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?
Keyframe Animation

These icons refer to explanatory videos on the class web site.

Blender:
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.

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

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.

```cpp
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.

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

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.

```cpp
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

```cpp
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 %8.3f\n", Xpos.GetFirstTime( ), Xpos.GetLastTime( ) );

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

Instead of Key Frames, I Like Specifying Key Times Better

```cpp
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 %8.3f\n", Xpos.GetFirstTime( ), Xpos.GetLastTime( ) );

    for( float t = 0.; t <= 2.0; t += 0.1 )
    {
        float v = Xpos.GetValue( t );
        fprintf( stderr, "%8.3f	%8.3f\n", t, v );
    }
}
```
Instead of Key Frames, I Like Specifying Key Times Better

Time runs from 0.000 to 2.000

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.100</td>
<td>0.232</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.200</td>
<td>0.806</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.300</td>
<td>1.535</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.400</td>
<td>2.234</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>2.718</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.600</td>
<td>2.989</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.700</td>
<td>3.170</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.800</td>
<td>3.250</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.900</td>
<td>3.250</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.000</td>
<td>3.142</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.100</td>
<td>2.935</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.200</td>
<td>2.646</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.300</td>
<td>1.924</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.400</td>
<td>1.539</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.500</td>
<td>1.169</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.600</td>
<td>0.840</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.700</td>
<td>0.574</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.800</td>
<td>0.397</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.900</td>
<td>0.333</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.000</td>
<td>0.333</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Instead of Key Frames, I Like Specifying Key Times Better

= the originally-specified key-time pairs

Using the System Clock in Display( ) for Timing

```c
#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
Forward Kinematics:
Change Parameters – Connected Things Move
(All children 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) 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?”

Forward Kinematics solves the problem “if I know the link transformation parameters, where are the links?”. 
The basic process is:

1. Emit
2. Display
3. Update
4. Random Number Generator
Particle Systems Examples

Chuck Evans

particles.mp4
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”
**Multiple Animation Techniques Can Be Combined**

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

![Image of particle system](image)

---

**Animating using Rigid-body Physics**

Newton's second law:

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

or

\[
\ddot{x} = \text{acceleration} = \text{force} / \text{mass}
\]

\[
x(T) = \int_{t=0}^{t=T} \dot{x} dt \approx \sum \sum \ddot{x} \Delta t
\]

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.
Computer Graphics

Animating using Fluid Physics

fluid.avi

Computer Graphics

Animating using Physics

D₀ = unloaded spring length

\[(D - D₀) = \frac{F}{k}\]

k = spring stiffness in Newtons/meter or pounds/inch

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

\[F = k(D - D₀)\]

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

Simulating a Bouncy String
Simulating a Bouncy String

Placing a Physical Barrier in the Scene

string.mp4
Cloth Example

Geri's Game, Pixar
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

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

Repulsion Coefficient

\[ F_{\text{repulsive}} = C_{\text{repulse}} \left( \frac{1}{d^p} \right) \]

Repulsion Exponent

Distance between the boundaries of the 2 bodies
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

avoid.mp4
Motion Capture ("MoCap") as an Input for Animation

Motion Capture is for Hands and Faces Too
Even Animals can be MoCapped

My cats would never have put up with this...

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

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

Pixar Animated Shorts

Pixar