

Dragging

October 21, 2008

1

Dragging Rectangles

- User interface effect that we want:
 - User depresses mouse button while mouse is over an rectangle to “pick up” the rectangle
 - User drags mouse (that is, moves the mouse while holding the mouse button down) to move rectangle
 - Rectangle should move same distance and direction as mouse moves
 - When user releases the mouse button, the rectangle should be “dropped”, (that is, it should stop moving)

2

Dragging Rectangles

```
public class DragARectangle {  
    public void onMousePress (Location point) {  
        // if mouse is over the rectangle, "pick up" the rectangle  
    }  
  
    public void onMouseDrag (Location point) {  
        // if the user picked up the rectangle, move it the same  
        // amount and direction as the mouse moved  
    }  
  
    public void onMouseRelease (Location point) {  
        // "drop" the rectangle  
    }  
}
```

3

Dragging Rectangles

Step 1: Picking up the Rectangle

- ☉ User depresses mouse button while mouse is over an rectangle to "pick up" the rectangle

```
private boolean rectGrabbed = false;
```

```
public void onMousePress (Location point) {  
    if (rect.contains (point)) {  
        rectGrabbed = true;  
    }  
}
```

4

Instance Variables

Instance variable is used to "remember" a value between methods

Instance variable
Declaration

```
private boolean rectGrabbed = false;
```

```
public void onMousePress (Location point) {  
    if (rect.contains (point)) {  
        rectGrabbed = true;  
    }  
}
```

Assignment statement

5

Instance variables vs. Local variables

```
private boolean rectGrabbed = false;
```

```
public void onMousePress (Location point) {  
    if (rect.contains (point)) {  
        rectGrabbed = true;  
    }  
}
```

Instance
variable

```
public House (DrawingCanvas mysteryCanvas) {  
    FilledRect wall = new FilledRect (...);  
    wall.setColor (WALL_COLOR);  
    ...  
}
```

Local
variable

6

If Statements

Like in Alice, "if statements" allow us to execute code only when some condition is true.

```
private boolean rectGrabbed = false;

public void onMousePress (Location point) {
    if (rect.contains (point)) {
        rectGrabbed = true;
    }
}
```

7

contains Method

contains method returns true if the shape (called "rect" here) contains a particular Location (called "point" here)

```
private boolean rectGrabbed = false;

public void onMousePress (Location point) {
    if (rect.contains (point)) {
        rectGrabbed = true;
    }
}
```

8

Dragging Rectangles

Step 2: Moving the Rectangle

- User drags mouse (that is, moves the mouse while holding the mouse button down) to move rectangle
- Rectangle should move same distance and direction as mouse moves

```
public void onMouseDrag (Location point) {  
    // Determine how far the mouse moved since  
    // mouse button depressed or since last call to  
    // onMouseDrag.  
    // Move the rectangle that distance  
}
```

9

Dragging Rectangles

Step 2a: Remembering last mouse location

```
private Location lastPoint; // Last mouse location  
public void onMousePress (Location point) {  
    if (rect.contains (point)) {  
        rectGrabbed = true;  
        lastPoint = point;  
    }  
}  
public void onMouseDrag (Location point) {  
    if (rectGrabbed) {  
        // Determine how far the mouse moved since mouse button  
        // depressed or since last call to onMouseDrag.  
        // Move the rectangle that distance  
        lastPoint = point;  
    }  
}
```

10

Dragging Rectangles

Step 2b: Calculating distance to move

```
public void onMouseDrag (Location point) {
    if (rectGrabbed) {
        // Determine how far the mouse moved since mouse button
        // depressed or since last call to onMouseDrag.
        double dx = point.getX() - lastPoint.getX();
        double dy = point.getY() - lastPoint.getY();

        // Move the rectangle that distance

        // Remember where this drag ended.
        lastPoint = point;
    }
}
```

