Game Engine Architecture IT3935

Lecture 3

Summary

- 3D Model Data
- Animation

What is a 3D Model?

- A 3D Model is a bunch of data (vertices, normals, texture coordinates, color etc) which *usually* represents an object in the 3D world

Usage of models

- 3D models can be used to

  Render an instance or entity in the scene

  Represent collision data (for trigger points, bounding volumes etc)

Categories of a 3D Model

- A model can be divided into several categories

  - Static Geometry
  - Screen Aligned Geometry (Billboard)
  - Animated Geometry

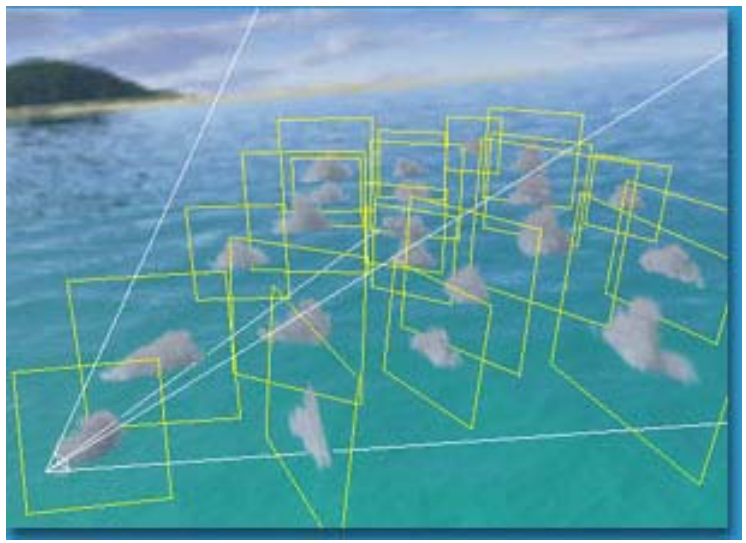
Static Geometry

- Are used for objects that do not animate, e.g. statues, walls, doors.
- Data can be represented in world space instead of local model space to avoid local->world transformation
- Static lighting can be baked into textures
- Common file formats are .X, .OBJ, .3DS

Screen Aligned Geometry

- A 2D plane which is always aligned with the camera, commonly known as billboards
- Can animate texture coordinates (UV), blend and render several billboards together, resize at runtime
- Can be used for particles (explosions, fire etc), clouds, skybox, lens flare, foliage, UI, shadow maps

