Lab 1 - PolyMorpher Product Description

Daniel Dang

Old Dominion University

CS411W

Professor Thomas Kennedy

1 January 2018

Version 1

Author Note

Daniel Dang, 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 Daniel Dang, Department of Computer Science, Old Dominion University, Norfolk, VA 23529.

Contact: ddang003@odu.edu

Table of Contents

List of Figures

List of Tables

Lab 1 - PolyMorpher Product Description

1 Introduction

Computer science students at Old Dominion University have a difficult time transitioning from procedural programming to Object Oriented Programming (OOP). Team Silver will introduce a solution to the declining retention rates in the computer science department at Old Dominion University by developing a gaming solution to help students learn the fundamental of computer programming and Object Oriented Programming concepts.

1.1 Team Members

Under the guidance of mentor Thomas Kennedy, Team Silver was formed to design and develop PolyMorpher. The members and roles of Team Silver are as follows:

- Matthew Tuckson (Project Manager and Team Lead)
- Casey Batten (Team Lead)
- Daniel Dang (Team Member)
- Nathaniel Dearce (Team Member)
- Colten Everitt (Web Designer and Team Member)
- Tyler Johnson (Team Member)
- Peter Riley (Team Member)
- Kevin Santos (Team Member)
- Joel Stokes (Team Lead)

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). Object oriented programming taught in lecture helps students understand at a conceptual level but can fail to show real world functionality. A game designed to visually demonstrate OOP concepts can help students better grasp OOP concepts.

1.3 Problem Characteristics

Object Oriented Programing is an organizational method for designing logical programming applications. ODU students are typically are not taught OOP concepts during entry level courses. The conceptual nature of OOP can lead students to have difficulty understanding the fundamentals, thus causing them to fall behind, drop out of classes, or even change majors.

Figure 1 displays the current process flow of an entry level computer science student at ODU. The diagram starts with new students taking CS 150, an introductory computer science course. During the introductory period, students are taught the fundamentals of programming. If the fundamentals are understood by the student, they can pass the class successfully. If the student does not understand the fundamentals, then they may seek help from outside resources, but if the resources does not help, the student will eventually fail the course. Students do have the option of retaking the course, but unless the student is able to grasp the fundamental concepts of programming they may eventually drop the class or even change majors. This is the problem PolyMorpher will address.

[This space is intentionally left blank]

