Brot und Spiele - Seminar
im Wintersemester 2011/2012

Utopya

Philipp Schardt, Christian Neurohr, Kevin Kohler, Peter Geßner,
Nikolaj Woroschilow, Tarek Mawad, Friedrich Van Schoor

01.03.2012

Dozenten

Dr. Lübomira Spassova, Dr. Michael Schmitz, Sven Gehring
Universität des Saarlandes

Prof. Burkhard Detzler, Mert Akbal
Hochschule der Bildenden Künste

Inhaltsverzeichnis

Zusammenfassung

Das Spiel Utopya enstand in einem Kooperationsprojekt zwischen dem Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI), der Hochschule der Bildenden Künste Saar (HBK), dem Studiengang Medieninformatik der Universität des Saarlandes und dem Saarländischen Rundfunk (SR).
Im neuen Foyer des saarländischen Rundfunks sollte eine auditiv-visuelle interaktive Projektionsumgebung entstehen. Diese soll Besuchergruppen des Hauses und der Konzerte in kurzweilige und gestalterisch ansprechende Anwendungen einbinden.
Spielerisch-immersive Ansätze waren ebenso möglich wie künstlerisch- responsive Ausgestaltungen. So kann eine Erlebniswelt hoher atmosphärischer Qualität geschaffen werden.
Der installierte reaktive Projektionsscreen (ca. 8 x 2,5 Meter) ist über mehrere eingebaute Kinect-Sensoren gesteuert. Über diese kann auch die angeschlossene Audioanlage und die LED Grundbeleuchtung kontrolliert werden.

1 CD-Inhalt

1.1 UnityProgramm

In diesem Ordner befindet sich das Unity-Projekt aus dem man die fertige exe erstellt. *Library* und *Temp* enthält nur die Bibliotheken mit Skripten die Unity schon bietet. Wichtig ist der *Assets*-Ordner. Dieser enthält all unsere Spieldateien (Skripte, 3D-Objekte, Sounds), die in Unterordner mit den entsprechenden Namen unterteilt sind.
Außerdem befindet sich hier auch der *UnityKinectPlugin_Philipp.dll*. (Normalerweise in einem Plguin-Ordner, aber in der Free-Version von Unity3D kann man es durch diesen Trick auch so hinbekommen, dass Unity-Free Plguins aktzeptiert). Dieses funktioniert für das Microsoft Kinect SDK Beta 2.
Möchte man unser Projekt mit dem Microsoft Kinect SDK 1.0 benutzen, muss man diesen mit der dll aus *Kinect SDK 1.0/Debug/UnityKinectPlugin_Philipp.dll* austauschen

1.2 Kinect SDK Beta 2

Hier befinden sich die zwei Unterordner *UnityExe* und *UnityKinectWrapper*. In *UnityExe* befindet sich eine ausführbare Datei von unserem Programm. Diese funktioniert - wie sich schon erahnen lässt - nur wenn man das Kinect SDK Beta 2 installiert hat.
UnityKinectWrapper enthält das Unity-Plugin mit dem die Kinect-Funktion bereitgestellt wird.

1.3 Kinect SDK 1.0

Enthält die selben Daten wie *Kinect SDK Beta 2* mit der kleinen Änderungen, dass die exe nur richtig ausgeführt, wenn das Microsoft Kinect SDK 1.0 installiert ist.

1.4 Screenshots und Videos

Hier sind die Screenshots und Videos aus unser Dokumentation zu finden.

2 Das Projekt

2.1 Realisierung

Für die Realisierung unseres 3D-Projekts benutzen wir das Programm Unity3D. In diesem kann man Grafiken und dreidimensionale Objekte aus dem Designprogramm 3ds Max importieren und weiterverarbeiten. Dabei können die einzelnen Objekte in Unity mit verschiedenen Skripten belegt werden und so kontrolliert werden. Die Skripte wurden in C# geschrieben.
Zusätzlich mussten die Kinect-Funktionen in Unity bereit gestellt werden. Dies gelang durch ein Plugin, das in C++ geschrieben wurde.

