

**SIGGRAPH** THINK BEYOND  
2020 S2020.SIGGRAPH.ORG

# A Whirlwind Introduction to Computer Graphics

Mike Bailey

© 2020 SIGGRAPH. ALL RIGHTS RESERVED.

**SIGGRAPH** THINK BEYOND

Mike Bailey  
Oregon State University  
mjb@cs.oregonstate.edu

Oregon State University  
Computer Graphics

2

CC BY-NC-ND  
This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License

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

© 2020 SIGGRAPH

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>

3

**SIGGRAPH** THINK BEYOND

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](mailto:mjb@cs.oregonstate.edu)

Welcome! I'm happy to be here. I hope you are, too!

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

4

**SIGGRAPH** THINK BEYOND

Sections

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

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

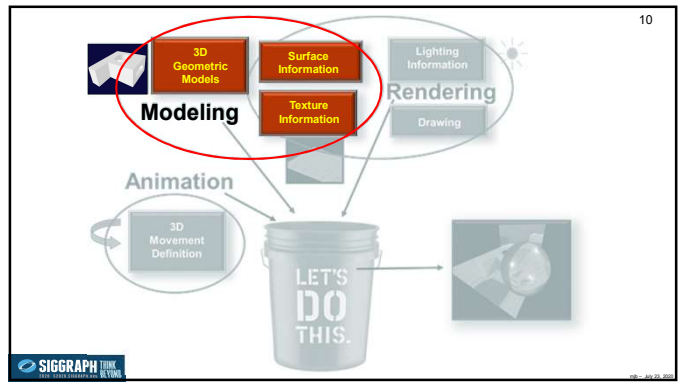
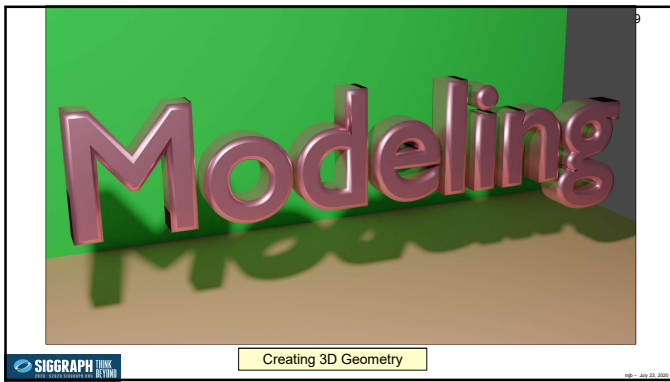
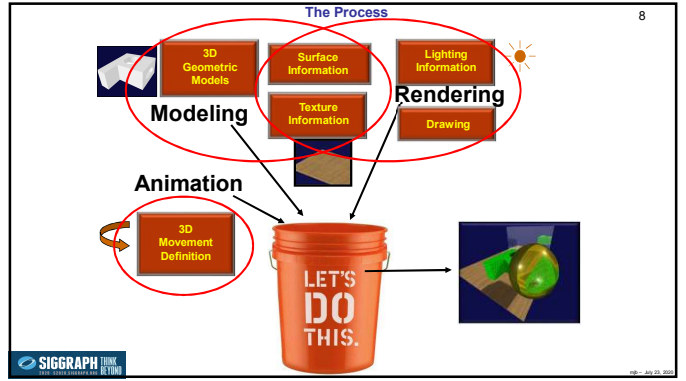
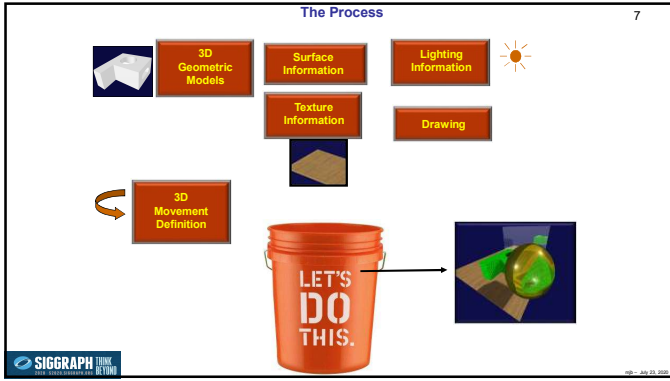
5

**SIGGRAPH** THINK BEYOND

What are all the pieces that go into making the graphics you will be seeing this week? What does it take to make them?

