Lab 2

Prototype Product Description for

PolyMorpher

Prepared by: Peter I. Riley

CS 411

Professor Thomas J. Kennedy

19 March 2018

Version 2

Table of Contents

1 Introduction 4
1.1 Purpose 4
1.2 Scope 4
1.3 Definitions, Acronyms, and Abbreviations 4
1.4 References 6
1.5 Overview 10
2 General Description 10
2.1 Prototype Architecture Description 10
2.2 Prototype Functional Description 10
2.3 External Interfaces 10
2.3.1 Hardware Interface 10
2.3.2 Software Interface 10
2.3.3 User Interface 10
2.3.4 API Book Interface 10
2.3.5 Compiler Interface 10

Table of Figures

Figure 1 10

Figure 2 14

Figure 3 15

Figure 4 16

Table of Tables

Table 1 11

This space intentionally left blank

1 Introduction

Object oriented programming is an important concept for computer science students. It is a key concept for major programming languages such as C++ and Java. However, learning object oriented programming is a hurdle many Computer Science students face.

1.1 Purpose

PolyMorpher is a programming game meant to aid in teaching object oriented programming concepts to Computer Science students. The intended end users for PolyMorpher are students currently enrolled in a Computer Science degree program at Old Dominion University. Students who play PolyMorpher will ideally gain a better understanding of object oriented programming, which will aid them in passing freshman and sophomore level classes. Each level of PolyMorpher will focus on a different concept of object oriented programming.

1.2 Scope

Programming, especially object oriented programming, can be difficult and intimidating for the uninitiated, resulting in a trend of students dropping out or switching majors. Existing tools fail to adequately narrow the gap for students who have trouble comprehending these concepts. While many programming games available use object oriented concepts in their design, these games neglect or do not adequately explain object oriented programming.

PolyMorpher will guide players through object oriented concepts through the use of a puzzle-oriented platform game with a Tangible User Interface (TUI). Puzzles are a commonly used strategy for instructional games. A TUI will allow the user to see how code functions and how their changes affect the game world.

1.3 Definitions, Acronyms, and Abbreviations

API: Application Program Interface

API Book: A list of functions that the player can access to use in their code.

Assets: Any component, such as models, sprites, audio, and code, that is used in the creation of a game.

Code Compiler Directory: File directory holding the portable compiler, and the associated companion files, which are used to manipulate the scripts in the Streaming Assets Directory.

Coding Interface: An in-game GUI accessible to the player that pulls specified scripts from the Streaming Assets Directory for them to edit and compile using the portable compiler from the Code Compiler Directory.

Git: A version control system used to coordinate work among a group of people.

GUI: Graphical User Interface

LoadScripts.cs: Manages files accessed by the entire portable compilation system by identifying which script is currently in focus as the "source" script for any selected object in game, pulling this file from the Streaming Assets Directory and passing it off to the Script Bundle Loader for compilation.

Non-Technical Game: A game optimized for use by a non-technical user

Non-Technical User: A user who lacks formal education or knowledge in computer science or programming

ScriptBundleLoader.cs: Takes scripts passed off from the Load Script file and marks them for compilation, setting up the Assembler and Compiler and running the selected script through them. In the event of any compilation errors it will send out an error report through the Unity Error and Log files, otherwise it will attach the script to whichever game object was selected.

Streaming Assets Directory: File directory where all scripts accessible to the player via the in-game Coding Interface are stored and organized according to level and programming concept relevance. It is unique in that it is one of the few source file directories that are accessible to the player in the Unity Engine under any condition.

TUI: Tangible User Interface

Unity: A popular game development platform that will be used to develop PolyMorpher

This space intentionally left blank

1.4 References

Batten, C. (Narrator). (2017). CS410 Dungeon Escape Demo (Short Version) [Online video]. Online: YouTube. Retrieved from https://www.youtube.com/watch?v=ynhdd1IKgps

Batten, C. (Narrator). (2017). CS410 Project Dungeon Demo [Online video]. Online: YouTube. Retrieved from https://www.youtube.com/watch?v=ynhdd1IKgps

Batten, C. (2017, November 21). CS410 Tech Demo 2 (Download Source Code). In PolyMorpher. Retrieved from http://www.cs.odu.edu/~410silver/references.html

Batten, C. (2017, November 29). VersionControlFlow. In draw.io. Retrieved December 21, 2017, from https://www.draw.io/?state=%7B%22ids%22:%5B%221IQj6SYJqC6YLAK_qMRVIQkHiUmr9laBu%22%5D,%22action%22:%22open%22,%22userId%22:%22108692003133590583047%22%7D#G1IQj6SYJqC6YLAK_qMRVIQkHiUmr9laBu