11

Location

Location represents an x, y coordinate pair

- getX() method returns the value of the x coordinate (as a double)
- getY() method returns the value of the y coordinate

```
public void onMouseDrag (Location point) {
    if (rectGrabbed) {
        // Determine how far the mouse moved since mouse button
        // depressed or since last call to onMouseDrag.
        double dx = point.getX() - lastPoint.getX();
        double dy = point.getY() - lastPoint.getY();
        ...
    }
}
```

12

Local Variables

Local variables are used to remember values during a single method

```
public void onMouseDrag (Location point) {
    if (rectGrabbed) {
        // Determine how far the mouse moved since mouse button
        // depressed or since last call to onMouseDrag.
        double dx = point.getX() - lastPoint.getX();
        double dy = point.getY() - lastPoint.getY();
        ...
    }
}
```

13

Dragging Rectangles Step 2c: Moving the Rectangle

```
public void onMouseDrag (Location point) {
    if (rectGrabbed) {
        // Determine how far the mouse moved since mouse button
        // depressed or since last call to onMouseDrag.
        double dx = point.getX() - lastPoint.getX();
        double dy = point.getY() - lastPoint.getY();

        // Move the rectangle that distance
        rect.move (dx, dy);

        // Remember where this drag ended.
        lastPoint = point;
    }
}
```

14

move Method

move method moves a shape (called "rect" here) some distance horizontally (called "dx" here) and some distance vertically (called "dy" here) from its current location

```
public void onMouseDrag (Location point) {
    if (rectGrabbed) {
        // Determine how far the mouse moved since mouse button
        // depressed or since last call to onMouseDrag.
        double dx = point.getX() - lastPoint.getX();
        double dy = point.getY() - lastPoint.getY();

        // Move the rectangle that distance
        rect.move (dx, dy);

        // Remember where this drag ended.
        lastPoint = point;
    }
}
```

15

Dragging Rectangles

Step 3: Dropping the Rectangle

- When user releases the mouse button, the rectangle should be "dropped", (that is, it should stop moving)

```
public void onMouseRelease (Location point) {
    // Drop the rectangle
}
```

16

Dragging Rectangles

Step 3: Dropping the Rectangle

- When user releases the mouse button, the rectangle should be "dropped", (that is, it should stop moving)

```
public void onMouseRelease (Location point) {  
    // Drop the rectangle  
    rectGrabbed = false;  
}
```

17

```
private boolean rectGrabbed = false;  
private Location lastPoint; // Last mouse location  
public void onMousePress (Location point) {  
    if (rect.contains (point)) {  
        rectGrabbed = true;  
        lastPoint = point;  
    }  
}  
public void onMouseDrag (Location point) {  
    if (rectGrabbed) {  
        // How far did mouse move?  
        double dx = point.getX() - lastPoint.getX();  
        double dy = point.getY() - lastPoint.getY();  
        // Move the rectangle that distance  
        rect.move (dx, dy);  
        lastPoint = point;  
    }  
}  
public void onMouseRelease (Location point) {  
    // Drop the rectangle  
    rectGrabbed = false;  
}
```

Picking up /
dropping

18

```
private boolean rectGrabbed = false;
private Location lastPoint; // Last mouse location
public void onMousePress (Location point) {
    if (rect.contains (point)) {
        rectGrabbed = true;
        lastPoint = point;
    }
}
public void onMouseDrag (Location point) {
    if (rectGrabbed) {
        // How far did mouse move?
        double dx = point.getX() - lastPoint.getX();
        double dy = point.getY() - lastPoint.getY();
        // Move the rectangle that distance
        rect.move (dx, dy);
        lastPoint = point;
    }
}
public void onMouseRelease (Location point) {
    // Drop the rectangle
    rectGrabbed = false;
}
}
```

Moving

19

Summary of Methods

- Defined for FilledRect, FilledOval, FramedRect, FramedOval
 - public boolean contains (Location point)
 - public void move (double dx, double dy)
- Sample code:

```
FilledRect shape;
Location loc;
if (shape.contains (loc)) {
    shape.move (5, 10);
}
```

20

Summary of Methods

- Defined for Location

- public double getX ()
- public double getY ()

- Sample code:

```
Location loc;  
double x = loc.getX();  
double y = loc.getY();
```