Geometric Modeling for Computer Graphics

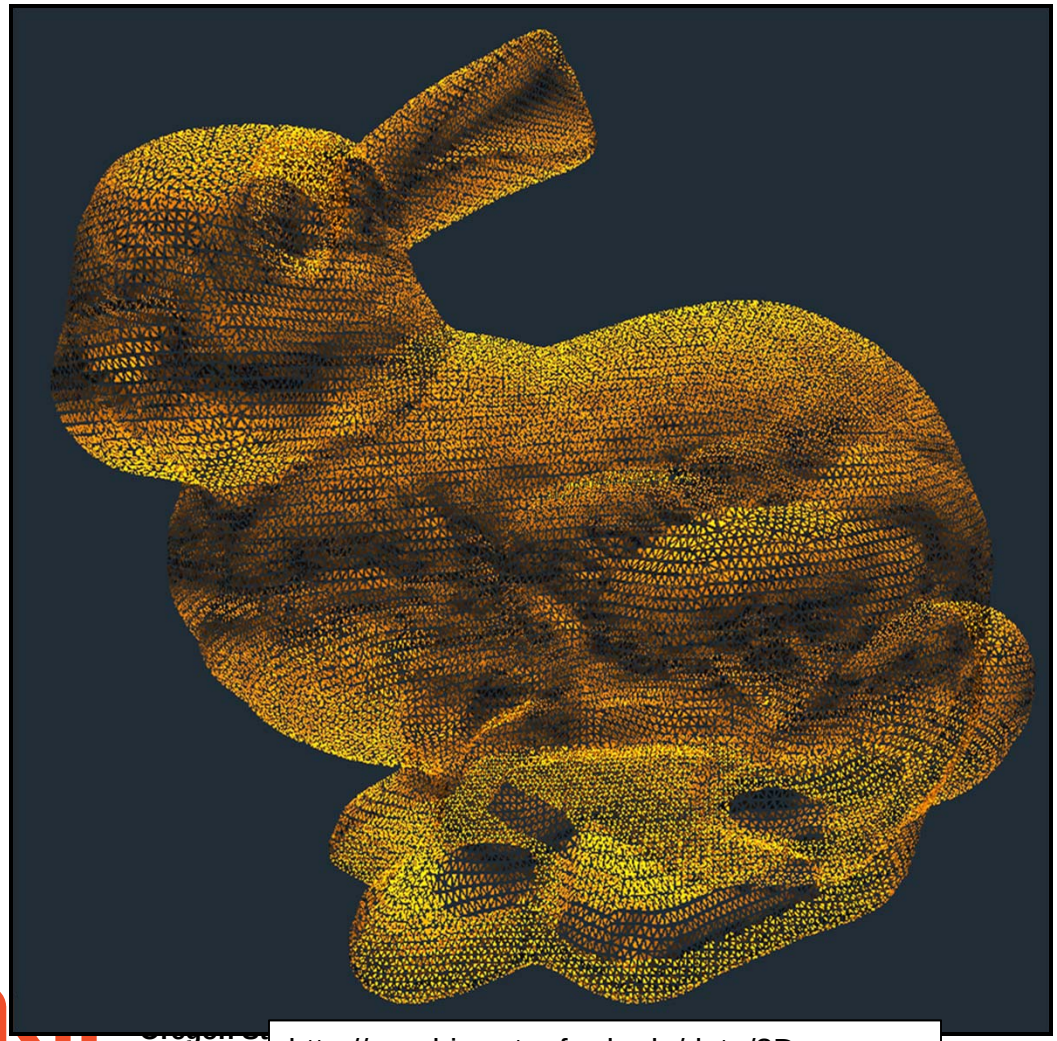
Mike Bailey

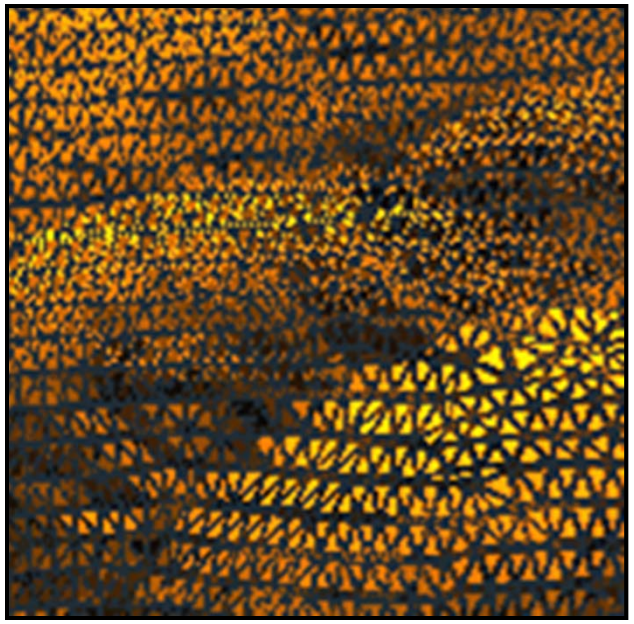
mjb@cs.oregonstate.edu

Oregon State University

Explicitly Listing Geometry and Topology

Models can consist of thousands of vertices and faces – we need some way to list them efficiently

