CYBERNUKE

POST-APOCALYPTIC CYBERPUNK ROGUELITE

J. Kenneth Wallace, Vance Brender-A-Brandis

Abstract

CYBERNUKE is a roguelite game developed in the C# WPF framework. You play as an amnesiac cyborg who awakens in the middle of a destroyed futuristic city and must recover their memories. The game combines a post-apocalyptic and cyberpunk setting to create a unique atmosphere not previously explored by other games. The setting of the game is inspired by popular stories such as *Fallout*, *Cyberpunk 2077*, and other game series like *Dark Souls* and *Megami Tensei*.

1. Introduction

For this project we aim to create a video game to gain experience with C#, WPF, and game design. We called the game CYBERNUKE, combining the game's cyberpunk and post-apocalyptic elements into a single title.

The setting of the game is rooted in the cyberpunk genre, taking place in a massive futuristic city where buildings as tall as the Empire State Building are commonplace, gangs are armed with high-tech laser guns, and the wealth gap is at its widest. On a peaceful day in the twenty-first century, the city is attacked by an unknown power, which is followed by a nuclear explosion and the displacement of the entire city into another dimension. The story's cyberpunk portion is inspired by the game *Cyberpunk 2077*, while the post-apocalyptic portion is inspired by *Fallout* and other post-apocalyptic games.

CYBERNUKE's gameplay is influenced by games such as the *Shin Megami* and *Dark Souls* series. Shin Megami inspired the game's turn-based combat, while Dark Souls inspired the overworld map and exploration. Moreover, CYBERNUKE is heavily influenced by the roguelike genre, particularly games like *Slay the Spire*, *The Binding of Isaac*, and *FTL: Faster Than Light*.

We hope that those who enjoy the gameplay and story of the works that inspired CYBERNUKE will enjoy the game we make as well.

1.1. Audience

CYBERNUKE has a diverse setting, story, and gameplay that draws inspiration from a wide range of games and stories. We want to draw in fans by fusing two sizable genres—cyberpunk and post-apocalyptic—and putting it into a roguelite game. Those looking for gameplay should expect turn-based combat and a top-down world filled with exploration and encounters akin to *Pokémon*. Those who come for the story will be treated to a mysterious tale about someone piecing together their memories in a new, hostile cyberpunk hell.

1.2. Background

CYBERNUKE is inspired by roguelike games, but it does not completely follow them, making CYBERNUKE a rogue-*lite*. Roguelite games are a subgenre of roguelike games. The Berlin Interpretation of a "roguelike," developed at the 2008 International Roguelike Development Conference, is the most widely accepted definition. It specifies that all roguelikes must have the following characteristics [3]:

1) Permadeath: the player has one life per game.
2) Random environment generation: no map or encounter is ever the same.
3) Exploration and discovery: the player explores the map and finds items.
4) Turn-based, grid-based, non-modal gameplay: all actions are turn-based, on a grid map, on a single screen that holds all windows and doesn't change.
5) Hack-n-slash: the player defeats a lot of monsters.
6) Resource management: the player manages a limited amount of resources.

Permadeath, random environment generation, and non-modal gameplay are not included in CYBERNUKE. As such, our game is classified as a roguelite because it lacks most of the required features. Specifically, roguelite as used by CYBERNUKE means the player has infinite lives, explores a pre-built map with loot spread out, encounters enemies at fixed rates, fights in turn-based combat, and manages a limited amount of resources.

CYBERNUKE's setting is a hybrid of two genres: cyberpunk and post-apocalyptic. Cyberpunk is defined as a *"science-fiction subgenre characterized by countercultural antiheroes trapped in a dehumanized, high-tech future,"* [1] or a dystopian future full of holograms, lasers, and crime. The post-apocalyptic genre is defined by the collapse of society as a result of a disaster, with a few survivors forming small civilizations. The combination of these two genres serves as the foundation for the rest of the story and gameplay.

