Ein Tag in der Flax-Engine

Description

Besonders in den letzten Wochen hörte ich vermehrt von der Flax-Engine. Sie wird gerne als Alternative zu Unreal und Unity bezeichnet. Da ich derzeit ohnehin nach 3D-Engines Ausschau halte, wurde es Zeit für einen intensiven Test. Ich verbrachte einen Arbeitstag in der Flax-Engine. Es war eine interessante Reise, die ein plötzliches Ende nahm.

Was ist die Flax-Engine?

Es handelt sich dabei um eine Echtzeit-3D-Game-Engine. Sie wurde in Polen erschaffen, um die Entwicklung von 3D-Spielen und interaktiven Anwendungen zu ermöglichen.

Hier ein Auszug, wie der Entwickler seine Engine beschreibt:

> „Von atemberaubenden Grafiken bis hin zu leistungsstarken Skripten – Flax kann alles für Ihre Spiele bieten. Entwickelt für schnelle Arbeitsabläufe mit vielen gebrauchsfertigen Funktionen, die jetzt auf Sie warten. Sehen Sie sich weitere Feature-Highlights an!“

Die Flax-Engine unterstützt die Entwicklung von Spielen für verschiedene Plattformen, darunter Windows-PCs und Konsolen. Sie bietet eine breite Palette von Tools und Funktionen für die Erstellung von 3D-Welten, Animationen, Physiksimulationen, Beleuchtung, Materialien und vielem mehr. Ich testete die Version 1.6.6344.

Mein Vorwissen

In den vergangenen Jahren hatte ich zwar immer wieder mit 3D-Engines zu tun, aber bei weitem nicht so intensiv wie im 2D-Bereich. Bei Unreal komme ich auf vielleicht fünf bis sieben Stunden, verteilt auf mehrere Jahre und Versionen. Bei Unity sind es keine hundert Stunden, ebenfalls auf viele Jahre verteilt. Meine letzten Experimente darin sind mindestens zwei Jahre her.

Zwar hatte ich noch weitere Erfahrungen, aber diese sind zum Teil noch älter. Das heißt, dass ich

verschiedene Bedienkonzepte und technische Hintergründe kenne, würde mich aber auf keinen Fall als Experten bezeichnen, weshalb ich mich in der Flax-Engine hier und da sicher etwas dumm angestellt habe. Aber genau das ist das Interessante: Wie kommt jemand damit zurecht, der keine Ahnung hat, oder zumindest nicht viel?

Schaut man sich Screenshots und Videos des Entwicklers an, wirkt alles total easy und die Resultate können durchaus beeindrucken. Was ich aber wissen will, ist die Alltagstauglichkeit für einen Anfänger.

Installation

Ich lade mir die Engine herunter und starte die Installation. Zunächst werde ich nach einem Account gefragt. Brauche ich den? Wahrscheinlich für Assets, also erstelle ich ihn. Daraufhin stelle ich fest, dass ich den Account für den Flax-Store brauche, aber den Store gibt es momentan noch nicht.

Der Willkommensbildschirm sieht recht bescheiden aus. Man muss dann erst noch die Engine installieren. Die Menüs sind nicht für 4k-Auflösungen optimiert, die Schriften wirken sehr klein und gelegentlich stupse ich mit der Nase an den Monitor.

Dann muss man auf Projects, um ein neues Projekt zu erstellen. Alles wirkt ein wenig fummelig. Ich kann nun wählen zwischen Basic-Scene und Blank und entscheide mich für Basic. So habe ich etwas, womit ich herumspielen kann. Nun will ich starten und es wird mir jetzt erst mitgeteilt, dass .NET 7.0 fehlt. Warum nicht schon bei der Installation?

Mich beschleicht der Eindruck, dass totale Anfänger bis hierhin schon ohne Tutorial komplett aufgeschmissen sind.

Nach der Installation von .NET 7.0 kann ich mein Projekt starten.

Erster Eindruck

Es sieht auf den ersten Blick aus, als hätten Unreal und Unity ein Kind bekommen. Die Fenster sind sehr ähnlich aufgebaut, um nicht zu sagen, 1:1 kopiert. Einerseits ist das sehr gut für Umsteiger, andererseits stelle ich mir langsam die Frage, wo die entscheidende Innovation ist. Warum soll man Flax nutzen?

