Lab 1 - Polymorpher

Tyler L. Johnson

CS 411

Professor Thomas J. Kennedy

27 January 2018

Version 1

Table of Contents

List of Figures

List of Tables

1. Introduction

Programming is typically viewed an intimidating skill to learn for those that are uninformed. The reality is that programming is a skill that can be learned, like any other skill such as drawing, or cooking. The main problem that arises is a lack of understanding of the fundamentals of programming. As a result of not knowing the fundamentals, many students find themselves lost, frustrated, and end up dropping their core CS class or switch majors. Current tools that exist teach some of the fundamentals but fail to provide an understanding of Object-Oriented programming concepts and problem-solving skills.

"Poor academics and knowledge decrement lead to stigma of video games being detrimental to the learning process. However, research evidence has shown that traditional learning through textbooks contributes to low engagement when compared to interactive media." The Office of Naval Research has found that 56 – 95% of people who play a game to learn a certain subject, on average, tend to display an improvement in understanding of the subject.

This problem provides the group an opportunity to fill the gap and provide students with an engaging and fun way to learn OOP concepts and problem-solving skills. The group under the direction of their mentor Professor Thomas Kennedy and managed by their project manager Matthew Tuckson are developing a solution to this growing problem. The team consists of 9 members: Cole Everitt, Casey Batten, Peter Riley, Kevin Santos, Joel Stokes, Matthew Tuckson, Nathaniel DeArce, Daniel Dang, and Tyler Johnson.

The group's solution is a game called PolyMorpher that will help players learn Object-Oriented Programing (OOP) concepts and program solving by using a management simulator and a Tangible User Interface (TUI). The game will have the following features: teach OOP concepts, teach problem solving, strive to teach multiple languages, be developed for multiple platforms, and potentially have multiplayer capability. As shown in Figure 1, the current process for new CS students being introduced to Computer Science is more complex and can more often lead to a student dropping their intended course or abandoning the major as whole. In Figure 2, it shows how this process can be improved by involving Polymorpher in the process flow and increasing the retention rate of students and the possibility of passing.

