UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Entwicklung eines interaktiven Spiels mit dem Wii™-Balance Board™

Studienarbeit

im Studiengang Computervisualistik

vorgelegt von
Mareike Stier

Betreuer: Dipl.-Inform. Stefan Rilling
(Institut für Computervisualistik, AG Computergraphik)

Koblenz, im August 2009

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

.....
(Ort, Datum) (Unterschrift)

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Die Entwicklung im Bereich der Videospiele generierte in den letzten Monaten durch innovative Konzepte und neue Steuerungsmöglichkeiten ein hohes Maß an Aufmerksamkeit. Einen Meilenstein setzte die Firma Nintendo® mit dem sogenannten Wii™ Balance Board™ . Dies ist ein Eingabegerät in Form eines Brettes, auf das sich der Spieler stellen muss, um ein Spiel mittels seiner Körperbalance steuern zu können. Mit dieser Form der Steuerung konnten neue Spielkonzepte erstellt und umgesetzt werden. Dadurch wurden erstmals Personengruppen angesprochen, die zuvor wenig bis gar kein Interesse an Videospielen hatten.
Die Computerspielebranche hingegen verfolgt weiter das Ziel eine möglichst reale Spielumgebung zu schaffen und hält an ihren gewöhnlichen Steuerungen mittels Tastatur, Maus und Joystick fest.

Im Rahmen dieser Studienarbeit wurde ein 3D-Computerspiel entwickelt, welches das Konzept der Videospiele verfolgt und die Möglichkeit bietet, mittels eigener Körperbalance zu steuern.

1.2 Zielsetzung

Ziel dieser Studienarbeit war es, innerhalb von sechs Monaten ein lauffähiges 3D-Computerspiel zu entwickeln, welches als Eingaberät das Nintendo® Wii ™ Balance Board™ verwendet. Durch diese praxisnahe Aufgabe, sollte der Entwickler einen Einblick in die 3D-Spieleentwicklung erhalten und zusätzlich eine objektorientierte Programmiersprache erlernen, sowie anwenden. Ein weiterer Schwerpunkt lag in der Entwicklung eines eigenen Spielkonzepts. Es musste darauf geachtet werden, dass das Spiel vorrangig mit der eigenen Körperbalance gesteuert wird.

1.3 Verwendete Hard- und Software

Der erste Schritt dieser Studienarbeit beinhaltete die Suche nach geeigneter Hard- und Software.

Das Nintendo® Wii™ Balance Board™ versendet die gemessenen Daten über Bluetooth® . Damit der Computer diese abfangen und benutzen kann, muss ein funktionsfähiges Bluetooth® -Gerät installiert sein. Es hat sich gezeigt, dass Bluetooth® -Treiber von Broadcom® zusammen mit *Bluecove* eine sehr stabile Verbindung zulassen und für dieses Projekt hervorragend geeignet sind. Bluetooth® -Treiber von Microsoft® werden nicht unterstützt. Um die empfangenen Daten weiter verarbeiten zu können, müssen sie mittels

Bibliotheken übersetzt werden. In diesem Fall wurde die Bibliothek *WiiRemoteJ* verwendet. Dies ist eine frei verfügbare Java-API und Bibliothek, die mit einem Nintendo® Wii™ Remote™ mittels Bluetooth® interagiert. Vorteil dieser Bibliothek ist die detaillierte Javadoc, die Entwicklern einen schnellen Überblick über Funktionen und Methoden liefert.

Im nächsten Schritt musste eine 3D-Game Engine gefunden werden. Anhand der 94.900.000 Suchergebnisse bei Google™ nach "3D-Game Engine" (Stand: Juli 2009) erkennt man die Komplexität dieses Schrittes. Um die Auswahl zu erleichtern, wurden folgende Kriterien angewandt:

Die 3D-Game Engine

- muss mit Java benutzt werden können,
- sollte frei verfügbar sein,
- sollte leicht erweiterbar sein, um eine Wii™Balance Board™ Integration zu gewährleisten und
- sollte gut dokumentiert sein.

Auf der Grundlage dieser Kriterien wurde die 3D-Game Engine *jMonkeyEngine* ausgewählt.

Bei *jMonkeyEngine* oder kurz *jME* handelt es sich um eine Szenegraph basierte 3D-Grafik API, welche komplett in Java programmiert ist und mit Hilfe der Lightweight Java Game Library (LJGL) eine Kommunikation zur Grafikkarte herstellt. *JMonkeyEngine* steht unter der BSD-Lizenz und entwickelt sich seit dem Jahre 2003 ständig weiter. Sie kann mit verschiedenen Modulen erweitert werden.

Als Programmierumgebung diente das Open-Source-Programm *Eclipse*. Die problemlose Installation und Integration der genannten Hard- und Software erlaubte einen raschen Einstieg in den Praxisteil.

2 Nintendo® Wii™ Balance Board™

2.1 Aufbau und Funktionalität

Das Wii™ Balance Board™ ist ein kabelloses Peripheriegerät für die Nintendo® Videospielekonsole Wii™ . Es besitzt einen eigenen Powerknopf und wird mit vier AA- Batterien betrieben, die auf der Rückseite (Abbildung 1) eingelegt werden müssen. Vor der ersten Verwendung muss das Wii™ Balance Board™ mit der Spielekonsole initialisiert werden. Dazu ist es notwendig, dass der Benutzer den Batteriedeckel auf der Rückseite öffnet und den

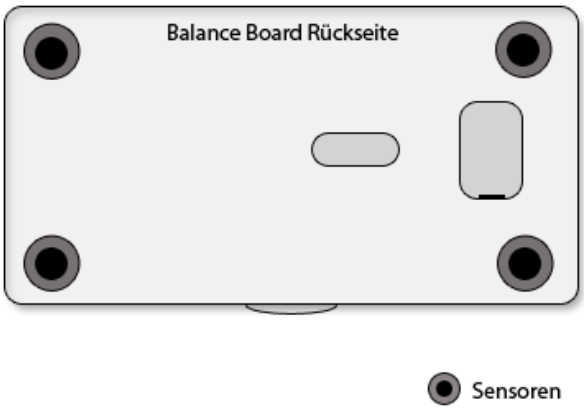

Abbildung 1: Rückseite eines Nintendo® Wii™ Balance Boards™

SYNCHRO-Knopf (Abbildung 2) gedrückt hält bis die LED-Kontrolllampe, die sich seitlich am Board befindet, dauerhaft blinkt. Im nächsten Schritt muss der SYNCHRO-Knopf an der Spielekonsole gedrückt werden. Erst nach erfolgreichem Initialisieren lässt sich das Wii™ Balance Board™ als Eingabegerät für den Computer verwenden. Das Wii™ Balance Board™ hat eine Abmessung von 51,1 x 31,6 x 5,26 cm und besitzt vier Sensoren, die auf der Unterseite angeordnet sind. Mit ihnen kann das Gewicht und auch die Gewichtsverlagerung gemessen werden. Diese Daten sind notwendig um

