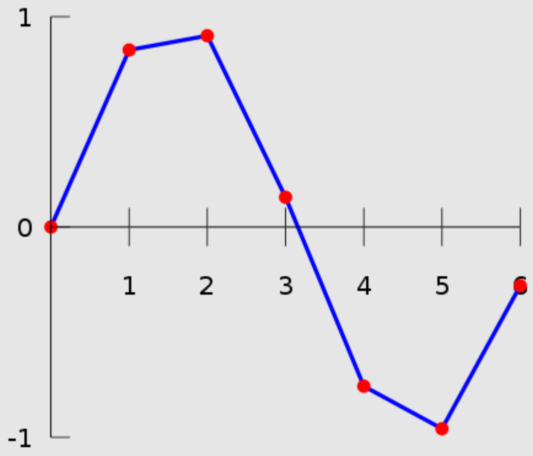
Animation

Interpolation

Alpha system

Animation System

Interpolation



- Estimate values between a start and target
- Linear Interpolation (lerp)

  a + (b – a) * t
- Vary a parameter from a starting to an ending value during a time interval
  - Manipulate transforms, colors, lights, ++
- Math library in jME offers interpolation methods like:
  - Vector3f.interpolate(Vector3f, Vector3f, float t)
  - Quaternion.slerp(Quaternion, Quaternion, float t)
- Use interpolation to animate objects

Interpolation

- jME features complex animation systems
  - Keyframe animation
  - Skeleton animation
- Geared towards animating models / position
- If you want to animate other things you have to implement your own animation system
  - Light, material, scale, ++

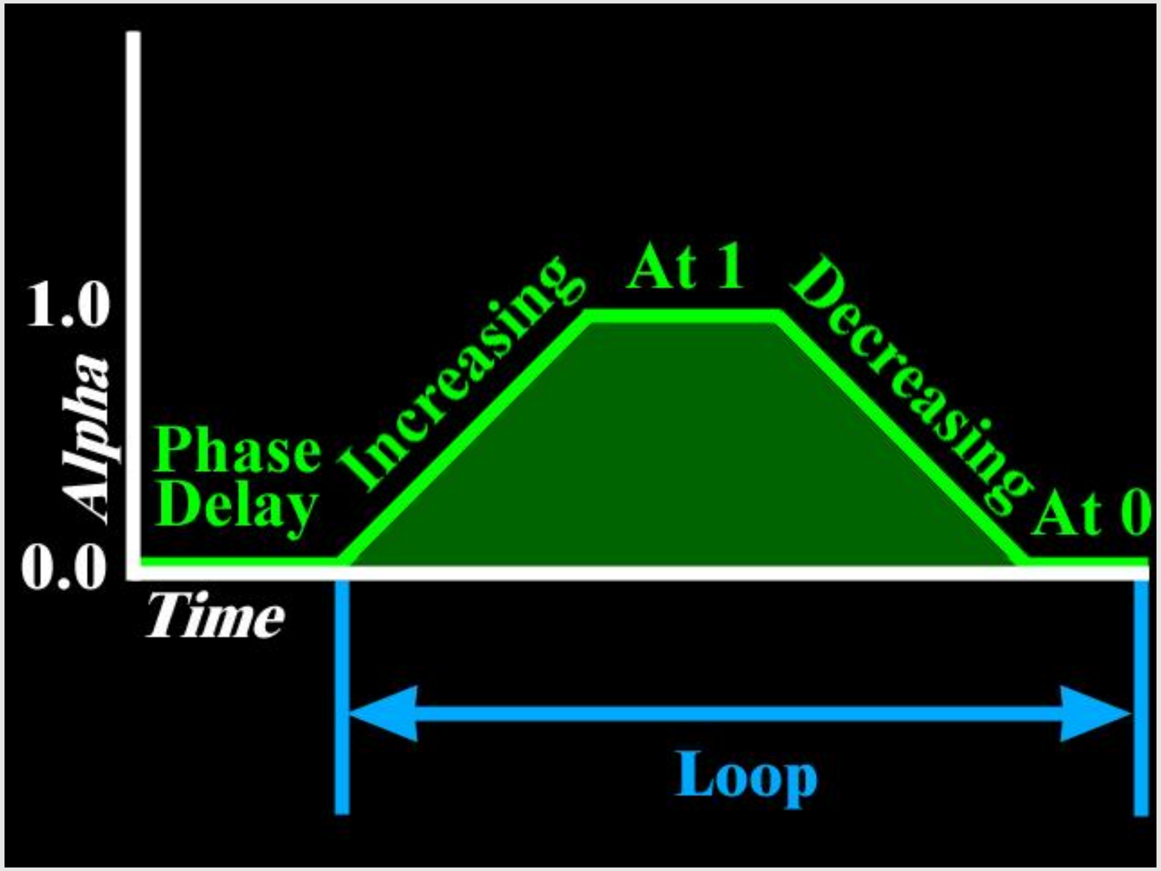
Alpha Animation System

- This is just an example of how you could build your own animation system!
- Implementation of animation system that uses lerp for defined time periods
- Alpha is a generalized value that varies from 0.0 to 1.0 over a time interval
- Use alpha value to map to specific values
  - Transforms, colors, lights, etc.
- Alpha concept is based on similar system in Java 3D

