Creating a physics Library for use in mobile app development.

## By Michael Shalaby

## Supervised by Keith Mannock Student ID: mshala02 Msc Computer Science Project Proposal

## 

Abstract:  
This proposal will outline the design requirements of to create a Physics Library for mobile game development using open source graphics libraries and rendering libraries. The proposed library will be built using kotlin as the language of choice in order to specialise the physics library for android app development. In order to assess the viability of the physics library, the library will require a lot more than unit testing to see if it works. Therefore the ideal testing ground for this library would be to use it to simulate the daring dam-busters raid “Operation Chastise” carried out by the British during WW2.

### Disclaimer:

This proposal is substantially the result of my own work, expressed in my own words, except where explicitly indicated in the text. I give my permission for it to be submitted to the JISC Plagiarism Detection Service.

# History of physical simulations in video games.

Since the introduction of video games in the 1970s, the first physics library came in the form of collision detection as seen in the game Pong (1972). Since then video game physics have seen a lot more development in recent years. These improvements vary by simulating variable motion in earlier racing games as demonstrated in pole position released 1982 to demonstrate how higher speed turning would lead to a larger turning radius to demonstrating the difference in driving on different surfaces like driving on tarmac then driving on grass or gravel.   
Over the following decades, more and more physical properties to video games became more and more mainstream. These would range from more simple applications like detecting collision between two separate objects and the transition from using ray tracing to simulate gun fire to implementing projectile motion to simulate the motion of a projectile like a bullet as travels through the air, calculating its new position at every iteration of time.

# Why build a new physics library for mobile apps

While there are many excellent physics libraries available with a few being suitable for android app development including jbullet, a physics engine ported to java. There isn’t currently an existing physics library built with Kotlin. While java and kotlin both use the same compiler, the main issue java as a programming language face is the fact that it typically uses a lot more boilerplate code than kotlin.   
As kotlin has become considered to be the default language for android development since the release of android studio 3.0 has led to popular IDE’s like Jetbrain’s IntelliJ to utilise a kotlin to java converter to help developers reduce the amount of code they need to write and create less error prone programs for android in comparison to using Java.  
While Java is still being updated today with the latest release of Java 12. It is still considered by many to not be a fully modern programming language whereas kotlin is being viewed more and more as the successor to Java for android development.

# Why simulate Operation Chastise:

During my undergraduate studies in 2013, one of my projects was to build a web app to simulate some aspects of the bouncing bomb operation. However, due to having an extremely limited understanding in programming, the amount of physical properties that was simulated was very limited. However, as my understanding in programming has improved, so has my understanding on how to implement a more realistic simulation of a bouncing bomb by using less constants like having the velocity of the bomb along the i direction dynamically be affected by skimming the surface rather than a flat value of velocity reduced every time it skims.

### A brief history of operation chastise

During the night time on the 16th and 17th of May, the RAF launched one of the most daring bombing raids to destroy three dams located by the industrial heartland of Germany. The reason why dams where chosen in specific was because it was believed that by destroying the dams, it would flood the surrounding areas around the dam and prevent any electricity being generated by the dams. The idea behind this was that by destroying the dams, the British could cripple Germany’s industrial capabilities a lot more effectively than launching conventional bombing raids directly over the city.

### What makes operation chastice the ideal testing ground for the physics library

As the dam where protected by anti-torpedo nets. The bombs dropped where designed to bounce across the lakes in a similar manner to ‘skipping stones’. Thus, negating the effect of torpedo nets underwater. Therefore, this particular event provides many interesting physical aspects to simulate. This would include:

* The height in which the bomb is dropped. During the original operation, the plane was required to fly 60ft above the surface of the water. Therefore, the user can be given the option to adjust the height in which the bomb is dropped from.
* The rate in which the bomb was spinning. During the original operation, the bouncing bomb was required to have a back spin at 700 rpm to increase the speed in which the bottom of the bomb would hit the water. Therefore the user could be given the option to change the rate in which the bomb is spinning in order to see
* As the bombs velocity is entirely dependant on the aircrafts velocity before it is launched, allow the user to adjust the aircrafts air speed to see how that will affect the characteristics of the bouncing bomb

# The physics Library should be able to do:

* Simulate the linear and Rotational Motion of an Object to simulate the motion of the bouncing bomb
* Simulate Collisions between objects and accurately simulate the result of collision in order to simulate the bouncing bomb colliding with the surface of the water and wall of the dam
* Simulate physical properties of both Soft and rigid body objects to simulate how the bouncing bomb will interact with surfaces of different properties i.e. water, concrete or more outlandish surfaces like grass or gravel.
* Be able to incorporate different shapes for different objects with different properties i.e. would the bomb be as effective if it was an oval cylinder rather than a circular cylinder.
* Be able to simulate multiple objects independent of each other in a realistic setting i.e. there was a lot more than one plane involved and therefore a lot more than one bouncing bomb being launched.

