



**SIGGRAPH  
ASIA 2019  
BRISBANE**

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



**Oregon State  
University**  
Computer Graphics




# A Whirlwind Introduction to Computer Graphics



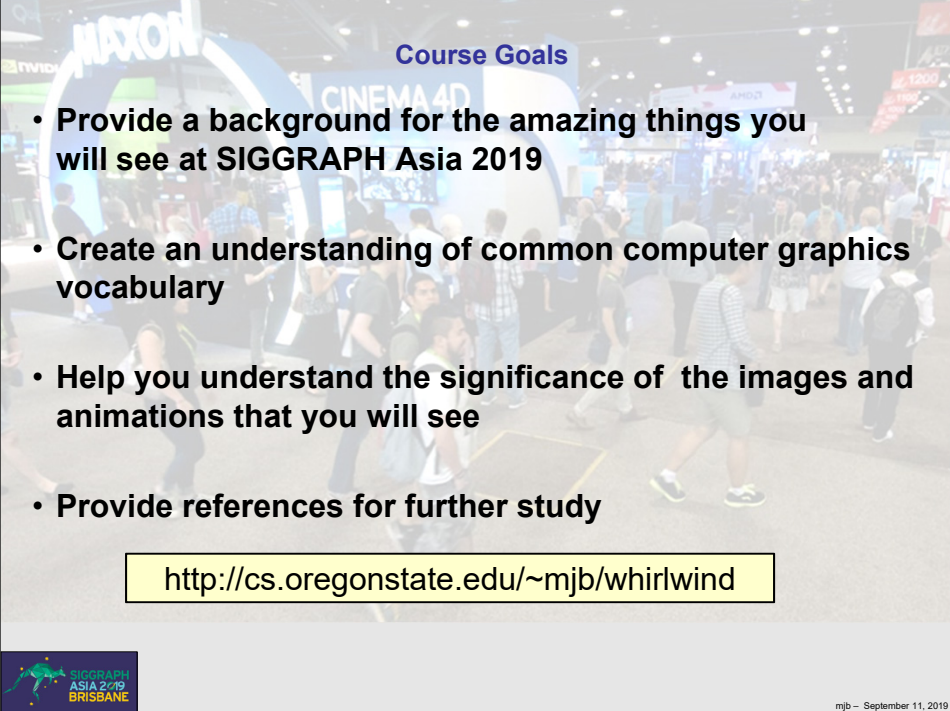
<http://cs.oregonstate.edu/~mjb/whirlwind>

mjb – September 11, 2019




**SIGGRAPH  
ASIA 2019  
BRISBANE**

**Course Goals**



- **Provide a background for the amazing things you will see at SIGGRAPH Asia 2019**
- **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>

mjb – September 11, 2019

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored.

For all other uses, contact the Owner/Author.

Copyright is held by the owner/author(s).

SA '19 Courses, November 17-20, 2019, Brisbane, QLD, Australia

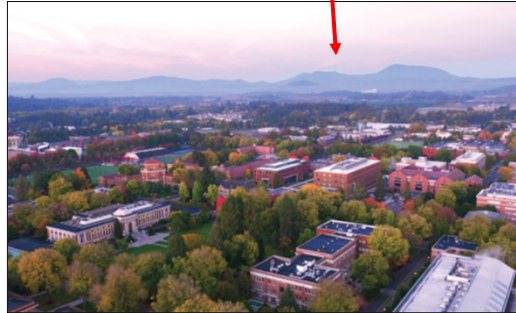
ACM 978-1-4503-6941-1/19/11.

10.1145/3355047.3359404

### Mike Bailey

- Professor of Computer Science, Oregon State University
- Has been in computer graphics for over 30 years
- Has had over 7,000 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>

mjb - September 11, 2019

## Schedule

**00:00** How the computer graphics pieces fit together

**00:15** Modeling

**00:35** Rendering

**00:55** Animation

**01:35** Questions and Discussion

**01:45** Finish

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

mjb - September 11, 2019

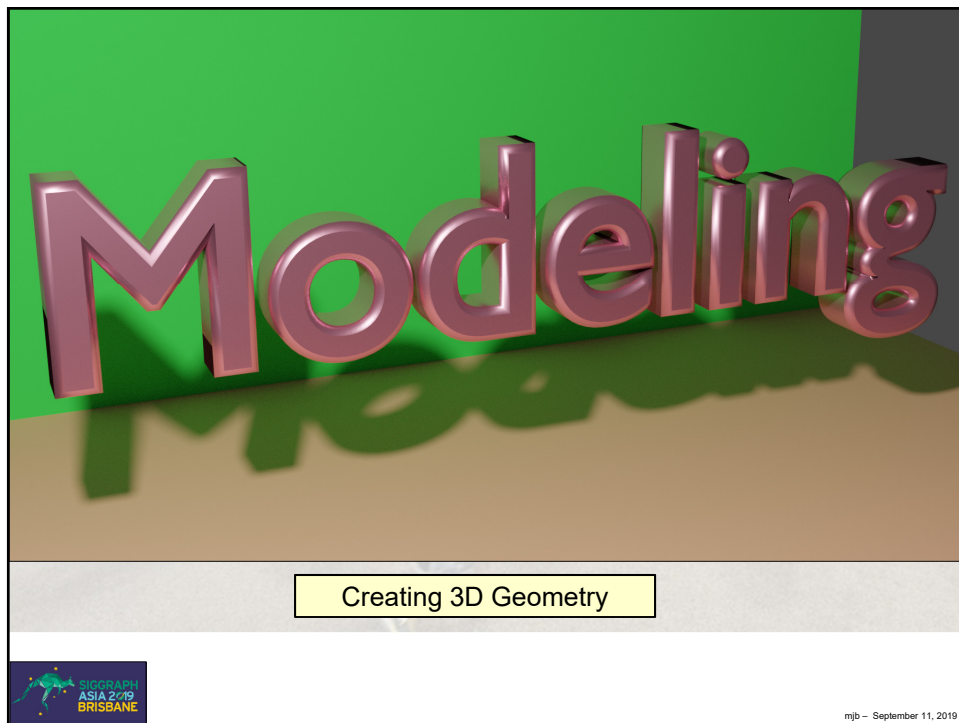
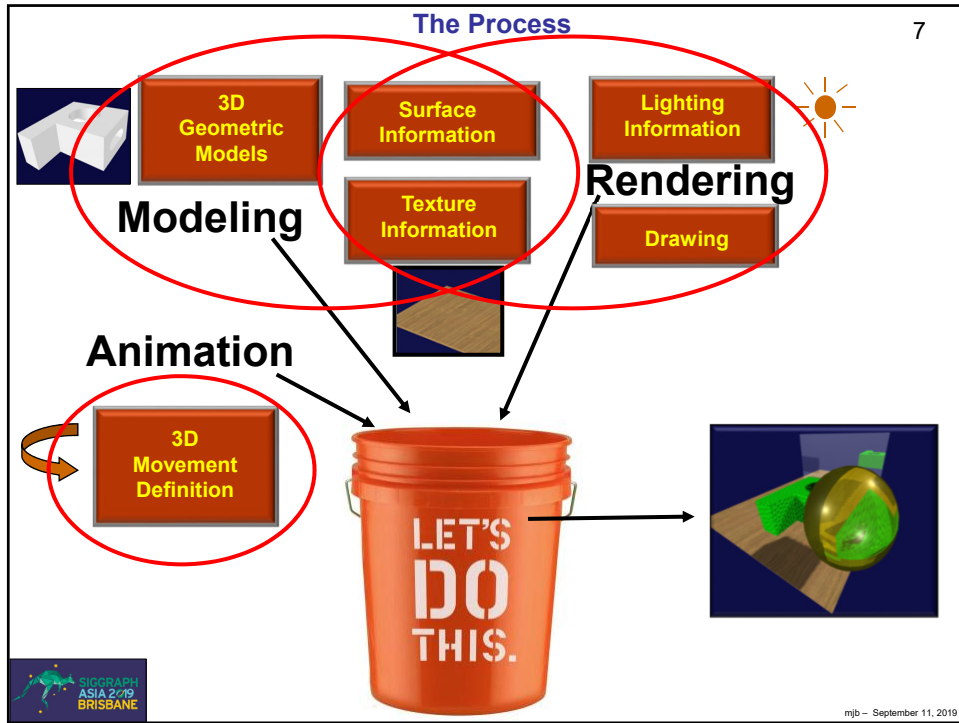
# The Graphics Process

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

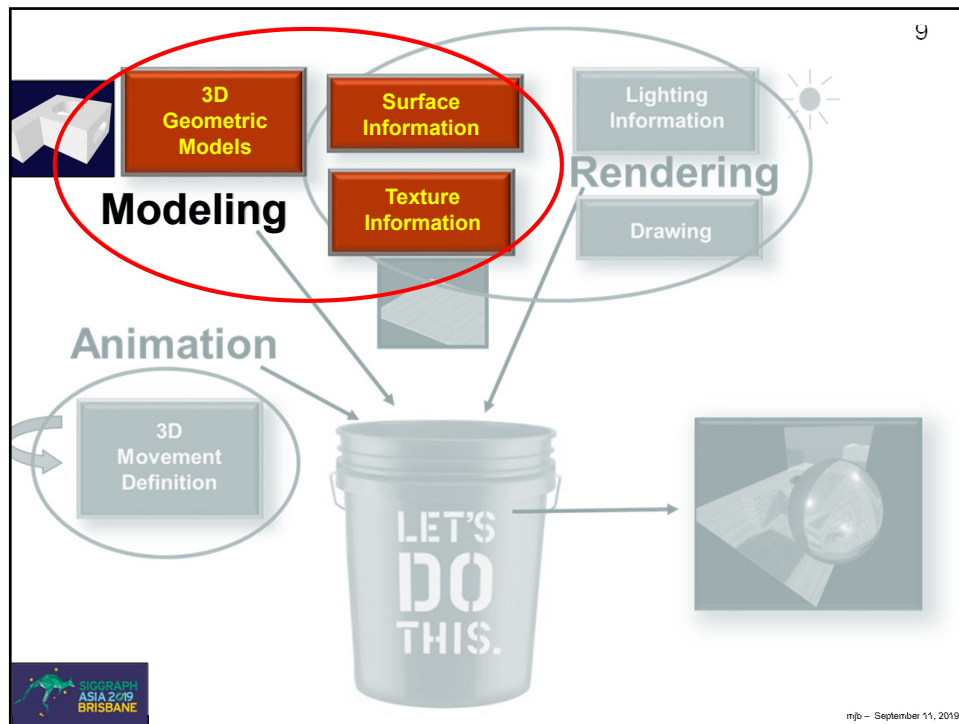


mjb - September 11, 2019









## What do we mean by “Modeling”?

10

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>

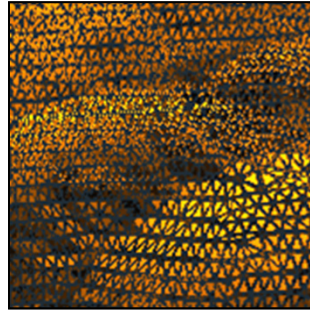
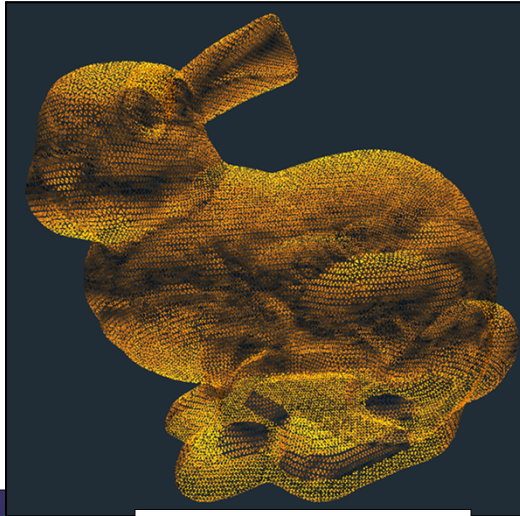
<http://cs.oregonstate.edu/~mjb/sketchup>

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

## Explicitly Listing Geometry and Topology

11

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



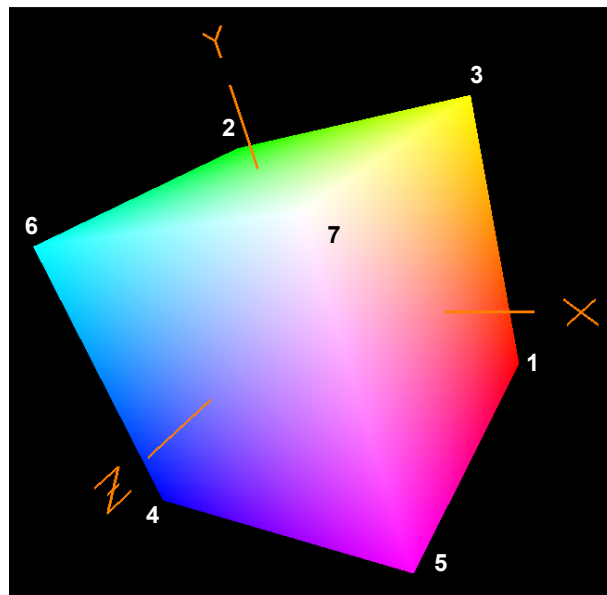
This is often called a **Mesh**.

