Lab 1 – PolyMorpher Product Description

Nathaniel DeArce

Old Dominion University

CS411W

Professor Thomas Kennedy

14 February 2018

Version 2

Table of Contents

List of Figures

List of Tables

Lab 1 – PolyMorpher Software Description

1 Introduction

Programming is intimidating for the uninitiated. As a result, first time Old Dominion University (ODU) programming students drop out or switch majors. Existing tools fail to teach Object-Oriented Programming (OOP) concepts and problem-solving skills.

1.1 Problem Characteristics

There exists a student progression dilemma for Computer Science (CS) students at ODU. Numbers obtained from the Student Factbook indicate significant drops in student enrollments during natural course progression. Introductory CS courses can require students to think differently than what they are used to which can make it difficult for newer students to pass introductory courses. Figure 1 shows the current process flow.

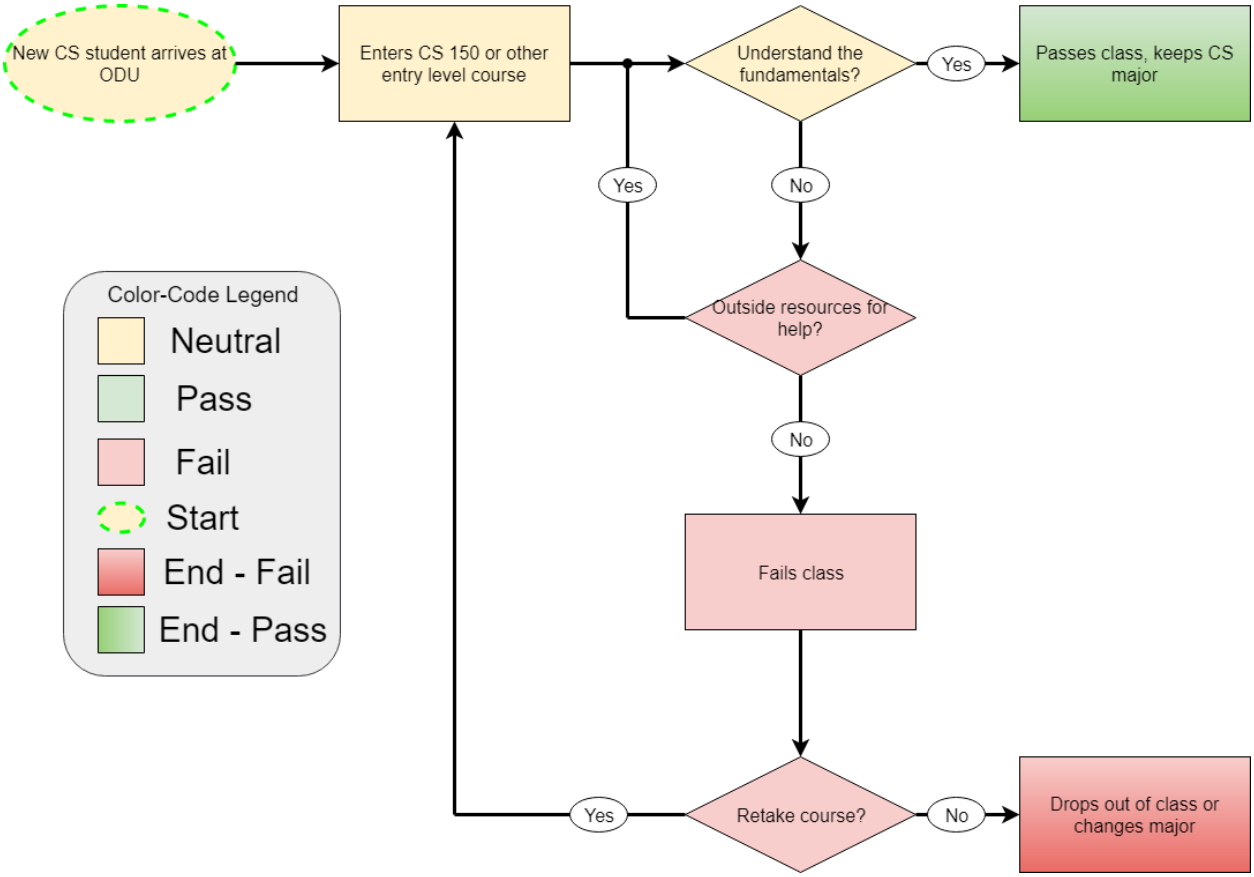

Figure 1 Current Process Flow

1.2 Solution Characteristics

Multiple competing educational games have been created to teach programming in general, but few of those games attempt to teach OOP. The existing solutions are discussed in Section 1.3. A solution that minimizes the required experience while also teaching a high level of OOP is needed. PolyMorpher is an educational video game designed to teach OOP concepts to novice programmers. PolyMorpher does this by allowing players to type and compile relevant code while playing the game. This game utilizes puzzles of increasing difficulty to develop the player problem-solving skills.

1.3 Current Solutions

Table 1 shows the competition matrix for PolyMorpher, which is highlighted in green. There is only one existing solution that teaches OOP during gameplay, but it requires the player to have mid-to-high prior programming experience. There are many games that use OOP concepts in gameplay, but they do not teach them.

Table 1 Competition Matrix

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
Ruby Warrior	Low	No	No	1	No
Empire of Code	Low-Mid	Yes	No	2	Yes
Code Combat	Low-Mid	Yes	No	5	No
Codewars	Mid-High	Yes	Yes	6	Yes
Codingame	Low-High	Yes	No	25+	Yes
Elevator Saga	Mid-High	Yes	No	1	No
Screeps	Mid-High	Yes	No	1	Yes

1.3 Team Members

Team Silver is undertaking the development of PolyMorpher in the Professional Workforce Development course at ODU taught by Professor Thomas Kennedy. Team Silver consists of Colten Everitt, Casey Batten, Peter Riley, Kevin Santos, Joel Stokes, Matthew Tuckson, Nathaniel DeArce, Daniel Dang, and Tyler Johnson.

2 PolyMorpher Product Description

PolyMorpher is an educational game that will provide players the opportunity to learn OOP concepts with minimal prior experience in programming. OOP will be used to help players understand how to connect their code to the real world.

2.1 Goals and Objectives

In order to make PolyMorpher as accessible as possible, PolyMorpher will assume minimal prior programming experience. Previous programming experience will not be assumed. Introduction of OOP concepts will be done in a way that does not overwhelm the player. Figure 2 shows the proposed solution process flow.