Für Skripte kann ich *C#* oder *C++* nutzen. Außerdem gibt es einen Visual-Code. Als Codeeditor wird Visual Studio Code von Microsoft empfohlen. Den hatte ich zum Glück schon.

Wenn mehrere Programmiersprachen von einer Engine unterstützt werden, habe ich immer etwas Bauchschmerzen. Sofern man nicht alles zu 100% selbst macht, sondern auf fremden Code zurückgreift, erhält man mit der Zeit eine kunterbunte Mischung. Witzig ist, dass Flax die Sprachen von Unity (*C#*) und Unreal (*C++*) unterstützt, statt eigene Wege zu gehen oder zumindest nur eine Sprache zu unterstützen.

Fenster und sonstige Ungereimtheiten

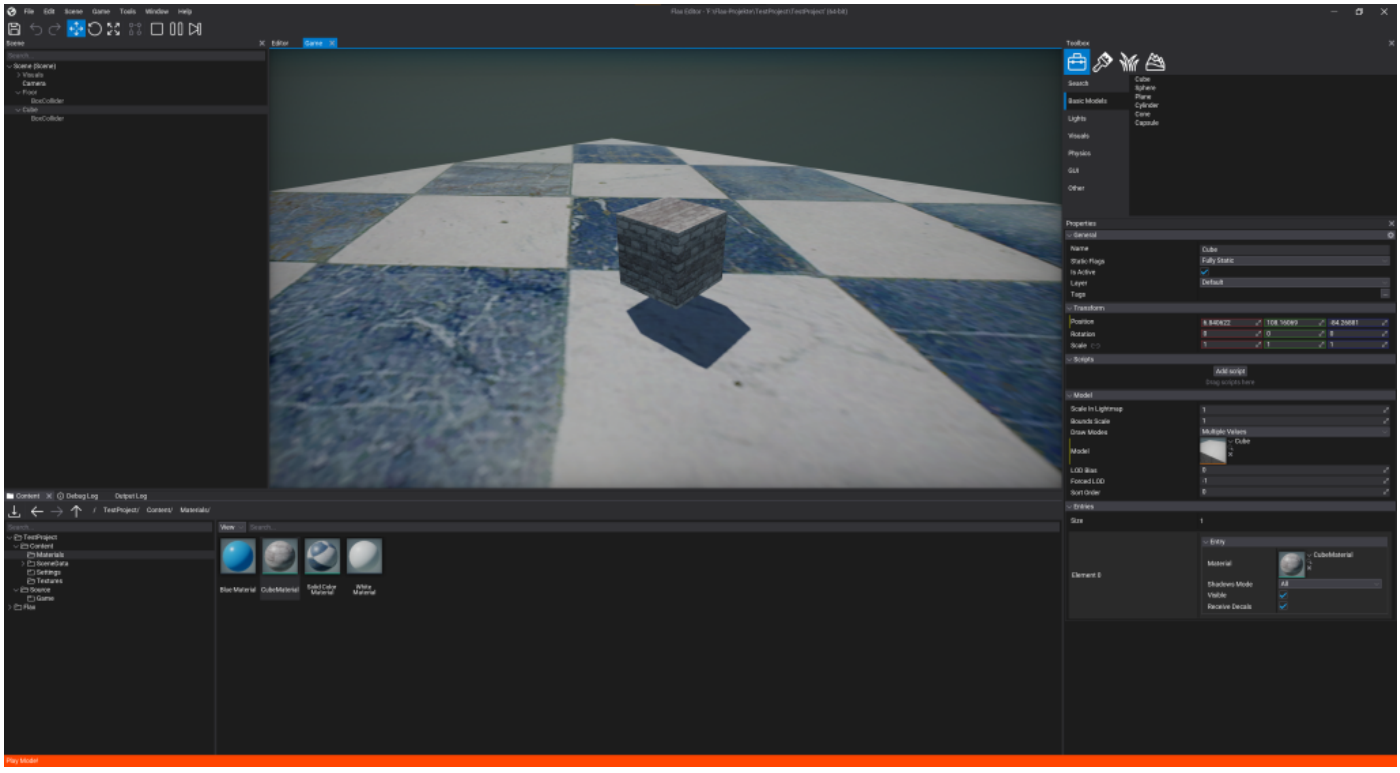
Die Fenster-Engine wirkt komisch. Wenn ich ein Fenster verschieben will, sehe ich nur eine Vorschau. Auch das Skalieren verhält sich eigenartig. Will ich die Größe anpassen, sehe ich zunächst nur eine verzerrte Perspektive meines Fensters. Erst, wenn ich loslasse, bekomme ich eine aktualisierte Version.

