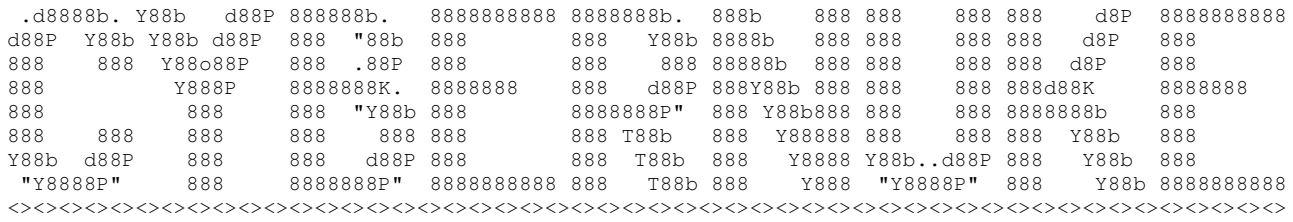


CYBERNUKE

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 Burj Khalifa 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 (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

1. The Operational Interfaces are the menus that are used to interact with the game when the player is not playing the game (Main Menu, Option Menu).
2. The maps are pre-made rather than generated automatically; in this case, it will only be a simple testing map until later.
3. The User Interface includes the Map Screen, which renders the current map to the screen. The Player HUD displays the player's statistics such as Hit Points, Money, and so on.
4. Character Movement refers to moving around the map with the player character. Character Interaction refers to how the player interacts with the map, such as unlocking locked doors or moving between maps.
5. Character Interface differs from User Interface in that it is a separate menu that displays the character's full stats such as HP, Strength, Luck, Agility, and so on, as well as their name.
6. 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. There may also be opportunities for pre-scripted encounters.
7. Combat refers to fights between the player character and enemies in this game, which will be turn-based. Actual combat is made possible by the Combat Interface.
8. Players can save their state (at specific points in the game) and load back in at a later time using Player Saves and the Load Menu.

After the fundamental game mechanics are in place, additional features that flesh out the game can be added.
Possible stretch-goal features (In no particular order):

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.)

### 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 mechanics are not required for the game to function, but are for fleshing the game out into a more playable state. They build upon the foundation laid by the functional mechanics to create something more than a roguelite mockup.

#### 2.1.1. Functional.

- User needs to be able to start a new game as well as load a previous game if one exists.
- Additionally, User must be able to save their game while playing
- User must be able to move around the game world - both on normal map and overworld map
- User must be able to participate in combat by performing actions like attacking, guarding, using items, etc.
- User must be able to interact with NPCs, like for shopping or for progressing the story

- User must be able to manage inventory - dropping or equipping/using items, using quest items, etc.
- User must be able to see and examine their character's stats - each stat should have a description of what it affects
- User must be able to manage companions - change order, dismiss or recruit, check equipment and skills of, etc.

#### 2.1.2. Non-Functional.

- Autosaves - User should have automatic saving in case of crashing, failing battles, and general convenience
- Simple Combat AI - Enemies should be able to attack, use skills, etc. during battle through use of algorithms or basic attack sequences
- Shop Stock - Basic inventory stocked so User can buy various items at different times (can reset shop every nth battle if we want)
- Map autoscrolling - Camera should center on User and map should scroll as User moves around
- I didn't know what else to put here.

### 2.2. Use Cases

This subsection is arguably part of how you define your project scope (why it is in the Scope section...). In a traditional Waterfall approach, as part of your requirements gathering phase (what does the product actually *need* to do?), you will typically sit down with a user to develop use cases.

You should have a table listing all use cases discussed in the document, the ID is just the order it is listed in, the name should be indicative of what should happen, the primary actor is typically most important in an application where you may have different levels of users (think admin vs normal user), complexity is a best-guess on your part as to how hard it should be. A lower number in priority indicates that it needs to happen sooner rather than later. A sample table, or Use Case Index can be seen in Table **??**.

Use Case Number: 1
Use Case Name: Add item to cart
Description: A shopper on our site has identified an item they wish to buy. They will click on a "Add to Cart" button. This will kick off a process to add one instance of the item to their cart.

You will then go on to (minimally) discuss a basic flow for the process:

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, this also updates the current total.

Termination Outcome: The user now has a single instance of the item in their cart.

You may need to also add in any alternative flows:

Alternative: Item already exists in the cart

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, showing that one more instance of the existing item has been added. This also updates the current total.

Termination Outcome: The user now has multiple instances of the item in their cart.

You will often also need to include pictures or diagrams. It is quite common to see use-case diagrams in such write-ups. To properly reference an image, you will need to use the `figure` environment and will need to reference it in your text (via the `ref` command) (see Figure 1). NOTE: this is not a use case diagram, but a capybara.

After fully describing a use case, it is time to move on to the next use case:

Use Case Number: 2
Use Case Name: Checkout
Description: A shopper on our site has finished shopping. They will click on a "Checkout" button. This will kick off a process to calculate cart total, any taxes, shipping rates, and collect payment from the shopper.

You will then need to continue to flesh out all use cases you have identified for your project.

### 2.3. Interface Mockups

At first, this will largely be completely made up, as you get further along in your project, and closer to a final product, this will typically become simple screenshots of your running application.

In this subsection, you will be showing what the screen should look like as the user moves through various use cases (make sure to tie the interface mockups back to the specific use cases they illustrate).