6

**SIGGRAPH** THINK BEYOND



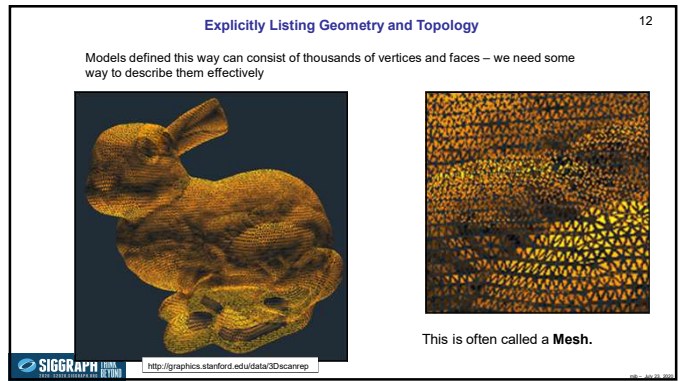
What do we mean by "Modeling"? 11

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>

SIGGRAPH THINK BEYOND



**Cube Mesh Example** 13

SIGGRAPH THINK RETURN

**Explicitly Listing Geometry and Topology -- Quadrilaterals** 14

```

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 }
};
  
```

SIGGRAPH THINK RETURN

**More likely we would use Triangles** 15

```

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 }
};

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 }
};
  
```

SIGGRAPH THINK RETURN

**Triangular Meshes are Very Important These Days Because 3D Printing Requires a Triangular Mesh Data Format** 16

SIGGRAPH THINK RETURN

**3D geometric modeling at its very best -- mmmm... :-)** 17

SIGGRAPH THINK RETURN

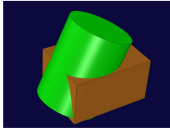
**Another way to Model: Remember Venn Diagrams (2D Boolean Operators) from High School?** 18

Two Overlapping Shapes      Union

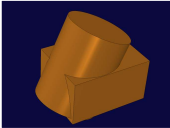
Intersection      Difference

SIGGRAPH THINK RETURN

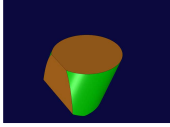
### Solid Modeling Using 3D Boolean Operators



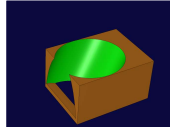
Two Overlapping Solids



Union



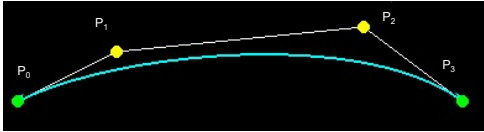
Intersection



Difference

This is often called **Constructive Solid Geometry**, or **CSG**

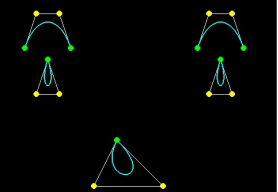
### Another way to Model: Curve Sculpting – Bézier Curve Sculpting

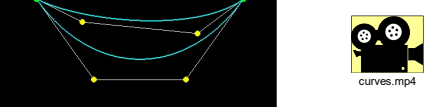


$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t) P_2 + t^3 P_3$$

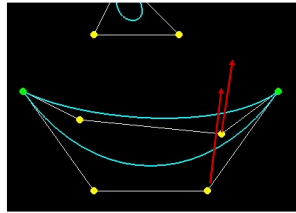
$$0 \leq t \leq 1.$$

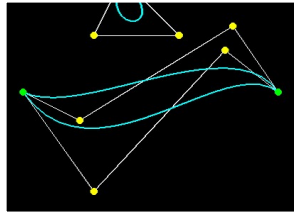
### Curve Sculpting – Bézier Curve Sculpting Example





