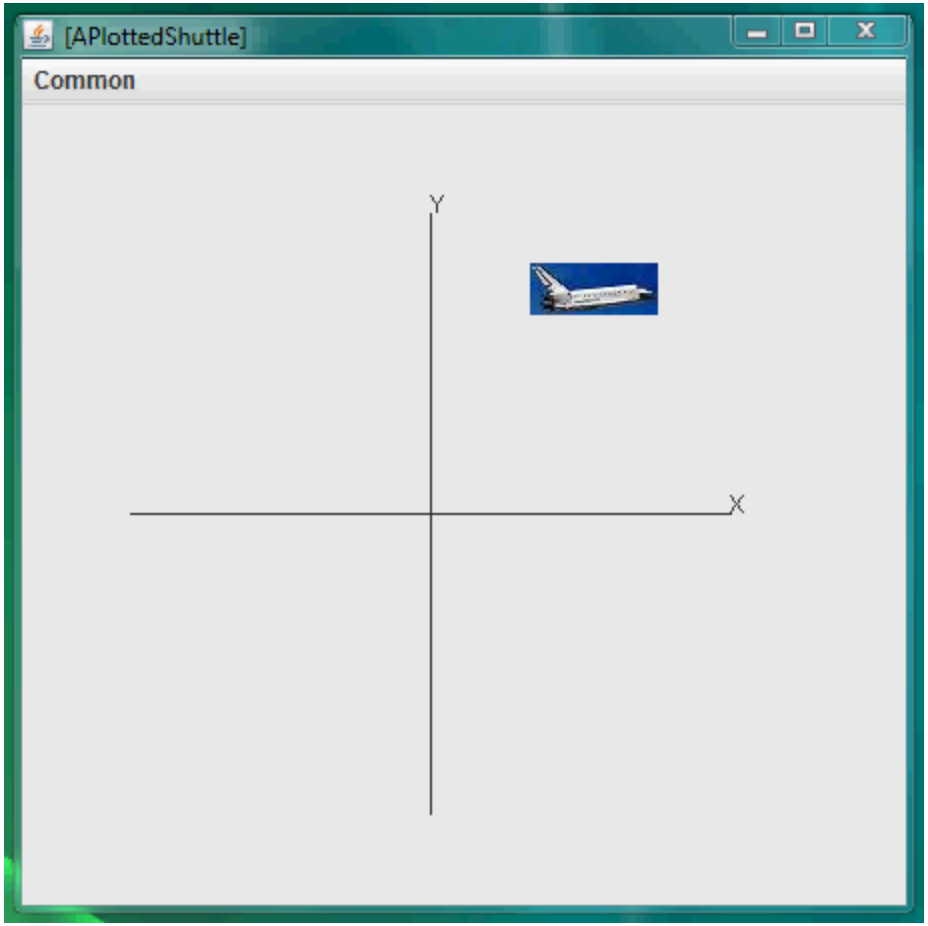
Comp 401
Animation: MVC

Instructor: Prasun Dewan

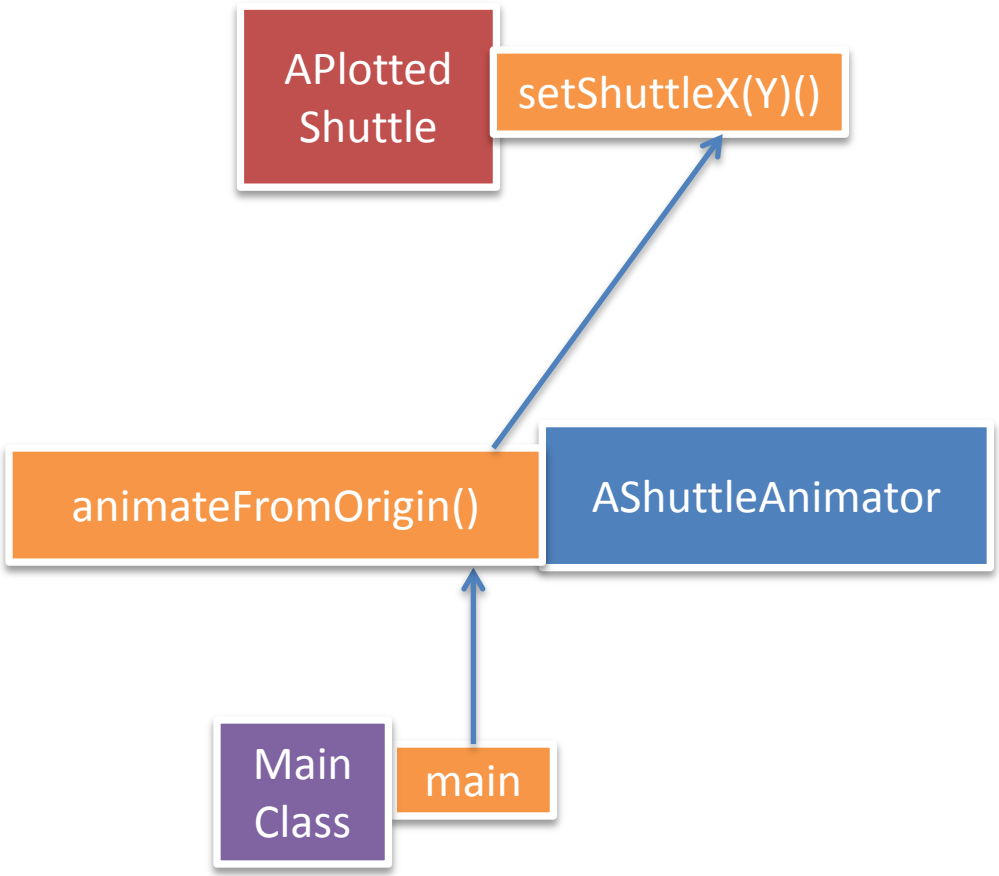
Prerequisite

- Animation: Loops

Shuttle Animation



Basic Architecture



AnimateFromOrigin

```
  public void animateFromOrigin(PlottedShuttle shuttle,
                int animationStep, int animationPauseTime) {
    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);
}
```

Animation in Y Direction

```
  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);
}
```

Main Method

```
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);
}
```

No animation, nothing happens on the screen

Using non observable version of plotted shuttle

Manual Full Refresh

```
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);
}
```

