Unity und Godot im Vergleich

Game Development Paper

Philipp Schunker
1910838010
philipp.schunker@stud.fh-campuswien.ac.at

Einleitung

Mit Unity und Godot existieren neben weiteren in dieser Arbeit nicht genannten Technologien zwei Plattformen, welche die Entwicklung von Computerspielen ermöglichen. Unity und Godot bauen grundsätzlich auf einer jeweils eigenen Sourcecode-Basis auf und werden komplett getrennt voneinander entwickelt [1] [2]. Die differente Umsetzung von Unity und Godot ergibt sich nicht zuletzt durch abweichende Komponenten und Features in mehreren Bereichen. Dabei sind diese neben den Meinungen auch von den Ideen und Plänen der Entwickler geprägt. Das Resultat sind zwei Endprodukte, denen neben bestimmten Eigenschaften auch Stärken und Schwächen beigemessen werden können. Das Ziel der vorliegenden Arbeit ist es, die grundlegenden Unterschiede zwischen den beiden genannten Technologien und Entwicklungsplattformen vorzustellen und zu beschreiben.

Historie

Unity wurde 2005 veröffentlicht ist und damit die ältere Plattform in der Gegenüberstellung zu Godot. Hergestellt und weiterentwickelt wird Unity von dem Unternehmen Unity Technologies [1]

Die Spiele-Engine Godot wurde 2014 in der Erstversion von Juan Linietsky und Ariel Menzur veröffentlicht. Die Entwicklung erfolgt im Rahmen eines Open-Source-Projektes, weswegen der Quellcode unter https://github.com/godotengine/godot für die Allgemeinheit einsehbar ist. Die Anpassung und Weiterentwicklung wird von der Community übernommen [2] [3].

Anforderungen

Unity als Entwicklungsplattform kann auf den Betriebssystemen Windows, macOS und Linux eingesetzt werden [4] [4].

Godot ist ähnlich zu Unity auf den Plattformen Windows, macOS und Linux lauffähig. Mit der eingebauten Unterstützung des X11 Window-Systems kann Godot zudem auf anderen unixoiden Betriebssystem wie den diversen Varianten von BSD ausgeführt werden [2] [4].

Zielplattformen

Über Unity können Spiele für die bereits genannten Computer-Betriebssysteme exportiert werden. Des Weiteren können die mit Unity entwickelten Spiele für Konsolensysteme wie die PlayStation 4 von Sony, die Xbox One von Microsoft und die Nintendo Switch exportiert werden. Auch mobile Plattformen und Smartphones mit den etablierten Betriebssystemen Android und iOS sowie die weniger verbreiteten Systeme Windows Phone und Tizen werden unterstützt. Auch TV-Systeme mit Android TV, Samsung SMART TV und tvOS werden von Unity unterstützt. Der Support von Unity deckt darüber hinaus auch die VR-Systeme Oculus Rift, SteamVR, Google Cardbord, Playstation VR, Gear VR und HoloLens ab. Unity-Projekte können auch in WebAssembly-Bytecode kompiliert und über den Browser ausgeführt werden [1] [4].

Im Vergleich zu Unity werden von Godot weniger verschiedene Zielplattformen und Systeme unterstützt. Wie bei Unity können die mit Godot erstellten Projekte auch für die Entwicklungsplattformen exportiert werden. Auf diesen Plattformen kann auch auf den VR-Systemen Oculus Rift und SteamVR aufgesetzt werden. Zudem können Spiele für die mobilen Plattformen Android und iOS entwickelt werden. Weniger verbreitete Smartphone-Betriebssysteme wie Windows Phone und Tizen werden nicht unterstützt [2] [4]. Da Windows Phone allerdings nicht mehr weiterentwickelt wird und der Marktanteil beider Systeme klein ist, ist der fehlende Support

von Godot in technischer und wirtschaftlicher Hinsicht nachvollziehbar. Die Unterstützung von momentan verfügbaren Konsolensystemen ist ohne zusätzliche Software von Drittanbietern nicht gegeben. Dieser Umstand ist auf mehrere Gründe zurückzuführen. Mitunter ist der rechtliche Aspekt ein wesentlicher Faktor dafür, dass Konsolen von Godot ohne zusätzliche Software nicht unterstützt werden. Bei Godot handelt es sich um ein Open-Source-Projekt und die Entwicklung von für Konsolen ist hauptsächlich für lizenzierte Unternehmen erlaubt. Außerdem sind die Software Development Kits (SDK) der etablierten Konsolensysteme proprietär. Die SDKs der etablierten Konsolensysteme sind daher rechtlich geschützt und mit dem Grundgedanken, der Lizenz sowie dem offenen Quellcode von Godot nicht kompatibel [5]. Jedoch ist auch mit Godot eine Kompilierung in WebAssembly-Bytecode und eine Ausführung über Webbrowser möglich, die den WebAssembly-Standard unterstützen [2] [4].