2.2 Umgebung

Das Projekt stellt eine dreidimensionale Dschungel-Szene dar. Der Betrachter blickt direkt auf einen Vordergrund voller Sträucher und Bäume. Dabei kann man verschiedene Gegenstände und Objekte durch Berührung bewegen oder eine Aktion von ihnen auslösen.

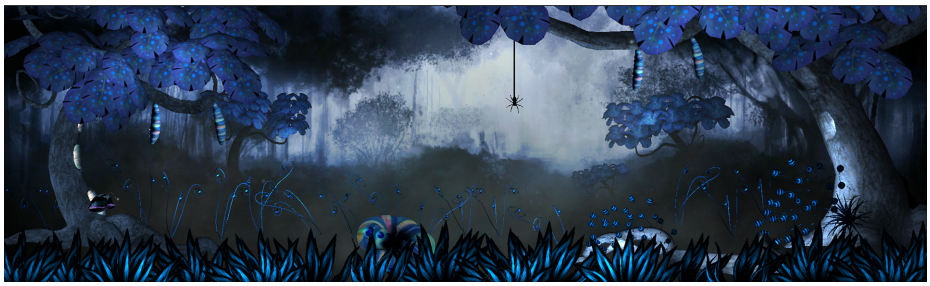

Abbildung 1: Ein Blick auf die Dschungelszene Utopya

2.3 Interaktion

Die Betrachter werden mithilfe der Kinect erfasst und jeder Benutzer erhält seine eigene Farbe, die die Farbe das Partikelsystemen der Hände, sowie die Pflanzen am unteren Rand bestimmt. Nun können die Betrachter mit der Umwelt interagieren und die einzelnen Objekte ansteuern und berühren. Diese reagieren dann individuell. Folgende Objekte sind dabei enthalten:

- Bei Berührung der Baumkronen der Bäume rechts und links fallen Blätter hinunter. Diese Blätter kann man berühren und so auffangen oder etwas wegschleudern
- Am linken und rechten Baum hängen jeweils zwei kleine Raupen, die man berühren kann. Es gibt drei verschiedene Interaktionen, die sich daran orientieren wie schnell man gegen die Raupe haut: Die Raupen bewegen sich etwas, fallen herunter oder werden aus den Bild geschleudert.

- Ein “Fisch im Baum”, der mit seiner Riesenzunge schlabbert, wenn man ihn normal berührt. Zusätzlich kann man ihn zurück in sein Loch hauen. Dann kommt er aus dem zweiten Loch im linken Baum wieder heraus.
- Ein Vogel am linken Rand, welcher erscheint, wenn man sich in der Nähe des linken Baumes befindet. Er pickt oder kräht zufällig.
- In der Mitte befinden sich Fledermäuse, die bei Berührung durch die Umgebung fliegen.
- Eine dicke Raupe, die sich in der Mitte befindet. Diese reagiert auch verschieden, je nachdem wie stark man sie berührt. Sie schaut sich leicht oder etwas stärker um. Außerdem wird sie ganz groß, wenn eine Person links oder rechts zu nahe (ca. 1 Meter) vor die Leinwand tritt.
- Eine Spinne am rechten Baum, die sich hochseilt, wenn eine Person sie berühren will. Geht man wieder weg, seilt sie sich ab und bewegt sich ein wenig.
- Ein Affe, welcher sich auf der rechten Seite im Baum befindet. Dieser wird aktiviert, wenn man rechts, oberhalb von der Canivore-Pflanze den Trigger trifft. Dann klettert der Affe über den Baum. Danach ist der Trigger am Baum deaktivert bis man einen zweiten Affen-Trigger trifft. Dieser befindet sich zwischen Spinne und den zwei kleinen Raupen am Baum. Er bewirkt, dass der Affe vom Baum fällt. Wenn man schnell genug ist, kann man den Affen sogar beim klettern schon herunterwerfen.
- Zwei Canivore-Pflanzen, die man mit den Händen berühren kann und die sich dann physikalisch korrekt bewegen.