Batten, C. (2017, October 26). CS410 Dungeon Escape Demo (Download Source Code). In PolyMorpher. Retrieved from http://www.cs.odu.edu/~410silver/references.html

Batten, C. (2017, October 26). CS410 Dungeon Escape Demo (Play Now). In PolyMorpher. Retrieved from http://www.cs.odu.edu/~410silver/references.html

"The Benefits of Video Games." abcnews (2011, December 26). Retrieved October 19, 2017, from http://abcnews.go.com/blogs/technology/2011/12/the-benefits-of-video-games/ Good-Morning-America

Edraw. (2017, May 12). Standard Flowchart Symbols and Their Usage. In Edraw Visualization Solutions. Retrieved from https://www.edrawsoft.com/flowchart-symbols.php

Everitt, C. (2017, September 6). Current Process Flow. In draw.io. Retrieved December 21, 2017, from https://www.draw.io/?state=%7B%22ids%22:%5B%220B-5KdQEdqLUPd nBFUnp2V05uMEE%22%5D,%22action%22:%22open%22,%22userId%22:%22108692003133590583047%22%7D#G0B-5KdQEdqLUPdnBFUnp2V05uMEE

Everitt, C., & Dang, D. (2017, September 24). currentProcessFlow. In draw.io. Retrieved December 21, 2017, from https://www.draw.io/?state=%7B%22ids%22:%5B%220B3Bc95zBWXg9TFZ6X0FMU1NTdEk%22%5D,%22action%22:%22open%22,%22userId%22:%22108692003133590583047%22%7D#G0B3Bc95zBWXg9TFZ6X0FMU1NTdEk

Everitt, C., Santos, K. & DeArce, N. (2017, November 27). Work Breakdown Structure (WBS). In draw.io. Retrieved December 21, 2017, from https://www.draw.io/?state=%7B%22ids%22:%5B%220B-5KdQEdqLUPWnNoSHhIUGg2OTQ%22%5D,%22action%22:%22open%22,%22userId%22:%22108692003133590583047%22%7D#G0B-5KdQEdqLUPWnNoSHhIUGg2OTQ

Everitt, C., Santos, K. & DeArce, N. (2017, October 13). ProcessFlowDiagram_silver. In draw.io. Retrieved December 21, 2017, from https://www.draw.io/?state=%7B%22ids%22:%5B%220B_xBnZ1ge4PlZTVjV3h6Y2pGSWc%22%5D,%22action%22:%22open%22,%22userId%22:%22108692003133590583047%22%7D#G0B_xBnZ1ge4PlZTVjV3h6Y2pGSWc

Few, S. (2008, February 5). Practical Rules for Using Color in Charts. In Perceptual Edge. Retrieved from http://www.perceptualedge.com/articles/visual_business_intelligence/Rules_for_using_color.pdf

Kennedy, T. (2017, September 6). kennedyData. In Google Drive. Retrieved from https://drive.google.com/drive/u/1/folders/0B_xCQd8Vk2BnSU1hNnJwSXB1NEE

O'Neill, M. (2017, March 6). Computer Science Before College. In Computer Science Online. Retrieved from https://www.computerscienceonline.org/cs-programs-before-college/

Riley, P. (2017, September 14). Using Games to Introduce Programming to Students [PowerPoint slides]. Retrieved from http://www.cs.odu.edu/~410silver/references.html

Stokes, J. (Narrator). (2017). CS410 Programming Game Pitch [Online video]. Online: YouTube. Retrieved from https://www.youtube.com/watch?v=QBvgzFgZaOQ&feature=youtu.be

Stokes, J. (2017, October 9). CS410 Programming Game Pitch (Download Source Code). In PolyMorpher. Retrieved from http://www.cs.odu.edu/~410silver/references.html

Santos, K., Riley, P. & Dang, D.(2017. December 7) Risk matrix and description tables in Design Presentation. Retrieved from https://docs.google.com/presentation/d/1oY9lkSAHvg2OIRkljYJNZWCqVTbiw45STKglsJUQjJI/edit#slide=id.g283e74317a_0_177

Unity Technologies. (2017, August 10). Company Facts. In Unity. Retrieved from https://unity3d.com/public-relations

Unity. (2016, July 6). Unity - Scripting API. In Unity. Retrieved December 21, 2017, from https://docs.unity3d.com/530/Documentation/ScriptReference/index.html

Unity. (2017, October 11). Asset Store. In Unity. Retrieved December 21, 2017, from https://www.assetstore.unity3d.com/en/

12 Free Games to Learn Programming. (2016, April 25). In Mybridge. Retrieved from https://medium.mybridge.co/12-free-resources-learn-to-code-while-playing-games-f7333043de11

This space intentionally left blank

1.5 Overview

This product description includes a general description of the PolyMorpher prototype. The information contains a description of architectural and functional components, and the external interface architecture built into the game.