Alpha

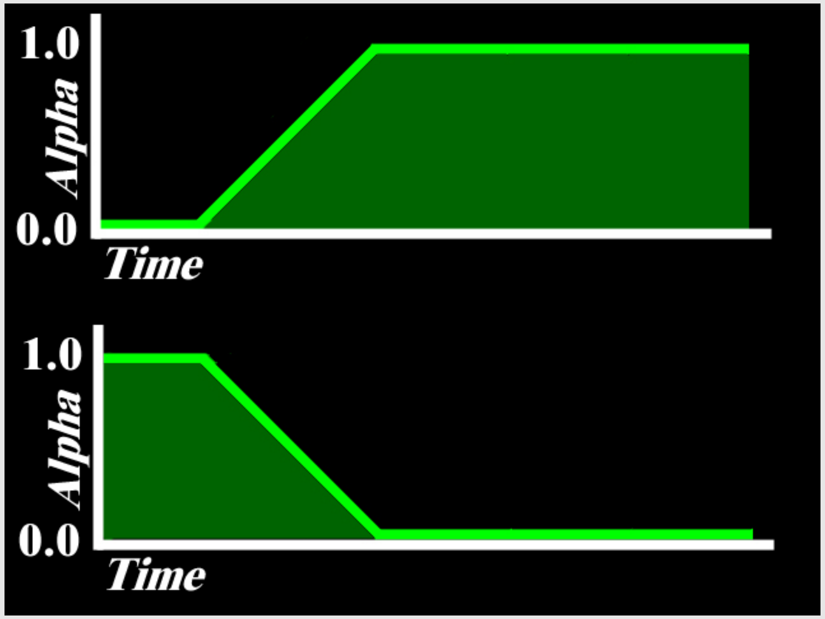
- The *Alpha* object contains:
  - Phase Delay (Start Delay) before initial alpha change
  - Duration of the increasing time for alpha
  - Duration of a faster (ramp) increasing time for alpha (part of normal increasing time)
  - Value for the faster (ramp) increasing to go to (instead of 1)
  - Duration for alpha to stay at 1
  - Duration of the decreasing time for alpha
  - Duration of a faster (ramp) increasing time for alpha (part of normal decreasing time)
  - Value for the faster (ramp) decreasing to go to (instead of 0)
  - Duration for alpha to stay at 0

