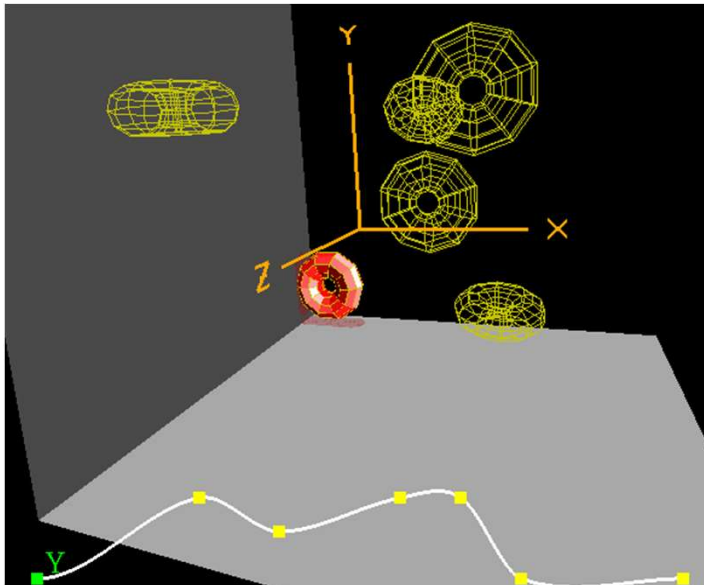


# Simple Keytime Animation for CS 450/550



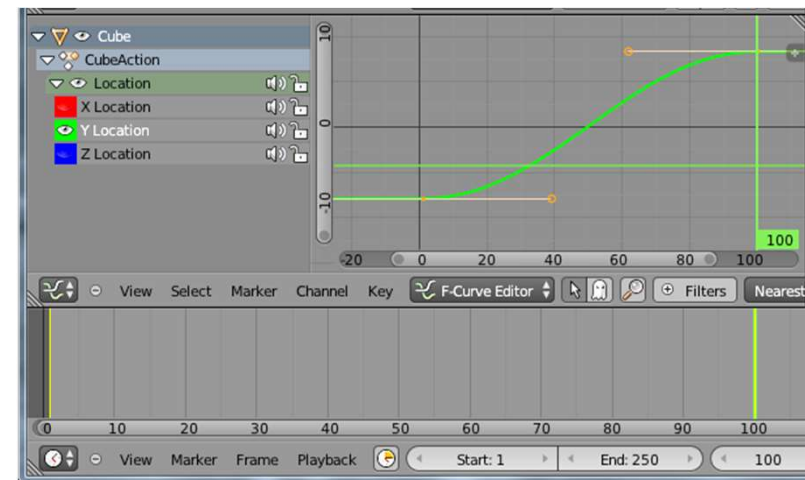
**Oregon State**  
University

**Mike Bailey**

mjb@cs.oregonstate.edu



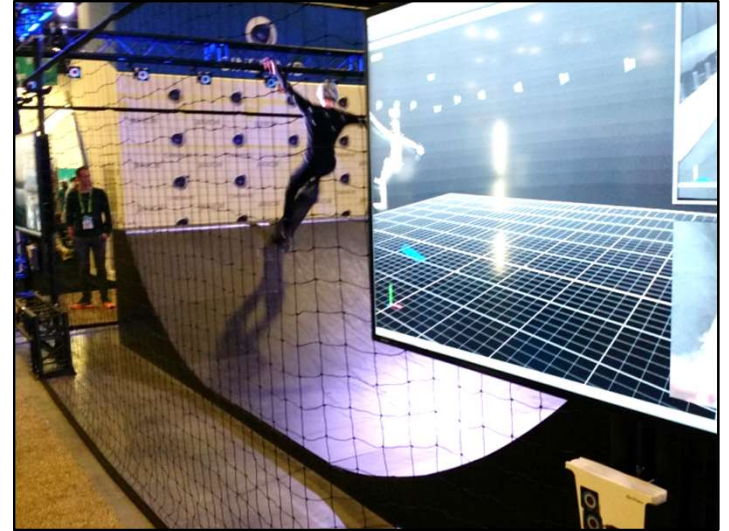
This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nc-nd/4.0/)



**Oregon State**  
University

Computer Graphics

# Approaches to Animation



1. Motion Capture (“MoCap”)
2. Using the laws of physics ( $F=ma$ )
3. Using functional (target-driven) animation
4. Using keyframing

We'll talk more about these  
in the Animation notes!



