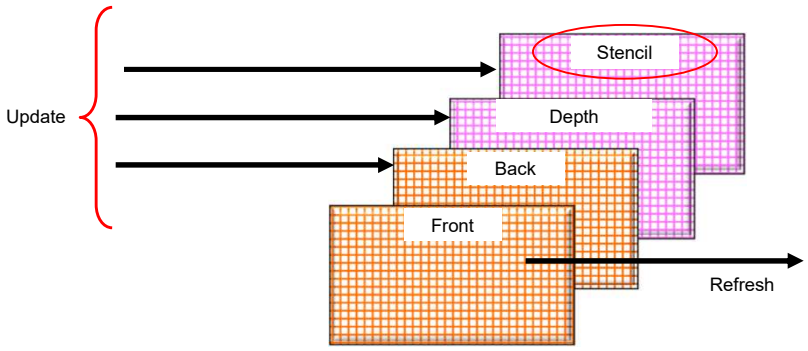
Using the Stencil Buffer

Mike Bailey

mjb@cs.oregonstate.edu

Oregon State
University
Computer Graphics

The Framebuffers



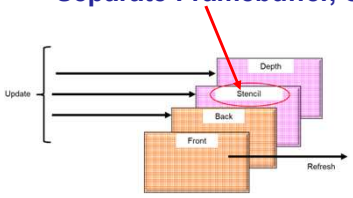
Here's what the Stencil Buffer can do for you:

1. While drawing into the Back Buffer, you can write values into the Stencil Buffer at the same time.
2. While drawing into the Back Buffer, you can do arithmetic on values in the Stencil Buffer at the same time.
3. While drawing into the Back Buffer, the Stencil Buffer can be used to write-protect certain parts of the Back Buffer.

Oregon
Unive
Computer

You Can Think of the Stencil Buffer as a Separate Framebuffer, or, You Can Think of it as being Per-Pixel

3



Both are correct, but I like thinking of it "per-pixel" better.

Oregon State University
Computer Graphics

mjb – September 2, 2024

3

The Stencil Buffer is Tested Per-Pixel, Very Much Like the Depth Buffer Is

4

glStencilFunc(func, ref, mask)

This specifies the comparison test that is to be done per-pixel.

func can be any of GL_NEVER, GL_ALWAYS, GL_EQUAL, GL_NOTEQUAL, GL_LESS, GL_LEQUAL, GL_GREATER, GL_GEQUAL

ref is an integer reference value that is used to test the pixel's existing stencil value against using the chosen ***func***

mask is set to 1 in all these examples

The stencil test produces a *true* or *false* value at each pixel where drawing is to be done.

```
if( ref  <func>  S_existing is true )
{
        Allow the color write to the existing pixel to take place;
        Modify the pixel's existing stencil value depending on what the glStencilOp says to do;
}
```

Computer Graphics

mjb – September 2, 2024

4

This Tells You What to Do with the *true* or *false* Value from the Stencil Test

glStencilOp(sfail, zfail, zpass)

This specifies how a pixel's stencil value is modified when a fragment passes or fails the stencil test depending on what combinations of *true* and *false* the stencil test and the depth buffer test produce. If the stencil test fails, then *sfail* happens. If the stencil test succeeds, then either *zfail* or *zpass* happen depending on if the depth-buffer test failed or succeeded.

The three values can be any of:

GL_KEEP	Retain the existing stencil value
GL_ZERO	Set the stencil value to zero
GL_REPLACE	Replace the stencil value with **ref** from the Stencil Func
GL_INCR	Increment the stencil value, with clamping
GL_INCR_WRAP	Increment the stencil value, without clamping
GL_DECR	Decrement the stencil value, with clamping
GL_DECR_WRAP	Decrement the stencil value, without clamping
GL_INVERT	Bitwise toggle the stencil bits: 0's → 1's, 1's → 0's

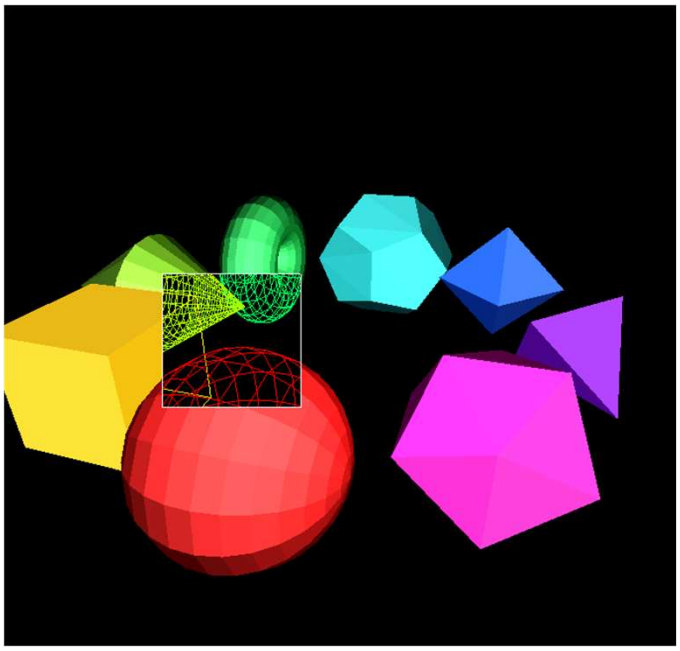

```
if( ref  <func>  S_existing is true )
{
        Allow the color write to the existing pixel to take place;
        Modify the pixel's existing stencil value depending on what the glStencilOp says to do;
}
```


Setting Up the Stencil Buffer

```
// at the top of the program:

const int STENCILBIT = 1;
const int DEFAULT_STENCIL = 0;
const float BIGX = 2.;
const float BIGY = BIGX;
const float CLOSEZ = -1.;
float Xlens, Ylens;
float Box = 0.40f;

// in InitGraphics( ):

glutInitDisplayMode( GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH | GLUT_STENCIL );

glClearColor( BACKGROUND_COLOR );
glClearStencil( DEFAULT_STENCIL );


// in Display( ):
. . .
glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT );
. . .
glEnable( GL_STENCIL_TEST );
. . .
```

Using the Stencil Buffer to Create a *Magic Lens*

7



Oregon State University
Computer Graphics

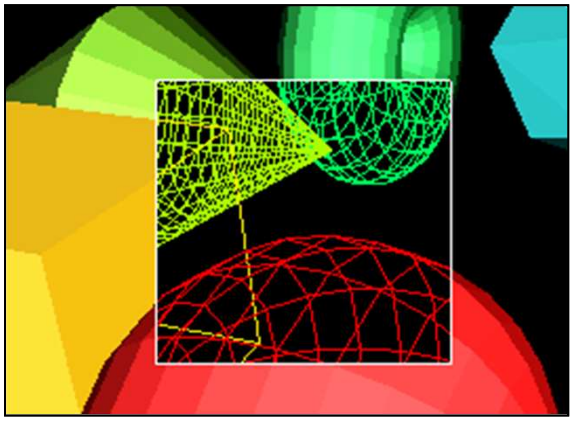
mjb – September 2, 2024

7

Using the Stencil Buffer to Create a *Magic Lens*

8

1. Clear the SB = 0
2. Enable the SB
3. Write protect the color buffer and depth buffer
4. Draw a filled square, while setting SB = 1
5. Write-enable the color buffer and depth buffer
6. Draw the solids wherever SB == 0
7. Draw the wireframes wherever SB == 1
8. Disable the SB



Oregon State University
Computer Graphics

mjb – September 2, 2024

8

Moving the Magic Lens with the Middle Mouse Button

9

```
// in MouseMotion( ):

 if( ActiveButton & MIDDLE )
{
	if( Stencil == LENS )
	{
		int w = glutGet( GLUT_WINDOW_WIDTH );
		int h = glutGet( GLUT_WINDOW_HEIGHT );
		Xlens =  2.*(float)x/(float)w  - 1.;
		Ylens = -2.*(float)y/(float)h  + 1.;
	}
	else
	{
		Scale += SCLFACT * (float) ( dx - dy );
	}
}
```

x/w ranges from 0. to 1.
y/h ranges from 1. to 0
Xlens and Ylens range from -1. to 1. (NDC)

Oregon State University
Computer Graphics

mjb – September 2, 2024

9

Using the Stencil Buffer to Create a *Magic Lens*

10

```
glMatrixMode( GL_PROJECTION );
glLoadIdentity( );

glMatrixMode( GL_MODELVIEW );
glLoadIdentity( );

glDepthMask( GL_FALSE );
glColorMask( GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE );

glStencilFunc( GL_ALWAYS, 1, STENCILBIT );
glStencilOp( GL_REPLACE, GL_REPLACE, GL_REPLACE );

glBegin( GL_QUADS );
	glVertex2f( Xlens-Box/2.,  Ylens-Box/2. );
	glVertex2f( Xlens+Box/2., Ylens-Box/2. );
	glVertex2f( Xlens+Box/2., Ylens+Box/2. );
	glVertex2f( Xlens-Box/2.,  Ylens+Box/2. );
glEnd( );

glColorMask( GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE );
glDepthMask( GL_TRUE );
```

These two identity transformation matrices cause the drawing to take place in NDC (-1 to 1.), which is what Xlens, Ylens, and Box are defined in

Write protect the depth and color buffers

Everywhere we draw, always replace the stencil value with a **1**

Draw a filled-in box

Write-enable the depth and color buffers

Oregon State University
Computer Graphics

mjb – September 2, 2024

10

Using the Stencil Buffer to Create a *Magic Lens*


```
<< set the GL_PROJECTION and GL_MODELVIEW matrices as normal >>

glEnable( GL_LIGHTING );
glStencilFunc( GL_EQUAL, 0, STENCILBIT );
glStencilOp( GL_KEEP, GL_KEEP, GL_KEEP );
glShadeModel( GL_SMOOTH );
for( int i = 0; i < 8; i++ )
{
        glCallList( SolidLists[ i ] );
}

glDisable( GL_LIGHTING );
glStencilFunc( GL_EQUAL, 1, STENCILBIT );
glStencilOp( GL_KEEP, GL_KEEP, GL_KEEP );
glShadeModel( GL_FLAT );
for( int i = 0; i < 8; i++ )
{
        glCallList( WireLists[ i ] );
}

<< set the GL_PROJECTION and GL_MODELVIEW matrices to identity again >>

glDisable( GL_LIGHTING );
glShadeModel( GL_FLAT );
glDisable( GL_DEPTH_TEST );
glColor3f( 1., 1., 1. );
glBegin( GL_LINE_LOOP );
        glVertex2f( Xlens-Box/2.,  Ylens-Box/2. );
        glVertex2f( Xlens+Box/2., Ylens-Box/2. );
        glVertex2f( Xlens+Box/2., Ylens+Box/2. );
        glVertex2f( Xlens-Box/2.,  Ylens+Box/2. );
glEnd( );
glEnable( GL_DEPTH_TEST );
```

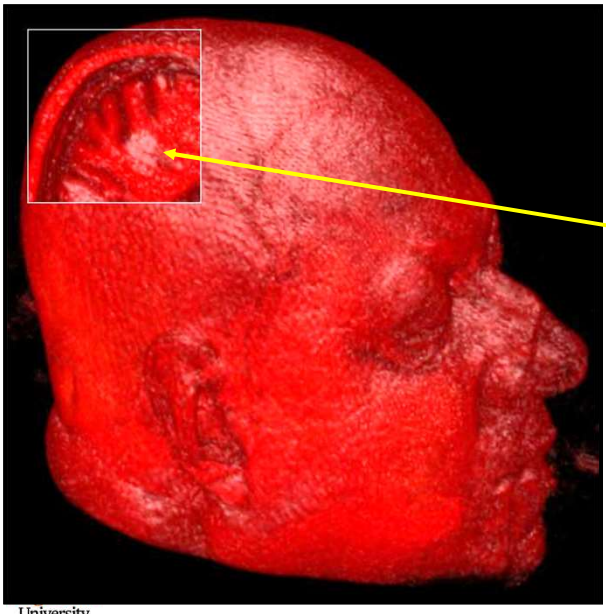
Draw the solids everywhere *except* inside the lens

Draw the wireframes *only* inside the lens

Draw the boundary of the lens

Oregon State University
Computer Graphic

I Once Used the Stencil Buffer to Create a *Magic Lens* for Volume Data

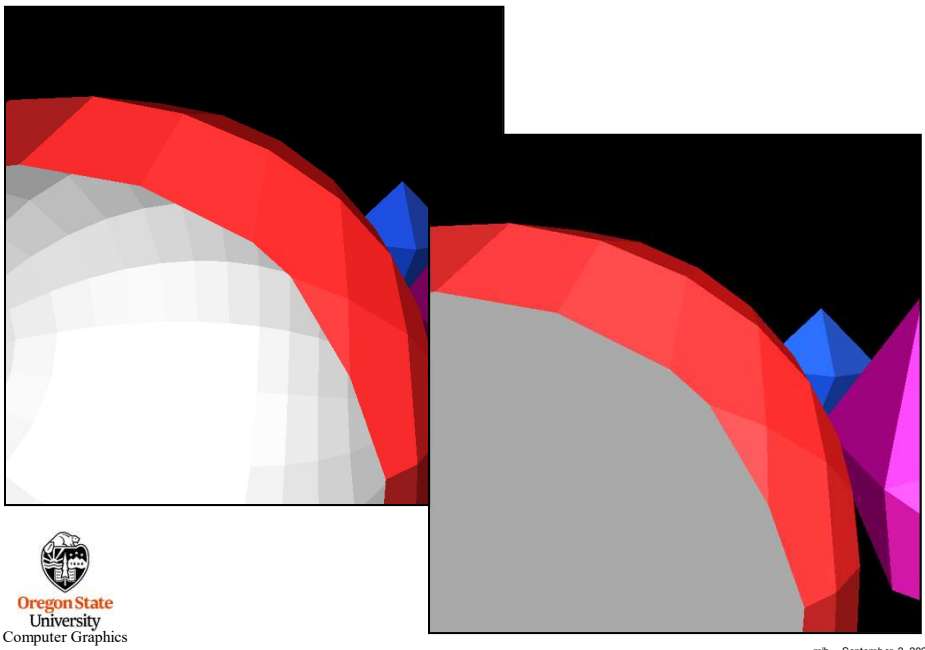


In this case, the scene inside the lens was created by drawing the same object, but drawing it with its near clipping plane positioned partways into the data

University
Computer Graphics

Using the Stencil Buffer to Perform *Polygon Capping*

13



Oregon State University
Computer Graphics

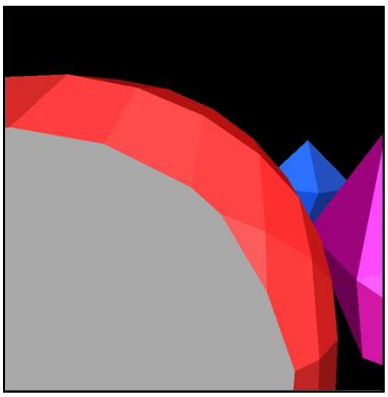
mjb – September 2, 2024

13

Using the Stencil Buffer to Perform *Polygon Capping*

14

1. Clear the SB = 0
2. Enable the SB
3. Draw the polygons, setting SB = ~ SB: 0's → 1's, 1's → 0's
4. Draw a large gray polygon in front of the entire scene wherever SB != 0
5. Disable the SB



Oregon State University
Computer Graphics

mjb – September 2, 2024

14

Using the Stencil Buffer to Perform *Polygon Capping*


```
glStencilFunc( GL_ALWAYS, 0, STENCILBIT );
glStencilOp( GL_INVERT, GL_INVERT, GL_INVERT );
<< draw all objects >>
```

As we draw the ***solid*** objects, always invert the stencil bits:
0's → 1's
1's→ 0's

