Software Requirements Specification

for

Kinects in Unity

Version 2.0

Prepared by

Group Name: Team Hex Pistols

Jerry Boyang Peng	4467056	jerry.boyang.peng@gmail.com
Sea Pong	4311197	sea@seapong.com
Alex Scarlett	4346581	scarlett.alex@gmail.com
Anthony Narsi	4171930	ajnarsi01@gmail.com
Kevin Sheridan	4160917	razor95kds@gmail.com

Instructor: Chandra Krintz

Course: CS Capstone 2013 - CS189A

Lab Section: *Friday 12pm*

Teaching Assistant: *Stratos Dimopoulos*

Date: Thursday March 6, 2013

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Jerry Peng, Sea Pong, Alex Scarlett, Anthony Narsi, Kevin Sheridan	Initial Version	02/11/13
2.0	Jerry Peng, Sea Pong, Alex Scarlett, Anthony Narsi, Kevin Sheridan	Revised Version	03/07/13

1 Introduction

The Multi-Kinect 3D Scene Visualizer integrates Microsoft® Kinect™ device(s) and the Unity 3D software to track a user's head rotation and position in order to naturalize the viewing experience similar to mimicking a real window with a large-format digital display. This system serves as an augmented reality workspace for the user.

1.1 Document Purpose

The purpose of this report is to fully document the project that will be implemented by The Hex Pistols. It will serve as an agreement between the design team and the sponsor, JPL, as to what the product features will be and how a finish product will be defined. Once this document is agreed upon by both parties, the design team will begin working on the product. When all features are implemented as stated in this document, the product will be completed.

If there is any disagreement between the design team and sponsor over the end product, this report will serve as a mediating document. All features and specifications listed on this report will be implemented by the design team. No changes in workflow will occur unless this document is modified and again agreed upon by both parties. After the product is finished, features may be added but must be added to this document and again agreed upon by both parties.

1.2 Product Scope

This project will create an integrated system of multiple Microsoft Kinect sensors and large format displays to create a naturalized viewing of 3D panoramas on non-panoramic screens, similar to looking outside a 'virtual' window. NASA's astrophysicists are looking forward to the day where they can simply enter a room, flip a switch, and be surrounded by a previously unreachable territory like Mars.
Kinects in Unity is setting the basis for these scientists to be virtually immersed in a complex augmented reality, just as any other scientist could simply go to a lab or field and perform experiments. The main intention of this project is to allow a user to realistically experience an environment that he or she could not otherwise be physically present in. The Human-Computer Interaction Department at JPL wanted to simplify the process of tracking movements by cameras and transferring these movements to an augmented reality. Simplifying the process entails decreasing the cost of the system as a whole. NASA is working with Microsoft Kinects to prove to astrophysicists that entering a virtual environment does not require highly expensive camera and tracking equipment. Our project provides the foundation for this significant benefit.

1.3 Intended Audience and Document Overview

This document is intended for the professors of CS189A and CS189B, as well as our sponsor JPL. The SRS will discuss the details and implementation of the project. It is recommended for those familiar with computer science to start in section 2, while those unfamiliar should start in section 1.4 to understand some of the terms used throughout the document. People who are reading this just for a product overview should only read section 2, while those seeking details about the implementation and how to use the product should continue onto section 3.

1.4 Definitions, Acronyms and Abbreviations

- *3D* – Three Dimensional
- *API* – See *Application Programming Interface*

- *Application Programming Interface* – A set of source code with a protocol intended for software packages to communicate with each other. It may also contain documentation that describes its code structure.

