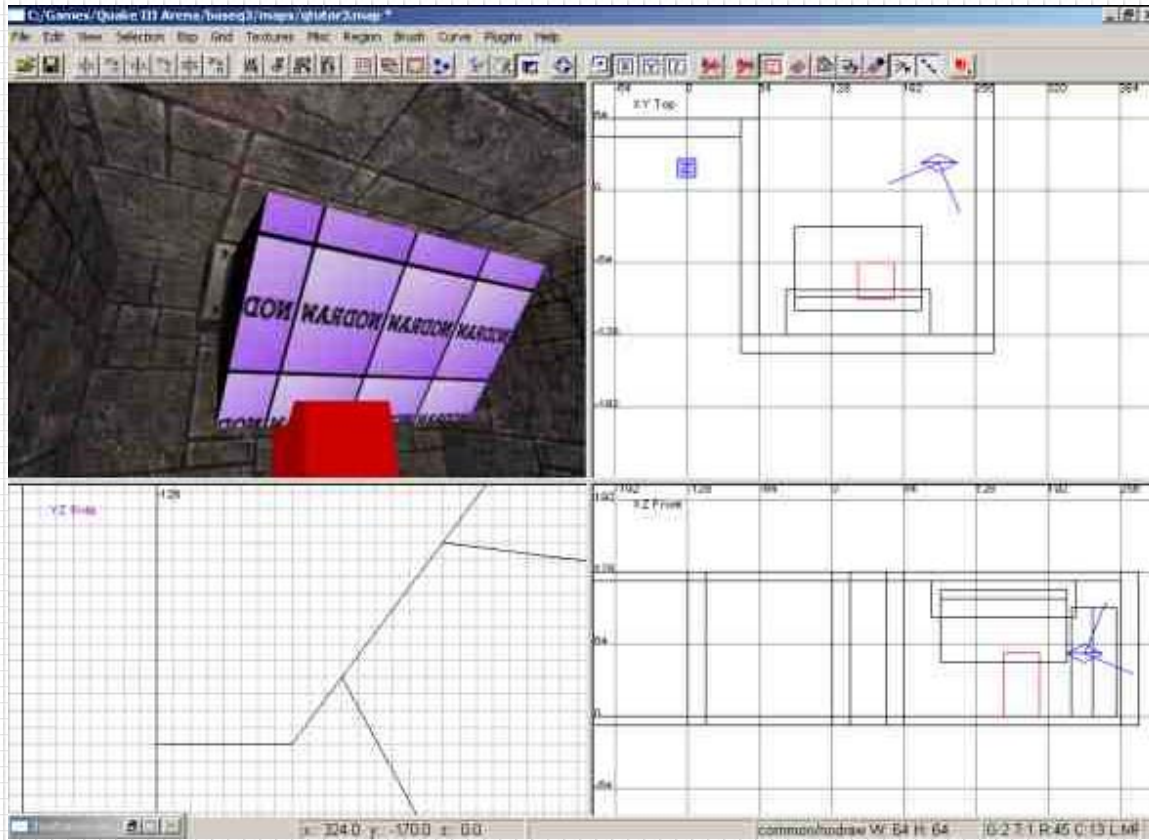
Wahrscheinlich musst du hier die Textur" etwas anpassen, besonders an den Seiten. Dazu musst du die Oberflächen einzeln selektieren (mit "SHIFT" + "STRG" + linke Maustaste) und dann "S" drücken, um den Surface Inspector aufzurufen. Hier musst du nun etwas mit dem Wert für "Rotate" herumspielen, da dieser Wert für den Seitenteil wichtig ist - schliesslich wollen wir ja, dass die Textur auch überall an der Lampe endet. Bei mir war das übrigens 50, aber das musst du halt selbst probieren.

Wenn du nun die Map compilieren würdest, würdest du feststellen, dass da die Lampe garnicht sichtbar ist. Dazu brauchen wir eine neue Textur. Sie heisst "nodraw" und befindet sich bei den "common"-Texturen.

ACHTUNG:

Die Texturen in diesem Ordner gibt es auch in RtCW, auch der Sinn ist der gleiche - d.h. dies ist auch für die RtCW-Mapper sehr wichtig.

Nun selektieren wir unsere Lampe und drücken die Taste "H" um den Brush zu verstecken. Diese Seite, die den Lampen-Brush berührt, musst du nun mit der "common/nodraw"-Textur belegen, schliesslich soll da das Licht ja durchscheinen. Dann drückst du wieder "SHIFT" + "H" um den Lampebrush wieder sichtbar zu machen. Nun musst du noch die grosse Seite (auf die man eigentlich sieht) mit der "common/nodraw"-Textur belegen:



So, das hätten wir geschafft !

[zurück zur Hauptseite](#)

183759

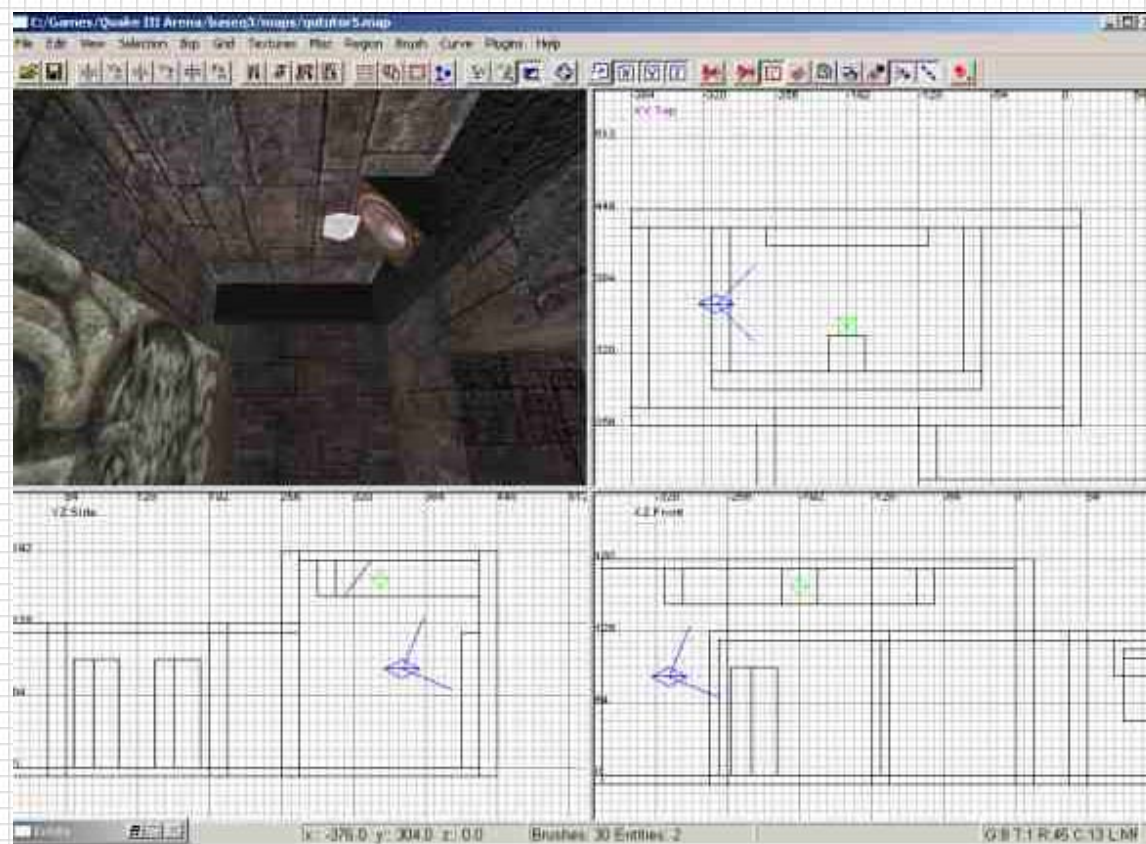


zusätzliche Lichteigenschaften:

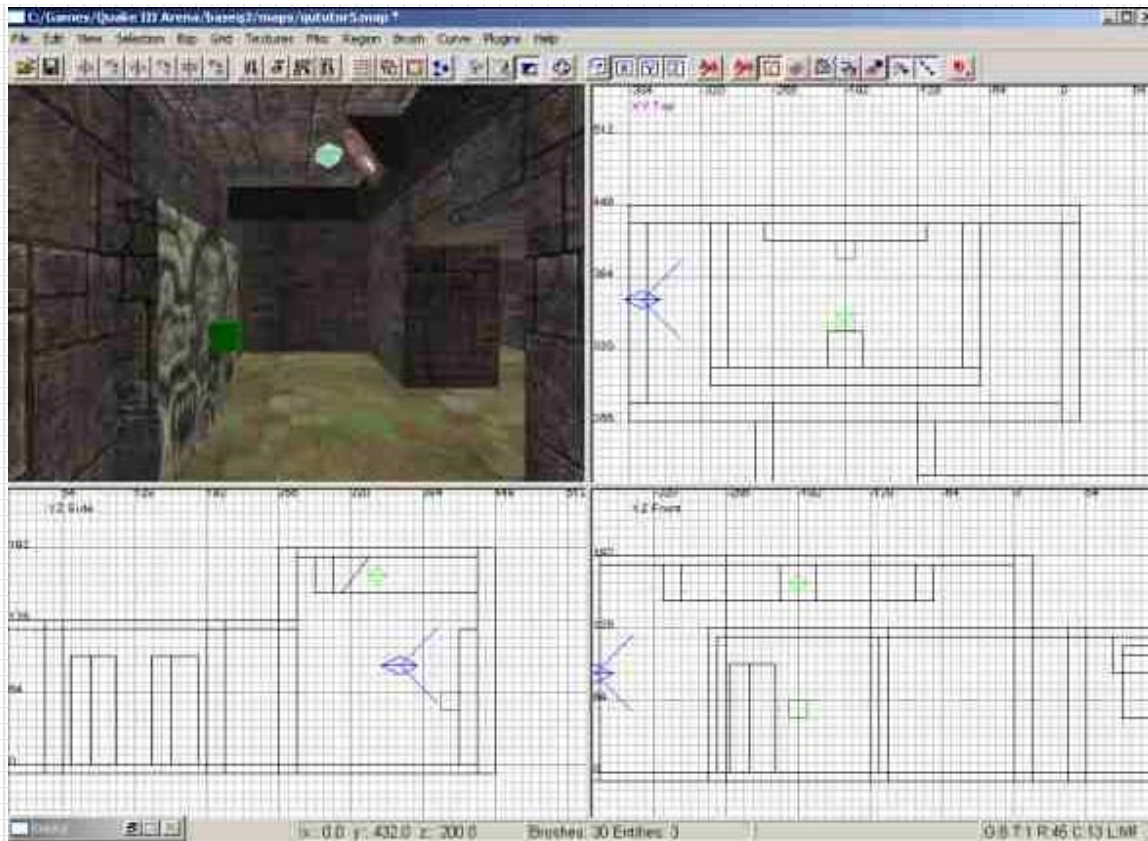
Beispielmap: "qtutor4.map"

Ergebnismap: "qtutor5.map"

Nun, ich habe in unserer Map in den zweiten Raum ein Wandbild gestellt - das wollen wir anstrahlen. Dazu habe ich den zweiten Raum etwas erhöht, und eine Art Strahler eingebaut - dieser soll dann das Bild anstrahlen. Nun klickst du in der Top Ansicht 2mal mit der rechten Maustaste und wählst "light". Es entsteht ein Light-Entity, das wir nun vor unseren Strahler setzen und gib dem Licht die Helligkeit 140:



Nun wollen wir natürlich das Licht farbig machen, dazu drücken wir "K" und wählen die Farbe aus. Ich habe mich für einen schwach-grünlichen Schein entschieden, da dieser gut zum Bild passt. So, jetzt brauchen wir natürlich noch ein Ziel - denn bisher ist unser Licht ja auch nur ein Licht. Dazu drückst du in der Top Ansicht zweimal mit der rechten Maustaste und wählst "info/info_null". Dieses Entity setzt du direkt vor das Bild:



So, ab jetzt gibt es zwei Möglichkeiten - eine einfache und eine komplizierte.. Und wir machen das wie in der Schule, erst das komplizierte ;)

komplizierte Möglichkeit:

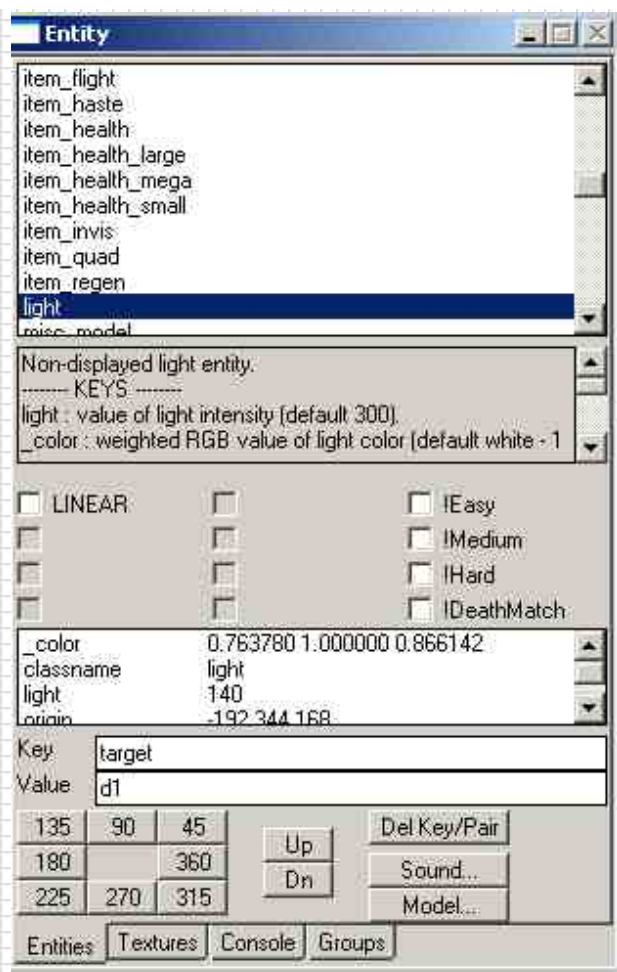
dazu deselektierst du erst einmal alles. Dann wählst du das Licht-Entity an und drückst "T" und gehst ganz unten auf "Entities". Nun hast du folgendes Bild vor dir:

Wie du siehst, habe ich dir ins Keyfeld schon den Befehl "target" eingetragen.

Als "Value" habe ich "d1" eingetragen. Das machst du jetzt auch nach und drückst dann "ENTER"

Und was bringt das?

Damit haben wir unser Licht nun ein Ziel (eng.: Target) angegeben. Nun brauchen wir natürlich noch das Ziel.

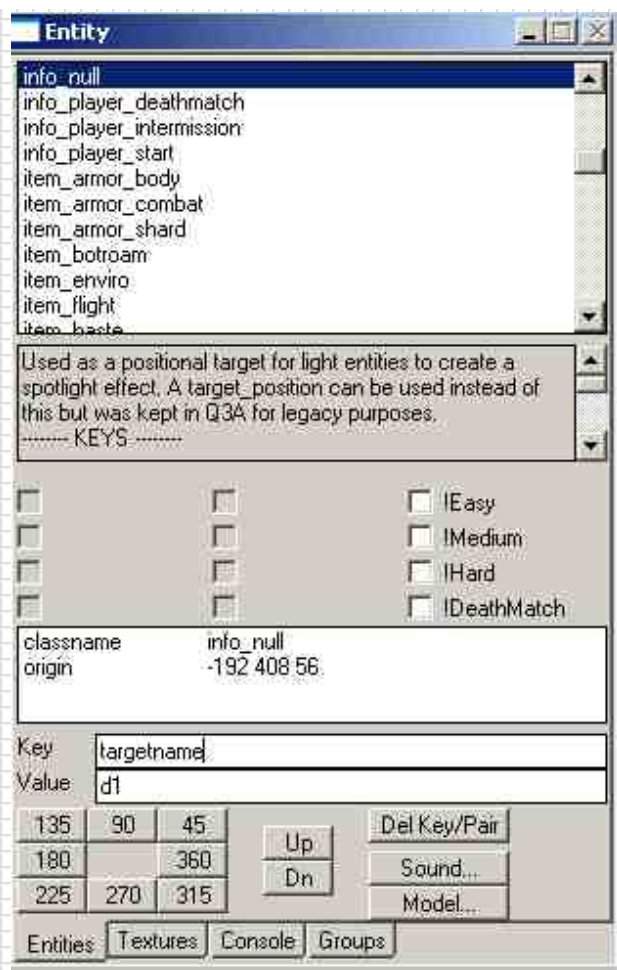


Das Ziel ist natürlich unser "info_null"-entity. Also selektierst du das Entity und gehst wieder in das Entity-Fenster (erst "T" drücken, dann ganz unten auf "Entities" gehen). Diesmal sieht es so aus:

Hier gibst du im Keyfeld "targetname" ein und als "Value" gibst du "d1" ein. Anschließend drückst du "ENTER". Nun deselektierst du alles.

Was bringt das?

Nun haben wir dem Licht ein Ziel angeboten, d.h. da die "Value" gleich ist. Nun zieht das Licht also auf das "Null-Entity"



Wenn du alles richtig gemacht hast, kannst du jetzt einen Strich sehen, der das Licht mit dem Null-Entity verbindet:



diese komplizierte Möglichkeit verwendest du bestimmt nie wieder - aber es hilft, den Sinn hinter der Sache zu verstehen und nun kennst du dich auch besser mit den "Keys" und "Values" aus. Und wie man sie löscht, natürlich auch.

einfache Möglichkeit:

dazu musst du erstmal diese Eingaben im Entity-Fenster rückgängig machen - d.h. vom Licht und vom "Info_Null"-Entity musst du

den Key "target" bzw. "targetname" löschen. Dann drückst du "ESC" um alles zu deselektieren.

Nun sind wir so wieder soweit, dass wir ein normales Licht haben - und eben den "info_Null"-Entity. Nun klickst du das Licht an (mit "SHIFT" + linke Maustaste) und klickst danach das "Info_Null"-Entity ebenfalls an (mit "SHIFT" + linke Maustaste). Nun müssten beide markiert sein. Nun drückst du "STRG" + "K". Nun siehst du wieder den Strich, der die beiden Entites verbindet.

WICHTIG: Du musst darauf achten, dass du die richtige Reihenfolge einhältst, sonst funktioniert dein Licht später im Spiel nicht. Hast du die Entities falsch verknüpft, musst du die Key-Einträge wieder löschen und neu verknüpfen. Normal lautet hier die Devise: Immer erst das Hauptentity anwählen, dann das Zusatzentity.

[zurück zur Hauptseite](#)

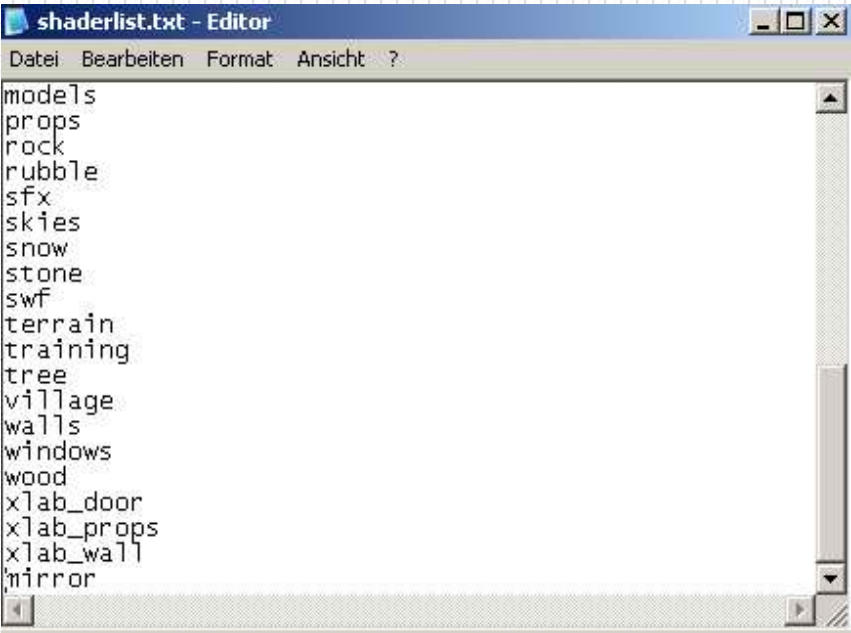
183759



einen Spiegel bauen:

Verwendete Map: "tutor14.map"
Ergebnis-Map: "tutor15.map"
Spiegel-Textur und Shader: "[Zip-Datei](#)"

Dieses Mal beginnt unsere Arbeit allerdings mit dem Einbinden des Shaders und der Spiegeltextur. Dazu öffnest du die Datei "shaderlist.txt" mit einem Texteditor, z.B. Wordpad. Hier gehst du bis zum Ende der Datei und schreibst ganz unten "mirror" hinein (ohne die Anführungszeichen). So sollte es dann ungefähr aussehen:

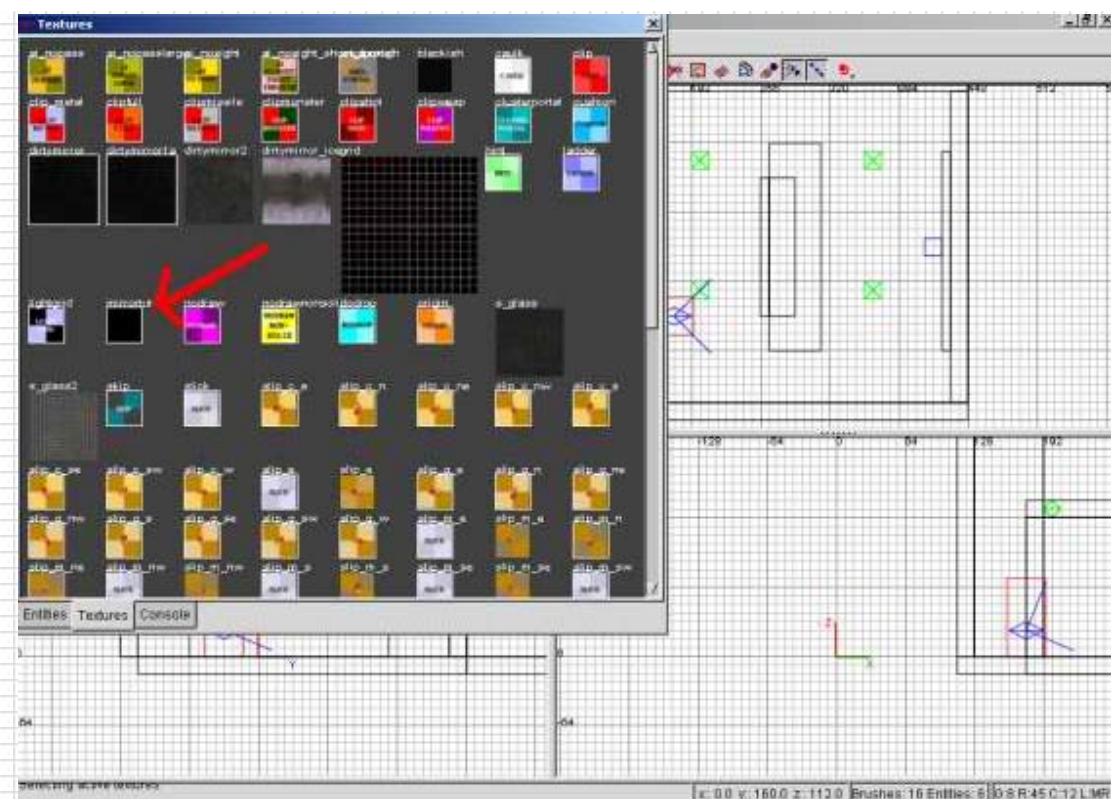


Und was bringt uns das? Ganz einfach, der Effekt für den Spiegel wird über einen Shader gesteuert, den wir hiermit geladen haben.

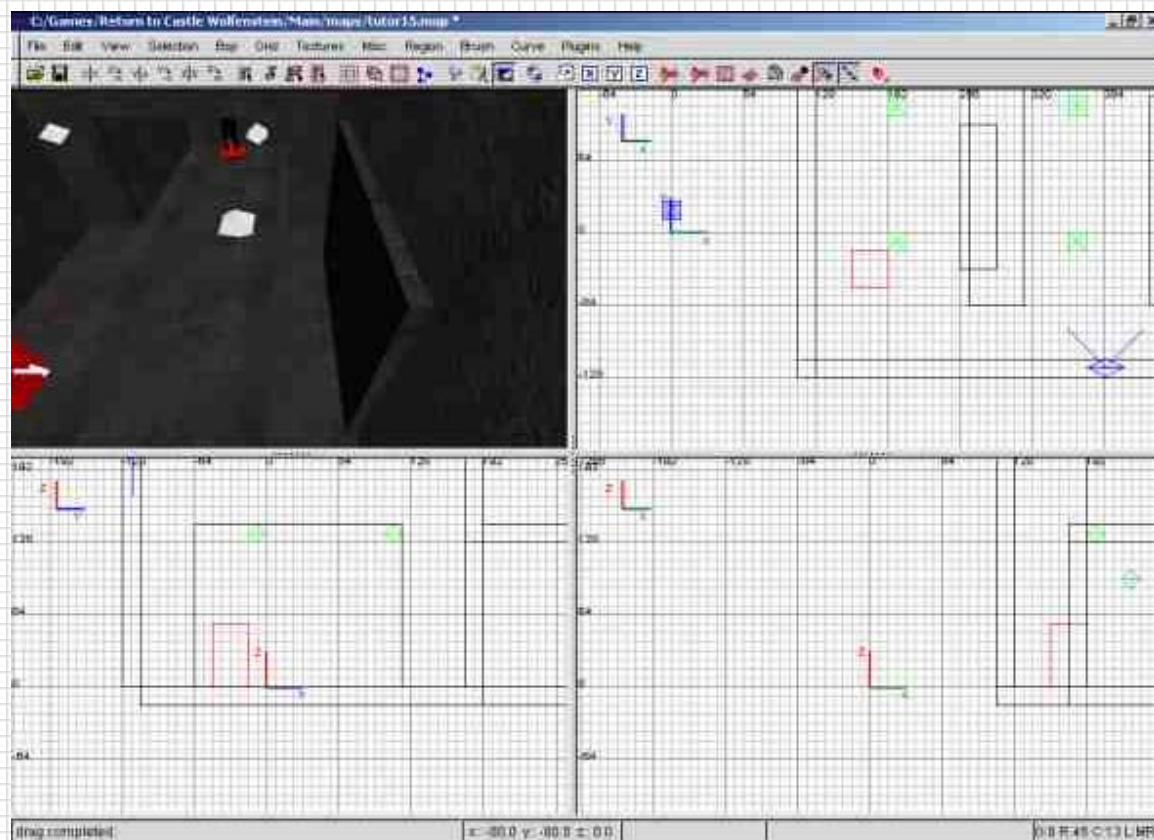
Nun benötigen wir noch den Shader selbst und die Textur, die wir für den Spiegel selbst brauchen. Dazu öffnest du diese [Zip-Datei](#), in dieser befinden sich 2 Dateien ("mirrortut.tga" und die Datei "mirror.shader"). Jetzt werden wir diese beiden Dateien so einsetzen, dass der Radiant auch damit umgehen kann. Dazu erstellst du im "Main/Textures" - Ordner einen neuen Ordner, den du "common" nennst. Hier kopierst du nun die Textur "mirrortut.tga" hinein.

Nun kopierst du die Datei "mirror.shader" in das Verzeichniss "Main/Scripts". Damit haben wir die gesamte Vorarbeit geleistet - wir haben den Shader und die Textur "importiert" und den Shader durch den Eintrag aktiviert. Nun können wir den Radiant laden.

Wie du siehst, habe ich dir in diese Map schon an der Wand einen Brush gesetzt, der später unser Spiegel darstellen soll. Nun wollen wir zunächst prüfen, ob der Radiant auch unsere Textur gefunden hat. Dazu lädst du nun die "common"-Texturen und drückst "T" um das Texturen-Fenster zu öffnen:

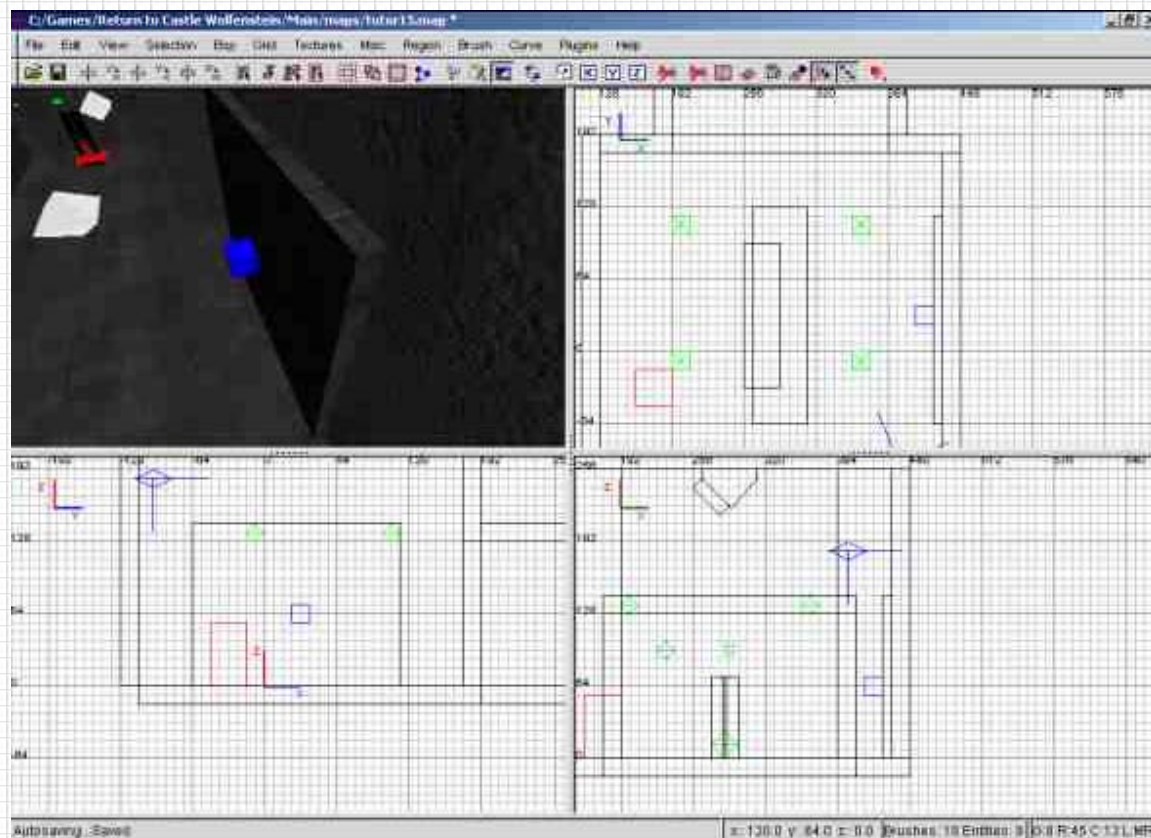


Nun klickst du die Vorderseite des Wandbrushes an (wir wollen NUR aus der Vorderseite einen Spiegel machen). Nochmal zur Erinnerung, dazu drückst du ("STRG" + "SHIFT" + linke Maustaste) und wählst die Textur "common/mirrortut" aus:



Nun sind wir schon fast fertig. Das einzige, was wir noch brauchen, ist ein "misc_portal_surface". Dazu gehst du mit der Maus in die Top Ansicht und drückst 2x die rechte Maustaste. Dann wählst du aus dem Menü "misc" das "misc_portal_surface". Nun erscheint eine blaue Box, die du nun vor deinen Spiegel stellst. Dabei ist nur zu beachten, dass die Box näher als 64 Units und mittig vor deinem

Spiegel sitzt:



So, das war es schon.

WICHTIG: Beim Bau eines Spiegels gibt es zwei Dinge zu beachten:

- das "misc_portal_surface" muss näher als 64 Units am Spiegel sitzen (am besten vor der Mitte des Brushes)
- nur eine Seite des Brushes darf mit der Spiegel-Textur belegt sein, da du sonst einen HOM-Effekt oder einen "Mirror-Universe"-Effekt bekommst. HOM steht für Hall Of Mirrors und bedeutet das gleiche wie Mirror Universe, nämlich dass die Spiegel sich nicht gegenüber stehen dürfen, da sie sich sonst immer und immer wieder selbst spiegeln.

Und so sieht es dann im Spiel aus:



Wie du siehst, verdoppelt der Spiegel alle Brushes, die er spiegelt. So entsteht eine enorme Rechenleistung, die besonders bei grossen gespiegelten Flächen auch schonmal die FPS in den Keller jagt. Hier solltest du genau überlegen, wo du Spiegel einsetzt.

[zurück zur Hauptseite](#)

183759



Modelle einbauen:

Verwendete Map: "tutor16.map"

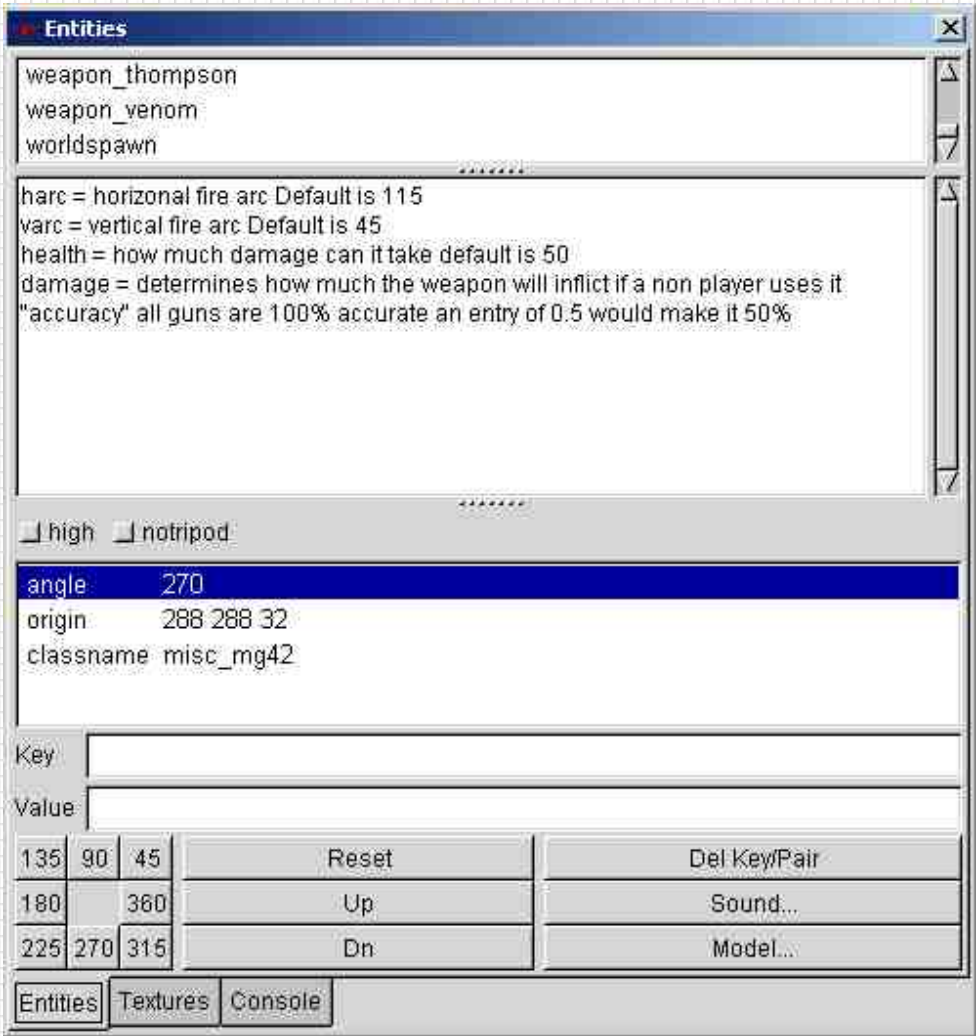
Ergebnis-Map: "tutor17.map"

Hier wollen wir lernen, wie man "Models" in eine Map setzt. Erstmal was zu den "Models" allgemein. Models zählen nicht zu den Brushes, da sie nicht mit dem Radiant erstellt wurden, sondern mit einem externen Programm, z.B. 3D Studio MAX. In RtCW und Q3 gibt es nun ein Entity, dass uns den Einbau der Modelle erleichtert.

Dazu klickst du in der Top-Ansicht 2x mit der rechten Maustaste, aus dem erscheinenden Menü wählen wir den Punkt "misc" und dort "misc_mg42". Hier gibt es schon etwas besonderes zu bemerken:

Im Gegensatz zu den anderen Models kann man durch die MG42 nicht hindurchlaufen

Nun erscheint in der Map ein roter Kasten, der unser MG42 darstellt. Leider sieht man bei manchen Models nur die Umrandung, also einen Würfel. In unserem Fall sehen wir nun einen roten Würfel. Da wir auch nicht sehen können, welche Richtung "vorne" ist. Deshalb hab ich dir [hier](#) nochmal den Link angegeben, wo du nachlesen kannst, wie man die Richtung der MG42 festlegen kann. Wenn du nun das Entity-Fenster öffnest (geht über "T" und da unten im Fenster auf Entities klicken, dabei muss aber die MG42 selektiert sein), erscheint dieses Bild:



- Varc = vertikaler Bewegungsradius
- Harc= horizontaler Bewegungsradius
- "Accurady" = Präzision (Zielgenauigkeit) der MG
- "Health" = gibt den Wert an, wieviel Schaden das MG aushält
- "damage" = gibt den Wert an, wieviel Schaden ein Nicht-Spieler mit der MG anrichten kann

Über dem Feld siehst du noch zwei andere Felder:

- high: die Bedeutung weiss ich leider noch nicht.
- notripod: der Dreifuss des MGs wird abgeschalten, d.h. du kannst dir deinen eigenen Untergrund für die MG bauen.

also gibst du jeweils ein:

Key: Varc
Value: 100

Key: Harc:
Value:180

Das bedeutet, der vertikale Bewegungsradius beträgt 100°, der horizontale Bewegungsradius beträgt 180°. Die anderen Keys kannst du weglassen, hier gelten nämlich jeweils die Standarts.

Nun bauen wir uns mal einen Ritter, den man später "kaputt" machen kann. Dazu benötigen wir auch erst ein "misc_model"-entity. Dazu drückst du im Top-Fenster 2x mit der rechten Maustaste und wählst "misc_model". Es erscheint ein neues Fenster. Hier öffnest du den Ordner "mapobjects" und öffnest hier den Ordner "Knight". Hier siehst du mehrere Versionen von Rittern -z.B. "knight.md3" oder "knight_anim.md3". "knight.md3" besitzt keine Animation, d.h. er ist auch nicht zerstörbar. Daher benutzen wir den "knight_anim.md3". Also klicken wir ihn 2x mit der linken Maustaste an - das Model erscheint in unserer Map. Jetzt musst du noch ein

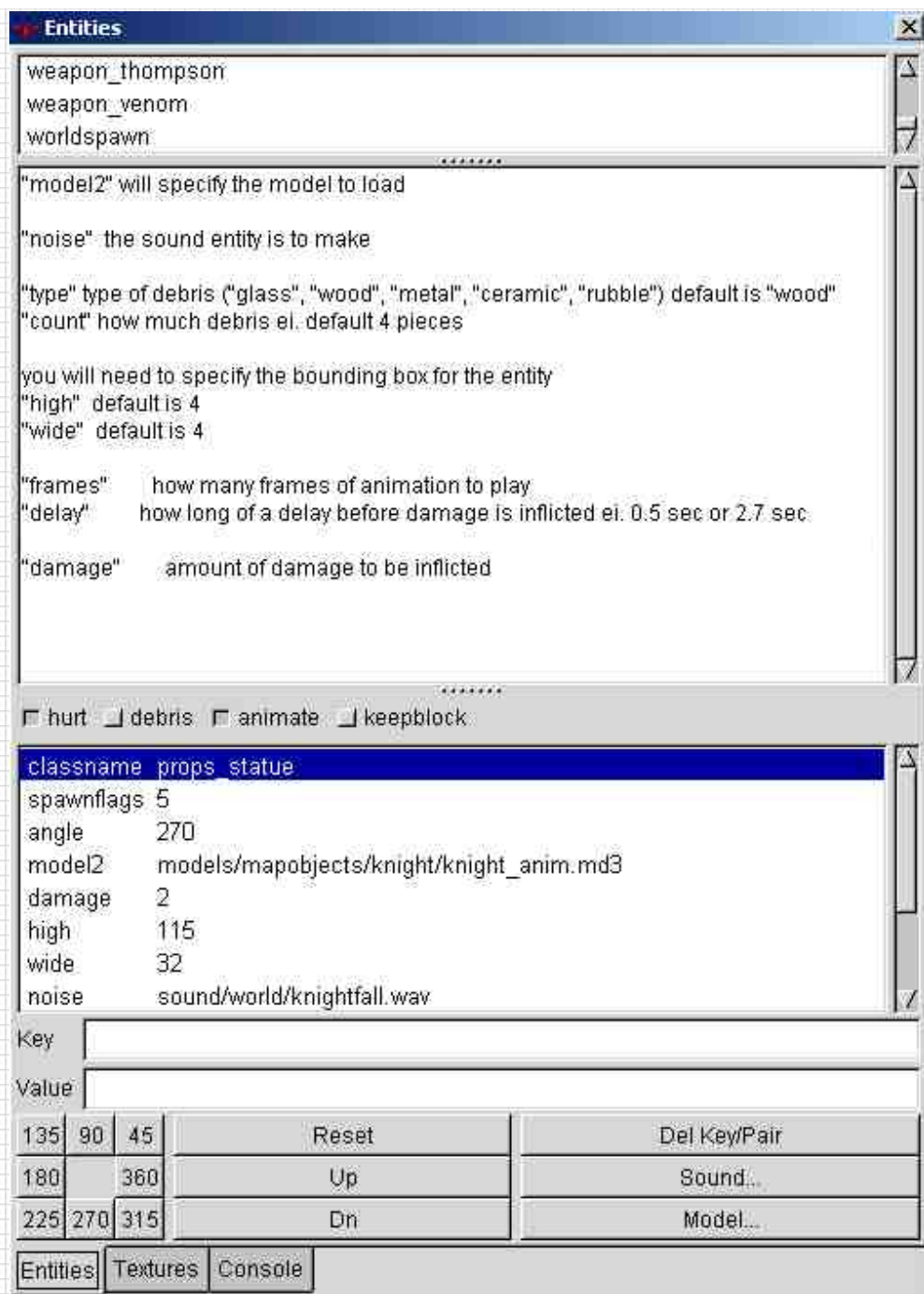
Plätzchen für das gute Stück finden. Ich habe ihn gegenüber der MG42 gestellt, so dass wir ihn auch gleich abschiessen können ;) Ich habe gleich das Entity-Fenster geöffnet, da wir noch ein paar Dinge verändern müssen, dass unser "Blechkamerad" auch das tut, was wir wollen.

Wenn du den Ritter angeklickt hast, musst du noch "hurt" und "animate" (das siehst du in der Bildmitte) aktivieren.

Nun gebe ich dir die Keys und Values an, die du eingeben musst:

- key: noise
value: sound/world/knightfall.wav
- key: frames
value: 39
- key: delay
value: 0.5
- key: damage
value: 2
- key: health
value: 15
- key: model2
value: models/mapobjects/knight/knight_anim.md3
- key: classname
value: props_statue

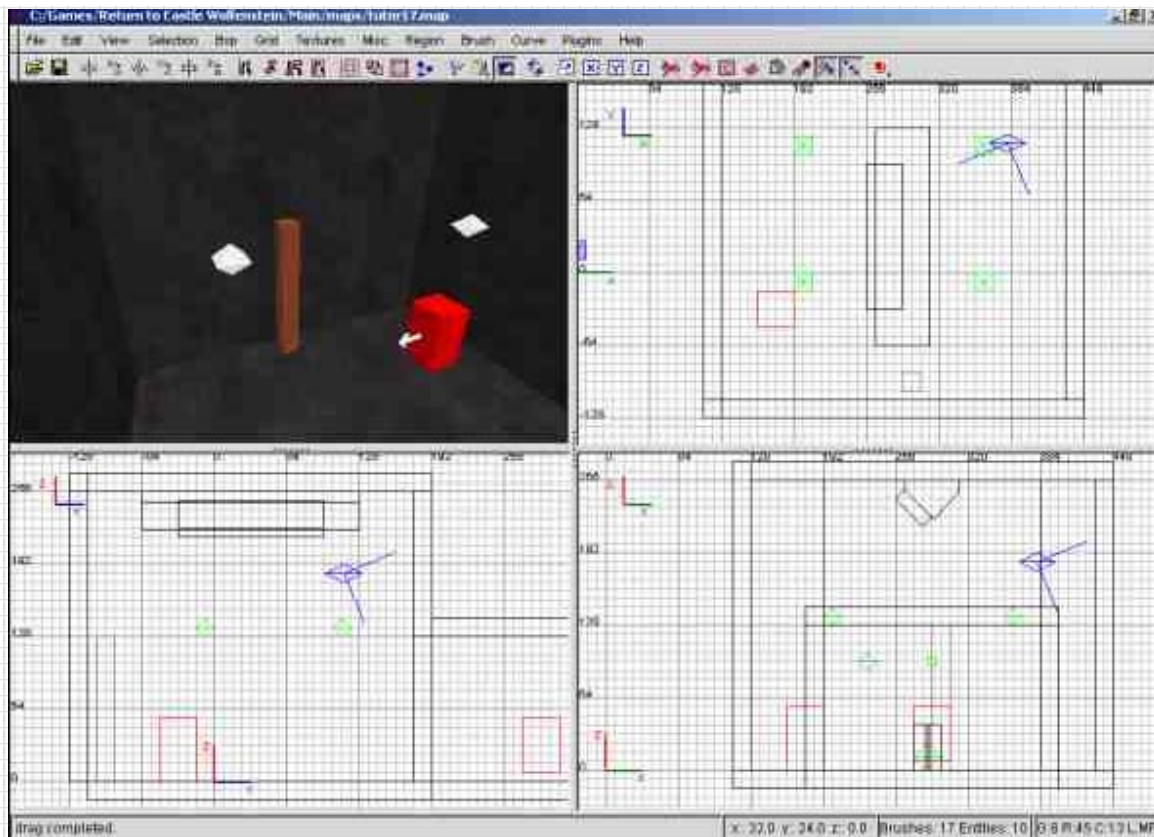
Dazu ein Bild:



Nun hast du also ein Model, dass:

- wenn es umfällt, der Ton "knightfall.wav" abgespielt wird
- die Animation für das Umfallen wird in 39 Bildern dargestellt
- Die Zeit, bis der Ritter umfällt, beträgt eine halbe Sekunde (0.5 sekunden)
- der Schaden, den der Ritter beim Spieler anrichtet, ist 2
- Der Ritter kann nur umgeworfen werden

Das Endergebnis siehst du hier:



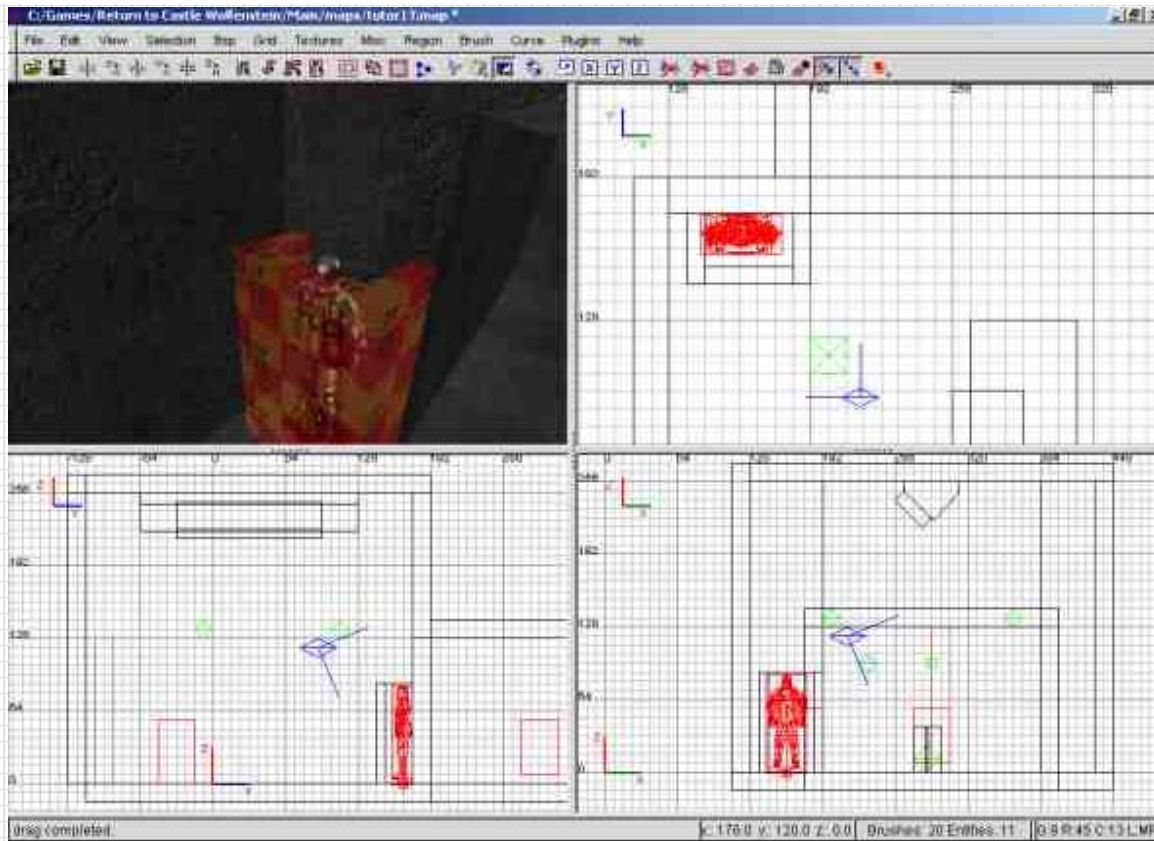
das du nun für den Ritter nur noch einen braunen Balken siehst, ist normal ;) also keine Panik

Nun wollen wir noch ein ganz normales Model machen - dazu verwenden wir einfach einen Ritter, der nicht umfallen kann. Dazu klickst du in der Top Ansicht 2x die rechte Maustaste und wählst "misc", dann "misc_model". Dann öffnet sich wieder das Auswahlfenster. Du öffnest den Ordner "mapobjects" und dann öffnest du den Ordner "Knight". Du klickst 2x mit der linken Maustaste auf "knight.md3". Jetzt suchst du dir wieder ein schickes Plätzlein für diesen Ritter und achtest darauf, dass er weder in der Wand noch in den Boden ragt. Dann musst du ihn noch richtig drehen. Wenn du nichtmehr weisst, wie das geht, guck einfach mal [hier](#) nach.

Würdest du nun diese Map kompilieren, würdest du feststellen, dass du durch den Ritter hindurchlaufen kannst. Das wollen wir natürlich nicht.

Dazu müssen wir wieder etwas Vorarbeit leisten. Du klickst in der Menüleiste auf "View" und dort auf "Filter". Nun klickst du auf "Clips". Damit werden nun auch die "Clip"-Brushs in der Map dargestellt. "Clip"-Brushs sind einfache Brushs, die mit der Textur "Clip" belegt sind. Nun wählen wir diese Textur an. Du wählst in der Menüleiste den Punkte "Textures" und wählst "common". Hier suchst du dir die "clipfull"-Textur und klickst sie an.

Jetzt machst du um deinen Ritter eine Art Käfig aus Brushs, die jetzt mit der "common/clipfull"-Textur belegt sind:



So, hier habe ich dir mal den Ritter komplett mit "clipfull"-Bruhs zugebaut.

Nun erkläre ich dir am besten, wozu man diese "Clip"-Textur noch brauchen kann. Damit kannst du z.B. alle Stellen abschrägen, an denen der Player hängen bleiben kann. Clip-Brushs sind unsichtbar, jedoch kann man nichtmehr hindurchlaufen. So kannst du auch komplette und besonders komplexe Architekturen in solche Clip-Brushs einpacken, damit erfolgt eine schnellere Berechnung - jedoch werden diese Stellen unerreichbar für den Player. Mehr will ich dir noch nicht über diese Textur sagen, mehr findest du in dem Thema "die Common-Texturen". So solltest du auch z.B. das Feuer in einen Clip-Brush einpacken, da so mehr Realismus aufkommt, wenn du nichtmehr durch das Feuer laufen kannst :)

[zurück zur Hauptseite](#)

183759

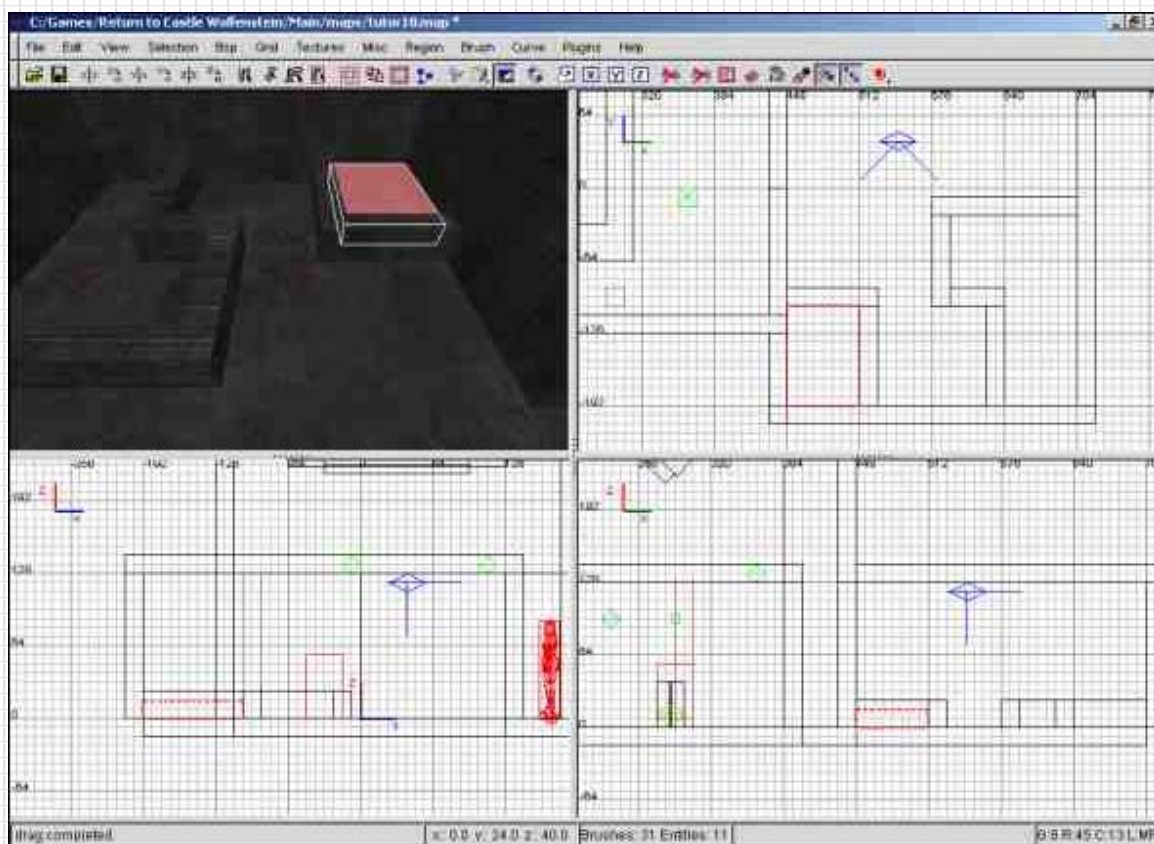


Flüssigkeiten:

verwendete Map: "tutor18.map"

Ergebnis-Map: "tutor19.map"

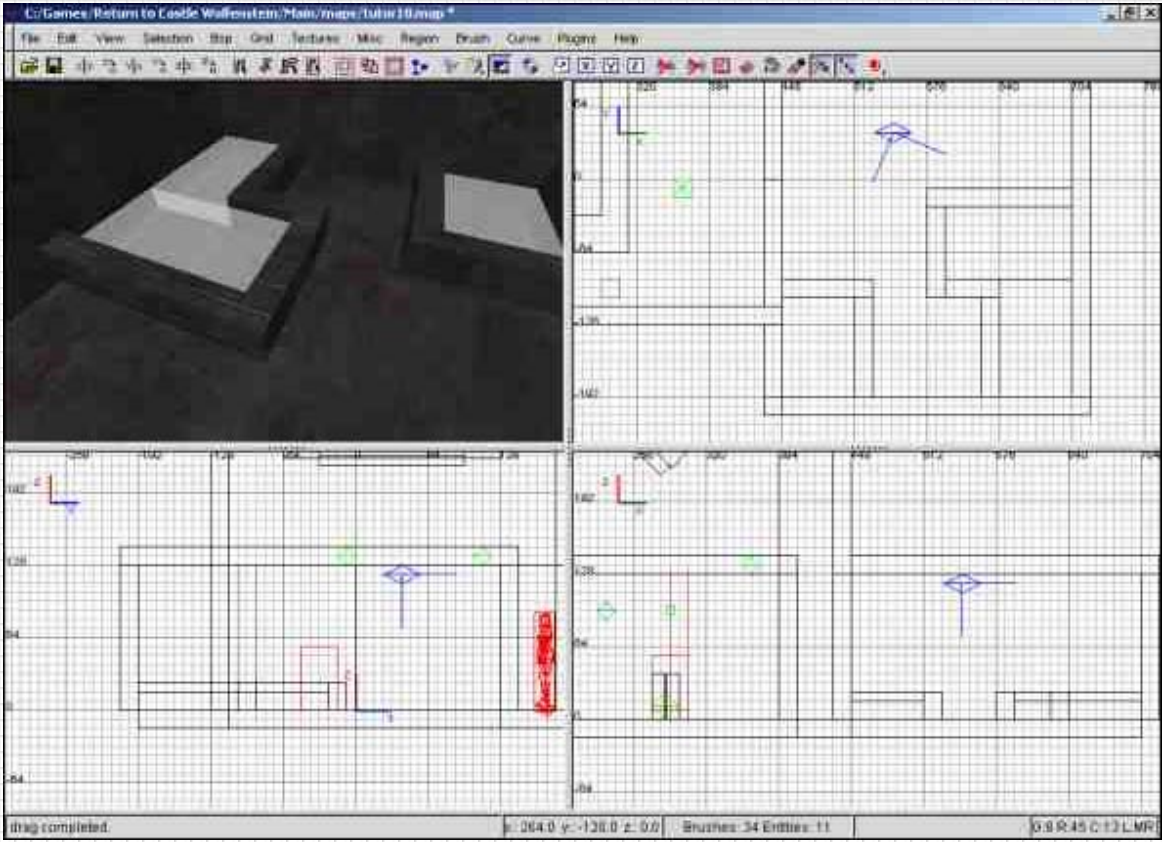
Wie ich schon gesagt habe, besitzen alle Wasser-Texturen einen Shader, d.h. sie haben einen weißen Rand. Wie du siehst, habe ich der Map einen weiteren Raum hinzugefügt. Wir wählen uns nun eine Wasser-Textur und legen einen Wasserbrush an, den wir in die Becken setzen, die ich schon erstellt habe. Dazu benutzen wir die Textur "liquids/water_beach_fast". So könnte dann das Wasserbecken aussehen:



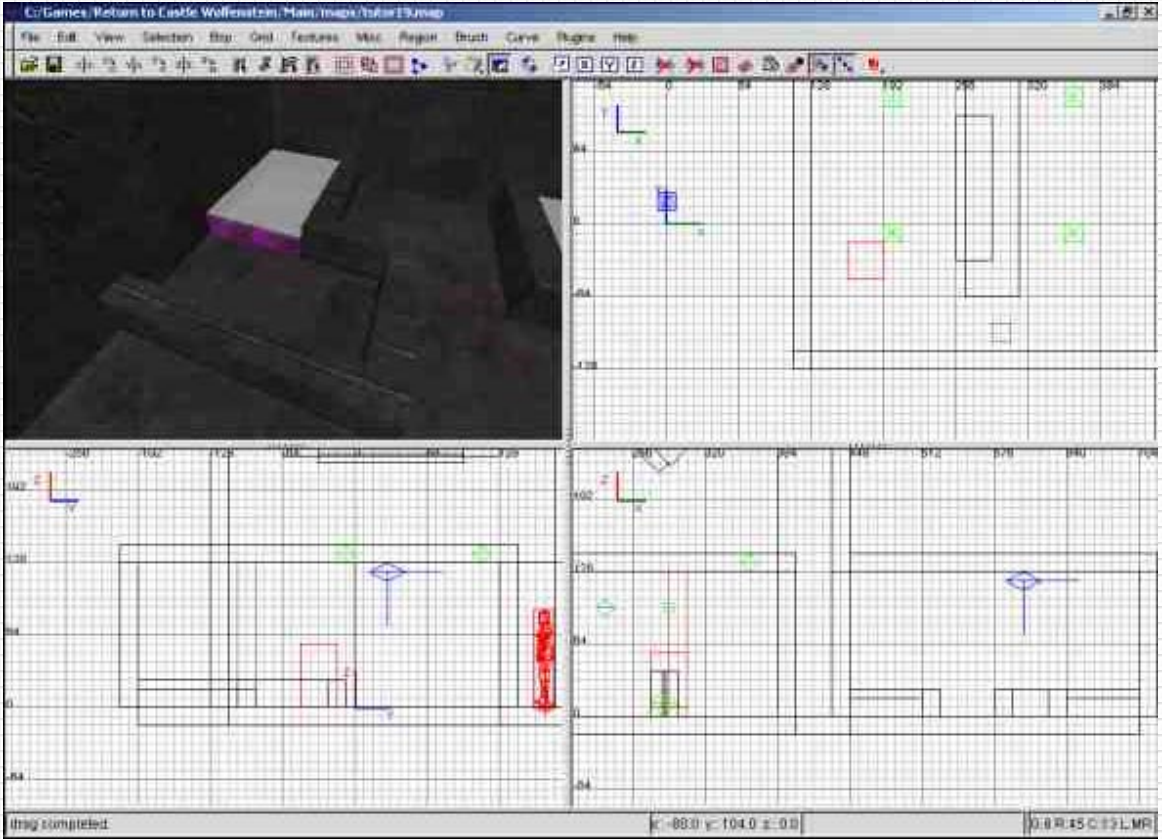
Du solltest in der Draufsicht darauf achten, dass der Wasser-Brush nicht bis an den Beckenrand reicht, sondern etwas darunter liegt, da es so realistischer ist.

WICHTIG: Wenn du später eine eigene Wasser-Textur benutzen willst, solltest du darauf achten, dass die Textur einen weißen Rand hat, also einen Shader besitzt. Besitzt sie keinen Shader, wäre es kein Wasser mehr, sondern eine ganz normale solide Textur.

Beim zweiten Becken wird die Sache nicht ganz so einfach. Wir haben hier einen Knick in dem Becken - also benutzen wir zwei Wasser-Brushs:



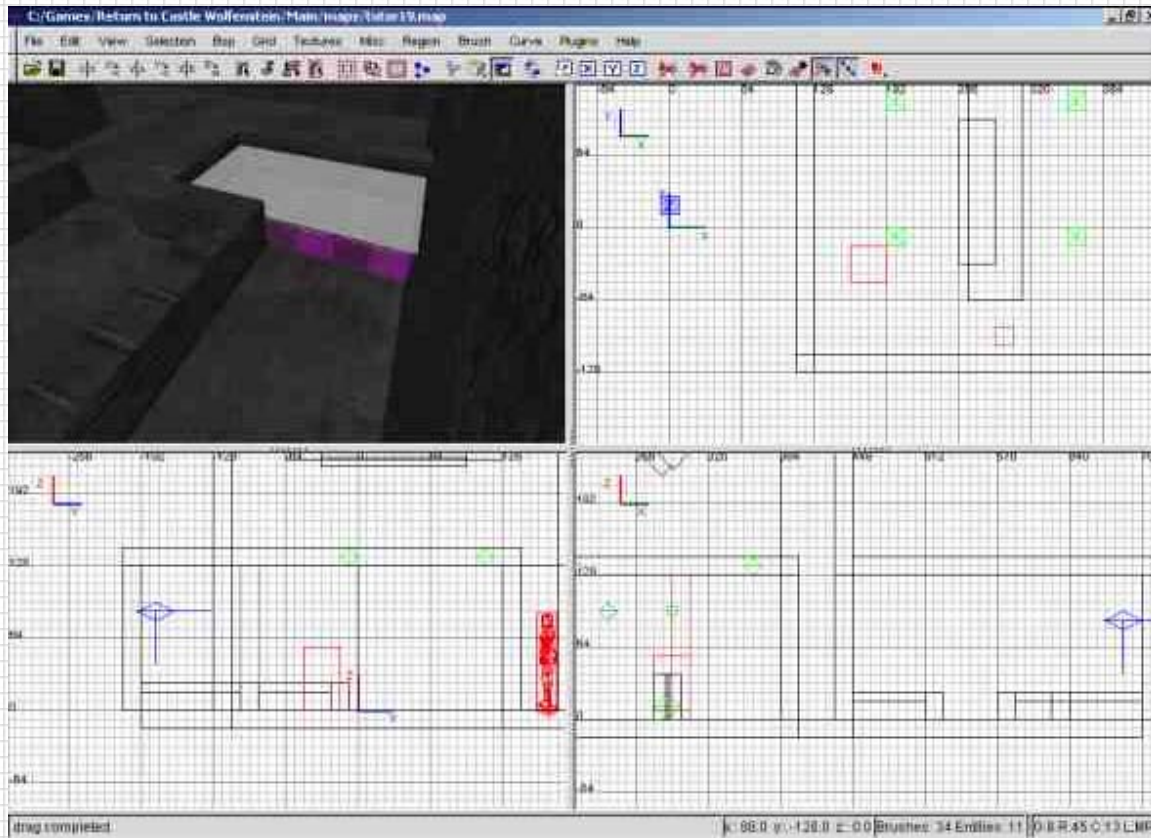
Würdest du nun diese Maß compilieren, würdest du im Spiel eine schwarze Fläche erkennen, die zwischen den beiden Brushs ist. Dies ist die Schnittfläche der beiden Brushs - und diese müssen wir verschwinden lassen. Dazu wählen wir zunächst den grösseren Wasser-Brush an und drücken "H", um diesen Brush zu verstecken:



Wie du siehst, habe ich hier die Seite, die den anderen (jetzt versteckten) Brush berührt, mit der Textur "nodraw" belegt. Das tun wir jetzt auch. Dazu wählst du nun diese Seite des Brushs aus (mit "SHIFT" + "STRG" + linke Maustaste) und wählst im Menüpunkt

"Textures" "common" die Textur "common/nodraw".

Jetzt holen wir uns den versteckten Brush mit "SHIFT" und "H" wieder zurück und verstecken nun den anderen Brush mit "H". Dann wählen wir wieder die Seite aus, die den versteckten Brush berührt (mit "SHIFT" + "STRG" + linke Maustaste):



Natürlich musst du darauf achten, dass du alle Schneideflächen mit der Textur "nodraw" belegst, sonst bekommst du immer diese schwarzen Schnittflächen.

ACHTUNG: Wasser und andere Flüssigkeiten ziehen die Performance nach unten, also solltest du darauf achten, wieviel Wasser du benutzt.

[zurück zur Hauptseite](#)

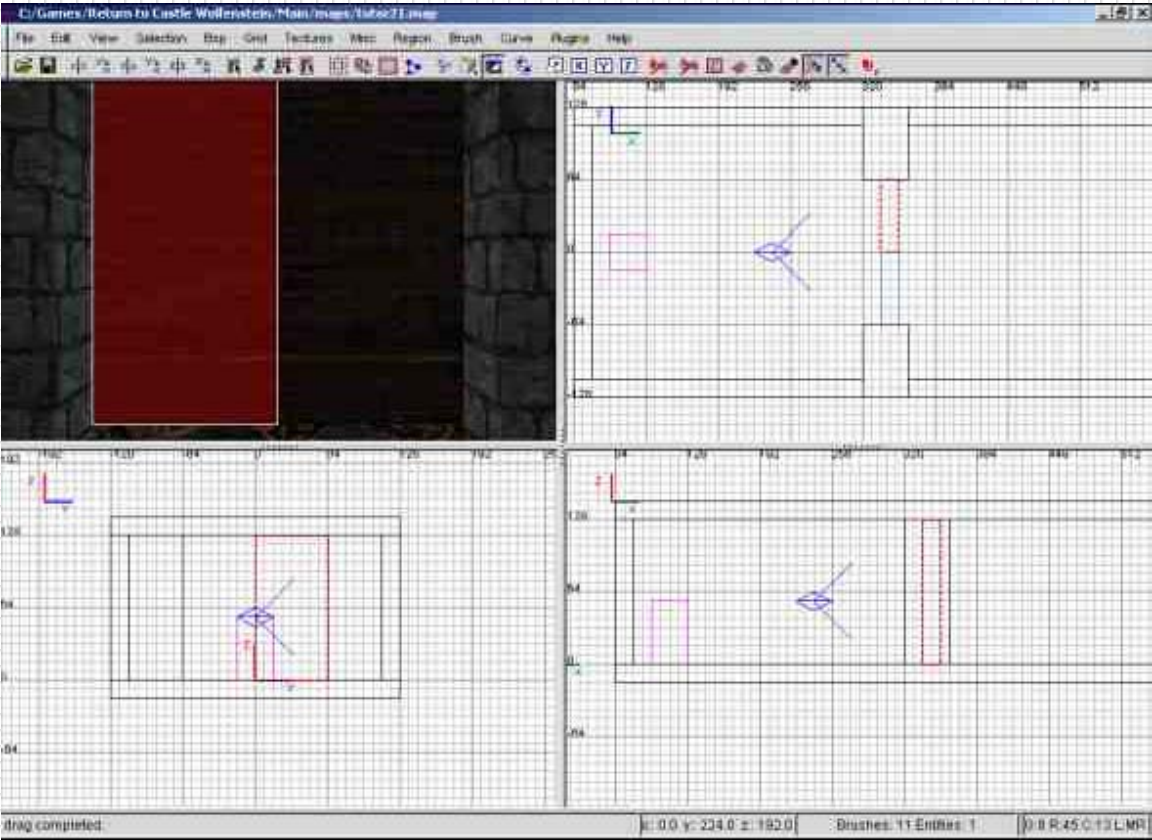
183759



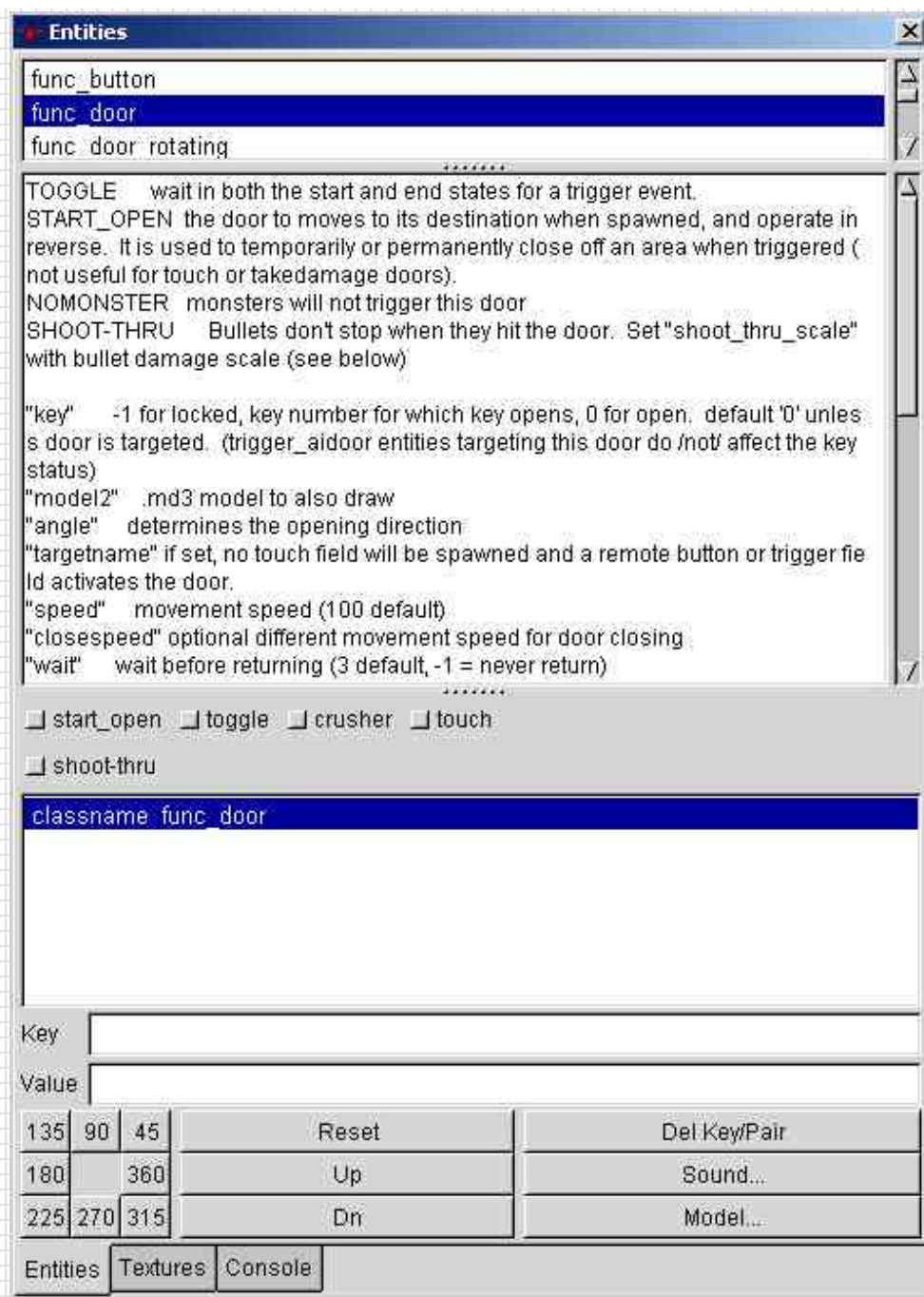
Türen:

verwendete Map: "tutor20.map"
Ergebnismap: "tutor21.map"

Also, hier haben wir das Ziel, eine Tür zu bauen, die dann die beiden Räume voneinander trennt. Dazu brauchen wir natürlich erstmal eine Tür-Textur. Dazu wählst du im Menüfenster "Textures" und da "castle" und dann "castle_door". Hier hast du nun eine Menge Auswahl. Ich habe mich für die Textur "castle_door/door_c12" entschieden. Wir wollen eine Tür bauen, die aus 2 Türhälften besteht, deshalb machst du jetzt einen Brush, der genau 64 Units breit ist. Jetzt selektierst du die eine Türhälfte. Nun drückst du 2x mit der rechten Maustaste in die Top Ansicht. Hier wählst du "function" und als Unterpunkt "function_door". Jetzt deselektierst du deine Türhälfte und machst es mit der anderen Türhälfte genauso. Jetzt sollte es ungefähr so aussehen:



Wie du siehst, sind jetzt die Tür-Brush nichtmehr schwarz, sondern blau. So, nun müssen wir natürlich noch dem Radi sagen, in welche Richtung usw. unsere Türe aufgehen soll, dazu drückst du die Taste "T" und gehst ganz unten auf "entities". Oder du drückst einfach die Taste "N". Dazu darf natürlich nur eine deiner beiden Türhälften selektiert sein:



Bevor ich dir die ganzen Keys und Values aufzähle und erkläre, legen wir die Richtung fest, in die sich die beiden Türhälften bewegen sollen. Die Türhälfte, die oben im Bild selektiert ist, soll sich im 90°-Winkel öffnen (also in der Top Ansicht nach oben). Die andere Türhälfte, die oben im Bild nicht selektiert ist, soll sich im 270°-Winkel öffnen (also in der Top Ansicht nach unten).

So, nun erkläre ich dir mal die wichtigsten Keys und deren Values:

- start_open = Tür steht beim Levelstart offen.
- toggle = Regelung mit Triggern.
- crusher = Die Türe fügt dir Schaden zu, wenn du zwischen den Türhälften stecken bleibst
- touch = Um diese Türe zu öffnen, musst du sie berühren, normal musst du sie "per Hand" öffnen
- shoot_thru = du kannst durch die Türe schießen
- nomoster = Monster durchqueren diese Tür nicht

Nun noch die richtigen Keys und deren Values:

- key: Key

value: "X" -> du benötigst den Schlüssel "X" um die Tür zu öffnen. Ein Schlüssel könnte z.B. ein Kelch usw. sein. Dieser muss natürlich den Wert "X" haben, sonst funktioniert das nicht

- key: angle
value: "180" -> hier gibst du die Richtung an, in die sich die Türe öffnet (hier 180°)
- key: targetname
value: "X" -> die Tür wird von einem Aktivator gesteuert, das könnte z.B. ein Schalter sein
- key: speed
value: 100 -> gibt die Geschwindigkeit an, mit der sich die Türe öffnet und schliesst (der Standart ist 100)
- key: wait
value: 3 -> gibt die Dauer (in Sekunden) an, wie lang die Tür offen steht (der Standart ist 3, -1 bedeutet, dass die Tür nie wieder zu geht)
- key: dmg
value: 2 -> gibt an, wieviel Schaden der Spieler erhält, der von dieser Tür eingeklemmt wird.
- key: team
value: "X" -> Türhälften, die die gleiche Value haben, öffnen sich synchron
- key: shoot_thru_scale
value: 1.0 -> gibt den Schaden an, der von der Türe zurückgehalten wird, der Rest geht durch die Türe hindurch (1.0 = der ganze Schaden geht durch, 0.0 = garkein Schaden wird durch die Türe weitergegeben)
- key: type
value: "0" -> gibt den Sound an, der bei dem Öffnen der Tür abgespielt wird (0 = keinen sound, 1 = metalener Sound usw.)

im Prinzip brauchen wir nur den "team"-Key, da sich ja unsere Türen sychron, also gleich bewegen sollen. Daher gibst du bei der einen Türhälfte für "team" die value "aaa" an - für die andere Türhälfte natürlich auch "aaa", da sie ja sonst nicht die gleiche "Value" haben.

[zurück zur Hauptseite](#)

183759



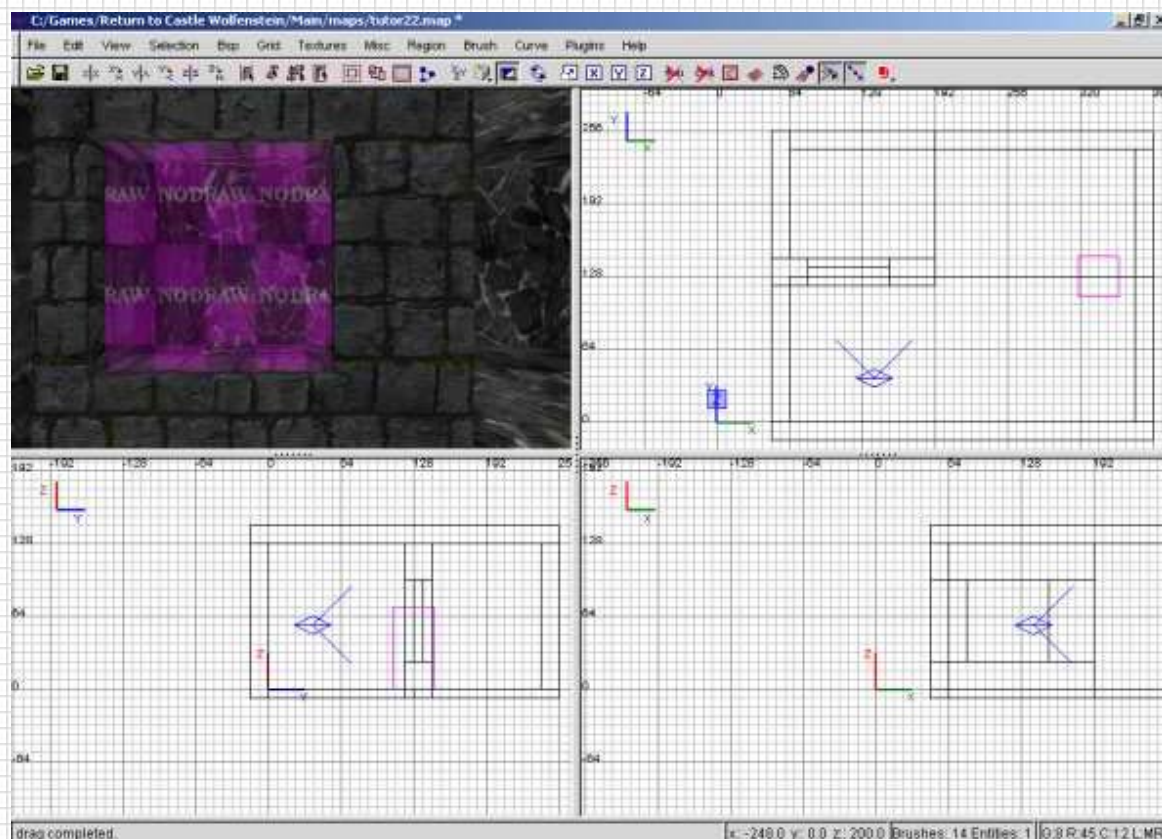
zerstörbare Fenster:

verwendete Map: "tutor22.map"

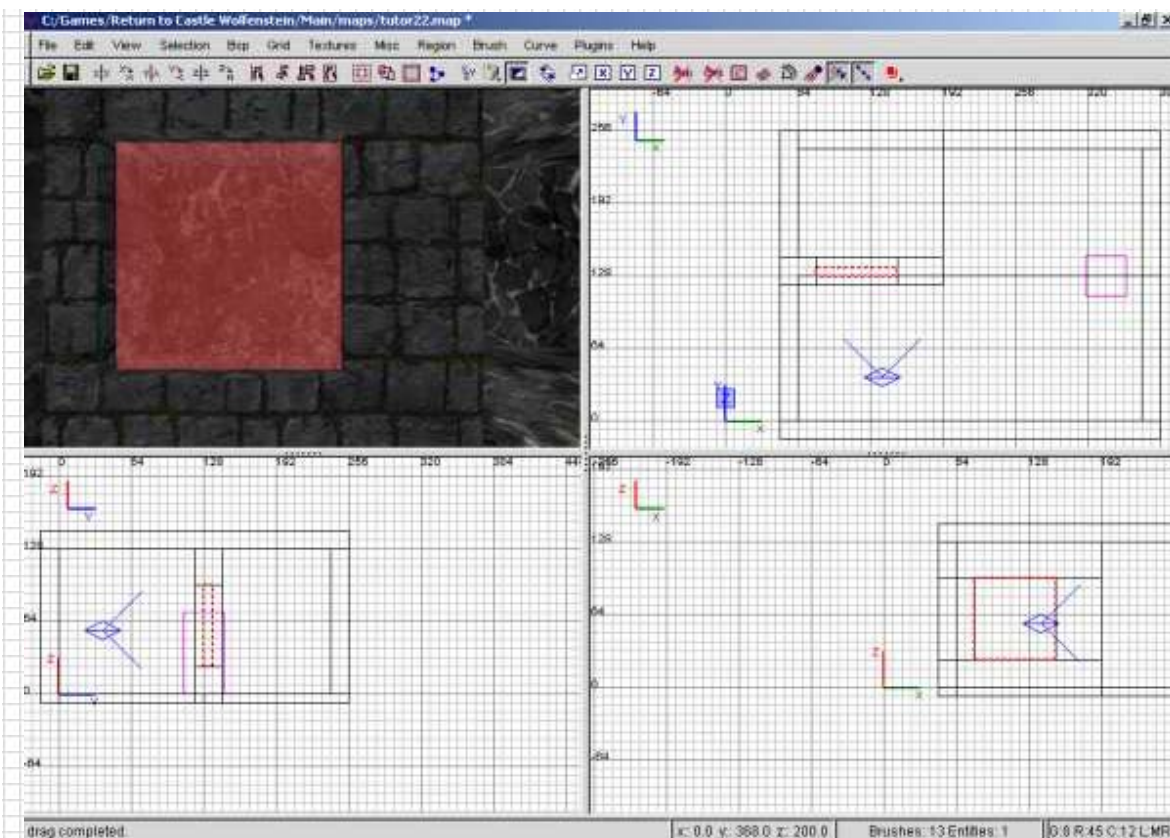
Ergebnismap: "tutor23.map"

Nun wollen wir in unsere Map ein Fenster machen - dazu habe ich bereits eine neue Map gebaut. Wenn du diese öffnest, siehst du bereits eine Öffnung für unser Fenster. Ich habe hier die Wand aus mehreren Brushes gebaut, so dass ein Fenster entsteht, du kannst natürlich auch den "CSG Subtract"-Modus dafür benutzen, davon will ich dir aber abraten, da es eigentlich richtiger "Pfusch" ist, so etwas zu benutzen. Echte Mapper bauen jeden Brush selbst.

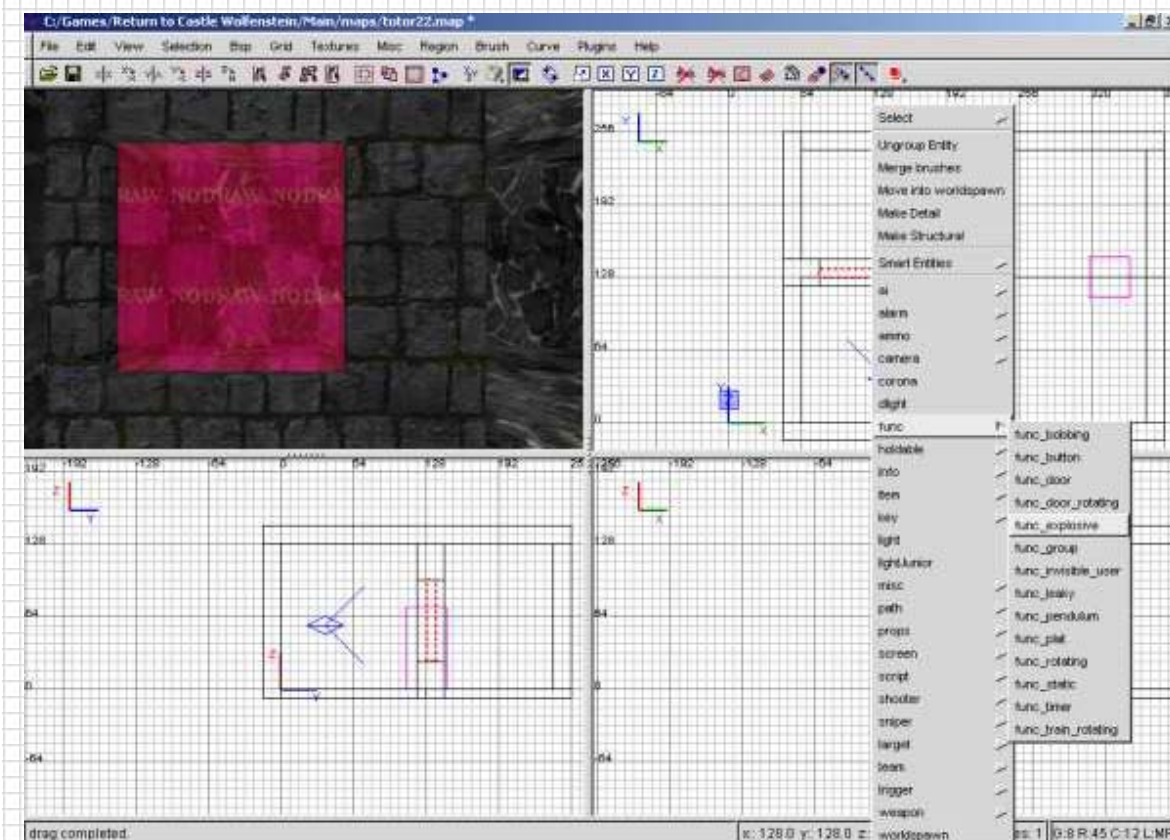
Fangen wir an. Du machst nun einen neuen Brush, der genau in die Öffnung passt, und 8 Units dick ist. Damit haben wir noch eine Fensterbank auf beiden Seiten von jeweils 8 Units, da ja die Brushes, die das Loch bilden, 24 Units dick sind. Diesen Brush belegst du mit der Textur "common/nodraw". Dazu aber ein Bild:



in der Top Ansicht kannst du jetzt gut erkennen, dass der neue Brush nur 8 Units dick ist, auf beiden Seiten entsteht also eine Art Fensterbank. Nun markierst du eine Seite des Brushes (indem du die Taste "STRG" und die Taste "SHIFT" drückst und den Brush mit der linken Maustaste anwählst) und belegst diese mit der Textur "sfx/glass":



Jetzt lässt du den Brush markiert und klickst 2 x mit der rechten Maustaste. Hier wählst du "func" und als Unterpunkt "func_explosive":

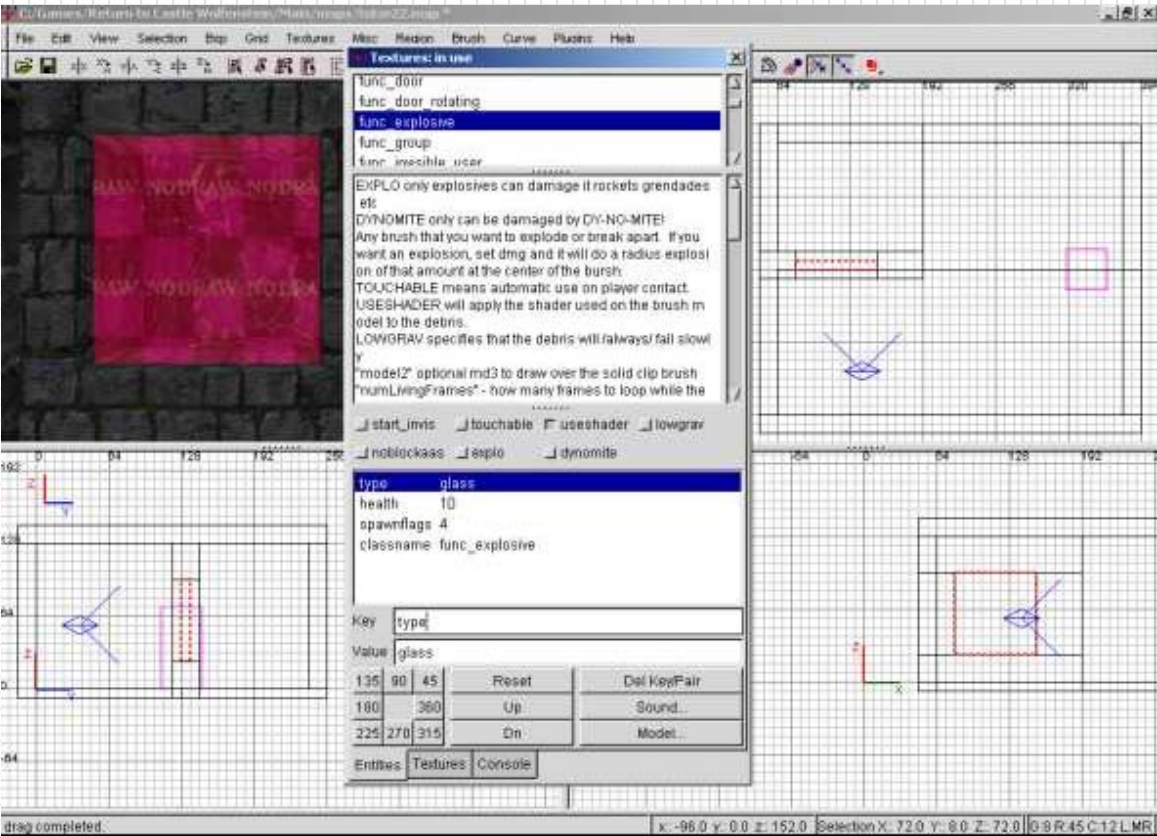


Nun legen wir noch fest, wieviel unser Fenster aushalten soll - schließlich hast du da sicher deine eigenen Vorstellungen. Du drückst also die Taste "N" um das Entity-Fenster aufzurufen. Hier gibst du ein:

- Key: "health"
- Value: "X"

- Key: "type"
Value: "glass"

natürlich stellt hier X eine Zahl dar, die du beliebig wählen kannst, jedoch würd ich als maximalen Wert für normales Glas 50 empfehlen. Ich habe mich für den Wert 10 entschieden. Vergiss nicht, hinterher "ENTER" zu drücken, um die Eingaben zu bestätigen. Wenn du sichergehen willst, dass der Key und seine Value eingegeben sind, kannst du im mittleren Fenster sehen:



Die Option "usesshader" musst du noch aktivieren. Damit lässt du zu, dass der Glas-Shader auch benutzt wird. Andernfalls ist dein Fenster im Spiel nicht durchsichtig.

[zurück zur Hauptseite](#)

183759



Treppen:

verwendete Map: "tutor24.map"

Ergebnismap 1: "tutor25.map"

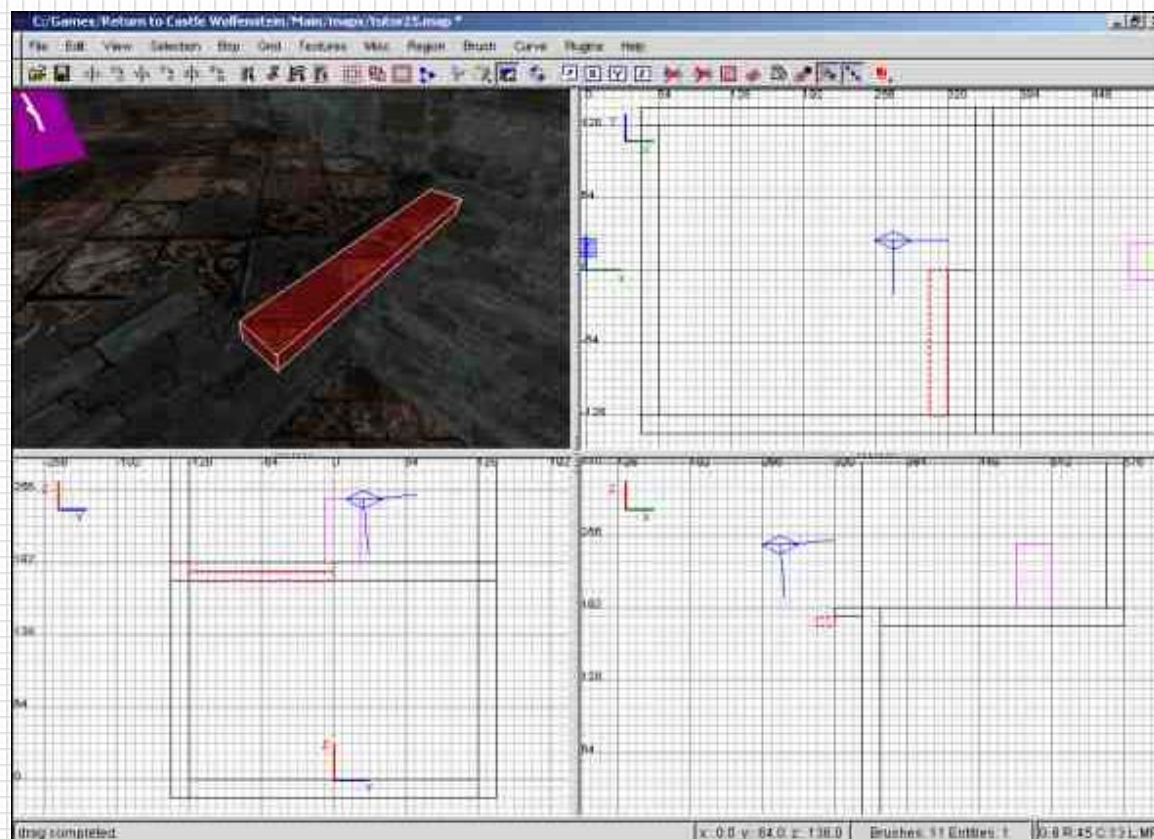
Ergebnismap 2: "tutor26.map"

So, hier will ich dir zeigen, wie man Treppen bauen kann. Zunächst gibt es dafür 2 Möglichkeiten.

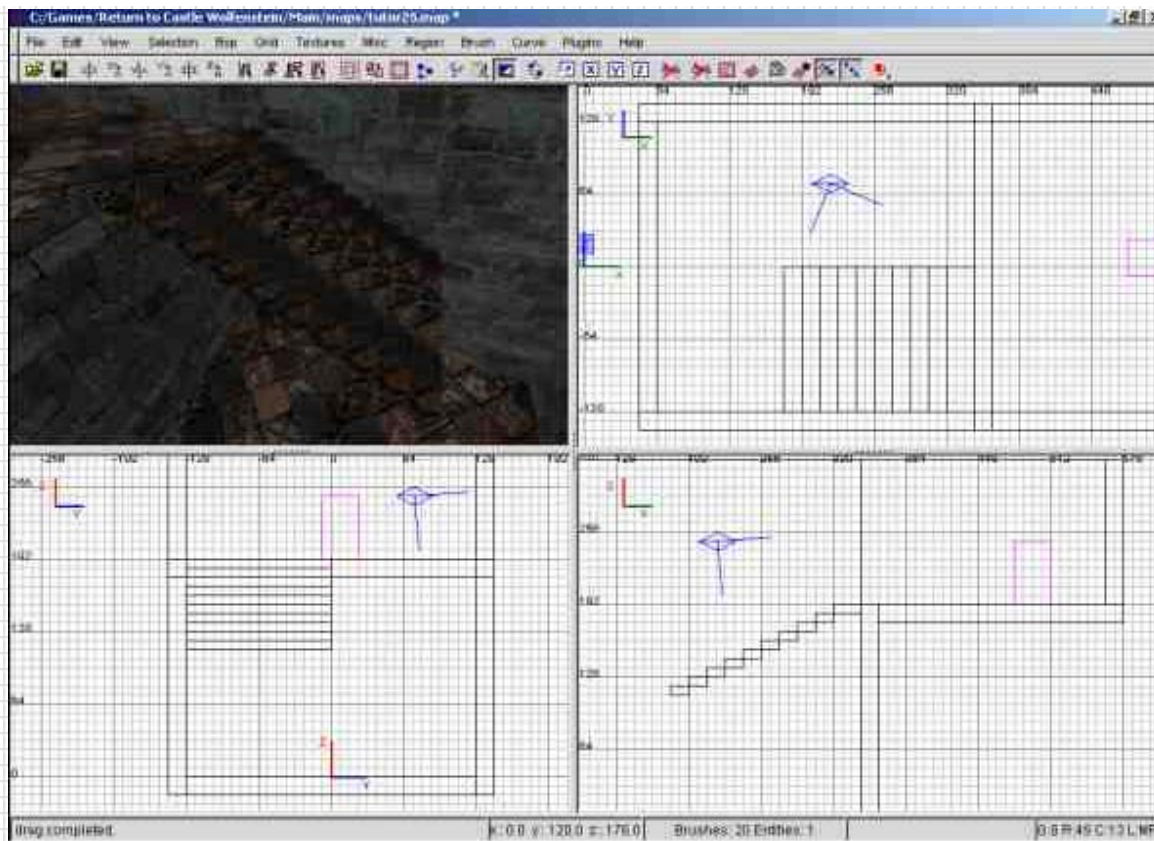
die Treppe selbst bauen

die Treppe von einem Tool bauen lassen

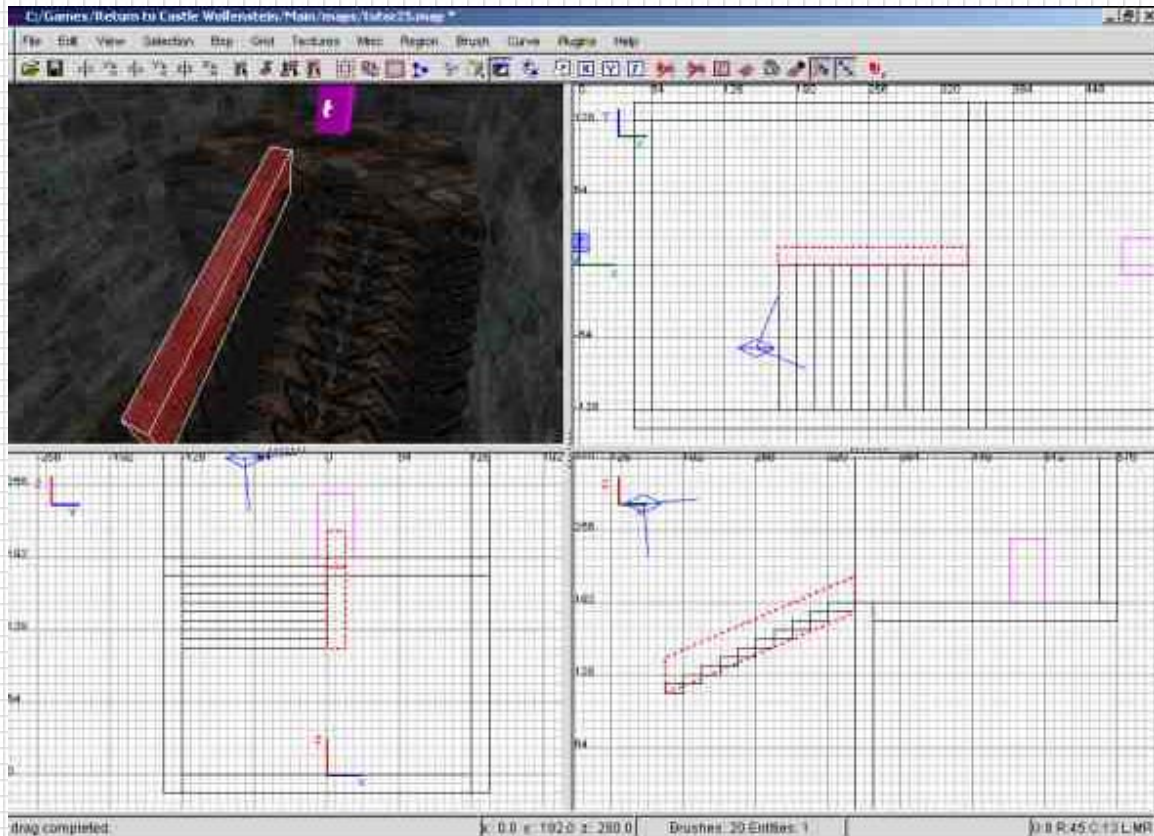
Zunächst will ich dir zeigen, wie man diese Treppe selbst baut. Ich habe dafür schon eine Map gebaut - der Spieler startet oben und er soll nun einen Weg nach unten bekommen, ohne dass er dabei hinunter springen muss. Ich habe als Startpunkt für unsere Treppe schon einen Brush gesetzt, an dem wir einfach weiterbauen. Du machst dir einen Brush, der 8 Units hoch und 16 Units lang ist und setzt ihn vor meinen vorbereiteten Brush:



Nun kopierst du den Brush (indem du die Taste "Space" drückst [Leertaste]). Du setzt ihn nun schräg unter den anderen Brush, so dass unsere Treppe langsam Gestalt annimmt. Nun machst du das ein paarmal, bis du ca. 8 Stufen hast. Nun sollte es etwa so aussehen:

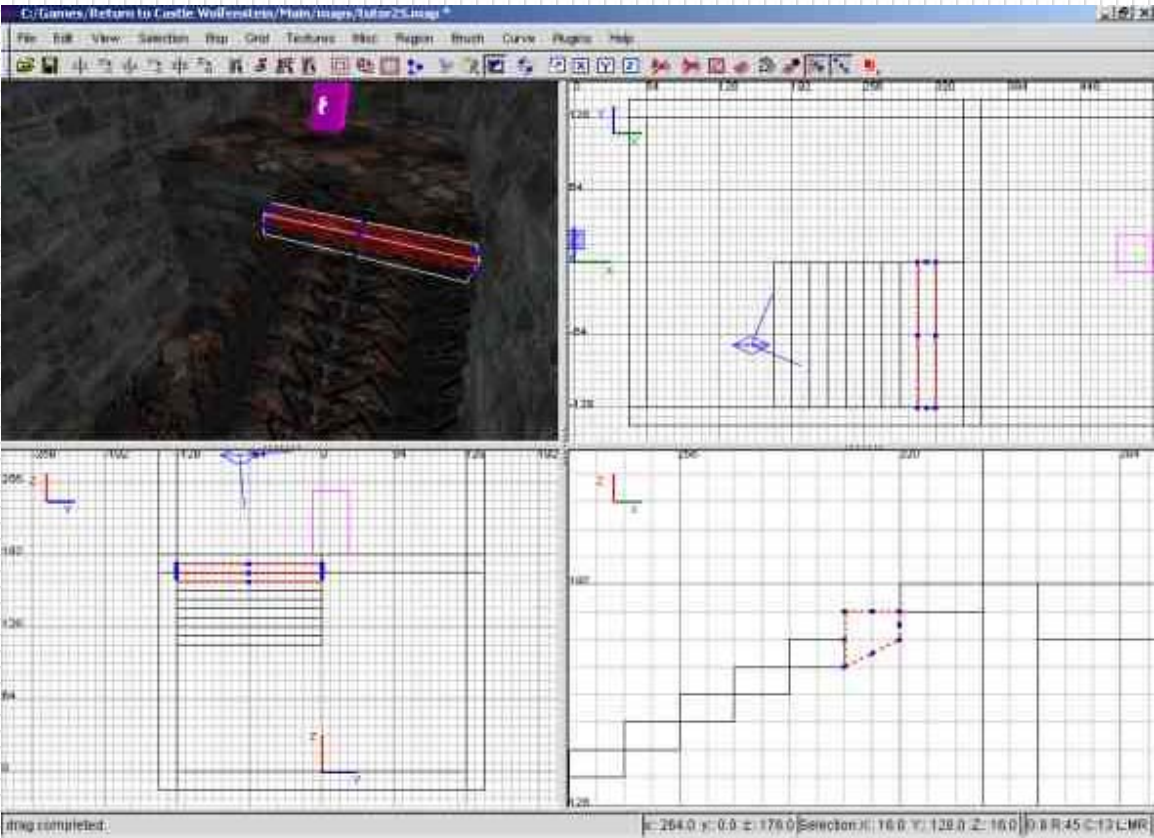


Natürlich kannst du der Treppe auch eine andere Textur geben, damit es nicht so gleich aussieht. Aber darum kümmern wir uns jetzt nicht. Wie du siehst, sieht diese Treppe von der Seite ziemlich öde aus, deshalb werden wir sie etwas verschönern. Da könntest du z.B. ein Gelände einbauen, dann würde es ungefähr so aussehen:

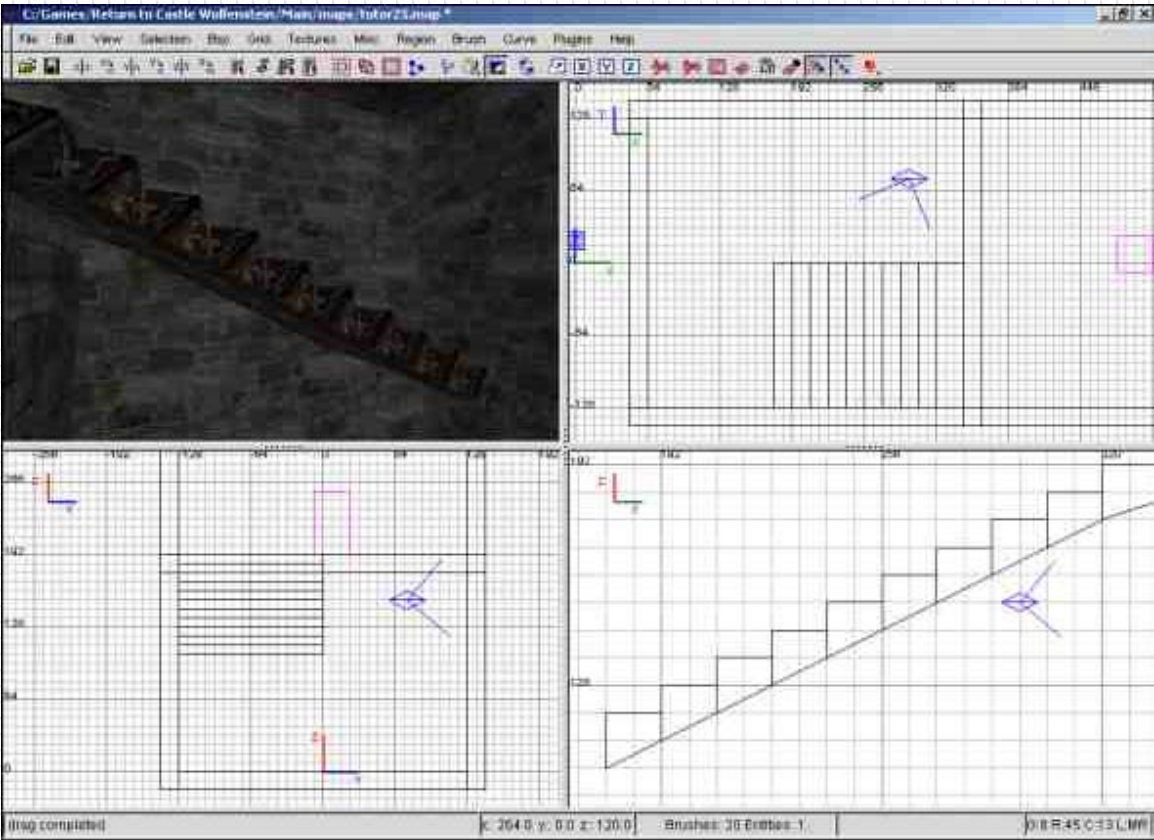


Nun sieht die Treppe von unten doch etwas unglaublich aus, meistens entsteht unter Treppen eine Schräge. Diese machen wir jetzt auch. Dazu löschst du am besten das Gelände und deselektierst erstmal alles. Dann wählst du die oberste Treppenstufe an. Nun

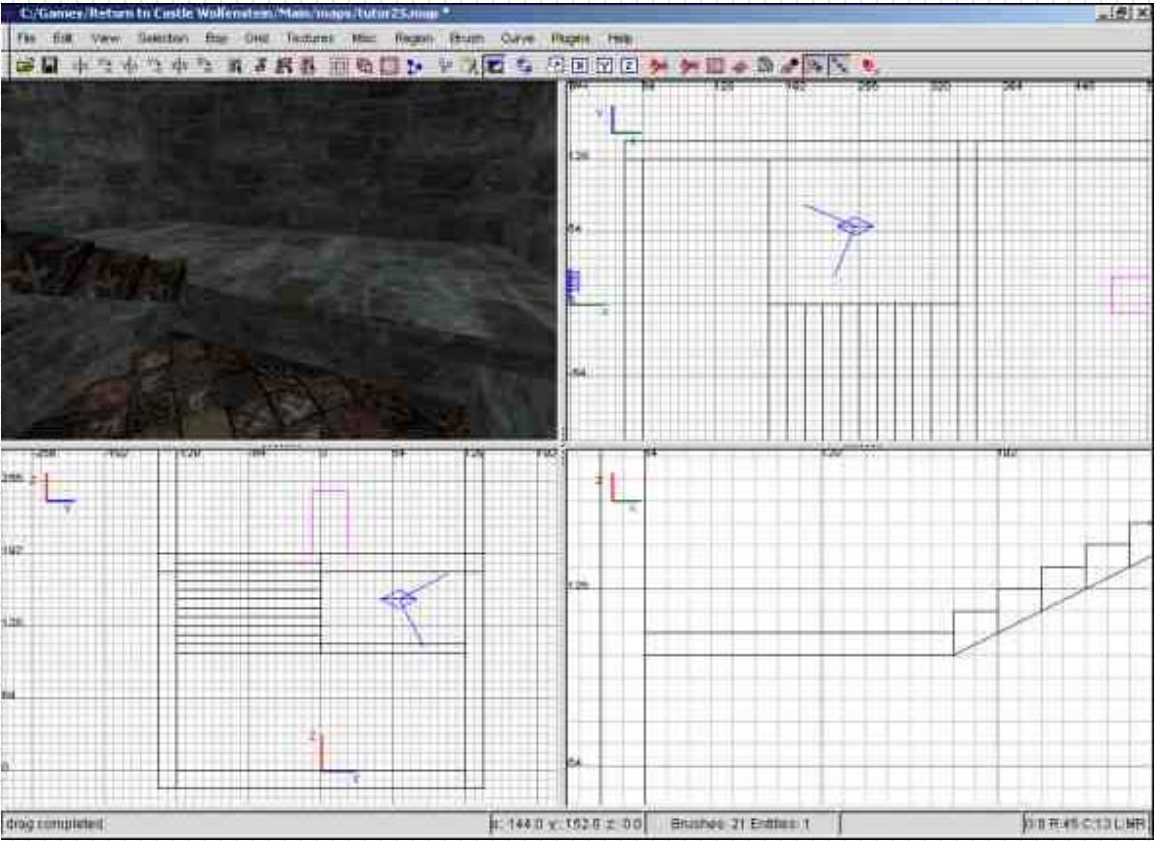
drückst du "E" um in den "EDGE"-Modus zu kommen. Nun wechselst du in die Draufsicht (das Fenster rechts unten) und ziehst den linken unteren Punkt um 8 Units nach unten. So, das schau wir uns jetzt aber mal an:



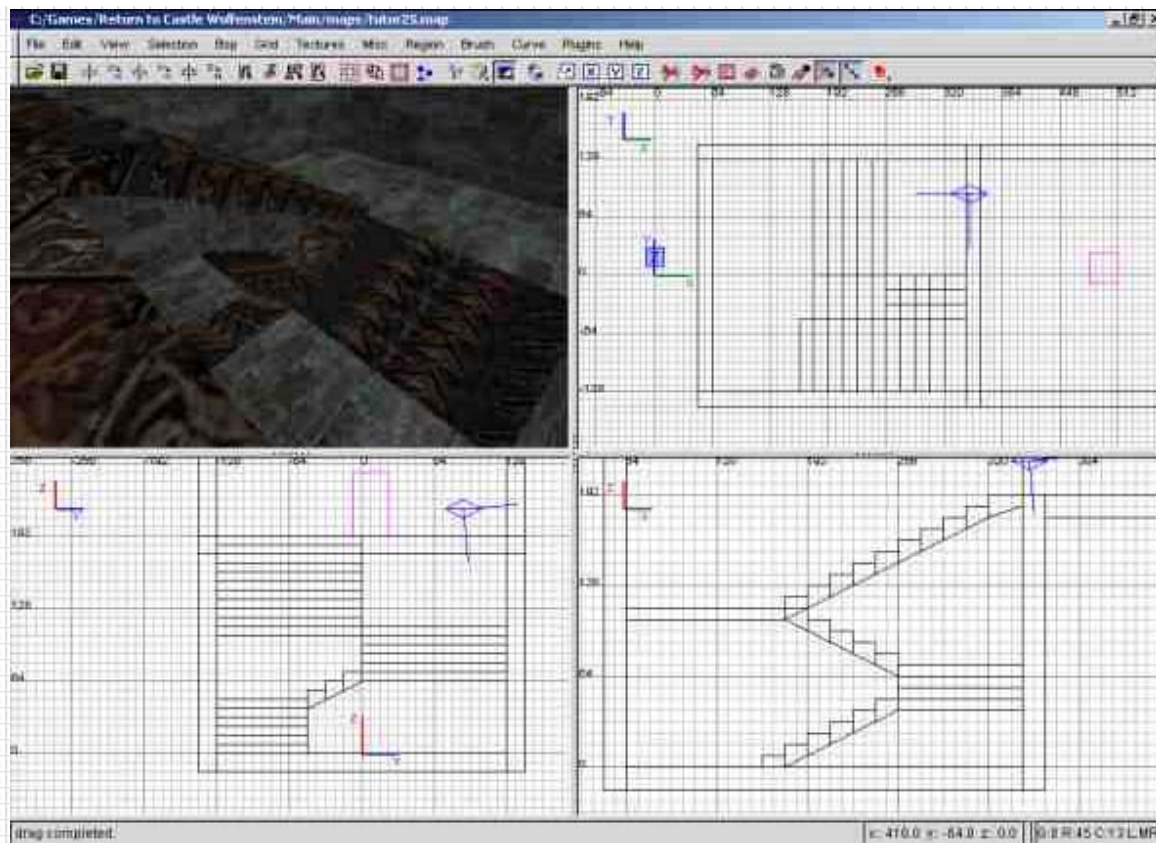
Wie du jetzt in der Draufsicht sehen kannst, schliesst sich jetzt diese Treppenstufe genau an die untere an. Das machen wir jetzt mit jeder einzelnen Stufe. Natürlich kannst du auch alle anderen Stufen löschen und deine veränderte Stufe kopieren, das geht etwas schneller. Dann sieht es so aus:



So, hier kannst du nun in der Draufsicht die neue Schräge gut erkennen. Nun wollen wir natürlich diese Treppe auch bis auf den Boden weiterführen. Dazu betreiben wir aber weniger Aufwand. Du setzt an das Ende der untersten Stufe eine Art Ebene, die du bis zur Wand ziehst. Das könnte dann so aussehen:

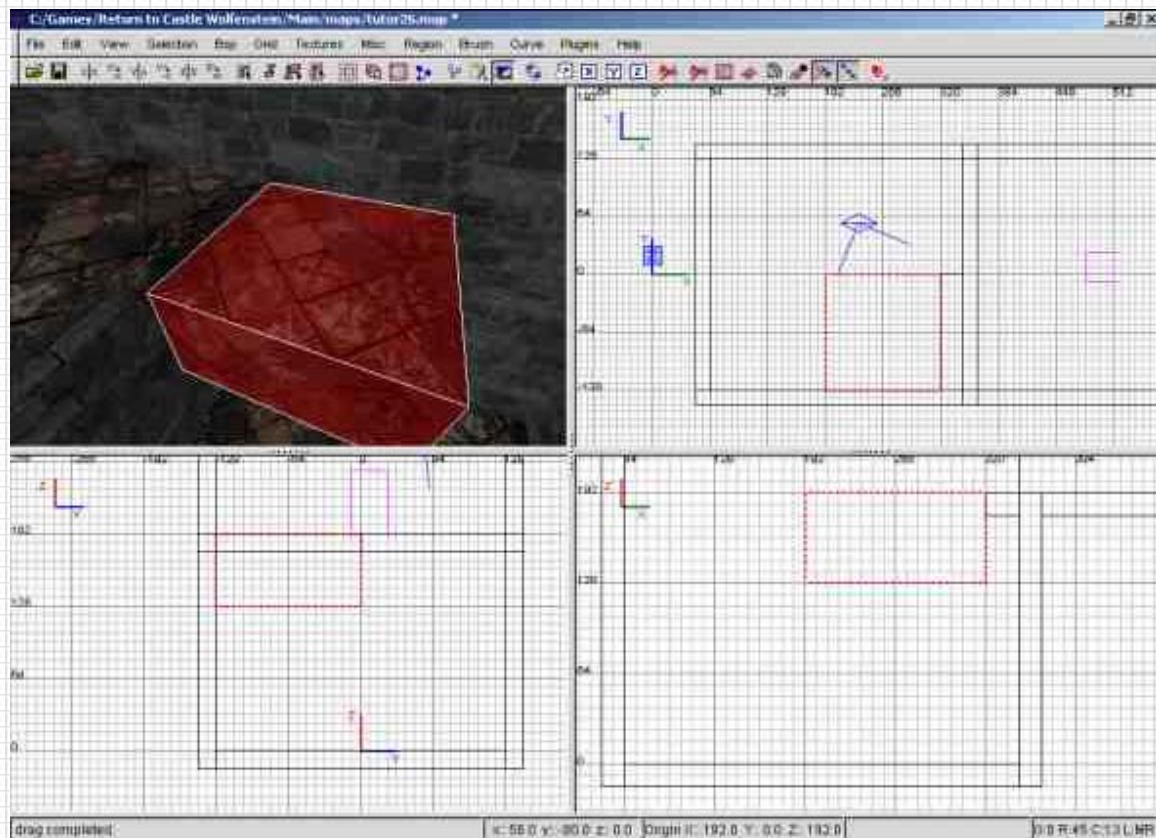


Nun selektierst du die ganze Treppe und kopierst sie. Anschliessend drehst du sie noch um 180° (das sie halt in die andere Richtung zeigt). Wenn du nichtmehr weißt, wie das geht: einen Brush oder eine Gruppe von Brushs kannst du mit der "z-axis-rotate"-Funktion um 90° drehen. Klickst du 2x auf diesen Knopf, drehst du die Treppe also um 180°. Kopiere am besten nicht die ganze Treppe, sonst stösst unser Spieler dann gegen die Wand. Am besten ist es, du baust noch 2 - 3 Ebenen ein und verkleinerst du die Treppe. Zum Schluss könnte es so aussehen:

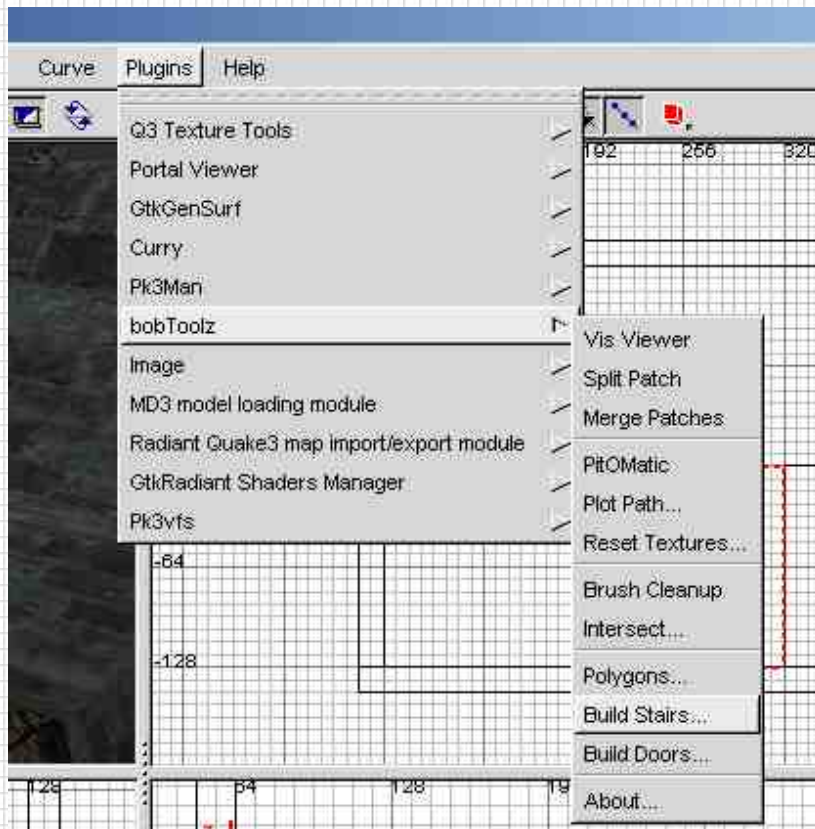


So, jetzt haben wir eine richtig schöne Treppe. Natürlich kannst du diese auch noch mit Geländer usw. verschönern. Das Ergebnis siehst du in der Map "tutor25.map"

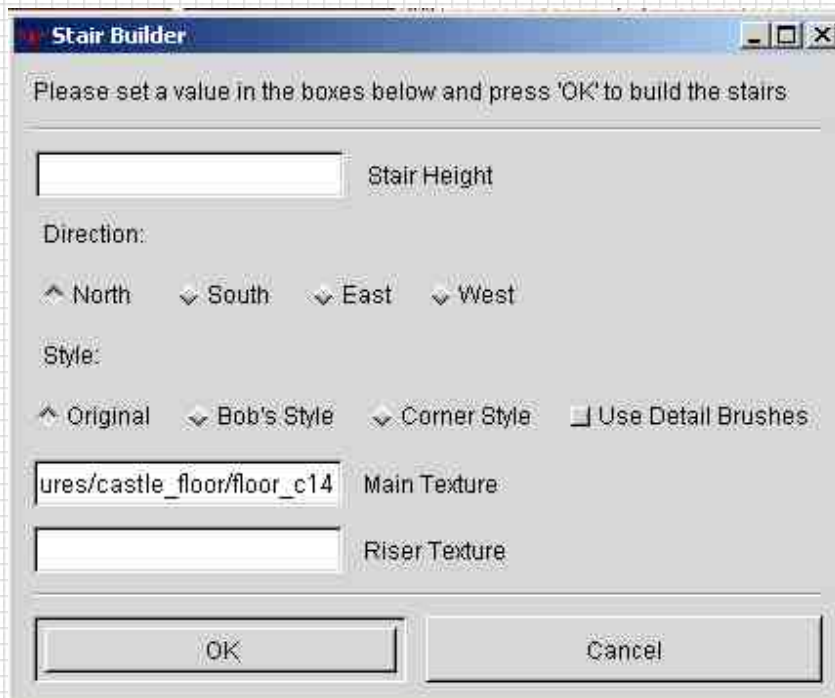
Nun kommen wir aber zu der leichteren Methode, wir lassen uns einfach eine Treppe bauen. Dazu löschst du alle Treppen oder fängst eine neue Map an. Nun bauen wir uns einen Brush, der 128 Units breit und 64 Units hoch ist:



Nun wählst du in der Menüleiste den Punkt "Pugins" und als Unterpunkt "bobtoolz" und hier wiederum "stairs". Dazu muss der Treppenbrush selektiert sein.



