TurboStar Progress Update

Noah Hanson Oliver Nigaba John Krueger Jamie Vanoverschelde

***Abstract*— This document discusses the progress of Synergistic Solutions on their BulletHellShooter project. This document will discuss the progression of the project, successes, challenges, and what is yet to be accomplished. This document also contains samples of the unit testing that will be used in preparation for final testing of the project.**

I. Progress Summary

Since the last milestone, we have been working on more of the intricacies of the game to make it feel more like a playable game and less like a tech demo. The basic loop of the game is being evolved through progress on:

- Enemy movement
- Music
- Collision reactions
- Game over/settings screen

Keep in mind that most of these are still in their initial stages of progress, but they are currently being refined and implemented.

II. Group Summary

A. *What has been going well*

The game is continually evolving and getting better. Originally when an enemy ship was hit, a simple print line output “The enemy was hit!” However, now the enemy is taken out of the game. Not only are enemies eliminated, but we are experimenting with more enemy movements as well. Another task that we are proud of is the music production. The music for the game matches the game style very well.

B. *Challenges since last update*

The settings and game over screen are not functioning at an acceptable level. Research has been done along with peer collaboration in order to resolve this issue. We have also been experiencing issues with getting Gradle support for all group members within netbeans.

III. Work To Be Completed

Some of the to-do examples from our work include: Start screen, difficulty ramp, powerups, particles, and high scores. These will be the next details worked on before the final presentation is formulated.

IV. Unit Tests

```
@Test
public void testUpdateRight() {
    System.out.println("update for right");
    if (Gdx.input.isKeyPressed(Input.Keys.D)) {
        //GetSprite().setX(GetSprite().getX() + GetXSpeed());
        System.out.println("Right Key is being pressed!");
    }
    else{
        System.out.println("Right key is not being pressed");
    }
}

@Test
public void testUpdateLeft() {
    System.out.println("update for left");
    if (Gdx.input.isKeyPressed(Input.Keys.A)) {
        //GetSprite().setX(GetSprite().getX() + GetXSpeed());
        System.out.println("Left Key is being pressed!");
    }
    else{
        System.out.println("Right key is not being pressed");
    }
}
```

This is a snippet from our unit tests. In this test we are looking at the Update method within the playership class. These two unit tests look to see if the D or A key is being pressed and to tell the tester whether the input is being red.

We are currently in the process of further detailing these tests to apply to all classes within the program.