Foundations of 3D Scene Modeling

3D scene in big picture

Cornerstones of image generation:

- **3D scene**
- Rendering algorithm
- Raster image

<IMAGE: high-level overview of three main components>

<IMAGE: tree-like structured knowledge of the course>

3D scene for image synthesis

- In order to synthesize image, a rendering process must be executed on a 3D scene
- Elements that are required for rendering process is similar as elements needed when taking real world photograph/video
- There should be observer with **camera** placed somewhere in space, this space should contain **objects** with various shapes and materials and finally, **light source** must be present to shed light onto this space.
  - <image to remember: observer, camera and light>

3D scene elements

- Description of discussed base elements:
  - Objects
  - Lights
  - Camera

  Is called **3D scene modeling** and way of representing virtual environment in memory of a computer.

3D scene modeling questions

Representing 3D scene requires answers to following questions:

- How do we represent scene elements in a computer?
  - Intuitive for user?
  - Tractable for rendering?
- How we create scene elements?
  - How to use scene representations to create real-world phenomena and objects.
  - How do we manipulate 3D models?

Recap

Elements of any 3D scene:

- 3D model(s)
- Light source(s)
- Camera(s)

<IMAGE: high-level overview of three main components>

3D scene

- 3D scene modeling goes hand in hand with object oriented design.
- 3D scene representation has inherent tree-like structure thus often represented with so called **scene-graph**
  - Arrangement between user who builds 3D scene and renderer
  - https://learnopengl.com/Guest-Articles/2021/Scene/Scene-Graph
  - Scene modeling tools: DCC examples

<IMAGE: COMPONENTS OF 3D SCENE AND SCENE GRAPH>

Scene graph

- Hierarhical datastructure for organizing and structuring storing whole 3D scene, with all its elements
- User oriented datastructure for modeling and organizing elements and their relationships in hierarchy for easier manipulation and construction of scene
  - It is edited and created by user: artists and designers
  - Enables easier modeling and animation (e.g., whole subtrees can be translated)
- It serves as support to rendering algorithms. Pass through scene graphs pass for rendering
  - Depending on rules, rendering algorithm might use different materials, geometries, lights, animations, etc.

Scene graph

- Example of simple scene graph

Scene graph

- Elements:
  - Nodes: root, internal and leaf. Each node contains data, at least its position in graph which gives consistent structure
  - Edges
- Graph scene:
  - Root – starting point for whole scene
    - Data: type of coordinate system, scene units, etc.
  - Internal nodes – organize scene into hierarchy. Often those are transformation information (where and how are objects positioned)
  - Leaf nodes – contain elements of the scene: objects, camera, lights. As objects in the scene can repeat, these nodes can be duplicated
  - image

Leaf nodes

- Contain elements of the scene:
  - Object meshes
  - Object materials
  - Lights
  - Cameras
  - EXAMPLES

Internal nodes

- Often those are transformation nodes defining positions of sub-trees: translations, rotations, scaling, etc.
- Other types:
  - Grouping – contain nodes without any other function
  - Conditional – type of grouping node which enables activation only the particular children node
  - Level of detail – contains children nodes where each has a copy of objects with varying level of detail and only one is active depending on camera distance
  - Billboard – groupping node which orientates all children node towards camera
  - EXAMPLES
- As materials and textures are often shared between different objects, they can be stored as internal node which is references by children nodes

Scene graph traversal

- Recursive operation starting from root going through hierarchy
- Scene graph traversal can be used during rendering
- Therefore, it serves as **standardized scene description** that can be shared between different rendering, modeling and interaction tools

