Die Zukunft der Spieleentwicklung

Description

Der sogenannte „Unity-Skandal“ hat gezeigt, wie schnell aus der Entwicklungsumgebung des Vertrauens eine *Engine non grata* werden kann. Zwar ist Unity Technologies dabei, Weltrekorde in Rückwärtsrudern aufzustellen, aber dennoch bleibt die Frage, auf welche Engine man in Zukunft noch setzen kann.

Der nachfolgende Artikel ist eine sehr persönliche Sichtweise und geht auf die Erfahrungen des Autors ein. Wer anderer Meinung ist oder konstruktive Vorschläge hat, kann dies gerne als Kommentar schreiben. Wir freuen uns auf Diskussionen.

Geschichten von früher

Als ich 1993 mit meinem ersten Spielprojekt anfing, hatte ich nur *QBasic*. Eine simple, in seiner Ausführungsgeschwindigkeit stark begrenzte Programmiersprache, ohne die Möglichkeit, eine ausführbare Datei erstellen zu können. Game-Engines, wie man sie heute kennt, gab es nicht. Wenn man sich eine Engine besorgen wollte, konnte man sie von professionellen Entwicklern meist für sechs bis siebenstellige Beträge kaufen.

Wir hatten nur die Programmiersprache. Andere verwendeten *Assembler*, *C*, *C++* oder *Pascal*. Ich hatte vorerst nur mein *QBasic*. Um ein halbwegs professionelles Spiel zu entwickeln – davon war ich übrigens Lichtjahre entfernt – musste man sich selbst eine Engine schreiben. Zu DOS-Zeiten bedeutete dies zum Beispiel, dass man Grafikkarten und Soundkarten selbst ansprechen musste. Abgesehen von Treibern, von denen es tausende gab, existierte keine vernünftige Schnittstelle. Das heißt: Man musste schon sehr gut programmieren können und viel Zeit investieren, um etwas Vernünftiges auf den Bildschirm zu zaubern.

Die Gegenwart

2023, also dreißig Jahre nach meinen ersten Gehversuchen in *QBasic*, sieht es ganz anders aus. Es scheint, als gäbe es nichts Einfacheres, als Spiele zu entwickeln. Engines gibt es wie Sand am Meer

und es ist für jeden was dabei. Egal ob man am Wochenende ein kleines 2D-Spiel basteln oder das nächste 3D MMORPG machen möchte. Die meisten Engines sind sogar kostenlos. Sie leben entweder komplett von Spenden oder es gibt ein Lizenzmodell, welches erst ab einer bestimmten Verkaufszahl oder Umsatzzahl greift. Außerdem gibt es Abo-Modelle, etwa beim GameMaker (hat sich am 21. November 2023 etwas geändert).

Am fairsten finde ich die Beteiligung am Umsatz. Das bedeutet automatisch, dass die Engine frei ist, wenn man nur Freeware erstellt. Und ab einem bestimmten Einkommen gibt man ein paar Prozent ab.

Allerdings leben die allermeisten Engineentwickler nicht nur von Luft und Liebe, weshalb sich Lizenzen ändern können. Genau das ist mit Unity geschehen. Ab dem 1. Januar 2024 sollte sich alles ändern, auch rückwirkend. Das recht wirre, undurchsichtige und schlecht kommunizierte Modell löste einen Shitstorm aus. Innerhalb weniger Tage ruinierte Unity den über fast zwanzig Jahre aufgebauten guten Ruf. Zwar kassierte das US-Amerikanische Unternehmen Unity Technologies seine Pläne wieder ein und versprach, mit der Community zu reden, aber der Schaden war da und das Vertrauen weg. Leider ist das kein Einzelfall.

Rechnungen, die bezahlt werden müssen

Eine professionelle Game-Engine zu entwickeln ist heute wesentlich aufwändiger als vor dreißig Jahren. Die Ansprüche sind enorm gestiegen. Damals entwickelte man normalerweise eine Engine für eine Plattform, vorwiegend dem MS-DOS-PC, und für ein Genre. Heutige Engines sind idealerweise so ausgelegt, dass man jedes Genre umsetzen kann, egal ob 2D oder 3D. Außerdem soll das Spiel bequem für jede andere Plattform und Betriebssystem portiert werden können. Egal ob Windows, MacOS, Linux, Browser oder eine Konsole. Wie reibungslos das in der Praxis funktioniert, steht auf einem anderen Blatt.