Es öffnet sich ein neues Fenster, es erscheint der "Stair Builder":



Hier erkläre ich dir mal die einzelnen Felder:

"Stair Height" = hier gibst du an, wie hoch die einzelnen Treppenstufen sein sollen. Wir tragen hier die Zahl 8 ein.

"Direction" = gibt die Richtung an, in die die Treppe zeigen soll (ich habe "north" gewählt)

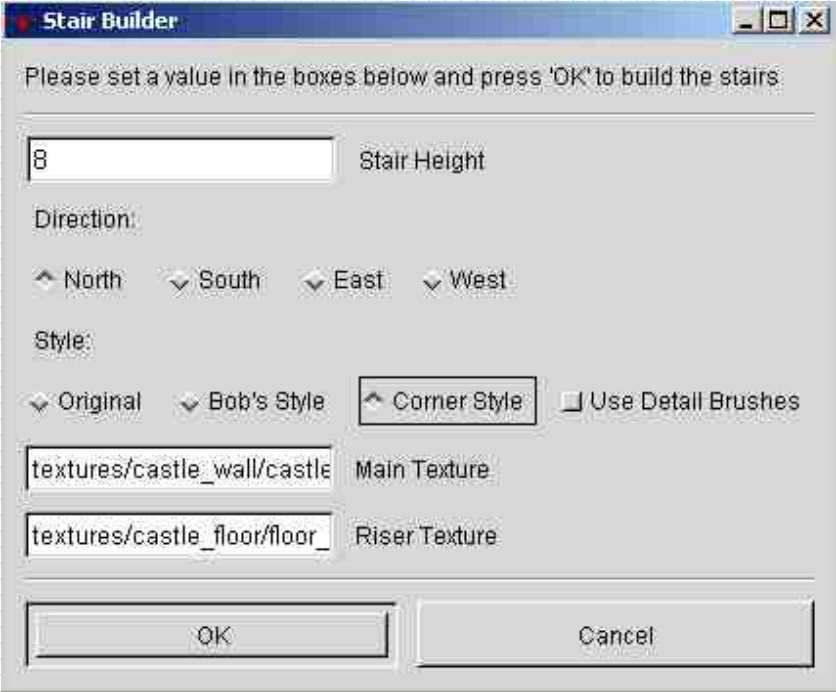
"Style" = hier kannst du angeben, was für eine Art Treppe du möchtest. Wir verwenden "Corner Style", das ist eine Wendeltreppe

"Main Texture" = Das ist die Textur, die dann auf der Treppe zu sehen ist (ich verwende die Textur "textures/castle_wall/castle_c46_on_grid")

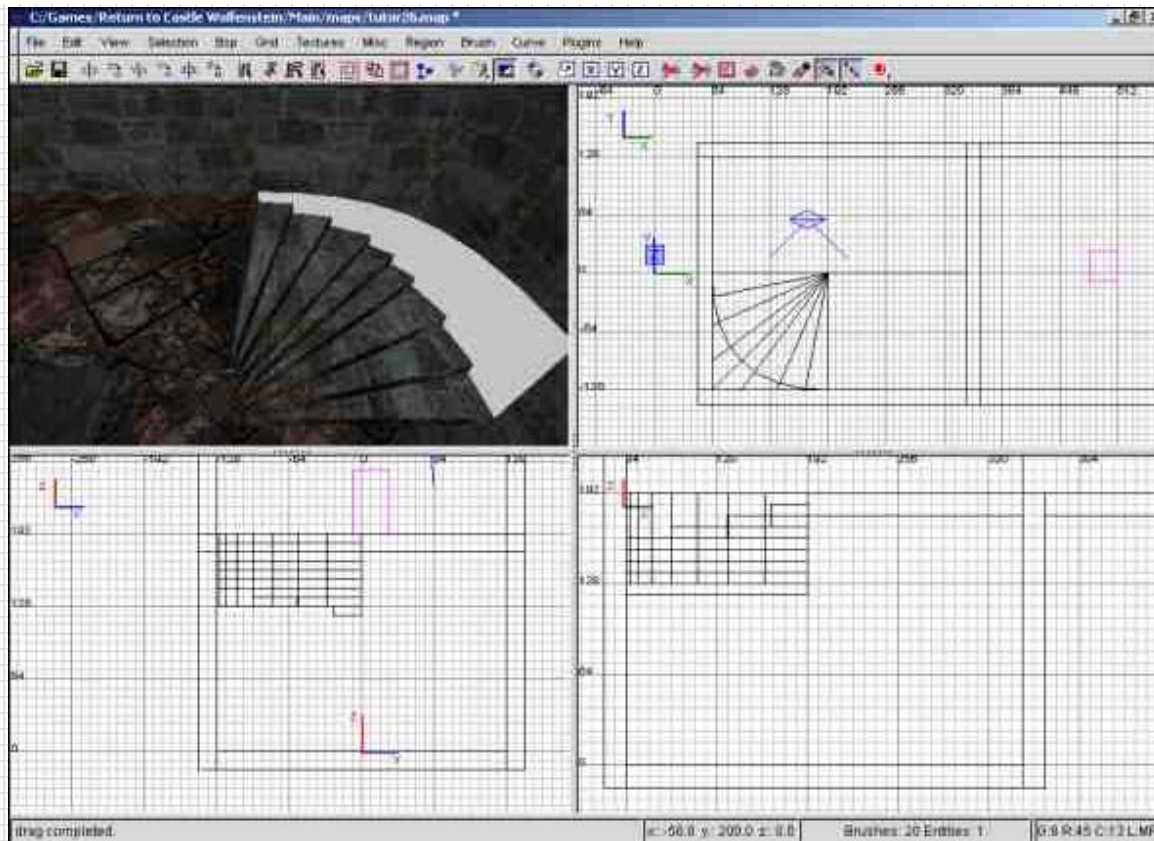
"Riser Texture" = Das ist die Textur, die später auf der Treppenstufenseite zu sehen ist (ich verwende die Textur "textures/castle_floor/floor_c14")

"Main Texture" und "Riser Texture" solltest du unterschiedllch wählen.

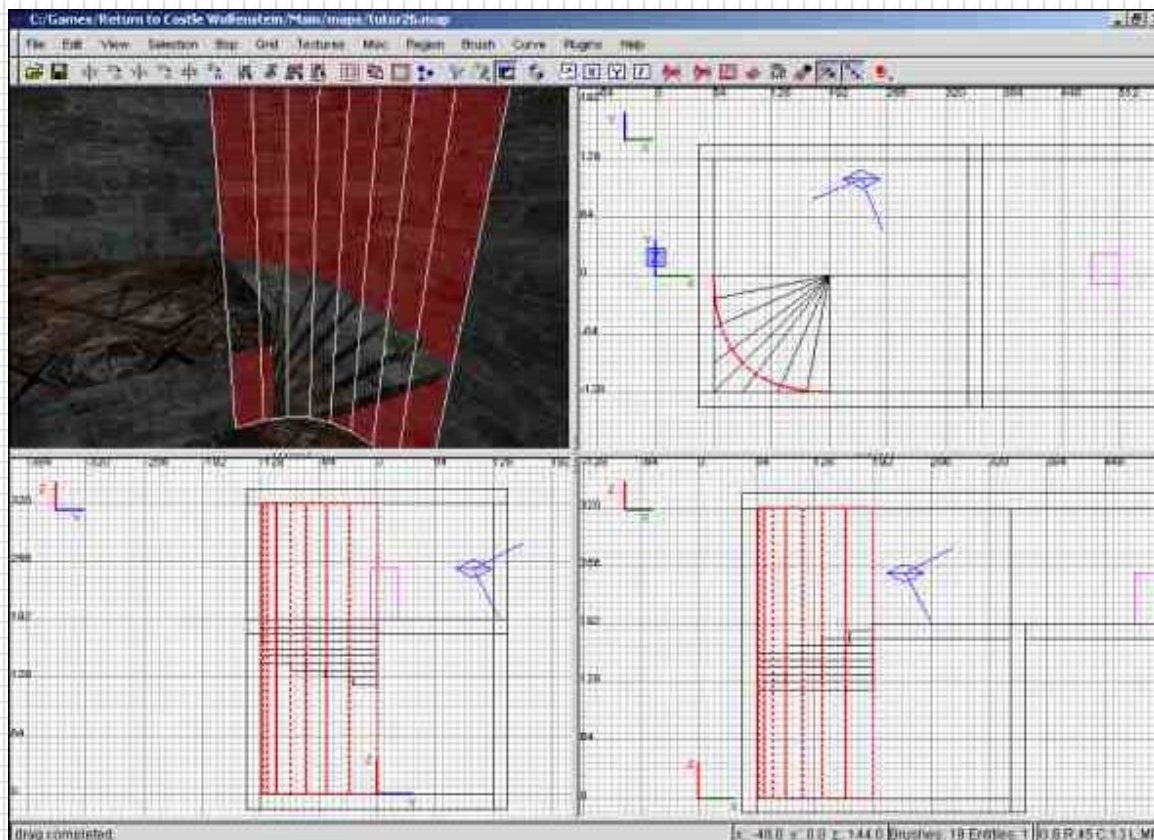
Wenn du alles richtig gemacht hast, sieht es so aus:



Nun drückst du auf "OK" - die Treppe ist fertig. Ich habe diese Treppe gleich noch fertig ausgerichtet und ganz an die Wand geschoben, mein Treppenanfang habe ich noch so verlängert, dass er jetzt an die Treppe reicht:



Jetzt fällt dir sicher diese weiße Textur am Rand auf. Das ist die "Caulk"-Textur. Für diese Textur habe ich ein eingenes Kapitel erstellt. Jetzt wollen wir aber diese Textur loswerden. Du deselektierst alles (dass es bei dir auch so aussieht, wie auf dem Bild). Nun klickst du die weiße Fläche an und drückst "T". Nun wählst du eine Textur aus, die später anstatt der weißen Fläche erscheinen soll. Dann musst du noch diesen Brush soweit verlängern, dass sie vom Boden bis zur Decke reicht. Schliesslich darf man nicht dahinter sehen:



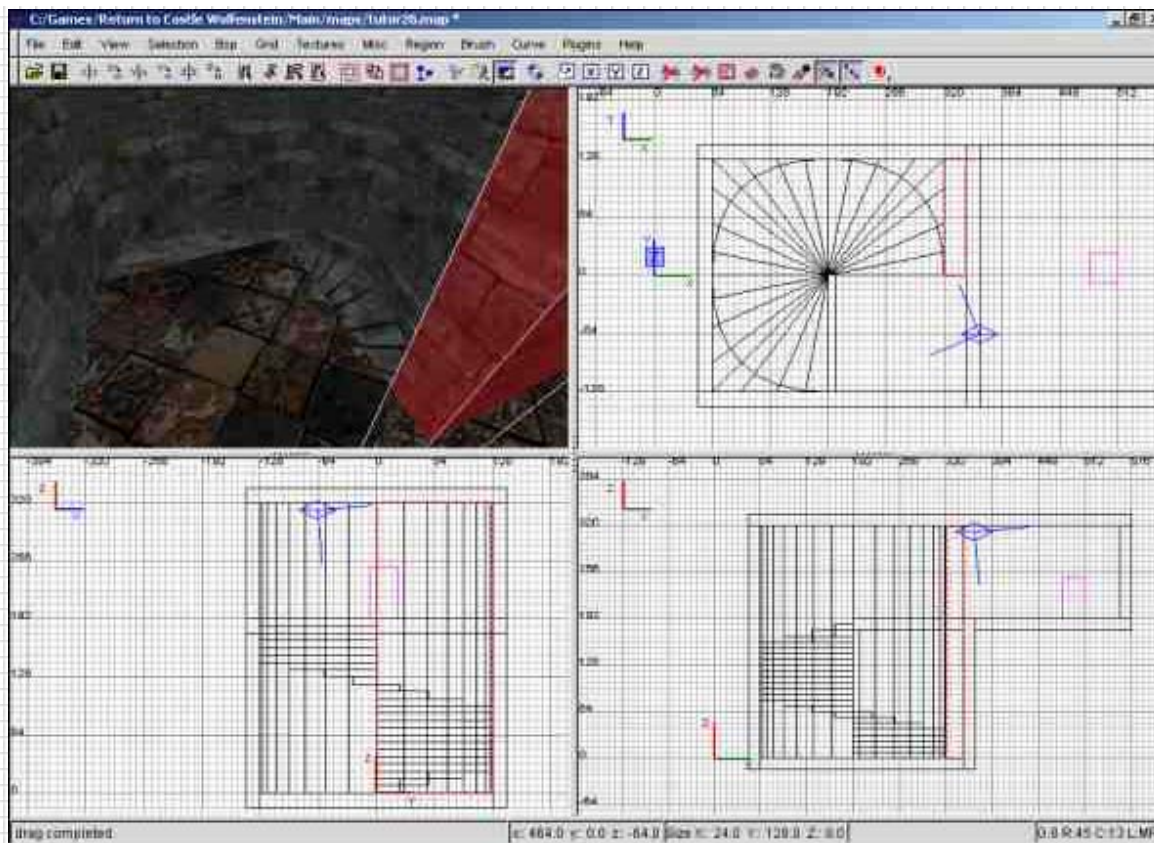
Nun passt die Textur auf der Oberfläche ganz und garnicht. Du drückst "S" - um den "Surface Inspector" aufzurufen und wählst "FIT" und gleich noch "NATURAL". Jetzt müsste die Textur gut sichtbar sein. Jetzt deselektierst du ersteinmal alles.

Nun markierst du den ganzen Treppenteil und drückst die Taste "SPACE" um die Treppe zu kopieren und drehen die neue Treppe um weitere 90°, so dass es bündig an den alten Treppenteil passt:

Brush verhindert die Sicht hinter die Treppe



Ich habe hier einen Brush eingezogen, weshalb kannst du dir ausdenken, wenn du dir diesen Brush weg denkst. Man kann die Treppe von der anderen Seite sehen, das wollen wir natürlich nicht. Deshalb setzen wir hier einen Brush. Nun kommt nocheinmal das selbe Problem, wenn du den dritten Teil einsetzen willst:



Wie du sicher erkannt hast, kann man von der oberen Plattform auch hinter die Treppe sehen - das wollen wir natürlich auch nicht. Daher baust du einfach einen Brush ein, so wie ich ihn hier im Bild schon gebaut habe.

Nun wiederholst du das so lange, bis du am Boden angekommen bist

[zurück zur Hauptseite](#)

183759



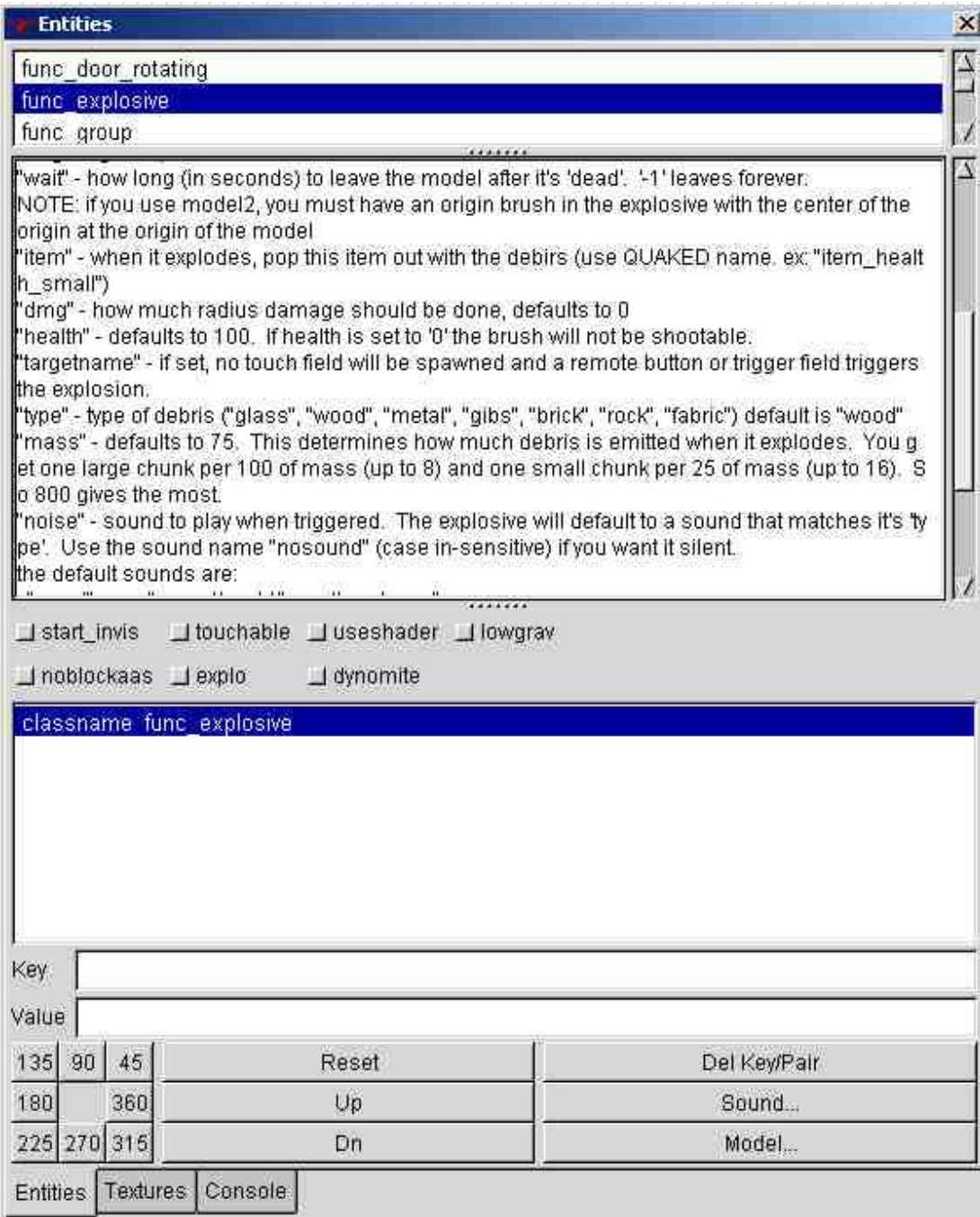
zerstörbare Brushes:

verwendete Map: "tutor27.map"
Ergebnismap: "tutor28.map"

Hier erkläre ich dir, wie man einen zerstörbaren Brush macht. Da Bilder, Wände, Tafeln usw. auch nur Brushes sind, kann man diese auch über diese Methode zerstörbar machen.

Zu diesem Zweck habe ich die Map mit ein paar Sachen ausgerüstet, die wir zerstörbar machen wollen. Zum einen haben wir hier eine Karte, die an der Wand hängt.

Jetzt wählst du diesen "Karten"-Brush an. Nun klickst du in der Top Ansicht 2x mit der rechten Maustaste und wählst "function_explosive". Dann drückst du "N" um die Eigenschaften dieser Karte festzulegen:



Wir aktivieren "notblockaas" und "usesshader". Nun stehen einige keys und values zur Auswahl:

- key: health
value: 5 -> gibt den Wert an, wieviel Lebenspunkte diese Karte hat
- key: type
value: wood -> gibt die Art von Splittern an, die bei der Explosion sichtbar sind
- key: item
value: durch das Zerstören erscheint z.B. ein Treasure (Gold)
- key: targetname
value: hier kannst du wieder einen Aktivator angeben um diese Tafel von weitem zu sprengen

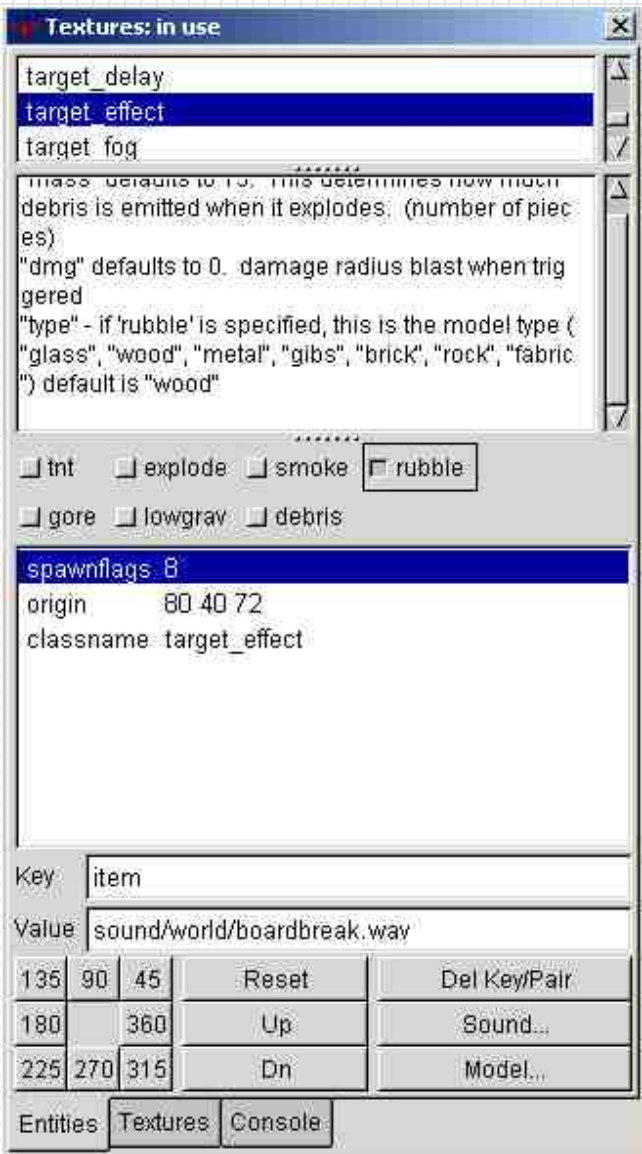
und nun die Keys und values, die wir hier eingeben:

- key: health
value: 5 -> gibt den Wert an, wieviel Lebenspunkte diese Karte hat
- key: type
value: wood -> gibt die Art von Splittern an, die bei der Explosion sichtbar sind

wenn du Lust hast, kannst du den Spieler noch mit etwas Gold belohnen:

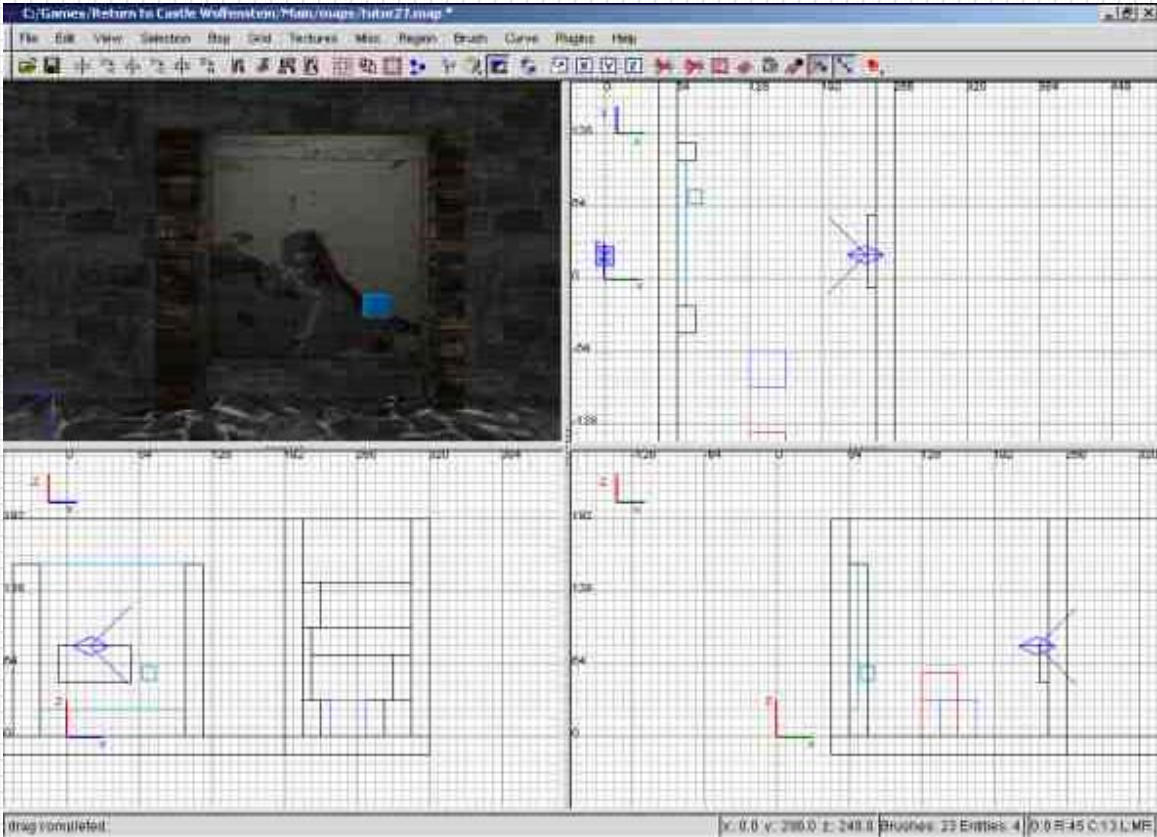
- key: item
value: durch das Zerstören erscheint z.B. ein Treasure (Gold)

Du drückst jetzt in der Top Ansicht 2x mit der rechten Maustaste und wählst im Menü "target" und da "target_effect". Es erscheint eine blaue Box, unser "Target". Nun drückst du "N" um die Eigenschaften dieses Targtes festzulegen:



Wir aktivieren das Feld "rubble"

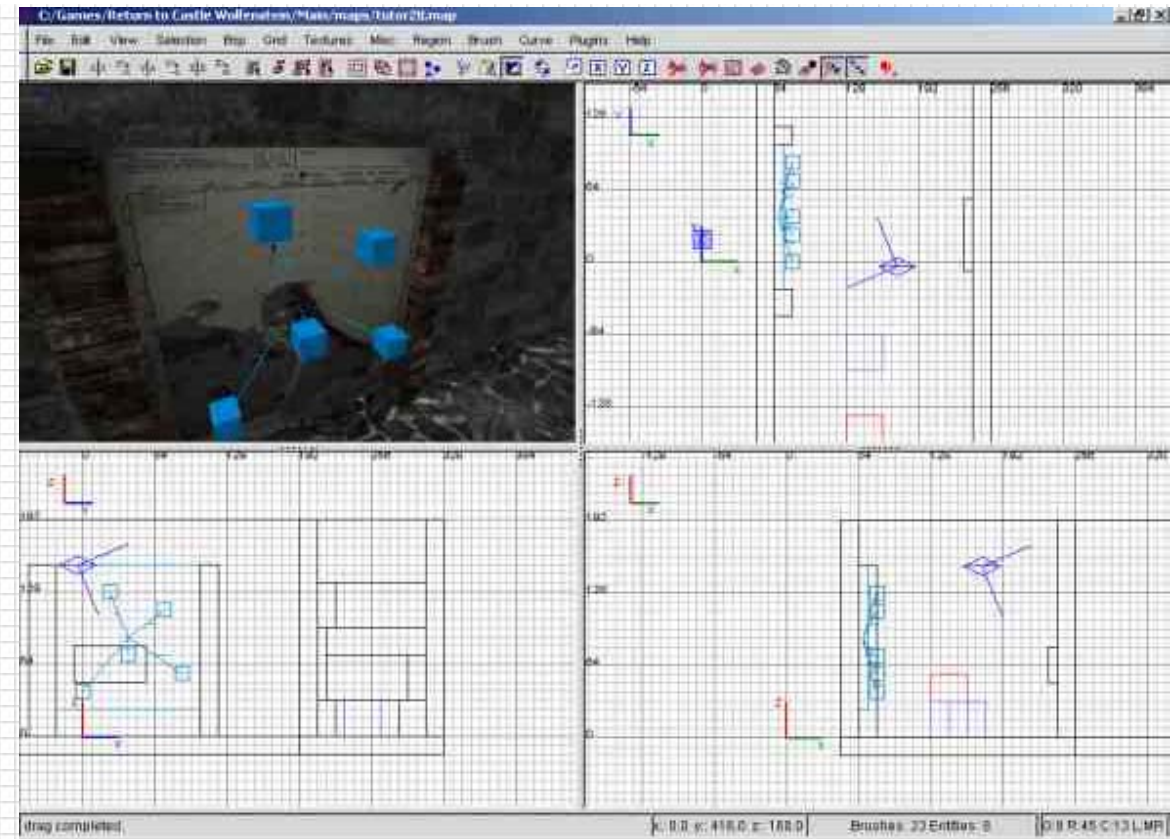
Diesen "target" setzt du nun vor die Tafel:



Das wiederholst du nun ein paarmal, bis du 3 - 5 Stück dieser "target_effect"s vor deiner Karte stehen hast. Nun markierst du zuerst einen "target_effect", anschliessend die Tafel. Nun drückst du "STRG" + "K" (du hältst "STRG" gedrückt und drückst noch dazu "K") den Target mit der Karte zu verbinden. Wenn du das richtig gemacht hast, müsstest du jetzt einen blauen Strich zwischen dem Target und der Karte sehen. Jetzt drückst du die "ESC"-Taste um alles zu deselektieren.

Das wiederholst du jetzt bei allen targets, die du gesetzt hast. Aber denk dran, immer nur einen einzelnen target markieren, dann die Tafel, dann "STRG" + "K" drücken. Machst du hier etwas falsch, funktioniert es nachher nicht.

Wenn du alles richtig gemacht hast, sieht es so aus:

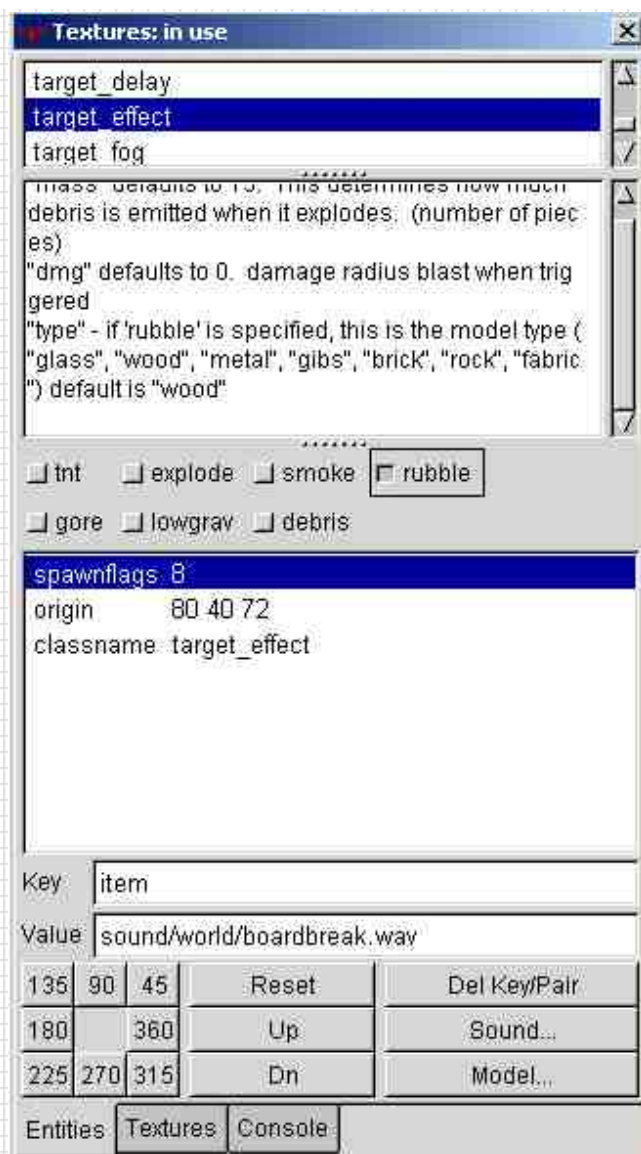


Das hätten wir also. Nun kümmern wir uns um das Schild mit der Aufschrift "Eintritt verboten". Dieses Schild wollen wir auch zerstörbar machen. Also klicken wir es wieder an und klicken in der Top Ansicht 2x mit der rechten Maustaste und wählen "function" und als Unterpunkt "function_explosive". Du drückst "N" und gibst folgende Keys und Values ein:

key: health
value: 5

ausserdem aktivierst du wieder "notblockaas" und "usesshader". Du drückst "ESC" um alles zu deselektieren

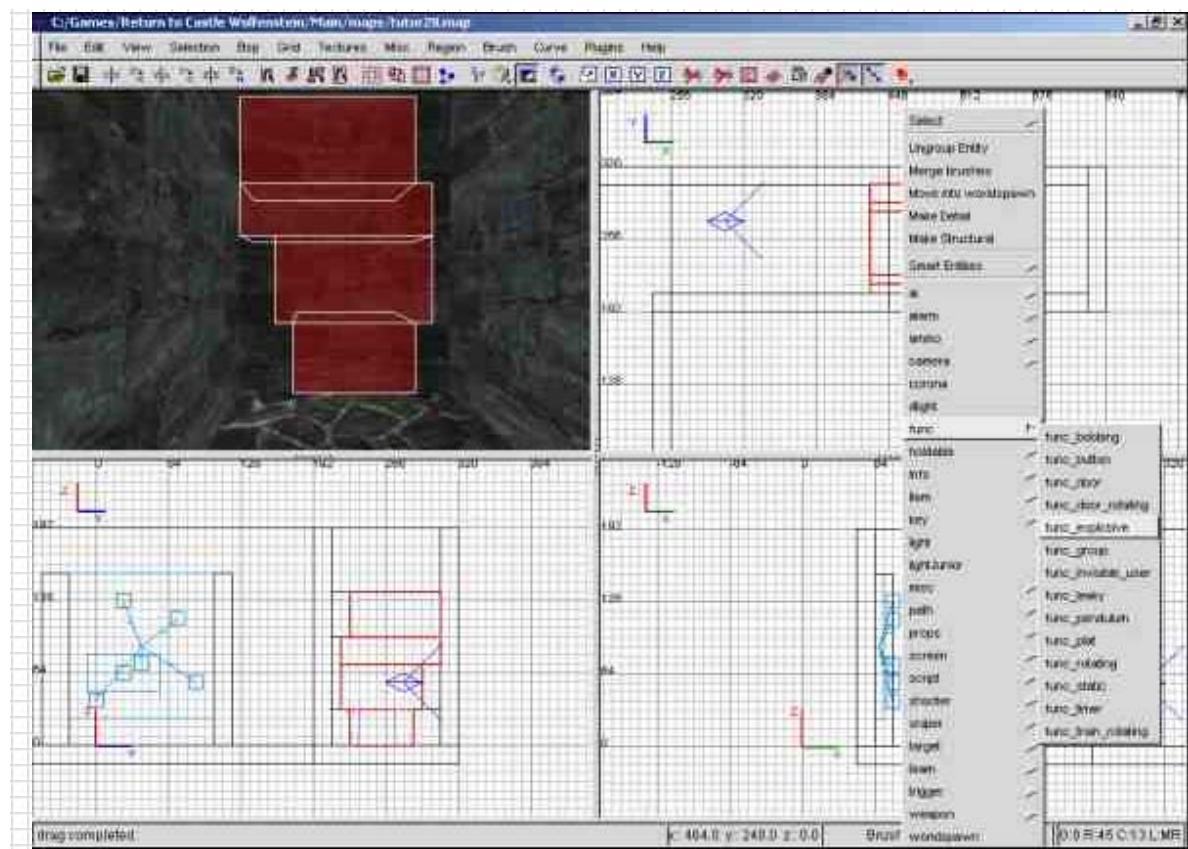
Nun setzt du wieder einen "target_effect" davor. Du wählst wieder "N" und aktivierst das Feld "rubble":



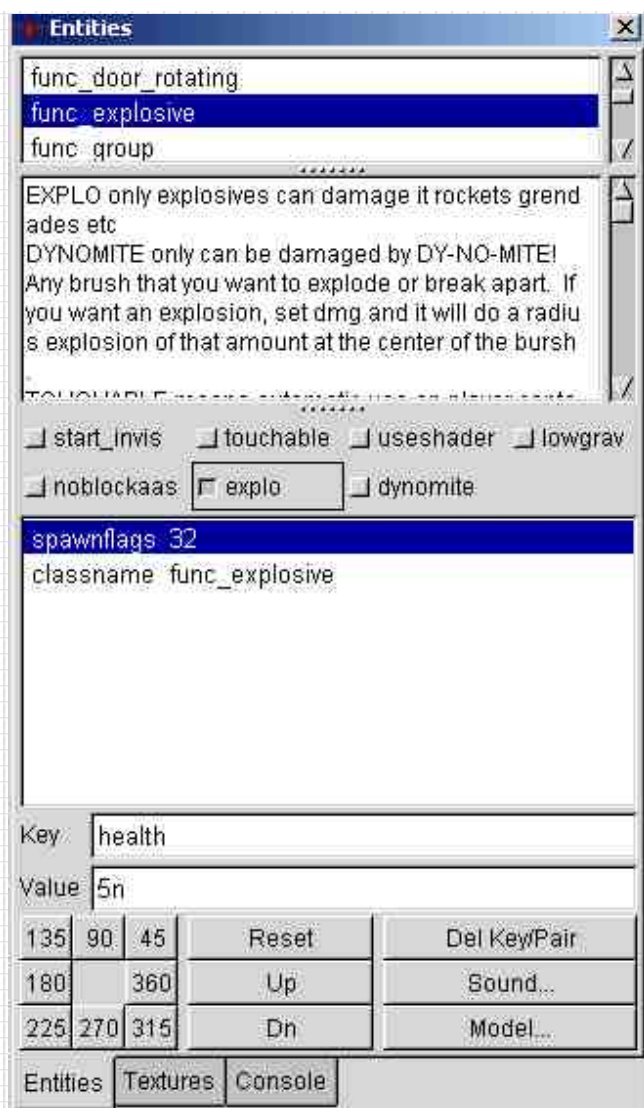
Du drückst wieder "ESC" um alles zu deselektieren. Nun wählst du erst das "target_effect" an, anschließend das Schild. Nun drückst du "STRG" + "K". Es erscheint ein Strich, der den Target und das Schild verbindet. Damit hätten wir auch das Schild fertig.

So, jetzt bleibt nur noch die Wand, die den Spieler daran hindert, den Gang bis zum Ende zu gehen. Das müssen wir natürlich ändern. An dem Eckpunkt des Ganges habe ich ein Dynamit-Paket hingelegt. Damit soll der Spieler die Wand sprengen können.

Wie du siehst, habe ich diese Trennwand so aufgebaut, dass sie aus mehreren kleinen Brushes besteht. Wir wollen nicht die ganze Wand explodieren lassen, da das unrealistisch aussieht. Vielmehr selektieren wir nur ein paar Brushes, die später explodieren sollen. Also wählst du ein paar Brushes aus und klickst in der Top Ansicht 2x mit der rechten Maustaste, im Menü wählst du wieder "function" und hier als Unterpunkt "function_explosive":



Anschliessend drückst du wieder "N" um das Entity-Fenster aufzurufen:



Hier aktivieren wir "explo". Das bedeutet, diese Wand kann man mittels Rockets usw. sprengen. Natürlich kannst du auch "dynamite" wählen, dann ist diese Wand wirklich nur mit Dynamit zu sprengen.

[zurück zur Hauptseite](#)

183759

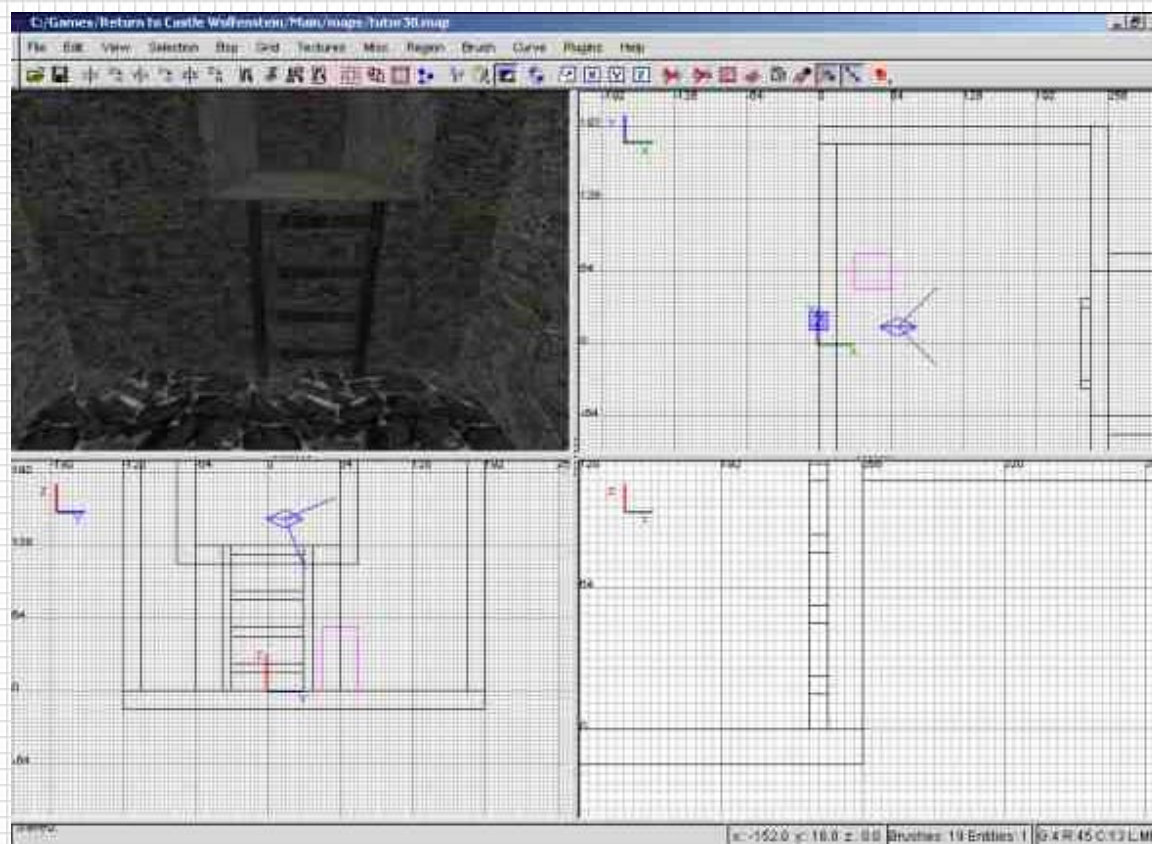


eine Leiter bauen:

Verwendete Map: "tutor29.map"

Ergebnis-Map: "tutor30.map"

Dazu habe ich eine Map gebaut, die wieder aus 2 Ebenen besteht, der Player startet unten und soll mittels einer Leiter die obere Ebene erreichen. Dazu baust du dir jetzt ersteinmal einen Brush, der an der der Mauer sitzt und bis an den Boden der oberen Ebene reicht. Dann kopierst du diesen Brush und setzt ihn etwa 64 Units Entfernung ebenfalls an die Mauer. Dann setzt du noch Sprossen an die beiden Holme, dass es ungefähr so aussieht:



Jetzt haben wir nur eine Ansammlung von Brushes. Und jetzt bauen wir uns erst die Leiter. Diese ist lediglich eine Textur, nämlich die Textur "common/ladder". Also machst du einen neuen Brush, den du bündig vor die Leiter setzt, die wir gerade gebaut haben. Nun wählst du nun im Menüpunkt "Textures" den "Common"-Ordner aus und wählst hier die Textur "ladder". Das wars. Also könntest du eigentlich auch z.B. eine Regenrinne bauen - und aussen herum einen weiteren Brush mit der "common/ladder"-Textur machen. Dann könntest du die Regenrinne hochklettern.

[zurück zur Hauptseite](#)

183759



Der Umgang mit der pk3-Datei:

Hier lernst du nun den Umgang mit den *.pk3-Dateien. PK3-Dateien sind eigentlich ganz normale Zipdateien, jedoch mit einer anderen Endung (also *.pk3 statt *.zip). Also probieren wir gleich mal etwas aus.

Alle PK3-Dateien für RtCW befinden sich im Verzeichnis "Main". Hast du eine PK3-Datei in einem anderen Ordner, so kannst du diese Map nicht unter RtCW starten, da RtCW diese Datei nicht findet.

Nun erstellst du dir auf einer Festplatte einen Ordner - diesen brauchen wir nur zu Übungszwecken, also kannst du ihm irgendeinen Namen geben. Ich nenne ihn "tutortemp":

Hier erstellst du nun die folgenden Ordner:

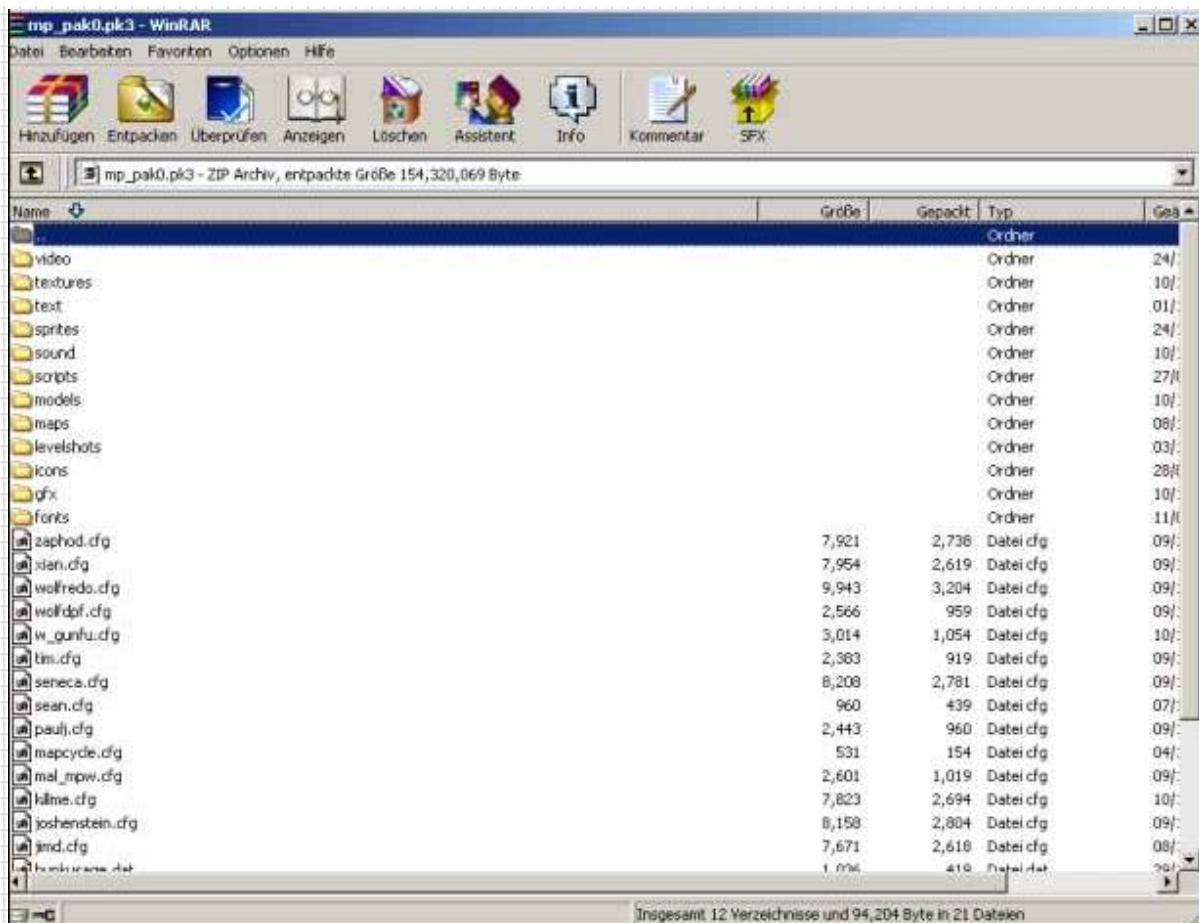
- levelshots
- maps
- scripts

Mehr brauchst du eigentlich nicht für eine Map. Nun erkläre ich dir, was es mit diesen Ordnern auf sich hat. In den Ordner "Levelshots" kopierst du dein Startbild, welches beim laden der Map erscheinen soll. Der Ordner "Maps" enthält logischerweise die BSP-Datei deiner Map. Im Script-Ordner befindet sich eine Datei, um die wir uns später kümmern. Nur soviel zum Verständnis: Es handelt sich dabei um eine Datei, die dem Spiel Informationen über die Map (die Spieldauer, Spawnzeit usw.) angibt.

Diese 3 Ordner brauchst du, um eine Map releasen zu können, da die Map so wenigstens im Menü auftaucht. Andernfalls müsste man die Map jedes Mal über die Console starten - und bei vielen installierten Maps spielt man dann eben die, die man bequem auswählen kann.

WICHTIG: Bitte kopiere diese PK3-Datei, denn wir werden sie so verstümmeln, dass du sie nichtmehr benutzen kannst, also **MUSST** du darauf achten, dass du sie auch wirklich **KOPIERST** - denn diese PK3-Datei, die du benutzt, wird mit Sicherheit unbrauchbar. Wenn du dir nicht sicher bist, ob du die Datei kopiert hast, schau einfach im "Main"-Ordner und im "tutortemp"-Ordner nach. Wenn die Datei in **BEIDEN** Ordnern ist, ist alles ok.

Nun löschst du einfach die 3 erstellten Ordner und schau dir mal den "Main"-Ordner genauer an. Hier findest du die verschiedenen PK3-Dateien. Hier suchst du dir nun eine (ich habe mich für die Datei "mp_pak0" entschieden) PK3-Datei aus und kopierst sie in unseren Ordner. Nun startest du WinZip (oder ein anderes Zipprogramm) und öffnest die pk3-Datei. Bei mir sieht diese so aus:



Hier findest du nun die 3 Ordner wieder, die ich dir oben erklärt habe. Dazu findest du hier einige andere, z.B:

- gfx
- models
- textures
- video
- sound
- music
- und ggf. einige andere

Dies ist die Verzeichnis-Struktur, die jeder Map zugrunde liegt, d.h. in ihr sind Texturen, Scripte, Modelle usw.. Zuerst erkläre ich dir, was in die jeweiligen Ordner reinkommt:

GFX: hier gibt es einen Unterordner namens "2d". In diesem Ordner ist ein weiterer Ordner, der "mp_objectives"-Ordner. Hier sind die Objectives-Bilder, die du siehst, wenn du die Map in RtCW öffnest und dir die einzelnen Objectives ansiehst.

LEVELSHOTS: In diesem Ordner findest du das Startbild der Map. Wenn du so ein Startbild hast, achte darauf, dass es im JPEG-Format oder im TAGA-Format vorliegt. Wenn du ein JPEG-Bild hast, musst du darauf achten, dass es nicht als "progressive" abgespeichert. Im Beispiel die Datei XXXX.bsp

MAPS: Hier findest du die BSP-Datei, also die compilierte Map aus dem Radiant, also z.B. die Datei XXXX.tga

MODELS: Hier findest du die Modelle, die in der Map enthalten sind.

SCRIPTS: Hier findest du 2 Typen von Dateien: *.arena-Dateien und *.shader-Dateien. Diese schauen wir uns später noch genauer an, da diese Dateien sehr wichtig sind, also z.B. XXXX.arena und XXXX.shader

TEXTURES: Hier findest du die Texturen, die in der Map vorkommen.

VIDEO: Hier findest du die XXXX.roq-Dateien - das sind die kleinen Videos, die man in der Levelauswahl ansehen kann, z.B. kann man hier einen kleinen Flug durch die Map genießen.

SOUND: Hier befinden sich die Sounds, die in der Map abgespielt werden

MUSIC: Hier befindet sich die Hintergrundmusik, die im Level zu hören ist.

WICHTIG: Wenn deine Map z.B. kein Video, keine Models usw. hat, kannst du diese Ordner auch einfach weglassen, d.h. wenn du keine eigenen Modelle einsetzt, brauchst du auch keinen "Models"-Ordner.

Nun kannst du ja in dem Ordner einfach alle Ordner erstellen, die du für deine Map brauchst und dort die entsprechenden Dateien hinein kopieren.

Nun kommen wir zu dem letzten Schritt - das Packen der Datei. Dazu musst du alle Ordner markieren (also die Ordner "Maps", "Scripts", und "levelshots"). Wenn du lediglich den Ordner packst, in dem sich die 3 Ordner befinden, kann das Spiel die Map nicht laden, da die Map nun nichtmehr der Verzeichnis-Struktur entspricht.

ÜBRIGENS: Wie du sicher weisst, kann man Zip-Dateien mit verschiedener "Dichte" packen, also z.B. beste Kompression, mittlere Kompression usw. Wenn du dabei bist, deine Map zu testen, kannst du die "mittlere Kompression" oder eine Kompression mit niedrigerer Qualität benutzen, da es ja nur zum Testen der Map ist. Willst du aber deine Map veröffentlichen, solltest du die beste Kompression wählen, da ja viele Leute vor grösseren Map-Downloads zurückschrecken.

Nun musst du die *.zip Datei noch in eine *.pk3 Datei umwandeln. Dazu markierst du einfach deine Zip-Datei und drückst die "F2"-Taste. Nun kannst du die Datei umbenennen. Heisst sie vorher z.B. "meinemap.zip" nennst du sie in "meinemap.pk3" um.

[zurück zur Hauptseite](#)

183759



Acceleratorpad, Jumppad:

ACHTUNG: Sicher wuerdest du dich, weshalb ich hier beide Pad-Arten bespreche. Das kommt aher, da beide von der technischen Seite komplett identisch sind. Das einzige, was zwischen den beiden Pads unterschiedlich ist, sind die Texturen und eben die Richtung, in die du geworfen wirst.

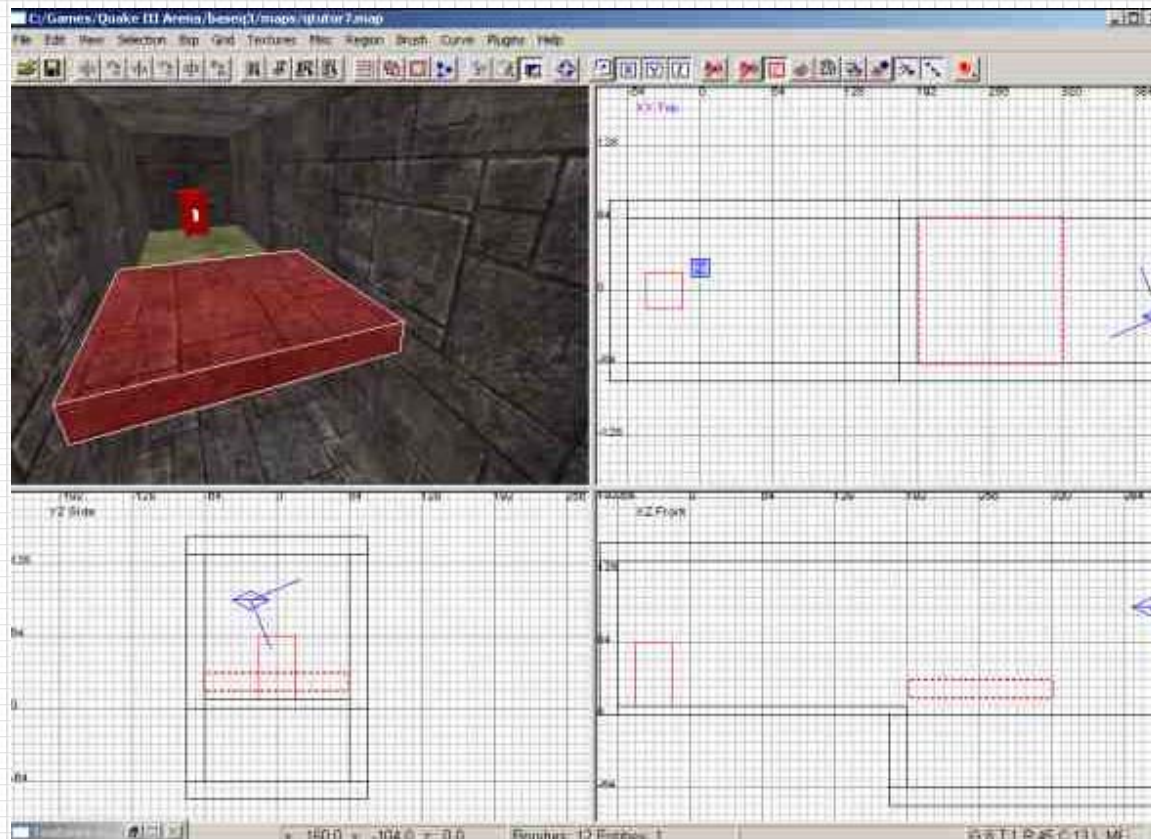
Zunächst mal das Acceleratorpad:

Verwendete Map: "qtutor6.map"

Ergebnis-Map: "qtutor7.map"

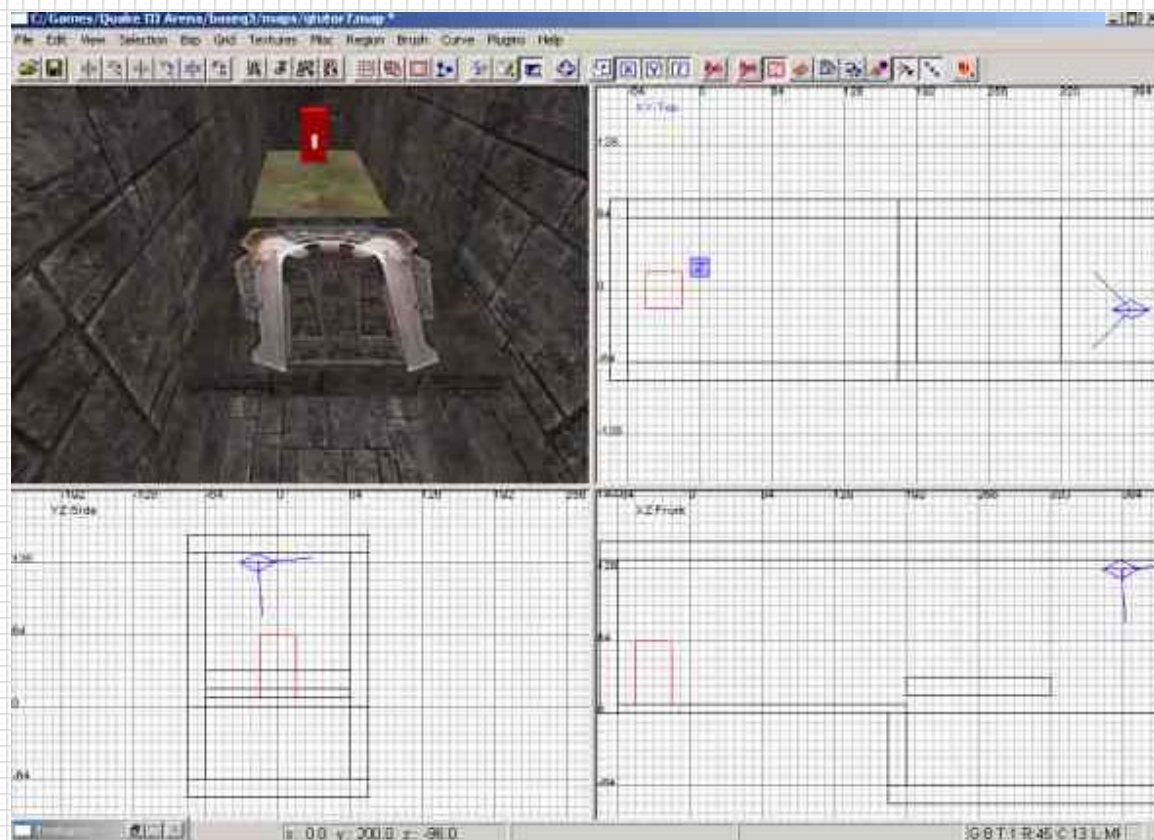
So, in diese Map wollen wir ein Acceleratorpad einbauen. Dazu habe ich dir eine Map gebaut, die nur aus einem langen Gang und einer Grube besteht. Nun wollen wir ein Pad bauen, mit dem der Spieler dann über die Grube geschleudert werden soll.

Dazu ziehst du dir als erstes einen Brush, der 128 Units lang, 128 Units breit und ca. 16 Units hoch ist. Ich habe den Brush noch so gesetzt, dass er schräg über der Kante zur Grube sitzt:

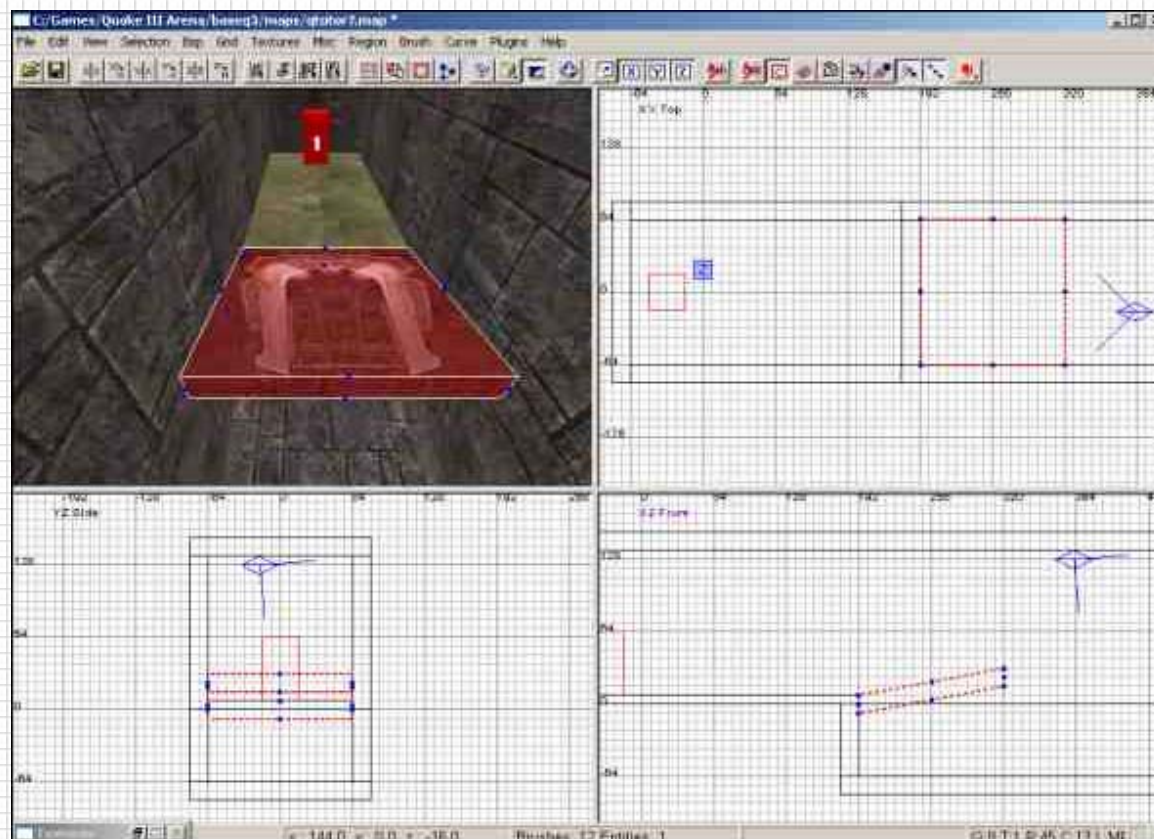


Nun drückst du "ESC" um den Brush zu deselektieren. Anschliessend markierst du die Oberseite dieses Brushes (um eine Seite anzuwählen, drückst du "STRG" + "SHIFT" + linke Maustaste) und belegst diese Oberfläche mit der Textur "sfx/launchpad_blocks18d". Jetzt musst du natürlich noch die Textur auf der Oberfläche ausrichten, dazu drückst du "S" um den "Surface Inspector" zu starten. Hier drückst du ziemlich in der Mitte des Fensters auf "FIT" - nun müsste die Textur mittig auf dem Brush liegen. Es kann sein, dass der Winkel der Textur nicht stimmt, d.h. das Pad zeigt in die falsche Richtung. Dazu drückst du wieder "S" und startest damit den "Surface Inspector". Hier spielst du nun mit der "Rotate"-Funktion herum. Dazu darf nur die Pad-Textur selektiert sein.

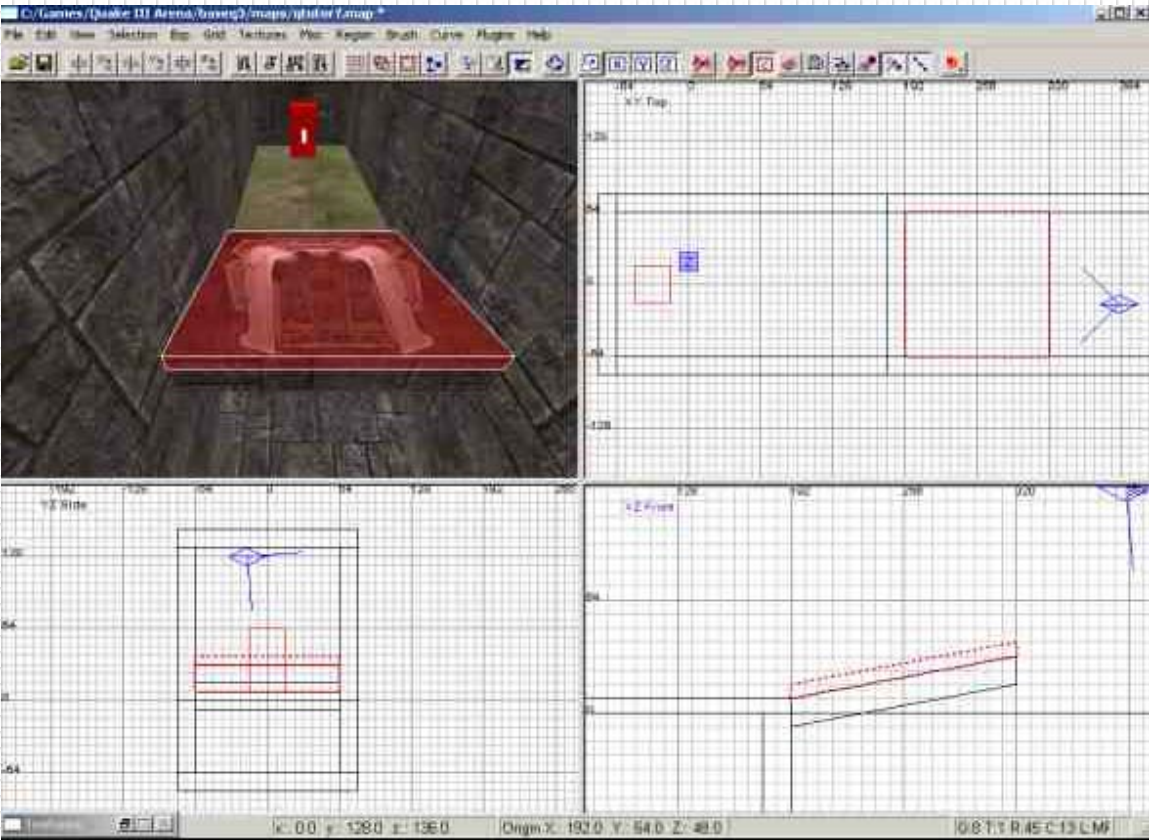
Nun sieht es ungefähr so aus:



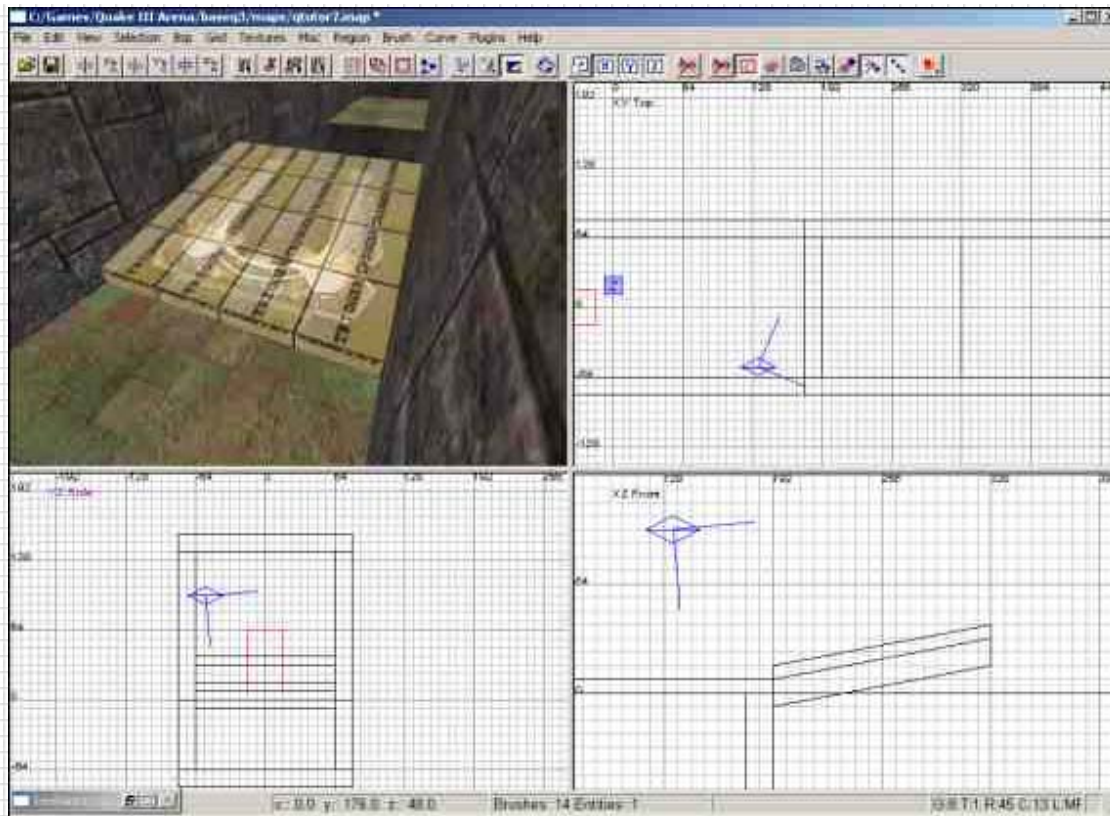
Mit dem Brush wären wir soweit fertig. Jetzt müssen wir natürlich noch den ganzen Brush etwas anders setzen, so dass es wie eine Rampe aussieht. Dazu wählst du den ganzen Brush an und drückst die Taste "E", um den "EDGE"-Modus zu aktivieren. Nun klickst du den unteren linken Punkt an und ziehst ihn 24 Units nach unten. Dann wählst du den oberen linken Punkt an und ziehst diesen soweit herunter, bis er die Kante zu der Grube berührt:



Wenn du willst, kannst du die rechten Punkte noch etwas nach oben ziehen, wenn dir der Winkel der Rampe noch etwas zu niedrig ist. Das tun wir jetzt aber nicht. Du drückst "E" um den "EDGE"-Modus zu verlassen und den Brush zu deselektieren. Nun selektierst du wieder den Brush und drückst die Taste "SPACE" (Leertaste) um den Brush zu kopieren. Nun setzt du ihn genau auf den andern Brush. Ich habe diesen oberen Brush noch etwas niedriger gemacht. Das kannst du tun, in dem du in der Draufsicht (das Fenster rechts unten) über den selektierten Brush klickst (mit der linken Maustaste) und dann die Maus langsam nach unten bewegst. Nun sieht es ungefähr so aus:

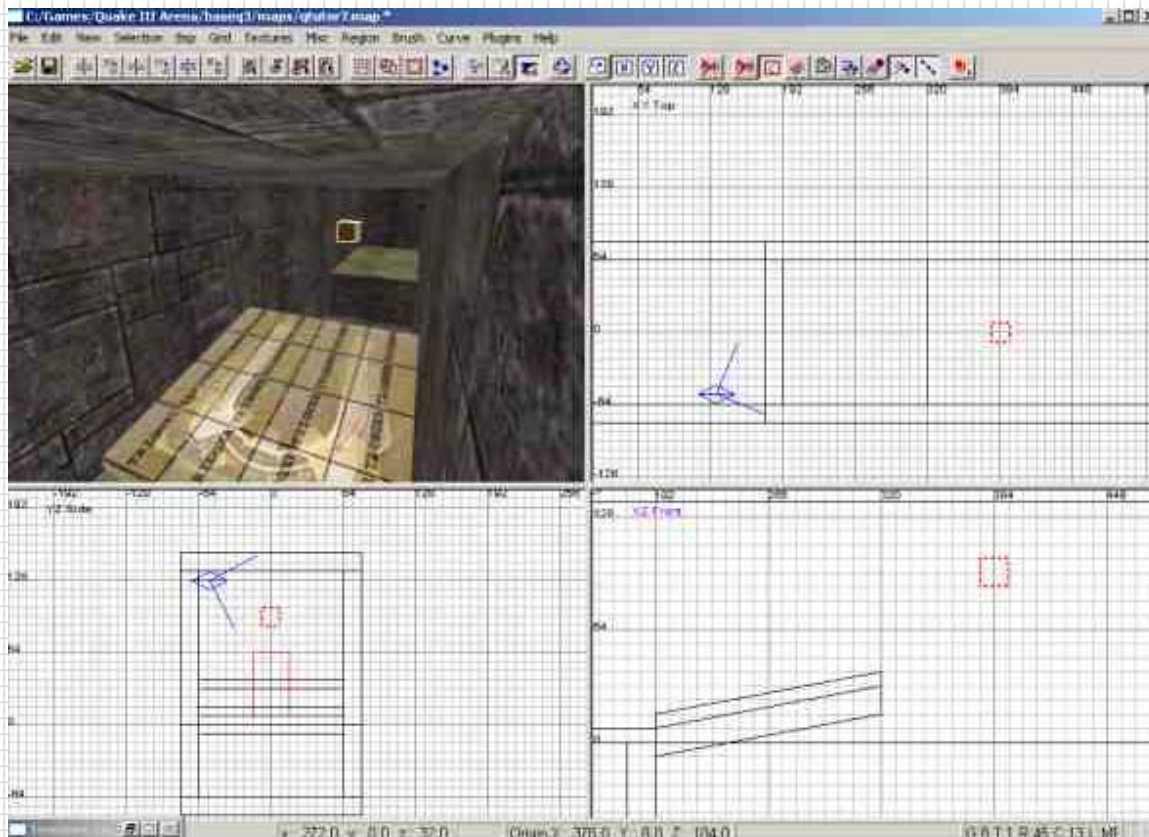


Jetzt wählst du in der Menüleiste "Textures" und da "common". Jetzt drückst du "T" um das Texturen-Fenster zu öffnen (wenn das nicht funktioniert, klicke einfach links unten auf den blauen Balken, auf dem "Textures" steht. Hier wählst du die Textur "trigger". Nun sieht es so aus:



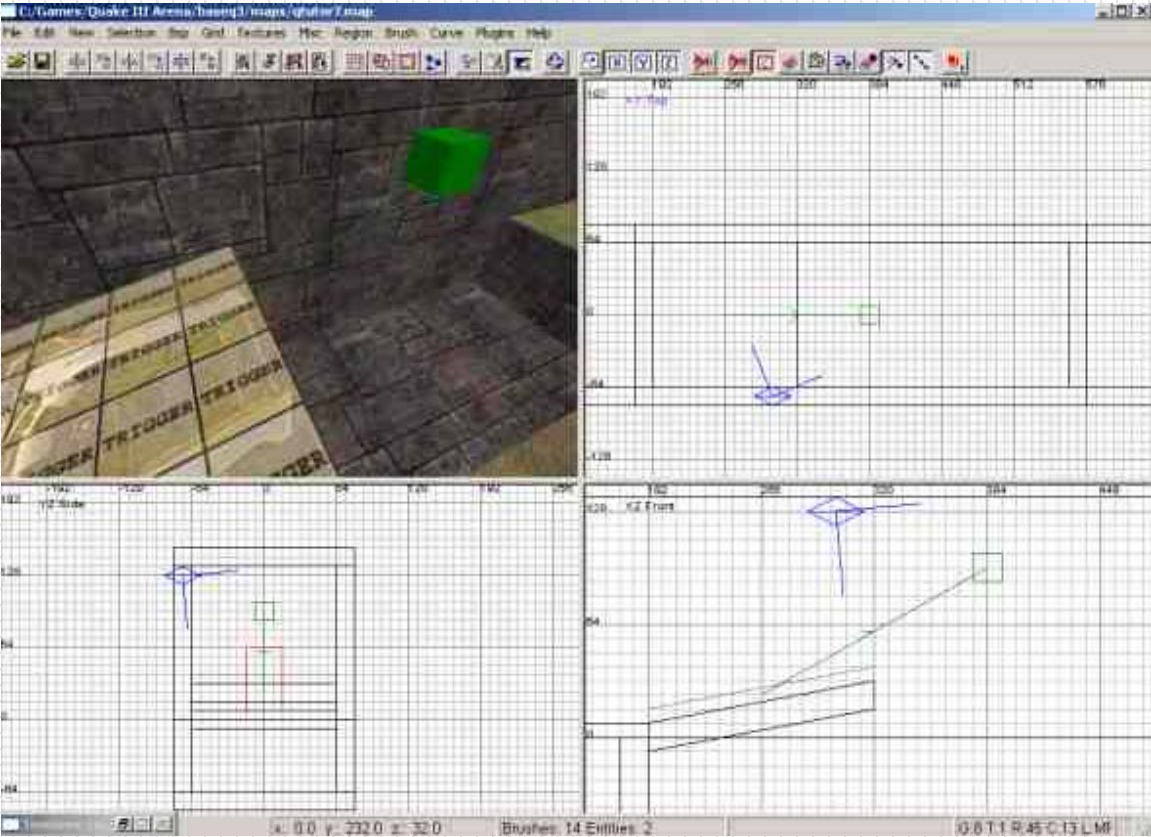
Jetzt drückst du in der Top Ansicht 2x die rechte Maustaste und wählst "trigger" und da "trigger_push".

Jetzt drückst du "ESC" um alles zu deselektieren. Nun drückst du nochmal in der Top Ansicht 2x die rechte Maustaste und wählst "target" und hier "target_position". Es erscheint ein kleines Kästchen. Dieses Kästchen ist das Ziel für unser Pad, damit es auch weiss, wo es den Spieler hinbefördern soll. Also setzen wir dieses Kästchen zunächst vor unser Pad, da ja das Pad den Spieler nach vorn befördern soll:



Wie du siehst, habe ich das target gleich noch etwas nach oben verschoben, da unser Spieler ja auch noch etwas nach oben geschleudert werden soll.

Jetzt drückst du "ESC" und deselektierst alles. Jetzt wählst du erst den Trigger-Brush an und danach den target_position. Jetzt drückst du "STRG" (und hältst es gedrückt) und "K". Es erscheint ein Faden, der den Trigger-Brush mit dem target verbindet:



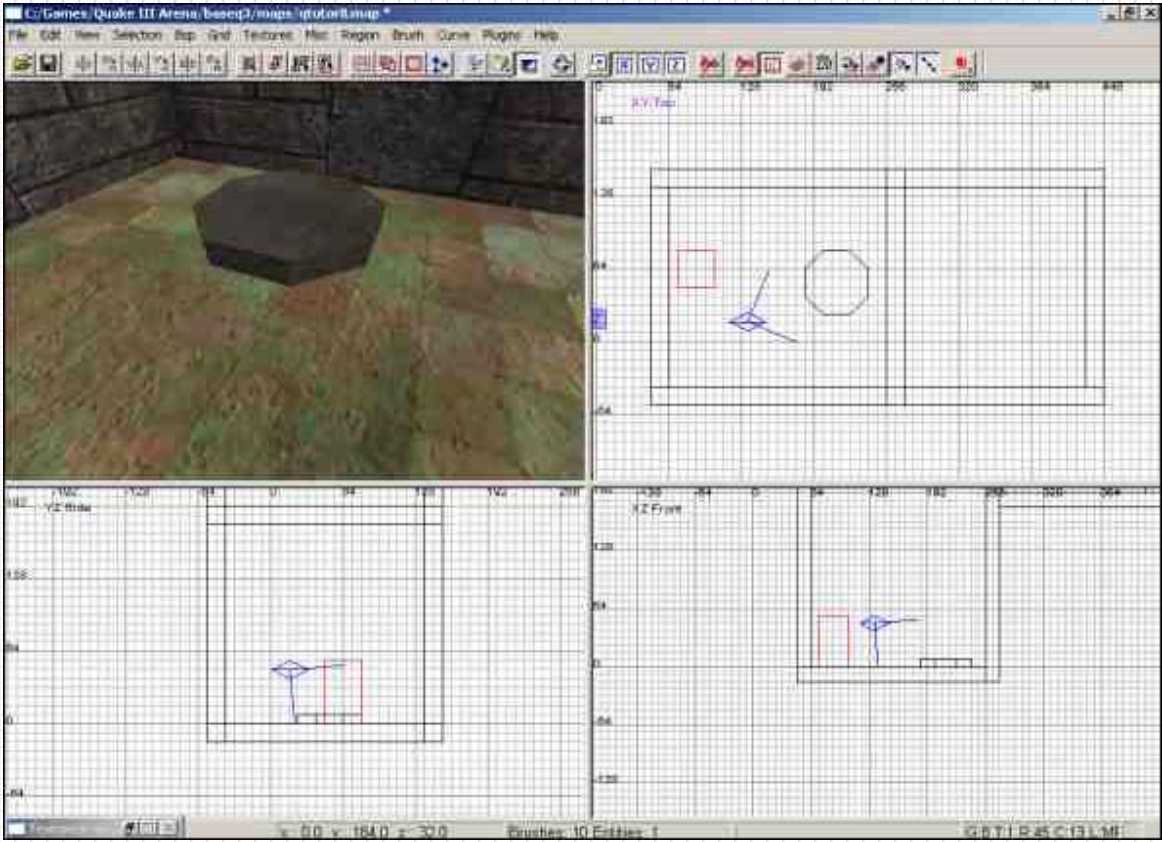
Natürlich kannst du den target_position nachträglich noch verschieben, z.B. solltest du ihn noch etwas weiter über die Grube setzen, da so der Spieler natürlich weiter fliegt.

Jetzt bauen wir gleich noch ein Jumppad:

verwendete Map: "qtutor8.map"
Ergebnis-Map: "qtutor9.map"

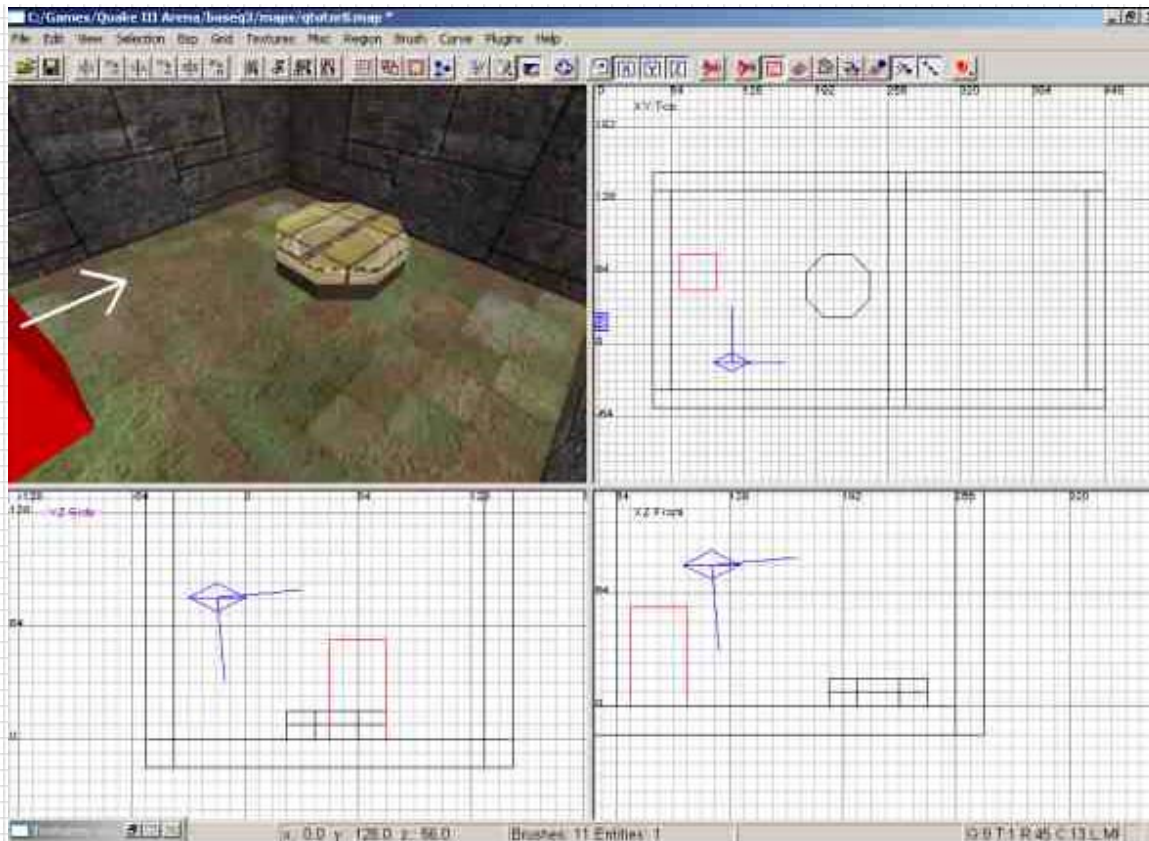
Nun hab ich eine Map gebaut, in der der Spieler von der unteren Ebene auf die obere Ebene kommen soll - und das natürlich über ein Jumppad. Dazu brauchen wir jedoch zunächst wieder einen Brush. Diesmal machen wir aber einen 8-eckigen Brush, dazu machst du dir einen ganz normalen, vierseitigen Brush, der 8 Units hoch ist.

Nun klickst du noch schnell in die Top Ansicht, um dieses Fenster zu aktivieren. Jetzt wählst du in der Menüleiste den Punkt "Brush" und wählst "8-sided". Nun hast du einen 8-seitigen Brush. Nun brauchen wir für diesen Brush eine Textur. Dazu wählst du in der Menüleiste "Textures" und da "base" und da als Unterordner "base_wall". Nun drückst du wieder "T" und wählst die Textur "concrete_dark" und drückst "ESC", um alles du deselektieren:

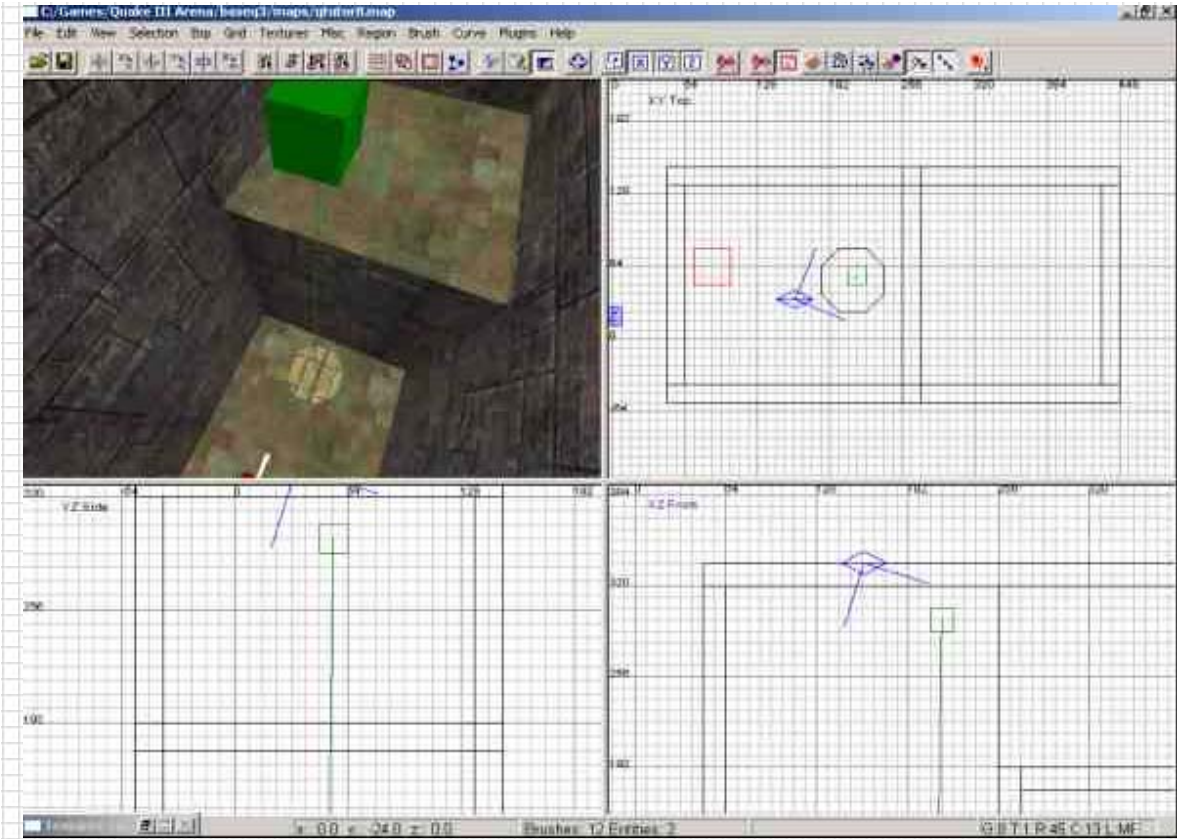


Jetzt selektierst du die Oberseite dieses Brushes, schliesslich wollen wir daraus unseren Jumppad machen. Also wählst du wieder in der Menüleiste den Punkt "Textures" und da "sfx". Hier wählst du die Textur "metalbridge06_bounce". Nun musst du wieder diese Textur auf dem Brush ausrichten. Dazu drückst du "S" um den "Surface Inspector" aufzurufen und drückst "FIT". Jetzt sitzt die Textur wieder perfekt auf dem Brush.

Nun selektierst du den ganzen Brush und drückst "Space" (Leertaste) um diesen Brush zu kopieren. Nun setzt du diesen Brush genau auf unseren alten Brush und wählst in der Menüleiste den Punkt "Textures", dann "common" und hier die Textur "trigger". Nun drückst du "ESC" um alles zu deselektieren, jetzt sieht es so aus:



So, jetzt selektierst du wieder den Brush mit der "Trigger"-Textur und drückst in der Top Ansicht 2x mit der rechten Maustaste und wählst "trigger" und da "trigger_push". Damit wäre unser Pad fertig. Wir brauchen allerdings wieder unser Ziel, damit das Pad weiss, wohin die Reise geht. Also brauchen wir wieder ein "target_position". Dazu drückst du wieder in der Top Ansicht 2x mit der rechten Maustaste und wählst "target" und dann "target_position". Diesmal setzen wir dieses target lediglich ziemlich weit nach oben, schliesslich soll der Spieler nur nach oben katapultiert werden. Jetzt drückst du "ESC", damit alles deselektiert wird. Nun selektierst du wieder zuerst den "Trigger"-Brush, anschliessend das "target_position". Jetzt drückst du "STRG" (und hältst es gedrückt) und "K". Es erscheint ein Faden, der den Trigger-Brush mit dem target verbindet. Natürlich kannst du auch hier im Nachhinein das target_position verschieben, wenn der Spieler noch nicht hoch genug geschleudert wird. Dazu markierst du einfach NUR das target_position und schiebst es dorthin, wo du es haben willst:



WICHTIG: Wie du sicher weisst, gibt es in RtCW bekanntlich keine Jump pads und auch keine Accelerator pads. Dennoch gibt es den "trigger_push". Das "target_position" gibt es allerdings nicht, dafür kannst du aber einfach unter "info" das Entity "notnull" verwenden.

Ich habe dir ja oben schon gesagt, dass es auch nur auf diese beiden Dinge ankommt - du kannst also auch in RtCW die beiden Pad-Arten machen - jedoch musst du auf andere Texturen zurückgreifen.

[zurück zur Hauptseite](#)



Teleporter:

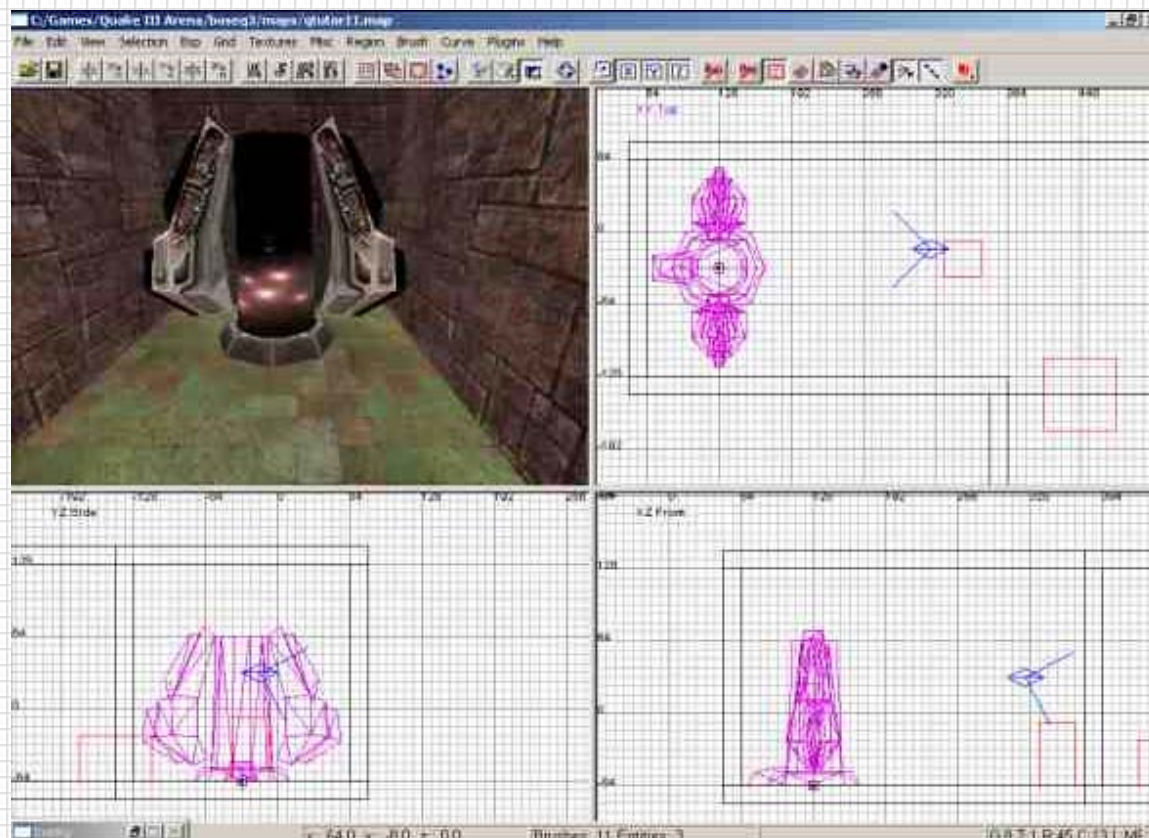
Verwendete Map: "qtutor10.map"

Ergebnis-Map: "qtutor11.map"

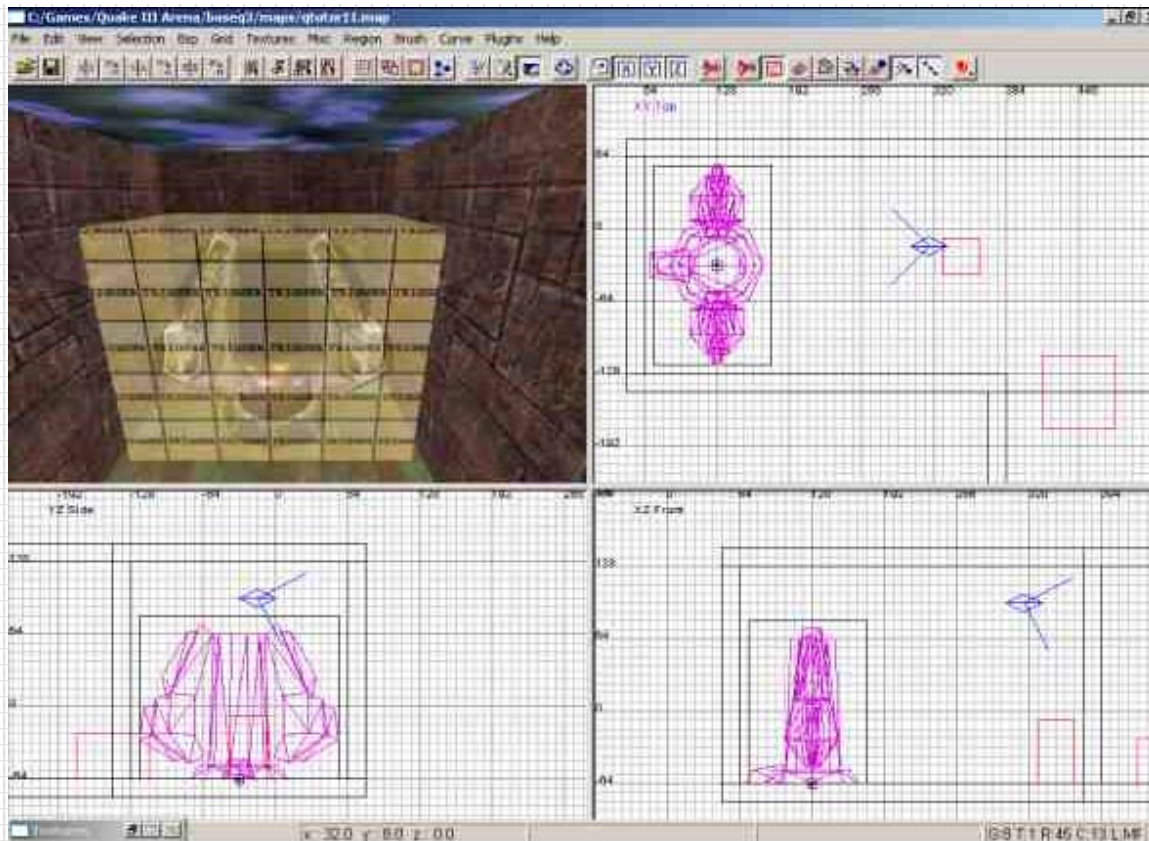
So, ich habe hier eine Map gebaut, in der wir einen Teleporter bauen wollen. Das Ziel soll es sein, dass der Spieler genau zum Rocket-Launcher teleportiert wird.

Wir wollen den Teleporter an die Wand bauen, auf die der Spieler direkt am Anfang schaut (das erkennst du an dem weißen Pfeil, der am Playerstart die Sichtrichtung angibt). Also fangen wir an. Zunächst brauchen wir ein geeignetes Aussehen für den Teleporter. Hier müssen wir mal nicht selbst Hand anlegen, sondern verwenden ein Model. Dazu klickst du in der Top Ansicht 2x mit der rechten Maustaste und wählst "misc" und hier als Unterpunkt "misc_model".

Es erscheint ein neues Fenster. Du öffnest den Ordner "mapobjects" und dort wählst du "teleporter" und dann noch einmal "teleporter.md3". Nun erscheint das Teleporter-Model in deiner Map. Nun musst du es noch richtig ausrichten, dazu öffnest du das Texturen-Fenster und drückst "N". Hier kannst du unten mit den 8 Zahlen die Richtung des Teleporters bestimmen:



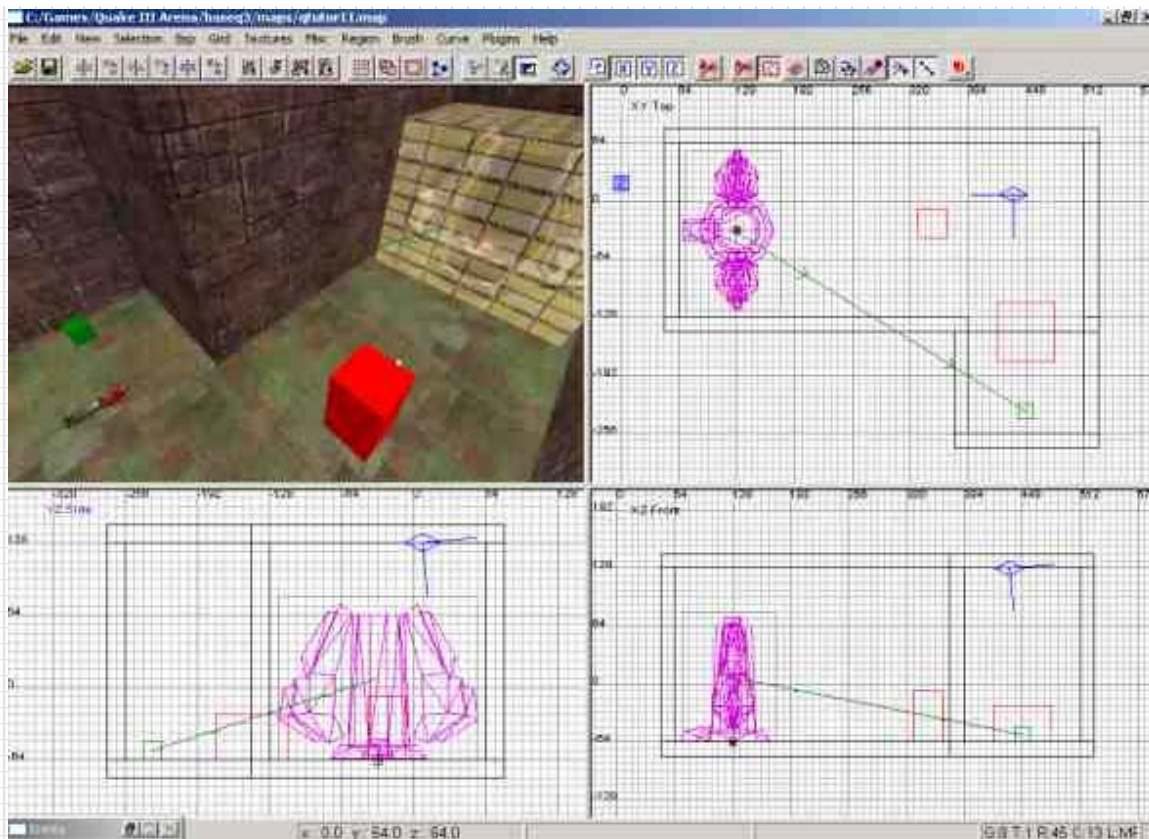
Nun fängt es aber erst richtig an. Doch davor drückst du noch einmal "ESC" um alles zu deselektieren. Nun machst du einen Brush über das Teleporter-Model, dass der Brush das Model komplett einhüllt. Dann wählst du in der Menüleiste den Punkt "Textures" und hier "common". Danach drückst du "T" und öffnest damit das Texturen-Fenster. Dann wählst du die Textur "trigger". Jetzt drückst du "ESC" um alles zu deselektieren. Nun sieht das ungefähr so aus:



So, jetzt musst du wieder den Trigger-Brush selektieren. Nun drückst du in der Top Ansicht 2x die rechte Maustaste, hier wählst du "trigger" und nun "trigger_teleport". Damit hätten wir den Teleporter fertig. Wie beim vorigen Thema brauchen wir jedoch noch ein Zielpunkt, dass der Teleporter auch weiss, wohin der Spieler teleportiert werden soll. Bevor wir aber damit anfangen, deselektierst du nocheinmal alles, indem du die "ESC"-Taste drückst.

Nun drückst du wieder in der Top Ansicht 2x die rechte Maustaste und wählst "target" und als Unterpunkt "target_position". Nun erscheint wieder ein Kästchen, dies ist das Entity für das "target_position". Nun drückst du wieder "ESC" um alles zu deselektieren.

Jetzt selektierst du zuerst die Trigger-Textur (pass auf, dass du nicht ausversehen noch das Model selektierst, sonst funktioniert später gar nichts) und dann den target_position. Nun drückst du "STRG" + "K". Natürlich müssen wir noch das target_position so ausrichten, dass es auf den Rocket-Launcher zeigt. Dazu setzt du es etwas weiter ins Eck und öffnest das Texturen-Fenster. Hier drückst du "N" und kannst mit den 8 Zahlen wieder die Richtung bestimmen, in die der teleportierte Spieler schaut. Nun sieht es ungefähr so aus:



Übrigens, es macht nichts aus, wenn der Faden dabei zwischendurch das Level verlässt. Was aber nicht passieren darf, dass der Spieler plötzlich ausserhalb der Map startet, dann gibt es nämlich ein Leak.

Sobald hier der Spieler den Trigger-Brush berührt, wird er teleportiert, d.h. er kommt erst garnicht an das Model heran. Willst du aber, dass nur das Model teleportiert, musst du den Trigger-Brush aussenherum natürlich verkleinern. Wenn du den Teleporter noch realistischer machen willst, kannst du z.B. die "Hörner" an dem Teleporter zuclippen, dass der Spieler hier nicht mehr durchlaufen kann.

Du brauchst also eigentlich keine besondere Grundlage für Teleporter, es bleibt dir zur Wahl, ob du dir selbst einen Teleporter-Brush machst, oder auf Models zurückgreifst. Z.B. habe ich in meiner Q3-Map "House Of Motion" simple Kreuze zu Teleportern umfunktioniert.

WICHTIG: In RtCW gibt es keinen "target_position" mehr. Aber als Ersatz für diesen "target_position" kannst du einfach unter "info" das Entity "notnull" verwenden.

[zurück zur Hauptseite](#)

183759



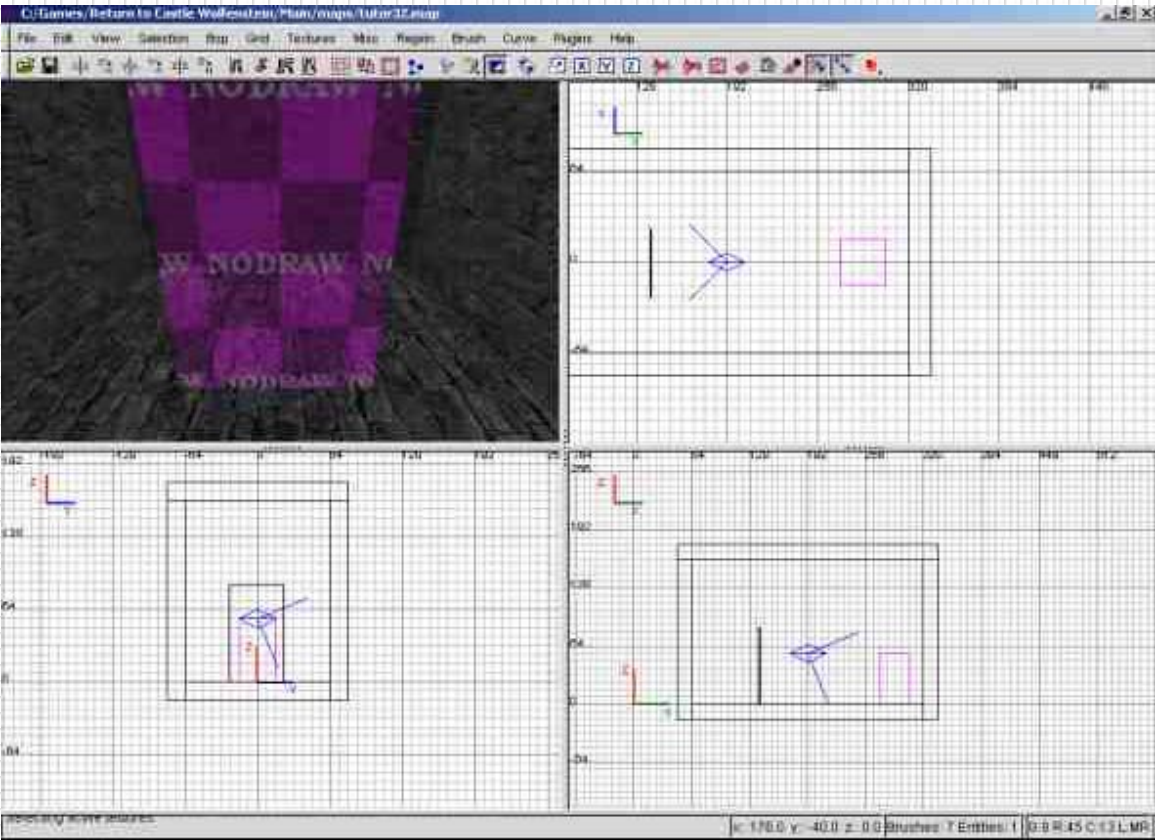
Feuer:

Verwendete Map: "tutor31.map"
Ergebnis-Map: "tutor32.map"

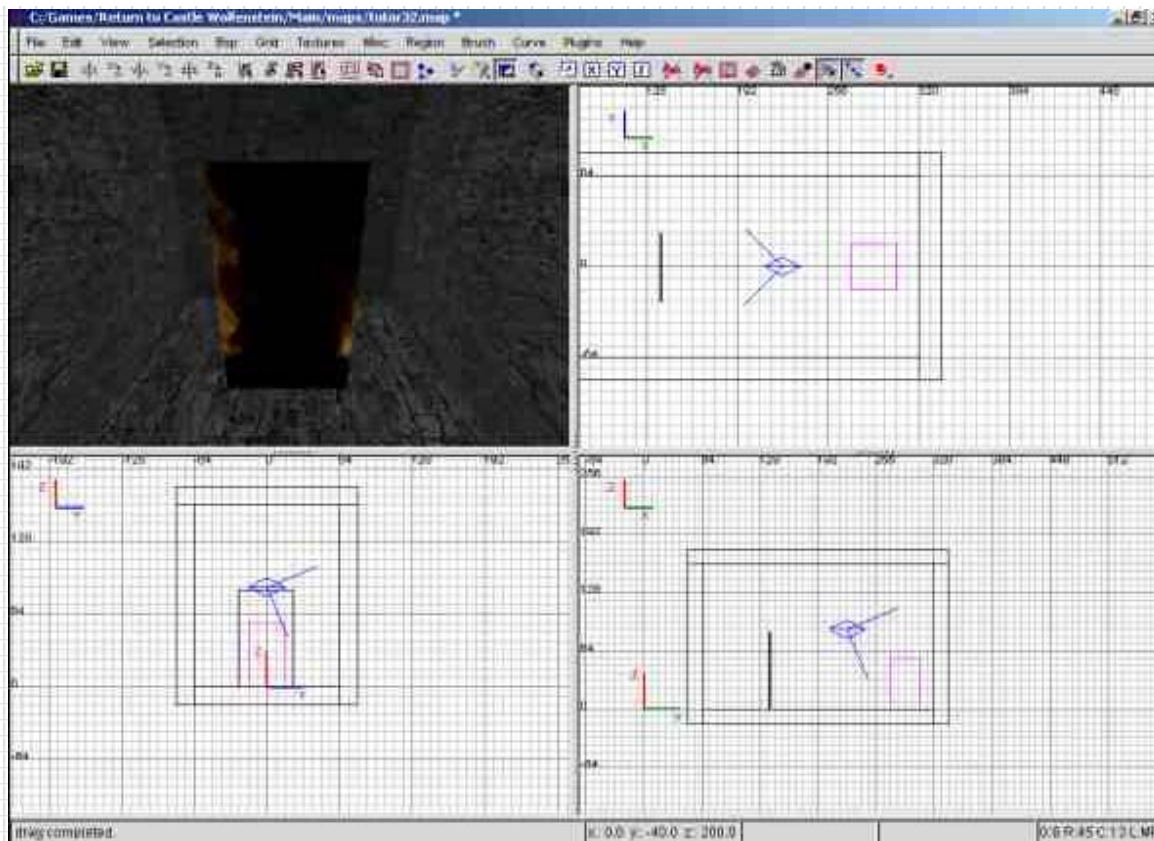
Hier will ich dir zeigen, wie du Feuerstellen machen kannst. Dazu habe ich dir eine Map gebaut, die noch keine Lichtquelle hat.

Für das Feuer brauchen wir natürlich zuerst einen Brush, der später unser Feuer werden soll. Diesen Brush wollen wir natürlich möglichst dünn halten, da Feuer ja eigentlich kein Volumen hat und es so unrealistisch aussehen würde. Also drückst du die Taste "1" um das Grid auf die feinste Einstellung zu setzen.

