Assignment 4: Ghost and Pellet Collisions

Due 4/11, 2016 10:00am

1 Overview

In this assignment you will be implementing player-entity collisions - *eating* pellets and the *game-over condition* when touched by a ghost. This assignment will emphasize using dynamic lists and list operations.

2 What has changed

The `Entity` class has two new methods: `touched` and `touchedPlayer`. These methods will be called when the the entity is touched by another entity, and touched specifically by the player (respectively). The GUI driver also now assumes there is an extra constructor argument for `Pellet` (the purpose of which is covered in **(3)** below).

3 What you need to do

(1) First, change the `EntityBag` class to use `List<Entity>` instead of `Entity[]`. Any methods which returns an array should now return a list instead. Notice that the `getAllEntities` method used to return a *copy* of the array - you should keep this behavior and return a *copy* of the list and not the list object iself. You should also implement the following new methods in the `EntityBag` class:

```
public void removeEntity(Entity e) { ... }
public List<Entity> getEntitiesAtDepth(Depth depth) { ... }
public void onMove(Entity e) { ... }
```

The `removeEntity` methods will be used when a pellet is eaten, or when a player touches a ghost. The `getEntitiesAtDepth` method should return an array list containing the entities that are at the given depth. This method is used in the GUI driver.

The `onMove` method will be called when an entity updates grid coordinates. This method is responsible for detecting collisions and calling the `touched` methods on the appropriate entity objects. Two entities are colliding if they share the same *grid coordinates*. If entities a, b, and c share the same grid coordinates of e, then this method should make the following method calls:

```
a.touched(e)                    e.touched(a)
b.touched(e)                    e.touched(b)
c.touched(e)                    e.touched(c)
```

(2) Add a call to `onMove` whenever a dynamic entity moves (successfully). Make sure you call this method **after** updating the coordinates of the dyanmic entity (as the `onMove` method looks at the entity's grid coordinates to determine if a collisions has occurred).

(3) Implement `touchedPlayer` method of `Pellet` and `Ghost`. When a player touches a pellet the pellet should be removed from the entity bag. When a player touches a ghost, the player should be removed from the entity bag. You will need to add a constructor argument to `Pellet` to get access to the entity bag from within the class.

(4) The `touchedPlayer` method is currently never called from anywhere - but it should be. Fix this! You will need to make a modification to `Player` to do so. We're leaving this for you to figure out on your own (but it should only take a single significant line of code).

4 A Note

The implementation of many Java collections are not *thread-safe*. That is, they cannot be read from one thread of execution while being modified in another thread of execution. One observable effect of this problem is an exception called `ConcurrentModificationException`. This is very likely caused by a call to `add` or `remove` methods on an array list while you are iterating over it:

```
for (Entity e : entities) {
  // Hey, yo! I'm iterating here!
}
```

An easy fix (there are many others) is to simply iterate a *copy* of the array list instead. Because you're iterating a copy (which no one has a reference to but you), no one can sneakily add or remove from the list while you're reading from it and multiple thread accesses are completely safe. I'd recommend this solution:

```
for (Entity e : getAllEntities()) {
  // Hey, yo! I'm _safely_ iterating here!
}
```

5 Submission

Create a *zip archive* of your Eclipse project (including all template files) and upload it to the correct D2L dropbox before the due date. Again, no late work will be accepted.