CIT2213 Game Engine Architecture

Lecture: Datastructure - Scene Graphs

Dr Minsi Chen

Computer Science

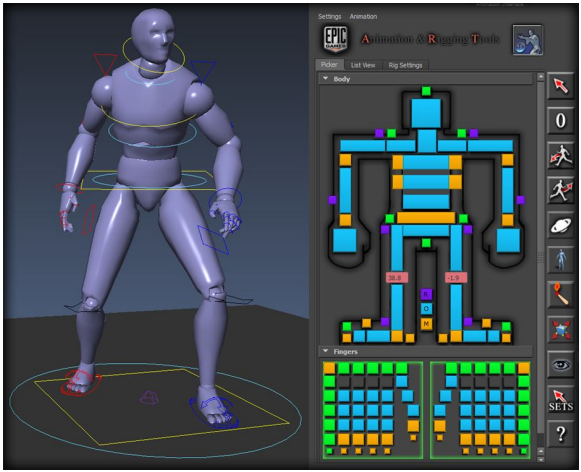
Overview

- Hierarchical data
- Graphs - represent assets and their relation
- Implementation - traversal

Observe This Object

source: halo.wikia.com

Another Example - Characters



Source: Unreal Engine 4.0, EPIC Games

A Naïve Routine

```
//Assume all parts are stored in a linear array
...
for each part P in the array
   Set world transformation of P
   Update pose of P

   for each part Q in the array
        Detect collision P and Q
        Resolve collision
   end
end
...
```

Hierarchical Data

Object

- An object may consist of multiple parts
- Each part can move independently as well as being dependent on another part

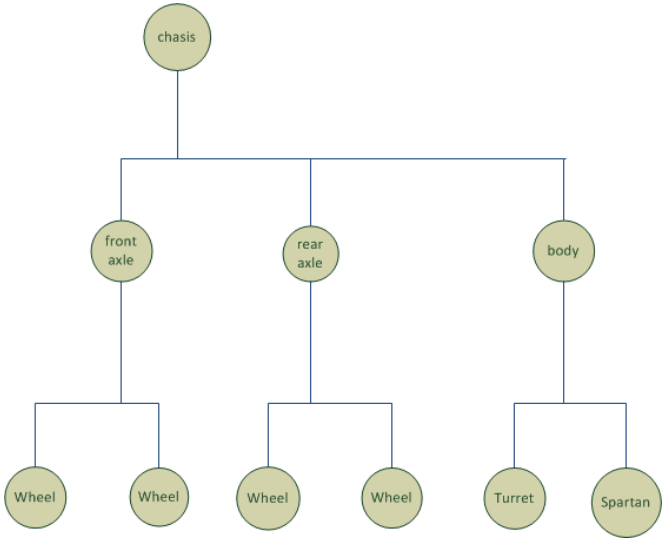
Scene

- The transformation of an object may be dependent on another object
- More intuitive to represent transformation relative to a "parent" object
- Multiple objects may be grouped to form a large cluster, e.g. a cluster of asteroids

Rendering

- Managing render states, e.g. textures, shaders, blending
- Minimising the frequency of switching render state by, e.g. grouping objects sharing same materials

A Scene Graph



- Graph - a set of edges E and vertices V (nodes)
- Directed acyclic graph (DAG)
- Hierarchical relation amongst given parts
- Most commonly a tree (DAG)

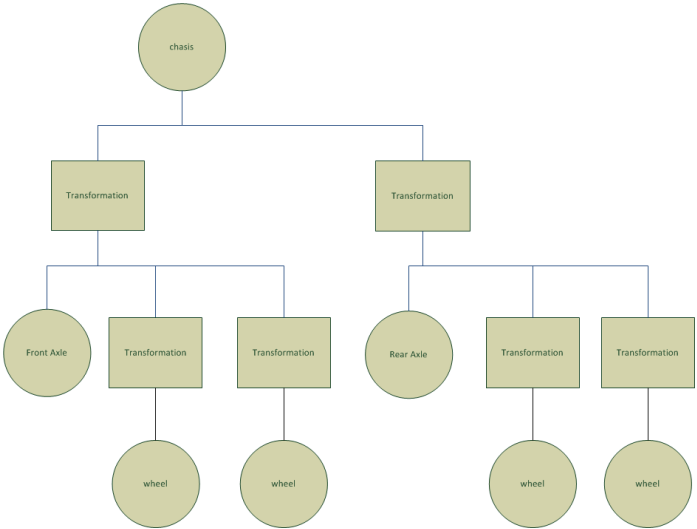
What Can A Scene Node Represent?

