

**Design Document for:**

# Canyon Adventure

**Taking the River Home**

“The journey to recovery starts here!”™

All work Copyright ©2012 by Ohio State University

Written by Tristan Reichardt, Daniel Bibyk, David Maung, Juan Roman

Version # 2.00

Saturday, December 08, 2012Table of Contents

Canyon Adventure 1

Design History 3

Game Overview 3

Philosophy 3

Philosophical point #1 3

Philosophical point #2 3

Common Questions 3

What is the game? 3

Why create this game? 3

Where does the game take place? 4

What do I control? 4

What is the main focus? 4

What’s different? 4

Hours of Gameplay 4

Victory Conditions 4

Feature Set 4

Landscape 4

Gameplay 6

The Game World 9

Overview 9

Canyon 9

Maze 9

The Physical World 9

Key Locations 9

Travel 9

Scale 9

Objects 9

Time 10

Rendering System 10

Camera 10

Lighting 10

User Interface 10

Overview 10

Kinect 10

Keyboard 11

Sound Effects 11

Overview 11

XACT 11

Sound Design 11

Character Rendering 12

Procedural Content 12

Gesture Recognition 13

Game State 16

Class Diagram 17

# Design History

|  |  |
| --- | --- |
| **Date** | **Description** |
| 9/11/2012 | Document was created and added to repository. Basic descriptions of major sections written. |
| 9/12/2012 | Class diagrams and state machines added. |
| 10/10/2012 | Information on procedural content and information updated. |
| 11/2/2012 | Interface description added with gesture information. |
| 11/8/2012 | Images added for procedural content, interface, and game features. |
| 12/1/2012 | Gesture recognition descriptions and game content updated. |
| 12/07/2012 | Final revisions made to all sections. |

# Game Overview

## Philosophy

### Philosophical point #1

This games main purpose is to try and help those who have suffered a stroke. A portion of the brain dies when a person suffers a stroke. In order to recover from the incident, the patient must train a different part of the brain to perform the same tasks they were able to do before. This can be an extremely trying process, having them achieve this rewiring of the brain with a game will hopefully alleviate the frustration. However, since it is a game, we want it to be playable by everyone. Hopefully we will be able to implement another version or mode that will allow the game to be enjoyable for the everyday gamer.

### Philosophical point #2

Our game will run on computers. The medical facilities will find it much easier to utilize a computer or laptop for their treatments than setting up an Xbox. This will make the game easier to transport to the patients who are in need of it.

## Common Questions

### What is the game?

Canyon Runner is a Kinect-based adventure game. The player will control a single protagonist in a boat through multiple environments. The main aspects of gameplay will be navigation, collection of items, and the evasion of obstacles. The player will progress from a simple outdoor river, to a maze-like cavern. As the player travels through water, he will have to explore procedurally developed paths, and collect items to progress through levels.

### Why create this game?

The creation of this game is meant to help stroke patients recover. The core mechanics of the gameplay are set around physical therapy motions. The gameplay will range from simple hand motions to complex full arm motions.

### Where does the game take place?

This game will take place in a river that is surrounded by mountains and trees. The boat will run down this river which will be filled with occasional obstacles. Eventually they will also enter a dark cave which they will have to navigate through before looping back around to the top of the river.

### What do I control?

As the player, you will control the robot inside of the boat. You will have to maneuver the boat and perform tasks in order to advance. More mechanics may be added later in order to diversify gameplay.

### What is the main focus?

The player will face a number of challenges and tasks in the world. These will be specially designed to ensure full movement in the players arm. In order for a stroke victim to fully recover they must be put in situations where they need to use their body in the same was as they had used it before. An example of this is the fishing aspect to the game; the player has to go through the motion of dragging a net through the water.

### What’s different?

The main thing that sets this game apart from almost all other games currently on the market is that it’s driven by medical use. While we want the game to be enjoyable and fun, the main goal is to help patients recover.

## Hours of Gameplay

The goal of this game is to have 30 hours of game play. Normally, a patient will undergo roughly 30 hours of physical therapy, if the game is to replace conventional therapy, it must be able to last as long as conventional therapy.