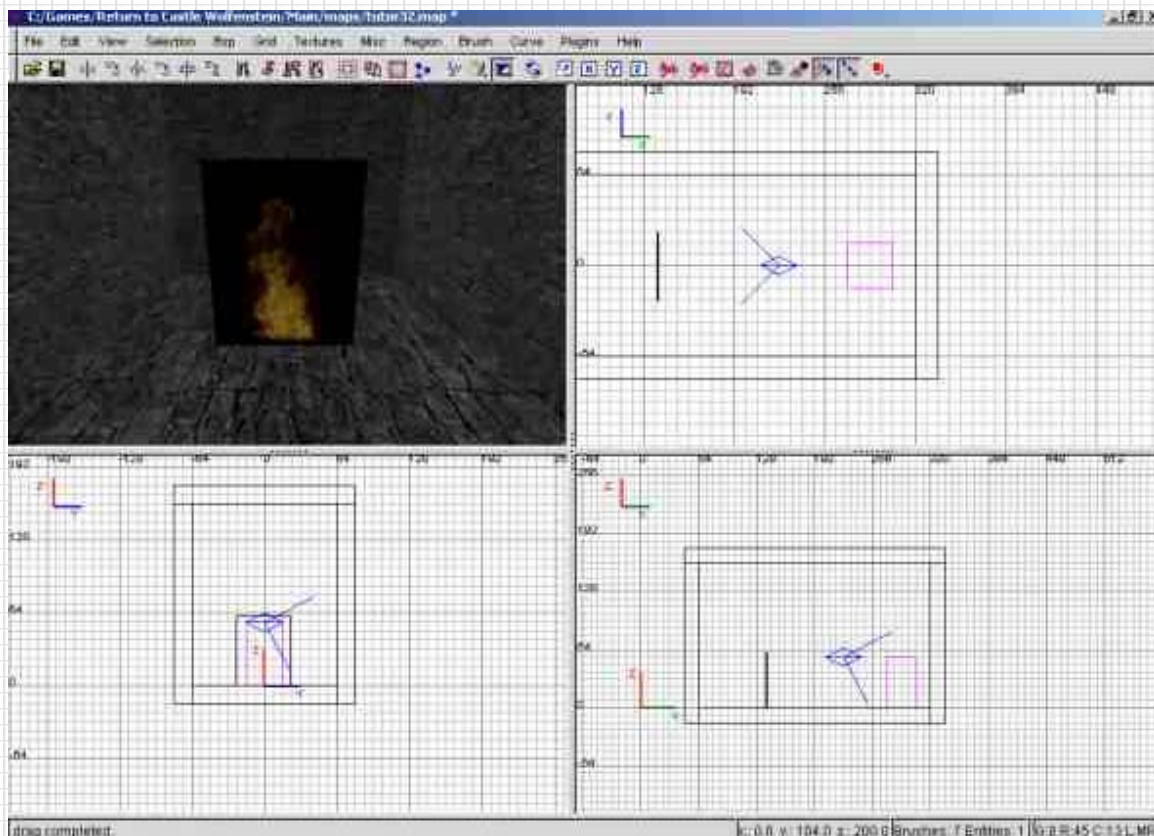
Nun ziehst du dir einen Brush, der genauso gross ist, wie später dein Feuer sein soll. Nun wählst du den Brush an und wählst in der Menüleiste den Punkt "Textures" und dort "common". Hier suchst du dir die Textur "nodraw". Du klickst auf diese Textur und dein Brush müsste nun mit der Textur "Nodraw" belegt sein. Nun sieht es ungefähr so aus:



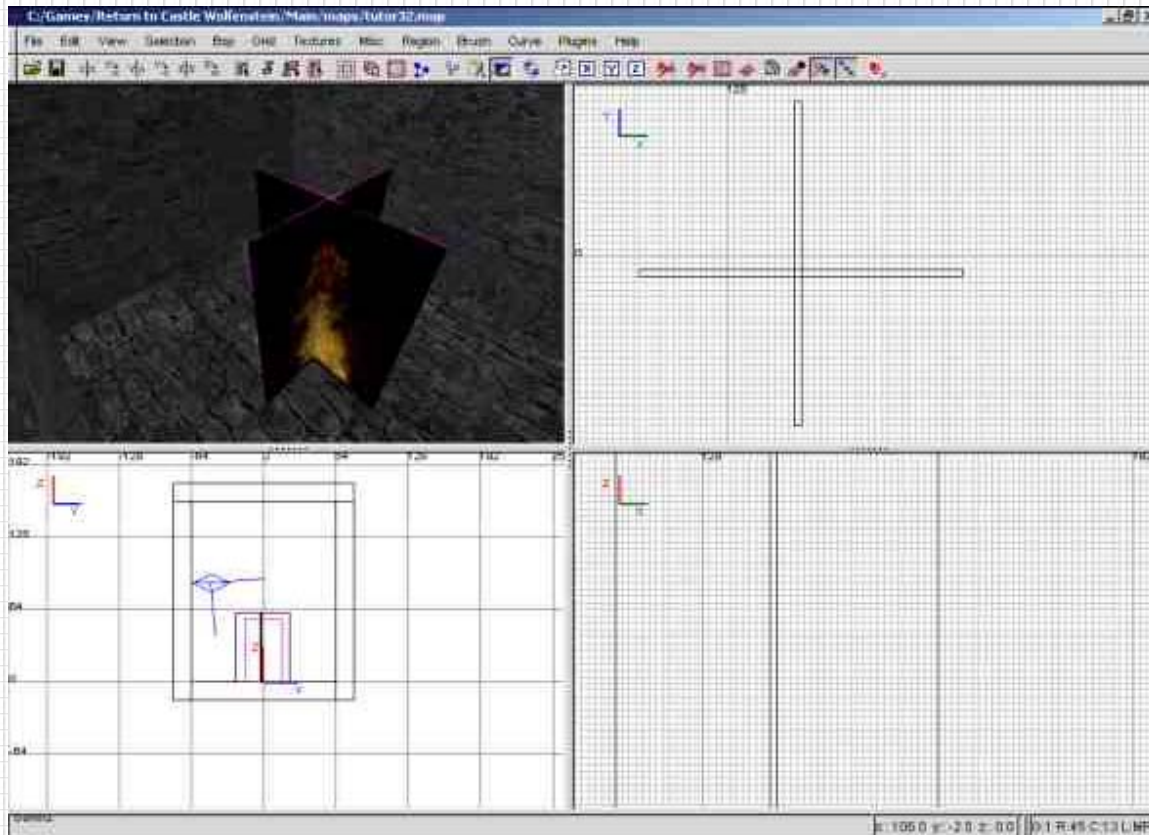
Jetzt selektierst du die Vorderseite des Brushs, dazu drückst du die Tasten "STRG" + "SHIFT" +linke Maustaste. Nun brauchen wir aber noch eine Feuertextur für unser Feuer. Dazu wählst du in der Menüleiste den Punkt "Textures" und dort "sfx". Jetzt drückst du "T" um das Texturen-Fenster zu öffnen. Hier suchst du dir eine Feuer-Textur aus und klickst diese an. Ich habe für unser Beispiel die Textur "sfx/flame1" entschieden.



Nun musst du natürlich die Textur noch auf den Brush "fitten". Dazu drückst du die Taste "S" um den "Surface Inspector" aufzurufen. Hier wählst du "FIT". Nun "hängt" deine Flamme aber über dem Boden, das wollen wir natürlich nicht. Dazu klicken wir jetzt die Feuertextur an (dazu drückst du die Tasten "STRG" + "SHIFT" + linke Maustaste) und passt diese so an, dass es so aussieht:



Wie du siehst, habe ich den Brush noch nachträglich etwas verkleinert, da sonst oben an der Flammentextur noch eine weitere Flamme auftauchen würde, das wollen wir aber nicht. Nun selektierst du den ganzen Brush und drückst die "Space"-Taste (Leertaste). Damit hast du den Brush kopiert. Nun stellst du diesen neuen Brush im 90°-Winkel gegen den alten, dass es so aussieht:



Um die beiden Brushes genau anzupassen, habe ich wieder in das feinste Grid gewechselt (mit der Taste "1"). Jetzt drückst du "ESC" um alles zu deselektieren.

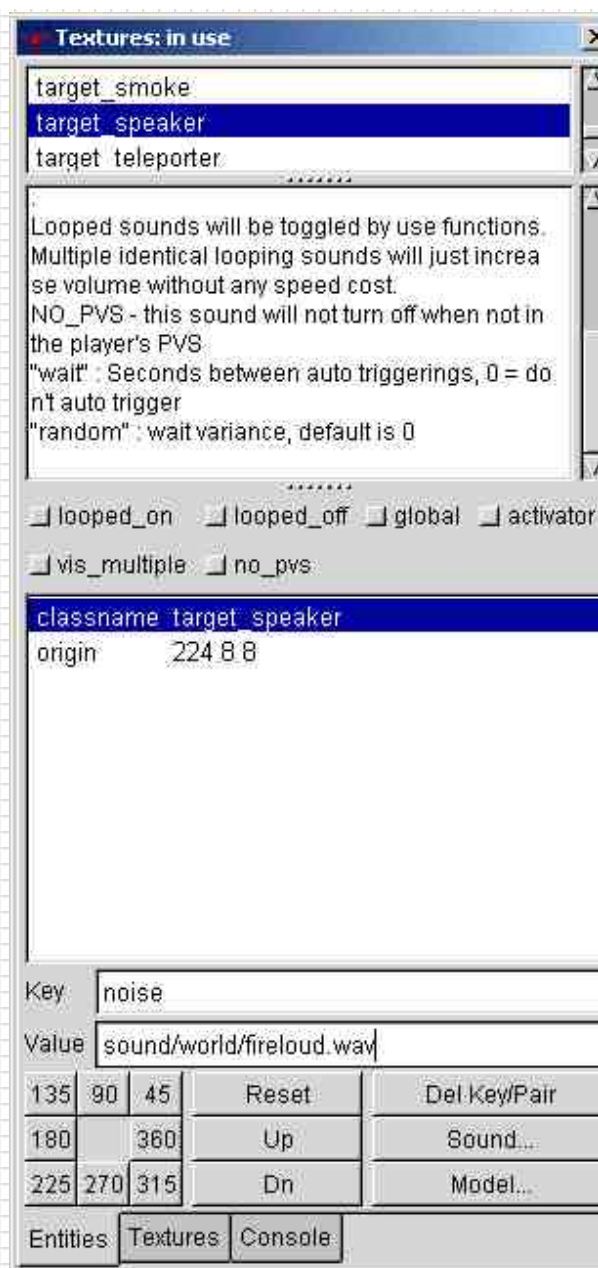
Natürlich kannst du in diese Flamme noch ein Dlight, Light_junior oder ein Corona setzen, dazu musst du einfach das entsprechende Entity erstellen und zwischen die Flammen schieben.

Nun kannst du die Map schon compilieren, aber das Feuer hat keinen Sound. Dazu klickst du in der Top Ansicht 2x mit der rechten Maustaste und wählst "target" und als Unterpunkt "target_speaker". Nun drückst du die Taste "N", da wir noch die Eigenschaften des Entities ändern müssen:

Erklärung:

looped_on: der Sound ist von Beginn an aktiv und wird immer wiederholt

looped_off: der Sound ist von Anfang an NICHT aktiv, d.h. er muss getriggert werden, dann wird



er gestartet und wiederholt sich auch immer.

global: jeder kann den Sound hören

activator: nur der, der den Sound getriggert hat, hört den Sound auch.

Sicher hast du erkannt, dass wir nur "looped_on" brauchen, den Rest benötigen wir für andere Sounds.

Du musst aber den folgenden Key und dessen Value eingeben und hinterher "ENTER" drücken.

- key: noise
value: sound/world/fireloud.wav

Nun musst du noch das Entity ins Feuer schieben, schliesslich soll ja das Feuer den Sound von sich geben. Wenn du willst, kannst du noch ein Licht in die Map setzen, falls es dir zu dunkel ist.

[zurück zur Hauptseite](#)

183759

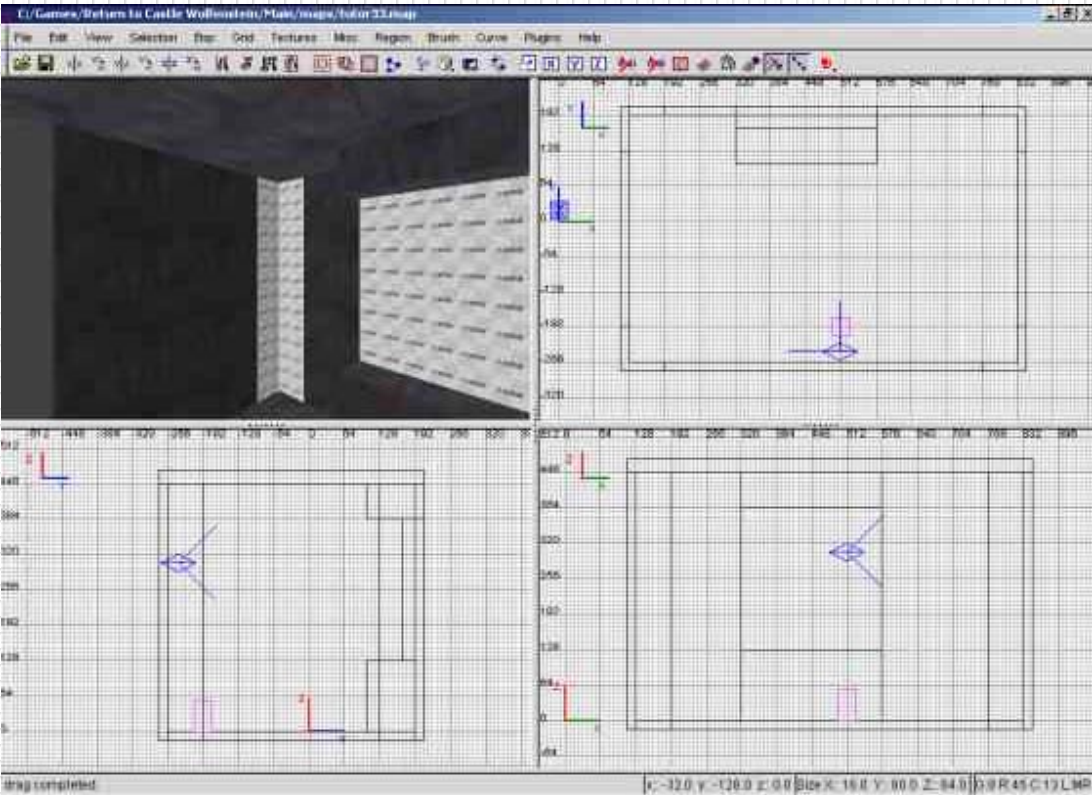


Curved Surfaces:

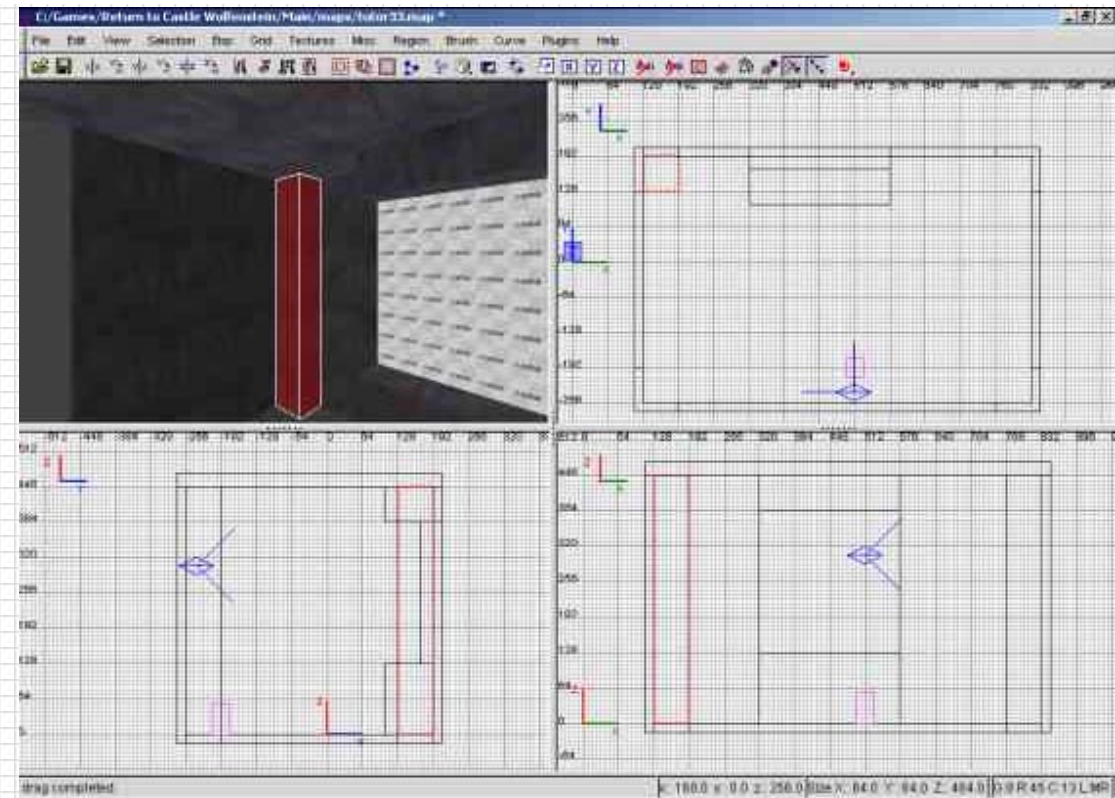
Verwendete Map: "tutor33.map"
Ergebnis-Map: "tutor34.map"

So, in diesem Thema geht es um die gekrümmten Oberflächen, also die "curved Surfaces". Stellenweise verlange ich dir hier wieder einiges ab, aber - wir schaffen das schon. Ich habe dir schon einen kleinen Raum gemacht, hier wollen wir ein paar curved surfaces einbauen. Wenn du willst, kannst du diesen Raum selbstverständlich auch selbst mappen, aber ich hab ihn schon so vorbereitet, dass du dich nur auf die curved Surfaces konzentrieren musst - also wäre es auch geschickt, wenn du mit diesem Raum arbeiten würdest ;), da du dich hier nicht mit dem nachmappen aufhalten musst.

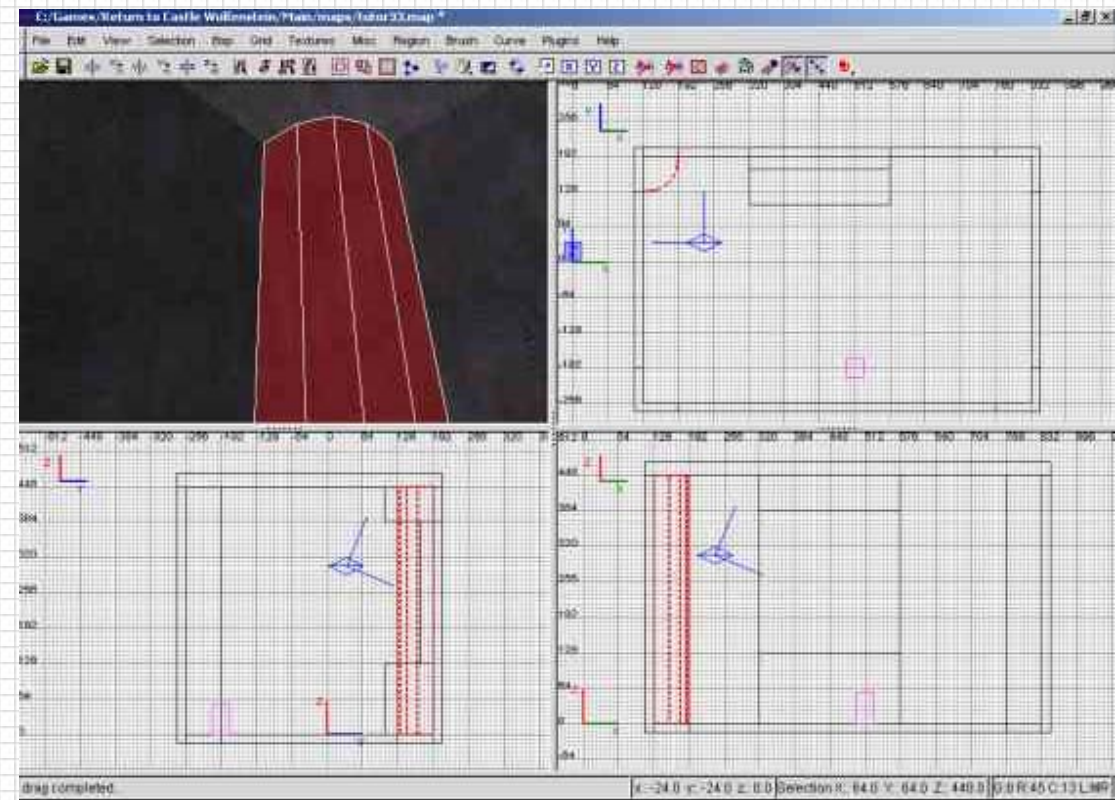
Wenn du die Map im Radiant geladen hast, wirst du feststellen, dass hier die Ecken offen sind, d.h. der Raum ist nicht zum Void hin geschlossen. Aber eine Ecke ist mit einer neuen Textur überzogen, der "caulk"-Textur. Diese ist für alle curved Surfaces zwingend notwendig, da es sonst zu Darstellungsfehlern hinter unseren curves kommen kann - und das wollen wir natürlich verhindern. Da ich an anderer Stelle nochmals auf diese Textur eingehe, will ich hier nur sehr oberflächlich auf ihre Eigenschaften eingehen. Die "Caulk"-Textur dient der curve sozusagen als Hintergrund, also als eine Art Aufhängung. Natürlich darf diese Textur hinterher nichtmehr sichtbar sein, da sie auch zu Darstellungsfehlern führt:



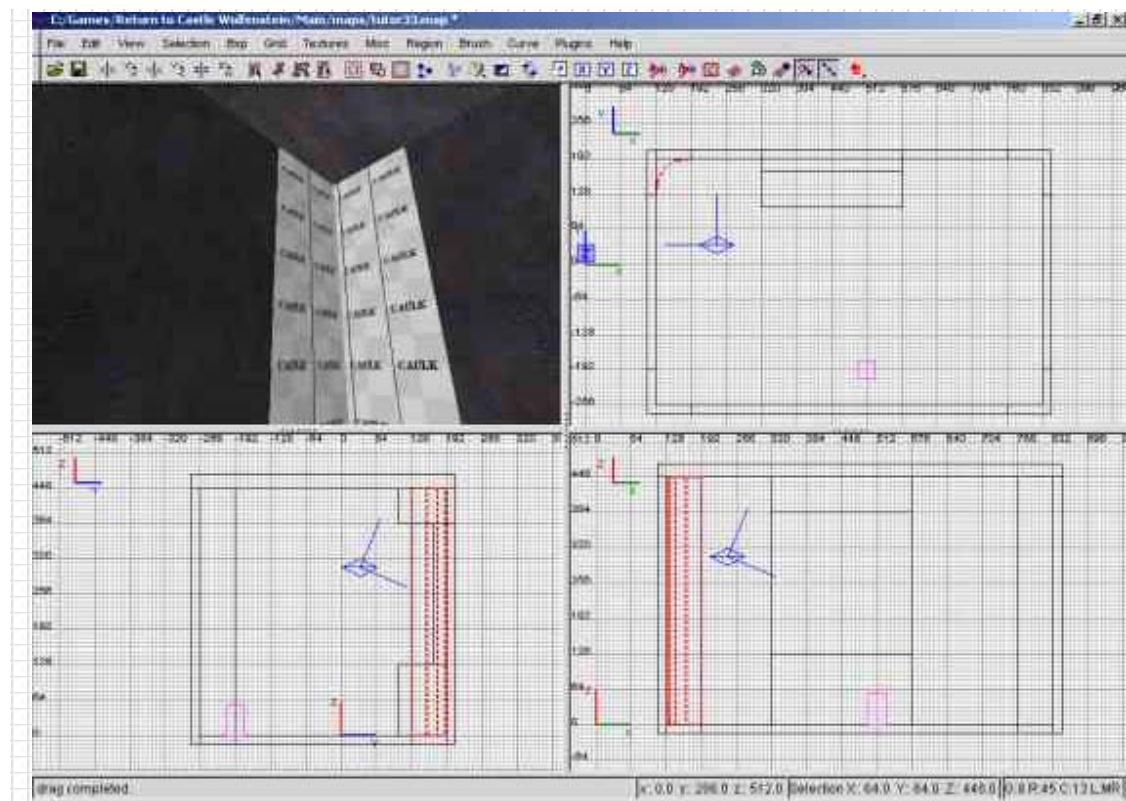
Nun fangen wir aber an - wir ziehen uns einen neuen Brush, der komplett vom Boden bis zur Decke reicht und bündig an die Brushes mit der "Caulk"-Textur liegt:



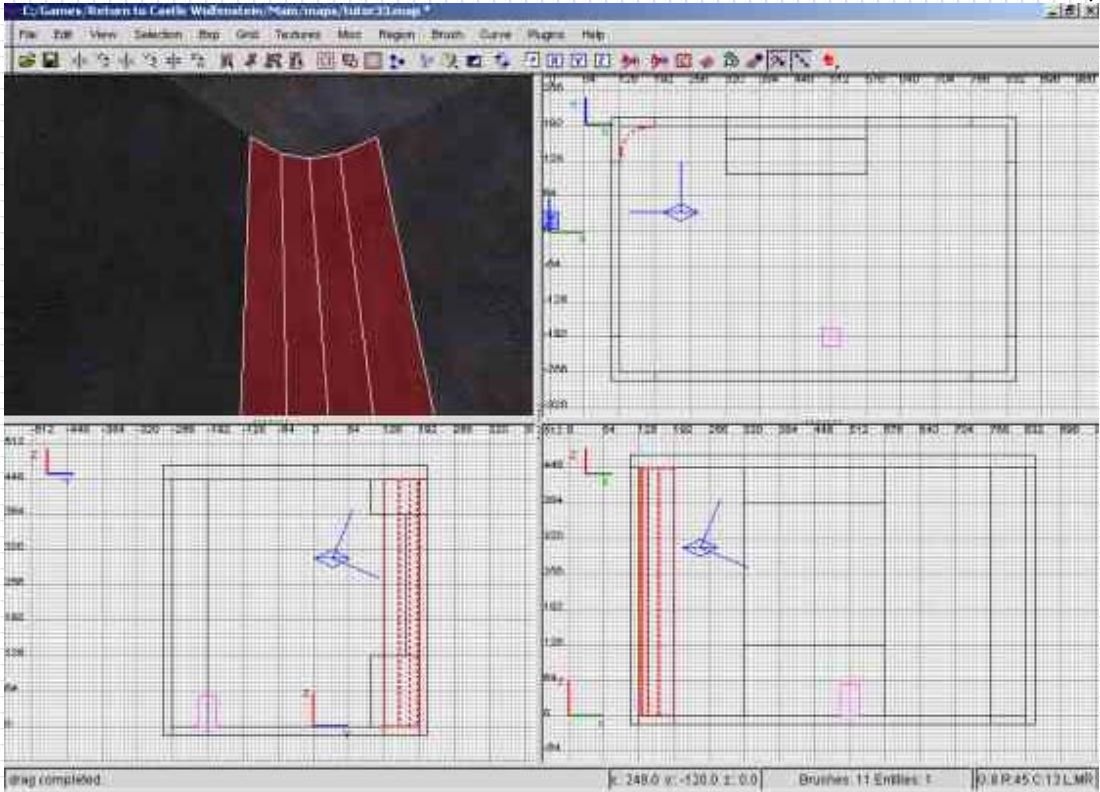
Nun wählen wir oben in der Menüleiste den Punkt "Curve" und in diesem Pull-Down-Menü wählen wir den Punkt "bevel". Das ist 1/4 von einem ganzen Kreis. Wenn er bei dir noch nicht richtig sitzt, kannst du diesen "curve"-Brush natürlich über die verschiedenen Buttons (x-axis Rotate, y-axis Rotate und z-axis Rotate) so ausrichten, wie du willst. Hier passt er nicht so ganz:



Eigentlich könnten wir die curve ja so lassen, aber eigentlich sollte der Brush genau andersrum sitzen, also drehen wir ihn gleich mal:



So, jetzt haben wir die curve schon richtig ausgerichtet, jedoch passt nun die Textur nicht, man sieht die "Caulk"-Textur durchscheinen - der Grund ist einfach die Textur ist auf der falschen Seite. Nun wählen wir wieder im Menü den Punkt "curve", als Unterpunkt Matrix

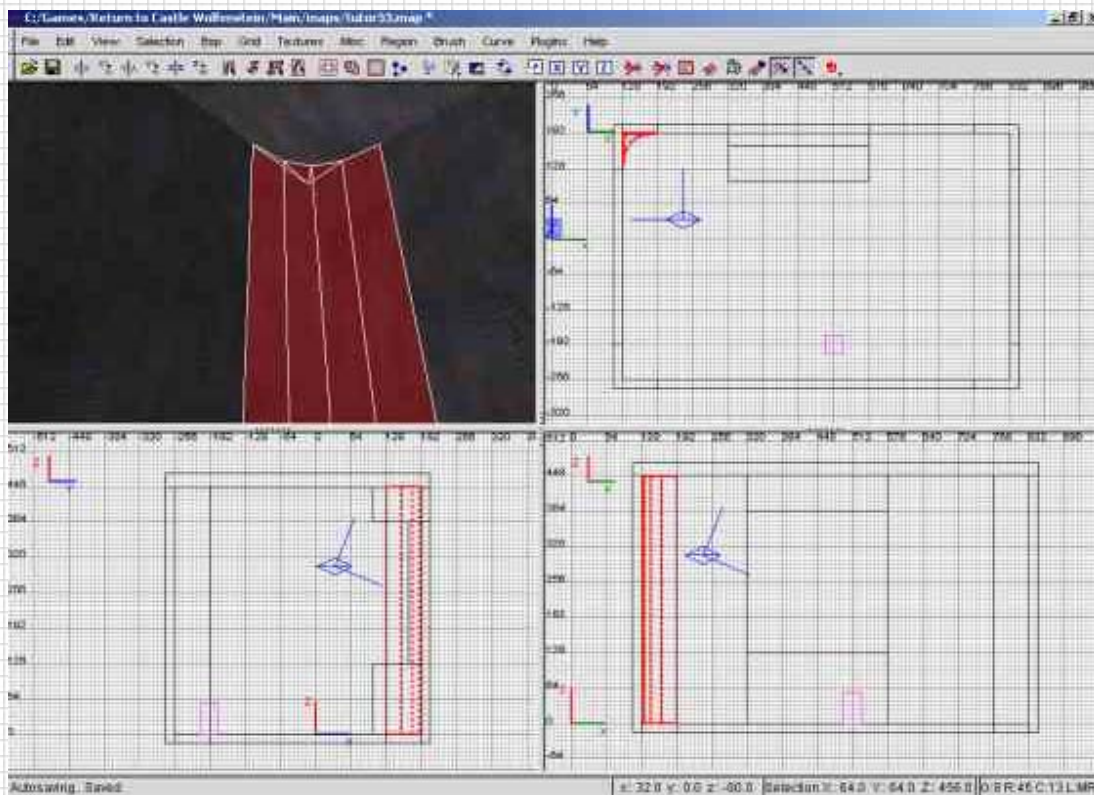


und hier den Punkt Invert.

Nun sind wir noch nicht ganz fertig - wir müssen noch ein "Cap" auf unsere Curve legen. Dazu wählen wir im Menü "curve" den Unterpunkt "cap selection". Es erscheint dieses Menü:



Dieses Menü ist sehr wichtig, ich erkläre hier mal eben die Eigenschaften. Die ersten 4 Punkte sind die Arten, die später deine curve abgeschlossen werden soll - das blaue gibt an, auf was für eine Fläche später das Cap liegen soll. Wir wählen jetzt natürlich Punkt 3, da die Cap immer in Richtung Wand angrenzen soll - man soll sie ja nicht sehen. Das untere Kästchen lassen wir gerade so, wie es ist, "Result to func_group" ist dafür verantwortlich, dass die beiden Caps (oben und unten an der curve) sowie die curve selbst eine Gruppe bilden, d.h. wenn du später die curve verschiebst, verschiebst du die Caps automatisch mit :) Nun drückst du "OK", und es sieht so aus:



Achtung:

Wenn du willst, kannst du diese "Caps" auch wieder löschen, sie sind hier nicht unbedingt notwendig. Nun haben wir sie allerdings in eine Gruppe zusammengesetzt, also drückst du 2 x die rechte Maustaste und wählst im folgenden Menü den Punkt "ungroup entities" aus - jetzt hast du die Gruppe aufgelöst, du kannst die Caps und die curve einzeln anwählen.

Willst du die Gruppierung nicht auflösen, und nur einen der beiden "Caps" löschen, drückst du "SHIFT" + "ALT" + linke Maustaste und kannst nun über die "BACKSPACE"-Taste ein "Cap" löschen.

So, das hätten wir schon mal, eine schöne "runde Ecke" in unserem Raum. Das ganze machst du jetzt in den anderen Ecken auch, aber vergiss nicht, die Brushes zu mappen, die die "Caulk"-Textur tragen - die curve selbst zählt nämlich nicht als solider Brush und ohne diese Brushes hättest du ein Leak in der Map.

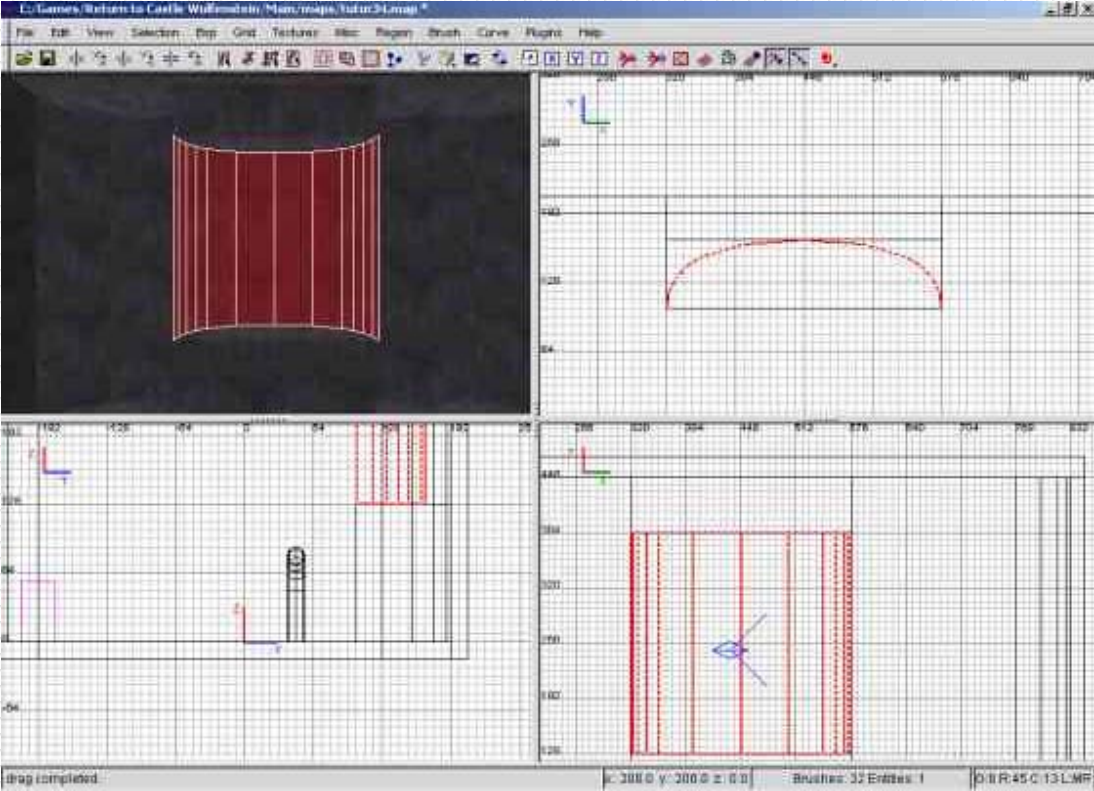
Jetzt versuchen wir mal, den Caps eine Textur zu verpassen, hier kommt es nämlich zu ein paar Besonderheiten. Zunächst sind die Caps und die curve ja noch gruppiert, also drückst du "SHIFT" + "ALT" + linke Maustaste und zeigst damit auf deinen Cap. Nun hast du einen Cap selektiert. Jetzt kannst du ganz normal eine Textur aussuchen. Nun kommt aber das neue, du musst nun die Textur über die Tasten "STRG" + "SHIFT" + "N" ausrichten.

Wenn du das ganze lieber im Surface Inspector erledigen willst, kannst du das gerne tun, du weißt ja, man kann ihn über die Taste "S"

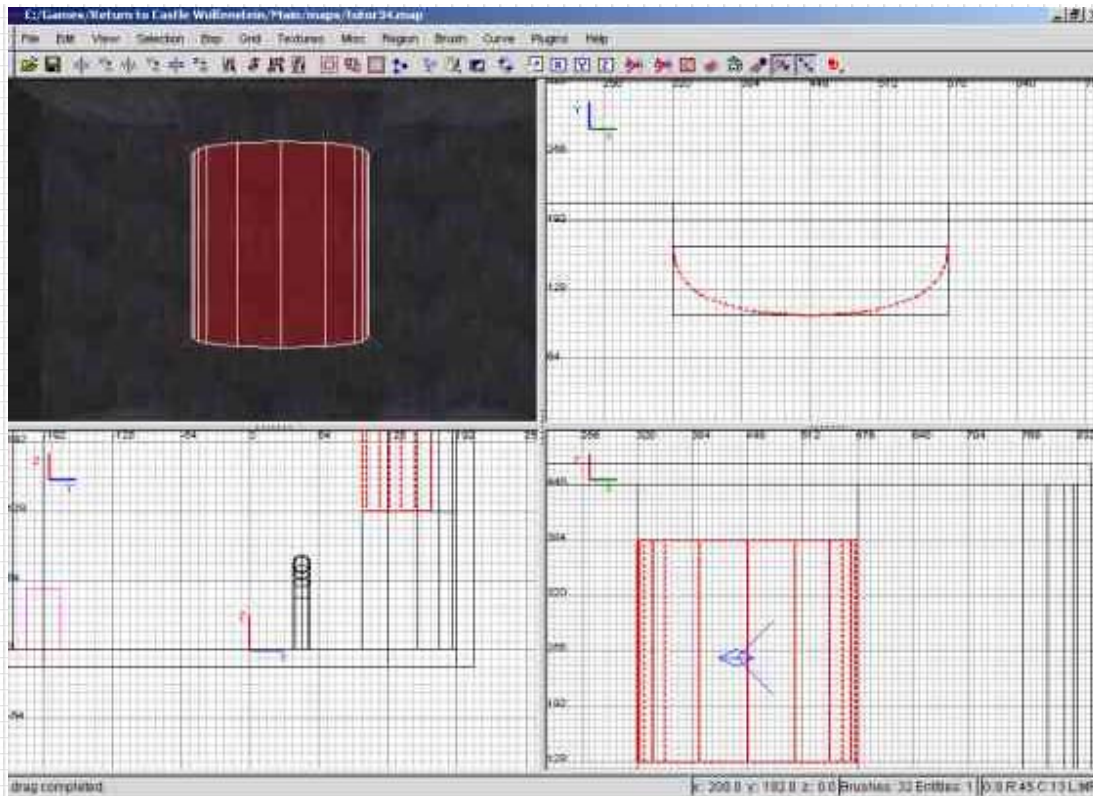
aufrufen. Hier benutzt du die Felder unterhalb von "Patch", also "CAP, SET, NATURAL, und FIT) bzw. zum verschieben der Textur die Felder Horizontal/Vertical shift und Horizontal/Vertical stretch. Wenn ihr hiermit keinen Erfolg habt, gibt es noch ein spezielles Patch-Properties Fenster, dazu drückst du "SHIFT" + "S" .Dieses Fenster ist ähnlich aufgebaut, wie der normale Surface Inspector, ist aber speziell für Patches gedacht. Also kannst du hier die Texturen verschieben ,stretchen usw.

ÜBGRIGENS: Ist dir beim Texturen belegen ein anderer Brush im Weg, kannst du diesen einfach markieren, und "H" drücken, um ihn auszublenden. Wenn du fertig bist, drückst du einfach "SHIFT" + "H" um den Brush wieder darzustellen.

Nun haben wir da noch in der Mitte der Map eine größere Fläche, die wir noch mit einer Curve verschönern wollen. Nun erzeugst du einen neuen Brush, und wählst im Menü den Punkt "Curve" und als Unterpunkt "End Cap":



Nun, wie du siehst, passt die Curve garnicht, wir wollten sie ja andersherum haben, aber das wirft uns nicht um, du drehst die Curve um 180° und wählst im Menü den Punkt "Curve", hier "Matrix" und hier "Invert":



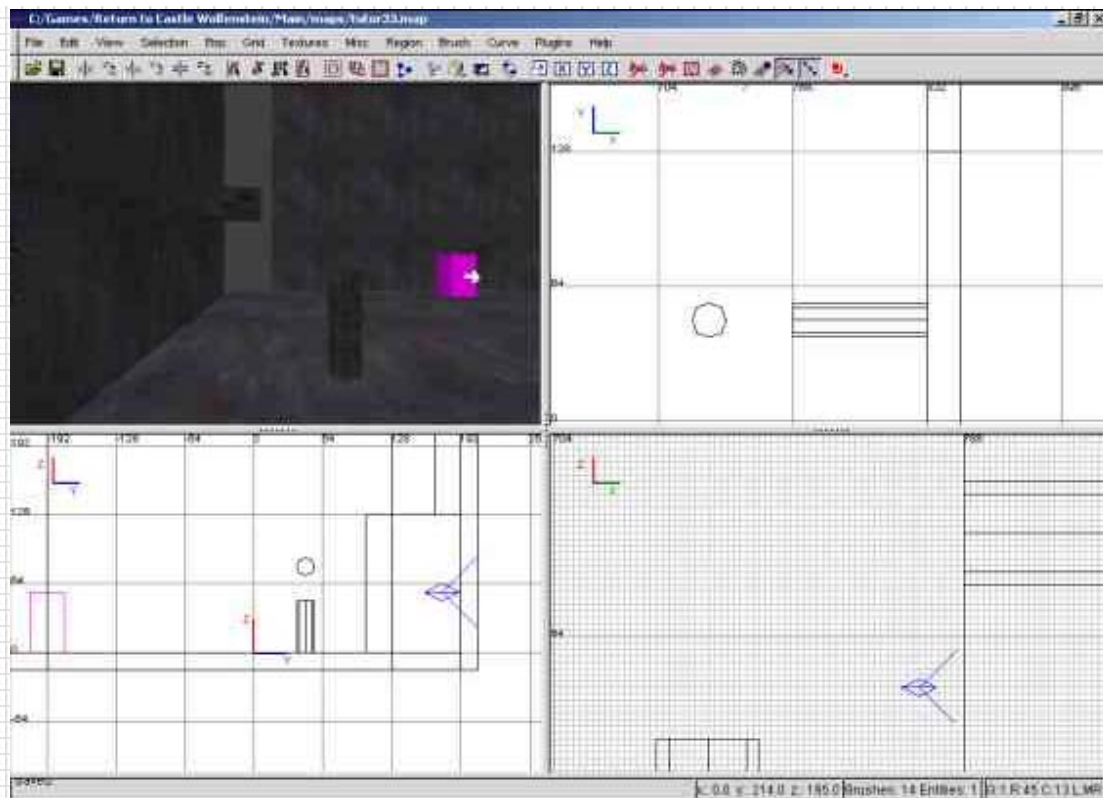
So, das hätten wir.

Jetzt machen wir uns mal einen Zylinder, hier bietet uns der Radiant 3 Möglichkeiten an, also klicke mal im Menü auf den Punkt "Curve", dort steht cylinder und untendran "more cylinders". Hier gibt es folgende Arten:

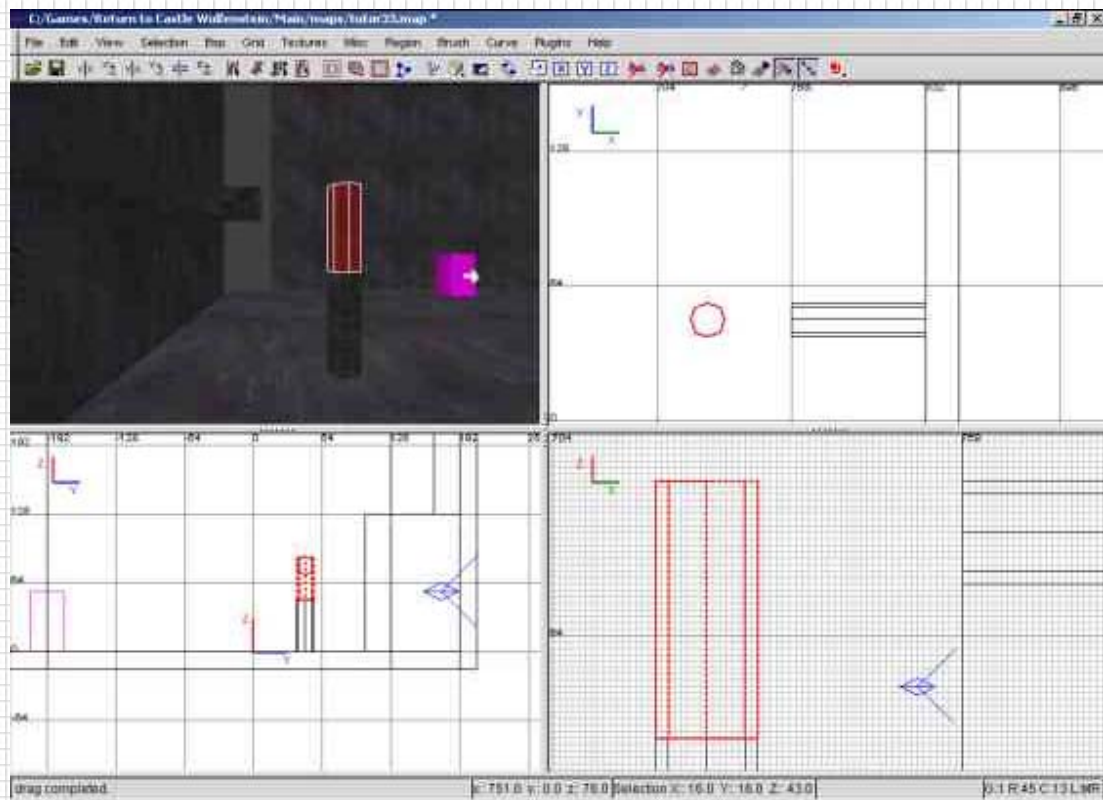
- Dense Cylinder = mehr Vertex Punkte (einfach mal die Taste V drücken)
- Very Dense Cylinder = noch mehr Vertex Punkte
- Square Cylinder = rechteckiger Zylinder (damit kannst du viereckige Zylinder erstellen und auch biegen)

Neben diesen natürlich noch den normalen Zylinder, der wenig Vertex-Punkte aufweist. Nun wollen wir mal mit diesem Zylinder arbeiten. Also erstellst du mal einen ganz normalen Brush, und wählst im Menü unter "Curve" den Punkt "Cylinder". Nun wird ein Zylinder erstellt, der außen mit der gewählten Textur belegt ist. Der Cylinder hat auch eine Innenfläche, die man "Matrix" nennt. Man kann diese Textur übers Menü ("Curve" -> "Matrix" -> "Invert") auch umdrehen. Nun wäre die Innenseite mit der Textur belegt, die Außenseite allerdings nicht. Wenn du einen Tunnel haben willst, bei dem außen UND innen eine Textur ist, musst du im Menü auf "Curve" -> "Thicken" gehen, aber dazu später mehr.

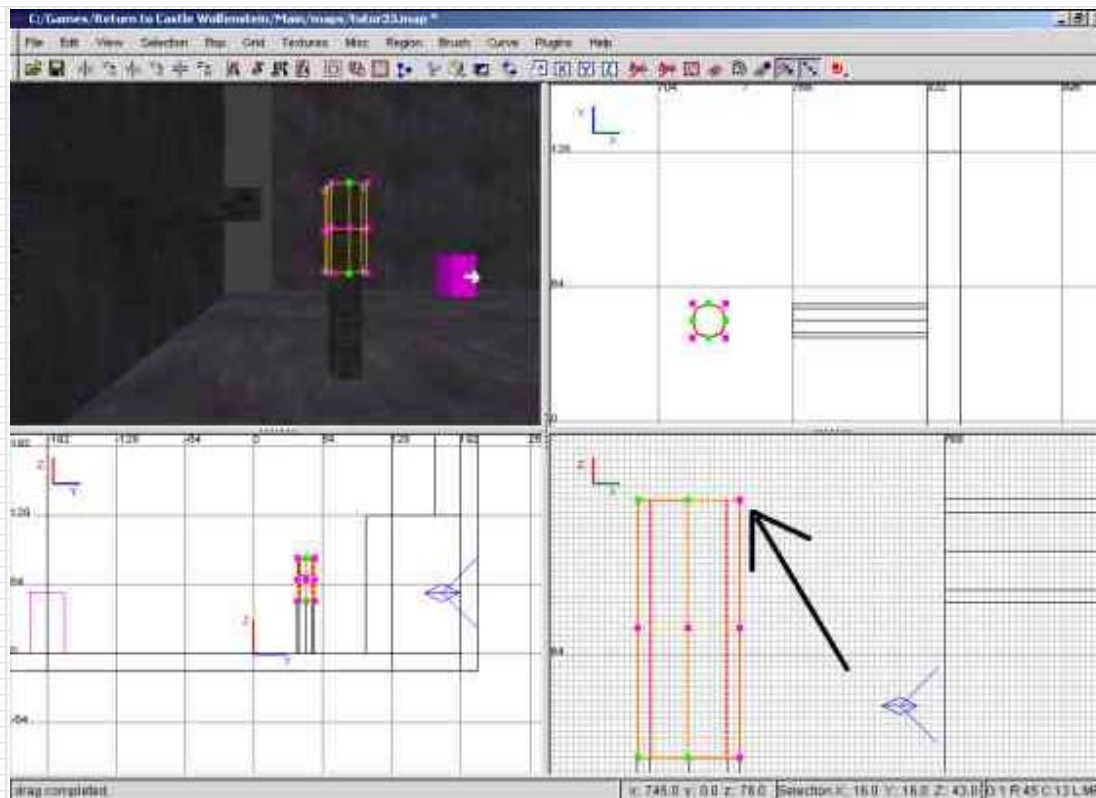
Also, nun bauen wir mal 2 Zylinder, und zwar wollen wir einen senkrechten und einen waagrechten Zylinder. Also erstellst du einen normalen Brush, der 16 Units breit ist und wählst im Menü "Curve" den Punkt "Cylinder". Nun machst du das ganze nochmal, aber diesen Zylinder wollen wir ja waagrecht haben, also benutzt du die Knöpfe x-axis Rotate, y-axis Rotate und z-axis Rotate. um ihn in die waagrechte zu legen. Nun sollte es ungefähr so aussehen:



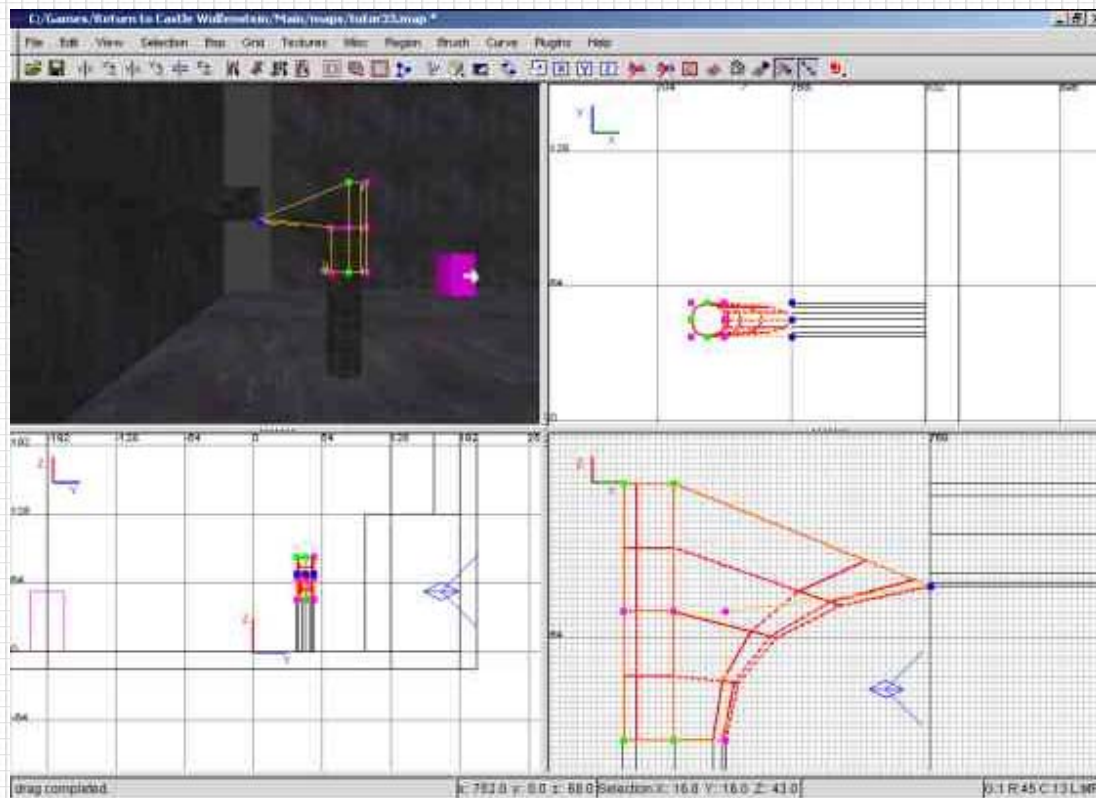
Ich bin jetzt in der Vorderansicht auf eine andere Grid-Größe gewechselt, damit wir den besseren Überblick haben. Tja, vielleicht hast du dir es schon gedacht, wir wollen jetzt einen Zylinder erstellen, den wir um 90° biegen und die beiden Zylinder verbindet. Also erstellen wir einen neuen Brush, und machen daraus über die Menüpunkte "Curve" -> "Cylinder" einen weiteren Zylinder. Diesen setzen wir jetzt auf den senkrechten Brush:



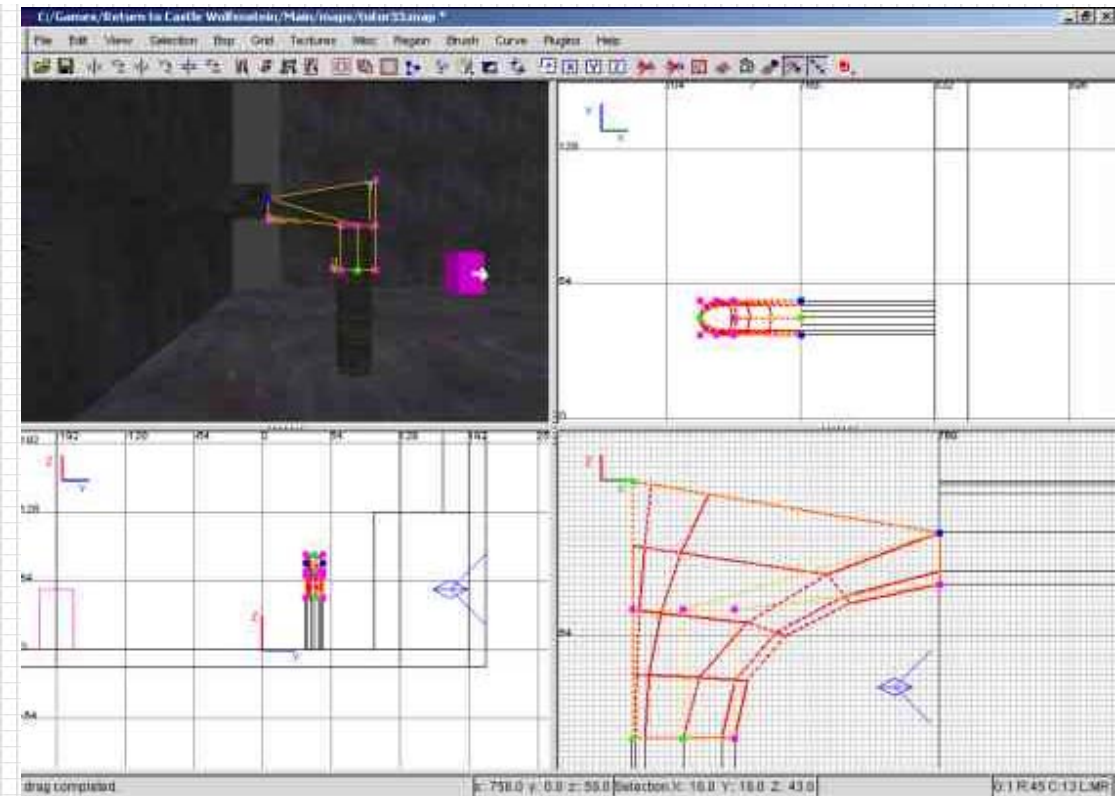
Soweit hätten wir schon mal den Grundstein gelegt, nun drücken wir die Taste "V", um in den "Vertex-Mode" zu kommen:



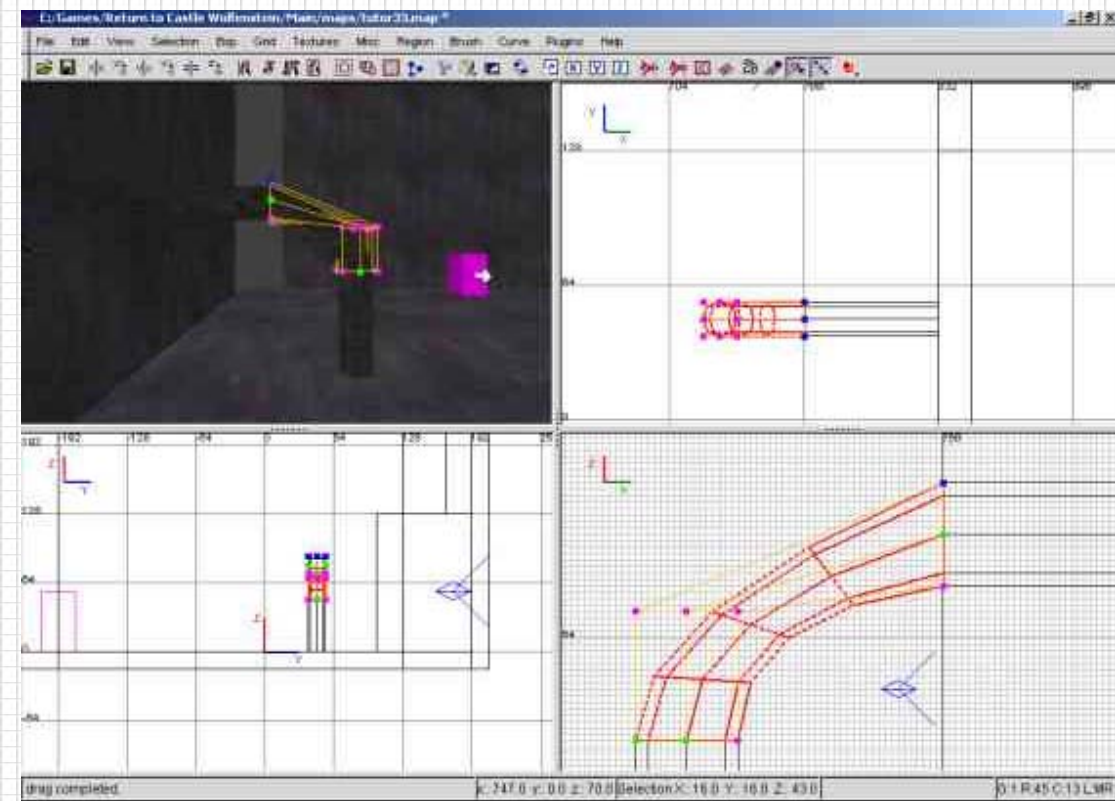
Nun klickst du diesen Punkt ganz oben rechts an, den ich dir mit dem Pfeil angedeutet habe. Diesen ziehst du nun an die Unterkante des waagrechten Zylinders:



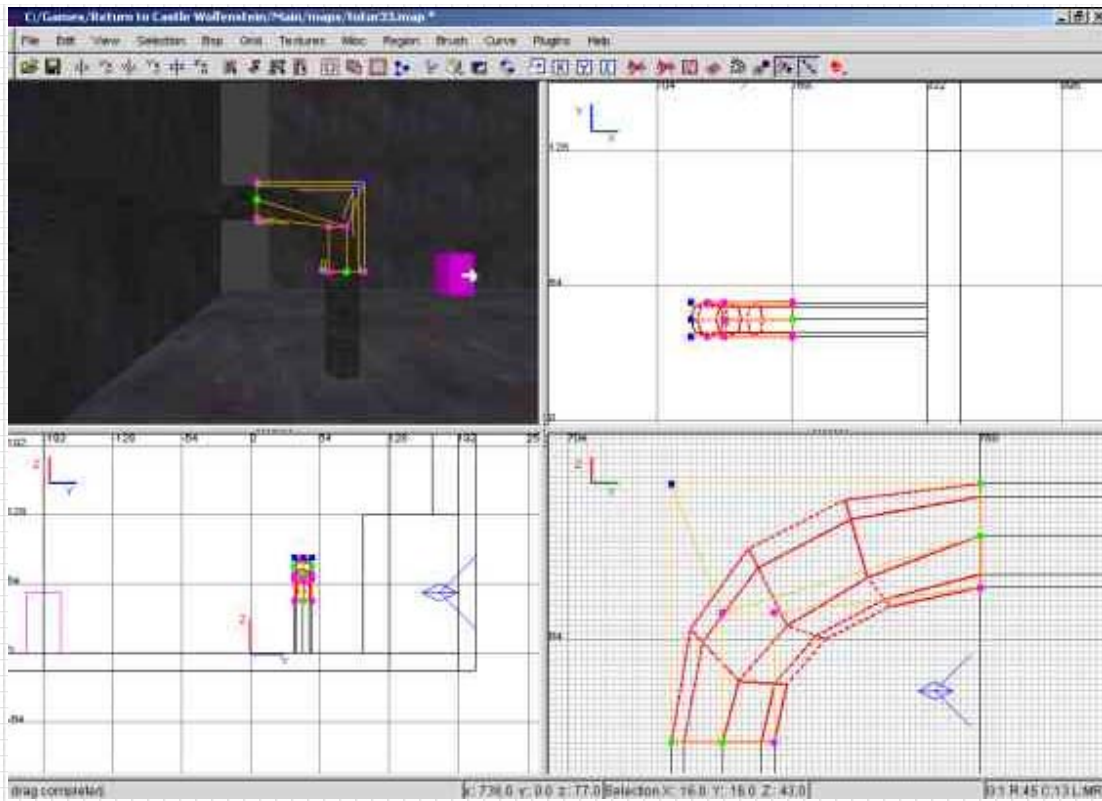
Nun klickst du den oberen mittleren grünen Punkt an, und verschiebst du an die mittlere Kante vom waagrechten Zylinder:



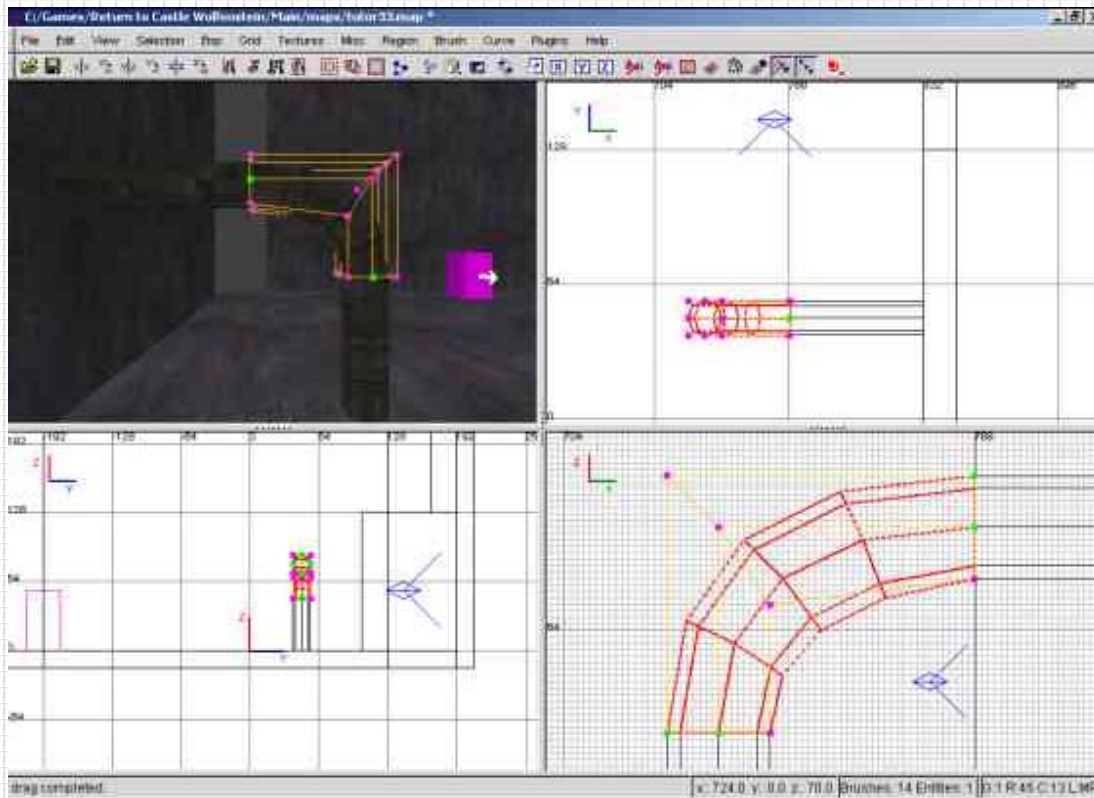
So, jetzt verschieben wir noch den oberen linken Punkt an die Oberkante vom waagrechten Zylinder:



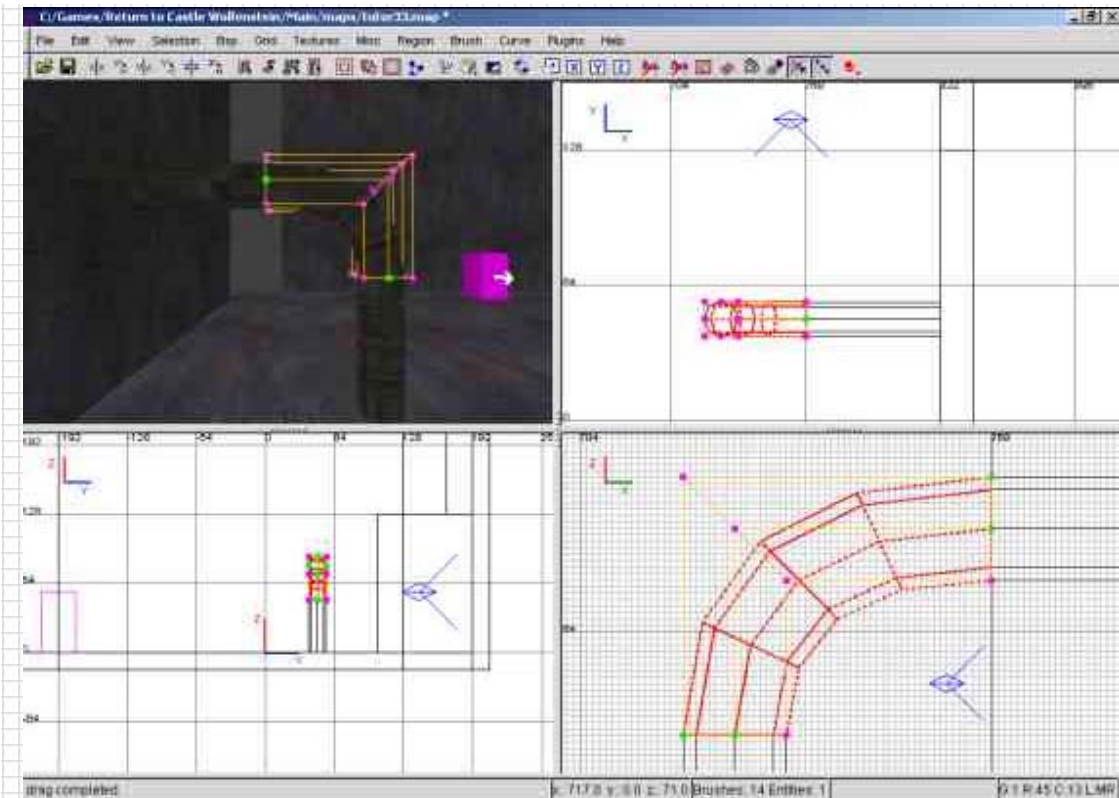
Nun sind wir mit dem "Anchließen" fertig, jetzt folgt eigentlich nur noch das Fein-Tuning. Dazu wählst du jetzt den linken mittleren Punkt aus, und schiebst ihn auf die Höhe der Oberkante des waagrechten Zylinders. Am besten schaust du dir das Bild dazu mal an:



So, nun nimmst du den mittleren Punkt in dieser Reihe und verschiebst ihn so, dass die gelbe Linie nach unten und nach rechts gerade sind:



Das machen wir jetzt noch mit dem rechten Punkt, der da so einsam und alleine sitzt:



So, das hätten wir geschafft - eine schöne um 90° gebogene Röhre. Natürlich kannst du die Röhre auch ohne die beiden anderen Rohre biegen, aber ich denke, für den Anfang ist es als Orientierungspunkt nicht schlecht. Natürlich musst du noch die Textur besser ausrichten, wie immer über den Surface Inspector (Per Taste "S") - hier kannst du mit den Knöpfen unter "Patch" die Textur besser ausrichten.

So, nun aber zum restlichen Menü "Curve":

Cylinder: Erstellt aus einem Brush einen Zylinder, siehe den 90° Zylinder

More Cylinders:

- Dense Cylinder
- Very Dense Cylinder
- Square Cylinde

End Cap: Erstellt eine halb-runde Curve, hier musst du daran denken, die Curve mit der "Caulk"-Textur nach hinten abzudichten.

Bevel: Hiermit kannst du eine Wand erstellen, die etwas "abgerundet" ist. Hier kannst du die Curve über die Vertrex-Punkte verändern. Auch hier musst du auf die "Caulk"-Abdichtung achten.

More End Caps, Bevels: Hier gibt es die selben Curves nochmal, aber sie sind mit zusätzlichen Seitenwänden ausgestattet.

Curve/Cone: Hiermit kannst du zulaufende Spitzen, aber auch Kugeln erstellen. Dazu wechselst du in den Vertex-Mode und flachst die Spitze ab - dann kopierst du die halbe Kugel und und setzt sie um 180° versetzt dagegen, fertig !

Curve/Simple Patch Mesh: Damit kannst du einen sehr flachen Patch erstellen, und ihn mit dem Vertex-Mode verschieben, wie du willst. Je mehr Zeilen und Spalten du vorher definierst, je mehr Vertex-Punkte hast du später. Aber vorsicht, Landschaften erstellst du damit besser nicht, und wenn, dann clipp diesen Patch komplett ab.

Curve/Insert & Delete: Hiermit kannst du im Nachhinein noch weitere Vertex-Punkte einfügen oder löschen.

Curve/Matrix/Invert: Mit dieser Funktion kannst du das Innere der Curve nach außen kehren, das hast du ja schon oben gemacht.

Curve/Matrix/Re-disperse: Wenn du mehrere Vertex-Punkte manuell einfügst, sind diese nicht gleichmäßig angeordnet, mit dieser Funktion kannst du die Punkte wieder gleichmäßig ausrichten.unkte nachträglich gleichmäßig ausrichten.

Curve/Matrix/Transpose: da hab ich leider auch keine Ahnung :(

Curve/Cap: Hiermit kannst du der Curve eine Art Deckel verpassen, und somit die Curve abschliessen. Es werden immer an beiden Seiten Caps hinzugefügt.

Curve/Cap/Inverted Bevel bzw. Inverted Endcap: Hiermit kannst du Deckel für deine Bögen und Kurven erstellen, diese werden invertiert (umgedreht) dargestellt.

Curve/Cap/Cycle Cap Texture: Hiermit kannst du die Textur-Ausrichtung bestimmen. Wählst du diesen Punkt mehrmals an, kannst du durch die verschiedenen Ausrichtungen schalten.

Curve/Overlay/Set: Mit dieser Funktion kannst du die Vertex-Punkte dauerhaft einblenden und mit Clear wieder löschen.

Curve/Thicken: Hiermit kannst du einem Zylinder eine Innenseite verpassen (wenn er nur eine Außenseite hat) und umgekehrt. Es erscheint ein kleines Fenster, in dem du mit "Amount" die Wandstärke angeben kannst - und mit "SEAMS", ob die Deckel auch noch mit erzeugt werden sollen.

[zurück zur Hauptseite](#)





Die Hint-Textur:

verwendete Beispielpmap: "tutor37.map"

Ergebnismap: "tutor38.map"

Die Hint-Textur ist eine Textur, die die Performance deiner Map sehr stark erhöhen kann. Leider kann man diese Textur nicht überall einsetzen, daher sollte deine Map einige "Grundvoraussetzungen" bieten:

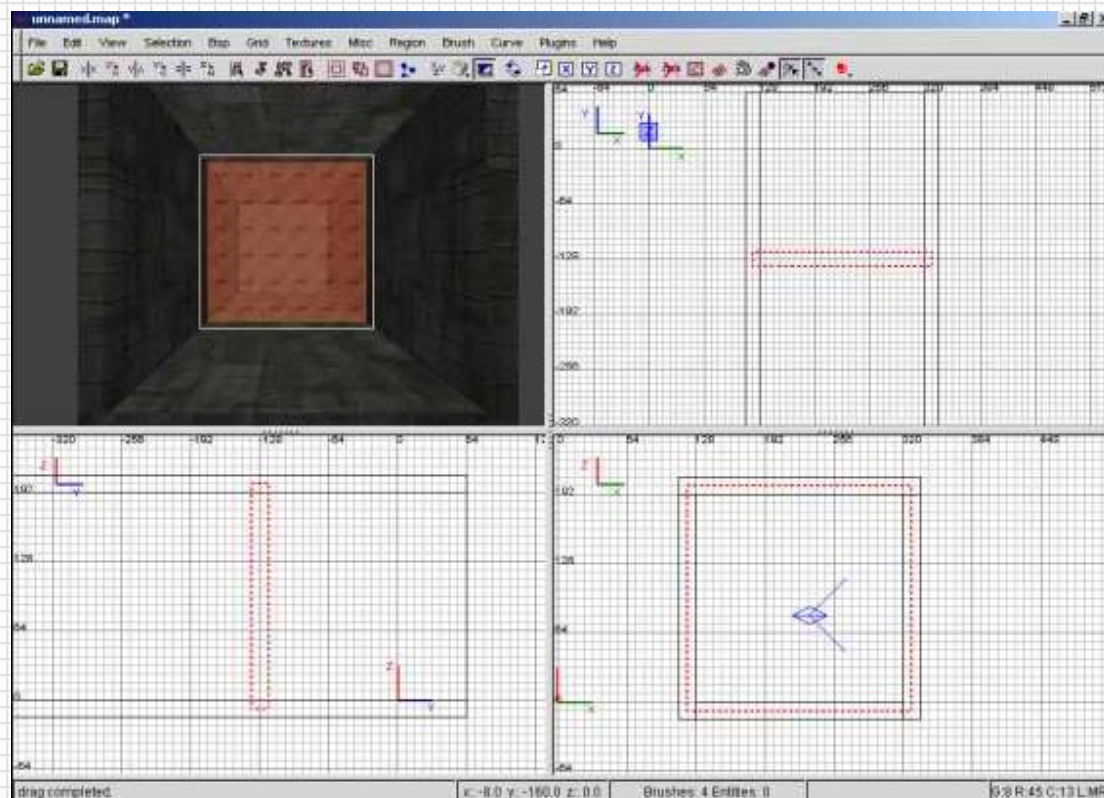
- sie sollte verwinkelt sein
- viele Räume, die in sich geschlossen sind

Maps, wie z.B. mp_beach, QDM17 usw. eignen sich weniger für den Einsatz der Hint-Textur, da hier die Flächen zu gross sind und es keine abgeschlossenen Räume gibt.

Nun will ich dir zunächst erklären, wieso man eine Hint-Textur nur in verwinkelten Maps einsetzen kann:

Mit der Hint-Textur wird dein Level in kleine Bereiche eingeteilt. Steht der Spieler in einem solchen Bereich, wird nur dieser eine Bereich berechnet, die anderen hingegen nicht. Nun wollen wir aber gleich eine solche Hint-Textur in ein Level einbauen. Dazu wählst du zunächst die Hint-Textur aus, diese findest du unter "Common/hint". Nun erstellst du einen Brush, der alle umliegenden Brushes um mindestens 8 Units überschneidet:

Achtung: Wenn du ein Hint-Portal bauen willst, und dieses verschwindet plötzlich im Editor, so ist das kein Fehler. Dazu öffnest du im Menü den Menüpunkt "View" -> "Show" -> "Show Hint" und nun aktivierst du diese Funktion. Damit blendest du die Hint-Portale ein. So kannst du z.B. auch alle Entities verschwinden lassen - wenn deine Map zu unübersichtlich geworden ist.



Wie du hier gut sehen kannst, überschneidet der Hint-Brush alle umliegenden Brushes um 8 Units. Das ist deshalb notwendig, dass die Engine den Hint-Brush auch wirklich anerkennt. Würde eine Seite nicht in einen anderen Brush schneiden, so würde auch der Hint-Brush nicht aktiv - und die Performance verbessert sich nicht. Dabei solltest du beachten, dass die umliegenden Brushes nicht aus Detail-Brushes bestehen dürfen, sonst funktioniert das Hint-Portal auch nicht.

Übrigens: In dem oberen Bild kannst du sehen, wie man ein solches Hint-Portal setzt. In einem solchen geraden Gang kann man natürlich normal kein Hint-Portal setzen, da man ja trotzdem hindurch sehen kann. Also solltest du Hint-Brushes am besten an Ecken usw. setzen

Nun solltest du dir mal die beiden Beispielmeps ansehen. Diesmal habe ich dir 2 Beispielmeps dazugepackt, eine Map (tutor37.map) hat keinerlei Hint-Portale eingebaut, die andere Map (tutor38.map" hingegen ist mit Hint-Portalen ausgestattet. Nun musst du erst einmal die beiden Maps kompilieren, hier solltest du mit "fullvis" kompilieren, da die Hint-Portale sonst nicht aktiv sind.

Nun startet ihr RtcW und gebt in die Console ein:

```
/sv_pure 0
/devmap tutor37
/r_showtris 1
```

Nun kannst du sehen, dass die komplette Map berechnet wird, obwohl die meisten Räume eigentlich garnicht sichtbar sind, d.h. sie werden unnötigerweise berechnet. Und genau das ändert sich beim Einsatz von Hint-Portalen.

Nun öffnest du wieder die Console und gibst ein:

```
/sv_pure 0
/devmap tutor38
```

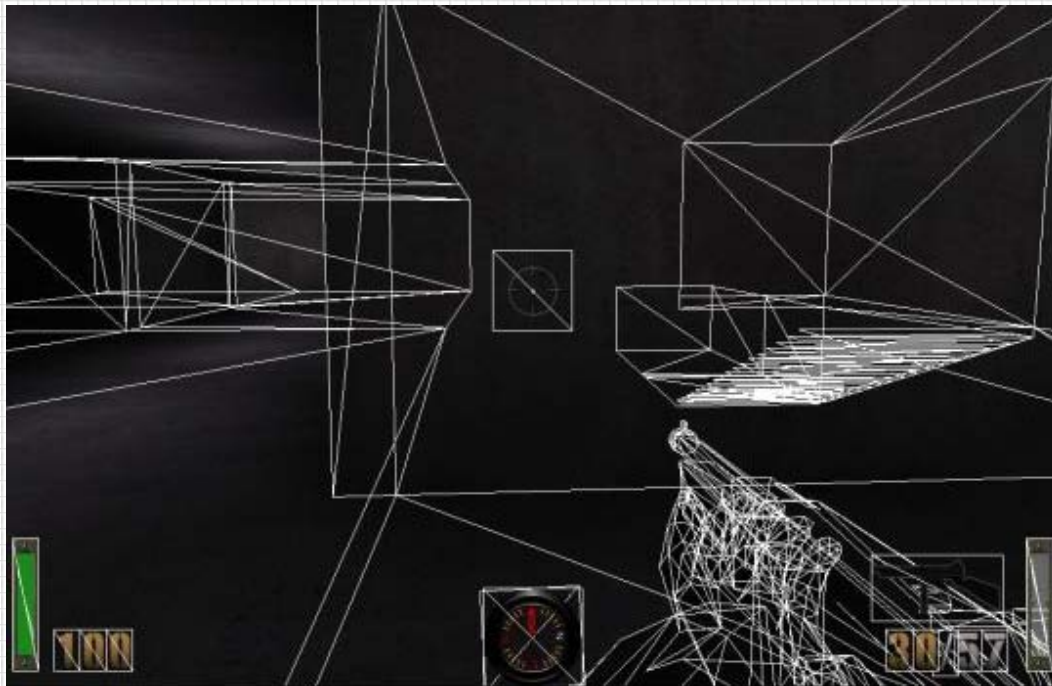
Den Befehl "r_showtris 1" solltest du nichtmehr eingeben müssen, er sollte noch aktiv sein. Ist er dennoch nicht aktiviert, gibst du den Befehl natürlich wieder ein, da wir uns ja den Effekt der Hint-Portale ansehen wollen.

Nun siehst du die Map und die Bereiche, die gerade berechnet werden. Nun kannst du mal durch die kleine Map laufen - du wirst feststellen, dass einige Bereiche kurz vor dem Betreten aufpoppen und sich beim verlassen wieder schliessen. So kannst du auch in den einzelnen Bereichen mehr Details einbauen - ohne dass die Performance leidet.

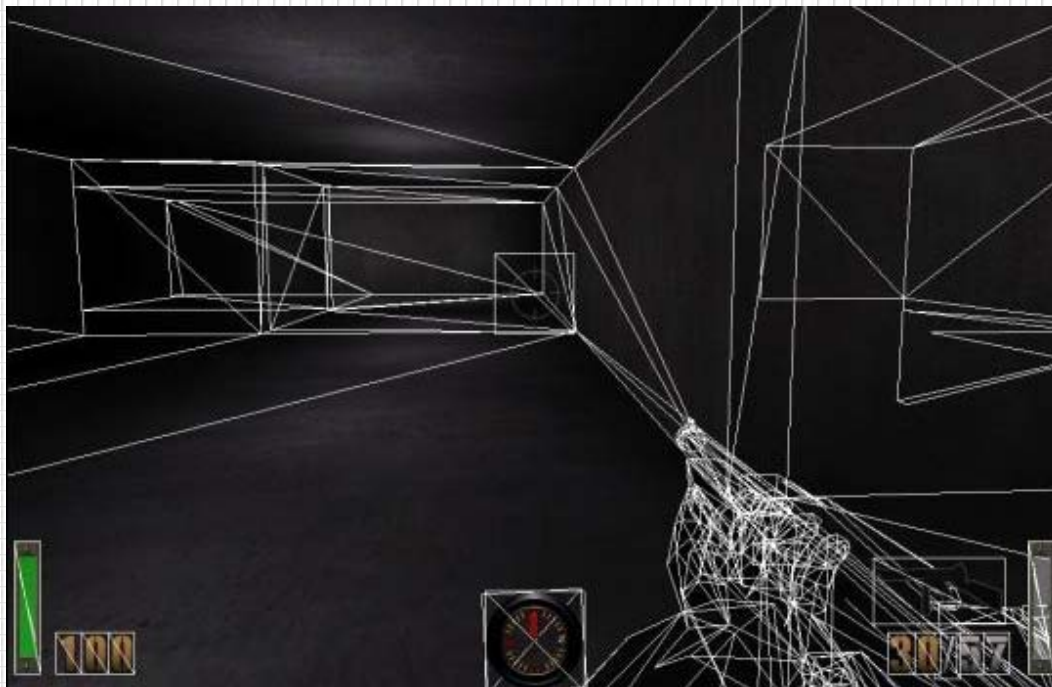
Nun zeige ich dir die Map, so dass du siehst, wo die Hint-Portale in dieser Map gesetzt worden sind:



Nun könnte es bei dir so aussehen:



Die Treppe wird noch mitberechnet, nun gehen wir aber ein paar Schritte nach links...



und die Treppe verschwindet langsam, d.h. sie wird nichtmehr berechnet und so haben wir schon ein wenig Performance herausgeholt. Natürlich kannst du den Nutzen in dieser kleinen Map nicht so gut sehen - aber wenn du viele verschachtelte Räume hast, macht sich dieser Effekt dann gut bemerkbar.

Hint-Portale machen also nur Sinn, wenn die Sichtlinie unterbrochen wird. Kannst du durch 4 Hint-Portale sehen, werden sie natürlich nicht aktiv. Also solltest du mit den Hint-Portalen etwas experimentieren. Jedoch solltest du deine Map im Vorraus so planen, dass du Hint-Portale einsetzen kannst.

Denn, eine fertige Map mit niedriger Performance so umzubauen, dass du Hint-Portale einsetzen kannst, ist eine sehr aufwendige Arbeit, die man aber durch gute Planung leicht umgehen kann.

ACHTUNG: Hast du z.B. ein Feuer und einen Sound in einem Teil der Map, so hakt der Sound etwas, wenn man durch das Hint-Portal läuft. Das Problem kannst du leicht lösen, indem du beim "target_speaker", der den Sound auslöst, das Entity-Fenster öffnest und einen Haken bei "no_pvs" setzt. Nun sollte der Sound nichtmehr so abgehakt werden.

[zurück zur Hauptseite](#)

183759



Die Area-Portal-Textur:

Beispielmap: "tutor43.map"

Ergebnismap: "tutor44.map"

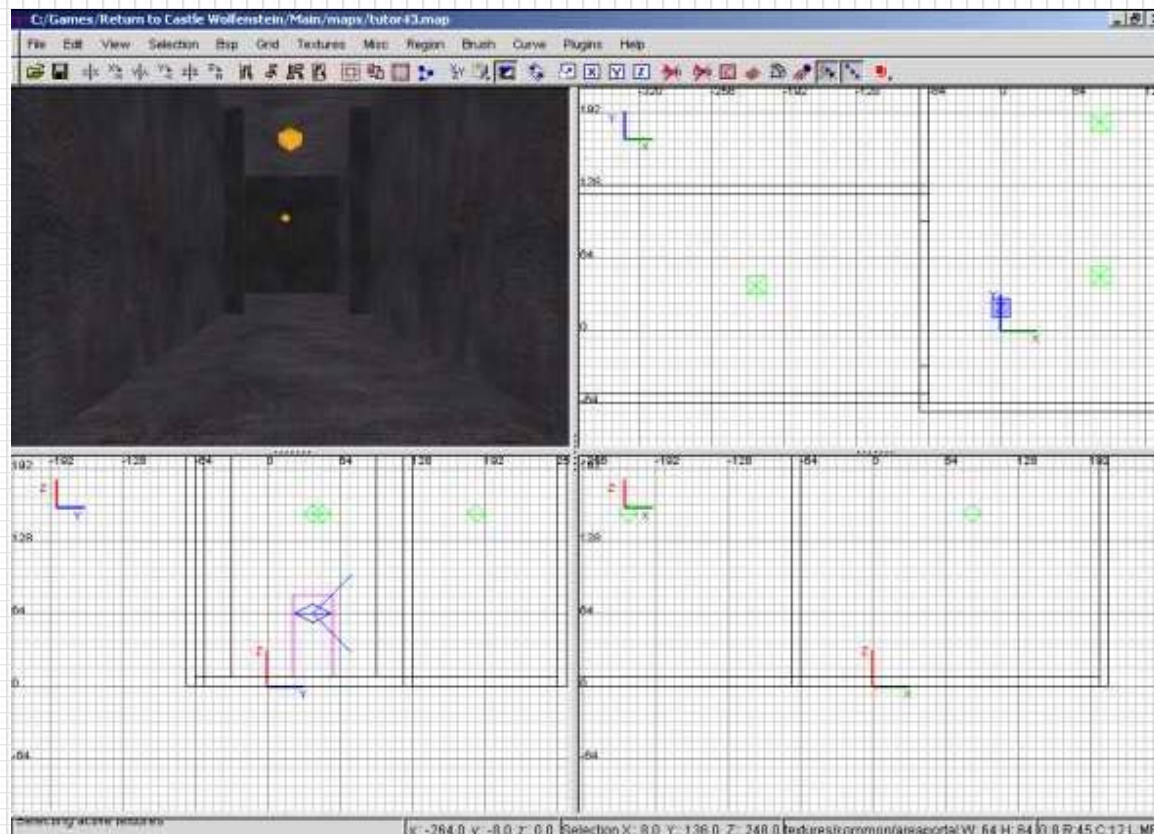
Areaportale sind in der Funktion dem Hint-Portal sehr ähnlich. Area-Portale sind genau wie Hint-Portale dazu da, die FPS der Map zu erhöhen. Um ein Area-Portal in deine Map einzubauen, muss die Map wiederum einige Voraussetzungen mit sich bringen:

- muss in den Türrahmen passen
- die Türe sollte aus wenigen Brushes bestehen
- die Türe sollte nicht durchsichtig sein
- der Raum, der das Area-Portal abschliesst, muss dicht sein, d.h. es darf keine Leaks geben.

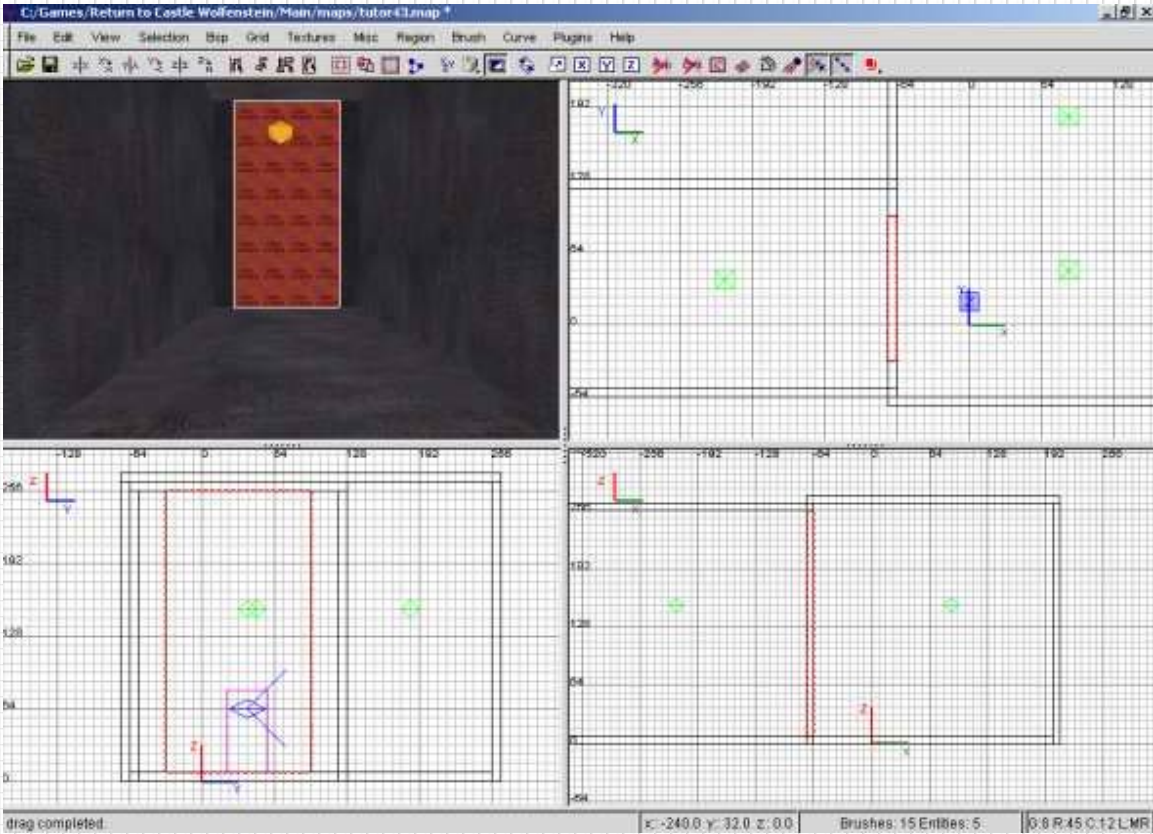
Der Sinn bei einem Area-Portal ist der: Die Türe zum Raum ist noch zu, also soll auch der Raum dahinter nicht berechnet werden, da man ihn ja auch nicht sehen kann. Dies wird durch das Area-Portal bewerkstelligt.

Nun wollen wir aber mal sehen, wo wir soetwas am geschicktesten verwenden. Dazu habe ich dir eine kleine Testmap gebaut, in der du sehen kannst, wie man ein Area-Portal setzt. Nun öffnest du die Datei "tutor43.map". Nun siehst du einen kleinen Raum, der über eine Türe mit einem grösseren Raum verbunden ist:

Nun markierst du die Türe (mit "SHIFT" + "linke Maustaste" die Türe anklicken) und drückst "H" um die Türe zu verstecken:



Nun ziehst du in dem Türrahmen einen Brush, der 8 Units breit ist, und genau passt (also nicht in die angrenzenden Brushes hineinragt). Diesen belegst du nun mit der Area-Portal-Textur (findest du unter "common/areaportal"):



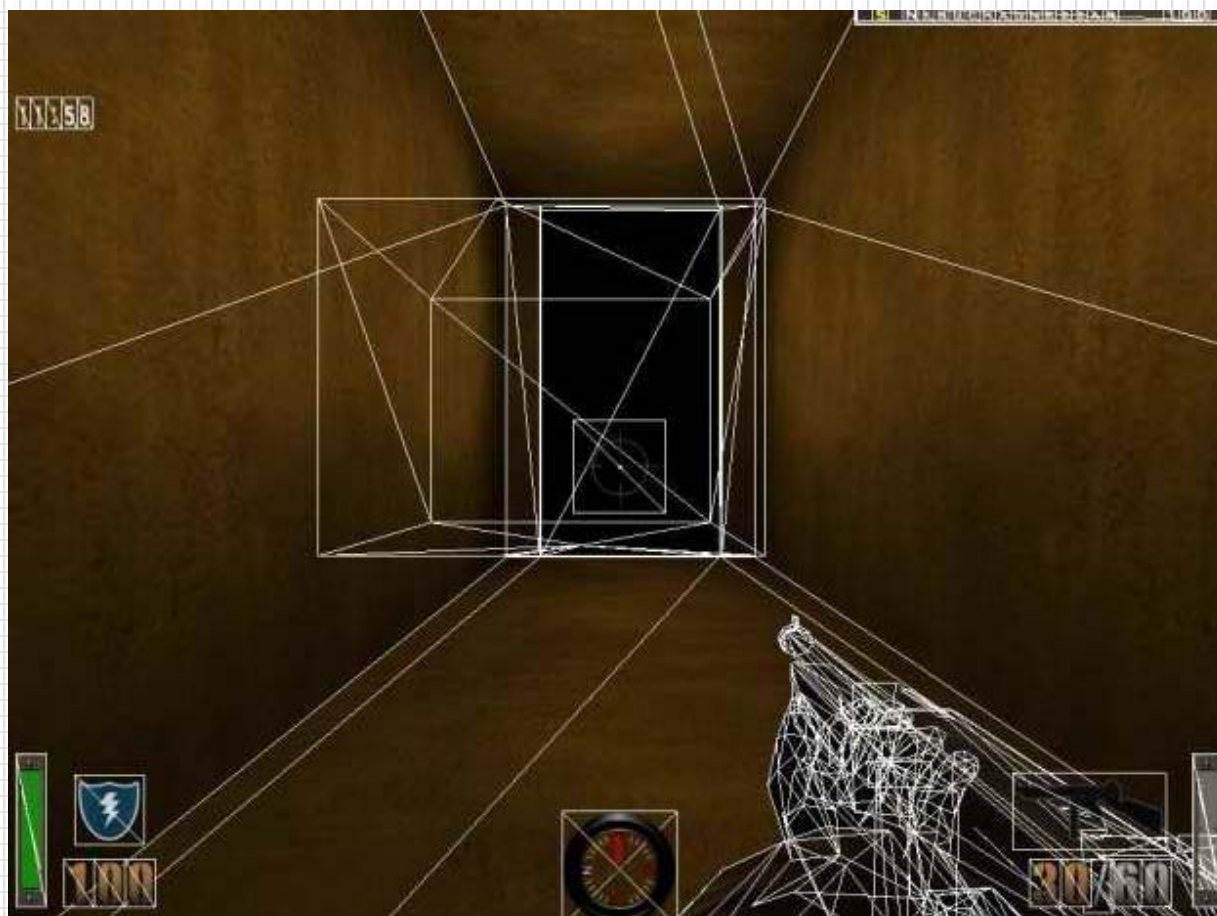
Nun kannst du die "ESC"-Taste drücken um den Brush zu deselektieren. Danach kannst du auch wieder "STRG" + "H" drücken, um die Türe wieder einzublenden. Nun kannst du die Map gleichmal compilieren. Dann öffnest du die Map wieder über die Befehle

```
/sv_pure 0
/devmap tutor43
/r_showtris 1
```

Nun sieht es bei dir in etwa so aus. Der Raum hinter der Türe wird noch nicht berechnet, unser Area-Portal ist also aktiv.



Nun öffnest du mal die Türe und schon wird der Raum berechnet:



Leider muss ich dich hier warnen, das Area-Portal funktioniert nur ein einziges Mal, also solltest du dich nicht zu sehr auf die Area-Portale verlassen, wenn du um bessere FPS kämpfst. Du solltest zusätzlich noch Hint-Portale verwenden, die ständig aktiv sind.

[zurück zur Hauptseite](#)

183759



Die Clip-Texturen:

Verwendete Beispielmap: "tutor39.map"
Ergebnis-Map: "tutor40.map"

Nun gut - jetzt wollen wir uns mal mit ein paar speziellen Common-Texturen beschäftigen, nämlich den Clip-Texturen. Diese Clip-Texturen findest du unter "textures/common" und nennen sich clipXXX (XXX steht z.B. für Missile, Metal usw.) Einige davon haben ein sehr spezielles Einsatzgebiet.

Doch für alle gilt eigentlich soweit das gleiche:
Sie sind zwar im Radiant als Textur auf einem Brush sichtbar, jedoch sind sie im Spiel selbst unsichtbar. Sie sind dazu da, Spieler, Waffen, usw. zu blocken, d.h. sie nicht durch zu lassen. Es gibt folgende Clip-Texturen:

Clip:	die normale clip-textur blockt den Spieler. Dennoch lässt sie aber Waffen, Granaten, Racketen sowie Airstrikes hindurch.
Clip_metal:	ClipMetal blockt den Spieler, Granaten sowie Racketen. Sie lässt aber Waffen und Airstikes durch. Legt man diese Clip-Textur auf einen dünnen Brush auf den Boden, so ertönt beim drüberlaufen im Spiel ein Metal-Sound.
Clipfull:	Clipfull blockt nur den Spieler. Sie lässt WAffen, Granaten, Racketen sowie Airstrikes durch.
Clipmissile:	Clipmissile blockt Granaten und Racketen. Aber Waffen, Airstrikes und der Flammenwerfer geht durch.
Clipmonster:	Blockt nur Monster (also im Singleplayer) und lässt sonst alles durch.
Clipshot:	Blockt Schusswaffen, lässt Racketen, den Spieler, Granaten, den Flammenwerfer und Airstrikes durch.
Clipweapon:	Blockt den Spieler, Waffen, Granaten, Racketen, den Flammenwerfer und den Airstrike.

Mit den Clip-Texturen könnt ihr eure Map zum Schluss noch den letzten Schliff verpassen. Nun fragst du dich sicher, was man wohl mit dieser Textur anstellen kann. Zuerst kannst du damit alle Stellen abclipen, an die der Spieler nicht kommen soll. Z.B. kannst du damit alle Stellen abclipen, die der Spieler nicht erreichen soll, also z.B. ein Model. Aber auch an Stellen, wo der Spieler durch eine Himmel-Textur in andere Teile der Map sehen kann, sollte man so absperren.

Ein anderes Einsatzgebiet findet sich an versetzten Wänden oder anderen Stellen, an denen der Spieler hängen bleiben kann. Das sollte man verhindern, da es ärgerlich ist, wenn man an einer Stelle hängen bleibt und deshalb eine gute Zielscheibe für den Gegner bietet.

Ebenso könnt ihr komplexe Gebilde, z.B. Curves mit der Clip-Textur komplett einpacken, dadurch geht der Compilierungs-Prozess schneller, da z.B. eine waagrechte Stelle in den Curves nicht als begehbare Fläche berechnet wird. Ausserdem ist dies nützlich, wenn die Bots mit dem Addon herauskommen (für RtCW). Dann wird auch die Bot-Berechnung besser mit der Map zurecht kommen.

In der Beispiel-Map habe ich dir 3 Stellen mit der Clip-Textur abgeclippt. Natürlich wird es in einer normalen Map wesentlich mehr Möglichkeiten geben, die Clip-Textur einzusetzen.

Achtung:

Willst du ein Clip-Brush erstellen und er verschwindet plötzlich im Radiant? Das lässt sich schnell ändern. Klick im Menü einfach auf "View" -> "Filter" -> Clips (oder über die Tastenkombination "ALT" + "7" ("ALT" gedrückt halten und dann die Taste "7" drücken). Hier kannst du auch Teile deiner Map wegfiltern, z.B. Liquids, Caulks usw. und hast somit einen besseren Überblick.

Ich will dich noch auf ein anderes Anwendungsgebiet für die Clips aufmerksam machen. Hier scheiden sich zwar die Geister, aber ich glaube, jeder hat da seine eigene Meinung. So kannst du z.B. Fackeln und andere kleine Models einpacken um mehr Realismus aufkommen zu lassen, da man sonst durchlaufen kann. Jedoch solltest du überlegen, ob der Spieler lieber flüssig durch die Map kommen soll, oder eben den Realismus vorzieht. In manchen Maps fällt dies allerdings kaum ins Gewicht und die wenigsten Spieler achten darauf.

Natürlich gilt dies nicht für die großen Models, z.B. Flugzeuge usw.. Diese sollte man auf jeden Fall abclippen.

Am besten ist es, wenn du im Spectator-Modus durch deine Map fliegst und dir nochmal alle Stellen anschaust, die vielleicht noch abgeclippt werden sollten, da man an jeder Stelle hängen bleibt, selbst wenn die Brushes nur um 1 Unite zueinander verschoben sind.

[zurück zur Hauptseite](#)

183759

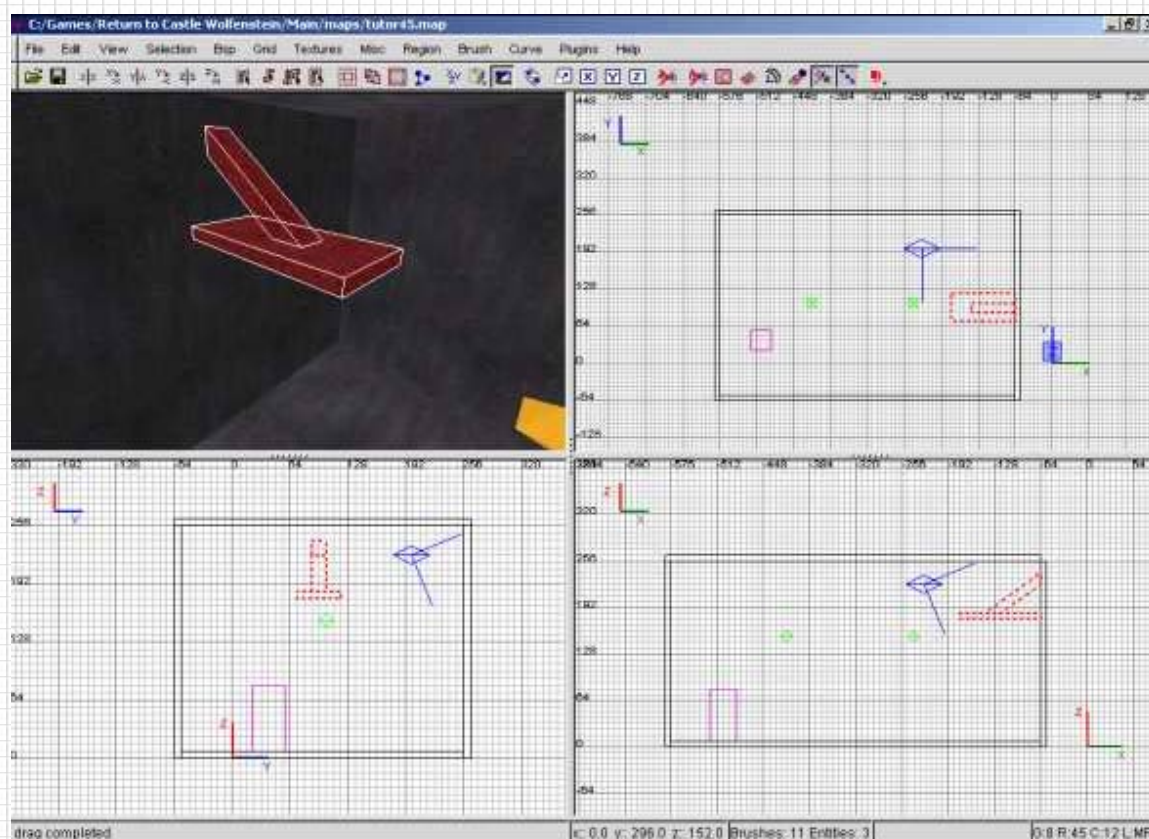


Ein Pendel erstellen:

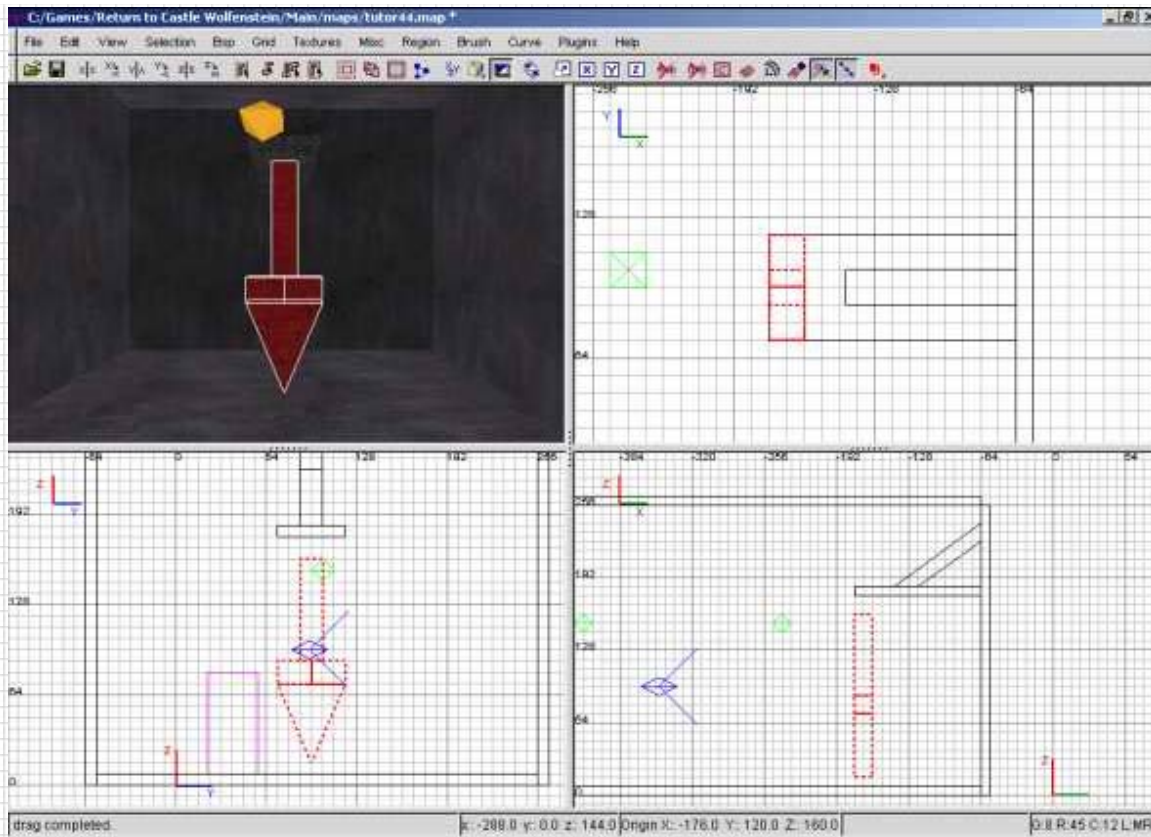
Beispielmap: "tutor45.map"

Ergebnismap: "tutor46.map"

Hier will ich dir zeigen, wie man ein Pendel erstellt. Ich habe dir eine Beispielmap erstellt, in der ich schon eine Art Aufhängung für unseren Pendel erstellt habe. Wenn dir dieser nicht gefällt, kannst du dir natürlich eine andere Aufhängung bauen, da es auf die Aufhängung selbst garnicht ankommt:

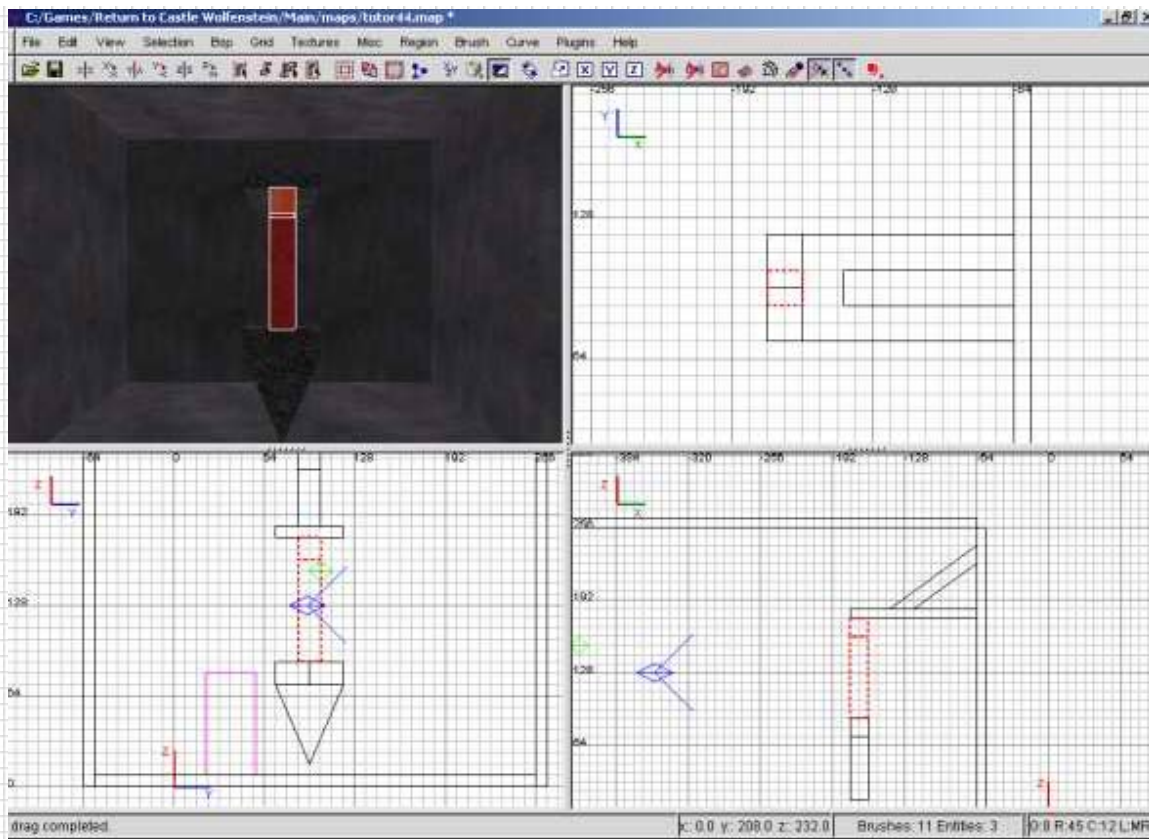


Nun deselektierst du die Aufhängung über die "ESC"-Taste und schon kann es losgehen. Nun brauchst du zunächst eine Konstruktion aus Brushes, die später dein Pendel darstellt. Ich habe diese Konstruktion hier erstellt:



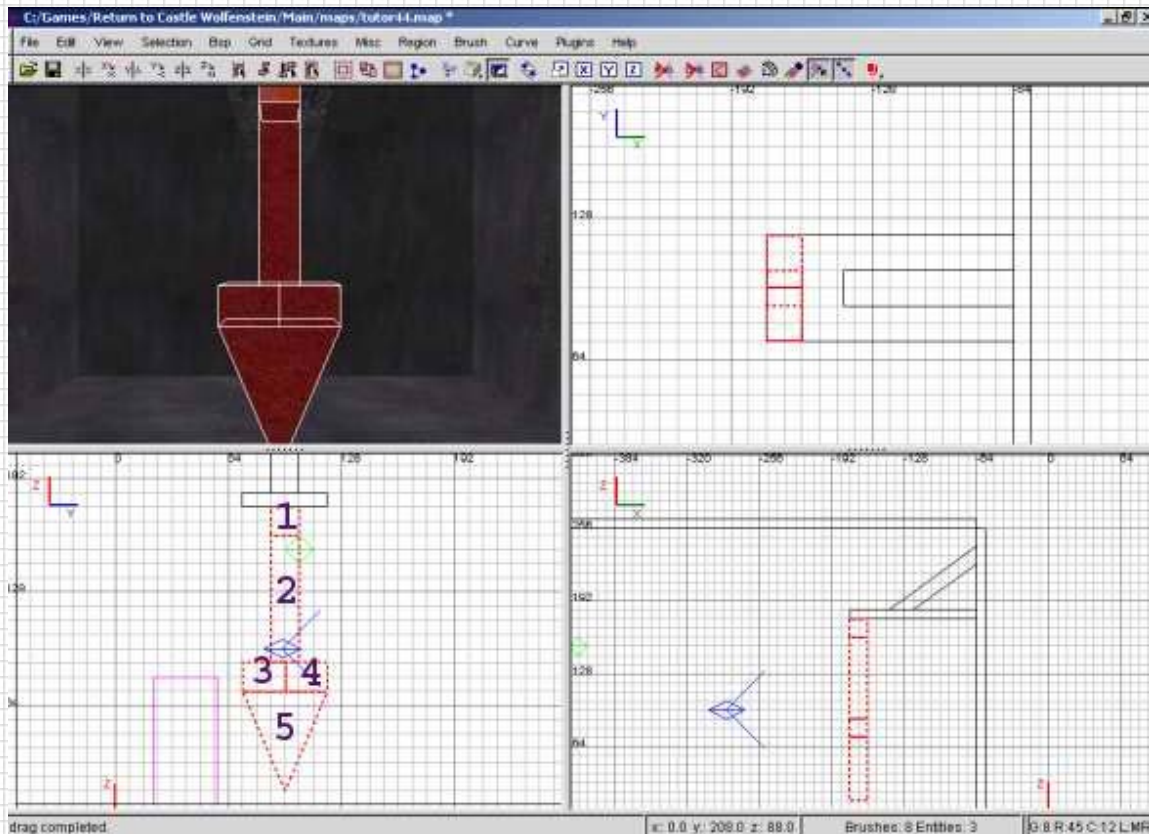
So, nun kommen wir gleichmal zu ein paar wichtigen Dingen, die man beim Erstellen des Pendels beachten sollte, da man so später viel weniger Aufwand hat. Wie du in der 3D-Ansicht sehen kannst, kommt (von oben nach unten gesehen) erst ein schmaler Schaft, dann gehen 2 Brushes nach aussen und es folgt zum Schluss die Pendelspitze. Mit dieser Konstruktion haben wir uns schon einige Einstellungen gespart. Da die Engine "merkt", dass das Pendel rechts und links breiter wird, wird es auch nach rechts und links schwingen. Würdest du diese "Verbreiterung" nach vorn und hinten gebaut, würde das Pendel später nach vorn und hinten ausschlagen.

Nun brauchen wir noch gewissermaßen den Drehpunkt, um den sich dann das Pendel drehen wird. Nun drückst du die "ESC"-Taste um das Pendel zu deselektieren. Nun brauchen wir einen Brush, der genauso gross ist, wie der Schaft des Pendels (also des oberen Brushes). Dazu aber gleich ein Bild:



Hier kannst du jetzt sehen, dass der neue Brush viereckig ist und die gleiche Seitenlänge hat, wie der Schaft des Pendels. Diesen neuen Brush belegst du jetzt mit der Origin-Textur. Diese findest du unter "common/origin".

