Projekt
Dokumentation

Steffen Mayer (Matr.Nr. 20946)
Medieninformatik (SS 2012)
Hochschule der Medien Stuttgart
Betreuer: Norman Pohl

Inhalt

Entstehung des Projekts

Die Idee zu diesem Projekt entstand schlagartig durch nahezu einen „Geistesblitz“ innerhalb der Vorlesung „Entwicklung von Computerspielen“ wobei dort die Idee in Form eines Game Design Scripts als Prüfungsleistung abgegeben wurde.

Da mich in dieser Vorlesung die Motivation zum Programmieren am meisten gepackt hat, entschied ich mich dazu, in meinem ersten ernsthaften/größeren eigenen Softwareprojekt mich durchzuarbeiten und die nötigen Fähigkeiten zu erlernen, um ein vollständiges Spiel zu entwickeln.

Erste Grundlagen in der Entwicklungsumgebung „Microsoft Visual Studio 2012“ bekam ich in dieser Vorlesung mit, um mich für dieses System zu entscheiden und somit wurde das Spiel mit der Technologie von XNA Game Studio 4.0 (Programmiersprache C#) geschrieben.

Projektbeschreibung

Es handelt sich um ein Computerspiel das sich in die Kategorie „Casual Game“ einordnen lässt, da man es ohne Vorkenntnis oder Übung oder besondere Fähigkeiten und ohne Einlernphase sofort intuitiv los spielen kann. Es soll einfach für spontanen, kurzweiligen Spielspaß sorgen.

Das Grundprinzip wie das Spiel abläuft ist ziemlich schnell beschrieben: Man steuert ein Fahrzeug aus der 2D Ansicht von oben betrachtet (bekannte Vergleichsmöglichkeit der Perspektive wären die ersten GTA spiele) auf einer unendlich großen Spielwelt. Dabei sammelt man auf der Karte liegende Items ein indem man sie berührt.

Es gibt Energie-Items (Fuel) und Geld-Items (Coins). Mit den Fuel-Items (die als Treibstoff-Tröpfchen dargestellt sind) füllt man den Treibstofftank des Fahrzeuges auf, damit man weiterfahren kann. Mit dem eingesammelten Geld, kann man sein Fahrzeug aufrüsten in den Parametern Maximalgeschwindigkeit, Beschleunigung, maximaler Tankinhalt und Reichweite zum Einsammeln der Items.

[Info Screen]

Der Spielspaß soll sich aus folgenden Faktoren zusammen setzen:

- Das Fahrzeug soll sich angenehm steuern lassen und mit der Zeit durch das Aufrüsten ein immer „rasanteres" Fahrgefühl vermitteln.
- Qualitativ hochwertige und vielschichtige Motorengeräusche sollen das Geschwindigkeitsgefühl unterstützen.
- Die Gefahr, dass der Treibstofftank leer geht soll dem Spieler bewusst sein.
- Es soll ein gewisser „natürlicher Sammlertrieb" des Spielers auf seine Kosten kommen.
- Das Aufrüstsystem vermittelt das Gefühl, dass es „immer weiter voran geht"
- abwechslungsreiche Soundeffekte sollen das Spielerlebnis verstärken.
- Optisch qualitativ hochwertige Item-Grafiken und Untergrundtexturen sollen das Spiel aufwerten.
- An einer High Score kann mein seine Erfolge messen.

Ursprünglich war das Konzept als Endlosspiel gedacht, aber das Resultat des Projektes geht zwar relativ lang, stößt aber im jetzigen Zustand noch recht schnell an seine Grenzen.

Ohne weiteres gerät das Spiel jedenfalls noch früher oder Später in einen Zustand bei dem es zu einfach ist oder unmöglich wird und somit nicht mehr Spaß machen kann – je nach dem wie man gewisse Parameter einstellt die große Auswirkung auf den Spiel verlauf haben.