- *Attribute* – A structure item of a class. Equivalent to *member variable*, *property*, *data member*, or *field*.
- *Boolean* – a 1-bit variable type that is either 1 (true) or 0 (false)
- *FPS* – Frames Per Second
- *IDE* – See *Integrated Development Environment*
- *Integrated Development Environment* – An application tool suite which assists in developing code
- *JPL* – Jet Propulsion Laboratory
- *Kinect SDK* – The libraries provided by Microsoft to communicate and effectively use data received from the Kinect
- *Method* – A behavior item of a class. It is the standard naming of a function that is defined within a class and is essentially a member of it. A method has all access to private and public data members of its class.
- *NASA* – National Aeronautics and Space Administration
- *SDK* – Software Development Kit
- *Singleton Class* – A class that is instantiated once and only once at program startup. Constructors are of private type to prevent the class from being instantiated again elsewhere. No member function should take advantage of this private constructor and call it any time throughout the program. The initial instance is of private static type and a public static getter function is instantiated for public access to this singleton class instance.
- *SRS* – Software Requirement Specification
- *UCSB* – University of California at Santa Barbara
- *UML* – See *Universal Modeling Language*
- *Unity 3D* – the rendering engine and IDE where the panoramic scene is generated and visualized. See *IDE*
- *Universal Modeling Language* – standardized visual modeling language used for representing object-oriented programming.
- *Studio* – An IDE created by Microsoft used for developing Windows applications
- *Workspace* – The area in which the user stands

1.5 Document Conventions

Bold Text - Signifies important content or keywords

1.6 References and Acknowledgments

- Kinect SDK Documentation (http://www.microsoft.com/en-us/kinectforwindows/develop/resources.aspx)

- Unity 3D API Documentation (http://unity3d.com/company/support/documentation/)

2 Overall Description

2.1 Product Perspective

This product was a concept conceived by JPL. The main intention of this product is to allow a user to realistically experience an environment that he or she could not otherwise be physically present in. An example use of this product could be a scientist that is observing a panoramic environment captured by the Mars Rover to fully experience the landscape. This is a self-contained product, and operates as stand alone system. Currently, JPL has the system working with highly expensive camera and tracking equipment, but this system is not ideal for reproduction and portability. They proposed this opportunity to us in order to set the stage for a scientist to be virtually immersed in a 360 degree augmented environment with the simplicity and inexpensiveness of Microsoft Kinects.

2.2 Product Functionality

- Microsoft Kinects track the user's location
- The screens display the augmented reality as an integrated panoramic viewing window.
- The augmented environment's viewable segment will be based on the perspective of the user according to the user's location.
- The user's location is calculated based on the distances and angles in reference to the Microsoft Kinects.
- The Kinects are mounted on the screens.
- The system is scalable, meaning multiple Kinects/Screens can be added and the system will remain functional and stable.

