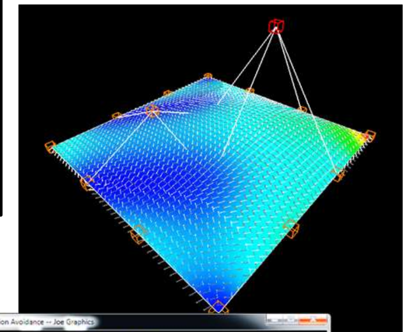
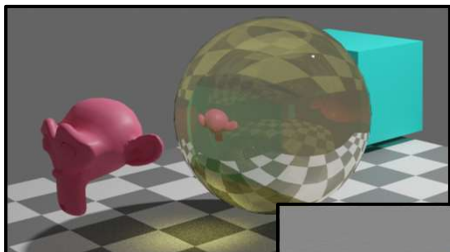
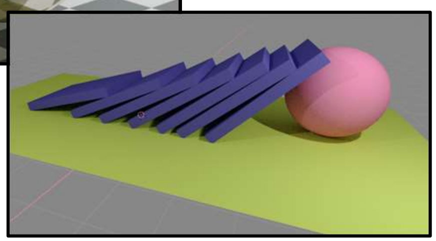
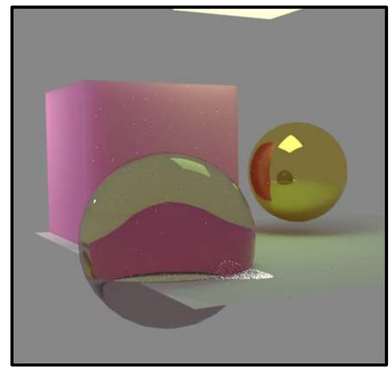
A Whirlwind Introduction to Computer Graphics

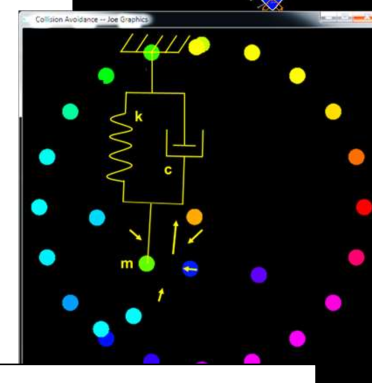
Mike Bailey
Oregon State University
mjb@cs.oregonstate.edu



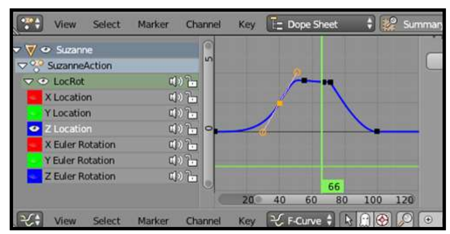


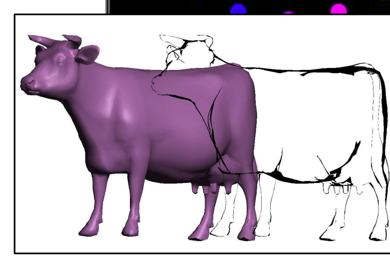






http://cs.oregonstate.edu/~mjb/whirlwind



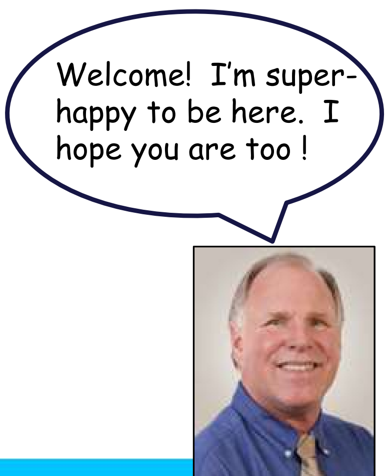


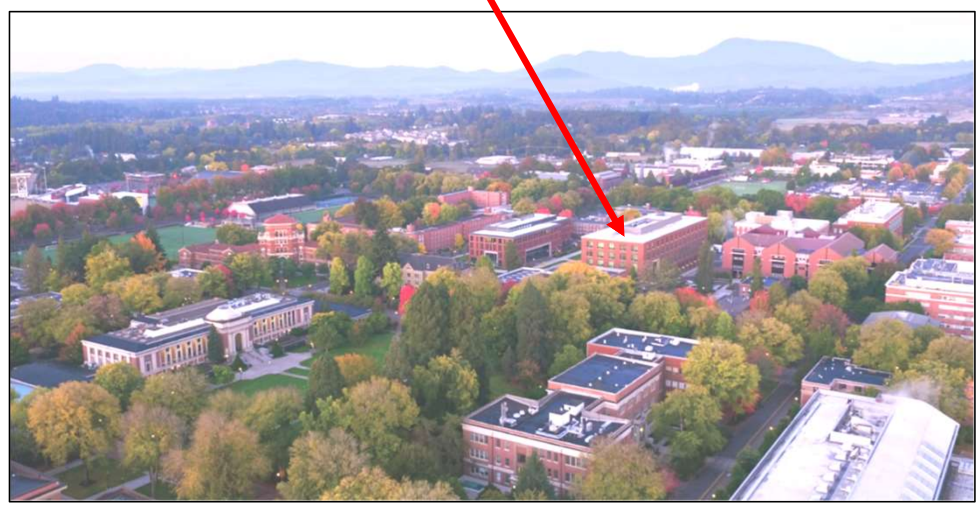
SIGGRAPH 2025
Vancouver+ 10-14 August

Mike Bailey

Oregon State University
Computer Graphics

- Professor of Computer Science, Oregon State University
- Has had over 14,000 students in his university classes
- Has taught over 100 conference and workshop short courses
- mjb@cs.oregonstate.edu





SIGGRAPH 2025
Vancouver+ 10-14 August

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

SIGGRAPH 2025
Vancouver+ 10-14 August

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

Course Learning Objectives

At the end of this course, you will know:

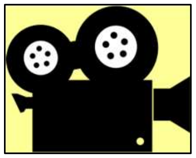
- The meaning of a lot of the jargon describing the amazing things at SIGGRAPH 2025. We call that “buzzword compliant”. ☺
- Some of what it took to make the images and animations that you will see
- How to find references for further study

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

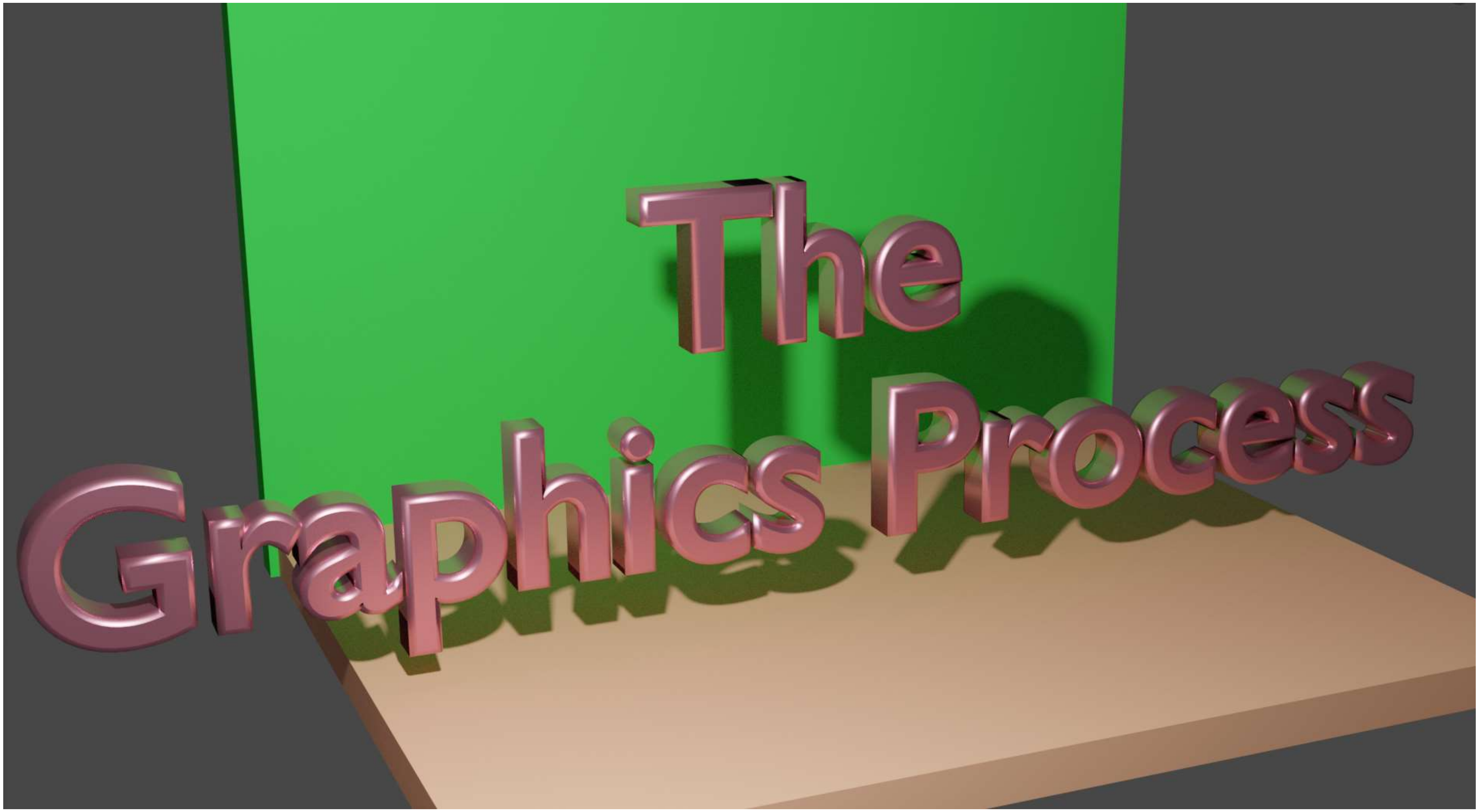
Schedule

1. 0:05 How the computer graphics pieces fit together
2. 0:20 Modeling
3. 0:20 Animation
4. 0:30 Rendering
5. 0:05 Finding More Information
6. 0:10 Q&A

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

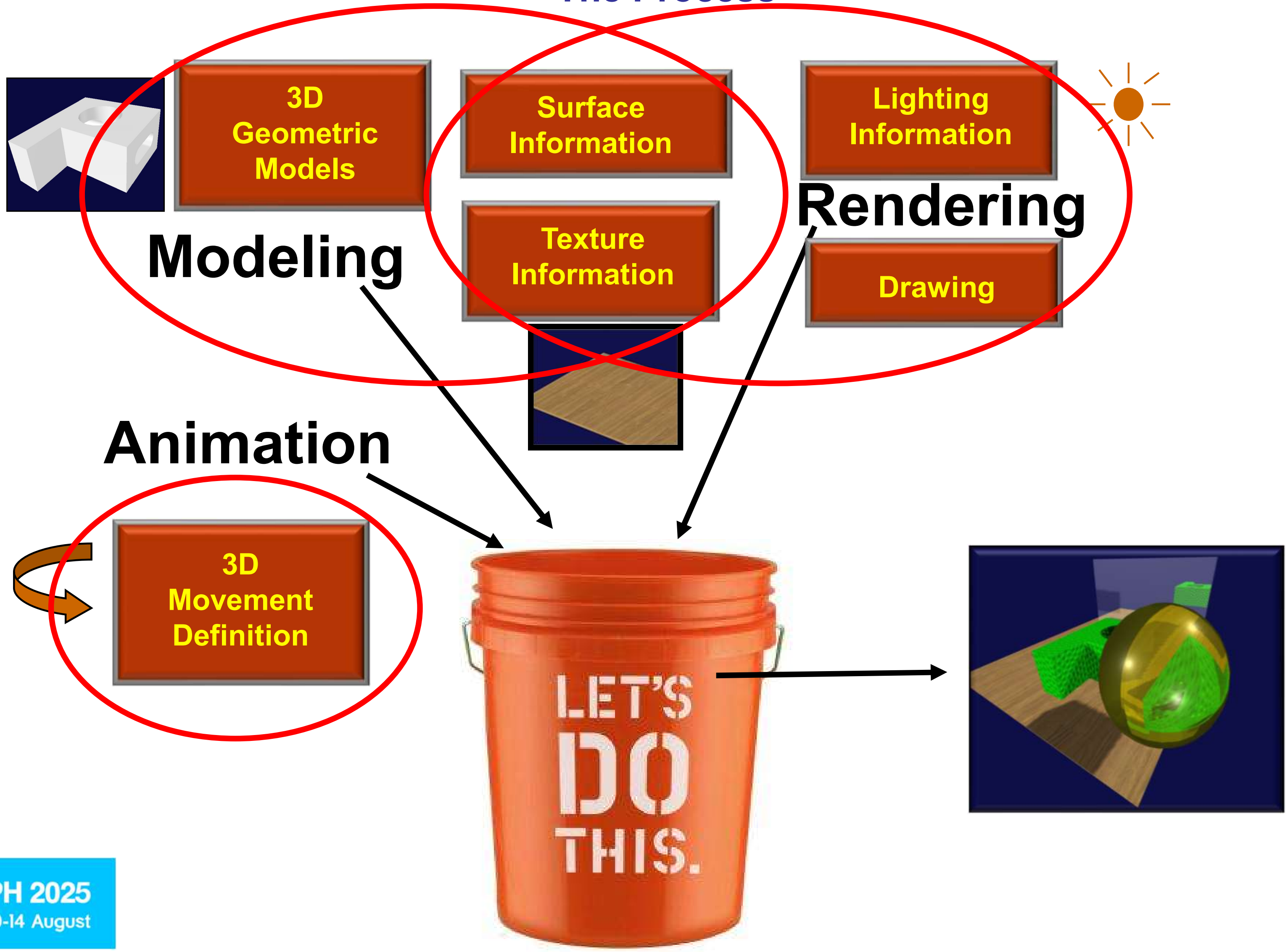


When you see a symbol like this, it means that there is a video on the Whirlwind page that you can watch for further information

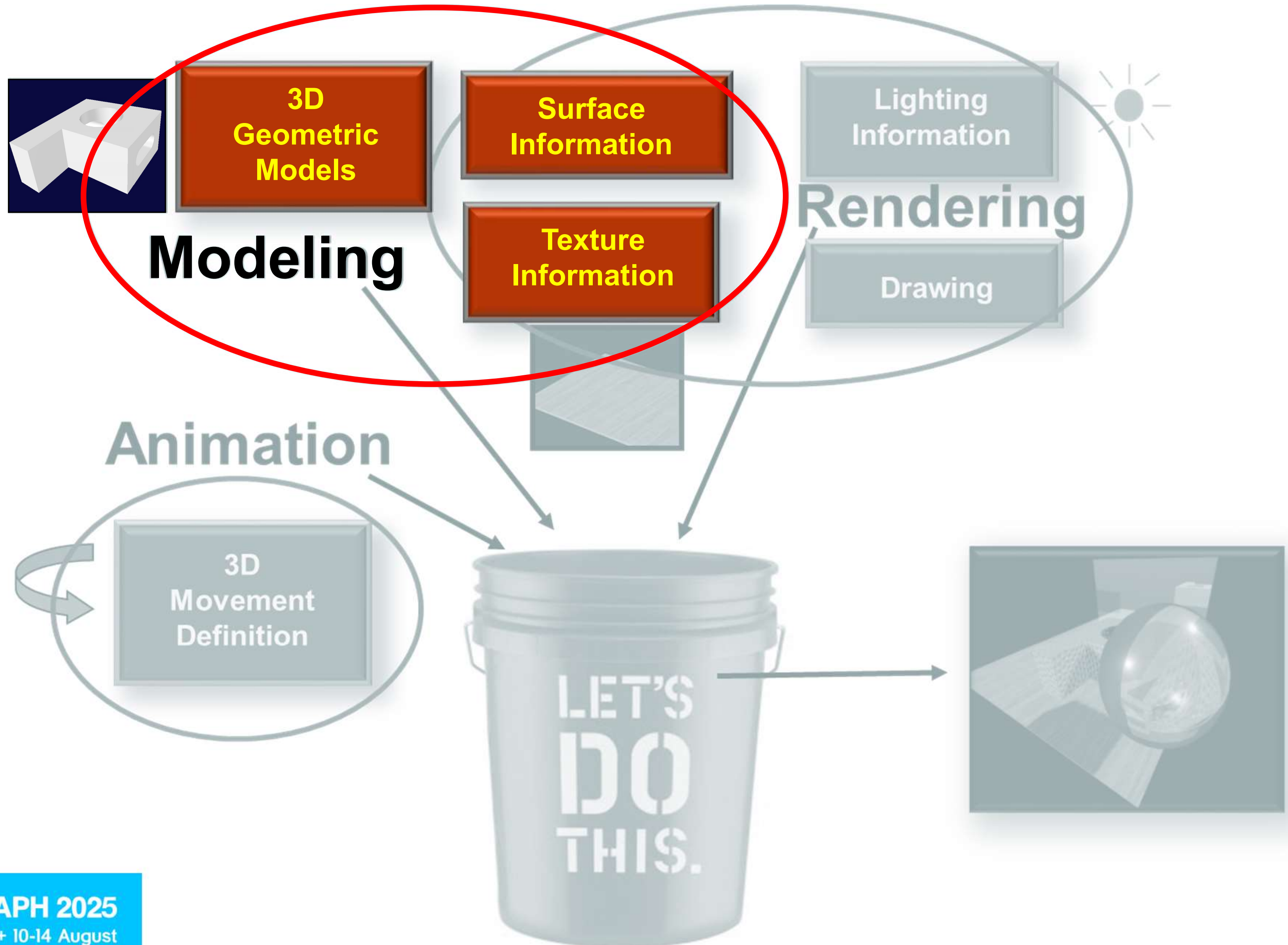


What are all the pieces that go into making the graphics you will be see?
What does it take to make them?

The Process


3D
Geometric
Models
Surface
Information
Lighting
Information
Rendering
Modeling
Texture
Information
Drawing
Animation
3D
Movement
Definition
LET'S
DO
THIS.

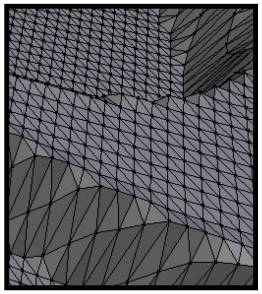
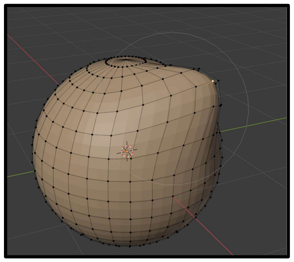
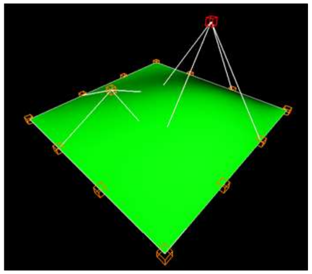
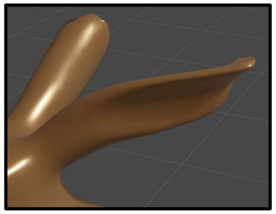
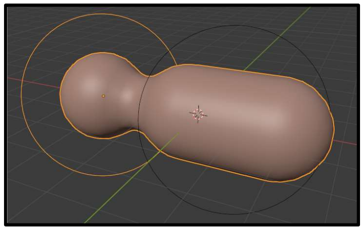
Creating 3D Geometry


3D
Geometric
Models
Surface
Information
Lighting
Information
Rendering
Modeling
Texture
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?
- Does it need to be 3D-printed?
- Etc.

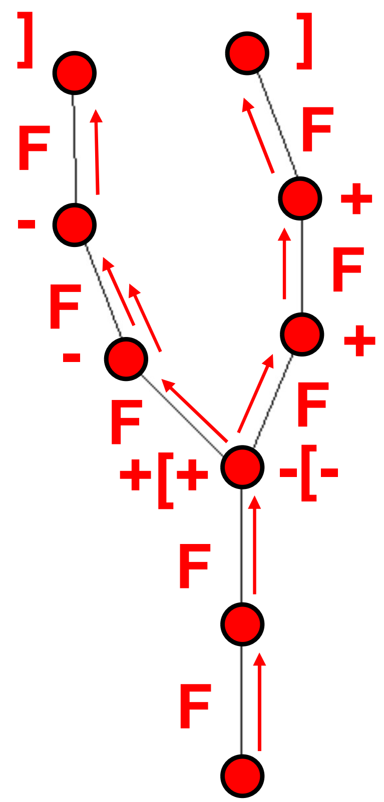
Want to experiment with some free modeling programs?
Want some notes to get you started?
http://cs.oregonstate.edu/~mjb/blender
http://cs.oregonstate.edu/~mjb/tinkercad

L-Systems are a Special Way to Model 3D Line Geometry

Introduced and developed in 1968 by Aristid Lindenmayer, L-systems are a way to apply grammar rules for generating fractal (self-similar) geometric shapes. For example, take the string:

"FF+[+F-F-F]-[-F+F+F]"

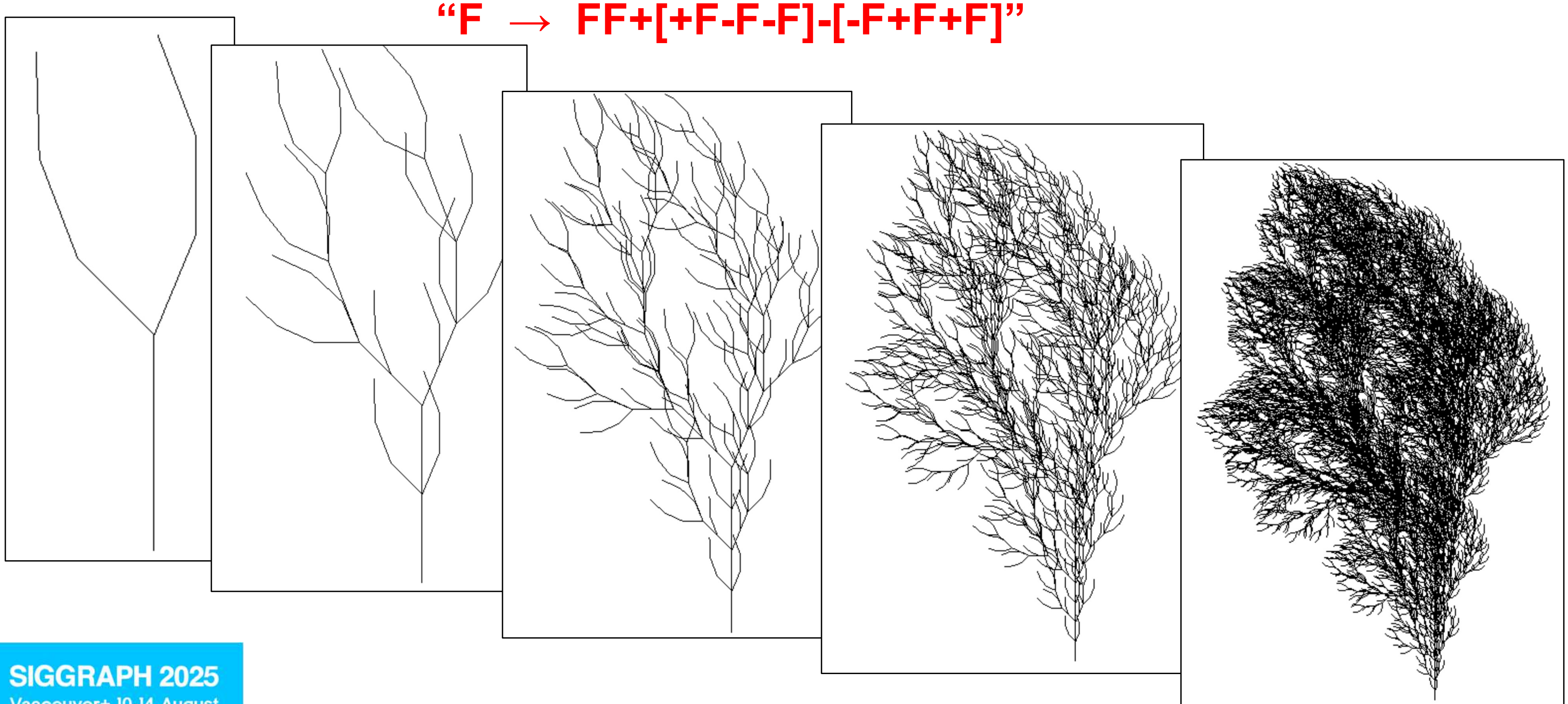
F	move forward one step
+	turn right
-	turn left
[	save position
]	restore position



L-Systems as a Special Way to Model 3D Line Geometry

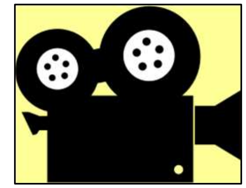
But the *real* fun comes when you call that string recursively. For every **F**, replicate it with that entire string but with smaller geometry:

"F → FF+[+F-F-F]-[-F+F+F]"



L-Systems as a Special Way to Model 3D Line Geometry

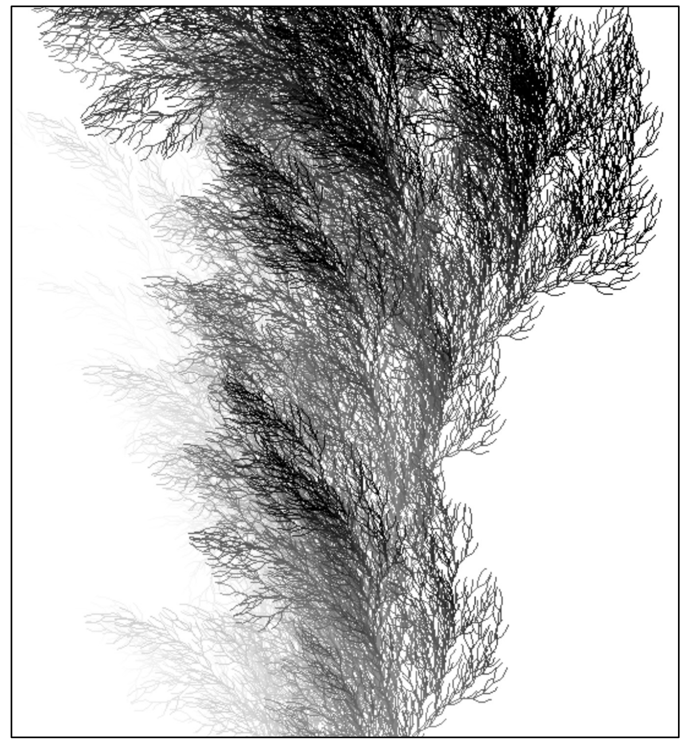
And, of course we can introduce more grammar to swing it into 3D



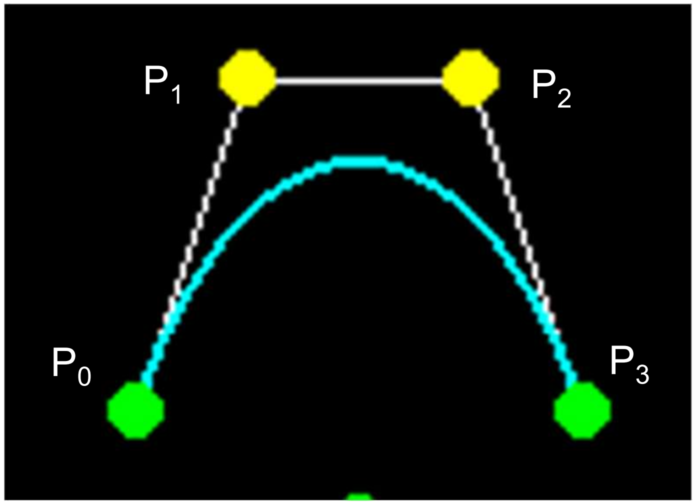
lsystems.mp4

"F → FF+[+F-<F->F]-[-F+^F+vF]"

F	move forward one step
+	rotate + about Z
-	rotate - about Z
<	rotate + about Y
>	rotate – about Y
v	rotate + about X
^	rotate – about X
[	save position
]	restore position



Another way to Model Geometry: Curve Sculpting

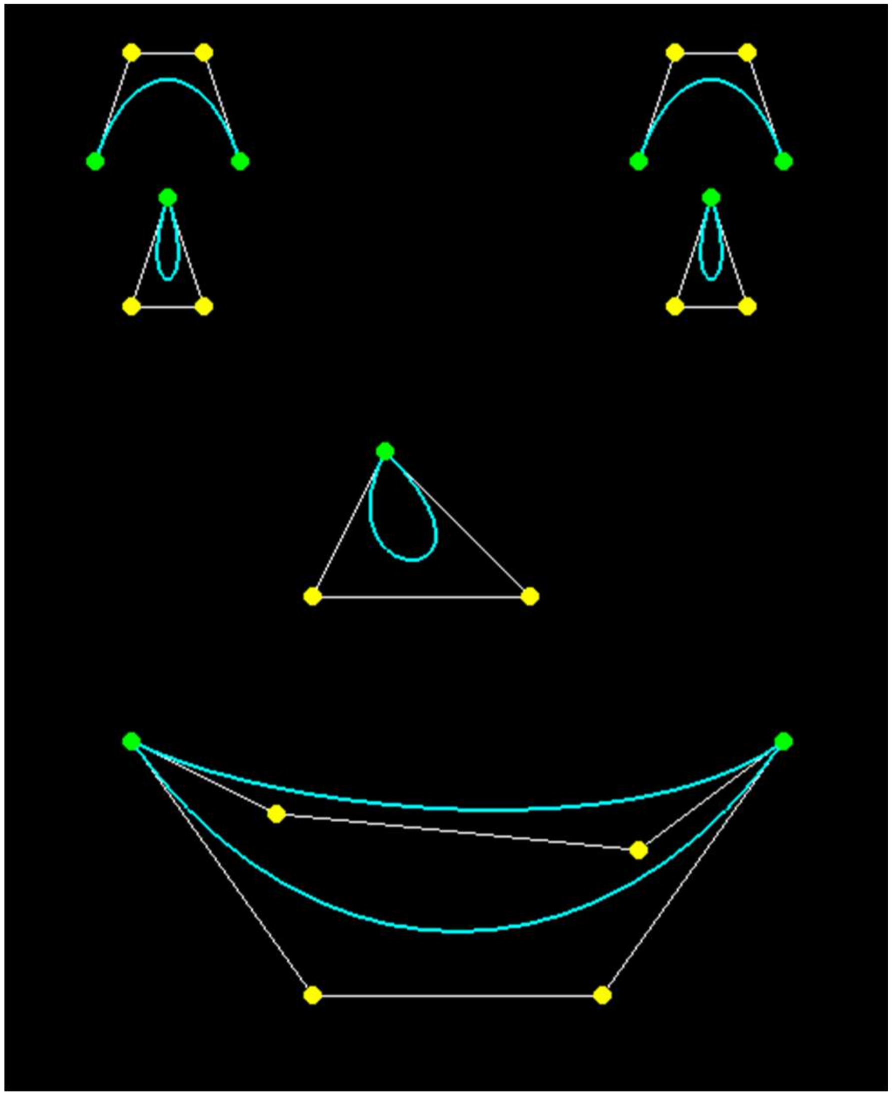


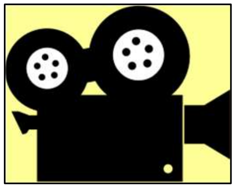
This equation is for a cubic Bezier curve:

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

$$0. \le t \le 1.$$

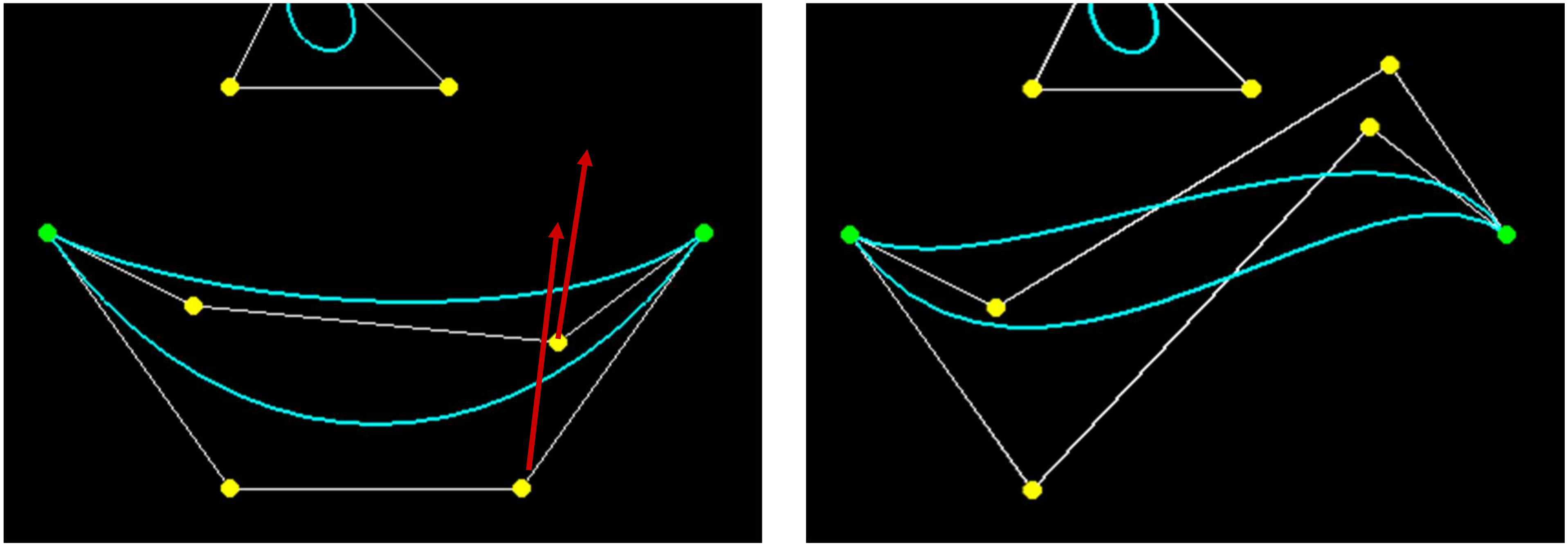
Curve Sculpting – Bézier Curve Sculpting Example





curves.mp4

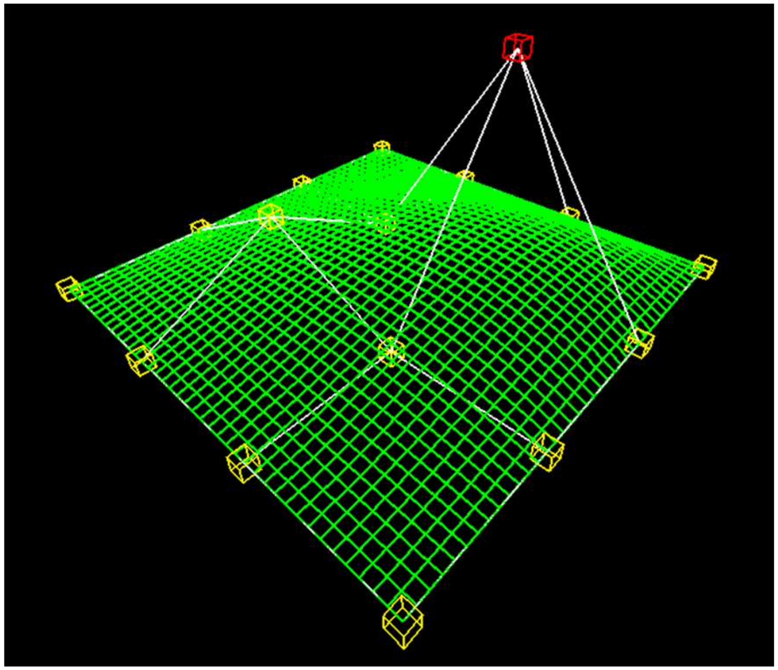
Curve Sculpting – Bézier Curve Sculpting Example



