Day2: GreenFoot Morning, Flappy & Penny

MARGARET ELLIS
DEPARTMENT OF COMPUTER SCIENCE
Create World and Bird

- FlappyBirdWorld
  - Set background image
- Create a New Subclass of Actor: Bird
  - Use Bird Image
New Scenario

- Open Greenfoot
- Open FlappyBird_Starter
Add Bird to World

- Edit FlappyBirdWorld
- Notice the dimensions of the world
- Add a Bird field variable
- Create a Bird Object in the constructor and add it to the World
- For the y position try using getHeight()/2 so it adjusts relative to the window height
Falling Bird

- Make the bird fall
- dy is positive 2 because downward because coordinates
- This would be a constant rate of change, or constant velocity
- Inspect the Bird to see how setLocation impacts the Y value.
Falling Bird

- Bird should fall according to the square of time
- After 1 sec: 1 ft
- 2 sec: 4 ft
- 3 sec: 9 ft
- Etc.
- Upside down parabola, speeding up and getting faster as it goes down
Falling Bird (change image, give values to try)

- add a gravity variable $g$
- Update $dy$ by $g$ every act
- Run again… looks more realistic!
- Experiment with various values for $dy$ and $g$
Continue from yesterday
KeyPresses

- Create a new method `checkForKeyPress()`
- Call the `checkForKeyPress()` from `act()`
- Run again!
- Inspect
- Experiment with values of `dy` and `g`
Some Tips

- An actor can call `getWorld()` and then can call methods on the world

- In the Greenfoot Help Menu reference:
  - “Greenfoot Class Documentation”
  - “Java Library Documentation”
AssignmentsA - Match the Demo!

- Add pipes to fly over. The pipes will be a new class. The pipes should move from right to left on the screen. What happens when the bird hits a pipe?
  - Create a Pipe class (use the anchor from Goldie as an example)
  - Add a pipe to the world
  - Make that pipe move right to left across the screen
  - Make the game end if the bird and pipe touch (use Greenfoot.stop() )

- Challenge: Automatically add new pipes at set intervals. Can you add them randomly?
public class Pipe extends Actor {

    public void act() {
        move(-5);
    }

    public void checkPipe() {
        if (isTouching(Pipe.class)) {
            Greenfoot.stop();
        }
    }
}

public FlappyBirdWorld() {
    super(600, 400, 1, false);
    Flappy = new Bird();
    addObject(Flappy, 100, getHeight() / 2);
}

public void act() {
    count = count + 1;
    if (count == 100) {
        Pipe nextPipe = new Pipe();
        addObject(nextPipe, 600, 300);
        count = 0;
    }
}
Match the Demo!

- Use 2 images for the bird, one for when it flaps up and one for when it is falling down.
  - Look in the code for when it would be flapping up and when it is falling down
  - Look for a method to use that changes the image
- Use a different constructor for the world so that it can be unbounded (the bird can go beyond the upper and lower boundaries)
- Extended:
  - Add pipes with spaces in between to fly through. Should the user be able to use the down arrow too?
  - Set angle of bird based on speed
Break
Brave, not perfect

- https://www.youtube.com/watch?v=fC9da6eqaqg
DEMO Penny Game
CliffWorld

- Open Greenfoot
- Open Penny Scenario
- Create 2 New Subclasses of Actor: **Mover** and **Platform**
  - No images needed yet
Mover Class Constants

- Create constants for the mover for vertical acceleration and sideways speed.
Mover Class Field Variable

- Create a field variable for vertical speed
- Write a setter method for vertical speed
Mover methods side to side

- Move right
- Mover left is symmetric
SubClasses

- Mover and Platform will be abstract classes
- Create a subclass of Mover: Penguin
- Create a subclass of Platform: Cliff
- Use the addObject method to add a new Cliff and Penguin to CliffWorld

```java
addObject(new Cliff(), 80, 500);
addObject(new Penguin(), 120, 200);
```
Subclasses Check Keys

- Penguin method to check for left and right arrow key input
- call method from act()
- Run it
- Move penguin right to left
- Does she fall?
Mover methods

- Need to check if mover is still on the platform
- Write onGround method
- How to use it?
  - Experiment with having Penguin class call the onGround method
  - What should the Penguin do when onGround() returns false? True?
- Ideas
  - Experiment with penguin placement and constant value
  - Consider how the Penguin will fall
  - Add a getter for vSpeed
How did you use `onGround()`?

- One possibility for `checkFall()` method. Remember to call it from `act()`
- How does it work?
- How can we adjust it?
- How can we improve it?
Add fall method to the Mover class

- Remember how FlappyBird fell?

Call it from the checkFall() method of the Penguin class

- Other Mover subclass may fall for different reasons

What about hitting the bottom boundary?
Cloud

- What are the clouds behaviors?
- How do we make the cloud move?
- Consider methods and fields available to Cloud.
- Add a Cloud to the World.
- Experiment and Run
Cloud Motion

• Does the cloud move back and forth between boundaries?
• || is the OR operator (arabic keyboard?)
• How to deal with bouncing off boundaries?
Cloud Behavior

- What else does a cloud do
  - It can catch penguins!
- Experiment and Run
Assignments

- Make your game match the demo
  - Make the Penguin jump when the space is pressed (hint: you copy and paste entire methods from previous projects)
  - Add another cliff

- Look for code that can be placed into reusable methods
  - atBottom(), gameEnd()

- Experiment with clouds and cliffs

- Add charms for Penguin to grab when jumps
Challenges

- Different scenery and actors?
- Arrows accelerate and break?
- Keyboard to control clouds?
- How would you keep score?
- How would you add levels?
- Sound effects?
- Experiment with the debugger
Recap

- What did you learn today?
- What do you need more practice with?
- What is fun?
- What is confusing?
**Lines of code**

- **Method calls**
  - `move(5), turn(45)`
  - `super.act()`

- **if statements**
  - `if (isAtEdge())`
  - `if (Greenfoot.isKeyDown("down"))`

- **Method definitions**
  - `checkForEdge()`
  - `isKeyPressed()`

- **Variable declarations**
  - `private int sSpeed,`
  - `private Bird flappy`

- **Constructors**
  - `flappy = new Bird()`

- **Assignment statements**
  - `dy = dy + g`
**Programming & Game Development Concepts**

- Object Oriented Design and class hierarchy
  - Consumer parent class
- Randomizing
  - Greenfoot.getRandomNumber()
- Keystrokes
  - isKeyDown()
- Window co-ordinates and relative positioning
- Velocity and acceleration (falling)
- Animation - change image based on action
  - wings
  - bird angle
- Environment changing
  - Clouds moving
  - Pipes appearing
- Keeping score
- Ending Game
Resources

- Greenfoot Help Menu
- TechGirls site (techgirls.cs.vt.edu)