Meshes and Resources

Wayne Rippin (UoD)
Dr Panagiotis Perakis (MC)

Last Week

- Last week, we looked at scene graphs.
- A scene graph is used to hold the structure of the scene, i.e. it represents the overall hierarchy of a scene
- At the moment, each node in the scene graph is responsible for managing its own vertices, indices and textures and for rendering them
- The problem is that this results in a lot of copies of the same vertices, index lists, textures, etc and the same operation (such as compiling shaders) being done multiple times
- In addition, we have been creating objects (e.g. the cube) manually. This is impractical for more complex models. We want to use models that have been created by an artist.

Meshes

- Model files contain meshes
- A mesh represents a complete model –a file may contain multiple meshes if the file included an animated model.
- A mesh consists of
  - A tree structure that represents the structure of the model
  - A series of sub-meshes

Meshes

- A sub-mesh consists of:
  - Vertices
  - Indices
  - Normals
  - UV coordinates
  - Materials
    - Texture names
    - Diffuse colours
    - Specular colours, etc

Meshes

- There are many different model formats
- They can be in either binary or text format

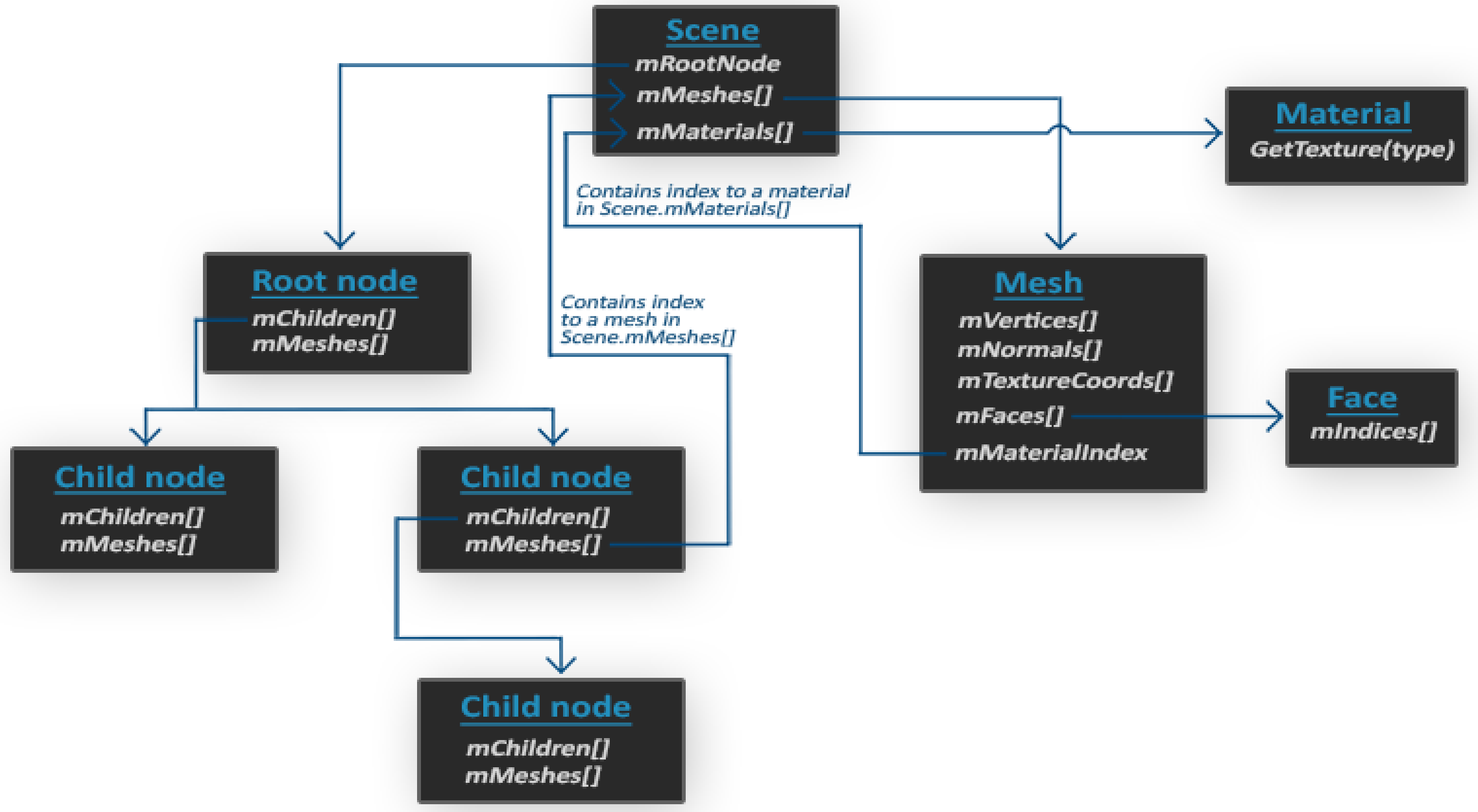
ASSIMP

- Writing your own model loading code can be a pain. You have to parse file formats and build up the appropriate structures
- For many file formats, ASSIMP (Open Asset Import Library) can make things easier:

  www.assimp.org

- ASSIMP supports a number of file formats
  - It loads the file and populates a set of standard structures so that you can handle all of the different file types in a consistent format.

