

Vapor, A PC Game Library Manager

Steven Gray and Andy Lum

Abstract

In the modern era of PC gaming, its too easy to lose track of games. Many of the biggest games are spread across multiple launchers, while a lot of smaller games find themselves disconnected from all of them. That's where Vapor comes in. Vapor is a minamalist game library manager to help gamers keep track of all of their games across various launchers.

1. Introduction

Increasing popularity of PC gaming has lead to a unique situation where many of the biggest developers want to create their own storefronts and game libraries to maximize the amount of profit they make on a game sale. This becomes a hassle for gamers who simply want to play their games without having to sift through multiple different programs to find where their game is hiding. This issue is further exacerbated by storefronts without their own launchers, or programs managed by storefronts/developers to sell their games, leading to a collection of stray programs on your computer that you download to play one or two games.

Vapor exists to solve this problem. Vapor is a video game library manager, a program that allows a user to manage their library of video games.

1.1. Background

PC games are often stored and distributed in what are known as "launchers", or programs that simply launch the game on their own. This can be a hassle as games can often be distributed between multiple launchers, making library management difficult.

1.2. Impacts

At the very least, our goal is to make game management easier and more flexible for the user. We want to create an environment where someone can avoid wasting time sifting through folders or launchers just to find what game they want to play, or avoid wasting money buying games they already own because they simply can't find what they're looking for.

1.3. Challenges

The immediate challenges this project brings up include determining what the name of a particular game is, finding the launcher that the game is associated with, and figuring out how to work with online databases. In addition, the bulk of this project was implemented with two windows: A Main Window for storing and accessing the games, and a secondary Game window to post the description, last played, last updated, game size, name, and give the user the ability to play the game.

2. Scope

The initial goal for this project is to create a functional interface that allows for a user to have all of their games accessible and playable through a single library manager. This would include:

- A main menu with all of their games available and playable.
- An options menu that would allow a user to add new games or treat certain folders as library folders (alongside an automatically created library folder handled by the program itself).
- The ability to remove a game or library folder from the library manager or delete them altogether
- The ability to refresh the library at any time, to find newly added games or remove recently removed ones.
- The ability to open and browse the location where a game is stored

A few possible stretch goals could be:

- Offline Mode, a way to use the launcher without needing to connect to any databases (with possibly limited functionality)

- Icons that allow for a user to discern where the games are from, such as Steam, Battle.net, or a user defined Library folder.
- A store page that allows a user to search/browse popular storefronts like Steam, GOG, and Humble.
- Allow the user to create and manage groups of games

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 locally stored list of games on the computer - including their locations and any necessary assets such as cover art or icons
- User needs to have cover art and icons for each game
- User needs to be able to launch games from the program

2.1.2. Non-Functional.

- Game information has to be saved somewhere convenient to reference whether online or offline

2.2. Use Cases

Use Case Number: 1

Use Case Name: Add a game to list

Description: A user wants to add a single game to their list. They open up a settings drop-down and click on an "Add Game To Vapor" button. This will then allow the user to select the file they want to add and subsequently adds it to the program

- 1) User opens up the settings menu.
- 2) User left-clicks on "Add Game To Vapor" button.
- 3) User searches for and selects their game.
- 4) The program finds everything needed and adds it to the user's list

Termination Outcome: The game the user selected is added to their list.

Alternative: Game exists in list

- 1) User opens up the settings menu.
- 2) User left-clicks on "Add Game To Vapor" button.
- 3) User searches for and selects their game.

Termination Outcome: A message pops up telling the user that the game is already available in their list, and the screen returns to the main list page.

Use Case Number: 2

Use Case Name: Add a folder of games to list

Description: A user wants to add a folder of games to the list. They open up a settings drop-down and click on an "Add Library Folder To Vapor" button. This will then allow the user to select the file they want to add and subsequently add it to the program.

- 1) User opens up the settings menu.
- 2) User left-clicks on "Add Library Folder To Vapor"
- 3) User searches for and selects the folder they want to add.
- 4) The program searches through the folder for any games it can find and adds them to the list.

Termination Outcome: The folder is added to the program, and the games inside of it are added to the list.

Alternative: The folder has already been added

- 1) User opens up the settings menu.
- 2) User left-clicks on "Add Library Folder To Vapor"
- 3) User searches for and selects the folder they want to add.

Termination Outcome: A message pops up saying the folder already exists in the program and returns to the list screen.