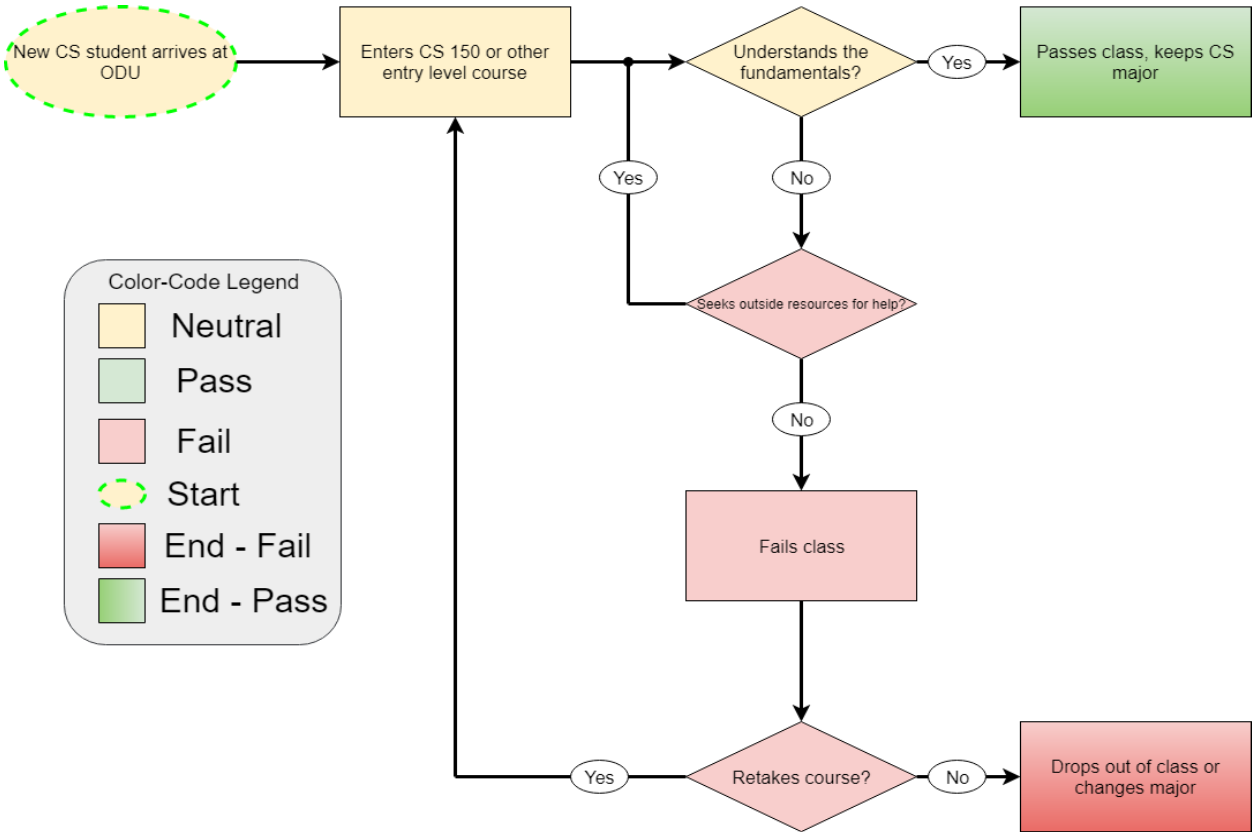

New CS student arrives at ODU
Enters CS 150 or other entry level course
Understands the fundamentals?
Yes
Passes class, keeps CS major
Yes
No
Seeks outside resources for help?
No
Fails class
Retakes course?
Yes
No
Drops out of class or changes major
Color-Code Legend
Neutral
Pass
Fail
Start
End - Fail
End - Pass

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

1.4 Solution Characteristics

PolyMorpher aims to address OOP concepts and problem solving through the use of a Management Simulator and a Tangible User Interface (TUI). A management simulator can bring experimental sandbox experiences to students who want to learn about complex processes. This type of engagement can have a larger impact on students than a traditional lecture setting. Seeing immediate tangible effects or consequences on simulated exercises can give students more confidence to experiment and learn.

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 test demonstrated a better understanding of that subject (Team Silver, 2017). Educational games can give students an incentive to think outside their normal domain of thought. Traditional methods of teaching OOP concepts are more conceptual and make it difficult for students to fully utilize. PolyMorpher will introduce interactive puzzles that will be aimed to teach the fundamentals of computer programming as a supplemental tool to strengthen the concepts taught in a traditional classroom.

PolyMorpher, as shown in Figure 2, will reduce or eliminate the possibility for students to fail and retake the course. PolyMorpher aims to be a supplemental solution for students to solidify the fundamentals needed in order to pass the class and continue with the computer science curriculum.

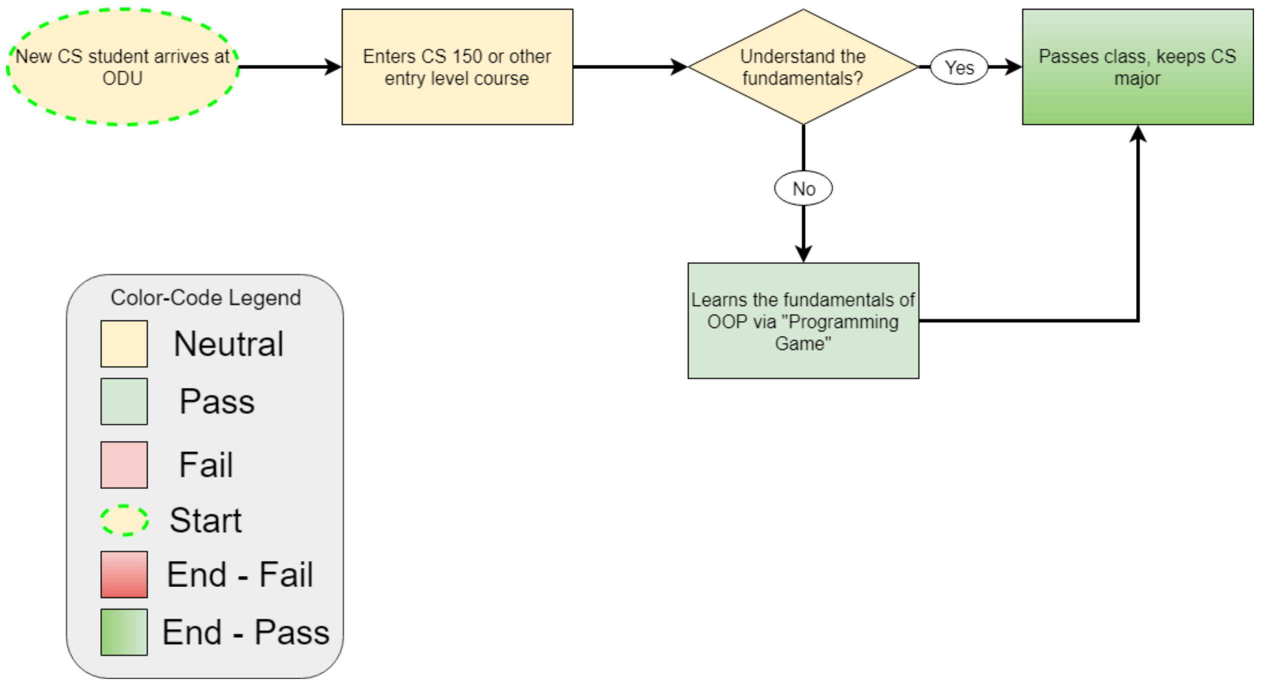

New CS student arrives at ODU
Enters CS 150 or other entry level course
Understand the fundamentals?
Yes
Passes class, keeps CS major
No
Learns the fundamentals of OOP via "Programming Game"
Color-Code Legend
Neutral
Pass
Fail
Start
End - Fail
End - Pass

