Save the Sweets

Softwareprojekt Dokumentation

Inhalt

Einleitung

Das hier dokumentierte Spiel ist im Rahmen der Veranstaltung “3D-Gameproject” entstanden. Es wurde in C# geschrieben und baut auf dem XNA-Framework auf.

Entwickelt wurde es von:

- Anne-Lena Simon: Projektleitung, 2D-Grafik, Entwicklung
- Björn Golla: Entwicklung
- Sebastian Rohde: 3D-Grafik, Entwicklung
- Svenja Handreck: GUI, 2D-Grafik, Entwicklung

Die genaue Verteilung der einzelnen Aufgaben auf die Teammitglieder kann an der farbigen Markierung am rechten Rand abgelesen werden.

Planung

In diesem Abschnitt wird darauf eingegangen, wie das Konzept für das Spiel entstand und welche Entscheidungen für das Game Design getroffen wurden. Dabei wird besonders auf die einzelnen Spielelemente eingegangen.

Initiale Idee

Das erste Spielprinzip, das in Erwägung gezogen wurde, war ein Simulationsspiel, in dem der Spieler eine Population steuern, aufbauen und entwickeln kann. Da es bei dieser Art von Spiel aber besonders schwer ist, den Spieler motiviert zu halten, und da bekannt wurde, dass das *MediBalance Pro* zur Verfügung steht, wurde die Idee zugunsten eines Jump&Run-Spiels mit Rätselelementen verworfen.

Genrewahl

Das *MediBalance Pro*, das als Eingabegerät angeboten werden soll, schränkt die Möglichkeiten bei der Genrewahl ein.

Ein Spiel, das zur Therapie mit diesem Gerät entwickelt wird, soll den Spieler davon ablenken, dass er seinen Gleichgewichtssinn trainiert. Es muss möglich sein, das Spiel nach zehn bis 20 Minuten zu beenden. In diesem Zeitraum müssen dem Spieler allerdings Erfolgserlebnisse geboten werden und er sollte sich auf etwas anderes als die Steuerung konzentrieren.

Um einen gewissen Grad an Immersion zu erreichen, bietet sich eine First- oder Third-Person-Perspective an. Gewählt wurde letztere, da den Spielern die Orientierung in ihrer Umgebung oft leichter fällt, wenn sie mehr von ihrer Umgebung sehen können.

Da bei einem Jump&Run das Bewegen durch die Spielwelt ein grundlegender Bestandteil der Spielmechanik ist, der Spieler sich aber voll auf das Absolvieren des aktuellen Spielabschnitts konzentrieren muss und dabei immer wieder kleine Erfolge erfährt, scheint das Genre gut geeignet.

Prototypen

Der Entwicklung dieses Spiels ging die Entwicklung von Prototypen von 3D-Spielen durch die einzelnen Teammitglieder voraus.

Prototyp 1 - Björn Golla, Sebastian Rohde & Bianca Hinz

Bei dem ersten Prototypen handelt es sich um ein Spiel, bei dem der Spieler eine Figur steuert, die zwischen drei Positionen auf gerader Strecke wechseln und zusätzlich springen kann.

Auf die Figur kommen Hindernisse zu, denen sie ausweichen muss, und Münzen, die der Spieler für zusätzliche Bonuspunkte einsammeln kann.

Ziel des Spiels ist es, so lange wie möglich zu überleben und so einen hohen Punktestand zu erzielen. Je länger der Spieler den Hindernissen ausweichen kann, desto höher steigt sein Punktestand. Läuft oder springt der Spieler gegen eines der Hindernisse, verliert er. Es wird ein *Game Over*-Bildschirm mit der erreichten Punktzahl und der bisherigen Bestleistung angezeigt und ein neues Spiel kann gestartet werden.

Für den Prototypen war es notwendig, Modelle von Spieler, Hindernissen und Münzen zu erstellen und in das Spiel einzubinden. Auch Hintergrundmusik und Soundeffekte beim Einsammeln von Münzen oder bei Kollision mit einem Hindernis mussten integriert werden. (Sebastian)

Das Verhalten der einzelnen Spielobjekte und ihre stetige Fortbewegung wurden implementiert. Auch eine zufällige Platzierung der Elemente auf den drei verfügbaren Spuren, die mit zunehmender Spielzeit immer anspruchsvoller wird, und die Punkteanzeige sowie ein Start- und ein Resetbildschirm wurden umgesetzt. (Björn)

Die Funktionalitäten des spielbaren Charakters sowie die Kameraeinstellungen wurden ebenfalls eingebaut. (Bianca)

Prototyp 2 - Anne-Lena Simon & Philipp Schladitz

In dem zweiten Prototypen spielt der Spieler ein fliegendes Objekt, das sich immer nur um 90° drehen kann und nur dann stoppt, wenn es von einem Hindernis aufgehalten wird. Die

Flugzeit ist dabei begrenzt. Wird die maximale Flugzeit überschritten, stürzt das Objekt ab und der Spieler muss neu beginnen.

Ziel ist es, in kürzester Zeit vom Start zum Ziel zu kommen, indem das Objekt geschickt durch ein Labyrinth navigiert wird. Dabei gibt es sowohl statische Hindernisse als auch solche, die vom Spieler von ihrer Plattform geschoben werden, sobald er sie berührt. Verschiebbare Hindernisse sind an ihrer grünen Farbe erkennbar.

Eine einfache Übersichtskarte ermöglicht es dem Spieler, seinen Weg im Voraus zu planen.

Das Spiel erforderte die Implementierung einer Kamera in drei verschiedenen Perspektiven (als Fußnote: Third Person View, First Person View, Aufsicht für die Karte). Auch die Kollision des Spielers mit den verschiedenen Hindernissen und das Herabfallen von grünen Hindernissen, nachdem der Spieler an sie gestoßen ist, mussten umgesetzt werden. Zudem musste das Level sorgfältig so entworfen werden, dass es lösbar, aber nicht zu einfach ist. (Anne)

Weiterhin mussten Modelle für den Spieler, die Hindernisse und die Plattformen, auf denen die Hindernisse liegen, erstellt und ins Spiel integriert werden. Bei der Umsetzung der Fortbewegung musste darauf geachtet werden, dass Drehungen immer nur um 90° möglich sind und der Spieler nicht auf Tastendruck anhalten kann. Außerdem musste das Level dem Design folgend in das Spiel eingebaut werden und die Überprüfung der Flugzeit zusammen mit entsprechenden Anzeigen auf dem Bildschirm während des Spiels, bei Sieg und bei Niederlage umgesetzt werden. (Philipp)

Prototyp 3 - Svenja Handreck & Jevgenij Huebert

Bei dem dritten Prototyp steuert der Spieler ein Raumschiff, das sich permanent nach vorn bewegt. Andere Raumschiffe kommen dabei auf den Spieler zugeflogen. Ziel ist es möglichst viele Punkte zu sammeln, indem möglichst viele Raumschiffe abschossen werden. Es gibt positive und negative Items, die von abgeschossenen Schiffen verloren werden und die der Spieler entweder abschießen oder einsammeln kann. Sie erhöhen oder verringern die Munition und die Geschwindigkeit und verändern weitere Parameter. Die Geschwindigkeit wird kontinuierlich erhöht. Wenn das Schiff mit anderen Schiffen kollidiert, wird es abgebremst. Das Spiel endet, wenn die Zeit abgelaufen ist. Dann erhält der Spieler eine Auskunft über die bisherigen Highscores.

Für das Spiel wurden eine Kamera, die Steuerung des eigenen Schiffes, sowie das Abfeuern von Lasergeschossen implementiert. Für die gegnerischen Schiffe werden zufällig Startpositionen erzeugt und es wird auf Kollisionen zwischen dem eigenen Schiff oder seinen Geschossen und anderen Schiffen getestet. (Svenja)

Für die Schiffe wurden Modelle erstellt und Effekte für die Kollisionen eingebaut. Das Spiel bekam ein HUD und die verschiedenen Items mit ihren positiven und negativen Effekten

wurden implementiert. Die Punkte mussten gezählt und das Highscoresystem implementiert werden. Die Kamerasteuerung und Kollisionen wurden nochmal überarbeitet und verbessert. Des Weiteren wurden Musik und Soundeffekte eingebunden. (Jevgenij)

Grundlegende Spielmechanik

Im Anschluss an die Entwicklung der Prototypen wurde genauer ausgearbeitet, wie das Spiel aussehen und funktionieren soll. Ausgegangen wurde dabei von der Entscheidung bezüglich des Genres und von zwei der Prototypen.

Grundsätzlich soll der Spieler sich in einer offenen Welt ohne Einschränkung, die über die Schwerkraft hinausgeht, bewegen können. Sollte er einen alternativen Weg finden, so soll das den weiteren Spielverlauf nicht stören und dem Spieler nicht an anderer Stelle Probleme bereiten.

Schieberätsel

Schieberätsel sind eines der Kernelemente des Spiels. Dabei geht es darum, Blöcke zu nutzen, um sich durch die Spielwelt zu bewegen. Das geschieht einerseits, indem Wege freigeräumt oder Brücken gebaut werden und andererseits, indem sie als Stopper für den Spieler oder andere Objekte genutzt werden. Diese Art von Rätsel wurde gewählt, da der Schwierigkeitsgrad gut variiert werden kann und es den Spieler dazu bewegt, sich auf die Lösung des Rätsels und nicht auf die Steuerung zu konzentrieren.

Es wurde ein Abschnitt geplant, in dem der Spieler sich frei bewegen kann. In diesen Rätseln muss er verschiebbare Blöcke und Schalter nutzen, um sich einen Weg durch die blockierte Spielwelt zu bahnen. Es gibt Schalter, die nach einmaligen Betreten immer aktiv bleiben, andere, die nur so lange aktiv sind, wie sie durch den Spieler oder ein Hindernis belastet werden und solche, die nur eine bestimmte Zeitspanne lang aktiv sind.