Außerdem bewegen sich die Blätter der vorderen Bäume, die Grasbüschel und die vereinzelten Grasfühler im Hintergrund permanent. Zusätzlich wird die Dschungleatmosphäre durch Partikelsysteme, wie der Rauch und Partikel, die sich langsam von links-unten nach rechts-oben bewegen, verstärkt.
Die gesamte Szene ist mit einem Urwald Geräusch hinterlegt und die einzelnen Charaktere und Aktionen haben einen spezifischen Sound.

Es befindet sich ein Video auf dieser CD, indem die Interaktionen ausgeführt werden.

2.4 Konzept

Die Umgebung ist eine Mischung aus Objekten mit Texturen und Objekten, die mehr oder weniger nur als Silhouette zu erkennen sind. Außerdem ist die Umgebung stark in Blautönen gehalten. Dadurch soll eine geheimnisvolle Stimmung aufgebaut werden.

Die Charaktere und Objekte sind so platziert, dass sie Animationen eines sichtbaren Objekts auslösen oder sogar Objekte auslösen, die man nicht direkt erkennt. Zum Beispiel dienen die kleinen Raupen am Baum auch dazu, dass man den Baum "schüttelt" und Blätter herunterfallen.

Die Avatare sind so aufgebaut, dass man sich schnell im Bild zurechtfindet. Die Hände sind mit einem Partikelsystem ausgestattet, das der Bewegung folgt. Je schneller man die Hände bewegt, desto größer werden die Partikel. Zusätzlich färben sich noch die unteren Pflanzen in die gleiche Farbe, wie die Partikel. Jeder Betrachter hat eine eigene Farbe.

2.5 Aufbau

2.5.1 Szenen

Die Szenen oder auch Level sind in Unity unter *Assets* zu finden.

Starting ist die vorgeschaltete GUI mit dem Auswahlmenü.

Utopya ist die komplette Szene mit der 3D-Umgebung.

Abbildung 2: Starting-Scene

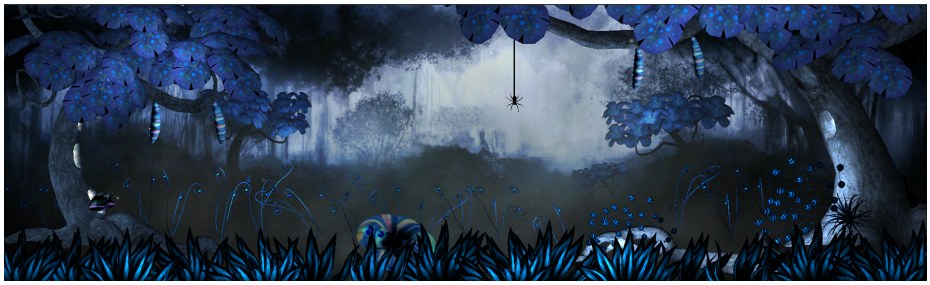

Abbildung 3: Utopya-Scene

2.5.2 Trigger

Trigger sind ein elementares Element unserer interaktiven Steuerung. Die Kinect steuert die Avatare und diese haben an den Händen Collider mit Rigidbody.
Collider sind Unity-Elemente die die Berührung mit anderen Objekten, die auch einen Collider besitzen, erkennen.
Mit einem Rigidbody kann man die Physik eines Objekts steuern, wie zum Beispiel die Gravity. Außerdem braucht man diesen, um die Unity-Methoden (z.B. OnTriggerEnter) - welche die Trigger steuern - auslösen.

