**I**

**Introduction**

The purpose of the project is to research and develop Algorithms and machine learning of various types accompanying progression of the knowledge base to further research and technical reports. With aim of contribution to a larger library of technical reports, the exploration of new efficient methods based on data structures and use cases should also be compared.

The potential to explore other areas programmatically, is also encompassed within the projects wider scope as time goes on. Other areas such as security, safe research and potentially even variants of other specialist areas would be branches to welcome and umbrella within the projects scope and drive specialist areas, light for example.

The initial beginnings of the project are to gain a best working practice and so trial and error is inevitable. However, not unavoidable. In example, the use of virtual machines without virtual network interfaces would be an ideal way of sandboxing for a lot of the exploratory work and is one way to mitigate irrational behaviour from getting to the infrastructure. Though is it?

Given we are using Java, which is a higher level abstraction of an embedded language, we are relinquishing control of embedding language and the code is more readable, we can see easily, what data types and where while the lower level which Java is built from is handling processing and not forcing embedding. There are no forceful modifications to the standard operation of the hardware components during testing and proof of concept stages.

As such, to get the project started, an idea, seemingly simple on the interface, yet complicated in the background with a lot of planning and thought required. Once the first iteration is completed, further review of the projects general direction and scope can occur.

**Aims**

* It has to be unique to this project
* It has to be beautiful in appearance and in code format
* It has to be planned out first
* It has to be researched
* It has to use algorithms to generate an interface and to work uniquely
* I have to be proud of the work afterwards

**Objectives**

* It has to be a solution to other apps
* It has to work fully planned out and has to be interactive
* It has to get people thinking about what they are doing
* It has to be safe and not illegal
* It must have potential and an alternative use case

**Interface**

To start the app is opened from its icon and the end user is presented with a multilingual selection scrolling drop down. Possibly even a secondary drop down selector to choose automatic or interactive. Potentially even a tertiary drop down selector for a word count selection to limit the count according to screen space 1-5 words in example.

Once the drop down has been selected, the next screen phases in and it has a very light background with a dark line horizontally across the centre of the screen.

Dots appear randomly spread out across the horizontal line and the keyboard is presented to the end user to encourage the end user to type in a word or a small set of words.

As the user types, the dots along the line begin to fray out from the line all interconnected as a mesh network and the letters begin to appear where some of the dots were on the horizontal line across the screen.

Once the end user finishes typing and or presses enter, the first shuffle occurs. After the first shuffle, the end user can either move the letters around or after a small delay of 5 seconds, the letters reshuffle as the algorithm finds another logical dictionary based use of all the letters or finds another word from the letters typed. Whilst the shuffle occurs the mesh stays connected and arrives back to the single line, expanding for each shuffle randomly moving and contracting back to the line.

The app continues to run until all possibilities are exhausted or until the end user closes the app. The app can be tilted from portrait to landscape and back again and as the shuffle occurs, the colours of the line and letters change in random pastel hues or stronger hues. In addition, the letters go back to the line across the screen to present the new word(s) match.

As each match is found, the remaining letter(s) should stay external to the line as a mesh and or fade to permit the mesh to fully go to the line again and then reappear as the mesh expands again for the next shuffle.

**Interface Functionality**

Looking at the work from Irregulars [Benedis-Grab, 2020] and the combined solutions provided from StackOverflow and Google [2022], it would be possible to get the screen sizes and animate the polygonal ‘network mesh’ and even make that interactive from the users perspective. Furthermore, coding the linear work [CodeJava, Lewis Loftus] and looking into the solutions provided for changing the hues on StackOverflow [2020] it would be possible to combine those object orientated solutions with other Algorithms [Sedgewick and Wayne, 2011] to form an aesthetically pleasing visual display on the front-end.

In addition, the animations generated from Fibonacci natural curves and the use of radial boundaries would and could work well both in functionality and aesthetically as the mesh expands and contracts. The helictical properties observed would be quite nice to view in combination with colour changes.

Using the RGBa alpha values on the letters for fade in and out transitions are definitely an option. Since HSV may not be appropriate for Alpha channels in animations, potentially the value model in the StackOverflow solutions [2020] may need to change to accommodate alpha channels else simply a new object or method built into the fade transitions object would be appropriate. They can then be applied to the letters fading per the interface description.

With the correct radial boundaries in place from the network mesh ‘connection points’ in the polygonal structure, it is likely that additional functionality and visual appeal will occur since the boundaries would be strict. As the user moves a point around with a letter attached to it, possible observed behaviour would be of other points and or letters moving out of the way of the users interaction as the boundaries collide.

Most of the tests for the interface generation itself are fine and would not need sandboxing since we aren’t really doing too much which is not already standardised. It is the back-end of the model and control layers which will need to be sandboxed during technical exploration and reporting stages.

**Interface – Algorithm use cases**

In search of appropriate Algorithms, the use of Prim’ variants or Kruskal’ work [Sedgewick & Wayne 2011] might be appropriate. On reading about Prim’ and Prim’ eager MST, though they seem appropriate, the Algorithms as they are, do not suite the mesh use case since they are spanning to shortest path first (spf) and newest vertices (node) in the forest of the spanning tree. With regards to Kruskal, they also do not seem suitable for such use cases when generating a polygonal mesh of vertices and edges.

It seems the edge weights will recalculate and new connections will be formed on drawing or redrawing the connections on the interface. As the vertices move, a lot of the possible connections between each node will be lost or discarded and new connections and weights formed because of the SPF rule in weighted connectivity. It is also noted the Algorithms are for finding efficient paths. We are not trying to find a path of efficiency in this instance, we are trying to draw polygonal structures between vertices of a certain radius or distance.

If the algorithm principals were adhered to, they may prove useful in understanding how to draw the paths, in particular, where the data structuring of path to vertices is concerned. Because of the rule of ‘all vertices must be connected’ [Sedgewick & Wayne 2011] in MST functionality, the possibility and code is there to be used and modified in this instance with the use of ArrayLists.

With those use cases in mind, proximity based connectivity has been decided upon as a hybrid algorithm making use of multi dimensional ArrayLists (MDAL) to store the data and the connectivity relationships semi-permanently in the first instance. The method has been chosen for several factors, the connections need to stay permanent which is not dissimilar to the example data text file (provided for MST in Sedgewick and Wayne’ work) and the data needs to be accessible for alternative functionality such as animations and redrawing linear graphics to represent those connections, thus, creating our mesh.

