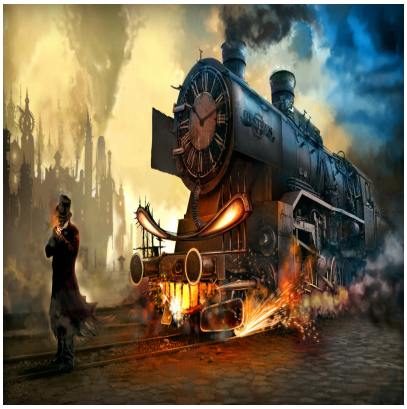


Die unbekannten Meister

Description

Für viele Programmierer bedeutet programmieren kaum etwas anderes, als Daten, durch die Anwendung gewisser Regeln, von A nach B zu schieben. Für wenige Programmierer ist es Kunst.

Handwerker

Spätestens seit den ersten Videospielen taucht immer wieder die Frage auf, wann Programmierung zur Kunst wird. Während man diese Frage in den klassischen Künsten schon lange beantwortet hat, oder es zumindest eine große Einigkeit darüber gibt, ist dies bei der Programmierung nicht so einfach. Programmierung ist in erster Linie Handwerk. Um den Vergleich zur klassischen Kunst weiter zu bemühen, könnte man diese Aussage ebenfalls auf die Malerei und auf die Musik und andere Bereiche anwenden. Im Kern ist es Handwerk.

Das Bestreichen von Wänden mit Farbe kann als solches nicht wirklich als Kunst angesehen werden. Ähnlich ist es bei einer Band oder einem Orchester, das bekannte Lieder nachspielt. Gefordert ist hierbei handwerkliches Können, die Kreativität kommt, wenn überhaupt, nur sehr Begrenzt zum Einsatz. Genau an dieser Stelle liegt der Hase im Pfeffer. Ab wann überschreitet die Kreativität eine Grenze, ab der das Handwerk zur Kunst wird?

Wie bei allem, dass eine gewisse Komplexität übersteigt, spricht man auch bei der Beherrschung der Programmierung gerne von Kunst. Tatsächlich ist damit aber nicht wirklich Kunst gemeint sondern soll zum Ausdruck bringen, dass die Ausübung der Tätigkeit schwierig und, wie gesagt, komplex ist. So kann man das Schachspiel ebenso als Kunst bezeichnen wie die Chemie, auch wenn in den

seltensten Fällen wirklich etwas künstlerisches empor kommt.

Die Schwierigkeit der Programmierung

Woher aber kommt diese Komplexität? Was macht das Programmieren so viel schwieriger als die Arbeit eines Bäckers oder eines Kochs? Im Kern wird es u. A. daran liegen, dass man es bei der Programmierung um sehr viele Disziplinen, aber vor allem mit mindestens zwei Sprachen zu tun hat. Da ist einerseits die Programmiersprache als solches und andererseits die Mathematik.

Je nach Programmiersprache wird das Verständnis immer schwieriger, die Hürde immer größer. Hochsprachen wie C++, Java oder Python sind relativ einfach zu lesen und zu schreiben, weil sie sich in der Regel an die englische Sprache anlehnen. Assemblersprachen hingegen sind sehr kryptisch und kaum noch mit dem normalen Vokabular von Menschen vergleichbar. Das fordert nicht nur ein Wissen über die Sprache, sondern auch über die Maschine, auf der ausgeführte Programme laufen müssen. Da wir uns hierbei in einer Welt befinden, die nur wenig mit unserer bekannten Realität zu tun hat, ist eine sehr hohe Abstraktion erforderlich, um sich darin zurecht zu finden.

Und dann war da noch die Mathematik. In vielen Fällen kommt man mit einfachen Berechnungen und Gleichungen klar, die kaum den Aufwand der 8. Klasse übersteigen. In vielen Entwicklungsumgebungen wird dem Programmierer bereits sehr viel Mathematik abgenommen, teilweise auch von der Hardware. Man sagt der Entwicklungsumgebung mit wenigen Befehlen, was man haben will, und bekommt ein entsprechendes Ergebnis. Doch dahinter verbergen sich oft komplexe mathematische Prozesse, wie Newtons Axiome bei einer Physik-Engine, Matrizen, Einheitsmatrix und vieles mehr im 3D-Bereich. Wer auch nur eine gute Wegfindung in einem Spiel selbst programmieren möchte, kommt ohne ein paar komplexerer Formeln nicht aus, wenn einem die Umgebung nicht bereits mit einem Befehlssatz versorgt.

