Lab 1 - PolyMorpher Product Description

Colten S. Everitt

Old Dominion University

CS411W

Professor Thomas Kennedy

29 January 2018

Version 1

Author Note

Colten S. Everitt, Department of Computer Science, Old Dominion University.

This research was done under the supervision and guidance of Thomas Kennedy.

Correspondence concerning this article should be addressed to Colten Everitt, Department of Computer Science, Old Dominion University, Norfolk, VA 23529.

Contact: cever015@odu.edu

Table of Contents

[This space intentionally left blank]

List of Figures

List of Tables

[This space intentionally left blank]

Lab 1 - PolyMorpher Product Description

1 Introduction

PolyMorpher is a programming game and outside resource solution for undergraduate (predominately first-year) Computer Science (CS) students at Old Dominion University (ODU). Object-Oriented Programming (OOP) and problem solving skills will be taught throughout the game in order to increase the probability of a CS student passing introductory CS classes, with the end goal of that student graduating with at least a bachelor's degree in CS. PolyMorpher is a downloadable executable file that is ran with a Management Simulator to a Tangible User Interface (TUI) for the player to interact with. The programming game was developed using the Unity Software Development Kit (SDK) with the C# and JavaScript programming languages.

1.1 Team Members

Under the guidance of mentor Thomas Kennedy, a group of college students from ODU formed the group "Team Silver" and developed PolyMorpher. The members of Team Silver are Colten Everitt, Matthew Tuckson, Kevin Santos, Casey Batten, Peter Riley, Joel Stokes, Daniel Dang, Tyler Brown, and Nathaniel DeArce. Colten Everitt is the website administrator. Matthew Tuckson is the program/project manager. The remaining team members each specialize in a different area of Unity SDK development.

1.2 Problem Statement

"Programming is intimidating for the uninitiated. As a result, first time ODU programming students drop out or switch majors. Existing tools fail to teach OOP concepts and problem solving skills" (Team Silver, 2017, 8). There is extensive evidence that a significant section of first time ODU CS students are struggling with programming. The professors are

failing to teach OOP concepts and problem solving skills effectively enough for students to truly understand them. These students might try to look for outside resources aside from what ODU provides, end up not finding any, and then fail classes, or worse case switch majors or drop out of college altogether.

1.3 Problem Characteristics

Figure 1 displays the current process flow of a new CS student arriving at ODU, taking an entry level course, and either passing the class and keeping the CS major, or dropping out of the class and changing their major from CS to something else. The two loops that the new student can get stuck in between entering CS150 or other entry level course, and either keeping or changing majors portrays the problem characteristics.

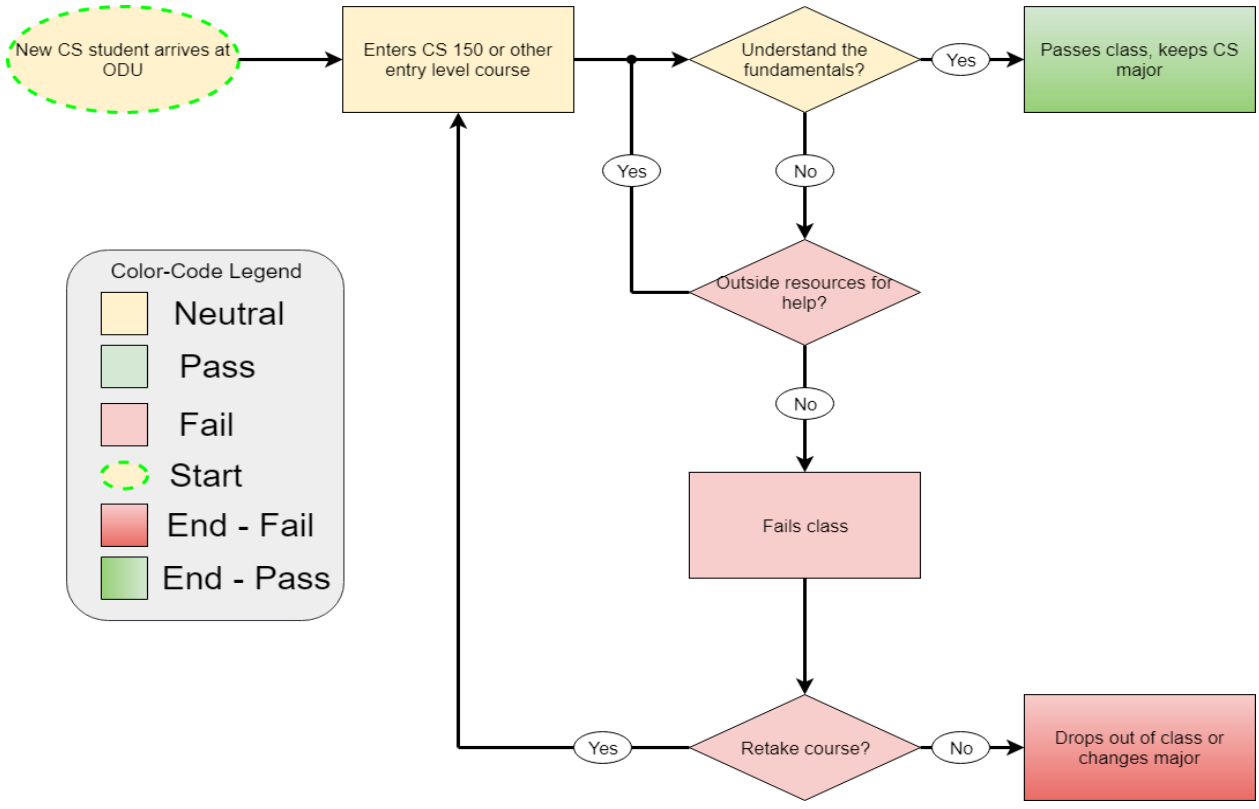

Figure 1: Current Process Flow (Team Silver, 2017)

As shown in Figure 1, if the new student does not grasp the fundamentals of programming and seeks outside resources for help, they could get stuck in the first loop. If the outside resources do not help the individual sufficiently enough to understand the fundamentals of the class or programming, then there will never be strong enough foundation knowledge to pass the class. On the other hand, if the new student does not seek outside resources for help then it is likely that he will also end up failing the class. The student then has the choice of retaking the course, or not retaking the course. If the course is not retaken, then the student will most likely drop out of the class or change their major. If the course is retaken, then the student will be stuck in the second loop of taking CS classes and failing them.

1.4 Solution Characteristics

The solution statement is: "PolyMorpher will address OOP concepts and problem solving through the use of a Management Simulator and a Tangible User Interface (TUI)" (Team Silver, 2017, 12). A Management Simulator, along with a TUI through a game application, has been shown to be the best way to teach OOP concepts and problem solving. In addition, games enhance interest among new learners. The basic nature of game interaction inherently gives players a more natural way to learn content (Team Silver, 2017). Games also change the learning style from traditional to more dynamic. One way this is accomplished is by not requiring an instructor to be present at all times.

Furthermore, games have a significant influence on learning. Poor academics and knowledge decrement lead to the stigma of video games being detrimental to the learning process. However, traditional learning through textbooks contributes to low engagement when compared to interactive media (Team Silver, 2017). According to the Office of Naval Research

(ONR), an average of 56 to 95 percent of people who play a particular game to learn a certain subject through tests demonstrated a better understanding of that subject (Team Silver, 2017). Thus, educational games keep the players engaged and promote enhanced concept learning. Due to the aforementioned facts, Figure 2 shows the solution process flow when PolyMorpher is introduced as a supplemental learning resource.