Particular points of interest are the starting points and the end points of the animation and the MDAL’ required to create, store and represent the information, in some cases three-dimensional ArrayLists are returned and used. What is also interesting is the starting point for the current state of the randomised meshes. The information provided [Porbst and Jain] has really helped as the starting point for those meshes where the rest of the classes were built from, without the screen dimension of the hardware and the knowledge of MDA, the rest of the calculations fall down since they are the foundation of the programmatic starting point.

**Interface – Animation and Transitions**

Further to the topic of Algorithms, in spite of initially deciding upon a set of in-house hybrid Algorithms for the creation of the datasets to be used; The use case of transitions from point A to point B as an addition to animations and data transfer, is going to be interesting.

Already, the vision of actually being able to watch Prim, Fibonacci and Kruskal’ work (amongst others) animated via the randomly generated meshes and even watching the randomised path of mesh A to mesh B via the OS languages access randomisation methods, is something to behold if they can be implemented correctly to the control layer of the interface.

Essentially, animations work from point A to point B and occurrences of change in each transition ‘frame’, something we learn in art lessons at school, in code ‘redrawing’ or ‘regenerating’ the changes and presenting them to the interface, is something which is largely handled by the animation API Google [2022] provide access to.

That does not mean we cannot write algorithms which help and assist in doing that for us. Neither does it mean, we cannot derive certain behaviour from algorithms as transitional patterns to display on the interface. Interestingly, we could set a point A and point B use a Fibonacci as the transition type in the animation from A to B and then use a reverse Fibonacci to reverse the helictical pattern of movement from point B back to point A again.

It will be interesting to observe and find out if we can integrate a mixture of Algorithms to create some aesthetically pleasing animations as the points from the mesh move across the screen.

**Interface – Colour**

After reviewing the two solutions of hue change from StackOverflow [2020], though applicable, the low level nature of the methods and controls though, more than adequate for the project, seemed too complex for the desired result. It is noted through the research into high level colour generation, standard RGB (sRGB), suite the necessary requirements for colour and integration into Android RadialDiameter method [Google, 2022] and canvas.Paint() colouration. However, they do not meet the requirement for generating an interface from Algorithms, necessarily.

When factoring in the abstraction from the very low level languages, they are required to manipulate sine waves into the form of visible light via the hardware;

The higher level languages which abstract from low level into readable code, does still apply, we are simply calling what has already been written and built into functionality to produce the desired visible spectrum of electromagnetic fields (EMF) using electromagnetic pulses (EMP) and various other hardware components.

As such, we could explore the spectrograph generation and solution from John Walker [1996], definitely a solution to explore, adapt and modify in the field of holographics because the code is lower level than higher abstractions. For the current project as it stands in its first iteration however, two solutions have been written to implement the sRGB scales in randomisation of pastel colours for several reasons.

* We do not need something too complex in the first iteration at present
* The two new classes can be reimplemented easily
* They are simple for entry level and young developers to make use of
* They are both effective and work well with two heavily used methods in Android development
* pastel colours are easy on the eyes and look aesthetically subtle

They suite purpose and do not need to interfere with the low level work which is already operating to generate light spectra and will be safe to implement as opposed to a large, heavily engineered set of classes and subclasses required to do the same job and those particular classes will not be sat in the public domain, potentially causing issues in the modern fibre infrastructure we have today.

package com.developing821.i;

import java.util.Random;

public class Colours {

/\*\*

\* Randomly generate an sRGB value in long format using starting point of 150

\* for pastel colours.

\*

\* For use with RadialGradient

\*

\* @return long - colour

\*/

public static long[] gen () {

int max = 105;

int min = 150;

Random rand = new Random(max);

long[] colours = new long[0];

int sum1, sum2, sum3;

for (int i = 0; i < 2; i++) {

sum1 = min + rand.nextInt();

sum2 = min + rand.nextInt();

sum3 = min + rand.nextInt();

colours[i] = Long.valueOf(String.valueOf(sum1) + String.valueOf(sum2) + String.valueOf(sum3));

}

return colours;

}

}

package com.developing821.i;

import java.util.Random;

public class ColourGen {

/\*\*

\* Randomly generate an sRGB value in long format using starting point of 150

\* for pastel colours.

\* @return long - colour

\*/

public static int gen () {

int max = 105;

int min = 150;

Random rand = new Random(max);

int colour = 0;

int sum1, sum2, sum3;

sum1 = min + rand.nextInt();

sum2 = min + rand.nextInt();

sum3 = min + rand.nextInt();

colour = Integer.valueOf(String.valueOf(sum1) + String.valueOf(sum2) + String.valueOf(sum3));

return colour;

}

}

**Interface - Mesh**

When approaching the task of creating the mesh, though it seemed daunting at first, the point by point approach seemed appropriate. Each point is factored in as x and y pixel coordinates in a single dimension which can be translated to an integer grid point using Math. A simple x \* y gives us the square value of the two grid points via the Interface\_Helper class.

The points are then stored in a two-dimensional ArrayList, That particular ArrayList houses the area and circumference methods to be called as proximities for each point. We access the data by calling the index n of the Ith index in the two-dimensional ArrayList.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Index 0 | | Index 1 | Index 2 | Index 3 | Index 4 | Index5 | Index 6 |
| X | Y | Touch Area | Touch Circumf | Green Area | Green Circumf | Red Area | Red Circumf |

The proximities are used as part of the connectivity requests and stored for use with drawing the linear components of the mesh as connectivity pairs. The base template is described in pairB; However, that is not to say multiple connections will not be stored in the 2nd dimension of the ArrayList.

pairB[Touch[

[[x,y],ta,tc,ga,gc,ra,rc], [[x,y],ta,tc,ga,gc,ra,rc]],

[[x,y],ta,tc,ga,gc,ra,rc], [[x,y],ta,tc,ga,gc,ra,rc]],

.... N pairs of connection pairs

], ..... inside connection

Green[],

Red []

]

Doing so makes a partial solution to Prim and Kruskal problem of ‘reconnecting’ based on weighted proximity. Instead we are storing the connections each time the Mesh is generated and thus, mitigating the selective connectivity if the proximity or weights change during user interaction.

Logical connectivity was then written derived from the circumferences making ‘connection’ with another point of the mesh if they are within pixel range, though that may need to be revisited and make use of the area in opposition to circumference. The sorting of connections is derived from horizontal symmetry of the screen making use of our horizontal line. The Horizontal line has become the starting pointA and the randomised mesh distribution points have become the end pointB.



More over, the Interface\_Helper class has been used to restrict scale and add boundaries of the screen to gather matrices of the pixel coordinates of the surface area in relation to the density in pixel (DiP) [Porbst, 2021]. The coordinate system has been coded in place originating from a solution via StackOverflow [2022]. The very beginnings and foundation of the hard/soft system restrictions forming from which the other calculations were derived.