Z.B. gibt es in der aktuellen Version neben den normalen Coins immer einen Goldschatz zufällig plaziert auf der Karte, der eine größere Geldmenge liefert, der nach de Einsammeln seinen Wert um einen konstanten Faktor erhöht und gleichzeitig seine Entfernung um einen konstanten Faktor erhöht. Diese beiden Faktoren sind Momentan z.B. maßgeblich entscheidend, ob es viel zu einfach oder zu schwierig wird.

Einarbeitung

Dieses Projekt ist meine erste ernsthafte umfangreichere Programmier-Arbeit so dass ich noch kaum Erfahrung und Übung im Programmieren zu Beginn des Projektes hatte.

Für mein verwendetes XNA Game Studio gibt es viele hilfreiche Internetseiten und auch auf Youtube einige hilfreiche Tutorials auf deutsch und auf englisch, die mir zu Beginn sehr geholfen haben zu verfolgen – auch z.B. um zu sehen wie leicht man Fehler machen kann und wie man diese wieder beseitigt und allgemein wie aufwändig man zum Teil mit Fehlern umgehen muss.

Viele hilfreiche Tipps und Erkenntnisse konnte ich durch diese Tutorials und spontanen nebensächliche Erklärungen daraus entnehmen, was mir sehr geholfen hat die richtige Denkweise zum Programmieren zu verwenden.

Mit voller Konzentration hab ich versucht wirklich alles zu verstehen was schritt für schritt im Tutorial gemacht wurde und viele Stellen dazu auch lieber mindestens zweimal hintereinander angeschaut.

Hilfsquellen

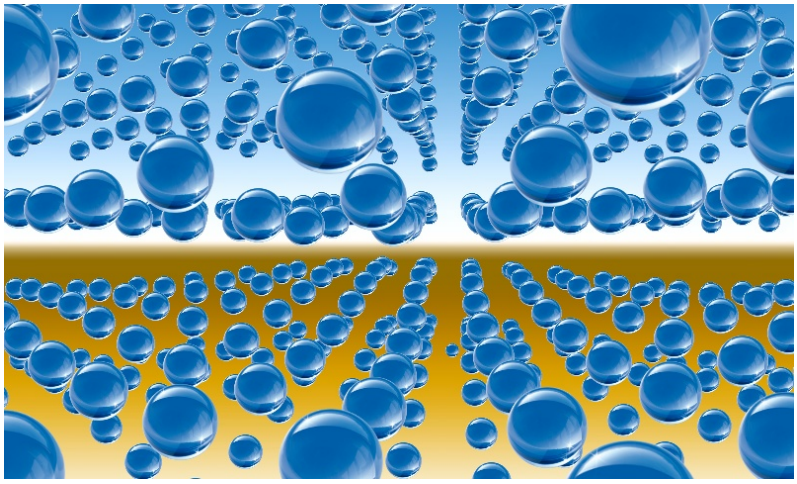
Folgende Hilfsquellen hab ich zur Einarbeitung und zum Nachschlagen während der Entwicklung verwendet:
http://msdn.microsoft.com/de-de/
http://www.xnadevelopment.com/tutorials.shtml
http://xnafaq.niklas-rother.de/content/6/15/de/verwinkelte-winkel.html
http://www.xnamag.de/
http://www.theorangeday.com/xna/understanding-c-2d-xna-hlsl/
http://www.facewound.com/tutorials/shader1/
sowie Scripte der Vorlesung „Entwicklung von Computerspielen“

Nebenprojekte

Bevor ich mit dem eigentlichen Projekt begonnen habe zu Programmieren, wollte ich erst ein paar Erfahrungen mit anderen Ideen sammeln – vor allem um etwas Routine im Strukturieren des Programmcodes bekomme.

Daraus entstanden Ansätze einer kleinen 3D Engine und minimal ansatzweise ein Tower Defence Spiel und ein Programm zum generieren von Julia Sets und der berühmten Mandelbrot Menge bzw. beliebige Ausschnitte davon.