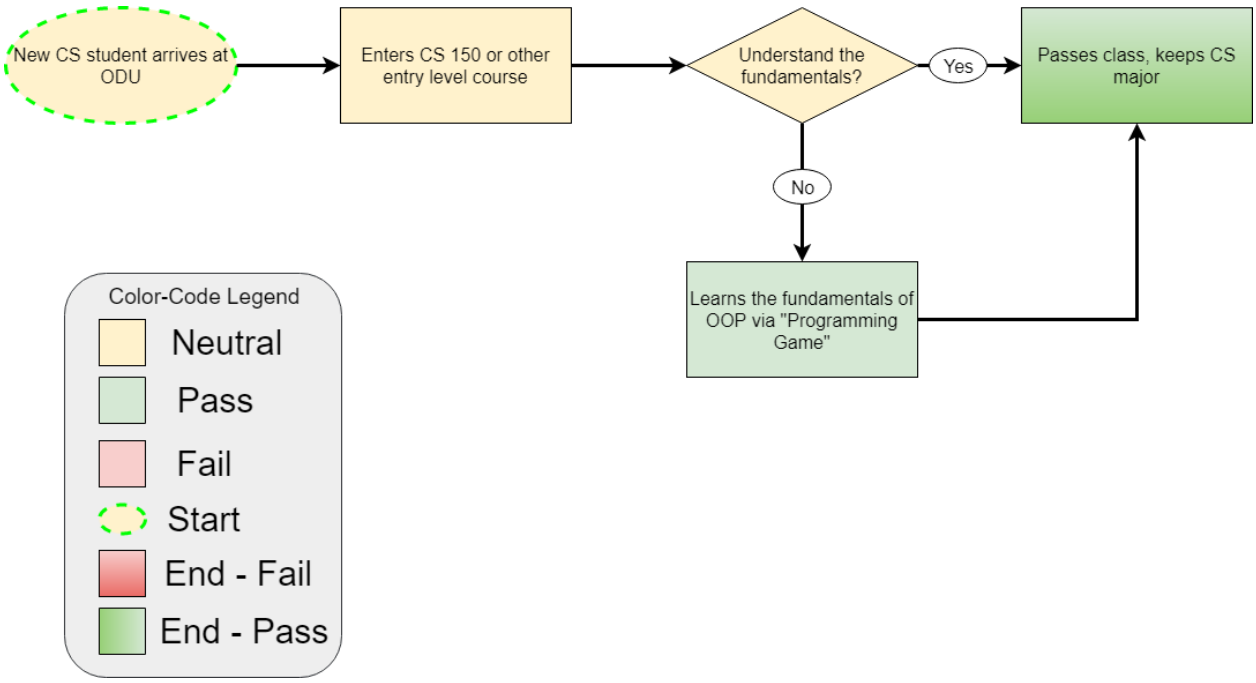

Figure 2: Solution Process Flow (Team Silver, 2017)

As shown in Figure 2, the outer loop of failing the class and having to choose to retake the class is eliminated. Once PolyMorpher is introduced, if the student does not understand the fundamentals of programming or the class, then they can learn and reinforce the concepts through PolyMorpher and pass the class while keeping the CS major. PolyMorpher essentially eliminates the likelihood of failing and having to make the choice of retaking the class, dropping out, or changing majors.

1.5 Current Solutions (Competition)

Today, most programming games will use OOP but not actually teach OOP to the players. This is a major problem with all of the current programming games available; CS and computer programming can not be fully learned without OOP concepts and problem solving skills. Table 1 and Table 2 display the various features that are provided by current programming games as opposed to PolyMorpher.

Game	Experience	Uses OOP	Teaches OOP	# Languages	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

Table 1: Competition Matrix - Part 1 (Team Silver, 2017)

Game	Experience	Uses OOP	Teaches OOP	# Languages	Multiplayer
PolyMorpher	Low-Mid	Yes	Yes	1	No
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

Table 2: Competition Matrix - Part 2 (Team Silver, 2017)

As shown from Table 1 and Table 2, the current solutions to programming games do not only require low experience levels, while both using and teaching OOP, and only focusing on one programming language. PolyMorpher will provide all these necessary features to create a programming game that will give students a solid foundation in CS.

PolyMorpher has distinct advantages over the other games listed in Tables 1 and 2. PolyMorpher is a game application that teaches users the fundamentals of computer programming and software development. In addition, this application will teach OOP concepts, problem solving skills, and a single language, be developed for multiple platforms, and have single player gameplay. PolyMorpher will be a standalone downloadable executable application that requires no WiFi or Internet connection. Gameplay for players with low programming experience will allow any type of player to play the game. One programming language will be implemented in the prototype, but more can be added easily in the future. Syntax will be a secondary objective, allowing more focus on OOP concepts and problem solving skills. Single player gameplay is the best option because multiple player interaction would sacrifice some crucial OOP concepts being grasped as well.

However, PolyMorpher also has a few disadvantages as compared to the other games listed in Tables 1 and 2. Most of PolyMorpher's competition games can only be played by players with middle to high level programming experience. If players have this higher level of experience, then they might like the competition's games better than PolyMorpher. In addition, the competition focuses on more than one programming language. This can be an advantage if players are already experienced in programming and would like to learn other languages. Other games also mainly teach syntax versus OOP concepts. Some brand new players to programming

might need this emphasis on syntax in order to then learn the more in depth concepts later on. Some players might like the multiplayer gameplay rather than the single player gameplay as well.

2 PolyMorpher Product Description

PolyMorpher is a programming game that runs off the player's machine and allows the player to play an engaging single player story, while learning the key concepts of CS. The gameplay offers a user-friendly TUI and incorporates all of the necessary aspects of modern computer-based games.

2.1 Goals and Objectives

PolyMorpher strives to set its concepts of gameplay and design choices above its competition. Implementing a more realistic approach to programming games will show players how programming works in real world applications. This can be done by using relatable and applicable examples throughout the game. PolyMorpher improves on actually teaching some of the more complex aspects of programming, like OOP and problem solving. These OOP concepts are easily explained by combining PolyMorpher's "Morph" and "API Book" functionality, as seen in Figure 3. Balancing gameplay and programming is a key goal, or objective, that PolyMorpher accomplishes. This simply means that the implementation of the gameplay will not sacrifice the player's experience.