**Mesh – Further development**

Further development of the mesh involves three dimensional proximities making use of Pi. The Neuron, stored in the proximity ArrayList as an Algorithm, much the same as the x,y point. Later we can scale the ‘layers’ of the mesh into a hypervisor or virtualisation layer with hyper threading to compute the artificial representation of data moving from neuron A to neuron(s) N according to proximity, three dimensionally and within computable boundaries.

As such, we could then have a single layer of neurons connected or even scale to the Nth number of layers and all randomly generated. Doing so would generate ‘brain cell’ composites with each neuron housing the same or even variable artificial intelligence algorithms via method pointers.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | Ai - 1 | X | Y | Touch Area | Touch Circ | Green Area | Green Circ | Red Area | Red Circ |
| ... | …... |  |  | …. | …. | …. | …. | …. | …. |
| N | Ai - N |  |  | …. | …. | …. | …. | …. | …. |

Further more, because the connections are stored, we can look back and assess, even at random, which connections there are, which were made, via which AI. Prim and or even Kruskal (Amongst others MST), could be used for analysis of shortest or best path first, including which path was decided by the AI and even implement MST into the AI method.

What we end up with is a two dimensional ArrayList housing methods callbacks for mathematical algorithms and virtual representations of physical neuron locations in the three dimensions of the neural network connectivity with stored connection paths for analysis.

In addition, and not dissimilar to address tables in wireless routers, each node is then capable of storing the ArrayList of each connection it has made for each time we randomise a whole scale of N layer and N neuron with their respective layers, the issue then unfortunately, is losing that data if the three dimensional neural network is shutdown. However, once we have reached a satisfactory variant which perform optimally, we can embed that configuration.

**Mesh – Neuron proof of concept**

When we consider the following example of a memory stored callback function, we must consider the call to add the primitive or method, the result of the method and the call to function [Mishra, 2021].

The test file was written to establish the best method of adding data and the caveats to the current methods deployed in the code which must be refactored and reassessed in small portions, particularly with the Mesh since it relies heavily on method callbacks and the method list.get(index).add(data).

After being tested, it was not the correct way to add the data because there are wrapper and primitive clashes in addition to type errors, we also have lang.exceptions with the primitives [JavaBrahman, 2015]. The following example is proof of concept for callback in Java to Math functions.

package main.java.javatestfiles;

import java.util.\*;

/\*\*

\* The class is designed to test a callback from ArrayList memory

\* storage. A method is built which returns a recogniseable value

\* and stored into an indexed memory space, then called via another

\* method.

\*/

public class MemoryCallBack {

public static int Ab() {

int A = 4;

int b = 5;

return A + b;

}

public static void main(String[] args) {

ArrayList<Integer> mc = new ArrayList<>();

mc.add(1);

mc.add(2);

mc.add(Ab());

System.out.print(mc.get(2));

}

}

returns 9

Conveniently, the methods and AI are accessible and easily modified for improvements and further testing as aggregates of the composite programming code; Leaving each method separated from the neuron and accessed via call by each neuron. We have the issue of method overload in that instance.

Other caveats immediately springing to mind, are stack overflows, data loss and corruption and even as the saying goes, ‘getting our wires crossed’ during hyper threading. At present, we haven’t factored in how to perform a live monitoring of the data flow between each neuron. These are of course second iterations from the first project herein.

The biggest concern and errors to factor in are multiple data types stored in an indexed ArrayList and whether that will impede on the operation of method callback, thus another test was performed to establish whether Object types and mixed data can be called.

package main.java.javatestfiles;

import java.util.\*;

/\*\*

\* The class is designed to test a callback from ArrayList memory

\* storage. A method is built which returns a recogniseable value

\* and stored into an indexed memory space, then called via another

\* method.

\*/

public class MemoryCallBackMixed {

public static int Ab() {

int A = 4;

int b = 5;

return A + b;

}

public static void main(String[] args) {

int x = 24;

int y = 12;

ArrayList<Object> coords = new ArrayList();

ArrayList<Object> mc = new ArrayList<>();

coords.add(x);

coords.add(y);

mc.add("Hello");

mc.add(2);

mc.add(Ab());

mc.add(coords);

System.out.println(mc.get(0));

System.out.println(mc.get(1));

System.out.println(mc.get(2));

System.out.println(mc.get(3));

}

}

returns

Hello

2

9

[24, 12]

We know we can call mixed data types but accessing the individual items of the second ArrayList in the second dimension is more difficult, especially if we had multiple methods embedded into the memory space. However, accessing the individual indices of a stored ArrayList inside a memory space of another, poses a problem.

In the modified example created in AIDE android application from TutorialsPoint [2022], we can see it is possible to store multiple data type callbacks if the correct syntax is used and can even call varying data types, as such the current project programming code needed to be revisited and adjusted to accommodate for the information held. Noteworthy implementations of ArrayLists were observed in Baeldung [2019] examples, in particular, the three dimensional ArrayLists.

package main.java.javatestfiles;

import java.util.\*;

//https://www.tutorialspoint.com/multidimensional-collections-in-java

public class MemoryCallBackRemixed {

public static double ring() {

double pi = 3.14159;

int rad = 8;

return pi \* (rad ^ 3);

}

static List multi\_dimensional() {

int z = 2;

int y = 3;

ArrayList<ArrayList<Object> > x = new ArrayList<ArrayList<Object> >();

ArrayList<Object> coords = new ArrayList();

ArrayList<Object> circle = new ArrayList();

coords.add(z);

coords.add(y);

circle.add(ring());

x.add(new ArrayList<Object>());

x.get(0).add(0, 45);

x.add(new ArrayList<Object>(Arrays.asList(56, 67, 89)));

x.get(1).add(0, 67);

x.get(1).add(4, 456);

x.add(2, new ArrayList<>(coords));

x.add(new ArrayList<Object>(Arrays.asList(83, 64, 77)));

x.add(new ArrayList<>(Arrays.asList(8)));

x.add(3, new ArrayList<>(circle));

System.out.println(x.get(1).get(0));

System.out.println(x.get(2).get(1));

System.out.println(x.get(3).get(0));

return x;

}

public static void main(String[] args)

{

System.out.println("The multidimensional arraylist is :");

System.out.println(multi\_dimensional());

}

}

returns

The multidimensional arraylist is :

67

3

34.55749

[[45], [67, 56, 67, 89, 456], [2, 3], [34.55749], [83, 64, 77], [8]]

**Mesh further development – MST**

The principals of Prim, Kruskal and Dijkstra are that of shortest path taken (SPT). It will be interesting to run some batch tests for efficiency from the datasets [Sedgewick & Wayne, 2011], first in their original state and then iterate the complexity to establish the efficiency levels for increasingly complex networks.