1.3. Impacts

If enough effort is put in, CYBERNUKE has the potential to amass a sizable fan base. An optimistic estimate would be that CYBERNUKE reaches a large audience that plays, discusses, and enjoys the game in general. The game would then inspire others to create stories or games in the same hybrid cyberpunk-post-apocalyptic genre. A more realistic estimate would be that the game is simply released and is only played by a small number of people. Expecting too much from such a small game is unrealistic.

1.4. Challenges

Three big challenges will need to be overcome in this project:

1) Inexperience in C# and WPF
2) Clear communication of ideas and implementations
3) Beating the time limit
4) How to design a game

We've both never worked with the WPF framework, let alone C#. As the project progresses, we will have to learn how to work with both. This can be mitigated by reading documentation or watching tutorials.

Clear communication can also be a big issue and can stop work if people aren't on the same page. What makes this worse is that we are both inspired by different games which makes it difficult to communicate ideas of one game that the other person hasn't experienced. The clear solution to a lack of clear communication is using standardized methods of communication like UMLs and Diagrams. UML and Discord will definitely be the work-horses of communication throughout the project.

Another issue is the project's time frame. Having to not only learn but also implement a large number of designs in a short period of time will be stressful, but it should be possible.

The most serious issue is that we have no experience designing games. There is a significant difference between playing and creating a game, which will need to be addressed as we continue to work.

2. Scope

In order of importance, these are the fundamental game mechanics that must be implemented for *CYBERNUKE* to be considered functional:

1. Operational Interfaces: The Operational Interfaces are the menus that the player uses to interact with the game. (Main Menu, Combat Menu, Town Menu, Overworld Menu, Cutscene Menu).
2. Menu Switching: The ability to switch between Menus.
3. Secondary Interfaces: The Secondary Interfaces are other menus that are accessed within other menus (Pause Menu).
4. Enemies: Enemies and Enemy party files to load in enemies and groups of those enemies from.
5. Overworld Maps: The ability to create Overworld maps and render them onto the screen from a file.
6. Character-Map Interaction: The ability to move a player's character while they are on the map and the ability to interact with parts of that map as the player.
7. Enemy Encounters: Enemies are the monsters/humans that the player will encounter throughout the game. Enemy Encounters refers to how you will find monsters in the first place, which is primarily done through a "percent encounter" system akin to the old *Pokemon* games.
8. Combat: Combat refers to fights between the player character and enemies in this game, which will be turn-based. Combat happens inside the Combat Menu.
9. Companions and Party System: Multiple player characters (up to 4) to bring into combat.
10. Basic Item System: A bare-bones item equip system that lets players equip different armor and weapons.

After the fundamental game mechanics are in place, additional features that flesh out the game can be added. Possible stretch-goal features:

1. Items, Loot, Weapons, Armor: A fully fleshed out equipment and item system.
2. Currency and Merchants: A money system with merchants you can find in Towns and buy items from.
3. Leveling System: Leveling system for players and enemies that scale in difficulty the higher the level you are.
4. More Enemy Types: Different types of unique enemies.
5. Saving and Loading: The ability to save and load the game, with an autosave for when you need to quit in the overworld and a proper save-load in the towns.
6. Hacking Puzzles: In-game puzzles that allow you to access certain areas of the map.
7. Story Content: A story tied to the game.

2.1. Requirements

The required mechanics for the game to function at a basic level are defined in the first part of the scope. These mechanics are required as they are the foundation of the game's most basic actions such as traversing menus, moving around the map, viewing your character's stats, and fighting enemies.

The non-required mechanics for the game are defined in the second part of the scope. These are taken from other finished roguelite games and are not required for the game to function but for fleshing the game out into a more appealing state for the audience. They build upon the foundation laid by the functional mechanics to create something more than a roguelite mockup.

2.1.1. Functional.

- User must be able to navigate the main menu.
- User must be able to interact with the overworld menu.
- User must be able to examine the stats of their character(s).
- User must be able to interact with the combat menu.
- User must be able to interact with the town menu.