Oregon State  
University

Computer Graphics

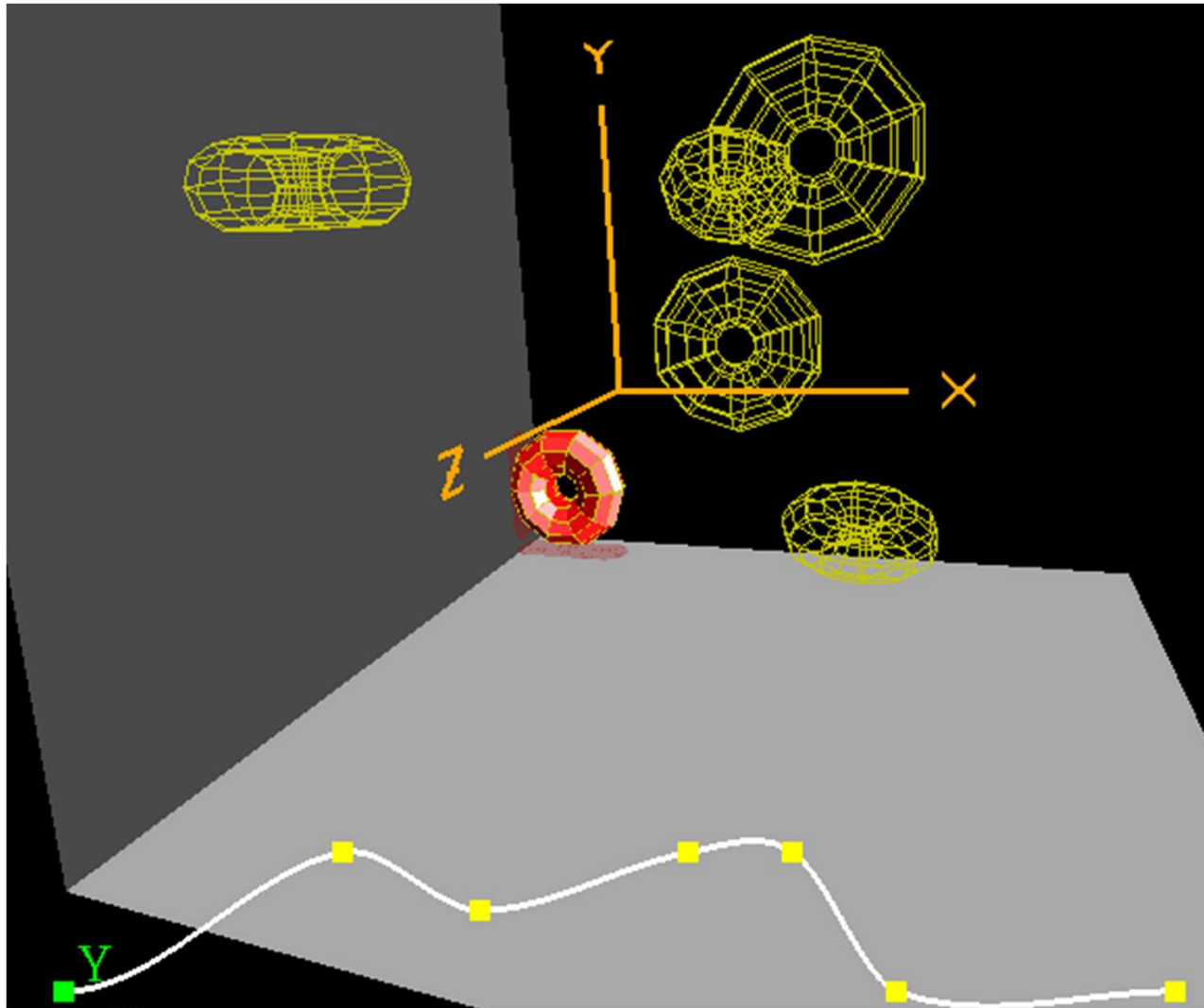
Keyframing involves creating certain *key* positions for the objects in the scene, and then the program later interpolating the animation frames *in between* the key frames.