# Road Map

This roadmap will explain the main areas of interest for the completion of this project

1. Implement linear and circular motion
2. Simulate different material conditions for objects of different properties
3. Simulate Different mediums to travel through i.e liquid or gas
4. Simulate multiple particles interacting with each other
   1. Multiple particles colliding with each other and how that affects their motion
   2. Simulate elastic (momentum is conserved) vs inelastic
5. Simulate particles interacting with different mediums or none at all
   1. Simulate motion through a vacuum
   2. Simulate motion through gas
   3. Simulate motion through liquids

# Simulating motion

## Linear motion

* Incorporate and explain suvat
* Explain how every equation of motion is dependant on time
* Incorporate variable motion, including differentiation or integration
  + This could take a variety of methods, as there is no way to integrate to an infinitely small number, the euler’s method will be one option to derive distance from varying velocity or the velocity itself from acceleration
  + Another method is the rutte kung methond

## Circular motion

* Incorporate and explain circular motion using suvat

# Collisions

* Briefly explain:
  + Conservation of momentum
  + How inelastic collisions results in energy lost in other areas
    - Incorporate the co-efficient of restitution into the collison models
  + Incorporate non-linear collisions
    - How collisions at an angle can be calculated through the use of I and j vectors (or k vectors in 3d physics)

|  |  |  |
| --- | --- | --- |
| m1v1o - m2v2o | = | m1v1fcosθ1 + m2v2fcosθ2 |

## Friction

Discuss

* Surface friction
  + Fr = µR
* Friction as a result of travelling through a medium

# Programming for motion

## Language of choice

Discus the use of kotlin and why it is the most preferential choice for android development

* How it is the number one choice for android development as of recently
* Advantages over other languages like java
  + Kotlin is both a functional and object oriented language
  + Removes redundencies that may be present in java

## Design patterns

Discuss the design patterns required for the development for the project

* Abstract factory: to implement similar objects that may have similar properties
* Adapter pattern: to adapt different objects that typically has similar behaviour
* Bridge pattern: to allow objects to be implemented independently

## Implementing motion into code

Explain how much of what we discussed in motion can be elaborated into vectors

#### Libraries required

* Open gl
* Kotlin.math (obv)