Hinzu kommt die Benutzeroberfläche, Support, digitales Handbuch, Betreuung einer Community und viele andere Aufgaben. Die großen Engines, allen voran Unity und Unreal 3D, brauchen zahlreiche Mitarbeiter, die bezahlt werden müssen.

Der Gegenentwurf heißt Open Source. Meistens beginnt es damit, dass ein findiger Programmierer eine gute Idee hat. Er startet mit der Entwicklung, steckt viel Arbeit in sein Projekt, veröffentlicht den Code und mit viel Glück finden sich andere Entwickler, die das Projekt unterstützen. Das beste Beispiel in der Spieleentwicklung ist aktuell die Godot Engine. Momentan gibt es sie mit den Versionen 3.5 und 4.1 in zwei Varianten. Die Community wird immer größer, die Engine immer reifer. Man kann für das Projekt spenden, ähnlich wie bei Blender.

Wie groß der Qualitative Unterschied zwischen Unity und Godot ist, darüber lässt sich streiten. Und auch Open Source ist nicht zwangsläufig eine heile Welt, die für die Ewigkeit geschaffen wurde. Von solchen Gedanken muss man sich in der Softwareentwicklung schnell verabschieden.

Bei einem professionellen Unternehmen ist es also durchaus legitim, Geld verdienen zu wollen oder gar zu müssen. So weit man gehört hat, waren die Zahlen bei Unity nicht mehr gut, also wollte man aus einer finanziellen Notwendigkeit heraus das Lizenzmodell ändern.

Unerwünschte Änderungen

Der Vorfall mit Unity weckte bei mir gleich mehrere Traumata. Zunächst war da 3D Gamestudio. Vor vielen Jahren beschäftigte ich mich einige Monate damit. Obwohl die Engine damals schon nicht state of the art war, fand ich ein paar Aspekte durchaus interessant. Ich kaufte mir die Pro-Lizenz und ein Buch, nur um kurz darauf festzustellen, dass sie nicht mehr weiterentwickelt wird. Schlechtes Timing.

Meine Leidensgeschichte mit GameMaker geht etwas länger. Ich mochte immer die grundlegende Logik dahinter. Das ist bis heute der Grund, warum ich es noch nutze und nicht auf Godot umgestiegen bin. Doch seit der Version 2 gibt es viele Dinge, die stören. Zunächst gab es einen Haufen Bugs, was bei einer Neuentwicklung passieren kann. Problematisch ist dies aber in Kombination mit einem schlechten Support.

Irgendwann hat der Entwickler das Lizenz-Modell umgestellt. Früher kaufte man es einmal und konnte weitere Exportmöglichkeiten dazukaufen. Ich entschied mich damals für den zusätzlichen HTML5-Export. Heute gibt es Abo-Modelle, wo man monatlich Geld überweisen muss. Für professionelle Entwickler ist das sicher kein Problem, für Hobbyentwickler ist das schwierig (Hat sich nach Jahren wieder geändert. Siehe oben). Um es zu verkomplizieren, werden vier Lizenz-Modelle angeboten. Und ja, es gibt auch eine freie Version. Mit der kann man die Spiele für das meiner Meinung nach recht überflüssige GX.games exportieren. Ohnehin wird man seit Neustem im GameMaker damit gegängelt, sich einen Opera-Account zuzulegen. *Diese Aussagen betreffen den Stand Oktober 2023.*

Zu allem Überfluss kommt die fragwürdige Updatephilosophie hinzu. Es kommen regelmäßig Updates, was eine gute Sache ist. Ein Großteil des Updates besteht aus Fehlerbereinigung. Ein kleiner Teil aus neuen Features. Das ist soweit eine feine Sache. Aber hinzu kommt, dass sich auch die Programmiersprache weiterentwickelt. Wenn neue Features in der Sprache eingeführt werden, ist das allgemein kein Problem, aber wenn alte Sachen gestrichen werden, ist das ein großes Problem. So kann es passieren, dass man sein Projekt immer wieder umschreiben muss, weil verwendete Befehle nicht mehr unterstützt werden. Oder plötzlich Fehlermeldungen angezeigt werden, die keine sind. Und da sich im selben Update auch Bugfixes befinden, die man möglicherweise dringend braucht, hat man keine Möglichkeit. Man muss immer wieder Codepassagen überarbeiten, die eigentlich als erledigt galten.

