Egoshooter

pat

08.06.2004

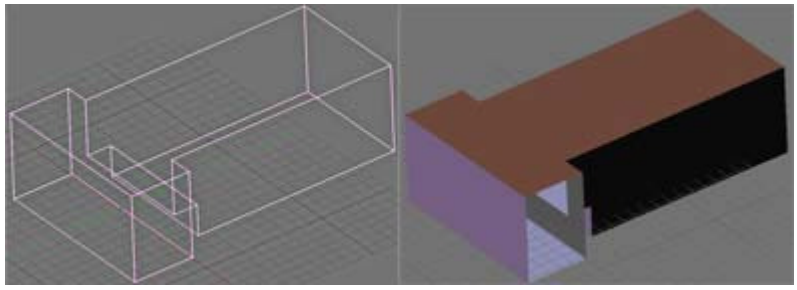
Einführung

Blender 2.0 war die erste Version von Blender, die die GameEngine eingebaut hatte. Nach und nach kamen weitere Funktionen dazu, um dem Benutzer das Leben und Spielen leichter zu machen. Das ging so weiter, bis mit 2.25 der letzte GameBlender veröffentlicht wurde und die Firma NAN Pleite ging. Seither benutzen spielfreudige Blenderfans den etwas veralteten Blender 2.25, um weiter ihre Ideen umzusetzen. Bis heute...

Mit Blender 2.33 wurde die GameEngine wieder eingeführt, zuerst auf dem Stand von 2.25, aber weitere Funktionen sind zu erwarten. Um diese neuen Möglichkeiten sinnvoll nutzen zu können, ist ein grundlegendes Verständnis der GameEngine notwendig. Um alle Funktionen voll auszuschöpfen, werden wir auch etwas mit der Programmiersprache Python arbeiten müssen, aber keine Angst, ihr müsst jetzt nicht programmieren lernen :)

Wir fangen mit einem kleinen Level an, in dem wir herumlaufen können, dann kommt unser Computerspieler an die Reihe, und ganz am Ende noch etwas, worauf wir schießen können. Und schon geht's los.

Level



Hier solltet ihr eigentlich eure eigenen Ideen verwirklichen. Da ich nicht vorhabe, gegnerische Spieler zu erzeugen, werde ich einfach eine Art Schießstand bauen.
Zuerst braucht ihr natürlich das Mesh, die Geometrie eures Levels. Ich denke, dass kann ich euch selbst überlassen. Ich hab hier einfach mal das Konzept meines Levels abgebildet, falls ihr genauso wenig eigene Ideen habt wie ich :) Dann braucht euer Level natürlich Texturen, davon geistern im Internet genug kostenlose rum, und wer ne Digicam hat, kann sich selber gute Texturen erzeugen.

Mit ALT Z wechselt ihr in die texturierte Ansicht, falls noch nicht geschehen, und gebt wie gewohnt Texturen. Achtet aber darauf, dass die Flächennormale in die richtige Richtung zeigt. Im Gegensatz zum normalen Rendern wird in der GameEngine nur eine Seite der Fläche gerendert, welche das ist könnt ihr im EditMode sehen, wenn "Draw Normals" (EditButtons, Mesh Tools 1) aktiviert ist. Der blaue Strich kommt dann auf der Seite der Fläche raus, die gerendert wird. In der Texturansicht seht ihr das auch, wenn eine Fläche nicht in die richtige Richtung zeigt. Dann müsst ihr die Fläche im EditMode auswählen und drückt W >> Flip Normals.

Blender kann die Normalen auch automatisch berechnen (STRG [SHIFT] N im EditMode, ohne Shift werden die Normalen nach außen gedreht, mit Shift nach innen), das Ergebnis muss aber manchmal manuell korrigiert werden.

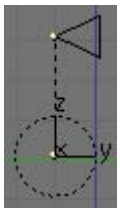
Spieler

Bevor ihr hier anfangt, kontrolliert erst einmal, ob bei den WorldButtons (F8) eine Physikengine eingestellt ist. Der Button links neben dem "Grav" Button darf nicht "None" anzeigen. Im Moment gibt es nur die Sumo-Engine, also wechselt auf "Sumo".

In der Oberansicht erzeugt ihr zuerst ein Empty, dann gleich noch eine Kamera in der Vorderansicht. Die Kamera bewegt ihr etwas nach oben.

