Lab 1- PolyMorpher Description

Peter I. Riley

CS 411

Professor Thomas J. Kennedy

23 January 2018

Version 1

Table of Contents

1 Introduction

Object oriented programming is an important concept for computer science students. It's a key concept for major programming languages like C++ and Java. However, learning object oriented programming is a hurdle that faces many computer science students.

1.1 Problem Statement

Programming, especially object oriented programming, can be difficult and intimidating for the uninitiated. Based on the enrollment data received from Professor Kennedy, this project's mentor, there is a notable drop off from CS 150 and 250 to 300 level courses like CS 330, 350, and 361, which is shown in figure 1. As also noted in figure 1, the data is subject to a margin of error because the enrollment data for Old Dominion University cited in this paper does not contain the declared major of each student, and CS 150 and 250 serve as prerequisite courses for other majors, including Engineering, Cyber Security, and Modelling and Simulation. However, even given the expectation that a number of students who take 100 and 200 level Computer Science courses are not required to take further courses for their major, there still remains a decrease in the number of students enrolled in 300 level Computer Science courses.

This space intentionally left blank

	CS 150	CS 250	CS 361	CS 330	CS 350
2013-2014	804	327	161	111	93
2014-2015	672	367	208	203	148
2015-2016	937	327	217	195	183
2016-2017	920	337	199	180	182

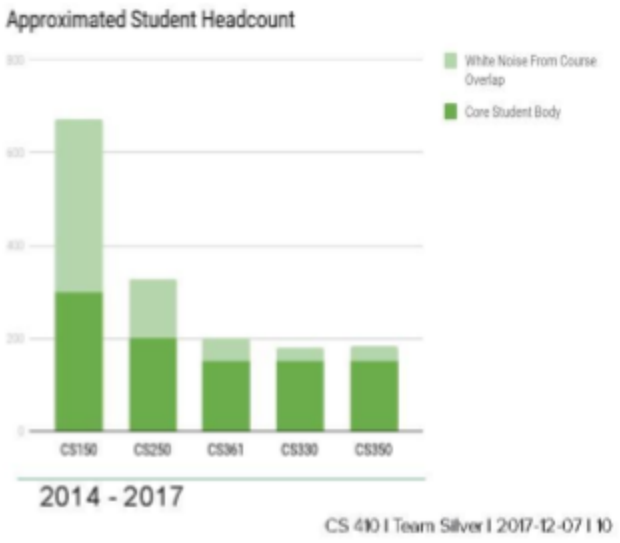

Figure 1 - Old Dominion University Approximate Student Headcount

This space intentionally left blank

1.2 Problem Characteristics

With the current situation, as outlined in figure 2, new Computer Science students can end up on a path to either pass or fail their introductory courses. If a student enters an introductory course and learns fundamental programming concepts over the course of that class, then that student is on course to pass their introductory class and progress in their Computer Science major. However, if a student struggles to understand programming fundamentals, that student could end up failing their introductory class and even dropping out or changing their major. Additionally, a student may end up having to retake an introductory class multiple times if they are still unable to learn the content being taught.

