Lab 1 - PolyMorpher Product Description

Matthew Tuckson

Old Dominion University

CS411W

Professor Thomas Kennedy

29 January 2018

Version 1

Table of Contents

List of Figures

List of Tables

1 Introduction

CS150 is an introductory course to computer programming at ODU that over the past four years over 2,500 students have taken. Although over 2,500 students start their college computer science education in this course, only approximately 600 students over the past four years have taken CS350, a more advanced computer programming course. This information lead Team Silver to determine why there is a drastic change in students that take these two courses. The conclusion found is that the more object-oriented programming (OOP) skills were required by a class, the less students took said class. Under the guidance of Professor Thomas Kennedy, Team Silver decided a potential solution to this problem is to create a game, named PolyMorpher, that would teach OOP to students.

1.1 Team Members

Team Silver consists of computer science students from ODU. Their names and roles on the team are defined 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

The problem statement for Team Silver is, “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” (Team Silver). Object Oriented Programming is an essential skill for students to learn for them to do well in advanced level CS courses. However, at entry level courses students do not get taught OOP concepts which can lead not the result of changing majors or leaving college entirely.

Figure 1 expresses the current process a student will go through upon entering an entry level CS course. They begin by entering the course, and after some time will be taught the fundamentals of programming. If they understand them, they will pass the class. If they do not

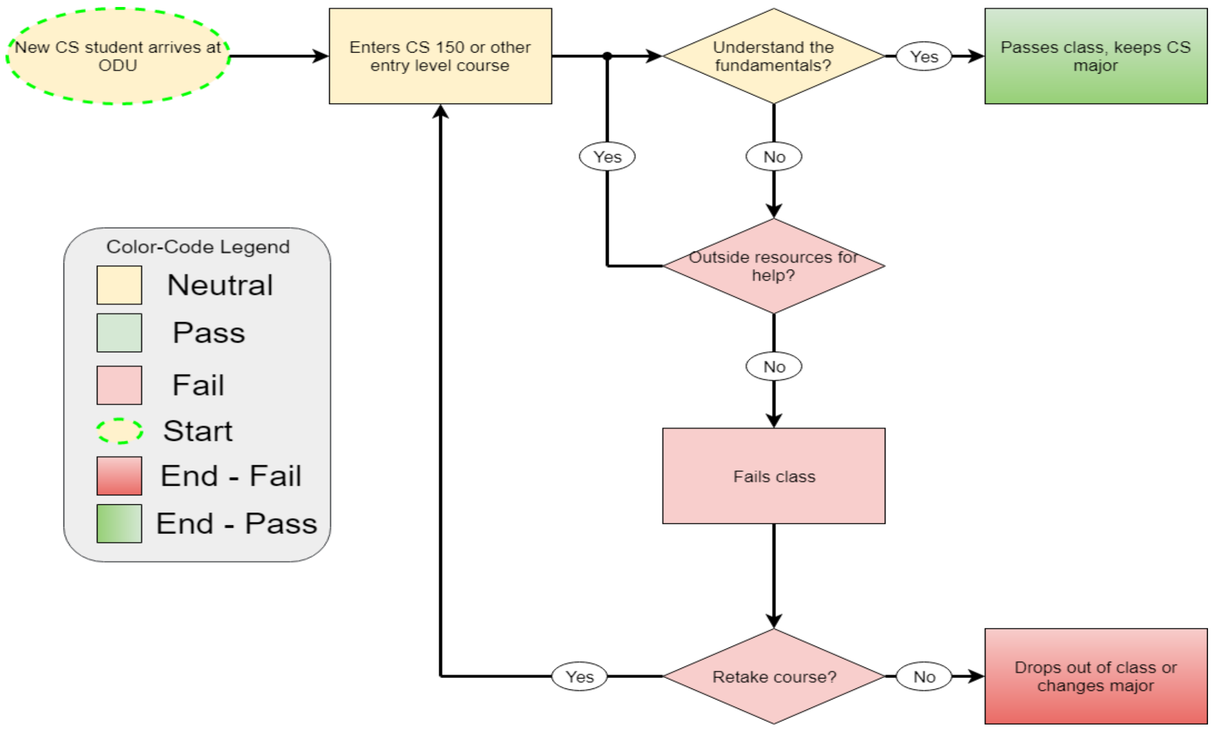

New CS student arrives at ODU
Enters CS 150 or other entry level course
Understand the fundamentals?
Yes
Passes class, keeps CS major
Yes
No
Outside resources for help?
No
Fails class
Retake course?
Yes
No
Drops out of class or changes major
Color-Code Legend
Neutral
Pass
Fail
Start
End - Fail
End -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

Figure 1. Current Process Flow.

they may seek for help, but if that is not enough they will eventually fail the course. They may retake the course, but unless they reach the point of understanding the fundamentals of programming they will eventually drop out of the class and potentially change majors. This is an undesirable result, and Team Silver believes this can be solved through proper understanding of Object Oriented Programming.

Team Silver's solution to this problem is PolyMorpher, a game in which the user will experience puzzles that they must solve using real code. The puzzles will be aimed to teach the user various concepts of OOP such as abstraction, inheritance, polymorphism, and encapsulation.

Figure 2 depicts the process a typical CS student would go through after PolyMorpher has been completed. The student would begin in an entry level course, however when faced with problems understanding the fundamentals they will now use PolyMorpher to gain understanding of OOP concepts. The foundational knowledge of OOP concepts would help the student then pass their class and continue in their CS major.

