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 2 - PolyMorpher Product Specification

1 Introduction

1.1 Purpose

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.

The targeted end users of PolyMorpher are students who are currently enrolled in the Computer Science (CS) degree program at ODU. Since the game is currently being made by ODU CS students, control of the game remains within the ODU computer science department. If PolyMorpher proves to be a successful education resource, the department may consider selling the game to other universities, colleges, publically, or educational institutions. While originally PolyMorpher will be made for CS students at ODU, it can be played by anyone who is generally interested in learning programing.

PolyMorpher will be a general introduction to OOP concepts. The game will introduce important concepts and provide puzzles for students to solve using OOP concepts. As the levels progress, students will build their toolkit and solve more problems in increasing difficulty. PolyMorpher will not go too in depth about technical programming concepts. It exists to help students understand programming organization at a conceptual level and apply it to future programming practices.

1.2 Scope

Programming is intimidating for the uninitiated. As a result, many first time computer programming students at Old Dominion University (ODU) end up dropping out or switching majors. Figure 1 displays the current process flow of an entry level computer science student at ODU. Existing tools teach programming and syntax but fail to teach Object Oriented Programming concepts. 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. This is the problem PolyMorpher will address. 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

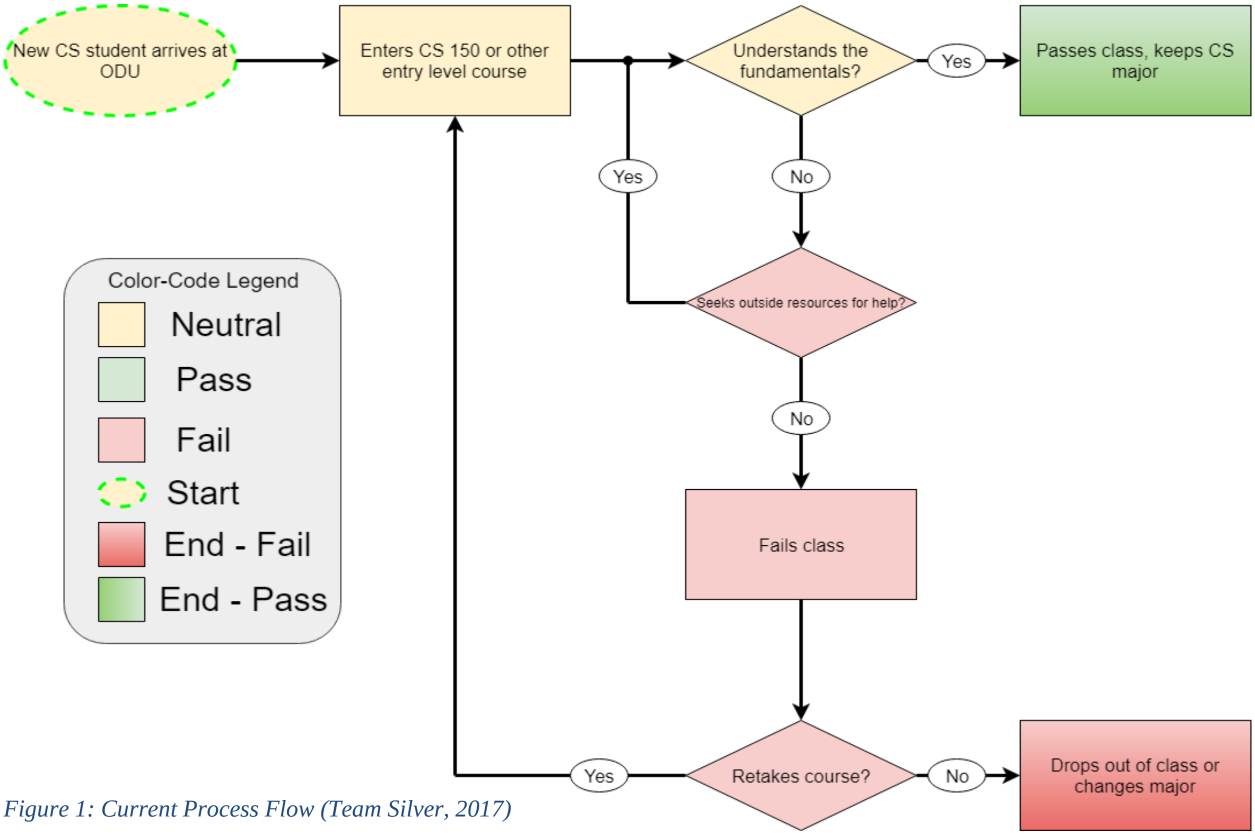

New CS student arrives at ODU
Enters CS 150 or other entry level course
Understands the fundamentals?
Yes
Passes class, keeps CS major
Yes
No
Color-Code Legend
Neutral
Pass
Fail
Start
End - Fail
End - Pass
Seeks outside resources for help?
No
Fails class
Yes
Retakes course?
No
Drops out of class or changes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

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

students to solidify the fundamentals needed in order to pass the class and continue with the computer science curriculum.

PolyMorpher will solve these problems through an interactive Management Simulator and Tangible User Interface (TUI). Video games have been proven to enhance interest and learning comprehension among new learners. The immersive nature of games allows users to quickly learn content in a fun and engaging way while working to solve interesting problems, thus increasing engagement and retention. Players can also progress at their own pace without an instructor.

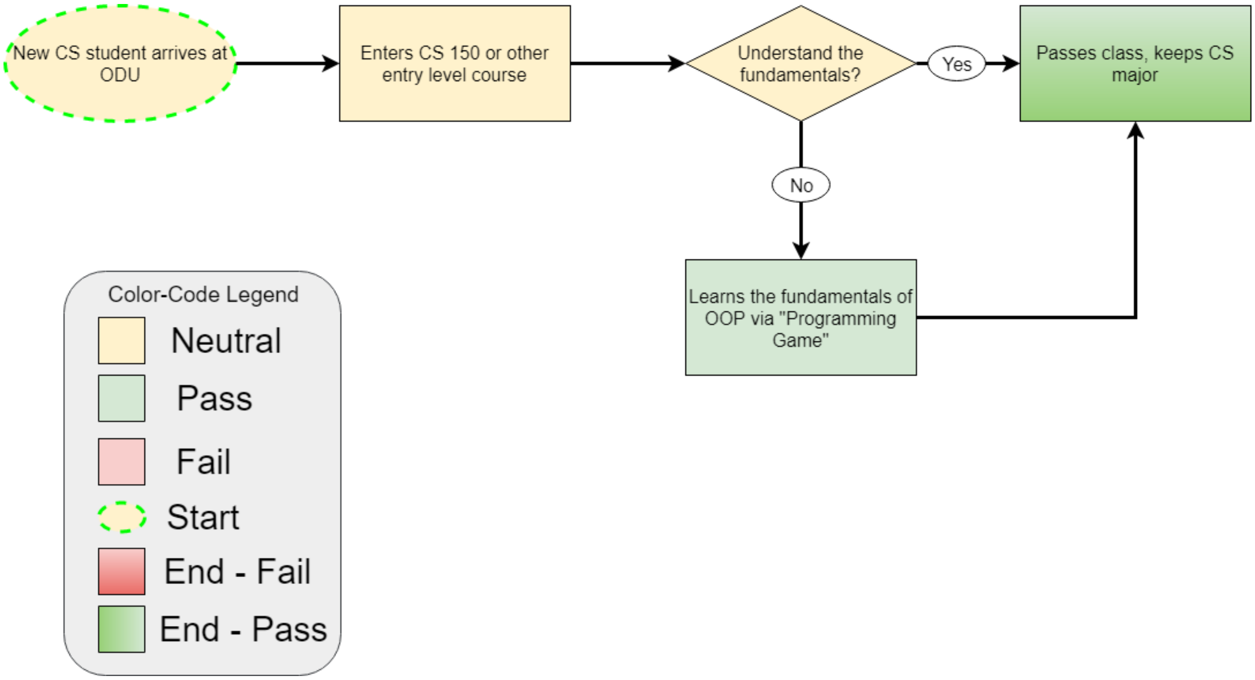

New CS student arrives at ODU
Enters CS 150 or other entry level course
Understand the fundamentals?
Yes
Passes class, keeps CS major
No
Learns the fundamentals of OOP via "Programming Game"
Color-Code Legend
Neutral
Pass
Fail
Start
End - Fail
End - Pass

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

[This space is intentionally left blank]

1.3 Definitions, Acronyms, and Abbreviations

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.

