Development Report: Maze Game

By Peter Mitchell (mitc0271)

1 Introduction

This was my first application developed for Android devices, so there was certainly a bit of learning curve. While developing there were a number of times that I had to search for solutions. Often tutorials with the basics of how to begin the implementation of features on the Android system were enough. The largest issues encountered while developing were more oversights of my own part rather than any amazingly complex issue. The most serious of these was related to the use of multithreading and caused crashes, failed application closing, and other issues. This is one of the things that will be discussed later as part of the development issues section of this report.

As part of this report a number of sections will be discussed in regard to the project as a whole and some of the specifics. The first section will cover an overview of how the game turned out using screenshots that may be compared against the concept art in the other document can be viewed. The next section will review the use of Eclipse as the IDE that was used for development and how that factored in as an element of the development. Following that will be an overview of the decisions that were made as part of the project that brought the application to what it has become. This will include identification of a few relevant sources that were used to create the basis for a number of the engine concepts. Finally the development issues and resolutions that had to be used will be covered too.

Figure 1: Launcher Icon

Before going on to discuss the rest of the application, Figure 1 shows the launcher icon. It is fitting this appear on the front page as this is the icon that I made to start the application. And so too is it an entry point to the rest of this report.

2 How It Turned Out

Overall the application works quite nicely. After completing the application I had my brother test it out. He thought I had made the movement of the ball and the iterating through the letters too slow as his main response. These were deliberately kept at the speed they are to make it easier to handle in what is really a prototype game. After spending many hours resolving issues that turned out to be stupidly simple fixes the game came out as the following series of images will show.

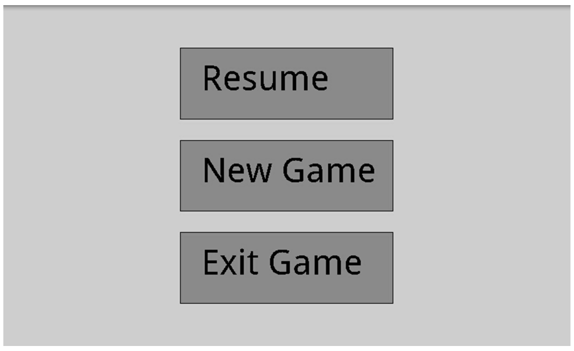


Figure 2: Pause Menu

There isn’t really very much that could be said about the Pause Menu seen in Figure 2. It appears and disappears when the back button is pressed. The buttons that appear in it function as they were described in the design document. Resume will return to the previous view, new game starts a new random maze, and exit game closes the game.

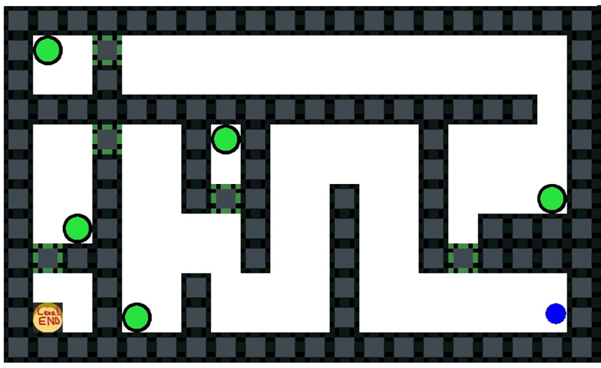


Figure 3: Maze (Level 1)

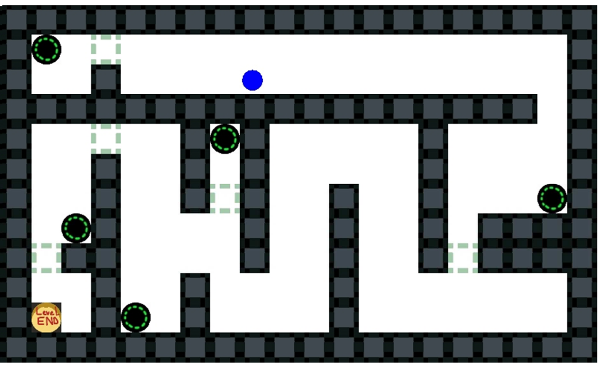


Figure 4: Maze (Level 1) buttons all hit.

