VI-Char

Augmented Reality Division Software Requirements Specifications

Matt Burke, Thomas Freeman, David Greene, Erin Jamroz, and Selah Ross

September 14th, 2012

Introduction

Vi-Char is a small startup that is developing platformer a game in which the main character will be controlled through virtual puppeteering. Additionally, other players will be able to view and interact with the game using augmented reality on mobile devices. *Vi-Char* has five divisions that will be collaborating on this project. This document was prepared by the Augmented Reality (AR) Division. It is a Software Requirements Specification (SRS) that is intended to accurately describe the the product that the AR Division will produce. It describes what the product will be capable of and how it will interact with components from the other divisions of *Vi-Char.* In the first draft, this document will help sort out the details of roles and responsibilities of each division within *Vi-Char*. Additionally this document will ensure that the client, Joel Ross, understands what the final product will be capable of, and that the AR team will be clear on the functionality that needs to be built into the product.

This five person team will rotate responsibilities as each member participates in all aspects of the development of the final product. All members of the team have significant programming experience with Java. Additionally, each brings a unique set of other skills. Selah Ross wrote the Executive Summary for this report, and she will be the team manager for the first two weeks. David Greene wrote the Application Context section of this report, and he brings significant experience with python, plus some experience with C++, Open GL, Open CV. Erin Jamroz spearheaded the Functional Requirements with input from the whole team. He also brings a strong background in mathematics and mathematical/theory intensive computer science. He has had management experience in both the professional and educational sectors. His strengths lie in designing/mapping a software system as opposed to actually coding one. He can code in python and java competently and is learning C++ right now. He also knows MySQL, PHP, and HTML. Thomas Freeman wrote the Non-Functional Requirements with input from the whole team. Thomas also brings extensive experience with Open GL, C++, and Java to the team. Matt Burke wrote the Introduction, Timeline, Potential Challenges, and Glossary for this Report. He brings the least coding experience of team, but he has a B.S. in Mathematics, and years of experience working in small teams, managing teams, and running team building exercises. He is currently learning Python, and has some 12 year old experience with C++. This team doesn't have any direct experience with Augmented Reality software, but they are ready to learn.

Executive Summary

The members of the *Vi-Char* Project are creating a game in which a puppet controlled by players using a Kinect will navigate through a virtual platformer world that is seen on the screen of an Android smart phone. Within the Vi-Char project, the Augmented Reality Development Team is charged with the task of providing vital position information in order to integrate 3D graphics with a limited space of the real world as seen through the camera portion of the smart phone. The AR Team works with the data provided by 0-4 players controlling the phones, using this input to determine the positioning of the player in terms of the game world. This information is also used to identify key targets, and acquire the user's position in relation to said targets provided that they have the correct application software downloaded and installed. These players will also be able to physically become involved in the actual gameplay beyond positioning coordinates. Due to the multiple users controlling the phones, the virtual environment will be rendered through several angles, providing unique

viewpoints for each individual player. As location information is fed through the smart phone and sent to the webserver, the inputted data will communicate with other divisions of the *Vi-Char* project as needed. The modified data will then be sent back to the Android smart phones as output, providing the user with a new 3D scene integrated with the real world through their screens.

Application Context

Our firm has been tasked with the creation of a game that incorporates somewhat unique input and output methods that utilize both motion capture as well as augmented reality. The game itself will be of a platforming genre, where a motion controlled puppet attempts to navigate levels while being assisted or assaulted by users of mobile devices. Our specific team is tasked with the creation of the augmented reality system, that is to be implemented on these phones.

This game will serve 2-7 users, or players, who will be split into two distinct groups. The first group, consisting of 2-3 players, will interact with the system via motion capture, posing and gesturing while in front of a Microsoft Kinect. The other less vital group, that will consist of 0-4 players, provides the system with input and receives output via Android smart phones. These players will interact with the game world in some varying capacity, and our team will be responsible for handingly a majority of their input and output. These devices will observe a game setting by looking at a specified region, and will be able to rotate and move around this region to attain different viewpoints. The core objective of our team lies in creating a working system that can correctly determine the position of the game world, and work in conjunction with other teams to achieve the full effect of a 3d game world portrayed in ‘real’ space.

