Design Document: Maze Game

By Peter Mitchell (mitc0271)

1 Introduction

The purpose of this project is to develop an application for the Android environment that utilises at least the gyroscope or accelerometer in some way. This project will accommodate this requirement by using the accelerometer in two ways. The first will be as part of a ball maze game. The accelerometer reading shall be applied as a velocity to a ball in the game screen that will move then dependent on the changing orientation of the device. The other part is that once a level has been completed in a high score entry screen the accelerometer will be used to control the cycling of letters. Tilting the device depending on the direction will result in either the letter position changing or the current selection's letter being iterated to the next character.

This document serves as a design document to show an outline of the content that will be included in the product. This document includes sections covering a number of areas. The overview will provide an overview of how the game play and features will be portrayed. The storyboard will show as a series of images and explanations how the game will appear visually. The class diagram and activity diagram show the structure and activity flow for the application as it will be in the final product.

2 Overview

This application will provide a simple maze game controlled by the orientation of the device and locally track high scores based on the time taken to complete levels. Each level will have a separate set of high scores. There will only be two levels made in the initial game; however, it will be built to make it easy to add more. The level will be randomly selected each time a new game is requested. High scores will be optionally inserted using the accelerometer. The option to insert high scores will only be presented if the achieved time is less than a time recorded for the specific map.

The maze levels will be more complicated than simply navigating the ball from one side of the maze to the other. They will consist of four different types of elements. Each of these will manage a different function within the maze to increase the variety of puzzle types that can be made. The elements are each in the game to be classified as types of walls. Those walls will be as follows.

- Normal Wall
- Door Wall
- Button Wall
- Level End Wall

In addition to these four, the rest of the levels will be represented as either empty space and not have an associated object or be a special invisible wall that indicates the start location for the ball.

Figure 1: Normal Wall

Figure 2: Level End Wall

The graphics shown above demonstrate the normal wall (seen in Figure1) and the level end wall (seen in Figure 2). The normal wall's function will be simply to block the movement of the ball as an obstruction. When a collision occurs with a wall object, the ball will be moved back to the previous position prior to when a collision occurred. It will be the most commonly appearing object in any level as it will be used to surround the levels with a barrier and form any corridors within levels. The level end tile will facilitate the function of ending the level when a ball collides with it. To reach the end tile the user will need to have navigated through the corridors of the maze likely including the buttons and doors that follow in the next section. As soon as a ball collides with the level end tile the game will change from displaying the maze to displaying the high score screen instead.

Figure 3: Button Pressed Wall

Figure 4: Button Not Pressed Wall

The graphics in Figure 3 and Figure 4 show the button graphic that will appear in the completed game. The purpose of the button is to open a specific door. When a use collides with an individual button the graphic will change from Figure 4 to show Figure 3. This will signify that the button has been pressed. An associated door somewhere else in the level will be triggered open as part of this action to. Buttons will have one unique number that they call out to all the doors that are part of the game. When that happens all the doors that are listening for that number will be triggered. This does mean that a single button could open multiple doors if that was a desired function. This will not be included as a feature of the example levels though. Collisions with the button will only occur once, and will not restrict the ball from moving over them.

To pair with the button graphics are the door graphics. Figure 5 is the default state that closed walls will take. When in this form the doors will act exactly like a wall and prevent entry of the ball. Once a corresponding button is pressed the door will change to Figure 6 to show that the way is open. Each door can listen for one number to be called out. But as was mentioned in the previous paragraph this does mean that multiple doors could be listening for the same number or even multiple buttons calling out to open the same door. Although the application of the later would be less useful as it would make buttons redundant.

Figure 5: Door Closed Wall

Figure 6: Door Open Wall

The ball will be represented by a simple sphere. For simplicity this will likely just be rendered with an oval in the application. This will save needing to create an additional graphic to accommodate the ball object. The ball will use collisions against objects based on its exact size, so this will mean the circle should be rendered exactly on top of where it will be matched to with the collision detection.

3 Storyboard

In this section there will be for each visible screen in the application a description of what it will include and how it fits into the application.

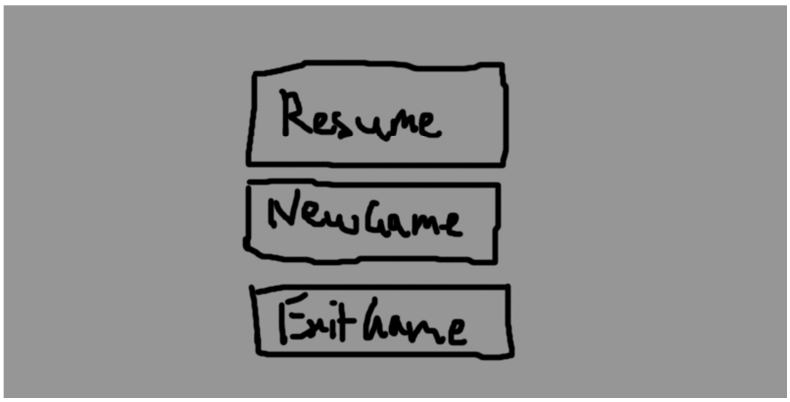

Figure 7: Pause Menu

The pause menu will appear only when the use presses the back button on the phone device. Pressing the button again will be the same as pressing “Resume”. This will just hide the menu again. New game will discard any current level or activity and begin a new random maze. And exit game will terminate the application.

