Design-PolyMorpher

CS 410 - Team Silver
Old Dominion University
December 7, 2017

Team Silver Mentor

Professor Thomas Kennedy

- Lecturer Professor, Old Dominion University
- M.S. in Computer Science, Old Dominion University
- Excels at:
  - Working directly with students
  - Outlining course information
  - Conveying an exceptional level of proficiency in the most current Computer Science practices

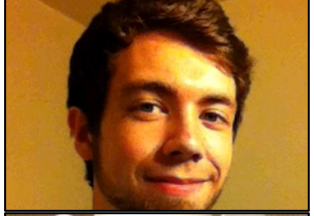
Team Silver

Colten Everitt
Website Administrator

Peter Riley
Software Engineering Expert

Casey Batten
Unity SDK Specialist

Kevin Santos
Database Manager



Joel Stokes
Game Development Specialist

Matthew Tuckson
Program/Project Manager

Daniel Dang
Web Development Expert

Nathaniel DeArce
Artificial Intelligence Agent

Tyler Johnson
Software Engineering Expert

Table of Contents

Why a Game?

- Enhance interest among new learners
- Nature of interaction inherently gives players a more natural way to learn content
- Change the learning style from traditional to more dynamic
- Does not require an instructor present at all times
- Object-Oriented Programming and problem solving can potentially be better grasped and understood
- See Appendix D for additional Unity SDK information

Image 1 source : http://www.cdm.depaul.edu/academics/Pages/BS-in-Game-Programming.asx
Image 2 source: Peter Riley's presentation

Influences of Games on Learning

Poor academics and knowledge decrement lead to the 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 (see Appendix D for additional Unity SDK information).

According to the **Office of Naval Research (ONR)**:

- Video games have positive aspects that help people become more engaged in the learning process
- On average, **56 - 95%** of people who play a particular game to learn a particular subject through tests, demonstrated a better understanding of such subject
- Educational games with a solid foundation and interactive components keep the players engaged and promote enhanced concept learning

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

Key Terms

Non-Technical User - User who lacks formal education/knowledge in computer science, computer programming, object-oriented programming, or problem solving

Non-Technical Game - User-Friendly gameplay able to be utilized by non-technical users

User-Friendly - Easy to comprehend by non-technical users

PolyMorpher

Our 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.

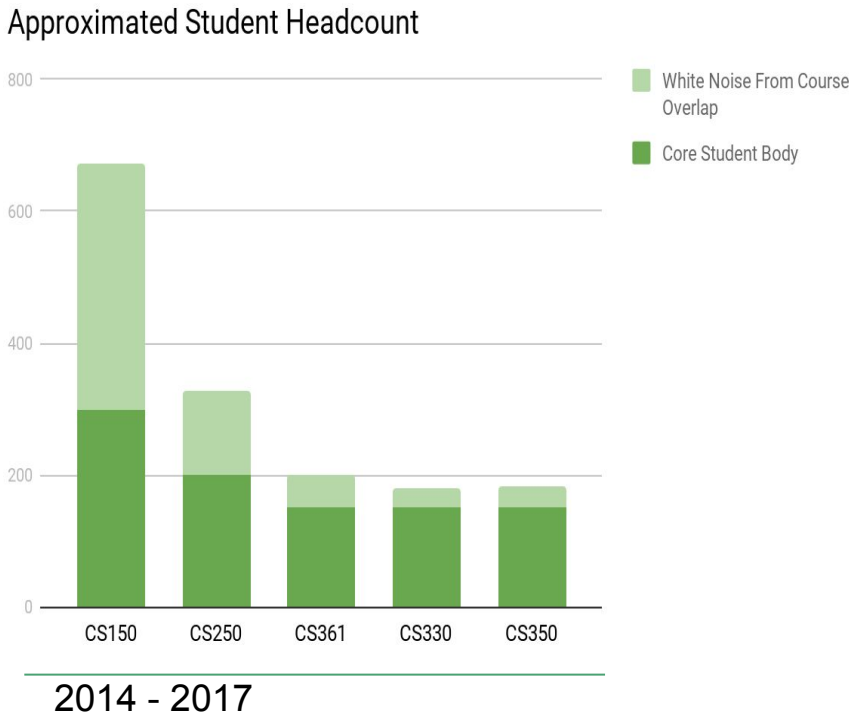
Student Progression Dilemma for CS Students at ODU

CS Courses as Requirements (See Appendix B for Additional Statistics)

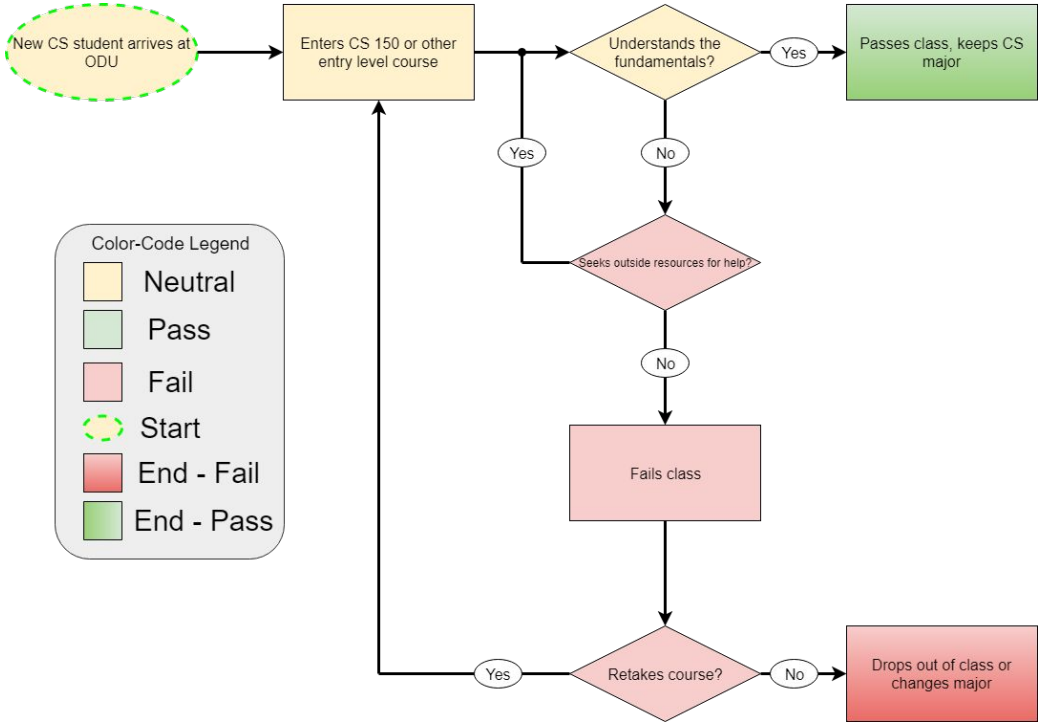
- CS150:
  - “Service Course”
  - Required to be taken by CS, Physics, Math, Engineering, & Mod-Simulation majors
- CS250:
  - Required to be taken by CS, Mod-Simulation, & Computer and Electrical Engineering majors
- CS330:
  - Required to be taken by CS & Mod-Simulation majors
- CS361:
  - Required to be taken by CS & Computer and Electrical Engineering majors
- CS350:
  - Required to be taken by CS & Computer Engineering majors

Student Progression Dilemma - Table & Graph

	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



Current Process Flow



PolyMorpher

Our Solution Statement

PolyMorpher will address Object-Oriented Programming (OOP) concepts and problem solving through the use of a management simulator and a Tangible User Interface (TUI).

Target Customers

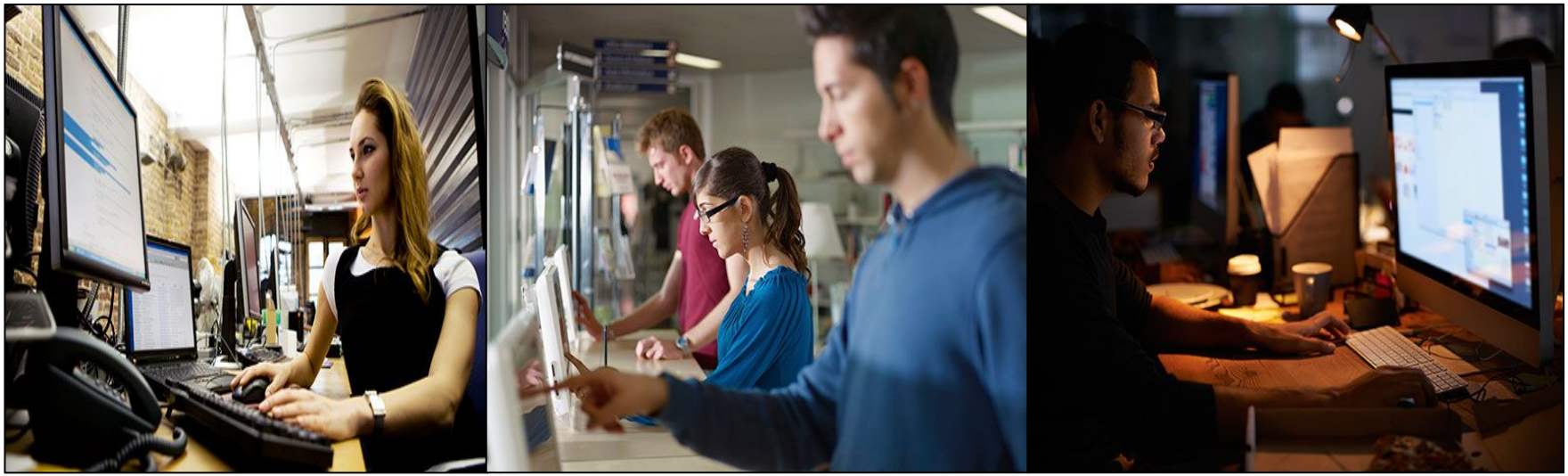
Initial focus will be Old Dominion University, as well as other universities, colleges, and educational institutions that currently offer a Computer Science degree program

Anyone could use this product in order to gain more knowledge in computer programming, Object-Oriented Programming concepts, and problem solving skills

Image Source : http://odu.edu/compsci

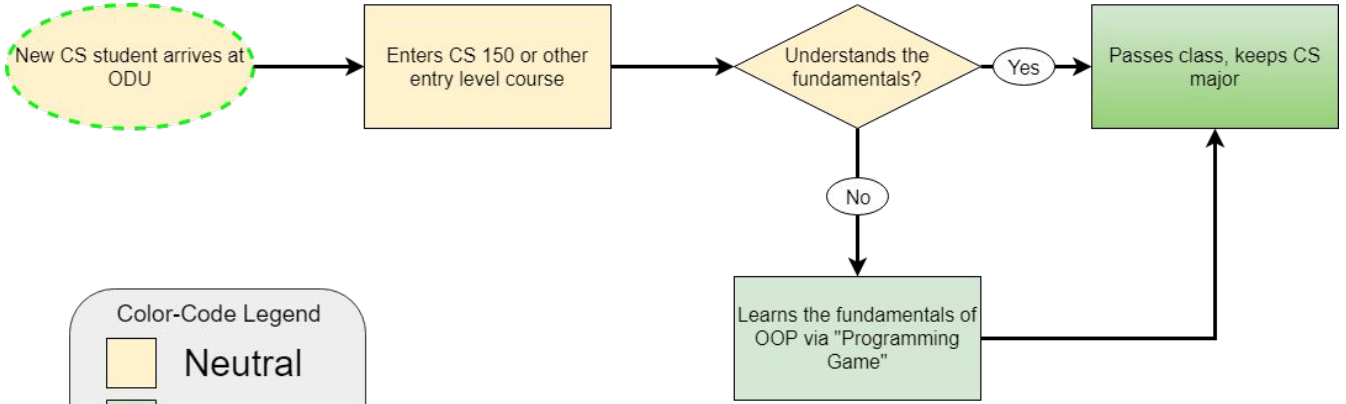
End Users

Students who are currently enrolled in a Computer Science degree program at Old Dominion University, or at other universities, colleges, or educational institutions