Another interesting set of tests would be to run those tests on increasingly complex three dimensional datasets of a similar format, yielding a good reference point for the sort of efficiency we are likely to expect if any where to be used in data transfer or even energy transfer at the hardware level, though we digress at the hardware level.

There are however, points to be considered, from the source (s) given vertices (v) and entropy (T), where are our neurons sending data for end point (E)? Why has the AI selected the Nth E in it’s learning path? Moreover, by which learning method are we going to use and which goal are we going to set? Should we force the paths to start with?

Whilst the indexed layers of the neural net hold their indexed method calls to classes which are also indexed in the following fashion Layer N holds X Neurons and N neurons have N of each type of AI deployed in variance or the same, there is definitely a way to categorise and locate Nth class of Nth neuron of the Nth Layer. That being said we can set N class of each neuron for the MST to find the SPT when data is being sent and could deploy the work of either Prim, Kruskal and Dijkstra amongst others to test their efficiency across a three dimensional mathematical representation of a neural network.

It would also be possible for each neuron’ AI to hold three variants of MST for the purposes of multi processing when sending a request to the other areas of the cortex if one or the other two MST are busy. In addition, it would be possible for the source neuron to offload some of the processing to other neurons in it’s library for assistance if they have the resources available and are not busy processing another request or learning / training.

The indexing of those components to the system lower the cost to method overloading, though increase compute cost, they index each method in the form of ‘AI\_TYPE\_1(), AI\_TYPE\_N()’ of one variant and across indexed layers in the three dimensional neural network and assist with data flow, because predictability of SPT is permitted using probability calculus. Data access to each neuron is controllable during tests and even manually encoded, and because we can predict the SPT and generate a standard deviation in automated and unsupervised tests, we then know approximately which part of the ‘cortex’ is being accessed and which path is likely to be used across the board. As such, we can build log functionality into each neuron.

|  |  |  |  |
| --- | --- | --- | --- |
| AI | Neuron | Side A | B | Layer Index |
| Ai\_X() | X | Probability of A | B | 0 |
| Ai\_N() | Nth Qty | Manual assignment A | B | N |

Finally, what sort of data in the more complex scenario should we use, we have SOFIA, making use of visual, audible and kinetic algorithms, should a new Ai be used after testing the functionality at several complexity levels and what are we going to measure? How are we going to get the neuron to calculate the probability of data loss in transit over complete data loss and mitigate important data being discarded and how to temporarily store the seemingly irrelevant data in neurological sets for compression and decompression? Perhaps convert the data to a mathematical algorithm.

With regards to data collection in a tree path of information requested from multiple neurons, it might be best to tag the traffic in a specific format with each neuron on the path from end neurons to source neuron in such a way the reports also fall into the logs for manual assessment. The proposal is neuron N, output number Z and the probability from all N is W as a mean average and as a standard deviance D, which is calculated via the source Neuron which also reports their own calculation.

We do that because each neuron should calculate those things in order to function in unity with their respective connections and the reports from all neurons which interacted to arrive at that solution. Because they too may have been asked by a different source neuron, to perform that task and so must also function the same way even though the method by which each neuron is learning (the AI) may differ. As such we could consider recurrent learning per neuron.

**Mesh further development – scale**

Because we have the boundaries set with regards to a devices interface and pixel density, the code already written is easily ported to an embedding language. With that in mind, we can also scale the boundaries accordingly, if we want to run batch tests on a ten by ten cube, we can organise that mathematically because of the design.

Whilst mathematical scaling is possible and whilst the concept is not dissimilar to a cpu, the difference is the individual blocks of the cpu are now capable of storing an AI with all of it’s neural connections, three dimensional relations and probability calculus mathematically speaking. What may be possible moving forward and after finding and optimal configuration of ‘intelligence’ is embedding each neuron to individual areas of the cpu at a test bed scale we are content with.

Doing so with the organised storage per neuron, permits us to add additional functionality to each neuron, in example, each neuron, can report which neuron it sent the data to after it was triggered via it’s learning through log reporting, can report is probability calculus, can report even why it decided to do that and what its calculation was causing it to decide on the respective neuron in its three dimensional proximity.

That being said, it’s a lot of logs to go through is something goes wrong. However, that style of logging and reporting would permit us to build graphical representations during testing giving us a human readable and visible representation of what is going on. We would be able to see when performing initial tests of source (s) to endpoint (E) which path was taken, what the reports say, what the logs say and what the probabilities were and assess the outcome.

Some issues we may encounter during the testing phases even at programmatic level are that of clustering. What is likely to happen is the dormant and over connectivity issue through the clustering even in a restricted virtual space. What we are likely to see is neurons randomly generated too far from other neurons and dense connectivity issues in certain cluster areas where there are more neurons in close proximity to others. It is going to be interesting to look at the even distribution techniques of the symmetrical portioning. Yes, we are building a brain.

Of course those scales could be assessed in small batch sizes of four by four cubes or other variants. However, what we lose is a more organic representation of what we are seeking when the neurons are generated randomly. The questions remain, do we change the proximity, causing higher density or do we organise the distribution better?

**Mesh further development and testing**

The biggest issue to address, quite literally is the efficiency. Whilst we have all these incredible assumptions, there are issues, storage, if we look at the code already built as base code, we have compression, stack shifting when data randomly needs to be accessed and logs need to be sent via each neuron; Moreover, the efficiency of decompression during method call from MDAL, connection checks, and various other tasks given the algorithm calling the neuron and its generic functionality.

Small batch tests are going to be required during tests for AI functionality, three neurons at most in example. They are easily monitored, give way for easy deployment of variable or exact replicas of AI and will not overload the processors which we will need to monitor the efficiency with. Because there are few, they’re easily monitored and handled. We could then easily deploy variable data set types or even a single data set to measure performance and learning capabilities at such a small programmatic scale. Like all tests, slow and cautious.

We’re going to have to look at how intensive the randomisation and layer generation is on the processor at initial boot since there is a lot of compression and data calls to the memory locations, and on generation, we could measure that with the same three neuron configuration or simulate a boot with no data, no call to functions to be more specific. A basic generate and store, in the process, we will have to monitor load, efficiency on data change or access and compression decompression via the cpu logs and with use of timers.