Für eine Version 1.6 wirkt das alles noch nicht richtig ausgereift. Eher wie 0.1 oder 0.2. Aber vielleicht überraschen mich die inneren Werte.

Texturierung

In meiner Basic-Szene habe ich neben Himmel, Licht und Kamera eine Bodenplatte und einen Würfel. Daraus lässt sich etwas machen. Ich will den Boden vergrößern und texturieren. Das Skalieren geht denkbar einfach, vor allem wenn man sich etwas mit Unity auskennt.

In die Texturierung muss man sich hingegen einarbeiten, vor allem wenn es um das Tiling geht. Wenn ich den Boden vergrößere, muss ich meine 1024×1024 Textur darauf kacheln, wenn es nicht aussehen soll wir Erbrochenes. Doch nach der passenden Einstellung kann man lange suchen. Zum Glück gibt es ein eigenes Forum, in dem man fündig wird. Hier ist zwar noch nicht so los wie in anderen Communitys, aber es wurden zumindest viele Fragen für Anfänger zufriedenstellend beantwortet.



Meine erste Testszene in der Flax Engine

Irgendwann gelingt es mir, dass der Boden halbwegs so aussieht, wie ich es mir vorstelle. Bei dieser Gelegenheit verpasse ich meinem Würfel ebenfalls eine Textur. Dank Normal-Maps sieht das sogar recht gut aus.

Zwar ist die Texturierung etwas fummelig, aber wenn man sich die Materialien anlegt, kann man sie später bequem duplizieren und die entsprechenden Texturen reinziehen.