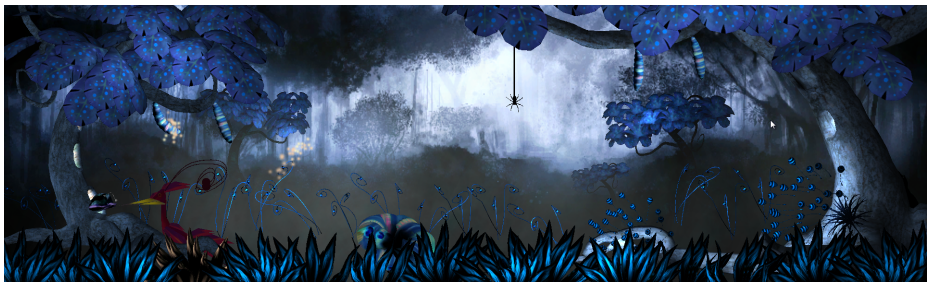

Abbildung 4: Utopya: Der Vogel wurde aktiviert und bewegt sich

2.5.3 Skripte

Über Skripte kann man alle GameObjects in der Szene und ihre Komponenten steuern und sogar - wie im Fall der Blätter - neue GameObjects erzeugen.
Public Variablen in Skripten kann man in der Unity-Oberfläche Werte oder sogar andere Objekte (Referenzen) leicht zuweisen, die man dann in seinem Skript verwenden kann.

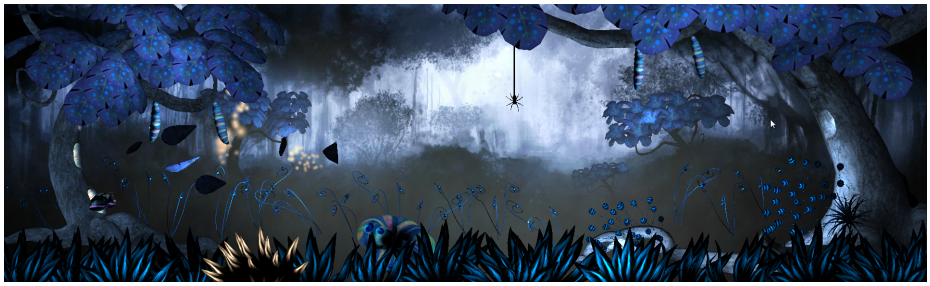

Abbildung 5: Utopya-Scene mit fallenden Blättern

2.5.4 zwei synchrone Seiten

Dadurch, dass Unity keinen Vollbildmodus für zwei Bilschirme bietet und auch das Skeleton-Tracking nur mit einer Kinect funktioniert, mussten wir auf zwei

PC's ausweichen.
Linke und rechte Seite sind komplett synchron aufgebaut, obwohl man immer nur eine Seite sieht. Wir verschieben beim Start die Kamera entweder auf die linke Seite, wenn es als Server gestartet wird oder auf die rechte Seite, wenn es als Client gestartet wird. Gleichzeitig werden auch die richtigen Avatare aktiviert (ChooseSide). Dies bietet den Vorteil, dass jeder PC Server und Client sein kann. Ein Nachteil ist natürlich, dass unnötig Performance beansprucht wird. Dies könnte man natürlich leicht ändern, aber wir wollten es so leicht wie mögliche startbar machen. Trotzdem haben wir darauf geachtet, dass es auf den gegebenen Computern abspielbar bleibt.

