P.A.C.R.L
Post-Apocalyptic Cyberpunk Roguelike

J. Kenneth Wallace, Vance Brenderabrandis

Abstract

PACRL is a roguelike built in the C# WPF framework. It is inspired by stories like *Fallout* and *Cyberpunk 2077*, with gameplay reminiscent of games such as *Cataclysm: Dark Days Ahead*. The dystopian cyberpunk future of PACRL depicts a society that has entered a "New Dark Age" following a brief nuclear exchange. You take on the role of a human who must make every effort to avoid being destroyed by both cyber-humans and the monsters that emerged from the radioactive ashes of the old world.

1. Introduction

For this project we aim to create a roguelike to gain some experience with C#, WPF, and game design. With PACRL, we're trying to showcase our understanding of the Cyberpunk and Post-apocalyptic genres by creating a unique setting that blends both ideas together. Our target audience is primarily those who enjoy roguelikes but would like to experience a new and unique setting. For reference, Cyberpunk is a *"science-fiction subgenre characterized by countercultural antiheroes trapped in a dehumanized, high-tech future"* [1]. Mixed in with this is the post-apocalyptic genre seen in popular games such as *Fallout*, *Wasteland*, and *Metro*. Although broad, the post-apocalyptic genre is defined by a setting where nuclear warfare, or any disaster, has caused modern society to collapse. We believe that a blend of these two settings can create an interesting theme for our game that hasn't been explored by others.

This game's estimated impact is very low. An optimistic prediction would be that our game reaches a larger audience and inspires others to create roguelikes or other games with a similar setting to ours. Other than that we hope to entertain at least one person with our game.

One of the most difficult challenges we face right now is our lack of experience not only with C# and WPF, but also with game design in general. For example, goals such as movement, combat, stats, and so on will all have to be developed and implemented without any prior knowledge of how to do these things. However, inexperience is a treatable disease. We have access to a plethora of online resources that we can study and use to create this project, and we will do so.

1.1. Audience

PACRL's target audience is those who enjoy playing RPGs and roguelikes but are looking for a fresh new setting. We also hope to appeal to fans of the cyberpunk and post-apocalyptic genres who have never played a roguelike before.

1.2. Background

We chose to make a game for our project because it is a genre in which we are both very familiar. A game is also much more enjoyable to create and play than a calculator or calendar app. PACRL is a hybrid of an RPG and a roguelike. RPGs, or role-playing games, are video games in which the player controls a fictional character; while roguelikes are RPG subgenres distinguished by turn-based gameplay, procedurally generated levels, and dungeon exploration.

The setting of PACRL is a blend between two popular genres: Cyberpunk and Post-apocalypse. Cyberpunk is a *"science-fiction subgenre characterized by countercultural antiheroes trapped in a dehumanized, high-tech future"* [1]; and Post-apocalypse is genre characterized by the collapse of society due to a disaster. We believe that we can create a new, unique setting for our game by blending these two genres together.

1.3. Impacts

We believe that the overall impact of this game will be minimal. An optimistic estimate would be that when our game is released, it will reach a larger audience that will play, discuss, and enjoy it. If this is the case, it may inspire others to create games or stories with a setting similar to PACRL.

1.4. Challenges

Going into this, we have a lot of challenges to face. One of the most significant is that we have almost no experience working with C#, WPF, or in game design. If we are to complete PACRL within the time frame, we will need a lot of reference material. We believe it is doable though. Apart from getting used to WPF, there is no ”most difficult” part that we can think of. All of the core mechanics that must be added will be difficult, some more than others; however, we lack the experience to differentiate the difficulty of each implementation.

2. Scope

For the scope of PACRL, we have defined the core mechanics that need to be implemented to consider the game playable: (Not in any particular order)

1) User interface (Heads up display, map rendering)
2) Menus (Main menu, Load menu, Options, etc)
3) Your character (Name, Stats)
4) Movement around map
5) Map generation
6) Enemies
7) Enemy encounters (Similar to *Pokemon*)
8) Combat

These are the features we will be rushing to complete first.

Once all of these are complete the game will be in a playable state and more features can be added on.

Examples of additional features: (stretch goals, no particular order)

1) Companions (Party system)
2) Hacking
3) More enemies
4) Further story/world-building
5) Items, loot, weapons
6) Armor, equipment
7) Cybernetic augmentation
8) Level system
9) Currency and item stores
10) Alternative player species (Monster)
11) Expanded player customization

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Add item to cart	Shopper	Med	1
2	Checkout	Shopper	Med	1

TABLE 1. SAMPLE USE CASE TABLE

2.1. Requirements

As part of fleshing out the scope of your requirements, you'll also need to keep in mind both your functional and non-functional requirements. These should be listed, and explained in detail as necessary. Use this area to explain how you gathered these requirements.

2.1.1. Functional.

- User needs to have a private shopping cart – this cannot be shared between users, and needs to maintain state across subsequent visits to the site
- Users need to have website accounts – this will help track recent purchases, keep shopping cart records, etc.
- You'll need more than 2 of these...

2.1.2. Non-Functional.

- Security – user credentials must be encrypted on disk, users should be able to reset their passwords if forgotten
- you'll typically have fewer non-functional than functional requirements

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 1.

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.

Figure 1. First picture, look at the capybara

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).

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!

Figure 2. Your figures should be in the *figure* environment, and have captions. Should also be of diagrams pertaining to your project, here we are teaching the capybaras to do our work.

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.

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.