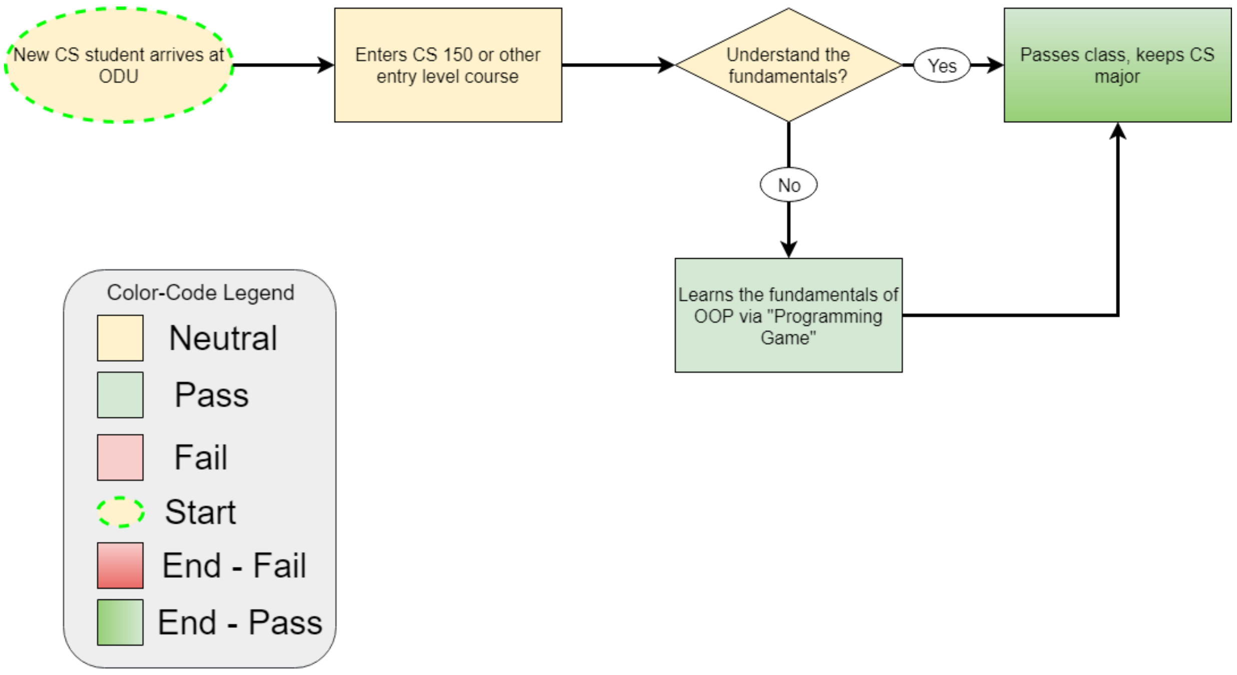
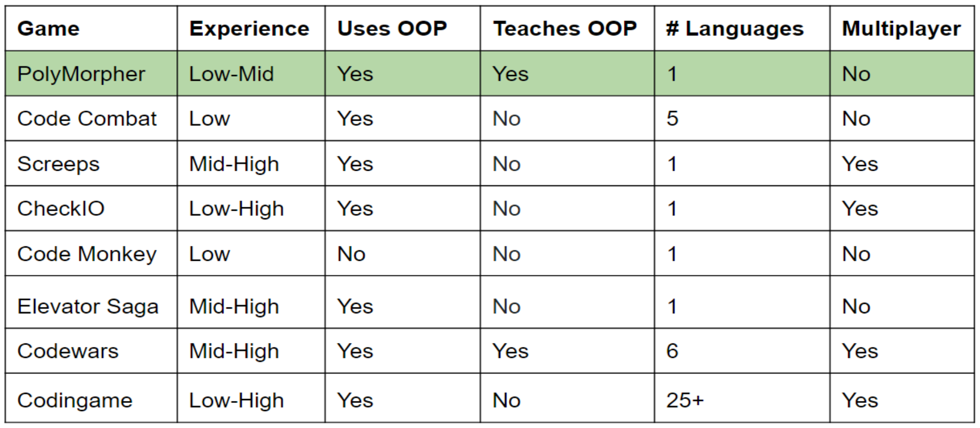

New CS student arrives at ODU
Enters CS 150 or other entry level course
Understand the fundamentals?
Yes
Passes class, keeps CS major
No
Learns the fundamentals of OOP via "Programming Game"
Color-Code Legend
Neutral
Pass
Fail
Start
End - Fail
End - Pass

Figure 2. Solution Process Flow.

2 PolyMorpher Product Description

PolyMorpher will be a game where the player learns and then applies OOP skills to win. They will be faced with challenges in the form of puzzles in which the only way to solve the puzzles is to code their way out. This focus on OOP is what will separate PolyMorpher from other games that currently exist that teach programming skills.

Figure 3 shows the competition matrix for PolyMorpher. As shown, although many different games use OOP in their games, very few teach the concepts behind it. The only one that does is Codewars, which teaches how to program in an object-oriented style, without teaching the underlining principles behind it. PolyMorpher will teach the basic principles of OOP such as abstraction, inheritance, encapsulation, and polymorphism through the various puzzles the player will encounter.



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

Figure 3. Competition Matrix.

2.1 Goals and Objectives

The goals for PolyMorpher are:

- Teach a basic understanding of abstraction and then have the user show the learned skill through application

- Teach a basic understanding of encapsulation and then have the user show the learned skill through application
- Teach a basic understanding of inheritance and then have the user show the learned skill through application
- Teach a basic understanding of polymorphism and then have the user show the learned skill through application
- Help the user develop problem solving skills
- Create a fun gaming experience for the user along with their learning experience

2.2 Key Product Features and Capabilities

There are two distinct aspects to the features and capabilities of PolyMorpher. The first lies in the software of the game itself, and the second lies in the minimal hardware required for the user to be able to play the game.

The story line in PolyMorpher is that the user has found themselves in a dungeon and need to escape. However, they have no fighting skills and are incapable of partaking in combat. In order to escape the dungeon, they need to manipulate their environment by hacking various game objects. A simple example of this would be that there is a block in the way of the user and the exit of a room. In order to get to the door, the user must "hack" the block, attach a script to the block telling it to move out of the way, and then go through the door. It is through this process of "hacking" the world that the user will solve various puzzles which will require an understanding of object-oriented concepts.