Zielgruppen

Unity hat sich seit der Vorstellung als eine umfangreiche und mächtige Plattform für die Spieleentwicklung bewahrheitet. Zu Beginn war Unity noch für simple Spiele von geringer Komplexität geeignet. Im Laufe der Zeit wurde Unity jedoch verbessert und größere Computerspiele konnten mit Unity und weiteren Features umgesetzt werden. Mit den hinzugefügten Komponenten und Werkzeugen ist zudem die Modellierung von 3D-Modellen möglich, weswegen Unity auch in der Filmproduktion oder anderen Einsatzgebieten Verwendung findet [3].

Mit Unity wurde eine Vielzahl von Spielen in verschiedenen Genres erstellt, darunter bekannte 2D-Spiele wie Among Us und Crossy Road. Ebenso wird Unity erfolgreich für die Produktion von 3D-Spielen wie Firewatch und Praey for the Gods verwendet. Auch etablierte Spielehersteller greifen auf Unity zurück. Von Nintendo wurden Super Mario Run und Mario Kart Tour mit Unity umgesetzt [6].

Die Godot-Engine wird mehrheitlich von einzelnen Spieleentwicklern oder kleineren Spielherstellern eingesetzt. Dies kann auf den kleineren Umfang von Entwicklerschnittstellen und weniger maßgebenden Optionen im 3D-Bereich zurückgeführt werden. Ungeachtet dessen können die mit Godot produzierten Spiele aber einen wichtigen Zwischenschritt für das Verständnis von anderen Plattformen und eine weitere Karriere auf dem Gebiet der Computerspiele darstellen [3]. Zudem wird Godot von mehreren versierten Entwicklern für die Entwicklung von Spielen gegenüber anderen Entwicklungsplattformen bevorzugt [7] [8]. Unter diesen Aspekten existiert eine beachtenswerte Menge von Spielen, deren Funktionalität auf der Godot-Engine aufbaut [9].

Editor

Sowohl Unity als auch Godot stellen dezidierte Editoren mit graphischen Benutzeroberflächen (GUI) als zentrale Programme für die Erstellung von Spielen bereit. Die Editor-Programme beider Plattformen werden für die Organisation und Bearbeitung von Projekten genutzt. Unter diesem Aspekt werden über den Editor die Dateien und Assets verwaltet und verändert, aus denen das Spiel besteht. Beide Editoren bieten darüber hinaus ein umfangreiches Set von Werkzeugen für die Entwicklung von Spielen.

Der Editor von Unity ist mit einer graphischen Oberfläche ausgestattet, welche die wichtigsten Komponenten in getrennten Bereichen darstellt. Die grafischen Gestaltungselemente und Bereiche können über einen integrierten Layoutmanager verschoben und positioniert werden [Abbildung 1].

Im Hierarchie-Panel werden die Szenen für ein Spiel hierarchisch angezeigt und dargestellt. Die Szenen in Unity fassen einen zwei- oder dreidimensionalen Raum und Spielobjekte zusammen und können selbst als essentielle Bestandteile eines Spiels angesehen werden [10].

In der Projektansicht werden die Ordner und Dateien visualisiert, aus denen das Spiel besteht. In Unity werden die Dateien in der gängigen Praxis in Ordner entsprechend der Dateitypen abgelegt. Bekannte Ordner sind beispielsweise Assets, Editor, Scenes, Scripts, Prefabs und Materials [11]. Der Ordner Assets ist das Hauptverzeichnis, in dem die Bestandteile eines Projektes abgelegt werden. Unter Editor werden Dateien abgelegt, welche zur Erweiterung des Editor-Programms mit zusätzlichen und angepassten Panels eingesetzt werden. In Scenes werden die Szenenobjekte eines Spieles gespeichert. Im Ordner Scripts werden Dateien eingeordnet, die Quellcode beinhalten. Scripts sind deswegen Dateien mit einer wesentlichen Bedeutung in jedem mit Unity erstellten Spiel. In Prefabs werden Spielobjekte gespeichert, welche als Vorlage für weitere Instanzen von Objekten in einer Szene dienen [12]. In Materials werden Dateien abgelegt, die für die Erscheinung von Szenen und Objekten verantwortlich sind [13]. Unity stellt mit Physic Material eine spezielle Art von Material zur Verfügung, mit dem die Eigenschaften und das Verhalten von kollidierenden Objekten bestimmt werden können [14].

