PREREQUISITE

- Loops Advanced
Topics

- Animation
SHUTTLE ANIMATION
**Animate From Origin Semantics**

- Basic Idea: Should animate from origin to current location
- Should move to origin
- Should animate first in Y direction to current Y
- Next should animate in X direction to current X
ANIMATING TO A DESTINATION LOCATION

1. Should move some distance (< total_distance) in straight line to destination
2. Go back to 1 if location not reached
ANIMATING TO A DESTINATION LOCATION

1. Should move a constant distance (< total_distance) in straight line to destination
2. Should pause for some time
3. Go back to 1 if location not reached
**Pausing Program**

```c
void sleep (int pauseTime) {
    int numberOfAssignments = pauseTime; //ASSIGNMENT_TIME;

    for (int i = 0; i < numberOfAssignments; i++) {
        int dummy = 0; // nonsense assignment
    }
}
```

- ASSIGNMENT_TIME different on different computers!
- Unnecessarily using the CPU (Busy Waiting)
- Need the OS to suspend to put to sleep program for pause time
**Pausing Program: ThreadSupport.sleep()**

```java
public void sleep(int pauseTime) {
    try {
        // OS suspends program for pauseTime
        Thread.sleep(pauseTime);
    } catch (Exception e) {
        // program may be forcibly interrupted while sleeping
        e.printStackTrace();
    }
}
```
protected void animateYFromOrigin(PlottedShuttle shuttle,
        int animationStep, int animationPauseTime,
        int startY, int endY) {
    // make sure we don't go past final Y position
    while (startY < endY) {
        ThreadSupport.sleep(animationPauseTime);
        startY += animationStep;
        shuttle.setShuttleY(startY);
    }
    // move to destination Y position
    shuttle.setShuttleY(endY);
}
public void animateFromOrigin(PlottedShuttle shuttle, int animationStep, int animationPauseTime) {
    //System.out.println ("Current thread:" + ());
    int originalX = shuttle.getShuttleX();
    int originalY = shuttle.getShuttleY();
    int curX = 0;
    int curY = 0;
    shuttle.setShuttleX(curX);
    shuttle.setShuttleY(curY);
    animateYFromOrigin(shuttle, animationStep,
                        animationPauseTime, curY, originalY);
    animateXFromOrigin(shuttle, animationStep,
                        animationPauseTime, curX, originalX);
}
public static void main(String[] args) {
    PlottedShuttle shuttle = new APlottedShuttle(50, 100);
    OEFrame oeFrame = ObjectEditor.edit(shuttle);
    oeFrame.hideMainPanel();
    oeFrame.setSize(450, 450);
    ShuttleAnimator shuttleAnimator = new AShuttleAnimator();
    shuttleAnimator.animateFromOrigin(aShuttle, 5, 100, anOEFrame);
}
Basic Architecture

Main Class

animate Shuttle()

APlotted Shuttle

setShuttleX(Y)()
protected void animateYFromOrigin(PlottedShuttle shuttle, 
    int animationStep, int animationPauseTime, 
    int startY, int endY) {

    // make sure we don't go past final Y position
    while (startY < endY) {
        ThreadSupport.sleep(animationPauseTime);
        startY += animationStep;
        shuttle.setShuttleY(startY);
    }

    // move to destination Y position
    shuttle.setShuttleY(endY);
}

Shuttle being displayed by some GUI code 
(e.g. ObjectEditor)

GUI needs to be told when shuttle 
coordinates change

Even if the method was called by it and it 
does automatic refresh

Automatic refresh done when the method 
returns, not at indeterminate intermediate 
points
protected void animateYFromOrigin(PlottedShuttle shuttle,
      int animationStep, int animationPauseTime,
      int startY, int endY, OEFrame frame) {

    // make sure we don't go past final Y position
    while (startY < endY) {
        ThreadSupport.sleep(animationPauseTime);
        startY += animationStep;
        shuttle.setShuttleY(startY);
        frame.refresh();
    }

    // move to destination Y position
    shuttle.setShuttleY(endY);
}
public class ShuttleAnimationDriver {
    public static void demoShuttleAnimation(
        ShuttleAnimator aShuttleAnimator,
        PlottedShuttle aShuttle,
        OEFrame anOEFrame)
    {
        aShuttleAnimator.animateFromOrigin(aShuttle, 5, 100);
        aShuttleAnimator.animateFromOrigin(aShuttle, 5, 100, anOEFrame);
    }
    public static void main(String[] args) {
        PlottedShuttle shuttle = new APlottedShuttle(50, 100);
        OEFrame oeFrame = ObjectEditor.edit(shuttle);
        oeFrame.hideMainPanel();
        oeFrame.setSize(450, 450);
        ShuttleAnimator shuttleAnimator = new AShuttleAnimator();
        demoShuttleAnimation(shuttleAnimator, shuttle, oeFrame);
    }
}

Both animateFromOrigin methods take the same amount of time.

