CSE1720

Week 05, Lecture 09

Winter 2014 ◆ Tuesday, Feb 4, 2014

YORK UNIVERSITÉ UNIVERSITY

Objectives for this class meeting

- Cover 2D Graphics topic: displaying images
- `Timer` class, a basic `ActionListener`

YORK UNIVERSITÉ UNIVERSITY

Big picture recap…

- via the services of the `Picture` class
  - our app asks the window manager for a window
  - the constructor creates and places a blank "canvas" inside this window
  - this canvas has an associated `Graphics2D` object, which we can access via `createGraphics()`
- via the services of the `Graphics2D` class
  - we use the services of this class to modify the *current settings*
  - we use the services of this component to perform drawing of shape primitives and text

The VM and the window manager **coordinate** in order to do the drawing

3

YORK UNIVERSITÉ UNIVERSITY

Shooter Games…

- Shooting is a basic behaviour that is a defining characteristic of shooter games
- We will employ encapsulation:
  - encapsulate the shooter
  - encapsulate the projectile
  - encapsulate the target
- Shooting entails:
  - waiting for user input
  - rendering the trajectory over a sequence of frames
  - collision behaviours

4

YORK UNIVERSITÉ UNIVERSITY

Frame Drawing…

- We need functionality to implement repeated frame drawing
- frames need to be drawn whether the user is performing actions or not
- we need a service that will **dispatch events** repeatedly, each of which signals “draw new frame”

5

YORK UNIVERSITÉ UNIVERSITY

Suspension of Disbelief

- Humans will suspend judgment about the implausibility of a narrative in order to engage with the material
  - this alternative is that human does not suspend judgment, disengages and rejects premise of the material
- Video games require suspension of disbelief
  - game mechanics are unrealistic (by design or by technological limitation)
- Designs seek to support suspension of disbelief and try to remove aspects that interfere with suspension of disbelief
  - things that interfere: high frame rate, uncanny valley

6

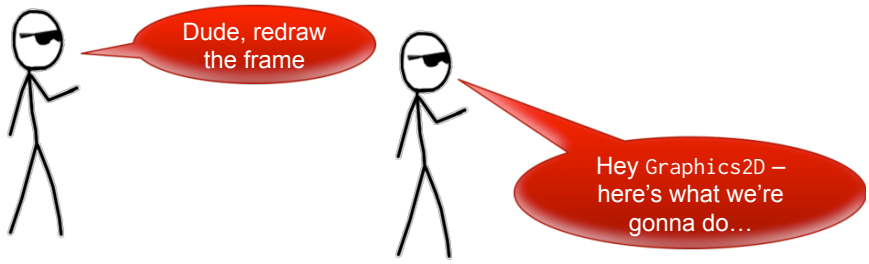
YORK UNIVERSITÉ UNIVERSITY

Frame Rate

- The speed at which frames are drawn is called the **frame rate**
  - **TV: 60 fps**
  - **Movies: 24 fps**
  - **The Hobbit 3D experiment: 48 fps**
  - **Lower threshold : 10-12 fps, phi phenomenon**
  - **Upper threshold : ~66 fps**
- A key concept: **suspension of disbelief**
  - humans will suspend judgment about the implausibility of a narrative in certain conditions, but not others
  - characteristics of the medium can trigger this

YORK UNIVERSITÉ UNIVERSITY

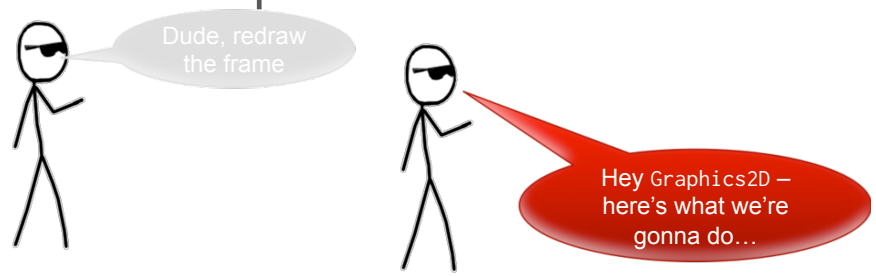
How we implement frames



- We will have two "players" here
  - one that says when it is time to redraw
  - the other says what happens to the screen when it is time to redraw

YORK UNIVERSITÉ UNIVERSITY

How we implement frames



- What happens to the screen when it is time to redraw
  - draw a red dot at the *current point*
  - update the current point to a new position, moving along the diagonal

this code can be found in Labo5

```
graphics.draw(dot);
theCanvas.repaint();
p = new Point(p.x + 1, p.y + 1);
dot = new Ellipse2D.Double(p.x, p.y, DIA, DIA);
```