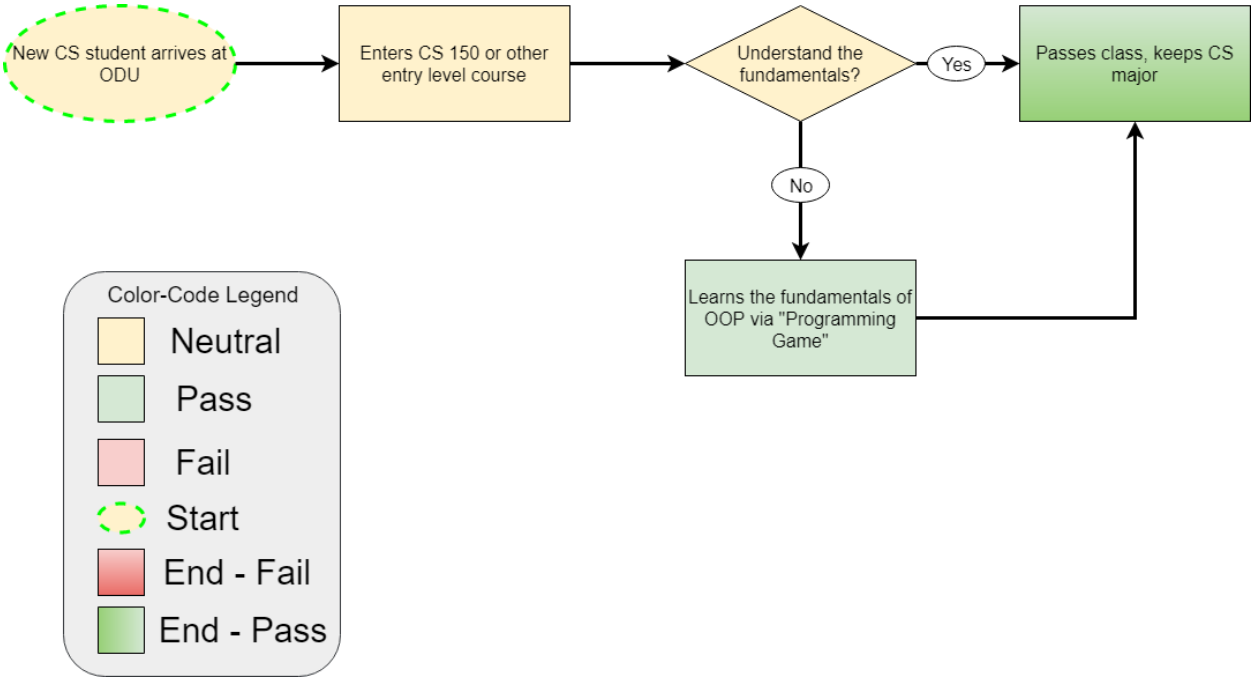

Figure 2 Solution Process Flow

2.2 Key Product Features and Capabilities

In PolyMorpher, the player will be able to interact and modify the code of objects. For example, if the player needs to move a box that will not budge, the player will be able to modify its code to make it respond to the player's actions. The player will need to use this mechanic to

solve puzzles throughout the game. PolyMorpher will support multiple object-oriented languages such as C++, C#, and Java. Code written by the player will be compiled in a sandboxed environment and the game will update according to the changes.

2.3 Major Components

PolyMorpher will be developed for PC, Mac, and Linux. To minimize system requirements and maximize accessibility, PolyMorpher will be implemented as a 2D game. The game will be available through the CS 411 Team Silver website as a downloadable executable and through other game distribution services such as Steam.

3 Identification of Case Study

PolyMorpher was born to remedy the student progression dilemma for ODU Computer Science students.

3.1 End User

The end users of PolyMorpher will be students, teachers, and anyone that wants to have a resource with which to learn object-oriented programming.

PolyMorpher is being developed with the goal of providing students an extra resource through CS 150 at ODU. This solution is not inherently exclusive though. PolyMorpher could just as easily benefit programming students from other universities.

PolyMorpher can be used by instructors as a resource for their students. PolyMorpher can be provided by instructors who wish to provide their students with another way to learn concepts they are teaching.

Although PolyMorpher is being developed as a game targeted toward traditional learners (e.g. students), non-traditional learners can benefit just as much from playing PolyMorpher. The