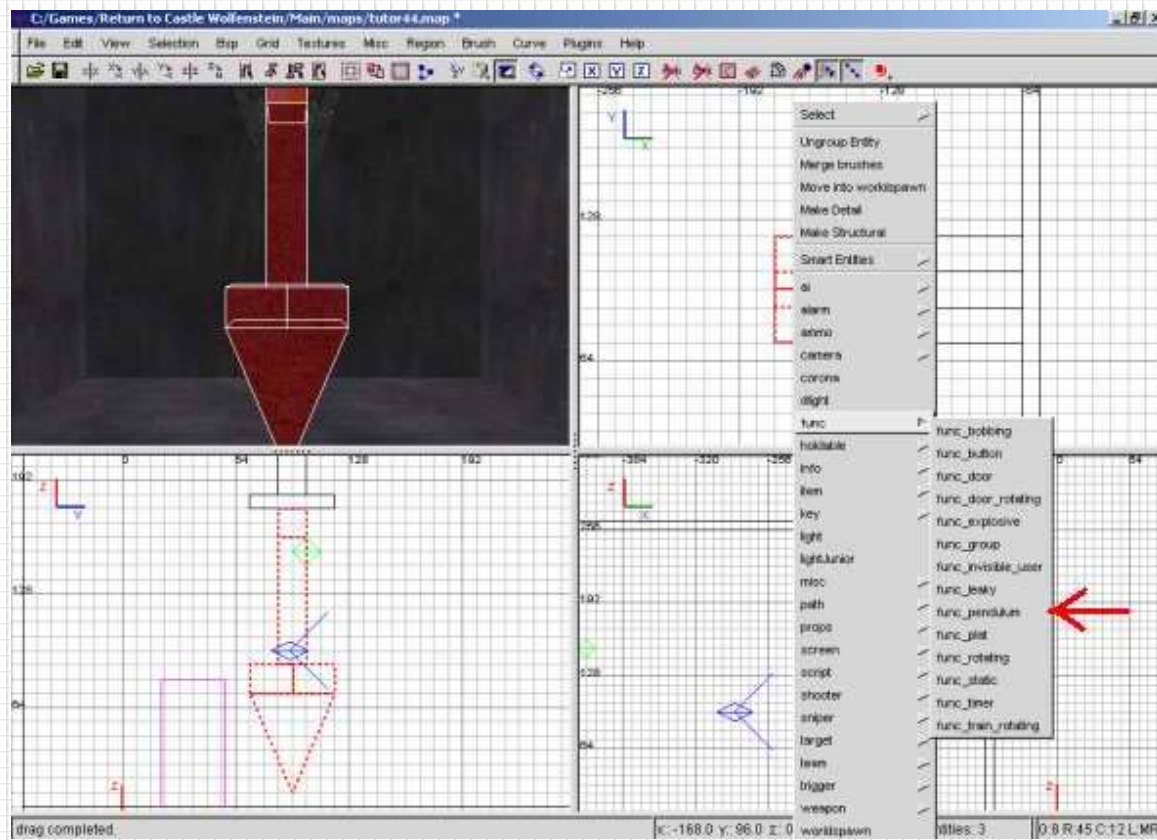
Nun wählst zuerst den Brush mit der Origin-Textur an ("STRG" + "linke Maustaste") und hinterher nach und nach von oben nach unten alle Brushes, die das Pendel bilden. Also so:



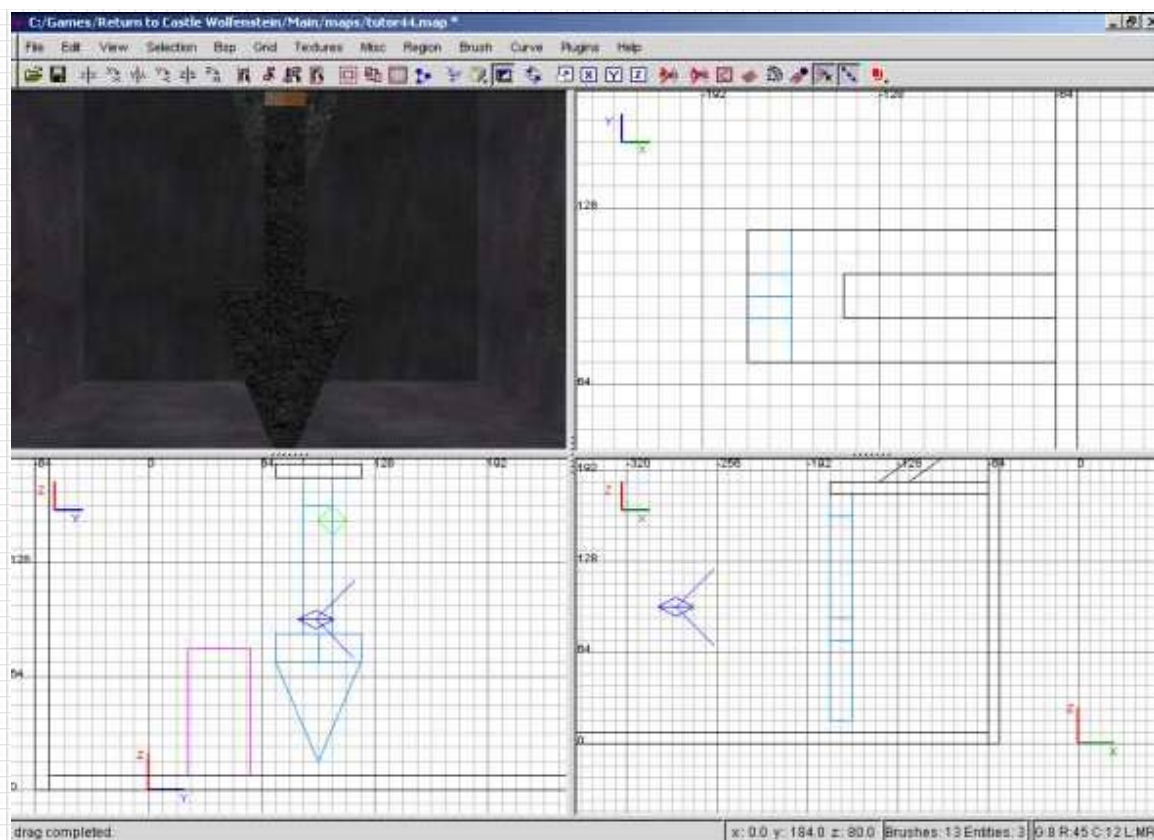
1. Zuerst den Brush mit der Origin-Textur

2. Dann den Brush mit der Nr. 2
3. Dann den Brush mit der Nr. 3
4. Dann den Brush mit der Nr. 4
5. Und zum Schluss den Brush mit der Nr. 5

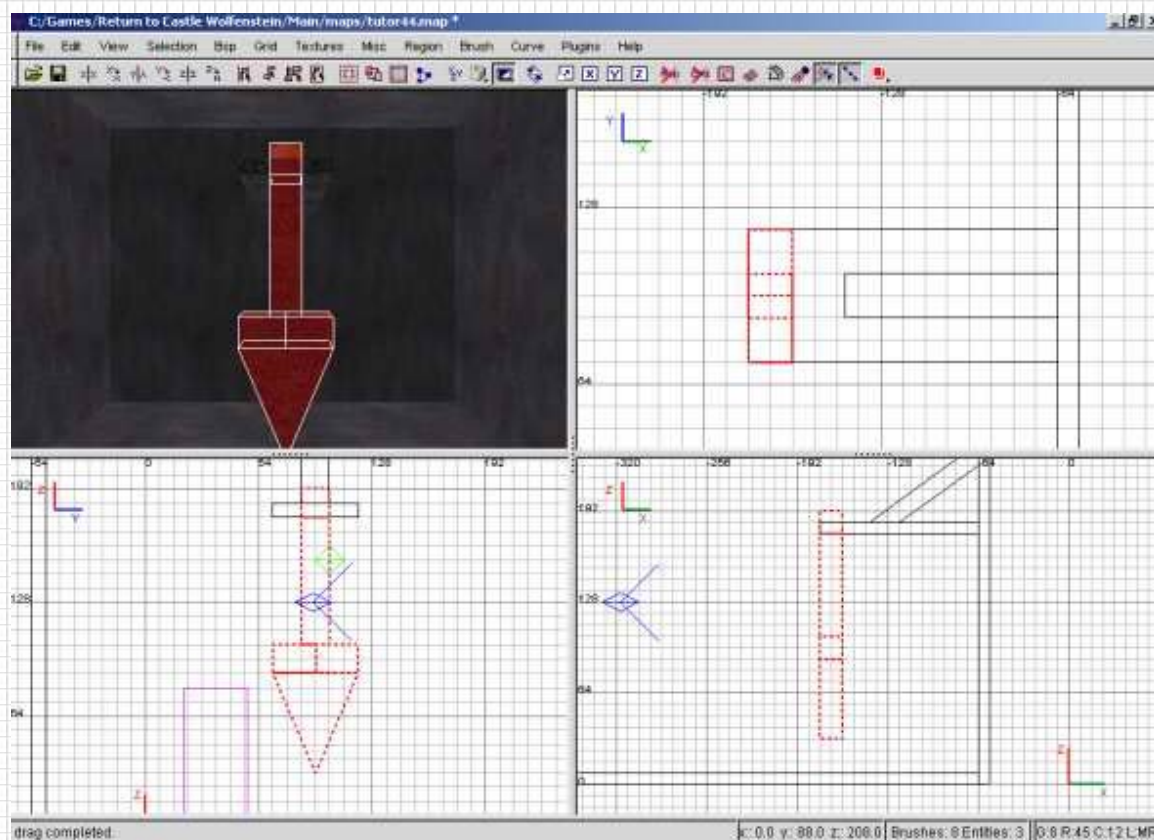
Nun lässt du alle 5 Brushes selektiert. Nun klickst du 2 x mit der rechten Maustaste und wählst "func" und dort als Unterpunkt "func_pendulum"



So, nun kannst du die "ESC"-Taste drücken, um alle Brushes zu deselektieren. Wenn du alles richtig gemacht hast, müssten alle Brushes, die das Pendel bilden, blaue Ränder haben:



Nun wären wir soweit fertig. Wenn du jetzt aber die Map compilierst, wirst du feststellen, dass das Pendel in der Luft hängt. Und wieso ist das so? Ganz einfach, der Origin-Brush wird ja im Spiel nicht dargestellt. Also wählst du nun das Pendel an ("STRG" + "linke Maustaste"). Nun wird plötzlich das ganze Pendel selektiert. Das hat natürlich seinen Sinn, immerhin haben wir ja eine func_pendulum erstellt, also eine Gruppe von Brushes, die zusammen ein Pendel bilden. Nun schieben wir das gesamte Pendel soweit nach oben, dass der Origin-Brush komplett in der Aufhängung sitzt und unser Schaft unten herauschaut:



Geschafft !! Jetzt kannst du die Map compilieren und gleich dein erstes Pendel bewundern.

[zurück zur Hauptseite](#)

183759



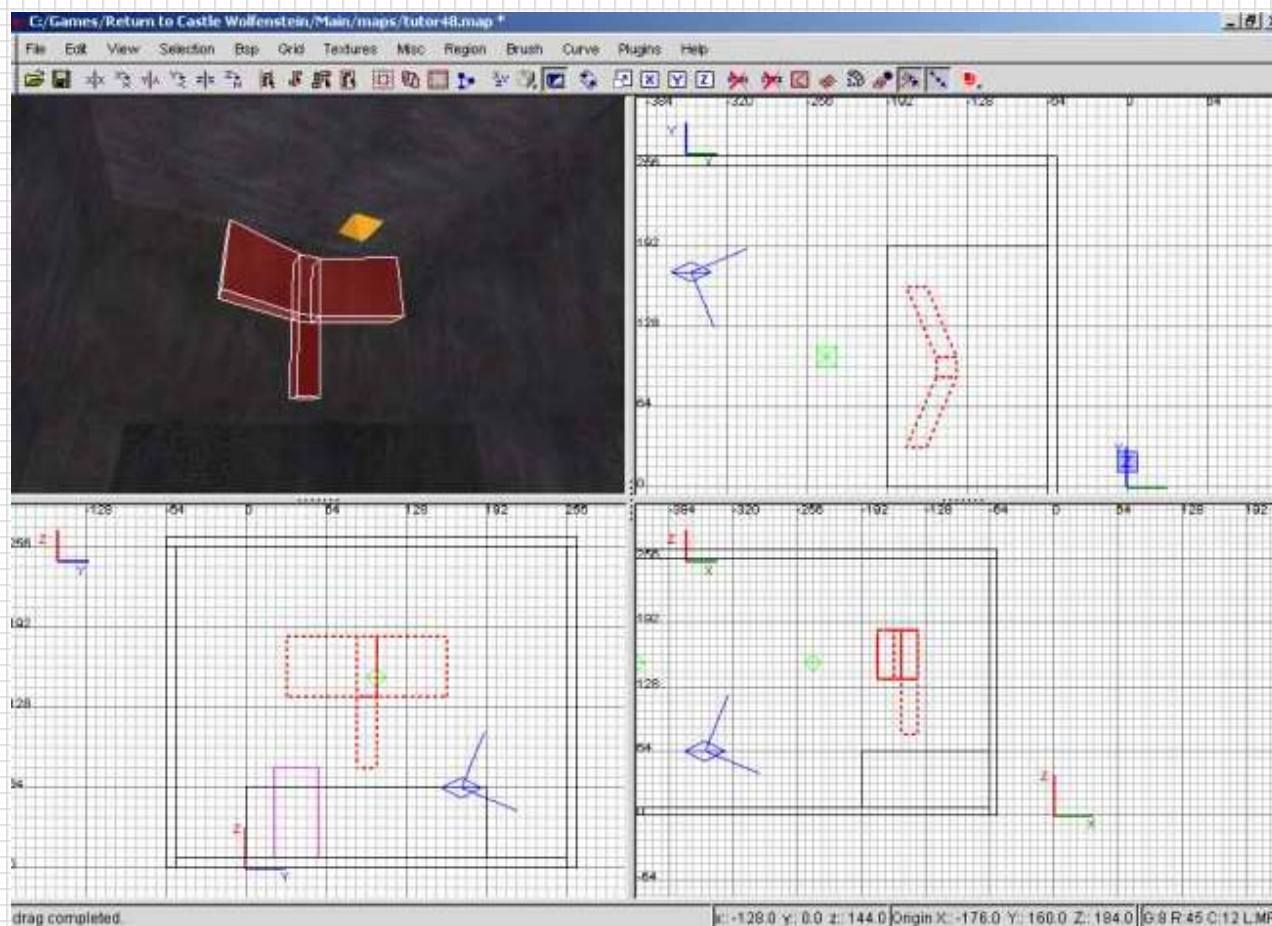
Rotierende Objekte:

verwendete Beispielpmap: "tutor47.map"

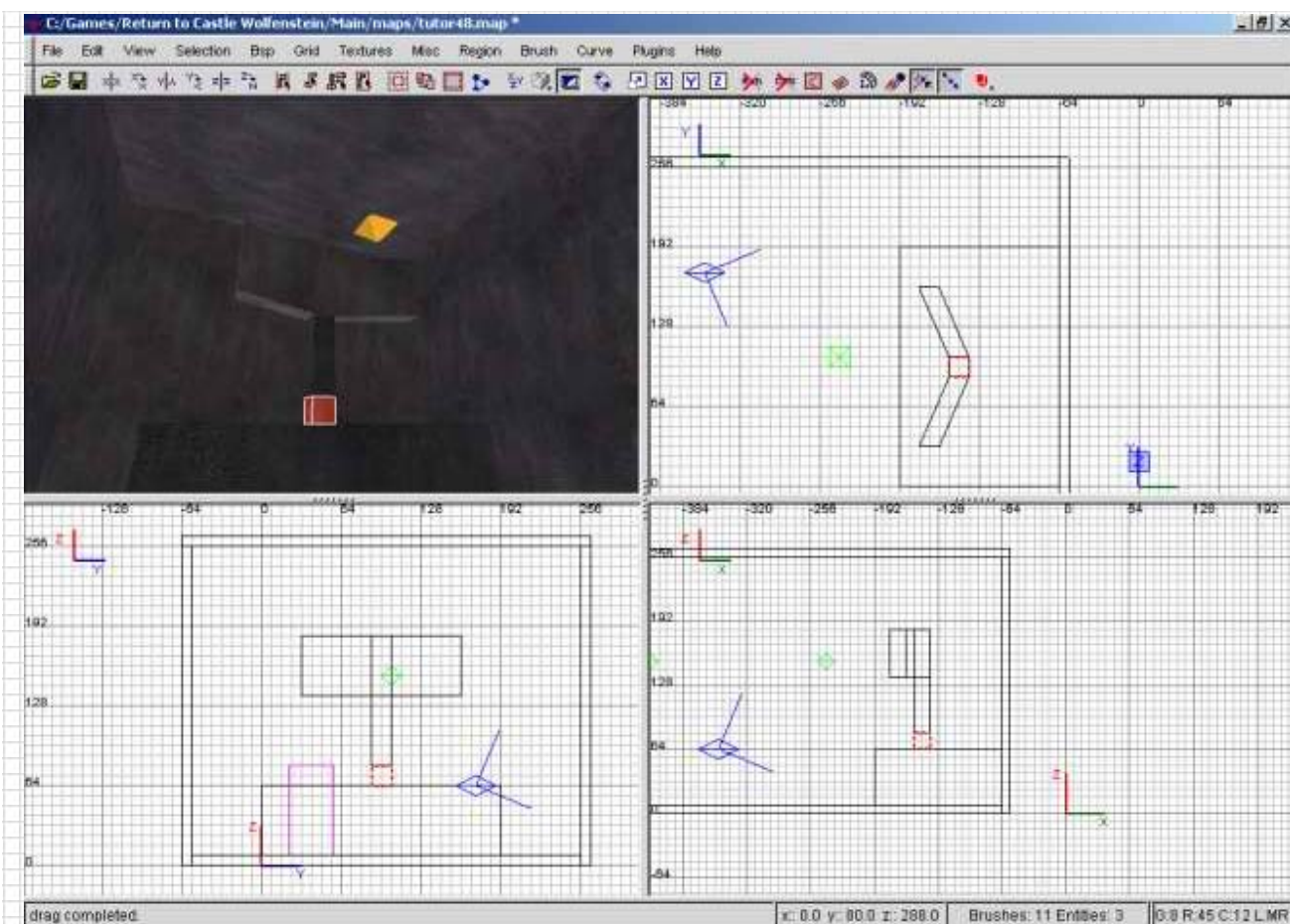
Ergebnismap: "tutor48.map"

So, nach dem Pendel will ich dir jetzt zeigen, wie man ein rotierendes Objekt, z.B. einen Radar oder eine Rührmaschine baut. Dazu geht man eigentlich genauso vor, wie beim Pendel auch - aber das wirst du selbst gleich merken.

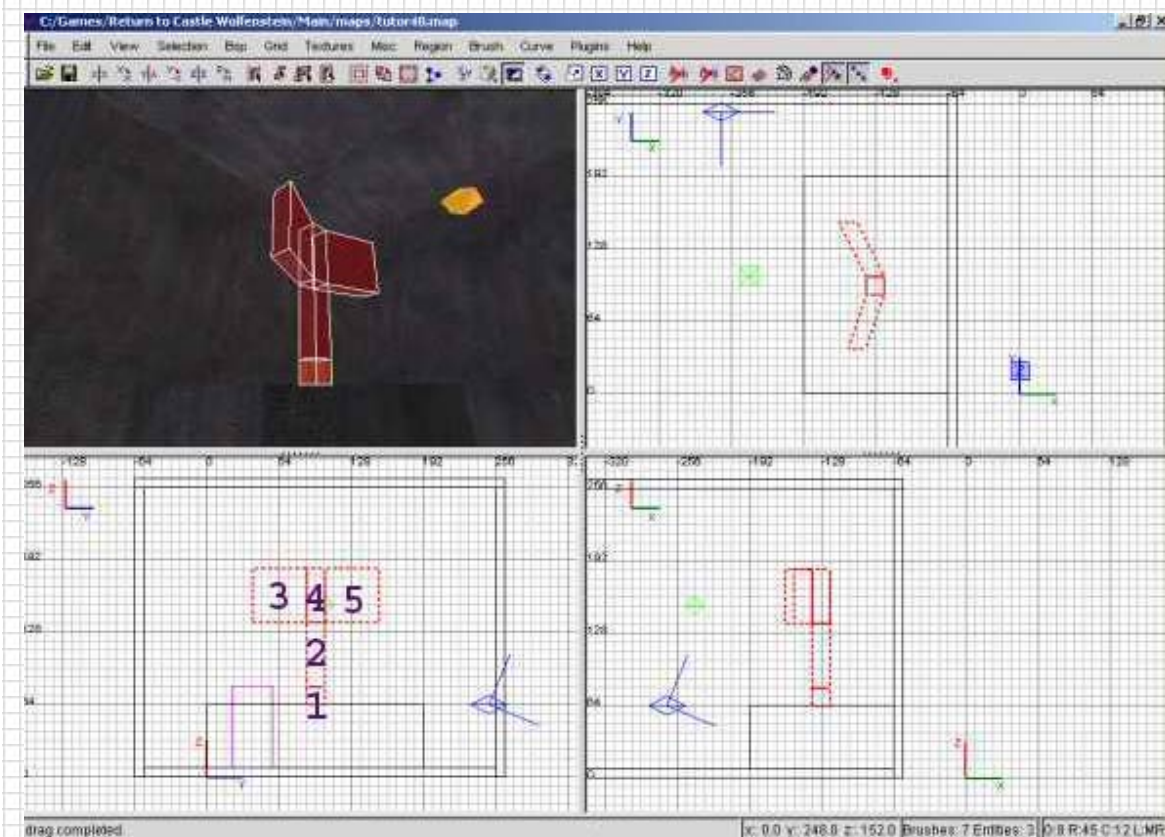
Ich habe in die Beispielpmap eine Art Podest gebaut, auf der wir unseren Radar setzen wollen. Nun baust du dir wieder eine Konstruktion, die später unser Radar geben soll. Meine sieht so aus:



Ich habe sie in die Luft gesetzt, weil untendran noch unser Origin-Brush kommt. So, jetzt drückst du wieder die "ESC"-Taste um die Radar-Anlage zu deselektieren. Nun erstellst du wieder einen Brush, der unten vom Schaft des Radars bis zum Podest reicht und 16 Units breit und 16 Units hoch ist. Diesem gibst du wieder die Origin-Textur (diese findest du unter "common/origin"). Nun sieht es ungefähr so aus:



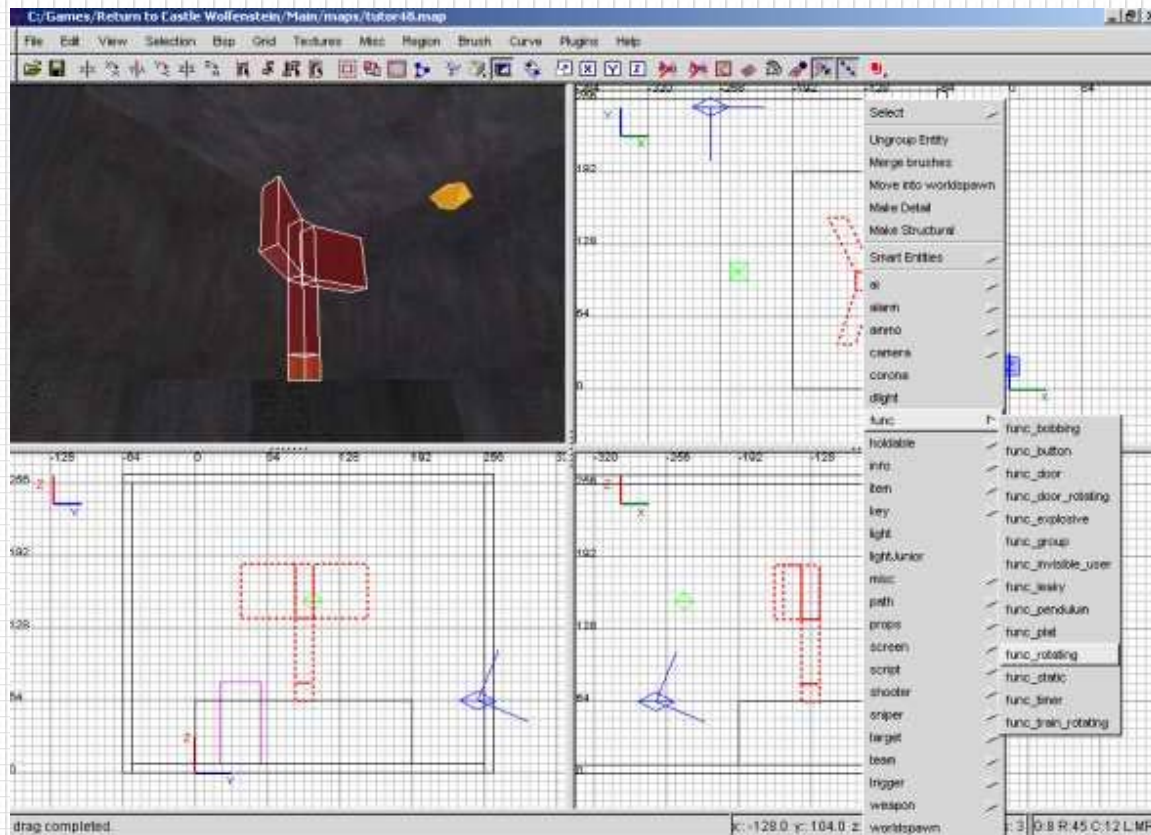
Nun drückst du wieder die "ESC"-Taste. Nun markierst du wieder nacheinander die Brushes von unten nach oben. Du kannst dir immer merken, dass man den Origin-Brush als erstes markiert und dann alle angrenzenden nacheinander:



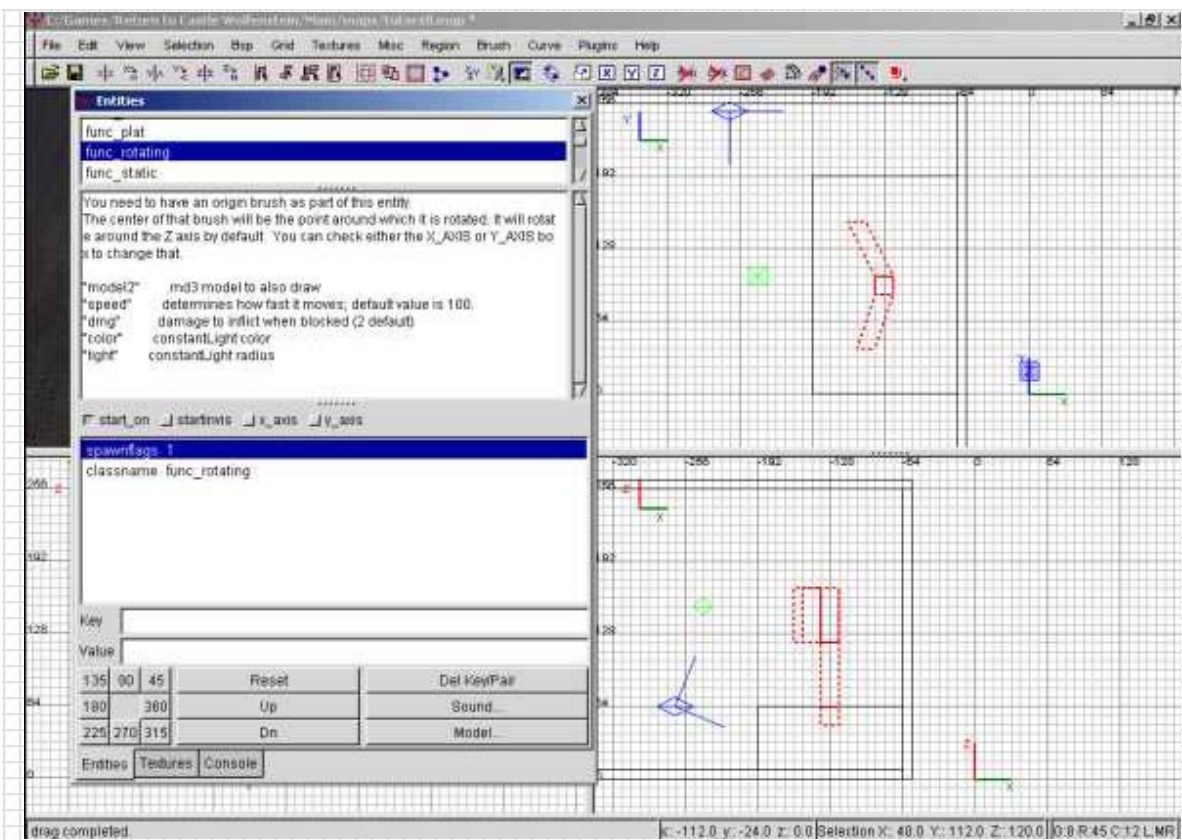
1. Zuerst den Origin-Brush

2. Dann den Schaft
3. Dann den einen Flügel des Radars
4. Dann den mittleren Brush des Radars
5. Zum Schluss den anderen Flügel des Radars

Nun klickst du 2 x mit der rechten Maustaste und wählst "func/func_rotating":



Nun deselektierst du alles. Wenn du alles richtig gemacht hast, erscheinen die Seiten der Radar-Brushes blau. Nun wären wir soweit fertig - aber wir müssen den Radar ja noch in das Podest schieben, dass der Origin-Brush ganz im Podest verschwindet. Immerhin wird er ja im Spiel nicht dargestellt und du willst bestimmt keinen schwebenden Radar. Also verschieben wir den Radar um 16 Units nach unten und drücken noch die Taste "N" um die Entities festzulegen:



Hier ist es wichtig, dass du einen Häkchen bei "start_on" setzt - sonst bewegt sich dein Radar kein bisschen. Nun schliesst du das Entity-Fenster und drückst "ESC" um alles zu deselektieren. Nun kannst du die Map compilieren und deinen Radar bewundern.

Zum Schluss erkläre ich dir noch schnell die anderen Keys, die du hier eintragen kannst:

- "Speed": hier kannst du einstellen, wie schnell sich der Radar drehen soll
- "dmg": dies gibt an, wieviel Schaden ein Spieler nimmt, wenn er den Radar berührt
- "color": Hier kannst du angeben, welche Farbe der Licht-Radius haben soll
- "light": Hier kannst du angeben, wie gross der konstante Licht-Radius ist

Und noch die Kästchen, die du durch einen Klick aktivieren kannst:

- Start_on: Mit dieser Flag beginnt sich die Entity gleich beim Levelstart zu drehen
- X Axis: Hiermit wird die Rotationsachse auf die X Ebene gelegt
- Y Axis: Hiermit wird die Rotationsachse auf die Y Ebene gelegt

WICHTIG: Curves kannst du nicht zu einem func_rotating machen, da sonst das gesamte func_rotating nicht funktioniert.

[zurück zur Hauptseite](#)

183759

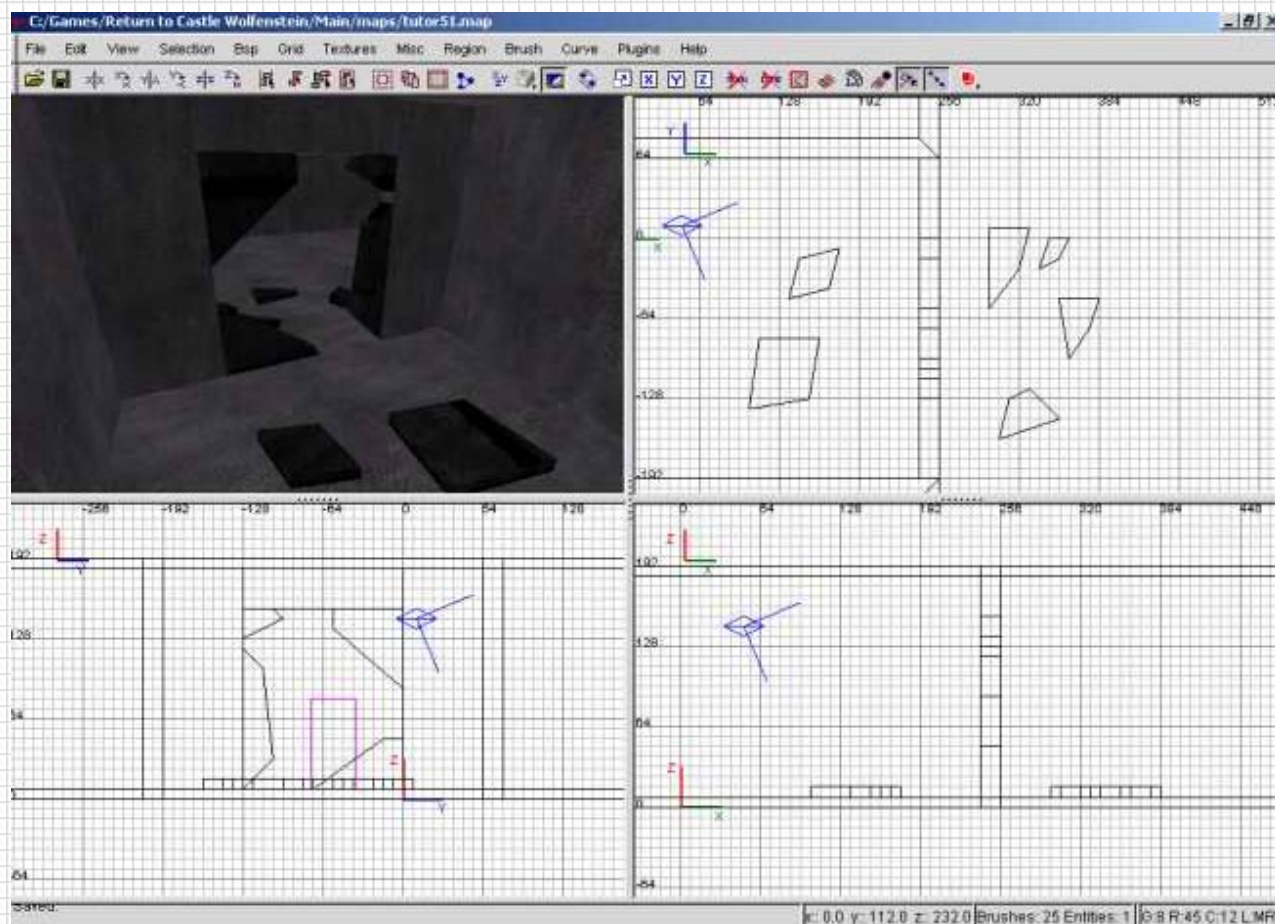


Überreste nach dem Sprengen:

verwendete Beispiemap: "tutor51.map"

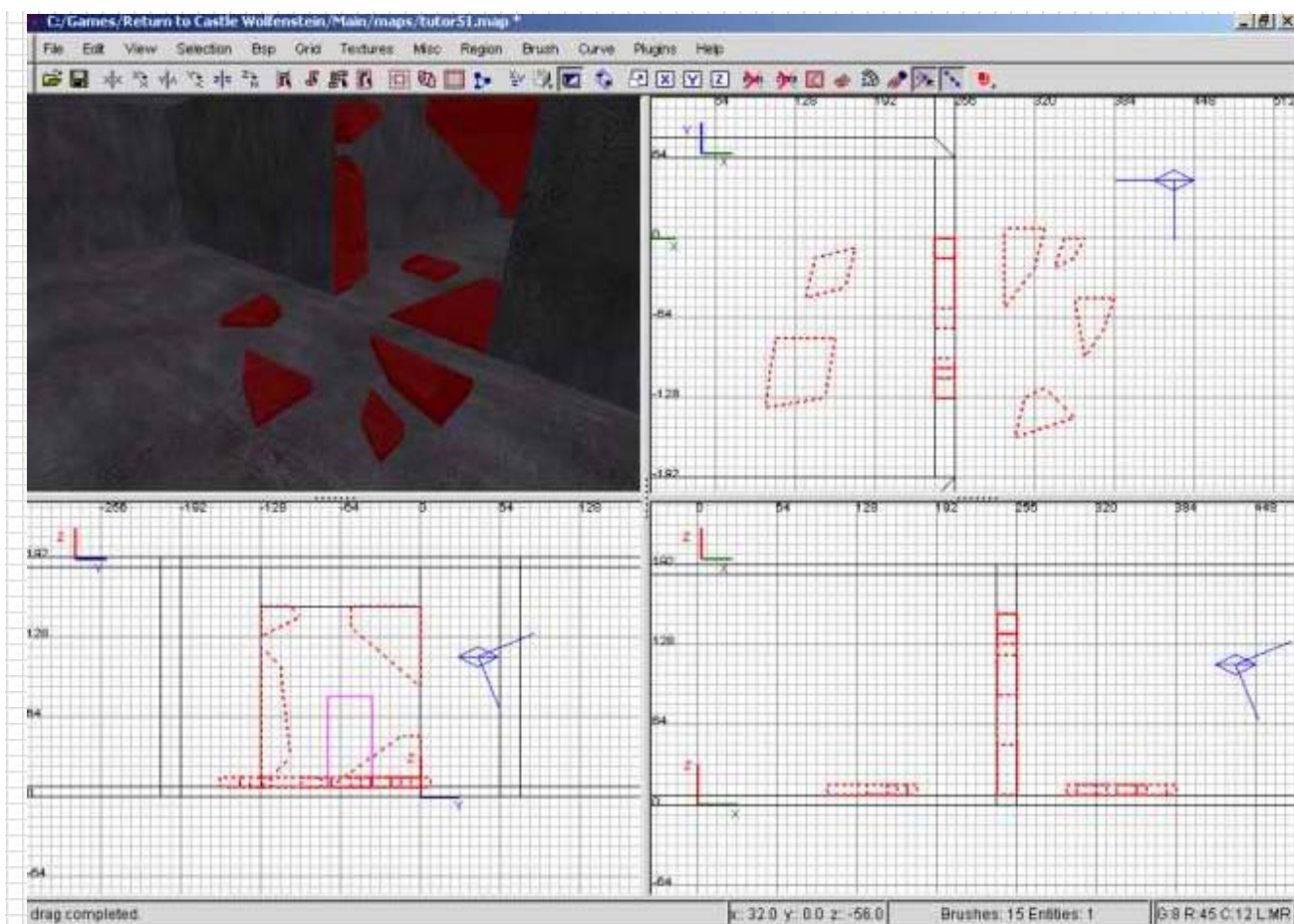
Ergebnismap: "tutor52.map"

Wie du sicher bei den Explosionen gesehen hast, bleiben da ja keine Bruchstücke liegen. Zwar liegen meistens noch ein paar kleine Brocken herum, aber die verschwinden ja dann. Hier will ich dir zeigen, wie man nach einer sauberen Explosion auch ein wenig Sauerei liegenbleibt. Dazu brauchen wir natürlich erstmal ein geeignetes Objekt, welches dann später gesprengt wird:



Also gut - hier siehst du die Beispiemap - ich habe hier schon die kaputte Wand gebaut, ausserdem liegen ein paar Bruchstücke um die Wand herum.

Nun selektierst du einfach alle Bruchstücke der Wand:

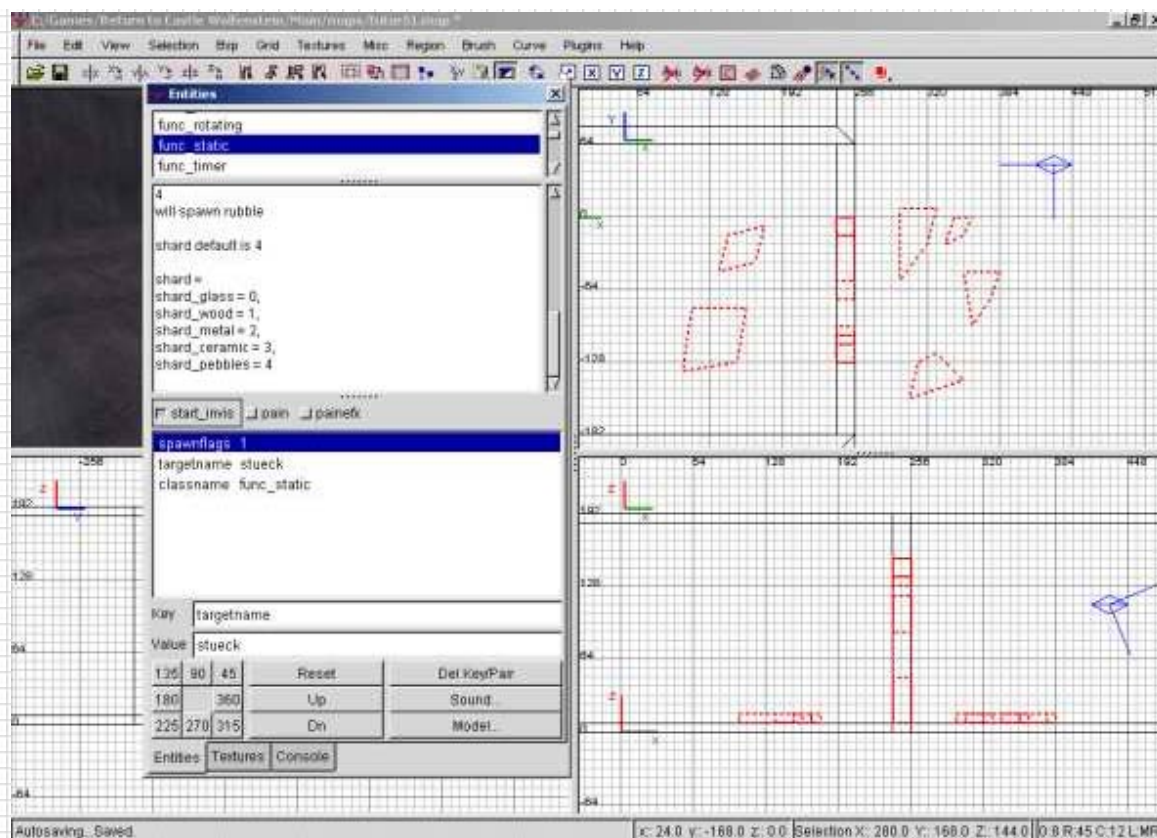


So, nun lässt du alle Stücke selektiert, und klickst 2 x mit der rechten Maustaste. Hier wählst du "func/func_static". Nun drückst du die Taste "N" um das Entity-Fenster zu öffnen. Nun gibst du folgendes ein:

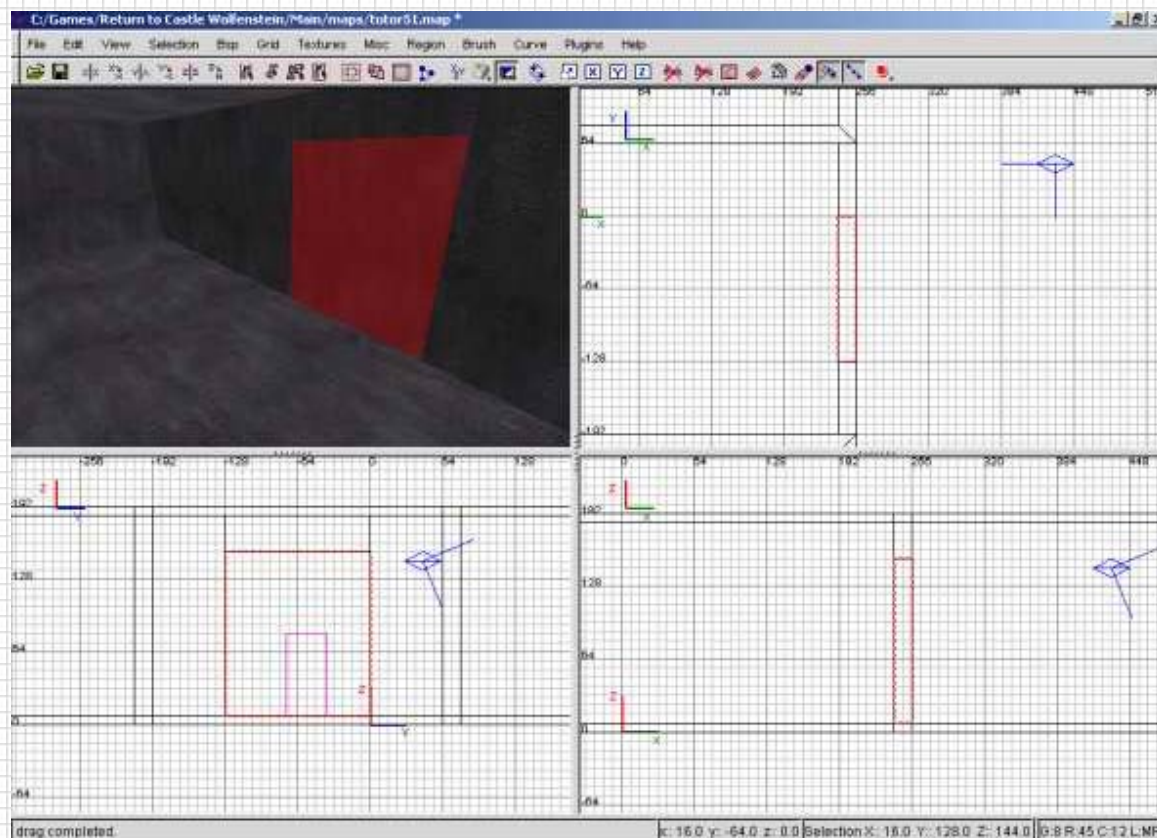
Key: "targetname"

Value: "stueck"

ausserdem aktivierst du noch die Funktion "start_invis":



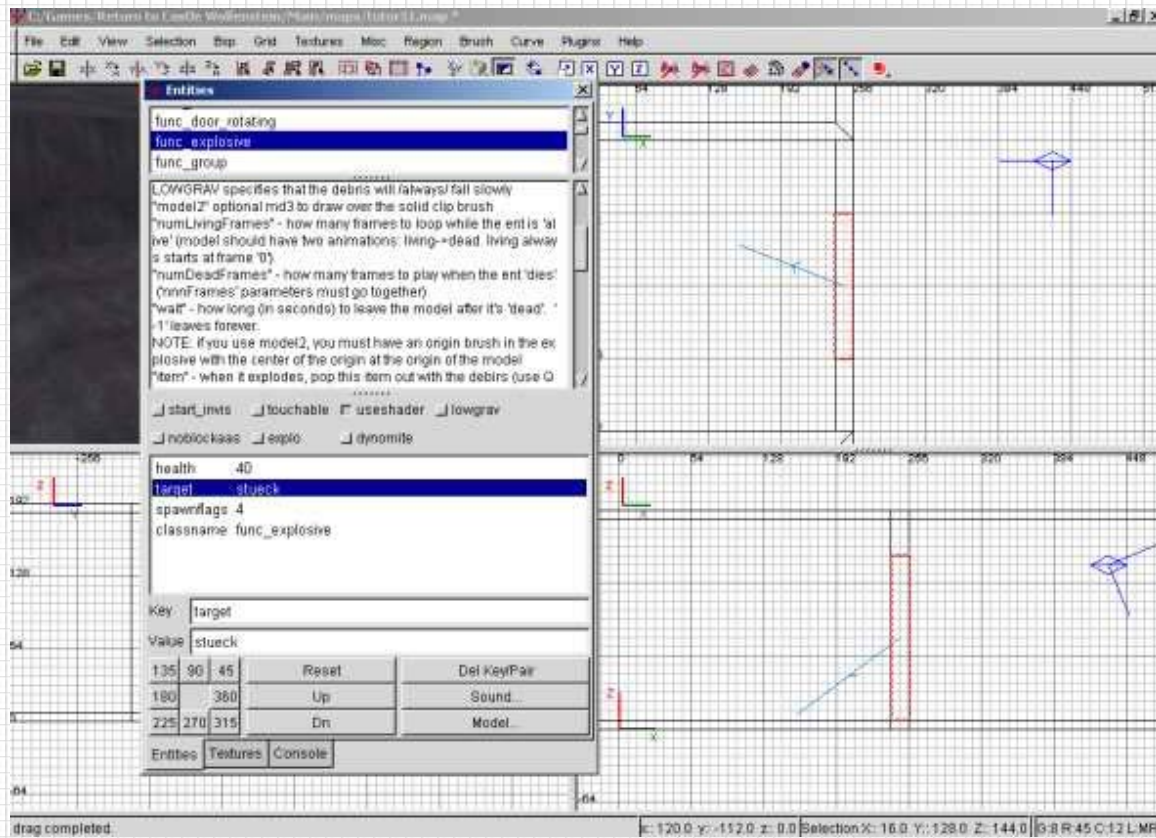
So, hier siehst du nun alle notwendigen Keys fertig eingetragen. Nun kannst du das Entity-Fenster schliessen. Nun drückst du die Taste "H" um alle Bruchstücke zu verstecken. Nun bauen wir in unsere Maueröffnung eine schöne, ganze Mauer. Du kannst hier auch eine Türen-Textur verwenden. Wichtig ist dabei nur, dass die Türe ja nicht funktioniert,



Nun selektierst du nur den neuen Wand-Brush und klickst 2x mit der rechten Maustaste und wählst "func/func_explosive". Nun drückst du gleich noch die "N"-Taste, denn wir wollen auch hier noch ein paar Keys hinzufügen:

- Key: "target"
Value: "stueck"
- Key: "health"
Value: "40"

und dann aktivierst du noch die Funktion "useshader". Das brauchst du nicht unbedingt, aber wenn du auf dem explodierenden Brush eine Textur mit einem Shader verwendest, musst du diese Funktion aktivieren, sonst funktioniert der Shader nicht.



So, hier erkläre ich dir schnell, was wir jetzt eigentlich gemacht haben. Wir haben den Bruchstücken ja den Namen "stueck" zugewiesen und nun zieht die ganze Mauer auf die Bruchstücke. Somit haben wir die beiden Entities verknüpft. Den Health-Wert von 40 habe ich deshalb gewählt, weil er ziemlich niedrig ist. Natürlich kannst du auch einen höheren Wert eingeben oder auch die Funktion "DY-NO-MITE" aktivieren.

Drückst du nun die "ESC"-Taste um alles zu deselektieren, müsste auf der einen Seite ein blauer Strich erscheinen, der die ganze Wand mit einem Bruchstück verbindet. Dann mal nichts wie compilieren und auf die Wand schießen.

Wir hätten es auch anders machen können, in dem du zuerst die Bruchstücke und die Mauer baust und dann die beiden Entities verknüpfst und zum Schluss die Türe in die Öffnung geschoben.

[zurück zur Hauptseite](#)

183759

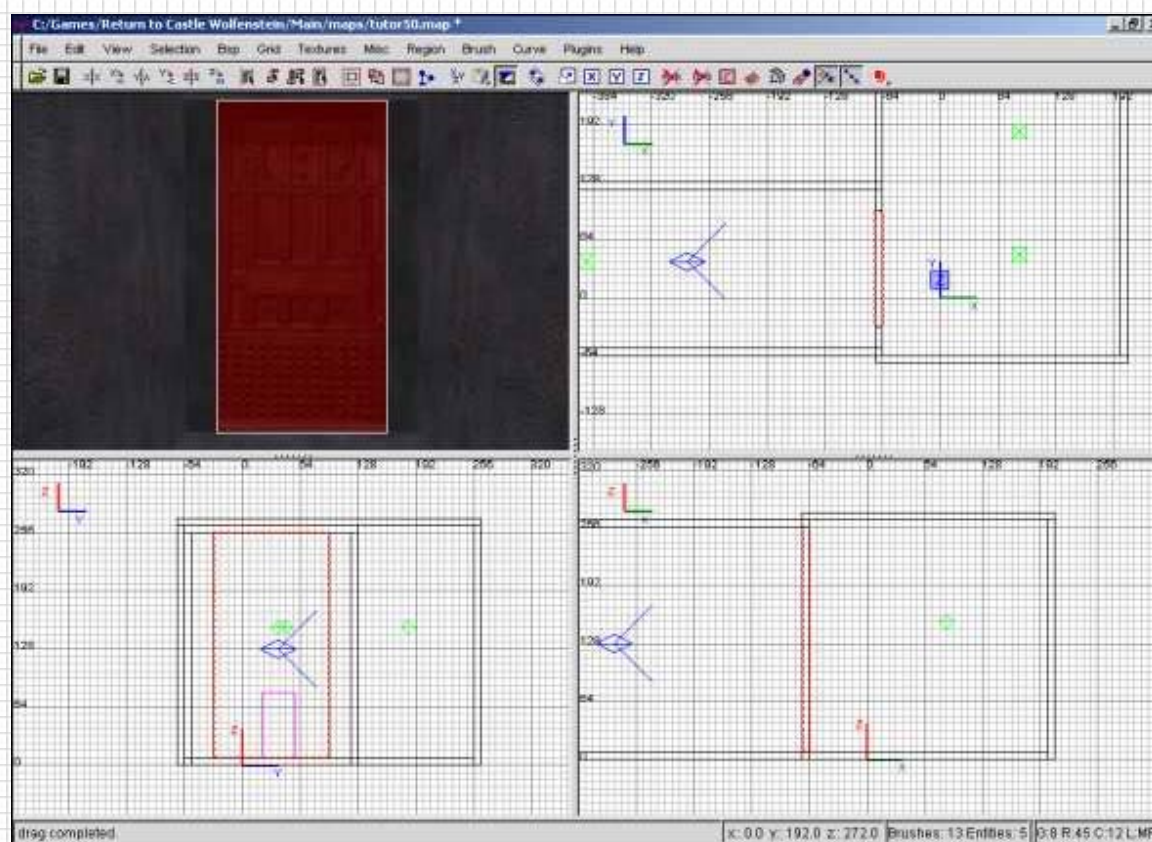


Eine drehende Türe erstellen:

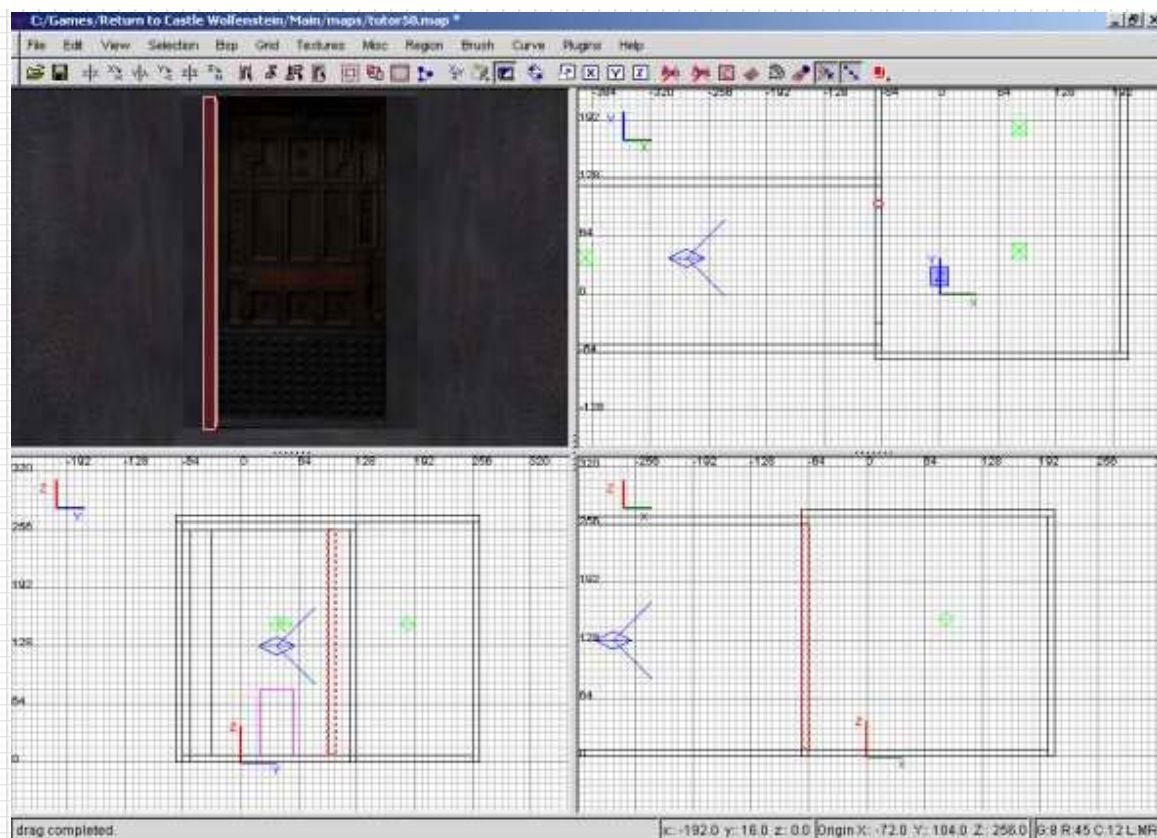
verwendete Beispielmap: "tutor49.map"

Ergebnismap: "tutor50.map"

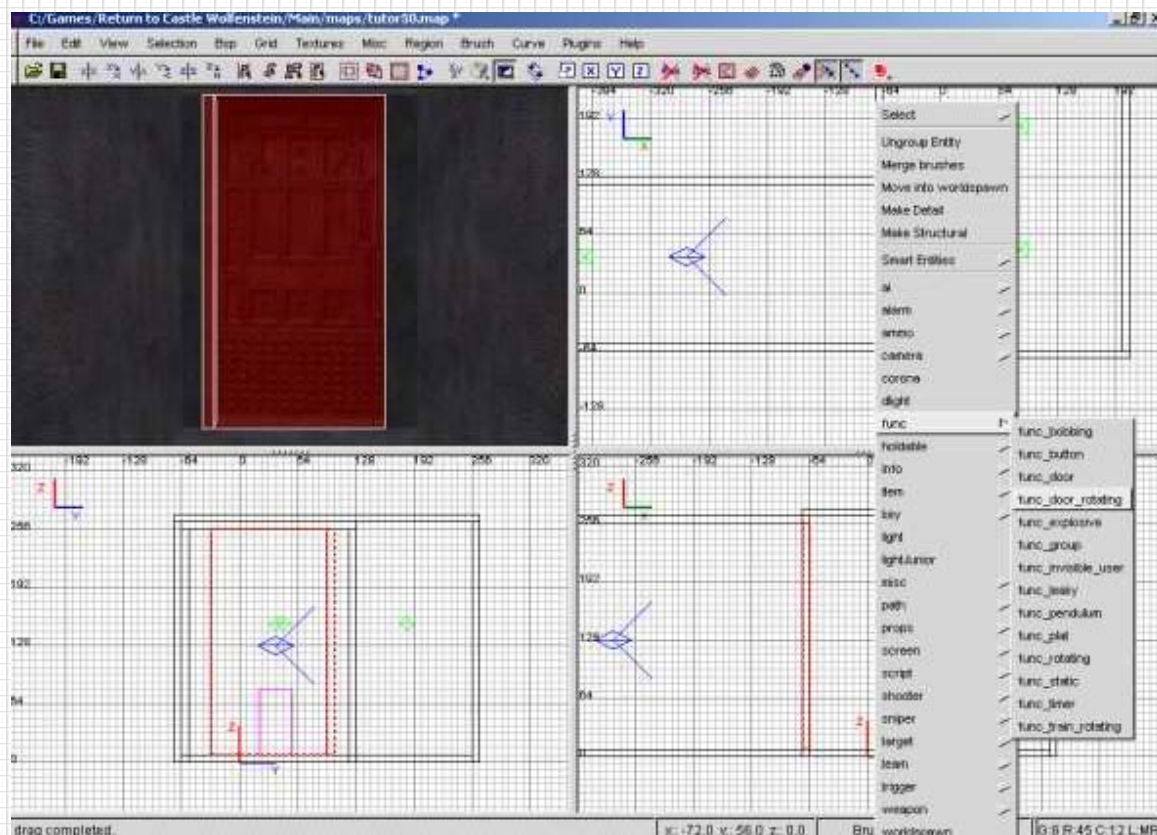
Nun will ich dir zeigen, wie man eine Türe "real" nachbauen kann - also eine Türe, die aufschwingt. Bisher habe ich dir ja nur Türen gezeigt, die sich zur Seite hin öffnen. Dazu brauchen wir natürlich zuerst eine Textur für die Türe. Dazu benutzt du am besten die Textur "castle_door/door_c17". Ich habe dir schon in der Beispielmap einen Türsturz gebaut, dort hinein setzt du nun einen Brush, der genau passt und 8 Units breit ist:



Nun deselektierst du alles und erstellst an der linken Seite dieser Türe einen Brush, der genauso hoch ist, wie die Türe und 8 Units breit und 8 Units lang ist. Diesen belegst du wieder mit der "Origin"-Textur (common/origin). Dieser Brush ist quasi unsere Türangel, an der die Türe hängt:



Nun deselektierst du alles, indem du die "ESC"-Taste drückst. Nun selektierst du zuerst den Origin-Bush und danach die Türe, und klickst 2 x mit der rechten Maustaste und wählst func/func_door_rotating:



Hier erkläre ich dir noch schnell die wichtigsten Entities für die Türe:

- SHOOT-Thru: Gibt an, ob die Spieler durch die Türe schießen können. Dann gibst du als Key "shoot_thru_scale" ein und als Value einen Wert zwischen 0 und 1. 0 bedeutet, dass alle Schüsse durch die Türe absorbiert werden und der Effekt eigentlich

verloren geht. 1 bedeutet, dass alle Schüsse durch die Türe kommen.

- Speed: gibt an, wie schnell die Türe aufschwingen soll.
- Time: wie lange die Türe braucht, bis sie sich beginnt zu öffnen.
- Force: Die Türe öffnet sich, selbst wenn sie geblockt wird.
- Team: Wenn du eine zweite Türe einbaust, kannst du sie zu einem Team vereinen, so dass sie sich auch synchron öffnen.

Nun deselektierst du unsere Türe. Wenn du alles richtig gemacht hast, müsste der Brush jetzt blau umrandet sein. Nun kannst du deine Map compilieren und die Türe benutzen.

Eine Drehtüre erstellen:

Wenn du eine richtige Drehtüre haben willst, erstellst du mit 4 Brushes ein Drehkreuz. Diese Brushes müssen sich natürlich berühren. In der Mitte erstellst du dann wieder einen Origin-Brush und machst aus dem Gebilde eine func_rotating, da sich ja dann die Türe um 360° drehen soll. Beim Selektieren musst du nur darauf achten, dass du zuerst wieder den Origin-Brush selektierst und dann erst die 4 Brushes.

Ich spare mir an dieser Stelle, dir die ganzen func_rotating Keys zu erklären, da ich das ja schon in einem anderen Thema gemacht habe.

[zurück zur Hauptseite](#)





Schnee erstellen:

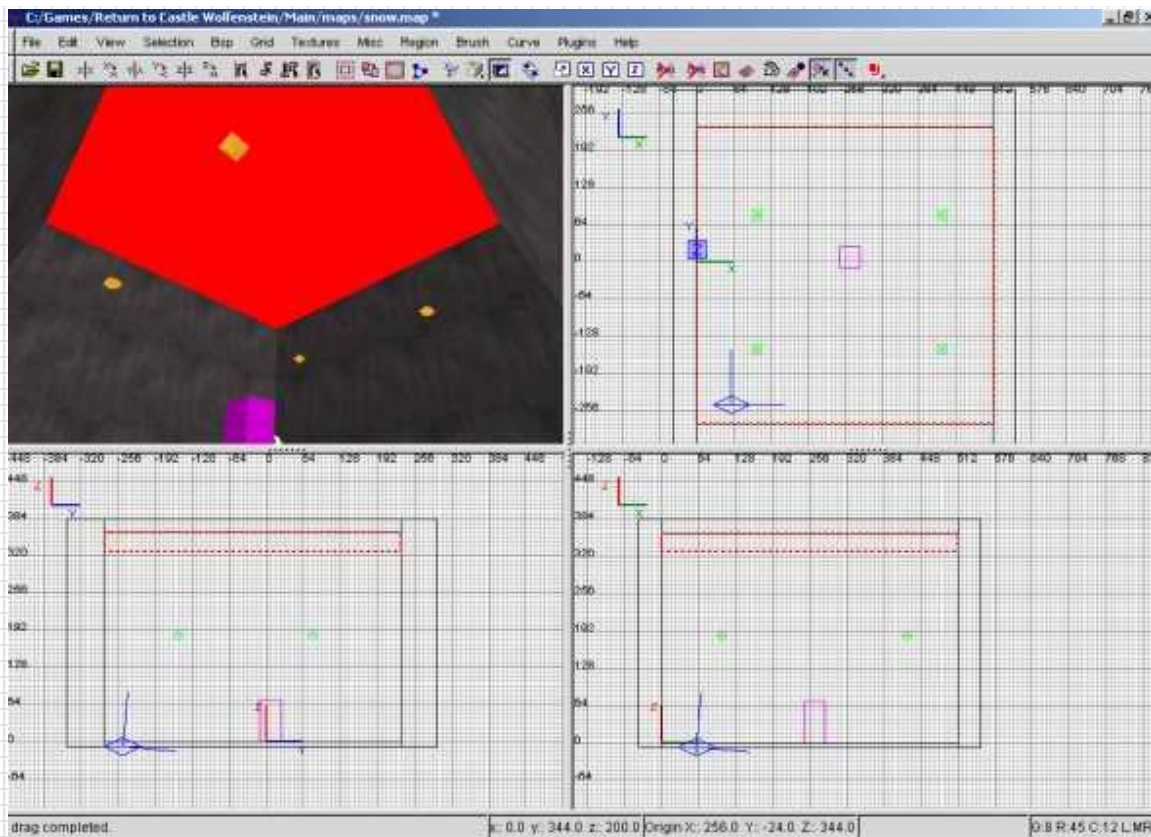
verwendete Beispiemap: "tutor52.map"
Ergebnismap: "tutor53.map"

Hier will ich dir zeigen, wie man es in der eigenen Map schneien lässt. Schnee solltest du aber in der Map sparsam einsetzen ,da Schnee sehr performance-lastig ist. Dazu klickst du 2 x mit der rechten Maustaste und wählst "misc". Hier befinden sich verschiedene Entities, die Schnee in der Map darstellen:

camera	➤ misc_bubbles16
corona	misc_bubbles32
dlight	misc_bubbles64
func	➤ misc_bubbles8
holdable	➤ misc_firetrails
info	➤ misc_flak
item	➤ misc_gamemodel
key	➤ misc_grabber_trap
light	misc_light_surface
lightJunior	misc_mg42
misc	➤ misc_model
path	➤ misc_portal_camera
props	➤ misc_portal_surface
screen	➤ misc_snow128
script	➤ misc_snow256
shooter	➤ misc_snow32
sniper	➤ misc_snow64

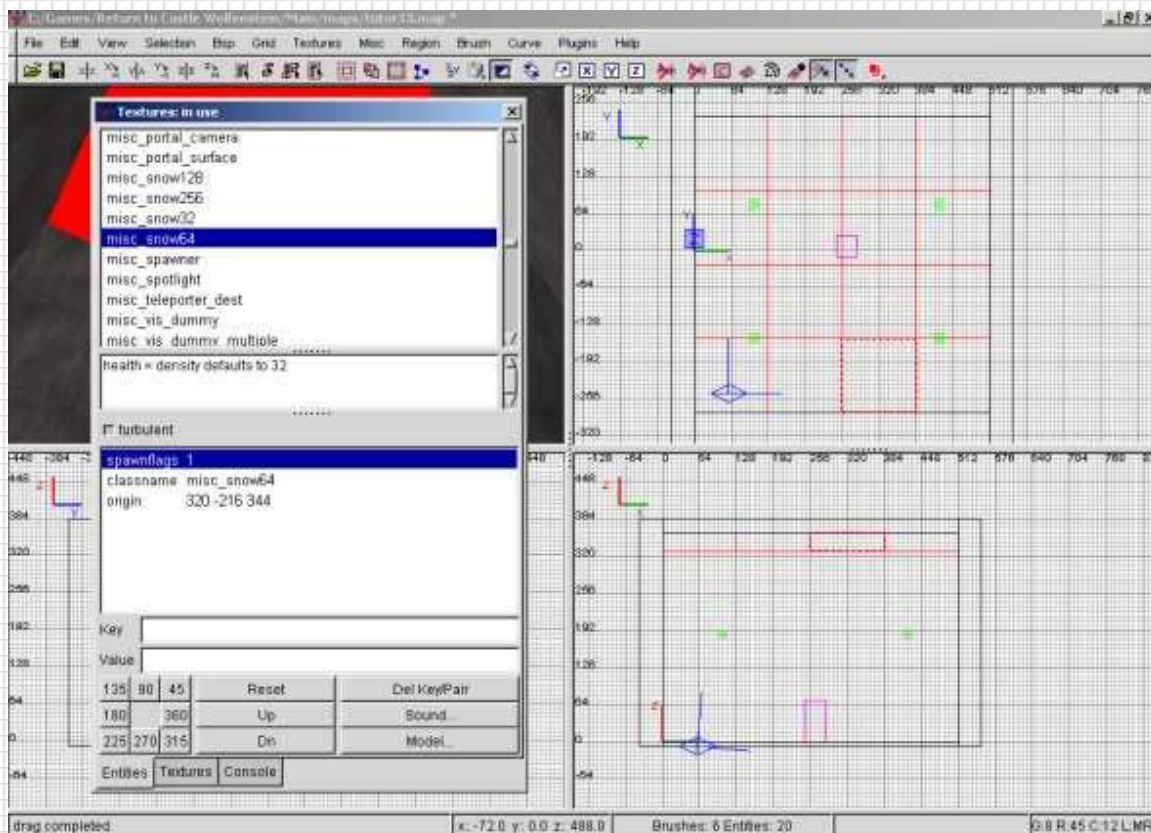
- misc_snow32 = erzeugt ein Snow-Entity, dass 64 Units lang ist
- misc_snow64 = erzeugt ein Snow-Entity, dass 128 Units lang ist
- misc_snow128 = erzeugt ein Snow-Entity, dass 256 Units lang ist
- misc_snow256 = erzeugt ein Snow-Entity, dass 512 Units lang ist

Es gibt bei all diesen Entities nur eine Funktion, die du anwählen kannst. Diese heisst "turbulent", dass bedeutet, der Schnee fällt sehr stürmisch vom Himmel. Ich habe dir in der Beispiemap ein misc_snow256 eingebaut. Wie du sehen kannst, ist der Raum so gross, wie das Entity gross ist.



Entities lassen sich nicht in der Grösse verändern, also musst du einen Raum so gross machen, dass das Entity genau passt. Notfalls kannst du ja mehrere Entities aneinander setzen. Nun kannst du die Map compilieren, und kannst es schon bewundern, wie die Schneeflocken tanzen.

Nun wollen wir aber ein bisschen mehr Schnee in der Map haben. Dazu löschst du das Snow-Entity und ersetzt es durch mehrere misc_snow64-Entities:



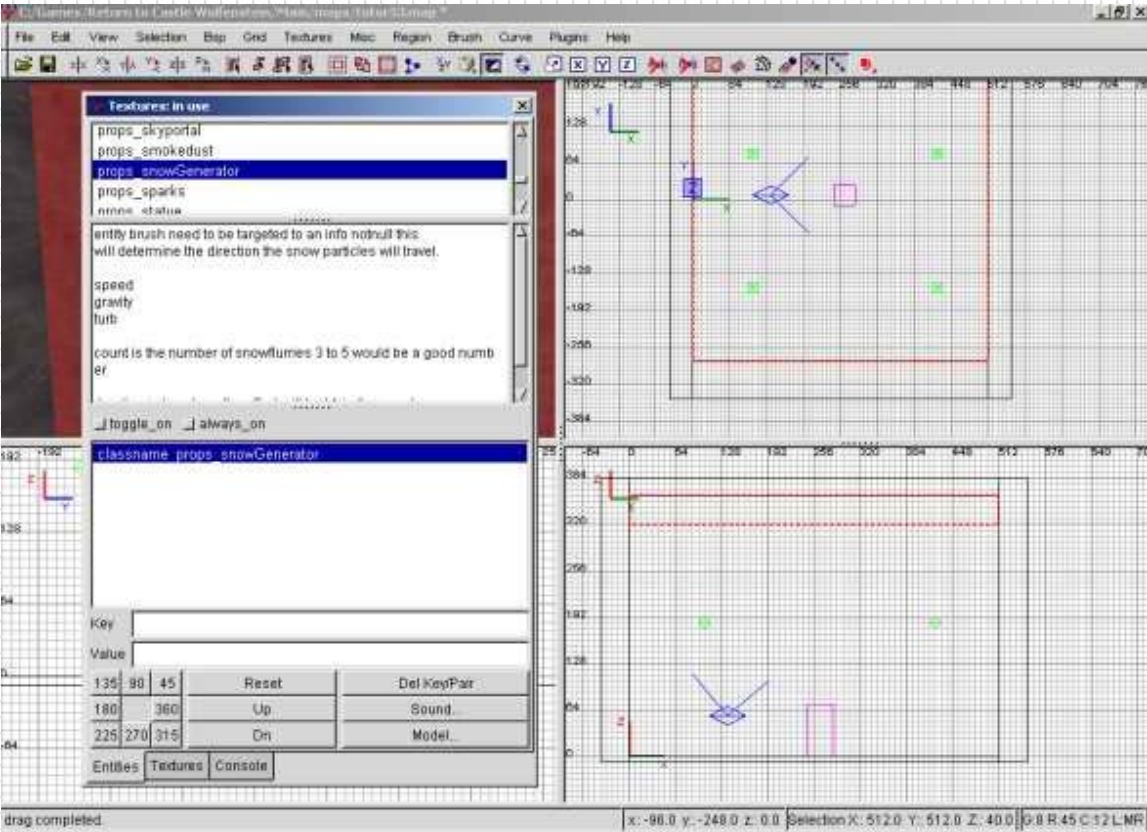
Wie du sehen kannst, habe ich hier 16 Entities eingesetzt. Nun wählst du jedes Entity einzeln an und drückst die Taste "N" um das Entity-Fenster zu öffnen. Hier kannst du die Funktion "turbulent" aktivieren, die ich dir schon erklärt habe. Nun schliesst du das Fenster und drückst "H" um das Entity zu verstecken. Nun wiederholst du das bei allen Entities. Wenn du jetzt die Map compilierst, wirst du feststellen, dass der Schnee wesentlich häufiger und durcheinander von der Decke fällt. Wenn du willst, kannst du auch noch mehrere Ebenen aus diesen Entities erstellen um einen richtigen Schneesturm zu erstellen.

Im Grunde solltest du Schnee sehr sorgfältig in deine Map einbauen und die Map auch auf schwächeren Systemen testen Es kann sein, dass die Map ja bei dir läuft, aber auf langsameren Systemen ruckelt die Map.

Nun zeige ich dir, wie man ein Schnee-Feld erstellt, dessen Grösse du selbst wählen kannst. Nun erstellst du einen Brush, der genau so gross ist, wie der Raum, in dem der Brush sitzen soll. Dann klickst du 2 x mit der rechten Maustaste und wählst "props" und hier "snowgenerator":

key	➤	props_FireColumn
light		props_flamebarrel
lightJunior		props_flamethrower
misc	➤	props_footlocker
path	➤	props_gunsparks
props	➤	props_locker_tall
screen	➤	props_radio
script	➤	props_radioSEVEN
shooter	➤	props_shard_generator
sniper	➤	props_skyportal
target	➤	props_smokedust
team	➤	props_snowGenerator
trigger	➤	props_sparks
weapon	➤	props_statue
worldspawn		props_statueBRUSH

Nun drückst du die Taste "N" um das Entity-Fenster zu öffnen:



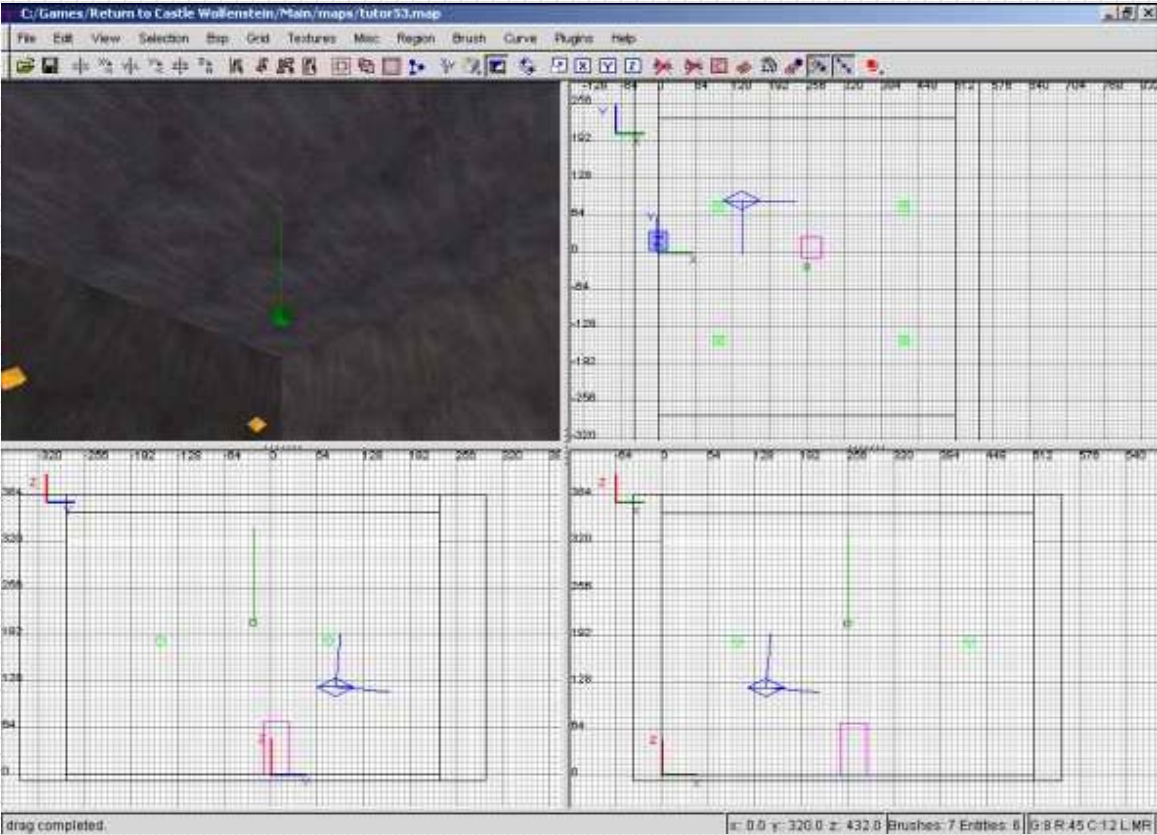
Hier erkläre ich dir schnell, welche Keys es bei diesem Entity gibt:

- Key: "gravity" -> gibt die Schwerkraft an, die herrscht. 800 ist der Standart.
Value: "800"
- Key: "count" -> gibt an, wieviel Schneeflocken auf einmal erstellt werden
Value: "6"
- Key: "duration" -> Zeit in Milisekunden, wie lange der Flug der Schneeflocken dauert
Value: "1200"

Ausserdem gibt es noch 2 Funktionen, die du anwählen kannst:

toggle_on: Der Schneefall wird durch einen Schalter eingeschalten. Wie das geht, zeige ich dir in einem anderem Thema
always_on: Der Schnee fällt von Anfang an

Nun müssen wir dem Entity noch mitteilen, in welche Richtung der Schnee fallen soll. Dazu deselektierst du das Entity, indem du die Taste "ESC" drückst. Nun drückst du 2 x die rechte Maustaste und wählst "info" und hier als Unterpunkt "info_notnull". Nun erscheint ein kleiner Kasten, den du unter das Schnee-Entity plazierst. Nun wählst du erst das Schnee-Entity an und danach das Notnull-Entity. Nun drückst du "STRG" + "K" um die beiden Entites zu verbinden:



Wenn du alles richtig gemacht hast, kannst du die Verbindungslinie zwischen dem Schnee-Entity und dem Notnull-Entity sehen. Nun kannst du gleich die Map compilieren und noch etwas mit dem Keys herumspielen.

[zurück zur Hauptseite](#)

183759



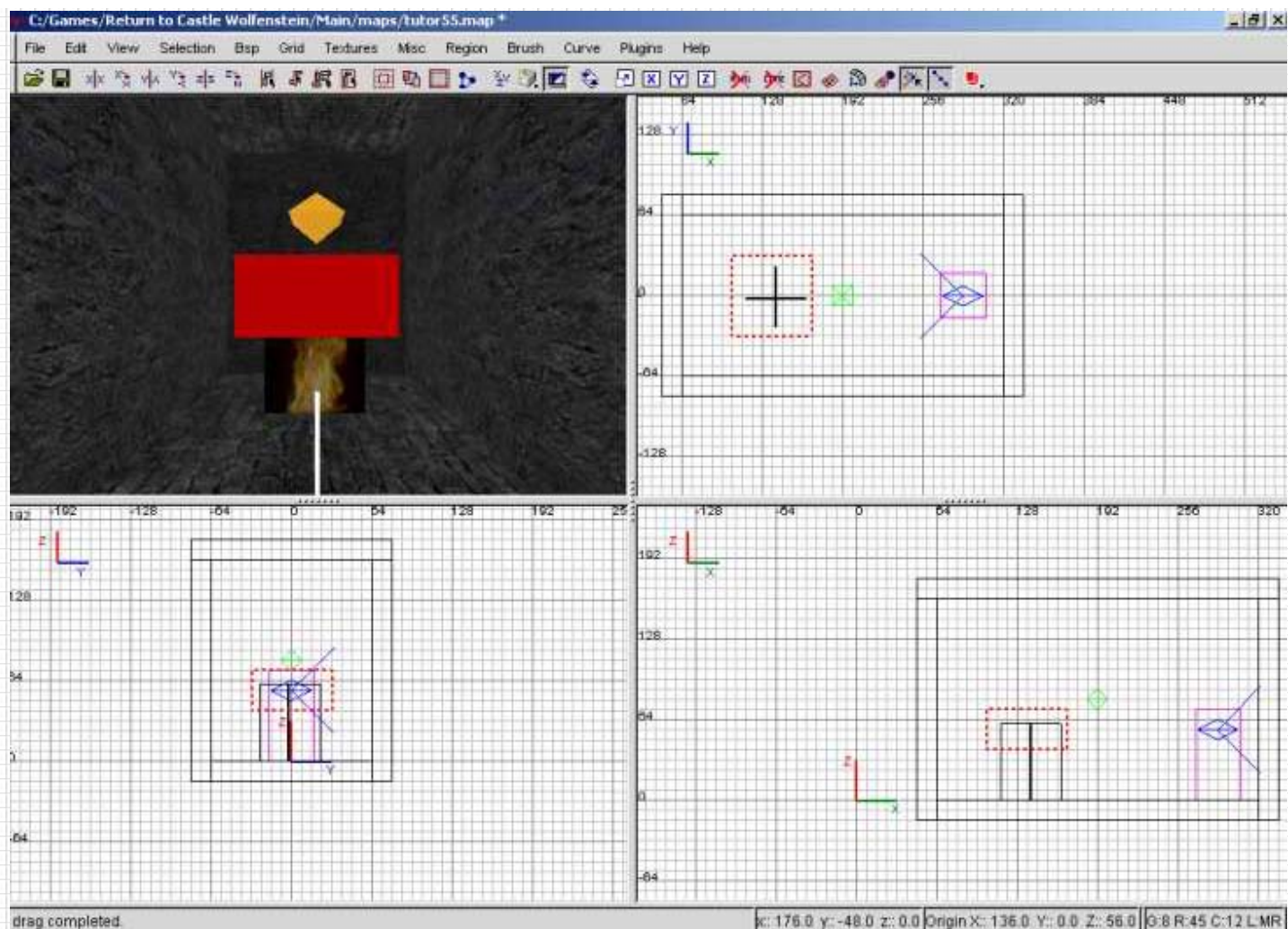
Rauch erstellen:

verwendete Beispielmap: "tutor54.map"
Ergebnismap: "tutor55.map"

Sicher erinnerst du dich an das Thema, in dem ich dir gezeigt habe, wie man ein Feuer macht. Nun will ich dir zeigen, wie man Rauch hinzufügt. Dieser Effekt funktioniert ab der Version 1.33, da das Entity, das für den Rauch zuständig ist, erst in dieser Version in den MP-Modus übernommen wurde: Nun klickst du 2 x mit der rechten Maustaste und wählst "target" und als Unterpunkt "target_smoke":

lightJunior	target_laser
misc	➤ target_location
path	➤ target_lock
props	➤ target_print
screen	➤ target_push
script	➤ target_relay
shooter	➤ target_remove_powerups
sniper	➤ target_rumble
target	➤ target_score
team	➤ target_script_trigger
trigger	➤ target_smoke
weapon	➤ target_speaker
worldspawn	target_teleporter

Nun erscheint ein ziemlich grosser, roter Kasten. Diesen schieben wir an die Oberkante unseres Feuers:



Nun drückst du die Taste "N" um das Entity-Fenster zu öffnen. Nun stehen dir folgende Funktionen zur Auswahl:

- "Black" -> schwarzer Rauch
- "White" -> weisser Rauch
- "SmokeOn" -> Rauch wird von Spielbeginn an dargestellt
- "Gravity" -> die Schwerkraft des Rauchs

Dann gibt es noch folgende Keys:

- Key: "delay" -> Pause, in der kein neuer Rauch erstellt wird
Value: "100" (100 = eine Millisekunde)
- Key: "time" -> gibt die Zeit an, bis der Rauch verschwindet
Value "500" (500 Milisekunden ist der Standartwert)
- Key: "duration" ->Wert, der angibt, ab wann der Rauch transparent wird
Value: "2000" (2000 Milisekunden ist der Standartwert)
- Key: "start_size" -> Grösse der Rauchwolke am Anfang
Value: "24" (24 Units ist der Standartwert)
- Key: "end_size" ->Grösse der Rauchwolke am Ende
Value: "96" (96 Units ist der Standartwert)

[zurück zur Hauptseite](#)

183759



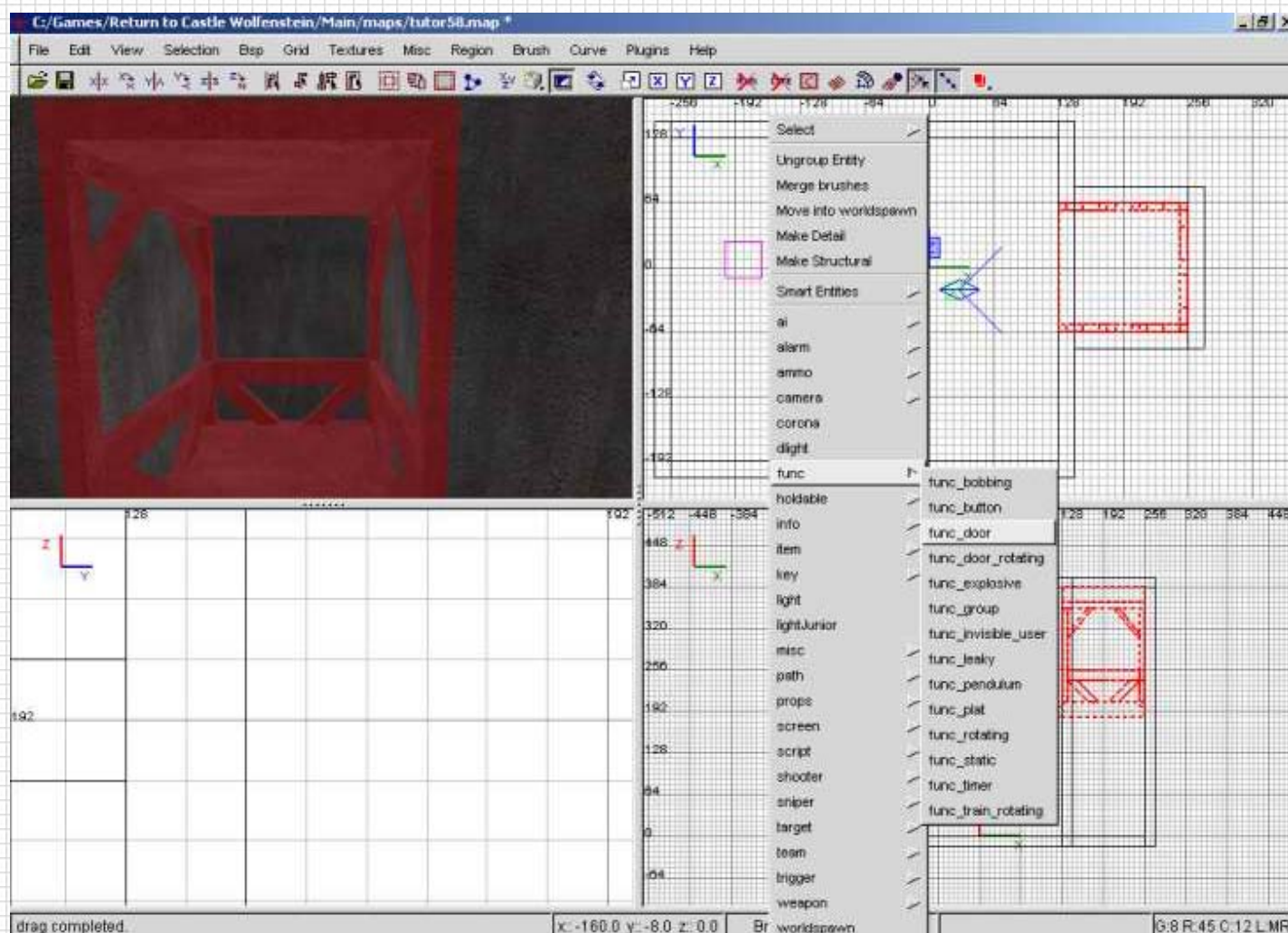


einen Lift einbauen:

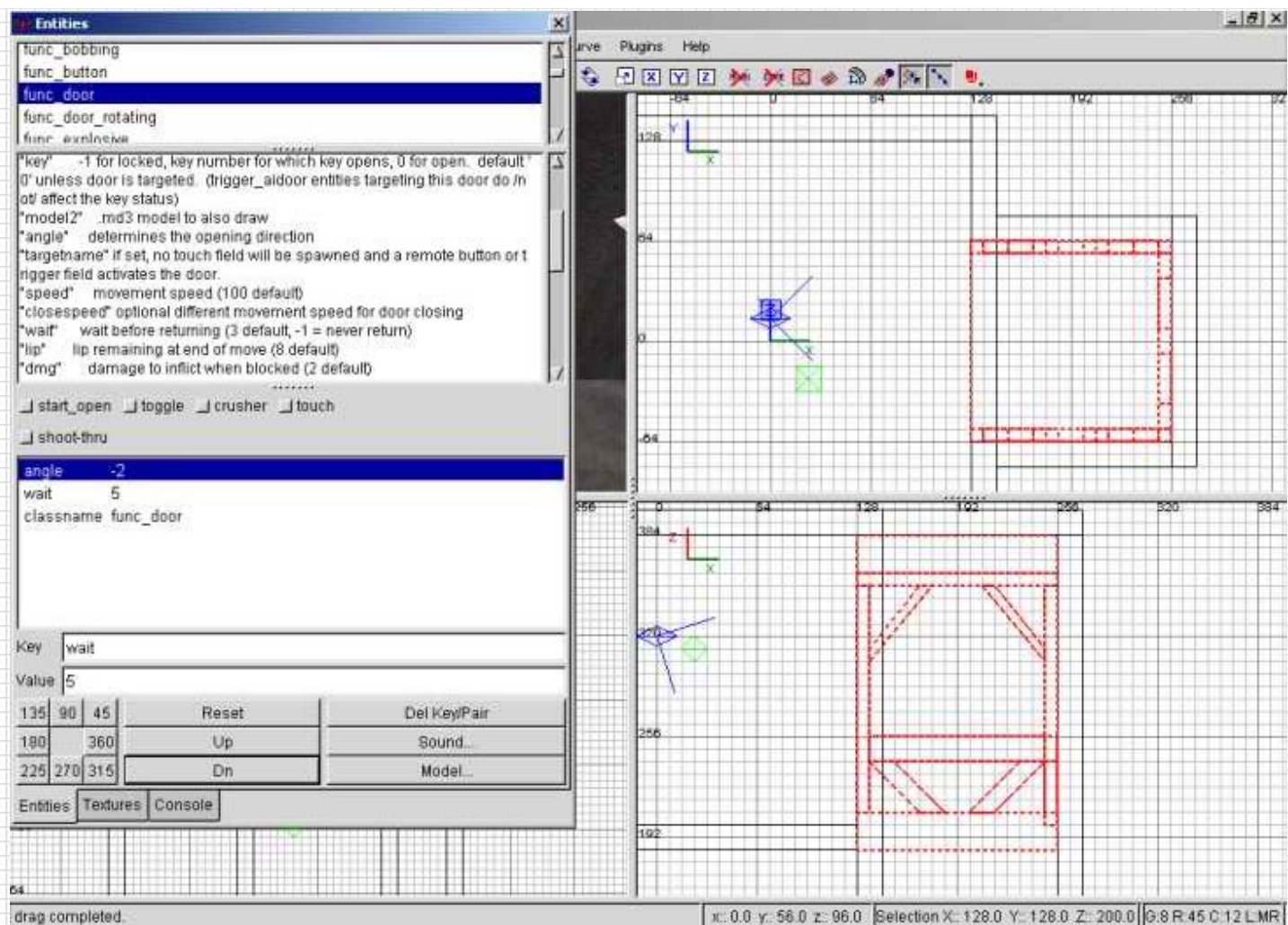
verwendete Beispielmep: "tutor56.map"

Ergebnismep: "tutor57.map"

So, sicher willst du jetzt auch mal eine Map mit mehreren Ebenen machen, die man auch anders erreichen kann, als über Leitern und Treppen. Lifte haben neben ihrem Nutzen (dass sie eben Lifte sind) auch noch etwas Leben in die Map. Also, fangen wir gleich an. Ich habe dir eine Beispielmep erstellt, die 2-stöckig ist und in den Schacht wollen wir einen Lift einbauen. Ich habe dir schon einen Lift hineingebaut. Diesen werden wir gleich selektieren und daraus ein "func_door" machen:



Jetzt wollen wir aber noch ein paar Keys festlegen, dass unser Lift auch weiss, was er später zu tun hat. Dazu drückst du einfach die Taste "N":

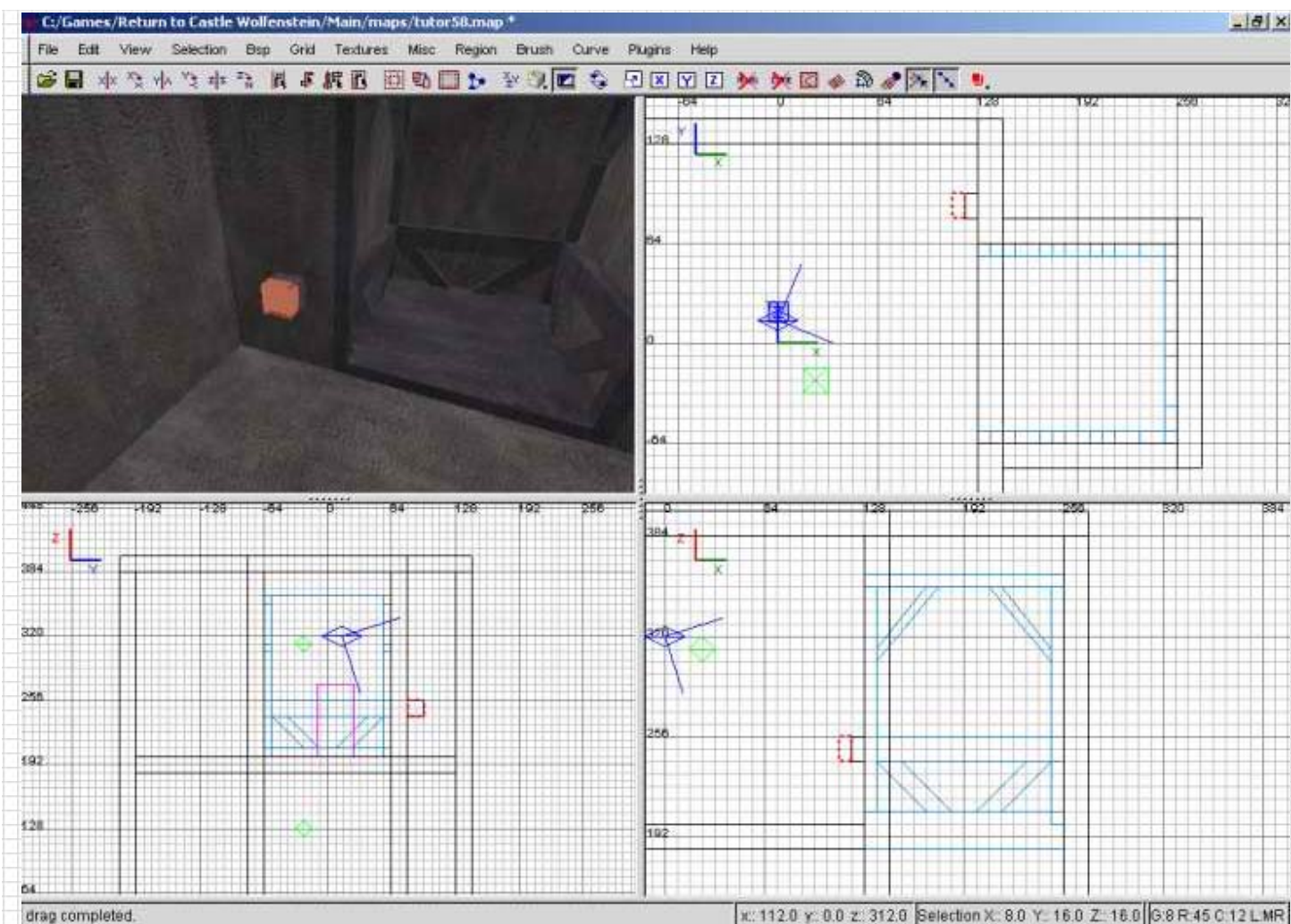


Nun drückst du einfach ganz unten den Knopf, wo "dn" draufsteht, damit sich der Lift nach unten bewegt. Dann gibst du noch ein:

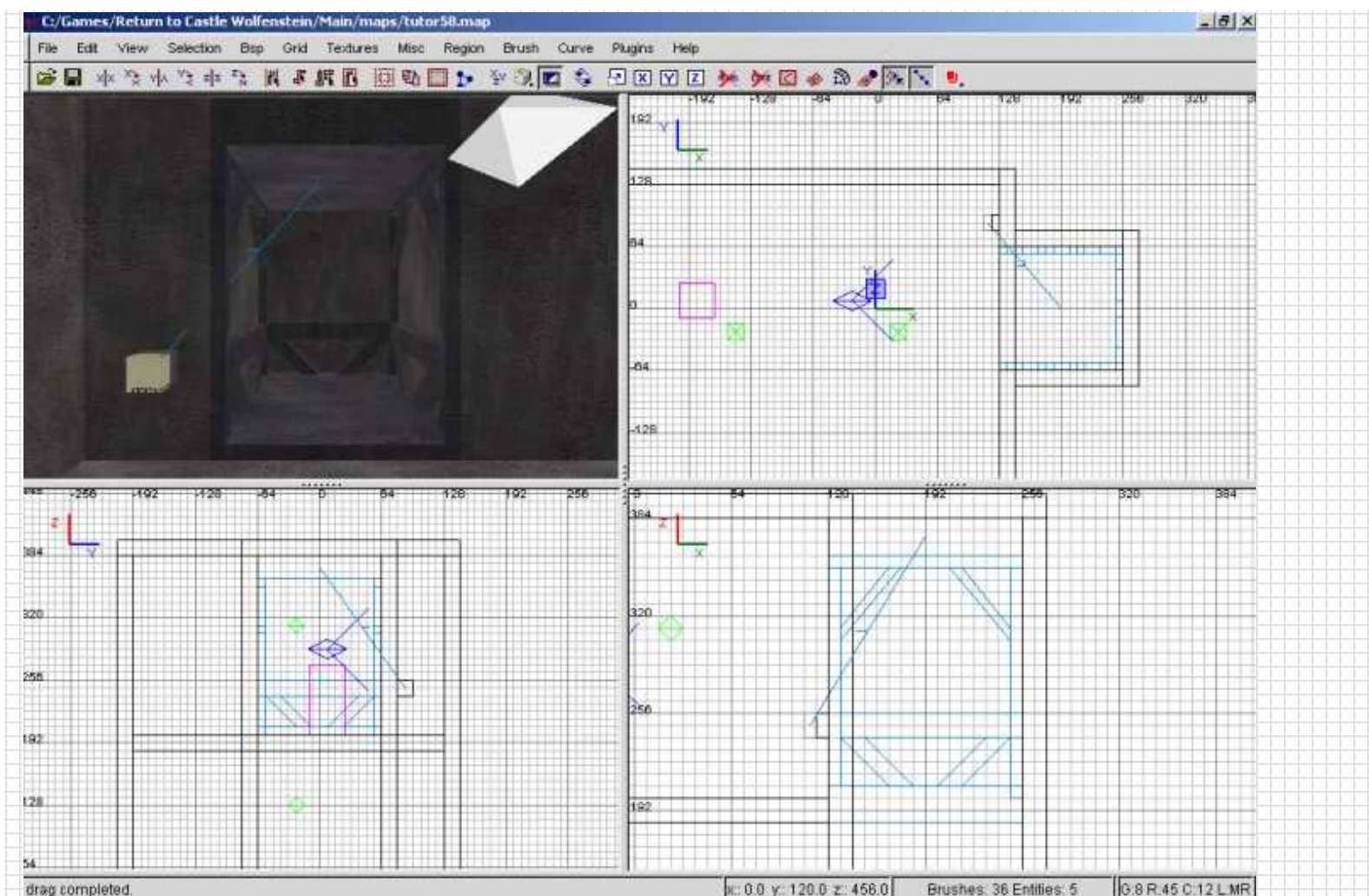
Key: "wait" ->die Zeit, in der der Lift auf einer Ebene stehen bleibt.

Value: "5"

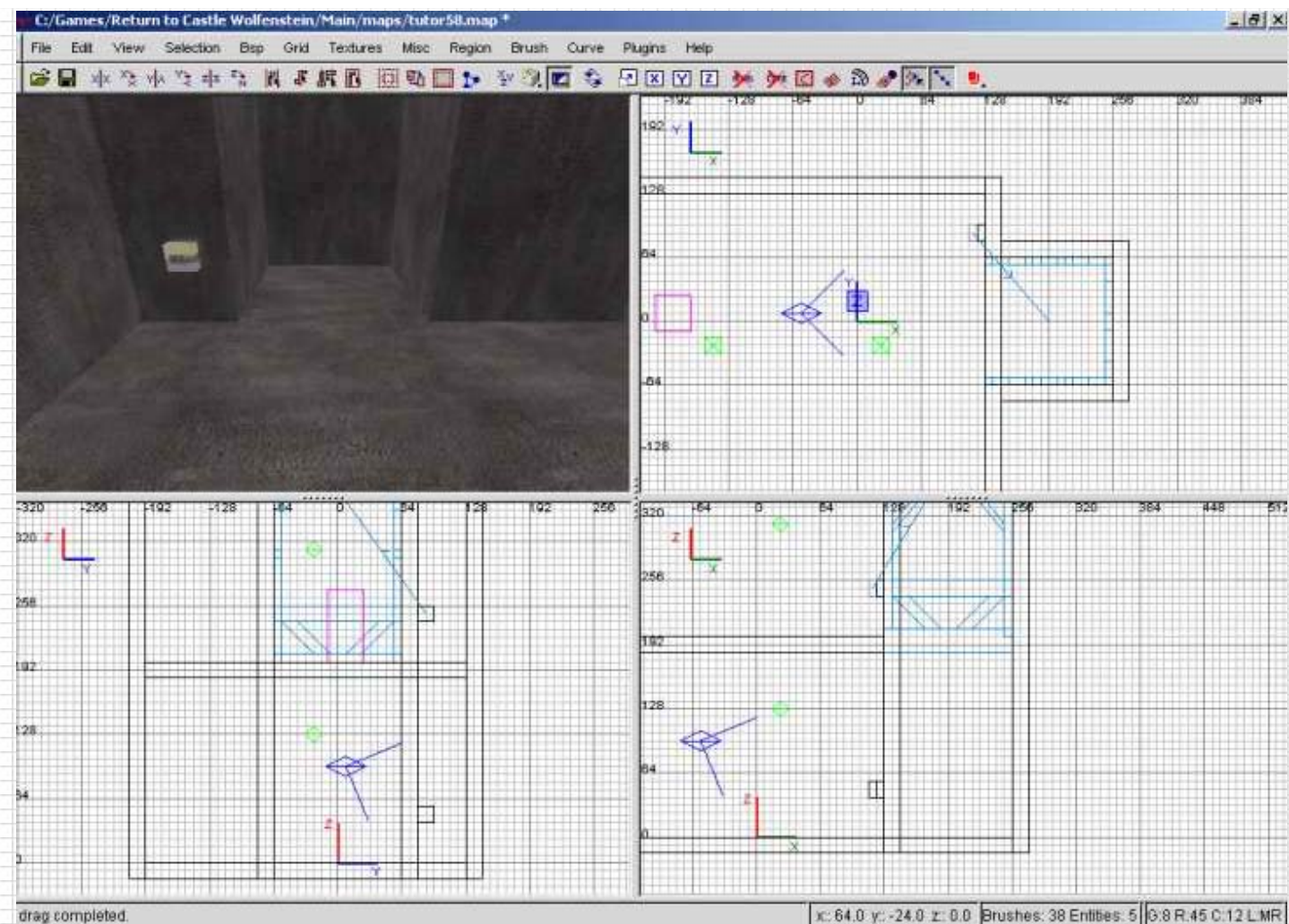
Nun deselektierst du den Lift und der Liftkorb hat nun eine blaue Farbe. Nun wollen wir natürlich noch einen Schalter haben, den man drücken kann, dass sich der Lift bewegt. Dazu erstellst du einfach einen Brush, und hängst ihn an die Wand. Was für ein Brush das ist, spielt keine Rolle, da er nur Verzierung ist. Hast du das gemacht, erstellst du kurz davor einen weiteren Brush, den du mit der Trigger-Textur belegst (diese findest du unter "common/trigger"). Nun klickst du 2 x mit der rechten Maustaste und wählst "trigger" und als Unterpunkt "trigger_multiple":



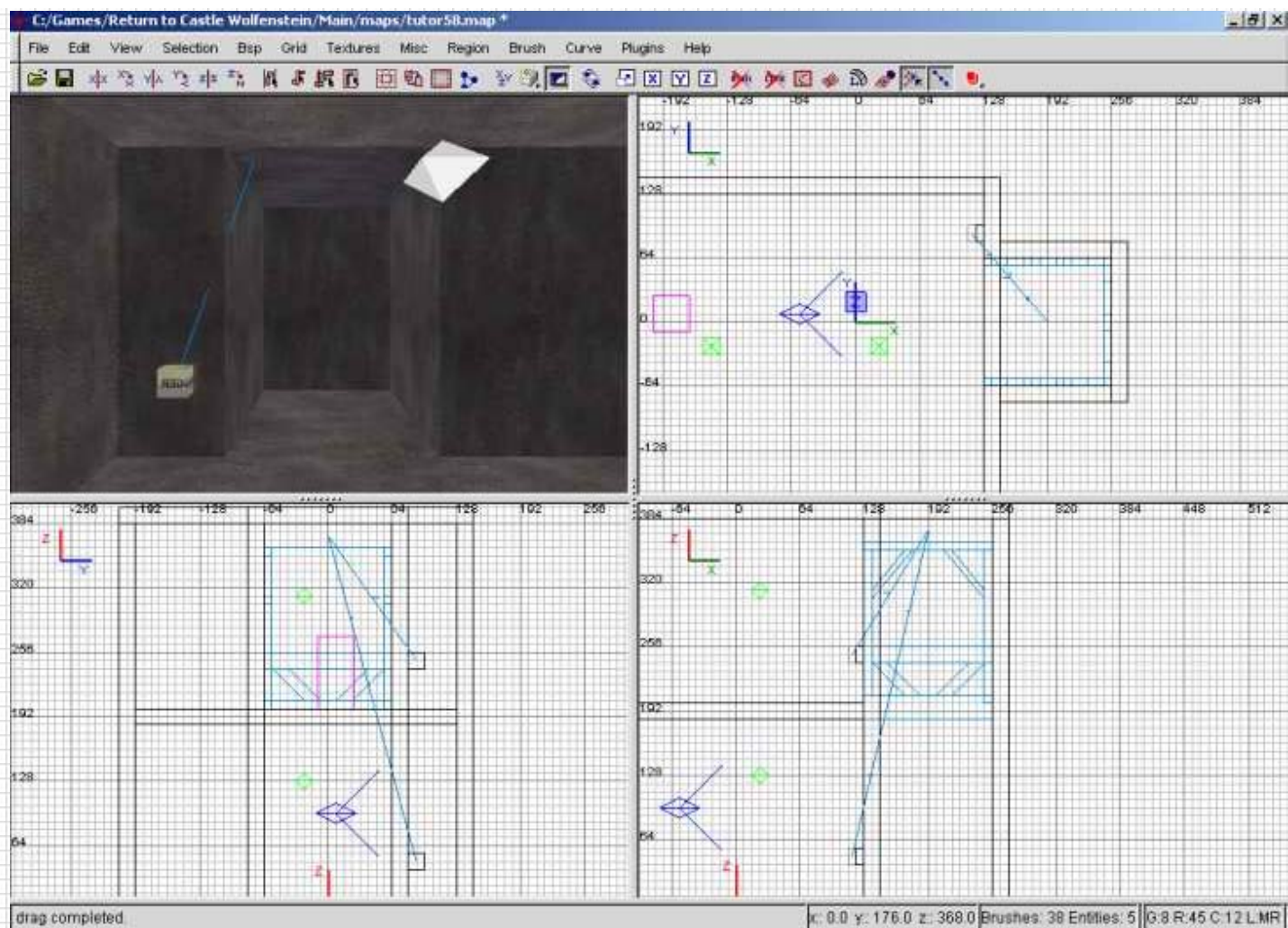
Hier kannst du die beiden Brushes an der Seite gut erkennen. Nun lässt du den Trigger_multiple selektiert und wählst noch den Lift an. Nun drückst du "STRG" und "K" um sie zu verbinden:



So, nun wären wir fast fertig. Nun erstellen wir aber noch im unteren Stockwerk einen "Verschönerungsbrush" und einen Brush mit der Trigger-Textur. Diese beiden Brushes setzt du genauso hoch vom boden entfernt, wie die beiden Brushes im oberen Stockwerk:



Wie du in den Fenstern sehen kannst, sind die trigger genau untereinander und mit dem gleichen Abstand zum Boden. Nun machst du aus dem unteren wieder einen trigger_multiple. Nun drückst du "ESC" und selektierst nur den unteren trigger_multiple, und selektierst noch den Lift. Nun drückst du wieder "STRG" und "K" um den trigger mit dem Lift zu verbinden:



Nun sind wir fertig und du kannst gleichmal deinen Lift ausprobieren.

WICHTIG: Du hast ja die beiden Verschönerungsbrushes und davor die beiden trigger_multiple. Das bedeutet, der Spieler berührt den Brush und der Lift bewegt sich. Dennoch bleibt unser "Verschönerungsbrush" fest wie ein Klotz (ist ja auch einer ;).

[zurück zur Hauptseite](#)

183759



Hintergrundmusik und Namen der Map:

Nun will ich dir zeigen, wie du eine Hintergrundmusik in deine Map machen kannst und wie man die Schwerkraft in der Map ändert. Da Einstellungen für die gesamte Map gelten, gibt es dafür ein extra Entity - das Worldspawn heisst. Nun deselektierst du alles in deiner Map, indem du die "ESC"-Taste drückst. Nun drückst du die Taste "N" um das Entity-Fenster zu öffnen. Hier scrollst du im obersten Fenster nach ganz unten. Dort steht das Entity "Worldspawn":

Entities

weapon_sten
weapon_tesla
weapon_thompson
weapon_venom
worldspawn

DHM - Nerve :: Worldspawn spawnflags to indicate if a gametype is not supported

Every map should have exactly one worldspawn.
"music" Music wav file
"gravity" 800 is default gravity
"message" Text to print during connection process
"ambient" Ambient light value (must use '_color')
"_color" Ambient light color (must be used with 'ambient')
"sun" Shader to use for 'sun' image
"blocksize" q3man always enters the BSP tree along the planes V-blocksize*

☐ no_gt_wolf ☐ no_stopwatch ☐ no_checkpoint

classname worldspawn

Key

Value

135	90	45	Reset	Del Key/Pair
180		360	Up	Sound...
225	270	315	Dn	Model...

Entities

Textures

Console

Wenn du trotzdem einen Brush oder ein Entity selektiert hast, kommt eine Error-meldung im Radiant: "Can't create an entity with worldspawn"

Hier erkläre ich dir die Keys und deren Values:

- Key: "gravity" -> die Schwerkraft. 800 ist der Standartwert. Bei einem höheren Wert (z.B. 900 kann der Spieler nichtmehr so hoch springen, bei einem kleineren Wert, z.B. 700 kann der Spieler höher und weiter springen.
Value: "800"
- Key: "message" -> Die Nachricht, die während dem Laden der Map ausgegeben wird
Value: "TestmMap"
- Key: "ambient" -> gibt den Lichtwert an, den deine Map überall hat
Value: "200" ist der Standart-Lichtwert, der überall in deiner Map ist. Macht alle Schatten usw. kaputt
- Key: "_color" -> wird in folgender Form eingegeben: 1 1 1 (rot-grün-blau-Wert)
Value: "0.000000 0.000000 1.000000" -> blaues Licht
- Key: "sun"
Value: hier gibst du den Pfad zum Shader an, der deine Sonne darstellen soll
- Key: "music" -> spielt die Hintergrundmusik ab, die du deiner Map beigelegt hast, Format: 22 Khz, 16 bit, stereo, signed
Value: "music/haradirki.wav" . Das Musik-File muss im Ordner "Music" liegen, da Worldspawn sonst das File nicht finden kann.

VORSICHT: Du solltest hier darauf achten, dass im Worldspawn keine Sonderzeichen vorkommen dürfen. Sonst macht deine Map beim Compilieren den Abgang. Ebenfalls solltest du darauf achten, dass Wav-Dateien besonders gross werden können. Wenn du also nicht auf eine Hintergrundmusik verzichten willst, solltest du versuchen, sie so kurz wie möglich zu halten und durch einen loop zu wiederholen.

[zurück zur Hauptseite](#)





Startbild erstellen:

So, hier will ich dir zeigen, wie man ein Startbild in seine Map einfügen kann - also das Bild, dass beim Laden der Map gezeigt wird. Dazu benötigen wir natürlich zuerst ein Bild. Wenn du lust hast, kannst du natürlich deine Map einfach so compilieren und dann im Spiel davon ein Shot erstellen. Dabei werden aber leider die Waffe des Spielers, der Radar usw. dargestellt. Das wollen wir natürlich nicht.

Diese Parameter werden wir helfen, das HUD, die Waffe und das Fadenkreuz ausblenden:

/cg_draw2d 0	entfernt die Score
/cg_drawstatus 0	entfernt health/ammo
/cg_drawgun 0	entferntdie Waffe
/cg_drawcrosshair 0	entfernt das Fadenkreuz

Mit den gleichen Parametern mit der Ziffer "1" statt "0" kannst du die Waffe, das HUD und das Fadenkreuz wieder einblenden lassen - also, z.B.

/cg_drawgun 1 schaltet die Waffe wieder ein

Deswegen habe ich dir hier mal ein Script erstellt, dass du in deine *.cfg einfügen kannst:

```
bind x "cg_drawgun 0; cg_drawTeamOverlay 0; cg_drawStatus 0; cg_drawcompass "0"; seta  
cg_noVoiceText "1"; clear; wait 5; screenshot; toggle cg_drawgun; toggle cg_drawTeamOverlay; toggle  
cg_drawStatus; toggle drawcompass; toggle cg_noVoiceText"
```

WICHTIG: Du musst aber darauf achten, dass du die Taste "X" noch nicht mit einem anderen Skript belegt hast

Zur Erklärung: Drückst du im Spiel die Taste "X", so wird die Waffe abgeschaltet und das gesamte HUD ausgeschaltet. Anschliessend wird ein Screenshot gemacht und hinterher wird die Waffe und das HUD wieder eingeblendet. Dieses Skript habe ich für mich erstellt, da man so sehr praktisch alles wunderbar steuern kann.

WICHTIG: Du solltest beim Erstellen des Screenshots die Auflösung im Spiel möglichst auf 640 x 480 oder 512 x 384 Pixel einstellen., da sonst das Bild unnötig gross wird. RtCW speichert die Screenshots als *.tga Datei abt. Kannst du dieses Dateiformat nicht editieren, würde ich dir empfehlen, dieses Script zu verwenden, da der Screenshot hier als *jpg Datei abgespeichert wird:

```
bind x "cg_drawgun 0; cg_drawTeamOverlay 0; cg_drawStatus 0; cg_drawcompass "0"; seta  
cg_noVoiceText "1"; clear; wait 5; screenshotJPEG; toggle cg_drawgun; toggle cg_drawTeamOverlay;  
toggle cg_drawStatus; toggle drawcompass; toggle cg_noVoiceText"
```

Startest du deine Map mit dem Befehl "/devmap name-deiner-map", so kannst du in der Console "/noclip" eingeben und so frei in deiner Map herumfliegen und so Screenshots von grösseren Flächen erstellen, die man normal nicht so gut überblicken kann.

Nun benennst du dein Screenshot noch in der-name-deiner-map.jpg ab und kopierst das Bild in den Ordner "Levelshots". Der Screenshot wird nur dann dargestellt, wenn der Name mit deiner Map genau übereinstimmt. Heisst also deine Map z.B. "haradirkimap.bsp", so muss der Screenshot auch "haradirkimap.jpg" heissen.

Wenn du natürlich dein Screenshot schwarz-weiss und mit sonstigen Effekten darstellen willst, benötigst du ein besseres Grafikprgramm wie Photoshop oder Paintshop. Hier gibt auch viele schicke Filter, mit denen du deinen Screenshot ausstatten kannst.

Screenshots zum Vorstellen der Map:

- Natürlich gehst du hier genauso vor, wie oben bereits beschrieben - aber du solltest einige Dinge beachten:
- Wenn du schon Screenshots veröffentlichst, compile die Map mit Licht und mit allem Drum und Dran. Nichts sieht schlimmer aus, als einen Shot mit Vertex-Licht
- Verrate nicht allzuviel von deiner Map, immerhin sollen die Leute ja noch was zu entdecken haben
- Du solltest die Shots verkleinern. Niemand sieht sich 2-Megabyte grosse Bilder von einer Custom-Map an

[zurück zur Hauptseite](#)

183759



eine Skybox erstellen:

Nun, zuerst will ich klären, was eigentlich überhaupt eine Skybox ist und wozu man sie braucht. Eine Skybox ist ein Rund-Um-Bild, z.B. eine Stadt, eine Landschaft oder auch Inseln usw. Eine Skybox besteht aus 6 Seiten - also für die 4 Wände + je ein Bild für den Boden und die Decke. Der Sinn ist auch schnell erklärt - der Spieler soll denken, er sieht eine weit entfernte Landschaft.

Hier will ich dir zeigen, wie man eine komplett eigene Skybox erstellt - also kannst du dir schonmal überlegen, was du überhaupt machen willst - eine Inselkette, eine Berglandschaft usw.