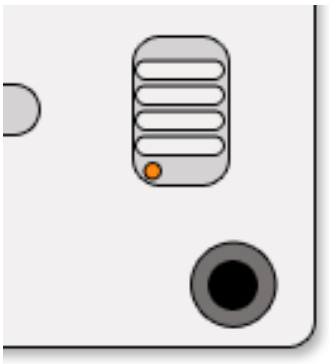

Abbildung 2: SYNCHRO-Knopf eines Nintendo® Wii™ Balance Boards™

beispielsweise eine Figur in einem Spiel steuern zu können. Für eine korrekte Messung wird empfohlen, das Wii™ Balance Board™ einen Meter vom Empfangsgerät entfernt, auf einer glatten Oberfläche, aufzustellen. Das Maximalgewicht liegt bei einhundertfünfzig Kilogramm. An der Seite des Wii™ Balance Boards™ befindet sich der Powerknopf, der blau leuchtet, sobald eine Verbindung mit einem Endgerät besteht. Die Oberfläche des

Wii™ Balance Boards™ist sehr rau, um Rutschgefahr zu mindern. In der Betriebsanleitung wird betont, dass das Wii™ Balance Board™ barfuß zu bedienen sei, um dieses nicht zu beschädigen.

2.2 Übertragung der Daten

Das Nintento® Wii™ Balance Board™ kann über eine Bluetooth® - Schnittstelle mit einem Computer verbunden werden. Dazu ist es notwendig, dass der Computer über ein Bluetooth® -Gerät verfügt. Die gemessenen Daten

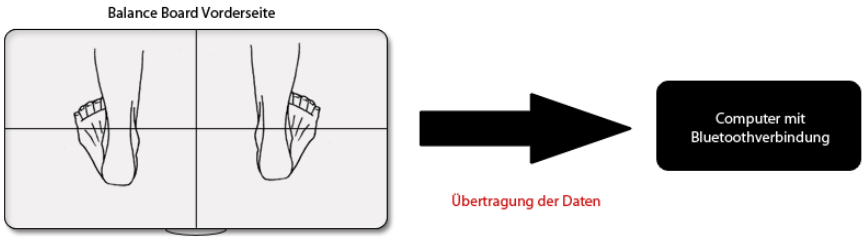

Abbildung 3: Übertragung der Daten zwischen Balance Board™ und PC

werden unkodiert vom Wii™ Balance Board™ als 24 Byte-Daten versendet. Um das Balance Board™ für dieses Computerspiel zu benutzen, muss es zunächst in der Bluetooth® -Umgebung gesucht werden. Bei einer erfolgreichen Suche wird es mit dem Namen Nintendo RVL-WBC-01 gefunden und besitzt eine MAC-Adresse. Für dieses Spiel sollte das Balance Board™ jedoch nicht manuell mit dem Computer verbunden werden. Es reicht völlig aus, wenn die Bluetooth® -Umgebung das Balance Board™ gefunden hat. Eine Verbindung wird innerhalb des Spieles automatisch hergestellt.

Das Balance Board™ besitzt vier Sensoren, die auf der Rückseite befestigt sind. Mit ihnen wird das Körpergewicht gemessen und in 24 Byte-Daten gespeichert. Die Messung des Gewichtes wird auf vier Bereiche geteilt. Dazu gehören TOP RIGHT, TOP LEFT, BOTTOM RIGHT und BOTTOM LEFT. Mittels Interpolation kann aus den Messungen der Felder ein Gesamtgewicht ermittelt werden.

2.3 Bibliothek

Um die empfangenen Daten für ein Computerspiel zu verwenden, müssen sie übersetzt und angepasst werden. Die Übersetzung der Daten in normale float- oder double-Werte übernimmt in dieser Studienarbeit die Bibliothek *WiiRemoteJ*. Diese frei verfügbare und sehr detaillierte Bibliothek wurde in Java programmiert und enthält viele Funktionen und Methoden, um mit einem Balance Board™ zu interagieren. Durch einfaches Hinzufügen der entsprechenden *WiiRemoteJ.jar*-Datei lassen sich alle Funktionen, Methoden und Konstanten dieser Bibliothek in eigenen Projekten verwenden.

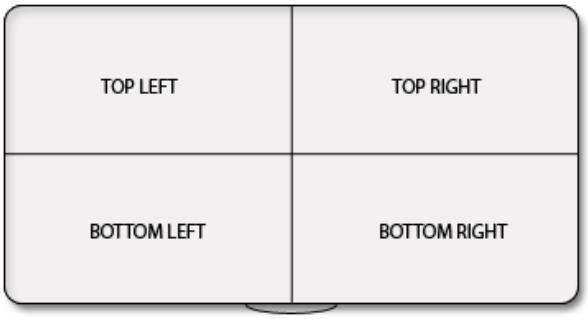

Abbildung 4: Einteilung der Sensorenfelder eines Balance Boards™

Um diese Bibliothek etwas genauer zu erklären, wird im Folgenden ein kleines Testprogramm vorgestellt. Dieses Testprogramm verbindet das Nintendo® Wii™ Balance Board™ mit einem Computer und gibt das Gewicht des Benutzers aus. Anschließend wird ein kurzer Überblick über wichtige Funktionen dieser Bibliothek gegeben, die in dieser Studienarbeit zum Einsatz kamen.

Um die Funktionen und Methoden von *WiiRemoteJ* verwenden zu können ist es notwendig, die Pakete dieser Bibliothek in das Projekt zu integrieren:

```
1 import wiiremotej.*;
2 import wiiremotej.event.*;
```

Die Klasse TestCode ist abgeleitet von der Klasse BalanceBoardAdapter und verwendet zusätzliche Funktionen der Klasse BalanceBoardListener. Es wird eine Instanz *remote* erstellt, die das Balance Board™ darstellt.

```
1 public class TestCode extends BalanceBoardAdapter
2 implements BalanceBoardListener{
3
4         private static BalanceBoard remote = null;
5
6         }
```

Um eine Verbindung mit einem Balance Board™ herstellen zu können, benötigt das Programm dessen MAC-Adresse. Bei einer erfolgreichen Verbindung erscheint der String *connected* auf der Konsole. Bei einer fehlerhaften Verbindung wird eine entsprechende Fehlermeldung auf der Konsole ausgegeben.