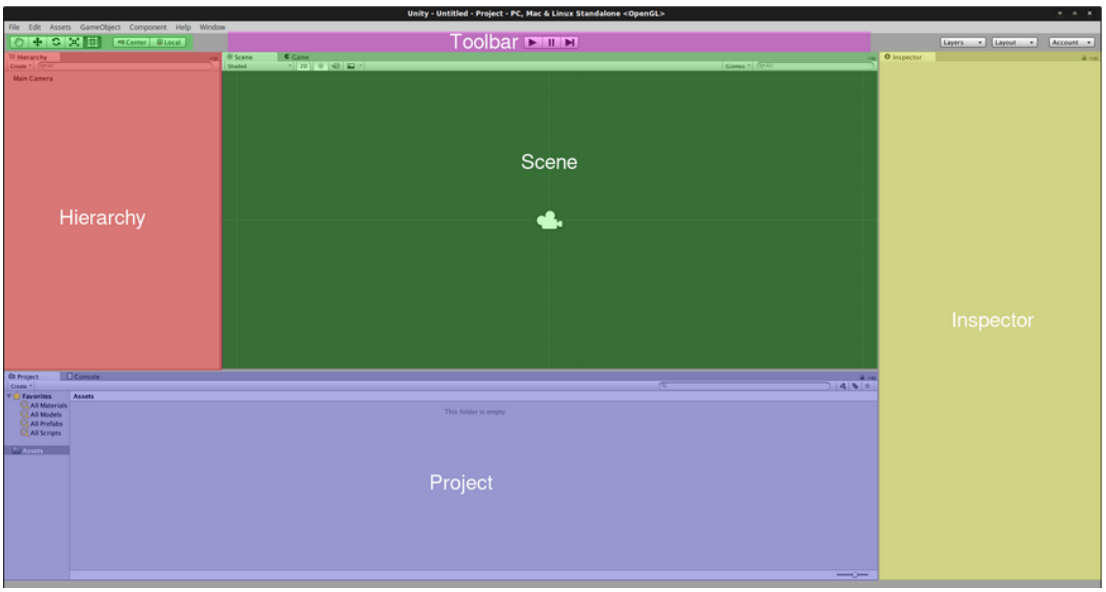

Abbildung 1: GUI Unity [4]

Im Szenen-Bereich wird die Ansicht auf die Szene angezeigt. Über die Maus und Tastatur können die Position und der Blickwinkel auf die Szene und Objekte verändert werden. Die Szenenansicht ist daher eine zentrale Komponente in der graphischen Oberfläche, weil diese für die Visualisierung der Spielobjekte hauptverantwortlich ist [15].

Das Inspektor-Fenster zeigt zusätzliche Informationen für das ausgewählte Spielobjekt und die Elemente einer Szene. Über den Inspektor können auch die Eigenschaften der Spielobjekte verändert und die Komponenten von Objekten zugewiesen werden. Der Zugriff des Inspektors auf die Membervariablen der Objekte wird über die Zugriffsmodifizierer geregelt [16] [17].

Die Toolbar befindet sich am oberen Rand des Programmfensters direkt unter dem Menüband. Über die in der Toolbar angezeigten Elemente können weitere Elemente des Editors wie die Szenenansicht und Spielansicht sowie zusätzliche Programm-Fenster und Menüs angesteuert

werden. Sie ist der einzige Bestandteil des Editors, welcher nicht beliebig positioniert werden kann [18].

Der Aufbau des Editors von Godot ist weitgehend ähnlich zu dem von Unity. Auch dieser kann in mehrere Hauptbereiche eingeteilt werden [Abbildung 2].

Für das Dateisystem ist ein separater Bereich vorgesehen, in dem die Projektdateien und Assets angezeigt und verwaltet werden können. Die Projektverwaltung ist in Godot über eine einzige Datei gelöst. Die Dateien beinhalten grundsätzlich Text in einem simplen Textformat, wodurch die Versionskontrolle der Projekte und Dateien besser unterstützt und vereinfacht wird [4] [19].

Im Szenen-Baum-Panel werden die Bearbeitung der Szenen und die darin enthaltenen Objekte ermöglicht. Anders als in Unity können Szenen aus weiteren untergeordneten Szenen bestehen, wodurch ein hierarchischer Baum aus Szenen erstellt werden kann. Angesichts dieser Tatsache werden die Szenen in Godot auch als Knoten. Die Knoten können genutzt werden, um Spielobjekte logisch zusammenzufassen [4] [19].