2.1.2. Non-Functional.

- User must be able to manage their inventory (Equip armor & weapons, use items).
- User must be able to shop with merchants in towns.
- User must be able to upgrade their base stats when leveling up.
- User must be able to save and load in the town menu.
- Game must be able to autosave when leaving while user is in overworld menu.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Main Menu: Start Demo	Player	Low	1
2	Main Menu: Change Options	Player	Med	1
3	Main Menu: Exit Game	Player	Low	1
4	Overworld Menu: Move Player	Player	Med	1
5	Overworld Menu: Toggle Pause Menu	Player	Low	1
6	Overworld Menu: Toggle Map	Player	Low	1
7	Overworld Menu: Interact	Player	Med	1
8	Pause Menu: Equip Item	Player	Med	3
9	Pause Menu: View Map	Player	Low	3
10	Pause Menu: Close Menu	Player	Low	1
11	Pause Menu: Exit Game	Player	Low	1
12	Combat Menu: Attack Enemy	Player	Med	1
13	Combat Menu: Wait	Player	Low	1
14	Combat Menu: Flee	Player	Low	1
15	Town Menu: Interact with NPC	Player	Low	1
16	Town Menu: Exit Town	Player	Low	1

TABLE I. CYBERNUKE USE CASE TABLE

2.2. Use Cases

Use Case Number: 1
Use Case Name: Start Demo
Description: Player is in the main menu and wants to start the demo.

1) Player is in the main menu.
2) Player clicks on “Start Demo”.
3) The main menu is closed and the overworld menu is opened.

Termination Outcome: The demo is started for the player.

Use Case Number: 2
Use Case Name: Change Options
Description: Player is in the main menu and wants to change the screen resolution.

1) Player is in the main menu.
2) Player clicks on “Options”.
3) Player clicks on Left or Right arrows under “Resolution” to change resolution.
4) Player clicks on “Fullscreen” to toggle fullscreen mode.

Termination Outcome: Player has changed the screen resolution or left it alone.

Use Case Number: 3
Use Case Name: Exit Game
Description: Player is in the main menu and wants to exit the game.

1) Player is in the main menu.
2) Player clicks “Exit Game”.
3) Player clicks “Yes” when prompted to exit the game.

Termination Outcome: The game is exited.

Use Case Number: 4
Use Case Name: Move Player
Description: Player is in the overworld menu and wants to move around the map.

1) Player is in the overworld menu.
2) The player can press or click “W” to move up.
3) The player can press or click “A” to move left.
4) The player can press or click “X” to move down.
5) The player can press or click “D” to move right.

Termination Outcome: The player moves on the overworld menu’s map.

Use Case Number: 5
Use Case Name: Toggle Pause Menu
Description: Player is in the overworld menu and wants to open the pause menu.

1) Player is within the overworld menu.
2) The player can press “Q” to toggle the pause menu.

Termination Outcome: The pause menu is toggled open or closed.

Use Case Number: 6
Use Case Name: Toggle Map
Description: Player is in the overworld menu and wants to open the map.
1) Player is in the overworld menu.
2) The player can press “C” to toggle the map that shows the layout of the current map.
Termination Outcome: The map is toggled open or closed.

Use Case Number: 7
Use Case Name: Interact
Description: Player is in the overworld menu and wants to interact with an element on the map.
1) Player is in the overworld menu.
2) The player can press “S” while on top of an element to interact with it.
Termination Outcome: The object the player is standing on is interacted with.

Use Case Number: 8
Use Case Name: Equip Item
Description: Player is in the pause menu and wants to change their character’s equipment.
1) Player is in the pause menu.
2) Player selects the “Character” sidebar option on the left side of the pause menu.
3) Player selects the character they want to change the equipment of.
4) Player clicks the “Armor” or “Weapon” drop-down box on the lower-half of the screen.
5) Player can freely switch out equipment they have.
Termination Outcome: The equipment of a character or characters are changed.