<http://graphics.stanford.edu/data/3Dscanrep>

mjb - September 11, 2019

## Cube Mesh Example

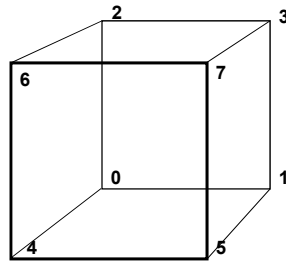
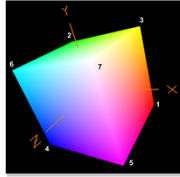
12



mjb - September 11, 2019

## Explicitly Listing Geometry and Topology -- Quadrilaterals

13



```
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. }
};
```

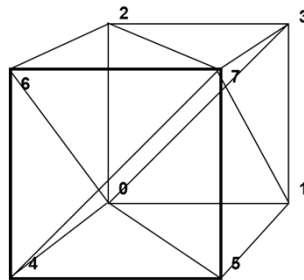
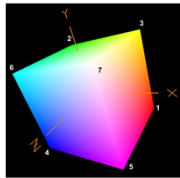
```
static GLuint CubeIndices[][4] =
{
    { 0, 2, 3, 1 },
    { 4, 5, 7, 6 },
    { 1, 3, 7, 5 },
    { 0, 4, 6, 2 },
    { 2, 6, 7, 3 },
    { 0, 1, 5, 4 }
};
```



mjb - September 11, 2019

## More likely we would use Triangles

14



```
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 }
};
```

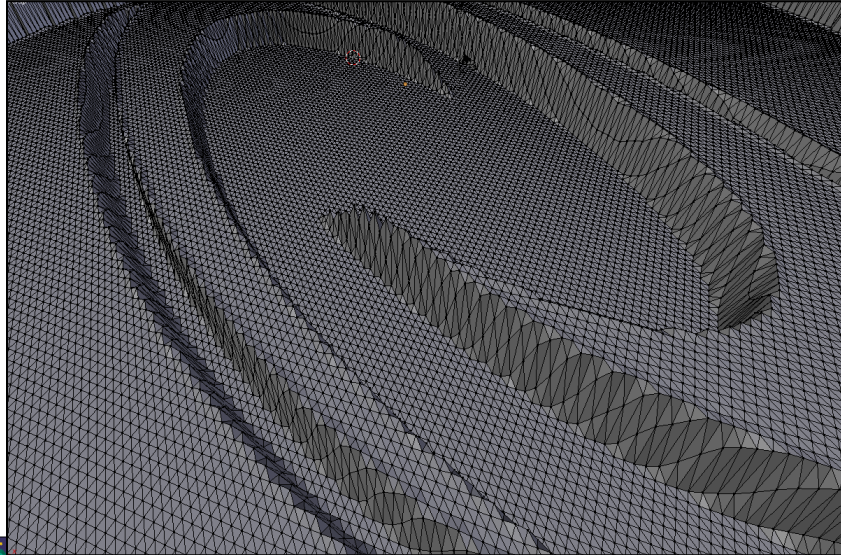
```
GLuint TriangleCubeIndices[][3] =
{
    { 0, 2, 3 },
    { 0, 3, 1 },
    { 4, 5, 7 },
    { 4, 7, 6 },
    { 1, 3, 7 },
    { 1, 7, 5 },
    { 0, 4, 6 },
    { 0, 6, 2 },
    { 2, 6, 7 },
    { 2, 7, 3 },
    { 0, 1, 5 },
    { 0, 5, 4 }
};
```



mjb - September 11, 2019

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

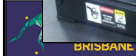
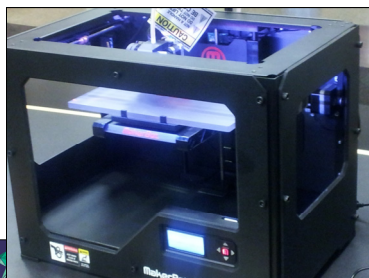
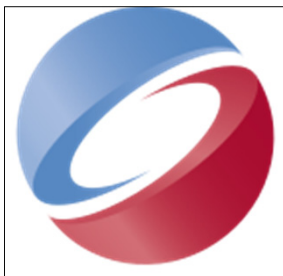
15



mjb - September 11, 2019

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

16

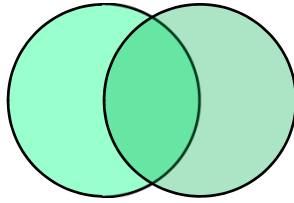


mjb - September 11, 2019

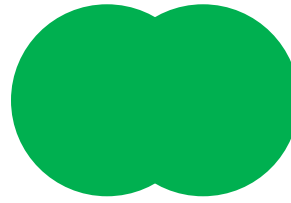


Another way to Model:  
Remember Venn Diagrams (2D Boolean Operators) from High School?

17



Two Overlapping Shapes



Union



Intersection



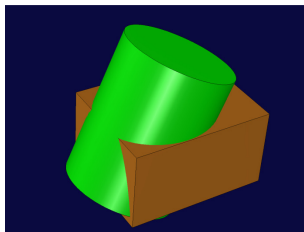
Difference



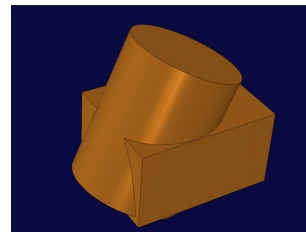
mjb - September 11, 2019

Solid Modeling Using 3D Boolean Operators

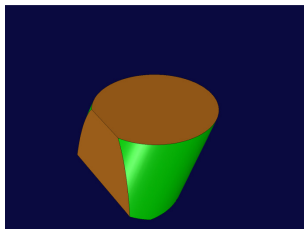
18



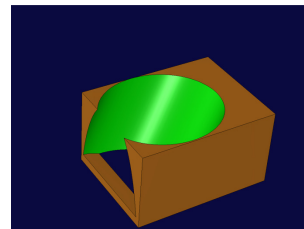
Two Overlapping Solids



Union



Intersection



Difference

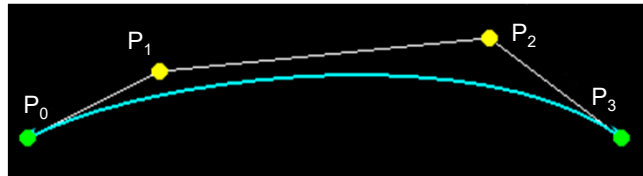


This is often called **Constructive Solid Geometry**, or **CSG**

mjb - September 11, 2019

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

19



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

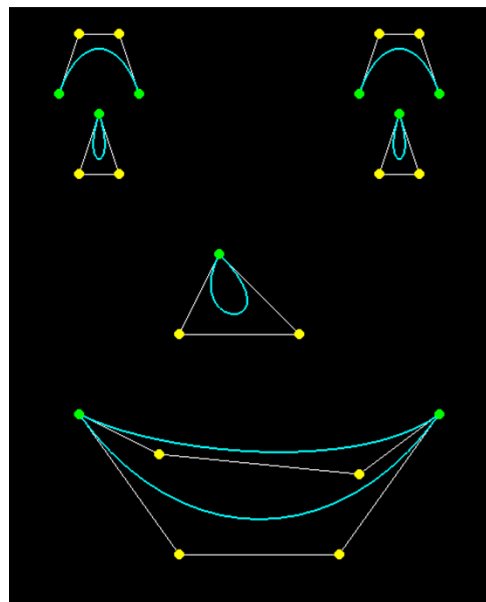
$$0 \leq t \leq 1.$$



mjb – September 11, 2019

## Curve Sculpting – Bézier Curve Sculpting Example

20



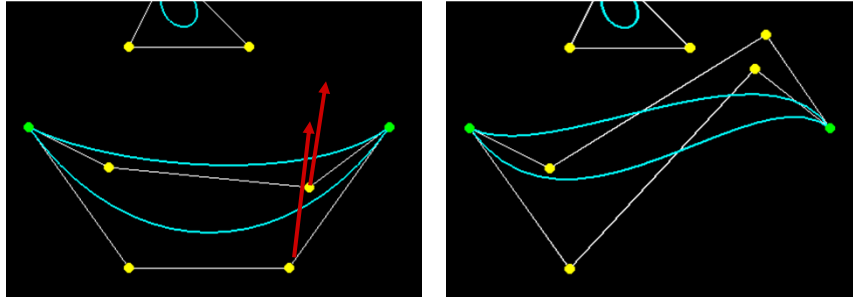
curves.mp4



mjb – September 11, 2019

## Curve Sculpting – Bézier Curve Sculpting Example

21



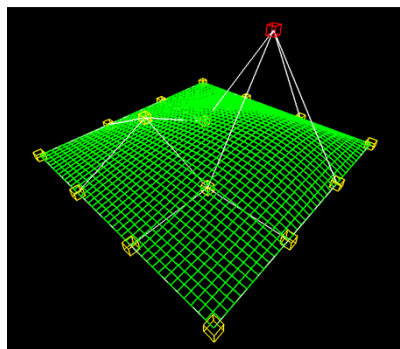
A *Small* Amount of Input Change Results in a *Large* Amount of Output Change



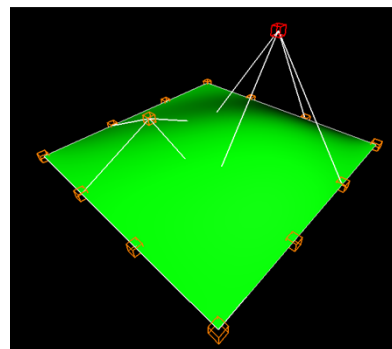
mjb – September 11, 2019

## Another way to Model: Bézier Surface Sculpting

22



Wireframe



Surface

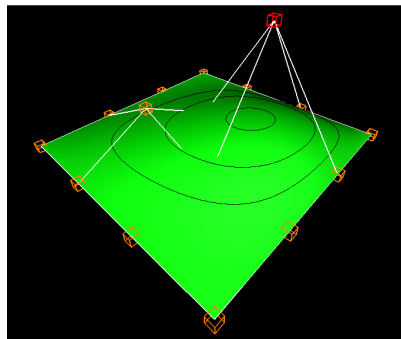
A *Small* Amount of Input Change Results in a *Large* Amount of Output Change



