Unity3d

Serious Gaming

Paolo De Chirico

depa1014@hs-karlsruhe.de

Inhaltsverzeichnis

Serious Gaming

Was ist Serious Gaming

Serious Games sind Videospiele in denen komplexe Sachverhalte simuliert werden, um Wissen über eben diese Sachverhalte zu vermitteln. Am ehesten kann man Serious Games noch mit E-Learning-Einheiten vergleichen. Sie besitzen jedoch einen viel höheren Grad an Interaktivität, Flexibilität und Detail, als gängige E-Learning-Formate.

Bei der Erstellung von Serious Games stehen den Programmierern alle Features zur Verfügung, die auch im restlichen Sektor der Videospielindustrie existieren. Da diese Industrie mittlerweile auf einer Stufe mit der Filmindustrie steht, sind die Möglichkeiten in dieser Hinsicht enorm.

Beispiele

Clinispace

Clinispace ist ein Serious Game, in dem verschiedene Szenarien im Bereich Medizin simuliert werden. Hierbei werden verschiedene Krankheitsbilder simuliert, um die Diagnosefähigkeit der User zu überprüfen und zu steigern.

Explore Mars

Bei Explore Mars handelt es sich hingegen um eine andere Art von Serious Game. Hier kann der Spieler selbst die simulierte Oberfläche des Mars mit Hilfe eines kleinen Mars-Rovers erforschen.

Was ist Unity3d

Das Programm

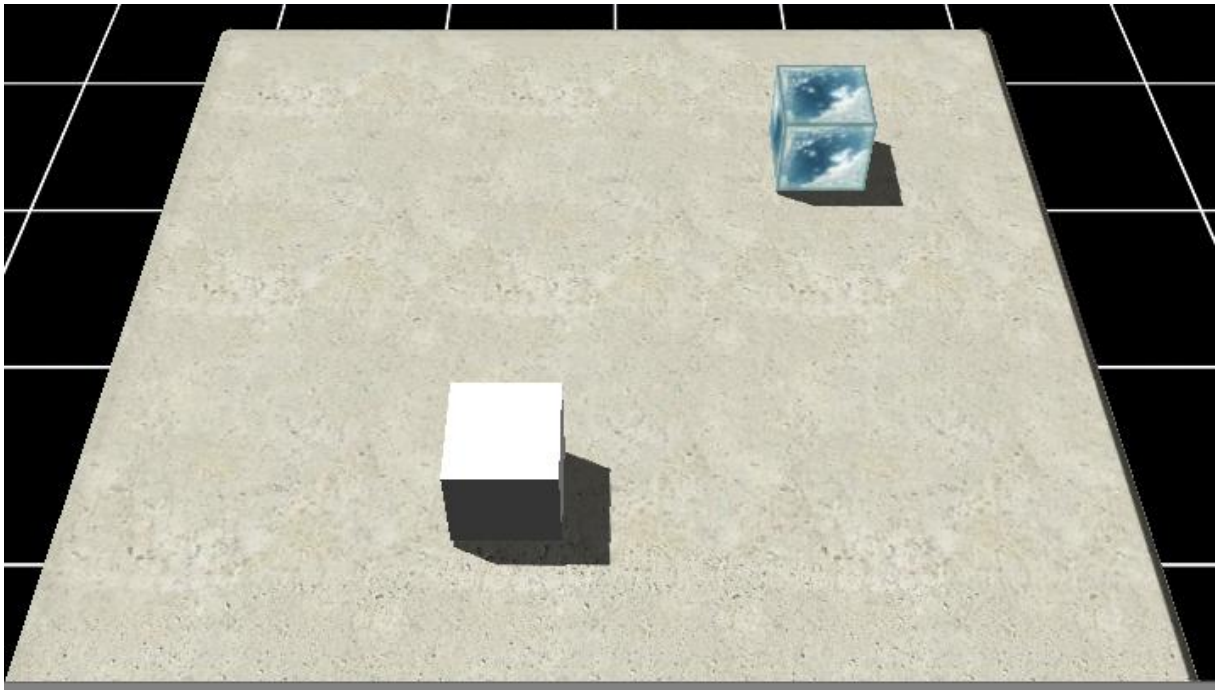
Unity3d ist eine Engine, auf der Videospiele programmiert werden können. Sie zeichnet sich vor allem durch die Möglichkeit aus, auf allen gängigen Plattformen lauffähige Games erstellen zu können. Browser, Smartphones, PC, Mac und alle aktuellen Gaming-Konsolen werden unterstützt.