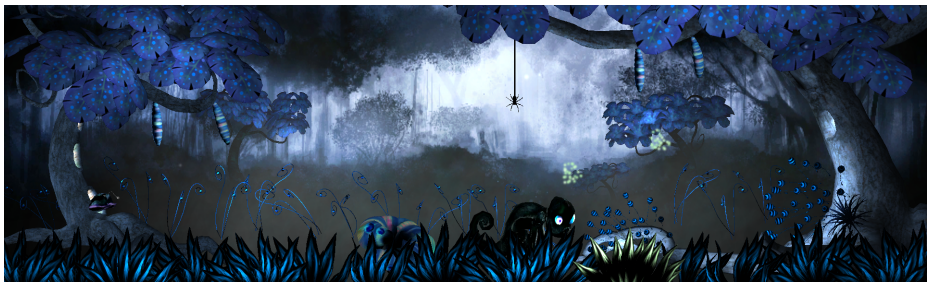

Abbildung 6: Affe springt weg

2.5.5 Netzwerk

Unity bietet schon eine Netzwerkunterstützung, die wir genutzt haben. Dabei muss jedes Objekt das man über das Netzwerk verbinden will eine NetworkView besitzen. Vom Server zum Client kann man die Position der Avatare (wirklich übertragen werden nur Hände, Füße, Kopf und Hüfte) automatisch synchronisieren. Umgekehrt geht dies über das SendPosition-Skript. Außerdem wird permanent der Status der einzelnen Körperteile synchroniseirt. Der Status bestimmt, ob der Avatar von der Kinect getracked ist und aktiviert die Collider für die Kollisionsbestimmung und die Partikel der Hände.
Um die Verbindung der beiden PC's zu erleichtern nutzen wir den MasterServer von Unity. Dadurch muss man nicht direkt eine IP-Adresse des zu verbindenden Servers angeben, sondern kann über den Typ und den Namen des Servers die Verbindung herstellen.

2.5.6 KinectPlugin/KinectWrapper

Der KinectWrapper stellt die Verbindung zwischen Kinect und Unity her. Im Plugin werden die Funktionen der Kinect so bereitgestellt, dass man sie teilweise in den Unity-Skripten aufrufen kann oder schon berechnete Daten benutzen kann.
Das Plugin überträgt die zwei maximal möglichen, voll getrackte Skelette an Unity, sowie die Tiefeninformationen. Die Tiefeninformationen nutzen wir zum

Beispiel um zu erkennen, ob die Benutzer zu nahe an der Leinwand stehen. Pro PC wäre es möglich drei Kinects anzuschließen und die Tiefendaten auszulesen. Die Skelettinformationen funktionieren nur bei einer der drei Kinects (hasSkeletonEngine), da dies durch das Microsoft Kinect SDK beschränkt ist. Es wäre auch möglich, die Hip-Center-Positionen von vier weiteren Skelettdaten in Unity zu benutzen, da die Kinect dies erlaubt. Dennoch benutzen wir nur die vollständig getrackten Skelette und nur eine Kinect pro PC.

3 Programm ausführen

3.1 Voraussetzungen

Man braucht zwei PC's mit Windows 7, die das Microsoft SDK Beta 2 installiert haben. Jeder PC ist jeweils an einen Beamer angeschlossen. Außerdem müssen Beide mit dem Internet verbunden sein, da die Verbindung über einen Master-Server von Unity läuft. (Dies kann man mit network_direct.cs umgehen, aber man muss dort explizit die IP-Adresse des Servers in den Client eintragen). Dabei ist zu beachten, dass die (Windows-)Firewall die utopya.exe akzepietert (Wird beim ersten Ausführen erfragt).
Jeder PC muss eine angeschlossene Kinect haben, die jeweils in der Mitte des Bildes steht. Es muss sichergestellt sein, dass eine Kinect angeschlossen ist, ansonsten hängt sich Unity (bzw. das Plugin) auf.

3.2 Startvorgang der exe

1. Utopya.exe (Utopya1.0.exe für Kinect SDK 1.0) starten

2. Es öffnet sich ein Auswahlfenster, in dem die Auflösung und die Grafik-Einstellungen auswählbar sind. Die Auflösung muss auf 1280x768 stehen (auch wenn der Beamer eine geringere Auflösung besitzt). Die Grafikeinstellungen können auf fantastic gesetzt werden. Windowed sollte ausgeschaltet sein.

