Assignment

You will work on creating and using your own game loop and escape from dependence on python ogre and the sample framework. You will also add mouse selection. 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.**

0.1 Architecture (70 points)

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

```python
# create, initialize, and run the 381 engine

import engine

game = engine.Engine()
game.init()
game.run()
```

The Engine class is defined here: [engine.py]. 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

2. **GfxMgr** to initialize python ogre graphics (as in Tutorial 6), initialize the camera, and initialize the scene (water plane). This class’ tick function calls python ogre’s `renderOneFrame` as shown in Tutorial 6.

3. **InputMgr** to initialize the keyboard and mouse and setup buffered input. Handle mouse selection and, for now, handle camera movement.

4. **SelectionMgr** to maintain and manage the list of selected entities

5. **ControlMgr** to handle entity control through the arrow keys on the keyboard. The arrow keys on your keyboard control all selected entities. Up/Down will increase and decrease `desiredSpeed` in the current direction of motion. Left/Right arrow keys turn the entity, they change the entity’s `desiredHeading`.

6. **NetMgr** to initialize networking (currently an empty implementation)

7. **GameMgr** to initialize and run your game

You have to design and implement **EntityMgr, GfxMgr, InputMgr, SelectionMgr, ControlMgr**. You may use and modify **EntityMgr** from my solution to assignment four. Most of the functionality and code for **GfxMgr** and **InputMgr** can be found in Tutorial 6 on the python 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, stop`.

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

1. **Physics**: Implements the same physics from the prior assignment

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

Otherwise, the assignment is very similar to the prior assignment. Select and load at least one example of each of the ten (10) different types of ship models (entity types) at http://www.cse.unr.edu/~sushil/models/381/ into your evolving game engine. 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.
0.2 Mouse Selection (20 points)

In addition, you will design and implement mouse selection. Specifically, left clicking the mouse selects the entity under the mouse cursor. Holding the shift key while left clicking the mouse should add the entity under the mouse to the current set of selected entities. Shift-tab should also add the next entity to the current set of selected entities. All selected entities are affected by arrow keys. Note carefully that to see X11’s mouse cursor in python ogre’s graphics window, you will need to initialize the input system a little differently than shown in Tutorial 6. This is the current code:

```python
renderWindow = self.root.getAutoCreatedWindow()
windowHandle = renderWindow.getCustomAttributeInt('WINDOW')
paramList = [('WINDOW', str(windowHandle))]
sel.inputManager = OIS.createPythonInputSystem(paramList)
```

We need to add another parameter and value to `paramList` before calling `OIS.createPythonInputSystem(paramList).

```python
renderWindow = self.root.getAutoCreatedWindow()
windowHandle = renderWindow.getCustomAttributeInt('WINDOW')
paramList = [('WINDOW', str(windowHandle))]
t = [('x11_mouse_grab', 'false'), ('x11_mouse_hide', 'false')]
paramList.extend(t)
sel.inputManager = OIS.createPythonInputSystem(paramList)
```

See the two extra lines? They make the default X11 cursor visible in python ogre’s window. They also allow the mouse cursor to be usable outside python ogre’s window.

0.3 Oriented Vector Physics (5 points)

Implement the same simple physics as for assignment four.

0.4 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.

0.5 Quitting

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

0.6 Design constraints

You cannot change `main.py`, `engine.py`, `netMgr.py`, `gameMgr.py.`
0.7 Cumulative Extra Credit

- 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 sushil/bin/sub381As5

   (a) Make sure your code has your name on it

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 Python to solve a problem

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

3. Demonstrate an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice

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