# Design Plan For Boss Shockwave-Style Attack

*Overview:*

Looking at option B, we can see that the boss creates multiple rings of small bullets that move outward at a slow pace. This is phase 1 of the attack. The second phase creates a semi-circle of 5 larger bullets that moves outward at a fast pace. In addition, these large bullets all push the slow bullets out of the way and some of the slow bullets are even pushed along by the larger bullets. This second attack targets the player's position. Lastly, when the large bullets hit the player or the edge of the screen, they explode into a multitude of smaller bullets that fly in random directions backward.

*Walkthrough:*

To implement this attack, we will need to first generate the rings of slow-moving bullets. For this, we will need two things: the boss' vector-based position, and a designated bullet object.

Several circles (set to non-visible status) need to be drawn around the position. Each circle should be increased by a small factor to create a target-like structure. There should also be one large circle that is kept separate. Next, the circles need to be divided into many small subsections and each should be given a designated value (section 1, section 2, etc). Then the large circle should be given corresponding values (see figure 1).

For each section, a bullet should be spawned. This bullet will have a boundary (represented by a shape such as a rectangle), an assigned speed, a damage counter, and also a destination; this destination will be determined by the section it was spawned in. For example, if Bullet1 was spawned in section 1 of the first ring, then its destination will be section 1 of the large circle. The bullets may continue once they hit their destination and will eventually exit the screen. It is a good idea to include an isOnscreen Boolean, and should the bullet leave the screen, then it is deleted to save memory.