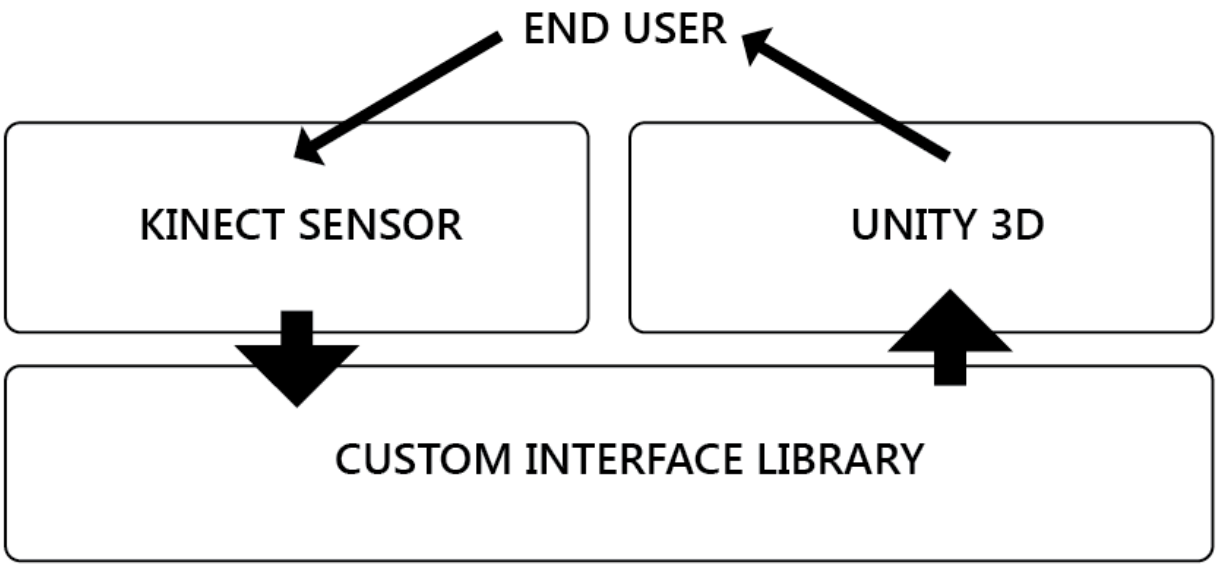


Figure 1. Data Flow Diagram

2.3 Users and Characteristics

The primary users of this product are going to be academics in the fields of science. The users are most likely to have advanced degree in his or her respective fields.

2.4 Operating Environment

- Windows 7 or 8
- Intel x86 architecture
- The system is built on top of the Windows operating system. Windows 7 or 8 is required for the Microsoft Kinect SDK. The libraries in the SDK are compiled for an Intel x86 architecture thus an x86 chipset is needed.

2.5 Design and Implementation Constraints

- Kinect
  - Hardware
    - 32-bit (x86) or 64-bit (x64) processors
    - Dual-core, 2.66-GHz or faster processor
    - USB 2.0 bus dedicated to the Kinect
    - 2 GB of RAM
    - Graphics card that supports DirectX 9.0c

- Software
  - Operating System
    - Windows 7
    - Windows 8
  - Unity 3D
    - Windows: XP SP2 or later; Mac OS X: Intel CPU & "Snow Leopard" 10.6 or later. Note that Unity was not tested on server versions of Windows and OS X.
    - Graphics card with DirectX 9 level (shader model 2.0) capabilities. Any card made since 2004 should work.
    - Using Occlusion Culling requires GPU with Occlusion Query support (some Intel GPUs do not support that).

2.6 User Documentation

- Developers
  - Detailed list of APIs
  - System configuration Instructions
- Users
  - Easy to follow setup directions
  - List of system features
  - User's manual

2.7 Assumptions and Dependencies

- User has a large enough workspace area to accommodate the usage of a Kinect
- User has a computer system running a recent iteration of the Windows Operating System
- The user has a powerful enough computer system that can handle the additional processing associated with introducing additional Kinects into the system

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The Multi-Kinect 3D Scene Visualizer utilizes minimal human interaction. Since it invisibly tracks the user's head motion, there is literally no user input required except for the initial calibration/training sequence.

Figure 2. Multi-Kinect Setup

3.1.2 Hardware Interfaces

This software requires heavy hardware integration. The data retrieved from the Kinect(s) includes a color video stream and an infrared depth field stream. The audio stream will not be used in this project. After collecting the data from the Kinects the final processed motion is sent to Unity3D which outputs a digital video stream to a large-format display device.

The hardware setup is very simple. To use a dual monitor setup with one computer, an additional device is needed to split the computer's VGA output into 2 separate VGA outputs. The device recommended to use is called the DualHead2Go. Simply connect VGA cables to the TVs and

attached them to the DualHead2Go. Then attach a VGA cable from the computer to DualHead2Go. The Kinects should also attach to the same computer through USB. If the computer does not have 2 free USB ports, a USB hub should be used. It is recommended that the Kinects be directly attached to the computer while other USB devices be attached to the USB hub.

VGA-Laptop-Display Connection

Laptop-Kinect-Power Connection

3.1.3 Software Interfaces

Since the Kinect SDK is designed for Windows, most of the development will be on a Windows based machine. The Kinect SDK will take care of retrieving the Kinect data and outputting two data streams per Kinect, the video and depth streams. It is up to our custom interface library to aggregate the data and create a final scene control stream that will be fed into the Unity 3D interface. From there, Unity will take care of the camera control.

