# Game Engine Development Team

Requirements Specification v1.0


Bahr, Dan
dbahr92@gmail.com

Bard, Etan
ebard@ups.edu

Burns, Nick
nbburns@ups.edu

Livingston, Chris
christopherlivingston92@gmail.com

Wilson, Robin
rkwilson@ups.edu


October 26, 2012

# Contents

## 0.1 Introduction

### 0.1.1 Overview

This document defines the requirements specifications for the development of the game engine for Vi-Char. Specifically this document will provide: summary of the requirements specified herein (see Executive Summary), detailed description of the deliverable and its intended uses (see Application Context), listing of primary and secondary functions provided to users of the system (see Functional Requirements), listing of non-behavioral qualities of the system (see Non-Functional Requirements), a timeline of projected milestones (see Timeline), listing of expected challenges accompanied by plans to address said challenges (see Potential Challenges), and finally a glossary (see Solutions & Coping).

### 0.1.2 Team Member Vital Stats

**Dan Bahr**
**Qualifications**: I have extensive knowledge of C#, previous experience with Unity3D and Blender, and 2 years of work experience with Windows Embedded devices. I have experience with game design and advanced software design.
**Strengths**: Code design and development

**Etan Bard**
**Qualifications**: some C#, some Unity experience
**Strengths**: Creativity, Design, Multi-Media Graphics, Japanese Translation

**Nick Burns**
**Qualifications**: C/C++, OpenGL, Latex, Blender
**Strengths**: Presentation, Graphics, Modeling, A.I.

**Christopher Livingston**
**Qualifications**: Some java experience, MySQL and SQL
**Strengths**: Game design

**Robin Wilson**
**Qualifications**: Experience using Blender and Unity, Fluent in Java, Business Management, Marketing, Video Game Play Experience, Video Game Design Experience
**Strengths**: Creativity, Design, Organization

## 0.2 Executive Summary

Vi-Char is creating a platformer game, incorporating augmented reality, mobile devices, and Kinect puppeteering. The Game Design group will design, develop, and implement the game portion of the aforementioned project The game will be primarily directed towards player-versus-environment gameplay. The

people controlling the in-game character(s) will be able to execute certain actions, including jump, attack, and directional movement. Various items and obstacles exist in the game world, that the player will be able to interact with. These obstacles will be dynamically placed at runtime with the aid of augmented reality. Gameplay will be dynamic and will change based on input from augmented reality and the puppeteers. Vi-Char will draw upon existing platformer games as inspiration to the gameplay including: Super Mario Bros., Traps Mines and a Sheep and a First-Person Super Mario Bros.

## 0.3 Application Context

The end product will produce a game that is aesthetically pleasing and fun to play. The game is a platformer with each person that has a role: move, jump, use special abilities and/or use items. We can have either one character doing all of this or have two characters in total working co-operatively (Mario and Luigi style gameplay). If we were to have two characters, two people would control one character. The Kinect will be able to pick up the actions of the players which will in turn make the character move accordingly. Our product at the end of development should be a game that takes movements from the Kinect and animates a character in Unity. It can take objects given to the game engine from the augmented reality group and create a world for the Character(s) to interact with. The group with the phones should be able to interact with the character(s). At the end of the game the score should be sent to the server to be stored. At the least four people will be able to play it. As per the requirements outlined in this document, the game design division will be taking input from the Kinect (for character movement) and from the android phones (for Augmented Reality information). The Kinect will be talking directly with the computer application within Unity3D, and it will be feeding us with live data about character movements. All communication with the phones will be done through networking via wifi and a server that the Web Development group is managing. In the end, the game engine will be able to take any Augmented Reality data on-the-fly and translate it into interactive objects. Examples of these objects could be crates, boxes, powerups, weapons, and so forth. The gameplay should be dynamic enough to allow drag-and-drop style play where anything could change in a moment's notice. We believe this will allow our users to have the most fun possible. The game, as mentioned before, is a platformer style game similar to Super Mario Bros. The game will be in either 2D or isometric 3D (for example, sidescroller or Real-time Strategy type games).

