Lab 1 - Polymorpher Product Description

Casey Batten

Old Dominion University

Version 1

CS411W

Professor Thomas Kennedy

January 29, 2018

Author Note

Casey Batten, 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 Casey Batten, Department of Computer Science, Old Dominion University, Norfolk, VA 23529.

Contact: CBATT015@odu.edu

Table of Contents

Contents

1 Introduction

Computer Science is a field that is constantly expanding, and every day becomes a more integral part of daily life. Because of this, the demand for workers in the field and the number of students seeking education to fill those very positions have both markedly increased over the course of more than a decade. However, despite the advancement of technology and programming practices, many of the skillsets necessary to be successful in this rapidly growing field continue to have a sharp learning curve and barrier of entry. From that problems arise and solutions to meet those problems must be developed to aid potential Computer Scientists and all parties interested in the field meet that curve and overcome it. Polymorpher is aimed at being one such solution, as it helps reinforce integral parts of the software design and programming process in a format that is more easily digested by a wider audience.

1.1 Team Members

ODU Team Silver and the team is composed of:

Matthew Tuckson, Project Manager Colten Everitt, Website Admin

Daniel Dang, Web Developer Joel Stokes, Game Developer

Kevin Santos, Database Management Nathaniel Dearce, AI Design

Peter Riley, Software Engineer Tyler Johnson, Software Engineer

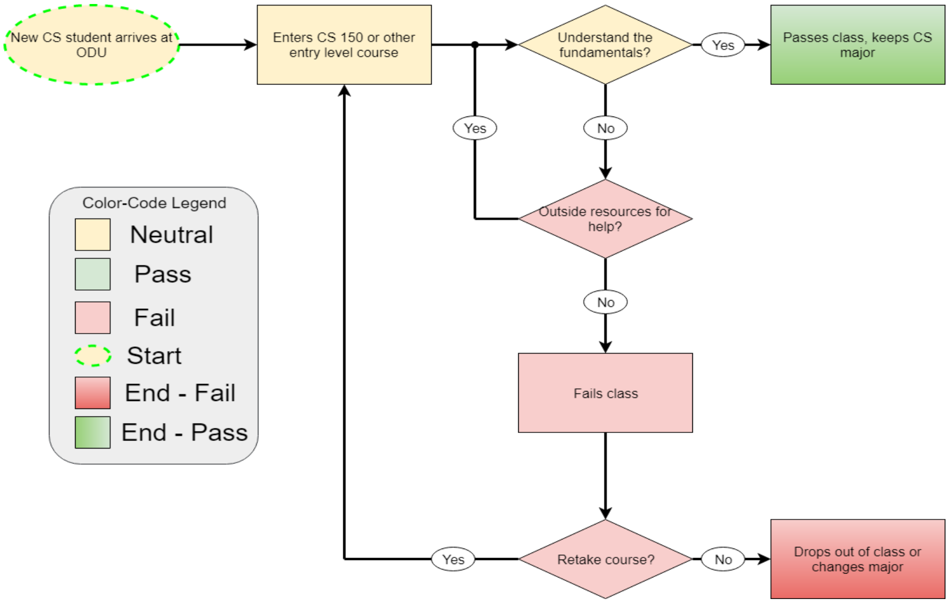
Casey Batten, Unity SDK Specialist

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 Object-Oriented Programming (OOP) concepts and problem-solving skills.