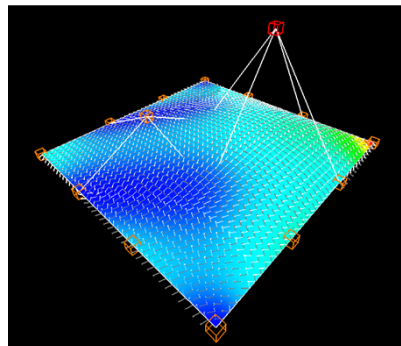
mjb – September 11, 2019

## Surface Equations can also be used for Analysis

23



Showing Contour Lines



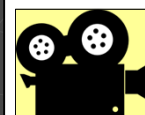
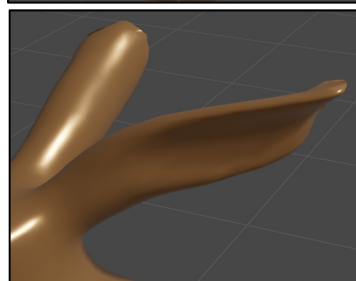
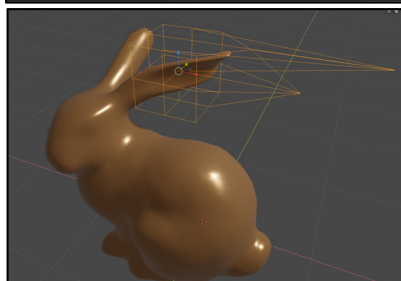
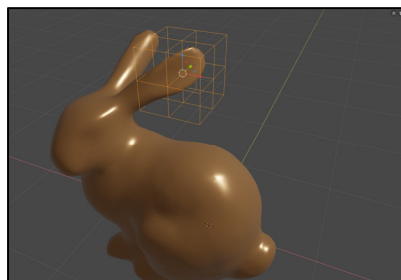
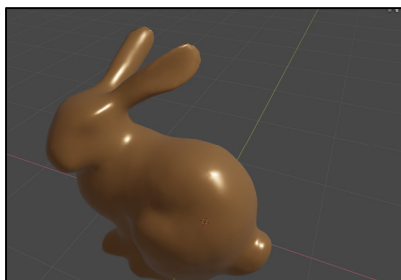
Showing Curvature



mjb - September 11, 2019

## Another Way to Model: Volume Sculpting

24



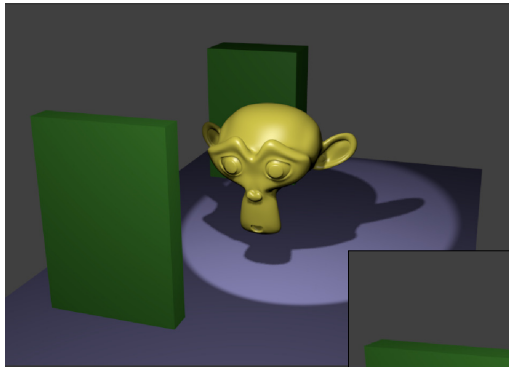
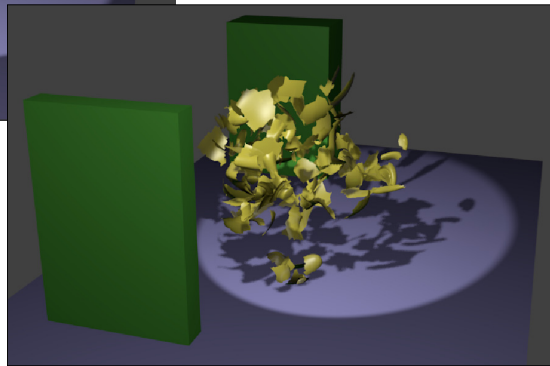
lattice.mp4

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



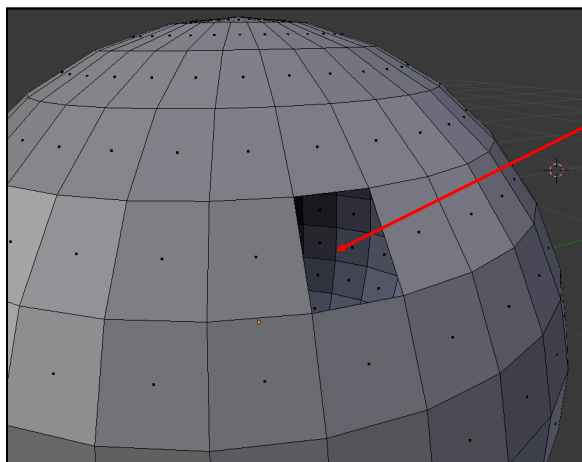
mjb - September 11, 2019



Blender's *Explosion* feature

mjb - September 11, 2019

**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



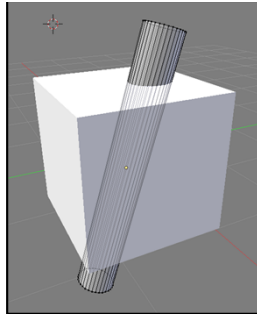
mjb - September 11, 2019

## Object Modeling Rules for 3D Printing

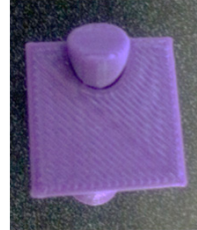
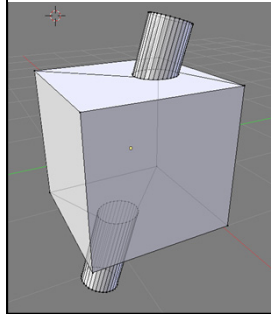
27

**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**



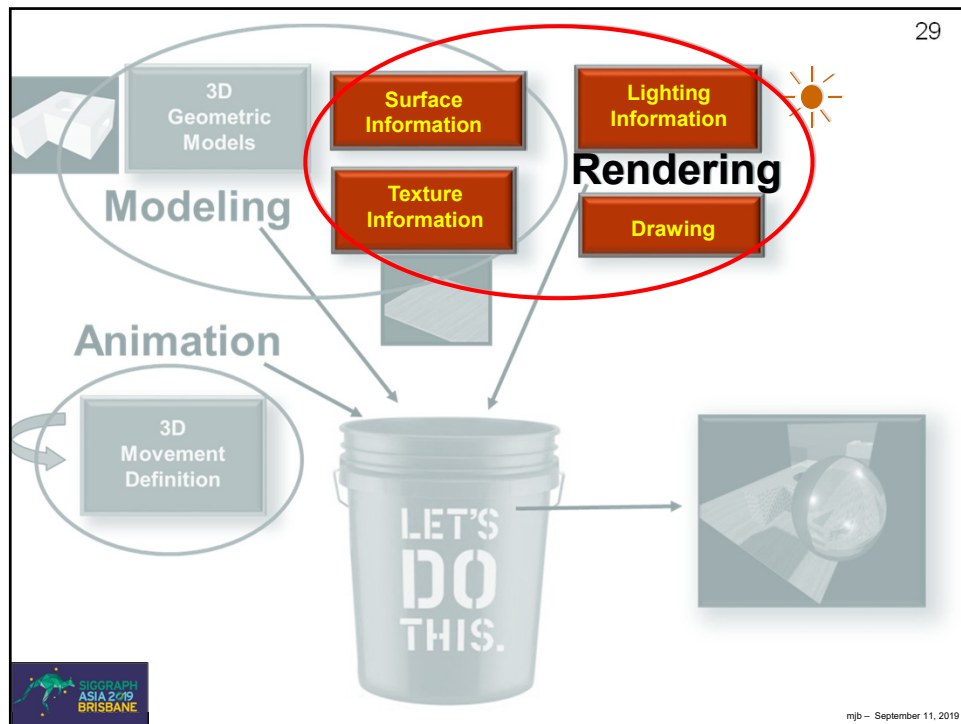
mjb - September 11, 2019



Creating an image



mjb - September 11, 2019



30

## 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?

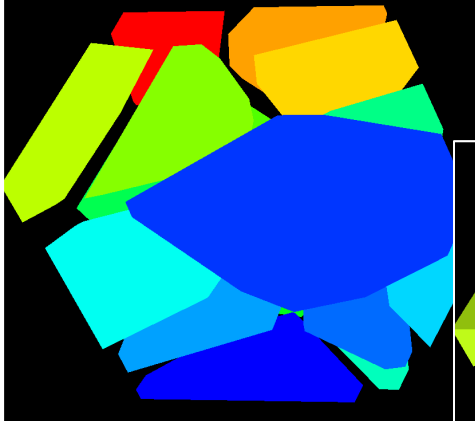
Want to experiment with a free rendering program?  
Want some notes on how to get started?  
<http://cs.oregonstate.edu/~mjb/blender>

SIGGRAPH ASIA 2019 BRISBANE

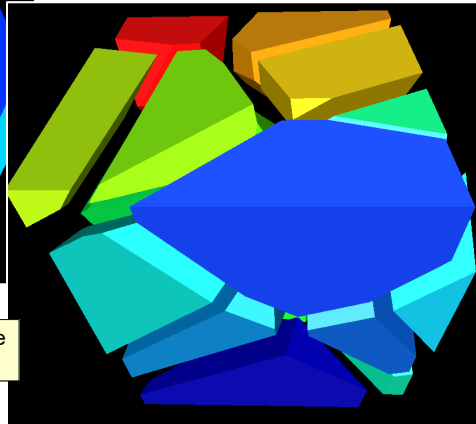
mjb - September 11, 2019

## Why Do We Care About Lighting?

No lighting



Lighting

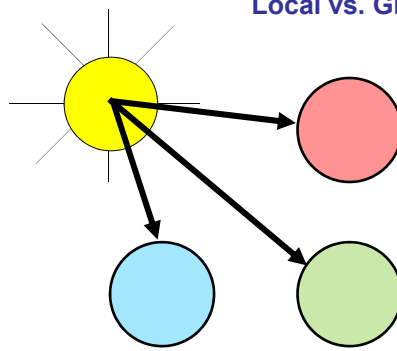


Lighting makes it possible to tell the difference between surfaces or parts of surfaces

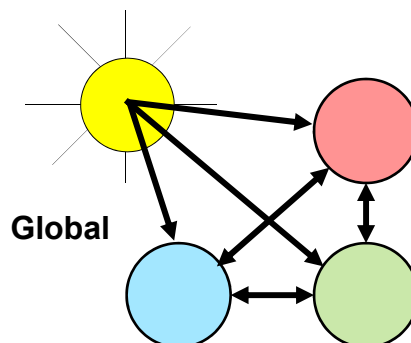


mjb - September 11, 2019

## Local vs. Global Illumination



Local



Global

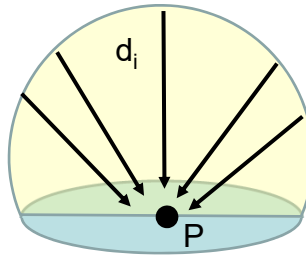


mjb - September 11, 2019



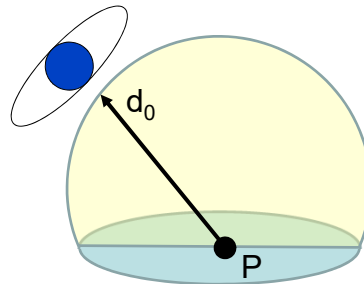
## The Rendering Equation

33



Light arriving at Point P  
from everywhere

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



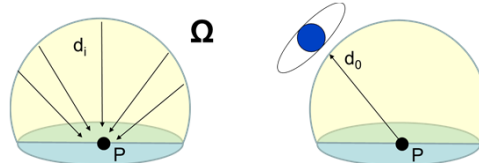
Light departing from Point P  
in the direction that we are  
viewing the scene from



mjb - September 11, 2019

## The Rendering Equation

34



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

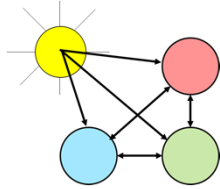
In plain language, this is a light balance equation:

**“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.”**



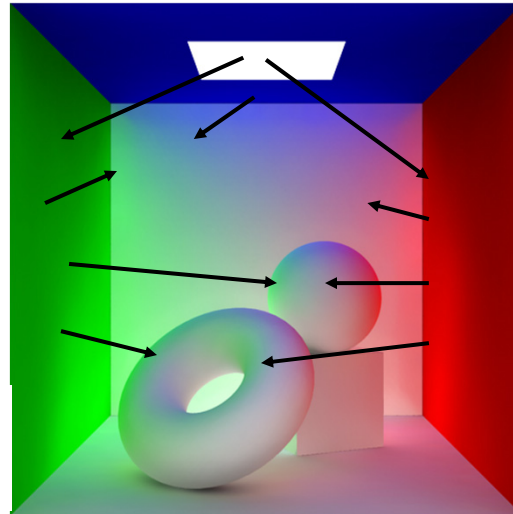
mjb - September 11, 2019

## 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/>

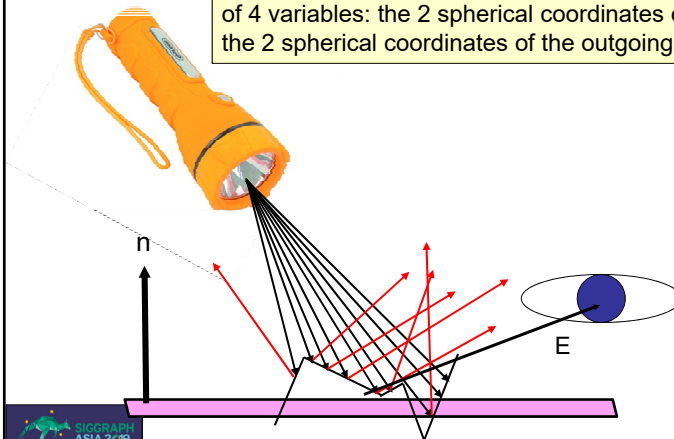


mjb - September 11, 2019

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

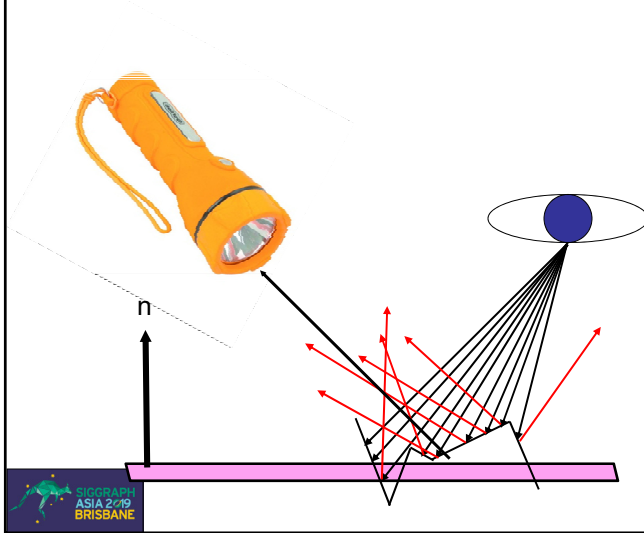
This distribution of bounced light rays is called the **Bidirectional Reflectance Distribution Function**, or **BRDF**.

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.



mjb - September 11, 2019

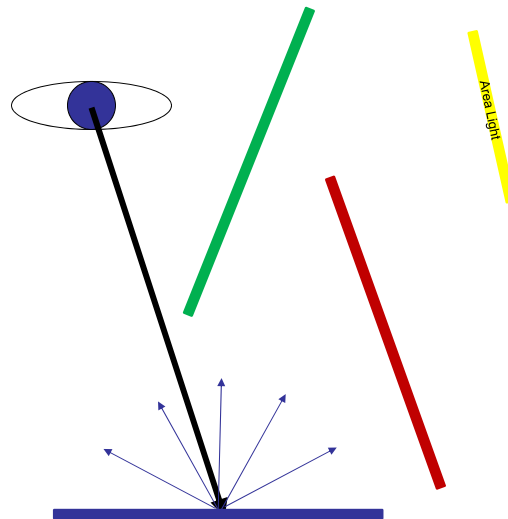
Usually it is easier to trace from the eye



mjb - September 11, 2019

### Physically-Based Rendering

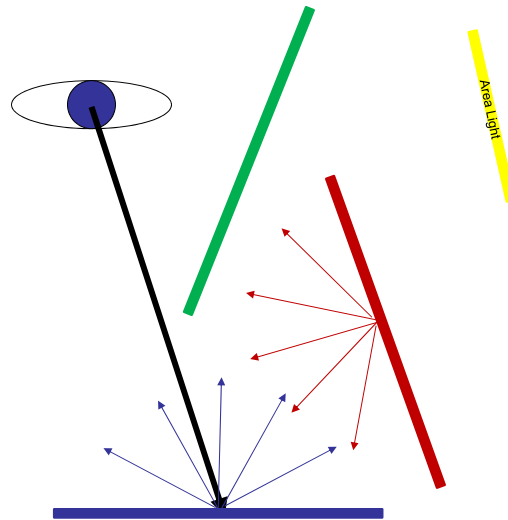
Let light can bounce around the scene, depending on how the different materials behave.



mjb - September 11, 2019

## Physically-Based Rendering

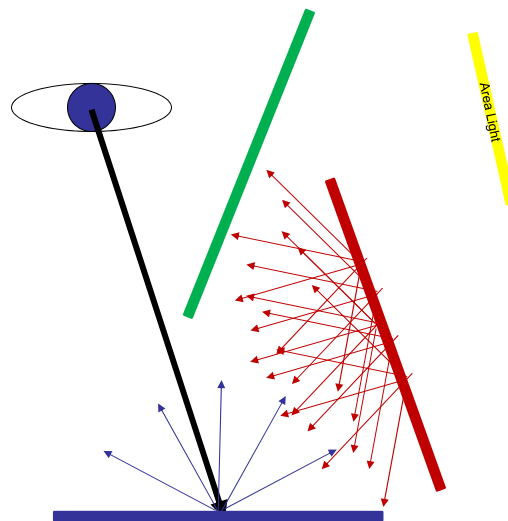
39



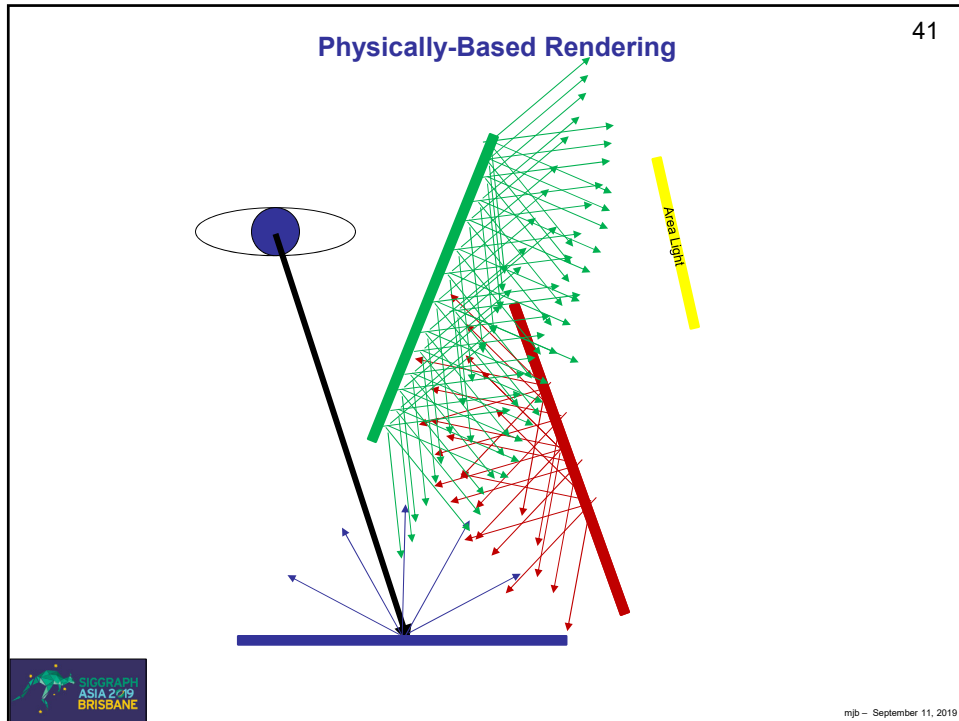
mjb - September 11, 2019

## Physically-Based Rendering

40



mjb - September 11, 2019



42

### 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!

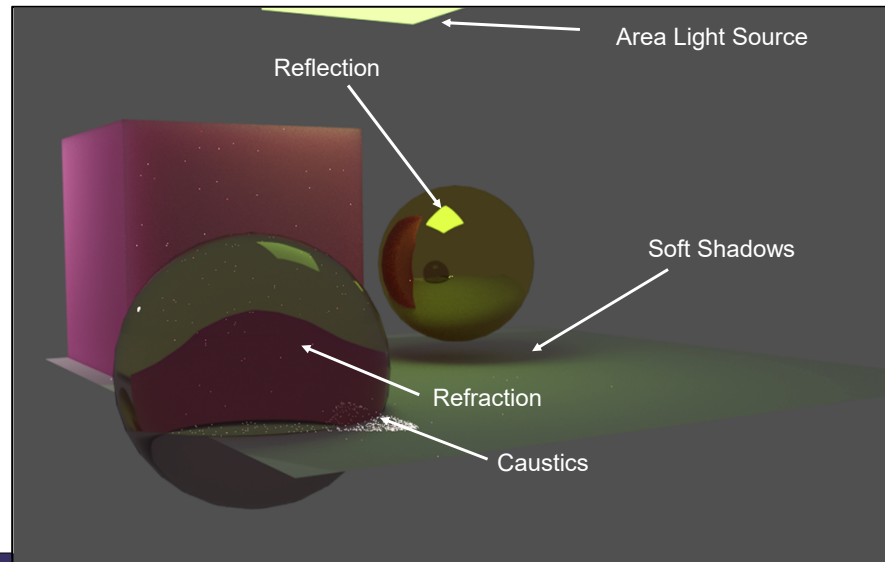
$$LightGathered = \frac{\sum_{i=0}^{N-1} ResultOfRaysCastInRandomDirection}{N}$$

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

mjb - September 11, 2019

## Physically-Based Rendering using the Blender Cycles Renderer

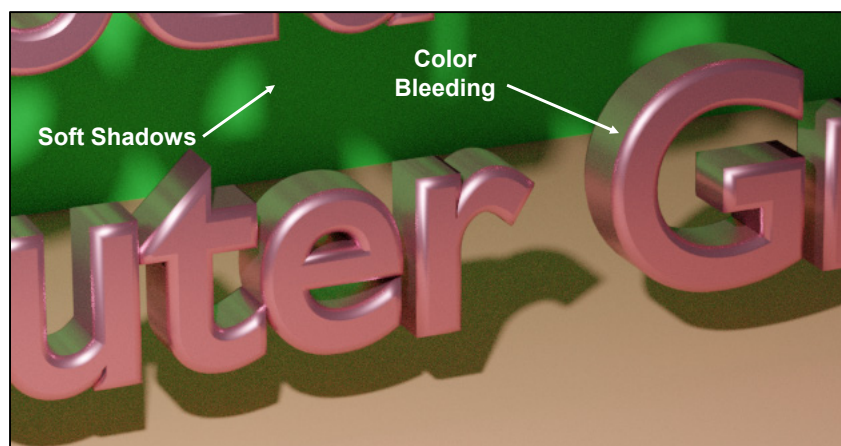
43



mjb - September 11, 2019

## An Example from the Title Slide

44



mjb - September 11, 2019

## Another Physically-Based Rendering Example

45

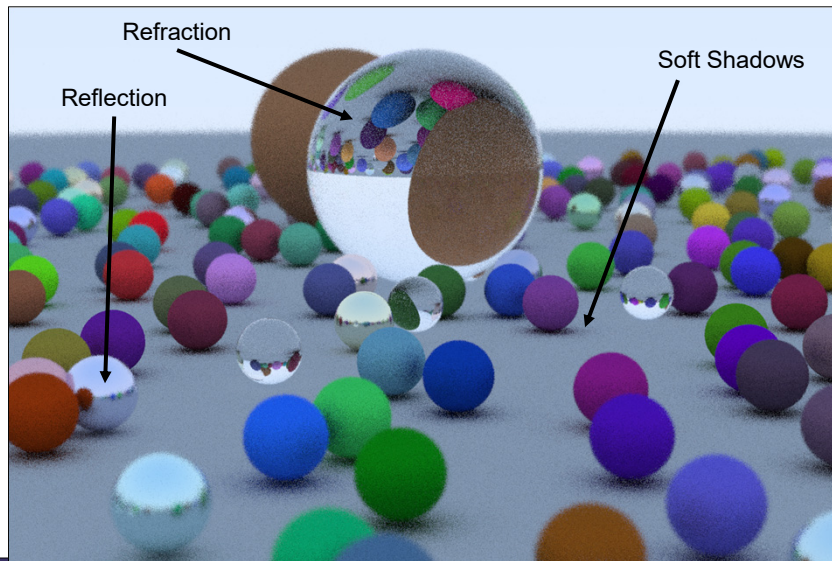
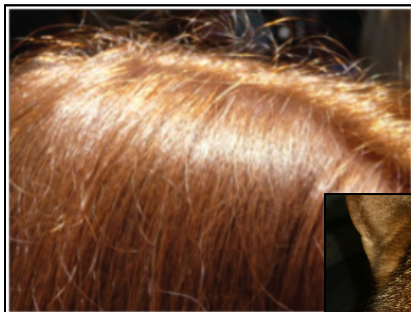