Also gut, fangen wir an. Zuerst brauchen wir die Bilder, was wohl den grössten Aufwand darstellt. Dazu lädst du dir das Programm "TerraGen" herunter. Dieses Programm findest du hier:

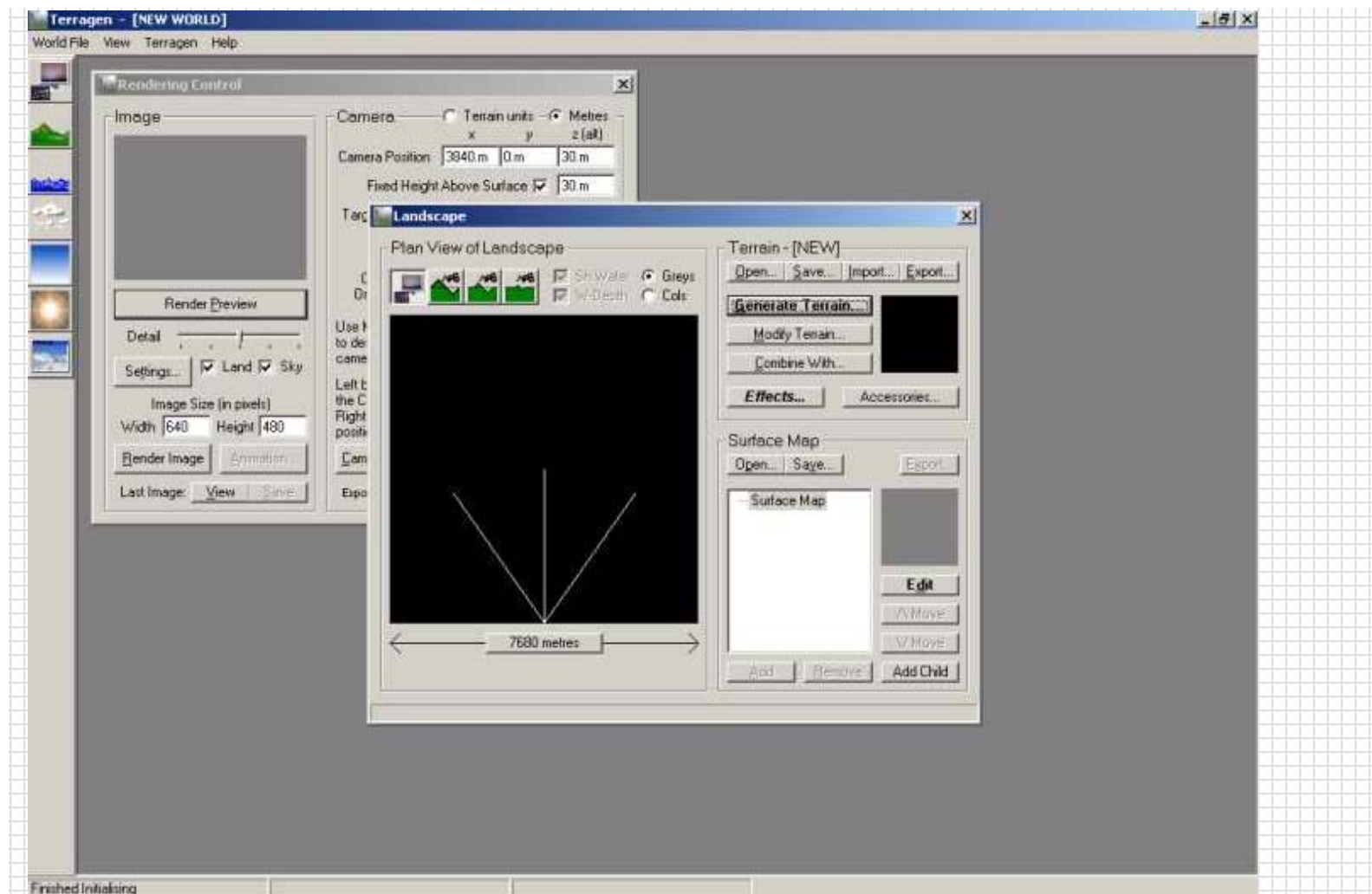
<http://www.planetside.co.uk/terrigen/> (ca 2,54 MB)

Die aktuelle Version ist 0.8.44 und ist momentan für den privaten Gebrauch kostenlos. Nun installierst du das Programm. Da es nichts mit Spielen usw. zu tun hat, brauchst du dir auch keine Gedanken machen, was die Installations-Pfade usw. angeht. Ich habe es ins Verzeichniss "c:\Programme\Terragen" installiert. Nun kopierstt du dir die 2 Dateien "hara1.ter" und "hara1.tgw" aus dem "Terragen-Files"-Ordner in den Ordner von Terragen (bei mir C:\Programme\Terragen). Diese sind in der Offline-Version des Tutorials enthalten. Jedoch kannst du dir die beiden Dateien hier downloaden:

<http://www.haradirki.de/tutor-files/hara1.ter>

<http://www.haradirki.de/tutor-files/hara1.tgw>

Nun wollen wir das Programm gleichmal starten:

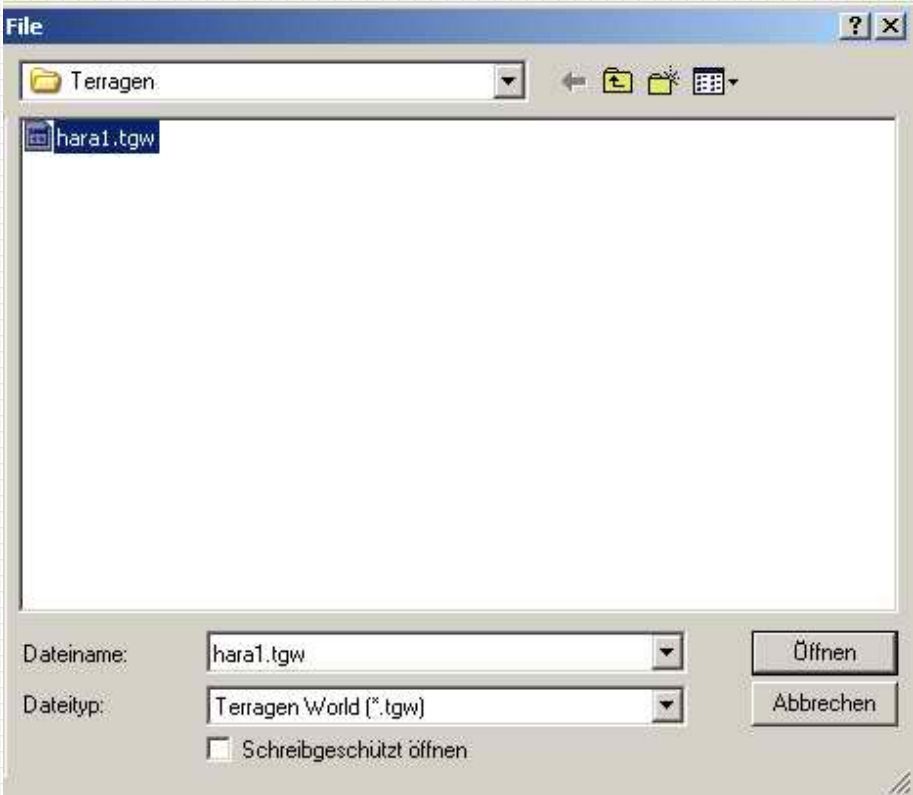


Das hintere Fenster ist das "Rendering-Controll"-Fenster, das vordere das "Landscape"-Fenster. Fehlt bei dir eines dieser beiden Fenster, kannst du sie über die Knöpfe ganz links laden (das mit dem Monitor ist das "Rendering-Control-Fenster", das mit dem grünen Hügel ist das "Landscape"-Fenster).

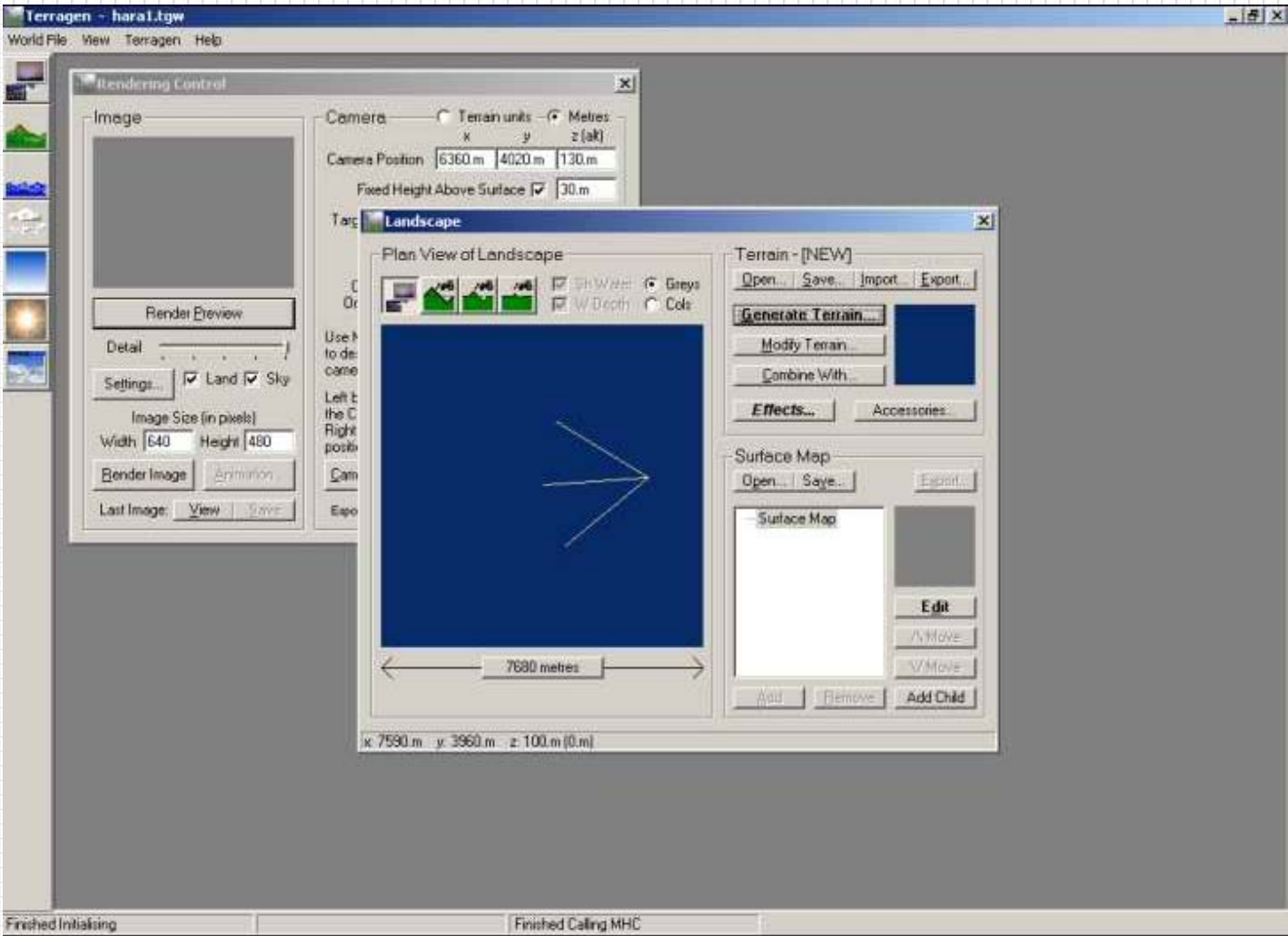
Jetzt klickst du ganz oben im Menü auf den Punkt "WorldFile" und wählst hier "open world". Nun kann die folgende Meldung erscheinen:



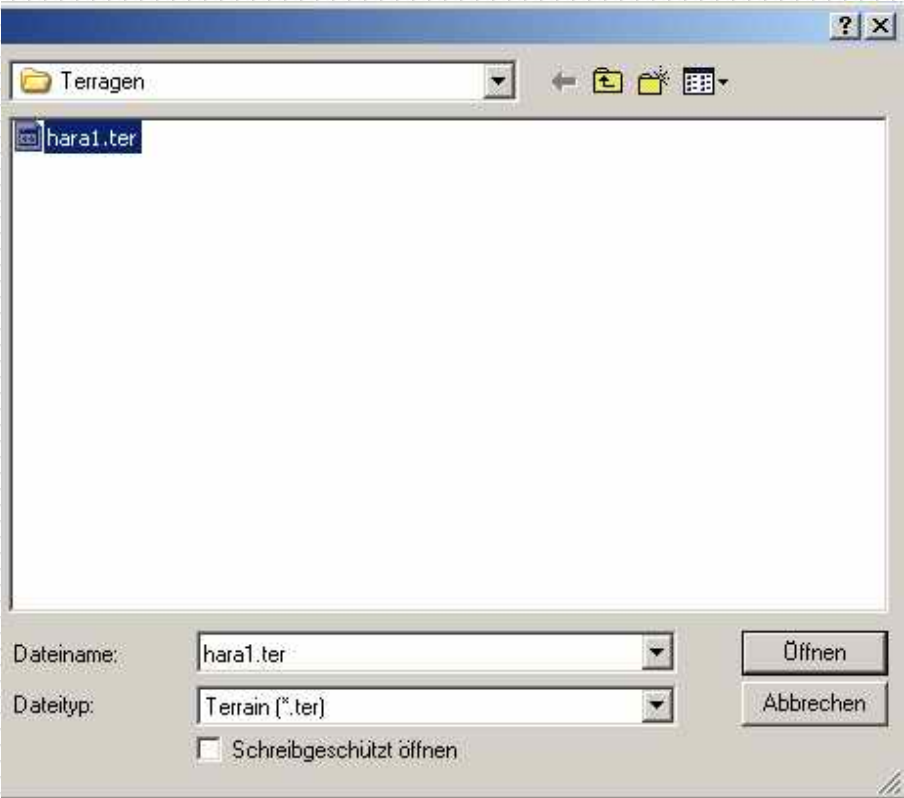
Hier wählst du "Ja". Nun öffnet sich das "Öffnen"-Fenster. Nun öffnest du die Datei "hara1.gtw". Nun klickst du sie an und drückst auf "OK":



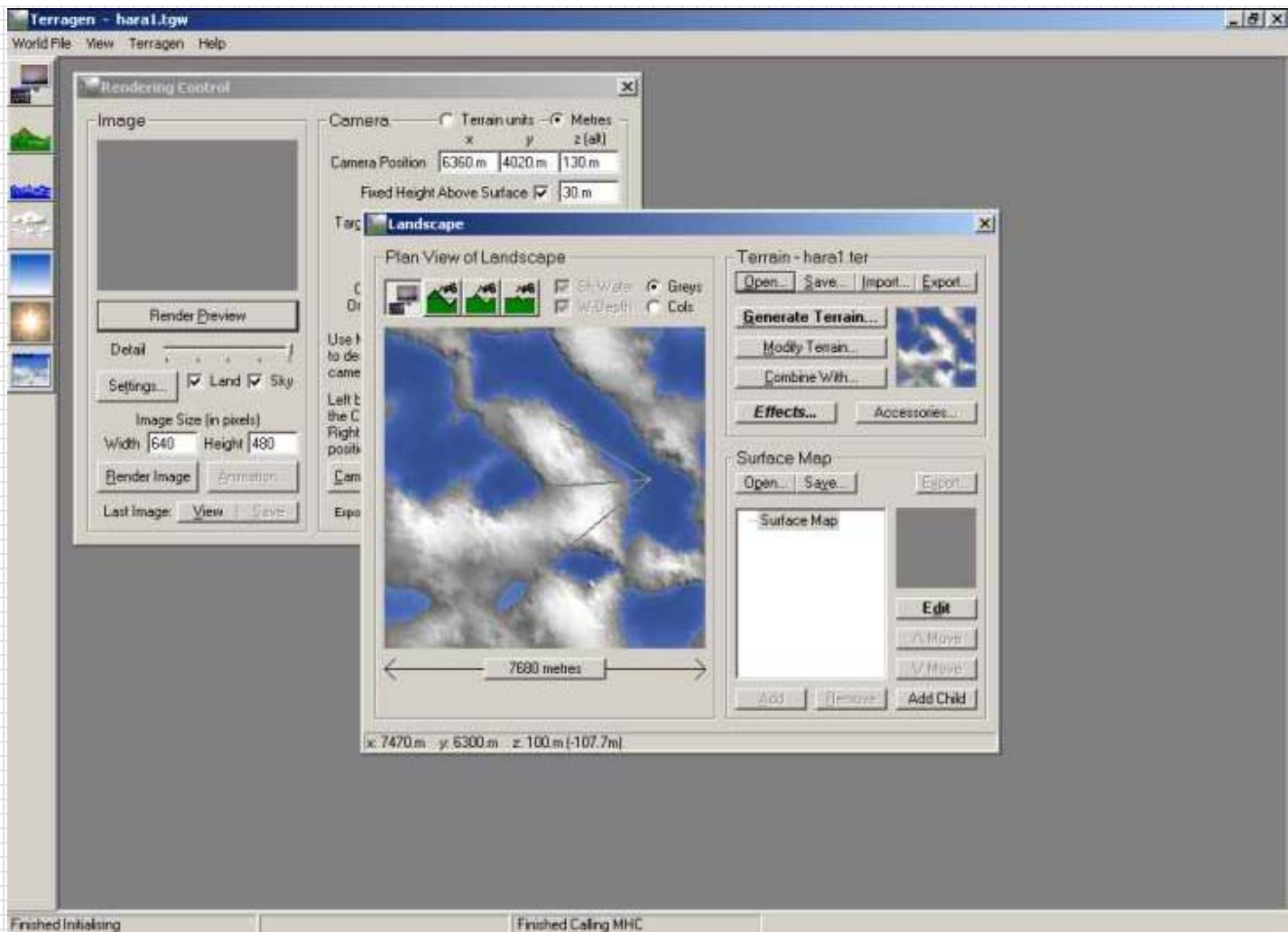
Nun hätten wir schonmal das Laden geschafft :). Nun sollte das kleine Fenster im "Landscape"-Dialog bläulich aussehen:



Jetzt haben wir die Welt-Informationen geladen - jetzt fehlen uns aber noch die Terrain-Informationen. Dazu klickst du im forderen "Landscape"-Fenster bei "Terrain - [NEW]" auf "open..." und öffnest die Datei "hara1.ter":



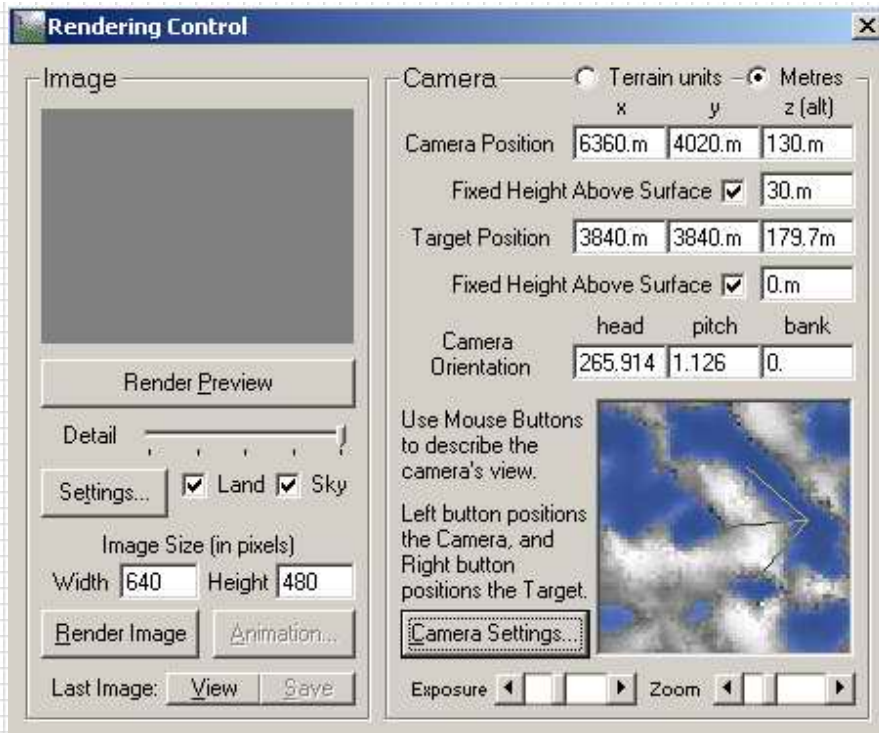
Nun klickst du wieder die Datei an und wählst "OK". Nun kannst du schon unsere Landschaft aus der Vogelperspektive ansehen:



In dem grossen Fenster kannst du ja nun unsere Landschaft aus der Vogelperspektive erkennen. Der Pfeil symbolisiert hier die Kamera, die Spitze ist zeigt an, von welcher Position die Kamera die Bilder ausgibt.

Wenn du diese Sicht ändern willst, kannst du einfach mit der linken Maustaste dorthin klicken, wo du deine Kamera stehen haben willst. Nun kommen wir dazu, unsere 6 Bilder zu erstellen. Dazu habe ich die Kameraeinstellung so belassen, wie der Pfeil beim Laden der Datei steht.

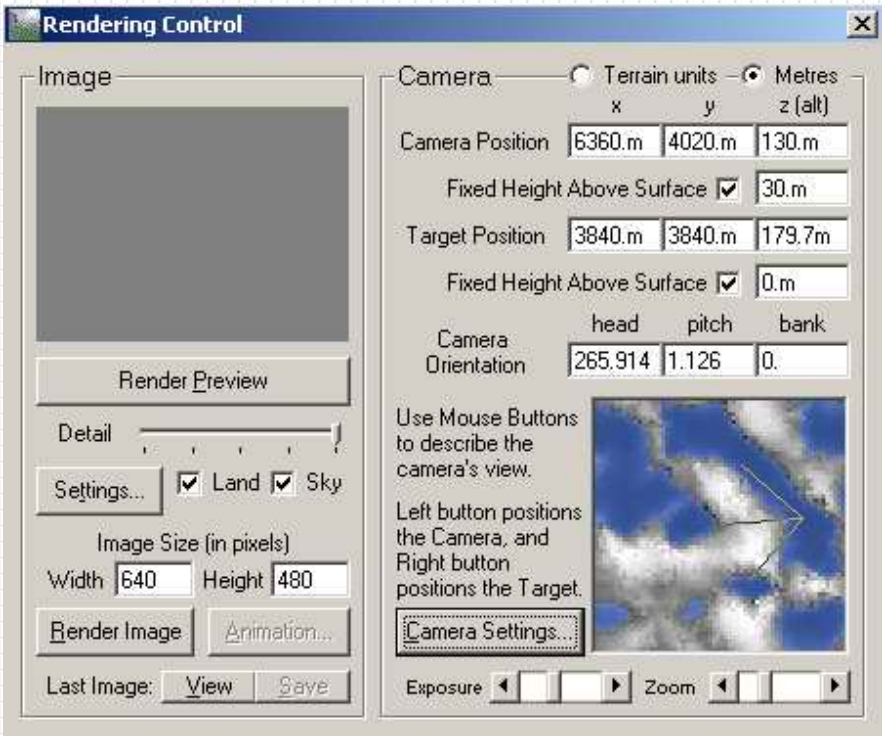
Den Rest des Programmes werden wir uns dem "Rendering-Control"-Fenster widmen - also habe ich dir hier mal eine Großaufnahme des Fensters gemacht. Ganz unten siehst du nochmal die topographische Ansicht unserer Landschaft und daneben den Knopf "Camera Settings":



Diesen Knopf klickst du jetzt an:



Hier trägst du bei "Zoom / Magnification" den Wert "1" ein. Tust du das nicht, werden sich die Bilder später verziehen und passen nichtmehr korrekt dargestellt. Nun klickst du auf "close". Nun erkläre ich dir mal die Einstellungen, die du im "Rendering-Control"-Fenster machen kannst:



"Render Preview" -> klickst du diesen Knopf, so rendert Terragen ein kleines Bild als Vorschau. Wenn du dieses Bild erstellst, solltest du das "Detail" (der Regler untendran) auf die erste Stufe setzen, da sonst das rendern unnötig lange dauert.

"Render Image" -> rendert das fertige Bild

Der Kasten Image Size (in pixels) gibt an, wie gross später unsere Bilder werden. Im Grunde geht hier jedes Format, aber du solltest eines wählen, mit dem der Radiant gut zurecht kommt. Das wären 64x64, 128x128 und. 256x256 Pixel. Für unsere Skybox wählen wir 256x256 Pixel, da es ein guter Kompromiss zwischen Dateigrösse und Qualität darstellt. Jetzt kommen wir zum erstellen unserer 6 Bilder. Eine Skybox besteht, wie oben schon erwähnt, immer aus 6 Bildern:

die Ansicht jeweils einmal von vorne, hinten, links, rechts, oben und unten.

Diese 6 Bilder werden vom gleichen Punkt aus erstellt - lediglich der Kamerablickwinkel ändert sich. Diesen Kamerablickwinkel lässt sich im rechten Teil über die drei Felder "Head", "Pitch" und "Bank" ändern.Für unsere Zwecke reichen allerdings die Punkte "Head" und "Pitch". Beide Werte sollten auf "0" stehen.

Bevor wir allerdings mit den Bildern anfangen, musst du noch das Detail auf die höchste Stufe setzen.

Nun klickst du auf "Render Image". Das Rendern des Bildes kann schon ein bisschen dauern. Wenn der Vorgang abgeschlossen ist, bekommst du das mit einer kleinen Meldung mitgeteilt. Nun kannst du in diesem neuen Fenster, in dem sich auch das gerenderte Bild befindet, das Bild abspeichern. Du gibst dem Bild den Namen "XXX1_rt". Du musst darauf achten, dass das XXX gleich bleibt, und auch die Ziffer danach immer dieselbe ist.

Nun kommen wir zum zweiten Bild. Bevor du dieses Bild renderst, gibst du bei "Head" den Wert "90" an, was 90° bedeutet. Das pitch-Feld lässt du mit dem Wert "0" stehen. Nun klickst du wieder auf "Render Image" - und deine Landschaft wird um 90° gedreht gerendert. Ist das Rendern fertig, speicherst du das Bild unter "XXX1.ft" ab.

	Head-Wert:	Pitch-Wert:	Dateiname:
Bild3:	180	0	XXX1_lf
Bild4:	270	0	XXX1_bk
Bild5:	0	90	XXX1_up
Bild6:	0	-90	XXX1_dn

Diese 4 Bilder renderst du jetzt so, wie ich es dir bei den ersten beiden erklärt habe. Bei den letzten beiden Bildern musst du eben noch darauf achten, dass du den Pitch-Wert änderst und den Head-Wert auf 0 setzt. Für die bessere Übersicht habe ich dir hier nochmal eine komplette Tabelle erstellt, wo du die Werte für alle 6 Bilder nachsehen kannst:

	Head-Wert:	Pitch-Wert:	Dateiname:
--	------------	-------------	------------

Bild1:	0	0	XXX1_rt
Bild2:	90	0	XXX1_ft
Bild3:	180	0	XXX1_lf
Bild4:	270	0	XXX1_bk
Bild5:	0	90	XXX1_up
Bild6:	0	-90	XXX1_dn

So, nachdem wir jetzt unsere 6 Bilder endlich haben, müssen wir sie noch in ein Format bringen, dass der Radiant lesen kann, da sie momentan im BMP-Format vorliegen. Also müssen wir sie nachträglich in das JPG-Format ändern. Nun kopierst du diese 6 Bilder in den Ordner "env". Wenn du diesen Ordner noch nicht in deinem Main-Ordner hast, erstellst du ihn einfach.

Jetzt kommen wir zum Shader, der die 6 Bilder erst zu einer Skybox macht. Dazu lädst du dir die Datei "XXX.shader" in dem Ordner "Shader" in einen Texteditor und siehst ihn dir an:

```
textures/h4ra/XXXskybox
{
    qer_editorimage textures/assault/awall_m04

    surfaceparm noimpact
    surfaceparm nolightmap
    surfaceparm sky

    q3map_sun 0.266383 0.274632 0.358662 100 50 55
    q3map_surfacelight 80

    skyparms env/berg1 - -
}
```

Nun erkläre ich dir mal, was die folgenden Zeilen hier bedeuten:

textures/h4ra/XXXskybox

Hier gibst du den Namen für deinen neuen Shader an. Dieser Name erscheint dann später im Radiant.

qer_editorimage textures/assault/awall_m04

Dies ist das Bild, dass die Skybox im Radiant repräsentiert. Ich habe hier eine Standart-Textur angegeben. Natürlich kannst du auch ein anderes Bild benutzen.

q3map_sun 0.266383 0.274632 0.358662 100 50 55

Gibt den Wert an, welche Farbe der Schein der Sonne in deiner Map hat.

q3map_surfacelight 80

Dieser Wert gibt die Helligkeit an, die von der Oberfläche der Skybox ausgestrahlt wird

skyparms env/XXX1 - -

Hier gibst du den Namen von deinen 6 Bildern ein. Wichtig ist, dass ihr NUR den Teil VOR dem Unterstrich in diese Stelle eintragt, also in unserem Fall XXX1 - -. Würde deine 6 Bilder weltraum1_rt usw. heissen, so müsstest du hier weltraum1 eintragen.

Nun speicherst du diesen Shader unter dem Namen XXX.shader ab. Nun kopierst du diese Datei in den Ordner "main/scripts". Hier findest du auch eine Datei, die "shaderlist.txt" heisst. Diese öffnest du jetzt und fügst am Ende der Datei noch eine neue Zeile ein, in die du "XXXX" schreibst. Nun wird unser Shader beim Öffnen des Radianten geladen.





nützliche Consolen-Kommandos:

Du hast ja schon in den vergangenen Themen gelesen, dass es einige nützliche Consolen Kommandos gibt. Dies ist eine Liste, die einige gute und nützliche Kommandos beinhaltet und dir beim Mappen helfen sollen:

cg_draw2d 0	Schaltet das HUD aus (mit dem Parameter "0") und an (mit dem Parameter "1")
cg_drawFPS 0	Schaltet die Bilder-Pro-Sekunde-Anzeige aus (über "0") und an (über "1")
cg_drawgun 0	Schaltet die Waffe aus (über "0") und an (über "1")
noclip	Schaltet die Kollisionskontrolle ab.
r_clear 1	Benutzt du Befehle wie "r_nocurves", "r_lockpvs" und "noclip" und verlässt die Map, so wird hiermit der HOM-Effekt ausgeschaltet.
r_lockpvs 1	Dieser Parameter befiehlt der Engine, nur diejenigen Bereiche zu rendern, welche in dem Moment sichtbar sind, wenn du diesen Befehl eintippst.
r_nocurves 0	Schaltet die Curves im Spiel ab (über "0") und an (über "1"). Dieser Befehl ist nützlich um zu sehen, ob die Curves von hinten auch geaulket sind. Um dem HOM-Effekt vorzubeugen, solltest du noch "r_clear" eingeben.
r_showtris 1	Damit kannst du der Engine befehlen, alle Polygone zu zeigen, die gerade gerendert werden. Diese werden dann durch die weissen Linien dargestellt. Mit der "1" kannst du diesen Effekt einschalten, mit der "0" kannst du ihn wieder ausschalten.
r_speeds 1	Dies zeigt dir an, wieviele Polygone im Moment von der Engine gezeigt werden. Die beiden andern Zahlen geben die Zahl der Polygonen an, die Shader einschliessen bzw. ausschliessen.
cg_drawstatus 0	Damit kannst du die Health- und die Ammo-Anzeige ausblenden und über den Parameter "1" wieder einschalten.
cg_drawcrosshair 0	Hiermit kannst du das Fadenkreuz ausblenden und über den Befehl "1" das Fadenkreuz wieder einblenden.

[zurück zur Hauptseite](#)



die Map einbinden:

Nachdem du nun die Map soweit fertig gestellt hast - also die Map und das Startbild erstellt hast - wollen wir nun die Map in das Spiel so einbinden, dass man die Map aus dem Menü starten kann.

Dazu erstellst du im Verzeichnis "Scripts" eine neue Textdatei. Diese Datei speicherst du nun als "Name-Der-Map.arena" ab. Heisst deine Map z.B. "mp_tutormap", so sollte diese Datei nun "mp_tutormap.arena" heissen.

Nun öffnest du diese Datei und schreibst die folgenden Zeilen hinein:

```
{
  map "mp_tutormap"
  longname "mp_Tutormap"
  // type "wolfmp wolfsw wolfcp"
  timelimit 12
  axisRespawnTime 18
  alliedRespawnTime 17
}
```

Du solltest darauf achten, dass du auch die Klammern mitkopierst, da die Datei sonst nicht gelesen werden kann. Jetzt erkläre ich dir noch schnell die einzelnen Parameter:

- map: hier gibst du den Namen deiner Map ein.
- longname: hier kannst du deiner Map einen längeren Namen geben, der auch Leerzeichen beinhaltet.
- Type: Hier gibst du ein, welche Spieltypen deine Map unterstützt. Jeweils brauchst du hierzu dann eine *.script Datei, die dann die Ziele innerhalb dieser Spieltypen festlegt.
- Timelimit: gibt an, wie lang eine Runde dauert.
- axisRespawnTime: gibt an, wie lange es dauert, bis die Axis-Respawnzeit dauert.
- alliedRespawnTime gibt an, wie lange es dauert, bis die Allies-Respawnzeit dauert.

Sollte hier etwas nicht so ganz funktionieren, habe ich dir im Script-Ordner eine solche Arena-Datei erstellt. Diese kannst du dann einfach an deine jeweilige Map anpassen.

[zurück zur Hauptseite](#)





Alphatexturen:

Beispielmap: "tutor58.map"

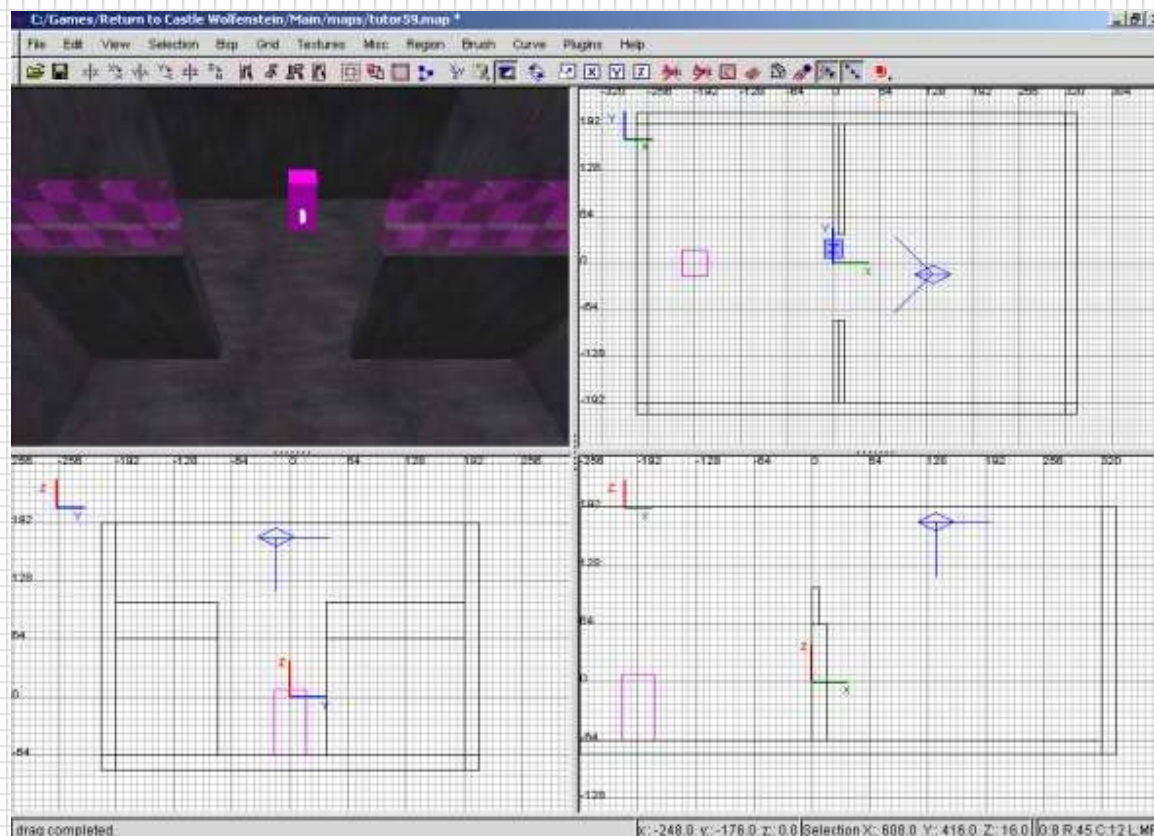
Ergebnismap: "tutor59.map"

Hier will ich dir erklären, was es sich mit den sogenannten "Alpha"-Texturen auf sich hat. Ich habe dir eine kleine Map gebaut, in der sich 2 Mauer-Teile befinden. Hier wollen wir Stacheldraht obendran setzen, dass es nach etwas aussieht. Für den Stacheldraht gibt es natürlich eine passende Alpha-Textur, sie heisst "alpha/fence_m06b".

Doch bevor wir beginnen, will ich dir zunächst erklären, wieso Alpha-Texturen überhaupt Alpha-Texturen heissen. Der Name kommt nämlich nicht von ungefähr. Bilder enthalten immer einen Kanal für die Grundfarben Rot, Gelb und Blau. Sie sind für die normale visuelle Darstellung verantwortlich. Hinzukommt bei den Alpha-Texturen der Alpha-Kanal - er ist dafür zuständig, welche Teile der Textur auch wirklich dargestellt werden. Im Alpha-Kanal gibt es nur 2 Farben: Weiss (steht für den Teil der Textur, der dargestellt wird) und schwarz (steht für den Teil der Textur, der nicht dargestellt werden soll). Solche Texturen kann man auch mit relativ wenig Aufwand selbst erstellen, das zeige ich dir jedoch an anderer Stelle.

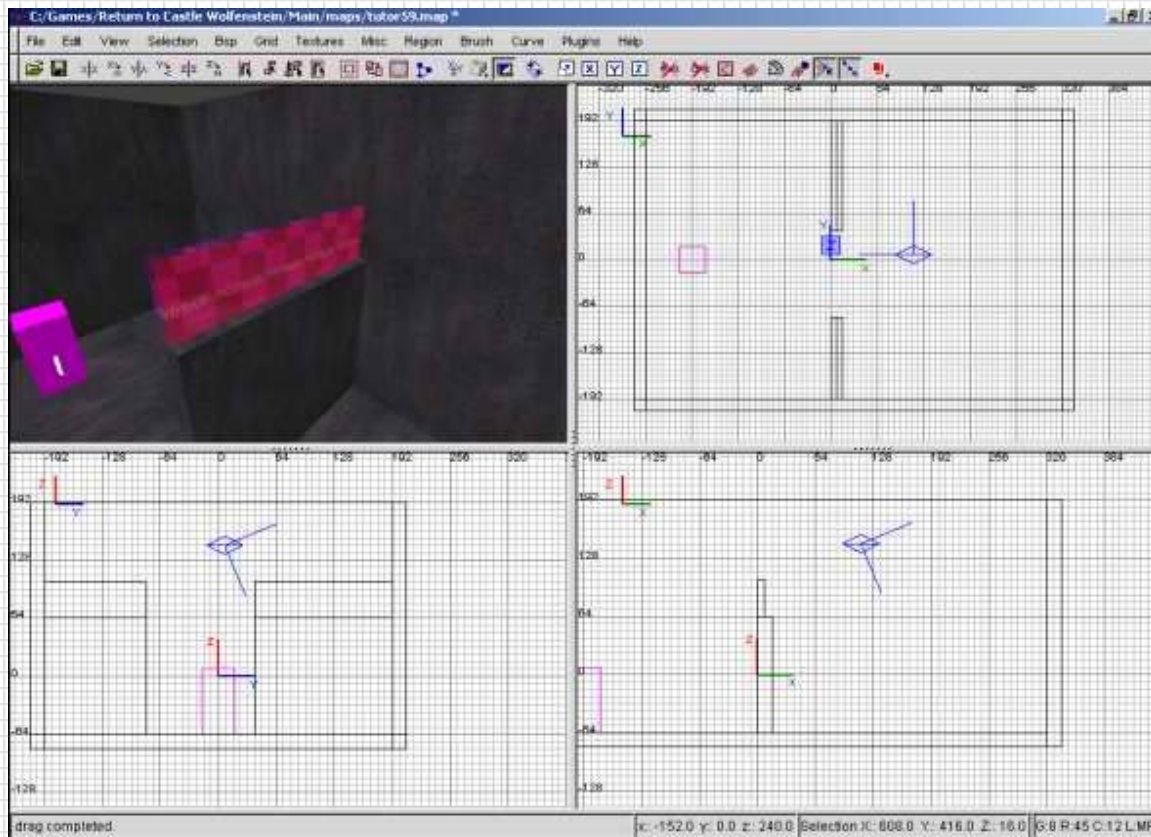
In unserem Fall mit dem Stacheldraht ist es so, dass der Stacheldraht selbst dargestellt wird - aber um die Drähte herum wird dann nichts dargestellt, d.h. hier ist die Textur durchsichtig. Damit diese Darstellung auch im Radiant und später im Spiel auch zu 100 % funktioniert, braucht man natürlich auch wie immer einen Shader, der der Textur bestimmte Eigenschaften verleiht.

Jetzt erstellst du auf den beiden Mauern jeweils ein Brush, der 40 Units hoch ist und 8 Units breit ist. Diesen belegst du mit der "Nodraw"-Textur (diese findest du unter "common/nodraw"):

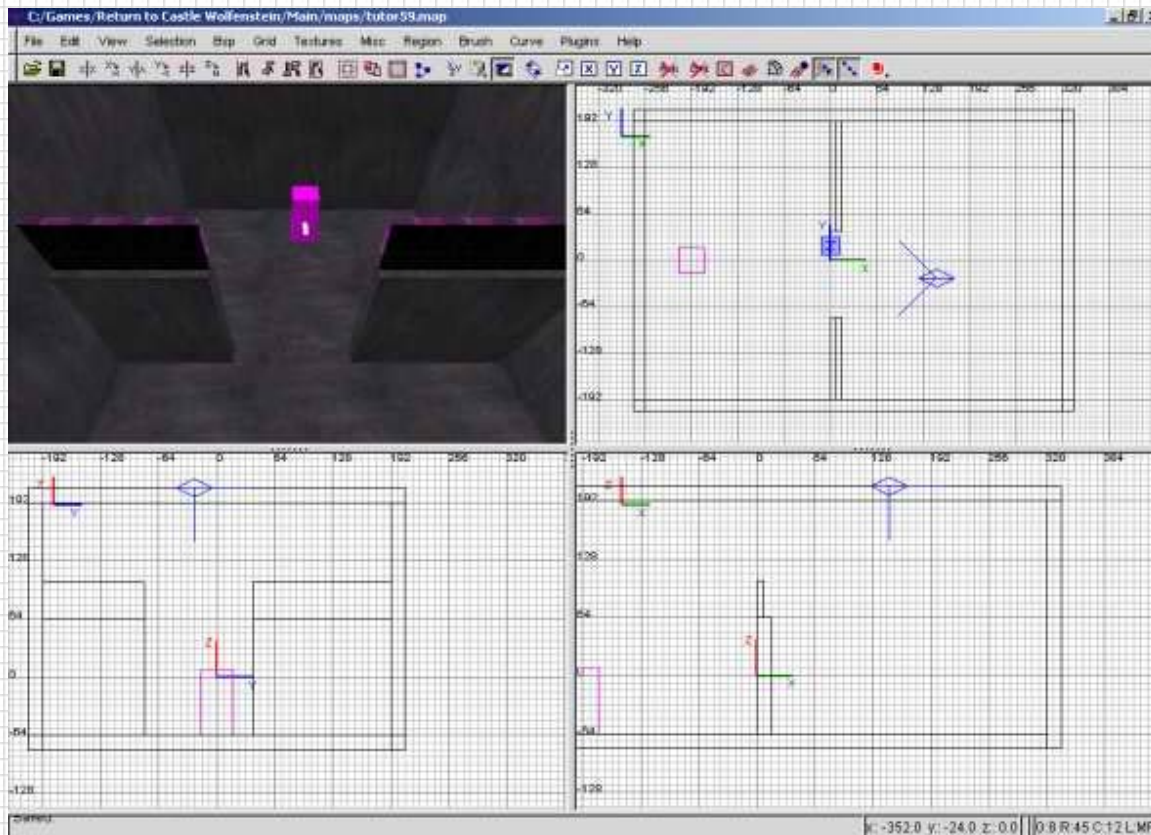


Nun lädst du den gesamten Texturenordner "Alpha". Wie du siehst, haben alle Texturen einen weissen Rand. Du wählst hier die Textur

"alpha/fence_m06b" aus. Wie du im oberen Bild sehen kannst, ist der Nodraw-Brush nur halb so gross wie der Mauer-Brush. Das ganze hat auch seinen Sinn - wir wollen die Textur später nur auf einer Seite des Nodraw-Brushes setzen, daher benutzen wir natürlich die Seite des Brush, die die Mitte der Mauer bildet. Also selektierst du nun diese eine Seite des Nodraw-Brushes:



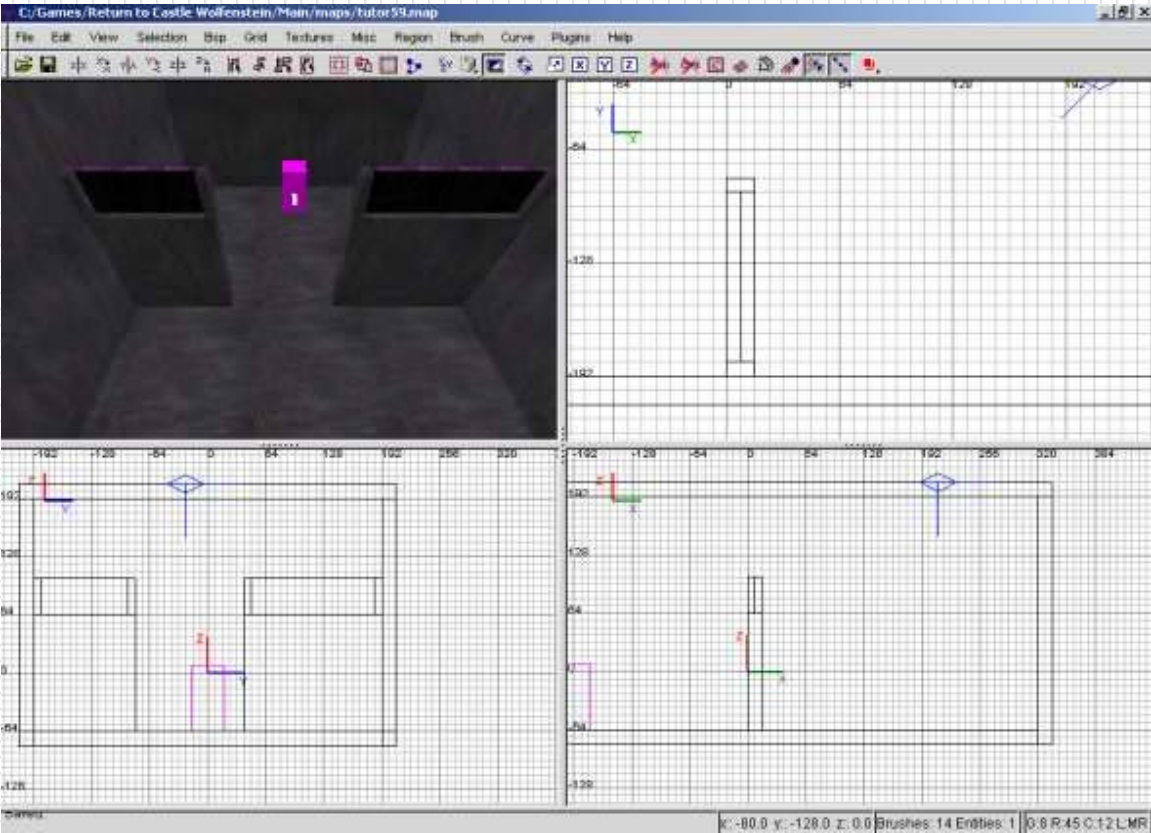
Diese Seite belegst du jetzt mit der Textur "alpha/fence_m06b". Das gleiche machst du jetzt noch mit dem anderen Brush. Dann sieht das ganze so aus:



Wenn du willst, kannst du die Map gleichmal testen und compilieren. Dabei wirst du 2 Dinge feststellen:

- man kann durch den Stacheldraht-Brush hindurchlaufen
- man sieht ihn von der Seite nicht (oder nur wenig)

Diese beiden Punkte lassen sich natürlich umgehen. Zuerst erledigen wir Punkt 2, da er sehr leicht zu erledigen ist - dazu erstellst du einfach 2 Brushes an der Seite des Nodraw-Brushes:



Nun kommen wir zum Problem Nr. 1. Je nachdem, wieviel Realismus du in deiner Map haben willst, kannst du nun wählen, ob der Stacheldrahtzaun "nur" als Sperre dienen soll, oder ob er auch gleich Schaden verteilen soll, wenn sich eine Spielfigur dem Zaun nähert.

- Kommen wir zunächst zum einfachen blocken:

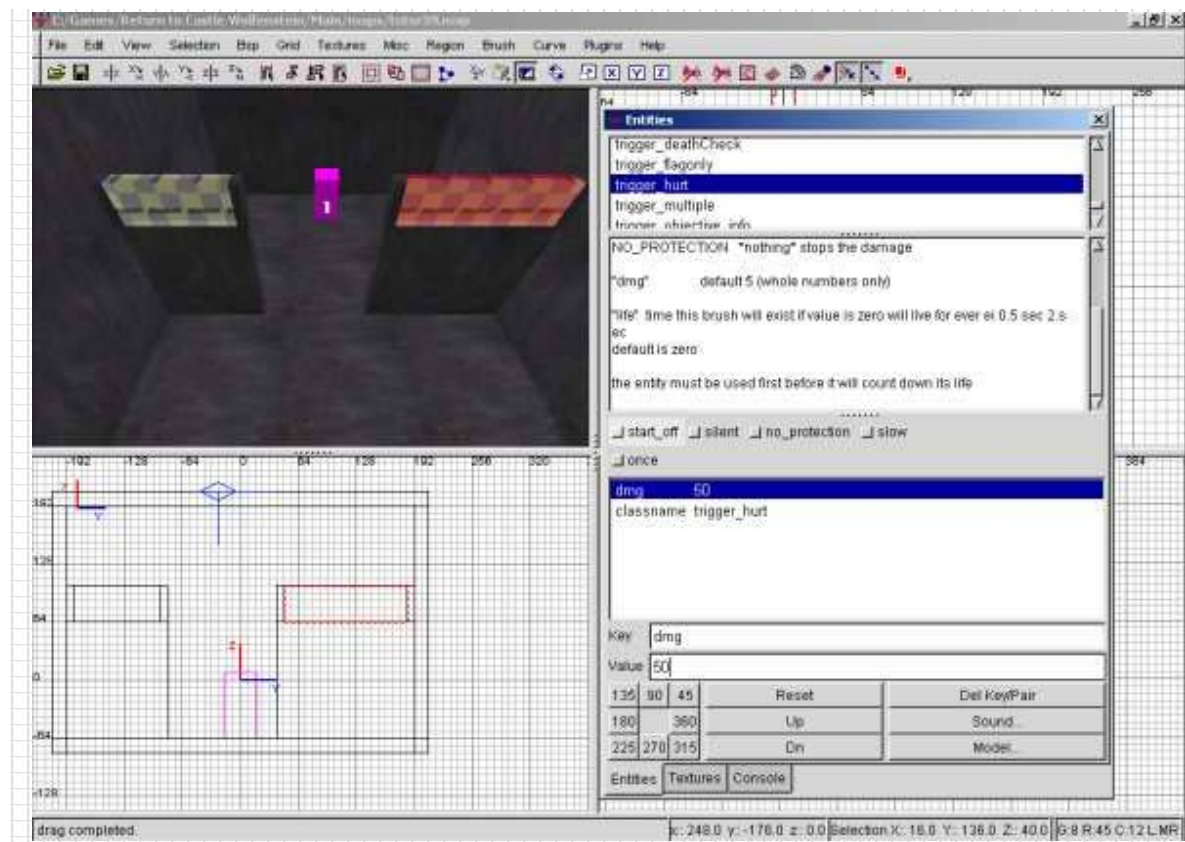
Hier benutzt du einfach statt der "nodraw"-Textur eine der vielen Clip-Texturen. Ebenfalls kannst du natürlich auch den Nodraw-Brush lassen und einfach einen grösseren Brush mit der Clip-Textur um den Zaun herum bauen.

- Nun noch die Erklärung, wie man einen richtigen Stacheldraht (mit Verletzungen) baut:

Hierzu baust du um den Stacheldraht noch einen weiteren Brush, den du mit der Textur "Trigger" (diese findest du unter "common/trigger") belegst. Nun lässt du den Trigger-Brush selektiert und klickst 2 mal mit der rechten Maustaste und wählst hier "Trigger" und als Unterpunkt "trigger_hurt". Nun drückst du die Taste "N" um das Entity-Fenster zu öffnen. Hier gibst du ein:

- Key: "dmg"
Value: "50"

Hierzu mal ein Bild:



Dmg bedeutet "Damage", also schaden. Es kommt darauf an, ob deine Map eine Singleplayer- oder Multiplayermap wird. Während man normal in Singleplayermaps eher eine geringe Value (z.B. 5) benutzt, benutzt man im Multiplayer die Value 50, manchmal sogar mehr. Die Stacheldrahtzäune in Beach haben z.B. eine Value von 50.

[zurück zur Hauptseite](#)

183759

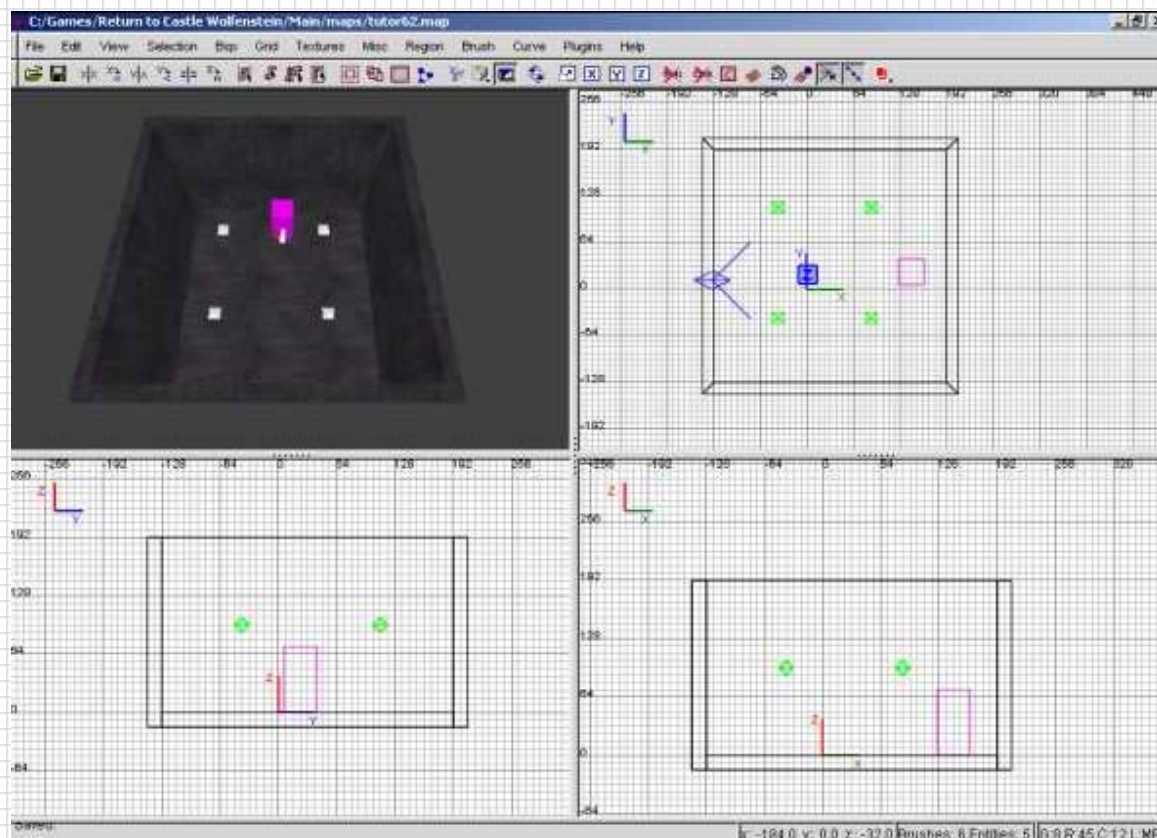


Nebel:

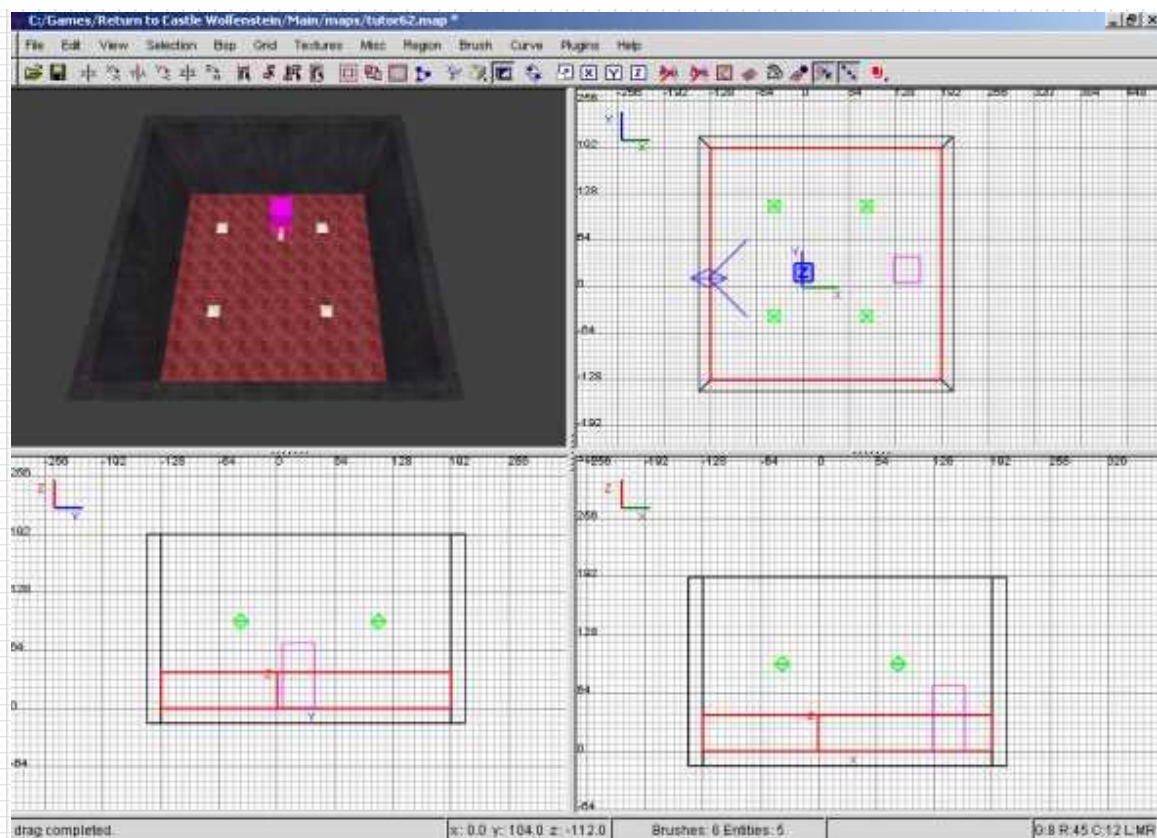
Beispielmap: "tutor62.map"

Ergebnissmap: "tutor63.map"

Nun will ich dir zeigen, wie man Nebel in seine Map einbaut. Das geht sehr leicht. Dazu brauchst du nur einen geeigneten Raum - er sollte nicht allzugross sein, da sonst die Performance sinkt. Zur besseren Ansicht habe ich mal die Decke per Taste "H" versteckt:



Nun ziehst du am Boden einen Brush, der an alle 4 Wände grenzt und 40 Units hoch ist (die Höhe spielt dabei eigentlich keine Rolle, jedoch wollen wir erstmal nur Bodennebel erstellen und machen ihn also nicht zu hoch). Nun lädst du die "sfx"-Texturen. Dort gibt es verschiedene Fog-Texturen, also Nebel-Texturen. Unseren Brush belegen wir mit der Textur "sfx/fog_crypt":



So, jetzt kannst du die Map gleichmal kompilieren. Dann sieht es ungefähr so aus:



Wenn dir diese Art von Nebel nicht so gut gefällt, kannst du ja mal die anderen Fog-Textures ausprobieren.

WICHTIG: Es gibt einige Radiant-Versionen, bei denen nun der Fehler "Entity 0, Brush X: fog brush has multiple visible sides" erscheint - obwohl du alles richtig gemacht hast. Dann kann es sein, dass der Radiant trotzdem die *.bsp Datei ausgespuckt hat. Ich bin auch schon selbst über diesen Fehler gestolpert.

Normalerweise erscheint der Fehler dann, wenn der Nebel-Bruhs von mehr als nur von einer Seite aus sichtbar ist.

[zurück zur Hauptseite](#)

183759



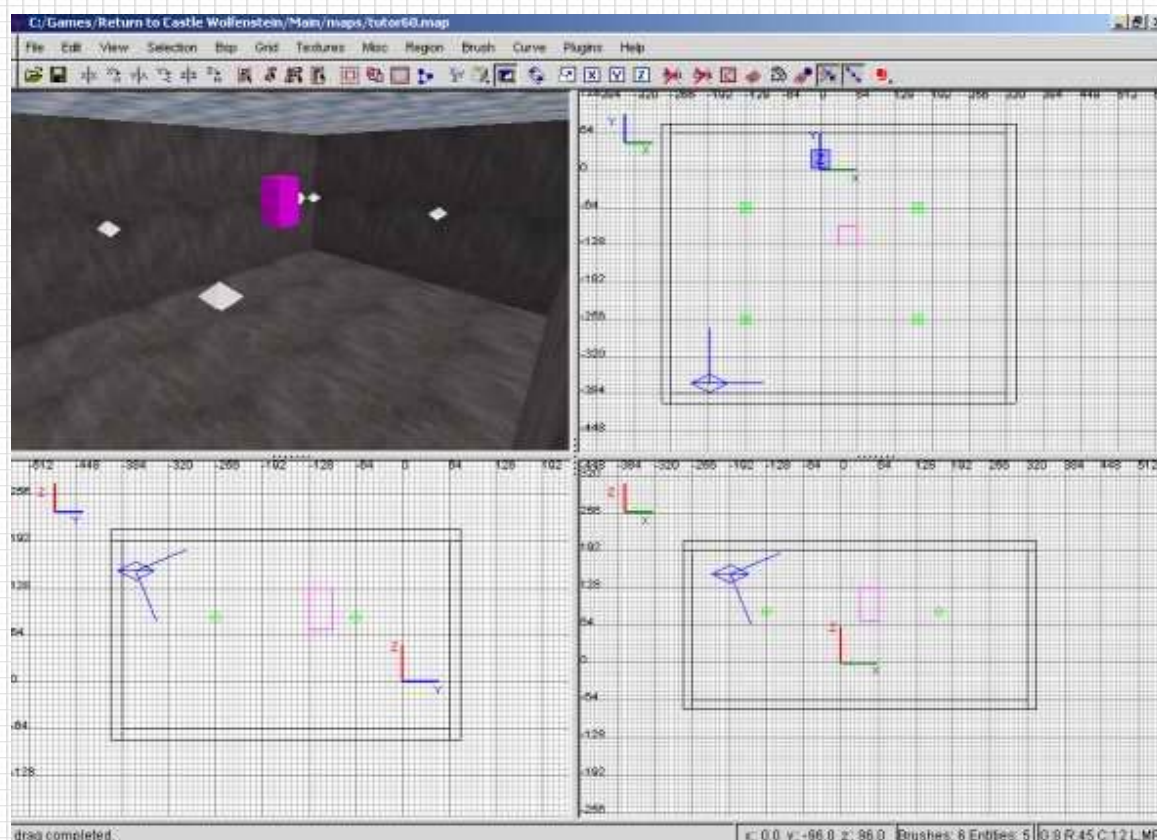
Team - Startpunkte:

Beispielmap: "tutor60.map"

Ergebnismap: "tutor61.map"

Nun hast du ja schon sehr viel über das Mappen gelernt, besonders auch für RtCW. Nun hast du sicher schon überlegt, wie man wohl die spezifischen Startpunkte für Allies bzw. die Axis setzt. Im Grunde geht das ganz einfach.

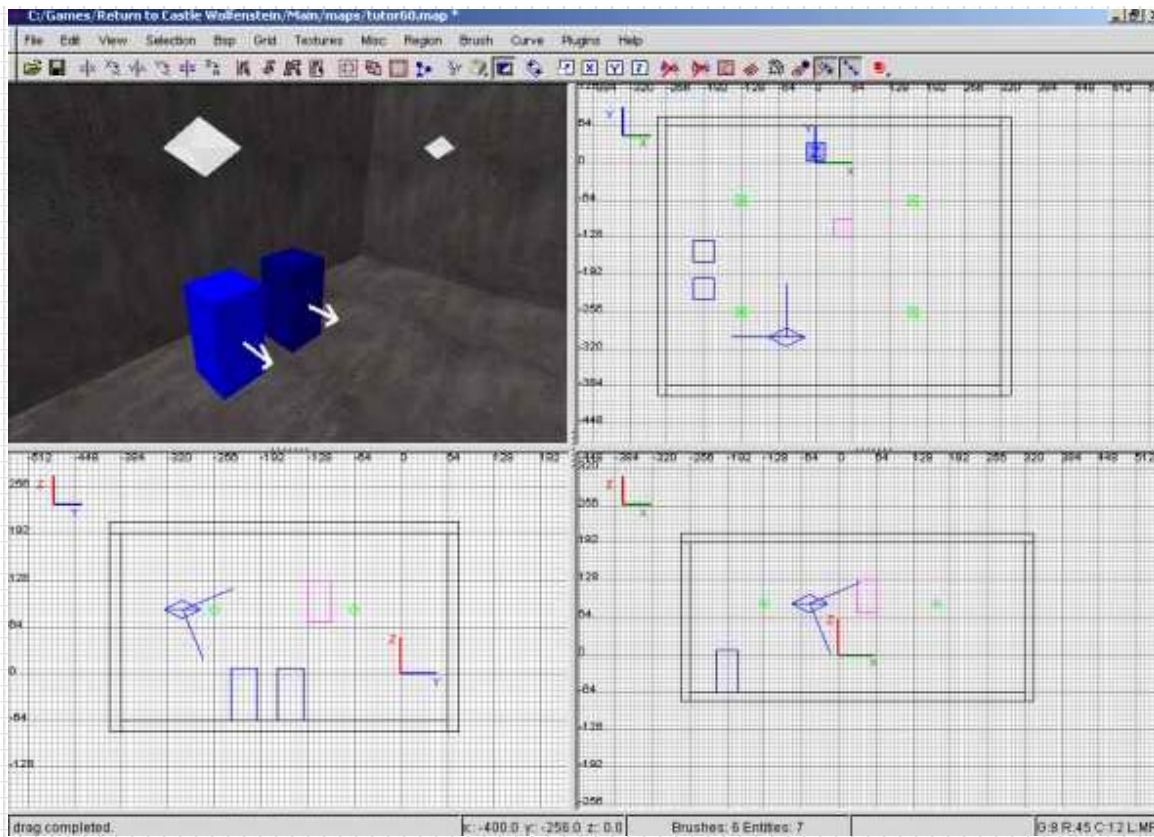
Zunächst einmal startest du die Map, die ich diesem Tutorial beigelegt habe:



Diesen Raum habe ich dir schon etwas präpariert. Ich habe z.B. einen "info_player_intermission" eingebaut. Wozu das? Das hat den Sinn, dass das Spiel die Map ohne einen "info-Startpunkt" nicht startet. Würdest du aber z.B. einen "info_player_deathmatch" setzen, würden auch dort Spieler starten - und unsere Team-Aufteilung wäre sinnlos.

Nun legen wir aber los. Nun klickst du 2 mal mit der rechten Maustaste in die Top-Ansicht und wählst dort "Team" und als Unterpunkt "team_CTF_blueplayer". Schon erscheint ein blauer Kasten, der den Startpunkt für die Allies darstellt.

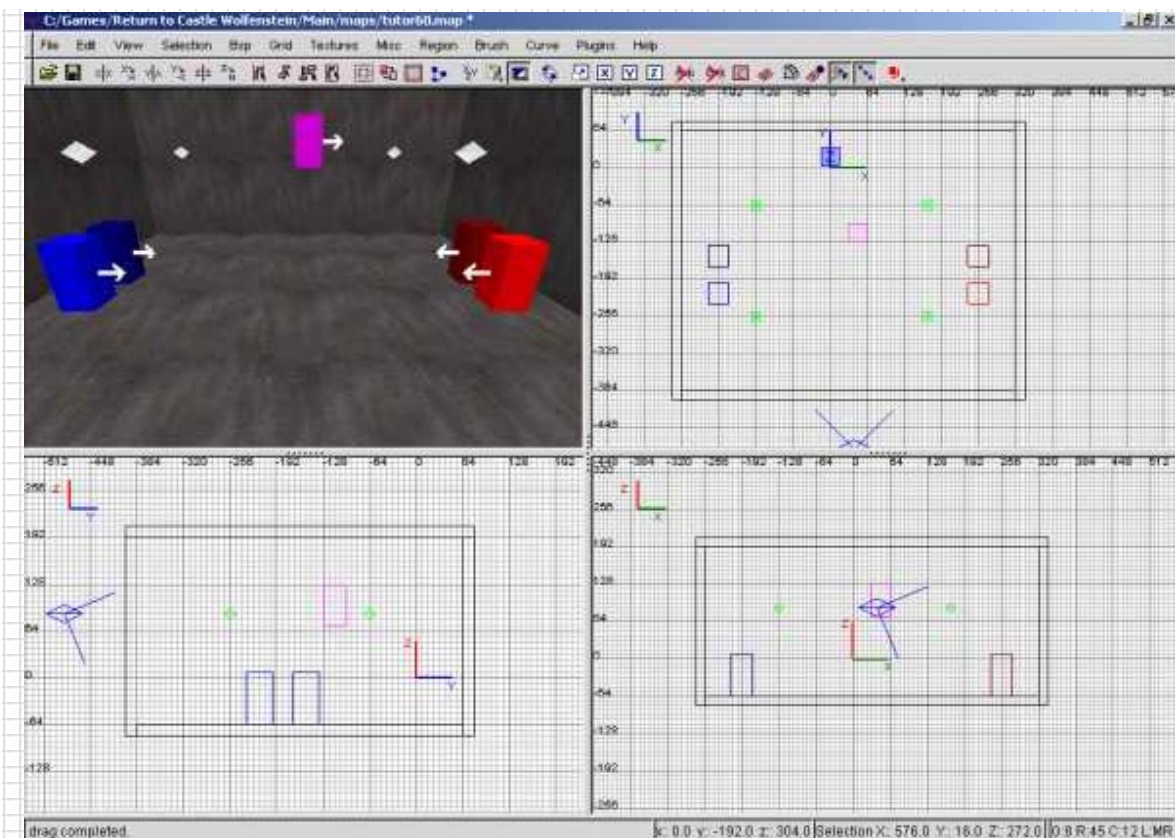
Nun drückst du die Taste "ESC" um den Startpunkt zu deselektieren und klickst wieder in der Top-Ansicht 2 mal mit der rechten Maustaste und wählst wieder "Team", jedoch als Unterpunkt "team_CTF_bluespawn". Hier spawnen die Allies. Nun sieht es so aus:



Natürlich kannst du auch die beiden Kästchen ineinander schieben, dann ist der Respawnpunkt und der Startpunkt der selbe. Du kannst in einer richtigen Map natürlich auch vorgezogene Respawnpunkte setzen, dazu kannst du ja die einzelnen Kästchen nach deinen Vorstellungen verschieben.

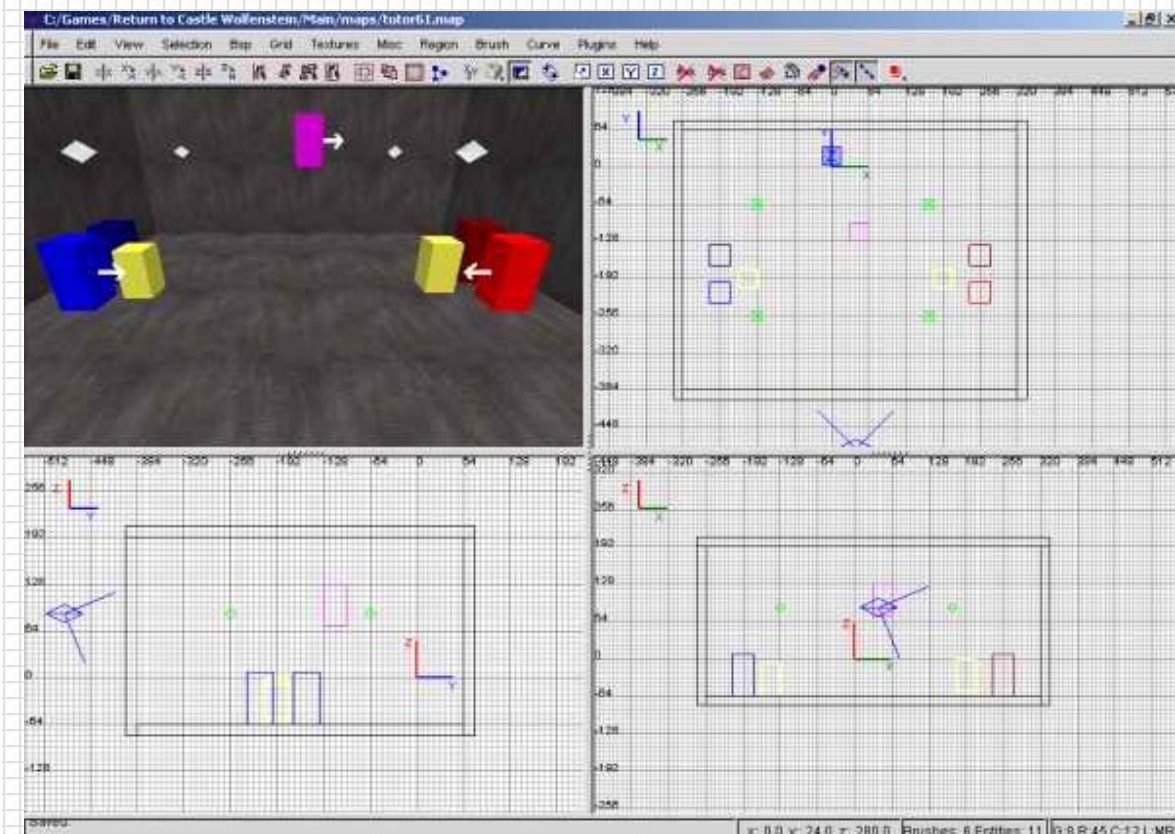
Nun wollen wir natürlich noch den Axis noch einen Startpunkt und einen Respawnpunkt spendieren. Also klickst du wieder 2 mal mit der rechten Maustaste in die Top-Ansicht und wählst "Team" und als Unterpunkt "team_CTF_redplayer".

Jetzt klickst du nochmals 2 mal mit der rechten Maustaste in die Top-Ansicht und wählst "Team" und als Unterpunkt "team_CTF_redspawn". Sieht es so aus:



Nun hätten wir es soweit geschafft. Wenn die Pfeile deiner Startpunkte nicht in die Raummitte zeigen, kannst du diese ja über das Entity-Fenster ändern.

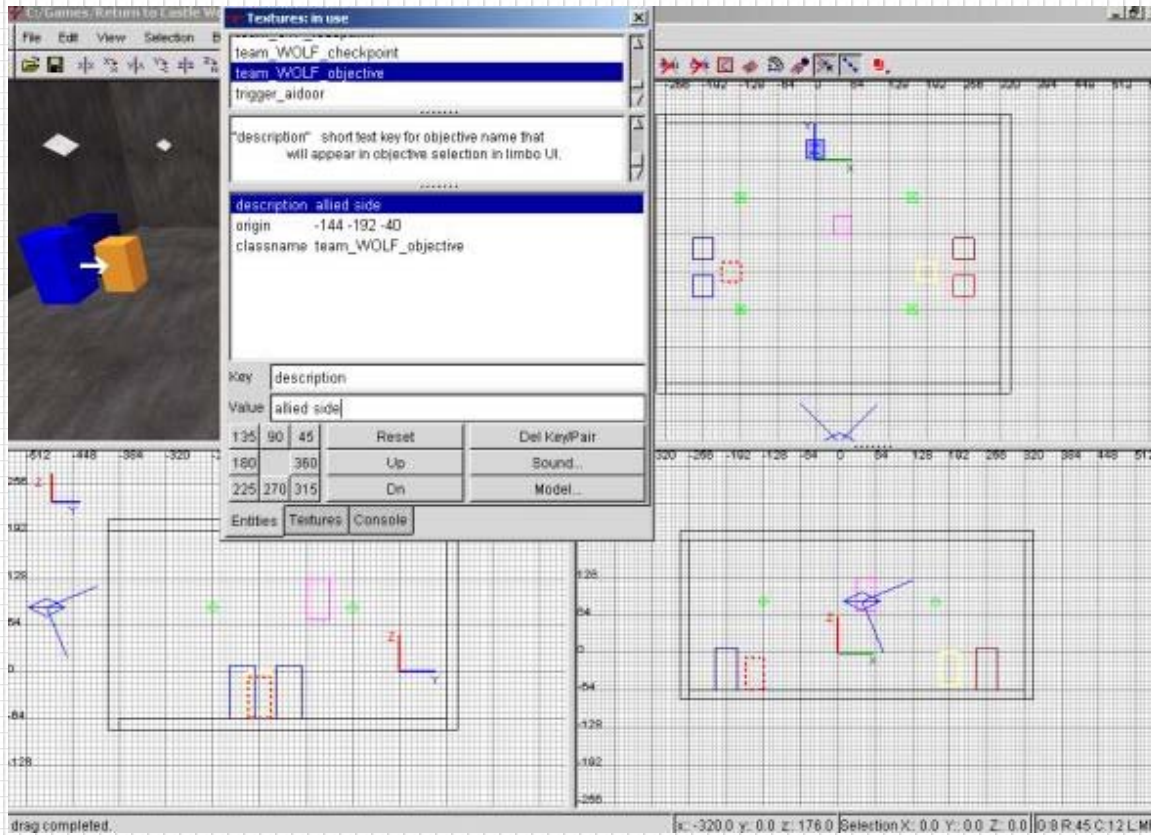
Nun wollen wir aber noch eine kleine Beschreibung der jeweiligen Seite hinzufügen. Dazu klickst du 2 mal mit der rechten Maustaste und wählst hier "team_WOLF_objective". Diesen schiebst du nun vor die 2 blauen Kästchen. Das wiederholst du nun bei den roten Kästchen. Dann sieht es so aus:



Jetzt wählst du das "team_WOLF_objective" vor den beiden blauen Kästchen an. Nun drückst du die Taste "N" um das Entity-Fenster zu öffnen. Hier gibst du ein:

- Key: "desription"
Value: "Allied Side"

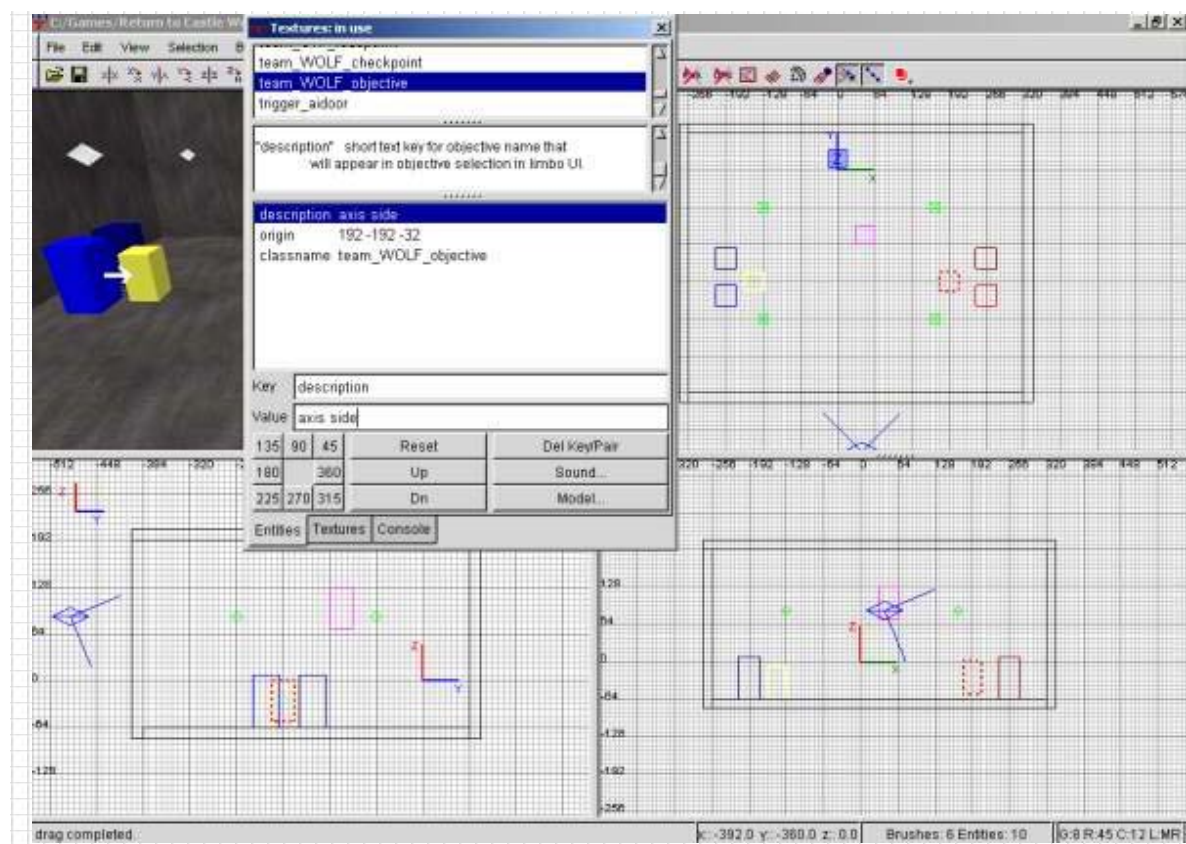
und bestätigst diese Eingabe mit "Enter". Nun sieht es so aus:



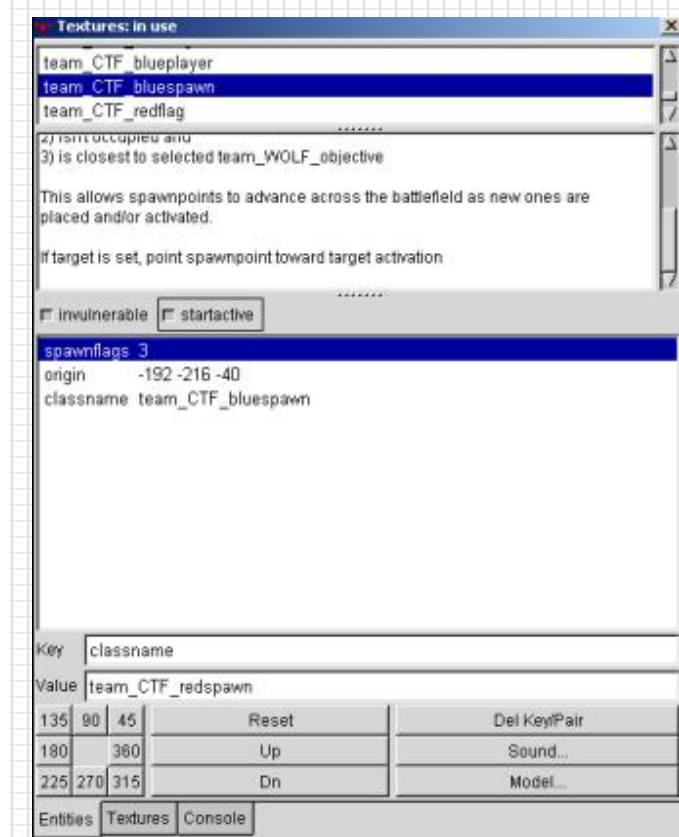
Nun schliesst du das Entity-Fenster und drückst die Taste "ESC", um das Entity zu deselektieren. Danach selektierst du den "team_WOLF_objective", der sich vor den beiden roten Kästchen befindet. Nun drückst du wieder die Taste "N" um das Entity-Fenster zu öffnen und hier gibst du nun ein:

- Key: "desription"
Value: "Axis Side"

Dann sieht es so aus:



Nun schliesst du wieder das Entity-Fenster und drückst wieder "ESC", um das Entity zu deselektieren. Nun selektierst du einen der beiden respawn-Punkte und drückst wieder "N", um das Entity-Fenster aufzurufen. Hier aktivierst du noch die beiden Felder "startactive" und "invulnerable":



Das gleiche machst du jetzt mit dem anderen Respawnpunkt:



Nun kannst du gleichmal die Map compilieren und dein Werk bewundern.

[zurück zur Hauptseite](#)

183759



Kamera:

Beispielmap: "tutor64.map"
Ergebnissmap: "tutor65.map"

Nun, mit diesem Tutorial will ich dir zeigen, wie man eine Kamera in die Map einbinden kann. Manche mögen das noch aus den alten Q3-Zeiten kennen, da dort manche Teleporter Stellen gezeigt haben, an die man teleportiert wird. Nun wollen wir so eine Kamera bauen - natürlich kannst du dir auch eine ganz normale Kamera bauen, z.B. zur Überwachung einer taktisch wichtigen Stelle in deiner Map.

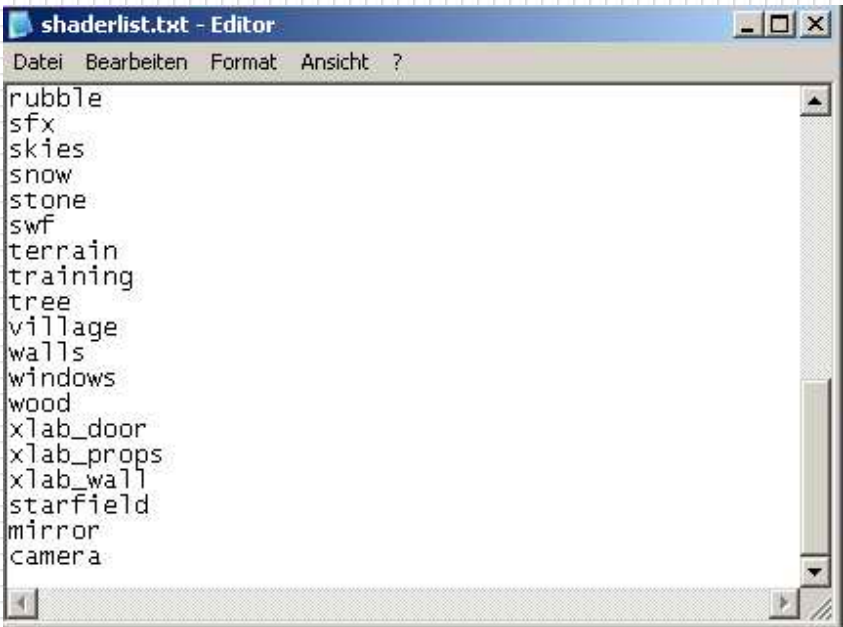
Fangen wir an, dieses Mal müssen wir wieder auf einige neue Texturen und einen Shader zurückgreifen, die du hier ein einer [Zip-Datei](#) finden kannst. Nun erstellst du im Textures-Verzeichnis einen neuen Ordner, den du "common" nennst. Nun brauchen wir noch einen zweiten Ordner, den du "tutcam" nennst und ebenfalls im Textures-Ordner erstellst.

Nun befinden sich in dieser Zip-Datei 6 Dateien - und diese müssen wir jetzt in unsere 2 Verzeichnisse kopieren. Dazu habe ich dir hier eine Tabelle erstellt, wo du sehen kannst, welche Datei wohin gehört:

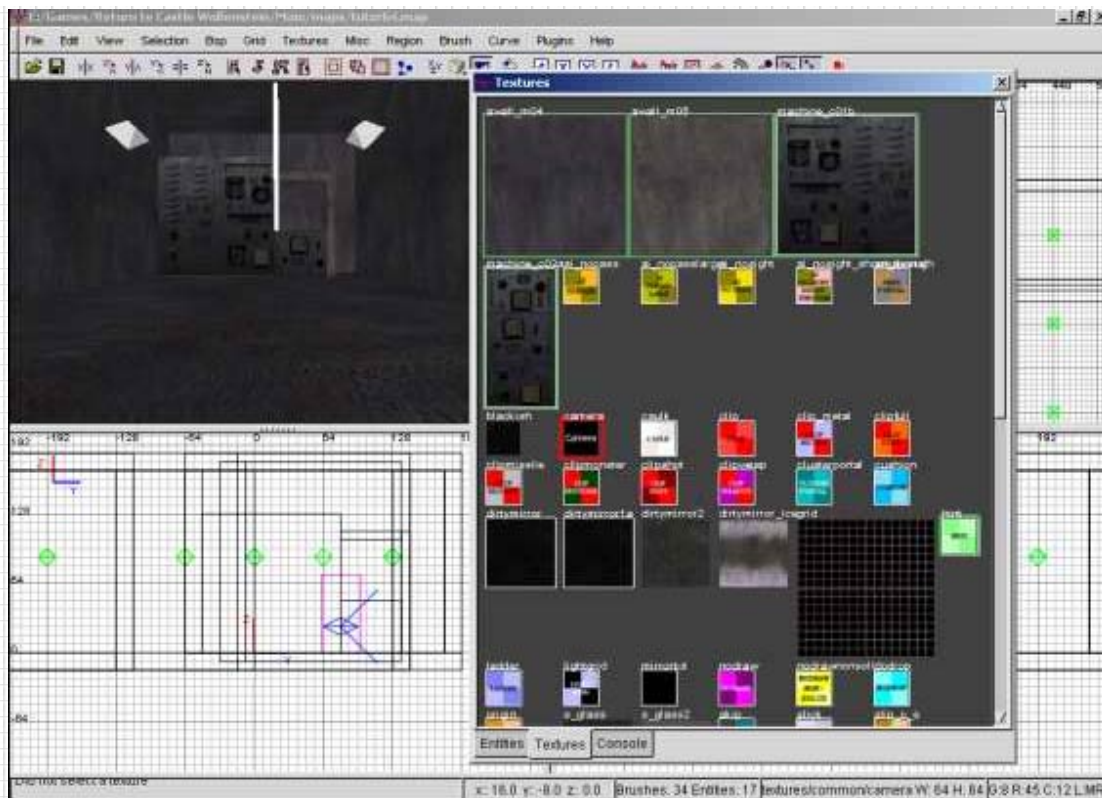
Datei-Name:	wird in folgenden Ordner kopiert:
camera.shader	Main/scripts
camera.tga	Main/textures/common
portal_ground.tga	Main/textures/tutcam
portal_noise.tga	Main/textures/tutcam
screenfx.jpg	Main/textures/tutcam
scrollbar_portal.tga	Main/textures/tutcam

Nun haben wir schon fast die ganze Vorarbeit geleistet - doch da wir auch einen Shader mit von der Partie haben, müssen wir diesen noch in die Shaderlist.txt eintragen. Die Datei befindet sich, wie du ja schon weißt, im Verzeichnis Main/scripts. Also öffnen wir nun diese Text-Datei und schreiben in die letzte Zeile "camera" (ohne die Anführungszeichen). Dann müsste es ungefähr so aussehen:

ACHTUNG: Die anderen Shader können bei dir abweichen, da zu z.B. aus anderen Tutors bereits Einträge in dieser Datei stehen hast.

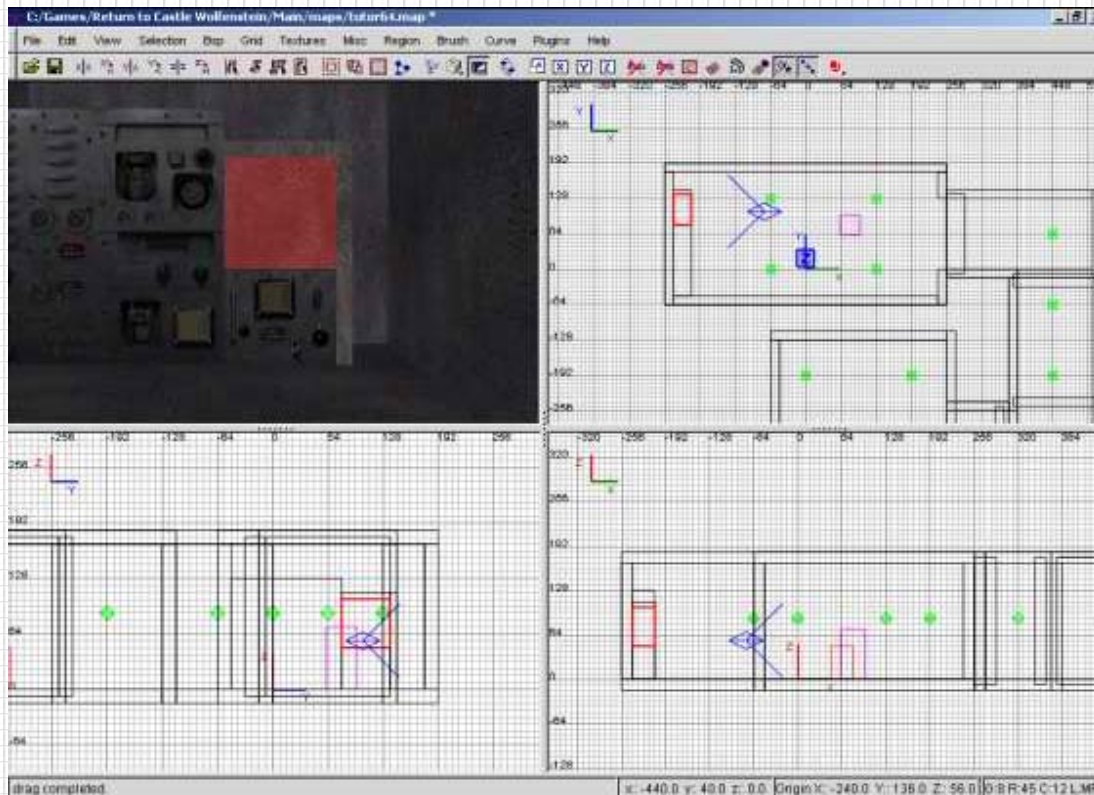


Nun können wir loslegen und starten den Radiant. Nun lädst du die Map "tutor64.map". Hier habe ich dir eine kleine Map gebaut, die ein kleines Kontroll-Terminal enthält. Hier soll man zum schluss sehen können, ob die MG42-Stellung besetzt ist oder ob es ungefährlich ist, den Gang zu passieren. Nun lädst du die Common-Texturen:

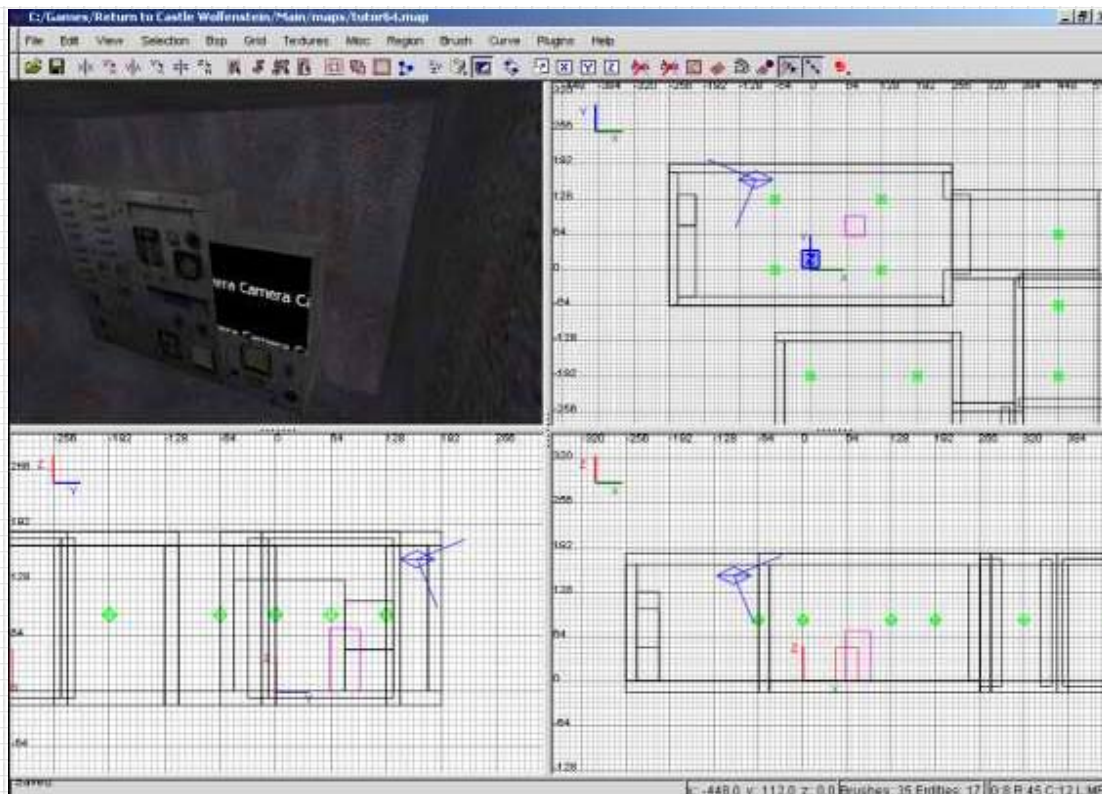


Hier siehst du nun die Camera-Textur. Wenn du alles richtig gemacht hast, hat sie einen weissen Rand, was bedeutet, dass es einen Shader gibt, der ihr weitere Eigenschaften gibt - in unserem Fall natürlich, eine Camera zu sein.

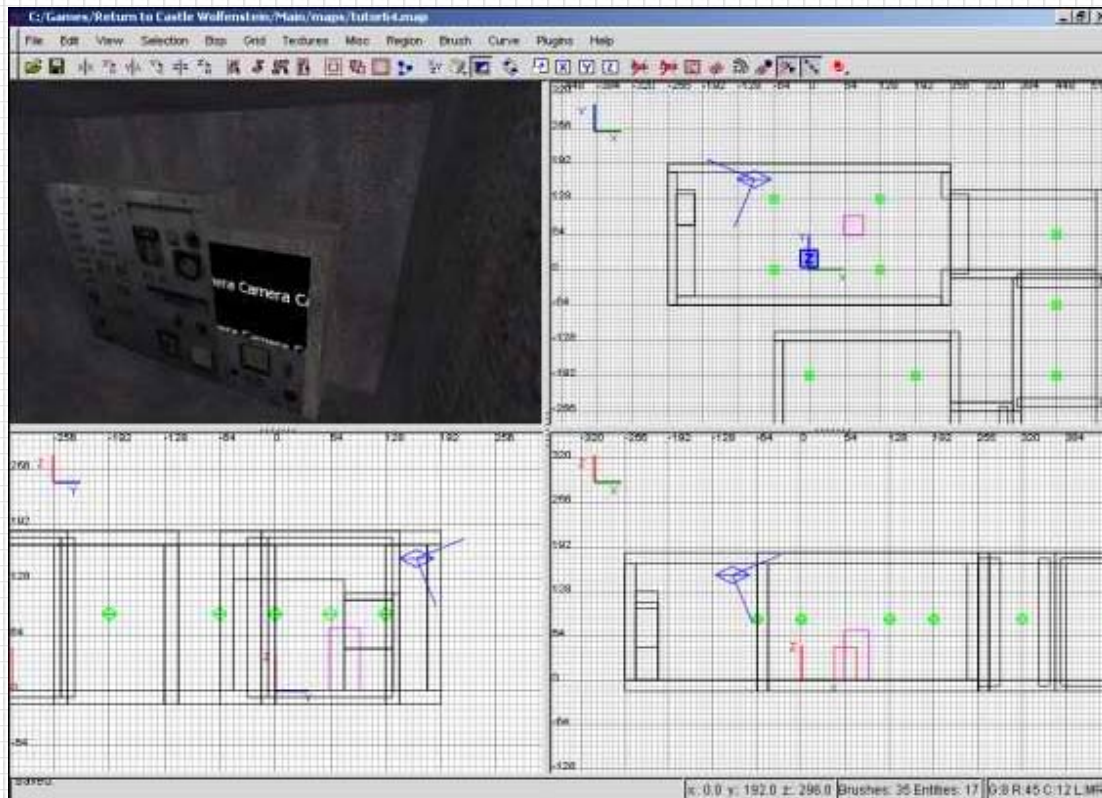
Nun bauen wir uns die Camera - ich setze sie in das Loch am Kontroll-Terminal ein - du kannst sie natürlich auch etwas anders gestalten. Du musst beachten, dass nur die Vorderseite mit der Camera-Textur belegt ist - die anderen Seiten belegst du am besten mit einer normalen Textur, z.B. wähle ich immer zuerst eine andere Textur und lege sie über den ganzen Brush:



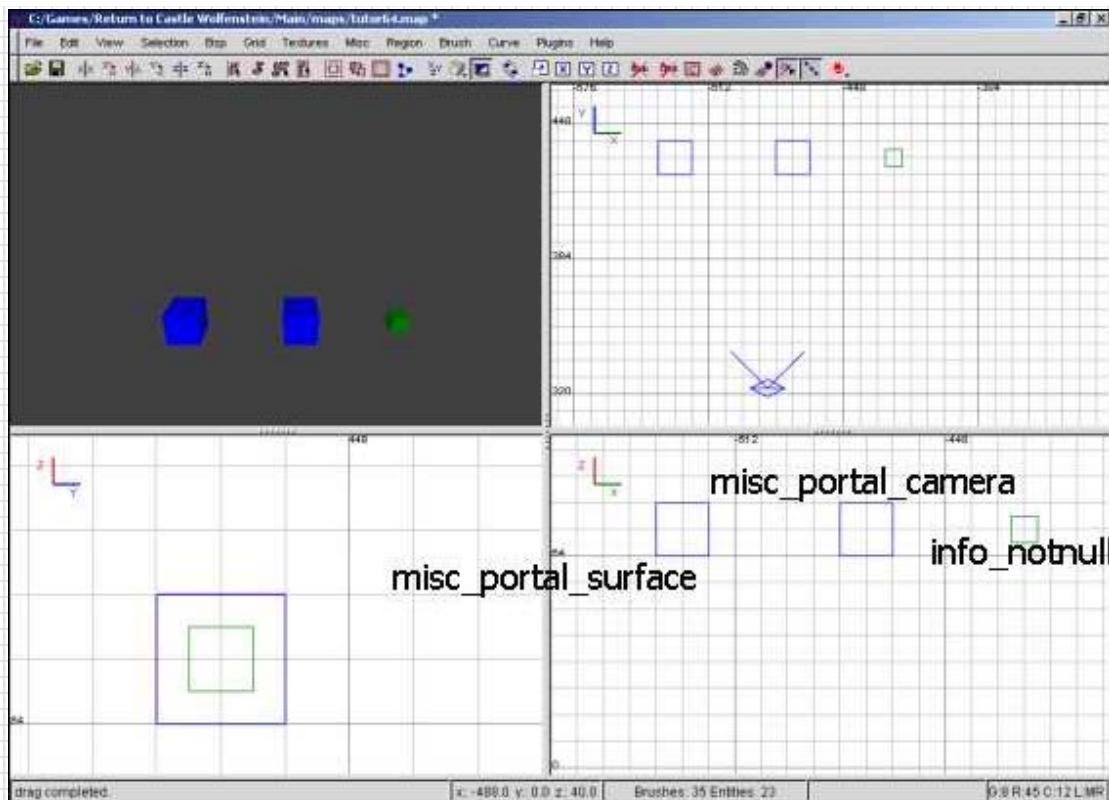
Und anschliessend belege ich die Vorderseite mit der Camera-Textur:



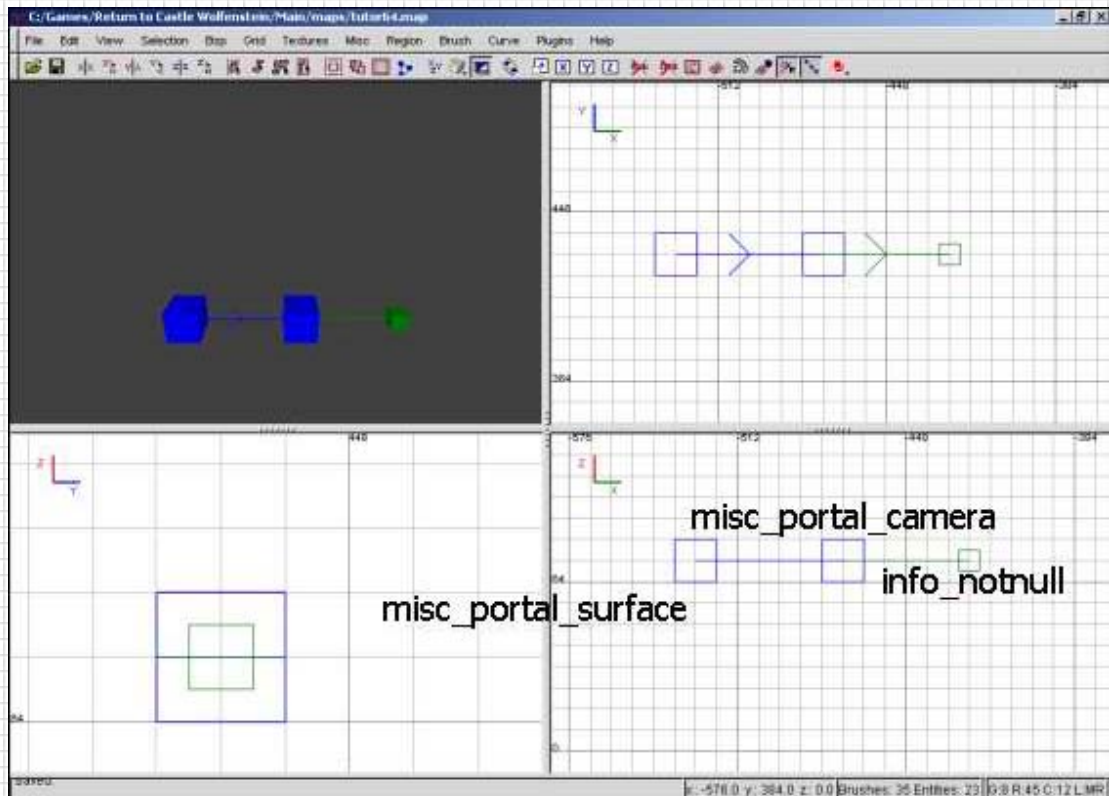
Ich habe die beiden "Rahmen-Brushes" gehidert, so dass du besser sehen kannst, dass auch wirklich nur die Vorderseite mit der Camera-Textur belegt ist. Nun dürfte es bei dir so aussehen:



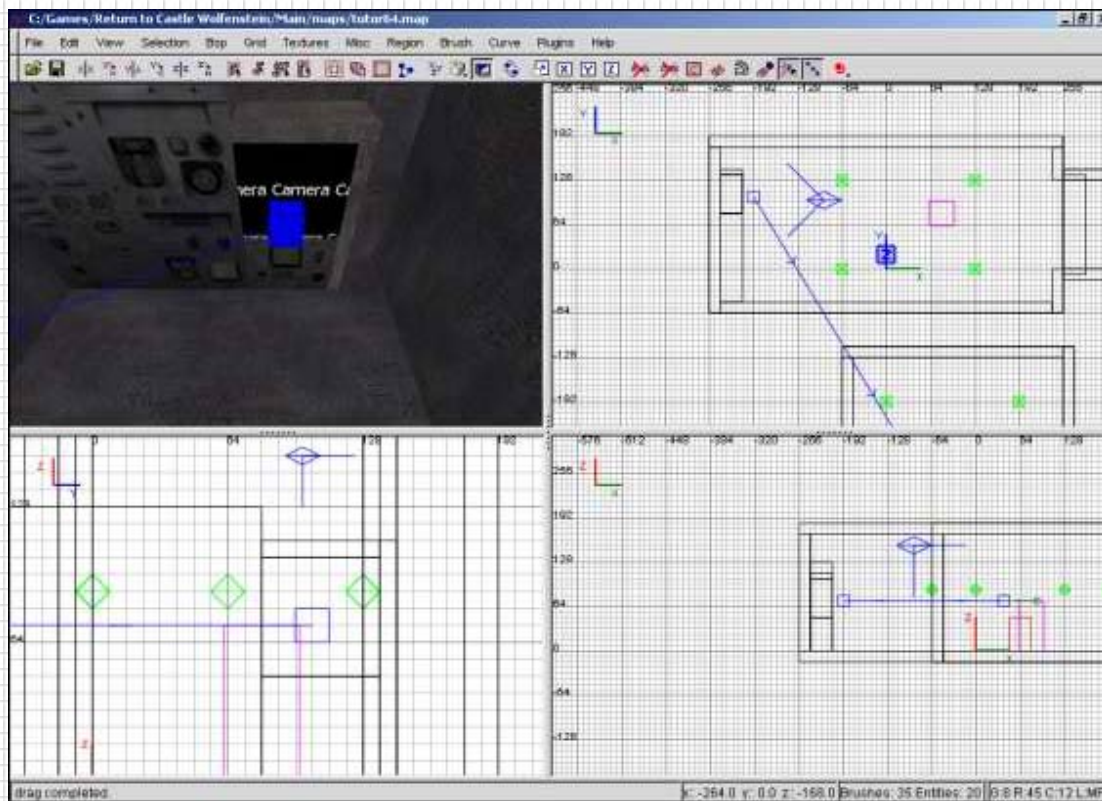
Das hätten wir soweit - jetzt haben wir schonmal den Monitor, an dem das Bild ausgegeben wird - jedoch fehlt uns noch die Quelle, die uns das Bild liefert. Dazu brauchen wir 3 Entities. Zum einen ist das ein "misc_portal_surface", ein "misc_portal_camera" und ein "info_notnull":



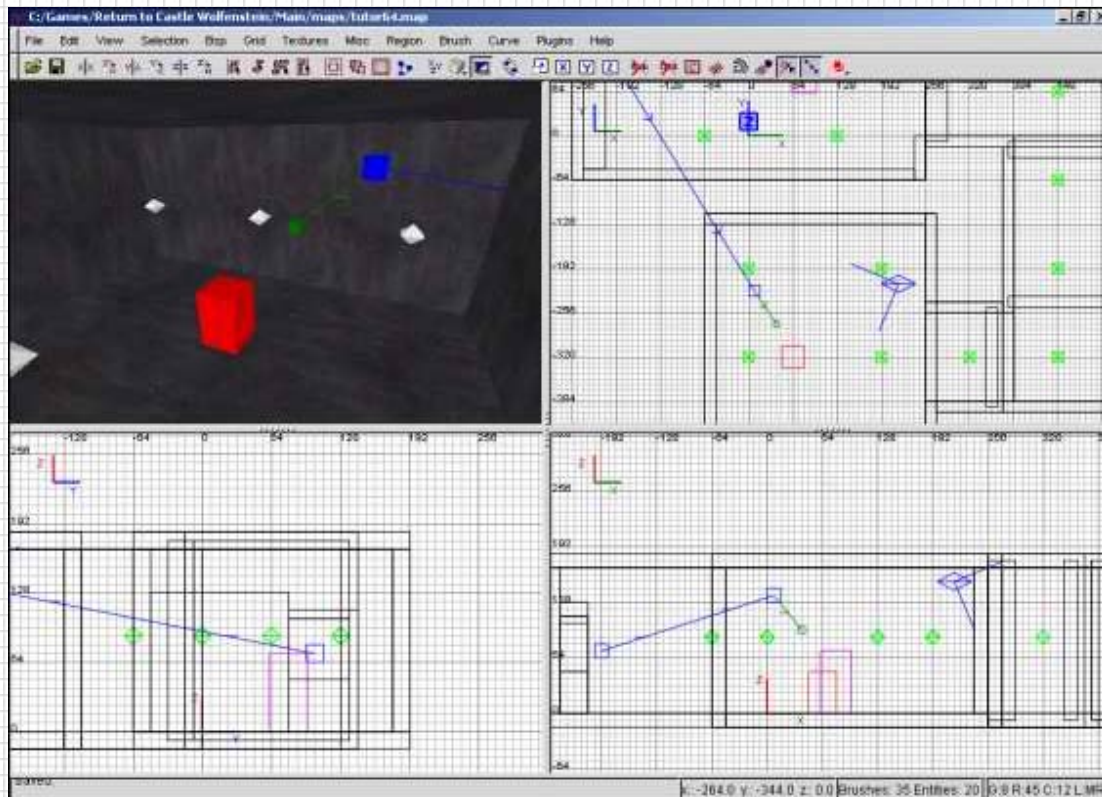
Diese 3 Entities müssen wir jetzt nur noch verbinden und natürlich noch an die richtigen Stellen schieben. Dazu aber später. Nun drückst du die "ESC"-Taste. Dann markierst du zuerst das "misc_portal_surface" und dann das "misc_portal_camera" Entity. Nun drückst du "STRG" + "K". Nun müsste eine blaue Linie erscheinen. Nun drückst du wieder "ESC" und markierst jetzt das "misc_portal_camera" und dann das "info_notnull" Entity. Nun drückst du auch wieder "STRG" + "K". Nun müssten die beiden Entities über eine grüne Linie verbunden sein:



So, nun müssen wir nur noch für die 3 Entities das passende Plätzchen suchen. Das "misc_portal_surface" setzen wir genau wie beim Spiegel genau vor den Brush mit der Camera-Textur, die beiden anderen Entities verschiebst du erstmal in den Raum, in der die MG42 steht. Um diese beiden Entities kümmern wir uns später. Nun sieht das ganze so aus:



Bis hierher dürfte dir das alles aus dem Spiegel-Thema bekannt vorkommen. Wie du in der Top-Ansicht gut erkennen kannst, sind die Entities immernoch untereinander verbunden, du kannst sie also frei verschieben, wie es dir gefällt. Nun wenden wir uns mal den beiden anderen Entities zu. Das `misc_portal_camera` ist quasi unser Auge, aus dem wir sehen - also positionieren wir es überhalb der MG-Stellung. Das `info_notnull` steht für die Blickrichtung, also habe ich dieses Entity schräg nach unten und auf die MG-Stellung blickend gesetzt. Du kannst dir natürlich auch einen anderen Blickwinkel raussuchen. Bei mir sieht es jetzt so aus:

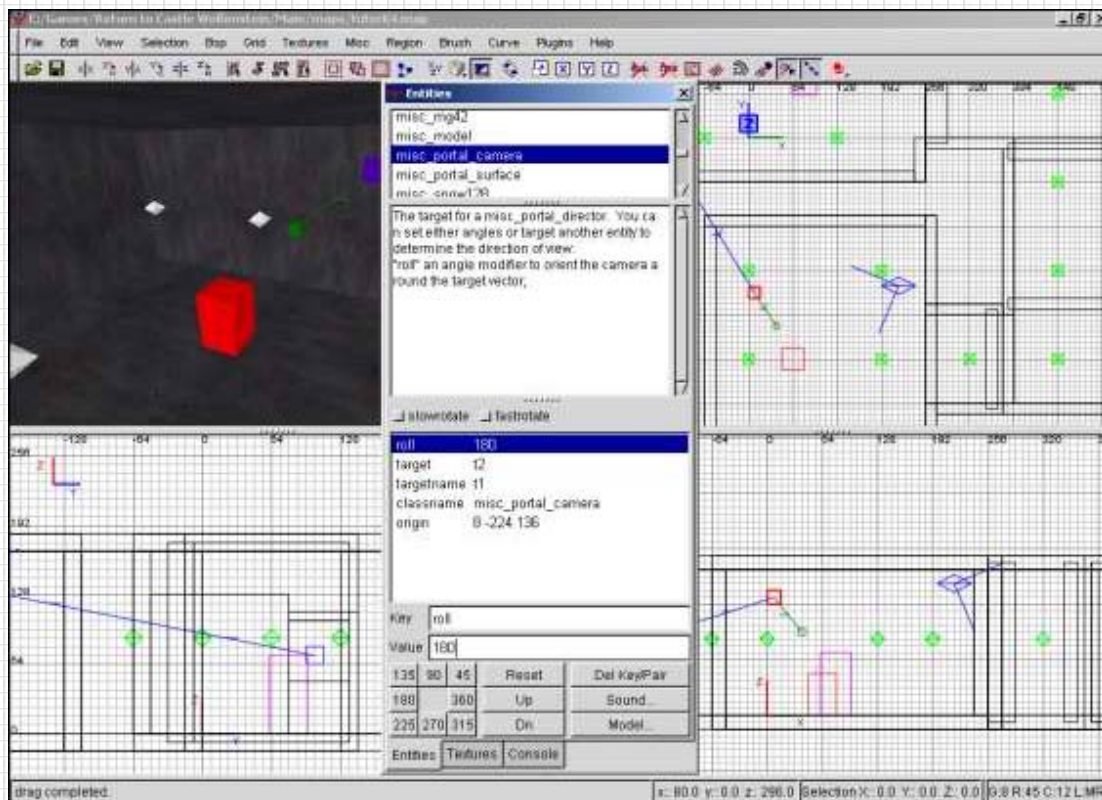


Nun haben wir noch eine kleine Einstellung zu tätigen. Wie beim menschlichen Auge auch funktioniert die Kamera nach dem Loch-Kamera-Prinzip. Das bedeutet, das Bild wird auf dem Kopf dargestellt, das müssen wir natürlich noch ändern. Dazu drückst du die "ESC"-Taste und klickst das `misc_portal_camera`-Entity an und drückst die Taste "N" um das Entity-Fenster aufzurufen. Hier gibst du

folgendes ein:

- Key: "roll"
Value: "180"

Fertig sieht es dann so aus:



Wenn du den Eintrag richtig gemacht hast, siehst du ihn in der Liste (ich habe dir den Eintrag blau markiert). Nun kannst du mal deine Map compilieren und dein Werk bewundern:



So, nun muss ich aber noch sagen, dass solche Kameras natürlich ziemliche Performance-Killer sein können - je nachdem, wieviel man durch sie von der Map sehen kann und wieviel Detail zu sehen ist, sacken hier die FPS auch schonmal ganz schön ab. Dann solltest du vielleicht Stellen suchen, die nicht so sehr viel Detail enthalten oder auch sonst nicht allzuviel im Blickfeld zu sehen ist. Aber lass dich davon nicht abschrecken, diese Kameras sind tolle Elemente, um Taktiken besser zu planen.

[zurück zur Hauptseite](#)

183759



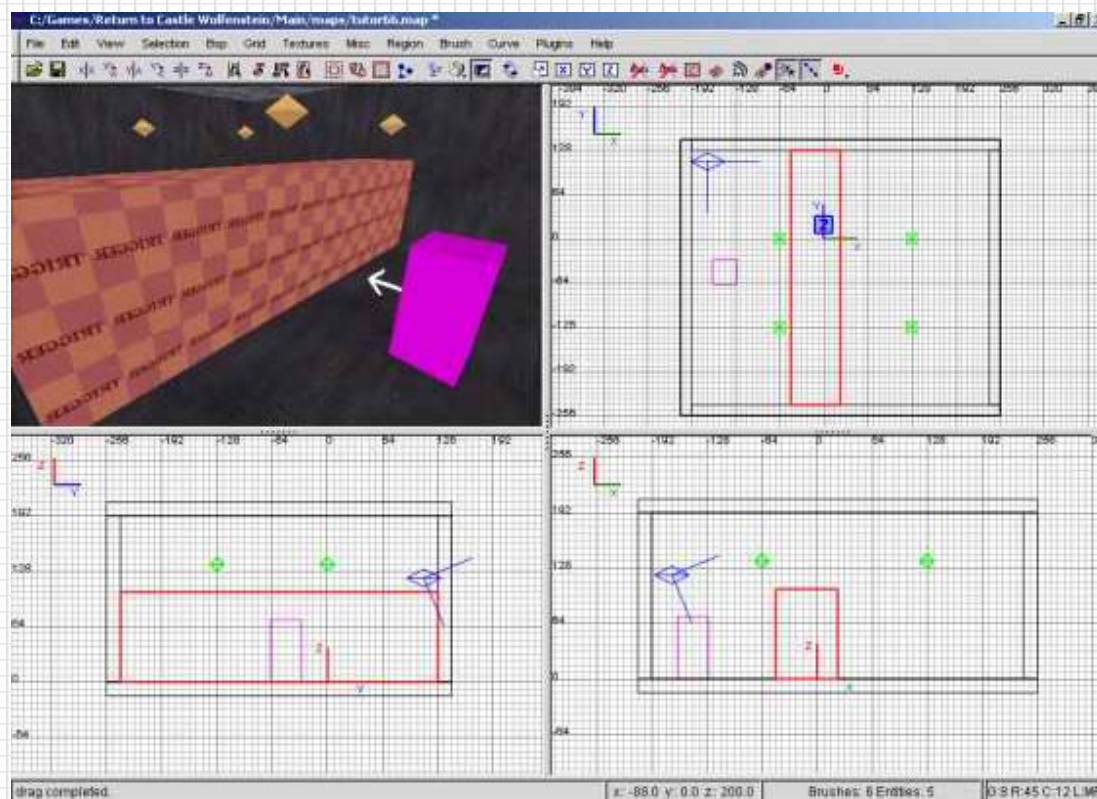
Explosionen:

Beispielmap: "tutor66.map"

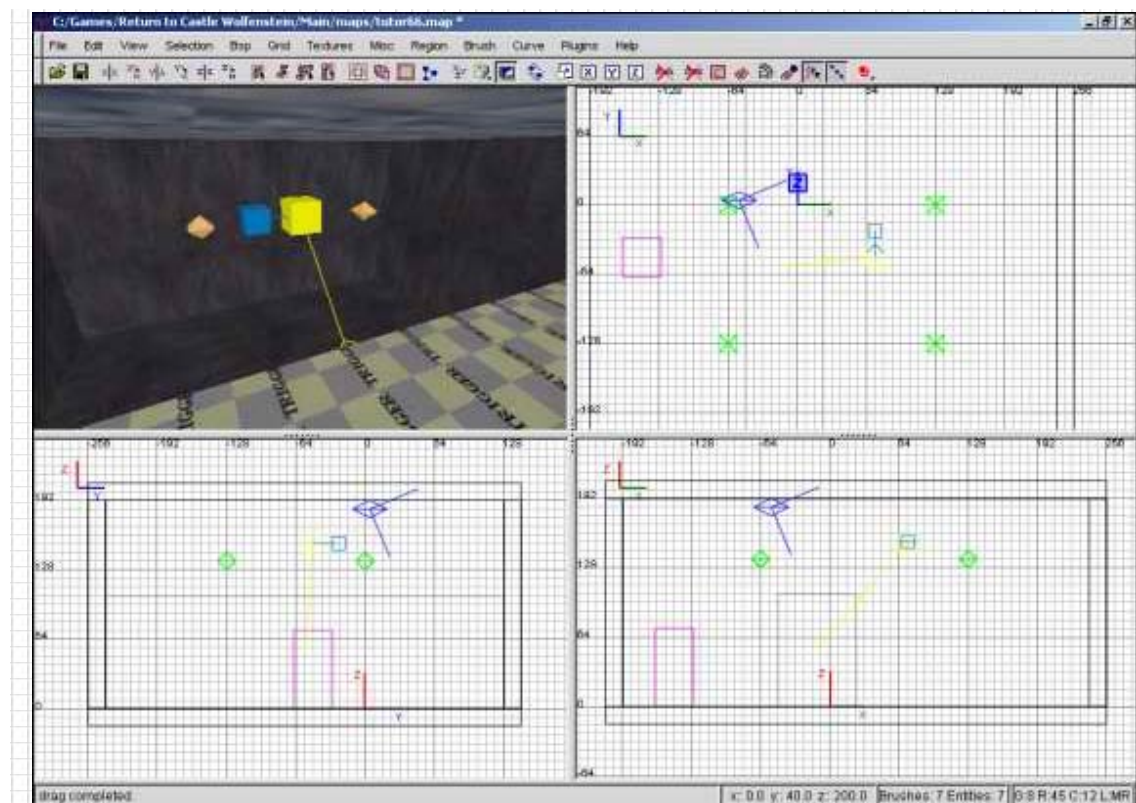
Ergebnissmap: "tutor67.map"

In dieser Map will ich dir zeigen, wie man Explosionen erstellt, die z.B. von einschlagenden Granaten, Mörsern oder auch Flakfeuer kommen können.

