Project Rob

Jerry Cai, Nick Clark, Kaden Gilbert,
Gavin Sloan, Nova Solarz

GitHub Repository:
https://github.com/gilbertk23/Code-R.O.B

Our Team

Team Lead – Kaden

- Created GitHub repository and Kanban project
- Completed Executive Summary, Timeline, etc.
- Coordinated with other team members to track progress
- Organized team meetings with other team members

Code Lead – Gavin

- Led the game's implementation
  - Determined elements within scope and out of scope
- Coordinated with other team members to update them on the development process

Design Lead – Nick

- Created data flow diagram, problem frames, and UML diagram
- Generated ideas relevant to the game's design, aesthetics, and features
  - Provided sprite designs and assisted with their implementation
- Coordinated with other team members to make sure the game's design matches what is being implemented

Documentation Lead – Nova

- Updated the physical documentation and repository as the project progressed
- Assisted with coding and created background music
- Designed main menu
- Coordinated with other team members to update them on the game's development

Security Lead – Jerry

- Completed testing on game elements and code
- Assisted with coding the game’s collision detection, sprites, and final boss
- Coordinated with Code Lead and Documentation Lead to make sure game components function properly

Executive Summary

The problem we are trying to solve is eliminating boredom, stress, and restlessness by utilizing entertainment. The process we are taking to resolve this issue is developing a roleplay video game with its design based off of the original Legend of Zelda game. We will be creating a main menu, various map features, background music, and other elements common in retro two-dimensional roleplay games. Our proposed solution is one that will accommodate players of all ages who wish to relax after a long day, on a weekend, or when visiting friends and family. Our primary expected deliverable is our roleplay game that will be gradually developed throughout this project.

Problem Statement

Our project aims to solve problems related to boredom, stress, and/or a lack of entertainment for the average person. By resolving these issues, we hope to provide a fun roleplay-style video game where the user can spend their free time exploring various "levels", items, and ultimately defeating a final boss.

Statement of Work

Project Objectives

- Create a fully functional two-dimentional roleplay game
- Include a graphical user interface (GUI)
- Include multiple "levels" that the game will progress through
- Include background music

Project Scope

The scope of our project is to create a roleplay game with a GUI displaying a top-down view, grid/tile-based levels, pixel art, background music, enemies, a final boss, and various game items. We will NOT be creating a game that takes multiple hours of progression to fully complete, multiplayer interaction, or internet connectivity.

Team Expectations

As a team, we expect to develop a fully functional two-dimentional roleplay game with a main end goal, multiple "levels" to progress through, and minimal bugs/errors.

Coding Standards/Practices

- The Python programming language will be the only programming language used.
- All directories are named in titlecase. ex: Example_Dir_Name
- All variables and functions are written in lowercase snake case. ex) example_var_name, example_function_name(param_1, param_2)
- Constants are named in uppercase snake case. ex) EXAMPLE_CONSTANT_NAME
- Comments must be utilized for each major function.
    - single # for notes and descriptions.
    - triple ### for named sections ex) ### Section Name ###
- All functions should have only one clearly-defined purpose, be no longer than ten lines long.
    - Note: Constructors and to-string functions are exempt from this clause.
    - Any other exceptions to this rule must have a clear rationale written in a comment.
- The program should follow proper OOP etiquette
    - private data attributes are only directly referenced within getters and setters
    - Superclass-subclass paradigm is followed if applicable
- Tiles on the game map will be 16x16 pixels
- Sprites (players, NPCs, items) on the game map will be 32x32 pixels

Outcome Summary

- Our final product will be a fully functional two-dimentional roleplay game.
- Implementing separate "levels" in the game.
- Having a final boss by the end of the project's initial development.
- Having enemies located throughout the game's map.
- Having background music that plays during gameplay.
- Having a main menu that displays upon the game's startup.