nature of games as educational tools makes PolyMorpher an effective tool for non-traditional learners.

3.2 Customers

The intended customers of PolyMorpher are educational institutions. ODU has been targeted as the primary customer Team Silver has chosen to target ODU in particular as that is where its members have identified the problem PolyMorpher aims to solve.

3.3 Influences on Product Development

Team Silver has identified a student progression dilemma for CS students at ODU. New CS students tend to drop out or change majors while in introductory courses of the CS program. Team Silver believes that this is because of the way programming is taught at the introductory levels. Object-Oriented Programming concepts are not introduced in the ODU CS curriculum until 200-level courses, and OOP concepts can help newer programmers understand programming better.

Figure 3 shows enrollment statistics for ODU CS courses. One will notice the drastic drop in student enrollment between CS 150 and CS 250. A majority of that difference is due to the fact that most Engineering, Math, and Physics students are required to take CS 150. There is some of this white noise in the headcounts for other CS courses, but they are smaller than the white noise in CS 150. Accounting for this, one will notice that there is still a drop-off worthy of attention. One may also notice a significant drop between CS 250 and CS 361 and CS 330. This drop is because the ODU CS curriculum branches off at this point. Students have the option of either CS 361 or CS 330 after CS 250.

	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

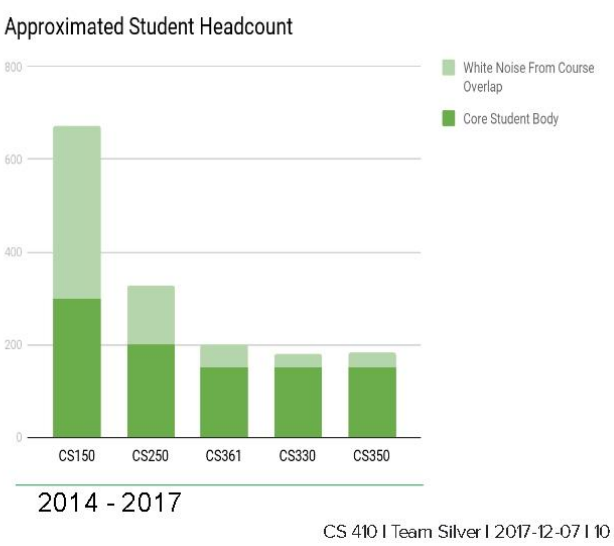

Figure 3 Statistics from the ODU Student Factbook

4 Product Prototype Description

To demonstrate the functionality and effectiveness of PolyMorpher as a teaching tool a prototype will be developed to showcase primary functions.

