Physics Simulation

This is a simulation that demonstrates physics programmed using C++ on Visual Studio.

Research done to make the physics simulation included following tutorials supplied by AIE.

Physics Simulation

Fixed Update Loop

* Fixed Update is a function that gets called on a RigidBody.
* It first takes in a vec2 for gravity and a float for timestep.
* It then starts by decreasing the RigidBody’s velocity by itself multiplied by m\_linearDrag and timestep, simulating friction.
* The RigidBody’s position is then increased by m\_velocity multiplied by timestep, simulating movement.
* The RigidBody’s orientation is then increased by m\_angularVelocity multiplied by timestep, simulating rotation.
* The ApplyForce function is then called, taking in gravity multiplied by m\_mass and timestep to simulate the force of gravity, and (0, 0) for the position on the RigidBody that the force is applied to.
* The RigidBody’s angular velocity is then decreased by itself multiplied by m\_angularDrag and timestep, simulating rotational friction.
* If the affected RigidBody is a trigger, then for all PhysicsObjects in the vector m\_objectsInside, if they’re at the end of m\_objectsInsideThisFrame, a vector of PhysicsObjects, then if triggerExit is true, triggerExit is called on a pointer to the current PhysicsObject, and the current PhysicsObject is erased from m\_objectsInside.
* If the current PhysicsObject is also at the end of m\_objectsInside, the for loop ends. m\_objectsInsideThisFrame is then cleared.
* If m\_velocity and m\_angularVelocity are below the minimum thresholds, they get set to 0. This stops them from moving forever.

Collision Loop

* Resolve Collision is a function that gets called on a RigidBody.
* It first takes in a RigidBody for the second actor, actor2, a vec2 for the point of contact, contact, a pointer to a vec2, collisionNormal, and a float for pen.
* If neither actor is a trigger, the function carries out as normal, otherwise, triggerEnter is called on both actors.
* actor2 gets pushed to the back of m\_objectsInsideThisFrame, followed by this RigidBody.
* A vec2 called normal is defined as the normalisation of actor2’s position minus the position of this RigidBody.
* Another vec2 called relativeVelocity is defined as actor2’s velocity minus this RigidBody’s velocity.
* Another vec2 called perp is defined as normal’s y value and normal’s x value as a negative.
* If the dot of normal and relativeVelocity is greater than or equal to 0, the function returns here.
* A float called r1 is defined as the dot of contact minus this RigidBody’s position, and perp as a negative.
* Another float, this one called r2, is defined the same as r1, but using actor2’s position instead of this RigidBody’s.
* A float called v1 is defined as the dot of this RigidBody’s velocity and normal, minus r1 multiplied by this RigidBody’s angular velocity.
* Another float, this one called v2, is similarly defined as the dot of actor2’s velocity and normal, plus r2 multiplied by actor’s angular velocity.
* If v1 is greater than v2, then the following floats are defined.
* mass1 is defined as 1.0f divided by 1.0f divided by this RigidBody’s mass plus r1 \* r1 divided by this RigidBody’s moment.
* mass2 is defined as 1.0f divided by 1.0f divided by RigidBody’s mass plus r2 \* r2 divided by actor2’s moment.
* A glm::vec2 called force is then defined as this RigidBody’s elasticity plus actor2’s elasticity multiplied by mass1 multiplied by mass 2 divided by mass1 plus mass2 multiplied by v1 minus v2 multiplied by normal.
* ApplyForce is then called on this RigidBody using force as a negative, and contact minus this RigidBody’s position.
* ApplyForce is then also called on actor2, using force, and contact minus actor2’s position.
* If collisionCallback isn’t null, collisionCallback gets called on actor2.
* If actor2’s collisionCallback returns true, it gets called on this RigidBody.
* If pen is greater than 0, ApplyContactForces is called using this RigidBody, actor2, normal, and pen.

Management of PhysicsObjects

PhysicsObjects are managed in a vector of pointers called m\_actors. This allows them to be easily accessed, added, and removed from the PhysicsScene.

Collision Callbacks

collisionCallback is a function that can be used to check for triggers when 2 actors collide. This allows for more reactions to be added to apps made using the engine.

Third Party Libraries

Third party libraries used in the physics sim include GLM and AIE’s Bootstrap.

Potential Improvements

The collision between PhysicsObjects could be made more resilient, as often when testing the engine a strong enough force could cause objects to pass through each other.

