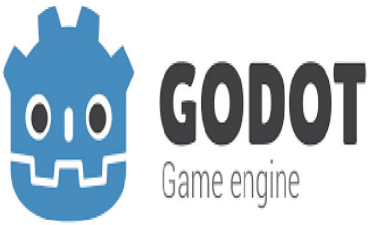


Gameentwicklung mit Godot – erste Schritte

Description

Letztes Jahr zu Halloween hat es mich auch erwischt – ich hatte eine Spielidee und die Ambition, diese Idee zu verwirklichen. Als Engine standen eigentlich nur Godot oder Unity zur engeren Auswahl. Da ich den Open-Source-Gedanken von Godot sehr sympathisch finde, und mich die Engine im Vorfeld einfach neugierig gemacht hat, entschied ich mich für Godot (zu Halloween 21 noch Version 3.34)

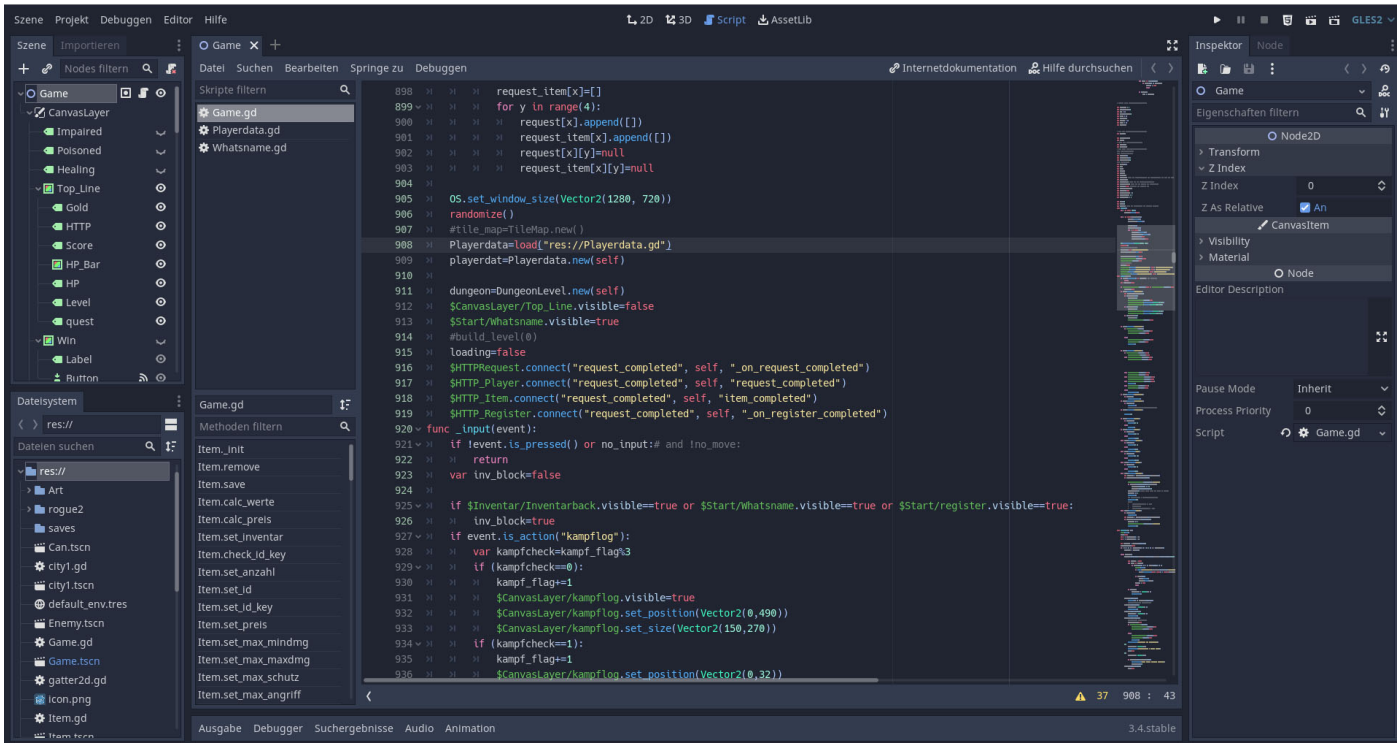
Vorüberlegungen

Godot ist OpenSource unter der recht freizügigen MIT-Lizenz. Das führt zum einen zu einer freizügigen Lizenz-Politik. Es gibt keine versteckten Abgaben oder Kosten. OpenSource bietet zudem die Möglichkeit, selbst im Projekt aktiv zu werden. Allerdings bedeutet es auch, dass eventuelle Bugs erst behoben werden, wenn sich jemand dazu berufen fühlt. So mancher Kritiker kommt mit diesem Prinzip nicht zurecht und erwartet einen Support, wie er nur bei kommerziellen Produkten üblich ist. OpenSource muss man schon mögen, wenn man sich darauf einlässt.

Die Godot-Engine unterstützt 2D- sowie 3D-Projekte, hat allerdings den Ruf, sich mit AAA-Grafik schwer zu tun. Allerdings – als Soloentwickler ohne Grafiker-Team wird man wohl kaum an dieser Grenze ernsthaft kratzen.

Installation von Godot

Auf godotengine.org kann man die Engine problemlos runterladen. Das Zip enthält nur eine exe mit ca. 75 MB. Das ist der Kern des Editors und reicht zum Basteln und ausprobieren völlig aus. Will man das Projekt später als lauffähige Exe exportieren, so benötigt man noch die Templates für diverse Plattformen – das sind dann allerdings noch mal ca. 500 MB, die aber innerhalb des Editors gedownloadet werden können.