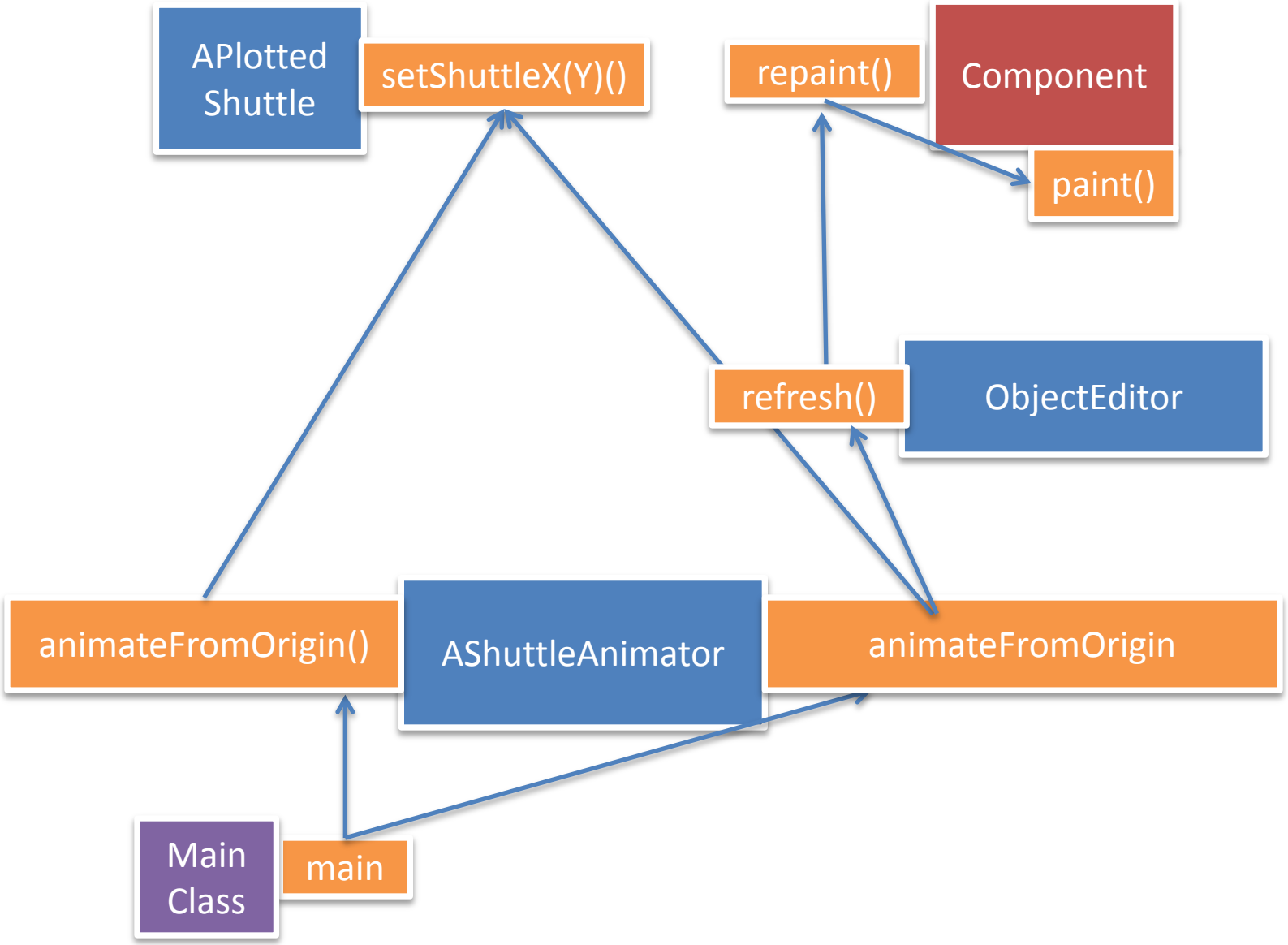
Manual Full Refresh

```
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 animateFromOrigins take the same amount of time

The refreshing one changes the display, the other one does not

Refreshing Architecture



Problem with Full Refresh

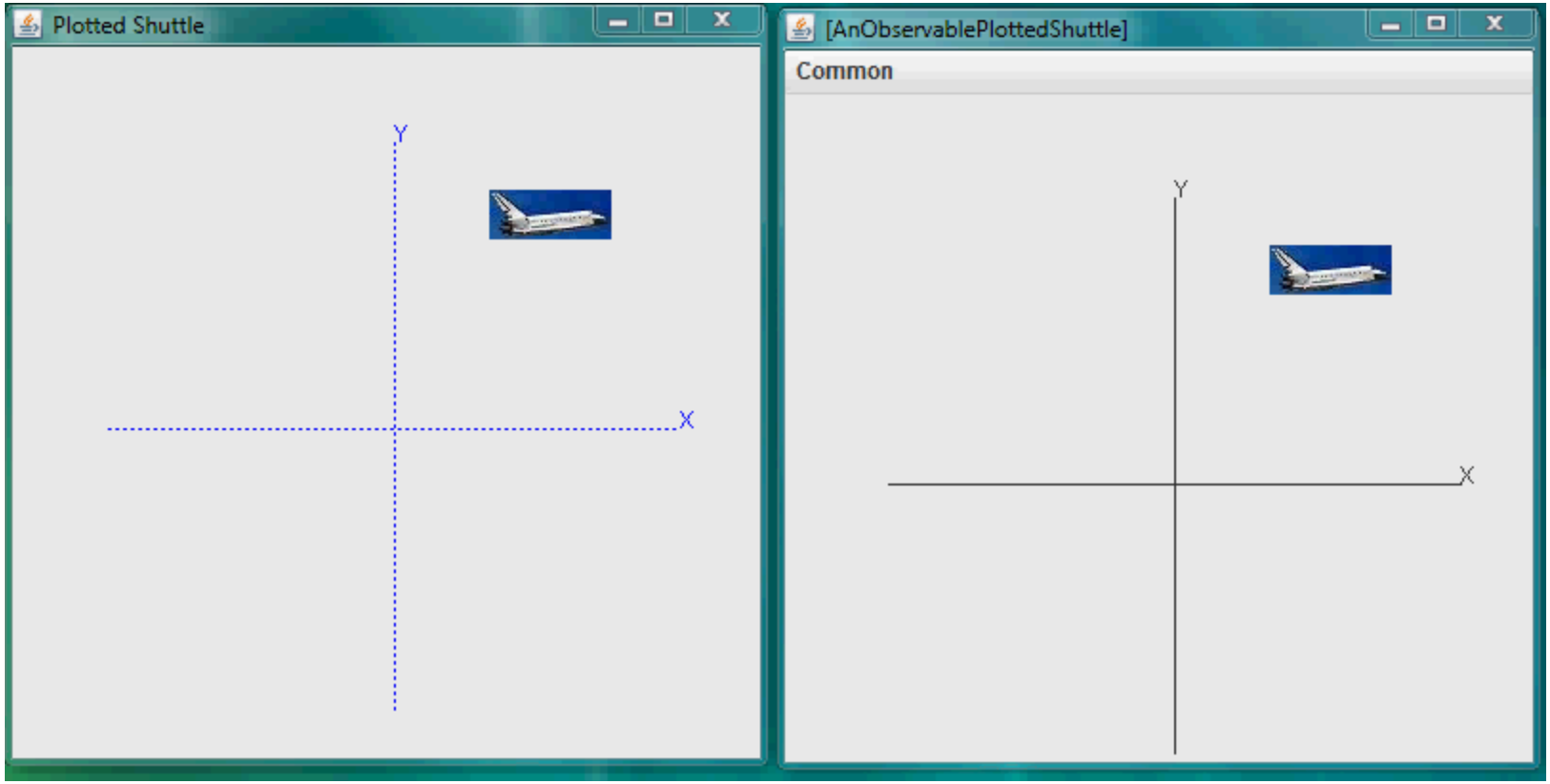
- The entire UI is refreshed, not just the part that changes
- Multiple GUIs (views) may be displaying the same object
- Should create animated objects as observables

Using MVC

