Tutorial zum Erstellen neuer Einheiten: Vom Modell bis zur fertigen Einheit

Turin Turambar, überarbeitet von Annuminor

27. Dezember 2018

Inhaltsverzeichnis

Hi

In diesem Tutorial möchte ich euch erklären, wie man eine komplett neue Einheit für das Spiel „Der Herr der Ringe: Die Schlacht um Mittelerde 2“ (SuM) und das Addon „Aufstieg des Hexenkönigs“ von Electronic Arts erstellt. Dieses Tutorial setzt bestimmte Tools und das Starten der eigenen Mod per „-Mod Command“ vorraus. Dazu später mehr.

ACHTUNG: Ich gehe in diesem Tutorial davon aus, dass ihr die Programme und die Spiele in den Standardpfad installiert, sprich `C:\Programme\`, sollte dies nicht der Fall sein, so müsst ihr die Pfade beim Kopieren entsprechend verändern.

1 Tools

Zum Modden für SuM werden einige Tools benötigt, damit man das Spiel entsprechend modifizieren kann. Diese Tools werde ich hier nun aufzählen und erklären wie man sie einrichtet.

1.1 Gmax und Renx

Gmax und Renx mit dem Plugin „Coolfile W3D-Importer“: Das brauchen wir zum Modellieren. Die Software könnt ihr hier herunterladen:

`https://www.the3rdage.net/download-39` : W3D Importer
`https://www.the3rdage.net/download-43` : Gmax
`https://www.the3rdage.net/download-45` : Renx

Installiert zuerst Gmax über den Installer, danach Renx ebenfalls mit dem Installer. Jetzt startet ihr Renx. Es sollte ein Fenster erscheinen in dem ihr eine Seriennummer eintippen müsst. Die Nummer erhaltet ihr„ wenn ihr euch hier registriert:

`https://www.turbosquid.com/RegisterGmax/Index.cfm`

Kopiert den Key in das Fenster und ihr könnt das Programm starten. Nach dem erfolgreichen Start schließt ihr es wieder.

Danach entpackt ihr die `W3DImporter.rar` mit einem beliebigen Entpackungsprogramm. Es sollte eine Datei namens `w3dimporter.ms` erscheinen. Diese kopiert ihr nach `C:\Programme\gmax\plugins\`. Öffnet nun wieder RenX und klickt oben in der Leiste auf „Customize - Customize User Interface“. Wählt den Reiter „Toolbars“ aus. Dort wählt ihr den W3D Importer aus und zieht ihn dann in die Userbar des Programmes. Nun könnt ihr per Klick auf den Button in der Userbar den W3D Importer öffnen.

1.2 GIMP

Wir werden Gimp als kostenlose Alternative zu Adobe Photoshop oder Ähnlichem benutzen, da diese einfach zu teuer sind. Gimp ist das wohl beste freie Bildbearbeitungsprogramm, das es gibt. Ihr könnt es hier herunterladen:

`https://www.gimp.org/downloads/`

Außerdem braucht ihr ein DDS-Plugin für Gimp, dieses könnt ihr hier herunterladen:

`http://nifelheim.dyndns.org/~cocidius/download.php?filename=gimp-dds-win32`

Installiert nun Gimp mit dem Installer auf eurem Computer. Ihr werdet gefragt ob ihr GTK installieren wollt - wenn ihr dieses schon auf eurem PC installiert habt, dann müsst ihr dies nicht installieren, sollte dies nicht der Fall sein, dann müsst ihr GTK mit installieren. Startet nun einmal Gimp und schaut ob es funktioniert. Schließt das Programm danach wieder.

Entpackt nun die .zip Datei des dds-Plugins. Es wird eine `dds.exe` erscheinen. Diese kopiert ihr in den Ordner `C:\Programme\GIMP-2.0\lib\gimp\2.0\plug-ins`. Startet nun Gimp erneut. Nun solltet ihr über „Datei - Öffnen“ auch .dds-Dateien öffnen können.

1.3 FinalBig

Downloadlink:

`http://wagnerma.de/yinger/download.php?index=1`

Dieses Programm benötigt ihr zum Bearbeiten von .big-Dateien. Da fast das ganze Spiel auf diesen basiert, ist dieses Tool wohl das Wichtigste, ohne dieses funktioniert gar nichts. Extrahiert die .zip-Datei und startet die FinalBig.exe, es sollte sich das Programm öffnen. Mit diesem könnt ihr jetzt über „File - Open“ .big Dateien öffnen.

1.4 Das BFME Mod SDK

Dieses ist das offizielle Mod SDK von EA Games. Eigentlich brauchen wir aus dieser Tool- und Script-Sammlung nur ein Tool, den W3D Viewer. Hier ist der Download Link:

`http://files.ea.com/downloads/eagames/official/bfme2/BFME2_ModSDKv2.exe`

Startet die EXE nach dem Download, dann wird das Mod SDK nach `C:\BFME2 MOD SDK\` installiert. Hinter `C:\BFME2 Mod SDK\W3DView\W3DView.exe` verbirgt sich der W3DViewer, mit dem ihr .w3d-Dateien, sprich, die Modelle und deren Animationen, betrachten könnt.

1.5 Sy's Asset Builder

Downloadlink:

`http://www.the3rdage.net/download-111`

Mit diesem Asset Builder erstellt ihr, wie der Name schon sagt, eine `asset.dat`. In dieser werden die Modell- und Texturnamen gespeichert, dazu später mehr. Den Asset Builder könnt ihr per Installer installieren. Es kann vorkommen, dass der Asset Builder unter Vista nicht richtig funktioniert, sollte dies der Fall sein, so startet einmal den AssetCacheBuilder.exe aus dem Mod SDK, der liegt unter `C:\BFME 2 MOD SDK\AssetCacheBuilder\AssetCacheBuilder.exe`, danach sollte Sy's Asset Builder richtig funktionieren.

1.6 Die lotr.str - Dateien

Diese dienen zur Bearbeitung der Texte im Spiel: Beide .str-Dateien sind auf dem Stand des aktuellen Patches, für das Original also 1.06, für das Addon 2.01. Download dieser Dateien:

`http://tombombadilmod.to.funpic.de/lotrstrs.rar`

2 Das -Mod Command

... oder auch die Vorbereitung.

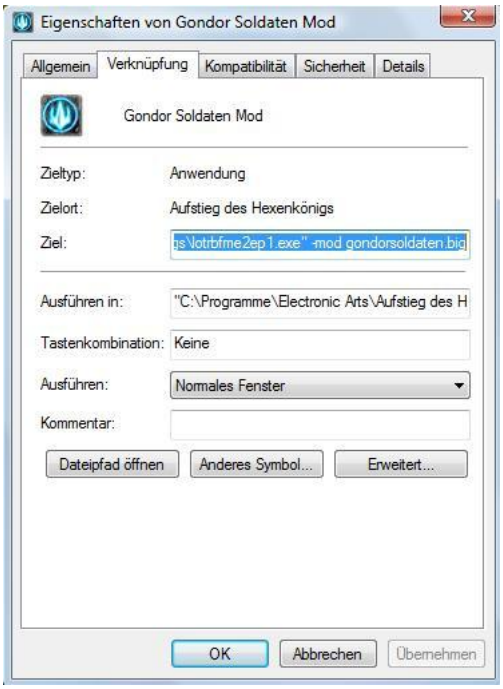
Das Mod- Command ist eine Technik, wie man Modifikationen für das Spiel „Schlacht um Mittelerde 2“ startet. Diese Technik ist die erste, allerdings auch wohl die ausgeklügelste. Wie man diese Technik anwendet werde ich euch nun erklären.