Use Case Number: 3

Use Case Name: Launch a game from Vapor

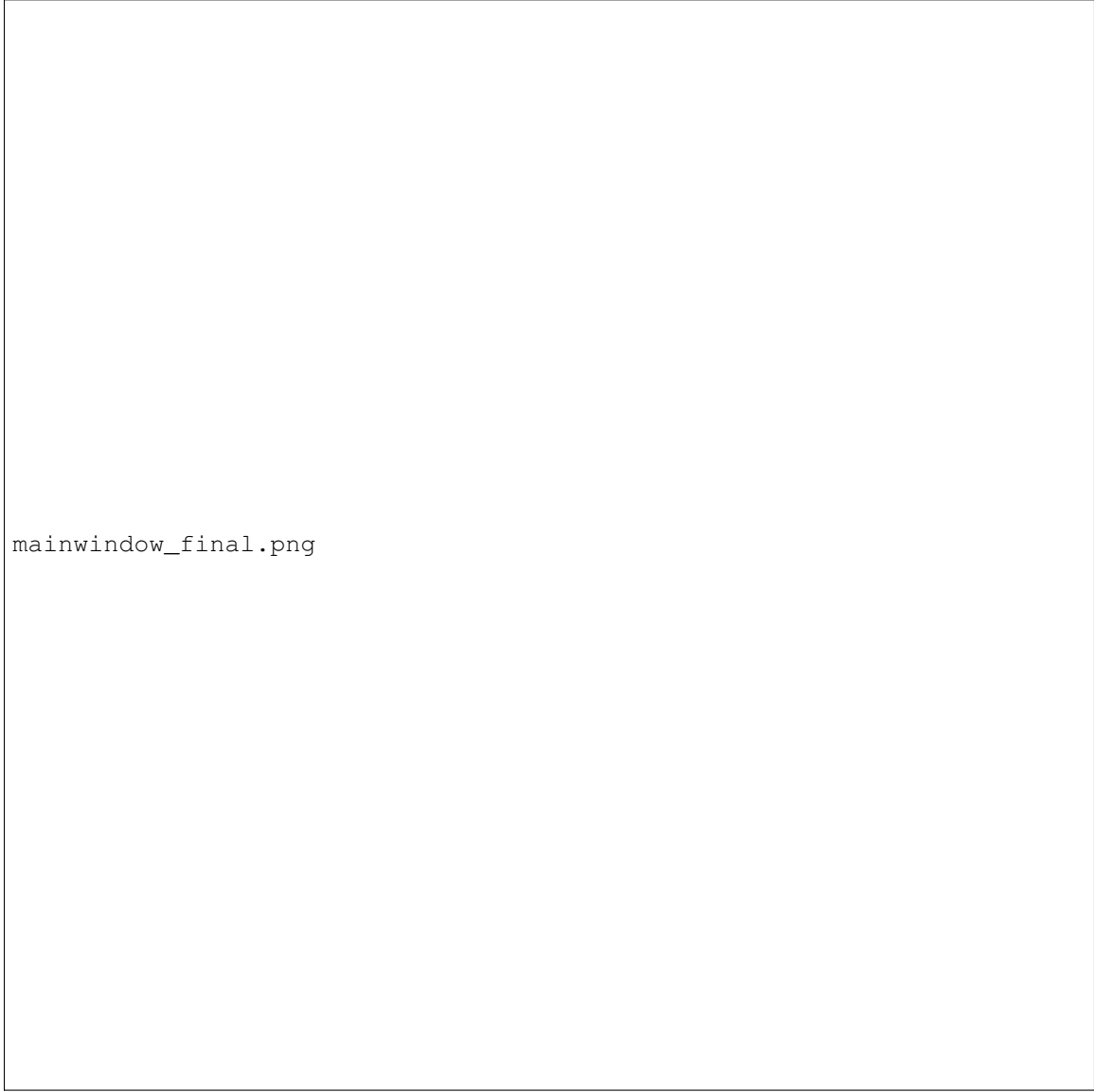
Description: A user wants to play a game. They left click on the game they want to play from their list, and then left click a "Play" button.

1) User left-clicks on the game they want to launch from their list.