Fehler entstehen, wenn die zuvor eingestellte MAC-Adresse des Boards nicht korrekt ist oder der Benutzer den SYNCHRO-Knopf während des Verbindungsversuchs nicht gedrückt hat.

```
1  public TestCode()
2  {
3          remote = null;
4
5          try
6          {
7          remote = WiiRemoteJ.connectToBalanceBoard("00000000");
8          remote.addBalanceBoardListener(this);
9          }
10         catch(Exception e)
11          {
12         e.printStackTrace();
13          }
14
15         connected = true;
16         System.out.println("connected");
17         }
```

Während einer erfolgreichen Verbindung leuchtet die blaue LED-Kontrolllampe dauerhaft an der Seite des Balance Boards™ auf. Drückt der Benutzer diesen Powerknopf, wird die Verbindung zwischen Computer und Balance Board™ beendet. Es erscheint der String *disconnected* auf der Konsole und die LED-Kontrolllampe leuchtet nicht mehr.

```
1  public void buttonInputReceived(BBButtonEvent evt)
2  {
3          if(evt.isPressed()) {
4                  remote.disconnect();
5                  System.out.println("disconnected");
6          }
7  }
```

Um nun das Gewicht des Benutzers auf der Konsole auszugeben wird eine Funktion von *WiiRemoteJ* verwendet. Das Gewicht wird in einer double Variable abgespeichert.

```
1  public void massInputReceived(BBMassEvent evt)
2  {
3          double totalmass = evt.getTotalMass();
```

```
4             System.out.println(totalmass);
5 }
```

Das Balance Board™ besitzt vier Sensoren für die Berechnung der Bewegungsimpulse. Mit Hilfe von *WiiRemoteJ* kann man die Werte dieser vier Sensorenfelder problemlos berechnen und in die eigene Funktion integrieren. Somit können individuelle Steuerungsmöglichkeiten programmiert werden.

```
1 double massRightTop =
2 evt.getMass(MassConstants.TOP, MassConstants.RIGHT);
3
4 double massLeftTop =
5 evt.getMass(MassConstants.TOP, MassConstants.LEFT);
6
7 double massRightBottom =
8  evt.getMass(MassConstants.BOTTOM, MassConstants.RIGHT);
9
10 double massLeftBottom =
11 evt.getMass(MassConstants.BOTTOM, MassConstants.LEFT);
```

Die oben genannten Funktionen sind die Wichtigsten, die in diesem Spiel zum Einsatz kamen. In der detaillierten *Javadoc* findet der Entwickler weitere Informationen zu Funktionen, Methoden und Konstanten dieser Bibliothek. *WiiRemoteJ* kann neben dem Wii™ Balance Board™ noch weitere Nintento® Wii™ Remotes™ integrieren. Eine komplette Liste befindet sich auf der Internetseite von *WiiRemoteJ*.

3 Spielkonzept

3.1 Spielidee

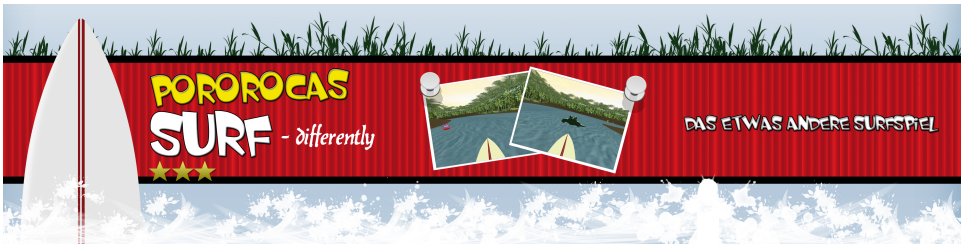

Abbildung 5: "Pororocas - Surf differently" Banner

"Pororocas - Surf differently" heißt das in dieser Studienarbeit entwickelte 3D-Computersspiel, bei dem der Spieler sein Surfboard über den Amazo-

nas lenken muss. Pororocas sind riesige Gezeitenwellen, die jedes Jahr im Februar und März entstehen. Dabei werden bei Voll- und Neumond täglich zweimal riesige Wassermengen vom Atlantischen Ozean in die Flussmündung des Amazonas gedrückt und große Wellen verursacht, die von Surfern gerne zur Ausübung ihres Sports genutzt werden. Bei "Pororocas - Surf differently" wurde dieses reale Ereignis als Spielidee verwendet. Der Surfer befindet sich im tropischen Regenwald und muss auf seinem Surfboard den Amazonas bezwingen und möglichst unversehrt und schnell das Ziel erreichen. Im Wasser lauern jedoch Gefahren, die auf Surfer warten und ihnen wichtige Lebenspunkte entziehen können. Sollte der Surfer keine Lebenspunkte mehr haben, hat er das Spiel verloren und scheidet aus. Gewonnen hat der Surfer, der als Schnellster das Ziel erreicht hat.

3.1.1 Allgemeiner Ablauf

Nachdem das Spiel gestartet wird, erscheint ein Menü mit mehreren Auswahlmöglichkeiten. Der Spieler kann sich vorab über die Spielregeln und Hintergrundinformationen informieren oder das Spiel direkt durch Auswahl einer Steuerungsmethode beginnen. Hierbei kann er zwischen dem Nintento® Wii™ Balance Board™ oder der Computertastatur auswählen. Sobald sich der Spieler für eine Methode entschieden hat, kann er einen Spielernamen angeben und mit der Levelauswahl fortfahren.

Es werden zwei Spielmodi angeboten: ein Anfängermodus und ein Fortgeschrittenenmodus. Beim Anfängermodus ist die Geschwindigkeit des Surfers, sowie die der Angreifer reduziert. Beim Fortgeschrittenenmodus ist die Surfgeschwindigkeit höher und die Angreifer agieren aggressiver.

Nach Auswahl des Levels wird das Spiel gestartet und der Spieler muss das Surfboard über den Amazonas lenken und versuchen, möglichst schnell das Ziel zu erreichen.

Es besteht die Möglichkeit seine Geschwindigkeit zu erhöhen, indem man einen Geschwindigkeitsbeschleuniger berührt; allerdings gibt es auch Geschwindigkeitsbremser, die die Geschwindigkeit des Surfers massiv reduzieren. Der Surfer hat zu Beginn des Spiels 300 Lebenspunkte, die entsprechend der Berührungszeit mit einem Gegner reduziert werden. Neue Lebenspunkte können durch Berührung einer Lebensbox erlangt werden. Sollte der Spieler keine Lebenspunkte mehr haben, ist das Spiel für ihn verloren und er scheidet aus.

Gewonnen hat der Spieler, der als Erster das Ziel erreicht. Die fünf besten Siegerzeiten, inklusive Spielername, werden in der Highscoreliste gespeichert.

