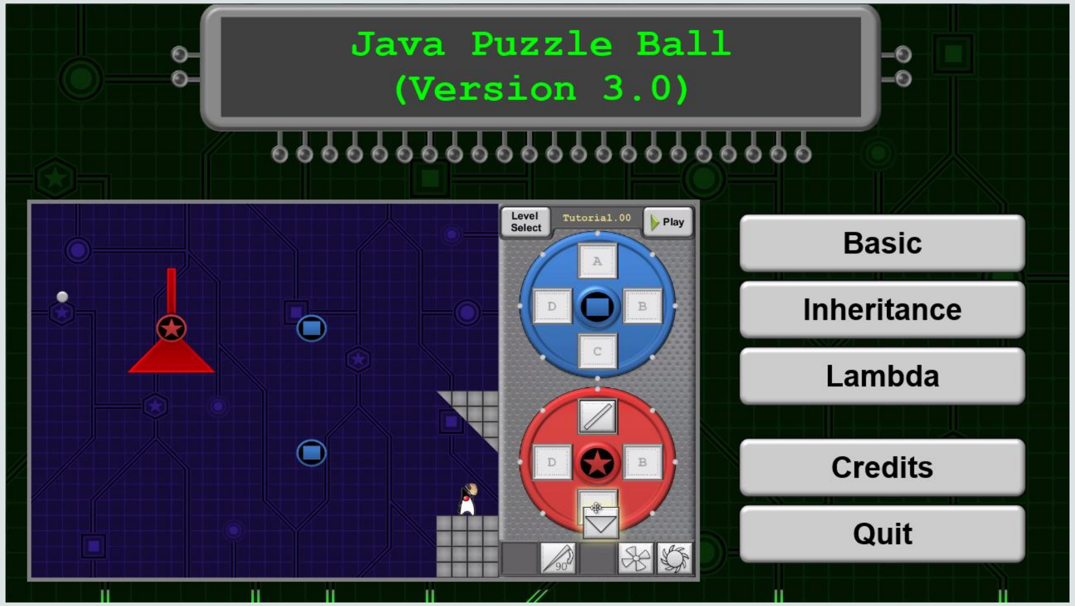
Java™

ORACLE®

Java Puzzle Ball

Nick Ristuccia

Lesson 2-1
Implementing More Game Behaviors



Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Programming is Iterative

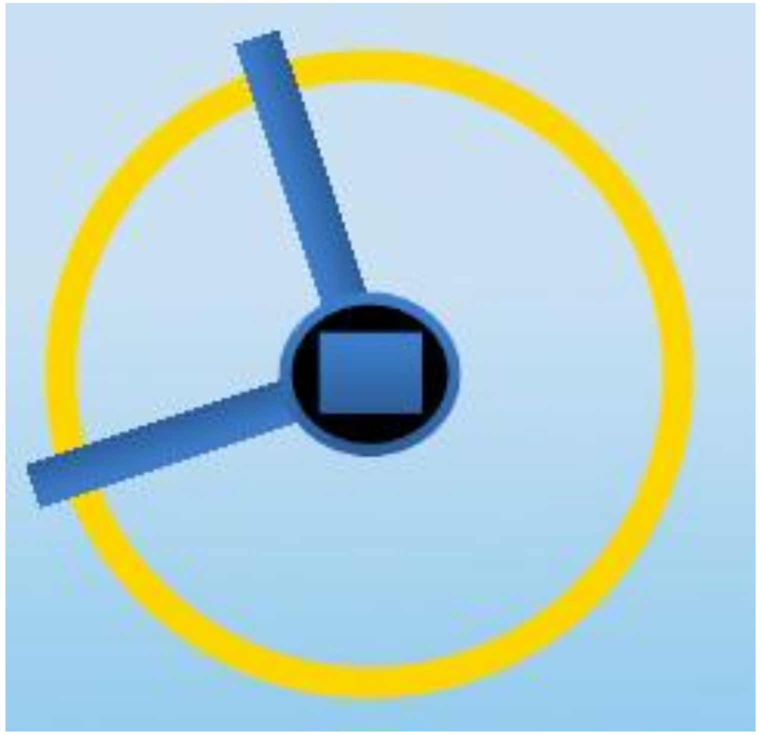
- Early progress might seem small and slow.
- Early programs might seem unimpressive.
  - Banking software is very traditional and lends itself to modification.
  - But the progress still feels good as a programmer.

- It's ok when development happens slowly.
  - In fact, it's very common.
  - First, you implement little features or make simple changes to existing features.
  - Then, you combine or build off this progress.
- Let's go behind the scenes of Java Puzzle Ball to demonstrate this…

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

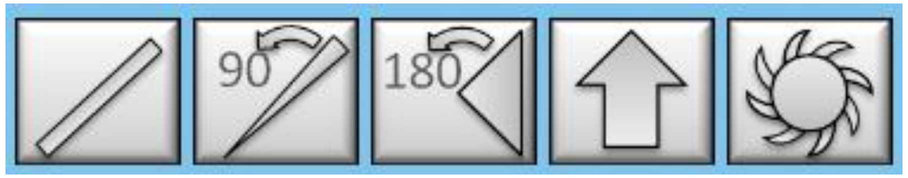
August 16, 2013

- This version isn't very fun
  - It's not a game yet.
- Goals of this version:
  - Have the developer learn Java FX.
  - Implement a few basic features.
- Notable features:
  - Display images on screen.
  - Detect mouse events.
  - Rotate BlueBumpers.
  - Drag and drop an icon into slots (N, E).



