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.

# Chapter 1. Introduction:

In this section, the history and importance of physics libraries will be discussed with regards to how they relate to simulating the daring air raid, Operation chastise.

## 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

On the night of the 16th and 17th of May, the RAF launched one of the most daring bombing raids in World War 2. The objective of this raid was to destroy three dams located by the industrial heartland of Germany. The reason why dams were 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. However, as every dam was protected by a series of torpedo nets, it was crucial for a new type of bomb that would skim over the water to avoid being caught by torpedo nets

### 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. Although this did very little to affect the way the bomb hit the water, it seems more likely due to the Magnus effect generating lift.
* 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 Incorporate:

* Physics models
  + This would require the computation of a variety of physical aspects such as:
    - The initial position of an object in vector form (for 2d <i,j> for 3d <i, j, k>
    - The velocity of an object in vector form
    - The acceleration of an object (acceleration can be derived from the force exerted by an object and the mass assigned to an object which will be discussed in the forces in play)
    - Basic material properties necessary for simulation i.e. the density which will affect the mass of an object depending on its size)
* Objects manager
* Collision detection and response
  + The collision detection model should be able to determine which objects will collide with each other based on the geometry and position of each object
  + The collision response model should incorporate how objects would react when colliding i.e. is there two objects in motion colliding or is an object colliding with a static object, how much energy is lost as a result of collisions and how the rotation of an object would affect the way they respond (note that these will be two independent entities but the collision response is heavily dependent on collision detection)
* Force’s in play
  + These can either be direct, ie the thrust generated from an airline or indirect like surface or air friction and gravity.

# Road Map

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

* In chapter 2, the Background information relating to the mathematics behind real life physical phenomena would be derived and analysed
* In chapter 3, The analysis of different languages for Android applications will be discussed as with how to model physical properties into usable code to model a variety of aspects relating to Operation Chastice
* In chapter 4, The methods of testing the physics library would be discussed as well as how to test the Simulation of operation chastise.
* Chapter 5 will elaborate on how the production schedule of the library as well as detailing when certain aspects of the physics library will be completed and tested.

# Chapter 2: Background

As this project relies almost entirely on physical equations and properties. This chapter would discuss the necessary equations needed to model motion, model collisions and collision response and the internal and external forces in play.   
this will incorporate simple demonstrations of how this is can be implemented using basic examples of code written in javascript (as that is the language the writer has prior to this project).

# Simulating motion

One of the most important aspects of any physics library is to describe how objects in motion work. This can be derived almost entirely from first principle equations. This section will very briefly discuss how these will be derived from first principles and how these can be applied into very basic code with diagrams included.

## Linear motion

* Incorporate and explain suvat
* Explain how every equation of motion is dependent 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 kutte-runge method

## 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
    - Where
* Friction as a result of travelling through a medium (drag)
* D = Cd \* A \* .5 \* r \* V^2

## Discussing lift through different means

* Very briefly explain how lift works using the Bernoulli principle
* Very briefly incorporate this into the magnus effect
* Explain why this is important to the sim (as the bombs had backspin

# 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

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>