ENSE 400/477

University of Regina
Software Systems Engineering
Fall/Winter 2019/2020 Semesters

Code Quality Review

Capstone Project:
Stride Champions
Platform Fighting Game

Developers:
Jeremy Cross
Taylen Jones

Advisors:
Craig Gelowitz
Tim Maciag

This document is to serve as a code review for our final capstone project, to allow us to comment on the quality and state of our code. This document will follow the code review checklist suggested at: https://nyu-cds.github.io/effective-code-reviews/03-checklist/

General

- Does the code work? Does it perform its intended function, the logic is correct etc.

  Yes, our unreal code compiles. The logic in almost every case works as intended except with a few cases where we have some bugs.

- Is all the code easily understood?

  All of our blueprint code is commented and organized within the grid.

- Does it conform to your agreed coding conventions?

  Being that it was our first time using blueprint coding and unreal, the methods and placements of some of the code are probably not common. There is likely a better way to implement a lot of the functionality that we included within our game. As well there are probably other locations that are better for implementing some of our code. In some cases, during the beginning of development we would find better forms of implementation and we would go back and modify our code. However as our window of time for the project became shorter we had to pick certain forms of implementation for our functionality and just stick with them.

- Is there any redundant or duplicate code?

  There is some duplicate code with our program. Specifically within our character blueprints and level blueprints for the stages. The code for our characters was duplicated because we didn't start off with a base class that contained the basic core functionality that would be present among all characters. By the time we realized this, we were already far enough with the character development that it would have taken us a fair chunk of time to go back and implement. This was again mentioned in the question above that for certain things we had to just choose an implementation for and just go with it because our time for the project was running low. This is a similar case with the stage code where the code is the same across both stages. The difference here however is that some of the code could have been implemented in better locations. If we were to add more characters and stages to the game we would likely go back and fix these things because the number of case conditions required to cover everything becomes unmanageable.

- Is the code as modular as possible?

  Yes our code is modular. The blueprints contain all the necessary logic to drive their Functionality as intended.

- Can any global variables be replaced?

  We didn't use globals in our coding design. They don't really exist from what we could tell with blueprint scripting. The closest things we used as global variables were save files. These hold the values that are stored in them across all of the code in the game and even after the game is closed.

- Is there any commented out code?

  No, we cleaned up the code and removed any variables that were not being used.

- Do loops have a set length and correct termination conditions?

  Yes, all loops have closing conditions.

- Do the names used in the program convey intent?

  All of the variable names effectively convey the purpose they are being used for. Any variables that did not serve a purpose were removed. This applies the same for file names as well.

Performance

- Are there any obvious optimizations that will improve performance?

  Graphical and performance optimization could be implemented to make for a more enjoyable and smoother gameplay experience. Making the character attack animations transition better is something we would work on in the future.

- Can any of the code be replaced with library or built-in functions?

  Blueprint scripting uses a lot of built in functions. So yes we used many built in functions throughout our code.

- Can any logging or debugging code be removed?

  Any Logging and debugging code was removed from our final build when we packaged the game.

Security

- Are all data inputs checked (for the correct type, length, format, and range) and encoded?

  Most inputs for our code are set values. The only values that changed are the stock

count integer variables that we use to keep track of the player stocks. These are reset when a stage level is reloaded again. This logic has been tested and works as intended.

- Where third-party utilities are used, are returning errors being caught?

  The only third-party utilities used were some asset packs from the unreal marketplace. These include clothing for the characters, attack animation, sound effects and particle effects.

- Are output values checked and encoded?

  Yes, all output values are checked and work as intended.

- Are invalid parameter values handled?

  The only invalid parameter values we have in the logic are the actor variables in the Shared camera blueprint class. However these do not cause any issues with running the game. Everything else is handled.

Documentation

- Do comments exist and describe the intent of the code?

  Yes, all of the comments accurately describe the code that they're referring to. All of our blocks of code are commented across our program.

- Are all functions commented?

  Yes, all code is commented.

- Is any unusual behavior or edge-case handling described?

  No, all of the code works as intended except for a few bugs.

- Is the use and function of third-party libraries documented?

  No third-party libraries are used aside from asset content packs mentioned above.

- Are data structures and units of measurement explained?

  Yes.

- Is there any incomplete code? If so, should it be removed or flagged with a suitable marker like ‘TODO’?

  No, any loose hanging code or unimplemented code was removed from the project. All blueprint code remaining uses logic for the game.

Testing

- Is the code testable? The code should be structured so that it doesn't add too many or hide dependencies, is unable to initialize objects, test frameworks can use methods etc.

  Yes the code is testable. We used both white and black box testing to evaluate the functionality of our game. In the case of black box testing, we tested every action both players could do and marked down the results to check if they worked as intended. We did the same with white box testing except we tested the internal code to see if it functioned correctly for these same cases. We also tested certain cases with the stages as well.

- Do tests exist, and are they comprehensive?

  Yes, the tests exist and are comprehensive.

- Do unit tests actually test that the code is performing the intended functionality?

  We used both white and black box testing for our game, but yes they do test the functionality of our code.

- Could any test code be replaced with the use of an existing API?

  n/a