This space intentionally left blank

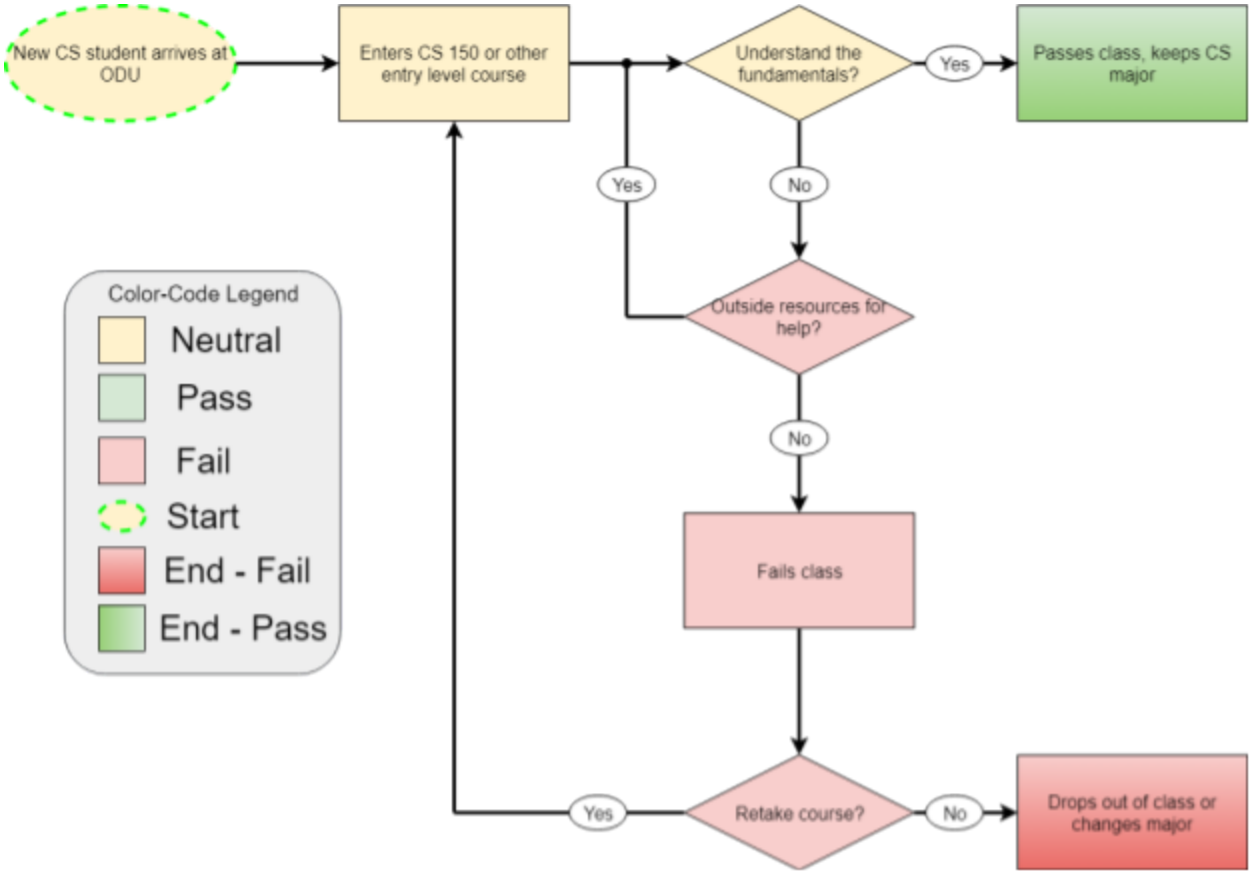

Figure 2 - Current Process Flow

This space intentionally left blank

1.3 Solution Characteristics

If the solution proposed in this paper, the PolyMorpher programming game, is used as a supplement to class instruction, a student who struggles to understand programming concepts from the lecture will be able to use the game to get a better understanding of those concepts. Use of the PolyMorpher game can help fill in gaps in knowledge and prevent a student from having to repeat a class.