In der Industrie hat Unity3d außerdem den Ruf, sehr intuitiv und leicht bedienbar zu sein. Aus diesem Grund haben wir uns im Rahmen dieses Projekts dafür entschieden, die Engine selbst auszutesten und eine kleine Beispielszene zu basteln und anhand dieser den Arbeitsaufwand auszuwerten der bei der Erstellung eben dieser Beispielszene anfällt.

Arbeiten mit der Oberfläche von Unity3D

Projektziele / Unsere Beispielszene

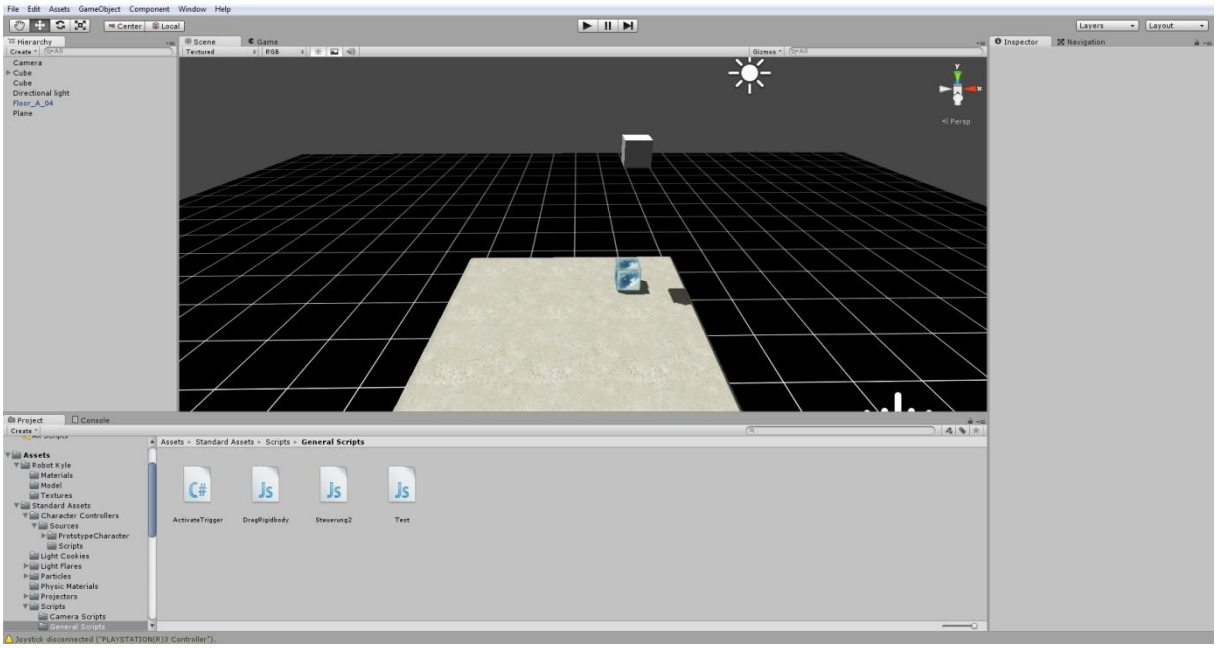
Bei unserem Projekt ging es uns darum, die Nutzbarkeit der Unity3D Engine im Hinblick auf die schnelle Erstellung von Serious Games zu untersuchen. Nachdem wir uns in einige Tutorials eingearbeitet haben, entschieden wir uns für eine einfache Beispielszene.



In dieser Szene soll es dem Spieler möglich sein, einen Würfel zu steuern, und eine simple Interaktion mit einem anderen Element der Umgebung ausführen zu können. Wir benutzen hierfür zwei unterschiedlich gefärbte Würfel und eine Lichtquelle. Wir benutzten anfänglich Boden und Wände für unsere Beispielszene, entschieden uns am Ende jedoch aus Übersichtsgründen lediglich für den Boden.

