Titel: FX Erstellung, Einführung ins Terrain

Schwierigkeit: Grundkenntnisse

Typ: Aufgaben, Aller Anfang hat einen Anfang (hä?)

Programm: Final BIG

Alien aka Infiltrator

Ich grüße euch Coder-Neulinge und ein herzliches Willkommen zum Tutorial

In diesem Tutorial behandeln wir die Grundlegende Struktur einer Mod-Big, also was ihr beim der Erstellung beachten müsst. Außerdem werde ich euch das sog. „Inlcude-System“ erklären, welches für deutlich mehr Übersicht und Struktur in euren Dateien sorgt.

Voraussetzung für dieses Tutorial ist lediglich das ihr FinalBig installiert habt. Sollte dies nicht der Fall sein, scheut euch nicht hier rein zu sehen ;-)

Modding Union – Tools (es macht keinen Unterschied ob ihr die Deutsche oder Englische Version wählt)

Inhaltsverzeichnis

DER ANFANG (WO ALLES BEGINNT)

## Grundlegende Einrichtung

Solltet Ihr FinalBig noch nie gestartet haben tut dies nun.
Jetzt könnt ihr es auch schon wieder schließen^^

Warum haben Wir das getan?
Ganz einfach: Jetzt können wir in den Schlacht Um Mittelerde-Ordner gehen (bzw. in den Aufstieg des Hexenkönigs-Ordner; je nachdem welche Version ihr modden wollt), und mit
Rechtsklick (auf eine freie Stelle) – Neu – FinalBigDocument
Eine neue Big-Datei erstellen.

Jetzt müssen wir der Big-Datei einen Namen geben.

## Big-Terminologie

Es gibt beim benennen der Big-Dateien gewisse Regeln die es zu beachten gilt.
Das Spiel lädt die Big-Dateien nämlich beim Spielstart in einer festen Reihenfolge.
Und zwar Alphabetisch, und das beginnend beim letzten Buchstaben (zuerst Z, dann A).
Sonderzeichen werden sogar nach dem Alphabetischen Big's geladen.
Aus unserem AddOn Ordner würde die Reihenfolge der Bigs so aussehen:
-Window.big
-W3D.big
-Textures4.big
-Textures3.big
-Textures2.big
-Textures1.big
-Textures0.big
-Terrain.big
-Shaders.big
-Music.big
-Maps.big
-lotrsec.big
-Libraries.big
-ini.big
-data2.big
-Data1.big
-Bases.big
-Audio.big
-AmbientStreams.big
-_patch201.big (diese Big existiert erst nach dem Patchen auf die neuste Spielversion)

_patch201.big wird nach ini.big geladen, und enthält einige Dateien die in ini.big enthalten sind, das Spiel nimmt dann die der _patch201.big. Stellt euch einfach vor es wird zuerst der Inhalt der ini.big in einen Ordner gelegt, und dann der der _patch201.big darüber geschrieben.

Warum ist das wichtig?
Wir wollen schließlich das unsere Dateien die sind, die zuletzt geladen werden.
Das heißt wir müssten für eine Mod für das Addon, unsere Mod-Big mit folgenden Zeichen beginnen lassen.

1. Zeichen: _
2. Zeichen: ein Buchstabe der im Alphabet vor p kommt.

Also z.B.: _mod.big oder _alien.big

Solltet ihr einmal eine Mod veröffentlicht haben könnt ihr aufgrund dieser Eigenschaften ganz einfach Update-Bigs veröffentlichen, indem ihr diese einfach Alphabetisch geringer macht als die Ursprüngliche Mod-Big.
z.B. durch Nummerierung: _a999.big
_a998.big (dies wäre die Update-Big)

<u>Zusätzlicher Hinweis für AddOn-Modder:</u>
Solltet ihr für das AddOn modden, dann ist es wichtig zu wissen, das zuerst alle Bigs von Schlacht um Mittelerde geladen werden, und dann die des AddOn's. Dies erklärt auch warum ihr einige Dateien in den Bigs des AddOn nicht finden werdet.

Ich empfehle euch an dieser Stelle, die ini-Big des SuM Ordners mit Edit – Extract all in einen Ordner zu entpacken (dies kann etwas dauern), und danach die ini.big des AdH Ordners über den selben Ordner mit Edit Extract All zu entpacken.
Dadurch bekommt ihr ALLE ini-Dateien wie sie im AddOn verwendet werden in einer übersichtlichen Ordner-Struktur, dies erspart euch (besonders am Anfang) den ständigen Wechsel zwischen den beiden ini.big-Dateien.

