**Talis Lincoln Fonseca Barbalho**

**GAME 205B – Game Dynamics**

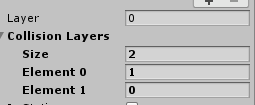
**Final Project – Physics engine**

**Features of the engine:**

* **Manage all the school work with this:**
  + This is only the first semester of the second year, and managing everything from school + creating our own game engine + real life job was insane. I’m sure that working in the game industry will have those crazy days/weeks/months, so I’ve been finding this experience quite amazing, frustrating and satisfying (when the job is done) at the same time.
* **Its on the git repository. *https://github.com/talislincoln/UnityPhysicsPlugin*** 
  + Using source control is mandatory skill for the industry, so I was very happy that this skill would count as mark since I already have a previous experience on doing it. I never really used it for school work, but I started on the day where I almost lost marks because I forgot to submit something, and you told me to start using git.
* **Create a unity plugin was pretty cool:**
  + Looking at your code and understand how each method was being called from unity to c++ actually felt really good. Reading someone else’s code and trying to understand it is something that we should be able to do in the industry, so I also felt that I improved a lot on that point.
* **Time accumulator.**
  + Before really understand how games work, I had never thought as fps differences would an issue with physics. So, the time accumulator is an awesome feature to have. Although I’ve been only using the “bonus” part that you provided us, its was good to understand a little bit more of this.
* **Broad-phase/narrow-phase**
  + I tried to work as much as I could with the code that was given to me. Inside the method UpdateRotationalInertia in the class Polygon.h there was a variable called averageRadius being calculated. I took that variable and transformed it in a member variable of Polygon. Now I have the average radius of each shape, transforming them into a “sphere”. I use the polygon position and that average radius to create a sphere collision as a broad-phase. The goal with that would be not to call the more in-depth collision check when the two polygons weren’t even colliding with the external spheres.



* **Collision layers:**
  + I changed the polygon in unity to inherit from a shape that holds the handle that every shape will have, but I also added the collisions layers in there. Now every shape has a layer and a int vector with all the layers that they can collide with. The the collision layers vector has a size of zero, it will collide with all other layers by default. These collision checks are made in the native library and if two polygons have layers that aren’t supposed to collide the library won’t check for in-depth collisions.



* **Scene:**
  + I changed the scene a little bit to demonstrate how the collisions layer would work as one polygon has the layer to collide with the ground and the other one doesn’t.

**Things that I tried to do:**

* **Support for circle or other collision shapes:**
  + I tried to implement this on unity and even tried to changed the GL draw calls from the debug file in unity, but I couldn’t really understand how that was being made. I would have to take more time to understand a little bit more on how to draw arch instead of drawing just lines and, more importantly, how to apply the physics that we had for those different shapes.
* **Created a Shape class in the native library:**
  + I was going to create a shape class where I would be able to create a sphere or a polygon shape from it, but again, due to the other things to do, I wasn’t able to go that far.
* **Real-time layer changing:**
  + I wanted to add the layer changing to be done in real-time if you changed that as the scene was being play, like unity does for everything else in there, but I also couldn’t finish this.
* **Debug draw**
  + I tried to add simple things such as show the velocity vector, but the line was being draw in a random position, and I couldn’t debug it properly.