Foundations of 3D Scene Modeling

Recap

Cornerstones of image generation:

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

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

Where we are today

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

Introduction

Elements of any 3D scene:

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

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

Complex example

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

Foundations of 3D models

Objects around us.

Representing 3D models requires:

- Representation of **shape** - surface geometry
- Representation of **material** – surface appearance

Foundations of shape representation

Foundational shape representations are:

- Meshes
- Curved surfaces

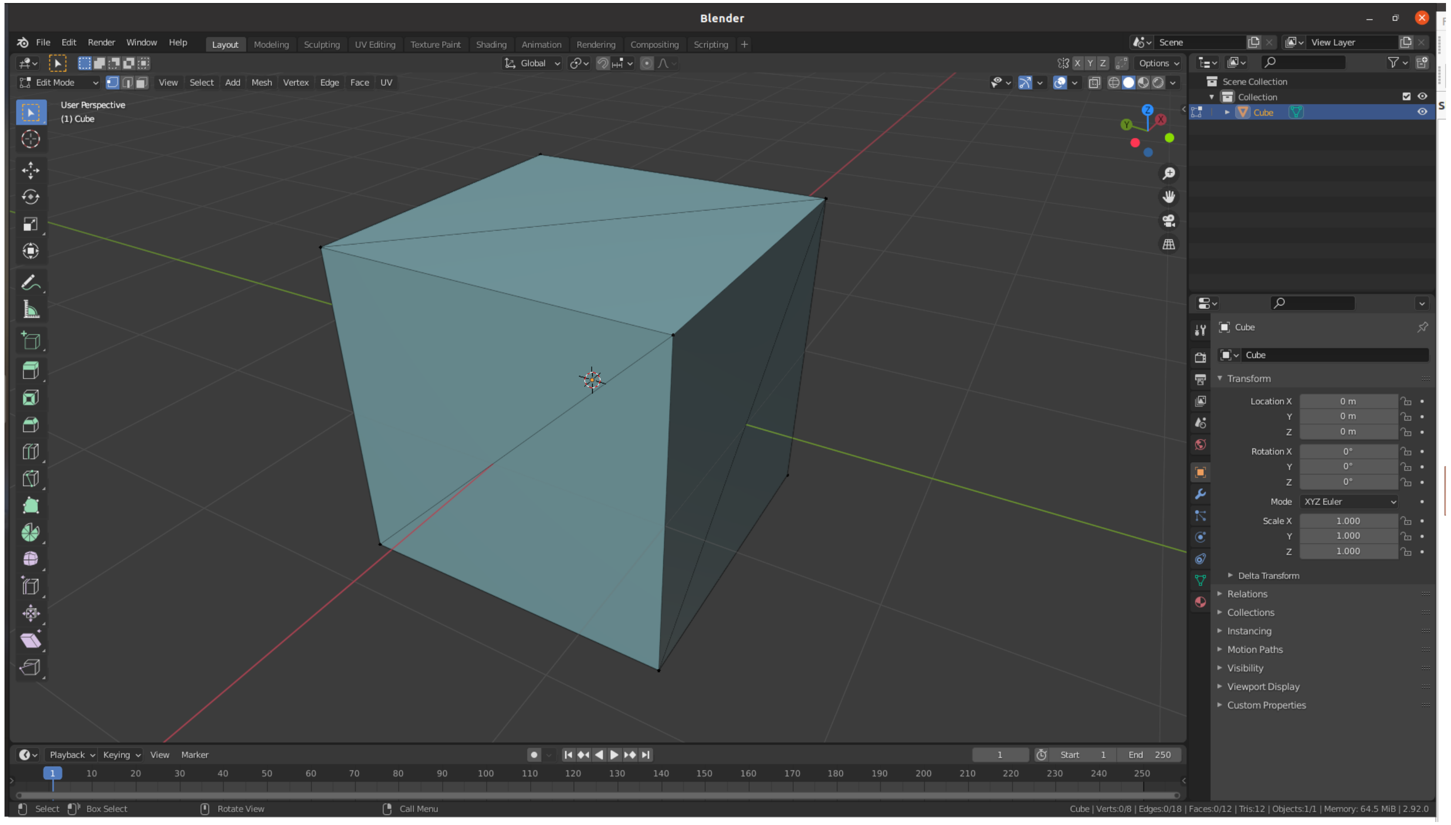
Foundations of Meshes

- Pros:
  - Simple, a lot of effort has been made to represent various shapes with meshes
- Cons:
  - Not every object is well suited to mesh representation:
    - Shapes that have geometrical detail at every level (e.g., fractured marble)
    - Some objects have structure which is unsuitable for mesh representation, e.g., hair which has more compact representations
- Common types:
  - Triangle mesh
  - Quad mesh

