Nicholas Valentine

Part 1:

What I did: For this I simply used my genericMovement class and plugged it onto the bug. I additionally in the next step created a bug class that extends from the genericMovement to give a few more options later on.

How Well it worked: This worked fairly well. After working on a later part I managed to work out a bug I was having with object where it would go out of bounds. This was resolved though.

What the Pros and Cons are: Pros of this is that it's simple and able to be modified. The cons is that it still has the bug of going out of the world.

What you might improve: As of this moment there isn't much I want to improve for this part.

Part 2:

What I did: For this part I used the ray object from task 4 and modified it such that after going 250 units it would die off. I used the collisionCheckLocal to determine if any bugs were in range of the ray. Using the ray's collision check I was able to replicate sending out hundreds of rays using a single ray. The check would go in 2 parts for each bug. part 1 was checking if the ray and bug were within 10 units distance from the home base (where it was deployed from) of the ray. If this was true we went to part 2. In part 2 the ray would be rotated such that it was now oriented towards the bug. The distance between these points was checked again to confirm that the object was indeed within range. The bug was then added to an arrayList within the ray that was cleared every frame before looking to see if there was a bug in range. Additionally a boolean was flipped within the bug telling it that it had been seen. This boolean was used to check if the bug needed to light up.

after this had been done for each ray the bat was then passed the list of rays and looked at the list of bugs they hit on the current frame. If a bug was hit it would be set as the target and the bat would start heading towards it. Due to the order of the ray list the closest ray is the last in the list so the bug selected it always the closest bug. In the instance of two bugs being hit by the same ray the closer bug to the bat was selected.In the instance the bat had a target the distance between the bat and it's target was compared to that of the potential new target before changing.

How Well it worked: This worked out quite well. In addition to it running smoothly due to the short number of rays, it also worked efficiently such that the bat wouldn't miss locating a bug.

What the Pros and Cons are: The pros of this was that it was resource efficient, and very effective. The cons of the system was that the bat could almost never miss detecting a bug.

What you might improve: I think I would improve this so that the bat could have a bit more limited ability to detect bugs than it currently has under this system.

Part 3:

What I did: For this part I added a few checks into the ray's collisionCheckLocal where I was determining if bugs were seen or not. The first part was creating a variable to hold the chance that I would get a true positive. From there after part 2 of checking a bug it would roll a random number and if it was below the true positive value then the bug was hit and it proceeded as normal. If the number was above the true positive the bug was ignored as we received a false negative.

After all this a new number would be used and in the instance of no bug in range or a false negative it would be determined if the bug had a true negative (or true false negative), or a false positive. In the case of the false positive a new bug on the current location of the ray. This bug was then rotated to a random position. Provided this location was within the bounds of the world it would be added to the arraylist that holds bugs in range.

How Well it worked: This worked out fairly well. As the ray went out further it's reliability went down and in addition fake bugs would be detected and the bat would chase after them.

What the Pros and Cons are: The pros to this system is that it's fairly simple compared to other versions that could be done. The cons of it are that due to the reliability of the echolocation originally there is not a huge number of fake bugs detected.

What you might improve: I think I would improve the original echolocation ability to be worse so that the bat detected more fake bugs.

Part 4:

What I did: For this part I implemented a function in the genericMovement object that allowed it to track a target. This proved a little difficult in getting the necessary heading out of it more than anything. This was due to the angle formula I was using only being able to calculate positive angles giving no indication of which direction I needed to turn. The work around to this I realized was simulate as though I rotated a few degrees to the right, and a few degrees to the left and then check the angle again and whichever was smaller determined the way I needed to turn.

Each frame I then checked if the bat was within 5 units of the bug it was chasing, and if it was the bug was told that it was dead. This was then used in the main area to decommission the bug so that it would not appear again. Additionally a check was added to the collisioncheckLocal of the ray so that it wouldn't detect bugs that weren't alive. A second boolean had to be added to the bug as well to check if it had been removed, otherwise an error would occur of trying to stop using a bug that was already not being used.

How Well it worked: This worked out pretty well. The bat would chase after the last known location of the bug that it knew from echolocation and would turn properly towards the bug.

What the Pros and Cons are: The pros of this were that it worked effectively and from the parent genericMovement object allowing for more modifications to the object later on and for better use of it in later assignments.

What you might improve: I think I would improve the way in which the angle is determined by the point as that caused quite a few issues when trying to figure out which was I needed to tell the bat to turn.

Overall Report

Overall this assignment felt a lot better than the blimp as I had a better understanding about the rays in this one and how to get them to work properly. Had the assignments been flipped I feel like this would have been the hard one while the blimp would have been a bit easier. While the ray the rays were used for this was a bit off it also would be possible in the case of the blimp to put the point in a random location within range of the blimp still giving it the more random jumps.