Die Schwierigkeit liegt somit vor allem darin, mindestens die drei Punkte Programmiersprache, Mathematik und Maschinenkenntnisse logisch miteinander zu vereinen. Das ist, ohne abwertend klingen zu wollen, ein bisschen mehr als nur einen Pinsel zu halten oder Teig zu kneten.

Die Kunst gewordenen Zahlen

Es gibt viele Wege, als Programmierer künstlerisch tätig zu

werden. Seit Mitte der 1980er Jahre gibt es eine Subkultur, die dies auf ihre ganz spezielle Art und Weise tut. Die **Demoszene**. In ihr haben sich Grafiker, Musiker und vor allem Programmierer zusammengefunden, um digitale Kunst zu erschaffen. Teils in vorberechneten Animationen, bevorzugt aber als Echtzeitanimationen. Die Animationen laufen wie ein Spiel in dem Moment, in dem der Betrachter das Werk bestaunt. Wie ein Spiel wurden diese Animationen programmiert.

Dahinter stecken meist komplexe Algorithmen, die dem Betrachter eine virtuelle, oft sehr abstrakte Welt zeigen. Ein zentrales Thema sind dabei digitale Unendlichkeiten, also eine Welt, die keinen Anfang und kein Ende kennt.

Was in den 1980ern mit kleinen Cracktros begann, also Effekten, die man einem gecrackten Spiel vorgeschaltet hat, entwickelte sich zu einer eigenen Kunstform. Das Handwerk entwickelte sich zur Kunst, indem Programmierer Dinge entwickelten, die lediglich einem Selbstzweck dienten.

Bis Ende der 1990er Jahren waren die Techniken der Szene-Programmierer professionellen Spielen oft gleichwertig, teilweise sogar überlegen. Man konnte auf seinem C64, Amiga oder DOS-PC Dinge bestaunen, die kein Spiel und kein Musikvideo bis zu dieser Zeit bieten konnte. Technisch hoch anspruchsvoll und meist mit einem enormen künstlerischen Anspruch.

Doch oft ist nicht nur das, was man sehen kann, die Kunst daran. Künstliche Limitationen wie Dateibegrenzungen setzten den technischen Maßstab immer höher. Ab Ende der 1990er war klar, dass ein paar Jugendliche oder Studenten in ihrer Freizeit nicht mehr mit professionellen Spielen mithalten konnten, also mussten andere Wege her. Sicherlich kann eine Demo, rein technisch, nicht mehr mit einer modernen Unreal-Engine mithalten, aber dafür können die Dateien wesentlich kleiner werden. Eine mehrminütige Echtzeitanimation mit Musik in 64kb? Für die Demoszene ist das schon lange kein Problem mehr. Selbst ein leeres Projekt einer modernen Entwicklungsumgebung für Spiele hat mehrere Megabyte, in der Demoszene ist man sogar in der Lage, komplexe 3D-Welten in nur 4 Kilobyte anzuzeigen. Mit Musik.

Wer sich ein wenig mit technischen Gegebenheiten auskennt, dem wird klar, dass es sich dabei schon um Kunst im Quadrat handelt. Nicht nur das Gezeigte an sich ist Kunst, sondern auch die Art, wie es erschaffen wurde. Eine kleine Datei von wenigen Kilobyte wird im Speicher des Computers mehrere Gigabyte groß, zaubert eine atemberaubende Welt auf den Bildschirm und presst Klänge aus den Lautsprechern.