Billboard Clouds



- Taken from *Real Time Cloud Rendering* by Mark J Harris

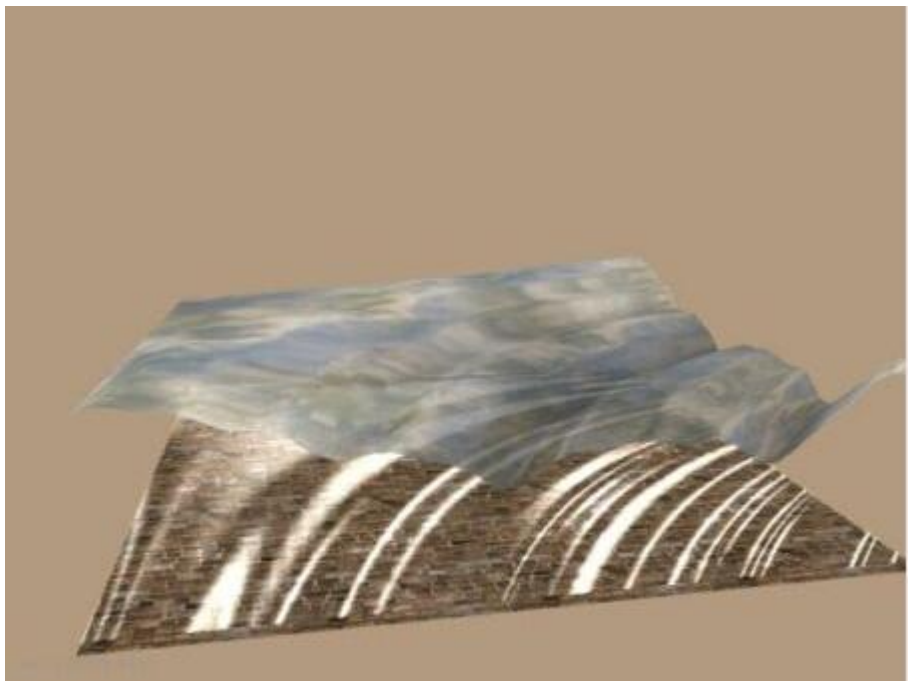
Animated Geometry

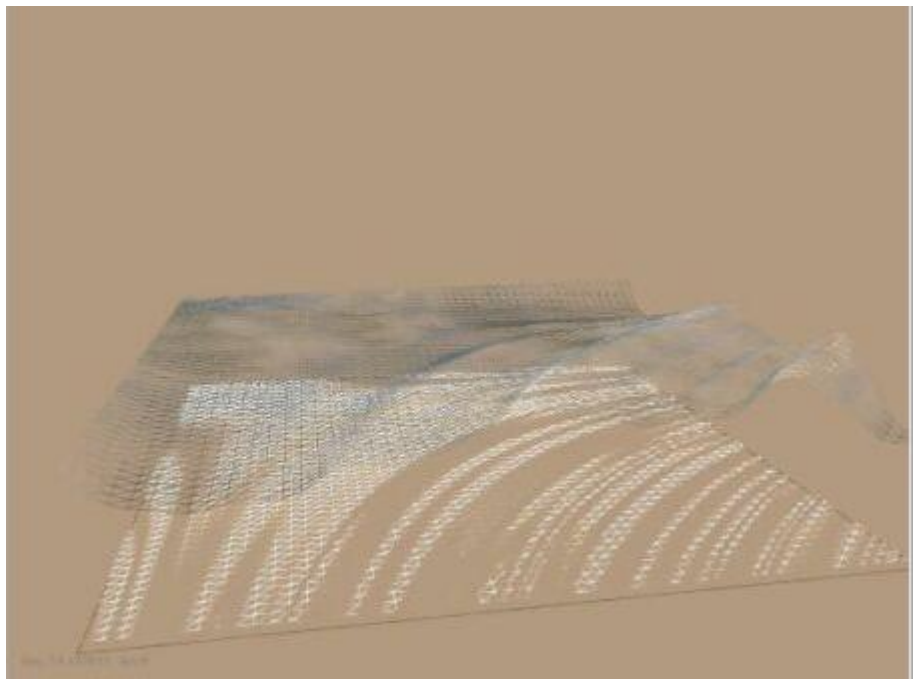
- Can be used for characters, morphing objects, water, cloth, fluid
- Animation data is divided into two types – vertex and bone

Vertex Animation

- Vertex animation or morphing, is moving the positions of each vertex every frame.
- Disadvantage is huge data size as each vertex must store vertex positions for each frame
- Used in older engines like Quake 2, .md2 format

Vertex Animation - Water





- Taken from "*Inexpensive Underwater Caustics Using Cg" – Gamasutra 2003*

Vertex Animation - Cloth

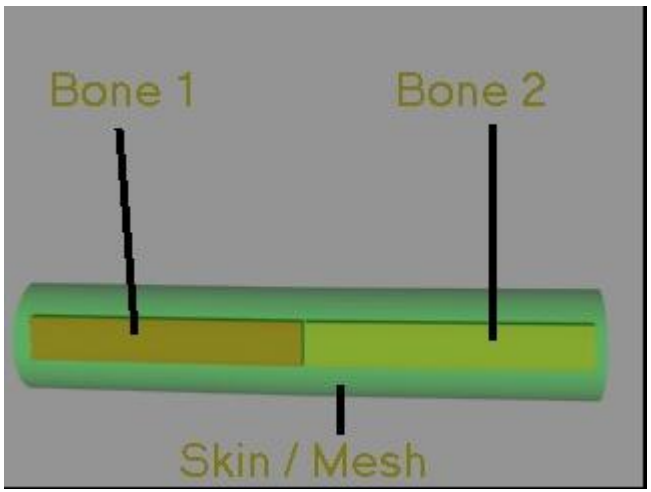
- Taken from *Interactive Cloth Sim* - Nvidia GDC 2001

