Project Written Report: PlayPal

Levent Ozay, Ivan Kraskov

Monday, April 3, 2023

Introduction

After the COVID-19 pandemic, the number of people who started playing video games skyrocketed and the computer gaming industry grew exponentially. All these people wanted to find a game to spend tens, hundreds, or even thousands of hours on, yet finding a game that clicks with you is easier said than done, and this is where PlayPal comes in. Currently, Steam (a company/platform that has a monopoly on engines for running games) is the most popular gaming platform with over 120 million monthly active users and a catalog of over 50 thousand games (Backlinko, 2021). PlayPal will be working with Steam's database to recommend games based on attributes such as reviews, genres, game descriptions, price, and more. The current Steam algorithm responsible for recommending games is far from perfect and is too vague which is not accurate enough to recommend the perfect game for the individual. The suggestions made by Steam's algorithm are based on a few basic parameters and are often useless for the user as the user can't to filtering based on multiple attributes but just one which is the genre. **PlayPal's main aim is to generate game recommendations for all types of gamers from beginners to experienced ones based on multiple factors. This will be done using decision trees to find all compatible games based on a set of questions/preferences as we're able to discard any unwanted attributes/games easily thanks to the structure of trees**. The reason why we wanted to pursue this idea for our project is that our team is composed of individuals that are passionate about video games, and we want to create something that we would personally use in our daily lives. Our group members are a part of the target audience of this project and this is why we also believe that the end product will be accurate in terms of what it is aiming to accomplish. Additionally, our knowledge of different genres of games, companies, game series, and player preferences makes us the perfect team for the task as we already have a sense of where we should put our effort and research when we start implementing PlayPal.

Datasets

For our project, we have only used a single dataset called "steam_games.csv" which is a CSV file that contains around forty-one thousand games that are present on Steam. We acquired this dataset online from:

https://www.kaggle.com/datasets/trolukovich/steam-games-complete-dataset?resource=download

(Also present in our bibliography)

In this dataset, each row represents either a game by itself (which is the case for 93% of the dataset) or a bundle (which is the rest 7%). There are in total 20 columns and our program uses 10 of those columns. What each column we use represents is as follows:

Row Index — Description

1 — Whether the row represents a game or a bundle
13 — The genres a game falls into
2 — The name of the game
5 — The reviews of the game
15 — Whether the game contains mature content or not
12 — Whether the game has achievements or not
18 — The price of the game

10 — Whether the game is singleplayer, multiplayer, or both
16 — The minimum hardware specifications required to run the game
17 — The recommended hardware specifications required to run the game

Computational Overview

Our program starts off by reading our dataset and creating a GameNode instance for each game we read off the CSV file. The reader (named "read_steam_data") also collects all possible genres in a separate set. To accomplish these two tasks, the reader uses multiple helper functions to extract the data from the CSV file as all the columns in the file are strings yet represent attributes that must be represented by an int, float, bool, etc.

After we create our GameNode instances, we generate our decision tree based on the values present in the instance attributes of each GameNode. We have decided to not put every attribute of a game as a deciding subtree and stored attributes that are hard to represent with a "yes" or "no" only inside the game's GameNode object. So, in the end, our tree has a depth of 5, and each layer is as follows:

Root (*) → Genres → Mature Content → Singleplayer/Multiplayer → Games

After our tree is created according to this structure (without the games at this point), our sorting algorithm places each GameNode object in its correct spot (these are all done under the function "load_game_tree"). As an implementation choice, we've decided to put games that fall into multiple genres into each genre subtree which means we have the same game under multiple subtrees. The reason for the implementation choice is if we put a game with multiple genres under just one genre, when the user selects one of the other genres the game falls in, it wouldn't show up in our recommendations even though it also falls under that genre too.

While all this action is happening in the background to generate the decision tree, the user will be selecting their preferences using the interface we created with **TkInter**.

This program uses the **TkInter** library to create a graphical user interface (GUI) for collecting user preferences for playing video games. The program uses various widgets such as labels, radiobuttons, checkboxes, and buttons to collect user input. The collected user input is then returned as a tuple by the **submit()** function.

The **Tk()** function is used to create the main window of the application. The **title()** method is used to set the title of the window to our program's name PlayPal.

The **PhotoImage** class is used to load an image file and create an image object, which we used to upload our created logo for PlayPal. The **Label** widget is used to display the logo on the top-middle of the window.

The **StringVar** and **BooleanVar** classes are used to create variables that store the user input for budget, play mode, reviews, achievements, and mature content. The **Entry** widget is used to allow the user to input a budget value, and the **Radiobutton** widget is used to allow the user to select a play mode, reviews, achievements, and mature content.

The **Frame** widget is used to group the checkboxes for different genres. The **Checkbutton** widget is used to allow the user to select multiple genres. The **grid()** method is used to position the checkboxes in the frame.

The **Button** widget is used to create a submit button, which when clicked, calls the **submit()** function to return the collected user input.

Finally, the **mainloop()** method is used to start the GUI and run the event loop, which listens for user input and updates the GUI accordingly.

