A Whirlwind Introduction to Computer Graphics

Mike Bailey
Course Goals

• Provide a background for the amazing things you will hear about in the other SIGGRAPH 2020 venues

• Create an understanding of common computer graphics vocabulary

• Help you understand the significance of the images and animations that you will see

• Provide references for further study

http://cs.oregonstate.edu/~mjb/whirlwind
Welcome! I'm happy to be here. I hope you are too!

Mike Bailey

- Professor of Computer Science, Oregon State University
- Has been in computer graphics for over 30 years
- Has had over 8,000 students in his university classes
- mjb@cs.oregonstate.edu

http://cs.oregonstate.edu/~mjb/whirlwind
Sections

• How the computer graphics pieces fit together
• Modeling
• Rendering
• Animation
• Finding More Information

http://cs.oregonstate.edu/~mjb/whirlwind
What are all the pieces that go into making the graphics you will be see this week?
What does it take to make them?
The Process

- 3D Geometric Models
- Surface Information
- Lighting Information
- Texture Information
- Drawing
- 3D Movement Definition

Let's Do This.
The Process

Modeling
- 3D Geometric Models
- Surface Information
- Texture Information

Rendering
- Lighting Information
- Drawing

Animation
- 3D Movement Definition

LET'S DO THIS.
Creating 3D Geometry
3D Geometric Models

Surface Information

Texture Information

Modeling

Rendering

Lighting Information

Drawing

Animation

3D Movement Definition

LET’S DO THIS.
What do we mean by “Modeling”?

In computer graphics applications, how we model geometry depends on what we would like to use the geometry for:

• Looking at its appearance
• Interacting with its shape?
• How does it interact with its environment?
• What is its surface area and volume?
• Will it be able to be 3D-printed?
• Etc.

Want to experiment with some free modeling programs? Want some notes on how to get started?
http://cs.oregonstate.edu/~mjb/blender
http://cs.oregonstate.edu/~mjb/sketchup
http://cs.oregonstate.edu/~mjb/tinkercad
Explicitly Listing Geometry and Topology

Models defined this way can consist of thousands of vertices and faces – we need some way to describe them effectively.

This is often called a **Mesh**.

http://graphics.stanford.edu/data/3Dscanrep
Explicitly Listing Geometry and Topology -- Quadrilaterals

static GLfloat CubeVertices[ ][3] =
{
    { -1., -1., -1. },
    {  1., -1., -1. },
    { -1.,  1., -1. },
    {  1.,  1., -1. },
    { -1., -1.,  1. },
    {  1., -1.,  1. },
    { -1.,  1.,  1. },
    {  1.,  1.,  1. }
};

static GLfloat CubeColors[ ][3] =
{
    { 0., 0., 0. },
    { 1., 0., 0. },
    { 0., 1., 0. },
    { 1., 1., 0. },
    { 0., 0., 1. },
    { 1., 0., 1. },
    { 0., 1., 1. },
    { 1., 1., 1. }
};

static GLuint CubeIndices[ ][4] =
{
    { 0, 2, 3, 1 },
    { 4, 5, 7, 6 },
    { 1, 3, 7, 5 },
    { 0, 4, 6, 2 },
    { 2, 6, 7, 3 },
    { 0, 1, 5, 4 }
};
More likely we would use Triangles

```c
GLuint CubeIndices[4] = {
    {0, 2, 3, 1},
    {4, 5, 7, 6},
    {1, 3, 7, 5},
    {0, 4, 6, 2},
    {2, 6, 7, 3},
    {0, 1, 5, 4}
};

GLuint TriangleCubeIndices[3] = {
    {0, 2, 3},
    {0, 3, 1},
    {4, 5, 7},
    {4, 7, 6},
    {1, 3, 7},
    {1, 7, 5},
    {0, 4, 6},
    {0, 6, 2},
    {2, 6, 7},
    {2, 7, 3},
    {0, 1, 5},
    {0, 5, 4}
};
```
Triangular Meshes are Very Important These Days Because 3D Printing Requires a Triangular Mesh Data Format
3D geometric modeling at its very best -- mmmm... :-)
Another way to Model:
Remember Venn Diagrams (2D Boolean Operators) from High School?

Two Overlapping Shapes

Union

Intersection

Difference
Solid Modeling Using 3D Boolean Operators

Two Overlapping Solids

Union

Intersection

Difference

This is often called Constructive Solid Geometry, or CSG
Another way to Model:
Curve Sculpting – Bézier Curve Sculpting

\[ P(t) = (1 - t)^3 P_0 + 3t(1 - t)^2 P_1 + 3t^2 (1 - t) P_2 + t^3 P_3 \]

\[ 0 \leq t \leq 1. \]
Curve Sculpting – Bézier Curve Sculpting Example

[Diagram showing Bézier curves being used to sculpt a face]

curves.mp4
A Small Amount of Input Change Results in a Large Amount of Output Change
Another way to Model: Surface Sculpting

In general, these are referred to as **Patches**. These, in particular, are Bezier patches. Non-uniform Rational B-spline Surfaces, or NURBS, are another popular type.