Left image source: https://online.odu.edu/programs/computer-science-ms
Middle image source: https://online.odu.edu/programs/computer-science
Right image: https://online.odu.edu/programs/computer-science-minor

Solution Process Flow



Plans for our Solution

Solution: Game application that teaches users the fundamentals of computer programming and software development

This application will:

- Teach Object-Oriented Programming (OOP) concepts
- Teach problem solving skills
- Strive to teach multiple languages
- Be developed for multiple platforms
- Be a single player game
- Be a standalone downloadable .exe application (no WiFi or Internet connection required)

Competition Matrix Part 1

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

Competition Matrix Part 2

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

Competition vs. Our Solution

Competition Trends:

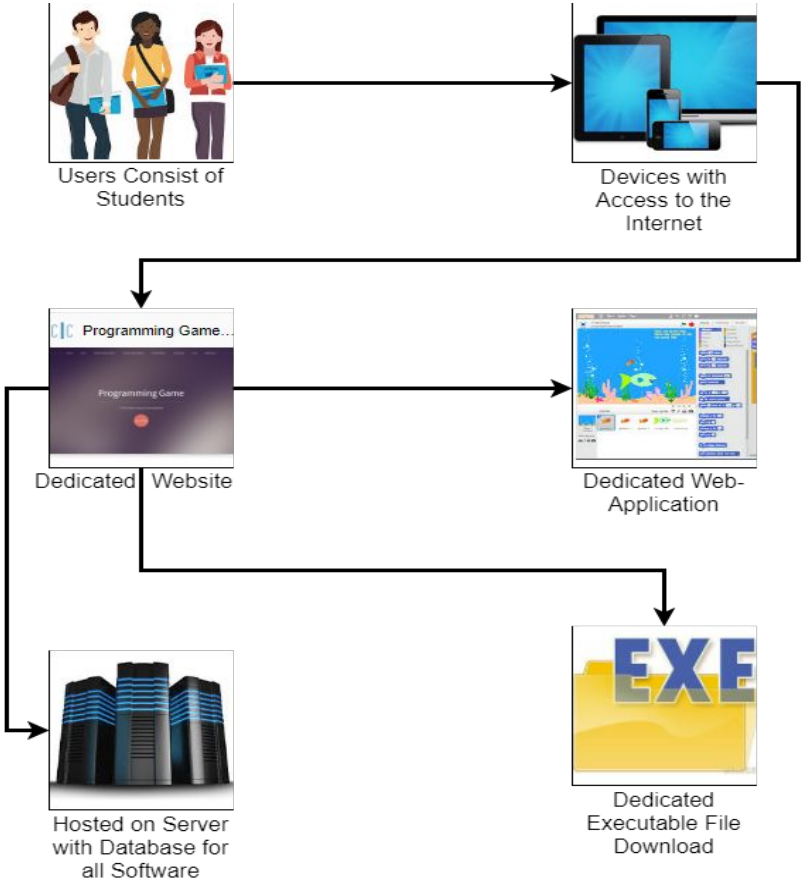
- Low programming experience
- Focus on one or two languages
- Mainly teaches syntax, rather than OOP
- Some include multiplayer

Our Solution:

- Low programming experience
- Start with one or two languages, but allow for teaching more in the future
- Focus on OOP concepts, with syntax being a secondary objective
- Multiplayer will depend on what gameplay features are implemented

Major Functional Components

- Users connect to the Internet using their preferred device
- Our game will teach users Object-Oriented Programming concepts as well as problem solving skills



Concepts of Gameplay and Possible Design Choices

- Realistic Approach:

  Using relatable and applicable examples

- Improvement on Teaching:

  More complex Object-Oriented Programming concepts that can be easily explained

- Balance Gameplay and Programming:

  Implementation of the gameplay will not sacrifice the player's experience

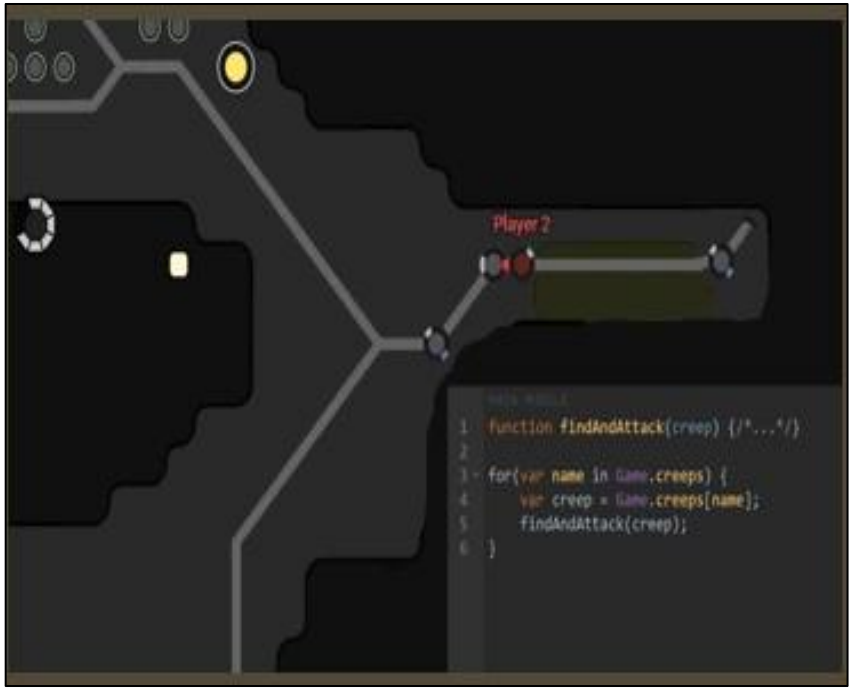


Image source 1: https://medium.com/@techloop.io/funniest-programming-memes-1da50c5229d

Structure of PolyMorpher

1. Level Structure
2. Objectives / Goals
3. Single Player
4. Sound Bar
5. Health Bar
6. Game Menu
7. Current Method Plan
8. Help Menu

Software Requirements

For Development:

1. C# Programming Language
2. Unity SDK
3. Third-Party Libraries & APIs
    a. Mono
    b. Microsoft C# CodeCompiler
4. Gitlab
5. SourceTree (Development Source Control)

For Production:

1. Unity SDK
2. Windows 7, 8, 8.1, 10
3. Linux (Any Version)
4. MacOS (Any Version)

Version Control Components

Unity:

- Game engine where source code is edited

SourceTree:

- Free git client for Windows and Mac
- Uses PuTTY

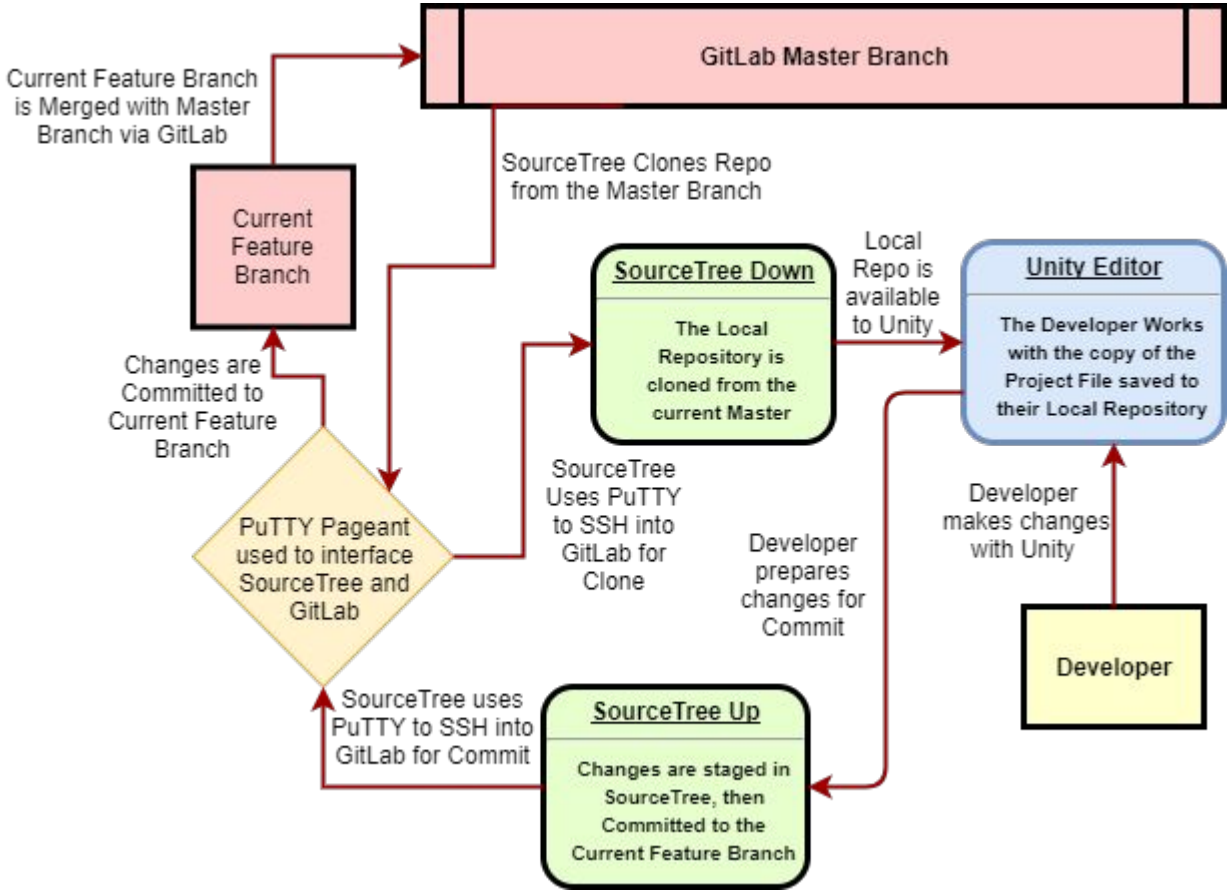
PuTTY:

- Pageant client interfaces with GitLab
- Continuously runs in the background

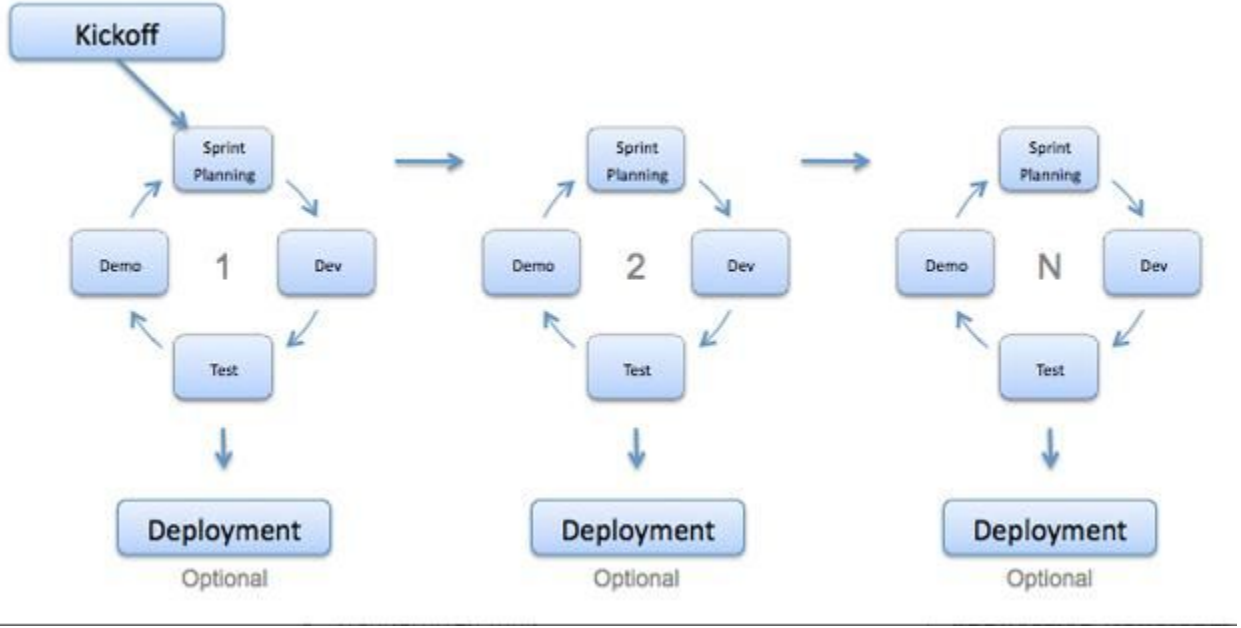
GitLab:

- Online git repository manager

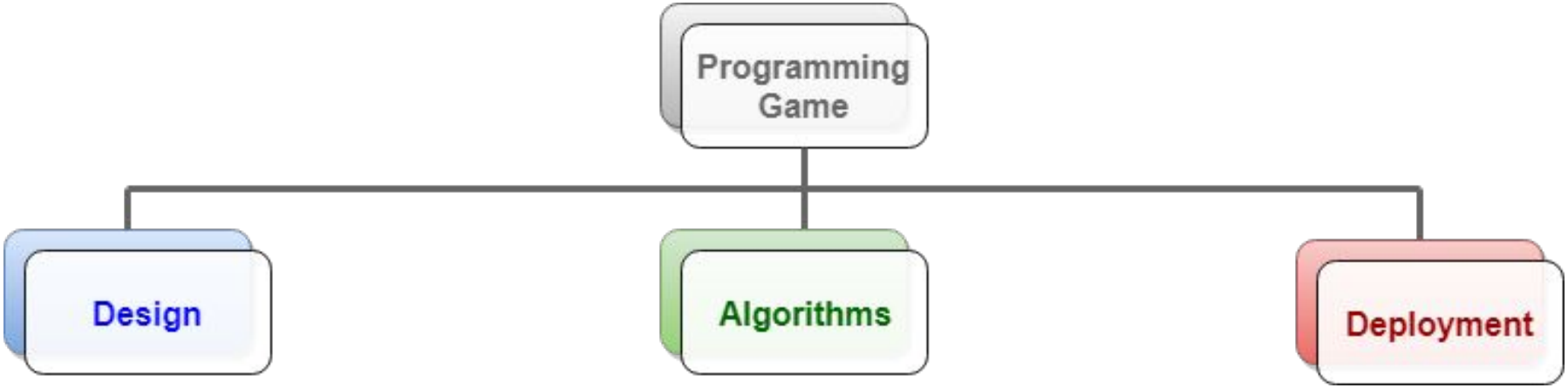
Version Control Flow Diagram



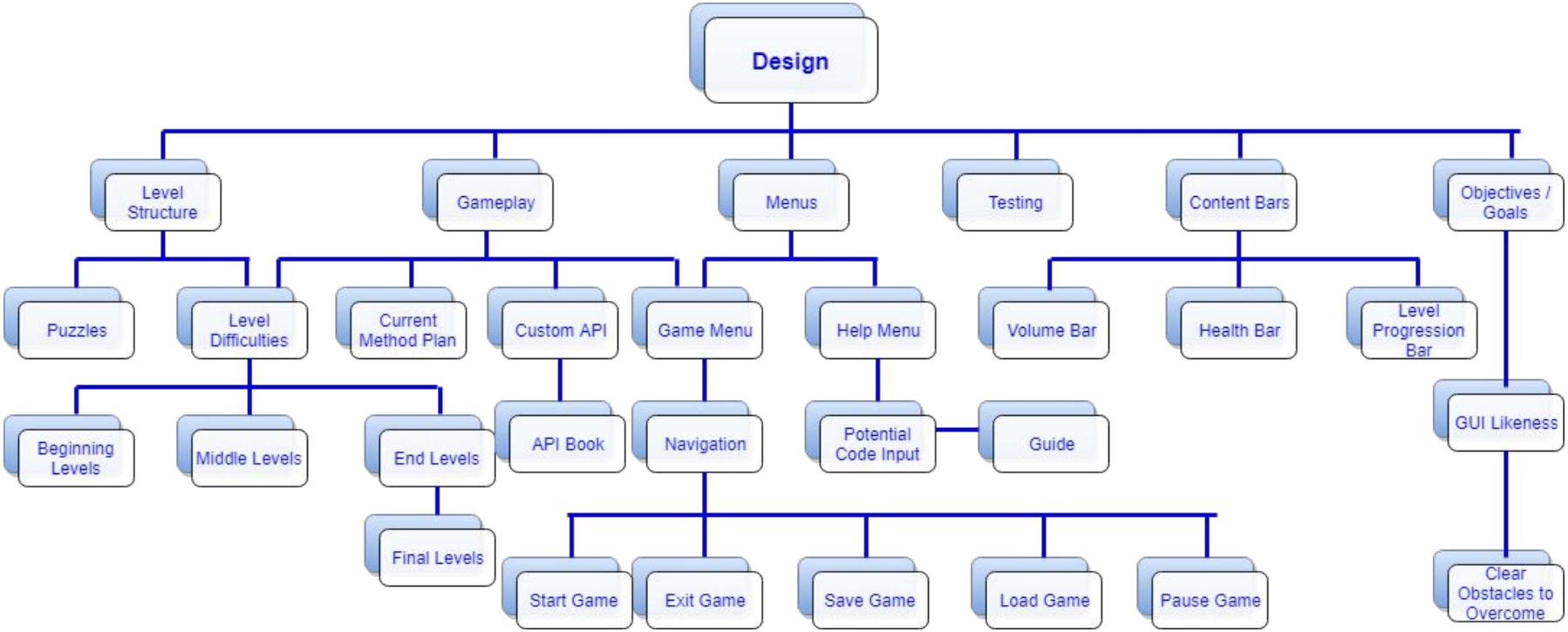
Agile Development Model



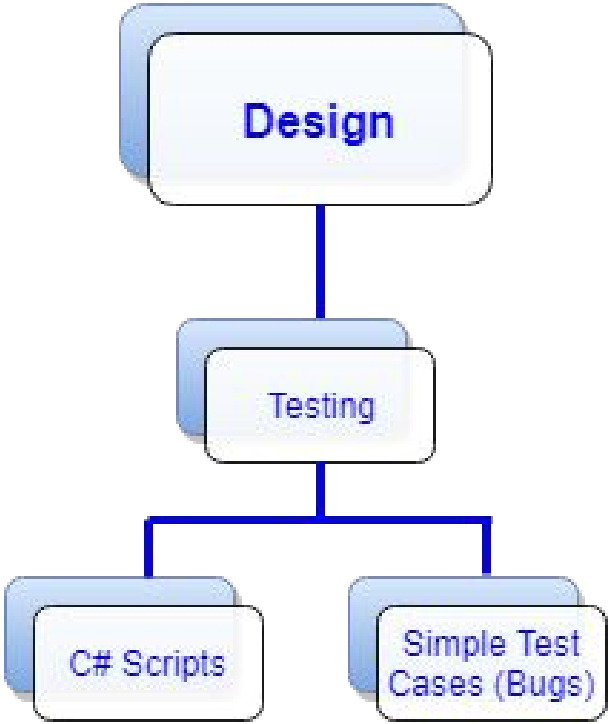
Work Breakdown Structure (WBS)



