CS324e - Elements of Graphics and Visualization

Intro to Animation
Animation

- **definition:** time-based alteration of graphical objects through different states, locations, sizes, and orientations
  - FRC chapter 12
- **alteration:** change the way we are drawing objects
  - all the stuff from graphics basics
- **time-based:** define how objects change *over time* and render objects based on this as time goes by
Make the Ship Move
Frame

• Frame: one still drawing
• illusion of motion achieved by drawing multiple frames with slight differences in how graphical object is rendered
  – a series of still drawings shown quickly
  – like a flip book
Time-Based vs. Frame-Based

• We will do time based animation
  – amount of movement or alteration based on how much time has passed
• In framed based animation the amount of movement or alteration is defined per frame (instead of per unit time)
• Frame rate: number of frames drawn per unit time, usually per second
• Frame based relies on strictly controlling the frame rate
Controlling Frame Rate

• Difficult based on
  – speed of system (which varies between systems your program will run on)
  – complexity of what is being rendered
Series of Attempts at Animation

• Attempt 1:

```java
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D)g;
    g2.drawImage(ufo, 20, 20, null);
    g2.setColor(getBackground());
    g2.fillRect(20, 20, ufo.getWidth(), ufo.getHeight());
    g2.drawImage(ufo, getWidth() - ufo.getWidth() - 20, 20, null);
}
```

• What is result when run?
• Why?
Problems

• Teleportation != Animation
• Too fast
• Swing Buffering
• Motion not time based
Second Attempt

```java
// attempt at animation 2 - slow it down?
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D)g;

    g2.drawImage(ufo, 20, 20, null);

    int x = 0;
    for (int i = 0; i < 1000000000; i++)
        for (int j = 0; j < 10000000; j++)
            x = i * j;
    System.out.println(x);

    g2.setColor(getBackground());
    g2.fillRect(20, 20, ufo.getWidth(), ufo.getHeight());

    g2.drawImage(ufo, getWidth() - ufo.getWidth() - 20, 20, null);
}
```

• What is wrong with this attempt?
Third Attempt

• Remember, swing uses a back buffer
• All the drawing done to the back buffer (essentially a buffered image) and when it is done the result is displayed
• All drawing from paintComponent appears at once
  — recall the random art assignment
  — didn't see a few pixels at a time did we?
Third Attempt

```java
// Animation Frame class
public void start() {
    setVisible(true);
    int x = 0;
    for (int i = 0; i < 1000000000; i++)
        for (int j = 0; j < 1000000; j++)
            x = i * j;
    System.out.println(x);
    thePanel.changeShip();
    repaint();
}
```

```java
public void changeShip() {
    xImg = getWidth() - ufo.getWidth() - 20;
}
```

```java
// attempt 3 - pauses elsewhere
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g;

    g2.drawImage(ufo, xImg, yImg, null);
}
```
Teleportation != Animation

• Must change the position of the ship a little bit at a time
• Change the x position of the ship a little bit at a time
• Eventually alter y as well
Fourth Attempt

```java
// Animation Panel class
public void moveShip() {
    xImg = yImg = prevXImg = prevYImg = 20;
    int endX = getWidth() - ufo.getWidth() - 20;
    for(int x = xImg; x < endX; x++) {
        repaint();
        // some time passes
        xImg += 1;
    }
}
```

```java
// attempt 4 - pauses elsewhere
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D)g;
    g2.drawImage(ufo, xImg, yImg, null);
}
```

• What happens?
Too Fast

• Need to allow time to pass between calls to repaint

• Kludge:

```java
// Animation Panel class
public void moveShip() {
    xImg = yImg = prevXImg = prevYImg = 20;
    int endX = getWidth() - ufo.getWidth() - 20;
    for(int x = xImg; x < endX; x++) {
        repaint();
    }
    int temp = 0;
    for(int i = 0; i < 10000000; i++)
        for(int j = 0; j < 1000000; j++)
            temp = i * j;
    xImg += 1;
}
```
Movement Achieved

