Tarquin Pereira CVA i7217082

Particle physics simulation

Intro
This project will be a physics simulation of spherical particles that are affected by gravity, collisions, air resistance and friction. Particles are emitted from a particle emitter and equations are used to evaluate the position of the particles after each frame, depending on their attributes such as mass and size. Multiple particle emitters can be set up and particles can interact with each other with collisions and a ground plane.

Particle emitters
These are points in 3D space that emit particles randomly in all directions. They will each have settings that determine the number of particles emitted per second, life span(s), initial velocity(m/s), size, mass(kg), drag and friction. These settings can also be randomised to make the simulation look less uniform, which will be done by multiplying each value with an inputted random float between 0.001 and 1 (0 is not used as a particle cannot have 0 mass or size).

Launching the particles
Particles are launched using the initial velocity setting of the particle emitter and a random unit vector for direction. In order to apply the physics equations to them, the velocity should be split into its x, y and z components which can be done by multiplying the random unit vector by the initial velocity.

```
float array directionVector[3]
rand directionVector[1]
rand directionVector[2]
rand directionVector[3]
input initialVelocity
unitScale = sqrt(directionVector[1]^2
+directionVector[2]^2
+directionVector[3]^2)
unitDirectionVector[1] = directionVector[1] / unitScale
unitDirectionVector[2] = directionVector[2] / unitScale
unitDirectionVector[3] = directionVector[3] / unitScale
velocityX = unitDirectionVector[1] * InitialVelocity
velocityY = unitDirectionVector[2] * InitialVelocity
velocityZ = unitDirectionVector[3] * InitialVelocity
```

Simulate motion & Gravity
The simulation should run at a maximum of 60fps with each frame incrementing the time by 1/60 seconds. The new XYZ positions of each particle will then be calculated with velocity and SUVAT equations using this time interval and its velocities.
distance = velocity * time
s = ut - 0.5*at^2

Gravity is -9.8 by default though this can be changed by the user.
For a single particle, the algorithm would appear as:

```
float timeInterval = 1/60
float totalTime = 0
positionX = particleEmitter.positionX
```

```
positionY = particleEmitter.positionY
positionZ = particleEmitter.positionZ
Repeat
        totalTime = totalTime + timeInterval
        resultantForceX = dragForce(velocityX) + frictionForce
        resultantForceY = dragForce(velocityY) + frictionForce
        resultantForceZ = dragForce(velocityZ) + frictionForce
        accelerationX = resultantForceX/mass
        accelerationY = (resultantForceY/mass) + gravity
        accelerationZ = resultantForceZ/mass
        positionX = positionX + (velocityX*timeInterval - 0.5*acccelerationX*timeInterval^2)
        positionY = positionY + (velocityY*timeInterval - 0.5*acccelerationY*timeInterval^2)
        positionZ = positionZ + (velocityZ*timeInterval - 0.5*acccelerationZ*timeInterval^2)
Until pause = true
```

Air resistance
The following equation would be used for air resistance:
F = -0.5CpAv^2

F = force that the particle will experience
C = drag coefficient (i.e. drag from user input)
p = air density
A = cross sectional Area (calculated from particle's radius)
v = particle's velocity

The equation f=ma would be used to calculate the acceleration from the particle's resultant force and mass.

Friction
Friction would occur when the particle is on the ground plane (i.e. when it's Y position is 0) which should prevent the particle from sliding, depending on the amount of friction input. The equation used is:
F=μN

μ = friction coefficient (i.e. friction from user input)
N = reaction force (as the plane is a flat surface it is equal to the objects weight, calculated from mass and gravity)

This friction force will be calculated and applied in the same manner as the drag force. As the particle would bounce a few times on the ground plane, the friction force may also be applied on each frame that the particle is in contact with the ground plane during bouncing, which would simulate the ball losing kinetic energy after each bounce.

Collisions
(referenced from http://www.wildbunny.co.uk/blog/2011/04/20/collision-detection-for-dummies/
)

Collisions will be detected after each frame by comparing a particle with other particles and detecting if any have intersected. This is done by comparing the magnitude of the distance between two particles with the sum of their radii with the equation:

p = ||A-B|| – (r1+r2)

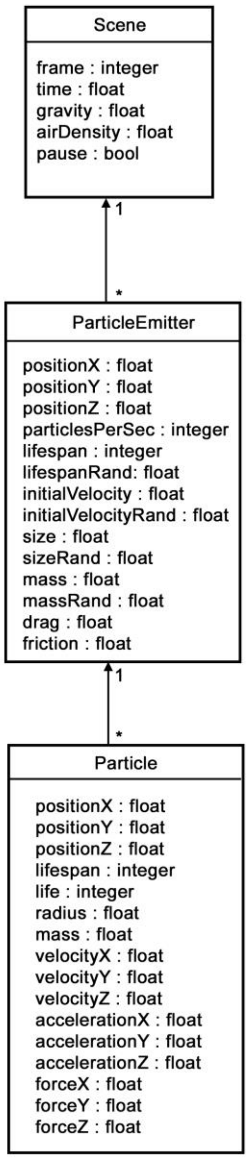
A = coordinates of particle1
B = coordinates of particle1
p = if less than 0, indicates a collision and the distance of penetration
||A-B|| = magnitude of distance between particles
(r1+r2) = sum of radii

To correct the collision, the normal vector must be found:
N = (A-B) / ||A-B||

then multiplied by the penetration distance to create the penetration vector needed to correct the collision:
P = N*p

The new velocity of the particle will then be calculated by multiplying the normal vector by the particle’s component velocities.

Class Diagram



References

SUVAT equations
https://sentynel.com/media/old/equations.html

Air resistance equation
http://www.grc.nasa.gov/WWW/k-12/airplane/falling.html

Collision info
http://www.wildbunny.co.uk/blog/2011/04/20/collision-detection-for-dummies/

Friction equation
http://physics.tutorvista.com/forces/sliding-friction.html