Erläuterung; Das Big-Format

Big? Nie gehört? Aber eigentlich ganz einfach!
Big-Dateien kann man sich am ehesten wie einen Ordner bzw. ein Verzeichnis vorstellen, nur eben in einer Datei gebündelt. Also ganz ähnlich als würdet ihr einen Ordner mit WinRar oder WinZip verpacken.
Innerhalb einer Big werden Dateinamen immer nach folgenden Schema angegeben:
data\ini\object\datei.ini

Also ziemlich genauso wie wir das auch von dem Windows Explorer oder einem Browser gewohnt sind. In dem Beispiel oben befindet sich also die Datei in einem Ordner „object“. Der (Big-Interne) Ordner „object“ befindet sich dabei in dem Ordner „ini“ und dieser wiederum in dem Ordner „data“.
Würden wir eine Big mit nur dieser Datei entpacken (über Edit – Extract All), würden wir als Ergebnis genau diese Ordner-Struktur bekommen.

Bevor wir jetzt weitermachen lest euch zuerst einmal aufmerksam den Abschnitt "Die Include-Zeile" durch. Ändert dabei am besten nichts an einer Mod-Big.

Die Include-zeile

Diese Zeile bindend die angegebene Datei in die Datei ein, in der sich die Zeile befindet. Und zwar genau an der Stelle an der sie sich befindet, dieses Detail ist sehr wichtig und kann bei der Fehlerbehebung sehr wichtig sein.

Eine Include-Zeile ist immer so aufgebaut:
#include "RelativerDateipfad\Dateiname"

Relativer Dateipfad?
Jap... relativer Dateipfad^^
Was ist damit gemeint?
Es ist so, das sich die Datei in der ihr die Include Zeile schreibt bereits in einem Ordner befindet, nehmen wir hier z.B. die Datei
data\ini\fxlist.ini

Ihr wollt jetzt für das Include-System eine Datei einbinden deren vollständiger Pfad
data\ini\include\fxlist_include.ini
lautet.

Der relative Dateipfad wäre von der Position aus gesehen, an der sich die Ursprungsdatei befindet, bis zur einzubindenden Datei. Es müssen also quasi alle Ordner angegeben die die Usprungsdatei öffnen müsste bis sie zur Einzubindenden gelangt hin geschrieben werden.
In unserem Beispiel wäre das dann:
#include "include\fxlist_include.ini"

Die Datei muss nur den include Ordner öffnen und dort die fxlist_include.ini, weil sie sich ja bereits in data\ini\ befindet.
Weiteres Beispiel:
data\ini\object\neutralobjects.ini
Möchte die Datei
data\ini\object\standardactions\neutral\BurnActions.ini
(diese Datei gibt es im ungemoddeten Spiel nicht, der Name dient nur zur Veranschaulichung)
einbinden, dann lautet die Include-Zeile:
#include "standardactions\neutral\BurnActions.ini"

Jetzt kann es natürlich auch sein das man eine Datei die sich in einem übergeordneten Ordner, oder sogar in einem Ganz anderem Ordnerzweig, befindet. Dazu kann man "..\" verwenden, denn dies geht von der Datei aus einen Ordner nach oben.
Will
data\ini\object\neutralobjects.ini
folgende Datei inkludieren:
data\ini\customAbilities.ini
Dann würde unsere Include-Zeile so aussehen:
#include "..\costumAbilities.ini"

"..\" kann natürlich auch aneinandergereiht werden, wenn eine Datei zwei Ordner höher liegt dementsprechend "..\..\Dateiname".

Eigener Code in einer Modbig...

Nun ja... jetzt geht es darum das Spiel zu ändern.
Wollt ihr z.B. einen neuen Button schreiben, müsstet ihr das normalerweise in der Commandbutton.ini tun (FullName: data\ini\commandbutton.ini)
Um dies innerhalb der Mod-Big zu tun (um die ini.big nicht zu ändern) können wir einfach diese Datei extrahieren und in die Modbig ziehen (bzw. die bereits extrahierte commandbutton.ini die ihr am Anfang des Tutorials mit "Edit - Extract All" erzeugt habt).
Die ini muss dabei in eurer ini den selben FullNamen wie das Original haben.
data\ini\commandbutton.ini
Jetzt würde eure commandbutton.ini geladen, die sich in der Mod-Big befindet, denn die ModBig überspreichert ja den Inhalt der ini.big.
Das heißt ihr könntet theoretisch alle Änderungen und neuen Einträge in die Commandbutton.ini schreiben, die sich in eurer Mod-Big befindet, das wollen wir allerdings nicht.

Wir möchten das unsere Änderungen seperiert und 100% getrennt in Include-Dateien enthalten sind, dies behandelt der nächste Abschnitt.