2 General Description

The primary goal of the PolyMorpher prototype is to give a working demonstration of the core mechanics and capabilities of a game centered around object oriented programming. The prototype will be playable and teach the object oriented concepts of abstraction, encapsulation, polymorphism, and inheritance.

2.1 Prototype Architecture Description

The prototype will consist of three components:

- The PolyMorpher application: An executable download that launches the PolyMorpher game.
- The Unity file structure: The file structure for the PolyMorpher game will be included in the executable. The "Streaming Assets" folder will be accessed by the player to add and modify code for game objects.
- The PolyMorpher website: The executable download will be hosted on the PolyMorpher website. A guide to game rules and tips will also be available for users.

These components are shown in Figure 1.

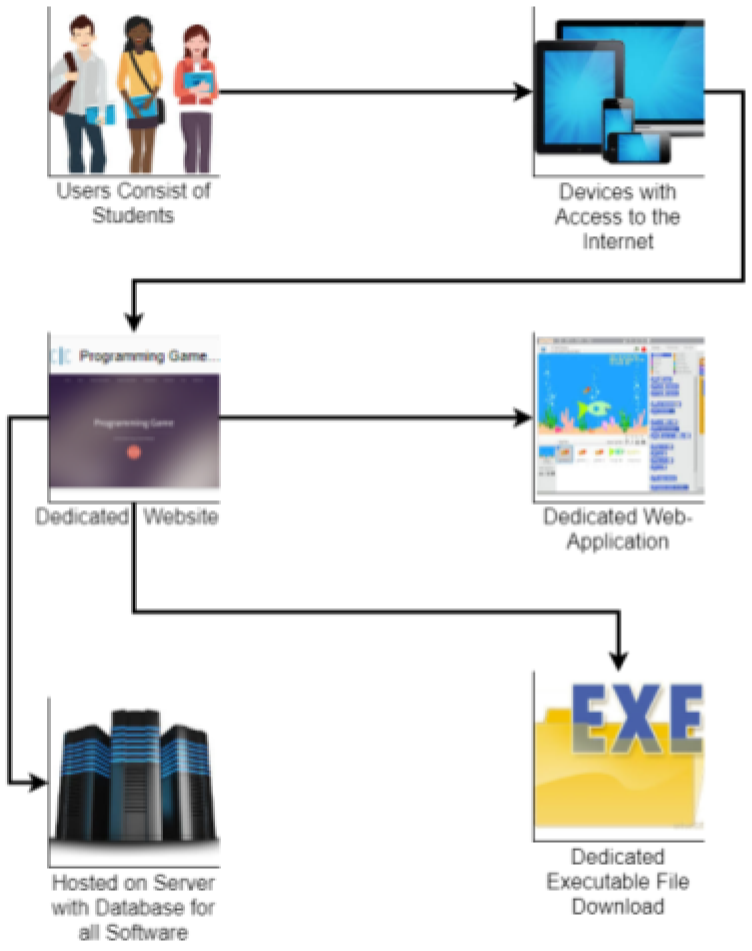

Users Consist of Students
Devices with Access to the Internet
Programming Game...
Dedicated Website
Dedicated Web-Application
Hosted on Server with Database for all Software
EXE
Dedicated Executable File Download

Figure 1 - Prototype Architecture

This space intentionally left blank

2.2 Prototype Functional Description

PolyMorpher will be compatible with Windows, MacOS, and Linux operating systems. The prototype and final product will be downloadable from the PolyMorpher website. Table 1 shows the features and differences of the prototype and final product.

