**Vi-Char Motion Capture Intermediate Report**

**Motion Capture Development Team**

Travis Dos Santos-Tam

Ian Earle

Max Gobel

Daniel Porter

**Introduction**

This document describes motion capture specifications of software requirements to support the overall Vi-Char project by building the interface between the Microsoft Kinect and the game.

Contributors to this document and members of the team include:

—Travis Dos Santos-Tam, a senior majoring in computer science, is experienced with independent and small group C++, C♯, Java and Python programming.

—Ian Earle, a senior majoring in computer science. Has experience with C++, Java, and Python programming languages. Brief experience working in parallel with small teams for software development projects, largest number was 3 team members. Very limited stint as a project manager for an independent student game project. Team was 6 people, and the project lasted about 3-4 months before folding due to lack of time availability for several team members.

—Max Gobel, a senior. Experience with C#/Visual C++, C/C++, Scala, Java, Haskell and Python. Limited group development experience.

—Daniel Porter, a sophomore majoring in computer science and psychology. Has experience with Java, C, Objective-C, and PHP. Some experience programming in a team of two developing various software projects.

**Executive Summary**

The Motion Capture Development Team is responsible for developing the interface between the physical world and the virtual puppet within the game world, by capturing motion with a Microsoft Kinect and transforming these data into digital game commands. While the team’s task is certainly challenging — converting physical gestures to digital instructions — accomplishing the goal to bring a whole new level of interaction and play to the game will prove rewarding.

A platformer game requires a few things: that characters be able to move about and characters may have to interact with others and/or their environment. Furthermore, unique to this particular platformer and motion capture project is the necessity of a single character being manipulated by multiple bodies, a feature the Kinect is not generally used for. The team will need to find a way to identify each “player”, potentially for a number of players larger than the Kinect is used to handling.

We will inevitably be working closely with the Game Development team, as they are the ones who will need to specify exactly the contents of the movement dictionary that we will be producing, so that the input from the puppet can interact with the rest of the world.

**Application Context**

The Motion Capture Development Team is responsible for developing a solution to capture physical motion with a Microsoft Kinect and transform these data into digital game commands. Kinect presents unique opportunities for game interaction as players are required to involve and immerse themselves on a whole new level. Leveraging this technology, our team aims to create the possibility of having four players control two characters, pairing individuals to work together and compete against the other pair. The goal of the team — and of all the other teams — is to deliver a fun, interactive game which appeals to people of all ages and abilities.

Our intention is to have characters that are able to move themselves in a variety of ways, especially considering that the game will be based on the “platformer model.” Moving through the different dimensions, the characters may also encounter obstacles, which through their actions will be able to manipulate, avoid or circumvent. Consequently, the characters must be able to move, in all sorts of directions including jumping and crouching. Other motions that may be possible include flying or hovering. All motions will need to have some sort of gesture or combination of gestures associated with them to execute.

Additionally, because the characters are competing against each other, some sort of weapons or fighting capability must be implemented. In either or both cases, a class of gestures will also need to be determined and the system will need to respond to these actions.

**Functional Requirements**

**Normal Game Play**

***Actors:***

-Player(s)

***Scenarios:***

-The players, once the game is configured and launched, will be able to immediately control their puppets with their teammates. Acting out certain moves will affect how the characters move and where they go about onscreen.

***Preconditions:***

-Kinect must be on, calibrated, and pointing at the desired puppeteering space

-Users must be within the puppeteering space, and understand the “dictionary” of moves available to control the puppet

-Kinect must be setup to transmit to the server hosting the game engine.

***Postconditions:***

-Recording of events will be saved for playback/”ghosting”

***Alternatives:***

1. User unfamiliar with Control Dictionary:

-Kinect will walk User through the various actions available and the physical motions associated with them.

2. Kinect is not connected to a valid server:

-Kinect will prompt user to verify the state of the connection, and ensure that the server is running and configured.

3. User not detected:

-Kinect will prompt user to either enter the playing space, or ensure that the Kinect is properly setup.

**Other Requirements:**

***Must Support up to 4 players.***

A total of 4 players must be able to be in the play area and issue commands to their portion of the puppet simultaneously.

***Must contain a dictionary which maps player motions to puppet actions***

To allow for communication with the game engine, the Kinect must have access to a dictionary of actions shared with the game engine, and map player motions to each of those actions.

**Non-Functional Requirements**

***Response Time***

One of our goals is to ensure that the system is responsive, as slight delays between player gesturing and system recognition and command execution will prove detrimental to the overall Vi-Char experience. However, we also recognize the constraints and limitations of imaging technology, especially when the environment is difficult to control, considering working in 3D space and swiftness of moves. It is finding that right balance of reactiveness and “failure” tolerance that will ensure that the user has a fun experience.

Additionally, accurate translation of physical gestures will contribute to successful accomplishment of this goal. A user may expect, so long as his or her gestures are clear and deliberate and conform to the tolerances hardwired in the Kinect and system, that the correct character action will occur.

***Consistency***

Good user experience is paramount to any good product and this is achieved by paying attention to detail and recognizing the value of consistency. One of the goals of the project must be to ensure that one type of physical motion corresponds to one type of action or a group of similar actions (e.g. a physical punch will translate into a punch by the character, an arm swing might translate into walking and, if faster, running, etc.).

***Platform Compatibility***

In order for the entire system to work, the Kinect system must be compatible with the platform the game engine is running on.

**Timeline**

1. Establish version control system (Sept 17th-19th)

2. Setup Kinect development environment (Sept 19th-24th)

3. Begin work on Kinect Motion-To-Event Translator (Sept 24th-Oct 22nd)

4. Discuss Events to be contained within the shared Control Dictionary with Game Engine Group (Oct 22nd-Oct 29th)

5. Adjustment time based on input from the Game Engine Group (Oct 29th-Nov 12th)

5. Determine whether or not other groups have any unique requirements for the Kinect.

(Nov 12th-Nov 19th)

6. Rework Requirements Document accordingly. (Nov 19th-Nov 24th)

7. Flex time for new requirements, additional input from other groups. (Nov 24th-Nov 31st)

**Potential Challenges**

We are currently unaware as to how well the Kinect will function while receiving simultaneous input from 3 to 4 different sources. Most games designed for Kinect use function optimally with one person in front of the device, and decently in cases requiring two people. Three people has been show to work in some cases, but we have seen no evidence of a Kinect being able to process simultaneous input from 4 individuals. This may prove to be a restriction on our potential options in the long run, especially if we only have one Kinect available to work with, and there’s no precise way to tell how possible our particular scenario will be without extensive testing on this particular subject.

Another potential challenge may be background movement. Some initial experience with the Kinect system has shown that unrelated background movement can cause problems with the motion capture, so steps will need to be taken to reduce this, either computationally or environmentally.

**Glossary and References**

**Glossary:**

-Control Dictionary: A set of events shared with the game engine, which are mapped to a set of player actions. Upon perceiving the appropriate player action, the Kinect will send the corresponding event to the game engine for processing.

**References:**

-Microsoft Kinect for Windows SDK (<http://www.microsoft.com/en-us/kinectforwindows/)>

-Kinect Dev Tutorials (<http://www.kinecthacks.com/kinect-sdk/>)

-Zigfu Development Kit (<http://zigfu.com/>)

-Possibly useful for integrating with the game engine

-Develop Kinect Community (<http://developkinect.com/>)

-Getting Started with Kinect Development (<http://channel9.msdn.com/coding4fun/kinect/Getting-started-with-Kinect-development-quickly-with-the-Kinect-for-Windows-Quickstart-Series>)