4.1 Prototype Architecture

This prototype will be offered as an executable file on PC, Mac, and Linux machines. The prototype will require that the player has a minimum 4th Gen Intel i3 Processor.

4.2 Prototype Features and Capabilities

The PolyMorpher prototype will not have the full capabilities of the real-world product. Table 2 highlights the differences between the prototype and real-world product. Table 3 is the associated key for Table 1.

Table 2 Real-World Product vs Prototype Features

Element	Description	Real-World Product	Prototype
Abstraction	The game will provide a section that covers Abstraction		
Developed Story	The game will feature a narrative to drive progression		
Encapsulation	The game will provide a section that covers Encapsulation		
Inheritance	The game will provide a section that covers Inheritance		
Polymorphism	The game will provide a section that covers Polymorphism		
Portable Compiler	A compiler will be provided that can compile player-edited code		
Single Player	The game will feature gameplay for only a single player		
Tutorial Level	Provides players a way to familiarize themselves with game controls and mechanics		
Multiple Languages	The game will support multiple object-oriented programming languages		
Player-Made Content	Players will be able to create their own puzzles or content		
Sandbox Level	Players will have access to all tools they would receive through normal progression		
Multiplayer	Gameplay features aspects that require the cooperation of multiple players		
Web Application	Web-based version of the game running in browser		

Table 3 Key to Table 1

Fully Functional
Partially Functional
Eliminated

4.2.1 Fully Functional Components

The goal of PolyMorpher is to teach OOP concepts. The PolyMorpher prototype will showcase how the real-world product will teach the OOP concepts to its players. To do this, the prototype will have a fully functional C# portable compiler that can compile code written by the player during gameplay. The language of focus for this prototype will be C# because Unity scripts are written in C#, so this simplifies development of the prototype to focus on the teaching aspects of PolyMorpher. This prototype will only be provided with a single-player component. The reasons for this will be discussed in section 4.2.3. A tutorial section will be provided in the prototype that will teach game mechanics, like controls and how to modify game objects, and teach players preliminary knowledge of C#, like control structures and data types. This prototype will also feature a story to drive progression through the game.

4.2.2 Partially Functional Components

These components are not necessary to be implemented within the prototype to demonstrate its effectiveness as an OOP teaching tool. The inclusion of these components will be dependent on time. The final real-world product of PolyMorpher will provide the player with a choice of object-oriented programming languages to play the game with. The prototype may not have all possible languages implemented and development will be focused on the use of C# as the primary language. PolyMorpher will include extra modes that give the player the ability to create their own puzzles and share them online with others. The ability for players to create their own content may not be developed in the prototype. PolyMorpher will also include a sandbox mode which allows the player to use any tools they receive through normal game progression in an endless level. The previous two components are not necessary to teach OOP concepts, so their development is subject to time.

4.2.3 Eliminated Components

The PolyMorpher prototype will not be capable of multiplayer. The ability of the portable compiler to compile any player written code in real-time can be very dangerous. Malicious code could damage a player's installation of the game, or their system itself. Steps that can be taken to mitigate this will be discussed later, but one mitigation is to not include multiplayer until a suitable solution can be developed. This prototype will also not be available as a web application. This is due to the nature of the portable compiler. The current implementation does not allow for it to be included in a web application.

4.2.4 Algorithms

This section will discuss the three major algorithms that make up gameplay in PolyMorpher. The three algorithms are the Core algorithm, API algorithm, and Compiler algorithm.

Figure 4 shows the flowchart of the Compiler Algorithm. Whenever the player requests for their code to be compiled the script they modified is first saved to the StreamingAssets directory. This directory holds files that are changed while the game is running. A runtime object is created with LoadScripts.cs which pulls the player-edited script and marks it as "source code." ScriptBundLoader.cs then compiles the code LoadScripts had marked as its "source code." The compiler checks for any errors and takes the appropriate actions.