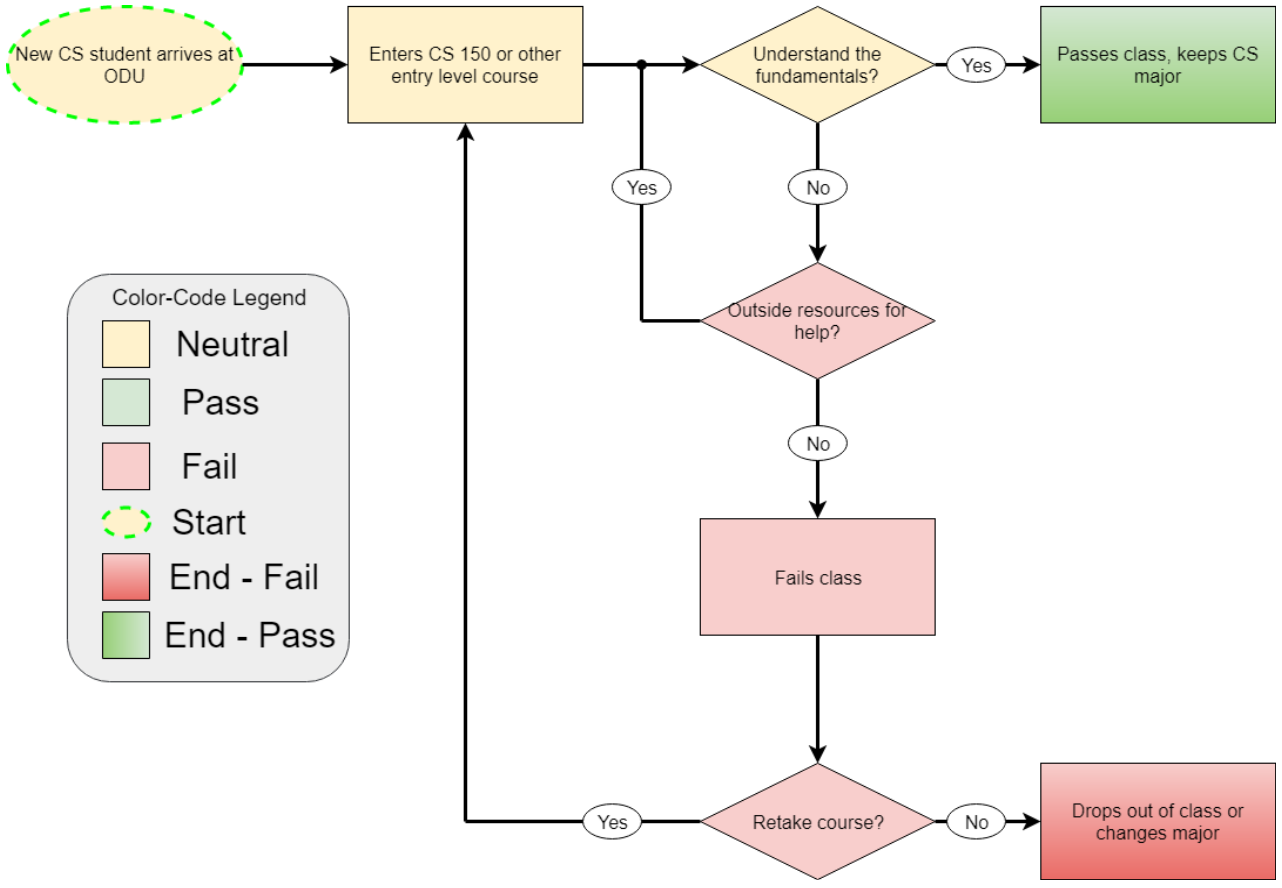

New CS student arrives at ODU
Enters CS 150 or other entry level course
Understand the fundamentals?
Yes
Passes class, keeps CS major
Yes
No
Outside resources for help?
No
Fails class
Retake course?
Yes
No
Drops out of class or changes major
Color-Code Legend
Neutral
Pass
Fail
Start
End - Fail
End -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

Figure 1 - ODU's Current Process Flow

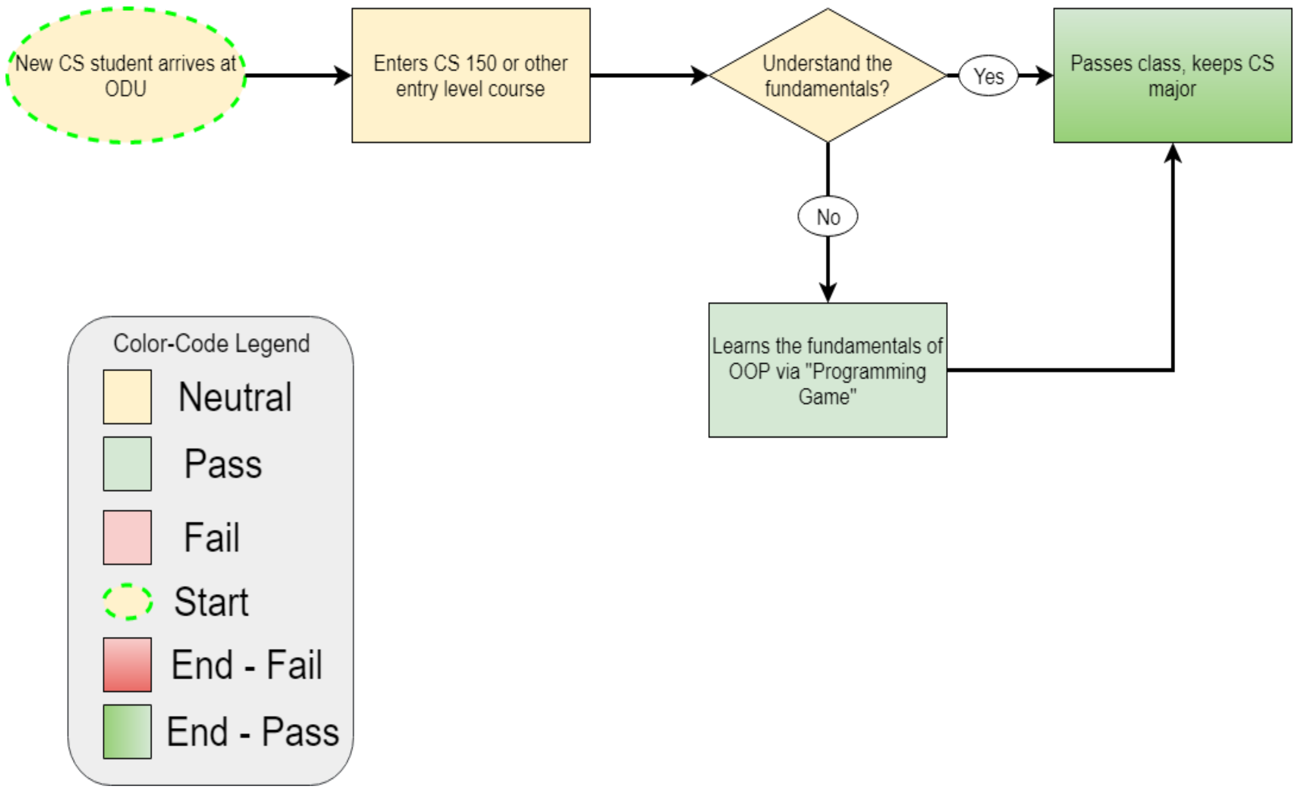
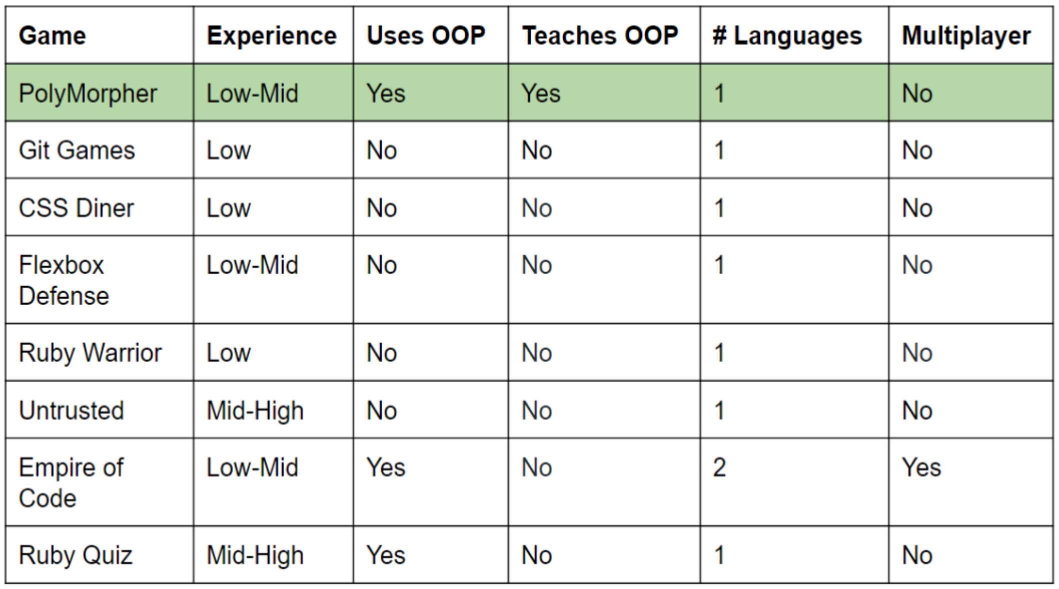