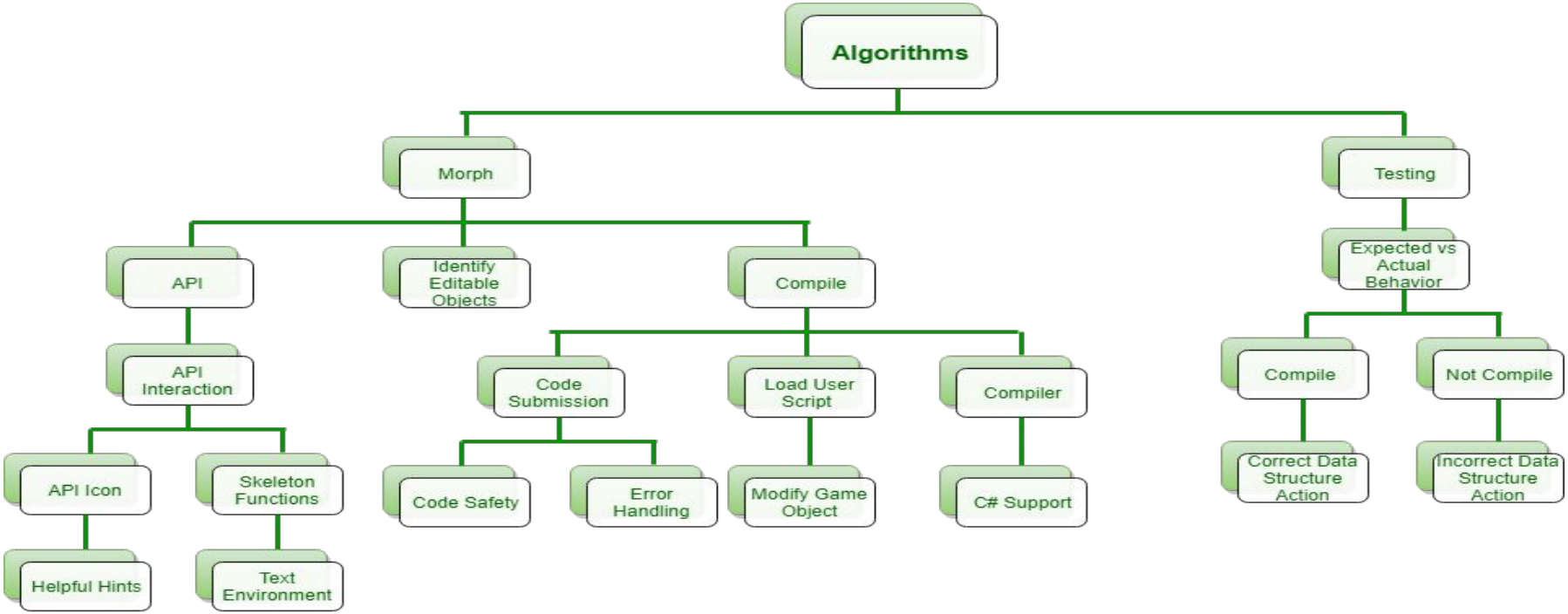
Work Breakdown Structure - Design



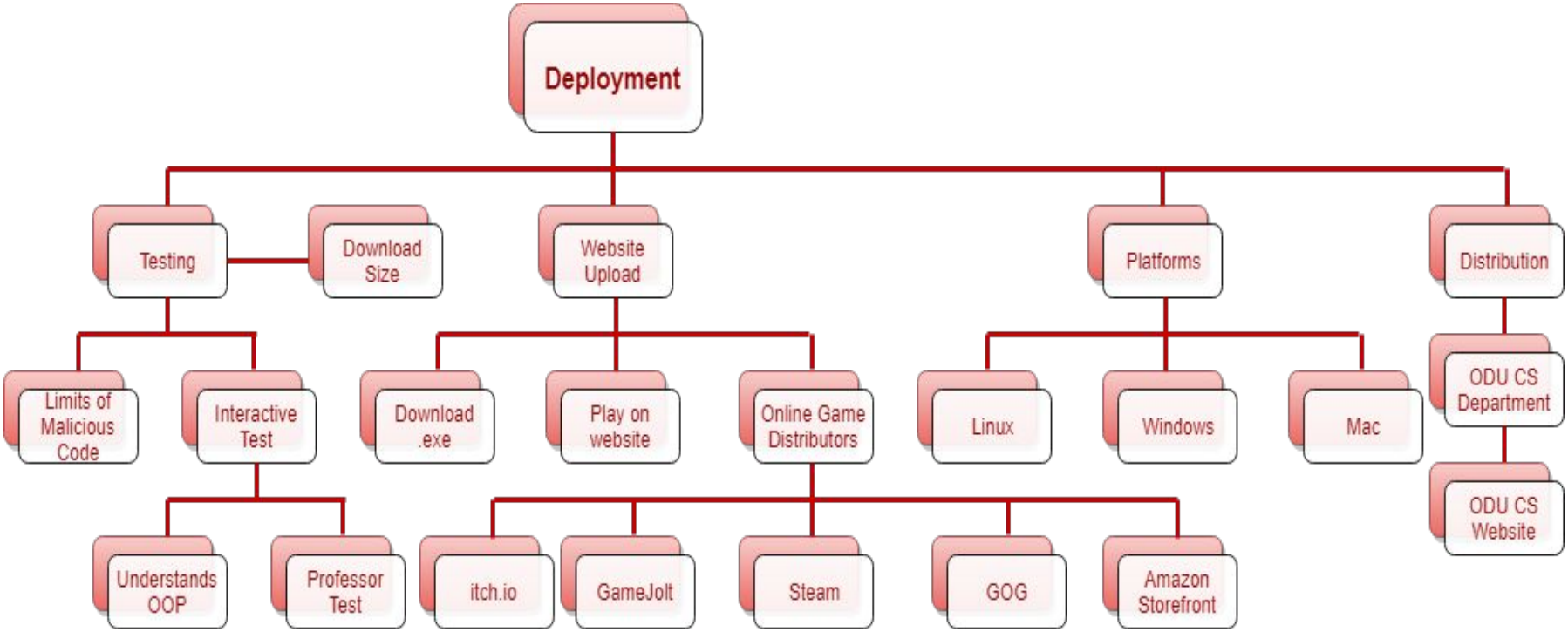
Work Breakdown Structure - Design (Testing)



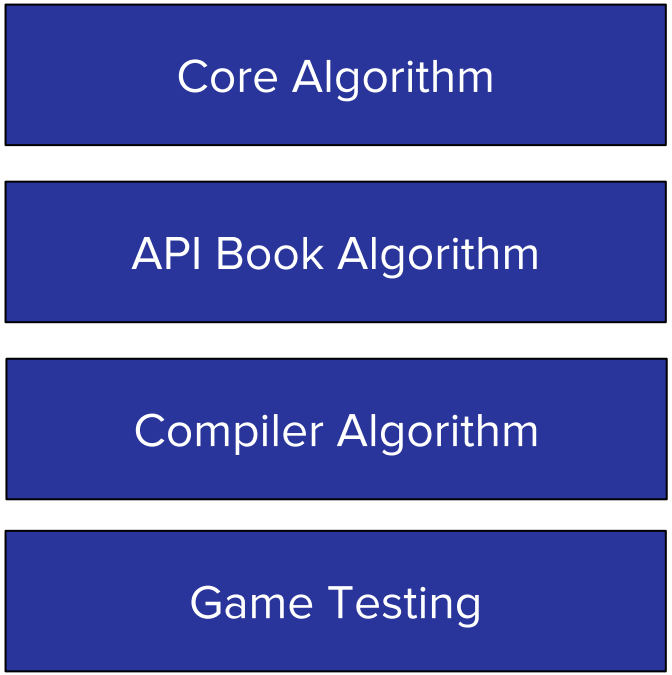
Work Breakdown Structure - Algorithms



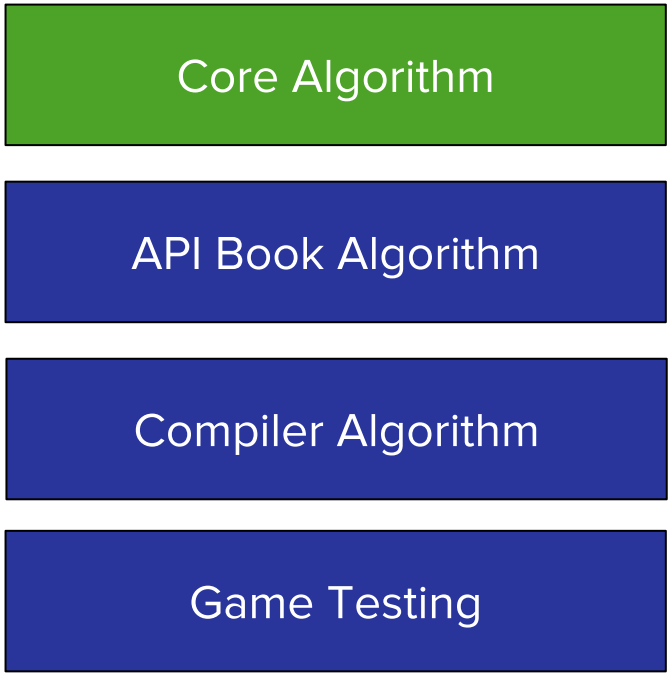
Work Breakdown Structure - Deployment



Software Iterations

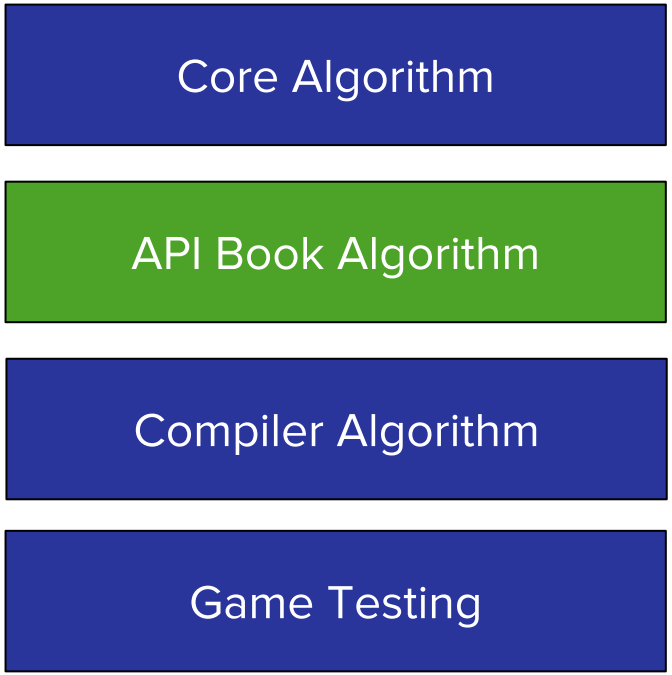


Software Iterations: Core Algorithm



- UI level algorithm
- Determines the base tools at the player's disposal
- Divides gameplay into distinct sections of:
  - Interaction through UI elements/Object selection
  - Information Provision through access to the API Book Algorithm
  - Alteration through the Compiler Algorithm

Software Iterations: API Book Algorithm



- Acts as primary method of information distribution from game designer to player
- Interacts directly with Compiler Algorithm by determining the knowledge base the player has to exploit in the Compiler Algorithm
- Directly influences the outcome of gameplay/challenges by offering the player a multitude of tools to interact with their environment

Software Iterations: Compiler Algorithm

Core Algorithm

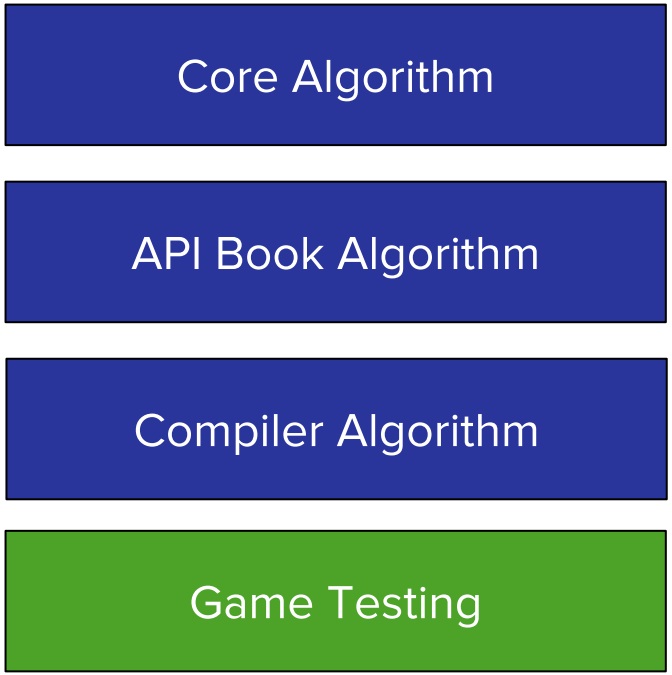
API Book Algorithm

Compiler Algorithm

Game Testing

- Controls and continuously affects the Back-End of the game itself
- Directly determines the behavior of objects in the game's environments at a fundamental level
- Responsible for the amount of control and open design power the player is granted during gameplay
- Co-dependant on the API Book Algorithm based on the tools the player is likely to find there to use within the Compiler Algorithms in-game dependencies

Software Iterations: Game Testing



Aspects to test:

- API Book in-game application
- Compiling of user-input source code
- Effectiveness of TUI and gameplay