Use Case Number: 9
Use Case Name: View Map
Description: Player is in the pause menu and wants to view the map.
1) Player is in the pause menu.
2) Player can click the “Map” sidebar option on the left side of the pause menu.
Termination Outcome: The map showing the current layout of the map is displayed in the pause menu.

Use Case Number: 10
Use Case Name: Close Menu
Description: Player is in the pause menu and wants to close it.
1) Player is in the pause menu.
2) Player can click the “Close Menu” sidebar option on the left side of the pause menu.
3) Alternatively, the player can also press “Q“ to toggle the pause menu.
Termination Outcome: The pause menu is closed.

Use Case Number: 11
Use Case Name: Exit Game
Description: Player is in the pause menu and wants to exit the game.
1) Player is in the pause menu.
2) Player can click the “Exit Game” sidebar option on the left side of the pause menu.
Termination Outcome: The entire game exits.

Use Case Number: 12
Use Case Name: Attack Enemy
Description: Player is in the combat menu and wants to attack an enemy.
1) Player is in the combat menu.
2) It must be one of the player character’s turns.
3) The player can select “Attack” on the control panel.
4) The player can select one of the enemies to attack.
Termination Outcome: The selected enemy is attacked by the current player character.

Use Case Number: 13
Use Case Name: Wait
Description: Player is in the combat menu and wants to skip their turn.
1) Player is in the combat menu.
2) It must be one of the player character’s turns.
3) The player can select “Wait” on the control panel.
Termination Outcome: The player character’s turn is skipped.

Use Case Number: 14
Use Case Name: Flee
Description: Player is in the combat menu and wants to flee the fight.
1) Player is in the combat menu.
2) It must be one of the player character's turns.
3) The player can select "Escape" on the control panel.
Termination Outcome: The player flees the fight.

Use Case Number: 15
Use Case Name: Interact with NPC
Description: Player is in the town menu and wants to interact with an NPC.
1) Player is in the town menu.
2) The player can select one of the NPC options in the town menu.
Termination Outcome: The player interacts with the NPC.

Use Case Number: 16
Use Case Name: Exit Town
Description: Player is in the town menu and wants to leave the town.
1) Player is in the town menu.
2) The player can select "Exit Town" option in the town menu.
Termination Outcome: The player leaves the town and returns to the overworld menu.

2.3. Interface Mockups

The following is a mock-up of CYBERNUKE's main menu screen [1].