Like the curve sculpting, a **Small** Amount of Input Change Results in a **Large** Amount of Output Change
Surface Equations can also be used for Analysis

Showing Contour Lines

Showing Curvature
Another Way to Model: Volume Sculpting

This is often called a “Lattice” or a “Cage”.

lattice.mp4
Geometric Models can also be used for Physical Simulation

Blender’s Explosion feature

Blender’s Smoke feature
Object Modeling Rules for 3D Printing

The object must be a legal solid. It must have a definite inside and a definite outside. It can’t have any missing face pieces.
Object Modeling Rules for 3D Printing

Objects cannot pass through other objects. If you want two shapes together, do a CSG union on them so that they become one complete object.

Overlapped in 3D -- bad

Boolean union -- good
Creating an image
Rendering

Rendering is the process of creating an image of geometric modes. Again, there are questions you need to ask first:

• Why am I doing this?
• How realistic do I want this image to be?
• How much compute time do I want this to take?
• Do I need to take lighting into account?
• Does the illumination need to be global or will local do?
• Do I need to create shadows?
• Do I need to create reflections and refractions?

Want to experiment with a free rendering program?
Want some notes on how to get started?
http://cs.oregonstate.edu/~mjb/blender
Why Do We Care About Lighting?

Lighting makes it possible to tell the difference between surfaces or parts of surfaces.
Local vs. Global Illumination

Local

Global
A Common type of Local Illumination: Ambient-Diffuse-Specular (ADS)

- Point being illuminated: P
- Light intensity: I
- Unit vector from point to light: L
- Unit vector surface normal: n
- Perfect reflection unit vector: R
- Unit vector to eye position: E

Common type of Local Illumination: Ambient-Diffuse-Specular (ADS)
A Common type of Local Illumination: Ambient-Diffuse-Specular (ADS)

1. Ambient = a constant
   Accounts for light bouncing “everywhere”

2. Diffuse = I*cosΘ
   Accounts for the angle between the incoming light and the surface normal

3. Specular = I*cos^Sφ
   Accounts for the angle between the “perfect reflector” and the eye; also the exponent, S, accounts for surface shininess

Note that cosΘ is just the dot product between unit vectors L and n

Note that cosφ is just the dot product between unit vectors R and E
Diffuse Lighting works because of spreading out the same amount of light energy across more surface area

\[ \text{Diffuse} = I \cdot \cos \Theta \]
The Specular Lighting equation is a heuristic that approximates reflection from a rough surface

\[
\text{Specular} = I^* \cos^S \phi
\]

\( S \approx \text{“shininess”} \)

\( 1/S \approx \text{“roughness”} \)
Put them all together!
Global Illumination: The Rendering Equation

\[ B(P, d_0, \lambda) = E(P, d_0, \lambda) + \int_{\Omega} B(P, d_i, \lambda) f(\lambda, d_i, d_0)(d_i \cdot \hat{n}) d\Omega \]

Light arriving at Point P from everywhere

Light departing from Point P in the direction that we are viewing the scene from
The Rendering Equation

\[ B(P, d_0, \lambda) = E(P, d_0, \lambda) + \int_{\Omega} B(P, d_i, \lambda) f(\lambda, d_i, d_0)(d_i \cdot \hat{n}) d\Omega \]

In plain language, this is a light balance equation:

“The light shining from the point P towards your eye is the reflection of the incoming light directed to the point P from all of the other points in the scene.”
The Lighting Equation at Work

- The left wall is green.
- The right wall is red.
- The back wall is white.
- The ceiling is blue with a light source in the middle of it.
- The objects sitting on the floor are white.

If the appearance of an object is also affected by the appearances of other objects, then you have **Global Illumination**.

http://www.swardson.com/unm/tutorials/mentalRay3/
When light hits a surface, it bounces in particular ways depending on the angle and the material.

This distribution of bounced light rays is called the 
**Bidirectional Reflectance Distribution Function**, or BRDF.

For a given material, the BRDF behavior of a light ray is a function of 4 variables: the 2 spherical coordinates of the incoming ray and the 2 spherical coordinates of the outgoing ray.
Usually it is easier to trace from the eye
Physically-Based Rendering

Let light bounce around the scene, depending on how the different materials behave.
Physically-Based Rendering
Physically-Based Rendering
Physically-Based Rendering
Clearly this is capable of spawning an infinite number of rays. How do we handle this?

For a small-ish number of bounces, we can evenly distribute a collection of rays.

For lots of bounces, it’s **Monte Carlo** simulation to the rescue!

\[
\text{LightGathered} = \frac{\sum_{0}^{N-1} \text{ResultOfRaysCastInRandomDirection}}{N}
\]

Recurse by applying this equation for all ray hits (yikes!)
Physically-Based Rendering using the Blender Cycles Renderer
An Example from the Title Slide
Another Physically-Based Rendering Example

Image by Josiah Blaisdell
Tricky Lighting Situations

Watch for these at the conference and in CG movies!
**Subsurface Scattering**

SS models light bouncing around *within* an object before coming back out.