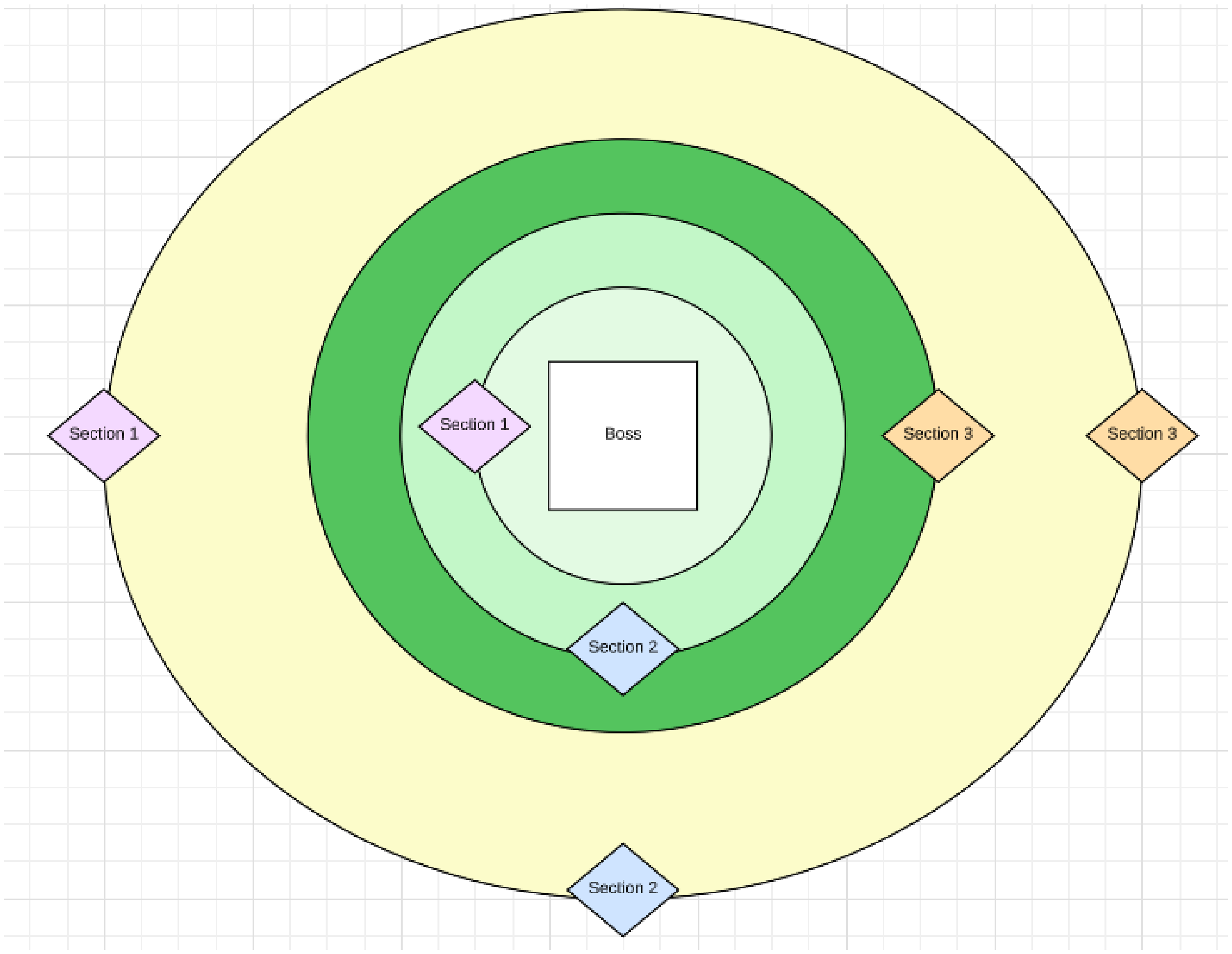


Figure 1.

For the second phase of the attack, an array list of 5 bullets must be created.

The position of the player must be passed, and this will serve as the destination of the 3rd bullet. The other bullets will be assigned corresponding sections in the smallest circle with their destinations determined by the matching sections in the large circle.

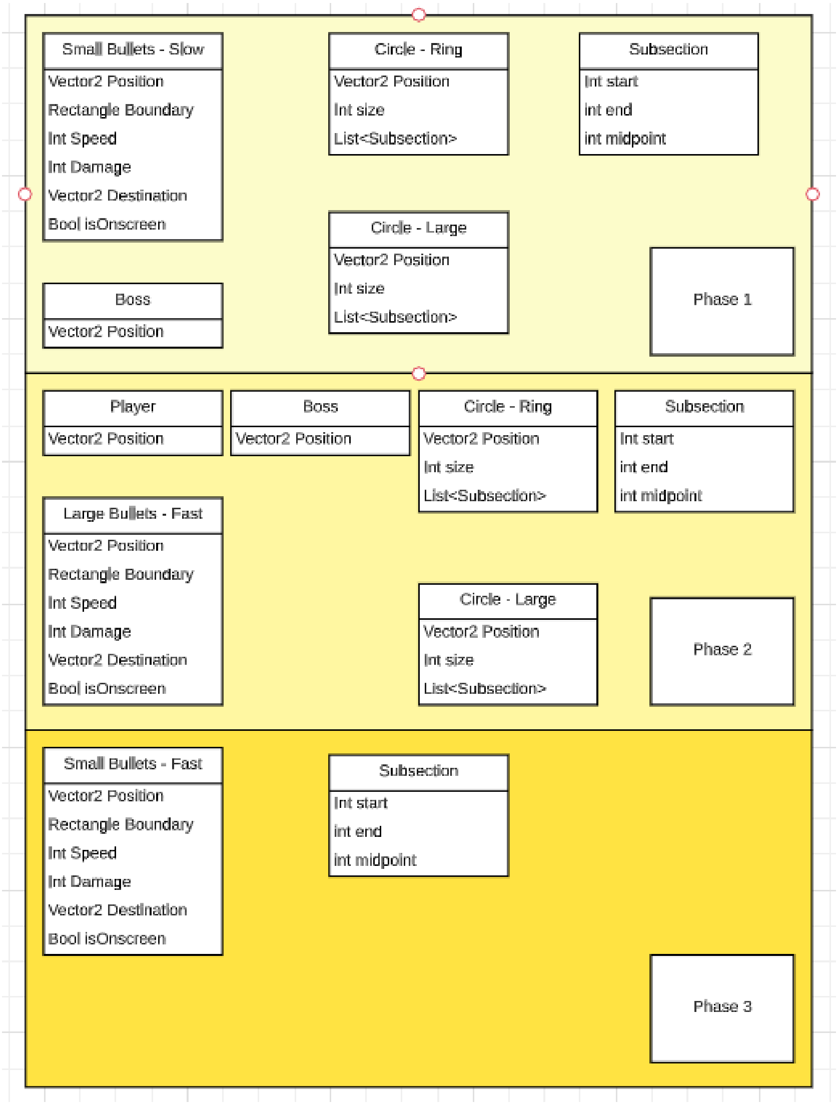
Now the bullets will be spawned. Their size/boundary will be much larger and speed faster than the smaller bullets. Once they make contact with the small bullets, advanced collision detection must be employed where the bullets are not destroyed but rather pushed away.