Roleplay Game Project Timeline

Week One

- Generate initial requirements/scope
- Create initial diagrams
- Develop/test first prototype
- **Milestone:** Completed prototype of roleplay game

Week 2

- Reevaluate requirements/scope
- Redesign diagrams accordingly
- Develop/test second prototype
- **Milestone:** Expanded prototype of roleplay game

Week 3

- Finalize requirements/scope
- Finalize diagrams
- Develop/test final prototype
- **Milestone:** Fully functional roleplay game

Final Product

- Finalize documentation/repository for submission
- **Milestone:** Final product demonstration

Requirements

Hardware Requirements

In order for the developed video game to work, the following hardware is required:

- A working computer, mouse, and keyboard
- A constant power supply to maintain the computer's operability
- An installation of the most recent Python interpereter

Software Requirements

The only programming language being used for this project is Python. In order for the video game to be developed, the following software is required:

- A standard internet browser/internet connection (for GitHub use)
- Music generation software (for background music creation)
- An integrated development environment (IDE) suited for running Python programs (for development and testing)
- A pixel art sprite creator (for designing items, characters, enemies, etc.)
- A diagramming software such as draw.io (for DFDs, Problem Frames, UMLs, etc.)

User Requirements

In order for the developed video game to work, the user will require the following:

- Motor skills such as hand and eye coordination
- The ability to solve problems related to the game's plot
- A computer (desktop/laptop), mouse, and keyboard

Security Requirements

Input Validation:

- All input needs to be validated whether it's out of bounds handling, type checking, etc..
- All data with no input or output (yet) has to be null.
- Try and excepts should be used on most if not all validation checks.
- If a try and except fails, display error message to explain why it failed.
- Maybe try to implement a black box function that checks for inputs.

Unit Tests

- Make sure to try and use unit test from the python libraries to run test.
- Try to create a test with each function (if possible).
- Try to do other ways of unit tests.

Security Requirements cont.

GUI

- Make sure inputs are working correctly.
- Make sure user can't crash the GUI by spamming random characters.
- Make sure user can't break the game.

OOP Practices

- Make sure variables are private especially since we are dealing with OOP.
- Make sure we use getters and setters.
- Make sure we use default constructors and construtors.

Security

Security Statement

As a security team lead, we want to make sure our data is as safe as possible. Data being safe is my main priority as the security lead so we'd like to put measures into place to counter these unsafe practices.

- Make sure we use private variables so outsiders can't access code
- Make sure we validate input so we get the correct input into whatever we need
- Make sure GUI syncs with the main code so there isn't any delay or lag between the two
- Make sure to have error checking ex: __private_variable = -1;
- Try and use try and catch exceptions to remove those pesky exceptions

Overall, let's run unit tests and let's create a safe RPG program where the user can't completely break the game! If we manage to follow the following practices then we'll be able to better the user's enjoyment of our lovely game! We hope to have a great outcome of a fantastic RPG and to do so, we must be able to put security ahead first.

Security Design Document

Assets and Their Threats

Assets to Consider

- GUI Main
- Main executable program
- Objects (Getters / Setters / Constructors)

Threats to Assets

Main / GUI Main Executable

•Malicious users
•Improper overwriting
•Pages being scraped
•Improper user input

Objects

•Incorrect use getters / setters / helpers
•Incorrect use of variables (not being private)
•Improper user input
•Malicious Users

Security Measures

Main / GUI Main Executable

- Try and except all input to handle, format, and validate all data correctly
- Input validation black box functions
- User given very limited access to limit any accidental or malicious problems cause by the user
- Clean, check, and return the correct user input matching the variable needed etc...