Zuerst geht ihr in den SuM 2 Ordner oder in den SuM 2 Addon Ordner, je nach dem welches Spiel ihr moddet wollt. Dort werdet ihr eine Datei finden, die `INI.big` heißt. Diese benennt ihr nun um, so könnt ihr z.B. dieser Datei den Namen eurer Mod (z.B. `auenland.big`) geben. Ich nenne die Big `gondorsoldaten.big`, da wir diese Einheit hinzufügen wollen. Danach kopiert man die Desktopverknüpfung des Spiels und benennt sie um, ich nenne

sie `Gondorsoldaten Mod`. Dann öffnet ihr per Rechtsklick die Eigenschaften der Verknüpfung und fügt bei Ziel hinten

`[leertaste]-mod gondorsoldaten.big`

ein. Der Bigname muss der selbe sein wie der, den ihr oben verwendet habt. Es sollte nun so aussehen:



Danach packt ihr eure .big-Datei in folgenden Ordner:

Unter XP:
`C:\Dokumente und Einstellungen\Benutzername\Anwendungsdaten \Meine Schlacht um Mittelerde 2 Dateien`, oder fürs Addon `Meine Aufstieg des HexenkÃűnigs Dateien`.

Unter Vista lautet der Pfad:
`C:\Benutzer\Username\AppDataRoaming\Meine` ...

Jetzt könnt ihr euren Mod über die Verknüpfung starten. Die Vorbereitungen sind nun abgeschlossen.

3 Das Modellieren

Nun wollen wir mit dem Modellieren anfangen. Wir fangen also jetzt mit dem eigentlichem Modding an.

3.1 Laden und grundlegendes Bearbeiten

Öffnet zuerst FinalBig und öffnet mit diesem Programm zuerst die `w3d.big` des Original Spiels.

Achtung, für den nächsten Schritt wird viel Festplattenspeicer benötigt. Nun wählt ihr mit FinalBig „Edit - Extract All“, nun gebt ihr einen Ordner an, wohin die Dateien extrahiert werden sollen. Danach wird FinalBig einige Zeit arbeiten.

Sollte FinalBig fertig sein mit arbeiten, öffnet ihr die `textures0.big` und extrahiert auch diese. Diese Schritt wiederhohlt ihr mit folgenden big-Dateien: `textures1.big`, `textures2.big`, `textures3.big`, `textures4.big`.

Falls ihr das Addon installiert habt und eure Mod für das Addon ist, so wiederholt ihr die Schritt oben mit genau den selben Dateien aus dem Addon Hauptordner.

Nun habt ihr alle Modelle und Texturen aus dem Spiel extrahiert.

Öffnet jetzt RenX. Wir wollen einen Gondor Speerträger hinzufügen, der so aussieht wie ein Gondor Soldat, aber statt einem Speer ein Schwert hat. Deshalb müssen wir zuerst das Modell des Gondor Soldaten importieren. Die richtigen Modelle zu finden ist schwer, wenn man das System dahinter nicht versteht. Das System funktioniert so:

Die Modelle sind in Ordnern, die z.B. `gu` heißen. Der Aufbau ist wie folgt:

```
art\w3d\DieErstenBeidenBuchstabenDesModells\Modellname
```

für Texturen ist es dieser Aufbau:

```
art\compiledtextures\DieErstenBeidenBuchstabenDerTextur\Textur
```

Damit wissen wir aber immer noch nicht wie wir an den Namen des Modells kommen. Doch auch dieser hat ein System. Die ersten beiden Buchstaben sind immer Angaben auf das Modell bezogen. Der 1. Buchstabe gibt das Volk an, der zweite ob es eine Einheit oder ein Gebäude ist.

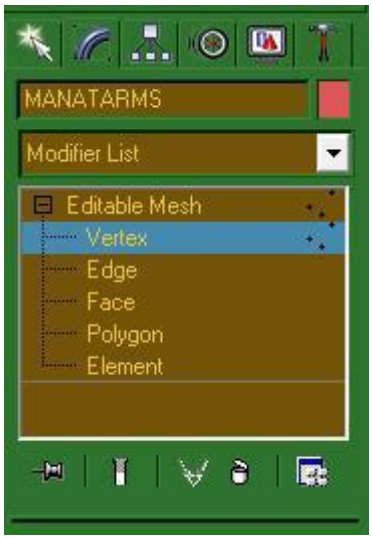
Wir suchen eine Einheit von Gondor, der erste Buchstabe muss also ein „g“ sein, da er zu Gondor gehört. Der zweite Buchstabe ist ein „u”, für Unit. Also befindet sich das Gondor Soldaten Modell in dem Ordner `gu`. Jetzt kommen wir zum Dateinamen. Diese sind meistens Beschreibungen. Das Modell welcher wir suchen heißt `gumaarms_skn`. „maarms“ ist eine Abkürzung für Man At Arms, was so viel heisst wie Bewaffneter Mann. Das „_skn sist“ immer die Endung für ein Modell, „skl“ ist die Endung für ein Skeleton, alle anderen Änderungen sind Animationen.

Skeletons benötigt man für die Animationen, die Animationen bewegen immer nur bestimmte Bones am Modell, also muss man das Modell an die verschiedenen Bones des Skeletons binden, damit dieses die Animationen richtig ausführt. Dazu später mehr.

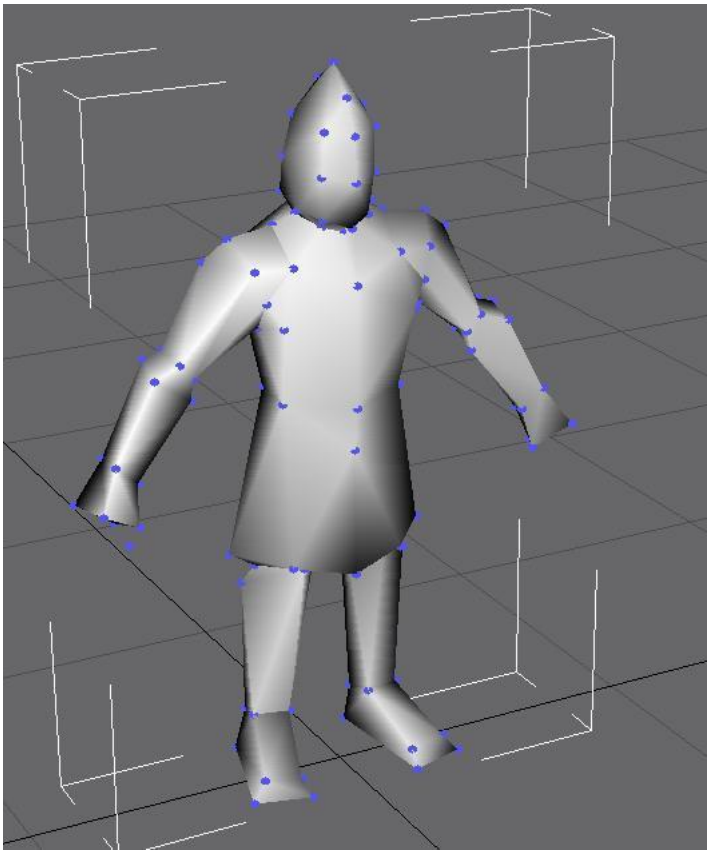
Wir importieren also mit dem vorher richtig eingestellten W3D Importer das Modell `gumaarms_skn` in RenX. Wählt nun das Fenster unten rechts aus und klickt auf diesen Button:

Nun solltet ihr nur noch dieses eine Fenster sehen. An dieser Stelle möchte ich euch die Steuerung des Programms etwas näher bringen. Mit dem Mausrad zoomt ihr rein und raus. Durch klicken auf das Mausrad und paralleles Bewegen der Maus könnt ihr eure Sichtweise ändern. Per Klick auf diesen Button könnt ihr die Kamera drehen:

