A-Level project document

Contents

[1. Analysis 3](#_Toc87656768)

[1.1. Problem Identification and Background 3](#_Toc87656769)

[1.2. Objectives 3](#_Toc87656770)

[1.3. Research 3](#_Toc87656771)

[1.3.1. Surveys 3](#_Toc87656772)

[1.3.2. Skills research 3](#_Toc87656773)

[1.3.3. Research Table 4](#_Toc87656774)

[1.4. Story board 4](#_Toc87656775)

[1.5. Prototypes 5](#_Toc87656776)

[1.6. Justification 7](#_Toc87656777)

[1.7. Data Modelling 8](#_Toc87656778)

[1.7.1. Identification of objects 8](#_Toc87656779)

[1.7.2. Analysis Data Dictionary 15](#_Toc87656780)

[1.7.3. Data flow diagrams 15](#_Toc87656781)

[2. Design 15](#_Toc87656782)

[2.1. Design Introduction 15](#_Toc87656783)

[2.1.1. Structure chart 15](#_Toc87656784)

[2.1.2. Nonstandard Algorithms using Code or pseudo code 15](#_Toc87656785)

[2.1.2.1. Thread implementation and map loading 15](#_Toc87656786)

[2.1.2.2. Creating the JFrame 17](#_Toc87656787)

[2.1.2.3. Rendering method 20](#_Toc87656788)

[2.1.2.4. Program functionality 22](#_Toc87656789)

[2.1.2.5. Switching to JavaFX 27](#_Toc87656790)

[2.1.2.6. Graphics improvement and lighting system prototype 37](#_Toc87656791)

[2.1.2.7. Keylistener implementation 41](#_Toc87656792)

[2.1.2.8. Main menu development 43](#_Toc87656793)

[2.1.2.9. Button animations 45](#_Toc87656794)

[2.1.2.10. Map select menu development 50](#_Toc87656795)

[2.1.2.11. Reintroducing screenshotting 54](#_Toc87656796)

[2.1.2.12. Creation of map list 55](#_Toc87656797)

[2.1.2.13. Switching from text files to binary files 59](#_Toc87656798)

[2.1.2.14. ‘Object’ object and light object 60](#_Toc87656799)

[2.1.2.15. Avoiding loading a null map 63](#_Toc87656800)

[2.1.2.16. Rendering issue solutions 64](#_Toc87656801)

[2.1.2.17. Rotation mechanic 66](#_Toc87656802)

[2.1.3. Object diagrams and class definitions 72](#_Toc87656803)

[2.1.4. Prototyping 72](#_Toc87656804)

[2.1.5. Design data dictionary and data structures 72](#_Toc87656805)

[2.1.6. File organisation 72](#_Toc87656806)

[2.1.7. E-R model and diagrams 72](#_Toc87656807)

[2.1.8. Human-Computer-interface design 72](#_Toc87656808)

[References 72](#_Toc87656809)

# 1. Analysis

## 1.1. Problem Identification and Background

Inspired by a popular sandbox game called ‘Garry’s Mod’, I aim to create a three-dimensional space in which the player can roam freely around it and interact with objects. Said objects, and the player, will be under the effects of a physics engine, so they will fall if unsupported, slow down from friction, etc. The user can choose to play on different worlds, or maps. Each map will be stored in separate files.

## 1.2. Objectives

1. The user must have independent movement.
   1. The user cannot move inside of an object or shape via use of a collision system.
   2. If the user is not standing on an object, they will fall at an increasing rate.
   3. Jumping must be possible.
   4. Looking around is controlled via mouse movement.
2. The program must have a clear and appealing menu, including:
   1. A title
   2. Multiple choices such as map select, options, etc
   3. A way to exit the game
   4. Map selection
   5. The menu accessible while playing the game
   6. A reset button in the in-game menu reloading the map when used
   7. A respawn button to simply respawn they player in case they get stuck
3. A physics-like mechanic should be present.
   1. If an object is suspended mid-air, it will fall like the player.
   2. If an object is on the ground, but in an unstable position (e.g a cube with one corner on the ground), the object will move to a stable position.
   3. If force is applied to an object, either from the player or another object, the object will move but will slow down depending on friction.
   4. The player will also be affected by force from another object.
4. A lighting system should also be present.
   1. A light entity will mark the coordinates for a source of light.
   2. The closer a shape is to a light source, the brighter its colour.
   3. A shape may also be lit when there is no shape above it.
   4. An object above a shape may dim the shape below slightly.

## 1.3. Research

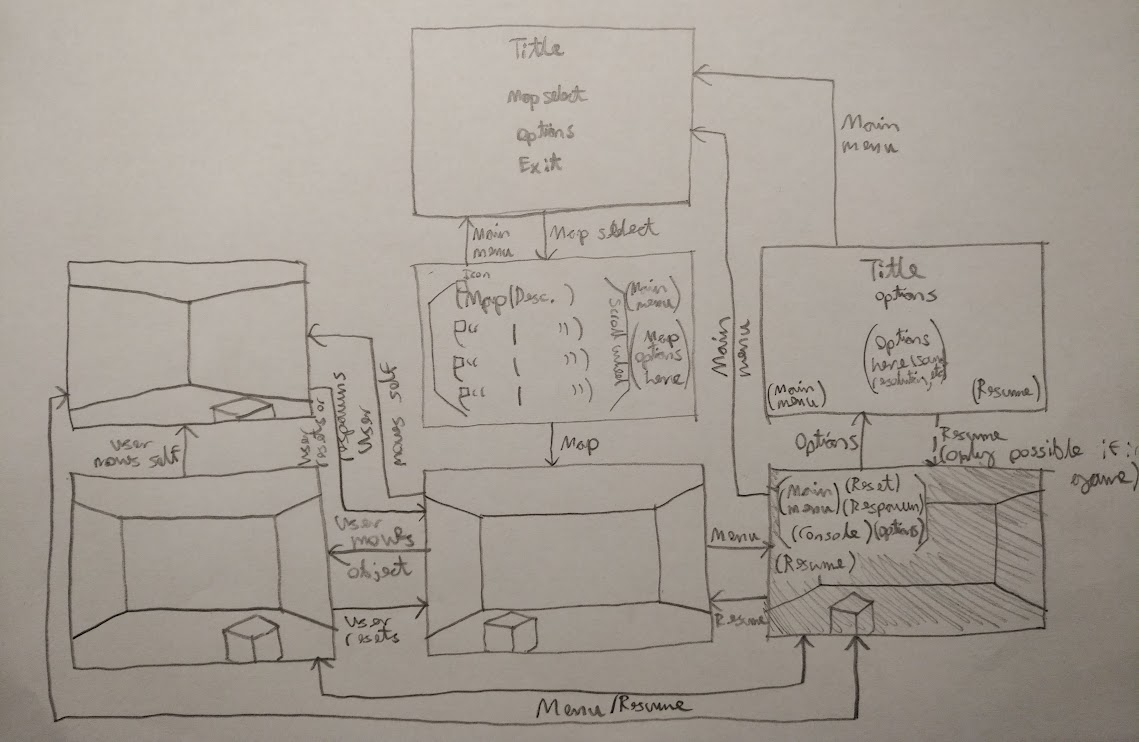
### 1.3.1. Surveys

### 1.3.2. Skills research

### 1.3.3. Research Table

|  |  |  |  |
| --- | --- | --- | --- |
| Research |  |  |  |
| Type | Date | Detail | Complete |
| Research into other games | 01/11/2021 | My game includes various aspects from another sandbox game called ‘Garry’s Mod’. I have researched this game thoroughly and the properties of the game I would like to incorporate in my game are the physics mechanic that makes objects and the player react to the environment and each other, the map feature where several maps could be chosen and played on individually with no game progression, and the open world aspect where the player is free to roam and can move objects as they wish. On the other hand, I would not involve such complexity or some specific features of the physics engine such as terminal velocity. |  |
| User feedback 1 | 01/11/2021 | User suggested pre-placement of objects stored in map file and the ability to modify world geometry. |  |
| User feedback 2 | 04/11/2021 | Initial colour of start button in map select menu was green, user suggested purple instead for variation. | ✓ |

## 1.4. Story board



Exit

The story board of my game depicts 7 possible general actions the user can perform. From the main menu, they can either proceed to the map selection screen, the options screen, or exit. In the options screen the user can change how the program behaves such as sound (if any) or resolution. From the map select screen, the user can pick the map they would like to go on, configure map options is necessary, and start the game. Once in the map, the user can move around, move objects, or pause the game. In the pause menu, the game will darken in tone and freeze while the menu offers the options to return to the main menu, reset the map, respawn the player, open the console, or go to the options screen. In this other version of the options screen, the user can either still return to the main menu, or resume the game, as well as configure options of course.

## 1.5. Prototypes

When designing how my project will look and behave, I planned to, as a prototype, have the simulation run in a JPane with a border, with controls in another JPane with a border, all with a simple appealing colour scheme.

Background pattern

Description automatically generated with medium confidence

A picture containing graphical user interface

Description automatically generatedAfter some experimentation, I began to fully move to using JavaFX rather than JFrame, because JFrame is very difficult to make use of a KeyboardListener, the key to the keyboard controls.

While JFrame has a Graphics utility, JavaFX also has one, and since at this point I was using JavaFX, I swapped for this rather than changing each pixel in a BufferedImage and then displaying it to the user.

Diagram

Description automatically generated with medium confidence

When drawing the shape, my code was drawing each of the four lines individually, and therefore the shape was just an outline. GraphicsContext has a useful method drawPolygonFill, where given an array of x-coordinates, an array of y-coordinates, and a colour to use, a filled polygon is drawn using said points and colour for the fill. Therefore, this was used over drawing the lines individually to solve the missing fill problem, as well as optimise the code further.

A picture containing diagram

Description automatically generated

## 1.6. Justification

In creating my program, I am writing it to be used on a Window PC. The PC would need to run Windows version 10 and Java version 8. I have written my game in Java 8 because I can access JavaFX tools, and this allows me to more easily create shapes and draw graphics on screen so they can be part of the game rather than my initial plan of modifying individual pixels in an image variable.

## 1.7. Data Modelling

### 1.7.1. Identification of objects

Text

Description automatically generated with low confidenceThe object ‘Quad’ contains the displayed attributes and behaviour. ID is unique to every shape and is assigned when the map is read in order, starting from 0. ‘coordinatesInWorld’ stores the coordinates of each point of the shape in three dimensions, which comes directly from the map file. ‘pointsOnScreen’ is set to null when the map is read, and the method ‘setPointsScreen’ is used in rendering to set the values. This array then stores the coordinates of each point of the shape as it appears to the user on the canvas. For example, a square on the floor in front of the user will appear as a two-dimensional trapezium on the canvas because of how the points are translated to the screen. ‘centreOfQuad’ is also calculated when the rendering commences and is initially set to null. It is a single point situated in the geometric centre of the shape. This will be used to calculate the distance to the user’s viewpoint, variable name ‘distanceToCamera’, and used as an element when calculating the lighting. ‘centreOfQuadOnScreen’ is again where the centre of the shape appears on the canvas. This object holds the details of each and every shape in the map file, all of which are stored in an ArrayList named ‘map’, each ‘Quad’ object being declared when loading the map.

A picture containing text

Description automatically generated

Text, application

Description automatically generated

Text, timeline

Description automatically generated with medium confidence

The ‘CustomImage’ object is used for the transitions of image-using menu elements, such as buttons. This object is the largest in terms of behaviour and attributes combined. Each ‘CustomImage’ instance stores its own:

* Image
* imageView of said image
* two variables to keep track of how much the image should be translated by
* one variable for moving on and off screen and the other for the hover-over animation
* how many pixels it should be translated by to the right (if any)
* four variables to keep track of moving on and off screen, hover-over animation, moving right animation, and fade in/out animation in order
* distance to travel on screen
* distance to travel to the right
* regular translation on the y-axis
* opacity
* transitioning boolean
* moving boolean

All the methods are rather similar in which they use the sine curve to move smoothly and not just linearly. Every time one of the methods is called, the corresponding angle is incremented or decremented by a global variable’s value and the image is modified appropriately. Corresponding booleans also are set; if a method is called, the image is checked to be in the right place. If not, the respective boolean is true, if so, then it is set to false. All these methods are called in the Timeline under specific circumstances, like if the map select screen is up, and the main menu elements have moved off, then the map select screen elements can move on screen.

Graphical user interface, text, application

Description automatically generated

A picture containing text

Description automatically generated

The ‘CustomText’ object has almost identical behaviour to the ‘CustomImage’ object, therefore inheritance may possibly be used. This object only has two methods which transition the text on and off screen, since this object is current only used for the title, and uses far less attributes:

* the string the text will display
* the actual Text object initialised with the string variable
* the linearly increasing ‘theta’ variable which, like before, is used for a sine curve for a smooth animation
* the distance the text should travel
* whether or not it is moving on or off screen

Table

Description automatically generated with medium confidence

Text

Description automatically generated with medium confidenceA picture containing text

Description automatically generated

Text

Description automatically generatedThis last object, named ‘ListMap’, is for listing the maps in the map select screen, and therefore contains many elements all held within a Group. It has a long constructor because it positions and formats the elements of the group to visually appeal. Its attributes are:

* the title of the map
* the description of the map (stored as an array since there are two lines)
* the text objects for the title and both description lines
* the base image used in which all elements will be placed atop
* the base image green coloured to indicate this map is selected
* the map icon
* the imageViews for the base images and icon
* the theta variables to again be used with the sine curve
* the group’s opacity
* whether or not the group is fading in or out
* the actual Group object
* the number of pixels the group is translated by due to the hover-over animation
* if the map is selected or not.

The group as an animation to move on and off screen, fades in and out rather than moves since the background it is placed on fades in and out too. The sine curve is still used so the fading animation is smooth too.

### 1.7.2. Analysis Data Dictionary

### 1.7.3. Data flow diagrams

# 2. Design

## 2.1. Design Introduction

### 2.1.1. Structure chart

### 2.1.2. Nonstandard Algorithms using Code or pseudo code

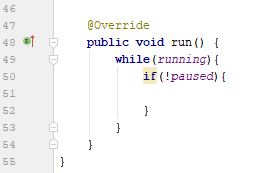
#### 2.1.2.1. Thread implementation and map loading

To begin, I first utilise multithreading, as my initial plan is to have the Main method initialise the things that need to be so, and a different class can oversee running the calculations.

To also make sure that the Main method doesn’t stop after one iteration, I add a ‘running’ boolean and a while loop using this boolean instead of a while(true) loop to add a stopping functionality. The ‘paused’ boolean is simply for later use to pause the program, adding to the functionality.