Die Fähigkeiten der Spielfigur beschränken sich auf Bewegung, das Verschieben von Blöcken und auf Sprünge.

Zudem werden spezielle Plattformen eingeführt, auf denen verschiebbare Blöcke selbstständig rutschen, sobald sie angeschoben wurden. Stößt ein beweglicher Block auf der rutschigen Fläche gegen einen weiteren, bleibt der erste stehen und der zweite setzt sich dafür in Bewegung.

Ebenenpuzzle

Um die Komplexität der Rätsel zu erhöhen, wird ein zweiter spielbarer Charakter eingeführt. Dieser kann weder springen, noch Blöcke verschieben, ist dafür aber in der Lage, brüchige Blöcke zu zerstören. In dem Spielabschnitt, der sich an die Einführung dieser neuen Spielfigur

anschließt, muss der Spieler den Hauptcharakter des Spiels auf eine hoch liegende Ebene bringen und bleibt mit dem neuen Charakter auf der unteren Ebene. Er muss nun immer wieder zwischen den Spielfiguren wechseln, um beiden zu ermöglichen, den Spielabschnitt zu durchqueren. Hauptsächlich funktioniert das über drei Mechanismen. Der Spieler schiebt Blöcke von der oberen Ebene herab, so dass sie in Löcher fallen und diese füllen, damit die Spielfigur, die nicht springen kann, weiterlaufen kann. Weiterhin können Blöcke, die den Weg versperren oder fehlende Plattformen durch Schalter auf der jeweils anderen Ebene entfernt oder hinzugefügt werden. Außerdem können auf der unteren Ebene Säulen aus Blöcken zerstört und damit auf der oberen Ebene der Weg freigeräumt werden.

In diesem Abschnitt werden zudem Plattformen, die nach Betreten in wenigen Sekunden zerbrechen, und Schaltergruppen eingeführt. Sind Schalter gruppiert, so müssen sie alle betätigt werden, bevor sie eine Änderung bewirken.

Rutschrätsel

Die Kernmechanik des nächsten Abschnitts ist dem zweiten Prototypen sehr ähnlich. Es wird auf den rutschigen Plattformen aus den Schieberätseln aufgebaut. Zusätzlich zu diesen gibt es dort Plattformen, auf denen auch der Spieler rutscht, wenn er sie betritt. Er ist nicht dazu in der Lage, seine Richtung zu ändern, während er rutscht. Springen ist jedoch möglich. Die Spielfigur kann nur anhalten, wenn sie die rutschige Fläche verlässt oder von einem Hindernis angehalten wird. Handelt es sich dabei um ein bewegliches Hindernis, wird es durch die Kollision in Bewegung versetzt.

Verfolgungsjagd

Eine andere Sequenz wurde dem Prototypen 1 entnommen. Dabei wird die Steuerung eingeschränkt, sodass die Spielfigur sich automatisch vorwärts bewegt und der Spieler nur nach links oder rechts steuern kann, um dem Weg zu folgen und Hindernissen auszuweichen. Er kann die Richtung dabei nicht wechseln, kann aber anhalten, um sich zu orientieren. Er steht zeitlich unter Druck, da zu langsames Vorankommen dazu führt, dass der Spielabschnitt wiederholt werden muss. Dieser Abschnitt wurde gewählt, um nahe des Spielendes für Spannung zu sorgen. Außerdem bietet er sich für die Steuerung mit dem *MediBalance Pro* an, da bei passendem Leveldesign Neigen nach links und rechts allein ausreicht, um den Abschnitt erfolgreich zu durchqueren. Durch Variation in der Pfadbreite und der Geschwindigkeit, mit der der Spieler sich bewegt, kann der Schwierigkeitsgrad gut verändert werden. Für den Fall, dass das Spiel nicht nur mit dem Balanceboard gespielt wird, können außerdem Hindernisse eingebaut werden, die übersprungen werden müssen.

Setting und Story

Das Spiel spielt in einer aus Süßigkeiten bestehenden Fantasiewelt. Diese Welt macht es möglich, eine niedliche, bunte Umgebung zu schaffen, die für eine entspannte und fröhliche Atmosphäre sorgt. Candyland ist eine verzauberte Welt, die größtenteils auf in der Luft schwebenden Keksen gebaut ist.

Alle Süßigkeiten in Candyland wurden von der großen Candymaschine hergestellt und bevölkern nun die Welt.

Doch das bunte Schlaraffenland wird von einem Salzlakritz-Bösewicht bedroht. Dieser plant, ganz Candyland in ein Lakritzland zu verwandeln. Kurz nach der Ankunft des Helden manipuliert er die Candymaschine, sodass nun keine neuen Süßigkeiten mehr hergestellt werden. Der Spieler schlüpft also in die Rolle des jungen Helden und macht sich auf, seine neue Heimat zu retten. Dabei begegnet er einem weiteren Bewohner, den er aus einer misslichen Lage befreit und der sich ihm daraufhin anschließt.

Vor der Konfrontation mit dem Bösewicht müssen die beiden einen weiten Weg durch ganz Candyland zurücklegen, der sie vor viele verschiedene Herausforderungen stellt.

Spielablauf

Der Protagonist startet in einem Dorf aus Cupcake Häusern. Dort begegnet er einem freundlichen Bewohner Candylands, der ihm Hinweise zur Bedienung des Spiels gibt und ihn durch das kurze Einführungslevel leitet. Von dort betritt der Spieler den ersten Korridor und trifft auf einen Händler, der ihm von dem Nutzen der Schokolinsen erzählt und ihm das Reisesystem erklärt. Außerdem erfährt der Spieler mehr zu der Gefahr, die Candyland bedroht.

Der Spieler durchquert den Schieberätselbereich und den zweiten Korridor und trifft im ersten Ebenenpuzzle auf seinen zukünftigen Begleiter. Dieser wurde von einer Bonbonfee hierher getragen, kann aber ohne Hilfe nicht weitergehen. Der Spieler entscheidet sich zu helfen und beide durchqueren zusammen die Ebenenpuzzle und danach den Rutschlevel-Bereich.

Im letzten Korridor verabschiedet sich der Begleiter, denn er sei im Kampf gegen die Lakritze wohl keine große Unterstützung und möchte stattdessen bei der Reparatur der Candymaschine helfen. Der Protagonist steht nun vor einem verschlossenen Tor, das den Weg zur Lakritzmaschine versperrt. Der Händler in der Nähe bietet dem Protagonisten an, das Tor für ihn zu öffnen. Dafür benötigt er jedoch die Krokantschlüssel der anderen Händler. Der Protagonist muss also zu ihnen zurück reisen. Ein jeder verlangt daraufhin einen kleinen Gefallen bevor er seinen Schlüssel aushändigt.

Mit den erlangten Schlüsseln kehrt der Spieler zum verschlossenen Tor zurück und spricht mit dem Händler. Dieser gibt daraufhin seine Tarnung auf und entpuppt sich als böse Lakritze, die sich hämisch für die Schlüssel bedankt, die sie zur Vervollständigung ihrer Maschine braucht. Mit ihrem Spazierstock öffnet sie das Tor und fliegt davon. Der Protagonist nimmt die Verfolgung auf und durchquert nun den letzten Abschnitt vor dem Bosskampf. Verliert er die Lakritze aus den Augen, muss der Abschnitt wiederholt werden.

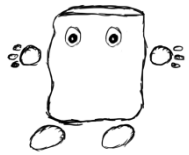
Hat der Held es geschafft, die Lakritze bis zu ihrer Basis zu verfolgen, kommt das Finale.

Um ihre teuflische Lakritzmaschine anzutreiben, hat die Lakritze fliegende Bonbonfeen gesammelt und eingesperrt. Um die Lakritze zu besiegen muss der Protagonist daher die Bonbonfeen befreien. Sobald ihre Energieversorgung abbricht, versucht sie zu fliehen, indem sie ihren eingebauten Schleudersitz betätigt. Aber der funktioniert nicht ganz planmäßig und die Lakritze befördert sich damit selbst in einen riesigen Wackelpudding und bleibt darin gefangen.

Spielelemente

Die spielbaren Charaktere

Um die Rätsel komplexer zu machen, wird der Spieler zwischen Haupt- und Nebencharakter wechseln können. Die beiden sollen sich gegenseitig weiterhelfen können und müssen.



Durch unterschiedliche Fähigkeiten der Charaktere wird der Spieler dazu gebracht, auf andere Art und Weise an die vielen Rätsel heranzugehen und sie zu lösen.

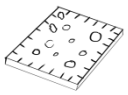
Der Hauptcharakter ähnelt einem Marshmallow und kann springen und Blöcke verschieben.

Der zweite Charakter kann bestimmte Hindernisse zerstören und aufgrund seiner geringen Größe an andere Orte mit niedrigen Decken gelangen. Er kann nicht springen und ist auch nicht stark genug, Blöcke zu verschieben. Er ist im Verhältnis zum Hauptcharakter sehr klein und sieht aus wie eine Pyramide aus Zucker.

Spielobjekte

Candyland besteht aus verschiedenen Süßigkeiten mit unterschiedlichen Funktionen. Diese Spielobjekte sollen klar voneinander unterscheidbar sein, sodass der Spieler nach kurzer Zeit direkt weiß, welches Objekt welche Funktion hat. Um dem Setting gerecht zu werden, wurde entschieden, die Formen sehr einfach zu halten und vor allem mit Quadern zu arbeiten.

Plattformen



Plattformen sind begehbare Flächen und bilden die Grundlage der Level. Sie sehen aus wie Kekse und es gibt sie in drei verschiedenen Größen. Sich bewegende Plattformen pendeln zwischen 2 Orten. Sie sehen ansonsten aus wie normale Plattformen und es gibt sie in kleiner und mittlerer Größe.