PolyMorpher will be optimized for a 4th gen i3 Intel Processor. The minimal operating system needed for PolyMorpher will be Windows 7. These are the only two requirements needed as PolyMorpher will be a 2D game requiring minimal processing power or graphic processing power. Customers will be able to obtain PolyMorpher by going to Team Silver's website, http://www.cs.odu.edu/~411silver/, and downloading the .exe file under the downloads tab.

3 Identification of Case Study

In the first stages of deployment, PolyMorpher will have different end-users compared to the customers who will but PolyMorpher.

3.1 Customers

PolyMorpher will be advertised to those who would teach OOP skills to students. Our preliminary customer will be ODU, however later in the deployment stages other universities will become target customers as well. Individual teachers, such as high school teachers, will also be target customers as PolyMorpher would be a great tool to help them teach their students more about OOP. Students of programming who are unaffiliated with any schools will become target customers. In this case the customer is also the end-user.

3.2 End-User

The end-users of PolyMorpher will be various kinds of students. The type of student depends on the customer affiliated with the customer. In the beginning of deployment, ODU students will be the primary end-users, as ODU will be the primary customer. Later in deployment all university students will become potential end-users, as all Universities will become target customers. If individual teachers, such as high school teachers, buy PolyMorpher, then their students would also be end-users.

3.3 Why Students Want PolyMorpher

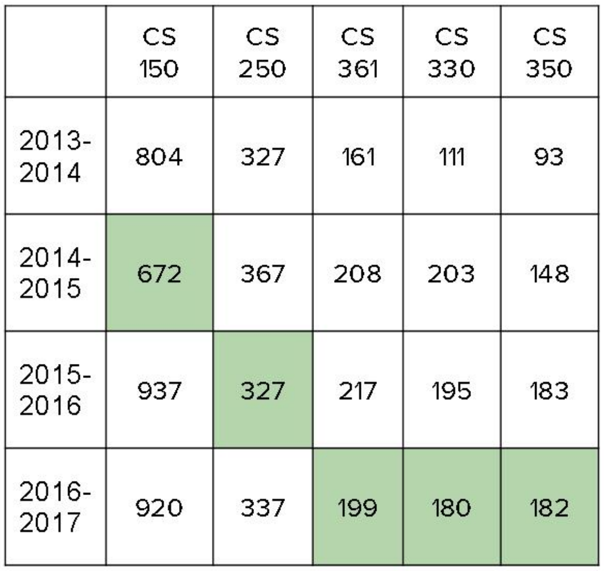
Students would want to use PolyMorpher to help them better understand OOP.

3.3.1 OOP is the Problem

Students pursuing a Bachelor of Science degree in computer science must successfully complete the following core courses:

- CS150: Required to be taken by computer science, physics, math, engineering, and mod-simulation majors
- CS250: Required to be taken by computer science, mod-simulation, computer engineering, and electrical engineering majors
- CS330: Required to be taken by computer science and mod-simulation majors
- CS361: Required to be taken by computer science, computer engineering, and electrical engineering majors
- CS 350: Required to be taken by computer science and computer engineering majors

Figure 4 contains the number of students who enrolled in these five courses over the past four years. It is important to note that the large drop in enrollment from CS150 to CS250 is primarily due to the vast number of majors which require CS150 compared to CS250. However, there is still about one-third less students in the 300 level courses in 2016-2017 compared to CS250 in 2015-2016, suggesting that CS250 causes many students to stop pursuing a major in computer science. CS250 is where OOP skills are required to understand in order to succeed in the course. Team Silver, through the development and deployment of PolyMorpher, hopes to aid students in understanding OOP concepts and this is why students will want to use PolyMorpher in their studies.



	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

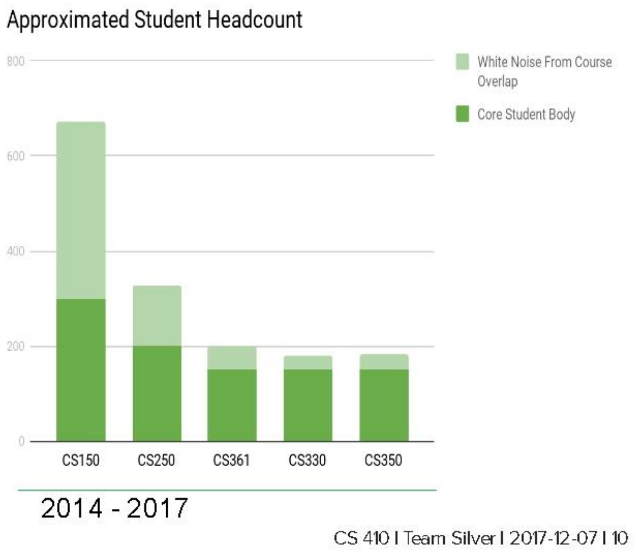


Figure 4. Student Enrollment Statistics.

4 Product Prototype Description

The PolyMorpher prototype will be released is April of 2018, and however it will not contain the full contents of the complete PolyMorpher product.

4.1 Prototype Architecture

One can obtain the PolyMorpher prototype from Team Silver's website, http://www.cs.odu.edu/~411silver/. The prototype will be playa ble through a .exe file. Figure 5 contains a diagram of the architecture of the completed PolyMorpher game. The prototype will not contain the dedicated web application, nor will it contain any databases.

4.2 Prototype Features and Capabilities

The PolyMorpher prototype will maintain the overarching structure of the completed version, however there will some eliminated features discussed in section 4.2.3 and 4.2.4.