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

1.5 Current Solutions (Competition)

In today's market, many programming games do not teach OOP concepts to the players. There is not a middle ground between basic games teaching syntax to fully fledged programming logic games. PolyMorpher bridges the gap between basic syntax level programming and logic intensive programming architectures. Tables 1 and 2 displays the various features provided by current programming games and PolyMorpher. PolyMorpher fills a niche in the market by targeting users with low to medium experience, teaching OOP concepts, and using OOP for core gameplay.

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

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

[This space is intentionally left blank]

Table 2: Competition Matrix - Part 2 (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

Currently there is only one language and no multiplayer. This allows PolyMorpher to cater towards a specific language and allows the user to figure out solutions on their own accord. The competitive advantage PolyMorpher has is that it allows users fully learn the fundamentals of computer science and software development. The game will teach OOP concepts, key programming and problem solving skills, and engaging single player gameplay.

2 Product Description

PolyMorpher will be a game where players must learn basic concepts and then apply them in game to win. There will be tutorials followed challenges for the players to solve. Players must code their own path the solution.

2.1 Goals and Objectives

PolyMorpher aims to provide a unique learning experience for new programmers who want to learn a skill through interactive learning puzzles. The main objective is to teach core

OOP design concepts such as: abstraction, inheritance, polymorphism, and encapsulation. The game will help users develop problem solving skills while providing a fun gaming experience.

2.2 Key Product Features and Capabilities

Important core gameplay features of PolyMorpher include using a realistic approach, supplementing concepts taught in class, and balancing fun gameplay with intensive gameplay. In order to help players better understand the fundamentals of programming, relatable examples help users solve problems on their own. More complex Object-Oriented Programming concepts can be more readily explained through visual interactive demonstrations rather than traditional audio and textual resources. By having a focus on balance of gameplay and player experience, PolyMorpher aims to achieve balance between engagement and learning.

2.2.1 Minimal Game Specifications

PolyMorpher will be optimized for machines with 4th geneartion Intel i3 Processors. The minimum operating system required to run PolyMorpher will be Windows 7. PolyMorpher will be a 2D game requiring minimal processing power.

2.2.2 Obtaining PolyMorpher

Customers can obtain PolyMorpher by going to Team Silver's website, http://www.cs.odu.edu/~411silver/, and downloading the appropriate game file under the downloads tab.

3 Identification of Case Study

PolyMorpher was introduced to solve student progression issues within the ODU computer science department. Although it is a solution designed for computer science students, PolyMorpher is accessible towards general audiences outside of the classroom as well. It was created in order to ease programming concepts to new students in an intuitive way.

3.1 End User

End users of PolyMorpher will include students, teachers, and people who want a supplemental resource for learning object oriented programming.

3.1.1 University Students

Although PolyMorpher is designed for all students, the original goal is intended to provide a supplimental resource to ODU students who are struggling with CS 150 and above. Although it was designed with these specific students in mind, PolyMorpher could fill the niche for students from other Universities as well.

3.1.2 Instructors

Individual instructors can utilize the game in classrooms to give students additional examples of certain concepts.

3.1.3 Self Learning Programming Students

Targeting self-studying programming who are not programming from traditional institutions can help give real world interactive examples to otherwise conceptual topics.

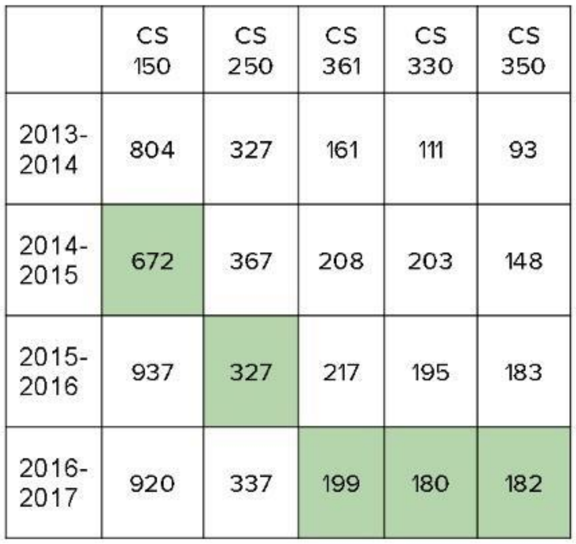
3.2 Target Customers: Old Dominion University

PolyMorpher intends to help Old Dominion University computer science students excel in the beginner to intermediate programming classes. Increasing student retention in the program will help student engagement, prestige, and revenue of the department. Ultimately PolyMorpher core consumer target includes anyone who is seeking to understand the fundamentals of computer programming, OOP concepts, and improve problem solving skills.