### Curve Sculpting – Bézier Curve Sculpting Example

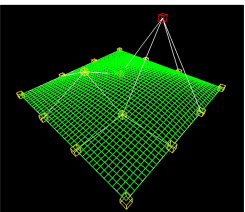




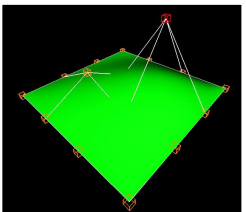
A *Small* Amount of Input Change Results in a *Large* Amount of Output Change

### Another way to Model: Surface Sculpting

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



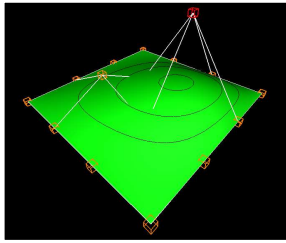
Wireframe



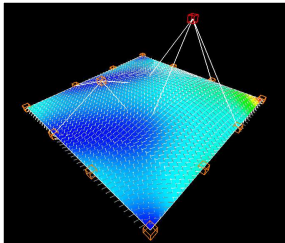
Surface

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 25

This is often called a "Lattice" or a "Cage".

lattice.mp4

Geometric Models can also be used for Physical Simulation 26

Blender's Explosion feature

Blender's Smoke feature

Object Modeling Rules for 3D Printing 27

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.

Missing face

Object Modeling Rules for 3D Printing 28

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

29

Creating an image

30

## Rendering

31

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/~mb/blender>

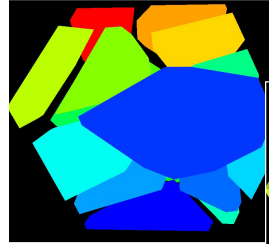


08 - 09 25 2008

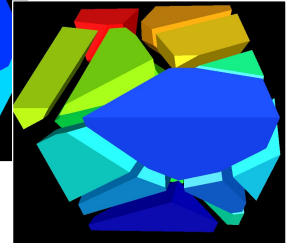
## Why Do We Care About Lighting?

32

No lighting



Lighting



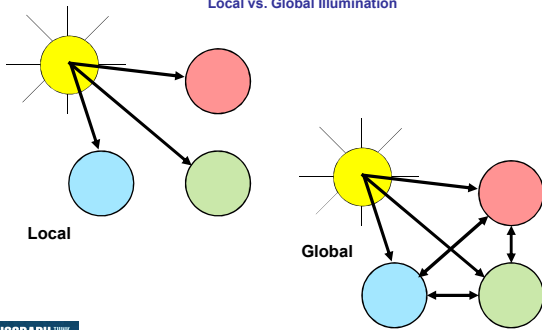
Lighting makes it possible to tell the difference between surfaces or parts of surfaces



08 - 09 25 2008

## Local vs. Global Illumination

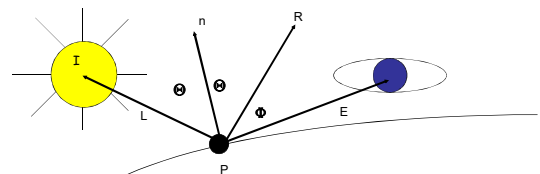
33



08 - 09 25 2008

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

34



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



08 - 09 25 2008

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

35

1. Ambient = a constant Accounts for light bouncing "everywhere"
2. Diffuse =  $I \cdot \cos\theta$  Accounts for the angle between the incoming light and the surface normal
3. Specular =  $I \cdot \cos^S\phi$  Accounts for the angle between the "perfect reflector" and the eye; also the exponent, S, accounts for surface shininess