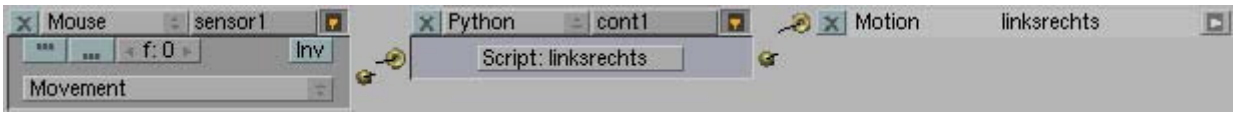
Danach wählt ihr wieder das Empty, wechselt mit F4 zu den EchtzeitButtons und klickt auf "Actor" und dann auf den neuen "Dynamic" Button. Jetzt wird das Empty von der Engine beachtet. Um das gleich zu testen, klickt ihr in der Sensor-Gruppe (links) auf "Add" und macht aus dem "Always" ein "Keyboard". Dann klickt ihr in das "Key"-Feld und drückt eine Taste. Als nächstes erzeugt ihr einen "Controller" und zieht mit der Maus eine Verbindung zwischen den gelben Kreisen bei Sensor und Controller. Ihr erzeugt auch noch einen Actuator, den ihr mit dem Controller verbindet. Diese Verbindungen könnt ihr übrigens löschen, indem ihr mit der Maus darüberfahrt, bis sie weiß werden, und dann X drückt. Im zweiten "linV"-Feld des Aktuators, das heißt in Y-Richtung, gebt ihr dann den Wert 5 ein und aktiviert den kleinen L-Button rechts daneben, damit wir unabhängig von der Drehung immer nach vorne laufen.

Um das Ganze auch zu sehen, erzeugt ihr einen Würfel an der Position des Emptys, wählt den Würfel und als zweites zusätzlich das Empty und drückt STRG P, damit der Würfel dem Empty folgt. Wenn ihr jetzt in der Seitenansicht P drückt, könnt ihr euren Spieler mit der Taste bewegen und seht, wie sich der Würfel mitbewegt.



Da das aber ein Egoshooter werden soll, könnt ihr den Würfel wieder löschen. Bindet jetzt die Kamera an das Empty (STRG P), dabei muss das Empty wieder als letztes ausgewählt werden. Jetzt könnt ihr euch in der Kameraansicht durch das Level bewegen, aber immer nur geradeaus ist langweilig, oder?! Deshalb kommt jetzt das erste Pythonscript. Das wird uns das Empty drehen, wenn wir die Maus bewegen, so wie wir es aus anderen Spielen kennen.

Zuerst brauchen wir für unser Empty einen "Mouse" Sensor, der auf "Movement" reagiert. Den verbindet ihr mit einem Python Controller und einem normalen "Motion" Actuator. Gebt dem Actuator einen eindeutigen Namen, z.B. "linksrechts".



Außerdem kriegt unser Empty eine bestimmte Eigenschaft zur leichteren Einstellung: klickt auf "ADD property" und ändert das Namensfeld auf "move". Als Wert für diese "Float" Eigenschaft stellt ihr 180 ein. Mit der Maus über einen 3D-Fenster drückt ihr SHIFT F11, um ein Textfenster zu öffnen. Da hinein kopiert ihr dann diesen Code: (unter windoze mit STRG SHIFT V)

```
#####
from GameLogic import *
from Rasterizer import *

Cont = getCurrentController()
Own = Cont.getOwner()
Sens = Cont.getSensors()
Sensor = Sens[0]

Height = getWindowHeight()/2
Width = getWindowWidth()/2

#aktuelle mausposition holen
Xpos = Sensor.getXPosition()

#actuator holen
linksrechts = Cont.getActuator("linksrechts")

#mausposition relativ zur bildschirmmitte messen
XDiff = Xpos - Width

#den actuator einstellen
linksrechts.setDRot(0,0,(XDiff/Own.move),1)

#und einmal kurz aktivieren
addActiveActuator(linksrechts,1)
addActiveActuator(linksrechts,0)
```

```
#dann die maus wieder in die bildschirmmitte setzen
setMousePosition(Width,Height)
#####
```

Diesem Text gebt ihr dann einen Namen, ich habe wieder "linksrechts" genommen, und tragt diesen Namen im Python Controller unter Script ein (Bild oben). Wenn ihr eurem Actuator einen anderen Namen gegeben habt, müsst ihr das Script ändern, da wo steht "getActuator". Achtung wenn ihr irgendwelche Namen ändert: Blender und Python achten auf Groß- und Kleinschreibung!

Jetzt solltet ihr in der Kameraansicht sehen, dass ihr euch seitlich drehen könnt. Die Drehgeschwindigkeit könnt ihr mit der "move" Eigenschaft des Emtys einstellen. Da wir die Kamera auch nach oben oder unten kippen wollen, machen wir das Ganze jetzt für die Kamera: Mouse-Movement Sensor, Pythoncontroller, Actuator "hochrunter" und "move" Eigenschaft. Dazu gehört dieses Script:

```
#####
from GameLogic import *
from Rasterizer import *

Cont = getCurrentController()
Own = Cont.getOwner()
Sens = Cont.getSensors()
Sensor = Sens[0]

Height = getWindowHeight()/2
Width = getWindowWidth()/2

Ypos = Sensor.getYPosition()

hochrunter = Cont.getActuator("hochrunter")

YDiff = Ypos - Height
hochrunter.setDRot((YDiff/Own.move),0,0,1)

addActiveActuator(hochrunter,1)
addActiveActuator(hochrunter,0)

setMousePosition(Width,Height)
#####
```

Jetzt könnt ihr euch in jede Richtung drehen, geradeaus laufen und dabei an die Decke schauen, wenn ihr wollt.