3.3 Why PolyMorpher?

Currently there is a student progression dilemma for computer science students at ODU. Recent statistics provided by Team Silver’s advisor, Thomas Kennedy, shows a decreasing number of students progressing to upper computer science classes. Although a portion of students from lower level programming classes are not computer science majors, there is still a disproportionate amount of students in higher level computer science classes compared to lower level classes. PolyMorpher can help increase student retention as they progress through the computer science degree program.

Figure 3 contains the number of students enrolled in the four main programming course within the last 4 years. The drop in progression from CS 150 to CS 250 can be attributed to the fact that CS 150 is a requirement for non-computer science majors. Students enrolling in CS 250 can be largely attributed to computer science, computer engineering, and modeling and simulation majors. There is still a large gap in retention with only computer science students.



	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

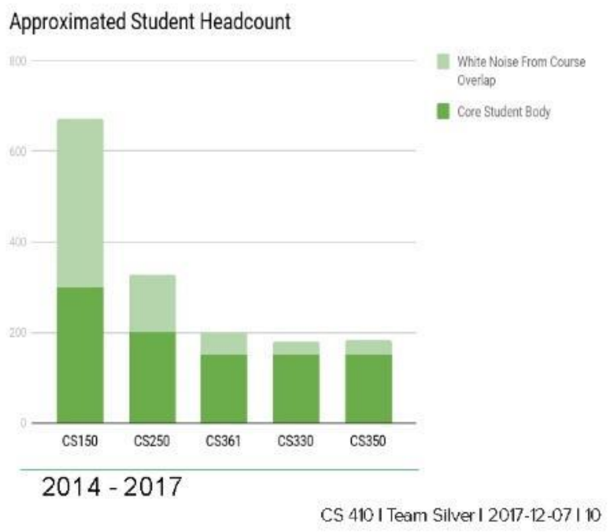


Figure 3: Student Enrollment Statistics

[This space is intentionally left blank]

4 PolyMorpher Prototype Description

A prototype of PolyMorpher will be released early in Q2 of 2018. The prototype will be in beta testing for initial user feedback. It will not contain a fully working product but will be functional enough to meet the core objectives and purpose of the game.

4.1 PolyMorpher Architecture

Figure 4 shows the core architecture implementation of the PolyMorpher. Although futures plans can be made to implement a dedicated web application, an initial prototype will not include one. A server and database will not be included in the initial prototype.