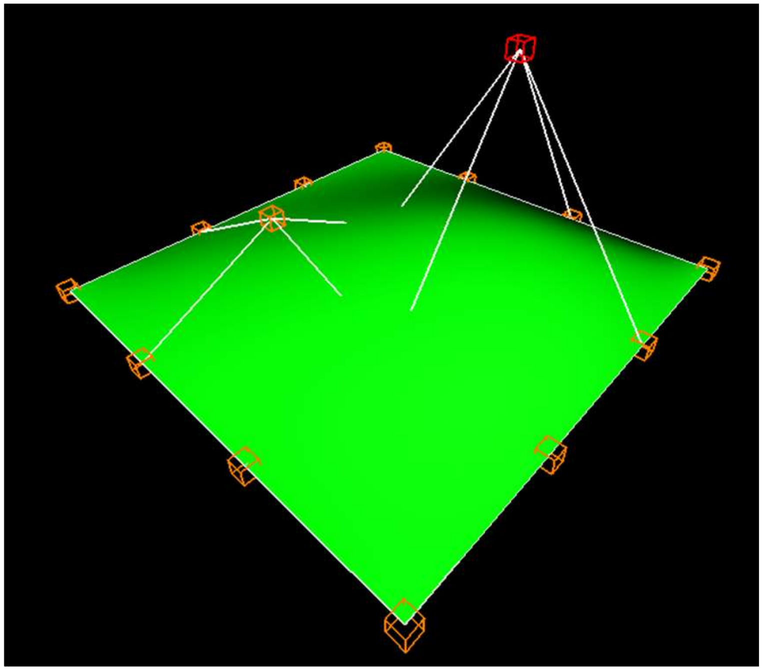
A *Small* Amount of Input Change Results in a *Large* Amount of Output Change

Another way to Model: Surface Sculpting

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



Wireframe

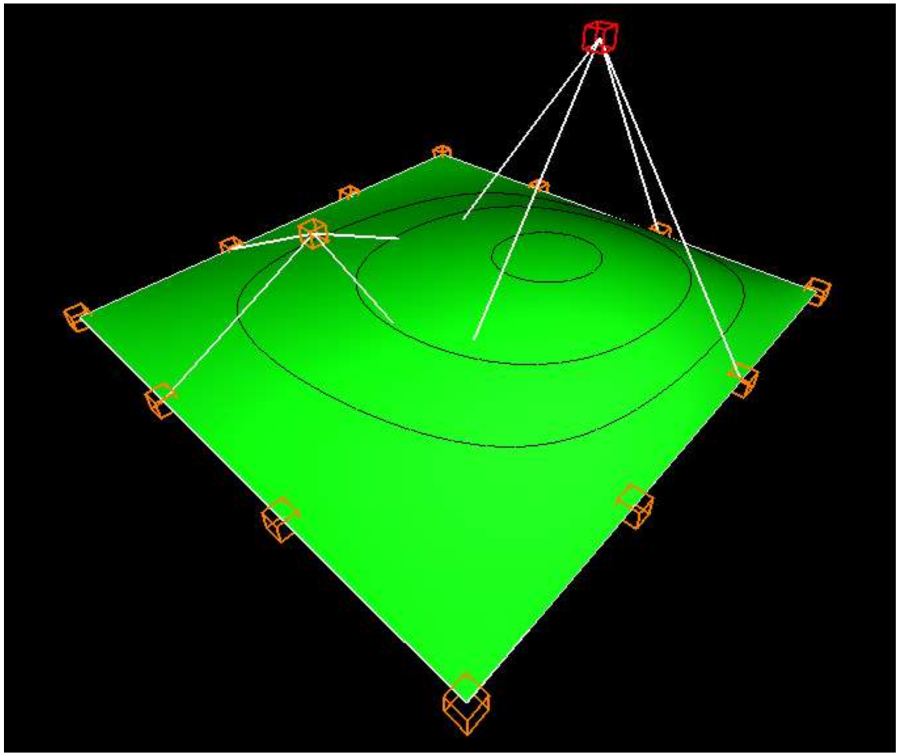


Surface

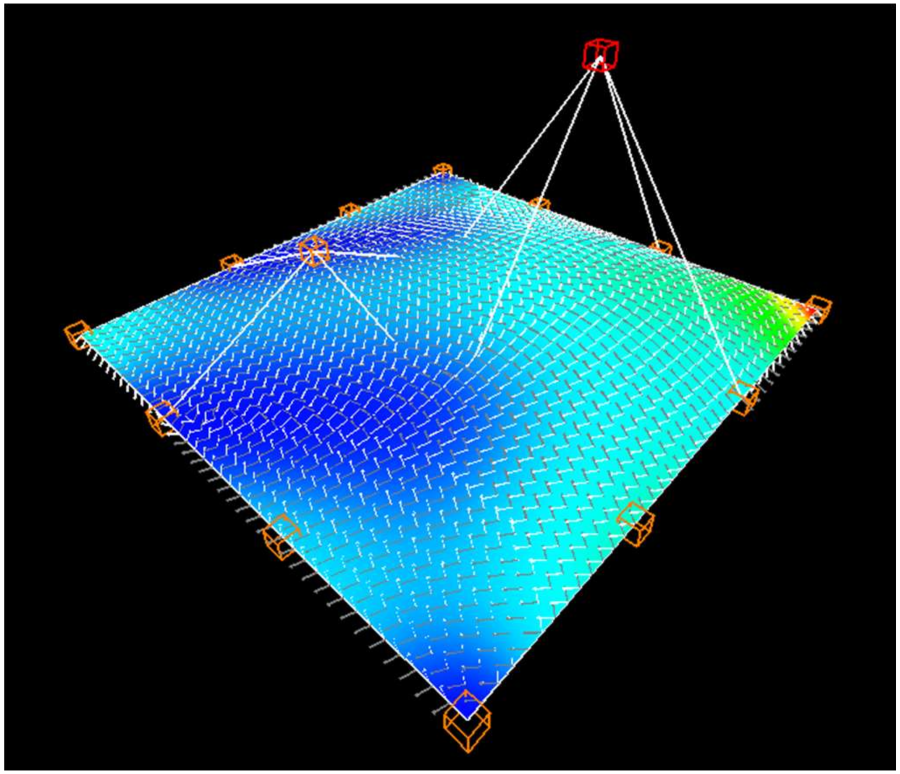
SIGGRAPH 2025 Vancouver+ 10-14 August

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 Mathematical Analysis



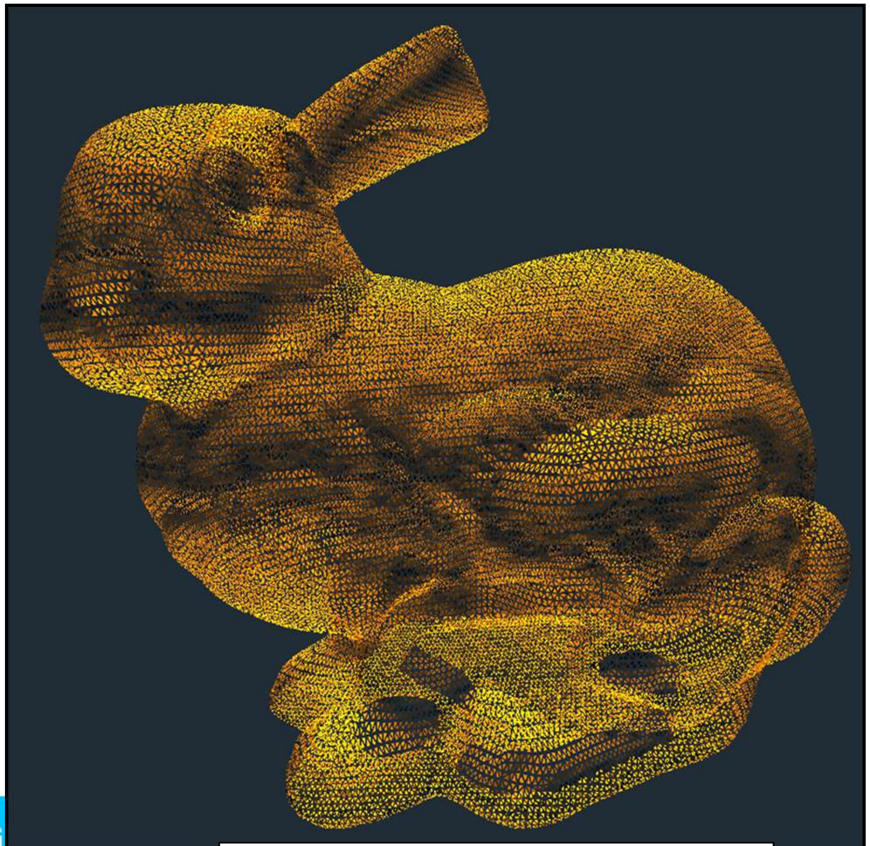
Showing Contour Lines



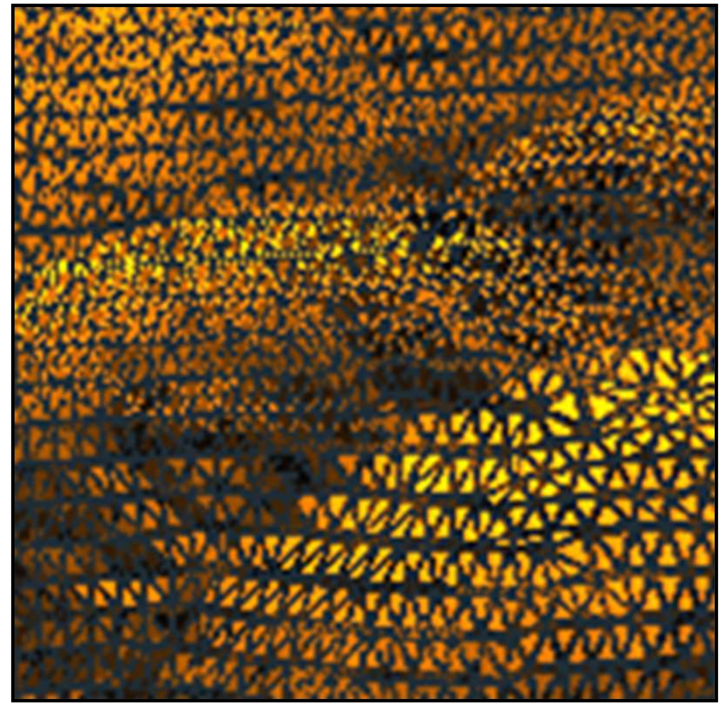
Showing Curvature

Explicitly Listing Geometry (3D Points) and Topology (How They Are Connected)

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

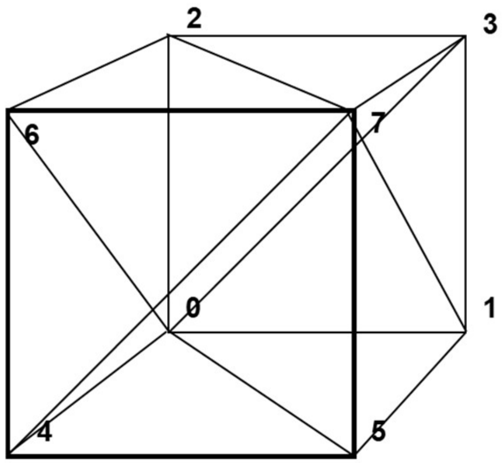


http://graphics.stanford.edu/data/3Dscanrep



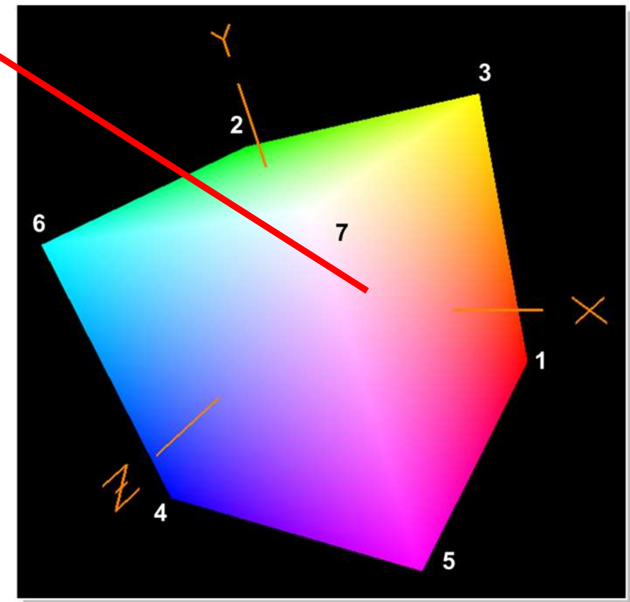
This is often called a **Mesh**, or sometimes a **Triangular Irregular Network** (TIN).

Explicitly Listing Triangular Mesh Coordinates and Connections in Tables

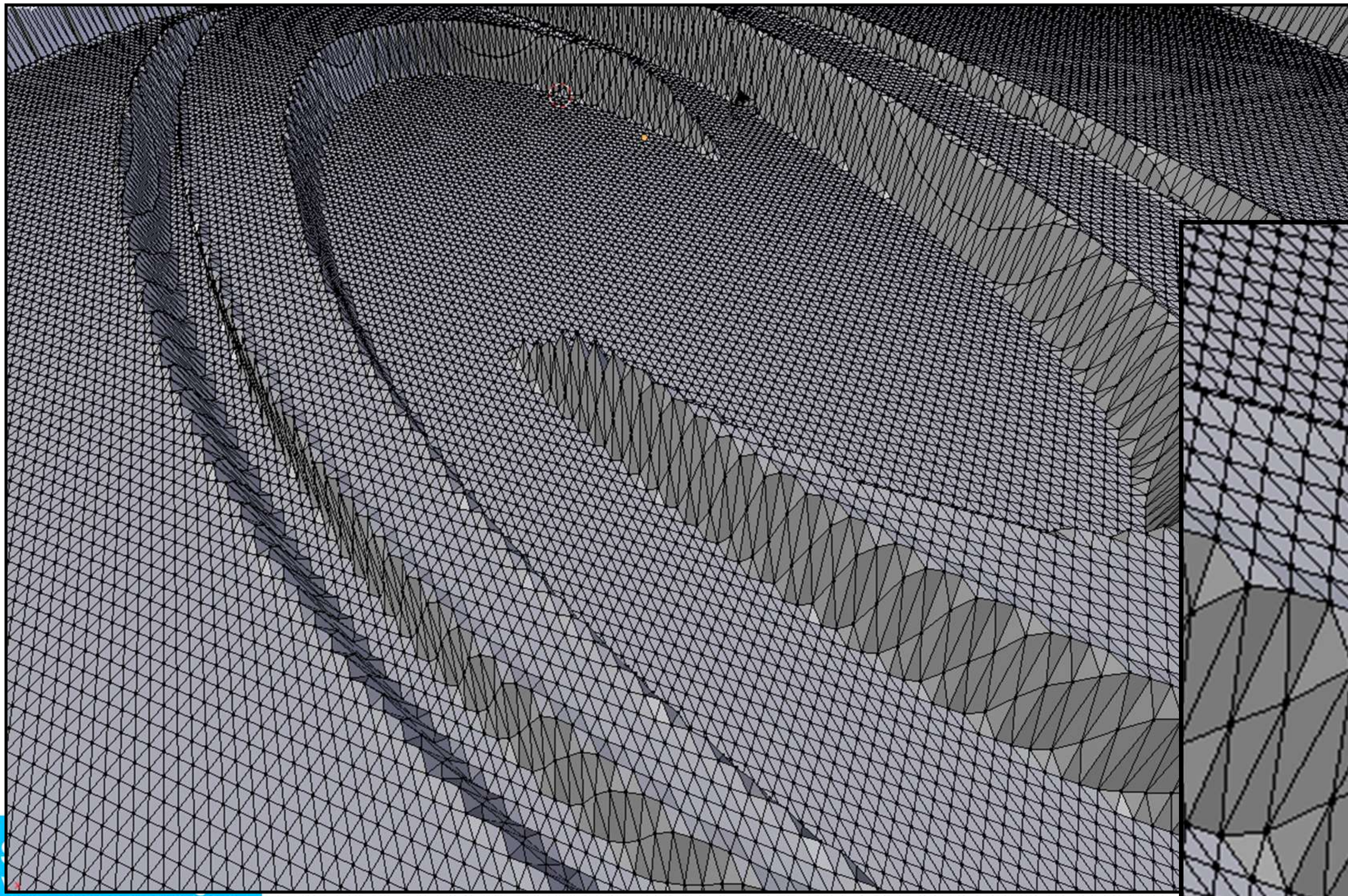


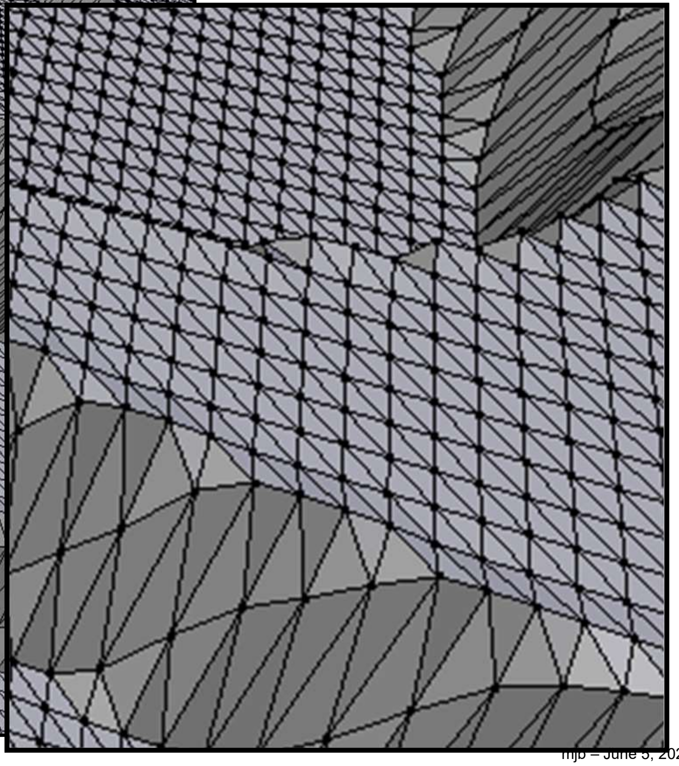

```
float VertexCoordinates[  ][3] =
{
                { -1., -1., -1. },
                {  1., -1., -1. },
                { -1.,  1., -1. },
                {  1.,  1., -1. },
                { -1., -1.,  1. },
                {  1., -1.,  1. },
                { -1.,  1.,  1. },
                {  1.,  1.,  1. }
};
```

```
int Indexes[  ][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 }
};
```

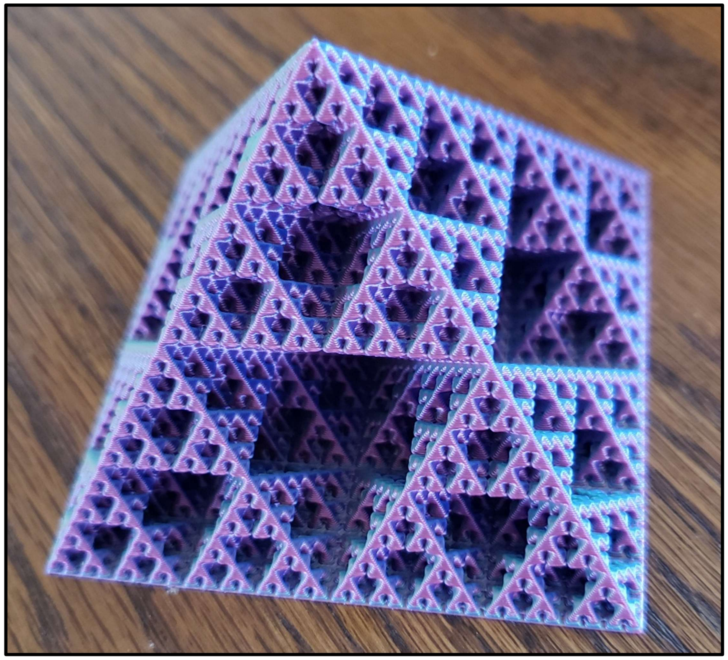


Triangular Meshes are Super Important These Days Because *3D Printing* Requires a Triangular Mesh Data Format





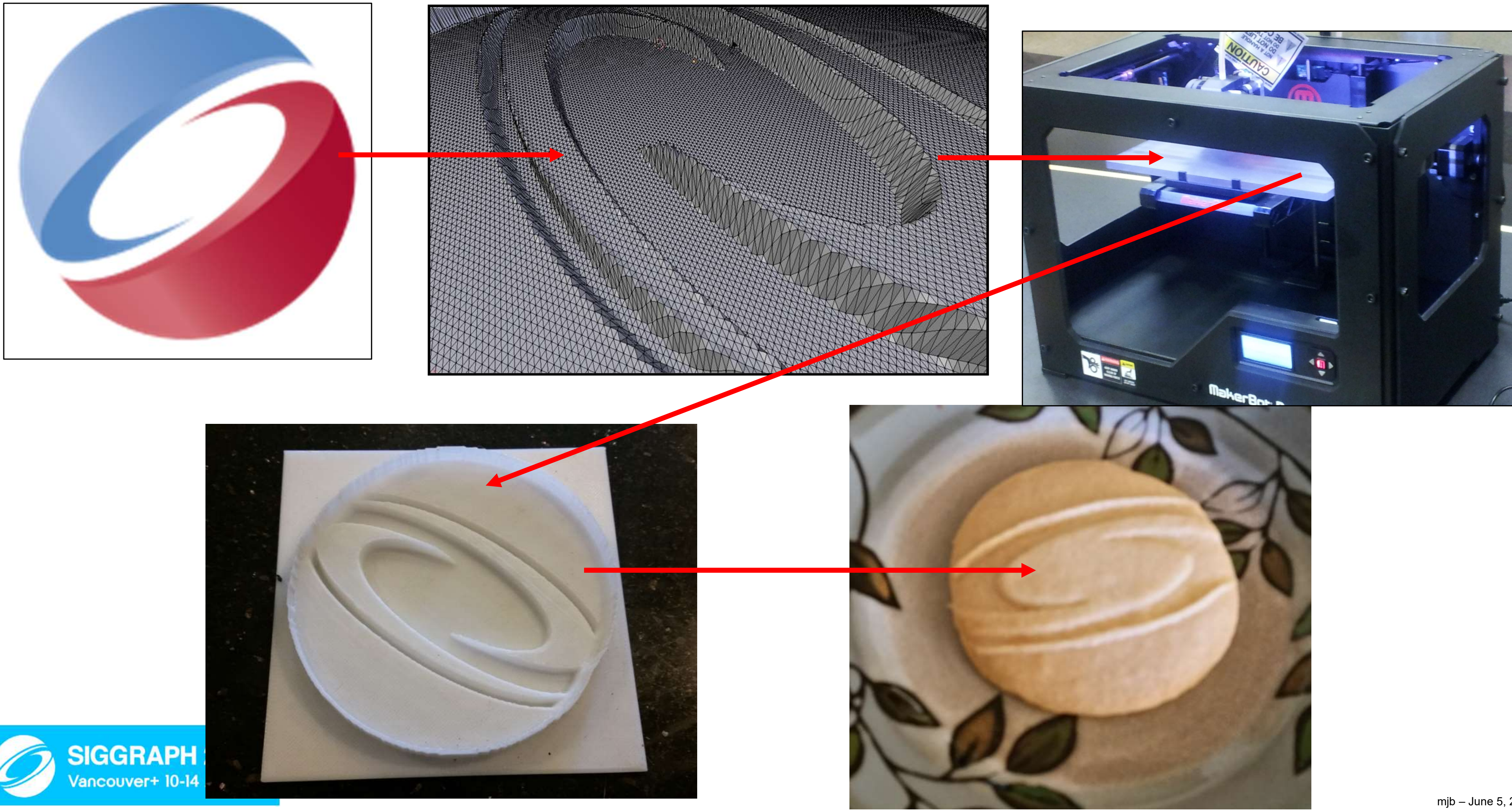
3D Printing Meshes Don't Always Look Very Mesh-ish Anymore – But They Are



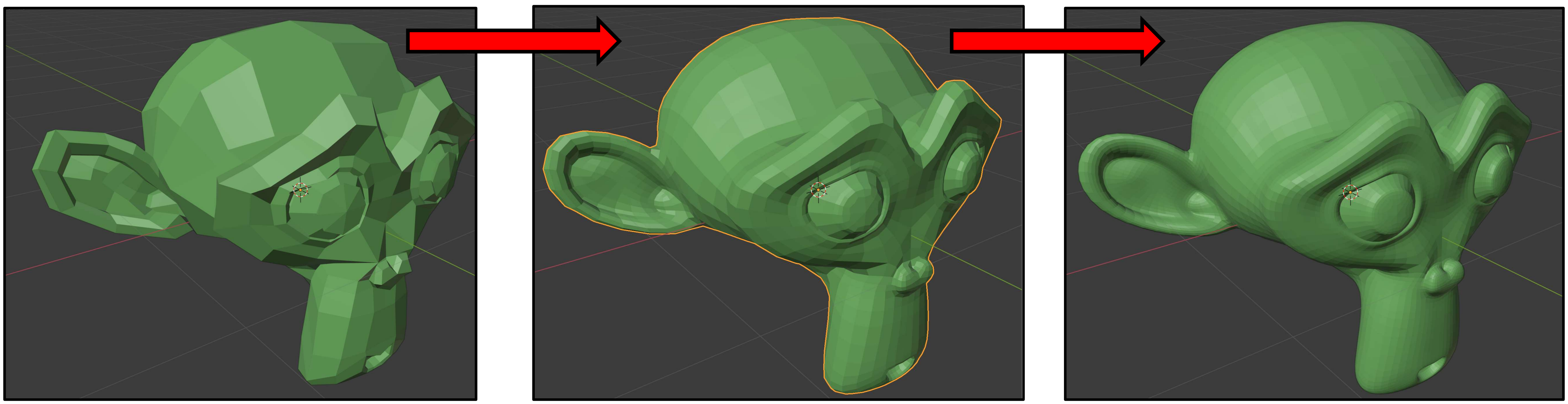
Models are from https://www.printables.com

3D Printed by Ryan Bailey
Images used by permission

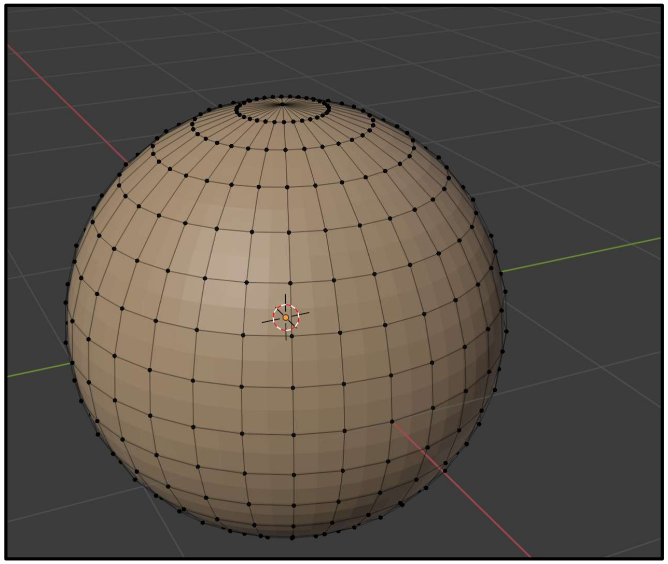
3D geometric modeling at its very best -- mmmm… :-)



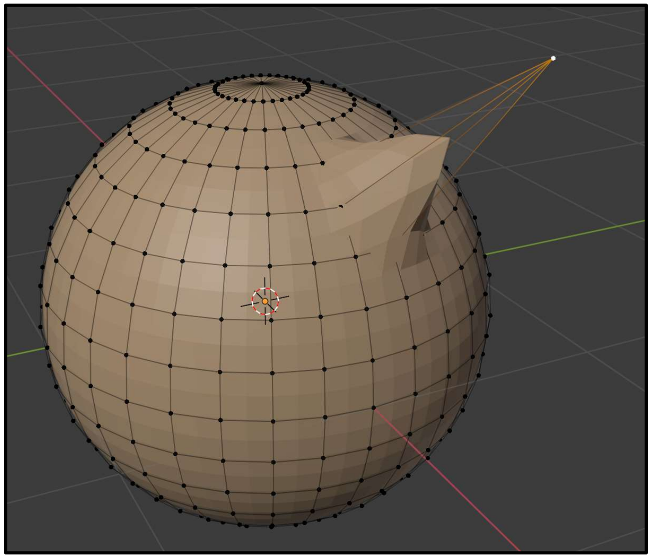
Meshes Can Be Smoothed



Meshes Can Be Edited

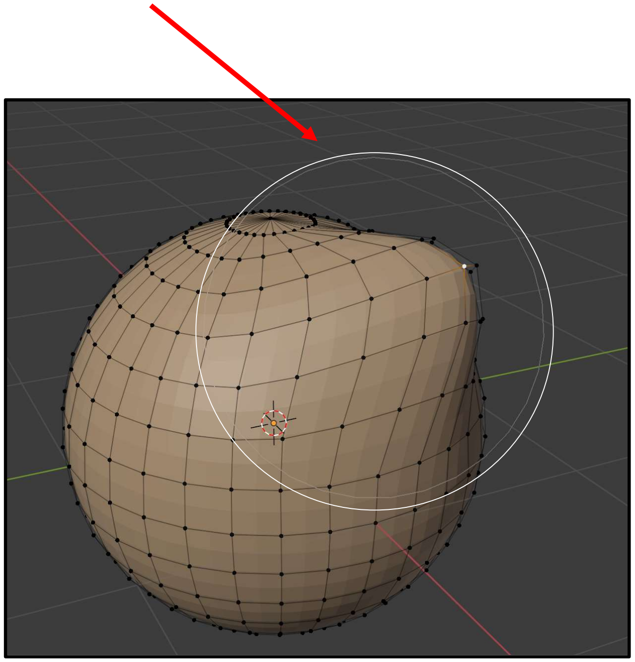


Original



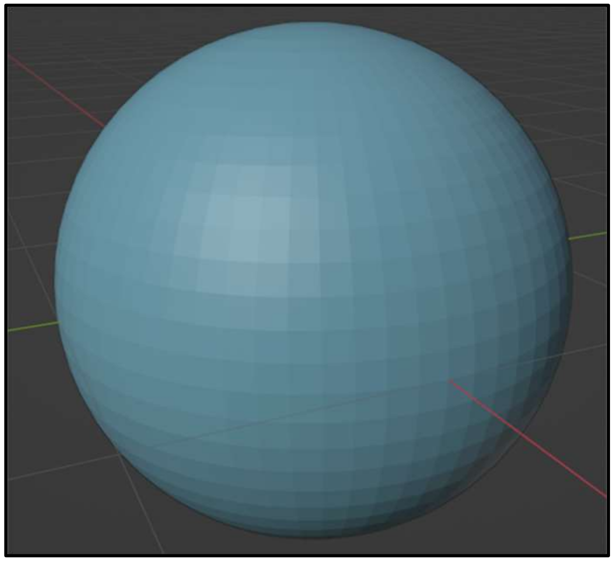
Pulling on a single Vertex

“Circle of Influence”

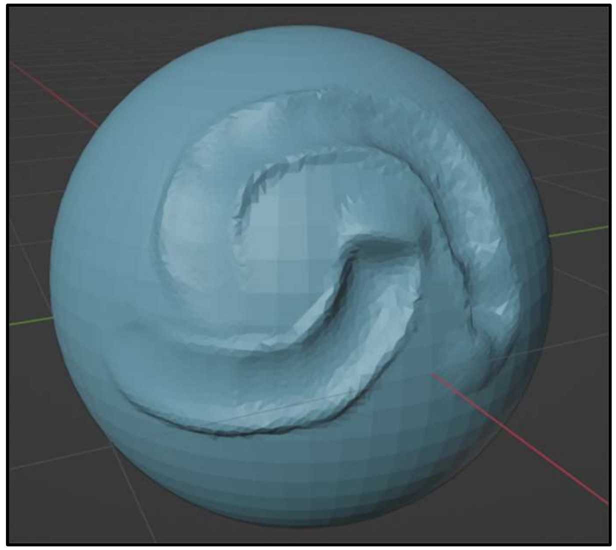


Pulling on a Vertex with
Proportional Editing Turned On

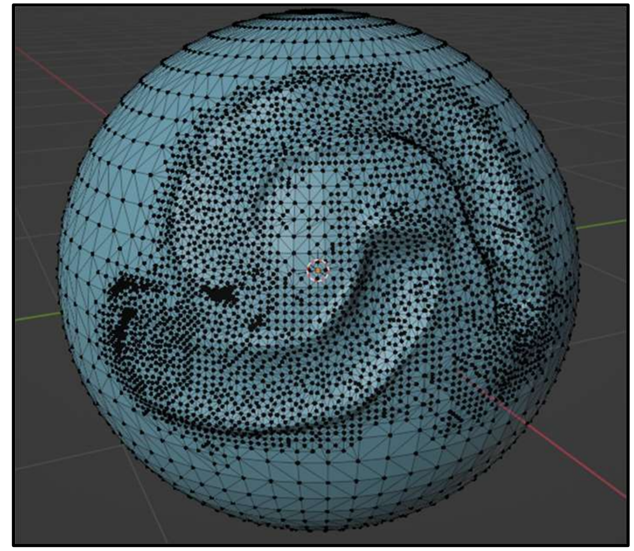
Meshes Can Be Sculpted



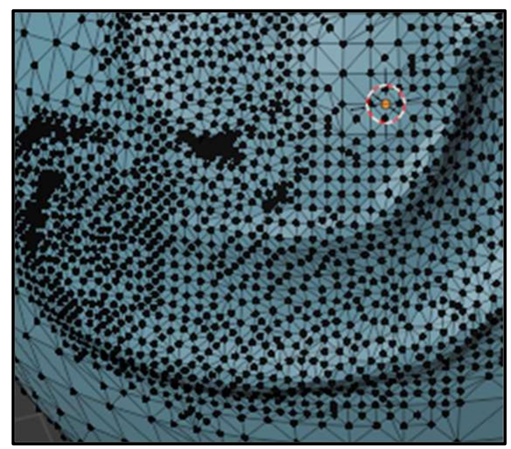
Original



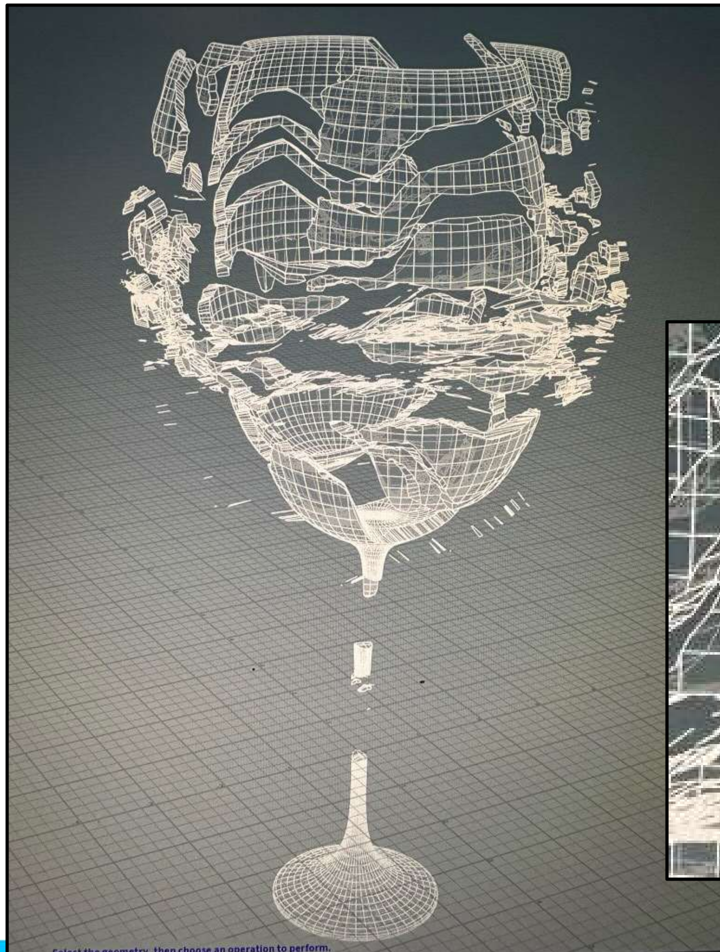
"Clay Thumb" Sculpting



Sculpting Can Produce Additional Mesh Vertices

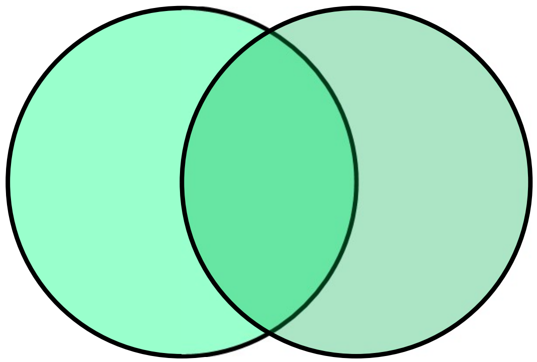


Meshes Can Be Used to Compute Physics

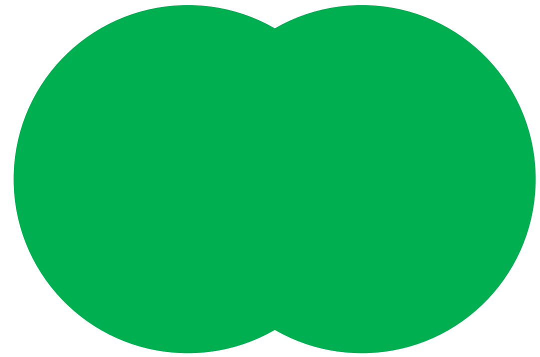


Natasha Anisimova, used by permission

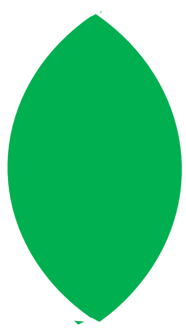
Another Way to Model in 3D:
Remember Venn Diagrams (2D Boolean Operators) from High School?



