Blender - Lektion S1
Game Engine

Blender V 2.77 – Skript V 1.0

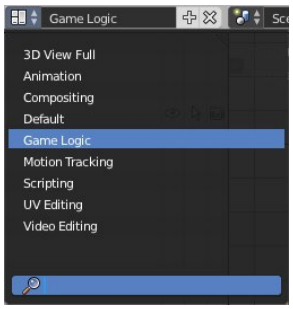
Sie haben eine Premiere vor sich: Die erste Blender Lektion der CoGra-Gruppe, geschrieben von einem der Schüler. Sie sind bei Johannes in guten Händen – ich wünsche Genuss und Erkenntnis beim Lesen.

Natürlich darf der Inhalt dieser Lektion nur zum Erstellen von pädagogisch wertvollen Simulationen verwendet werden. Aber in CoGra würde ja ohnehin niemand auf die Idee kommen, etwas so ganz und gar sinnloses wie ein Computerspiel zu erstellen oder sogar zu spielen …

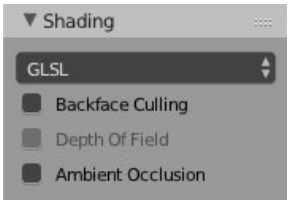
Autor und Konzept: Johannes Klatt, Franz-Ludwig-Gymnasium Bamberg, Computergrafikgruppe (CoGra-FLG)
Layout und Bilder: Uwe Gleiß
Kontakt über: cogra-flg@web.de
Dieses Werk ist lizenziert unter einer Creative Commons
Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz.

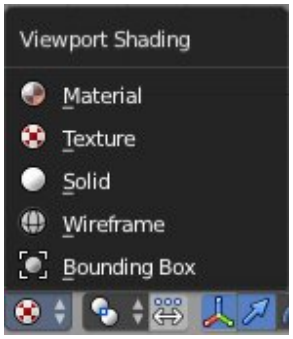

Bevor es wirklich los geht

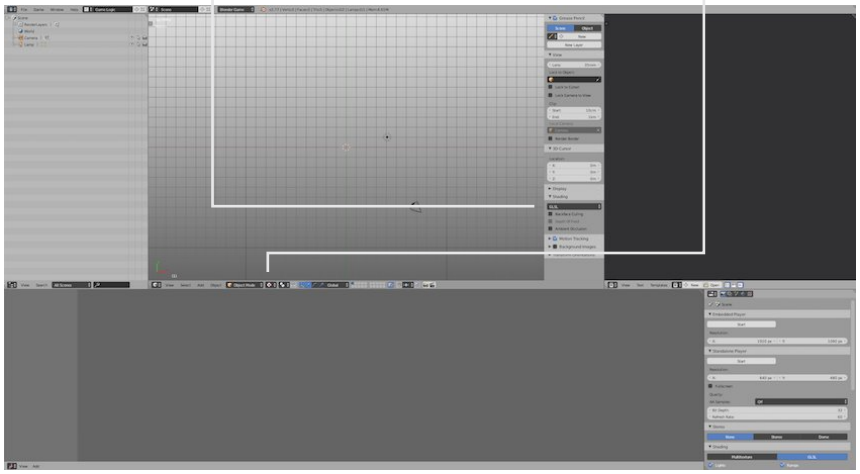
Vorbereitungen



Wechseln Sie zuerst in das *Game-Logic-Layout*. Schalten Sie im Info–Frame die Renderengine auf *Blender Game* und im rechten Panel des 3D–Frames das Shading auf *GLSL*.[1] Letzteres ermöglicht die Verwendung von Bildtexturen im Spiel, während ansonsten gar keine benutzt werden können. Damit die Texturen im Spiel auch angezeigt werden, muss aber zusätzlich der 3D–Frame auf *Texture* geschaltet werden.







[1] Für die, die es nicht wissen: Dieses Panel lässt sich öffnen, indem man mit der Maus über dem 3D–Frame auf N drückt.