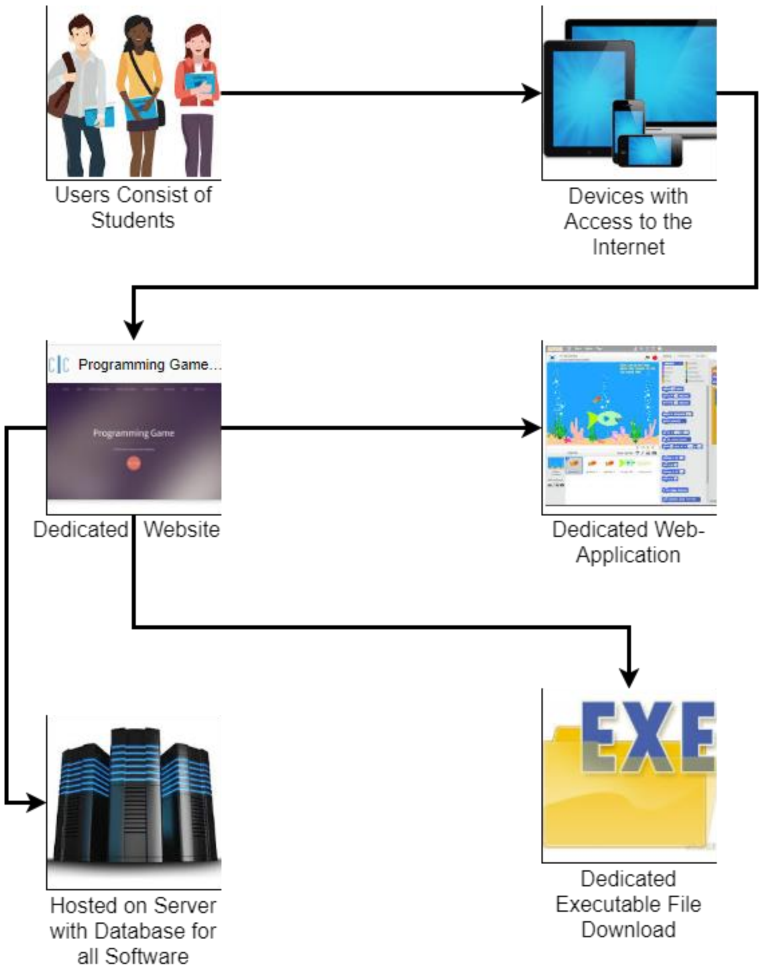

Users Consist of Students
Devices with Access to the Internet
Programming Game...
Programming Game
Dedicated Website
Dedicated Web-Application
Hosted on Server with Database for all Software
EXE
Dedicated Executable File Download

Figure 4: PolyMorpher Architecture

4.2 Functional Prototype Features and Capabilities

In order to maintain within the scope of the core goal and objective, PolyMorpher will ship with key programming concepts needed to succeed ODU's object oriented programming class (i.e., CS 330).

4.2.1 PolyMorpher Function Features

The main object of PolyMorpher is to give students a supplemental resource when learning OOP.

Lessons will be taught through interactive puzzles. Players must apply what they have learned to the challenges in order to progress through the game. Lessons on each of the key concepts above will be given and corresponding puzzles will be natively created to help players build a solid foundation while progressing through the game. As the game levels progress, game challenges will become more difficult and players must apply previously learned programming and game knowledge to progress. An initial tutorial will be included to show the basic game mechanics of the game, controls, and preliminary programming fundamentals in order to successfully progress.

Table 3 and key show the core features will and will not be implemented in PolyMorpher. The core feature of PoloyMorpher will include a fully functional C# portable compiler that can compile code written by the player within the game. The prototype will allow users to test and debug their code in order to find a working solution to solve the puzzle. This will initially be a single player experience. An interactive story will provide a seamless user experience and transition players between the various game puzzles.

Table 3: Prototype Features (Team Silver, 2017)

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	🟩	🟥
Multiple Languages Taught	Alternative programming languages for the player to use and learn in-game	🟩	🟥

Table 4: Prototype Features Keys

KEY
🟩 Fully Functional
🟨 Partially Functional
🟥 Eliminated

4.3 Algorithms

PolyMorpher needs three algorithms in order to meet the aforementioned objectives. These three algorithms allow the player to change the properties of selected objects within the game. The three algorithms include the Core Algorithm, API Book Algorithm, and Compiler Algorithm.

4.3.1 Core Algorithm

The Core Algorithm will consist of UI level interactions used to manipulate an in game object. This algorithm allows the player to choose and manipulate in game objects need to solve the puzzle. The algorithm starts when a player chooses the morph button in game. The morph feature will allow the player to choose which object they want to morph. The morphing option will range from predetermined game objects to player chosen game objects. A morphable item is an item that players can attach a script to the object. Players will need to write the script on their own to morph the object and complete the puzzle. After selecting a morphable object, the player will be taken to an in game programming environment with a text box with Start() and Update() buttons. Depending on the action the player chooses to take, different algorithms will initiate. If they player chooses to enter the API Book Algorithm shown in Figure 5, a book will appear and players can select predetermined scripts to apply to the selected object. Once a selected script is applied, the player will need to press the compile button to check for coding errors. If there are errors, then the player will be sent back to the text editor to fix the errors. If there are no errors, then a success icon will appear the object will attain the input desired.