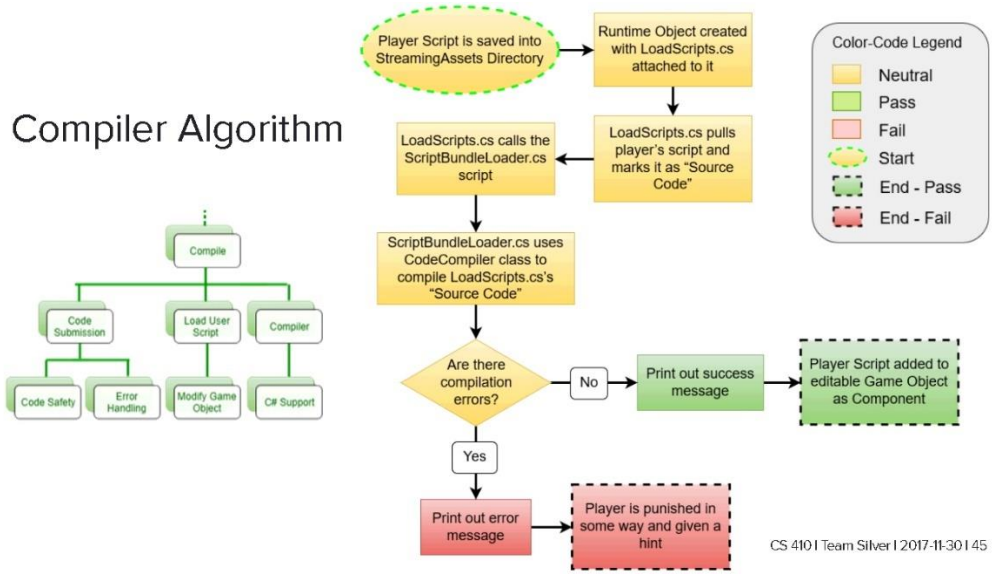

Figure 4 Compiler Algorithm

The API Book algorithm refers to the ability for the player to select function templates to edit from a list. Figure 5 shows the flowchart for the API Book algorithm. When the player clicks on the API icon (in PolyMorpher it will take the form of a book) the API UI will appear. The player will select a function inside the API and it will appear inside the editable text window. The player may then edit it to their needs.