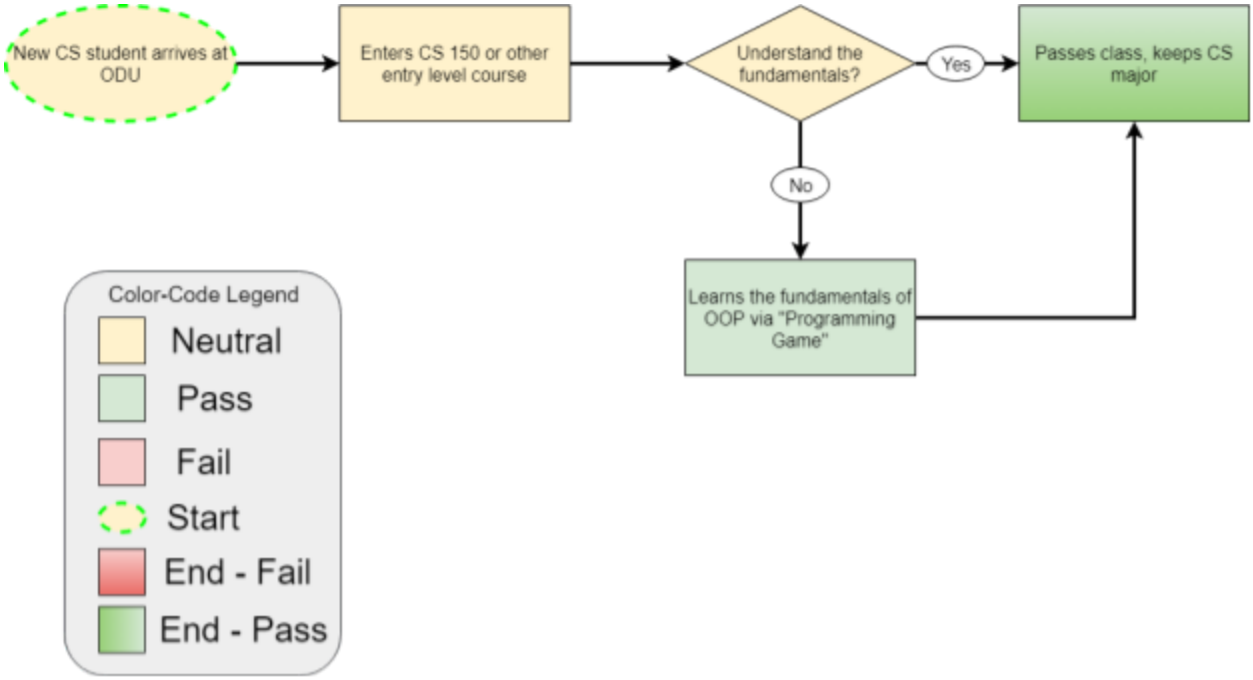

Figure 3 - Revised Process Flow

The idea to use a game as a medium for learning is not a new one, and there have been many games already made to teach programming. The data compiled in Figure 4 shows the characteristics of each game that this group felt were of note. The majority of games surveyed

were aimed at users with low to mid-range programming experience, which is defined as no knowledge of programming and beginner's knowledge of programming respectively. Most existing games also aim for this demographic. Object oriented concepts were separated into two categories, using and teaching, because many games included object oriented concepts but did not include any tutorials or instruction on those concepts. PolyMorpher will fill in this gap by focusing heavily on the object oriented concepts that are ignored in other games. Most games focused on one programming language. PolyMorpher will also only use one language initially, C#, with the possibility of adding one or more other languages in the future. The inclusion of multiplayer varied based on how each game implemented programming in its gameplay, and PolyMorpher will not have multiplayer because it would clash with the chosen method of gameplay.

This space intentionally left blank

Game	Experience	Uses OOP	Teaches OOP	# of Lang.	Multiplayer
PolyMorpher	Low-Mid	Yes	Yes	1	No
Code Combat	Low	Yes	No	5	No
Screeps	Mid-High	Yes	No	1	Yes
CheckIO	Low-High	Yes	No	1	Yes
Code Monkey	Low	No	No	1	No
Elevator Saga	Mid-High	Yes	No	1	No
Codewars	Mid-High	Yes	Yes	6	Yes
Codingame	Low-High	Yes	No	25+	Yes
Git Games	Low	No	No	1	No
CSS Diner	Low	No	No	1	No
Flexbox Defense	Low-Mid	No	No	1	No
Ruby Warrior	Low	No	No	1	No
Untrusted	Mid-High	No	No	1	No
Empire of Code	Low-Mid	Yes	No	2	Yes
Ruby Quiz	Mid-High	Yes	No	1	No

Figure 4 - Competition Matrix

2 PolyMorpher Product Description

PolyMorpher is a computer game made to teach object oriented programming to beginning Computer Science students.

2.1 Goals and Objectives

While PolyMorpher will teach general programming concepts, its focus will be on object oriented programming. Users will learn these concepts naturally by progressing through the game's levels.

2.2 Key Product Features

The gameplay in PolyMorpher will revolve around selecting objects and obstacles in the game environment and changing their code to progress through puzzles. The object code will be compiled during runtime to change the level while it is being played.

The levels in PolyMorpher will be designed to teach the object oriented concepts of encapsulation, inheritance, abstraction, and polymorphism. Each concept will have one level dedicated to it, along with a tutorial level to teach the gameplay mechanics.

2.3 Hardware and Software

PolyMorpher will be a 2D game and will be optimized to avoid requiring a powerful graphics processor. A fourth generation Intel i3 processor will be used as a benchmark for minimum hardware requirements. PolyMorpher will be distributed for Windows, MacOS, and Linux to be as accessible as possible.

3 Identification of Case Study

PolyMorpher will primarily be used in an education environment.

3.1 End User

PolyMorpher is being developed to help alleviate the enrollment drop from introductory to 300 level classes. So, students will be the primary end users; although, it will be available for anyone who wants to expand their programming knowledge.

3.2 Customers

Universities will be the customers for PolyMorpher, with Old Dominion University being the primary customer. There are no plans for a price at this time.

4 Product Prototype Design

The PolyMorpher prototype will consist of an executable program that the user will download. The prototype will contain almost all the functioning components that the real world final product would and is built around three major algorithms.

4.1 Prototype Hardware and Software

The prototype will consist of an executable that users will access by downloading it from a dedicated website onto their computer. The download website will be hosted on the Computer Science department servers.