Each large bullet will have an isOnscreen Boolean. Once they hit the edge of the screen, this Boolean will switch to false and the third phase of the attack will commence.

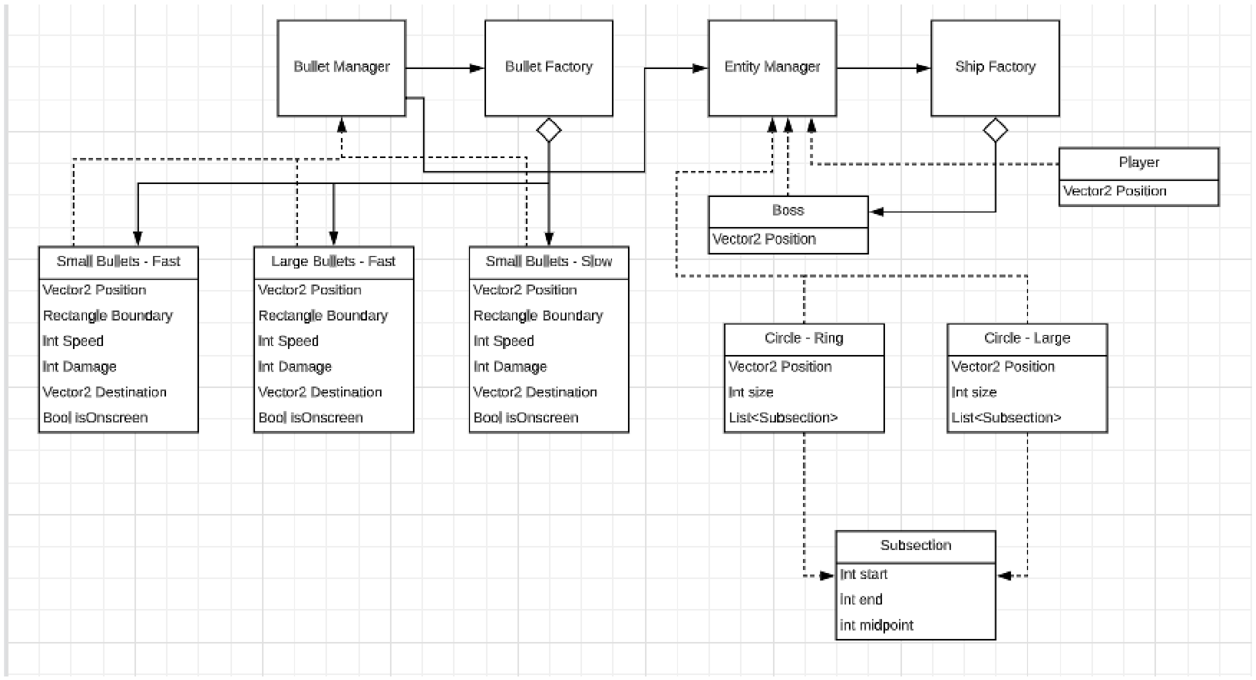
For the third phase, a list array of small bullets will be used to generate a barrage of bullets from the position the large bullet was in when its isOnscreen Boolean switched to false.

These small bullets will be given a randomized destination of any of the subsections of any of the rings or large circle.

*Classes Used:*



*Proposed Layout:*



Considering how the program is already built, we can utilize most of the structures already included. As it is currently set up, the entity manager is in charge of building the boss via the Ship Factory. Likewise, all bullets are controlled via the Bullet Manager, which, in turn, feeds information to the Entity Manager. Currently, there is just one type of bullet used in the build, so it would be a good idea to set up an abstract bullet factory along with concrete factories for each type of bullet (3 in this case).

Next, the circles have to be designed. The simplest maneuver would be to have the entity manager create and manage them; because the entity manager also tracks the boss, the circles could get the boss' position via this method instead of directly querying the boss object itself. If greater division is desired, a separate Enemy Manager could be created, and separate factories could be set up for the circles. Both the circles and the Boss would be managed by the Enemy Manager while the Enemy Manager feeds into the Entity Manager. The Enemy Manager would also be in charge of managing the ListArray of 5 bullets when the second phase begins.

Lastly, for keeping things simple, the Player should also be introduced to the Entity Manager, so that it could pass its position.