New CS student arrives at ODU
Enters CS 150 or other entry level course
Understand the fundamentals?
Yes
Passes class, keeps CS major
No
Learns the fundamentals of OOP via "Programming Game"
Color-Code Legend
Neutral
Pass
Fail
Start
End - Fail
End - Pass

Figure 2 - Solution Process Flow

There are currently many similar solutions to teaching new programmers how to program. Most of the competition has low bar for entry, they focus mainly one or two languages and teach mostly syntax. The group's solution also has low bar for entry, focuses on one language, but eventually featuring more, and focuses mostly on OOP concepts as its main priority. Very few programming games teach OOP which is major paradigm of modern programming. The fact that this is the main feature of the solution puts the group's solution above the rest.



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

Table 1 - Competition Matrix

2. PolyMorpher Product Description

Polymorpher is a game that will be initially aimed at ODU students and other educational institutes that teaches players computer programming, Object-Oriented Programming concepts, and problem-solving skills. The game's goal is to better prepare students for the rigors of their respective computer science program. This will be done by teaching them the fundamentals of programming in a fun and engaging way.

2.1 Key Product Features and Capabilities

The Polymorpher game will feature a main player character that the player of the game will control. The player will then face several challenges at each level that can only be solved by programming. The game will do this by featuring an in-game code editor which will allow the player to run and compile scripts. The player's script will then be checked for correctness, and if they pass they will move on to the next level.

2.2 Major Components (Hardware/Software)

Most modern computers will be able to run the game. The game will be an executable that will be available to download on the group's website. The system requires for the game will include: i5 Intel or AMD equivalent processor, 4GB of RAM or more and at least 500MB of available Hard Disk space.

3 Identification of Case Study

The solution is mainly aimed at ODU students who are entering ODU’s computer science program. The reason this is being made for students is because of the difficulty many students have with first learning program. As shown in Figure 3, the majority of students are in CS150 which is the first class you must take as a CS major as well several other majors. If the other majors are removed and the focus is solely upon CS majors as in Table 2, then the results are clear.

A majority of students are in CS150 and CS250 whereas much less students are in the upper level CS classes. It can be inferred that those students have either dropped the introductory courses or have switched majors. In those cases, Polymorpher could be used to help those students learn the material needed to pass those courses.

Polymorpher would not only be for ODU students but could be used as well by anyone who is interested in learning programming. It could also be used by professors or programming instructors to teach beginner students programming concepts such as OOP. Polymorpher could even be used by anyone who is just interested in learning programming. There are many courses available online, but most of them have the students going through repetitive exercises. This could leave eager to learn programmers bored and thinking they may have made a mistake. A game, which most people enjoy, could be used to alleviate that bored and make learning programming a fun process.