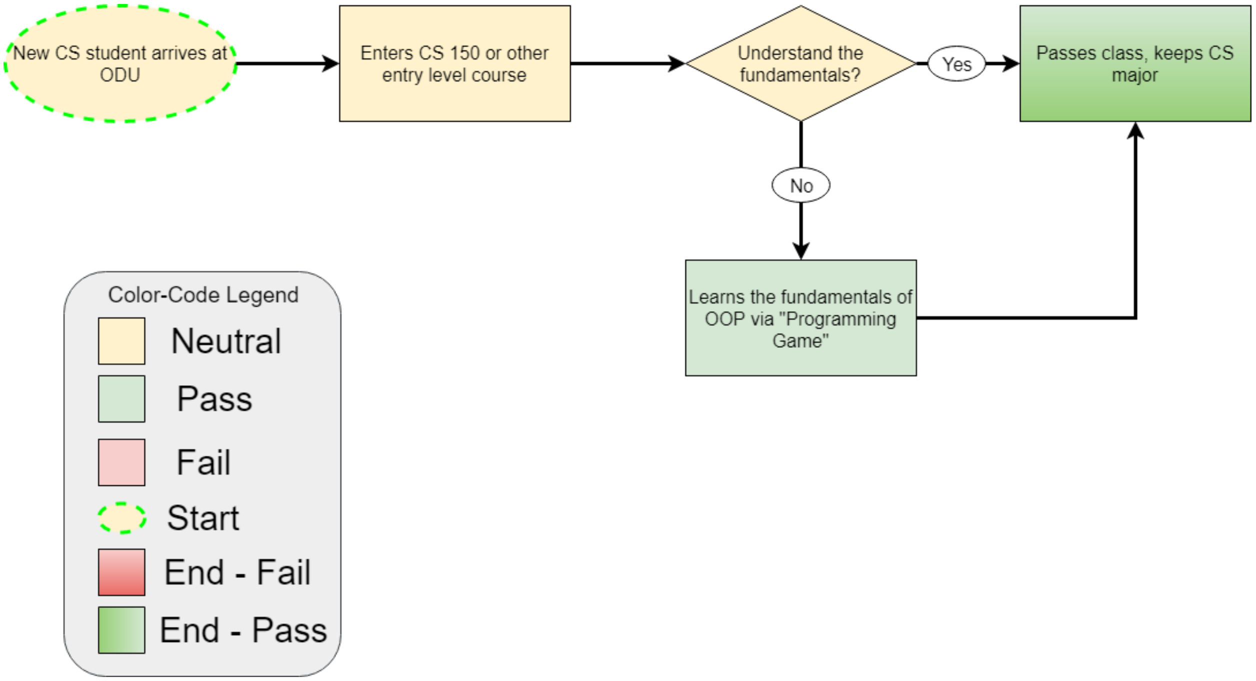
1.3 Problem Characteristics

The problem characteristics detailed above show a simplified breakdown of the issue from the perspective of a student in the Computer Science curriculum. Over the course of a semester the team has theorized that this basic outline matches the pattern for how students may fall in to a negative cycle eventually leading to the possible dropping of the class or worse the entire major. These students are attempting to progress through a curriculum without truly understanding it or having the tools to overcome its challenges. This is where the need for a support system or tool to handle this negative cycle becomes relevant.


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

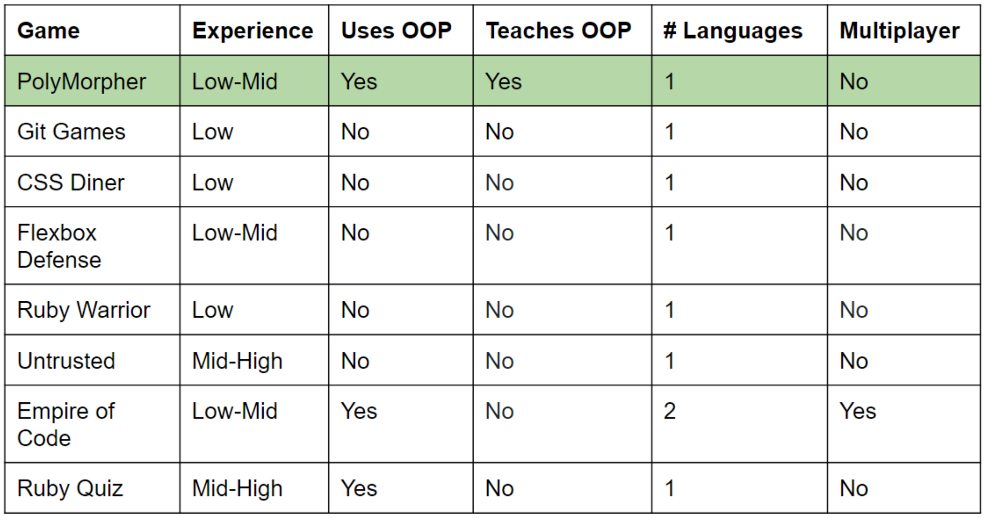
1.4 Solution Characteristics

The primary characteristics of this solution revolve around the idea that the programming-based game, Polymorpher, will help not only reinforce but offer an alternative understanding of the core principles of Object Oriented Programming. This approach of giving the player a new way of thinking on a concept which is particularly integral to software design is what is believed will help them escape the negative cycle covered in the problem characteristics diagram.


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

The strengths of the intended method include a low barrier of entry and a focus on Object Oriented Programming and design specifically. The low barrier of entry will allow users inexperienced in higher level programming, especially Computer Science students trapped in the negative cycle, to pick up and rapidly adapt to the new toolsets the team intend to introduce them to. These toolsets will help elaborate the tenants of Object Oriented Programming and give the user a new way of thinking regarding the application of these concepts.

The weaknesses of the intended method are focused on those of security and the nature of the experience itself. On the matter of the player’s experience, there currently is only one supported language and no planned multiplayer as from a developmental standpoint it would pose serious challenges and risks. One such risk ties in to the security weakness, given that the player will be executing raw code which could potentially put their system at risk as well as any other user they theoretically networked with. To mitigate that risk the team has decided to move away from multiplayer aspects and focus on a single player experience, which will be discussed in further detail in section 4.



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

2 Polymorpher Product Description

2.1 Goals and Objectives

The primary goal of Polymorpher is to create a video game which teaches its player Object Oriented Programming using an expansive in-game toolset. This toolset will be used by the player to not only manipulate the game, but also garner a better understanding of their actions in the game and in effect of Object Oriented design as a whole.

2.2 Key Product Features and Capabilities

Polymorpher is a puzzle-based coding themed platformer that utilizes a unique player input system in which the user can select and edit the source code, and by extension the behavior, of objects within the game environment to solve puzzles and overcome challenges. The amount of freedom given to the player regarding how they can manipulate an object's behavior is quite significant given that any changes to an objects source code will be rendered in the game in real time, allowing the player to see and better understand the effects of certain coding practices.

As previously stated the coding practices that will be focused on will be those that most closely relate to Object Oriented Programming. The player will be subjected to a number of puzzles and level-based challenges that focus on specific pillars of OOP, the intention behind this being that if the player can be tasked with creative thinking in relation to these concepts they will naturally better understand them inside and outside of the game's context. This is also how the team intends to make Polymorpher different than the competition. Other programming themed games exist, and some may even serve a similar purpose, but the goal behind the project is to achieve a balance between a fun experience and an educational one where others fail to. Polymorpher will give the player a varied enough experience to not only continue to hold their interest from an entertainment standpoint but also to inform and educate them from a software engineering standpoint.

2.3 Major Components (Hardware/Software)

The projected minimum system requirements are intended to fall within the range of an Intel i5 processor, approximately 2 Gb of RAM, and the Windows 7 Operating system. Alongside this the game will be delivered as a downloadable desktop application in the form of an EXE file, requiring an internet connection. Currently there is not a projected hard-drive space requirement.