1.4 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-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:%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/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-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-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/

1.5 Overview

This product specification paper provides the general software configuration of PolyMorpher. The information provided in the remaining sections of this document includes a detailed description of the hardware, software, and external interface architecture of PolyMorpher prototype.

2 General Description

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.

2.1 Prototype Architecture 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. 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. The PolyMorpher prototype will have three main components: PolyMorpher Website, PolyMorpher application, and the Unity File Structure.

- PolyMorpher Website: The website will contain the PolyMorpher game application that users can download and execute on their local machine. The website will also provide information about the game, the team, and a description of the product.
- PolyMorpher Application: PolyMorpher will be a downloadable executable that launches the PolyMorpher game.
- The Unity File Structure: The file structure for the PolyMorpher game will be included in conjunction with the executable. There will be a “Streaming Assets” folder that can be accessed by the player to add and modify code game objects.

[This space is intentionally left blank]

The three components are shown in Figure 3.

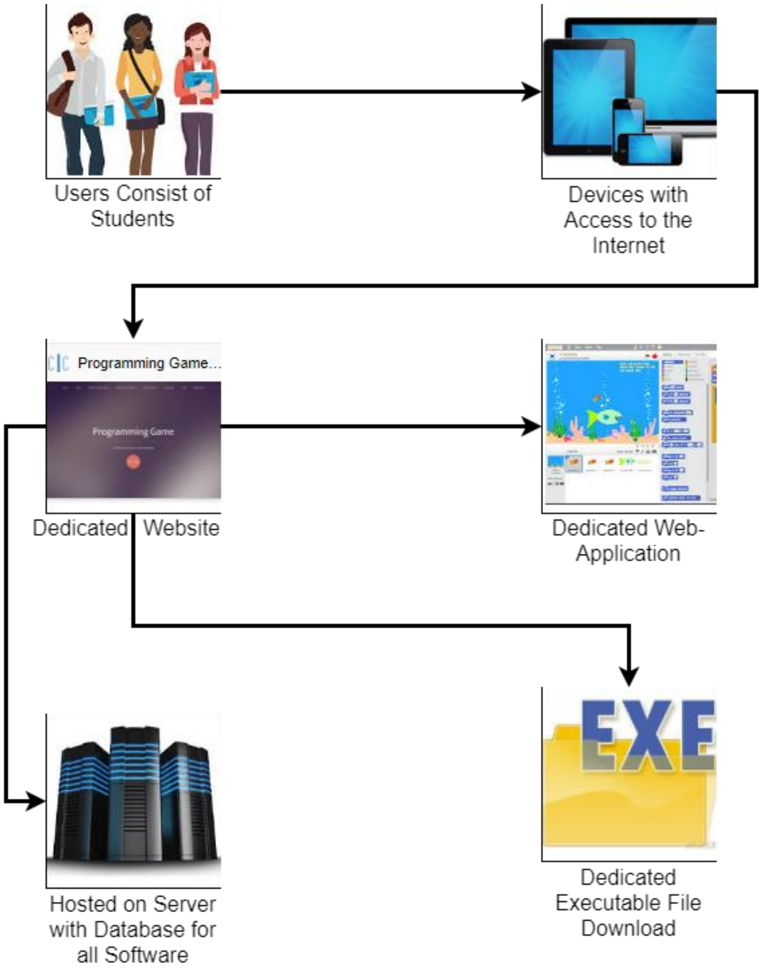

Users Consist of Students
Devices with Access to the Internet
Programming Game...
Programming Game
Dedicated Website
Dedicated Web-Application
EXE
Hosted on Server with Database for all Software
Dedicated Executable File Download

Figure 3: PolyMorpher Architecutre (Team Silver, 2017)

2.2 Prototype Functional Description

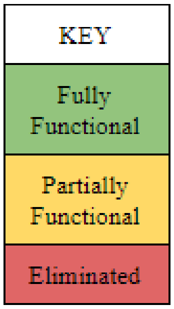
Since PolyMorpher is being developed within the Unity Software Development Kit (SDK), it will be compatible with Windows, MacOS, and Linux operating systems. Table 1 and key show the core features will and will not be implemented in PolyMorpher. The core feature of PoloyMorpher will include a fully functional 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 1: 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 2: Prototype Features Keys (Team Silver, 2017)



KEY
🟩 Fully Functional
🟨 Partially Functional
🟥 Eliminated

One of the key differences in the prototype and the final version of the game is the number of languages being used and the introduction of a web game played only on the browser.

