CSE 113 A

March 15 - 19, 2010

Announcements

- Lab 3 posted this week
- Friday, March 26th – Review for Exam 3
- Monday, March 29th – Exam 3
- Wednesday, March 31st – Go over Exam 3
- Friday, April 2nd – Class cancelled
  - (Adrienne will be out of town April 1st – 4th)

Chapter 6

- Planets and gravity simulation
- Note SmoothMover and Vector – they will be part of Lab 3 assignment as well.

3

Overloading

- Note that there are two constructors in some of the classes (like Body).
- Normally, you would not be allowed to create two methods with the same name, but in this case it is allowed and is called method overloading.
- Method overloading (having two methods with the same name in the same class) is only allowed when the methods differ in the number and/or type of parameters.

4

Apply Gravity (to a particular planet)

- First, get all the bodies in the scenario
- Then, apply the force of gravity from each body so that it impacts the motion (Vector) of the "current" planet

5

Getting the planets

- getWorld().getObjects(Body.class)
  - Returns a list that we need to store
- java.util.List<Body> bodies;
  - Creates a variable that holds onto a list of Body objects
- bodies = getWorld().getObjects(Body.class);
  - Assigns the list of bodies to the variable we've just created

6

Now what?

- So, we have a list of planets, but now we need to cycle through the list and use each planet to help calculate movement.
- We can use a for-each loop to cycle though (or iterate over) the list of planets.

7

For-each loop (Syntax)

```
for( TypeOfElementInCollection variableName: nameOfCollection )
{
    //what to do with each element
}
```

8

For the planets

```
for(Body b: bodies)
{
    applyGravity(body);
}
```

9

Make the ball move

- call to move() is already there
- need to create a Vector with a (dx, dy) or (angle, length)
- then set the Vector of the Ball to be that Vector

↑ or ↗

10

Check for Edges

if(ball hits top)
- bounce (reverse direction Vertically)

else if (ball hits bottom)
- ? - bounce
- ! - end game ← need this eventually

if(ball hits left)
- bounce (reverse direction Horizontally)

else if(ball hits right)
- bounce (reverse direction Horizontally)

11

Check for collisions with paddle

- write the method to do this & call it from act()

if(collide with paddle)
change direction of ball

12

Optional

- Making the ball change its angle when it hits the paddle.
  - Reverse the y-direction the ball was moving
  - Find the center of the ball
  - Find the center of the paddle
  - If the difference between them is zero, do nothing.
  - Otherwise, set the dx of the ball to be
    - center X of ball – center X of paddle

13

Move paddle

MouseInfo m = Greenfoot.getMouseInfo();

m has a getX() method

We could set the location of our paddle to be the same as m's getX()

14

Check for Bricks

if(~~Ball~~ collides with Brick)
- remove Brick from World
- need to decide if a random activity will happen
  - random activities happen 30% of the time

15

Random Actvity

- Each happen with equal probability

Get a random number
if (number == 0)
  speedUpBall();
if (number == 1)
  slowDownBall();

16

For the other three:
- Get a list of all the Bricks in the world
- Use for-each loop to do something to each Brick

17

java.awt.Color

- There are several pre-defined colors in Java that you can use
- java.awt.Color.PINK, java.awt.Color.RED, java.awt.Color.ORANGE, java.awt.Color.YELLOW, java.awt.Color.GREEN, java.awt.Color.CYAN, java.awt.Color.BLUE, java.awt.Color.MAGENTA, java.awt.Color.LIGHT_GRAY, java.awt.Color.GRAY, java.awt.Color.DARK_GRAY, java.awt.Color.BLACK, java.awt.Color.WHITE

18

java.awt.Color

- You can also create a color using

  new Color(red, green, blue)

- where you substitute a number within the range 0-255 for each of red, green, and blue

19