Custom Physics Simulation

Documentation for My Custom Physics Simulation

Trey Gleason

This is a stub of what could be the beginnings of your documentation for your custom physics simulation.

# Demonstration Brief

-Implementation of static and dynamic rigid body physics that interact together:

static square (middle mouse click), dynamic square (right mouse click) and dynamic circles (left mouse click). Static square will stay in place no matter what dynamic shape collides with it.

-Forces applied to physics bodies:

-addForce()

-addImpulse()

-addVelocity()

-addAccel()

-useGravity()

-Visualisation of physics bodies:

-circles and aabbs

# Class DiagramA picture containing graphical user interface Description automatically generated

# Research Material

# <http://www.randygaul.net/2013/03/28/custom-physics-engine-part-2-manifold-generation/>

<https://learnopengl.com/In-Practice/2D-Game/Collisions/Collision-detection>

# Third Party Libraries

Raylib: <https://www.raylib.com/> , <https://github.com/raysan5/raylib>

Glm: <https://github.com/g-truc/glm>

# Public API

## game

### game() - constructor

Initialize 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.

### float fixedTimeStep

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

### void init()

Create our window and initializes our rendering context.

void draw()

Draws the words current state

void exit()

Closes the game and shutdown

void tick()

poll for input, update timers, etc.

void shouldTickFixed()

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

bool isStatic

checks to see if an object should be affected by forces from other objects or not

physObject()

initializes variables to suitable defaults

void addForce()

Adds a continuous force to the rigidbody, using its mass

void addImpulse()

Adds an instant force impulse to the rigidbody, using its mass

void addVelocity()

Add an instant velocity change to the rigidbody, ignoring its mass

void addAccel()

Adds a continuous acceleration to the rigidbody, ignoring its mass

void useGravity()

using bool gravityOn to choose whether or not a object should be subjected to gravitational forces

float resolveCollision()

given two physics objects return the force needed to be applied

bool checkCircleCircle()

uses shapes radius and distance between the two shapes to know if they are colliding or not

# bool checkAABBAABB()

looks for collision between abb and another aabb

bool checkCircleAABB

checks to see if there is collision between a circle and aabb

glm::vec2 depenetrateCircleCircle

calculates the direction and amount that two circles need to separate from one another

glm::vec2 depenetrateAABBAABB

calculates the direction and amount that two aabbs need to separate from one another

glm::vec2 depenetrateCircleAABB

calculates the direction and amount that a circle and aabb need to separate from one another

# Potential Future Improvements

Please identify any future optimizations, bug fixes, or enhancements that could be done to improve the usability of this custom physics simulation.

Unity project:

I would Improve my UI. Change key mapping for some of my slime controls. Add in working a jellybody. Also would try adding the ability to pickup and carry slimes.

C++ custom physics simulation:

Add borders at the ends of the screen that would either cause shapes to teleport from one end to the opposite end of the screen or bounce off bounce off the edges.

# Credits

Trey Gleason © 2021