Regarding efficiency, when virtualising there is the impacted performance of having to virtualise and emulate an enclosed operating system. However, we can gain a flat rate according the resources allocated to the virtual machine and still have the added advantage of sandboxing if configured correctly and since we are not embedding, that practice is in theory, relatively safe to perform tests with. The issues would occur when network sockets are involved. Also, there is the limitation of the resources allocated as added safety measure to mitigate too much power to the processes and they can be allocated according to necessity during tests.

**Control & Algorithms**

Sorting and Searching Algorithms

* Quick3Way – E.W.Dijskra – p.299
* Union Find (Case Study) – Sedgewick & Wayne – 2011 – p.216 – 240
* BST – p.397 – 415
* Hybrids – Dev – 2022-23
* Breadth First Search – p.538-541
* Depth First Search – 530- 537

Matching Algorithms

* Hybrids – Dev – 2022-2023
* Multiway Merge priority - queue client (Heap Sorting) – Chapter 2, p.308 – 327

Randomising Algorithms

* MST – Prims Algorithm - Vojtěch Jarník – 1930 – p.605 – 622
* RA – p.198
* LV – p.778
* MC – p.776
* Quickselect – 345-347
* Quicksort – p.290, 307
* RKA – p.776
* 3 way string quick sort – p.722
* Random Number p.30-32
* RQDT – p.168

Hue Randomising Algorithms

* Hybrid – Dev – 2022-23

Network algorithms

* Chapter 4
* Maze Explortion – p.530
* Path Finding – p.535 – 537
* Tremaux Explortion – p.530
* Dijkstra’s Algorithm – p.656-657
* Dynamic Connectivity – p.216
* Biconnected – p.562
* Path – p.519

Unique or altered Algorithms per use-case - (**MISC**)

* Hybrids – Dev – 2022-23
* Merge Sort and BT – p.278-281
* Computability – p.910
* Computational Complexity – p.910-918, 279-282
* Connectivity – p.936
* Fail Fast Iterator – p.160, 171
* FIFO policy – p.126

GUI Algorithms

* Circular Rotation – p.114
* Collision Resolution – p.458
* Array – p.18-21, 72
* TBBoA

Vector Algorithms

* Fibonacci Heap – p.628, 682
* Fibonacci Numbers – p.57
* Pythagorean Theorem – TBBoA – p.44-47

**Control an overview**

Batch and efficiency tests to accompany technical reports will need to be done for each Algorithm and their derivatives. It would be wise to sandbox the tests on virtual systems which are not connected to a virtual network after all the necessary development tools and upgrades have been performed. Data should be collected and presented in readable and easily digestible forms.

Further batch and efficiency tests will need to be performed with analysis of behaviour in the closed off system. In addition, virtual networks will need to be setup for network testing and security tests to accompany the behavioural tests if network sockets are used. Furthermore, it is imperative such tests are performed, analysed and reported where the exploratory Algorithms and their derivatives are used if network sockets are used.

Explorative experiments could be performed on localised VM systems and or devices with no network hardware (easily removed) for the purposes of Algorithmic interactions with the interface of such devices, Mobile devices in example.

The integration of the model and interface with the control layer should be considered carefully, particularly if the decision to host and provision the applications features via a central location (PaaS, NaaS and SaaS) is reached.

Reviews of testing devices and systems should be performed as part of audits and a lot of monitoring will be required for live alpha testing of the hard/soft systems. Security engineers may have to be contracted with limitation of access to ‘break into’ the model and networked services as part of pentesting, a technique to try and force the system to break to compromise safety and security levels. Those techniques are usually contracted, with strict rules and boundaries.

To accompany their tests, monitoring tools to monitor their behaviour and ensure they are not breaching their bounds could and probably should be used. Else, as a company / team, perform those tests ourselves.

The integration of the control layer with the model and interface would be the most difficult; Efficiency, performance and behavioural scrutiny is required and forecast of live implementation also required. Perhaps the first iteration should be a localised one on one device and once tested move to live provisioning. As such, the negative results would be kept out of the live domain resulting in continuation of services without impact.

In addition, as a team we have more time to correct erroneous behaviour and deploy mitigation processes and test them further. However, though the internal tests do not outweigh the plethora of environments and real world scenarios, we could certainly make a good go at getting as much of those holes closed in advance. Practices such as coding standards, test cases and secure coding can be deployed well in advance of live implementation.

As a result, it lowers our work load after deployment and permits the team to explore new directions for the next iteration and naturally steer the team / company in its direction.

**Control Functionality**

The standard function calls at set intervals can be set and used with the interface behaviour. The use of time delays for certain behaviours is also a viable option from interface to back-end operations. In addition, range type methods can be used safely in effects such as transitions and movement and or animations.

**Control Functionality - Algorithms**

The use of Algorithms to perform and or even learn efficiency in searching and sorting will obviously need to be tested and measured. In combination with the storage and structure of the data to be used, research into efficiency learning through machine learning will contribute to wider technical reporting for such patterns.

Such Algorithms already exist in data centres and search engines to optimise the efficiency in returned results today from seemingly simple scraping algorithms to very large and complicated artificial intelligences deemed inaccessible and even proprietary with copyrights from large corporations such as Google and Microsoft. On the other hand, open solutions are potentially available from the OpenAi foundation and will have to be ported (rewritten in a different language for the use-case).

In addition, several ‘in-house’ functionality can be added to generate the interface as opposed to the use of xml, in particular for performing calculations relative to the screen area and x,y coordinates 0when placing vertices and calculating drawables within the bounds of those restrictions.

Furthermore, those calculations will be useful when deciding upon and writing Algorithms within the control and interface layers.

**Control Functionality – Data Structures**

With regards to the data structure of the dictionaries, current attempts so far for this project, have yielded much larger file sizes than the original source dictionary file when attempting to output the data to a different structure from say .txt to .xml kvp. In all cases, from MB to GB in difference between to source and output respectively.

As such, keeping the data in simple line by line .txt format would be beneficial if the data is to be stored on the device or even in a container. Reducing the need for complex parser method(s) and improving efficiency in search and sort right from the off and even reducing the storage size requirements.

**Control Functionality - Approach**

Object-orientated programming in combination with procedural and functional is certainly an appropriate hybrid approach to the completion and functionality of the control layer. As the functionality proceeds to call objects to act on the data, the returns can be piped back to the interface class functionality for end-user interpretation.

Standard functions and logical switches can and will be used to interpret, interoperate with and even control and move data form the back through control to the front-end. Some encapsulation of data and data access between each class may and probably will even be required to prevent data leaks and loss of data in transit.