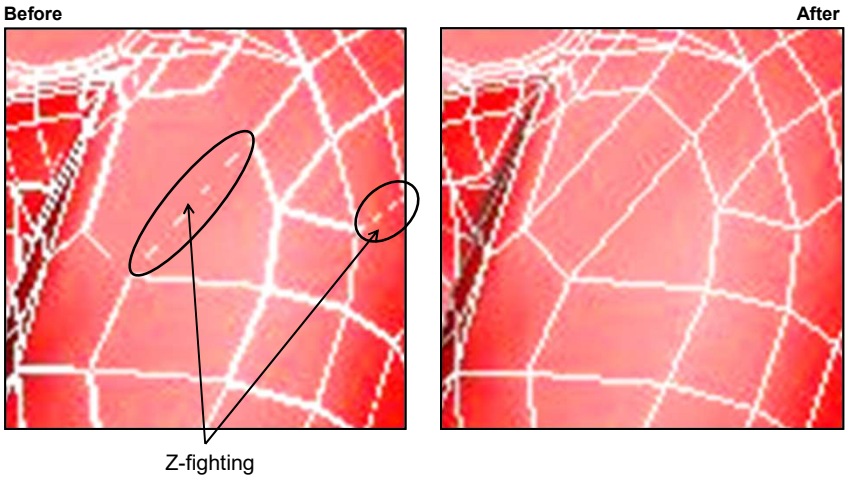
Because these were all ***solid*** objects, they had a front face and a back face drawn. Thus, most of the time, the SB values got inverted back to 0. If they didn't, that means that the solid object penetrated the near clipping plane and now needs to be capped.

```
glMatrixMode( GL_PROJECTION );
glLoadIdentity( );

glMatrixMode( GL_MODELVIEW );
glLoadIdentity( );

glDisable( GL_LIGHTING );
glDisable( GL_LIGHT0 );
glStencilFunc( GL_NOTEQUAL, 0, STENCILBIT );
glStencilOp( GL_KEEP, GL_KEEP, GL_KEEP );
glShadeModel( GL_FLAT );
glColor3f( .5f, .5f, .5f );
glBegin( GL_QUADS );
        glVertex3f( -BIGX, -BIGY, CLOSEZ );
        glVertex3f(  BIGX, -BIGY, CLOSEZ );
        glVertex3f(  BIGX,  BIGY, CLOSEZ );
        glVertex3f( -BIGX,  BIGY, CLOSEZ );
glEnd( );
```

Only draw the large gray plane in front where the SB != 0

Oregon State University
Computer Graphics

Using the Stencil Buffer to Draw Better Polygon Outlines



Oregon State University
Computer Graphics

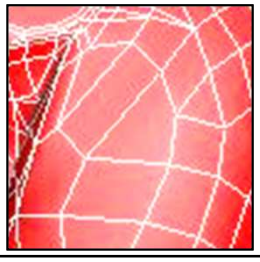
Using the Stencil Buffer to Draw Better Polygon Outlines

17

```
Clear the SB = 0

Enable the SB

for( each polygon )
{
        Draw the edges, setting SB = 1
        Draw the filled polygon wherever SB  !=  1
        Draw the edges again, setting SB = 0
}

Disable the SB
```

Before

After



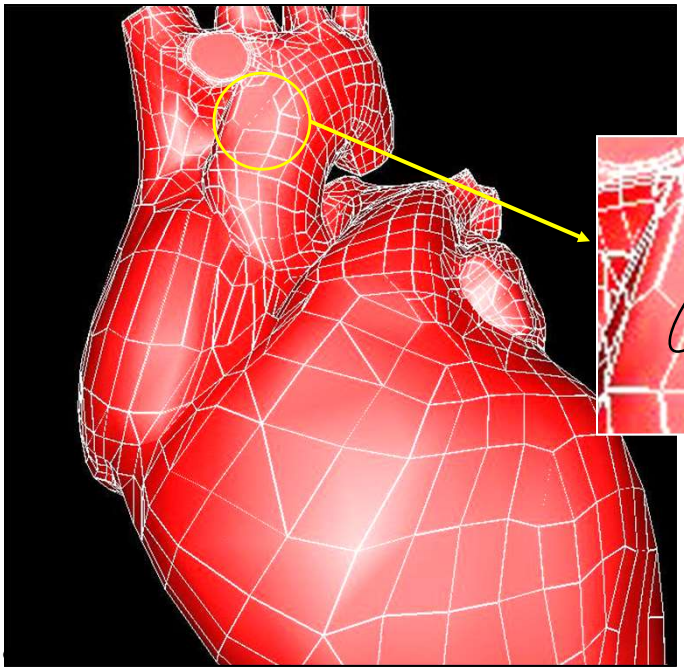
Oregon State University
Computer Graphics

mjb – September 2, 2024

17

Outlining Polygons the Naïve Way Results in Z-Fighting

18



1. Draw all polygons
2. Draw all edges

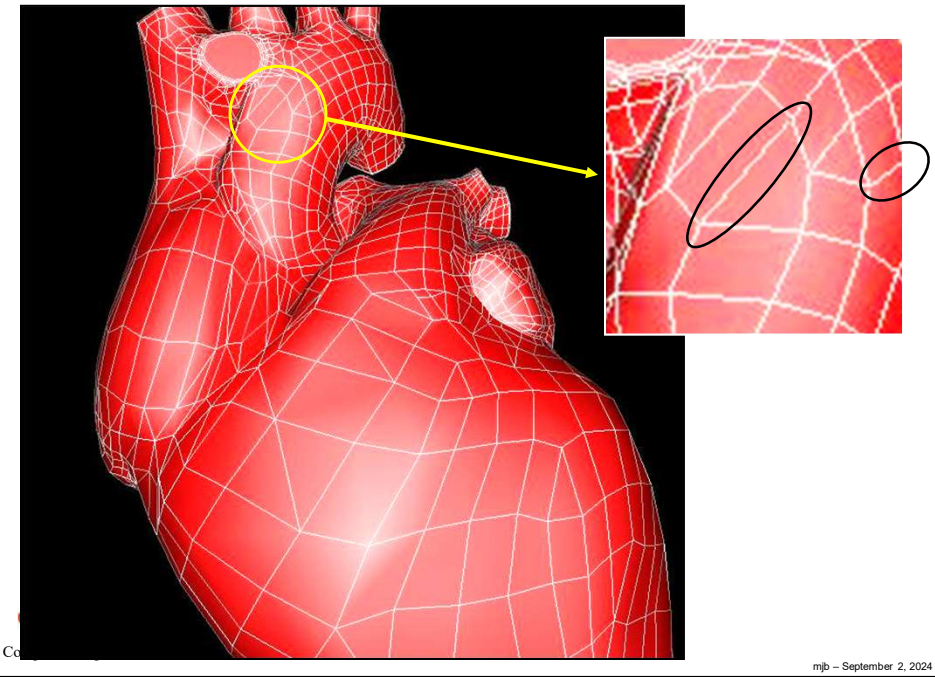
Z-fighting

mjb – September 2, 2024

18

Using the Stencil Buffer to Draw Better Polygon Outlines

19



mjb – September 2, 2024

19

Using the Stencil Buffer to Draw Better Polygon Outlines

20

```
for( int f = 0; f < NumFaces; f++ )
{
        glStencilFunc( GL_ALWAYS, 1, STENCILBIT );
        glStencilOp( GL_REPLACE, GL_ REPLACE, GL_ REPLACE );
        glDisable( GL_LIGHTING );
        glShadeModel( GL_FLAT );
        glColor3f( 1., 1., 1. );
        glBegin( GL_LINE_LOOP );
        for( int v = FirstVertex[f];  v < FirstVertex[f+1];  v++ )
        {
                glVertex3f( Vertices[v].x, Vertices[v].y, Vertices[v].z );
        }
        glEnd( );

        glStencilFunc( GL_EQUAL, 0, STENCILBIT );
        glStencilOp( GL_KEEP, GL_KEEP, GL_KEEP );
        glEnable( GL_LIGHTING );
        glShadeModel( GL_SMOOTH );
        glMaterialfv( … );
        glBegin( GL_POLYGON );
        for( int v = FirstVertex[f];  v < FirstVertex[f+1];  v++ )
        {
                glNormal3f( Normals[v].x, Normals[v].y, Normals[v].z );
                glVertex3f( Vertices[v].x,  Vertices[v].y, Vertices[v].z );
        }
        glEnd( );

        glStencilFunc( GL_ALWAYS, 0, STENCILBIT );
        glStencilOp( GL_ REPLACE, GL_ REPLACE, GL_ REPLACE );
        glDisable( GL_LIGHTING );
        glShadeModel( GL_FLAT );
        glColor3f( 1., 1., 1. );
        glBegin( GL_LINE_LOOP );
        for( int v = FirstVertex[f];  v < FirstVertex[f+1];  v++ )
        {
                glVertex3f( Vertices[v].x, Vertices[v].y, Vertices[v].z );
        }
        glEnd( );
}
```

Put "masking tape" down on the polygon edges

Paint the polygon, which also paints the edges

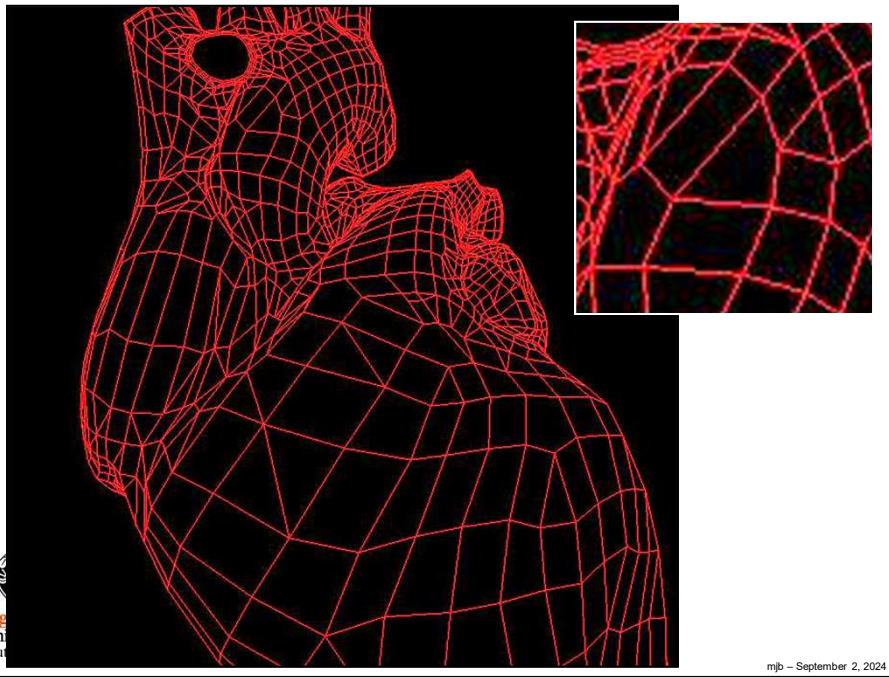
Pull the "masking tape" up and paint just the polygon edges

Oregon State University Computer Graphics

mjb – September 2, 2024

20

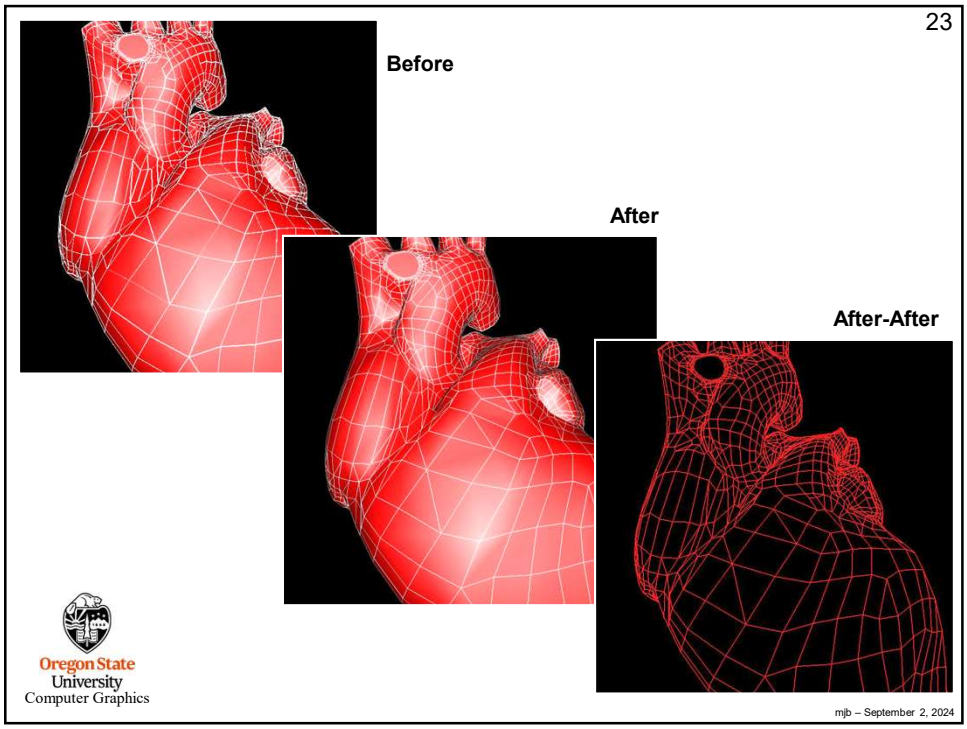
Using the Stencil Buffer to Perform *Hidden Line Removal*



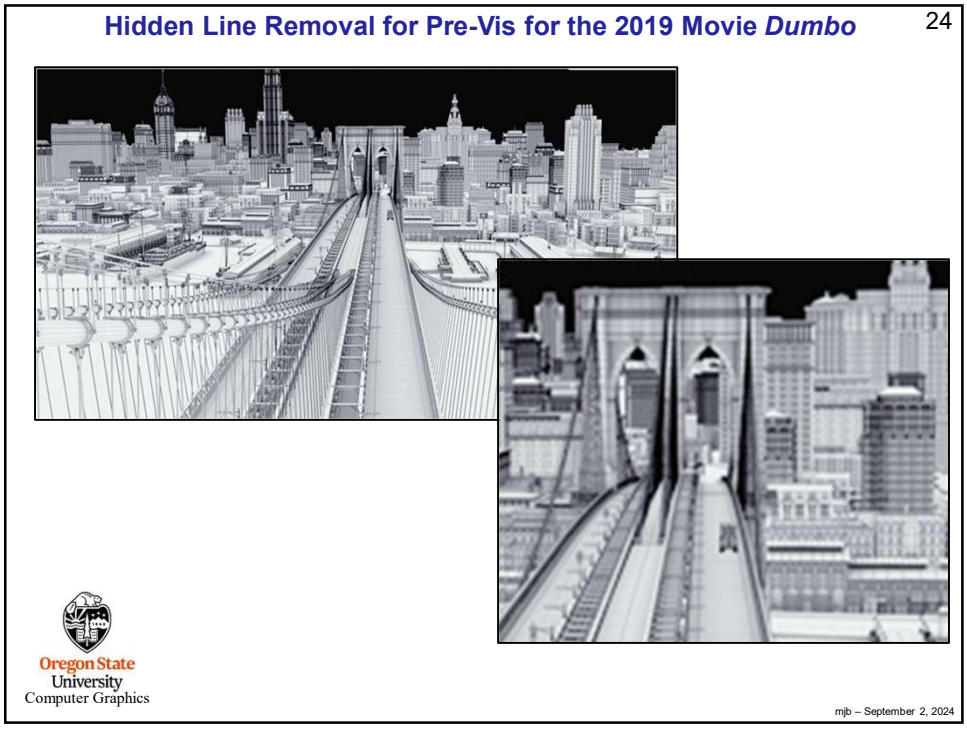
Using the Stencil Buffer to Perform *Hidden Line Removal*


```
Clear the SB = 0
Enable the SB

for( each polygon )
{
        Draw the edges, setting SB = 1
        Draw the polygon, unlit and flat shaded, in the background color wherever SB  !=  1
        Draw the edges again, setting SB = 0
}

Disable the SB
```

Oregon State University
Computer Graphics


23
Before
After
After-After
Oregon State
University
Computer Graphics
mjb – September 2, 2024

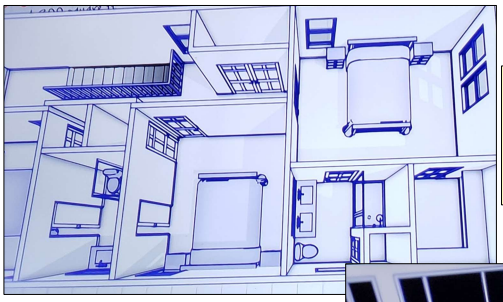
23


Hidden Line Removal for Pre-Vis for the 2019 Movie *Dumbo*
24
Oregon State
University
Computer Graphics
mjb – September 2, 2024

24

Hidden Line Removal in Construction Shows

25



I've noticed that some of the construction reality TV shows use hidden line removal like this, presumably to create a blueprint-ish effect. This came from the show *Good Bones*.