Objects

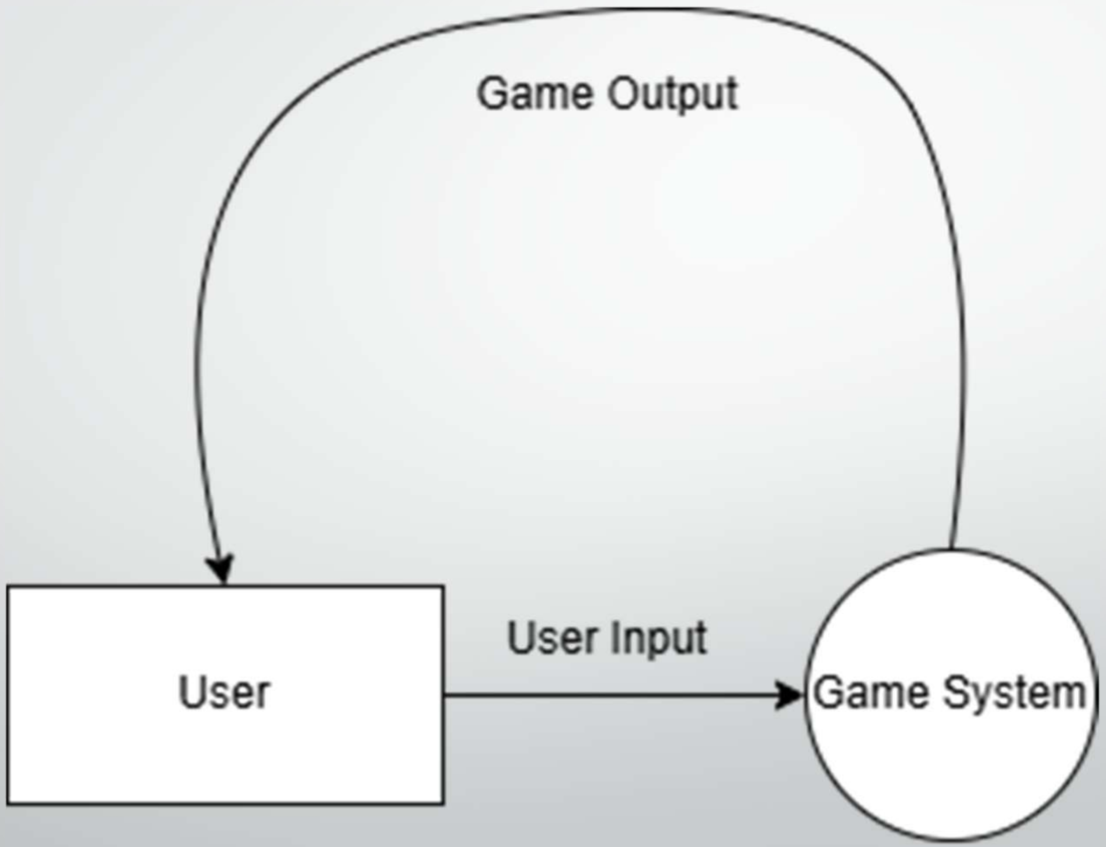
- Try and except all input to handle, format, and validate all data correctly (if can be done)
- Input validation make sure setters and getters are correct
- Make sure the variables ARE PRIVATE