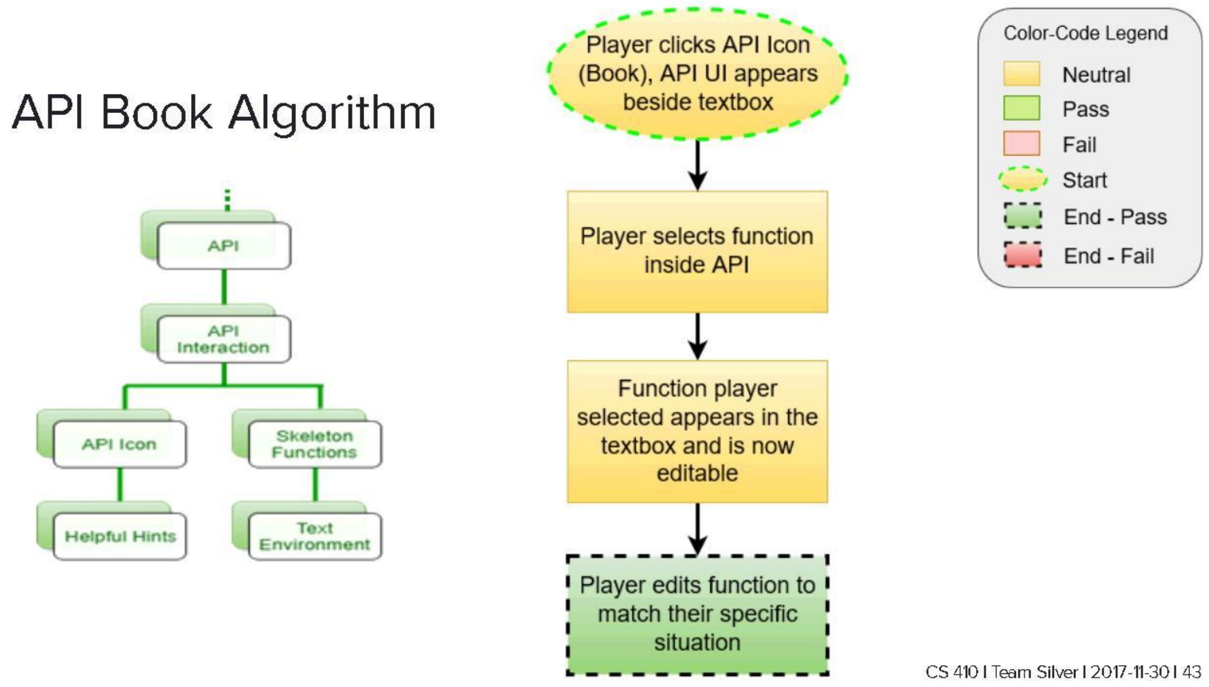

API Book Algorithm
API
API Interaction
API Icon
Skeleton Functions
Helpful Hints
Text Environment
Player clicks API Icon (Book), API UI appears beside textbox
Player selects function inside API
Function player selected appears in the textbox and is now editable
Player edits function to match their specific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

Figure 5: API Book Algorithm (Team Silver, 2017)

[This space is intentionally left blank]

4.3.2 Compiler Algorithm

The Compiler Algorithm in Figure 6 is needed for the player to run their inputted code during compilation runtime. The Compiler Algorithm will allow the player to run the code they have entered during the API book algorithm.

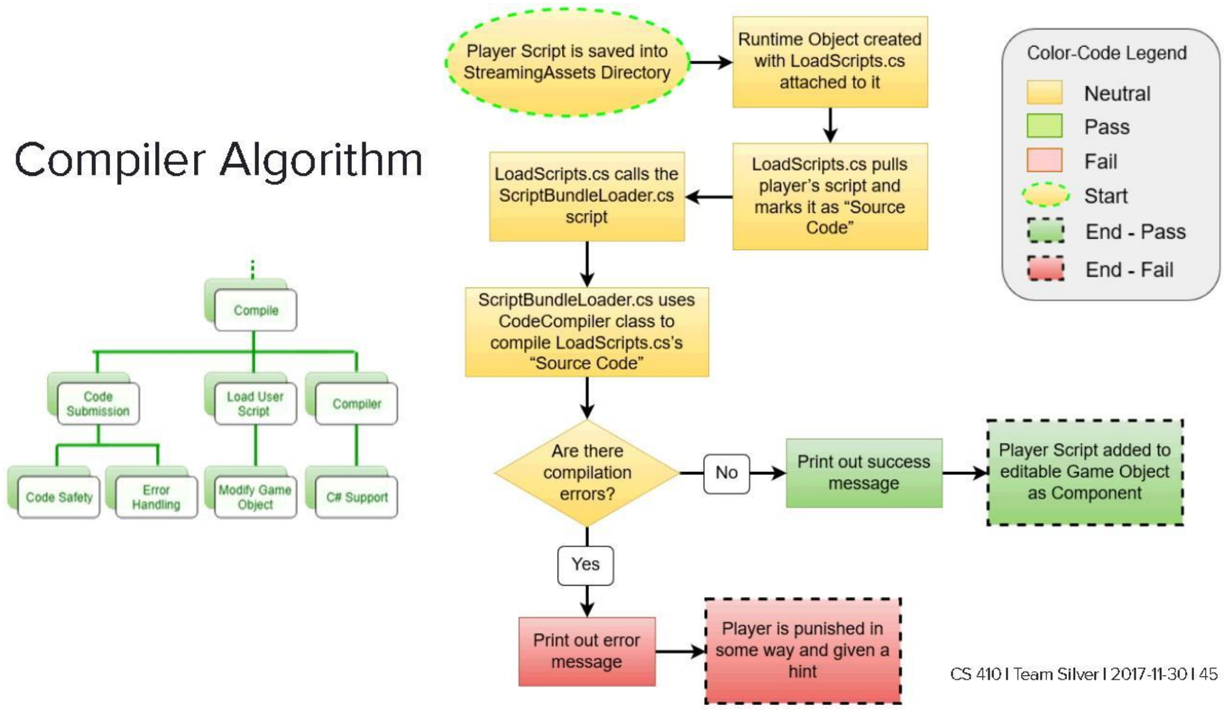
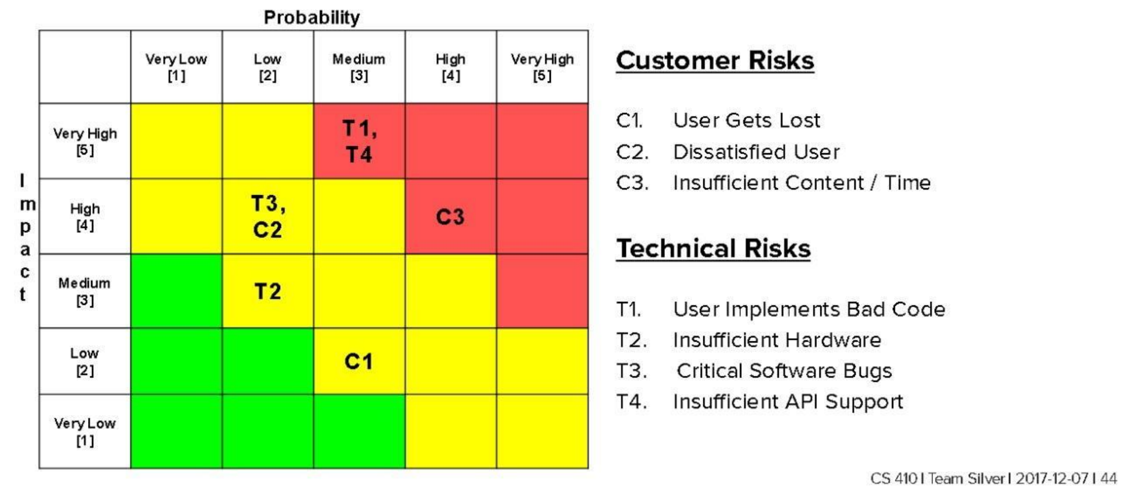