3. Das Programm startet mit einem Auswahlmenü. Auf dem linken (Bildschirm) der beiden PC's muss man "Starte Linke Seite (Server)" klicken. Auf dem rechten PC muss man warten bis der Server angezeigt wird und sich dann mit diesem über den Button "verbinde mit Linker Seite (Server)" verbinden.

4. Das eigentliche Programm startet und man kann spielen. Die Szene startet erst, wenn Server und Client wirklich verbunden sind.

5. Schließen kann man das Programm mit der Escape-Taste der Tastatur.

Die Benutzer können im Bereich von 80 cm bis vier Metern vor der Leinwand interagieren. Bei zu geringer Entfernung zur Leinwand (ca 1m) wird man durch die Raupe in der Mitte darauf hingewiesen, dass man weiter zurück gehen sollte.

4 Installation der Software

4.1 Microsoft Kinect SDK Beta 2

Essentiell wichtig ist die Installation des Microsoft Kinect SDK Beta. Diese kann von der offiziellen Hompage heruntergeladen werden. Die Installationsanleitung ist auch dort zu finden.
Wir verwenden die 64-Bit-Version. Es sollte aber auch mit 32-Bit funktionieren. Durch die Installation funktioniert das Kinect-Plugin für Unity.

4.2 Microsoft Kinect SDK 1.0

Das Kinect Plugin funktioniert auch mit dem Microsoft Kinect SDK Version 1.0. Dazu muss man KinectWrapper1.0 öffnen, kompilieren und die entstehende dll im Unity-Programm ersetzen.

4.3 Microsoft Visual Studio

Um das KinectPlugin/KinectWrapper für Unity zu kompilieren braucht man Visual Studio. Dies kann man auf Microsofts Homepage herunterladen.

4.4 Unity3D

Um das Programm über die exe auszuführen muss Unity3D selbst nicht installiert sein.
Zum Betrachten des Programms muss Unity3D installiert werden.

4.5 3ds Max

3ds Max ist das Designprogramm mit dem die Objekte für die Umgebung entworfen wurden. Man kann sich die verwendeten Objekte auch in Unity anschauen, doch zum genaueren Betrachten dieser Objekte muss man 3ds Max installieren.

5 Arbeitsaufteilung

Eine grobe Historie unseres Fortschrittes ist auf game.philippschardt.net nachzulesen.

5.1 Philipp

Philipp hat sich um die Verbindung zwischen Kinect und Unity gekümmert und ein Plugin für Unity entwickelt, das die Funktionen der Kinect bereitstellt. Außerdem war er für das Netzwerk zuständig, um die zwei Bildschirmhälften und die Avatare zu synchronisieren. Dafür war auch eine vorgeschaltete Szene notwendig, die die Verbindung der PC's erleichtert und den Start der beiden Seiten steuert. Ein weiteres Produkt seiner Experimente mit Unity ist die Canivore-Pflanze, die die Physik-Engine von Unity ausnutzt. Dadurch kann man mit seiner Hand durch die Fühler der Pflanze greifen und diese bewegen.
Zusätzlich war er bei der Lösungsfindung des Problems des fehlerhaften Imports beschäftigt. Nachdem dies gelöst war, war er für den richtigen Import und das Einstellen der Animationen der Objekte zuständig.

Im Folgenden eine genaue Auflistung seiner Arbeiten:

- KinectPlugin (C++, UnityKinectPlugin, stdafx, targetver)
- KinectWrapper und KinectPointController (in Unity)
- Network (Network_MS, SendPosition, SendData, ChooseSide; nicht verwendet: Network_direct; hier muss die IP-Adresse des Servers im Skript explizit angegeben werden, aber funktioniert)
- Starting-Scene (mit GUI)
- Canivore-Pflanze (Ausnutzen der Physik von Unity)
- Animationssteuerung der Objekte (AnimRaupe, AnimSpinne, AnimVogel)
- Einstellung der Umgebungsanimationen (Baumblätter, Fühler-Pflanze)
- Soundeinstellungen (Sound mit Animationen synchronisieren)
- Dokumentation (Beschreibungen, Screenshots, Video)
- Kommunikationsleitung (Gruppentreffen planen)

5.2 Christian

Christian hat sich zum einen um die Darstellung der Spieler-Avatare in Utopya gekümmert, welche letztendlich durch Partikelsysteme sowie die Einfärbung der Pflanzen unten realisiert wurde. Die Größe der Partikel an den Händen variiert dabei auch je nach Stärke der Bewegung.