Elements	Description	Real World Product	Prototype
Teaches Polymorphism	Provision of a single interface to entities of different types	🟩	🟩
Teaches Abstraction	Technique for arranging complexity of systems	🟩	🟩
Teaches Encapsulation	Building of data with the methods that operate on that data	🟩	🟩
Teaches Inheritance	When an object or class is based on another object or class, using the same implementation	🟩	🟩
Single Language Taught	A single programming language will be focused on C#.	🟩	🟩
Single Player	Focused on an experience targeted to interact with only one player	🟩	🟩
Downloadable .EXE File	Desktop application version of the game	🟩	🟩
Game Assets	Primary components that are used as building block to construct the more complex features and levels of the game	🟩	🟩
Developed Story	Narrative used to drive progression or direct player throughout a more guided/linear experience	🟩	🟩
Portable Compiler	Code compiler used to run player-made code on the fly in game	🟩	🟩

Elements	Description	Real World Product	Prototype
Tutorial Section	Precursor series of levels meant to help the player adjust to the in-game toolset given to them and also prep them with knowledge of the language(s) they will be working with	🟩	🟩
Multiple Platforms	Version support for multiple operating systems (Windows, Mac OS, Linux)	🟩	🟩
Sandbox Level	Open level where the player has access to all tools at once and can build their own level sequences and puzzles	🟩	🟨
Player-Made Content	Variant of Sandbox Level, potentially allows the player to share custom levels with one another	🟩	🟨
Multiple Player	An experience geared toward multiple players interacting with a game environment together	🟩	🟥
Web Application	Web based version of the game running in-browser	🟩	🟥
Multiple Languages Taught	Alternative programming languages for the player to use and learn in-game	🟩	🟥

KEY
🟩 Fully Functional
🟨 Partially Functional
🟥 Eliminated

Table 1 - Prototype and Real World Product Features

The main difference in features between the prototype and final product is the number of languages being taught and the web functionality of the game. The multiple language feature will be excluded because of time constraints. Multiplayer and web functionality will be excluded from the prototype because of the difficulty of secure implementation.

2.3 External Interfaces

There are five types of external interfaces for PolyMorpher: hardware, software, user, API book, and compiler.

2.3.1 Hardware Interface

PolyMorpher will be optimized to run on the user's machine without requiring a powerful graphics processor. The minimum requirement is a fourth generation Intel i3 processor.

2.3.2 Software Interface

The software hosted on the PolyMorpher site will compatible with Windows, MacOS, and Linux operating systems. The intent with this product is to make it cross-platform in order to reach a greater possible pool of users.

2.3.3 User Interface

The PolyMorpher game will be displayed through Unity's game interface on the user's computer. The user will be able to interact with the game through their mouse and keyboard. The keyboard will be used to move the player's character and type code in the game's built-in code editor. The mouse will be used to select objects for the user to see and edit their code.

2.3.4 API Book Interface

The API book will open when the user selects the API book button with their mouse. The functionality of the API book is described in Figure 2. If the player selects the API book, a menu opens that contains example code and player-created solutions. The player may choose one of these and the chosen code is inserted into the code editor for the player to use or change.