[This space intentionally left blank]

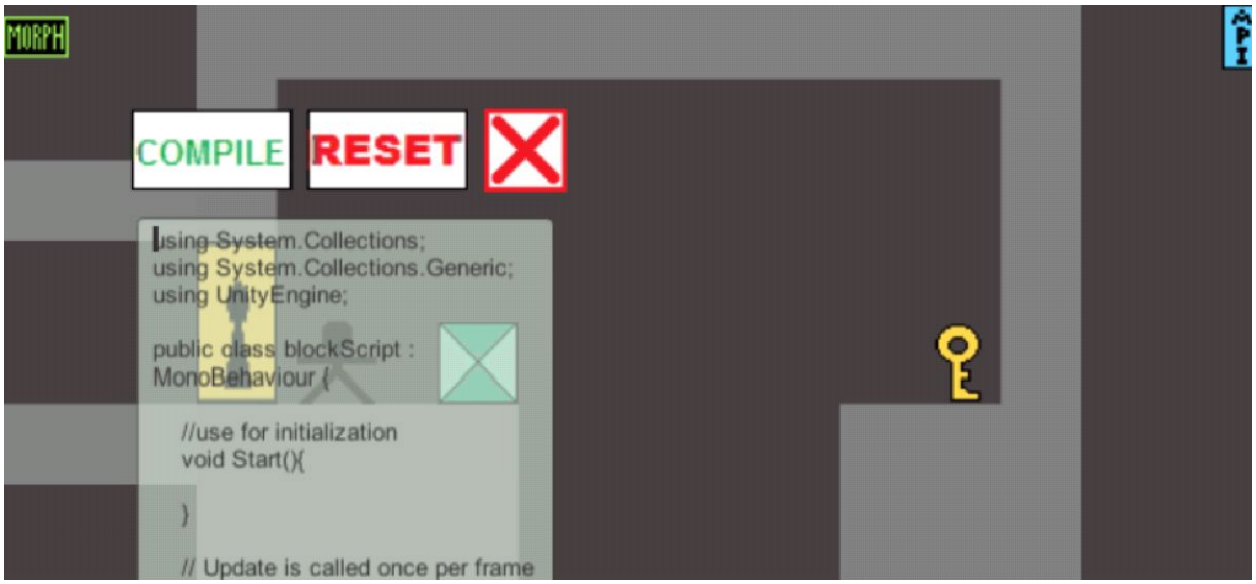

Figure 3: CS410 Technical Demonstration #2, By: Casey Batten (Team Silver, 2017)

2.2 Key Product Features and Capabilities

PolyMorpher has many features that contribute to making it one of the best programming game options. The gameplay is simple, but just complex enough to accurately teach OOP and problem solving. Professionally made vector art will be used, versus the pixelated art that is used in some of the competitions' products. Additionally, PolyMorper is a side-scroller style game rather than the top-down style game. A better quantity and quality of puzzles are worked into the gameplay in this style of game. Some of these key features and capabilities of PolyMorpher can be seen in Figure 4.

[This space intentionally left blank]

Figure 4: Key Features & Capabilities of PolyMorpher, By: Joel Stokes (Team Silver, 2017)

The game concept, or basic idea, of the game PolyMorpher is based off of the current solutions that students are already using. For example, all programming games have some kind of level structure, objectives/goals, player type, sound bar, health bar, game menu, story, direction, and help menu. Similarly, in PolyMorpher there ia a level structure, objectives/goals, single player type, sound bar, health bar, game menu, current method plan, and help menu. These features and capabilities can be seen in that order in Figure 5.

[This space intentionally left blank]

Figure 5: Structure of PolyMorpher (Team Silver, 2017)

As seen in Figure 5, all of the listed features from this competitor are used in PolyMorpher. PolyMorpher has a level structure built into it that starts out with the beginning levels consisting of purely English, without introducing any programming languages. In the middle levels, a pseudo-language like pseudo-code is introduced in order for the player to begin to learn OOP concepts and problem solving. In the final levels, the player should be up to enough speed to only be using the desired programming language in order to complete the game.

In addition, PolyMorpher has certain objectives (goals) that are kept track of in order to progress through the game. PolyMorpher is only a single player game because of the code compiling conflicts with it being multiplayer. There will be one sound bar implemented to allow the player to adjust game volume. Each player will have a health bar to determine if a player takes enough damage and dies, essentially making the player lose the level and have to restart. A game menu allows the player to navigate through the game. It allows the player to enter, exit, save, load, or pause the game. The current method plan is a prompt that is given so that each

player knows what actions need to be performed, as well as what lines of code need to be typed into the game. Lastly, there is a help menu that pops up if a player is struggling with getting through a certain task or level. It has a list of different potential options of code input that might work to solve the puzzle.

2.3 Major Functional Components (Hardware/Software)

The major functional components for PolyMorpher are users, devices, website, executable file, and server with database for all software. First, a single user or player can play the game; usually users will be students. The player can play the game through a compatible device, a desktop or laptop computer with access to the Internet. The player can navigate to PolyMorpher's dedicated website, go to the Download tab, and then click the big Download button to download the entire game as an executable file. How PolyMorpher is accessed in order to be played is displayed in Figure 6.

[This space intentionally left blank]

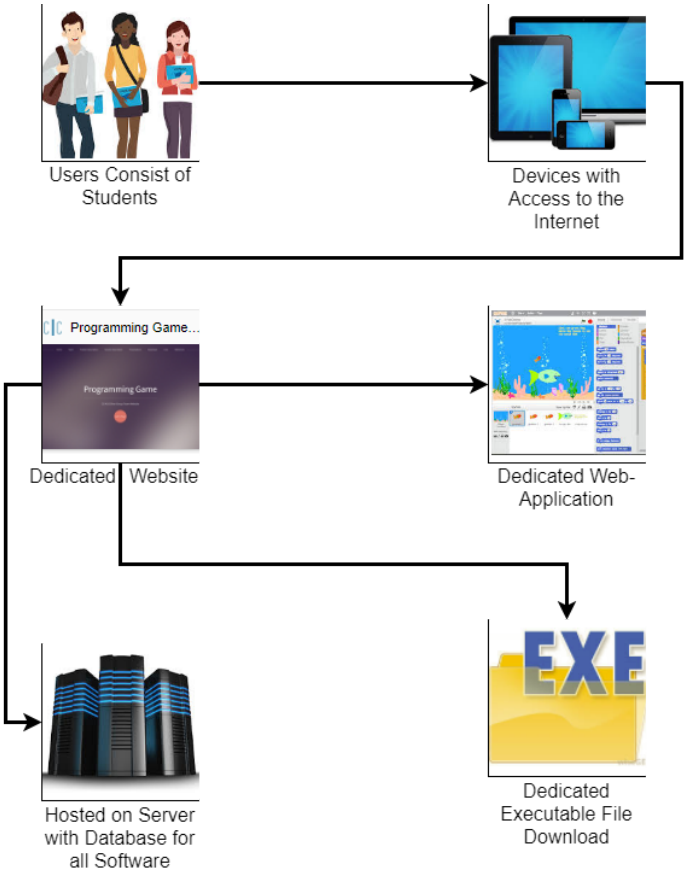

Figure 6: Major Functional Components (Team Silver, 2017)

The player will require a compatible device with the necessary hardware and software specifications to operate the game. Also, all of the software behind PolyMorpher is hosted on the CS server at ODU.

There are specific specifications pertaining to the hardware and software of the computer in order to run the game. For the specific hardware the player has to have to run the game, a fourth generation i3 Intel Processor is the minimum requirement. With PolyMorpher being a two-dimensional game style, it will have minimal draw on the computer's resources. The specific software in order to run the game is Windows 7, 8, 10, or any Linux or MacOS

operating system. The software required to run the game is cross-platformed so that way basically anyone in the world can play it.

Moreover, the physical game of PolyMorpher has to be downloaded from the PolyMorpher website as an executable file. PolyMorpher was going to be available as a web-application and an executable downloadable file. Both ways could have been accessed on the website as different tabs. However, PolyMorpher has to be only a downloadable executable file because of the way that players physically change the source code of each game to essentially beat the game. If PolyMorpher was a web-application as well, then players could potentially change the source code of the game that other players could potentially download.