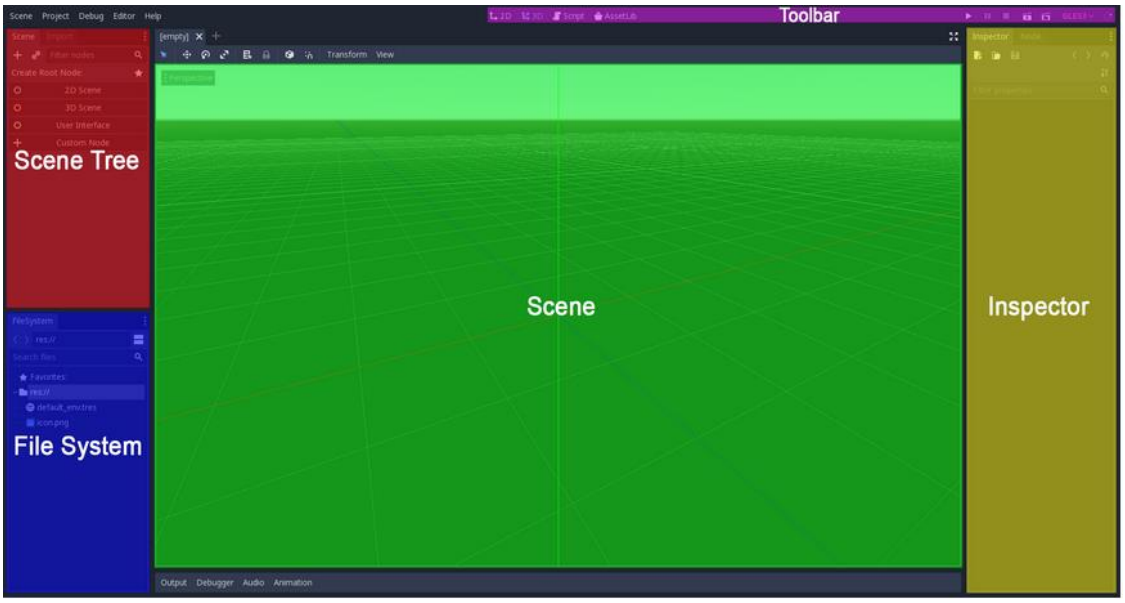

Abbildung 2: GUI Godot [4]

Über den Hauptbereich in der Mitte des Godot-Editors werden, wie im Editor von Unity, die Szene und die darin enthaltenen Objekte visualisiert. Der Hauptarbeitsbereich kann neben der Szenenansicht, und im Unterschied zu dem Unity-Editor, auch als Arbeitsbereich für die Programmierung von Skripten verwendet werden. Die Bearbeitung von Scripts und Quellcode des Projektes kann daher direkt im Editor von Godot vorgenommen werden [17] [19].

Die Inspektor-Ansicht erlaubt die Bearbeitung der Szeneneigenschaften. Wie in Unity können Attribute von Spielobjekte über den Inspektor angezeigt und bearbeitet werden. Im Unterschied zu Unity müssen die Attribute der Objekte explizit mit dem Schlüsselwort [Export] deklariert werden, um diese für den Inspektor sichtbar und zugreifbar zu machen [17].

Am oberen Rand des Godot-Editors in direkter Nähe zur Menüleiste befindet sich die Toolbar. Die Toolbar ermöglicht das Umschalten der Arbeitsbereiche in mehrere Typen über die von ihr dargestellten Schaltflächen. Neben den Arbeitsbereichen für 2D und 3D kann der Arbeitsbereich für die Bearbeitung von Skripten (Scripts) und die Auswahl von Assets (AssetLib) angezeigt werden [19].

Die Ausführung von Spielen erfolgt über den Godot-Editor in einem separaten Fenster. Das Starten und Beenden von Projekten soll schneller sein als in Unity, weil das Projekt nicht zusätzlich gespeichert und neugeladen werden muss. Zudem ermöglicht der integrierte Code-Editor ein Editieren des Spiels zur Laufzeit, wodurch es möglich ist, das Spiel während des Spielens zu erweitern und der Arbeitsfluss nicht unterbrochen werden muss. Um die Stabilität und Performanz des Editors zu verbessern, wird das Spiel in einem zusätzlichen Prozess ausgeführt [4].