3D Engine



Nachdem ich viel gelernt hab und noch nichts eigenes programmiert, hatte ich das Gefühl ich könnte alles mögliche erdenkliche nun realisieren und ich bekam den extrem hoch angesetzten (utopischen) Wunsch, eine 3D Engine zu entwickeln die sich von gewohnter Grafik einerseits komplett unterscheidet, andererseits auch teilweise Methoden früherer Pseudo-3D-darstellungen anlehnt, allerdings mit heutiger Rechnerleistung.

Und zwar sollte eine 3 dimensionale Welt generiert werden aus punkten unterschiedlicher Typen, die mit passenden Texturen versehen werden so dass das Bild quasi aus lauter einzelnen kleinen „granularen" Bitmaps

besteht, die nur gezeichnet werden, wenn sie im Vordergrund sind auf dem entsprechenden Bereich der Bildfläche.

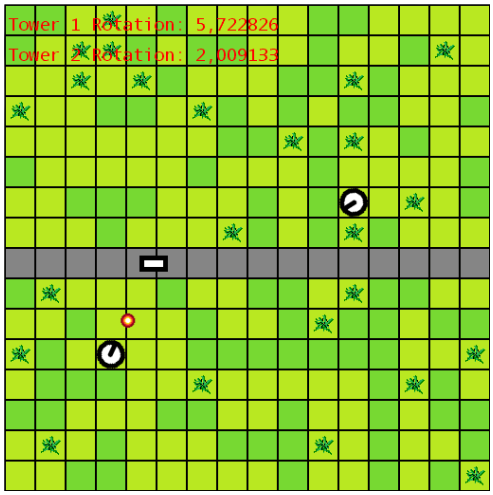
Bis zur Generierung von Welten und deren Visualisierung durch unterschiedliche Textur-Elemente bin ich zwar nicht gekommen, aber immerhin konnte ich eine Textur im 3 Dimensionalen Raum platzieren und mit Tastatureingaben „um die Textur herumgehen"

Dabei hab ich mathematisches Wissen über Rotationsmatrix und Vektor Addition usw. verwendet, ohne Verwendung von vorhandenen Klassen, die Projektion der Textur auf dem Bildschirm in Abhängigkeit der Positionen von Betrachter und Objekt und Rotation des Betrachters zu realisieren.

Erstaunlicherweise gelang dies mit wenigen Code - Zeilen und Vereinfachungen, allerdings zunächst mit etwas „Kopfzerbrechen", so dass ich hinterher aus dem einen Objekt viele gemacht habe, die sich im 3D-Raum befinden und bewegen/schwingen und man sich durch sie hindurch bewegen und drehen kann.

Die Projektion der Objekte wurde lediglich durch Normalisierung des Vektors aus der Sicht des Betrachters auf eine Einheitskugel um ihn herum berechnet, was zwar leichte Verzerrungen verursacht, die aber bei einem realen Sichtfeldausschnitt nicht wirklich auffallen.