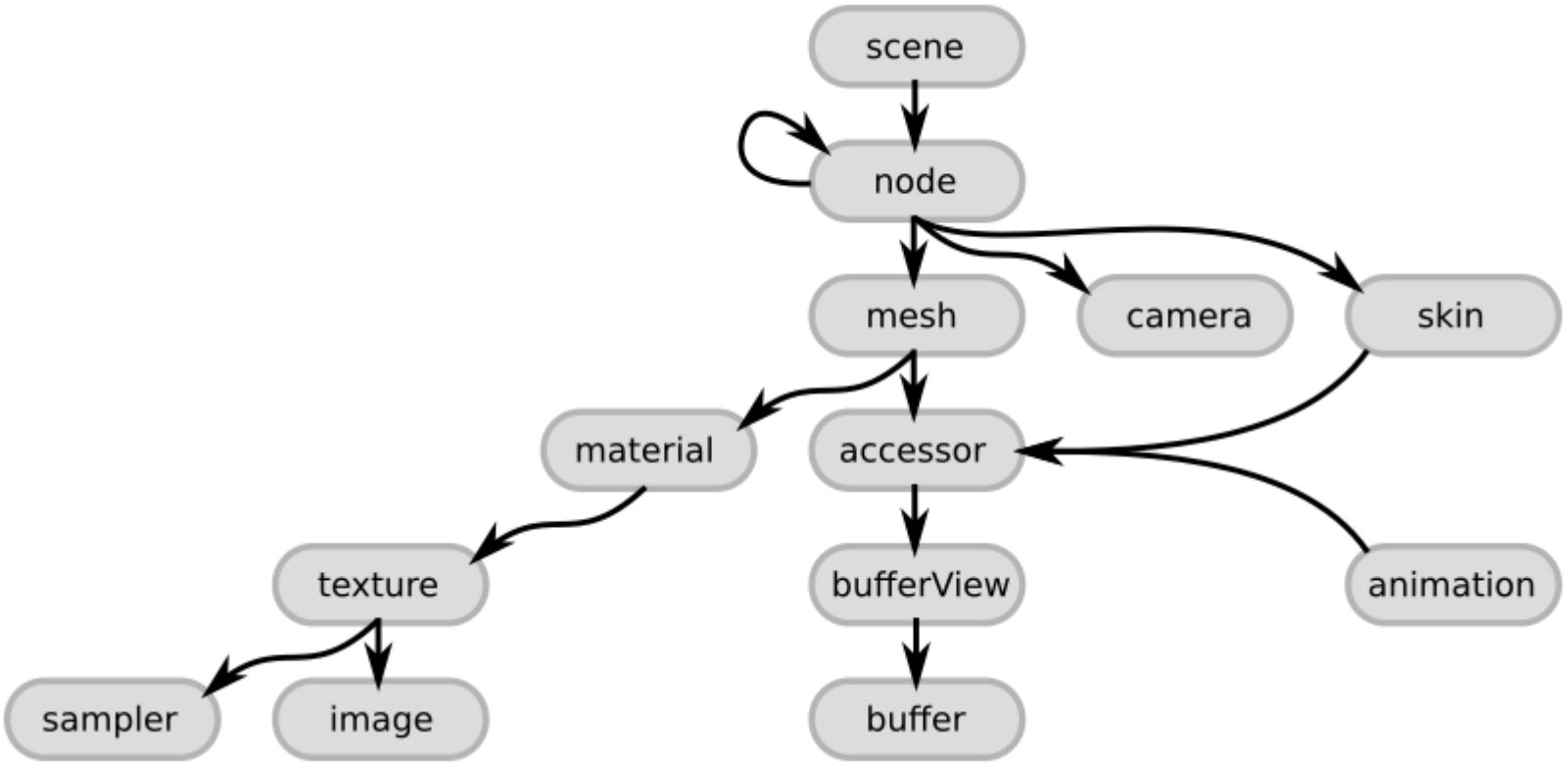
Scene graph and scene transfer

- Different modeling, rendering and interaction software has different formats for scene description
  - Transfer between them requires **standardized formats**
  - For different formats, different **importer/exporter** functions are needed
  - Different formats exists which differ by **scene elements support**
- Storing and transferring scene requires data described by scene graph

Scene graph and scene transfer

- Tendency towards standardized formats is required
- Popular scene description formats:
  - glTF
  - USD: https://graphics.pixar.com/usd/release/index.html

Scene graph example: glTF



https://github.com/KhronosGroup/glTF-Tutorials/blob/master/gltfTutorial/gltfTutorial_002_BasicGltfStructure.md

Scene graph example: glTF

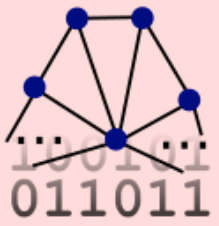
.gltf (JSON) file

```
"scenes": [ ... ],
"nodes": [ ... ],
"cameras": [ ... ],
"animations": [ ... ],
...



"buffers": [
  {
    "uri": "buffer01.bin",
    "byteLength": 102040
  }
],



"images": [
  {
    "uri": "image01.png"
  }
],
```

The JSON part describes the general scene structure, and elements like cameras and animations.
Additionally, it contains links to files with binary data and images:

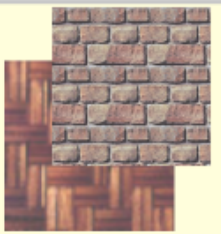
.bin files

Raw data for geometry, animations and skins



.jpg or .png files

Images for the textures of the models



Elements of 3D scene in production

- https://github.com/appleseedhq/appleseed/wiki/Project-File-Format
- Scene graph can be stored in various formats: e.g., XML

- `<project>` !
  - `<scene>` !
    - `<assembly>` *
      - `<assembly>` *
      - `<assembly_instance>` *
        - `<transform>` *
      - `<bsdf>` *
      - `<color>` *
      - `<edf>` *
      - `<light>` *
        - `<transform>` *
      - `<material>` *
      - `<object>` *
      - `<object_instance>` *
        - `<assign_material>` *
        - `<transform>` *
      - `<surface_shader>` *
      - `<texture>` *
      - `<texture_instance>` *
    - `<assembly_instance>` *
      - `<transform>` *
    - `<camera>` *
    - `<color>` *
    - `<environment>` ?
    - `<environment_edf>` ?
    - `<environment_shader>` ?
    - `<texture>` *
    - `<texture_instance>` *
  - `<rules>` +
    - `<render_layer_assignment>` ?
  - `<output>` !
    - `<frame>` ?
  - `<configurations>` !
    - `<configuration>` *

Complex scene

<IMAGE: An motivation image that we will understand by the end of the lecture.>

Literature

- https://github.com/lorentzo/IntroductionToComputerGraphics/wiki/Foundations-of-3D-scene-modeling