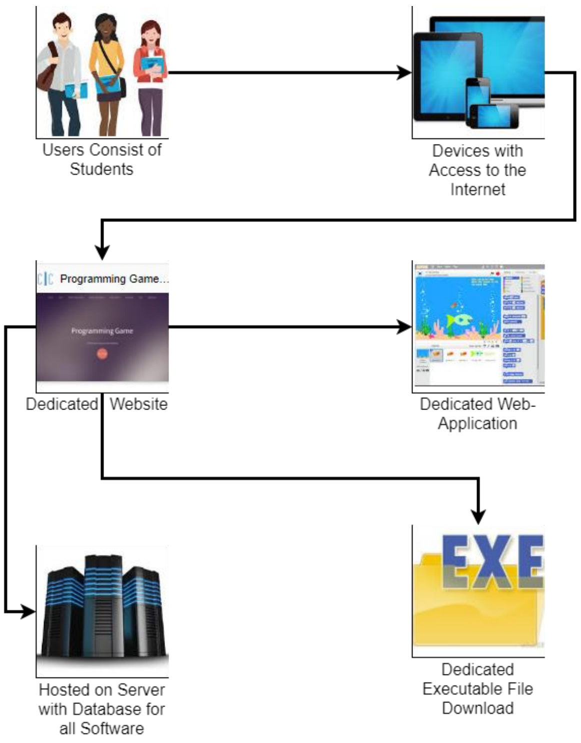

Users Consist of Students
Devices with Access to the Internet
Programming Game...
Dedicated Website
Dedicated Web-Application
EXE
Hosted on Server with Database for all Software
Dedicated Executable File Download

Figure 5. PolyMorpher Architecture.

4.2.1 Prototype Deliverable Features

The primary goal of PolyMorpher is to teach OOP, and the prototype will show how the complete version of PolyMorpher will do so. The PolyMorpher prototype will successfully teach a basic understanding of the following OOP principles:

- Abstraction
- Inheritance
- Encapsulation
- Polymorphism

These lessons will be taught through puzzles the player must complete to win the game. The player must obtain a basic understanding of the principles to be able solve the puzzles put in front of them. The player will be using the C# programming language, and therefore will gain hands on experience with C# and how to program using it.

4.2.2 Fully Functional Components

As PolyMorhper's primary goal is to teach OOP, the focus of Team Silver while developing the prototype will be to give a overall view on how PolyMorpher will do that. There were two approaches to the prototype that could have been taken. The first approach is to have only one principle of OOP taught, but have it completely taught through various puzzles making sure the player completely understands the principle. The second approach, and the one Team Silver decided to go with, is to have all four principles of OOP taught, but to have only one to two puzzles in each principle and therefore only teach an aspect or two of each principle. The primary reason Team Silver decided to go with the latter is to show how the overall structure of PolyMorpher, and to determine if these four principles could actually be taught through a puzzle solving game.

4.2.2.1 Game Assets

The PolyMorpher prototype will contain various game assets such as sprites, various sound effects, and textures.

4.2.2.2 Developed Story

The PolyMorpher prototype will contain a developed story with a background, progression, and completion. The story Team Silver has chosen is that the player is an old wizard who has been trapped in a dungeon. The only way they can get through the dungeon is by using their magic, sense they are too weak to fight in hand to hand combat. The way they will use this magic is by attaching scripts to specific game objects to make them act differently than originally intended.

4.2.2.3 Portable Compiler

Due to PolyMorpher's source code changing throughout the game, there will be a compiler inside the game able to read and compile the scripts the player makes.

4.2.2.4 Scenes

The PolyMorpher prototype will contain scenes that teach all four OOP principles, however only on a basic level, each principle containing the necessary scenes to go through one to two puzzles each. There will also be a scene in the beginning of each principle where the student will receive a more standard "textbook" understanding of the principle, as Team Silver has identified not all aspects of these principles can be taught through a game.

4.2.2.8 Tutorial section

Prior to entering the primary part of PolyMorpher, the player will go through a two-part tutorial. The tutorial will contain a section on the C# language, where basic syntax and any concepts in C# that are game development specific will be explained. It will be assumed that the player has used an OOP language before, as the purpose of this tutorial will primarily be to help the user understand any game development concepts that they may not have experience with. The second part of the tutorial will teach the mechanics of the game such as how to attach a script to a game object, the different tools in the coding environment, and how to successfully complete a level.

4.2.3 Time Permitted Functional Components

A final sandbox level is an intriguing aspect of the game that Team Silver would like to complete, but it is dependent on how efficient the rest of the prototype is developed. The sandbox level would contain most, if not all, of the game objects used to create PolyMorpher. Opposed to most scenes, where only some objects will allow the player to attach scripts to them, in the sandbox level all objects can have scripts attached to them. In this level the player will simply be able to do as they will with the objects, creating a learning environment driven by the player.

4.2.4 Eliminated Capabilities

There have been a few eliminated capabilities from the prototype version of PolyMorpher. Some will be included in the completed version of PolyMorpher, and some have been eliminated until future notice. The features are shown in Table 1 and Table 2.

4.2.4.1 Multiplayer

Originally Team Silver wanted to make PolyMorpher multiplayer, however through researching the security issues with a multiplayer game where players are able to edit the source code, it was decided to eliminate this capability until further notice.

4.2.4.2 Web Application

Similarly, Team Silver wanted to have PolyMorpher hosted on a web application, however with the current process for compiling code in the game, this is not possible. This feature has been eliminated until further notice.

A summary of what will and will not be included in the prototype of PolyMorpher compared to the finished product of PolyMorpher can be found in Table 2 which contains a summary of PolyMorpher's functions and capabilities.

