Assignment 5

CS 381: Game Engine Architecture
Spring 2019
Max Score: 100

Assignment

You will build on top of the posted solution to assignment four and implement 1) left-mouse-click selection and 2) Flying entities. Use the posted solution to assignment four as the basis for this assignment. Your mission:

- Use buffered input to implement left-mouse-click selection. As before, an axis aligned bounding box should bound the selected entity.
- Add `desiredAltitude` and `altitude` variables and change the `altitude` towards the `desiredAltitude` at `climbRate` in your physics aspect to implement movement along the y-axis.

Architecture (10 points)

Understand and use the posted solution to assignment four and implement the two requirements by adding to this codebase.

Simple 3D Physics (20 points)

Just like `speed`, `desiredSpeed`, and `acceleration`, flying entities simply need to change `altitude` to `desiredAltitude` according to a `climbRate`.

Controlling desired altitude (10 points)

Players control a selected flying entity's desired altitude using the numeric PgUp and numeric PgDn keys on your keyboard.

New flying entity mesh (10 points)

I have uploaded a new flying entity's mesh and material to our model website. Only this *Banshee* entity can fly. You will need to take, place, and use these files to implement your flying entity. None of the other ship models or the alien vessel may leave the water surface. Make sure to load at least one example of each of the five (5) different types of ship models (entity types) at `http://www.cse.unr.edu/~sushil/models/381/` into your evolving game engine.

Camera control (5 points)

You will now control the camera with the WASD and **E (up) and F (down)** keys (get my permission if you would like to use different keys).

Quitting

Hitting the escape key should shut down your running game engine.

Tab selection (5 points)

Continue to leave the tab key bound to selection - that is - pressing the tab key will select the "next" entity. Only selected entities have visible axis aligned bounding boxes. So now, once you implement mouse selection, we will have two ways of selecting entities. Both can and should work together. Once you select an entity with a mouse left click, you should be able to select the "next" entity by pressing the tab button.

Mouse selection (40 points)

Use buffered input to implement mouse selection in the input manager. Left clicking the mouse over an entity in our game world should select the entity under the mouse (and unselect any other selected entity). To use buffered input, you will need to

1. Make your input manager inherit from `OIS::MouseListener` and `OIS::KeyListener`
2. Register the input manager to handle buffered mouse and keyboard events with, for example:

   ```
   mMouse->setEventCallback(this);
   mKeyboard->setEventCallback(this);
   ```

   during contruction or initialization of the input manager, as is done in the base application.
3. Make your input manager capture the mouse and keyboard every tick with, for example:

```
mKeyboard->capture();
mMouse->capture();
```

4. Override `bool mousePressed(const OIS::MouseEvent &arg, OIS::MouseButtonID id)` and put your ray-casting, distance thresholding, and finding the closest entity here. You will want to use the code shown in lab as a guide (40 points).

Cumulative Extra Credit

- Add shift-selection, where pressing the left-shift-key and tab key (or mouse), adds the next entity to the list of selected entities. User control applies to all selected entities (+5)
- Add a specific selection sound for each different type of entity. For example, when an entity of type, say, destroyer gets selected, it says `Ready to destroy` while when the frigate gets selected it says `Let's go`. Nothing obscene please (+10).
- Third person view forward from a selected entity's point of view (+5)
- Add group mouse selection (+5)
- Add the ability for selected entities to intercept another entity. Use right-mouse click to indicate the target ship to be intercepted (+5)
- Add wakes to all entities. This code should be added to Renderer (+5)

Turning in your assignment

Assume that this format will be used for all your laboratory assignments throughout the semester unless otherwise specified.

1. Demonstrate your working program in the lab on the due date. You will be graded on your demonstration.
2. In lab, submit the assignment using canvas
   (a) Make a subdirectory in your home directory named `as5`.
   (b) Place all your project files in as5 (you will demo from this folder in lab).
   (c) Tar and gzip or Zip the entire folder and submit using Canvas

Ask me or one of the TAs if you have questions.

Objectives

1. Demonstrate an ability to apply knowledge of computing, mathematics, science, and engineering by learning and applying knowledge of C++ and game engine architecture to solve the problem of implementing a game engine

2. Demonstrate an ability to analyze a problem, and identify, formulate and use the appropriate computing and engineering requirements for obtaining its solution by using inheritance, callbacks, and polymorphism

3. Demonstrate an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice by using the ogre engine framework and the eclipse IDE

4. Demonstrate an ability to apply design and development principles in the construction of software systems or computer systems of varying complexity such as our game engine