3 Identification of Case Study

3.1 End-User: Students

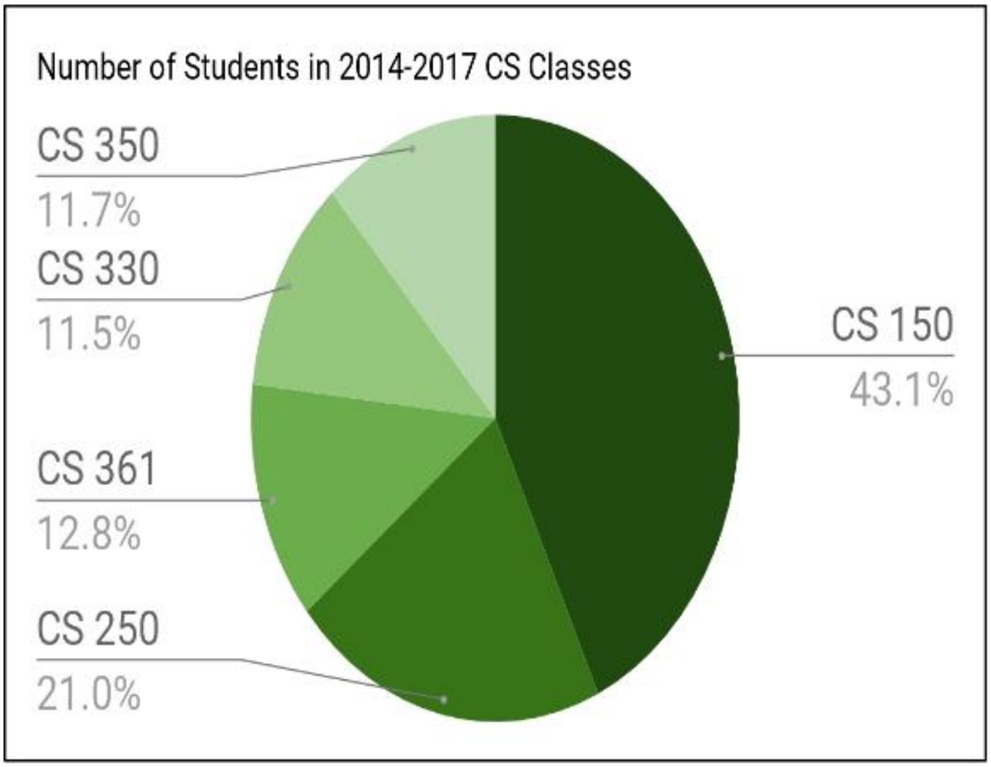
At the current time the primary End-User for this product would be students in the computer science degree path, or in any computer science classes, at Old Dominion University. Beyond this of course students in similar degree paths or classes at other universities and colleges would fall under the same End-User classification, assuming the products completion and success at Old Dominion. Finally, the End-User could also be any individual with a genuine interest in or desire to learn Object Oriented Programming.

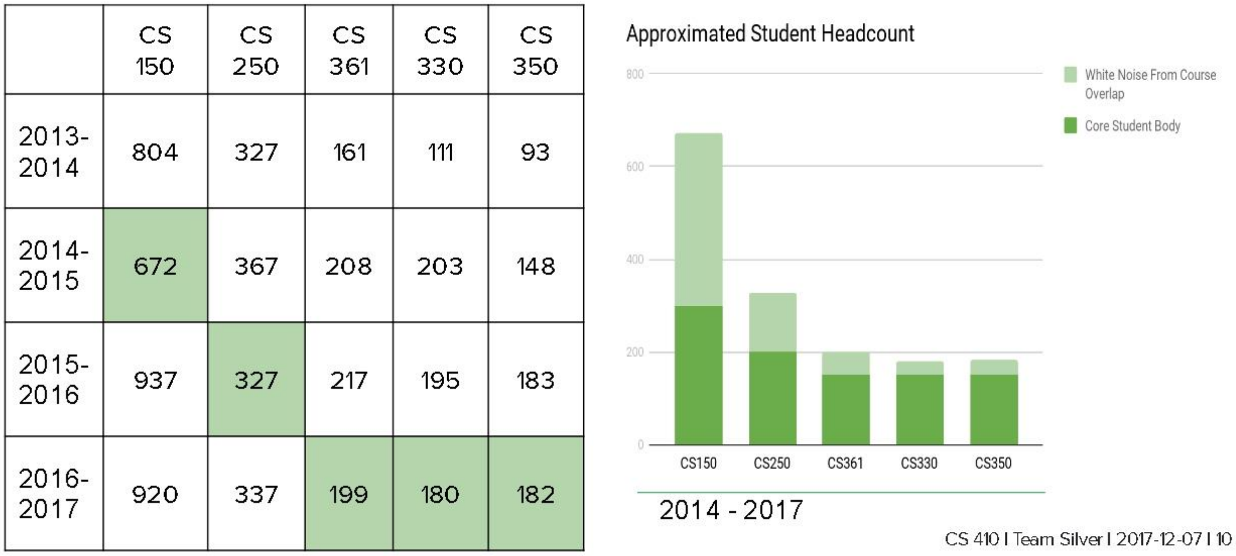
3.2 Customers

The customer base is, for the most part, the End-User. Aside from being focused on individuals however, Polymorpher also may appeal to parties interested in using it as a learning tool on a larger scale. Old Dominion University may aim to offer it as training software to interested students. Other colleges or universities may also choose to acquire the software in bulk to provide it as an available service to their students.

3.3 Why Make this for Students

As stated in the problem statement, the intimidating nature of programming can potentially cause new CS students to drop a class or switch majors. The figures above have been propagated with data pulled from the ODU Factbook, which points to a significant drop in the number of students following the CS course curriculum from 2014 to 2017. A portion of this can be attributed to white noise due to course overlap between other STEM degrees and Computer Science, however in the later courses where white noise is much lower the disparity in the size of the student body and the initially projected class size is still significant enough to support the conclusion proposed in the problem statement.