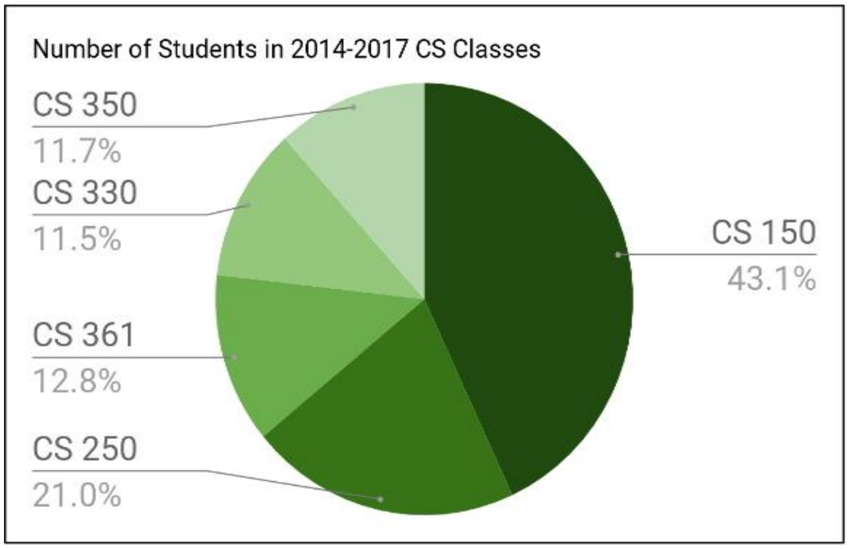
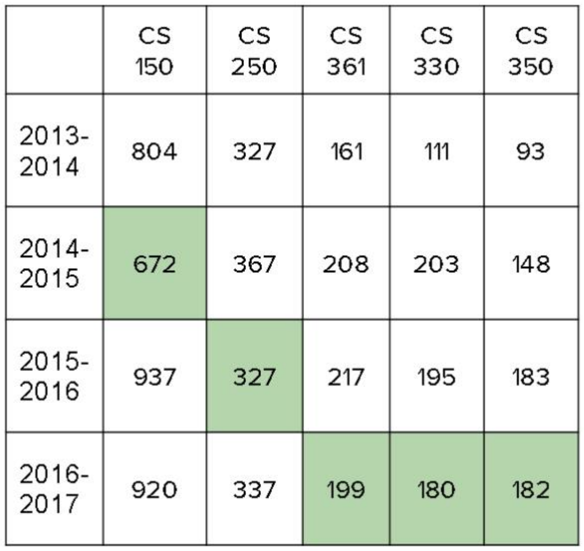


Figure 3 - CS Course Demographics



	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

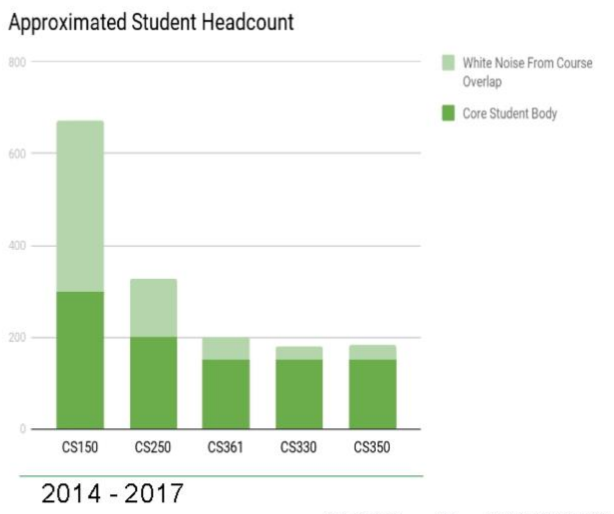


Table 2 - Student Progression Dilemma

4. Product Prototype Description

The goal of the Polymorpher prototype is to provide a baseline for the product. It will be minimalistic but will contain all the absolutely needed features. It will be playable but may not be as engaging as the full-fledge game.

4.1 Prototype Architecture (Hardware/Software)

The prototype will contain a .exe file that the end users will be able to launch. This will be all that is needed to play the game.

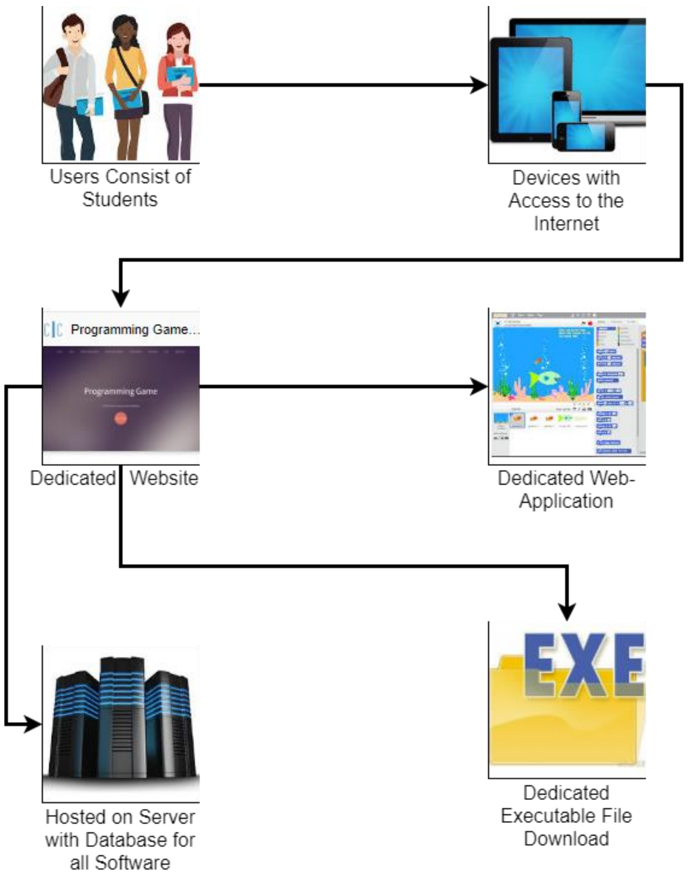

Users Consist of Students
Devices with Access to the Internet
Programming Game...
Programming Game
Dedicated Website
Dedicated Web-Application
EXE
Hosted on Server with Database for all Software
Dedicated Executable File Download

Table 3 - Deployment Flow

4.2 Prototype Features and Capabilities

The prototype will teach players the key concepts of OOP, it will only feature a single programming language and will be single player. The real-world version of the product would contain the features of the prototype but would include multiplayer, multiple languages and be deployed also as a web application. The differences between the prototype and the real-world version of the product can be found in Table 4.