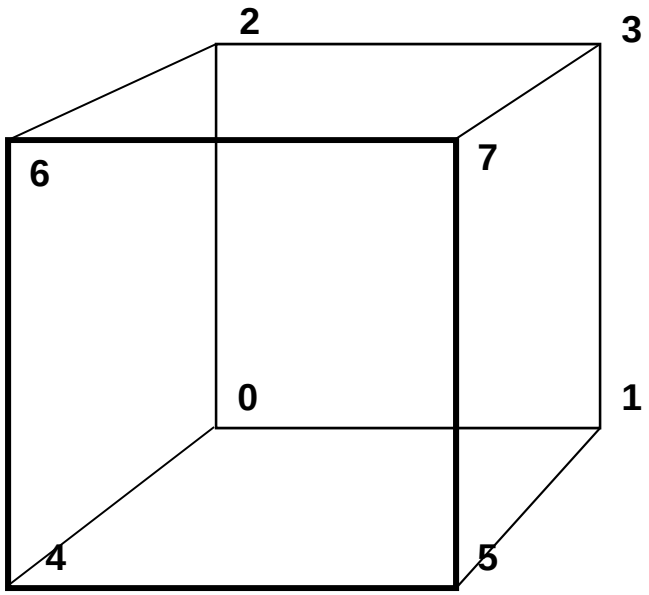




This is called a **Mesh.**

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

Explicitly Listing Geometry and Topology

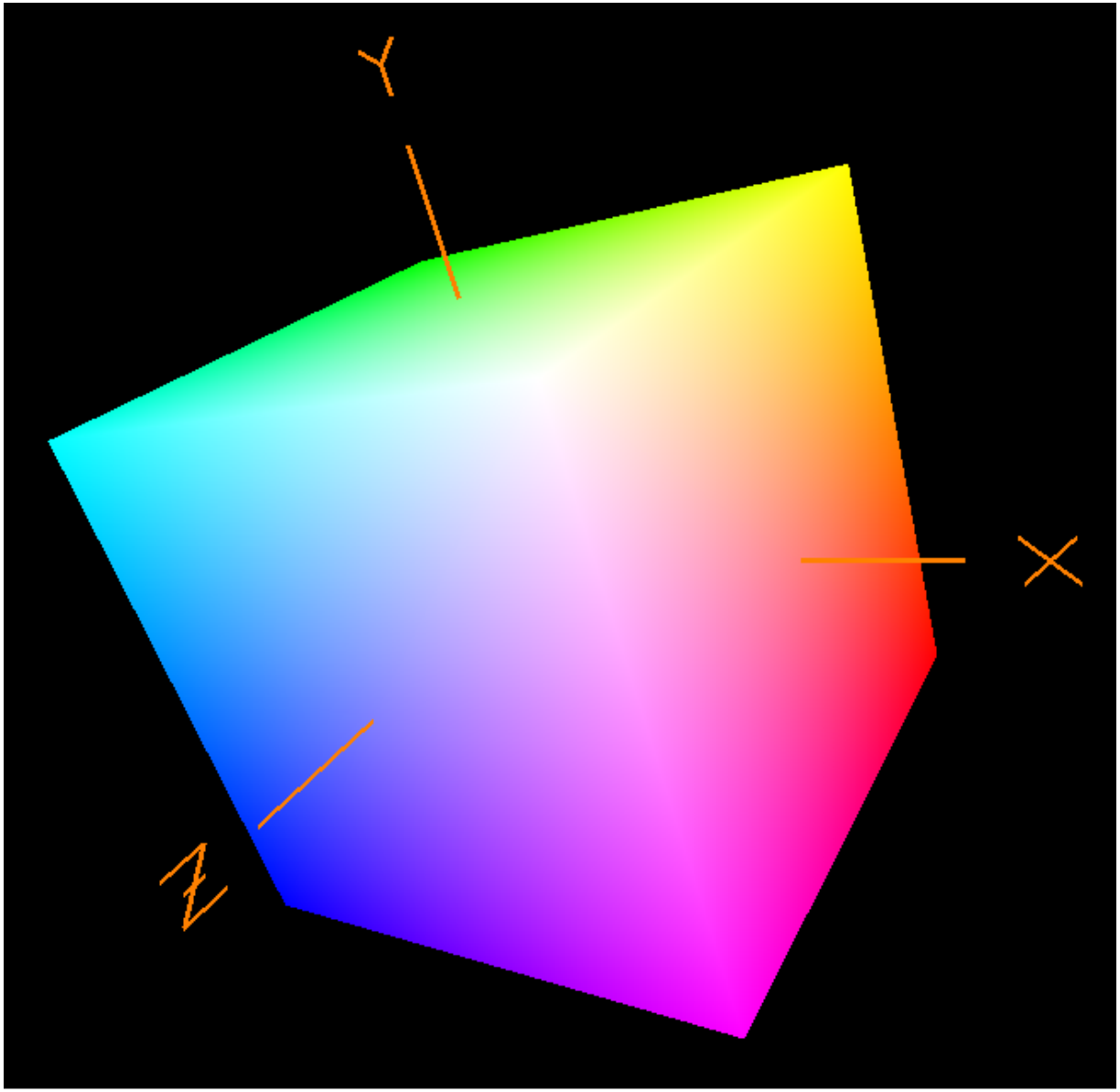



```
static GLfloat CubeVertices[ ][3] =
{
        { -1., -1., -1. },
        {  1., -1., -1. },
        { -1.,  1., -1. },
        {  1.,  1., -1. },
        { -1., -1.,  1. },
        {  1., -1.,  1. },
        { -1.,  1.,  1. },
        {  1.,  1.,  1. }
};
```

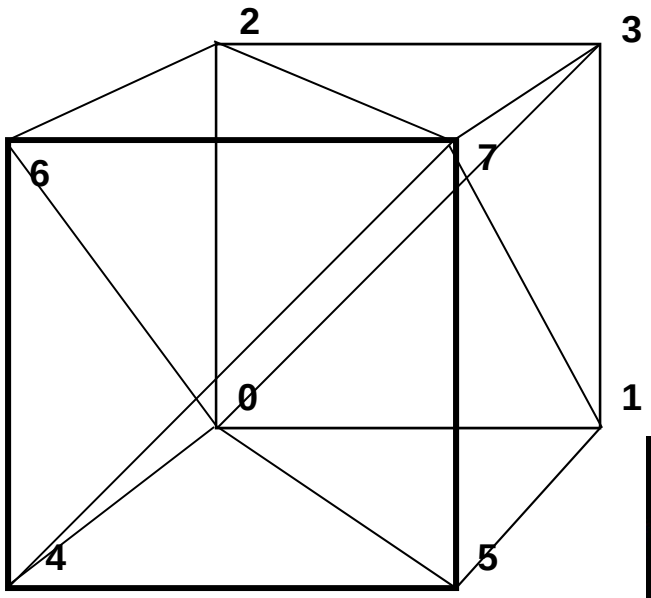
```
static GLfloat CubeColors[ ][3] =
{
        { 0., 0., 0. },
        { 1., 0., 0. },
        { 0., 1., 0. },
        { 1., 1., 0. },
        { 0., 0., 1. },
        { 1., 0., 1. },
        { 0., 1., 1. },
        { 1., 1., 1. },
};
```

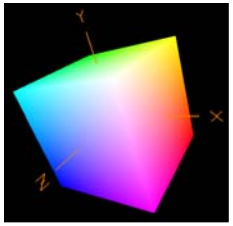
```
static GLuint CubeIndices[ ][4] =
{
        { 0, 2, 3, 1 },
        { 4, 5, 7, 6 },
        { 1, 3, 7, 5 },
        { 0, 4, 6, 2 },
        { 2, 6, 7, 3 },
        { 0, 1, 5, 4 }
};
```