...nur innerhalb der Include-Dateien

Endlich ist es soweit, der Kernpunkt unseres Dateisystems das wir innerhalb unserer Big befolgen sollten/wollen.
Dies dient keiner höheren Macht oder einem besonderem Kapitalistischem Vorteil, nein, es spart einfach nur Zeit, oder auch nicht...
Zum einen dauert es am Anfang wesentlich länger, da man für gut die Hälfte aller ini-Dateien eine Include einbauen muss (immer zu dem Zeitpunkt an dem man sie das erste mal verwendet).
Zum anderen hat man dann eine strikte Trennung von Mod und Originalen Dateien, dies ermöglicht ein schnelles finden von Codes und zahlt sich spätestens nach zwei Monaten deutlich in einem Entwicklungszeit-Vorteil aus.
Große Mod-Projekte wie z.B. die Edain-Mod wären ohne dieses System warscheinlich unmöglich für neue Teammember zu modden, da diese sich erst sehr lange einarbeiten müssten um zu wissen wo sich welche Codes befinden.

Das Include-System folgt ein paar simplen Regeln:
1.Für alle ini-Datein werden Include-Dateien erstellt in die der Mod-Inhalt kommt
2.Orignale Einträge werden NICHT verändert, stattdessen wird eine Kopie mit geändertem Namen in der Include erstellt und neu mit den Nutzenden Einträgen verknüpft
3.Modinhalte dürfen nicht in orginale Spiele-inis geschrieben werden.
4.Die Include-Dateien sollten überwiegend den selben Pfad besitzen, empfehlenswert wäre hier:
data\ini\INCLUDE\include-datei
- wobei INCLUDE mit einer beliebigen Modspezifischen (und TEAMINTERN FESTGELEGTEN Bezeichnung) tauschbar ist. Sollte eure Mod zum Beispiel "Doofe Mod" heißen, würde sich \doofe\ anbieten.

Viele Modifikationen, sowie ich persönlich für meine Mini-Mods, ignorieren obrigen Regeln für "Object" Elemente, also alle Einheiten, Gebäude und Projectile Einträge, da der damit entstehende Mehraufwand sich später nicht durch deutlich mehr Übersicht lohnt. Daher eine Empfehlung meinerseits:
Haltet euch an die 4 Regeln, mit Ausnahme der Object-inis.
Dies ist der perfekte Mittelweg zwischen Mehraufwand und erhöhter Übersichtlichkeit und Struktur.

Teaminterne Regelungen:
Teamintern sollten zusätzlich Bezeichnungsregeln aufgestellt werden, wie z.B. Kopien eines Original-Eintrags am Namen immer mit dem Zusatz "_mod" zu versehen, am besten orientiert man sich dabei auch an EA Games Vorlagen.
Zum Beispiel das beschworene Einheiten immer den Zusatz "_summoned" im Namen tragen, und das im ObjectName die Volkszugehörigkeit beinhaltet ist (Beispiel: "ElvenWarrior" oder "RohanEomer").

Allgemeine Tipps zum Mod-Big-Bau (löl)
-Legt am Anfang am besten gleich mal eine
data\ini\object\projectile.ini
an, in die ihr alle Projectiles schreibt die ihr im laufe der Zeit so macht. Dies hat sich bereits bei anderen Mods als sehr Übersichtliche schnelle Lösung erwiesen (hier ein netter Verweis auf die Edain-Mod).
-Object-inis können nur in dem Verzeichnis data\ini\object\ erstellt werden
-es ist ratsam die Object-inis in dem selben Schema wie EA Games zu halten:
data\ini\object\good oder evilfaction\structure oder unit\volk\objectname.ini
-Es empfiehlt sich Textdateien (inis) und Art, also Modelle, Bilder und Sounds getrennt voneinander in zumindest zwei verschiedenen Big-Dateien zu verstauen, dies sorgt für mehr Übersichtlichkeit.

Schlusswort

Ich gebe keine Garantie für meine Erklärungen, ich habe das alles aus dem Kopf geschrieben und daher auch nur meine Meinung zu Papier (PDF :-p ) gebracht, die Vorgehensweise hat sich bereits bewährt und ist durchaus sinnvoll. Bei Fragen könnt ihr euch an mich wenden, bin für Fragen jederzeit zu haben ;-)

Grüße
Alien aka Infiltrator

Big-Fachsimpler und Erfinder des Begriffs „Big-Terminologie“

stylize.de.tc

(ACHTUNG: ES BESTEHT KEINERLEI GARANTIE AUF OBRIGEN INHALT, DER VERFASSER STAND ZUM ZEITPUNKT DES VERFASSEN UNTER PERMANENTER ALTERNATIVE-METAL-BESCHALLUNG)