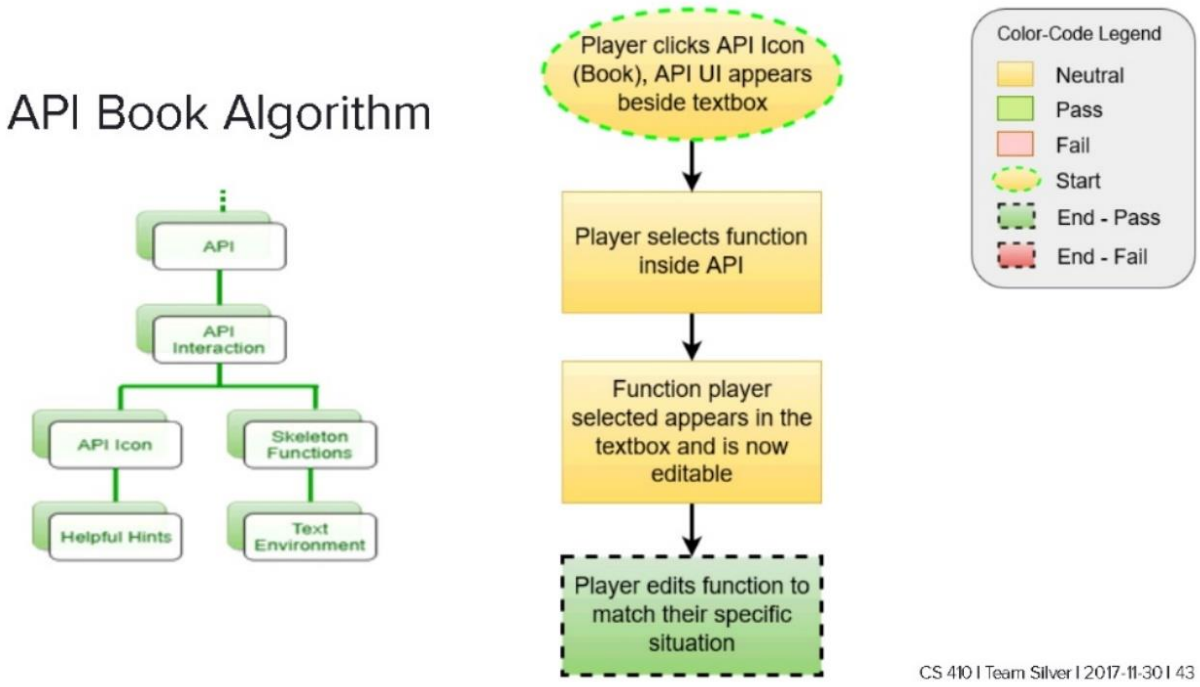

Figure 5 API Book Algorithm

This Core algorithm dictates most of the gameplay involving the modification of code in-game. Figure 6 shows the flowchart for the Core algorithm. The player will select the “Morph” button on the UI. This will highlight any object that can be edited by the player. A text box will appear with options to modify the object. The player may type their own code directly into the text box to modify the script, choose a function from the API Book, or reset the script to its original, stable contents. The player will then compile their code, and the game takes the appropriate actions according to the presence of compiler errors.

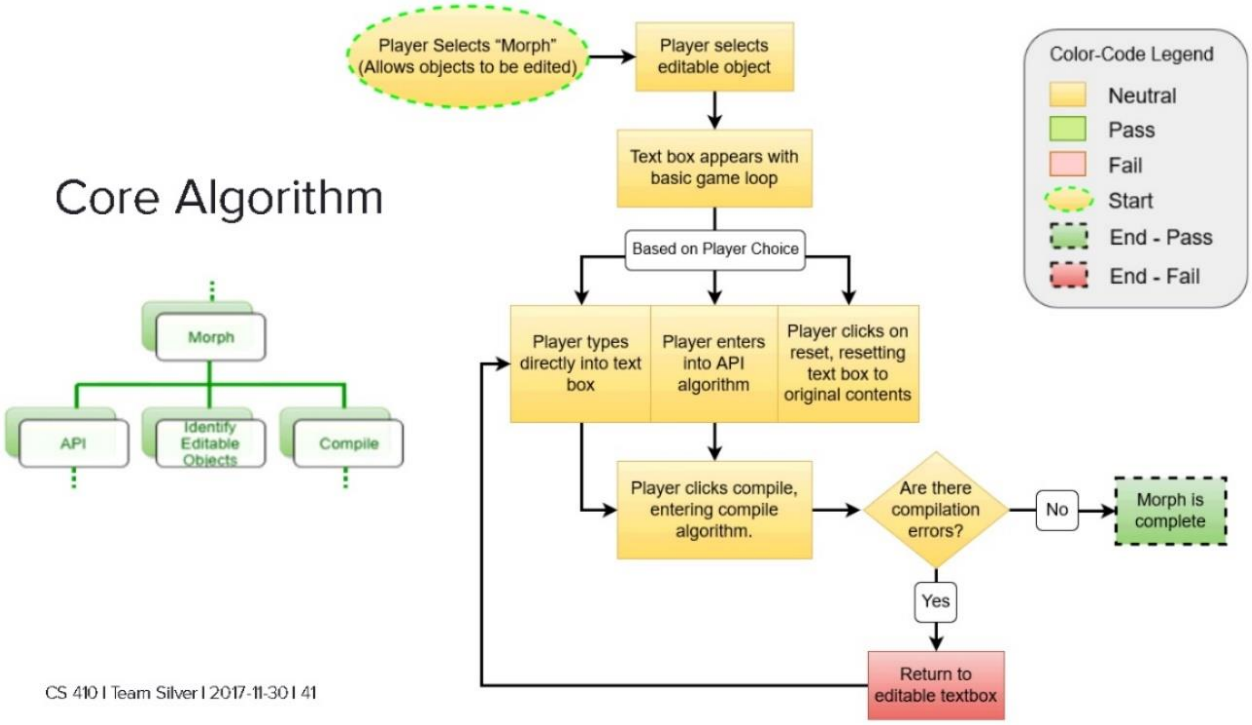

Figure 6 Core Algorithm

4.3 Prototype Development Challenges

Development of the PolyMorpher prototype will come with many challenges. It is important for the game to remain cohesive as it is being developed. It is also important for PolyMorpher to teach its lessons effectively without losing the interest of its players.