4.2.1 Prototype Deliverable Features

The prototype will teach polymorphism, abstraction, encapsulation and inheritance that will be driven by a developed story. As the players engage in the story of the game, they will face varying OOP based challenges. These challenges will be contained throughout the various levels of the game. The player must demonstrate a measured level of competency for each concept in order to move on. The main goal is to keep the players engaged while teaching them OOP and problem-solving skills.

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	🟩	🟥

Table 4 - Prototype vs. Real-world Features

KEY
🟩 Fully Functional
🟨 Partially Functional
🟥 Eliminated

4.2.2 Fully Functional Components

A fully functional game will have many more assets than the prototype. Game assets will be essential in providing a good gaming experience. The full game will also include a fully developed story, more scenes teaching OOP, and a tutorial section.

4.2.3 Partially/Maybe Functional Components

A partially functional component would be a sandbox level. This level would allow the player to with much more freedom in regard to what they can do. The current levels will be much more restricting.

4.2.4 Eliminated Capabilities

Multiplayer and hosting the game on a web server has been eliminated from the prototype’s features. Multiplayer gameplay will be difficult from a security stand point since each player is given access to an editor that allows them to run C# code. As for the hosting the game on the web server, this would be difficult to maintain from a maintenance perspective.

4.2.5 Algorithms

There are three groups of algorithms that will be present in the prototype as well as the full product. These three groups of algorithms are essential to the game and are necessary for gameplay.

4.2.5.1 Core Algorithms

The core algorithm of the game is the main algorithm of the game's flow. It is the main entry point for all the other algorithms. The core algorithm features the initial GUI interaction the player will have during gameplay.

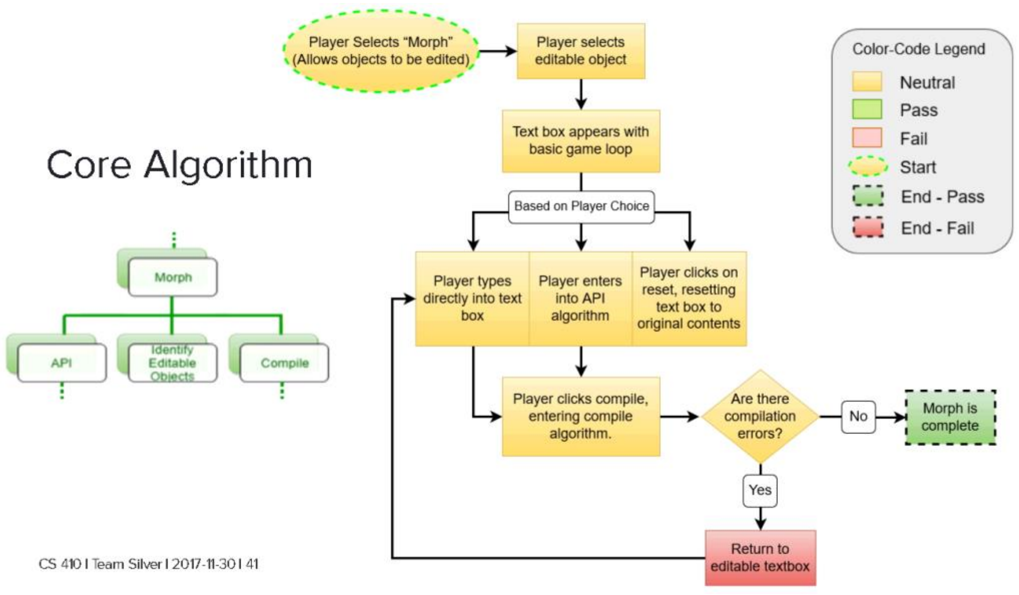

Core Algorithm
Morph
API
Identify Editable Objects
Compile
CS 410 I Team Silver I 2017-11-30 I 41
Player Selects "Morph" (Allows objects to be edited)
Player selects editable object
Text box appears with basic game loop
Based on Player Choice
Player types directly into text box
Player enters into API 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 4 - Core Algorithm Flow

4.2.5.2 API Algorithms

The API algorithm is the main way the user will solve the challenges that are presented to them. It will allow them to edit C# code and run it. The API algorithm is one of the most important algorithms of the game.