This is a good way to represent skin, wax, milk, paraffin, etc.
Creating the motion you want
3D Movement Definition

Animation

Modeling

Rendering

3D Geometric Models
Surface Information
Lighting Information
Texture Information
Drawing

LET'S DO THIS.
Rendering is the process of giving motion to your geometric modes. Again, there are questions you need to ask first:

- Why am I doing this?
- Do I want the animation to obey the real laws of physics?
- Am I willing to “fake” the physics to get the objects to want to move in a way that I tell it?
- Do I have specific key positions I want the objects to pass through no matter what?
- Do I want to simply record the motion of a real person, animal, etc., and then play it back?

Want to experiment with a free animation program? Want some notes on how to get started? http://cs.oregonstate.edu/~mjb/blender
Keyframe Animation

Forcing the geometry to smoothly pass through key positions
Keyframe Animation

anim2.mp4
Rigging Animation

Control the movement of groups of vertices with an armature
Forward Kinematics: Transformation Hierarchies

**Input:** Angles

**Output:** Locations
Forward Kinematics:
Change Parameters – Things Move
(All Children Understand This)
Inverse Kinematics

Forward Kinematics solves the problem “if I know the link transformation parameters, where are the links?".

Inverse Kinematics (IK) solves the problem “If I know where I want the end of the chain to be (X*,Y*), what transformation parameters will put it there?"
Inverse Kinematics (IK):
Things Need to Move – What Parameters Will Make Them Do That?
Animating a Human-ish Form

Start with this …

… and turn it into a kinematic model:
Particle Systems: A Cross Between Modeling and Animation?

Check out this movie! These are particles animated on a GPU.
Particle Systems: A Cross Between Modeling and Animation?

The basic process is:

- Emit
- Display
- Update
- Random Number Generator
Particle Systems Examples

Images by Chuck Evans
Particle Systems Examples
Newton’s second law:

\[ \text{force} = \text{mass} \times \text{acceleration} \]

or:

\[ \text{acceleration} = \frac{\text{force}}{\text{mass}} \]

In order to make this work, you need to supply physical properties such as mass, center of mass, moment of inertia, coefficients of friction, coefficients of restitution, etc.
Animating using Fluid Physics

fluid.avi
Animing using Spring Physics

\[ \text{Force} = F \]

\[ (D - D_0) = \frac{F}{k} \]

\[ k = \text{spring stiffness} \text{ in Newtons/meter or pounds/inch} \]

Or, if you know the displacement, the force exerted by the spring is:

\[ F = k(D - D_0) \]

This is known as Hooke’s Law
Animating using the Physics of a Mesh of Springs

“Lumped Masses”
Simulating a Bouncy Chain
Placing a Physical Barrier in the Scene
Animating Cloth
Cloth Examples
Cloth Example

cloth.mp4
The Challenge: animate a collection of objects, each trying to move to a target, but without colliding with each other
Functional Animation:
Make the Object *Want* to Move Towards a Goal Position . . .

\[ m\ddot{x} + c\dot{x} + k = 0 \]
Functional Animation:

. . . While Making it *Want* to Keep Away from all other Objects

\[ m\ddot{x} = \sum F_{\text{repulsive}} \]

- Repulsion Coefficient
- \[ F_{\text{repulsive}} = \frac{C_{\text{repulse}}}{d^\text{Power}} \]
- Distance between the boundaries of the 2 bodies
- Repulsion Exponent
Total Goal – Make the Free Body Move Towards its Final Position While Being Repelled by the Other Bodies

\[ m\ddot{x} + c\dot{x} + k = \sum F \]
Functional Animation

avoid.mp4
Increasing the Stiffness

Stiffness = 3, 6, 9
Increasing the Repulsion Coefficient

Repulse = 10, 30, 50
Motion Capture ("MoCap") as an Input for Animation
Conclusions!

- SIGGRAPH moments will never come again. Well, this is usually true, but through 2020 videos, they might. But, be aware of what is going to be archived and what isn’t.

- Especially take advantage of the un-archived events because you cannot re-live them afterwards.

- Combine what you have just learned here with what else you learn at the conference and relate them to your career and life goals.

- Have fun!
Finding More Information

http://cs.oregonstate.edu/~mjb/whirlwind
Check out the More Information Document!

Where to Find More Information about Computer Graphics and Related Topics

Mike Bailey  
Oregon State University

1. References

1.1 General Computer Graphics

SIGGRAPH Online Bibliography Database:  
http://www.siggraph.org/learn/computer-graphics-bibliography-database


http://cs.oregonstate.edu/~mjb/whirlwind
This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License

http://cs.oregonstate.edu/~mjb/whirlwind