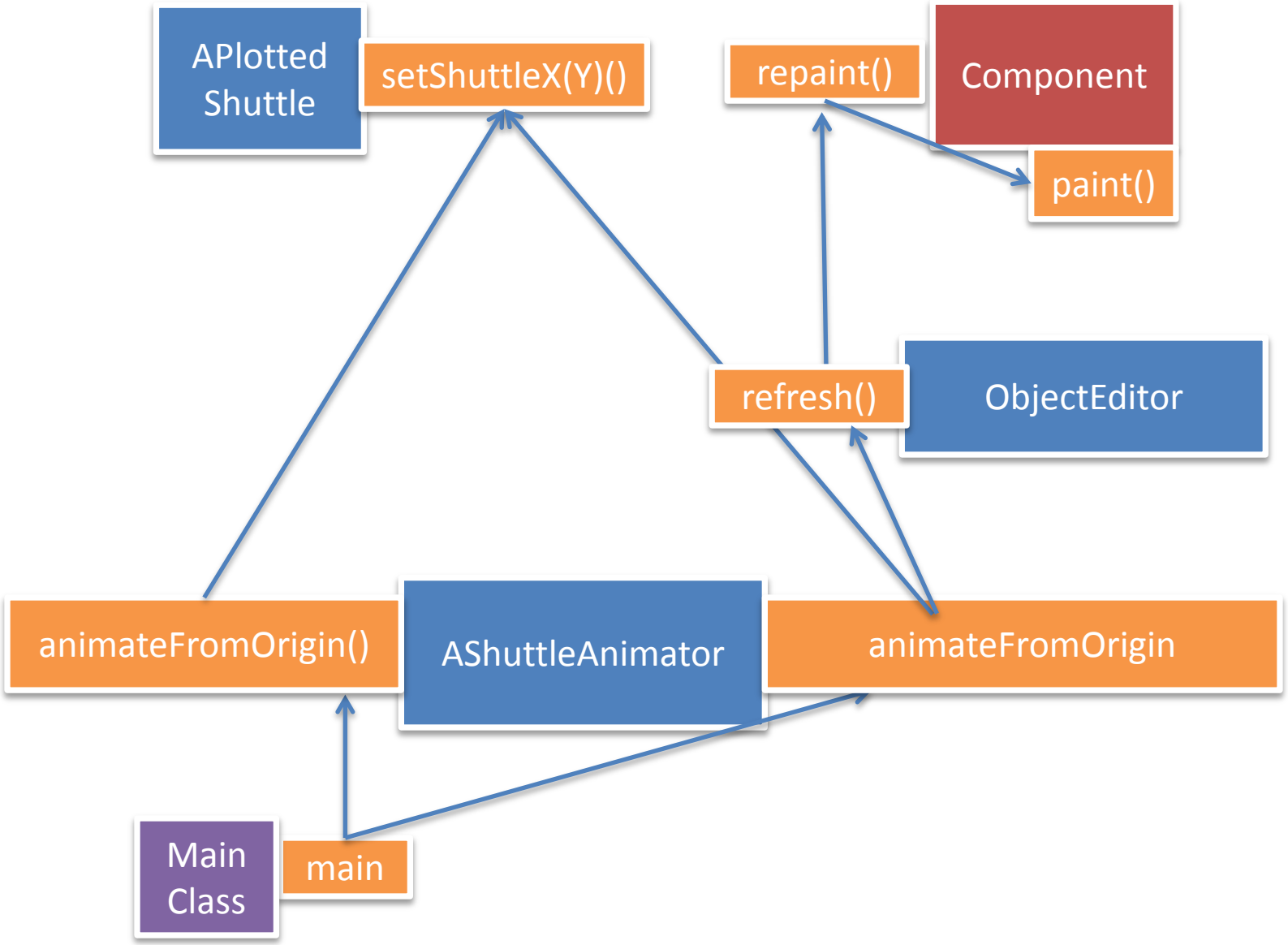
```
  public static void demoShuttleAnimation(
                        ShuttleAnimator aShuttleAnimator,
                        PlottedShuttle aShuttle,
                        OEFrame anOEFrame) {
    aShuttleAnimator.animateFromOrigin(aShuttle, 5, 100);
    aShuttleAnimator.animateFromOrigin(aShuttle, 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);
  PropertyChangeListener view = new APlottedShuttleView(shuttle);
  shuttle.addPropertyChangeListener(view);