KEY
🟩 Fully Functional
🟨 Partially Functional
🟥 Eliminated

Table 1. Key to Table 2

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)	🟩	🟩

Table 2. Summary of Prototype Functions and Capabilities

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	🟩	🟥

Table 2. Summary of Prototype Functions and Capabilities

4.2.5 Algorithms

There is truly only one algorithm needed for the development of PolyMorpher, which Team Silver has called the core algorithm as shown in Figure 6. However, due to its size it was decided to split the algorithm up into two parts, the API algorithm and the compiler algorithm.

4.2.5.1 Core Algorithm

The Core Algorithm consists of the process a play will go through from deciding to attach a script of an object to the completion of the script. The algorithm begins with the player selecting the "morph" icon. This icon will indicate that the player wants to attach a script to an object. They will then select an object that is "morphable" or that is predetermined to allow scripts to be attached to it. This will take them to the coding environment where they will have a text box containing Start() and Update(), two functions that are included is most scripts. They player can then do one of three things. They can either enter into the API algorithm shown in Figure 7, type directly into the textbox, or select the reset icon, resetting the textbox back to its original contents. Once completed editing the script they will click the compile icon and the script will be checked for errors. If errors are present, they will be sent back to their editable textbox. If there are no errors present, the script will be attached to the object and the player can continue adding scripts.

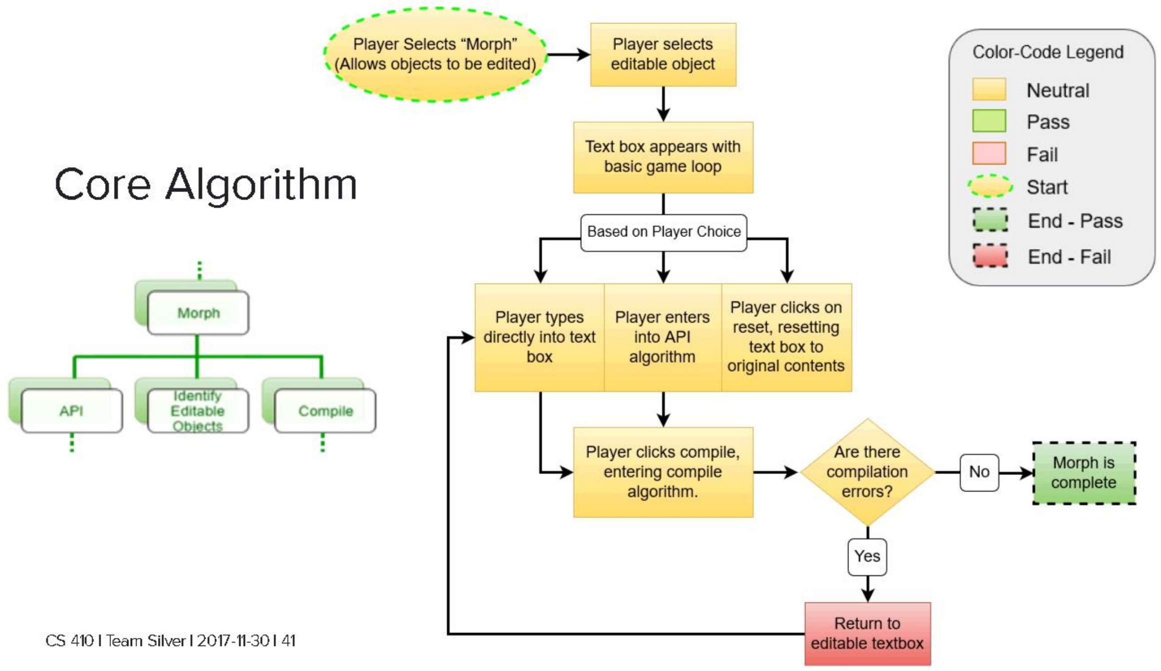

Core Algorithm
Morph
API
Identify Editable Objects
Compile
CS 410 I Team Silver I 2017-11-30 I 41
Player Selects "Morph" (Allows objects to be edited)
Player selects editable object
Text box appears with basic game loop
Based on Player Choice
Player types directly into text box
Player enters into API 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

Figure 6. Core Algorithm.

4.2.5.2 API Algorithm

The purpose of the API algorithm, shown in Figure 7, is to help the user code in a similar way the common APIs help software engineers code. While editing a script the player can click an API icon, which causes the API UI to appear. The player finds the function inside the API they wish to input into their code and selects it. The function will appear in the textbox and is now editable to fulfill the player's needs. This process will occur each time the player selects the API icon.