## 0.4 Functional Requirements

### 0.4.1 Main Use Case

#### Actors

Server, Kinect, Display, Mobile Devices(phones), Game Computer, Puppeteers (Jump, Movement, Special-Movement, Item-use)

#### Preconditions

The Kinect works on the game computer and can read actions of the puppeteers. The puppeteers know which action corresponds to their role. The augmented reality system can identify objects. The network is avalible and functioning with low lag

**Postconditions**

The game computer, along with the phones, and server are functional. The score has been displayed and logged.

**Main Scenario**

1 **The Game Computer starts game engine.**
At this point, the vast majority of the game initialization will be done by the Unity engine. All images and meshes, for example, will be preloaded (but this is all invisible to the user).

2 **The Game Computer contacts the Server.**
This will we done with a request for connection with a predetermined protocol. This connection will be used for the remainder of the gaming session (until termination for any reason).

3 **The game computer connects to the Kinect.**
This step involves any initialization required for establishing and maintaining a connection similar to the connection made with the server.

4 **The game engine displays the main menu.**
This will include options for the puppeteers to pick their roles.

5 **The game engine displays instructions, to the user.**
English language visual instructions will direct players to start the game. Puppeteers will be prompted with instructions on how to choose their roles.

6 **The puppeteers motion to start the game.**
We will receive a command from the Kinect notifying that the game is to start. Visually, this will be where the game appears to begin.

7 **The computer gets augmented reality data from the phones.**
The locations of augmented reality objects will be detected by the augmented reality group and relayed to the game engine through the web server.

8 **The game engine creates the world using the data collected from the augmented reality group about the real-world environment.**
Codes from the augmented reality group will be placemarks for 3D models in the rendered world. These could be buildings, platforms, enemy spawners, etc.

9 **While game continues, the following will occur.**

   a **The actions of the player are used to modify the game state.**
   For example, when we get a command from the Kinect to move forward, the character on screen will do the appropriate action. Player actions can occur at any time.

   b **Augmented Reality and other dynamic data is sent from the phones to the game engine.**
   For example, we may receive a command from the server to place an object in a certain location for the character to interact with.

**c The game state is updated and tracked.**
Any necessary updating is finished before rendering.

**d The game engine renders the world.**
This is all handled by Unity automatically. The game will be displayed on a screen.

**e The world data is sent to the phones.**
Character data, obstacle data, and so forth may all be sent to the server to forward on the phones for rendering.

**10 The game completes.**
The game is finished when a certain victory condition or loss condition is achieved (for example, if the character dies).

**11 The game's statistics are displayed.**
Details about the puppeteer's performance, such as overall score, time until imminent death, enemies killed, in-game accomplishments, etc. are displayed on the screen.

**12 The server logs the game's score.**
We send the score data to the server.

**13 Return to the main menu.**
Here the users can choose to play again (going back to step 4) or quit the game.

**14 A player selects the quit option.**

**15 All network connections are terminated.**
The connections to the server and the Kinect will both be shut down, forcefully if need be.

**16 The game engine shuts down.**
This is handled by Unity. It will clean up any resources used by the game engine and exit the application.

**Alternative Scenarios**

**1-4** Unity fails to load properly.

**Solution:** May be resolved by computer restart or simply restarting Unity3D. At this point, there's not much we can do, as Unity is supposed to handle all game initialization automatically without any user control.

**5-6** If there is a failure to accurately display main menu.

**Solution:** Correct the video settings. It is possible that the computer's graphics settings are not configured correctly. This is the burden of the user to fix.

**Solution:** Restart the game. Again, this is probably Unity's fault.

**any** low frame rate, lag

**Solution:** Reduce the number of displayed objects, reduce textures, reduce anti-aliasing. There are a number of tricks we can do to reduce the number of objects on screen to alleviate this problem.