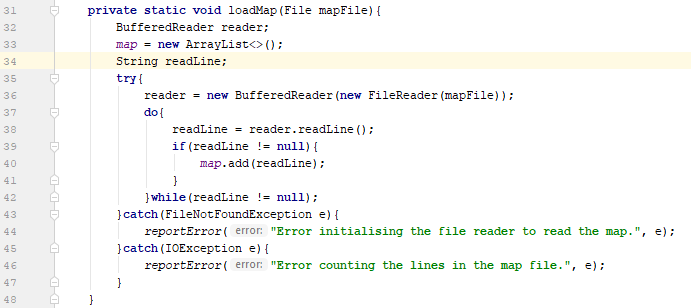
A method to report errors is added, and made public in order for other threads to access it. It receives a string and the exception and reports the error in a visually appealing way using ANSI colour codes (source of codes: <https://stackoverflow.com/questions/5762491/how-to-print-color-in-console-using-system-out-println>).



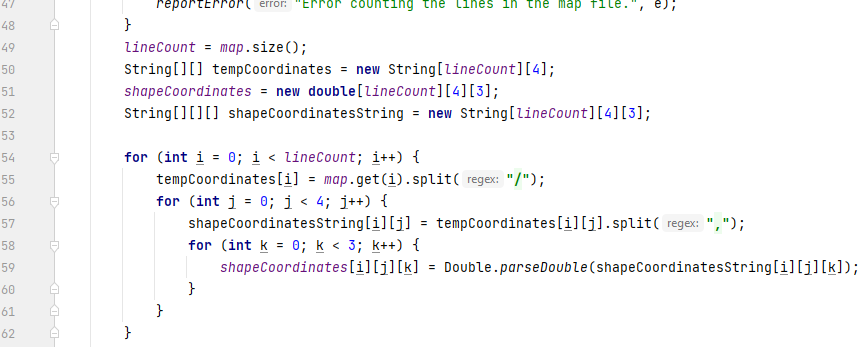
An ArrayList is used to store the map details to save having a separate method to count the lines in the file for a static array to use, and to make accessing data more versatile.



The method to load the map is then created with the map file parameter. After initialising the map ArrayList, the first line is read to a temporary string, which is then copied to the map ArrayList if it is not null. If it is null, the ‘if’ statement fails and the loop is ended.



To organise the data just collected, each line in the map ArrayList is split into its elements and stored in an array, which in turn is split too into another array, then the final array is converted from a String[][][] into a double[][][] to get the individual coordinates into a three-dimensional array using nested ‘for’ loops.



#### 2.1.2.2. Creating the JFrame

Next thing I did was extend the JFrame class to allow for constructing the window. However, JavaFX may be able to be used in place later to make designing easier.



Text

Description automatically generatedThe window must be created, along with a JPanel containing a BufferedImage acting as the screen, which can be edited and updated for every calculation. The easiest way to add an image to a JPanel is via an ImageIcon and JLabel, which is what is used here. The ‘view’ BufferedImage is initialised as a global variable.

Graphical user interface, text, application

Description automatically generatedA screenshot of a computer

Description automatically generatedText

Description automatically generatedJust to strain the eyes less and to, once again, visually appeal, a red border, grey background, and title are added to the JPanel containing the view image. The JPanel is then added to the JFrame, which in turn has its size set to a little over the image dimensions, default close operation to exit on close, and to be visible. Finally, the whole subroutine is to be called in the main method.

The next step is the maths and logic. To begin I created a separate class, so a few threads of this class can run parallel to the Main class, one doing calculations and the other updating the image.

A picture containing timeline

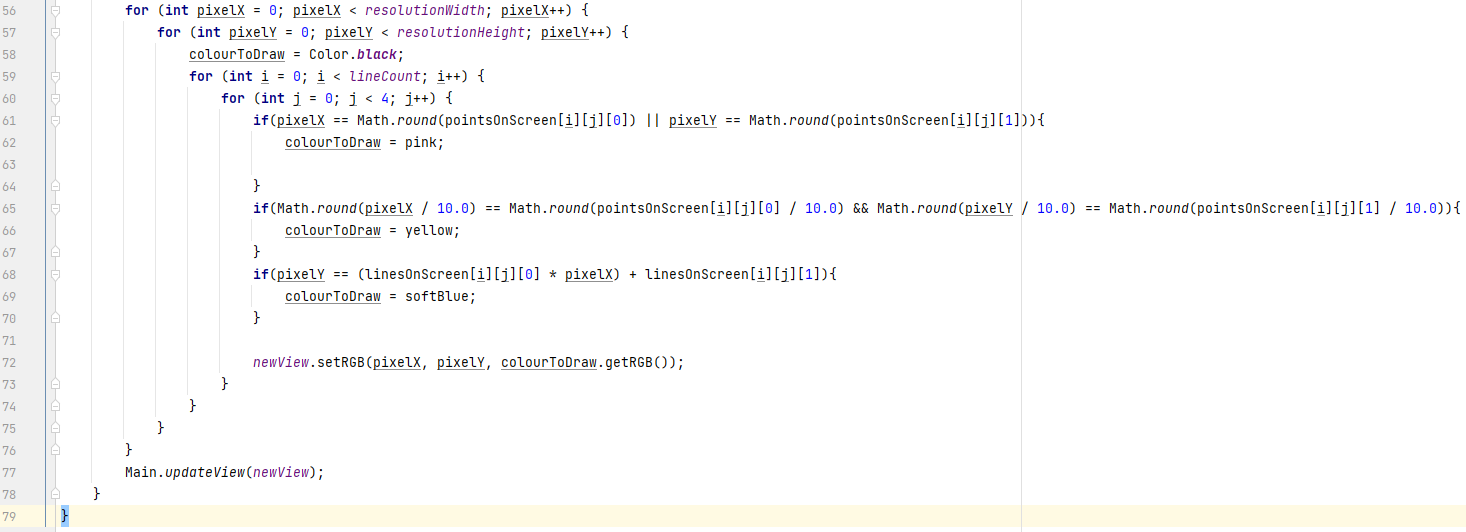
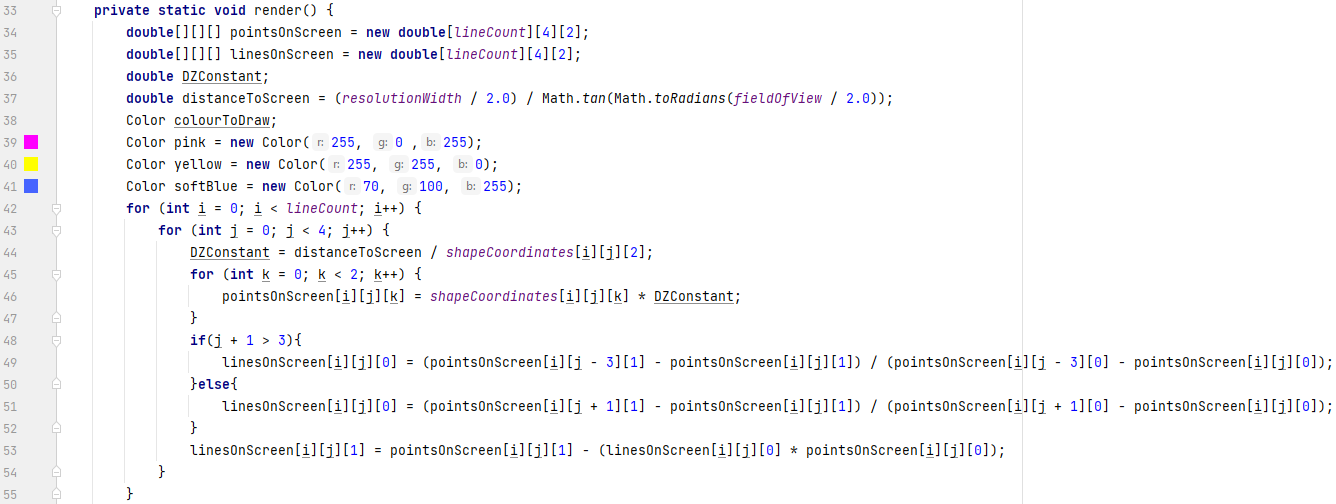
Description automatically generated

Graphical user interface, text, application

Description automatically generatedNow the methods to continually call the rendering method, the rendering method itself, and the method to fetch the data are complete, arithmetic is ready to be used.

#### 2.1.2.3. Rendering method

This code involves the first mathematical aspects. To start, two arrays are created, one to hold the points as they appear on the image, and the other to hold the line equations between said points. Doubles are created to hold important variables to the calculations, including the distance to the screen in the world. Colours are next; the colour that will be drawn is first set to black, as well as pink, yellow, and blue to be used later. Two ‘for’ loops are then started: one to account for every shape in the map and the other to account for each point the shape has, typically 4. 3D Points in the array are projected to the screen by multiplying the x and y coordinates by the z coordinate divided by the distance to the screen. Hence the lines are also made immediately after by using the next point as well as the current point to calculate the gradient. If the next point is intended to actually be the first point due to the lines connecting the first and last points, then an ‘ArrayIndexOutOfBoundsException’ is avoided by checking if the exception would happen, and instead using the first point. Note this method uses the assumption that the points were not inputted irregularly, or the lines would cross across the shape. This is likely to be fixed later. Now the points and lines are made, the ‘for’ loops are ended, and each individual pixel in the image is checked if it matches a point on the x axis or the y axis, if its roughly near where the point is, and if it lies on one of the lines. If so, the variable controlling the current colour is changed accordingly and the image within this thread is edited for that pixel. Once the entire frame is drawn, it is finally returned to the Main class to be used to update the view.



The result of this ordeal is the lines and points indeed being highlighted in the space, but the centre of the screen is set to the top left; a problem that needs to be resolved.

A screenshot of a computer

Description automatically generated with medium confidenceBackground pattern

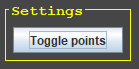
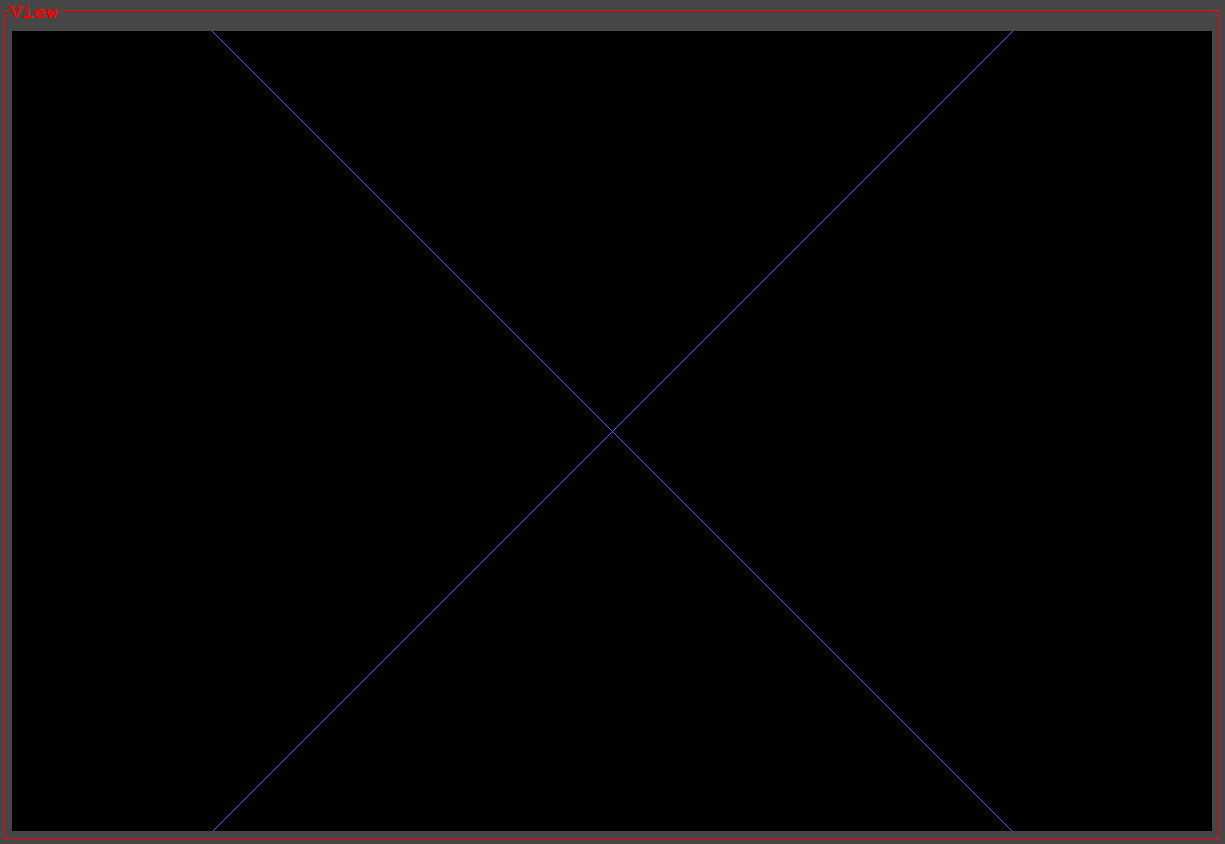
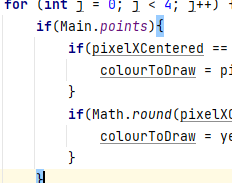
Description automatically generatedA solution to the problem is found by creating two more variables to be used by the ‘if’ statements so the centre of the screen is now in the actual centre. The points were moved to fit in the frame.

#### 2.1.2.4. Program functionality

In order to see if the blue lines had been drawn correctly horizontally, a newJPanel containing an option to disable or enable the point markers should be created. It would also hold future settings.

A new public boolean called ‘points’ is created so it can be used by the rendering class. FlowLayout is set to be used by the JFrame, as it is the simplest layout and without one, only one JPanel can occupy the entire window at once. The ‘points’ boolean is used in an ‘if’ statement containing the code to draw the points on screen.

The result reveals that not all the blue lines were correctly drawn as the horizontal ones are absent.