Alpha Test

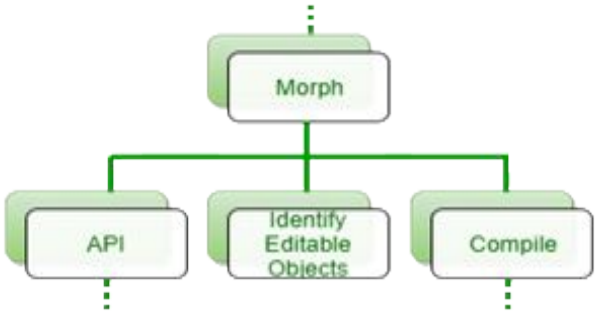
- Closed testing conducted by Team Silver
- The API Book will be the focus

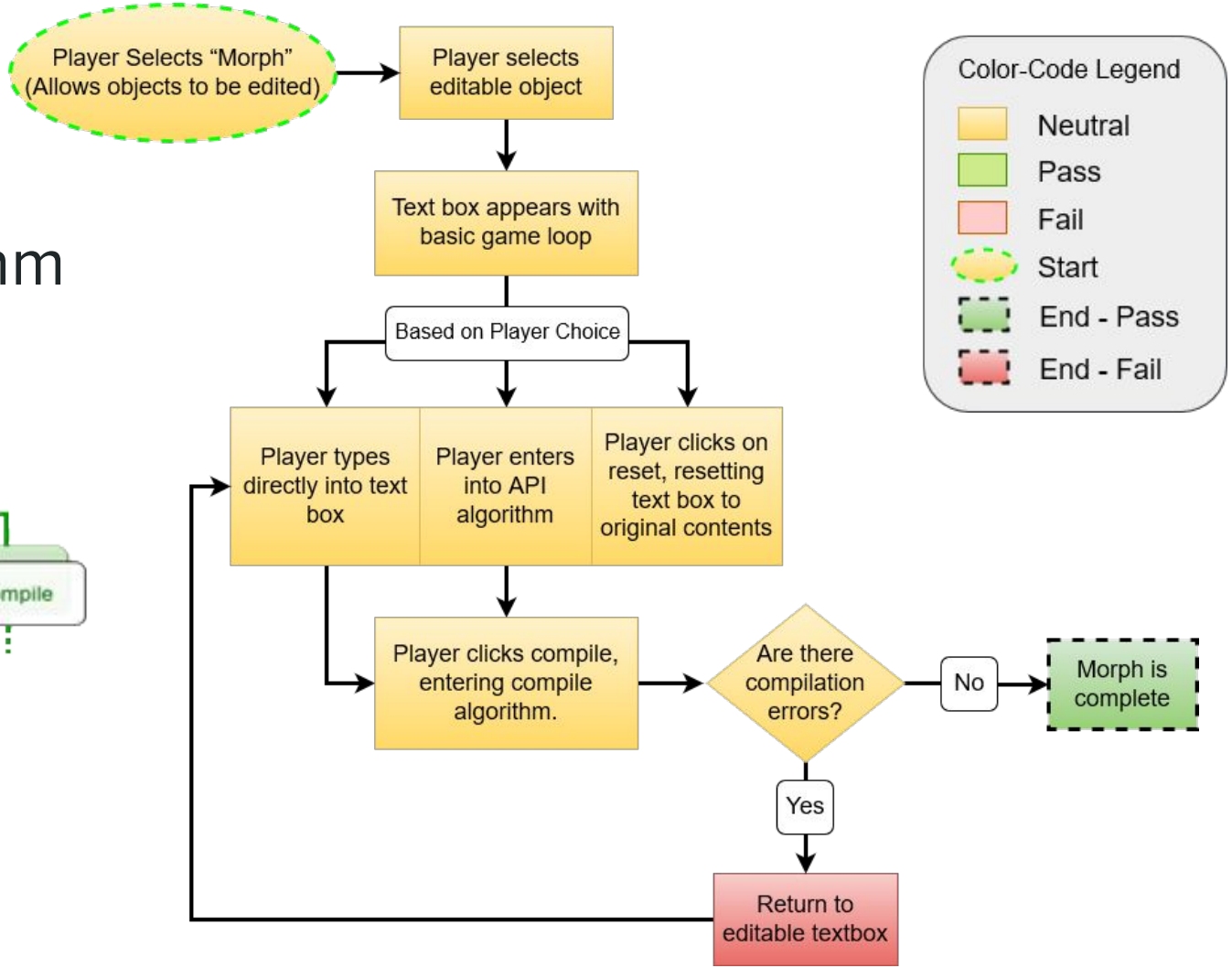
Beta Test

- Open testing conducted by ODU faculty
- The Compiling, TUI, and Gameplay will be the focus

Dataflow Algorithms

Core Algorithm

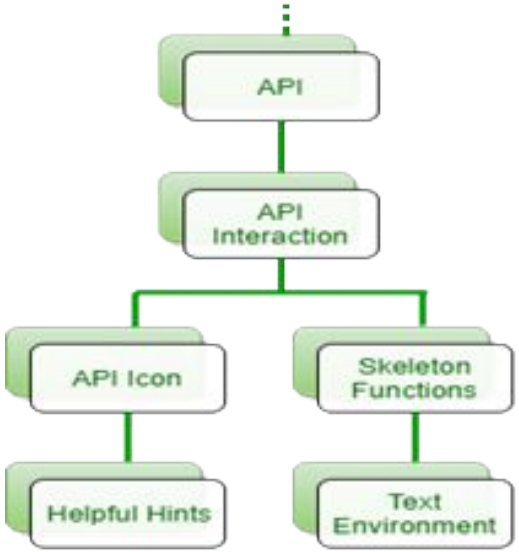


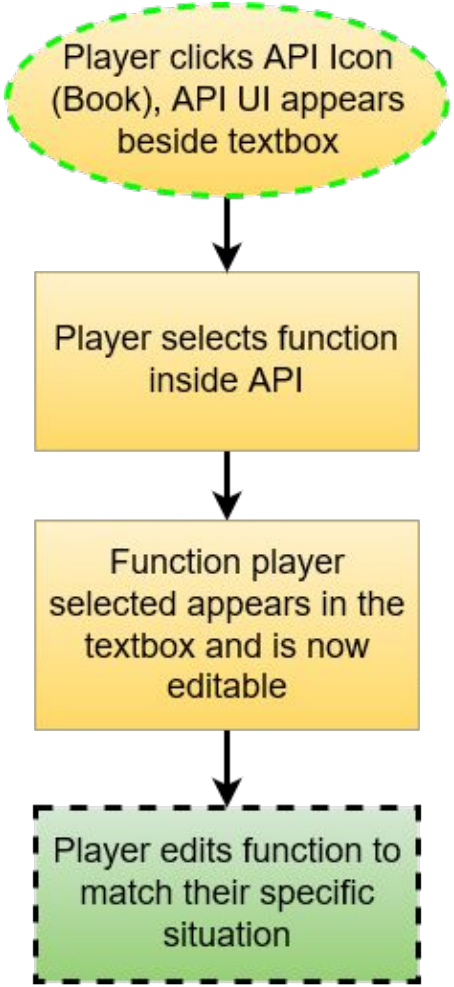


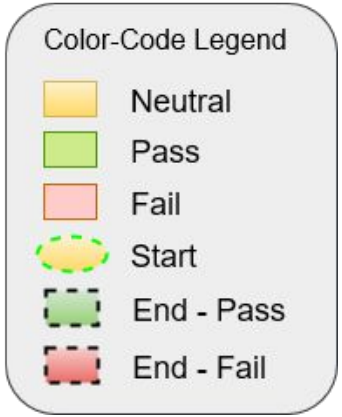
Core Algorithm: Pseudocode

```
Core Algorithm{
        if (Button Clicked = MORPH Button){
                Selectable System{
                        if (Object in Level is marked as Editable){
                                Make Objects Selectable
                        }
                }
                if (Object is Selected){
                        Make Objects no longer Selectable
                        Create Instance Of(Code Editing Interface)
                        Pass Selected Object Name onto Compiler Algorithm
                }
        }
        if (Button Clicked = API Button){
                Create Instance Of(API Book User Interface)
        }
}
```

API Book Algorithm





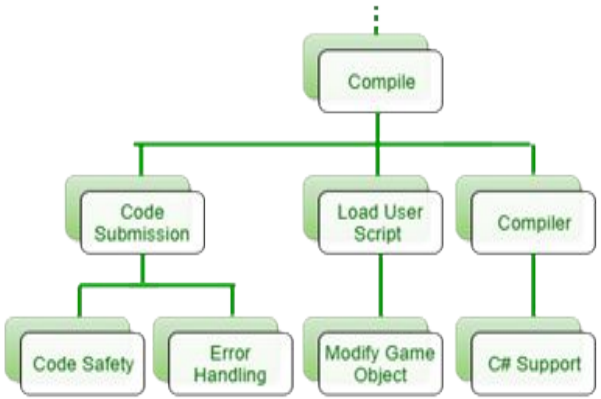


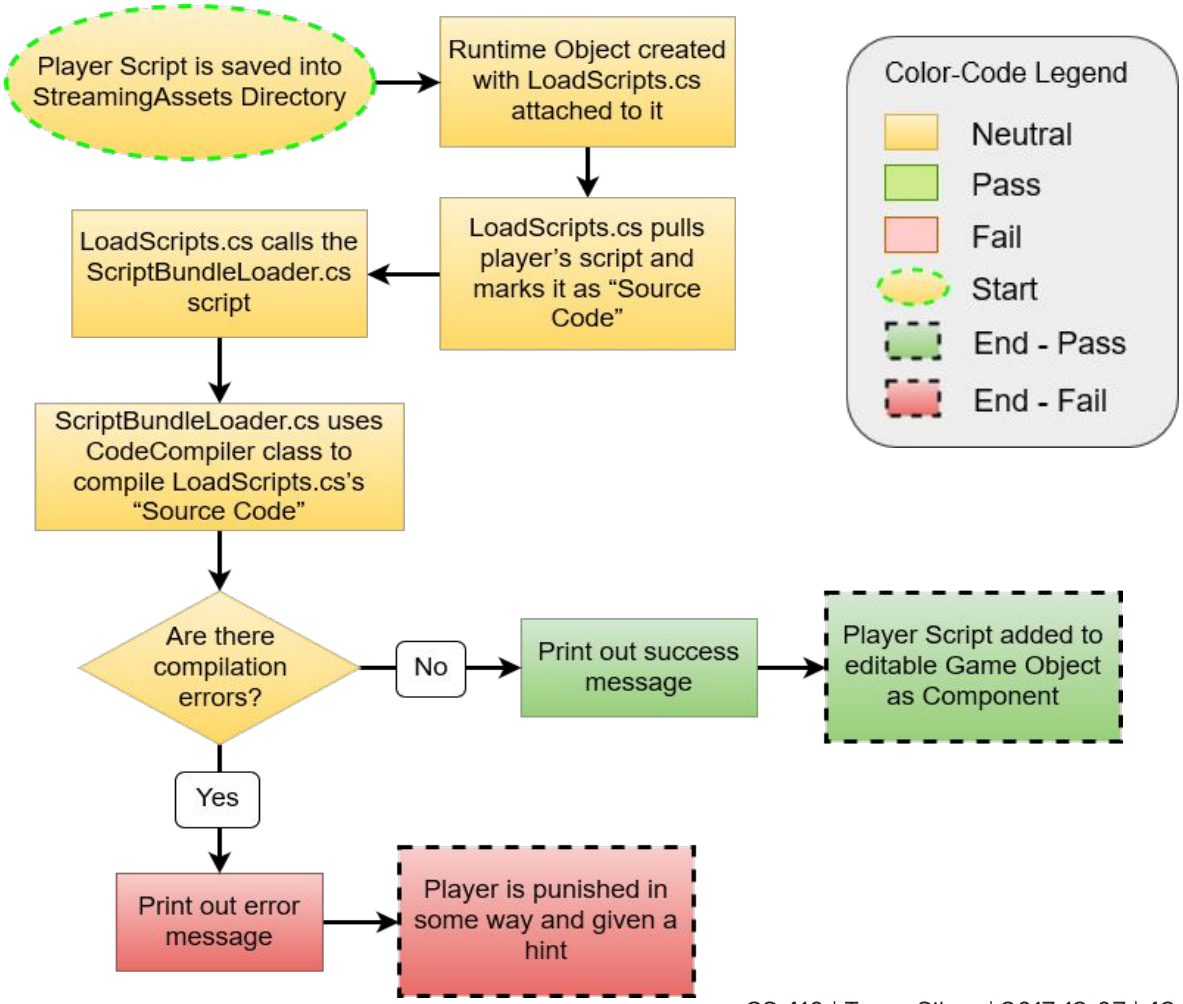
API Book Algorithm: Pseudocode

```
API Book Algorithm{
        Pull predetermined list of "functions" and "tools" for the player to learn this Level
        Fill Book{
                Proliferate "Pages" of the API Book with individual tool info
                Fill Text Box at bottom of Page with example code

                if (Button Clicked = Use This Code Button){
                        if (Code Editing Interface is Open){
                                Copy example Code to Code Editing Interface
                        }
                }
        }
}
```