Figure 1. CYBERNUKE Combat Screen Mockup

## 3. Project Timeline

Go back to your notes and look up a typical project development life cycle for the Waterfall approach. How will you follow this life cycle over the remainder of this semester? This will usually involve a chart showing your proposed timeline, with specific milestones plotted out. Make sure you have deliverable dates from the course schedule listed, with a plan to meet them (NOTE: these are generally optimistic deadlines).

## 4. Project Structure

At first, this will be a little empty (it will need to be filled in by the time you turn in your final report). This is your chance to discuss all of your design decisions (consider this the README's big brother).

### 4.1. UML Outline

Show the full structure of your program. Make sure to keep on updating this section as your project evolves (you often start out with one plan, but end up modifying things as you move along). As a note, while Dia fails miserably at generating pdfs (probably my fault), I have had much success with png files. Make sure to wrap your images in a `figure` environment, and to reference with the `ref` command. For example, see Figure 2.

### 4.2. Design Patterns Used

Make sure to actually use at least 2 design patterns from this class. This is not normally part of such documentation, but largely just specific to this class – I want to see you use the patterns!

## 5. Results

This section will start out a little vague, but it should grow as your project evolves. With each deliverable you hand in, give me a final summary of where your project stands. By the end, this should be a reflective section discussing how many of your original goals you managed to attain/how many desired use cases you implemented/how many extra features you added.

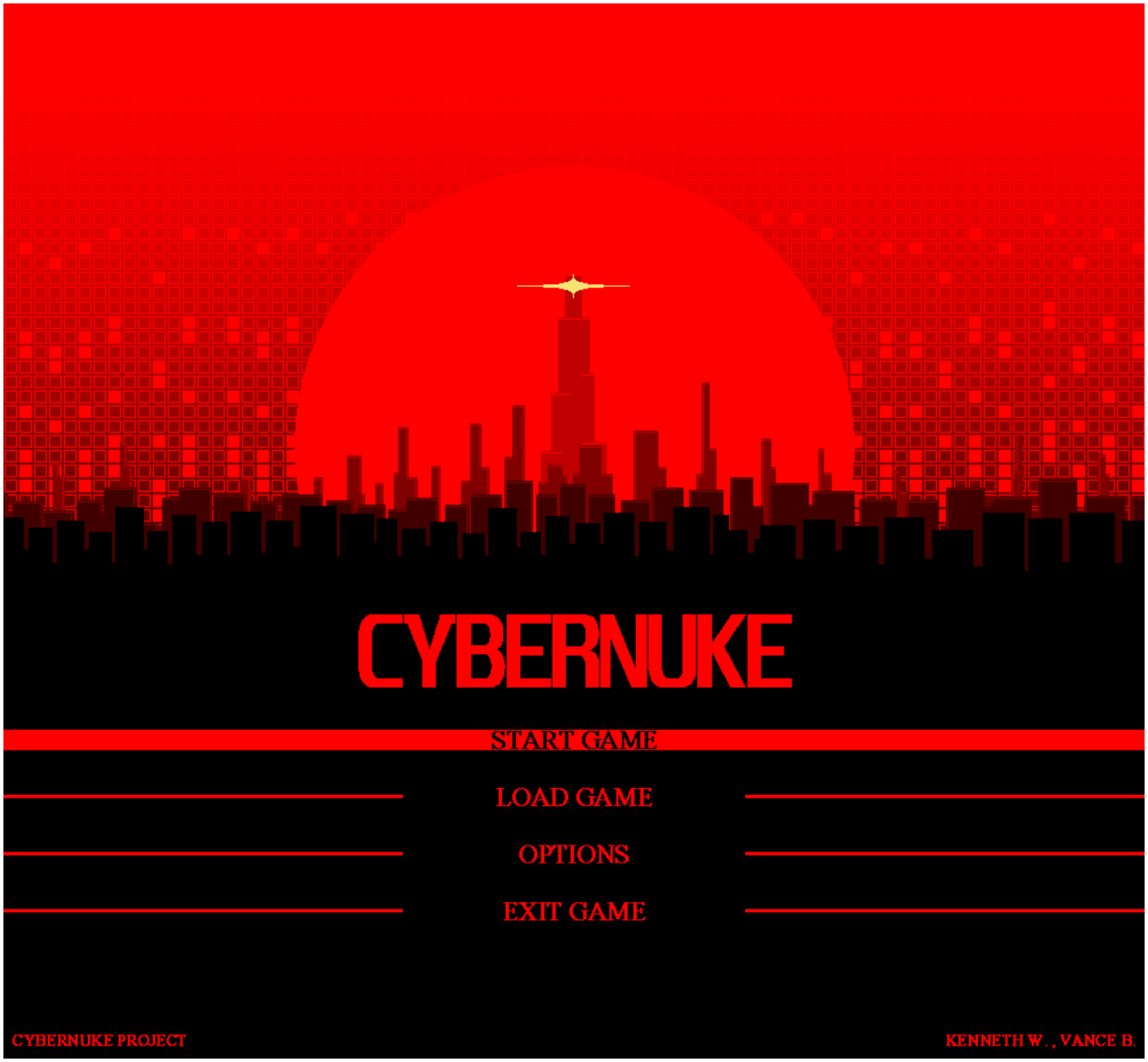


Figure 2. CYBERNUKE Main Menu Screen Mockup

### 5.1. Future Work

Where are you going next with your project? For early deliverables, what are your next steps? (HINT: you will typically want to look back at your timeline and evaluate: did you meet your expected goals? Are you ahead of schedule? Did you decide to shift gears and implement a new feature?) By the end, what do you plan on doing with this project? Will you try to sell it? Set it on fire? Link to it on your resume and forget it exists?

## 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/.