

Squares

Nick Whyte & Dylan Martin
Year 12 Python/Pygame Programming Task
November 2012



<i>Introduction</i>	<i>3</i>
About Squares	3
About NWPi	3
<i>System Requirements</i>	<i>4</i>
Software Requirements:	4
Hardware Requirements:	4
<i>Tools in Development</i>	<i>5</i>
Gantt Chart	5
Storyboard	6
Design Mock Ups	7
Home Screen	7
Main Game Screen	7
Pseudocode	8
Flowcharts	8
Source/Revision Control	8
<i>Basic Rules and Gameplay</i>	<i>9</i>
<i>Project Requirements</i>	<i>10</i>
Needs	10
Wants	10
Maybe	10
<i>Programming Report</i>	<i>11</i>
NWPi	11
The Project (Squares)	12
checkThreeInARow() method	12
generateGameBoard() method	13
<i>Project Evaluation</i>	<i>14</i>
<i>Problems Encountered</i>	<i>15</i>
<i>Logbook</i>	<i>16</i>
<i>Bibliography</i>	<i>19</i>
Pygame class reference	19
Basics - Learning pygame	19
Managing events	19

Introduction

About Squares

Squares is a basic game created using multiple different tools. Squares was initially drawn and designed on paper, along with game rules and other notes. Once rules and basic game designs were made final, the designs were drawn in photoshop, and coloured for later use and reference in the project. The use of a simple 'template' allowed the project scope to stay on track and ensured all features of the program were implemented.

About NWPi

NWPi is an objective framework developed individually by Nick Whyte for explicit use in this project. The game is heavily dependent on this framework. The framework will be available for use, modification, distribution and non-commercial use after the due date of this assessment. The project page for the framework can be found on Github, along with code examples from this project. <https://github.com/nickw444/NWPi>

System Requirements

Squares was developed on Mac OS using Python version 2.7.2 32Bit.

Software Requirements:

- Python 2.7.2
- Pygame 1.9 for Python 2.7.2
- Mac OS 10.7, 10.8, or Windows XP, Vista, Seven

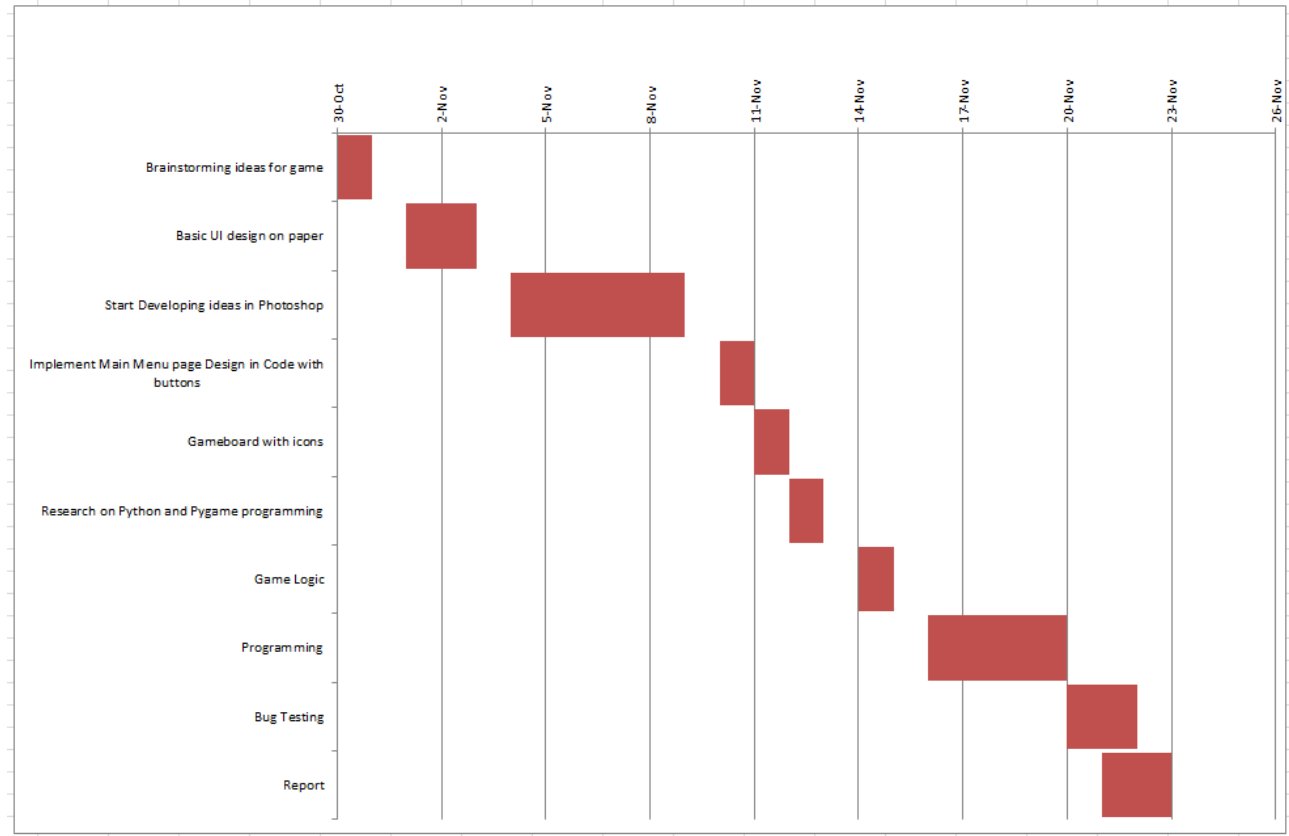
Hardware Requirements:

- 1.6Ghz Processor or better
- 1024MB RAM
- Monitor with a resolution greater than 800x600
- Mouse and Keyboard

Tools in Development

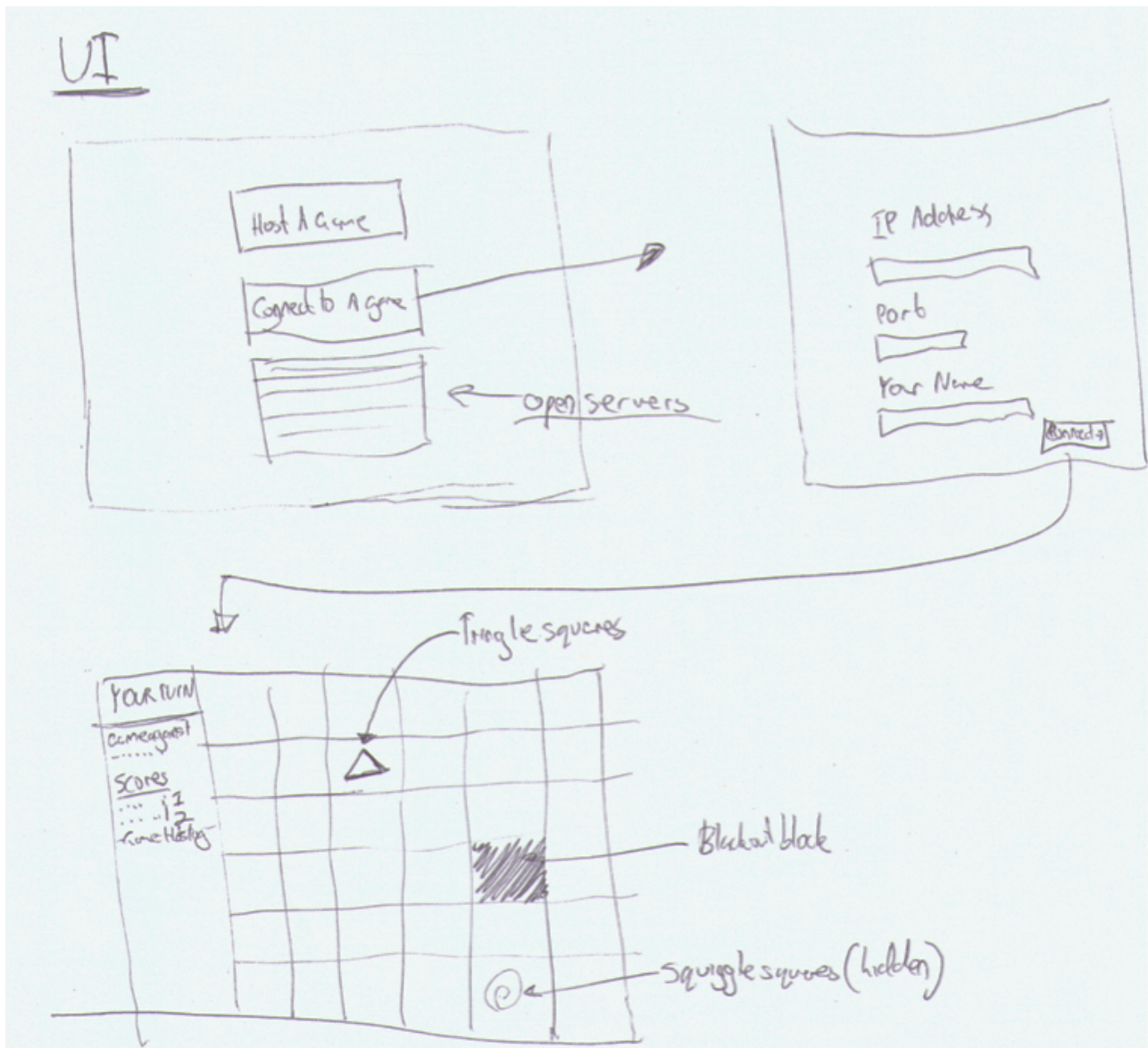
Various tools were used in the development of the project. Time management tools such as gantt charts were used, along with storyboards, design mock ups, and source/revision control. These are outlined in detail below.