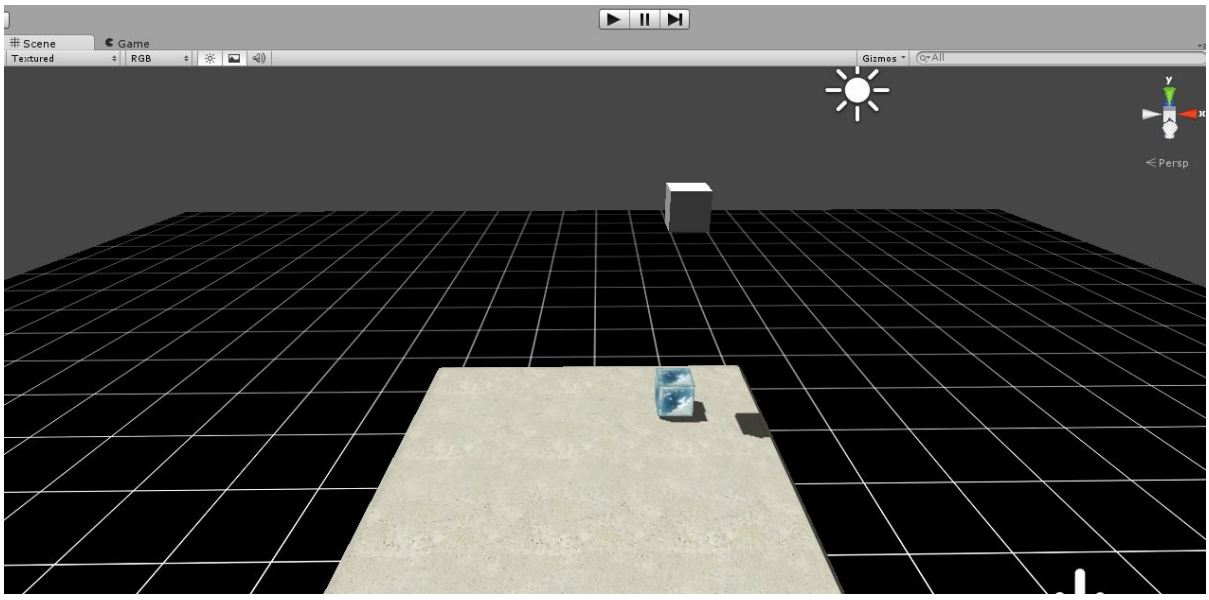
Das Ziel der Szene ist es, die Würfel miteinander in Kontakt zu bringen, und dadurch das Licht auszuschalten. Der Spieler steuert den weißen Würfel.

Übersicht der Benutzeroberfläche



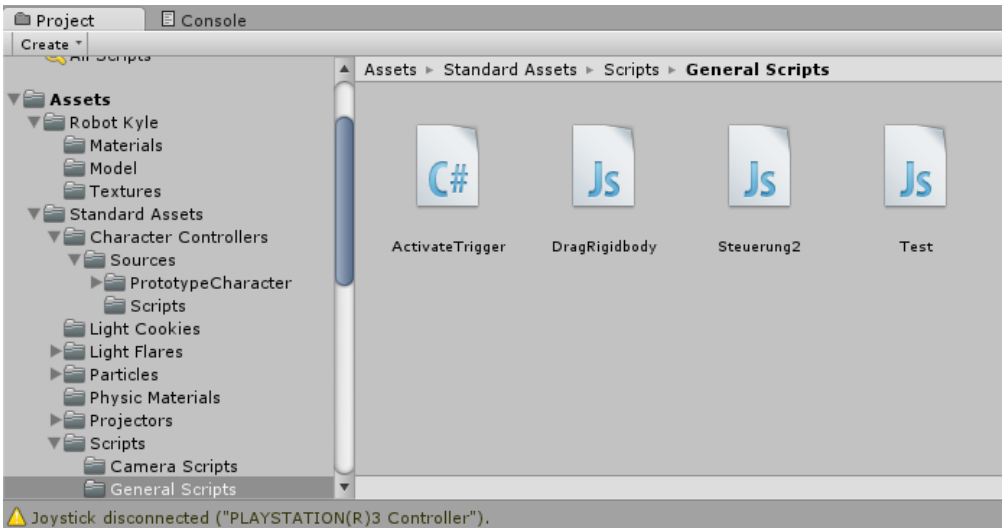
Der Editor ist aufgebaut wie gängige 3D-Modelling Tools (z.B. DAZ 3D oder Poser).