Oregon State University
Computer Graphics

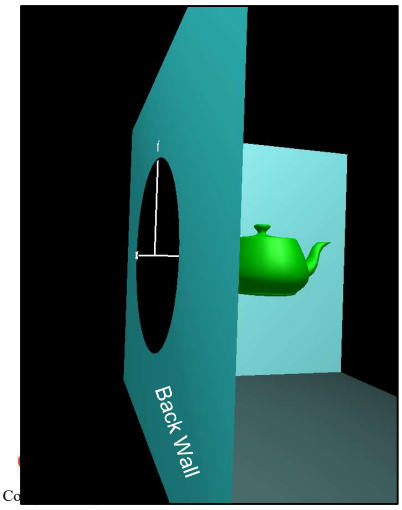
mjb – September 2, 2024

25

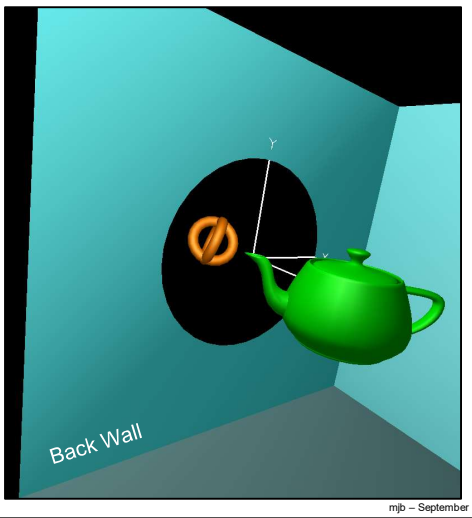
Stencil Buffers can be used to Create Portals

26

Here we have a room with a teapot in it and a hole in the Back Wall. You can tell that it is a hole because the axes obviously go through it. Clearly there is nothing behind the Back Wall.



But if we look at the scene from within the room, there is indeed a 3D object (the orange torii) seen through the hole. The hole is acting like a "portal" to another 3D space.



mjb – September 2, 2024

26

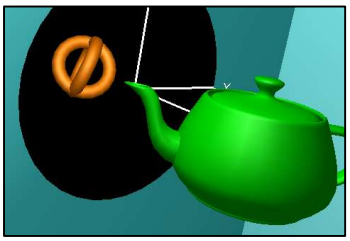
Stencil Buffers can be used to Create Portals

Here's the process:

1. Draw the teapot and the axes like normal
2. Turn off writing into the depth and color framebuffers
3. Draw an invisible large square behind the Back Wall setting SB=1
4. Draw an invisible circle on that wall setting SB=0
5. Draw an invisible large square behind the back Wall between there and the viewer setting SB=1
6. Draw the cyan room walls and the orange torii only where SB=0 – this makes the portal

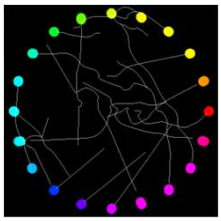
The first large square (SB=1) plus circle (SB=0) ends up making the Stencil Buffer look like this:

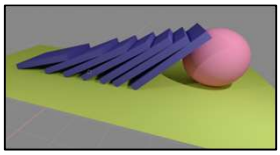
1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	1	1	1
1	1	1	1	1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1



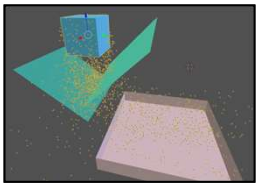
Oregon State University Computer Graphics

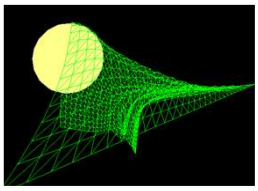
Animation





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





Oregon State University
Computer Graphics

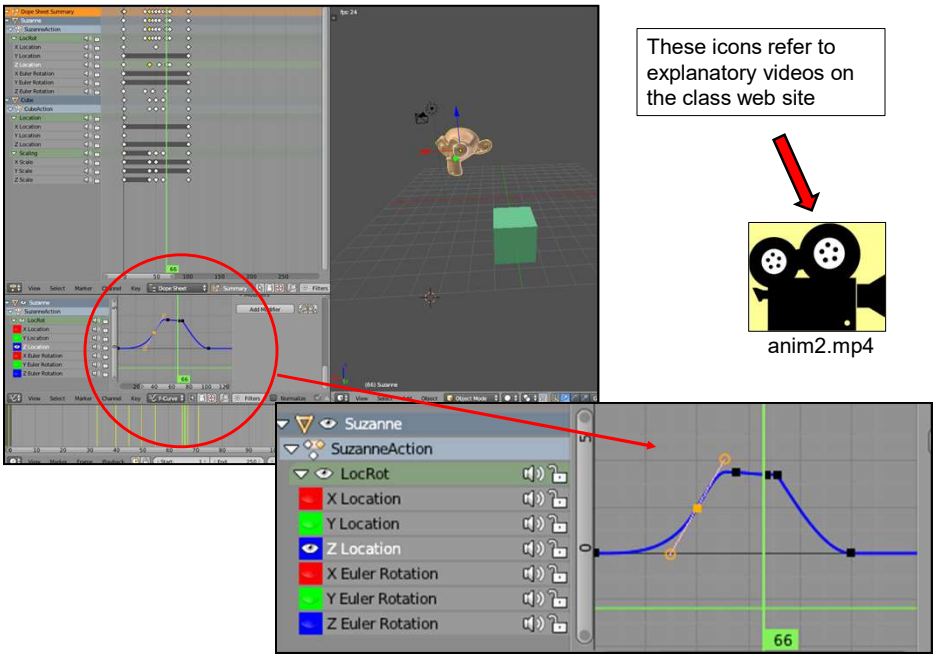
Animation

Animation is the process of giving motion to your geometric models. Before animating, 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? Partially? Which elements?
- 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?

Oregon State University
Computer Graphics

Keyframe Animation



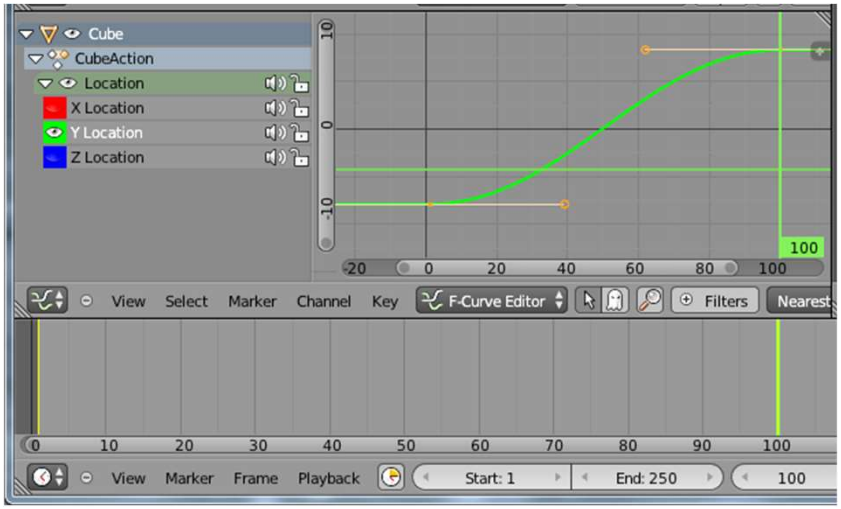
These icons refer to explanatory videos on the class web site

anim2.mp4

Oregon State University
Computer Graphics

Keyframe Animation

Blender:



Oregon State University
Computer Graphics

Here's Some Code that Lets You Create DIY Keyframe Animations

For my own work, instead of *Key Frames*, I like specifying *Key Times* better. And, so, I created a C++ class to do it for you.

```
class Keytimes:

        void    AddTimeValue( float time, float value );
        float   GetFirstTime( );
        float   GetLastTime( );
        int     GetNumKeytimes( );
        float   GetValue( float time );
        void    Init( );
        void    PrintTimeValues( );
```

Instead of Key *Frames*, I Like Specifying Key *Times* Better

```
Keytimes Xpos;

int
main( int argc, char *argv[ ] )
{
        Xpos.Init( );
        Xpos.AddTimeValue(  0.0,  0.000 );
        Xpos.AddTimeValue(  2.0,  0.333 );
        Xpos.AddTimeValue(  1.0,  3.142 );
        Xpos.AddTimeValue(  0.5,  2.718 );
        fprintf( stderr, "%d time-value pairs:\n", Xpos.GetNumKeytimes( ) );
        Xpos.PrintTimeValues( );

        fprintf( stderr, "Time runs from %8.3f to %8.3

        for( float t = 0.; t <= 2.0; t += 0.1 )
        {
                float v = Xpos.GetValue( t );
                fprintf( stderr, "%8.3f\t%8.3f\n", t, v );
        }
}
```

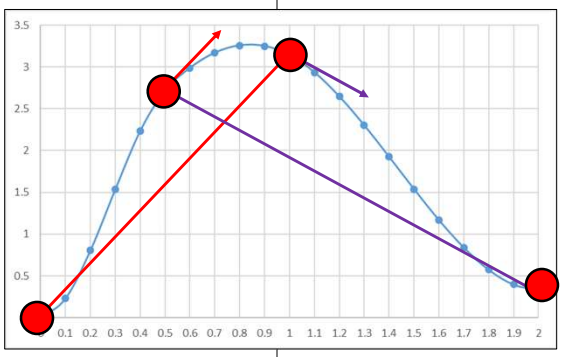
Declare one class per parameter you are animating. Here it wants to interpolate and animate the x-location of something.

Key-Time pairs (Time and Xlocation) can be specified in any order

Just to demonstrate, this for-loop runs through a collection of time values and looks up the interpolate x-location at those values. Normally you would get the time from the system clock.

Instead of Key *Frames*, I Like Specifying Key *Times* Better

```
( 0.00,  0.000)
( 0.00,  0.000)  ( 2.00,  0.333)
( 0.00,  0.000)  ( 1.00,  3.142)  ( 2.00,  0.333)
( 0.00,  0.000)  ( 0.50,  2.718)  ( 1.00,  3.142)  ( 2.00,  0.333)
4 time-value pairs
Time runs from   0.000 to   2.000
  0.000         0.000
  0.100         0.232
  0.200         0.806
  0.300         1.535
  0.400         2.234
  0.500         2.718
  0.600         2.989
  0.700         3.170
  0.800         3.258
  0.900         3.250
  1.000         3.142
  1.100         2.935
  1.200         2.646
  1.300         2.302
  1.400         1.924
  1.500         1.539
  1.600         1.169
  1.700         0.840
  1.800         0.574
  1.900         0.397
  2.000         0.333
```



= the originally-specified key-time pairs

Oregon State University
Computer Graphics

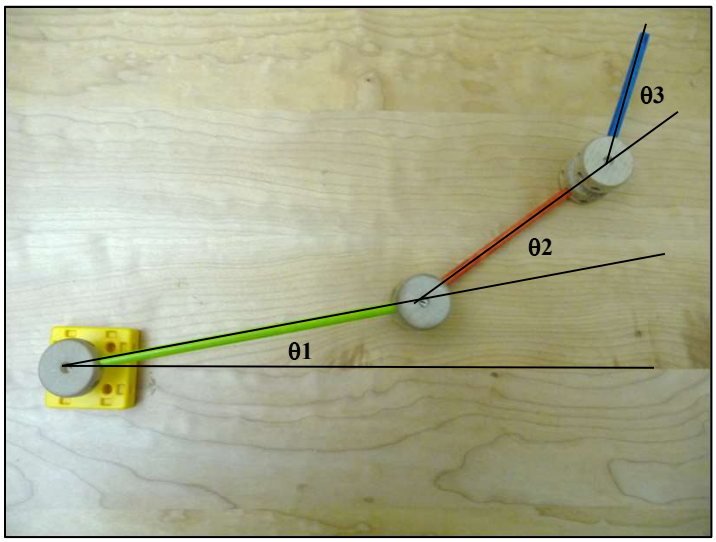
Using the System Clock in Display( ) for Timing

```
#define MSEC       10000                    // i.e., 10 seconds
Keytimes Xpos, Ypos, Zpos;
Keytimes ThetaX, ThetaY, ThetaZ;

. . .

if( AnimationIsOn )
{
      // # msec into the cycle ( 0 - MSEC-1 ):
      int msec = glutGet( GLUT_ELAPSED_TIME )  %  MSEC;

      // turn that into a time in seconds:
      float nowTime = (float)msec  / 1000.;
      glPushMatrix( );
            glTranslatef( Xpos.GetValue( nowTime ), Ypos.GetValue( nowTime ), Zpos.GetValue( nowTime )
);
            glRotatef( ThetaX.GetValue( nowTime ),  1., 0., 0. );
            glRotatef( ThetaY.GetValue( nowTime ),  0., 1., 0. );
            glRotatef( ThetaZ.GetValue( nowTime ),  0., 0., 1. );
            << draw the object >>
      glPopMatrix( );
}
```

Number of msec in the animation cycle

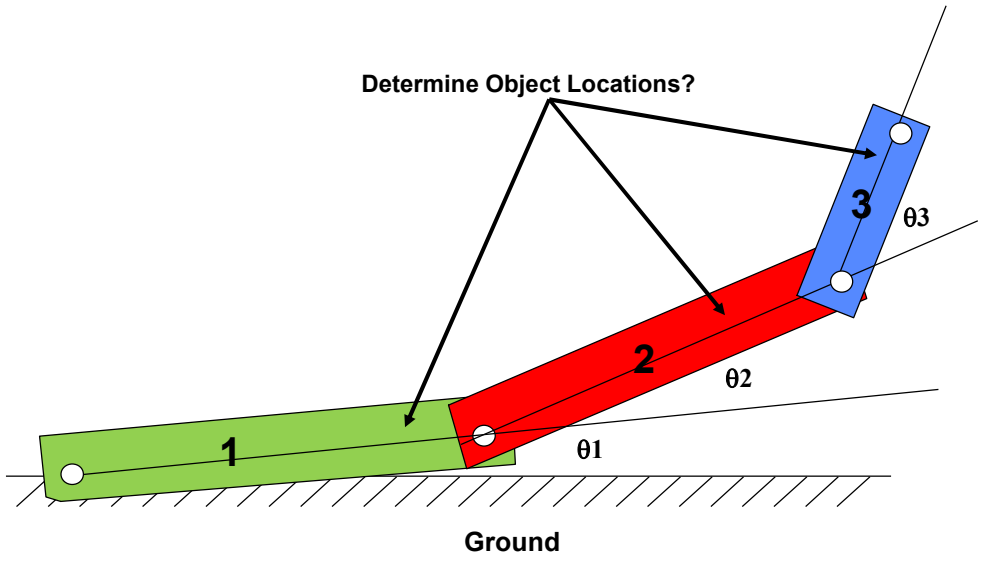
Oregon State University
Computer Graphics

Forward Kinematics:
Change Parameters – Connected Things Move
(All children understand this)



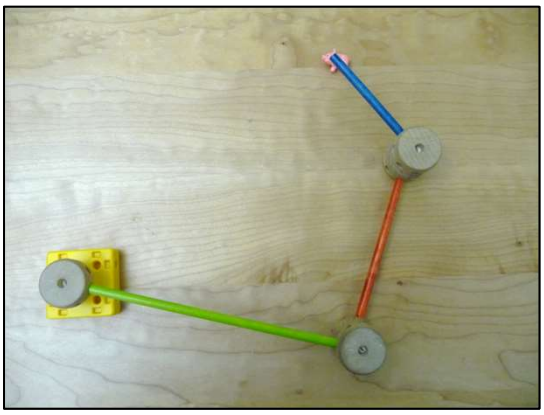
Oregon State University
Computer Graphics

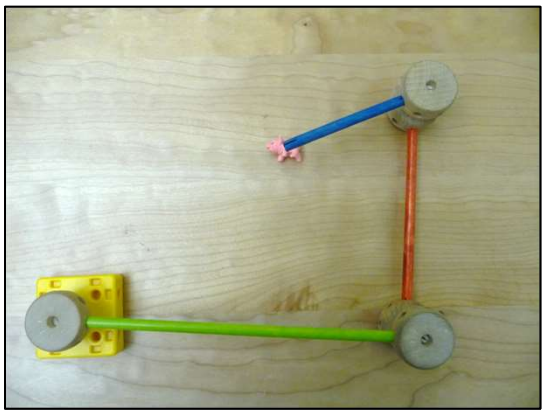
Forward Kinematics: Transformation Hierarchies



Oregon State University
Computer Graphics

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



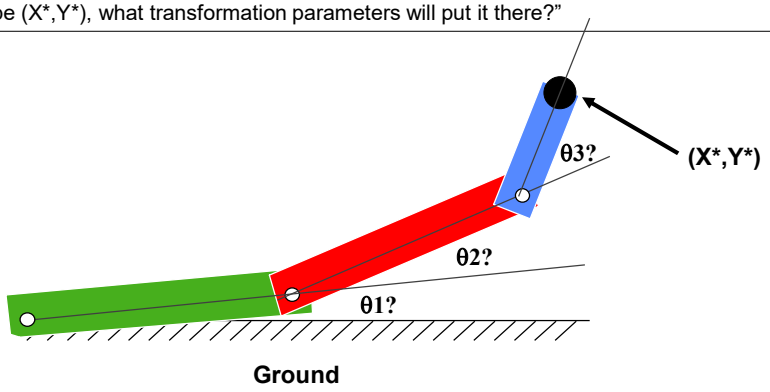


Of course, there will always be target locations that can *never* be reached.
Think about that spot in the middle of your back that you can never scratch! ☺

Inverse Kinematics (IK)

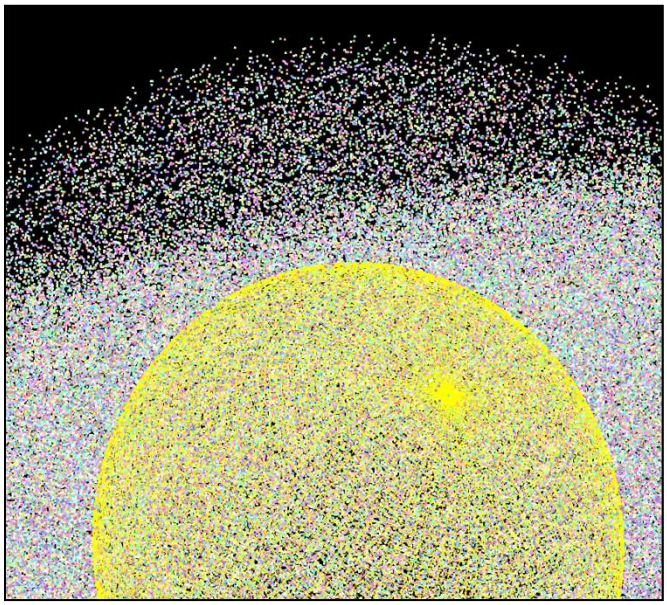
Forward Kinematics solves the problem "if I know the link transformation parameters, where are the links?".

