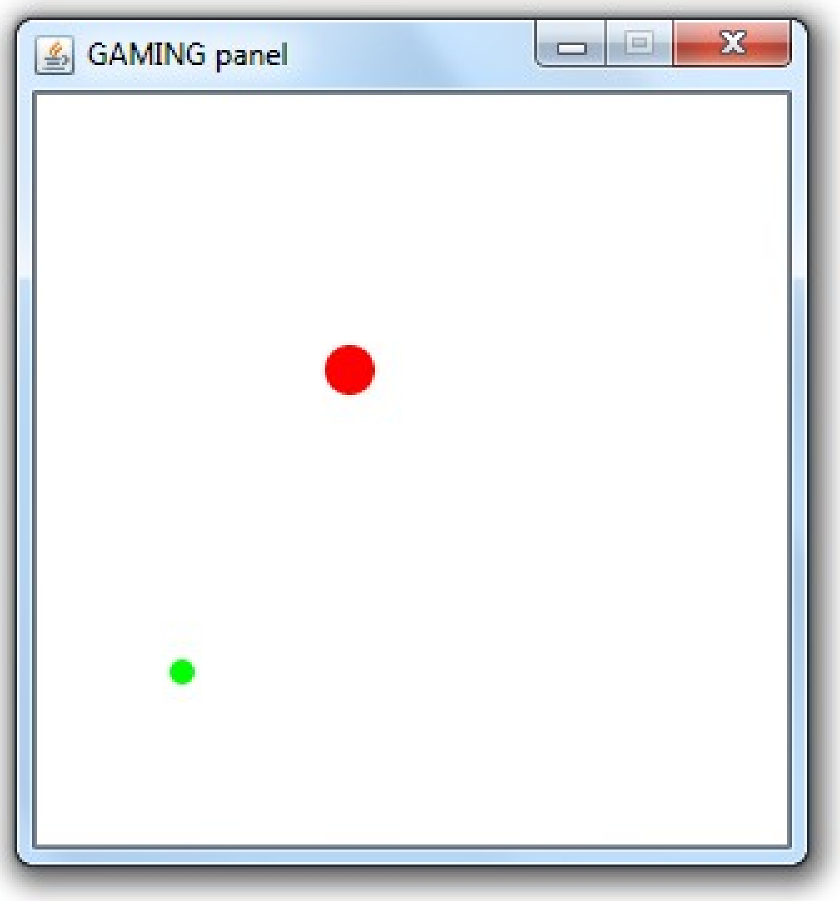
UW GAME DEV CLUB

Intro to game programming in Java (with almost-drawingpanel)



Contacts:

Shiny (mootothemax@gmail.com)

Aleksander (aleksander_01@live.com)