ASSIMP Structure I



https://learnopengl.com/Model-Loading/Assimp

ASSIMP Structure II

A (simplistic) model of Assimp's structure is:

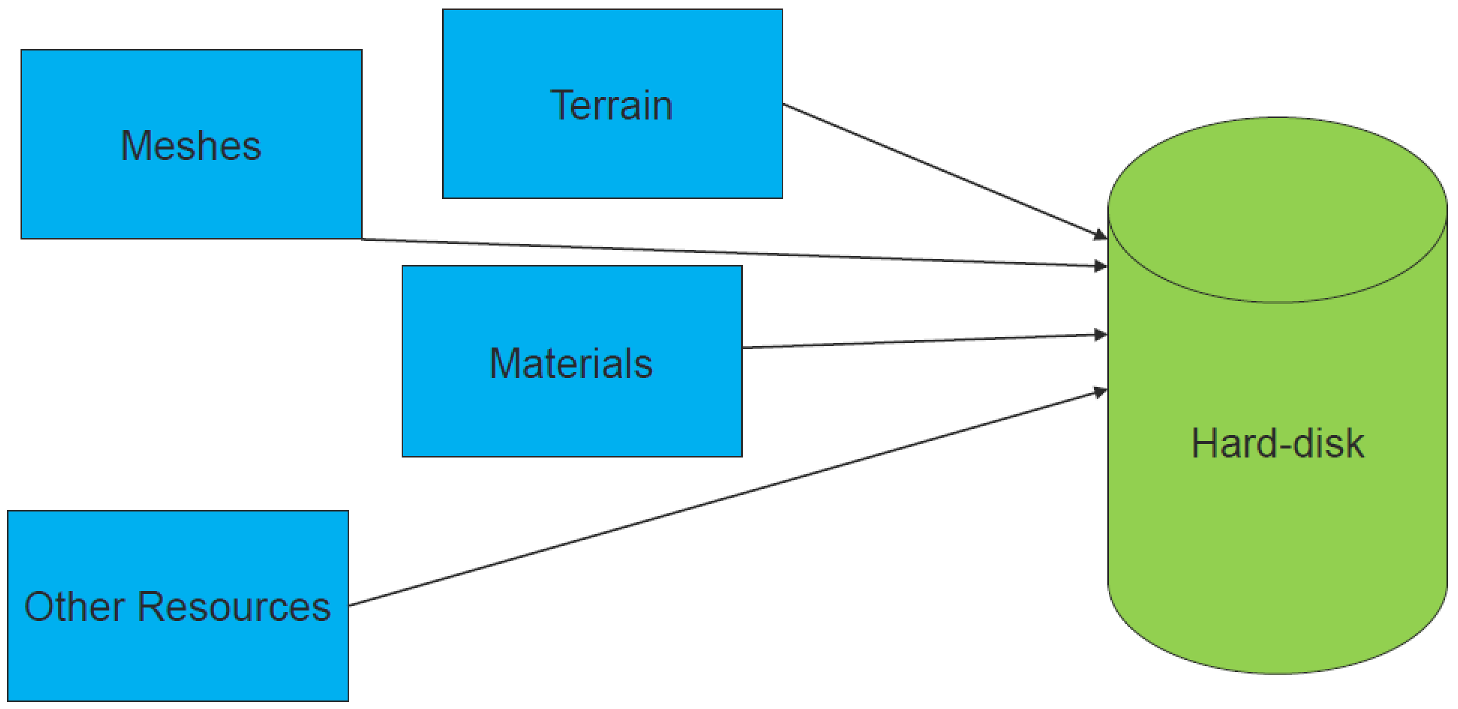
- All the data of the scene/model is contained in the Scene object like all the materials and the meshes. It also contains a reference to the root node of the scene.
- The Root node of the scene may contain children nodes (like all other nodes) and could have a set of indices that point to mesh data in the scene object's `mMeshes` array. The scene's `mMeshes` array contains the actual Mesh objects, the values in the `mMeshes` array of a node are only indices for the scene's meshes array.

ASSIMP Structure III

A (simplistic) model of Assimp's structure is (cont.):

- A Mesh object itself contains all the relevant data required for rendering, think of vertex positions, normal vectors, texture coordinates, faces, and the material of the object.
- A mesh contains several faces. A Face represents a render primitive of the object (triangles, squares, points). A face contains the indices of the vertices that form a primitive. Because the vertices and the indices are separated, this makes it easy for us to render via an index buffer.
- Finally, a mesh also links to a Material object that hosts several functions to retrieve the material properties of an object. Think of colors and/or texture maps (like diffuse and specular maps).