**Solution:** Provide a mechanism for changing graphics settings to improve performance.

**5-20** There are networking problems.

**Solution:** Check to make sure that all network communications are available. Perform checks on the connection to the server and allow the user to try to reconnect.

**any** If the game fails to respond.

**Solution:** Restart the game. Sorry game users! Let's just hope it never gets to this point ...

## 0.5 Non-Functional Requirements

### 0.5.1 Usability

**1 Interface is Easily Navigable**

The Puppeteers can navigate menus displayed on the main projector, and are able to choose options that they want. Players are able to choose buttons using the Kinect or use the phones. We can tell we accomplished this from Puppeteers feedback.

**2 Legibility of Textual Feedback**

Puppeteers must be able to read text written on the projector from across the room, and mobile users can read text on Mobile Devices from a distance of at least one meter. (Puppeteers feedback)

**3 Replay Value**

The game should be fun to play, and provide challenging gameplay no matter the skill level. Players should be compelled either by the story, the gameplay, or interaction with the other players. We can tell if we achieved this through Puppeteers and Mobile Device operators feedback. If 85

### 0.5.2 Reliability

**1 Stand-Alone Nature of Game Engine**

If the internet or the connections with the Mobile Devices, augmented reality, or Server get interrupted, gameplay should be able to continue through Kinect input.

### 0.5.3 Performance requirements

**1 Fluidity of Gameplay**

The game should run with little to no lag at a framerate of at least 20. The movements done by the Puppeteers should be able to register with the Kinect and the avatar should respond on the main display in the proper way in less than one second. All data sent between the

**2 Ability to support a large number of generated objects.**

The gameplay and the length of time the game loads should not depend on only a few objects being generated.

**3 The game is playable.**

The kinect needs to pick up the movements and display them quickly, the enemies on screen should die when the character kills them, the character animations should work according to the motions of the players.

**4 The kinect identifies movements in a timely manner.**
From when the Puppeteers do a physical movement on the Kinect the Character should move accordingly inside the game and on the Display. It is possible the Motion Capture group will need to deal with this but we will have to take the input from the Kinect and make the Character move.

### 0.5.4 Supportability

**1 Windows 7 on the class' server.**
Because we have machines that run Windows 7 we need to make an engine that works on Windows 7.

### 0.5.5 Implementation

**1 The work is done in Unity3D.**
The Customer would like us to work in Unity because it is a free application.

**2 C# for coding.**
Unity uses C#.

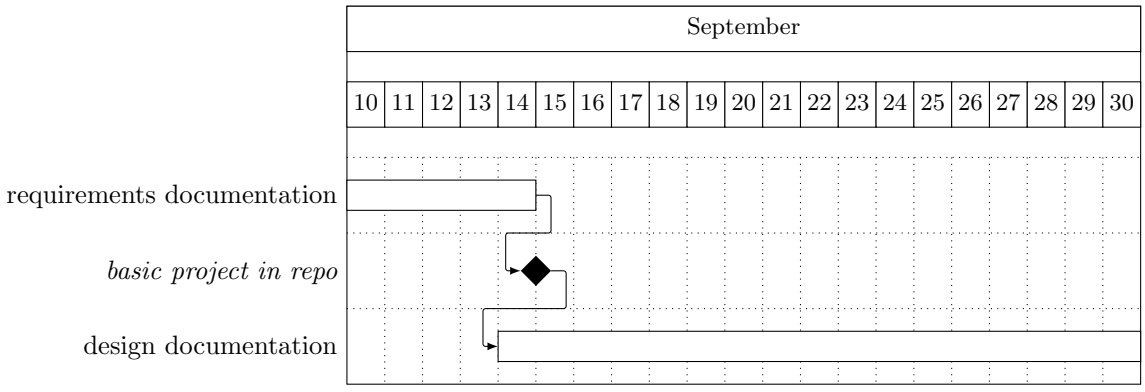
### 0.5.6 Environmental

**1 Aesthetically pleasing**
The graphics should not look only like gray blocky graphic shapes. We would like there to be textures covering the skeletons of the boxes, enemies generated, and on the Character. Although this is not necessary for the gameplay, the Game Design group would like to uphold this standard.