3 Identification of Case Study

PolyMorpher is a responsive, educational programming game made with the Unity SDK, built with C# and JavaScript, and released on behalf of ODU. It is perfect for middle and high school teachers and students as an introduction to programming concepts. PolyMorpher is also readily available to beginning programmers outside of a school environment. It was created in order to introduce programming to new students, while minimizing how intimidating coding can be.

3.1 End Users: Students

The initially targeted end users for PolyMorpher are students who are currently enrolled in the CS degree program at ODU. Since the game was made by a team of students currently in the CS degree program at ODU, the game would be partially owned by ODU and therefore sold by ODU. After students at ODU begin to utilize PolyMorpher, ODU might make the decision to sell the game to other universities, colleges, or educational institutions. PolyMorpher is also

made for anyone who is generally interested in programming, outside of the CS department or at ODU.

3.2 Target Customers: Old Dominion University

Similar to the end users, the initially targeted customer for PolyMorpher is ODU. Since the game was made at ODU by students in the CS department, ODU will be the first targeted customer. PolyMorpher could also be sold to other universities, colleges, or educational institutions that currently offer a CS degree program. Middle or high school teachers might be interested in buying a license to PolyMorpher for their classes. PolyMorpher is also targeted towards anyone seeking to gain more knowledge in computer programming, OOP concepts, and problem solving skills.

3.3 Reasons for Creating PolyMorpher for Students

Recent statistics show that students in the CS degree program at ODU are in decreasing class sizes as they progress from the lower level to the upper level CS classes. Some of this decrease in class size may be due to the fact that some of the lower level CS classes are required for other majors as well as CS wherein the higher level CS classes are only required for CS degrees. However, there is still a significant decrease in class size when the focus is only on CS students. PolyMorpher is going to help balance and lessen this decrease in class size as CS students progress through their CS degree.

Reiterating the Problem Statement, students are in desperate need of a supplemental resource that will teach them the programming skills needed to pass CS classes at ODU. Once PolyMorpher is introduced to students, there will be a drastic reduction in

situations where they do not understand the basic fundamentals of CS, fail CS classes, drop out, or switch majors.

In addition, there is a "Student Progression Dilemma" at ODU. Students are not following the CS course series in the expected order. The collected data provides purpose behind the need to reassess methods of assisting students' progress in how they learn the topics at hand. The provided data also shows exactly when during the learning process that this assistance would be most effective. Changes in class volume could indicate that students are leaving the CS major for less programming intensive fields (Team Silver, 2017).

Furthermore, there is a significant decline in enrollment of students who progress from the classes CS150 to CS250, and then continue on to the classes CS330, CS361, and CS350 at ODU. This decline in enrollment may be directly related to differing major requirements and course overlap. However, the decrease in student body is still significant enough to warrant a deeper look into later classes in the major path. In addition, decreasing class sizes show a steady decline in CS course enrollments as course level difficulty advances (Team Silver, 2017). These decreases may be indicative of students either dropping out of the CS program or changing majors. The Student Progression Dilemma statistics from the ODU Factbook are displayed in Figure 7 as a pie graph.

[This space intentionally left blank]

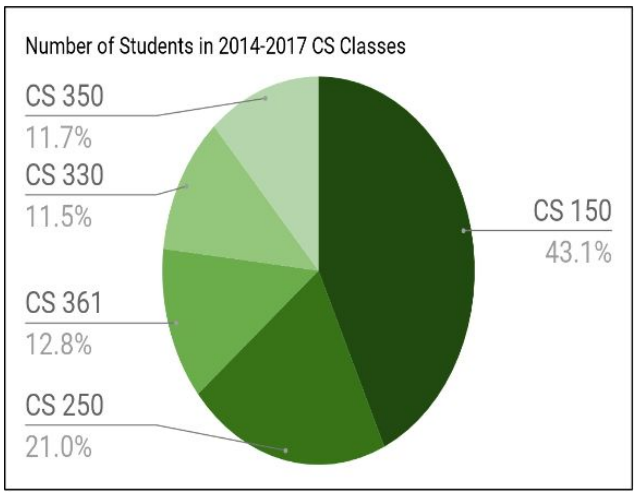

Figure 7: Student Progression Dilemma - Pie Chart (Team Silver, 2017)

Figure 7 further shows the breakdown of the total number of students from 2014 to 2017 in CS classes at ODU. According to the ODU Factbook, from 2012 to 2016 the number of undergraduate CS majors increased from 284 to 429 (Team Silver, 2017). This shows the high demand in the CS degree path at ODU. From 2014 to 2015 there were roughly 672 students enrolled in CS150, compared to only about 327 students enrolled in CS250 from 2015 to 2016 (Team Silver, 2017). In the years of 2016 to 2017, there were roughly only 199 students enrolled in CS361, 180 students enrolled in CS330, and 182 students enrolled in CS350. Figure 7 displays these statistics in the form of percentages to emphasize the impact of decreasing course sizes.

The "white noise" of students progressing from the classes CS150 to CS250 at ODU has to be accounted for. There is a less dramatic decrease in course sizes than initially thought from the CS courses that are requirements for other majors. For example, CS150 is the first course that CS majors have to take and is considered a "service course" by ODU. This means that it is also required to be taken by Physics, Math, Engineering, and Mod-Simulation majors (Team Silver, 2017). CS250 is required to be taken by CS, Mod-Simulation, and Computer and

Electrical Engineering majors. CS330 is required to be taken by CS and Mod-Simulation majors. CS361 is required to be taken by CS and Computer and Electrical Engineering majors. Lastly, CS350 is required to be taken by CS and Computer Engineering majors (Team Silver, 2017). Table 3 further displays the Student Progression Dilemma taking into account the "white noise" of overlapping different majors.

	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

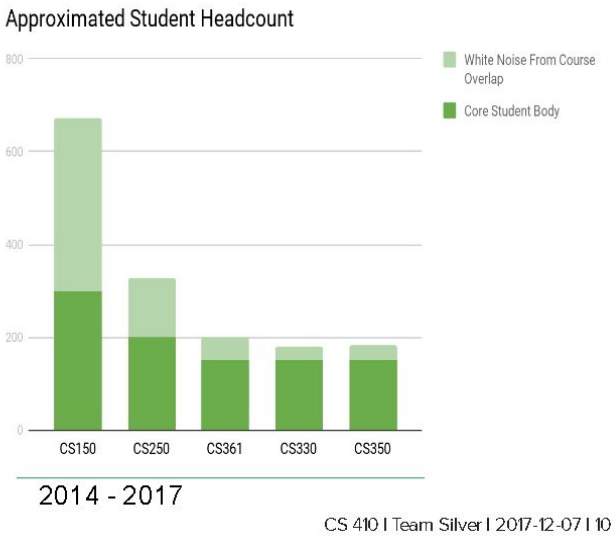

Table 3: Student Progression Dilemma - Table & Graph (Team Silver, 2017)

As shown from Table 3, the approximated student headcount of the core student body for CS majors is significantly less than when also considering the "white noise" from course overlap. Again, the values with the "white noise" from course overlap are shown in Figure 7. The core student body from 2014 to 2017 consists of approximately 300 students in CS150, 200 students in CS250, 175 students in CS361, 180 students in CS330, and 178 students in CS350. As evidenced above, only CS150 and CS250 are drastically affected by the "white noise" from course overlap.

