Lab 2: Forces

Published | Edit | ⋮

GPR-350 Game Physics
Instructor: Daniel S. Buckstein
Lab 2: Forces

Summary:
This week we expand upon our particle physics by implementing force generators.

Submission:
Submit a link to your online repository with the completed assignment's branch name and commit ID/index. If you have not created an online repository to keep track of your work, you should do so as part of this assignment; it will be checked.

Instructions:

Step 1: Mass
Add a public mass member (float) to our existing particle interface, and a private inverse mass member which is calculated on start. Mass should never be negative, with a positive value representing a movable particle and zero representing a static particle.
Write a 'set mass' function which locks in any public mass and calculates the reciprocal on start.

Step 2: Force
Add a public force member (Vector2) to the particle interface. This will represent the accumulated sum of all environmental forces (see generators below). The force vector is used to update acceleration using Newton's second law: f = ma -> a = 1/m * f
Write an 'update acceleration' function which converts force to acceleration before resetting force. This should be called in your update after integration.
Write an 'apply force' function which adds a force to the total force acting on the particle (D'Alembert's principle).

Step 3: Force generators
Implement a new interface or static (non-component) class to generate forces. Implement the following functions:

```
// f_gravity = mg
Vector2 GenerateForce_gravity(Vector2 worldUp, float gravitationalConstant, float particleMass);
// f_normal = proj(f_gravity, surfaceNormal_unit)
Vector2 GenerateForce_normal(Vector2 f_gravity, Vector2 surfaceNormal_unit);
// f_sliding = f_gravity + f_normal
Vector2 GenerateForce_sliding(Vector2 f_gravity, Vector2 f_normal);
// f_friction_s = -f_opposing if less than max, else -coeff*f_normal (max amount is coeff*|f_normal|)
Vector2 GenerateForce_friction_static(Vector2 f_normal, Vector2 f_opposing, float frictionCoefficient_static);
// f_friction_k = -coeff*|f_normal| * unit(vel)
Vector2 GenerateForce_friction_kinetic(Vector2 f_normal, Vector2 particleVelocity, float frictionCoefficient_kinetic);
// f_drag = (p * u^2 * area * coeff)/2
Vector2 GenerateForce_drag(Vector2 particleVelocity, Vector2 fluidVelocity, float fluidDensity, float objectArea_crossSection, float objectDragCoefficient);
// f_spring = -coeff*(spring length - spring resting length)
Vector2 GenerateForce_spring(Vector2 particlePosition, Vector2 anchorPosition, float springRestingLength, float springStiffnessCoefficient);
```

Step 4: Test
Implement test scenarios to demonstrate that your force generators work.

Bonus:
Implement some dynamic spring system with damping. Use scenarios in Millington c. 6 for inspiration.

Points	8
Submitting	a text entry box

Due	For	Available from	Until
-	Everyone	-	-

+ Rubric