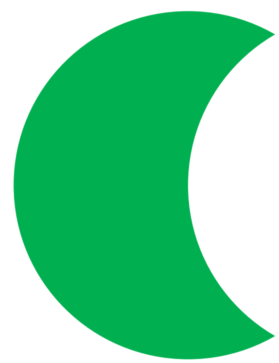
Two Overlapping Shapes



Their Union



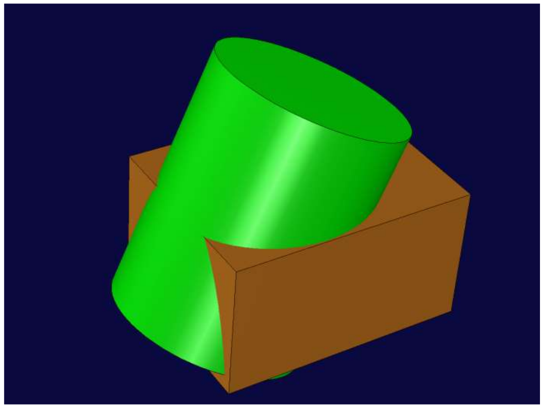
Their Intersection



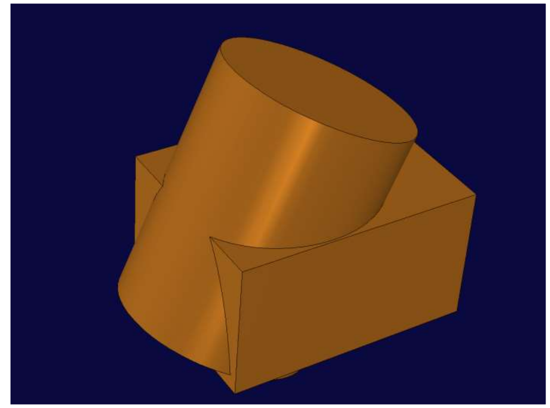
Their Difference

Blech! I thought I left Venn
Diagrams behind years ago !

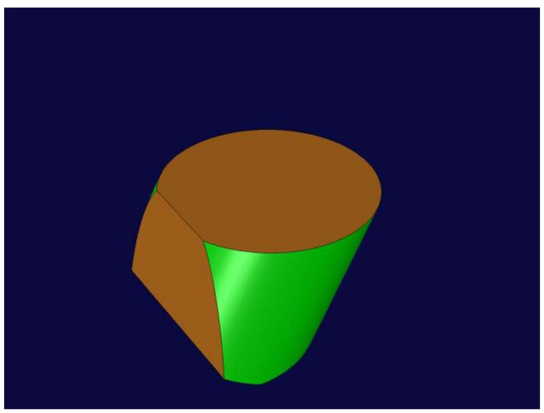
Solid Modeling Using 3D Boolean Operators



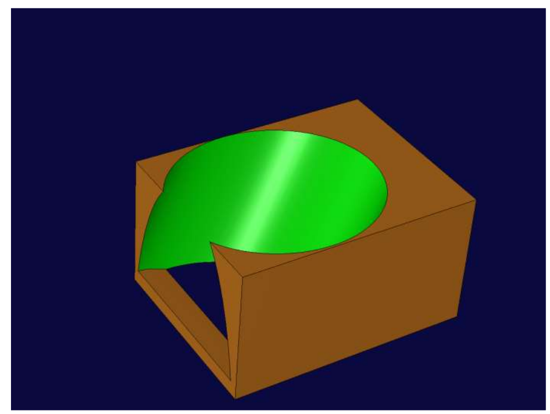
Two Overlapping Solids



Their Union



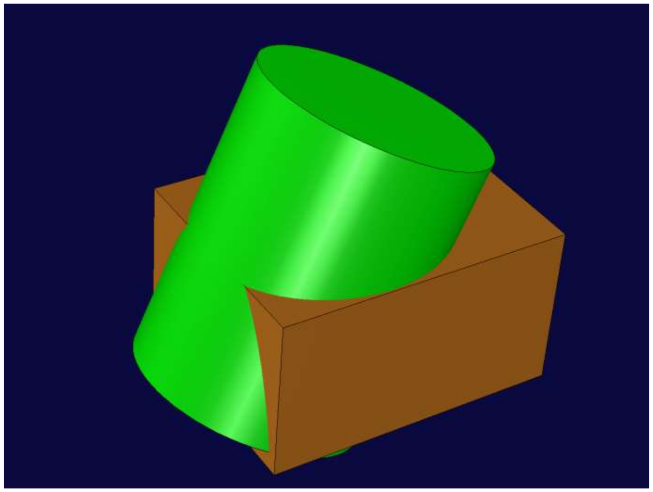
Their Intersection



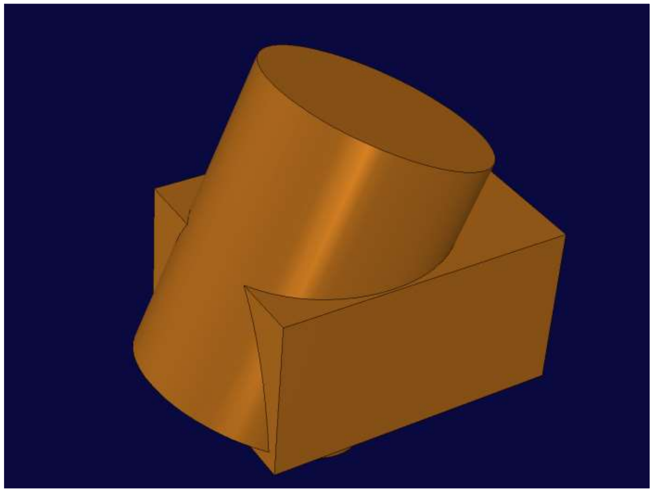
Their Difference

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

3D Boolean Operators are Important in 3D Printing as well as General Modeling



Two Overlapping Solids –
They Cannot Be 3D Printed

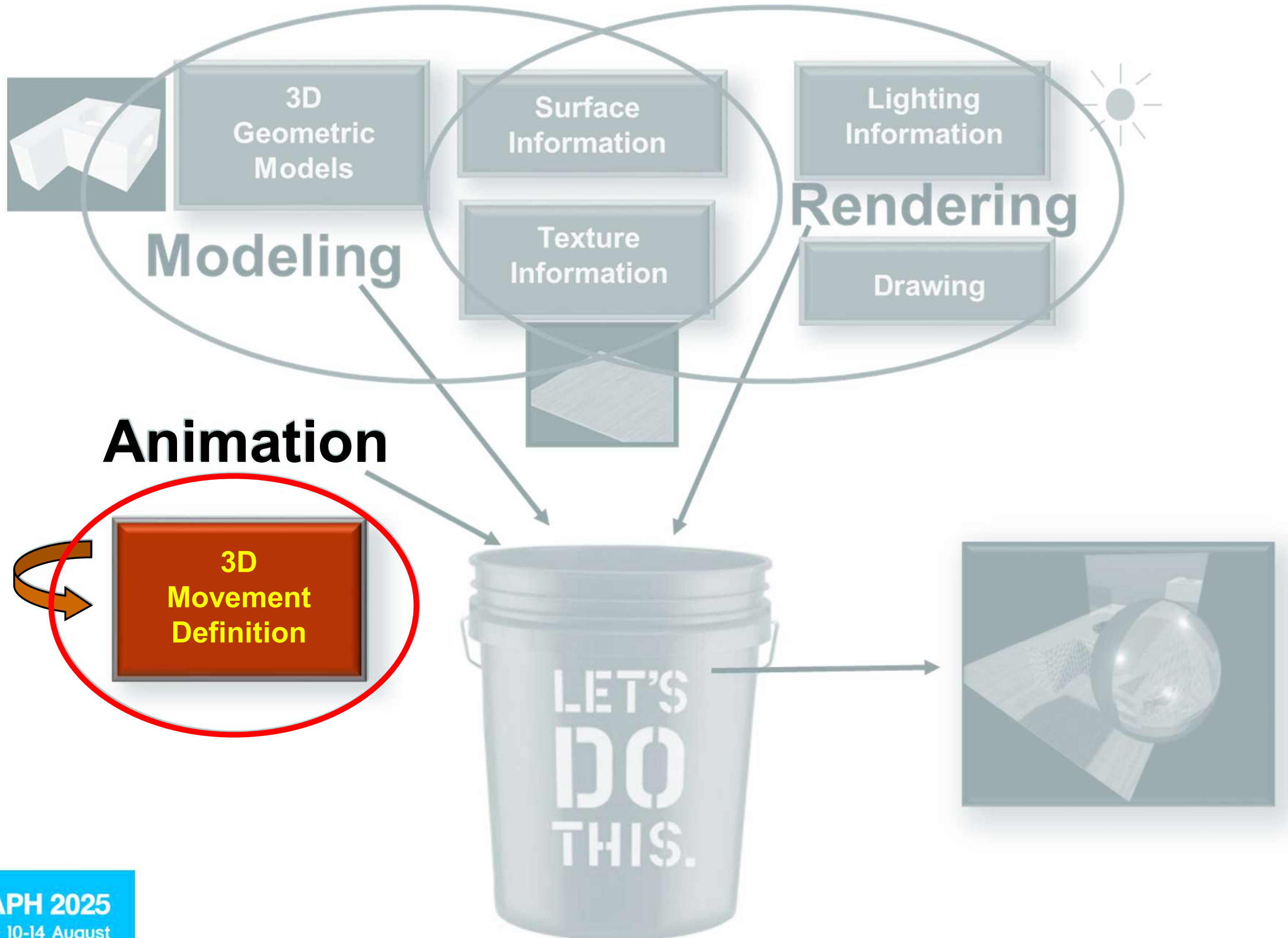


Two Overlapping Solids that have been Unioned – Now They Can Be 3D Printed

Intersected and Differenced Solids Can be 3D Printed as Well

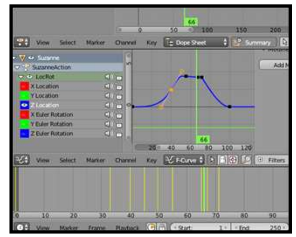
Want to experiment with 3D Booleans for free?
Want some notes to get you started?
http://cs.oregonstate.edu/~mjb/blender
http://cs.oregonstate.edu/~mjb/tinkercad

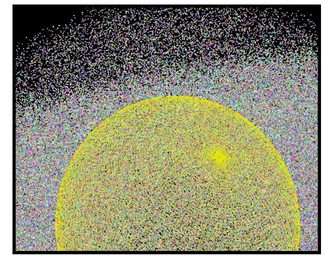
Creating the motion you want

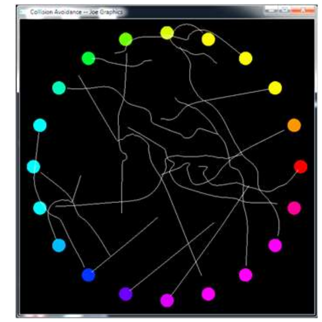

3D
Geometric
Models
Surface
Information
Lighting
Information
Rendering
Modeling
Texture
Information
Drawing
Animation
3D
Movement
Definition
LET'S
DO
THIS.

Animation

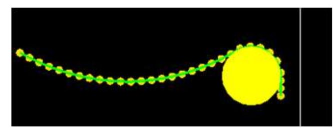
Rendering is the process of giving motion to your geometric modes. Again, there are questions you need to ask first:

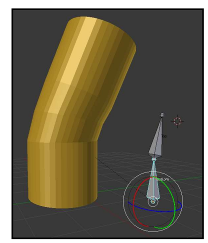


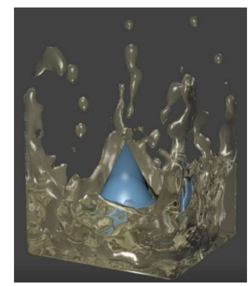


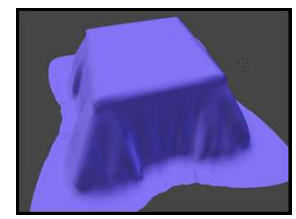


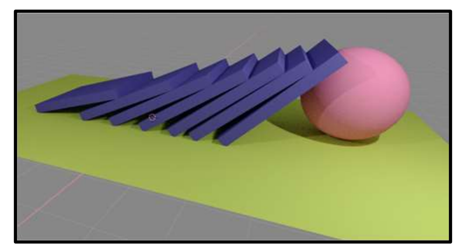
- Why am I doing this?
- Do I want the animation to obey the real laws of physics?
- Am I willing to “fake” the physics to get the objects to *want* to move in a way that I tell them?
- Do I have specific key positions I want the objects to pass through no matter what?
- Do I want to simply record the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





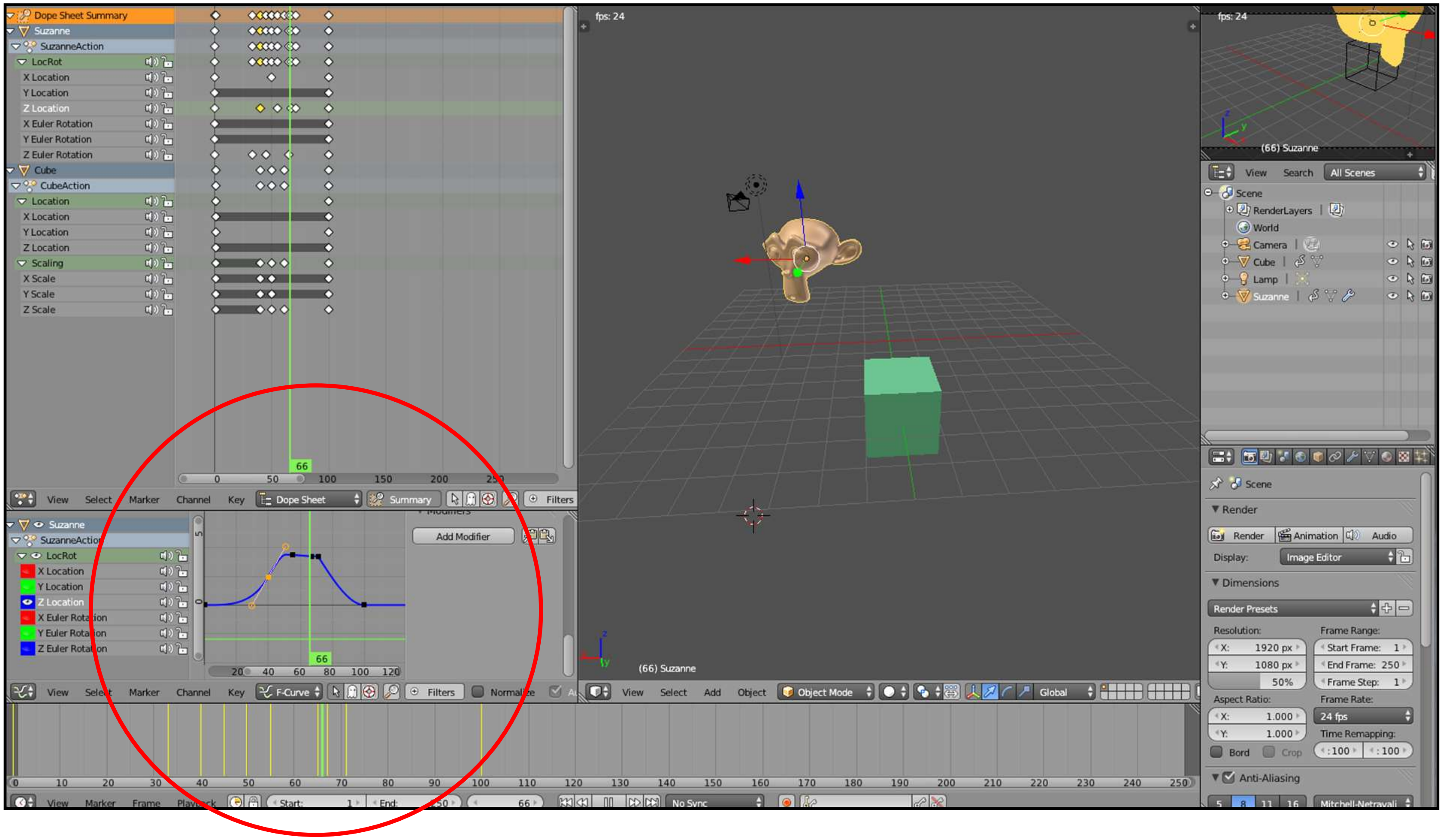






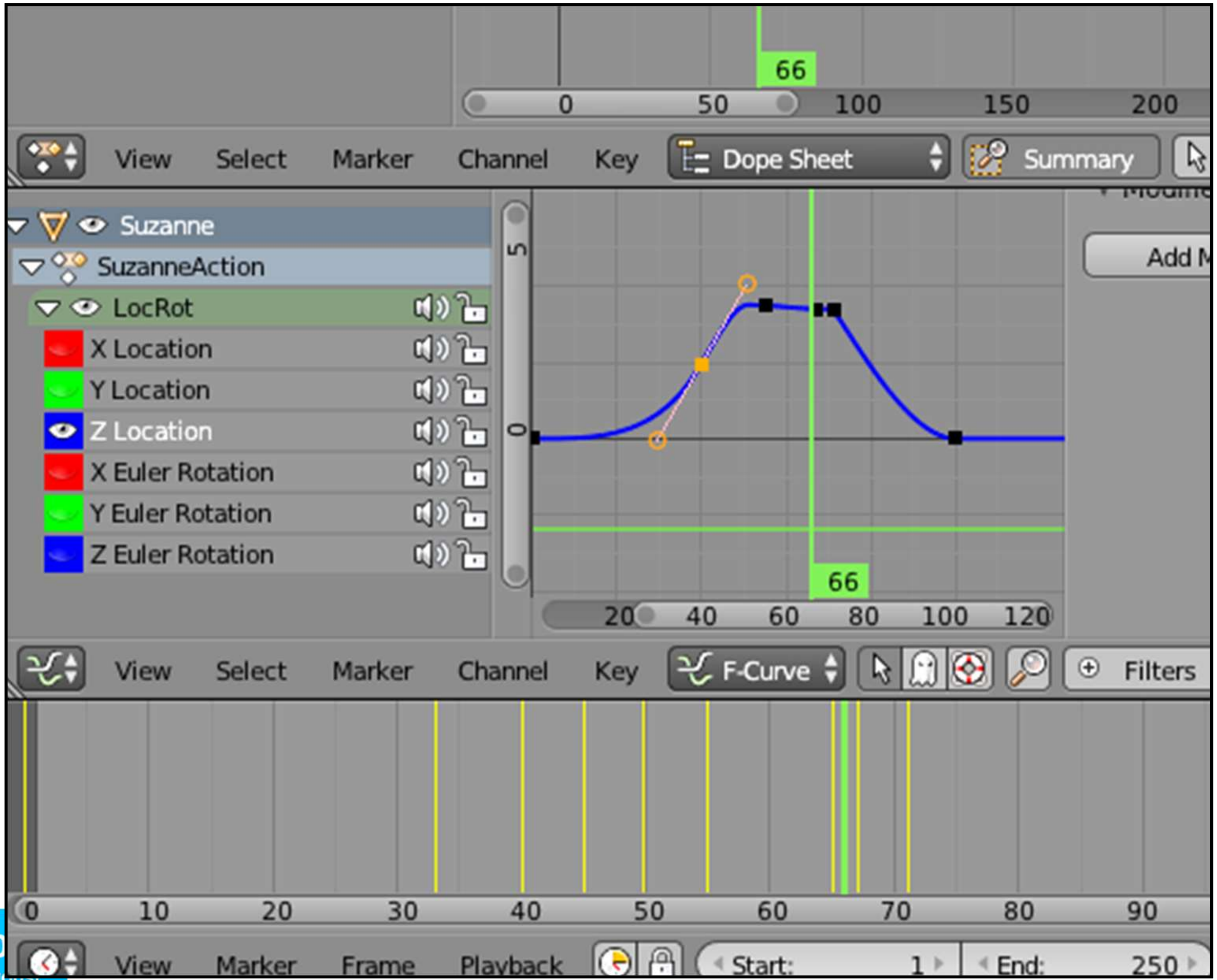
Want to experiment with free animation programs?
Want some notes to get you started?
http://cs.oregonstate.edu/~mjb/blender
http://cs.oregonstate.edu/~mjb/tinkercad

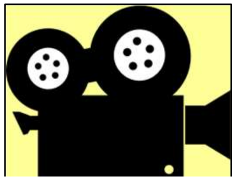
Keyframe Animation



Forcing the geometry to smoothly pass through key positions

Keyframe Animation

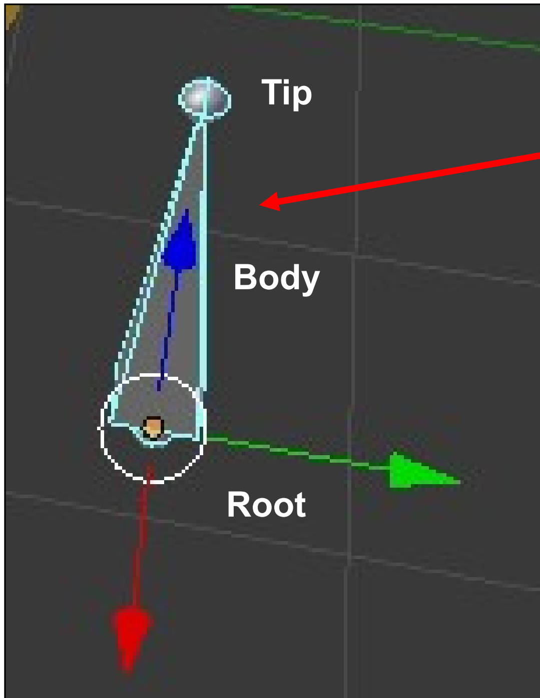


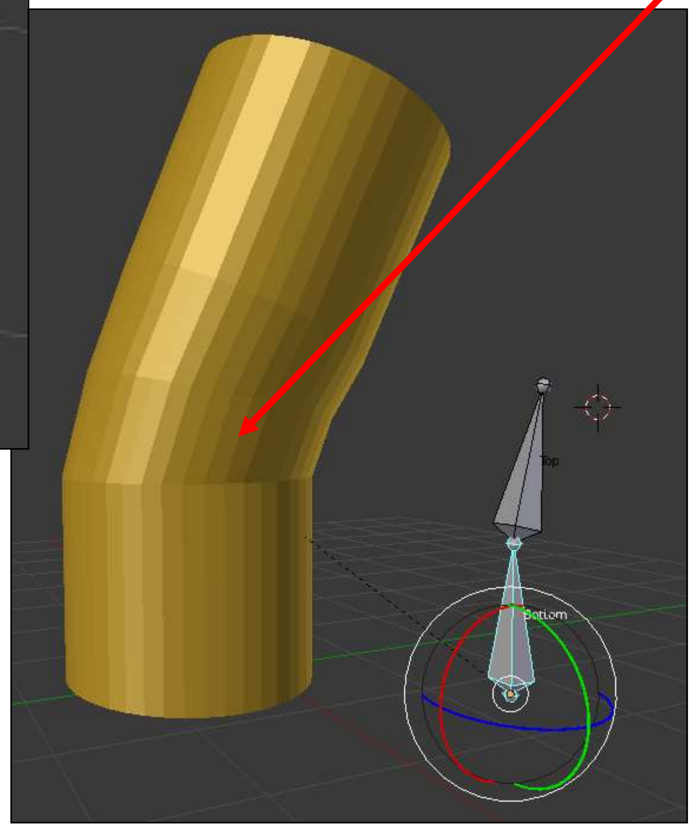


anim2.mp4

Rigging Animation

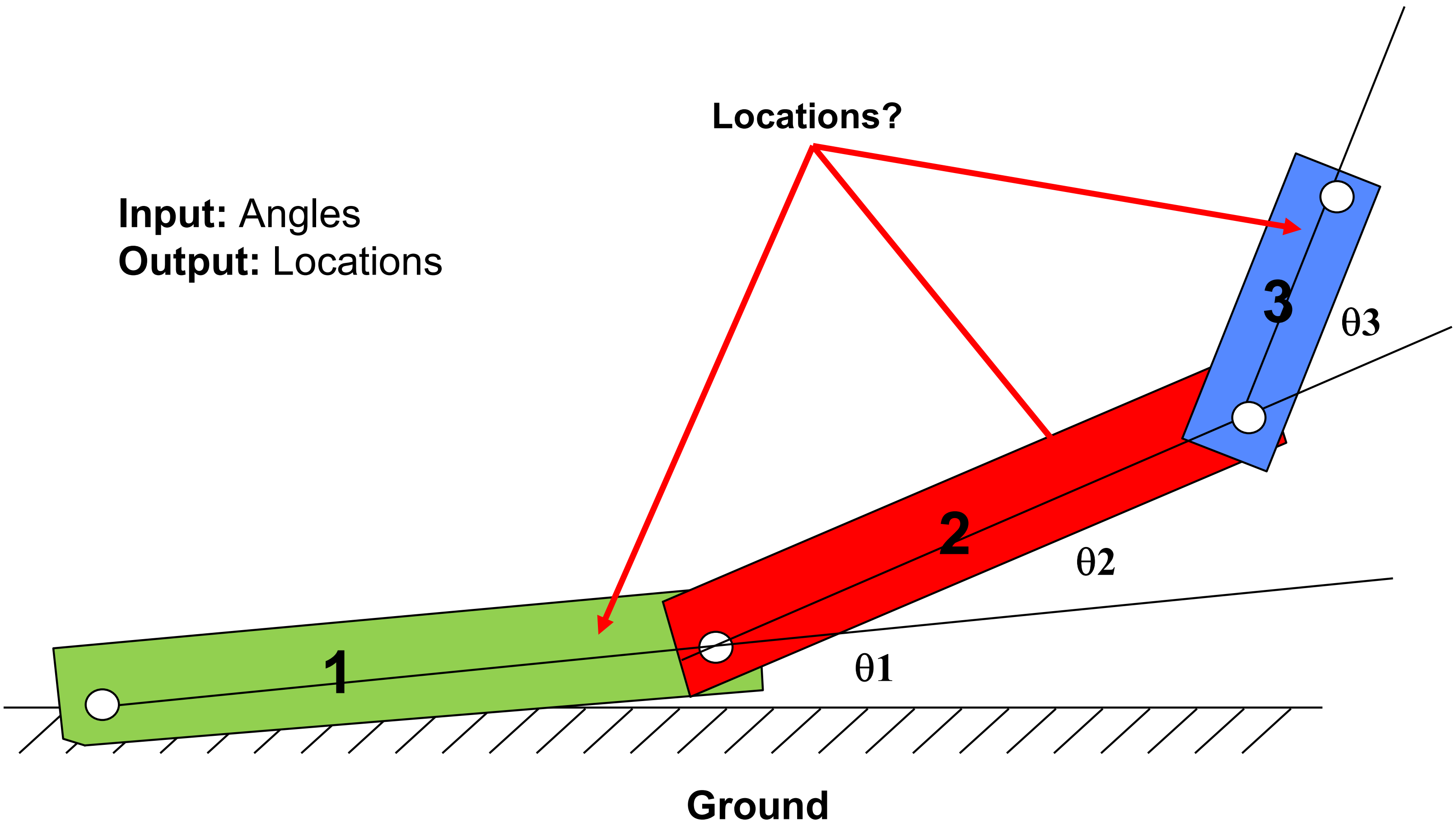
Use an ***armature*** to control the movement of groups of vertices



