
Spieleentwicklung: Ein Hobby für alle!

### Description

Menschen brauchen ein Hobby. Als Ausgleich zum Beruf, Ventil der eigenen Kreativität oder als zentrale Identifikation. Warum also nicht Computerspiele entwickeln? Wir zeigen die aufregende Welt der Hobbyspieleentwicklung.

## Alle lieben Spiele

Es hat etwas gedauert, doch Videospiele sind mitten in der Gesellschaft angekommen. Ob stationär oder Mobil, es kann und wird immer und überall gezockt. Nie gab es, unabhängig von Alter und Geschlecht, so viele abwechslungsreiche Games.

Betagtere spielen Karten-, Brettspiele oder Retro-Klassiker, Jüngere stürzen sich in Online-Multiplayerschlachten. Manchmal ist es sogar umgekehrt. Trotz des Wettbewerbs und teils daraus folgenden Entgleisungen werden Milliarden Menschen durch ein wunderbares Hobby vereint. Doch man muss nicht nur konsumieren.

## Kreieren statt konsumieren

So genial viele Spiele auch sind, macht es mindestens ebenso Freude, etwas Eigenes zu erschaffen. Das Grandiose bei der Spieleentwicklung: Während man vielleicht im Kunst- oder Musikunterricht Probleme beziehungsweise weniger Spaß hatte, ist die Spieleentwicklung so umfangreich und abwechslungsreich, dass quasi für jeden etwas dabei ist.

In jeden Menschen steckt eine gewisse Kreativität. Ein Drang, Eigenes zu erschaffen. Wer an Spielen arbeitet, wird mit einer solchen Fülle mannigfaltiger Aufgaben konfrontiert wie bei keinem anderen Hobby.

Von vielen Menschen wird ein Computer entweder als Arbeitsgerät oder Spielekiste gesehen. Tatsächlich aber ist es der genialste Werkzeugkasten, den Menschen je entwickelten. Man kann – auf einer virtuellen Ebene – so ziemlich alles erschaffen, was sich jemand ausdenken kann. Egal oballeine oder im Team. Es gibt sogar Familien, in denen Kinder mit ihren Eltern zusammen Spieleerfinden. Unendliche, interaktive Welten.

## Warum soll ich Spiele entwickeln?

Diese Frage stellt sich womöglich einige Menschen. Ja, mit Games lässt sich Geld verdienen, teilweise sogar sehr viel. Doch das ist selten und reich wurden nur wenige Spieleentwickler damit. Wenn man talentiert ist und Glück hat, kann man, zumindest für eine gewisse Zeit, davon leben.

Dennoch ist die Hobbyspieleentwicklung eine grandiose Freizeitbeschäftigung. Das hat u. a. folgende Gründe:

1. Kreative Tätigkeit.
2. Abwechslungsreich.
3. Anspruchsvoll.
4. Schließt nahezu alle Arten der Kreativität ein.
5. Mit dem, was man macht, kann man anderen eine Freude bereiten.

Jeder Mensch hat Stärken und Schwächen. Bei der Spieleentwicklung kann man sich auf die Fähigkeiten fokussieren und etwas erschaffen, dass diese widerspiegelt. Angenommen, die Begabung besteht darin, Texte zu schreiben. So könnte man sich darauf beschränken und lediglich Kurzgeschichten und Bücher veröffentlichen. Oder man entwickelt ein Spiel und schafft mit seinen Geschichten ein interaktives Erlebnis.

Die Entwicklung von Computerspielen ist eine sehr gute Möglichkeit, die eigenen, innersten kreativen Potentiale zu verwirklichen.

Man kann sich dabei mit so vielen Dingen befassen, um das eigene Videospiel zu erschaffen oder es besser zu machen, wie bei kaum einem anderen Hobby. Mathematik, Psychologie, Grafikdesign, Animationstechnik, Sounddesign, Game-Design, Leveldesign, Typographie und vieles mehr. Die Möglichkeiten sind schier unendlich – und am Ende hat man ein kleines, interaktives Kunstwerk.

