Godot 4.2 3D-Tech-Demo

Jan Meyering

Motivation

Im Game Engineering Studiengang an der TUM ist Unity eine weit verbreitete Engine (selbiges gilt für Indie Game Development). Jedoch leidet Unity unter einigen Problemen wie zum Beispiel Feature Creep welcher in langen Editor Ladezeiten, sowie verringerter Einsteigerfreundlichkeit resultiert. Des Weiteren schlechtem Cross Plattform Support des Editors (spezifisch im Bezug auf Linux). Spätestens mit Unity Technologies Ankündigung von sog. “Runtime Fees” (Gebühren für Entwickler pro Installation ihrer Spiele), ist meiner Meinung nach klar geworden, dass es notwendig ist, sich nach Alternativen zur Unity Engine umzuschauen.

Godot ist eine Cross-Platform, FOSS (Free and Open Source) Game Engine unter der MIT Lizenz welche seit 2014 Community getrieben entwickelt wird. Mit der Veröffentlichung von Godot Version 4 am 1. März 2023 hat die Engine einen großen Schritt nach vorne gemacht, insbesondere im Hinblick auf Grafikqualität in 3D Anwendungen.

Dieses Projekt ist eine Tech-Demo, welche Godot als Alternative bzw. Ersatz für Unity austesten soll, hier mit besonderem Augenmerk auf 3D-Anwendungen.

.

Allgemeine Development Erfahrung

Ein großer Vorteil des Godot Editors im Vergleich zum Unity Editor ist der signifikant bessere Cross-platform Support. Der Unity Editor ist meiner Erfahrung nach unter Linux oft sehr störrisch und manche Features wie zum Beispiel Unitys High Definition Render Pipeline (HDRP) funktionieren gar nicht.

An dieser Stelle kann sich Godots Editor positiv abheben und durch exzellenten Linux Support überzeugen. Alle drei Desktop Plattformen (Windows, Linux, Mac) unterstützen in ihrem Editor uneingeschränkt Godots vollen Funktionsumfang.

Ein weiterer erwähnenswerter Vorteil des Godot Editors ist die verhältnismäßig geringe Größe der Engine (28–189.3 Megabyte je nach Plattform) und des Editors. Folglich profitiert der Editor von verhältnismäßig kurzen Ladezeiten (für gewöhnlich wenige Sekunden zum Öffnen eines durchschnittlichen Projekts).

3D-Asset Pipeline
Godot unterstützt den Import der meisten gängigen 3D-Objekt-Dateiformate (*glTF 2.0*, *.blend*, *DAE*, *OBJ*, *FBX*). Insbesondere unterstützt Godot seit dem Release von Godot 4.X voll umfassend das glTF 2.0 Format. Dadurch ist es möglich, automatisch .blend Dateien in Godot zu importieren (Voraussetzung: Blender muss auf demselben System installiert sein). Diese nahtlose Integration hat große Vorteile für die gemeinsame Nutzung von Blender und Godot zur Entwicklung von 3D-Projekten.
Modelle können in Blender angepasst werden und Veränderungen werden automatisch in Godot geladen, ohne einen erneuten manuellen Import / Export zu benötigen. Dies ermöglicht einen schnellen iterativen Aufbau von Szenen. Somit können Levels bzw. Szenen bereits zu großen Teilen in Blender erstellt werden. Außerdem können Godot Szenen auch als *glTF* exportiert werden, wodurch es möglich wird, Godot Szenen in Blender zu importieren.
Der Native Import von Materialien und Animation wird auch unterstützt, dies hat sich in meiner persönlichen Erfahrung jedoch manchmal als ein wenig unzuverlässig erwiesen.
Um bestimmte Elemente wie zum Beispiel Lichtquellen und Kameras beim Import zu ignorieren, können diese in Blender mit dem Suffix *-noimp* benannt werden, wodurch diese von Godot ignoriert werden. Dies ist potentiell wünschenswert, da Kameras für die Zwecke von Spielen für gewöhnlich direkt von der Engine kontrolliert werden müssen. Auch Licht kann i.d.R aufgrund der unterschiedlichen Renderer zwischen Blender und Godot nicht direkt übernommen werden. Mit Hilfe des *-col* Suffix können beim Import außerdem automatisch Kollisionen generiert werden.

Tech-Demo
Die Demo enthält eine simple Indoor-Szene mit einer Fabrikhalle. Die Fabrikhalle selber wurde basierend auf dem oben erwähnten iterativen Prozess in Blender erstellt und in Godot importiert. Zur Belichtung der Halle kommen mehrere dynamische Lichtquellen zum Einsatz (Direktionale Lichtquelle, Spotlights von der Decke, Taschenlampe). Dabei kommt Godots Global Illumination System (*Signed Distance Field Global Illumination SDFG*I) zum Einsatz. Dieses interagiert auch mit dem Global Volumetric Fog in der Szene (dieser ist für die Godrays aus den Fenstern verantwortlich). Die Animation des Ventilators ist direkt aus Blender importiert.

Sonstiges / Probleme
- Im Vergleich zu Unity funktioniert Godot zuverlässig mit Git als Versionskontrollsystem.
- Godots *.exr* Importer ist leider aktuell verbuggt, wodurch diese Dateien manchmal nicht importiert werden können. Dieses Problem lässt sich lösen, indem man die Dateien als .hdr neu exportiert (zum Beispiel mit GIMP).
- Aktuelle ist die Auswahl an eingebauten Post Processing Filtern noch relativ begrenzt
- Dadurch das Debug Builds nicht im Editor laufen ist der Debug Prozess gewöhnungsbedürftig
- Gelegentliche Editor Crashes