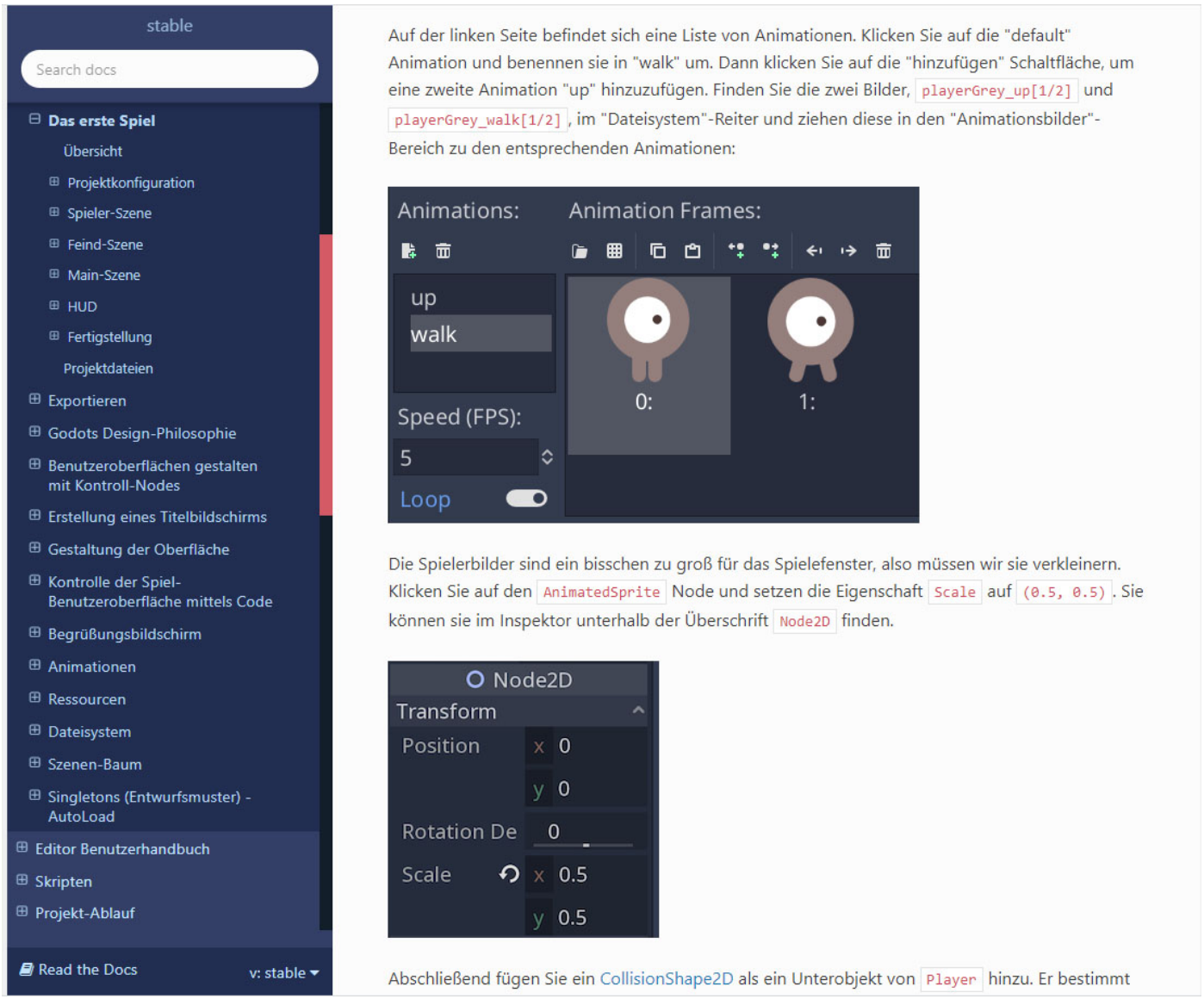


Der Editor besitzt umfangreiches Highlighting und Autovervollständigung im Codebereich

Programmierung

Ich benutze das GDScript, eine an Python angelehnte Skriptsprache die sehr eng mit dem Editor verbunden ist. Mit GDScript kann man objektorientiert programmieren, aber Godot kann auch C++ und andere Programmiersprachen – es wird jedoch an allen Ecken in der Doku GDScript empfohlen, weil es quasi die natürliche Sprache der Engine ist. Ich persönlich komme mit der Syntax ganz gut zurecht, zumal der Editor eine umfangreiche Syntax-Vervollständigung anbietet und viele Tippfehler schon vor dem kompilieren entdeckt.

Godot setzt voll auf das Szenenmodell mit Nodes. Nodes sind alle möglichen Objekte und können eigene Scripte besitzen. Nodes werden in Szenen organisiert. D.h. es gibt Szenen mit eigener Funktionalität, die man separat erstellen und in einem Baukastensystem verwenden kann. So habe ich beispielsweise eine eigene Szene für meine Spielfigur, welche neben dem Sprite auch eine Belichtungs-Node sowie ein eigenes Script besitzt. Diese Szene instanziere ich dann in meiner Hauptszene und kann auf alle Elemente zugreifen.



Die offizielle Doku von Godot ist in diverse Sprachen übersetzt und liefert neben umfangreichen Infos auch praktische Beispiele.

