COMP 401
USER-INTERFACE VS. MAIN THREADS

Instructor: Prasun Dewan
**Prerequisite**

- Animation Threads Commands
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);
}
Animations from Main

```java
public static void main(String[] args) {
    PlottedShuttle shuttle1 = new AnObservablePlottedShuttle(50, 100);
    OEFrame oeFrame1 = ObjectEditor.edit(shuttle1);
    oeFrame1.hideMainPanel();
    oeFrame1.setLocation(0, 0);
    oeFrame1.setSize(400, 400);
    PlottedShuttle shuttle2 = new AnObservablePlottedShuttle(100, 50);
    OEFrame oeFrame2 = ObjectEditor.edit(shuttle2);
    oeFrame2.hideMainPanel();
    oeFrame2.setLocation(400, 0);
    oeFrame2.setSize(400, 400);
    ShuttleAnimator shuttleAnimator1 = new AShuttleAnimator();
    ShuttleAnimator shuttleAnimator2 = new AShuttleAnimator();
    concurrentDemoShuttleAnimation(shuttleAnimator1, shuttle1);
    concurrentDemoShuttleAnimation(shuttleAnimator2, shuttle2);
}
```

Threads created, as multiple independent animations wanted
**SINGLE ANIMATION FROM MAIN: NO SPECIAL THREAD**

```
// APlottedShuttle

setShuttleX(\(Y\))(\)

// repaint()

// JPanel

paint()```

```
// AShuttleAnimator

animateShuttle()```

```
// Main Class

main```

```
// Main Thread```

```
Consider Single Animation

```java
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);
}
```

Start animation from the user interface?

Extension of ShuttleAnimator that allows parameters to be properties

We can edit these properties interactively and start animation with them as parameters
GUI
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);
    }
}
VIDEO
**When does Main Terminate?**

```java
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);
}
```

**Main thread executes loop**

```
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);
}
```

**Main thread terminates**

UI Thread (created by Java) executes loop
CONSIDER SINGLE ANIMATION

ShuttleAnimationDriver [Java Application]
   lectures.animation.loops.ShuttleAnimationDriver at localhost:58863
   Thread [main] (Suspended (breakpoint at line 16 in ASHuttleAnimator)) (Running)
   AShuttleAnimator.animateFromOrigin(PlottedS)
   ShuttleAnimationDriver.main(String[]) line: 20
   Thread [AWT-Shutdown] (Running)
   Daemon Thread [AWT-Windows] (Running)
   Thread [AWT-EventQueue-0] (Running)
   Thread [Tool Tip Thread] (Running)

AFancyShuttleAnimator [Java Application]
   lectures.animation.threads.AFancyShuttleAnimator at localhost:58579
   Thread [AWT-Shutdown] (Running)
   Daemon Thread [AWT-Windows] (Running)
   Thread [AWT-EventQueue-0] (Suspended (breakpoint at line 32 in AFancyShuttleAnimator)) (Running)
   AFancyShuttleAnimator(ASHuttleAnimator).animate() line: 32
   AFancyShuttleAnimator.animateShuttle() line: 32

Main thread executes loop
Main thread starts UI and terminates
UI Thread (created by Java) executes loop
**Interactive Animation: No special Thread**

- **AFancy ShuttleAnimator**
  - animate FromOrigin()
  - animate Shuttle()
- **APIlotted Shuttle**
  - setShuttleX(Y)()
- **JPanel**
  - repaint()
  - paint()
- **AWT Thread**
  - setVisible()
  - setVisible()
- **Main Class**
  - main
INTERACTIVE ANIMATION: NO SPECIAL THREAD (REVIEW)

- APlotted Shuttle
  - setShuttleX(Y)()
  - repaint()
- JPanel
  - paint()
- AWT Thread
  - setVisible()
  - setVisible()
  - setVisible()
- JFrame
- JFrame
- JFrame
- Main Class
  - main
- Main Thread
- AFancy ShuttleAnimator
  - animate FromOrigin()
  - animate Shuttle()
SINGLE ANIMATION FROM MAIN: NO SPECIAL THREAD

- APlotted Shuttle
  - setShuttleX(Y)()
  - repaint()

- AShuttleAnimator
  - animate FromOrigin()
  - animate Shuttle()

- JPanel
  - paint()

- AWT Thread
  - setVisible()

- JFrame

- Main Class
  - main

- Main Thread
while (true){
    //wait for and process paint,
    // menu and other events
    waitForAndProcessNextQueuedUIEvent();
}
LOOP EXECUTES

while (true) {
    //wait for and process paint,
    // menu and other events
    waitForAndProcessNextQueuedUIEvent();
}

while (curY < originalY) {
    ThreadSupport.sleep(
        animationPauseTime);
    curY += animationStep;
    shuttle.setShuttleY(curY);
}
while (true) {
    //wait for and process paint,  
    // menu and other events    
    waitForAndProcessNextQueuedUIEvent();
}

while (curY < originalY) {
    ThreadSupport.sleep(
        animationPauseTime);
    curY += animationStep;
    shuttle.setShuttleY(curY);
}

//in Java component showing   
//the shuttle   
public void repaint() {
    enqueueRepaintEvent(this);
}
**LOOP RE-EXECUTES**