• The ship appears to move, but the approach couldn't be worse
• Must not burn cycles to allow the passage of time
• run on a different machine?
• Also, motion is frame based not time based
  — moving one pixel at a time
Time Based Motion

• define a speed for the object / motion and update \( x \) (and \( y \)) based on how much time has passed

• \( x = x_0 + t \times (x_1 - x_0) \)
  
• \( x \) current position of ship
  
• \( x_0 \) start position of ship
  
• \( x_1 \) ending value of ship
  
• \( t = \text{fraction of time elapsed, from 0 to 1} \)
Time Based Motion

```java
// Animation Panel class
public void moveShip() {
    xImg = yImg = prevXImg = prevYImg = 20;
    int startX = xImg;
    int endX = getWidth() - ufo.getWidth() - 20;
    long animationDuration = 4500; // milliseconds
    long startTime = getTime(); // getTime doesn't exist
    long currentTime = startTime;
    long endTime = startTime + animationDuration;
    while (currentTime < endTime) {
        long elapsedTime = currentTime - startTime;
        float t = ((float) elapsedTime) / animationDuration;
        xImg = (int) (startX + t * (endX - startX));
        repaint();

        // make some time pass

        currentTime = getTime();
    }
}
```
What's The Time

• Two ways of getting system time in Java
  – the time the computer thinks it is
• System.currentTimeMillis()
• System.nanoTime()
Aside - From Last Time

• The uncanny valley
  —Masahiro Mori
System.currentTimeMillis()

• the number of milliseconds (thousandths of a second) that have passed since January 1, 1970

• arbitrary date and time know as the Unix Epoch

• useful for determining how much time has passed between events in the program

• measurements often limited to 10s of milliseconds
System.nanoTime()

- based on an arbitrary point in time
  - don't assume Unix Epoch
  - may even be in future
- only used for elapsed time
- smaller granularity
  - billionths of a second
- better resolution than System.currentTimeMillis()
private float getFPS() {
    numFrames++;
    if(startTime == 0) {
        startTime = System.nanoTime();
    } else {
        long currentTime = System.nanoTime();
        long delta = (currentTime - startTime);
        // Average the fps every five seconds
        if(delta > FPS_WINDOW) {
            fps = ((float) numFrames) / delta * BILLION;
            numFrames = 0;
            startTime = currentTime;
            System.out.println(fps);
        }
    }
    return fps;
}
Move Ship Back and Forth

- x coordinate (xImg) changed to double

```java
// back and forth
public void moveShip() {
    xImg = yImg = prevXImg = prevYImg = 20;
    double startX = xImg;
    int endX = getWidth() - ufo.getWidth() - 20;
    long previousTime = System.nanoTime();
    int speed = 100; // pixels per second
    while(true) {
        long currentTime = System.nanoTime();
        long elapsedTime = currentTime - previousTime;
        xImg = xImg + 1.0 * elapsedTime / BILLION * speed;
        System.out.println(xImg + " " + previousTime
            + " " + currentTime + " " + elapsedTime);
        previousTime = currentTime;
        if(xImg > endX || xImg < startX)
            speed = speed * -1;
        repaint();
        int x = 0;
        for(int i = 0; i < 1000; i++)
            for(int j = 0; j < 10000; j++)
                x = i * j;
    }
```
Pausing

- The delay loop is a horrible kludge
- First option: pause the thread of execution using Thread.sleep() method
- Thread making call is paused by system
- Doesn't do any work, but doesn't burn CPU cycles either
- argument is milliseconds to sleep
Thread.sleep(int millis)

- call pause method from moveShip
- DELAY set to 30 milliseconds

```java
private void pause() {
    try {
        Thread.sleep(Delay);
    }
    catch (InterruptedException e) {
        System.out.println(e);
    }
}
```

- Compare two versions of pausing
Problems With Sleeping

• Thread.sleep() causes the whole thread (program) to stop
• What if we have a lot of computations to do?
• Imagine the random art program
• what if we wanted to "animate" the drawing of the art by showing a few hundred pixels at a time?
• Does Thread.sleep help?
Timers