Klickt nun mit der Maus auf die Box die über dem Modell liegt und entfernt sie mit „Entf“. Dasselbe tut ihr mit dem Schild und dem schwarzem Feld am Schwert. Wählt jetzt das eigentliche Modell aus und klickt rechts auf dieses. Wählt „Hide Selection“. Das Modell ist nun versteckt. Nun solltet ihr Blaue Würfel sehen, die miteinander verbunden sind. Das sind die Bones. Wählt alle per Ziehen eines Quadrates aus und löscht sie. Klickt jetzt irgendwo rechts und wählt „Unhide All“. Nun solltet ihr euer Modell wieder sehen. Wählt es aus und klickt rechts auf das Modell, wählt „Convert To - Convert To Editable Mesh“. Jetzt klickt ihr rechts bei Editable Mesh auf das + und wählt Vertex, hier ein Bild zur Erklärung:



Jetzt sollten am Modell blaue Punkte erscheinen. Durch Verschieben oder Löschen dieser Punkte könnt ihr das Modell bearbeiten. Wir löschen zuerst die Punkte, die die Schlaufe für das Schild ergeben, diese Schlaufe brauchen wir nicht mehr, da wir das Schild vorher gelöscht haben. Klickt zuerst auf die Punkte und dann auf „Entf“. Wenn ihr zuerst einen Punkt auswählt und während des Auswählens des nächsten Punktes „Strg“ drückt, könnt ihr mehrere Punkte gleichzeitig bewegen oder löschen. Löscht danach die Punkte die das Schwert ergeben, sowie die Vertex Punkte der Scheide. Euer Modell sollte nun so aussehen:



Speichert jetzt euer Modell per „File Save“ irgendwo als .gmax-Datei ab. Dieses Dateiformat hat den Vorteil, dass z.B. die Verlinkung erhalten bleibt und ihr nicht nach jedem importieren immer wieder neu verlinken müsst.

3.2 Texturen und neue Bones für den Speer

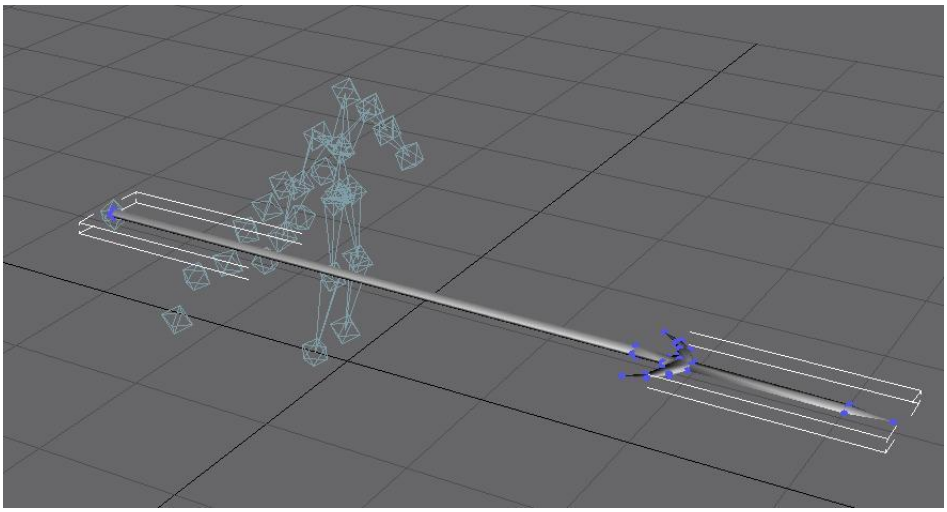
Jetzt machen wir einen kleinen Abstecher in Sachen Texturen. Wählt das Modell aus und drückt „M“, damit der Material Editor erscheint. Geht in

den Reiter „Pass1“. Bei „Specular“ und bei „Emissive“ stellt ihr die Farbe Weiß ein. Und drückt danach diesen Button:

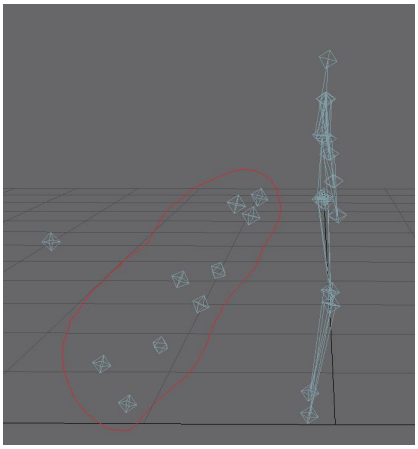
Jetzt wechselt ihr in den Reiter „Textures“. Bei „Stage 0 Texture“ seht ihr nun welche Textur euer Modell benutzt. Da RenX allerdings nur tga-Dateien lesen kann und die Texturen von EA alle im dds-Format vorliegen, müssen wir die EA Textur konvertieren. Das tolle am tga-Format ist, dass das Spiel auch dieses lesen kann.
Startet nun Gimp. Der Aufbau bei den Texturen ist genau der selbe wie bei den Modellen, wir gehen also nach `art\compiledtextures\gu\gumanatarms.dds`. Diese öffnen wir mit Gimp. Achtung: Man muss Gimp erst starten und dann per „Datei - Öffnen“ die dds-Datei öffnen, anders kann das dds-Plugin leider die Textur nicht laden. Nun solltet ihr die Textur sehen.
Wählt in dem Fenster des Bildes „Datei - Speichern unter...“, wählt jetzt euren Speicherort aus und speichert die Datei als tga-Datei. Bitte mit einem neuem Namen, da sonst im Spiel auch die Textur der normalen Soldaten verändert wird. Schließt danach Gimp. Klickt nun auf den Button, auf dem der Texturname steht. Es sollte sich ein Explorer öffnen. Wählt dort eure vorher konvertierte tga-Datei aus. Klickt danach diesen Button:

Eure Textur sollte nun angezeigt werden, so sieht es jetzt bei mir aus, nicht wundern, ich habe eine höher aufgelöste Textur aus meiner Mod genommen, deswegen sollte meine Textur schärfer sein als eure:

Speichert das Modell nun erneut ab. Kommen wir jetzt zum Speer. Wir nehmen als Vorlage den Speer der Turmwachen. Wählt also zuerst euer Modell aus, klickt rechts auf dieses und wählt „Hide Selection“. Wir wollen auch die Animationen der Turmwachen benutzen. Wir importieren das Modell der Turmwachen, `gutwrgrd_skn`. Wählt die Box über dem Modell aus und klickt rechts auf sie, wählt „Hide Selection“. Das selbe macht ihr mit der Box um dem Speer.
Wählt nun das Modell der Turmwache aus und wählt „Convert To - Convert To Editable Mesh“. Dann wählt ihr wie vorher schon Vertex aus. Löscht nun alle Punkte die nichts mit dem Speer zu tun haben. Es sollte nun so aussehen:



Verlasst nun den Punkt Vertex und hided auch den Speer. Jetzt solltet ihr nur noch die Bones haben. Da unser Modell im Gegensatz zu den Turmwachen aber keinen Umhang hat, können wir die Bones des Umhangs löschen, sie würden uns nur beim Verlinken behindern. Dreht die Kamera so, dass es so aussieht wie auf dem Screen. Die Bones im Rotem Rahmen auf dem Screen löscht ihr.



Klickt nun irgendwo rechts hin und wählt „Unhide All“. Hidet danach wieder die beiden Boxen. Wie man nun sieht, passt das Modell der Soldaten noch nicht zum Skeleton, dieses passen wir nun an. Die Arme des Soldaten sind nicht hoch genug. Klickt also wieder auf das Modell des Soldaten und wählt die Vertex Punkte aus. Wählt das „Verschieben“ Werkzeug (die Pfeile in alle vier Richtungen), wählt den Oberkörper aus und verschiebt ihn ein wenig nach oben, so dass er auf die Bones passt. Auf dem Screen unten könnt ihr sehen welche Punkte ihr auswählen musst. Verschiebt die Punkte so weit nach oben, bis die Schultern des Modells auf den Bones der Schultern liegen.