UW GAME DEV CLUB

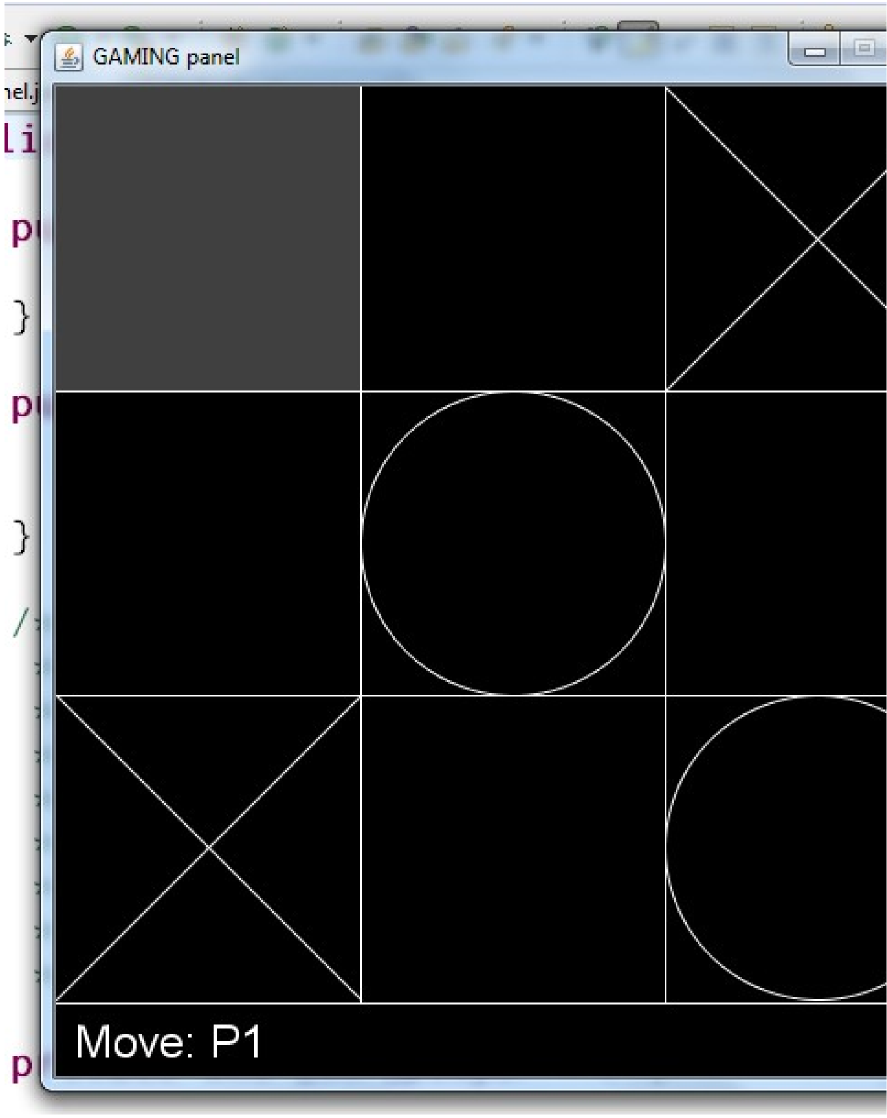
GamePanel

-Based off DrawingPanel and in Java (familiar)

-Way better than any of the other Java alternatives I've used

-Exports as a standalone runnable JAR file.

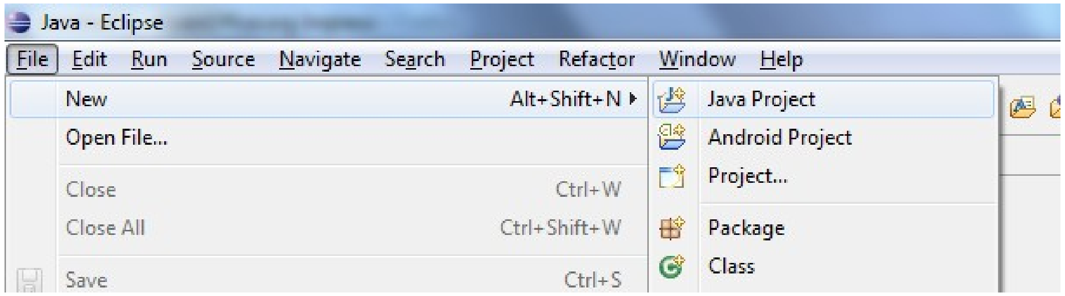
-Download here (click download zip):
https://github.com/spotco/GamePanel



We're not going to use this forever, but it's a good place to learn the basics!

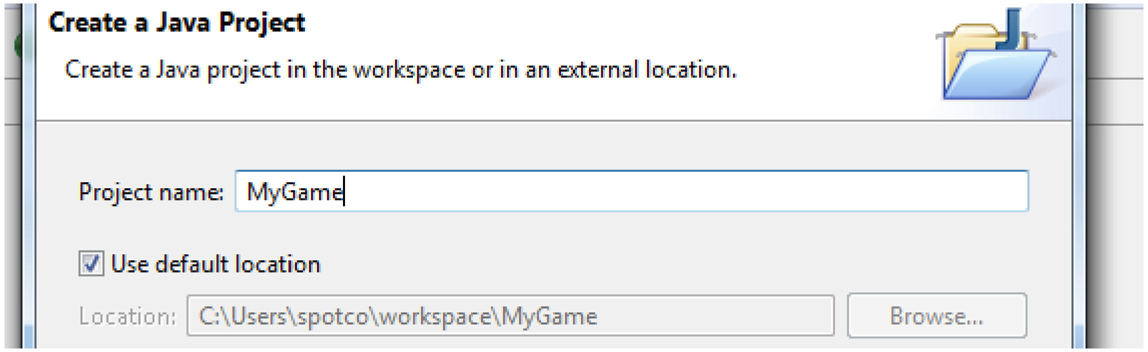
UW GAME DEV CLUB

Setup in Eclipse



New java project

Give it a name


Create a Java Project

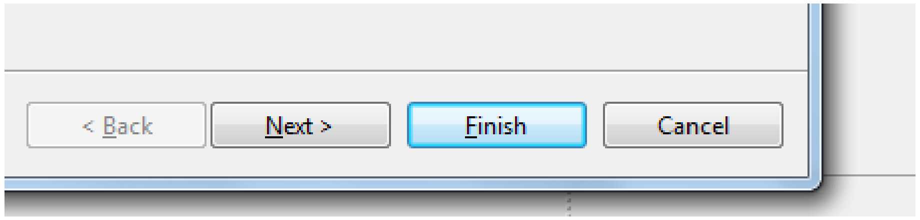
Create a Java project in the workspace or in an external location.

