What do we mean by “Modeling”?

How we model geometry depends on what we would like to use the geometry for:

- Looking at its appearance
- Will we need to interact with its shape?
- How does it interact with its environment?
- How does it interact with other objects?
- What is its surface area and volume?
- Will it need to be 3D-printed?
- Etc.

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.
If it’s in nice neat rows like this, it is called a Regular Mesh.
If it’s not, it is called an Irregular Mesh, or oftentimes called a Triangular Irregular Network, or TIN.
3D Printing uses an Irregular Triangular Mesh Data Format

Meshes Can Be Smoothed

Mesh Vertices Can Be Edited

Remember Venn Diagrams (2D Boolean Operators) from High School?

Well, Welcome to Venn Diagrams in 3D
Geometric Modeling Using 3D Boolean Operators on Meshes

Two Overlapping Solids
Union: A ∪ B
Intersection: A ∩ B
Difference: A - B

Another Way to Edit Meshes: Volume Sculpting

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

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

Procedural Geometric Modeling Using TinkerCad/Codeblocks

Another way to Model:
Curve Sculpting – Bézier Curve Sculpting

P(t) = (1 − t)^3 P_0 + 3t(1 − t)^2 P_1 + 3t^2(1 − t) P_2 + t^3 P_3
0 ≤ t ≤ 1.

where P represents \[
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\]

You draw the curve as a series of lines
GL_LINE_STRIP is a good topology for this
Curve Sculpting – Bézier Curve Sculpting Example

Moving a single control point moves its entire curve

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

The Yellow 4-Point Bézier Curve

The Catmull-Rom curve consists of any number of points.
The first point influences how the curve starts.
The last point influences how the curve ends.
The overall curve goes smoothly through all other points.

To draw the curve, grab points 0, 1, 2, and 3, call them \( P_0, P_1, P_2, \) and \( P_3 \) and loop through the following equation, varying \( t \) from 0 to 1, in an increment of your own choosing:

\[
P(t) = 0.5 \times \left[ (2 \times P_1 + t \times (P_2 + P_3)) + t^2 \times (2 \times P_2 - 5 \times P_1 + 4 \times P_3 - P_0) \right]\]

where \( P_0 \) represents \( t \).

For each set of 4 points, this equation just draws the line between the second and third points. That's why you keep having to use subsequent sets of 4 points.

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

Another way to Model:
Curve Sculpting – Catmull-Rom Curve Sculpting

For each set of 4 points, this equation just draws the line between the second and third points. That's why you keep having to use subsequent sets of 4 points.

The Yellow 6-Point Catmull-Rom Curve

For each set of 4 points, this equation just draws the line between the second and third points. That's why you keep having to use subsequent sets of 4 points.

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

Another way to Model:
Curve Sculpting – Catmull-Rom Curve Sculpting

To draw the curve, grab points 0, 1, 2, and 3, call them \( P_0, P_1, P_2, \) and \( P_3 \) and loop through the following equation, varying \( t \) from 0 to 1, in an increment of your own choosing:

\[
P(t) = P(t) = 0.5 \times \left[ (2 \times P_1 + t \times (P_2 + P_3)) + t^2 \times (2 \times P_2 - 5 \times P_1 + 4 \times P_3 - P_0) \right]\]

where \( P_0 \) represents \( t \).

For each set of 4 points, this equation just draws the line between the second and third points. That's why you keep having to use subsequent sets of 4 points.

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

Another way to Model:
Bézier Surface Sculpting

Moving a single point moves its entire surface

A Small Amount of Input Change Results in a Large Amount of Output Change
Surface Equations can also be used for Analysis

- Showing Contour Lines
- Showing Curvature

Another Way to Model: Metaball Objects

The cool thing is that, if you move them close enough together, they will "glom" into a single object.

Metaball Objects Can Be Turned into Meshes for Later Editing

Voxelization as a Special Way to Model 3D Geometry

Displacement Textures as a Special Way to Model 3D Geometry
Displacement Textures as a Special Way to Model 3D Geometry

### Image Texture

![Image Texture](image1.png)

Displacement Texture

(light = high, dark = low)

Displacement Textures as a Special Way to Model 3D Geometry

### moondisp.vert

```glsl
uniform float uScale;
uniform sampler2D uDispUnit;
out vec2   vST;
out vec3 vNormal;
void main( )
{
    vec2 st = gl_MultiTexCoord0.st;
    vST = st; // to send to fragment shader
    vec3 norm = normalize( gl_Normal);
    vNormal= normalize( gl_NormalMatrix * gl_Normal);
    float disp = texture( uDispUnit, st ).r;
    // in half-meters, relative to a radius of 1,727,400 meters
    disp *= uScale;
    vec3 vert = gl_Vertex.xyz;
    vert += norm * disp;
    gl_Position = gl_ModelViewProjectionMatrix * vec4( vert, 1. );
}
```

### moondisp.frag

```glsl
#version 330 compatibility
uniform float uLightX, uLightY, uLightZ;
uniform float uKd;
uniform sampler2D uColorUnit;
in vec2 vST;
in vec3 vNormal;
void main( )
{
    vec3 light = normalize( vec3( uLightX, uLightY, uLightZ ) );
    float intensity = uKd * abs( dot( vNormal, light ) );
    intensity += (1.-uKd); // ambient
    vec3 newcolor = texture( uColorUnit, vST ).rgb;
    gl_FragColor = vec4( newcolor*intensity, 1. );
}
```

L-Systems as a Special Way to Model 3D 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:

```
F → FF+[-F-F-F] [+F+F+F]
```

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

```
F → FF+[-F-F-F] [+F+F+F]
```

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

```
F → FF+[-F+F-][+F-F+F]
```

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

![L-Systems Diagram](image2.png)
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.

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

Object Modeling Rules for 3D Printing

The Simplified Euler’s Formula* for Legal Solids

*sometimes called the Euler-Poincaré formula

\[ F - E + V = 2 \]

For a cube, \( 6 - 12 + 8 = 2 \)

The full formula is:

\[ F - E + V - L = 2(B - G) \]

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.