A picture containing text

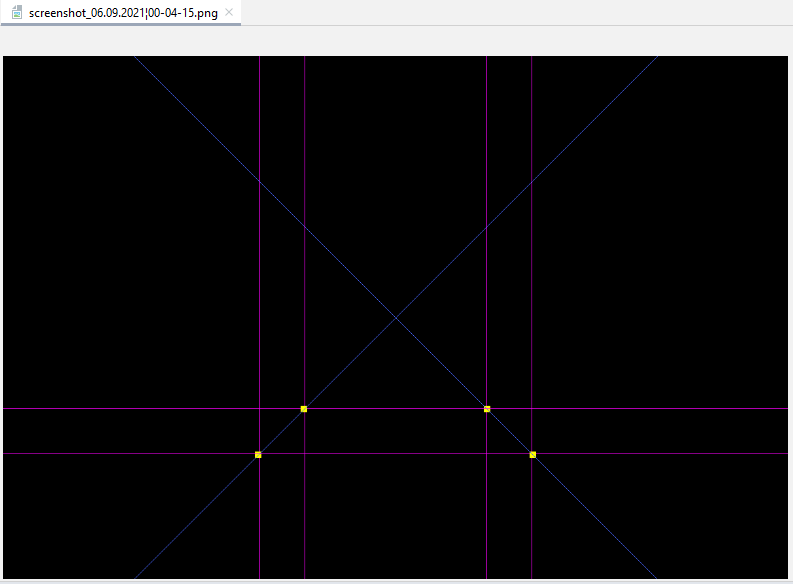
Description automatically generatedFor convenience and further documentation, a small method to take a screenshot of the frame is created. The date and time of creation is recorded in the image name and all screenshots are kept in a folder in the project (source: <https://stackoverflow.com/questions/12674064/how-to-save-a-bufferedimage-as-a-file>).

To trigger the action, a screenshot button is added to the settings panel and is coloured to stand out from others.

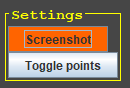
Text, letter

Description automatically generated

Using the screenshot function prints this message in the console and saves this image (please excuse the absurd time it was taken).



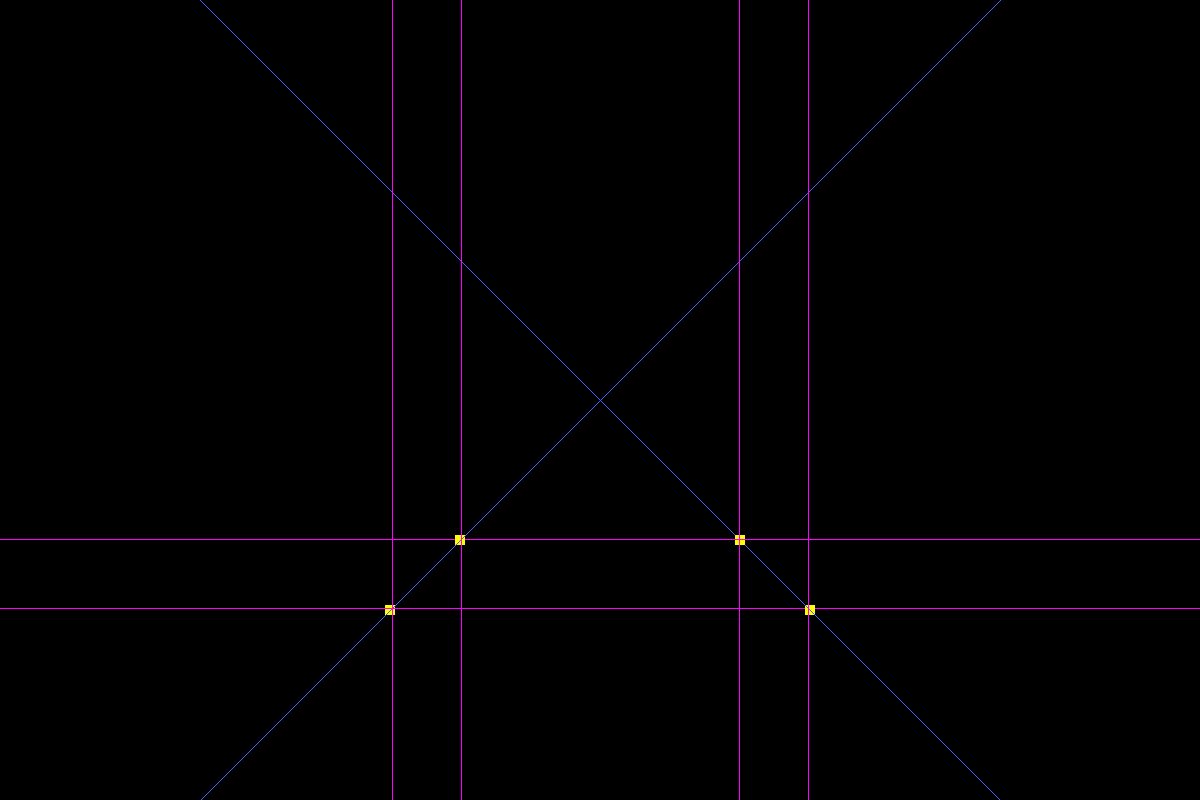
Adding the screenshot button to the settings panel creates the problem that each button is placed side by side instead of in a column like intended. A BoxLayout was simply used (source: <https://docs.oracle.com/javase/tutorial/uiswing/layout/box.html>) to align the buttons along the y-axis. However, there are no gaps between the elements and a different layout is bound to be used eventually.



Background pattern

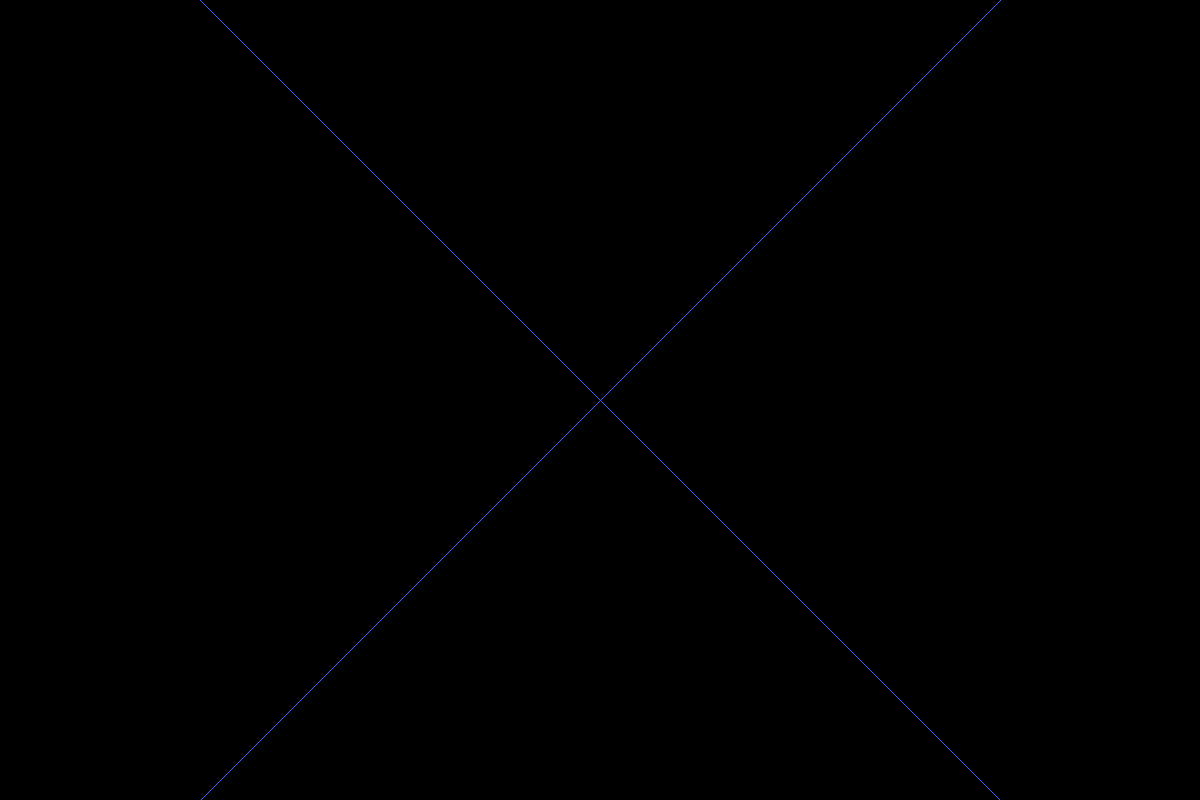
Description automatically generated with medium confidenceBy now, the project looks like so.

I was curious as to if the simulation could support multiple squares, although it is unlikely. If not, a solution could be found. The coordinates I entered were intended to draw the square 4 units above, but this did not appear.



Shape, rectangle

Description automatically generatedRemoving the first square, it turns out it was out of the frame, meaning the y-axis is the wrong way.

A picture containing dark, light

Description automatically generatedChanging the y-coordinates of the square’s points from 6 to -2 shows that, surprisingly, the code can support multiple squares, but the blue lines still need to all be drawn correctly, and the y-axis corrected.

To tackle the inverted y-axis issue, debugger comes in useful. Setting the line to set the pixel colour to yellow as a breakpoint, we can see the y coordinate of the top left point. Top left, because the program scans the frame from the top left corner to the bottom right corner due to how the ‘for’ loops are set up. The y-coordinate of this point was negative, implying that the cause of the problem at hand was the order in which it scans the frame.

After problem-solving for a long time, the square could actually be behind the point of view as it still would be projected onto the screen if it was, only inverted. The z-coordinates were changed to negative to draw the square correctly, however a movement mechanic would have to be fitted to confirm it is in the right place.

At first I tried using a KeyListener, but this only tracks the last key pressed, and it was unresponsive anyway.

Graphical user interface, text, application

Description automatically generated

#### 2.1.2.5. Switching to JavaFX

Finally, after some experimentation, threads and JavaFX were working side by side. JavaFX could be key to many problems that would’ve been encountered. The method that constructs the JFrame is not redundant, and a new GUI can now be constructed, as well as keys and mouse movement Graphical user interface, text, application

Description automatically generatedhopefully now being picked up. As shown here, the main thread is only started after the JavaFX stage is given a chance to be created.

Graphical user interface, application, Word

Description automatically generatedUnfortunately, a way to successfully update the image could not be found. Using a JFXPanel with awt Graphics was looked at (source: <https://stackoverflow.com/questions/1965347/how-do-i-initialize-a-graphics-object-in-java>), but this only provided support for FX elements in a JFrame, not the other way round. However, another way is to use GraphicsContext, after discovering it is the FX equivalent of awt Graphics. As a benefit, it could greatly optimise drawing the lines, as instead of drawing directly to a BufferedImage itself, the methods to draw lines using said GraphicsContext could be used instead (source: <https://docs.oracle.com/javafx/2/canvas/jfxpub-canvas.htm>).

This test of the GraphicsContext and canvas basically only involves drawing a single red line from the top left corner diagonally right-down, which turned out as the case.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

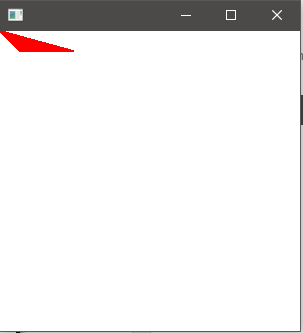
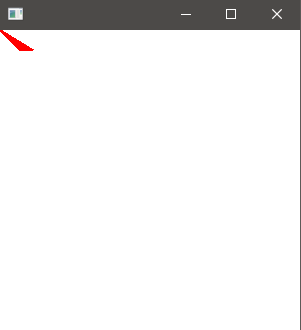
Description automatically generatedNow, to avoid coding the rendering using the GraphicsContext all for nothing, it should be tested if it is able to update first. Sadly, the initial updating problem is still present, but after a lot of searching, I found that JavaFX works terribly with Threads, and an alternative of ‘Tasks’ exist. Creating a Task with a loop and starting it as a thread alongside the Application, it works out.

Text

Description automatically generated with medium confidenceTesting the problem was difficult, but in the end, I got a clear indication that it had finally worked. The line is not cleared from the canvas, and instead drawn repeatedly further to the right each time before eventually going off screen, creating a triangle.

Graphical user interface, text

Description automatically generatedNow a reliable drawing method is finally in place, so its time to adapt the current rendering method to it. Although the GraphicsContext method is better and the way to go, much of the code to calculate and draw on the screen unfortunately goes to waste but is kept in the rendering class in case it is needed, and the rendering class is redundant after moving the needed code to the main class anyway. The graphics test method is renamed appropriately, and the old render method adapted too.



Finishing off the draw method, in theory it should work just like before, only with a white background. The same “jPlus1” method is used to use the next point but not to cause an ArrayOutOfBoundsException.

Graphical user interface, text, application, email

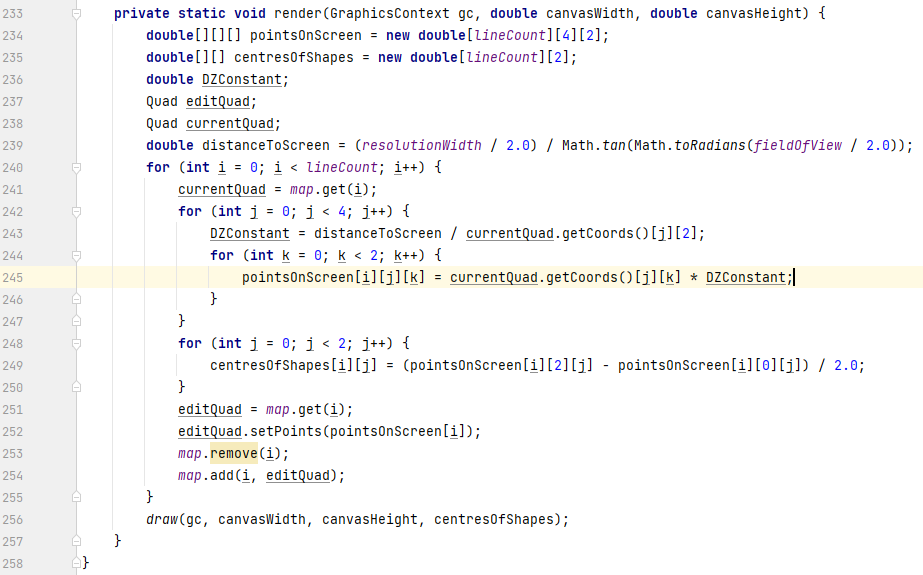
Description automatically generatedWhen the program is run, lines flash on screen, quickly appearing then disappearing, and then after a second or so it stops. This is problematic, especially since results are different each time, and sometimes even results in a BufferOverflowException.