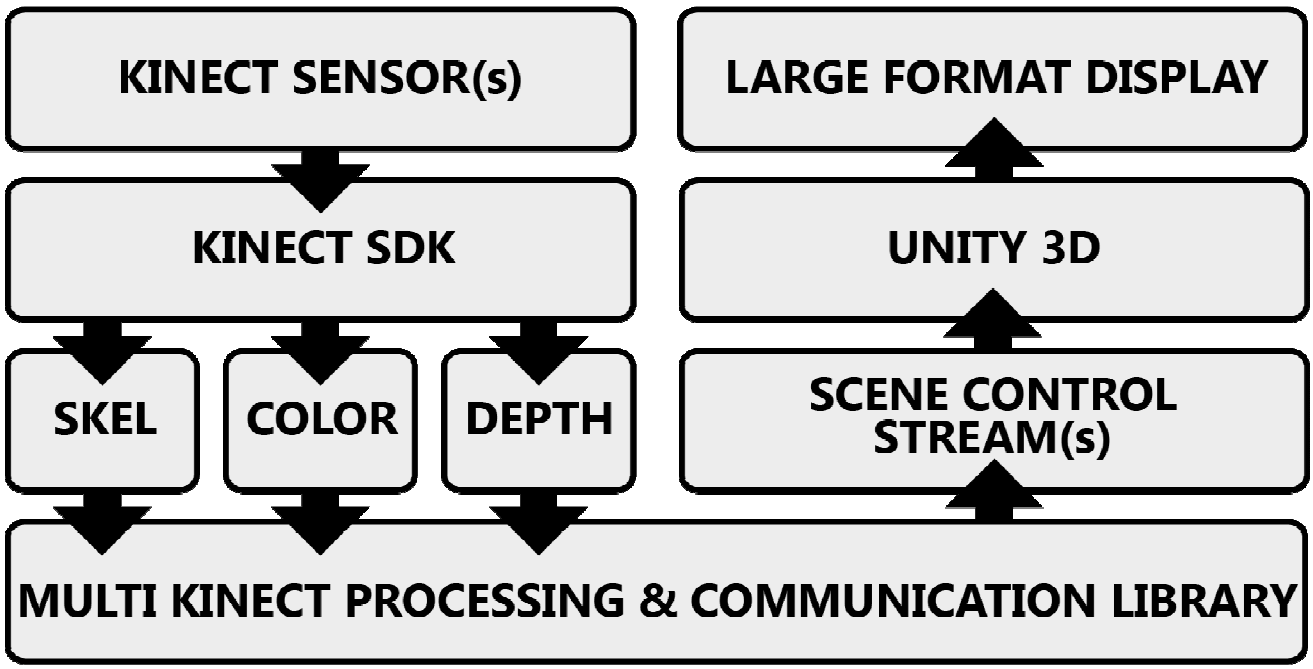
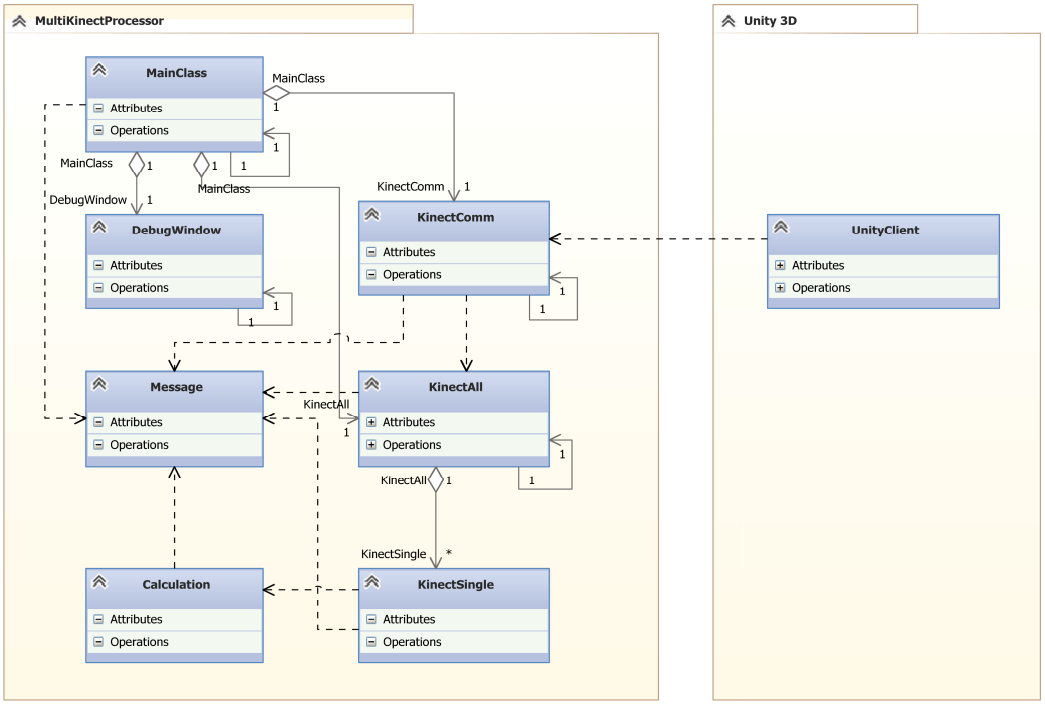