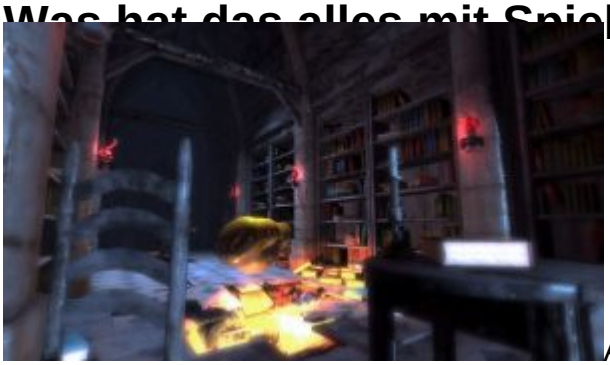
Der wissende Programmierer

Solche Echtzeitanimationen gehen weit über das Schaufeln von Daten hinaus. Mit einer normalen Entwicklungsumgebung für Spiele oder anderen normalen 3D

Engines kommt man hier nicht weit. Man muss so gut wie alles selbst erschaffen und dabei äußerst kreativ sein, um die technischen Probleme zu lösen. Wie etwa soll man ein 10 Minuten Lied in CD Qualität so klein bekommen, dass es nur wenige Kilobyte klein ist? Wie erzeugt man 3D Modelle und Texturen, damit das alles, mit der Musik, in eine winzige Datei passt?

Die Antworten darauf sind so komplex, dass selbst Laien klar wird, dass die Tätigkeit mit purem Handwerk nichts mehr zu tun hat. Die Programmierer schreiben nicht nur ihre eigenen Sound- und Grafikengines, sondern auch eigene Formate für Musik, Grafiken und 3D Modelle und Tools, in denen sie das alles erstellen können. Dies alles mit dem Ziel, am Ende eine winzige Datei zu haben, die eine Echtzeitanimation anzeigt, welche vorher niemand für möglich hielt. Sicher werden erfahrenere Programmierer einwerfen, dass man heute bereits mit wenigen Zeilen Shader-Code sehr viel auf den Bildschirm zaubern kann, aber die Demoszene ist viel mehr. Shader sind nur ein kleiner Bestandteil dieser hohen Kunst und die Programmierer zeigten bereits lange vor Shadern, was zur jeweiligen Zeit alles möglich war. Selbst heute werden noch Demos für den C64 und den Amiga programmiert und für Maschinen, auf denen man nie für möglich gehalten hätte, komplexe Effekte sehen zu können.

Was hat das alles mit Spielen zu tun?



Als Spieleentwickler fragt man sich, was man mit dem Wissen über diese Kunst anfangen soll? Schließlich schert sich heute fast niemand darum, ob ein Spiel 1 Megabyte oder 5 Gigabyte groß ist. Auch wenn ich das für einen Irrglauben halte, kann man sich grundsätzlich von der Kunst im allgemeinen und den Techniken im besonderen inspirieren lassen. In den 1990er Jahren ließen sich die Entwickler noch mehr inspirieren und integrierten Effekte und Techniken in ihren Spielen, die ihren Ursprung in der Demoszene hatten. Seien es Lieder, die mit Trackern gemacht wurden oder Effekte wie Plasmas oder Metaballs, immer wieder tauchten typische Szene-Effekte in spielen auf und Techniken, die ein Spiel effizienter machten.

Darüber hinaus sensibilisiert es Entwickler für Effizienz. Man kann mit sehr wenig Aufwand sehr viel Datenmüll erzeugen und somit auch sehr große Spiele erschaffen, welche die Festplatte voll stopfen. Man kann es aber auch anders machen und Platz sparen. Nicht jeder Shooter muss ein 96kb Spiel wie **.kkrieger** werden, aber mit einfachen Methoden wie Texturkompression oder dem intelligenten Einsatz von Musik und Samples lässt sich bereits viel Platz sparen.

Abgesehen von den rein technischen Dingen, stehen kreative, visuelle Effekte auch Spielen gut zur Gesicht. Egal ob in Menüs, Credits, Intro, Abspann, Hintergrundeffekt, Effekte im Spielgeschehen oder sogar als Spiel- und Levelbestandteil, der Kreativität sind keine Grenzen gesetzt und man kann sich von den tausenden Produktionen der Demoszene inspirieren zu lassen, um selbst kreativ zu programmieren. Das macht nicht nur viel Spaß, sondern wertet das Spiel auch in jeder Hinsicht auf.

Empfohlenes Video

Weiterführende Links

Die Demoszene (Wikipedia)
Pouet – Demoszene Archiv
ModPlug Tracker

Date Created
31. Oktober 2016
Author
sven