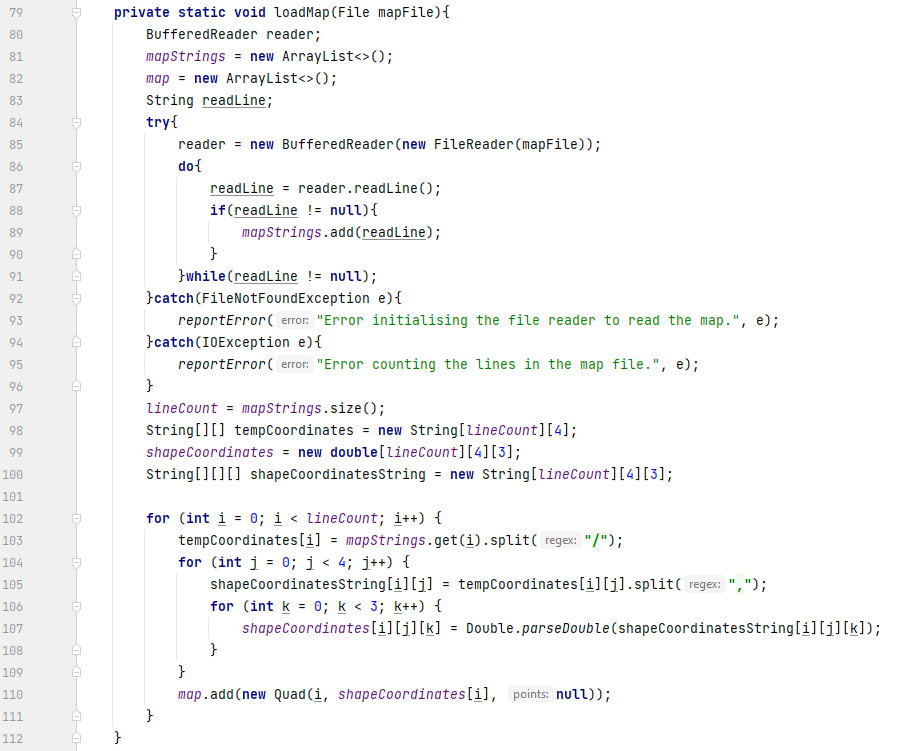
Graphical user interface, application

Description automatically generated with medium confidenceA solution is found after yet more searching: the method to clear the canvas was moved outside the loops since it was just erasing every line after it has been drawn. Secondly, the Task idea wasn’t the right choice, and a Timeline is much more appropriate (source: <https://stackoverflow.com/questions/36048234/javafx-canvas-stops-displaying-after-a-few-seconds-im-trying-million-ovals>). Since the timeline has a set delay, a target fps variable is created, and 1 is divided by it to calculate the delay needed for the target to be met. The same method for checking if the canvas updated as when it was first created is used. A variable ‘a’ (only for debugging purposes) is used and incremented every time the render method loops and is used to move the shape across the screen.

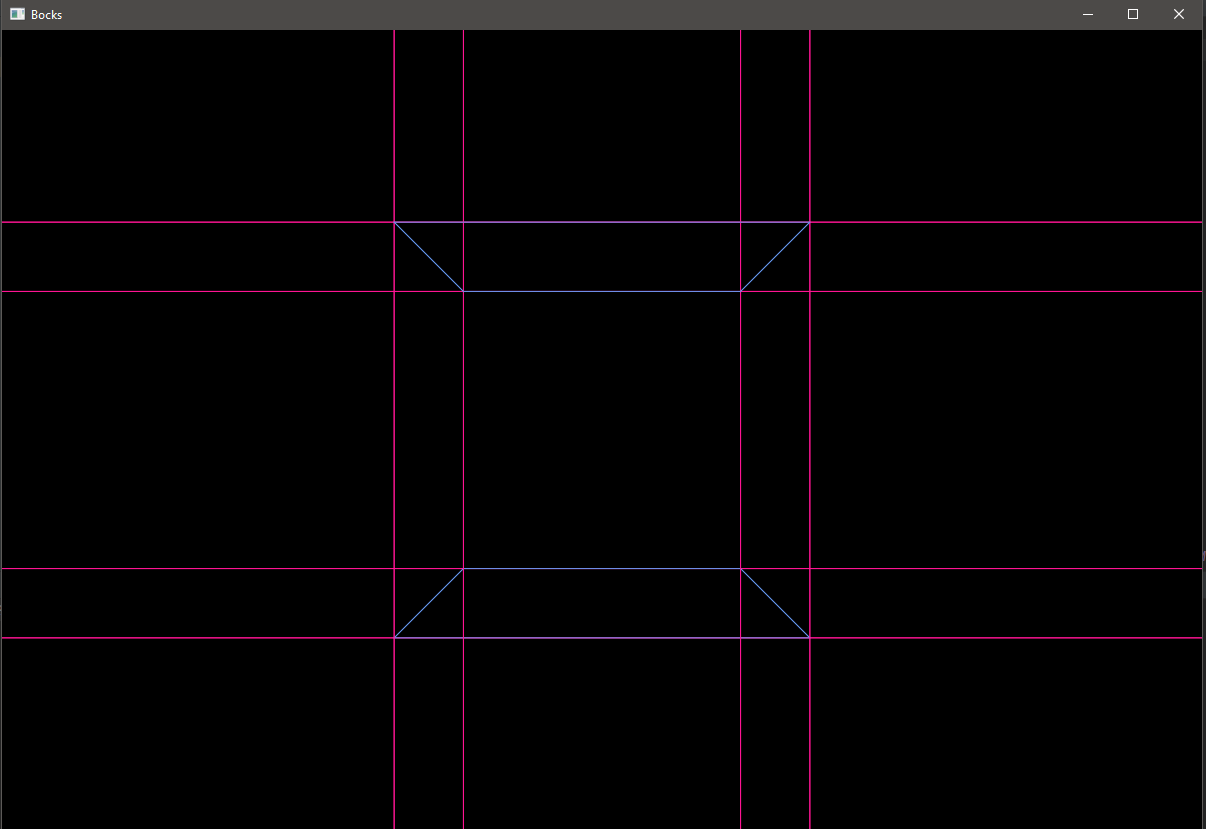
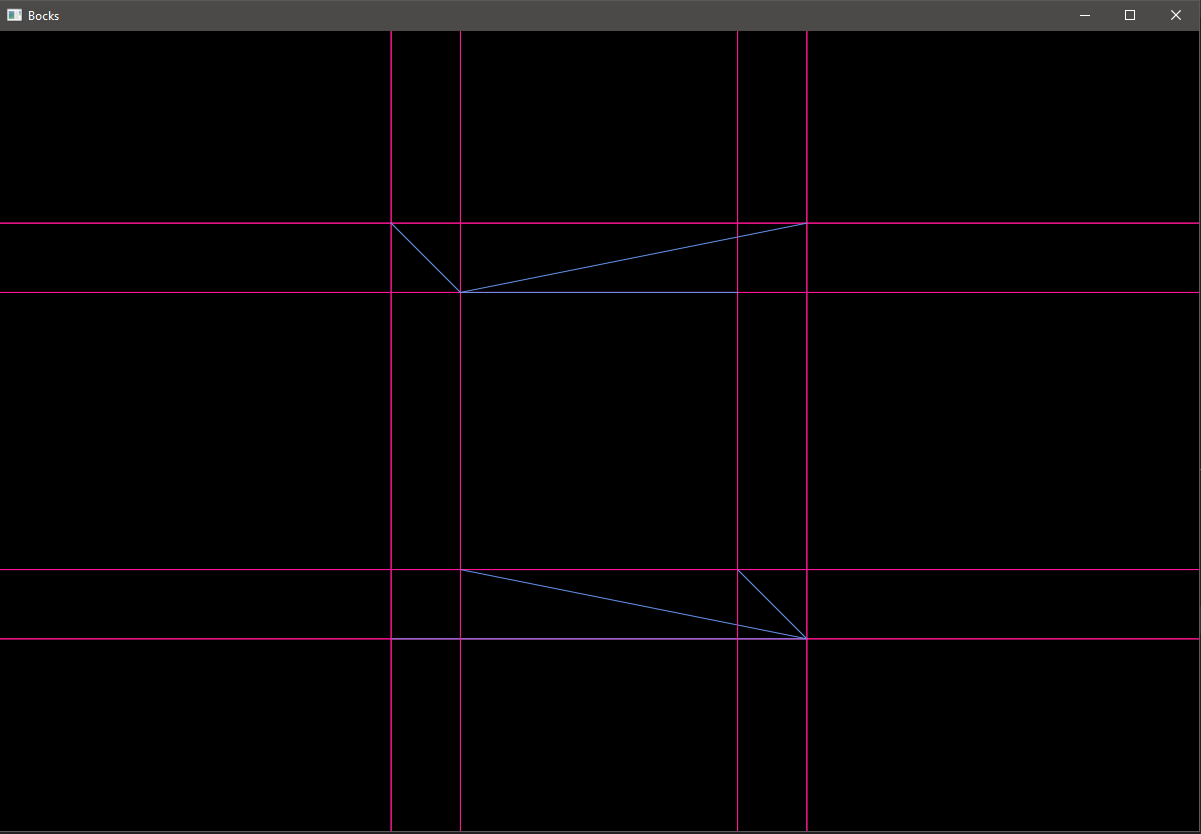
A picture containing chart

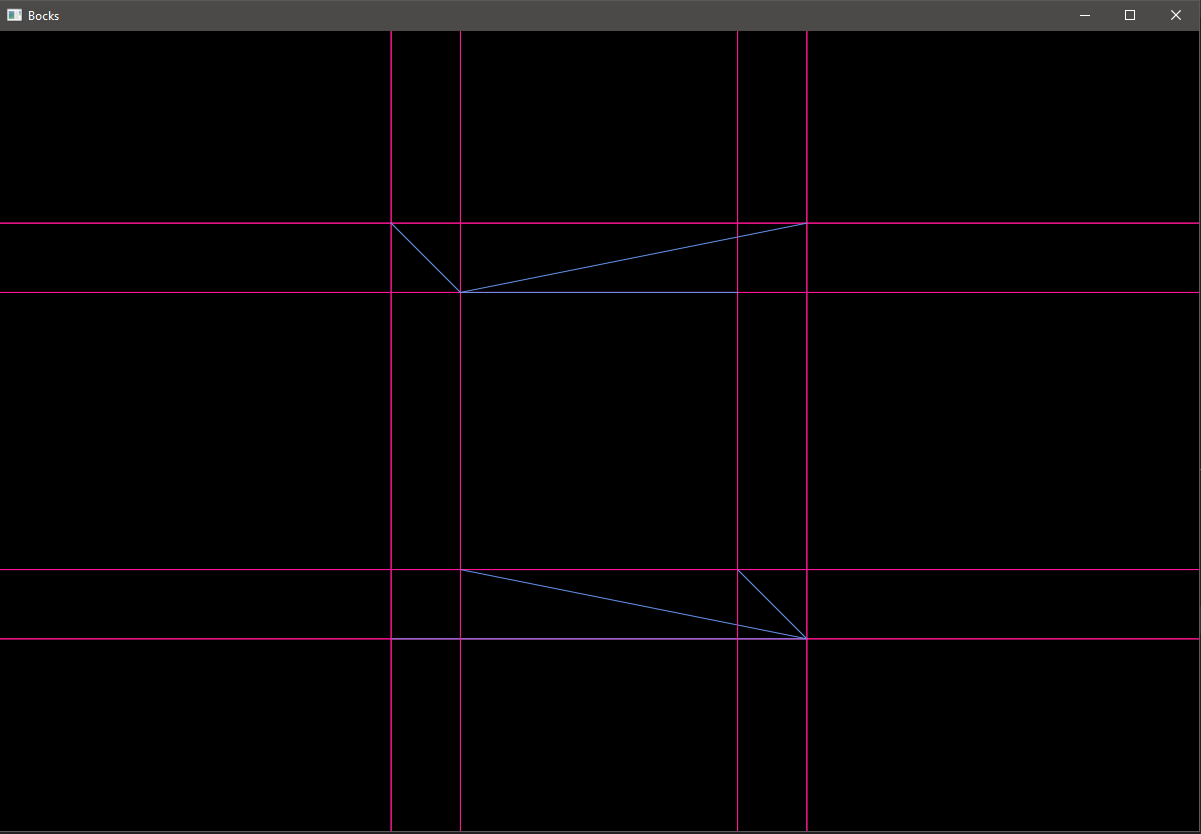
Description automatically generatedAfter some code cleanup, tweaks, and colour scheming, the program finally looks like it did in the JFrame.

Although this should have been done earlier, a new custom object is created within the Main class called ‘Quad’. It will hold the details of each individual square, hence ‘Quad’. The initial ArrayList holding the strings of the quads details is renamed to mapStrings, and a new ArrayList with the ‘map’ name is created, specialized to hold the new type of object. Subroutines are slightly modified so the new ArrayList of accounted for.