**Control Functionality – Processing and Storage**

Deciding whether or not to act on the device APU or remote (if used) device resources, will inevitable either depend on our own research, efficiency and the operating systems native decisions. The preferred method is to allow the native decisions of the operating systems to decide which parts of the control layers are put where. The typical is stack, through to C/G/APU depending on what is being processed. For a larger game, the typical is to force graphics to be processed by the GPU and sometimes even some of the learning algorithms depending on efficiency in frame rates and response times or even ‘intelligence’ of the ‘target enemy’.

**Control Functionality – So Far**

For this project in it’s first iteration, we will let the operating system decide where to set the processing and simply control the data flow, presentation and interoperation of the Algorithms. As a result, we do not need to interrupt any other processes, we minimise our interaction with data we should not have access to or damage, and due to the relatively simple use of graphics rendering, do not need to force the graphics processor to do anything out of the ordinary which cannot be simply told by the transition methods.

As a result of Objectifying each method into pockets of functionality, each can be tested and easily observed should any issues arise thus making it easier for developers to examine and improve or even simply read and document the logic. That being said, each should be annotated for documentation and debug purposes, where it sends its output and how it operates on the data.

**Model**

As discussed already in previous sections, the multilingual dictionary will be line by line and probably a limited set. Whether to store locally or remote depends on storage, whether a need for containers in the cloud is required and safety of general data and code.

**Model - Back-end storage**

With regards to the dictionaries or online access to them. It might be feasible to make use of Googles translate libraries to extract and convert the words / symbols to extensible markup libraries if they aren’t already stored as such. From there, relational matching making use of standardised algorithms. In addition, a C program to extract and convert into structured data with key to value paired structures.

Alternatively, enumeration might be feasible or even dictionary lists / array lists encoded in Java files as opposed to xml files or in combination. Testing needs to be performed for efficiency, particularly since they will be potentially stored and processed on the device as opposed to the cloud; However, the security risks of having dictionaries stored on the device in full and accessible pose quite a heavy cyber security threat, as do the storage of algorithms.

**Model - Other Avenues**

Potential avenues to explore are server / data centre containers for remote security purposes and provisioning the app as it is accessed for each user. As such, measurements can be taken from a central location and access to the dictionaries and algorithms can be controlled better.

It might be a good idea to setup a small container for alpha testing and security testing. Penetration tests are essential, since access to a large data set of multilingual dictionaries and a whole plethora of search, sort and network algorithms is a huge security risk, it would be essential to run those tests and security tests. The risk is compared to storing rainbow tables and the tools to use them in one location, not a good idea without locking them down.

Weighing up the pro’s and con’s of horizontal storage to vertical and the trade off of multiple devices trying to hack into a central location or multiple devices having the same data attacking other devices; What is easier to manage and control from a security perspective and which could be used as a security tool better and with more control, the vertical or the horizontal? That is of course, if further iterations move into the security route instead of, in a wild example, holography.

What is interesting is the mapping functionality in Android development, we could hash map one data set to perform certain tests about derivative decompression during the training and even the data access testing phases and compare to line by line search and sort and as such can then compare the learning capabilities as to what is best suited to which type of neurological configuration. There is of course, with other AI, the potential to measure their capabilities in these variable configurations and even test if the AI can detect whether it is receiving a word or an image per configuration after training and test that performance per configuration without telling the neurological configuration, what data it is receiving.

**Model - So far**

To repeat our brief discussion in control functionality, current attempts so far for this project, have yielded much larger file sizes than the original source dictionary in all cases, from MB to GB in difference between the source and output respectively.

As such, we will be keeping the data in simple line by line .txt format to improve efficiency in search and sort right from the off and even reducing the storage size requirements.

Search and sort algorithms [Sedgewick & Wayne, 2011] will be investigated for efficiency and accuracy and further research will be performed to assess other avenues for efficiency and machine learning.

**Safety and Risk factors**

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Details | Mitigation | Value (low1 | high10) |
| Human Error | We make mistakes, trust too much, blame too often, fail to factor in everything, and cannot always pre-empt | Planning, reading, research, batch tests, risk assessments, security tests, closing holes, reports, audits, legal, repeat | 10 |
| Human Safety | Electricity, Fire, ergonomics, well being, (stress, anxiety, fatigue), over exposure to computer screen, exposure to wifi router (proximity based) | Common sense, training, pat testing, moving or removing deadlines, recommendations, discussions (121, group), plenty of breaks, time off | 7 |
| Environmental | Blackouts, environmental events beyond our control, fire, evacuations, flooding, snowed in/out | Wait or receive notifications, else, insurance and work from home plans | 3 |
| External | ByoD, Social Engineering, Theft, Internal Hacking, Internal breaches of security, sickness, legal (copyrights, rights to own code, lack of code referencing), SVN/Git Issues, lack of training, | Background checks, Device policies, Group Policies, Sandboxing, contracts of employment, ensure developers put all references in per block , plagiarism checks, local and remote backups and versioning, work based training, on the job training | 7 |
| Over Development | Too many iterations, not recognising when to disseminate the project and leave as is with just patching, failing to recognise potential incorrect use, lack of corrective, preventative, and responsive measures | Reviews, Audits, Meetings, Monitoring, Liabilities in terms and conditions, standardised processes and procedures, including legal. | 5 |
| Documentation | Incorrectly documenting or not documenting, old documentation not archived, insufficient documentation (not enough detail) | Ensure documentation follows a standard and versioning and is backed up with dates, make time to dedicate to documentation, make certain it is thorough enough | 7 |
| Steak Holders | Pulling out, applying pressure, demanding too much from the MoSCoW, trying to steer in a direction the developers and/or company are not content with, requesting Illegal code behaviour, falsifying authority documentations, theft, lawsuits, legal theft, not paying | Meetings, Compromises, legal, Backup plans, ensure the MoS are done before the CoW, debrief in discussion about the direction of the project, contracts, denial of services for illegal activity, cancellation of contract terms (financial gain on pull out, both sides), solicitors, lawyers, court | 10 |