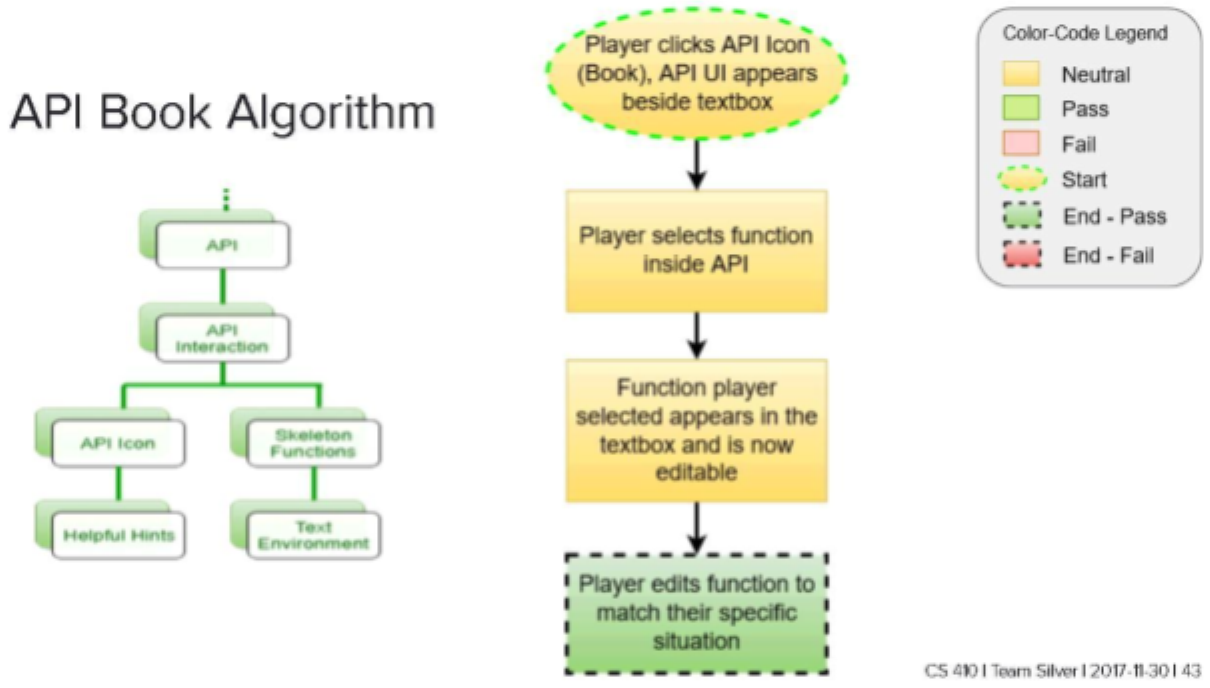
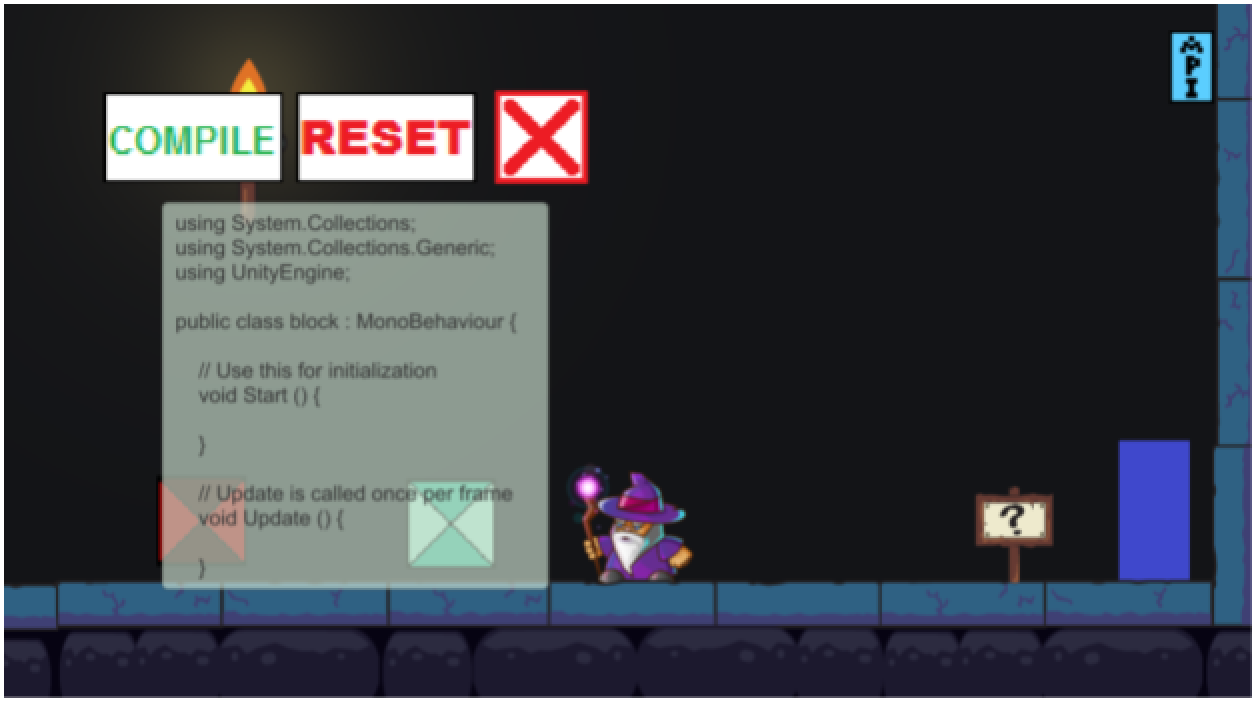

API Book Algorithm
API
API Interaction
API Icon
Skeleton Functions
Helpful Hints
Text Environment
Player clicks API Icon (Book), API UI appears beside textbox
Player selects function inside API
Function player selected appears in the textbox and is now editable
Player edits function to match their specific situation
Color-Code Legend
Neutral
Pass
Fail
Start
End - Pass
End - Fail
CS 410 | Team Silver | 2017-11-30 | 43