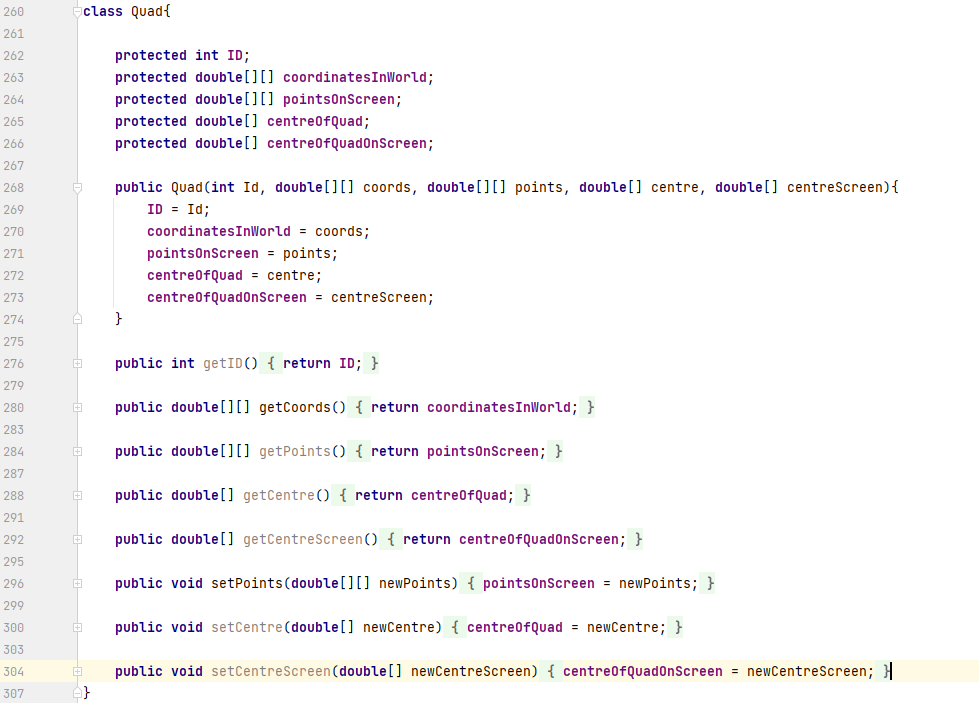


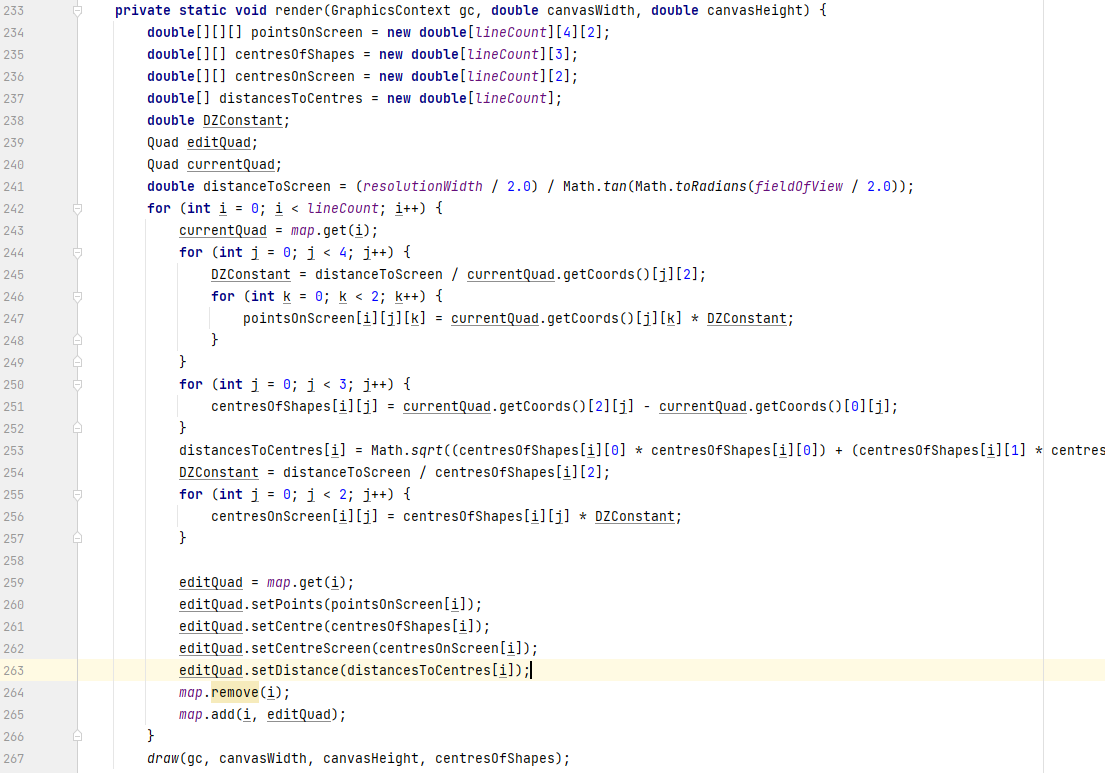
Checking the program, we can see the blue lines are now being drawn incorrectly, but as it turns out, it was just a simple typo.

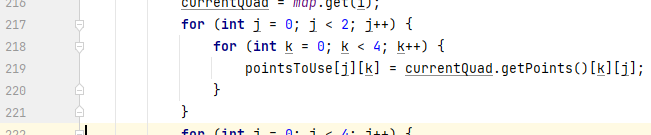


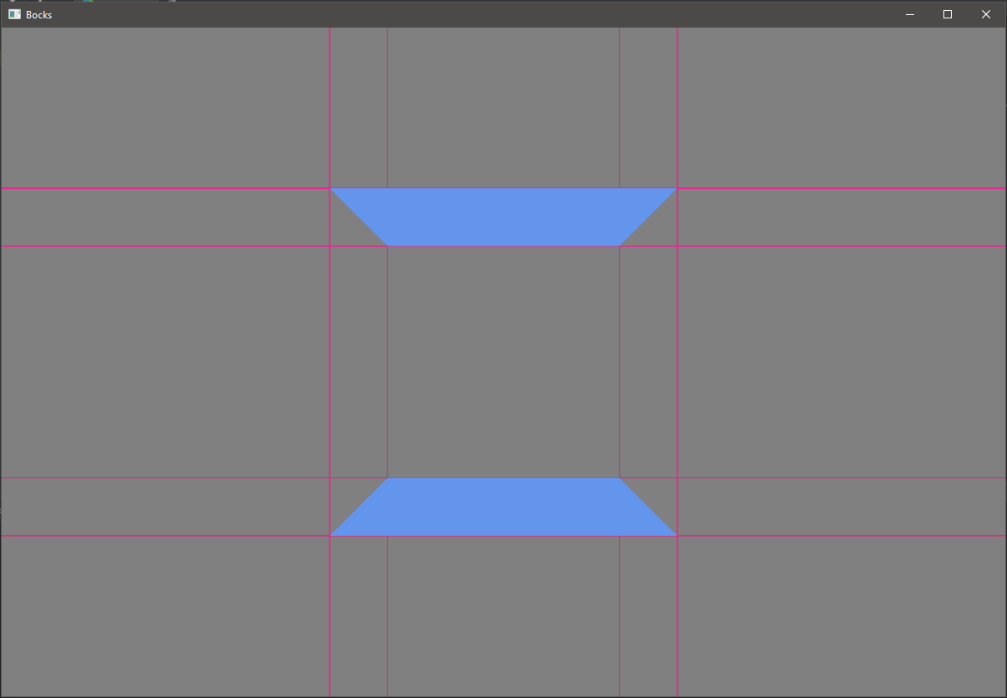


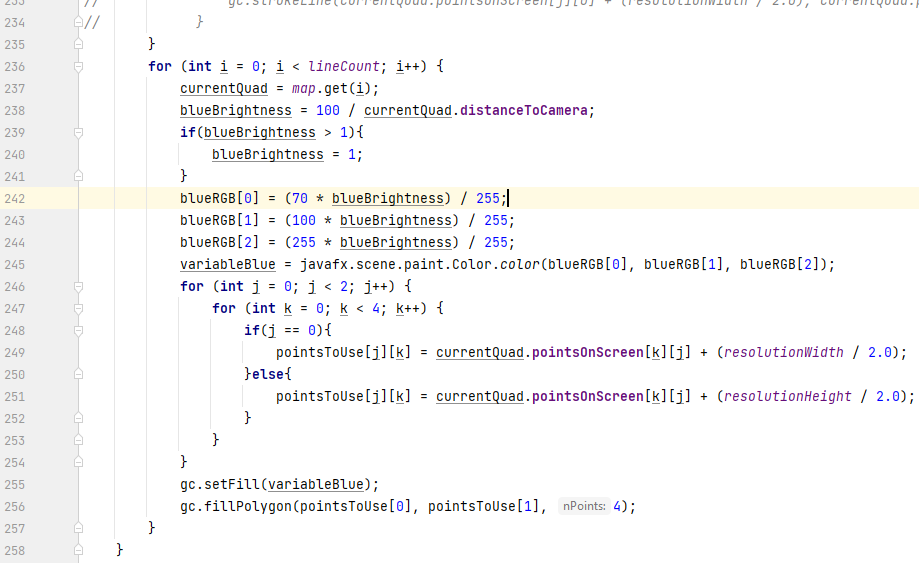
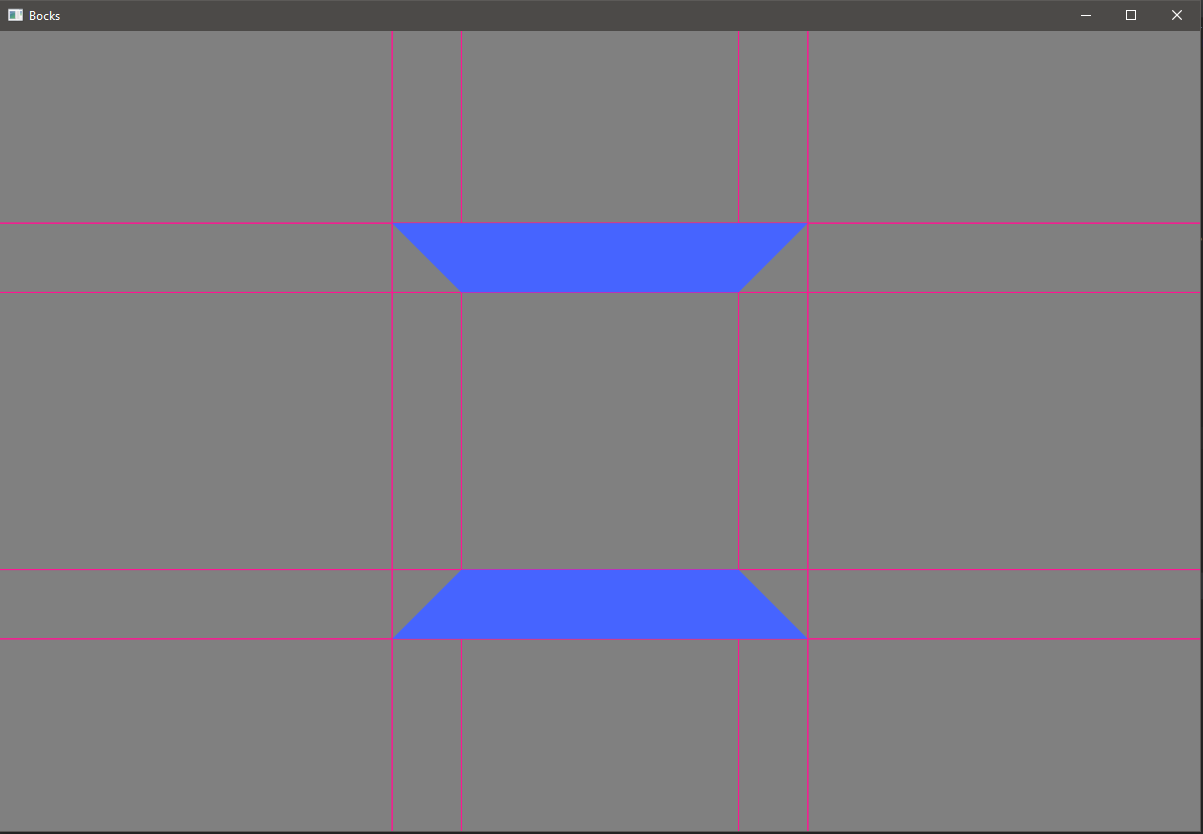
#### 2.1.2.6. Graphics improvement and lighting system prototype

A new goal is to now fill in the squares, and lighting could be introduced at the same time. To begin, two more arrays are added to the Quad object, as well as their respective get and set methods. Each Quad object will therefore hold their three-dimensional centres and where those centres are on the canvas just like with its’ points, possibly only for debugging purposes. The distance between each three-dimensional centre and the camera will be calculated and used as a brightness indicator for the fill colour of each square. Although primitive, any more complex would be a massive difference in difficulty to implement.

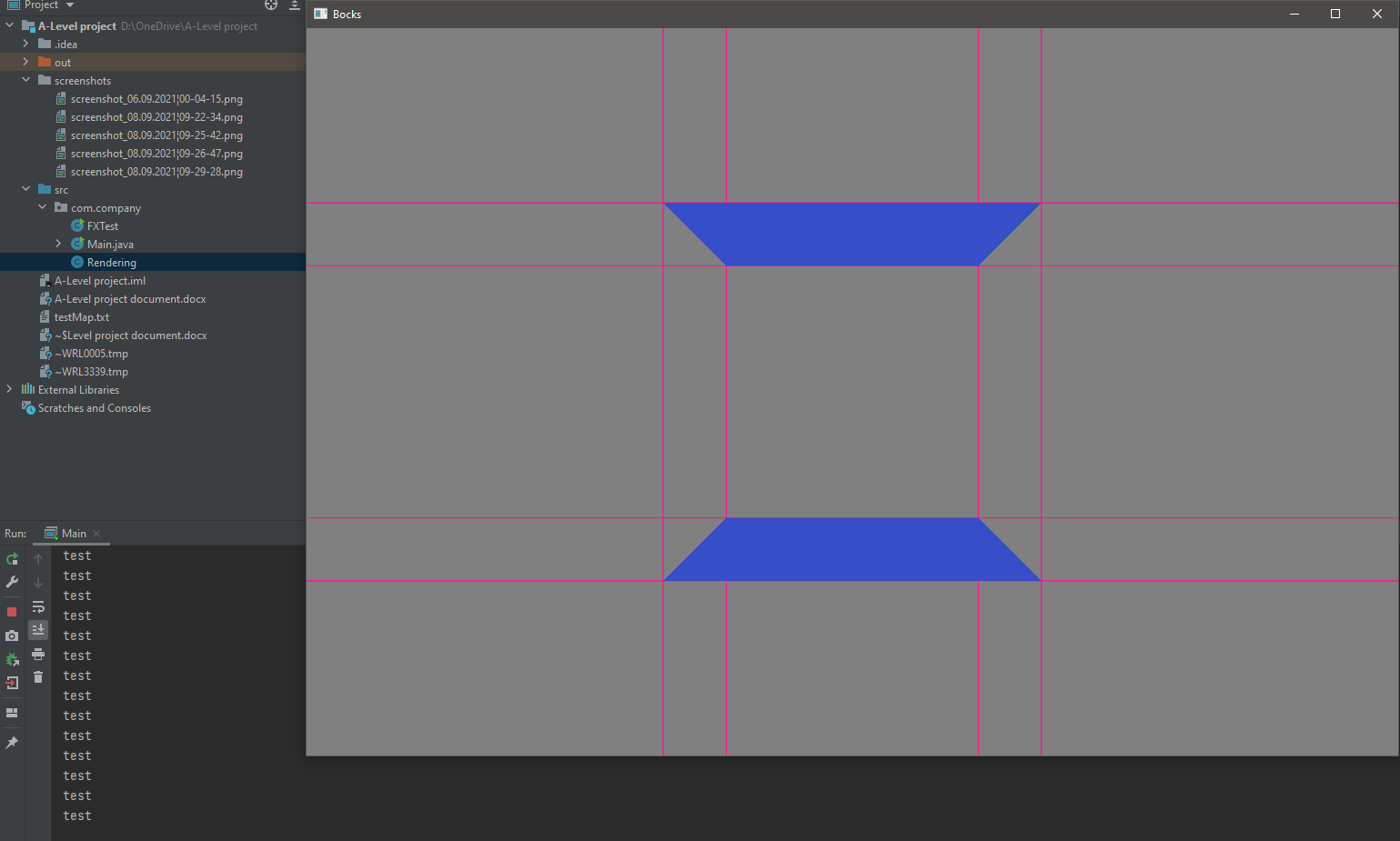
The render method is changed, and even more arrays are introduced. In the end, all Quad objects are also set their centres, centres as it appears on screen, and the distances from the camera to the centres. Unfortunately, this requires a LOT of dealing with arrays, and is hard to follow through.