Security Summary

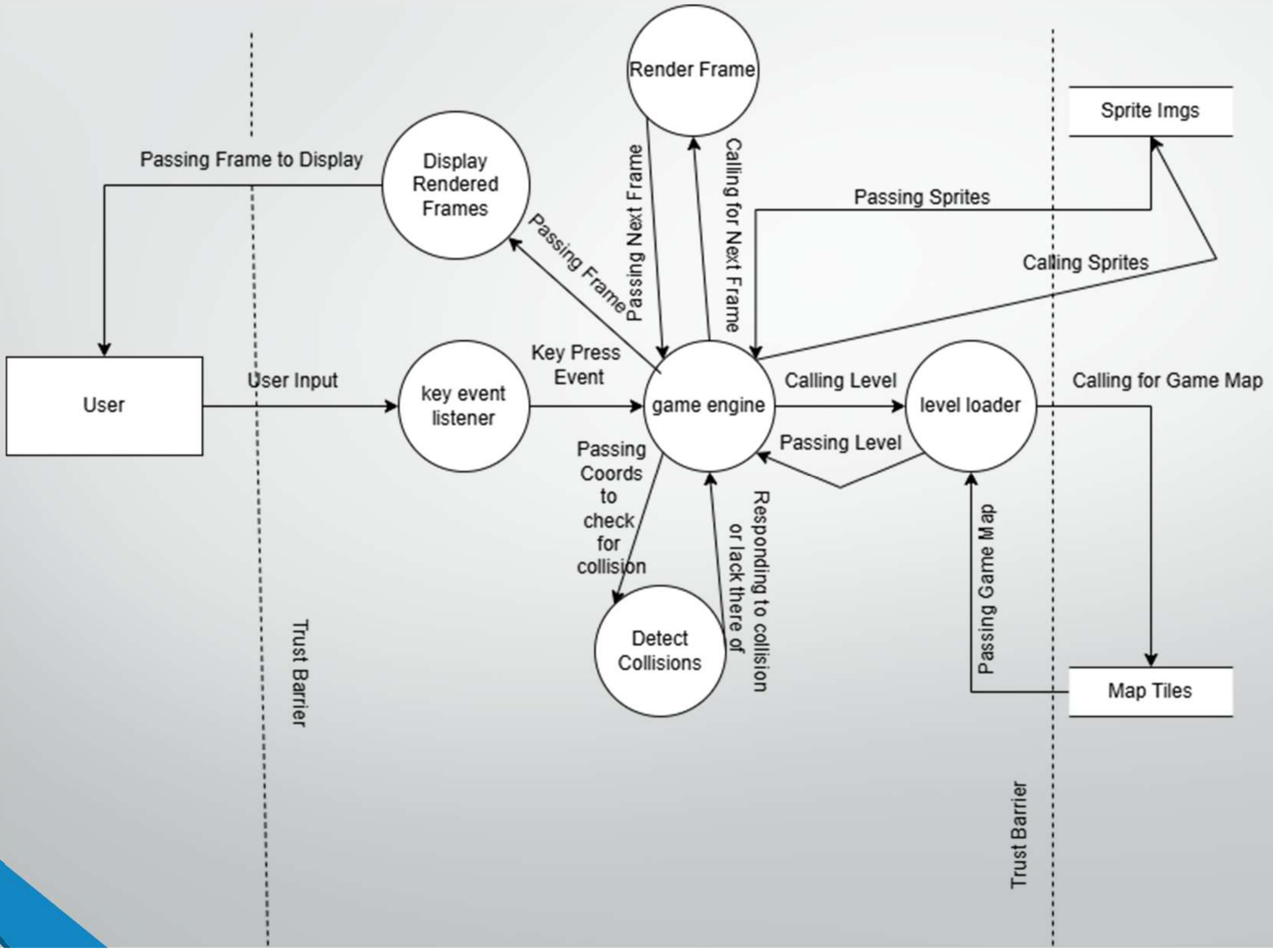
Overall, the security we're dealing with mainly comes from the input validation and the clumsiness of the user itself and poorly written code. We must make sure that we validate all functions and try to make sure we use try and catch statements. Our code must be safe with private variables and be clean. Another thing that would be absolutely great just to add is the code being readable so that any person can gloss through it and see which function does what.