Cube Example



The Cube Can Also Be Defined with Triangles

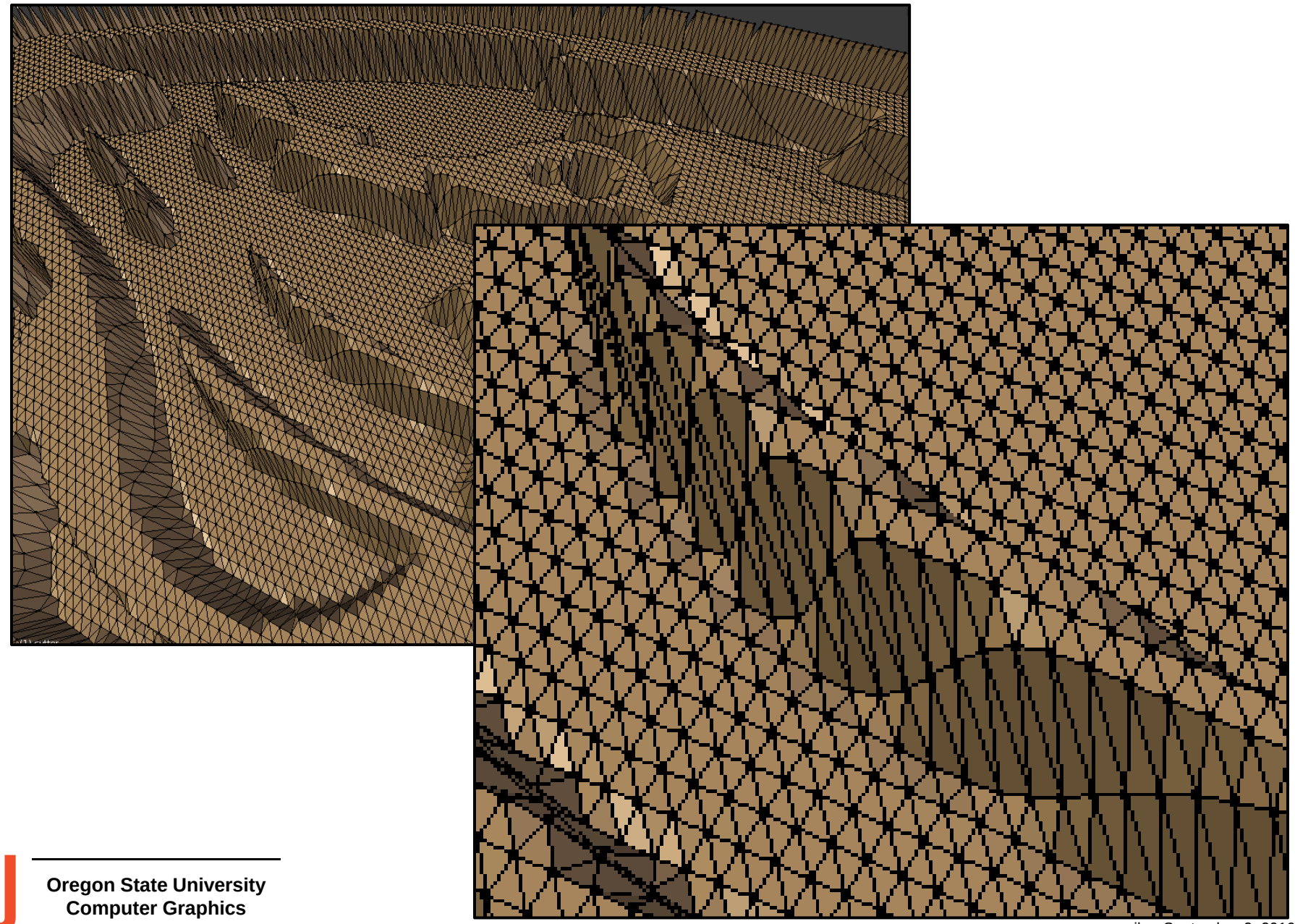




```
GLuint CubeIndices[ ][4] =
{
        { 0, 2, 3, 1 },
        { 4, 5, 7, 6 },
        { 1, 3, 7, 5 },
        { 0, 4, 6, 2 },
        { 2, 6, 7, 3 },
        { 0, 1, 5, 4 }
};
```

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

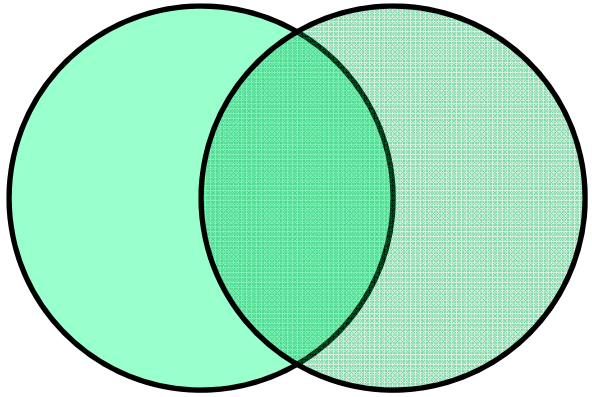
3D Printing uses a Triangular Mesh Data Format



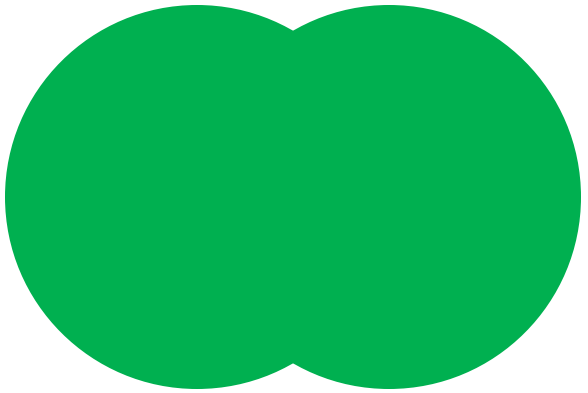
3D Printing uses a Triangular Mesh Data Format

Christmas Eve at a Graphics Nerd's House

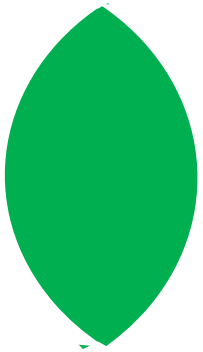
Another way to Model:
Remember Venn Diagrams (2D Boolean Operators) from High School?