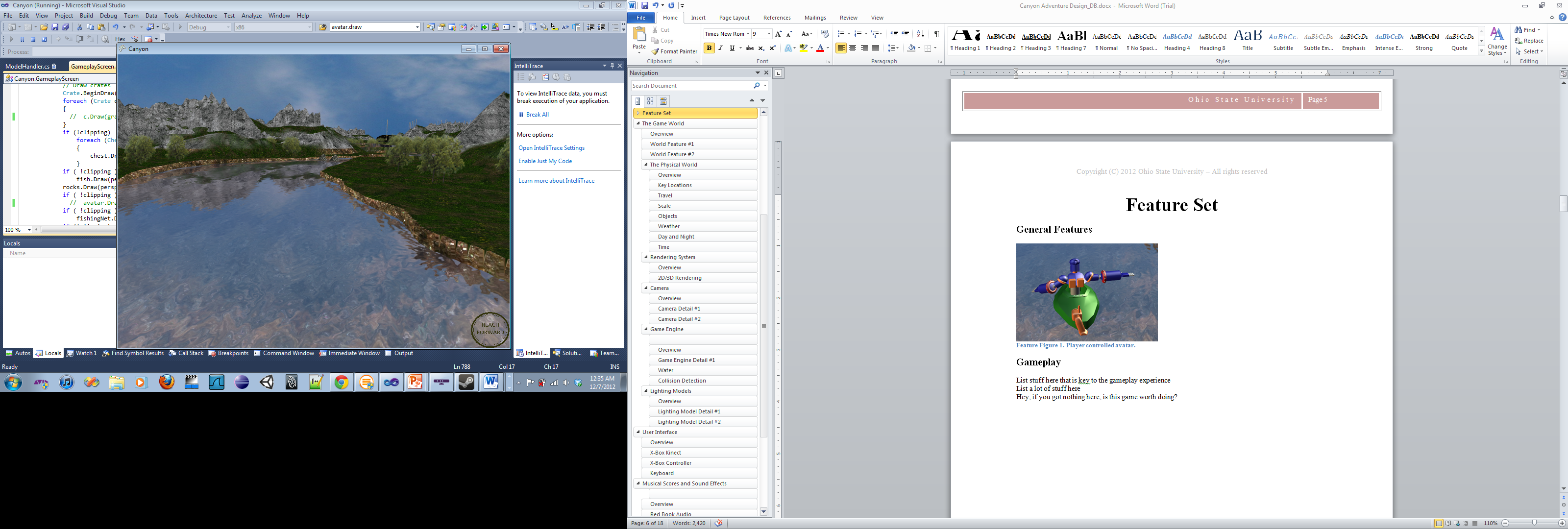
## Victory Conditions

Currently there are no victory conditions. The game is built like a multiple lap race. The faster the patient can navigate the river and cave the better they’re doing. To make the game more than just racing, the amount of fish caught and amount of treasure chests collected are also tracked.