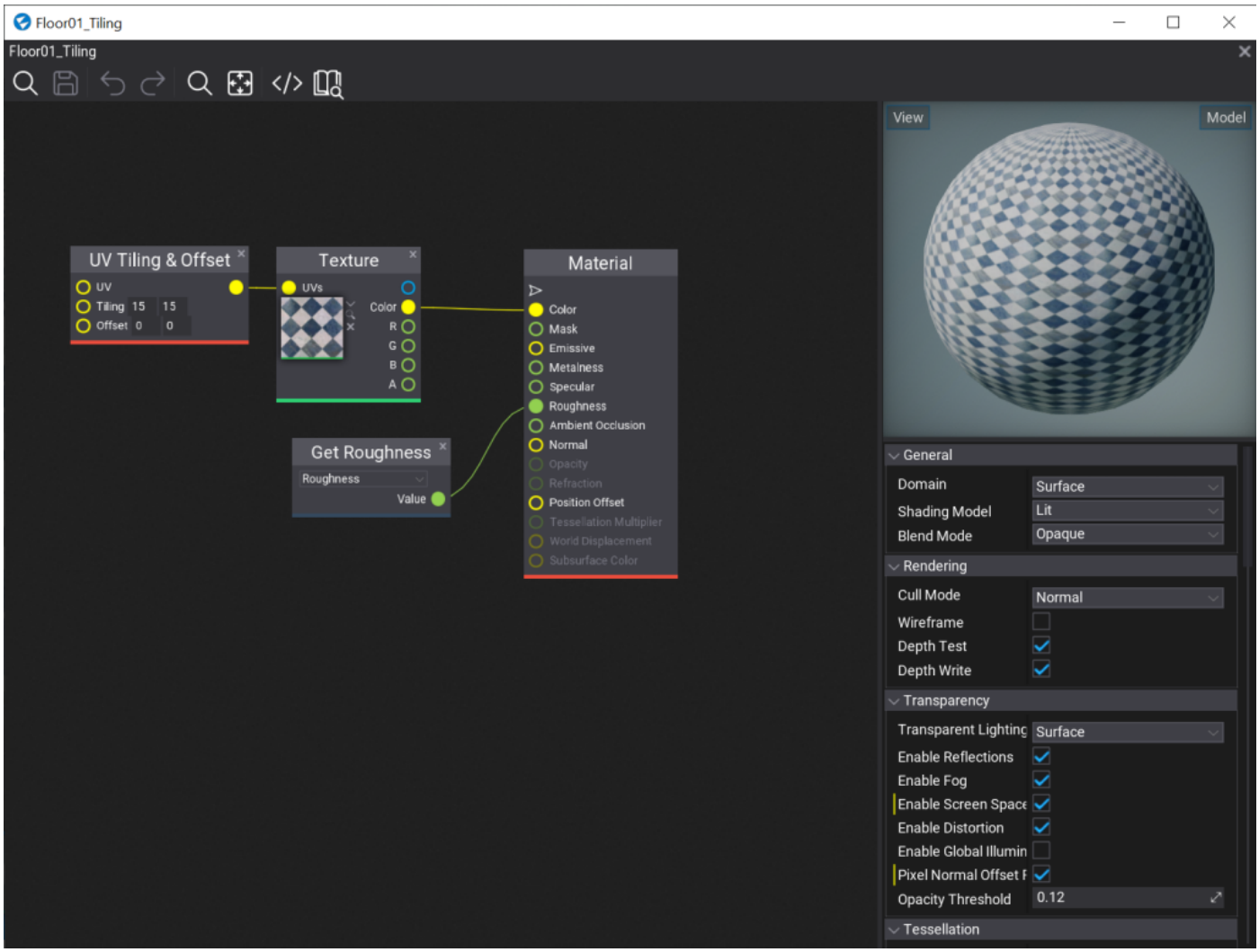
Das Problem mit dem Levelbau

Flax scheint ein ähnliches Problem zu haben wie die allermeisten mir bekannten modernen 3D-Engines. Man kann sehr einfach weitläufige Landschaften erschaffen und – wie ich später erfahren werde – bequem 3D-Objekte einladen. Aber will man einen klassischen Shooter wie in den 1990er Jahren bauen, hakt es am Levelbau.

Heute ist es eher so, dass man nahezu alles in einem 3D-Programm wie Blender baut und diese Objekte in der Engine integriert. Früher nutzte man, je nach Spiel, eher Brushes. Man baute also die Blöcke innerhalb des Editors und schuf damit Böden, Treppen, Wände und den ganzen Rest. Diese Technik hat einige Vorteile. Wenn alles richtig eingestellt ist, wird das Tiling der Texturen automatisch erzeugt. Man kann einzelne Flächen texturieren, ist extrem schnell und muss sich nicht mit Blender und ähnlichen Programmen herumschlagen. Das scheint hier alles nicht möglich zu sein, zumindest habe ich im Laufe meines Tests nichts Entsprechendes gefunden.

Scripting

Wenn ich mir meine texturierte Szene anschaue, kommen alte Demoszene-Reflexe in mir zum Vorschein. Der Würfel soll sich drehen. Dafür brauche ich ein Script. Das Erzeugen geht sehr einfach, aber um vernünftig zu programmieren, sollte man auch die Befehle kennen. Somit schaue ich in die Dokumentation und stelle fest, dass ein Anfänger spätestens hier aufhören würde. Das ist keine richtige Dokumentation, lediglich eine Auflistung der Befehle mit einer sehr rudimentären Beschreibung. Mir macht das nicht viel aus, aber wer noch nie programmiert hat, tritt Flax an dieser Stelle in die Mülltonne.



Material und Tiling in der Flax Engine

Die sogenannten Beschreibungen wirken wie ein Reiseführer, in dem steht: „Gehen sie selbst hin und schauen sie sich um.“ Danke für nichts. Was hier eindeutig fehlt, sind ein oder zwei konkrete Beispiele. Das würde Anfängern wirklich helfen.