Note that  $\cos\theta$  is just the dot product between unit vectors L and n

Note that  $\cos\phi$  is just the dot product between unit vectors R and E

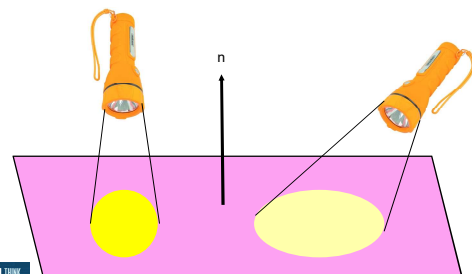


08 - 09 25 2008

## Diffuse Lighting works because of spreading out the same amount of light energy across more surface area

36

$$\text{Diffuse} = I \cdot \cos\theta$$



08 - 09 25 2008

The **Specular Lighting** equation is a heuristic that approximates reflection from a rough surface

37

**Specular** =  $I \cdot \cos^2 \phi$

S = "shininess"  
1/S = "roughness"

SIGGRAPH THINK BEYOND

Put them all together!

38

Ambient + Diffuse + Specular =

lighting.mp4

SIGGRAPH THINK BEYOND

**Global Illumination: The Rendering Equation**

39

$\Omega$  Light arriving at Point P from everywhere

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

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

SIGGRAPH THINK BEYOND

**The Rendering Equation**

40

$\Omega$

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

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."

SIGGRAPH THINK BEYOND

**The Lighting Equation at Work**

41

- 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/>

SIGGRAPH THINK BEYOND

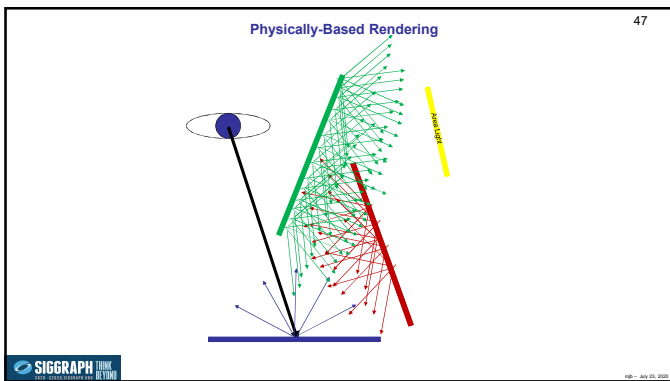
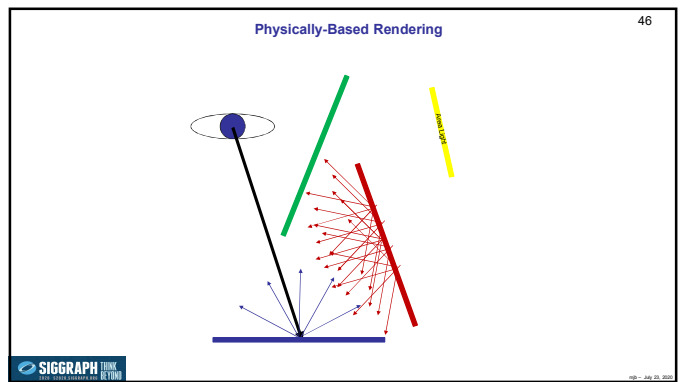
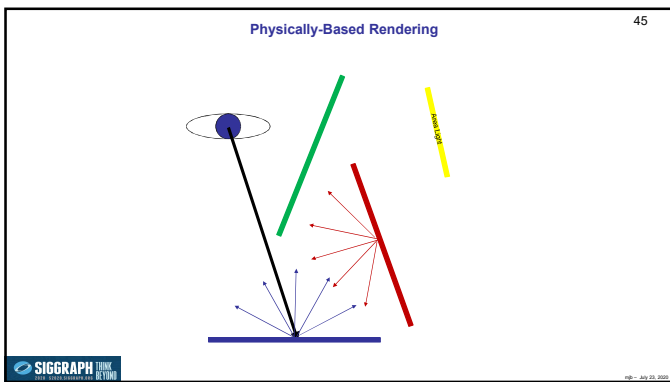
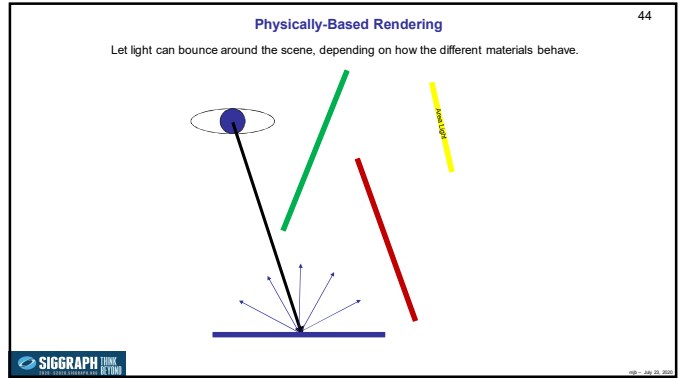
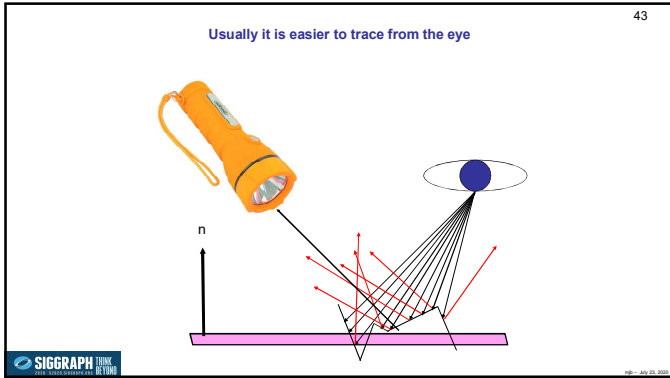
**When light hits a surface, it bounces in particular ways depending on the angle and the material**