2.3 External Interfaces

The PolyMorpher prototype will have five types of external interfaces: Hardware, Software, User, API Book, and Compiler interfaces. The user will also require a suitable device with the minimum required hardware specifications to run the game.

2.3.1 Hardware Interface

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.

2.3.2 Software Interface

All the software needed to run will be hosted on the CS servers provided by the ODU CS department. The operating systems required to run the game will be Windows 7+, Linux, or Mac OS.

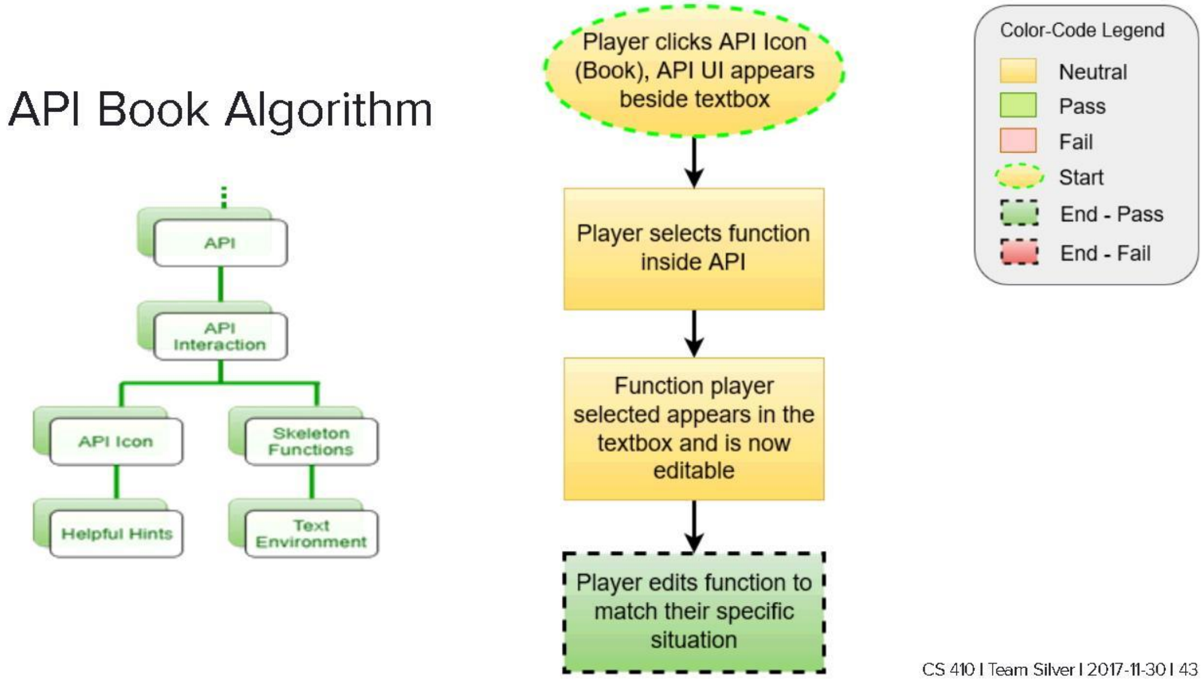
2.3.3 User Interface

There will be three primary interfaces that the user will need in order to play PolyMorpher.

- Computer Screen: The computer screen will be used to display the game components to the player. The player will receive information from the screen and react accordingly. The resolution will need to be large enough to display all the information the has displays.
- Computer Mouse: The computer mouse will be used to by the player to interact with the in game world. The mouse will allow users to "interact" with objects within the game world and "hack" them in order to solve the puzzle. The user will be required to navigate and click the mouse. There is no mouse specification to play PolyMorpher.
- Computer Keybaord: They keyboard will the main interface that the player uses to program and create scripts. The keyboard will also be the main interface to control the in game character.