In hand-drawn animation, the key frames are created by the senior animators, and the in-between frames are developed by the junior animators.

In our case, you are going to be the senior animator, and the computer will do the in-betweening.



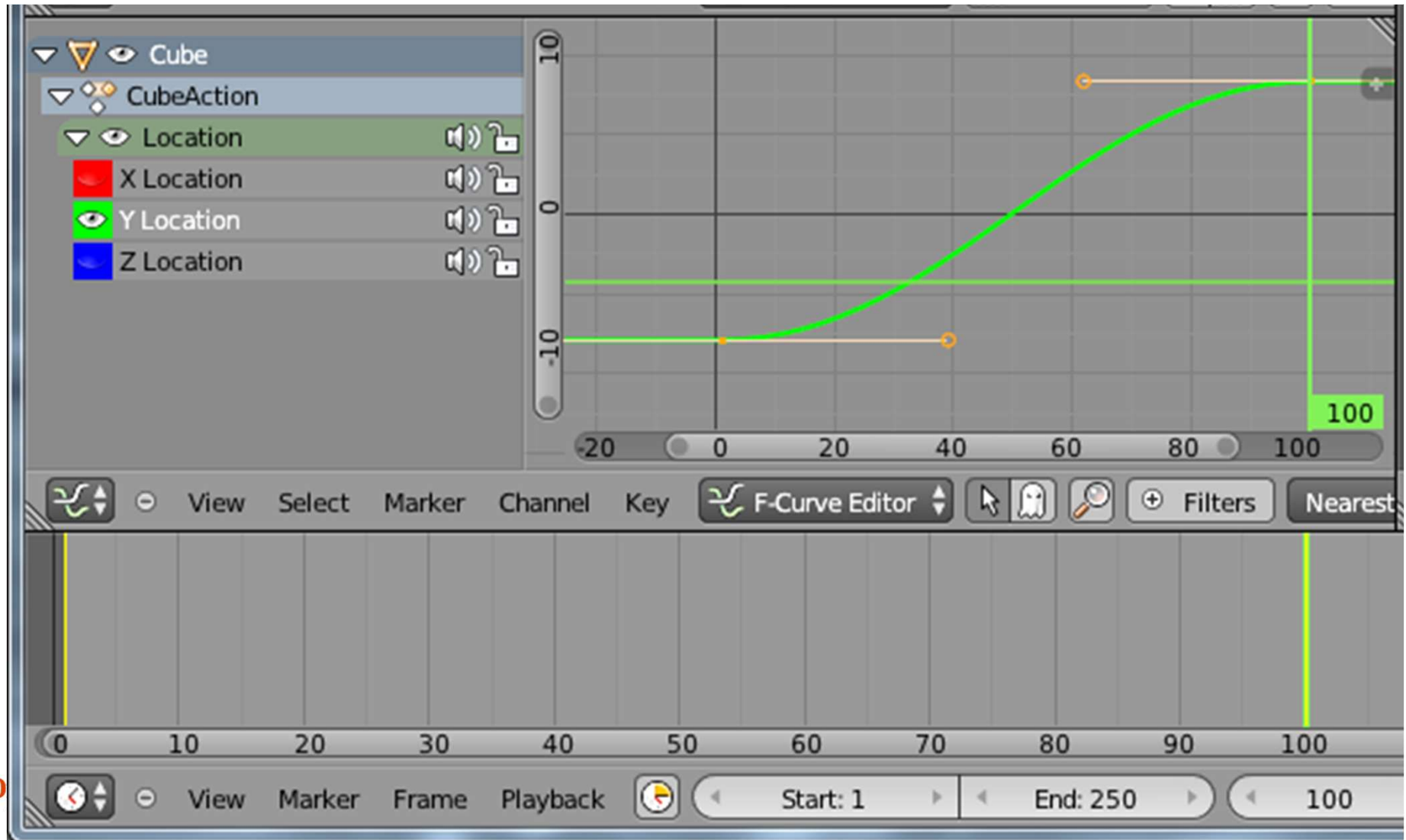
# The General Idea is to Interpolate the In-between Frames from the Smooth Curves Fit through the Key Frames



