# Gunnerman Essay

080008650

December 16, 2011

## 1 Design decisions and implementation solutions

### 1.1 Starting with controls

It was believed that the controls for the player would prove to be the most challenging to implement hence they were queued first for development. Thus meaning the architectural design and state would be delayed. It is difficult to evaluate if this was a poor decision, however the aim of the this choice was to provide the best possible control interface.

### 1.2 Fine adjustment

After user feedback, it has been suggested to adjust the left analog pad to support the user dragging outside of the pad range. This prevents the character from suddenly stopping when the player drags the analog control outside of the pad range.

### 1.3 Map and viewport

The map is loaded from a map stored on the SD-card storage of the phone. This however proved troublesome when the game was installed on other phones, and since the other devices did have the required file, the application crashed when it tried to read from the non-existent file. Hence this has been rectified by making a check for the existence of this file. If the file cannot be located, a default map is created and used in game.

The viewport has been implemented by having a GameObject superclass contain a screen coordinate and a world sets. Thus on every render operation the object has to retrieve its screen coordinates to be drawn in the correct position.

### 1.4 Network interaction

Please find a demonstration of a network game here: `http://www.youtube.com/watch?v=EWfbAi-8y-8`

The multiplayer protocol consist of two network phases.

**TCP setup.** The game client connects to a server via a long lived TCP connection. The server immediately responds with an identity notification and a broadcast to all connected players about the new player joining. The client updates its current state accordingly. The player can then send a ready message to the server, which broadcasts a ready notification. Once all the players are ready, the server first sends a game-about-to-start notification containing a time delay in seconds and after the delay broadcasts a game start message. This should change all clients into game state and initialise the second protocol phase. The server then closes all TCP sockets.

**UDP game phase.** All parties start a UDP listener. The clients start broadcasting dead reckoning messages once in a predefined time interval, providing coordinate, acceleration and direction facing information. Once the player shoots, the client sends a firing message sending the coordinates and the direction facing. The server upon receiving the messages broadcasts to all connected clients. And once the client receives the broadcast message updates the state accordingly. In this phase local client state is responsible for determining the game flow.

## 2 Improvements

### 2.1 Cell migration

The game has proven collision cell migration for players to be quite costly. This can be resolved by changing the way the player is added to new cells.

### 2.2 Network delay

The optimal dead reckoning message delay has been chosen to be 250 milliseconds, this provides the most responsive updates while not overloading the server and causing noticeable delay in message transition.

Word count: `Words in text: 480`