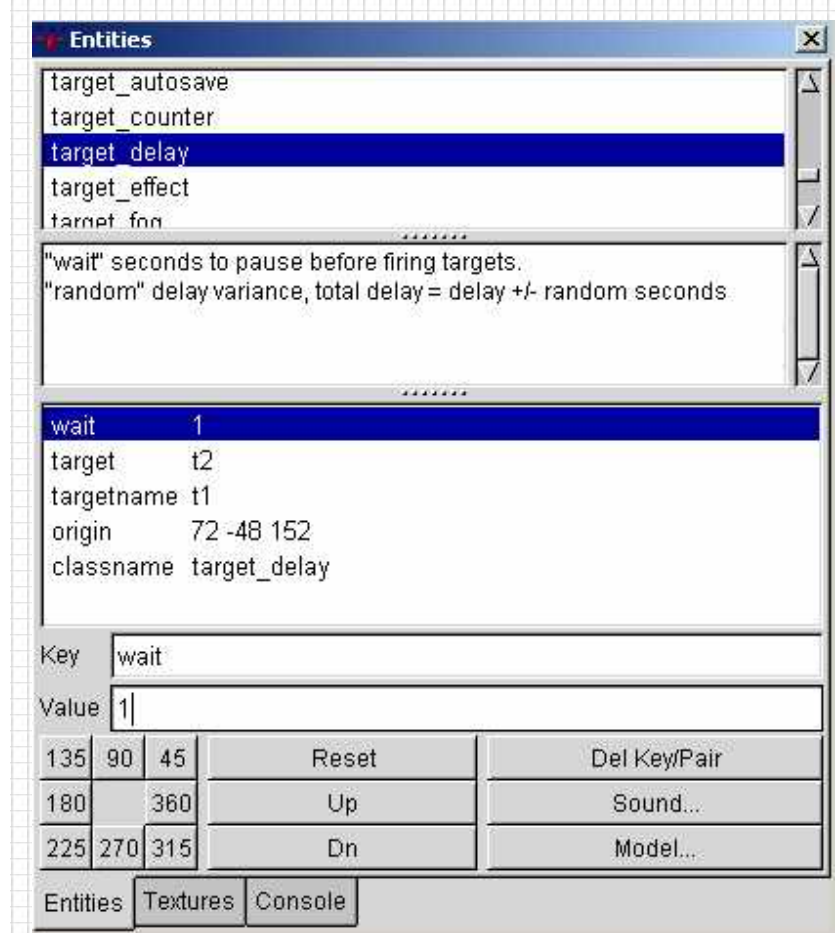
Dazu öffnest du nun die Beispielmap. Ich will dir in dem Beispiel den Angriff einer Stalinorgel zeigen (die Granaten explodieren in der Luft, normal sollte hier die Lunge zerreißen). Dazu benötigen wir zuerst einen Auslöser. Dazu baust du dir einen Brush, der unser Schalter darstellt. Diesen belegst du mit der Textur "common/trigger" und klickst jetzt 2 mal mit der rechten Maustaste. Hier wählst du "trigger" und als Unterpunkt "trigger_multiple". Nun könnte es bei dir so aussehen:



Nun klickst du 2 mal mit der rechten Maustaste und wählst "target" und als Unterpunkt "target_effect". Dann klickst du wieder 2 mal mit der rechten Maustaste und wählst "target" und als Unterpunkt "target_delay". Je nachdem, wo du nun die Explosionen haben willst, setzt du nun das "target_effect"-Entity. Genau daneben setzt du das "target_delay". Nun drückst du "ESC" um alles zu deselektieren. Dann klickst du zuerst den Trigger-Brush an, danach das "target_delay"-Entity. Jetzt drückst du die "STRG" Taste und die "K"-Taste. Nun müsste ein gelber Strich erscheinen, der den Trigger mit dem Target_delay verbindet. Nun drückst du wieder die Taste "ESC" und dann den target_delay und hinterher den target_effect. Nun drückst du wieder "STRG" und "K" um die beiden Entities zu verbinden. Nun sieht es ungefähr so aus:



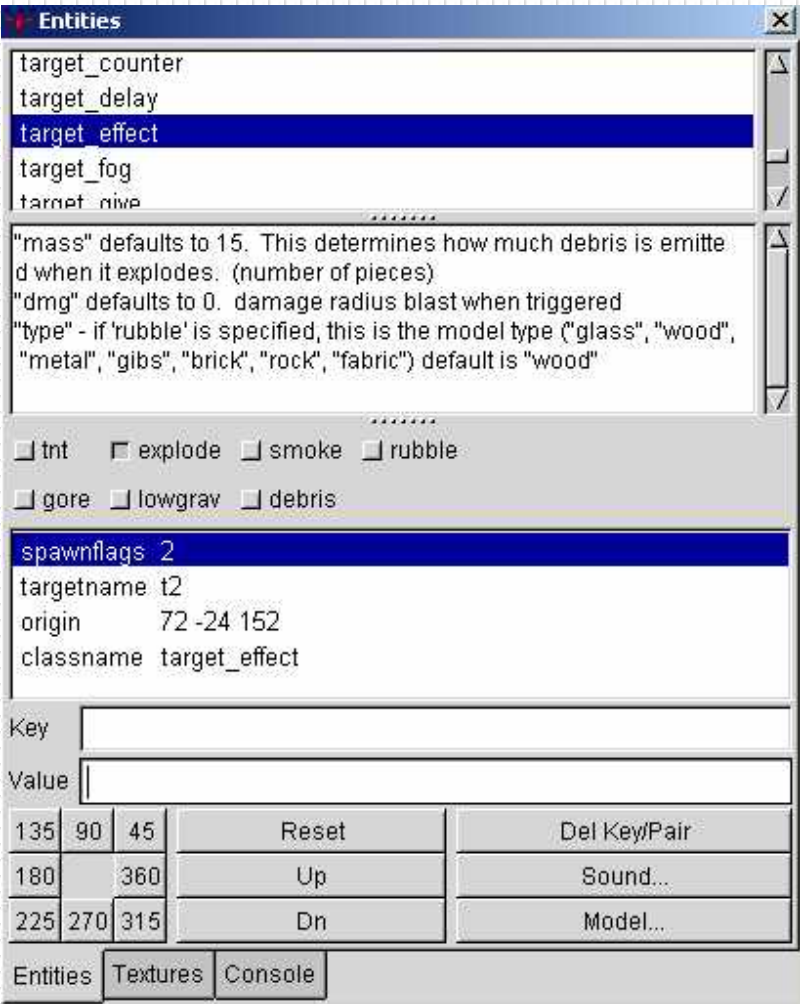
Nun drückst du "ESC" um alles zu deselektieren. Danach wählst du das target_delay aus und drückst die Taste "N", um die Entities zu öffnen:



Hier gibst du folgendes ein:

- Key: "wait"
Value: "X" (X steht für einen Wert in Sekunden, z.B. im Bild oben 1 Sekunde Delay)

Nun bedeutet das, dass der target_delay den target_effect jede Sekunde nur einmal aktiviert. Nun kannst du das target_delay deselektieren und das target_effect selektieren. Nun drückst du wieder die "N"-Taste:



Hier kannst du bei den Spawnflags die Eigenschaft des Effekts festlegen. In unserem Fall wählen wir "explode", jedoch kannst du auch die anderen Effekte ausprobieren. Z..B. beim Gore-Effekt erscheinen in einer MP-Map rot-grün-blau-farbige Dreiecke. Das liegt daran, dass der Effekt nur für den Singleplayer konzipiert ist.

Nun kannst du natürlich noch einige solcher target_effect und target_delays einsetzen und sie mit verschiedenen "Wait"-Keys bestücken, so dass es etwas natürlicher aussieht.

[zurück zur Hauptseite](#)

183759



Shader:

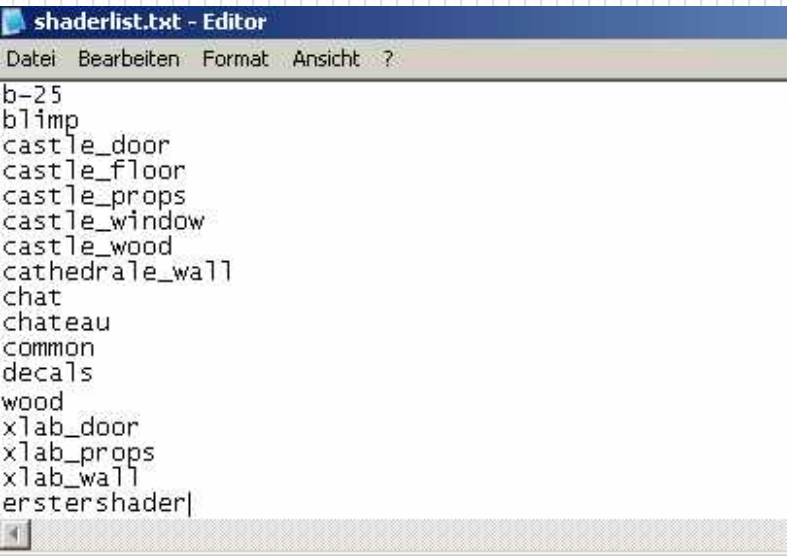
Nachdem du dich nun in den verschiedenen Themen mit vorgegebenen Shadern rumschlagen musstest, will ich dir hier erklären, wie man einen Shader schreibt und worauf man zu achten hat.

Shader sind Textdateien, deren Inhalt den Texturen Eigenschaften verleihen können, z.B: metallischen Glanz, Wassereigenschaften, Spielgeeigenschaften oder auch Metal-Sound von sich zu geben, wenn man über sie läuft.

Da Shader Textdateien sind, kann man sie mit jedem beliebigen Textbearbeitungsprogramm schreiben, z.B. auch NotePad oder WordPad. Um einen Shader zu schreiben, gibt es einige Zeichen, ohne die ein Shader nicht auskommt:

- die geschwungenen Klammern: { } Sie markieren den Anfang und das Ende des Shaders, sowie Anfang und Ende einer Schicht (auch Stage genannt)
- die zwei Slashes: // Sie dienen nur dazu, einen Kommentar in den Textdateien zuzulassen. Alles, was hinter diesen beiden Zeichen bis zum Ende der Zeile steht, wird von der Engine nichtmehr zum Shader gerechnet.
- Du solltest beachten, dass du im Shader auf Grossschreibung und Umlaute verzichtest.

Wie du schon weisst, müssen Shader in die Shaderlist.txt eingetragen werden, damit sie funktionieren. Dazu öffnest du jetzt die Datei Shaderlist.txt und schreibst dort in eine neue Zeile "erstershader":



Nun öffnest du dein Textprogramm und schreibst hinein:

```
// mein erster Shader
textures/shader1/textur
{
    {
        map textures/shader1/textur.tga
    }
}
```

Dies ist der wahrscheinlich einfachste Shader überhaupt - denn er verändert die Textur selbst garnicht. Doch für unsere Zwecke reicht er

völlig aus. Kommen wir nun zur Erklärung des Shaders:

// mein erster Shader	Ist lediglich ein Kommentar, er dient nur zur Erklärung oder dazu, bestimmte Teile des Shaders zu testzwecken abzuschalten.
textures/shader1/textur	Name des Shaders - damit legst du fest, wo du später die Textur finden kannst.
{	Beginn des Shaders
{	Hier beginnt die erste Schicht
map textures/shader1/textur.tga	Mit dieser Anweisung (map <Pfad zur Textur>) gibt man der Engine die Anweisung, die Oberfläche der Textur mit der angegebenen Textur zu belegen.
}	Hier endet die erste Schicht
}	Hier endet der Shader

Nun speicherst du diese Datei im Ordner "Main/scripts" unter dem Namen "erstershader.shader" ab.

Erstellst du nun einen Ordner namens "shader1" im Textures-Ordner und kopierst dort eine beliebige Textur namens "textur" hinein, läuft der Shader.

Nun wagen wir uns an einen neuen Shader, der diesmal etwas anspruchsvoller wird. Wir wollen einen Shader haben, der aus 2 Texturen besteht. Diese beiden [Texturen](#) findest du [hier](#). Nun gucken wir sie uns gleichmal an:



Diese beiden Texturen sollen dargestellt werden. Du kannst natürlich auch beide Texturen durch andere ersetzen. Bei uns soll folgender Effekt entstehen: Logo1.tga UND Logo2.tga sollen gleichzeitig dargestellt werden. Dazu benutzen wir einfach den folgenden Shader:

```
textures/meinshader/logo2
{
    {
        map textures/meinshader/logo1.tga
    }

    {
        map textures/meinshader/logo2.tga
        blendfunc add
    }

    {
        map $lightmap
        blendfunc filter
    }
}
```

Nun erkläre ich dir erstmal, was es mit den neuen Befehlen auf sich hat:

textures/meinshader/logo2	Name des Shaders - damit legst du fest, wo du später die Textur finden kannst.
{	Beginn des Shaders
{	Beginn der ersten Stage
map textures/meinshader/logo1.tga	Der Pfad zur Textur, die in der ersten Stage dargestellt wird.
}	Ende der ersten Stage
{	Beginn der zweiten Stage
map textures/meinshader/logo2.tga	Der Pfad zur zweiten Textur, die in der zweiten Stage dargestellt wird.
blendfunc add	Dieser Parameter gibt an, dass die beiden Texturen additiv (also übereinander) dargestellt werden.
}	Ende der zweiten Stage
{	Anfang der dritten Stage
map \$lightmap	Gibt an, dass die normale Textur (logo1.tga) als Lightmap verwendet wird.
blendfunc filter	Gibt an, wie die Lightmap überblendet wird.
}	Ende der dritten Stage
}	Ende des Shaders

So, jetzt sind wir ja schon ziemlich weit - einen Shader mit 3 Stages ist schon ziemlich schwer zu verstehen. Nun toppen wir das ganze, in dem wir das Logo nun noch blinkend dargestellt haben wollen. Dazu kommt lediglich zwei Zeilen im Shader dazu - zum Schluss sieht der Shader so aus:

```
textures/meinshader/logo1
{
  qer_editorimage textures/meinshader/logo2.tga

  {
    map
    textures/meinshader/logo1.tga
  }

  {
    map
    textures/meinshader/logo2.tga
    rgbGen wave square 0.5 0.5 0
    1.1
    blendfunc add
  }

  {
    map $lightmap
    blendfunc filter
  }
}
```

```
}  
}
```

Nun erkläre ich dir auch gleich, was die beiden neuen Zeilen bedeuten:

textures/meinshader/logo1	Name des Shaders - damit legst du fest, wo du später die Textur finden kannst.
{	Anfang des Shaders
qer_editorimage textures/meinshader/logo2.tga	Gibt an, welches Bild der Shader im Radiant hat (vergist du diesen nicht, so erscheint "Shader Image Missing")
 {	 Anfang der ersten Stage
map textures/meinshader/logo1.tga	Der Pfad zur Textur, die in der ersten Stage dargestellt wird.
}	Ende der ersten Stage
 {	 Beginn der zweiten Stage
map textures/meinshader/logo2.tga	Der Pfad zur zweiten Textur, die in der zweiten Stage dargestellt wird.
rgbGen wave square 0.5 0.5 0 1.1	generiert die Vertexfarbe. Darauf gehe ich gleich näher ein. Der Syntax "wave" generiert den "Blinkabstand" der Textur.
blendfunc add	Dieser Parameter gibt an, dass die beiden Texturen additiv (also übereinander) dargestellt werden.
}	Ende der zweiten Stage
 {	 Anfang der dritten Stage
map \$lightmap	Gibt an, dass die normale Textur (logo1.tga) als Lightmap verwendet wird.
blendfunc filter	Gibt an, wie die Lightmap überblendet wird.
}	Ende der dritten Stage
}	Ende des Shaders

Nun gehe ich etwas näher auf folgenden Teil im Shader ein:

```
rgbGen wave square 0.5 0.5 0 1.1
```

rgbGen generiert ja die Vertex-Farbe. Diese wird mit der Farbe der Textur multipliziert um dann die entgültige Farbe zu erhalten. Normal ist die Vertexfarbe weiss, d.h. die Farbe ändert sich nicht ([Texturfarbe] * 1 1 1). Mit dem Syntax "wave" wird die Engine dazu veranlasst, diese Farbe zu wechseln, d.h. man erzeugt blinkende Effekte. Die komplette Reihenfolge des Befehls lautet so:

```
rgbGen wave <func> <base> <amp> <phase> <freq>
```

<func>
gibt an, in welcher Weisse sich die Vertexfarbe ändern soll - es gibt 5 Parameter für diese Änderung:

- sin (gleichmässige Ab- und Aufwärtsbewegung, entspricht genau der Sinuswelle)
- triangle (Dreiecksbewegung, d.h. Gradliniger Auf- und Abbau der Werte)
- square (Wechsel zwischen zwei Werten, also z.B. an und aus.
- sawtooth (geradliniger Aufbau, radikaler Abfall der Kurve)
- invertedsawtooth (radikaler Aufbau, geradliniger Abbau, Umgekehrte Sawtooth-Funktion)

<base>
gibt den Basiswert an, auf den sich später der <amp>-Wert bezieht.

<amp>
gibt den maximalen Ausschlag der Kurve in Bezug auf den Basis-Wert (<base>) an.

<phase>
Hier mit kann man eine zeitliche Verzerrung der Kurve erzielen, damit z.B. nicht alle Shader in der gleichen Abfolge blinken. Meistens wird <phase> allerdings auf den Wert 0 gesetzt.

<freq>
steht sinnigerweise für "Frequenz", also wie oft eine Kurve innerhalb einer Sekunde wiederholt wird. Der Wert 2 veranlasst den Shader dazu, die Kurve innerhalb einer Sekunde 2mal zu wiederholen. Der Wert 0.5 veranlasst den Shader dazu, die Kurve nur alle 2 Sekunden zu wiederholen.

Bei unserem Shader wechselt also die Vertexfarbe immer zwischen den Werten 1 1 1 und 0 0 0, d.h. das Logo wird angezeigt, wenn der Wert auf 1 1 1 steht. Wechselt der Wert auf 0 0 0, verschwindet das Logo.

Alpha-Texturen:

Wie du bei dem Logo gesehen hast, wird nur der "haradirki"-Schriftzug dargestellt, der Rest wird von der Engine abgeschnitten, d.h. er wird nicht dargestellt. Nun zeige ich dir, wie man eine solche Alphashader selbst hinkommt. Zuerst erstellst du ganz normal deine Textur.



Hier kannst du die Textur "logo2.tga" und deren Alpha-Channel sehen. Zwar kann man hier nicht genau den Unterschied zur richtigen Textur erkennen - doch sind die drei Innenräume von den Buchstaben "a" und "d" komplett weiss - was daraufhin deutet, dass dieser Innenraum auch von der Engine dargestellt werden soll.

Nun kommen wir aber zum Shader:

```
textures/meinshader/logo2
{
    {
        map
        textures/meinshader/logo2.tga
        alphaFunc GT0
    }
}
```

Nun folgt auch gleich die Erklärung zum Shader:

textures/meinshader/logo2	Name des Shaders - damit legst du fest, wo du später die Textur finden kannst.
---------------------------	--

{	Anfang des Shaders
{	Anfang der ersten Stage
map textures/meinshader/logo2.tga	Der Pfad zur Textur, die in der ersten Stage dargestellt wird.
alphaFunc GT0	Durch diesen Parameter zeichnet die Engine nur die Bereiche der Textur, die im Alphakanal einen höheren Wert als 0 0 0 haben (die nicht schwarz sind)
}	Ende der ersten Stage
}	Ende des Shaders

Nun erkläre ich dir diese Zeile noch etwas genauer:

alphaFunc <syn>

Folgende Syntaxe sind hier möglich:

- GT0 "GREATER THAN 0" (diesen Syntax haben wir in unserem Beispiel verwendet)
- LT128 "LESS THAN 128"
- GE128 "GREATER THAN OR EQUAL TO 128"

ACHTUNG: Hierbei sind nur die Farbwerte von 0 - 255 zulässig, nicht die gesamte 16- oder gar die 32-Bit-Palette



Unser Alpha-Kanal besteht nur aus Pixeln, die auf 0 0 0 stehen (die schwarz sind) und aus Pixeln, die auf 1 1 1 stehen (die weiss sind), d.h. das weisse wird dargestellt, das schwarze nicht.

Shader sind, wie du siehst, eine recht schwierige Angelegenheit. Jedoch lassen sich damit wirklich tolle Effekte erzielen, die jede Map enorm aufwerten. Du solltest dich hier mit dem Thema etwas genauer beschäftigen und erstmal vorhandene Shader modifizieren. Später kannst du dann versuchen, einen eigenen Shader zu schreiben.

Jedoch darfst du nie vergessen, deinen Shader in die Shaderlist.txt einzutragen - da sonst der Shader nicht geladen wird. Das kannst du leicht erkennen, da dann die Textur nicht weiss umrandet ist.

[zurück zur Hauptseite](#)





Multiplayerziel: Deathmatch-Skript

Beispielmap: "tutor68.map"
Ergebnissmap: "tutor69.map"

In diesem Thema will ich dir zeigen, wie man aus seiner Map eine schicke Deathmatch-Map machen kann. Dies geht über ein Skript, wie alle anderen Missionsziele übrigens auch. Wir fangen aber mit dem Deathmatch-Ziel an, da das Skript dafür sehr einfach zu verstehen ist.

Daher will ich dir gleich das Skript vorstellen:

```
game_manager
{
    spawn
    {
        wm_mapdescription "eine kleine Deathmatch-Map"

        wm_allied_respawntime 30
        wm_axis_respawntime 30

        wm_set_round_timelimit 10

        wm_number_of_objectives 0
    }
}
```

Das ganze kommt dir vielleicht von den Shaders bekannt vor. Und tatsächlich - dieser Vergleich ist richtig. So kannst du hier auch z.B. die zwei Slashes (//) benutzen, um z.B. einen Teil des Codes zu kommentieren.

Nun kann ich dir gleichmal Mut zusprechen und sagen, dass das auch schon alles war. Nun will ich dir aber noch erklären, was das alles bedeutet:

game_manager	Der Anfang des Skripts
{	
spawn	der Anfang des Spawn-Blocks. Entspricht einem "Stage" bei den Shaders.
{	
wm_mapdescription "eine kleine Deathmatch-Map"	eine kurze Beschreibung der Map
wm_allied_respawntime 30	die Zeit, wie lange es dauert, bis die Allies wieder belebt werden
wm_axis_respawntime 30	die Zeit, wie lange es dauert, bis die Axis wieder belebt werden

<pre>wm_set_round_timelimit 10</pre>	Gibt die Zeit an, wie lang eine Runde maximal dauern darf
<pre>wm_number_of_objectives 0</pre>	Gibt die Anzahl der Objectives (Spielziele) an. In diesem Fall 0, da nur Bombenziele und Dokumente als Objective zählen.
<pre>}</pre>	Ende des Spawn-Blocks
<pre>}</pre>	Ende des Skripts
<pre>// kein Kommentar</pre>	Dies ist ein Kommentar, der "kein Kommentar" enthält

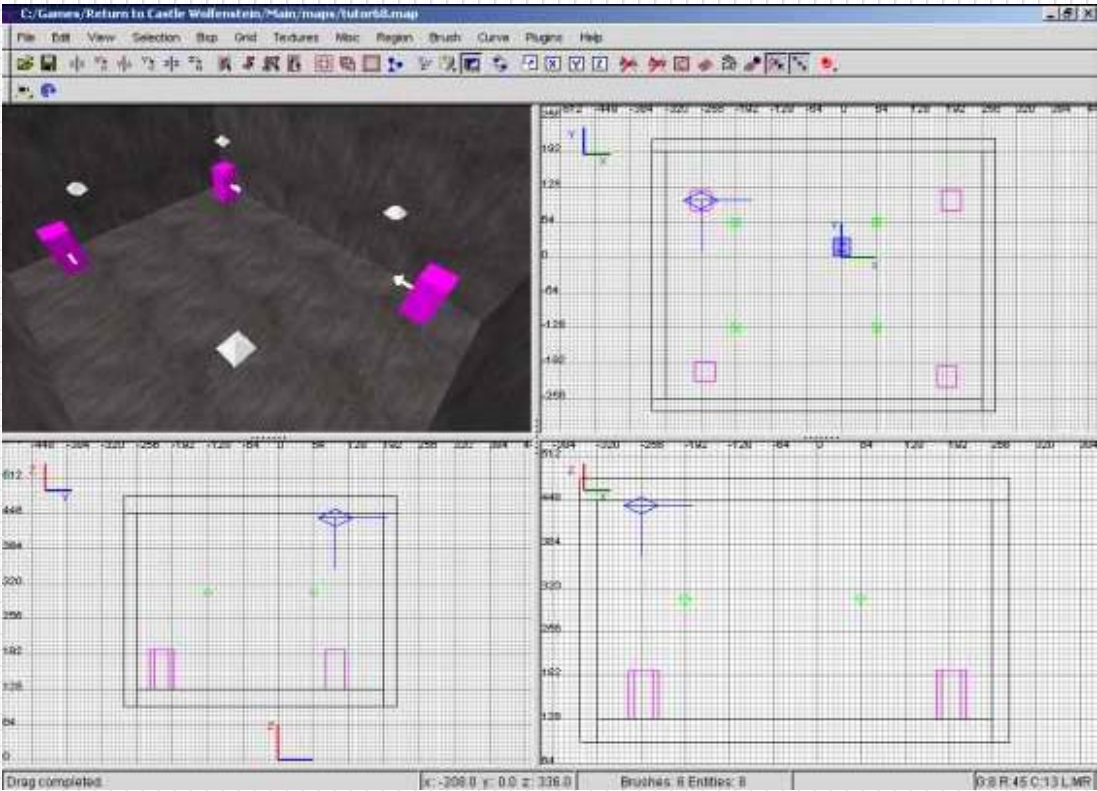
Ich habe dir dieses Skript schon vorbereitet, falls du es nichtmehr in eine Textdatei einfügen willst. Das ganze geht aber recht einfach, einfach ein Textprogramm öffnen, den ganzen Text einfügen, und als "deinname.script" abspeichern. Solltest du dir bei der Handhabe nicht sicher sein, findest du hier die [fertige Version](#).

Nun musst du eigentlich nur noch folgendes beachten, bevor wir uns der Map selbst widmen:

- Das Skript muss mit der compilierten Map zusammen im "Maps"- Ordner liegen
- Das Skript muss vom Namen her genau identisch mit der compilierten Map sein.

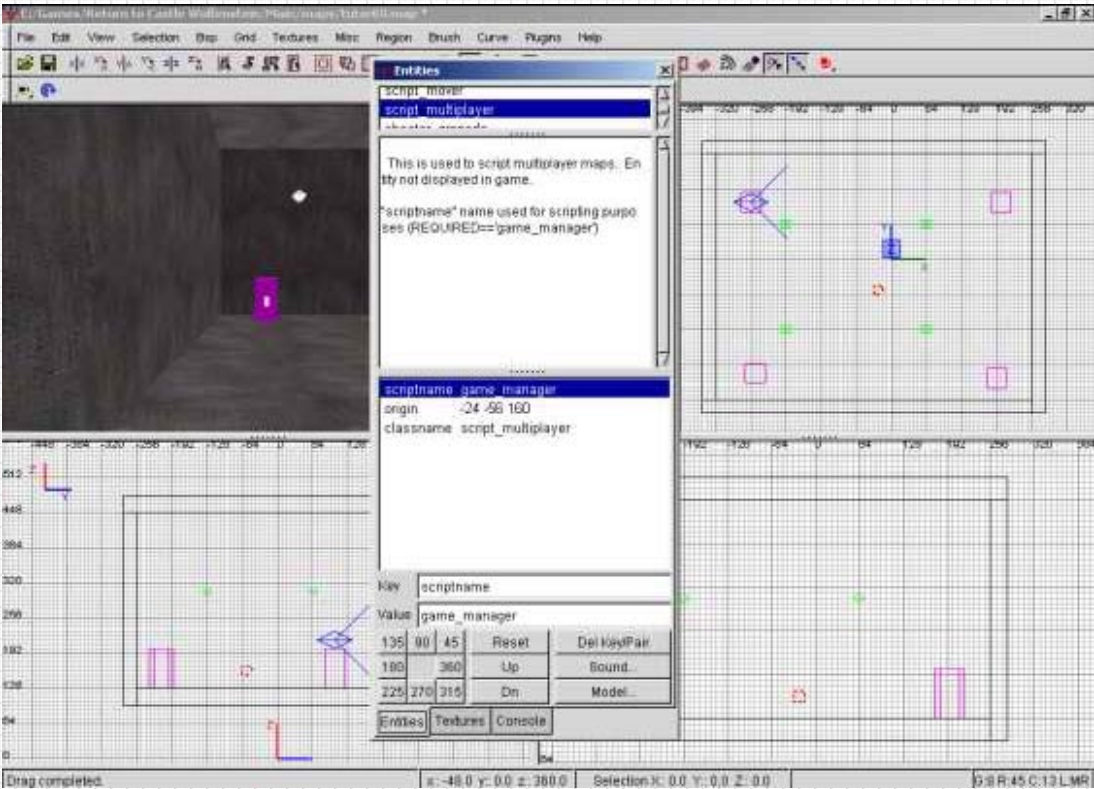
heisst deine Map z.B. **"hossa-die-waldfee.bsp"**, so heisst dann dein Skript **"hossa-die-waldfee.script"**. Heisst deine Map **"meinversuch.bsp"**, so heisst dein Skript **"meinversuch.script"**. Um dem ganzen noch ein Beispiel zu geben, habe ich in der Zip-Datei die Map "tutor69.bsp" genannt, das Skript folglich "tutor69.script".

Nun öffnest du den Radianten und erstellst eine kleine Deathmatch-Map. Sie kann auch nur aus einem Rau bestehen, das ist egal. Meine sieht z.B. so aus:



Kommen wir nun zum eigentlichen. Dazu deselektierst du alles, in dem du die "ESC"-Taste drückst. Nun klickst du 2 x mit der rechten Maustaste, und wählst im Menü "script" und dort als Unterpunkt "script_multiplayer". Es erscheint ein kleiner, pinkfarbiger Kasten. Diesen kannst du überall in der Map hinsetzen. Zur Demonstration setze ich ihn allerdings mal voll in die Mitte.

Nun drückst du die Taste "N", um das Entity-Fenster zu öffnen:



Hier gibst du folgendes ein:

- Key: "scriptname"
Value: "game_manager"

So, damit hätten wir schon alles geschafft - jetzt kannst du einfach mal deine Map compilieren und deine Map ansehen.

[zurück zur Hauptseite](#)

183759



Türen auf Knopfdruck öffnen:

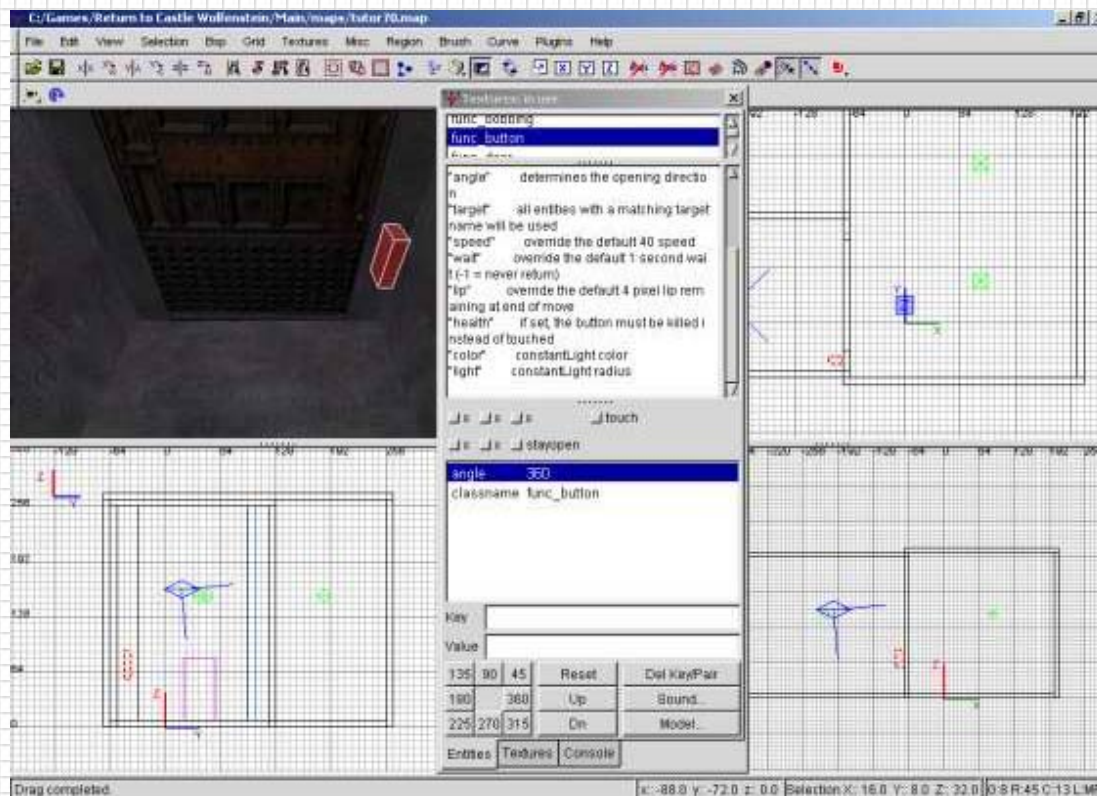
Beispielmap: "tutor70.map"

Ergebnissmap: "tutor71.map"

Hier will ich dir erklären, wie man eine Türe erstellt, die nur zu öffnen ist, wenn sie durch einen Knopf entriegelt wurde.

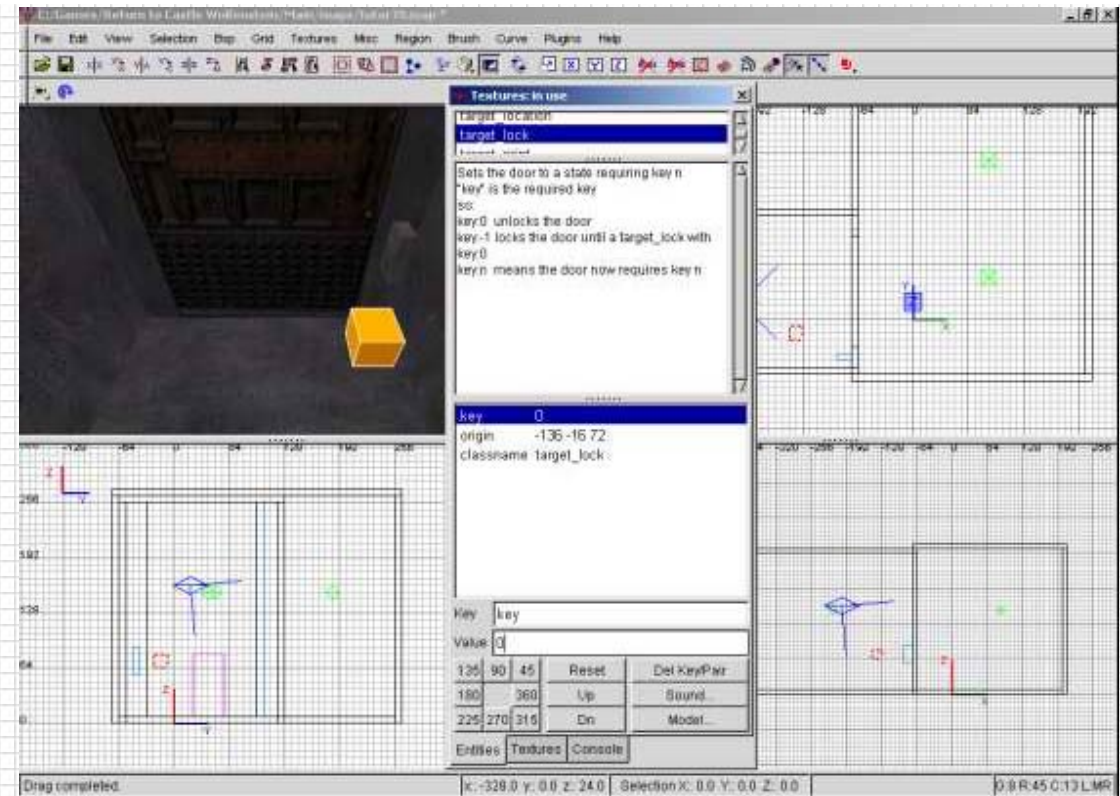
Zuerst benötigen wir dazu natürlich eine Map mit einer Türe - natürlich kannst du dir diese auch selbst bauen, aber ich habe dir natürlich wie immer schon alles vorbereitet, so dass du dich auf das wesentliche konzentrieren kannst.

Zuerst bauen wir uns einen Brush, den du mit einer normalen Textur belegst - er soll schliesslich zu unserem Knopf werden und man soll ihn auch gut sehen können. Dann klickst du 2 mal mit der rechten Maustaste und wählst "func" und als Unterpunkt "func_button". Nun drückst du die Taste "N", um das Entity-Fenster zu öffnen:



Wie du sehen kannst, habe ich als "angle" einen Wert von 360° angegeben, das bedeutet in unserem Fall, der Knopf bewegt sich in die Wand hinein. Das kannst du erkennen, indem du dir ganz unten im Entity-Menü das Kreuz mit den 8 Zahlen ansiehst. Du kannst hier genau in diese Richtung drücken, in die sich dein Knopf bewegen soll. In unserem Fall sind es 360°, weil sich der Knopf ja wie gesagt in die Wand bewegen soll.

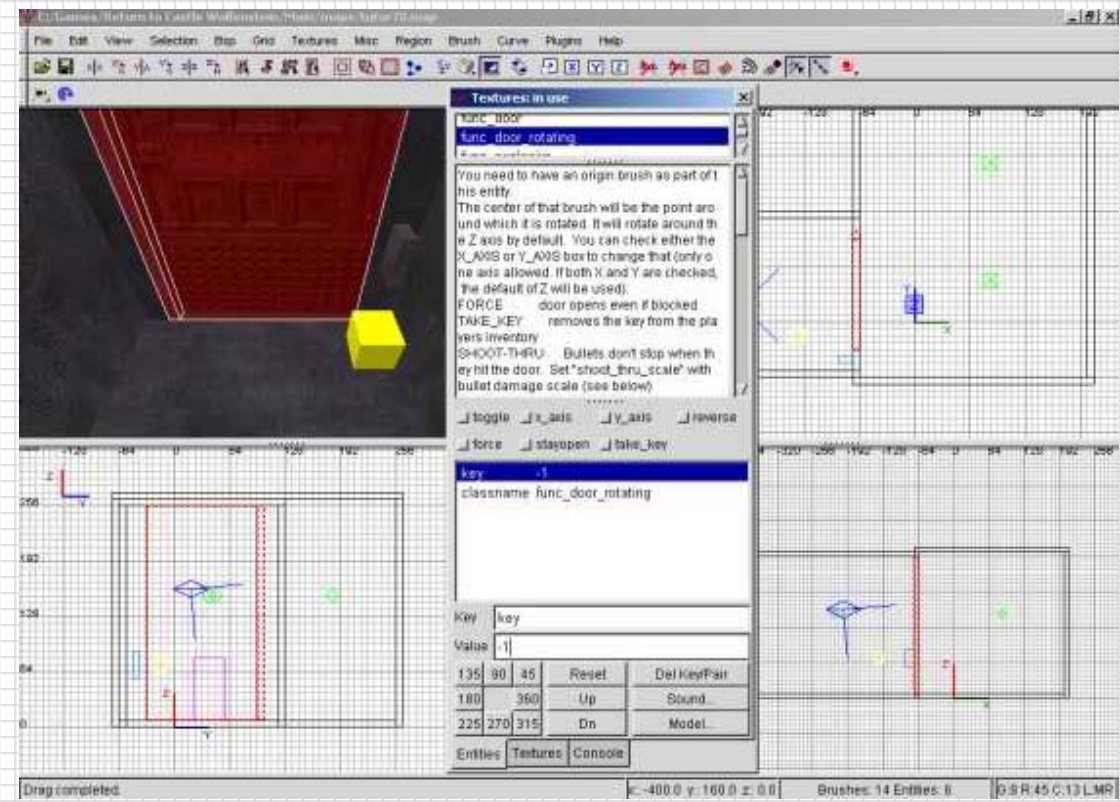
Nun benötigen wir ein "target_lock". Dazu deselektierst du erst einmal alles, indem du die "ESC"-Taste drückst. Nun klickst du 2 x mit der rechten Maustaste und wählst dort "target" und als Unterpunkt "target_lock". Nun erscheint ein gelber Kasten. Nun drückst du gleich wieder die Taste "N", um das Entity-Fenster zu öffnen:



Hier gibst du folgendes ein:

- Key: "key"
Value: "0"

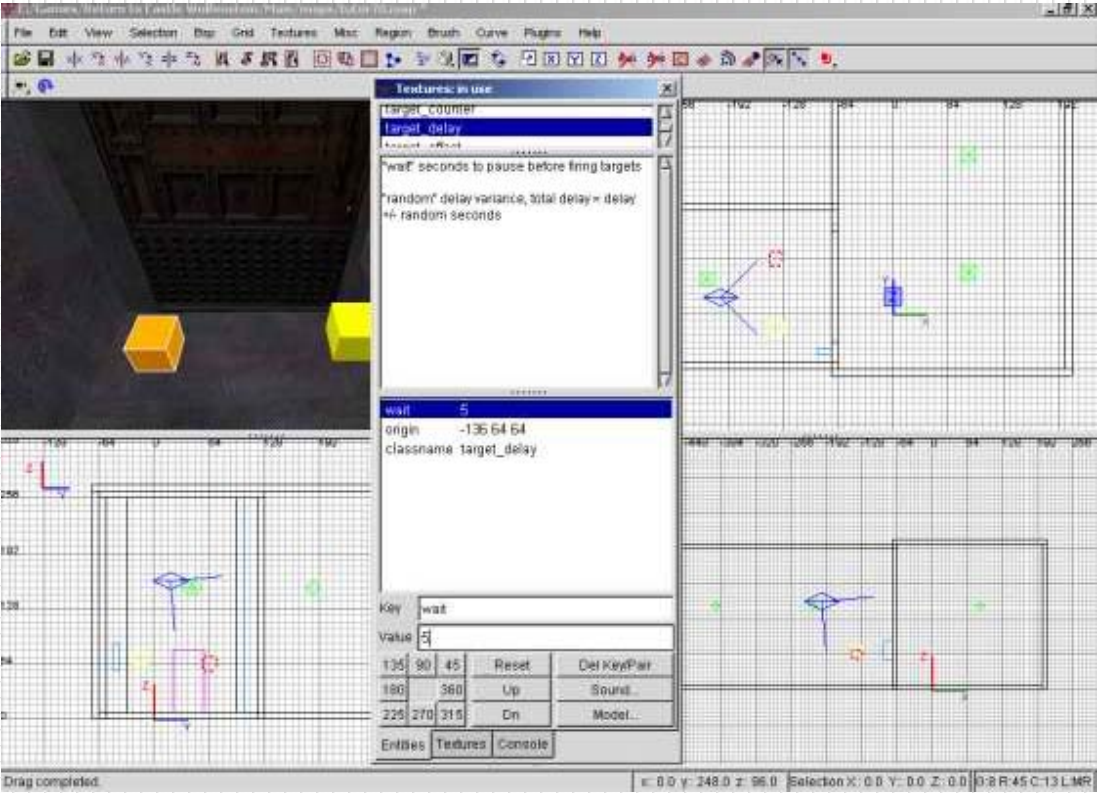
Nicht, dass du denkst, wir machen hier etwas falsch - im Gegenteil, dieser target_Lock wird die Türe später öffnen. Nun drückst du wieder "ESC", um alles zu deselektieren. Jetzt wenden wir uns nämlich der Türe zu. Dazu selektierst du sie und drückst wieder die Taste "N", um das Entity-Fenster zu öffnen:



Hier gibst du folgendes ein:

- Key: "key"
Value: "-1"

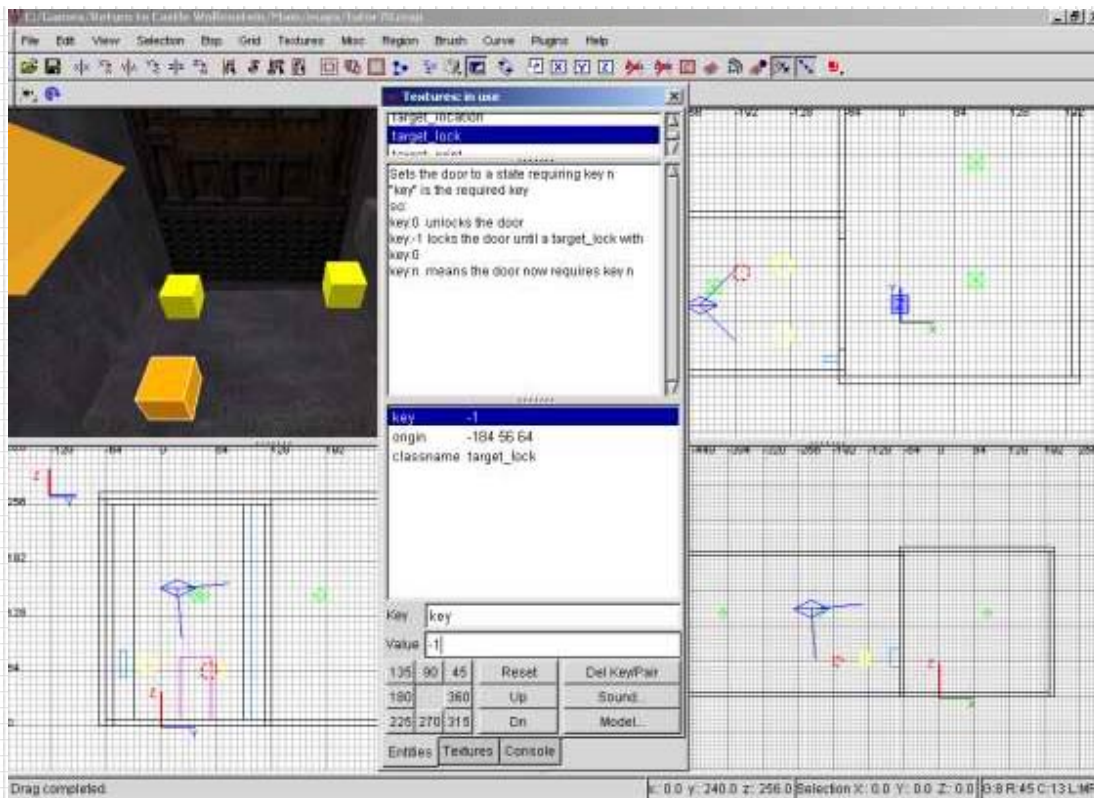
So ,das bedeutet, dass die Türe abgeschlossen ist und sich nur durch unseren Schalter öffnen lässt. Nun benötigen wir noch einen target_delay, der angibt, wie lange die Türe offen stehen soll. Dazu klickst du 2 x mit der rechten Maustaste, wählst dort "target" und als Unterpunkt "target_delay". Nun drücken wir wieder die Taste "N", um das Entity-Fenster zu öffnen:



Hier gibst du folgendes ein:

- Key: "wait"
Value: "5"

Das bedeutet, dass die Türe 5 Sekunden lang offen stehen bleibt, bevor sie sich wieder schliesst. Nun benötigen wir wieder einen "target_lock". Dazu deselektierst du alles. Dann drückst du 2 x mit der rechten Maustaste und wählst "target" und als Unterpunkt "target_lock", danach drückst du gleich wieder die Taste "N", um das Entity-Fenster zu öffnen:

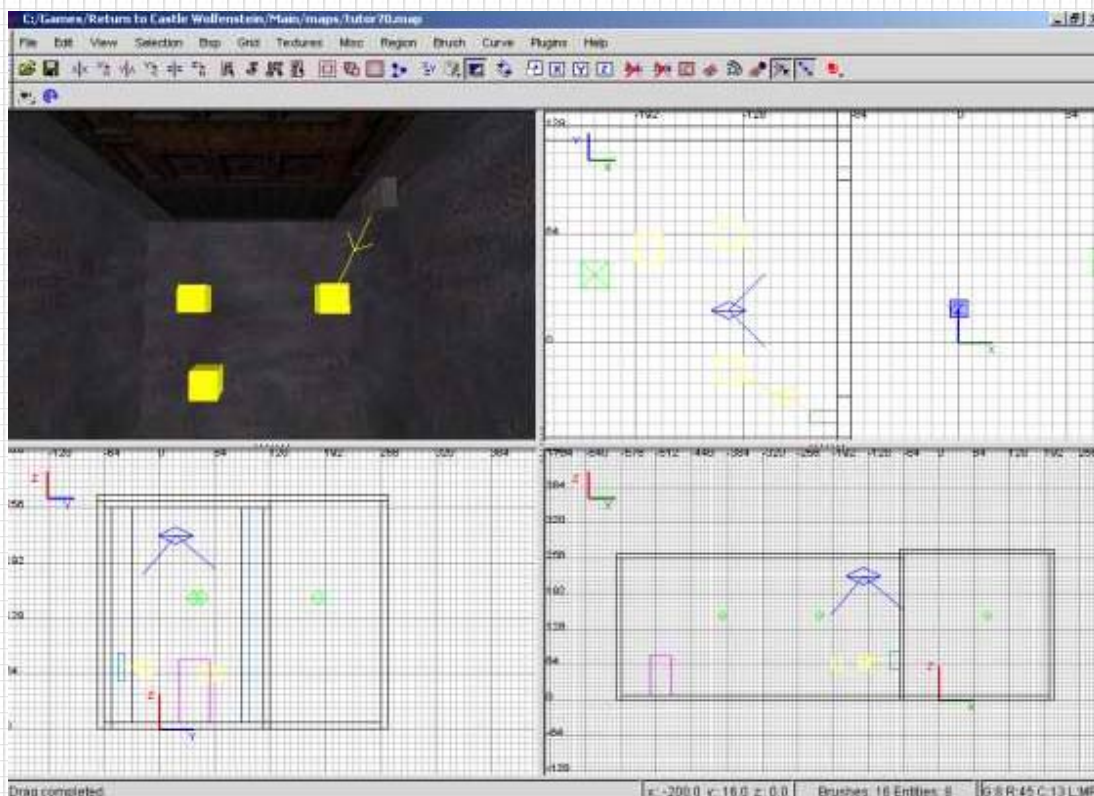


Hier gibst du folgendes ein:

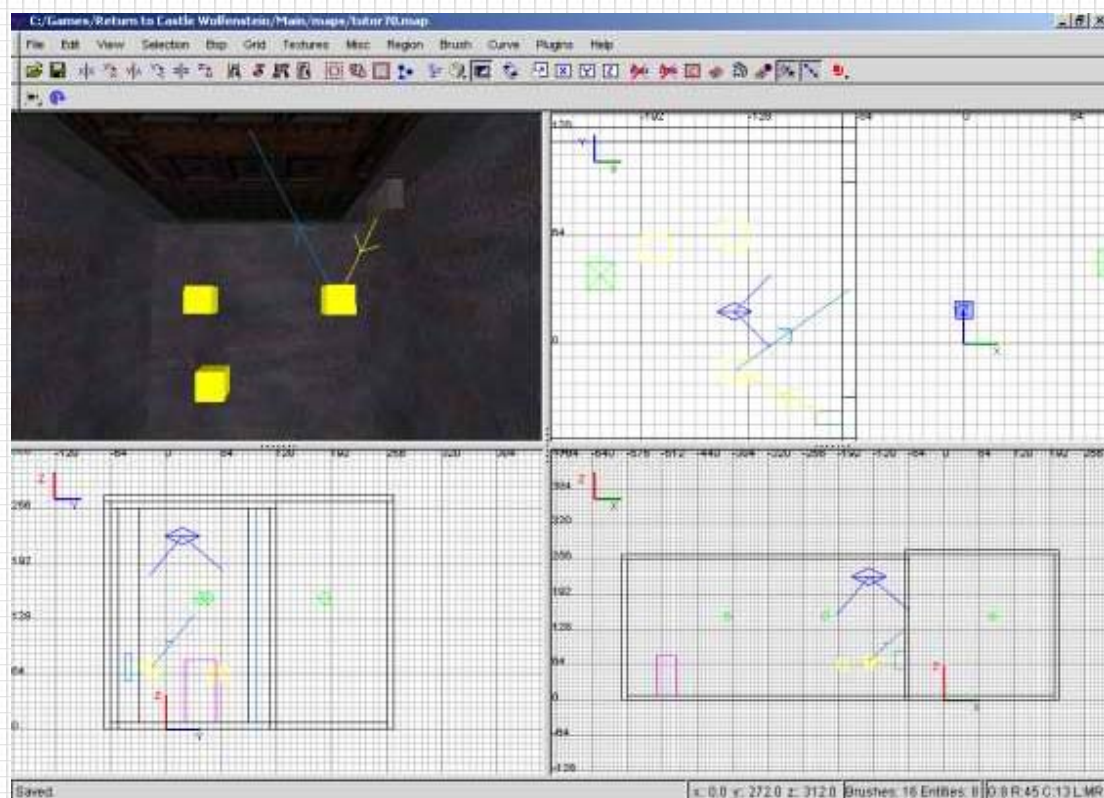
Key: "key"
Value: "-1"

Damit geben wir diesem target_lock den Befehl, die Türe wieder zu schliessen. Jetzt müssen wir die ganzen Entities natürlich Stück für Stück miteinander verbinden. Dazu deselektierst du nochmal alles, in dem du die "ESC"-Taste drückst.

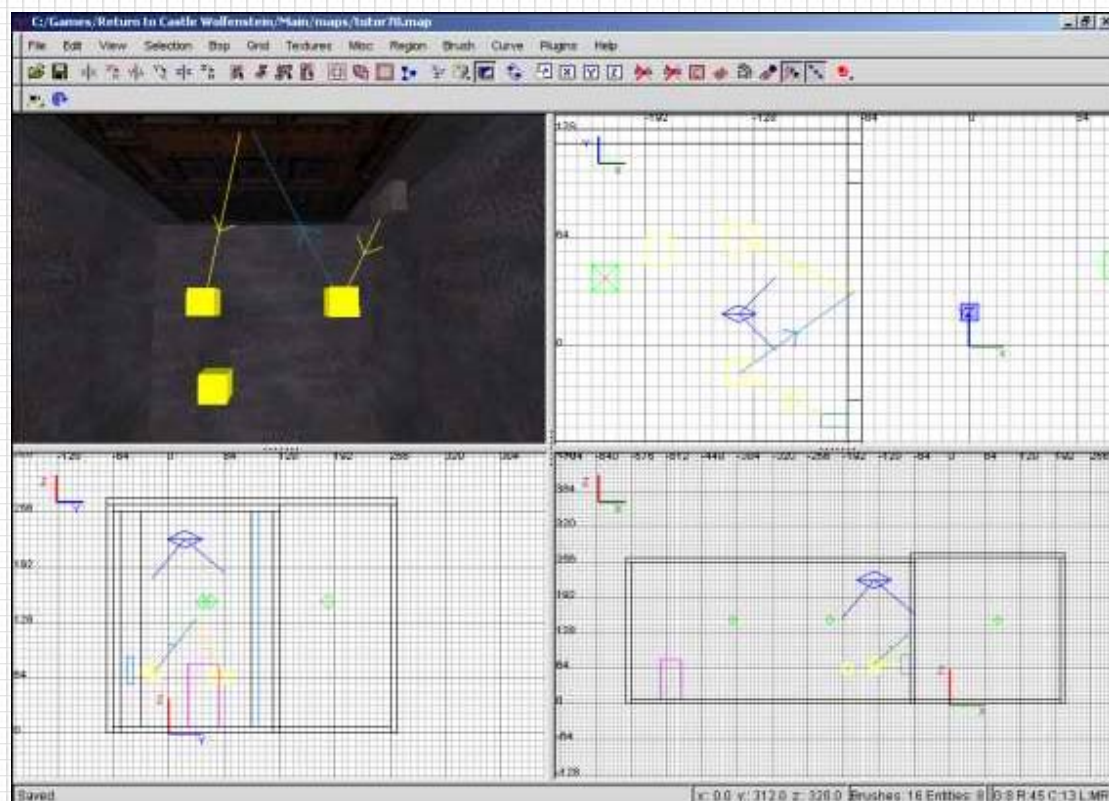
Zuerst selektierst du unseren Knopf, dann das Entity "target_lock", dass wir zuerst gesetzt haben und drückst dann "STRG" + "K":



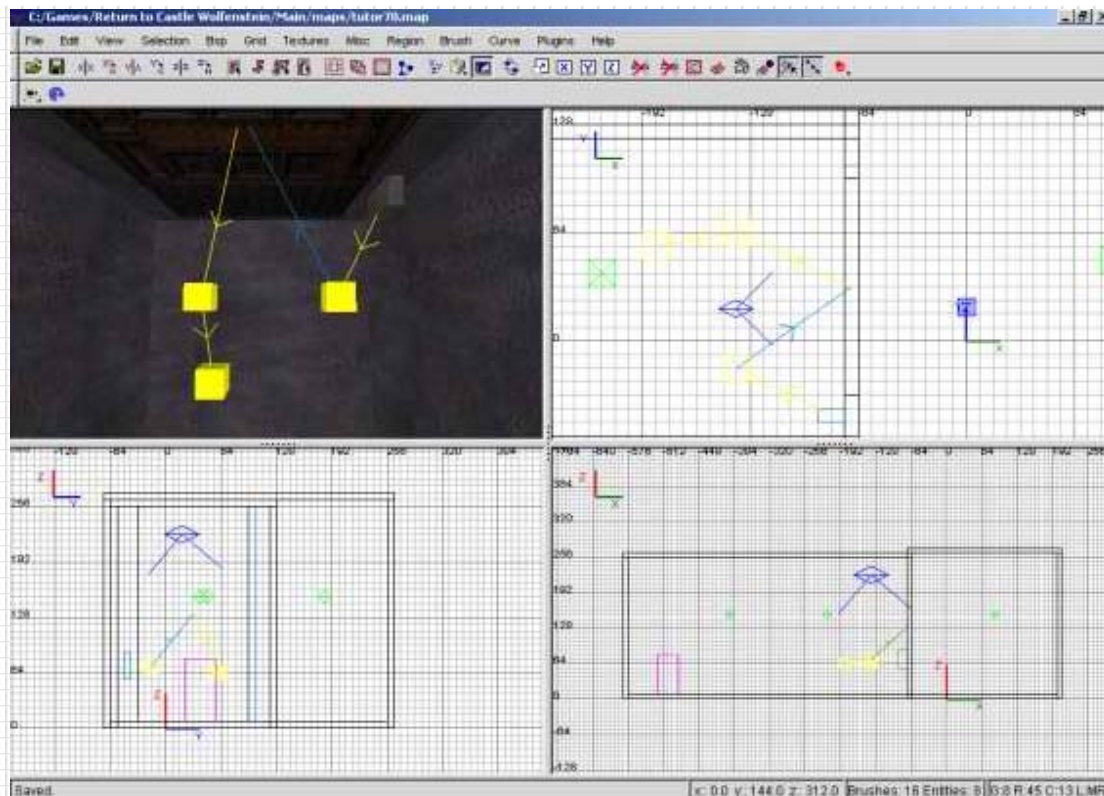
Nun deselektierst du alles. Nun wählst du das "target_lock" von eben an und danach die Türe und drückst wieder "STRG" + "K", um die Entities zu verbinden:



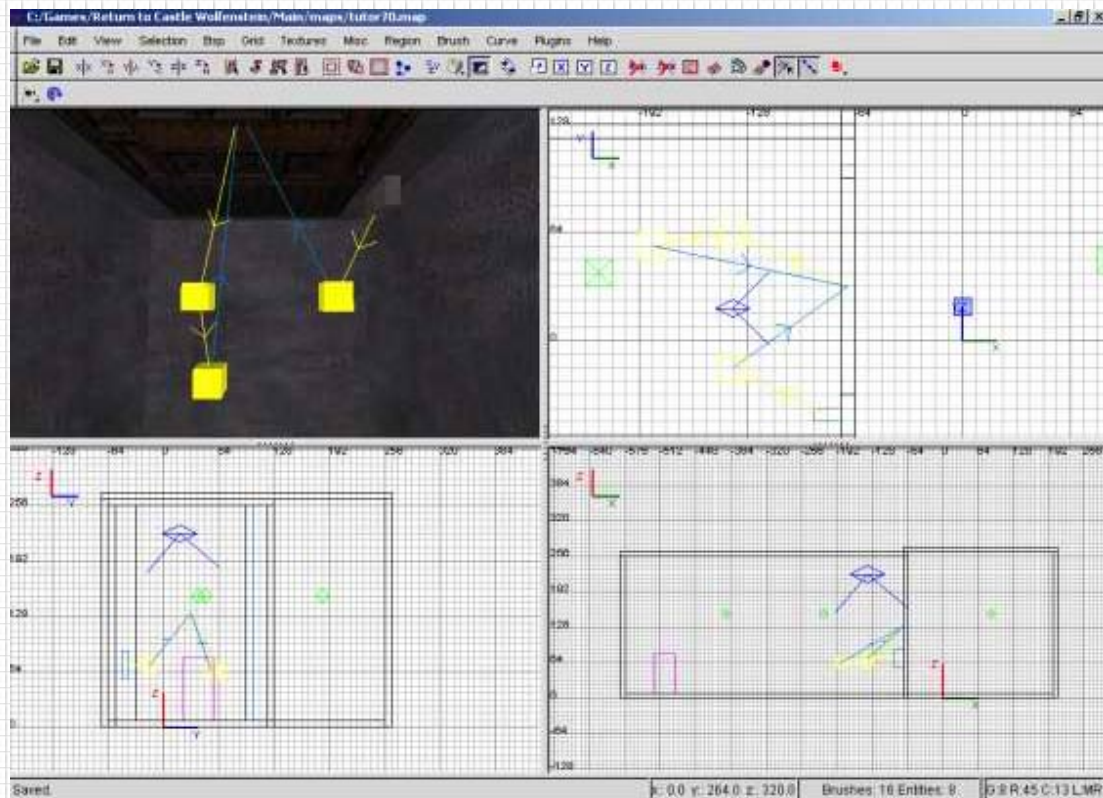
Nun deselektierst du wieder alles und selektierst dann die Türe, danach das "target_delay". Nun drückst du wieder "STRG" + "K". Dann sollte es so aussehen:



Nun deselektierst du wieder alles und selektierst zuerst das "target_delay" und danach das "target_lock". Nun drückst du wieder "STRG" + "K". Dann sollte es so aussehen:



Nun deselektierst du wieder alles und selektierst zuerst das "target_lock" und danach die Türe. Nun drückst du wieder "STRG" + "K". Dann sollte es so aussehen:



So, das hätten wir geschafft - nun will ich dir noch erklären, wieso wir das ganze so umständlich verknüpft haben. Also: Der Spieler bedient den Schalter -> der Schalter betätigt das target_lock -> das Target_Lock öffnet die Türe -> die Türe aktiviert das Target:_Delay und bleibt so für 5 Sekunden offen stehen -> das target_delay aktiviert das target_lock -> und das target_lock sagt der Türe, dass sie nun schliessen soll.

[zurück zur Hauptseite](#)

183759

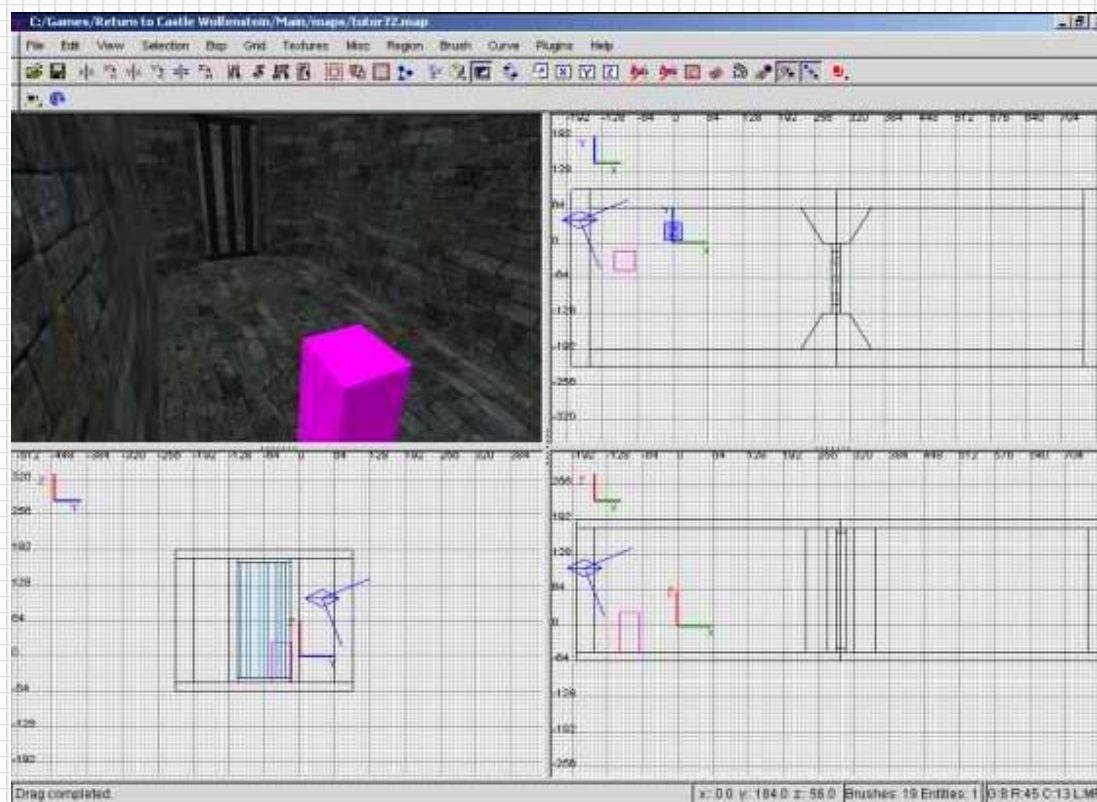


Türen erstellen, die sich nur über einen Knopf öffnen lassen:

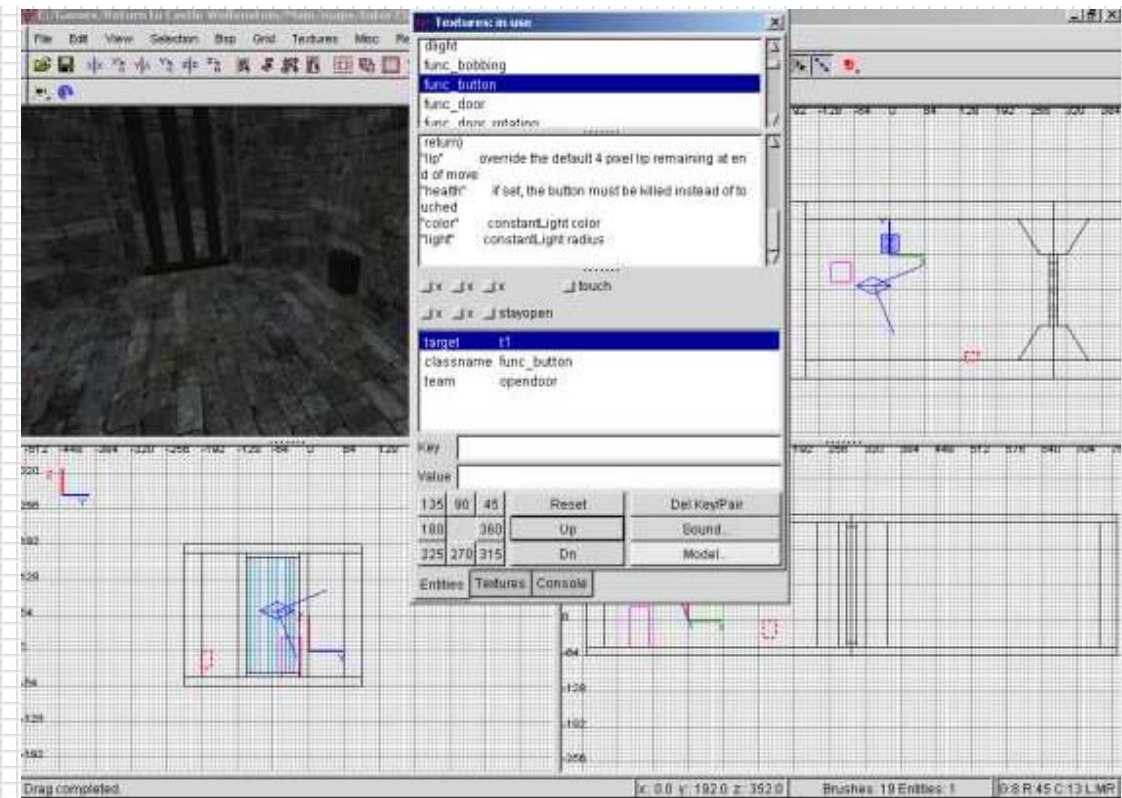
Beispielmap: "tutor72.map"

Ergebnissmap: "tutor73.map"

So, nachdem wir nun schon wirklich viel über Türen gelernt haben, wollen wir nun eine Türe bauen, die sich nur über einen Knopfdruck öffnen lässt. Dazu benötigen wir natürlich einen Raum und eine Türe - in der Beispielmap habe ich dir bereits alles hineingebaut, um sofort loszulegen:



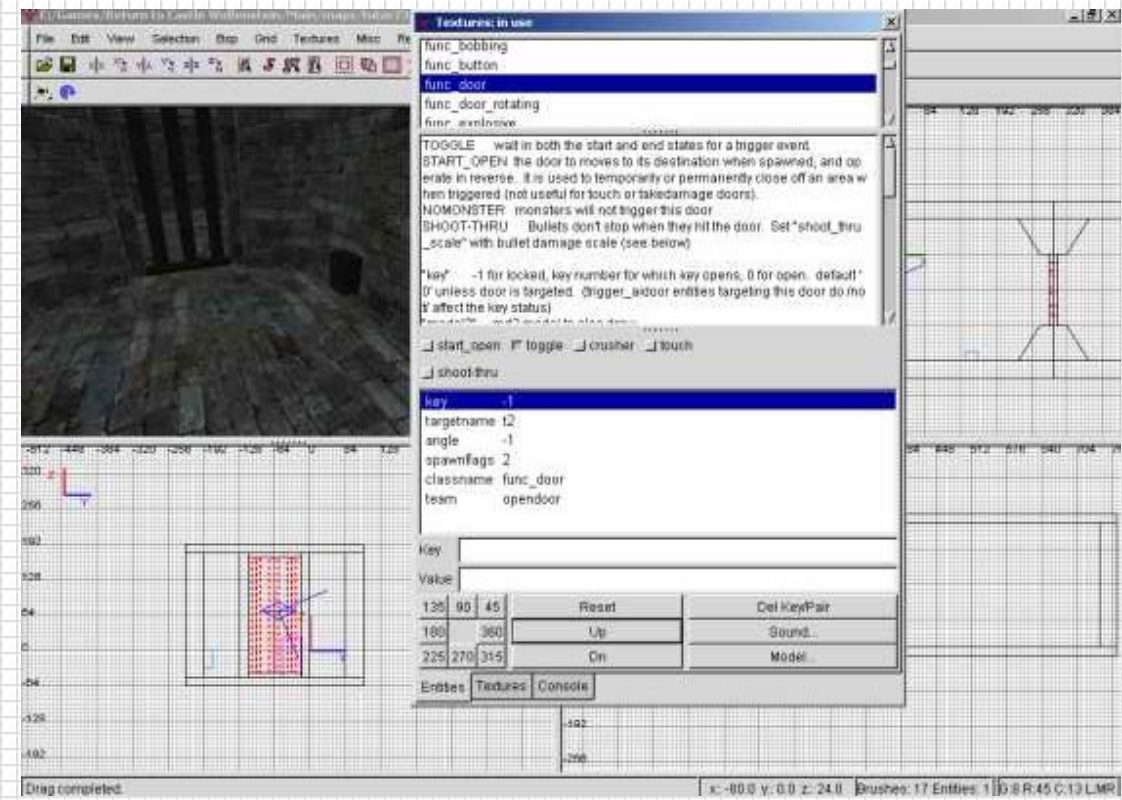
Zuerst benötigen wir natürlich einen Schalter, der zum Schluss die Türe öffnet. Dazu erstellst du jetzt einen Brush mit einer beliebigen Textur. Diesen machen wir nun zu einem Schalter, in dem du 2 x mit der rechten Maustaste drückst und im Menü "func" und dort als Unterpunkt "func_button" anwählst. Nun drückst du die Taste "N", um das Entity-Fenster zu öffnen:



Hier gibst du folgendes ein:

- Key: "team"
Value: "opendoor"

und vergiss nicht, die Eingabe mit der ENTER-Taste zu bestätigen. Nun drückst du die Taste "ESC" und wählst die Türe an. Hier drückst du wieder auf die "N"-Taste.



Nun gibst du hier folgendes ein:

- Key: "team"
Value: "opendoor"
- Key: "key"
Value: "-1"

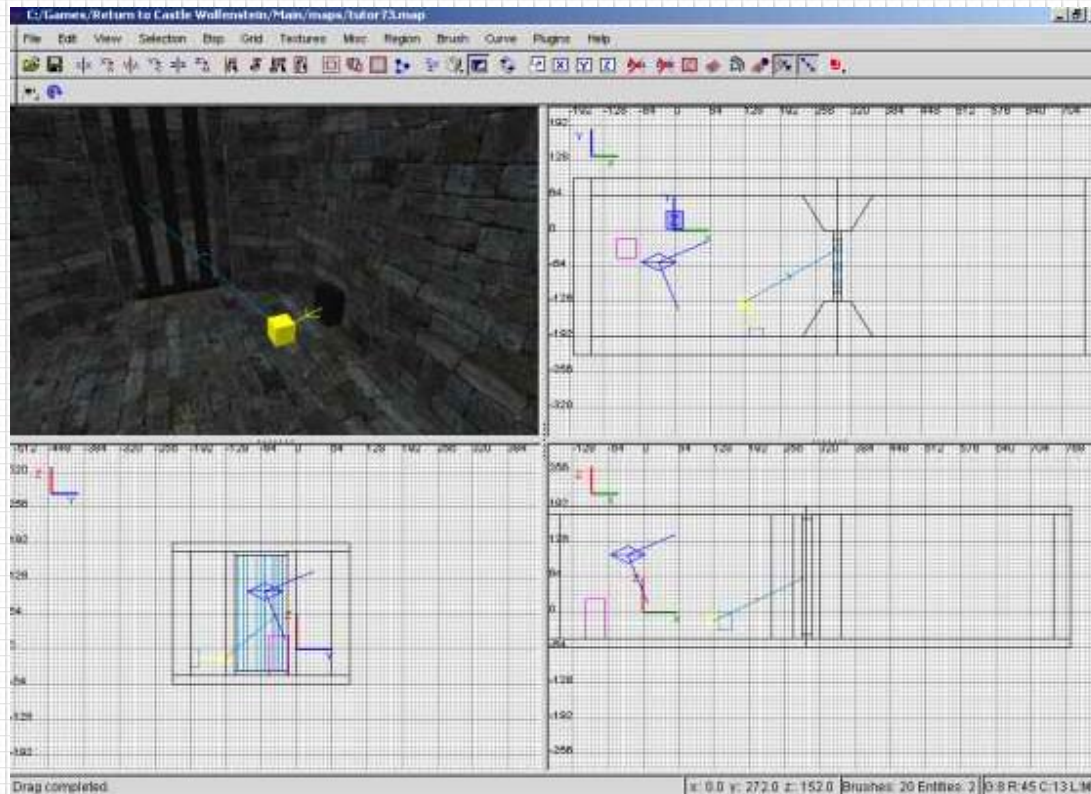
ausserdem drückst du noch die "Up"-Taste sowie die Taste "toggle".

Und was soll das? Ganz einfach, mit der Team-Vergabe werden beide Entities ausgelöst. Und mit der Variablen "-1" schliesst du die Türe ab. Nun benötigen wir nur noch ein Entity, um die Türe aufzuschliessen.

Dazu benötigen wir einen target_lock. Dazu klickst du 2 x mit der rechten Maustaste, und wählst "target" und als Unterpunkt "target_lock". Nun drückst du wieder auf die "N"-Taste. Nun gibst du folgendes ein:

- Key: "key"
Value: "0"

mit diesem Befehl wird die Türe über das target_lock aufgeschliessen. Nun müssen wir die Entities nur noch verbinden. Dazu deselektierst du zuerst einmal alles. Nun wählst du zuerst den func_button und dann den target_lock an. Nun drückst du die Taste "STRG" + "K". Nun deselektierst du wieder alles. Nun wählst du den target_lock und danach die Türe an. Nun drückst du nochmals die Taste "STRG" + "K". So, das hätten wir geschafft!



Vielleicht fragst du dich jetzt, wieso wir nicht noch den func_button direkt mit der Türe verbinden. Das brauchen wir deshalb nicht, weil die Türe bereits über die Team-Zugehörigkeit mit dem func_button verbunden ist.

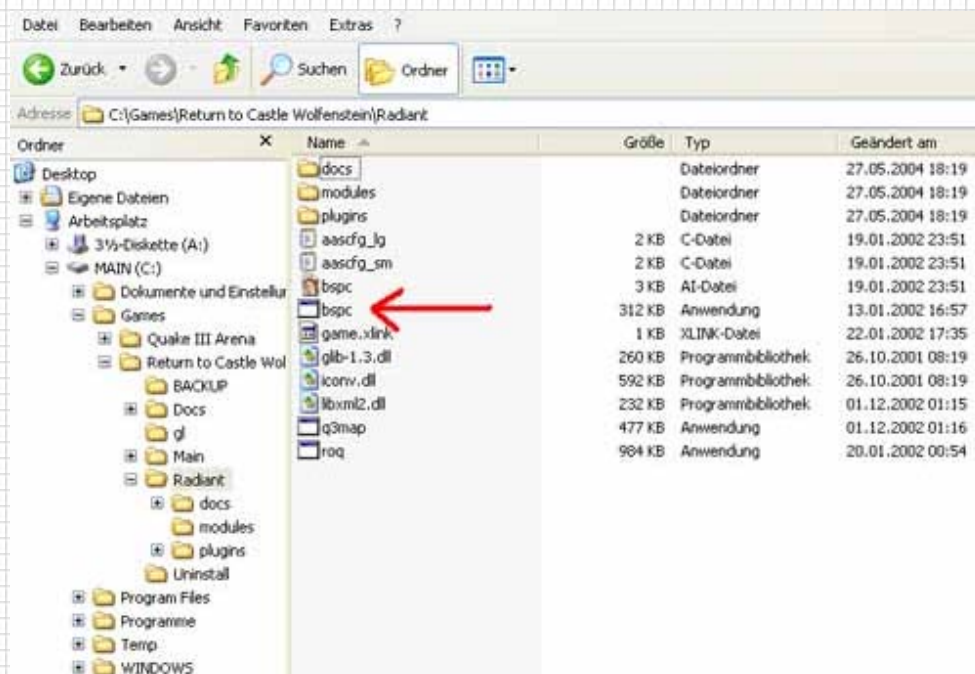
[zurück zur Hauptseite](#)

183759



*.bsp zu *.map konvertieren:

Hier will ich dir zeigen, wie man aus einer bestehenden Map (z.B. Test.bsp) das Map-File (Test.map) konvertieren kann. Dazu benötigt man das Tool "bspc" - dieses befindet sich netterweise gleich im jeweiligen "Radiant"-Verzeichnis im Spiel-Ordner:



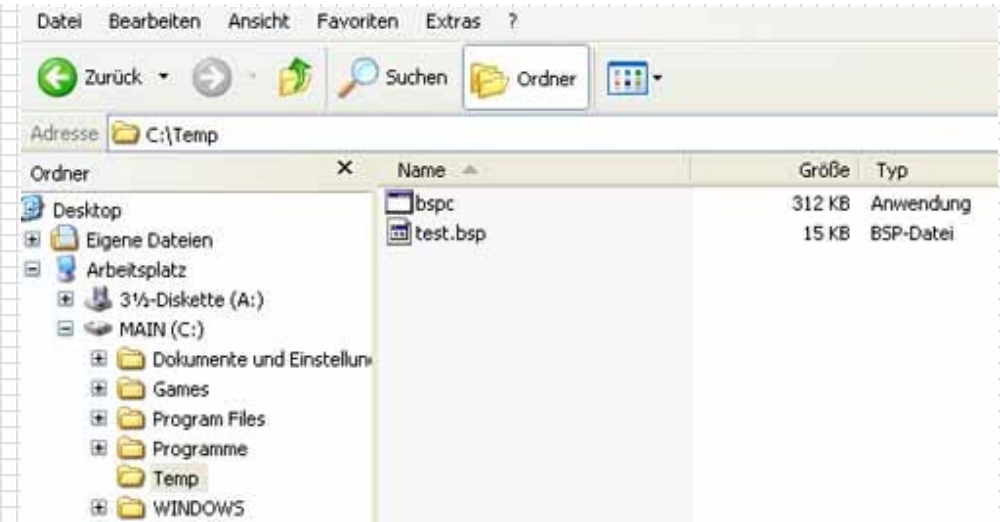
So z.B. bei Return To Castle Wolfenstein befindet sich die Datei hier:

- ...\\Return To Castle Wolfenstein\\Radiant\\bspc.exe

bei Quake III Arena befindet sich die Datei bspc.exe hier:

- ...\\Quake III Arena\\Radiant\\bspc.exe

Eigentlich könnten wir auch die Datei hier lassen und jetzt sofort eine Map konvertieren, jedoch müssen wir später etwas in der DOS-Box tippen, daher erstellen wir nun einen Ordner auf C:\\, ich nenne ihn "temp". Dort hinein kopieren wir nun die Datei "bspc.exe" und eine *.bsp Datei eurer Wahl. Nun sieht es so aus:



Nun öffnen wir die DOS-Box, nun müsste es so aussehen:

C:\Windows\

und geben nun ein:

"cd.." (ohne die Anführungszeichen) -> nun wechseln wir auf das Laufwerk "C:" dann müsste folgendes da stehen:

WICHTIG: Steht bei euch schon "C:\", dann könnt ihr euch diesen Schritt sparen!

C:\

Nun gebt ihr weiter ein:

"Cd temp" -> damit wechseln wir in das Verzeichnis "Temp" jetzt steht da:

"C:\temp" hier gebt ihr jetzt diese Zeile ein:

bspc -bsp2map C:\temp\test.bsp	
-bsp2map	der Befehl, um die *.bsp in eine *.map Datei umzuwandeln
C:\temp\	der gesamte Pfad zur *.bsp

WICHTIG: Hätten wir vorhin die Datei im Ordner "...\Return To Castle Wolfenstein\Radiant\" gelassen, müsste der Befehl jetzt so aussehen:

C:\...\Return To Castle Wolfenstein\Radiant\bspc -bsp2map c:\temp\test.bsp

Nun wird die *.map Datei errechnet - das recht recht fix, hier in meiner Bsp-Datei dauerte der Vorgang ganze 0 Sekunden. Da das ganze jetzt etwas schnell aufeinander ging, hier nochmal ein Bild mit der DOS-Box, wo ihr nochmal alles sehen könnt:

```
C:\>cd temp

C:\Temp>bspc -bsp2map c:\temp\test.bsp
Opened log bspc.log
BSPC version 2.1c by Mr Elusive
GtkRadiant 1.2.1-nightly Jan 13 2002
bsp2map: c:\temp\test.bsp to test.map
-- Q3_LoadMapFromBSP --
Loading map from c:\temp\test.bsp...
creating planar surface planes...
searching visible brush sides...
    150 brush sides
    51 brush sides textured out of 150
nummapbrushsides = 150
    0 curve brushes
writing test.map
written 22 brushes
map file written in      0 seconds
Closed log bspc.log

C:\Temp>
```

Nun müsst ihr nur noch die Datei test.map in eueren normalen "maps"-Ordner von RtCW oder Q3 kopieren und nun könnt ihr die Map im Radiant ansehen. Aber nicht erschrecken, wenn im Radiant die Texturen nicht angezeigt, werden das ist normal. Also müsst ihr ggf. Brushes selbst texturieren, wenn ihr Teile der Map übernehmen wollt. Jedoch solltet ihr beachten nicht zu klauen, sondern eigene Ideen in eure Maps einzuarbeiten.

[zurück zur Hauptseite](#)



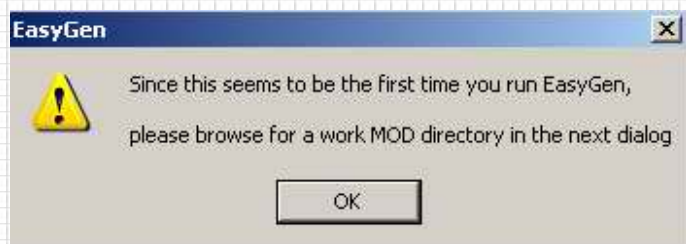
EasyGen Installation:

Hier will ich dir erklären, wie man schöne Aussenlandschaften erstellen kann. Das ganze ist recht leicht - dazu benötigt man ein Tool Namens "EasyGen". Hier gehts zur [EasyGen-Seite](#), wo du auch den Download findest.

Zuerst kommen wir zur Installation:

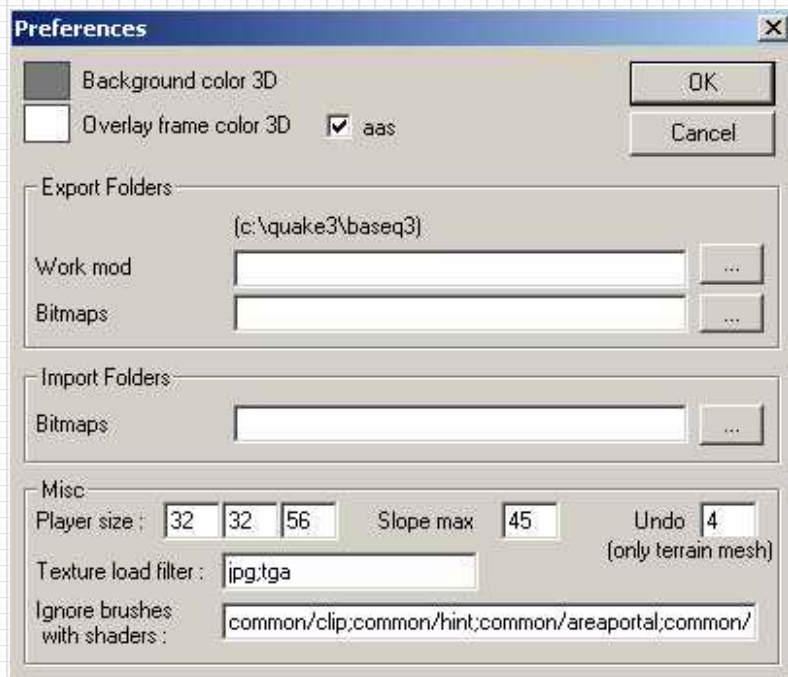
Du downloadest dir das Zip-File von der Seite und entpackst es in ein Verzeichniss nach deiner Wahl - ich entpacke es ins Verzeichniss C:\tmp\easygen\

Nach dem Entpacken startest du die Datei "EasyGen.exe". Darauf taucht auch sofort ein Fenster auf:



Hier wirst du freundlich darauf aufmerksam gemacht, dass du EasyGen warscheinlich zum ersten Mal startest und im nächsten Dialog ein paar Einstellungen tätigen sollst. Dazu klickst du natürlich auf "OK".

Nun erscheint folgenes Fenster:



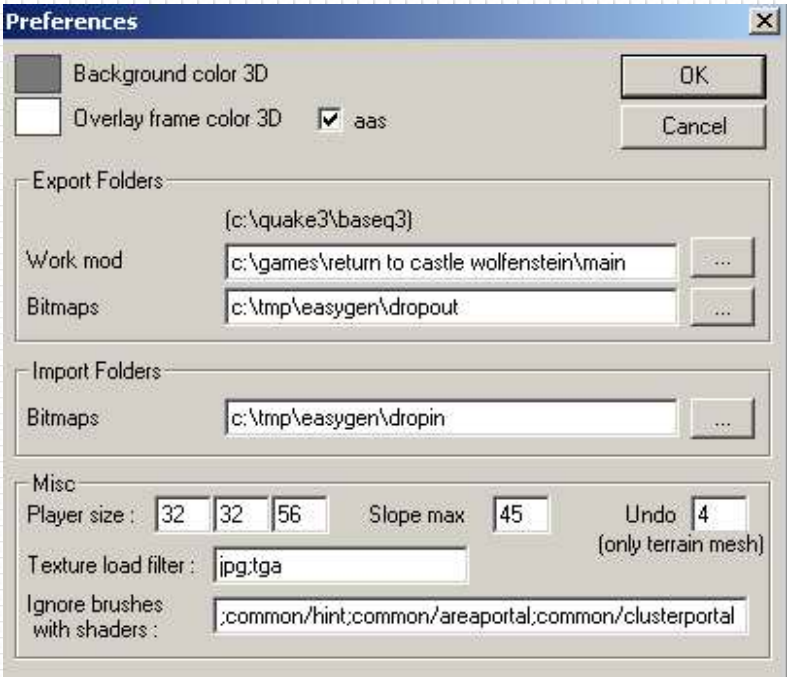
Hier kannst du jeweils das Spiel eintragen, für welches du ein Terrain erstellen willst. Wir erstellen ein Terrain für RtCW ,also klickst du in der Zeile "Work Mod" auf die drei Punkte und suchst dort nach dem "Main"-Ordner von RtCW.

Nun erkläre ich dir schnell, was du bei den anderen Zeilen angeben solltest: Rechts siehst du die fertigen Preferences:

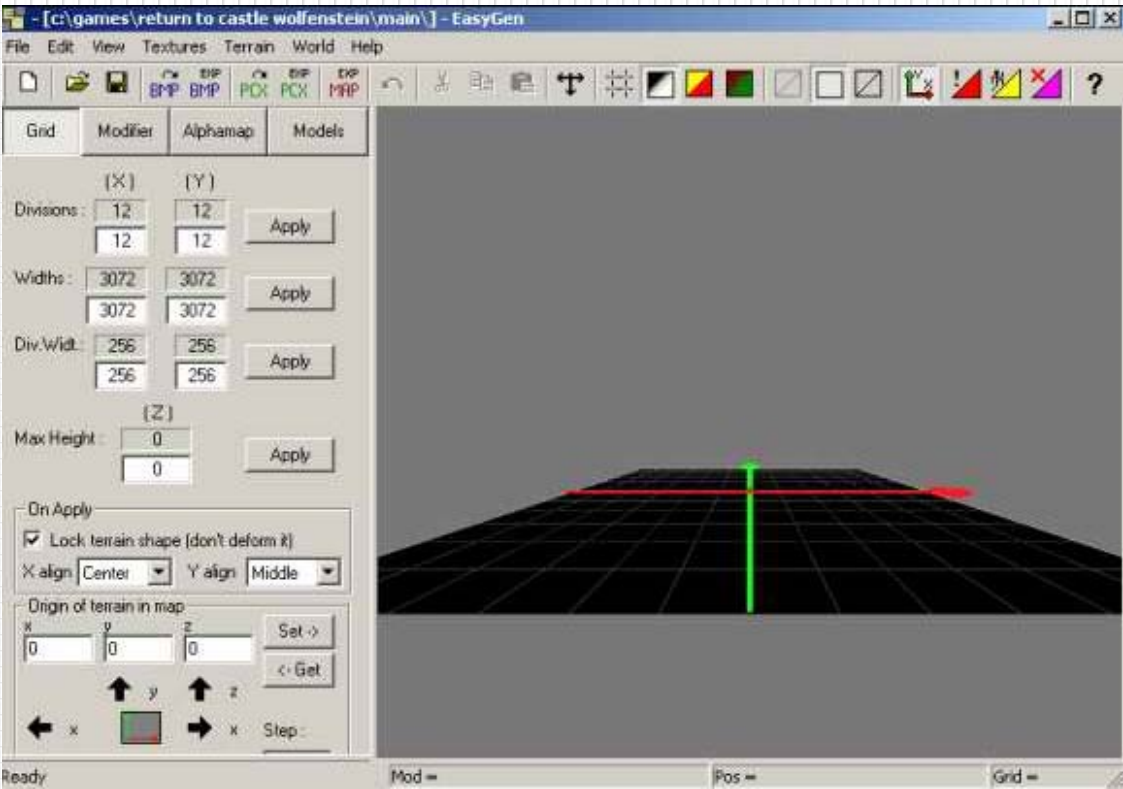
Bitmaps: EasyGen erstellt eine Alphamap, d.h. ein kleines Bild mit dem Farbindex für dein späteres Terrain. Dieser Pfad gibt an, wohin diese Alphamap gespeichert wird. Ich wähle als Ordner "c:\tmp\easygen\dropout"

Unter "Import Folders" gibt es nochmals eine Zeile, in der du einen Pfad zu einem Bitmap angeben sollst. Von hier aus kannst du dann später Graustufenbilder importieren.

Den Rest lassen wir so, wie er ist. EasyGen trägt hier schon alle nötigen Einstellungen ein.



Nun drückst du auf "OK" und EasyGen startet:



Damit hättest du EasyGen erfolgreich auf deiner Festplatte installiert!

[zurück zum Abschnitt EasyGen](#)

[zurück zur Hauptseite](#)



Graustufenbild:

Nachdem wir EasyGen erfolgreich installiert haben, wollen wir gleichmal ein paar Dinge ausprobieren. Doch dazu benötigen wir zuerst ein Graustufenbild.

Dazu kannst du alle gängigen Paint-Programme verwenden. Ich benutze dafür Photoshop. Diese Einstellungen weichen zwar von Programm zu Programm etwas ab, aber dennoch solltest du die nötigen Einstellungen selbst recht schnell finden. Ich klicke dazu auf "Datei" und dort auf "Neu" um ein neues Bild zu erstellen:

Name: Graustufenbild

Der Name ist total egal, du kannst auch einen anderen Namen benutzen.

Breite und Höhe: 32 Pixel

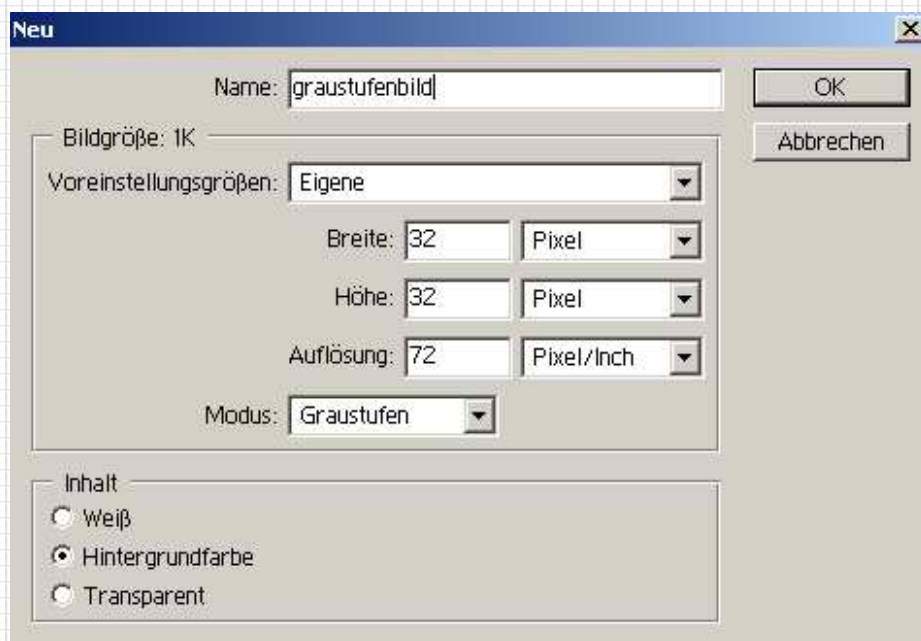
Diese Einstellung solltest du benutzen - sie reicht völlig für unsere Zwecke aus.

Modus: Graustufen

Diese Einstellung ist die wichtigste - ohne Graustufen kein Graustufenbild.

Inhalt: Hintergrundfarbe

Diese Einstellung ist bei Photoshop nötig, um einen schwarzen Hintergrund zu erzeugen. Dazu muss natürlich schwarz die Hintergrundfarbe sein.



Der Sinn an diesem Graustufenbild ist sehr einfach - du kennst sicher bei einer geographischen Karte die Höhenlinien. Diese erstellen wir mittels unserem Graustufenbild. D.h. jede Farbe bildet später eine eigene Höhe. Schwarz entspricht dabei Normal Null, also quasi der Meereshöhe. Die Farbe weiss symbolisiert die höchstmögliche Höhe

Nun kannst du deiner Fantasie freien Lauf lassen und einfach mal drauflos malen. Bei mir ist dieses Bild dabei rausgekommen:



Da EasyGen nur BMP-Dateien zum Import zulässt, habe ich dir gleich die BMP-Version davon mitgeliefert - einfach auf das Bild klicken um im Dropin-Ordner speichern.

Damit hättest du dein erstes Graustufenbild erstellt.

[zurück zum Abschnitt EasyGen](#)

[zurück zur Hauptseite](#)

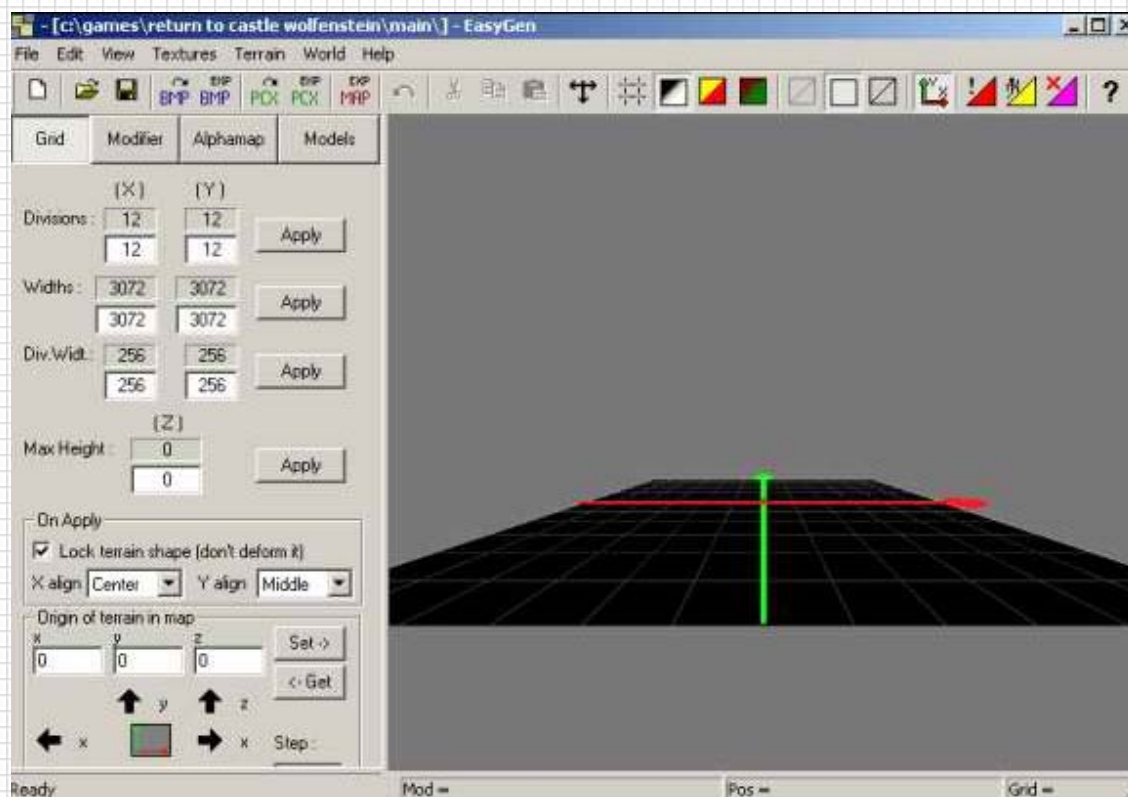
183759



Einführung:

Hier will ich dich kurz in EasyGen einführen. Da die Steuerung etwas gewöhnungsbedürftig ist, erkläre ich dir zunächst einmal die Steuerung:

- Linke Maustaste = Freier Blick in alle Richtungen
- Rechte Maustaste = Bewegung nach links/rechts/vorne/hinten
- Linke und rechte Maustaste = Damit kannst du dich in der Höhe bewegen
- Shift und linke Mousetaste = Hiermit kannst du den Playerstartpunkt verschieben (den roten Kasten)



Hier solltest du dich etwas mit der Steuerung vertraut machen, da es hier nur das 3D-Fenster zur Navigation gibt. An dieser Stelle möchte ich dir übrigens nochmal meine komplette [EasyGen-Anleitung](#) ans Herz legen - sie ist eine ideale Ergänzung zu diesem Tutorial.

[zurück zum Abschnitt EasyGen](#)

[zurück zur Hauptseite](#)

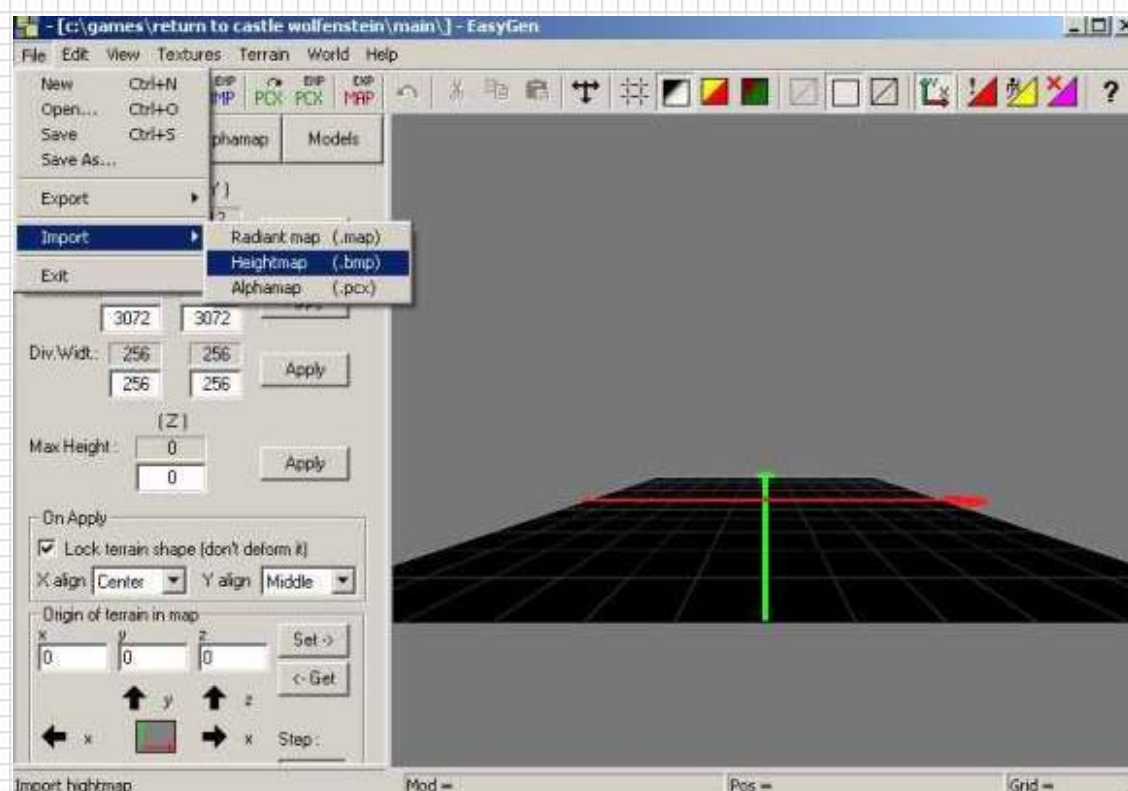
183759



das Graustufenbild importieren:

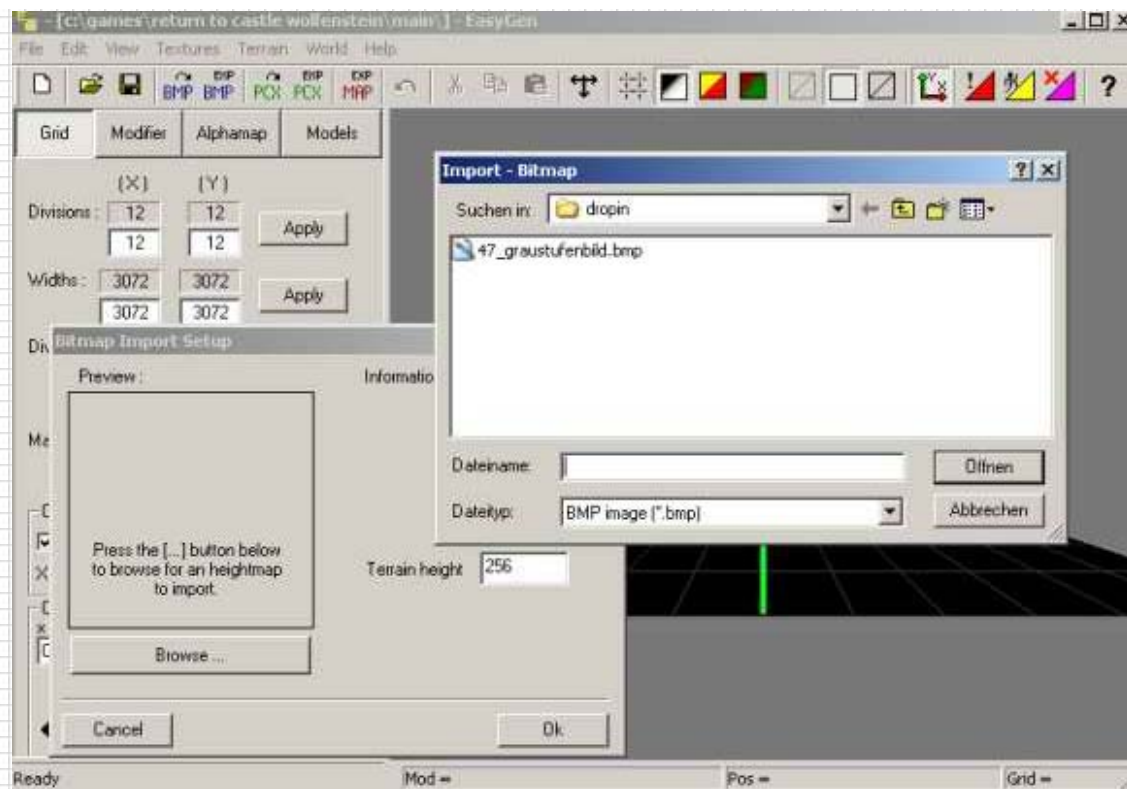
Hier will ich dir zeigen, wie man das Graustufenbild in EasyGen importiert. Dazu musst du natürlich zunächst das Graustufenbild in den Ordner "dropin" kopieren.

Nun klickst du in der Menüleiste auf "File", dann auf den Unterpunkt "Import" und dort auf den Unterpunkt "Heightmap":

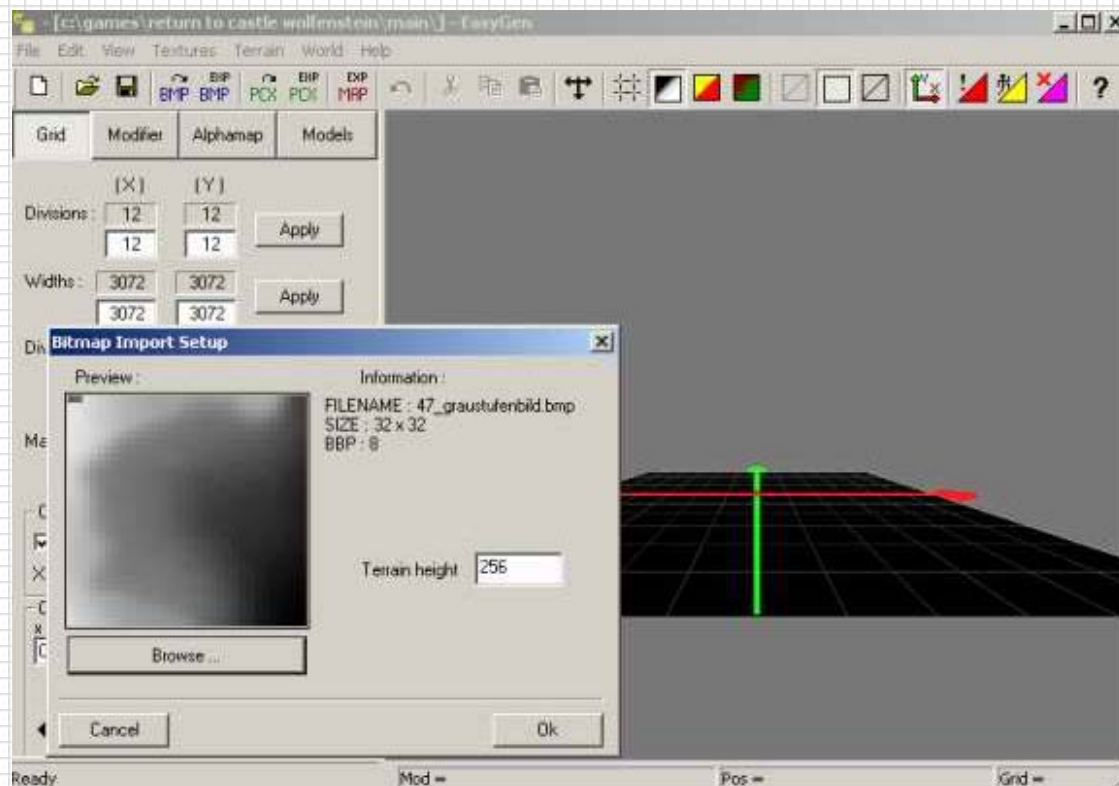


Nun erscheint ein kleineres Fenster, mit dem man nach der Heightmap suchen kann. Hier ist bereits das Graustufenbild angegeben. Wenn du es noch nicht in das Verzeichnis "dropin" kopiert hast, kannst du das schnell noch nachholen:

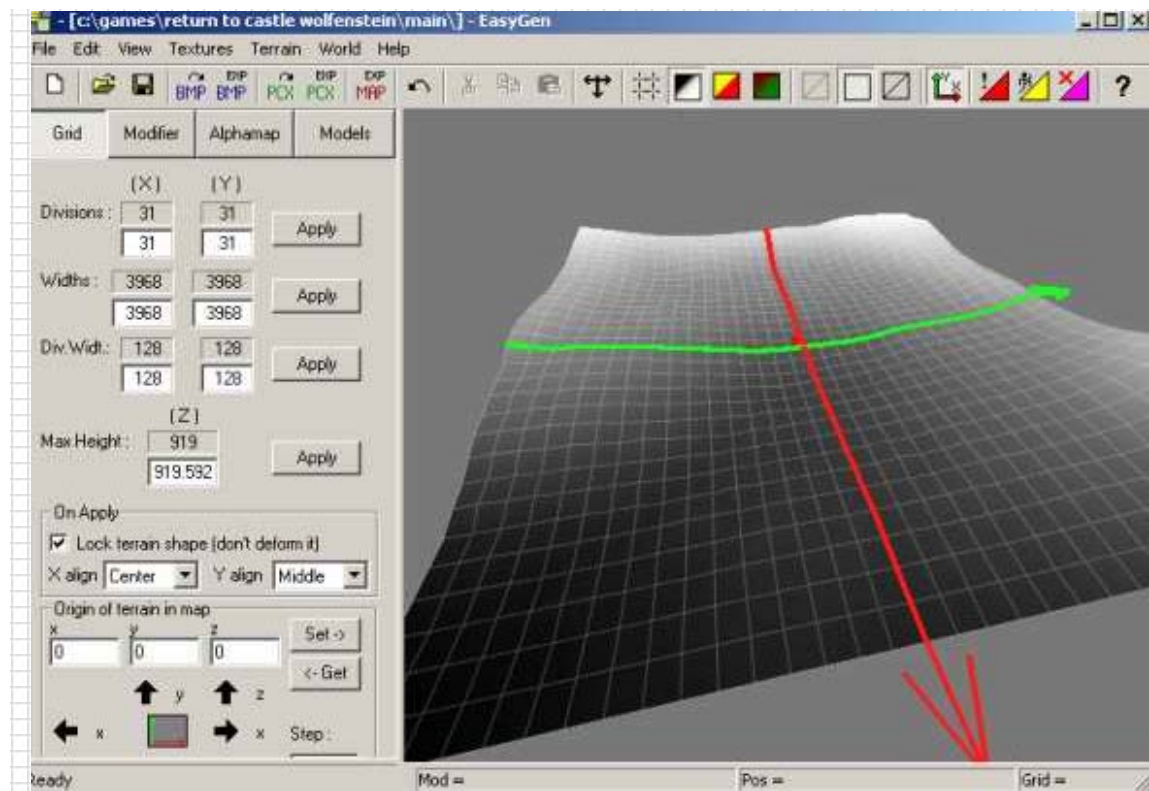




Hier klickst du einfach doppelt auf die Datei "47_graustufenbild.bmp":



und nun tragen wir noch in der weissen Zeile, in der der Wert "256" steht, "1024" ein. Damit geben wir an, wie hoch die höchste Stelle in der Map sein darf (das ist dann die Farbe "Weiss" im Graustufenbild). Nun drücken wir auf "OK". Nun wird unser Terrain erstellt:



Wie du sehen kannst, haben wir schon ein paar kleine Hügel, was du anhand des Playerstarts sehen kannst (das ist die rote Box, wo sich der grüne und der rote Pfeil kreuzen)

Nun speichern wir das ganze gleich ab - damit hätten wir auch das geschafft!

[zurück zum Abschnitt EasyGen](#)

[zurück zur Hauptseite](#)

183759

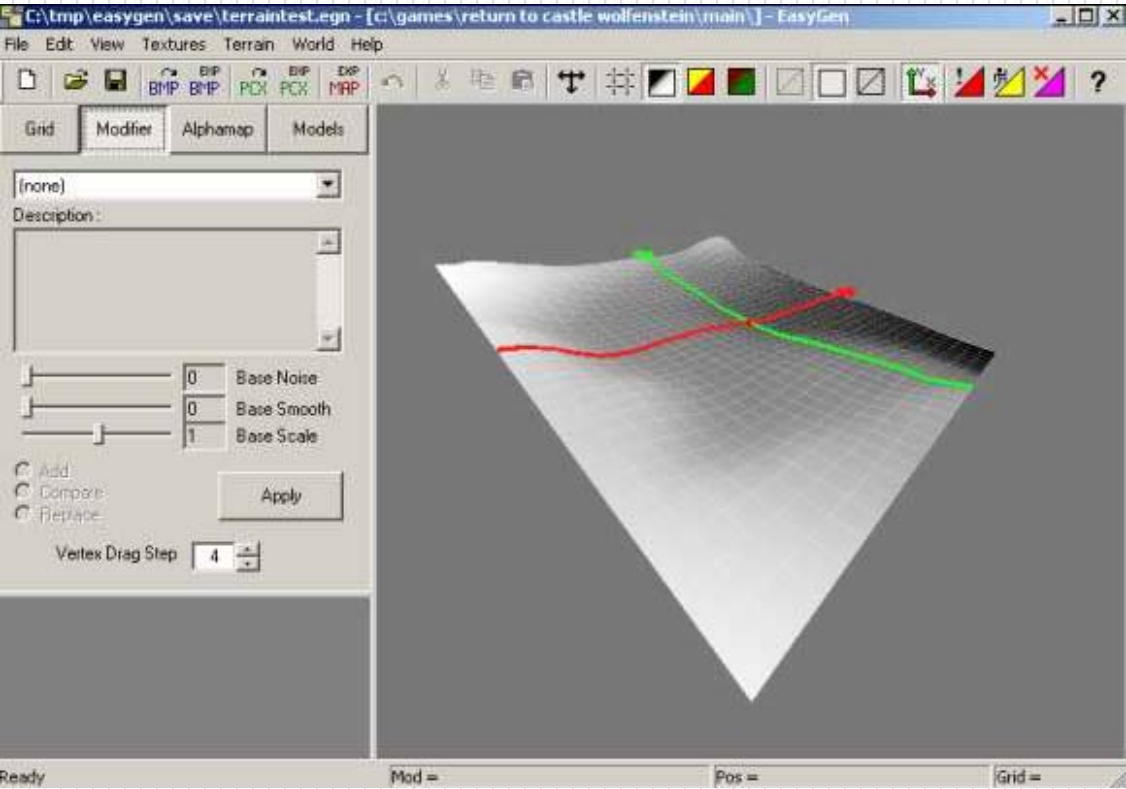


Nachträgliche Terrain-Änderungen:

Nachdem wir ja schon mittels der Heightmap ein kleines Terrain erstellt haben, wollen wir dieses noch etwas verändern - schliesslich passt ja nie alles gleich zu 100%.