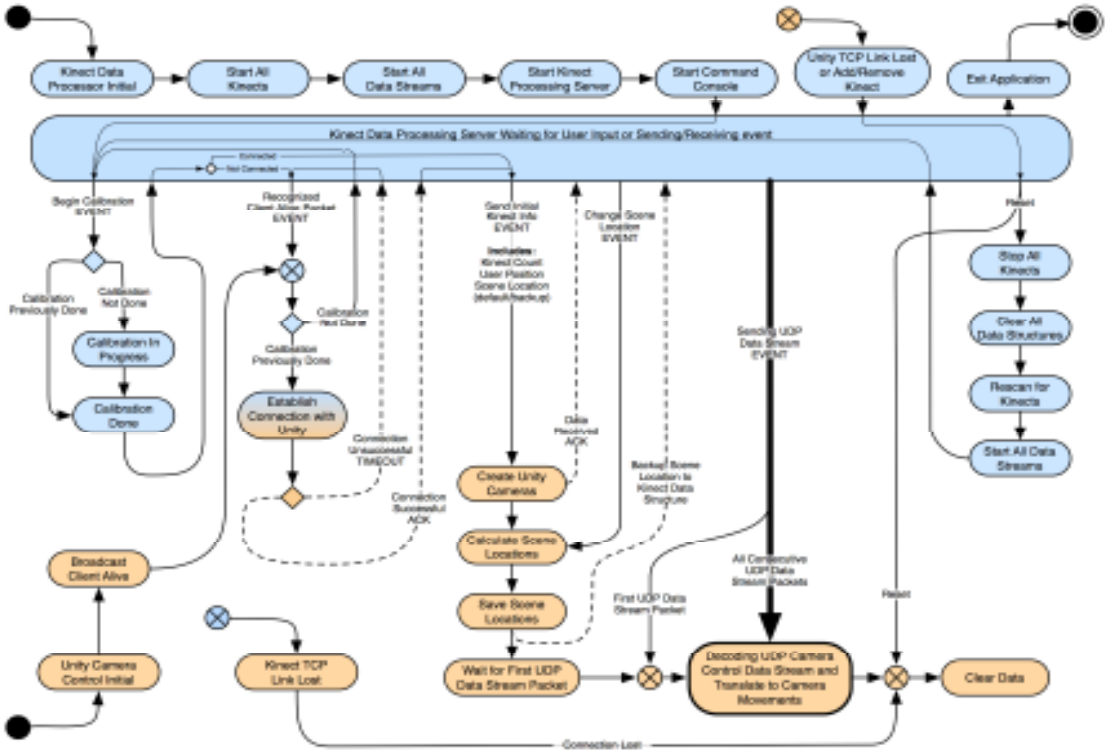