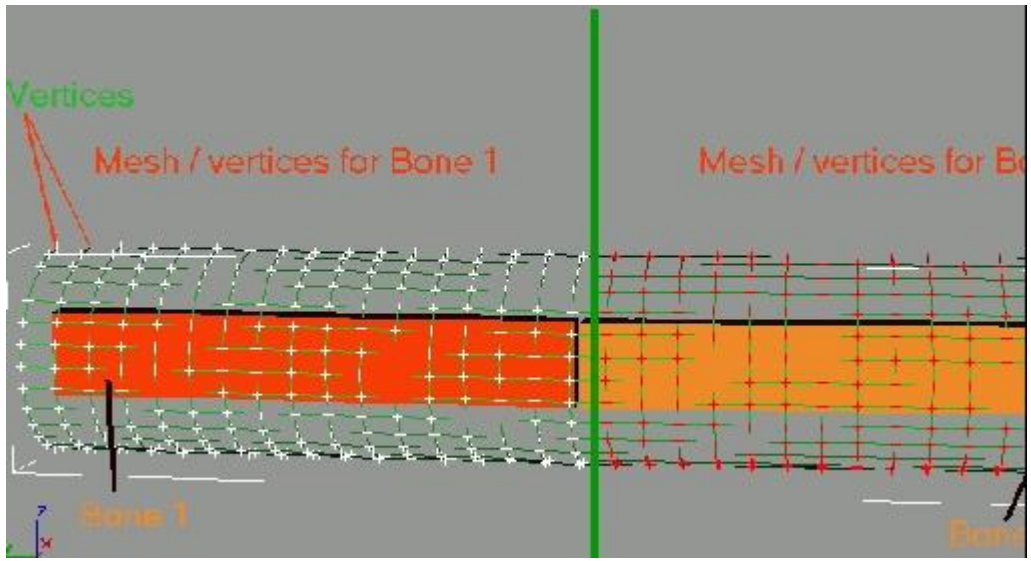
Bone Animation - Skinning

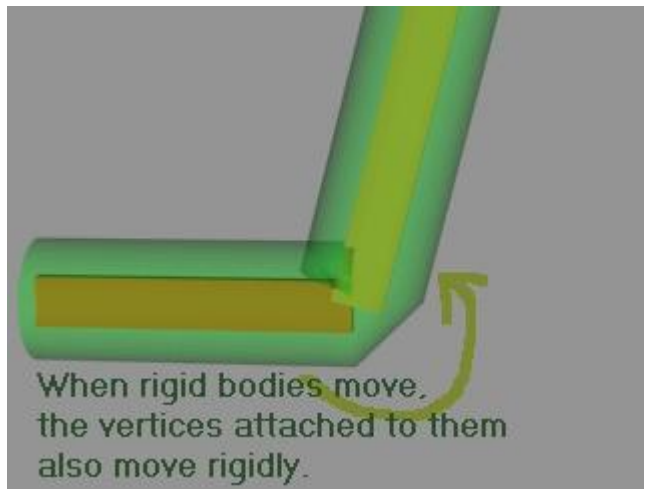
- Instead of storing the positions of each vertex, we store a bone hierarchy and animate the bone. Vertices are calculated and blended at runtime based on the bone they are affected by. These vertices are called a skin

- Skinning is slower since we need to calculate vertex positions. We trade memory with speed.