Triangle mesh introduction

- Triangle mesh is foundational and most widely used data-structure for representation of a shape in graphics
- Triangle mesh consists of many triangles joined along their edges to form a surface
- Triangle is fundamental and simple primitive:
  - All vertices lie in the same plane
- Triangle mesh has nice properties:
  - Uniformity: simple operations
    - Subdivision: single triangle is replaced with several smaller triangles. Used for smoothing
    - Simplification: replacing the mesh with the simpler one which has the similar shape (topological or geometrical). Used for level of detail

Example how certain flat shapes are created with triangles



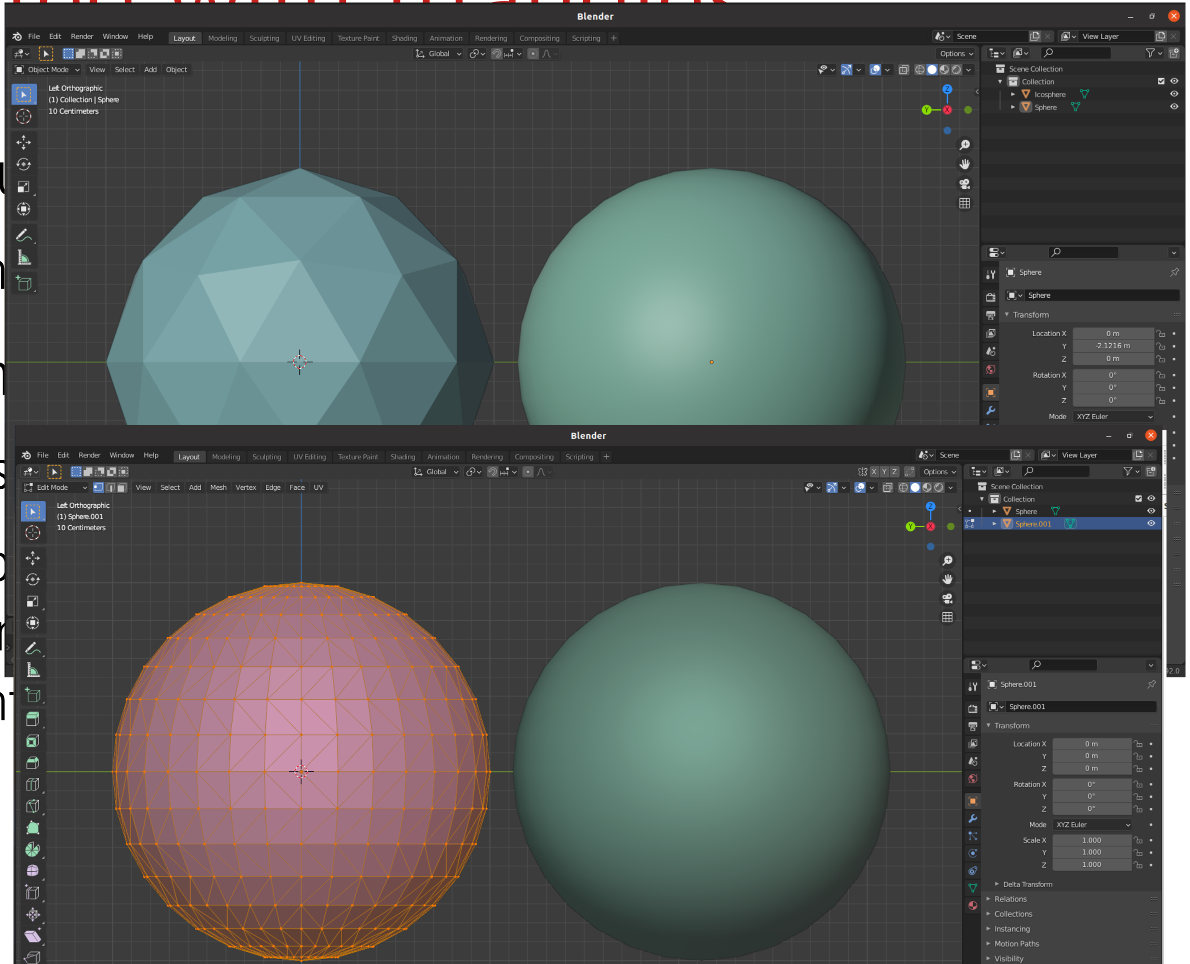
Location X 0 m
Y 0 m
Z 0 m
Rotation X 0°
Y 0°
Z 0°
Mode XYZ Euler
Scale X 1.000
Y 1.000
Z 1.000