Figure 2 - API Book Algorithm

This space intentionally left blank

2.3.5 Compiler Interface

The compiler interface, shown in Figure 3, is opened when an editable object is selected by the player using the mouse. The compiler interface includes a space to input code that can be used to modify an object, a button to compile the code, a button to reset the object's code to a default setting, and a button to close the compiler interface.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class block : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
```

Figure 3 - Compiler Interface

When the player selects the compile button, as detailed in Figure 4, the compiler algorithm creates a runtime object with LoadScripts.cs attached to it. LoadScripts.cs puts the players code into the source code for the object and calls the compiler. The compiler checks for any errors. If there are errors, the player is returned to the code editor with an error message. If there are no errors, the selected object is recompiled with the new source code and the player is returned to standard gameplay.

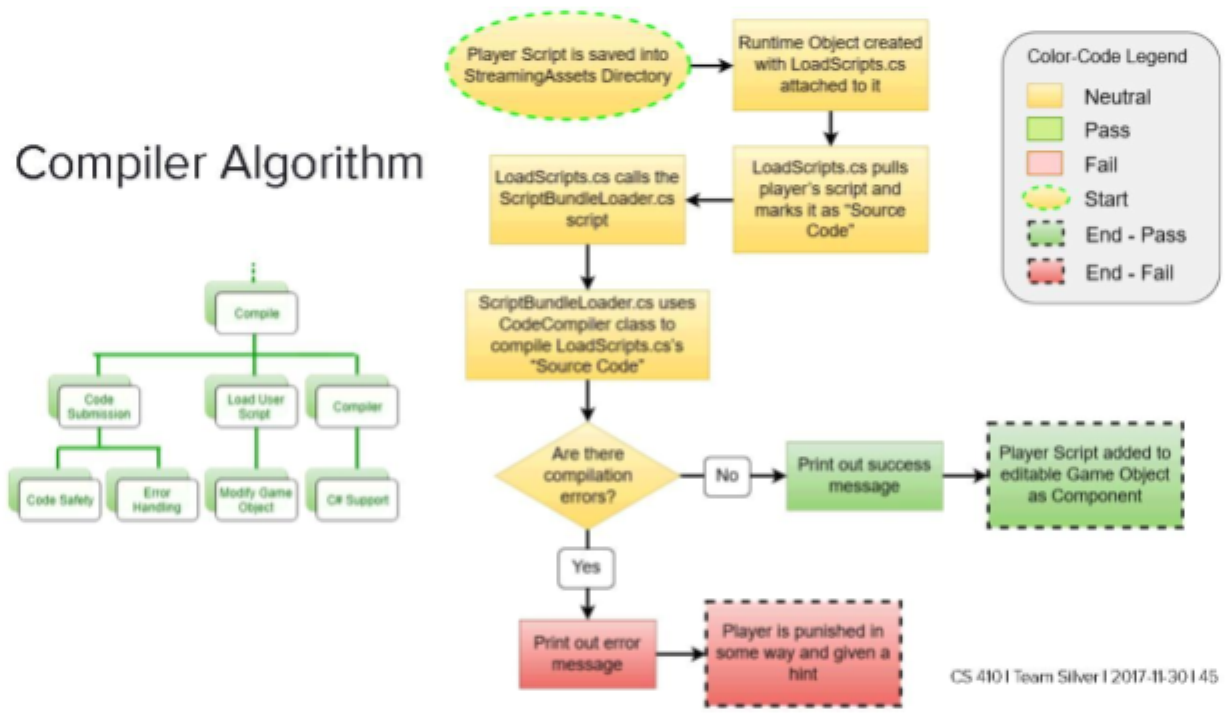

Compiler Algorithm
Compile
Code Submission
Load User Script
Compiler
Code Safety
Error Handling
Modify Game Object
C# Support
Player Script is saved into StreamingAssets Directory
Runtime Object created with LoadScripts.cs attached to it
LoadScripts.cs pulls player's script and marks it as "Source Code"
LoadScripts.cs calls the ScriptBundleLoader.cs script
ScriptBundleLoader.cs uses CodeCompiler class to compile LoadScripts.cs's "Source Code"
Are there compilation errors?
No
Print out success message
Player Script added to editable Game Object as Component
Yes
Print out error message
Player is punished in some way and given a hint
Color-Code Legend
Neutral
Pass
Fail
Start
End - Pass
End - Fail
CS 410 | Team Silver | 2017-11-30 | 45

Figure 4 - Compiler Algorithm

This space intentionally left blank