3.1.2 Gegner

Es gibt in diesem Spiel verschiedene Arten von Gegnern, die alle das gleiche Ziel verfolgen: sie wollen den Surfer nicht unversehrt auf dem Fluss surfen lassen. Die Krokodile, Piranhas und Fische lauern anfangs ruhig im Wasser. Erst wenn der Surfer in ihre Nähe kommt greifen sie an und verfolgen ihn solange, bis der Surfer außer Reichweite ist. Falls der Angriff gelingt, werden dem Surfer Lebenspunkte abgezogen. Dieser muss dann versuchen so schnell wie möglich die Flucht zu ergreifen. Sollte ihm dies nicht gelingen und er alle Lebenspunkte verlieren, ist das Spiel verloren. Eine weitere Gefahr stellen die Wasserpflanzen dar. Auch diese entziehen dem Surfer bei Berührung Lebenspunkte.

3.1.3 Weitere Spielgegenstände

Neben den Gegnern gibt es weitere Spielgegenstände, die für den Surfer eine große Bedeutung darstellen. Der mit Abstand wichtigste Spielgegenstand ist die grüne Lebensbox. Diese lädt verloren gegangene Lebenspunkte wieder auf. Dabei ist die Berührungszeit entscheidend, denn die Höhe der Aufladung wird anhand dieser gemessen.

Ein weiterer wichtiger Spielgegenstand ist der Geschwindigkeitsbeschleuniger, eine gelbe Box, die die Geschwindigkeit des Surfers bei Berührung für drei Sekunden verdoppelt.

Auch gibt es einen negativen Spielgegenstand, den Geschwindigkeitsbremser, der die Surfgeschwindigkeit des Spielers massiv reduziert. Solange der Surfer sich auf diesem roten Feld bewegt, ist seine Surfgeschwindigkeit heruntergesetzt. Erst nach Verlassen des Feldes erhält er seine normale Surfgeschwindigkeit zurück.

3.1.4 Steuerung

Das Surfboard kann mit zwei unterschiedlichen Eingabegeräten bewegt werden, die der Spieler im Menü festlegt.

Die erste Möglichkeit ist das Steuern des Surfboards mit der Computertastatur. Dazu sind drei Tasten notwendig. Zum Vorwärts fahren wird die Taste W verwendet. Gelenkt wird nach Rechts mit der Taste D und nach Links mit der Taste A.

Die zweite Möglichkeit ist das Steuern des Surfboards mit der eigenen Körperbalance. Dazu wird ein Nintento® Wii™ Balance Board™ eingesetzt, welches die Gewichtsverlagerung des Spielers misst und die entsprechenden Daten über Bluetooth® an den Rechner übergibt.

4 Implementierung

4.1 Aufbau

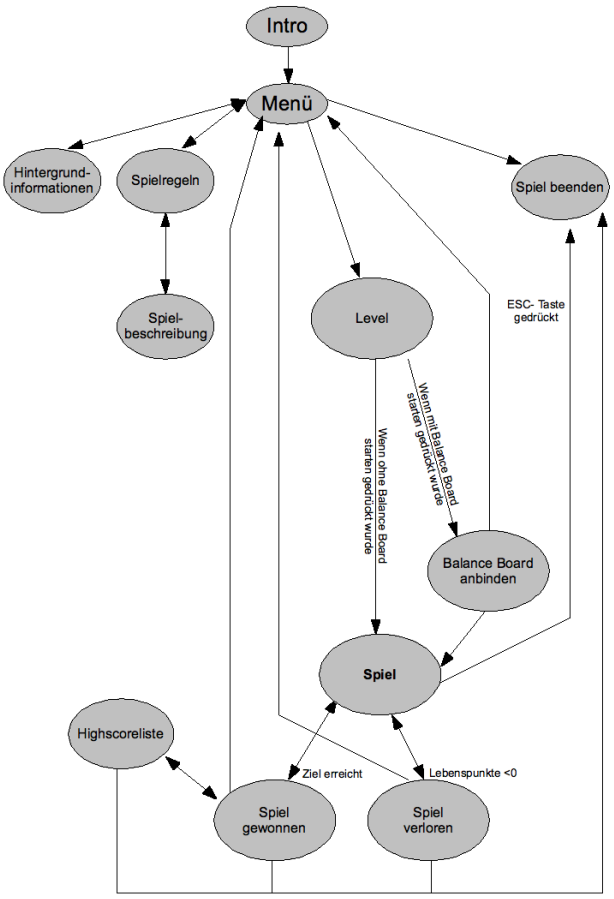

Abbildung 6: Aufbau dieser Studienarbeit

Es gibt verschiedene Zustände, in denen sich der Spieler befinden kann. Diese werden in Abbildung 6 durch graue Ellipsen dargestellt. Die Pfeile deuten auf die Bewegungsrichtung zwischen den Zuständen hin.
Die wichtigsten Zustände sind das Menü, das Spiel und das Beenden des Programms. Vom Menü aus kann der Spieler verschiedene Zustände wählen, zum Beispiel eine Auswahl treffen mit welchem Steuerungsgerät er das Spiel spielen möchte. Durch Drücken der ESC-Taste kann der Spieler während des Spiels problemlos ins Menü gelangen und das Spiel beenden.

Die folgende Tabelle bietet eine Auflistung aller Klassen mit dazugehörigen Eigenschaften:

Paket	Klassenname	Eigenschaften
core	Game	Spielklasse, die Eigenschaften wie Pause, Beenden und Größe des Bildschirms festlegt.
	SDExceptionHandler	Gibt schwerwiegende Fehlermeldungen aus und beendet danach das Spiel.
	Start	Erstellt eine Spielinstanz und startet sie. Hier werden GameStates erstellt und Pfade für externe Dateien wie Grafiken und Modelle festgelegt.
effects	ParticleEffectFactory	Erstellt Kollisionseffekte für dieses Spiel.
gamestates	AboutState	Ausgabe der Hintergrundinformationen.
	ConnectToBalanceBoardInfoState	Ausgabe der Informationen um ein Balance Board™ mit dem Computer zu verbinden.
	GameOverState	Verliererzustand
	HighScoreState	Ausgabe der Highscoreliste.
	HUDGameState	Grafische Ausgabe des Timers, welche auf dem Bildschirm innerhalb des Spiels erscheint.
	InGameState	Spiel mit allen Spielfunktionen.
	LevelGameState	Auswahlmöglichkeit der Level.
	PassManagerGameState	Regelt das Zusammenspiel zwischen den einzelnen GameStates.