Figure 3. Software Data Flow

3.1.4 Software Data Flow Diagram



3.1.5 Software State Diagram



3.1.6 Hardware Communications Interfaces

- (multiple) USB is used to communicate between Kinect and PC
- (multiple) HDMI or VGA is used to send data to the display from the PC

3.2 Functional Requirements

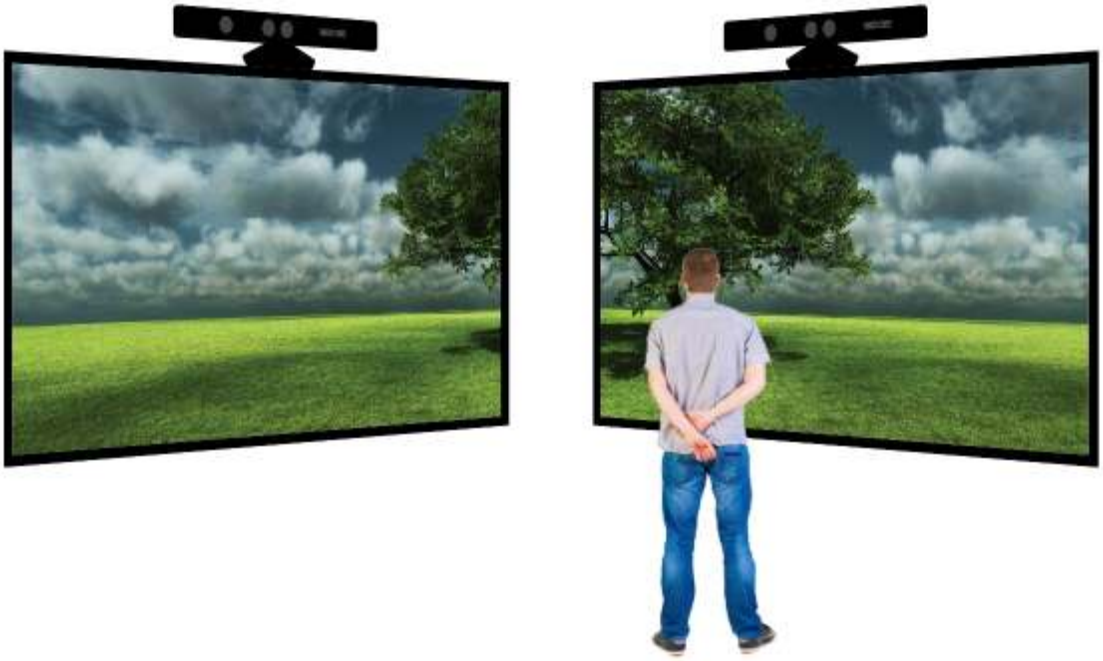


Figure 4. User standing on right. Sees behind corner on right display

Figure 5. User standing on left. Sees behind corner on left display

3.3 Behaviour Requirements

3.3.1 Use Case View

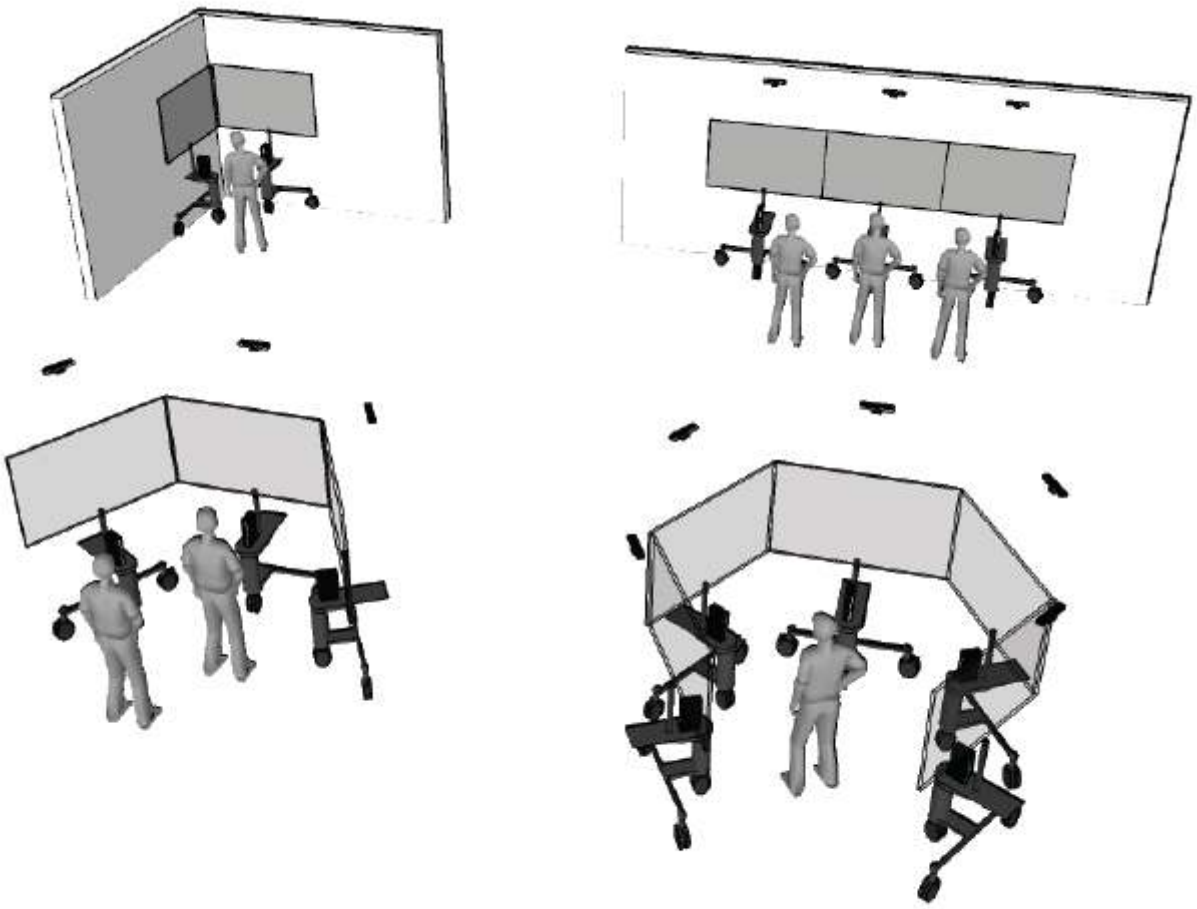


Figure 6: CAVE-like (inside-facing-out) arrangements of Choir

4 Other Non-functional Requirements

4.1 Performance Requirements

- The user is to stand between 2 and 8 feet from each of the Kinect
- The system should run in Real-time
- Should be able to track only 1 person in the Workspace
- Should be able to scale up to multiple Kinect-screen pairs
- Should be able to run on commercially available computing platforms
- Stretch goals include increasing the number of Kinects for a wider range of view and increasing the number of users present in the workspace.

4.2 Safety and Security Requirements

Safety

- The workspace is to have no obstacles or obstructions in the way of the user.
- Elongated continuous usage of the system may be harmful, so it is highly recommended that two hour continued use sessions are followed by a fifteen minute break with the system powered off. Also, the system should not be used for more than four sequential two-hour sessions (total of 9 hours with 15 minute breaks).