(Fast) alles ist anders

Sie werden noch merken, dass sehr vieles, was man aus Blender kennt, in der Game Engine auf einmal anders funktioniert:

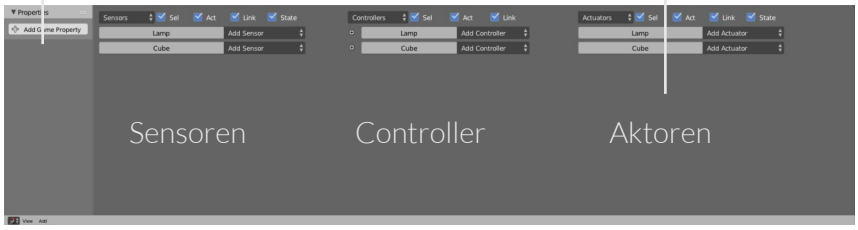
- Es gibt neue Rendereinstellungen, die aber zum Glück weitgehend selbsterklärend sind.
- Eine Art verbesserte Editoransicht ersetzt das wirklich gerenderte Bild.
- Es können mehrere Szenen übereinandergelegt werden, dafür gibt es keine Renderlayer.
- Filter ersetzen Compositing.
- Partikel und Constraints funktionieren nicht mehr (auch wenn es passende Nodes als Ersatz gibt).
- Es lassen sich auf einmal Dinge wie Parents „animieren".
- Die Physiksimulation funktioniert anders.
- …

Weil ich mich sehr viel mit der Game Engine beschäftigt habe, komme ich sogar immer wieder in Situationen, wo ich zwar weiß, wie man etwas in der Game engine schafft, aber keine Ahnung habe, wie es im „richtigen" Blender funktioniert.

Nodes und Noodles ... schon wieder

Diesmal geht es nicht um Compositing, sondern um den *Logic Editor*. Dieser befindet sich unter dem 3D-Frame und ist in zwei Bereiche aufgeteilt:

Links die *Game-Property-Liste* (würde jetzt noch zu weit führen) und rechts der eigentliche *Logic Node Editor*, in dem das Programm geschrieben wird.



Logic–Editor, Würfel und Lampe sind ausgewählt

Wie Sie wahrscheinlich schon bemerkt haben, ist dieser Bereich seinerseits in drei Spalten eingeteilt: Eine für *Sensoren*, eine für *Controller* und eine für *Aktoren*. Wählen Sie nun mehrere Objekte, beispielsweise den Standardwürfel und die Lampe aus: Für jedes ausgewählte Objekt erscheint in jeder der drei Spalten eine Zeile mit dem Namen des Objekts und einem Knopf *Add Sensor/Controller/Actuator* daneben. Mit diesen Tasten können Sie dem jeweiligen Objekt - Groooooße Überraschung - neue Logic Nodes hinzufügen.

Auf der linken Seite und damit am Anfang der Schaltung stehen ein oder mehrere Sensoren. Sobald diese ausgelöst werden, senden Sie ein Signal an den angeschlossenen Controller. Dieser führt eine logische Operation mit den Signalen durch und sendet, wenn seine Bedingung erfüllt wurde, ein Signal an die angeschlossenen Aktoren, was diese auslöst.[1]

Diese Struktur ist zwingend. Versucht man, einen Sensor direkt mit einem Aktor zu verbinden, wird automatisch ein And-Controller dazwischengeschaltet. Es ist auch unmöglich, mehrere Controller hintereinanderzuschalten. Dafür können die Verbindungen auch zwischen den Nodes von verschiedenen Objekten gelegt werden.

[1] Python tanzt in dieser Beziehung aus der Reihe, ist für diese Lektion aber definitiv zu kompliziert.