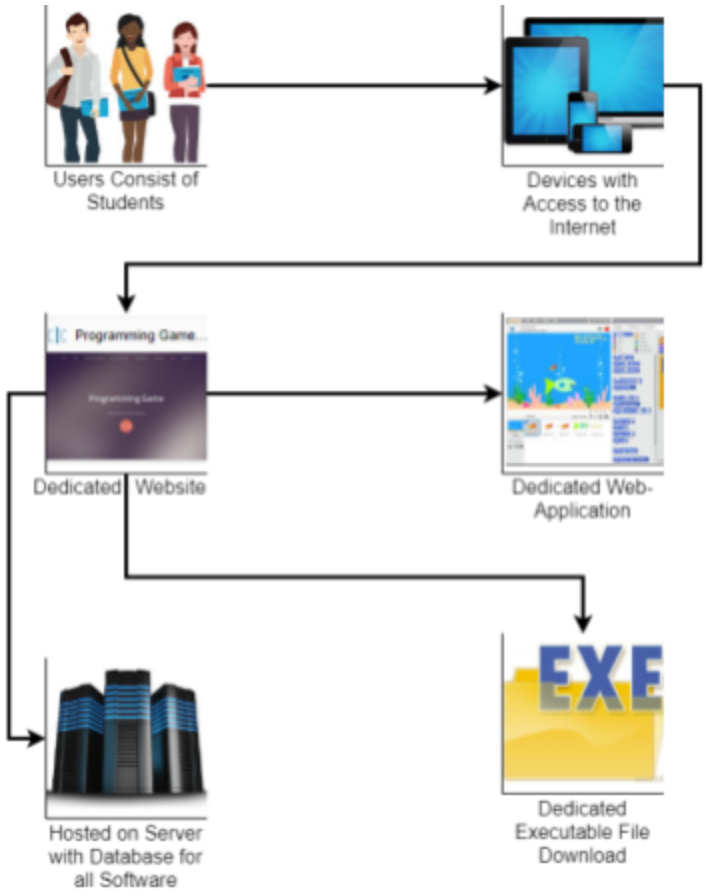

Figure 5 - Prototype Architecture

4.2 Prototype Features and Capabilities

The PolyMorpher prototype will contain most of the elements in the real world product, as shown in Figure 6. The sandbox level and player-made content are not vital to the prototype. So, these elements will be included if sufficient time remains after the rest of the prototype has been completed.

Multiplayer features and a web-based implementation of the game will not be included in the prototype because of the difficulty in implementation as well as the additional security features needed to prevent players from affect other players' computers.

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		
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		

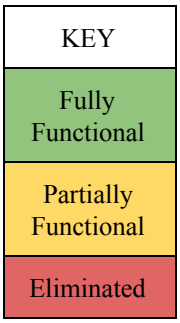

Figure 6 - Real World Product vs. Prototype

4.3 Algorithms

PolyMorpher is based around three major algorithms that allow the player to change the code of selected objects and recompile the game during runtime.

4.3.1 Core Algorithm

The core algorithm manages the other two based on the actions of the player. It launches when the player selects an object to change its code. The algorithm loads the source code for the selected object in a text editor. From there, the player can write their own edits of the code in the text editor, enter the API book algorithm to insert sample code, or click a reset button to reset the code in the editor to its original contents. When the player chooses to compile, the compiler algorithm is run. If the compiler algorithm passes, the code editor is closed and the player returns to the standard gameplay mode.