After the user makes their preferences, they submit their response and our selection algorithm picks out the GameNodes that fall with the user's needs and returns a list of GameNodes that will be recommended to the user.

Upon getting the list of recommended games, our program opens up another window (using **TkInter** again) and shows the recommended games one by one with every detail we have collected about it from the dataset.

After going through the recommendations, the user can close the recommendation window, change their preferences and get new recommendations instead.

Instructions For Obtaining Datasets & Running PlayPal

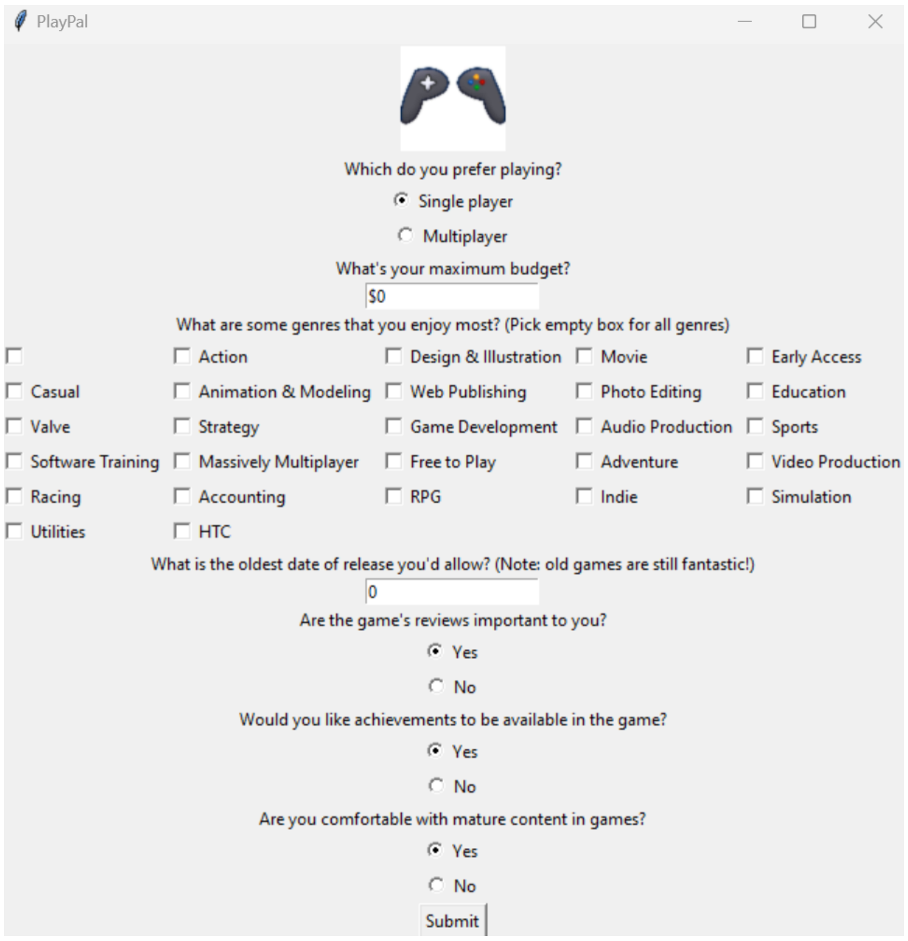
Step 1: Downloading the files

Download all the files that we have uploaded to MarkUs and put them all into a single file named ”PlayPal”.

Step 2: Downloading the dataset

Step 3: Running PlayPal

In the file, run the file named ”main.py”. This should open the window seen below.



Which do you prefer playing?

[✓ Single player]

Multiplayer

What's your maximum budget?

$0

What are some genres that you enjoy most? (Pick empty box for all genres)

☐	☐ Action	☐ Design & Illustration	☐ Movie	☐ Early Access
☐ Casual	☐ Animation & Modeling	☐ Web Publishing	☐ Photo Editing	☐ Education
☐ Valve	☐ Strategy	☐ Game Development	☐ Audio Production	☐ Sports
☐ Software Training	☐ Massively Multiplayer	☐ Free to Play	☐ Adventure	☐ Video Production
☐ Racing	☐ Accounting	☐ RPG	☐ Indie	☐ Simulation
☐ Utilities	☐ HTC			

What is the oldest date of release you'd allow? (Note: old games are still fantastic!)

0

Are the game's reviews important to you?

[✓ Yes]

No

Would you like achievements to be available in the game?

[✓ Yes]

No

Are you comfortable with mature content in games?