Außerdem hat er verschiedene Skripte geschrieben, wie zum Beispiel das TriggerLeaves-Skript und das AnimBlaetter-Skript. Diese beiden sind für die fallenden Blätter zuständig. Während AnimBlaetter die Bewegung der fallenden Blätter in Verbindung mit einem sogenannten Fixed-Joint simuliert, steuert TriggerLeaves die Ausschüttung der Blätter durch Wedeln mit den Händen in der Baumkrone. Bei dieser Art der Interaktion konnte die Physik-Engine von Unity gut genutzt werden.
Bei den Skripten für die Character-Animationen hat er außerdem dafür gesorgt, dass meistens je nach Stärke der Handbewegung eine andere passende Animation des Characters abgespielt wird.

Im Folgenden eine genaue Auflistung seiner Arbeiten:

- Partikelsystem der Avatare mit verschiedenen Farben (Farbänderung im KinectPointController)
- Partikelsystem der Umgebung (Rauch und diffuse Partikel, Particles-Skripts)
- Pflanzen im Vordergrund, die die Farbe der getrackten Person annehmen und deren Textur bei Kontakt animiert wird
- fallende Blätter (TriggerLeaves, AnimBlaetter)
- Kameraposition und Kameraperspektive (nicht verwendet, die Kamera würde sich an allen vier Avataren orientieren)
- Animationssteuerung der kleinen und großen Raupe (AnimRaupe, AnimKleineRaupe)
- Animationssteuerung der Objekte (AnimAlbino, AnimRaupe, AnimSpinne, AnimVogel, AnimFisch)
- Sounds finden und anpassen
- Verpackungsdesign

5.3 Kevin

Kevin hat die einzelnen Objekte von 3DS Max eingebaut und angepasst. Er hat die verschiedenen Texturen der Objekte eingebaut und dem Licht angepasst. Außerdem hat er sich auch mit der Animation des Vogels beschäftigt und hat die einzelnen Sounds zugewiesen und bearbeitet. Desweiteren hat er sich mit dem allgemeinen Gesamtbild des Projektes auseinandergesetzt. Er war für die Dokumentation des Projektes verantwortlich.

Im Folgenden eine genaue Auflistung seiner Arbeiten:

- Lichteinstellungen

- hat sich mit dem Importproblem beschäfigt (Texturen richtig setzen, Shader anpassen)
- Dokumentation
- Animationssteuerung des Vogels
- Sounds finden und anpassen

5.4 Peter

Peter hat sich mit Unity und seinen einzelnen Funktionen auseinandergesetzt. Er hat sich teils mit dem Problem „Kinect-Unity“ über den KinectWrapper befasst und Skripte für einzelne Objekte und Animationen geschrieben. Außerdem hat er beim Aufbau der Umgebung in Unity mitgearbeitet.

Im Folgenden eine genaue Auflistung seiner Arbeiten:

- hat sich mit der Einbindung und dem Steuern von Animationen beschäftigt
- hat sich mit dem Importproblem beschäfitgt
- Animationssteuerung des Vogels

5.5 Nikolaj, Friedrich, Tarek (Designer)

Die Designer waren für die Grafiken, die dreidimensionalen Objekte und deren Animation, sowie für die Ausarbeitung des Stils und des Konzeptes zuständig. Außerdem haben sie die Programmierer bei der Anordnung der Objekte in Unity unterstützt.