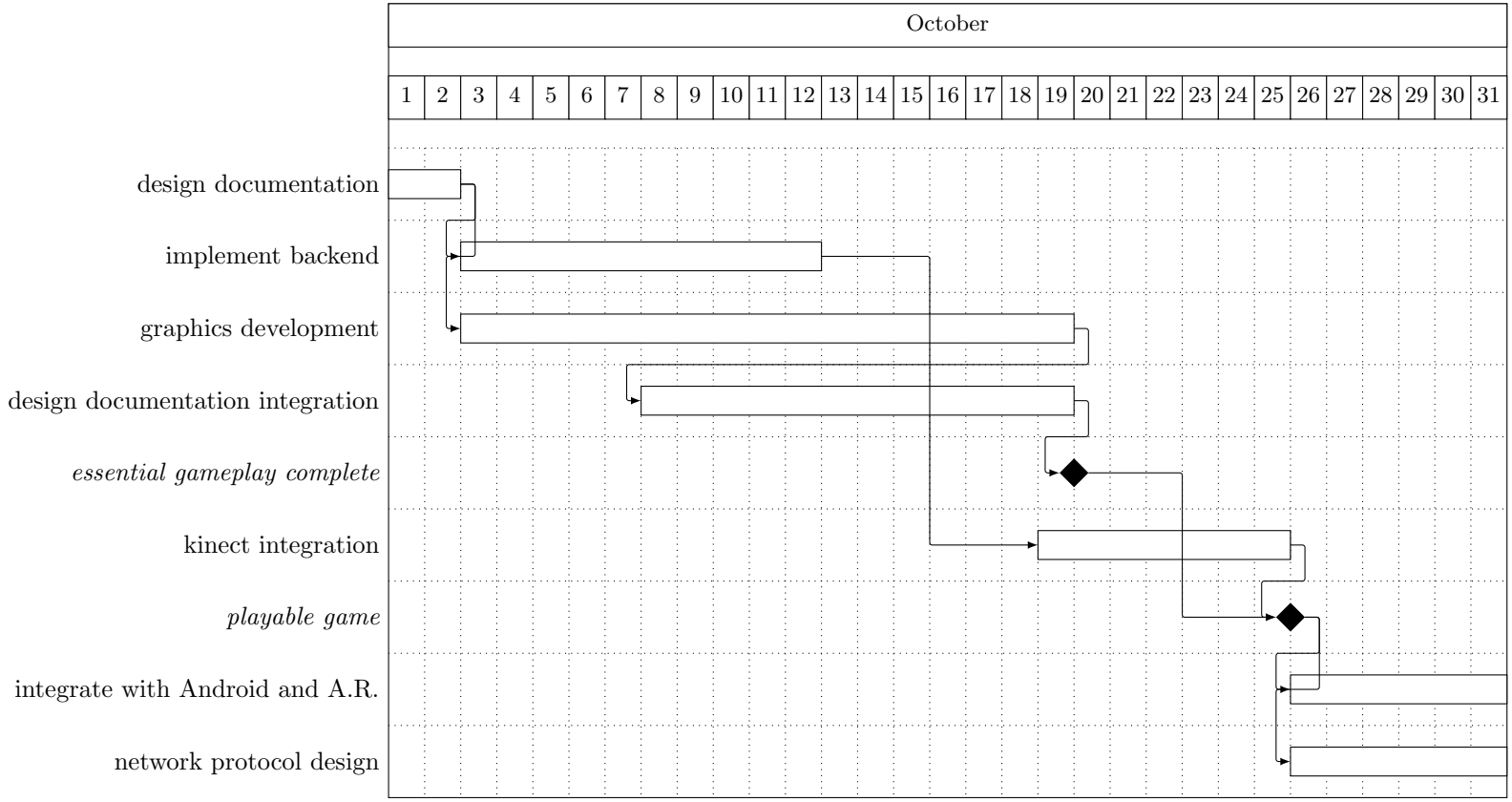
## 0.6 Timeline

The following Gantt charts document the proposed timeline, and dependencies between steps in the project. Horizontal boxes represent work to be done over a time period, and black diamonds denote deadlines prescribed for the project.