[This space intentionally left blank]

4 PolyMorpher Prototype Description

The primary goal of the PolyMorpher prototype is to provide a working demonstration of the PolyMorpher product. This is accomplished by implementing only the necessary components and features of the full product.

4.1 Prototype Architecture (Hardware/Software)

PolyMorpher prototype architecture will be identical to that found in the completed product.

4.2 Prototype Features and Capabilities

The PolyMorpher prototype will be compatible with virtually all operating systems: Windows, Linux, and MacOS. Both the completed product and the prototype will be able to be downloaded off the PolyMorpher website.

A summary of the PolyMorpher prototype deliverable features can be seen in Tables 4 and 5 below. The primary differences between PolyMorpher's completed product and its prototype are displayed in Table 5.

Table 4: Key for Features of the Complete Product versus the Prototype (Team Silver, 2017)

[This space intentionally left blank]

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		
Player-Made Content	Variant of Sandbox Level, potentially allows the player to share custom levels with one another		
Sandbox Level	Open level where the player has access to all tools at once and can build their own level sequences and puzzles		
Multiple Languages	Alternative programming languages for the player to use and learn in-game		
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		

Table 5: Features of the Complete Product versus the Prototype (Team Silver, 2017)

Table 5 shows that the prototype will not be implementing multiple programming languages as alternative programming languages for the player to use and learn in-game. It will not be implementing the multiplayer gameplay feature of experience being geared toward multiple players interacting with the game's environment simultaneously. The prototype also will not be implementing the ability to access and play the game through a web-application; instead, a downloadable executable file will be available for game access. The features that will only partially be implemented are the player-made content, along with the sandbox level. These features allow the player to share custom levels with other players. A sandbox level is defined as an open level where the player has access to all tools at once and can build their own level sequences and puzzles. All else will be fully implemented.

Algorithms are the foundation that make up the PolyMorpher prototype features and capabilities. All of the puzzles that teach OOP and problem solving concepts are created and implemented with the algorithms used in the game. The software iterations of the algorithms used in the game are Core, API Book, and Compiler Algorithms. Extensive game testing will also be done in order to insure that the game is ready for the players.

To begin, the Core Algorithm is used as an overview of the other two algorithms that make up the PolyMorpher "Morph" function: API Book Algorithm and Compiler Algorithm. The Core Algorithm is a UI level algorithm that provides the functionality to modify the identified editable objects in the game. It determines the base tools at the player's disposal. It also divides the gameplay into distinct sections in order to move past levels and beat the game (Team Silver, 2017). The distinct sections include: interaction through UI elements/object selection, information provision through access to the API Book Algorithm, and alteration through the Compiler Algorithm. The dataflow algorithm for the Core Algorithm is displayed in Figure 8.

[This space intentionally left blank]

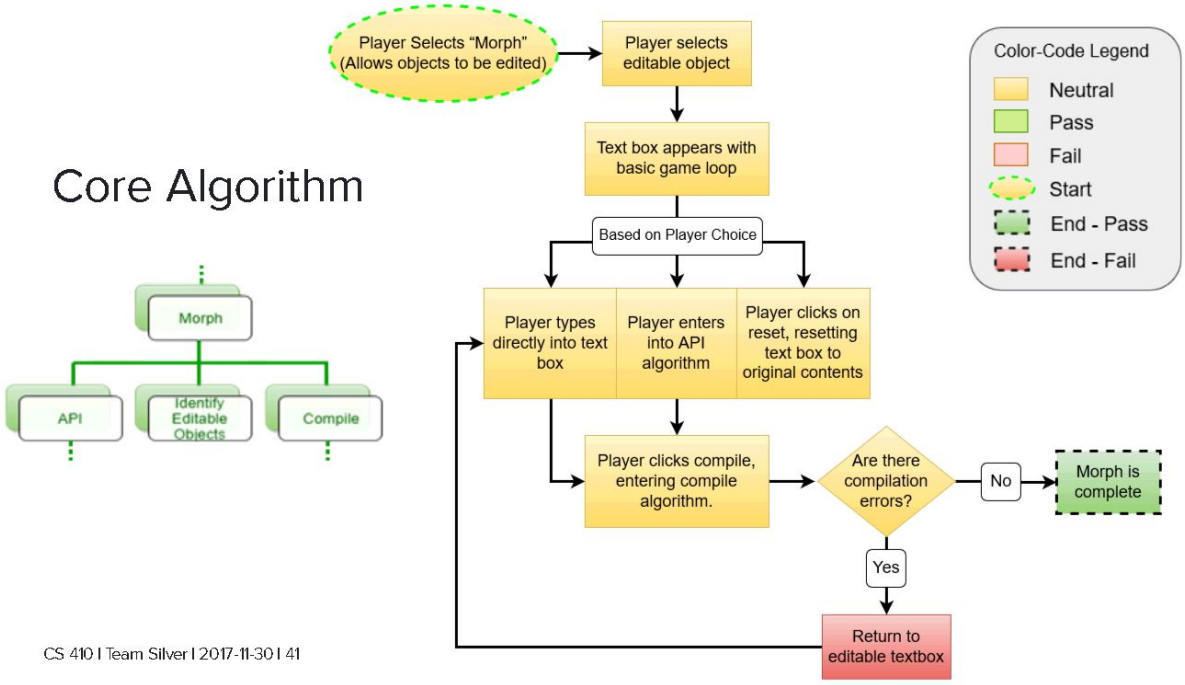

Figure 8: Dataflow Algorithm - Core Algorithm (Team Silver, 2017)

Next, the API Book Algorithm describes the functionality of PolyMorpher's "API Book" button in the game. The API Book Algorithm acts as the primary method of information distribution from game designer to player (Team Silver, 2017). It interacts directly with the Compiler Algorithm by determining the knowledge base the player has to exploit in the Compiler Algorithm. It also directly influences the outcome of the gameplay/challenges by offering the player a multitude of tools to interact with their environment (Team Silver, 2017). The dataflow algorithm for the API Book Algorithm is displayed in Figure 9.

[This space intentionally left blank]