This space intentionally left blank

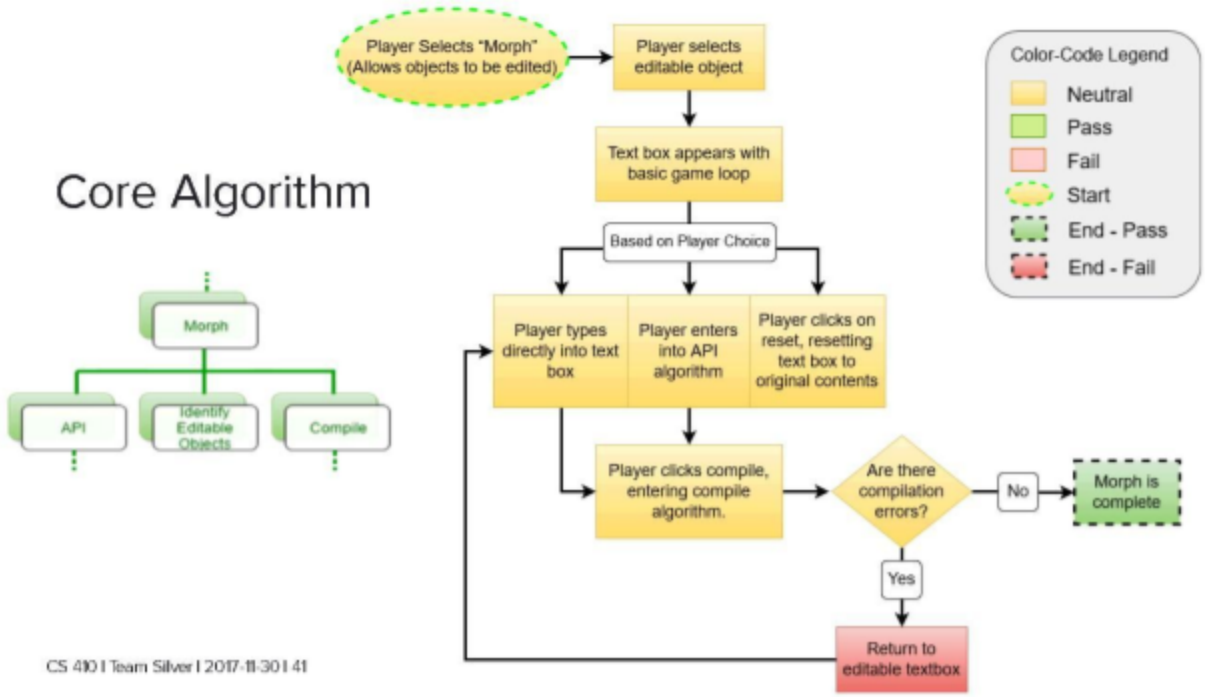

Figure 7 - Core Algorithm

This space intentionally left blank

4.3.2 API Book Algorithm

The API book algorithm is a straightforward algorithm. 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.

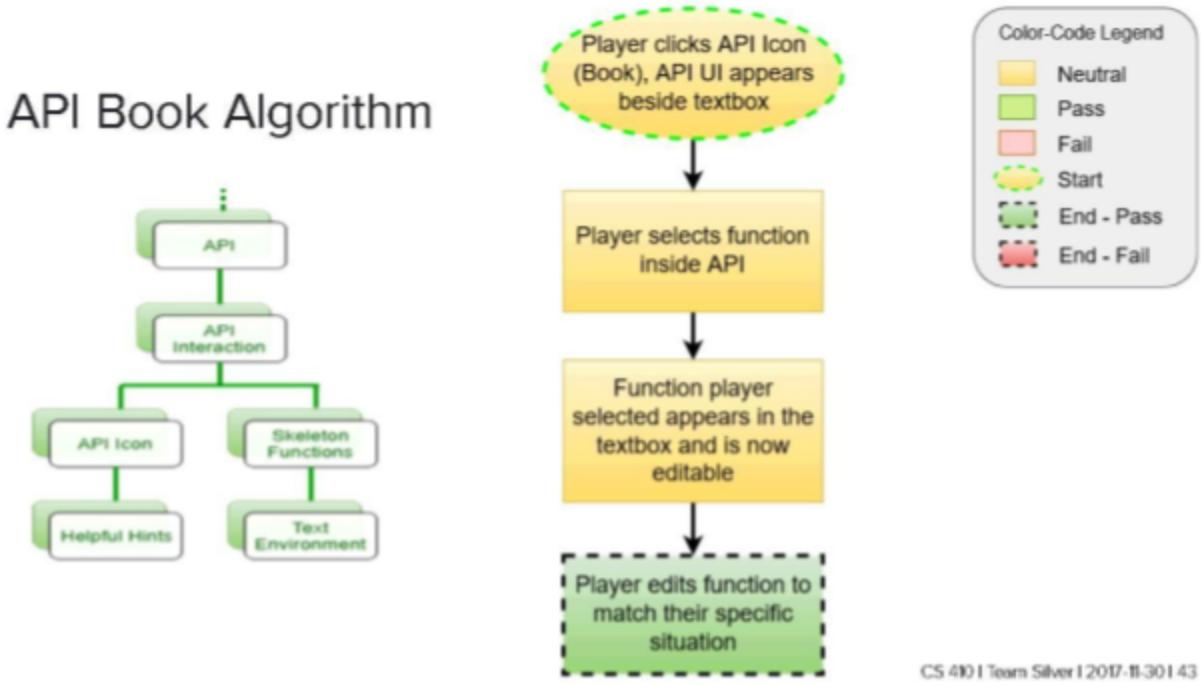

Figure 8 - API Book Algorithm

This space intentionally left blank

4.3.3 Compiler Algorithm

The compiler algorithm enables the game to change the source code of an object and recompile it during runtime. When the player selects the compile button, 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.