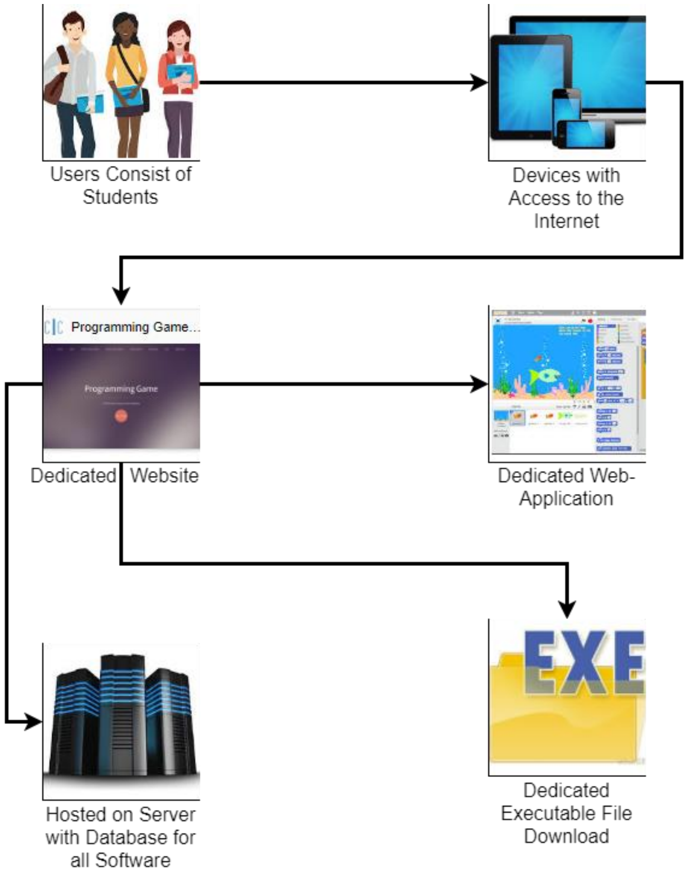
	CS 150	CS 250	CS 361	CS 330	CS 350
2013-2014	804	327	161	111	93
2014-2015	672	367	208	203	148
2015-2016	937	327	217	195	183
2016-2017	920	337	199	180	182

CS 410 I Team Silver I 2017-12-07 I 10

4 Product Prototype Description

4.1 Prototype Architecture (Hardware/Software)

The Prototype will consist of a standalone desktop application that can be downloaded from Team Silver’s dedicated website. This Prototype will be an education focused game that will cover the core tenants of Object Oriented Programming, with light coverage of the prerequisite knowledge to work with those principals.


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

4.2 Prototype Features and Capabilities

The primary intended features of Polymorpher are the ones that pertain to the core tenants of Object Oriented Programming. These core principals will be organized into a level structure that will provide the player with puzzles and challenges to help not only reinforce these integral concepts but expand upon them in unique ways. Alongside the game's core content is the implementation of a single programming language, C#, which the team decided to use because of its similarity to other popular OOP languages and the relatively low proficiency level needed to learn the language. Combined, these factors will assist the user in mastering the skills necessary to become a successful software developer.

These core principals break down in to six main groupings in the games level structure, starting with two sections of tutorials to covering the controls of the game and primary systems of interaction, and the basic syntax necessary to use C#. Following this, the game presents the player with four primary challenge levels that follow each of the core principals of Object Oriented Programming. The section on Polymorphism will focus on introducing the player to concepts involving objects changing in relation and reaction to other objects around them and the behavior of the player, all revolving around the object source code that the player will be interacting with directly. Inheritance will be taught through in game examples of inheriting states and object lineage in the game's context, shown through class interaction in the code the player will be working with. The Abstraction section will be used to build off the Polymorphism and Inheritance sections, introducing the player to abstract data types and type manipulation. Beyond this the section on Encapsulation will introduce the player to closed off event sequences with incremental changes based on differing conditions under the players control, allowing them to observe changes within and outside of encapsulated objects based on their own work with the object source.

Beyond the core lesson content of Polymorpher, the team also has plans for expanded content to fill out the gaming experience more. The first of thing to fall into this development category is the implementation of game assets, particularly art and sound assets. Given that the team currently lacks dedicate sound and digital artists, these resources would have to be procured either through royalty free online resource databases or through art commission and licensing outlets. With these assets are secured the final product will not only look and feel more professional, the overall experience will be more appealing to the End-User and have a more lasting effect. Alongside this the team is also dedicated to giving Polymorpher some variety of dedicated narrative to help tie together the lesson content distributed through out the game. This will help increase player engagement, while also smoothing the games structure linear structure and pacing. Finally, there are gameplay stretch goals the team is dedicated to pursuing, including a sandbox level for the player to have full access to all the tools of gameplay available in the individual lessons in a single unified area. To expand on that, a "create and save" function may also be implemented into the sandbox feature with enough development time, paired with a method of exporting saved levels to share user-created challenges with other players. This would give the game a form of pass-off multiplayer, expanding replayability and engaging the player community with each other in new and creative ways.

While planning out Polymorpher's development, the team gradually eliminated certain development paths due to limitations and redirected design goals. One of the eliminated capabilities of Polymorpher was active multiplayer gameplay. This option was ruled out as a serious security risk, given that the players would be executing raw code in a shared environment and could potentially cause harm to one another's PC. Another ruled out capability was having Polymorpher function as a web-based application in browser. This option is, at the time being, impossible at a technical level. Not only does the current portable compilation method pose a serious design strain but allowing raw code to be compiled on a website hosting the game could negatively affect not only the user's personal game but any users using that web service and the website itself. The removal of these features has helped focus the direction the team intends to take the project and has lessened the potential risks involved with allowing the End-User as much control over the game's content as Polymorpher does.