  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);
}
```

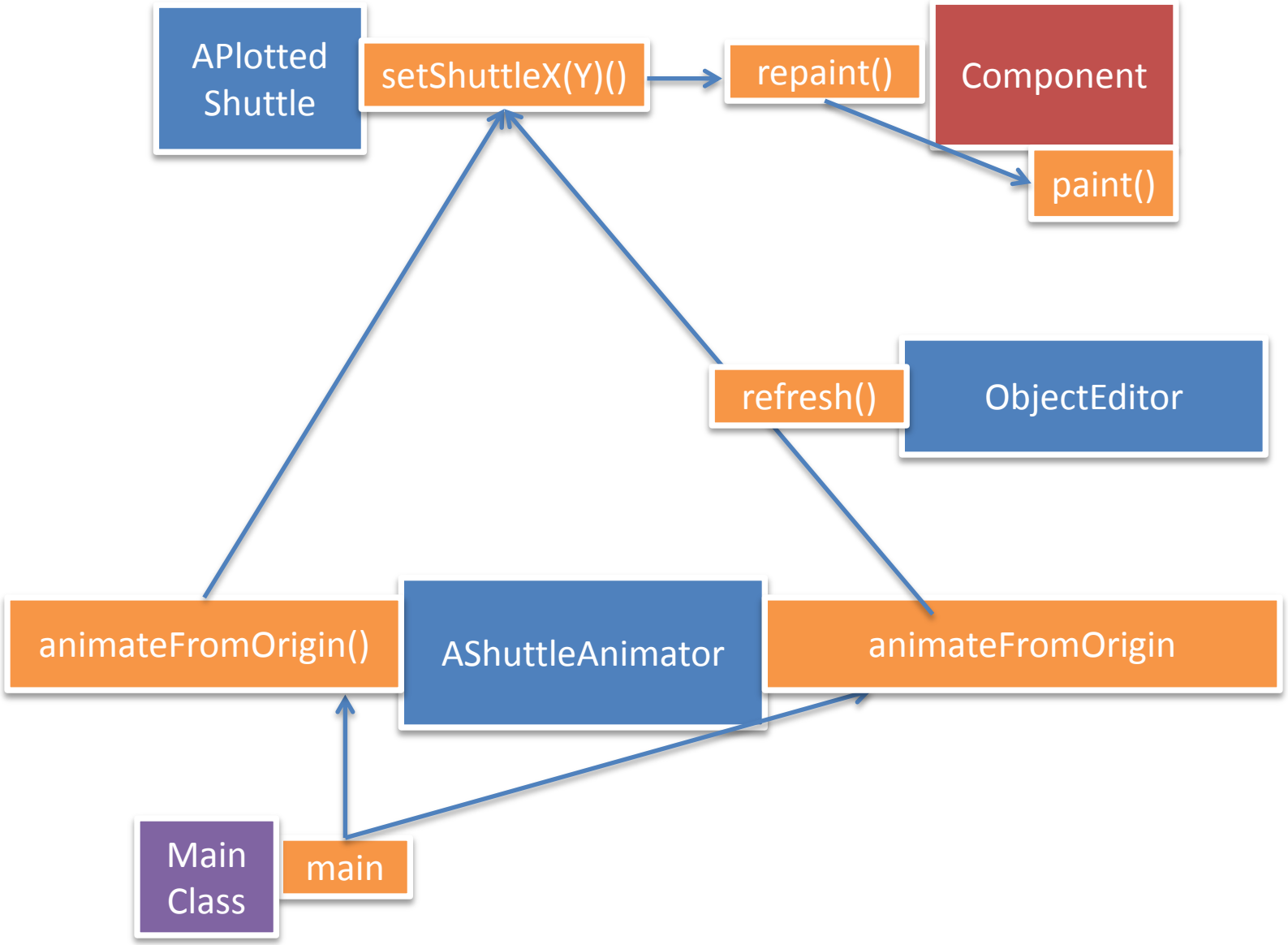
Animation and MVC



Non Observable Architecture



Observable Architecture



Incremental Updates

- After each animation step, all displays of the animation must be updated
- The refresh operation can be used to do ensure these updates are made.
- Better, the observable-observer concept can be used to ensure these updates are made
  - for each graphical property changed by the animation, the class of the property should