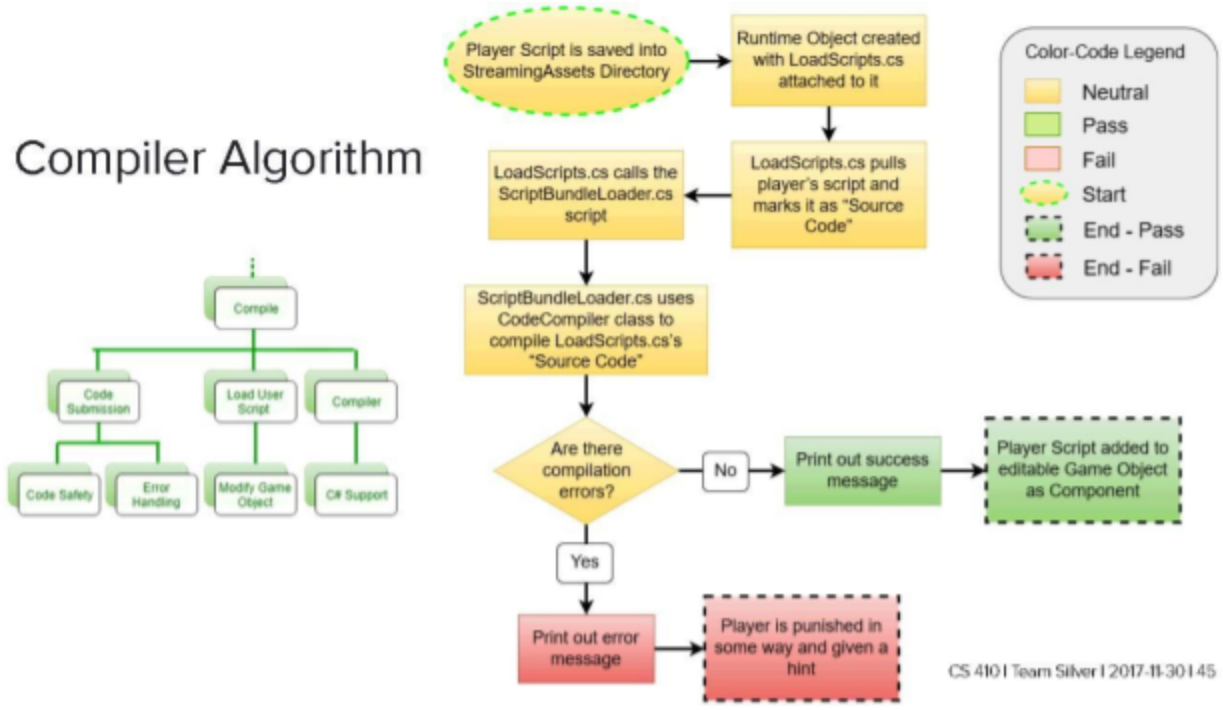

Figure 9 - Compiler Algorithm

4.4 Prototype Development Risks and Mitigation

Development of PolyMorpher has some risks, both with the customers and technical aspects, that will be addressed. These are shown in Figure 10. If the user gets lost in the game and cannot figure out the proper solution, that would defeat the purpose of the game. To mitigate this, the PolyMorpher game will go through thorough playtesting to fine tune the rigor of gameplay and helpfulness of the tutorials.

A user might be dissatisfied with PolyMorpher because of its gameplay or the player's inability to learn object oriented concepts. Mitigation for this will again be handled through playtesting.

A user may feel that the game is lacking in content or does not have proper pacing. Once again, this will be mitigated through playtesting. Playtesting is an important step in game design and helps to mitigate most customer related risks.

If the user implements unintentionally bad or intentionally malicious code, this could negatively impact the user's computer and possibly other computers. In order to mitigate this, the entire PolyMorpher game will be sandboxed. The game will have no access to other files on the user's computer. Also, the compiler will scan the player's code for potentially malicious or destructive scripts. There will also be a page on the development website dedicated to informing the player on possible threats and providing the player with a guide to safe solutions.

A user without sufficient hardware to run the game will prevent the game from helping the user at all. To prevent this, PolyMorpher will be designed to minimize graphics processing. The target minimum requirements are a fourth generation Intel i3 processor and 2 Gigabytes of RAM.

To prevent critical software bugs and insufficient API support, PolyMorpher will be playtested specifically to test functionality.

