CPSC 427: Video Game Programming
Intro to 2D Animation

Due: 23:59 PM, Friday November 8, 2019

1 Introduction

The goal of this assignment is to introduce you to basic 2D animation.

In the assignment you will extend the salmon game you made for assignment 2 by including in it a basic particle system implementation.

2 Template

You should use your own assignment 2 code as a starting point. You will find comments throughout the files to help you guide in the right direction. The directory is structured as follows:

- The directory `src` contains all the header (`.hpp`) and source (`.cpp`) files used by the project. The entry point is located in `main.cpp` while most of the logic will be implemented in `world.cpp` together with the respective salmon, fish, turtle, and pebbles `.cpp` files.
- The `data` directory contains all audio files, meshes, and textures used in the code.
- The `shaders` directory contains all shader files used in the code.
- The external dependencies are located in the `ext` subdirectory, which is referenced by the project files, it contains header files and precompiled libraries for:
  - `gl3w`: OpenGL function pointer loading (header-only)
  - `GLFW`: Cross-platform window and input
  - `SDL/SDL_mixer`: Playing music and sounds
  - `stb_image`: Image loading (header-only)

2.1 Particle Animation

You will make your salmon shoot pebbles by finishing the template for the particle system class `Pebbles`.

2.2 Bouncing Pebbles

To make the pebbles collide with other pebbles and characters you will need to complete the function `Pebbles::collides_with()`.

3 Required Work (90%)

1. Getting Started
   (a) Copy your assignment 2 codebase to a new directory and compile it as described in the instructions for assignment 1.
   (b) Play the `a3_solution_demo.mp4` video to get a sense of what a possible assignment solution should look like.
2. Particle Animation (70%)
   (a) Implement a particle system which generates pebbles that shoot from the salmon's mouth every few seconds (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:
      i. Generate periodic pebbles that shoot from the salmon's mouth and follow a fixed straight-line path at a fixed speed. Ensure that pebbles render correctly when (partially) overlapping other assets in your scene.
      ii. Randomize initial pebble directions and velocities.
      iii. Add gravity into the system to produce physically plausible (non-straight) pebble paths.
3. Bouncing Pebbles (20%)
   (a) Add interaction in-between pebbles. When two pebbles collide make both bounce using physically correct bounce direction, speed, and acceleration computations.
   (b) Add interaction between pebbles and other assets. When a pebble collides with a salmon, fish, or turtle, make it bounce using physically correct bounce direction, speed, and acceleration computations.

4 Creative Part(10%)

The required code changes described so far will let you earn up to 90% of the grade. To earn the remaining 10% as well as possible bonus marks you need to implement more advanced animation features in your game. **Marks for advanced features will be granted only if both they and all basic features are fully implemented and functional.** Possible additional features include:

1. Machine-oblivious time-stepping that produces consistent animation across platforms and computational loads
2. Asset collision response that incorporates moment of inertia and depends on exact collision location on the colliding objects
3. Pebble collisions triggering change in motion of collided characters.
4. Including force of water currents in pebble motion computations.

To get full credit you should add at least one of the features above and make it fully functional **and** free from bugs. The grading of additional bonuses, features, and the size of bonuses will be at the marker's discretion. **Multiple partially implemented features will not receive full credit.**
Use your imagination to make any other changes, however please make sure you focus on tasks involving animation.
To support both basic and advanced visualization and control features, you need to add a toggle option where the user switches between the two modes by pushing the 'a' and 'b' keys ('a' for advanced mode and 'b' for basic mode).
Document all the features you add in the README file you submit with the assignment. Advice: implement and test all the required tasks first before starting the free-form part.

5 Hand-in Instructions

1. You do not have to hand in any printed code. Create a README.txt file that includes your name, student number, and login ID, and any information you would like to pass on to the marker. Create a folder called "a3" under your "cs-427" directory. Within this directory have included all your source, data and make files as present in the template.
2. The assignment should be handed in with the exact command:

   ```
   handin cs-427 a3
   ```

   This will handin your entire a3 directory tree by making a copy of your a3 directory, and deleting all subdirectories. If you want to know more about this handin command, use: man handin. You can also use the web interface on your myCS page to upload the assignment.