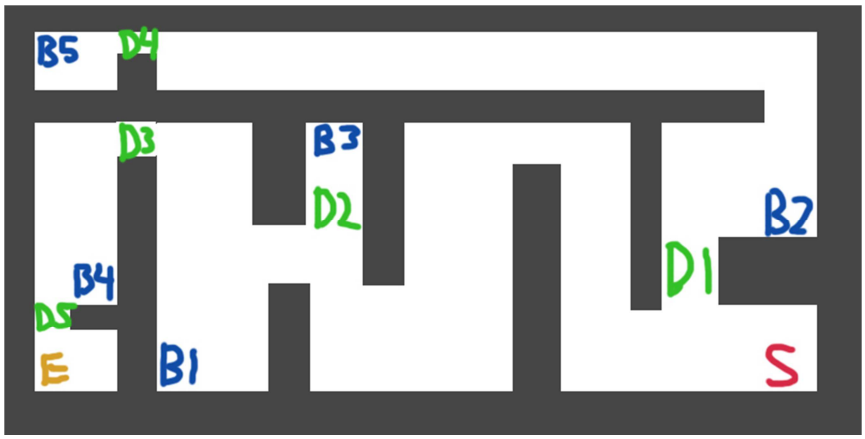

Figure 8: Maze (Level 1)

Figure 8 shows one of two mazes that the user might be presented with. In this maze the red "S" represents where the ball would start, and the orange "E" where the level would end. The blue markers with "B" are button locations, and green markers with "D" are doors. To solve this level the user will need to hit the buttons in the order of: B1, B2, B3, B4, and B5. This will be very straight forward as after each button there will just be one button that can next be hit. Once the ball reaches the orange E the user will be taken to the high score screens.

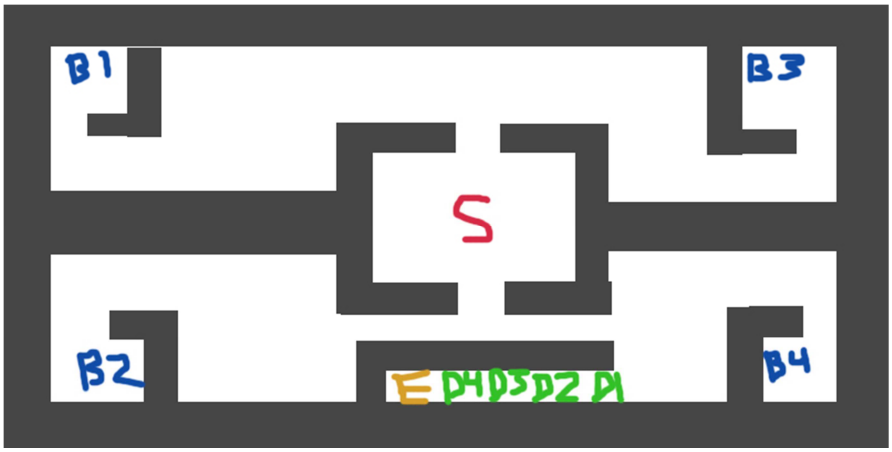

Figure 9: Maze (Level 2)

Figure 9 shows the other maze that the user may be presented with. The representation method is the same as the previous example. The main difference with this one is that it does not matter at all the order that the buttons are pressed in. Again, once the user has opened all the doors using the buttons and reached the orange "E" they will be taken to the high score screens.