In order to anchor the game world into the real world, we need localization. This will be accomplished through the use of a large printed ‘AR target’ that will establish where the game world will be drawn. The goal of this printed target is to provide a basis for our team’s software to determine the android phones position and angle relative to the target. Our team’s software will gather this positional information, and supply it to some aspect of the game engine via the web server for it’s interpretation. Then, with this positional data, the game engine will be able to generate images from these given perspectives that will then be displayed on the mobile devices. This system will provide this functionality for up to four android phones at once, which all will eventually have their respective viewpoint of the game world drawn upon their displays.

Functional Requirements

Primary Features:

- Be able to identify AR targets and our position relative to them
- Be able to send/receive information from the game application
- Be able to render the environment from multiple angles i.e. support multiple users
- Be able to display the virtual platformer world

Primary Use Case: Playing the Game with Multiple Observers

Preconditions:

- The server is up and running
- There is both a team of “actors” and “observers”
- All phones have our application software downloaded and installed
- Motion capture hardware and software are available and running

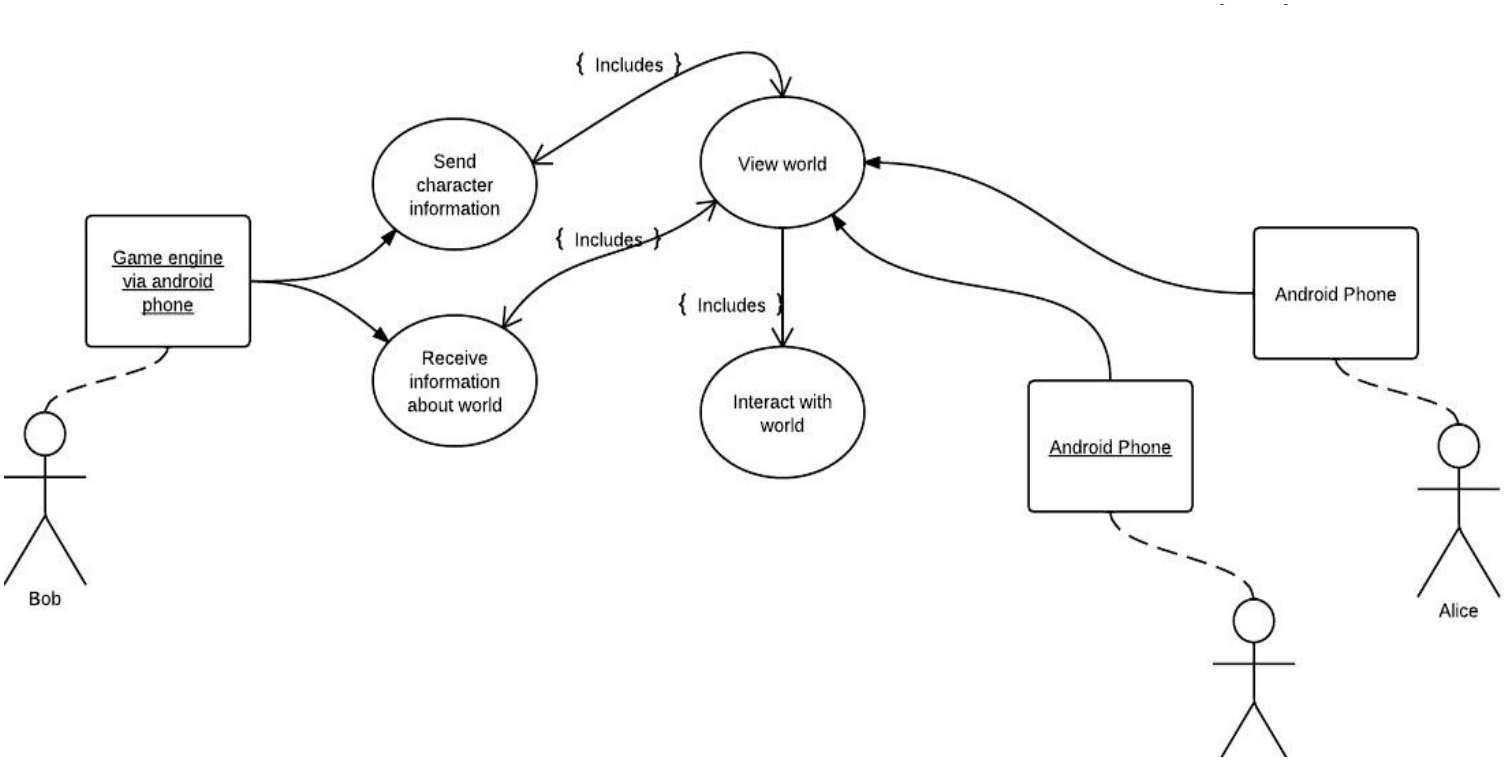
Postconditions:

- Connection with the server has been terminated

Actors:

- Game engine via android phone
- Android phone
- Character controller (Bob)
- Observer (Alice)
- Observer (Dave)

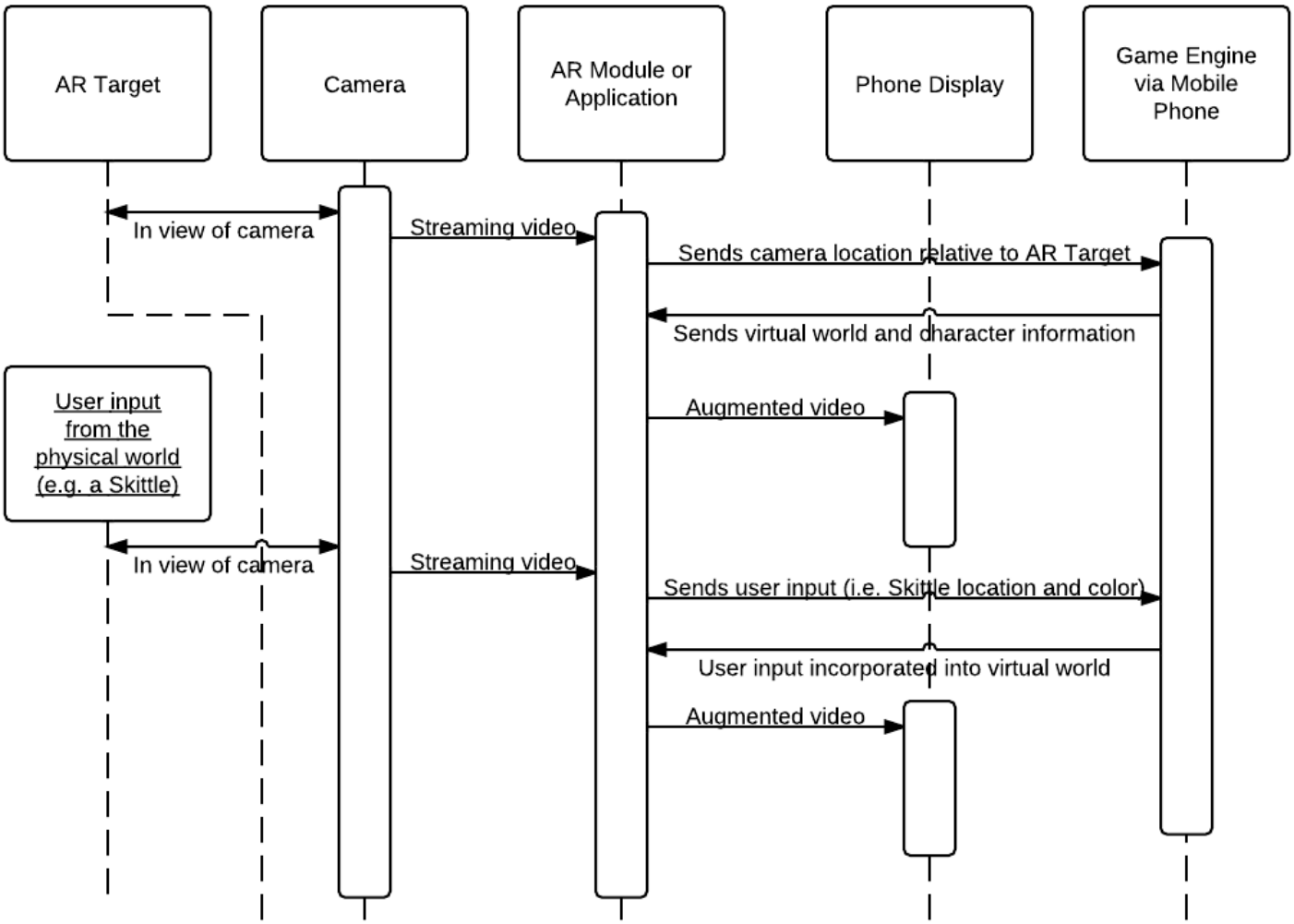
Scenario: Bob, Alice, and Dave want to play the game together