4.3.2 Play Testing for Effective Content

Games should be difficult, but not too difficult so as not to isolate an audience. Puzzles within PolyMorpher should be difficult and rewarding. If they fail to be either of those the player may not learn anything from the experience.

PolyMorpher aims to teach OOP concepts. One of the main development challenges will be to decide if PolyMorpher is teaching enough to its audience, or even if its audience is invested enough in the game to learn those concepts. Steps to mitigate this risk will be discussed in section 4.4.

4.4 **Identification of Risks**

This section describes identified risks and potential mitigations. Figure 7 shows the Risk Matrix for PolyMorpher.

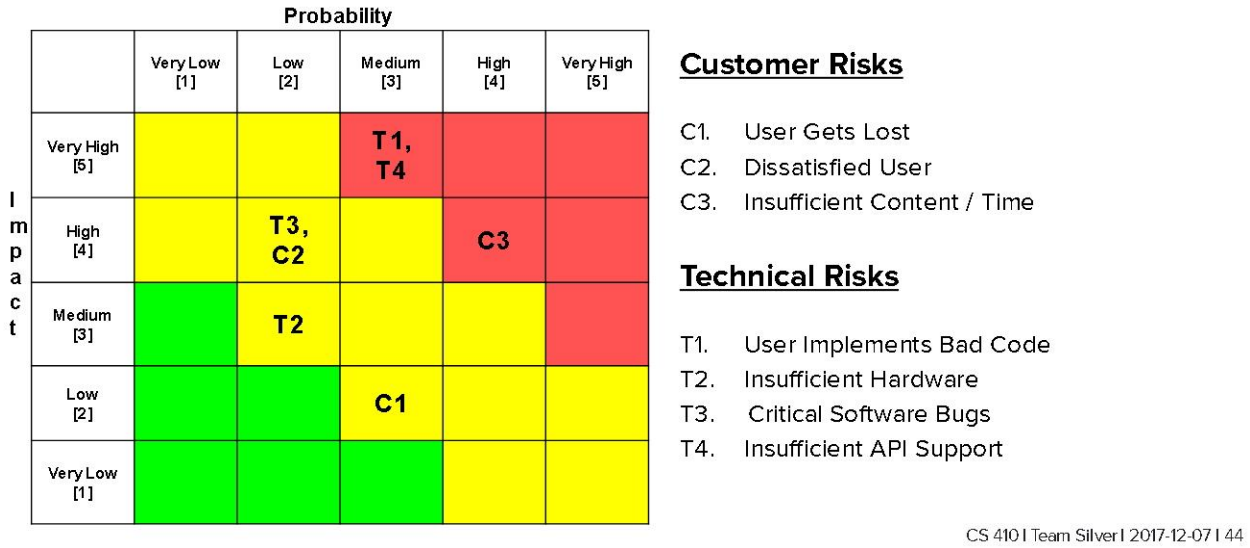

Figure 7 Risk Matrix

4.4.1 Technical Risks and Mitigations

The first technical risk is the user's ability to implement bad code. This code could be their own error or a malicious third party. Bad code has the capability of crashing a player's game or damaging their system. A mitigation has not been selected, but potential mitigations include sandboxing the game, scanning player code for known malicious code, and offering official solutions to code puzzles. Another technical risk is that players may have insufficient hardware to play the game. To mitigate this, PolyMorpher will be a 2D game to minimize the draw on computer resources, and the game will be optimized for 4th generation Intel i3 processors. As with any software, there will be bugs in the implementation of the game. This will be mitigated by continuously testing the game throughout development. If the API does not have sufficient support for the player to reference they may not be able to progress in the game or

learn any OOP concepts. The API will have all the necessary tools to help the user understand code syntax and OOP concepts to mitigate this risk.

4.4.2 Consumer Risks and Mitigations

There are three consumer risks that have been identified. The first is that the user gets stuck in gameplay and cannot progress due to the lack of helpful resources. The mitigation for this is to supply the player with enough resources to get them through the game's puzzles while also challenging them. The second consumer risk is that the user dislikes the UI. This seems small, but can be game breaking. If the UI is not helpful to the player it is only a distraction. To mitigate this the UI will serve to enhance an approachable interface. Menu options will be clear and concise. The last consumer risk is a failure of PolyMorpher. If PolyMorpher is not effective, its players will not be able to use it as a resource to pass introductory courses in CS. To mitigate this the game will be continually playtested to optimize the pace and content of the game.