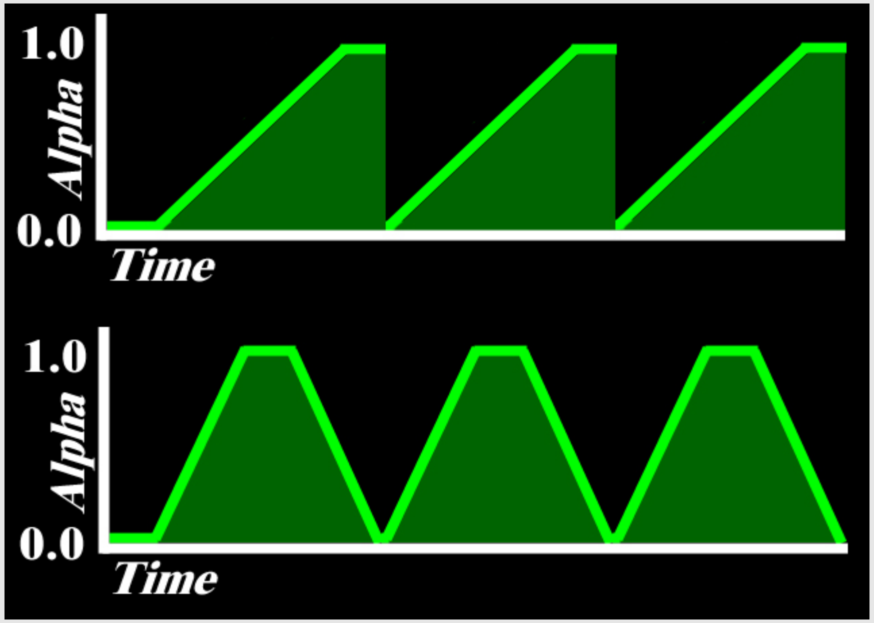
Alpha


1.0
Alpha
0.0
Time
Phase Delay
Increasing
At 1
Decreasing
At 0
Loop

Alpha

- Alpha can be used to animate different changes over time



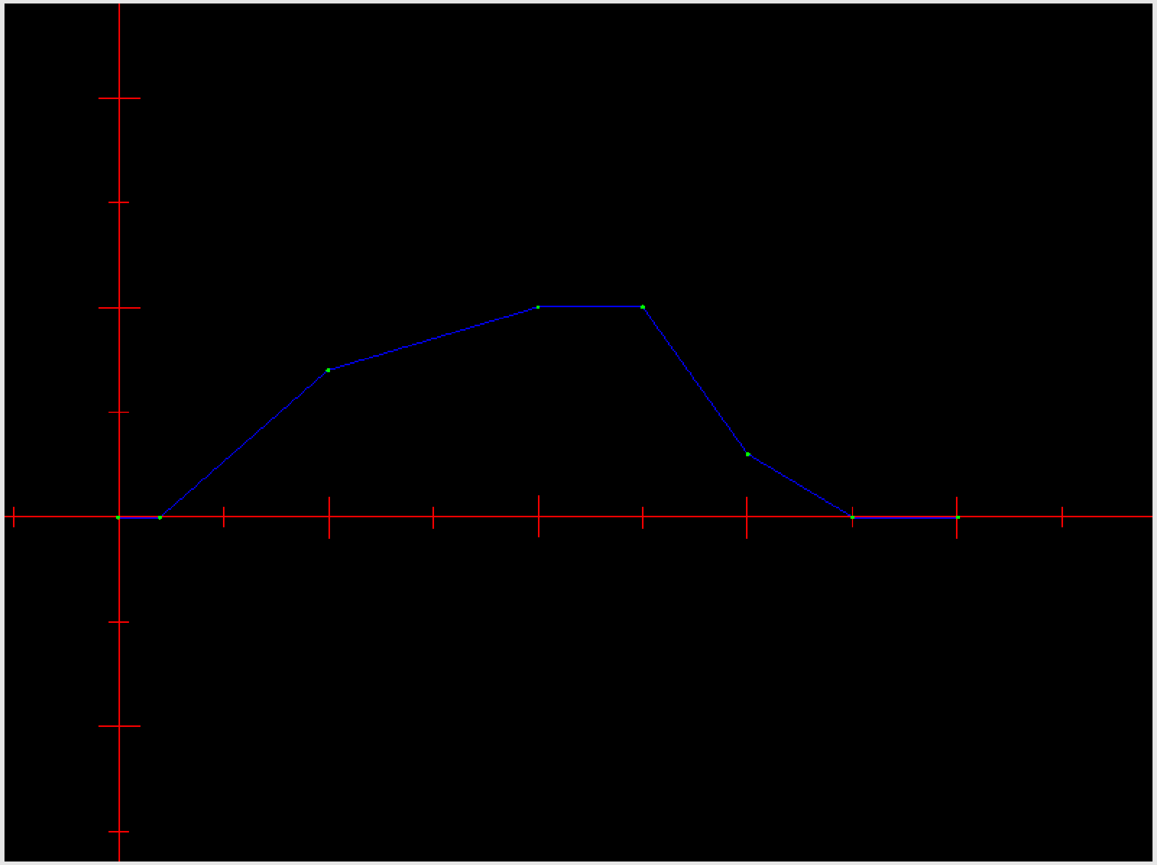


Alpha class methods

- Time values are given in seconds

```
public Alpha(float startDelay,
         float increasingAlphaDuration,
         float increaseingAlphaRampDuration,
         float alphaAtOneDuration,
         float decreasingAlphaDuration,
         float decreasingAlphaRampDuration,
         float alphaAtZeroDuration,
         float increaseRampValue, float decreaseRampValue);

public float value(float tpf);
public float getTotalTimeDuration();
public float getDuration();
public float getEndValue();
public void reset();
public boolean isFinished();
```

Alpha Visualisation Example

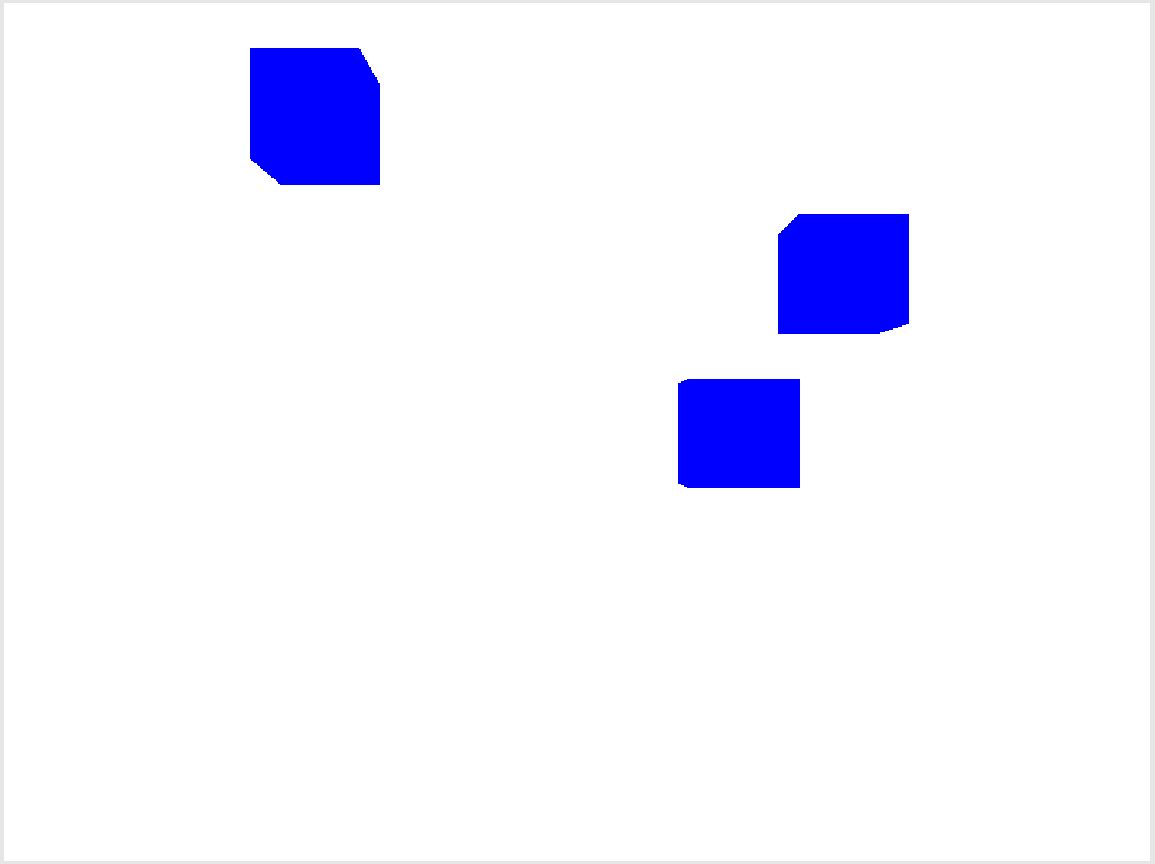


AlphaVisualisation.java

Building custom animation

- Can be built using Controls (in jME)
- Use for example the Alpha value to transition between start and end target
- Build the interpolation controls that you need