So kannst du auch ohne eine Heightmap bzw. ohne ein Graustufenbild ein Terrain erstellen - das geht dann zwar nicht so schnell, sieht aber meist viel besser aus.

Legen wir los - wie du siehst, sind die Hügel im Terrain so flach, dass man dort ohne Probleme an den Rand der Karte kommen kann, das wollen wir natürlich nicht. Also werden wir am Rand der Map ein paar richtig hohe Hügel einbauen. Dazu wechselst du jetzt die Registrierkarte auf "Modifizier":



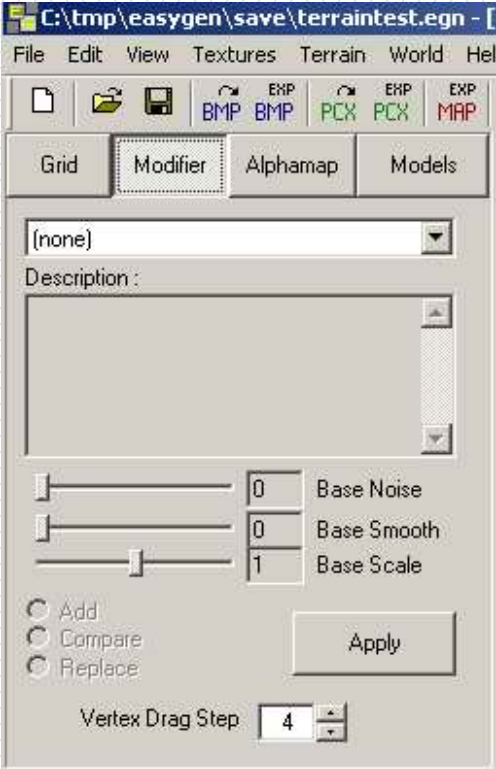
Nun wenden wir uns mal diesem Fenster etwas genauer zu:

Im Grunde sind hier nur 2 Dinge wichtig.

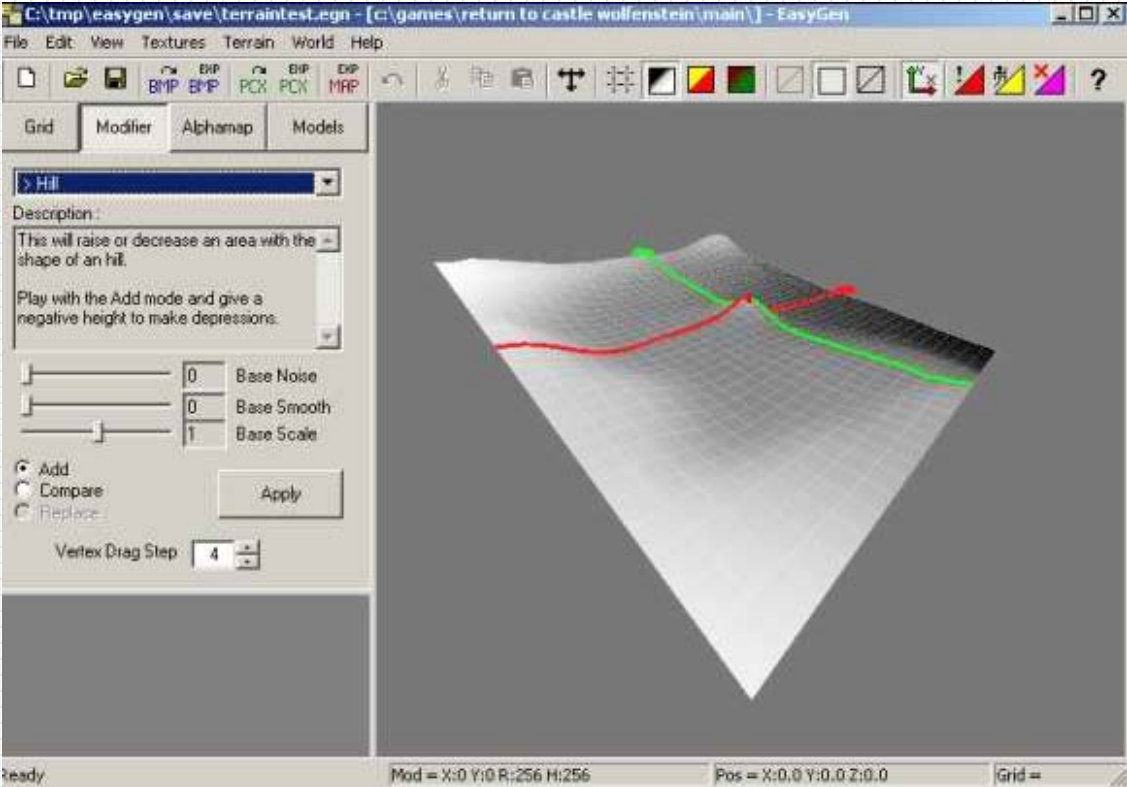
Zum einen, Das Pull-Down-Menü, wo "(none)" geschrieben steht. Hier befinden sich die

Modifier, also die Geländeformen, die wir nutzen können, um das Terrain zu verändern.

Zum zweiten, der grosse Knopf, auf dem "Apply" steht. Mit ihm werden dann der momentane Modifier angewendet.



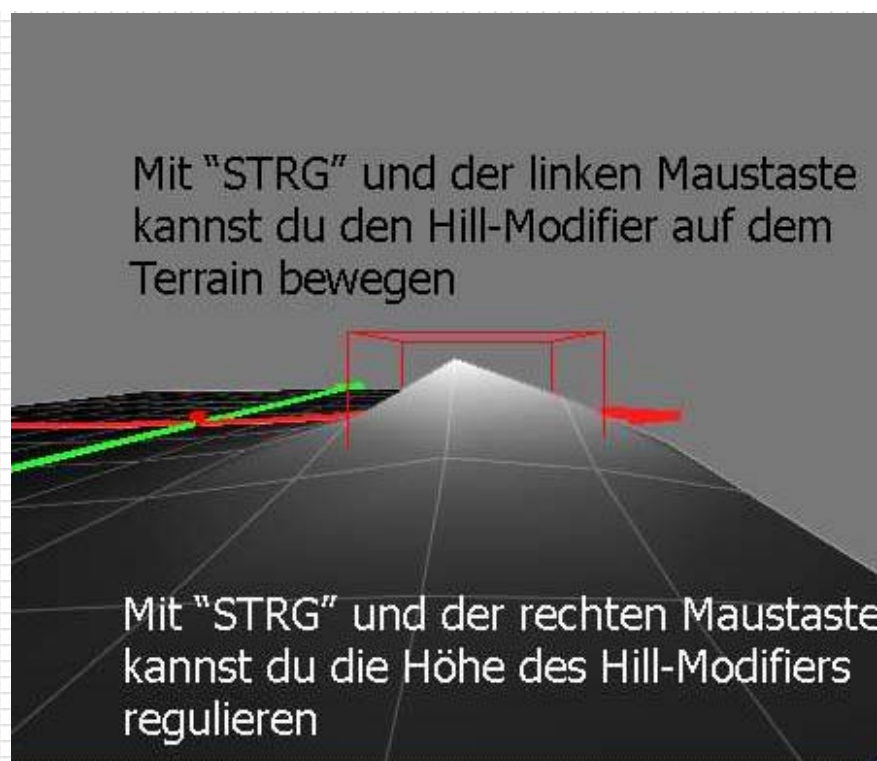
Nun wechselst du hier den Modifier auf "Hill". Wie du sehen kannst, entsteht im Zentrum des Terrains schon eine Erhebung, die den Hügel symbolisiert. Drückst du nun "Apply", so wird der Hügel genau an dieser Stelle und genau so hoch erstellt, wie du es jetzt sehen kannst:



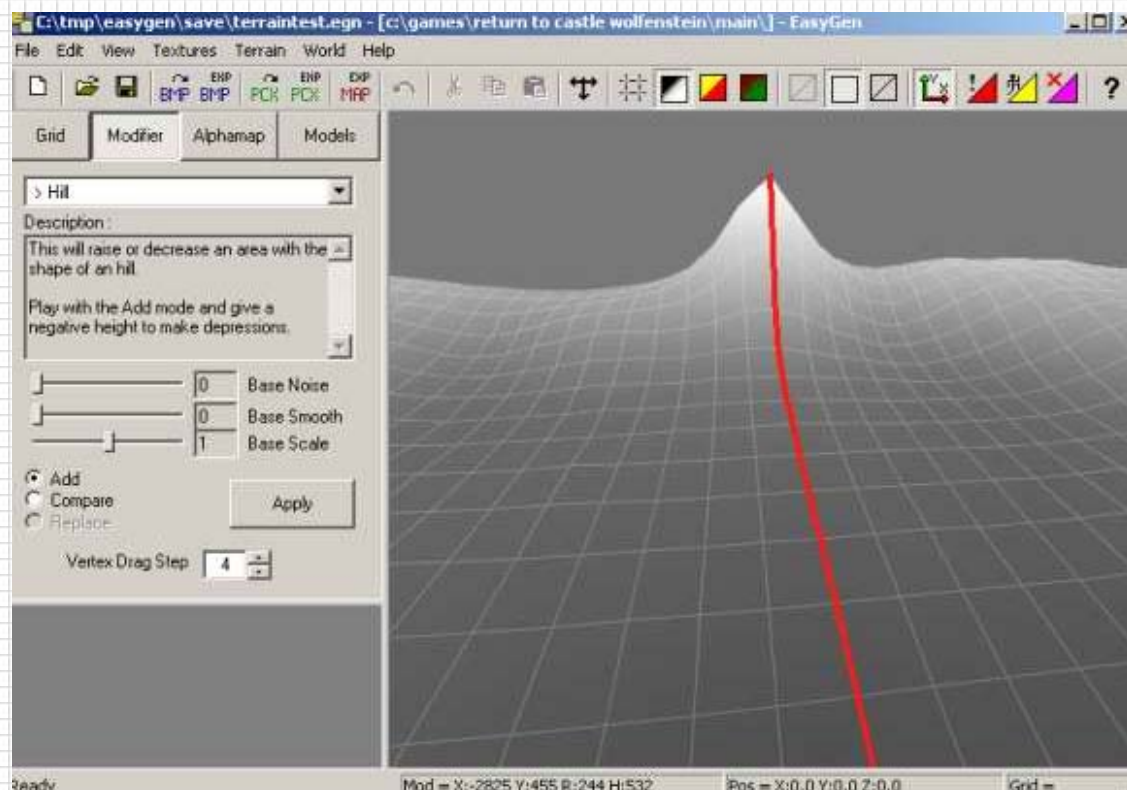
Nun wollen wir aber den Hill nicht im Zentrum, sondern ja am Rand des Terrains haben. Im nebenstehenden Bild erkläre ich dir, wie man den Hill verschieben und wie man ihn vergrößern kann.

Du kannst auch den Hill-Modifier nach rechts und links vergrößern, wenn dir der Hill zu dünn erscheint. Dazu drückst du "STRG" und beide Maustasten.

Hier solltest du etwas probieren, den Hill-Modifier an verschiedene Stellen im Terrain zu schieben.

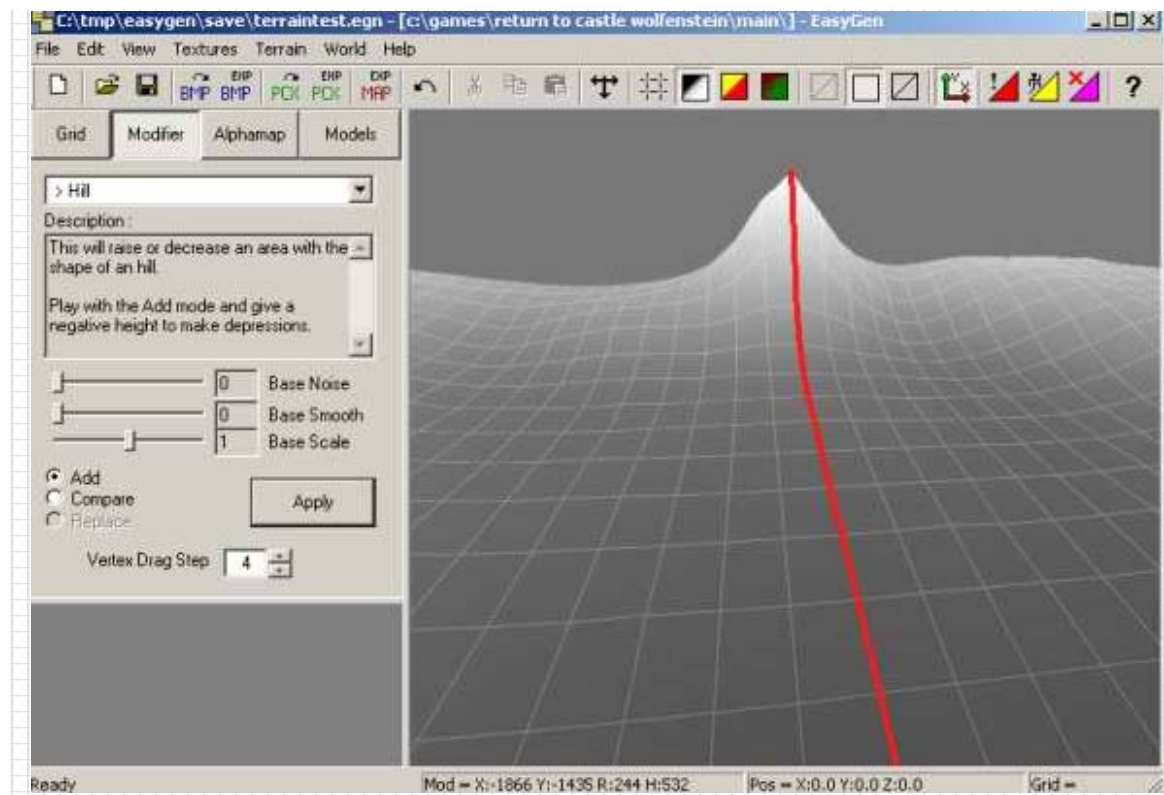


Nun schiebst du den Hill-Modifier an den Rand des Terrains, dann sollte es so aussehen:



Nun drückst du auf den "Apply"-Knopf und nun erscheint der Hill plötzlich doppelt so hoch wie vorher. Wieso ist das so? Ganz einfach - dadurch, dass du Apply drückst, wird der Hill-Modifier angewendet. Da aber der Hill-Modifier selbst noch an der Stelle sitzt, wird gleich dargestellt, wie es aussieht, wenn du noch einmal Apply drückst. Das wollen wir aber nicht.

Nun schiebst du den Hill-Modifier zur Seite, und wirst die richtige Höhe des Hills sehen:



So, damit hättest du es geschafft, deinen ersten Berg selbst zu erstellen.

[zurück zum Abschnitt EasyGen](#)

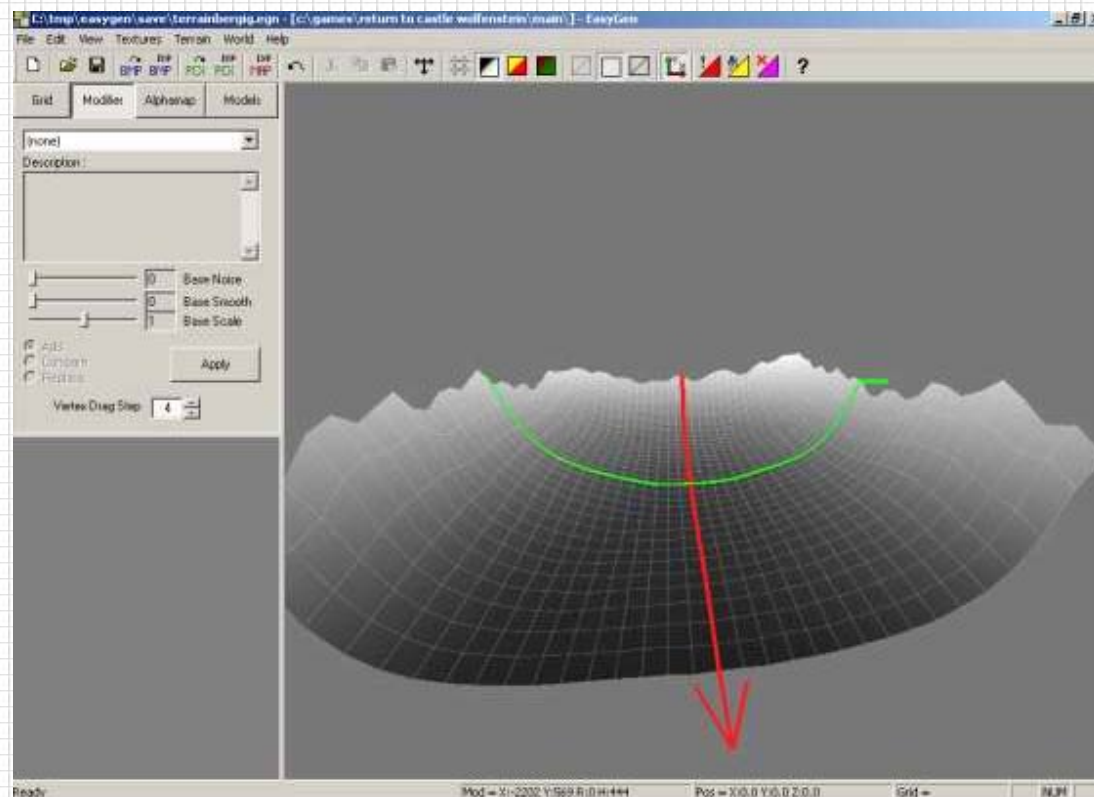
[zurück zur Hauptseite](#)

183759

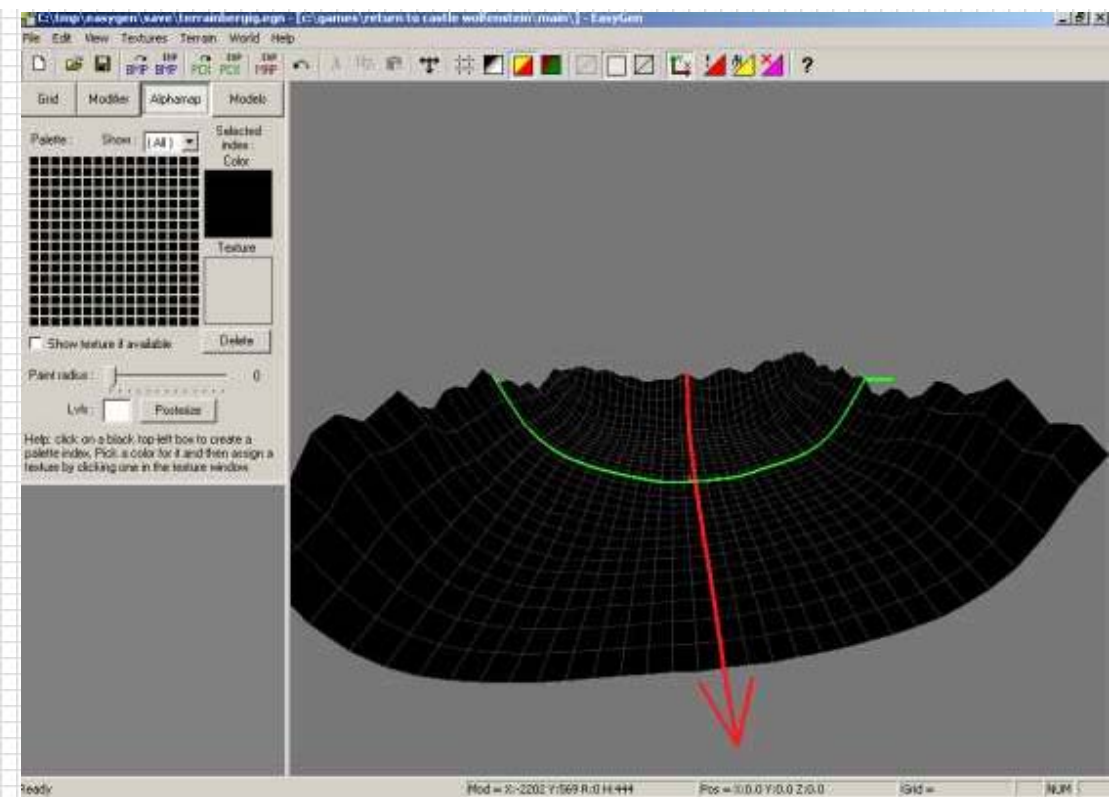


erstes Texturing:

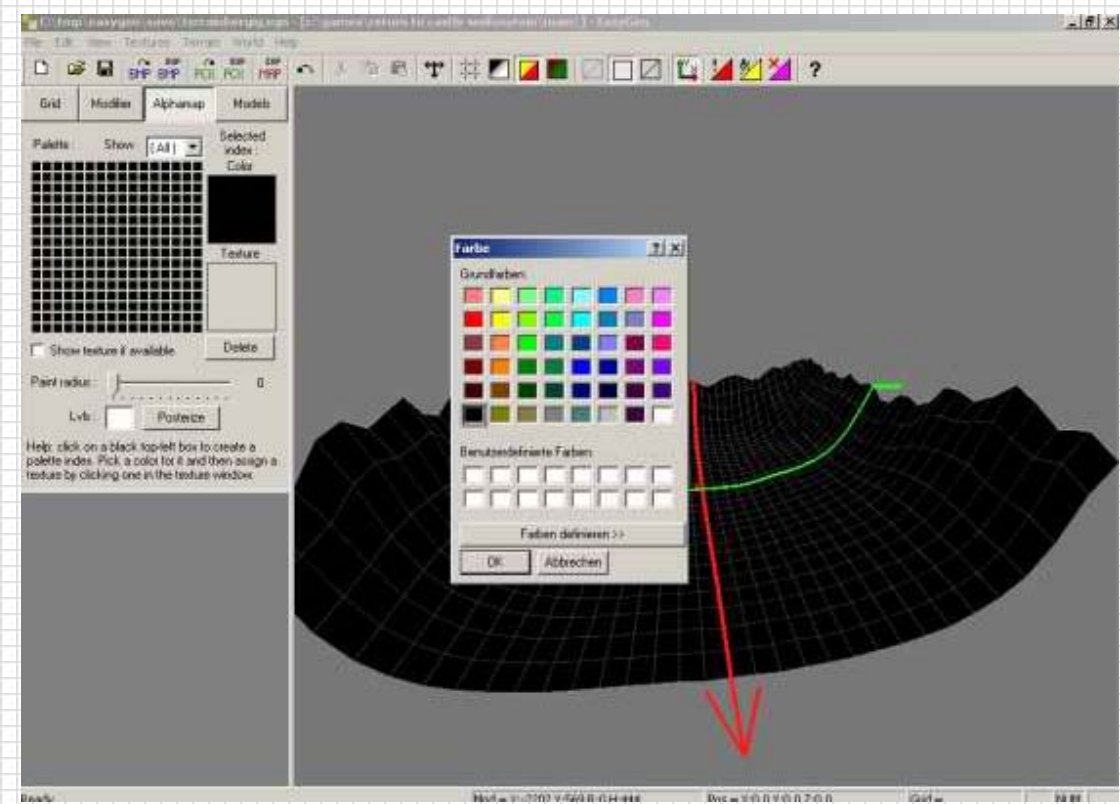
Nachdem wir bisher immer nur mit dem grauen Terrain gearbeitet haben, wollen wir jetzt langsam ans Eingemachte gehen - und Texturen so einbinden, dass das Terrain auch mal nach etwas aussieht.



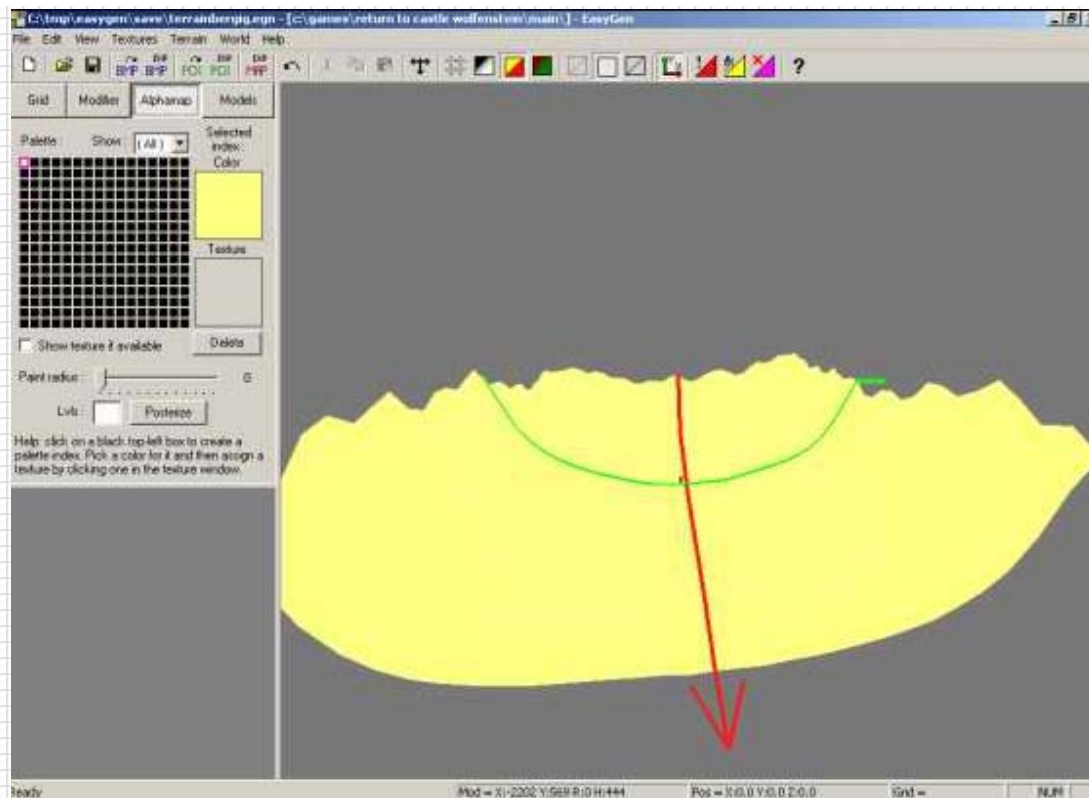
Wie du sehen kannst, habe ich inzwischen den Rand unseres Terrains bergig gestaltet, so dass der Spieler dort nichtmehr an den Rand des Terrains gehen kann. Nun klickst du links im Menü auf "Alphamap". Nun erscheint ein neues Menü:



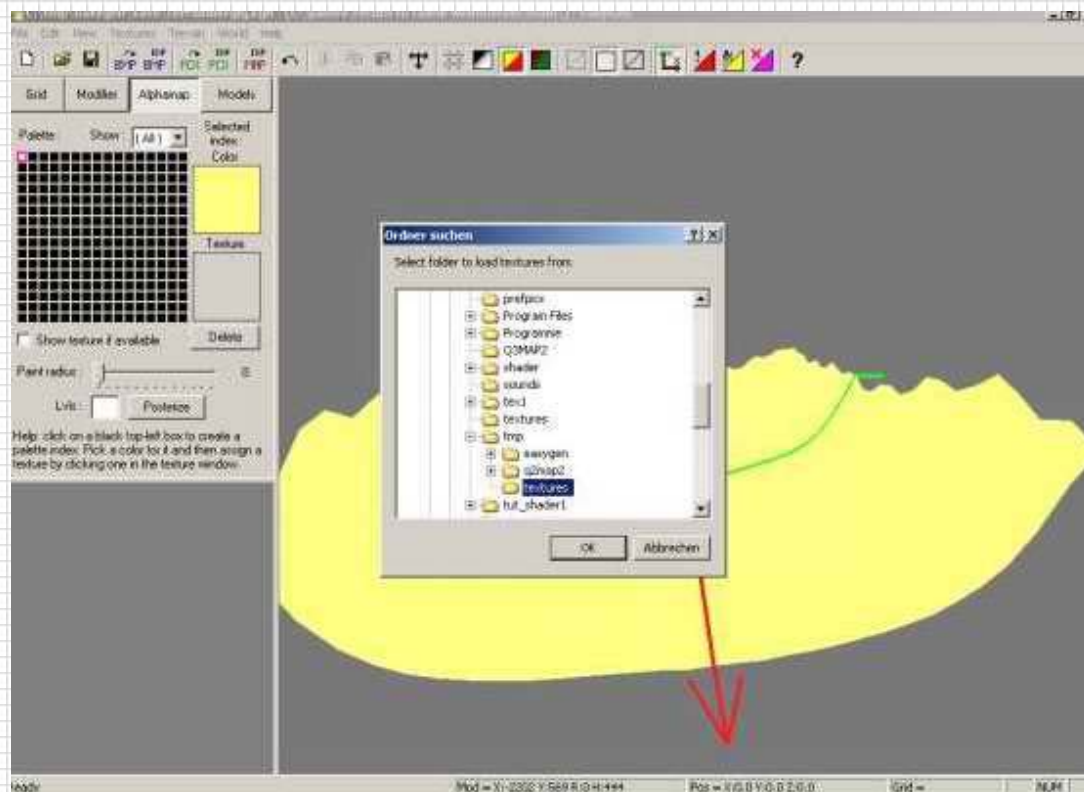
Nun hat sich auch etwas am Terrain geändert. Es ist ganz schwarz geworden. Schwarze Berge wollen wir aber nicht haben, also werden wir das ändern. Dazu klickst du im linken Bereich auf das erste der vielen kleinen schwarzen Vierecke und schon erscheint ein kleines Fenster mit der Farb-Auswahl:



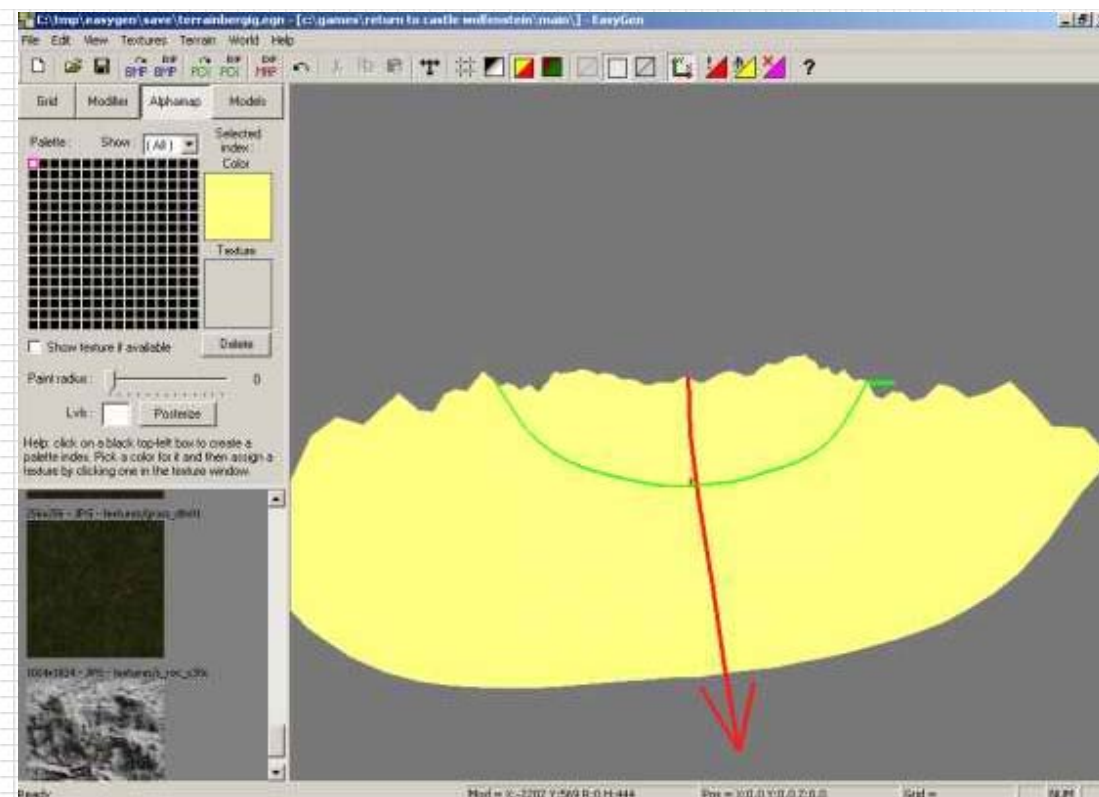
Hier kannst du dir eine Farbe raussuchen - ausser schwarz. Ich nehme mal gelb. Also klickst du einfach auf das gelbe Kästchen (welches ist eigentlich egal) und klickst auf "OK". Und schon sind die Berge gelb:



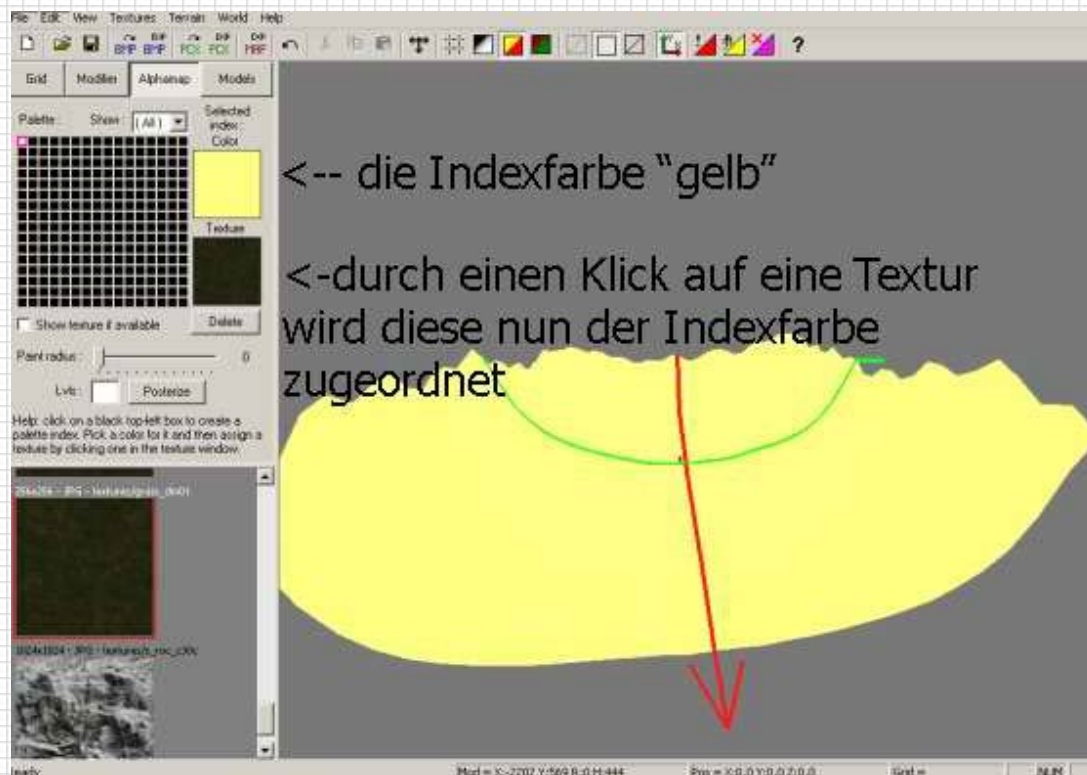
Die Farbe Gelb dient EasyGen dazu, ein Indexbild zu erstellen, welches später dann die Texturen richtig auf unser Terrain setzt. Nun haben wir die Index-Farbe, fehlt nur noch die Textur dazu. Dazu klickst du oben in der Menüleiste auf "Textures" und da auf "Add Folder":



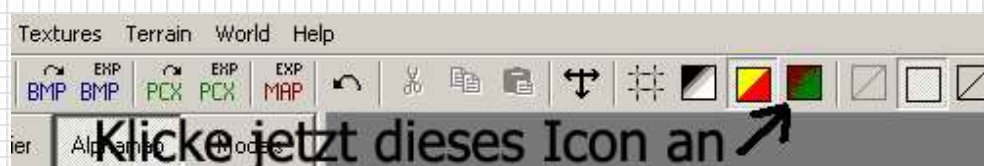
Wie du sehen kannst, habe ich mit im "Tmp"-Ordnern einen "textures"-Ordnern angelegt, in dem sich die Texturen befinden, die ich für mein Terrain verwenden will. Diese Texturen habe ich aus den Pk3-Dateien von RtcW genommen. Natürlich kannst du auch andere Texturen nehmen. Nun drücken wir auf "OK" und die Texturen werden geladen (die geladenen Texturen siehst du ganz unten rechts in dem kleinen Fenster):



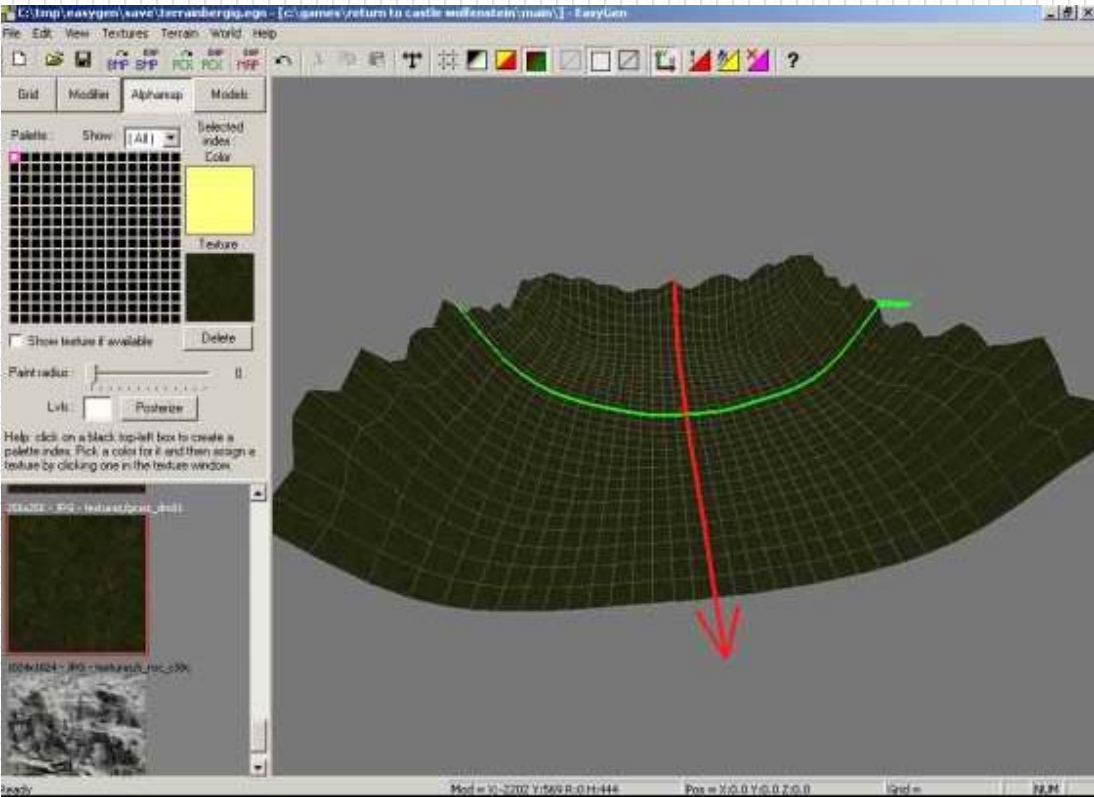
Nun klickst du einfach auf irgendeine Textur - ich wähle die grüne Textur, die du im Bild sehen kannst. Nun wird der gelben Index-Farbe diese Textur zugeteilt:



Nun klicken wir auf dieses Icon:



Und schon erstrahlt dein Terrain in einem schönen Wiesen-Grün:



So, damit hätten wir das soweit fertig.

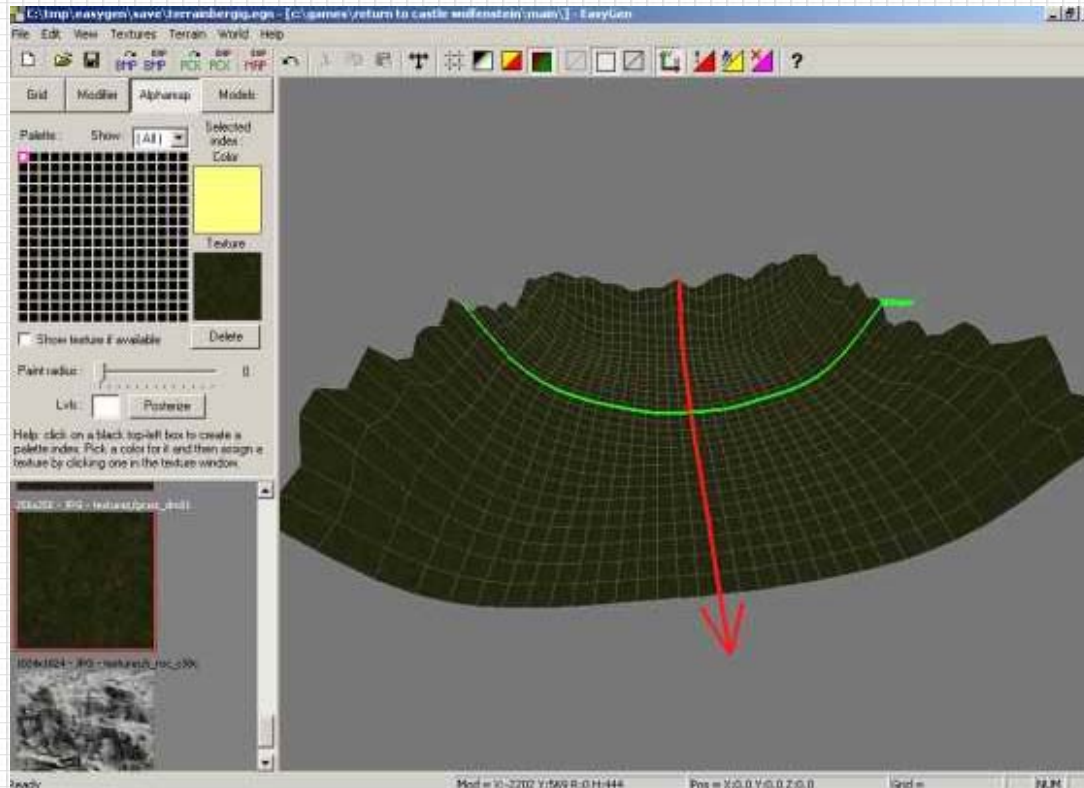
[zurück zum Abschnitt EasyGen](#)

[zurück zur Hauptseite](#)

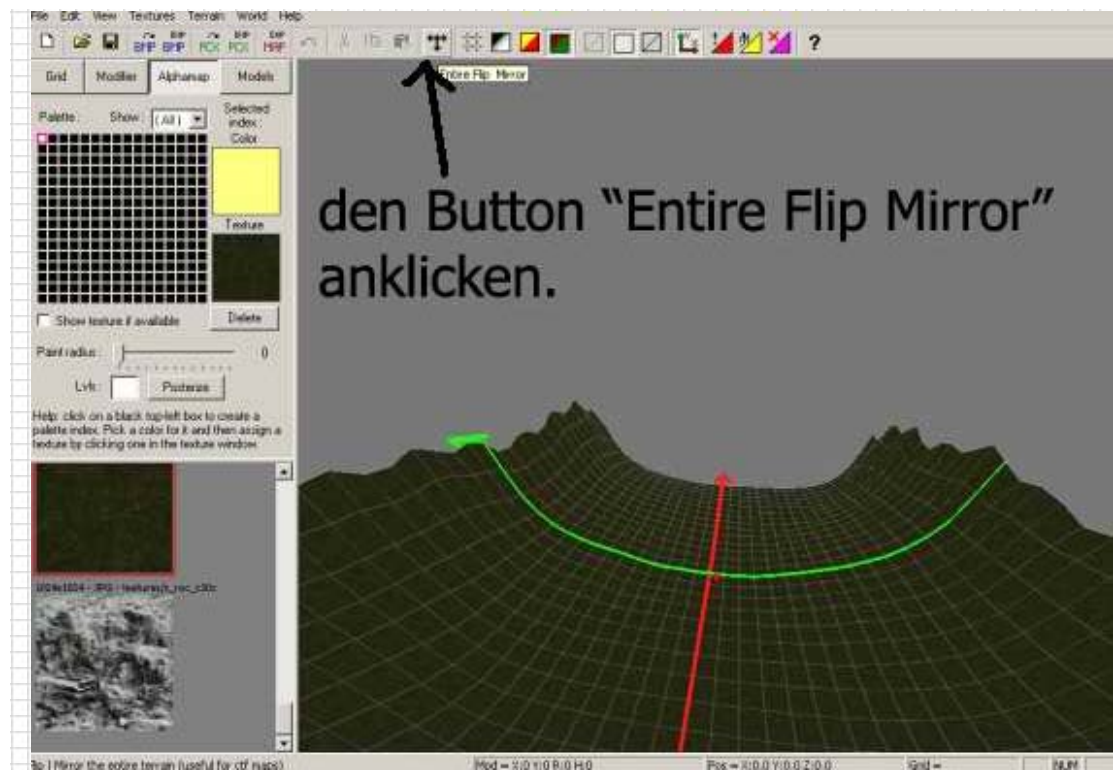
183759



Terrains schliessen :

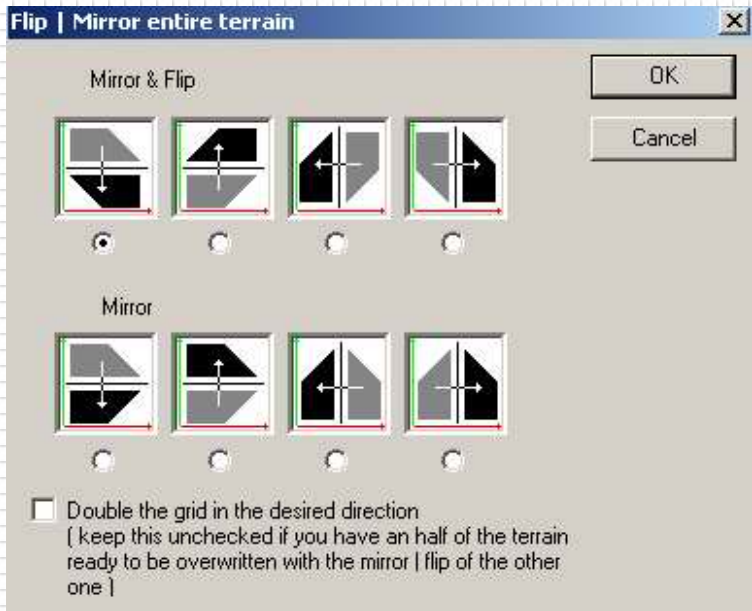


So, nun haben wir unser schönes Terrain und es ist auf einer Seite noch ganz offen. Natürlich gibt es jetzt mehrere Möglichkeiten, diese Seite zu schliessen - wir haben bisher nur manuell Hügel eingefügt und damit die Seiten geschlossen. Nun wollen wir zu anderen Mitteln greifen. Dazu klickst du auf den Button "Entire Flip Mirror" (mit den drei schwarzen Pfeilen darauf):



den Button "Entire Flip Mirror" anklicken.

Hier kann man nun aus ein folgenden Eigenschaften wählen:

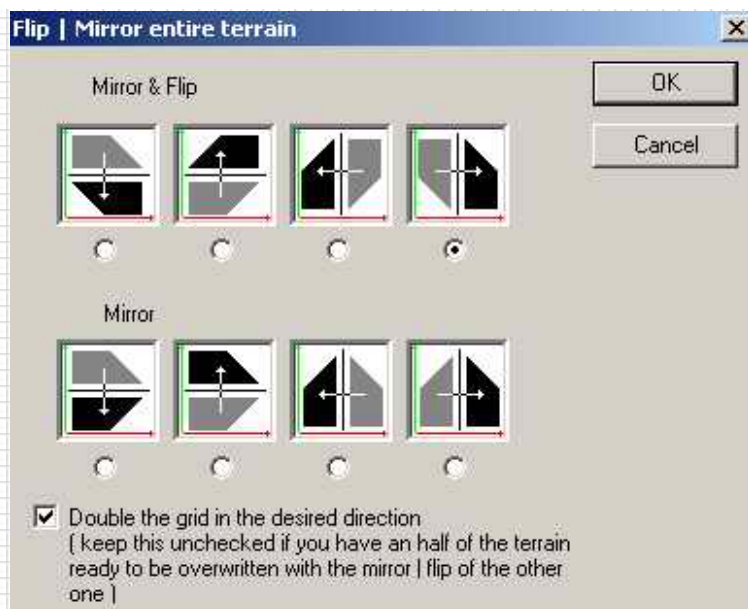


Mirror & Flip: Die Auswahl in diesem Teil lässt das Terrain komplett duplizieren.

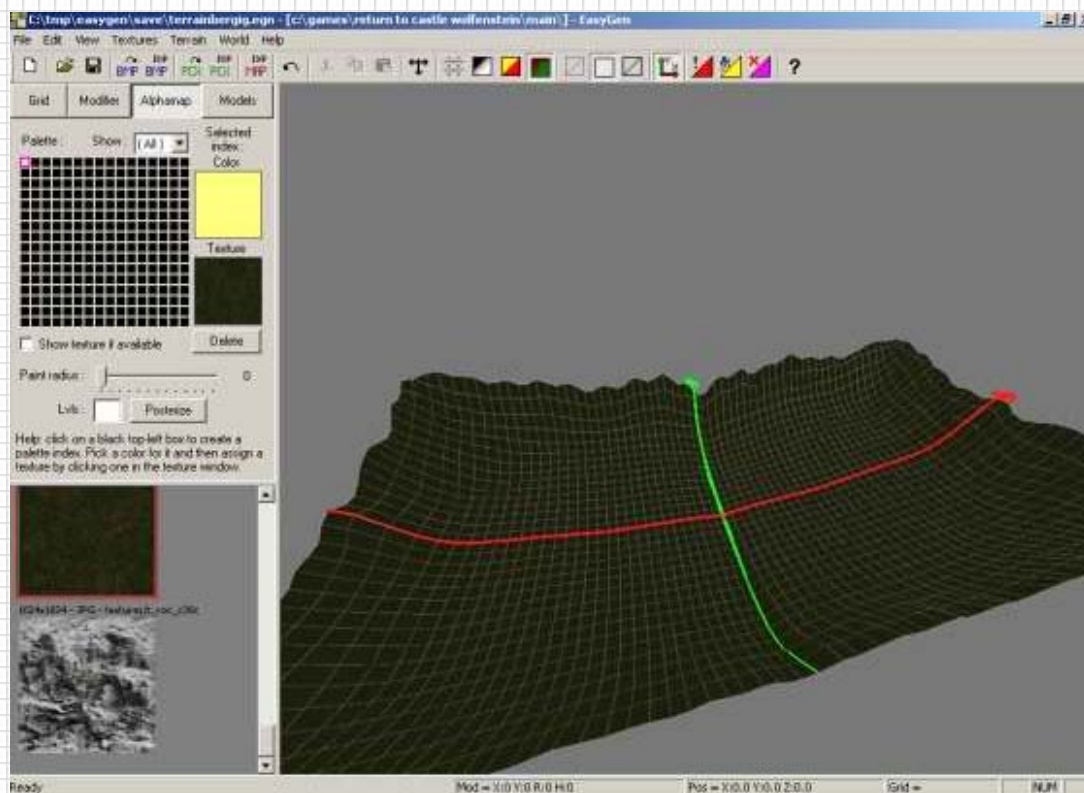
Mirror: Durch diese Auswahl wird nur eine Seite des Terrains gespiegelt.

Durch das Häkchen bei "Double The Grid..." wird das Terrain gespiegelt und gedreht

Wir wählen das letzte Kästchen im "Mirror & Flip" und setzen ein Häkchen bei "Double The Grid...":



Nun drückst du auf "OK", und schon verdoppelt sich das gesamte Terrain:



Damit hätten wir das auch geschafft. Nun können wir das Terrain schon testen. Wie das geht, siehst du im nächsten Thema.

[zurück zum Abschnitt EasyGen](#)

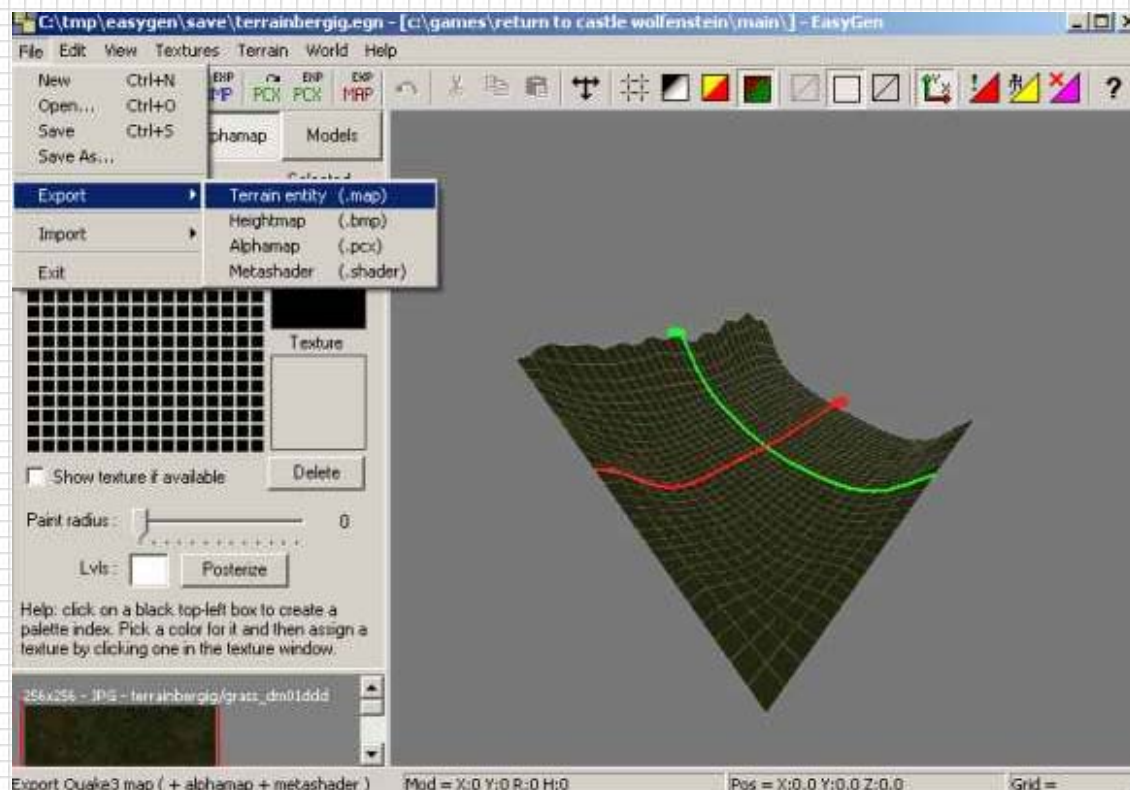
[zurück zur Hauptseite](#)

183759



Export des Terrains:

Nachdem wir ja jetzt unser schönes Terrain erstellt haben, möchten wir dieses natürlich auch testen. Dazu öffnest du erst deine Map. Nun klickst du auf "Export", und da auf "Terrain Entity (.map)":



Nun erscheint ein neues Fenster:

Hier solltest du darauf achten, dass der Shader einen einmaligen Namen hat, da es sonst zu Fehlern bei der Darstellung kommt. Ebenfalls solltest du oben rechts bei "General / Worldspawn Options" die 4 Häkchen setzen.

Wenn du das alles gemacht hast, kannst du auf "Continue" klicken.



Nun kannst du die Map im Radiant ansehen. EasyGen belegt die Skybox mit der Caulk-Textur. Diese musst du noch durch eine andere Himmels-Textur ersetzen. Dann nur noch compilieren, und schon sieht es so aus:



[zurück zum Abschnitt EasyGen](#)

[zurück zur Hauptseite](#)

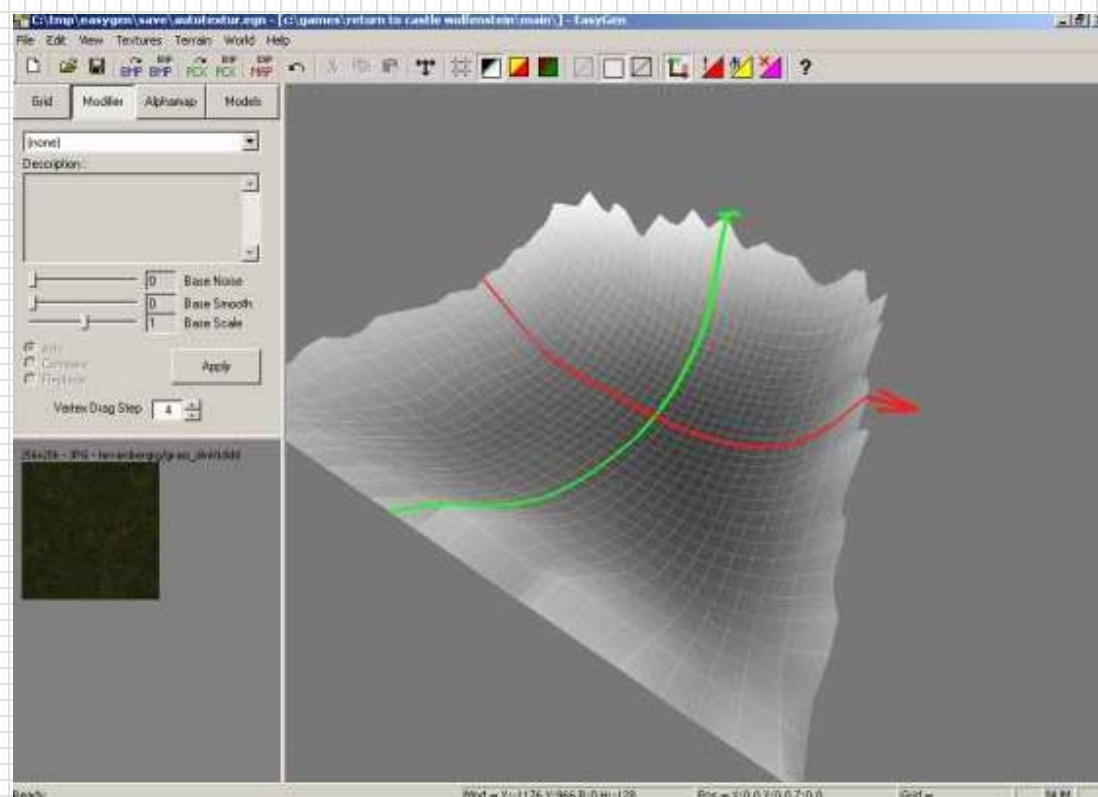
183759



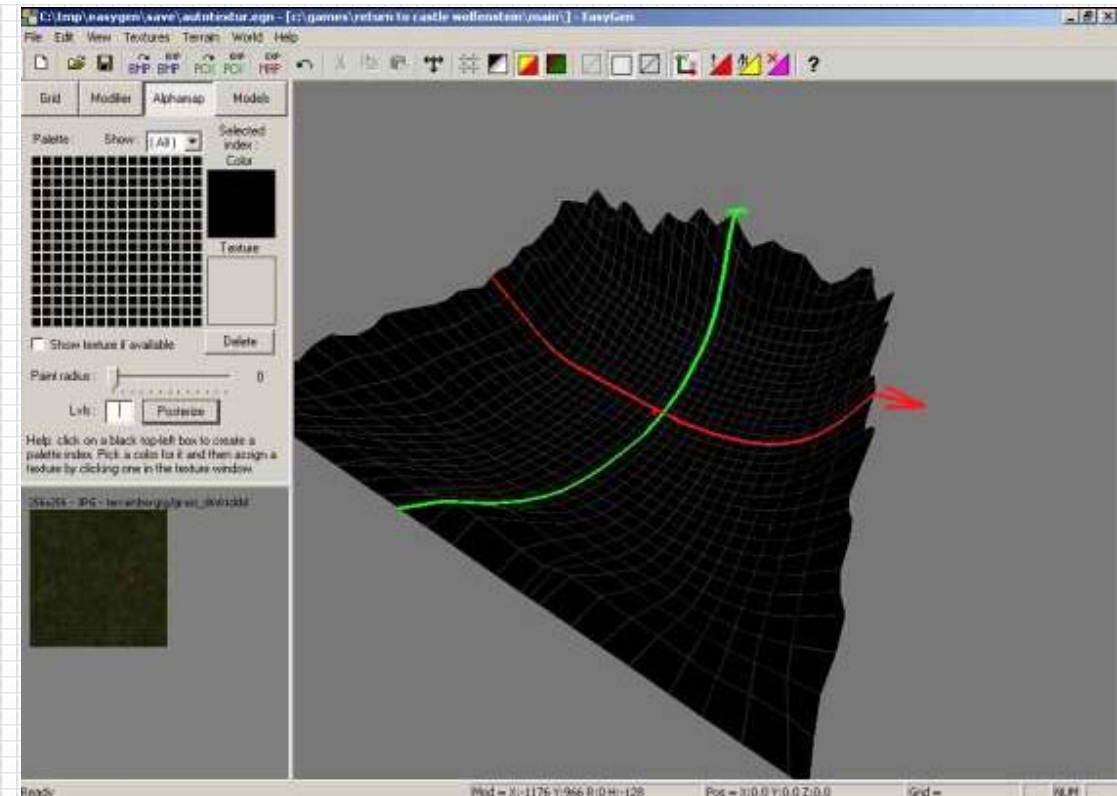
Automatische Mehrfachtexturierung:

Natürlich gibt es auch bei EasyGen die Möglichkeit, das Terrain mit mehreren Texturen zu versehen. Dies geht zum einen manuell, d.h. man pinselt selbst das Terrain an, oder man lässt EasyGen für sich arbeiten. In diesem Thema will ich dir die zweite Methode erklären. Dazu brauchen wir natürlich zuerst ein Terrain.

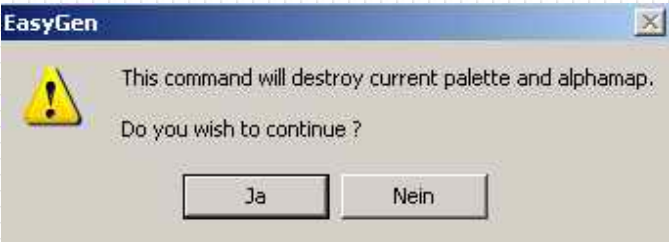
Ich benutze das Terrain, das wir mittels der Graustufenmap erstellt haben und dann die Seiten erhöht haben. Nun habe ich aber noch die eine Seite geschlossen:



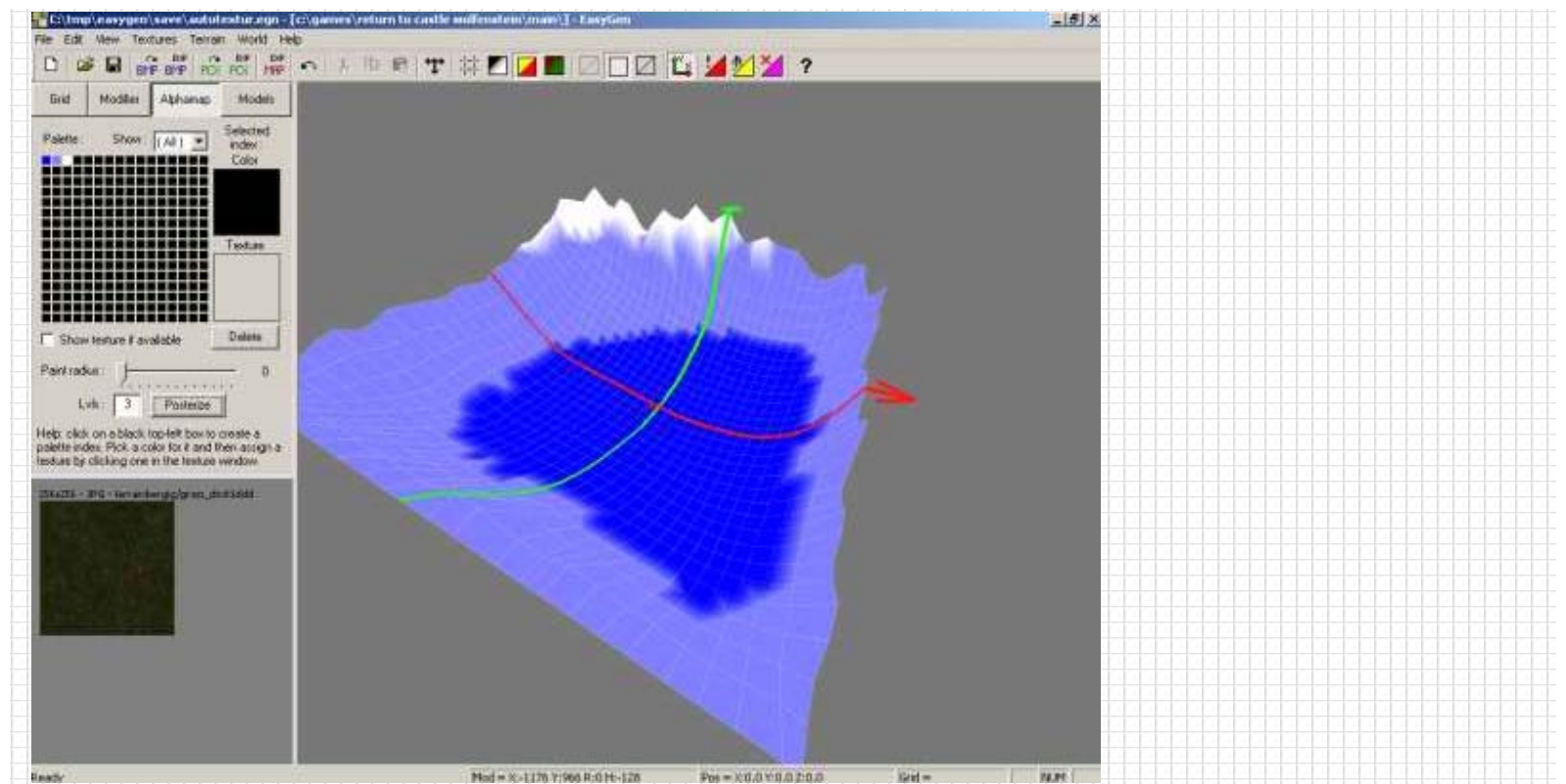
Nun klickst du auf "Alphamap". Nun erscheint das neue Menü auf der linken Seite:



Hier wenden wir uns jetzt dem mittleren Teil des Menüs zu, dort - wo "Posterize" steht. Hier gibst du nun den Wert "3" ein und nun kannst du sehen, was passiert:



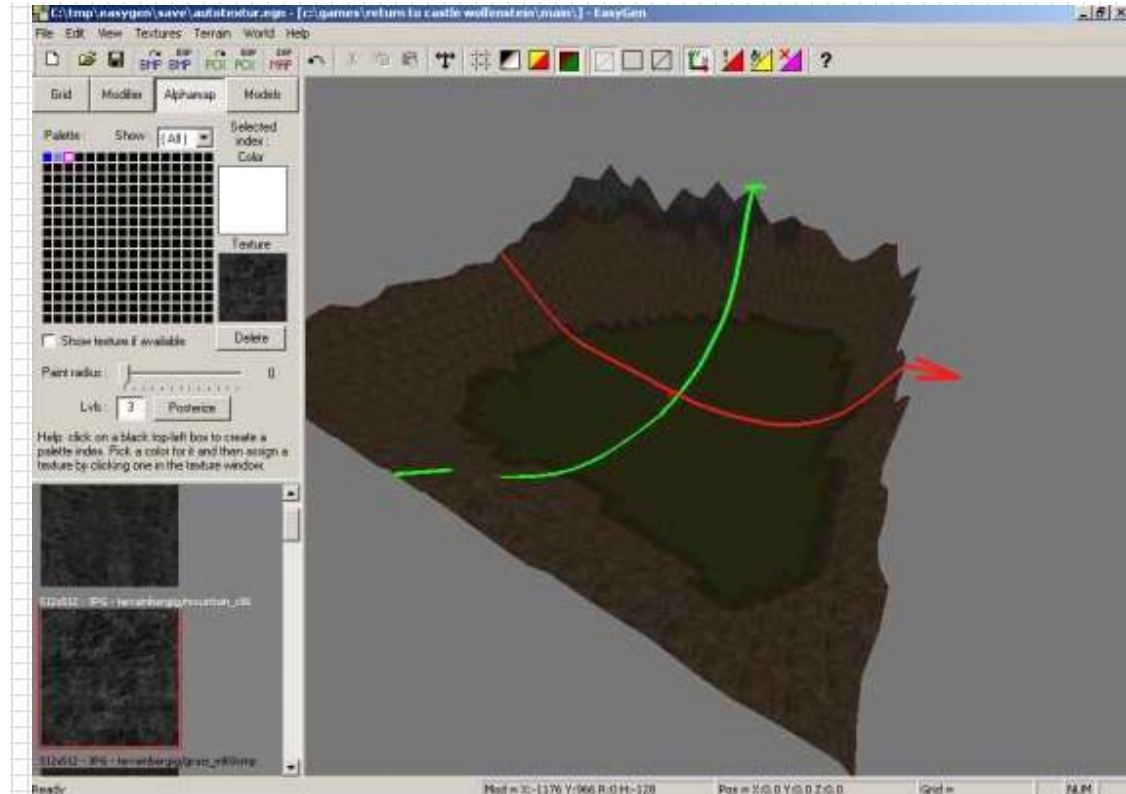
Hier klicken wir auf "Ja" und schon geht es los:



Und schon erstellt uns EasyGen eine wunderschöne Alphamap mit drei Texturen, die wir noch schnell angeben können. Nun noch einen klick auf dieses Icon:



Und schon haben wir ein Terrain, dass sich auch von den Texturen her wirklich sehen lassen kann:



[zurück zum Abschnitt EasyGen](#)

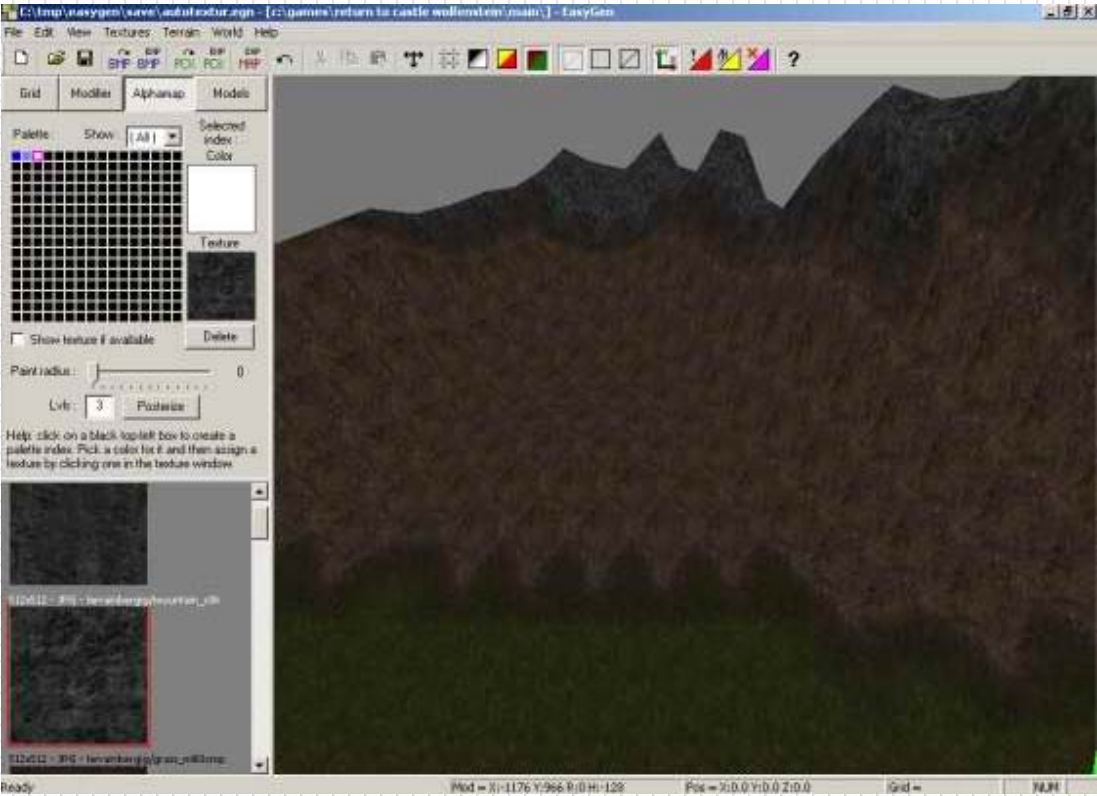
[zurück zur Hauptseite](#)

183759



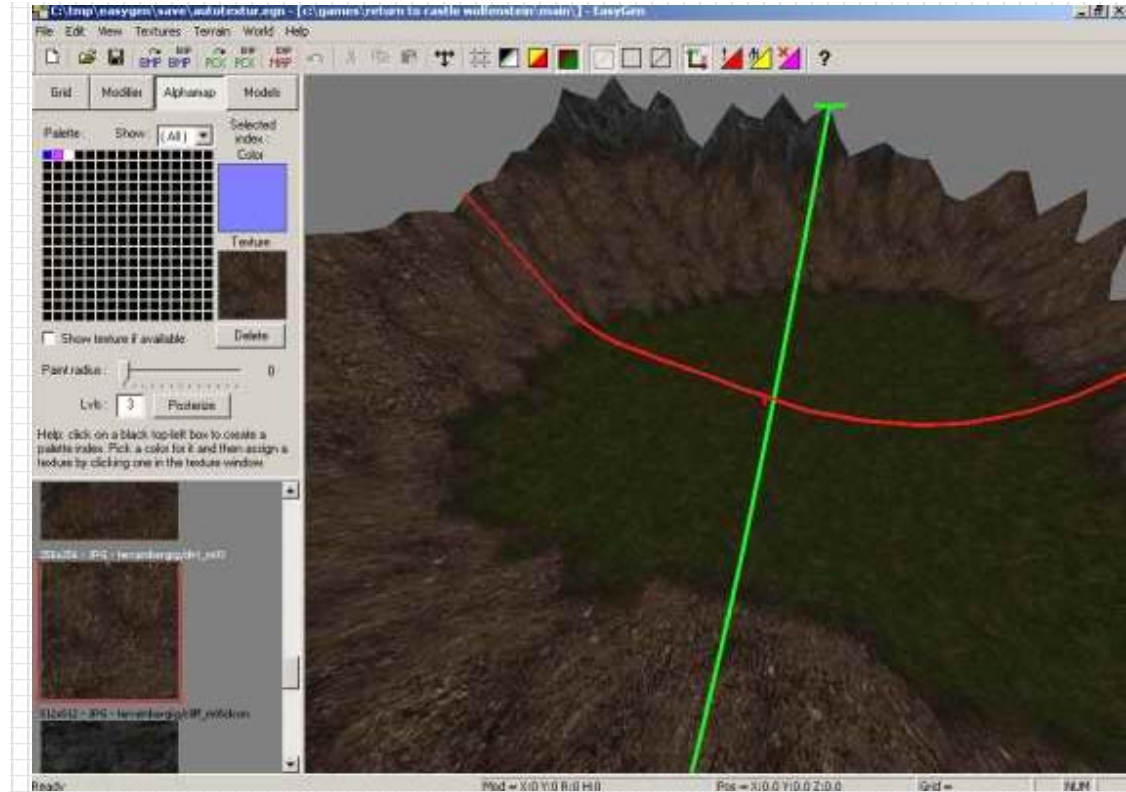
Texturen Scalling:

In unserem letzten Thema haben wir ja diese Map erstellt:



Nun wirst du mir wohl zustimmen, dass die mittlere Textur leicht bescheiden aussieht, da die vielen Texturwiederholungen doch etwas auffallen. Das wollen wir in diesem Thema ändern. Dazu klickst du nun die Indexfarbe an, die die Textur vertritt, in diesem Fall die Mittlere (also das helle Blau).

Nun kannst du auf dem Nummern-Block auf deiner Tastatur die Tasten "+" und "-" drücken, um die Texturen zu scallieren. d.h. ihre Grösse zu ändern. So kommt man z.B. durch 6-maliges Drücken der Taste "+" auf dieses Ergebnis:



So sieht es doch schon viel ordentlicher aus.

[zurück zum Abschnitt EasyGen](#)

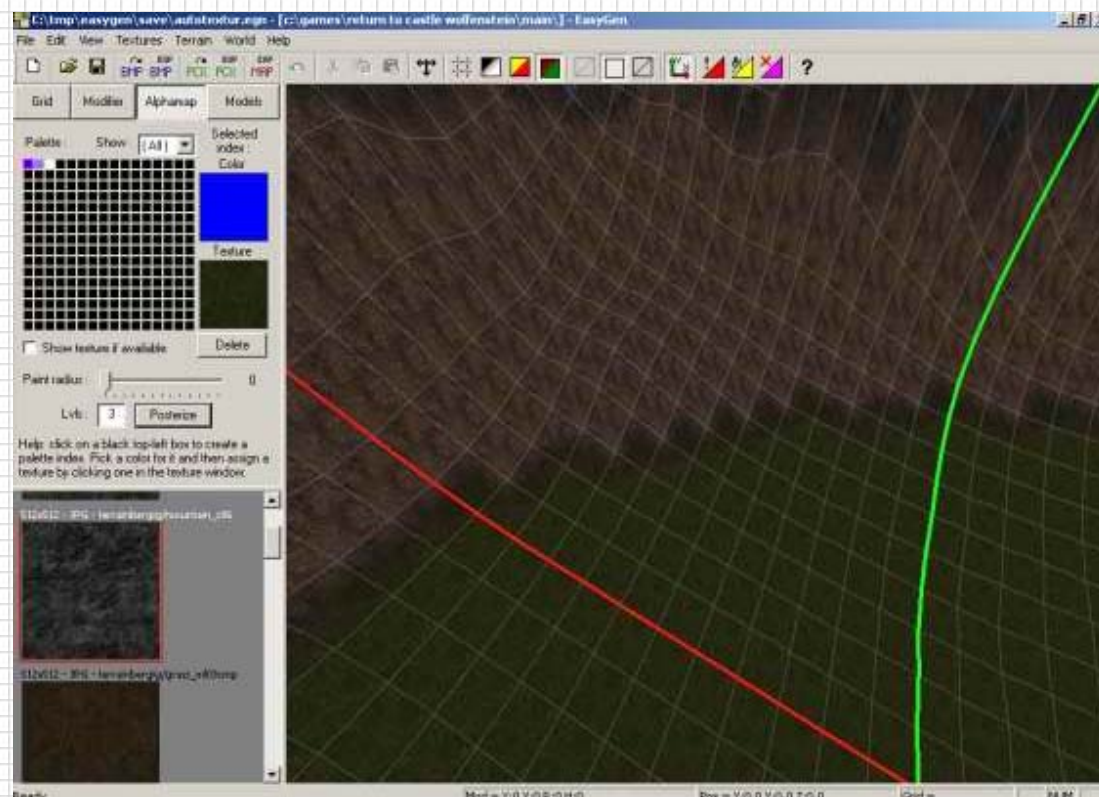
[zurück zur Hauptseite](#)

183759

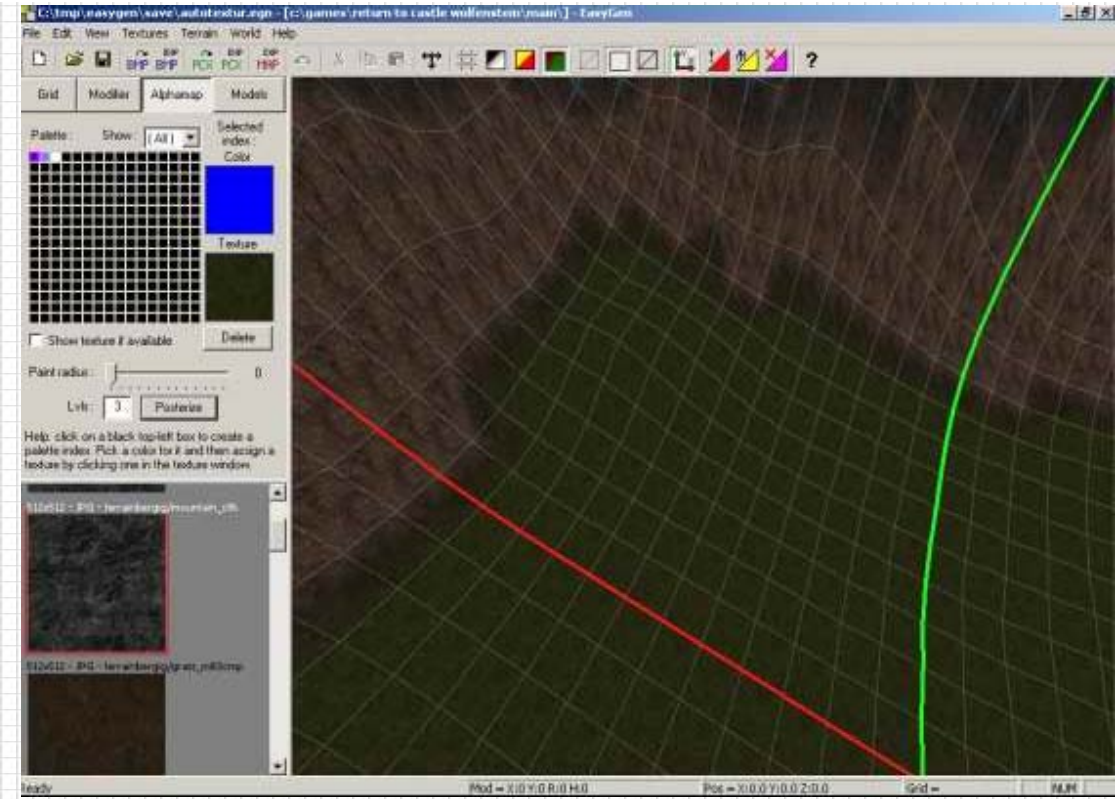


Manuelle Texturierung:

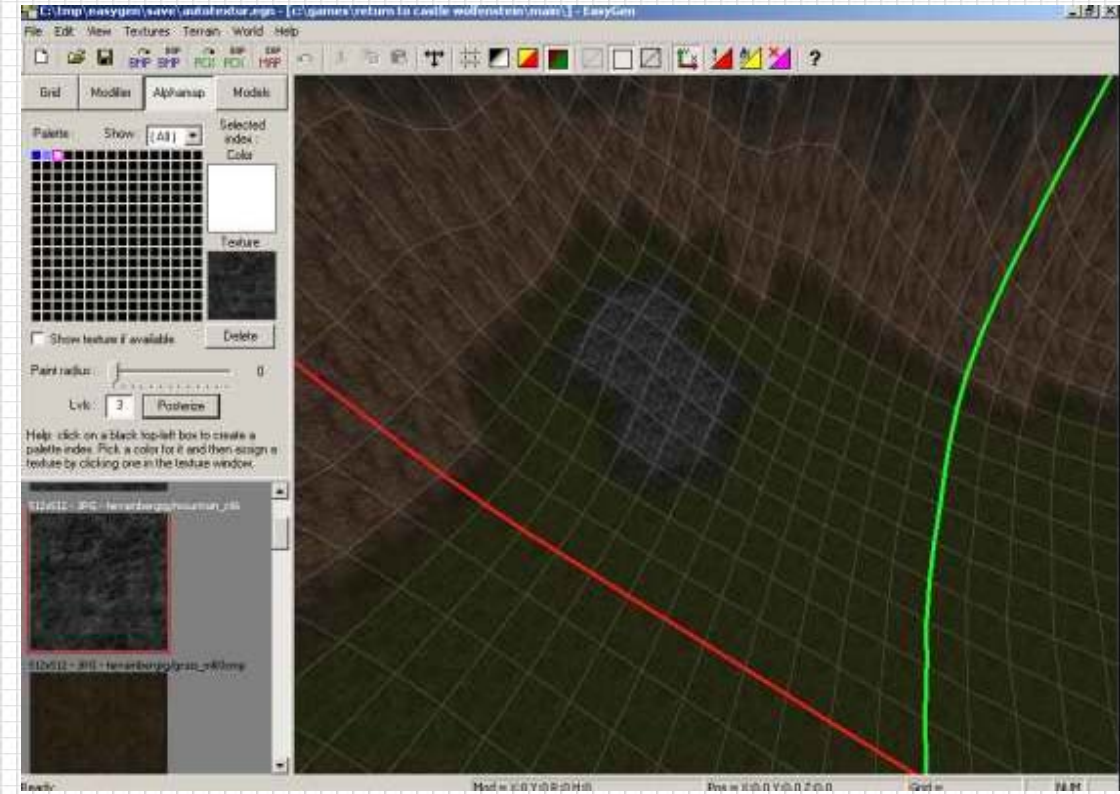
So, nachdem wir nun die ganze Zeit diese Arbeit mit EasyGen umgangen haben, will ich dir jetzt erklären, wie man selbst das Terrain texturieren kann. Dazu brauchen wir natürlich erst wieder ein Terrain, und wechseln auch gleich in die "Alphamap". Hier solltest du auch schon ein paar Index-Farben und Texturen angelegt haben. Ich benutze immernoch das Terrain aus der automatischen Texturierung für dieses Thema:



Ich denke, an dieser Erhöhung sollte man die Wiesen-Textur bis an den höheren Berg setzen, da es so doch etwas komisch aussieht. Dazu wählst du nun die Index-Farbe an, die für deine Wiesen-Textur steht (in unserem Fall die Farbe "Dunkelblau"). Nun drückst du die Tasten "STRG", "SHIFT" und die linke Maustaste und fährst auf diesem Gelände herum. Nun siehst du, dass du das Terrain mit der neuen Textur belegst:



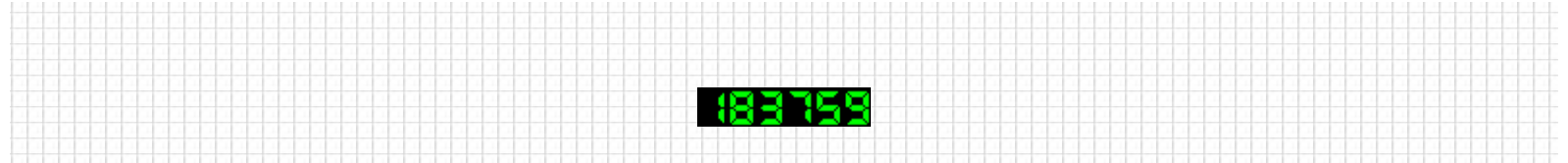
So sieht es doch schon etwas besser aus. Nun wollen wir noch etwas von der Steintextur am Rand haben, dass es etwas nach Steinbruch aussieht. Dazu wählst du die Indexfarbe für die Steintextur (in unserem Falle die Farbe "Weiss"):



So, das ganze kann sich doch sehen lassen, oder ?? Jetzt kannst du die Map ja gleichmal testen und angucken, was du vielleicht noch von den Texturen her besser oder schöner machen könntest.

[zurück zum Abschnitt EasyGen](#)

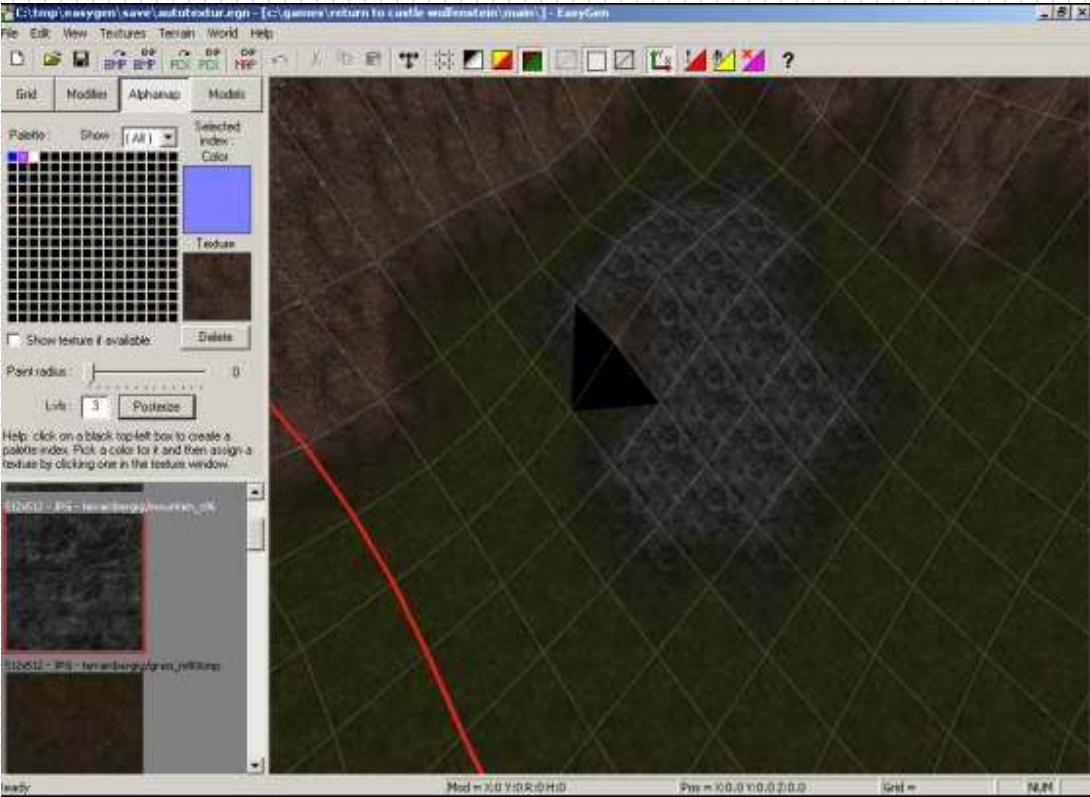
[zurück zur Hauptseite](#)



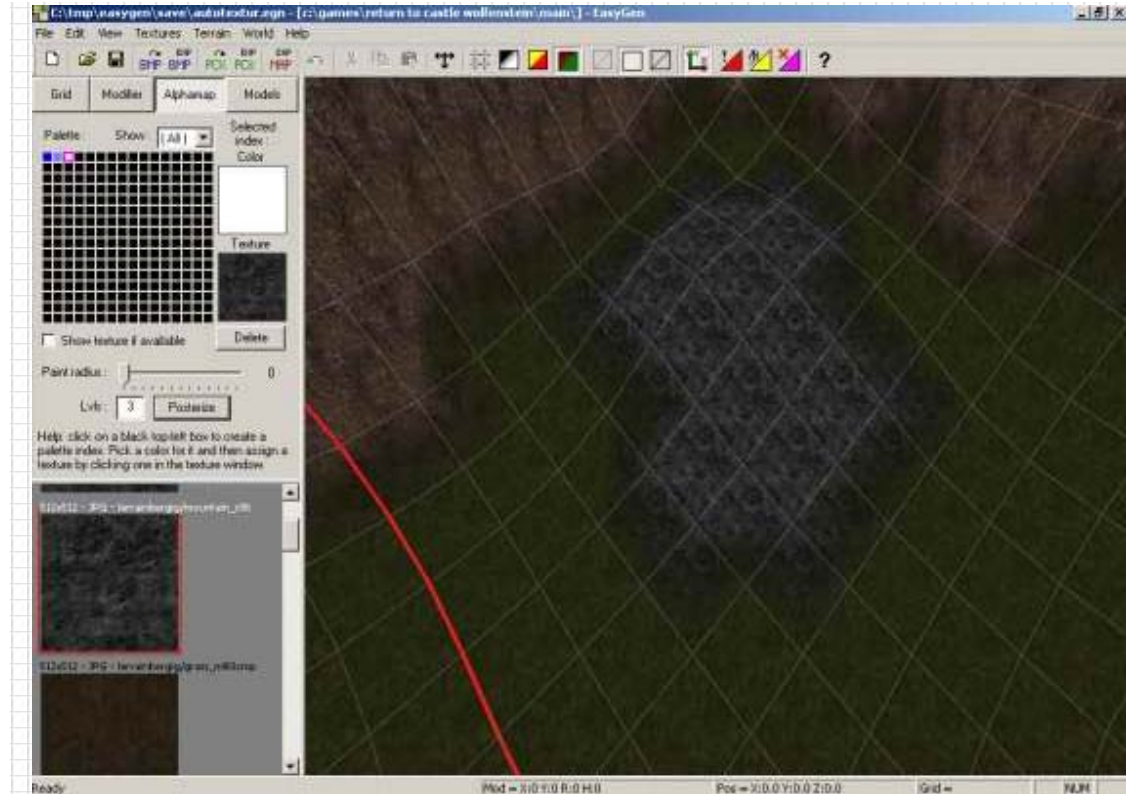


Texturfehler:

Vielleicht ist es dir auch schon passiert, dass in deinem Terrain eine Art "schwarzes Dreieck" oder ein "schwarzer Fleck" aufgetaucht ist. Das ist ein Fehler von EasyGen. Hier treffen nämlich mehr als 2 Texturen aufeinander und genau dies ist nicht erlaubt:



Wie du siehst, treffen hier drei Texturen aufeinander - die grüne Wiesen-Textur, die Felsen-Textur und oberhalb des schwarzen Flecks die schmutzige Mitteltextur. Hier bleibt nichts übrig, als selbst Hand anzulegen. Zunächst solltest du dir eine Textur aussuchen, die den schwarzen Fleck überdecken soll. Logischerweise sollte das eine Textur sein, die dort schon vorkommt, da sonst der Fehler ja wieder passiert. Ich entscheide mich für die Felsen-Textur. Also wähle ich links die Farbe "Weiss" aus, die ja in unserem Beispiel die Felsen-Textur darstellt. Nun drückst du die Tasten "STRG" + "SHIFT" und die linke Maustaste und bermalst den schwarzen Fleck mit der neuen Textur. Dies tust du so lange, bis der schwarze Fleck verschwunden ist:



Und schon ist der schwarze Fleck weg und du kannst hier gleichmal die Map testen, wie sie aussieht.

[zurück zum Abschnitt EasyGen](#)

[zurück zur Hauptseite](#)

183759



Korrektter Export in die Pk3-Datei:

Um ein Terrain erfolgreich in die fertige Map zu exportieren, benötigt man natürlich mehr als nur die *.map Datei. Dazu gehören leider alle Dateien, die EasyGen ausspuckt. Hier will ich dir erklären, wo du die Dateien in die Pk3-Verzeichnisse kopieren musst:

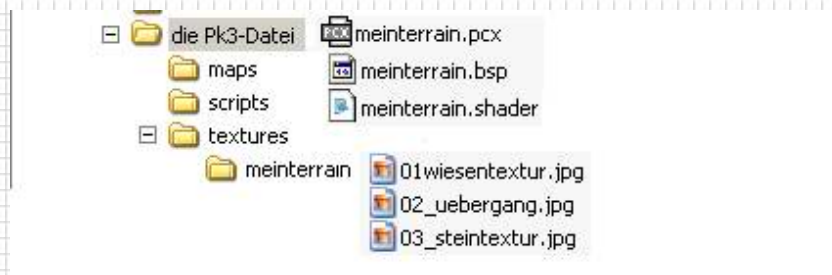
Nehmen wir an, dein Terrain heisst "meinterrain". Nun gibt es verschiedene Dateien, die EasyGen ausspuckt:

- meinterrain.map
- meinterrain.shader
- meinterrain.pcx
- [meinterrain.egn](#) <- muss nicht in die Pk3 - dies ist das abgespeicherte Terrain unter EasyGen

Nun haben wir also 4 Dateien, die wir in die Pk3-Verzeichnisse kopieren müssen:

- meinterrain.map -> wird zur Datei "meinterrain.bsp" kompiliert und kommt in den "Maps"-Ordner
- meinterrain.shader -> kommt in den "Scripts"-Ordner
- meinterrain.pcx -> muss beim kompilieren im Main-Ordner sein, sonst kann das Terrain nicht korrekt erstellt werden. Wir kopieren es aber in den Root-Ordner deiner Pk3-Datei.

Da Bilder mehr als tausend Worte sagen, hoffe ich, dass dir dieses Bild bei der Erstellung bei deiner Pk3 hilft:



Du darfst natürlich nicht vergessen, dass du die Texturen beilegst, die nicht in den Original-Pk3-Verzeichnissen enthalten sind. In unserem Beispiel habe ich drei Texturen benutzt, diese 3 Texturen habe ich hier auch eingesetzt.

[zurück zum Abschnitt EasyGen](#)

[zurück zur Hauptseite](#)

183759



Installation:

Q3Map2 - das ist der Name des neuen Compilers, der viele neue Effekte in die Q3-Engine bringt. Bump-Mapping, Fenster-Reflexionen und vieles vieles mehr. Dieses Tutorial ist nicht unbedingt einfach, da besonders die Einrichtung des Compilers einiges abverlangt und man relativ lange nur im Nebel rumrührt. Deshalb wird es hier auch weniger Screenshots geben, da man wirklich viel verändern muss. Aber fangen wir an:

Zunächst benötigst du die aktuelle Version der Q3Map2. Diese findest du hier:

<http://shaderlab.com/q3map2/>

Hier kannst du auch im Ordner "manual" das Handbuch finden - leider komplett in englischer Sprache. Nun solltest du dir einen Ordner anlegen, in den du die ganzen Dateien entpackst, z.B. "C:\Q3map2". Wie du siehst, gibt es im Ordner "Extras" 4 Shader-Dateien:

- cel.shader
- common_extra.shader
- lightgrid.shader
- q3map.shader

Diese kopierst du natürlich in deinen "Main/Scripts"-Ordner. Wie ich dir gezeigt habe, musst du nun noch die Datei "Shaderlist.txt" öffnen und dort die folgenden Einträge vornehmen:

Wie du siehst, darfst du nicht vergessen, die Endung ".shader" wegzulassen.



Ausserdem noch 2 Bilder, "skip.tga" und "antiportal.tga". Diese beiden Bilder kopierst du in den Ordner "Main/Textures".

Nun benötigen wir noch eine Bat-Datei um die Q3map2.exe anzusteuern. Dazu öffnest du einen Texteditor und kopierst folgendes hinein:

```
echo off
set bsp_var= -meta -threads 1
set vis_var= -vis -fast
set light_var= -light -fast
set quake_path=c:/games/Retur~1 -connect 127.0.0.1:39000
set mod_dir=main
set map_path=c:/radiant/wolfenstein/main/maps/test.map
set q3map_path=c:/radiant/q3map2/q3map2.exe
set gen_options= -game wolf -fs_basepath %quake_path% -fs_game %mod_dir% -v
%q3map_path% %bsp_var% %gen_options% %map_path%
%q3map_path% %vis_var% %gen_options% %map_path%
%q3map_path% %light_var% %gen_options% %map_path%
```

pause

Nun speicherst du das ganze ab als "q3map2cfg.bat". Nun will ich dir noch schnell hier die ganzen Parameter erklären:

echo off	dies ist der Startbefehl der Bat-Datei
set bsp_var =	hier werden alle allgemeinen Variablen eingegeben
set vis_var=	hier werden alle Variablen für die Vis-Berechnung eingegeben
set light_var	hier werden alle Variablen für die Lichtberechnung eingegeben
set quake_path=	hier wird der Pfad zum Spiel angegeben. Da Bat-Dateien nicht mit Leerzeichen zurechtkommen, solltest du "Retur~1" anstatt "Return To Castle Wolfenstein" eingeben
set mod_dir=	Dies ist der Ordner, in dem sich die Pk3-Dateien befinden. Bei RtCW ist dies "Main", bei Quake3 heisst dieser Ordner "Baseq3"
set map_path=c:/games/Retur~1/main/maps/test.map	Dies ist der Pfad zur Map-Datei, die compiliert werden soll
set q3map_path=c:/q3map2/q3map2.exe	Dies ist der Pfad zur Q3map2.exe, wenn du sie in den Ordner "C:\Q3map2" entpackt hast
set gen_options=	Voreinstellungen, die du übernehmen kannst
%q3map_path%	Voreinstellungen, die du übernehmen kannst
%q3map_path%	Voreinstellungen, die du übernehmen kannst
%q3map_path%	Voreinstellungen, die du übernehmen kannst
pause	Dies ist das Ende der Bat-Datei

Nun stellst du dir sicher die Frage, wieso man hier alles selbst einstellen muss - die Q3Map2 ist doch in den GtkRadiant 1.211 integriert. Damit hast du recht - jedoch lässt sich mit unserer Bat-Datei viel mehr einstellen und individuell anpassen.

Wenn du nun an deiner Bat-Datei etwas ändern willst, klicke sie im Explorer mit der rechten Maustaste an und wähle "bearbeiten" in dem Untermenü.

Wenn das Erstellen der Bat-Datei bei dir nicht funktioniert hat, findest du hier eine [Zip-Datei](#), die eine solche Bat-Datei enthält.

[zurück zum Abschnitt Q3Map2](#)

[zurück zur Hauptseite](#)





Terrain Lightmappen:

Nachdem wir uns nun die Bat-Datei eingerichtet haben, wollen wir auch gleich loslegen, indem wir uns dem Terrain widmen. Wie du sicher schon gesehen hast, werden Schatten usw. nicht genau auf dem Boden dargestellt. Das wird sich durch die Q3Map2 ändern. Um das ganze zu bewerkstelligen, benötigst du nur sehr wenig Aufwand. Also los:

Dazu öffnest du die Datei "Common.shader" im "Scripts"- Ordner und suchst nach dem Wort "Terrain". Nun müsste es ungefähr so aussehen:

```
textures/common/terrain
{
    surfaceparm grassesteps
    surfaceparm nodraw
    surfaceparm nomarks
    surfaceparm nolightmap
}

textures/common/terrain2
{
    surfaceparm grassesteps
    surfaceparm dust
    surfaceparm nodraw
    surfaceparm nomarks
    surfaceparm nolightmap
}
```

Nun verändern wir den Shader, dass er so aussieht (keine Angst, es sind nur 2 ganz kleine Änderungen):

```
textures/common/terrain
{
    surfaceparm grassesteps
    surfaceparm nodraw
    surfaceparm nomarks
    surfaceparm nolightmap
    q3map_terrain
}

textures/common/terrain2
{
    surfaceparm grassesteps
    surfaceparm dust
    surfaceparm nodraw
    surfaceparm nomarks
    surfaceparm nolightmap
```

q3map_terrain

}

Wir haben also jeweils nur "q3map_terrain" dazu geschrieben. Was bringt das? Jetzt wird auch unser Terrain mit Lightmap berechnet, und nichtmehr nur mit Vertex, wie das bisher der Fall war.

Hier habe ich dir mal ein Terrain normal und mit der Q3Map2 compiliert, hier kannst du sehen, dass der Baun keinen Schatten wirft:



Nun das gleiche Bild mit der Q3Map2 compiliert:



und siehe da, der Schatten ist zwar nicht sonderlich gross, aber doch deutlich sichtbar.

[zurück zum Abschnitt Q3Map2](#)

[zurück zur Hauptseite](#)

183759



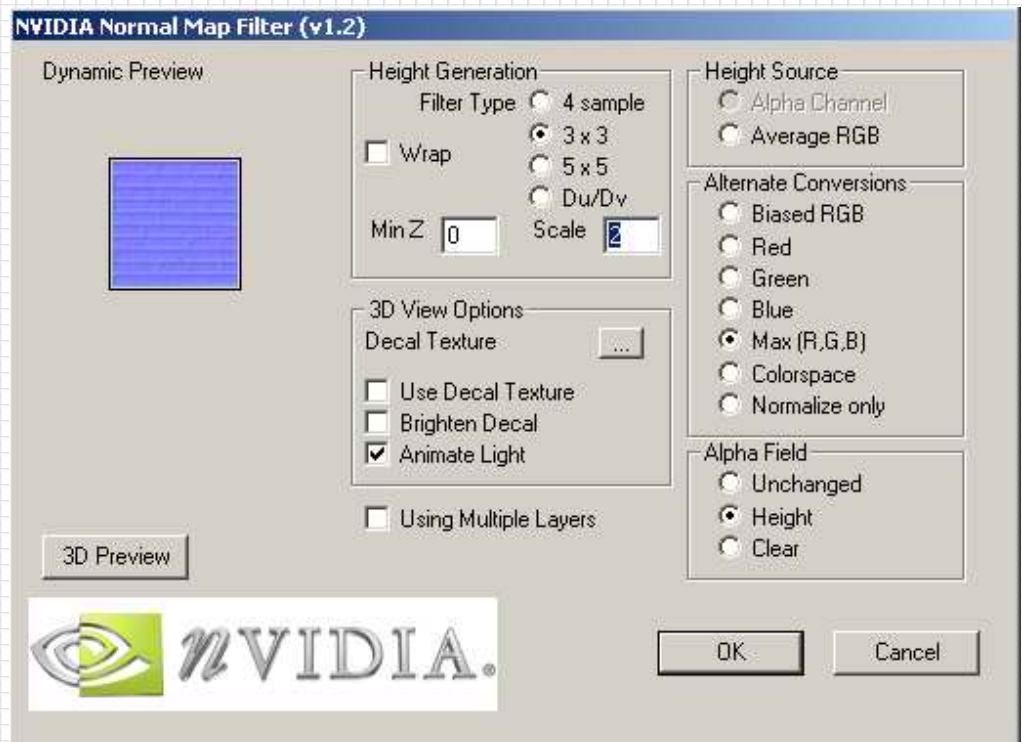
Bump Mapping :

Zuerst stellt sich wohl die Frage, was "Bump Mapping" überhaupt ist. Bump Mapping ist eine Technik, bei der auf den Brush nicht nur die Textur gelegt wird, sondern eine Normalmap, welche der Textur Höhen und Tiefen verleiht. Damit ist es also möglich, z.B. eine Steinwand plastischer wirken zu lassen.

Nun wollen wir anfangen. Natürlich benötigen wir zuerst eine Textur. Dazu kannst du eine Textur aus RtCW nehmen, oder eine eigene verwenden. Ich verwende diese Textur:



So, wie bekommt man nun diese Normalmap hin? Oder woher kommt sie? Ganz einfach, diese kann man mittels Photoshop selbst erstellen. Dazu benötigst du eine Textur, die möglichst dunkel ist und einen Filter, der aus dem Hause NVidia kommt. Wenn du keine NVidia-Grafikkarte dein Eigen nennt, funktioniert dieser Filter leider nicht. Aber dazu kommen wir später. Nun lädst du dir zuerst besagten [Filter](#) herunter. Diesen findest du auf der Seite unter "Attatchments". Diese Datei kopierst du dann in den Ordner "Photoshop X/Zusatzmodule/Effekte/". Wobei natürlich "Photoshop X" dein Photoshop-Ordner ist. Nun startest du Photoshop und lädst die Textur. Nun klickst du auf "Filter" und ganz unten auf "nvtools". Nun öffnet sich folgendes Fenster:



Kommt bei dir folgender Fehler:
"Image width or height is not power of 2"?

Dann hat deine Textur leider die falschen Abmessungen. Möglich sind alle Vielfache von 2 - also 256x512, 512x512, 1024x1024 usw.

Oben links kannst du schon eine kleine Vorschau unserer Textur ansehen. Bei "Height Generation" stehen folgende Filter zur Auswahl 3 x 3 bis zu 5 x 5. Wir benutzen 5 x 5, da es die beste und feinste Einstellung ist - schliesslich werden wir hinterher mit einer schöneren Bummap belohnt.

Unter "Height Source" versteckt sich lediglich die Quelle, von wo aus die Normalmap-Textur errechnet wird. Hätten wir z.B. einen Alpha-Kanal, der Höhen und Tiefen angibt, könnten wir diesen benutzen. Leider haben wir diese Auswahl nicht, also belassen wir es bei Average RGB. Das bedeutet, die Normalmap wird aus dem ganz normalen Bild berechnet. Nun drückst du auf "OK" und schon wird die Normalmap berechnet:



Jetzt speicherst du die Normalmap unter einem anderen Namen ab, da sonst unsere alte Textur überschrieben wird. Und da wir beide Texturen brauchen, wäre das nicht gut. Am besten, du speicherst die normalmap als *.tga mit 32 Bit Farbtiefe. Nun stellt sich natürlich die Frage, was zu tun ist, wenn man keine NVidia-Karte hat. Nun gibt es 2 Möglichkeiten, zum einen kann man fragen, ob es von seinem Grafikkartenhersteller auch einen solchen Filter gibt - oder man macht sich diese Normalmap einfach selbst. Wenn du die Normalmap in Photoshop öffnest, wirst du feststellen, dass der Blaue Kanal komplett weiss ist. Mit etwas Übung lässt sich das auch per Hand machen.

Nun benötigen wir noch den Shader:

```
textures/bump-test/bricka
{
  qer_editorimage textures/bump-test/brick.tga
  q3map_normalimage textures/bump-test/brick_bump.tga
  {
    map $lightmap
    rgbGen identity
  }
  {
    map textures/bump-test/brick.tga
    blendFunc GL_DST_COLOR GL_ZERO
    rgbgen identity
  }
}
```

Nun speicherst du den Shader unter dem Namen "Bump" im Verzeichnis "Scripts" ab. Nun öffnest du noch die Datei "shaderlist.txt" und trägst ganz unten "bump" ein. Also ohne die Endung ".shader". Nun kannst du die Textur benutzen. Natürlich musst du jetzt die Q3map2 zum compilieren benutzen, da die normale Q3map.exe diesen Feature nicht besitzt. Um zu sehen, ob die Sache letztlich funktioniert, musst du die Map mit dem Befehl

```
/devmap deinemap ENTER
/r_lightmap 1 ENTER
```

eingeben. Wenn dir der Schatten zu verwaschen oder zu undeutlich erscheint, kannst du die Map mit einem niedrigeren Lightmapscale berechnen lassen, dann werden die Schatten feiner berechnet. Wenn dir das aber zu lange dauert, kannst du einfach den Brush, auf dem die Bump-Textur liegt zu einer func_group machen und bei den Entities folgenden Key eingeben:

```
Key: "lightmapscale"
Value: "0.1"
```

183759



Kompilier-Parameter:

Für die Q3Map2 gibt es viele verschiedene Möglichkeiten, eine Map zu berechnen. Einige davon kennst du sicher aus dem "BSP"-Menüpunkt im Radiant. Hier wurden nur einige Compile-Möglichkeiten zusammengestellt. Natürlich kann dies schon ausreichen, um eine Map zu erstellen. Jedoch kann die Q3Map2 soviel mehr, dass es sich wirklich lohnt, ein bisschen herzuspielen - wesentlich schönere Maps werden es dir danken!

Nachdem ich dir ja schon gezeigt habe, wie eine Batch-Datei zum Compilieren aufgebaut ist, will ich dir jetzt hier die wichtigsten Befehle zeigen, was man mit der Q3Map2 noch alles anstellen kann:

In der Q3Map2 gibt es wie bei der normalen Q3map 3 Compile-Stages. Der [BSP-Compile](#), der [Vis-Compile](#) und der [Light-Compile](#). Für jeden dieser drei Teile gibt es eigene Parameter. Daher sortiere ich diese Parameter gleich nach ihrem Anwendungsgebiet.

bsp_var:

- meta	der Meta-Befehl ist sehr praktisch, wenn es darum geht, FPS einzusparen. Hier werden Flächen zusammengefügt, so dass r_Speeds eingespart werden können. Wichtig: Wenn du hinterher die Map mit "bspc" compilierst, kommt es zum Abbruch, wenn du nicht "-forcesidesvisible" in die DOS-Zeile hinzufügst.
- threads x	gibt an, wie viele CPUs dein System hat. Bei "normalen" PCs steht statt dem X eine 1. Habt ihr mehrere CPUs, könnt ihr hier eine Anzahl eingeben. Dadurch werden alle CPUs angesprochen.
- leaktest	Leaks dürftest du schon zur Genüge kennen. Tritt also während dem Compile ein Leak auf, stoppt der Compile.
- patchmeta	Hier werden Patches zusammengefügt und dadurch r_speeds eingespart. Das gleiche wie der normale - Meta Befehl, allerdings eben nur für Patches.
- v	Mit diesem Befehl wird der komplette Compile-Vorgang angezeigt. Zur Fehlersuche bestens geeignet!
-samplesize x	Standart ist 16. Je kleiner der Samplesize gewählt wird, desto feiner wird die Map compiliert. Jedoch frisst es viel Zeit und bei grösseren Maps kann es zum Abbruch kommen.

- nocurves	Curves werden nicht mitberechnet
- nodetail	Detail-Brushes werden nicht mitberechnet
- nowater	Waser, Lava und Schleim wird nicht mitberechnet

vis_var:

- fast	der Compile wird schneller. Dafür werden Hint-Portale usw. übersprungen und nicht mitberechnet
- merge	bevor die sichtbaren Flächen berechnet werden, werden die BSP-leaves zusammengefügt. Dadurch wird der Compile schneller, aber wesentlich mehr Polygone werden dargestellt als notwendig.
- nosort	Das Sortieren der Portale nach Grösse wird übersprungen. Dadurch wird die Kalkulation der sichtbaren Flächen schneller, da die komplexeren Portale Informationen der kleineren Portale übernehmen können, also Position usw.
- v	Auch hier gibt es wieder den Befehl, dass der gesamte Compile dargestellt wird.

light_var:

- bounce (n)	der gesamte Light-Compile wird n-mal durchgeführt. Nach jedem Bounce wird eine bsp-Datei ausgegeben, dadurch kann man den compile ohne Datenverlust abbrechen. Natürlich dauert dadurch der gesamte Compile wesentlich länger.
- debugsurfaces	Hier werden die Lightmaps eingefärbt. Jede Oberfläche erhält eine eigene Farbe.
- extra	ist die Abkürzung für "- super 2". Wenn du also zu faul bist, "- super 2" einzutippen, tippe einfach "- extra"

- extrawide

Wenn du zu faul bist, "- supper 2 - filter" einzugeben, kannst du statt dessen "- extrawide" eingeben.

- fastbounce

das gleiche wie der normale "- bounce" Parameter. Jedoch werden hier wie beim "- fast"-Parameter auch Hints nicht mitberechnet.

- patchshadows

Patches werfen Schatten

- v

Auch hier gibt es wieder den Befehl, dass der gesamte Compile dargestellt wird.

- smooth

Schatten werden feiner berechnet. Benötigt natürlich wesentlich mehr Zeit

- super x

Standart ist 2 - also sollte nur mit Zahlen >2 angegeben werden. Dadurch wird die komplette Map in einem feineren Raster berechnet. Benötigt wesentlich mehr Zeit, wesentlich mehr Arbeitsspeicher wird benötigt.

- samplesize (n)

Dadurch wird die Grösse der Lightmap geändert. Je kleiner, desto feiner. Der Compile dauert auch hier wesentlich länger. Der Befehl wird von Oberflächen ignoriert, die im Shader eine genaue Grösse der Lightmapgrösse "q3map_lightmapsamplesize X" stehen haben.

Vielleicht wundest du dich, wieso gerade dem Light-Prozess so viele Variablen spendiert wurden. Ganz einfach, die Q3Map2 arbeitet hauptsächlich in diesem dritten Sektor. Natürlich gibt es noch viel mehr an solchen Variablen, diese kannst du dir im [Q3Map2 - Handbuch](#) durchlesen.

[zurück zum Abschnitt Q3Map2](#)

[zurück zur Hauptseite](#)

183759



Entity - keys:

Nichtnur bei der Batch-Datei lässt sich einiges mit der Q3Map2 ändern. Natürlich auch im Radiant selbst.

- `_celshader`

Hier werde ich zu gegebener Zeit ein extra Tutorial schreiben. Celshading bedeutet, dass alle Kanten einen schwarzen Rand bekommen. Das ganze hat dann einen leichten Comic-Charakter.

- `modelscale:`

Hiermit kannst du die Grösse eines Models skalieren, also vergrößern und auch verkleinern. Dazu fügst du wie gewohnt ein "misc_model" ein, und gibst bei den Entities ein:

Key: "modelscale"
Value: "X X"

X X steht dabei für z.B. 3 3 oder eine andere Zahl. Im Radiant (Gtk) werden ab Version 1.2.4 die Änderungen gleich angezeigt. Hast du eine ältere/andere Version, kann es vorkommen, dass du das Resultat erst nach dem Compile sehen kannst.

- `angles`

Wie der Name schon sagt, kannst du hiermit Modelle drehen.Dazu gibst du bei deinem misc_model einfach ein:

key: "angles"
Value: " X Y Z"

wobei X Y und Z natürlich die Koordinaten für den dreidimensionalen Raum darstellen.Den Resultat sieht man momentan aber erst nach dem Compile.

[zurück zum Abschnitt Q3Map2](#)

[zurück zur Hauptseite](#)





Allgemeine Fragen & Antworten:

Sicher hast du auch die ein oder andere Frage zum GtkRadiant, zum Mapping usw., die bisher nicht erklärt worden sind. Ich hoffe, dass diese Fragen hier geklärt werden:

I. Installation:

[Wie installiere ich mein Spiel richtig?](#)

[Wie installiere ich den Radiant richtig?](#)

[Welche sind die besten Voreinstellungen für den Radiant?](#)

II. GtkRadiant:

[Gibt es Änderungen zu älteren Versionen?](#)

[Muss ich den GtkRadiant benutzen, oder geht z.B. für SOF2 auch der SOF-Radiant?](#)

III. Texturen:

[Wie setze ich Hint-Portale richtig?](#)

[Wie setze ich Area-Portale richtig?](#)

[Wozu ist die Caulk-Textur da?](#)

[Welchen Zweck hat die Origin-Textur?](#)

[Welche Clip-Textur benutze ich wann?](#)

[Wie lege ich eine Textur auf EINE Seite eines Brushes?](#)

[Wie füge ich eigene Texturen ein?](#)

[Was für Grafiken darf ich benutzen?](#)

[Welche Grafik-Formate unterstützt der Radiant?](#)

[Welche Grösse darf eine Textur haben?](#)

[Wieso haben manche Texturen einen weissen Rand?](#)

[Warum darf ich keine Texturen benutzen, die "progressive" gesichert wurden?](#)

IV. Compilierung:

[Wieso dauert das bei mir so ewig?](#)

[Bei meiner Map gibt es keine Bsp-Datei, sondern nur eine Map-Datei?](#)

[Manche Fehler sind weg, wenn ich mit der Q3Map2 compile?](#)

[Was ist so toll an der Q3Map2?](#)

[Die Map sieht nach dem Compile mit der Q3Map2 genauso aus, wie nach einem Compile mit der normalen Q3map?](#)

V. Konstruktion:

- [Wie öffne ich eine Map?](#)
- [Was ist ein Prefab?](#)
- [Was ist ein Entity?](#)
- [Was ist ein Model?](#)
- [Wie hoch ist eine Spielfigur?](#)
- [Wie weit kann ein Spieler springen?](#)
- [Wie kann ich schnell mehrere Brushes kopieren?](#)
- [Wie kann ich einen verdeckten Brush selektieren?](#)
- [Wieso werfen meine Säulen keine Schatten?](#)
- [Wieso kann ich durch meine Säulen hindurchlaufen?](#)
- [Wieso kann ich durch meine Models hindurchlaufen?](#)

VI. Performance:

- [Was sind VIS-Blocker und wie funktionieren sie?](#)
- [Wie erhöhe ich die Performance meiner Map?](#)

VII Easygen:

- [Wieso darf ich nicht mit CSG Substract arbeiten?](#)
- [Welche von den ganzen Easygen-Dateien brauche ich denn für meine Pk3-Datei?](#)
- [Wieso kann ich keine Täler erstellen?](#)

VIII. Sonstiges:

- [In den Tutorials stehen manchmal Entities beschrieben, die es bei mir garnicht gibt?](#)
- [Wieso kann ich durch den Himmel andere Räume sehen?](#)
- [Was ist eine Pk3-Datei?](#)
- [Was ist das Grid?](#)
- [Meine func_groups sind nach der Installation des GtkRadiant 1.2.11 nicht mehr da?](#)
- [Was bringen eigentlich diese PlugIns im Radiant?](#)
- [Warum muss ich ständig "sv_pure 0" eingeben?](#)
- [Was bringt eine Pk3-Datei?](#)
- [Welchen Zweck hat die shaderlist.txt?](#)
- [Sagmal, hast du eigentlich zuviel Zeit, Haradarki?](#)

I. Installation

Frage: Wie installiere ich mein Spiel richtig?

