MSc Project Proposal

Developing a physics based game engine for use on further game development on the android platform.

Introduction:

History of video game engines

The use of Video game engines has been an integral part of video game development since the 1990s during the growth of 3D games. As time progressed, Video games where able to reach a higher degree of realism as the technology and programming of physics engines improved. This can be demonstrated with the introduction of projectile ballistics in more recent video games that is slowly replacing the older ray-casting model that used to be used for video games.
As a result of the accessibility to game engines for computer's and games consoles alike, the recent growth in indie game development was aided heavily by focussing on the mechanics of the video game rather than having to write a library to build the video game on from scratch.

This trend has been demonstrated the most on the mobile platform as more and more people have access to video games as a result of the explosive growth in mobile phone performance and the customer base surrounding mobile phones. Becoming the new big gaming platform.

History of android app development

talk about how java was and is one of the main programming languages for android. Talk about how java is becoming more and more outdated. Talk about the rise of kotlin as a platform for mobile apps. Talk about the pros and cons of kotlin over java. Talk about why kotlin is used overall. also add how google had announced kotlin as the defacto programming language for mobile app development

Observed shortcomings in Mobile Physics engines

talk about how the vast majority of physics engines are using languages like c# and c++

current lack of support for game development with kotlin as most code will need to be converted to one of the default languages used by popular game engines

Proposed solution

elaborate more on how building a physics engine using kotlin could increase the number of apps using kotlin for development)

Roadmap:

1. *Start with deriving and programming linear motion*
2. *Using the methodology to derive linear motion, apply it to demonstrate how rotational motion is derived*
3. *Understand how to implement linear motion into usable code through the use of vectors i.e. for 2d motion:* $magnitude\ of\ velocity = \sqrt{i^2 + j^2}$ *or 3d motion*

$magnitude\ of\ velocity = \sqrt{i^2 + j^2 + k^2}$
also implement trig equations to fascilitate the calculation of angles

4. *Understand how to implement rotational motion through the use of matrix algebra to simulate the rotation of an object*
5. *Implement boundary mechanics for objects, this should incorporate properties like:*
    a. *Rigid body properties*
    b. *Soft body properties*
6. *Implement a collision model between objects of different boundary mechanics this should include:*
    a. *Collision between different object with different properties*
    b. *Collisions at an angle*

Background

being a former mechanical engineer, explain how this has given you more than adequate background knowledge in understanding physics

Include

- *Strong understanding in deriving more complex equations from first principles*
- *Relatively strong understanding in mathematical and physical concepts including linear and rotational motion, momentum and collisions, fluid dynamics and matrix geometry.*

Analysis, requirement and design:

explain how during 2012 as part of your mechanical engineering degree, you made a realistic 2d simulation of the bouncing bomb and that on top of using test driven development to build your physics engine, the re-creation of this relatively simple game to showcase the performance of the physics engine.

include:

- *How derivations of first principles can be incorporated into code to further develop into more advanced principles (e.g. demonstrating how using a velocity time graph can be used to derive differential expressions for acceleration and displacement)*
- *Then begin the programming by implementing linear motion and expanding on it based on examples of how more complex laws of motion are derived.*
- *Use of openGL for kotlin as the graphics library so that the main focus revolve around the actual physical properties being programmed in the game engine.*

To be completed…