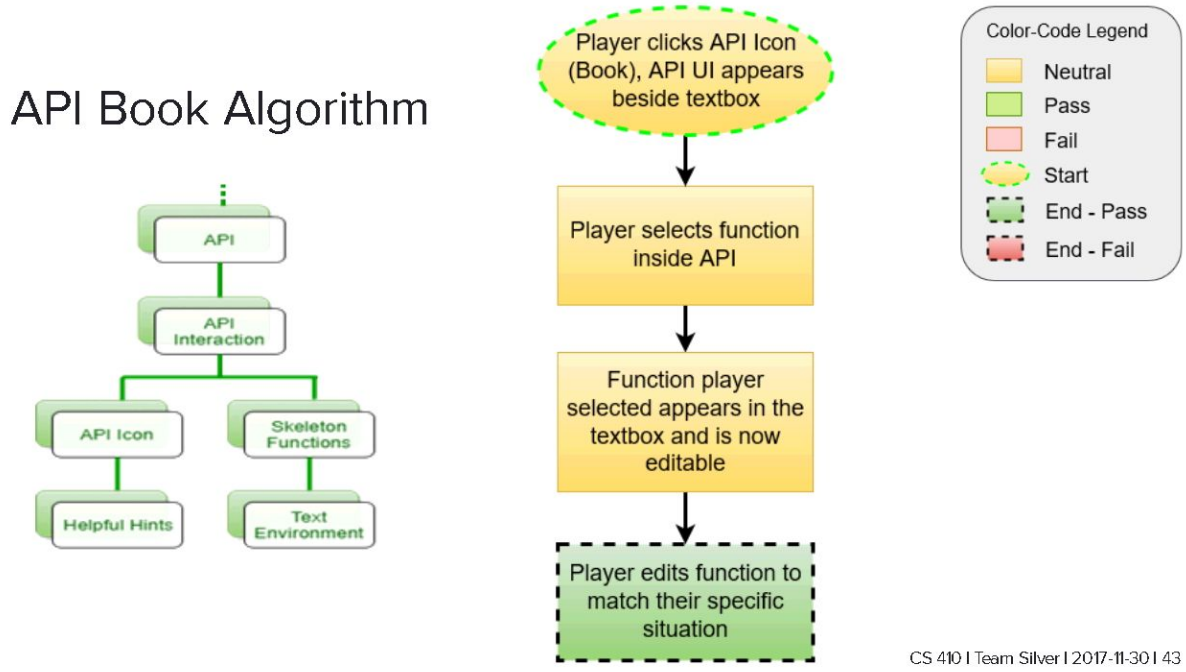

Figure 9: Dataflow Algorithm - API Book Algorithm (Team Silver, 2017)

Lastly, the Compiler Algorithm allows the player to write custom scripts in order to morph certain objects to progress in PolyMorpher. The Compiler Algorithm controls and continuously affects the back-end system of the game itself. It directly determines the behavior of objects in the game's environments at a fundamental level (Team Silver, 2017). It is also responsible for the amount of control and open design power the player is granted during gameplay. The Compiler Algorithm is, in fact, co-dependant on the API Book Algorithm. This is based on the tools the player is likely to find there to use within the Compiler Algorithm's in-game dependencies (Team Silver, 2017). The dataflow algorithm for the Compiler Algorithm is displayed in Figure 10.

[This space intentionally left blank]

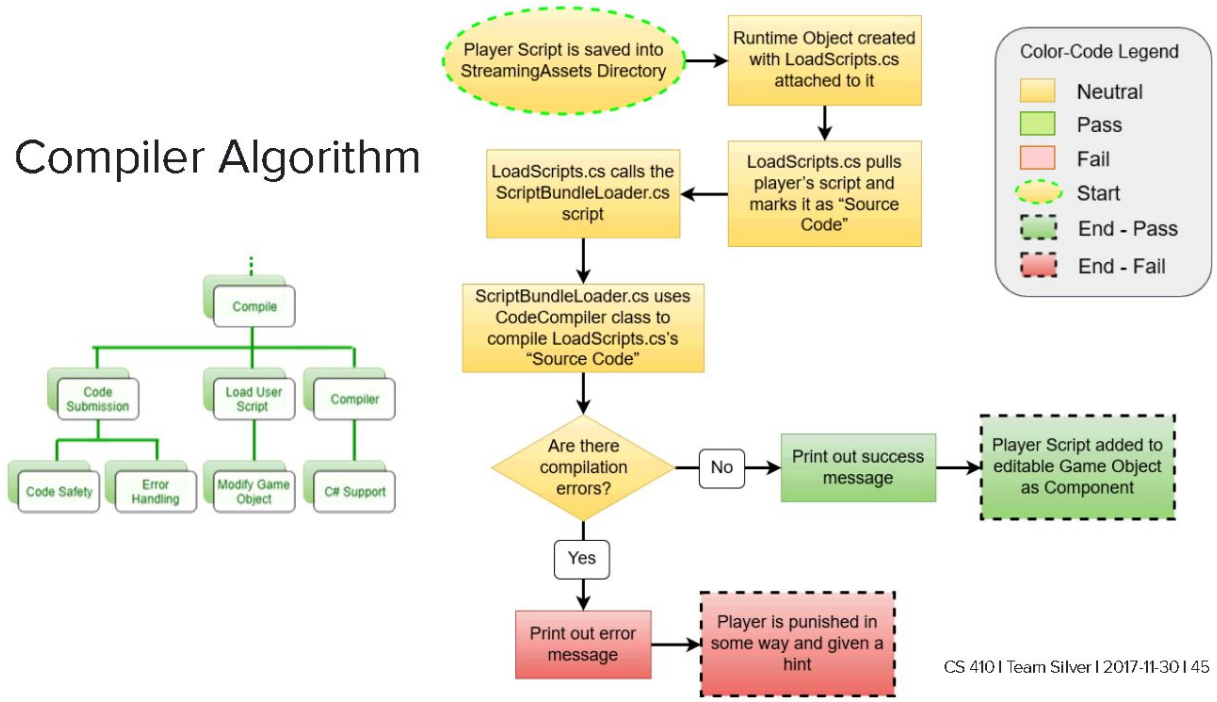

Figure 10: Dataflow Algorithm - Compiler Algorithm (Team Silver, 2017)

As shown in Figure 10, the Compiler Algorithm has some key components that need to be defined in order to be fully understood. The Streaming Assets Directory, Code Compiler Directory, LoadScripts.cs, ScriptBundleLoader.cs, and Coding Interface are all key components of the Compiler Algorithm. The Streaming Assets Directory is a 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 (Team Silver, 2017). It is unique in that it is one of the few source file directories that are accessible to the player in the Unity Engine.

Moreover, the Code Compiler Directory is a file directory holding the portable compiler and associated companion files. Both of these are used to manipulate the scripts in the Streaming Assets Directory (Team Silver, 2017). LoadScripts.cs is a file that 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 the game (Team Silver, 2017). This file is pulled from the Streaming Assets Directory and passed to the ScriptBundleLoader.cs file for compilation.

Additionally, the ScriptBundleLoader.cs file takes scripts passed off from the LoadScripts.cs file and marks them for compilation, setting up the Assembler and Compiler and running the selected scripts through them. Any compilation errors will cause an error report to be sent out through the Unity Error and Log files; otherwise, the script will be attached to whichever game object was selected (Team Silver, 2017). The Coding Interface is 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.

4.3 Prototype Development Challenges

PolyMorpher's prototype development challenges consist of design continuity, difficulty debugging, playtesting, maintaining player engagement, teaching enough, and game testing. Game testing is the main development challenge that is focused on. The particular aspects to the game testing are to test the: API Book in-game application, compiling of user-input source code, and effectiveness of the TUI and gameplay (Team Silver, 2017). An Alpha Test will include closed testing conducted by Team Silver such that the API Book is the focus. A Beta Test will include open testing conducted by ODU faculty such that the compiling, TUI, and gameplay are the focus.

4.4 Risk Matrix/Risk Mitigation

The Risk Matrix and Risk Description for PolyMorpher is displayed in Table 6. The red color represents a high and very high impact and probability of the risk. The yellow color

represents a medium impact and probability of the risk. The green color represents a low and very low impact and probability of the risk.