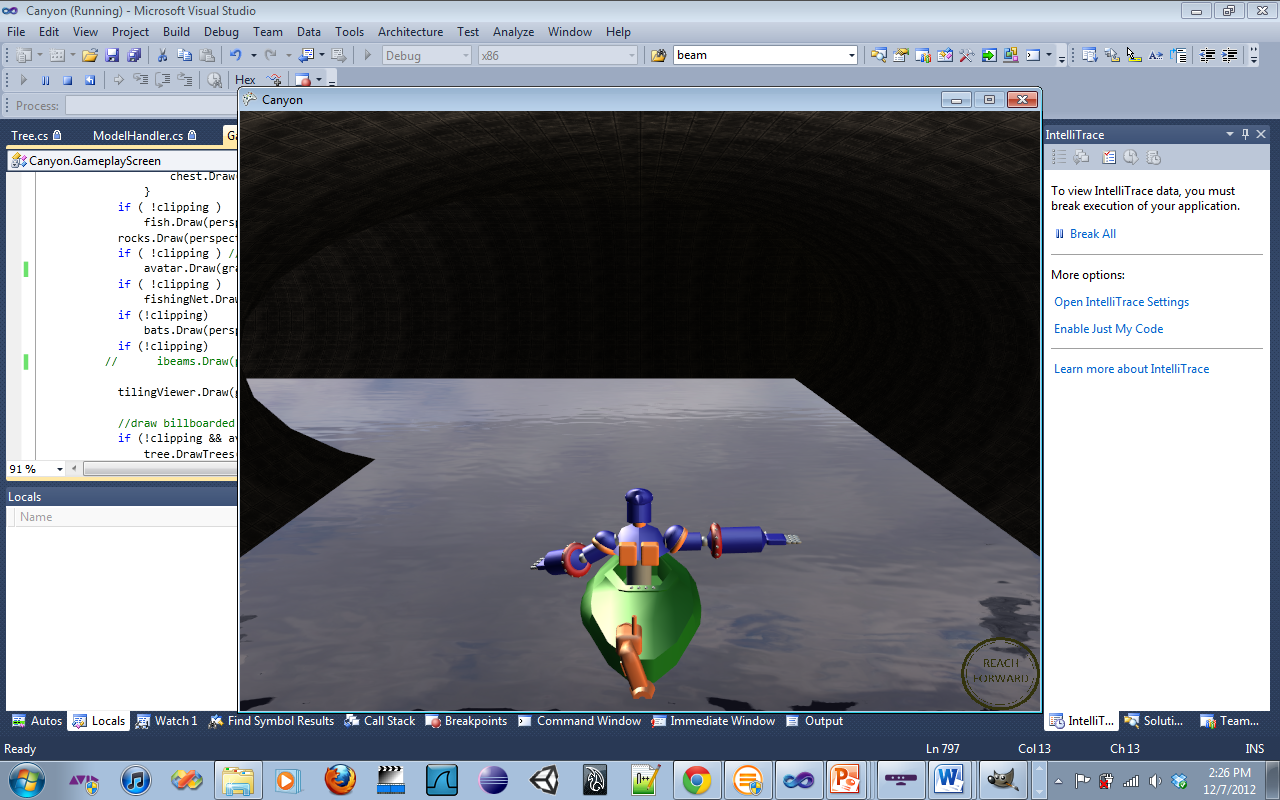
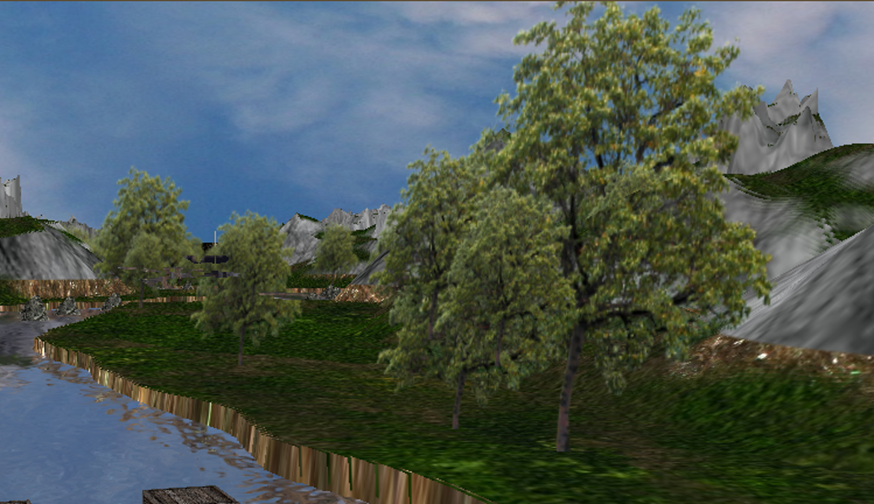
# Feature Set

## Landscape

The game alternates between two environments, an outdoor canyon, and an indoor maze. The majority of the graphical work is involved with the procedural generation of the canyon landscape. All of the landscape is based around the curve of the river. Blended textures and bump maps are also used on the rocks, crates, and ground. A shader and bump map are used for the rippling effects of the water. Procedurally placed trees (as seen in Figure 2) are given a random size and location along the riverbed to give the environment a more natural feel. A skybox is also used for the outdoor region.



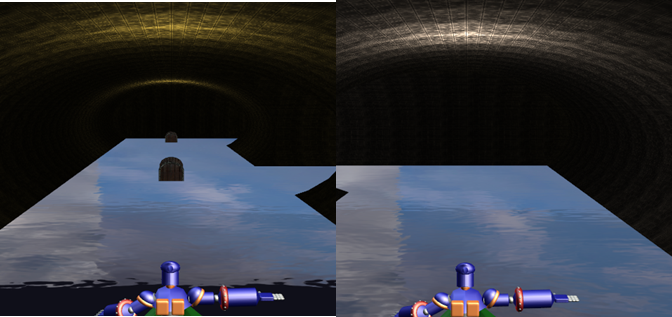
Feature Figure 1. Outdoor Landscape



Feature Figure 2. Randomly sized trees.

