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

Instead of Key Frames, I Like Specifying Key Times Better.

Just to demonstrate, this for-loop runs through a collection of time values and looks up the interpolate x-location at those values. Normally you would get the time from the system clock.

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

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:
", Xpos.GetNumKeytimes( ) );
  Xpos.PrintTimeValues( );
  fprintf( stderr, "Time runs from %8.3f to %8.3f
", 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
", t, v );
  }
}
```
Computer Graphics

Using the System Clock in Display() for Timing

Instead of Key Frames, I Like Specifying Key Times Better

Forward Kinematics:
Change Parameters – Connected Things Move
(All children understand this)

Inverse Kinematics (IK):
Things Need to Move to a Particular Location – What Parameters Will Make Them Do That?

Number of msec in the animation cycle

#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();
}

Determine Object Locations?

Forward Kinematics: Transformation Hierarchies

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 the final 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?"
Particle Systems: A Cross Between Modeling and Animation?

The basic process is:

- 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"

Animating using Rigid-body Physics

Newton’s second law:

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

or

\[
\ddot{x} = \frac{\text{force}}{\text{mass}}
\]

\[
x(t) = \int \dot{x} \, dt = \sum \ddot{x} \Delta t
\]

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.

Animating using Fluid Physics

D-D

0

k = spring stiffness in 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"
**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

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

Increasing the Repulsion Coefficient
Motion Capture ("MoCap") as an Input for Animation

Motion Capture is for Faces Too

Even Animals can be MoCapped

Tron I –
They probably should have used physics, but didn’t

Card Trick