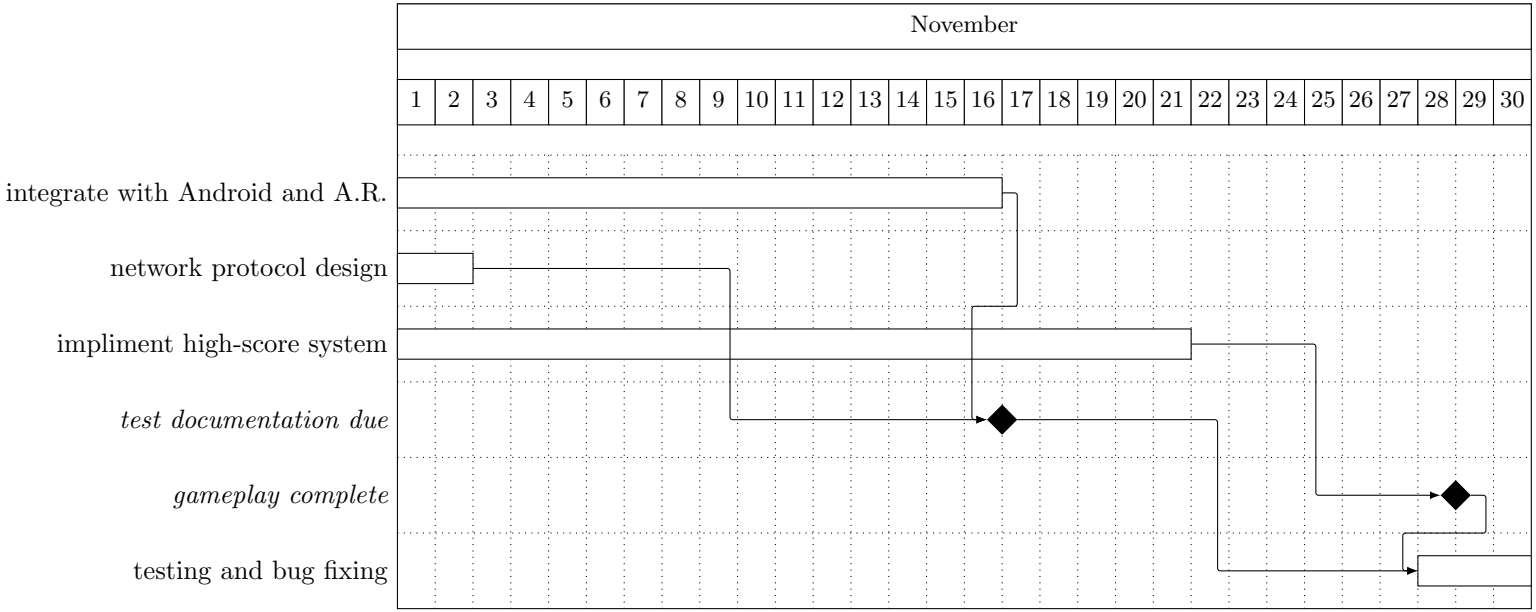
### 0.6.1 September



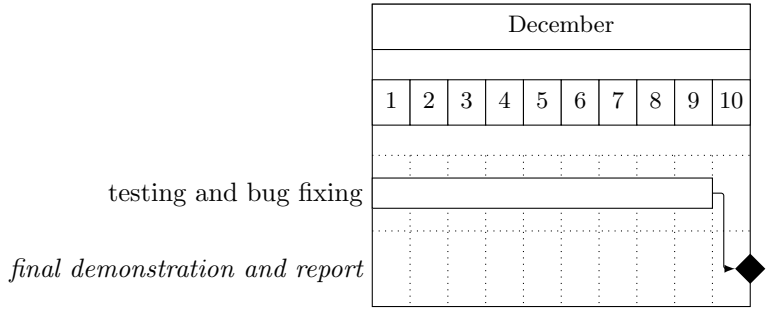
### 0.6.2 October


October
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
design documentation
implement backend
graphics development
design documentation integration
essential gameplay complete
kinect integration
playable game
integrate with Android and A.R.
network protocol design