Compiler Algorithm
Compile
Code Submission
Load User Script
Compiler
Code Safety
Error Handling
Modify Game Object
C# Support
Player Script is saved into StreamingAssets Directory
Runtime Object created with LoadScripts.cs attached to it
LoadScripts.cs pulls player's script and marks it as "Source Code"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

Figure 6: Compiler Algorithm (Team Silver, 2017)

4.4 Prototype Development Risks and Mitigation

There is some inherent risk in the development of PolyMorpher. Figure 7 shows the risk matrix associated with the development of PolyMorpher. The biggest risk in implementing PolyMorpher is the user entering malicious code. It would be very difficult for the development team to test for every single situation in which malicious code can be used to break the game. Team Silver will test for the most likely scenarios and try to mitigate the possibility of code malfunction. Another high priority risk that there will not be enough material provided to support the API. The object of the game is to teach OOP concepts. If the game does not provide sufficient player support, then the player is going to be dissatisfied and quit the game. Team Silver will focus mainly on these two risks and apply the appropriate mitigation strategies.



Probability

Impact	Very Low [1]	Low [2]	Medium [3]	High [4]	Very High [5]
Very High [5]	🟨	🟨	🟥 T1, T4	🟥	🟥
High [4]	🟨	🟨 T3, C2	🟨	🟥 C3	🟥
Medium [3]	🟩	🟨 T2	🟨	🟨	🟥
Low [2]	🟩	🟩	🟨 C1	🟨	🟨
Very Low [1]	🟩	🟩	🟩	🟨	🟨

Customer Risks

C1. User Gets Lost
C2. Dissatisfied User
C3. Insufficient Content / Time

Technical Risks

T1. User Implements Bad Code
T2. Insufficient Hardware
T3. Critical Software Bugs
T4. Insufficient API Support

CS 410 I Team Silver I 2017-12-07 I 44

Figure 7: Risk Matrix (Team Silver, 2017)

5. Development Pipeline

Figure 8 shows the development pipeline for PolyMorpher. The core game development will be done on the Unity Software Develop Kit (SDK). Unity's SDK has a built in IDE called MonoDevelop which will be used by Team Silver to develop the overall game. Code will be stored on GitLab as the main version control mechanism. To interact with the Unity SDK and GitLab, SourceTree will act as a local intermediary between the GitLap and Unity. SourceTree allows individual developers to push to the development
repositories on GitLab.

