Centipede HW 4

Firing Lasers

Centipede HW4 Overall Description

- Add a laser when user presses SPACE KEY. After it creates an instance of laser, it needs to appear just above the top edge of the player.
- Lasers that have been created move up by **speed** each frame.
- When a laser is fully off screen, it should be removed from the world.
- Only one laser can be on screen at a time, so when pressing space, make sure there isn't already a laser before firing
- The final product should look like this: ZIP | JAR

Laser Firing Logic

```
// in the Player act method
if(space is pressed){
    //check there is any laser in custom world
    //if there is no laser, then create an instance of laser
    //add the laser to the world above the top edge of the player
}
```

Getting A List Of Actors

There are many methods in greenfoot that return a List of objects. In order to use a List, first you need to import java.util.List at the top of your code, just below import greenfoot.*

```
import greenfoot.*;
import java.util.List;
public class SomeClass extends Actor {
...
}
```

You might notice that the methods always say they return java.util.List<A> and be wondering what the A is. The A inside the <> refers to the type of objects in the list. For example, List<Car> is a list containing Car objects. In order to make a variable that stores a List, you can say something like:

```
List<Car> cars = methodThatReturnsAListOfCars();
```

Using a List

You can see the full List API to see what methods are available.

Here is a short summary of the most useful methods:

```
List API
List<E>   (E is the class of the objects in the list)
Methods:
public int size()          Returns the number of items in the list
public E get(int i)        Returns the item at index i
```

Let's say you wanted to get a reference to the Player from the Centipede class. You could do this:

```
public class Centipede extends Actor {
    public void act() {
        // call getObjects on the world, to get a List of Player objects
        List<Player> players = getWorld().getObjects(Player.class);
        // Get the item at index 0 (this assumes there is exactly 1 player)
        Player play = players.get(0); // will crash if there is no player
    }
}
```

Centipede HW4 Detail Directions

- Right-click Actor and choose New subclass
- Name the class Laser and select the laser image you want
- Open your Player class
  - In the act() function, when user presses SPACE, create an instance of Laser class and add to your custom World above the top edge of the player
    - Only one laser can be on screen at once, so only fire if there is no laser in the world.
    - You can use getObjects(java.lang.Class<A> cls) to get a List of objects of a given type that are in the world
- Open your Laser class
  - Give the Laser an instance variable for the speed of the laser
  - In the act() function, move laser up by the speed (You can choose whether to use setLocation or move as long as it moves up.

Note: How to detect keyboard presses, Look at the Intro Java Greenfoot Notes

- Submit your code to this submission form.

HW 4 Checkpoint (Optional Self-check in 2021)

1. Declare a subclass of World named LaserWorld.
2. Declare a private instance variable named accuracy that holds a decimal value.
3. Define a default constructor for LaserWorld that creates a width 300, height 400 world with cell size 1 that is unbounded. Initialize accuracy to 0.
4. Create a public getter and setter for accuracy.
5. Create a public instance method named getLasers() that returns a list of Laser objects in the world.
6. Override the act() method of the world to do the following:
    a. When the spacebar is down, if there are not more than 5 lasers in the world already, add a laser centered horizontally in the world with its bottom edge even with the bottom edge of the world.

java.util.List API
List<E> (E is the class of the objects in the list)
Methods:

public int size()	Returns the number of items in the list
public E get(int i)	Returns the item at index i