Project name: MyGame

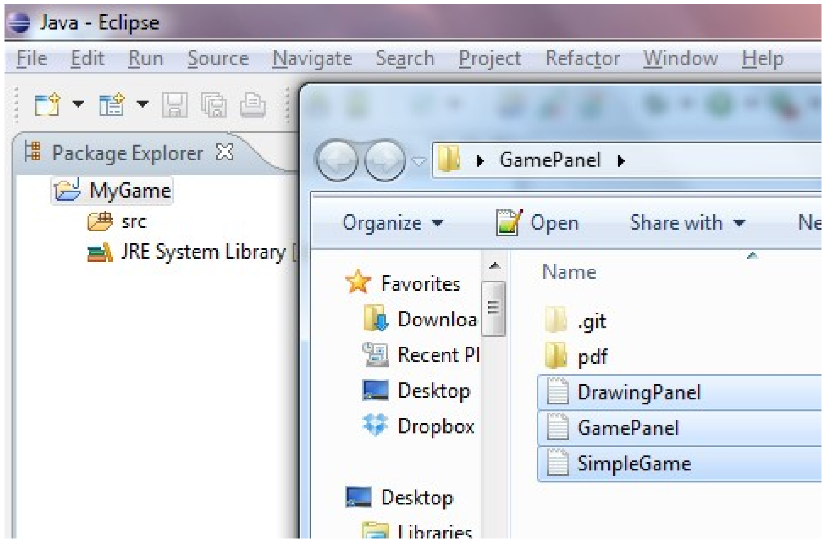
☑ Use default location

Location: C:\Users\spotco\workspace\MyGame

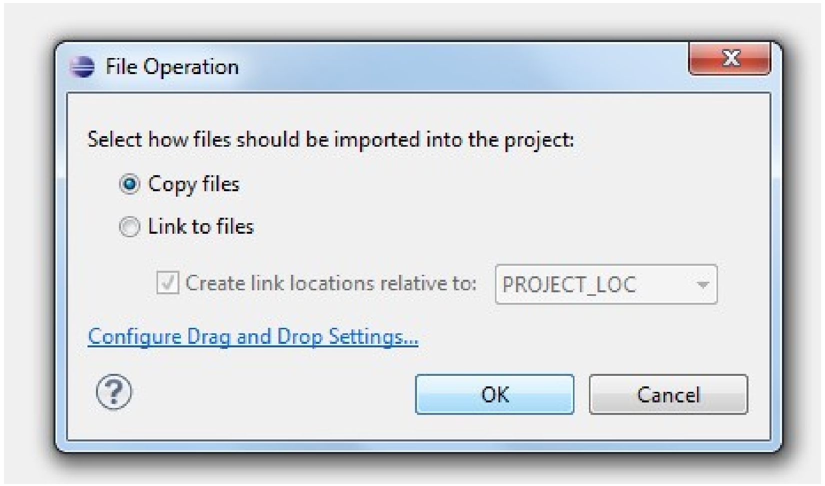
Finish



UW GAME DEV CLUB



Drag and drop the .java files into the "src" folder



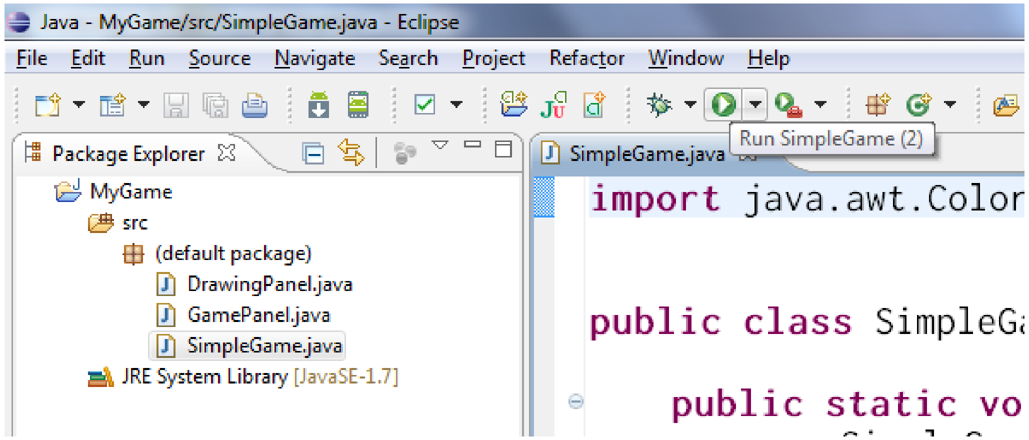
Select how files should be imported into the project:

[● Copy files]

○ Link to files

☑ Create link locations relative to: PROJECT_LOC

Ok


Java - MyGame/src/SimpleGame.java - Eclipse
File Edit Run Source Navigate Search Project Refactor Window Help
Run SimpleGame (2)
Package Explorer
MyGame
src
(default package)
DrawingPanel.java
GamePanel.java
SimpleGame.java
JRE System Library [JavaSE-1.7]
SimpleGame.java

```
import java.awt.Color

public class SimpleGa

    public static vo
```

Try running "SimpleGame" !

Making a new game

```
public class MyGame extends GamePanel {

    public MyGame() {
        super(300,300); //the window will be 300 width, 300 height
    }

}
```

And then somewhere else, add a main...

```
public static void main(String[] args) {
    new MyGame();
}
```

UW GAME DEV CLUB

Drawing Stuff

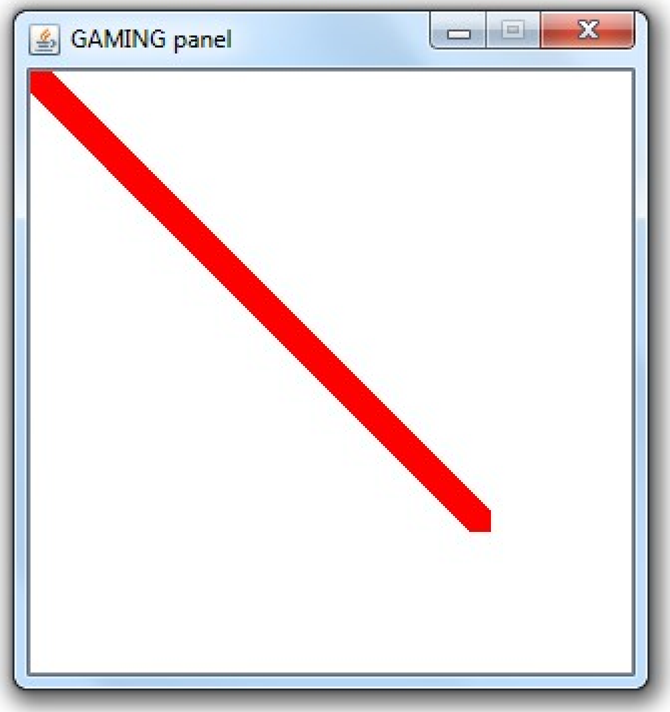
```
import java.awt.Color;

public class MyGame extends GamePanel {

    public MyGame() {
        super(300,300);

        /*
         * _g is an inherited field of type "Graphics"
         * (This is the same Graphics you know and love!)
         */
        _g.setColor(Color.red);
        _g.fillRect(50, 50, 100, 100);
    }
}
```

The "Update Cycle"

```
public class MyGame extends GamePanel {

    public MyGame() {...} //Do initialization stuff here!

    @Override
    public void update() { //This is a method we're overriding
        /*
         * This gets run every 20 milliseconds!
         */
        System.out.println("test");
    }
}
```

Animating something

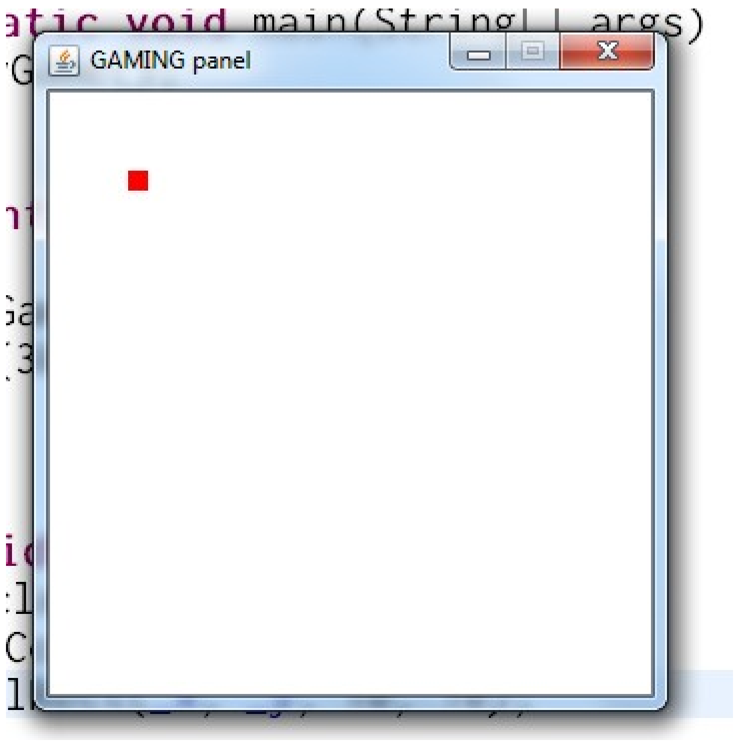
```
public class MyGame extends GamePanel {
    private int _x,_y;

    public MyGame() {...}

    @Override
    public void update() {
        _g.setColor(Color.red);
        _g.fillRect(_x, _y, 10, 10);
        _x++;
        _y++;
    }
}
```