Die Scene-View



Der auffälligste Hauptbereich des Editors ist die sogenannte „Scene-View“. Sie ist der Hauptarbeitsbereich bei der Erstellung von Szenen wie unserer Beispielszene hier im Screenshot. Mit dem Bedienfeld im oberen Bereich lässt sich zur Game-View umschalten, in der die Szene interaktiv dargestellt wird. Man kann hierbei aus mehreren Darstellungsmöglichkeiten wählen (z.B. Wireframe-Modell, mit Lighting oder ohne, oder ob Audio abgespielt werden soll, oder nicht).

Das Project-Panel

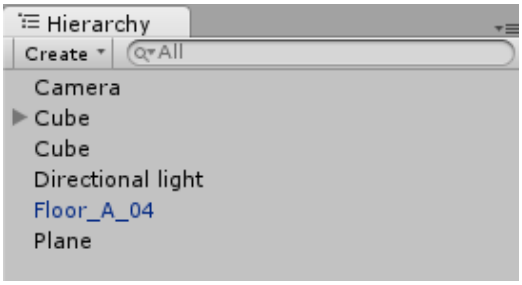


In diesem Bereich werden alle Ressourcen des Projekts hierarchisch angezeigt. Diese sogenannten Assets können jederzeit per Drag & Drop in ein Projekt eingefügt werden. Wenn ein Ordner im linken Bereich markiert wird, werden im benachbarten Bereich die beinhalteten Assets in einer kleinen Vorschau angezeigt. Im folgenden Beispiel eine Textur für ein Robotermodell:

Hier wurde der Ordner „Assets/Robot Kyle/Textures“ und die darin befindlichen Texturdateien angezeigt.

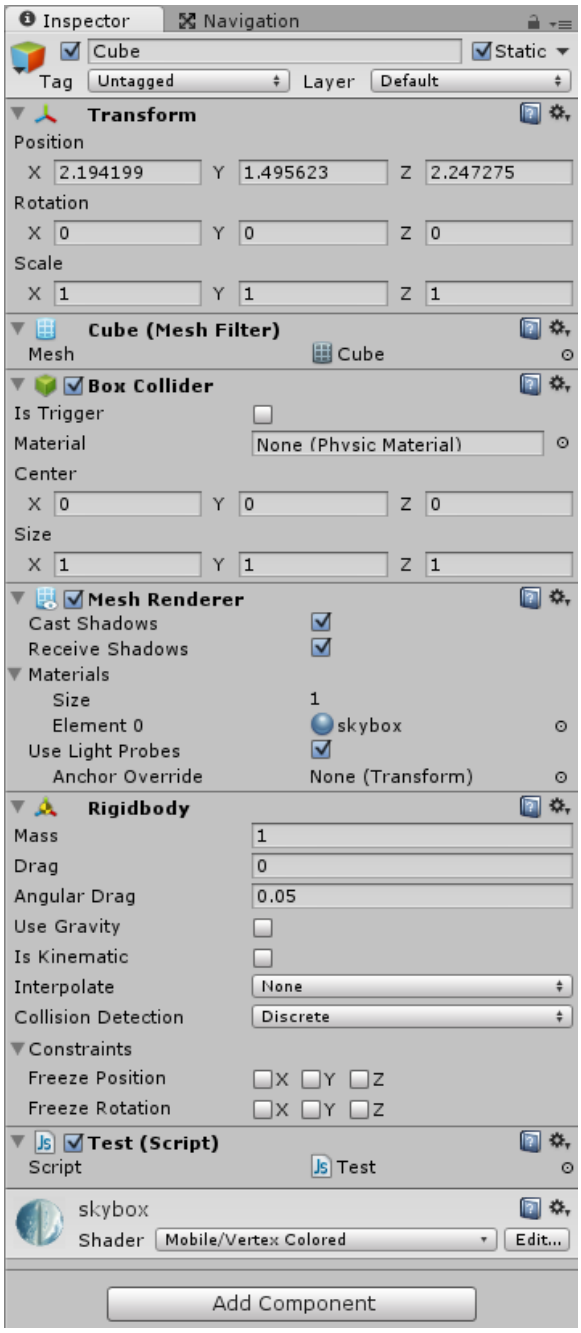
Der Console-Tab beinhaltet ein Log über aller Meldungen, die bei der Ausführung der Szene erscheinen.