2.3.4 API Book Interface

The API Book Interface will provide the player with a host of programming tools used to progress though the game. The tools will help the user complete the puzzles needed to advance through the game. When the player chooses to enter the API Book Interface shown in Figure 4, a book will appear and players can select predetermined scripts too 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

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

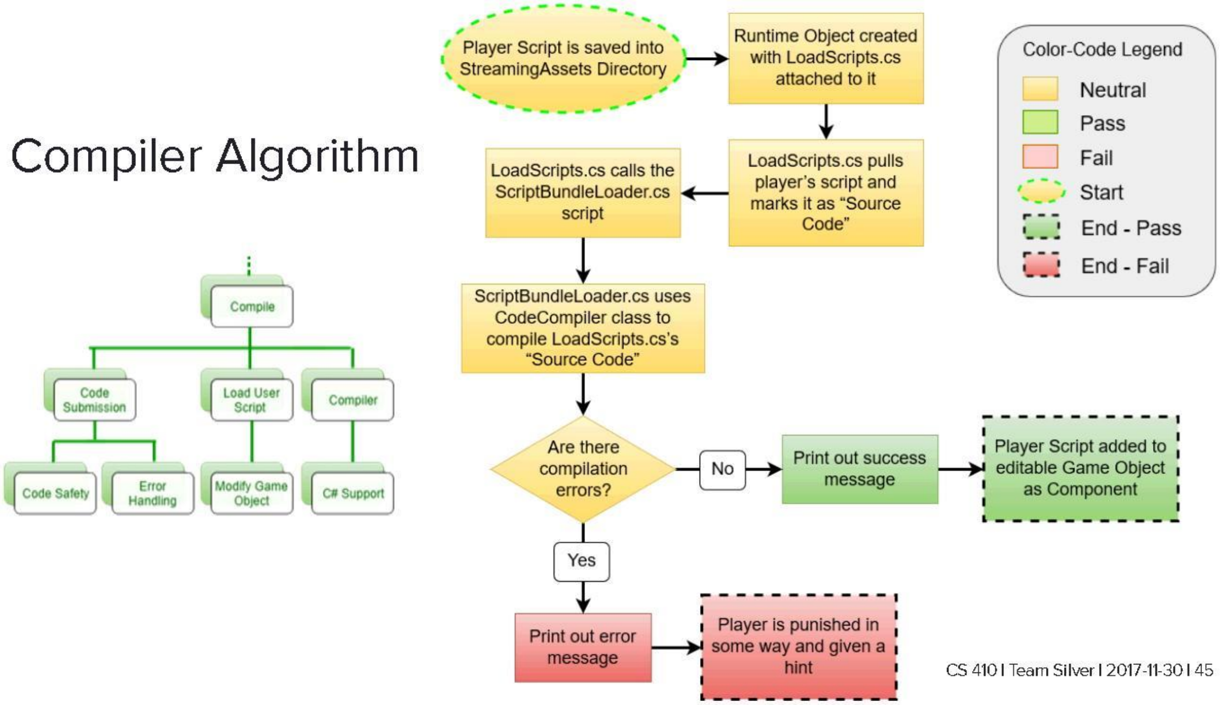
🟨	Neutral
🟩	Pass
🟥	Fail
○	Start
▨	End - Pass
▨	End - Fail

CS 410 I Team Silver I 2017-11-30 I 43

Figure 4: API Book Algorithm (Team Silver, 2017)

2.3.5 Compiler Interface

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. The Compiler Interface will have a text box on the display screen that contains partially complete C# code that the user has to fill out in order to become correct. The Compiler Interface will become available when the user clicks the “morph” button on an object. If the user correctly completes the missing code, then the puzzle will be solved and the user will be allowed to move onwards. If the program is incomplete or incorrect the compiler interface will allow the user to try again.



Compiler Algorithm

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 5: Compiler Algorithm (Team Silver, 2017)

[This space is intentionally left blank]