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?"



Particle Systems:
A Cross Between Modeling and Animation?

13



Oregon State University
Computer Graphics

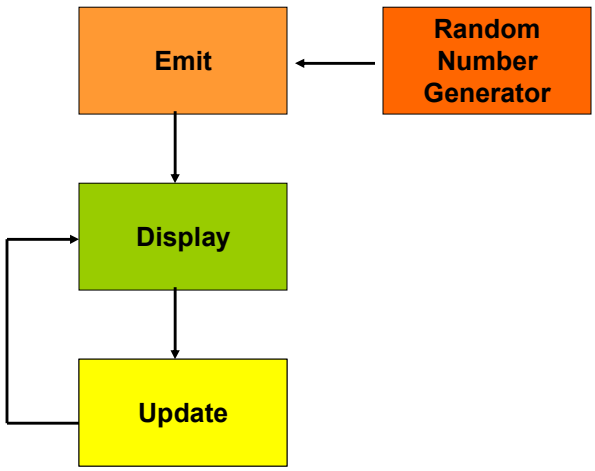
mjb – August 30, 2024

13

Particle Systems:
A Cross Between Modeling and Animation?

14

The basic process is:



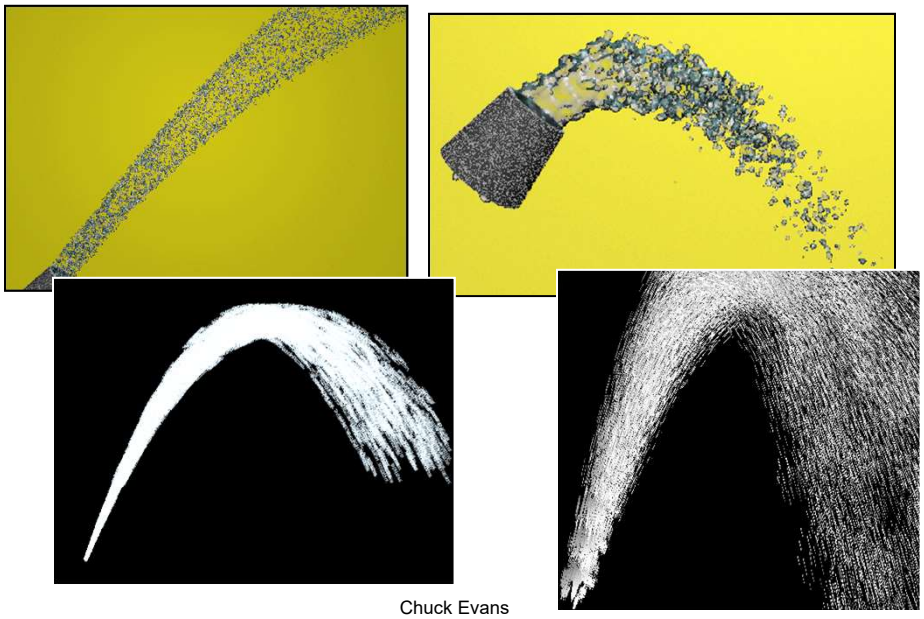
Oregon State University
Computer Graphics

mjb – August 30, 2024

14

15

Particle Systems Examples



Chuck Evans

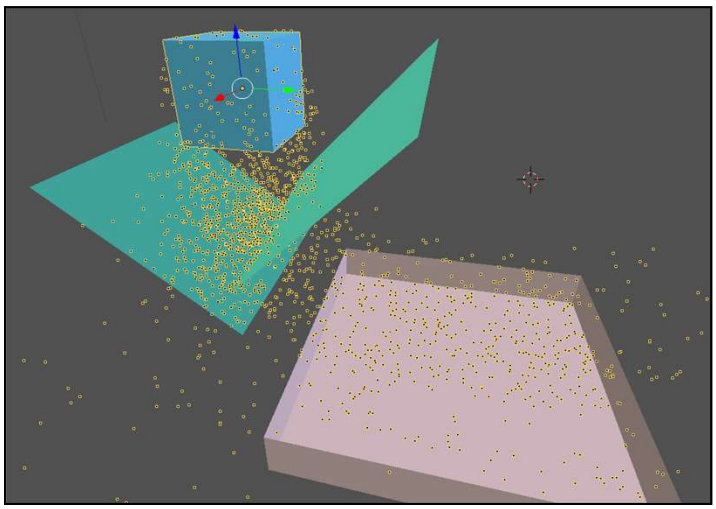
Oregon State University
Computer Graphics

mjb – August 30, 2024

15

16

Particle Systems Examples



particles.mp4

Oregon State University
Computer Graphics

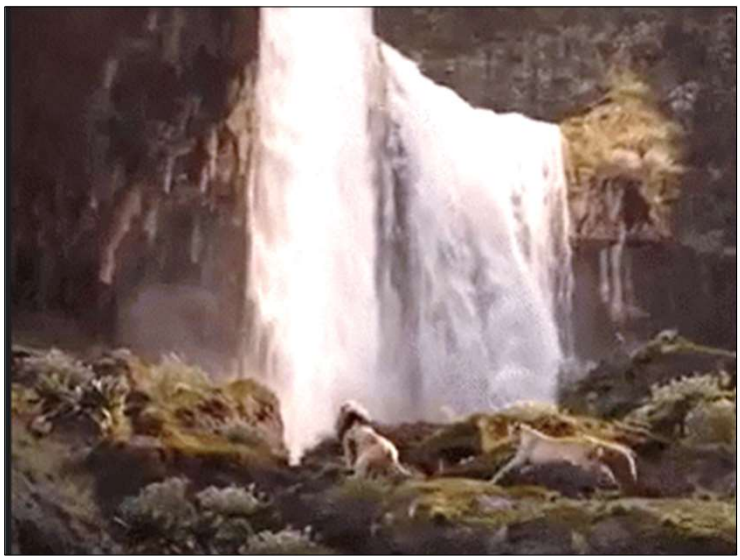
mjb – August 30, 2024

16

Particle Systems Examples

Oregon State University
Computer Graphics

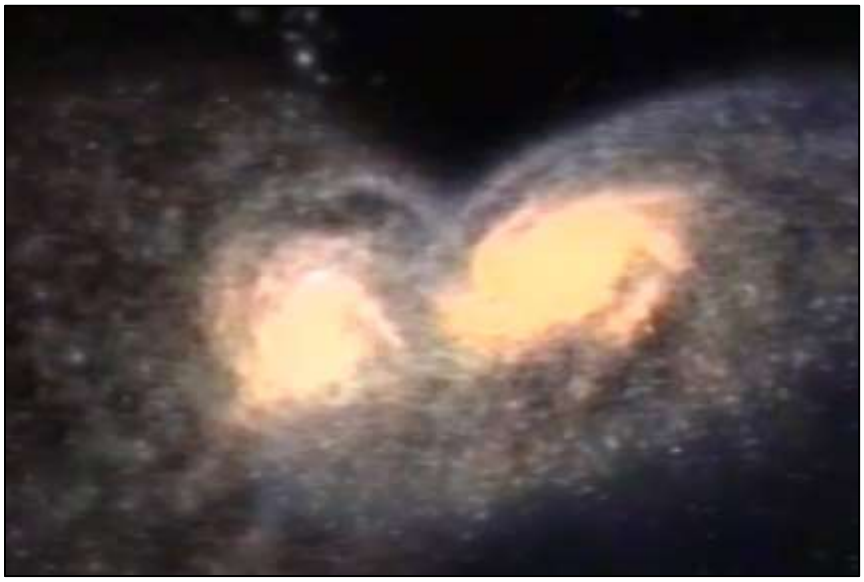
Particle Systems Examples



The Lion King (2019) -- Disney

Oregon State University
Computer Graphics

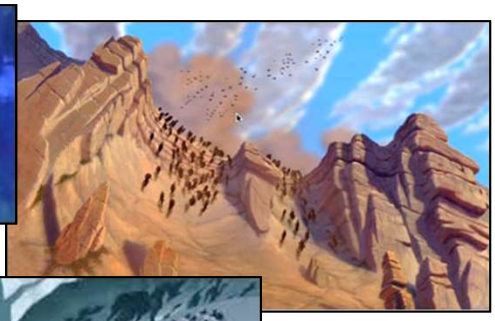
A Particle System to Simulate Colliding Galaxies in *Cosmic Voyage*



Cosmic Voyage

Particles Don't Actually Have to Be "Particles"

Avatar

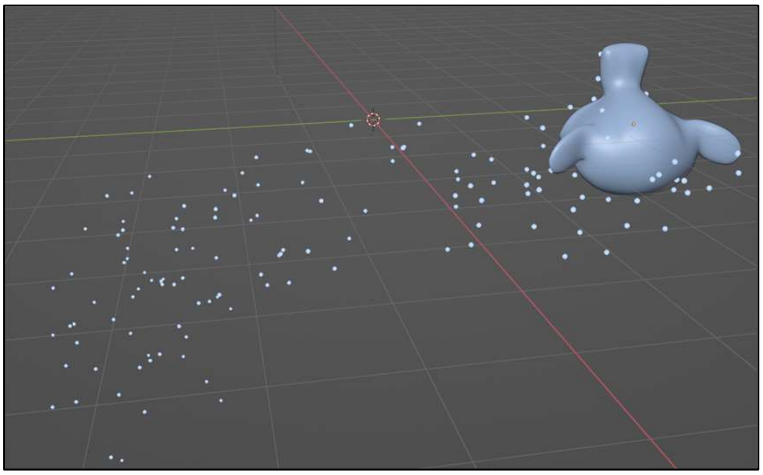


The Lion King

Mulan

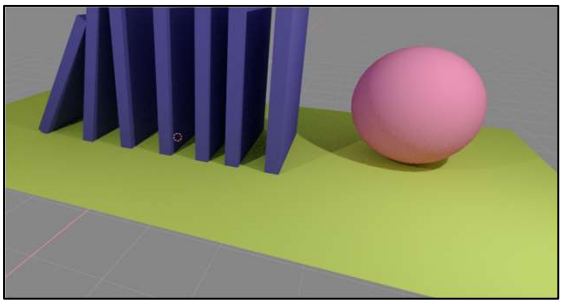
Multiple Animation Techniques Can Be Combined

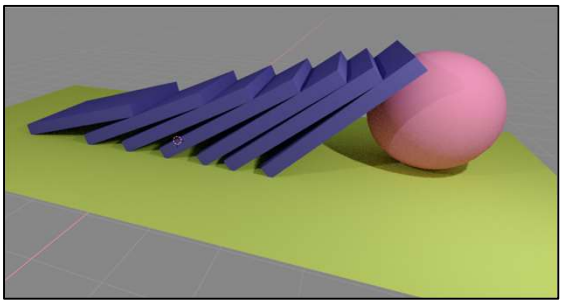
A Particle System coming from a moving keyframed object in Blender:



Oregon State University Computer Graphics

Animating using Rigid-body Physics





Newton's second law:

force = mass * acceleration

or

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

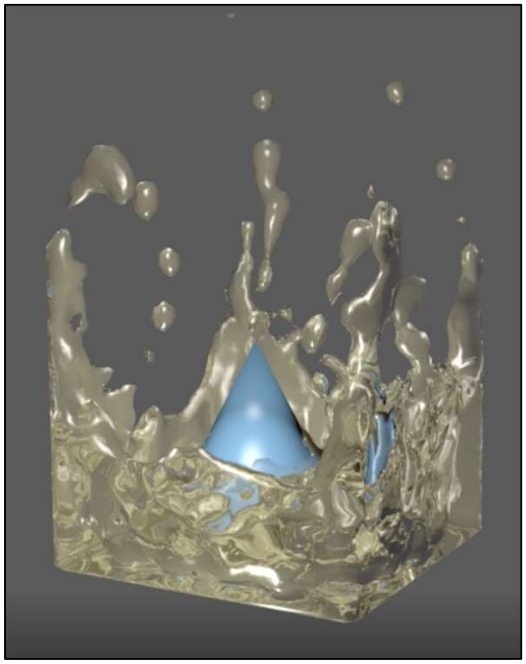
$$x(T) = \iint_{t=0}^{t=T} \ddot{x}\,dt \approx \sum \sum \ddot{x}\,\Delta t$$

dominos.mp4

Oregon State University Computer Graphics

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

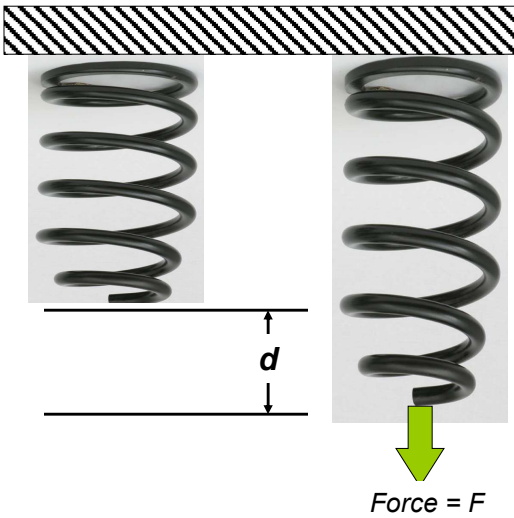
Animating using Fluid Physics



fluid.avi

Oregon State University
Computer Graphics

Animating using Physics



$$d = \frac{-F}{k}$$

d = ***spring displacement*** in inches

k = ***spring stiffness*** in pounds/inch

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

$$F = -kd$$

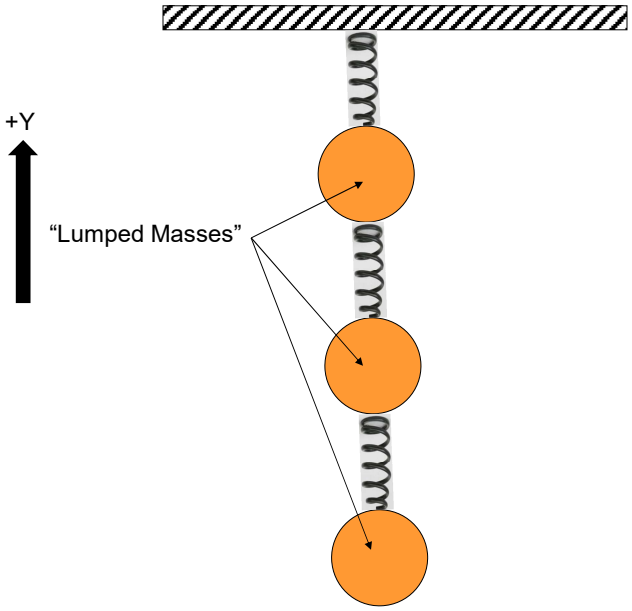
This is known as **Hooke's Law**

Why the minus sign? Because a spring's force acts in a direction opposite the displacement. In other words, a spring's force tries to correct its displacement.