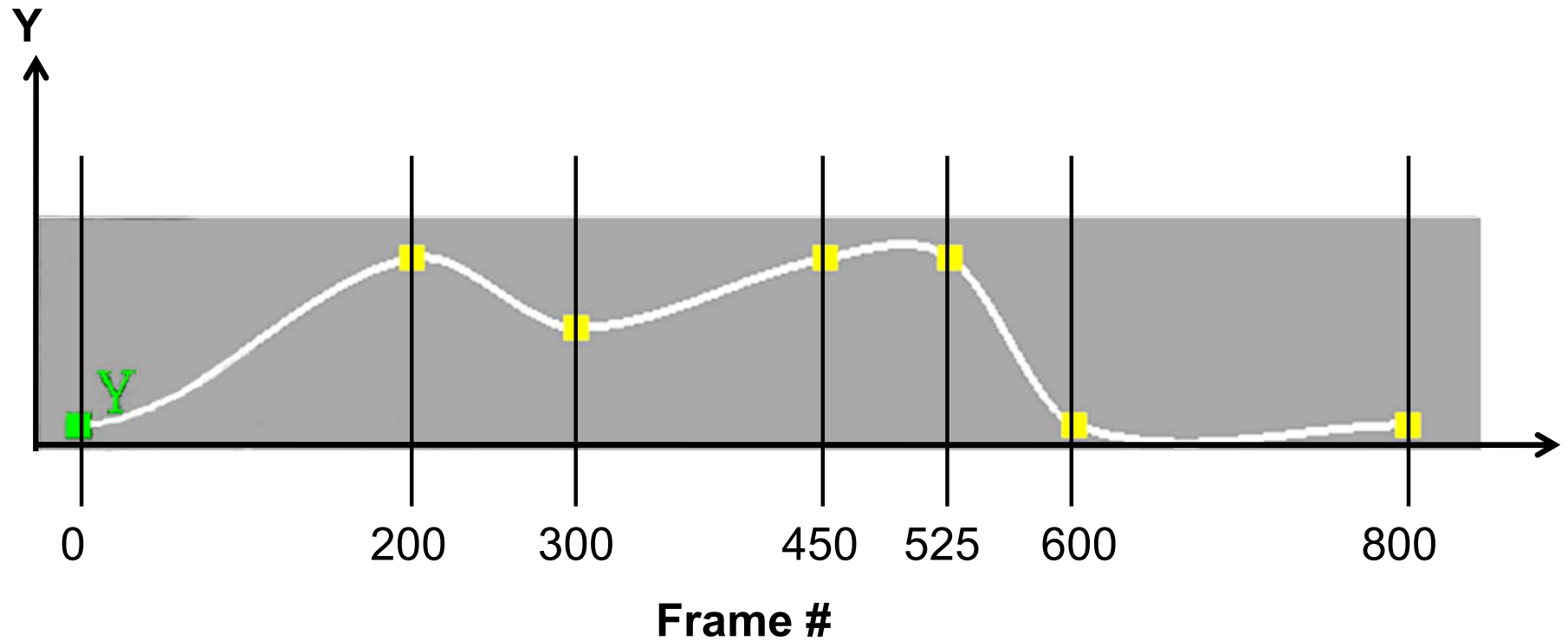
To make this simple to use, our goal is to just specify the keyframe *values*, not the *slopes*. We will let the computer compute the slopes for us, which will then let the in-between frames be computed.

# Many Professional Animation Packages Make You Sculpt the Slopes (but we won't . . .)

Blender:



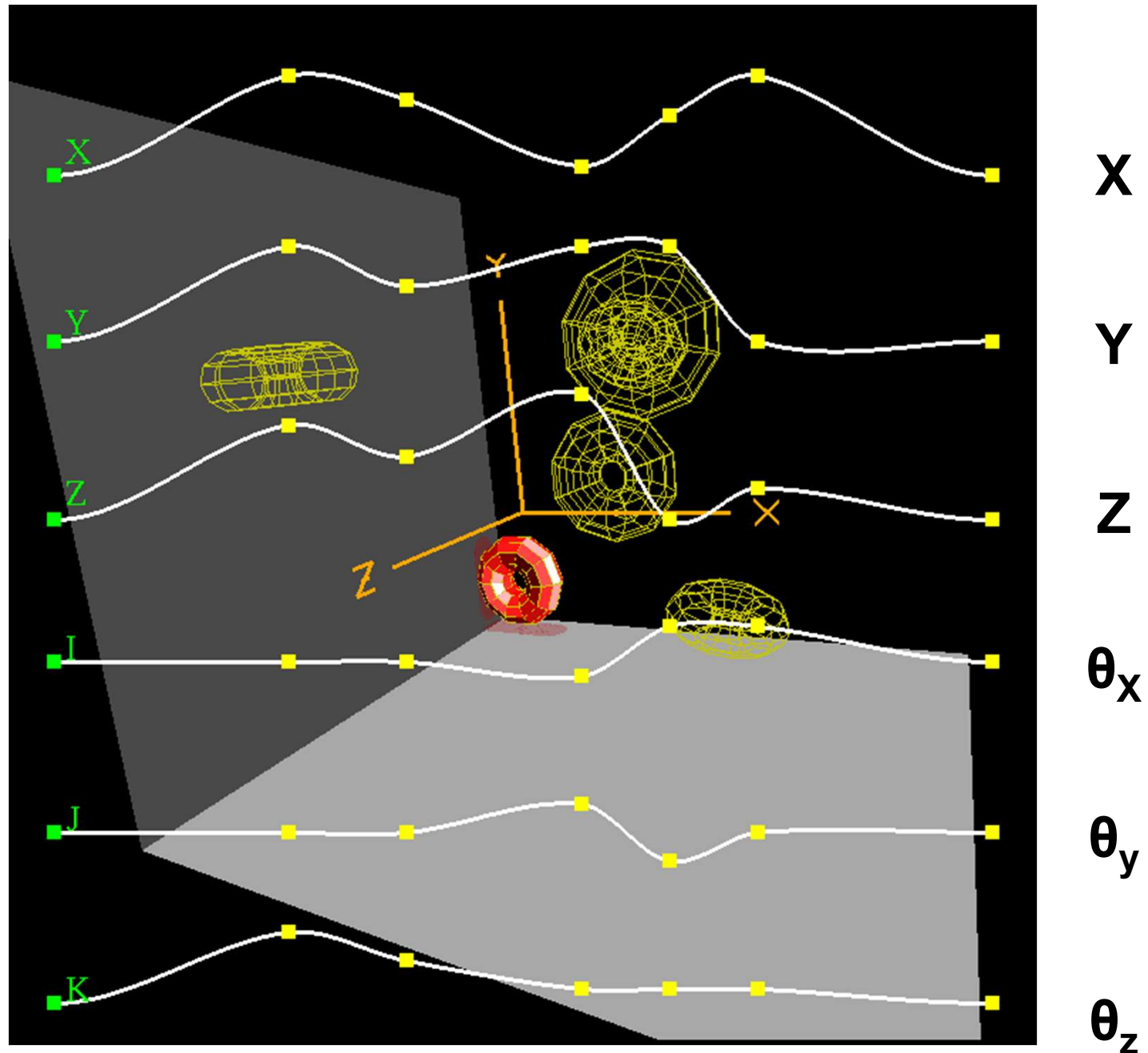
## The “Y vs. Frame” Curve Looks Like This



**Oregon State**  
University

Computer Graphics

# Do This Same Thing for the X, Y, and Z Translations and the X, Y, and Z Rotations



Oregon State  
University

Computer Graphics



## Instead of Key *Frames*, I Like Specifying Key *Times* Better

We created a C++ class to do the interpolation for you

```
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