Position Interpolator Example

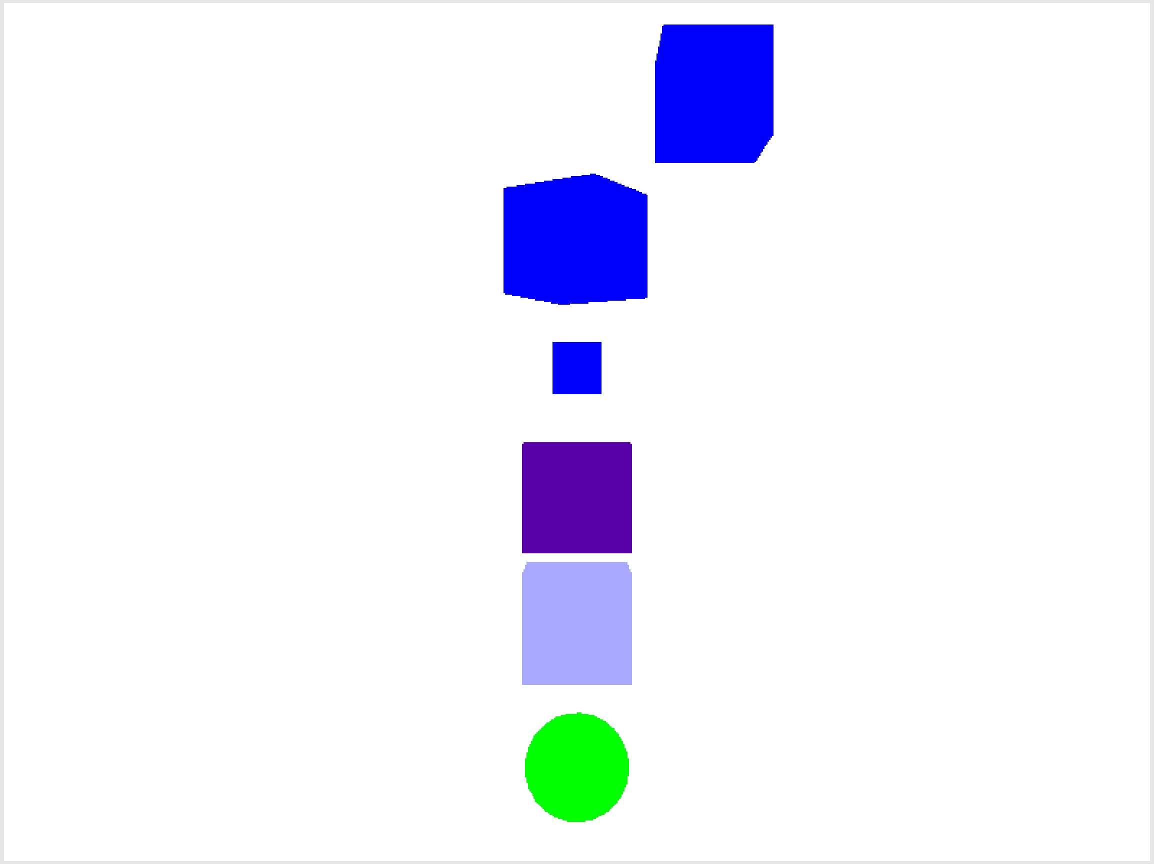


PositionInterpolation.java

Custom animation using Alpha

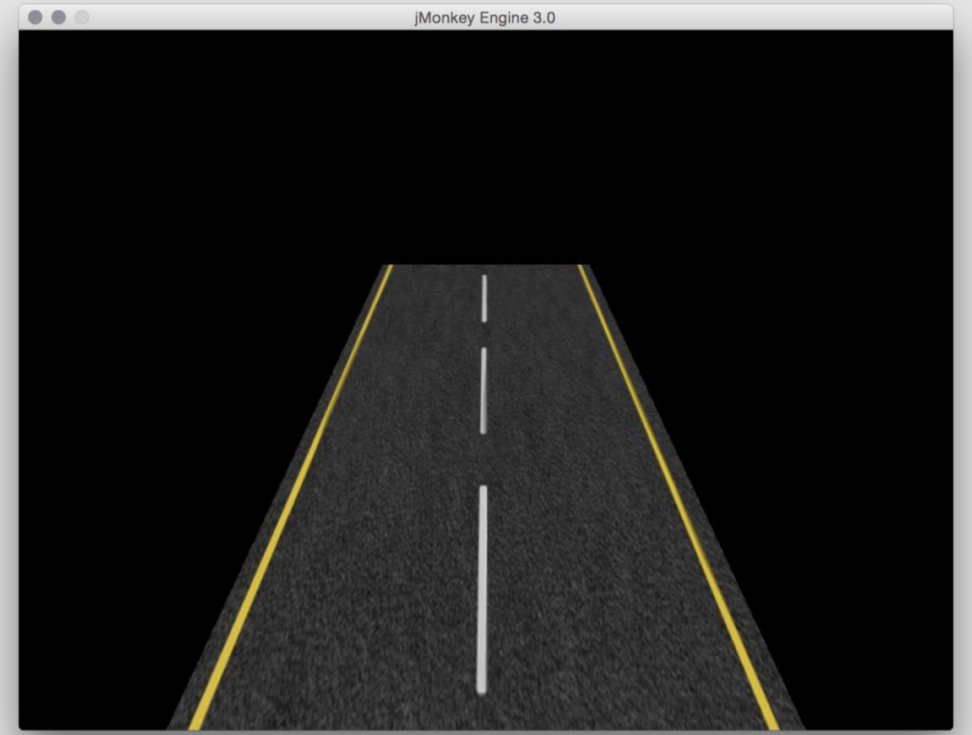
- Some examples:
  - PositionInterpolatorControl
  - RotationInterpolatorControl
  - ScaleInterpolatorControl
  - ColorInterpolatorControl
  - TransparencyInterpolatorControl
  - LightInterpolatorControl
  - SwitchInterpolatorControl