Two Overlapping Shapes



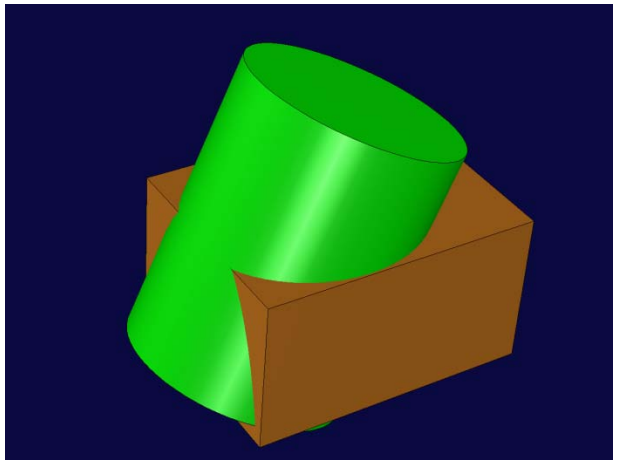
Union



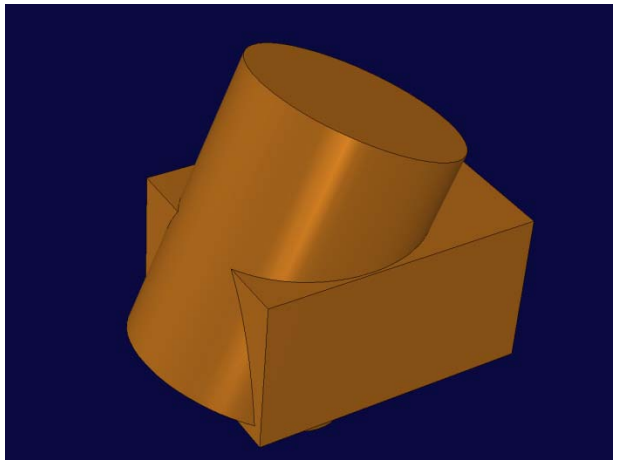
Intersection

Difference

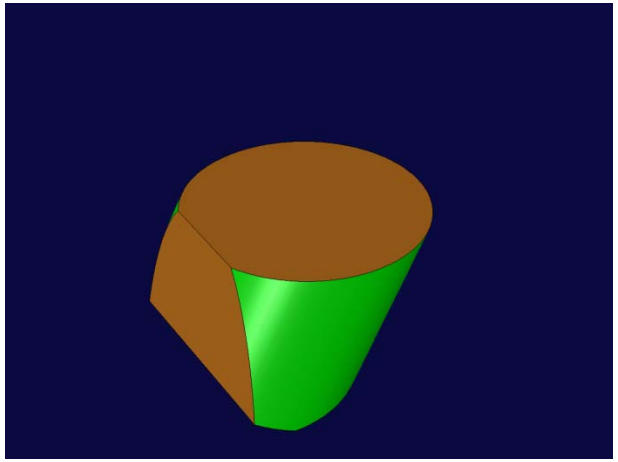
Solid Modeling Using 3D Boolean Operators



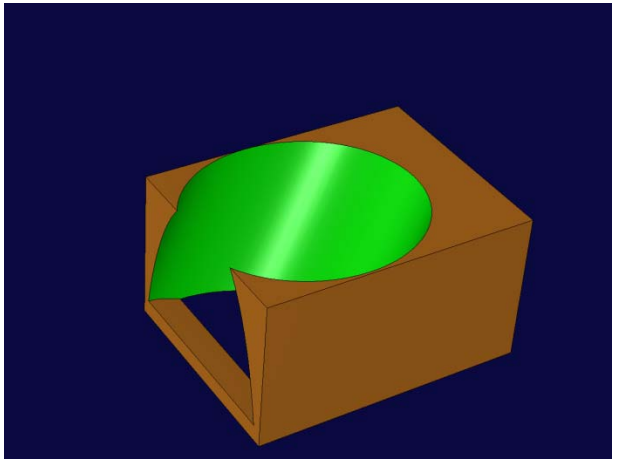
Two Overlapping Solids



Union



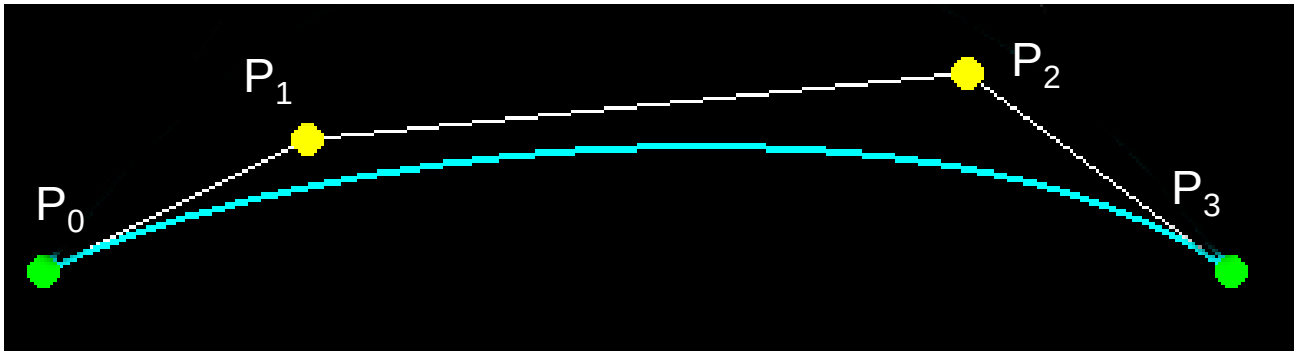
Intersection



Difference

This is often called Constructive Solid Geometry (CSG)