Godot-Doku

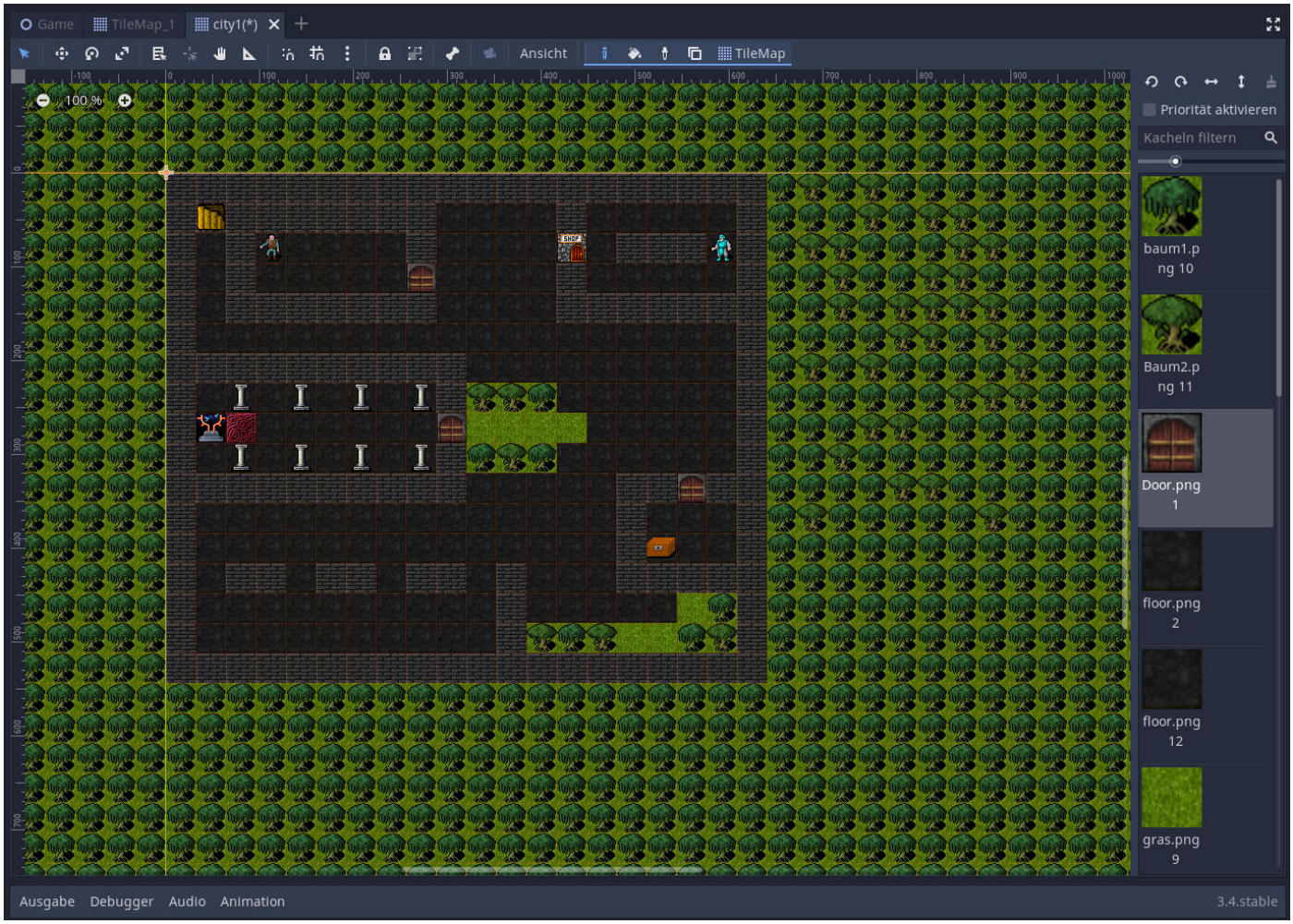
Es gibt eine in mehrere Sprachen übersetzte Grund-Doku auf der Hauptwebseite. Dort wird im Laufe der Einführung auch ein erstes Spiel vorgestellt. Ich selbst lerne am besten durch ausprobieren. So verlasse ich relativ zügig die Gleise der Dokumentation und stürze mich auf mein Projekt.

Es gibt ein englischsprachiges, offizielles Forum und auch bei Reddit etc. stößt man bei Google immer auf hilfreiche Code-Schnipsel. Allerdings ist dieser Doku-Content größtenteils auf Englisch. Ich habe bisher kein einziges deutschsprachiges und lebendiges Godot-Forum im Netzt finden können. Ebenso gibt es diverse Tutorials auf Youtube – aber auch hier sind die wenigsten in Deutsch. Somit erleichtern ein paar Englischkenntnisse die Recherche deutlich.

Trotz der umfangreichen Doku auf der Hauptseite: manche Dinge lernt man aber auch erst „zufällig" durch einen kleinen Tipp in einem Forum. So hatte ich das Problem, dass mein Label (für Text) unter

den Sprites dargestellt wurde. Die Label-Node selbst besitzt aber keine Einstellung für einen Z-Index (Darstellungs-Hierachie). Also ist der Trick, eine Standard-Node als Eltern-Node zu verwenden (Node2D) welche über einen Z-Index verfügt und somit das Sprite mit „nach oben bringt". Doch wie greife ich dann auf das Label zu, wenn ich im Programmcode den Text ändern möchte? Dazu erstelle ich eine Funktion in der Eltern-Node (z.Bsp. namens set_text), über welche ich problemlos auf das Kind zugreifen kann. Das Prinzip ist einfach – man muss es nur erst einmal drauf kommen!