Various Interpolators



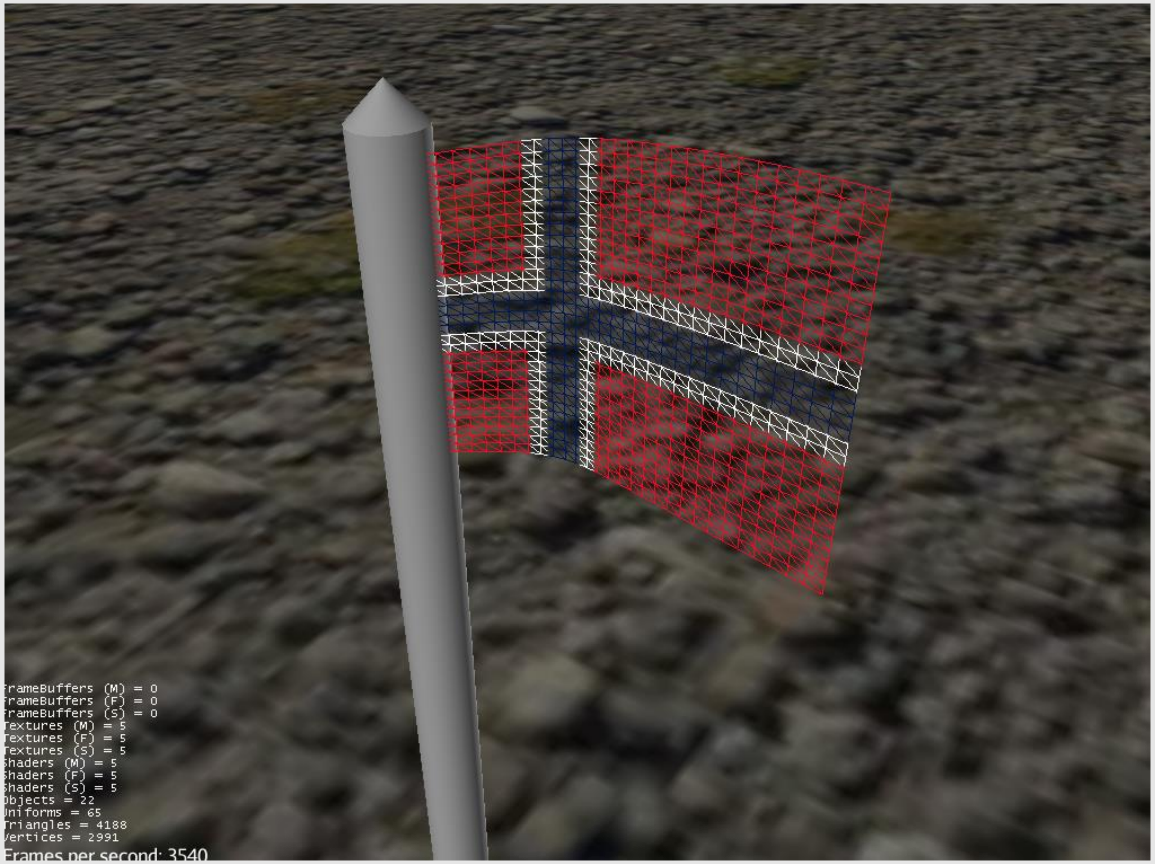
InterpolationExamples.java

Animating Texture Coordinates Example



ScrollingTexture.java

Vertex Shader Animation Example



```
rameBuffers (M) = 0
rameBuffers (F) = 0
rameBuffers (S) = 0
extures (M) = 5
extures (F) = 5
extures (S) = 5
haders (M) = 5
haders (F) = 5
haders (S) = 5
bjects = 22
niforms = 65
riangles = 4188
ertices = 2991
Frames per second: 3540
```