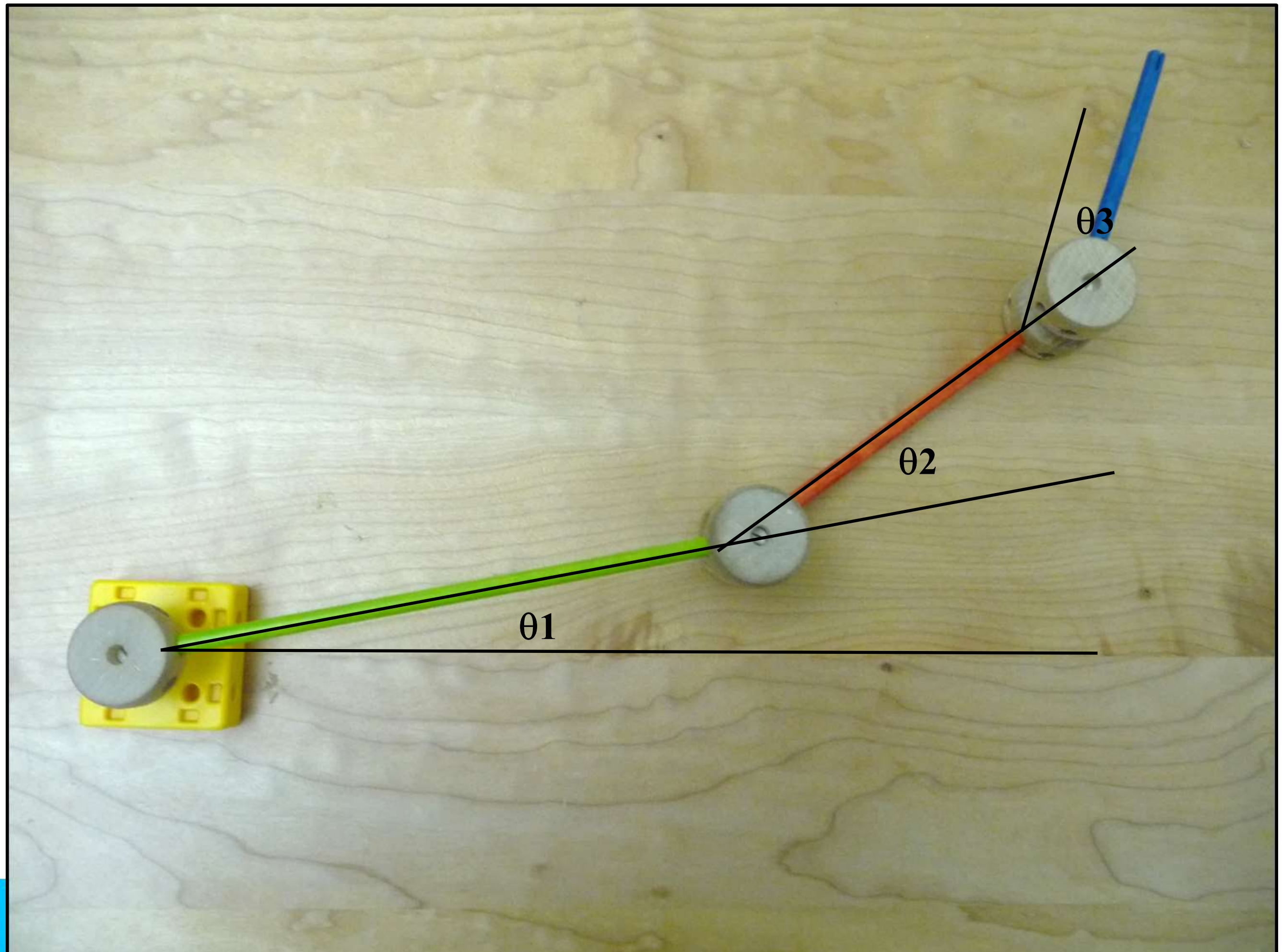


Forward Kinematics: Transformation Hierarchies

Input: Angles
Output: Locations



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

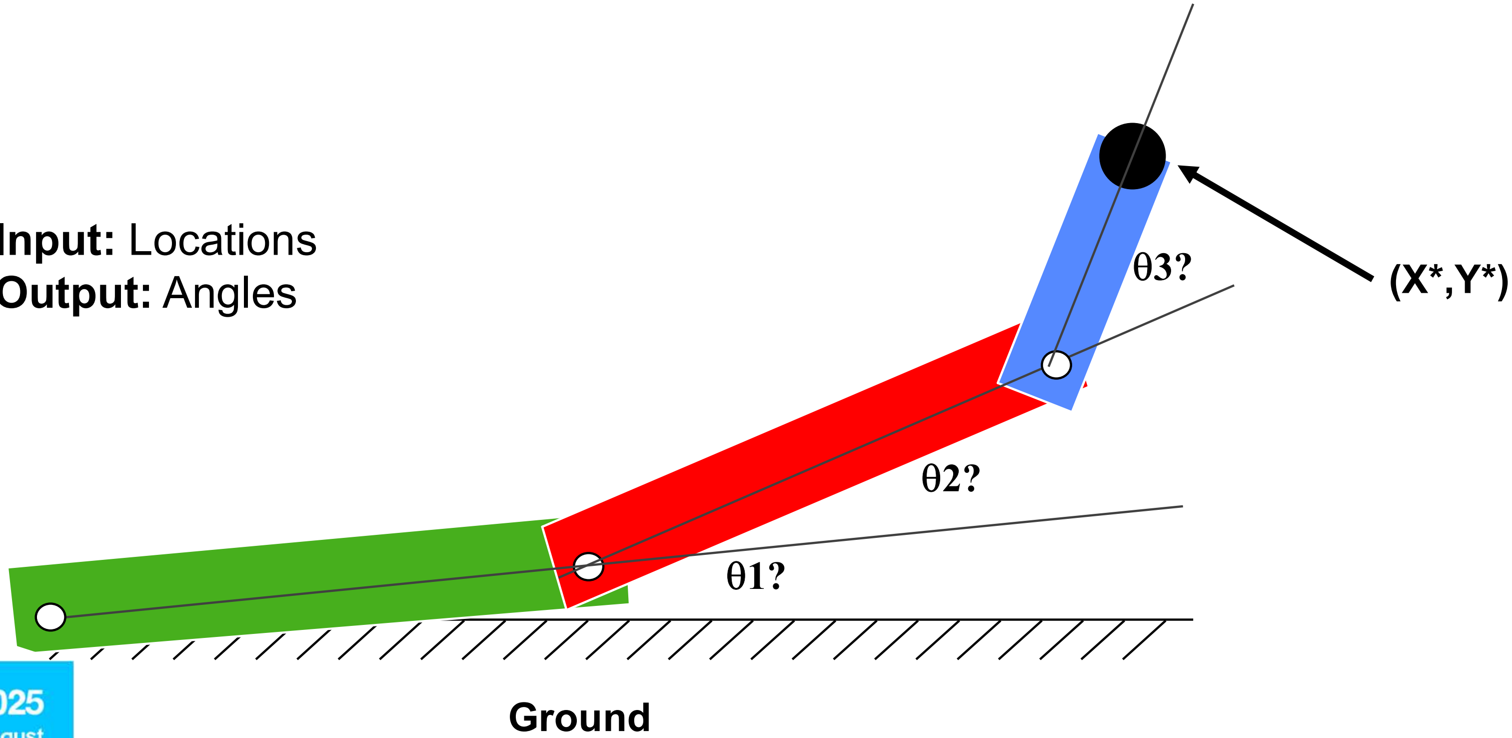


Inverse Kinematics

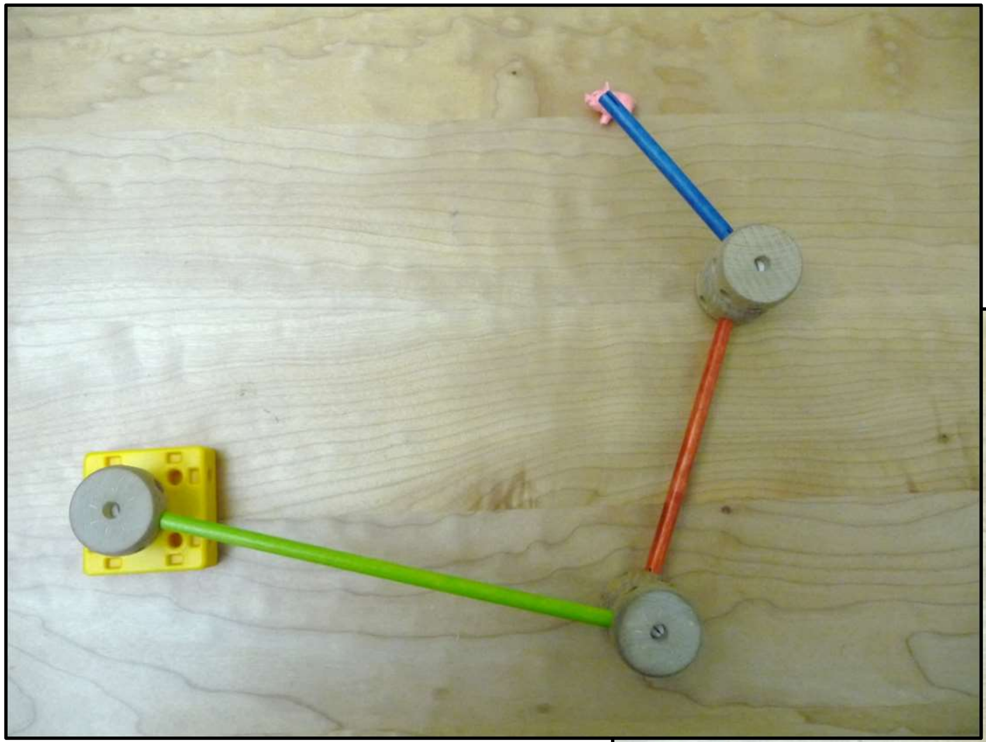
Forward Kinematics solves the problem "if I know the link transformation parameters, where are the links?".

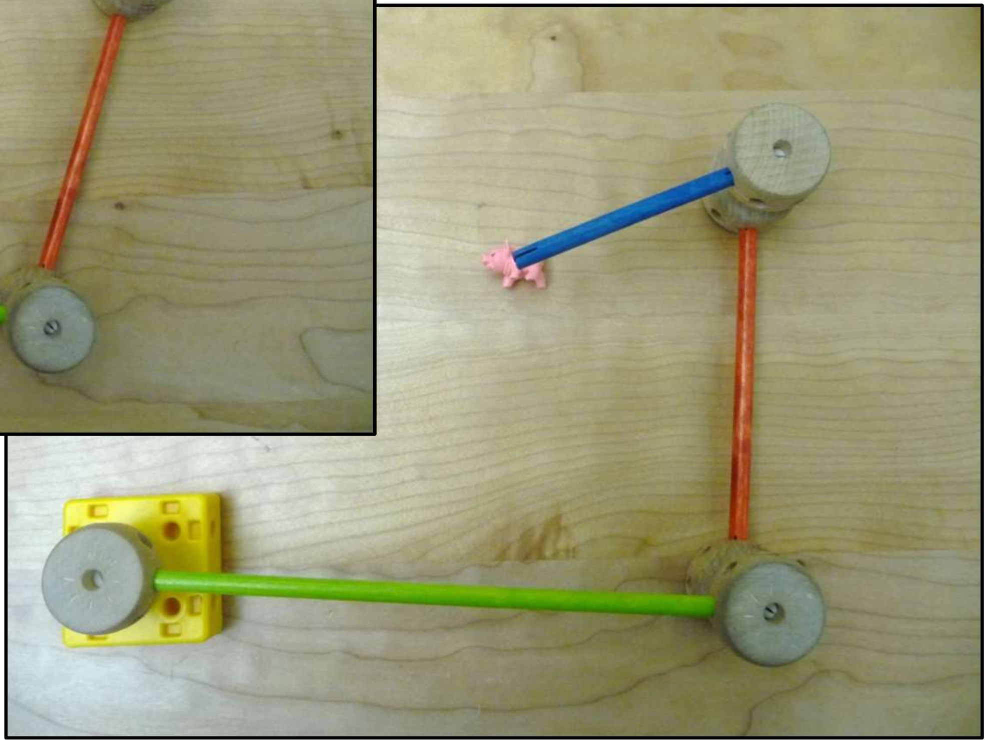
Inverse Kinematics (IK) solves the problem "If I know where I want the end of the chain to be (X*,Y*), what transformation parameters will put it there?"

Input: Locations
Output: Angles

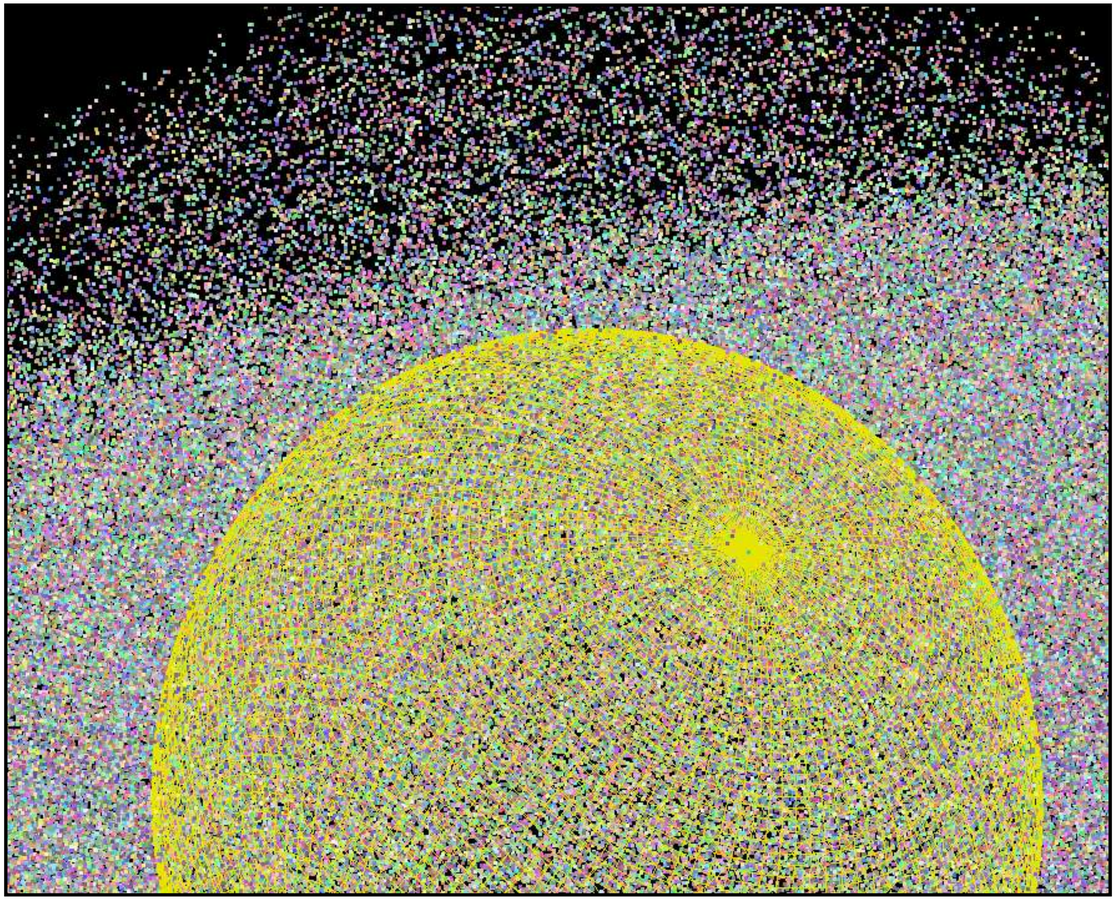


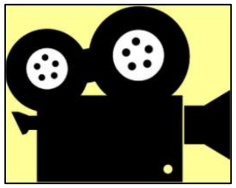
Inverse Kinematics (IK): Things Need to Move – What Parameters Will Make Them Do That?





Particle Systems:
A Cross Between Modeling and Animation?



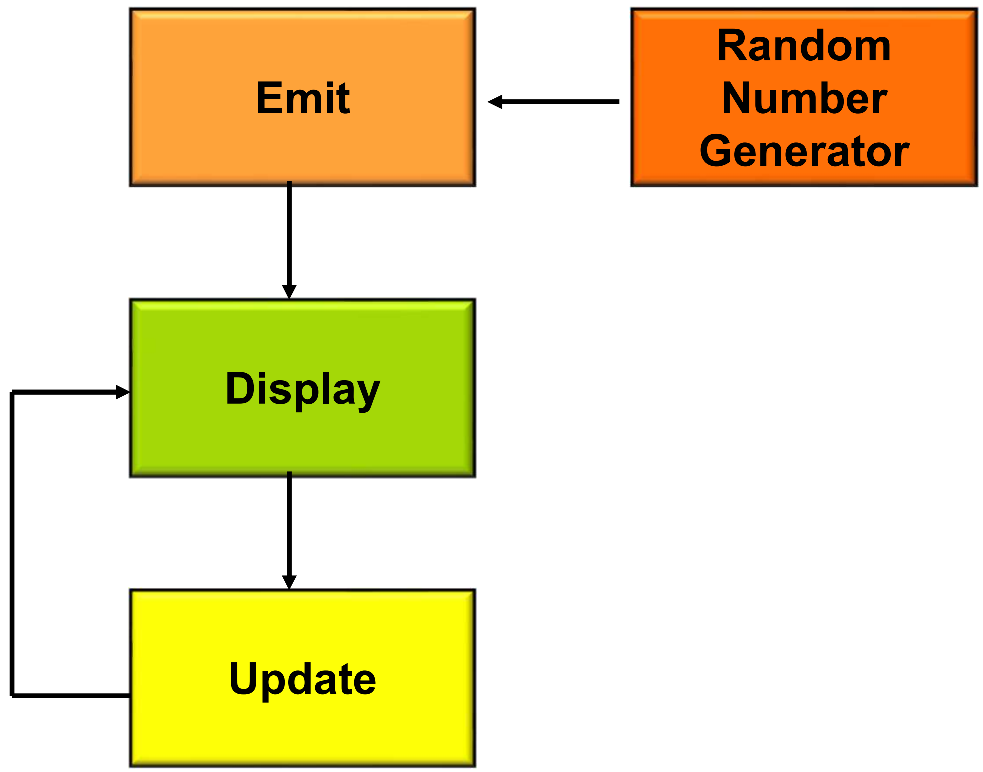


gpu_particles.mp4

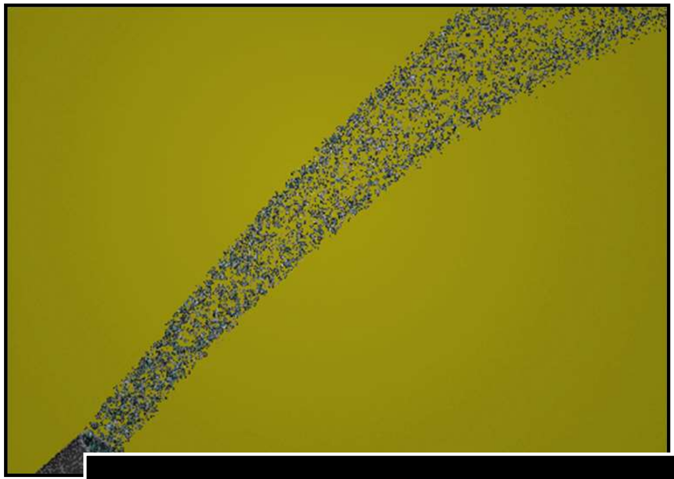
Check out this movie! These are particles animated on a GPU.

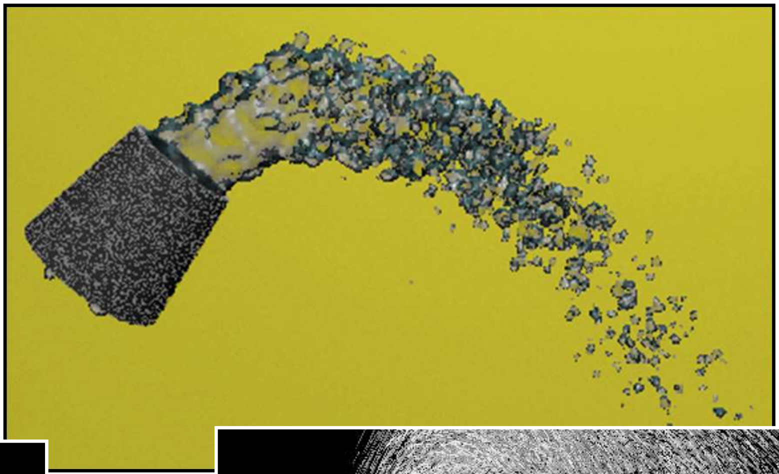
Particle Systems:
A Cross Between Modeling and Animation?

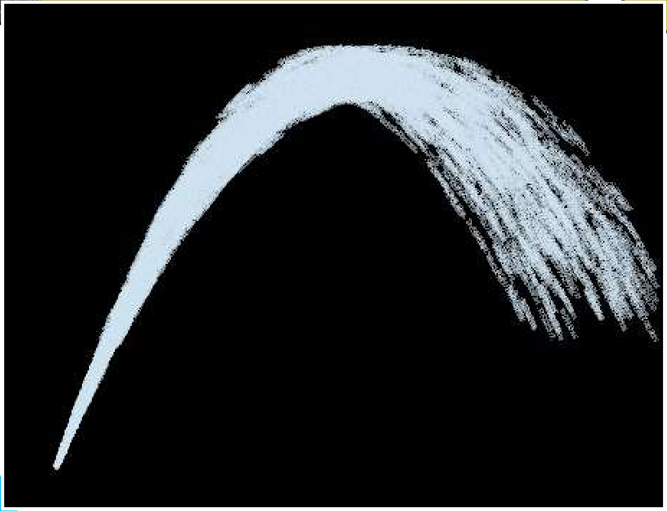
The basic process is:

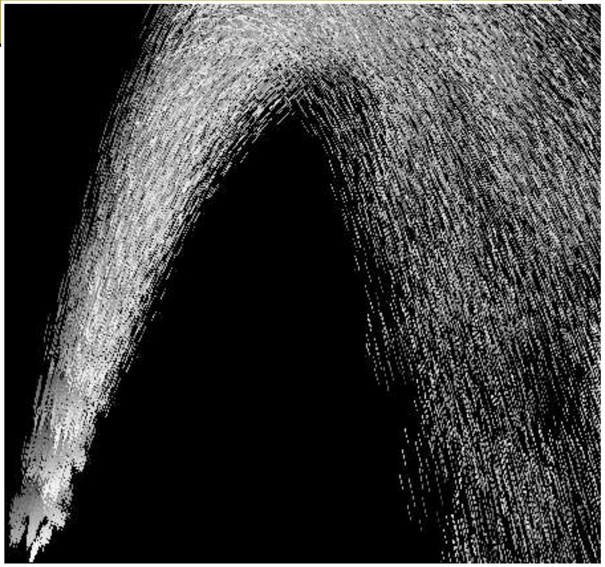


Particle Systems Examples



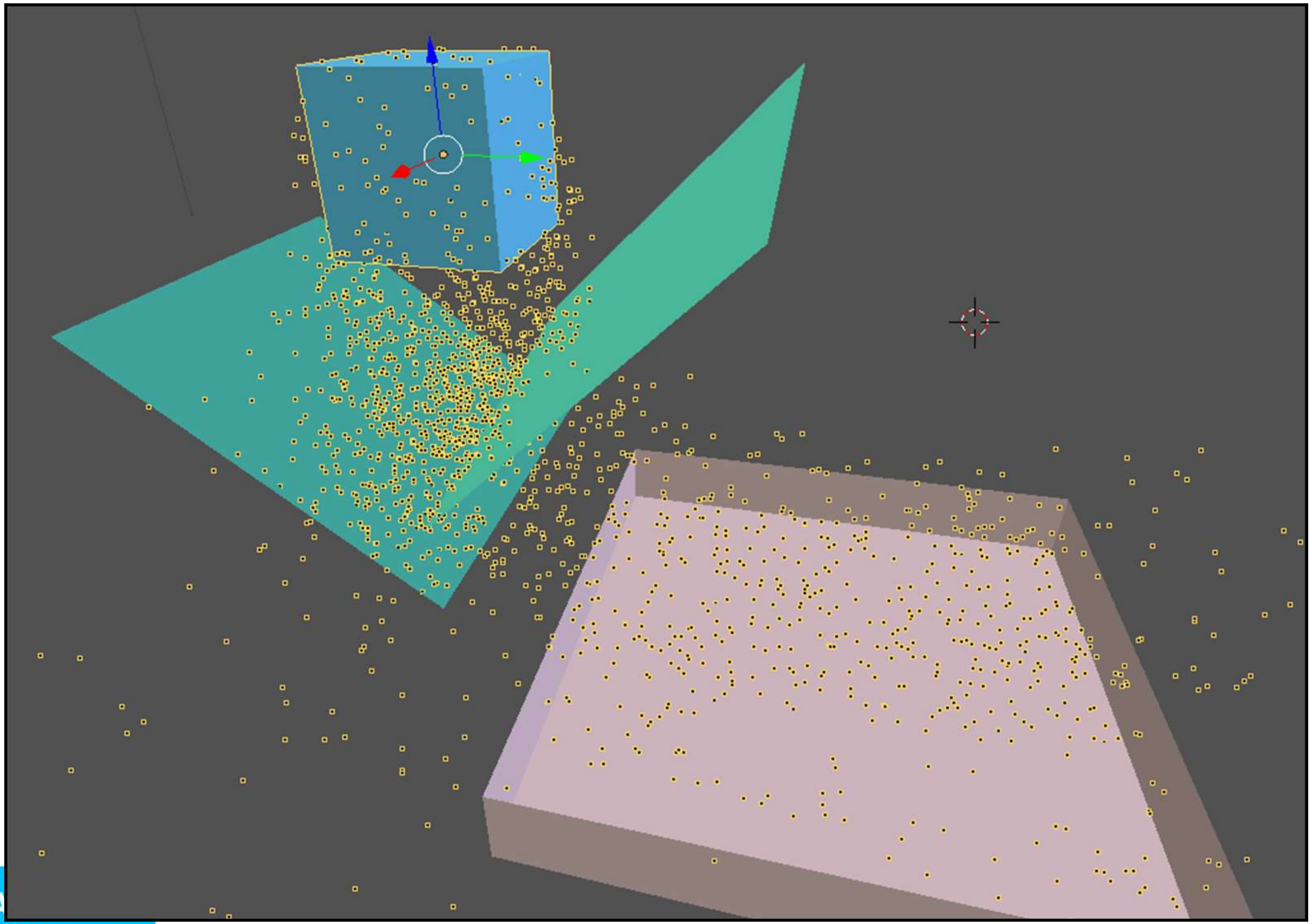


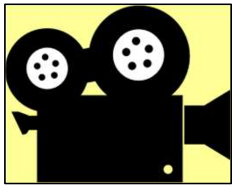




Images: Chuck Evans, used by permission

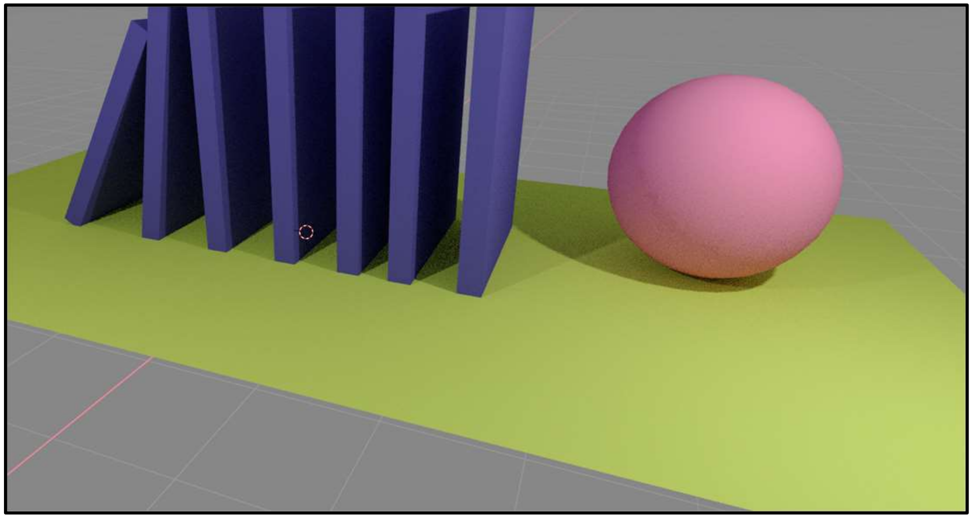
Particle Systems Examples

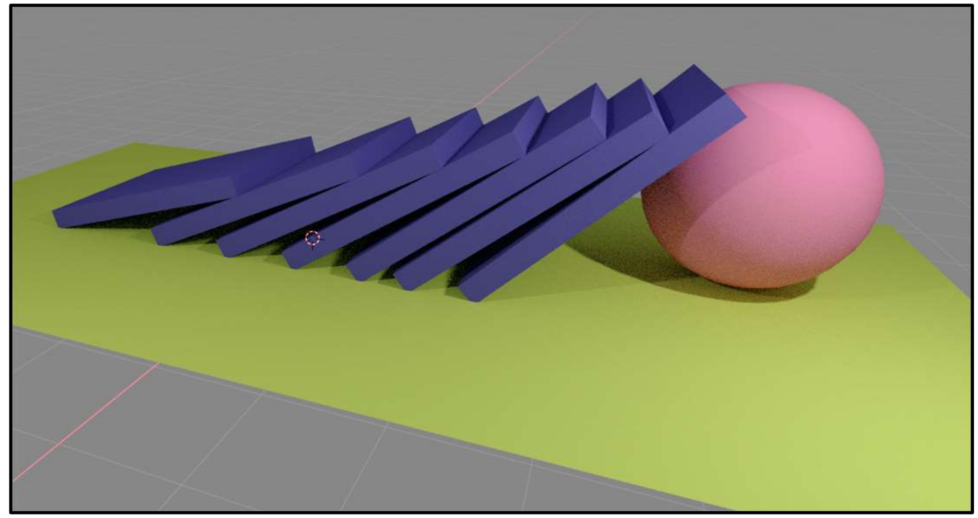




particles.mp4

Animating using Rigid-body Physics





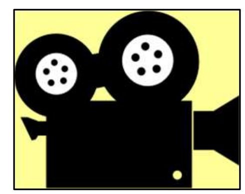
Newton's second law:

force = mass * acceleration

or:

acceleration = $\{\ddot{x}\}$ = force / mass

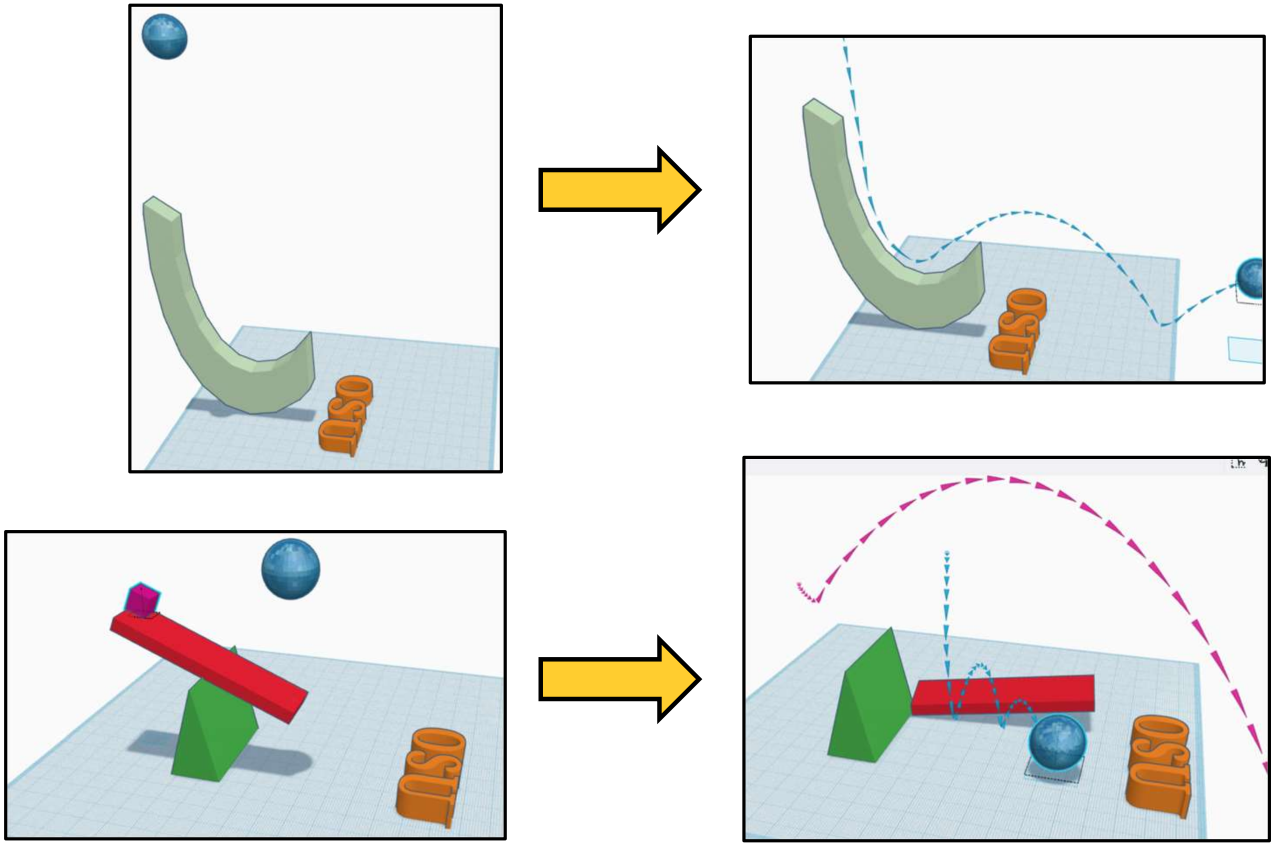
Newton's Second Law



dominos2.mp4

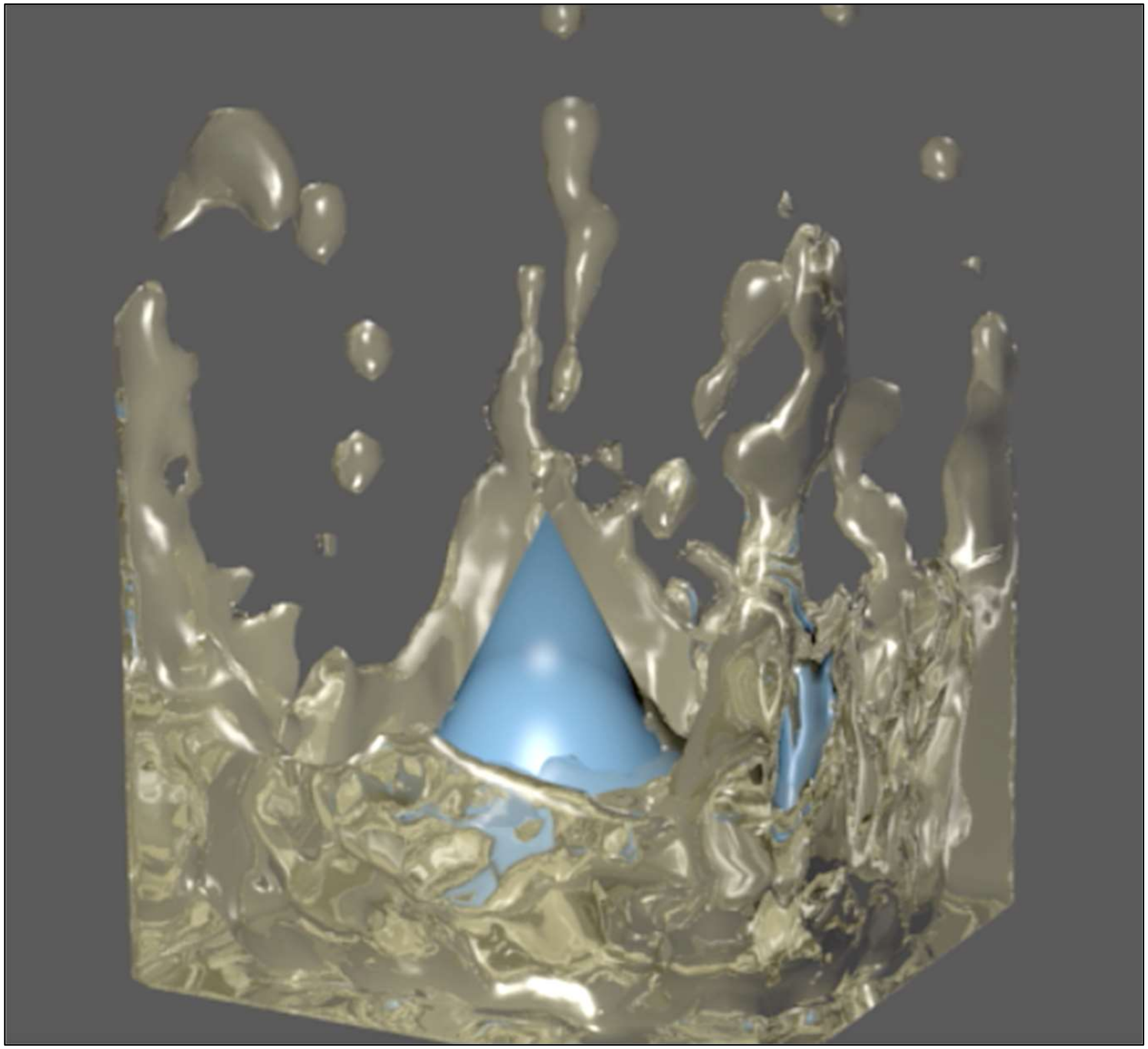
To make this work, you need to supply physical properties such as mass, center of mass, moment of inertia, coefficients of friction, coefficients of restitution, etc.