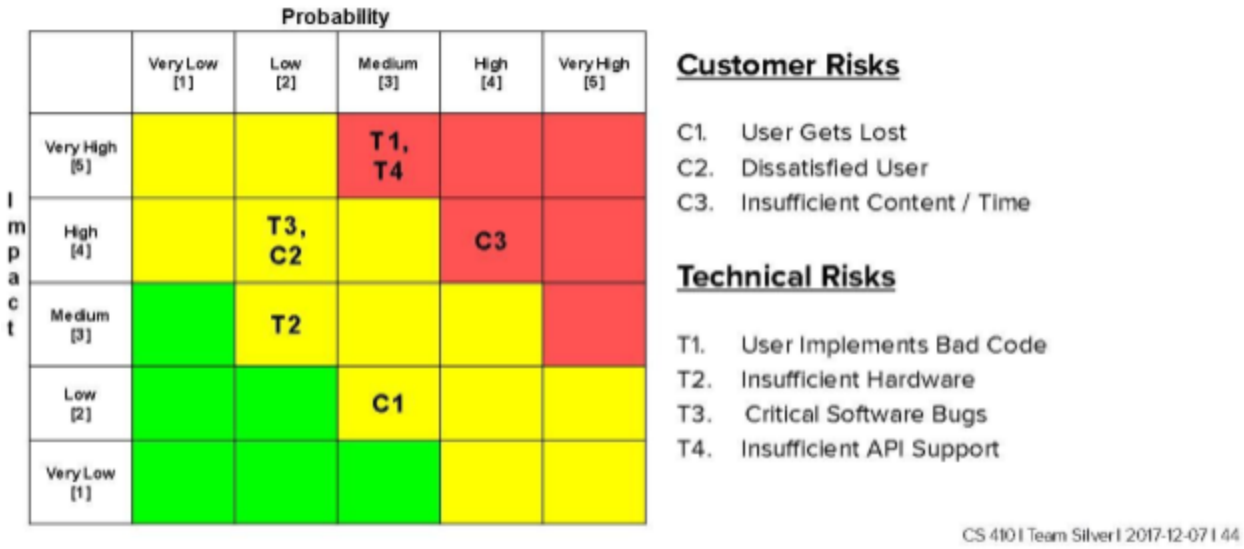

Figure 10 - Risk Matrix

5 Development Pipeline

PolyMorpher will be developed using the Unity game engine. Source Tree will be used to integrate git version control since Unity does not support it natively. Work will be separated into groups by level using an agile development model.

5.1 Unity Game Engine

PolyMorpher will be developed on the Unity game engine. Unity is a free game creation program commonly used in the video game industry. PolyMorpher will be coded with and teach C# because it is a powerful, widely-used language that Unity supports out of the box. Monodevelop will also be used in conjunction with Unity for C# development.

5.2 Source Tree Software

Source Tree will be used to connect Unity game files to the group's git repository because Unity does not have any built-in version control. Source Tree uses PuTTY to interface with the repository via SSH.

5.3 Version Control

The organization of version control for PolyMorpher is shown in Figure 11. All editing of the game code will be done through Unity. Source Tree will use PuTTY to interface with the git repository in order to push and pull changes. Each development team will have its own branch for the level they are working on. These will be merged with the main branch near the end of development to export the full game.