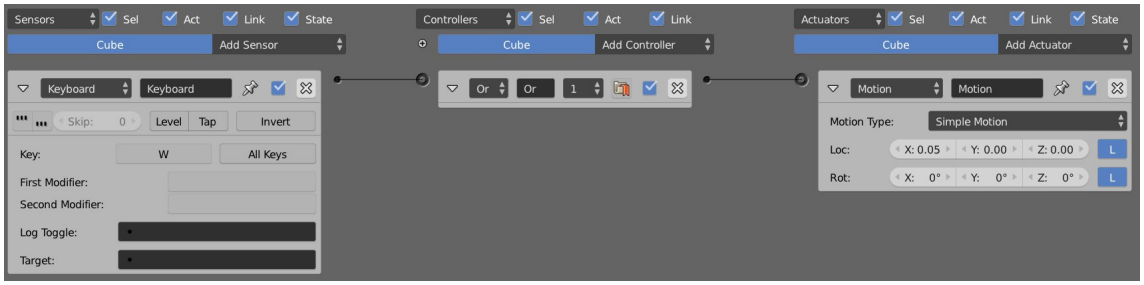
Eine erste Schaltung

Fügen Sie dem Würfel folgende Nodes hinzu: Einen *Keyboard-Sensor*, einen *Or-Controller* und einen *Motion-Actuator* und verbinden Sie diese miteinander.

Stellen Sie im Keyboard-Sensor eine Taste ein, die Ihnen gefällt (einmal auf das Feld neben *Key* klicken und dann die Taste drücken) und legen Sie im Motion Actuator eine Bewegung um 0,05 in X-Richtung fest, indem Sie die Zahlen neben *Loc* entsprechend verändern.

Sobald Sie das Spiel starten, (einfach P drücken, wenn die Maus sich über dem 3D-Frame befindet; Esc beendet das Spiel[1]) wird dieses Programm 60 mal pro Sekunde ausgeführt.

Diese Zeiteinheit von 1/60 Sekunde heißt *Logic Tick* und wird in der Game Engine als Ersatz für die Framerate, die bei Computerspielen schwanken kann, verwendet.



In jedem Logic Tick wird vom Keyboard-Sensor geprüft, ob die Taste gedrückt ist oder nicht. Ist sie gedrückt, sendet er ein Signal an den Or-Controller.

Dieser leitet ein Signal weiter, wenn er von einem beliebigen der angeschlossenen Sensoren eines erhält. Da in diesem Fall nur ein Sensor angeschlossen ist, sorgt das für eine direkte Weiterleitung zum Aktor.

Motion lässt das Objekt, das den Node besitzt, eine Bewegung ausführen, in diesem Fall um 0,05 in X-Richtung. Da das Ganze 60 mal pro Sekunde passiert, erreicht der Würfel trotzdem eine Geschwindigkeit von 3 pro Sekunde.

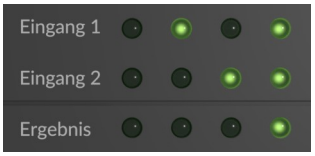
[1] Wenn ihnen das nicht gefällt, können Sie in den Rendereinstelkungen unter System einen anderen Exit Key einstellen.

Controller

Schließen sie einen zweiten Keyboard-Sensor für eine andere Taste an den Or-Controller an und starten Sie das Spiel. Wie gesagt leitet dieser Controller das Signal weiter, wenn er von einem beliebigen der angeschlossenen Sensoren ein Signal empfängt. Es ist also egal, welche der beiden Tasten gedrückt wird, beide lösen die Bewegung aus.

Darstellung des Verhaltens des Or–Controllers

Aber es gibt noch andere Controller (deren Funktionsweise Sie mit dieser Schaltung auch einfach ausprobieren können):



- *And* leitet das Signal nur weiter, wenn er von allen angeschlossenen Sensoren eines empfängt.

- *Xor* macht das Gleiche wie Or, allerdings löst er nicht aus, wenn mehrere angeschlossene Sensoren ein Signal an ihn senden

- *Nand* sendet ein Signal weiter, wenn nicht alle (also auch gar keiner) der angeschlossenen Sensoren ein Signal senden.

