Lab 1: Particles & Integration

Published Edit ⋮

GPR-350 Game Physics
Instructor: Daniel S. Buckstein
Lab 1: Particles & Integration

Summary:
This week we explore introductory particle physics and integration algorithms in 2D.

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.

Step 1: Define particle
Create a particle class/data structure that will be responsible for controlling motion in 2D, e.g. "Particle2D". A particle should have the following member variables to start:
position (2D vector), velocity (2D vector), acceleration (2D vector), rotation (float), angular velocity (float), angular acceleration (float). Expose the variables to the user so that initial values can be set.

Step 2: Integration algorithms
Define and implement the following functions (choose a return type depending on whether you want them to just perform the action or also return the result):
-updatePositionEulerExplicit(float dt)
-updatePositionKinematic(float dt)
-updateRotationEulerExplicit(float dt)
-updateRotationKinematic(float dt)
Each function should update the specified variable by integrating the appropriate derivative using the selected method, then update respective velocity using explicit Euler/first-order.

Step 3: Update
In the particle's main update function, start by calling the preferred integration method, passing the global delta-time (dt) as an argument. Next, ensure the graphics system has

access to the results; for example, in Unity, set the transform component's position X and Y values to the particle's position (2D), and set the rotation Z value to the rotation (Z-axis controls rotation in a 2D system).

Step 4: Demo
Implement a simple scene that demonstrates a set of particles moving. Implement a test function in the particle interface to control acceleration/angular acceleration and let your integrator solve position and rotation. Use starting values for each particle that will produce a known end result. For example: to make particles oscillate as if controlled by a sine function, the velocity about the oscillation axis would be the derivative of sine and the acceleration would be the second derivative; don't forget that this is an initial value problem so set starting values accordingly.

Bonus: Bells and whistles
Implement friendlier user controls for a particle. For example: add a drop-down menu to allow the user to select the integration method used; create another class that is responsible for managing a set of particles or telling them to update in a certain way; etc.

Points 8

Submitting a text entry box

Due	For	Available from	Until
-	Everyone	-	-

+ Rubric