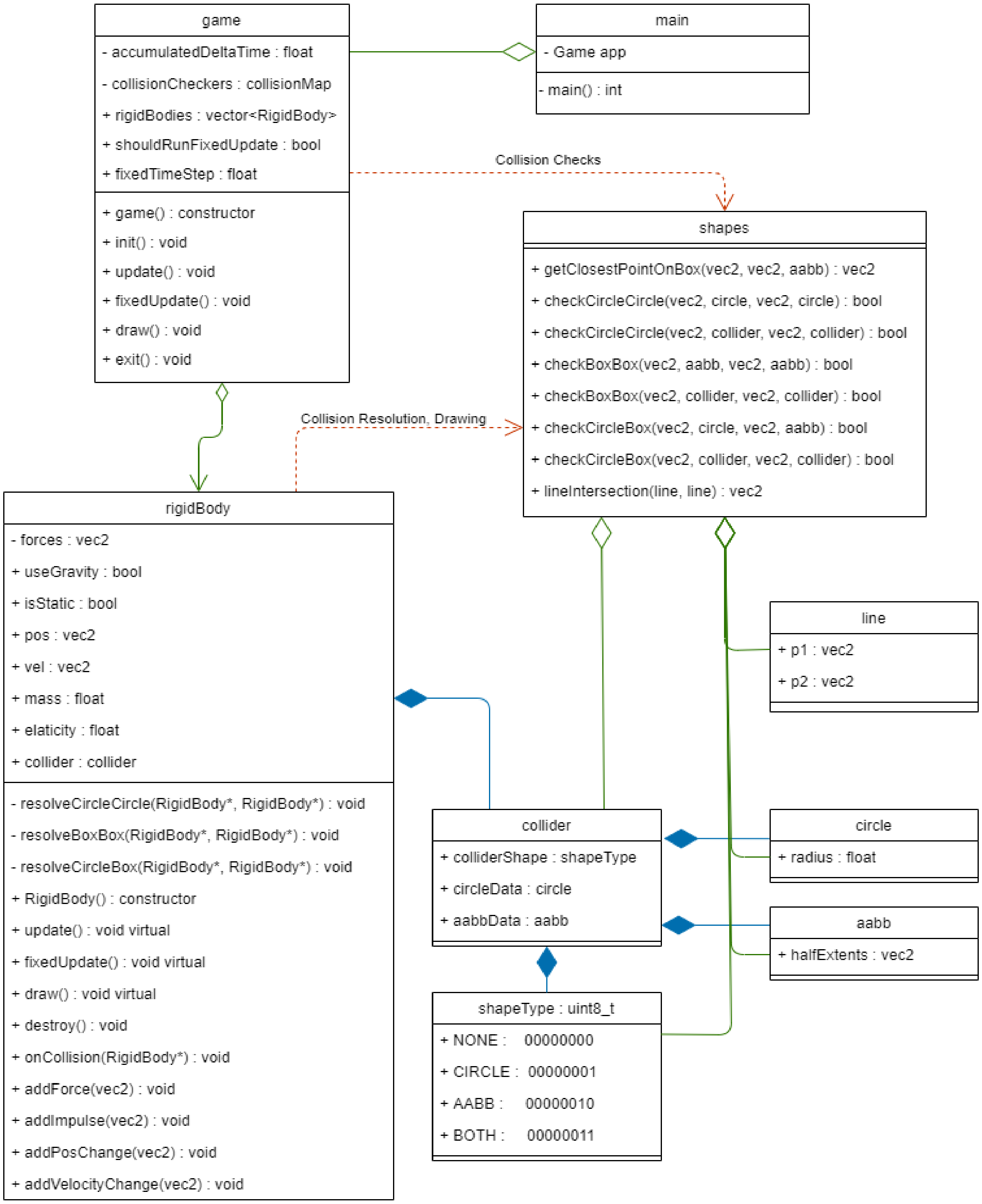
Custom Physics Simulation

Tristin Newby

Demonstration Brief

Just a basic window that lets you spawn in circles by left clicking and squares by right clicking. Hold left control to disable gravity on newly spawned objects.

Class Diagram


game
- accumulatedDeltaTime : float
- collisionCheckers : collisionMap
+ rigidBodies : vector<RigidBody>
+ shouldRunFixedUpdate : bool
+ fixedTimeStep : float
+ game() : constructor
+ init() : void
+ update() : void
+ fixedUpdate() : void
+ draw() : void
+ exit() : void
main
- Game app
- main() : int
Collision Checks
shapes
+ getClosestPointOnBox(vec2, vec2, aabb) : vec2
+ checkCircleCircle(vec2, circle, vec2, circle) : bool
+ checkCircleCircle(vec2, collider, vec2, collider) : bool
+ checkBoxBox(vec2, aabb, vec2, aabb) : bool
+ checkBoxBox(vec2, collider, vec2, collider) : bool
+ checkCircleBox(vec2, circle, vec2, aabb) : bool
+ checkCircleBox(vec2, collider, vec2, collider) : bool
+ lineIntersection(line, line) : vec2
Collision Resolution, Drawing
rigidBody
- forces : vec2
+ useGravity : bool
+ isStatic : bool
+ pos : vec2
+ vel : vec2
+ mass : float
+ elaticity : float
+ collider : collider
- resolveCircleCircle(RigidBody*, RigidBody*) : void
- resolveBoxBox(RigidBody*, RigidBody*) : void
- resolveCircleBox(RigidBody*, RigidBody*) : void
+ RigidBody() : constructor
+ update() : void virtual
+ fixedUpdate() : void virtual
+ draw() : void virtual
+ destroy() : void
+ onCollision(RigidBody*) : void
+ addForce(vec2) : void
+ addImpulse(vec2) : void
+ addPosChange(vec2) : void
+ addVelocityChange(vec2) : void
line
+ p1 : vec2
+ p2 : vec2
collider
+ colliderShape : shapeType
+ circleData : circle
+ aabbData : aabb
circle
+ radius : float
aabb
+ halfExtents : vec2
shapeType : uint8_t
+ NONE : 00000000
+ CIRCLE : 00000001
+ AABB : 00000010
+ BOTH : 00000011