allow observers to be registered and the setter of the property informs the observers about the update
- ObjectEditor requires the JavaBeans observer-observer approach based around the PropertyChangeListener interface

Extra

Main vs. Interactive Animation

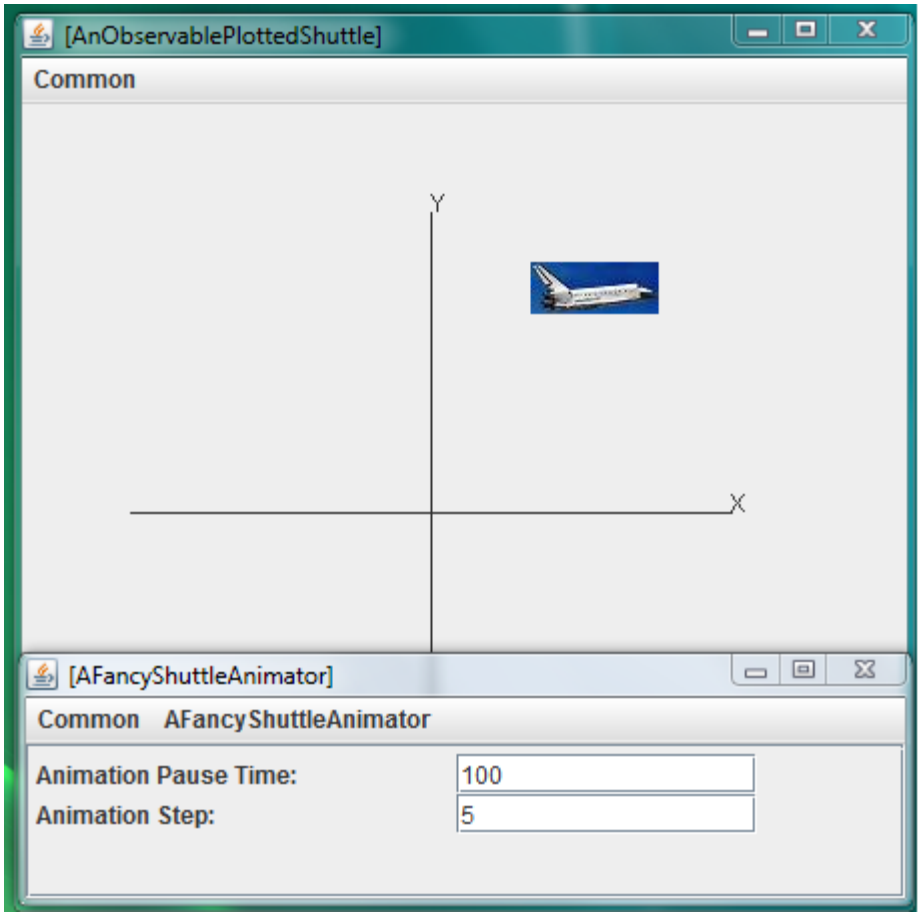
```
public static void main(String[] args) {
  PlottedShuttle shuttle = new APlottedShuttle(50, 100);
  OEFrame oeFrame = ObjectEditor.edit(shuttle);
  oeFrame.hideMainPanel();
  oeFrame.setSize (450, 450);