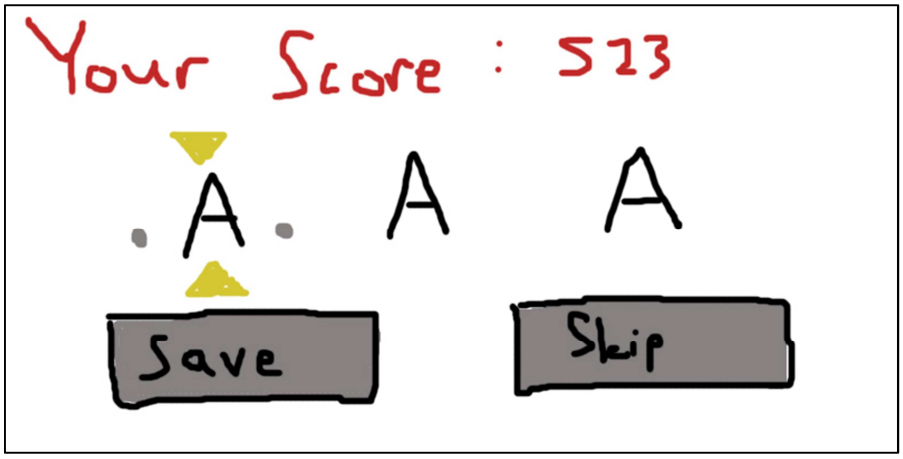

Figure 10: High Score Entry

Figure 10 shows how the high score screen will initially appear if the score was good enough to reach the high scores table. High scoring users can store their name as three letters. The currently selected letter is designated by the pair of yellow pointing arrows. Tilting the device up will decrement the letter that is currently displayed for the selected character. (Eg, Z would be next). Tilting the device down would increment the character. (Eg, B would be next). The transition between these characters is represented by the two grey dots alongside. These dots will move as a user approaches changing to the next letter and then reset to the middle after the change. Tilting the device to the right would shift the selected character to the right until it can't go further right. Likewise, tilting to the left would shift the selected character to the left. A close to flat orientation on an axis will halt movement in a direction. Pressing save once the desired letters are selected will save the score. If the score is to be not saved for some reason, then pressing skip will continue without saving. In either case after completion progress will continue to show the full score list. As seen in the next figure.

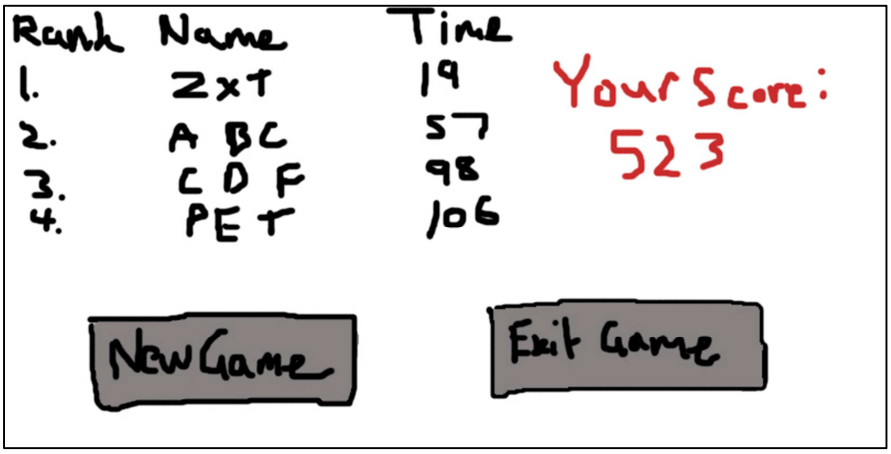

Figure 11: High Score List

This will simply show the high scores (up to 5) and the score that was achieved. Pressing New Game will start a new game on a random maze, and exit game will close the game.