Skinning





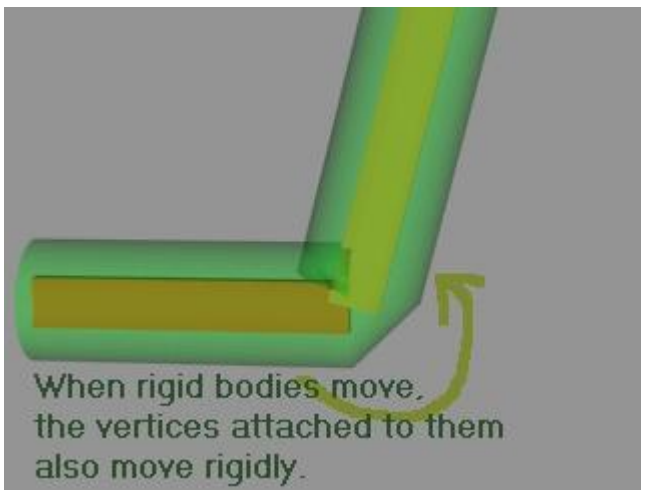


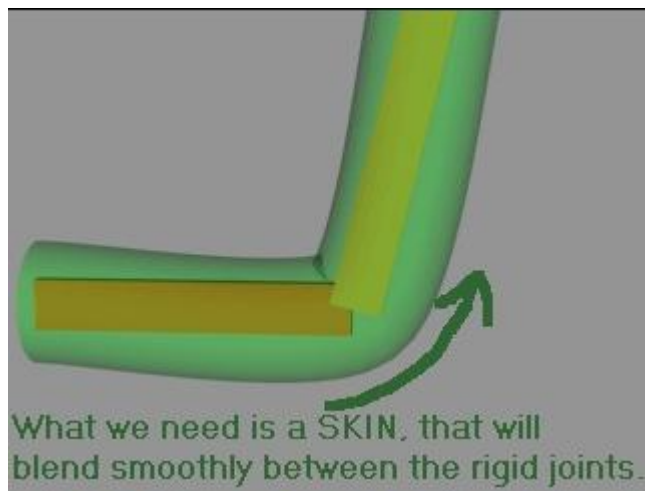
- Taken from *Flipcode.com*

Skinning

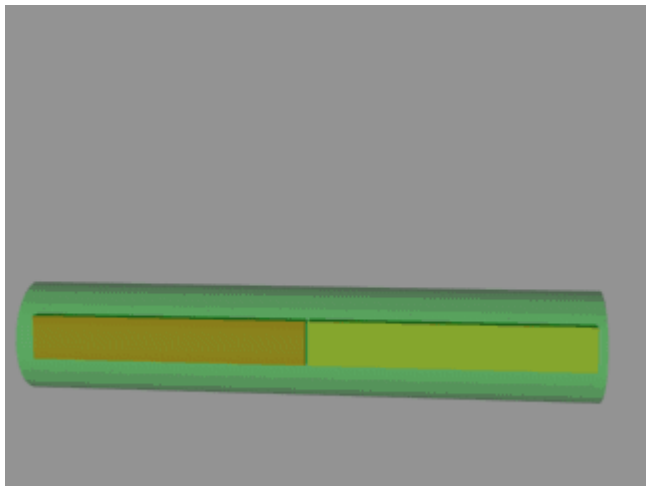
- One problem is skinning is the vertex movement is rigid

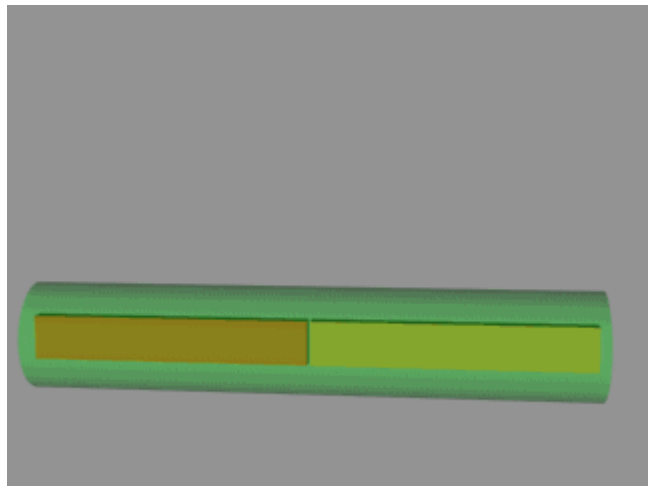
- Require vertex blending when each vertex is affected by multiple bones. Each vertex stores a weight percentage per bone and is usually limited to 4 bones (hardware limit).





Skinning





Animation Data

- *What do you need to store?*

KeyFrame

- Animation data is stored per keyframe instead of every frame. A keyframe is a pose in the animation cycle and stores the position and orientation of every bone in that pose.

- Two kind of keyframes – forward and inverse kinematics

Forward Kinematics

- The position and orientation of each bone is stored per keyframe is stored and interpolation (linear, cubic etc) is used to calculate the vertices between each keyframe.

- This is just simple bone animation

Inverse kinematics

- Given a final position, how should my bone bend so I can reach it?
- No animation data is stored as the bone animation data is calculated at runtime based on desired position

End

- Questions?