Image by Josiah Blaisdell

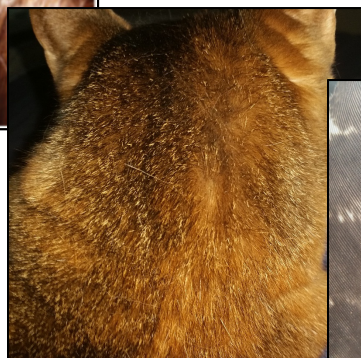
mjb - September 11, 2019

## Tricky Lighting Situations

46



Hair



Fur



Feathers

Watch for these at the  
conference and in CG movies!

mjb - September 11, 2019

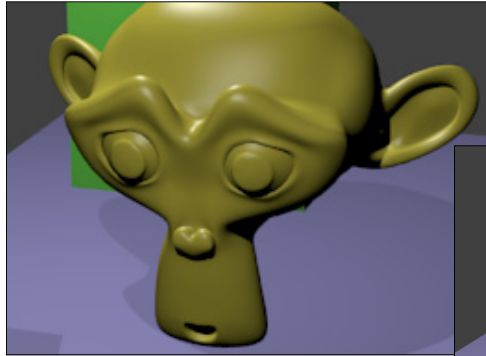


## Subsurface Scattering

47

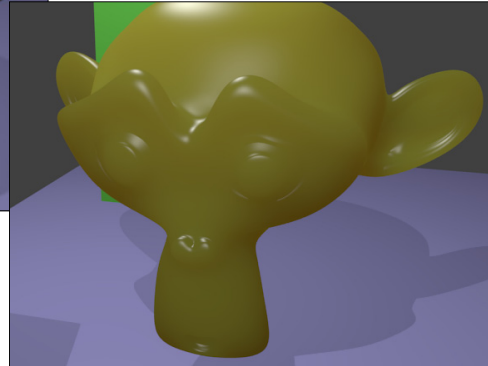
SS models light bouncing around *within* an object before coming back out.

This is a good way to represent skin, wax, milk, paraffin, etc.



Original rendering

With Subsurface Scattering



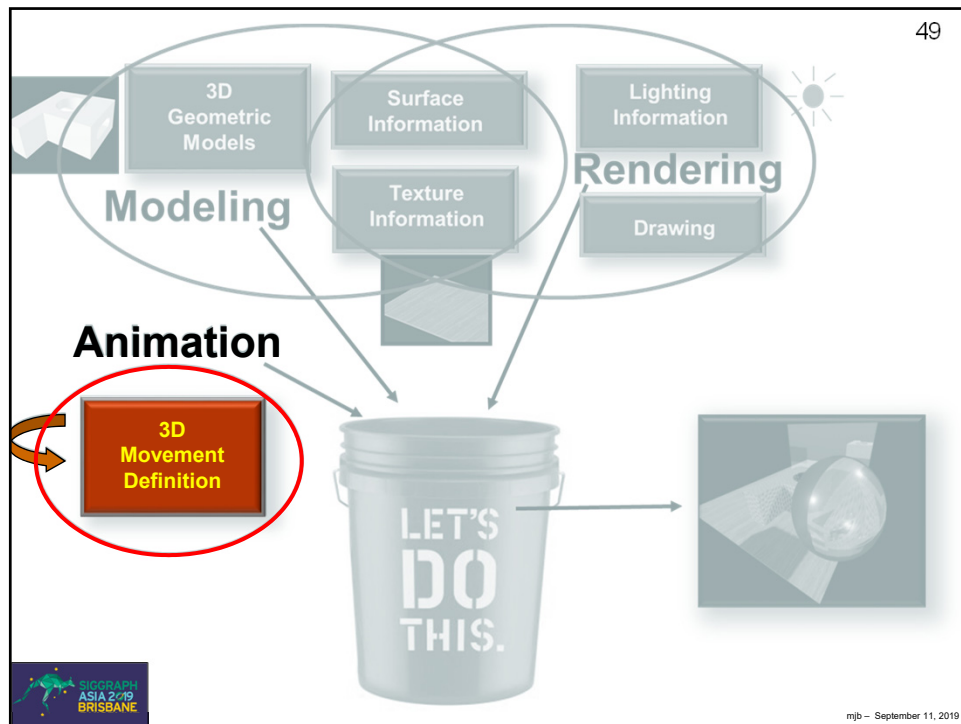
mjb - September 11, 2019



Creating the motion you want



mjb - September 11, 2019



50

## 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?

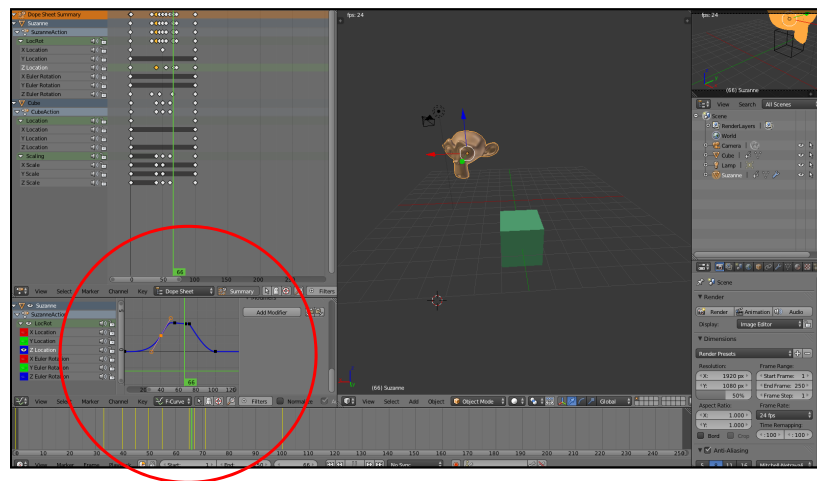
Want to experiment with a free animation program?  
 Want some notes on how to get started?  
<http://cs.oregonstate.edu/~mjb/blender>

**SIGGRAPH ASIA 2019 BRISBANE**

mjb - September 11, 2019

## Keyframe Animation

51



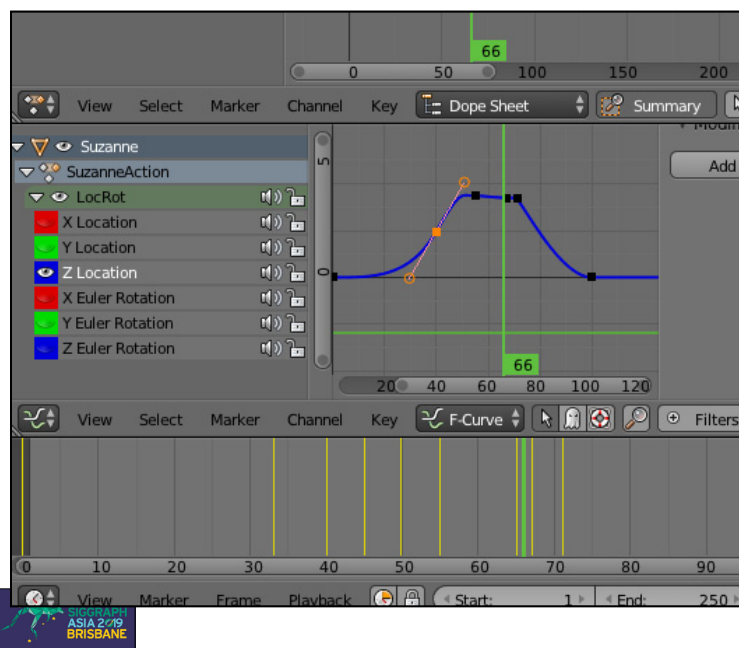
Forcing the geometry to smoothly pass through key positions



mjb - September 11, 2019

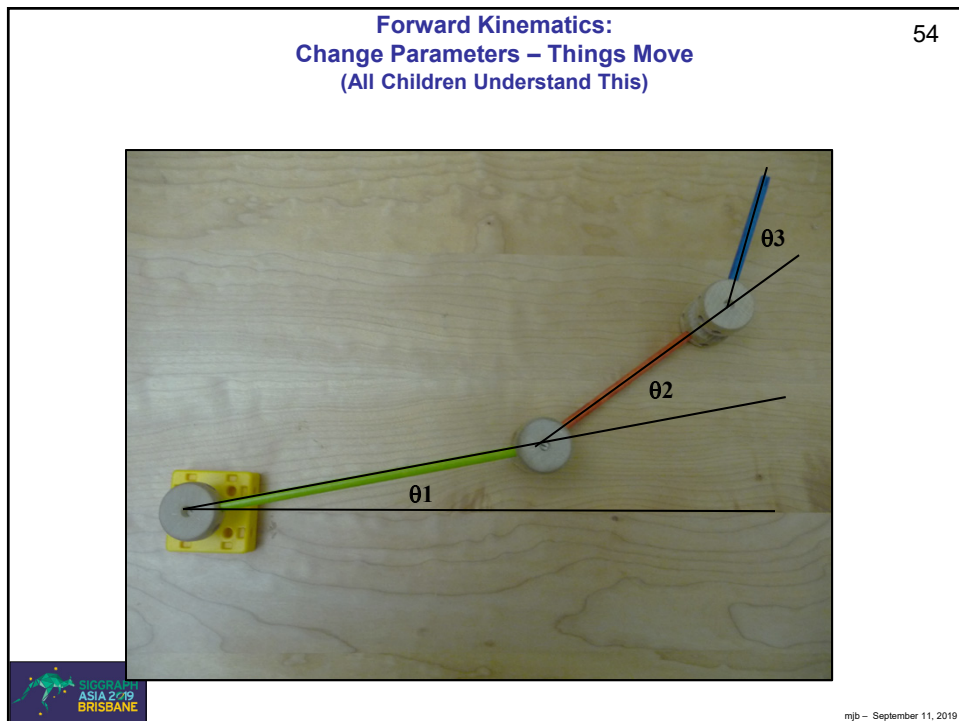
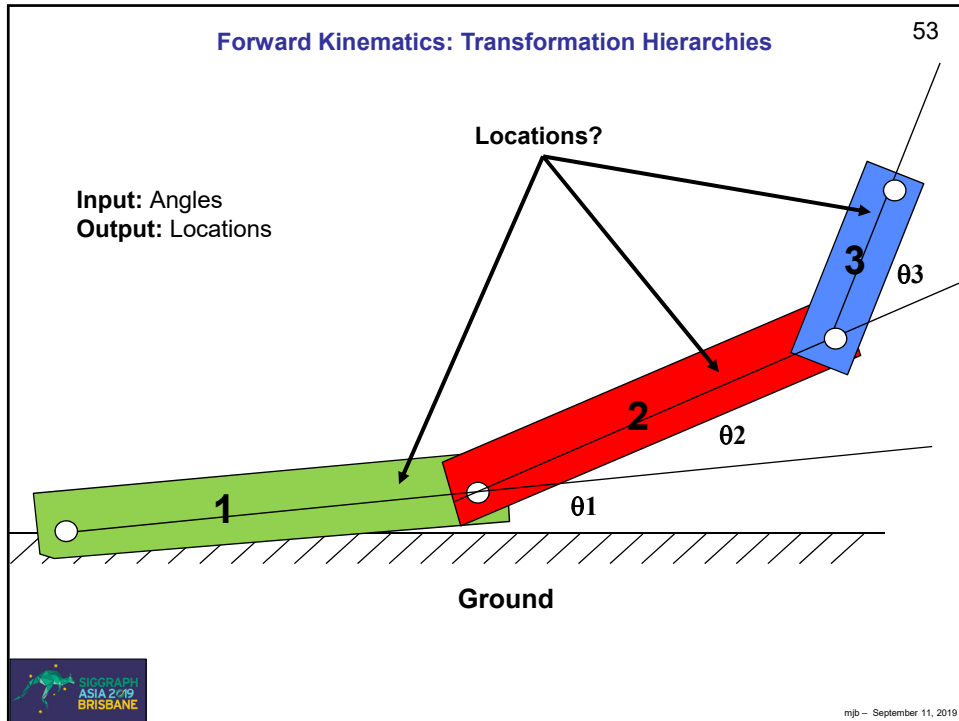
## Keyframe Animation