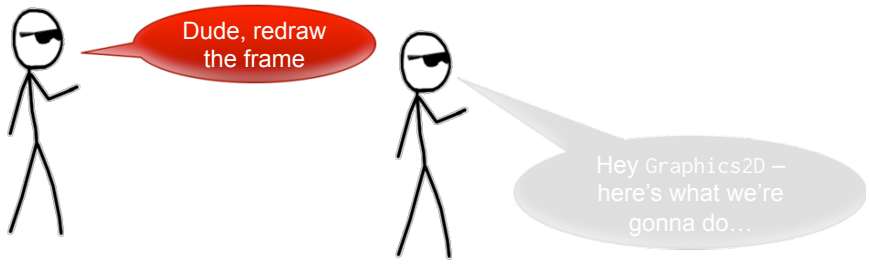

this code can be found in Labo5

```
public class FrameAdvancer implements ActionListener {
        private Picture theCanvas;
        private Graphics2D graphics;
        private Point p;
        private Shape dot;
        private final int DIA = 10;
        private long timeOfInstantiation = System.currentTimeMillis();

        public FrameAdvancer(Picture gameCanvas) {
                this.theCanvas = gameCanvas;
                p = new Point(0, 0);
                dot = new Ellipse2D.Double(p.x, p.y, DIA, DIA);
                graphics = theCanvas.createGraphics();
                graphics.setPaint(Color.RED);
        }

        @Override
        public void actionPerformed(ActionEvent ae) {
                graphics.draw(dot);
                theCanvas.repaint();
                p = new Point(p.x + 1, p.y + 1);
                dot = new Ellipse2D.Double(p.x, p.y, DIA, DIA);
        }
}
```

How we implement frames



- We need functionality to implement repeated frame drawing
  - we need to define the inter-frame interval (msec)
  - we instantiate a `Timer` object
  - this launches a new thread
  - the thread **fires events** at the specified inter-frame interval

YORK UNIVERSITÉ UNIVERSITY

How we implement frames

```
FrameAdvancer frameAdvancer = new FrameAdvancer(pict);


// change this value to indicate the desired number of frames per
second
final int NUM_FRAME_PER_SEC = 30;
// here we determine how much time each frame should spend on-
screen
final int NUM_MSEC_PER_SEC = 1000;
int msecPerFrame = NUM_MSEC_PER_SEC / NUM_FRAME_PER_SEC;

Timer frameAdvancerTimer = new Timer(msecPerFrame, frameAdvancer);
frameAdvancerTimer.start();
```

this code can be found in Labo5

YORK UNIVERSITÉ UNIVERSITY

How to “tune your radio”…

1. Identify the **observee** component
   - this is the component that is dispatching events that you care about
2. Create an **observer** component
   - this will be a component that is capable of “listening” to those types of events
   - this is like “tuning” to the station
3. Use the services of the **observee** to tell it that it has an **observer**

13

YORK UNIVERSITÉ UNIVERSITY

The Observer Pattern

- Create an the “observee” component (the component that will be observed)
  - this is the component that is dispatching events that you care about
  - e.g., the `Timer` object dispatches `ActionEvent` events
- Create an observer component
  - this will be a component that is capable of “listening” to those types of events
  - e.g., the `FrameAdvancer` is an `ActionListener`
- Use the services of the observee to tell it that it has an observer
  - e.g., we provide the `FrameAdvancer` to the constructor of `Timer`

14

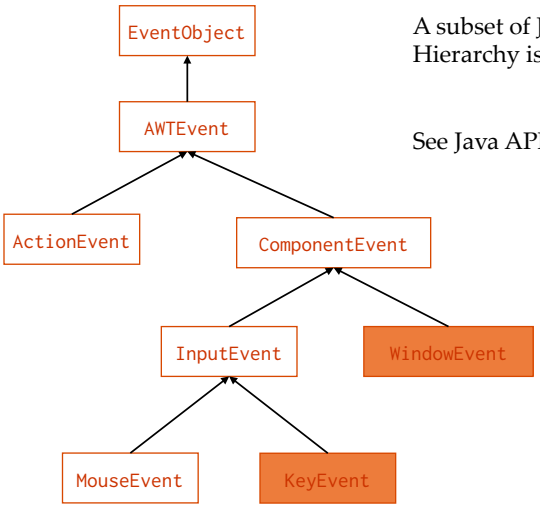
YORK UNIVERSITÉ UNIVERSITY

About Events in General…

- Events are objects that encapsulate some sort of external "happening"
  - the user did something
    - e.g., performed a mouse or keyboard action
  - the window manager did something
    - e.g., opened a window, shifted focus

YORK UNIVERSITÉ UNIVERSITY

Java's Event Class Hierarchy



A subset of Java's Event Class Hierarchy is shown here

See Java API for full hierarchy

YORK UNIVERSITÉ UNIVERSITY

Back to the sample app…

- Can you identify the observer and the observee?

YORK UNIVERSITÉ UNIVERSITY

Tasks

- slow down the projectile to have a slower trajectory
- make the projectile expire before it reaches the edge of the screen

YORK UNIVERSITÉ UNIVERSITY