What's going on here?

Animating something

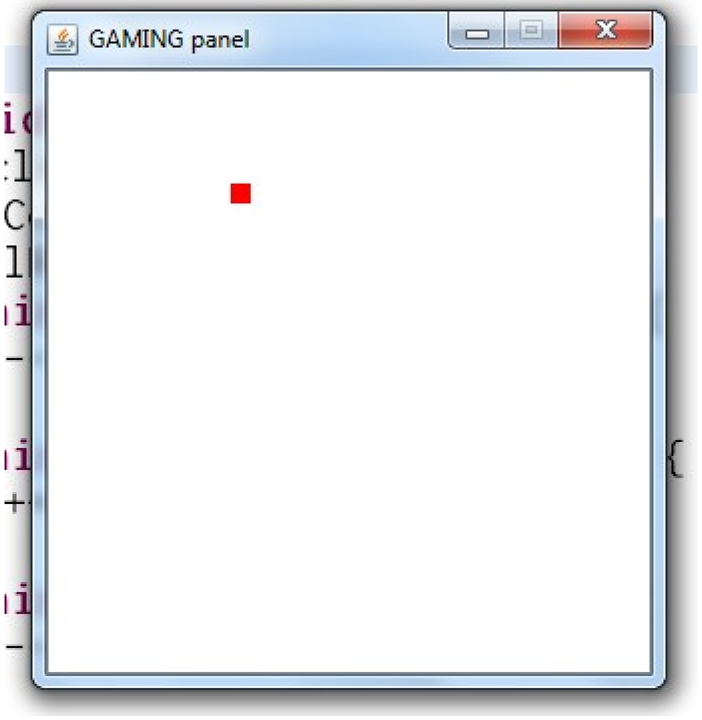
```
public class MyGame extends GamePanel {
    private int _x,_y;

    public MyGame() {...}

    @Override
    public void update() {
        this.clear();
        _g.setColor(Color.red);
        _g.fillRect(_x, _y, 10, 10);
        _x++;
        _y++;
    }
}
```



Clear the screen before drawing.
Fill a white rectangle the size of the screen
OR
this.clear().

Controlling Something

```
@Override
public void update() {
    this.clear();
    _g.setColor(Color.red);
    _g.fillRect(_x, _y, 10, 10);
    if (this.is_key_down(KEY_LEFT)) {
        _x--;
    }
    if (this.is_key_down(KEY_RIGHT)) {
        _x++;
    }
    if (this.is_key_down(KEY_UP)) {
        _y--;
    }
    if (this.is_key_down(KEY_DOWN)) {
        _y++;
    }
}
```



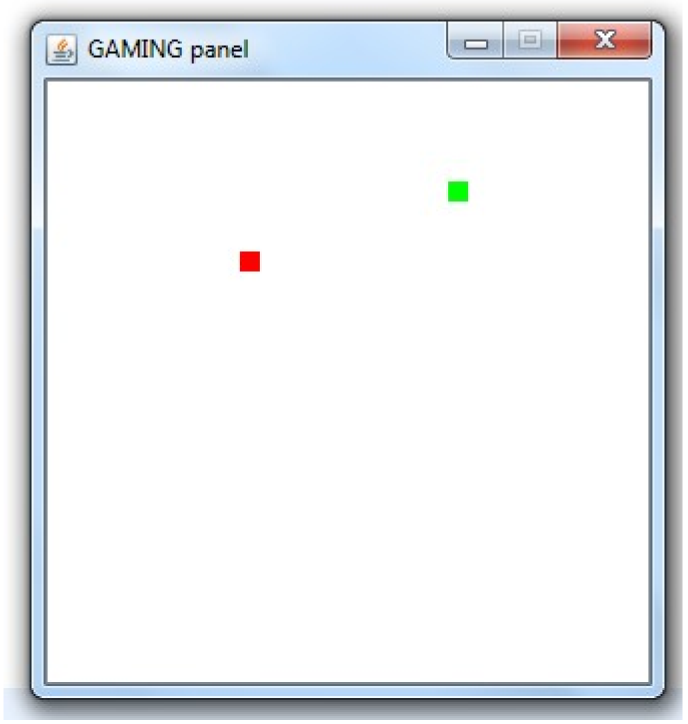
Let's make “Snake”!

