Animator Help Session

Outline

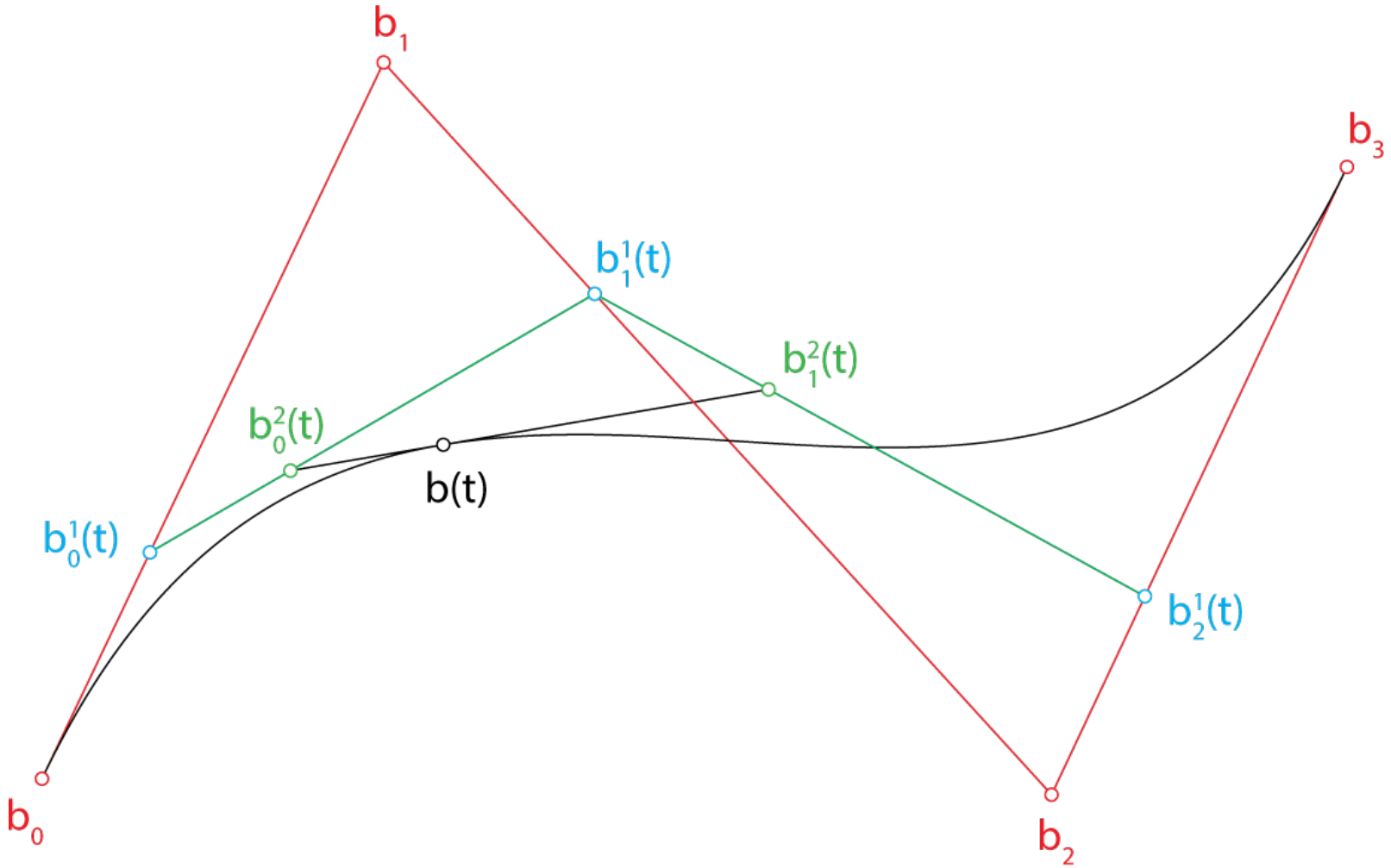
- Intro
- Requirements
  - Curves
  - Particle System
- Artifact

Getting Started

- Replace sample.cpp with your model from Modeler
- Also copy over your shaders and any texture files you used
- Better to do this earlier rather than later

- 2 Views: Modeler View & Curves View
  - Mostly working in curves view for Animator
- Familiarize yourself with the Graph Widget Interface