Gantt Chart



Storyboard

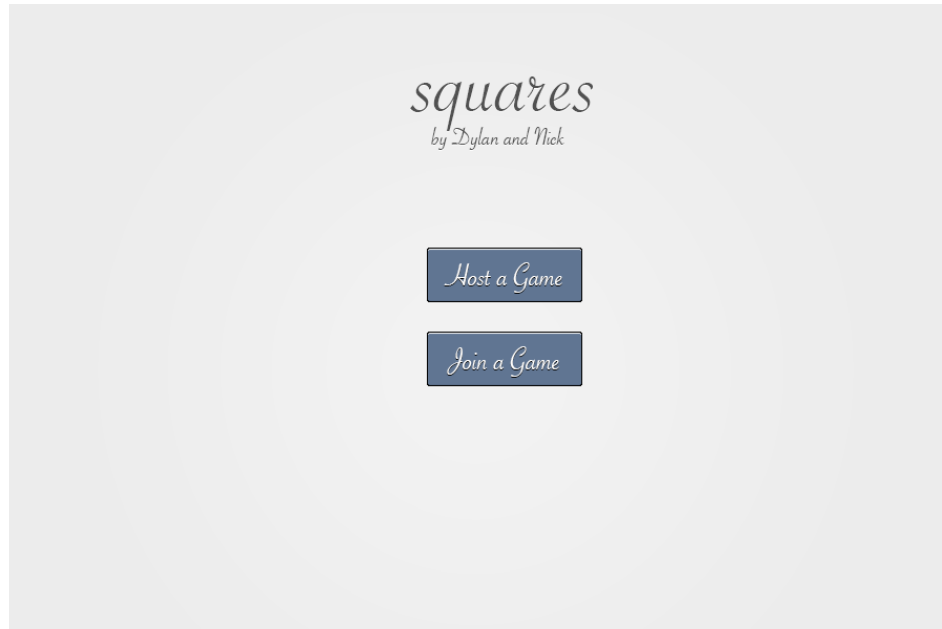
A basic drawn storyboard was created at the initial planning stage. This can be seen below:



Design Mock Ups

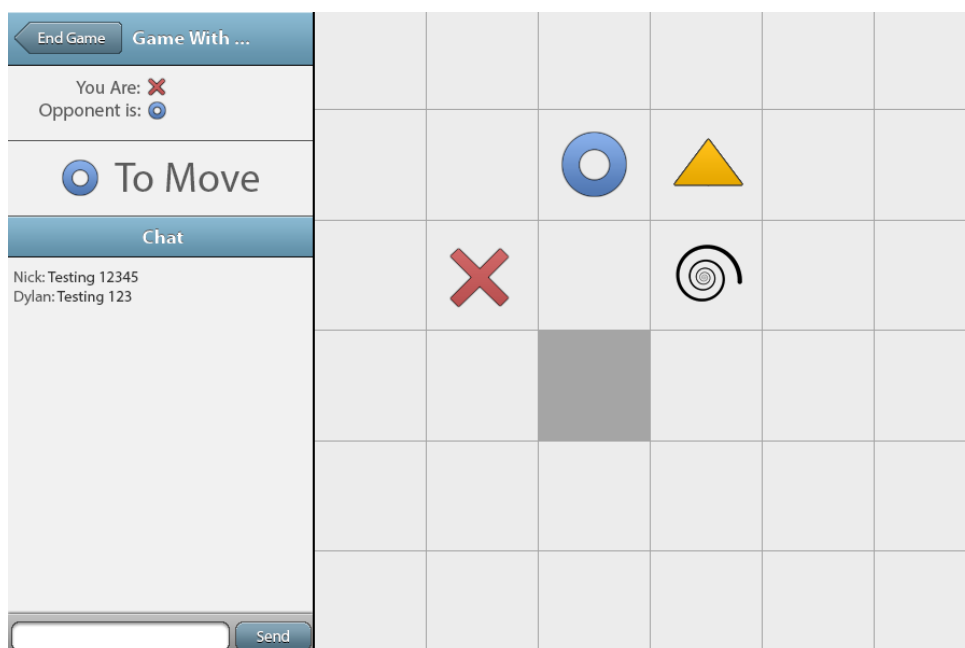
Basic designs were initially drawn by hand. These can be seen in the previous heading. Once drawings were made final, these were re-drawn in photoshop. See screenshots below:

Home Screen



This is the original home screen. It has almost been exactly copied into the game. Some elements, such as rounded edges were unachievable due to the nature of rendering objects in pygame.

Main Game Screen

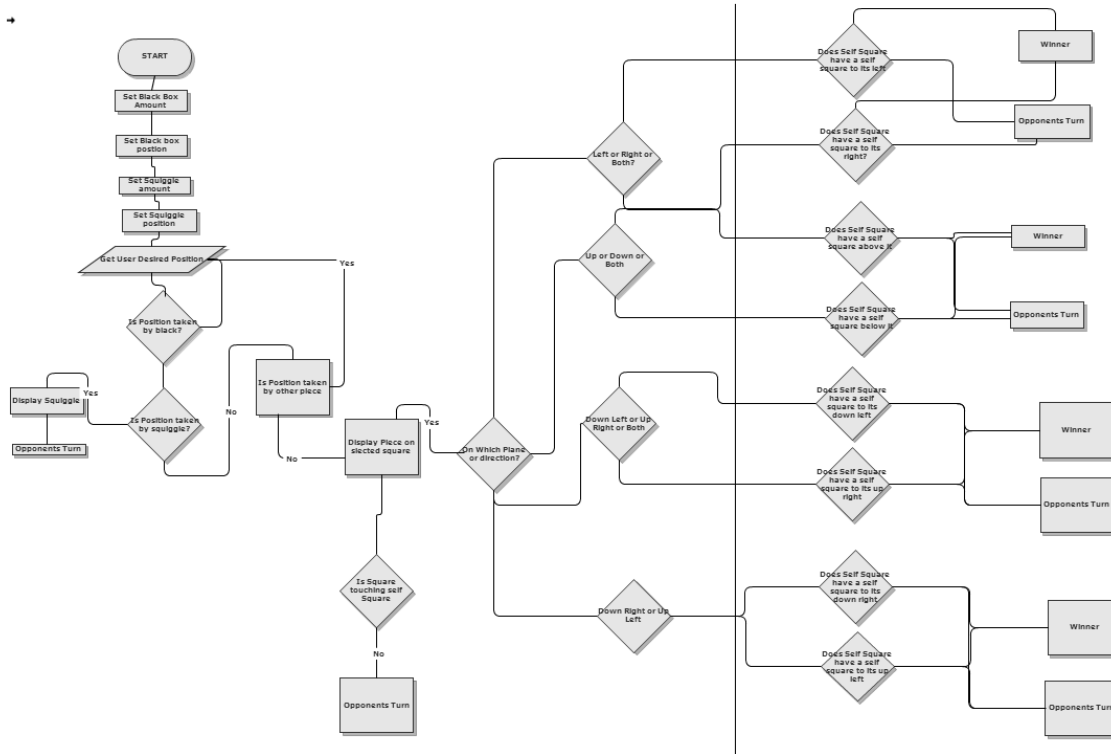


This is the original game screen. Many elements such as the X's and O's were extracted from here. Some features (such as network chat and networking) had to be removed due to the small timeframe for implementation.

Pseudocode

Pseudocode was used wherever possible when developing complex algorithms in the program. These complex algorithms are outlined in the following section.

Flowcharts



Source/Revision Control

A revision control platform called git was used for this project. The online web interface and hosting platform “github” was the git service used. This not only acted as a secure backup for all code modifications, but also allowed multiple programmers to collaborate, and contribute their work, and at the same time, complete a logbook of changes made.

Basic Rules and Gameplay

Squares is a Tic-Tac-Tow (or three in a row) game variant. The aim of the game is to place three of your own counters in a row on the game board. Squares is a two player game, and is played by the two players sharing the mouse and clicking the position they would like to place their counter.

In Squares, new counters have been introduced, along with a larger game board. The game board is now 6 x 6 in dimensions.