Danach wählt ihr die Vertexpunkte der Unterarme und der Hände und verschiebt sie so hoch, dass die Unterarme auf den Bones der Unterarme liegen. Am Ende wählt ihr dann die Vertex Punkte der Hände und zieht sie so noch oben, dass sie auf den Bones der Hände liegen. Danach wählt ihr noch den

Speer aus und schiebt ihn in die Hand des Soldaten. Falls ihr wollt könnt ihr noch die Textur des Speeres hinzufügen, genauso wie oben die Textur des Soldaten, dies ist aber nicht nötig. Euer Modell sollte nun so aussehen:

Klickt jetzt wieder irgendwo rechts hin und wählt „Unhide All“.

3.3 Exportsettings und Verlinken

Jetzt kommen wir zu den Exportsettings und dem Verlinken.

Wählt zuerst die Box über dem Model aus, klickt auf diesen Button und wählt W3D-Tools.

Jetzt zeige ich euch welche Exportsettings ihr dort für die einzelnen Teile angeben müsst.

AABox (Box über dem Modell)	ForgedBlades (Box um das Schwert)	Das Modell (Schwert, Körper etc.)

Würden wir das Modell jetzt zum W3D Format exportieren, würde der Forged_Blade-Effekt aber immer noch nicht richtig angezeigt werden. Wählt deshalb den Forged_Blade-Effekt aus und öffnet den Material Editor durch das Drücken der Taste „M“. Wechselt wieder in den Reiter „Pass1“.

Jetzt stellt ihr wieder „Specular“ und „Emissive“ auf weiß und drückt den Button wie oben beschirben. Alle 4 Einträge sollten jetzt schwarz werden. Ändert die Farbe von „Emissive“. Die Farbe gibt die Farbe des Effektes im Spiel an. Habt ihr die Farbe geändert wechselt ihr in den Reiter „Shader“. Wählt bei „Blend Mode“ „Add“. Drückt den Button zu anzeigen und beendet den Material Editor. Speichert euer Modell nun ab. Jetzt kommen wir zum schwierigsten Teile des Modellierens, dem Verlinken. Versteckt alles, so dass ihr nur noch die Bones seht. Wählt nun alle Bones aus. Klickt zuerst auf diesen Button:

Danach auf diesen Button:

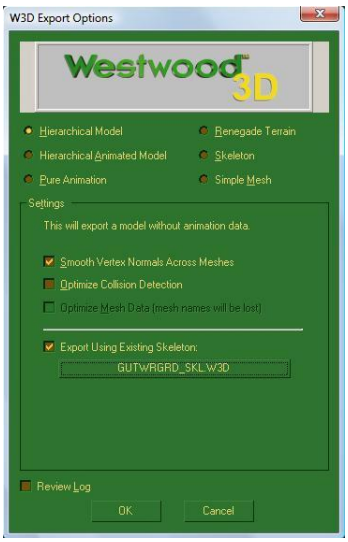
Klickt nun auf „WWSkin“, danach auf „Add Bones“. Es sollte ein Fenster erscheinen in dem alle Bonenamen blau leuchten, sollte dies nicht der Fall sein, so wählt alle Bones in dem Fenster aus. Wenn ihr alle ausgewählt habt klick ihr auf „OK“. Klickt danach irgendwo auf dem Feld hin, dort erscheint nun

der WWSkin. Wählt wieder „Unhide All“. Versteckt wieder die Boundingbox und den Forged_Blade Effekt. Klickt nun auf diesen Button:

Klickt dann auf den WWSkin und zieht zum Menschen. Danach zieht ihr das ganze nochmal, diesmal allerdings vom WWSkin zum Speer. Wählt nun den Speer aus und klickt auf diesen Button:

Klickt auf „WWSkin Binding“ und wählt „Vertices“. Nun solltet ihr wieder die blauen Punkte sehen. Wählt alle aus und klickt auf „Link to Bone by Name“. Wählt den Bone „B_SWORD“ und klickt auf „Ok“. Wählt jetzt das Soldaten Modell aus und geht wieder auf „WWSkin Binding“ und „Vertices“. Wählt wieder alle Bones aus, doch klickt diesmal auf „Auto-Link“. Danach wählt ihr die Vertices an der rechten Hand aus und wählt wieder „Link to Bone by Name“. Hier wählt ihr den Bone „B_HANDR“. Das selbe macht ihr mit der rechten Hand, wählt dort aber den Bone „B_HANDL“. Danach auch noch mit dem Kopf, dort wählt ihr den Bone „B_HAD“. Speichert das Modell und Unhidet die Bounding Box und den Forged Blade Effekt.

Wählt nun „File - Export...“. Wählt den Dateityp .w3d. Und speichert die Datei dort, wo ihr die Animationen und das Skeleton der Turmwachen entpackt habt. Es sollte ein Fenster erscheinen, wählt dort folgenden Einstellungen:



Und klickt auf „OK“. Kopiert jetzt die Texturen die ihr vorhin in das .tga Format konvertiert habt in den Ordner des gerade exportierten Modells und

der Animationen der Towerguards. Startet nun den W3D-Viewer. Geht auf „File Open“ und wählt dort euer Modell aus, sowie die Animationen des Skeletons, mit welchem ihr euer Modell verlinkt habt. Klickt auf das Plus vor „Hierarchy“, klickt dann dort auf das Plus vor dem Namen eures Modells. Dort könnt ihr euch jetzt angucken ob euer Modell richtig verlinkt ist und welche Stellen ihr ggf. noch einmal neu verlinkt. Dies geht indem ihr einfach wieder „Link to Bone by Name“ wählt und den entsprechenden Bone dort einfügt. Mein Modell ist allerdings richtig verlinkt. Hier ein Screen:

Damit wäre Kapitel 3 abgeschlossen, im nächstem Kapitel seht ihr dann, wie ihr die Textur noch überarbeiten könnt.

4 Das Scinnen

Unser Modell hat im Moment einen Rüstungsskin. Diesen Skin wollen wir an der Brust änder, so dass dort Kleidung ist. Dafür öffnen wir die vorhin konvertierte .tga Datei mit Gimp.

Gimp ist in 3 Fenster aufgeteilt, einmal das Fenster mit Bild dem, dann ein Fenster mit Tools, wie z.B. dem Pinsel und zuletzt den Ebenen Fenster, wo ihr neue Ebenen anlegen könnt. Wir legen zuerst eine neue Ebene an, dafür klickt wir auf diesen Button:

Wählt diese Ebene nun im Ebenen Fenster aus. Wir malen jetzt mit dem Stift:

Dort malen wir jetzt mit einem Grau Ton über die Brust, ich nehme „2f2c2c“ als Farbe. Danach zeichnen wir den Bereich in der Mitte des Brustpanzers mit einem dunklerem Grau, in der Mitte dieses dunkleren Graus noch einmal mit einem dunklerem Grau. Meine Textur sieht nun so aus:

Da der Übergang aber noch nicht gut aussieht verwischen wir diesen nun mit dem Verwischen Tool, diese verbirgt sich hinter diesem Button:

Die Textur sieht danach so aus:

Jetzt wollen wir der Kleidung eine plastischere Wirking geben. Dazu geben wir der Textur zuerst Streifen aus dunkleren und helleren Tönen und verwischen diese danach. Das Tool sieht so aus:

Wählt für die dunkleren Streifen folgende Einstellungen:

Pinsel: Circle Fuzzy(3) Typ: Nachbelichten, Modus: Glanzlichter

Für die helleren Töne wählt ihr die selben Einstellungen, nur beim Typ wählt ihr Abwedeln. Danach verwischt ihr die Streifen wieder mit dem Verwischen Tool, das ganze kann man auf den zwei nächsten Bildern sehen. Beim Verwischen Tool solltet ihr den selben Pinsel wie beim belichten Wählen. Tipp: Wählt ihr unten bei der Größe 400$ könnt ihr die Textur leichter bearbeiten.