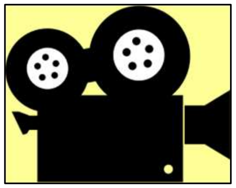
Even TinkerCad Now Has Rigid Body Physics Animation



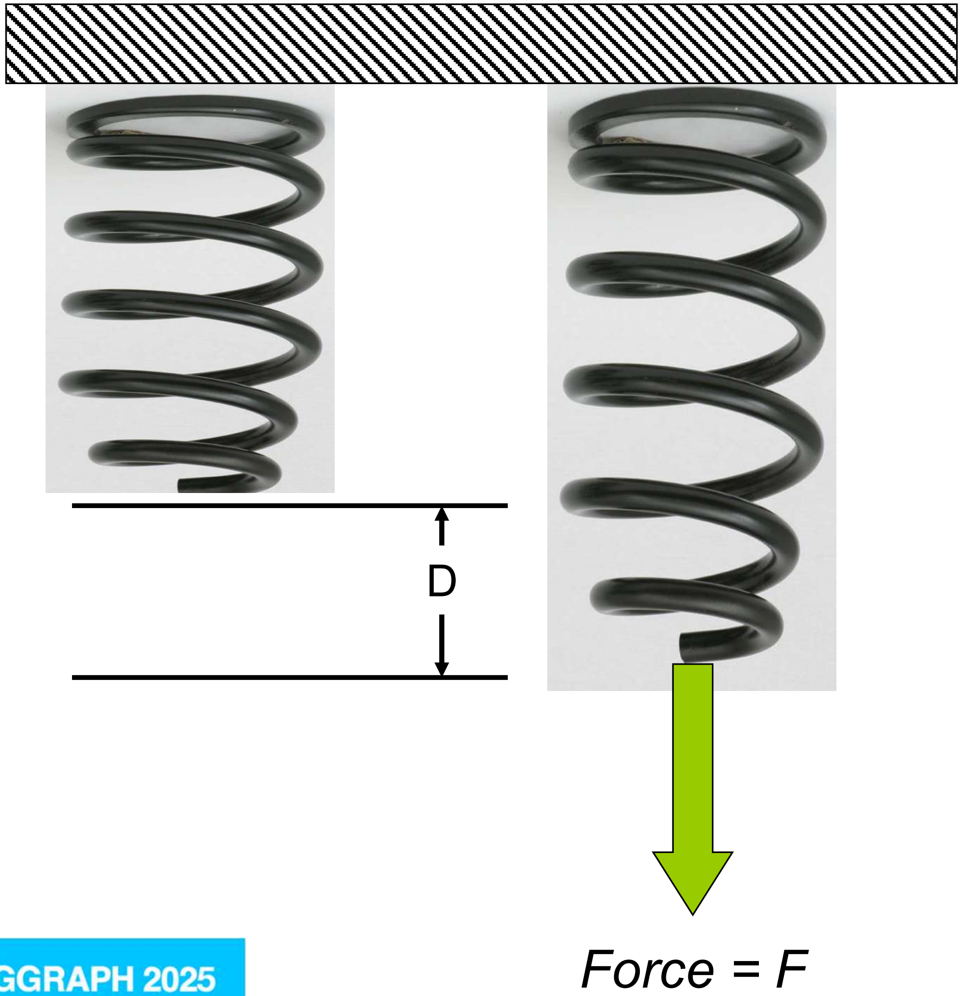
Want to experiment with TinkerCad physics animation for free?
Want some notes to get you started?
http://cs.oregonstate.edu/~mjb/tinkercad

Animating using Fluid Physics





fluid.avi



k = ***spring stiffness*** in newtons/cm or pounds/inch

If you know the force, the distance the spring stretches or compresses will be:

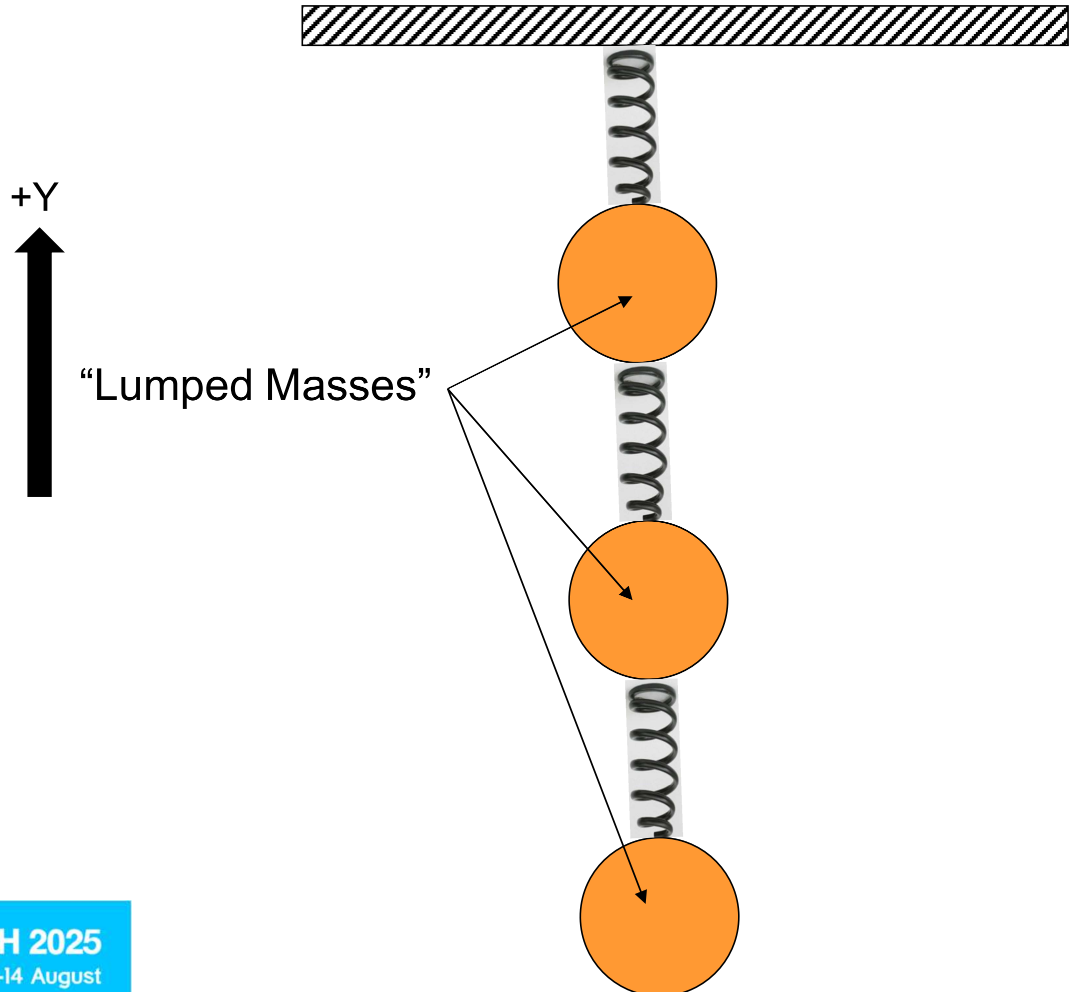
$$D = \frac{F}{k}$$

Or, if you know the distance the spring stretches or compresses, the force exerted by the spring will be:

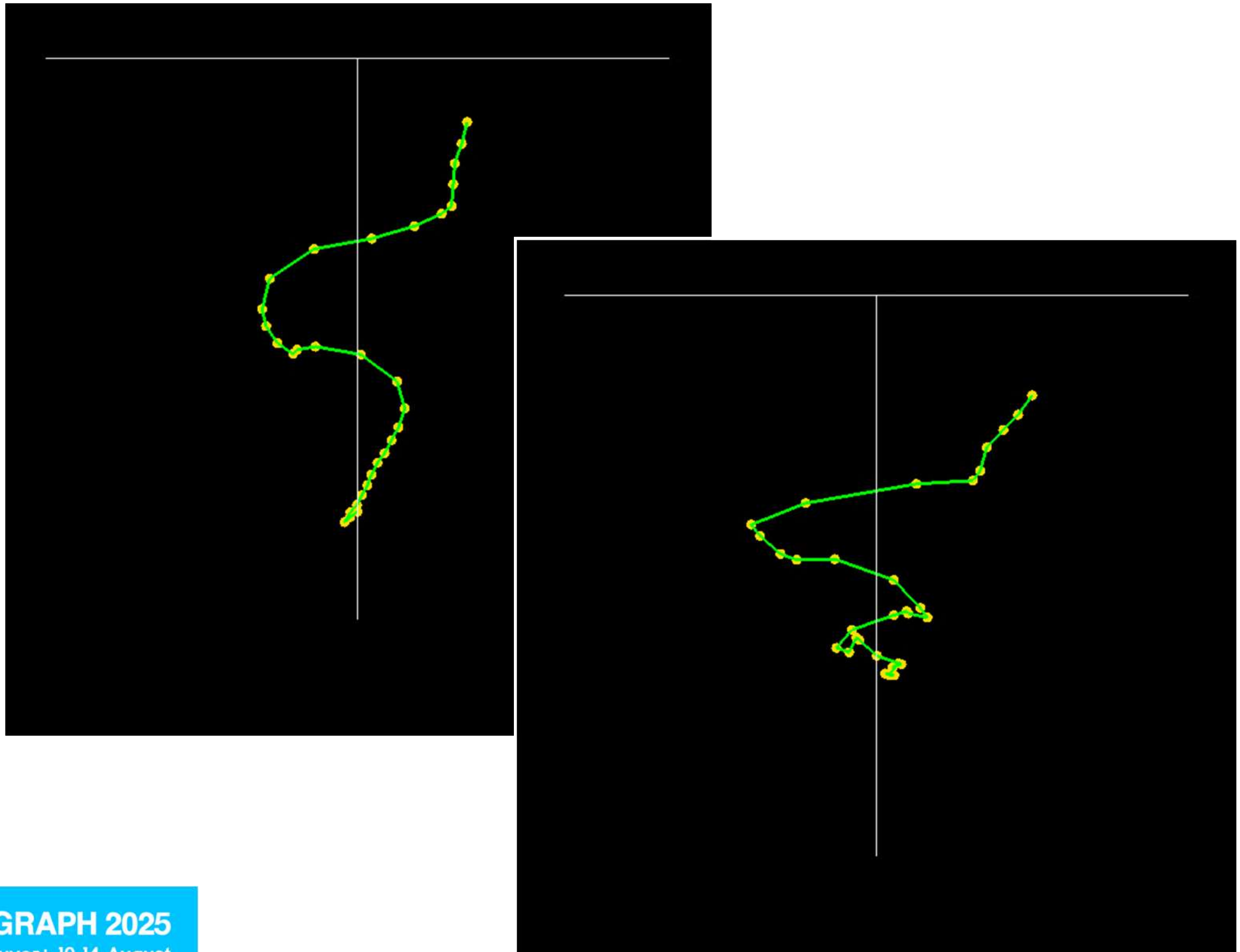
$$F = kD$$

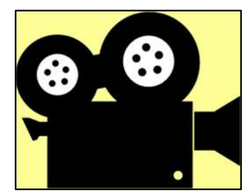
This is known as *Hooke's Law*

Animating using the Physics of a Mesh of Springs



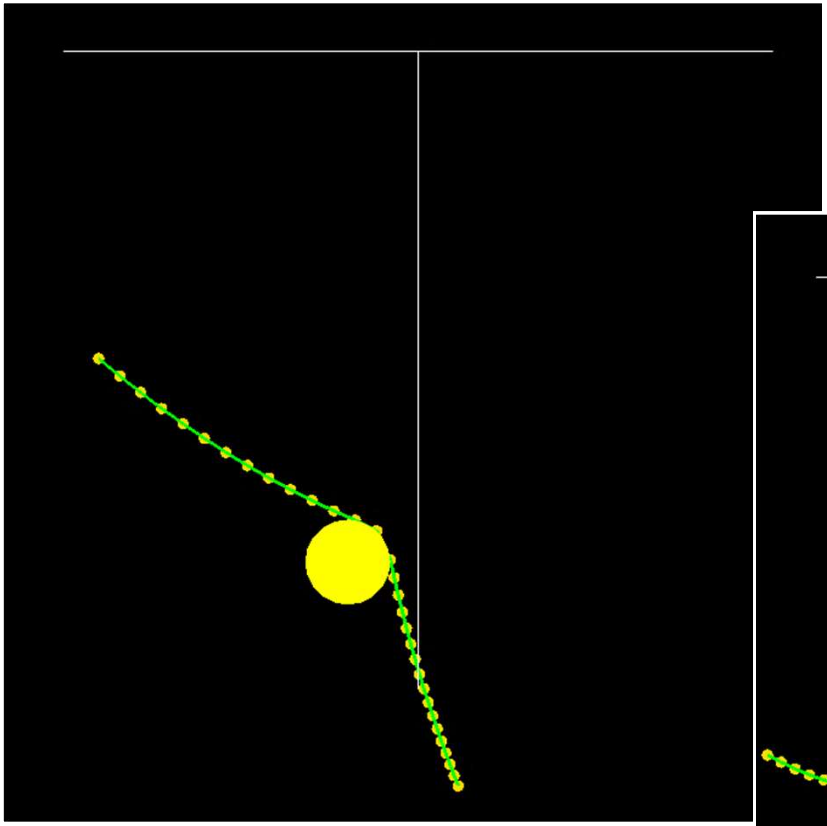
Simulating a Bouncy Chain

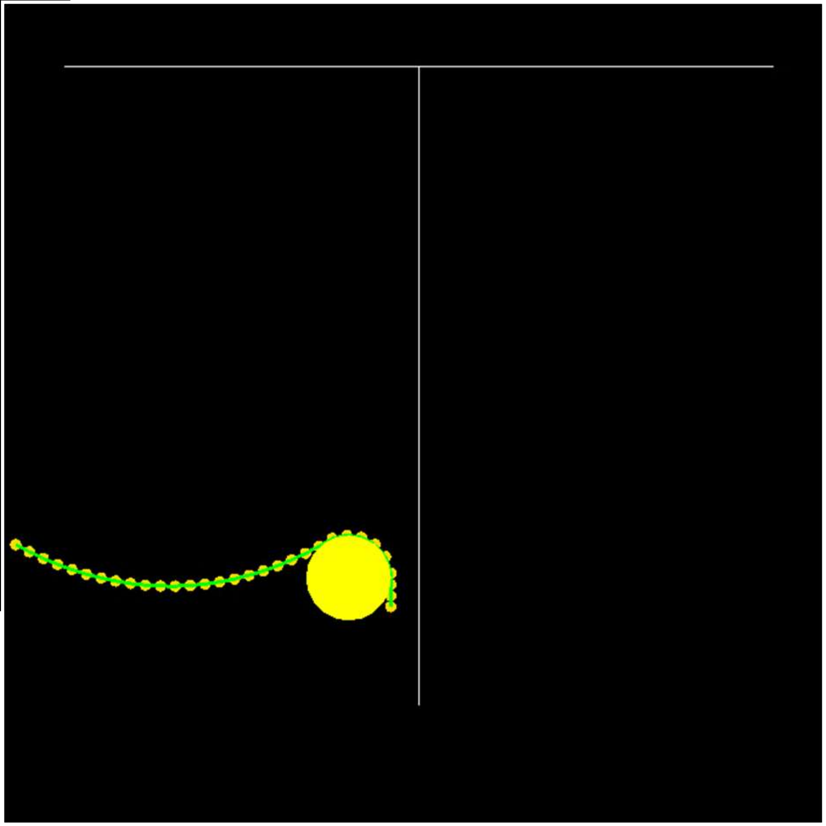




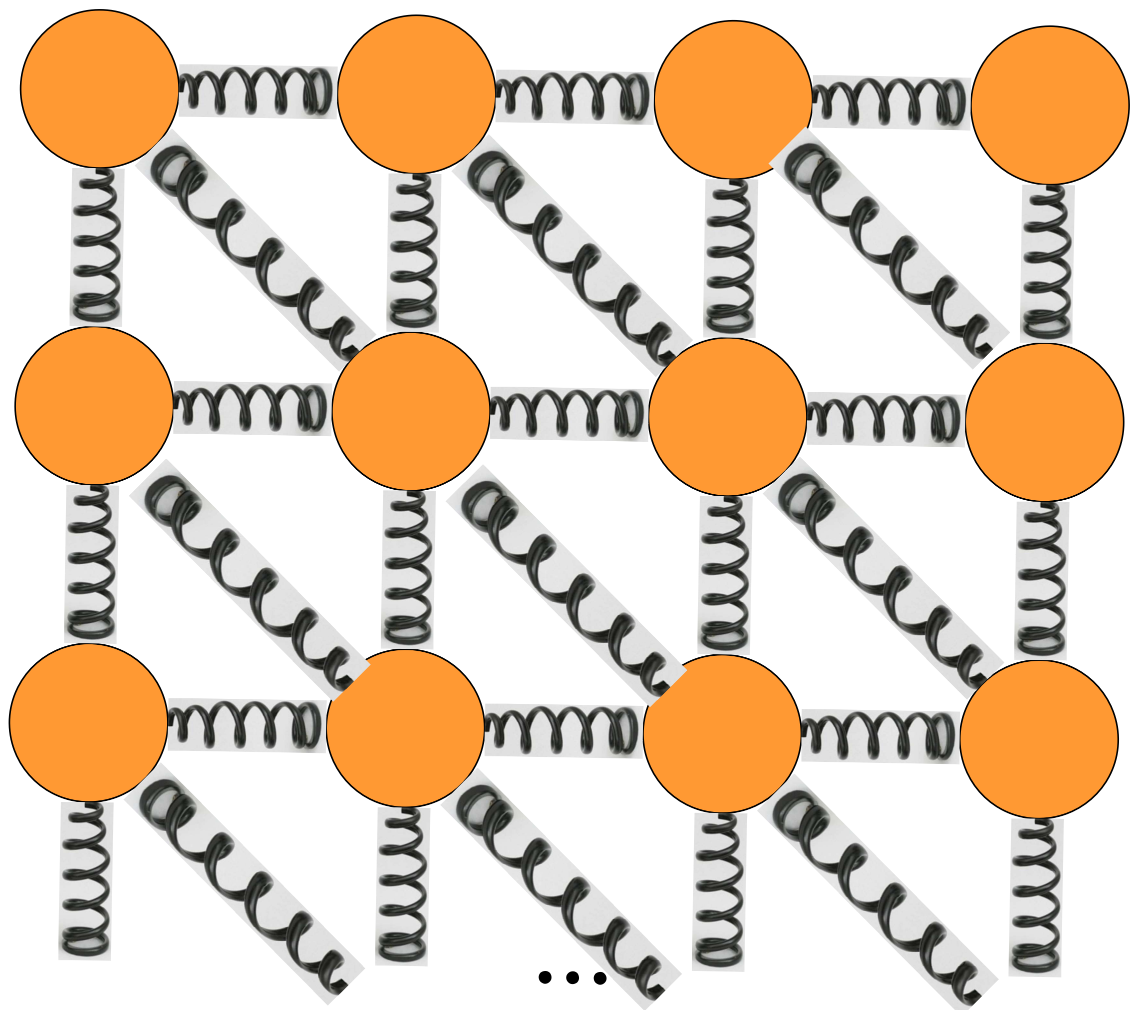
chain.mp4

Placing a Physical Barrier in the Scene



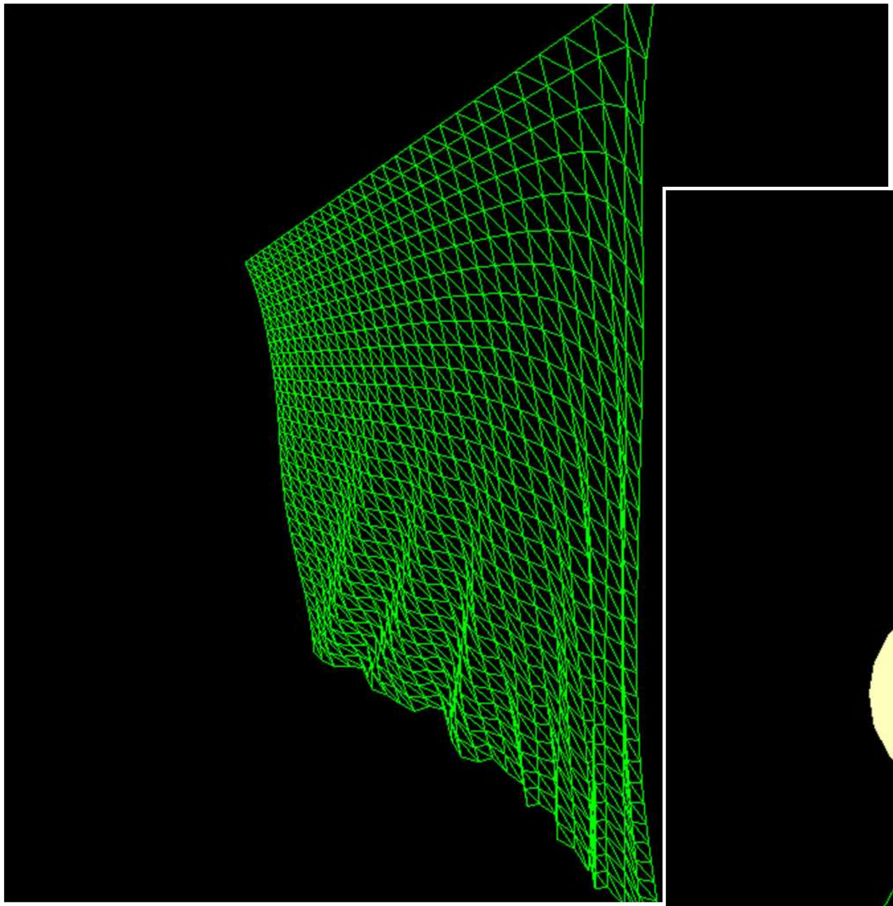


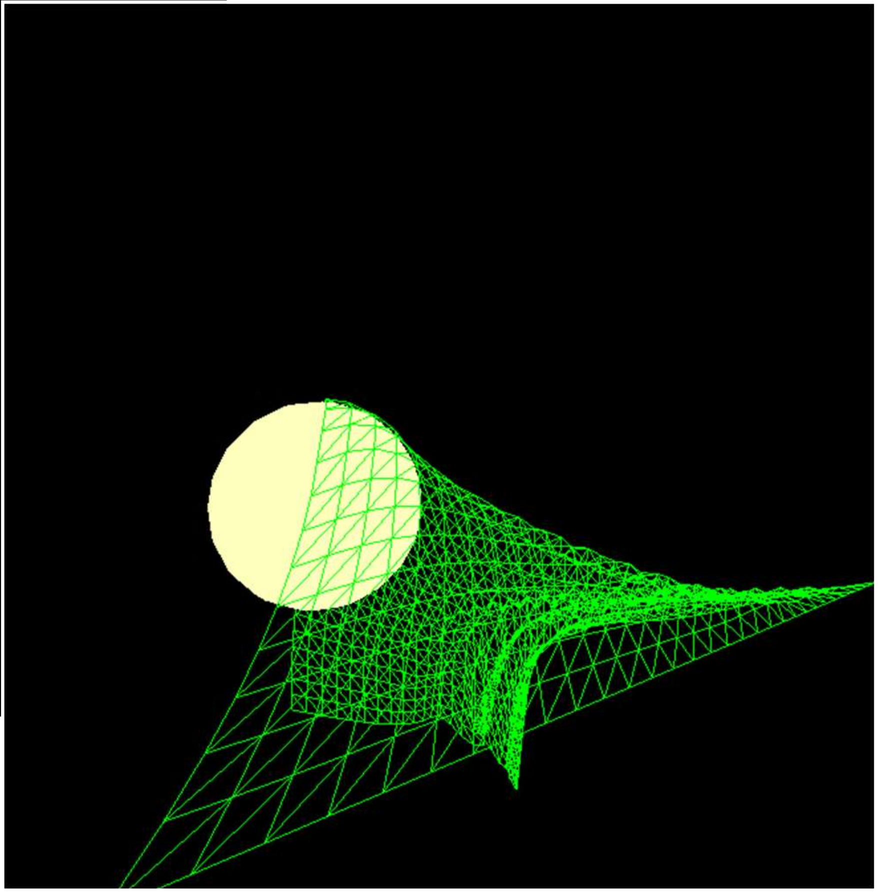
Animating Cloth



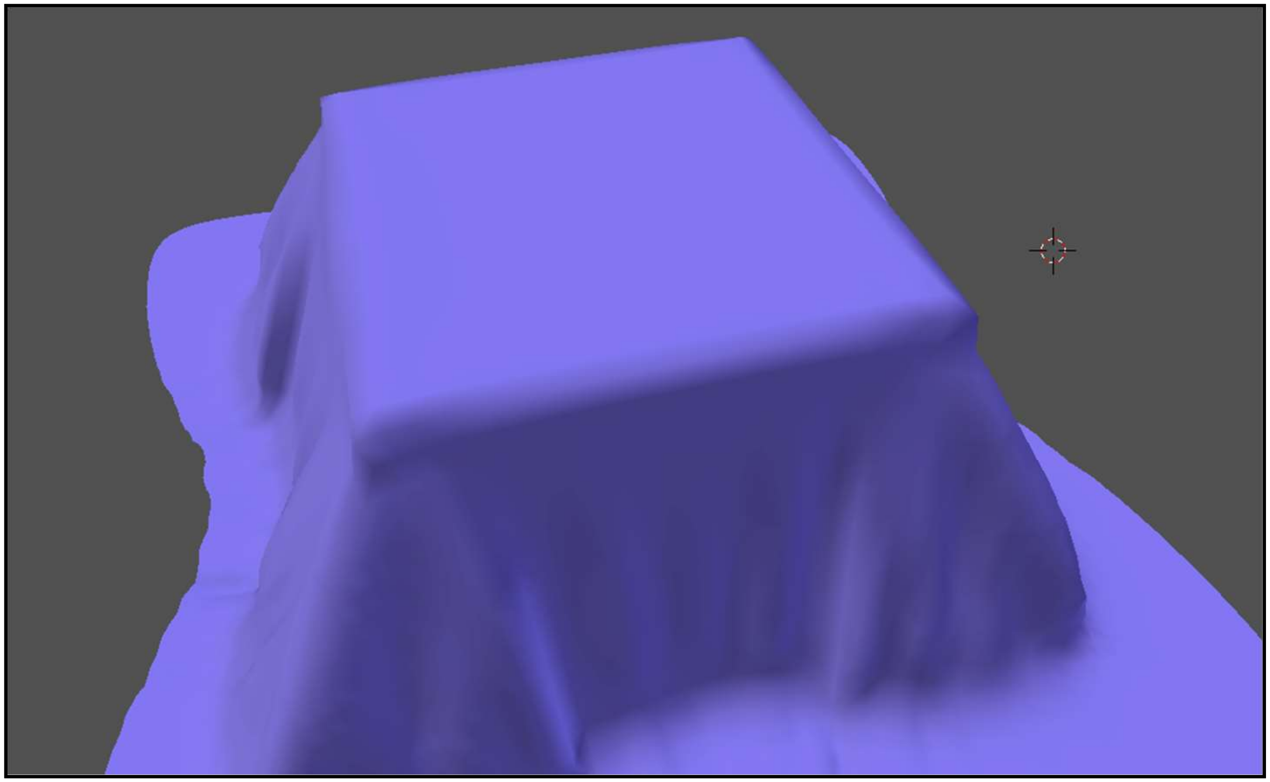
SIGGRAPH 2025
Vancouver+ 10-14 August

Cloth Example





Cloth Example



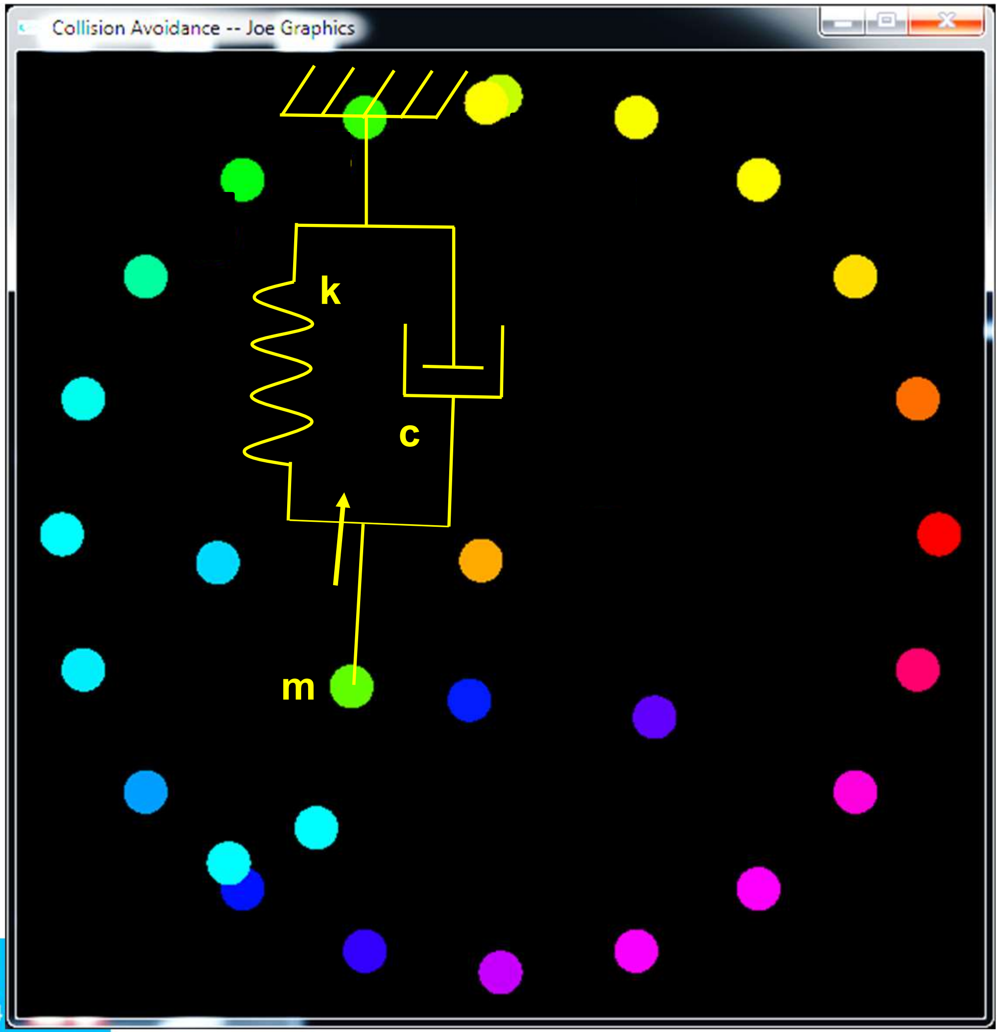
cloth.mp4

Want to experiment with a free cloth animation program?
Want some notes to get you started?
http://cs.oregonstate.edu/~mjb/blender

Functional Animation – "Fake Physics"

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

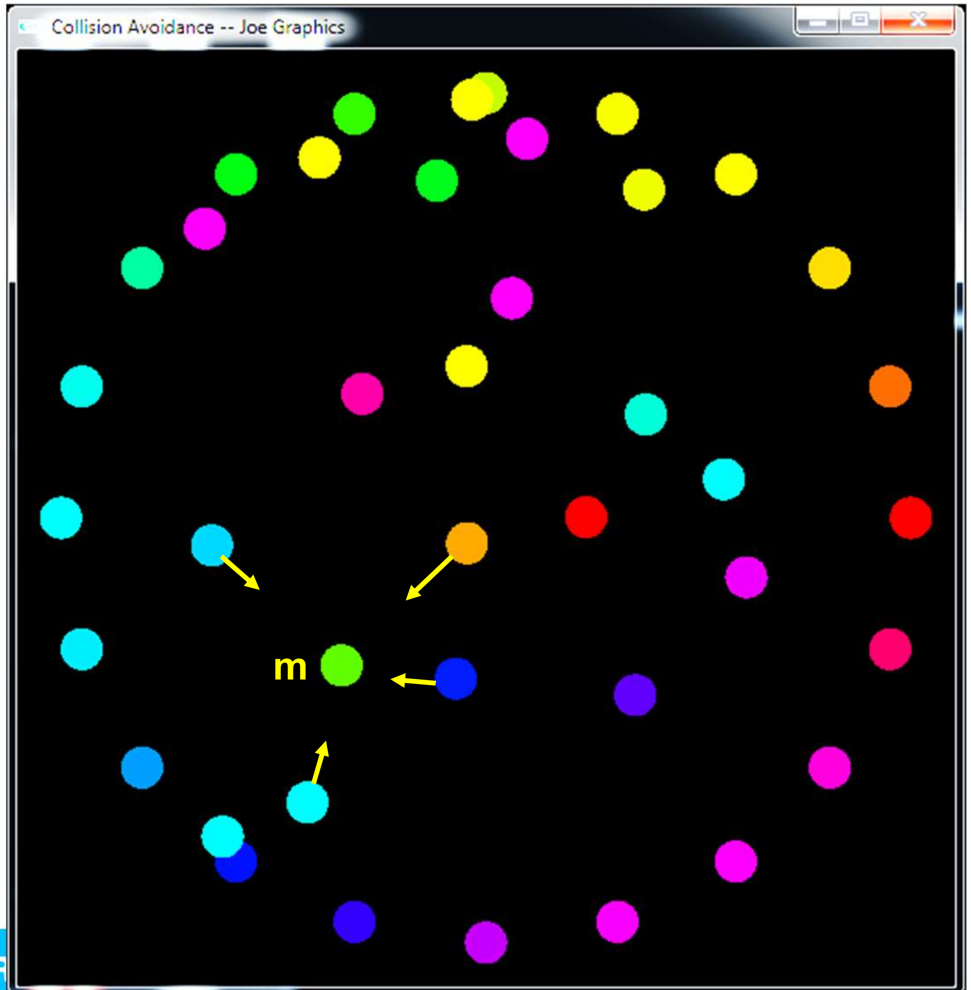
Functional Animation:
Make the Object *Want* to Move Towards a Goal Position . . .