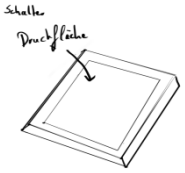
Plattformen gibt es in Candyland in vielen verschiedenen Varianten.

Rutschige Plattformen sind teilweise nur für Objekte, teilweise jedoch auch für den Spieler rutschig. Hier ist weder das Wechseln der Richtung noch Anhalten während des Rutschens möglich, sondern es muss ein Hindernis getroffen oder die rutschige Fläche verlassen werden, um anzuhalten und die Richtung zu wechseln. Rutschige Plattformen sehen aus wie Wassereis. Dabei sind die Plattformen, die nur für Blöcke rutschig sind, blau. Die Plattformen, die auch den Spieler rutschen lassen, sind grün.

Zerbrechende Plattformen sehen bereits brüchig aus und zerbröckeln kurze Zeit, nachdem sie das erste Mal betreten wurden. Das Zerbrechen ist animiert und wird mit einem passenden Geräusch deutlich signalisiert.

Teleportationsplattformen ermöglichen dem Spieler, von einem Ort an einen anderen Ort in beliebiger Entfernung zu springen. Jede dieser Plattformen hat jedoch ein fest definiertes Ziel. Teleportationsplattformen sehen aus wie normale Plattformen, über ihnen schwebt aber eine den Raum krümmende Sphäre.

Schalter

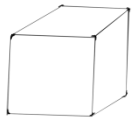


Sehr häufig kommen zudem Schalter aus Schokolade vor. Auch bei ihnen handelt es sich letztendlich um spezielle Plattformen. Besteht die Druckfläche aus weißer Schokolade, so handelt es sich um einen permanenten Schalter, der nur ein einziges Mal aktiviert werden muss und seinen Zustand danach nicht mehr ändert. Ist die Druckfläche aus dunkler Schokolade, handelt es sich um einen Schalter, der nur bei Belastung aktiv bleibt und wenn auf der Druckfläche ein Ziffernblatt zu sehen ist, ist der Schalter nur für begrenzte Zeit aktiv. Bei jedem Schalter verändert sich die Druckfläche, wenn er aktiv ist. Zusätzlich gibt es ein Geräusch, das signalisiert, wenn ein Schalter aktiviert oder deaktiviert wird. Bei Zeitschaltern erklingt zudem ein Ticken, während sie aktiv sind.

Die Schalter können zu Schaltergruppen zusammengefasst werden. Jeder Gruppe wird ein Objekt zugeordnet, das dann erscheint oder verschwindet, wenn alle Schalter der Gruppe aktiviert wurden. Eine spezielle Form der Gruppe erfordert außerdem, dass die Schalter in einer festen Reihenfolge aktiviert werden. Macht der Spieler einen Fehler, werden alle Schalter der Gruppe wieder deaktiviert.

Die speziellen Plattformen gibt es alle nur in der Größe der kleinsten Plattform.

Hindernisse



Statische Hindernisse sind würfelförmige Blöcke aus Karamell. Es gibt sie in zwei Größen, einmal mit einer Grundfläche in der Größe einer kleinen Plattform, einmal passend zu einer mittelgroßen Plattform. Die Hindernisse sind so hoch, dass der Spieler nicht vom Boden aus auf sie springen kann.

Es gibt allerdings auch halbhohe kleine Hindernisse, bei denen das möglich ist.

Variationen der Hindernisse sind wie Zuckerwürfel aussehende zerbrechliche und verschiebbare Blöcke, wobei der zerbrechliche Block sichtbare Risse hat. Verschiebbare Hindernisse können genutzt werden, um Schalter zu aktivieren oder Wege zu erschließen. Zerbrechliche Blöcke versperren den Weg und können vom Helfer zerstört werden. Beim Zerbrechen wird eine Animation abgespielt und ein passendes Geräusch abgespielt.

Schokolinsen

Die Währung in Candyland sind Schokolinsen, welche vom Spieler überall gefunden und eingesammelt werden können. Beim Einsammeln soll ein Soundeffekt abgespielt werden. Schokolinsen werden von einem Leuchtreif umgeben, der sie besser sichtbar macht.

NPCs

Bei den NPCs steht im Vordergrund, dem Spieler möglichst einfach unterschiedliche Features zu bieten, die für ein hohes Spielvergnügen und wenig Frustration sorgen. Außerdem sollen die Modelle möglichst nachvollziehbar sein.

Um den Schokolinsen einen Nutzen zu geben, gibt es Händler, die dem Spieler Bonuscontent anbieten. Das soll den Spieler dazu antreiben, alle Schokolinsen im Spiel zu sammeln.

Des Weiteren soll zu späteren Zeitpunkten an besuchte Orte zurückgekehrt werden können, um alle Spielinhalte zu erkunden. Auch diese Funktion übernimmt der Händler. Zusätzlich dazu sollte ursprünglich direkt beim Händler gespeichert werden können. Dieses Speichersystem wurde jedoch später verworfen. Stattdessen wurde entschieden, dass das Spiel sich automatisch speichern soll.

Da der Händler eine Auswahl von Extras anbietet, soll er einer Schultüte ähneln, die typischerweise mit schönen Dingen gefüllt ist.

Um weniger erfahrene Spieler an das Spiel heranzuführen und um neue Spielelemente zu erklären, gibt es einen Tutor, der wie das Acagamics-Maskottchen aussieht.

Der Endgegner ist ein Lakritzmännchen, das Candyland zerstören will.

Motivation für den Spieler

Zunächst soll das Erfolgserlebnis, wenn ein Rätsel gelöst wurde, motivieren. Die Rätsel sollen stets mit dem zur Verfügung stehenden Wissen lösbar sein, um den Spieler nicht zu frustrieren. Gleichzeitig sollen die Spieler aber nicht unterfordert werden. Dafür nehmen Schwierigkeitsgrad und Komplexität der Rätsel im Laufe des Spiels zu. Es gibt jedoch auch immer wieder Hinweise, die der Spieler sich bei Bedarf ansehen kann.

Um den Spieler dazu zu bewegen, die Welt weiter zu erkunden und neue Wege zu beschreiten, sind im Spiel Schokolinsen verteilt, die bei dem Ingame-Händler gegen zusätzliche Inhalte eingetauscht werden können. Jeder Händler bietet sein eigenes Bonuslevel als größte Belohnung an.

Die Geschichte soll dazu motivieren, das Spiel durchzuspielen. Die Anzeige der eingesammelten Schokolinsen, die zusätzlich angibt, wie viele der Linsen es insgesamt im Spiel gibt, soll Spieler langfristig motivieren. Das wirkt besonders bei den Spielern, die ein Spiel gerne vollständig durchspielen.

Der Spieler kann jederzeit an bereits besuchte Orte zurückkehren. So wird gewährleistet, dass er wirklich die Möglichkeit hat, alle Schokolinsen einzusammeln, wenn er sich denn dazu entscheidet.

User Interface

Im Spiel soll angezeigt werden, wie viele Schokolinsen es im Spiel gibt und wie viele davon bereits gesammelt wurden. Außerdem bekommt der Spieler Hinweise zu den Tasten eingeblendet mit denen der Wechsel zwischen den Spielern oder ein Level Reset erfolgt und die Kartenansicht oder das Hauptmenü geöffnet wird.

Es soll einen Titelbilschirm, ein Hauptmenü, Optionen, Extras, Credits und einen Ladebildschirm geben. Im Spiel sind Dialoge und ein Menü für Händler NPCs mit einem Einkaufs- und einem Reisebildschirm geplant.

Zielgruppe

Das Spiel richtet sich grundsätzlich an zwei verschiedene Zielgruppen.

Einerseits sind das Gelegenheitsspieler aller Altersgruppen. Aufgrund des Tutorials und der ausführlichen Erklärungen können auch Personen, die sonst wenig mit Videospielen zu tun haben, leicht in das Spiel einsteigen. Auf das direkte Bekämpfen von Gegnern wird verzichtet, um das Spiel besonders älteren und sehr jungen Spielern besser zugänglich zu machen.

Andererseits soll das Spiel für Therapie mit dem *MediBalance Pro* eingesetzt werden können. Dafür wurde eine eigene Version des Spiels geplant. Diese zeichnet sich durch klar definierte, kurze Spielabschnitte und die Möglichkeit, den Schwierigkeitsgrad nach Belieben anzupas-

sen, aus. Für das Balanceboard eine eigene Version zu entwickeln, ist ferner sinnvoll, da die Steuerungsmöglichkeiten dort stark eingeschränkt sind. Im Bestfall sollte sich das gesamte Spiel nur durch Gewichtsverlagerung spielen lassen. Das ist bei dem Hauptspiel aber nicht der Fall.

Technische Überlegungen

Zusätzlich zu den inhaltlichen Aspekten musste auch die interne Struktur des Spiels geplant werden. Darauf wird in diesem Abschnitt eingegangen.

Struktureller Aufbau des Spiels

Die Welt wird aufgeteilt in Gebiete, die wiederum aus mehreren Leveln bestehen. Dabei soll eine Klasse die gesamte Spielwelt verwalten und Kontrolle über die spielbaren Charaktere und Gebiete bekommen. Diese Gebiete kontrollieren wiederum ihre Level.

Es sollen immer nur die Objekte in dem Gebiet, in dem die aktive Spielfigur sich befindet, und in den zwei angrenzenden Gebieten gezeichnet werden. Das Prüfen auf Kollisionen und die Aktualisierung der Spielobjekte sollen zudem nur in dem Level, in dem die aktive Spielfigur sich befindet, und in dem zuletzt besuchten Level, erfolgen. Während zwei Spielfiguren zur Verfügung stehen, soll weiterhin nur das Level aktualisiert werden, in dem sich die Figur befindet, die zum jeweiligen Zeitpunkt aktiv ist.