Außerdem sind die Meldungen beim Compiler kaum lesbar. Der Hintergrund des Fensters ist dunkelgrau, die Fehlermeldung rot. Dazu kommt eine sehr kleine Schrift. Bei jedem Fehler muss ich die Meldung in ein anderes Textfenster kopieren. Kann man das umstellen? Ich finde nichts. Na super.

Der drehende Würfel

Es war eine schöne Fingerübung. Wie ich es aus Unity kenne, kann ich später die definierten Parameter bequem im Editor anpassen. Mein Script sieht so aus, dass ich für alle drei Achsen eine eigene Geschwindigkeit definieren kann. Hier ist der Code:

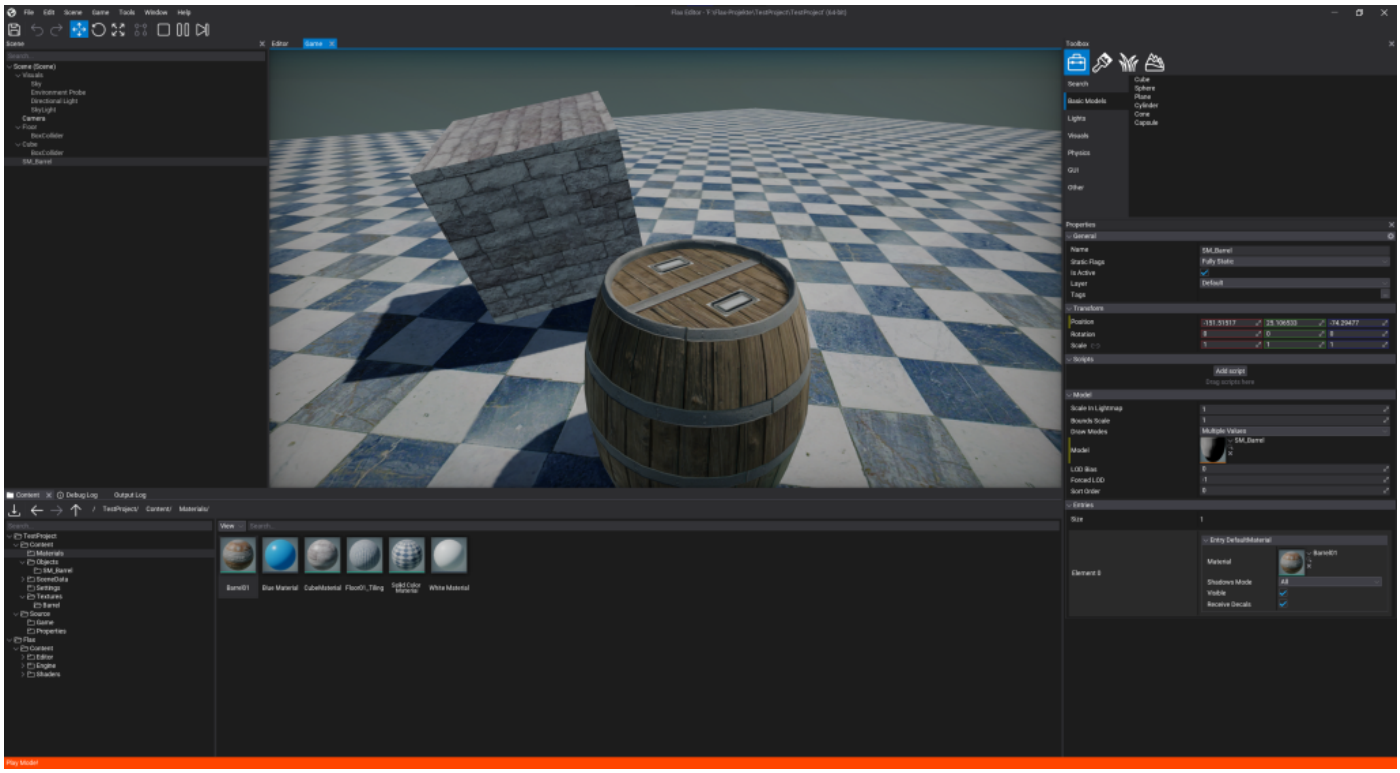
```
using FlaxEngine;

public class TestRotateAround : Script
{
```

```
    public float RotationSpeedX = 0.0f; // Drehgeschwindigkeit um die X-Achse
    public float RotationSpeedY = 90.0f; // Drehgeschwindigkeit um die Y-Achse
    public float RotationSpeedZ = 0.0f; // Drehgeschwindigkeit um die Z-Achse

    /// <inheritdoc/>
    public override void OnUpdate()
    {
        // Drehgeschwindigkeiten für jede Achse separat
        RotateAround(Vector3.Right, RotationSpeedX * Time.DeltaTime); // Um di
        RotateAround(Vector3.Up, RotationSpeedY * Time.DeltaTime); // Um die Y
        RotateAround(Vector3.Forward, RotationSpeedZ * Time.DeltaTime); // Um
    }

    public void RotateAround(Vector3 axis, float angle)
    {
        Transform transform = Actor.Transform;
        Quaternion q = Quaternion.RotationAxis(axis, angle * Mathf.DegreesToRa
        transform.Orientation = q * transform.Orientation;
        Actor.Transform = transform;
    }
}
```

