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

.anim2.mp4

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

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 PrintTimeValues();
```

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

Instead of Key Frames, I like specifying Key Times better.
And, so, I created a C++ class to do it for you.

```cpp
Keytimes Xpos; // Declare one class per parameter you are animating. Here it wants to
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\t%8.3f\n", t, v );
    }
}
```

Instead of Key Frames, I Like Specifying Key Times Better

Keytimes Xpos; // Declare one class per parameter you are animating. Here it wants to
```cpp
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 ) // Just to demonstrate, this for-loop runs through a collection of
t    {
        float v = Xpos.GetValue( t );
        fprintf( stderr, "%8.3f\t%8.3f\n", t, v );
    }
}
```
Instead of Key Frames, I Like Specifying Key Times Better

4 time-value pairs

Time runs from 0.000 to 2.000

0.000  0.000
0.200  0.806
0.300  1.535
0.400  2.234
0.500  2.989
0.600  3.717
0.700  3.250
0.800  3.170
0.900  2.718
1.000  2.142
1.100  1.835
1.200  1.612
1.300  1.539
1.400  1.446
1.500  1.539
1.600  1.827
1.700  2.142
1.800  2.530
1.900  2.978
2.000  3.333

Instead of Key Frames, I Like Specifying Key Times Better

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 )
{
    int msec = glutGet( GLUT_ELAPSED_TIME ) % MSEC;

    // turn that into a time in seconds:
    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: Transformation Hierarchies

Determine Object Locations?

Ground

θ1

θ2

θ3
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?”

\[
\begin{align*}
\theta_1? \\
\theta_2? \\
\theta_3? \\
(X^*, Y^*)
\end{align*}
\]

Ground
Particle Systems: A Cross Between Modeling and Animation?

The basic process is:

1. Emit
2. Random Number Generator
3. Display
4. Update

Flowchart:
- Emit
- Random Number Generator
- Display
- Update
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:
force = mass * acceleration

or
\( \ddot{x} = \text{acceleration} = \frac{\text{force}}{\text{mass}} \)

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

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.
Animating using Physics

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

\( k = \text{spring stiffness} \) in Newtons/meter or pounds/inch

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

string.mp4
Placing a Physical Barrier in the Scene

Animating Cloth
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

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

- **Repulsion Coefficient**: \( C_{\text{repulse}} \)
- **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 = 9

Stiffness = 6

Stiffness = 3

Increasing the Repulsion Coefficient

Repulse = 30

Repulse = 10
Functional Animation

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

Natural Point
Motion Capture is for Faces Too

Even Animals can be MoCapped

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

My cat would never have put up with this…
Tron I –
They probably should have used physics, but didn’t