Figure 3 and Figure 4 show a pair of different screen shot views of level 1. In Figure 3 it shows the map as it was designed in the concept art for the storyboard of the design document. Figure 4 shows a near completed run through of the maze. It can be seen that in this capture all the doors have been opened and all that is left is for the user to navigate the ball to the end of the level.

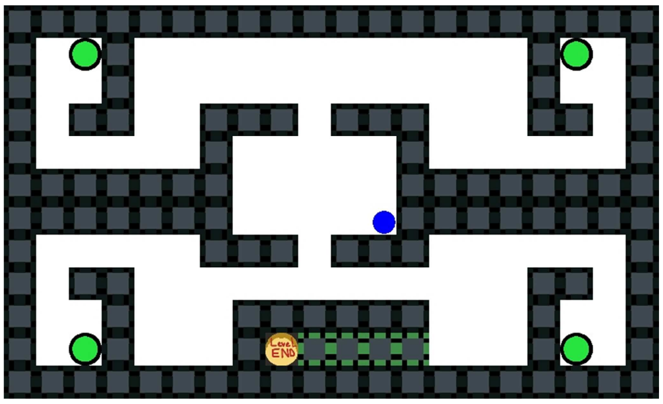


Figure 5: Maze (Level 2)

Figure 2 shows the completed maze level 2. The level operates as it was described in the design document. The four buttons in each of the corners can be pressed in any order and they will each open one of the four doors that block the path to the level end. After opening all four of the doors the player can reach the level end.

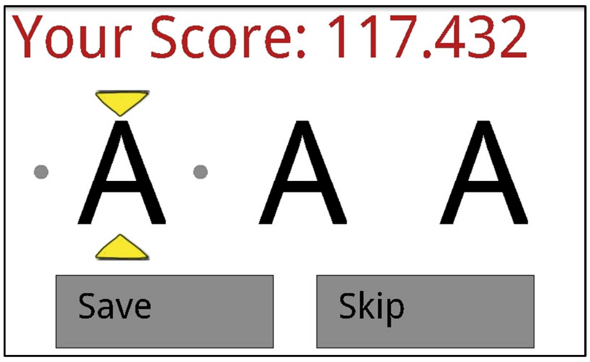
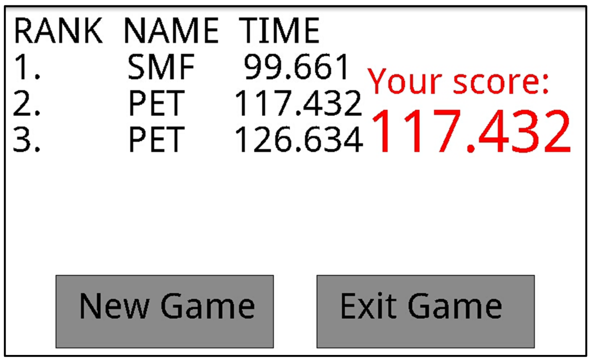


Figure 6: High Score Input



```
RANK  NAME  TIME
1.    SMF    99.661
2.    PET   117.432
3.    PET   126.634
```

Figure 7: High Score Display

The SQLite database was the main part that made the high score screen take a long time to develop. A number of issues that I will discuss in the development issues section made it difficult to work out what was wrong at first. Apart from the issues discovered the high score input in Figure 6 uses the accelerometer to modify the characters. Then the user can save the score or skip straight to the scores screen seen in Figure 7. The database will be updated between these screens as necessary removing scores that are the largest if the number of scores that were in there was 5 (the maximum per maze level).

3 IDE Used

The normal IDE I use for Java based development is NetBeans. I had tried to use Eclipse early on when trying out NetBeans and Eclipse for the first time. I had chosen NetBeans over Eclipse mostly because Eclipse broke on me and NetBeans did everything that I wanted and needed of it. The reason for selecting to use Eclipse to develop this application was the obvious reason that there is the most support in place for Eclipse for developing Android applications. After getting all the way through the installation of the packages required to develop for Android I found it quite easy to develop in. As I am used to using the NetBeans interface I do still have to go looking for functions that I'm used to knowing instantly where they are in NetBeans. I did not bother using an emulator at any point during development from Eclipse, although I did configure one. Instead I deployed each build directly to my own Galaxy S2 that was running version 2.3.3 of the Android operating system. As almost all of the interaction that I needed to test was generally controlled by the orientation of the device this made it far easier to manage.