Man könnt nun noch beispielsweise den weißen Baum von Gondor auf die Textur bringen. Dazu nimmt man das Bild eines weißen Baumes, löscht das den Rand, sodass man nur den weißen Baum hat. Dann wählt man mit dem Auswähl Tool die Hälfte des Baumes aus und fügt diesen an der Linke Bildschirmhälfte ein. Ungefähr so:

Speichert nun euer Modell wieder als .tga, und zwar unter dem Namen wie ihr ihn beim Modell gegeben habt. Nun kopiert ihr die .tga in den Ordner mit den Animationen und eurem exportiertem Modell. Öffnet jetzt wieder den W3D-Viewer und guckt euch das Modell mit der neuen Textur an. Wenn ihr das Modell als .tga speichert wird eine Meldung kommen, in der ihr „RLE Kompression“ und „Ursprung unten Links“ auswählen konnt. Macht den Haken vor „RLE Kompression“ weg, lasst den Haken vor „Ursprung unten Links“. Dies ist mein Resultat:

Nicht wundern. Der W3D-Viewer zeigt nicht die volle Schärfe der Textur, deshalb sieht man im W3D-Viewer die vorhin angelegten Falten nicht so gut. Im Spiel sieht man sie deutlich besser.

Das wars mit Teil 4. Euer Modell samt Textur ist nun komplett fertig. Wir kommen jetzt zum Programmieren, oder auch dem Coden. Wir werden mit den Basis Codes anfangen, so dass wir unser Modell im Spiel sehen, danach kommen wir zu den Einheit spezifischen Einträgen.

5 Der Basiscode

In diesem Kapitel werden wir einrichten, dass die Einheit im Spiel angezeigt wird.

5.1 Modelle und Texturen ins Spiel bringen

Öffnet eure `mod.big` mit FinalBig. Klickt auf „Edit - Add File“. Wählt nun zuerst eure Textur aus und speichert sie nach dem Prinzip mit den beiden Buchstaben entsprechend. Bei mir wäre das `art\compiledtextures\gu\textbackslashguspeeri.tga`. Das selbe macht ihr mit eurem Modell, bei mir ist dies: `art\w3d\gu\guxspe_skn.w3d`.

Jetzt müssen wir die Dateien noch der `asset.dat` hinzufügen, diese ist sozusagen eine Stapelungsdatei aller Texturen. Öffnet nun den Asset Builder. Wählt den Button „Create asset.dat from files in list only“. Jetzt zieht ihr per Drag'n'Drop eure Textur und Modell in das Fenster. Klickt dann auf „Save...“ und speichert die Datei mit dem Namen `asset.dat`. Importiert diese Datei nun in eure big-Datei unter dem Ordner `asset.dat`, sprich keine Unterordner, nur der Dateiname. Jetzt kann das Spiel euer Modell und eure Textur benutzen.

5.2 Code für die Einheit

Jetzt kommen wir zum eigentlichem Programmieren. Unsere Einheit basiert auf den Turmwachen Gondors, deshalb ist es der einfachste Weg die .ini-Dateien der Turmwachen zu kopieren und später anzupassen. Das Kopieren machen wir nun, das Anpassen in Kapitel 7. Wir öffnen also wieder unsere big-Datei. Auch in den .ini-Dateien gibt es ein System für die Objekte.

`data\ini\object\good- oder evilfaction\units oder structures\volk\*.ini`

Wir müssen also unter `data\ini\object\goodfaction\units\men\gondortowershieldguard.ini` gucken. Wählt diese ini-Datei aus und wählt dann „Edit - Extract“. Speichert die Datei auf dem Desktop. Schließt die .big nun und bennent die .ini auf dem Desktop zu `gondorpikemen.ini` um und öffnet diese Datei.

Wir werden jetzt die Teile verändern, die das Spiel verändert haben muss, damit unser Modell angenommen wird und die Einheit ein neues Objekt wird. Zuerst ändern wir den Objektnamen. Oben steht:

```
Object GondorTowerShieldGuard
```

Dies ändern wir zu:

```
Object GondorPikeman
```

Wir scrollen ein wenig runter und werden diese Zeilen finde:

```
DefaultModelConditionState
        Model = GUTwrGrd_SKN
        Skeleton = GUTwrGrd_SKL
End
ModelConditionState WEAPONSET_PLAYER_UPGRADE
        Model = GUTwrGrd_SKN
        Skeleton = GUTwrGrd_SKL
End
```

Dort können wir nun unser neues Modell angeben. Gebt bei **Model** einfach den Dateinamen eurer .w3d-Datei ohne das .w3d an. Meine Änderung sieht also so aus:

```
DefaultModelConditionState
        Model = guxspe_skn
        Skeleton = GUTwrGrd_SKL
End
ModelConditionState WEAPONSET_PLAYER_UPGRADE
        Model = guxspe_skn
        Skeleton = GUTwrGrd_SKL
End
```

Wir scrollen weiter runter bis wir auf diese Zeilen treffen:

```
ChildObject GondorTowerShieldGuard_Summoned GondorTowerShieldGuard
        IsTrainable = No
        CommandPoints = 0
        EquivalentTo = GondorTowerShieldGuard
        Behavior = LifetimeUpdate ModuleTag_LifetimeUpdate
                // This one does the work, but the one in the horde displays the timer
```

```
                MinLifetime = CREATE_A_HERO_REINFORCEMENT_LIFETIME
                MaxLifetime = CREATE_A_HERO_REINFORCEMENT_LIFETIME
                DeathType = FADED
        End
End
```

Diesen Abschnitt löschen wir. Das war der Basis Code der einzelnen Einheit. Der Rest, wie neuer Name, neues Palantirbild, individuelle Commandpoints, Rüstung und Schanden usw. passen wir in Kapitel 6 und 7 an.

Erster Test
Wir importieren jetzt unsere INI wieder in die BIG. Unter folgendem Pfad:

`data\ini\object\goodfaction\units\men\gondorpikemen.ini`

Starten wir nun unsere Mod über die vorhin angelegte Verknüpfung und seht ob sie startet. Wenn sie startet, dann können wir weitermachen, sollte das Spiel crashen, so überprüft alle Schritte oder meldet euch im Modding-Union Forum. Die Links gibt es am Ende des Tutorials.

5.3 Code für die Horde

Wir kommen jetzt zum Code der Horde, da unsere Einheit als Horde und nicht als einzelne Einheit auftreten soll. Dafür gehen wir nun in diese INI:

`data\ini\object\goodfaction\hordes\men\mehordes.ini`

In dieser INI suchen wir nun nach der **GondorTowerShieldGuardHorde**. Kopiert das ganze Objekt bis ihr das nächste Objekt unten seht. Fügt das Kopierte oben in der INI wieder ein. Wir ändern zunächst den Objekt Namen zu **GondorPikemenHorde**.

Wir scrollen runter bis wir diese Zeile finden:

```
InitialPayload = GondorTowerShieldGuard GOOD_MEN_GIANT_HORDE_SIZE
```

Diese ändern wir zu:

```
InitialPayload = GondorPikeman GOOD_MEN_GIANT_HORDE_SIZE
```

Danach scrollen wir ein wenig runter. Wir werden mehrere Zeilen finden, bei denen **UnitType:GondorTowerShieldGuard** steht. Diese verändern wir zu dieser: **UnitType:GondorPikeman**. Die Zeile, die mit **AlternateFormation** anfängt, löscht ihr. Speichert ab.

