CPSC 427: Video Game Programming
Intro to Game AI and Collision Processing Assignment

Due: 23:59 PM, Friday October 25, 2019

1 Introduction

The goal of this assignment is to introduce you to basic collision handling and game AI.

In the assignment you will extend the salmon game you made for assignment 1 by including additional features. The assignment includes both a required (80%) and a free-form component (20%). The goal of the latter is to let you experiment with computer graphics and have fun.

2 Template

You should use the same template as for assignment 1 along with your own assignment 1 code as a starting point. You will find comments throughout the files to help guide you 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 Collision

You will write additional functions in the `Salmon` class to test for and handle salmon collisions with walls, making the collisions exact and efficient as discussed below.

2.2 AI

To make your fish (and potentially turtles) smarter you will change their location update mechanism - `Fish::update(), (Turtle::update())`.

2.3 Debug

The shaders `coloured.vs.glsl` and `coloured.fs.glsl` will be useful for drawing debug information on the screen.

3 Required Work (75%)

1. Getting Started (15%):
   (a) Copy your assignment 1 codebase to a new directory and compile it as described in the instructions for assignment 1.
   (b) Play the `a2_solution_demo.mp4` video to get a sense of what a possible assignment solution should look like. Note the differences in the character behavior compared to A1.
   (c) Reduce the speed and frequency of fish and turtles to make the new features you are about to add easier to see.
   (d) Change the movement of the salmon to be consistent with its orientation, so that the left/right keys rotate the salmon and the up/down keys move the salmon along the direction it is aligned with.
2. Collisions (30%):
   (a) Make the salmon bounce off the walls if it bumps into them, flipping its direction of motion. Use **exact** collision computation (test for collisions between the salmon **mesh** and the walls). Use the salmon's bounding box(es) to make the computation reasonably efficient.
   (b) Include a debug mode toggle to visualize the impact of collisions. You will need to write your own debug `draw()` call to show the **exact** spots on the salmon that are colliding with the walls, freeze the view for a few milliseconds to make the effect more visible, and return the rendering to normal once the salmon has bounced far enough away.

3. Game AI (30%)

   (a) Make the fish smarter by enabling them to avoid the salmon. Each fish should follow the shortest path to the wall opposite its starting point, while also avoiding the salmon by staying at least some minimum distance ϵ away from it. The paths should be updated every X frames. The frequency X with which these paths are recomputed should be user-controllable.

   (b) Extend the debug mode from the previous part to visualize the computed paths of each fish on the screen and the spatial data structures used to implement this feature.

4 Creative Part(25%)

The required code changes described so far will let you earn up to 75% of the grade. To earn the remaining 25% as well as possible bonus marks you need to make the collision detection and AI more sophisticated. **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. Making the fish even smarter by accounting not just for the salmon's current position, but also for its anticipated motion trajectory. Similar to the previous part, add an advanced debug mode toggle that visualizes fish paths and spatial data structures used.

2. Avoid salmon/wall penetrations by accurately testing for **expected** collisions instead of existing penetrations. Like before, visualize the expected collision locations with an advanced debug mode toggle.

3. Replace the randomly spawned turtles with a single smart turtle that chases the salmon, while at the same time avoiding the fish. The turtles path should be updated every X frames. The frequency X with which the path is recomputed should be user-controllable. Visualize the spatial data structures used to enable this feature and the turtles path with an advanced debug mode toggle.

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 collision computation or AI.
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 "a2" 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 a2
   ```

   This will handin your entire a2 directory tree by making a copy of your a2 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.