  ShuttleAnimator shuttleAnimator = new AShuttleAnimator();
  shuttleAnimator.animateFromOrigin(aShuttle, 5, 100);
}
```

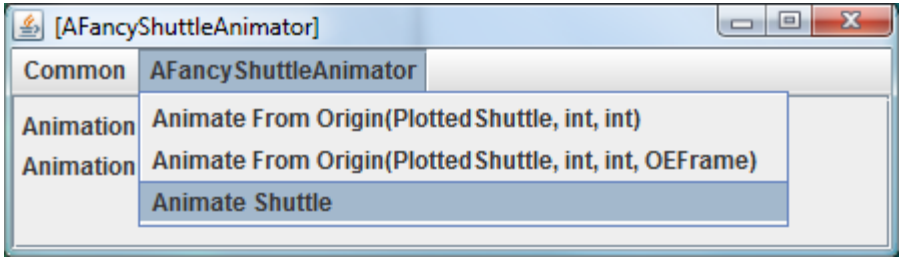
```
public static void main (String[] args) {
  PlottedShuttle shuttle = new APlottedShuttle(50, 100);
  OEFrame oeFrame = ObjectEditor.edit(shuttle);
  oeFrame.hideMainPanel();
  oeFrame.setSize (450, 450);
  FancyShuttleAnimator shuttleAnimator = new AFancyShuttleAnimator();
  ObjectEditor.edit(shuttleAnimator);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