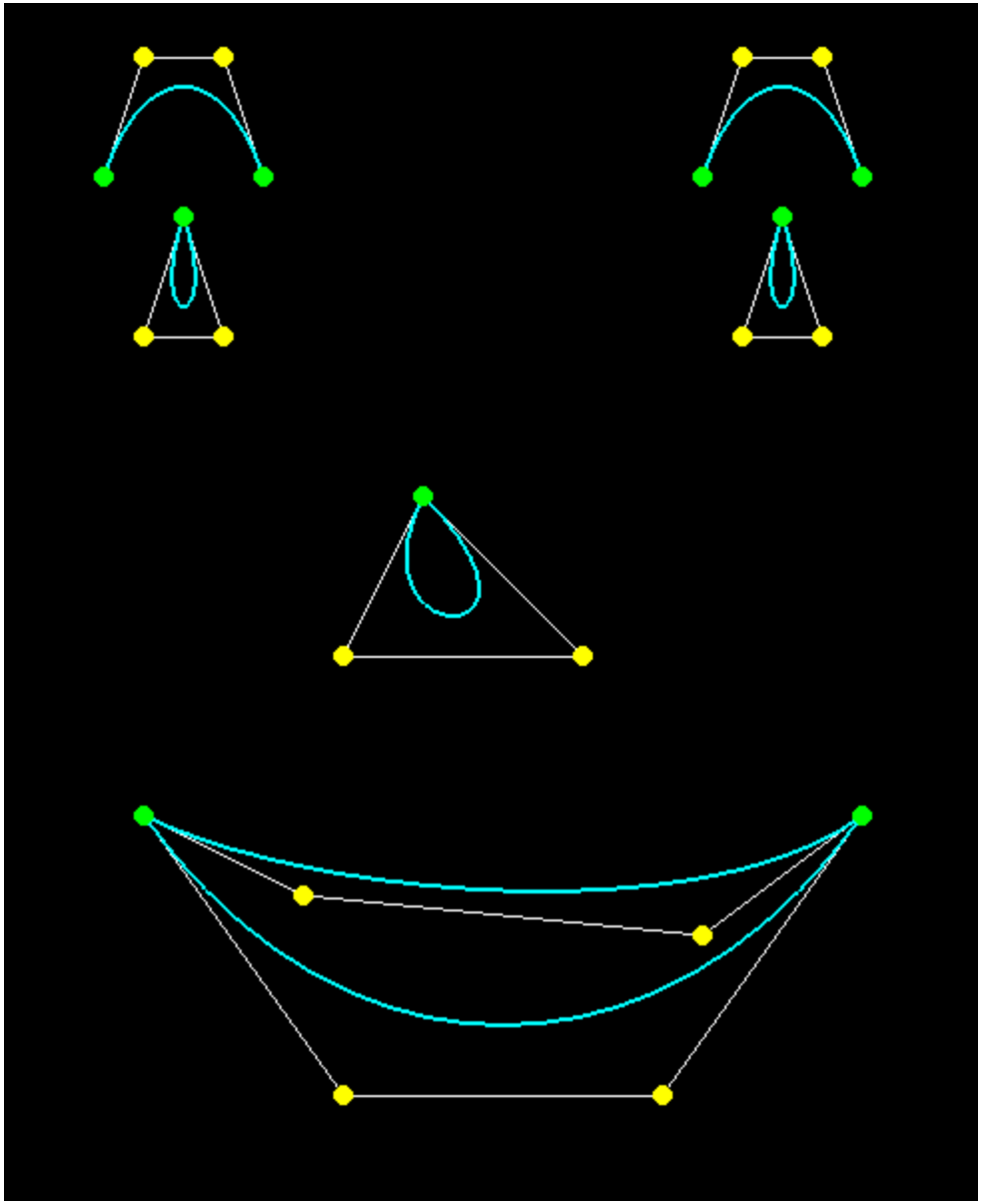
Another way to Model:
Curve Sculpting – Bezier Curve Sculpting



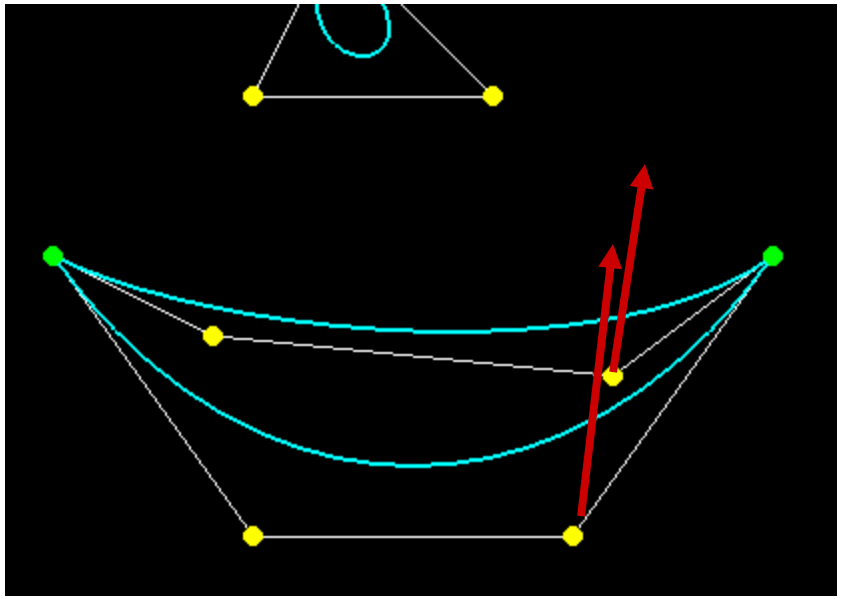
$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