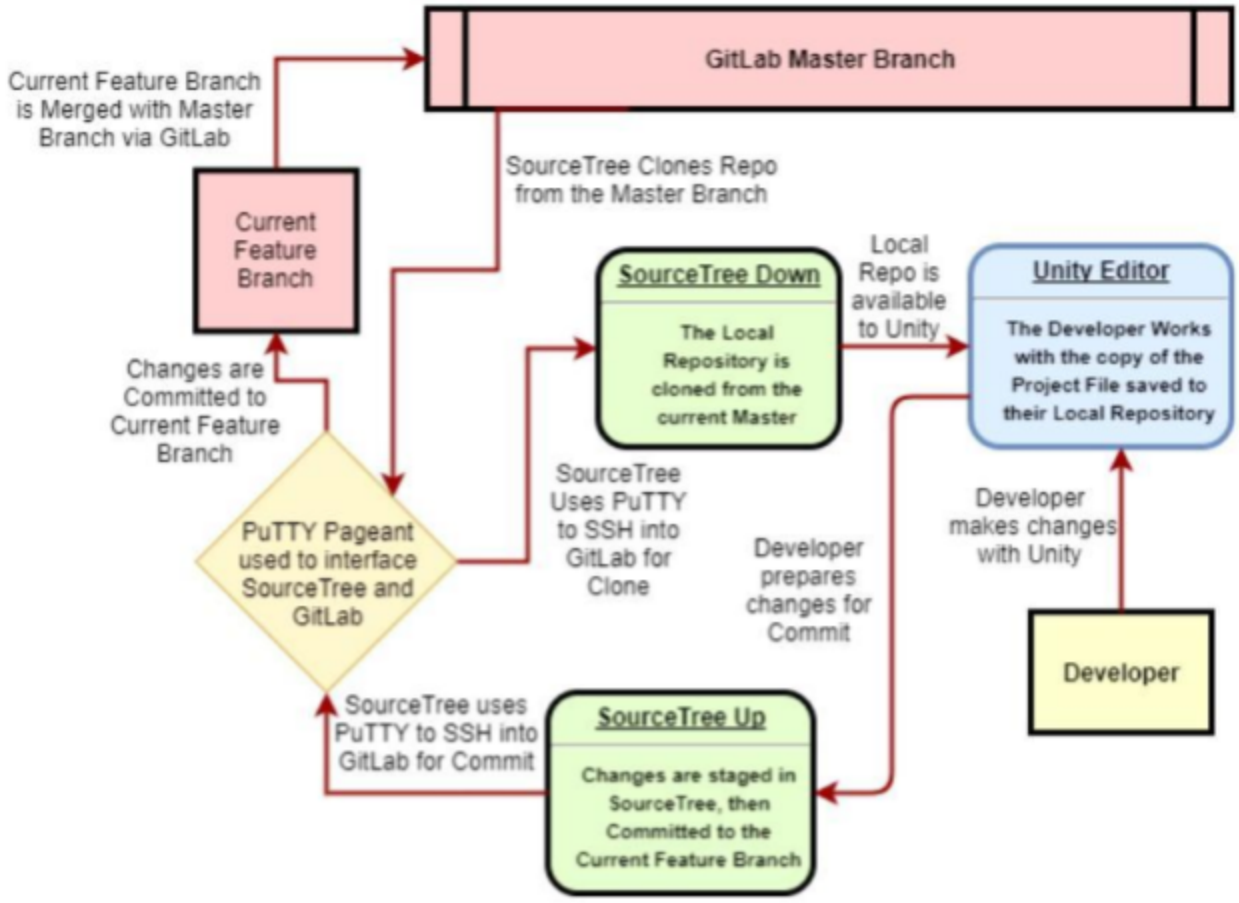

Figure 11 - Version Control Diagram

This space intentionally left blank

5.4 Agile Development

PolyMorpher will be developed using an agile development approach. Each group will work in short development cycles to create workable demos, from which elements of the game will be worked on, tested, and implemented.

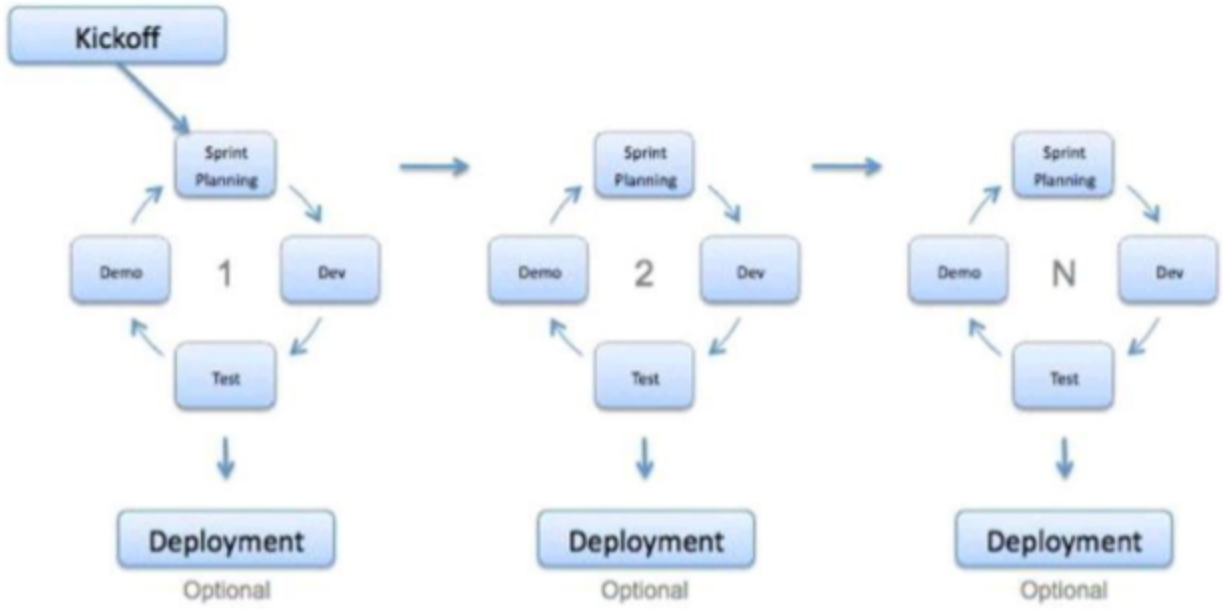

Figure 12 - Agile Development Model

5.5 Work Management

Development for PolyMorpher has been split into four groups. Each group will take one of the elements of object oriented programming (Encapsulation, Inheritance, Abstraction and Polymorphism), with the group working on inheritance also working on a tutorial level. Each

group will create a level based on the element assigned, and the levels will be tweaked later in development so they form one cohesive game.

6 Glossary

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.

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.

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.

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.

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.

This space intentionally left blank

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:%221086920031335905830 47%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:%221086920031335905830 47%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:%221086920031335905830 47%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