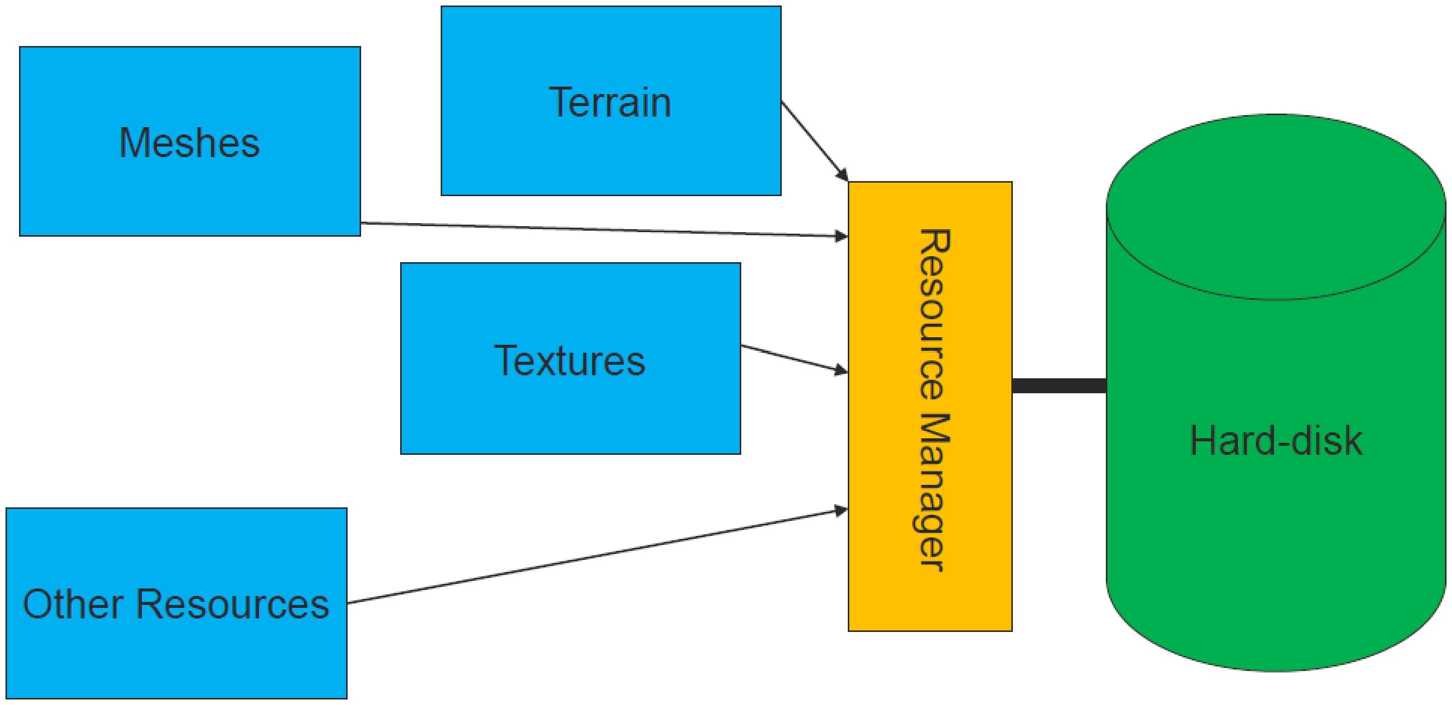
Resource Management



Resource Manager

- Central means of accessing resources
- Prevents more than one copy of the same resource being loaded
- All file/disk access
- Can do batch loading
- Can maintain ‘default’ materials
- Can provide decompression if required (e.g. Zip)
- Avoids each node containing its own mesh, textures, materials, etc

Resource Management



A Resource Manager using ASSIMP

- Code provided this week:
  - Mesh class (along with Node, SubMesh and Material) used to hold information retrieved from model file
  - ResourceManager class (loads resources and reference counts them so that when all nodes have finished with a resource, it can be freed up). Makes use of ASSIMP to load models.
  - MeshNode class. An example of a class that inherits from SceneNode and makes use of the ResourceManager
  - MeshRenderer class. An example rendering class that handles a particular vertex and constant buffer format and a particular set of shaders. This inherits from a base abstract Renderer class. Accompanying this is a new shader file (TexturedShaders.hlsl).
  - A compiled version of ASSIMP built for 64-bit Windows.

What’s Next?

- This week, a tutorial has been provided that shows you how to integrate ASSIMP and the resource manager into your code
- The week after, we will start to look at creating terrain.