Curves



CurveEvaluator

- Create a CurveEvaluator for each curve, and implement the evaluateCurve function
  - ptvCtrlPts
    - A collection of control points specified in the curve editor
  - ptvEvaluatedPts
    - A collection of points you've evaluated based on the curve type's formula.
  - fAniLength
    - Max time the curve is defined
  - bWrap
    - flag indicating whether curve should be wrapped. LinearCurveEvaluator does this, you can add it to your curves for EC

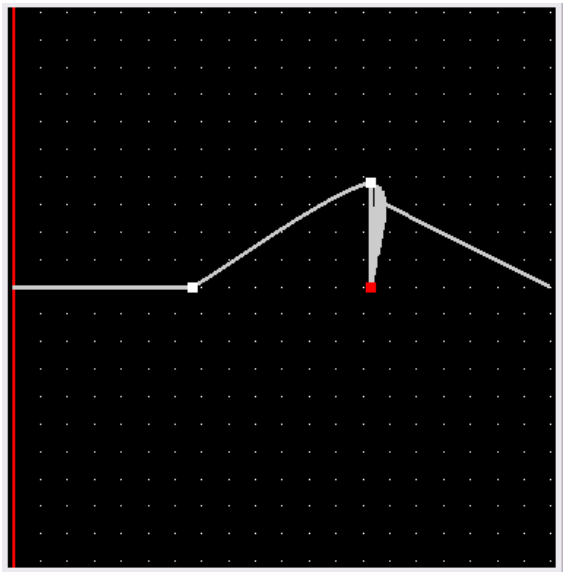
- Look at LinearCurveEvaluator for an example.

Required Curves

- Bezier curve
  - Can linearly interpolate if there are not enough control points (< 4 or for the last couple points)
  - Base requirement: sample u at regular intervals for 0 <= u <= 1
  - EC: Adaptive subdivision
- B-Spline
  - Interpolate endpoints (triple them)
- Catmull Rom
  - Interpolate endpoints (double them)
  - Make sure your curve is a function! -> see next slide