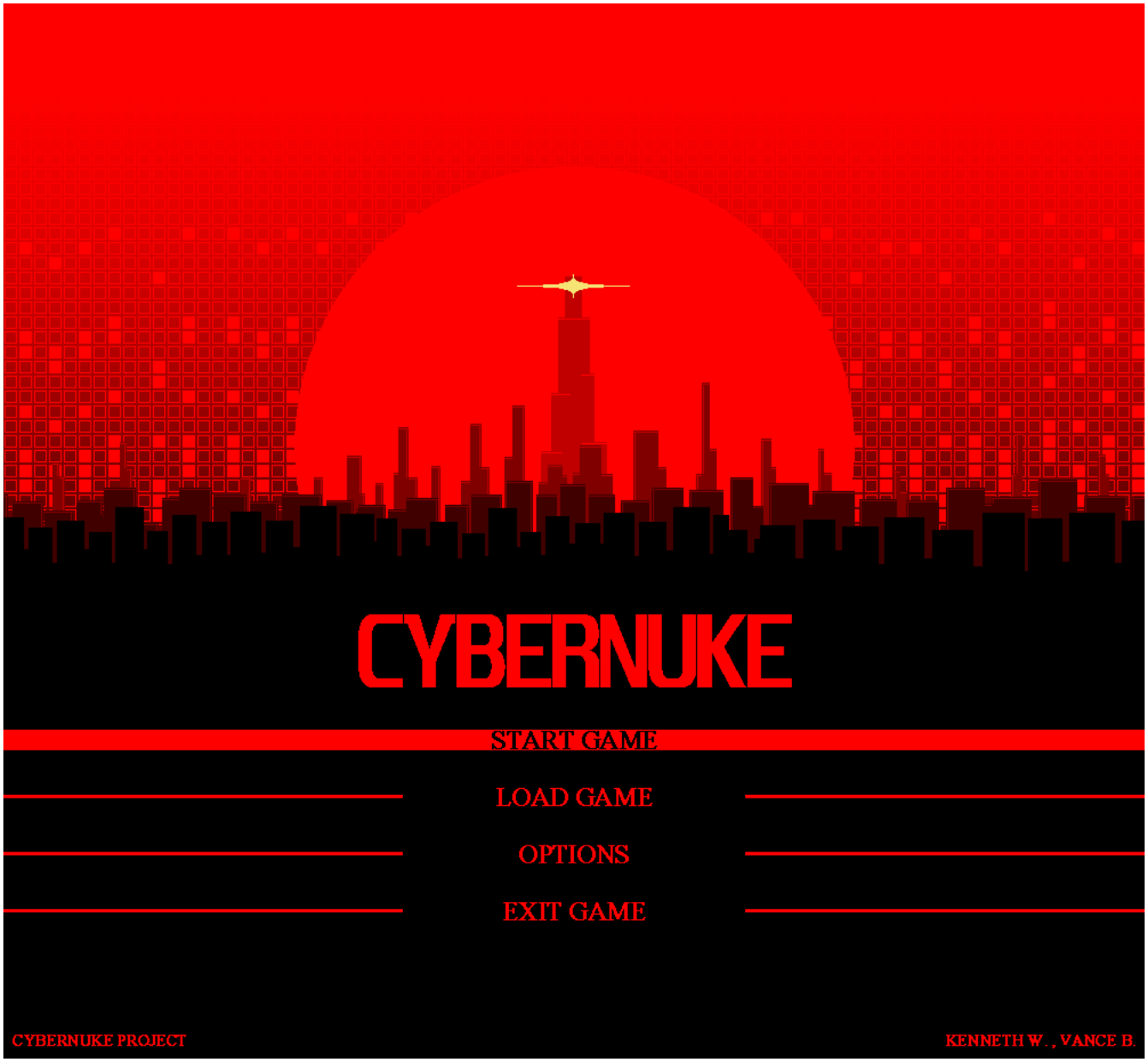

Figure 1. CYBERNUKE Main Menu Screen Mockup. Where "Start Game" would start a new game for the player, "Load Game" would allow the player to load a previously saved game, "Options" would allow the player to change options like screen resolution and volume, and "Exit Game" would let the player exit the game.

The following is a mock-up of CYBERNUKE's combat screen.

Figure 2. CYBERNUKE Combat Screen Mockup. Where "Attack" lets the player attack an enemy, "Skill" lets the player use a skill, "Item" lets the player use an item from the inventory, "Analyze" lets the player view the stats of a friendly or enemy, "Guard" lets the player raise their defense and end their turn, "Wait" lets the player skip their turn, "Escape" lets the player flee the battle, and "Auto" repeats the character's last action.

3. Project Timeline

1. Requirements

Feb. 4 Proposal Draft: The first requirements we identified were based on a set of games we wanted to combine. This included features such as:

1) Operational Interfaces (Main Menu, Option Menu)
2) Maps
3) User Interface (Map Screen, Player HUD)
4) Character Movement, Interaction
5) Character Interface (Name, Stats)
6) Enemies, Enemy Encounters
7) Combat, Combat Interface
8) Player Saves & Load Menu

Along with stretch-goals such as:

1) Companions and Party System
2) Hacking Puzzles
3) More Enemy Types
4) More Story Content
5) Items, Loot, Weapons, Armor, etc.
6) Cybernetic Augmentation
7) Leveling System
8) Currency and Merchants
9) Expanded Character Creation (Species, Backstory, etc.)

Feb. 28 Proposal Update: No changes in requirements from the proposed draft.