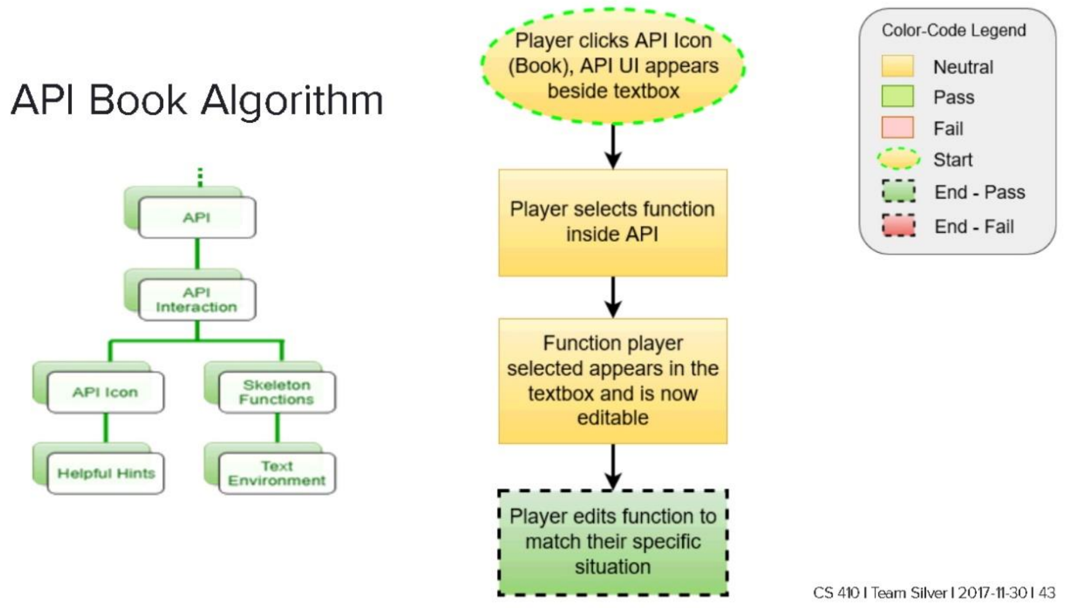

API Book Algorithm
API
API Interaction
API Icon
Skeleton Functions
Helpful Hints
Text Environment
Player clicks API Icon (Book), API UI appears beside textbox
Player selects function inside API
Function player selected appears in the textbox and is now editable
Player edits function to match their specific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

Figure 7. API Algorithm

4.2.5.3 Compiler Algorithm

The Compiler Algorithm, shown in Figure 8, explains what happens to the script the player writes after they click compile. It begins by being saved to the streaming assets directory, then at runtime an the object being edited is created with generic script called “LoadScript.cs” attached to it. LoadScript.cs will then mark the player’s edited script as its source code. LoadScripts.cs is then compiled through ScriptBundleLoader.cs which uses the CodeCompiler class. If there were errors in compilation, the errors will be printed to the player and they will retry. If there were no errors the script is attached to the object and the player continues playing the game.

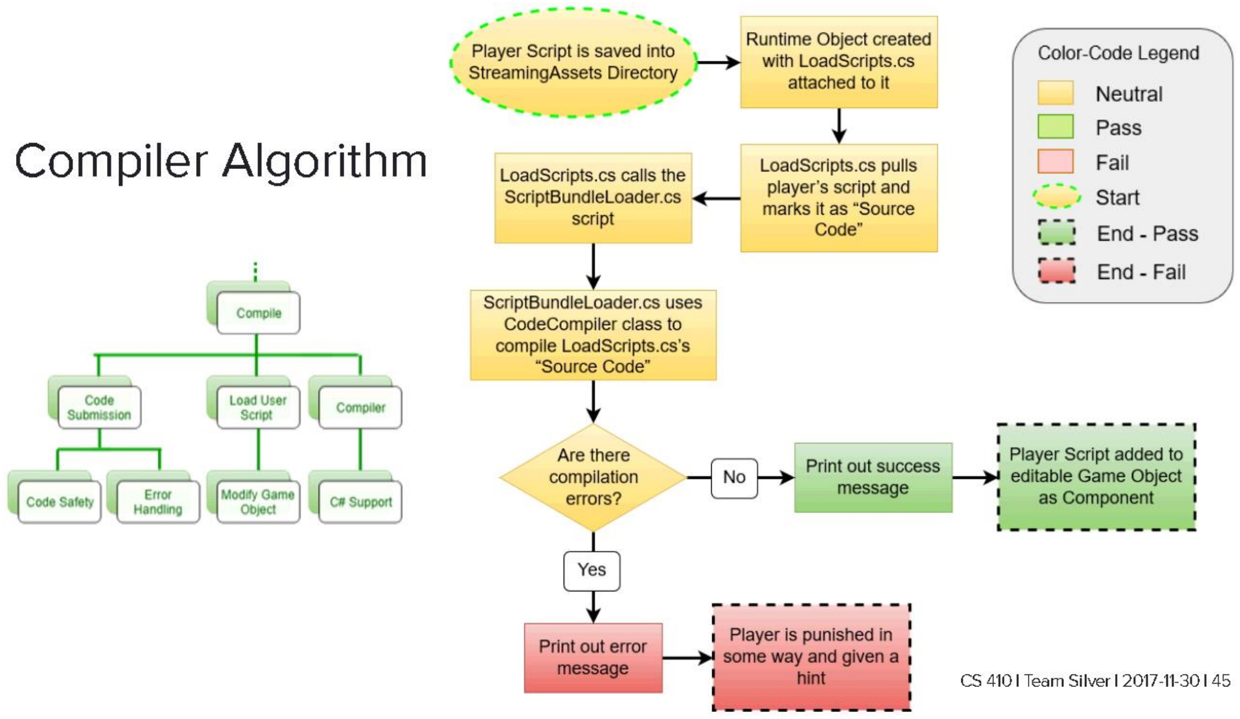

Compiler Algorithm
Compile
Code Submission
Load User Script
Compiler
Code Safety
Error Handling
Modify Game Object
C# Support
Player Script is saved into StreamingAssets Directory
Runtime Object created with LoadScripts.cs attached to it
LoadScripts.cs pulls player's script and marks it as "Source Code"
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 8. Compiler Algorithm.

4.3 Prototype Development Challenges

Through the designing of the PolyMorpher prototype, it became apparent that there would be plenty of development challenges that needed to be addressed.

4.3.1 Design Continuity

Keeping the design of a game consistent is extremely important when it comes to game design. Through communication and weekly meetings Team Silver plans to keep the overall design of PolyMorpher consistent.

4.3.2 Difficulty Debugging

Debugging PolyMorpher will have its own unique challenges, as the different codes the players can input into their scripts are countless. Team Silver plans to approach this primarily through playtesting, and having various users try to break the game and cause glitches. These glitches will then be debugged.

4.3.3 Playtesting

Playtesting is another challenge, as the PolyMorpher prototype will take approximately an hour to play for the average player. This can lead to long playtesting sessions, which leads to a lower number of times PolyMorpher gets playtested.