- *Nor* sendet ein Signal, wenn er von keinem der angeschlossenen Sensoren eines empfängt.

- *Xnor* sendet ein Signal, wenn er von keinem oder mehreren der angeschlossenen Sensoren eines empfängt. Ist nur ein angeschlossener Sensor aktiv bleibt Xnor inaktiv.

Sensoren

Hier gibt es die mit Abstand größte Auswahl, aber es sind auch recht viele Nodes dabei, die man nur sehr selten bis nie braucht.

- *Keyboard* kennen Sie Schon. Allerdings kann man nicht nur mit Key eine Taste einstellen, sondern mit *First*, bzw. *Second Modifier* andere Tasten hinzufügen um eine Tastenkombination wie Ctrl+Alt+U zu verwenden. Außerdem kann mit *Any Key* eingestellt werden, dass der Sensor auf alle Tasten anspricht.

- *Joystick* steht allgemein für Spielecontroller. Es kann entweder ein Knopf auf dem Gerät oder eine Auslenkung einer Achse in eine bestimmte Richtung eingestellt werden. Erkannt wird aber nur, dass die Achse ausgelenkt wird, wie stark ist unwichtig. Außerdem braucht man zum Einstellen die *Axis-*, bzw. *Button Number* und um die herauszufinden, hilft nur Ausprobieren.

- *Mouse* erkennt, je nach Einstellung, ob eine bestimmte Maustaste gedrückt ist, wobei Mausrad hoch und Mausrad runter als Knöpfe gesehen werden. *Mouse over Any* schließlich erkennt, wenn der Mauszeiger über ein Objekt bewegt wird. Die Reaktion kann zusätzlich auf ein zu treffendes Material beschränkt werden.

- *Collision* erkennt Kollisionen des Objekts, das den Node besitzt, mit anderen Objekten. Auch hier kann feiner unterteilt werden: Wenn Sie auf *M/P* drücken, können Sie ein Material einstellen. Dann lösen nur noch Kollisionen mit Vertices, auf denen das gewählte Material liegt, den Sensor aus

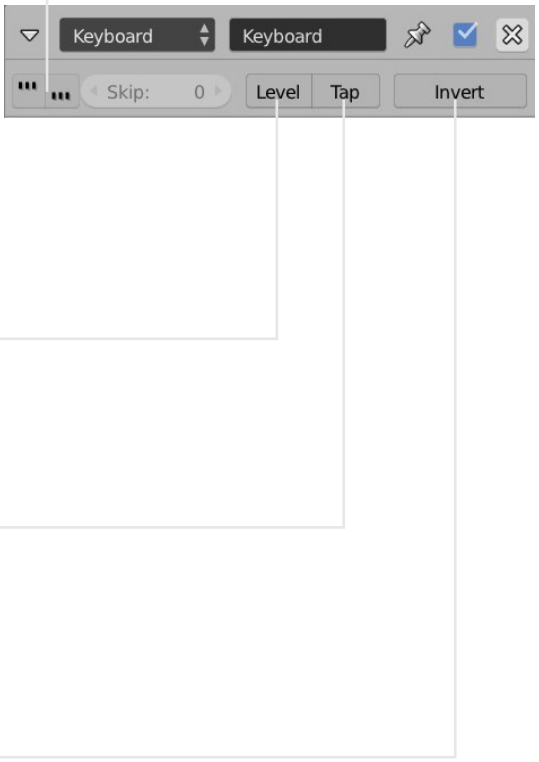
- *Random* sendet auf Basis eines einstellbaren Seeds zu zufälligen Zeitpunkten ein Signal. Zwei solche Nodes mit dem gleichen Seed feuern also stets gleichzeitig.

- *Actuator* sendet, sobald der eingestellte Aktor ausgelöst wird.

- *Delay* zählt erst die unter *Delay* eingestellte Anzahl an Logic Ticks herunter und sendet dann für die unter *Duration* eingestellte Zeit in Logic Ticks ein Signal. Ist *Repeat* aktiviert, wiederholt sich das.

