A2: Animation and Physics

Course: CPSC 427 - Sep 2023
Due: see course schedule

1 Introduction

The goal of this assignment is to introduce you to basic 2D animation. You will extend the salmon game you made for Assignment 1 by including in it a basic particle system implementation.

2 Template

You should use your own Assignment 1 code as a starting point. You will find comments throughout the files to help you guide in the right direction. Entry points are marked with `TODO A2`. The following classes will be important.

Particle Animation You will make your salmon shoot pebbles by implementing a CPU particle system that instanciates spherical objects and simulates their path in water.

Bouncing Pebbles To make the pebbles collide with other pebbles and characters you will need to add functionality to `PhysicsSystem::step`.

3 Required Work

1. Getting Started
   (a) We recommend pushing your code to a **private** git repository. Commit all your edits from Assignment 1 to git, such that you can track and potentially revert changes.
   (b) Keep a separate copy of your Assignment 1 executable and dlls, to be able to showcase your A1 solutions to TAs on request.
   (c) Play the reference video `https://youtu.be/f3vpzaYaewA` to get a sense of what a possible assignment solution should look like.

2. Particle Animation (60%, prereq Rigid Body Physics lecture)

   Implement a particle system which generates pebbles that shoot from the salmon's mouth several times per second (adjust the timing for a compelling visual effect). The pebbles should have randomized initial directions (away from the salmon) and initial velocities. Their subsequent motion should be driven by a combination of these initial properties and gravity. Implement this animation in stages:

   (a) Generate periodic pebbles that shoot from the salmon's mouth and follow a fixed straight-line path at a fixed speed. The `WorldSystem::restart_game()` function provides example code for creating static pebbles on the floor. Remove the line that adds the HardShell component to each pebble in `createPebble()` as this would make them deadly to the salmon.

   (b) Randomize initial pebble direction, velocity, and size.

   (c) Introduce a new `Physics` component and add it at pebble creation time. It should indicate that an entity is affected by physics and may store related properties such as object mass.

   (d) Add gravity into the system to produce physically plausible (parabolic) pebble paths. Note that the game template computes in pixel units and milliseconds; not meters and seconds as common in physics;

3. Bouncing Pebbles (40%, prereq Rigid Body Physics lecture)

   (a) Add interaction in-between pebbles. Detect when two pebbles collide based on their radius. Make both bounce using physically plausible bounce direction and speed computations.

   (b) Handle the interaction between pebbles and other assets. Make pebbles bounce with the turtles but pass by fish and salmon (these are slim and avoid collisions). Make pebble and turtle bounce assuming both have spherical geometry with a suitable radius and mass. You will have to replace the 'SUPER APPROXIMATE' `collides()` function that is provided in the template.

   (c) One could implement the above with if conditions on HardShell and Physics components, but this is inflexible. Introduce a new component alongside `Physics` such that you can ensure that turtles bounce with pebbles but are unaffected by gravity. You should implement it such that there is one component to indicate that an object is affected by gravity and another that it bounces on collision.

4 Hand-in Instructions

1. Zip all code and CMake files as present in the template source code but exclude any executables and compiled files. Upload the zip file via MTA. Double check that you excluded all generated files, such as /build, .vs, /out! These would consume a lot of space on our server.

2. In addition, create a README.md file (Markdown language as used on github) that includes your name, student number, and any information you would like to pass on to the marker.

Recall, do not publish your solution on github or any other place. Neither during the course nor after; both is considered cheating.