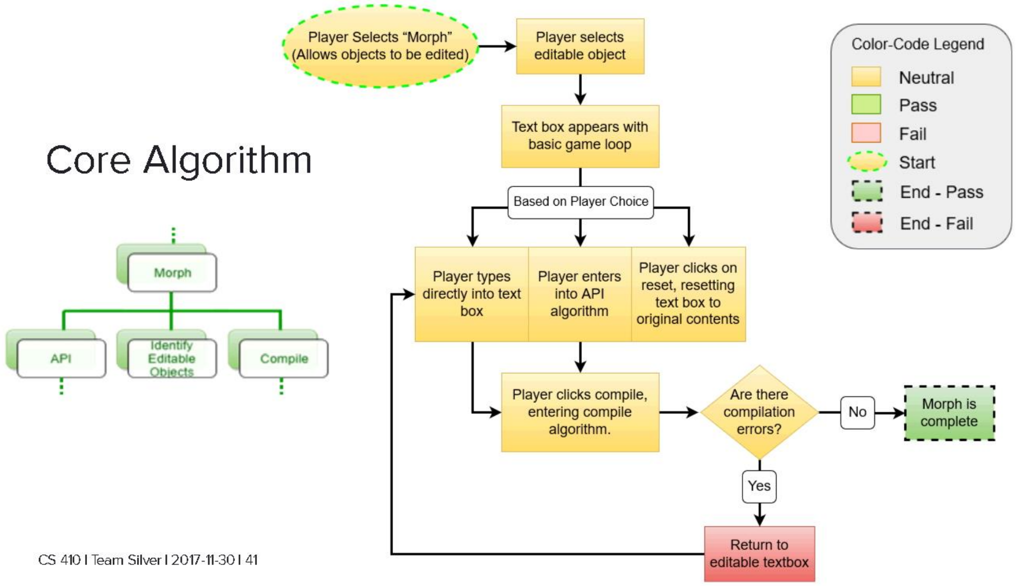
KEY
🟩 Fully Functional
🟨 Partially Functional
🟥 Eliminated

Elements	**Description**	**Real World Product**	**Prototype**
Teaches Polymorphism	Provision of a single interface to entities of different types	🟩	🟩
Teaches Abstraction	Technique for arranging complexity of systems	🟩	🟩
Teaches Encapsulation	Building of data with the methods that operate on that data	🟩	🟩
Teaches Inheritance	When an object or class is based on another object or class, using the same implementation	🟩	🟩
Single Language Taught	A single programming language will be focused on C#.	🟩	🟩
Single Player	Focused on an experience targeted to interact with only one player	🟩	🟩

Downloadable .EXE File	Desktop application version of the game	🟩	🟩
Game Assets	Primary components that are used as building block to construct the more complex features and levels of the game	🟩	🟩
Developed Story	Narrative used to drive progression or direct player throughout a more guided/linear experience	🟩	🟩
Portable Compiler	Code compiler used to run player-made code on the fly in game	🟩	🟩
Tutorial Section	Precursor series of levels meant to help the player adjust to the in-game toolset given to them and also prep them with knowledge of the language(s) they will be working with	🟩	🟩
Multiple Platforms	Version support for multiple operating systems (Windows, Mac OS, Linux)	🟩	🟩
Sandbox Level	Open level where the player has access to all tools at once and can build their own level sequences and puzzles	🟩	🟨
Player-Made Content	Variant of Sandbox Level, potentially allows the player to share custom levels with one another	🟩	🟨
Multiple Player	An experience geared toward multiple players interacting with a game environment together	🟩	🟥
Web Application	Web based version of the game running in-browser	🟩	🟥
Multiple Languages Taught	Alternative programming languages for the player to use and learn in-game	🟩	🟥

4.2.5 Algorithms

The Core Algorithm of Polymorpher is what commands the central mechanic of gameplay. The system revolves around a balance of using the in-game API and the Morph function. When the player selects the “Morph” function on the UI, they can then select pre-determined objects within a level or puzzle and begin interacting with their source code. This allows the player to change the objects behavior in a way that will assist them in overcoming the puzzle presented to them. The UI systems the player interacts with the most will be the Coding Interface, which functions as a sort of in-game IDE, and the API Book. The Coding Interface will allow them to write and compile code, which they will learn through the API Book and its subsequent entries provided by the development team to aid the player through each challenge. Assuming the script compilation is successful the “Morph” function will be completed, and any behavioral changes made by the player will be rendered in real time. Otherwise, an error report will be sent to the player and they will be routed back to the Coding Interface.