Compiler Algorithm

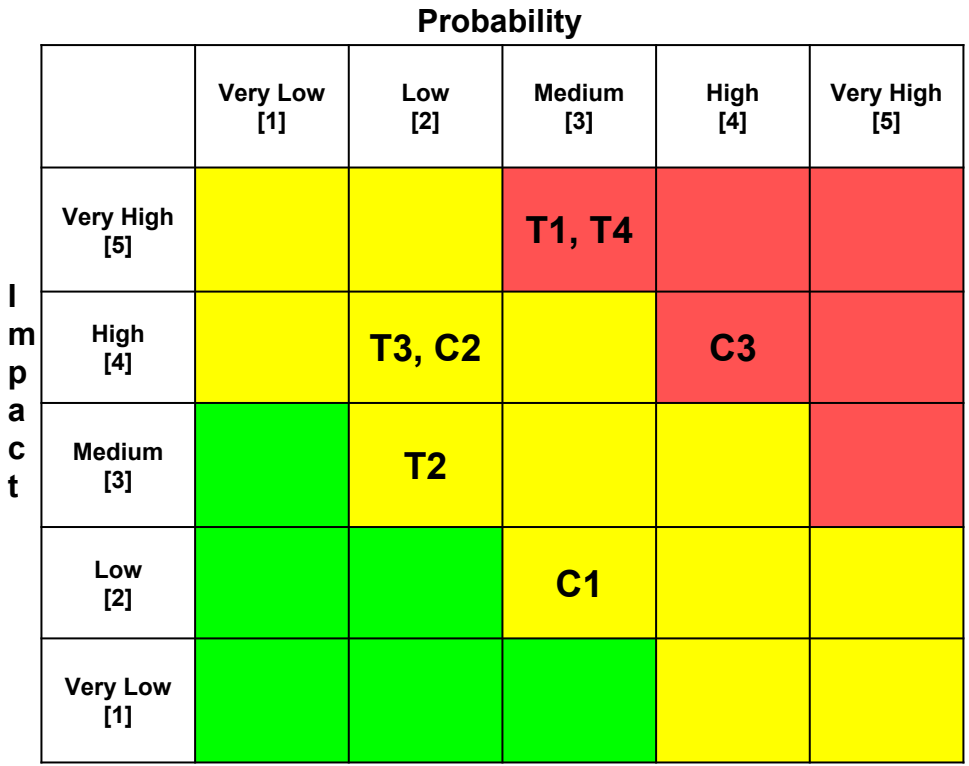




Compiler Algorithm: Pseudocode

```
Compiler Algorithm{
        Take name of Selected Object from Core Algorithm
        Load related Script from "StreamingAssets" folder into the Code Editing Interface
        if (Button Clicked = Close){
                Close the Code Editing Interface
        }
        if (Button Clicked = Reset){
                Revert Current Script back to template version
        }
        if (Button Clicked = Compile){
                Save new code to Script's source file
                Run a Sandboxed Compilation of that file in real time
                if (Compilation FAILS){
                        Create Text: ERROR Message containing Compilation Error line/type
                        Create Instance Of(Window with ERROR Message)
                }
                else (Compilation PASSES){
                        Create Instance Of(Window with SUCCESS Message)
                        Attach edited and compiled Script to Selected Object
                        Re-render level elements in real-time to calculate new object behavior
                }
        }
}
```

Risk Matrix



Risk Description

Customer Risks

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

Technical Risks

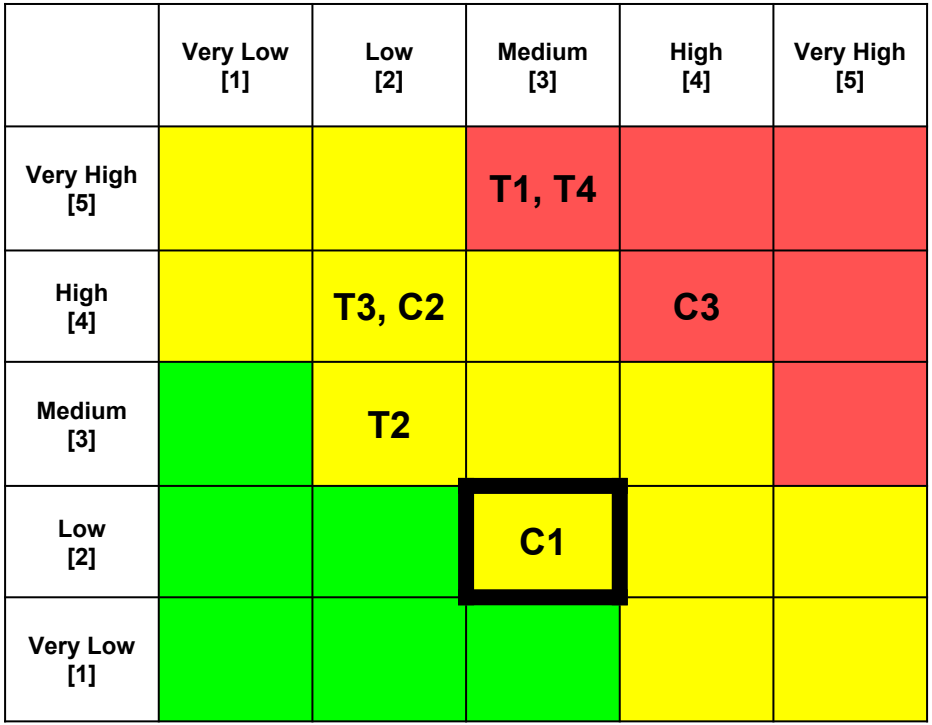
T1. User Implements Bad Code
T2. Insufficient Hardware
T3. Critical Software Bugs
T4. Insufficient API Support

C1 : User Gets Lost

C1: User gets stuck and does not know what to do - low level of technical experience

Medium Probability - Low Impact

Mitigation: Include enough resources and hints that the user can effectively learn the material

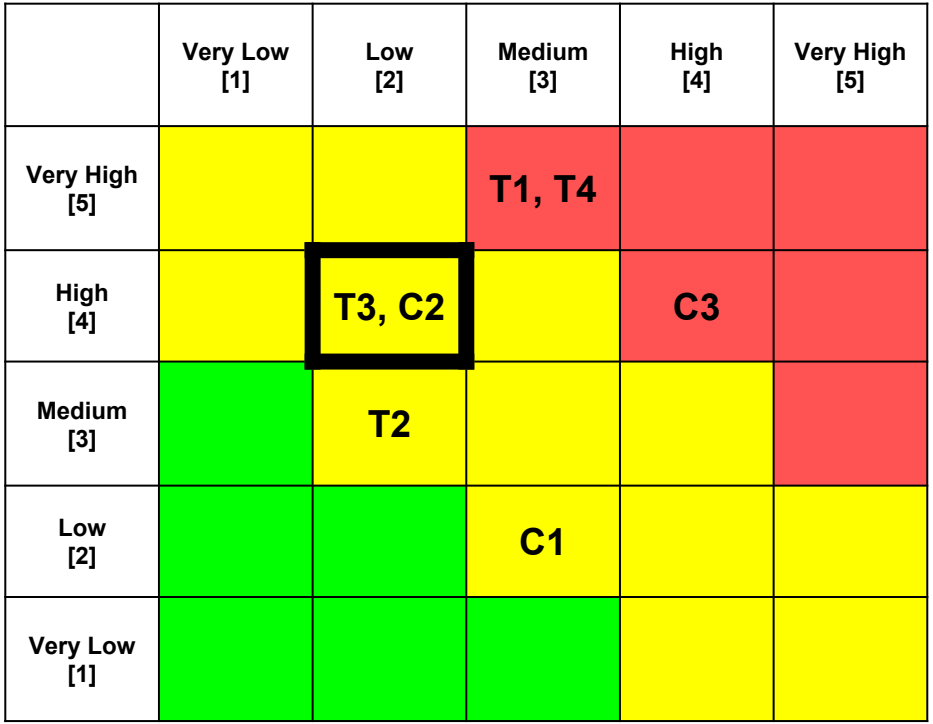


C2 : Dissatisfied User

C2: User dislikes the user interface(UI/UX)

Low Probability - High Impact

Mitigation: The UI/UX design will enhance an approachable interface, various menu options, and an interface that will include clear objects throughout each level

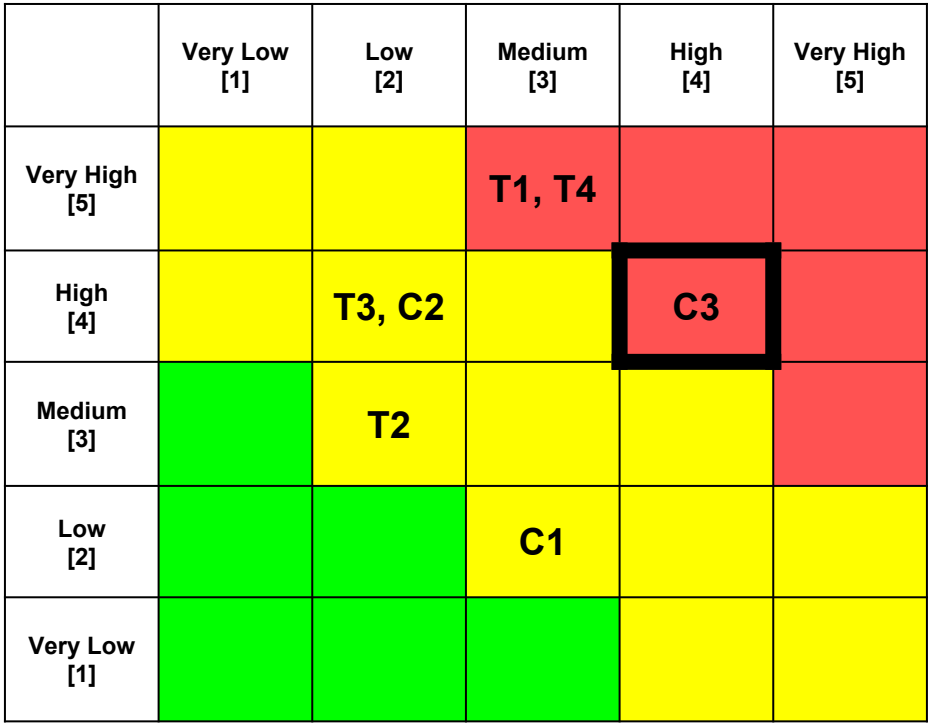


C3 : Insufficient Content / Time

C3: Not enough content / time in order to successfully pass introductory CS classes