5.4 Einheit baubar machen

Jetzt wollen wir die Einheit noch in der Kaserne baubar machen. Dafür müssen wir zuerst einen Baubutton erstellen. Geht dafür in die

`data\ini\commandbutton.ini`

Sucht in der INI nach **TowerShieldGuardHorde**. Ihr werdet diesen Commandbutton finden:

```
CommandButton Command_ConstructGondorTowerShieldGuardHorde
        Command = UNIT_BUILD
        Object = GondorTowerShieldGuardHorde
        Options = NEED_UPGRADE CANCELABLE
        NeededUpgrade = Upgrade_GondorBarracksLevel2
        TextLabel = CONTROLBAR:ConstructGondorShieldGuardHorde
        ButtonImage = BGBarracks_TowerGuard
        ButtonBorderType = BUILD
        DescriptLabel = CONTROLBAR:ToolTipBuildGondorShieldGuardHorde
        Radial = Yes
        InPalantir = Yes
        ShowProductionCount = Yes
End
```

Diesen müssen wir nun ein wenig verändern. Wir kopieren den Button und fügen ihn ganz oben in der INI ein. Bei Objekt schreibt ihr nun eure Horde hin, **GondorPikemenHorde** bei mir. Das **NEEDED_UPGRADE** in den Options, sowie die Zeile darunter löscht ihr, da unsere Einheit schon ab Level 1 der Kaserne gebaut werden soll und nicht erst ab Level 2. Mein Commandbutton sieht nun so aus:

```
CommandButton Command_ConstructGondorPikemenHorde
        Command = UNIT_BUILD
        Object = GondorPikemenHorde
        Options = CANCELABLE
        TextLabel = CONTROLBAR:ConstructGondorShieldGuardHorde
        ButtonImage = BGBarracks_TowerGuard
        ButtonBorderType = BUILD
        DescriptLabel = CONTROLBAR:ToolTipBuildGondorShieldGuardHorde
        Radial = Yes
        InPalantir = Yes
        ShowProductionCount = Yes
End
```

Speichert nun ab und geht in diese INI:

`data\ini\commandset.ini`

Dort suchen wir nach **GondorBarracks**, wir werden die Commandsets der drei Level der Gondorkaserne finden. Wir wollen unsere Einheiten vor den Turmwachen und hinter den Rohan Speeträgern anzeigen lassen. Deshalb fügen wir unsere Button als Nummer 3 ein und ändern die Nummern unter dem neuem Button um +1. So sieht jetzt meine Commandset aus:

```
CommandSet GondorBarracksCommandSet
        1 = Command_ConstructGondorFighterHorde
        2 = Command_ConstructRohanSpearmenHorde
        3 = Command_ConstructGondorPikemenHorde
        4 = Command_ConstructGondorTowerShieldGuardHorde
        5 = Command_PurchaseUpgradeGondorBarracksLevel2
        6 = Command_Sell
End
```

5.5 Erster Test im Spiel

Speichert nun ab und testet ob ihr eure Einheiten im Spiel kaufen könnt. Solltet ihr sie nicht bauen können oder sollte das Spiel crashen, so testet noch Mal alle Schritte oder meldet euch im Modding-Union Forum. Hier sind meine Einheiten im Spiel:

Unsere Einheiten sind jetzt in der Basis im Spiel. Allerdings sind sie haargenau gleich zu den Turmwachen, bis auf, dass sie ein anderes Modell haben. Das wollen wir in den nächsten 2 Kapiteln ändern. Das nächste Kapitel handelt von Palantirbildern und Baubuttons. Dafür werdet ihr wieder Gimp und FinalBig benutzen müssen.

6 Palantirbilder und Buttons

Kommen wir nun zu den Palantirbildern und Buttons. Zuerst müsst ihr euch dieses Packet herunterladen und extrahieren, es enthält Palantir- und Buttonvorlagen für Gimp:

`http://tombombadilmod.to.funpic.de/palaundbutton.rar`

6.1 Button-Bilder erstellen

Öffnet nun euer Modell im W3D-Viewer inklusive der Animationen. Wählt eine Animation aus die euch gut gefällt und die gut für ein Palantirbild ist. Klickt mit dem rechtem Mausrad und bewegt die Maus und näher ran/heraus zu zoomen. Drückt beide Maustasten gleichzeitig um eurem Standpunkt zu verändern. Drückt nun „Druck“ um einen Screenshot zu machen. Öffnet nun Gimp und fügt den Screenshot ein. Geht auf „Bild - Skalieren“ und wählt für den oberen Wert 190 Pixel, der untere wird automatisch angepasst. Klickt auf OK. Dies ist meine Vorlage für die Buttons:

Öffnet nun die Palantirbildvorlage mit Gimp. Wählt das Bild des Soldaten aus und drückt „Strg-A“ und danach „Strg-C“ um das Bild auszuwählen und es zu kopieren. Wählt das Palantirbild aus und fügt das Kopierte als neue Ebene ein. Verschiebt die Ebene so, dass sie auf dem braunen Bereich liegt. Wählt nun das „Nach Farbe auswählen“ Werkzeug aus. Stellt die Schwelle auf 2,0 und klickt auf den grauen Bereich. Danach auf „Bearbeiten - Ausschneiden“. Wählt nun im Ebenenfenster den Ebenenmodus „Überlagern“. Speichert nun diese Datei genau wie die Texturen vorhin ab, hier ist mein Ergebnis:

Skaliert nun das Bild, welches ihr vorhin im W3D-Viewer gemacht habt, auf 64 Pixel. Öffnet nun die Bildatei für die Buttons der Guten. Fügt dort eure Vorlage ein und löscht wieder den grauen Hintergrund. Danach speichert ihr wieder als .tga mit den selben Einstellungen ab. Hier ist mein Ergebnis:

Fügt nun eure beiden Dateien wie die Texturen vorhin in eure `mod.big` ein und macht danach wieder eine `asset.dat` mit all euren Modellen, Texturen und Buttons.

6.2 Button-Bilder eincoden

Geht nun in folgende INI:

`data\ini\mappedimages\aptimages\unitportraits.ini`

Ich werde euch nun zeigen welche Einträge ihr für eure Buttons einfügen musst und diese dann erläutern.

```
MappedImage GondorSpearmenHordeButton
        Texture = gugonspeb.tga
        TextureWidth = 64
        TextureHeight = 64
        Coords = Left:0 Top:0 Right:64 Bottom:64
        Status = NONE
End

MappedImage UPGondorSpearmenHorde
        Texture = gugonspe.tga
        TextureWidth = 256
        TextureHeight = 256
        Coords = Left:0 Top:0 Right:192 Bottom:192
        Status = NONE
End
```

Das erste ist der Baubutton, das zweite das Palantirbild. Ganz oben haben wir die Definition des Namens, ähnlich dem des Objektnamens, diesen Namen müssen wir dann in den anderen INIs angeben, damit das Bild geladen wird.

Texture: Dort muss der Name eures Bildes hin.

TextureWidth und **TextureHeight**: Dies sind die Angaben für die Größe eures Bildes. Unser Button ist 64*64 Pixel groß, deswegen tragen wir dort 64*64 ein.

Coords: Gibt an welcher Teil des Bilder angezeigt werden soll. Die Angaben geben die Grenze an. Also wird in unserem Beispiel die 192*192 Pixel von Oben-Links angezeigt.

Geht nun zuerst in die `Commandbutton.ini` zu eurem Button, fügt dort bei **ButtonImage** den Namen des Baubuttons ein, den ihr hinter dem **MappedImage** stehen habt, mein Button sieht nun so aus:

```
CommandButton Command_ConstructGondorPikemenHorde
        Command = UNIT_BUILD
        Object = GondorPikemenHorde
        Options = CANCELABLE
        TextLabel = CONTROLBAR:ConstructGondorShieldGuardHorde
        ButtonImage = GondorSpearmenHordeButton
        ButtonBorderType = BUILD
        DescriptLabel = CONTROLBAR:ToolTipBuildGondorShieldGuardHorde
        Radial = Yes
        InPalantir = Yes
        ShowProductionCount = Yes
End
```

Geht nun zuerst in die INI der Menhordes. Ändert dort unter **SelectPortrait** euer Palantirbild. Das Gleiche macht ihr dann in der INI der einzelnen Einheit. Startet nun euer Spiel und guckt, ob in der Kaserne der neue Button angezeigt wird und ob eure Einheit das neue Palantirbild benutzt. Sollte dies der Fall sein habt ihr Kapitel 6 abgeschlossen, sollte es ein Problem geben, meldet euch auf der Modding-Union, den Link gibt es in den Anhängen.