Fancy Animator

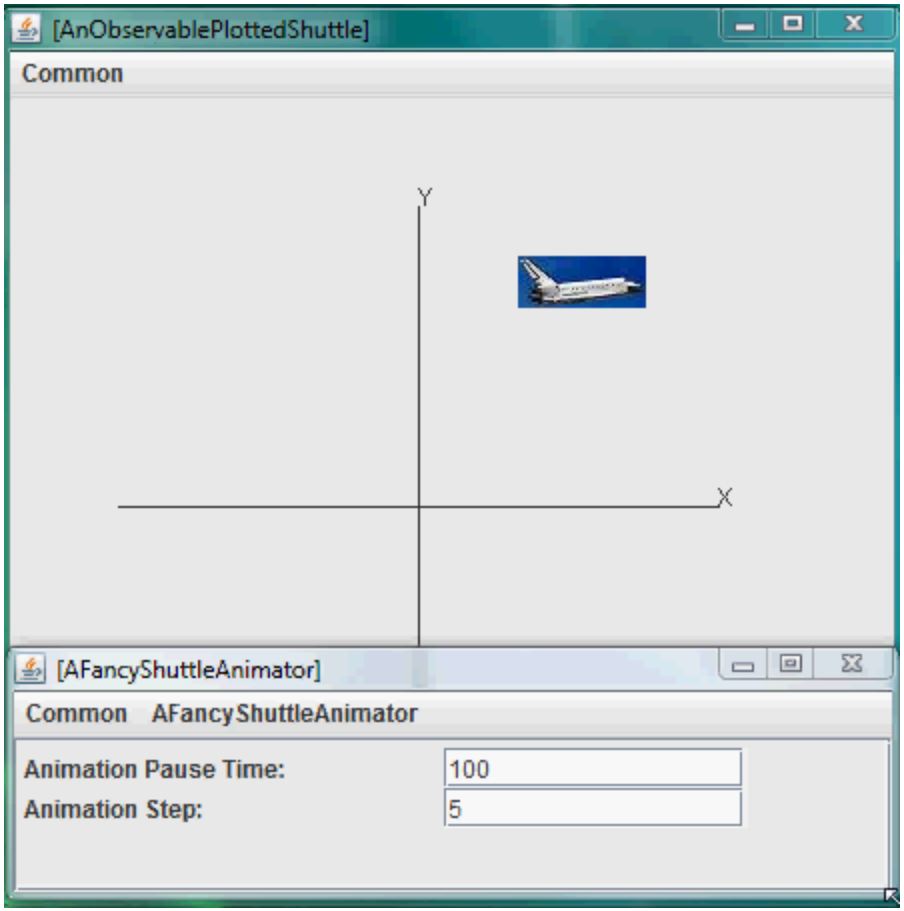
```
public class AFancyShuttleAnimator extends AShuttleAnimator
                                   implements FancyShuttleAnimator {
  int animationStep = 5;
  int animationPauseTime = 100;
  PlottedShuttle shuttle;
  public AFancyShuttleAnimator(PlottedShuttle theShuttle) {
    shuttle = theShuttle;
  }
  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


[AnObservablePlottedShuttle]
Common
Y
X
[AFancyShuttleAnimator]
Common
AFancyShuttleAnimator
Animation Pause Time:
100
Animation Step:
5


[AFancyShuttleAnimator]
Common
AFancyShuttleAnimator
Animation
Animation
Animate From Origin(PlottedShuttle, int, int)
Animate From Origin(PlottedShuttle, int, int, OEFrame)
Animate Shuttle

Video


[AnObservablePlottedShuttle]
Common
Y
X
[AFancyShuttleAnimator]
Common
AFancyShuttleAnimator
Animation Pause Time:
100
Animation Step:
5

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

Asking ObjectEditor to Interactive Call in Separate Thread

```
@SeparateThread(true)
 public void animateShuttle() {
   animateFromOrigin(shuttle, animationStep, animationPauseTime);
 }
```