5 Development Pipeline

This section highlights key aspects of the development process Team Silver will be using during the development. This will include tools used by Team Silver to aid in development, software used to develop PolyMorpher, and the software development methodology that will be used by Team Silver.

5.1 Unity

Team Silver will be using the Unity SDK for the development of PolyMorpher. Unity is a cross-platform game engine developed by Unity Technologies. Unity supports 2D and 3D game development. It is powerful enough that is has gained popularity in uses outside of game development such as simulations.

5.1.1 C#

C# is the default scripting language supported by Unity. It is an object-oriented language and is the most flexible and powerful language that Unity supports. It has a large suite of built-in functions and methods. The large community that surrounds Unity development ensures that support is always available.

5.1.2 MonoDevelop

MonoDevelop is the default IDE built into any installation of Unity. Use of this IDE is usually personal preference, however, the portable compiler that will be used by the game is built from the Mono C# compiler. Attempting to use the portable compiler with Microsoft's Visual Studio will actually prevent the game from building properly.

5.2 Sourcetree

Sourcetree is a free Git client created by Atlassian. Sourcetree will be used to interface with Git without the command line. Sourcetree supports Git LFS (Large File Storage) out-of-the-box. Git LFS will be used to manage the tracking of large files often included in a video game including sound files and image files.

5.3 Version Control

Git will be the version control system used in development. It is a familiar, simple, and powerful software to Team Silver. The ODU CS Department's Community GitLab will host Team Silver's development repositories.

5.4 Agile Development

Development will follow the Agile development methodology. Agile development can be described in Figure 8. Small teams work together to develop assigned features and will demo

their feature to the team. Feedback from the team will be given and any necessary changes will be made to the feature.

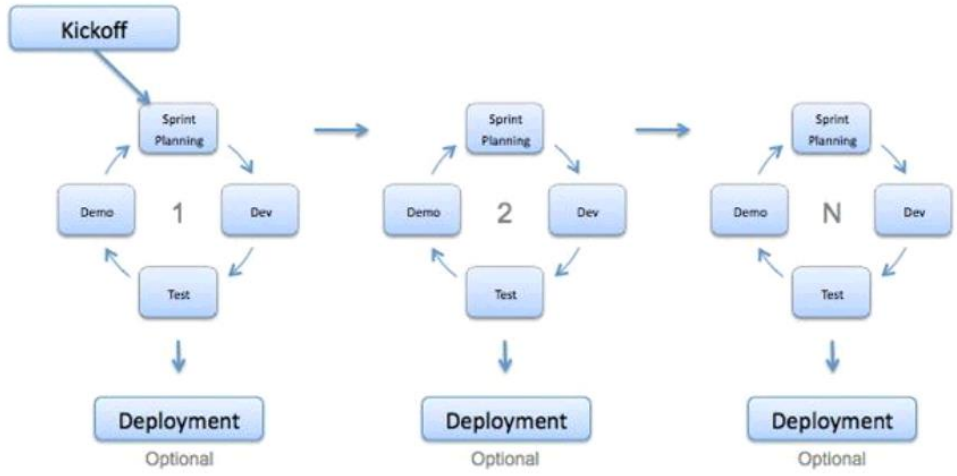

Figure 8 Agile Development Model

5.5 Work Management

Teams will be assigned to develop certain features of the prototype. A team consisting of Matthew Tuckson, Colten Everitt, and Tyler Johnson will focus on the topic Abstraction and a tutorial of game mechanics. The team of Joel Stokes, Nathaniel DeArce, and Kevin Santos will focus on the topics of Encapsulation and Polymorphism. The team of Casey Batten, Peter Riley, and Daniel Dang will focus on the topics of Inheritance and a tutorial of the C# programming language.

6 Glossary

API: Application Program Interface

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.

GUI: Graphical User Interface

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

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

ODU: Abbreviation for Old Dominion University.

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

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

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

TUI: Tangible User Interface

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

Virtual Machines: an emulation of a computer system that provides functionality of a physical computer.

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.

7 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