Scripting

Der Bereich Scripting ist ein wesentlicher Bestandteil von Unity und Godot. In beiden Entwicklungsplattformen sind Scripts für die Funktionalität der erstellten Spiele verantwortlich, weil in diesen der Quellcode definiert und die Logik programmiert ist. Mittels der Scripts werden letztendlich das Verhalten und die Eigenschaften aller Objekte eines Spiels programmatisch implementiert. Zu diesem Zweck können die Application Programming Interfaces (APIs) der jeweiligen Plattform verwendet und angesprochen werden [17].

Die primäre Programmiersprache für Scripts in Unity ist C#. C# ist eine typsichere sowie objektorientierte Programmiersprache und wird neben Unity in vielen weiteren Bereichen eingesetzt. Die Programmierung von C#-Code für Unity wird in einer dafür geeigneten integrierten Entwicklungsumgebung (IDE) und externen Programm vorgenommen [4].

In Unity ist es grundsätzlich möglich einem Spielobjekt (GameObject) mehrere Scripts zuzuweisen. Für gewöhnlich wird in Unity in den Scripts das Verhalten der Spielobjekte implementiert. Die Scripts müssen dazu von der Basisklasse MonoBehaviour erben [20]. Zur Laufzeit werden mit den in Scripts definierten Klassen schließlich konkrete Objekte der Basisklasse MonoBehaviour instanziiert. Über die Vererbungshierarchie ist jedes MonoBehaviour-Objekt auch eine Instanz der Klasse Component. Ein Objekt der Component-Klasse enthält mit der Membervariable gameObject eine Referenz zu einem GameObject, zu welchem die Komponente angehängt wurde. Bei dem Hinzufügen eines Scripts zu einem GameObject erfolgt die entsprechende Zuweisung der Referenz [4]. In der Scripting API von Unity ist der Einsatz von Komposition daher von großer Bedeutung. Auch in anderen Fällen werden die Interaktion und der Zugriff auf andere Objekte über Objekt-Aggregation oder -Komposition gelöst. Die Zuweisung von Objekten über Membervariablen kann im Programmcode oder über den Editor und Inspector vorgenommen werden.

In Godot werden mehrere Programmiersprachen unterstützt. Neben der Möglichkeit C# als Programmiersprache zu verwenden, ist die Verwendung der eigenen Programmiersprache GDScript möglich. Im Vergleich zu anderen Skripting- und Programmiersprachen wurde diese explizit für die Godot-Engine entwickelt und den Einsatz für Spiele optimiert [21]. Unter diesem Aspekt wird eine saubere Codebasis und Implementierung des Spiels gefördert [22]. Als Vorteile von GDScript werden die Unterstützung von Multi-Threading, eingebaute Vektor-Datentypen, eine optimierte Garbage-Collection und eine verbesserte Performanz in Bezug auf das Programmiermodel von Godot genannt [7] [8]. Weiters wird GDScript als leicht zu erlernende Programmiersprache bezeichnet, wodurch die Einarbeitungszeit für Entwickler und gegebenenfalls auch Entwicklungsdauer für Neueinsteiger verkürzt werden kann [3] [8] [23]. Darüber hinaus können Spiele mit Godot auch über GDNative sowie in C++ programmiert werden. Die APIs von Godot können hierzu direkt in C++ genutzt werden [4] [24] In Unity wird die Nutzung von C++ nicht explizit ausgeschlossen, offiziell wird jedoch ausschließlich auf C# hingewiesen [25] [26].

Das Äquivalent zum MonoBehaviour-Objekt und GameObject ist in Godot ein Objekt der Klasse Node. Im Gegensatz zu Unity ist es in Godot nicht möglich, einem Node-Objekt mehrere Scripts

zuzuweisen. In Godot wird das Spiel über einen Baum von Knoten aufgebaut [Abbildung 3]. Für den Einsatz von mehreren Skripten gibt es demnach zwei Lösungen. Eine Option wäre, einen zusätzlich Knoten zwischen dem Elternknoten und dem Zielknoten im Objektbaum hinzufügen. Anderenfalls kann der Zielknoten in zwei Kindknoten aufgetrennt werden [4]. Mit dem Konzept dieses Szenenbaums in Godot wird die Wiederverwendbarkeit von Objekten erhöht. Außerdem können Objekte orchestriert und Szenen feingranular kombiniert werden. Die Notwendigkeit für dezidierte Objekt-Vorlagen (Prefabs) ist mit dieser Technologie und Arbeitsweise nicht gegeben [3] [4].