Erste Schritte zur Installation und ersten Einstellungen gibt es unter anderem hier.



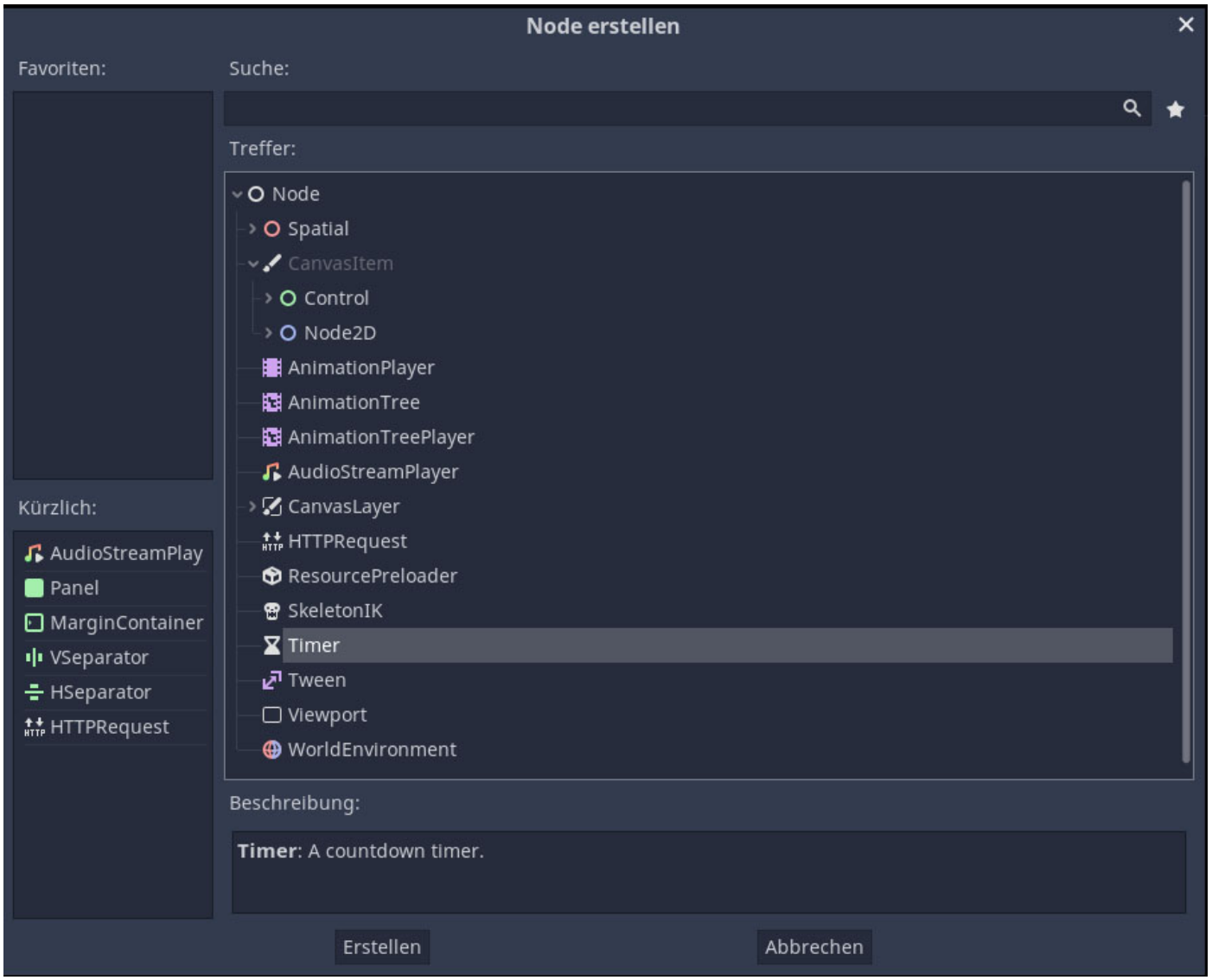
Der TileMap-Editor ermöglicht es, mit der Maus seine Game-Maps zu erstellen.