Start 1 End 250

Cube | Verts:0/8 | Edges:0/18 | Faces:0/12 | Tris:12 | Objects:1/1 | Memory: 64.5 MiB | 2.92.0

Example how certain curved shapes are approximated with triangles

- Conceptual approximation: find adjacent points with a mesh str
  - For example: scanning and recon
- Example: sphere vs icosahedron
  - Each point on icosahedron is clos
  - Each normal vector of icosahedro point. But, function that assigns n icosahedron is piecewise constan



Blender
File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Scripting
Object Mode View Select Add Object
Left Orthographic
(1) Collection | Sphere
10 Centimeters
Scene Collection
Collection
Icosphere
Sphere
Transform
Edit Mode View Select Add Mesh Vertex Edge Face UV
(1) Sphere.001
Sphere.001
Delta Transform
Relations
Collections
Instancing
Motion Paths
Visibility

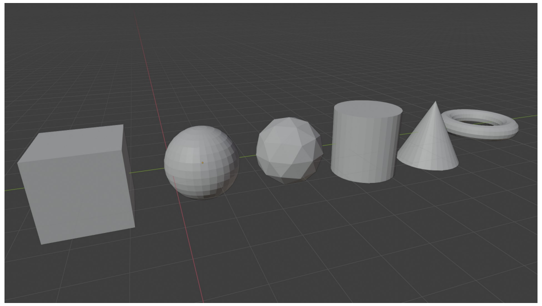
Location X 0 m
Y -2.1216 m
Z 0 m
Rotation X 0°
Y 0°
Z 0°
Mode XYZ Euler

Location X 0 m
Y 0 m
Z 0 m
Rotation X 0°
Y 0°
Z 0°
Mode XYZ Euler
Scale X 1.000
Y 1.000
Z 1.000

Common basic shapes

- Now we can understand how to represent basic shapes using triangle meshes
- Every DCC Tool provides basic shapes:
  - Blender[1], Maya[2], 3DSMax[3], Houdini[4], etc.

1. https://docs.blender.org/manual/en/latest/modeling/meshes/primitives.html
2. https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2022/ENU/Maya-Basics/files/GUID-45D2EAD4-5BCF-42DA-A1AB-EC6EE09FE705-htm.html
3. https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2021/ENU/3DSMax-Modeling/files/GUID-66152BDE-BA64-423F-8472-C1F0EB409E16-htm.html
4. https://www.sidefx.com/docs/houdini/model/create.html



Complex shapes?

- How to represent complex shapes with triangle meshes?
- Digression: drawing complex form (3D shape)
  - Anything can be decomposed in simple forms[1,2]: box, sphere, cylinder, torus, cones, etc.

1. http://www.thedrawingwebsite.com/2015/02/18/practicing-your-draw-fu-forms-forms-are-like-sentences/