42

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.

SIGGRAPH THINK BEYOND



Physically-Based Rendering

48

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!

$$LightGathered = \frac{\sum_{i=1}^{N-1} ResultOfRaysCastInRandomDirection}{N}$$

Recurse by applying this equation for all ray hits (yikes!)

SIGGRAPH THINK BEYOND

Physically-Based Rendering using the Blender Cycles Renderer 49

An Example from the Title Slide 50

Another Physically-Based Rendering Example 51

Image by Josiah Blaisdell

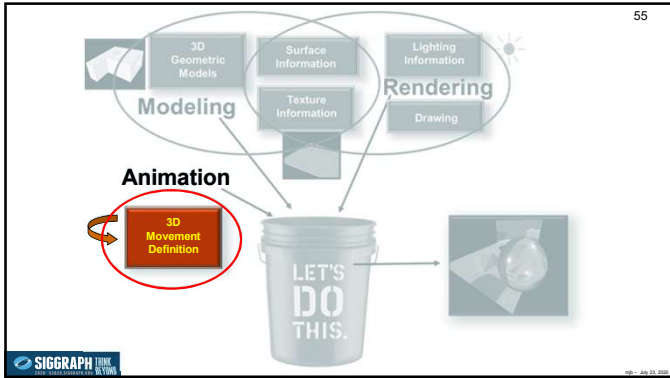
Tricky Lighting Situations 52

Watch for these at the conference and in CG movies!