```

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.f; t <= 2.f; t += 0.1f )
    {
        float v = Xpos.GetValue( t );
        fprintf( stderr, "%8.3ft%8.3f\n", t, v );
    }
}

```



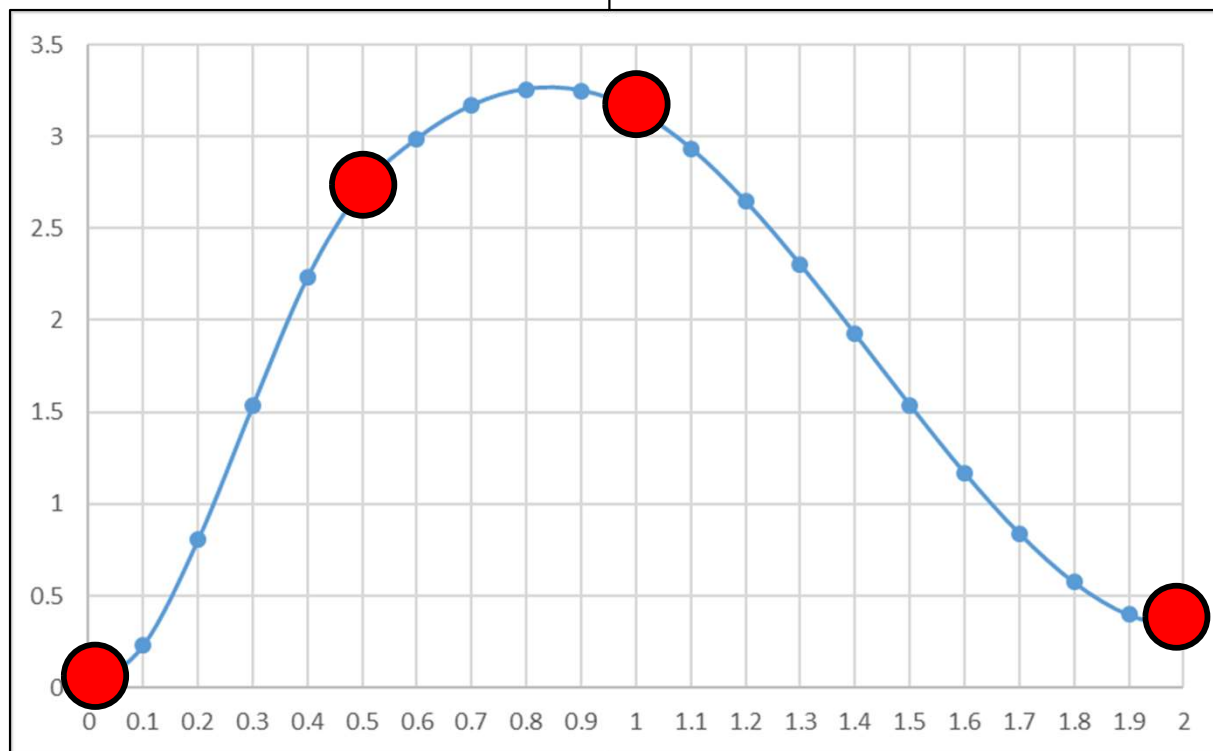
## Instead of Key *Frames*, I Like Specifying Key *Times* Better

( 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

0.000	0.000
0.100	0.232
0.200	0.806
0.300	1.535
0.400	2.234
0.500	2.718
0.600	2.989
0.700	3.170
0.800	3.258
0.900	3.250
1.000	3.142
1.100	2.935
1.200	2.646
1.300	2.302
1.400	1.924
1.500	1.539
1.600	1.169
1.700	0.840
1.800	0.574
1.900	0.397
2.000	0.333



## Using the System Clock in Display( ) for Timing

```
#define MSEC    10000           // i.e., 10 seconds
Keytimes Xpos, Ypos, Zpos;
Keytimes ThetaX, ThetaY, ThetaZ;
```

Number of msec in the animation cycle

**// in InitGraphics( ):**

```
<< init the Keytime classes and add the keyframe values >>
...
```

**// in Display( ):**

*// # msec into the cycle ( 0 - MSEC-1 ):*

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

*// turn that into a time in seconds:*

```
float nowSecs = (float)msec / 1000.f;
```

```
glPushMatrix( );
```

```
    glTranslatef( Xpos.GetValue(nowSecs), Ypos.GetValue(nowSecs), Zpos.GetValue(nowSecs) );
```

```
    glRotatef( ThetaX.GetValue(nowSecs), 1., 0., 0. );
```

```
    glRotatef( ThetaY.GetValue(nowSecs), 0., 1., 0. );
```

```
    glRotatef( ThetaZ.GetValue(nowSecs), 0., 0., 1. );
```

```
    << draw the object >>
```

```
glPopMatrix( );
```

```
}
```