The two new counters/squares that have been introduced are:

Blackout Squares:

Blackout squares are simply a blocking square which will not allow any counter to be placed on top of. These make the game board a little more fun and interesting.

Squiggle Squares:

Squiggle Squares are placed randomly around the board, and remain hidden to the players of the game until they are clicked on. Once clicked, they will become shown, and now act as a Blackout Square. These do not count as a game piece. The player who clicked on this square has now forfeited his/her turn and should return the mouse to the other player.

Project Requirements

Needs

- Simple gameplay, quick games
- Easy to read, nice graphics
- New twists to the game
- Random generated game board
- 6 x 6 game board
- Different kinds of squares for the twist: Squiggle Squares, and Blackout Squares
- Squiggle squares will be invisible to the players, and will cause the player who clicks them to forfeit his/her turn.

Wants

- Network games

Maybe

- Changeable game board size

Programming Report

NWPI

The NWPI is an objective module for pygame and python. Developed from the ground upwards by Nick Whyte, it is a strong and stable solution. It features easy subclassing, simple object placement, and allows objects to be placed inside objects.

NWPI was inspired from the structure of Objective C and Cocoa/Foundation. It uses object names and methods similar to that of Foundation. Inspirations vary from the way subViews can be added to ANY view, to the way that every class is a subclass of the UIView class.

The overall goal for the creation of the NWPI framework was to allow easy screen rendering and object management. As a side effect of this, an easy way to implement “button up” actions - callbacks was created. Any UIView object or subclass can have a “callback” assigned, which will be run whenever a click is detected within. The use of callbacks within the game allowed buttons to work easily, with minimal programming effort.

NWPI also followed a “program once, use many” structure. Instead of re-writing similar code all around the framework, objects are subclassed from UIView, which by default implements a canvas for the object, adding of subViews/objects, and event/callback management.

In the overall project, NWPI simplified many areas of the program. From rendering objects to the screen to determining who won the game, NWPI plays a huge role in Squares

The Project (Squares)

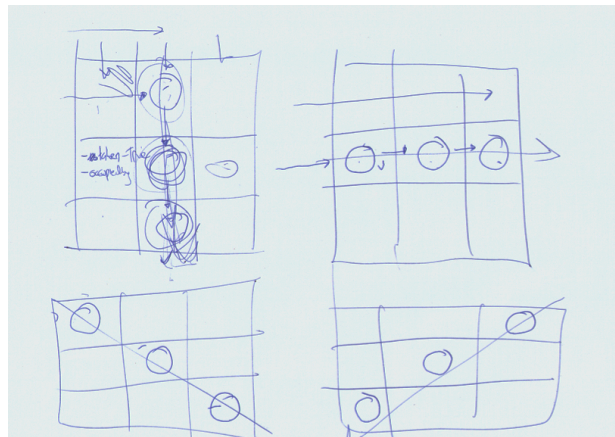
The overall programming in the project was a great success. There were many little issues found throughout the project, but with a bit of work, these were easily ironed out.

There are various algorithms in the program which were found to be of interest. Each of these algorithms will be explained below:

checkThreeInARow() method

The checkThreeInARow method can be found on the “mainGameViewController” class. This method is a complex checker to see if there are any three pieces in a row.

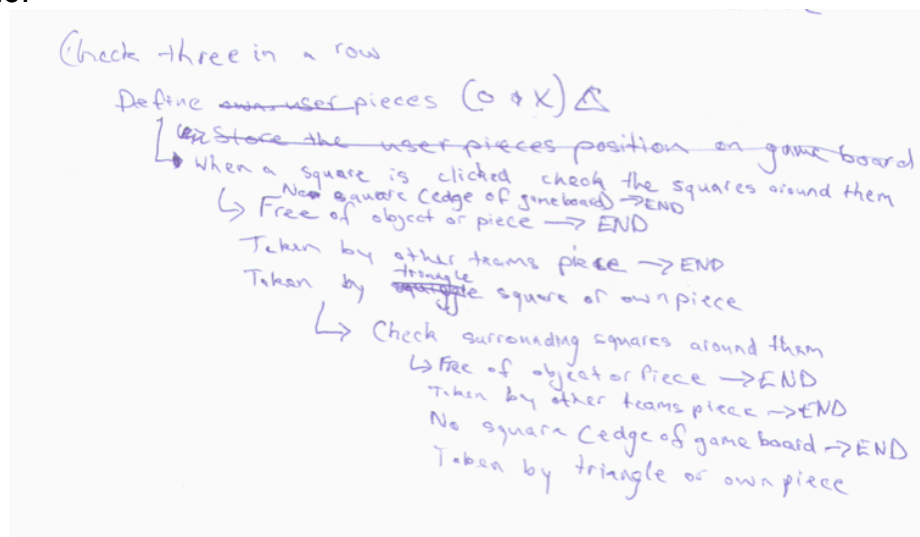
The checkThreeInARow method was initially designed on paper, using a diagram and pseudocode, as seen below:



The diagram showed there were three different types of three in a row. There could either be vertically downwards, horizontally right, diagonally down right, and diagonally down left.