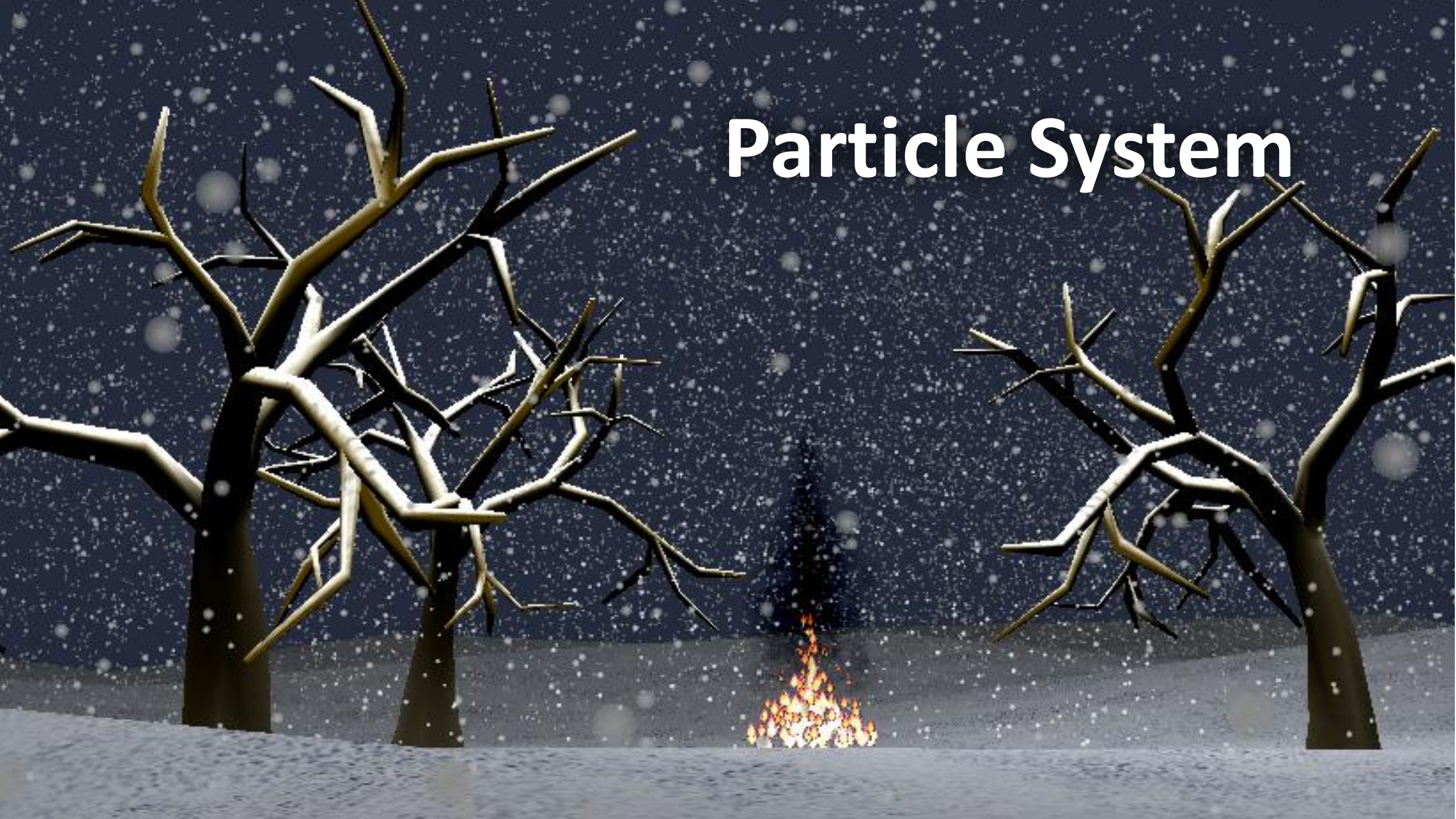
How it works

- Control points will be sorted for you
- The evaluated control points are also sorted
  - Your evaluated control points must be a function!



This is bad

- Evaluation function draws line segments between each of your evaluated points to create a smooth curve
- You'll need to know how to use C++ vectors.
  - Dynamically resizeable array, like an ArrayList in Java
  - http://www.cplusplus.com/reference/stl/vector/
  - http://en.cppreference.com/w/cpp/iterator
- Add your CurveEvaluators in the GraphWidget constructor (UI is already taken care of)


Particle System

Requirements

- Use Euler's method (see lecture notes)
- 2 distinct forces
  - Distinct may mean forces whose values are calculated with different equations (gravity and drag are distinct because gravity eq is of form f=ma, where drag is defined in terms of a drag coefficient and velocity)
- Collision detection
  - With the sphere
  - Another primitive of your choice (ground plane is easiest)
  - Add a slider to control the restitution coefficient
- Attach to a node in your hierarchy that's not your root node

ParticleSystem class

- Simulation Engine
- Skeleton provides rough outline with an interface that must be implemented to work with the UI
- Should have pointers to all particles and a marching variable (time) for simulation
- Implement
  - drawParticles()
  - startSimulation()
  - computeForcesAndUpdateParticles()
  - stopSimulation()
  - + constructor, destructor, etc.

Particle & Force Representation Suggestion

- Particle class
  - you may have several of these if you have multiple types of simulations
  - If this is the case, take advantage of inheritance
- Force class
  - An elegant implementation would include a generic Force class and a variety of distinct forces that inherit from it
- Hint: You can model the collisions as forces

Note: We stress inheritance because it will make your implementation easier (and less messy) and in general will make your life easier. If the TA that grades your project must look at your code, clean object oriented code is a great headache-prevention-tool.

Not the root node!

- Spawn your particles from a node in your hierarchy that is not the root
- Need to find the world coordinates for your particles
  - Why? Ex. If we apply gravity in the local coordinates of your node, then the force in the –y direction is dependent on the orientation of that node, not –y of the world.
  - ModelViewMatrix, Inverse Camera Transformation
- Hook up the particle system with your model