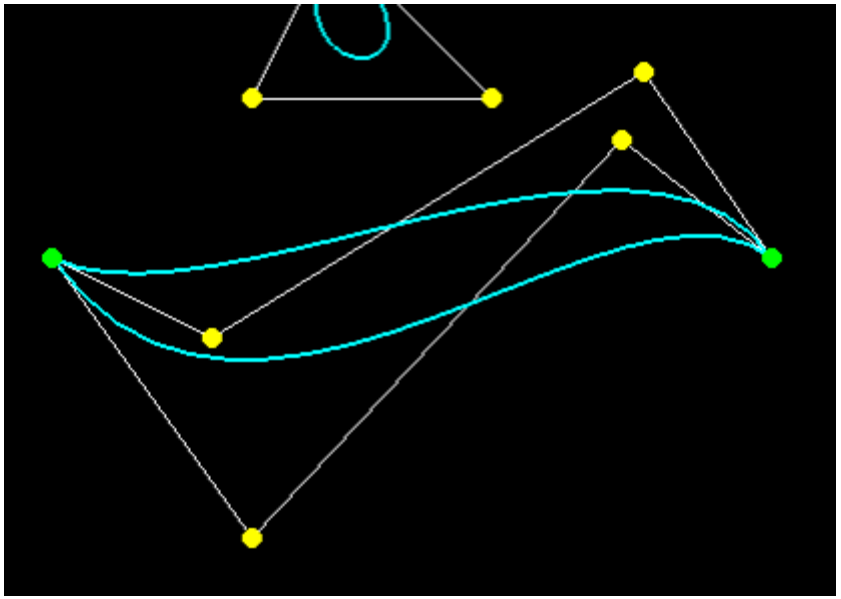
$$0. \le t \le 1.$$

Curve Sculpting – Bezier Curve Sculpting Example

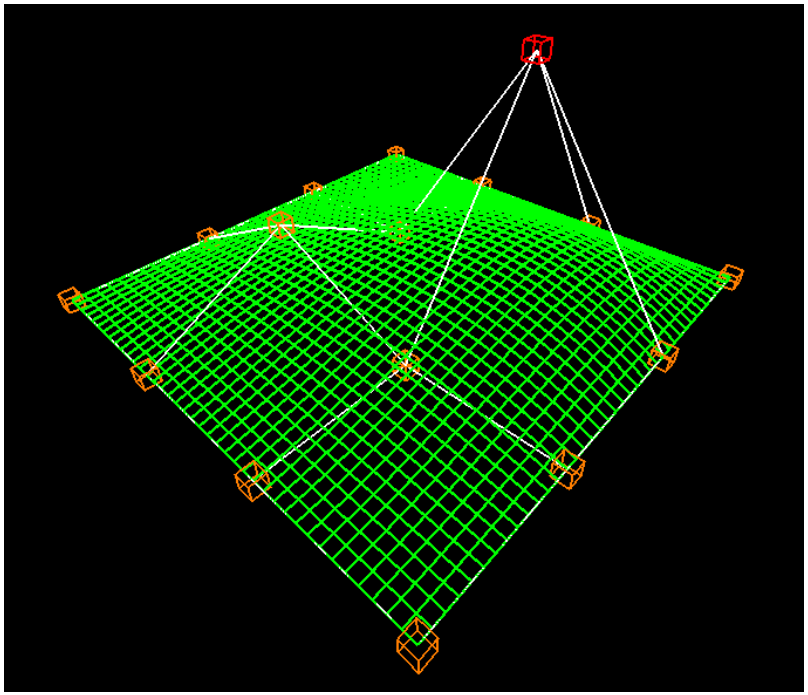


Curve Sculpting – Bezier Curve Sculpting Example

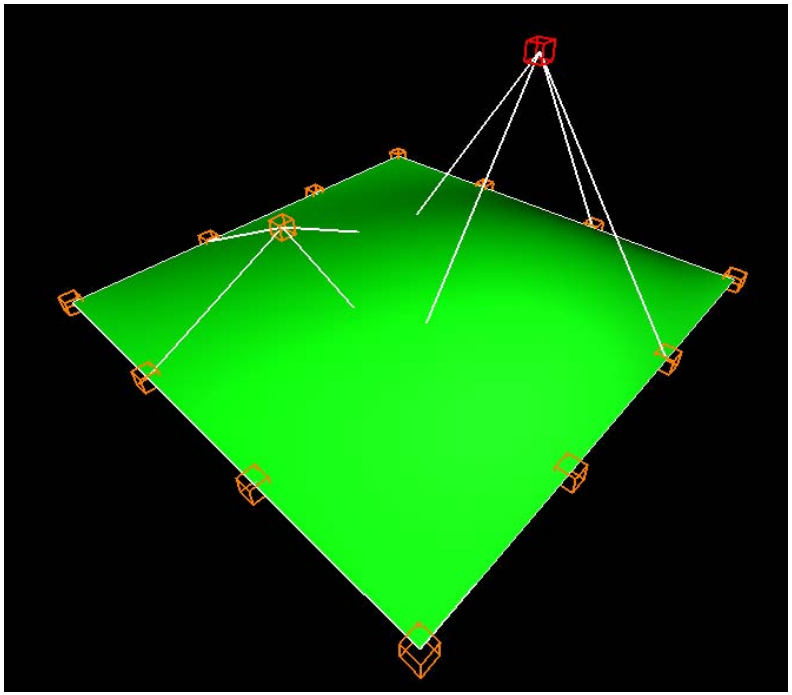




Another way to Model: Surface Sculpting

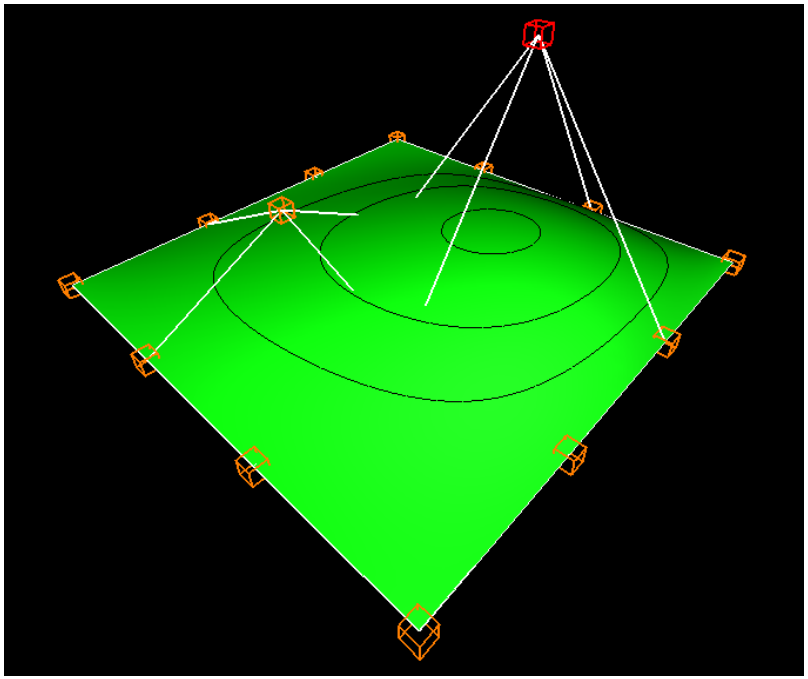


Wireframe

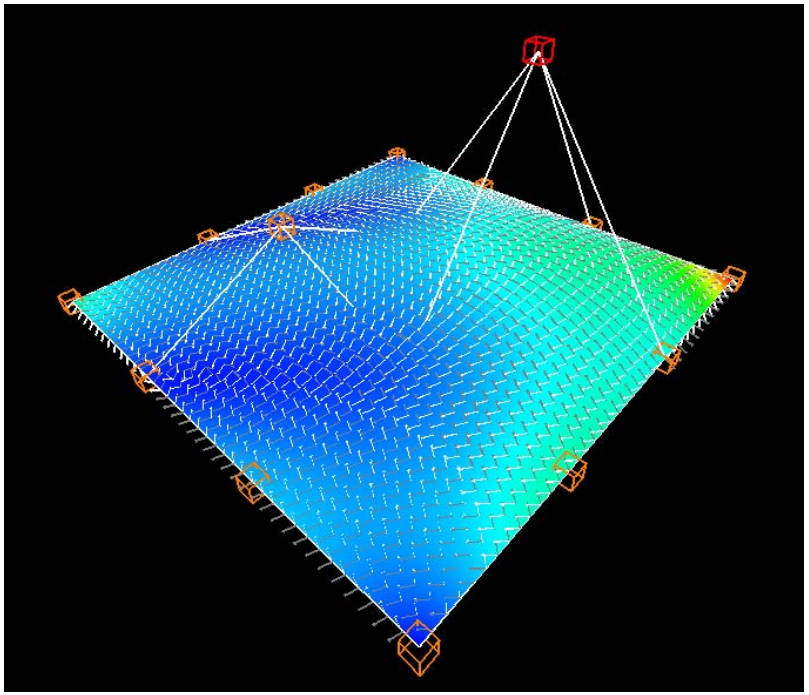


