Animation
Animation

Rendering is the process of giving motion to your geometric modes. 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?
- 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?
Keyframe Animation

These icons refer to explanatory videos on the class web site

anim2.mp4
Keyframe Animation

Blender:
Forward Kinematics:
Change Parameters – Connected Things Move
(All Tinker Toy users understand this)
Forward Kinematics: Transformation Hierarchies

Determine Object Locations?
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?”

\[
\begin{align*}
\theta_1？ \\
\theta_2？ \\
\theta_3？ \\
(X^*, Y^*)
\end{align*}
\]
Particle Systems: A Cross Between Modeling and Animation?
Particle Systems: A Cross Between Modeling and Animation?

The basic process is:

- Emit
- Random Number Generator
- Display
- Update
Particle Systems Examples

Chuck Evans
Particle Systems Examples
Particle Systems Examples
Particle Systems Examples

The Lion King (2019) -- Disney
A Particle System to Simulate Colliding Galaxies in *Cosmic Voyage*
Particles Don’t Actually Have to Be “Particles”
Newton’s second law:

\[ \text{force} = \text{mass} \times \text{acceleration} \]

or

\[ \text{acceleration} = \frac{\text{force}}{\text{mass}} \]

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

fluid.avi
Animating using Physics

\[ k = \text{spring stiffness} \in \text{Newtons/meter or pounds/inch} \]

\[ (D - D_0) = \frac{F}{k} \]

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

\[ F = k(D - D_0) \]

This is known as Hooke’s Law.
Animating using the Physics of a Mesh of Springs

"Lumped Masses"
Simulating a Bouncy String
Simulating a Bouncy String
Placing a Physical Barrier in the Scene
Animating Cloth
Cloth Example

cloth.mp4
Cloth Example
Functional Animation:
Make the Object *Want* to Move Towards a Goal Position

$$m\ddot{x} + c\dot{x} + kx = 0$$
Functional Animation:
While Making it Want to Move Away from all other Objects

\[ m\ddot{x} = \sum F_{\text{repulsive}} \]

- \( F_{\text{repulsive}} = \frac{C_{\text{repulse}}}{d^\text{Power}} \)
  - Repulsion Coefficient
  - Distance between the boundaries of the 2 bodies
  - Repulsion Exponent
Total Goal – Make the Free Body Move Towards its Final Position While Being Repelled by the Other Bodies

\[ m\ddot{x} + c\dot{x} + kx = \sum F \]
Increasing the Stiffness

Stiffness = 3

Stiffness = 6

Stiffness = 9
Increasing the Repulsion Coefficient

Repulse = 10

Repulse = 30
Functional Animation
Motion Capture as an Input for Animation

Natural Point
Motion Capture is for Faces Too
Tron I –
They probably should have used physics, but didn’t
Pixar Animated Shorts
class Keytimes:

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

Sidebar: DIY KeyTime Animation

A C++ class can do it all for you
DIY KeyTime Animation

```c
Keytimes Xpos;

int main( int argc, char *argv[ ] )
{
    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 %8.3f\n", Xpos.GetFirstTime( ), Xpos.GetLastTime( ) );

    for( float t = 0.; t <= 2.01; t += 0.1 )
    {
        float v = Xpos.GetValue( t );
        fprintf( stderr, "%8.3f	%8.3f\n", t, v );
    }
}
```
DIY KeyTime Animation

( 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

<table>
<thead>
<tr>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.000</td>
</tr>
<tr>
<td>0.100</td>
<td>0.232</td>
</tr>
<tr>
<td>0.200</td>
<td>0.806</td>
</tr>
<tr>
<td>0.300</td>
<td>1.535</td>
</tr>
<tr>
<td>0.400</td>
<td>2.234</td>
</tr>
<tr>
<td>0.500</td>
<td>2.718</td>
</tr>
<tr>
<td>0.600</td>
<td>2.989</td>
</tr>
<tr>
<td>0.700</td>
<td>3.170</td>
</tr>
<tr>
<td>0.800</td>
<td>3.258</td>
</tr>
<tr>
<td>0.900</td>
<td>3.250</td>
</tr>
<tr>
<td>1.000</td>
<td>3.142</td>
</tr>
<tr>
<td>1.100</td>
<td>2.935</td>
</tr>
<tr>
<td>1.200</td>
<td>2.646</td>
</tr>
<tr>
<td>1.300</td>
<td>2.302</td>
</tr>
<tr>
<td>1.400</td>
<td>1.924</td>
</tr>
<tr>
<td>1.500</td>
<td>1.539</td>
</tr>
<tr>
<td>1.600</td>
<td>1.169</td>
</tr>
<tr>
<td>1.700</td>
<td>0.840</td>
</tr>
<tr>
<td>1.800</td>
<td>0.574</td>
</tr>
<tr>
<td>1.900</td>
<td>0.397</td>
</tr>
<tr>
<td>2.000</td>
<td>0.333</td>
</tr>
</tbody>
</table>
Using the System Clock for Timing in Display( )

#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( );
}