Das hat eine Weile gedauert, aber ich habe viel gelernt und es hat auch Spaß gemacht. Doch – so viel sei gesagt – der Spaß wird nicht ewig anhalten.

Die ersten Objekte

Nachdem die ersten Hürden gemeistert wurden, ging es darum, ein 3D-Objekt einzuladen. Ich hatte ein Fass herumliegen, mit einer UV-Map. Das ging erstaunlich schnell und unkompliziert. Nach einigen Klicks und Einstellungen wurde es mir perfekt im Spiel angezeigt. Danach noch einen Totenschädel, ebenfalls mit UV-Map, aber dafür sehr groß skaliert. Das war auch kein Problem.



Mein erstes 3D-Objekt in der Flax Engine: Ein texturiertes Fass

Sehr erfreulich ist, dass sowohl OBJ als auch FBX unterstützt wird. Bei den Texturen hatte ich weder mit JPG, PNG noch TGA irgendwelche Probleme.

Zum Abschluss nahm ich noch mein vorheriges Script, um den Schädel zu drehen. Funktionierte einwandfrei. Ein schönes Gefühl!

Seltsame Einstellungen

Mir fällt auf, dass die Qualität des Schattens schlecht ist. Die kanten Flackern. Irgendwo muss man das umstellen können, also befasse ich mich mit den Einstellungen. Davon gibt es einige in Flax und irgendwann finde ich auch das richtige Fenster.

Merkwürdigerweise werden meine Einstellungen zunächst nicht gespeichert. Also wieder öffnen, ändern, schließen. Irgendwann klappte es, aber der Schatten sah gleich aus. Gibt es wirklich keinen Unterschied zwischen Medium und Ultra? Sehr seltsam.