	RulesState	Ausgabe der Spielregeln.
	StatisticGameState	Gibt die Lebenspunkte auf dem Bildschirm aus.
	StoryState	Ausgabe der Spielstory.
	SwingMenuState	Menüklasse
	WinnerState	Gewinnerzustand
gamestates.controller	InGameListener	Regelt das Verhalten, falls eine Taste der Computertastatur während dem Spiel gedrückt wird.
scene	BB_Support	Speichert/Enthält den bool-Wert, ob das Spiel mit einem Balance Board™ gespielt wird oder nicht.
	Billboards	Erstellt die Billboards.
	DaySkyBox	Erstellt die Skybox.
	Enemy	Enthält Eigenschaften der Gegner.
	HealthQuad	Erstellt die Lebensboxen.
	Level	Speichert den ausgewählten Spielmodus.
	LightManager	Enthält Informationen zu der Beleuchtung der Szene.
	Player	Enthält Eigenschaften des Spielers.
	PlayerHandler	Enthält Eigenschaften zu der Bewegung des Spielers.
	PointSystemHUD	Grafiken, die den Status der Lebenspunkte auf dem Bildschirm anzeigen.
	SavePlayerName	Speichert/Enthält den Spielernamen.

	SlowDownQuad	Erstellt die Geschwindigkeitsbremser.
	SpeedSignQuad	Erstellt die Geschwindigkeitsbeschleuniger.
	TerrainManager	Erstellt ein Terrain mittels einer Heightmap.
	Water	Erstellt den Amazonas.
scene.action	ForwardAndBackwardAction	Regelt die Surfbewegung (Vorwärts, Rückwärts)
	PlayerRotateAction	Regelt die Drehbewegung des Surfboards.
util	HighScoreList	Erstellt eine Highscoreliste.
	ImagePanel	Zusatzklasse für Swing, um Grafiken einzubinden.
	SaveMAC	Speichert/Enthält die eingegebene MAC-Adresse des Balance Boards™.
	WiiRemoteProxy	Verbindet ein Balance Board™ mit einem Computer und gibt die gemessenen Werte an das Spiel. Dies ist die Hauptklasse des Balance Boards™.

In den folgenden Abschnitten soll auf die Implementierung des Spiels eingegangen werden. Es soll nicht der gesamte Programmcode dargestellt werden, sondern nur seine wichtigsten Bausteine.

4.2 Menü

In das Menü gelangt der Spieler kurz nach starten des Programms automatisch. Um in einen anderen Zustand zu gelangen, muss er mit der linken Maustaste auf den jeweiligen Button klicken. Es wird dem Spieler die Möglichkeit gegeben seinen Spielernamen in das dafür vorgesehene Textfeld einzugeben. Möchte er das Spiel starten, muss eine Auswahl über die Steuerungsart, nämlich durch Klicken des jeweiligen Bottons, getroffen werden.

Der eingetragene Name und die Auswahl der Steuerungsart wird in einer eigenen Variable gespeichert. Das komplette Menü basiert auf *Swing* und verwendet die jMonkeyEngine Klasse JMEDesktopState.

Abbildung 7: Menüansicht

4.3 Spiel

Im Folgenden wird auf die Umsetzung einzelner Spielelemente eingegangen.

4.3.1 Landschaft

Als Vorlage für die Landschaft diente die reale Umgebung des Amazonasgebiets. Wichtig hierbei war es, dem Spieler den Eindruck zu vermitteln, dass er sich auf dem Amazonas im tropischen Regenwald befindet.

4.3.1.1 Skybox

Eine Skybox ist ein Würfel mit sechs Seiten, auf denen jeweils eine 2D-Textur gelegt wird, um ein Panorama der Umgebung zu erstellen. Die Oberseite des Würfels wird für den Himmel verwendet; die Unterseite für den Boden. Die restlichen vier Seiten repräsentieren die jeweiligen Himmelsrichtungen. Die Kamera befindet sich zu jeder Zeit genau in der Mitte des Würfels. Wenn sie

sich bewegt, wird auch die Skybox mit bewegt. In diesem Spiel simuliert die Kamera das Auge des Spielers. Somit wird immer der Eindruck vermittelt, dass die Skybox um den Spieler herum liegt, egal in welche Richtung er blickt.

Die Texturen für die Skybox wurden in einem 2D-Pixelgrafikprogramm erstellt und anschließend mit der jME eigenen ResourceLoader-Klasse in das Spiel geladen und an die richtige Stelle des Würfels gesetzt.

Damit die Skybox richtig gezeichnet wird, musste der RenderState des Spiels aktualisiert werden. Anschließend wurde die Skybox an den rootNode des Spiels angehängt.

4.3.1.2 Terrain

Die Landschaftsvisualisierung hat einen entscheidenden Einfluss auf die Spielwahrnehmung. In diesem Spiel bewegt sich der Surfer ausschließlich auf dem Amazonas. Dementsprechend musste ein Terrain mit einem Flussverlauf geschaffen werden.

Für das Modellieren des Terrains wurde eine sogenannt Heightmap erstellt. Dies ist ein 2D-Schwarz-Weiß Bild, welches auf ein Terrain angewendet wird. Wie der Name schon vermuten lässt, enthält das Bild Höhenwerte. Schwarz repräsentiert den tiefsten und weiß den höchsten Punkt in der Szene. In diesem Spiel kam die Heightmap aus Abbildung 8 zum Einsatz.

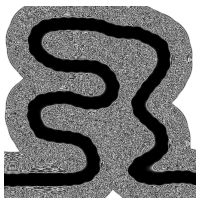

Abbildung 8: Heightmap

Diese wurde auf die erstellte Terrain Page angewendet und anschließend durch die 2D-Textur aus Abbildung 9 verschönert.

Abbildung 9: 2D-Textur der Terrain Page

Damit das erstelle Terrain auch im Spiel sichtbar ist, musste es an den root-Node des Spiels angehängt werden.

4.3.1.3 Billboard

Um den Eindruck des tropischen Regenwaldes zu erwecken, sind Pflanzen notwendig. Je mehr Pflanzenmodelle in ein Spiel integriert werden, desto realitätsnäher wirkt es. Allerdings sind Pflanzenmodelle sehr rechenintensiv, da sie aus vielen 3D-Meshes bestehen. Eine Alternative bieten sogenannte Billboards. Dies sind dünne Quadrate, ähnlich einer Platte, auf die 2D-Texturen gelegt werden.

Die Idee der Billboards wurde in diesem Spiel für die Visualisierung des tropischen Regenwaldes verwendet. Allerdings fügte man an der Oberseite der Billboards noch ein 3D-Modell einer Palme hinzu. Um die komplette Szene

Abbildung 10: Billboard mit 3D-Modell einer Palme

zu füllen, waren 68 Billboards notwendig, die manuell positioniert wurden. Aus Performancegründen wurde nur ein Billboard erstellt, das anschließend 67 mal kopiert wurde. Um Kopien von Modellen zu erstellen bietet jME eine eigene Klasse namens SharedMesh an.

Nach der Erstellung der Billboards musste darauf geachtet werden, dass diese richtig beleuchtet und an den rootNode des Spiels übergeben werden.

4.3.1.4 Wasser

Eines der wichtigsten optischen Elemente dieses Spiels ist das Wasser, auf dem sich der Spieler bewegt.