Es ist wichtig, dass die Level sehr klein sind. Sie enthalten nie mehr als ein oder zwei in sich abgeschlossene Rätsel. Damit ist sichergestellt, dass die Anzahl vorgenommener Kollisionstests, die sehr rechenintensiv sind, niedrig bleibt.

Eine Klasse soll alle veränderlichen Variablen, die viele Objekte betreffen, zentral verwalten. Alle Spielobjekte bekommen dabei Zugriff auf die Instanz dieser Klasse.

Eine weitere Klasse soll alle Konstanten des Spiels enthalten und global zugreifbar sein. Weiterhin soll es eine Klasse geben, die dafür zuständig ist, alle Assets, also Texturen, Modelle, Effects (Fußnote: Shader) und Musik und Soundeffekte, zentral zu laden und Referenzen auf die geladenen Objekte an die entsprechenden Spielelemente zu geben.

Kollision

Die Kollisionserkennung soll auf der in XNA vorhandenen Boundingbox-Kollision (Fußnote: AABB) basieren. Direkt nach der Kollisionserkennung soll dabei von den beteiligten Objekten auf die Kollision reagiert werden.

Steuerung

Die Steuerung lässt sich auf wenige Eingaben beschränken. Die Kamera soll mit der Maus gesteuert, W, A, S und D zur Fortbewegung genutzt werden. Mit der Leertaste sollen NPCs

angesprochen werden können und es ist geplant, das Gespräch mit Leertaste oder Enter fortzuführen. Außerdem soll die Leertaste die Spezialfähigkeit der aktiven Spielfigur auslösen, also entweder Springen oder das Zerbrechen von Blöcken. Nachdem der Spieler den zweiten Charakter kennengelernt hat, soll es möglich sein, zwischen den beiden Figuren mit C zu wechseln. Mit R soll ein Reset des aktiven Levels ausgelöst werden. M ist dafür vorgesehen, die Kartenansicht zu öffnen und ESC soll das Menü öffnen. Die Hintergrundmusik soll mit L ausgeschaltet werden können, Soundeffekte dagegen sollen nicht beeinflusst werden, da sie dem Spieler wichtige Rückmeldung zu seinen Aktionen in der Spielwelt geben.

Auch mit Gamepad soll gesteuert werden können.

Hierbei soll der Spieler mit dem linken Control-Stick die Spielfigur bewegen und mit dem Rechten die Kamera steuern können.

A ist zum Benutzen der Spezialfähigkeit, Ansprechen, sowie zum Bestätigen in Menüs gedacht. Steuerkreuz-Unten soll den Spieler wechseln und Steuerkreuz -Oben die Karte anzeigen. Die Start Taste soll das Menu öffnen. Die beiden Trigger sollen, wenn sie gleichzeitig gedrückt werden, das Level in den Ausgangszustand zurücksetzen. Das Ausschalten der Musik wird auf Steuerkreuz-Rechts gelegt.

Für die spezielle *MediBalance Pro*-Version soll dazu auch Steuerung nur mithilfe des Balanceboards ermöglicht werden. Dabei ersetzt das Balanceboard den linken Control-Stick. Auswahl des Levels und Anpassung des Levels an die Bedürfnisse des Spielers soll mit der Tastatur möglich sein.

Grafik und Modelle

Als 3D-Grafiksoftware zur Erstellung der Modelle und Animationen sollte die Freeware Blender (Version 2.66a, in Fußnote) verwendet werden. Aufgrund des gewählten niedlichen Settings und des geringen Wissensstandes bezüglich 3D-Modellierung sollten die Spielelemente mit möglichst einfachen Modellen dargestellt werden können.

Die Texturierung der Modelle sollte teilweise in *Blender* selbst durch *Texture Paint* oder aber mit dem Bildbearbeitungsprogramm *Photoshop CS5* umgesetzt werden.

Passend zum Setting soll ein Toon-Shader das comicartige Aussehen der Modelle unterstreichen.

Levelbau

Zum Leveldesign soll die Software *Tiled* benutzt werden. Sie kann zum Erstellen von kachelbasierten Leveln genutzt werden und bietet die Möglichkeit, Objekte auf verschiedenen

Ebenen in ein Raster einzusetzen und sie mit verschiedenen Attributen zu versehen. Die so erstellte Karte lässt sich dann als .tmx-Datei exportieren.

Diese Dateien sollen von einem selbst geschriebenen Editor eingelesen, in ihre Gebiete eingeordnet und mit noch fehlenden Informationen versehen werden. Die gesamte Welt soll dann als .xml-Datei exportiert und vom Spiel zur Laufzeit eingelesen werden können.

Zustandsverwaltung und besondere Spielfunktionen

Das Speichern soll beim Betreten eines neuen Levels und beim Wechsel ins Hauptmenü automatisch geschehen. Da die einzelnen Level sehr klein gehalten werden, gehen so bei einem Absturz oder beim Beenden des Spiels kaum Daten verloren. Dass der Spieler auf diese Weise einen kleinen Teil seines Fortschritts verliert, wurde zugunsten der guten Umsetzbarkeit und der Bequemlichkeit und Sicherheit, die ein automatischer Speicherprozess dem Spieler bieten, hingenommen.

Fällt der Spieler von einer Plattform oder drückt R, um das aktive Level auf den Ausgangszustand zurückzusetzen, so soll er am letzten Speicherpunkt erscheinen. Alle Elemente des Levels sollen auf ihren Ausgangszustand zurückgesetzt werden. Bereits eingesammelte Schokolinsen sollen jedoch nicht verloren gehen und auch einige der Ereignisse, die die Geschichte des Spiels vorantreiben, sollen sich nicht wiederholen. Das gilt besonders für die Einführung neuer Spielelemente, da es den Spielfluss stark stören würde, wenn der Spieler sich nach jedem Fehler alle Erklärungen zum aktuellen Abschnitt erneut anhören müsste.

Bei der Verfolgungsjagd sowie im letzten Spielabschnitt darf der Spieler sich nicht zu weit von dem Endgegner entfernen. Ist die Distanz zwischen Spielfigur und Lakritze zu groß, soll das Spiel daher automatisch auf den letzten Speicherpunkt zurückgesetzt werden, sodass der Spieler das Level von vorne beginnen muss. Damit der Spieler diese neue Bedingung bemerkt, soll er einerseits von einem NPC darauf hingewiesen werden und andererseits soll eine Anzeige erscheinen, die dem Spieler mitteilt, wenn die Entfernung sich ändert.

Projektplanung

Das Online-Ticket-System von ClockingIT (www.clockingit.com, Fußnote) wurde zur Aufgabenverteilung und -überprüfung vorgesehen und gibt Überblick über den Entwicklungsfortschritt. Auch Bugs sollten dort in Tickets dokumentiert werden. Das Projekt an sich wird auf GitHub (www.github.com, Fußnote) verwaltet und aktualisiert.

In regelmäßigen Treffen wurden Aufgaben verteilt, Probleme und Vorschläge besprochen und teilweise auch gemeinsam entwickelt.

Umsetzung

Auf der Grundlage der bisher beschriebenen Überlegungen wurde das Spiel umgesetzt. Die Details zur Umsetzung sind in diesem Abschnitt festgehalten.

Struktureller Aufbau des Spiels

Kern des Spiels ist der *SceneManager*. Er ist für die *Update*- und die *Draw*-Methode des Spiels zuständig. Über dem Scenemanager steht der Screenmanager, der die verschiedenen Gamescreens verwaltet.

Die Klasse *UpdateInfo* enthält Variablen, die für viele verschiedene Klassen zur gleichen Zeit relevant sind und die sich während des Spiels häufig ändern. Aufgrund dessen hat jede Klasse eine Referenz auf die *UpdateInfo.* Die *GameConstants* Klasse ist ähnlich der *UpdateInfo*. Sie enthält jedoch konstante Parameter die vor allem zum Debbugen und zur Anpassung von Details wie der Lichtrichtung und -farbe oder der Bewegungsgeschwindigkeit der Spielobjekte benötigt werden. Der *AssetManager* lädt alle Modelle und Texturen und speichert jeweils eine Instanz. Alle Objekte des Spiels bekommen dann nur noch Referenzen auf die Instanz, die sie benötigen, übergeben.

Die Objekte selbst erben alle von der Klasse *GameObjects*. Sie werden weiter unterteilt in *DynamicObject, Platform* und *Obstacle.* Plattformen und Obstacles haben jeweils noch diverse Kindklassen. Von *DynamicObject* erben auch spielbare Charaktere und NPCs. Die Spielobjekte implementieren ihre Reaktion auf Kollisionen und ihr Verhalten bei einem Update jeweils selbst.

Die Welt besteht aus Gebieten, die in einer Liste der *SceneManager*-Instanz des Spiels organisiert sind, und die Referenzen auf ihre Level besitzen. Die Level wiederum besitzen Referenzen auf alle Objekte, die sich in ihnen befinden.

Der *SceneManager* besitzt außerdem je eine Instanz der Klasse *InputManager* und der beiden spielbaren Charaktere. Um die Nutzereingaben umzusetzen, nutzt der *SceneManager* Methoden des *InputManagers*, die Referenzen auf die beiden spielbaren Charaktere benötigen.