Feature Figure 3. Entering the cave.

For the indoor maze, the river continues, but follows a maze pattern, rather than the trigonometric curves of the canyon region. The cave is separated into a series of detours and a single path that leads to the exit. Figure 4 shows how the detoured paths are indicated by dimmer lights. Ideally players stay on the correct path to exit the cave quickly, and get to the next canyon sequence. However, collectibles such as treasure chests may also appear in detoured areas. Collectable items and landscape navigation will be covered in the next section about gameplay features.

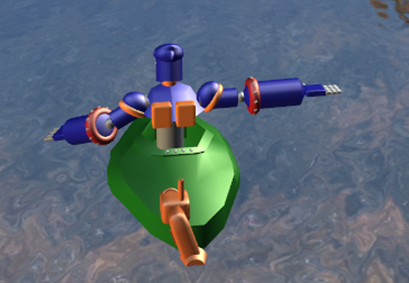


Feature Figure 4. Detoured path lighting (left) vs. correct path lighting (right).

## Gameplay

Gameplay can be divided into three parts: navigation, obstacle evasion, and item collection. Each of these aspects will contribute to the overall score, which is displayed in the upper left corner of the screen (Figure 5). The goal of the game is to navigate the river efficiently to finish with a low time. Bonuses to the score are included with gathering collectible fish and treasure chests. Obstacles will slow the player down, adding to the overall time penalty.

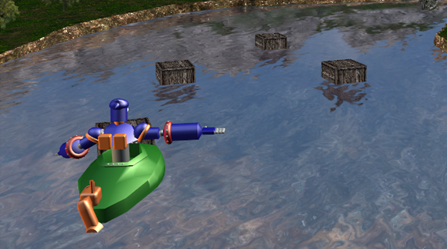
Feature Figure 5. Scorebox displaying elapsed time and collectables.



Feature Figure 6. Player avatar

**Navigation**

Basic navigation involves moving the player forward, and side-to-side in the river. The player is represented by the avatar in the boat, as shown in Figure 6. The avatar’s basic movements follow the player’s tracked motions by the Microsoft Kinect. For more detail on how this works, view the [Gesture Recognition](#_Gesture_Recognition) section. In default moments of the game, the player moves the avatar forward manually to progress down the river. However, in select areas the player will hit rapids where the water automatically moves the player forward. This will be covered more in the description of obstacles. Inside of the maze, the player navigates based on which of three lanes they occupy. A middle lane will always go straight or stop at an intersection. Residing in the left or right lane will turn the player left or right respectively at an intersection.



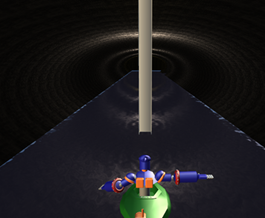
Feature Figure 7. Crates

**Obstacles**

Crates: Floating crates reside in regular portions of the river where the player moves the avatar forward manually. Crates are enabled with colliders that will halt the players forward motion if hit.

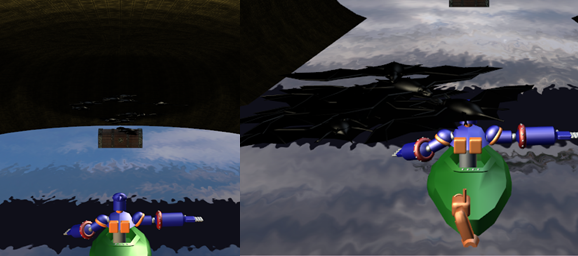
Rocks: Rocks are found in the rapids of the river. The player must sway left or right to dodge rocks to avoid being stopped. Like crates, rocks have colliders that will halt a player until they move aside.

Feature Figure 8. Rocks in river, and rock model with bump map.



Beams: Hanging beams block the paths in the maze segment of the game. The player must dodge the beams by moving his/her head. If the head collides with a beam, a time penalty is induced.