Oregon State University
Computer Graphics

Animating using the Physics of a Mesh of Springs

25



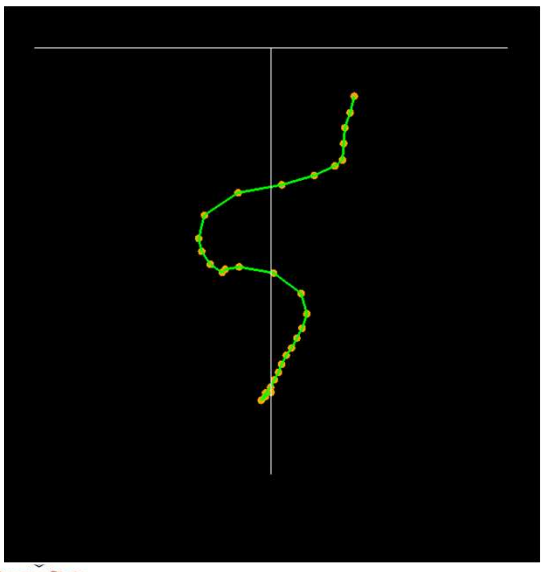
Oregon State University
Computer Graphics

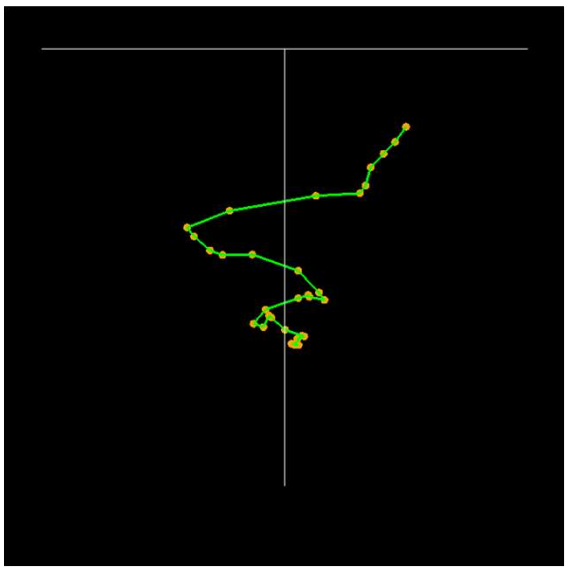
mjb – August 30, 2024

25

Simulating a Bouncy String or a Chain

26





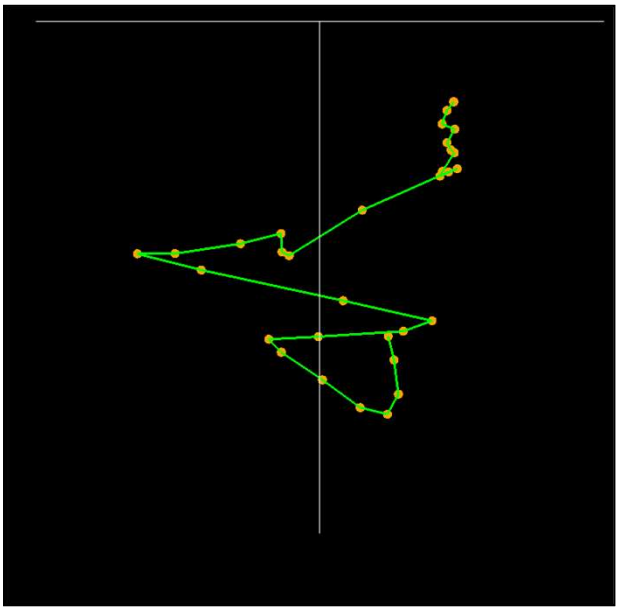
Oregon State University
Computer Graphics

mjb – August 30, 2024

26

27

Simulating a Bouncy String or a Chain



string.mp4

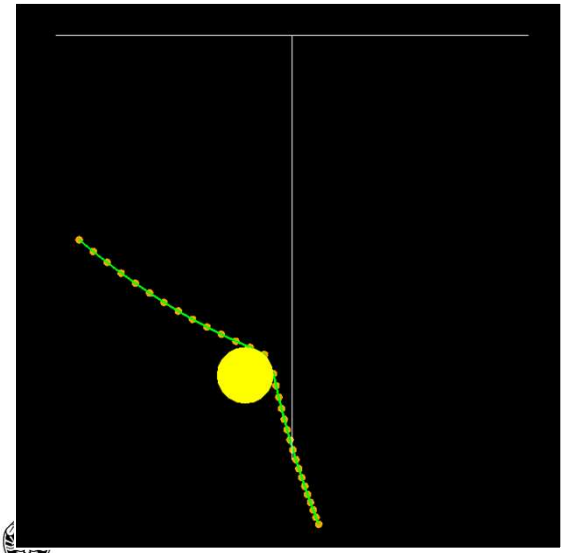
Oregon State University
Computer Graphics

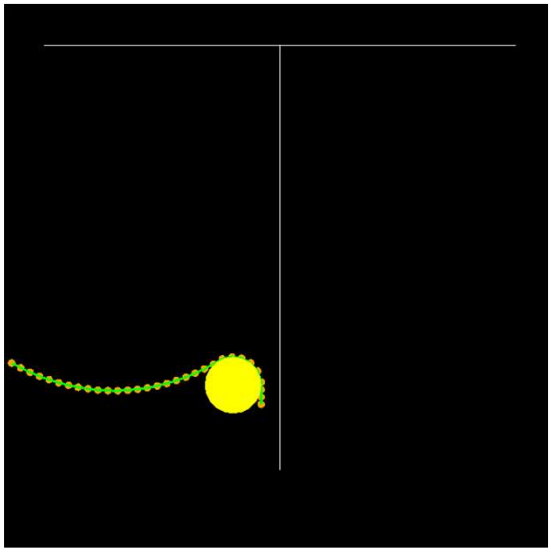
mjb – August 30, 2024

27

28

Placing a Physical Barrier in the Scene





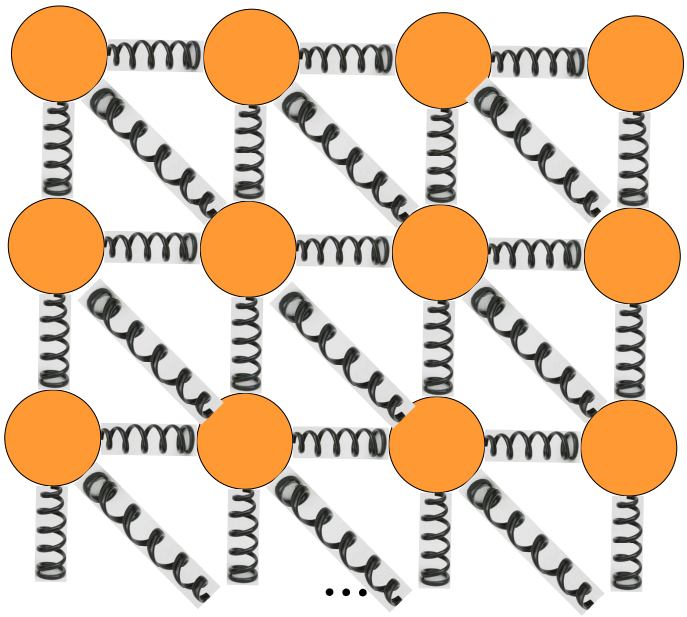
Oregon State University
Computer Graphics

mjb – August 30, 2024

28

29

Animating Cloth



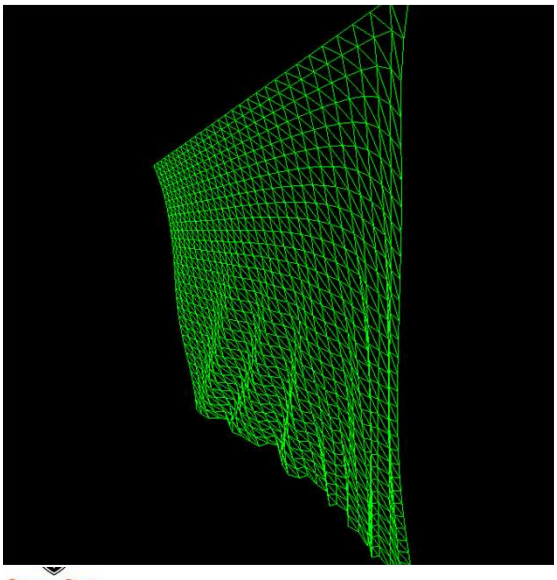
Oregon State University
Computer Graphics

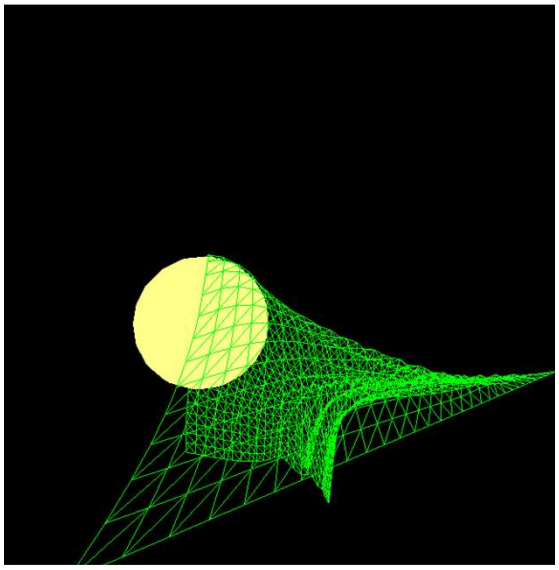
mjb – August 30, 2024

29

30

Cloth Examples





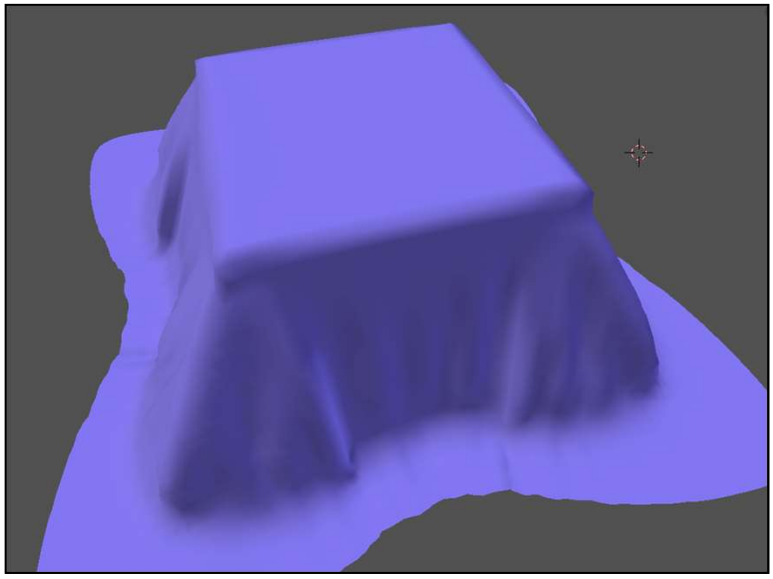
Oregon State University
Computer Graphics

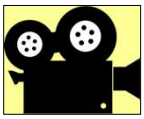
mjb – August 30, 2024

30

31

Cloth Example





cloth.mp4

Oregon State
University
Computer Graphics

mjb – August 30, 2024

31

32

Cloth Example

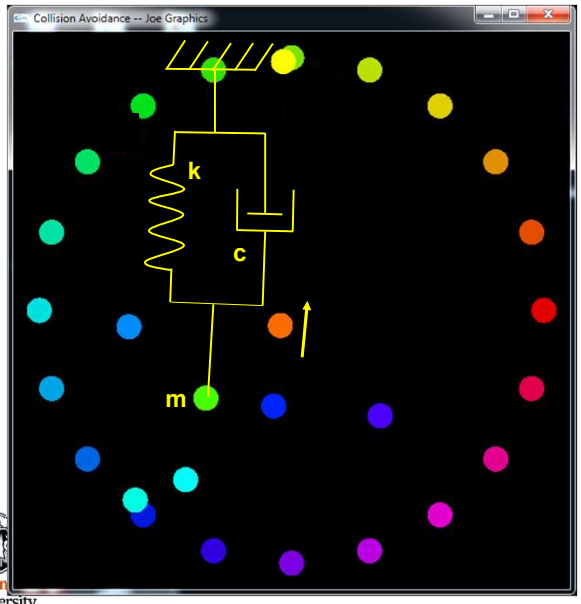
Geri's Game, Pixar

Oregon State
University
Computer Graphics

mjb – August 30, 2024

32

Functional Animation:
Setup Imaginary Physics to Make the Object *Want* to Move Towards a Goal Position



Newton's second law:

force = mass * acceleration

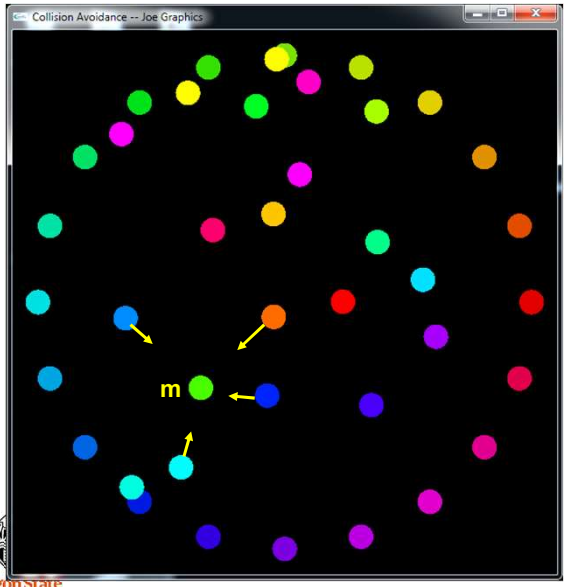
or

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

$$\ddot{x} = \frac{-c\dot{x} - kx}{m}$$

- The ***k*** is an imaginary spring.
- The ***c*** is an imaginary "damper", which you can think of as a shock absorber which absorbs energy. (Your car has these to soften bouncing.)

Functional Animation:
Setup Imaginary Physics to Make the Object *Want* to Move Away from all other Objects



Newton's second law:

force = mass * acceleration

or

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

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

Repulsion Coefficient

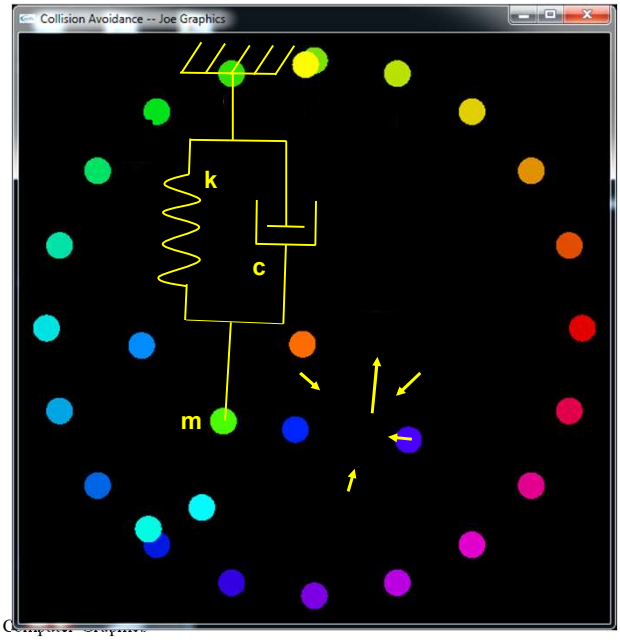
$$F_{repulsive} = \frac{C_{repulse}}{d^{Power}}$$

Distance between the boundaries of the 2 bodies

Repulsion Exponent

This isn't from a book. I just made this up. It seemed like a good idea.