- *Always* sendet permanent ein Signal. Der einfachste Node, den Sie finden werden.

Jeder Sensor besitzt einen Header mit Einstellungen, die sich alle Sensoren teilen:



- Die Knöpfe ''' (*true Level triggering*), ,,, (*false Level triggering*) und *Skip* sind für den Pulse Mode zuständig. Ist true Level triggering aktiviert, sendet der Sensor, wenn er normalerweise durchgehend senden würde, jeweils einen Logic Tick dauernde Signale mit so vielen Logic Ticks dazwischen, wie bei Skip eingestellt sind. False Level triggering macht das Gleiche, allerdings dann, wenn der Sensor normalerweise nicht senden würde. Ist Skip auf 0 gestellt, wird jeweils ein durchgängiges Signal statt der beschriebenen Folge gesendet.

- Ist *Level* aktiviert, sendet der Sensor nur noch ein einen Logic Tick langes Signal, wenn er normalerweise anfangen oder aufhören würde zu senden.

- *Tap* lässt den Sensor nur noch einmalige Signale senden. Würde der Sensor also normalerweise ein dauerhaftes Signal senden, sendet er nun stattdessen nur diesen Puls, sobald er aktiviert wird.

- Mit *Invert* wird das vom Sensor gesendete Signal umgedreht; Wenn der Sensor normalerweise nicht Senden würde, tut er es nun und umgekehrt.

Die Einstellungen werden auch in dieser Reihenfolge abgearbeitet, d.h. erst Pulse Mode, dann Level, dann Tap und zuletzt Invert.

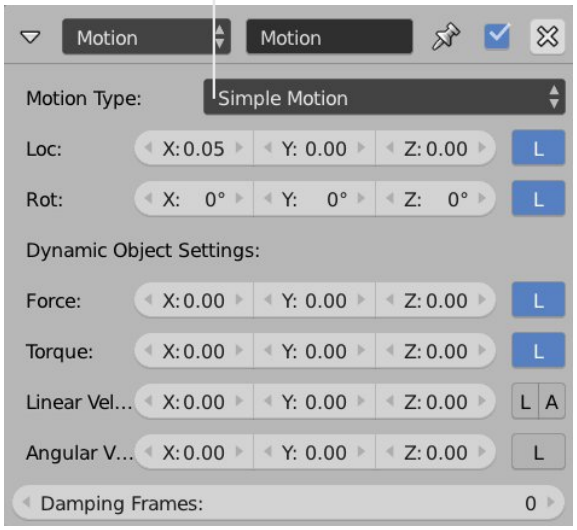
Aktoren

Das sind die einzigen Nodes, die auch wirklich etwas bewirken. Wir beschränken uns hier zunächst auf eine kleine Auswahl:

Motion

sollten Sie noch besser auf *Simple Motion* eingestellt lassen. Je nachdem, welchen Physiktyp das Objekt hat (kann in den Physikeinstellungen des Objekts festgelegt werden), können Dinge wie Ort, Geschwindigkeit, Kraft, Drehung etc. eingegeben werden.

Die Abkürzungen sollten aus dem restlichen Programm bekannt sein. Mit dem Klick auf das *L* werden alle Bewegungen im lokalen Koordinatensystem des Objekts statt im globalen ausgeführt. Das kann vor allem bei sich drehenden Dingen sinnvoll sein. Ein an gleicher Stelle befindliches *A* kann aktiviert werden, um die Wirkung durch die Steuerung aufzuaddieren statt auf einen bestimmten Wert zu setzen.

