Animation in Java 3D

- Animation is a change without any direct user action
- Time-based animation
  - Interpolators
  - Alpha objects
  - Custom behaviours
- Collision detection
  - Same as for 2D
  - Bounding boxes and spheres
Time in Java 3D

- Java 3D rendering engine runs in a continuous loop at as fast a frame rate as possible
- WakeUp criteria
  - On elapsed time
  - One elapsed frames
- Alpha object
  - Value from 0.0 to 1.0
  - Value changes as a continuous function of time

Alpha Objects

- Synchronized against Java 3D system start time
- Four phases

![Figure 5-2 Phases of the Alpha Waveform](image-url)
Alpha Objects (2)

- Patterns using one to all four phases
- Fixed number of cycles or continuous

![Waveforms Diagram](image)

Figure 5-3: Some Basic Waveforms Easily Made with an Alpha Object.

Alpha Objects (3)

- Waveforms can be smoothed
  - Useful for acceleration / deceleration effects

![Smoothing Diagram](image)

Figure 5-4: Smoothing of the Waveform Produced by Alpha.
Interpolators

- Change a property of a visual object
  - Location (translation) and orientation (rotation)
  - Size, colour and transparency
- Property changes over time
  - Interpolated against an Alpha object value

Built-in Interpolators

- Utility classes provided for interpolating most object properties of interest
Hand-crafted Animation

- Can write custom behaviours that use Alphas
  - value() method of Alpha object
- But movement in games is not always regular
  - Determined by interaction between sprites
  - Game state
- Custom animation behaviors using time or frame-based wakeup criteria
  - Identical to approach used for 2D games

The Animated Hand

- Davison, "Killer Game Programming" chapter 18
- Movement updated every few milliseconds via a custom behavior
- TimeBehavior has a reference to the AlienSprite object
- TimeBehavior calls the AlienSprite update() method every time it wakes up
- AlienSprite update() calculates the new sprite position based on location of player sprite and required distance to move on each update
**TimeBehavior**

public class TimeBehavior extends Behavior {
    private WakeupCondition timeOut;
    private AlienSprite alien;

    public TimeBehavior(int timeDelay, AlienSprite as) {
        alien = as;
        timeOut = new WakeupOnElapsedTime(timeDelay);
    }

    public void initialize() {
        wakeupOn(timeOut);
    }

    public void processStimulus(Enumeration criteria) {
        // ignore criteria
        alien.update();
        wakeupOn(timeOut);
    }
}

**AlienSprite**

public class AlienSprite extends TourSprite {
    private TourSprite ts;
    private double currAngle; // current y-axis angle

    public AlienSprite(Obstacles obs, TourSprite ts) {
        super(fnm, obs);
        this.ts = ts;
        currAngle = 0.0;
    }

    public void update() {
        if (isActive()) {
            headTowardsTourist();
        }
    }
}
Collision Detection

- Collision detection based on bounding regions of objects
  - Boxes (cubes), spheres
  - More complex polytopes
  - Define explicitly for each object of interest
- Need to detect when bounding regions of colliding objects overlap
  - Bounding regions in Java 3D have a built-in method intersect() that can detect overlap with another bounding region
- Take appropriate action in response to collision
Example Time Behavior

```java
public TimeBehavior(int timeDelay, TransformGroup oTG, float orad, Bounds obsBnds)
// oTG is transform group of object to be controlled
// orad is radius of object (for collision detection)
// obsBnds is bounds of obstacle (for collision detection)
{
    objectTG = oTG;
    collRad = orad;
    obsBounds = obsBnds;
    t3d = new Transform3D();
    toMove = new Transform3D();

    currMove = new Vector3d(0, 0, -MOVERATE);

    timeOut = new WakeupOnElapsedTime(timeDelay);
}

public void initialize()
{ wakeupOn( timeOut );
}
```

Example Time Behavior (2)

```java
public void processStimulus( Enumeration criteria )
{ // ignore criteria
    currMove = doMove( currMove );
    wakeupOn( timeOut );
} // end of processStimulus()

private Vector3d doMove(Vector3d theMove)
// Move the sprite by the amount in theMove
{
    objectTG.getTransform( t3d );
    toMove.setTranslation(theMove); // overwrite previous trans
    t3d.mul(toMove);
    Vector3d trans = new Vector3d();
    t3d.get( trans ); // get translational component of transform
    Point3d newLoc = new Point3d( trans.x, trans.y, trans.z); // next location
    BoundingSphere testBnds = new BoundingSphere(newLoc, collRad);
    // only allow move if does not intersect with obstacle
    if ( !obsBounds.intersect(testBnds) )
        objectTG.setTransform(t3d);
    return theMove;
} // end of doMove()
```
Collision Detection Recipe

- Give each object an associated bounding region
  - Cube, sphere
  - E.g. BoundingSphere playerbnds = new BoundingSphere();
  - When an object moves, its bounding region must be moved with it
    - Manually apply transform or recreate bounds at new location
    - Add bounds to a BoundingLeaf node

- Time-based behavior checks for (predicted) intersections between bounds of visible objects
  - Schemes for checking similar to 2D
  - E.g. if (playerbnds.intersect(enemybnds)) { fix collision }

The End