Eclipse automatically detected my phone the first time and was able to deploy to that without any hassle. Likewise for the rest of the set of features used the worked well. Admittedly the majority of use within the development environment consisted of ordinary typing of code and execution. The next most used feature would have been the integrated LogCat display. The LogCat display was particularly useful. Most of the time the useful information I needed appeared in an easy to view way within the LogCat dialog. There were times when it falsely displayed error messages. When registering and unregistering listeners on earlier versions of Android the messages that are supposed to be simply informational messages become displayed as error messages. This is an issue that is part of the Android operating system rather than one that is part of Eclipse. Overall I feel that Eclipse worked well as a development environment for working with Android.

4 Decisions Made

4.1 Slow movement speed

It was mentioned earlier in this report that my brother tested the game and found the input to be slow. This was a deliberate decision made, and initially it was made for two reasons. One of those reasons was partially removed as an issue later in development though. Firstly in relation to the ball movement, the ball has a locked maximum speed. The acceleration is applied with scaled time to modify the relative velocity of the ball. This means that to quickly change direction the orientation of the device must change a lot quickly too. Initially the collisions between the ball and object resulted in a case of if it couldn't move forward in the direction it was headed then the balls location would be frozen back to the last non-colliding location. Not only this, but the velocity would be retained and thus to make the ball move again a large orientation change was needed. This difficult control mechanism made it hard to navigate the maze as you would be stopped constantly. To improve these difficulties the decision was made to do two things. To have the ball's velocity reset to 0 when a collision is detected, and to improve the collision detection. At the cost of an additional cycle or two per update for the ball to check if it was colliding if it instead moved in just one of the x or y directions. If a movement in just one of the directions is successful it is taken and the movement is made. A collision that stops both directions will halt the ball still though. This instantly made the movement of the ball far easier to manage. The ball's speed was still left slow so that it would be still very easy to navigate in this prototype game though.

The other part that had slow speed was the character selection input group for the high score screen. Initially a technique much like the ball was used to apply acceleration to a velocity and then after passing a certain value the letter would be changed. This didn't really work particularly well when trying it out. Certainly it could work if a perfect set of values to use are found, but it was far simpler to ignore updating a velocity. Instead it was decided that if the orientation of the device was enough in a particular direction the velocity would be set instantly to a specific speed. This of course would then be reset to 0 if you levelled the device.

4.2 Game Thread

The aim for the game thread was to simulate as much as possible the approach provided by the XNA engine. Since the Windows version of this application will be developed in XNA it was logical to try and do this with the Android version. Not only is XNA development more native to what I have been doing plenty of recently, but also it makes porting the application far easier. To accomplish this two separate game loop tutorials were combined.

Firstly the one at: http://obviam.net/index.php/a-very-basic-the-game-loop-for-android/

This particular one showed a very simplistic game loop that included creating the class files required, but the loop itself wasn't very useful. So additionally I also found the following:

http://examples.javacodegeeks.com/android/games/main-loop/game-loop-example

In this one it has just the loop part, so the skeleton of the other code was used and this was placed inside. For the most part this worked when initially tested. There were some issues that required additional thread management modifications that neither of the sets of code accounted for though. These will be covered in the develop issues and resolutions though. The most significant change to how the game loop operates other than the changes to the thread management is that the code here assumes that time if it needs to be, is tracked by the individual objects. To truly simulate the XNA engine of course when updating the game time must be passed, so I had it calculate this and pass it through. This is mainly used for calculation of updating positions based on velocities. It could be adapted to really any game though.

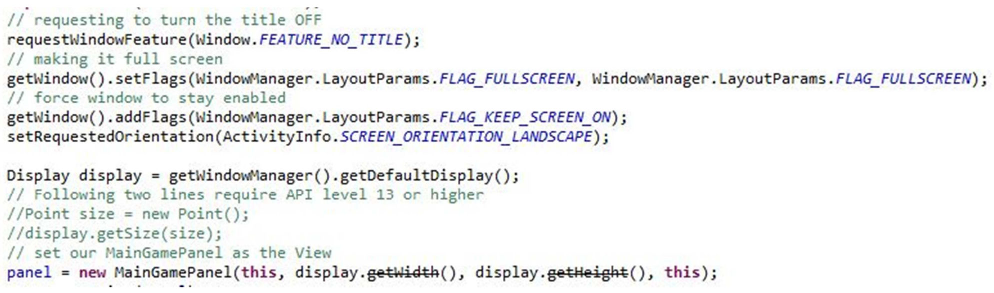
5 Development Issues and Resolutions

5.1 Configuring the Screen

The configuration of the initial game setup provided by the game loop described in 4.2 did not take into account configuring the screen to how it needed to be. For the game being developed it really needed to have the following properties.