### 0.6.3 November




### 0.6.4 December



## 0.7 Potential Challenges

**1** Integrating the augmented reality group's work with ours

- **a** augmenting generated world to match up with real world images
- **b** Interaction of objects detected by augmented reality's mobile devices
- **c** We anticipate interacting with the phones and new objects the mobile device and augmented reality group generate will hinder the gameplay by making it lag or dificult to integrate into the gameplay.

**2** Integrating the Kinect commands to work with our keys

- **a** Low latency between Kinect inputs and updating the Display. We anticipate lag issues between the Kinect actions and getting the animation of that action onto the screen and interacting with the environment inside the game.

**3** Interaction from the Mobile group

- **a** Responding to events created by mobile phone users in close to real-time
- **b** Displaying and rendering of game world may slow down mobile users when many objects must be drawn on the screen.
- **c** The connection between the augmented reality world generation and the augmented reality interaction with our character may be too laggy for the players to interact.

**4** Streaming of game feed to Web Group to connect to the mobile devices

- **a** When someone using the mobile device drops an object or foe into the world, we need to ensure the time it takes to generate is not long and does not interrupt the players controlling the character.
- **b** Combining of the two groups' works
- **c** Stemming from the fact that we don't understand the role of the Web Group, we do not know if our computer and the connection with the mobile devices will be consistent for interaction between the two groups.

**5** Maintaining simplicity while still keeping elements of challenge and/or fun

- **a** Enough unique levels to keep audience engaged (less cookie-cutting)
- **b** Maintaining the ability for environmental interaction from mobile users
- **c** Easy learning curve with easily distinguishable world objects
- **d** Because the genre of platformer has been done before repeatedly, we are hoping that the interaction with the AR, Kinect, and the mobile devices will be enough to make the game new and exciting. We want the game to be interesting to look at in addition to being fun especially for people who are only watching. We do not know how difficult the controlling the character will be for the players and if the character dies over and over again because the learning curve is too steep, we know the game will not be fun for the players.

**6** Working within a larger group

- **a** Game Design group needs to accommodate and interact with every division
- **b** communication and integration

**Solutions & Coping**

Working with so many people on this project will certainly be difficult especially since the Game Design group needs to accommodate and interact with every division. We plan on communicating with every group and letting them know what we are going to need from their part when creating this game. As we are designing the program and using Unity3D, we are going to keep our code as efficient as possibly especially where there is the potential to have lag during gameplay. We plan to create a working prototype of the game as soon as possible in order to test the gameplay and the game's "fun factor."

# Glossary

**augmented reality** in this case, a live and direct view of the real world, augmented with computer generated video, graphics, and data.

**game engine** a system designed for the creation and development of video games (from Wikipedia).

**Kinect** a motion sensing device, featuring 3D depth sensors, and a camera.

**Mario and Luigi style gameplay** refers to gameplay in Super Mario Bros, where two players control their own avatar in order to achieve a common goal within a platformer world.

**puppeteers** players who are controlling the in game avatar through body motions captured by the Kinect.

**Super Mario Bros.** a 1985 platform video game developed by Nintendo, published for the Nintendo Entertainment System as a sequel to the 1983 game Mario Bros.

**Traps Mines and a Sheep** An online flash game in which the player deactivates traps inorder to move the avatar as far as possible.

**Unity3D** an integrated authoring tool for creating 3D video games or other interactive content such as architectural visualizations or real-time 3D animations (from Wikipedia).