Game Engine Architecture

With python ogre

Architectural constraints

- Driven by updates
- Need multi-threading for networking
- 15+ frames a second
- Good design
    - Clear separation of components
    - Clean
    - Pluggable components

Components

- Entities
- Graphics
- Physics
- Networking
- Input
- UI
- AI (later)

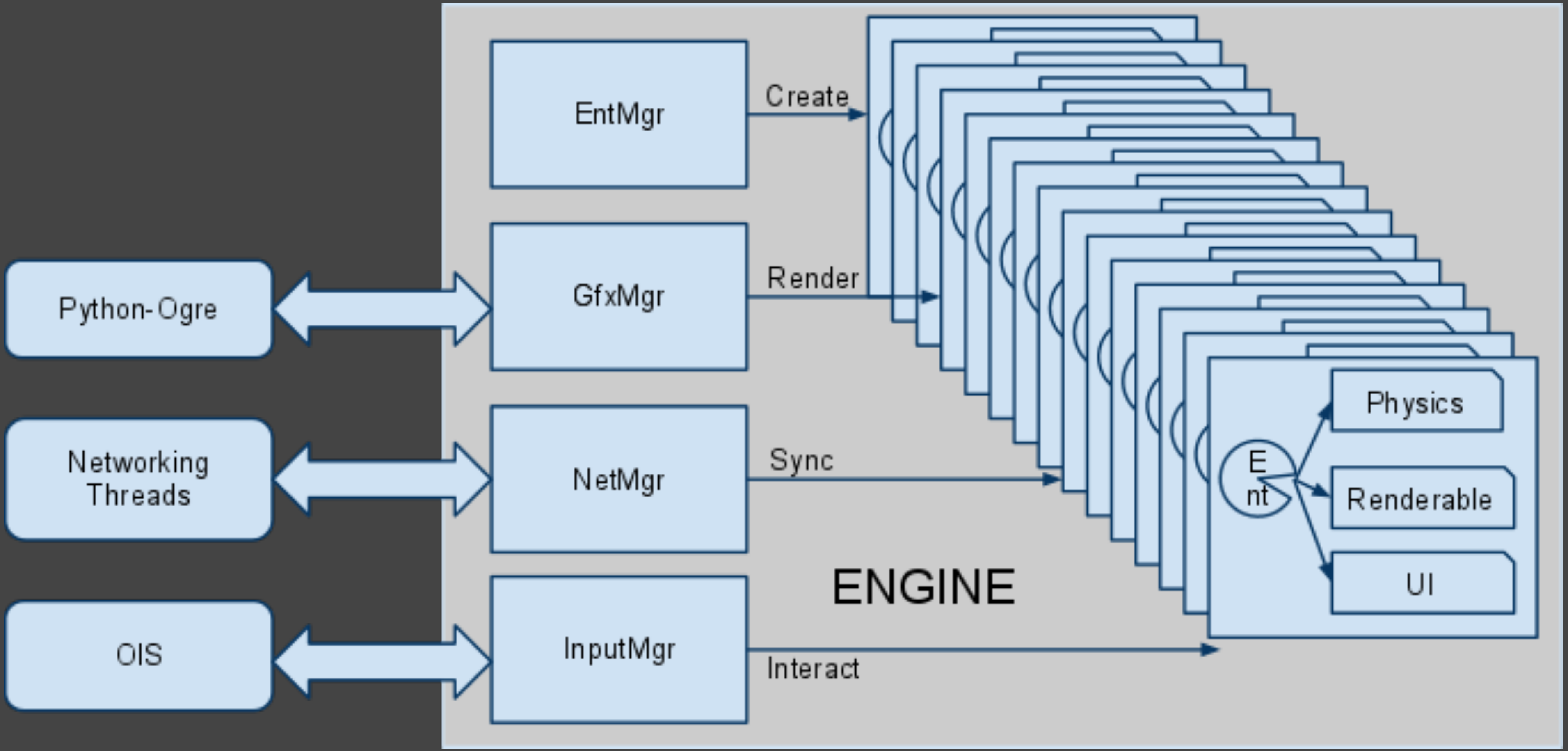
Constraints (things to keep in mind)

1. Each movable entity has a mesh to render, some state to keep track of, and moves according to some physics
2. **Selected** entities respond to input
3. There are **many** entities

Game Engine Components

- EntMgr to manage entities
- NetMgr to manage networking
- GfxMgr to manage rendering
- InputMgr to manage keyboard, mouse
- UIMgr to manage on screen widgets (buttons, text, menus...)
- Each entity has:
  - physics aspect to specify position, orientation
  - graphics aspect to render
  - control aspect to respond to user
  - AI aspect to respond smartly to user and to other entities

Game Engine



Not this clean. Each Mgr may need to talk to every other Mgr and to Ents

Game Engine - Sequence

1. Create components - construct
2. Initialize
3. Cross link - so Mgr's have handles for each other
4. Load Level - make scene, ents, ...
5. Run - tick(dtime)
6. Release Level - free resources
7. Exit

Main

```
import engine

engine = engine.Engine()
engine.initialize()
engine.crosslink()
engine.loadLevel()
engine.run()
engine.releaseLevel()
```

Engine

```
class Engine(object):
    '''
    The root of the global manager tree
    '''
    def __init__(self):
        self.entMgr = None
        self.gfxMgr = None
        self.netMgr = None
        import time
        self.oldTime = time.time() # Use time.clock() for windows
        self.runTime = 0;
        self.keepRunning = True
        pass
```

engine.initialize

```
def initialize(self):
  import entMgr
  self.entMgr = entMgr.EntMgr(self)
  self.entMgr.initialize()

  import gfxMgr
  self.gfxMgr = gfxMgr.GfxMgr(self)
  self.gfxMgr.initialize()

  import netMgr
  self.netMgr = netMgr.NetMgr(self)
  self.netMgr.initialize()

  import inputMgr
  self.inputMgr = inputMgr.InputMgr(self)
  self.inputMgr.initialize()

  import selectionMgr
  self.selectionMgr = selectionMgr.SelectionMgr(self)
  self.selectionMgr.initialize()
```

engine.crosslink()

```
def crosslink(self):
    self.entMgr.crosslink()
    self.gfxMgr.crosslink()
    self.netMgr.crosslink()
    self.inputMgr.crosslink()
    self.selectionMgr.crosslink()
```