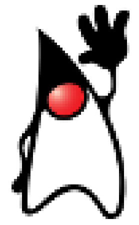
August 22, 2013

- One week later:
  - This version still isn't a game.
  - But it's looking more impressive.
- Notable features:
  - User Interface (UI) wheels and icons positioned on the right
  - A RedBumper
  - Colorized attachments
  - More icons to drag and drop



September 27, 2013

- About one month later:
  - This version could be called a game.
  - The goal is to deflect the ball to Duke.
- Notable features:
  - A Play button and a goal (Duke)
  - A ball that can move and be deflected
  - More shapes that can be attached
  - Yellow lines (for collision detection)
  - Wheels that snap to the nearest 45-degree increment


Duke

A Quick Note about Yellow Lines

- These lines must rotate along with each bumper.
- The Math to do this is messy:

```
        theta = Math.toRadians(180-theta);
        double r = image.getHeight();
        double x = r*Math.sin(theta) -(image.getWidth()/2)*Math.cos(theta) +pivotX;
        double y = r*Math.cos(theta) +(image.getWidth()/2)*Math.sin(theta) +pivotY;
        p1.setLocation(x,y);
        x = r*Math.sin(theta) +(image.getWidth()/2)*Math.cos(theta) +pivotX;
        y = r*Math.cos(theta) -(image.getWidth()/2)*Math.sin(theta) +pivotY;
        p2.setLocation(x,y);
        r = 0;
        x = r*Math.sin(theta) +pivotX;
        y = r*Math.cos(theta) +pivotY;
        p3.setLocation(x,y);
        walls.get(0).setLine(p1,p2);
        walls.get(1).setLine(p2,p3);
        walls.get(2).setLine(p3,p1);
```

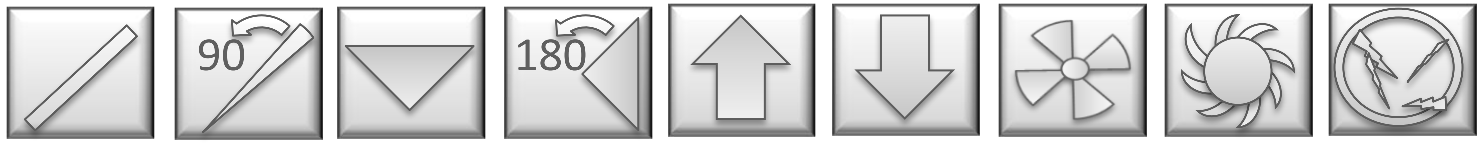
Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

When Working with Complex Code…

- It's easy to make mistakes with complex code.
- You might wreck your code and not know how to fix it!
- **Version Control** lets you do a few helpful things:
  - Store different versions of your code.
  - Compare these versions side-by-side (diff)
  - If you break something, you can roll back to an earlier version of your code.
- Oracle accommodates a version control solution for developers through **Oracle Developer Cloud Service.**

Differences Between Early and Final Versions

- Yellow debug lines for collision detection.
  - These are hidden in the final version.
- Wheels snap every 45 degrees.
  - It's easier to design levels and program collisions.
- When dragging an icon to a slot, it usually just implements a Simple Wall.
  - There were plans to implement other behaviors
  - Some didn't make the cut.



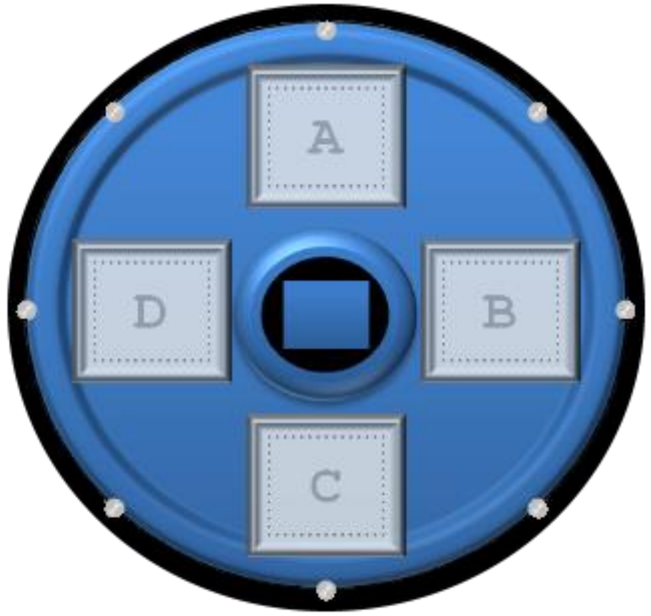
- There's one particular behavior that Lesson 2 is designed around.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Exercise 2

6 & 7 too, if you're curious to see other behaviors

- Play **Basic Puzzles 8 through 11.**
- Consider the following:
  - What happens when you rotate the BlueWheel?
  - How else can you affect the rotation of bumpers?



Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Java™

ORACLE®