Active Objects

• Animations!

• What are they?
  – Snapshots of objects in different positions
  – When viewed quickly appear to be moving

A Virtual Flipbook:

Active Objects

• Animations!

• What are they?
  – Snapshots of objects in different positions
  – When viewed quickly appear to be moving

Active Objects

• Animations!

• What are they?
  – Snapshots of objects in different positions
  – When viewed quickly appear to be moving

Active Objects

• Animations!

• What are they?
  – Snapshots of objects in different positions
  – When viewed quickly appear to be moving

Active Objects

• Animations!

• What are they?
  – Snapshots of objects in different positions
  – When viewed quickly appear to be moving

FallingBall

public class FallingBall extends ActiveObject {

  public FallingBall( Location initialLocation, DrawingCanvas aCanvas ){
    canvas = aCanvas;
    ballGraphic = new FilledOval (initialLocation, SIZE, SIZE, canvas );
    start();
  }

  public void run() {
    while ( ballGraphic.getY() < canvas.getHeight() ) {
      if ( ballGraphic.getY() < canvas.getHeight() ) {
        ballGraphic.move( 0, Y_STEP );
        pause( DELAY_TIME );
      } else {
        ballGraphic.removeFromCanvas();
      }
    }
  }
}

Defining an Active Object

• define a class that extends ActiveObject

• ensure the class includes a run method

• pause occasionally within the run method

• include start(); usually as the last statement in the constructor
Using Active Objects

```java
public class FallingBallController extends WindowController {
    private FallingBall droppedBall;
    public void begin() {
        new Text("Click to make a falling ball…", INSTR_LOCATION, canvas);
    }
    public void onMouseClick(Location point) {
        droppedBall = new FallingBall(point, canvas);
    }
}
```

What if…

- We wanted the falling object to look like a raindrop

```
public class FallingRainPicController extends WindowController {
    private Image rainPicture;
    public void begin() {
        new Text("Click to make a falling raindrop…", INSTR_LOCATION, canvas);
        rainPicture = getImage("raindrop.gif");
    }
    public void onMouseClick(Location point) {
        new FallingRainDrop(rainPicture, point, canvas);
    }
}
```

Image and VisibleImage Classes

- In the controller class:
  ```java
  private Image rainPicture;
  rainPicture = getImage("raindrop.gif");
  ```
- Can be used anywhere:
  ```java
  new VisibleImage(rainPicture, 0, canvas.getWidth()/2, canvas);
  ```

A Controller Class

```java
public class FallingRainPicController extends WindowController {
    private Image rainPicture;
    public void begin() {
        new Text("Click to make a falling raindrop…", INSTR_LOCATION, canvas);
        rainPicture = getImage("raindrop.gif");
    }
    public void onMouseClick(Location point) {
        new FallingRainDrop(rainPicture, point, canvas);
    }
}
```

A Falling Raindrop

```java
public class FallingRainDrop extends ActiveObject {
    private static final int DELAY_TIME = 33;
    private static final double Y_SPEED = 4;
    private VisibleImage ballGraphic;
    private DrawingCanvas canvas;
    public FallingRainDrop(Image rainPic, Location initialLocation, DrawingCanvas aCanvas) {
        canvas = aCanvas;
        ballGraphic = new VisibleImage(rainPic, initialLocation, canvas);
        start();
    }
    public void run() {
        while (ballGraphic.getY() < canvas.getHeight()) {
            ballGraphic.move(0, Y_SPEED);
            pause(DELAY_TIME);
            ballGraphic.removeFromCanvas();
        }
    }
}
```

Active Object Interactions

- Accomplished with methods and parameters

A RainCloud:
- RainCloud continuously generates drops
- Drops fall and disappear at the bottom
The RainCloud Class

```java
public class RainCloud extends ActiveObject {
    public RainCloud( DrawingCanvas aCanvas, Image aRainPic) {
        rainPic = aRainPic;
        canvas = aCanvas;
        start();
    }
    public void run() {
        RandomIntGenerator xGenerator = new RandomIntGenerator(0, canvas.getWidth());
        int dropCount = 0;
        while (dropCount < MAX_DROPS) {
            new FallingRainDrop(rainPic, new Location(xGenerator.nextValue(), 0), canvas);
            pause(DELAY_TIME);
            dropCount++;
        }
    }
}
```

Interactions Between Active and Non-Active Objects

- **An Active Object:**
  - Droplet in the shape of an oval
    - Created at the top of canvas, falls to the bottom
- **A Non-Active Object:**
  - Waterline in the shape of a FilledRect
    - Rises as drops fall

A New Kind of Raindrop

1. Created at the top of the screen
2. Falls to the bottom
3. Raises the waterline

Active Objects without Loops

- **Not required to loop**

```java
public class CreditScroller extends ActiveObject {
    public CreditScroller( DrawingCanvas aCanvas ) {
        canvas = aCanvas;
        start();
        new Credit( "Producer . . . Martha Washington ", canvas );
        pause(DELAY_TIME);
        new Credit( "Director . . . George Washington ", canvas );
        pause(DELAY_TIME);
    }
    public void run() {
        new Credit( "Producer . . . Martha Washington ", canvas );
        pause(DELAY_TIME);
        new Credit( "Director . . . George Washington ", canvas );
        pause(DELAY_TIME);
    }
}
```

```java
public class FallingDroplet extends ActiveObject {
    private FilledOval dropletGraphic;
    private FilledRect collector;
    public FallingDroplet( Location initialLocation, DrawingCanvas canvas, FilledRect aCollector ) {
        dropletGraphic = new FilledOval( initialLocation, SIZE, SIZE, canvas );
        collector = aCollector;
        start();
    }
    public void run() {
        while (dropletGraphic.getY() < collector.getY()) {
            dropletGraphic.move(0, Y_STEP);
            pause(DELAY_TIME);
        }
        dropletGraphic.removeFromCanvas();
        if (collector.getY() > 0) {
            collector.setHeight(collector.getHeight() + SIZE/4);
            collector.move(0, -SIZE/4);
        }
    }
}
```

```java
public class Credit extends ActiveObject {
    private Text roleAndName;
    public Credit( String scrollingLine, DrawingCanvas canvas ) {
        roleAndName = new Text( scrollingLine, 0, canvas.getHeight(), canvas );
        roleAndName.move( (canvas.getWidth() – roleAndName.getWidth())/2, 0 );
        start();
    }
    public void run() {
        while( roleAndName.getY() > -roleAndName.getHeight() ) {
            roleAndName.move(0, -Y_STEP);
            pause( DELAY_TIME );
        }
        roleAndName.removeFromCanvas();
    }
}
```

```java
public class Credit extends ActiveObject {
    private Text roleAndName;
    public Credit( String scrollingLine, DrawingCanvas canvas ) {
        roleAndName = new Text( scrollingLine, 0, canvas.getHeight(), canvas );
        roleAndName.move( (canvas.getWidth() – roleAndName.getWidth())/2, 0 );
        start();
    }
    public void run() {
        while( roleAndName.getY() > -roleAndName.getHeight() ) {
            roleAndName.move(0, -Y_STEP);
            pause( DELAY_TIME );
        }
        roleAndName.removeFromCanvas();
    }
```
Final Notes

- Smoothness of animation depends on:
  - The delay between movements
  - The size of each movement