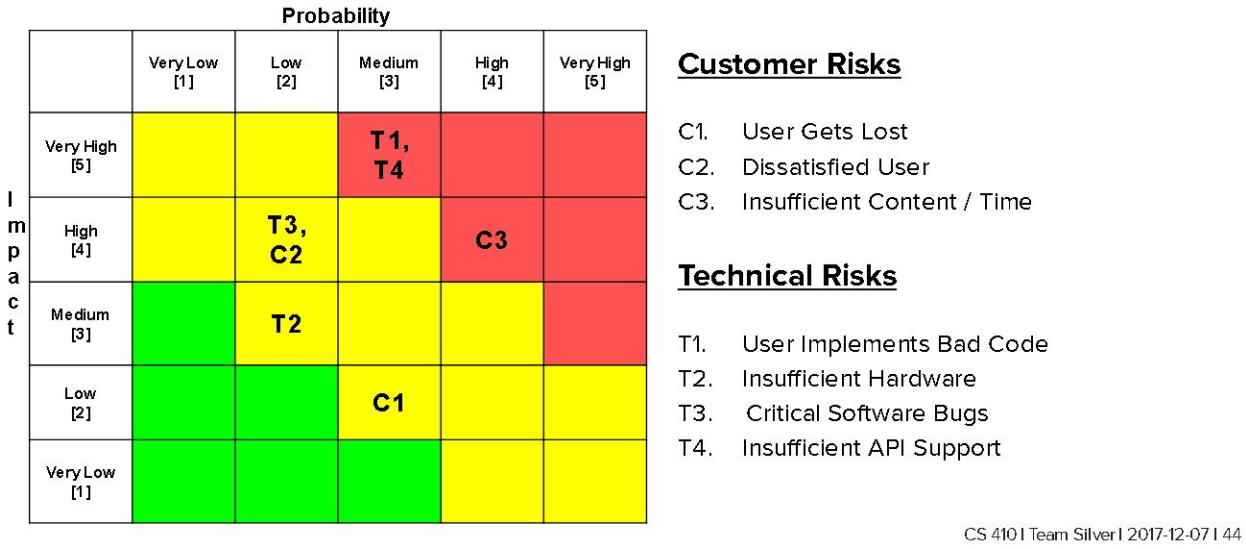

Table 6: Risk Matrix & Risk Description - Customer & Technical Risks (Team Silver, 2017)

The customer risks include: user gets lost, dissatisfied user, and insufficient content/time. The user gets lost risk deals with a user that gets stuck and does not know what to do in the game. Usually, this type of user has a low level of technical experience in programming. This risk has a medium probability and a low impact. The mitigation for this risk is to include sufficient resources and hints in the gameplay so as to enable the user to effectively learn the material (Team Silver, 2017). The dissatisfied user risk deals with a user that simply dislikes the user interface (UI/UX). This risk has a low probability and a high impact. The mitigation for this risk is to make the UI/UX design enhance an approachable interface, have various menu options, and include an interface that will incorporate clear objects throughout each level (Team Silver, 2017). The insufficient content/time risk deals with not having enough content/time in order to successfully pass introductory CS classes. This risk has a high

probability and a high impact as well. The mitigation for this risk is to use play testing to optimize the pacing and content of the game (Team Silver, 2017).

The technical risks include: user implements bad code, insufficient hardware, critical software bugs, and insufficient API support. The user implements bad code risk deals with the user implementing bad code that crashes the game, either because of their own error or because of a malicious post on a forum. This risk has a medium probability and a very high impact. The mitigation for this risk is currently under evaluation by Team Silver; Team Silver has identified a few potential solutions, but is still actively looking for one central solution. The most promising mitigations include: sandboxing the game, making the game a downloadable executable file, making the game single player, giving players a way to "reset" the level, scanning players' code for known malicious content/code, and providing an official online guide on PolyMorpher's website with code solutions that are "safe" (Team Silver, 2017).

The insufficient hardware risk deals with the user not having the sufficient hardware to run the game. This risk has a low probability and a medium impact. The mitigation for this risk is to implement a two-dimensional gameplay style that will have a minimal draw on the computer's resources (Team Silver, 2017). The critical software bugs risk deals with the occurrence of possible critical software bugs in the game. This risk has a low probability and a high impact. The mitigation for this risk is to continuously test the software through gameplay and ensure that all possible interactions work appropriately and optimally (Team Silver, 2017). The insufficient API support risk deals with potential insufficient API support for the game through a help or aid functionality. This risk has a medium probability and a very high impact. The mitigation for this risk is for the built-in API to have all the necessary tools to help the user refer to the specific

code syntax and OOP concepts that they need to learn to get through the levels of the game successfully (Team Silver, 2017).

5 PolyMorpher Development Pipeline

The software requirements for the development of PolyMorpher include the C# programming language, Unity SDK, third-party libraries and APIs like Mono and Microsoft C# Code Compiler, Gitlab, and Sourcetree acting as the development source control. The software requirements for production of PolyMorpher include the Unity SDK, Windows 7, 8, 8.1, 10, any Linux, and any MacOS operating systems.

5.1 Unity Software

Unity is the game engine that is used to create PolyMorpher. Unity is a flexible UI and developer workflow system. It allows users to develop a product efficiently, without reinventing common game development processes (Team Silver, 2017). Unity has essential tools for software game development already built in. For instance, MonoDevelop IDE and support for multiple platforms and build environments is built into Unity. Unity uses the C# programming language; C# is also the most flexible and powerful out-of-the-box programming language that Unity supports. Developer support is available through Unity's Scripting API website (Team Silver, 2017). This includes code examples and complex breakdowns of Unity specific functionality within C#.

5.2 Sourcetree Software

Sourcetree is a free git client for Windows and MacOS operating systems. It uses PuTTY to interface with Gitlab in order to manage the version control for PolyMorpher. PuTTY is a

Pageant client that communicates with Gitlab, which is an online git repository manager, that continuously runs in the background.

5.3 Version Control

The version control components for PolyMorpher consist of Unity, Sourcetree, PuTTY, and Gitlab all working together. Figure 11 is the version control flow diagram for PolyMorpher that shows how all these constituents work together.

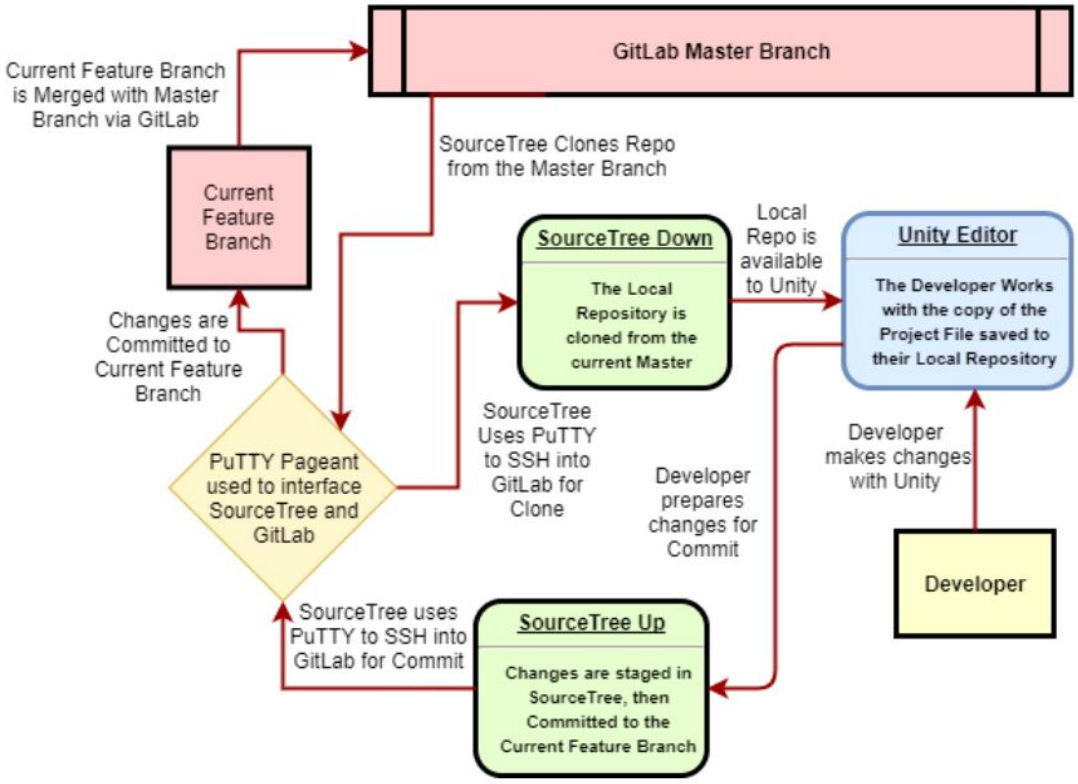

Figure 11: Version Control Flow Diagram (Team Silver, 2017)