Physik

In jedem Update-Zyklus wird zunächst die Eingabe durch den Spieler verarbeitet. Dann wird die Spieleranimation aktualisiert. Falls die Spielfigur momentan fällt, wird sie nach unten beschleunigt und in jedem Fall wird die Kamera aktualisiert. Anschließend wird der aktive Spieler mit allen Objekten des Levels, in dem er sich befindet, auf Kollisionen überprüft. Befindet sich der zweite spielbare Charakter auch in dem Level, so wird die Prozedur auch für ihn durchgeführt. Wird dabei eine Kollision festgestellt, wird als erstes geprüft, in welcher Position sich der Spieler vor dem Update relativ zu dem Objekt befunden hat. Dann wird der

Spieler dementsprechend aus dem Objekt geschoben, mit dem er kollidiert. Ob weitere Änderungen an den Objektzuständen vorgenommen werden, hängt dann von der Art der Kollision ab, also mit was und an welcher Stelle die Kollision stattfand. Anschließend werden alle dynamischen Objekte aktualisiert und auf Kollisionen mit den Objekten des aktuellen Levels geprüft. Analog zu der des Spielers funktioniert auch die Behandlung von Kollisionen bei diesen Objekten.

Spielbare Charaktere und die Steuerung

Die *Playable*-Klasse ist die Basisklasse für die beiden spielbaren Charaktere. Sie hat ihre eigene Kollisionslogik und Updatelogik. Die Kamera ist außerdem hier eingebaut, da beide Spieler jeweils eine eigene Kamera besitzen. Die Kamera befindet sich immer ein Stück hinter und über dem Spieler. Bewegung wird in Bezug auf die Kamerablickrichtung umgesetzt, ohne dabei die höhere Lage der Kamera zu beachten. Eine Bewegung nach vorne bedeutet also eine Bewegung in die Kamerablickrichtung, zur Seite eine Bewegung orthogonal zu dieser Blickrichtung. Grundlegende Funktionen zum Bewegen des Spielers und der Kamera sind vorhanden und werden von dem *InputManager* bei der Verarbeitung der Eingabe genutzt. Der Inputmanager transformiert letztendlich die Eingabe in von der *Playable*-Klasse nutzbare Daten. So wird zum Beispiel hier die Eingabe der Pfeiltasten in Weltkoordinaten umgerechnet, indem die Kamerablickrichtung einbezogen wird. Der *InputManager* bezieht aus der *UpdateInfo* auch Informationen über den Zustand des Spiels und passt die Eingabe dementsprechend an. Ist der Spieler etwa in einer Dialogsequenz oder ist Bewegung aktuell nicht erlaubt, wird die Eingabe ignoriert.

Wie bei den Spielelementen beschrieben, gibt es zwei spielbare Charaktere mit unterschiedlichen Fähigkeiten. Der Hauptcharakter ist ein Marshmallow mit Händen, Füßen und einem Gesicht und er kann springen sowie Blöcke verschieben. Der zweite spielbare Charakter ist eine Zuckerpyramide mit Füßen und Gesicht, die durch kleine Gänge laufen und zerbrechliche Hindernisse zerstören kann.

Die Interaktionen mit anderen Objekten werden von den Kollisionsmethoden gesteuert, die jedes Objekt, das auf eine Kollision reagieren kann, besitzt. Die spielbaren Charaktere selbst lösen die Prozesse nur durch ihre Prüfung auf Kollision aus.

Feedback wird mit Hilfe von Lauf- und Aktionsanimationen gegeben.

Zur Steuerung werden genau die Eingabetasten und -geräte verwendet, die in der Planung definiert sind. Der Spieler muss jedoch nach Erscheinen eines Dialoges kurz warten, bis er mit der Leertaste fortfahren kann. Nutzt er Enter, ist das nicht der Fall. Diese Einschränkung ergab sich aus dem Umstand, dass einige Testspieler Dialoge versehentlich geschlossen haben, weil sie impulsiv auf die Leertaste drückten um zu Springen.

Die Controller- und *MediBalance Pro*-Steuerung sind allerdings noch nicht vollständig implementiert und stehen daher noch nicht zur Verfügung.

Musik

Für die Hintergrundmusik sowie die Soundeffekte wurde nach freien Ressourcen gesucht, die die jeweils gewünschte Stimmung widerspiegeln. So sollte die Hintergrundmusik vor allem zum Setting und zur Atmosphäre passen. Um das Gefühl des Zeitdrucks akustisch zu unterstützen, ändert sich die Musik im Verfolgungs- und im Bosslevel und wird deutlich hektischer.

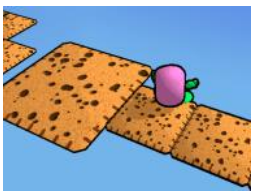
Für die verschiedenen Interaktionen wurden Soundeffekte gewählt, die sie allein am Ton erkennbar und unterscheidbar machen. Der Ton ist hierbei ein wichtiges Mittel, um dem Spieler die Auswirkungen seiner Tätigkeiten zu vermitteln.

Für die Implementierung wurden die XNA-eigenen Media- und Audio-Frameworks verwendet.

Spielobjekte

Das Spiel besteht aus diversen Bausteinen, die für die unterschiedlichen Rätsel alle eine eigene Funktion erfüllen.

Plattformen



Plattformen bilden die Grundlage des Spiels. Da hierfür ein möglichst ebenes und quadratisches Design am besten erschien, um sinnvolle Level zu erstellen und diverse Mechaniken wie das normale Fortbewegen oder auch das Verschieben von Objekten zu ermöglichen, wurden, wie bereits beschrieben, Kekse als Vorbild für das Modell genommen.

Zerbrechende Plattformen wirken hierbei brüchig und beim Betreten der Plattformen wird eine passende Animation abgespielt. Das geht einher mit einem entsprechenden Geräusch. Ist die Animation beendet, wird die Plattform unsichtbar und auch nicht mehr in die Updates der Szene einbezogen.

Die sich bewegenden Plattformen unterscheiden sich optisch nicht von den normalen Plattformen, sondern bewegen sich lediglich zwischen den für sie definierten Start- und Endpunkten hin und her. Dabei wird die Plattform immer zu einem der beiden Punkte bewegt. Kommt sie dort an, wird die Bewegungsrichtung umgekehrt und sie bewegt sich zurück.

Zusätzlich gibt es Teleportationsplattformen, die an der Sphäre erkennbar sind, die über ihnen schwebt. Sie bieten Zugang zu Bonusleveln und eine dieser Plattformen führt in das letzte Level.

Auch die rutschigen Plattformen wurden in beiden Varianten implementiert. Ihnen wird ein Material zugewiesen, das einen hohen spekularen Anteil besitzt und glänzen daher deutlich. Sobald sich der Schwerpunkt eines Hindernisses oder einer Spielfigur auf einer rutschigen Plattform befindet, wird dies registriert. Wird der Charakter nun bewegt oder das Hindernis angestoßen, bleibt der Bewegungsvektor unverändert, bis das Objekt von einem Hindernis gebremst wird. Wird ein bewegliches Hindernis, das auf einer rutschigen Plattform ruht, vom Hauptcharakter oder einem anderen sich bewegenden Hindernis angestoßen, wird geprüft, aus welcher Richtung der Stoß kam und dann erhält das angestoßene Hindernis einen Bewegungsvektor in die entgegengesetzte Richtung, während die Bewegung des stoßenden Objekts beendet wird.

Schalter

Die Schalter sollen sich in der Form nicht von Plattformen unterscheiden. Aus dem Grund wird das Plattformmodell verwendet, bekommt aber passende Texturen zugewiesen, um den Eindruck von Schokolade zu erwecken. Auch bei Schaltern wurde dem verwendeten Material ein spekularer Anteil hinzugefügt. Je nach Art des Schalters wird eine andere Textur genutzt, wobei sich hauptsächlich die Farbe und, im Fall von Zeitschaltern, das Muster unterscheiden. Beim Aktivieren und Deaktivieren der Schalter ändert sich die Textur deutlich und es ertönt in beiden Fällen ein Geräusch, um dem Spieler zu signalisieren, dass etwas passiert ist. Zeitschalter lösen außerdem ein Ticken aus, das so lange anhält, wie sie aktiv sind.

Schalter können Blöcke und Plattformen erscheinen und verschwinden lassen und so neue Wege für den Spieler öffnen.

Ein Schalter wird dann aktiviert, wenn ein spielbarer Charakter oder ein verschiebbares Hindernis mit ihm kollidiert und sich der Schwerpunkt dieses aktivierenden Objekts auf dem Schalter befindet.

Damit ein Schalter Veränderungen in der Welt auslösen kann, muss er Teil einer Schaltergruppe sein. Diese können auch nur einen einzigen Schalter beinhalten, kümmern sich aber in jedem Fall um das Verknüpfen von Schaltern mit den Objekten, die sie manipulieren. Schaltergruppen werden in einer separaten XML-Datei definiert und nach der Spielwelt eingelesen. Es gibt keinen Editor für Schaltergruppen, da nur angegeben werden muss, welche Schalter in der Gruppe sein sollen, welches Objekt manipuliert werden soll und ob die Gruppe sortiert ist oder nicht. Bei einer sortierten Gruppe müssen die Schalter in der Reihenfolge betätigt werden, in der sie in der XML-Datei gelistet sind.

Schaltergruppen besitzen Referenzen auf das ihnen zugeordnete Objekt und auf alle ihre Schalter, um sie alle bei Bedarf zurücksetzen zu können. Außerdem besitzt jeder Schalter eine Referenz auf seine Gruppe und teilt ihr so mit, wenn er aktiviert oder deaktiviert wird.

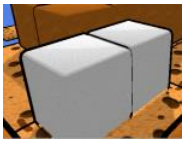
Die Schaltergruppe prüft dann, ob ihre Bedingungen erfüllt sind und löst entsprechendes Verhalten bei den ihr zugeordneten Objekten aus.

Ein Schalter kann mehreren Schaltergruppen zugeordnet sein.