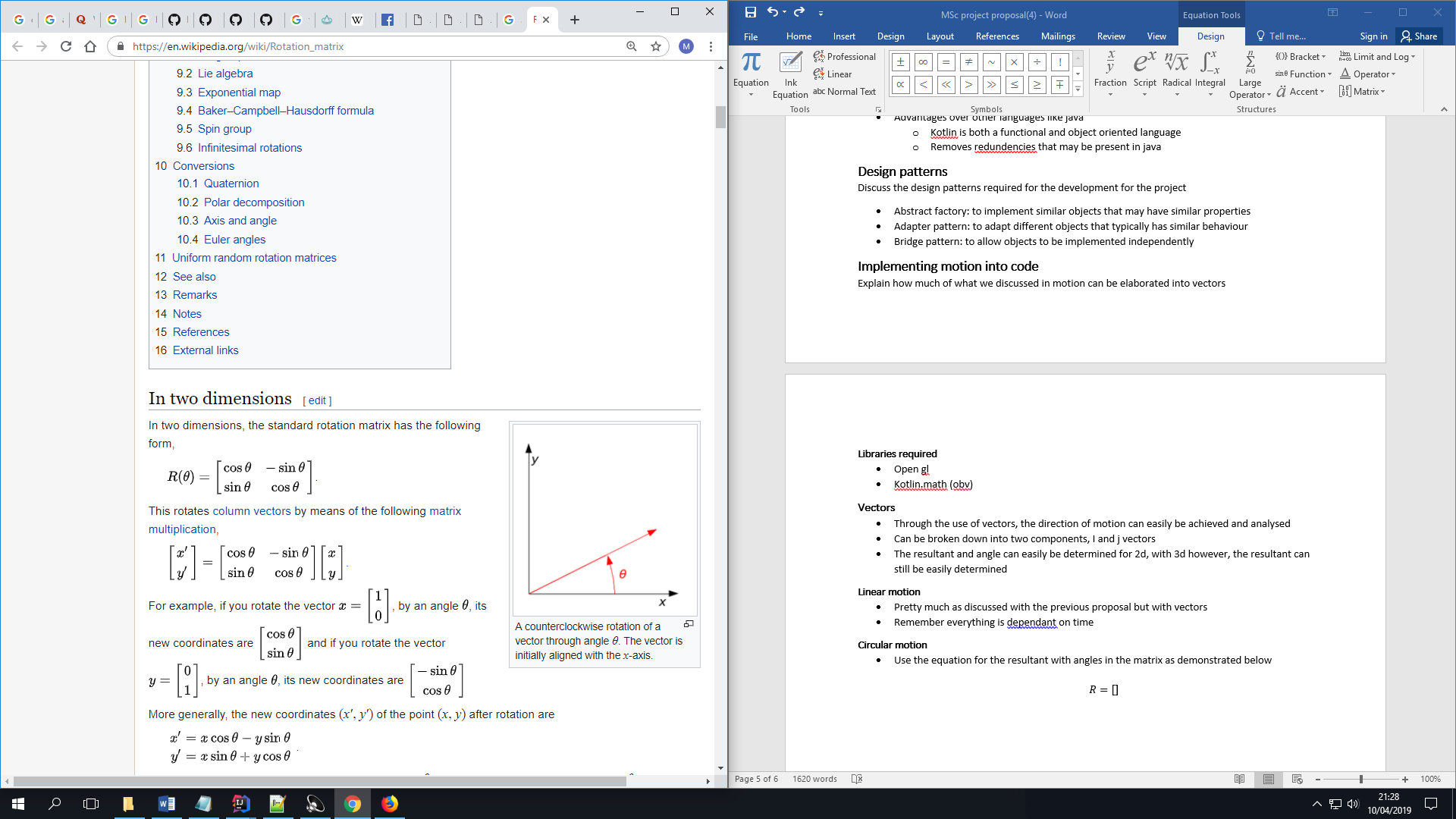
### Vectors

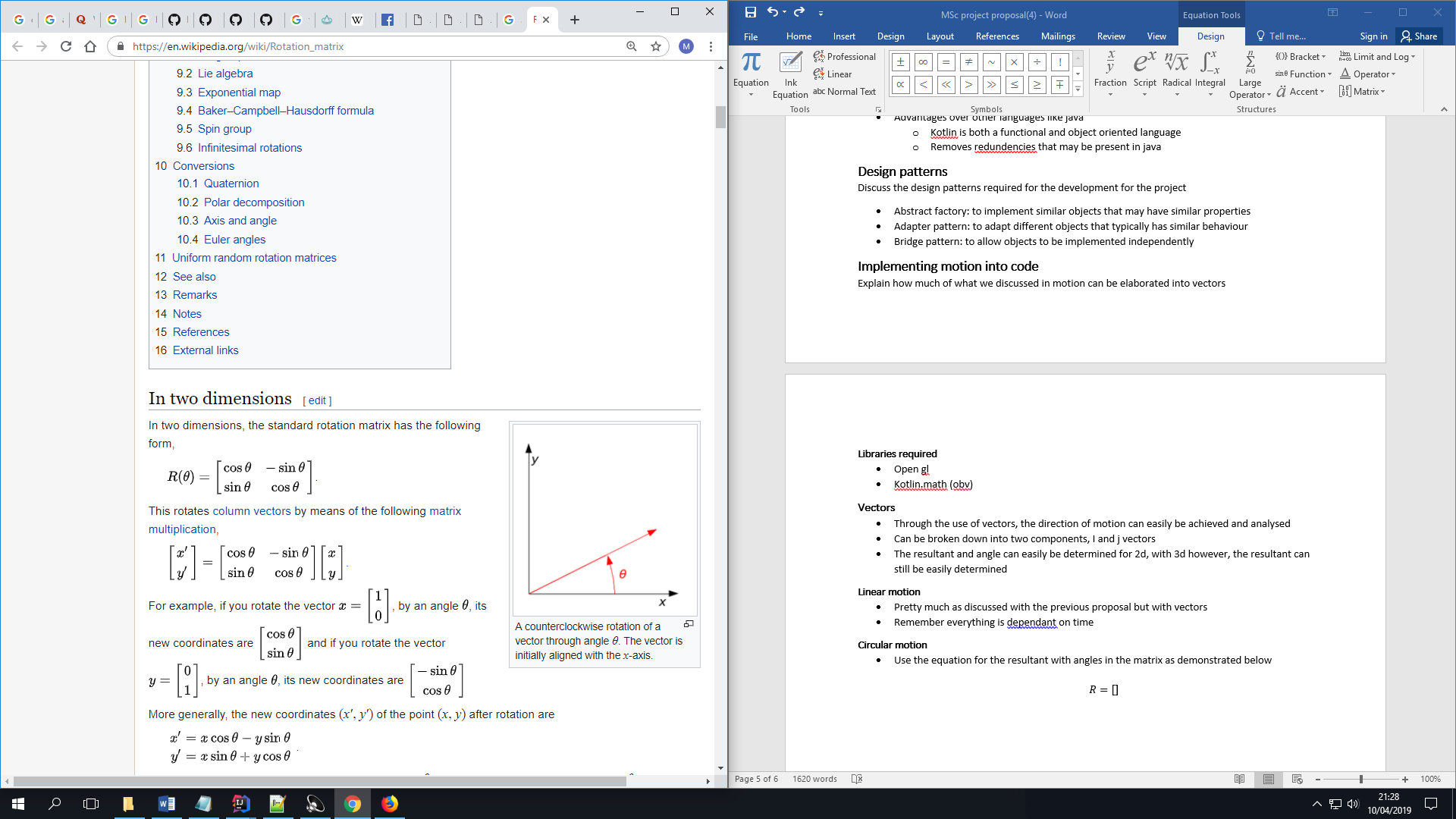
* Through the use of vectors, the direction of motion can easily be achieved and analysed
* Can be broken down into two components, I and j vectors
* The resultant and angle can easily be determined for 2d, with 3d however, the resultant can still be easily determined

