Software Implementation and Testing Document

For

Group <12>

Version 3.0

Authors(name, FSUID, GithubID):

Modibo Traore, mt16g, modibot

David Lee, dyl20b, dyl20b

Alejandro Osuna, ao18f, aosuna5861

Daniel Kovacs, djk19f, LivingBrovacs

Matthew Papageorge, mp20gu, mattpapa3

1. Programming Languages (5 points)

SuperLizard is written in C# and utilizes the Unity game engine.

2. Platforms, APIs, Databases, and other technologies used (5 points)

Unity game engine and Visual Studio.

3. Execution-based Functional Testing (10 points)

Tested how long cars stayed on the screen and what the appropriate time was to delete the car after it scrolled past the screen to avoid cars disappearing in front of the user. Cars would infinitely produce before but with testing realized that too many clones of the cars were produced and eventually ate up all the memory. This resulted in us making the clones into game objects so that we may delete them and still produce cars. After the initial run, it came to our knowledge that the lizard could traverse out of the playing field and come back in which was not desirable and resulted in boundaries being implemented. Cars initially moved by just shifting across the x axis but later was changed to include a forward direction implemented through the transform function which would allow us in the future to change directions of cars on a whim. Tested death screen by making sure when you die it turns to it and when you click new game on the death screen the score resets to 0 and the game restarts.

4. Execution-based Non-Functional Testing (10 points)

*Describe how/if you performed non-functional testing for your project (i.e., tested for the **non-functional requirements** listed in your RD).*

Tested the memory management of the program by making sure the cars would keep cloning themselves infinitely and no crashes would occur.

5. Non-Execution-based Testing (10 points)

Describe how/if you performed non-execution-based testing (such as code reviews/inspections/walkthroughs).

Made sure multiple people in our group were able to compile the code and run through the game. Originally a person in our group had a compiler error when trying to run the code, so we had other people test which we ended up finding out that it was a personal problem within their editor and not the program.