SFAS Diary

# Day 1

On the first day I initially began by fixing the compile and run-time errors so that I could run the game. The errors were quite simple, such as an overflowing char buffer, and didn’t take long to fix.

I then began to plan out the game. I decided to make a matching game, where the user must match up two of the same shapes. I started making mock ups of some of the images I wanted to use. In order to give the game more of a “Search for a Star” theme, I took inspiration from the “Search for a Star” banner, which features 3 alien-type characters. I decided to base my designs off these characters. I tried to make the design “cartoony” and light in order to match the theme of the game.

My first changes were quite simple, and involved: changing the timer to count down rather than up, stripping out the grid and replacing each icon with a star, adding new backgrounds, and rendering the game over scene when time ran out.

# Day 2

On the second day I began implementing the game functionality. I added functionality to randomly generate characters which would initially start off as invisible until the player touched the stars on the grid. I designed 12 different types of characters, with 4 colours and 3 different designs. Each of these types of characters would appear exactly twice on the grid, such that the player could match up two of the same character types.

I incorporated states in to the main game scene in order to keep track of what state the player was in. At this point I worked on implementing the matching functionality so that the player score would increase if the player managed to correctly match two characters.

I added a timer class so that asynchronous events could be used. This timer class was taken from the Marmalade SDK tutorials. This was used to remove characters from the scene after a certain time if the characters were matched correctly.

I then worked on displaying the end score of the player on the results screen.

# Day 3

On the third day I worked on making the design resolution independent. Whilst Marmalade SDK offers a way of doing this automatically using virtual width and height, I decided to control this myself with the hopes of providing higher resolution textures when the game is run on a device with a larger screen.

I then worked on adding music and sound effects to the scenes. I used BFXR, a sound effects synth, in order to generate the sound effects, and the music I found on <http://freepd.com/>, where the music is listed as public domain. I tried to go for music which was upbeat but still relaxing at the same time in order to match the casual nature of the game.

I added buttons such that the player could toggle the music and sound in the game. Whilst doing this I noticed a known bug in Marmalade SDK, where on some devices the first sound effect used is silent. As this doesn’t happen on all devices you cannot just play the first sound twice. I got around this by playing a “dummy” sound, which made no noise.

# Day 4

On the fourth day I worked on implementing the power ups. I included two power ups, one in the form of a gold character and the other in the form of a silver character. Initially I had planned that the gold character would display the whole board to the player, and allow them to see all matches, however due to time constraints I did not implement this. Instead the gold character, if matched correctly, increases the player’s time limit by 20 seconds and gives a score of 50 points, and the silver character, if matched correctly, gives double points for 10 seconds and gives a score of 20 points.

The player only has one chance to match the power up character from the first time they unveil the character. I spent some time thinking about whether this mechanic was too harsh on the player, however I felt that this gave the player a reason to be more strategic rather than just randomly touching the device as fast as they could to unveil the characters. I tampered with a number of probabilities for the chance that a gold or a silver pair of characters would appear. I settled on a probability of 0.2 and 0.3 for gold and silver characters, respectively.

# Day 5

On the fifth day I decided to clean up the code and make sure that I was following the coding standards outline in this assessment. As I was developing quite rapidly with a loose plan it was easy to fall into the trap of writing messy code.

Whilst doing this I found a case which I had not handled concerning power ups. Since there could be multiple pairs of gold/silver characters on the same board, and that any pair of gold or silver characters could be matched up, this caused a problem if the player matched up two gold or silver characters which then left two odd characters which could not be matched up on the board. I solved this and then was satisfied that the power ups were fully working.

# Day 6

On the sixth day I worked on designing a new settings menu for the game. I was unsatisfied with the size of the music and sound buttons so I decided to implement a game menu which could also be used to pause and exit the game. This settings menu was shared across all scenes in the game, so that the player could turn off music in the title scene, for example, and no music would play in the game scene.

I then started work on creating a persistent leader board. I initially tried saving the player’s names and scores using Marmalade’s Secure Storage API. However the documentation is scarce for this API, and its use appears to be discouraged by the Marmalade community. I tried numerous attempts in order to use this API however it seemed to return corrupt data so I decided it would be best to tackle this problem with a different approach.

# Day 7

On the seventh day I again worked on creating a persistent leader board. Rather than using the Secure Storage API, I instead decided to save the players’ names and scores to a file, which could be read from at a later date. The game stores the top 5 highest scores, and asks the player to input a name whenever a new high score is made.

I was conscious about the amount of time left before the SFAS deadline, however I really wanted to implement some multiplayer functionality. I decided at the start that I wanted to create a game mechanic where the device’s screen is split into two grids, one for player 1 and one for player 2, and each user tries to match up the most characters before the time ran out. I decided to commit to this as I thought it would add much more fun to my game. I decided to change the gold power up to give triple points to the player, rather than increasing the time limit by 20 seconds. Whilst there were numerous ways in which I could handle a mismatch between the amount of time left between player 1 and player 2, I thought it would add complexity which I did not have time to handle. I eventually got the multiplayer functionality to a point I was happy with and decided to use the rest of the time to clean up the code and add some visual improvements to the game.

I added an instructions page to the game which explains the goal of the game and describes what each power up in the game does. I also added text, which fades away after a certain time, underneath the player score and time (on single player) in order to give some feedback on what the player did to increase their time/score.

There were a number of images which were not scaling properly when used on a device with a different resolution, which I fixed. Unfortunately I did not have enough time to add higher resolution images and then intelligently pick out the closest resolution for a device. I included only low resolution designs as I noticed that the game was taking a while to load on low powered devices due to the amount of textures, and was even causing the game to crash on my phone.

Lastly I cleaned up some variables and methods and tracked down a few stray memory leaks.

All in all, I had a lot of fun. My plan is to extend this game and upload it the android/windows 8 phone market.