Subsurface Scattering 53

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



56

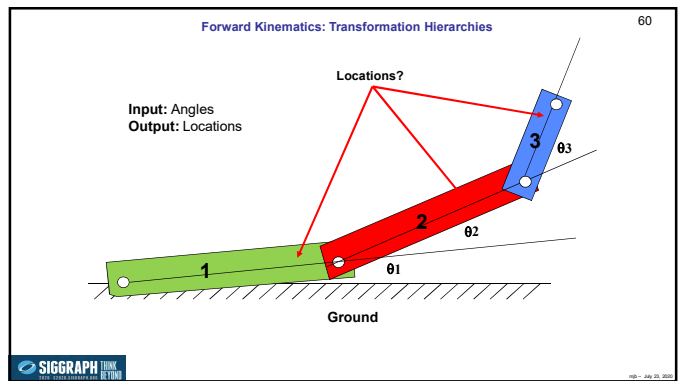
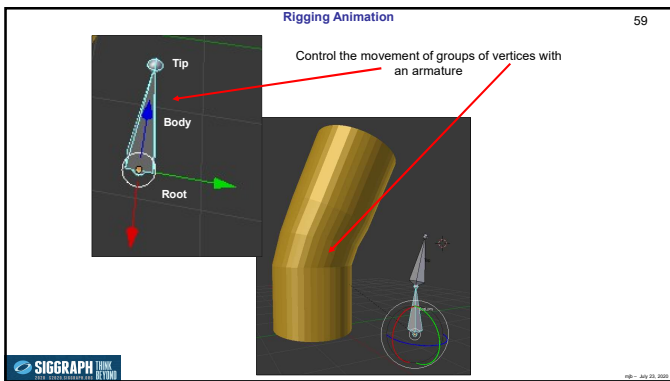
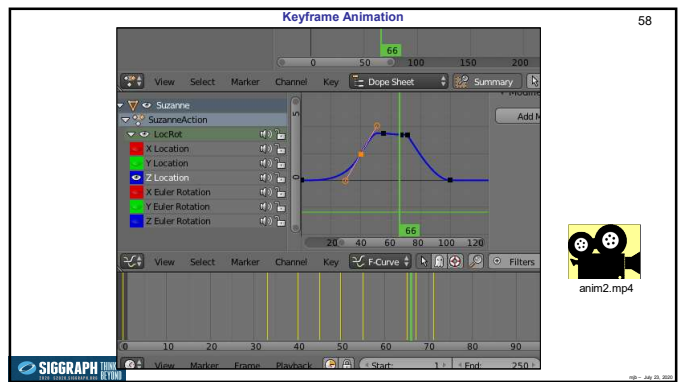
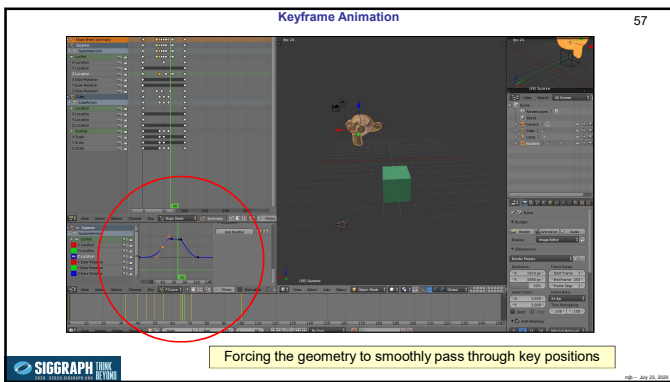
**Animation**

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>

SIGGRAPH THINK BY UNIVERSITY OF TWIN



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

61

SIGGRAPH THINK BEYOND

**Inverse Kinematics**

62

**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?"

Input: Locations  
Output: Angles

SIGGRAPH THINK BEYOND

**Inverse Kinematics (IK):**  
Things Need to Move – What Parameters Will Make Them Do That?

63

SIGGRAPH THINK BEYOND

**Animating a Human-ish Form**

64

Start with this ...

... and turn it into a kinematic model:

SIGGRAPH THINK BEYOND

**Particle Systems:**  
A Cross Between Modeling and Animation?

65

gpu\_particles.mp4

Check out this movie! These are particles animated on a GPU.

SIGGRAPH THINK BEYOND

**Particle Systems:**  
A Cross Between Modeling and Animation?

66

The basic process is:

```

    graph TD
      RNG[Random Number Generator] --> Emit[Emit]
      Emit --> Display[Display]
      Display --> Update[Update]
      Update --> Display
  
```

SIGGRAPH THINK BEYOND

Particle Systems Examples 67

Images by Chuck Evans

SIGGRAPH THINK RETURN

Particle Systems Examples 68

particles.mp4

SIGGRAPH THINK RETURN

Animating using Rigid-body Physics 69

Newton's second law:  
force = mass \* acceleration  
or:  
acceleration = force / mass

dominos2.mp4

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.