In code, each of these conditions would be turned into their own separate if statement. Each statement would check the next corresponding square for a matching counter - if a matching counter was found, it would move to the next square, and so forth until 3 in a row were found.

Once a winner was found, a UIView would be shown to the user, displaying the winner of the game.

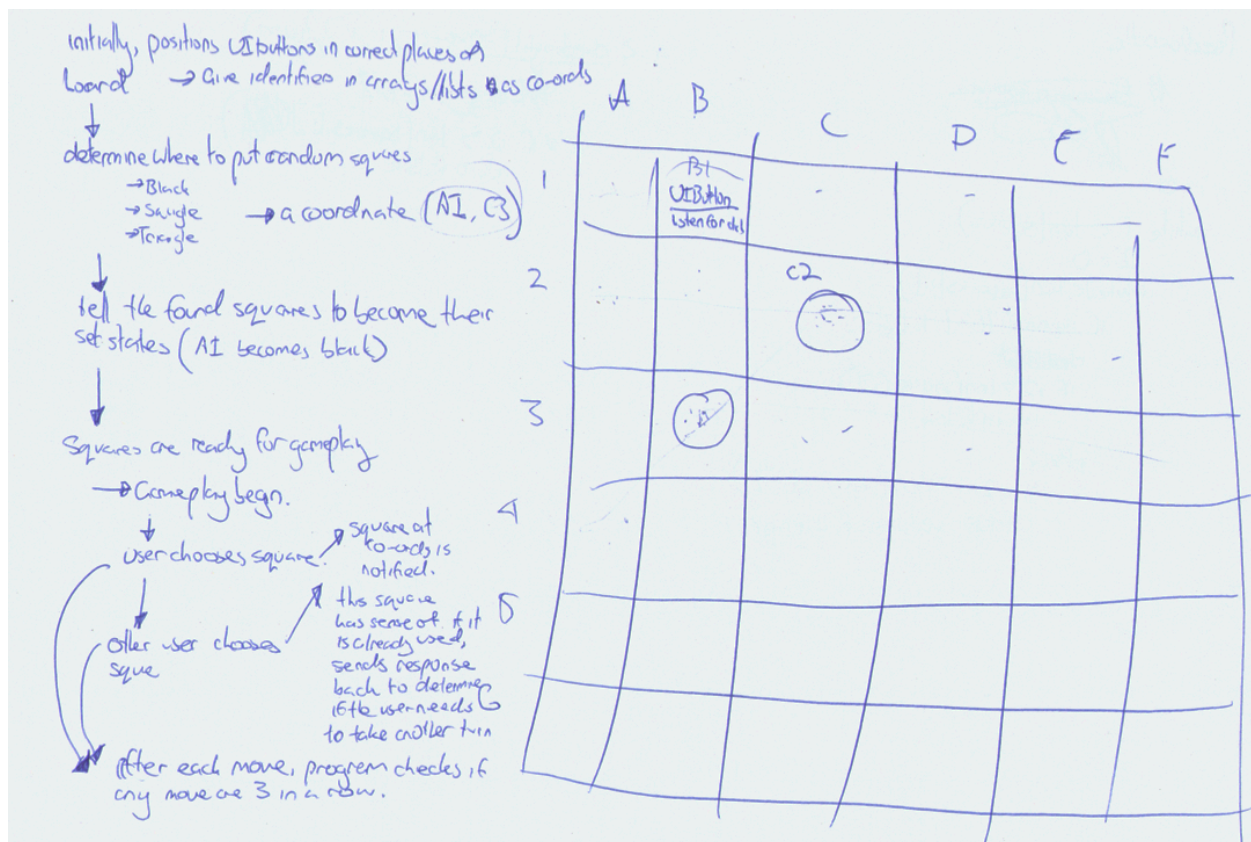


generateGameBoard() method

This is the method that generates the game board for gameplay. The method can be found on the “mainGameViewController” class.