- To get repeated notifications that some time has passed without putting the whole thread to sleep
- "Timers allow the program to perform repetitive operations at regular time intervals in a way that allows other work to happen asynchronously."
  — FRC
Timer Classes

• java.util.Timer
  – general purpose timer class
• Creates a separate thread of execution (your program forks)
• schedule TimerTasks with a run() method that is called by the Timer
• fixed delay times (adjusts on fly) or fixed rate times (doesn't adjust
javax.swing.Timer

• Create a time and it will make callbacks
• much like our action listeners for buttons and mouse listeners
• Create a time and then create listeners for when the timer goes off
• `javax.swing.Timer` specifically for Swing applications / GUIs
  — the callbacks are to the Swing Event Dispatch Thread
Fixed Delay Timing

• Timer will adjust delay times to meet desired wakeup call interval

• Events are coalesced:
  – if it gets to far behind some timing events are simply discarded
  – repaint does the same thing
Comparison of Timers

• SwingTimerDemo creates two swing timers
  – one using fixed delay (default)
  – one using fixed rate (events not coalesced)
Swing Fixed Delay

```java
// Run a default fixed-delay timer
Timer timer = new Timer(DELAY, new SwingTimerDemo());
startTime = prevTime = System.currentTimeMillis();
System.out.println("Fixed Delay Times");
timer.start();
```

- **results:**
  - Fixed Delay Times
  - Elapsed time = 134
  - Elapsed time = 270
  - Elapsed time = 100
  - Elapsed time = 100
  - Elapsed time = 100
  - Elapsed time = 100
  - Elapsed time = 100
  - Elapsed time = 100
  - Elapsed time = 100
  - Elapsed time = 100
  - Elapsed time = 100
  - Elapsed time = 100
Swing Fixed Rate

```java
// Run a timer with no coalescing to get fixed-rate behavior
Timer timer = new Timer(Delay, new SwingTimerDemo());
startTime = prevTime = System.currentTimeMillis();
timer.setCoalesce(false);
System.out.println("\nFixed Rate Times");
timer.start();
```

• result

Fixed Rate Times
Elapsed time = 0
Elapsed time = 100
Elapsed time = 30
Elapsed time = 200
Elapsed time = 30
Elapsed time = 30
Elapsed time = 30
Elapsed time = 30
Elapsed time = 30
Elapsed time = 50
Elapsed time = 30
Using Timer in UFO Program

• Loop no longer in moveShip
• constructor for AnimationPanel

```java
public AnimationPanel() {
    this.setPreferredSize(new Dimension(WIDTH, HEIGHT));
    loadImage();
    setBackground(Color.WHITE);
    xImg = startX = 20;
    yImg = 20;
    endX = 800 - ufo.getWidth() - 20;
    System.out.println(startX + " " + endX);
    addTimer();
}
```
AnimationPanel With Timer

• Create timer
• ActionListener is an annoynomoyus inner class that calls update method on the panel

```java
private void addTimer() {
    timer = new Timer(30, new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            // System.out.println(e);
            update();
        }
    });
}
```
AnimationPanel With Timer

• start method to begin animation

```java
public void start() {
    previousTime = System.nanoTime();
    speed = 200;
    timer.start();
}
```

• update method called when timer goes off

```java
private void update() {
    moveShip();
    repaint();
}
```
AnimationPanel With Timer

- moveShip
- no loop
- must make many variables instance variables - (what happens if speed local?)

```java
// responding to a timer going off
public void moveShip() {
    long currentTime = System.nanoTime();
    long elapsedTime = currentTime - previousTime;
    xImg = xImg + 1.0 * elapsedTime / BILLION * speed;
    //
    System.out.println(xImg + " " + previousTime + " " + currentTime + " " + elapsedTime);
    previousTime = currentTime;
    if(xImg > endX || xImg < startX)
        speed = speed * -1;
}
```
What's Next?

• Clearly the logic for the ship does not belong in the AnimationPanel class
• Create a Ship class that contains logic for moving ship
• Move ship in something other than a straight line
• animate ship in another way (shrink, fade out)