Hindernisse

Als Hindernisse werden würfelförmige Blöcke mit abgerundeten Kanten genutzt, die von der

Größe her an die Plattformen angepasst wurden. Dies dient vor allem dem Zweck, dass sie vielseitig für die unterschiedlichen Rätsel einsetzbar sind. Je nach Art des Hindernisses unterscheiden sie sich in Textur und Größe.



So sind statische Blöcke mit einer Karamelltextur versehen, verschiebbare und zerbrechliche Hindernisse hingegen haben eine Zuckertextur. Der zerbrechliche Block weist dabei auch klare Risse auf. Zudem ändert sich die Farbe seiner Umrandung, wenn es möglich ist, ihn zu zerbrechen. Der Nebencharakter kann den Block zerbrechen, wenn, während er mit ihm kollidiert, die Aktionstaste gedrückt wird. Bei den beweglichen Hindernissen wird zwischen einer Kollision von

oben oder unten und Kollisionen von den Seiten unterschieden. Nur eine seitliche Kollision mit dem Hauptcharakter führt zum Verschieben, wobei noch getestet wird aus welcher der vier Himmelsrichtungen die Kollision kam.

Schokolinsen

Als Belohnungssystem wurden Schokolinsen eingebaut, die der Spieler überall im Spiel finden und sammeln kann. Mit diesen ist es dem Spieler möglich beim Händler Bonus-Inhalte wie Concept Art oder Bonuslevel zu erwerben. Hat der Spieler eine Schokolinse einmal eingesammelt, so verliert er sie bei einem Reset nicht mehr. Dafür ist der *BonusTracker* zuständig, der außerdem die Schokolinsen-Anzeige auf dem Bildschirm mit der Anzahl gefundener und insgesamt vorhandener Linsen versorgt. Kollidiert die aktive Spielfigur mit einer Schokolinse, so teilt sie das dem Tracker mit, der dann speichert, dass die Linse gefunden wurde.

Schokolinsen besitzen ein Billboard, das sie mit einem grünen Ring umgibt. Damit sind sie für den Spieler besser zu finden. Zudem ändert sich das Billboard für die Kartenansicht. Der Ring wird dicker, damit er auch aus großer Entfernung noch gut sichtbar ist.

Leveltypen

Die einzelnen Leveltypen orientieren sich sehr nah an den geplanten Varianten.

Die Rätsel nutzen die bereits beschriebenen Spielelemente und setzen sie so ein, wie es im Abschnitt *Planung* definiert ist.

In allen Leveln ist eine Übersichtskarte verfügbar, bei der es sich um eine orthogonale Aufsicht auf die Szene handelt. Sie soll es dem Spieler ermöglichen, sich einen Überblick über seine Position und die Position der Elemente, die er für ein Rätsel benötigen könnte, zu verschaffen.

[BILDER!]

Beim Verfolgungslevel, welches als eine Art “Vorspann” des Bosslevels fungiert, ist die Steuerung des Spielers so eingeschränkt, sodass er sich automatisch nach vorne bewegt. Die Richtung kann so lediglich diagonal nach links (mit A) oder diagonal nach rechts (mit D) geändert werden. Zusätzlich ist es jedoch auch möglich mit S anzuhalten und wie zuvor mit der Leertaste zu springen.

Der Spieler verfolgt hier den Endgegner zu seinem Unterschlupf und darf ihn dabei nicht aus den Augen verlieren. Ist die Distanz zwischen Spielfigur und Gegner zu hoch, muss der Spieler das Level neu starten.

Eine Aktion leitet diesen Abschnitt ein, erklärt die Eigenschaften der neuen Steuerung und stellt die Steuerung so um, dass der Bewegungsvektor in die gewünschte Richtung dauerhaft, statt nur auf Tastendruck, gegeben ist. Zudem wird die Kamera von der Bewegungsrichtung entkoppelt und die Distanzanzeige und -überprüfung werden aktiviert. Dabei wird der euklidische Abstand zwischen aktivem Spieler und der Lakritze berechnet und in eine von fünf Kategorien eingeteilt. Es wird unterschieden zwischen einer sicheren, einer ungefährlichen, einer gefährlichen und einer kritischen Distanz. Dazu kommt die Distanz, ab der der Abstand zu groß geworden ist.

Der letzte Abschnitt, das Bosslevel, bedient sich bei allen verfügbaren Spielelementen. Es wurde allerdings aus Zeitmangel noch nicht passend zur Geschichte des Spiels umgesetzt.

NPCs

Die NPCs sind bis auf wenige Details so umgesetzt worden, wie sie im Abschnitt zur Planung bereits beschrieben wurden.

So ist der Endgegner eine Lakritzfigur mit Zylinder und Monokel, die auf ihrem schwebenden Sessel sitzt, was ihre Arroganz und Überheblichkeit verdeutlichen soll. Die Interaktion mit diesem Charakter beschränkt sich auf Dialoge und Verfolgung und die Implementierung basiert komplett auf dem Aktions-System.

Der Händler wird durch eine Schultüte mit Augen und Füßen dargestellt. Die geplanten Interaktionsmöglichkeiten sind alle umgesetzt worden, sodass der Spieler durch Ansprechen mit der Aktionstaste mit dem Händler über Neuigkeiten sprechen, an bereits besuchte Orte reisen und Bonus-Inhalte kaufen kann. Wenn der Spieler den Händler ansprechen kann, wird das durch eine Änderung in der Farbe der Umrandung des Händlers angezeigt. Der Händler besitzt dann ein kleines Menü in dem zwischen *Reden*, *Einkaufen*, und *Reisen* gewählt wer-

den kann. Beim *Reden* wird ein für den jeweiligen Händler festgelegter Dialog gezeigt, beim *Einkaufen* öffnet sich ein neuer Screen, in dem aus verschiedenen Verkaufsobjekten gewählt werden kann. Die Preise der Objekte werden mit den noch verfügbaren Schokolinsen verglichen. Ist das gewählte Objekt zu teuer wird ein negatives Feedback durch ein kurzes Geräusch zurückgegeben, ansonsten wird nochmal gefragt ob das Item tatsächlich gekauft werden soll. Welches Objekt gekauft wurde und wieviele Schokolinsen ausgegeben wurden, wird gespeichert. Handelt es sich bei dem gekauften Objekt um Concept Art, kann dieses im Hauptmenü unter Extras angesehen werden. Wurde ein Bonuslevel gekauft, erscheint nach Verlassen des Händlermenüs eine neue Teleportplattform über die der Spieler in das Bonuslevel gelangt. Wählt der Spieler die Option *Reisen*, gelangt er in einen Screen mit einer Kartenansicht, auf der die bereits besuchten Gebiete markiert wurden, die nun als Reiseziel ausgewählt werden können. Beim Reisen, wird das aktuelle Level des Protagonisten neu gesetzt und für dieses Level ein Reset ausgelöst.

screenshot zumindest zum einkaufen

Um den Spieler in das Spielgeschehen einzuführen, wurden ein Tutorial-Level sowie eine Art Tutor ins Spiel aufgenommen. Dieser Tutor sieht aus wie das Acagamicsmännchen und erklärt dem Spieler die Grundlagen des Spiels. Er ist der erste Charakter, mit dem der Spieler in Kontakt kommt und er erscheint immer dann, wenn der Spieler auf ein neues Spielelement trifft, das nicht bereits von einem anderen NPC erklärt wird. Er basiert vollständig auf dem Aktions-System.

Nicht in der ursprünglichen Planung enthalten sind rote und blaue Bonbonfeen. Sie sehen aus wie schwebende Bonbons. Steht der Spieler nah genug an den Feen, um mit ihnen interagieren zu können, ändert sich die Farbe ihrer Umrandung.

Die roten Feen können den Begleiter zum Hauptcharakter bringen, indem sie seine Position auf ihre eigene umsetzen. Sie wurden eingebaut, damit der Spieler den Begleiter nicht immer den weiten Weg zum Helden laufen lassen muss. Da bei der Händler-Teleportation zudem nur der Held transportiert wird, sind sie notwendig, damit auch der Begleiter an Orte zurückreisen kann, die er durch Laufen nicht mehr erreichen kann.

Die blauen Feen geben dem Spieler zusätzliche Hinweise zu den Rätseln.

Das Aktions-System

Das Aktions-Sytem macht es möglich Dialoge und Animationen umzusetzen, die durch Betreten von bestimmten Plattformen ausgelöst werden. Diese werden im Spiel genutzt, um dem Spieler die Steuerung und Spielobjekte zu erklären und um neue Spielelemente einzuführen. Sie werden auch genutzt um in kleinen Zwischensequenzen Teile der Geschichte zu erzählen. Bei allen Interaktionen mit der Lakritze führt diese Aktionen aus. Auch das Treffen des Begleiters und die Trennung von ihm sind Aktionen.

Die einzelnen Aktionen liegen als XML-Datei vor und werden zur Laufzeit direkt nach der Spielwelt eingelesen.

Eine Aktion kann einmalig sein, oder bei jedem Betreten der auslösenden Plattform immer wieder abgespielt werden.

Jeder Aktion wird ein auslösendes Spielobjekt zugeordnet, nicht zwingend muss es sich dabei um eine Plattform handeln. Weiterhin gehört zu jeder Aktion ein weiteres Spielobjekt vom Typ *ActionActor*. Das ist das Objekt, das die Aktion ausführen wird. Eine Aktion besteht aus mehreren einzelnen Handlungen. Diese sind entweder Erscheinen oder Verschwinden, eine lineare Bewegung an einen definierten Punkt im Level, deren Zielpunkt in Levelkoordinaten angegeben wird, oder das Anzeigen eines Dialoges, dessen Text ebenfalls in der Aktionsdefinition angegeben wird.

Für Aktionen gibt es aktuell keinen Editor, sie lassen sich aber leicht manuell in die XML-Datei schreiben. Die Datei ist nach Leveln strukturiert und bleibt damit, wenn die kleine Levelgröße berücksichtigt wird, gut zu überblicken.