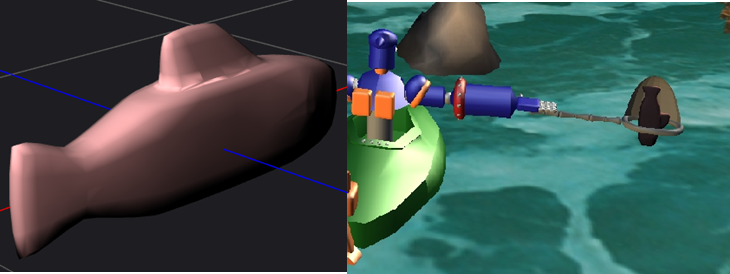
Feature Figure 9. Hanging beam.



Bats: At each dead-end in the maze, a flock of bats resides. Bats will fly around their dead-end until the player enters. When this happens they will attack the player as a time penalty.

Feature Figure 10. Dead end with flocking bats.

**Collectables:**

Fish: Fish reside in specific parts of the river. These will be indicated by a dialog box and appearance of leaping fish in front of the avatar. Fish can be caught by selecting the players net and attempting to scoop fish out of the water. Success is randomized by each attempt, so the more attempted scoops, the better a players chances are of catching fish.

Feature Figure 11. Fish model, and image of fish in net.

Treasure chests: Treausure chests appear randomly in detoured routes of the maze. If the player collides with one he/she will collect it. However, the player must be careful not to waste too much time gathering chests, rather than staying on the correct path.

Feature Figure 12. Treasure chest in detoured path, and original model.

# The Game World

## Overview

The game world takes place in a river valley, with the addition of a maze. The player’s goal in this game is to “get home”. Progress towards this goal is timed. The player, of course, will never actually get home, but only make progress towards this goal. The gameplay cycles through the river valley and maze stages repeatedly, with the player’s score being a combination of time and number of items collected.

## Canyon

The canyon portion of the game is a winding river with the player inside a boat in the river. The player must navigate downstream using gestures and navigate around obstacles such as crates and rocks.

## Maze

The maze portion of the game provides an underground terrain with a single entrance and a single exit. There are several dead-end paths within the maze. The direct path to the exit is slightly better illuminated than the other paths, to give the player a hint to the most direct route. Straying from the direct path has its rewards, including chests which can be collected.

## The Physical World

### Key Locations

The canyon section of the world will have two special sections: rapids and fishing regions. In the rapids, the player will move forward automatically and must turn their shoulders to avoid obstacles. In the fishing section, the player may elect to fish using fishing gestures and collect fish for points.

The maze section of the world will contain dead ends. Dead ends will contain treasure chests to be collected, or bats, or both. Bats have the opportunity to decrease the player score by dive bombing and harassing the player. Bats also can be swatted to increase the player score.

### Travel

The player is confined to the boat and must always navigate in a waterway.

### Scale

1 unit is about 1 meter. The canyon run is about 100 meters and the maze is 55 meters square. This terrain is continuous. See Procedural Content.

### Objects

Crates – Crates are obstacles to be avoided.

Rocks—Rocks in the river are obstacles to be avoided.

Fish – Fish are collectible objects using a net.

Beams – Beams are obstacles in the maze to be avoided.

Signs – Signs will give the user information about navigation.

Bats – Bats can be interacted with and can either hinder or help the player. See above.

Chests – Chests are collectibles that improve the player’s score.

### Time

The player’s progress is tracked in real time. The faster the player completes the canyon and maze, the better the score.

## Rendering System

The game is designed using Microsoft XNA in C#. Graphics are provided using Microsoft XNA libraries with DirectX 9.0c shaders.

## Camera

The camera will be a 3D perspective camera that follows the player in an over-the-shoulder view.

## Lighting

Canyon lighting will be provided using a simple directional light to simulate and outdoor daytime setting. The maze will be lit by point lights near the ceiling of the tunnels. These point lights will be distributed evenly. The intensity of the light will give a hint as to the correct path in the maze.

# User Interface

## Overview

The user can interact with the game either through a X-Box Kinect or a keyboard. Using one of these two they can play the game. Using the keyboard they can raise or lower the Kinect as well as change which hand is their primary hand.

