Centipede HW 3

Navigating Mushrooms & Adding a Player

Finish/fix bugs in Centipede HW2

You can't do today's work until you have completed and understand last HW. Ask for help from peers/teacher.

CheckList

1) **Centipedes start moving east and bounce off edges correctly.**
2) **Mushrooms spawn in grid system.**
3) **Hint: make sure the centipede moves by an amount that divides evenly into the grid size. For example, if the grid size is 10, the centipede could move 1,2,5 or 10 since those all go evenly into 10.**

Centipede HW3 Overall Requirements

- When a centipede touches the right or left of a mushroom, it should move down one grid row and reverse horizontal direction.
    - Make sure to first move the centipede back horizontally so it is no longer touching the mushroom. That way it will not be touching the mushroom directly below the one that was touched (similar to keeping fully on-screen)
    - Make sure the centipede never gets placed off screen, even if that means it ends up being on top of a mushroom.
    - If moving directly down puts the centipede on a mushroom, that is ok. Centipedes should only respond to mushrooms they moved horizontally into, not ones they moved vertically into.
- Add a player to (center, bottom edge - half player height)
- Make a player moves by keyboard input. However, make sure player can't go outside of left, right, bottom edge and 80% of the world's height.
- Here is a demo of what it should look like.
- Here is a demo of an advanced version that animates the centipede legs moving.

Keyboard Movement and Collision

- Watch the following videos to learn about how to detect keyboard input and how to detect and respond to collisions between actors.
- You can watch them at 1.5x or 2x speed if it seems I am talking too slow. Just make sure you are comprehending the content.

Video 1 (Keyboard Movement)

Video 2 (Handling Collisions)

Add collision detection of Centipede to mushrooms

- Detect collisions using isTouching(Class name)

```
// Within Centipede class
if (isTouching(Mushroom.class)) {
    // Do something
}
```

- Each frame, after moving, centipedes that are touching a mushroom should move back, go down a level and reverse direction.
- This strategy will work great for basic cases, but the next slide will cover how to get more information so you can handle cases where the centipede moved down onto a mushroom.

Getting a reference to a touching mushroom

- You can get a reference to a touched object with [getOneIntersectingObject(Class cls)]

```
// inside an instance method such as act() in a class that extends Actor:
Actor garfield = getOneIntersectingObject(Cat.class);
```

- After that code, garfield is now either an instance of the Cat class that is touching the actor, or is null if the actor is not touching any instances of Cat. Note that the variable is of type Actor because getOneIntersectingObject returns an Actor, which means you can call any methods on garfield that the Actor class has, but cannot call methods that are only defined in Cat. Since garfield will be null if there is no intersecting Cat, you need to check for that first. Here is an example:

```
if (garfield != null) {
      garfield.move(10); // call move on the touched cat
}
```

- In cases where you need to call methods declared in Cat but not in Actor, you need to typecast garfield to be a Cat. If Cat defined a method *public void eat()*, you could call eat() on an intersecting Cat like this:

```
// declare a variable of type Cat and store a reference to an intersecting Cat
Cat garfield = (Cat) getOneIntersectingObject(Cat.class); // typecast garfield as a Cat
if (garfield != null) { // if there is an intersecting Cat
      garfield.eat(); // call eat() on the intersecting Cat
}
```

Special Case

When a centipede is touching a mushroom, they might have moved onto it completely from above, or could have just moved into it horizontally. Here are some tools you can use to find out more about the situation:

Return Type	Method
`Actor`	`getOneIntersectingObject(java.lang.Class<?> cls)` Return an object that intersects this object (or null if none exists)
`Actor`	`getOneObjectAtOffset(int dx, int dy, java.lang.Class<?> cls)` Return one object that is located at the specified cell (relative to this objects location). Will return null if none exists.

Get a reference to the mushroom and find out where it is in relation to the centipede, or simply try to get a mushroom at a specific location. In either case, if no such mushroom exists, null will be returned.

Add the Player and Player Movement

- Right-click Actor and choose New subclass
- Name the class Player and select the player image you want
- Open your custom World class
    - Add the player to your World (same way as you added a Centipede)
    - Don't hard-code the player's position. Add it to (center, bottom edge - half player height);
- Open your Player class
    - In the act() function, while the player is pressing an arrow key, the player should move in the direction of the arrow. Review how to detect keyboard input in previous videos or in Greenfoot Notes
    - Make sure the player does not go outside of left edge, right edge, bottom edge, and can't go above 80% of World's height (i.e. the player must stay in the bottom 20%).

Submission

Name Your project PX_Lastname_Firstname_Centipede

Submit your zipped project folder to [this form](#).

Here is a video on [how to zip files/a folder in windows](#).

Here is a video on [how to zip files/a folder in a mac.](#)