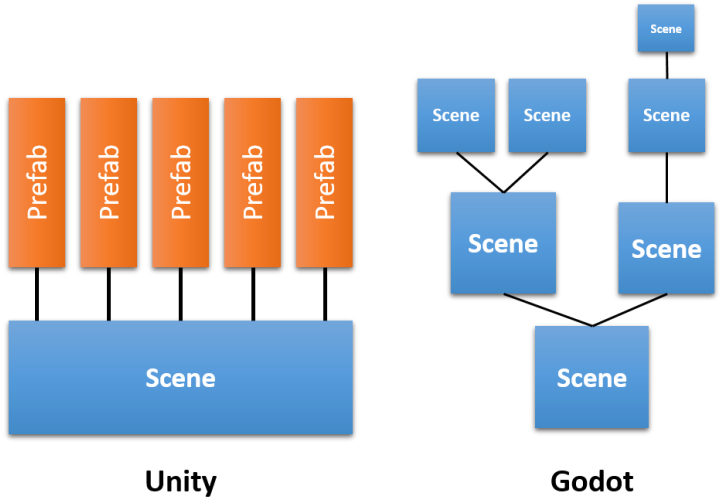

Abbildung 3: Scene System [7]

Trotz der Unterschiede und verschiedenen Namen in den APIs von Unity und Godot bieten viele Schnittstellen die gleiche oder eine ähnliche Funktionalität [Tabelle 1]. Wie auch Unity stellt Godot samt der-Engine ein umfangreiches SDK bereit, mit dem sich insgesamt betrachtet vergleichbare Ergebnisse erzielen lassen [17].

Unity	Godot
MonoBehaviour	Node
Awake()	_init()
Start()	_ready()
Update()	_process(float delta)
Time.deltaTime	Node.get_process_delta_time()
Mathf.Lerp(float a, float b, float t)	(1 - t) * a + t * b lerp(Variant a, Variant b, float t)
Vector3.Lerp(startPoint, endPoint, float t)	vector3.linear_interpolate(Vector3 b, float t)
Debug.Log()	GD.print()

Tabelle 1: Unity und Godot APIs

Community und Erweiterungen

Aufgrund seiner Etablierung auf dem Markt und der ausgereiften Technologie ist Unity eine Entwicklungsplattform mit einer großen und breiten Benutzergruppe. Unity wird von einzelnen Entwicklern bin hin zu erfolgreichen Spieleherstellern genutzt [6] [17]. Zugleich bestehen Webseiten und Foren, die ausschließlich für den Informationsaustausch für Unity verwendet werden [27].

Mit dem Asset Store für Unity existiert eine Plattform die viele verschiedene Projekte und Produktionen von anderen Entwicklern beinhaltet. Über den Asset Store von Unity können unter anderem vorgefertigte 2D- und 3D-Objekte, Texturen, Materialen und ebenso Animationen bezogen werden. Des Weiteren befinden sich im diesem Store Assets wie spezielle Werkzeuge und zusätzlichen Scripts, mit denen die Entwicklung des eigenen Spiels und von Spielobjekten unterstützt wird [28].

Die Community von Godot wird von vielen Seiten als ein großer Vorteil beschrieben und positiv hervorgehoben [3] [8] [23]. Auch werden im Entwicklungsprozess von Godot Anforderungen aus der Gemeinschaft berücksichtigt und umgesetzt. In diesem Kontext besteht eine hohe Wahrscheinlichkeit für eine Implementierung einer guten Idee und neuen Funktionen durch einen anderen Entwickler in der Community. In gleicher Weise werden können etwaige Fehler im Rahmen des Projektes durch mehrere Entwickler behoben werden [8] [23].

Für die Godot-Engine besteht mit der Asset Library eine umfangreiche Bibliothek von Erweiterungen und Komponenten, die für die Entwicklung eines Spiels genutzt werden können. Verfügbar sind beispielsweise zusätzliche Tools für die Entwicklung von 2D- und 3D-Spielen, Shaders und Materials. Auch vorgefertigte Scripts, Templates sowie komplette Projekte und Demospiele sind verfügbar [29].

Kosten und Lizenz

Die Entwicklungsplattform Unity, welche die Engine, den Unity-Editor und das SDK umfasst, ist ein proprietäres Produkt. Die Benutzung von Unity erfolgt unter dem Einverständnis eines Endbenutzer-Lizenzvertrags (EULA). Sofern die Einnahmen mit einem Spiel, welches mit Unity entwickelt worden ist, über ein Jahr betrachtet $100.000 übersteigen, betragen die Kosten für die weitere Benutzung von Unity Plus pro Person und Jahr $399. Bei Einnahmen von über $200.000 pro Jahr muss Unity Pro eingesetzt werden. Die Kosten für Unity Pro belaufen sich auf $1800 pro Entwickler für ein Jahr. Für Unity Pro besteht keine Grenze für Einnahmen [1] [30].