WavingFlagExample.java

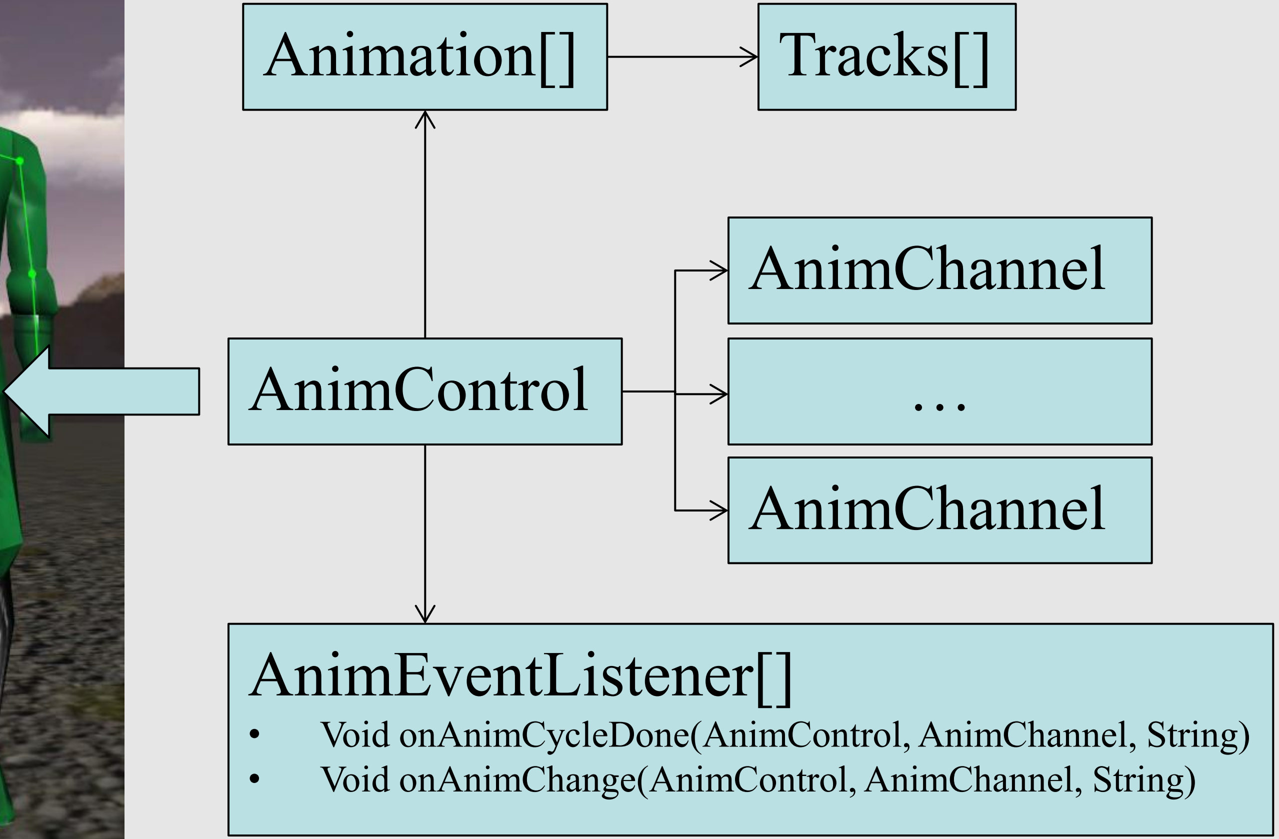
Animation System

Animation System in jME

- Rigging of internal skeleton (bones)
- Mapping polygons affected by bones (skinning)
- Animate using keyframes

- This is represented in jME as
  - Animation Controls
  - Animations
  - Animation Channels
  - Animation Listeners
  - +++

Animation System in jME



AnimEventListener[]

- Void onAnimCycleDone(AnimControl, AnimChannel, String)
- Void onAnimChange(AnimControl, AnimChannel, String)

Animation Controls

- The *AimControl* class
  - One AnimControl per animated model
  - It is a Spatial Control
  - Contains the skeleton
  - Contains the animation
  - Gives access to available animation sequences
  - Contains multiple Animation Channels
  - Contains multiple Animation Event Listeners

Animation

- The *Animation* class
  - Represents an animation
  - Consists of animation Tracks
  - Updates the animation target with the tracks
- Different Tracks
  - SpatialTrack, BoneTrack, AudioTrack, PoseTrack, EffectTrack

Animation Channels

- The *AimChannel* class
  - Used for playing an animation
  - Run the animation
    - Play, pause, fast forward, etc.
    - Set loop mode
- Different channels can play different animations on the same model at the same time
  - One channel plays walking animation
  - Another channel plays shooting animation

Animation Listeners

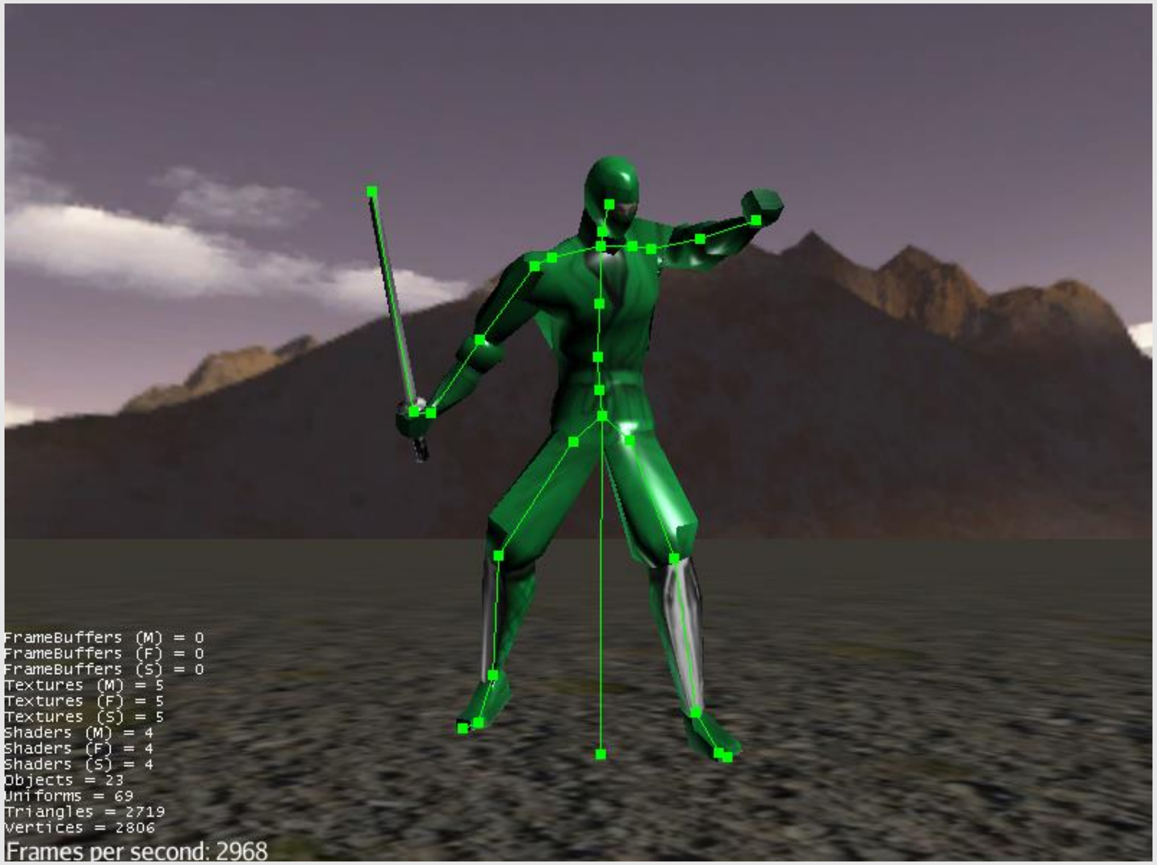
- Listener is added to the AnimControl
- Callback when animations start and end
- Two method callbacks:
  - onAnimCycleDone(AnimControl, AnimChannel, String)
  - onAnimChange(AnimControl, AnimChannel, String)

Animations

- Animations can be loaded from file, or done programatically
- Skeleton animation and skinning is usually done in modelling tools

```
Spatial player = assetManager.loadModel("Models/Oto/Oto.mesh.xml";
AnimControl playerControl = player.getControl(AnimControl.class);

AnimChannel channel_walk = playerControl.createChannel();
AnimChannel channel_jump = playerControl.createChannel();

channel_walk.setAnim("Walk");
channel_jump.setAnim("Jump");
```