## Kinect

* Using the X-Box Kinect when various gestures are completed associated actions with those gestures are trigger. To read on how these gestures are recognized, read section: “Gesture Recognition”.
* Gestures:

|  |  |
| --- | --- |
| **Gesture name** | **Action** |
| Analog | Tracks hand during Tool selection and when answering a Question Screen |
| ArmToSide | N/A |
| BendDown | Open Toolbox |
| BilateralShoulderFlexion | N/A |
| BilateralWave | N/A |
| ReachForward | Move avatar forward |
| ReachScoop | Go fishing |
| ReachUp | N/A |
| ReachUpAndForward | N/A |
| Rowing | Move avatar forward |
| Stirring | N/A |
| Stop | N/A |
| Wave | N/A |

## Keyboard

Another way to control the avatar is through a keyboard. This was developed with mainly debugging in mind. Hitting certain keys will trigger certain actions.

|  |  |
| --- | --- |
| **Key** | **Action** |
| Up Arrow | Move avatar forward/ Cycle through options |
| Right Arrow | Move avatar to right |
| Left Arrow | Move avatar to left |
| Down Arrow | Cycle through options |
| Enter | Select highlighted menu option |
| E | Open ToolBox |

# 

# Sound Effects

## Overview

Sound has two different uses in this game. The first are sound effects, to liven up a game certain sounds are triggered based on user interaction. This ranges from moving and selecting options from any of the menus to collecting fish and treasure chests. The second serves as background noise. There will be three different sets of background music. The first will be for the menu screens, the second will serve as filler while on the canyon, and the last is for the cave.

## XACT

Along with the XNA framework, an audio program called XACT is also imported. This allows for simple sound integration. By added the sound files to the sound banks we can give them their own specific cues. Upon fully making the banks, it is possible to access all sound files from their cues by importing a single XACT project into the game.

## Sound Design

Since the game is based entirely on a body of water, the idea was to follow that with matching music and sound effects. The Canyon Adventure is meant to be a therapy game so all sounds needed to be calming and light to the ears. The patients have been through enough and are on the road to healing, the game should only help that.

# Character Rendering

The goal is to provide an avatar which mimics the player’s positions using an Xbox Kinect. Since the players will be seated for safety, below the waist is not being modeled. The avatar will show the player in a boat.

The Kinect provides position and rotation information for each joint and bone in the body. To mimic the user’s position, the absolute rotation of each bone will be applied to each bone in the avatar, regardless of position. This poses the avatar in the same pose as the player.

# Procedural Content

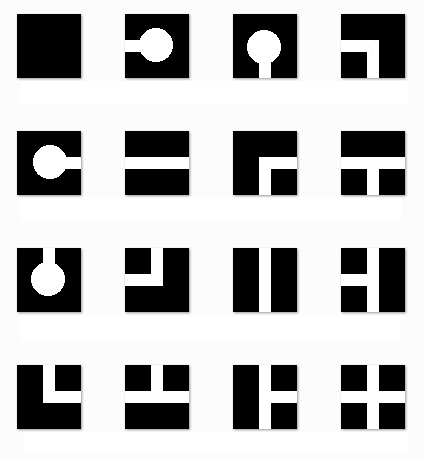
All content in this game will be generated procedurally.

The canyon consists of a river of random width cut through a valley. The heightmap of the valley will be generated using Perlin Noise and Fractal Noise from the Libnoise library. The width of the river and valley floor will also be determined using Perlin Noise.

Once the heightmap of the terrain has been determined, it is skewed in the z direction to generate curves in the river. The equation controlling the skewing of the heightmap is:

Where A, B, C, and D are constants.

The maze is a randomly generated maze. First the maze will be randomly generated by choosing from the following set of initial tiles:

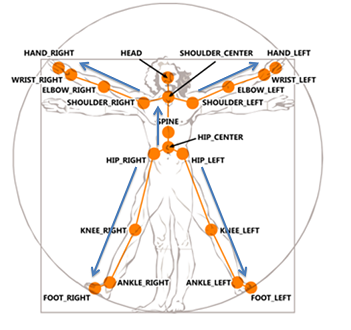


Then the disconnected components of the maze are joined together to create a complete maze. Finally, the entrance and exit to the maze are added.