You, (the red dot) want to grab the green dot.
How to do this?

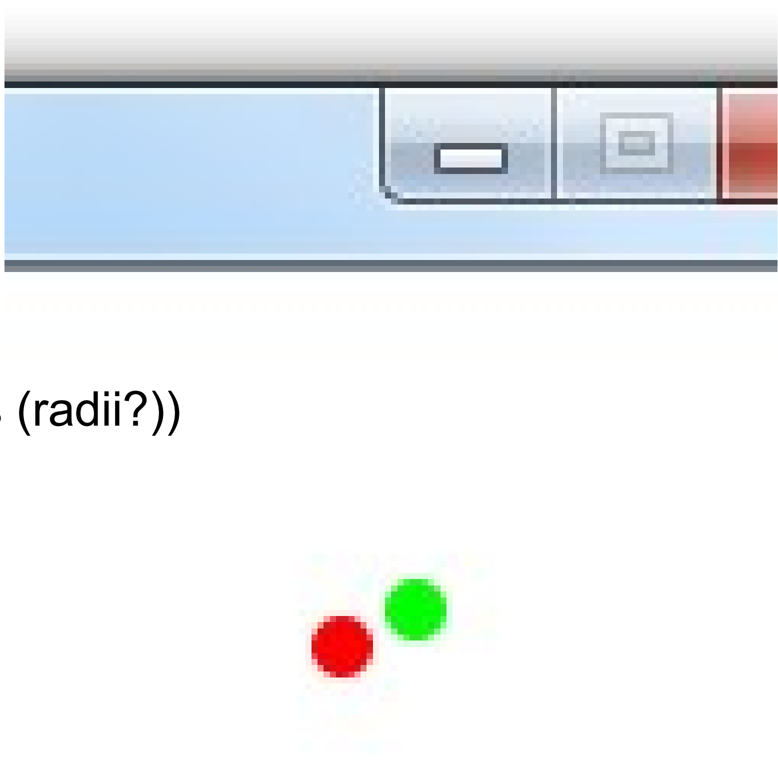
-Store the location of the green dot

-If the red dot (player) is close to the green dot, the player just “ate” the green dot

-Increment the score, and move the green dot somewhere random on the screen



How to detect if the player “hits” the green dot?



Here's a real easy way to do it:
see if circles are colliding!

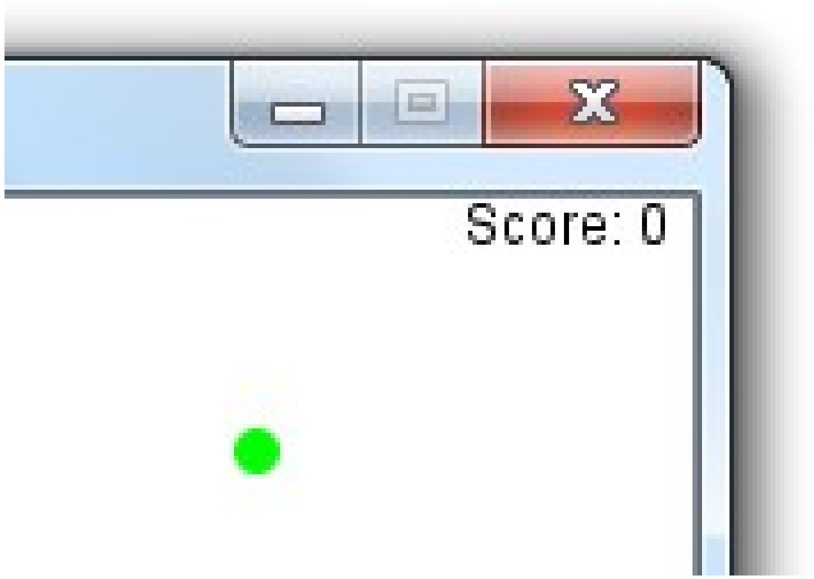
Two circles are colliding if

(distance between the centers) < (sum of the two radius (radii?))

```
public bool is_collide(int x1, int y1, int x2, int y2, int radius_sum){...}
```