#### Linear motion

* Pretty much as discussed with the previous proposal but with vectors
* Remember everything is dependant on time

#### Circular motion

* Use the equation for the resultant with angles in the matrix as demonstrated below  
    
    
  where :  
  R is the resultatnt  
  ϑ is the angle



Where   
x and y is the original co-ordinates (in our case it will be I and j)  
x’ and y’ are the new co-ordinates



1. [Physics for Game Developers, 2nd Edition](https://learning.oreilly.com/library/view/physics-for-game/9781449361037/) by Bryan Bywalec; David M Bourg  
   (viewed from <https://learning.oreilly.com/library/view/physics-for-game/9781449361037/ch02.html>**)**
2. A summarised comparison of kotlin vs java  
   <https://www.konstantinfo.com/blog/kotlin-vs-java/>
3. History of the bouncing bomb operation

<http://www.dambusters.org.uk/the-dam-raids/the-bomb/the-bouncing-bomb/>

1. How the bouncing bomb worked  
   <http://www.chm.bris.ac.uk/webprojects2001/moorcraft/The%20Bouncing%20Bomb.htm>
2. Circular motion  
   <http://web.mit.edu/8.01t/www/materials/modules/chapter06.pdf>
3. Relation of rotation vector and angular velocity  
   <https://physics.stackexchange.com/questions/433102/relation-between-rotation-vector-derivative-and-angular-velocity-when-the-rotati?rq=1>
4. Circular motion equaitons  
   <https://www.khanacademy.org/science/ap-physics-1/ap-centripetal-force-and-gravitation/introduction-to-uniform-circular-motion-ap/a/circular-motion-basics-ap1>
5. Physical simulations computed  
   <https://www.falstad.com/mathphysics.html>
6. <https://gamedevelopment.tutsplus.com/tutorials/how-to-create-a-custom-2d-physics-engine-the-basics-and-impulse-resolution--gamedev-6331>
7. <https://www.toptal.com/game/video-game-physics-part-i-an-introduction-to-rigid-body-dynamics>
8. <https://pet.timetocode.org/>
9. <https://www.intmath.com/differential-equations/11-eulers-method-des.php>
10. <https://github.com/dbacinski/Design-Patterns-In-Kotlin/tree/master/patterns/src/test/kotlin>
11. <http://mathworld.wolfram.com/RotationMatrix.html>SSS