DH2323 DGI15

INTRODUCTION TO
COMPUTER GRAPHICS AND INTERACTION

IMAGE-BASED RENDERING AND ANIMATION

Christopher Peters
HPCViz, KTH Royal Institute of Technology, Sweden
chpeters@kth.se
http://kth.academia.edu/ChristopherEdwardPeters

Based on DGI12 notes by Carl Henrik Ek
Can divide pipeline into three conceptual stages:

- **Application** (input, animations, think SDL)
- **Geometry** (transforms, projections, lighting)
- **Rasteriser** (draw image as pixels)

These define the core structure of the pipeline.
Geometry Stage

Responsible for polygon and vertex operations
Consists of five sub-stages:

- Model and View Transform
- Lighting and Shading
- Projection
- Clipping
- Screen Mapping
What is OpenGL?

Software interface to graphics hardware
Commands for interactive three-dimensional graphics
Hardware independent interface
Drawing operations performed by underlying system and hardware
I. Image-based Rendering
Image-based Rendering

X constitutes:
- Geometry
- Texture
- Lighting
- Material

Parameterisation of the 'world'
Image-based Rendering

How does $f(.)$ manipulate $X$?
- Light
- Surface interaction
- Light transport

Formulate model $f(.)$ through assumptions
Image-based Rendering

Computer Vision
  – Image easy to acquire
  – Solve inverse problem
  – What parameters have generated the image?
Image-based Rendering

Challenge:
Given an image recover the parameters
- Texture
- Light
- Geometry
Models still valid
- $x_i = f^{-1}(y_i)$
Image-based Rendering

Rendering: generate images from viewpoints
Image-based rendering: replace geometry and material attributes with real images
Most realistic image? A photograph
  - Lacks flexibility
  - Cannot change lighting or viewpoint
  - Combine images to produce a new one
Epipolar Geometry

The geometry of stereo vision
Two cameras viewing 3D scene from different positions
Study geometric relations between 3D points and their 2D projections
More images = more constraints
II.

Animation
Animation

Basis
Showing consecutive related static images one after another produces the perception of a moving image

Traditional Animation
Master artists draw certain important key-frames in the animation
Apprentices draw the multitude of frames in-between these key-frames
Called tweens
In Practice

For computers animation:
Object has an initial configuration and a final configuration (often specified by the artist)
Computer must figure out the intermediate configurations: \textit{interpolation}

Orientation Interpolation:
Given two key-frame orientations, figure out an intermediate orientation at a certain point between them

Possible solutions: Euler angles and rotation matrix interpolation
#1: Euler Angles

- An Euler angle is a rotation around a single axis
- Any orientation can be specified by composing three rotations
  - Each rotation is around one of the principle axes
  - i.e. \((x, y, z)\) – first rotate around \(x\), then \(y\), then \(z\)
  - Think of roll, pitch and yaw of a flight simulator
- When rotating about a single axis, is possible to interpolate a single value
  - However, for more than one axis, interpolating individual angles will not work well
  - Unpredictable intermediate rotations
  - Gimbal lock
#2: Rotation Matrices

Interpolating between two rotation matrices does not result in a rotation matrix

- Does not preserve rigidity of angles and lengths
- This result of an interpolation of 0.5 between the identity matrix and 90 degrees around the x-axis does not produce a valid rotation matrix:

\[
\text{Interpolate} \left( \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \right) \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & -0.5 & 0.5 \end{bmatrix}
\]
Solution

- Use *quaternion interpolation*
- Quaternions don’t suffer from Gimbal lock
- Can be represented as 4 numbers instead of 9 of a 3x3 matrix
- Trivial conversion between angle/axis representation
- Interpolation between two quaternions is easy (once you know how)
- Quaternion looks like this:
  \[ q[w,(x,y,z)] \quad \text{also written } q[w,v] \text{ where } v = (x,y,z) \]
  \[ q = w + xi + yj + zk \]
Representation

• For a right-hand rotation of $\theta$ radians about unit vector $v$, quaternion is:

$$q = (\cos(\theta/2); v \sin(\theta/2))$$

- Note how the 3 imaginary coordinates are noted as a vector
- Only unit quaternions represent rotations
  - Such a quaternion describes a point on the 4D unit hypersphere
- Important note: $q$ and $-q$ represent the exact same orientation
- Different methods for doing quaternion interpolation: LERP, SLERP (Spherical linear interpolation)
In Practice

• Not always the best choice
  • Quaternions are (as you will have noticed) hard to visualise and think about
  • If another method will do and is simpler, it will be a more appropriate choice

• But…
  • Extremely useful in many situations where other representations are awkward
  • Easy to use in your own programs once you have a quaternion class
  • See GLM library
III.

Where to next?
Advice: learn Fixed Function Pipeline first

Shading languages include: HLSL, GLSL, CG

Simple GLSL shader example

```
#version 400
layout( triangles, equal_spacing ) in;

void main( )
{
    vec4 p0 = gl_in[0].gl_Position;
    vec4 p1 = gl_in[1].gl_Position;
    vec4 p2 = gl_in[2].gl_Position;

    vec3 p = gl_TessCoord.xyz;

    gl_Position = p0*p.x + p1*p.y + p2*p.z;
}
```
Middleware

Relieve the tedium of recoding everything
Already used in labs: SDL and GLM
Many other libraries and engines available
Some examples:
  – GLEW: http://glew.sourceforge.net/
  – Assimp: http://assimp.sourceforge.net/
  – ODE: http://www.ode.org/
  – FMOD: http://www.fmod.org
  – OGRE: http://www.ogre3d.org/
  – Torque 3D: http://www.garagegames.com/
Crowds

 Aggregate Dynamics for Dense Crowd Simulation, Narain et al., UNC, 2010

Metropolis Project, GV2, Trinity College Dublin
Toolchains

<- Environment mark-up tool (*Metroped*)

Results transferred into final rendering client ->

Metropolis Project, GV2, Trinity College Dublin
Procedural Cities

Introversion Procedural City Generator

ESRI City Engine
Virtual Sculpting project supervised by Mario Romero

The system intersects the kinect-computed depth map with the volume and removes the voxels in the intersection.
Real-time Procedural Universe

Infinity: The Quest for Earth, I-Novae Studios
Links

http://www.pandromeda.com/
MojoWorld: planet synthesis

Paper on middleware for games and graphics
Reminders

- You should be working on Lab 3
- Labs due on 8\textsuperscript{th} May
  - Bilda DGI15 DH2323 lab submission is now open

- Next lab help session:
  - More information to follow…
Next lecture

Bezier Curves, Splines and Surfaces

- Guest speaker: Prof. Tino Weinkauf
- This Wednesday (6th May), B2
- 15:00 – 17:00