Gerade diese Interaktivität ist der Schlüssel. Bei nahezu allen anderen kreativen Tätigkeiten wird der „Konsument“ zum passiven Beobachter. Egal ob Buch, Film, Musik, Bühnenarbeit. Die Zielgruppe wird zur Passivität „gezwungen“. Spiele hingegen sind interaktiv. Dies schafft eine ganz andere, oft viel tiefere Verbindung zwischen Werk und Nutzer und somit auch zwischen Erschaffer und Nutzer.

Der Lohn der Mühe besteht meistens nicht darin, Geld zu verdienen, sondern die Spieler zu beäugeln, ihre Reaktionen zu studieren. Wer ein Buch schreibt, hat später wenig Freude daran, Leser zu beobachten. Bei Games ist das ganz anders. Man fiebert mit den Spielern mit und hofft, dass sie eine kritische Stelle schaffen. Es ist großartig zu erleben, wenn der Spieler den Gedanken des Leveldesigns folgen kann. Wenn er zunächst scheitert, seinen Fehler erkennt und es letztlich schafft. Das ist mit nichts zu vergleichen.

## Der Einstieg

Früher hatte man das Problem, dass man nicht an die Programmiersprachen kam, sofern man überhaupt wusste, was eine Programmiersprache ist. Klar, Basic hatten die meisten Computer, aber

da stieß man relativ schnell an die Grenzen, falls ein Handbuch zur Verfügung stand, mit dem sich Basic erlernen ließ.

Heute ist es umgekehrt. Die Fülle an Möglichkeiten ist so gewaltig, dass man vor einem riesigen Berg steht. Programmiersprachen, Entwicklungsumgebungen, verschiedene Plattformen. Soll es ein Mobile-Game werden? Für Desktops oder doch lieber HTML5? Welche Sprache sollte man lernen und muss man heute überhaupt noch programmieren lernen?

Es gibt keine perfekte Empfehlung, da die Ziele, eigenen Ambitionen und Vorstellungen sehr unterschiedlich ausfallen können. Wer meint, er könne auf Anhieb sein Traumspiel erschaffen, welches kommerziellen Ansprüchen genügt, wird wahrscheinlich scheitern und einsehen, dass es sinnvoll ist, zunächst kleinere Brötchen zu backen.

## Leveldesign

Vor allem für Jüngere ist es ziemlich ideal, zunächst nur eigene Levels zu erstellen. Einige Games haben Editoren, in denen man sich austoben kann. Das hat viele Vorteile. Das Spiel ist fertig und man kann sich sofort kreativ betätigen.

StarCraft II

Egal ob Shooter, Strategiespiele wie **StarCraft II**, Denk- oder Geschicklichkeitsspiele: Wer nur ein

wenig reinschnuppern und bauen möchte, hat hier sehr viele Möglichkeiten. Einige diese Editoren bieten sogar eine Skriptsprache an, wie etwa Lua, mit der man bspw. eine Geschichte erzählen oder schlicht Ereignisse auslösen kann.

## Skriptsprachen

Damit sind Programmiersprachen gemeint, die keinen Compiler nutzen, sondern „interpretiert“ ausgeführt werden. Zahlreiche dieser Sprachen sind, verglichen mit C, C++ etc. eher simpler gestrickt. Das heißt, dass viele Dinge abgenommen werden und man so einen relativ einfachen Einstieg in die Welt der Programmierung hat.

Wer bspw. über einen Leveleditor mit Lua in Berührung kommt, kann dieses Wissen nutzen, um mit Pico-8 zu programmieren.