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



Newton's second law:

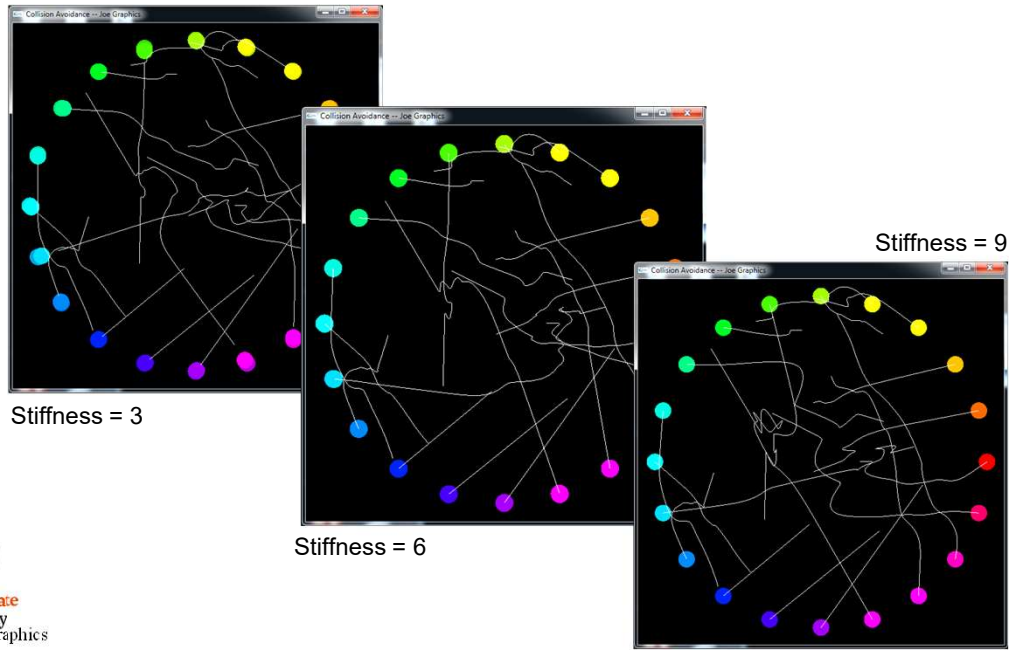
force = mass * acceleration

or

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

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

Increasing the Stiffness



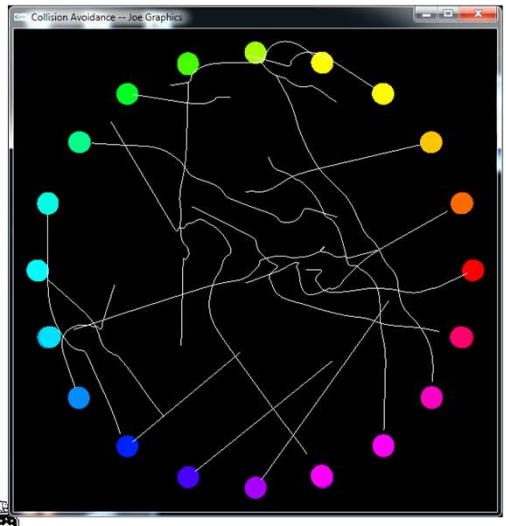
Stiffness = 3

Stiffness = 6

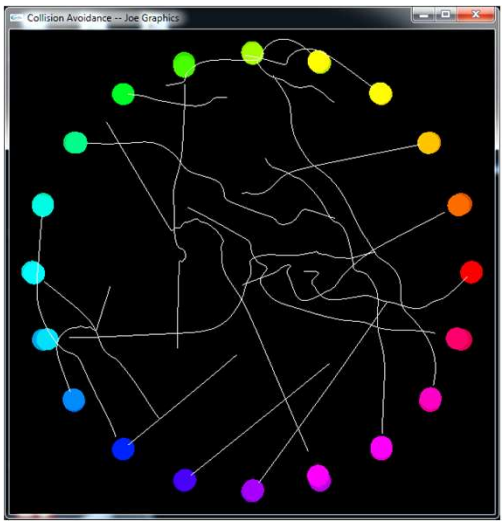
Stiffness = 9

Oregon State University Computer Graphics

Increasing the Repulsion Coefficient



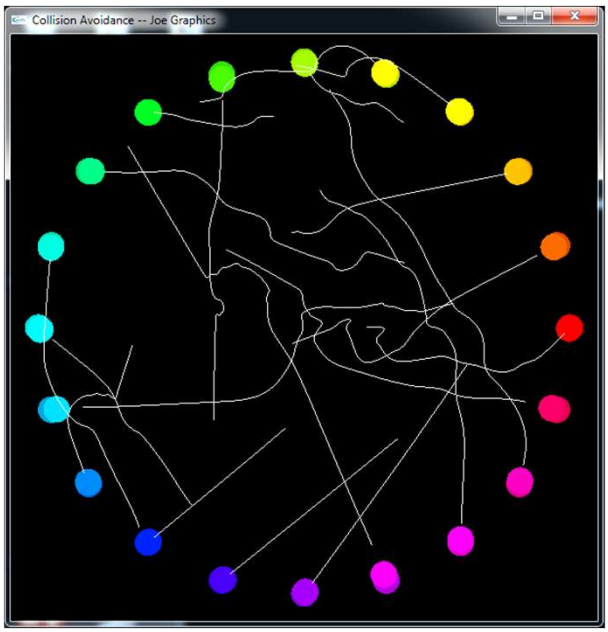
Repulse = 10



Repulse = 30

Oregon State University
Computer Graphics

Functional Animation:
Setup Imaginary Physics to Make the Object *Want* to Move Like We Want it To

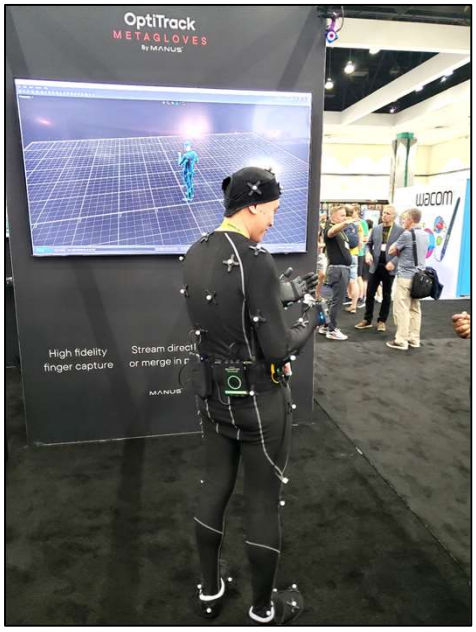


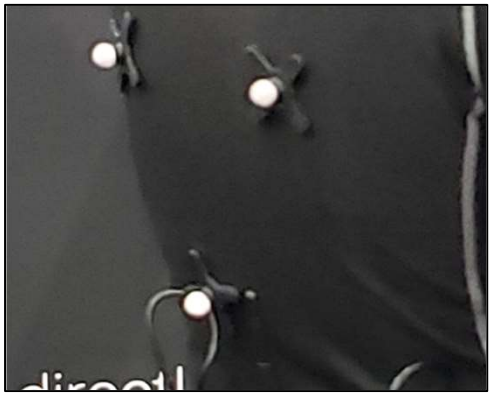
avoid.mp4

Oregon State University
Computer Graphics

39

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





Natural Point

Oregon State University
Computer Graphics

mjb – August 30, 2024

39

40

Motion Capture is for Hands and Faces Too

Natural Point

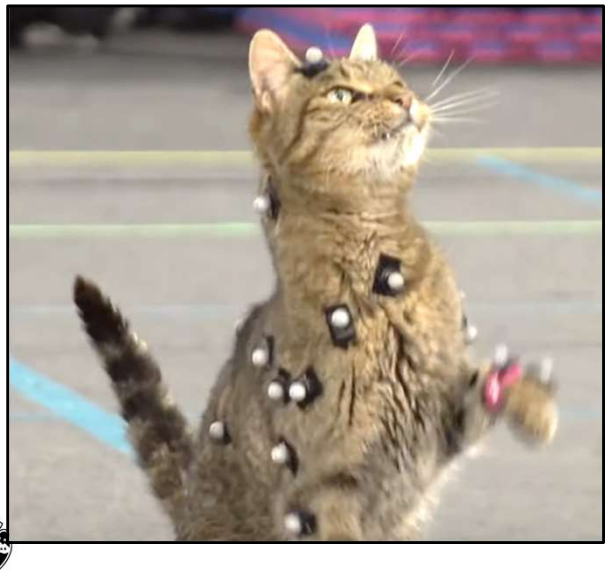
Oregon State University
Computer Graphics

mjb – August 30, 2024

40

41

Even Animals can be MoCapped



My cats would never have put up with this…

Oregon State University
Computer Graphics

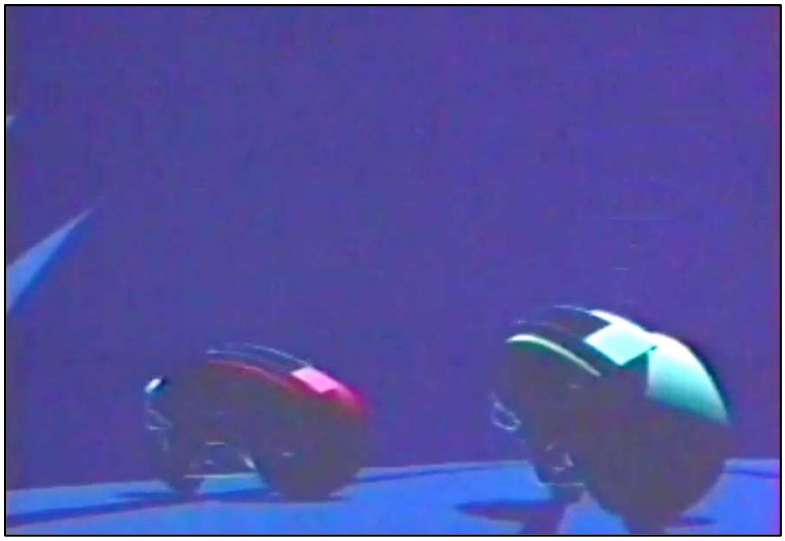
https://www.youtube.com/watch?v=zyq_LQrHpoo

mjb – August 30, 2024

41

42

Tron I –
They probably should have used physics, but didn't



MAGI

Oregon State University
Computer Graphics

mjb – August 30, 2024

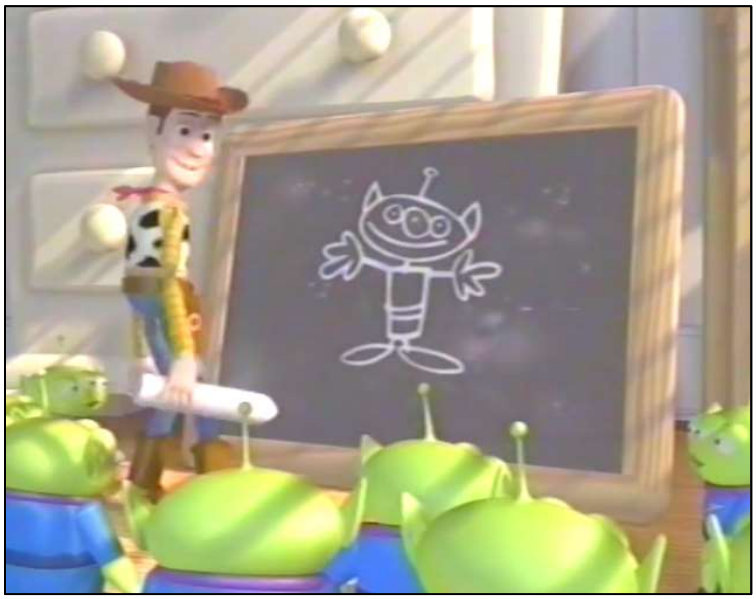
42

Card Trick

Rob Russ

Oregon State University Computer Graphics

Pixar Animated Shorts



Pixar

Oregon State University Computer Graphics

3D Printing

(which I consider to be a legitimate form of Computer Graphics display…)

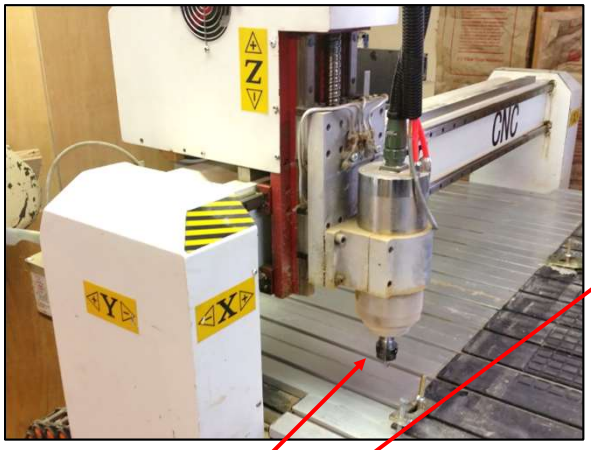
1

Mike Bailey

mjb@cs.oregonstate.edu

In the Beginning, All Manufacturing was "Subtractive"

2



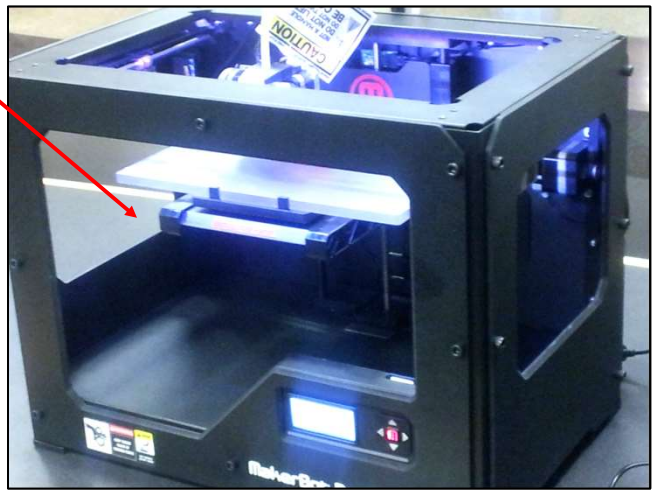
1. The whirling drill bit follows a 3D path around a block of metal, wax, or wood
2. Chips fly
3. A block of metal becomes a part

Today's 3D Printing Process

3

"3D Printing" is generally considered to be some sort of **"Additive**" process in which layers of material get deposited on previous layers. (Additive manufacturing is also sometimes called *Stereolithography*.)

The current frenzy in 3D Printing consists mostly of desktop systems that deposit layers of molten plastic, like this one. But there are many others.



Oregon State University
Computer Graphics

mjb – November 20, 2024

3

Examples of 3D Printing using a Variety of Materials

4

Oregon State University
Computer Graphics

mjb – November 20, 2024

4

Portland's Laika uses Color 3D Printing to Make Faces for Stop-motion Movies

5

From *Kubo and the Two Strings*

Oregon State University
Computer Graphics

5

The 3D Printing Geometry File

6

3D Printers are fed a file called an "STL File", which lists all the triangles in the object. All 3D CAD-ish systems (as well as TinkerCad, Thingiverse, and Blender) can produce this type of file for you.

```
solid

 facet normal   0.00   0.00   -1.00
  outer loop
    vertex   -2.000000   -2.000000   0.250000
    vertex   -1.980000   -1.980000   0.250000
    vertex   -1.980000   -2.000000   0.250000
  endloop
 endfacet

 facet normal   0.00   0.00   -1.00
  outer loop
    vertex   -2.000000   -2.000000   0.250000
    vertex   -2.000000   -1.980000   0.250000
    vertex   -1.980000   -1.980000   0.250000
  endloop
 endfacet

       . . .

endsolid
```

In this particular file, these coordinates were in units of inches.

Some 3D Printers still use **inches**, but most now seem to use **millimeters**.

Check! It matters!