- Full screen
- Locked orientation
- Always on
- Knowledge of the dimensions of the screen

Initially the default for the Android device has the screen cut down with the bar at the top, the orientation changes with the device, and the screen turns off if you don't touch it regularly enough.

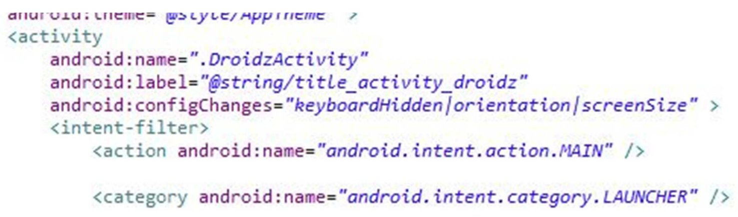


```
// requesting to turn the title OFF
requestWindowFeature(Window.FEATURE_NO_TITLE);
// making it full screen
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);
// force window to stay enabled
getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

Display display = getWindowManager().getDefaultDisplay();
// Following two lines require API level 13 or higher
//Point size = new Point();
//display.getSize(size);
// set our MainGamePanel as the View
panel = new MainGamePanel(this, display.~~getWidth~~(), display.~~getHeight~~(), this);
```

Figure 8: Screen Configuration code from DroidzActivity.java

The code above is the resolution to most of this. It removes the title bar, requests full screen, locks the screen in an on state and keeps the orientation in landscape. While on the topic of orientation it was found that this setting could not be changed by the loading application if the lock screen was up while loading the application via Eclipse. Therefore the screen would be locked in portrait mode instead. This would not occur ever when running from the device so it isn't really an issue. Also in the comments near the bottom it can be seen that deprecated methods had to be used for getting the width and height as the new method is not supported below API version 13. Additionally to the code above a change in the manifest file had to be made to modify the screen to full size and to allow locking the orientation. That change can be seen below in the configChanges having added "orientation" and "screenSize" as options.



```
android:theme="@style/AppTheme" >
<activity
    android:name=".DroidzActivity"
    android:label="@string/title_activity_droidz"
    android:configChanges="keyboardHidden|orientation|screenSize" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
```

Figure 9: Screen option change request in AndroidManifest.xml

5.2 Thread Management

The thread management changes could be discussed at length, but to keep this short, the main issue encountered was that the thread would never shut down. This resulted in a number of subsequent issues. Starting the application subsequent times on the device would result in a locked screen that you couldn't interact with and had to spam the home button till task manager could be used to close the process. Initially this would also indicate that the process would be using 25% CPU. The thread sitting in the background sometimes retained the screen on property, so not only was the application chewing up CPU cycles; it was also draining the battery using the screen. To solve this particular issue it was found that the thread was being told to close when a pause event occurred. This then was made worse that the thread had a line of broken code the blocked it from ever terminating. To resolve this issue an additional layer of thread synchronisation was added by using the onPause, onResume, onStop, and onDestroyed methods to tell the thread what it was supposed to be doing when. This modification that may be seen through the use of mPauseLock causes a vastly improved application. It resolved the thread management so that successful termination when the application has to completely close can occur; it also resolved the pausing issue. When the application is paused now, it will not use any CPU until a resume has occurred.

5.3 Collision Detect Irregularities

Collision detection has already been discussed, but there is one issue in particular that was not mentioned earlier. The collision detection works fine, except that using the default x and y location for the centre of the circle did not result in correct collisions. It was found that modification of subtracting 1.4 times the radius from the x and a y location was required to roughly match it up. This particular issue is still a mystery, and the dodgy fix of modifying the position worked well enough for the game to work.

5.4 SQLite

A tutorial was used to base the code for connecting to the SQLite database. http://www.androidhive.info/2011/11/android-sqlite-database-tutorial/

When initially starting to use the content of the tutorial it was an oversight to have created the configuration of the database's table in the wrong order. Unlike other languages such as PHP that allow access using the columns name, in this system a Cursor object is used that steps through each column in order. After discovering this issue and that I can forgotten to include a line to add elements to the list of high scores being returned it all worked perfectly.