Custom Physics Simulation

Documentation for My Custom Physics Simulation

# Demonstration Brief

The capabilities of my custom physics simulator is demonstrated by my application in the following ways:

* Creation of dynamic and static physics objects
  + By clicking the left mouse button, a dynamic physics object with a random velocity is created, represented by a blue circle outline which moves along the screen.
  + By clicking the right mouse button, a static physics object is created, represented by an orange circle with stays in place, regardless of forces put upon it.
* Addition of forces to physics objects
  + Gravity can be turned on or off by clicking the center mouse button, which adjusts the forces put on the physics objects in the scene. The gravity applied can also be changed with the arrow buttons. This is represented by the dynamic physics objects moving down or up on the screen in accordance with the gravity applied.
  + There are also forces applied when collisions occur, as detailed below.
* Collision detection and resolution between circles
  + When collisions occur, forces are applied to both objects involved in the collision to push them away from each other (unless one or both objects are static, in which case only dynamic objects are shifted). This is demonstrated by the dynamic circles on the screen “bouncing” off of each other or sitting on top of static objects if their velocity isn’t strong enough to pull them one way or the other.

Graphical user interface

Description automatically generated with medium confidence

# Public API

## baseGame

float targetFixedStep

The target amount of time between time steps as expressed in seconds. Evaluated by the shouldTickFixed() function to determine if a physics update is due.

baseGame()

Initializes all variables to suitable defaults. Note that the fixedTimeStep is not initialized – this is up to the user to do in their own game loop.

void init()

Creates game window and initializes the rendering context.

void tick()

What should occur each frame. Polls for player input, updates timers, etc.

void tickFixed()

What should occur each time that a fixed tick occurs, which is determined by the number of seconds listed in targetFixedStep. Primarilly used for physics updates to objects.

void draw() const

Draws the current world state.

void exit()

Closes the game window, destroying all shapes.

bool shouldClose() const

Returns true if the game is trying to shut down.

bool shouldTickFixed() const

Returns true when enough time has passed for a fixed tick to occur.

## physObject

glm::vec2 pos

The position of the associated object.

glm::vec2 vel

The velocity of the associated object.

bool gravity

A boolean value set to all objects in the scene, true if gravity should be applied and false if not.

glm::vec2 gravPull

The gravitational pull applied to all objects in the scene that have gravity applied to them. If the object is static or the boolean gravity is set to false, this value can still be changed but has no effect on the scene.

float mass

The mass of the associated object, which determines how quickly the objects move when a force is applied. Note: this is hard-coded at a value of 1.0f for all objects, though this could be changed to a random value if need be.

bool isStatic

A boolean value set to all objects in the scene, true if the object should not move regardless of forces applied to it, and false if the object should be moved when forces are applied.

shape collider

Gives the object a shape (per the shape struct within Shapes.h), which in turn gives the shape a shapetype, and access to a radius (in case the shape is a circle) and a length, min, and max (in case the shape is an aabb, which is currently not fully implemented)

physObject()

Initializes all variables for a shape as listed above, with all booleans set to false, a position of (0,0), and no forces other than the world gravity (initialized to 50) being applied. Note: when an object is instantiated, its position is set to the position of the cursor on the screen, but as an error could occur if the position is not initialized, it is set by default to (0,0) to avoid issues.

void tickPhys(float delta)

Applies physics to each object every time a fixed tick occurs.

void addForce(glm::vec2 force)

Applies a continuous force to the associated object that considers said object’s mass.

void addAcceleration(glm::vec2 force)

Applies a continuous for to the associated object, regardless of its mass.

void addImpulse(glm::vec2 force)

Applies an instantaneous force to the associated object that considers said object’s mass.

void addVelocity(glm::vec2 force)

Applies an instantaneous for to the associated object, regardless of its mass.

void draw() const

Draws aspects of the world other than the objects within it, such as setting a background color or putting text on the screen.

# Potential Future Improvements

I hope to come back in the future to implement functioning AABB-AABB collisions and Circle-AABB collisions, as well as more elements to make the application function as more of a game than a simulation.

# Credits

Josie DeRosier © 2021