Antwort: Hier gibt es sicher keine Patentlösung. Zumindest solltest du dir die neuste Version des Radianten herunterladen, bevor du irgendeine veraltete Version installierst. Bei der Installation des Spiels selbst solltest du darauf achten, dass, wenn möglich, keine Leerzeichen enthalten sind. Das ist zwar für den Radiant nicht unbedingt wichtig (ein solcher Fehler wurde mittlerweile behoben) jedoch mit der Q3Map2 und der DOS-Box tut man sich doch schwer, wenn Leerzeichen enthalten sind und die Sache dann schon gern ausartet. Also könntet ihr statt "Return To Castle Wolfenstein" einfach nur "Wolfenstein" oder "RtCW" nennen. Vorher musste im Verzeichnisname der Name des Hauptprogramms angegeben werden.

C:/Games/blutigesballerspiel1/

Nichtnur, dass es ohnehin eine sinnlose Betitelung ist, sollte man wenigstens selbst wissen, was dahinter steckt. So wäre es deutlich besser:

C:/Games/Soldier Of Fortune - Double Helix
C:/Games/Sof2

C:/Games/Return To Castle Wolfenstein
C:/Games/RtCW

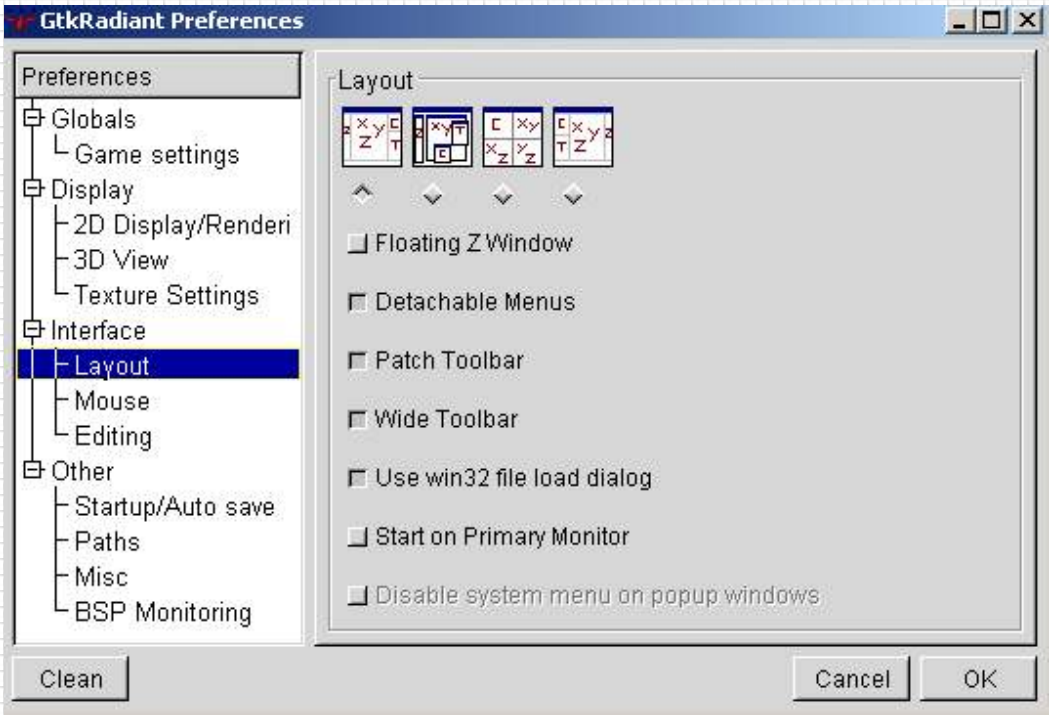
C:/Games/Jedi Knight 2
C:/Games/Jk2

Frage: Wie installiere ich den Radiant richtig?

Antwort: Hier solltest du lediglich darauf achten, dass du nicht 10 verschiedene Versionen des Radianten installiert hast. Benutze stets die neuste Version. Damit hast du weniger Bugs, und die meisten Tutorials verwenden vielleicht Features, die es in älteren Radiant-Versionen noch nicht gab.

Frage: Welche sind die besten Voreinstellungen für den Radiant?

Antwort: Hier siehst du ein Bild der Preferences, diese findest du unter "Edit -> Preferences"



Ausserdem solltest du noch bei "Mouse" angeben, ob du eine 2- oder eine 3-Tasten-Maus besitzt. Bei einer 3-Tasten-Maus ist die mittlere Maustaste dazu da, um Texturen auf eine selektierte Seite eines Brushes zu übertragen.

II. GtkRadiant:

Frage: Gibt es Änderungen zu älteren Versionen?

Antwort: Natürlich, sonst bräuchte man ja keine neue Version. Aber natürlich gibt es Versions-Sprünge, wo sich ausser der Versions-Nummer nichts ändert. Meist werden hier "nur" kleinere Bugs beseitigt. Einen grossen Sprung gibt es bei dem Gtk Radiant 1.2.11 hier wurden einige Änderungen vorgenommen:

- Die Shortcuts für func_groups hat sich geändert. Func_groups wurden früher mit "SHIFT" und der linken Maustaste angeklickt. Jetzt wird mit "SHIFT" und der linken Maustaste lediglich ein Brush der func_group selektiert. Um alle Teile zu selektieren, muss man "SHIFT" + "STRG" + linke Maustaste drücken.
- Die Q3Map2 wurde eingebaut. Damit wird Bumpmapping und Cel-Shading und noch einige andere Features unterstützt. Da dies ein recht komplexes Thema ist, habe ich dazu ein [eigenes Thema](#) geschrieben.
- Ab sofort wird der Lichtkegel von Light-Entities dargestellt. So kann man schon im Radiant sehen, wie weit ein Licht einen Raum ausleuchtet, bzw. welche Ecken vielleicht noch ein bisschen mehr Licht vertragen könnten.

Frage: Muss ich den GtkRadiant benutzen, oder geht z.B. für SOF2 auch der SOF-Radiant?

Antwort: Natürlich kannst du den Radiant benutzen, der dir am besten gefällt. Jedoch muss man sagen, dass bisher nur der GtkRadiant weiter entwickelt wird, während andere Radianten, z.B. der Q3Radiant seit rund einem Jahr nicht mehr weiterentwickelt wird. So kann es sein, dass der Gtk neue Features bietet, während du dich mit einer alten Version herumplagst. Dazu solltest du einfach von Zeit zu Zeit mal auf queradiant.com surfen und die changelogs durchlesen.

III. Texturen:

Frage: Wie setze ich Hint-Portale richtig?

Antwort: Dazu solltest du dir am besten einmal das Thema über [Hint-Portale](#) durchlesen. Hier wird eigentlich schon fast alles erklärt, was an Wissen notwendig ist, um mit Hint-Portalen zu arbeiten. Hint-Portale werden so gesetzt, dass sie alle anliegenden Brushes überschneiden. Ausserdem sollten sie mind. 16 Units dick sein. Ausserdem MUSS ein VIS-Blocker aktiv sein, d.h. wenn man in einen langen, geraden Gang ein Hint-Portal setzt, bringt es nichts. Es muss verwinkelt sein, so dass ein Teil der Architektur komplett ausserhalb des Sichtfeldes befindet.

Frage: Wie setze ich Area-Portale richtig?

Antwort: Auch dazu habe ich ein eigenes [Thema](#) geschrieben, welches die [Area-Portale](#) genau beschreibt.

Frage: Wozu ist die Caulk-Textur da?

Antwort: Die Caulk-Textur ist wohl eine der wichtigsten Texturen überhaupt. Während Area- Hint- und sonstige Texturen nur recht selten zum Einsatz kommen, wird die Caulk-Textur fast ständig eingesetzt. Sie kennzeichnet Stellen, die nicht mitberechnet werden. Also z.B. nicht sichtbare Hinterseiten von Brushes, Unterseiten von Terrains usw. Zwar caulkt der Radiant diese Flächen automatisch, jedoch wird dies nicht 100%tig gemacht, so dass man manuell doch noch ein Quäntchen Performance aus seiner Map schlagen kann. So wird nicht nur die Compile-Dauer wesentlich verkürzt, auch die FPS steigen enorm an, da der Radiant nicht alle Aussenseiten automatisch caulkt. So werden z.B. nicht sichtbare Seiten noch berechnet, wenn Licht auf diese Seiten fällt. Dann wird trotzdem die Lightmap usw. berechnet.

Frage: Welchen Zweck hat die Origin-Textur?

Antwort: Der Brush, der mit der Origin-Textur belegt ist, legt die Ursprungskordinaten für Bewegungen fest. So benötigt man einen Origin-Brush, um eine func_rotating oder ein func_pendulum zu steuern. Aber auch misc_models greifen auf die Origin-Textur zurück. Willst du z.B. ein Model an einer Türe abringen, welches sich mit der Türe bewegt, kannst du dies mit dem Einsatz der Origin-Textur lösen. So bildet der Brush mit der Origin-Textur den Ersatz für das Entity "misc_model".

Frage: Welche Clip-Textur benutze ich wann?

Antwort: [Hier](#) findest du eine [Übersicht aller Clip-Texturen](#) für RtCW. Darunter sollte sich auch diejenige verbergen, die du benötigst.

Frage: Wie lege ich eine Textur auf EINE Seite eines Brushes?

Antwort: Du markierst mit "SHIFT" + "STRG" + linke Maustaste die Seite, des Brushes, auf die du die Textur legen willst. Nun drückst du die Taste "T" um das Texturen-Fenster zu öffnen. Hier suchst du dir die passende Textur aus.

Frage: Wie füge ich eigene Texturen ein?

Antwort: Das ist ganz einfach. Du erstellst in deinem Main-Ordner einen weiteren Ordner, den du "textures" nennst. Hier kannst du dann Order erstellen, die z.B. "decke", "Boden" usw. In diese Unterordner kannst du dann deine Texturen kopieren. Wenn du dann den Radiant startest, tauchen diese Ordner "Decke", Boden" usw. in der Texturen-Liste auf.

Frage: Was für Grafiken darf ich benutzen?

Antwort: Zunächst einmal solltest du dir überlegen, was für Grafiken du haben willst. So sollte z.B. kein Grafikenwirwar aus verschiedenen Typen, z.B. Future- und dann plötzlich Wild-West-Texturen verwenden. Ausserdem solltest du nachsehen, ob die Texturen kachelbar sind, d.h. ob es zu hässlichen Übergängen kommt, wenn sich die Textur wiederholt.

Ausserdem solltest du immer Copyrights und Rechtslagen unbedingt berücksichtigen! Bilder von HKs haben in Maps nichts verloren. Wenn du solche Bilder trotzdem verwendest, kannst du richtig Ärger bekommen. Ansonsten bleibt nur zu sagen, dass du Copyrights beachten solltest. Aber keine Panik, es gibt noch genug coole Texturen, die du verwenden kannst!

Frage: Welche Grafik-Formate unterstützt der Radiant?

Antwort: Solange du normale Texturen verwendest, reicht das JPEG-Format völlig aus. Wenn du hingegen Texturen haben willst, die du über einen Shader modifizierst, solltest du das TARGA-Format benutzen. Dieses ist zwar wesentlich grösser, ist dafür aber genauer und unterstützt einige Features, die es bei JPG nicht gibt.

Frage: Welche Grösse darf eine Textur haben?

Antwort: Im Grund gibt es keine richtige Beschränkung. Aber da sonst deine Map sehr gross wird, solltest 512 x 512 Pixel die Obergrenze sein. An sonsten solltest du nur beachten, dass die Texturen genauso lang wie breit sind, also z.B. 128 x 128, 256 x 256 usw. Wenn du eine Normalmap aus einer anderen Textur erstellen willst, MUSS die Grösse ein Vielfaches von 2 sein. Also z.B. darf sie nicht die Abmessung 128 x 111 haben.

Frage: Wieso haben manche Texturen einen weissen Rand?

Antwort: Diese Texturen besitzen einen Shader, d.h. ein Shader ist eine Text-Datei, die die Eigenschaften von der Textur beschreibt oder verändert. So kann ein Shader der Textur eine völlig neue Gestalt geben, z.B. metallischen Glanz, Transparenz usw. Manche Shader sind allerdings sehr rechenintensiv. Daher solltest du eher mit Bedacht Shader einsetzen

Frage: Warum darf ich keine Texturen benutzen, die "progressive" gespeichert wurden?

Antwort: "Progressive" wurde für das Web entworfen. So sieht man zu Beginn die Grafik nur sehr grob, d.h. nur die Umrisse. Je mehr von der Grafik geladen wird, desto besser wird die Grafik sichtbar. Dieser Effekt ist natürlich in Spielen hässlich. Daher bricht der Compiler ab, wenn eine Grafik "progressive" gespeichert wurde.

IV. Compilierung:

Frage: Wieso dauert das bei mir so ewig?

Antwort: Hier kann man eigentlich kein genaues Limit angeben. Maps zu compilieren kann wirklich lange dauern - manchmal sogar mehrere Tage. Es kommt natürlich darauf an, mit welchen Parametern du die Map compilierst. Zu einem Test während du die Map baust, würde ich dir zu einem Compile mit bsp -vis raten. Hier wird die komplette Lichtberechnung übersprungen. Diese ist aber zum Bau sowieso nicht wichtig. Das kannst du noch später machen, wenn du die Ausleuchtung selbst testest. Jedoch gibt es ein paar Tips, wie man seine Map schneller berechnen kann. Zum einen solltest du so sauber wie möglich mappen, also möglichst keine Brush-Überschneidungen erzeugen und auch nicht mit CSG Subtract arbeiten. Auch wenn du alle nicht sichtbaren Brush-Seiten im Radiant mit der Caulk-Textur belegst, dürfte der Compile etwas rasanter von statten gehen.

Frage: Bei meiner Map gibt es keine Bsp-Datei, sondern nur eine Map-Datei?

Antwort: Die Map wurde noch nicht compiliert. Die Map-Datei ist nur für den Radiant da, um die Map zu laden und zu verändern. Damit du die Map spielen kannst, musst du sie in ein Format umwandeln, dass es für die Q3-Engine lesbar macht, d.h. die Map muss compiliert werden. Wenn du das allerdings gemacht hast, hast du vielleicht ein Leak in der Map oder einen anderen Error. Dazu solltest du nach dem Compile das DOS-Fenster ansehen, ob dort etwas mit "Error" oder "Leak" zu finden ist.

Frage: Manche Fehler sind weg, wenn ich mit der Q3Map2 compile?

Antwort: Ja, manche Fehler tauchen beim Compilieren mit der Q3Map2 garnichtmehr auf. Das liegt daran, dass die Q3Map2 wesentlich mehr Formate unterstützt und auch generell etwas besser ist.

Frage: Was ist so toll an der Q3Map2?

Antwort: Die Q3Map2 ist deshalb so "toll", weil man mit ihrer Hilfe Bump-Mapping, Cel-Shading und andere Features in den Maps verwenden kann. Diese Techniken waren vorher in der Q3-Engine nicht möglich.

Frage: Die Map sieht nach dem Compile mit der Q3Map2 genauso aus, wie nach einem Compile mit der normalen Q3map?

Antwort: Hier liegt es wohl daran, dass du die Möglichkeiten der Q3map2 nicht ausreizt. Du solltest dir vielleicht einmal [dieses Tutorial](#) durchlesen und dann wirst du schon sehen, dass es da einen Unterschied gibt.

V. Konstruktion:

Frage: Wie öffne ich eine Map?

Antwort: Ok, eigentlich klingt diese Frage recht dumm - aber es gibt tatsächlich eine Sache, bei der man gehörig viel falsch machen kann. Es gibt zwei Möglichkeiten, um eine Map zu öffnen:

- unter File -> Open

Mit "open" wird eine neue Map geöffnet, die alte wird komplett überschrieben.

- unter File -> Load

Mit "load" wird zu der bestehenden Map eine andere hinzugeladen. Das bedeutet, beide Maps befinden sich dann

Frage: Was ist ein Prefab?

Antwort: Ein Prefab ist eine Ansammlung von Brushes, also einer kleinen Konstruktion z.B. ein einzelnes Zimmer, eine Telefonzelle oder ähnliches. Prefabs werden von vielen Internet-Seiten zum Download angeboten. Diese kann man dann in seine Map einbauen, in dem man den Radi startet und dann auf "EDIT" -> "LOAD PREFAB" klickt. In dem neuen Fenster öffnet man dann das gewünschte Prefab. Meistens befindet sich dann dieses Prefab irgendwo in der Map, d.h. man muss es noch an die richtige Stelle setzen.

Frage: Was ist eigentlich ein Entity?

Antwort: Es gibt 2 Arten von Entities:

- Brush-Entities:

Ein Brush-Entity besteht aus Brushes. So z.B. ein Brush mit der "common/trigger"-Textur wird in eine trigger_multiple umgewandelt. Nun wurde aus dem normalen Brush ein Brush-Entity. Ein paar Beispiele für andere Brush-Entities: trigger_push, func_trains, func_buttons, func_bobbing, func_pendulum, func_plat. Da die Anzahl der Entities beschränkt ist, solltest du versuchen, manche Entities (wenn möglich) zu einem einzigen hinzuzufügen. So zählen mehrere Brush-Entities als ein einziges Entity.

- rechteck-Entities:

Leider gibt es momentan noch keinen technisierten Namen für diese Gruppe von Entities. Aber das macht nichts, bleiben wir bei "rechteck-Entities". Sobald du mit der rechten Maustaste klickst und ein Entity aus der Liste wählst, also z.B. target_give, aber auch sämtliche Ammo- und Weapon-Entities zählen zu dieser Gruppe. Aber auch natürlich die ganzen path_corner, target_speaker, target_give, target_kill, misc_model sind rechteck-Entities.

Übrigens kommt es zu einem üblen Fehler, wenn du mehr als 255 Entities in einem sichtbaren Raum auftauchen. So kannst du z.B. deine Waffe abfeuern, ohne dass du ein Projektil oder eine Explosion sehen kannst. Überhalb dieser 255 Entities werden also alle anderen ausgeblendet, bis die vorhandenen 255 verschwinden. Dies kann dadurch geschehen, dass sie in einen Teil der Map wandern, der nicht einsehbar ist, oder aber wenn eine Rakete an einer Wand explodiert. Also solltest du solche Fehler haben, solltest du deine Map etwas verschachtelter aufbauen..

Frage: Was ist ein Model?

Antwort: Ein Model ist ein Baustein, den man mittels "misc_model" einfügen kann. Models werden nicht mit dem Radiant erstellt, sondern mit Programmen wie 3D Studio Max, Bryce oder Maya. Diese kosten natürlich eine ganze Stange Geld. Es gibt aber auch sogenannte "low-Poly" Programme, die umsonst sind, z.B. Milkshape. Hiermit kann man Models erstellen, die knapp über 1000 Polygone (Dreiecke) haben.

Frage: Wie hoch ist eine Spielfigur?

Antwort: In RtCW ist der Spieler ungefähr 72 Units hoch. Das kannst du selbst testen, indem du ein misc_model erstellst und dann unter "size" das Model "bjsize" lädst. Das ist die Größe von dem Spieler.

Frage: Wie weit kann ein Spieler springen?

Antwort: Wird der Schwerkraftswert von 800 nicht geändert, kann der Spieler ungefähr 260 Units weit springen. Jedoch ist es sinnvoller, einen geringeren Wert anzugeben.

Frage: Wie kann ich schnell mehrere Brushes kopieren?

Antwort: Du ziehst um alle Brushes, die du selektieren willst, einen grossen Brush, der alle Brushes einschliesst. Nun wählst du in der Menüleiste "Selection" -> "Select" -> "Select Inside". Nun sind alle Brushes markiert, die in dem Brush eingerahmt waren. Nun kannst du die Leer-Taste drücken um die Brushes zu kopieren. Die Kopie dieser Brushes wird um ein Unit nach links und ein Unit nach unten versetzt dargestellt.

Frage: Wie kann ich einen verdeckten Brush selektieren?

Antwort: Manchmal ist es ziemlich schwer, einen bestimmten Brush zu selektieren, z.B. wenn einige andere diesen einen Brush verdecken. Wenn du die Taste "ALT" + "SHIFT" + linke Maustaste auf den Brush klickst, werden nach und nach alle hintereinander liegenden Brushes selektiert. Also z.B. werden in der TOP-Ansicht alle Brushes von oben nach unten durchselektiert, die an der Stelle liegen, wo man mit dem Mauszeiger klickt.

Frage: Wieso werfen meine Säulen keine Schatten?

Antwort: Wenn du eine Säule aus einem Zylinder baust, so erzeugt dieser keinen Schatten, da curves alleine für die Engine unsichtbar sind. Du benötigst also den Einsatz der "Caulk"-Textur. Wenn du damit einfach einen Block im Inneren der Säule erstellst, reicht dies aus. Nun wirft der Block mit der "Caulk"-Textur einen Schatten.

Frage: Wieso kann ich durch meine Säulen hindurchlaufen?

Antwort: Auch hier liegt es daran, dass curves alleine für die Engine unsichtbar sind. Nun benötigst du allerdings keine "Caulk"-Textur, sondern einen Brush mit der Clip-Textur. Diesen ziehst du einfach komplett um die Säule herum.

Frage: Wieso kann ich durch meine Models hindurchlaufen?

Antwort: Da die Engine eigentlich nicht weiss, wie gross ein Model ist, kann man durch das Model hindurchlaufen. Das einzige, was die Engine weiss, ist der Ursprung des Models, nämlich dem Entity "misc_model". Also hilft nur eins, das Model komplett in einen Clip-Brush zu verpacken.

VI. Performance:

Frage: Was sind VIS-Blocker und wie funktionieren sie?

Antwort: VIS-Blocker sind im Prinzip nur Brushes, die die Sicht auf ein grösseres Gelände, einen grossen Raum usw. verdecken. Somit werden diese Level-Teile erst dann berechnet, wenn man diesen VIS-Blocker passiert.

Frage: Wie erhöhe ich die Performance meiner Map?

Antwort: Zunächst solltest du deine Map sehr verwinkelt bauen. Es macht keinen Sinn, ein Level wie Q3DM17 mit VIS-Blockern auszustatten. Dann solltest du versuchen, so wenig wie möglich Detail-Work zu benutzen. Ausserdem solltest du hier auf alle Brushes die Caulk-Textur legen, die im Level nicht sichtbar sind. Ausserdem solltest du so genau wie möglich bauen - Brushüberschneidungen solltest du vermeiden. Ebenso den Einsatz von FPS-fressenden Shadern, z.B. Spiegel, die ein grosses Areal einsehen und so spiegeln, solltest du nicht verwenden.

VII. Easygen:

Frage: Wieso darf ich nicht mit CSG Substruct arbeiten?

Antwort: CSG Substruct löscht keinen Brush, sondern schneidet diesen nur. Nach dem Schneiden bleiben sehr oft noch Stücke übrig.

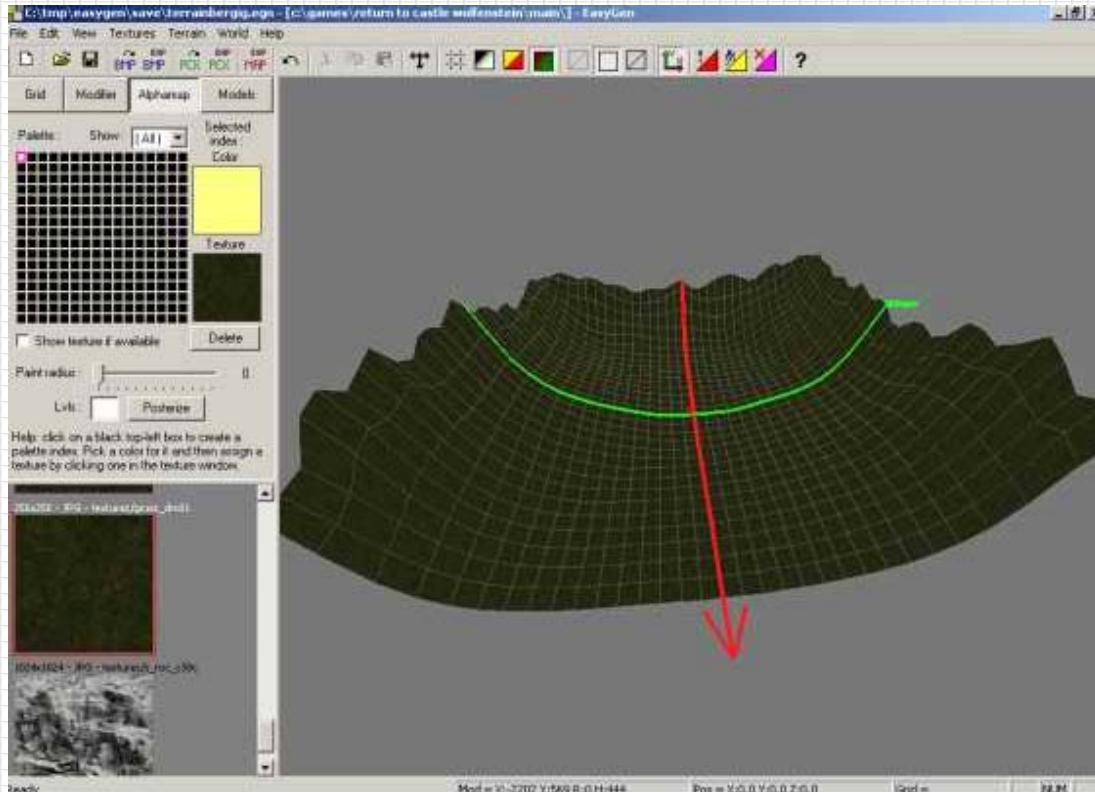
Diese Stücke gehören aber immernoch zum Terrain, d.h. die Alphamap mit der Texturzuweisung greift auch auf diese Stücke zu und belegt sie mit der entsprechenden Textur. So entstehen sehr hässliche Fehler, auf die man wirklich verzichten sollte.

Frage: Welche von den ganzen Easygen-Dateien brauche ich denn für meine Pk3-Datei?

Antwort: Zu diesem Thema habe ich ein eigenes Thema verfasst, dieses findest du [hier](#).

Frage: Wieso kann ich keine Täler erstellen?

Antwort: Wie du im unteren Bild sehen kannst, entsteht in Easygen immer eine Art Wanne, auf der das ganze Terrain aufbaut. Es gibt keinen niedrigeren Punkt als der unterste Punkt in dieser Wanne. Um also ein Tal zu erstellen, müsstest du zunächst das gesamte Terrain so erhöhen, dass du wieder ein Tal erstellen kannst.



VIII. Sonstiges:

Frage: In den Tutorials stehen manchmal Entities beschrieben, die es bei mir garnicht gibt?

Antwort: Das liegt daran, dass die meisten Entities hier nur in RtCW existieren. Existiert ein solches Entity bei dir nicht, mappst du warscheinlich für ein anderes Spiel. Dann solltest du einfach auf das entsprechende Thema verzichten.

Frage: Wieso kann ich durch den Himmel andere Räume sehen?

Antwort: Durch animierten Himmel kann man in höher gelegene Räume sehen. Das kannst du verhindern, indem zu dazwischen mehrere Brushes mit der Caulk-Textur setzt.

Frage: Was ist eine Pk3-Datei?

Antwort: Die Pk3-Datei ist eine Zip-Datei, die in .pk3 umbenannt wurde. In ihr befinden sich alle Informationen, die für die richtige Wiedergabe der Map nötig sind, also Shader, Texturen, das BSP-File, das ROQ-Video usw.

Frage: Was ist das Grid?

Antwort: Das Grid ist das Gitternetz, dass du im Radiant sehn kannst. Mit der Taste "0" kannst du es ausschalten und wenn du wieder auf die Taste "0" drückst, blendest du es wieder ein. Mit den Tasten 1 - kannst du den "Maßstab" ändern. Je kleiner die Zahl, desto feiner der Maßstab. 1 ist das kleinste Grid.

Frage: Meine func_groups sind nach der Installation des GtkRadiant 1.2.11 nicht mehr da?

Antwort: Die Shortcuts für func_groups hat sich geändert. Func_groups wurden früher mit "SHIFT" und der linken Maustaste angeklickt. Jetzt wird mit "SHIFT" und der linken Maustaste lediglich ein Brush der func_group selektiert. Um alle Teile zu selektieren, muss man "SHIFT" + "STRG" + linke Maustaste drücken. Wenn du noch einige andere Änderungen bemerkt hast, und du nicht genau weisst, was das alles soll, kannst du dir das [hier](#) durchlesen.

Frage: Was bringen eigentlich diese PlugIns im Radiant?

Antwort: Es gibt viele PlugIns im Radiant. Sie sind zwar unscheinbar, aber ihnen steckt echt eine Menge an Arbeitserleichterung:

- Curry - Plugin: Damit kann man in Echtzeit einen Shader betrachten
 - GtkGenSurf: Damit lassen sich Terrains erstellen
 - Pk3Man: Hilft beim Aufbau der Pk3-Datei
 - bobtoolz: Cooles Tool. Die wichtigsten Funktionen sind wohl das "Brush Cleanup", "Build Doors" und "Build Stairs". Aber auch die anderen Funktionen sind klasse und wirklich einige Tests wert
-

Frage: Warum muss ich ständig "sv_pure 0" eingeben?

Antwort: Durch den Befehl "sv_pure 0" durchsucht das Spiel auch die Ordner, die im "Main"-Ordner liegen, d.h. das Spiel lässt eigene/modifizierte Texturen/Levels usw. zu. Durch die Eingabe von "sv_pure 1" wird diese Funktion abgeschaltet und das Spiel liest nur die Dateien, die sich in den Pk3-Dateien befinden.

Frage: Was bringt eine Pk3-Datei?

Antwort: Da in der Pk3-Datei viele andere Dateien enthalten sind, spart man sich eine Menge Durcheinander und wahrt so den Überblick. Aber auch Cheatern wird so der Zugriff auf die Map verweigert. Alles, was in den Pk3-Dateien enthalten ist, wird getestet, in wie fern es noch "pure" ist. Fehlt z.B. eine oder mehrere Dateien in der Pk3-Datei, so wird man unpure und kann nicht mitspielen.

Frage: Welchen Zweck hat die shaderlist.txt?

Antwort: Die shaderlist.txt befindet sich im "Scripts"-Ordner. In der shaderlist.txt stehen alle Shader ohne Endung. Diese Shader werden nun beim Start des Radianten geladen. Erst jetzt sind die Effekte sichtbar, die durch die Shader gesteuert werden.

Frage: Sagmal, hast du eigentlich zuviel Zeit, Haradirki?

Antwort: Nein, aber Mapping ist eine tolle Sache, und es macht Spass, wenn andere Leute mitmachen.

[zurück zur Hauptseite](#)

183759



Links:

Hier findest du meine Links und noch einige weitere zum Thema mappen:

haradirki.de - hier ist das "Mutterschiff" meiner Tutorials. Natürlich kannst du hier vorbeikommen um dir die neusten Tutors zu ziehen.

<http://www.shaderlab.com/q3map2/> - hier findet ihr immer die neuste Version von der Q3Map2, also dem neusten Compiler von Ydnar. Absolut empfehlenswert und einen Test wert!

<http://www.shaderlab.com/q3map2/manual/> - Hier findet ihr das Handbuch zur Q3Map2. Viele Fragen werden hier erörtert, jedoch ist dafür ein gewisser englischer Wortschatz voraussetzung.

http://qeradiant.com/manual/Q3AShader_Manual/ - die wohl beste Anleitung, wenn es um Shader geht. Hier findet ihr wirklich alle Parameter, die es für Shader gibt. Wenn es hier etwas nicht gibt, findet ihr es warscheinlich nirgends.

<http://www.qeradiant.com> - Hier findest du die jeweils neuste Version des Radianten.

http://digilander.libero.it/ilbanca/files_easygen.htm - hier findet ihr jeweils die neuste Version von Easygen, dem Terrain-Programm. Zwar etwas gewöhnungsbedürfig, aber ein mächtiges Tool.

[zurück zur Hauptseite](#)





Lexikon:

Bobtoolz:

Die Bobtoolz erleichtern das Mappen deutlich. Egal, ob man Wendeltreppen oder Türen bauen will, oder einfach das Level "caulken" will.

Brush:

Jede Map setzt sich aus Brushs zusammen - sie sind die Grundlage für jede Map, da ja auf die Brushs die Texturen gelegt werden können.

Common-Texturen:

Sie bilden die wichtigsten Texturen im Spiel, z.B. gehören dazu die Texturen: "common/caulk", "common/trigger", "common/origin", "common/hint", "common/clip".

Compilieren:

Wenn du im Radiant eine Map erstellst, wird diese im *.map-Format abgespeichert. Dies ist ein Format, das der Radiant lesen kann - Q3 und RtCW hingegen nicht. Daher musst du deine *.map-Datei erst in *.bsp umwandeln, d.h. compilieren - um die Map für Q3 und RtCW lesbar zu machen. Je nach Compilierungsart wird die volle Map berechnet, oder aber nur die Map ohne bestimmte Dinge, z.B. Flüssigkeiten oder Curved Surfaces.

Curved Surfaces:

Q3 wurde dadurch bekannt, dass hier erstmals runde Oberflächen im Spiel zu sehen waren. Diese runden Oberflächen nennt man Curved Surfaces.

Entities:

Entities bilden eine besondere Gruppe unter den Brushs. Im Radiant sind sie in den Koordinatenfenstern über die rechte Maustaste anwählbar. Zu dem Entities gehören die Waffen, Player-Startpunkte usw. In manchen Entities werden Eigenschaften gespeichert, die später in der Map zwar nicht sichtbar sind, aber wichtige Effekte auslösen.

FPS:

Bilder pro Sekunde. Das Auge kann Bilder unter 25 Bilder pro sekunde als Bilder identifizieren, alles darüber wird als "fließend" erkannt, d.h. es ruckelt nichtmehr. Du musst in deiner Map darauf achten, dass die FPS nicht zu sehr in den Keller gehen, da sonst das Spielen unmöglich ist oder der Spielspass zerstört ist. Der Radiant kommt mit vielen Hilfen zur FPS-Steigerung, z.B. die Caulk-Textur, die Hint-Textur, aber auch Areaportale können bei guter Positionierung die FPS bedeutend steigern.

Die FPS kannst du dir in Q3/RtCW über folgenden Befehl (ohne die Anführungszeichen) anzeigen lassen:

`"/cg_drawfps 1"` = die FPS sind eingeschalten, sie erscheinen oben rechts
`"/cg_drawfps 0"` = die FPS sind ausgeschalten, d.h. sie werden nichtmehr angezeigt

GtkRadiant:

Dies ist die Radiant-Version, die etwas auf das Linux-Outfit getrimmt wurde.

Leak:

Das ist ein Loch, dass heisst, deine Map ist nicht komplett abgeschlossen - und es besteht die Möglichkeit, aus der Map zu fallen. Du solltest Leaks vermeiden, da es Probleme oder sogar Abstürze beim Compilieren oder der Botberechnung geben kann.

Map:

Für die einen Map, für die anderen Level, für die anderen Welt. Egal - Maps sind die Arenen, in denen die Spieler gegeneinander antreten.

Mapper:

Das sind die Menschen, die Maps entwerfen/programmieren/erstellen.

Models:

Das sind Gegenstände oder Statuen, die mit einem externen Programm erstellt wurden. Im Radiant werden sie über das Entity "misc_model" eingebunden.

PK3:

Das ist ein Zip-ähnliches Format, indem sich allerlei Informationen zu den Maps befinden, z.B. das Startbild, zusätzliche Texturen, Script-Dateien, Shader, und natürlich die *.bsp-Datei

Plugin:

Ist eine Art "Mod" für den Radiant. Hier kannst du zusätzliche Tools wie die Bobtoolz finden.

Q3Radiant:

Ist die Windows-ähnliche Variante des Editors

Script:

Ein Script ist nichts anderes als eine Art "Vorgabe", nach denen z.B. die Gegner vorgehen. Sie bestimmen auch die Intelligenz der Gegenspieler in RtcW. Ausserdem gibt es auch im Multiplayer teilweise ein recht kompliziertes Scripting.

Shader:

Texturen, die einen Shader haben, sind mit einem weißen Rand gekennzeichnet. Shader sind Textdateien, die die besonderen Eigenschaften der Texturen beinhalten.

Textur:

Eine Textur ist eine "Tapete", die auf die Brushes gelegt wird.

Unit:

Ist der Maßstab im Radiant. Ein Kästchen beträgt 8 Units. Das kannst du nachrechnen, indem du dir mal die Skalierung des Radianten ansiehst. Das Koordinatenkreuz beginnt bei 0, der nächste breite Strich liegt bei 64. Und 64 durch 8 (Anzahl der Kästchen von 0 bis 64) ergibt 8. Die Units ergeben zusammen den Grid, also das Koordinatensystem. Als Standartgrösse ist "4" eingetragen. Mit den Zahlen unter "4" kannst du den Grid feiner einstellen, d.h. feinere Brushes erstellen. Mit den Zahlen über "4" wird das Grid kleiner, du kannst gröbere Brushes erstellen.

Vertex-Punkt:

Vertex-Punkte sind Punkte auf einem Brush, mit dem du diesen verformen kannst (mit dem Vertex-Modus). Alternativ kannst du auch den Brush mit dem Edge-Modus verformen.

Void:

Ist eine Map nicht abgeschlossen, gibt es eine Verbindung zum "Void", also ins Nichts. Gibt es in deiner Map ein Loch, so bekommst du beim Compilieren die Fehlermeldung: ***leak***

[zurück zur Hauptseite](#)





Impressum:

Mein Dank geht an:

PhatJay

Dangerzone

Beowulf

SweedMarmelade

Hbee99

und www.planetenemyterritory.de

sowie an den Moderator meines Forums:

Micro-Gerbil

sowie allen, die mitgeholfen haben, diese Seite bekannt zu machen und natürlich meinen Beta-Testern, die x-mal jedes Thema durchackern mussten, dass auch jeder Fehler ausgemerzt wurde - sowie allen, die mich in meiner Arbeit unterstützt haben.

[zurück zur Hauptseite](#)

183759



Tastenbelegungen:

Im Radiant bewegen:

a	Kamera-Winkel hoch
z	Kamera-Winkel runter
d	Kamera hoch
c	Kamera runter
,	Kamera nach links (Blickwinkel bleibt gleich)
.	Kamera nach rechts (Blickwinkel bleibt gleich)
ENTF	Arbeitsfläche vergrößern
EINFG	Arbeitsfläche verkleinern
Seite runter	nach unten auf die nächste mögliche Ebene springen
Seite hoch	nach oben auf die nächste mögliche Ebene springen
ENDE	die Kamera zentrieren (wenn der Winkel verstellt wurde)
STRG + z	den letzten Schritt rückgängig machen (Undo)
STRG + SHIFT + c	Camera fixieren ein/ausschalten
STRG + ß	CubicClipping Sichtfeld vergrößern
STRG + ´	CubicClipping Sichtfeld verkleinern

Grid (Koordinatensystem-Einstellungen):

1	Grid auf 1 Unit setzen
2	Grid auf 2 Units setzen
3	Grid auf 4 Units setzen
4	Grid auf 8 Units setzen
5	Grid auf 16 Units setzen
6	Grid auf 32 Units setzen
7	Grid auf 64 Units setzen
ß	Gitternetz feiner darstellen
´	Gitternetz grober darstellen
STRG + TAB	Zyklisches durchschalten der 2D Sichten
0	Grid ein/ausschalten

mit Brushes umgehen:

STRG +3	Dreieckigen Brush erzeugen
STRG +4	Viereckigen Brush erzeugen
STRG +5	Fünfeckigen Brush erzeugen
STRG +6	Sechseckigen Brush erzeugen
STRG +7	Siebeneckigen Brush erzeugen

STRG +8	Achteckigen Brush erzeugen
STRG +9	Neuneckigen Brush erzeugen
ALT + Pfeil hoch	verschiebt selektierten Brush um jeweils eine Unit nach oben
ALT + Pfeil runter	verschiebt selektierten Brush um jeweils eine Unit nach unten
ALT + Pfeil links	verschiebt selektierten Brush um jeweils eine Unit nach links
ALT + Pfeil rechts	verschiebt selektierten Brush um jeweils eine Unit nach rechts
Minus	selektierten Brush/Entity nach unten verschieben
Plus	selektierten Brush/Entity nach oben verschieben
SHIFT + Linke Maustaste	einen kompletten Brush selektieren
STRG + K	zwei Entities miteinander verknüpfen
STRG + c	einen Brush in die Zwischenablage kopieren
STRG + v	einen Brush aus der Zwischenablage einfügen
ALT + Pfeil hoch	verschiebt selektierten Bruhs um jeweils eine Unit nach oben
ALT + Pfeil runter	verschiebt selektierten Bruhs um jeweils eine Unit nach unten
ALT + Pfeil links	verschiebt selektierten Bruhs um jeweils eine Unit nach links
ALT + Pfeil rechts	verschiebt selektierten Bruhs um jeweils eine Unit nach rechts
BACKSPACE	Selektierten Brush löschen
SPACE	einen Brush verdoppeln (klonen) und versetzt anzeigen
h	einen selektierten Brush verstecken
SHIFT + h	einen selektierten Brush wieder anzeigen
STRG + m	Detail-Brushs erstellen
SHIFT + STRG + s	Structural-Brushs erstellen
ESC	alles deselektieren
STRG + g	selektierten Brush auf Gitternetz einrasten lassen
SHIFT + d	Erst einen Brush anwählen, dann SHIFT + d drücken - in der Console erscheinen einige Informationen über den selektierten Brush
r	Freies Rotieren über die Maus ein/ausschalten
STRG + SHIFT + k	nächstes Leak anzeigen
STRG + SHIFT + l	vorhergehendes Leak anzeigen
Rechte Maustaste	in der 2D Sicht das Fenster verschieben

Radiant steuern:

STRG + x	Gtk- bzw. Q3Radiant verlassen
STRG + o	Map öffnen
STRG + s	Map speichern
SHIFT + b	FitBrush, die Textur auf einen Brush genau ausrichten
p	Preferences (Einstellungen-Fenster)
o	Console ein/ausschalten
STRG + d	Detail-Brushs ein/ausblenden
STRG + p	Curved Surfaces verbergen ein/ausschalten
m	Information über die Map

Entities:

SHIFT + Linke Maustaste	ein komplettes Entity selektieren
STRG + K	zwei Entities miteinander verknüpfen
STRG + c	ein Entity in die Zwischenablage kopieren
STRG + v	ein Entity aus der Zwischenablage einfügen
l	Entity Liste

BACKSPACE	Selektiertes Entity löschen
k	für farbige Lichter: es erscheint eine Farbauswahlbox, Farbe wird für das Licht übernommen.
SPACE	ein Entity verdoppeln (klonen) und versetzt anzeigen
h	einen selektiertes Entity verstecken
SHIFT + h	einen selektiertes Entity wieder anzeigen
ESC	alles deselektieren
STRG + g	selektiertes Entity auf Gitternetz einrasten lassen
n	die Schlüssel-Werte (Keys und Values) eines Entities anzeigen
SHIFT + a	einen Entity-Typ auswählen, SHIFT+a drücken, alle gleichen Typen werden angezeigt.

Texturen:

SHIFT + Seite runter	Textur im Gegenuhrzeigersinn drehen
SHIFT + Seite hoch	Textur im Uhrzeigersinn drehen
STRG + Pfeil runter	die Textur verbreitern/verschmälern nach oben/unten
STRG + Pfeil hoch	die Textur verbreitern/verschmälern nach oben/unten
STRG + Pfeil links	die Textur verbreitern/verschmälern nach links/rechts
STRG + Pfeil rechts	die Textur verbreitern/verschmälern nach links/rechts
SHIFT + Pfeil runter	die Textur nach unten verschieben
SHIFT + Pfeil hoch	die Textur nach oben verschieben
SHIFT + Pfeil links	die Textur nach links verschieben
SHIFT + Pfeil rechts	die Textur nach rechts verschieben
SHIFT + t	Textur beim Verschieben sperren ein/ausschalten
SHIFT + r	Textur beim Rotieren sperren ein/ausschalten
STRG + SHIFT + n	die Textur von einer Curved Surface zyklisch durchschalten (Ausrichten)
STRG + SHIFT + i	die Textur einer Curved Surface in X Richtung invertieren
SHIFT + i	die Textur eines Curved Surface in Y Richtung invertieren
STRG + n	richtet die Textur bei Curved Surfaces automatisch aus
SHIFT + s	Eingabefeld, um Curved Surfaces Texturen zu bearbeiten
SHIFT + STRG + Linke Maustaste	eine einzelne Fläche eines Brushs selektieren (Textur)
SHIFT + STRG + ALT + Linke Maustaste	mehrere einzelne Flächen eines Brushs selektieren (Textur)
t	das Texturen-Fenster ein/ausschalten
STRG + a	alle verwendeten Texturen anzeigen
u	nur die benutzten Texturen anzeigen
Mittlere Maustaste	für einen selektierten Brush die Textur von einen anderen Brush übernehmen

Tools:

s	Surface Inspector aufrufen
e	Kanten auswählen (Edge-Modus)
v	Eckpunkte auswählen (Vertex-Modus)
x	Zerschneide-Tool ein/ausschalten (einen Brush zerschneiden)
STRG + t	Thicken-Tool, doppelseitige Curved Surfaces erstellen
SHIFT + c	Caps (Abschlüsse) hinzufügen (Cap-Menü)
SHIFT + x	Fadenkreuz in den 3 Sichtfenstern ein/ausschalten

STRG + \	CubicClipping ein/ausschalten
STRG + ENTER	von einem geteilten Brush die zwei Seiten vertauschen
SHIFT + ENTER	die zerschnittene Hälfte auswählen (vorher mit X einen Brush zerschneiden)
ENTER	Geteilten Brush übernehmen (X-Taste beachten)

Vertex-Punkte:

STRG + SHIFT + e	Vertex-Punkte gleichmäßig ausrichten
STRG + e	Vertex-Punkte gleichmäßig ausrichten
y	Vertex-Punkte eines Patches dauerhaft anzeigen
STRG + I	Vertex-Punkte eines Patches verbergen

[zurück zur Hauptseite](#)

183759



Fehlermeldungen - Error Messages :

Leider gibt es beim Radiant häufig eine Error Message, mit der man nicht so viel anfangen kann. Deshalb findest du hier jede Menge an Fehlermeldungen für den Radianten, sowie Lösungshinweise, wie du deinen Fehler wieder los wirst:

Fehler:	Lösung:
Die Map lädt garnicht:	<p>Hier gibt es leider viele Möglichkeiten, wieso deine Map sich im Spiel nicht laden lässt. Hier aber die 4 Hauptgründe:</p> <ul style="list-style-type: none">• Du musst "sv_pure 0" in die Console eingeben, bevor du die Map mit dem Befehl "map XXX" oder "devmap XXX" startest• Es liegt ein Fehler in einem eigens erstellten Shader vor.• Du hast vergessen, einen Player-Startpunkt zu setzen.• Die Map ist noch nicht kompiliert.
ERROR: MAX_SURFACE_VERTS	Du hast in deiner Map zuviele Curves und Patches. Hier bleibt dir nichts anderes übrig, als einige davon zu löschen.
ERROR MAX_MAP_LIGHTGRID	Du hast zuviele Lichtquellen in deiner Map. Du deselektierst alles in deiner Map, und öffnest das Entity-Fenster. Hier scrollst du ganz runter bis zu "Worldspawn". Hier gibst du als Key "Gridsize" und als Value "128 128 128" ein
WARNING : Couldn't find image for shader noshader	Dies ist ein Überbleibsel der alten Engine, und kann ignoriert werden.
ERROR: MAX_MAP_BRUSHSIDES	du hast einen zu komplexen Brush. Entweder du löschst ihn, oder du packst ihn mit der Clip-Textur ein.
WARNING: Node without a volume / node has 0 tiny portals	Du hast einen kaputten Brush, du musst ihn löschen. Wenn du nicht weißt, um welchen es sich handelt, musst du zu einer Sicherungskopie greifen.
ERROR: RE_LoadWorldMap: maps/test.bsp not found	Du mußt die Map mit dem Befehl "sv_pure 0" starten, da deine Map nicht in einer PK3-Datei vorliegt, nimmst du das Spiel an, du hast diese test.bsp nicht installiert.
FloatPlane:bad normal	Dies wird häufig durch eine Vertex-Manipulation erzeugt. Du hast einen Vertex-Punkt des Brushes verschoben und der Brush verschwindet plötzlich. Drücke sofort "UNDO" um diesen Schritt rückgängig zu machen.
MatchToken("(") failed at line xxx	Du hast einen Fehler in einem eigenem Shader. Schau einfach einmal nach, ob du alle Klammern richtig gesetzt hast, oder die letzte Klammer vergessen hast.
trigger_push start solid\n	Dein Trigger-Brush befindet sich ganz oder teilweise in einem soliden Brush. Entweder du machst den trigger_push grösser, oder du verschiebst ihn etwas.

target_teleporter without target	Dein Teleporter hat noch kein Ziel, zu dem er teleportieren soll.
***** leaked *****	Deine Map hat eine Verbindung zum Void, dieses Loch musst du verschließen. Es gibt viele Möglichkeiten, wie ein Leak zustande kommt. Dazu habe ich dir hier ein eigenes Thema geschrieben, dass sich näher mit Leaks beschäftigt.
ERROR: MAX_FACETS during vlight (maxfacets >= 65XXX)	Du hast zuviele spheres und/oder cones in deiner Map. Das Maximum liegt bei 63335
LoadTGA: Only type 2 (RGB), 3 (gray), and 10 (RGB) TGA images supported	Du versuchst hier, ein Bild in einem Shader zu benutzen. Alle Shader-Script-Bilder sollten im TGA-Format vorliegen. Das JPG-Format geht auch, allerdings musst du hier darauf achten, das Bild nicht im "progressive"-Mode abzuspeichern. Die Engine kann mit diesen Bilder nichts anfangen. Du musst deine JPGs also ohne "progressive" speichern.
ERROR: SV_SetBrushModel: NULL	Dieser Fehler tritt auf, wenn du z.B. ein func_rotating in deiner Map hast, aber dazugehörig nur ein "common/origin"-Brush.
*****error*****	Deine Map compiliert überhaupt nicht - das kann daran liegen, dass du ein Leerzeichen im Namen deiner Map hast, oder einen Umlaut oder ein Sonderzeichen. Also musst du dieses Zeichen löschen.
Error opening[map filename]: No such file or directory	Der Radiant findet deine Map nicht, dieser Fehler kann auch durch ein Leak oder einen fehlenden Meta-Shader ausgelöst werden.
illegal operation	Dies ist wirklich ein übler Fehler, der Radiant kann ihn selbst leider nicht finden.
WARNING! HashVec: point [float] [float] [float] outside valid range	Wow.. hier hast du die Größenverhältnisse des Radianten geschlagen. Du musst beim mappen darauf achten, im folgendem Grid zu bleiben: 8000 x 8000 x 8000 Units Bei Q3/RtCW sind es 64.000 x 64.000 x 64.000 Units
trigger_multiple not in any area	Dein trigger_multiple liegt im Void.
too many entities in BSP file	Du hast zuviele Entities in deiner Map
areaportals only allowed in world	Du hast einen Fehler bei einem deiner Area-Portale. Vielleicht hat es eine Verbindung zum Void. Daher schaust du am besten mal alle Area-Portale durch, und kontrollierst diese.
Node with unbounded volume	Du hast einen Brush, der keine festgelegten Punkte mehr besitzt, d.h. er ist kaputt. Du musst diesen löschen und neu machen. Dann einfach erneut compilieren.
SetQdirFromPath: no 'quake3' in C:\MAPS\q3test.map SetQdirFromPath: no 'wolfenstein' in C:\RtCW\Main\maps\test.map	Du musst dein Verzeichniss so umbenennen, dass das Verzeichniss den Namen 'Quake3', 'Wolf' oder eben das entsprechende Spiel enthält.

Kann auftreten, wenn du einen Zylinder rotieren lässt und die Vertex-Punkte nichtmehr auf dem

SubDivideError: can't split the polygons	<p>Grid liegen, oder wenn du überhaupt die Vertex-Funktion benutzt. Häufig kann es auch passieren, dass 2 Vertex-Punkte auf dem gleichen Platz sind, oder eben mehr als ein Vertex-Punkt nicht auf dem Grid liegen.</p> <p>Du musst also alle Vertex-Verschiebungen rückgängig machen. Wenn das nicht geht, musst du den Brush neu machen.</p>
HashVec: point outside valid range	Dieser Fehler tritt auf, wenn du die Grenzen des Grids überschreitest. Entweder, du mappst etwas kleiner, oder du schiebst deine Map mehr in die Mitte.
stylenum == MAX_SWITCHED_LIGHTS	Dieser Fehler tritt auf, wenn du auf ein Licht-Entity ziehst. Seit Q3 gibt es keine Lichtstyles mehr.
ERROR: MAX_PATCH_PLANES	Eine sichtbare Brush-Oberfläche wurde zu oft gesplittet. Zusätzliche Vertex-Punkte werden von der q3map erstellt, wenn sie von anderen sichtbaren Punkten gesplittet werden (z.B. Vertices, Corners). Hast du also zuviele sichtbare Oberflächen-Vertex-Punkte, die die Ecken von anderen Oberflächen berühren, so wird diese Oberfläche gesplittet und du hast zuviele Vertex-Punkte.
WARNING: too many light styles on a face	<p>Jedes "Face" kann nur von 4 verschiedenen Lichtquellen angestrahlt werden - jede weitere Lichtquelle verursacht diesen Fehler.</p> <p>Es ist in Ordnung diesen Fehler zu ignorieren, wenn du keine Licht-Probleme in deiner Map hast, falls doch, solltest du deine Lichtquellen reduzieren/verringern.</p>
ERROR: Leaf with too many portals	Der Detailgrad in einem kleinen Gebiet übersteigt die Möglichkeiten vom VIS-Prozess (es ist zu kompliziert beim compilieren). Finde diesen Teil in deiner Map und vereinfache es.
Chose a 0 valued axis	Dieser Fehler taucht auf, wenn du versuchst, eine Cone mit einem Cap zu versehen. Das Problem dabei ist, dass die Spitze der Cone eine 0*0 Cap erstellt wird. Also musst du den Cap an der Spitze der Cone entfernen.
WindingFromDrawSurf failed: MAX_POINTS_ON_WINDING exceeded	Passiert oft, wenn du eine Menge an Brushes hast, die an nur einen Brush grenzen. Das Maximum liegt hier bei 64 Brushes. Dieser Fehler tritt z.B. dann auf, wenn du eine Leiter mit vielen Sprossen hast. Hast du 64 oder mehr Sprossen eingebaut, berühren 64 Brushes einen einzigen Brush. Nun taucht dieser Fehler auf.
AllocWinding failed: MAX_POINTS_ON_WINDING exceeded	Du musst einige davon löschen, dass du unter die 64 Brushes kommst.
Line 178 is incomplete	Dein Shader hat einen Fehler in der Linie 178. Denkbar wäre hier ein "Surfaceparm" ohne Parameter
WARNING potential hash mismatch at XXX	Hier liegt ein Fehler in der BSP-Datei vor. Du musst im Radiant nach den Koordinaten suchen, die ich hier mit den XXX dargestellt habe. Nach Brushes, Entities usw. suchst du über die Funktion "Search Brush oder Entity". Hast du den Fehler gefunden, musst du ihn löschen. Startet deine Map nichtmehr im Radiant, musst du die BSP-Datei mit einem Text-Editor öffnen und die Koordinaten über "Bearbeiten" -> "Suchen" ausfindig machen und löschen.
ERROR:No file to process Run with-?	Hier solltest du sicherstellen, dass du deine *.map Datei auch im Maps-Ordner abspeicherst und sich keine Leerzeichen im Namen deiner Map befinden - so z.B. "name der map.bsp" solltest du in "name_der_map.bsp" umbenennen.

183759



Leaks:

verwendete Beispielmap: "tutor42.map"

Ein Leak in der Map - das ist wohl der Fehler, der Anfängern am häufigsten passiert. Deshalb will ich dir hier einige Beispiele zeigen, die einen *****leaked***** Error hervorrufen und auch, wie man diese verhindert.

Leaks kommen durch folgende Konstruktionen zustande:

- Du hast einen Spalt zwischen 2 Brushes.

Dies ist wohl der am häufigsten auftretende Fehler und ebenso leicht ist er zu verhindern. Wenn du mappst, solltest du keinen kleineren Grid nehmen, als 8 Units pro Kästchen (Taste "4"). Man braucht die kleineren Einstellungen zum normalen Mappen nicht - lediglich zum "Fein-Tuning". Wenn du mit einem kleineren Grid arbeitest, kann es häufig passieren, dass du kleine Zwischenräume übersiehst.

- Lava, Wasser, Schleim usw. und andere "nicht-solide" Texturen müssen in "Behältern" sein.

Wenn du z.B. einen See mappen willst, solltest du zuerst den Seeboden und die Wände machen, und erst zuletzt das Wasser einsetzen, da Flüssigkeiten die Map nicht "abdichten". Grenzen sie also in den Void, hast du ein Leak.

- Du benutzt die falschen Texturen um deine Map "einzuhüllen"

Clip, Full_Clip, ClipMissile und die anderen unsichtbaren Texturen können deine Map nicht vom Void abgrenzen. Zusätzlich darfst du nicht vergessen, die äussersten Brushes in deiner Map dürfen keine detail-Brushes sein.

- Türen und Plattformen haben keine Brushes, die sie einhüllen

Wenn deine rote Linie irgendwo nahe an eine Tür oder eine Plattform zeigt, ist dies der Fehler. Bevor du Aufzüge, Plattformen und Türen baust, solltest du darauf achten, dass sie ein *****Leak***** Error hervorrufen, wenn sie in den Void grenzen. Also solltest du unbedingt darauf achten, zuerst den Fahrstuhl-Schacht und anschliessend den Fahrstuhl selbst mappen.

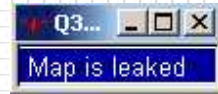
Einer der häufigsten Fehler ist der, dass du den Bodenbrush im Fahrstuhl nicht mappst, da er ja nie in der Map sichtbar ist. Damit hast du zwar recht, aber nun steht der Fahrstuhl in direkter Verbindung zum Void. Nun hast du wieder ein Leak.

- Entites befinden sich im Void.

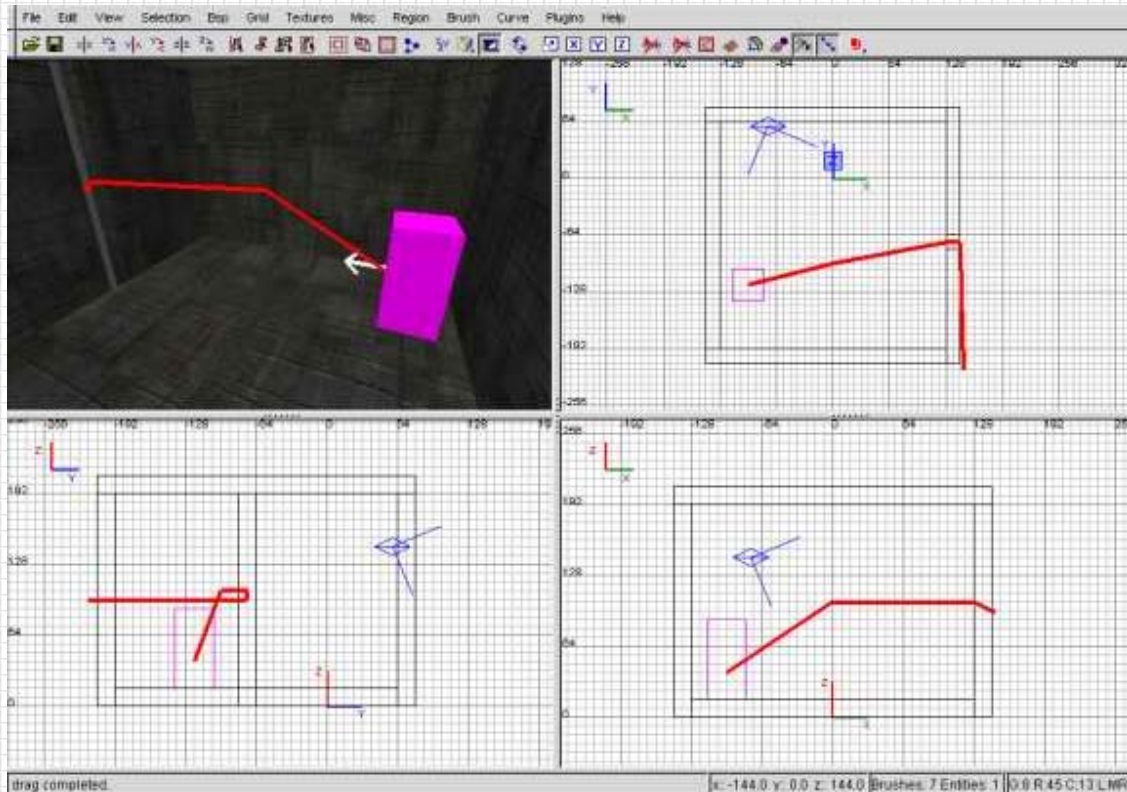
Liegen lights, target_speakers, misc_teleporters und andere Entities im Void, hast du auch ein Leak.

Aber die Entities können teilweise im Void liegen, z.B. eine func_door kann teilweise ausserhalb der Map liegen, ohne dass es ein Leak gibt. Nun solltest du aber darauf achten, dass es keine Darstellungsfehler gibt, wenn die func_door komplett in der Map zu sehen ist. Manchmal führt dies dazu, dass die Seitentextur komplett schwarz ist.

Nun gibt es im Radiant eine sehr nützliche Funktion, wie du Leaks auf die Schliche kommst, die du in der Map garnicht auf einen Blick sehen kannst. Dazu öffnest du am besten meine Map, die ich für dieses Thema erstellt habe. Nun compilierst du die Map einfach. Kurz darauf erscheint dieses kleine Fenster:



Klickst du nun auf die blaue Schrift "Map is leaked", so erscheint eine rote Linie in der Map:



So, wie du nun siehst, zeigt die Linie von einem Entity zum Leak in der Map. Ich habe hier ein sehr einfaches Leak erstellt und zwar durch den Spalt zwischen den Wand-Brushes.

ACHTUNG: Wenn du mit einer anderen Version des Radianten arbeitest, kann es passieren, dass das kleine Fenster nicht auftaucht. Nun musst du diese Linie manuell laden. Das kannst du tun, in dem du in der Menüleiste "File" anwählst und dort "Pointfile" anwählst. Nun erscheint die rote Linie.

Nun wollen wir natürlich die rote Linie und das Leak loswerden. Also markierst du einen der beiden Brushes und verlängerst ihn soweit, dass er an den anderen angrenzt. Speicherst du nun die Map ab, verschwindet die Linie, das Leak ist geschlossen.

Leider kommt es vor, dass die rote Linie in der Map so klein ist, dass du sieh nicht sehen kannst. Das kann daran liegen, dass die Linie z.B. an einem Light-Entity startet, das sich schon im Void befindet. Hier ist die Linie so klein, dass du sie nicht sehen kannst. Es kommt auch vor, dass die Linie vor einem Wand-Brush aufhört. Hier ist der Wand-Brush "detail", d.h. er dichtet die Map nicht ab.

So kann es vorkommen, dass du den Fehler nicht sofort findest. Das ist aber kein Grund um aufzugeben. Zwar ist es oft verzwickelt, aber meist ist der Fehler leicht zu finden.

[zurück zur Hauptseite](#)

183759



Performance:

verwendete Beispielmap: "tutor37.map"

Ergebnismap: "tutor38.map"

So, nun bist du schon dabei, deine erste Map zu erstellen. Deshalb will ich dir an dieser Stelle erklären, wie man die Performance überprüft. Die Performance heisst übersetzt "Leistung", und gibt an, wie flüssig deine Map auf einem System läuft. Nichts ist ärgerlicher, als wenn die eigene Map nicht auf dem eigenen Rechner läuft.

Als erstes wollen wir uns mal die `r_speeds` von der Map ansehen. `R_Speeds` sind mehrere Zahlen, die die Auslastung des Systems anzeigen und durchs Bild fliegen. Um diese `r_speeds` anzeigen zu lassen, musst du die Map natürlich erst einmal compilieren. Dann schaust du nochmal nach, ob die `*.bsp` Datei im Ordner "Main/Maps" befindet. Sonst kommt nämlich ein Error. Befindet sich diese Datei nicht im Maps-Ordner, hast du die Map noch nicht compiliert, oder du hast einen Fehler.

Befindet sich die Datei im Ordner, lädst du erstmal das Spiel, und drückst die Taste "^", damit öffnest du die Console. Hier tippst du die folgenden Befehle ein:

```
/sv_pure 0
/developer 1
/devmap X (das X steht für den Namen deiner Map)
```

Beim "/devmap X"-Befehl darfst du nicht vergessen, dass du die `*.bsp` Endung nicht eintippst, sonst lädt sich die Map ebenfalls nicht. Nun läd deine Map. Wenn die Map fertig geladen ist, drückst du nochmals die Taste "^", um wieder die Console zu öffnen. Jetzt tippst du ein:

```
/r_speeds 1
```

Nun erscheinen die oben erwähnten Zahlen, die die Auslastung des Systems anzeigen. So ungefähr sieht das dann aus:

```
99/664 SHADERS/SURFS 129 LEAFS 6468 VERTS 4830/5710 TRIS 1.36 MTEX 0.00 DC
99/664 SHADERS/SURFS 129 LEAFS 6468 VERTS 4830/5710 TRIS 1.36 MTEX 0.00 DC
99/664 SHADERS/SURFS 129 LEAFS 6468 VERTS 4830/5710 TRIS 1.36 MTEX 0.00 DC
```

Die markierten Zahlen geben an, wie viele Polygone oder Dreiecke die Engine rendert. Du solltest darauf achten, dass diese Zahl im schlimmsten Fall den Wert 12.000 bis 14.000 nicht überschreitet (z.b. vom obersten Bunker auf der Map `mp_beach` liegt diese Zahl bei 13.000. Am besten ist jedoch ein Durchschnitt von 8.000 Polygonen. Du solltest durch deine gesamte Map laufen um diese Polygonen-Anzahl zu überprüfen, da sich diese Anzahl mit jeder Blickwinkeländerung steigen/sinken kann.

WICHTIG: Da nachher neben der Map auch noch die Spieler/Bots berechnet werden, solltest du pro Spieler/Bot 800 - 1.000 Polygone hinzuzählen.

Nun gibst du in die Console den folgenden Befehl ein:

```
/r_speeds 0
```

Mit diesem Befehl schaltest du die `r_speeds`-Darstellung wieder aus.

r_speeds und VIS:

Nun tippst du noch

```
/r_showtris 1
```

ein. Nun siehst du, was die Engine von deiner Map berechnet. Im Idealfall sollte nur das berechnet werden, was der Spieler auch sieht. Meist wird jedoch mehr berechnet, als man tatsächlich sieht.

Sieht der Spieler beispielsweise durch ein Fenster nur einen kleinen Teil eines Raumes, zeichnet die Engine den ganzen Raum. Die Engine zeichnet nämlich zuerst die am weitesten entfernten Teile und überdeckt diese dann mit den näheren Teilen des Raums. Dies führt dazu, dass auch wenn man nur einen kleinen Teil sieht die Engine dennoch auch das dahinter liegende rendert.

Wie du sicher schon gesehen hast, wird eine Map in 3 Prozessen compiliert: BSP, VIS und RAD:

BSP	berechnet die Geometrie, also Wände, Böden, Säulen usw.
VIS	erstellt eine Tabelle die der Engine darüber aufschluss gibt, was von einem gegebenen Punkt aus gesehen werden kann
RAD	berechnet die Beleuchtung

Für die r_speeds ist VIS wichtig. Beim VIS gibt es mehrere Optionen, 2 davon sind fast-vis und full-vis. Für korrekte r_speeds-Angaben solltest du die Map mit full-vis compilieren lassen, da nur hier die VIS-Blocker funktionieren und damit die r_speeds eine genauere Aussagefähigkeit besitzen.

r_speeds und verwandte Befehle:

Nun hast du sicher bemerkt, dass wir die Map immer mit dem Befehl "/devmap [Mapname]" starten. Das liegt daran, dass r_speeds und andere nützliche Befehle leider Cheat-geschützt sind - das ist auch gut so, mit r_showtris z.b. werden ja auch die Spieler hinter den Wänden dargestellt. So ist z.b. "/noclip" ein nützlicher Befehl, um die Map aus jedem Blickwinkel betrachten zu können.

Hierzu habe ich mal einen Ausschnitt eines Posts von Paul Jaquays gefunden:

```
These are my development keys. You have to be in devmap mode to use them.  
The actual key binds are up to you.  
// Cheat/development Keys
```

```
bind F2 "toggle r_nocurves"  
bind F3 "toggle r_showtris"  
bind F4 "toggle r_lockpvs"  
bind F5 "toggle r_drawentities"  
bind F6 "toggle r_fastsky"  
bind F7 "toggle r_speeds"  
bind F8 "toggle r_clear"  
bind F9 "noclip"  
bind F10 "god"  
bind F11 "screenshot"  
bind F12 "give all"
```

r_lockpvs locks your point of view in place so you can wander your map and see what is seen and not seen from that point of view (with fastsky on the not-seen will be yellow)

r_showtris is showing you EXACTLY how the world is being broken into triangles. This is your diagnostic for looking for problems caused by brushes.

Use the `r_nocurves` command to remove the confusing curve data from your view.

Same with `r_drawentities`

Paul Jaquays
designer
id Software

Nun will ich dir mal ein paar Beispiele für Dinge geben, die stark von der Performance abhängig sind, so z.B. beeinflussen Shader mit mehreren Schichten, Terrains, Spiegel und Curves die Performance sehr stark.

- Curves:

Um eine Curve zu berechnen, werden natürlich viel komplexere Operationen durchgeführt, als bei der Berechnung eines gewöhnlichen Bruhes. Zum einen muss die Engine die Curve ja erst aus den paar gegebenen Punkten komplett berechnet werden. Zum anderen gibt es bei Curves das so genannte "Level of Detail". Je mehr freie Rechenleistung zur Verfügung steht, je komplexer wird die Curve dargestellt. So kann eine komplexe Curve die `r_speeds` in die Höhe schnellen lassen.

- Shaders:

Gewöhnliche Texturen werden von der Engine zwei Mal gerendert, einmal Texture und einmal Lightmap. Besitzt die Textur z.B. einen Überblendungs-Effekt, so muss diese Textur drei Mal gerendert werden. für jede weitere Stufe muss die Textur einmal mehr gerendert werden.

Mit am rechenintensivsten sind transparente Texturen, bei denen man ein Polygon durch ein anderes Polygon sehen kann. Ein gutes Beispiel hierfür ist Wasser. So kann aber auch Nebel die Performance stark beeinflussen.

- Terrain:

Besonders grosses Terrain ist sehr rechenintensiv. Gerade bei Terrain mit grossen Sichtwinkeln steigen die `r_speeds` ins Unermessliche. Gerade beim Terrain lässt sich durch frühe Planung des Terrains z.B. der Sichtwinkel einschränken oder durch das Einsetzen von Hint-Brushs die `r_speeds` im grünen Bereich bleiben. So ist nicht nur das Terrain an sich rechenintensiv, sondern auch, was sich auf dem Terrain befindet, so z.B. Häuser, aber auch Modelle wie Bäume, Fahrzeuge usw. können sich sehr negativ auf die Performance auswirken.

- Spiegel:

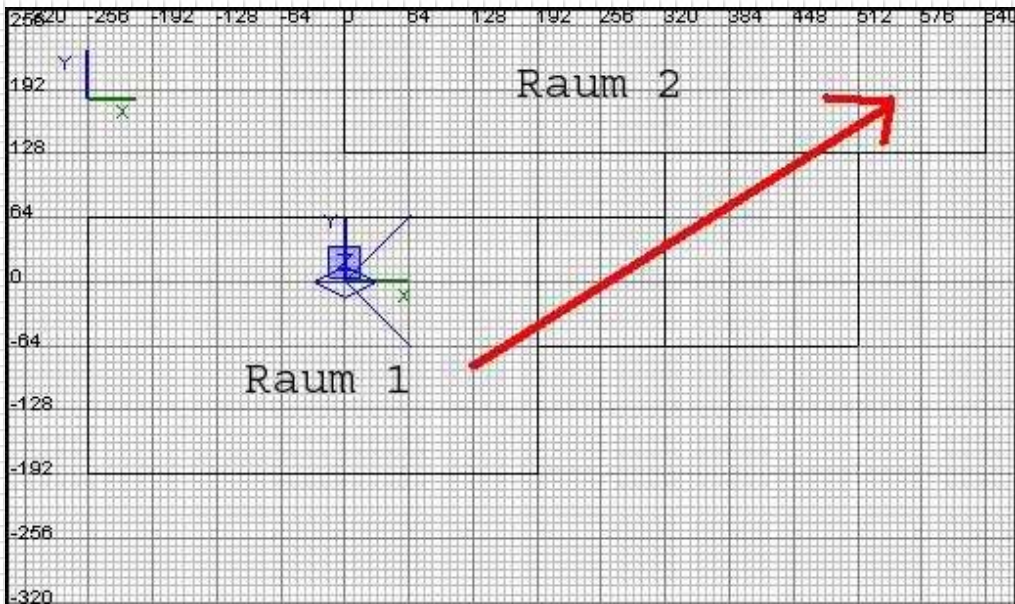
Spiegel können sehr rechenintensiv sein, wenn sie einen grossen Sichtwinkel haben. Also solltest du darauf achten, dass du sie nur in kleinen Räumen oder so platzierst, dass eine möglichst kleine Fläche gespiegelt wird. Schliesslich wird jeder sichtbare Brush für den Spiegel gespiegelt und so sind also die doppelte Menge an Brushs zu errechnen. Wichtig ist auch, dass man keine komplexen Strukturen, wie z.b. komplizierte Curves im Winkel des Spiegels stehen sollten - ebenso dürfen sich auch keine Spiegel gegenüberstehen, d.h. sich selbst spiegeln.

Nun will ich dir mal einige Beispiele zeigen, wie man die Performance etwas verbessern kann:

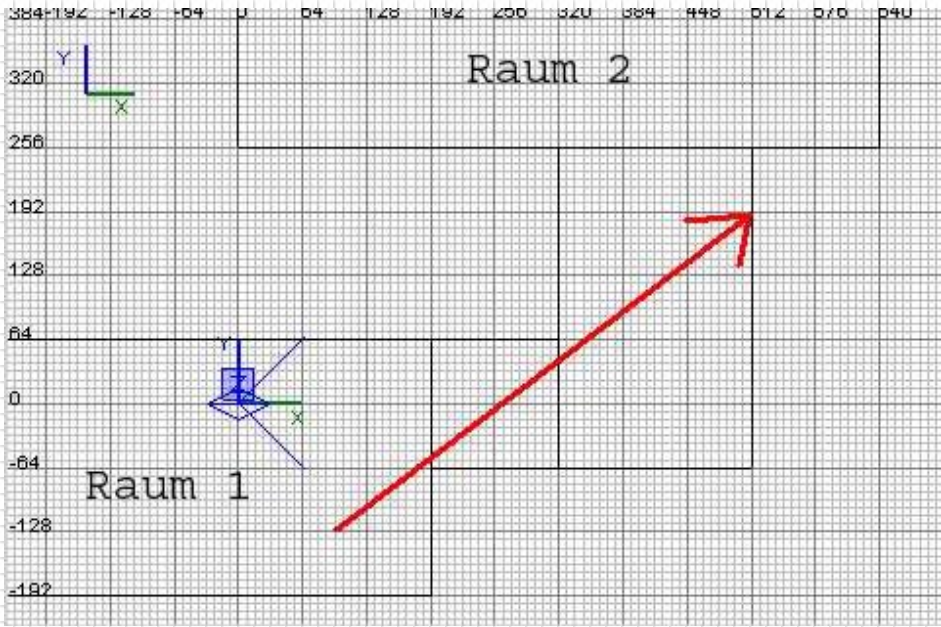
VIS-Blocker:

VIS-Blocker sind gaz normale Konstruktionen, die verhindern sollen, dass der Spieler zuviel auf einmal von der Map sieht und so die `r_speeds` in den roten Bereich treibt. So soll also verhindert werden, dass der Spieler im Raum 1 nicht in den Raum 2 sehen kann. Es gibt viele VIS-Blocker-Methoden, aber die folgenden 2 kommen sehr häufig vor:

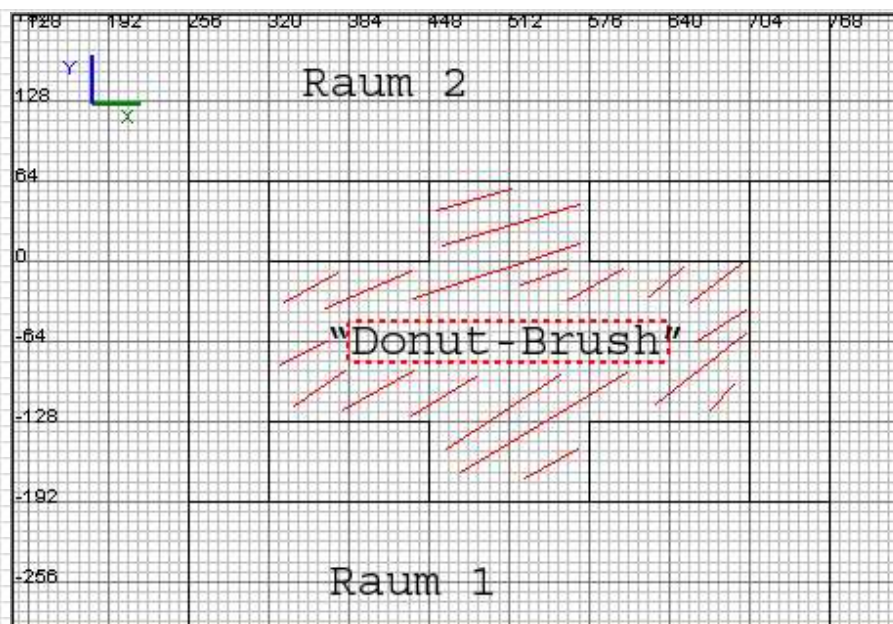
Ein gängiger Vis-Blocker ist diese Eck-Konstruktion. Hier solltest du auf die rote Sichtlinie vom Spieler achten. Im folgenden Bild funktioniert der VIS-Blocker nicht, da der Spieler vom Raum 1 in den Raum 2 sehen kann:



Verlängern wir aber den nach oben laufenden Gang, so kann der Spieler nichtmehr in den Raum 2 sehen. Nun funktioniert der VIS-Blocker:



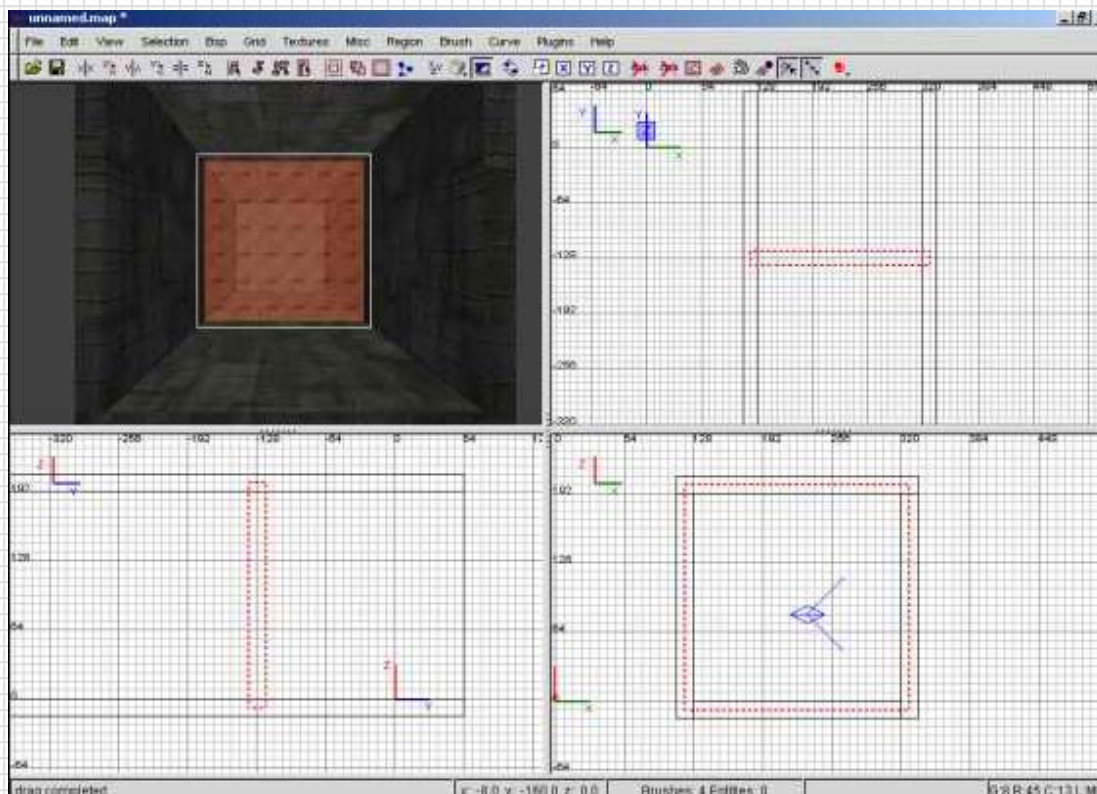
Eine andere Methode ist die sogenannte "Donut-Konstruktion". Hierbei wird in einen Zwischenraum eine Trennwand eingebaut, der die Sicht vom Raum 1 in den Raum 2 verhindert:



Ich habe dir hier mal den Zwischenraum schraffiert, und den "Donut-Brush" markiert. Dieser Brush nimmt nun die Sicht, vom Raum 1 kann man nun nichtmehr in den Raum 2 sehen. Dabei ist zu beachten, dass der Donut-Brush vom Boden bis an die Decke geht, sonst funktioniert er nicht. Ebenso darf sich auf dem Brush keine transparente Textur befinden, sonst kann man ja hindurchsehen, und die ganze Aktion war unnötig.

Hint-Brushs:

Hint-Brushes sind dünne Brushes, die mit der "common/hint"-Textur belegt sind. Sie dienen dazu, die Map in mehrere VIS-Sektoren zu unterteilen. Sie funktionieren jedoch nur dann, wenn sie mindestens 8 Units in die umliegenden Brushes hineinragen. So muss z.B. ein Hint-Brush in einem Gang den Boden, die Decke und die beiden Wände um jeweils 8 Units überschneiden:



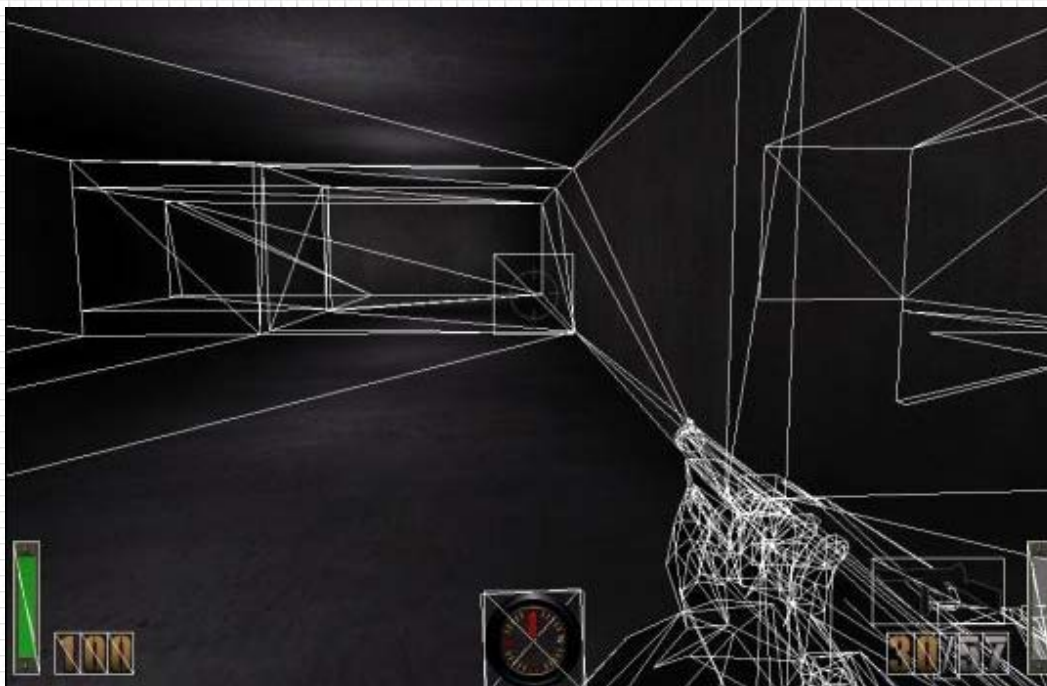
Solche Hint-Brushs werden an Stellen eingesetzt, die dem Spieler die Sicht verdecken, z.B. einer Ecke in einem Gang. In einem geraden Gang macht ein Hint-Brush keinen Sinn, aber ich habe mal einen solchen Bruhs gebaut, dass du die Überschneidungen gut sehen kannst.

Oft macht es sogar Sinn, in deine Map horizontale Hint-Brushes einzubauen - besonders dann, wenn sich dein Level über mehrere Ebenen erstreckt. Hint-Brushes können auch an andere Hint-Brushes angrenzen. Wichtig ist nur, dass es keine freistehenden Kanten gibt, da sonst der Hint-Brush seine Wirkung verliert. Hier kannst du dir z.B. mal die Maps ansehen, die im Radiant mitgeliefert werden, hier werden die Hint-Portale sehr oft eingesetzt.

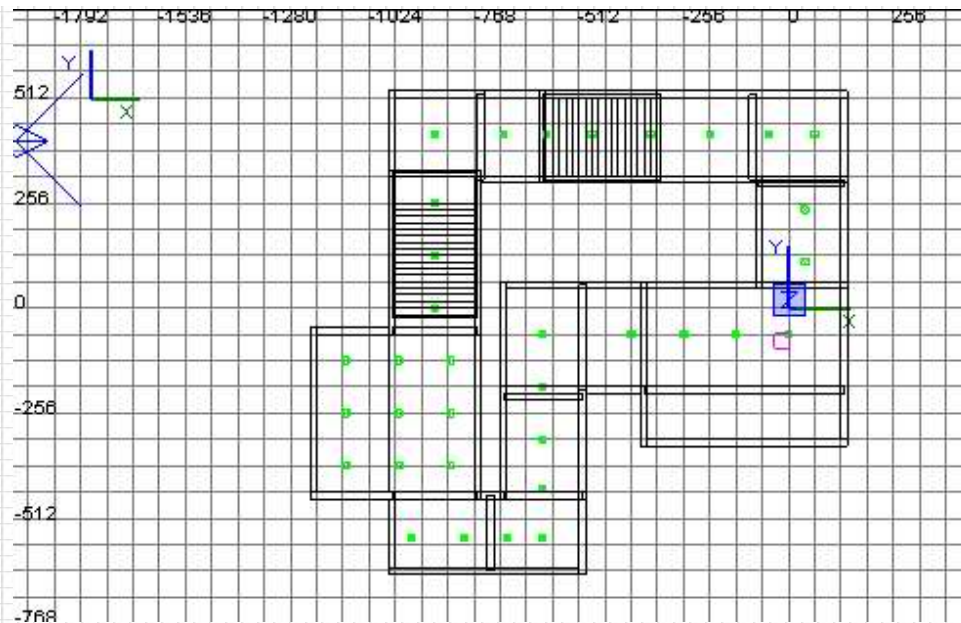
Ich habe dir ja, wie du oben gesehen hast, eine Beispielmap gebaut, in der du die Hint-Portale gut sehen kannst. Hier mal ein Beispiel:



In diesem Bild sieht man die Treppe, doch macht man ein paar Sidesteps, dann sieht es so aus:



Wie du siehst, fehlt die gesamte Treppe, beim weiterlaufen poppen hier ganze Räume zu und dafür andere wieder auf. Jetzt zeige ich dir mal die Map "tutor37.map" aus der Vogelperspektive. Hier erkennst du zunächst nur die normalen Wände:



Jetzt habe ich dir mal alle Hint-Portale markiert. Zusätzlich habe ich noch 2 horizontale Hint-Brushes eingebaut, die direkt über der Treppe liegen. Alles weitere kannst du dir ja dann in der Map selbst ansehen.



Caulk-Textur:

Die Caulk-Textur haben wir ja schon bei den Curves kennengelernt - aber auch bei der Performance erledigen sie wertvolle Dienste. Brush-Seiten, die im Spiel nicht sichtbar sind, können mit der Textur belegt werden, und werden so für die Engine unsichtbar. So bringt diese Textur nicht nur bei der Performance Vorteile, sondern auch beim Compilieren, da hier weniger Licht-Oberfläche berechnet werden muss. Ich gehe hier nicht weiter auf die Hint-Textur ein, da diese Textur so wichtig ist, dass sie ein eigenes Thema verdient.

In der Map habe ich sie jedoch recht häufig eingesetzt - so habe ich jede Brush-Seite, die nicht in der Map zu sehen ist, mit dieser Textur belegt.

[zurück zur Hauptseite](#)

183759



Die Common-Texturen:

Die Common-Texturen sind die mit die wichtigsten Texturen im Spiel. Oft sind sie in den Maps unsichtbar, sie steuern aber wichtige Effekte, z.B. bei Jump pads oder Teleportern. Was man im Level selbst sieht, ist für den Teleporter selbst nicht relevant. Für den Teleporter selbst sorgt z.B. die Textur "common/trigger". Hier findest du eine Auflistung aller Common-Texturen, und ihr "Einsatzgebiet".

common/caulk	Wenn du eine Fläche mit dieser Textur belegst, so wird diese im Spiel nicht berechnet. Dadurch sparst du dir 2 Polygone, weil man 2 Dreiecke zur Darstellung einer Fläche benötigt. Je mehr Flächen du mit der Caulk-Textur belegst, desto kleiner wird deine Lightmap, ausserdem wird deine Performance mit jeder gesetzten Caulk-Textur etwas besser. Mehr will ich hier nicht über diese Textur sagen, da ich dir darüber mehr in einem anderen Thema erzählen werde.
common/ainopass common/ainopasslarge	Hier denken die Computergegner, sie hätten eine Wand vor sich, die sie nicht durchqueren können. Für den Spieler existiert sie jedoch nicht, er kann hindurchgehen.
common/ainosight	Hier kann die AI nicht hindurchsehen und auch nicht hindurchhören.
common/ainosightshot	Hier kann die AI nicht hindurchschieszen.
common/area_portal	Diese Textur ist für die Performance-Verbesserung gedacht. Belegt man einen Brush mit dieser Textur und setzt diesen genau in eine Türe, so wird bei der geschlossenen Türe hinter der Türe nichts berechnet. Erst wenn sie sich öffnet, wird der Teil dahinter berechnet. Auch hier will ich nicht zuviel verraten, mehr in einem anderen Thema.
common/clip	Diese Texturen clippen nur den Spieler. (mehr über die Clip-Texturen erfährst du hier)
common/clip_missile	Hier wird ein Rackengeschosse aufgehalten. (mehr über die Clip-Texturen erfährst du hier)
common/clip_monster	Diese Textur hindert nur die Computergegner daran, an komplexen Gebilden wie Curves usw. hängen zu bleiben und von Mauervorsprüngen herunterzufallen. (mehr über die Clip-Texturen erfährst du hier)
common/clip_shot	Mit dieser Textur lassen sich Schüsse aufhalten. (mehr über die Clip-Texturen erfährst du hier)
common/clipweapon	Diese Textur clippt sowohl den Spieler, als auch Schüsse. (mehr über die Clip-Texturen erfährst du hier)
common/clusterportal	Die Clusterportale funktionieren ähnlich wie Hint-Portale. Sie sind jedoch nur für die AI der Computergegner zuständig. Sie entlasten die CPU beim Berechnen der Laufwege (das Level wird über die Clusterportale in kleine Bereiche unterteilt) und stellt somit mehr

	Rechenleistung für andere Dinge bereit.
common/terrain common/terrain2	Diese Texturen sind einfache Terrain-Texturen. Terrain benötigt eine func_group und einen Metashader, der die Belichtung (Vertex oder lightmap) festlegt. Im Terrain solltest du vermeiden, mit "CSG-Substract" zu arbeiten, da es hier zu hässlichen Darstellungsfehlern kommt.
common/cushion	Fällt der Spieler von einem hohen Baum oder einem Haus, und landet auf dieser Textur, so erleidet er keinen Fallschaden. Du solltest darauf achten, diesen Brush sehr dünn zu mappen, da er solide ist. Würdest du den Brush sehr dick machen, würde der Spieler später in der Luft stehen.
common/hint	Die Hint-Textur ist eine sehr wichtige Textur. Sie verhindern, dass der Spieler zuviel von der Map sieht. Hints wirken leider nur über Winkel, in einem geraden Gang zeigen sie keine Wirkung. Zu dieser Textur habe ich ein eigenes Tut geschrieben, ausserdem findest du weitere Informationen im Performance-Thema.
common/ladder	Diese Textur ist lediglich ein Shader, der dem Spieler die Erlaubnis gibt, an diesem Brush hochzuklettern. Er wird zum bauen von Leitern benutzt.
common/lightgrid	Mit dieser Textur kann man Brushes einhüllen und damit die Lightmap gering halten. Dies ist besonders bei großen Maps wirkungsvoll.
common/nodraw common/nodrawnonsolid	Diese Textur ist durchsichtig und auch nicht solide. So kann man z.B. Feuer damit erstellen.
common/nodrop	Stirbt jemand auf dieser Textur, so lässt er keine Waffen oder andere Items zurück.
common/origin	Belegt man einen Brush mit dieser Textur, so bildet dieser Brush den Ausgangspunkt einer Bewegung, z.B. die "Aufhängung" bei einem Pendel, oder aber als Angelpunkt bei Türen, sowie als Drehpunkt bei einem Radar. Mehr über diese Textur gibts in einem anderen Thema.
common/skip	Was von dieser Textur eingehüllt ist, wird nicht berechnet, nur die Lightmap.
common/slick	Dieser Brush ist dazu da, eine glitschige Oberfläche darzustellen. Hier rutscht der Spieler ab. Über diverse Entity-Keys lässt sich auch die Richtung bestimmen, in die der Spieler rutschen soll.
common/trigger	Diese Textur bildet den Ausgangspunkt für einen Schalter oder eines Vorgangs, z.B. den trigger_teleport oder einen trigger_hurt.

[zurück zur Hauptseite](#)



TerraGen:

Nachdem ich dir schon in dem Thema "Skyboxen" ein paar Dinge über Terragen gezeigt habe, will ich dir hier das Programm und seine verschiedenen Fenster erklären, da man mit Terragen wirklich tolle Umgebungen erstellen kann.

Wenn du hier auf die Menüleiste klickst, kommst du gleich zur Erklärung, was du alles in der Menüleiste einstellen kannst. Klickst du auf die Dialog-Leiste, so kommst du gleich zur Erklärung, wofür du die einzelnen Dialogfenster nutzen kannst:



Zuerst erkläre ich dir die Menüleiste:

In der Menüleiste gibt es 4 verschiedene Menüeinträge:

- World File (Weltdatei)
 - New World (Neue Welt) stellt so gut wie alle Einstellungen von Terragen auf ihren Standard und löscht die Landkarte. Außerdem zeigt es noch einmal den Infobildschirm von Terragen an.
 - Open World (Welt öffnen) öffnet eine gespeicherte Terragen-Welt-Datei (.tgworld).
 - Re-open World (Welt neu öffnen) öffnet die momentane Welt-Datei erneut und macht ungespeicherte Änderungen rückgängig.
 - Save World (Welt speichern) speichert eine Welt.
 - Exit (Beenden) Beendet das Programm.

- View (Ansicht)
 - Hier können Sie auf die verschiedenen Dialogboxen wie "Landscape", "Water", "Cloudscape", "atmosphere" und "lighting" zugreifen.
- Terragen
 - Execute Script (Script aufrufen) ermöglicht Ihnen eine Script-Datei auszuwählen und diese auszuführen. Achten Sie darauf, dass die Einstellungen für die Atmosphäre, für das Rendern (Bildgröße, Details usw.) und Wolken korrekt sind und Sie die richtige Terraindatei geladen haben, bevor Sie das Script starten.
 - Preferences (Voreinstellungen) erlaubt Ihnen, den Wert der Gammakorrektion zu setzen, der beim Zeichnen von Karten genutzt wird. Mit einem passenden Wert ist es einfacher mit den Karten zu arbeiten.
 - Run Scripter (Den Scripter aufrufen) startet ein Utility von Matt Fairclough: Scripter.
 - Run Terranim (Terranim aufrufen) startet Daniel Parnham's Terranim Programm, das wesentlich flexibler als der Scripter ist.
- Help (Hilfe)
 - About (Info) zeigt nochmals das Infofenster Terragens an. Momentan existiert allerdings noch keine richtige Hilfe-Datei.

Nun kommen wir zu den Dialogfenstern:

[Render Controls \(Render Dialog\)](#)

[Landscape \(Landschafts-Dialog\)](#)

[Water \(Wasser-Dialog\)](#)

[Cloudscape \(Wolken-Dialog\)](#)

[Atmosphere \(Atmosphären-Dialog\)](#)

[Lighting \(Beleuchtungs-Dialog\)](#)

[last rendered image \(Render-Bild Dialog\)](#)

[zurück zur Hauptseite](#)

183759



Terragen F.A.Q.:

Wie bei dem F.A.Q. für RtCW gilt auch hier, dass du dir zuerst diese Liste durchsehen solltest, bevor du mir eine Mail schreibst. Natürlich bin ich selbst auch nicht so versiert, was Terragen angeht - jedoch bin ich gerne bereit, dir zu helfen, wenn es bei dir nicht funktioniert.

Problem	Lösung:
Der Himmel bedeckt nicht das ganze Bild bis zur Landschaft, sondern lässt eine schwarze Gegend frei	Du solltest die Wolkengrösse im Wolkendialog erhöhen und es noch einmal probieren.
Wenn Ich Bilder rendere, gehen die Farben nicht ineinander über, sondern bilden "Treppen"	Hier musst du kontrollieren, dass dein Desktop auf TrueColor steht (24 oder 32 Bit Farbtiefe). Auf gespeicherte Bilder hat es allerdings keinen Einfluss. Sollte aber genau hier der Fehler liegen, solltest du die Einstellungen im Dialogfenster "Atmosphäre" ändern.
Das Bild wird nicht richtig dargestellt, z.b. fehlen Teile der Landschaft	Hier musst du kontrollieren, dass die Kamera nicht unter Wasser bzw. unter der Landschaft liegt (also, den z-Wert der Kamera kontrollieren). Hier solltest du bei "Fixed Heigt above Surface" (Fester Punkt über der Oberfläche) einen positiven Wert eintragen. Weiterhin könnte es daran liegen, dass du "Backface Culling" aktiviert hast. Dann solltest du es deaktivieren. Diese Einstellung findest du bei den Render-Einstellungen im Render-Control-Dialog.
Das Wasser sieht in der Entfernung zu glatt aus	Hier solltest du die Amplitude der Wellen erhöhen um diesen Effekt zu reduzieren.
Wie kann ich den Rendervorgang beschleunigen	Hier gibt es viele Möglichkeiten, die Schatten deaktivieren, oder auf 3D-Wolken verzichten.
Wieso sehen niedrige Zoom-Werte so schlecht aus	Echte Kameras benutzen eine gewölbte Linse. Terragen simuliert aber noch nicht den Effekt einer solchen Linse
Wie rechne ich zwischen Sichtfeld und Zoom um	FOV (Field of fiew = Sichtfeld) = $2 * \arctan(1/\text{Zoom})$. Das Sichtfeld wird in Radianten angegeben : Grad = Radianten * 180 / Pi
Was sind DEM-Dateien	Hier handelt es sich um Digital Elevation Maps (digitale Höhenkarten). Damit lassen sich wirklich-existierende Orte in Terragen rendern.

[zurück zur Terragen-Übersicht](#)

[zurück zur Hauptseite](#)





Terragen Lexikon:

Da Terragen einige Fachbegriffe benutzt, die eigentlich nichts mit dem Radiant zu tun haben, habe ich hier eigens ein kleines Lexikon geschrieben, indem du alle Begriffe und deren Erklärung nachlesen kannst:

Bank:

Der Winkel der Neigung der Kamera nach Links oder Rechts. Negative Werte bedeuten eine Drehung entgegen dem Uhrzeigersinn.

Canyonism:

"Schärft" die Grenzen der Täler, so dass ein ähnlicher Effekt wie bei Canyons in der Wüste entsteht (obwohl nicht so betont).

Corona:

Hier wird nicht das simuliert, was die Astronomen die Korona der Sonne nennen, sondern es handelt sich um einen Ring um die Sonne, der die Grenze zwischen der Sonnescheibe und dem umgebenden Himmel verwischt um so teilweise dem Fakt entgegenzuwirken, dass Terragen noch keine Lensflares rendern kann.

Density (Haze):

Die "Effektivität" des Dunst/Nebels.

Density Contrast:

Verändert, wie scharf die Grenzen der Wolken sind und lässt sie unter normalen Einstellungen dunkler erscheinen. Je tiefer der Wert gesetzt wird, desto weicher sind die Wolken.

Density Shift:

Erlaubt Ihnen einzustellen, wieviel des Himmels aus Wolken besteht. Große negative Werte führen zu einem Himmel ohne Wolken, große positive hingegen zu einem völlig bedeckten Himmel.

Detail:

Der Detailgrad, der für die Vorschau und das endgültige Rendern benutzt wird. Je höher er gesetzt wird, desto kleiner sind die gezeichneten Dreiecke.

Exposure:

Nicht einfach nur Helligkeit, sondern es imitiert die Beleuchtungszeit einer echten Kamera.

Glaciation:

Verändert die Landschaft, indem Täler und starke Neigungsveränderungen geglättet werden.

Half-height:

Die Auswirkungen("Effektivität") der atmosphärischen Komponenten sinkt mit dem Steigen der Höhe. Die "Halbwertshöhe" legt fest, nach was für einen Höhenunterschied die Auswirkung nur noch halb so groß wie vorher ist.

Head:

Der Winkel zwischen der Kamera und dem Fokuspunkt.

Pitch:

Der vertikale Winkel des Blickfeldes, gemessen von einer flachen Ebene.

Preview:

Eine Vorschau, die nur klein und mit wenigen Details dargestellt wird, um ein "Gefühl" für die momentanen Einstellungen zu vermitteln.

Realism:

Eine höhere Einstellung sorgt für ein realistischeres Terrain mit weichen Übergängen zwischen hohen und niedrigen Teilen der Landschaft. Ein niedriger Wert hingegen führt zu eine "felsigeren" Landschaft.

Rendering:

Der Prozess des Erstellen eines Bildes von allen benutzen Einstellungen.

Size (waves):

Die Größe der Wellen.

Smoothing:

Stell ein, wie weich die Landschaft sein wird. Ein zu hoher Wert kann Probleme mit der Realismus-Einstellung ergeben und ungewollte Ergebnisse hervorrufen.

Surface Map:

Ähnlich wie ein Textur, allerdings handelt es sich nicht um ein Bild. Die Oberflächengrafik hängt von der Höhe und der Steigung ab, je nachdem, was der Benutzer eingestellt hat.

Surface:

Die Land- und die Wasserschicht zusammen, eben die Oberfläche.

Zoom:

Eine Erhöhung des Zooms bringt die Kamera nicht näher ans Ziel, sondern vergrößert, wie in Wirklichkeit, das Blickfeld.

[zurück zur Terragen-Übersicht](#)

[zurück zur Hauptseite](#)

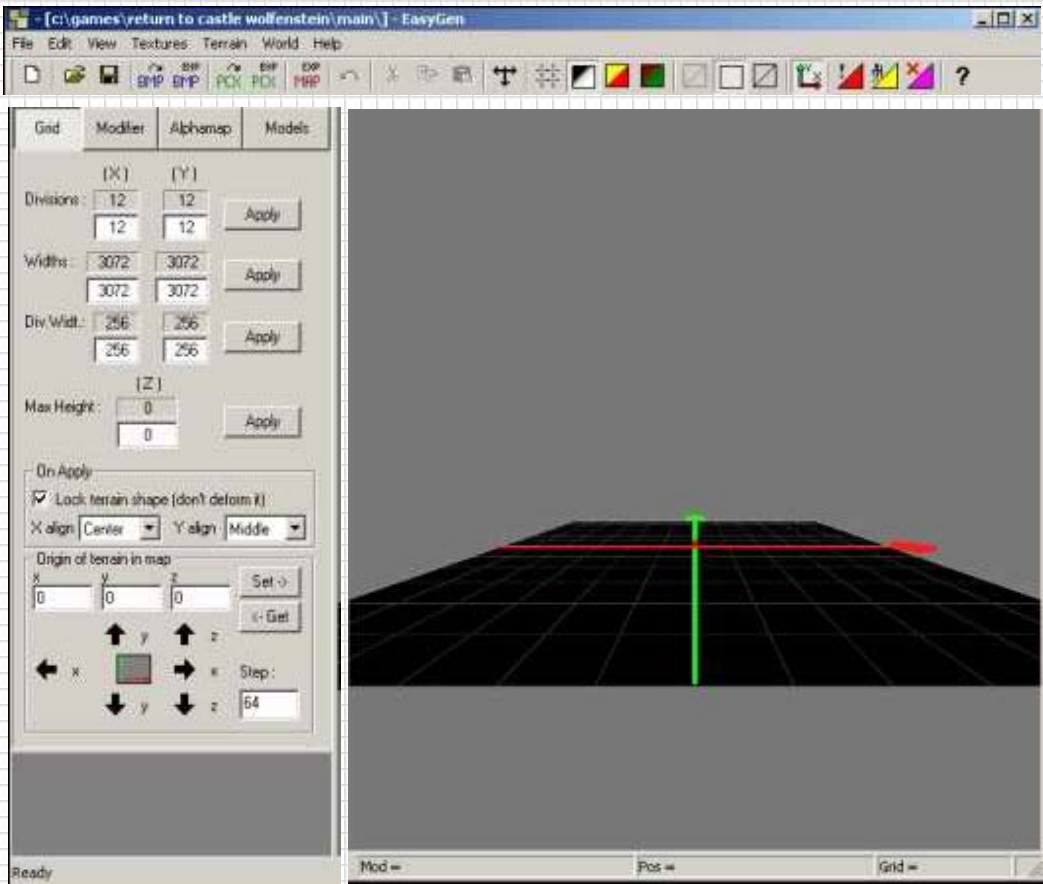
183759



EasyGen:

Sicher hast du schon die verschiedensten Terrains gesehen. Nun stellt sich natürlich die Frage, wie man eine solche Landschaft wohl selbst erstellen kann. Dazu benötigt man nur das Programm "EasyGen". Damit lässt sich wirklich leicht und einfach tolle Terrains und ganze Aussen-Areale erstellen.

Wenn du hier auf die Menüleiste klickst, kommst du gleich zur Erklärung, was du alles in der Menüleiste einstellen kannst. Klickst du auf die Dialogfenster, so kommst du gleich zur Erklärung, wofür du die einzelnen Dialogfenster nutzen kannst:



Zuerst erkläre ich dir die Menüleiste:

In der Menüleiste gibt es 7 verschiedene Menüeinträge:

- File (Datei)
 - New (Neue): hiermit lässt sich ein neues, leeres Terrain laden
 - Open (öffnen): öffnet eine gespeicherte EasyGen-Datei (.egn)
 - Save (speichern): hiermit kannst du das Terrain abspeichern
 - Save as...(speichern als): Das selbe wie speichern, jedoch kannst du hier einen anderen Pfad angeben
 - Export (exportieren): hier kannst dir die *.map Datei, die Höhenkarte, die Alpamap und den Shader exportieren

- Import (Importieren): Hier kannst du eine *.map Datei, eine Höhenkarte und eine Alphamap importieren
- Exit (Beenden): Beendet das Programm
- Edit (Editieren) :
 - Undo (rückgängig): Wenn du einen Fehler oder einen Arbeitsschritt gemacht hast, der dir nicht gefällt, kannst du diesen über die Funktion "rückgängig" diesen Fehler wieder rückgängig machen
 - Cut (schneiden): Hiermit lässt sich ein selektierter Teil des Terrains entfernen
 - Copy (kopieren): Hiermit kannst du einen selektierten Teil des Terrains kopieren
 - Paste (ausschneiden): Schneidet einen selektierter Teil des Terrains aus
 - Entire Flip / Mirror: Spiegelt das gesamte Terrain
 - Preferences
- View (Ansicht)
 - Toolbar (Menüleiste) schaltet die Leiste an oder aus
 - Status bar (Status-Leiste) schaltet unten die Statusleiste an oder aus
 - Wires (Draht): wählt die Drahtgitter-Ansicht
 - Flat (Flach): wählt die flache Ansicht
 - Texture (texturiert): wählt die texturierte Ansicht
 - Alphamap (Alphakarte): wählt die Alpha-Ansicht (mit den verschied. Platzhaltern, die die Texturen darstellen.
 - ... +noframe (keine Unterteilung): Das Terrain besitzt keine Unterteilung
 - ... + quad frame: Das Terrain wird mit Vierecken gerastert dargestellt.
 - ... + tris frame: Das Terrain wird mit Dreiecken gerastert dargestellt.
 - Flat Shade
 - Flat Smooth
 - Culling
- Textures (Texturen):
 - Add Folder (Pfad hinzufügen): Hier kannst du Ordner laden, in denen sich dann die Terrain-Texturen befinden.
 - Clear (löschen): löscht alle bereits geladenen Texturen
 - Zoom in: Zoomt in die Textur hinein
 - Zoom out: Zoomt aus der Textur heraus
 - Zoom 1:1: Stellt die Textur in original-Grösse dar
- Terrain (Landschaft):
 - Clear excluded Tris (löscht ausgeschlossene Dreiecke): löscht ausgeschlossene Dreiecke
- World (Welt):
 - Remove (Entfernt): entfernt alles
 - Show (Zeigt): Zeigt alles
- Help (Hilfe)

- About Easygen (über Easygen): Hier erscheint der Autor des Programms, ausserdem noch eine Liste aller Shortcuts die in Easygen existieren.

Nun kommen wir zu den Dialogfenstern:

- [Grid \(Koordinaten\)](#)
- [Modifier \(Modifizierer\)](#)
- [Alphamap \(Alphakarte\)](#)
- [Models \(Modelle\)](#)

Und zum Schluss noch das Navigationsfenster:

Hierzu gibts nicht so viel zu sagen. Es entspricht ungefähr dem 3D-Ansichts Fenster des Radianten. Jedoch ist die Steuerung etwas gewöhnungsbedürftig. Deswegen hier mal eine kleine Liste, wie man sich in diesem Fenster bewegt:

Rechte Maustaste gedrückt halten = vor und zurück
Linke Maustaste gedrückt halten = Sicht hoch und runter
Beide Maustasten gedrückt halten = Höhe regulieren

[zurück zur Hauptseite](#)

183759



Easygen F.A.Q.:

Leider ist Easygen auch nicht so ganz leicht zu benutzen, so gibt es einige Dinge, die unklar sind, bzw. die man ausversehen falsch macht:

Daher habe ich dir hier einige Fehler/Fragen und natürlich die Antworten darauf aufgelistet, die dir vielleicht selbst während deiner Arbeit mit Easygen passieren können:

[Ich habe schwarze Flecken in meinem Terrain?](#)

[Was fang ich eigentlich mit den ganzen Dateien an, die mir Easygen ausspuckt?](#)

[Irgendwie sieht mein Terrain so eckig aus?](#)

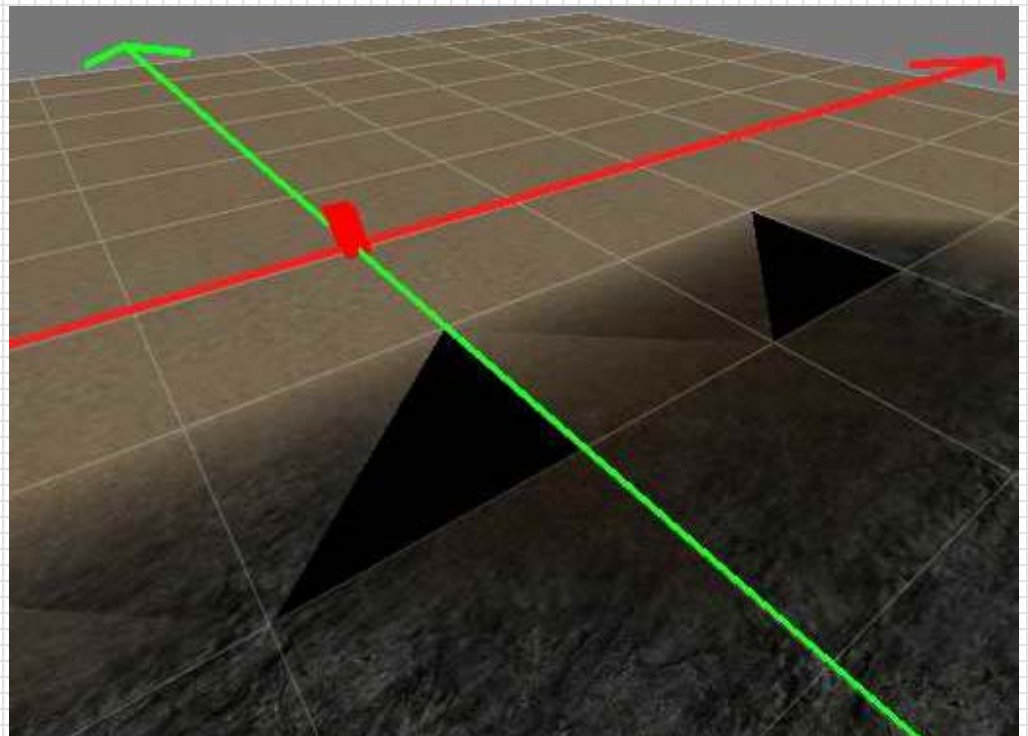
[Wenn ich meine Map in Easygen importiere, erscheint nur ein grosser, viereckiger Kasten?](#)

- Ich habe schwarze Flecken in meinem Terrain?

Dieser Fehler tritt immer dann auf, wenn viele verschiedene Texturen zusammentreffen. So kann dieser Fehler aber schon ab 3 "Posterize"-Ebenen auftreten (also 3 verschiedene Texturen). Dabei kann Easygen nichtmehr die Übergänge richtig darstellen.

Um diesen Fehler zu beheben, musst du dir eine Textur suchen, dann "Shift + Strg + die linke Maustaste" drücken und die schwarzen Flächen übermalen.

Dieser Fehler liegt nicht an Easygen selbst - es ist vielmehr ein Problem im Shader, da hier nicht mehr als 1 Farbübergang dargestellt werden können.



- Was fang ich eigentlich mit den ganzen Dateien an, die mir Easygen ausspuckt?

Ganz einfach, es gibt 3 Dateien, die Easygen exportiert, den Shader, die *.map-Datei sowie die Alphamap. Die Alphamap muss in den Root-Ordner der PK3-Datei, der Shader wie gewohnt in den "scripts"-Ordner. Die Map-

Datei musst du ja noch kompilieren, bzw. das restliche Level mit einbauen. Die fertige BSP-Datei kommt ebenfalls wie immer in den "Maps"-Ordner.

WICHTIG: Du darfst natürlich nicht vergessen, die ganzen eigenen Terrain-Texturen mit in den "Textures"-Ordner zu packen.

- Irgendwie sieht mein Terrain so eckig aus?

Das liegt daran, dass du dein Terrain noch nicht "nachbearbeitet" hast. Dazu gibt es bei dem Modifier-Ordner einen Modifier, der sich "Smooth" (Glätten) nennt. Mit diesem kannst du ganz zum Schluss nochmal ordentlich über deine Map fahren und so nochmal ein paar hässliche Ecken etwas retuschieren.

- Wenn ich meine Map in Easygen importiere, erscheint nur ein grosser, viereckiger Kasten?

Easygen importiert immer die gesamte Map. Was du von der Map siehst, ist nur die Skybox, bzw. die Umrandung deiner Map. Du solltest die Skybox im Radiant löschen bzw. eine zweite Map ohne Skybox anlegen.

[Zurück zur Easygen-Anleitung](#)

[zurück zur Hauptseite](#)

183759



Easygen Lexikon:

Easygen hat natürlich ein paar Begriffe, die man im Radiant nicht braucht, bzw. die dort nicht auftauchen. Hier mal eine Erklärung einiger Begriffe:

Alphamap:

Hier werden statt den Texturen die Farben aus einer Palette genommen, die dann später von Easygen in den Shader umgerechnet werden. Die Farben sind so zu sagen eine Art Platzhalter für die richtigen Texturen.

Graustufen-Bild:

Dieses Bild entspricht einer normalen, topographischen Ansicht eines Geländes. Dunkle Flächen sind weniger hoch, helle Flächen sind hoch.

Modifizier:

Sie sind quasi die Werkzeuge in Easygen, mit ihnen kann man das Terrain verändern, verformen usw.

Posterize:

Hier kannst du Easygen veranlassen, selbst Index-Colors zu setzen. Und zwar so, dass ein Verlauf entsteht, den man mit einem Graustufen-Bild vergleichen kann. Nimmt man Blau als Grundfarbe, so ist helles blau eher auf den Bergen vertreten, dunkles Blau hingegen in den Ebenen. Nimmt man Grün als Grundfarbe, so ist helles Grün eher auf den Bergen vertreten, dunkles Grün hingegen in den Ebenen

Preferences:

Hier kannst du genau wie in den Preferences vom Radiant alles einstellen, also Pfade usw.

Selected Index Color:

Die Farbe, die hier angegeben ist, repräsentiert sozusagen eine Textur.

[Zurück zur Easygen-Anleitung](#)

[zurück zur Hauptseite](#)

183759