A scene node can take on a role relevant to the purpose of an application. For a rendering engine, we typically have a node that represents:

- A renderable part, e.g. meshes, collision primitives
- Transformation, spatial relationship, e.g. offset, relative transformation
- Resource group, e.g. material group, algorithm group
- User inputs/interface

A parent node always applies to its descendent nodes.

A Slightly Enhanced Scene Graph



A Simple Base SceneNode Class

```
class SceneNode {
   ...
   protected:
      SceneNode* m_pParent;
      std::vector<SceneNode*> m_children;
      //include more properties

   public:
   ...
      SceneNode* getParent();
      void addChild(SceneNode* pChild);
      SceneNode* getChild(int nth);

      //This is known as a pure virtual function (PVF)
      //Class containing one or more PVF is an abstract class
      //Cannot be instantiated
      virtual void render() = 0;
   ...
};
```

A scene node should have:

- a reference to its parent; node without a parent node is the root
- a list of children; node with no child is a leaf node

Scene Graph Traversal - I

Depth first traversal

```
void traverse_scenenode( SceneNode* node )
{
   int i = 0;

   if ( !node )
     return;

   int nChildren = node->getNumChildren();


   for ( i = 0; i < nChildren; i++ )
      traverse_scenenode( node->getChild(i) );
}
```

Note: Why is `traverse_scenenode` not a member of the class?

Scene Graph Traversal - II

Render traversal

```
void render_scenenode( SceneNode* node, Matrix* m )
{
    int i = 0;

    if ( !node )
       return;

   int nChildren = node->getNumChildren();

      for ( i = 0; i < nChildren; i++ )
   {
      Matrix world = node->getTransformation();
      world = (*m)*world;

      //Do some rendering
      ...
      render_scenenode( node->getChild(i), &world );
   }
}
```

Extending from the Base SceneNode

```
//transformation node
class XfmSceneNode : public SceneNode
{
    protected:
       ...
       Matrix4 m_transformation;
       ...
    public:
       void render(); //must implement this
};

//mesh/geometry node
class GeomNode : public SceneNode
{
    protected:
       ...
       Mesh* m_pMesh;
       ...
    public:
       void render(); //must implement this
};
```

Modifying the Graph

- Change graph structure at runtime
  - add/remove nodes
  - rearrange parenthood
- Modify node parameters
  - update transformations
  - geometry data
  - materials, e.g. textures, colours
- Adding new node types
  - particle system
  - lights
  - camera

Ogre3D Case

```
...
Ogre::Light* light = scnMgr->createLight("MainLight");
Ogre::SceneNode* lightNode = scnMgr->getRootSceneNode()->createChildSceneNode();
lightNode->setPosition(0, 10, 15);
lightNode->attachObject(light);
...
Ogre::SceneNode* camNode = scnMgr->getRootSceneNode()->createChildSceneNode();
camNode->setPosition(0, 0, 15);
camNode->lookAt(Ogre::Vector3(0, 0, -1), Ogre::Node::TS_PARENT);
...
Ogre::Camera* cam = scnMgr->createCamera("myCam");
cam->setNearClipDistance(5); // specific to this sample
cam->setAutoAspectRatio(true);
camNode->attachObject(cam);
...
Ogre::Entity* ent = scnMgr->createEntity("Sinbad.mesh");
Ogre::SceneNode* node = scnMgr->getRootSceneNode()->createChildSceneNode();
node->attachObject(ent);
```

Benefits of a Scene Graph

- Represent hierarchy
- Group objects by properties
- Abstraction from low level API's and Libraries e.g. D3D, OpenGL
- Consistent traversal
- Can incorporate spatial partitioning data structure

Scene Graph Based API's and Libraries

- Unity and Unreal Engine both use graph based structure
- Ogre3D (`www.ogre3d.org`) - its scene organisation is based on scene graph
- OpenSG (`www.opensg.org`) - an open source general purpose scene graph

Scene Graph related Research

- Rigid-body simulation
  - C. Dawei and T. Hongmin, "A rigid-body simulation application framework for scene graph system and its implementation," 2010 International Conference on Audio, Language and Image Processing, Shanghai, 2010, pp. 654-658. doi: 10.1109/ICALIP.2010.5685050
- Generating game scene from texts (a form of procedurally generated contents)
  - Bob Coyne and Richard Sproat. 2001. WordsEye: an automatic text-to-scene conversion system. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH '01). ACM, New York, NY, USA, 487-496. DOI: `https://doi.org/10.1145/383259.383316`
  - NLP Group at Stanford University: `https://nlp.stanford.edu/projects/text2scene.shtml`
- Scene recognition (Computer Vision and Image Understanding)
  - Johnson, J., Krishna, R., Stark, M., Li, L., Shamma, D.A., Bernstein, M.S., and Fei-Fei, L. (2015). Image retrieval using scene graphs. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3668-3678.