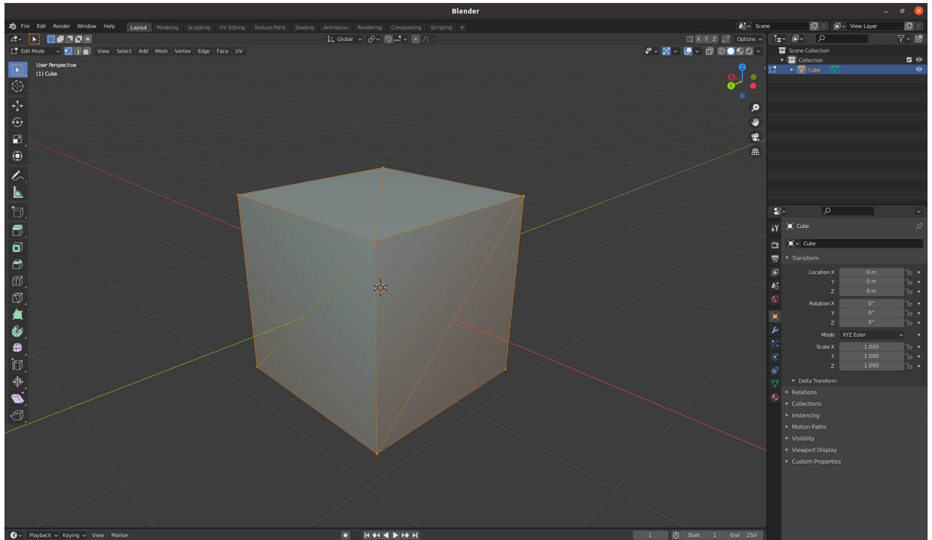
2. https://www.youtube.com/watch?v=6T_-DiAzYBc&list=RDCMUClM2LuQ1q5WEc23462tQzBg&start_radio=1&rv=6T_-DiAzYBc&t=1343&ab_channel=Proko

Complex shapes

- Choose right basic shape: https://www.youtube.com/watch?v=DcyY4RAHcA4&ab_channel=CGCookie
- Modeling with basic shapes (Blender):
  - https://www.youtube.com/watch?v=Q0qKO2JYR3Y&ab_channel=BlenderSecrets
  - https://www.youtube.com/watch?v=AD3gn2AyzgA&ab_channel=LeeDanielsART
  - Procedural (and funny one): https://www.youtube.com/watch?v=Hf8s1Ckycdo&ab_channel=CGMatter
- Modeling with basic shapes (Maya)
  - https://www.youtube.com/watch?v=j3jwVfN8EcU&ab_channel=AnetaV
- Procedural shapes (Houdini):
  - https://www.youtube.com/watch?v=afHVjiNeH7A&ab_channel=AdrienLambert
  - https://www.youtube.com/watch?v=fxOxygaEOFk&ab_channel=SimonHoudini
  - Note that other geometry representation are used as well

Description of meshes

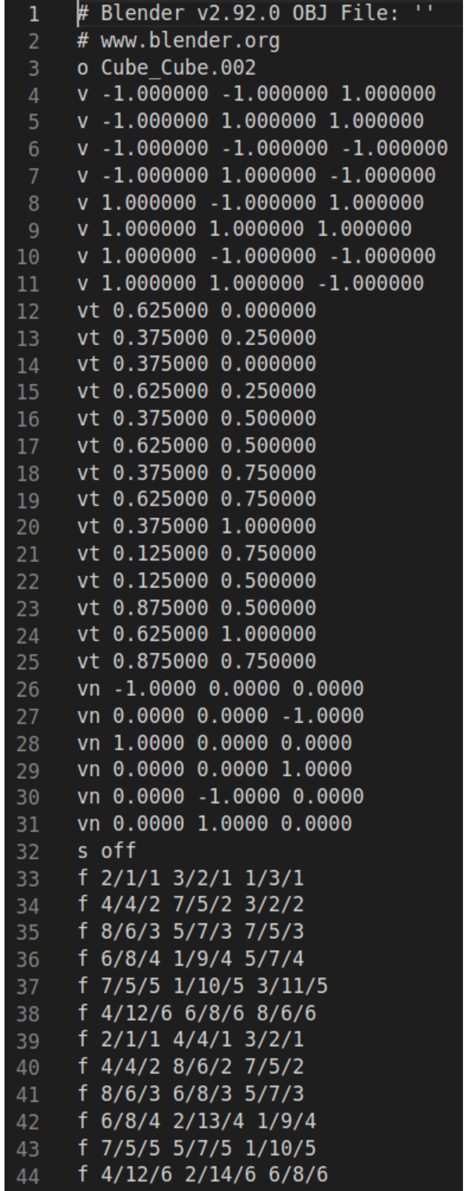
- Description of mesh requires:
  - List of vertices and triangles (edges are inferred from triangles)
  - Vertex table – geometry
  - Triangle (faces) table - topology


Blender
File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Scripting
Edit Mode View Select Add Mesh Vertex Edge Face UV
User Perspective
(1) Cube
Scene Collection
Collection
Cube
Playback Keying View Marker
Start 1 End 250

Transform

Location X 0 m
Y 0 m
Z 0 m
Rotation X 0°
Y 0°
Z 0°
Mode XYZ Euler
Scale X 1.000
Y 1.000
Z 1.000