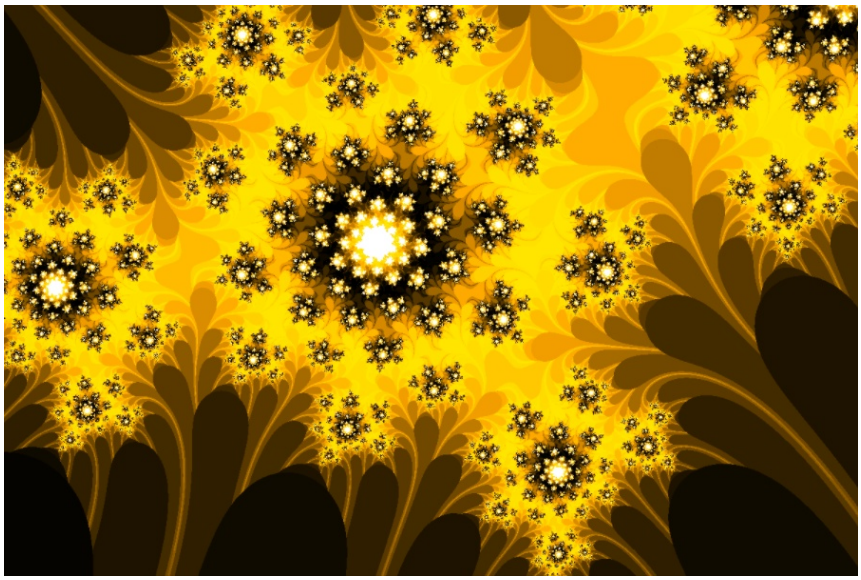
Tower Defence



Mein zweiter Programmierversuch war der Beginn eines kleinen Tower Defence Spiels bei dem ich erstmals eine zufällig generierte Karte erstellen konnte.

Sehr weit bin ich damit nicht gegangen aber immerhin steht die Karte und ein „Gegner" wird von ausrichtenden Geschütztürmen beschossen die eine begrenzte Reichweite haben.

Mandelbrot und Julia Set



Mitten in der Hauptprojektphase hab ich mich begonnen für die Entstehung von Mandelbrot Grafiken zu interessieren und glücklicherweise fand ich passende Videos auf YouTube die das Prinzip mir verständlich machen konnten so dass ich die erstaunlich primitiven Anweisungen zur Bildung solcher unendlich komplexen Mustern relativ schnell im XNA Game Studio umsetzen konnte.

Üblicherweise benötigt man zum Programmieren länger als vermutet aber in diesem Fall hatte ich eher ein „was schon fertig?" Erlebnis, nachdem sich plötzlich von selbst ein perfekter Kreis aus der Liste der quadratisch angeordneten Pixel gebildet hat und ich nur noch einen Parameter hinzufügen musste, damit sich daraus ein sogenanntes „Julia Set" bildet.

Programmierung

Bei der Programmierung des eigentlichen Spieles, hab ich mich weitgehend an mein Konzept aus dem Game Design Script der Vorlesung „Entwicklung von Computerspielen" gehalten, welches unterteilt war in verschiedene Entwicklungsstufen, von ganz simpel ohne Spielinhalt bis immer komplexer und Umfangreicher. Aus den 6 Stufen hab ich im Endeffekt bis in die 5. Stufe hinein entwickelt.

Zunächst wurde nur die unendlich große Welt angelegt mit abwechselnden Untergründen die sich logarithmisch im Verhältnis zum Abstand zum Mittelpunkt abwechseln.

Da mir klar war, dass die Steuerung des Autos insgesamt ziemlich komplex zu programmieren ist, wollte ich alles was mit dem Auto zu tun hat in eine extra Klasse schreiben und dabei bin ich an meine ersten Schwierigkeiten aus Erfahrungsmangel gestoßen, weil ich z.B. nicht wusste wie man eine einzelnes Objekt erstellt ohne dafür zunächst eine Klasse zu schreiben und davon eine Instanz zu erstellen. Die Lösung war dann das Stichwort „static". Aber mir war vorher nicht mal richtig bewusst, dass Klassen manchmal nur einzeln Vorkommen und manchmal mehrfach und es dafür jeweils eine saubere Lösung gibt.

Da ich viel Wert legte auf ein angenehmes Fahrverhalten mit der Möglichkeit per Handbremse zu driften, gab ich mir ziemlich viel Mühe und setzte meine ganze Gehirnkapazität mit höchster Konzentration ein um mit raffinierten Tricks das Bewegungsverhalten des Autos immer weiter zu optimieren, was immer weitere Probleme auslöste, die noch trickreicher zu beseitigen waren. Z.B. als Extramfall sollte der Rotationspunkt der Auto-Textur sich vom Standardpunkt im Bereich der Hinterachsen möglichst unbemerkt (!) nach vorne verlagern, wenn man seitlich driftet, damit die Drehbewegung ein „Ausbrechen" der Hinterreifen (wie zu erwarten) darstellt während bei Lenkungen mit langsamen Geschwindigkeiten das Auto eher um einen Punkt weiter hinten rotiert.

