Game Engines

381

Game Engines

Game Engines are composed from

- Entities – A collection of game entities
- GFX – a graphics engine to display entities
- PHX – a physics engine to move entities
- UI – to select, command, and describe entities
- AI – an AI engine to carry out commands
- Net – to connect to other game engines

- We will study how to integrate
    - Entities, GFX, PHX, UI, AI and perhaps Net in this class

PYTHON OGRE

- We are using python-ogre because
  - Python-ogre is cross-platform and open source
  - Python is much quicker for prototyping than C/C++
  - Python-ogre is under active development
  - With a little effort, you can create a windows, mac, and linux executables for distribution

Framework

```
import ogre.renderer.OGRE as ogre
import SampleFramework as sf

class TutorialApplication(sf.Application):

   def _createScene(self):
      pass

if __name__ == '__main__':
   ta = TutorialApplication()
   ta.go()
```

GFX basics: Scene Managers

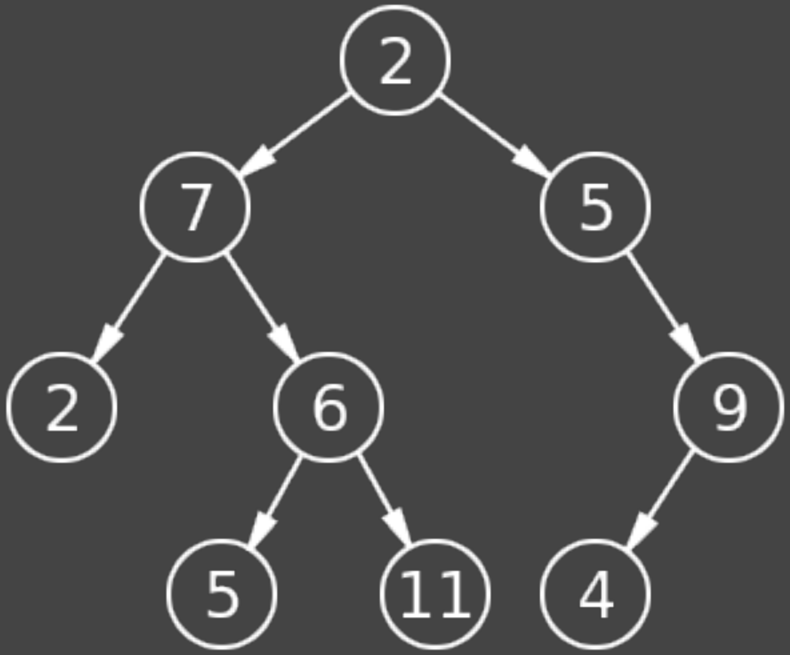
- Scene managers manage your display and are optimized for types of scenes. Terrain scene managers may not be best suited for first person shooters but may be well suited for RTS games
- Scene is an abstract representation of what is displayed on the screen
  - Static geometry like terrain or building
  - Models – trees, boats, monsters, …
  - Light sources – let there be light
  - Cameras – so users can have a point of view

Movable Objects

- Renderable (displayable) and Movable
- Python Ogre Entity is a subclass of MovableObject
  - 3D Mesh
  - No location or orientation
- SceneNode is a class that contains location and orientation information
- To display an entity object, attach object to a scene node object
- Lights, Particles, Cameras are not Entities but are MovableObjects

Renderables and Scene Nodes

- Attach an entity (3D mesh) to a scene node (position, orientation) to render (make it appear) it on screen (in the scene)
- The scene manager gives you root scene node. This root scene node forms the root of a tree of scene nodes



Scene Nodes

- Position and orientation of a scene node N are relative to that of N's parent.
- In general
  - Attach independently moving entities to scene nodes that are children of the root scene node
  - Example: Attach Ferrari's mesh to node1, child of root scene node. Attach Ferrari's wheel's meshes to node1
- You will usually create a shallow hierarchy of scene nodes

Code for _createScene

sceneManager = self.sceneManager sceneManager.ambientLight = ogre. ColourValue (1, 1, 1)
sceneManager is created for us by the SampleFramework. We will continue to use the Sa
for a couple of assignments.

ent1 = sceneManager.createEntity ("Robot", "robot.mesh")
First parameter must be a UNIQUE name
Second parameter must be the name of a file that contains the 3D robot's mesh in Ogre's 3D mesh format. This file has been found and loaded by the SampleFramework

node1 = sceneManager.getRootSceneNode().createChildSceneNode ("RobotNode")
Create a scene node that is a child of the root scene node. Scene node name must also be UNIQUE

node1.attachObject (ent1)
Now you can run your code

Coordinate axes!

- X – Z horizontal plane
- Y vertical axis
- -X left, +X right
- -Y down, +Y up
- -Z into screen, +Z out of screen towards you
- Meshes that you load can face any direction. It is up to you and your modeling program (blender) to orient your meshes

Vectors - ogre.Vector3(x, y, z)

Vector3
ogre.Vector3(0, 50, 0) or
(0, 50, 0)

ogre.ColorValue(1, 1, 1) or
(1, 1, 1)

```
node2 = sceneManager.getRootSceneNode().
createChildSceneNode ("RobotNode2", ogre.Vector3 (50, 0, 0))
node2 = sceneManager.getRootSceneNode().
createChildSceneNode ("RobotNode2", (50, 0, 0))
```

Framework

```
import ogre.renderer.OGRE as ogre
import SampleFramework as sf

class TutorialApplication(sf.Application):

   def _createScene(self):
       pass

if __name__ == '__main__':
   ta = TutorialApplication()
   ta.go()
```