Assignment 6

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

Assignment

Design and implement simple AI commands for entities in your game engine. You will design and implement two commmands: move and predictive-intercept, for entities in your game engine. To do this you need to design and implement a command class, a unit AI aspect that manages sequences of commands, and the move and predictive-intercept commands. Use my solution to assignment five as your starting point.

The left mouse button selects the entity under the mouse. The right mouse button commands selected entities. Pressing the shift key while right mouse clicking *adds* a command to the list of commands for the selected entity.

You may work in groups of two.

AI

Simple AI (80 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 **Intercept**.

MoveTo (30 points)

MoveTo moves the currently selected entity to the location right-button-clicked by your mouse.

Intercept (30 points)

Intercept (short for predictive intercept), makes the currently selected entity predictively intercept the right-button-clicked entity (the target entity) under your mouse.

Essentially, if you right click your mouse button over an entity, you will choose Intercept. 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.

Command sequencing (20 points)

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 (first in first out) 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 Intercept. Here is a sample `Command.h` that also provides a declaration for 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.

More 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 (AWSDQEZXRF).

Quitting

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

Design contraints

Build on top of my solution to assignment 5.

Cumulative Extra Credit

- Add group mouse selection and commands apply to all selected entities (+5)
- Add a specific selection sound for each different type of entity (+5)
- Third person view forward from a selected entity's point of view (+5)
- Add the ability for selected entities to follow another entity. Use right-mouse click to indicate the target ship to follow (+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 `as6`.
   (b) Place all your project files in as6 (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 any 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