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

Actual equation of motion for a spring-mass-damper system

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



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

Repulsion Coefficient

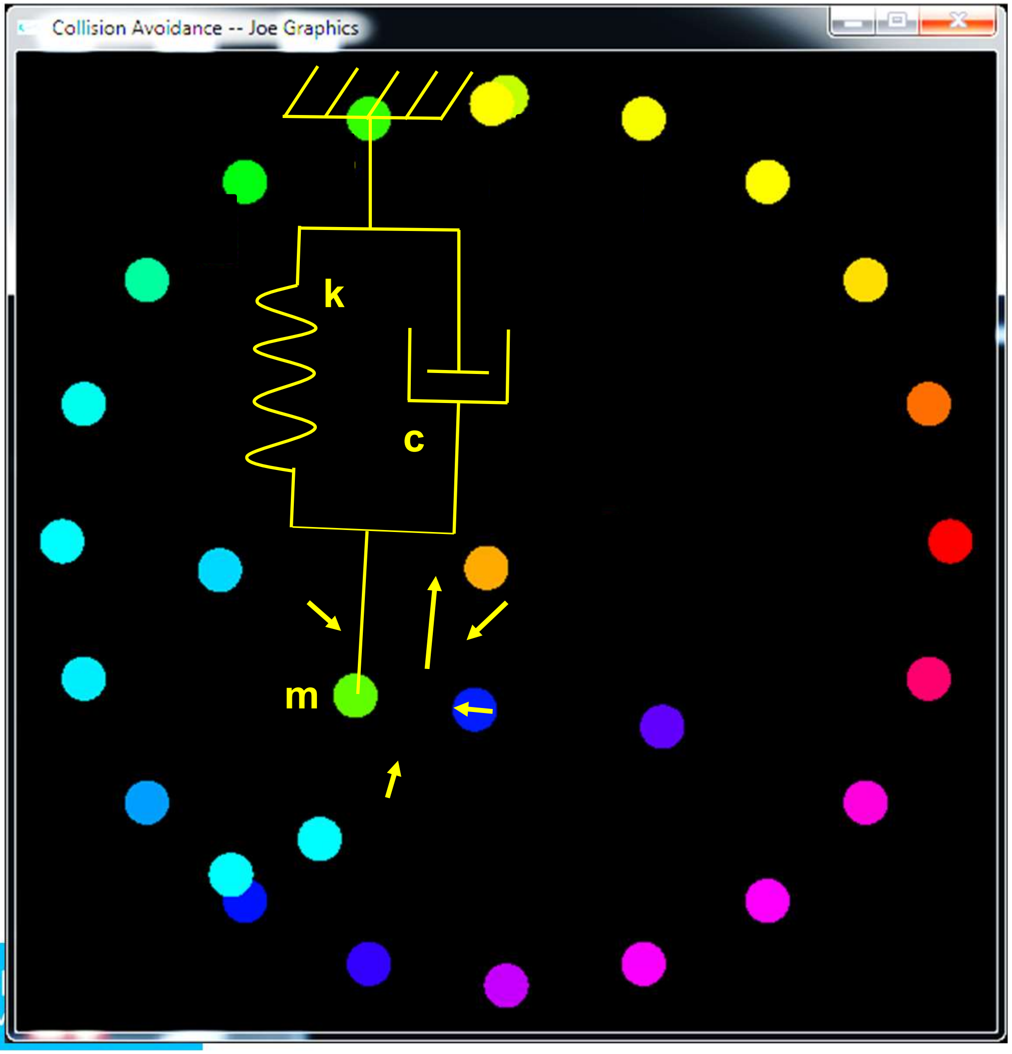
$$F_{repulsive} = \frac{C_{repulse}}{d^{Power}}$$

Distance between the boundaries of the 2 bodies

Repulsion Exponent

Fake equation of motion for two masses trying to push each other away – I just made this up…

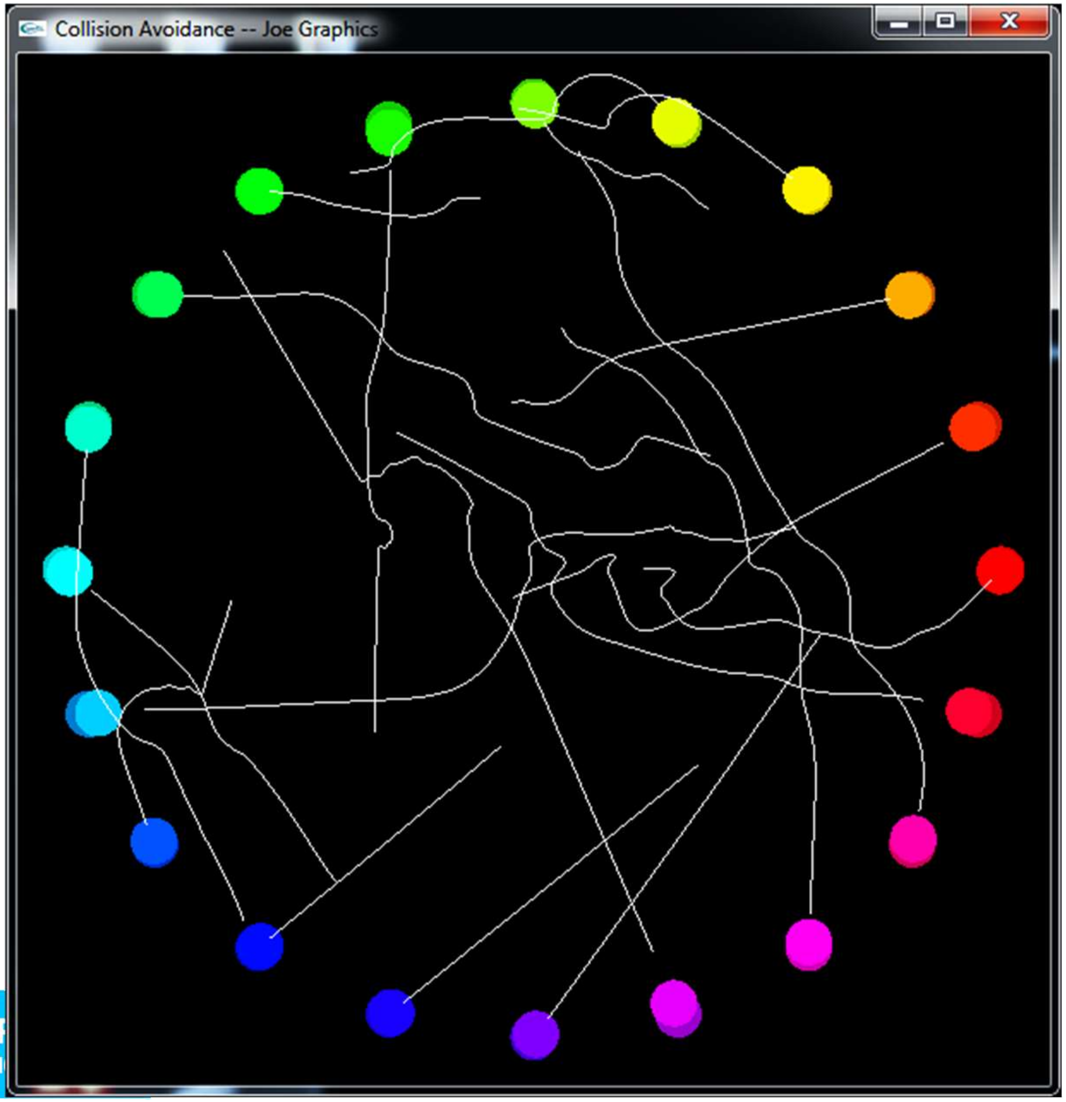
Total Goal – Make the Free Body Move Towards its Final Position While Being Repelled by the Other Bodies



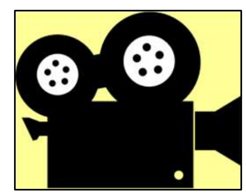
$$m\ddot{x}+c\dot{x}+kx=\sum F=\sum F_{repulsive}$$

If we set the actual and fake equations in motion, what will happen?

Functional Animation

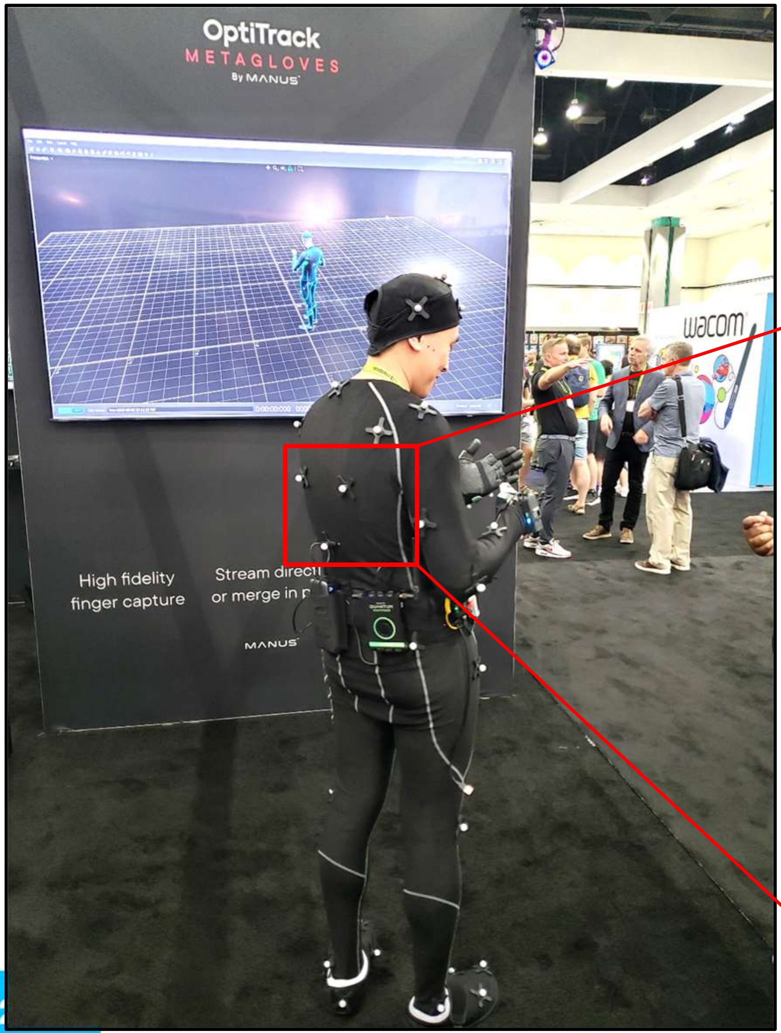


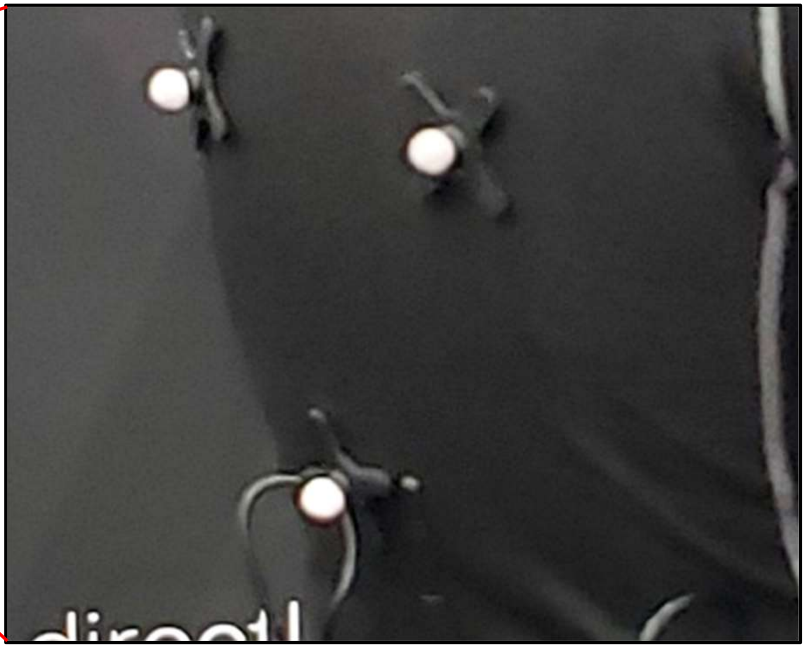
We get a collection of objects, each trying to move to a target, but without colliding with each other.



avoid.mp4

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





Natural Point

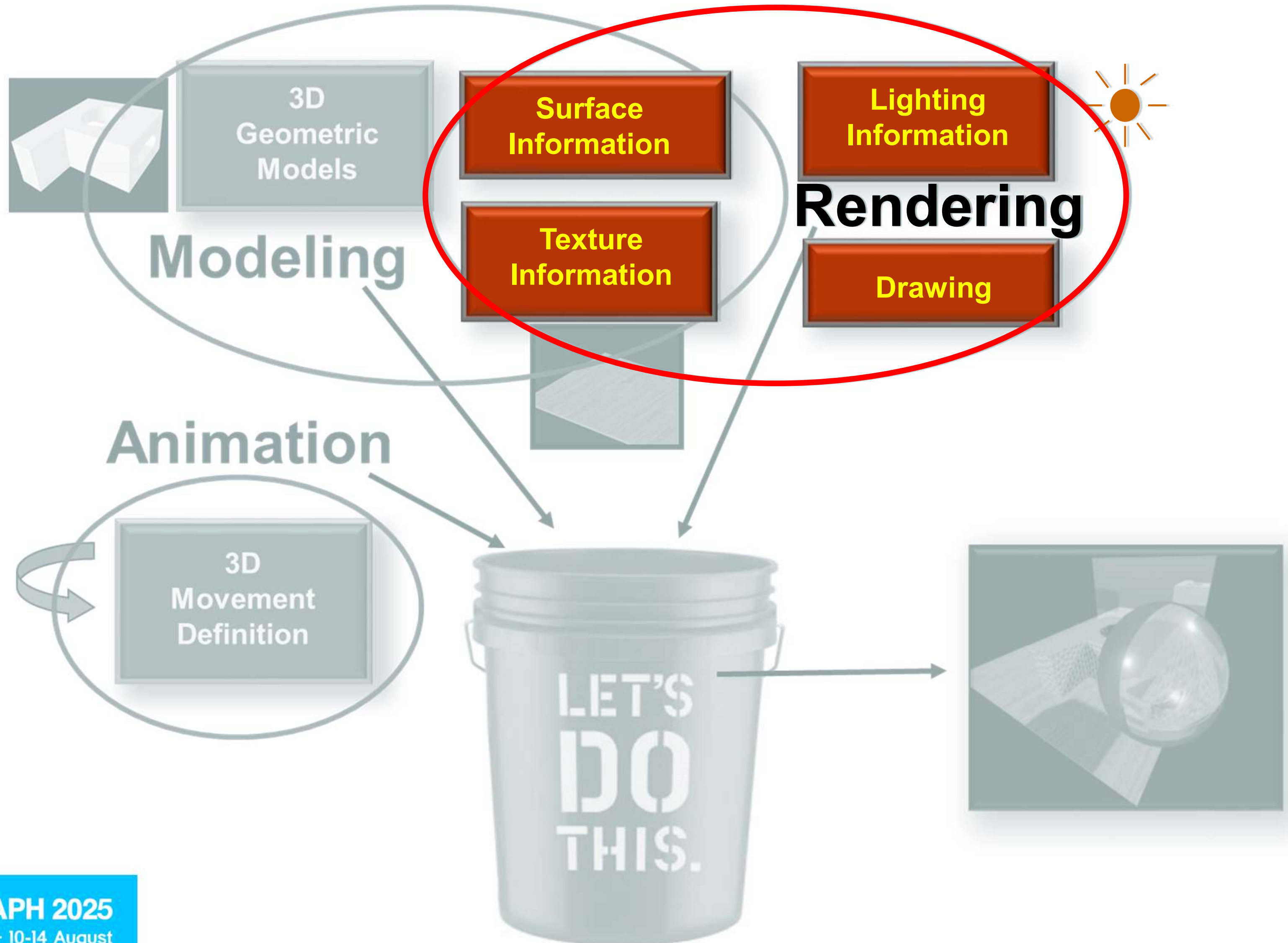
Photos: Mike Bailey

Even Animals can be MoCapped (if you dare…)

https://www.youtube.com/watch?v=zyq_LQrHpoo

Photo courtesy of: DIGIC Services' Mocap Studio, used by permission

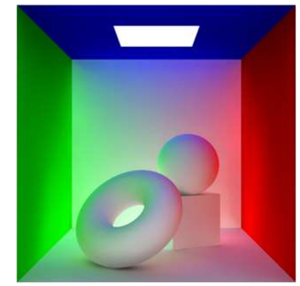
Creating an image of your scene

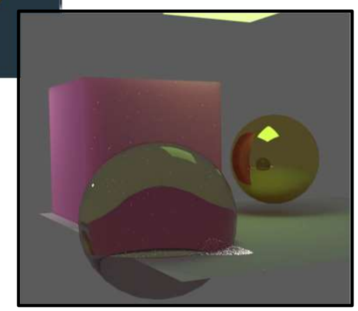

3D
Geometric
Models
Modeling
Surface
Information
Texture
Information
Lighting
Information
Rendering
Drawing
Animation
3D
Movement
Definition
LET'S
DO
THIS.

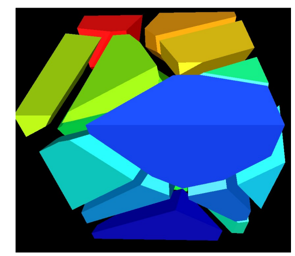
Rendering

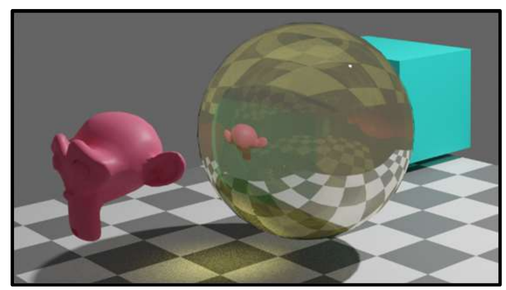
Rendering is the process of creating an image of geometric models. Again, there are questions you need to ask first:

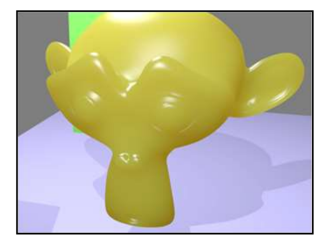
- Why am I doing this?
- How realistic do I want this image to be?
- How much compute time do I want this to take?
- Do I need to take lighting into account?
- Does the illumination need to be global or will local do?
- Do I need to create shadows?
- Do I need to create reflections and refractions?









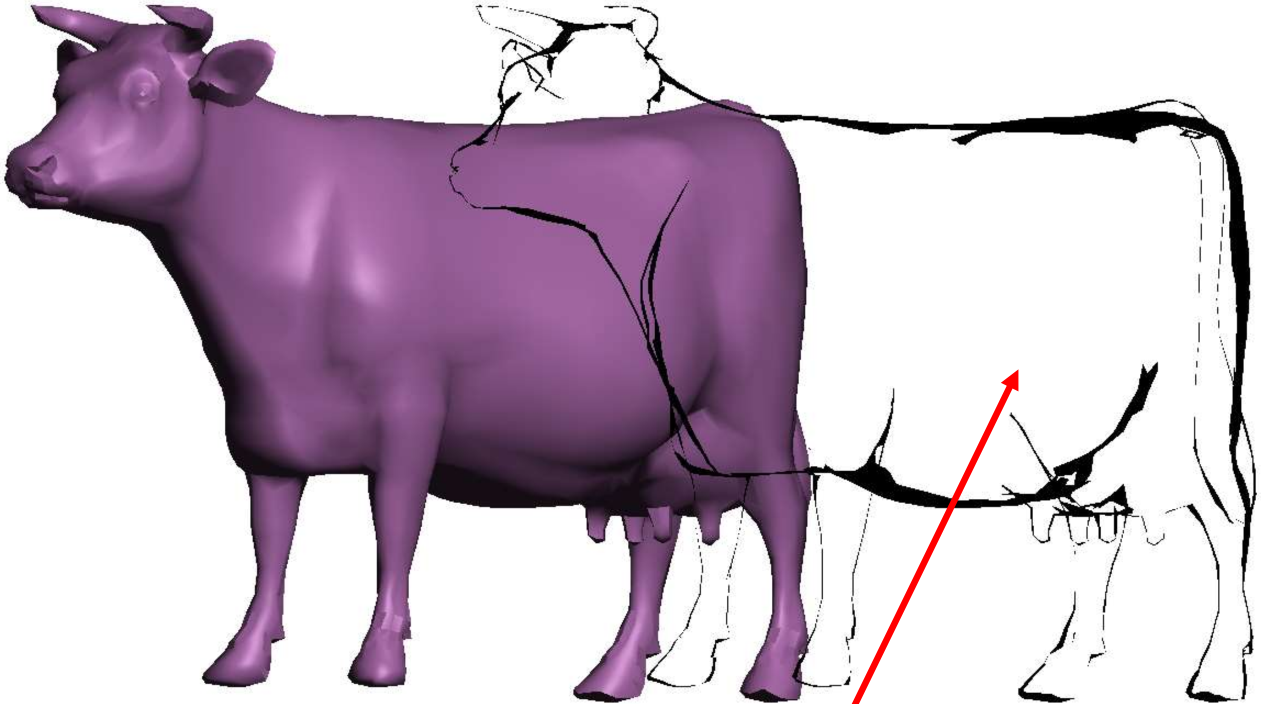


Want to experiment with a free rendering program?
Want some notes to get you started?
http://cs.oregonstate.edu/~mjb/blender

Non-Photorealistic Rendering: Toon Shading

Photo by Steve Cunningham, used with permission

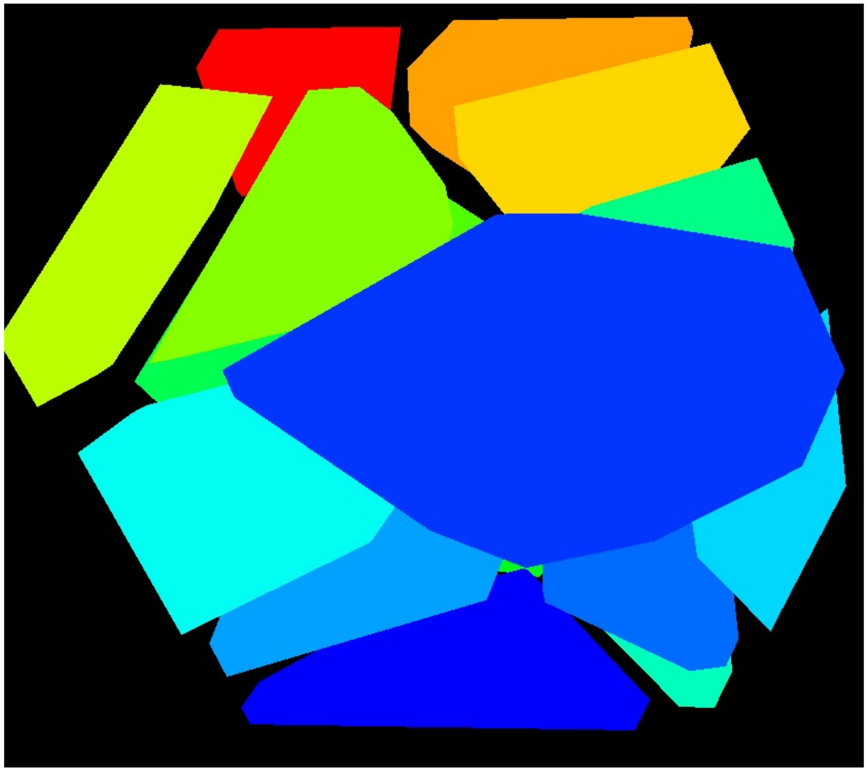
Non-Photorealistic Rendering: Silhouettes using Shaders



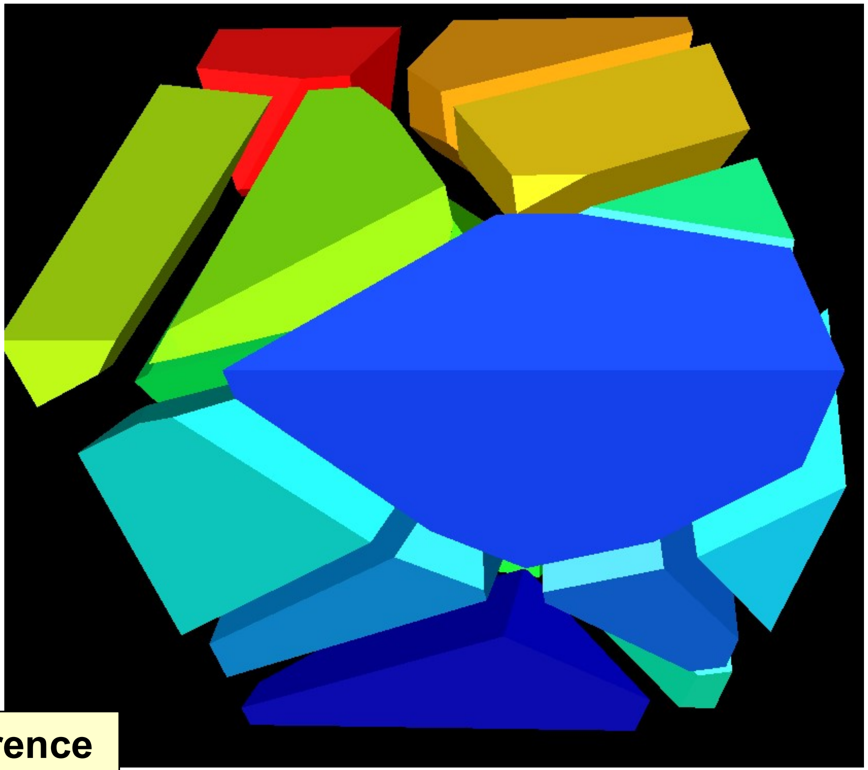
```
if( abs( dot(Eye,Normal) ) > uTol )
        discard;
else
        gl_FragColor = vec4( SILHCOLOR, 1. );
```

Why Do We Care About Lighting?

No lighting

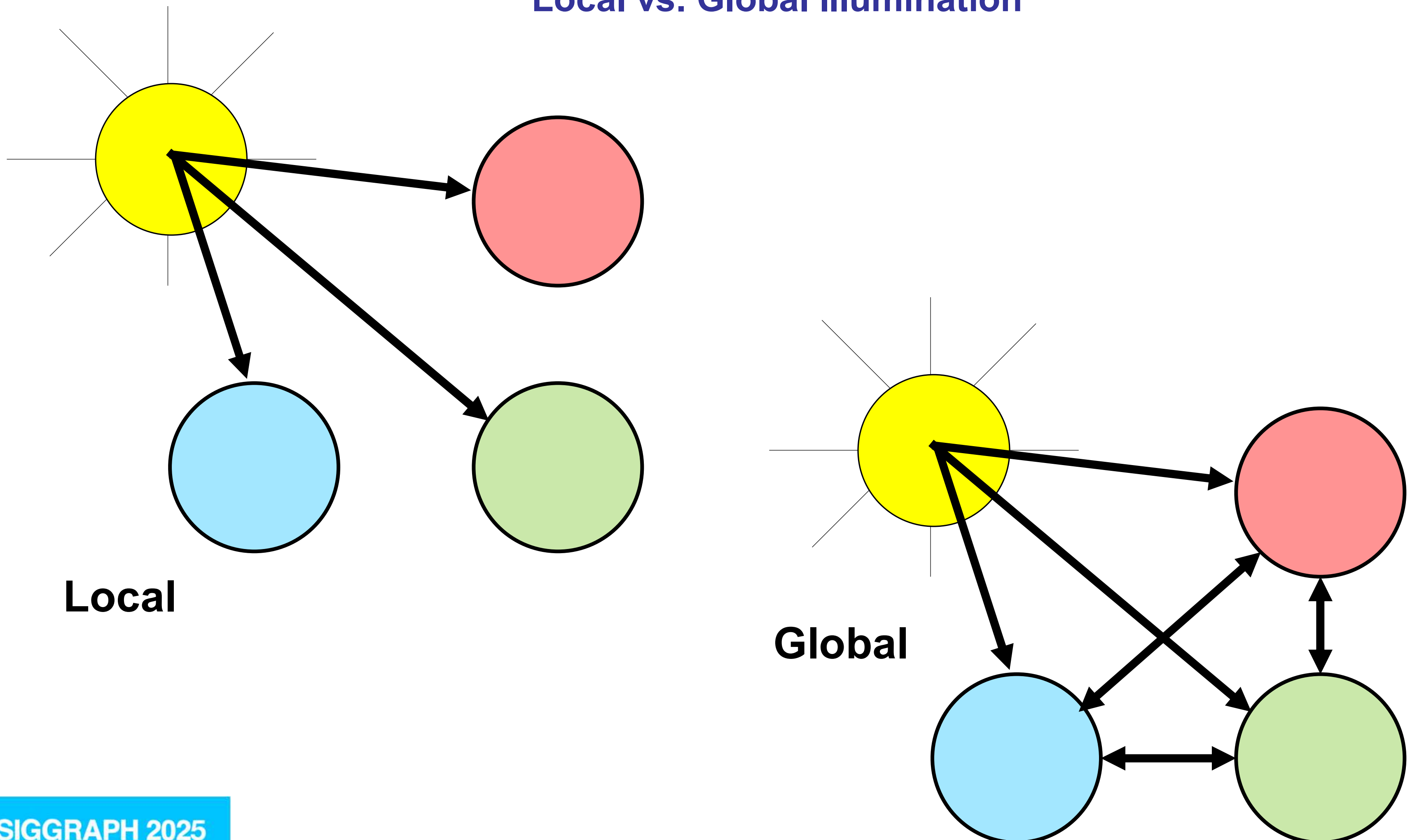


Lighting



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

Local vs. Global Illumination

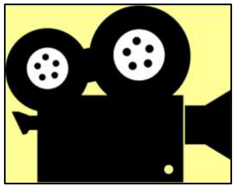


A Common type of *Local* Illumination: Ambient-Diffuse-Specular (ADS)

+

+

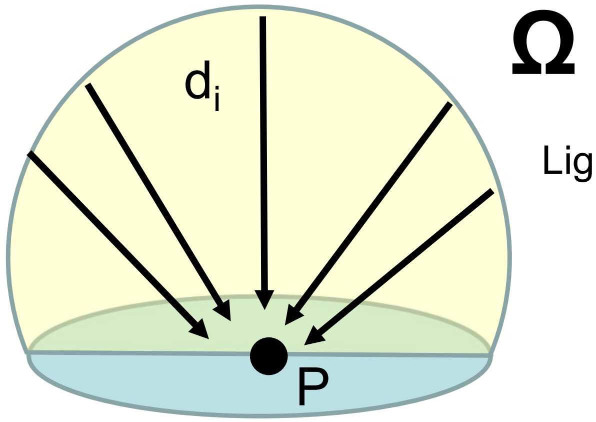
=



lighting.mp4

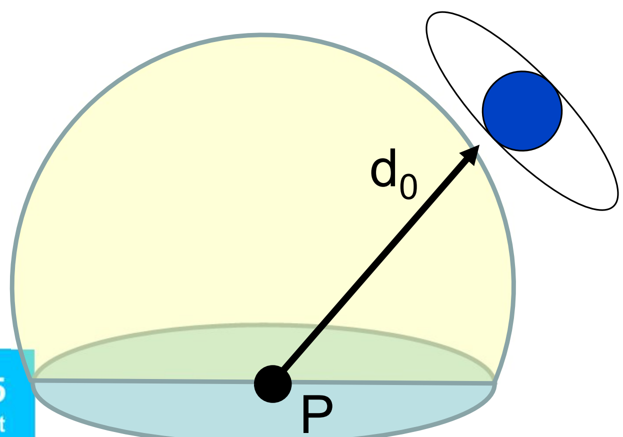
Putting them all together!

Global Illumination: The Rendering Equation



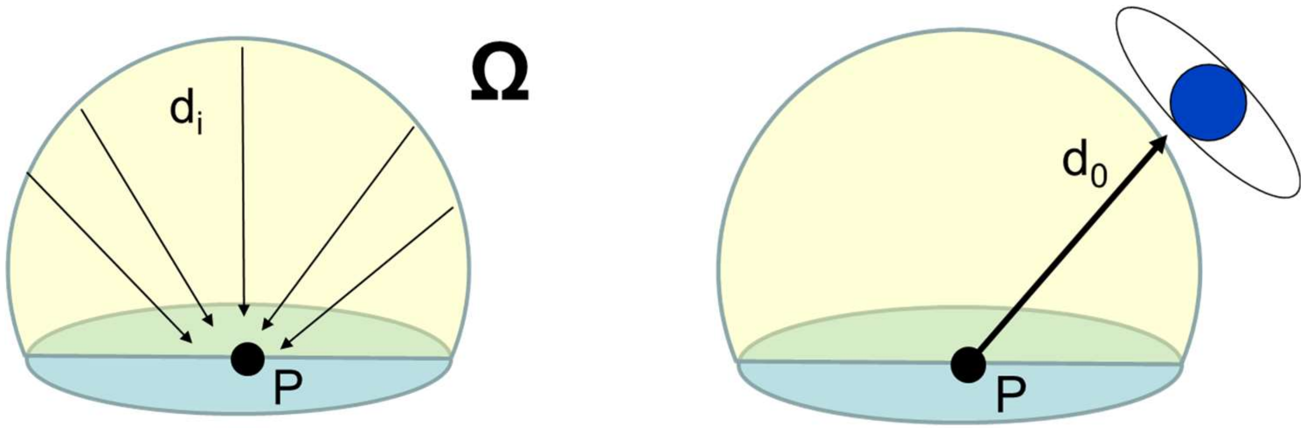
Light arriving at Point P from everywhere

Insert messy calculus equation here… ☺



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

The Rendering Equation

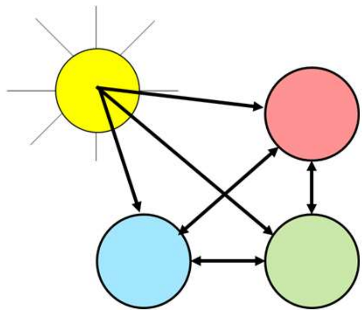


Insert messy calculus equation here… ☺

In plain language, the messy calculus describes a "light balance":

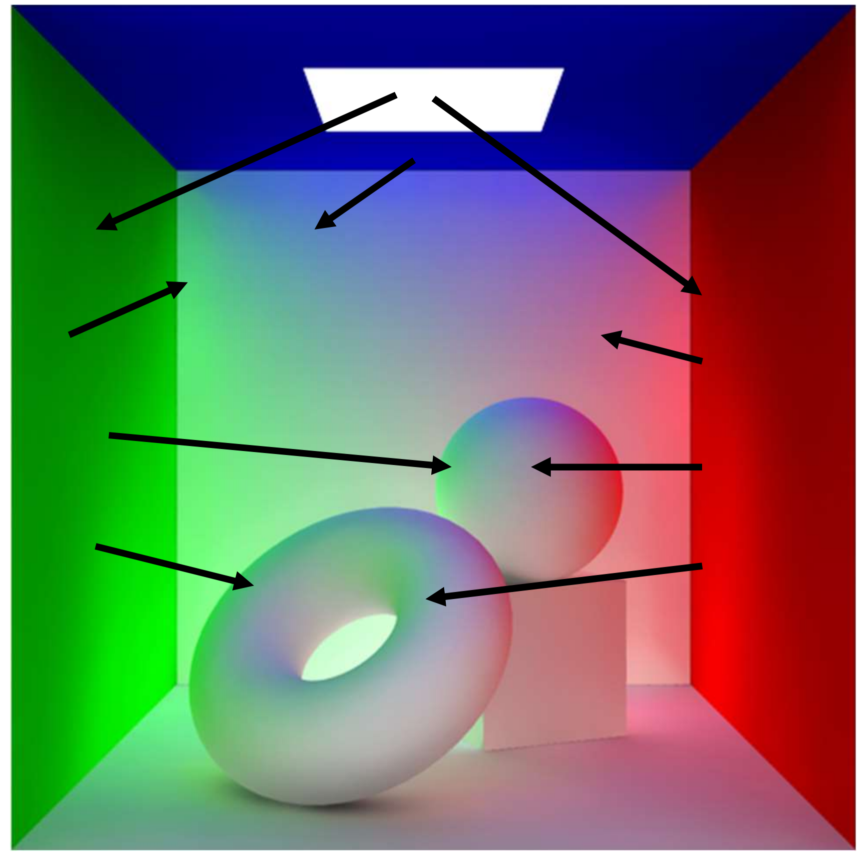
"The light shining from the point P towards your eye is the directional reflection of all the incoming light directed at the point P from all of the other objects in the scene."