Skeletal Animation Example



```
FrameBuffers (M) = 0
FrameBuffers (F) = 0
FrameBuffers (S) = 0
Textures (M) = 5
Textures (F) = 5
Textures (S) = 5
Shaders (M) = 4
Shaders (F) = 4
Shaders (S) = 4
Objects = 23
Uniforms = 69
Triangles = 2719
Vertices = 2806
Frames per second: 2968
```

SkeletalAnimation.java

TWi March 15

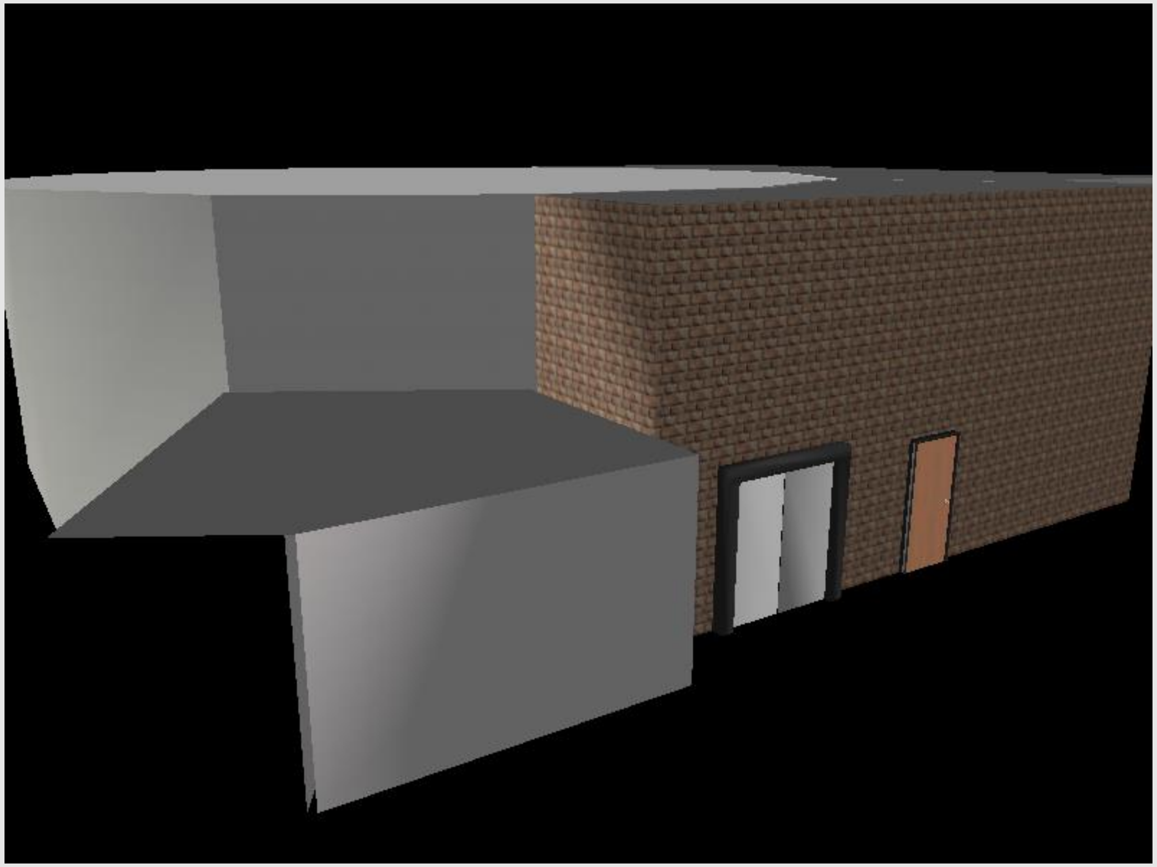
Animations

- Can be built programatically
- The helper class *AnimationFactory*
  - Animation duration, framerate
  - Transforms for keyframes
  - Builds the animation