High Probability - High Impact

Mitigation: Use play testing to optimize the pacing and content of the game

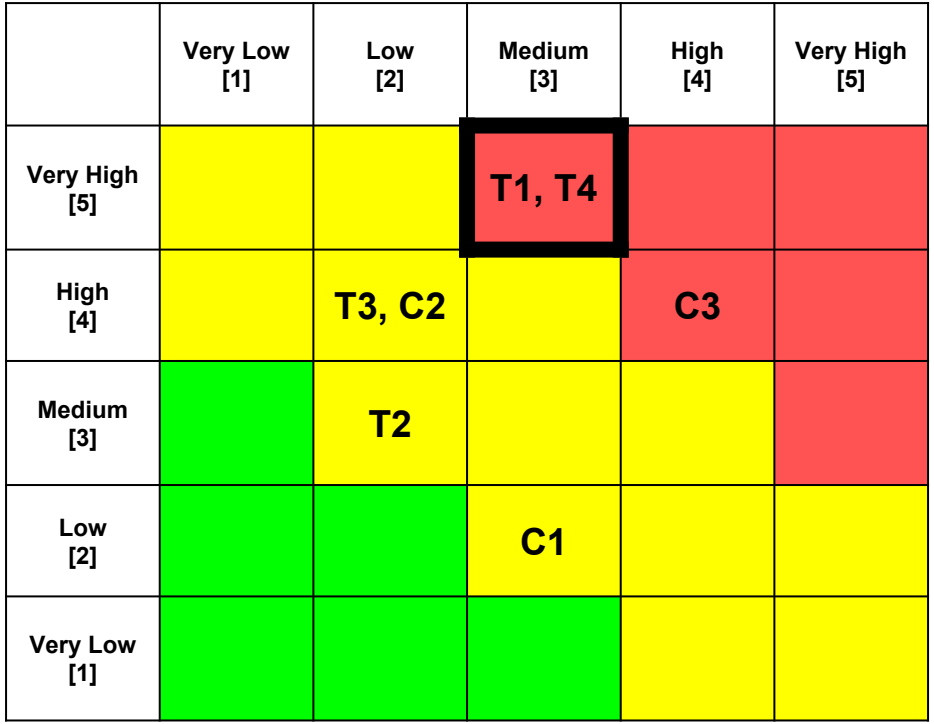


T1 : User Implements Bad Code

T1: User implements bad code that crashes the game either because of their own error or because of a malicious post on a forum

Medium Probability - Very High Impact

Mitigation: Under Evaluation - Identified a few potential solutions, but looking for one central solution



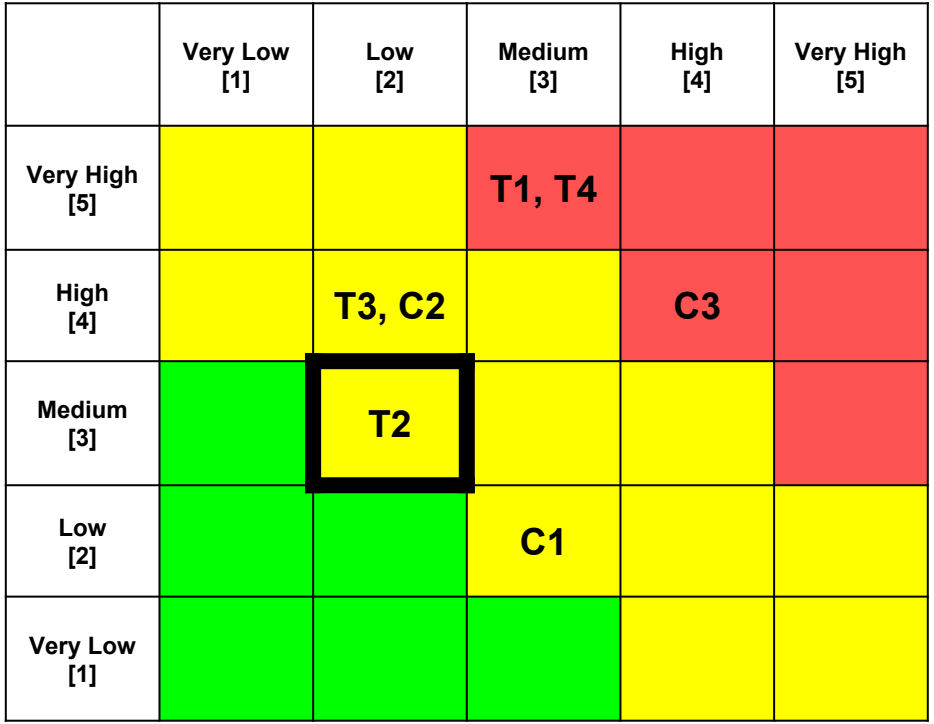
T2 : Insufficient Hardware

T2: User does not have sufficient hardware to run the game

Low Probability - Medium Impact

Mitigation: Implement a 2D game style that will have minimal draw on a computer's resources

Game will be optimized 4th gen i3 Intel Processor



T3 : Critical Software Bugs

T3: Possible critical software bugs in the game

Low Probability - High Impact

Mitigation: Continuously test software through gameplay and ensure that all possible interactions work well

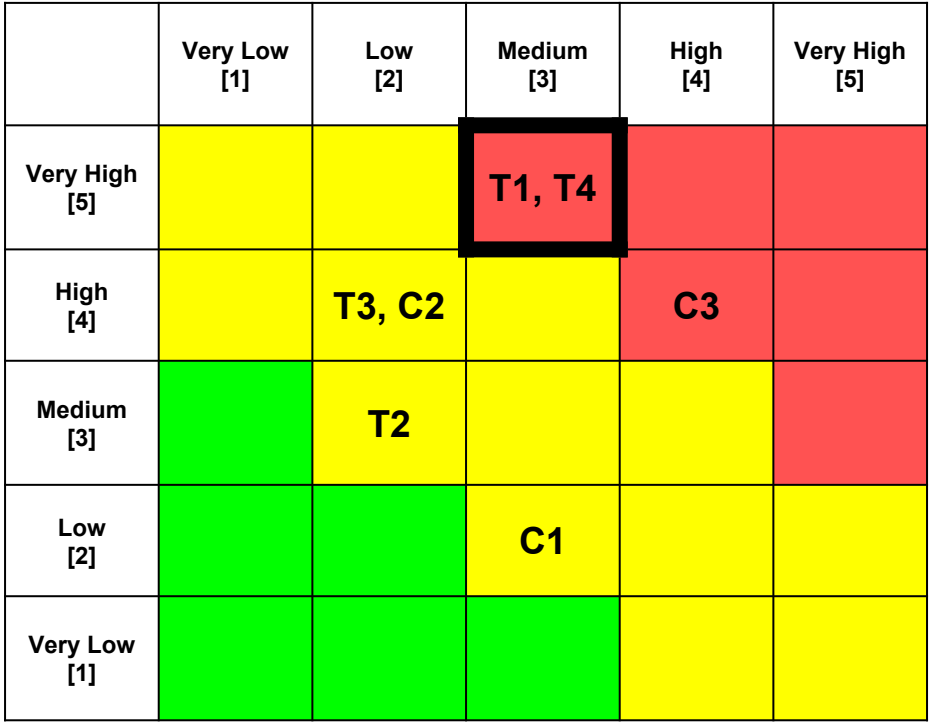
	Very Low [1]	Low [2]	Medium [3]	High [4]	Very High [5]
Very High [5]			T1, T4		
High [4]		T3, C2		C3	
Medium [3]		T2			
Low [2]			C1		
Very Low [1]					

T4 : Insufficient API Support

T4: Insufficient API support for Game help / aid

Medium Probability - Very High Impact

Mitigation: The built in API will have all the necessary tools to help the user refer the specific code syntax and OOP concepts they need to learn to get through the levels of the game



Benefits to Customer

- Supplies customer with a supplemental programming resource
- Naturally learns Object Oriented Design practices and the benefits of employing them
- In the case of the initially targeted student demographic, advantages not only in knowledge of potential course material but substantially better odds of success in said course
- A genuinely **Interactive** and **Enjoyable** experience

Conclusion

PolyMorpher

- CS students not introduced to OOP or problem solving skills early on
- Skills are essential to build a solid foundation for understanding CS, including programming
- Web-application or executable downloadable game using the Unity SDK with C# and JavaScript
- User learns Object-Oriented Programming (OOP) concepts and problem solving skills in depth
- Skills allow end user to become more proficient in Computer Science, as well as Object Oriented Design

Our Solution Makes the Process Painless & Fun

References

- “Fast Facts.” *Unity*, Unity Technologies, unity3d.com/public-relations.
- *Asset Store*, Unity Technologies, www.assetstore.unity3d.com/en/.
- Technologies, Unity. “Welcome to the Unity Scripting Reference!” *Unity - Scripting API:* Unity Technologies, docs.unity3d.com/530/Documentation/ScriptReference/index.html.
- O'Neill, M. (n.d.). Computer Science Before College. Retrieved October 05, 2017, from https://www.computerscienceonline.org/cs-programs-before-college/
- Peter Riley’s presentation. It will be one of the main source of references for our project.
- “kennedyData.” Thomas Kennedy, https://drive.google.com/drive/u/1/folders/0B_xCQd8Vk2BnSU1hNnJwSXB1NEE
- “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
- CS410 Programming Game Pitch, By: Joel Stokes - https://youtu.be/QBvgzFgZaOQ
- CS410 Project Dungeon Demo, By: Casey Batten - https://www.youtube.com/watch?v=ynhdd1lKgps
- CS410 Dungeon Escape Demo (Short Ver.), By: Casey Batten - https://www.youtube.com/watch?v=VnHRaWl8Y8w