```java
while (true) {
    // wait for and process paint,
    // menu and other events
    waitForAndProcessNextQueuedUIEvent();
}

while (curY < originalY) {
    ThreadSupport.sleep(
        animationPauseTime);
    curY += animationStep;
    shuttle.setShuttleY(curY);
}
```
while (true) {
    // wait for and process paint,
    // menu and other events
    waitForAndProcessNextQueuedUIEvent();
}

while (curY < originalY) {
    ThreadSupport.sleep(animationPauseTime);
    curY += animationStep;
    shuttle.setShuttleY(curY);
}

// in Java component showing
// the shuttle
public void repaint() {
    enqueueRepaintEvent(this);
}
while (true)
{
    // wait for and process paint, menu and other events
    processNextQueuedUIEvent();
}
while (true) {
    //wait for and process paint,
    // menu and other events
    processNextQueuedUIEvent();
}

public void paint(Graphics g) {
    //draw shuttle
}
UI THREAD WAITS FOR NEXT EVENT

```java
while (true) {
    // wait for and process paint,
    // menu and other events
    waitForAndProcessNextQueuedUIEvent();
}
```
UI Event Loop and Animations

while (true) {
    // wait for and process paint, menu and other events
    waitForAndProcessNextQueuedUIEvent();
}

while (curY < originalY) {
    ThreadSupport.sleep(animationPauseTime);
    curY += animationStep;
    shuttle.setShuttleY(curY);
}

New UI event not processed until listeners for previous event finish

Animating listener should create new thread for animation code
INTERACTIVE ANIMATION: SPECIAL THREAD

- APlotted Shuttle
  - setShuttleX(Y)()
  - repaint()

- JPanel
  - paint()

- AConcurrentFancy ShuttleAnimator
  - animate FromOrigin()
  - run()

- Shuttle Animation Thread
  - animate Shuttle()

- AShuttleAnimation Command
  - run()

- AWT Thread
  - setVisible()
  - setVisible()
public class AConcurrentShuttleAnimator extends AFancyShuttleAnimator {
    public AConcurrentShuttleAnimator(PlottedShuttle theShuttle) {
        super(theShuttle);
    }
    public void animateShuttle() {
        Thread thread =
        new Thread((
            (new AShuttleAnimationCommand(
                this, shuttle, animationStep,
                animationPauseTime)));
        thread.start();
    }
}
```java
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 AConcurrentShuttleAnimator();
    ObjectEditor.edit(shuttleAnimator);
}
```
The animation method is synchronized
GUI Processing

- Even if main thread terminates, the application continues to run as long as a GUI has been created, which creates the GUI thread.
- A single GUI thread is created for processing the controller (menu/button/... processing) and view (repaint) actions of all models.
- View updates cannot occur until controller returns.
- Controller action should result in a new thread if it starts an animation.
- If a single animation is started from main then no thread needs to be created a main thread executes loop and separate GUI thread updates view.
Repaint and UI Thread Semantics

- When a thread (GUI or some other) calls repaint on a component C
  - It puts a repaint event for C in the GUI event queue if such an event is not already in the queue.
  - This queue contains all UI events such as mouse and key clicks.

- The GUI thread performs the following loop
  1. Waits for the GUI event queue to be non empty
  2. Removes and services the next event from the queue
     - If the next event is a repaint event for component C, it calls the update() method in C, passing it a graphics object. The update() method clears the component and calls paint() method.
     - If the next event is not a repaint() event, then it does some event-specific processing such as calling a listener for the event.
  3. Goes back to 1.