Design Diagrams

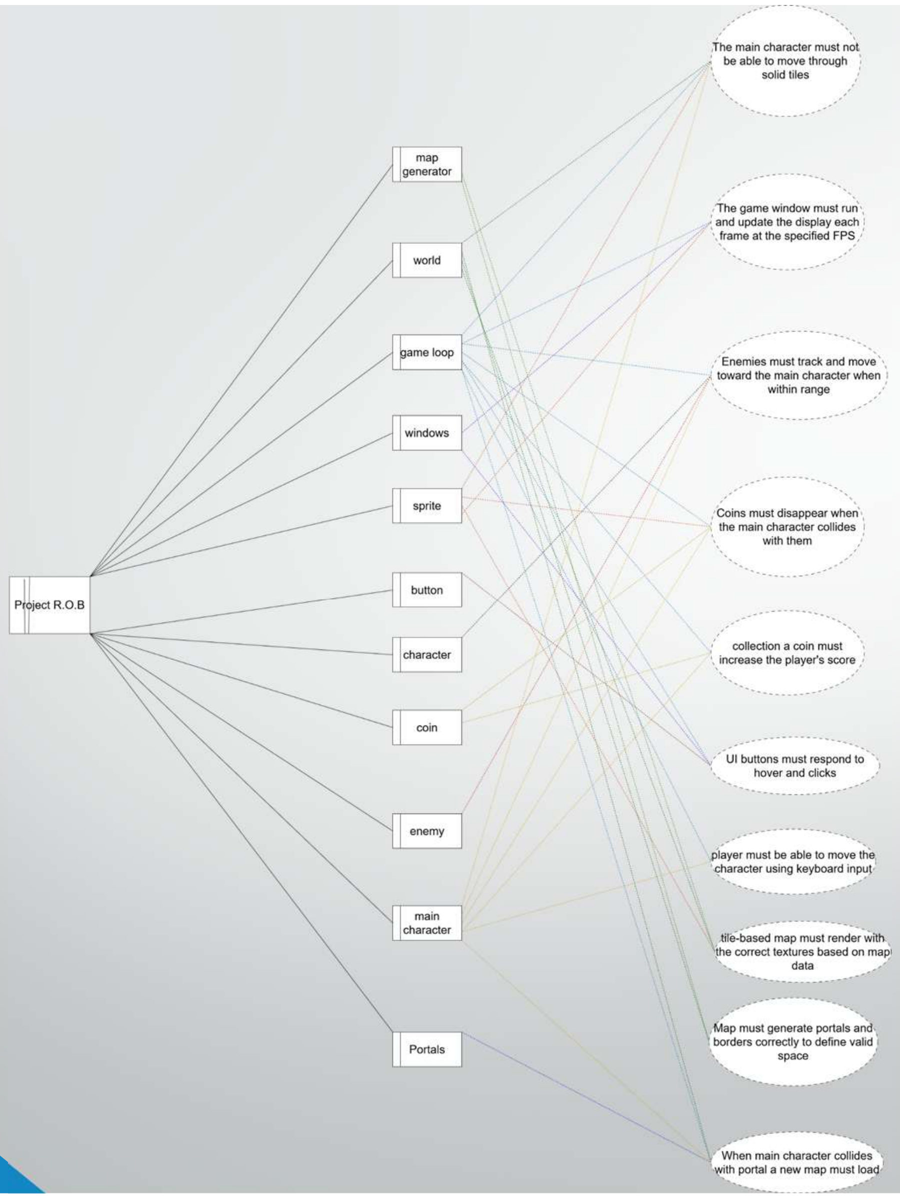
Level 0 Data Flow Diagram



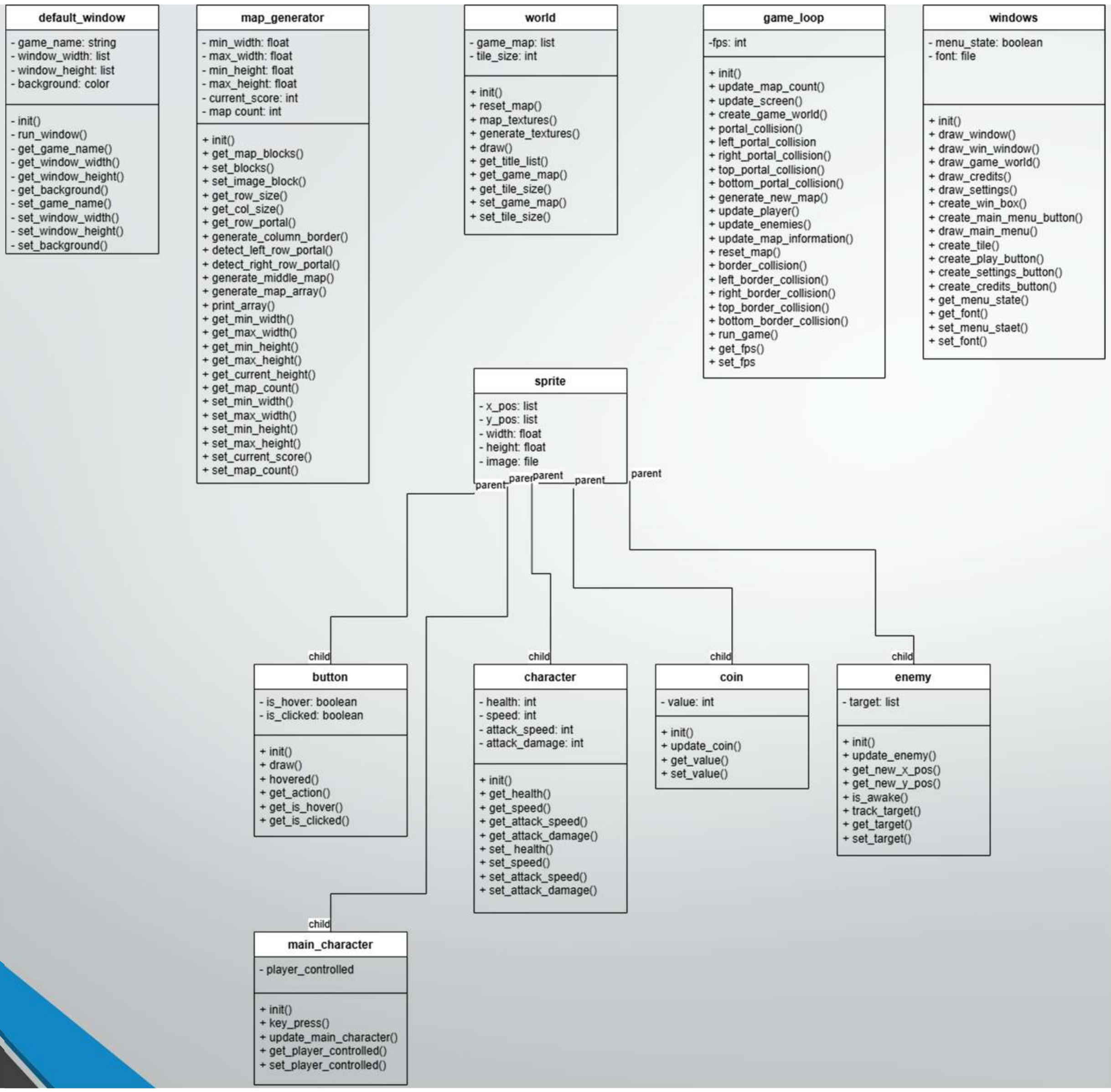
Level 1 Data Flow Diagram



Problem Frames


Project R.O.B
map generator
world
game loop
windows
sprite
button
character
coin
enemy
main character
Portals
The main character must not be able to move through solid tiles
The game window must run and update the display each frame at the specified FPS
Enemies must track and move toward the main character when within range
Coins must disappear when the main character collides with them
collection a coin must increase the player's score
UI buttons must respond to hover and clicks
player must be able to move the character using keyboard input
tile-based map must render with the correct textures based on map data
Map must generate portals and borders correctly to define valid space
When main character collides with portal a new map must load

UML Diagrams



Maintenance

Maintenance Overview

The purpose of performing maintenance on this project is to fix various bugs and performance issues users experience while playing the video game. Additionally, new features and game elements must be added to create interest from seasoned players of the game.

The following types of maintenance will be conducted throughout the software's lifecycle:

- Corrective: fixing any bugs, errors, and glitches that are experienced during gameplay
- Adaptive: allowing the game to remain functional on new, updated, and improved hardware/software environments
- Perfective: constantly improve the performance of the game to maintain an acceptable level of operability
- Preventive: implementing fixes to potential issues, resolving security vulnerabilities, and supporting new computing environments before the become popular enough to cause noticable problems during gameplay

Maintenance Overview cont.

We will conduct regular testing on our programs to ensure they function as intended. These will be completed before the presentation of a new prototype and will be altered if unsuccessful until an adequate level of success has been reached.

Along with any updates/patches to the game, documentation of these changes, processes, and testing will be completed to communicate progress in the game's lifecycle.

Maintenance Timeline

Week 1 of Development

- Identify initial software, hardware, security, and user requirements
- Conduct testing on any developed elements to ensure proper functionality
- Document changes, testing, and all communication to track progress

Week 2 of Development

- Reevaluate the project's requirements, scope, components, storyline, and other implementation elements to suit the needs of the solution to the identified problem
- Conduct testing on any developed elements to ensure proper functionality

Maintenance Timeline

Week 3 of Development

- Finalize initial software, hardware, security, and user requirements
- Conduct final testing on any developed elements to ensure proper functionality

Post-Release

- Note bugs, errors, and glitches (if any) that users encounter during gameplay
- Release updates, security patches, and changes for any identified issues to realign the game with its intended functionality
- Design new elements to add to the game to increase the longevity of seasoned players' interest

Demo Time!