Assignment 4

CS 381: Game Engine Architecture
Spring 2017
Max Score: 100

Assignment

You will now design and implement mouse selection and some simple AI and add this to your evolving game engine. Please use my solution to assignment 3 as your starting point.

The left mouse button will be used for selecting the entity under the mouse. The right mouse button will be used to provide this selected entity with simple commands that will require rudimentary intelligence to execute.

You may work in groups of two.

Architecture (80 points)

Mouse Selection (25 points)

Using what you have learned in lab and class, implement simple mouse selection in your Input Manager using buffered mouse input. Indicate the selected entity by an Axis Aligned Bounding Box (AABB). Tab selection may remain as an alternative selection method.

Simple AI (55 points)

You will design and implement an AI aspect named UnitAI (UnitAI.h, UnitAI.cpp) which manages a sequence of commands that are executed in order. Commands are of two types: MoveTo and Follow.

MoveTo (25 points)

MoveTo moves the currently selected entity to the location right-button-clicked by your mouse.
Follow (30 points)

Follow, makes the currently selected entity follow the right-button-clicked entity (the target entity) under your mouse at a relative offset of \((-100, 0, 0)\). That is the selected entity should strive to reach a position 100 units behind the target entity.

Essentially, if you right click your mouse button over an entity, you will choose Follow. If there is no entity under your mouse when you right click the button, you will choose MoveTo.

These two behaviors are the basis for much more intelligent entity behavior. For example, executing a sequence of MoveTos enables an entity to follow a path, patrol, or attack another entity from a different angle. Similarly, following another entity at different relative offsets enables formations.

Finally, when you press the the Left-Shift key while pressing the right mouse button, the command (one of the two above) is added to the list of commands maintained by UnitAI. When a command is done, the UnitAI aspect executes the next command in the list or does nothing if there are no commands to execute.

Like all aspects, UnitAI will inherit from the Aspect class and define a Tick method. In addition, UnitAI will define and implement

1. `void SetCommand(Command *c)` to clear all current commands and set the new command to `c`.
2. `void AddCommand(Command *c)` to add `c` to the end of the current list of commands.

The player presses the LEFT SHIFT key and the right mouse button to add a command to the list.

Note that the UnitAI class keeps a `std::list<Command*> commands` of commands. AddCommand adds to the end of this list (queue) and SetCommand clears the list and places the new command as the only member of this list.

A command inherits from the Command class and can be one of MoveTo or Follow. Here is a sample `Command.h` that also provides a declaration of the MoveTo subclass. Note that commands have a `bool done()` method that returns true when a command is done (reached the MoveTo location). UnitAI uses `done` to move on to the next command in the command list.

Camera control (20 points)

Improve camera control in the game engine by providing pitch (Z, X) and yaw (Q, E) keyboard controls. Use R and F for motion along the y-axis and ensure that the camera cannot go under your water plane. In addition, implement two speeds for movement and rotation. The faster speed is used when you press the left shift key while pressing one of the other camera control keys (AWSDQEZXRFS).

Quitting

Hitting the escape key should shut down your running game engine.
Design contraints
Build on top of my solution to assignment 3.

Cumulative Extra Credit

- Add group mouse selection (+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 jetski 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 the ability for selected entities to intercept another entity. Use right-mouse click to indicate the target ship to be intercepted (+5). Intercept does not mean follow.
- Add wakes to all entities. This code should be added to Renderer (+10)

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.
2. In lab, submit the assignment using canvas
   (a) Make a subdirectory in your home directory named as4.
   (b) Place all your project files in as4 (you will demo from this folder in lab).
   (c) Tar and gzip or Zip the entire folder and submit using Canvas

Ask me (sushil@cse.unr.edu) or chad (chad.adams7@gmail.com) 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.