Note: there are 25.4 mm/inch

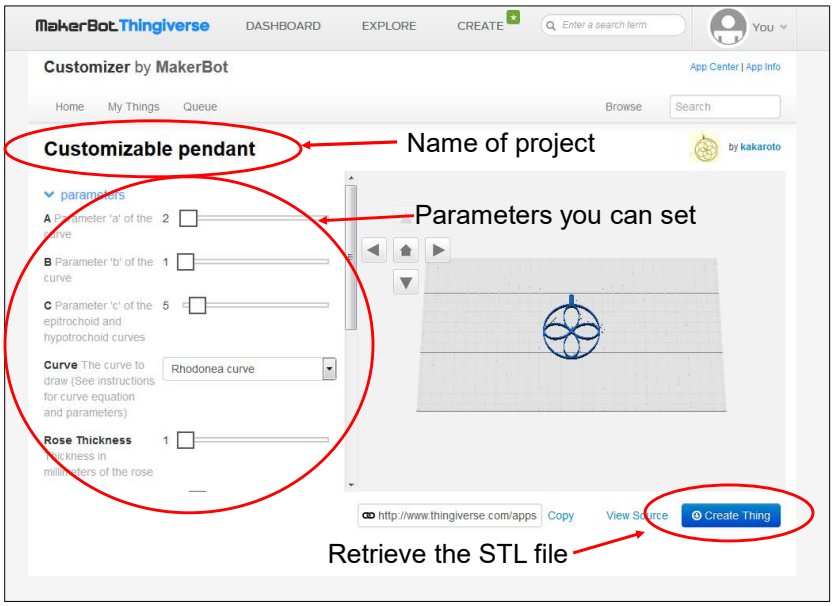
If you sent this file to a millimeter-based 3D printer, the part would come out very, very tiny! ☺

Oregon State University
Computer Graphics

6

thingiverse.com

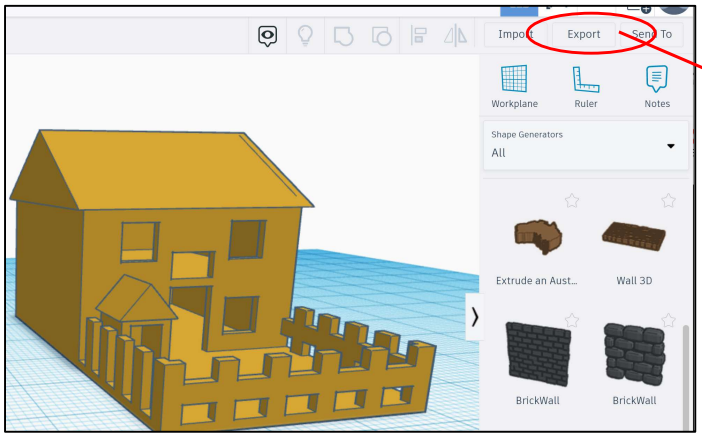
7

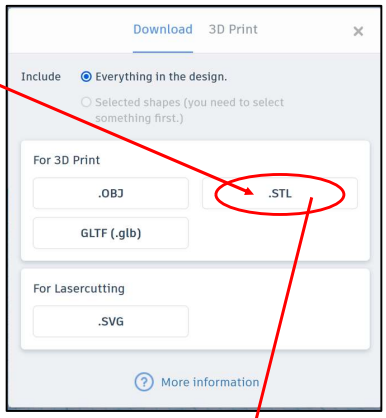


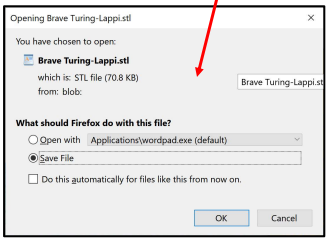
Oregon State University
Computer Graphics

TinkerCad

8







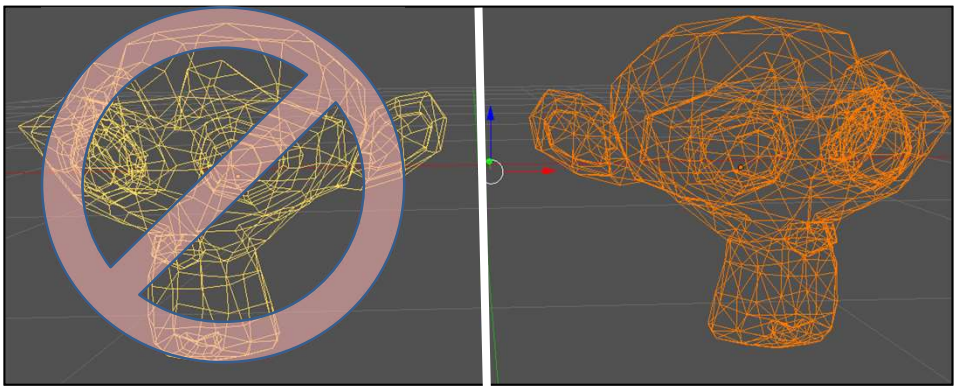
Click on **Import→STL**

Oregon State University
Computer Graphics

Object Rules for 3D Printing

9

1. The object must be a mesh and ***consist only of triangles***.



Quads and triangles **Triangles only**

Oregon State University
Computer Graphics

In Blender: Modifiers → Add Modifier → Triangulate

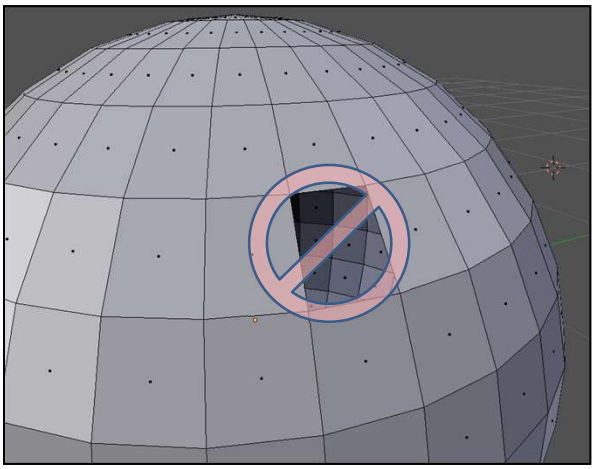
mjb – November 20, 2024

9

Object Rules for 3D Printing

10

2. 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.



Oregon State University
Computer Graphics

"Definite inside and outside" is sometimes called "Two-manifold" or "Watertight"

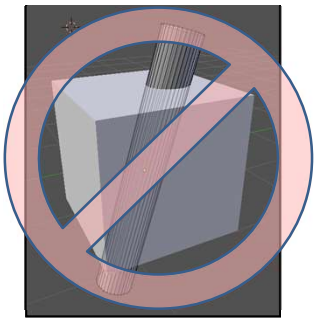
mjb – November 20, 2024

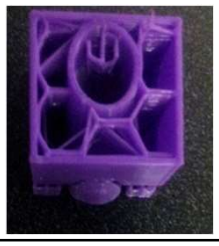
10

Object Modeling Rules for 3D Printing

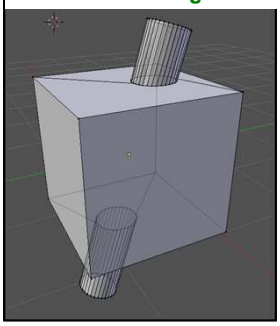
3. You can't make a compound object by simply overlapping two objects in 3D. If you want both shapes together, do a Boolean union on them so that they become one complete, legal object.

Overlapped in 3D -- bad





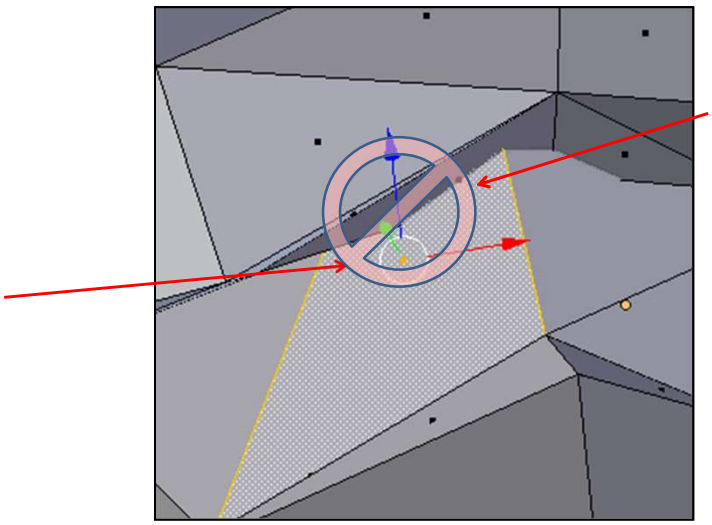
Boolean union -- good



Oregon State University
Computer Graphics

Object Rules for 3D Printing

4. Each edge in the mesh must bound 2 and only 2 triangles (this is known as the **Vertex-to-Vertex Rule**)



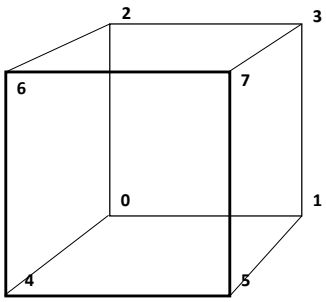
Oregon State University
Computer Graphics

The Simplified Euler's Formula* for Legal Solids

13

$$F - E + V = 2$$

F	Faces
E	Edges
V	Vertices



For a cube: 6 – 12 + 8 = 2

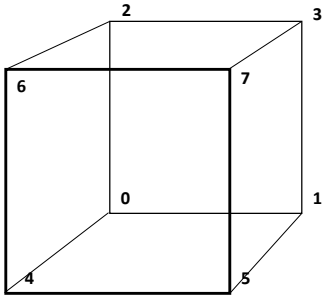
*sometimes called the Euler-Poincaré formula

The Full Euler's Formula* for Legal Solids

14

$$F - E + V - L = 2(B - G)$$

F	Faces
E	Edges
V	Vertices
L	Inner Edge Loops (within faces)
B	Bodies
G	Genus (number of through-holes)



For a cube: 6 – 12 + 8 – 0 = 2(1 – 0)

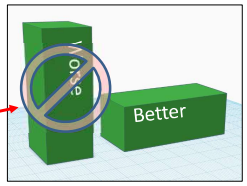
*sometimes called the Euler-Poincaré formula

Advice on 3D Printing

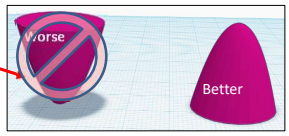
15

Don't make the part too big to start – it will take a long time to 3D print. It's nice if you can fit several models in a single run.

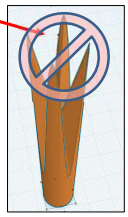
Try to rotate the part so the smallest dimension is vertical. It's stronger that way, and it builds faster.



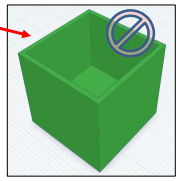
The 3D Printer will like it better if the part gets smaller as it goes up, not the other way around.



Don't design the part with long, thin edges. They will snap right off.



Don't make walls too thin – they will break.



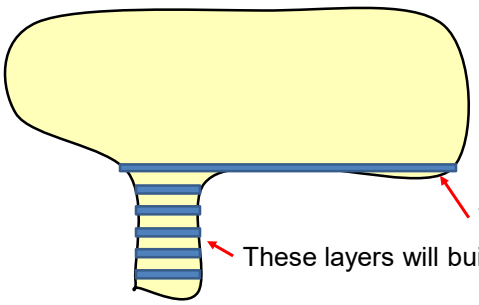
Oregon State University
Computer Graphics

mjb – November 20, 2024

15

Watch Out for Overhangs!

16



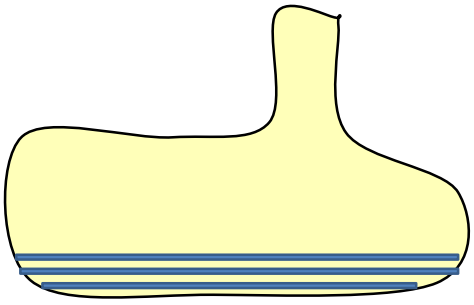
Oregon State University
Computer Graphics

mjb – November 20, 2024

16

Note that, if you build this object upside-down, it will probably be fine

17



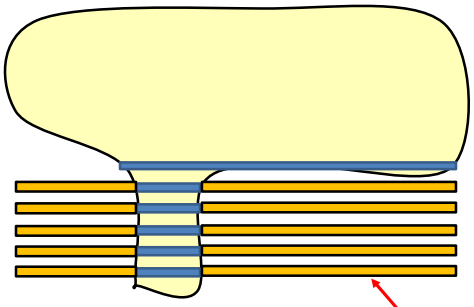
Oregon State University
Computer Graphics

mjb – November 20, 2024

17

Watch Out for Overhangs!

18



Some 3D printers handle overhangs by leaving unused material in place to support the overhangs

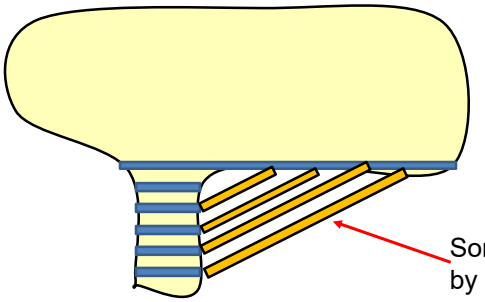
Oregon State University
Computer Graphics

mjb – November 20, 2024

18

Watch Out for Overhangs!

19



Some 3D printers handle overhangs by using software to add “support structures” to the overhangs

Some 3D printers handle this better than others…

Oregon State University
Computer Graphics

mjb – November 20, 2024

19

What Happens if You Don’t Follow the Rules?

20

Check here:
http://twistedsifter.com/2013/08/when-3d-printing-goes-wrong/

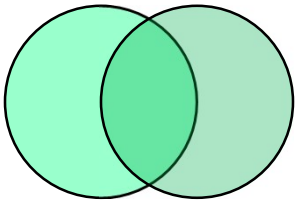
Oregon State University
Computer Graphics

mjb – November 20, 2024

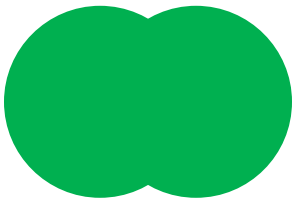
20

How Can You Guarantee That You Are Modeling With Legal Solids? Remember the "3D Venn Diagrams"

21



Two Overlapping Shapes



Union

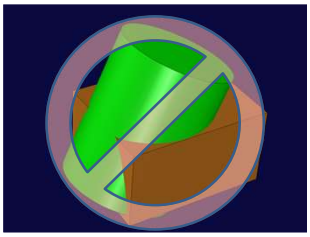
Intersection

Difference

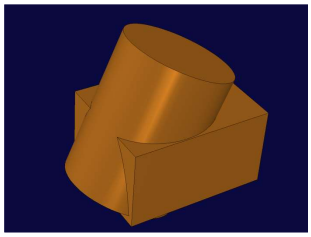
21

How Can You Guarantee That You Are Modeling With Legal Solids? Remember the "3D Venn Diagrams"

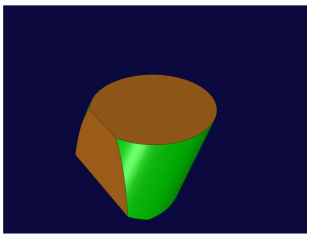
22



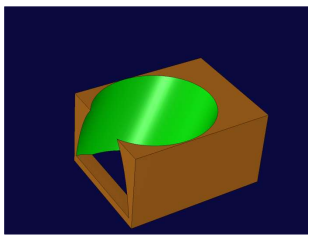
Two Overlapping Solids



Union



Intersection



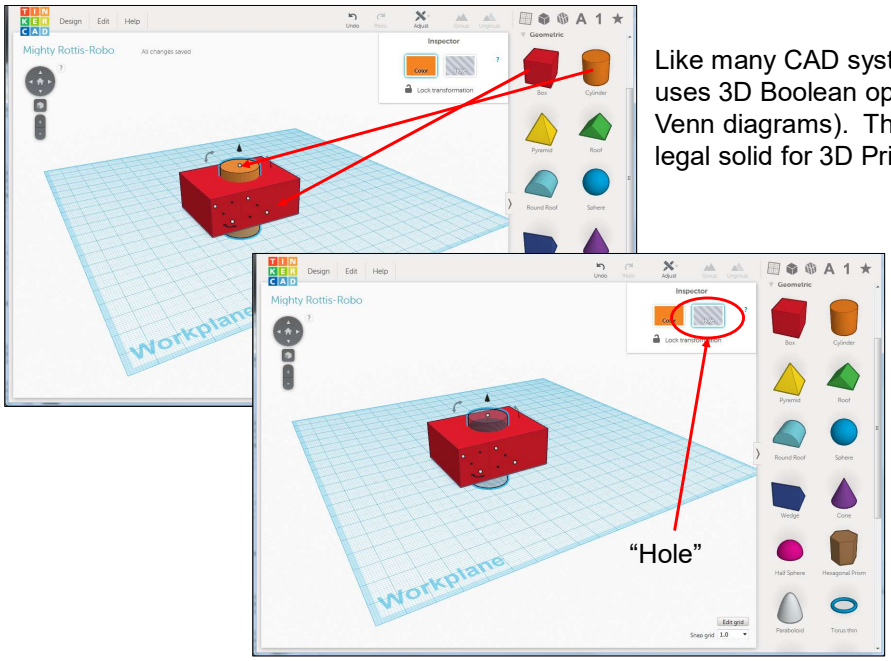
Difference

This is often called Constructive Solid Geometry (CSG)

22

TinkerCad: http://www.tinkercad.com

23



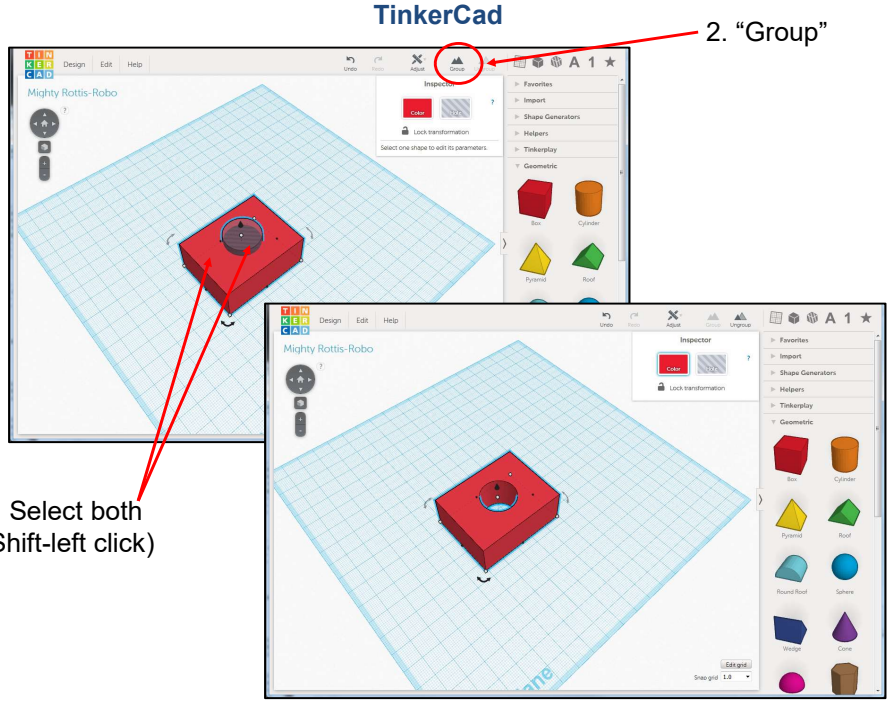
Like many CAD systems, TinkerCad uses 3D Boolean operators (3D Venn diagrams). This guarantees a legal solid for 3D Printing.