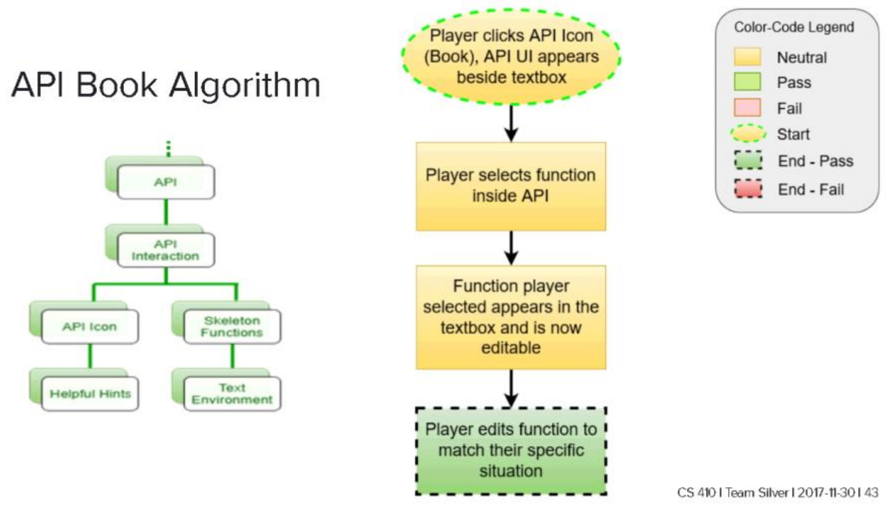

API Book Algorithm
API
API Interaction
API Icon
Skeleton Functions
Helpful Hints
Text Environment
Player clicks API Icon (Book), API UI appears beside textbox
Player selects function inside API
Function player selected appears in the textbox and is now editable
Player edits function to match their specific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

Figure 5 - API Book Algorithm Flow

4.2.5.3 Compiler Algorithms

The complier algorithm is the algorithm that is necessary for allowing the player to have their code run during runtime. This algorithm will allow the user to run the code that they have entered during the Book API.

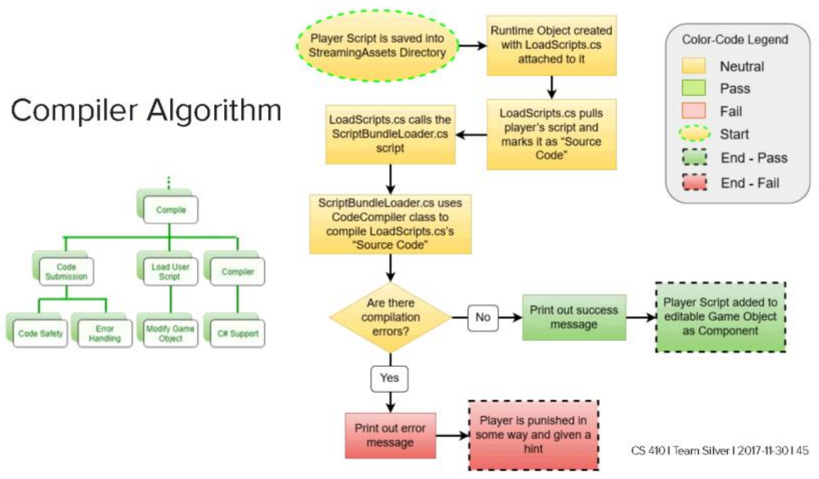

Compiler Algorithm
Compile
Code Submission
Load User Script
Compiler
Code Safety
Error Handling
Modify Game Object
C# Support
Player Script is saved into StreamingAssets Directory
Runtime Object created with LoadScripts.cs attached to it
LoadScripts.cs pulls player's script and marks it as "Source Code"
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 | Team Silver | 2017-11-30 | 45