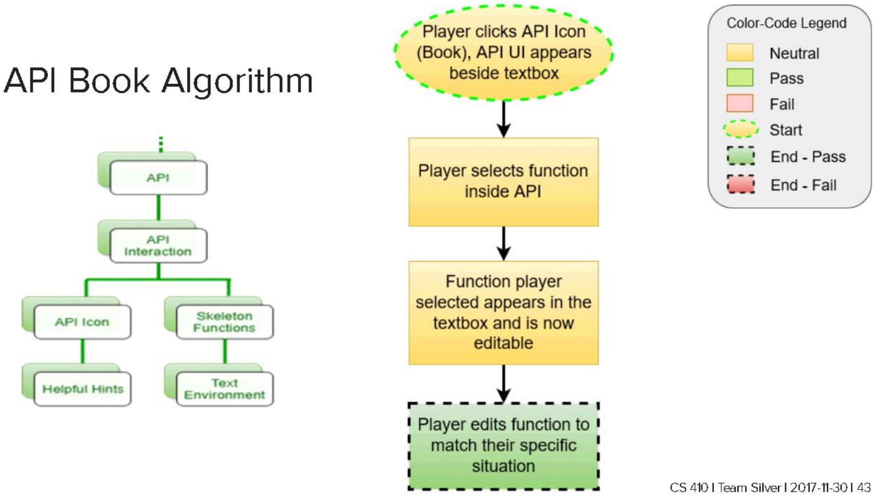


Core Algorithm

Player Selects "Morph" (Allows objects to be edited)
Player selects editable object
Text box appears with basic game loop
Based on Player Choice
Player types directly into text box
Player enters into API algorithm
Player clicks on reset, resetting text box to original contents
Player clicks compile, entering compile algorithm.
Are there compilation errors?
No
Morph is complete
Yes
Return to editable textbox
Color-Code Legend
Neutral
Pass
Fail
Start
End - Pass
End - Fail
Morph
API
Identify Editable Objects
Compile

CS 410 | Team Silver | 2017-11-30 | 41

The API Algorithm, which is runs as a subsequent result of the Core Algorithm, is the primary method of information delivery from the developers to the player. It will be host to the API Book, which functions based on certain interactions by the player. While navigating the API Book the Algorithm will sort in related helpful hints, and recommended functions for certain puzzles and scenarios. On top of this, when viewing the page related to a function listed within the API Book the player will be shown example psuedo-code of the proper use of the function and be given the option to import the function to any script currently open in the Coding Interface with the press of a button.



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

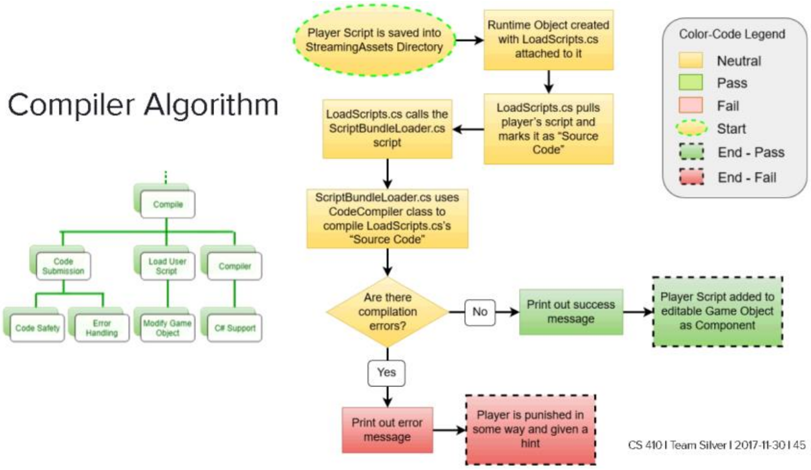
Player edits function to match their specific situation

Color-Code Legend

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

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

The Compiler Algorithm is the most important Algorithm to the functionality of Polymorpher from a gameplay and mechanical standpoint. It handles the process of object selection, passing the selected object's source code forward to the Coding Interface for editing. Following a compilation request, the Algorithm will save and identify the new source code through the "LoadScripts.cs" script and pass it on to the "ScriptBundleLoader.cs" script. Here it will be marked for Assembly and Compilation. If compilation is successful it will then proceed to attach the newly made script to the initially selected game object and being the re-rendering process, allowing its new behavior to take effect. Otherwise, it will trigger and error message for the player via the Unity Error and Log scripts, informing them of the nature and exact line of the initial error. This intricate system is possibly the most important function in the entire backend of the software and is what gives the player the freedom to manipulate the core of the game's functionality in any potential way they can imagine.



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
Compile
Code Submission
Load User Script
Compiler
Code Safety
Error Handling
Modify Game Object
C# Support

CS 410 | Team Silver | 2017-11-30 | 45

4.3 Prototype Development Challenges

One of the most challenging developmental tasks to overcome in game design is continuity in regard to the game's overall structure, in it's tone, in the nature of it's levels and challenges, and most importantly in the quality of the game. If the quality of any one of these factors diminishes or does not match the standards set by the rest of the game, it is very likely to put off the End-User and reflect negatively on the overall experience of the game. Because of this, keeping a certain development standard among all members of the group and having clear and consistent communication with all members actively involved in Polymorpher's development is integral to the success of the product.

Beyond that, a serious challenge that plagues the design process of every video game across the board is bug testing. Bugs or consistent errors in a game can range from mildly disruptive to experience-ruining, and in some serious cases can even break the software's capacity to function as a whole. Therefore iterating on problem areas and refinement of every major functional component of Polymorpher will be incredibly important in ensuring quality of life in regard to the game's continued value.

Another general issue for game development is the issue of maintaining player engagement. If your player or even player base cannot find value in continuing to use and support your game, it will find itself short lived and incapable of standing on its own two legs on the market. For these reasons it is not only important to ensure that the core experience you are providing is of significant quality, but that the replayability of your product enables its player's and community to grow and remain passionate about the product's existence.

