Fahrzeugassets in TPF2

Grundsätzlich ist das Erstellen von Dekorationsobjekten/Assets in TPF 1 und 2 einfach: aus einer .con-Datei werden .mdls angesprochen, und man kann diese dann im Spiel per Mausklick verteilen.

Beginnen wir mit der .con. Diese gehört ins Verzeichnis res\construction\asset und sieht in einer einfachen Form so aus:

Code

```
function                                                                    data()
return                                                                          {

                                                                        name         =

    },

                                                        categories             =             {



        {

                                                            values     =     {     _("Version

        },
    },



                                                                                        if
models[#result.models+1]=vehicle/asset/Model1.mdl",transf[illegible],
                    end

                                                                                        if
models[#result.models+1]=vehicle/asset/Model2.mdl",transf[illegible],
                    end



    end
}

end
```

Alles anzeigen

Vieles davon ist selbsterklärend, wenn man schon mal ein wenig programmiert hat.

Unter "name = " definiert man, wie das Modell heißt; unter "description = " kann eine kurze Beschreibung rein. Bei "availability" kann zwischen die Schweifklammern "startYear = " und "endYear = " stehen, so kann man definieren, zwischen welchen Jahren das Objekt zur Verfügung stehen soll.

"Category" definiert, wie das Modell im Assetmenü im Spiel eingeordnet wird. Hier kann man beliebige eigene Kategorien festlegen. Wenn man ein kleines Vorschaubildchen festlegen möchte, dann muß es genauso heißen wie die Kategorie. Für "Kategorie" wäre das Vorschaubild also "Kategorie.tga"; es gehört ins Verzeichnis res\textures\ui\construction\categories. Dann sollte das Spiel das Bild automatisch finden und anzeigen.

"Order" gibt an, an welcher Stelle in dieser Kategorie die Assets aufgeführt sein sollen. Bei zwei .cons mit gleicher Nummer wird alphabetisch geordnet.

"SkipCollision": dürfen Zombies und Fahrzeuge das Objekt durchqueren?

"autoRemovable": darf das Spiel das Objekt selbst wegräumen und überbauen?

Der Block unter "params = " definiert ein Auswahlmenü.

"key=" legt einen internen Variablennamen fest, in dem der ausgewählte Menüpunkt gespeichert wird; hier sollte man einen einmaligen Namen wählen wie (Modersteller)_(Modname)_1 - bei meiner 727 heißt die Variable also dh106_B727_1.

"name = " definiert eine Titelzeile des Menüs.

"values = " legt die Namen der einzelnen Menüpunkte fest. Diese sind frei zu wählen, müssen also nicht so heißen wie ein Mod. _("Essiggurke"), _("Euphrat"), _("Lebertran") gehen genauso wie _("Holzroller") usw.

Um also dem Spiel zu sagen, welche Auswahl welche .mdl bewirken soll, kommt der Block mit den diversen "if"-Abfragen. Die Menüpunkte sind einfach der Reihe nach durchgezählt; wenn man also im obigen Beispiel auf "Essiggurke" klickt, wird in der festgelegten Variablen der Wert 0 gespeichert. Bei Euphrat 1, bei Lebertran 2 und so weiter.

if params.dh106_B727_1 == 0 then

result.models[#result.models+1] = {id = "vehicle/asset/Modell 1.mdl",transf = {1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, }, }

end

...heißt also nichts anderes als: Wenn auf "Essiggurke" geklickt wurde, dann kommt das Modell "vehicle/asset/Modell 1.mdl". Für eine Erklärung der Transformationsmatrix dahinter gibt es hier einen schönen Artikel.

Diese .con ist leider nur bedingt fehlertolerant.Wichtig ist, daß es für jeden dieser Menüpunkte auch eine "Auflösung" gibt, daß also jede Option auch zu einer .mdl führt. Wenn das nicht so sein sollte, wenn also ein Menüpunkt das Spiel ins Nirwana führt, dann kommt genau das in Form eines CTDs.

Im Menü soll natürlich auch ein Vorschaubild auftauchen. Hier bietet sich an, eines der Bilder aus ui/models20/... zu nehmen. Kopiert es einfach ins Verzeichnis res/textures/ui/construction/asset und benennt es genau wie die .con - das Vorschaubild für eine "Beispiel.con" muß also "Beispiel.tga" heißen.

Möglich ist, die .mdls einfach so zu nehmen, wie man sie als Fahrzeuge gebaut hat. Dabei gibt es aber mehrere Probleme:

- alle nur selektiv angezeigten Meshes sind immer angezeigt. Das gilt für Lichter, drehende Propeller und anderes, was man nicht immer bei einem stehenden Fahrzeug sehen möchte.

- alle Bereiche, die über die Cblend-Funktion variabel gestrichen werden können, sind schwarz.

Deswegen empfehle ich, eine separate .mdl für die Dekorationsobjekte zu erzeugen und ihr ein eigenes, einfacheres Material ohne Cblend und Alterung zu verpassen.

Aus der Asset-.mdl fliegen folgende Teile raus:

- alle Meshes und Gruppen, die nicht immer angezeigt werden. Drehende Propeller und auch viele Lichter gehören in diese Kategorie.

- alle Ansprachen für nicht immer angezeigte Teile unter "config = ".

- der ganze Block "transportVehicle=" am Ende. Das bewirkt, daß die Asset-.mdl nicht im Kaufmenü als Fahrzeug angeboten wird.

Viel Mühe spart man sich, wenn man schon von vorneherein alle nicht immer angezeigten Meshes ganz hinten in der .mdl eingetragen hat, dann kann man sie einfach rausschneiden und muß nur die Aufrufe

rausschießen. Sonst muß man gelegentlich auch die Numerierung in der Liste von Hand wieder korrigieren.

Als Material für Dekorationsobjekte empfehle ich "PHYSICAL_NRML_MAP", mit der habe ich immer gute Erfahrungen gemacht. "PHYSICAL_TRANSPARENT_NRML_MAP_CBLEND_DIRT_RUST" funktioniert auch, wenn man das Objekt gerne gealtert hinstellen möchte. Man kann einfach ein neues solches Material im Editor anlegen und entsprechend speichern; in der Asset-.mdl ist es schnell über die Suchen- und-Ersetzen-Funktion des Editors ausgetauscht.