Godot einschließlich der Engine, dem Editor und dem SDK ist komplett kostenlos nutzbar. Das Godot-Projekt wird im Rahmen eines Open-Source-Projektes öffentlich von der Community weiterentwickelt und unter der MIT-Lizenz bereitgestellt [31]. Bei potenziellen Einnahmen durch mit Godot entwickelte Spiele entstehen keine Kosten und monetären Pflichten. An das Projekt müssen auch keine Gebühren abgeführt werden und die Einnahmen stehen daher uneingeschränkt dem Entwickler des Spiels zu [32].

Konklusion

Beide Plattformen, Unity und Godot, sind zwei mächtige und umfangreiche Spiele-Engines für die Entwicklung von Computerspielen. Die von Unity und Godot zur Verfügung gestellten Editoren sind in vielen Bereichen ähnlich. Die Differenzen bei den Editoren können aus meiner Sicht bei der Wahl der Plattform vernachlässigt werden. In gleicher Weise kann Unity und Godot auf den weitverbreitetsten Plattformen ausgeführt werden. Dennoch gibt es größere Unterschiede technischer Natur, angefangen bei den Zielplattformen für die entwickelt werden kann. Der große Gegensatz auf diesem Gebiet ist die Möglichkeit mit Unity Spiele für Konsolensysteme zu entwickeln. Des Weiteren können die Plattformen durch die unterstützten Programmiersprachen abgegrenzt werden. Spiele werden mit Unity primär in der Sprache C# programmiert. Mehr Auswahlmöglichkeiten werden andererseits von Godot angeboten. Mit Godot kann ein Spiel in GDScript, C# sowie in C++ und GDNative entwickelt werden. Abweichungen können auch bei dem Design der Softwarearchitektur und der Umsetzung der APIs festgestellt werden. Welcher Ansatz und welche Umsetzung moderner oder besser sind, lässt sich jedoch ohne einer tiefergehenden

Prüfung und empirischen Analyse nicht genau beurteilen. Für beide Plattformen bestehen aktive und große Benutzergemeinschaften. Genügend Ressourcen und Hilfestellungen gibt es sowohl für Unity als auch für Godot. Die höhere Quantität und Qualität der verfügbaren Assets kann indessen Unity zugerechnet werden. Der Großteil von Assets für Godot ist hingegen kostenlos und kann frei verwendet werden. Ein entscheidender Unterschied könnte die Lizenzen von Unity und Godot darstellen. Bei der Verwendung von Unity entstehen gegebenenfalls höhere Kosten. Für Godot ist möglicherweise der Grundgedanke von Open-Source der ausschlaggebende Grund. Beide Plattformen und Technologien sind für die Entwicklung von Spielen mehr als gut geeignet. Die Entscheidung für eine der beiden diskutierten Plattformen wird letztendlich in vielen Fällen auf Basis einer subjektiven Meinung getroffen, welche sich eine jede Entwicklerin und ein jeder Entwickler auf eine andere Art und Weise bilden.

Literaturverzeichnis

[1] Wikipedia, „Unity Spiele-Engine," 2020. [Online]. Available: https://de.wikipedia.org/wiki/Unity_(Spiel-Engine). [Zugriff am 9 Januar 2021].

[2] Wikipedia, „Godot Engine," 2020. [Online]. Available: https://de.wikipedia.org/wiki/Godot_Engine. [Zugriff am 9 Januar 2021].

[3] Bryan W., „GameDesigning," 20 Dezember 2020. [Online]. Available: https://www.gamedesigning.org/engines/unity-vs-godot/. [Zugriff am 9 Januar 2021].

[4] Godot Docs, „From Unity to Godot Engine," 2020. [Online]. Available: https://docs.godotengine.org/en/stable/getting_started/editor/unity_to_godot.html. [Zugriff am 9 Januar 2021].

[5] Godot Docs, „Console support in Godot," 2020. [Online]. [Zugriff am 10 Januar 2020].

[6] Wikipedia, „List of Unity games," Januar 2021. [Online]. Available: https://en.wikipedia.org/wiki/List_of_Unity_games. [Zugriff am 10 Januar 2021].

[7] R. Milk, „Rock Milk Articles," 26 September 2017. [Online]. Available: https://medium.com/rock-milk/why-godot-engine-e0d4736d6eb0. [Zugriff am 9 01 2021].

[8] P. Friedland, 23 März 2020. [Online]. Available: https://t2informatik.de/blog/softwareentwicklung/warum-ich-bei-godot-gelandet-bin/. [Zugriff am 9 Januar 2020].