Ziele

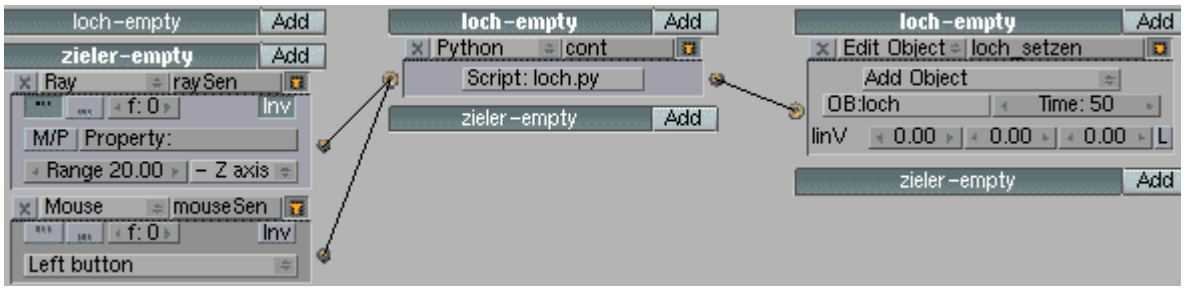
Zum Zielen braucht man einen Zieler, also erzeugt ihr eine einzelne Fläche und bewegt sie direkt vor die Kamera. Macht sie nicht zu groß; je nachdem welche Art von Zieler ihr verwendet, soll der ja nicht das ganze Bild ausfüllen. Dann gebt ihr eine Textur von einem Zielerkreuz, am Besten weiß auf schwarzem Hintergrund. Ihr könnt den Zieler später mit Vertexfarben einfärben. Außerdem bindet ihr den Zieler mit STRG P an die Kamera.

Wenn ihr im FaceSelect Modus seid, geht ihr zu den EditButtons, in der Gruppe "Texture face" klickt ihr dann auf Add, um den Modus für die Textur umzustellen. Falls ihr eine Textur mit Alphakanal verwendet, könnt ihr auch Alpha wählen. Damit wird dann der (schwarze) Hintergrund ausgeblendet, nur der weiße Vordergrund bleibt sichtbar. Wenn ihr mein Bild verwendet (Format PNG), könnt ihr Alpha aktivieren, ich hab einen Alphakanal drin. Und weil man weiß auf weiß schlecht sieht, hab ich hier einen Rahmen eingebaut :)

Damit wir sehen, wo wir getroffen haben, brauchen wir ... Einschusslöcher! Und so geht's: Wir brauchen eine weitere kleine Fläche mit der Textur des Lochs (wieder die gleichen Einstellungen für den Texturmodus, dazu noch Halo), die ihr in der Oberansicht (wichtig!) erzeugt und im EditMode so dreht, dass die Normale in die negative X-Richtung zeigt (Oberansicht: nach links). Außerdem noch ein Empty, das ihr von der Kamera aus etwas hinter dem Zielerobjekt platziert und auch an die Kamera bindet. Für dieses Empty erzeugt ihr einen "Ray" Sensor mit großer Reichweite (Range); 20 oder so sollte genug sein, kann aber von euerm Level abhängen. Im Feld

rechts daneben stellt ihr die Achse des Emptys ein, die von Kamera und Zieler weg zeigt. (Bei mir -Z, ich hab das Empty in der Vorderansicht erzeugt.) Außerdem aktiviert ihr den Button "Inv" rechts oben und den Button links oben mit den drei Punkten. Der Ray Sensor heißt "raySen".

Dann braucht ihr noch ein Empty (es wird langsam richtig voll :) mit einem "Edit Object" Actuator namens "loch_setzen". Im OB: Feld tragt ihr den Namen eures Lochobjekts ein. Wenn ihr bei Time einen Wert größer als null einstellt, verschwindet das Loch nach einer Weile wieder. Das war unser Loch-Empty.



Unser Zieler-Empty von gerade eben bekommt jetzt noch einen Mouse-Left button Sensor "mouseSen". Dann wählt ihr zusätzlich zum Zieler-Empty noch das Loch-Empty aus, erzeugt einen Python Controller für das Loch-Empty, der mit dem Ray Sensor, dem Mouse Sensor und dem EditObject Actuator verbunden wird. Und hier kommt das Script "loch":

```
#####
import GameLogic

con = GameLogic.getCurrentController()

mouse = con.getSensor("mouseSen")
if mouse.isPositive():
  own = con.getOwner()

  raySen = con.getSensor("raySen")
  pos = raySen.getHitPosition()

  own.setPosition(pos)

  addHole = con.getActuator("loch_setzen")
  GameLogic.addActiveActuator(addHole, 1)
  GameLogic.addActiveActuator(addHole, 0)
#####
```

Ihr müsst noch den Scriptnamen im Python Controller eintragen und solltet dann die Einschusslöcher sehen.

Und weil wir immer noch kein Ziel haben, könnt ihr einfach eine Zielscheibe aufhängen.