Initially, some variables are set, which change the overall outcome of the game board. This includes the amount of rows and columns a game board should have. It was chosen that the board should be a 6x6 playing field through much testing.

This method is also where the randomization function of the program is found. The position of the “squiggle squares” and “blackout squares” randomized. Along with this the total number of squiggle squares and blackout squares is also generated randomly. Basic structure of the generateGameBoard method was initially written before programming took pace. See below:



The game board has a structure as follows:

```
row = [sq, sq, sq, sq, sq, sq]
gameBoardObjects = [row, row, row, row, row]
```

Each sq object is simply of the NWPi.UIView class. It holds details of who it is occupied by (if any). This is what the checkThreeInARow() method looks for.

As sq objects are created, they are checked if their desired co-ordinates are in the array for the “blackout squares” or the “squiggle squares”. Their retrospective properties and callbacks are added if either of these are found to be true.

Project Evaluation

Overall, the project was a great success, however some of the requirements were not fulfilled. Such features were not vital to the project. These included; network games and network chat.

Although these features were unable to be implemented, the game stuff functions correctly and gameplay is simply shared between the two players by swapping who holds the mouse.

The graphical interface of the program overall almost is an identical replica of that in the photoshop mock up. The overall user experience (UX), is of very high quality, as most elements of the UI are kept consistent for the user.

Problems Encountered

Various problems were encountered during development of the application. These issues ranged from trying to integrate network connections into the game, to trying to determine if there were three counters placed in a row

The main reason why network connections were not implemented was due to the small timeframe. There was simply not enough time to implement a working network game mode.

As with this issue, an AI engine was to be developed later down the track for player vs computer games, however, time did not permit for this development.

The method to determine if three counters were placed in a row initially was a hard task due to the nature of the triangle square. Soon-after, the triangle square was removed from the program, and this issue was resolved.

Logbook

NWPI / Commit History

Keyboard shortcuts available

Nov 21, 2012



Added a tonne of comments. Still not complete. Just a little more to ...

8f3364a564

...go!!!!

[Browse code](#)

Nick Whyte authored 9 hours ago



Removed `secondAlternativeView`

da72c78a96

Nick Whyte authored 9 hours ago

[Browse code](#)



Half cleaned out the project, added a tonne of comments

5fb94b4a25

Nick Whyte authored 9 hours ago

[Browse code](#)

Nov 20, 2012



Implementation of `UIAlertView`, half complete

3ba919c7d2

Nick Whyte authored 16 hours ago

[Browse code](#)



Modified the winning/losing engine. Added draw

709d3ee781

Nick Whyte authored 18 hours ago

[Browse code](#)



Changed the default font, also added the left indicator of who's turn...

... it is

bfa2dc9c8

Nick Whyte authored 18 hours ago

[Browse code](#)



Merged back to master

6ba5242765

Nick Whyte authored 20 hours ago

[Browse code](#)



`gitignore`

46c7b74c0e

Nick Whyte authored 20 hours ago

[Browse code](#)



Added `DS_store` to ignore

d6a2370431

Nick Whyte authored 20 hours ago

[Browse code](#)



Added `ds_store` to ignorables

ebd5cc8747

Nick Whyte authored 20 hours ago

[Browse code](#)



Game now knows who wins.

f11e05d63f

nickw444 authored a day ago

[Browse code](#)

Nov 18, 2012



Made things work. Squiggle Squares and Blackout squares now randomly ...

...generated

c55c771793

nickw444 authored 3 days ago

[Browse code](#)

Nov 14, 2012



Basic game logic

FML so much more to do kind of.

40aeeb39c2

nickw444 authored 7 days ago

[Browse code](#)



Began basic programming for checking if we have a winner

More tomorrow night/tomorrow period 3&4.

66244970a4

nickw444 authored 7 days ago

[Browse code](#)



Began implemation of `UIAlertView`

`UIAlertView` half complete, edited some comments, updated the main game view controller a little bit.

48e3318169

nickw444 authored 7 days ago

[Browse code](#)



Added methods for `UIView` to `setBackgroundimage`

Also worked the naughts and crosses into the program. Yay!