[9] Godot Engine, „Showcase," 2020. [Online]. Available: https://godotengine.org/showcase. [Zugriff am 10 Januar 2021].

[10] Unity Documentation, „Scenes," 2020. [Online]. Available: https://docs.unity3d.com/Manual/CreatingScenes.html. [Zugriff am 11 Januar 2021].

[11] Unity Documentation, „Special Folder Names," 2020. [Online]. Available: https://docs.unity3d.com/Manual/SpecialFolders.html. [Zugriff am 11 Januar 2021].

[12] Unity Documentation, „Prefabs," 2020. [Online]. Available: https://docs.unity3d.com/Manual/Prefabs.html. [Zugriff am 11 Januar 2021].

[13] Unity Documentation, „Materials,“ 2020. [Online]. Available: https://docs.unity3d.com/Manual/Materials.html. [Zugriff am 11 Januar 2021].

[14] Unity Documentation, „Physic Material,“ 2020. [Online]. Available: https://docs.unity3d.com/Manual/class-PhysicMaterial.html. [Zugriff am 11 Januar 2021].

[15] Unity Documentation, „Scene View,“ 2020. [Online]. Available: https://docs.unity3d.com/Manual/UsingTheSceneView.html. [Zugriff am 11 Januar 2021].

[16] Unity Documentation, „Inspector Window,“ 2020. [Online]. Available: https://docs.unity3d.com/Manual/UsingTheInspector.html. [Zugriff am 11 Januar 2021].

[17] A. Thorn, Moving from Unity to Godot: An In-Depth Handbook to Godot for Unity Users, Apress, 2020.

[18] Unity Documentation, „Toolbar,“ 2020, [Online]. Available: https://docs.unity3d.com/Manual/Toolbar.html. [Zugriff am 11 Januar 2021].

[19] Godot Docs, „Introduction to Godot's Editor,“ 2020. [Online]. Available: https://docs.godotengine.org/en/stable/getting_started/step_by_step/intro_to_the_editor_interface.html. [Zugriff am 11 Januar 2021].

[20] Unity Documentation, „MonoBehaviour,“ 2020. [Online]. Available: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html. [Zugriff am 11 Januar 2021].

[21] Godot Docs, „GDScript Basics,“ 2020. [Online]. Available: https://docs.godotengine.org/en/stable/getting_started/scripting/gdscript/gdscript_basics.html. [Zugriff am 9 Januar 2021].

[22] Godot Engine Q&A, „Well, I'm leaving Godot for some time,“ 11 Juni 2016. [Online]. Available: https://godotengine.org/qa/5142/well-im-leaving-godot-for-some-time. [Zugriff am 9 Januar 2021].

[23] Godot Engine Q&A, „Godot vs Unity in 2019,“ 22 März 2019. [Online]. Available: https://godotengine.org/qa/42280/godot-vs-unity-in-2019. [Zugriff am 9 Januar 2021].

[24] Godot Docs, „Introduction to Godot development,“ 2020. [Online]. Available: https://docs.godotengine.org/en/stable/development/cpp/introduction_to_godot_development.html#doc-introduction-to-godot-development. [Zugriff am 12 Januar 2021].

[25] Unity How-To, „Skripting in Unity für erfahrene Programmierer,“ 2020. [Online]. Available: https://unity.com/de/how-to/programming-unity. [Zugriff am 12 Januar 2021].

[26] Godot Engine Q&A, „Coming from Unity (and Unreal) - is Godot for me?,“ 8 November 2017. [Online]. Available: https://godotengine.org/qa/19672/coming-from-unity-and-unreal-is-godot-for-me. [Zugriff am 9 Januar 2020].

[27] Unity, „Forum,“ 2021. [Online]. Available: https://forum.unity.com/. [Zugriff am 12 Januar 2021].

[28] Unity, „Asset Store,“ 2021. [Online]. Available: https://assetstore.unity.com/. [Zugriff am 12 Januar 2020].

[29] Godot Engine, „Asset Library,“ 2021. [Online]. Available: https://godotengine.org/asset-library/asset. [Zugriff am 12 Januar 2021].

[30] Unity, „FAQ: Licensing & Activation,“ [Online]. Available: https://unity3d.com/unity/faq/2491. [Zugriff am 12 Januar 2021].

[31] Godot Engine, „Godot,“ 2021. [Online]. Available: https://github.com/godotengine/godot. [Zugriff am 12 Januar 2021].

[32] Godot Engine, „License,“ [Online]. Available: https://godotengine.org/license. [Zugriff am 12 Januar 2021].