Research Material

https://www.youtube.com/watch?v=3HjO_RGIjCU&t=2s - Line Intersection
https://en.cppreference.com/w/cpp/container/vector - Vectors

Third Party Libraries

https://github.com/g-truc/glm - Math library used in project
https://github.com/AIE-Seattle-Prog/raygame - Window Rendering library used

Public API

class game

vector<Rigidbody> rigidBodies

Container of every RigidBody in the Game. After being spawned they should be added to this container. A RigidBody will erase itself from the vector if the Destroy() function is called.

bool shouldRunFixedUpdate

Flag that gets set to true every fixedTimeStep seconds. This can be set true manually for an extra fixedUpdate() call. This will not alter the timing of the next fixedUpdate() call.

float fixedTimeStep

The target amount of time between every time fixedUpdate() is called.

game() - constructor

Initialize all variables to suitable defaults.

void init()

Create our window and initialize our rendering context.

void update()

Called 60 times a second. Used for updating all the logic of our game.

void fixedUpdate()

Called every fixedTimeStep amount of seconds. Use for handling physics.

void draw()

Calls the draw() function on every RigidBody.

void exit()

Shuts down game and closes window.

class RigidBody

bool useGravity

If true this RigidBody will constantly be pulled downward.

bool isStatic

If true this RigidBody will not move around by physics at all and can only move by manually changing its pos.

vec2 pos

The position in world space of this RigidBody.

vec2 vel
Velocity of RigidBody in window pixels per second.

float mass
Mass of the RigidBody.

float elaticity
The amount of energy kept after bouncing off another object 0.0 – 1.0.

collider collider
The collider and shape of the RigidBody.

RigidBody() - constructor
Initializes all variables to default values.

void virtual update()
Updates logic of the RigidBody.

void virtual fixedUpdate()
Updates physics, pos, and forces of the RigidBody.

void virtual draw()
Draws the RigidBody's collider.

void destroy()
Erases this RigidBody from game::rigidbodies deleting it from the game.

void onCollision(RigidBody* other)
Called when this RigidBody contacts another RigidBody, then calls the proper resolution function.

void addForce(vec2 force)
Creates a recurring force to alter the RigidBody's vel. Does not affect static RigidBodies.

void addImpulse(vec2 Impulse)
Instantly pushes the impulse against the RigidBody. Does not affect static RigidBodies.

void addPosChange(vec2 change)
Directly moves the RigidBody's pos regardless of mass and other forces. Does not affect static RigidBodies.

void addVelocityChange(vec2 velChange)
Directly changes the RigidBody's vel regardless of mass and other forces. Does not affect static RigidBodies.

shapes.h

vec2 getClosetPointOnBox(vec2 pos, vec2 posBox, aabb box)
Returns the point within the aabb that is closest to pos in local space relative to the aabb.

bool checkCircleCircle(vec2 posA, circle circleA, vec2 posB, circle circleB)
Returns true if the circles are colliding.

bool checkCircleCircle(vec2 posA, collider circleA, vec2 posB, collider circleB)
Returns true if the circles are colliding. Colliders must contain circleData.

bool checkBoxBox(vec2 posA, aabb boxA, vec2 posB, aabb boxB)
Returns true if the boxes are colliding.

bool checkBoxBox(vec2 posA, collider boxA, vec2 posB, collider boxB)
Returns true if the boxes are colliding. Colliders must contain aabbData.

bool checkCircleBox(vec2 posA, circle circle, vec2 posB, aabb box)
Returns true if the circle and box are colliding.

bool checkCircleBox(vec2 posA, collider circle, vec2 posB, collider box)
Returns true if the circle and box are colliding. Colliders must contain circleData and aabbData respectively.

bool checkCirclePoint(vec2 pos, vec2 posCirc, circle circle)
Returns true if the point is within the bounds of the circle.

bool checkBoxPoint(vec2 pos, vec2 posBox, aabb box)
Returns true if the point is within the bounds of the box.

bool checkPoint(vec2 pos, vec2 posObj, collider col)
Returns true if the point is within the bounds of the collider. Can use either circle or aabb.

vec2 lineIntersection(line a, line b)
Returns the point of intersection for the two lines. Parallel lines will return a vec2 of { 0,0 }.

struct collider

shapeType colliderShape
Stores what data should be accessed, either CIRCLE or AABB. Both circleData and aabbData read/write to the same memory location so only one can be used per collider.

circle circleData
Stores circle data.

aabb aabbData
Stores aabb data.

struct circle

float radius
Radius of the circle.

struct aabb

vec2 halfExtents
Size of the aabb divided by 2.

struct shapeType : uint8_t

NONE
Shorthand for 00000000.

CIRCLE
Shorthand for 00000001.

AABB
Shorthand for 00000010.

BOTH
Shorthand for 00000011.

struct line

vec2 p1
One point on the line.

vec2 p2
Second point on the line.

Potential Future Improvements

- Add raycasting.
- Add support for more shapes.
- Add spatial partitioning.

Credits