DoodleJump
Design Section
Agenda

I. Icebreaker
II. Check-in
III. Overview
IV. Containment/program structure: understanding the relationship between various objects and how they interact with one another
V. Math: how to calculate doodle’s position/velocity and check for collisions
VI. Overview of key concepts: ArrayLists, For Loops, Generics, Java FX
VII. Wrap-up
Icebreaker

[whispering]
HAVE YOU BEEN INTRODUCED TO KEVIN?
Check-in
Overview

- Concepts being tested
- Relevant lectures
- Logical vs. graphical objects
- Classes needed
Checkpoint #1: logic vs. graphics

- You should have an understanding of which classes will handle logical components of the game and which classes will handle graphical components.
Containment, Inheritance, and Association
Checkpoint #2: classes

- At this point, everyone should have a general understanding of the classes you’ll need to write and the purpose of those classes
Coding incrementally

- Get PaneOrganizer, the quit button, and the window to show up
- Get Doodle to show up
- Get the first platforms to show up
- Get Doodle to respond to keys
- Start simulating physics
- Etc.
Pseudocoding the math

**Goal:** understand constraints on platform generation

**Given:**

- Start with a 7x10 grid
- Each platform is 3 horizontal pixels
- The doodle is 1 pixel
- Doodle can jump 4 pieces vertically (max)
- Doodle can move 1 piece horizontally in either direction (max)
Pseudocoding the math (continued)

Generating a random number between two vertical bounds

method randomInt(int low, int high):
    return low + (int) (Math.random()*(high - low + 1))
Movement overview

At every timestep:

- Update the doodle’s velocity
- If the doodle will be less than halfway up the screen:
  - Update the doodle’s position
- Else:
  - Calculate the difference between the midpoint and the doodle’s position
  - Move all platforms down by the amount of difference
  - Remove platforms that are now off screen
  - Generate new platforms as necessary
  - Set the doodle to the midpoint
- Check for collisions between the doodle and every platform
Pseudocode: updateDoodle (#4)

updateDoodle:
    _doodle’s velocity = _doodle’s velocity + ACCELERATION * DURATION
    updated_yPosition = _doodle’s yPosition + _doodle’s velocity*DURATION
    set y location of _doodle to be (updated_yPosition)
scrollPlatforms:
    if _doodleY < PANEL_HEIGHT/2 and _doodleVelocity < 0:
        difference = PANEL_HEIGHT/2 - _doodleY
        for plat in ArrayList _platforms:
            Set plat loc to be (platformX, platformY + difference)
        _doodle.setLocation(_doodleX, PANEL_HEIGHT/2)
Pseudocode: bounce (#5)

```
bounce:
    for all platforms in ArrayList _platforms:
        if _doodle’s velocity >= 0 and _doodle.intersects(platform X, platform Y, platform width, platform height):
            _doodle’s velocity = REBOUND VELOCITY;
```
Checkpoint #3: math

- At this point you should have a good mathematical understanding of:
  - Constraints for platform generation
  - Updating the Doodle’s position and velocity at each point
  - Moving the Doodle up or scrolling the platforms down depending on the Doodle’s position
  - When and how to check if the Doodle intersects with a platform; what action to take if it does intersect
New concepts covered

- For loops
- java.util.ArrayList
- Generics
- Coding incrementally
Anonymous Mid-Semester Survey

https://tinyurl.com/cs15fall2019

Leave us feedback!! We read all of it!!

If you want to leave more feedback throughout the course, we have an anonymous feedback form on the course website.
Responsible CS: Algorithmic Bias

Activity: Drawing!

- Take out a piece of paper
- You will draw three things: one at a time, 20 seconds each
- Try to be as accurate and detailed!
First Drawing: Shoe
Second Drawing: Doctor
Third Drawing: Eating Utensil
Wrap-up & Last Questions