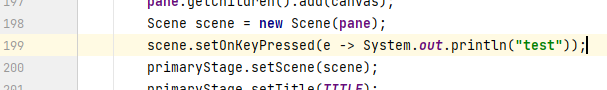
The second step is to fill the squares. Because the squares are made of lines, there is no way to fill them in. Fortunately, GraphicsContext has a method to draw a polygon using arrays of x and y coordinates, and this method also has a filled polygon alternative, which is perfect. The only problem now is that the point arrays held by the Quad objects are in the format [number][x, y, or z value]. However, this had an easy solution.

Quite a bit of testing and adjusting later, the whole idea works like a charm. A problem much earlier involves the background being black, not grey as specified, but what it should have set was the fill colour, not the stroke colour. Secondly, drawing the pink lines and blue squares were placed in separate ‘for’ loops, to ensure all blue squares are drawn on top of all pink lines. Thirdly, the loops arranging the arrays to be used by the fill polygon method now takes into consideration the problem from earlier when the colours were drawn in the top left instead of the centre. Finally the old code drawing the blue lines alone was commented out and most likely will be removed, although it’s good to make sure I really don’t need it anymore later on.

The fill colour’s brightness is finally sorted out, provided there were many errors relating to fiddly maths. The blue colour is now different because it is the old blue colour from when the JFrame was being used. To actually test this, a number treated as a sort of render distance was changed.

#### 2.1.2.7. Keylistener implementation

Now, to test if this entire JavaFX shebang was worth it, a simple print statement was set to run when a key is pressed in the scene, and that’s finally exactly what happens.

This being sorted, movement in the game should be a breeze in comparison.

Application

Description automatically generated with medium confidenceLinking the scene key pressed method with a custom method, the pressed key is dealt with appropriately. Testing it reveals that all elements previously implemented like the brightness feature work fine, but an issue arises, being that only one key can be held down at a time. (The cycleMovement method is called repeatedly in the Timeline just to multiply the camera’s movement distance by 0.9 repeatedly to gradually slow it down and the camera’s position is updated.)

Graphical user interface, text, application

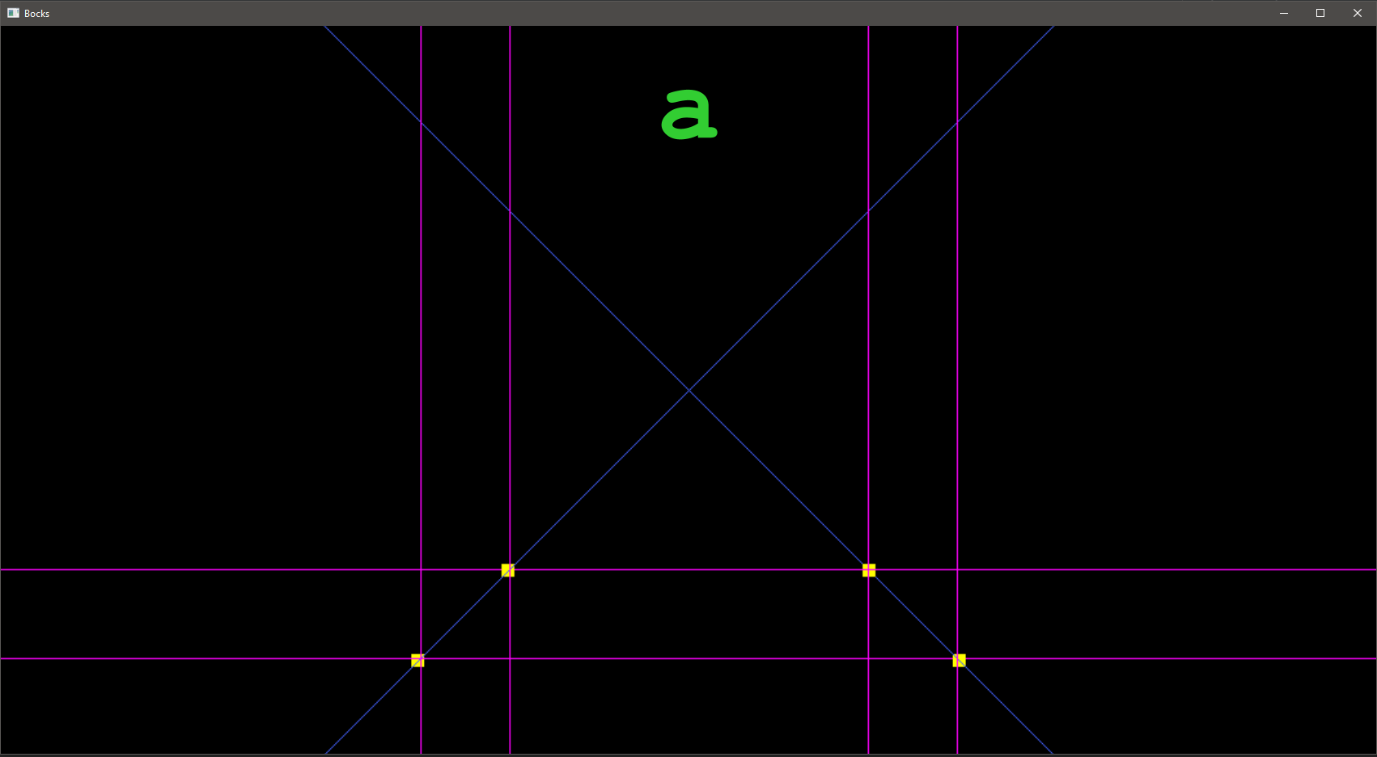
Description automatically generatedFollowing trial and error, a solution is somewhat swiftly found by assigning the ‘hash code’ of the key currently being pressed to a ‘BitSet’ with length 256, then checking whether the bit at the specified hash code position is 0 or 1. If 1, then move in the correct manner (source: <https://stackoverflow.com/questions/11851155/handle-multiple-key-presses-ignoring-repeated-key>). This seemingly works better than an array, for example, as a BitSet is much quicker to read and write and therefore bits in the BitSet are cleared properly. This also solves a lesser problem of a delay in continuous movement when a key is pressed down.

#### 2.1.2.8. Main menu development

A picture containing diagram

Description automatically generatedA picture containing timeline

Description automatically generatedAt this point, attention is directed to the absence of a menu, so a starting one was created. All elements now are temporary and may be changed later. Firstly, a title and background are created, the title centred because of a VBox, and the background created using default constants for all the objects needed in the process (the title is only a placeholder).

To see the window better, its dimensions were increased, but this created a white space around the background where the image was not large enough to fill the entire area. The solution (source: <https://stackoverflow.com/questions/54876509/how-to-fit-the-background-size-into-the-window-size-in-javafx>) was to simply use a defined BackgroundSize, as it contains booleans that could be set to true to fit the background image to the window. The math element involved is to make sure the image isn’t warped from difference in aspect ratio, by setting the image to go off screen on both axes.

Graphical user interface, text, application

Description automatically generated with medium confidenceA picture containing qr code

Description automatically generatedButtons are next to do, so I create an image to be used as a button in paint, and then add transparency using another program (the shadow in this image is just word formatting to show it’s transparent). For added detail, I want the title and buttons to move from off screen to on screen smoothly, and when a button is hovered over and then not, the button should lift and lower a little bit.

Another object in the Main class is created, called “CustomButton”. Each instance of this object will hold its Image, ImageView, and variables controlling the hover and startup animation, along with the methods that use said variables. One method is defined already; “hoverAnim” will be called using the ImageViews’ “setOnMouseEntered()” and “setOnMouseExited()” Icon, qr code

Description automatically generatedmethods. Testing the code, however shows that the methods only trigger when the mouse is touching a non-transparent part of the button, so the button is Text

Description automatically generatedgiven a semi-transparent background.

#### 2.1.2.9. Button animations

Once these methods are confirmed to work correctly, two methods are created to handle the events. One to raise the button and the other to lower it. A translation of the sine graph was used, as when translated correctly, a linear increase in the x value results in a gradual speed up, then slow down of the y value, which would do nicely.

Text

Description automatically generatedImplementing and debugging this ensures the math works, but the button’s position is only updated after the method is called.

To fix this, the while loops are changed to if statements, and the methods ‘onMouseEnter’ and vice versa are removed. In the timeline, if statements are added using a boolean whether or not the button is currently being hovered over, and said if statements calling the ‘hoverAnimUp’ and ‘hoverAnimDown’ methods.

A picture containing calendar

Description automatically generatedTimeline

Description automatically generatedGraphical user interface, text, application

Description automatically generatedTwo more buttons are created; an “Options” button and an “Exit” button, both behaving in the same way. As it is simple at this stage, the exit button is given functionality. An alert (source: <https://stackoverflow.com/questions/8309981/how-to-create-and-show-common-dialog-error-warning-confirmation-in-javafx-2>) pops up asking if the user really wants to quit, to avoid misclick of the exit button.

A picture containing timeline

Description automatically generatedGraphical user interface, text, application

Description automatically generatedA picture containing diagram

Description automatically generated

Graphical user interface, application

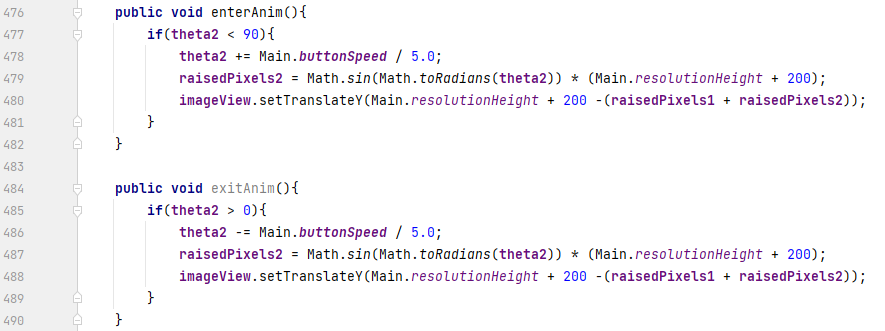
Description automatically generated

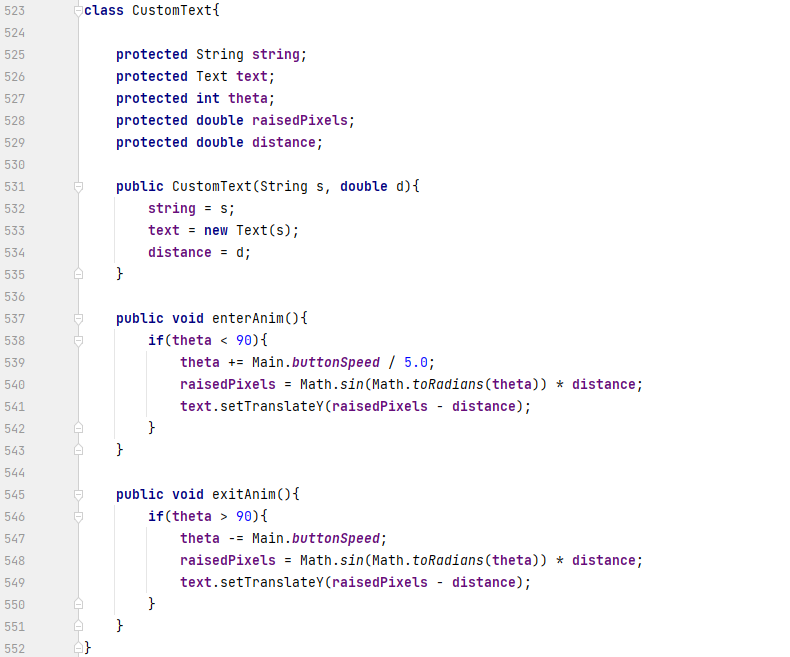
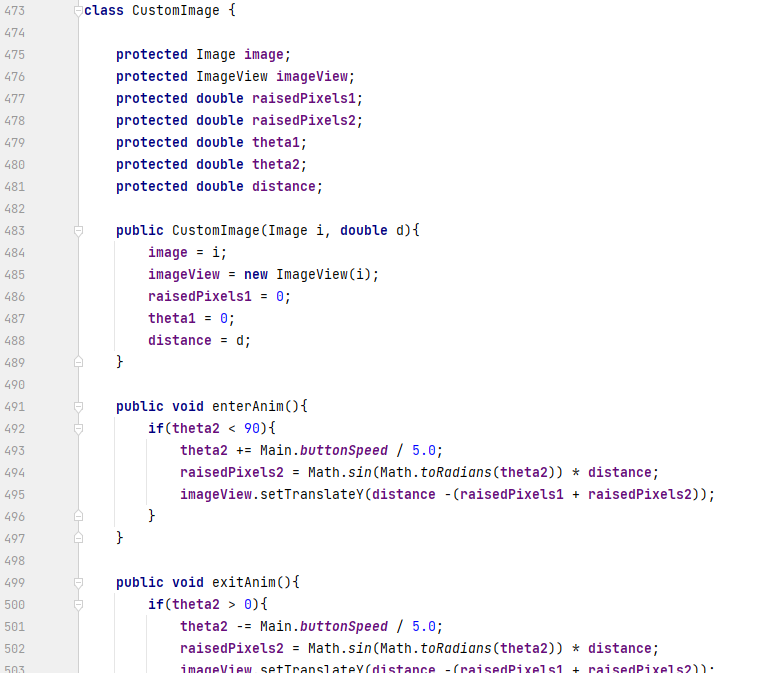
Icon

Description automatically generatedIcon

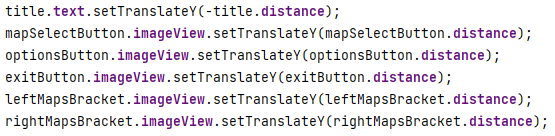
Description automatically generated

Following on from the button animations, they are also animated when entering or leaving the main menu using a similar code, and the variable theta is renamed to ‘theta1’.