4 Activity Diagram

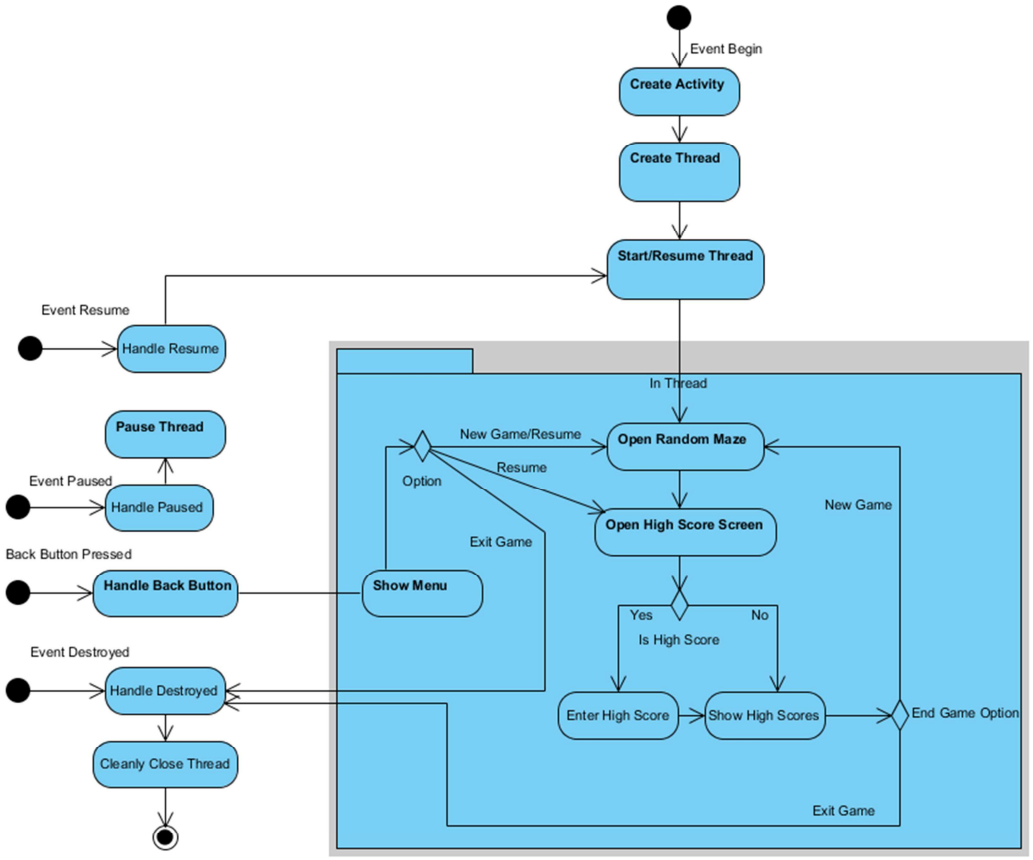

Figure 12: Activity Diagram

Figure 12 above shows the flow of progress as described in the storyboard. In this particular figure though, it also includes the creation of the game and handling of the multithreading. This is a particular important element for consideration. In extension to the storyboard it shows that when the events for pausing, resuming, and destroying occur there need to be additional handling to make everything run smoothly. The figure's actions are self-explanatory. The reason for them is to reduce the CPU usage on mobile devices so that there isn't a continually running thread in the background.

5 Class Diagram

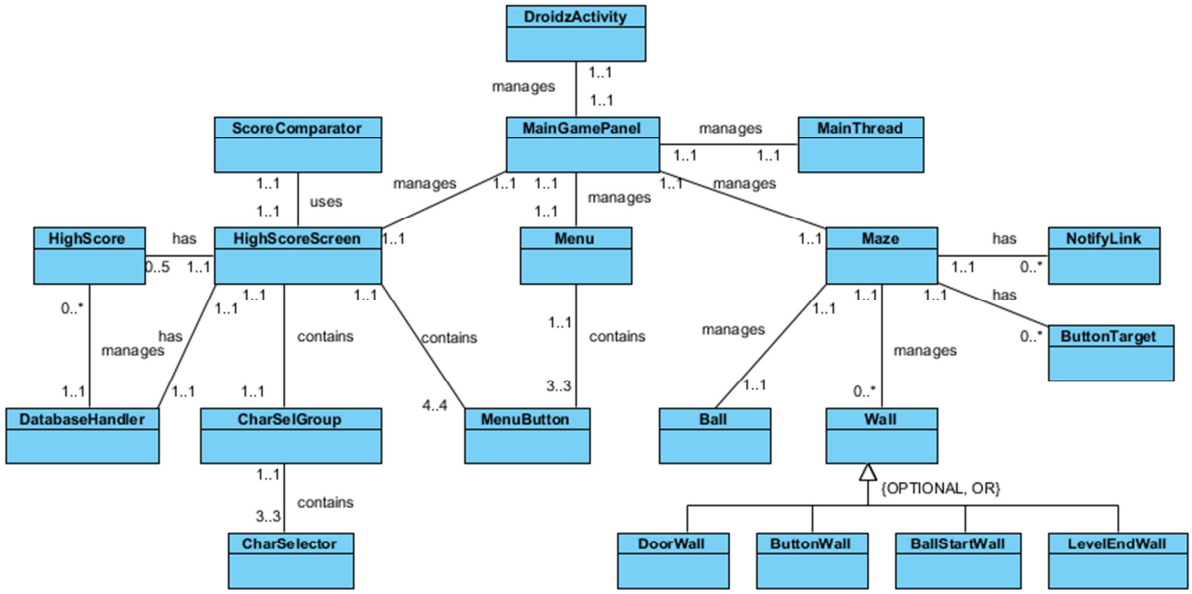

Figure 13: Class Diagram

The class diagram above simply shows how everything in the complete application will be structured. The DroidzActivity is the starting point that creates and passes control to the MainGamePanel and MainThread. These two classes manage the three screens “Menu”, “Maze”, and “HighScoreScreen”. The Menu just has buttons. Maze will have a ball and walls of different types in addition to supporting objects NotifyLink and ButtonTarget. Those last two will be for assisting in matching up where walls and buttons are linking. The HighScoreScreen tracks HighScores and uses a DatabaseHandler to access a SQLite database on the phone. The scores can be sorted using ScoreComparator. Finally the group of character selection elements are represented by the CharSelGroup and then individually as CharSelector objects in this diagram. Together these make what the game will be built up of.