Assignment 2

CS 381: The Game Development Pipeline
Spring 2019
Max Score: 100

Assignment

You will work on handling multiple entities of two different types and implement tab selection. For now, our physics remains the same. Please use the posted solution to assignment one as the basis for this assignment.

0.1 Multiple entities and selection (30)

First, make a large (10000×10000) planar textured surface located at the origin on the xz plane at a height specified by a `surfaceHeight` variable (10 points). Experiment with the different ways of texturing your ground plane with water. You may use the previous assignment's posted solution for pointers.

Create and load five (5) cubes and five (5) spheres in two rows of five into your evolving game engine. Bind the tab key to selection - that is - pressing the tab key will round-robin select the "next" entity. A selected entity, and only the selected entity, has a visible axis aligned bounding box (20 points for selection).

We need to distinguish between Ogre's Entity class and our Entity as used in the paragraph above. To be clear I will use **Entity381** when I do NOT mean to refer to Ogre's entity class. So, to make this clear, I reproduce the above paragraph with Entity381 replacing Entity.

> Create and load five (5) cubes and five (5) spheres in two rows of five 381Entities into your evolving game engine. Bind the tab key to selection - that is - pressing the tab key will round-robin select the "next" Entity381. A selected Entity381, and only the selected Entity381, has a visible axis aligned bounding box (20 points for selection).

0.2 Simple Physics (15)

The arrow keys (or the numpad keys) on your keyboard control the **selected** Entity381's velocity. Make the selected Entity381 move in 3D with a **velocity** determined by the numeric

pad's arrow keys and PGUP/PGDOWN. Key presses increase or decrease the x, y, and z components of the Entity381's **velocity**.

0.3 Camera control (5)

You will continue to control the camera with the WASD and E and F keys. In addition, you will now also use, the LeftShift+A and LeftShift+D keys to control camera yaw.

0.4 Quitting (0)

Hitting the `q` key should gracefully shut down your running game engine.

0.5 Engine Architecture and Design (50)

Here are architecture and design elements that you MUST follow. I assume that your `_createScene` function is in `as2.cpp`

- (10 points) Create a Entity381 class to hold each Entity381's position, velocity, `Ogre::SceneNode` pointer, mesh file name, and other information. More information about 381Entities follows, for now, note that 381Entities will be managed by an Entity Manager.
- (10 points) You will create one instance of an `EntityMgr` class in `_createScene`. The `EntityMgr` manages entities. This means the `EntityMgr`
  1. maintains a list (or map or array) of all entities in your game engine
  2. tracks which entity is currently selected
  3. creates entities and assigns them a unique identifier. This means you will need a `Entity381* CreateEntityOfTypeAtPosition(EntityType type, Ogre::Vector3 pos)` method. I specify `Entity381` below.
  4. has a `void Tick(float dt)` method. This method iterates through the list of Entity381s and calls each Entity381's `Tick` method.
- (10 points) Create and use an `Entity381` class to hold information about your entities. Entity381 members must include
  1. entityId, entityName
  2. position, velocity
  3. ogreSceneNode, ogreEntity (pointers to the entity's Ogre::SceneNode and Ogre::Entity)
  4. a list of `Aspects`. We will specify Aspects below.
  5. Apart from the constructor, Entity381 will also declare and define a `void Tick(float dt)` method. This method will iterate through the list of this Entity381's Aspects and call each Aspect's Tick method. For this assignment you will create the two aspects listed and described below.

6. other members as needed

- (5 points) Each of the two Entity381 types (Sphere and Cube) should be implemented as a separate subclass of `Entity381`.
- (15 points) Create and use an `Aspect` class and three subclasses of Aspect. The Aspect class is simply a base class to hold
  - a pointer to this aspect's Entity381
  - a virtual `void Tick(float dt)` method that will be overridden by subclasses.

  Each entity will have three aspects one of each subclass type below.

  1. Physics (5 points) - This aspect will will simply update the Entity381's position inside `Tick(pos = pos + vel * dt)`.
  2. Renderable (5 points) - Renderable manages the scene node (or nodes) and `Ogre::Entity` associated with our `Entity381s` and on every tick, copies the position from our `Entity381` to the `Entity381`'s scene node. The equivalent of `cubeSceneNode->setPosition(position);` from my solution to assignment one.
  3. Rotator (5 poitns) - Rotator slowly rotates each entity. You will be changing the 381Entity's SceneNode's yaw by a small amount every tick
- Please use Ogre's vector classes for vector math.
- In your `frameRenderingQueued` method you will call, your `EntityMgr`'s `Tick` method. This will go through the list of Entity381s and call each entity381's Tick method. Each Entity381's Tick method will then iterate through that Entity381's two Aspects and call each Aspect's Tick method.

These constraints will result in a cleaner, better, design for your emerging game engine. They will also increase your understanding of inheritance and game engine architecture.

Extra Credit

This is cumulative!

- Add mouse selection (+5)
- Add islands in your waterworld (+3)
- Find and use new 3D models. Not the ones found on the ogre wiki: `http://wiki.ogre3d.org/Ogre+Models`. Hand in a document that provides a step by step tutorial on how to incorporate each of your models in the game engine.
- Add group mouse selection (+10)
- Add the ability to use standard RTS game mouse commands in order to have a selected (left-clicked) ship or boat to intercept another (right clicked) ship (+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 your code using Canvas.
   (a) Make a directory (folder) named `as2`.
   (b) Make a movie demoing your assignment and place this in `as2`
   (c) Place all your project files in as2 (you will demo from this folder in lab).
   (d) Tar and gzip or Zip the entire folder and submit using Canvas. **If you do not submit this file, you will lose** 20% **points.**

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 (1)
2. Demonstrate an ability to analyze a problem, and identify, formulate and use the appropriate computing and engineering requirements for obtaining its solution (5)
3. Demonstrate an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (11)
4. Demonstrate an ability to apply design and development principles in the construction of software systems or computer systems of varying complexity (13)