The Lighting Equation at Work



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

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



http://www.swardson.com/unm/tutorials/mentalRay3/

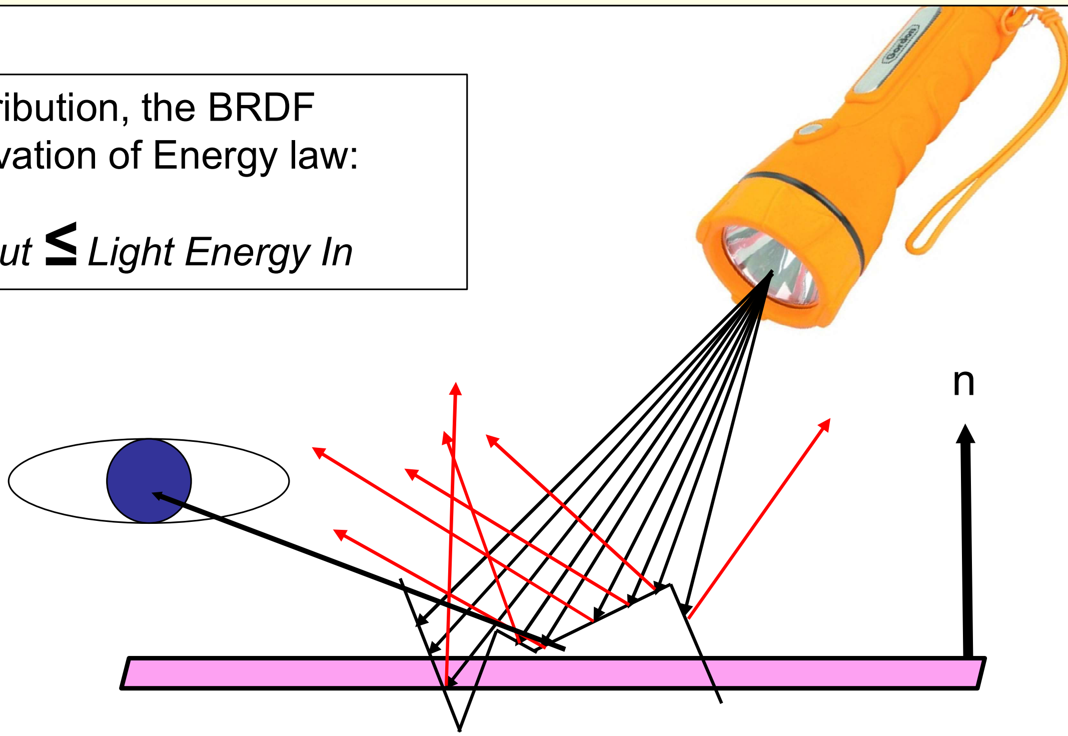
When light hits a surface, it bounces in particular ways depending on the angle and the material

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

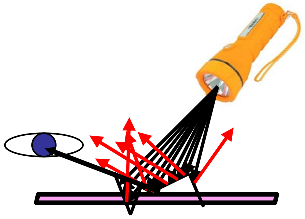
For a given material, the BRDF behavior of a light ray is a function of 4 variables: the 2 spherical coordinates of the incoming ray and the 2 spherical coordinates of the outgoing ray.

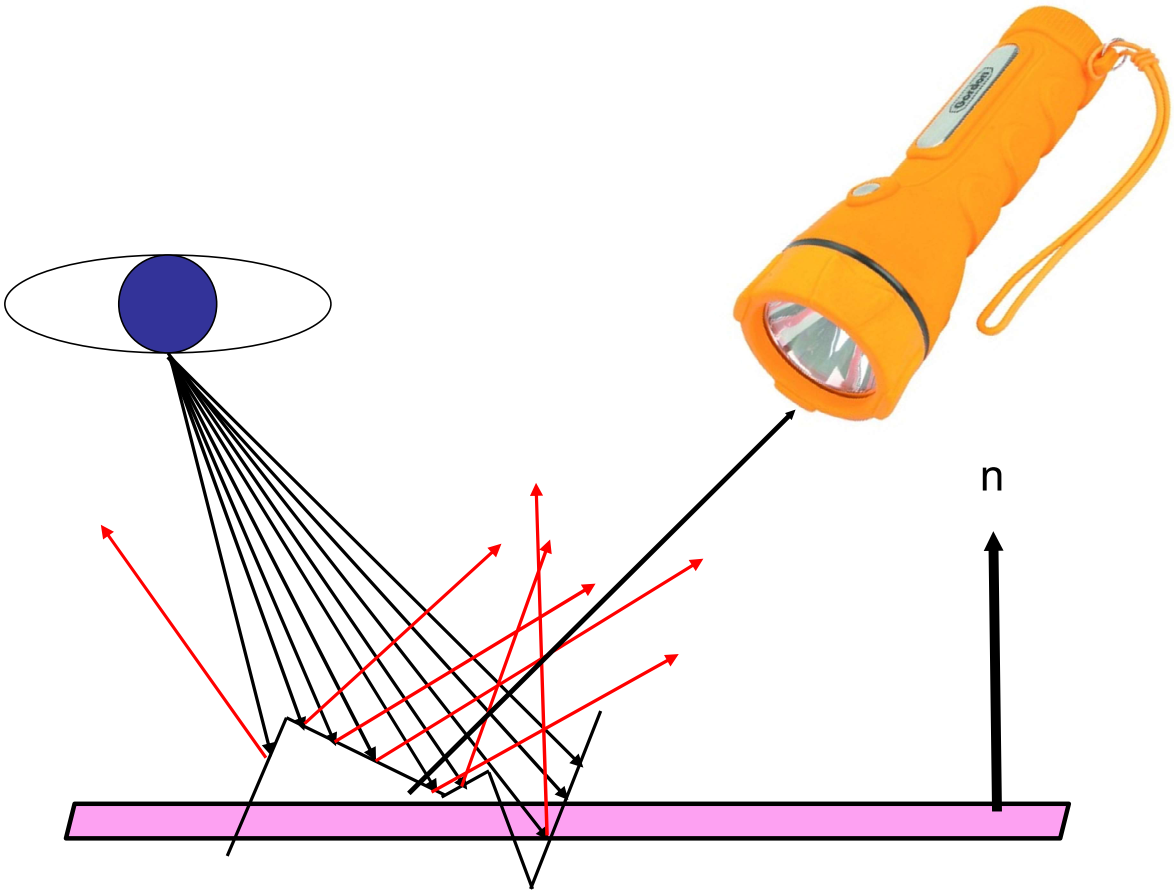
Besides being a distribution, the BRDF enforces the Conservation of Energy law:

Light Energy Out ≤ *Light Energy In*



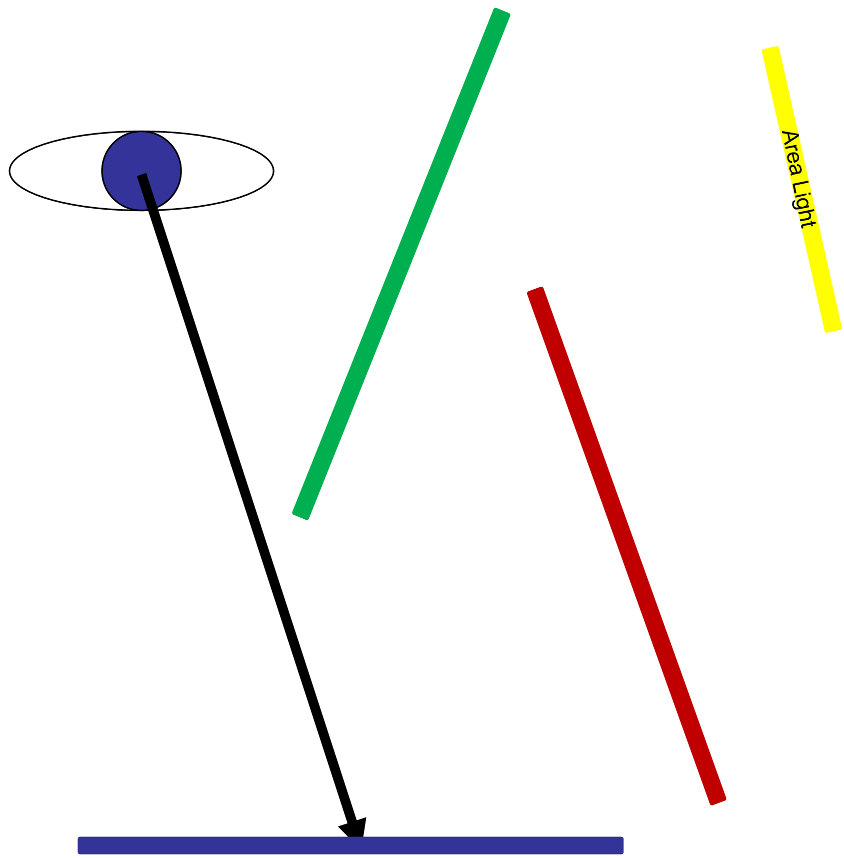
Usually, it is easier to trace from the eye





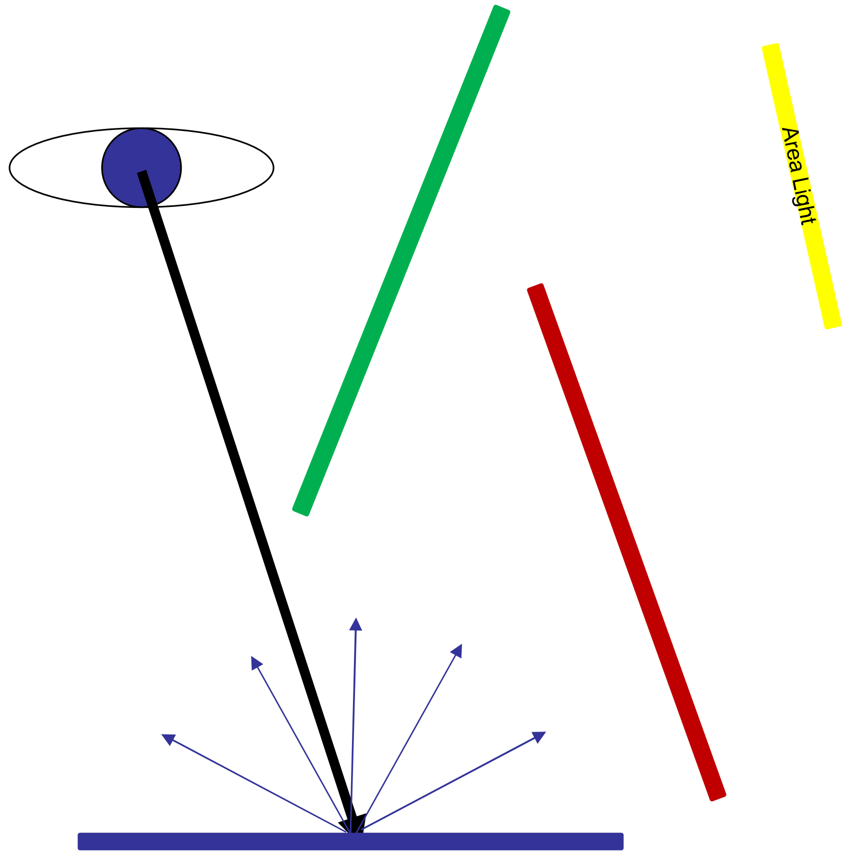
Physically-Based Rendering

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

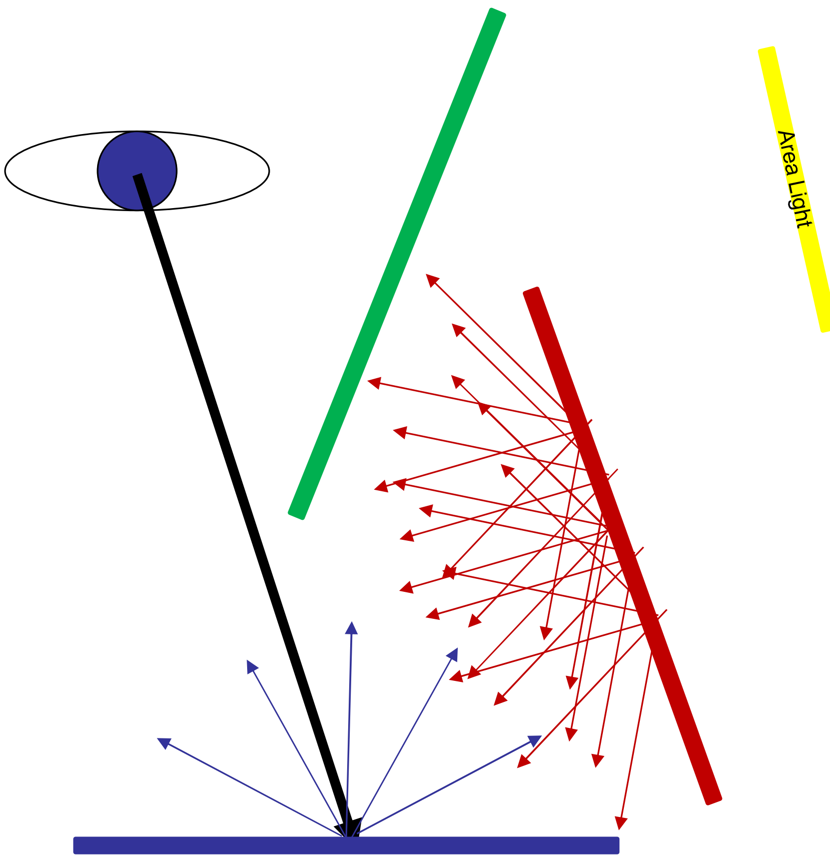


Physically-Based Rendering

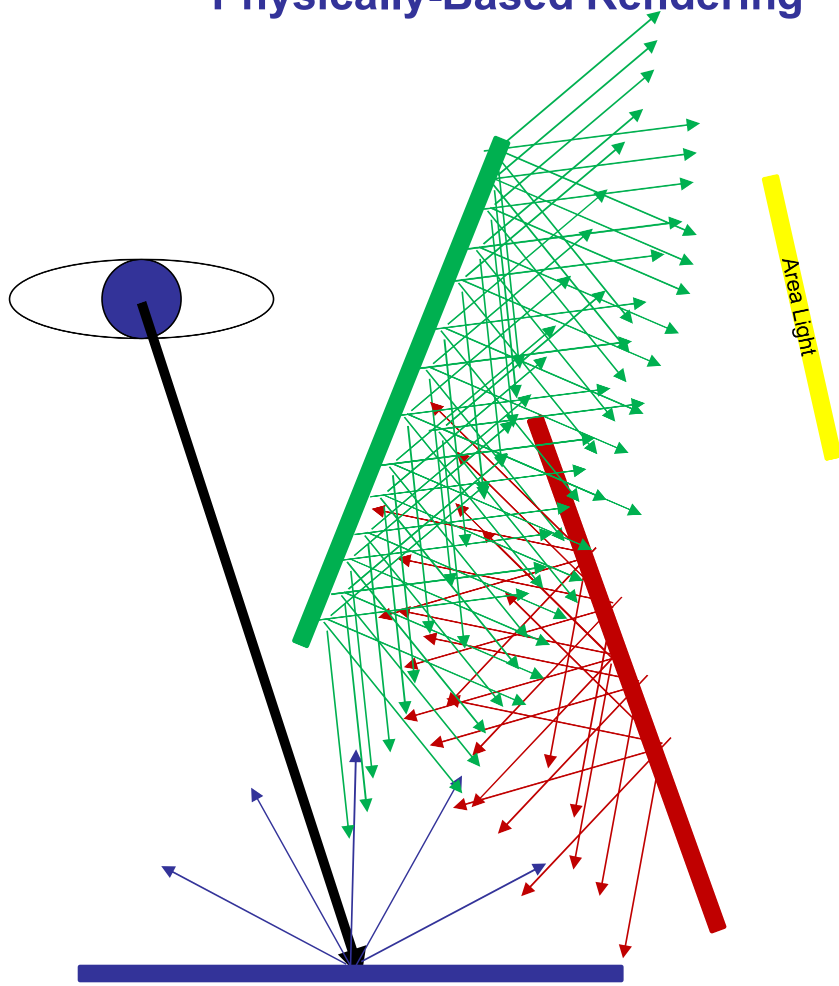
Let light can bounce around the scene, depending on how the different materials behave.



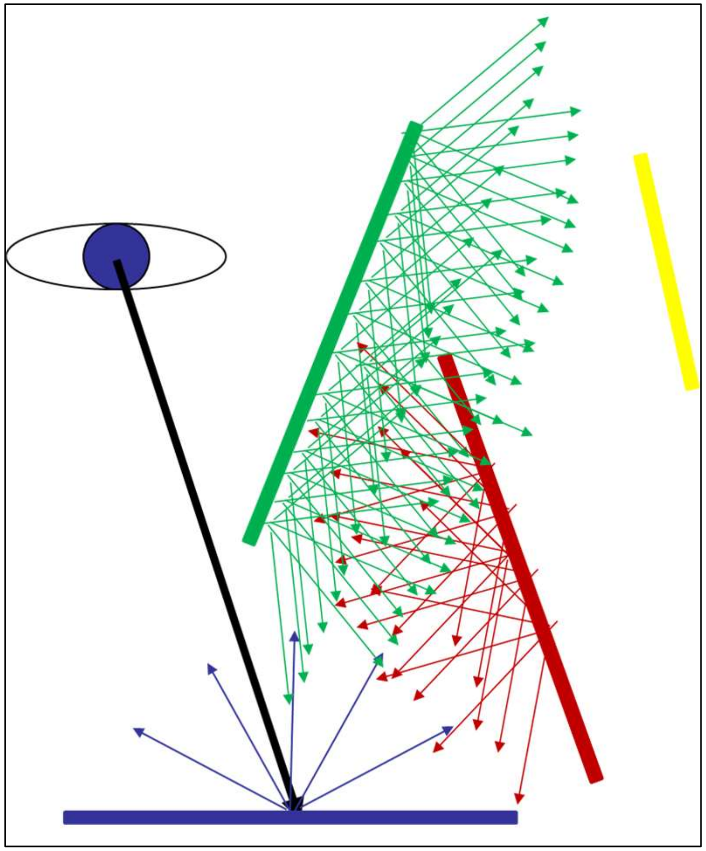
Physically-Based Rendering



Physically-Based Rendering



Physically-Based Rendering

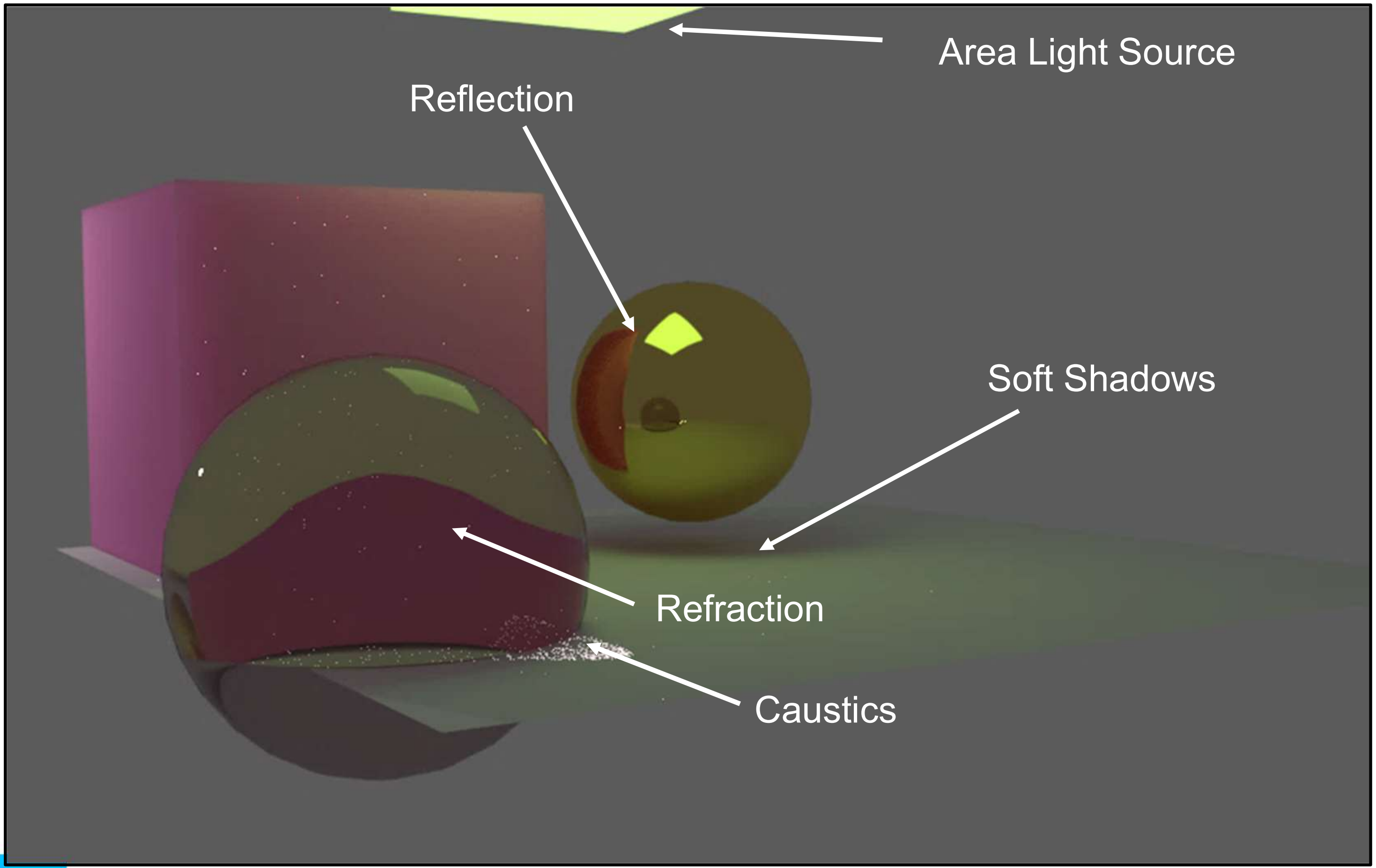


Clearly this process can spawn an infinite number of rays. How do we handle this?

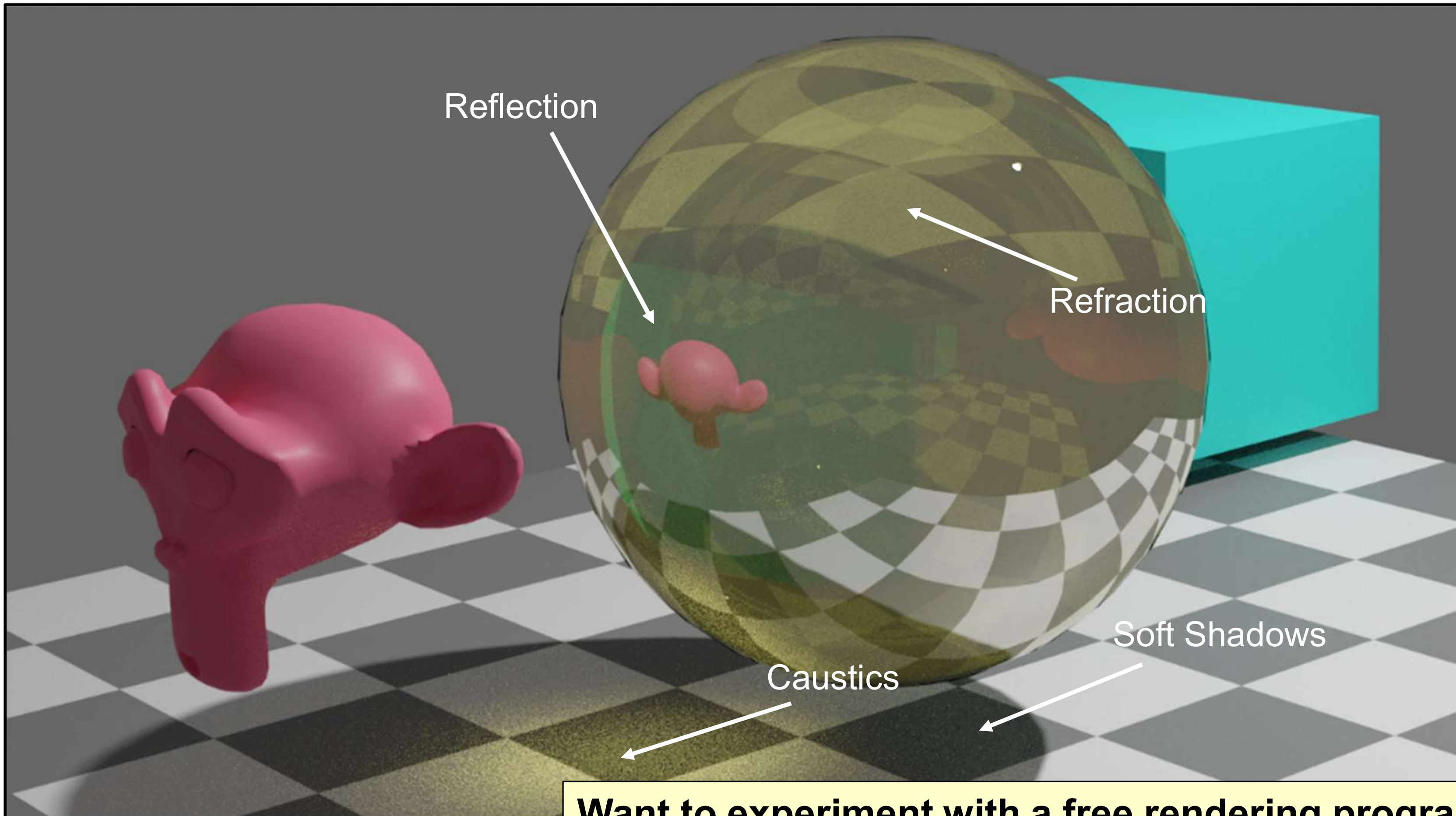
We can't use all rays that are possible, so we use a statistical subset of the rays.

This is known as **Monte Carlo** simulation.

Physically-Based Rendering using the Blender *Cycles* Renderer



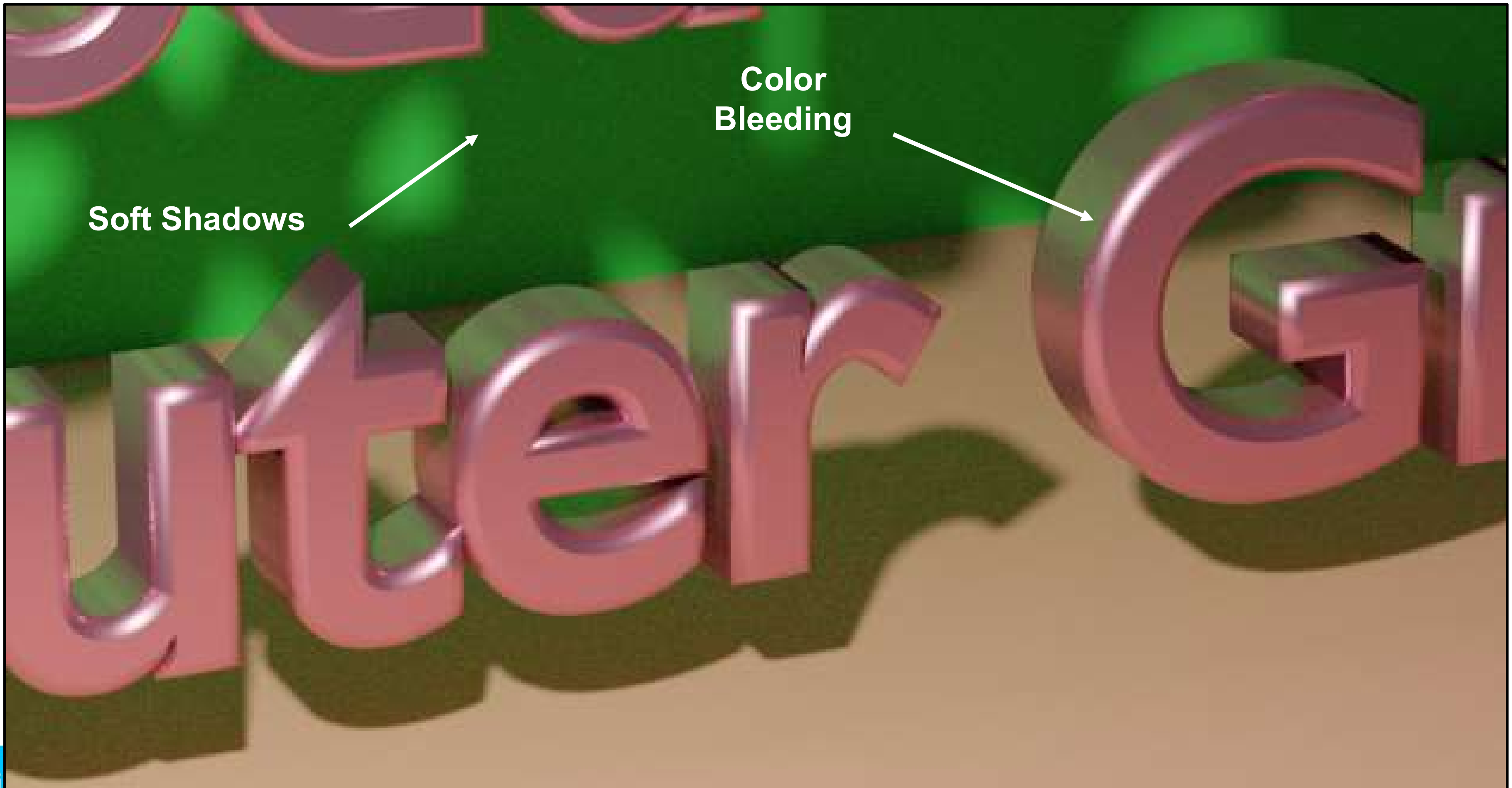
Physically-Based Rendering using the Blender *Cycles* Renderer



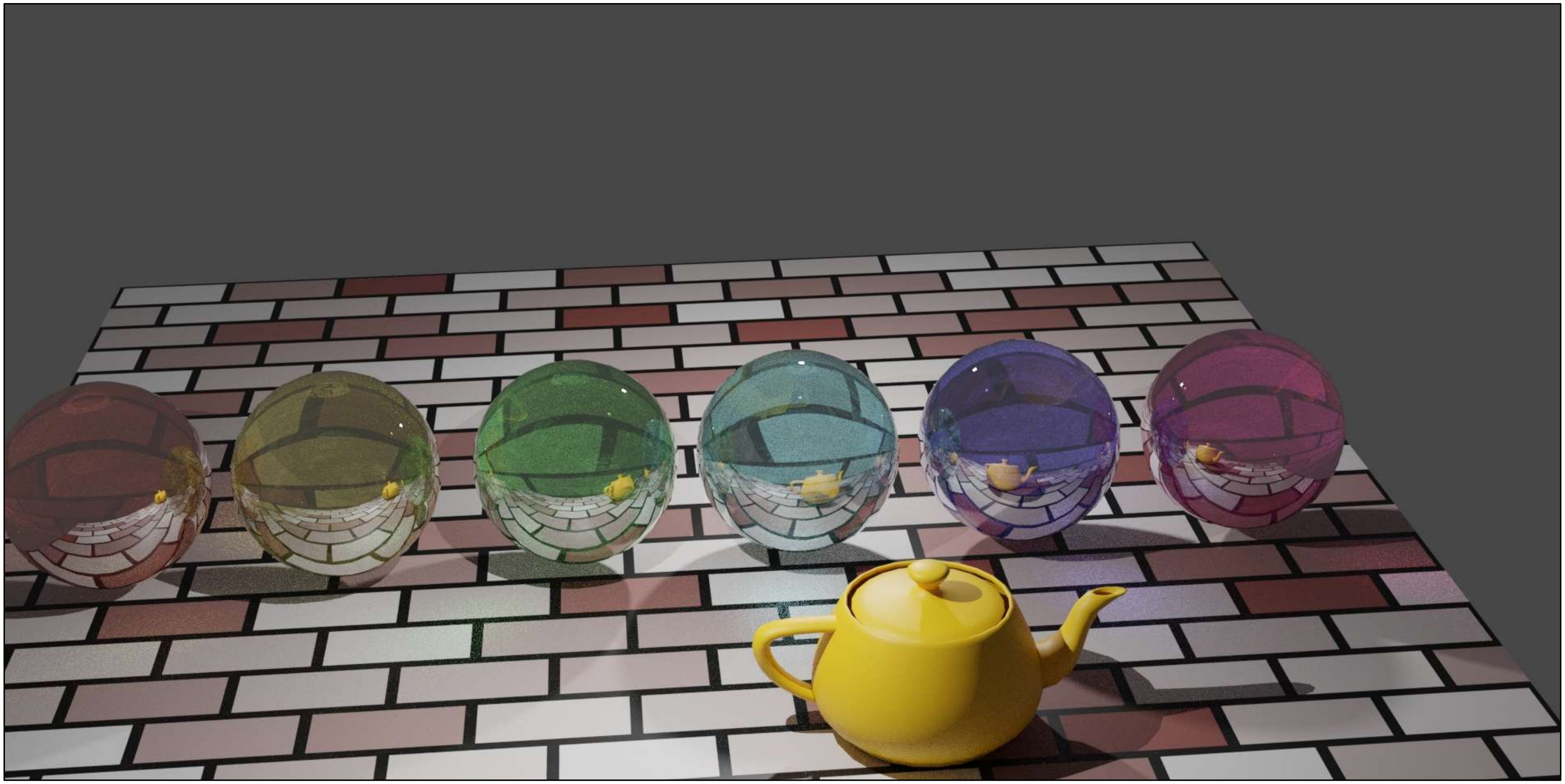
Want to experiment with a free rendering program?
Want some notes to get you started?
http://cs.oregonstate.edu/~mjb/blender

SIGGRAPH 2025
Vancouver+ 10-14 August

An Example from the Title Slide



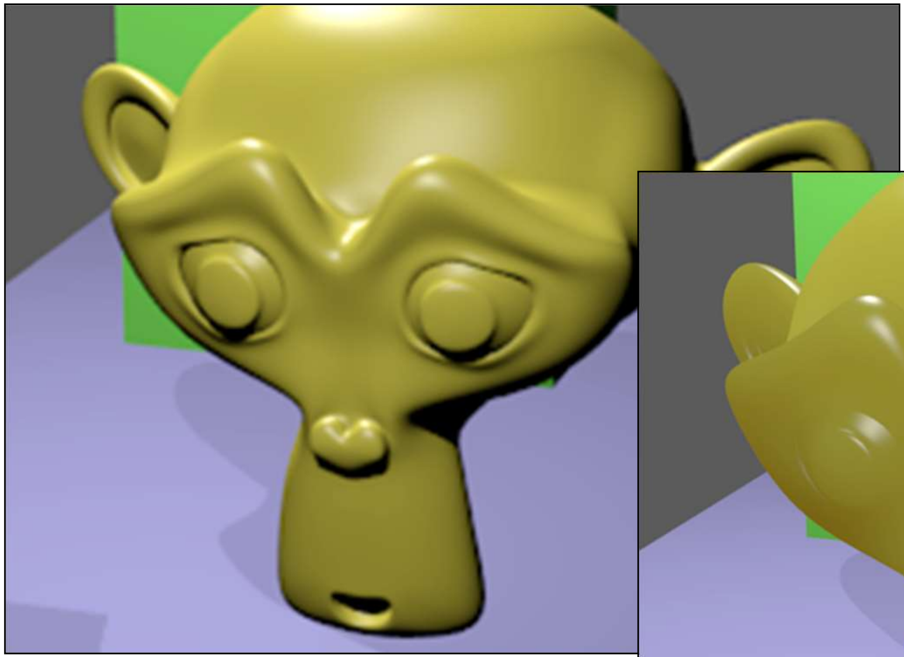
More from the Blender *Cycles* Renderer



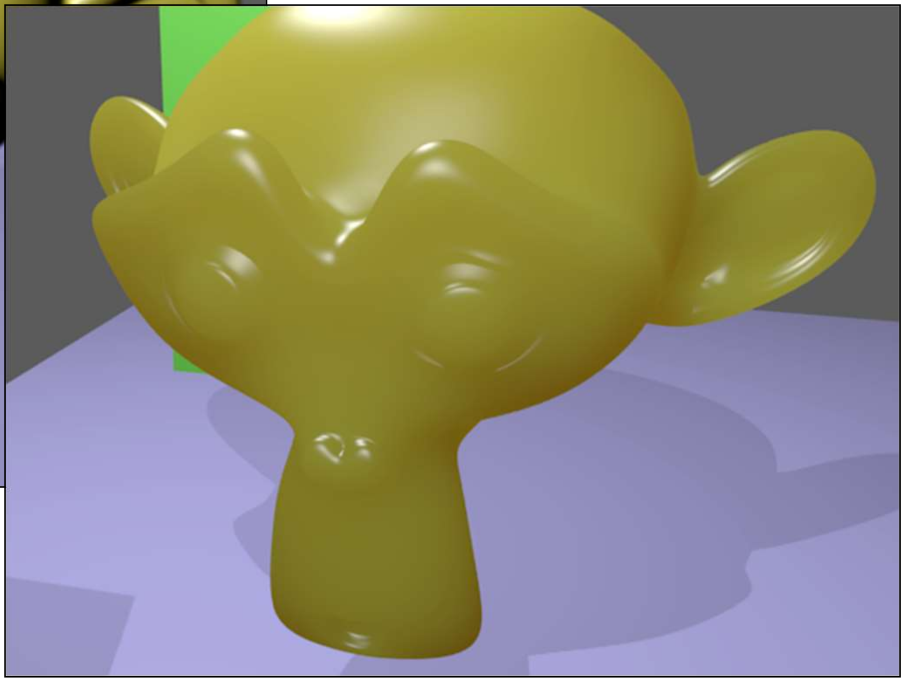
Want to experiment with a free rendering program?
Want some notes to get you started?
http://cs.oregonstate.edu/~mjb/blender

Tricky Lighting Situations -- Subsurface Scattering

Subsurface Scattering can model light bouncing around *within* an object before coming back out. This is a good way to represent skin, wax, milk, etc.