Apr. 10 Project Update: Implementation was started and we changed the requirements to better the fit MVVM design pattern we had adopted for CYBERNUKE.

Updated requirements:

1) Operational Interfaces (Main Menu, Combat Menu, Town Menu, etc.)
2) Menu-switching Functionality
3) Secondary Interfaces (Character Menu, Inventory Menu)
4) Enemies
5) Overworld Maps and Map Rendering
6) Overworld Map Functionality (Character Movement, Interaction, Enemy Encounters)
7) Combat Implementation
8) Player Saves & Load Menu

The stretch-goals were not updated.

Apr. 21 Penultimate Update: Due to time constraints we removed unnecessary features and made them stretch-goals.

Updated requirements:

1) Operational Interfaces (Main Menu, Combat Menu, Town Menu, Overworld Menu, Cutscene Menu)
2) Menu Switching
3) Secondary Interfaces (Pause Menu)
4) Enemies & Enemy Party & Enemy Data Files
5) Overworld Maps
6) Character-Map Interaction
7) Enemy Encounters
8) Combat
9) Companions and Party System
10) Basic Item System

Updated stretch-goals:

1) Items, Loot, Weapons, Armor
2) Currency and Merchants
3) Leveling System
4) More Enemy Types

5) Saving and Loading
6) Hacking Puzzles
7) Story Content

Apr. 28 Final Presentation: Not reached

2. Design

Feb. 4 Proposal Draft: Design drafts were made on paper.

Feb. 28 Proposal Update: UML design was started.

Apr. 10 Project Update: UML was re-factored for the new MVVM design pattern being used.

Apr. 21 Penultimate Update: UML was worked on to accomodate new implementations.

Apr. 28 Final Presentation: Not reached
This stage resulted in the creation of the first UML diagram for CYBERNUKE.

3. Implementation

Feb. 4 Proposal Draft: No implementation done.

Feb. 28 Proposal Update: No implementation done.

Apr. 10 Project Update: By this point we had the main menu 90% done, it was missing functionality that let it switch between windows on its own. The combat menu was bare-bones and non-functional. The town, overworld, and cutscene menus were un-implemented but their framework was ready.

Apr. 21 Penultimate Update: The combat menu, overworld menu, and town menu were being worked on with the overworld menu being the closest to finished.

Apr. 28 Final Presentation: Not reached

4. Testing

Feb. 4 Proposal Draft: No testing done.

Feb. 28 Proposal Update: No testing done.

Apr. 10 Project Update: Main menu functionality was verified.

Apr. 21 Penultimate Update: Overworld functionality was verified. Testing began on combat and town menus.

Apr. 28 Final Presentation: Not reached

5. Maintenance

OMITTED.

4. Project Structure

The basis of Cybernuke's structure is built on top of the MVVM (Model-View-ViewModel) pattern, a variant of the Observer design pattern. MVVM allows us to build Windows, referred to as "Views", that contain the functionality of a menu. For instance, the MainMenuView is graphically the main menu of the game, and the CombatView is graphically the GUI and functions used in facilitating combat between the player and enemies. The main code and functionality for the views is located in each view's xaml.cs file. Each View will have functionality that allows said view to navigate to other views: for example, the CombatView will be able to navigate to the OverworldView, and vice versa upon finishing and starting combat.

- The MainMenuView, (which is opened as the initial menu available to the player upon launching the game) it connects to the CutsceneView and OverworldView when starting the game.