**Security**

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Details | Mitigation | Value (low1 | high10) |
| Network | DoS, MitM, (Spoof, Cert replication, Proxy Routing), backdoors, outages | Provisioning, TLS/SSL, VPN, VLAN, Account logins, Key Rotation, Auth Standards, 2FA, Multiple containers, Cannot always mitigate outages, backup servers, use of horizontal plane? Planning, Research, Safety and Risk Mitigations, NaaS | 10 |
| Storage | Theft, Unprivileged Access, Misuse, Unauthorised use, Corruption, Failure, lack of backups, lack of space | OWIOWO, Multi-Layered Access (AAAA), RSA, Backups, Planning, Research, Safety and Risk Mitigations, Use of Horizontal Plane, Local Backups, Provisioned Backups, GPG | 7 |
| Vertical | Unauthorised use, Misuse, Debt incurrent, environmental, Unauthorised physical access to hardware, lack of locks on doors, SpoF, social engineering, firmware attacks, | Not using Vertical Services, Efficient Processing, Wake on LAN, Sophisticated Door locks, SaaS, PaaS, Multi Layered Traffic Transport, Get to know your team members, Background checks, Clearance levels, Work on a minimum trust basis or just trust people to want to do the right thing (cautiously) | 7 |
| Horizontal | Multiple variable environments, larger attack plane, misuse of provisioned service, direct access to provisioned service, over processing on end-users device, misuse of their storage, overheating, outages, Social engineering | Work based on the standard OS model as opposed to variants, MIMO, OWIOWO, Monitoring tools (SaaS), Distributed processing, Collective (Group) Policies – Central, Employee, Public, Minimum Trust, Contact Lists, Auth tools | 10 |
| Leak | ByoD, Social Engineering, Theft, Internal Hacking, Internal breaches of security, sickness, legal (copy writes, rights to own code, lack of code referencing), Memory, discarded work (skip divers) | Make use of PaaS, local storage and separate accounts for rights, permit share holding, firing, days off, employee rights, contracts of employment, company devices | 8 |
| Algorithms | Misuse, irrational behaviour (particularly with hybrid or new), misuse, incorrect use, sometimes unpredictable, rewrites and legal | Use Standardised, Batch tests, Functionality and Exploitability tests, Virtual systems, Virtual Networks, Emulations, Sandboxing, Technical Reports, Research, Rewrite, Reverse engineering (as tests and research), behavioural recognition builds (Antidotes) | 10 |
| Dictionaries | Misuse, storage requirements, unauthorised access, legal | Central Storage, Use of controlled libraries from service providers (i.e. Google, Microsoft), Non-local storage on the horizontal plane, multi-layered access (AAAA), RSA, No public releases to control code or backend storage, PaaS, SaaS, NaaS | 10 |
| Environmental | Impact with high levels of processing, physical access to junctions and base stations / switches, outages, social engineering, power outages, | Efficiency tests and technical reports, Research, some cannot be helped so NaaS, PaaS, Multi distribution Points and multi-layered distribution, local storage, Contacts lists, Multi layered Traffic, SPF? | 10 |

**Security in overview**

Further to PaaS, NaaS and SaaS, in-house built systems and monitoring tools would pave way for further opportunities of the project and the potential to further the companies reach into specialist areas thus generating further revenue. In the short term, those systems would be preferred over PaaS, NaaS and SaaS since they are unpaid services. In addition, the in-house systems can be tailored to specific use-case as required and further team members / employees knowledge, stimulate their interests, understanding and improve their professional experience.

The requirement for such systems and tools to be in place, furthers potential to steak holders and may even move the company / team into top-level companies outsource list if they see potential in the team / group as an asset to their own. That however presents issues during their own verification checks.

That is to say however, they might decide to have us work remotely following their own procedures. A route which may or may not be explored depending on our own interests as a team and direction we would like to go with the work.

With such tools already in place, and with their own functionality serving its purpose, there might not be any need for PaaS, NaaS or SaaS. However, that does then bring us to physical security, company policy, contractual obligations, contracts of employment, law, various other tasks legally required to be fulfilled and of course, requirement for steak holders and share holders in the company / team. It does not however, refute PaaS, NaaS or SaaS.

Breaches of security are difficult to cope with, the legal ramifications, evidence required, handing over of all documentation and hard / soft systems to forensics and security professionals is, in itself, breach of security and as such, is sometimes their plan or the plan of another in the first instance by means of social engineering on a large scale.

Cases have most probably occurred whereby which, a ‘criminal’ (security professional) has been sent in purely to get sensitive data which is being held securely for very good reasons on both sides of the case. The professional will breach security to cause a legal case to have the sensitive data legally handed over for further investigation, it happens in lots of different variants of crime, crime prevention and law suits, even in fiction.

The tools developed should in fact protect both the company, team, and public access to sensitive data, they should be capable of mitigating remote attacks and remote breaches of sensitive data. PaaS, NaaS and SaaS systems are designed in practice to do just that. Hosting with well known providers do just that and expose the team to such companies for potential to highlight the team / company to them.

As such, if any law suits are to occur, those providers autonomously hand over the data only after the legal documentation and justification have been suitably met, that is not to say if they see something they will hide it, that is to say, if our team and company is doing nothing wrong, often the provider will fight the legal battle on our behalf with plenty of leverage and even hand over the non-sensitive data for good reasons, even the document you are reading right now. They themselves as service providers have legal obligations.

Physical security is very costly, building rental, lock installation, key cards, security doors, fire doors, extraction fans, large rooms for storing hardware, large offices, network routers and switches, energy bills, insurance, the list is endless.

In some cases even physical security personnel and in very sensitive cases armed security professionals, their training, certifications and licenses and insurance policies, they are considered to be employees in those cases.

The need for testing and research of programming code behaviour and irrational behaviour to be reported is also part of security and should be reviewed once more across the whole section of control and model overviews enforcing the need to plan, research, test, report and audit. The stages leading up to the audits and technical reports are there to protect people and the infrastructure, we don’t want rogue algorithms and new code running without technical documentation and resolutions. We also do not want code running ‘in the wild’ which will cause harm at the wireless distribution points.

Memory leaks and virtualisation leaks are also to be looked into because although technically, the virtualisation technology acts as a hypervisor with multithreading capability and sits ’on top’ of the operating systems; They are still prone to memory leaks and local storage data theft. Particularly without fde. Although a lot of companies opt for fde, there is a marginal performance drop as a result. Furthermore, there is nothing to say a remote accessor via a web backdoor or and application back door (application level attacks) have not already breached the security levels and policies.

On that note, one should be cautious about cookie acceptance from javascript pop-ups, recall there could be a potential 24Mb of cookies stored from a single site via your browser, that is a large computer program if they were to compile. More over, javascript itself has the potential to be a huge security flaw when browsing.

**Development Lifecycle**

The section on development lifecycle is simple, an agility in hybrid approach is preferred to cope with the different scenarios throughout each iteration and stage of development, from the first for loop to the largest automated test case.

Some parts of development are simply a waterfall use case, some are iterative and others are cyclic or agile. Developers learn those skills and associations to each use of the hybrid approach (which encompasses several approaches) as they grow within the team / company.

As such some stresses, pressures and work flow inhibitants are mitigated, permitting a healthier work atmosphere. Some developers are more experienced than others and can recognise a simple waterfall approach, (step by step), in comparison to an agile hybrid (responding rapidly to a tailored algorithm and its behaviour).