Surface

Surface Equations can also be used for Analysis

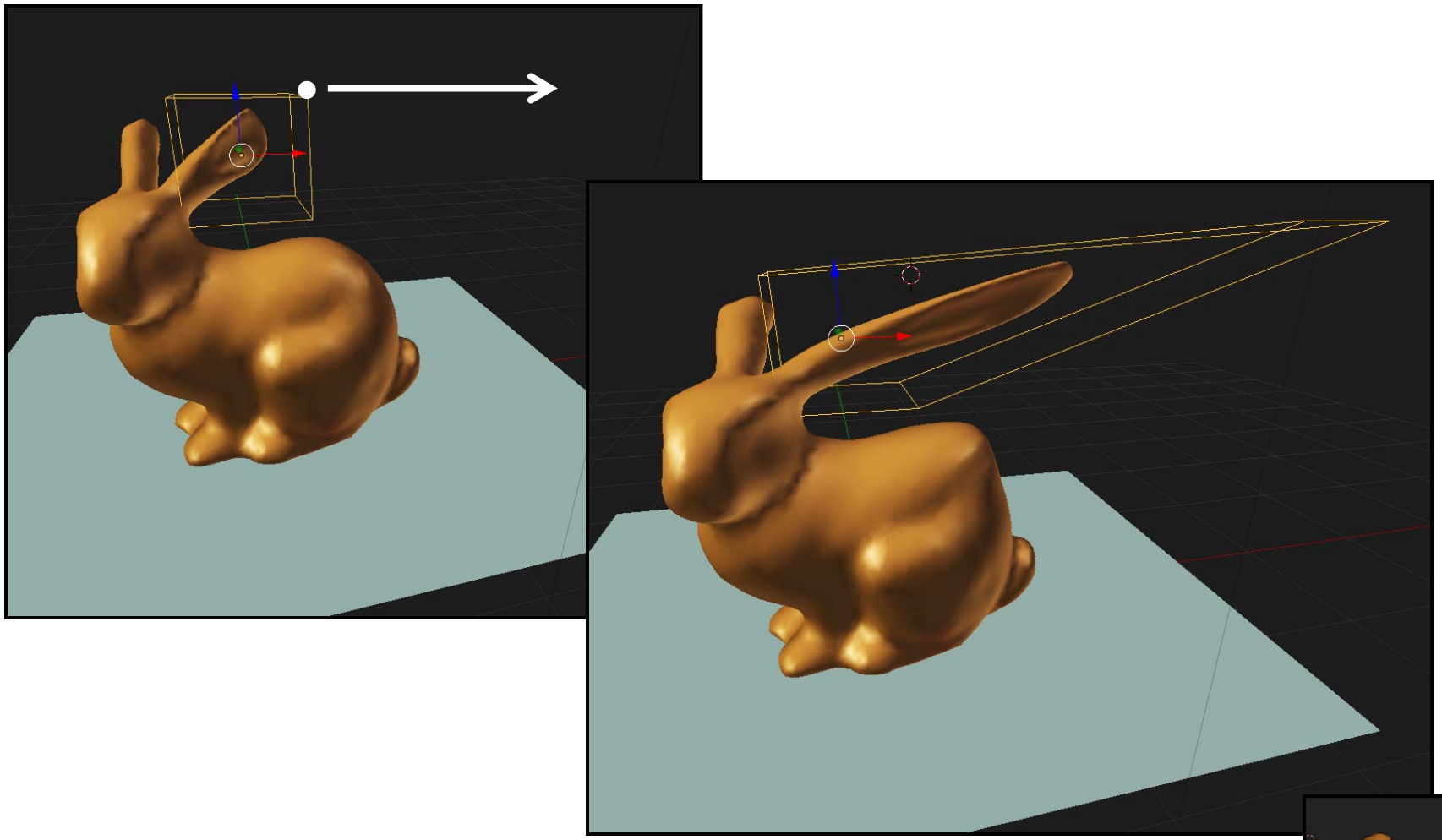


With Contour Lines



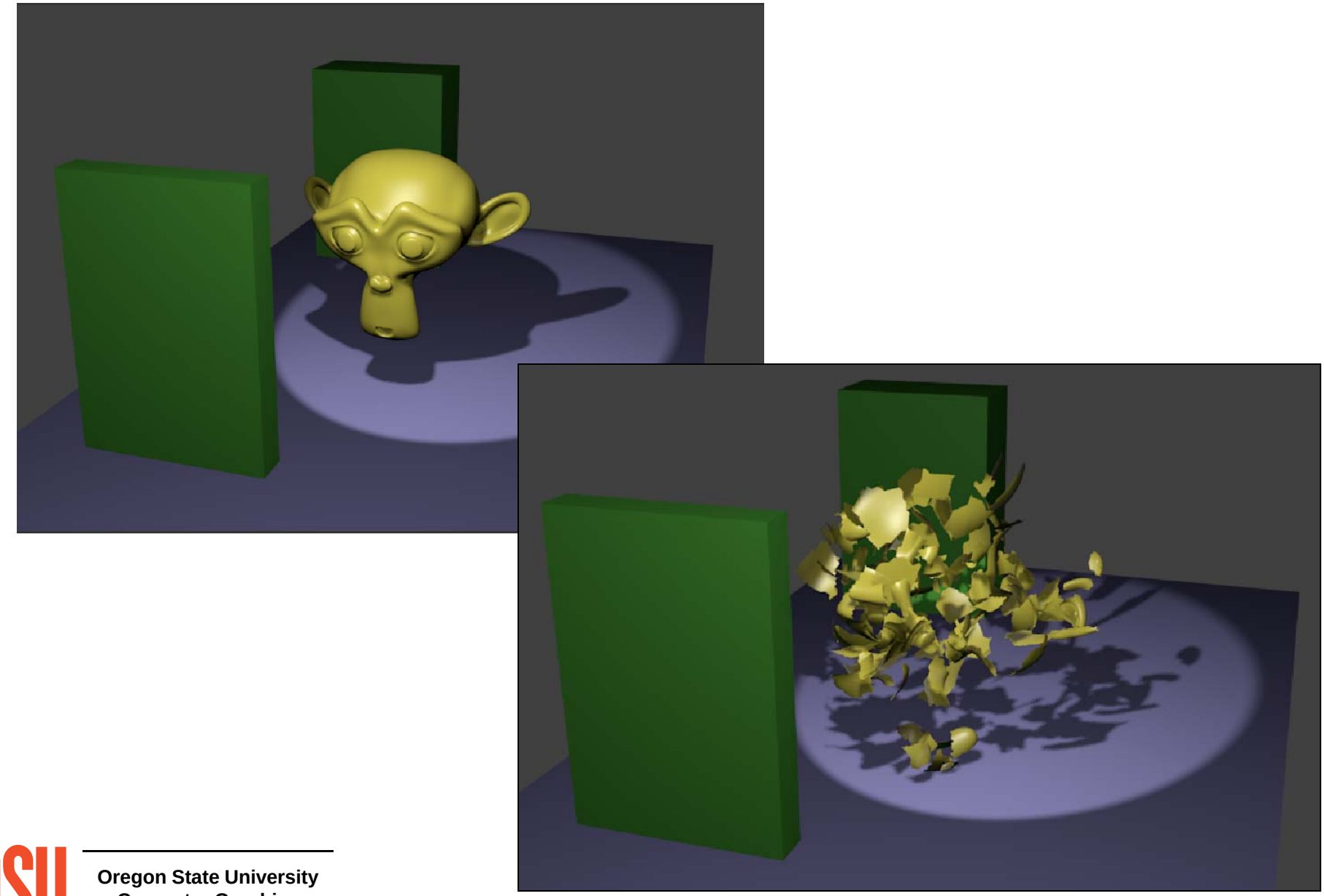
Showing Curvature

Another way to Model: Volume Sculpting



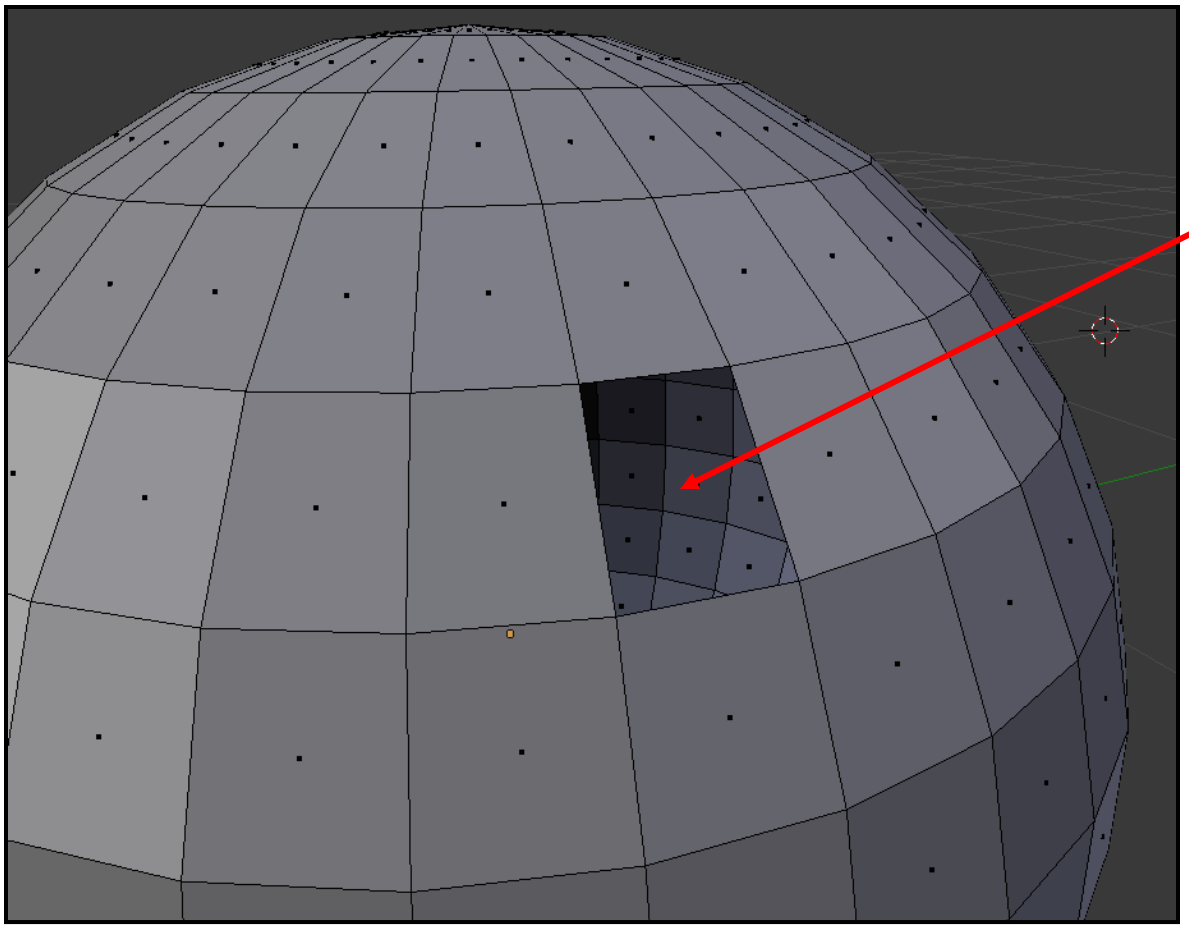
This is often called a **“Lattice”.**

Modeling → Simulation



Object Modeling Rules for 3D Printing