9fa38b19fe

nickw444 authored 7 days ago

[Browse code](#)



Re-classed `UIButton` (YAY)



















`UIButton` now is a subclass of `UIView`.










ce6f33206d

nickw444 authored 8 days ago

[Browse code](#)

Nov 13, 2012

	Added GameViewController implementation. ... Also edited view controller variable issue, background color was wrong nickw444 authored 8 days ago	4e6074bbed + Browse code ➔
	Fixed readme nickw444 authored 8 days ago	d95618e2da + Browse code ➔
	Fixed readme conflict nickw444 authored 8 days ago	6706e888cb + Browse code ➔
	Now ignoring log.txt nickw444 authored 8 days ago	851ed3a415 + Browse code ➔
	Now ignoring log.txt nickw444 authored 8 days ago	c4fd1401ac + Browse code ➔
	Merged back to master ... also we're now ignoring the log file. nickw444 authored 8 days ago	325f49eae8 + Browse code ➔
	Merged back to master nickw444 authored 8 days ago	6cd9f9ab2e + Browse code ➔
	Log nickw444 authored 8 days ago	38c3dca657 + Browse code ➔
	Fixed the callbacks nickw444 authored 8 days ago	5168fb7851 + Browse code ➔
	Fixed UIView callbacks. ... Still needs to be fixed to allow the parent to be called. nickw444 authored 8 days ago	b4eb394e61 + Browse code ➔
	Removed the DylanTest nickw444 authored 8 days ago	7d599d7ff8 + Browse code ➔
	Log nickw444 authored 8 days ago	4a06f61a09 + Browse code ➔
	Fixed the callbacks nickw444 authored 8 days ago	e3a77a0c73 + Browse code ➔
	Dylan is Amazing Dylan Martin authored 8 days ago	c2c32d944c + Browse code ➔
	Fixed UIView callbacks. ... Still needs to be fixed to allow the parent to be called. nickw444 authored 8 days ago	abd525736e + Browse code ➔
	Removed the DylanTest nickw444 authored 8 days ago	730bff4e63 + Browse code ➔
	Dylan test ... typed in 1234 Dylan Martin authored 8 days ago	6cb3e92c2d + Browse code ➔
	Project Modified nickw444 authored 8 days ago	1d2b52b7fe + Browse code ➔

Nov 12, 2012		
	Event management is still broken for subUIViews. ... Eh, I'll look at it tomorrow. nickw444 authored 9 days ago	8a46625386 + Browse code ➡
	Fixed UIView Top Level ... Top Level UIViews can have callbacks. Still need to fix sub UIViews nickw444 authored 9 days ago	1e3a92ac0b + Browse code ➡
	some ... Something happened inside the managing of UIViews and the way they interact with subViews. Needs some serious looking at. nickw444 authored 9 days ago	fddec6d385 + Browse code ➡
	UIView now has borders nickw444 authored 9 days ago	14a1cee58a + Browse code ➡
	Revert "Borders now available on UIButton" ... This reverts commit 5b61eca. nickw444 authored 9 days ago	ab6047118e + Browse code ➡
	Borders now available on UIButton nickw444 authored 9 days ago	5b61eca395 + Browse code ➡
	Updated Projects nickw444 authored 9 days ago	feed45d756 + Browse code ➡
	Initial Commit nickw444 authored 9 days ago	f8b44bf848 + Browse code ➡
	Initial commit nickw444 authored 9 days ago	1d9c149356 + Browse code ➡

This logbook can be found online at:
<https://github.com/nickw444/NWPi/commits/master>

Bibliography

Pygame class reference

<http://www.pygame.org/docs/>
<http://www.pygame.org/docs/ref/sprite.html>
<http://www.pygame.org/docs/ref/pygame.html>
<http://www.pygame.org/docs/ref/draw.html>
<http://www.pygame.org/docs/ref/rect.html>
<http://www.pygame.org/docs/ref/font.html>
<http://www.pygame.org/docs/ref/image.html>

Basics - Learning pygame

<http://www.devshed.com/c/a/Python/PyGame-for-Game-Development-Font-and-Sprites/>
<http://www.linuxjournal.com/article/7694>

Managing events

<http://stackoverflow.com/questions/6775897/pygame-making-a-sprite-face-the-mouse>
<http://stackoverflow.com/questions/380420/how-do-i-respond-to-mouse-clicks-on-sprites-in-pygame>
<http://stackoverflow.com/questions/6356840/how-to-detect-if-the-sprite-has-been-clicked-in-pygame>
<http://floppsie.comp.glam.ac.uk/Glamorgan/gaius/games/8.html>
<http://www.daniweb.com/software-development/python/threads/299664/buttons-help>
<http://stackoverflow.com/questions/380420/how-do-i-respond-to-mouse-clicks-on-sprites-in-pygame>
<http://stackoverflow.com/questions/6775897/pygame-making-a-sprite-face-the-mouse>