7 Die spezifischen Codes

Kommen wir nun zu den spezifischen Codes, zu diesen gehören die Namen, der Schaden den sie machen, ihre Lebenspunkte, die Kosten und Bauzeit, die Kommandopunkte uvm.

Ich werde versuchen möglichst viel euch von diesen Codes zu zeigen und zu erklären. Öffnet zuerst die INI der einzelnen Einheit. Scrollt nun in die Design

Parameter herunter. Das erste was wir verändern sind die Commandpoints. Wir sehen, dass dort **Commandpoints = 5** steht. Die Commandpoints der Horde errechnen sich aus der Anzahl der Einheiten in der Horde mal der Angabe in der INI der einzelnen Unit. In unserer Horde sind 15 Units. Unsere Horde soll 40 Kommandopunkte benutzen, deshalb rechnen wir 40:15 um den Wert einer einzelnen Einheit auszurechnen, simpler Dreisatz den jeder kennen sollte. Das Ergebnis ist 2, Periode 6. Also tippen wir bei den Commandpoints 2.6666666666666 ein.

Scrollen wir ein wenig herunter. Wir werden den Eintrag der Waffen finden, dieser Eintrag nennt sich **Weaponset**. Ändert bei **PrimaryWeapon** den Namen zu einem anderem, den Namen eurer neuen Weapon, ich nehme **GondorSpearmenSpear** - merkt euch aber die vorherige Weapon.

Geht nun in die `data\ini\weapon.ini` und sucht dort nach dem Namen der vorherigen Weapon. Kopiert dieses Weaponset. Bei Damage schreibt ihr eine Zahl hin, dies ist der Schaden den die Einheit macht,der zweite Damage Eintrag gibt den Schaden nach dem Upgraden mit geschmiedeten Klingen an. Dies ist mein Eintrag:

```
Weapon GondorSpearmenSpear

        LeechRangeWeapon = Yes
        AttackRange = 35.0
        MeleeWeapon = Yes
        DelayBetweenShots = GONDOR_TOWERGUARD_DELAYBETWEENSHOTS
        ClipSize = INFINITE_CLIP_SIZE

        PreAttackDelay = GONDOR_TOWERGUARD_PREATTACKDELAY
        ; 400 is sword swing delay time before contact with target.

        PreAttackType = PER_SHOT
        ; Do the delay each time we attack a new target

        FireFX = FX_GondorSwordHit
        FireFlankFX = FX_Flanking

        FiringDuration = GONDOR_TOWERGUARD_FIRINGDURATION
        ; Duration of the sword swing

        DamageNugget
                Damage = 40
                DelayTime = 40
                DamageType = SPECIALIST
                DamageFXType = SWORD_SLASH
                DeathType = NORMAL
                FlankingBonus = 50%
```

```
            ForbiddenUpgradeNames = Upgrade_GondorForgedBlades
        End

        DamageNugget
            Damage = 65
            DelayTime = 40
            DamageType = SPECIALIST
            DamageFXType = SWORD_SLASH
            DeathType = NORMAL
            FlankingBonus = 50%
            RequiredUpgradeNames = Upgrade_GondorForgedBlades
        End

End
```

Gehen wir nun wieder in die INI der Einheit. Wir springen von dem Weaponset zum Armorset. Das ganze ist eigentlich das selbe wie das Weaponset, der Unterschied besteht darin, dass dieser Eintrag die Rüstung angibt. Das Armorset ist der normale Armor Eintrag, der zweite ist der Armor Eintrag für die besseren Rüstungen. Wir verändern die Namen beider. Ich nenne sie GondorSpearmenArmor und GondorSpearmenHeavyArmor. Nun gehen wir in die `data\ini\armor.ini`. Wir kopieren wieder die Einträge der normale Towerguards und verändern sie nach unseren Wünschen. Hierbei ist zu beachten, dass die Prozentzahlen angeben, wie viel Prozent des Schadens eines Gegners abbekommen, nicht der, der geblockt wird. Ich erhöhe überall die Werte um 20%, sie sollen schließlich schwächer als die Turmwachen sein. Dies sind meine Einträge in der armor.ini:

```
Armor GondorSpearmenArmor
        Armor = DEFAULT      100% ; 100% originally 100%
        Armor = SLASH        140% ; originally 200%
        Armor = PIERCE        95% ; 50%  125% RotWK originally 90%
        Armor = SPECIALIST   105% ; originally 100%
        Armor = CAVALRY       25% ; originally 20%
        Armor = CRUSH         10% ; 50%
        Armor = SIEGE        120% ;
        Armor = FLAME        120% ; 30%
        Armor = FROST        120% ;
        Armor = MAGIC        120% ;
        Armor = HERO         220% ;
        Armor = HERO_RANGED  220%
        Armor = STRUCTURAL    55% ; originally 100%
        FlankedPenalty = 75%
End
;-------------------------------------------------------------------------------
Armor GondorSpearmenHeavyArmor
        Armor = DEFAULT       60% ; 40% originally 50%
```

```
        Armor = SLASH          90% ; originally 150%
        Armor = PIERCE         60% ; 20% originally 65% RotWK originally 50%
        Armor = SPECIALIST     50% ; originally 50%
        Armor = CRUSH           5% ; 40%
        Armor = CAVALRY        10% ; originally 10%
        Armor = SIEGE          70% ; 100%
        Armor = FLAME          70% ; 0%
        Armor = FROST          95% ;
        Armor = MAGIC          70% ; 100%
        Armor = HERO          120% ; 50%
        Armor = HERO_RANGED   120% ; 50%
        Armor = STRUCTURAL     20% ;
        FlankedPenalty = 75%
End
```

Zurück in die Einheiten INI. Nach dem Armorset kommen ein paar Einträge, deren Bedeutung ihr durch Übersetzung herausbekommt, die meisten müssen nicht verändert werden, da es Standardeinträge sind.

Der nächste wichtige Eintrag ist „DisplayName“. Fügt nun die lotr.str hinzu, passt auf, dass ihr auch die für euer Spiel nimmt und nicht die falsche. Fügt sie unter `data\lotr.str` ein. Leider kann man die lotr.str nicht in FinalBig bearbeiten, deswegen braucht ihr die lotr.str weiterhin nicht in der BIG um sie zu bearbeiten. Ich ändere nun bei DisplayName meinen Eintrag zu: OBJECT:GondorSpearman.

Öffnet nun die lotr.str. Die lotr.str ist nach folgendem Prinzip aufgebaut: Jeder Block besteht aus 3 Zeilen, die erste Zeile ist die Zeile, die ihr in der INI angebt. Die zweite Zeile, wie sie im Spiel angezeigt wird, die dritte ein End. Wenn ihr im Spiel wollt, dass im Text in den nächste Zeile gesprungen wird, dann schreibt \n dorthin, wo der Text in der nächsten Zeile weitergehen soll. Ich füge also folgenden Eintrag hinzu:

```
OBJECT:GondorSpearman
        "Gondor-Speerträger"
END
```

Speichert die lotr.str und fügt sie wieder eurer Big hinzu.