The refreshing one changes the display, the other one does not.
REFRESHING ARCHITECTURE

APIlotted Shuttle

setShuttleX(Y)()

repaint()

JPanel

paint()

animate Shuttle()

AShuttleAnimator

animate Shuttle()

Main Class

main
**Problem with Full Refresh**

- The entire UI is refreshed, not just the part that changes
- Multiple GUIs (views) may be displaying the same object
- Should create animated objects as observables
public static void demoShuttleAnimation(
    ShuttleAnimator aShuttleAnimator,
    PlottedShuttle aShuttle,
    OEFrame anOEFrame) {
    aShuttleAnimator.animateFromOrigin(aShuttle, 5, 100);
    aShuttleAnimator.animateFromOrigin(aShuttle, 5, 100, anOEFrame);
}

public class ObservableShuttleAnimationDriver extends ShuttleAnimationDriver {
    public static void main(String[] args) {
        ObservablePlottedShuttle shuttle =
            new AnObservablePlottedShuttle(50, 100);
        OEFrame oeFrame = ObjectEditor.edit(shuttle);
        oeFrame.hideMainPanel();
        oeFrame.setSize(400, 400);
        oeFrame.setLocation(400, 0);
        BeanView view = new APlottedShuttleJPanelAndView(shuttle);
        JFrame frame = new JFrame("Plotted Shuttle");
        frame.add((Component) view);
        frame.setSize(400, 400);
        frame.setVisible(true);
        ShuttleAnimator shuttleAnimator = new AShuttleAnimator();
        demoShuttleAnimation(shuttleAnimator, shuttle, oeFrame);
    }
}
MVC Architecture

- APlottedShuttle
- setShuttleX(Y)()
- repaint()
- JPanel
- paint()
- animateShuttle()
- AShuttleAnimator
- animateShuttle()
- MainClass
- main
ANIMATION AND MVC
public static void main(String[] args) {
    PlottedShuttle shuttle = new APlottedShuttle(50, 100);
    OEFrame oeFrame = ObjectEditor.edit(shuttle);
    oeFrame.hideMainPanel();
    oeFrame.setSize(450, 450);
    ShuttleAnimator shuttleAnimator = new AShuttleAnimator();
    shuttleAnimator.animateFromOrigin(aShuttle, 5, 100);
}

public static void main (String[] args) {
    PlottedShuttle shuttle = new APlottedShuttle(50, 100);
    OEFrame oeFrame = ObjectEditor.edit(shuttle);
    oeFrame.hideMainPanel();
    oeFrame.setSize(450, 450);
    FancyShuttleAnimator shuttleAnimator = new AFancyShuttleAnimator();
    ObjectEditor.edit(shuttleAnimator);
}
public class AFancyShuttleAnimator extends AShuttleAnimator implements FancyShuttleAnimator {

    int animationStep = 5;
    int animationPauseTime = 100;
    PlottedShuttle shuttle;

    public AFancyShuttleAnimator(PlottedShuttle theShuttle) {
        shuttle = theShuttle;
    }

    public int getAnimationStep() {
        return animationStep;
    }

    public void setAnimationStep(int animationStep) {
        this.animationStep = animationStep;
    }

    public int getAnimationPauseTime() {
        return animationPauseTime;
    }

    public void setAnimationPauseTime(int animationPauseTime) {
        this.animationPauseTime = animationPauseTime;
    }

    public void animateShuttle() {
        animateFromOrigin(shuttle, animationStep, animationPauseTime);
    }
}
GUI
VIDEO
UI vs Main Thread

Problem is that the user interface both executes loop and does the redraws the screen

Redrawing of screen done after loop is over

In the main case, the man thread or activity, executes loop

The UI thread or activity redraws the screen
@SeparateThread(true)
public void animateShuttle() {
    animateFromOrigin(shuttle, animationStep, animationPauseTime);
}