AnimationFactory

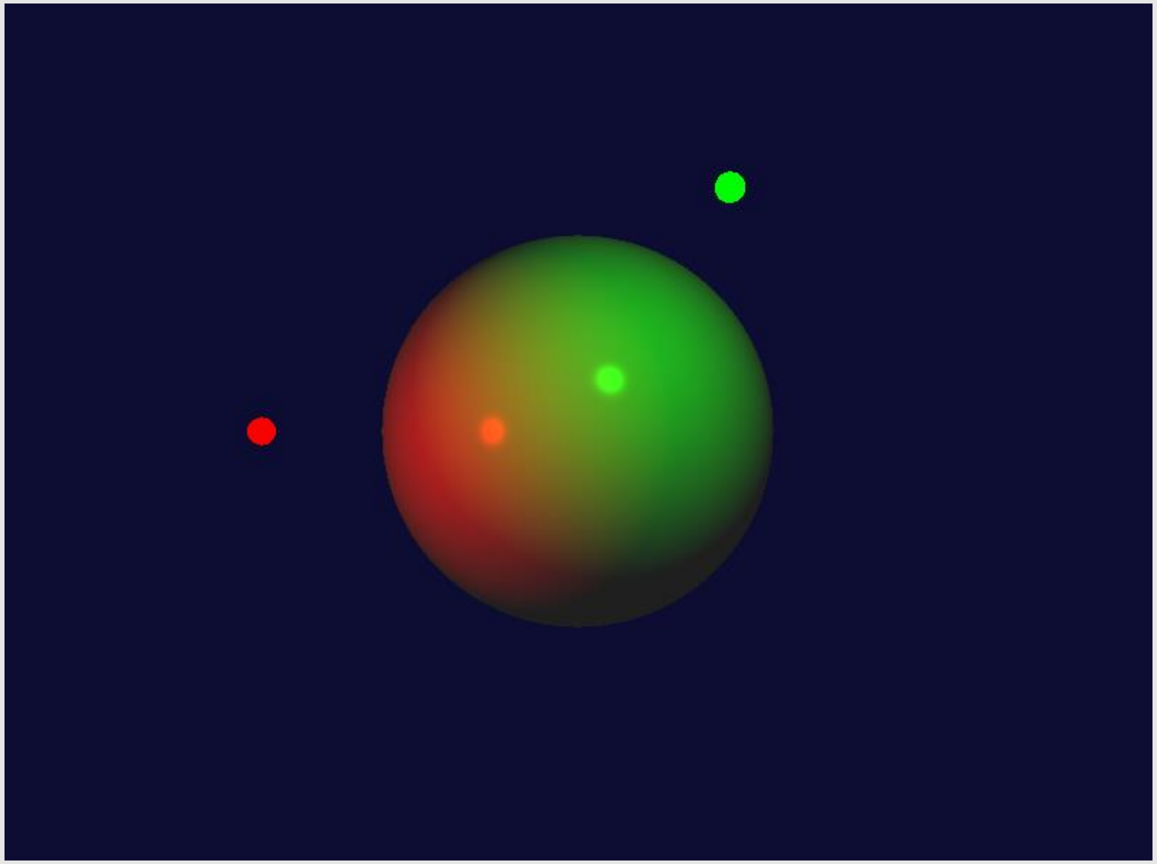
```
AnimationFactory animFactory = new AnimationFactory(30, "move", 60);
animFactory.addTimeTransform(0, new Transform(new Vector3f(0, 0, 0),
                                  new Quaternion(),Vector3f.UNIT_XYZ);
animFactory.addTimeTransform(15, new Transform(new Vector3f(0, 0, -5),
                                  new Quaternion(),Vector3f.UNIT_XYZ);
animFactory.addTimeTransform(30, new Transform(new Vector3f(0, 0, 0),
                                  new Quaternion(),Vector3f.UNIT_XYZ);
Animation animation = animFactory.buildAnimation();
AnimControl animControl = new AnimControl();
HashMap<String, Animation> animationMap = new HashMap<String, Anim>();
animationMap.put("move", animation);
animControl.setAnimations(animationMap);

AnimChannel animChannel = animControl.createChannel();
animChannel.setAnim("move");
animChannel.setLoopMode(LoopMode.DontLoop);
```

Keyframe Animation Example



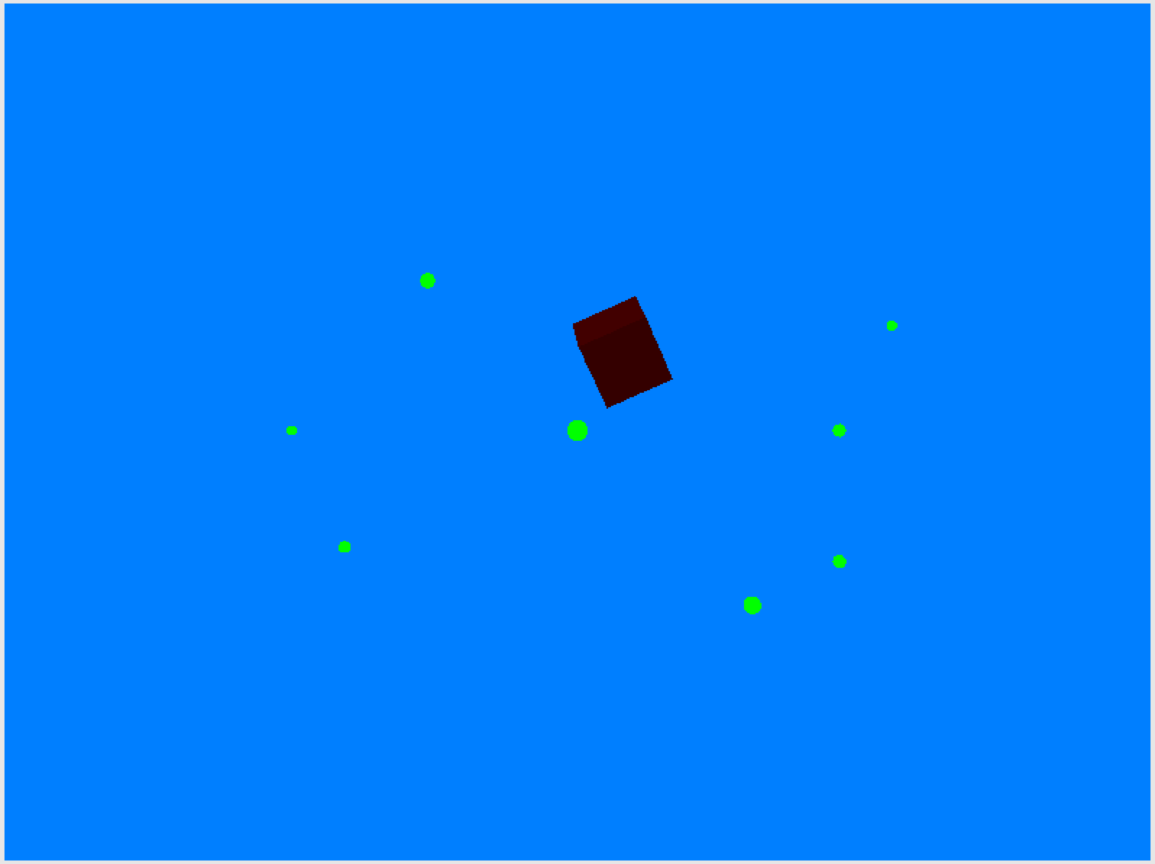
AnimationFactoryExample.java

Manual Keyframe Animation Example



ManualAnimationExample.java

Keyframe Animation Example



RotPosPathExample.java

TWi March 15