UW GAME DEV CLUB

Draw some UI

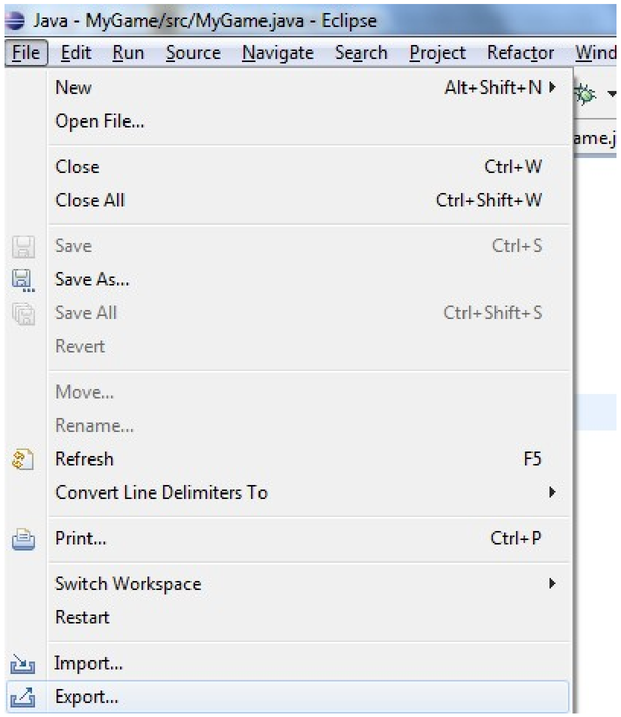


```
_g.setColor(Color.black);
_g.drawString("Score: 0", 250, 10);
```

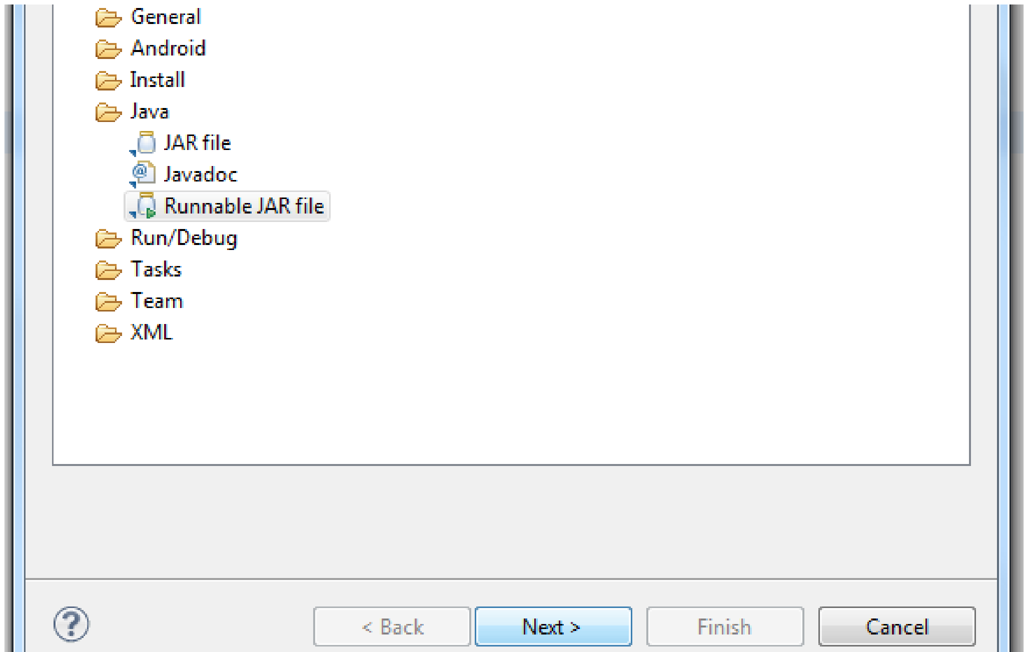
Make it show the “actual” score, of course!