Delta Transform
Relations
Collections
Instancing
Motion Paths
Visibility
Viewport Display
Custom Properties



```
# Blender v2.92.0 OBJ File: ''
# www.blender.org
o Cube_Cube.002
v -1.000000 -1.000000 1.000000
v -1.000000 1.000000 1.000000
v -1.000000 -1.000000 -1.000000
v -1.000000 1.000000 -1.000000
v 1.000000 -1.000000 1.000000
v 1.000000 1.000000 1.000000
v 1.000000 -1.000000 -1.000000
v 1.000000 1.000000 -1.000000
vt 0.625000 0.000000
vt 0.375000 0.250000
vt 0.375000 0.000000
vt 0.625000 0.250000
vt 0.375000 0.500000
vt 0.625000 0.500000
vt 0.375000 0.750000
vt 0.625000 0.750000
vt 0.375000 1.000000
vt 0.125000 0.750000
vt 0.125000 0.500000
vt 0.875000 0.500000
vt 0.625000 1.000000
vt 0.875000 0.750000
vn -1.0000 0.0000 0.0000
vn 0.0000 0.0000 -1.0000
vn 1.0000 0.0000 0.0000
vn 0.0000 0.0000 1.0000
vn 0.0000 -1.0000 0.0000
vn 0.0000 1.0000 0.0000
s off
f 2/1/1 3/2/1 1/3/1
f 4/4/2 7/5/2 3/2/2
f 8/6/3 5/7/3 7/5/3
f 6/8/4 1/9/4 5/7/4
f 7/5/5 1/10/5 3/11/5
f 4/12/6 6/8/6 8/6/6
f 2/1/1 4/4/1 3/2/1
f 4/4/2 8/6/2 7/5/2
f 8/6/3 6/8/3 5/7/3
f 6/8/4 2/13/4 1/9/4
f 7/5/5 5/7/5 1/10/5
f 4/12/6 2/14/6 6/8/6
```

Storing and transferring mesh

- Mesh datastructure can be stored in various formats which:
  - Are more or less compact
  - Are more or less human-readable
  - Can contain additional object data which is described with the mesh (textures, materials, etc.)
  - Can contain various metadata (e.g., physical behavior of object described with mesh)
  - Store only mesh information
  - Store whole scene and mesh is only one of elements
- Popular formats:
  - https://all3dp.com/2/most-common-3d-file-formats-model/
  - https://www.sidefx.com/docs/houdini/io/formats/geometry_formats.html
- 3D scene is not necessary created, rendered and used in same software. Usually, whole pipeline of software is used, at least:
  - DCC → game engines

Important properties of meshes

- Goal: not to go deep into definitions but rather to verify properties using simpler methods
- Mesh boundary: formal sum of vertices
- Closed mesh: mesh boundary is zero. Required for defining what is “inside” and “outside” by winding number rule
- Manifold mesh: each vertex has arriving and leaving edge
  - Manifolds are desired since it is easy to work with them (both manually and algorithmically)
  - Smooth vs not smooth manifolds (e.g., cube)
  - Self-intersecting meshes are not manifolds
  - In graphics we generally use polyhedral manifolds
- Oriented vs unoriented meshes
  - We use oriented meshes so that boundary can be defined

Quad mesh

- Often used as a modeling primitive
- Complexity:
  - Easy to create a quad where not all vertices lie on a plane
- In graphics pipeline it is always transformed to triangle.
- In raytracing rendering plane-ray intersection must be defined

Foundations of Curved surfaces

Foundations of surface material

Foundations of Light sources

- Sources of Light
- Models of light:
  - Distant lights
  - Point lights

Distant lights

Point lights

Foundations of camera

How camera works

Pinhole camera model and parameters

Cameras introduction

- In rendering camera model is required to define perspective projection and simulate a real world camera.
- Now, we will understand how cameras work and how to simulate a real world camera.
- Such camera model is similar to ones used in production software (e.g., Blender, Maya, 3DSMax, Houdini)

Transformations of 3D scene elements

Translation

Rotation

Scale

Coordinate systems

Coming together

A 3D scene completed

Back to the complex example

- What have we learned

Literature

- Computer graphics: Principles and practice (J.F.Hughes)
- https://github.com/lorentzo/IntroductionToComputerGraphics/wiki/Foundations-of-3D-scene-modeling