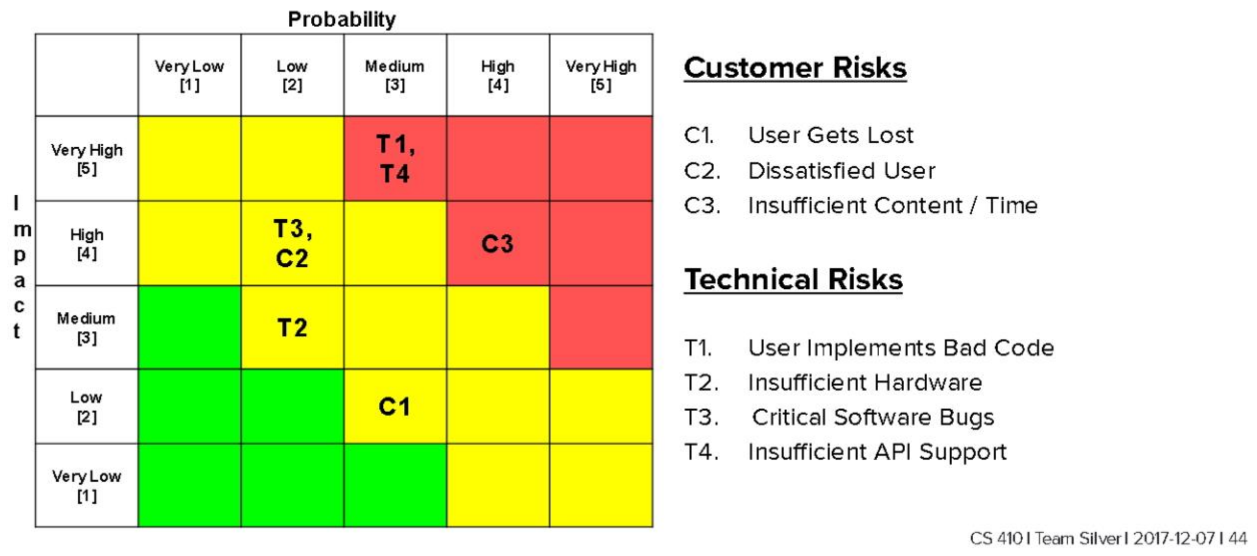
Figure 6 - Complier Algorithm Flow

4.3 Prototype Development Challenges

Some difficulties that will occur through the development process will be items such as: design continuity, issues with debugging code, playtesting, maintaining player engagement and whether the game has taught the play enough material. There will multiple levels of the game with groups designing their own level. This will present a challenge for group since the group must maintain a level of continuity in the design of game assets. Another issue is debugging, the main feature of the prototype will be allowing the user to run code during runtime. This presents a great challenge to the group as it is impossible to try every single possible piece of code a user may try. Another issue is playtesting, this issue is similar to the previous issue of debugging. As for the last two issues, this represent game design challenges. Each group that creates a level must ensure they are engaging the player and thoroughly teaching their required OOP concept.

4.4 Risk Mitigation/Risk Matrix

There are several risks that this product has that could impact the end product. The biggest risk is the user implementing bad code. As stated previously, it is impossible to test for every possible piece of code a user will implement. It is possible to test for the most likely scenarios which is what will need to be done. The other two higher risks would be not having enough material and insufficient API support. As long as each level contains enough material to teach the user the core concepts of its assigned OOP concept than this risk can be mitigated. The insufficient API ties into the risk of user implementing bad code. If the API does not support a majority of the user's actions than the user may leave the product. This can be mitigated by ensuring the API is capable of supporting the most likely user code implementations.



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

Table 5 - Risk Matrix

5. Development Pipeline

All of the development will be done in Unity. Unity its own built-in IDE called MonoDevelop which be used for all of the coding aspects of the prototype. The project will be put under version control and stored in a repository on GitLab. The software SourceTree will be used locally on each development machine to push and pull code to the repository in GitLab.

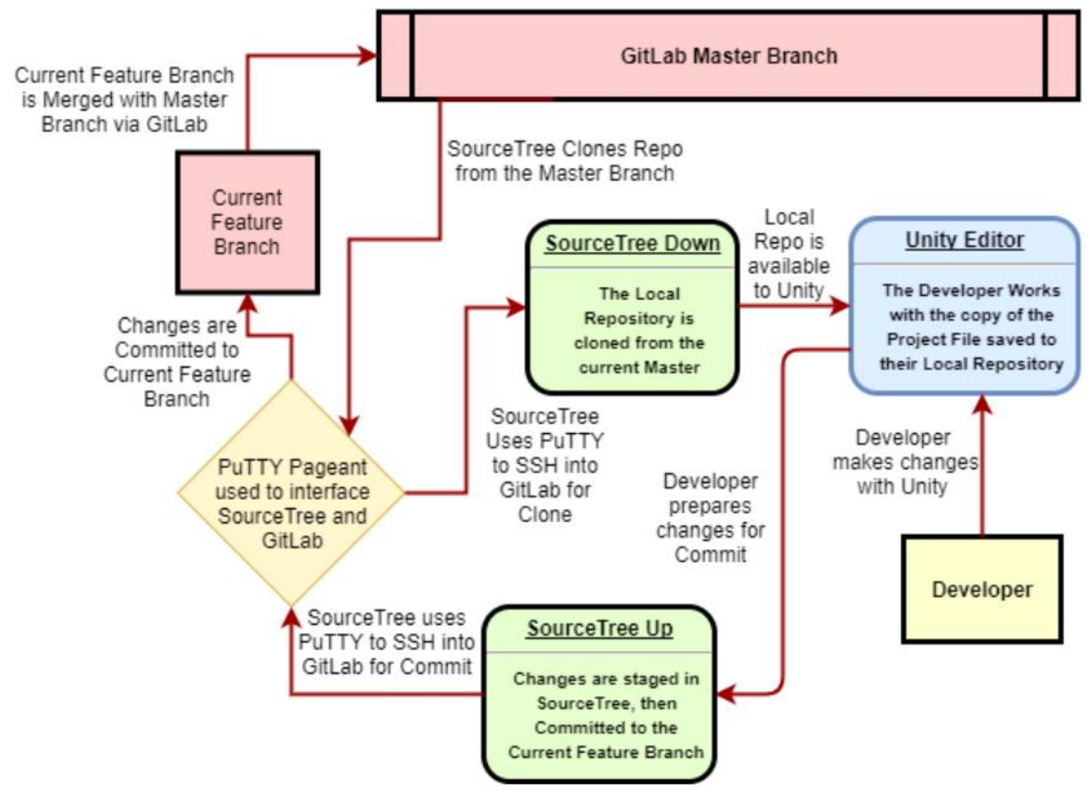

GitLab Master Branch
Current Feature Branch is Merged with Master Branch via GitLab
Current Feature Branch
SourceTree Clones Repo from the Master Branch
SourceTree Down
The Local Repository is cloned from the current Master
Local Repo is available to Unity
Unity Editor
The Developer Works with the copy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

Figure 7- Version Control Process Flow

5.4 Agile Development

The project will be run using an Agile Development process. This method of development is well designed for a game since a game is often an ever-evolving project. This will allow the group as a whole to remain flexible when making changes to the game.