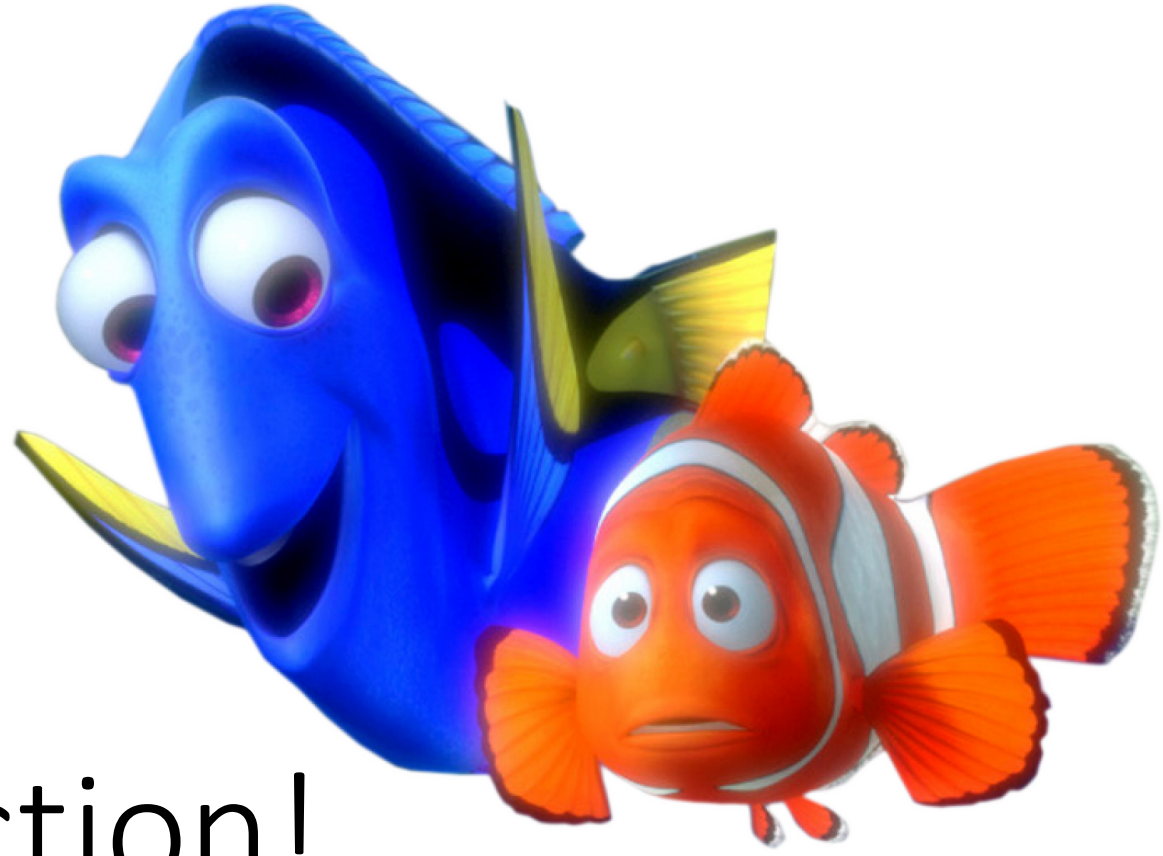
EC ideas (this is just a short list)

Particles

- Cloth simulation
- Flocking
- Billboarding
  - More realistic. Ex. Fire, Snow,
- Baking
  - For complicated simulations with lots of particles

Curves

- Tension control for Catmull Rom
- Allow control points to have (or not have) C0, C1, C2 continuity



Lights, Camera, Action!

Tips for creating a good artifact

Have a plan

- This artifact takes more time than the other projects
- Keep it simple, have realistic goals. If you finish early, then go back and enhance
- Sketch out storyboards and key poses/frames before implementing. It's much easier to iterate on paper than it is using the animator program
- Complicated doesn't always mean better. Well animated simple models are more entertaining than poorly animated complicated models.
- Read John Lasseter's article on animation principles https://courses.cs.washington.edu/courses/cse457/15sp/projects/animator/linkedItems/lasseter.pdf

Choice of curves

- Catmull-Rom is usually the preferred curve choice…
  - but unless your project supports the option to add C1 discontinuity at will, you might find yourself trying to fight the Catmull-Rom to create pauses and other timing goodies
- Bezier spline for things like animating a bouncing ball

Important compositional components

- Timing
  - You can change the animation length of your shots
  - Consider timing before you bother to get specific about joint rotations or object positions
- Music
  - Sound and music can greatly enhance the cohesion of your artifact
  - If your artifact idea includes a theme or stylization, it can be very effective to time your animation with events in the theme music.
- Lighting
  - Like sound, light is very important compositionally
  - Anything you can do to be creative with lighting will help
- Camera Angle
  - Changing the perspective between two shots or panning/zooming the camera for the duration of a shot can add depth

Putting it all together

- You can save your progress and continue later by using “save animation script” and “load animation script”
- We recommended that you break your intended artifact up into shorter clips combining them all in the end.
  - Splitting up work is straightforward
  - Changing camera angles is GOOD for a composition
  - You can incrementally complete your artifact
- Adobe Premier
- 60 seconds MAX

THE END

GOOD LUCK