Post mortem - William de Beer

For collision detection I created a contact listener class as a child of the box2D b2ContactListener. I wanted to use this for applying and receiving damage in objects. I used the PreSolve virtual function from the parent which would trigger just before a collision takes place. Using this I was able to check the velocity that the objects collided at and hence how much force was behind the collision. If the impact is not great enough, no damage is done. If it goes above that threshold, the damage will be based on the impact power and the mass of the opposing object. This damage is then applied to the object by subtracting from the durability variable which can be accessed through the user data of the object.

The PreSolve function contains the only remaining issue I know of in the release build. Since the object is damaged before the collision actually takes place, a projectile does not collide with an object if it breaks. So the projectile would maintain its velocity through that object and future objects. This was solved for the most part. In the PreSolve function I apply a slow multiplier to an object if it breaks the opposing object. The only issue that remains is the Que-Bird looks a bit strange when going through those structures as it slows down despite it being very heavy and dense. I have thought of a couple of solutions which could potentially reduce or solve the issue. The first solution is for the type of object to be passed in through the user data, so if the projectile was the Que Bird it would ignore the slow multiplier. The other solution is to read the mass and adjust the slowness multiplier based on the mass of the projectile.

My primary strategy for solving box2D related issues was to read the documentation at https://box2d.org/ and compare how they are creating objects to how I'm creating mine. If that does not prove to be useful, I can compare my code to the test bed provided by Erin Catto. As for file related issues, if I could not figure it out myself I would ask one of my classmates to take a look with me. They may have been able to spot something I could not as the solution can sometimes be fairly obscure.

Due to issues early in the project with storing the box2D world in the heap, I had to create it as a local variable and pass it around by reference. If it were in the heap, what I would do for deleting memory is destroy all my world objects in the deconstructor and then destroy the world. But since the world is a local variable and is destroyed as soon as it goes out of scope I had to create a delete function which would delete all world objects in the heap before the end of the main loop function. This function would take care of all joint objects, birds, victims and blocks. Other items stored in the heap such as textures are still deleted using the deconstructor.