Entwicklungsumgebungen, wie etwa der GameMaker, setzen auf solche Sprachen, auch wenn man GML nicht mehr wirklich als Skriptsprache bezeichnen kann. Sprachen, die in Entwicklungsumgebungen eingebettet wurden, nehmen viel Arbeit ab, indem zahlreiche Funktionen, die für die Spieleentwicklung benötigt werden, vorliegen. Etwa Zeichenfunktionen, Kollisionsabfragen u. v. m.

Früher hing die Wahl der Sprache vor allem vom System, für welches entwickelt werden soll, und der Performance ab. Heute eher von der Entwicklungsumgebung, die einen zu einer bestimmten Sprache zwingt. Die Unreal Engine setzt auf UnrealScript, Unity auf C# (C-Sharp gesprochen) und die Godot Engine bietet gleich drei Sprachen an: C++, C# und GDScript. Dafür kann man aus solchen Umgebungen meist für zahlreiche Plattformen kompilieren. Zumindest in der Theorie könnte man dasselbe Spiel gleichzeitig für Windows, Linux und Android entwickeln.

## Muss man noch programmieren?

Einige Engines werben damit, dass man nichts programmieren muss. Man kann sich die Befehle auch zusammenklicken. Der GameMaker bietet dafür ein komplexes Drag&Drop-System an. Die Unreal Engine löst das mit Blueprint.

Problem: Ab einer gewissen Komplexität verliert man den Überblick und es wird immer beschwerlicher, Fehlern auf die Schliche zu kommen. Je mehr Zeit man mit solchen Systemen verbringt, umso schwieriger ist der spätere Einstieg in die Programmierung.

Um es also kurz zu beantworten: Nein, man muss nicht unbedingt programmieren. Wer nur kleine Projekte realisieren will, kann dies i. d. R. auch ohne Programmierkenntnisse schaffen. Doch wenn man coden kann, ist so viel mehr möglich. Und ja, es gibt zahllose Sprachen, aber diese lassen sich in Gruppen unterteilen. Beherrscht man bspw. eine Programmiersprache, die den C-Syntax nutzt, versteht man Dutzende andere, etwa C++, C#, PHP, Java, JavaScript, GML u. s. f.

Man hat es dann mit anderen Sprachen, mitunter Python, Basic oder Ableitungen von Pascal etwas schwerer, aber die grundlegende Mechanik der Programmierung bleibt gleich. Deshalb empfehlen wir, sich mindestens mit einer Sprache intensiv zu befassen, egal welche. Dies ist der Grund, warum wir in Tutorials ausschließlich Code und keine Drag&Drop-Anleitungen bringen.

## Es muss nicht gleich ein ganzes Spiel sein!

Anfänger machen häufig einen Fehler: Sie beginnen mit einem großen Game. Von der ersten Minute an wird auf das Ziel hingearbeitet, ohne die Grundlagen zu verstehen. Das ist, als wolle man irgendwann einen Marathon laufen und würde gleich mit den 42,195 km starten.

Wie immer, wenn man vor einer großen Aufgabe steht, sollte man sie in kleine Bereiche aufteilen und sich damit vertraut machen. Die Entwicklungsumgebung und seine Möglichkeiten kennenlernen. Ganz allgemein Programmieren lernen. Wurden die Grundlagen verstanden, kann man mit kleinen Übungsaufgaben weitermachen. Sich vielleicht mit Shadern befassen, weil man hier bereits nach wenigen Zeilen Code schon einiges sehen kann.

Dies gilt dann auch für das eigentliche Spiel. Wie bewege ich eine Spielfigur? Wie scrollt die Kamera? Was muss ich bei einem Menü im Spiel alles beachten? Und ja, wie funktionieren grundlegend Savegames?

Als Spieler nimmt man alles als selbstverständlich hin, tatsächlich steckt aber auch in kleinen Games

sehr viel Arbeit und Wissen. Doch auch den großen Berg der Spieleentwicklung erklimmt man mit dem ersten Schritt. Also los!

**Date Created**
20. Mai 2022
**Author**
sven