User Interface

Shreenshots

Das Spiel besteht aus mehreren Screens, die Gamestates entsprechen, wie Titelbildschirm, Hauptmenü, Ladebildschirm oder das eigentliche Spiel. Ihre Wechsel und Übergänge werden vom Screen Manager verwaltet. Dieser kümmert sich auch darum, dass nur die gerade sichtbaren Screens gezeichnet und nur für den aktiven Screen Userinput ausgewertet wird. Für ihre Initialisierung, das Update und die Darstellung sind die Screens dann selbst zuständig.

Shader

Um den comicartigen Effekt des Spiels zu erhalten wurde ein Toon- bzw. Cel-Shader eingebaut.

Hierbei findet nicht mehr, wie normal, ein kontinuierlicher Wechsel zwischen den Farben statt, sondern es werden Schwellen für Farbintensitäten festgelegt. Damit wird jeder Farbton nur mit einer kleinen Farbpalette schattiert, wodurch die Spielobjekte passend zum gewünschten Stil schattiert werden. Diese Quantisierung beschränkt sich in dem umgesetzten Beleuchtungssystem allerdings auf den diffusen Lichtanteil des Materials des Objekts. Der ambiente und spekulare Anteil wird nach dem Phong-Modell berechnet und auf die quantisierte Farbe aufaddiert, wodurch die Glanzeffekte bei glatten Oberflächen deutlich schöner aussehen. Die Grundfarbe jedes Pixels wird aus der Textur entnommen, die dem jeweiligen Modell zugeordnet ist.

Außerdem werden die Vertices des Modells zusätzlich zum normalen Zeichnen ein kleines Stück nach „außen“ bewegt und in schwarz erneut gezeichnet. Durch Anwendung von

„Backface Culling“ wird verhindert, dass das zweite Modell das erste verdeckt. Dies sorgt letztendlich für die schwarzen Ränder der Modelle.

Die Schlagschatten in der Szene werden durch einfaches Shadowmapping erzeugt.

Für die Animationen der Modelle in XNA wird zudem das von Microsoft zur Verfügung gestellte SkinningSample und die dazu gehörende SkinningPipeline genutzt. Mit Hilfe dieser ist es in XNA möglich, die Animationen, die bereits im Modell gespeichert sind, abzuspielen und zu zeichnen.

Bei der Sonne und der Umrandung der Schokolinsen handelt es sich außerdem um Billboards, die mit ihrem eigenen Effekt gezeichnet werden.

Sowohl dieser Effekt als auch der Effekt, mit dem die normalen Objekte gezeichnet werden, implementieren einen optionalen Nebel-Effekt, der einen besseren Tiefeneindruck erzeugt. Dieser Nebel hat einen speziellen Modus für die Übersichtskarte, der dazu führt, dass die einzelnen Höhenebenen der Welt gut voneinander unterschieden werden können, was sich als große Orientierungshilfe erwiesen hat.

Levelerstellung

Die einzelnen Level werden im Editor *Tiled* zusammengestellt. Dafür gibt es ein vorbereitetes Basislevel, in dem alle verfügbaren Spielelemente mit ihren Eigenschaften bereits einmal enthalten sind. Diese können über Copy&Paste vervielfältigt und dann an die beabsichtigte Position geschoben werden. Die Elemente werden auf Ebenen angeordnet, die der y-Koordinate des jeweiligen Objekts entspricht. Dynamische Objekte werden manuell mit einer ID versehen, damit diese später für Schaltergruppen und das Aktivitäts-System verwendet werden können.

Die fertigen Level können dann im selbstgeschriebenen Extraprogramm *SceneEditor* eingeladen werden. In diesem Editor wird die Welt zusammengesetzt. Der Benutzer kann Gebiete erstellen, denen er eine ID und eine Position in Weltkoordinaten geben muss. Zudem muss er spezifizieren, welche Gebiete vor und nach dem aktuell bearbeiteten Gebiet liegen. Dann können zu den Gebieten Level hinzugefügt werden. Auch diese bekommen eine ID und eine Position, diese aber in Gebietskoordinaten. Zudem wird definiert, auf welchen Plattformen die spielbaren Charaktere nach einem Reset starten sollen. Es ist möglich, manuell Spielobjekte anzulegen. Es ist aber weiterhin möglich, eine .tmx-Datei zu importieren. Manuelle Bearbeitung der Level sollte möglichst vermieden werden.

Der *SceneEditor* schreibt eine .xml-Datei, die vom Spiel geladen werden kann. Sie durchläuft dabei drei Parser. Der *AreaParser* erstellt die Gebiets-Objekte, gibt ihnen ID, Position, Vorgänger und Nachfolger und übergibt dem Gebiet den Abschnitt der Datei, der die Level des jeweiligen Gebiets spezifiziert. Die Gebiete werden in einer Liste im *SceneManager* gespeichert. Jedes Gebiet ruft für seine Level dann den *LevelParser* auf, der die Level-Objekte erzeugt, mit ihren Parametern versieht und in einer Liste an das Gebiet zurückgibt. Die einzel-

nen Level bekommen das XML-Segment übergeben, das die Objekte beschreibt, die in ihnen enthalten sind. Diese werden dann vom *ObjectParser* erzeugt und in Listen einsortiert. Dabei wird unterschieden zwischen statischen Objekten, dynamischen Objekten und Schaltern, da alle drei Kategorien hinsichtlich Kollision und Update unterschiedlich behandelt werden.

Optionen

Im Optionsmenü ist es möglich, die Lautstärke und die Qualität, in der die Schlagschatten dargestellt werden, zu ändern. Bei Bedarf kann der Schattenwurf auch ausgeschaltet werden.

Weiterhin kann zwischen Vollbild- und Fenstermodus gewechselt werden und es ist möglich, auszuwählen, ob die Erklärungen des Tutorials angezeigt werden sollen.

In den Optionen lässt sich außerdem die Tastenbelegung einsehen.

Optimierungen

Im Laufe der Entwicklung wurden nicht viele umfangreichere Optimierungen vorgenommen, da der zeitliche Rahmen dies nicht hergab. In einigen Fällen wurde der Aufwand allerdings doch betrieben, eine bessere Lösung zu entwickeln.

Lange Zeit wurden alle Assets von jedem Objekt selbst geladen. Das bedeutete aber, dass das gleiche Asset vielfach geladen wurde. Es erhöhte die Ladezeit des Spiels massiv und beanspruchte auch mehr Arbeitsspeicher. Aus diesem Grund wurde der *AssetManager* implementiert. Er lädt nun alle Ressourcen zentral und verteilt dann nur noch Referenzen auf diese an die Objekte.

Auch das XML-Format, in dem die Szene gespeichert wird, und die zugehörigen Parser wurden überarbeitet. Einerseits verursachte das alte Format lange Ladezeiten, andererseits war es sehr anfällig für Fehler. Daher wurde von einem hauptsächlich auf Tags basierenden Ansatz umgestiegen auf eine Variante, die hauptsächlich Attribute verwendet. Damit verringerte sich auch die Dateigröße. Da die gesamte Welt in einer Datei gespeichert wird, ist auch dieser Vorteil von Bedeutung.

Technische Mindestanforderungen

Um das Spiel installieren und auszuführen zu können, müssen folgende Anforderungen erfüllt sein:

- Grafikkarte mit Unterstützung für Shader Version 3
- mindestens 2 GB Arbeitsspeicher
- mindestens 600 MB freier Speicherplatz
- XNA Framework Redistributable 4.0 muss installiert sein

optional:

- *MediBalancePro* ist vorhanden und funktionsfähig
- Gamepad ist vorhanden und funktionsfähig

Feedback / Kritiken

In diesem Abschnitt sind die Meinungen einiger Tester gesammelt. Dabei beziehen sich nicht alle Kommentare auf die aktuellste Version.

“Mir hat das Spiel sehr gut gefallen. Der Spieler sammelt in einem Candy-Wonderland alle Schokoladenlinsen auf, die er finden kann. Dazu springt er von Keks zu Keks, überwindet Hindernisse oder versperrte Wege, etwa indem er Zuckerwürfel verschiebt. Der Einstieg gelingt durch das anfängliche Tutorial sehr leicht, in dem sowohl die Steuerung als auch die Interaktion mit dem anderen Spieler erklärt wird. Die einzelnen Levels sind durch die vielen Rätsel abwechslungsreich gestaltet, wobei viele Lösungen erst in Zusammenarbeit mit der anderen Spielfigur gefunden werden.” - Simone Bexten

“Ich finde die Idee des Spiels, eine Süßigkeitenwelt zu schaffen, ist wirklich gut umgesetzt für ein Studentenprojekt. Die Texturen und Texte der Figuren passen sehr gut zum Thema. Es wird versucht, eine Geschichte zu erzählen, welche sich aber nicht in den Vordergrund drängt und sich eher darin versucht, durch den Spieler herausgefunden zu werden, indem man mit den Figuren spricht.

Das Toonshading rundet die Gesamtgrafik letztendlich ganz gut ab, da es knallige Farben bietet, so wie es bunte Süßigkeiten tun.

Zum Spielprinzip kann ich sagen, dass es eine gute Mischung aus Jump 'n' Run und Rätselspiel ist. Es fordert den Spieler, geschickt zu Springen, aber sich trotzdem die Spielwelt genau anzuschauen, damit man weiterkommt. Der "Einzelspieler-Coop" hilft dabei sehr, das einfache Spielprinzip etwas fordernder für den Spieler werden zu lassen, da er beide Spielfiguren gut händeln muss. Frustrierend kann es jedoch schon werden, wenn man zu oft runterfällt, aber ich glaube das bringt jedes Jump 'n' run mit. An dieser Stelle sind jedoch die Checkpoints recht großzügig gesetzt, sodass man nicht all zu viel nochmal spielen muss.