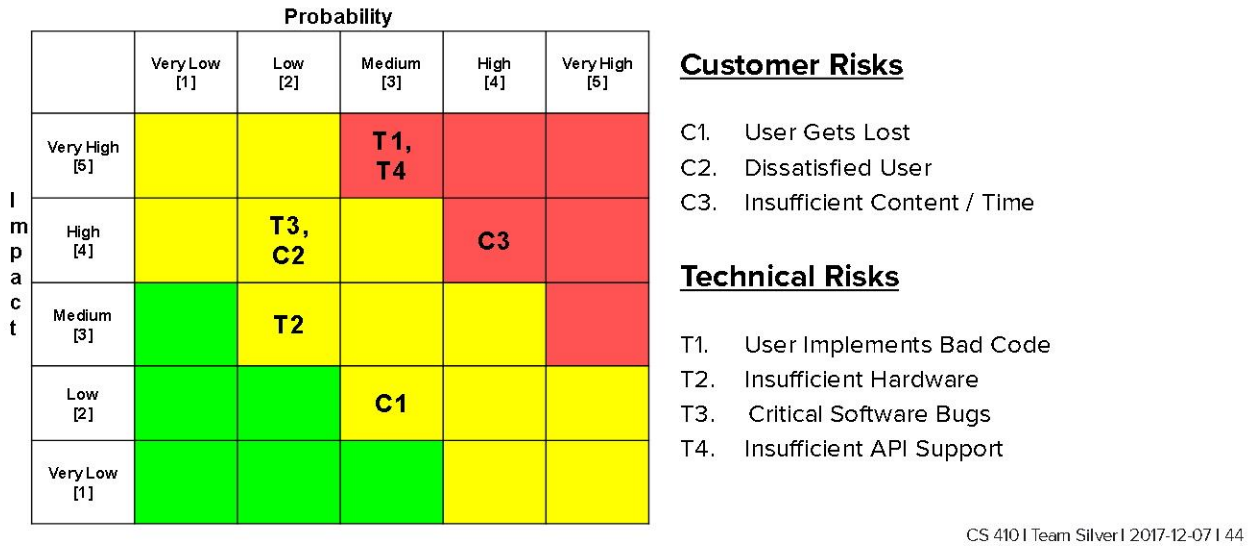
One major risk specifically relevant to Polymorpher is ensuring that the game is educating well and has enough content to be considered beneficial to its users. At its core this is educational software meant to enrich its users in their capacity to design software, so if it doesn't achieve the goals set out by it's designers or in any way fails to educate its audience its purpose for existence becomes invalidated.

One of the best ways to prevent each of these major development risks is through the use of playtesting. If the team ensures that we as developers continue to enjoy using our product, we can move on to having others play it in focus groups. The team can analyze feedback from these playtester focus groups, likely made of volunteers from the student body at Old Dominion University, and use their feedback to refine and better the overall quality of Polymorpher.

4.4 Risk Mitigation/Risk Matrix

Of the Technical risks, the two most impactful are the End-User implementing malicious code and insufficient support from the in-game API. The risk involved with malicious code is one that is intertwined into the core functionality of Polymorpher. Since the user can execute almost any variety of code they want inside the game's environment they could very easily run code that could harm their PC. The ways this has been mitigated vary from removing the possibility of online multiplayer, and potentially providing a full walkthrough of the puzzles and their solutions while providing developer-approved code solutions that will not harm the user's PC. As for the second major technical risk, if the game has an insufficient API available to the player they may find themselves incapable of adapting to the situation and solving the puzzles in the game. To avoid this the team intends to playtest intensely and iterate on the API provided to the player until it is considered sufficiently capable of providing the player with a satisfactory experience.

Of the Consumer risks, the primary concern is that there is insufficient content and/or insufficient time from a developer standpoint to provide a full and fulfilling experience. Mitigation for that once again involves not only iteration on the product throughout development but also focused playtesting. To confront the issue of a lack of development time the team has divided the workload in to concise and manageable portions that have been distributed into sub-teams with three members each. This method will increase the entire groups overall efficiency and capacity to produce a complete product given the current development timeline.



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

5 Development Pipeline

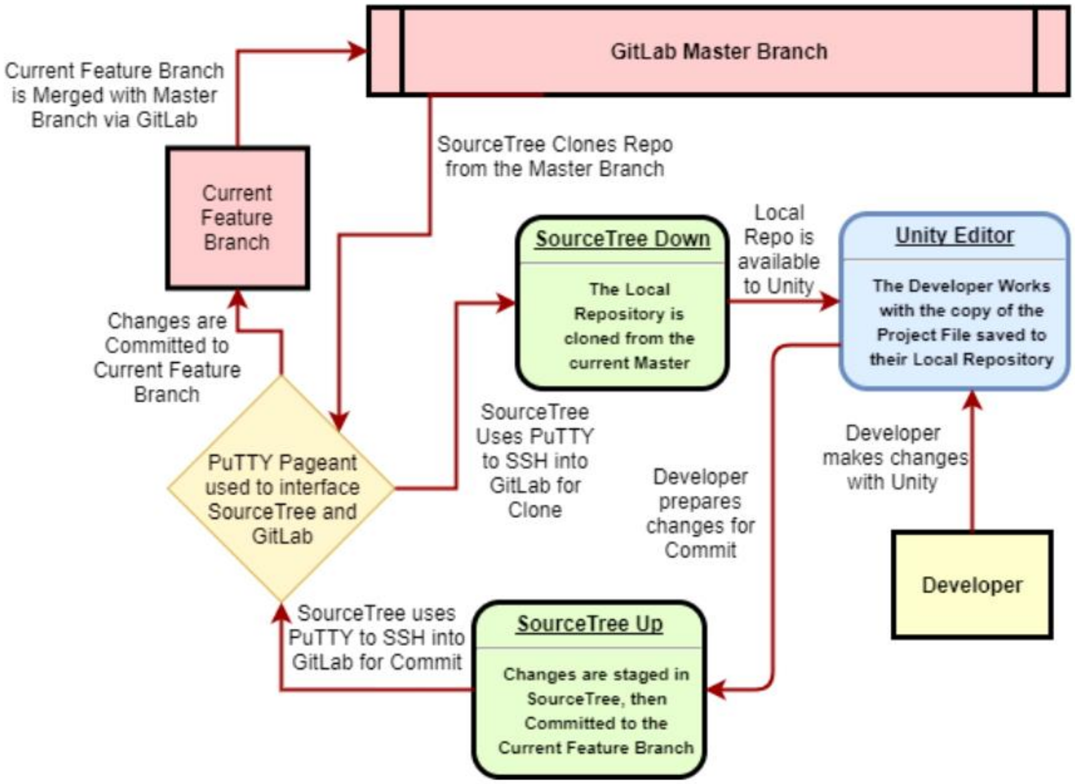
5.1 Core Development Software

The development process for Polymorpher centers around three major software toolsets. The first of these major toolsets is the Unity Software Development Kit. Unity is a popular Game Engine that comes with a suite of extremely powerful game development tools. Among these tools is the Engine's well-designed user interface which cleanly organizes, simplifies, and streamlines substantial portions of the game development process. These include animation rigging systems, game UI toolkits, component management systems, and a scene manager system with integrated prefab functionality.