SIGGRAPH THINK RETURN

Animating using Fluid Physics 70

fluid.avi

SIGGRAPH THINK RETURN

Animating using Spring Physics 71

$D_0$  = unloaded spring length

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

$k$  = spring stiffness 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

SIGGRAPH THINK RETURN

Animating using the Physics of a Mesh of Springs 72

"Lumped Masses"

SIGGRAPH THINK RETURN

73

Simulating a Bouncy Chain

chain.mp4

SIGGRAPH THINK BEYOND

74

Placing a Physical Barrier in the Scene

SIGGRAPH THINK BEYOND

75

Animating Cloth

SIGGRAPH THINK BEYOND

76

Cloth Examples

SIGGRAPH THINK BEYOND

77

Cloth Example

cloth.mp4

SIGGRAPH THINK BEYOND

78

Functional Animation – “Fake Physics”

The Challenge: animate a collection of objects, each trying to move to a target, but without colliding with each other

SIGGRAPH THINK BEYOND

Functional Animation:  
Make the Object Want to Move Towards a Goal Position ...

79

$m\ddot{x} + c\dot{x} + kx = 0$

SIGGRAPH

Functional Animation:  
... While Making it Want to Keep Away from all other Objects

80

$m\ddot{x} = \sum F_{repulsive}$

Repulsion Coefficient

$$F_{repulsive} = \frac{C_{repulsive}}{d^{Power}}$$

Distance between the boundaries of the 2 bodies

Repulsion Exponent

SIGGRAPH

Total Goal - Make the Free Body Move Towards its Final Position  
While Being Repelled by the Other Bodies

81

$m\ddot{x} + c\dot{x} + kx = \sum F$

SIGGRAPH

Functional Animation

82

avoid.mp4

SIGGRAPH

Increasing the Stiffness

83

Stiffness = 3, 6, 9

SIGGRAPH

Increasing the Repulsion Coefficient

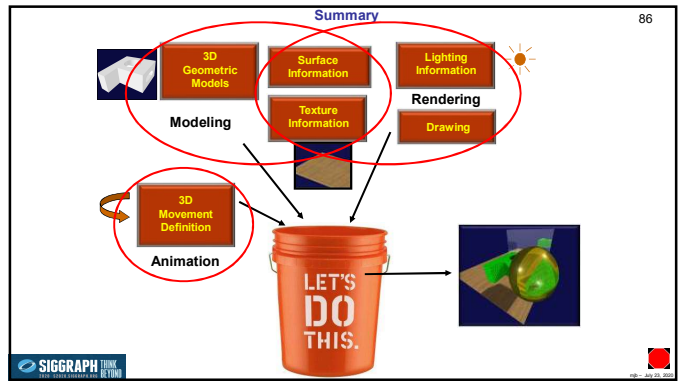
84

Repulse = 10, 30, 50

SIGGRAPH

Motion Capture ("MoCap") as an Input for Animation 85

SIGGRAPH THINK BEYOND



Conclusions ! 87

- 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!

SIGGRAPH THINK BEYOND

88

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

SIGGRAPH THINK BEYOND

Check out the More Information Document! 89

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>

Edward Angel and Dave Shreiner, *Interactive Computer Graphics: A Top-down Approach with OpenGL*, 6th Edition, Addison-Wesley, 2011.

Francis Hill and Stephen Kelley, *Computer Graphics Using OpenGL*, 3rd Edition, Prentice Hall, 2006.

Steve Cunningham, *Computer Graphics: Programming in OpenGL for Visual Communication*, Prentice-Hall, 2007.

Alan Watt, *3D Computer Graphics*, <http://cs.oregonstate.edu/~mjb/whirlwind>

Peter Shirley, *Fundamentals of Computer Graphics*, 2nd Edition, AK Peters, 2005.

Andrew Glassner, *Graphics Gems*, Academic Press, 1990.

SIGGRAPH THINK BEYOND

Mike Bailey  
Oregon State University  
mjb@cs.oregonstate.edu 90

Oregon State University  
Computer Graphics

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

SIGGRAPH THINK BEYOND