4.3.4 Maintaining Player Engagement

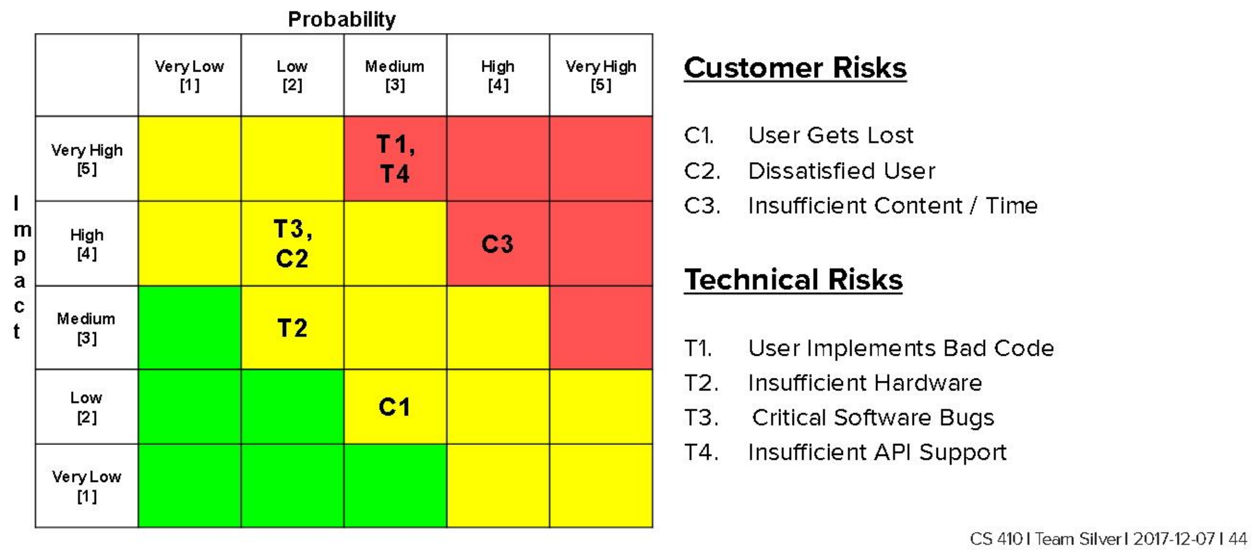
Maintaining player engagement will be a new challenge for most of Team Silver. This will be handled primarily through playtesting and receiving feedback from play testers of their enjoyment of the game.

4.3.5 Teaching Adequate material

Teaching adequate OOP material will not be a problem for the prototype, as it is in the plans to leave out portions of OOP concepts. However, when developing the complete game this will became a large problem, and will be handled through playtesting and receiving feedback to what the players learned in their test.

4.4 Risk Identification and Mitigation

Figure 9 contains Team Silver's risk matrix for PolyMorpher. There are both customer risks as well as technical risks that need to be addressed.



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 | Team Silver | 2017-12-07 | 44

Figure 9. Risk Matrix.

4.4.1 Technical risks

There are four technical risks that need to be addressed:

- T1: User Implements Bad or Harmful Code – The primary way this will be mitigated will be through sandboxing the game. This will allow scripts to only effect PolyMorpher. Secondarily, if they implement harmful code the player will be able to restart the level, resetting any changes they may have made.
- T2: Insufficient Hardware – This risk will be mitigated through developing the game in 2D, and therefore will be compatible for any modern PC or laptop.
- T3: Critical Software Bugs – Vigurous testing, playtesting, and debugging will all aid in mitigation this risk.
- T4: Insufficient API Support – The built in API will have all the necessary tools to help the player refer the specific code syntax and OOP concepts they need to learn to get through the levels of the game.

4.4.2 Consumer risks

There are three consumer risks that need to be addressed and they are as follows:

- C1: Player Gets Lost or Stuck – Helpful hints and tips will be included as well as a tutorial section to help the player not get stuck on any specific puzzle.
- C2: Dissatisfied User with UI – The UI/UX design will contain an approachable interface, clear menu options, and an easy to interact with interface for each level.
- C3: Insufficient Content/Time – The play may not learn enough or have enough time to learn the materiel for their CS course. This will be mitigated through vigorous playtesting.

5 Development Pipeline

The development pipeline, or the process in which Team Silver plans to develop PolyMorpher, contains game engine, IDE, version control, development model, and the work management of the team.

5.1 Unity Software

Unity is a powerful game engine used to create a multitude of games, and it will be used to develop PolyMorpher. This decision was decided due to members of Team Silver who have game development experience being most familiar with the Unity game engine.

5.1.1 Language: C#

The language Team Silver will use both to code as well as to teach the player will be C# due to the team's familiarity with C++ and Java, two very similar languages to C# yet not supported by Unity.

5.1.2 IDE: Monodevelop

The IDE Team Silver will use is Monodevelop. This is due to the simplicity of the IDE, the tools it contains for debugging, as well as the team's familiarity with the IDE.

5.2 Source Tree

Source tree is a free git client for windows or mac. The team will use SourceTree to interact with GitLab and manage their local repositories.

5.3 Version Control: GitLab

Team Silver's version control is summarized in Figure 10. The process begins with the developer updating their local repository from the GitLab master branch through the PuTTY pageant. This is completed by the developer through source tree, which will then allow the user to start editing the source code. When finished their work session, the developer will use source tree to update either a branch or the master branch of PolyMorpher.