- A very small percentage of people may experience a seizure when exposed to certain visual images, including flashing lights or patterns that may appear in video games. Even people who have no history of seizures or epilepsy may have an undiagnosed condition that can cause these “photosensitive epileptic seizures” while observing the displays.

Security

- While the presentation of our product will be open to all viewers, the source code and project implementation is to stay closed.

4.3 Software Quality Attributes

- **Robustness** - The system needs to be robust enough generate a realistic and accurate environment based on the locations of the kinect/screen pairs. The system should also be able to dynamically accommodate additional kinect/screen pairs so that any reasonable number of Kinects and screen pairs can be added and placed at arbitrary positions around the user and a correct and accurate environment can still be generated.
- **Reliability** - The system should able to continuously run for a long duration of time (mutliple hours) and not suffer from system slowdowns or crashes caused from memory leaks and zombie process.
- **Portability** - The software should be able to run on any Microsoft windows based platform. To set up and tear down the entire system, displays need to be set in place, Kinects need to be placed on the displays, Kinects need to be connected to the computer(s), and the program start button is pressed. Ideal set up/tear down time should be approximately five minutes.

- **Ease of use** - Someone with little to none technical experience in the operations of electronics should be able setup and use this system by following a simple set of instructions
- **Ease of Learning** - The learning curve for this software should be short since the software should perform the corresponding tasks based on natural human motions.