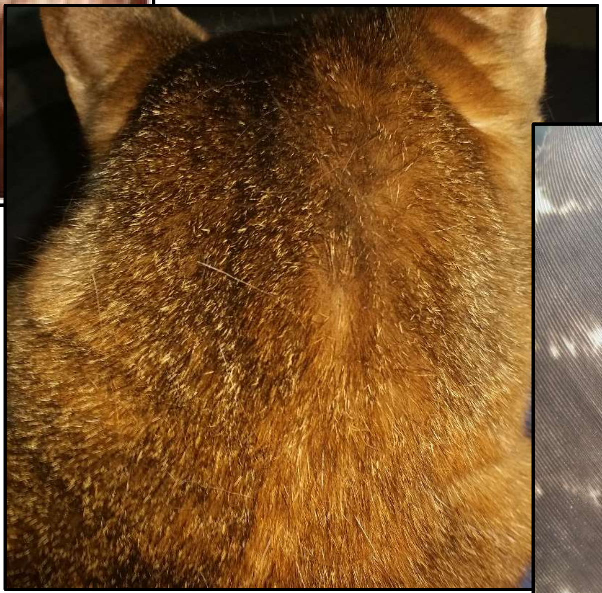
Without Subsurface Scattering



With Subsurface Scattering

More Tricky Lighting Situations

Hair



Fur

Feathers

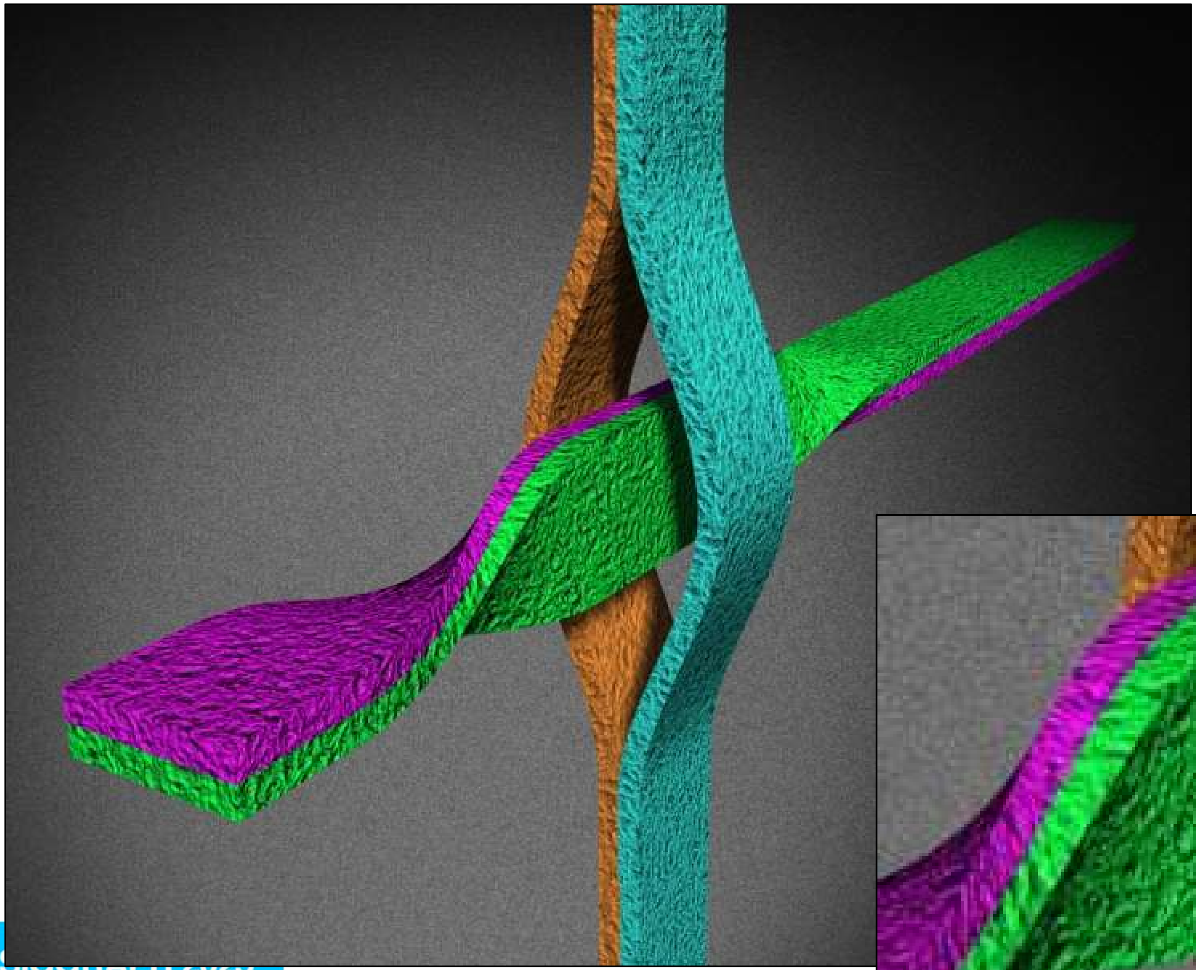
SIGGRAPH 2025
Vancouver+ 10-14 August

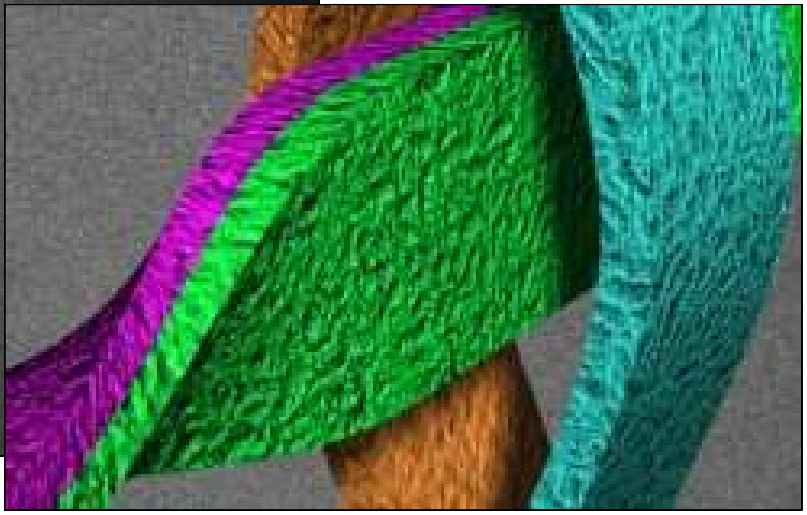
Watch for these rendering situations at SIGGRAPH and in CG movies!


Cool Renderings
for Your Enjoyment

Cool Renderings: Bumpy Cloth (Celtics Knots)

Andrew Glassner, used by permission





What makes this awesome:

- The gentle twist in the cloth
- The "divots" in the cloth and how light interacts with them

Cool Renderings: Foliage Rendering



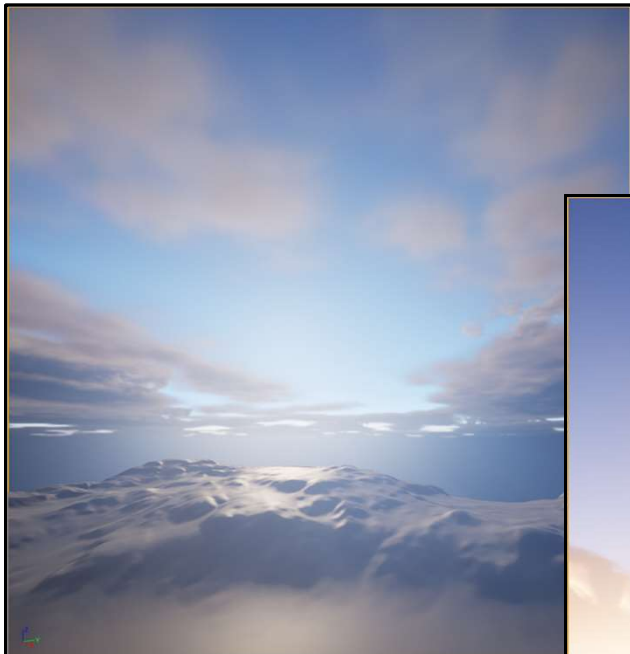
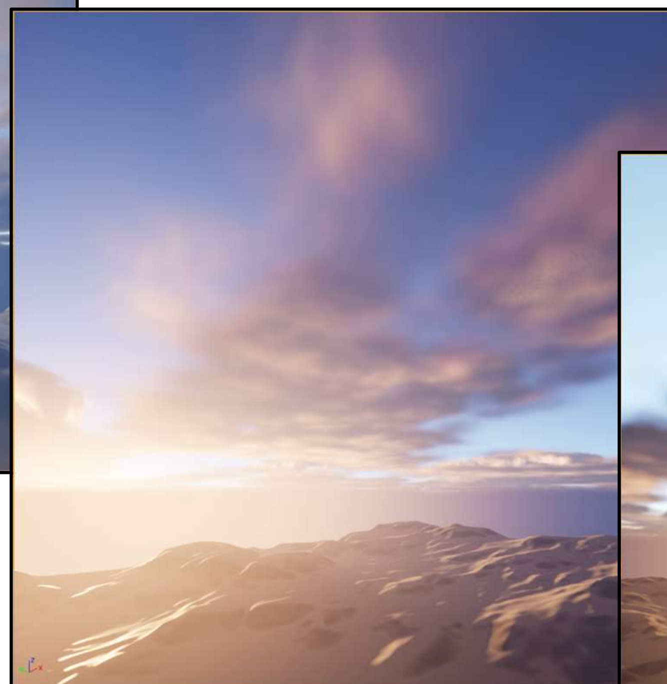

Grace Todd, used by permission

What makes this awesome:

- The automated generation of the foliage using generative AI and Blender geometry nodes

Cool Renderings: Atmospherics

What makes this awesome:
- Various atmospheric conditions diffuse light in different ways from different parts of the scene

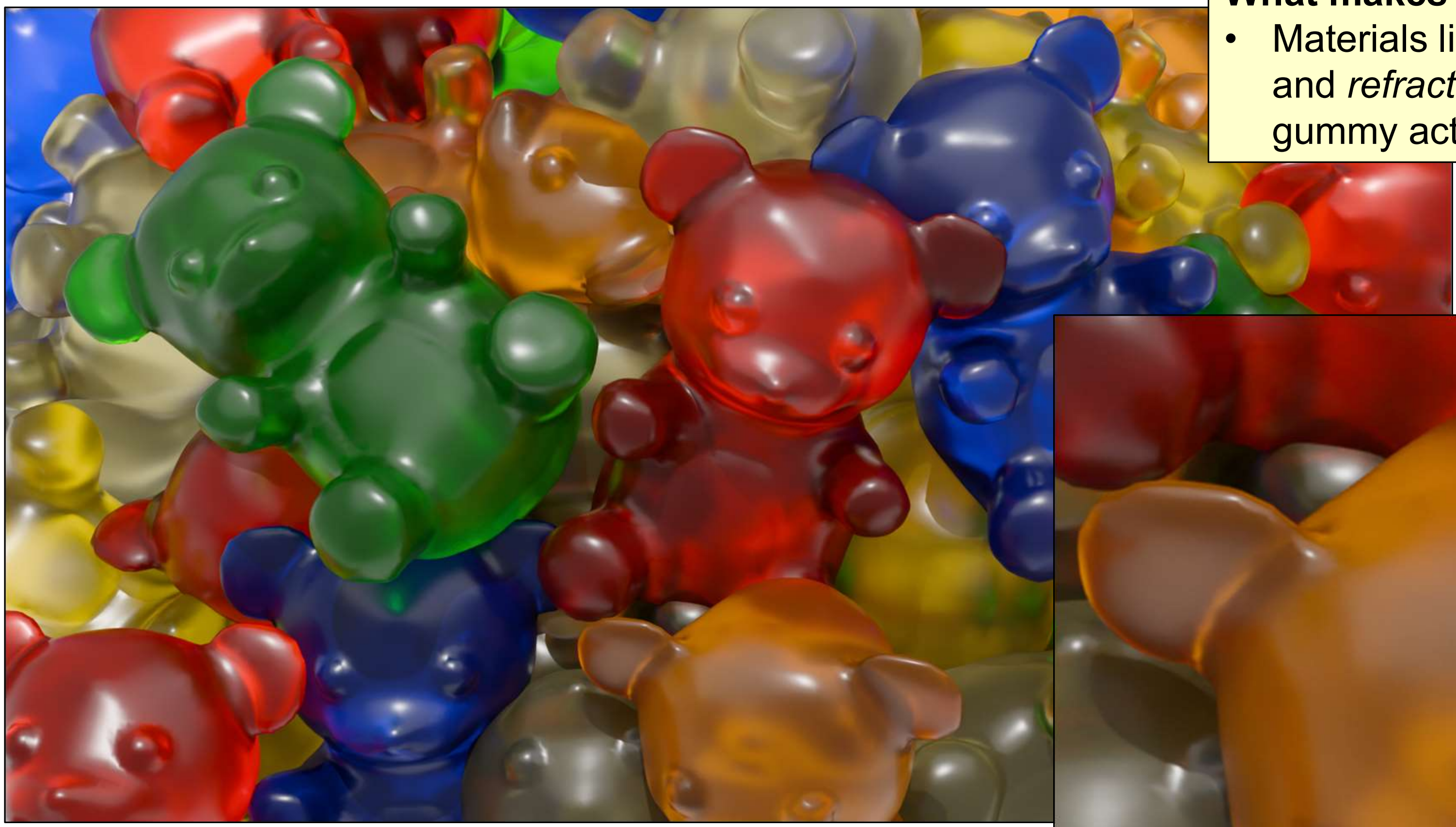




Nathan DeStafeno, used by permission

Cool Renderings: Mmmmm, Gummies!

What makes this awesome:

- Materials like gummies both *reflect* and *refract* light, with the color of the gummy acting like a color filter

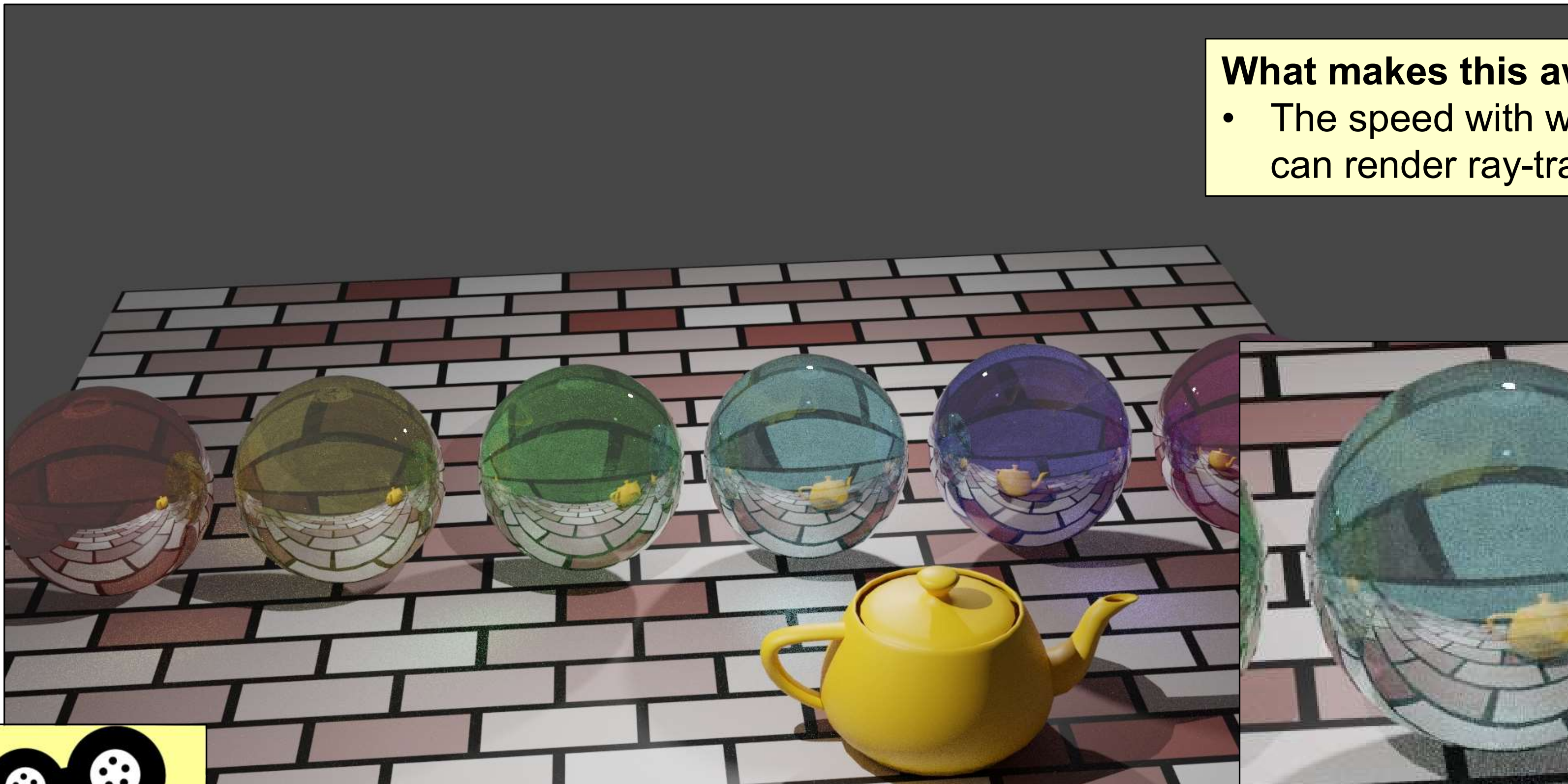


Grace Todd, used by permission

Cool Renderings: Hardware Ray-Tracing

What makes this awesome:

- The speed with which today's hardware can render ray-traced images



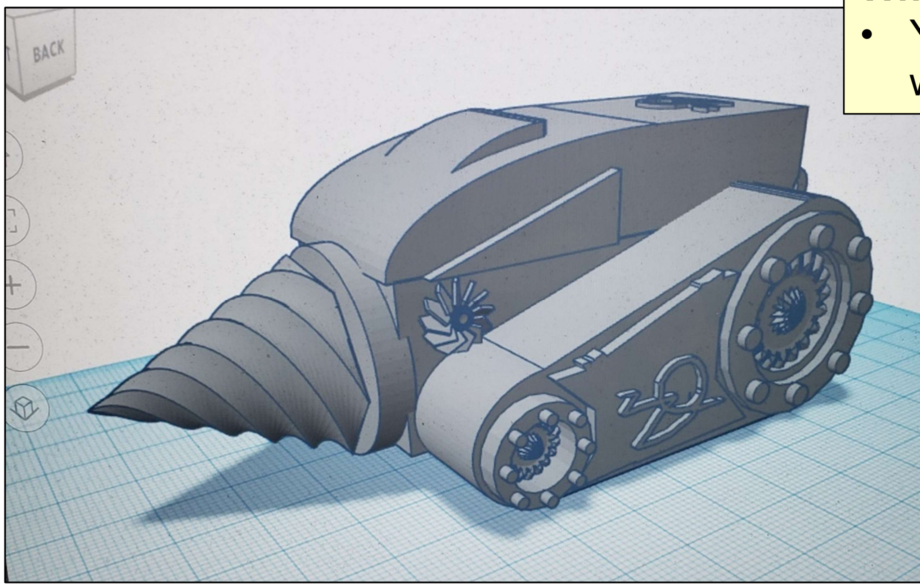
SixSpheres.mp4

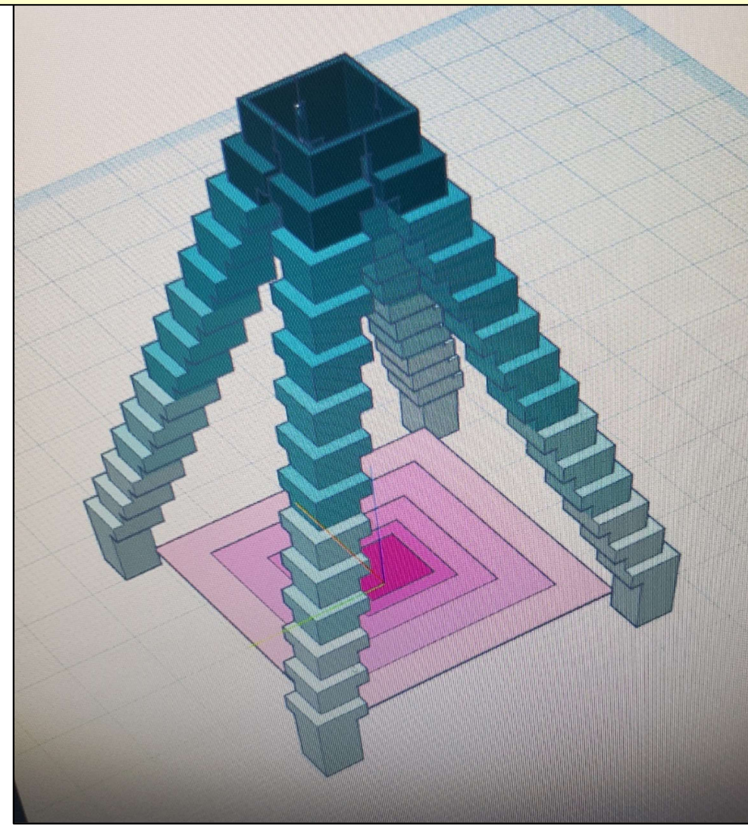
1920x1080 pixels rendered in ***2.68*** seconds using Blender on an Nvidia RTX A6000

Cool Renderings: Grades 3-8 using TinkerCad and CodeBlocks

What makes this awesome:

- Young kids are impressively good at this sort of thing. They will soon be at SIGGRAPH to kick our butts. ☺



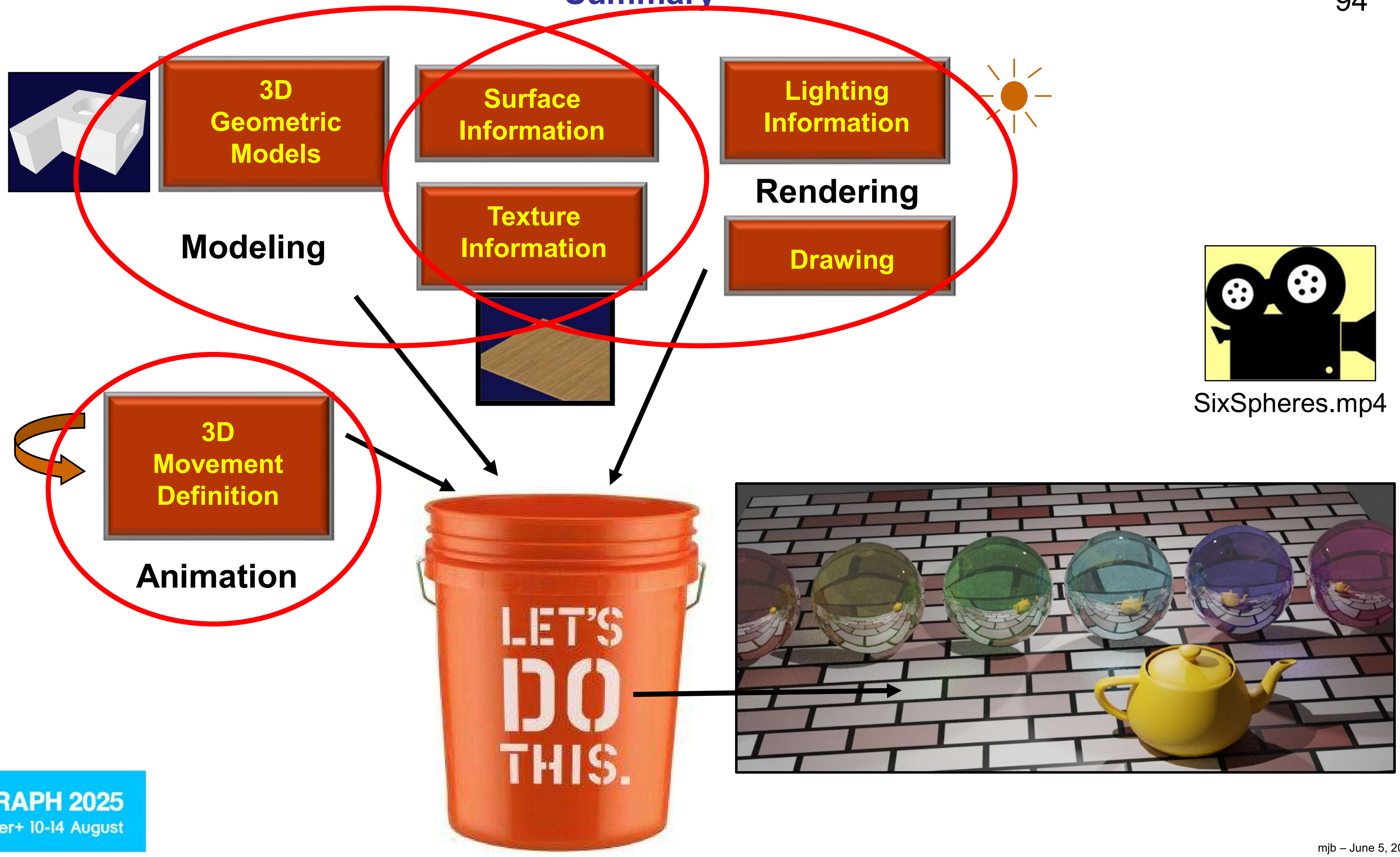


Cool Renderings: High Schoolers using Blender

What makes this awesome:

- High school kids are even more impressively good at this sort of thing. They will soon be at SIGGRAPH to kick our butts. ☺

Summary


3D Geometric Models
Surface Information
Lighting Information
Rendering
Modeling
Texture Information
Drawing
SixSpheres.mp4
3D Movement Definition
Animation
LET'S DO THIS.

Conclusions !

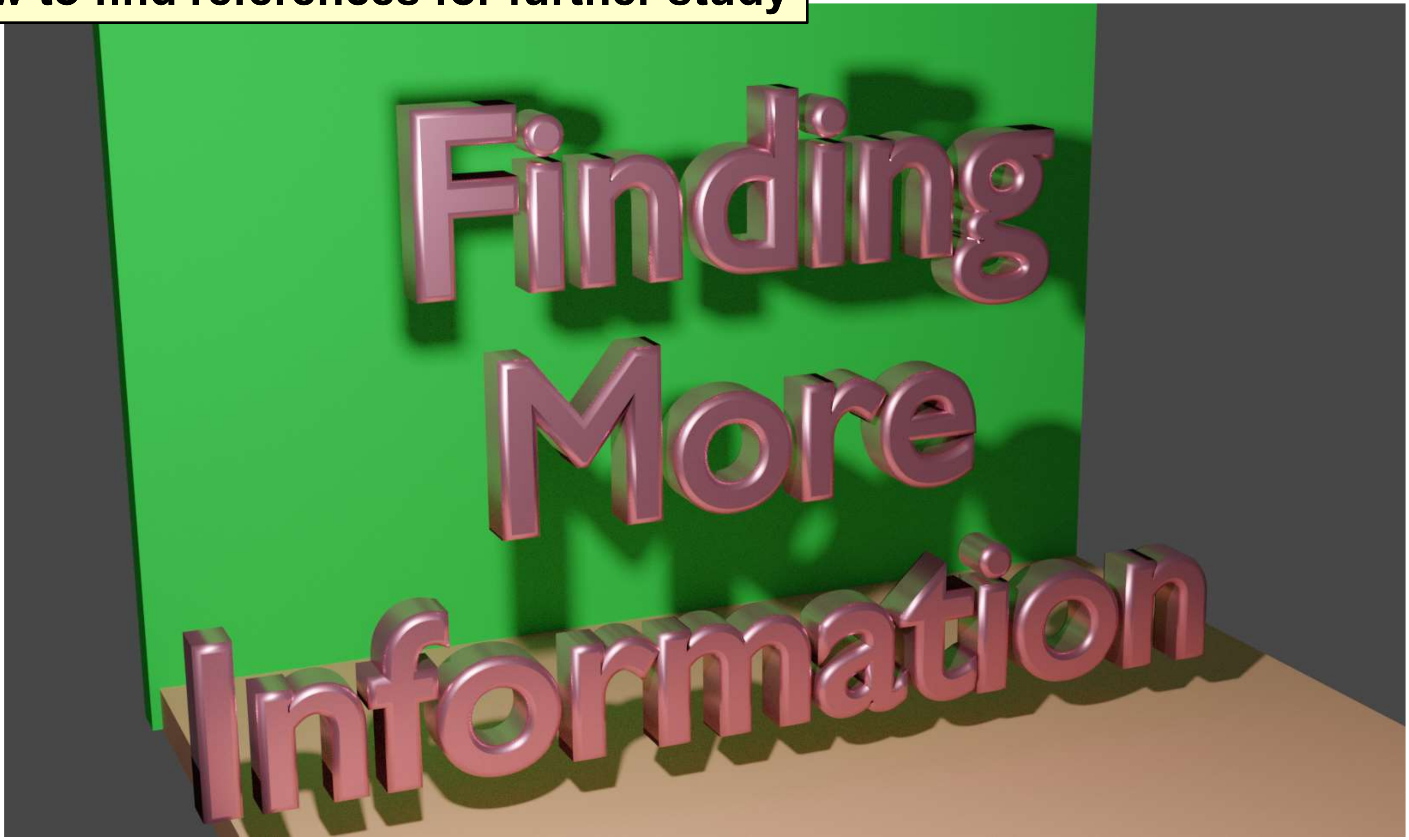
- SIGGRAPH moments will never come again. Well, this is usually true, but through the magic of the 2025 videos, they might reappear. *But be aware of what is going to be recorded and archived and what isn't.* And, if it is to be archived, how long will you have access to it?

- Especially take advantage of the not-to-be-archived or not-to-be-archived-for-very-long events because you cannot re-live them forever.

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

- Have fun doing it!

- **How to find references for further study**



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

Check Out the *More Information* Document to be found at:

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*

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

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

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

■ ■ ■

Check Out Other Sets of Free Notes:

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

University course notes:

- Introduction to Computer Graphics
- Computer Graphics Shaders
- CS Skills for Simulation and Game Programming
- Parallel Programming
- Scientific Visualization
- Vulkan

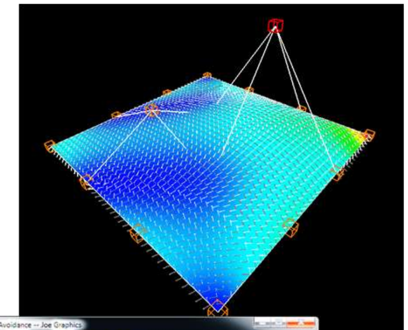
SIGGRAPH course notes:

- A Whirlwind Introduction to Computer Graphics

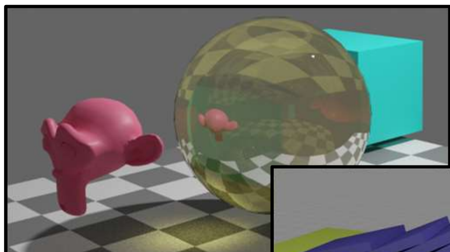
K-12 notes:

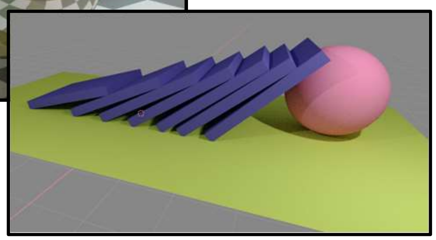
- Blender
- CodeBlocks
- Processing
- Scratch
- Scratch Jr.
- SimLab
- Tinkercad

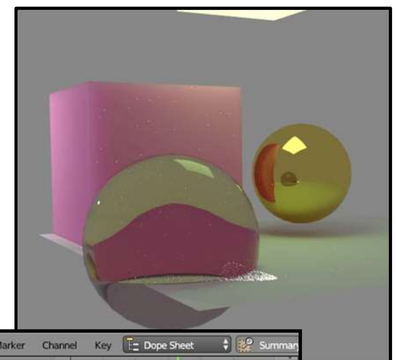
Thank You!

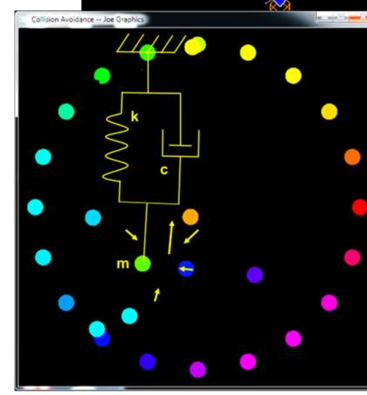


Mike Bailey
mjb@cs.oregonstate.edu

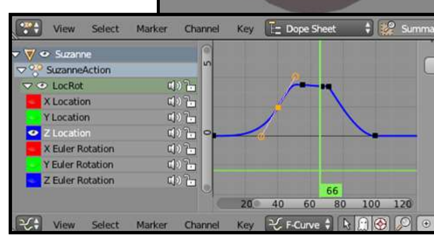








http://cs.oregonstate.edu/~mjb/whirlwind



SIGGRAPH 2025
Vancouver+ 10-14 August