Colum Bennett / 20044622

BreakOut

Multimedia Programming

Contents

[Introduction 2](#_Toc355809722)

[Technologies 2](#_Toc355809723)

[Hardware 2](#_Toc355809724)

[Software 2](#_Toc355809725)

[Coding strategy 2](#_Toc355809726)

[Class Diagram 4](#_Toc355809727)

[GameClient Class 5](#_Toc355809728)

[Block Class 5](#_Toc355809729)

[User Class 5](#_Toc355809730)

[Paddle Class 5](#_Toc355809731)

[Ball Class 5](#_Toc355809732)

[LoadData Class 5](#_Toc355809733)

[WriteData Class 6](#_Toc355809734)

[SimpleThread Class 6](#_Toc355809735)

[Conclusion 6](#_Toc355809736)

# Introduction

For my final assignment for Multimedia Programming, I have decided to recreate the classic retro game, Break Out. As per the project design specifications, I will be required to read data from a text file and implement within the program.

# Technologies

## Hardware

* MacBook Pro 13 inch
* Blacknight Shared Server

## Software

* IDE – Eclipse Juno Mac OS – Game / JavaDoc Development
* IDE – TextWrangler – Web Development
* FTP – Filezilla
* CS6 – Photoshop – Image Editing
* Processing with Eclipse
* Processing P5 plug in – OS applications (Mac, Windows, Linux)
* Java Version 1.6 (Stable compatible version for Mac OS)

# Coding strategy

I initially dived straight into the development of the game, not considering all elements required and how they would interact with each other. Having got my game to a display the blocks and adding user control, I then realised that I would need to consider the collision or intersection of the ball in order to remove the block from the applet. This required me to re think how my program would operate. I particularly concentrated on how I could store, instantiations of my Block class. After much research and trying various options such as 2D arrays, I derived that using an Array List would be my best option, particularly because of the dynamic nature of an array list.

I began my project again by mapping out in pseudo code what I wanted to achieve in terms of the project. As there was a requirement for use of data input from a text file, I decided that I would use this to plot my blocks. Adding the X and Y coordinates of the blocks to a text file “vert.txt”. In order to accurately plot the blocks I sketched a rough diagram and plotted each block. Considering the need to apply colours to the blocks I also created “rgb.txt” to read the RGB colour values for the blocks. Included in the specification of the project was the requirement to use a Thread, as I couldn’t decide how best to use the thread within my project, I decided to also create an “output.txt” file. This would be run by the Thread class separately to the initial game.

From this planning stage I derived that I would need the following classes in order to accomplish my project target,

1. GameClient – Main Class
2. Block – Generate Blocks
3. User – Generate User profile
4. Paddle – Generate Paddle
5. Ball – Generate Ball
6. LoadData – Read text file data for use within program
7. WriteData – Create output text file
8. Simple Thread – Thread Class to run WriteData process

## Class Diagram

## GameClient Class

This is the main class for my program, extending the Papplet in this class. I did most of my programming work within this class. I used an array list here to add **instantiations** of the Block class. From here I was able to remove a block from the applet, based on a Boolean of whether the ball’s X/Y coordinates intersects that of the blocks. If so I could remove the block using the remove() method.

This class also included code to present with a welcome screen, using the Swing class to that in players name and displaying to screen. As I progressed with development I decided to add an extra level and some game stats to improve User interaction with the game.

## Block Class

This is the main block class used to create the blocks for the game. Setting height and width in the constructor and reading the data from the text files to create the blocks. The blocks are created using the rect() method within Processing.

## User Class

This class stores details about the user. In this class I mad full use of Accessor and Mutator methods allowing me to retrieve data such as name, current live and score. These details will then be printed to screen and output.txt file.

## Paddle Class

I had considered making the paddle from an instantiations of the block class, upon further thought I decided for intersection detection it would be easier to create the paddle class. This class includes methods to move the paddle along the x-axis and a Boolean method to detect intersection with the ball.

## Ball Class

This is a basic class to generate the ball for use within the game. It uses the ellipse method in Processing. This class also includes the method to create the ball animation.

## LoadData Class

This is the main class for reading data from a text file. In consists of 2 main methods, one for reading the vertices of the blocks and the other for reading RGB values.

## WriteData Class

This is a basic class using BufferedWriter class to create my output.txt file.

## SimpleThread Class

This class is used to create the output.txt concurrently with the running of the applet.

# Conclusion

I enjoyed working on this assignment. I did feel however that there was a steep learning curve particularly in the case of getting certain Java element to work with Processing. It would also have been preferable to have cover elements such as deploying as a web application or using Google App Engine.

While I was successful in the deployment of a working web application, I initially lost all my code when deployed to the web. After reworking the code I was able to display elements of the program, but I was still unable to display the blocks. After endless research on the net, I derived that due to the export of the runnable jar, the application was unable to find the text files for use within the game, even though the files are located in the root of the jar when uncompressed. I was able to overcome this issue by creating an InputStream will located the resource in question, then passing it on the Scanner to parse the vertices for the blocks.

I discovered during this project that exporting an application for the web is not as straight forward as clicking export. I had to change elements of my code in order for it to work online. More knowledge in this area would have been appreciated.

I also initially had trouble trying to store the vertices in a 2D array using the Processing method loadStrings, eventually I abandoned this option, and studied ArrayLists to store my Block objects. This was a better option in the end as adding and removing the objects was a lot easier.

I tried to embed fonts into my project. For some reason I was unable to achieve this as I kept receiving an error that the font file could not be found. Having tried creating a data directory to store the files I was still unable to embed the fonts. I still don’t understand why this does not work as any example I looked at stated that the files should be kept in the same location as any images for the project. The fonts do work locally.

Another issue that kept cropping up was during export of the runnable jar all my class files were deleted from the bin folder. I was able to overcome this issue initially by creating a new project. Eventually I was informed that you can use a option built into Eclipse to rebuild the class files.

While I enjoyed the project, all aspects including the challenges I did not get to complete the game to the specification that I set myself. Due to my time constraints I had to abandon the top scores screen and printing this detail to the output text file. While the output file is working and createssome test data, it would have been preferable to have it workingas initially expected.

I didn’t like using processing so much in this assignment, although I do believe I will return to Processing but using it with JavaScript in the future.

Overall I believe I have expanded my knowledge of Java from this assignment.