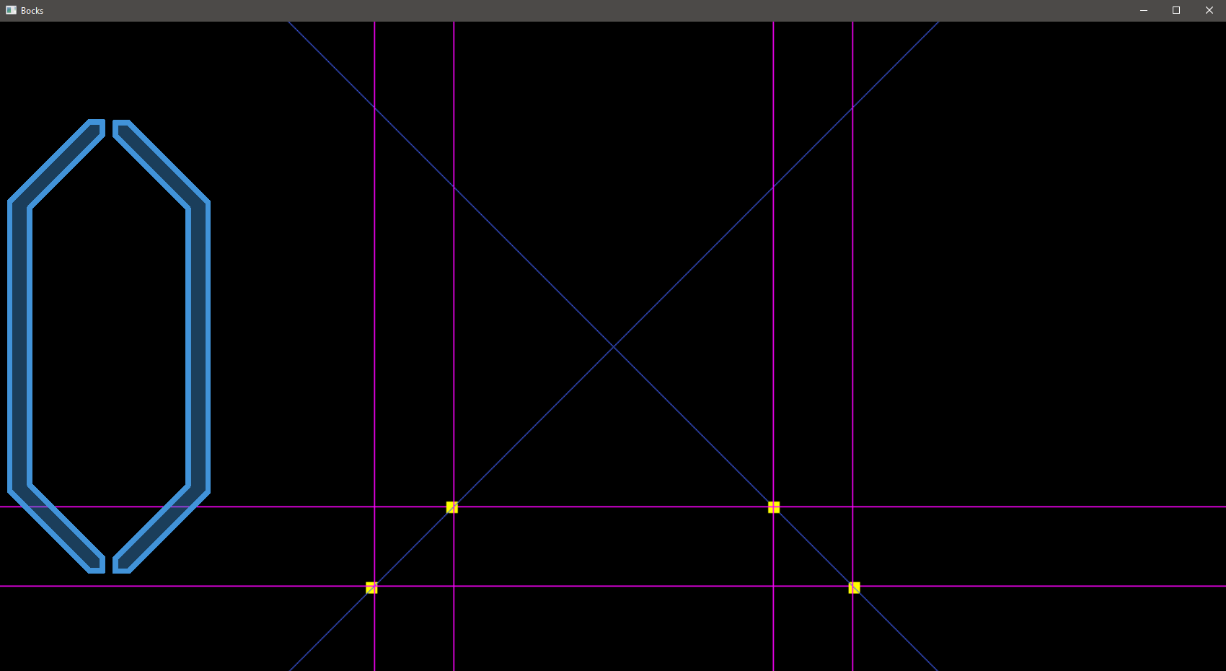
The title would need a similar animation, only coming from the top, so another object is created which is similar to the ‘CustomButton’ object, but only with text instead. The distance each element will travel is also given at initialisation to reduce hassle.



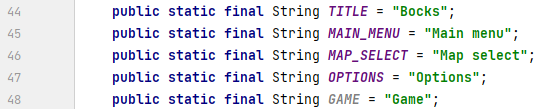


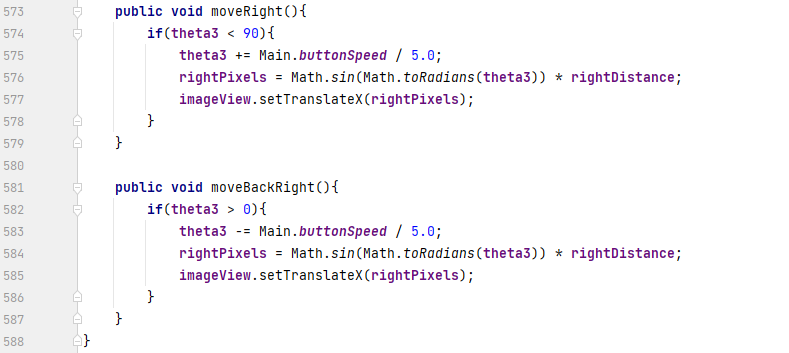


#### 2.1.2.10. Map select menu development

A new scene is needed for the map select menu, so such a scene is created. The hugely long background variable used by the menuBox is moved to a local variable so it can be used by all menus. Before the transitioning code is begun, the scene is tested to see if it works correctly, in which it does. The GUI will be explained shortly after more elements are added.

Transitions begin with two variables being declared to keep track of which menu is meant to be on screen, and whether or not each element of the menu is in the process of getting on the screen. Constant strings are declared to avoid having to type out the correct string many times. The transitioning variable is present in both ‘Custom’ objects.



Two large brackets are planned to move on screen, then the right bracket moves most of the window distance to create a space for the map selection menu. To make the right bracket move after it had fully moved on screen, another method is created in the ‘CustomImage’ (renamed from ‘CustomButton’) object and is called once the right bracket’s transitioning boolean is false. Both brackets are given the all-clear to move on screen when the last element of the previous menu (in this case, the map select button) moves fully off screen.

Background pattern

Description automatically generatedText

Description automatically generatedThe main menu button is supposed to go in the lower left of the map select menu, but HBox does not allow this, so it is swapped out for a GridPane. The brackets are rearranged to their initial positions after some formatting, and the main menu button is put into place.

Timeline

Description automatically generated with low confidenceThe main menu button is given basic functionality, but a problem arises with changing the scene too early, as when the main menu button is clicked the focusedWindow String changes to “Main menu”, therefore the scene is changed before the elements can move off screen. To fix this, the scene changing code is moved into separate ‘if’ statements after the first ‘if’ statements.

The animations now work as expected right up to the moving left animation of the right large bracket. After it does this, no other elements move, so in the ‘if’ statement a word is printed to check if this boolean works correctly.

A screenshot of a text message

Description automatically generated with low confidence

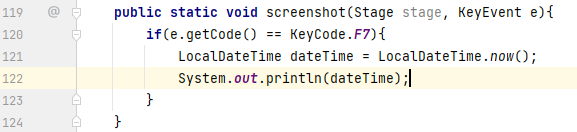
A little while later it was realised the boolean is misleading, as it only is true when the element is moving on and off screen, not moving at all, so a new boolean is the easiest fix to this. The animations now gladly all work as expected .

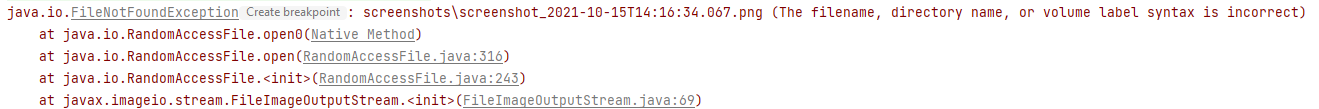
Text

Description automatically generatedAt this point, some code clean-up would be beneficial, so old methods such as the JFrame creation are removed. In the clean-up, ‘get’ statements in the custom ‘Quad’ object are removed as they can just be accessed by using ‘(quad name).(variable name)’. The IntelliJ warning count is reduced from 15 + 4 weak ones, to 9.

#### 2.1.2.11. Reintroducing screenshotting

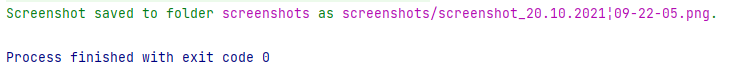
To introduce the screenshot functionality back, the date and time code is introduced back, but in an easier way, using the LocalDateTime.now method (source: <https://www.w3schools.com/java/java_date.asp>). The method is triggered when F7 is pressed.



Text

Description automatically generatedNow for the saving mechanic. Although complex, and could possibly somehow be optimised, this method would seem to work. The only problem is the LocalDateTime method giving colons in its returned string, which cannot be used in file names.

Diagram

Description automatically generated with low confidenceThe solution is a simple DateTimeFormatter to format the date in an acceptable way, and with that, the method works correctly.

#### 2.1.2.12. Creation of map list

Table

Description automatically generated with medium confidenceMoving back to the map select screen, maps need to appear in the brackets to be selected to play. Another custom object (ListMap) is created, and holds more variables than the other custom objects, because it is essentially a group of objects that will fade in and out.

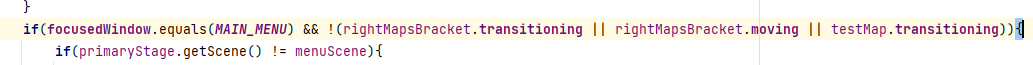
A picture containing background pattern

Description automatically generatedText

Description automatically generatedAfter a few changes to the initialisation of the ListMap object (setting the initial opacity to 0 and initialising the Group object after a few NullPointerExceptions) and adding the screenshot function to the map select scene, the entry animation works as intended, but on going back to the main menu, the menu immediately appears and its elements exit the screen then enter back on the screen. The ‘movingLeft’ boolean was repurposed to be true when the right bracket is moving at all to avoid having to use another boolean.

Text

Description automatically generated

The solution is to simply involve the transitioning boolean of the test map in the code that changes the scene to the main menu.

During adding a hover animation to the test map, the ‘if’ statements seemed to be building up quick, so an array of CustomImages was created and holds each button object that needs a hover animation. This array is then used in a ‘for’ loop applying the same logic to every element in the array. This proved to work as intended; there is no difference in animation.

Graphical user interface, text, application

Description automatically generated

The same animation is applied to the test map object, only outside the ‘for’ loop as it is not a CustomImage. The text in the test map is positioned accordingly.

Graphical user interface

Description automatically generated with medium confidenceGraphical user interface, text

Description automatically generated

A background between the large brackets is missing, however, so a new image is simply applied to the scene (before any other element so it is behind all elements and therefore does not conflict with actions), positioned accordingly, and given the same fade in and fade out animation as the ListMap object in the CustomImage object.

A picture containing background pattern

Description automatically generatedA green coloured version of the background of the map object is made and is used as the background when a map is selected. When a map object is clicked, it now has a ‘selected’ boolean that will be set to true, and since all map objects are now in an ArrayList, this ArrayList can be used to set the selected boolean of all other map objects to false. This is to make sure only one map is selected at a time. To the right under the main menu button, map options will be added later.

Icon, qr code

Description automatically generatedThe last step to being ready to link the game and menu is creating a start button to the lower right, below where the options will be. My first idea was to use green again, but a friend suggested I use purple for colour variation, which looks better.

Timeline

Description automatically generatedThe button is given the default hover-over animation, move on/off screen animation, and design.

#### 2.1.2.13. Switching from text files to binary files

Text

Description automatically generatedRealising the program would have to read every text file in the project folder when searching for maps, I changed the file extension to ‘.mfb’ (Map File Bocks, where Bocks is the game name) while also formatting its contents better.

I thought a new type of file reader would be needed because of the switch from a text file to a binary file, but as it turns out, the current map reader still works. Only the code organising the data is changed.

#### 2.1.2.14. ‘Object’ object and light object

This is a good opportunity to differentiate world shapes from objects, and world shapes and objects from lights. Since these are currently the three possible map elements, the ‘map’ ArrayList holding all strings in the map file is renamed to ‘world’, and ‘mapStrings’ to ‘worldStrings’. Two other ArrayLists are made for the strings holding the objects and lights separately, named ‘objectStrings’ and ‘lightStrings’ respectively. The variable ‘lineCount’ is too split into three: ‘quadCount’, ‘objectCount’, and ‘lightCount’.

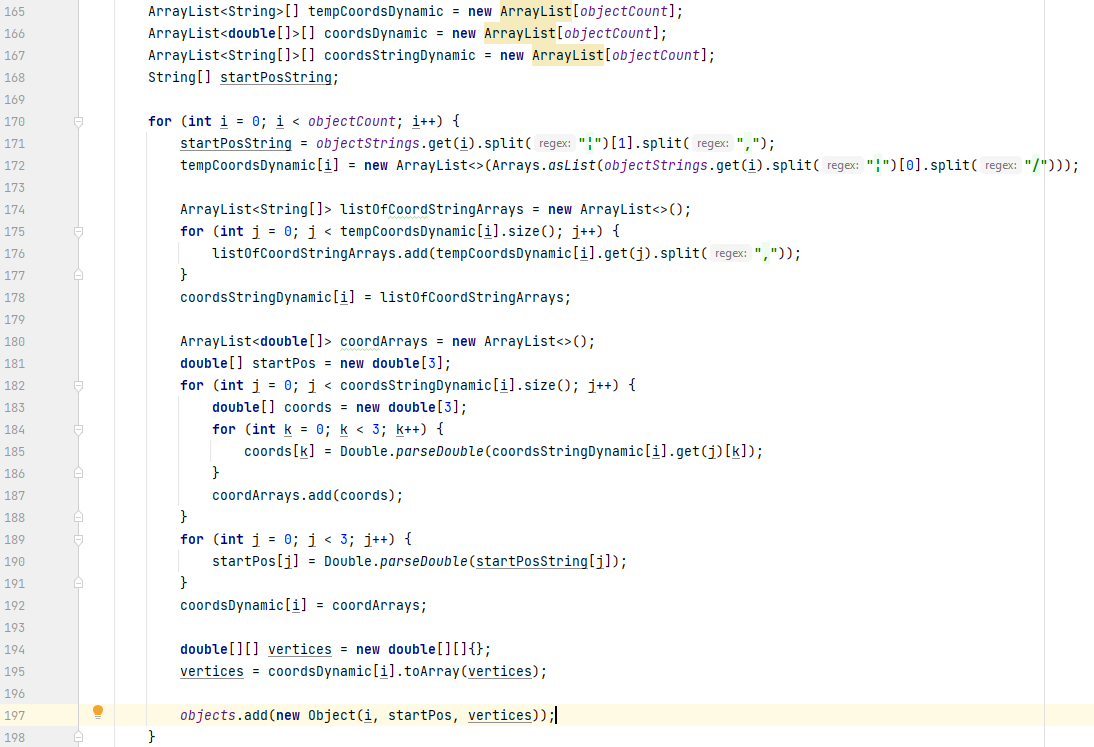
Text

Description automatically generatedText

Description automatically generatedWhen it comes to the local variables of organizing the data, they will be kept the same and reused for world geometry, objects, and lights. However, objects and lights need their own ArrayLists to be stored in, which means creating new objects (objects in java).

These two objects will do for now and will be added to and explained later on. Basically the variables are only based on future anticipation, and those who are not set when the instance is initialised are intended to be set in the rendering method via the methods given.