52



anim2.mp4

mjb - September 11, 2019



## Inverse Kinematics

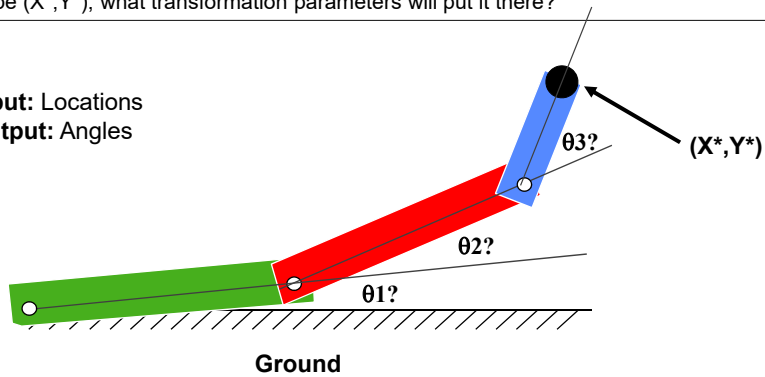
55

**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

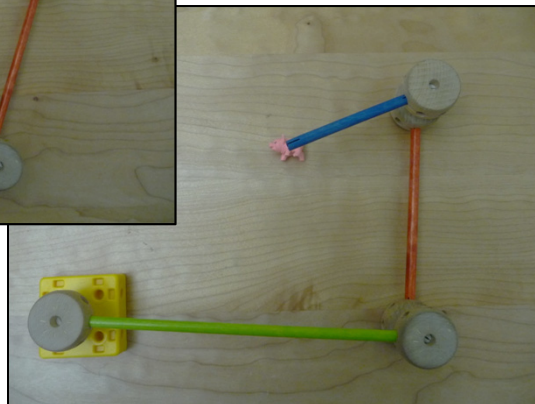
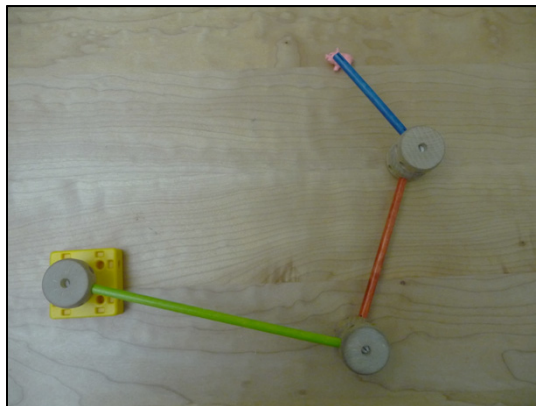
**Output:** Angles



mjb - September 11, 2019

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

56

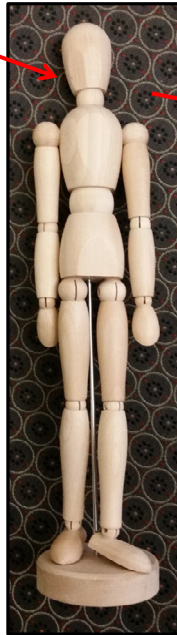


mjb - September 11, 2019

## Animating a Human-ish Form

57

Start with this ...



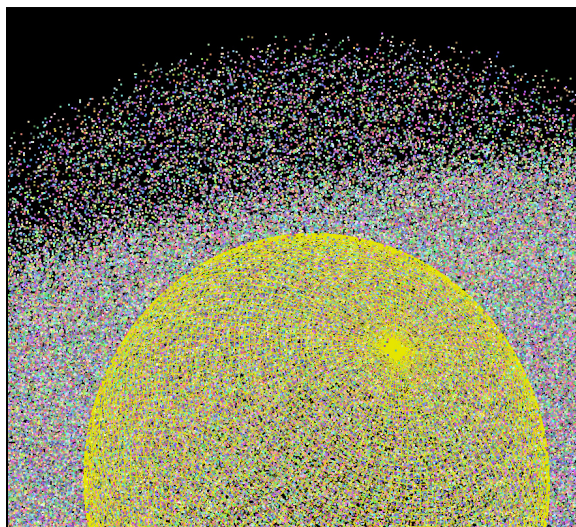
... and turn it into a kinematic model:



mjb - September 11, 2019

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

58



gpu\_particles.mp4

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



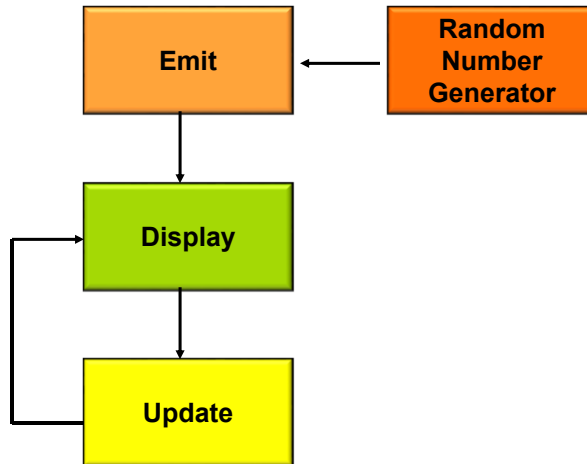
mjb - September 11, 2019



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

59

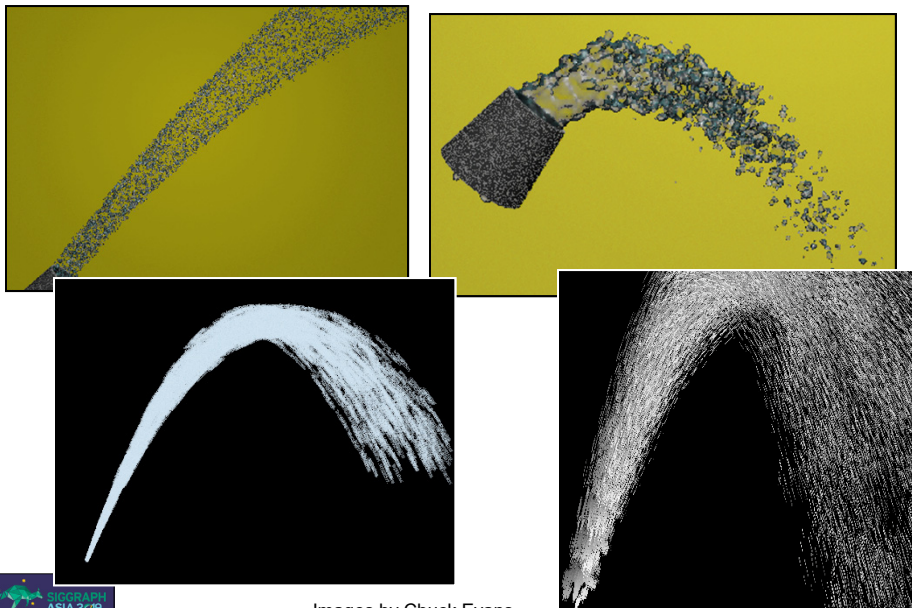
The basic process is:



mjb - September 11, 2019

## Particle Systems Examples

60



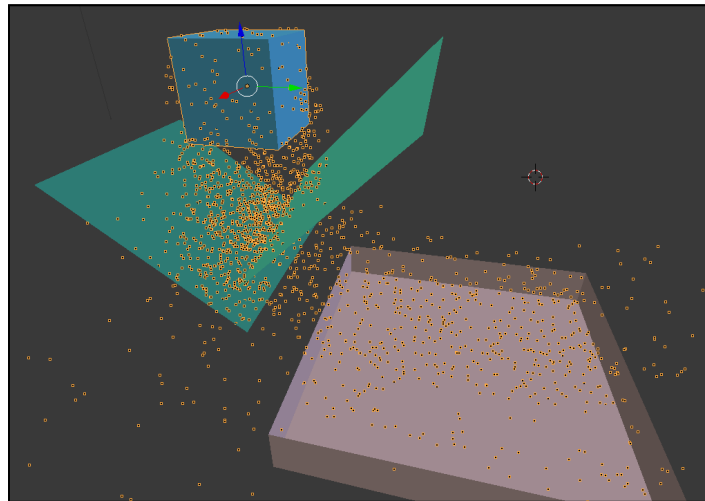
Images by Chuck Evans

mjb - September 11, 2019



## Particle Systems Examples

61



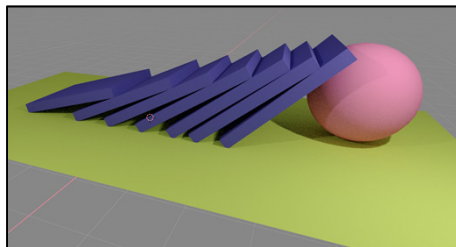
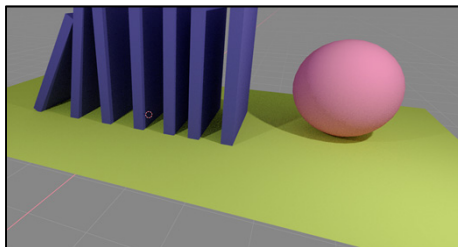
particles.mp4



mjb - September 11, 2019

## Animating using Rigid-body Physics

62



Newton's first law:

$$\text{force} = \text{mass} * \text{acceleration}$$

or

$$\text{acceleration} = \text{force} / \text{mass}$$

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.