{ Includes }
Send character information
View world
Game engine via android phone
{ Includes }
Receive information about world
{ Includes }
Interact with world
Android Phone
Android Phone
Bob
Alice

Bob, Dave, and Alice decide to play the game together. Dave and Alice will open the application on their android phones, which allows them to view the virtual environment overlayed on the actual world. Bob's virtual character is places in this virtual environment through the motion capture system and transmitted via the network server. Bob is able to control the character in the environment and respond to Dave and Alice's interaction with the world via their phones. Together they go on playing the game until they decide to stop or the fame ends at which point the connection is terminated and all parties go their separate ways.

System Sequence Diagram

Here is a diagram that helps explain the sequence of events in the primary use case:



Nonfunctional Requirements

Performance:

- Memory and Runtime efficient. With the primary target of this program to run on a smartphone which have limited resources it is in the AR groups best interest to strive for the most optimum efficiency.
- Consistent location updates with minimal latency. To ensure an accurate image of the game world with maximized smoothness. Inconsistent or poor latency can produce distracting and or unplayable game conditions.
- AR graphics should remain at playable Frames Per Second. Approximate lower bound for FPS on augmented graphics is 12, this might change at a later date.

Usability:

- It should be easy for users to set up their game environment (i.e. where the targets are in the real world) and lock on to the targets in appropriate environments.

Documentation:

- Clear and understandable documentation. To facilitate intergroup collaboration and understanding of the project code. Effective documentation will additionally assist “border” teams (web server and android) to more appropriately interface their code to ours.

Environmental:

- The Augmentable Zone size will be determined upon at a later date. Targets outside of the AZ will not be drawn. This is it limit the size of the playing area to match the optimization of the AR software.
- Consistent and adequate lighting levels to allow for accurate and quick target recognition. Adequate levels to be determined at a later date. Poor lighting or an inconsistent light level can adversely affect the software's ability to run properly.

Other Requirements:

- Platform compatibility with Android mobile devices.
- Robustness. Be able to deal with improper parameters from external packages without crashing the program.

Timeline

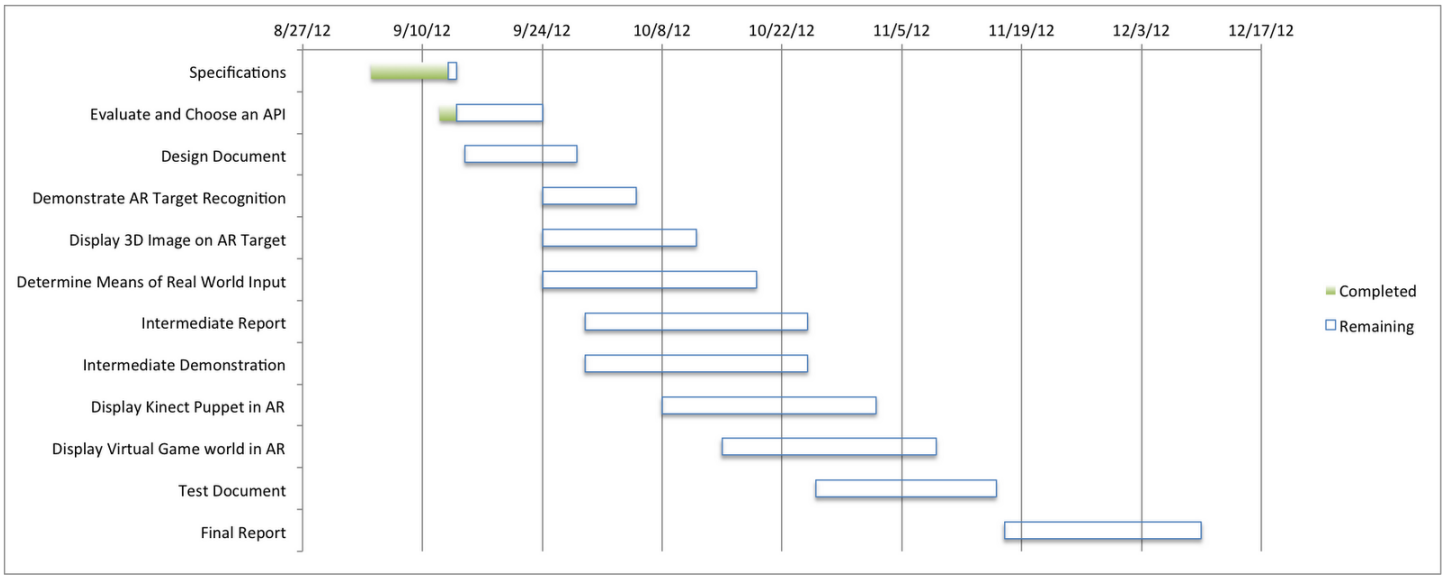
This project has a strict timeline dictated by the client. The final demonstration must be conducted no later than December 10, 2012. Additionally, a number of intermediate reports for the client have strict deadlines:

- Specifications - Due 9/14/2012
- Design Document - Due 9/24/2012
- Intermediate Report and Demonstration - Due 10/25/2012
- Test Document - Due 11/16/2012
- Final Report and Demonstration - Due 12/10/2012

In addition to the deadlines provided by the client, the AR Division has come up with a number of internal deadlines in order to measure progress, and ensure that the final product is delivered on time. These internal deadlines currently include:

- Evaluate and choose an API - by 9/24/2012
- Demonstrate AR target recognition - by 10/5/12
- Display a 3D image on an AR target - by 10/12/2012
- Determine a means or method of input from the real world - by 10/19/2012
- Display the Kinect puppet through AR - by 11/2/2012
- Display virtual game world through AR - by 11/9/2012

This timeline will be updated frequently as the team learns where the challenges lie, and as they discover new tasks to add.



Augmented Reality Division's Timeline

Potential Challenges

- One challenge that this team faces, is that no one on the team has previously worked on an augmented reality project. There are many ways that this could lead to specific challenges. To start with, the AR team needs to be conservative with their timelines as they will be researching, selecting and learning the capabilities of specific AR tools as the project progresses.
- As the team works on the requirements specification phase, it is challenging to know which limitations or possibilities need to be communicated to the other divisions, because the AR team is learning the capabilities of the tools as they go.
- Depending on the size of real world area that the game is to be played on, some unknowns and challenges arise. How big of an AR target is viable, both in terms of ease of printing and in terms of use? As far as use goes, the majority of the target needs to stay in the view of the camera for it to function. Otherwise, how would the system work with multiple targets? Will the character be able to move fluidly from one target to another?
- The AR team wants to mindful of latency issues with the display and game play. If the system requires streaming video or complex graphics over wireless this may be an issue. More research, and smart design should address this challenge.

Glossary

Augmented Reality (AR) is a technology that combines a live, direct or indirect, view of the physical world with computer-generated sensory input such as graphics or sound.[1]
AR Target - also *marker* or *tag* - this report frequently uses the words 'AR target' to describe an object placed in the field of view of an imaging system to be used as a point of reference for augmented reality software.[3] In this context, the target is typically a two dimensional pattern, often printed on a piece of paper. For this game, the imaging system will be the camera on an Android smartphone.
Game Engine - A Software component to be developed by other divisions of *Vi-Char.* From the AR division's point of view, the game engine will provide coordinates, and methods for rendering the virtual world.
Platformer Game - A platformer is "a video game characterized by requiring the player to jump to and from suspended platforms or over obstacles."[2]
Virtual Puppeteering is the act of controlling a virtual character through physical movements with one's body. In this case, the task of capturing a puppeteers movements will be performed by an Xbox Kinect.
Augmentable Zone - The area space that the AR targets are placed in the physical world. AR targets are to be placed in the Augmentable Zone in order to be read. Targets outside of the Augmentable Zone are not augmented.

References

1. “Augmented reality,” Wikipedia, last modified 9/12/2012, accessed 9/13/2012, http://en.wikipedia.org/wiki/Augmented_reality
2. “Platform game,” Wikipedia, last modified 9/11/2012 , accessed 9/14/2012 http://en.wikipedia.org/wiki/Platform_game
3. “Fiduciary mark,” Wikipedia, last modified 9/9/2012, accessed 9/14/2012, http://en.wikipedia.org/wiki/Fiduciary_marker