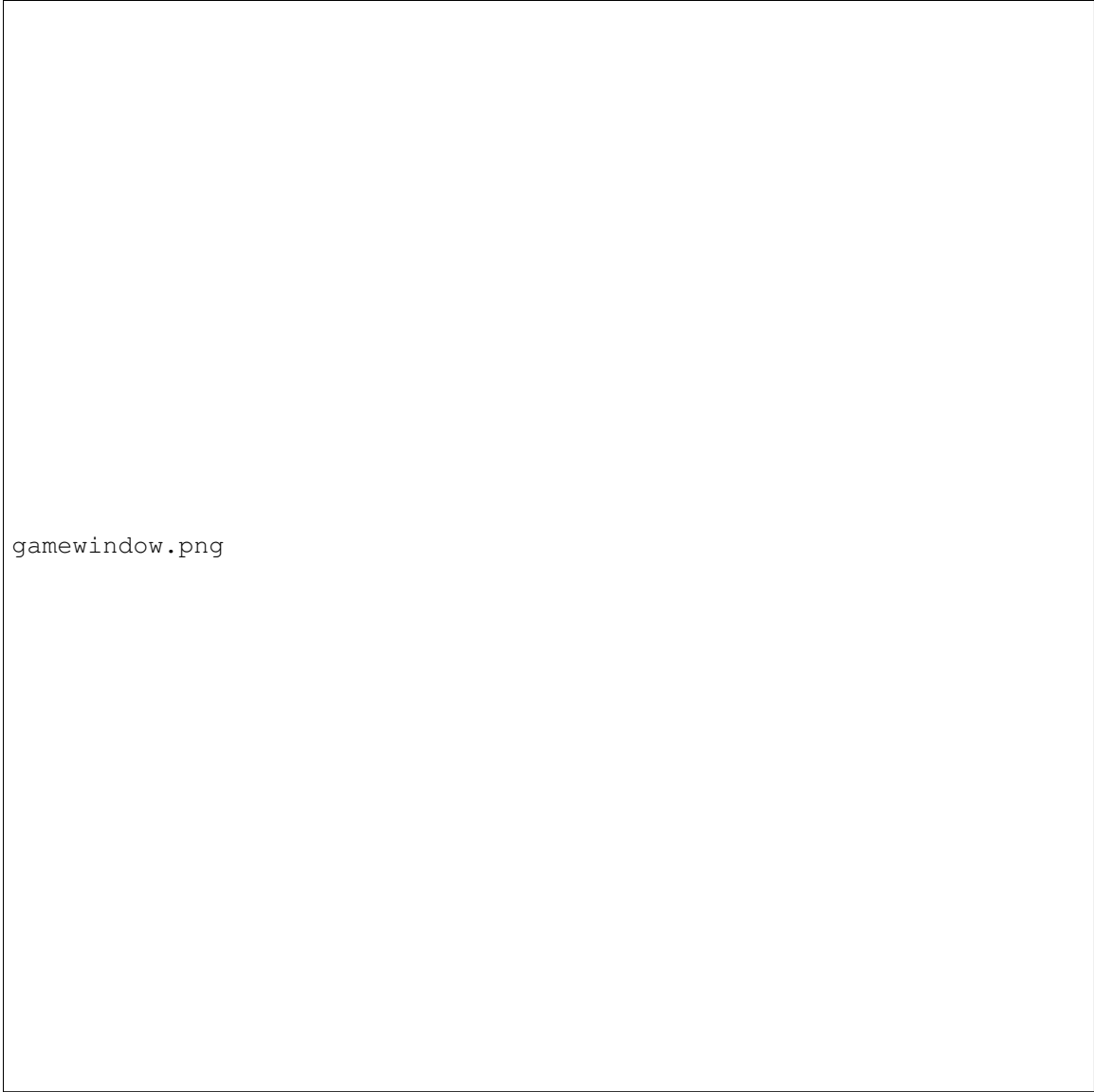
2) User navigates over to a large "Play" button

Termination Outcome: The game is run.

2.3. Interface Mockups



mainwindow_final.png



gamewindow.png

3. Project Timeline

Throughout the course of the project, we followed the Waterfall approach. First, we set requirements, in which we got together and created a proposal draft for our project that talks about background, impacts, challenges of our project, stretch goals, and a brief introduction of our project (September 13). Secondly, we started our design phase where we begin to layout our thoughts on design patterns and created a project structure UML Outline (October 28). Later after our Demo day, we were able to take our feedbacks from the audience to apply it towards updating our diagram structure to wrap up our design phase.

4. Project Structure

In this project, we wanted to make game management easier and more flexible for the user. In addition to that, we wanted to create a better experience, avoid wasting time looking through folders, and avoid wasting money buying a game they already own just because they weren't able to find it in their launcher. There is two windows: a Main for storing and accessing the games, and a secondary Game window to post any additional information present in the game and give the user the ability to play the game.

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. Design Patterns Used

Factory pattern and Observer pattern.

5. Results

For our project, we have created a user-friendly game library manager that allows the user to organize and play their games. The user is able to add and delete games, view their most recently played game, a recommended game, and an unplayed game on the main interface. In a secondary window, which pops up once a game is selected, provides a brief description of that game, the last time it was played and updated, and the size of the game. We had intended to provide cover art of the games, however, we were unable to get this to functionally work.

5.1. Future Work

Throughout the course of this project, we provided weekly updates for the class that set goals for the upcoming week. We also gave a brief overview of what we had accomplished in the previous week, as we looked towards the next. During the duration of the project,