Oregon State University
Computer Graphics

mjb – November 20, 2024

TinkerCad

24

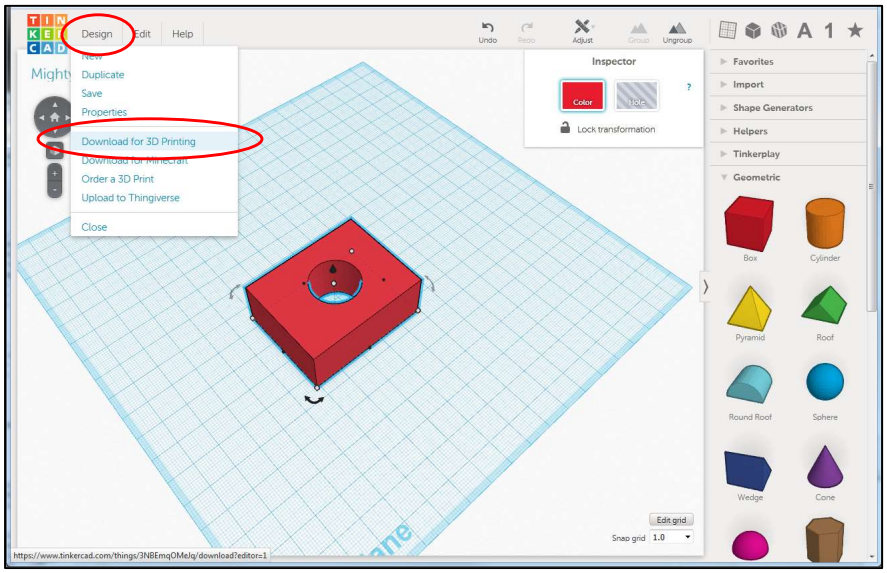


Oregon State University
Computer Graphics

mjb – November 20, 2024

TinkerCad

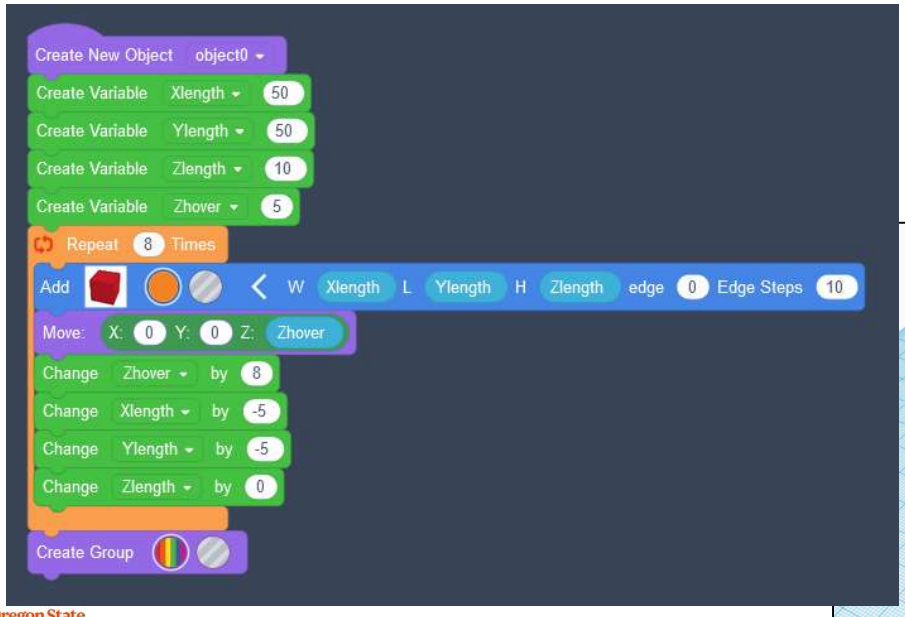
25

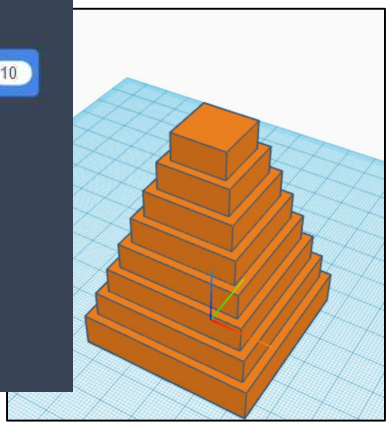


25

Procedural CSG Using TinkerCad/Codeblocks

26

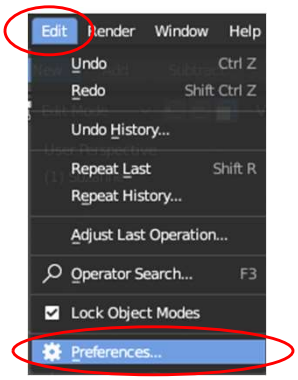




26

Blender's 3D Printing Utility isn't there by Default

27



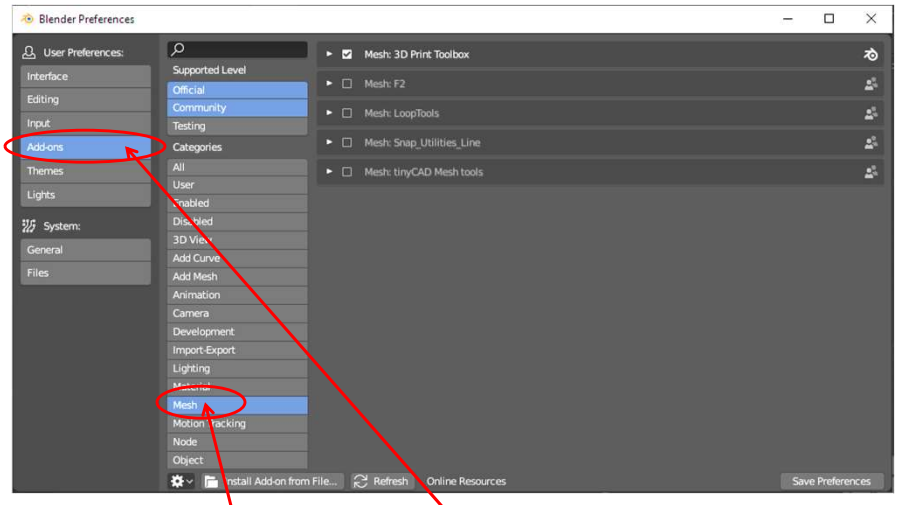
By default, Blender doesn't let you see its 3D Printing utility. You need to tell Blender to turn it on.

1. Click **Edit** → **Preferences**

Oregon State University
Computer Graphics

Blender's 3D Printing Options aren't there by Default

28

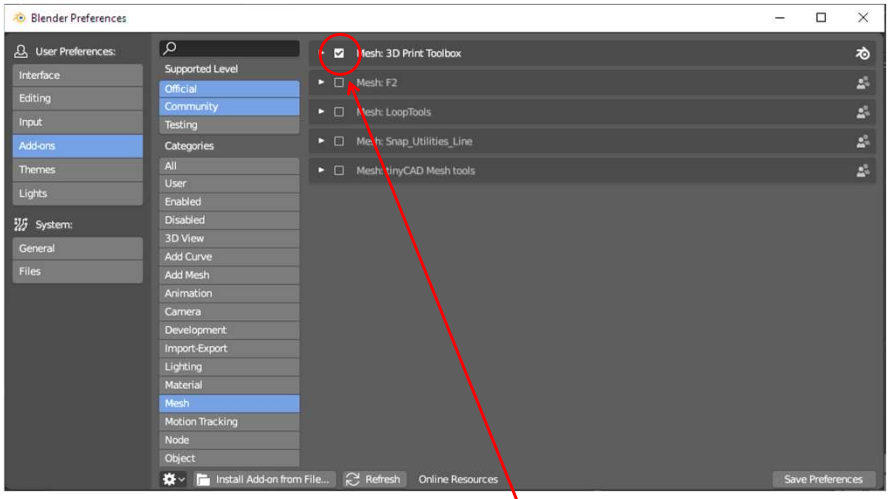


2. Click on the **Addons** tab
3. Click on **Mesh**

Oregon State University
Computer Graphics

Blender's 3D Printing Options aren't there by Default

29



4. Click the **Mesh: 3D Print Toolbox**

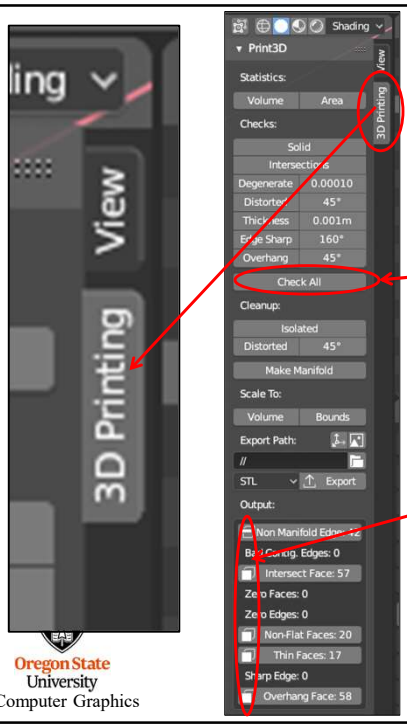
Oregon State University
Computer Graphics

mjb – November 20, 2024

29

Blender Options for 3D Printing

30



3D Printing now shows up in your Properties Region (hit the 'n' key if you're not seeing it)

Objects destined for 3D Printing must be "legal solids". Clicking on object and then on **Check All** will try to determine that

Tab over to Edit Mode. Clicking on any of these will highlight where they are on your object.

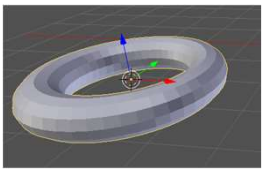
Oregon State University
Computer Graphics

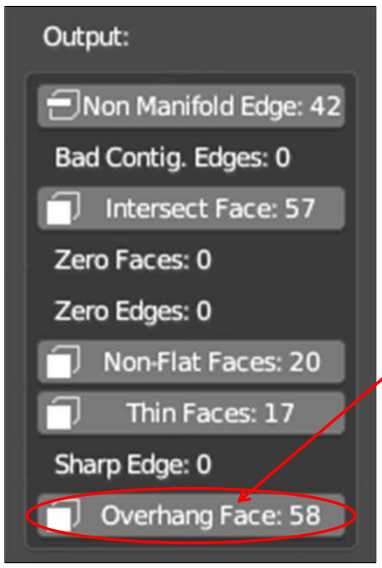
mjb – November 20, 2024

30

Blender Options for 3D Printing

31





An overhang face is not necessarily a bad thing. The entire bottom of the part will consist of, by necessity, overhang faces.

However, overhang faces that are not the bottom of the part could be a problem.

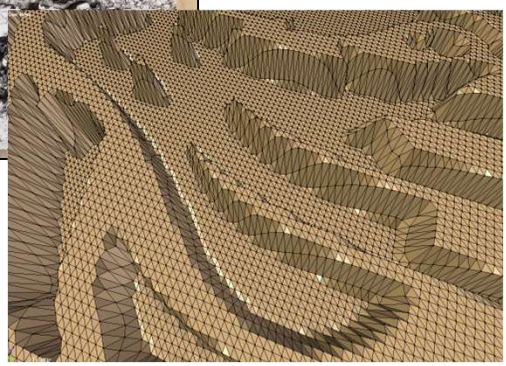
Oregon State University
Computer Graphics

mjb – November 20, 2024

31

Heightmap Files are Straightforward to use with 3D Printing

32



Oregon State University
Computer Graphics

mjb – November 20, 2024

32

A Very Special Heightmap 3D Printing Model (mmm…)

33

Oregon State University
Computer Graphics

mjb – November 20, 2024

33

A Very Special Heightmap 3D Printing Model (mmm…)

34

Oregon State University
Computer Graphics

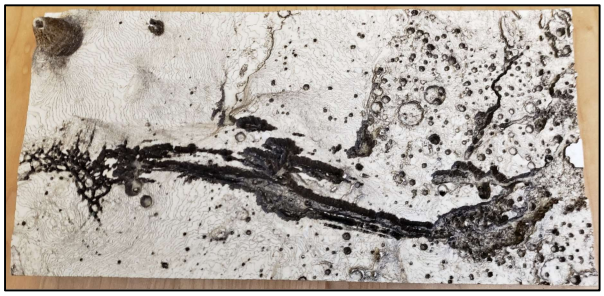
mjb – November 20, 2024

34

Most Any *Data* Can be Turned into a 3D-Printable Model

35

Mars



Mars

3D Ultrasound

Molecular Modeling

Ore University Computer Graphics

3D Ultrasound

35

Most Any *Data* Can be Turned into a 3D-Printable Model

36

Oregon

Grand Canyon

Dinosaur Egg

Earth

University Computer Graphics

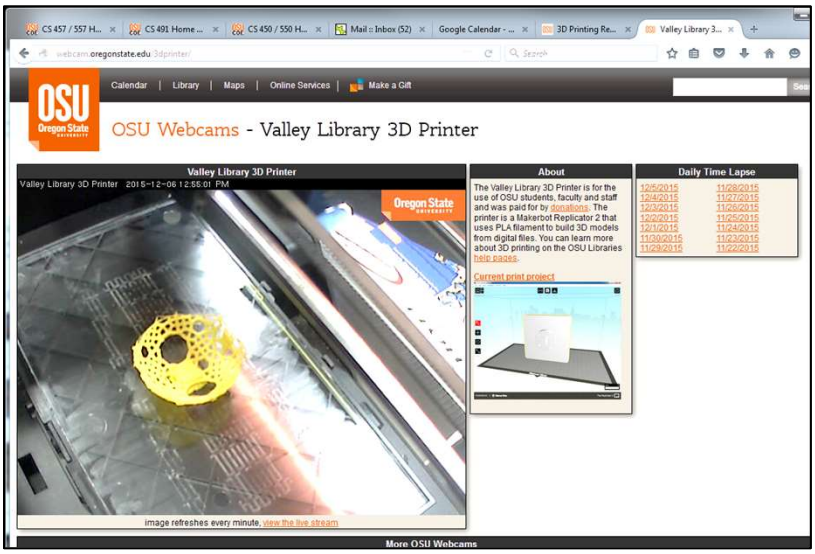
MRI Scan

36

The OSU Library Has Four 3D Printers for Student Use

To watch the OSU Library's 3D Printers, go to:

http://webcam.oregonstate.edu/3dprinter/



The OSU Library Has Four 3D Printers for Student Use

To send an STL model to the OSU Library's 3D Printers, go to:

http://guides.library.oregonstate.edu/3Dprinting/3Dprintform