Um das Wasser darzustellen wurden zwei Quads erstellt, auf die jeweils eine 2D-Textur gelegt wurde. Anschließend wurden sie skaliert um realistischer zu wirken. Zur Simulierung einer Bewegung musste eine eigene update-Methode

innerhalb der Wasserklasse implementiert werden, die in gewissen Zeitabständen die Position der Quads verändert.

Zum Schluss wurde auf die richtige Beleuchtung geachtet und das Wasser ebenfalls an den rootNode des Spiels angehängt.

4.3.2 Modelle

Es wurden viele verschiedene 3D-Modelle eingesetzt. Die 3D-Game Engine jME unterstützt fast alle 3D-Formate, sodass darauf nicht weiter geachtet werden musste.

Die einfachste Möglichkeit ist es eine .obj-Datei zu erstellen und diese mit Hilfe der jME ModelLoader-Klasse in ein optimiertes jME Objekt zu konvertieren und anschließend in das Spiel zu integrieren.

Das Surfboardmodell wurde mit dem Programm *Autodesk® Maya®* erstellt. Einige weitere 3D-Modelle, die in diesem Spiel erscheinen, wurden von der Internetseite 3DXtras.com bezogen.

4.3.3 Spieler

Im Folgenden wird auf die Implementierung der Eigenschaften und Bewegung des Spielers eingegangen.

4.3.3.1 Eigenschaften

Es gibt in diesem Spiel nur einen einzigen Spieler. In der Spielerklasse werden die wichtigsten Eigenschaften und das Verhalten des Spielers festgelegt.

Wichtige Eigenschaften sind:

- Beschleunigung
- Minimale und maximale Geschwindigkeit
- Lebenspunkte
- Steuerung
- Kollisionsverhalten
- Translation

Der Spieler hat zu Beginn des Spiels 300 Lebenspunkte.

4.3.3.2 Bewegung

Die Bewegungseigenschaften des Spielers werden in der PlayerHandler-Klasse festgelegt. Es gibt zwei unterschiedliche Bewegungsformeln entsprechend der Steuerungsart.

Für die Steuerung mittels Tastatur werden die Tasten A,D und W verwendet. Der Spieler kann sich vorwärts bewegen und dabei nach links oder rechts lenken.

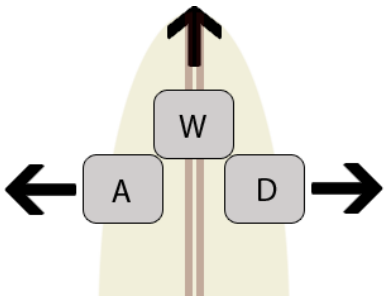

Abbildung 11: Steuerung des Spielers mittels der Tastatur

Die Bewegungen bei der Steuerung mittels Tastatur ist in der Klasse ForwardAndBackwardAction wie folgt implementiert:

```
1 public void performAction(InputActionEvent evt) {
2         if(direction == FORWARD) {
3             node.accelerate(evt.getTime());
4         } else if(direction == BACKWARD){
5             node.brake(evt.getTime());
6         }
7     }
```

Die Drehbewegung des Spielers ist in der Klasse PlayerRotationAction festgelegt. Bei der Steuerung des Spielers mit der Tastatur spielt die Haltedauer einer Taste eine wichtige Rolle, mittels derer die Bewegung umgesetzt wird.

Diese Funktion kann bei der Steuerung mittels der eigenen Körperbalance nicht verwendet werden, da die Person zu jeder Zeit auf dem Balance Board™ steht. Daher wurde die Bewegungsformel wie folgt angepasst:

```
1 if(BB_Support.getBalanceBoardSupport()== true){
2                 if(((float) balanceBoard.getXAxis()) > 0){
```

```
3                   Player.setVelocity(20f + 20f * (1 +
4                           (float) balanceBoard.getXAxis()));
5           }
6           else if(((float) balanceBoard.getXAxis()) < 0){
7                   Player.setVelocity(20f + 30f *(1 +
8                     (float) balanceBoard.getXAxis()));
9           }
10  }
```

Zunächst wird geprüft, ob das Spiel mit einem Balance Board™ gespielt wird. Falls dies der Fall ist, wird der gemessene Wert für die x-Achse aus der Klasse WiiRemoteProxy geladen und überprüft.

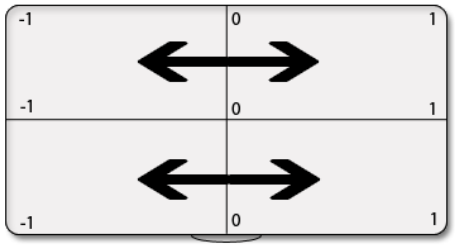

Abbildung 12: Verteilung der x-Werte des Balance Boards™

Die Werte für balanceBoard.getXAxis() ergeben sich aus der Gewichtsverlagerung des Spielers auf dem Balance Board™ , welche in der Abbildung 12 verdeutlicht werden. Der Monitor mit dem Spielgeschehen befindet sich auf der linken Seite. Ist der Wert größer Null, ist das Gewicht des Spielers nach rechts verlagert und er versucht ein wenig zu bremsen. Ist der Wert allerdings kleiner Null, lehnt sich der Spieler nach links und möchte möglichst schnell surfen. Auch für die Drehbewegung mittels eigener Körperbalance wurde eine eigene Funktion implementiert. Allerdings wurden dort die balanceBoard.getYAxis() - Werte verwendet. Wenn die Werte größer Null sind, lenkt der Surfer nach rechts, wenn sie kleiner Null sind, nach links.

In der update-Methode wird die Position des Spielers entsprechend angepasst. Solange der Spieler sich auf dem Wasser befindet, kann er sich bewegen. Berührt er den Rand, wird auf die vorherige Position zurückgesetzt. Somit wird sichergestellt, dass der Spieler stets auf dem Wasser bleibt.

Die Geschwindigkeit des Surfers ist angepasst an das Level, welches der Spieler zu Beginn des Spiels ausgewählt hat. Im Anfängermodus ist die maximale

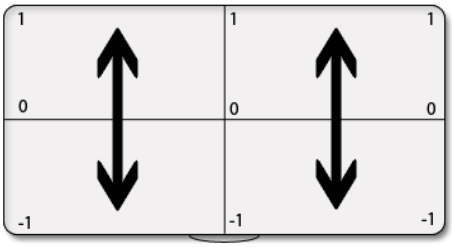

Abbildung 13: Verteilung der y-Werte des Balance Boards™

Surfgeschwindigkeit geringer als im Fortgeschrittenenmodus.

4.3.4 Gegner

Im Folgenden wird auf die Implementierung der Eigenschaften und Bewegung der Gegner eingegangen.

4.3.4.1 Eigenschaften

Es gibt vier Arten von Gegnern.