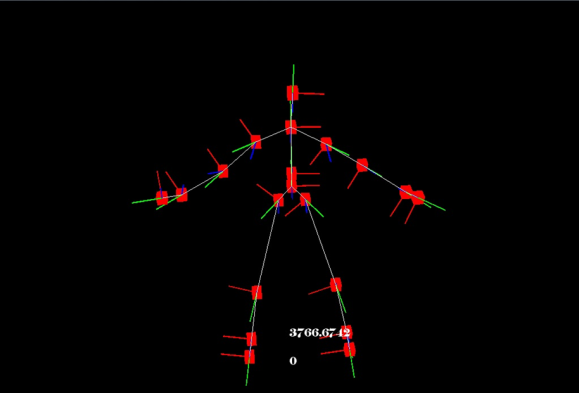
# Gesture Recognition

For this game gesture recognition is the backbone of the gameplay mechanics. The primary objective of this game is to encourage physical patients and other players to perform the motions intuitively and routinely. To achieve this, a major portion of development time was devoted to developing accurate gesture states and scalable parameters.

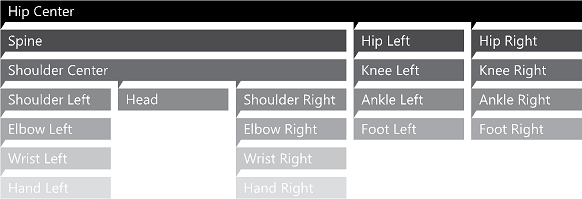
**Kinect:**

To capture motion data, the Microsoft Kinect was used. By integrating the Kinect SDK, we were able to track the user’s skeleton in a 3D space with 20 joints. View Figures 1 and 2 to see the theoretical and actual data of the skeleton.. Refer to Figure 3 for the hierarchy the joints follow. For majority of the gestures, “Spine” and all of its children (shoulders and up were used). However, the entire skeleton is tracked in case future versions require bones from “Hip Center” and lower to be captured.

Gesture Figure 1. Kinect Skeleton Map - msdn.microsoft.com



Gesture Figure 2. Kinect vector skeleton from SDK viewer.

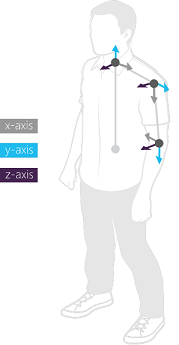


Gesture Figure 3. Kinect Skeletal Hierarchy - msdn.microsoft.com

**Coordinates Followed:**

Most of the implemented gestures are tracked through a combination of the Cartesian and the polar coordinate system. This process is done through several routine steps:

1. Each joint is converted into a 3D vector.
2. Each vector is then flattened along its respective plane. For example, if we want to track the forward extension of the arm, we will take the vector of the forearm and flatten it along the YZ plane. Use Figure 4 as a reference of this orientation.
3. Each plane is normalized, and a dot product is taken between the two connected vectors (such as forearm and upper arm).
4. The cosine is calculated from the dot product (in this case for the elbow angle).

**Gesture States:**

Gesture Figure 4. Coordinate System of Neck, Shoulder, and Elbow -msdn.microsoft.com

Each gesture follows a cycle of states. Each gesture has an initial state, middle state(s), and an execution state. Multiple middle states are used in some gestures to ensure precision of movement.

For instance, in a reach forward gesture the states are as follow:

1. Waiting - Initial state. The YZ angle of the elbow is at the lowest point, essentially for the arm resting.
2. Neutral - First motion. The elbow begins to close and rise.
3. Reaching1 - Follow through. The elbow is raised the most and begins to extend out.
4. Reaching 2 - Execution. The elbow is still raised and fully extended. The triggered player action (such as moving forward) is executed and the state returns to the initial point.

**Feedback:**

In order for players to understand gestures properly feedback icons are used. These icons are based on the gesture states, and will become highlighted as the player gets closer to executing a gesture. Figure 5 is a shot from the game feedback icons as the player goes through the 4 states of the reach forward motion.