Um diese Verschiebung wieder unbemerkt zurück zu stellen, muss die

Geschwindigkeit der Zurückstellung von der Fahrtgeschwindigkeit abhängen, denn je schneller man fährt, um so weniger fällt es auf allerdings sollte es möglichst schnell zurückgestellt werden, da bei scharfen Kurven mit niedriger Geschwindigkeit z.B. direkt nach dem Ausdriften und loslassen der Handbremse, sofort der Rotationsmittelpunkt sichtbar wird.

Viele Versuche waren nötig um diesen Trick möglichst optimal zu vertuschen.

Eine große Herausforderung war es auch das Auto dem Spielprinzip nach von winzigen Anfangsmaximalgeschwindigkeiten durchgängig stufenlos bis zu höchsten unrealistischen Maximalgeschwindigkeiten im späteren Spielverlauf angenehm und realistisch fahrbar zu programmieren.

Bedingungen die ich mir dabei (zu Gunsten des Realismus) gestellt habe waren unter anderem: Das Auto soll nicht bis zur Maximalgeschwindigkeit konstant beschleunigen sondern ab einem Gewissen Anteil davon sich langsam der Maximalgeschwindigkeit annähern.

Im großen und ganzen kann man sagen sämtliche Tricks und Kompromisse waren einerseits nötig um keine komplett realistische Physik realisieren zu müssen und andererseits um den Spielspaß im Vordergrund zu bevorzugen zu Lasten des Realismus. Da allein die hohen Geschwindigkeiten bei fortschrittlicherem Spielverlauf unrealistisch sind, macht es auch keinen Sinn, dafür physikalisch korrekte Reibungswerte z.B. zu verwenden, damit das Auto nicht ständig völlig außer Kontrolle gerät.

Ein Trick war es auch, den Vektor der Fahrtgeschwindigkeit beim driften aufzuteilen in Abhängigkeit der Rotation des Autos um hinterher beim Loslassen der Handbremse den neuen umgelenkten Geschwindigkeitsvektor um den Anteil des „Driftvektors“ zu verringern und dann langsam der ursprünglichen Geschwindigkeit wieder anzunähern, so dass das Auto keine eckigen Bewegungen macht und sich zwar nicht realistisch aber irgendwie logisch und angenehm steuern lässt.

Viele bedingte Anweisungen in der Auto Klasse sind so komplex unter höchster Konzentration entstanden, dass man es kaum jemand zumuten kann das Geschehen ganzheitlich nach zu vollziehen und es mir leider selber zum Teil kaum noch gelingt. Aber das macht das Programm am Ende um so wertvoller.

Um die Performance gering zu halten, wurden sowohl sämtliche Kollisionsbedingungen einigermaßen optimiert als auch sämtliche „Draw" Befehle lediglich im Bereich der sichtbaren Bildschirmausgabe ausgeführt.

Dabei entdeckte ich z.B. die Möglichkeit den Wert der Einsammelreichweite im Quadrat abzuspeichern, damit beim Vergleich ob sich ein Item innerhalb des Kreises mit dem Radius der Wurzel aus diesem Wert die aufaddierten Quadratflächen nicht erst mit der aufwändigen Wurzelfunktion berechnet werden müssen sondern man den Satz des Pythagoras einfach mit „$a^2 + b^2 < c^2$" verwenden kann und nicht erst die Wurzel aus „$a^2 + b^2$" ziehen muss.

Im jetzigen Zustand des Spieles sind die Coin Items und Fuel Items kreisförmig (mit zufälliger Streuung) um den Mittelpunkt der Spielwelt angeordnet. Der Abstand der Ringe vergrößert sich logarithmisch.