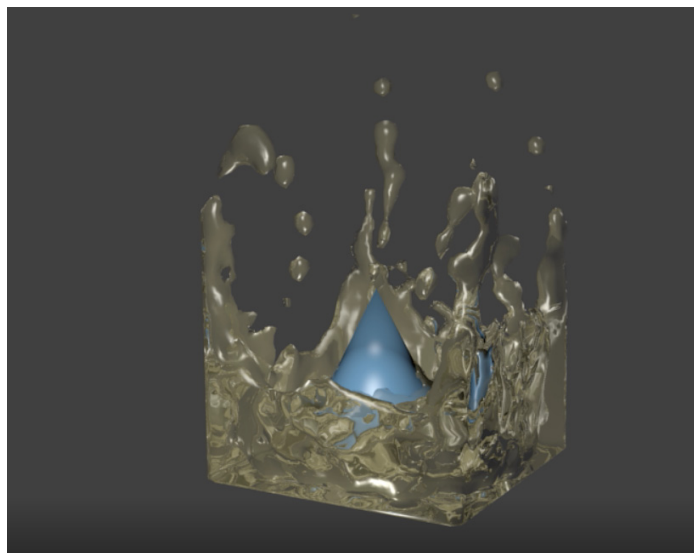
dominos2.mp4



mjb - September 11, 2019

## Animating using Fluid Physics

63



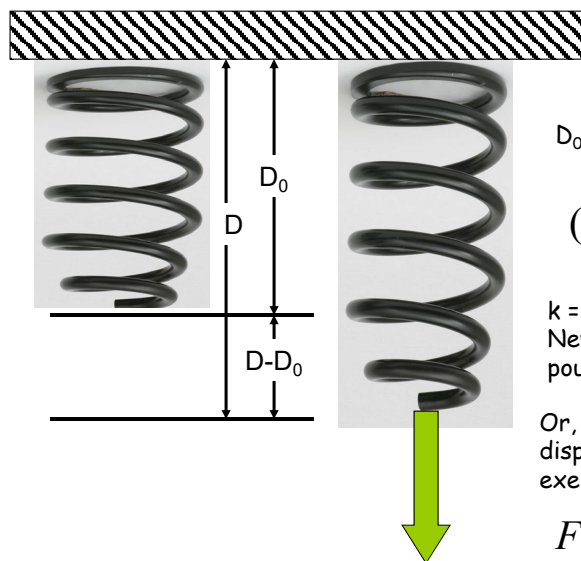
fluid.avi



mjb - September 11, 2019

## Animating using Spring Physics

64



$D_0$  = 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:

$$F = k(D - D_0)$$

Force = F

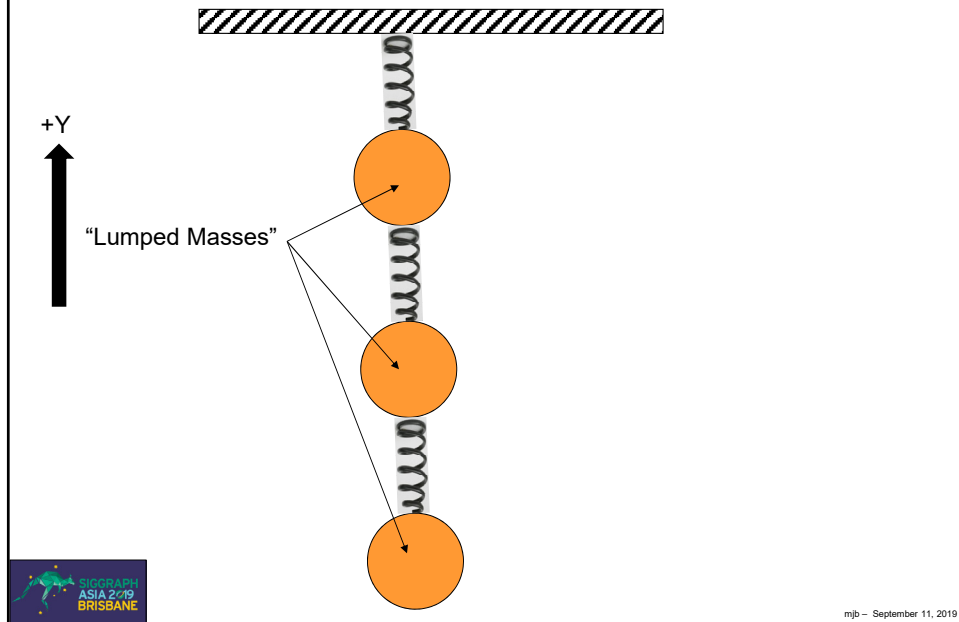
This is known as **Hooke's Law**



mjb - September 11, 2019

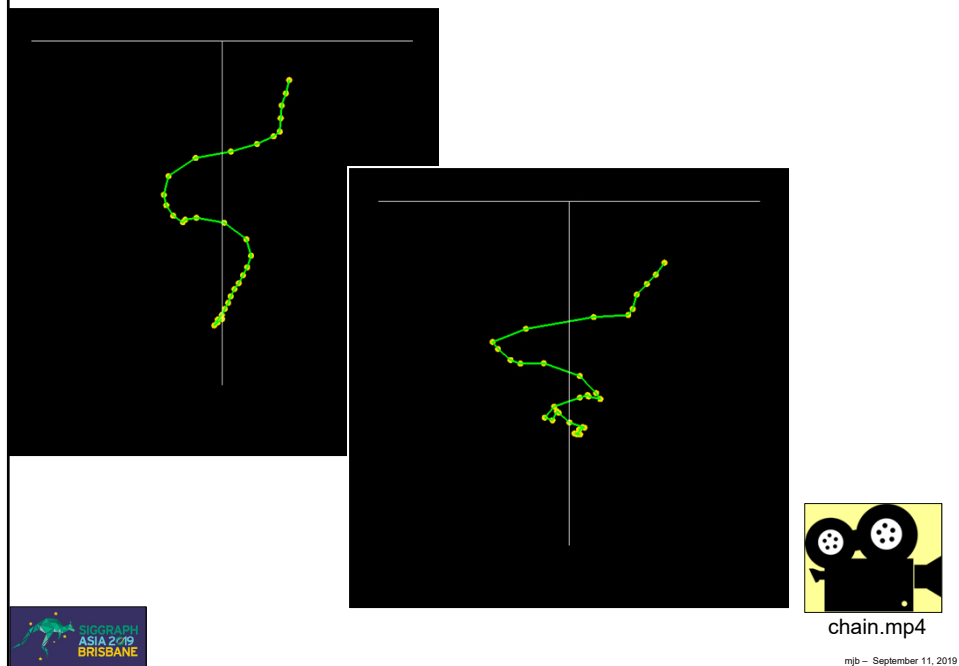
## Animating using the Physics of a Mesh of Springs

65



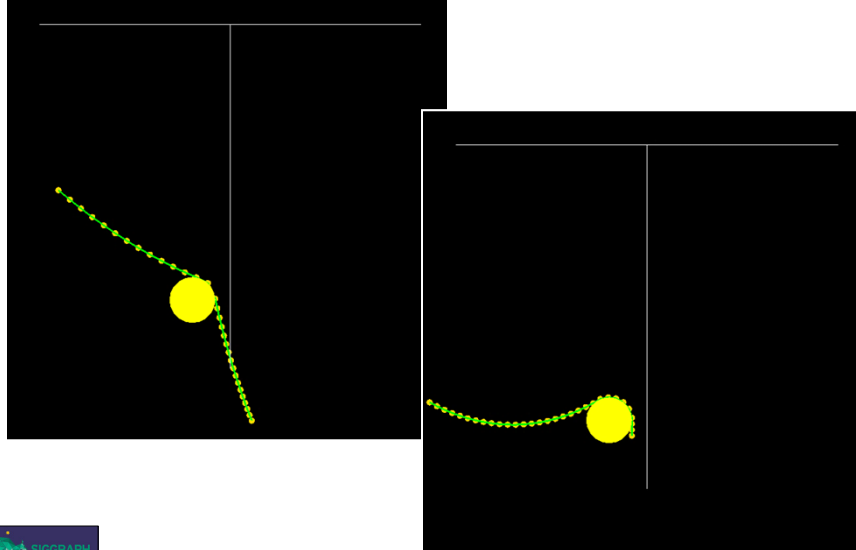
## Simulating a Bouncy Chain

66



## Placing a Physical Barrier in the Scene

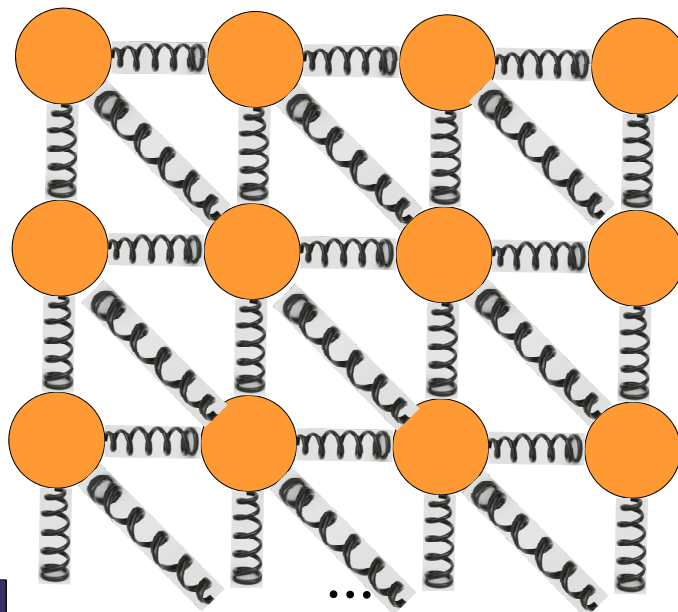
67



mjb - September 11, 2019

## Animating Cloth

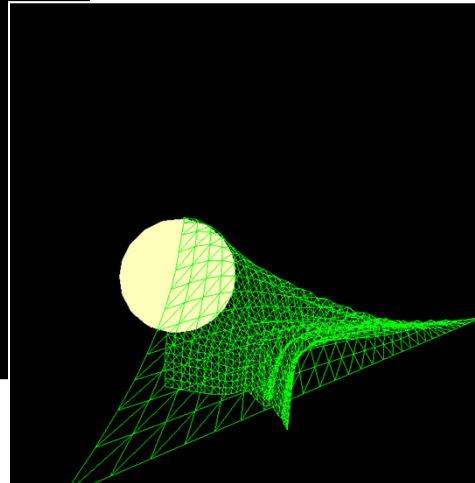
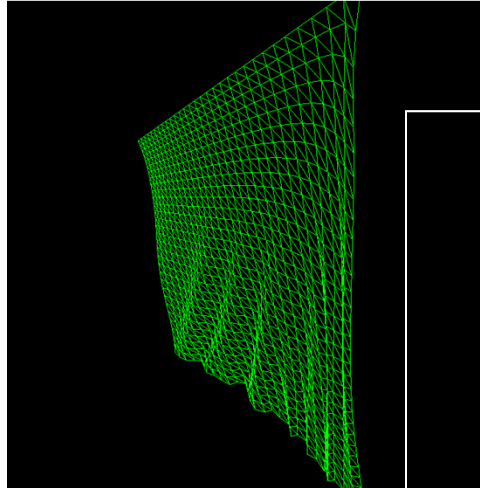
68



mjb - September 11, 2019

## Cloth Examples

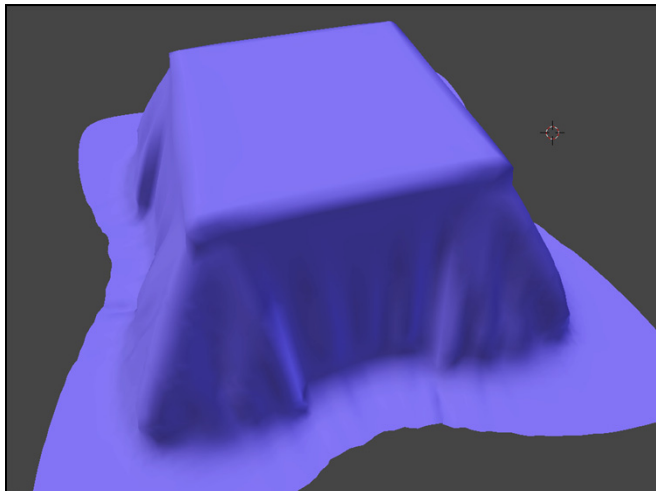
69



mjb - September 11, 2019

## Cloth Example

70



cloth.mp4



mjb - September 11, 2019

## Functional Animation – “Fake Physics”

71



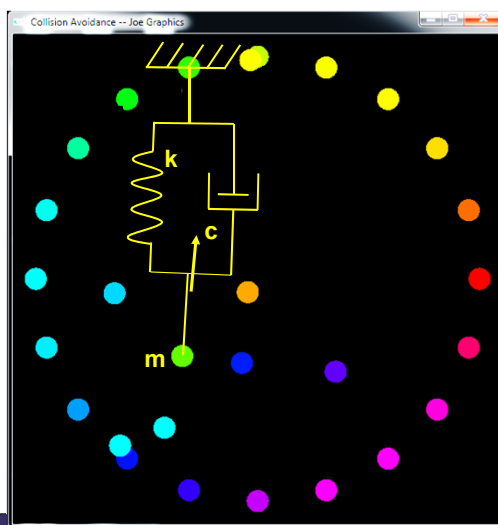
The Challenge: animate a collection of objects, each trying to move to a target, but without colliding with each other