Editor

Der Editor besteht aus einem sehr komfortablen Text-Editor und einem grafischen Editor zur Erstellung von z. Bsp. Maps oder Nutzeroberflächen. Der Text-Editor überzeugt mit Syntax-Highlighting und Syntax-Vervollständigung – zudem erkennt der Editor die gröbsten Tippfehler zur Laufzeit.

Die Optionen des grafischen Editors sind für mich immer noch etwas unübersichtlich – jeder Node hat eigene Optionen, und oft suche ich Anfangs nach ganz einfachen Dingen wie die Position oder den Z-Index. Das gibt sich zum Glück nach einer gewissen Zeit.

Insgesamt entwickelt es sich aber erstaunlich flüssig mit der Godot-Engine. So habe ich oft eine Idee, dann einen Plan für die Umsetzung – und bin dann überrascht, dass sich mein Plan auch genau so umsetzen lässt.



Es gibt eine Reihe von Nodes, welche man in sein Projekt einbinden kann. Alle Nodes haben Eigenschaften und können mit einem Script erweitert werden.

Assets

Die offizielle Godot-Asset-Library kann man nicht mit der Unity-Asset-Library vergleichen. Ohne finanziellen Anreiz gibt es leider in unserer Welt auch weniger Anbieter. Dennoch findet man auf in der Godot-Asset-Library einige vielleicht ganz hilfreiche Tools und Hilfen, und es kann auf jeden Fall nicht schaden, sich zumindest ein Bild davon zu machen, was angeboten wird. Für mein eigenes Projekt bin ich jedoch auf opengamesart.org eher fündig geworden – zumindest was Grafiken und Sound angeht.

Für mein erstes Projekt fand ich zum Glück bei Youtube ein Video (Building a Roguelike from Scratch)

(leider auf englisch und sehr schnell gesprochen). Allerdings mit den Scripten auf Github. Dieses Grundgerüst erwies sich als enorm hilfreich, denn das meiste konnte ich nach und nach verstehen und mir erarbeiten. Ich habe bei der Gelegenheit auch festgestellt, das gerade englischsprachige Videos für mich oft nicht so informativ sind wie ein paar Zeilen Code mit Erläuterung in einem Forum.

Fazit

Godot ist eine vollwertige Entwicklungsumgebung und bietet alle professionellen Möglichkeiten einer modernen Gameengine. Durch das OpenSource-Modell ist zudem die Möglichkeit gegeben, an der Entwicklung der Engine teilzuhaben. OpenSource-Typisch ist allerdings auch der Support: Wenn man auf einen Bug stößt, der nicht primär relevant ist, dann muss man selbst anpacken oder Geduld haben – niemand ist hier verpflichtet.

Ich empfand GDScript als vollständig in die Engine integrierte Sprache sehr angenehm für meine ersten Experimente. Ob die Sprache einer sehr großen professionellen Entwicklung standhält, kann ich nicht beurteilen. Für mittlere Projekte erscheint mir GDScript jedoch als Sprache der Wahl, wenn man mit Godot entwickeln möchte.

Ich denke, dass die Kosten bei lizensierten Engines nicht unbedingt der abschreckende Faktor ist. Es ist eher die Entscheidung zwischen kommerzieller (sowie lizensierter) oder Opensource-Lösung, die hier zum Tragen kommt. Wer sich hier für die OpenSource-Lösung Godot entscheidet, wird mit einer voll funktionstüchtigen Gameengine belohnt. Doch selbst wenn man ein Unity-Jünger ist (oder in Zukunft gerne einer wäre) schadet es nicht, einen Blick in die Godot-Engine zu werfen. Es lohnt sich.

Links

godotengine.org
Grund-Doku auf der Hauptwebseite
Installations-Anleitung
Godot-Asset-Library
opengamesart.org

Date Created
16. Januar 2022
Author
harald