References

- CS410 Tech Demo 2, By: Casey Batten - Download Source Code - http://www.cs.odu.edu/~410silver/demos.html
- “12 Free Games to Learn Programming.” Mybridge, https://medium.mybridge.co/12-free-resources-learn-to-code-while-playing-games-f7333043de11
- currentProcessFlow.html, https://www.draw.io/?state=%7B%22ids%22:%5B%220B3Bc95zBWXg9TFZ6X0FMU1NTdEk%22%5D,%22action%22:%22open%22,%22userId%22:%221086920031335905830 47%22%7D#G0B3Bc95zBWXg9TFZ6X0FMU1NTdEk
- ProcessFlowDiagram_silver.html, https://www.draw.io/?state=%7B%22ids%22:%5B%220B_xBnZ1ge4PlZTVjV3h6Y2pGSWc%22%5D,%22action%22:%22open%22,%22userId%22:%221086920031335905830 47%22%7D#G0B_xBnZ1ge4PlZTVjV3h6Y2pGSWc
- Current Process Flow, https://www.draw.io/?state=%7B%22ids%22:%5B%220B-5KdQEdqLUPdnBFUnp2V05uMEE%22%5D,%22action%22:%22open%22,%22userId%22:%221086920031335905830 47%22%7D#G0B-5KdQEdqLUPdnBFUnp2V05uMEE

References

- Practical Rules for Using Color in Charts, http://www.perceptualedge.com/articles/visual_business_intelligence/rules_for_using_color.pdf
- Standard Flowchart Symbols and Their Usage, https://www.edrawsoft.com/flowchart-symbols.php
- Work Breakdown Structure (WBS), https://www.draw.io/?state=%7B%22ids%22:%5B%220B-5KdQEdqLUPWnNoSHhlUGg2OTQ%22%5D,%22action%22:%22open%22,%22userId%22:%221086920031335905830 47%22%7D#G0B-5KdQEdqLUPWnNoSHhlUGg2OTQ
- VersionControlFlow.html, https://www.draw.io/?state=%7B%22ids%22:%5B%221lQj6SYJqC6YLAK_qMRVlQkHiUmr9laBu%22%5D,%22action%22:%22open%22,%22userId%22:%221086920031335905830 47%22%7D#G1lQj6SYJqC6YLAK_qMRVlQkHiUmr9laBu

Appendix A : User Stories - Roles Defined

Student: A person playing our game with the goal of learning more about OOP

Player: A person playing our game with the goal of having an enjoyable game experience

*__Student__ and __Player__ may be the __same person__ in many cases wanting to both learn and enjoy the game

Appendix A : User Stories - Student

1. As a student, I would like to be taught inheritance.
2. As a student, I would like to be taught abstraction.
3. As a student, I would like to be taught encapsulation.
4. As a student, I would like to be taught polymorphism.
5. As a student, I would like to be taught functional design patterns.
6. As a student, I would like to be taught how to use an API.
7. As a student, I would like to be taught what all is contained inside an object.

Appendix A : User Stories - Player

1. As a player, I would like to interact with an intuitive UI.
2. As a player, I would like to be able to track my progress.
3. As a player, I would like to be able to freely manipulate game objects.
4. As a player, I would like challenges that are meaningful and rewarding (quality).
5. As a player, I would like challenges that are engaging (difficulty).
6. As a player, I would like save my progress.
7. As a player, I would like to access to various game menus.
8. As a player, I would like help when I fail at a challenge.
9. As a player, I would like a tutorial on how to move in this game.
10. As a player, I would like a tutorial on how to edit objects in this game.
11. As a player, I would like a tutorial on how to use the coding environment in this game.

Appendix B : Student Progression Dilemma Statistics

- Students are not following the course series in the expected order
  - This data provides purpose behind the need to reassess methods of assisting students' progress in how they learn the topics at hand
  - When during the learning process this assistance would be most effective
- Changes in class volume, which could indicate that students are leaving the major for less intense fields
- Drop-off in student numbers from CS150 to CS250
  - Related to differing major requirements and course overlap, but the decrease in student body is significant enough to warrant a deeper look into later classes in the major path
- Decreasing class sizes show a steady decline in CS course enrollments as course level difficulty advances
  - Decreases may be indicative of students dropping out of the CS program or changing majors

Appendix B : Student Progression Dilemma Statistics

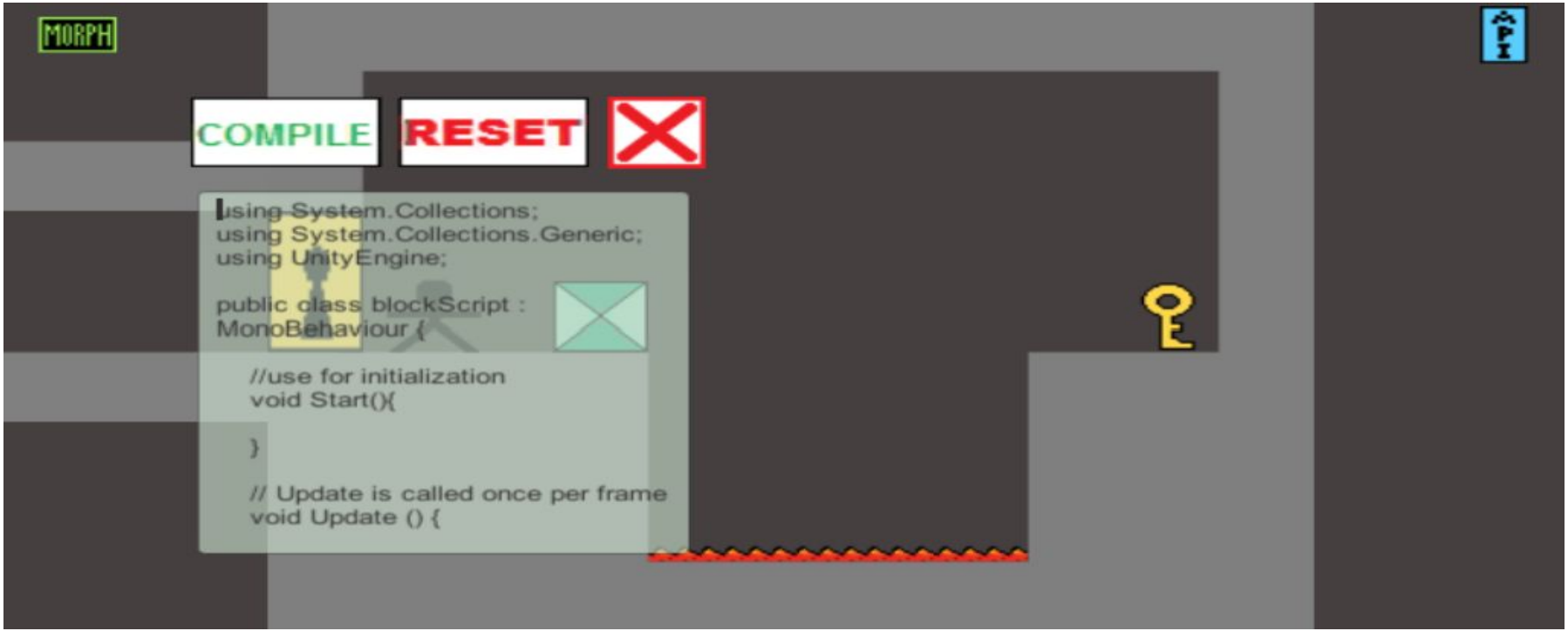
According to the **ODU Factbook**:

- 2012 - 2016:
  - Number of undergraduate CS majors increased from 284 to 429
    - Showing the high demand in the degree path
- 2014 - 2015:
  - Roughly 672 students enrolled in CS150
- 2015 - 2016:
  - Roughly 327 students enrolled in CS250
- 2016 - 2017:
  - Roughly 199 students enrolled in CS361
  - Roughly 180 students enrolled in CS330
  - Roughly 182 students enrolled in CS350

Appendix C : Rapid Prototype GUI Sample - Joel

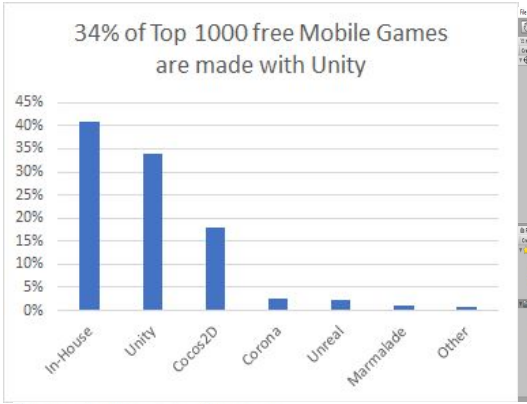
Appendix C : Rapid Prototype GUI Sample - Casey

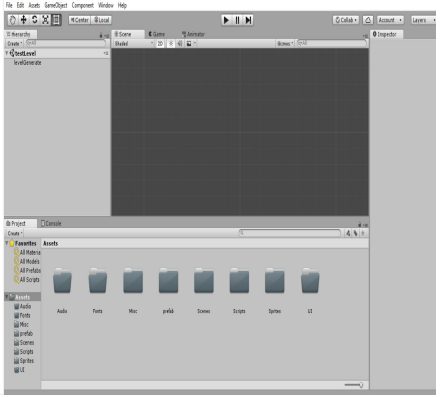
Appendix C : Rapid Prototype GUI Sample - Casey



Appendix D : Unity SDK Information

- Flexible UI and developer workflow system
  - Allows users to develop a product efficiently
- Tools for software development:
  - MonoDevelop IDE and support for multiple platforms and build environments
- According to **Unity Technologies**, there were over 5 billion downloads of products made with Unity in quarter one of 2016, with an extra 2.4 billion in mobile product downloads





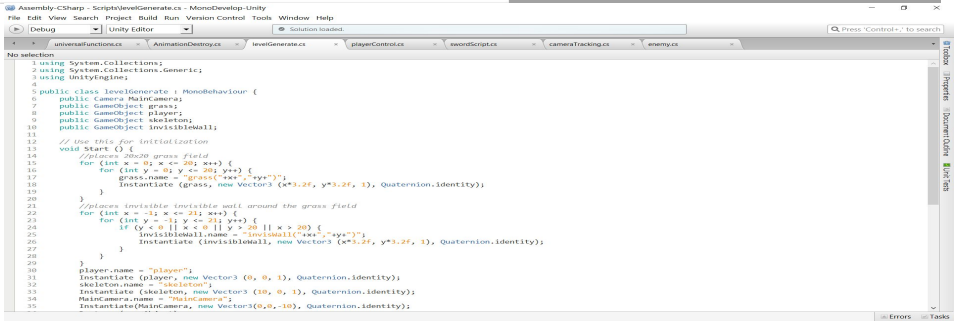


Image Source: https://unity3d.com/

Appendix D : Unity SDK Information

- Most flexible and powerful of the Out-of-the-Box programming languages Unity supports
  - Large suite of built in functions/methods
  - Unity-specific tools available when used alongside the Engine
- Developer support available through Unity's Scripting API website
  - Code examples
  - Complex breakdowns of Unity-specific functionality within C#

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class universalFunctions : MonoBehaviour {

    public static GameObject universalObjectSearch(string obj_name){
        GameObject[] gameObjects = (GameObject[])FindObjectsOfType(typeof(GameObject));
        for (int i = 0; i < gameObjects.Length; i++) {
            if (gameObjects [i].name.Contains (obj_name)) {
                return gameObjects[i];
            }
        }
        return null;
    }
}
```

Image source: https://docs.unity3d.com/Manual/index.html?_ga=2.37814243.1199564661.1509584726-986472080.1506709735