mjb – September 11, 2019

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

72



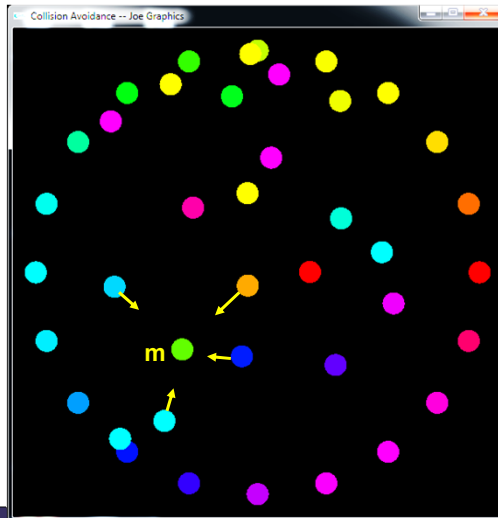
$$m\ddot{x} + c\dot{x} + kx = 0$$



mjb – September 11, 2019

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

73



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

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

Repulsion Coefficient

Distance between the boundaries of the 2 bodies

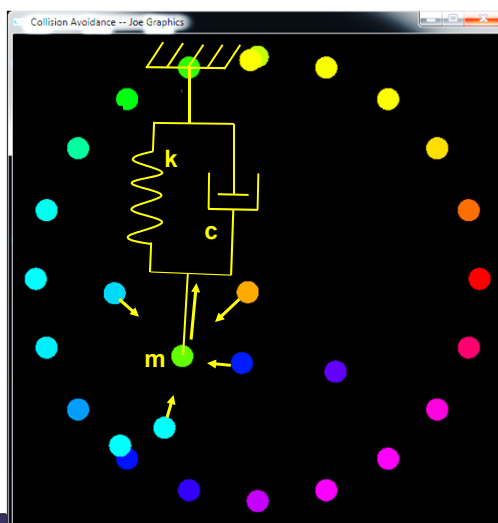
Repulsion Exponent



mjb - September 11, 2019

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

74



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

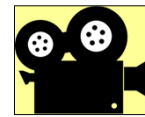
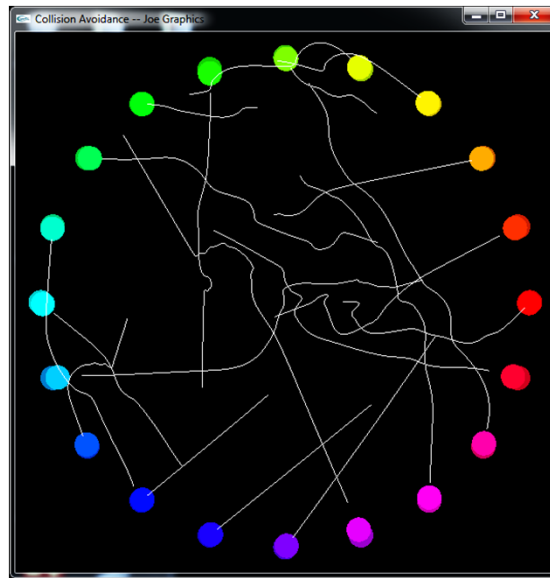


mjb - September 11, 2019



## Functional Animation

75



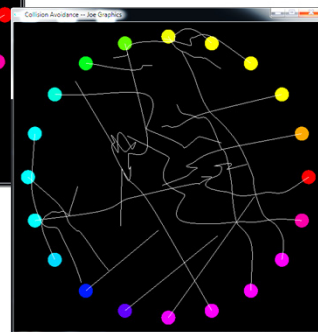
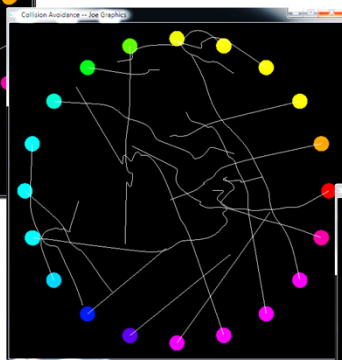
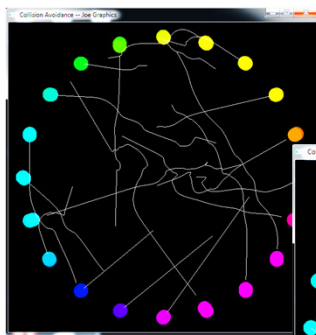
avoid.mp4



mjb - September 11, 2019

## Increasing the Stiffness

76



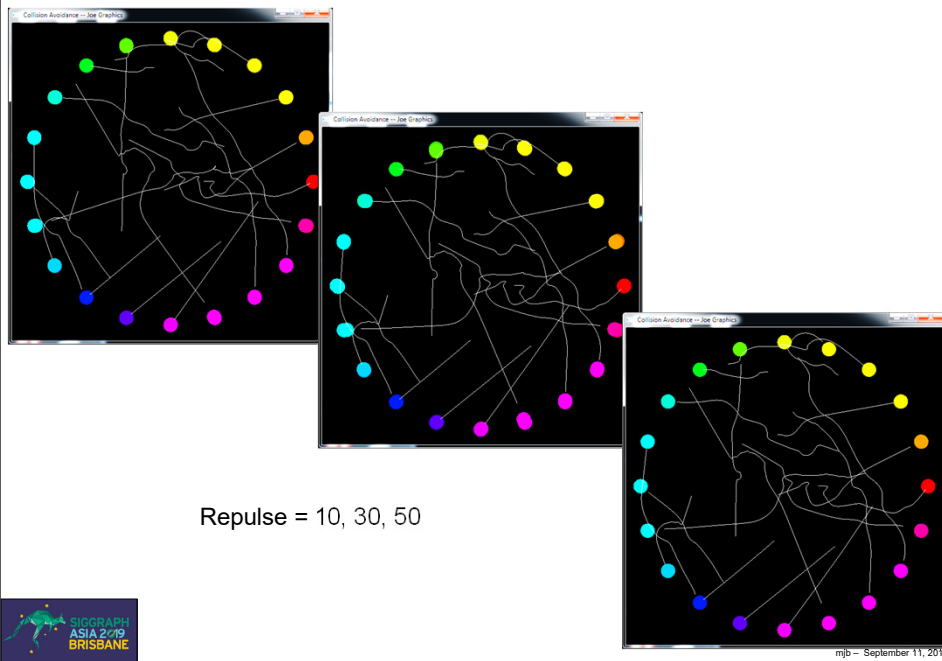
Stiffness = 3, 6, 9



mjb - September 11, 2019

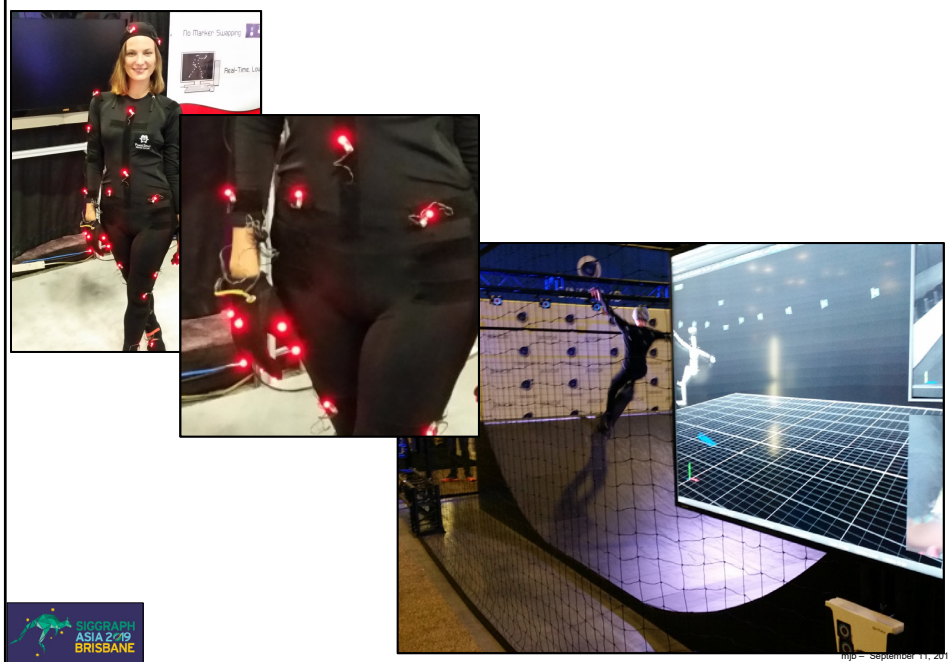
## Increasing the Repulsion Coefficient

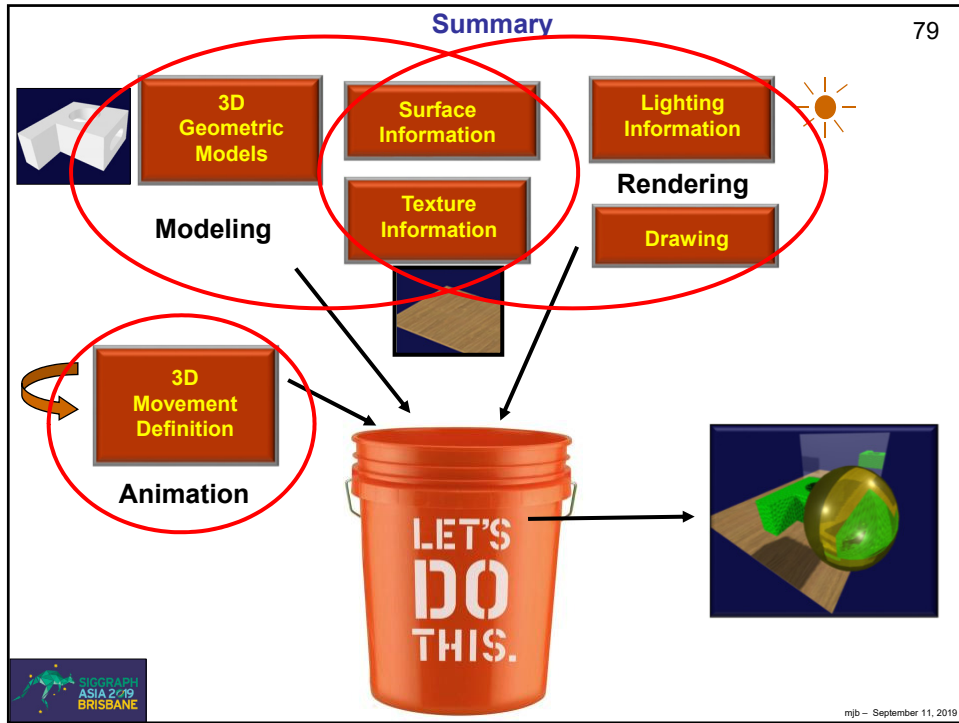
77



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

78





### 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*, 6<sup>th</sup> Edition, Addison-Wesley, 2011.

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

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

Alan Watt, *3D Computer Graphics*, 3<sup>rd</sup> Edition, Addison-Wesley, 2000.

Peter Shirley, *Fundamentals of Computer Graphics*, 2<sup>nd</sup> Edition, AK Peters, 2005.

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



### Conclusions !

- This week, these moments, will never come again.
- Take advantage of them.
- Combine what you have just learned here with what you see this week and relate them to your career and life goals.
- Especially pay attention to elements of the conference that you cannot re-live afterwards.
- Have fun!





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



83



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

mjb - September 11, 2019