The Unity SDK also comes packaged with an excellent IDE called Monodevelop. Monodevelop's core integration into Unity makes for a convenient and fast method of editing code and managing backend functionality for many important game components. On top of that it is primarily intended for C# development, one of the most powerful languages supported by the Unity SDK and the language Polymorpher is being written in.

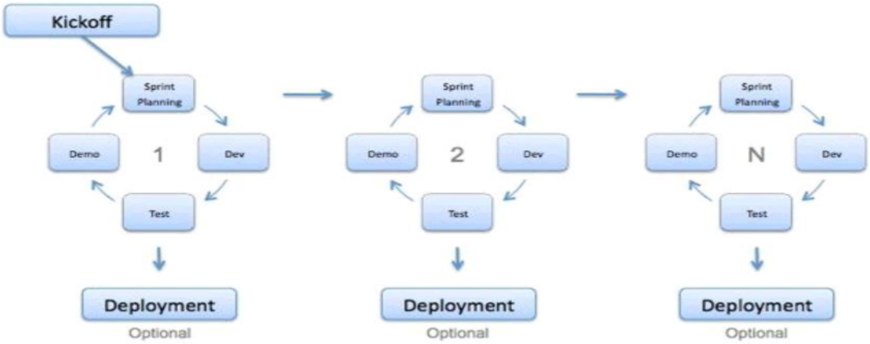
The second major toolset integral to Polymorpher's development is the SourceTree version control software. SourceTree is a file management software that interfaces each team members local repository, where they work with Unity to develop their portion of Polymorpher, with the team git repository. SourceTree manages this by handling SSH connections through the PuTTY SSH client software. From there it can either clone the master repository and update the developer's local repository with it, or push the developer's changes up to their respective Current Feature Branch on the git repository.

The final toolset involved in the development of Polymorpher is the git repository itself, maintained through the GitLab service. GitLab allows the team to manage all branches and keep tabs on all aspects of development between each member of the team, tracking commit history and update information from each developer's update logs.


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

5.2 Agile Development

The Agile Development process for Polymorpher involves the development of individual features. After a feature is first planned and developed, an implementation test is preformed to check its functionality. After this a working demo is produced to be approved, and after more test, refinement and approval it is sent for deployment. At this stage development focus is shifted towards the next feature and the process begins again. This process is iterated on until the products feature set is complete and the final build is approved for deployment.



5.3 Work Management

The development of Polymorpher has been divided among the team members in a variety of ways. First and foremost, the nine member team has been split in to three sub-teams, each with three members. Each sub-team has a team-leader who manages the development goals of that specific team week to week. Polymorpher's six core lesson sections have been distributed equally among the three groups, each group handling two lessons. The three team leaders also are tasked with the organization of asset request lists which are then handed over to the Team Silver Project Manager Matthew to place as an official request to procure the assets from any given outlet. This method of work division and development management will help to hasten the development of the product and ensure efficiency in the development pipeline.

6 Glossary

Assembly/Assembler: A process/program which converts assembly language to an object file or Machine Language format.

Code Compiler Directory: File directory holding the portable compiler, and the associated companion files, which are used to manipulate the scripts in the Streaming Assets Directory.

Coding Interface: An in-game GUI accessible to the player that pulls specified scripts from the Streaming Assets Directory for them to edit and compile using the portable compiler from the Code Compiler Directory.

Compilation/Compiler: A process/program which translates source code from a given programming language to Machine Language in order to be executed.

CS: Short for Computer Science, often in relation to the degree path at Old Dominion University but also in referral to the field of Computer Science as a whole.

Game Engine: A suite of software development tools with a User Interface geared towards streamlining the development process of applications, primarily video games.

IDE: Short for Integrated Development Environment, this kind of software is used to build and develop software in a variety of ways and often includes a suite of tools to assist the developer.

LoadScripts.cs: Manages files accessed by the entire portable compilation system by identifying which script is currently in focus as the "source" script for any selected object in game, pulling this file from the Streaming Assets Directory and passing it off to the Script Bundle Loader for compilation.

OOP: Object Oriented Programming

Playtester: An individual tasked with playing through an incomplete series of builds for video game software to assist in the refinement of said video game.

Polymorpher: The programming themed puzzle platformer being developed by Team Silver.

Prefab: A pre-fabricated component of some kind, usually used to combine multiple individual components of a segment of software into a single functional unit.

Replayability: The capacity of a game to be played more than once or for an expended period beyond initial completion.

ScriptBundleLoader.cs: Takes scripts passed off from the Load Script file and marks them for compilation, setting up the Assembler and Compiler and running the selected script through them. In the event of any compilation errors it will send out an error report through the Unity Error and Log files, otherwise it will attach the script to whichever game object was selected.

SSH: Stands for Secure Shell, a cryptographic network protocol for operating network services securely over an unsecured network.

Streaming Assets Directory: File directory where all scripts accessible to the player via the in-game Coding Interface are stored and organized according to level and programming concept relevance. It is unique in that it is one of the few source file directories that are accessible to the player in the Unity Engine under any condition.

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-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

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