Assignment 3

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

Assignment

You will work on creating and using your own game loop and escape from dependence on ogre's frame listener and the sample framework. We have talked extensively about the architecture of the game engine in class and your design and implementation will fit within the game engine architecture that we have been discussing in class. All pre-written code for this assignment is here. **You may work in groups of two**. Your mission is to re-implement assignment two in this new architecture as specified below.

Architecture (70 points)

The game engine will now be implemented in a class called `Engine`. Your main program (in as3.cpp) will create, initialize, and run our 381 game engine as shown below.

```
# create, initialize, and run the 381 engine
  Engine *engine;
  engine = new Engine();
  engine->init();
  engine->run();
```

The `Engine` class is defined here: engine.cpp. The Engine class

1. Constructs manager instances
2. Initializes manager instances
3. Implements the game tick. It computes the time for one tick, and passes this on to all your Manager instances.
4. Brings the engine down gracefully

It should be clear from the Engine implementation that the engine contains the following Managers.

1. `EntityMgr` to create and keep track of all game entities. This will change little from my posted solution to assignment two (10 points)

2. `GfxMgr` to initialize ogre graphics (as in Tutorial 6), and initialize the camera. This class' tick function calls ogre's `renderOneFrame` as shown in Tutorial 6 (25 points)

3. `InputMgr` to initialize the keyboard, mouse, and setup buffered input. Handle tab selection, handle camera movemement, and handle user input to change entity desired speed and desired heading. (25 points)

4. `GameMgr` to initialize the game level (scene) and create game entities (20 points)

You have to design and implement `EntityMgr, GfxMgr, InputMgr, GameMgr`. All managers inherit from the Mgr class and call their superclass (`Mgr`'s) constructor to store a pointer to the game engine. You may use and modify `EntityMgr` from my solution to assignment two. Most of the functionality and code for `GfxMgr` and `InputMgr` can be found in Tutorial 6 on the ogre website and in the solutions to prior assignments on our class website. Note that, apart from the class constructor, each manager implements the following methods: `init, tick, loadLevel, stop`.

You will also need to design and implement two aspects to be attached to each entity.

1. `Physics`: Implements the same physics from our prior assignment

2. `Renderer`: Creates and manages a 381 entity's ogre scene node and takes care of copying the 381 entity's position and orientation to its scene node.

You may copy these from the posted solution to assignment two. No changes will be needed to `Entity381`, `Aspect`, or the the above two aspects.

This assignment is very similar to the prior assignment and you are to choose and load at least one example of each of five (5) different types of ship models (entity types) at `http://www.cse.unr.edu/~sushil/models/381/` into your evolving game engine.

Oriented Vector Physics (5 points)

Implement the same simple physics as for assignment two. You may use my solution.

Camera control (5 points)

You will continue to control the camera with the WASD and PgUp and PgDown keys (get my permission if you would like to use different keys). You will also use, the Q and E keys to control camera yaw and the Z and X keys to control camera pitch.

Quitting

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

Tab selection (10 points)

Bind the tab key to selection - that is - pressing the tab key will select the "next" entity. Only selected entities have visible axis aligned bounding boxes. **Use buffered input callbacks to implement tab selection**.

Design contraints

You cannot change `as3.cpp, Aspect.cpp, Entity381.cpp, mgr.cpp, engine.cpp` or their associated header files.

Cumulative Extra Credit

- Add mouse selection (+5)
- Add shift-selection, where pressing the left-shift-key and tab key, 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 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 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.
2. In lab, submit the assignment using canvas
   (a) Make a subdirectory in your home directory named `as3`.
   (b) Place all your project files in as3 (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) 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