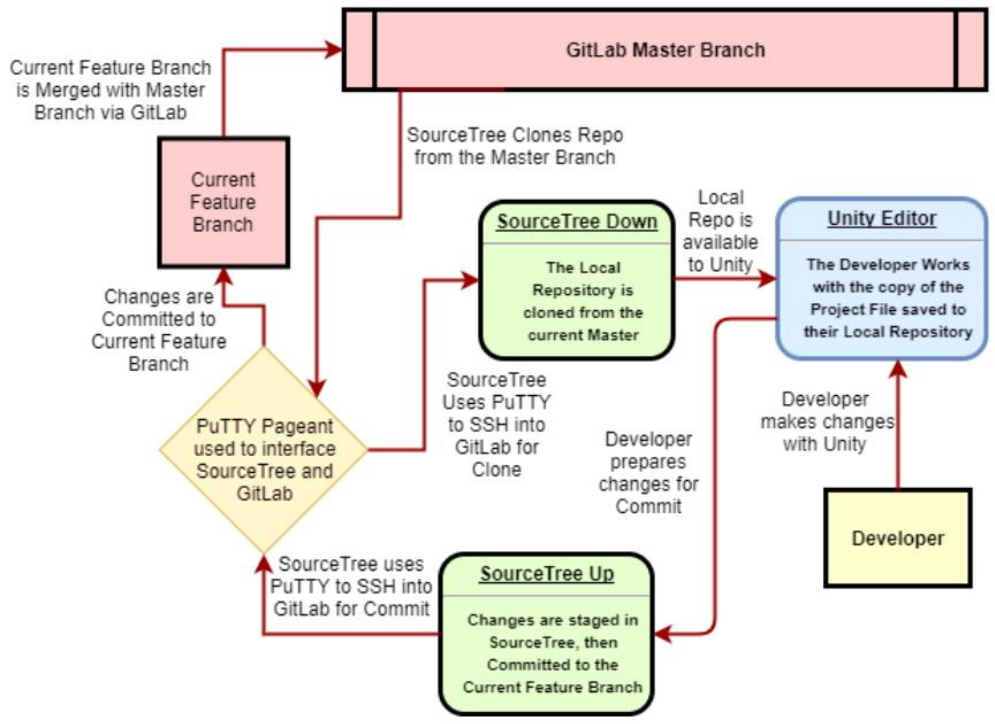

GitLab Master Branch
Current Feature Branch is Merged with Master Branch via GitLab
SourceTree Clones Repo from the Master Branch
Current Feature Branch
Changes are Committed to Current Feature Branch
SourceTree Down
The Local Repository is cloned from the current Master
Local Repo is available to Unity
Unity Editor
The Developer Works with the copy of the Project File saved to their Local Repository
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 10. Version Control

5.4 Agile Development

Team Silver has decided to use the agile development model, shown in Figure 11, to develop PolyMorpher. This is due to being able to see prototypes of features quickly, which allows other team members to develop with every other developer's work in mind.

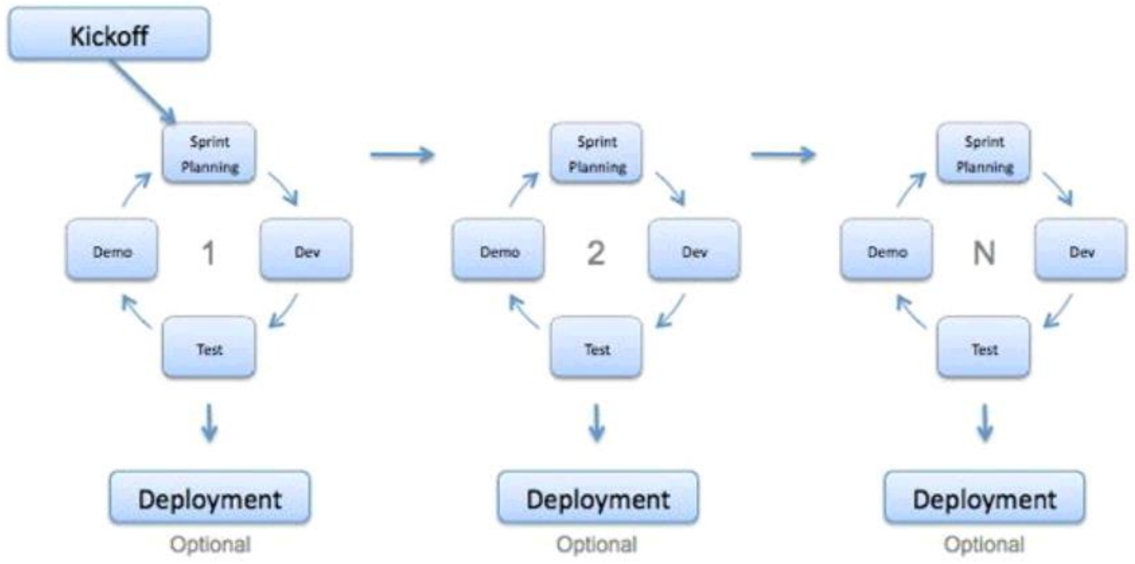


Figure 11. Agile Development

5.5 Work Management

Team Silver will begin development by making a prototype level, which will contain the common assets amongst the levels, the algorithms used throughout the game, as well as any common scripts that will be used throughout the game. Doing so will create a much more consistent theme throughout the various levels and create a more efficient development process.

Team Silver had decided to split into three sub groups to divide the work, each group having a team lead. The groups have been assigned two topics each and will work primarily amongst themselves to compete these assignments. The groups and assigned topics are as follows:

- Team A
    - Members: Matthew Tuckson (team lead), Colten Everitt, and Tyler Johnson
    - Topics: Abstraction and tutorial of game mechanics
- Team B
    - Members: Casey Batten (team lead), Daniel Dang, and Peter Riley
    - Topics: Inheritance and tutorial of C#
- Team C
    - Members: Joel Stokes (team lead), Kevin Santos, and Nathaniel Dearce
    - Topics: Polymorphism and encapsulation

Glossary

.exe: Executable file

API: Application Program Interface

CS: Computer Science

Game Object: any object in a game that the player can see and/or interact with.

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

IDE: Integrated Development Enviornment

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

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.

OOP: Object Oriented Programming

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.

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

“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%220B5KdQEdqLUPWnNoSHhIUGg2OTQ%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

Team Silver. “PolyMorpher.” December 2017. PolyMorpher Design Presentation.

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/