Die einzelne Einheit hat im Spiel nun ihren eigenen Namen. Nun öffnen wir wieder unsere Einheiten INI. Wir werden den Eintrag CommandSet = finden. Dieser Eintrag gibt an, welche CommandSet die Einheit benutzt, die CommandSet ist sozusagen ein Eintrag, der bestimmt welche Buttons beim auswählen der Einheit erscheinen. Unsere Einheit soll zwar das ForgedBlade Upgrade kaufen könne, aber nicht die schweren Rüstungen. Deshalb müssen wir die wir die CommandSet verändern, ich schreibe dort CommandSet =

GondorPikemanCommandSet hin. Danach gehe ich in die Commandset.ini und füge diesen Block hinzu, er sollte sich durch übersetzen von selbst klären:

```
CommandSet GondorPikemanCommandSet
          1 = Command_ToggleStance
          2 = Command_PurchaseUpgradeGondorForgedBlades
          3 = Command_PurchaseUpgradeGondorBasicTraining
         12 = Command_CaptureBuilding
         13 = Command_AttackMove
         14 = Command_Stop
         16 = Command_SetStanceBattle
         17 = Command_SetStanceAggressive
         18 = Command_SetStanceHoldGround
End
```

Als nächstes kommen die Audio-Parameter, diese geben die Sounds an, die die Einheit von sich gibt, wir werden sie so belassen. Scrollen wir ein wenig herunter, werden wir den ActiveBody Behavior finden. Dort könnt ihr bei „MaxHealth“ die Energie eurer Einheit ändern, Bei „MaxHealthDamaged“ muss die Hälfte der maximalen Lebenspunkte hin. Dies ist mein Behavior:

```
Body = ActiveBody ModuleTag_02
         CheerRadius = EMOTION_CHEER_RADIUS
         MaxHealth = 250
         BurningDeathFX = FX_InfantryBurningFlame
End
```

Das war es mit der INI der Einheit, gehen wir nun wieder in die ini der Horde. Dort haben wir wieder den DisplayName, dieses verändert ihr genau wie oben und fügt hinten ein Horde ein. Die Zeile unter DisplayName ist eine Beschreibung der Einheit, die dritte eine Beschreibung wogegen die Einheit stark ist. Da unsere Einheit auf der Basis der Turmwachen basiert können wir die dritte Zeile so belassen, die zweite Zeile verändern wir. Hier sind meine Codes aus der INI und der lotr.str:

```
OBJECT:HordeGondorSpearmen
         "Gondor-Speerträger Bataillon"
END

CONTROLBAR:LW_Unit_GondorSpearmenHorde
         "Speerträger-Bataillon aus Gondor"
END

DisplayName = OBJECT:HordeGondorSpearmen
DisplayNameStrategic = CONTROLBAR:LW_Unit_GondorSpearmenHorde
DescriptionStrategic = CONTROLBAR:LW_ToolTip_GondorTowerGuardHorde
```

Scrollen wir weiter runter. Wir werden die Einträge BuildCost und BuildTime, diese könnt ihr natürlich so anpassen wie ihr wollt, ich habe bei BuildCost = 200 hingeschrieben, bei BuildTime = 16, Buildtime ist in Sekunden angegeben. Ein wenig weiter unten findet ihr wieder die Commandpoints, schreibt dort nun 40 hin. Bei CommandSet wieder die selbe Commandset, die ihr schon in der Einheitenini genommen habt. Das war's in der Horden-INI.

7.0.1 Die Einheit von der KI baubar machen

Unsere Einheit soll natürlich auch von der KI gebaut werden, geht dafür in folgende INI:

`data\ini\default\skirmishaidata.ini`

Scrollt bis zu dieser Zeile herunter:

```
ArmyDefinition MenOfTheWestArmy
```

Nun noch ein wenig weiter herunter bis ihr zu den Einträgen mit den Einheiten kommt. Dort fügen wir jetzt einen Eintrag mit unserer neuen Einheit hinzu.

Unsere Einheit soll häufiger gebaut werden, als die Turmwachen, deswegen nehmen wir höhere Prozentzahlen als bei den Turmwachen. Im Lategame sollen die Einheiten allerdings nicht so häufig wie die Turmwachen gebaut werden, deswegen machen wir es dort niedriger, dies ist mein Eintrag:

```
ArmyMemberDefinition GondorPikemenHorde_Member
        Unit = GondorPikemenHorde ;pikeman
        PercentageOfArmyPhase1 = 35.0
        PercentageOfArmyPhase2 = 25.0
        PercentageOfArmyPhase3 = 15.0
End
```

7.0.2 Buttonbeschreibungen anpassen

Zum Schluss müssen wir noch die Beschreibungen beim Button verändern.

Geht dafür in die commandbutton.ini und verändert bei eurem neuem Button das TextLabel und das DescriptLabel. Fügt sich genauso wie vorhin in die lotr.str hinzu, hier sind meine Codes:

```
CONTROLBAR:ConstructGondorSpearmenHorde
        "Gondor-Speerträger"
END
```

```
CONTROLBAR:ToolTipBuildGondorSpearmenHorde
        "Stärken: Kavallerie"
END

CommandButton Command_ConstructGondorPikemenHorde
        Command = UNIT_BUILD
        Object = GondorPikemenHorde
        Options = CANCELABLE
        TextLabel = CONTROLBAR:ConstructGondorSpearmenHorde
        ButtonImage = GondorSpearmenHordeButton
        ButtonBorderType = BUILD
        DescriptLabel = CONTROLBAR:ToolTipBuildSpearmenHorde
        Radial = Yes
        InPalantir = Yes
        ShowProductionCount = Yes
End
```

8 Schlussworte

Das war es, eure Einheit ist nun komplett fertig, startet nun das Spiel und testet sie. Solltet ihr dieses Tutorial erfolgreich gemeistert haben seit ihr nun auf dem bestem Weg ein guter Modder zu werden, es steht am Ende nur das Verstehen. Ich möchte euch am Ende noch einige Weisheiten in Sachen Modding geben:

1. Seit nicht frustriert, wenn das Game crasht, meistens sind es nur Tippfehler und da ihr im Crash immer eine Zeilen und Ini Angabe habt könnt ihr diese schnell fixen.
2. Es gilt: Learning by doing
3. Außerdem gilt auch: Probieren geht über studieren
4. Fast nichts ist in Sachen Modding unmöglich, man muss es nur probieren
5. Fragt, wenn ihr nicht weiterkommt!

Nun möchte ich euch noch den Downloadlink meiner BIG geben, damit ihr sie mit eurer INI vergleichen könnt, falls ihr Probleme habt:

`http://modding-union.com/updates/tbfn/gondorsoldaten.rar`

Ich hoffe ihr habt alles verstanden und ihr werdet ein paar der unten zu sehenden Links besuchen.

9 Danksagungen und Links

Zuerst möchte ich Turgon, Herr von Gondolin, sowie Herunor danken, die beiden sind die beiden, die mich zum Modding gebracht haben und ohne die ich jetzt nicht ein Modder mit diesem Wissen wäre.

- Ealendril der Dunkle, Simbyte und das Edain Mod Team, für die Modding-Union.
- Arkani und Crafty, fürs gute Zusammenarbeiten
- Adamin, für die Hilfe in Sachen Skinnung und Buttons
- Eugen, für die tolle Zusammenfassung des Kleidungsskinnens mit Gimp
- uvm.!

Links

- www.modding-union.com Die Website der größten Deutschen SuM Modding Community
- www.the3rdage.net Die größte Englische SuM Modding Community
- www.tbfn.de.vu Die Website meiner Modifikationen
- www.sum-fanpage.de Die größte deutsche SuM Fanpage
- www.totalwar-zone.de Eine tolle Community zu den Total-War Titeln

Ich hoffe ihr habt in diesem Tutorial einiges gelernt und werdet dieses auch behalten. Ich habe versucht euch das Modding der Schlacht um Mittelerde Spiele auf eine einfache Weise zu erklären und näher zu bringen, ich hoffe dies ist mir gelungen

Solltet ihr irgendwelche Probleme mit dem Tutorial, den Schlacht um Mittelerde Spielen oder auch ganz anderen Dingen haben, so könnt ihr euch immer auf der Modding-Union melden und wir werden versuchen euch dort möglichst gut zu helfen.

Mit freundlichen Grüßen,

Turin Turumbar

auch Earendil genannt.