Kontrollverlust

Wer meint, am Computer alles unter Kontrolle zu haben, irrt sich – oder benutzt ihn nicht. Das Problem beginnt bei fehlerhafter Hardware, geht weiter über das Betriebssystem und endet bei der Software. Bei Game-Engines wird der Kontrollverlust besonders deutlich. So lange es sich nur um kleine Hobbyprojekte handelt, ist der Schaden womöglich überschaubar. Wer aber auf eine Engine setzt, um ein Unternehmen zu gründen, kann bei entsprechend schlechtem Timing komplett auf die Nase fallen.

Angenommen, ich hätte vor drei Jahren eine neue Firma gegründet. Kredite aufgenommen, Investoren gesucht, Mitarbeiter eingestellt und ein Spiel entwickelt. Release wäre 2024. Ein ambitioniertes Indie-Spiel, mit viel Herzblut in Unity entwickelt. Und dann kommt ein neues Lizenz-Modell. Nachdem der Taschenrechner bemüht wurde, stellt man fest, dass die Ausgaben nicht reingeholt werden können.

Was tun?

Drei Jahre Entwicklung in die Tonne treten? Alles mit einer anderen Engine umsetzen? Das ist Wahnsinn – und erklärt den Aufschrei der Community.

Spieleentwickler brauchen vor allem eine Engine, die zuverlässig funktioniert und mit der man langfristig planen kann. Unity hätte mehrere Möglichkeiten gehabt, ihr Problem zu lösen:

- ein besseres Lizenz-Modell,
- das vorgeschlagene Lizenz-Modell nur für Projekte ab dem 1. Januar 2024,
- eine neue Version, bei der das neue Lizenz-Modell gilt.

Man hätte also sagen können: Ab 2024 gibt es für die aktuelle Version nur noch Bugfixes. Wir bringen aber eine neue Version heraus, die hat ein paar richtig tolle Features, hat aber ein anderes Lizenz-Modell. Der Aufschrei hätte sich in Grenzen gehalten.

Interessanterweise setzt Unity einige dieser Punkte um und ergänzt sie durch weitere Zugeständnisse an die Community.

> For games that are subject to the runtime fee, we are giving you a choice of either a 2.5% revenue share or the calculated amount based on the number of new people engaging with your game each month. Both of these numbers are self-reported from data you already have available. You will always be billed the lesser amount.

Das ist der aktuelle Stand. Wie sehr man den Versprechungen glauben kann, steht in den Sternen. Wenn 2025 das Unternehmen den Besitzer wechselt, kann schon alles anders werden. Ähnlich wie beim GameMaker.

Alternativen

Mich persönlich ärgert die Unity-Geschichte, weil ich drauf und dran war, mich ernsthafter damit zu beschäftigen. GameMaker nervt mich immer mehr, Unity schien mir eine gute Alternative zu sein, zumal ich mehr mit 3D machen wollte. Dass GameMaker den Unity-Skandal ausnutzt, um Abwanderungswillige zu sich zu locken, ist angesichts der Unterschiede ein Witz. Zwar kann man auch im GameMaker Spiele in 3D machen, aber das will man niemandem empfehlen, der nicht ausdrücklich auf mentale Schmerzen steht. Und mir ist natürlich klar, dass es auch Entwickler gibt, die mit Unity 2D Spiele machen.

Für fortgeschrittene Programmierer gibt es natürlich die Möglichkeit, auf das ganze Zeug zu pfeifen. Mit *C*++ und SDL lässt sich eine eigene kleine Engine entwickeln, mit der man tolle Spiele machen kann. Und irgendjemand ist jetzt schon dabei in die Kommentare zu schreiben, dass *Java* immer die beste Alternative ist.

Würde ich nur für mich entwickeln wollen, wäre nun die Zeit für Unreal 3D. Aus Sicht des Bytegame-Redakteurs der gerne schreibt, sehe ich allerdings einige Probleme. Die Engine ist sehr komplex und mit *C*++ hat man auch nicht gerade eine Programmiersprache, die auf die Schnelle beigebracht werden kann. Ach ja, es gibt noch Blueprints, also die *Visual Scripts*. Das ist dann in Tutorials

bestimmt großartig, wenn ich seitenweise beschreibe, was man anklicken muss.

Letztlich bliebe da noch Godot, wobei ich hier insbesondere die Möglichkeiten im 3D-Bereich schlecht einschätzen kann. Aber möglicherweise ist das derzeit die – zumindest für mich – vernünftigste Alternative.

Was denkt ihr?

Externe Links

Unity Engine
Unreal Engine
GameMaker
Godot Engine
3D Gamestudio
SDL
Blender

Date Created
6. Oktober 2023
Author
sven