Alles in allem kann ich sagen, dass das Spiel mit seiner Geschichte ein gutes Kinderspiel ist. Die Rätsel sind für diese Spielgruppe genau richtig und die Knalligkeit der Farben unterstreicht dies. Für Erwachsene sind die Rätsel jedoch noch zu einfach.” - Sebastian Heerwald

“Overall I found the game pretty entertaining, at no point in my playthrough did I feel underchallenged or that the game parts were dragging on for too long without introducing something new. I would happily play much more levels.

The 'Chocolate chips' provided a fun challenge to collect that I thouroughly enjoyed.

The models, animations and level design could be improved as the floor textures seem too repetitive and the hand models seem a little out of place.

Also, as a suggestion I think an unlocked sprint button could be a good addition for a character, one that makes the main character jump a bit further or the pyramid character able to get somewhere fast enough after activating a timer, or even a new character that uses sprint in some fashion.” - Graeme Fitzapack

“The game is good and has potential, it's fun and relaxing. The black block texture should be changed, though. Also, I think the level design needs a bit more work and that the world should be bigger and have more complicated puzzles.” - Jin

“Das Spiel ist zum gespielten Zeitpunkt (Ende August 2013) weitgehend fertiggestellt. Dabei ist positiv zu vermerken, dass die Priorität auf der Spielbarkeit und dem Spielfluss lag. Trotz kleinerer Bugs ist das Spiel durchspielbar. An die, in Quadern arbeitende, Kollisionsabfrage gewöhnt man sich schnell. Die Spielfiguren können dabei am Rand eines Würfels quasi "in der Luft hängen". Dies wirkt sich jedoch nicht negativ auf den Spielspaß aus.

Das Sammeln von Linsen bietet einen Anreiz, die gesamte Welt in allen Ecken zu erkunden und die Möglichkeit, damit kleine Bilder im Spiel zu kaufen, ist eine schöne Idee, um damit auch etwas anfangen zu können. Hier könnte noch am Preis der Bilder oder der Anzahl der zu findenden Linsen etwas geändert werden, da die Bilder schon früh im Spiel alle kaufbar sind und die weiteren Linsen keine Verwendung mehr finden.

Der Schwierigkeitsgrad der Rätsel macht das Spiel auch für Anfänger geeignet, ohne dabei immer direkt eine offensichtliche Lösung zu präsentieren. Das halte ich für sehr gelungen.

Grafisch ist das Spiel schön gestaltet, mit Details wie kleinen Bewegungsanimationen der Spielfiguren.

Mangels weiterer Charaktere, die den Handlungsstrang hätten tragen können, ist die Idee der Mund-zu-Mund-Propaganda durch die Händler eine schöne Idee und gleichzeitig interessanter, als die Story aus dem Off zu erzählen.

Zum Spielzeitpunkt gab es noch kleinere Bugs bzw. einige nicht umgesetzte Punkte. Die Speicher- und Ladefunktion ist im Spiel noch nicht integriert, was bei der aktuellen Spiellänge von ca. 0,5-1 Stunde aber auch nicht zwingend erforderlich ist. Sollte man am Ende des Spiels in einer der beiden Verfolgungsjagden mit der Lakritze einen Reset benötigen, wird die Lakritze beim zweiten Durchgang nicht mehr angezeigt, was ein wenig auf Kosten der Atmosphäre geht.” - Sebastian Laubmeyer

Ausblick

Die Zeit, die für die Entwicklung des Spiels veranschlagt wurde, ist vorbei. Es gibt jedoch einige Features, die noch umgesetzt werden sollen.

Fehler

Die Kollision ist noch nicht optimal umgesetzt und benötigt Feinschliff. Objekte bleiben im aktuellen Zustand an Kanten hängen, obwohl das nicht erwünscht ist. Dieser Fehler macht das Spiel zwar nicht unspielbar, er schränkt das Leveldesign aber erheblich ein und sollte daher korrigiert werden.

Features

Die Steuerung mittels Gamepad und Balanceboard ist noch nicht vollständig integriert. Beide Eingabemöglichkeiten sind bereits getestet worden, es fehlen jedoch noch entsprechende Anpassungen am User Interface. Zusätzlich ist die Steuerung mit dem Balanceboard noch nicht angenehm, sondern braucht eine genauere Einstellung der Empfindlichkeit.

Für die Umsetzung der *MediBalance Pro*-Version des Spiels, wie sie in der Planung beschrieben wurde, blieb nicht genug Zeit.

Falls aber Interesse besteht, wird diese Version unter Berücksichtigung möglicher weiterer Wünsche und mit Fokus auf die Anwendung in Therapie-Sitzungen mit einer Länge von zehn bis 20 Minuten noch fertig gestellt.

Generell hat die aktuelle Version des Spiels noch nicht den Umfang, der ursprünglich angedacht war. Es ist geplant, das Spiel um mehrere Level zu erweitern und die Geschichte vollständig zu integrieren.

Zudem wurden von den Testern einige Vorschläge gemacht, wie das Spiel noch verbessert werden könnte. Die Vorschläge sollen noch darauf geprüft werden, ob sie sinnvoll und umsetzbar sind, und dementsprechend umgesetzt oder verworfen werden.

Projektfazit

Das Team hat in den vergangenen 4 Monaten viel Arbeit in das Projekt investiert. An dieser Stelle soll kurz darauf eingegangen werden, was wir in diesem Zeitraum gelernt haben, womit wir zufrieden sind und was wir anders machen würden, wenn wir das Projekt noch einmal von vorne beginnen könnten.

Positive Aspekte

Die Kommunikation zwischen den Teammitgliedern funktionierte gut und es konnten regelmäßig Treffen vereinbart werden. In kritischen Phasen hat das gesamte Team viel Zeit für das Projekt aufgebracht und über viele Stunden konzentriert an dem Spiel gearbeitet.

Mit den Entscheidungen, die wir bezüglich des Settings und der Spielmechaniken getroffen haben, sind wir sehr zufrieden. Sie ermöglichten uns, trotz mangelnder Erfahrung ein Spiel zu entwickeln, das bei den Testern durchgehend gut ankam. Viele haben sich mehr Level und teilweise sogar weitere Mechaniken für das Spiel gewünscht. Auch wir glauben, dass das Spiel so viel Spaß macht, dass wir eine weitere Version mit mehr Leveln und einer vollständigen Geschichte entwickeln wollen.

Alle Teammitglieder haben im Verlaufe des Projekts viel über Projektplanung und -durchführung auf der einen Seite und über Programmierung im Allgemeinen und speziell mit C# und dem XNA-Framework auf der anderen Seite gelernt.

Wir hatten viel Spaß bei der Entwicklung, auch wenn wir teilweise auf Probleme gestoßen sind, mit denen wir nicht gerechnet hatten.

Negative Aspekte

Da keine Erfahrungen bezüglich Modellerstellung (vor allem im Hinblick auf Animation) in Blender und der zugehörigen Implementierung in XNA vorhanden waren, kam es dort zu einigen Schwierigkeiten. So waren Modelle zu Anfang teils nicht sichtbar oder stark verzerrt. Besonders bei den Animationen gab es des Öfteren Probleme, die aber meistens gelöst und andernfalls umgangen werden konnten.

Es war schwierig die Physik so umzusetzen, dass sie sich erwartungsgemäß verhielt. Sie stellte uns vor viele Probleme, mit denen wir ursprünglich nicht gerechnet haben. Vor allem die Komplexität wurde unterschätzt. Die Physik benötigt im Moment viel Rechenkraft.

Man hätte statt einer eigenen Implementation eine bestehende Physikbibliothek nutzen können. Diese hätte allerdings von Anfang an eingeplant werden müssen. Wir haben zwar zwischenzeitlich einen Umstieg in Erwägung gezogen, dieser hätte aber sehr tiefgreifende Änderungen erfordert, für die wir leider keine Zeit aufbringen konnten.

Zudem hatten wir Probleme mit dem Technical Design Document. Da wir keine nennenswerte Erfahrung mit der Programmierung eines Spiels, das über einen kleinen Prototypen hinausgeht, hatten, konnten wir die technischen Anforderungen schlecht abschätzen. Das führte auch dazu, dass uns das TDD während der Entwicklung kaum helfen konnte, da es nicht konkret genug war.

Bewertung des Projekts durch das Team

Wir wollten im Rahmen dieses Projekts ein Spiel entwickeln, das Spielern verschiedener Zielgruppen Freude bereitet. Dafür haben wir es von verschiedensten Menschen testen lassen und abgesehen von Problemen mit der Steuerung mit Maus und Tastatur, die sich mit der Möglichkeit der Controllersteuerung erübrigen sollten, konnten alle Tester problemlos in das Spiel einsteigen. Sie spielten das Spiel ausgiebig und einige von ihnen auch mehrfach. Alle Tester schienen Spaß am Spiel zu haben und äußerten sich auch entsprechend.

Damit haben wir unser Hauptziel erreicht.

Wir haben es nicht geschafft, alles umzusetzen, was wir uns vorgenommen hatten. Die Planung hätte also besser sein können. Wir haben aber ein voll funktionsfähiges Spiel auf die Beine stellen können, das auch frei von schwerwiegenden Bugs zu sein scheint. Die Pläne, die wir nicht umsetzen konnten, bezogen sich fast alle auf Zusatzfeatures oder Inhalte, nicht aber auf die Spielmechaniken.

Das Projekt hat uns alle vor Herausforderungen gestellt, die wir oft gemeinsam bewältigen konnten. Wir haben viel gelernt und hatten Spaß dabei, am Projekt zu arbeiten.

Insgesamt sind wir sehr zufrieden mit dem Ergebnis dieser vier Monate.

Anhang