- The CombatView is connected to the OverworldView which randomly decides enemy encounters using a randomized percent-based system where each step you take has a random chance to start an enemy encounter depending on the values set in the map's file. Inside the CombatView, data is pulled from the MainWindow which stores all of the persistent data such as character information, what map you're on, etc. It then stores that information locally within the CombatView and uses it throughout the battle.
- For the OverworldView, this will serve as the player's method for moving throughout the game's world and exploring locations and places of interest. This is done through use of a grid-like movement system where the player can move one tile each action to cover distance. During each movement step, chances of a random encounter will be factored depending on the location the player is in (for instance, the player will not have random encounters in a safe-designated area while having lots of encounters in "dungeons").
- From the OverworldView, the player can open the Pause Menu, which functions to allow the player to examine character info, and the more general map (of the current map's layout), as well as closing the Pause Menu and exiting the game.
- For the TownView (accessed upon reaching a town in the OverworldView), the user will use a sequence of buttons to access features the town holds. For example, to visit certain NPCs, clickable buttons will be used to do that. To visit shops and leave the town, buttons will be used for that as well. The ability to save and load game is available here as well.
- Finally, the CutsceneView relies on data files to display images and text dialogue.

4.1. UML Outline

This is the entire UML for CYBERNUKE so far:

Figure 3. Entire UML for the CYBERNUKE project, divided into sections depending on the folder the actual code file is in

As listed in the caption, each section of the UML is sorted based on the folder and function the classes serve. As is evident, all of the Views for CYBERNUKE are contained in the [View] section, the User Controls in the [UserControls] section, and so on.

- First, in the top-left, the Model-View-ViewModel Manager (MVVM) section. This portion of the UML holds the classes used to handle switching from view to view. The NavigationService inherits from the

INavigationService interface and from the ObservableObject class, and is primarily used to manage the current view of the program as well as navigating to other views as required. The ViewModel class, though without implementation inside the classes, are registered within the program on startup and link View to View. The RelayCommand class is used to program the navigation between the Views, but only after the Singletons from the ViewModels are registered.

- Next, in the bottom-left, the View section holds all the Views (and the Main Window) used in CYBERNUKE. Each View correlates to a different part of CYBERNUKE's gameplay elements.
    1) The MainMenuView correlates to the Main Menu that is loaded upon starting CYBERNUKE and holds its functionality: Starting the game, loading a game, configuring options, and leaving.
    2) The OverworldView correlates to the game world and handles rendering the player's movement, the player interacting with objects, and random encounters that occur as the player moves about the world.
    3) From there, the CombatView manages combat: counting the number of players (the player and their companions) and the number of enemies the players are facing. It is also responsible for handling actions the player takes throughout combat, like attacking, guarding, fleeing, and so forth.
    4) The CharacterView is essentially the "Pause Menu" of CYBERNUKE: you can manage the equipment of the main character and his companions in this view, see a map of the overworld (currently the mockup we drew at the beginning), edit options, and exit the game. And, of course, return back to the overworld (where this view is accessed from).
    5) Behind all of these Views is the MainWindow, which provides easy access to relevant data used as the player plays. For instance, the list of characters in the player's party, the equipment the player has in their inventory, and even the list of maps used is always accessible from every view through use of the MainWindow.
- Further, in the bottom-right, the Model section holds all of the classes used for gameplay and map loading (at present). The "Character" class manages all the variables relevant to every Character object like health (HP), skill points (SP), and their various stats and equipped items. Similarly, the Skill and Item classes holds the variables relevant for their aspects of gameplay: names, costs, descriptions, etc. And, finally, the "Map" class is used to manage the maps to-be-featured in CYBERNUKE and manages the encounter chances, maps to be loaded in, spawn locations, and more. The LocationData class is part of the Map class in that it tracks the locations of various elements on the map: objects (ObjectData) and NPCs (NPCData). This will soon be expanded to have towns to enter, treasure/ interesting events, and more.
- For the last section in the top-center, the UserControls section contains classes which inherit from UserControl and are used in various aspects of CYBERNUKE. For now, the EnemyBox class is used in the CombatView to display enemies' health, skill points, and other stats and such in a visual format. The TurnOrderBox class is much the same, a visual element used in the TurnOrder aspect of combat.

4.2. Design Patterns Used

1. Prototype Design Pattern

We used a Prototype for the enemies in CYBERNUKE. When the Combat Menu is opened it will be given a data file that contains which enemies to spawn. It will then take the data files for those enemies and instantiate an Enemy UserControl Object to display on the Combat Menu.