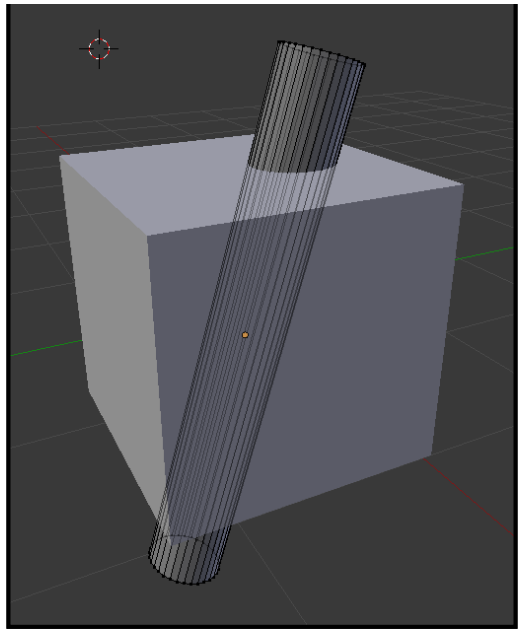
The object must be a legal solid. It must have a definite inside and a definite outside. It can't have any missing face pieces.

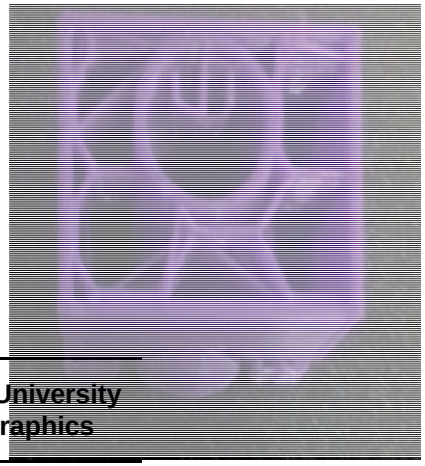


Object Modeling Rules for 3D Printing

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

Overlapped in 3D -- bad





Boolean union -- good