UW GAME DEV CLUB

Exporting as a Runnable JAR

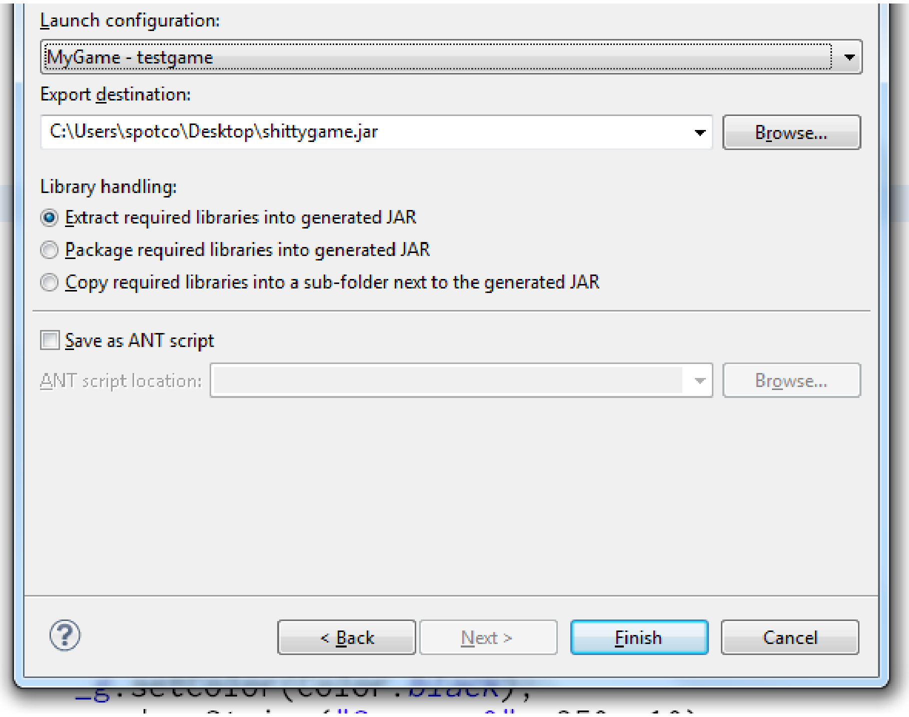


File -> export



Select Runnable JAR file, then next

UW GAME DEV CLUB



Launch configuration: MyGame - testgame

Export destination: C:\Users\spotco\Desktop\shittygame.jar

Library handling:

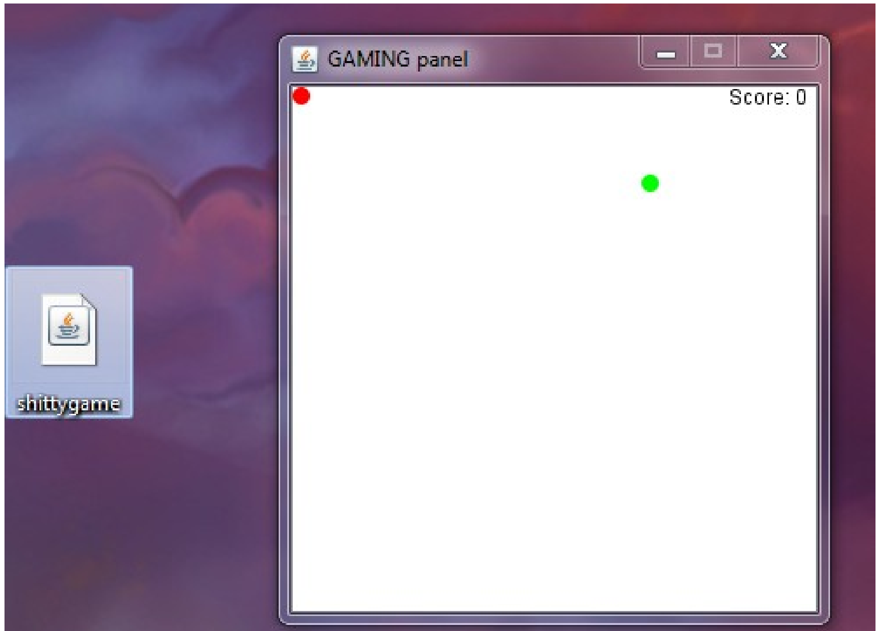
- [✓ Extract required libraries into generated JAR]
- Package required libraries into generated JAR
- Copy required libraries into a sub-folder next to the generated JAR

Save as ANT script

Select the class with your “main” in Launch configuration, and specify the output JAR file name.

Then, Finish.

Run the jar to play the game outside of eclipse.



UW GAME DEV CLUB

A few ideas to get going...

Easy:

Implement the score functionality. Collect the dot to increment score by 1.

If the player walks out of the screen, return him back to the screen.

Medium:

Make multiple green dots that could be collected (maybe of different size and point value). What would be the “smart” way of doing this?

Hard:

Make the dots run away from the player. They don't want to get eaten!