Now for adding the ‘Object’s to the objects ArrayList. At first, I thought it would be as easy as the same method for adding the Quads to the world ArrayList, but a problem arises, that is the number of points an ‘Object’ can have is not fixed. This turned out to be an incredibly fiddly problem, but the solution was found eventually, and will be a pain to explain, so when talking about different arrays, text and the image provided will be colour coded to avoid confusion.



Text

Description automatically generatedStarting off, a static array of ArrayLists is made to hold the data in the form [“x1,y1,z1”,”x2,y2,z2”…], [“x1,y1,z1”,”x2,y2,z2”…]… where the quotation marks hold Strings in the ArrayLists, and the square brackets hold the individual ArrayLists. There is an ArrayList for each ‘Object’. Next, there is a static array of ArrayLists, each holding static String arrays. This is to simply split the Strings in the previous array, e.g. “x1,y1,z1”, into x1, y1, and z1. Finally, the last array is the same as the previous array, but for doubles, not Strings.

The starting position of the ‘Object’ is first extracted by splitting between the symbol ‘¦’, and picking out the second variable from that split. To get the first array (form: [“x1,y1,z1”,”x2,y2,z2”…], …), the String x1,y1,z1/x2,y2,z2/…¦Xstart,Ystart,Zstart is split again between the symbol ‘¦’, and this time the first element containing the data is used. This element is then further split between the symbol ‘/’, and all elements from this split is inserted into the first ArrayList of this array (first because let’s say the first object in the map is being looked at).

Then an ArrayList of static String arrays is temporarily made, the elements looked at previously are split even further between the symbol ‘,’, and *these* elements inserted into the temporary ArrayList sequentially via a ‘for’ loop. This ArrayList is then stored in the first element of the ‘coordsStringDynamic’ array (form: [“x1”,”y1”,”z1”,”x2”,”y2”,”z2”…],) (first again because we are looking at the first object in the map).

Next, using three nested ‘for’ loops and more temporary arrays, every single String in the previous Array is converted into doubles, and all these doubles are placed back into the static array of ArrayLists of static double arrays.

Finally, the last array just mentioned is converted into a two-dimensional static double array and passed onto the statement to add a new ‘Object’ to the list, along with the variable ‘i’ for the ID and the start position created earlier.

This certainly may be overcomplicated, but at this point it works very well and could be changed in the future.

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generatedFinally, the ‘Light’ object was way easier in comparison to the previous ordeal, using similar principles but nothing to do with ArrayLists anymore thankfully. The String in the file is in the form [x,y,z/brightness/red,green,blue] (without square brackets), so it is loaded into a String array after being split between the symbol ‘/’. The data is then retrieved and converted from there into their corresponding arrays again using the same method from before and given to the respective new ‘Light’ object when added to the list.

#### 2.1.2.15. Avoiding loading a null map

In the map select menu, if the user does not choose a map but clicks the start button anyway, the scene turns to grey and nothing happens, assuming this is a map with nothing in it. To avoid this, a version of the start button only showing the purple brackets is made and added to the code. Now this image is used by the start button when the variable ‘mapToLoad’ (which holds the selected map file) is null, otherwise it uses the original image. The hover-over animation is disabled when the selected map file is null, too.

Another functionality is given: when the selected map is clicked again, it is unselected. Now the method called by the ‘setOnMouseClicked’ procedure looks like so.

Timeline

Description automatically generated with medium confidence

This method first locates the map in the map list, then checks if it is already selected or not. If so, it is unselected and the mapToLoad variable is set to null. If not, then of course it is set to selected and the mapToLoad variable now contains the map file associated with the selected map. All other maps in the list are set to unselected.

#### 2.1.2.16. Rendering issue solutions

Chart, funnel chart

Description automatically generatedChart

Description automatically generatedWhen overviewing the code one final time for bugs, one emerges in the form of only one quad’s brightness value being used by all quads.

This was only a mathematical error though and was quickly resolved.

Chart, funnel chart

Description automatically generatedDiagram

Description automatically generated with low confidenceAnother possible harder to resolve problem also appears though.

Graphical user interface, application, Teams

Description automatically generatedThe pink lines can now be disabled with the F3 key, and to look at this problem, some text was added to the game scene, containing the coordinates that the GraphicsContext object is using to draw the shapes.

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated with medium confidenceSeeing the coordinates are fine when this glitch occurs, that meant the problem had to do with drawing the shape. After extensive debugging (mostly thinking) I realised the shape is drawn mirrored on both axes if the shape is behind the player. This glitch was a product of two points being above the player and two being below. I wrote down the positive and negative signs of all the points and the variable responsible for calculating them and realised this constant must always be negative.

A picture containing text

Description automatically generatedThis small bit of code makes half the difference, placed just after when this constant is calculated. Now when moving past the square, it is no longer mirrored, but still appears when it isn’t meant to.

#### 2.1.2.17. Rotation mechanic

I decided to create a more versatile way of preventing these visual glitches, which meant implementing the camera rotation mechanic. Although it seemed easy, it took several methods to find one that was suitable.

Firstly, I found equations for the two-dimensional rotation of a point but found no way to properly implement it in three dimensions (source: <https://academo.org/demos/rotation-about-point/>).

Next I saw a supposed method for three-dimensional rotation and attempted to implement it, but the visuals would warp and distort in strange ways rather than rotating (source: <https://stackoverflow.com/questions/45357715/how-to-rotate-point-around-another-one>).

Further finding maths equations written in hard-to-understand ways did not help the case either (sources: <https://math.stackexchange.com/questions/2628960/rotate-around-a-specific-point-instead-of-0-0-0>, <https://community.khronos.org/t/rotation-around-a-point-in-3d/21932>).

Graphical user interface, website

Description automatically generatedThen I found an algorithm online which seemed to do the trick (source: <https://stackoverflow.com/questions/34050929/3d-point-rotation-algorithm/34060479>), however further testing it revealed the up and down rotation was fixed on an axis, not on the camera. Looking into how this algorithm was created shows that the problematic part of it was matrix multiplication, and recalling from a previously looked at method that the order of multiplication is important meant that this is the problem.

A link is given in the post of the algorithm to the (german) Wikipedia page of Euler angles (pitch, roll, and yaw).

From this point I decided to create my own algorithm for this matrix multiplication, although using arrays to make it easier to understand. The matrices were fetched from said wiki page (<https://de.wikipedia.org/wiki/Eulersche_Winkel>, I could have used the English page but chrome conveniently prompted me to translate the page to english) and the algorithm from a new webpage about matrix multiplication (source: <https://www.baeldung.com/java-matrix-multiplication>).

Graphical user interface, text, application

Description automatically generated

A picture containing text, light

Description automatically generated

Text

Description automatically generated

Even this encountered a problem however; since my interpretation of which axes are which proved to be wrong, there was no way of telling which is which. Eventually though, through a lot of trial and error of experimenting with the order of matrix multiplication and angle changing, it was working correctly even if the world shapes were at a 90-degree angle.

A picture containing transport, aircraft, airplane

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidenceHaving got this close, I decided to just swap the end combination of the matrix values and point coordinates. In the snapshot, note how the value ‘pz’ is multiplying the y values in the matrix and vice versa.

In the end, this was the method that worked.

Text

Description automatically generated with medium confidenceTable

Description automatically generated with medium confidenceFinally, back to the issue that brought this about, I figured all it needed was if the centre of a shape was behind the player, then that shape would not be drawn. Utilising more three-dimensional mathematics, I was able to create code that created a vector, then the plane perpendicular to that vector, and a boolean checking if the centre of the shape is on the right side of that plane. After perfecting this, I noticed the ‘y’ angle could be hard coded and its trig function is therefore redundant, and the whole algorithm cold be simplified to a simple inequality. The ‘inFrontOfCamera’ boolean resets to false for each shape rendered.

Graphical user interface, text, application

Description automatically generatedText

Description automatically generatedNow the rotation mechanic is mastered, the movement should be made to no longer be fixed to the axes. Simply using trigonometry (again) and two separate arrays for the forwards-backwards motion and the left-right motion to avoid the player only being able to move in one direction.

A picture containing shape

Description automatically generatedAs a little side task, I noticed my project doesn’t have a window icon, so I created a box in game, screenshotted it, removed its background while cropping it, and use that as the icon (code for setting the icon source: <https://stackoverflow.com/questions/10121991/javafx-application-icon>)



#### Icon Description automatically generated2.1.2.18 Mouse controls for camera rotation

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generatedThe final aspect of this three-dimensional business is using mouse input to rotate the camera (currently using WASD). This begins with keeping the mouse in the centre of the screen with it being transparent too, then any movement will be recorded and used, then negated. A ‘Robot’ handles the automatic mouse movement (source: <https://stackoverflow.com/questions/4231458/moving-the-cursor-in-java>), and earlier while creating the main menu, I noticed a ‘setMouseTransparent’ method in the IntelliJ drop-down.

Combining these two, the mouse stays in the top left of my monitor, but only because the values used by the Robot are both 0, meaning the Robot uses the monitor x and y instead of the stage’s.

Passing the stage to the ‘gameMouse’ method and using the stage’s position and size to move the mouse to fixes this problem.

Text

Description automatically generated

Now the mouse stays in the centre of the stage but is not transparent (no screenshots because it’s difficult to do so with an immovable mouse). Reading up on a similar issue online (source: <https://stackoverflow.com/questions/29043081/javafx-mouse-transparent-is-not-working>), the method used simply makes buttons and such ignore the cursor instead of actually making it visibly disappear. The solution (source: <https://stackoverflow.com/questions/27714288/javafx-8-how-to-make-mouse-cursor-invisible>) thankfully is just as seemingly simple, and in this case, it works.

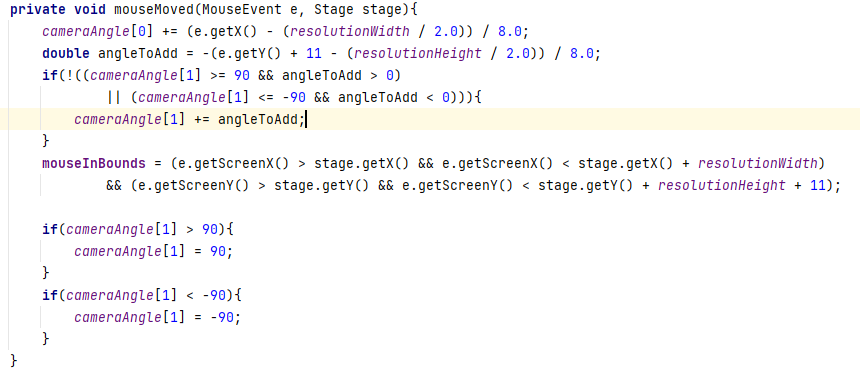
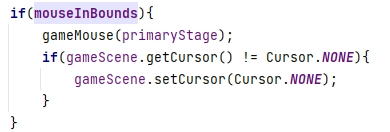
Text

Description automatically generated

Graphical user interface, text, application

Description automatically generatedRecording and using mouse movement is simple; linking this method to the ‘onMouseMoved’ method of the game scene and some small trail and error, this functionality now works as intended (11 is added to the y-angle to account for the window’s bar at the top; the stage’s getY method used by the Robot includes this bar while the game scene’s getY method does not).

Additionally, three-dimensional games almost always have some sort of restriction to the vertical camera movement, stopping when looking straight up or down, so I made a semi-simple if-statement to create this.

The ‘mouseInBounds’ boolean is used by the method moving the mouse to decide whether to do so or not; if the user’s mouse is not in the window, they probably wouldn’t want the cursor to disappear anyway.

### 2.1.3. Object diagrams and class definitions

### 2.1.4. Prototyping

### 2.1.5. Design data dictionary and data structures

### 2.1.6. File organisation

### 2.1.7. E-R model and diagrams

### 2.1.8. Human-Computer-interface design

# References

<https://stackoverflow.com/questions/5762491/how-to-print-color-in-console-using-system-out-println> - ANSI Colour codes to colour the output in the console.

<https://docs.oracle.com/javase/tutorial/uiswing/components/border.html> - Compound border code surrounding JPanels.

<https://stackoverflow.com/questions/12674064/how-to-save-a-bufferedimage-as-a-file> - Image saving mechanic used in the screenshot function.

<https://docs.oracle.com/javase/tutorial/uiswing/layout/box.html> - BoxLayout used in the settings JPanel to align the JButtons along the y-axis.

<https://stackoverflow.com/questions/1965347/how-do-i-initialize-a-graphics-object-in-java> - Introduction to creating the Graphics object.

<https://docs.oracle.com/javafx/2/canvas/jfxpub-canvas.htm> - Canvas and GraphicsContext basic initialisation.

<https://stackoverflow.com/questions/36048234/javafx-canvas-stops-displaying-after-a-few-seconds-im-trying-million-ovals> - Timeline to fix many problems introduced by Tasks.

<https://stackoverflow.com/questions/11851155/handle-multiple-key-presses-ignoring-repeated-key> - BitSet concept to process multiple key presses at once.

<https://stackoverflow.com/questions/54876509/how-to-fit-the-background-size-into-the-window-size-in-javafx> - Background image resizing to fit the window.

<https://stackoverflow.com/questions/8309981/how-to-create-and-show-common-dialog-error-warning-confirmation-in-javafx-2> - Exit prompt when exiting the program.

<https://www.w3schools.com/java/java_date.asp> - Date and time method for the screenshot functionality.

<https://academo.org/demos/rotation-about-point/> - First three-dimensional rotation method.

<https://stackoverflow.com/questions/45357715/how-to-rotate-point-around-another-one> – Second three-dimensional rotation method.

<https://math.stackexchange.com/questions/2628960/rotate-around-a-specific-point-instead-of-0-0-0> – Third three-dimensional rotation method.

<https://community.khronos.org/t/rotation-around-a-point-in-3d/21932-> Fourth three-dimensional rotation method.

<https://stackoverflow.com/questions/34050929/3d-point-rotation-algorithm/34060479> – Partially working rotation method.

<https://de.wikipedia.org/wiki/Eulersche_Winkel> – Euler angle matrices.

<https://www.baeldung.com/java-matrix-multiplication> – Matrix multiplication algorithm.

<https://stackoverflow.com/questions/10121991/javafx-application-icon> – Stage icon setting.

<https://stackoverflow.com/questions/4231458/moving-the-cursor-in-java> – Robot to keep the cursor centred.

<https://stackoverflow.com/questions/29043081/javafx-mouse-transparent-is-not-working> – Explanation of mouse transparency for a node.

<https://stackoverflow.com/questions/27714288/javafx-8-how-to-make-mouse-cursor-invisible> - Mouse invisibility.