Die Hierarchy



In diesem Bereich werden alle in der Szene platzierten Elemente aufgelistet. Child- und Parent-Beziehungen werden hierbei in einer aufklappenden Hierarchiestruktur dargestellt. Die einzelnen Elemente können so auch schnell ausgewählt werden, ohne dass man in die Scene-View klicken muss. Zusätzlich bietet sie noch eine Suchfunktion, um entsprechend benannte Objekte schnell ausfindig zu machen.

Der Inspector



Sobald ein Element einer Szene ausgewählt ist, werden alle dazugehörigen „Components“ angezeigt. Diese Components sind die Attribute, die das Element definieren. In unserem Beispiel wurde der farbige Würfel aus unserer Beispielszene ausgewählt.

Szenen und Szenenelemente

Eine Szene in Unity besteht aus Szenenelementen, denen wiederum Attribute zugewiesen sind. Die Art der Attribute ist von der Art des Elements abhängig. Im obigen Screenshot wurden die Attribute des farbigen Würfels aus unserer Beispielszene aufgelistet.

Er besitzt ein simples 3D-Modell („**Mesh**“), „**Transform**“-Werte, die seine Position, Größe und Rotation beschreiben,

einen „**Collider**“, der die physikalische Interaktion mit der Umgebung ermöglicht,

einen „**Mesh Renderer**“, der Informationen über die Textur und Oberflächenbeschaffenheit des 3D-Modells beinhaltet,

einen „**Rigidbody**“, der die Masse des Objekts definiert,

ein „**Script**“, um Interaktion zu ermöglichen,

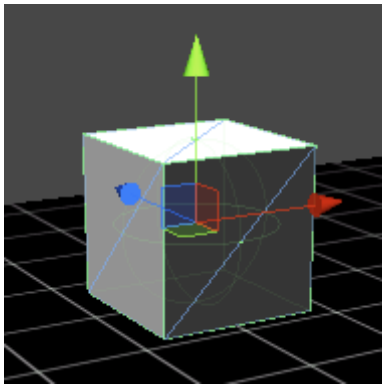
und ein „**Material**“, das Informationen über Texturen- und Lightmaps beinhaltet.

Mit wenigen Handgriffen lassen sich die erstellten Szenen in der „Scene View“ aus verschiedenen Blickwinkeln betrachten. Die Steuerung der Kameraansicht im Editor war anfangs zwar gewöhnungsbedürftig, ließ sich aber mit Hilfe der auf der Unity-Homepage zur Verfügung gestellten Tutorial-Videos schnell verinnerlichen.

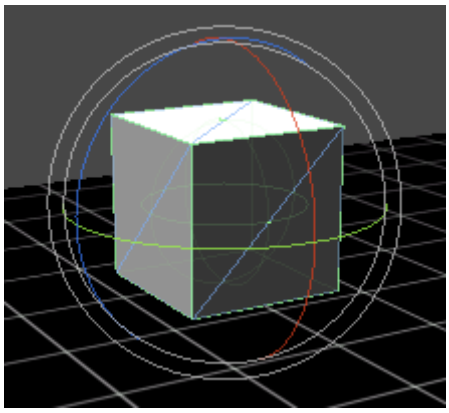
In der Scene-View existieren verschiedene Tools, die auf die Tasten „Q“, „W“, „E“ und „R“ gelegt sind und eine schnelle Interaktion ermöglichen.

Das **Hand**-Tool verhält sich genauso, wie man es von allen gängigen 3D-Modeling-Tools kennt und ermöglicht es dem Benutzer, die Szene auf dem Bildschirm zu verschieben.

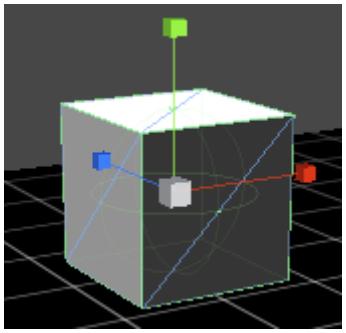
Das **Positioning**-Tool ermöglicht das Verschieben von platzierten Objekten in der Szene. Bei einem Klick auf ein Objekt werden 3 Achsen eingeblendet, mit denen sich das Objekt intuitiv im Raum bewegen lässt.