[This space is intentionally left blank]

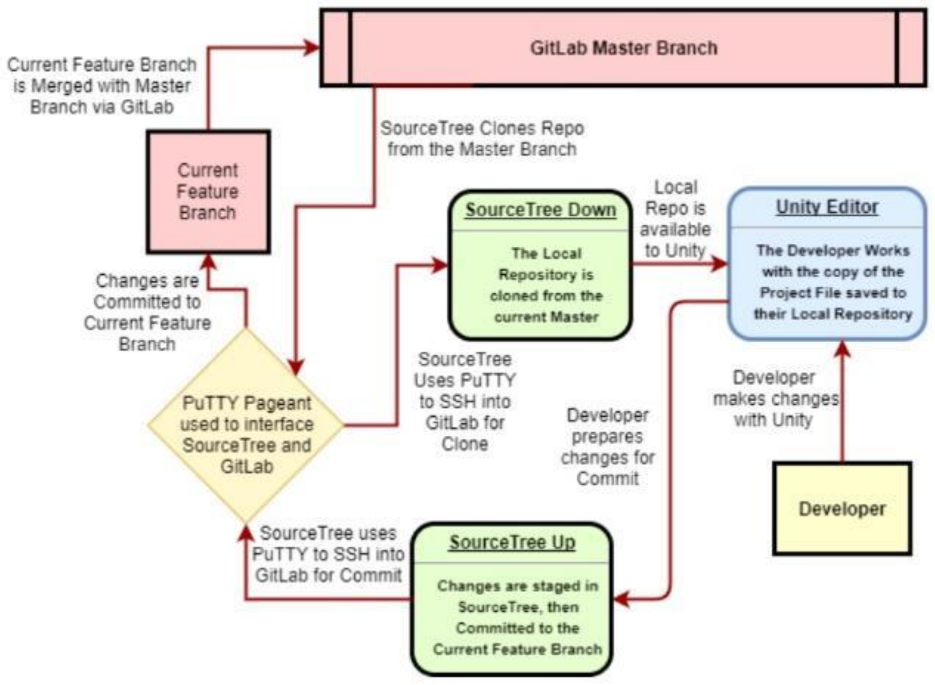

GitLab Master Branch
Current Feature Branch is Merged with Master Branch via GitLab
SourceTree Clones Repo from the Master Branch
Current Feature Branch
SourceTree Down
The Local Repository is cloned from the current Master
Local Repo is available to Unity
Unity Editor
The Developer Works with the copy of the Project File saved to their Local Repository
Changes are Committed to Current Feature Branch
PuTTY Pageant used to interface SourceTree and GitLab
SourceTree Uses PuTTY to SSH into GitLab for Clone
Developer prepares changes for Commit
Developer makes changes with Unity
Developer
SourceTree uses PuTTY to SSH into GitLab for Commit
SourceTree Up
Changes are staged in SourceTree, then Committed to the Current Feature Branch

Figure 8: Development Pipeline (Team Silver, 2017)

5.1 Agile Development

Team Silver will employ the agile development process to deploy individual features. Developers will work in sprints to plan, develop, test, and deploy new features. A demo application will be approved by the group and then pushed into the master branch of the game. After a successful push, team leads will determine the next development aspect the game and continue with the new sprint. The process iterates until the core gameplay has been developed.

5.2 Work Management

There are currently nine developers building PolyMorpher. There are three teams, with three members in each. Within the sub-team there is a leader who manages the team. The sub team leaders report to the project manager during the weekly stand up meetings. This method of team division will evenly distribute development work to everyone and not allow team members to feel overwhelmed or stuck. Having sub teams will also allow rapid development and testing within teams. Using sub teams and version control allows individual developers to be efficient and not allow single issues to block development.

[This space is intentionally left blank]

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.

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.

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.

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

Ubuntu: open-source Linux operating system.

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

Virtual Machines: an emulation of a computer system that provide 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.

[This space is intentionally left blank]

7. References

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

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-5KdQEdqLUPd nBFUnp2V05uMEE%22%5D,%22action%22:%22open%22,%22userId%22:%22108692003133590583047%22%7D#G0B-5KdQEdqLUPdnBFUnp2V05uMEE

Everitt, C., & Dang, D. (2017, September 24). currentProcessFlow. In draw.io. Retrieved December 21, 2017, from https://www.draw.io/?state=%7B%22ids%22:%5B%220B3Bc95zBWXg9TFZ6X0FMU1NTdEk%22%5D,%22action%22:%22open%22,%22userId%22:%22108692003133590583047%22%7D#G0B3Bc95zBWXg9TFZ6X0FMU1NtdEk

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

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

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

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

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

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

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/1oY9lkSAHvg2OIR kljYJNZWCqVTbiw45STKglsJUQjJI/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-1vSidnjCKAu VEtKshHkyO7A-OfW3qWIKRkxcp0em412WwL1ig6SFmnqrMUyHr8-FMvzvaRjmcK YiCytq/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-1vSllslBDmSvRfMI9nbrp0R mRaPRsHNz7YWDfKNiF5sg15cp7ycQ774MuMgm4G4qhR6hohTiUQrrjRdo/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/