Kimin Lee Project 2 Documentation

My code can be run on a Windows build through the solution. Open CrashLoyal.sln in Visual Studio and click on “Local Windows Debugger”. Adjustments of how the game is set up can be changed in Controller_AI_KevinDill.cpp. Hold the “a” key and left click on a position to spawn an archer. Similarly, use the “s” key to spawn a swordsmen and use the “g” key to spawn a giant.

A main part of completing this assignment was handling the collisions. I handled collisions with the buildings, the river/bridge edges, map edges, and other mobs. For buildings, rivers, and map edges, I checked if the position was exceeding the set requirements and/or overlapping them to determine a collision. When a collision is found, I calculate the amount I need to shift the mob back for rivers, buildings and other mobs to make such adjustments to the position so that they don’t collide. For the map edges, I made it so that when the x or y coordinate reaches the map edges, the x or y coordinates won’t increase or decrease so that they go out of bounds. I applied the same logic for the river for the x axis so that mobs can't get pushed inside of the river. I set a restriction like so to avoid the mobs going out of the screen.

For mobs colliding against mobs, I considered and respected the mob’s masses. If a large mass mob A collides with a lighter mass mob B, mob A will push the mob B backwards and/or sideways. Mob B can’t go in the way of mob A due to a collision. This was done by applying the opposite shift force/vector to the mob position when they were supposed to be pushed back. For the other mob, I had to apply the shift force/vector to shift outside of the collision. Additionally, I set constraints on the speed of the mobs moving due to collision (and ofcourse just their movement speed) so that it doesn’t exceed the max speed of the mob. This was set by using the minimum between the shift speed value and the mob’s speed value to actually make the shift.

For processing mob collisions, I had to consider how making the adjustment/shift would cause any other collisions with the buildings, map boundaries, and rivers. To do this, every time I was about to make adjustments, I checked if the new position of the mob would cause any other collisions. If so, I didn’t make the adjustment.

I utilized a steering force in my project, as explained in piazza by Professor Dill. I used this source (https://gamedevelopment.tutsplus.com/tutorials/understanding-steering-behaviors-collision-avoidance--gamedev-7777) as a reference to apply my collision avoidance force. I used the velocity vector and multiplied it by a float to determine how far away the mob is able to detect an incoming collision from, which became my ahead vector. I then determined the closest other mob that the current mob will collide into with the ahead vector, then calculated the avoidance force. The avoidance force was calculated by negating the other mob’s position from the ahead vector, then normalizing the vector, and finally multiplying it by the half of the diagonal length of the other mob. Then the avoidance force was used to calculate the acceleration by dividing it by the mob’s mass, and I used that to apply the avoidance force to my vector for mob movement.

I think that the collisions worked pretty well, where no mobs are colliding into anything. All collision instances are taken care of, which makes the game very functional. The game is playable, and it works as expected. The mobs move to waypoints, attack turrets or enemy mobs, etc. I also think I organized the code in an intuitive way, where the structure and order is easy to comprehend.

Although collisions was one of the things I thought that went particularly well in this project, I would like to improve on some of the slight jitterings that occur when collisions are processed. I believe this is potential due to the framerate/elapsed time (especially after discussing with a TA). Making the jittering go away would get my game closer to a perfectly polished visual (for a bit Crash Loyal game).