https://www.bytegame.de/wp-content/uploads/2023/08/Flax-Engine-001.mp4

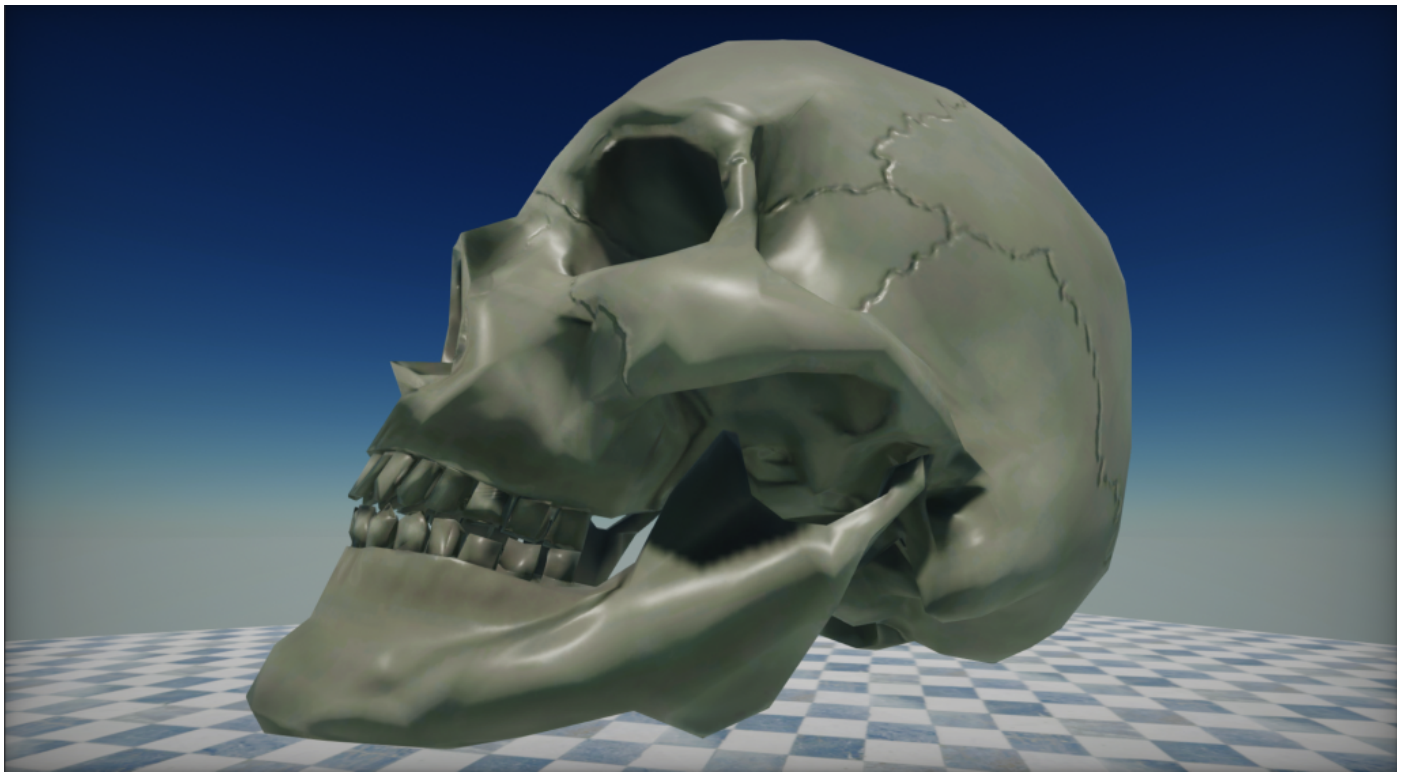
Auch die Auflösung kann ich nicht einfach so umstellen. Es wird immer im Fenstermodus angezeigt. Immerhin kann ich mit Alt+Enter in den Vollbildmodus. Ich nehme an, man muss das per Code machen und lasse es. Mein geplanter Arbeitstag geht nicht mehr so lange.

Sonstige Erfahrungen

Ich teste einige Dinge und schaue in irgendwelche Anleitungen. Das Visual-Spripting sieht ganz nett

aus, aber ab einer bestimmten Komplexität: Viel Spaß beim Debugging!

Die Kinderkrankheiten erkennt man u. a. daran, dass man das Projekt schließen und neu starten muss, wenn man neue Layer hinzufügt. Zumindest zeigte sich das in einem Tutorial mit einer etwas älteren Version.



Ein Texturierter 3D-Schädel

Animationen zu implementieren ist nicht so schwierig, aber nicht so intuitiv, als das man es ohne Tutorials hinbekommen würde. Allgemein zeigt sich, dass man in Flax nicht gut angeleitet wird. Möglicherweise liegt das auch an der Komplexität. Wer aber gedacht hat, er kann hier auf die Schnelle ohne große Hilfe Erfolge erzielen, hat sich geschnitten.

Projekt Ego-Shooter

Nachdem ich bisher viel Zeit in der Dokumentation, im Forum und diversen Seiten verbrachte, wurde es Zeit für ein praktisches Projekt. Es soll ein Ego-Shooter werden.