The Gitlab master branch is where the most up-to-date version of PolyMorpher is located. Only code that is extensively tested and validated by the entire team is allowed to be committed and pushed to this branch. This branch is also where the final version of the

executable download for the game will be located and put on the website to be downloaded by players.

5.4 Development Model

The development model used to create PolyMorpher is the Agile Development model. Figure 12 displays the life cycle of the Agile Development model. For example, the development of PolyMorpher was "kicked off" with one person of the team developing, testing, and demonstrating a prototype of the game to the team. If the team likes the prototype enough, then the team can vote to use it as the version to be deployed as the final product. Otherwise, another member of the team would have to begin developing another prototype, and the cycle continues.

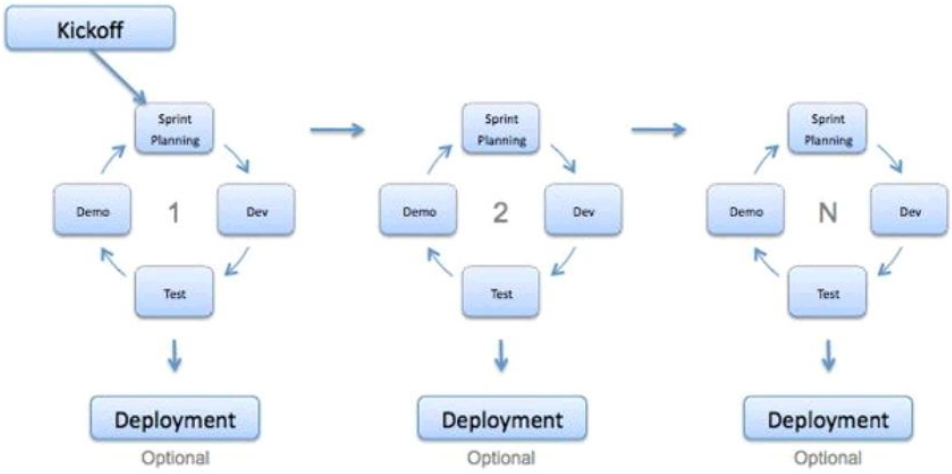

Figure 12: Agile Development Model (Team Silver, 2017)

5.5 Work Management

The development of PolyMorpher is broken down into three separate groups of three team members each from within Team Silver. Each group will work on two topics in order to create the complete game of PolyMorpher. The topics of Abstraction and Tutorial Mechanics,

Encapsulation and Polymorphism, Tutorial Code and Inheritance will be completed before deployment of PolyMorpher.

[This space intentionally left blank]

Glossary

API: Application Program Interface

Computer: a programmable electronic device designed to accept data, perform prescribed mathematical and logical operations at high speed, and display the results of these operations

Computer Programming: a process that leads from an original formulation of a computing problem to executable computer programs

Computer Science (CS): the science that deals with the theory and methods of processing information in digital computers, the design of computer hardware and software, and the applications of computers

Design: an outline, sketch, or plan, as of the form and structure of a work of art, an edifice, or a machine to be executed or constructed

Git: version control system for tracking changes in computer files and coordinating work on those files among multiple people

GitLab: web-based git repository manager the includes wiki and issue tracking

Gradle: an open-source build automation system that was designed for multi-project builds

GUI: Graphical User Interface

JavaScript: a programming language commonly used in web development where the the code is processed by the client's browser

Management Simulator: a way to simulate the management of a game in an organized fashion

MySQL: an open source multi-user database management system

Non-Technical Game: user-friendly gameplay able to be utilized by non-technical users

Non-Technical User: user who lacks formal education or knowledge in computer science, computer programming, object-oriented programming, or problem solving skills

Object-Oriented Programming (OOP): A schematic paradigm for computer programming in which the linear concepts of procedures and tasks are replaced by the concepts of objects and messages

ODU: Abbreviation for Old Dominion University

Platform: an integrated set of packaged and custom applications tied together with middleware

PolyMorpher: a programming game that focuses strictly on teaching OOP and problem solving skills

Problem Solving: the process of finding solutions to difficult or complex issues

Programming Game: a video game which incorporates elements of computer programming into the game, which enables the player to direct otherwise autonomous units within the game to follow commands in a domain-specific programming language

Regression Testing: a type of application testing that determines if modifications to the application have altered the application negatively

Software Development Kit (SDK): a set of software development tools that allows the creation of applications for a certain software package

Student Involvement: the amount of physical energy students exert and the amount of psychological energy they put into their college experience

Student Progression Dilemma: the problem of CS majors at ODU not advancing through the CS course schedule in order to graduate with a CS degree

TUI: Tangible User Interface

Ubuntu: open-source Linux operating system

Unity: a popular game development platform

User-Friendly: easy to comprehend by non-technical users

Virtual Machines: emulations of computer systems that provide functionalities of physical computers

Web Application: a client-server computer program in which the client (including the user interface and client-side logic) runs in a web browser

Wiki: a website on which users collaboratively modify content and structure directly from the web browser

[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

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-5KdQEdqLUPdnBFUnp2V05uMEE%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

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

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

Team Silver. (2017, December 13). Prototype PowerPoint Presentation. In *PolyMorpher*. Retrieved from https://docs.google.com/presentation/d/e/2PACX-1vSidnjCKAuVEtKshHkyO7A-OfW3qWIKRkxcp0em412WwL1ig6SFmnqrMUyHr8-FMvzvaRjmcKYiCytq/pub?start=false&loop=false&delayms=3000&slide=id.g25ab9a9d23_0_1542

Team Silver. (2017, November 21). Design PowerPoint Presentation. In *PolyMorpher*. Retrieved from https://docs.google.com/presentation/d/e/2PACX-1vSllslBDmSvRfMI9nbrp0RmRaPRsHNz7YWDfKNiF5sg15cp7ycQ774MuMgm4G4qhR6hohTiUQrrjRdo/pub?start=false&loop=false&delayms=3000&slide=id.g25ab9a9d23_0_1542

Team Silver. (2017, October 25). Feasibility PowerPoint Presentation. In *PolyMorpher*. Retrieved from https://docs.google.com/presentation/d/e/2PACX-1vReG6Sodx-gVFro1ByYMOYHSyiSRiU5HW-Su-PyMVGO8F4CQ7pY49tB_pJecVApruksoGaP_00ozhmR/pub?start=false&loop=false&delayms=3000&slide=id.g25ab9a9d23_0_1542

"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

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