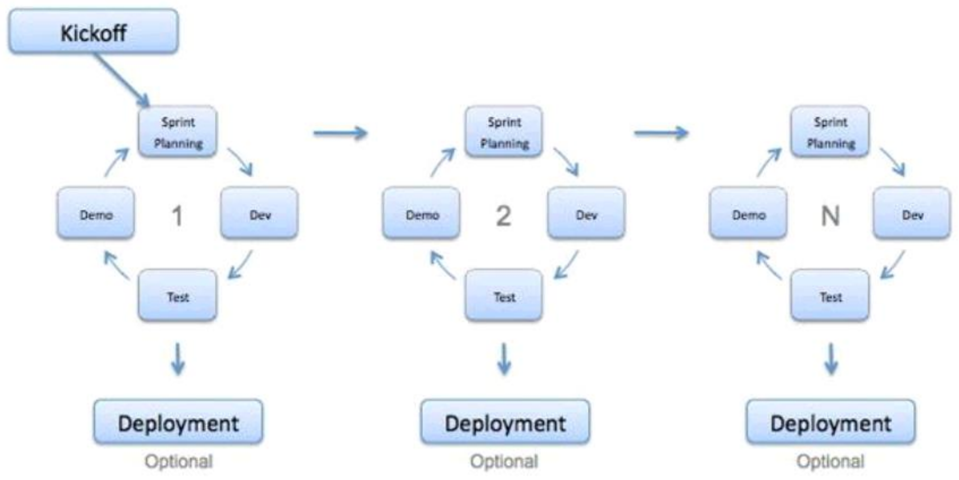


Figure 8 - Agile Development Flow

5.5 Work Management

The game will be divided into levels with each level teaching one kind of OOP concept. The entire group will then be divided into sub-groups that will design and develop each level. Each group will be responsible for teaching their assigned OOP topic. This will allow the group cover developing a wide range of levels at once.

6. Glossary

OOP (Object Oriented Programming): a programming paradigm that requires programmers to define data types of data structures and also functions of those data structures.

API: Application Programming Interface

GUI: Graphical User Interface

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.

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.

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.

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.

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.

7. References

Batten, C. (Narrator). (2017). CS410 Dungeon Escape Demo (Short Version) [Online video]. Online: YouTube. Retrieved from https://www.youtube.com/watch?v=ynhdd1IKgps

Batten, C. (Narrator). (2017). CS410 Project Dungeon Demo [Online video]. Online: YouTube. Retrieved from https://www.youtube.com/watch?v=ynhdd1IKgps

Batten, C. (2017, November 21). CS410 Tech Demo 2 (Download Source Code). In PolyMorpher. Retrieved from http://www.cs.odu.edu/~410silver/references.html

Batten, C. (2017, November 29). VersionControlFlow. In draw.io. Retrieved December 21, 2017, from https://www.draw.io/?state=%7B%22ids%22:%5B%221IQj6SYJqC6YLAK_qMRVIQkHiUmr9laBu%22%5D,%22action%22:%22open%22,%22userId%22:%22108692003133590583047%22%7D#G1IQj6SYJqC6YLAK_qMRVIQkHiUmr9laBu

Batten, C. (2017, October 26). CS410 Dungeon Escape Demo (Download Source Code). In PolyMorpher. Retrieved from http://www.cs.odu.edu/~410silver/references.html

Batten, C. (2017, October 26). CS410 Dungeon Escape Demo (Play Now). In PolyMorpher. Retrieved from http://www.cs.odu.edu/~410silver/references.html

“The Benefits of Video Games.” abcnews (2011, December 26). Retrieved October 19, 2017, from http://abcnews.go.com/blogs/technology/2011/12/the-benefits-of-video-games/

Good-Morning-America

Edraw. (2017, May 12). Standard Flowchart Symbols and Their Usage. In Edraw Visualization Solutions. Retrieved from https://www.edrawsoft.com/flowchart-symbols.php

Everitt, C. (2017, September 6). Current Process Flow. In draw.io. Retrieved December 21, 2017, from https://www.draw.io/?state=%7B%22ids%22:%5B%220B-5KdQEdqLUPd

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

Everitt, C., & Dang, D. (2017, September 24). currentProcessFlow. In draw.io. Retrieved December 21, 2017, from https://www.draw.io/?state=%7B%22ids%22:%5B%220B3Bc9

5zBWXg9TFZ6X0FMU1NTdEk%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