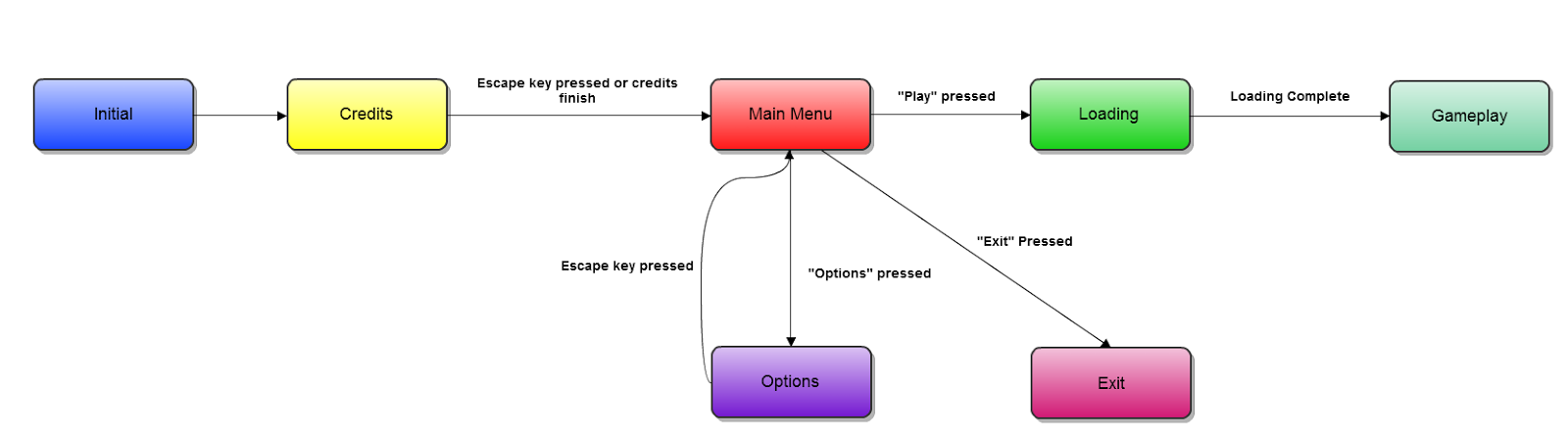
Gesture Figure 5. Reach Forward Icons from game.

**Future of gestures:**

The games gestures are fully implemented for the gameplay functions. However, the games purpose as a physical therapy tool is an evolving process. For this reason the parameters need to be adjustable so that the game is more or less forgiving on the accuracy of each gesture. To help scale this, each of the gestures required coordinates are parameterized variables. These variables can be modified by an XML parser to take the gestures states, joints and angle parameters. This allows the game to be more dynamic in the long run.

# Game State

**Game State:**



**Game State Description:**

Credits -> Main Menu : Escape/Back button pressed or credits finish

Main Menu -> Credits : “Credits” selected

Main Menu -> Options : “Options” selected

Main Menu -> Loading : “Play” selected

Loading -> Gameplay : Loading assets complete

Gameplay -> Main Menu : Escape/Back button pressed

# Class Diagram

Diagram of how the different screens work and how they are managed

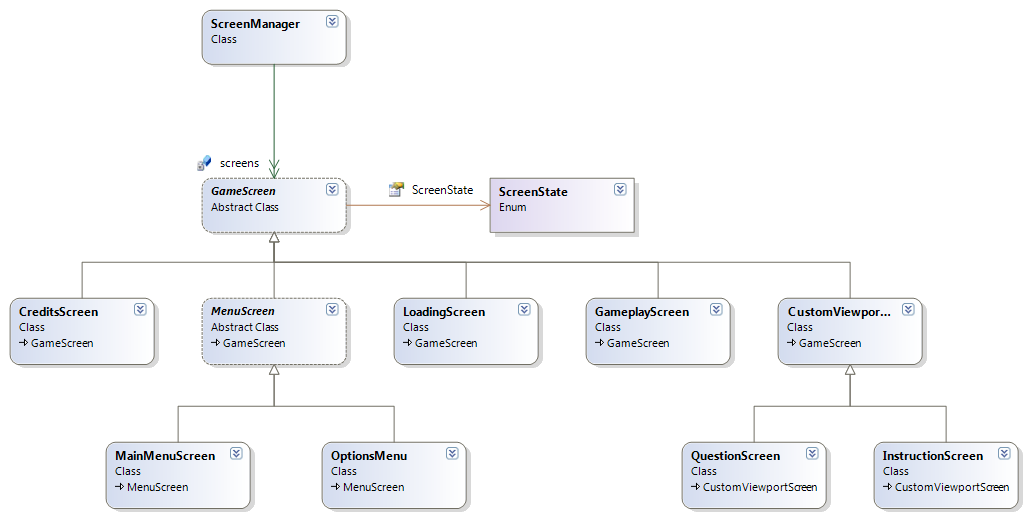


Diagram of how gestures work using the rowing gesture as an example