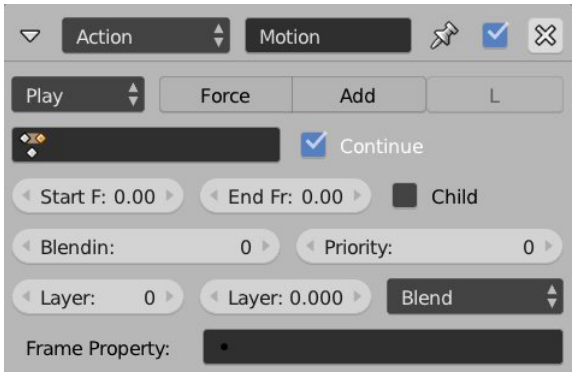


Motion Aktor bei einem Objekt mit Physics Type Dynamic

Action

Action lässt das Objekt eine Animation abspielen.[1] Neben der Einstellung, welche Frames und welche Action verwendet werden findet sich hier vor allem auch ein kleines Menü, in dem man einen Modus für den Node auswählen kann:

Play spielt die Animation normal ab, *Pingpong* spielt sie beim ersten Empfang eines Signals vorwärts, beim zweiten rückwärts, beim dritten wieder vorwärts usw. ab, *Flipper* spielt sie auch bei Empfang eines Signals vorwärts ab und lässt sie zurücklaufen, wenn er kein Signal mehr empfängt. *Loop Start* und *Loop End* spielen die Animation, solange sie ein Signal erhalten, immer wieder durch, Loop Start vorwärts und Loop end rückwärts.



Game

Game führt einfache das Spiel betreffende Operationen durch: Start Game from file lädt eine neue .blend-Datei und führt das dort eingebaute Programm aus, restart Game startet das Spiel neu und end Game - na was wohl? In Verbindung mit zwei passenden Python-Skripten ließen sich mit diesem Node auch Spielstände laden und speichern.

[1] Mehr zu Actions gibt es voraussichtlich in Lektion A4.

Edit Object

Edit Object ist mehr oder weniger eine Sammlung für Objekte betreffende Operationen, die sonst nirgendwo hinpassen.

Mit *Add Object* kann ein zweites Objekt am Ursprung des Objekts, das den Node besitzt, mit einer einstellbaren Startgeschwindigkeit eingefügt werden.

End Object zerstört das Objekt, dem der Node zugeordnet ist, und zwar unwiderbringlich, solange das Spiel danach weiterläuft.

Unter *Dynamics* kann man die Masse des Objekts verändern, mit dis-/enable rigid body Kollisionen an- und abschalten und mit suspend/free dynamics das Objekt komplett einfrieren, bzw. auftauen.

Track to richtet das Objekt mit dem Aktor auf ein Zielobjekt aus – sehr beliebt für Kameras, die dem Geschehen folgen sollen.

Mit *Replace Mesh* kann das Mesh und damit das Aussehen des Objekts ausgetauscht werden. Wenn man diesen Node oft verwendet, muss man also eventuell sogar seine Meshes benennen …

Wo Benennung aber wirklich ein Muss ist, sind die Nodes. Denn niemand hat Lust, wenn er eine ganz bestimmte Bewegung schneller machen will, alle Nodes von Motion1 bis Motion18 auf- und wieder zuzuklappen.

Übungsaufgaben

1. Verpassen Sie dem Standardwürfel eine Schaltung, mit der er sich mit den Pfeiltasten auf einer Ebene in alle Richtungen bewegen lässt

2. Bauen Sie ein kleines Labyrinth, indem Sie Objekte auf einer Ebene verteilen. Platzieren Sie darin einen Zylinder mit Physics Type Character oder Dynamic. Dieser Zylinder soll sich über vorwärts- und rückwärtslaufen und nach links, bzw. rechts drehen steuern lassen.

3. Fügen Sie dem Labyrinth ein Ziel hinzu und denken Sie sich etwas aus, das passiert, wenn der Zylinder es erreicht.

4. Bauen Sie eine Schwierigkeit, z.B. ein Hindernis, das Stacheln ausfährt oder eine in regelmäßigen Abständen schießende Kanone ein. Natürlich soll das Spiel beendet werden, wenn man mit einer davon zusammenstößt.

5. Tun Sie etwas Kreatives!