**References**

Baeldung, 2019. Multi Dimensional ArrayLists in Java [online] Available from: <https://www.baeldung.com/java-multi-dimensional-arraylist> [1/11/2022].

Benedis-Grab., G. 2020. Irregulars, *Coding Stories: Programming Polygons* [online] Available from: <https://aperiodical.com/2020/06/coding-stories-programming-polygons/> [02/10/2022].

Bouzout. T., 2022. Dictionaries [online] Available from: <https://github.com/titoBouzout/Dictionaries> [14/09/2022].

Code Java,. Minh Ha., N. [2019]. Drawing lines examples with Java Graphics2D [online] Available from: [https://www.codejava.net/java-se/graphics/drawing-lines-examples-with-graphics2d#:~:text=A%20line%20is%20a%20graphics%20primitive%20that%20connects,%28int%20x1%2C%20int%20y1%2C%20int%20x2%2C%20int%20y2%29](https://www.codejava.net/java-se/graphics/drawing-lines-examples-with-graphics2d" \l ":~:text=A line is a graphics primitive that connects,(int x1%2C int y1%2C int x2%2C int y2)) [02/10/2022].

Google. 2022. Animations and Transitions, *Overview* [online] Available from: <https://developer.android.com/develop/ui/views/animations> [03/10/2022].

Google. 2022. BlendModeColorFilter [online] Available from: <https://developer.android.com/reference/android/graphics/BlendModeColorFilter> [18/10/2022].

Google. 2022. Fundamentals of testing Android Apps [online] Available from: <https://developer.android.com/training/testing/fundamentals> [02/10/2022].

Google. 2022. RadialGradient [online] Available from: <https://developer.android.com/reference/android/graphics/RadialGradient> [18/10/2022].

Jain., R. 2022. 2D Array List in Java [online] Available from: [https://iq.opengenus.org/2d-array-list-java/#:~:text=Multidimensional%20Array%20List%20It%20is%20a%20collection%20of,of%20elements%20in%20a%20group%20whenever%20we%20want](https://iq.opengenus.org/2d-array-list-java/" \l ":~:text=Multidimensional Array List It is a collection of,of elements in a group whenever we want) [12/10/2022].

JavaBrahman, 2015. How to Resolve – Syntax error, insert “Dimensions” to complete ReferenceType [online] Available from: [https://www.javabrahman.com/corejava/how-to-resolve-syntax-error-insert-dimensions-to-complete-referencetype/#:~:text=If%20you%20are%20getting%20the%20following%20error%20at,generic%20types%20always%20expect%20a%20Wrapper%20Class%20object](https://www.javabrahman.com/corejava/how-to-resolve-syntax-error-insert-dimensions-to-complete-referencetype/" \l ":~:text=If you are getting the following error at,generic types always expect a Wrapper Class object) [31/10/2022].

Lewis. J., Loftus. W., 2012. Java Software Solutions Seventh Ed. *Foundations of program design*. Addison Wesley, Boston, MA.

Mishra., N. 2021. The Java Programmer. *Solve Error insert “Dimensions” to complete ReferenceType in Java* [online] Available from: <https://www.thejavaprogrammer.com/insert-dimensions-to-complete-referencetype/> [31/10/2022].

Princeton., 2022. Package edu.princeton.cs.algs4, *Javadoc* [online]. Available from: [*https://algs4.cs.princeton.edu/code/javadoc/*](https://algs4.cs.princeton.edu/code/javadoc/)[14/09/2022].

Princeton., 2022. Java Algorithms and Clients, *Algorithms 4th Edition* [online] Available from:<https://algs4.cs.princeton.edu/code/> [14/09/2022].

Probst., R. 2021. In Android, how do I set margins in dp programmatically? [online] Available from: <https://stackoverflow.com/questions/12728255/in-android-how-do-i-set-margins-in-dp-programmatically> [12/10/2022].

Regaldo. D., Et al., 2015. Grey Hat Hacking 4th Edition, *The ethical hackers handbook.* McGraw Hill. U.S. IL.

Samsung., 2022. Smartphone Deals & Offers, *New Samsung Phones,* [online] Available from: <https://www.samsung.com/uk/smartphones/?product1=sm-f936bzabeub&product2=sm-f711bzebeua&product3=sm-g990blgdeua> [05/09/2022].

Sedgewick. R., Wayne. K., Et. al. 2011. Algorithms Fourth Ed. Addison Wesley, Princeton University, U.S. NJ.

Sedgewick. R., Wayne. K., Et. al. 2011. algs4 [online] Available from: <https://github.com/kevin-wayne?tab=repositories> [14/09/2022].

StackOverflow. 2020. Android: How to change hue of an Image? [online] Available from: <https://stackoverflow.com/questions/10254630/android-how-to-change-hue-of-an-image> [02/10/2022].

StackOverflow. 2022. Get screen width and height in Android [online] Available from: <https://stackoverflow.com/questions/4743116/get-screen-width-and-height-in-android> [03/10/2022].

StackOverflow. 2020. How to change hue, saturation, and luminance of an image in android [online] Available from: <https://stackoverflow.com/questions/64084953/how-to-change-hue-saturation-and-luminance-of-an-image-in-android> [02/10/2022].

StackOverflow. 2015. How to draw a line by dragging [online] Available from: [https://stackoverflow.com/questions/28291956/how-to-draw-line-in-android-by-dragging#:~:text=the%20best%20solution%20to%20draw%20a%20line%20by,positions.%20Code%20of%20these%20all%20explanation%20is%20below%3A](https://stackoverflow.com/questions/28291956/how-to-draw-line-in-android-by-dragging" \l ":~:text=the best solution to draw a line by,positions. Code of these all explanation is below%3A) [07/10/2022].

StackOverflow. 2011. How to draw a line in android? [online] Available from: <https://stackoverflow.com/questions/5166469/how-to-draw-a-line-in-android> [10/10/2022].

TutorialsPoint, 2022. Multidimensional Collections in Java [online] Available from: <https://www.tutorialspoint.com/multidimensional-collections-in-java> [1/11/2022].

Walker., J 1996. Colour Rendering of Spectra [online] Available from: <https://www.fourmilab.ch/documents/specrend/> [2016].

Willers. M., 2009. The Bedside Book of Algebra. *From vectors to Variables The ABC of X plus Z.* New Burlington Books, London.

https://www.javatpoint.com/java-logger#:~:text=There%20are%20the%20five%20logging%20handlers%20in%20Java%3A,the%20buffer%20log%20records%20resides%20in%20the%20memory.

**Wireframes**

****

**UML Diagrams**