In the UML in the above subsection, this is seen in the CombatView class and EnemyBox UserControl. In CombatView's constructor, the data for the enemy party is read in using the StreamReader input and then passed to the AddEnemy() function, which creates an EnemyBox object for each enemy and adds it to the Combat Menu's enemy GUI.

2. Decorator Design Pattern

We used a Decorator for the Pause Menu in CYBERNUKE. When opening the Pause Menu it unhides a Content Presenter that is present on the MainWindow. This Content Presenter holds the Pause Menu and renders over the underlying window so that the user cannot accidentally activate anything while paused.

In the UML, the Pause Menu refers to the CharacterView (dubbed as such before becoming known as the Pause Menu). In the constructor for the CharacterView, as stated, it unhides a Content Presenter on the Main Window to prevents the player from taking actions with the Pause Menu open.

3. MVVM (Model-View-ViewModel) Design Pattern

We used the MVVM design pattern for CYBERNUKE. We chose to use it as it would allow the program to seamlessly switch between menus without breaking the user's immersion in the game. It also allows us to store data in MainWindow as it stays open persistently.

This is seen very clearly in the UML as the sections are split mainly into the MVVM manager, the different Views used in CYBERNUKE's coding, as well as the Models and ViewModels associated with them.

5. Results

With the amount of time and experience we had starting off, we were able to get a lot implemented successfully:

- Operational interfaces. We were able to get a functioning main menu, combat menu, overworld menu, and town menu all working.
- Menu switching. We were able to implement buttons that allowed players to switch between the different in-game menus.
- Pause Menu. We got the pause menu working, only in the overworld menu though.
- Map Files. Maps are loaded in from .txt files stored in the game's files.
- Enemies, Enemy parties, and Enemy data. We were able to get enemies to load in from files and then load them into combat through enemy party files.
- Enemy Encounters in the overworld. We were able to implement enemy encounters when in the overworld through the map's data file.
- Combat menu. We got the combat menu to work where players and enemies can attack each other.
- Companion and party system. We got the game to load in multiple player characters in the overworld and combat menu.
- Basic item system. We got a basic item system with armor and weapons able to be equipped by players.

We over-estimated the amount of work we would be able to complete in the time we had. This led to a lot of planned features being scrapped and moved to the "Stretch-goals" section of the scope. A list of features we had to remove:

- Player and Enemy physical and elemental resistances. This feature required too much in terms of programming and our code wasn't able to handle it.
- Music and Sounds. Our lack of knowledge in threading led to this being scrapped after numerous attempts at implementing it.
- Fleshed out armor and weapon system. We originally had slots for 3 armor pieces and a melee + ranged weapon. This had to be cut down to a single Armor and Weapon slot in each character's inventory.
- Inventory system. This would've included multiple different item types such as consumbles, armors, weapons, quest items, etc. It had to be scrapped due to time constraints.
- Save/Load system. While possible, we were limited by the amount of time we had which prevented implementation.
- Game story. We weren't able to get all fundamental features down so a Story was out of question with the time we had left.
- Merchants and currency. Time constraints.
- Hacking puzzles. Time constraints.
- More enemy types. Time constraints.

5.1. Future Work

Overall this project was a good learning experience in many ways. We were able to learn a lot about WPF, game design, UI design, and more. CYBERNUKE will probably end up on our resumes and never see the light of day ever again.

References

[1] Britannica, T. Editors of Encyclopaedia. *"cyberpunk."* Encyclopedia Britannica, 1999. https://www.britannica.com/art/cyberpunk.

[2] Wiktionary. *"roguelite"* https://en.wiktionary.org/wiki/rogue-lite

[3] whatNerd, Joel Lee. *"What's a Roguelike vs. Roguelite? Is the Difference Really That Important?"* whatNerd, 2021. https://whatnerd.com/what-is-a-roguelike-roguelite-difference/.