Die Karte mit ihren Items und Dekorationen wird in Fahrtrichtung des Autos mit einem „Motion Blur" Effekt versehen, bei dem das vorhandene Bild drei mal hintereinander jeweils 9 mal in unterschiedlicher Stärke in Fahrtrichtung „aufgesplittet" wird, damit insgesammt über 700 Bilder der Spielwelt in Fahrtrichtung „verschmiert" werden, damit wird bei höchster Geschwindigkeit ein stufenloses weiches Bild gewährleistet.

Dazu hab ich die Möglichkeit kennen gelernt, die Bildschirmausgabe virtuell zwischen zu speichern – auch mit Transparentem Hintergrund – und nachher diese zwischengespeicherten Schichten als letztendliche Bildausgabe am Bildschirm zu kombinieren als ob es normale Texturen wären.

Content

Motorengeräusche

Für aufwändige Motorengeräusche wurde das Audio Creation Tool „XACT" verwendet, welches vom XNA Game Studio mitgeliefert wird.

Darin kann man Audio-Dateien kombinieren als Soundbank und mit Effekten versehen, die sich nachher im Programmcode durch Übergabe von Parametern beeinflussen lassen.

In diesem Fall hab ich mehrere verschiedene selbst ausgeschnittene Loops von echten Motorgeräuschen aus einem Sample Library unterschiedlicher Geschwindigkeiten verwendet und zusätzlich ein Windgeräusch mit einbezogen. Außerdem hatte die Idee, dem Fahrzeug bei unrealistisch hohen Geschwindigkeit auch einen unrealistischen bzw. synthetischen Motorensound zu verpassen.

Diese sämtlichen Audio-Spuren werden abhängig von dem übergebenen Geschwindigkeitsparameter der Reihe nach übereinander geblendet und in der Tonhöhe variiert, so dass sich eine extrem lange Geschwindigkeitserhöhung akustisch abwechslungsreich und glaubwürdig unterstützen lässt.

Schwierigkeiten gab es ein Bisschen wegen den zusätzlichen Motorgeräuschen beim Motorstarten und wenn der Motor aus geht, wenn der Tank leer ist, da ich hierbei etwas durcheinander gekommen bin, wann ich welche Soundquelle wie am besten aus und ein schalte und überblende um meinen Qualitätserwartungen gerecht zu werden.

Texturen

Als Untergrund wurden verschiedene übergangslos 'kachelbare' Texturen

aus einem kostenlosen Internetangebot in Auflösungen von ca. 700 Pixel Kantenlänge eingesetzt.

Die Items wurden mit Photoshop selbst erstellt und in unterschiedlichen Größen abgespeichert, mit der Vorstellung, dass 1:1 dargestellte Objekte am wenigsten Ressourcen verbrauchen.

Die grafische Benutzeroberfläche wurde vom Design her an Windows 7 angelehnt mit glasigen, durchsichtigen Flächen damit das Spiel etwas plastischer bzw. aufgewerteter oder auch „verspielter" aussieht.

Gegen Ende der bisherigen Entwicklung kam die Idee eines größeren Goldschatzes zum Einsammeln, den ich mehrschichtigen Grafiken animiert hab. Unter anderem „glitzern" kleine zufällig zerstreute weise Partikel um einen Goldring die ihn in einer angedeuteten Kugel halb umhüllen sollen. Auch dafür wurde alles mit Photoshop generiert.

Soundeffekte

Sämtliche Soundeffekte im Menü und im Spiel beim Einsammeln von Items, sind aus alten großen Sample-Sammlungen die sich keiner wirklichen Quelle mehr zuordnen lassen können. Sie wurden intuitiv ausgewählt mit dem Ziel dass der verwendete Sound „in einem Spiel so klingen könnte" in der entsprechenden Situation in der er auftritt.

Die Sounds dienen als kleine Symbole für jede kleine Aktion, um das Spiel geschehen mehr wahr zu nehmen.

Quellen

http://www.cgtextures.com/ (Untergrund Texturen)
http://web.vw.com/why-vw/safety/media/images/slides/car-top-view4.png (Auto Textur)
http://pixlr.com/o-matic/ (Effekt für Startbild)