Abbildung 14: Gefahr: Krokodil

Krokodile sind die gefährlichste Art. Sie besitzen die größte Form und können den Surfer am schnellsten verfolgen. Außerdem entziehen sie dem Surfer mehr Lebenspunkte als andere Gegner. Es gibt in diesem Spiel neun Krokodile, die manuell platziert wurden.

Abbildung 15: Gefahr: Piranha und gelber Fisch

Prianhas und gelbe Fische lauern überall im Wasser und entziehen dem Surfer ebenfalls Lebenspunkte. Dabei spielt die Berührungszeit eine wichtige

Rolle. Es gibt in diesem Spiel sechs Piranhas und fünf gelbe Fische, die manuell auf dem Wasser platziert wurden.

Abbildung 16: Gefahr: Wasserlilie

Wasserlilien sind Gegner, die sich allerdings nicht bewegen. Auch sie entziehen dem Spieler Lebenspunkte. Insgesamt wurden 22 Wasserlinien manuell auf dem Wasser platziert.

Jeder dieser Gegner besitzt ein eigenes 3D-Modell. Die Position der Krokodile, Piranhas und gelben Fische ändert sich, sobald sie den Spieler angreifen. Die Position der Wasserlilien bleibt konstant.

4.3.4.2 Bewegung und Reaktion

Nur die Krokodile, Piranhas und gelben Fische können sich auf dem Wasser bewegen. Sie blicken zu Beginn in die Richtung des Spielers und drehen sich automatisch mit, sobald sich der Spieler bewegt. Zusätzlich zu der Blickrichtung wird ständig die Entfernung zwischen Spieler und Gegner gemessen und als float-Wert in einer Variable gespeichert. Dabei gibt es 3 Einteilungen:

- Entfernung > 120:
  Der Gegner bewegt sich nicht. Er blickt nur in die Richtung des Surfers.
- Entfernung zwischen 30 und 120:
  Der Gegner greift sehr schnell an, indem er seine Geschwindigkeit erhöht und in die Richtung des Surfers steuert.
- Entfernung < 30:
  Die Geschwindigkeit des Gegners wird halbiert, damit der Surfer noch eine Chance zum Entkommen hat.

Falls es einem Gegner gelingt den Surfer zu berühren, verliert dieser mit jeder Sekunde wichtige Lebenspunkte. Dabei wird die Kollision der Bounding Box des Gegners mit der Bounding Box des Surfers überprüft: Berühren sich diese, besteht eine Kollision.

Die Angreifgeschwindigkeit ist je nach gewähltem Level unterschiedlich hoch. Hat der Spieler den Anfängermodus gewählt, ist die Geschwindigkeit geringer als im Fortgeschrittenenmodus.

4.3.5 Spielgegenstände

Im Folgenden wird auf die Implementierung der Eigenschaften und Bewegung der zusätzlichen Spielgegenstände eingegangen.

4.3.5.1 Eigenschaften

Es gibt in diesem Spiel drei wichtige Spielgegenstände, die den Spielverlauf beeinflussen können:

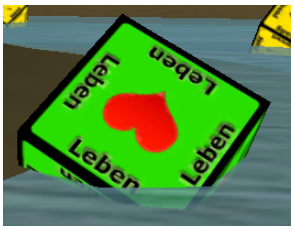

Abbildung 17: Lebensbox

Die Lebensbox besteht aus einem 3D-Würfel mit einer 2D-Textur. Bei jeder Berührung wird die Berührungszeit gemessen und die Anzahl der Lebenspunkte entsprechend um 5 erhöht. Da der Spieler sich immer bewegt, kann er nicht auf einer Lebensbox stehen bleiben.

Abbildung 18: Geschwindigkeitsbeschleuniger

Der Geschwindigkeitsbeschleuniger besteht ebenfalls aus einem 3D-Würfel mit einer 2D-Textur und sorgt bei Berührung dafür, dass die Geschwindigkeit des Spielers für drei Sekunden verdoppelt wird. Da die Endzeit entscheidend ist, ist dieser Spielgegenstand sehr wichtig.

Der Geschwindigkeitsbremser ist ein negativer Spielgegenstand. Er drosselt die Geschwindigkeit des Spielers, solange sich dieser auf dem Feld befindet. Beim Verlassen erhält er seine anfängliche Geschwindigkeit zurück.

Bei allen Spielgegenständen wird die Kollision der jeweiligen Bounding Box

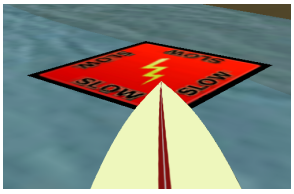

Abbildung 19: Geschwindigkeitsbremser

mit der Bounding Box des Surfers überprüft. Kollidieren diese beiden Bounding Boxen miteinander, tritt die jeweilige Eigenschaft in Kraft.

Alle Spielgegenstände werden per Zufall auf dem Wasser positioniert. Es gibt von jedem Spielgegenstand zwölf Stück.

4.3.5.2 Bewegung

Der Geschwindigkeitsbeschleuniger und die Lebensbox rotieren während des Spiels ständig. Dabei ist der Grad der Rotation und die Geschwindigkeit festgelegt. Der Geschwindigkeitsbremser bewegt sich nicht.

4.3.6 HUD

Auf dem Bildschirm des Spiels werden einige Statistiken angezeigt. Auf der linken Seite befindet sich die grafische und textuelle Anzeige der aktuellen Lebenspunkte.

Abbildung 20: Lebenspunkte als Grafiken

- **Grün:** Lebenspunkte im Bereich > 200
- **Gelb:** Lebenspunkte im Bereich von 100 - 200
- **Rot:** Lebenspunkte im Bereich < 100

Auf der rechten Seite des Bildschirms befindet sich ein orangefarbener Kreis auf dem die aktuelle Spielzeit in Sekunden angezeigt wird.

Abbildung 21: Grafik des Timers

4.3.7 Sound

In der Klasse SoundManager werden das Verhalten und die Eigenschaften der Sounds dieses Spiels festgelegt. Es gibt zwei Arten von Sounds:

- **Hintergrundmusik:** Menü, Gewinnerzustand, Verliererzustand, Highscoreliste und Spiel
- **Kollisionssound:** Bei Kollision des Spielers mit einem Gegner oder einem Spielgegenstand

jMonkeyEngine unterstützt verschiedene Audiodateien. In diesem Fall kamen .ogg- und .wav-Dateien zum Einsatz.

4.3.8 Effekte

Bei der Kollision des Spielers mit einem Gegner erscheint ein explosionsartiger visueller Effekt. Dieser wird in der Klasse ParticleEffectFactory erstellt.

4.4 Gewinnerzustand

Erreicht der Spieler das Ziel, gelangt er automatisch in den Gewinnerzustand.