Ich schaue mich ein wenig um, wie andere das Problem mit den Wänden und dem Tiling gelöst haben. Soweit ich sehe, bauen die alles Würfel für Würfel. Furchtbar, aber für den ersten Test sollte es reichen. Außerdem fallen mir gleich ein paar schöne Ideen ein. Wenn alles aus Würfeln besteht, kann man das am Levelanfang entsprechend aufbauen. Alles, bis auf den Boden, auf unsichtbar stellen und dann nacheinander einblenden. Das ist bestimmt ein krasser Effekt, aber es sollte nicht so weit kommen.

Ich beginne wieder mit der Basic-Szene. Meinen Boden mache ich sehr groß, lade mir eine hübsche Bodentextur ein, inklusive Normal-Map. Durch das entsprechende Tiling sieht es toll aus, aber viel zu

hell. Liegt wahrscheinlich am Licht, aber sollte mich jetzt nicht stören.

Anschließend kommt die Wand dran. So lange ich Würfel nehme, sieht es gut aus. Aber wehe, ich mache die Wand schmal, dann wird die Textur verzerrt. Dafür gibt es sicher auch eine Lösung, aber das sollte ein Problem von Future-Sven werden. Ich kümmere mich lieber erst um den Player und die Steuerung.

Das geht zunächst recht einfach. Ich erzeuge ein entsprechendes Objekt, hefte die Kamera dran, nehme Einstellungen vor und es sieht alles aus, wie man es aus **Doom** oder **Quake** kennt.

Nun bin ich nicht der erste Blödmannsgehilfe, der einen Shooter mit Flax realisieren will. Für die Steuerung gibt es bereits fertige Skripte. Ich baue eines davon ein. Lege die Kamera fest, das Target, den Player. Dann dreht der Compiler durch.

Das frühzeitige Ende

Der Compiler zeigt mir Fehler in einer Schleife an. Der Text rast förmlich über das Fenster, ich kann nichts lesen und kann es nicht anhalten. Also schaue ich zunächst im Code, wo das Problem sein könnte. Da wird mir aber kein Fehler angezeigt, alles sieht normal aus.

Was tut man, wenn man als IT-Mensch so ein Problem hat?

> „Haben sie es schon einmal mit ausschalten und wieder einschalten versucht?“

Genau. Aber vorher speichern. Sicher ist sicher. Ich speichere ab, schließe Flax. Anschließend wird alles neu gestartet. Ich rufe mein Projekt auf, dann kommt eine Fehlermeldung. Ich denke, es liegt am Skript, also klicke ich auf okay. Eine andere Möglichkeit habe ich auch nicht.

Und dann? Alles weg. Wände, Boden, Himmel, Licht, Kamera. Es ist nichts mehr da. Flax hat alles gelöscht. An dieser Stelle breche ich ab.

Fazit

Am Anfang stellte ich mir die Frage, warum ich Flax statt Unity oder Unreal nehmen sollte. Mir fällt kein Grund ein, außer man möchte die Entwicklung eines sicherlich spannenden Projekts begleiten. Als Content-Creator könnte es durchaus interessant sein, wenn man einer der ersten Anbieter im kommenden Flax-Store ist.

Möglicherweise habe ich mich hier und da dumm angestellt, aber mein Eindruck ist, dass Flax noch ein gutes Stück davon entfernt ist, dass man relativ bequem ein größeres Spielprojekt umsetzen kann. Viele Dinge sind unausgereift, fehlerhaft und nicht wirklich intuitiv. Ich erkenne auch keinen Mehrwert gegenüber den beiden Platzhirschen.

Der Praxistest hat auch gezeigt, dass man nicht viel auf Werbevideos der Entwickler geben sollte. Journalistisch betrachtet neigt man gerne dazu, solchen Videos zu glauben und Pressemeldungen zu übernehmen, weil es sehr viel Zeit kostet, die Dinge selbst zu testen. Ich bin froh, diesen Weg

gegangen zu sein, und bin gespannt, wie sich Flax weiterentwickelt.

Weiterführende Links

Erste Schritte in Pico-8
Gameentwicklung mit Godot – erste Schritte
Ein Raubvogel erobert die Herzen – Interview mit Scott Host – Teil 1

Externe Links

Flax Engine
Blender
Unreal Engine
Unity Engine

Date Created
29. September 2023
Author
sven