Das **Rotating**-Tool funktioniert ähnlich wie das Positioning-Tool. Bei einem Klick werden hier jedoch Rotationsachsen eingeblendet, mit denen sich das Objekt präzise ausrichten lässt. In unserer Beispielszene war dies vor allem bei Lichtquellen von großer Wichtigkeit.



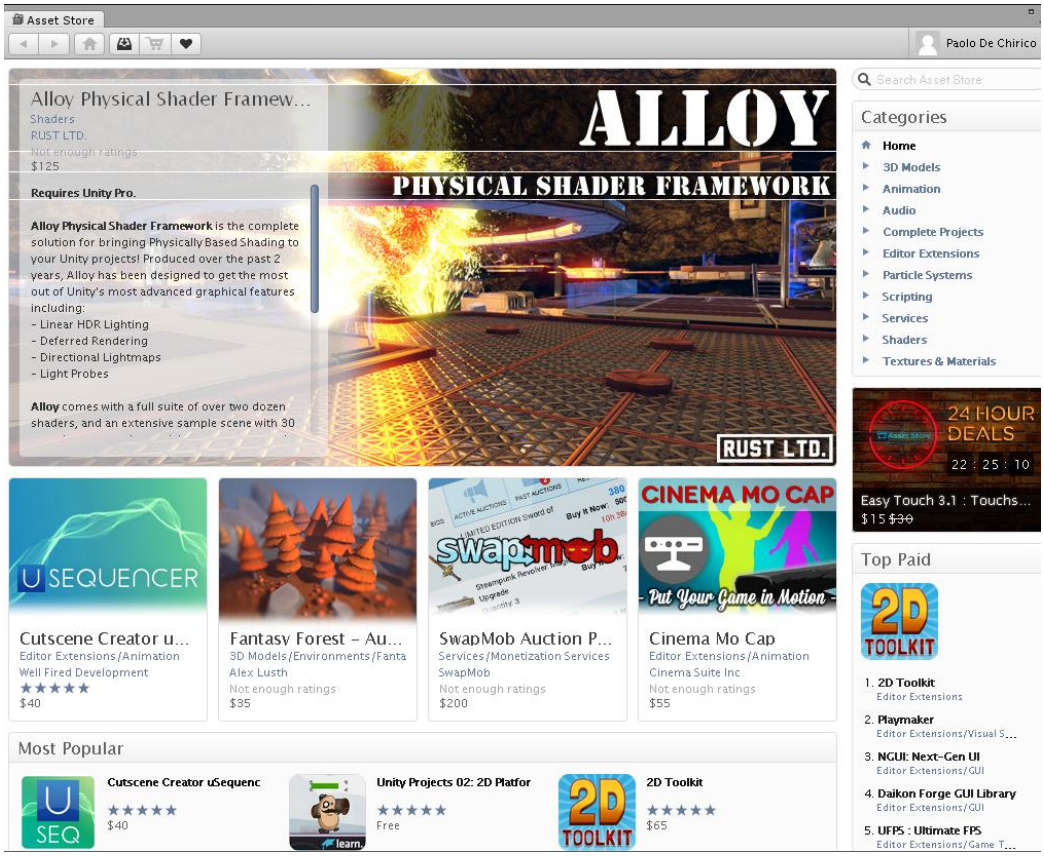
Das **Resizing**-Tool funktioniert ebenfalls nach dem gleichen Prinzip.



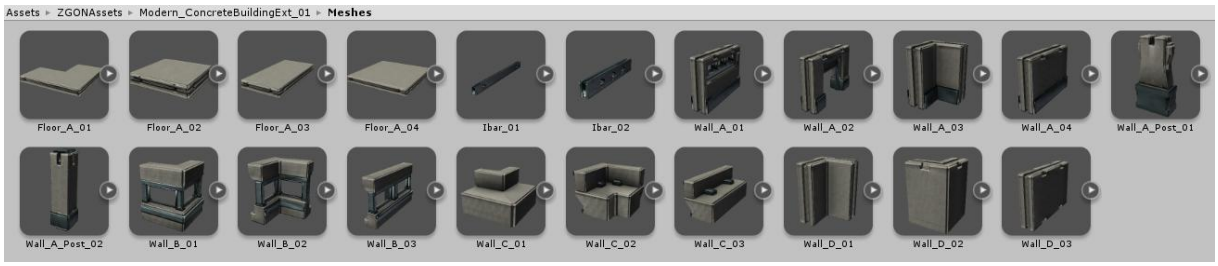
3D-Modelle und Animationen (Meshes) lassen sich nicht direkt im Editor erstellen, sondern benötigen externe Tools.