Von hier aus hat er verschiedene Auswahlmöglichkeiten:

- Spiel erneut spielen (und optional einen Spielernamen eingeben)
- Highscoreliste betrachten
- Menü aufrufen
- Spiel beenden

Sollte er das Spiel erneut spielen, bleibt die Verbindung des Steuerungsgerätes erhalten. Es wird zudem eine Funktion aufgerufen, die alle Elemente des Spiels an die Anfangsposition zurücksetzt. Dazu gehört die Position des Spielers, die Position der Gegner, der Timer und das Setzen der Lebenspunkte auf 300.

Abbildung 22: Ziellinie

Abbildung 23: Gewinnerstatus

4.5 Verliererzustand

Sobald die Lebenspunkte des Spielers bei Null liegen, hat er das Spiel verloren und gelangt automatisch in den Verliererzustand. Hier hat er die gleichen Auswahlmöglichkeiten wie im Gewinnerzustand, allerdings ohne Highscoreliste. Falls er das Spiel erneut spielen will, bleibt die Verbindung zum Steuerungsgerät erhalten und die Spielelemente werden zurück gesetzt.

4.6 Highscoreliste

Die Highscoreliste ist in mehrere Klassen geteilt. Die Funktionalität der Highscoreliste ist in der Klasse HighScoreList implementiert. Die Top 5 Spieler werden in der Klasse HighScoreState grafisch ausgegeben.

Sobald ein Spieler die Ziellinie überquert, wird in einer Textdatei der Spie-

Abbildung 24: Highscoreliste

lername und die benötigte Zeit gespeichert. Anschließend werden die Zeiten miteinander verglichen und eine Reihenfolge festgelegt.

5 Zusammenfassung

5.1 Erweiterungsmöglichkeiten

Während der Umsetzung dieses 3D-Computerspiels sind stets neue Ideen entstanden, die aus Zeitgründen leider nicht alle berücksichtigt werden konnten. Anhand dieser Ideen zeigt sich deutlich, dass ein Computerspiel immer Erweiterungsmöglichkeiten besitzt. Im Folgenden sollen Einige erwähnt werden:

Um die Orientierung auf dem Amazonas nicht zu verlieren, wäre es sinnvoll einen Richtungsanzeiger zu implementieren. Dieser würde den Surfer ggf. darauf Aufmerksam machen, wenn er in die falsche Richtung surft. Somit würde der Surfer den Überblick nicht verlieren und schneller das Ziel erreichen können. Damit das Spiel gleich zu Beginn zusätzliche Aufmerksamkeit des Spielers gewinnt bietet es sich an, eine riesige Gezeitenwelle zu erstellen, die den Surfer bei Spielbeginn beschleunigt. Damit diese Welle nicht zu Performanceeinbußen im Spiel führt wäre die Programmierung eines eigenen Shaders notwendig. Um das Wasser optisch zu verfeinern, könnte man zusätz-

liche äußere Einflüsse mit hinzuziehen, zum Beispiel Wind. Demnach würde das Wasser die umliegende Szene reflektieren und dadurch sehr realistisch wirken. Um dies noch weiter auszubauen, könnte man den Wellengang mit seinen Bewegungen animieren. Es würden Wassereffekte wie Spritzer oder Ähnliches erscheinen. Zusätzlich würde das Wasser am Ufer realistisch abprallen. Die Optik eines 3D-Computerspiels beeinflusst das Spielvergnügen in hohem Maße. Damit die Umgebung des Spiels noch realistischer wirkt, könnte man mehrere Pflanzen in das Spiel einsetzen. Der tropische Regenwald besteht fast ausschließlich aus Pflanzen und Tieren. Man könnte die komplette Szene damit sinnvoll füllen. Zusätzlich zu den gerade erwähnten Erweiterungsmöglichkeiten, gibt es noch einen Entscheidenden in der Bedienung des Spiels. Während der Spieler das Spiel mit einem Nintendo® Wii™ Balance Board™ spielt, befindet er sich etwa einen Meter vom Computer entfernt. Bei jeder neuen Auswahl muss er zuerst zum Computer gehen und die entsprechende Auswahl mit der Maus tätigen. Dementsprechend wäre die Integration eines Nintendo® Wii ™ Remotes™ sinnvoll. Dies ist ein Controller von Nintendo® , welcher sich wie eine schnurlose Maus verhält. Die Verbindung mit dem Computer läuft wie bei einem Nintendo® Wii™ Balance Board™ über eine Bluetooth® - Schnittstelle.

Zu den oben genannten Erweiterungen gibt es natürlich noch viele weitere Möglichkeiten, das Spiel zu ergänzen.

5.2 Fazit

Das Thema dieser Studienarbeit wurde ausgewählt, um einen umfangreichen Einblick in die vielfältigen Bereiche der Computergrafik zu erhalten und in ein eigenes Projekt umzusetzen. Das Ergebnis dieser Studienarbeit ist ein funktionsfähiges 3D-Computerspiel, welches zusätzlich zu der Computertastatur, die Steuerung mittels eigener Körperbalance bietet.

Sämtliche Zielvorgaben wurden durch die erfolgreiche Umsetzung dieses 3D-Computerspiels erreicht.

5.3 Ausblick

Anhand dieser Studienarbeit erkennt man sehr deutlich, dass die neuen Steuerungsmöglichkeiten der Videospielebranche leicht in die Computerspielebranche integriert werden könnten. Dementsprechend wäre es möglich, Spiele und Applikationen für den Computer zu entwickeln, die weitere Personengruppen ansprechen würden. Beim diesjährigen Computervisualistik-Tag der Universität Koblenz war ein großes Interesse an dieser neuen Steuerungsmöglichkeit zu verzeichnen. Personengruppen unterschiedlichen Alters haben dieses Spiel gespielt und waren begeistert von den Möglichkeiten der Steuerung.

6 Literaturverzeichnis

Literatur

[Eber05] Eberly, David H. : *3D Game Engine Architecture*. Morgan Kaufmann, 2005.

[Eber06] Eberly, David H. : *3D Game Engine Design - 2nd ed.*. Elsevier Ltd, 2006.

[Moe08] Akenine-Möller, Tomas; Haines, Eric; Hoffman, Naty: *Real-time Rendering*. Peters Wesley, 2008.

[Sal03] Salen, Katie; Zimmerman, Eric : *Rules of Play: Game Design Fundamentals*. Mit Pr, 2003.

Webverzeichnis

[Dia09] Diamon, Michael: *WiiRemoteJ Bibliothek*. 2009 *http://www.wiili.com/forum/wiiremotej-f68.html*

[Wiki09] User-Wiki, *Technische Grundlagen des Balance Boards*^{TM}. 2009 *http://www.wiibrew.org/wiki/Wii_Balance_Board*