Der Asset-Store

Der eingebaute „Asset-Store“ ist nahtlos in den Editor integriert und erlaubt das schnelle und unkomplizierte Importieren gekaufter Ressourcen ins Projekt. Dies ist vor allem dann wichtig, wenn man schnellen Zugriff auf grafisch ansprechende Modelle benötigt.



Für unsere Beispielszene wurde z.B. eine umsonst erhältliche Fabrikhalle in Einzelteilen importiert.



Auf diesem Screenshot werden alle Bauteile der Fabrikhalle in einer Vorschauansicht aufgelistet. Sie lassen sich einfach über Drag & Drop in jede Szene einfügen. Der Store ist sehr schnell, effizient und integriert die heruntergeladenen Assets direkt in die Projekthierarchie.

[Meshes und Materialien beschreiben]

Fazit

Der Editor der Unity3D Engine hat sich während der Projektarbeit bereits in der freien Variante als sehr umfangreich und flexibel erwiesen. Besonders hilfreich war die sogenannte „Game View“, bei der die momentan bearbeitete Szene direkt so dargestellt wird, wie sie später im fertigen Spiel auch aussehen würde. Hierbei ist die gesamte Interaktivität gegeben, die bereits implementiert wurde. Die Eigenschaften der einzelnen Komponenten werden in Echtzeit wiedergegeben. Am hilfreichsten erwies sich hier vor allem beim Testen des Steuerungsskripts die Funktion, die „Game View“ zu jedem Zeitpunkt pausieren und wieder in der „Scene View“ bearbeiten zu können.

Das große Problem bei der Erstellung unserer Beispielszene war die Physiklogik der Engine. Auf den ersten Blick erschien uns die Szene hinreichend simpel, um sie schnell umsetzen zu können. Jedoch zeigten sich schon bald Probleme.

Unser Interaktionsskript zum ausschalten des Lichts funktionierte im Test einwandfrei, ebenso wie die verschiedenen Steuerungsskripts, die wir ausprobiert hatten. Die Probleme traten erst dann auf, wenn beide Skripte zusammen ausgeführt werden sollen. Wir vermuten, dass das Problem mit den Collidern und Rigidbodies zu tun hat, denn die Steuerungsprobleme verschwinden, sobald wir die physikalischen Interaktionsmöglichkeiten des steuerbaren Würfels ausschalten.

Zur schnellen Erstellung von Serious Games eignet sich die Unity3D Engine deshalb nur beschränkt. Für Laien, die keine wirkliche Erfahrung mit der Erstellung von Games haben, tauchen sehr schnell Probleme auf. So hilft dem Ersteller leider auch der sehr nutzerfreundlich integrierte Asset-Store nichts.

Selbst unsere simple Beispielszene hat also bereits den Rahmen unseres Projekts gesprengt und uns gezeigt, wie komplex die logischen Abläufe hinter Serious Games sein können.

Anmerkung

Ganz am Ende des Projektes ergab sich noch ein weiteres Problem. Die Präsentation hätte eigentlich auf einem Laptop direkt im System gehalten werden sollen. Erst wenige Stunden vor der Präsentation stellten wir im finalen Test fest, dass Unity scheinbar deutlich anspruchsvoller an Grafikkarte und Treiber ist, als gedacht. Dies zumindest ist unser Schluss, weswegen auf dem eigentlich recht neuen Laptop das Programm nicht funktionierte, sondern dauerhaft die Fehlermeldung "Unable to initialize unity graphics" ausgab und das Programm schloss. Sehr kurzfristig entstand daher eine kleine Power-Point-Präsentation, die gemeinsam mit dem zuvor erstellten Video als Präsentation dienen musste.