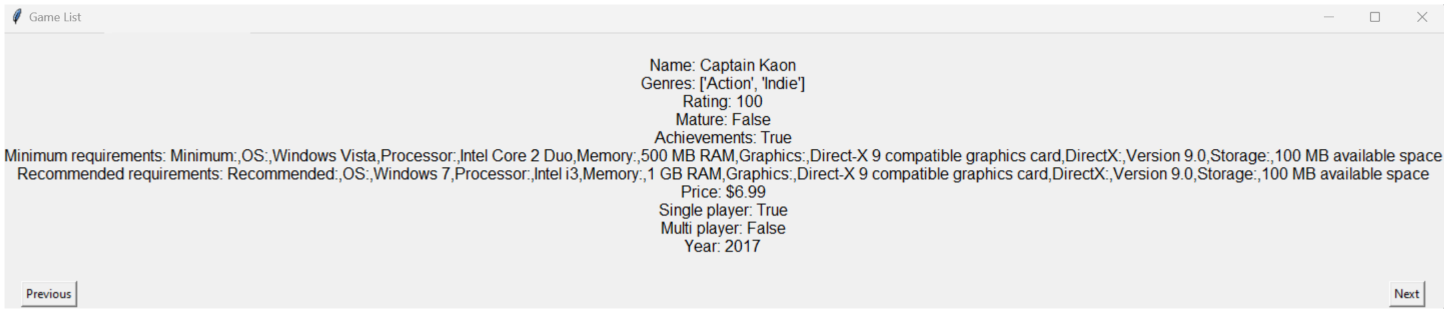
[✓ Yes]

No

Figure 1: The user interface of PlayPal

After you make your preferences, click Submit. This should open a new window similar to the one given below.

On this window, click the Previous or Next buttons to go through the games PlayPal is recommending. To change your preferences and get recommended new games, just close the recommendation and submit your preferences again from the main PlayPal window.


Game List

```
Name: Captain Kaon
Genres: ['Action', 'Indie']
Rating: 100
Mature: False
Achievements: True
Minimum requirements: Minimum:,OS:,Windows Vista,Processor:,Intel Core 2 Duo,Memory:,500 MB RAM,Graphics:,Direct-X 9 compatible graphics card,DirectX:,Version 9.0,Storage:,100 MB available space
Recommended requirements: Recommended:,OS:,Windows 7,Processor:,Intel i3,Memory:,1 GB RAM,Graphics:,Direct-X 9 compatible graphics card,DirectX:,Version 9.0,Storage:,100 MB available space
Price: $6.99
Single player: True
Multi player: False
Year: 2017
```

Figure 2: The user interface of PlayPal

Changes To Our Project Plan

From our initial plan, after learning that we couldn't use external programs that required the user to download and use them separately, we had to change to using a static dataset that can be read without extra steps rather than using the Depressurizer. In addition, after discussing further we decided to divert from using graphs with weighted values to a decision tree that can eliminate unwanted games faster as we can ignore them as we run the program. We also didn't implement a compatibility rating as after further discussion the information we are receiving from the users wasn't comprehensive enough to relate games to the users on that specific level.

Discussion

The results of our computational exploration helped us achieve our goal as the algorithm did recommend us group members games that we liked. We have quite different tastes when it comes to games but PlayPal was able to filter the games in the database it is given and make relevant and informative suggestions to us.

The first obstacle we faced was not being able to use the program Depressurizer which would have let us personalize the user experience. We made the decision to not use it since we needed the program to be pure Python. In addition, when we moved to the dataset that we decided to use at the end we still had to go around its structure and create multiple helper functions for the function "read_steam_data" as every column in the file was a string, not the data type we needed for our instance attributes. Lastly, we had some issues connecting the backend and frontend of our program but we were able to fix any issues related to that in a short amount of time through trial and error. Other than these instances, we did not face any major setbacks and only had to do some simple debugging throughout the code as we went further with the project. Only on the decorative side, we wanted PlayPal to have some identity. Therefore, we designed a logo that resembles a gaming controller made with two P's facing each other which symbolize the P's in PlayPal.

Some next steps for further exploration would be trying to personalize the user experience more by implementing by ourselves what the Depressurizer would've done for us. The first step to moving in that direction would be finding a way to import a user's game library from Steam manually without a third party application. This would allow us to see additional information such as how many hours the individual has spent on certain games and according to that recommend games that are closer to those genres and playstyles. Other than that, putting more parameters into the equation such as the number of reviews the game got, critic reviews, cross platform compatibility, and much more would make PlayPal more accurate and specific with the recommendations it is making. Additionally, the biggest step forward for PlayPal would be integrating into the web so that anyone who wants to use it wouldn't have to download it on their computer and just google it.

In conclusion, PlayPal is a game recommending algorithm for all types of gamers from beginners to experienced ones based on multiple factors. It enables the user to further customize their preferences compared to Steam's filtering algorithm and provides vital information about the game that the user will need before they make their purchase/download. PlayPal uses an extremely big CSV dataset of games that have over forty thousand elements and uses a decision tree in addition to deciding instance attributes to sort games into different categories and then uses the user's specifications to carefully pick games to recommend. While this is the backend of PlayPal, the interface created with TkInter provides a simple yet effective interaction to the user without making them go through unnecessary elements.

References

1. Antonov, Aleksandr. "*Steam Games Complete Dataset.*" Kaggle, 16 June 2019, https://www.kaggle.com/datasets/trolukovich/steam-games-complete-dataset?resource=download.
2. Real Python. "*Python GUI Programming with Tkinter.*" Real Python, Real Python, 30 Jan. 2023, https://realpython.com/python-gui-tkinter/.
3. "*Steam Usage and Catalog Stats for 2022.*" Backlinko, 13 Apr. 2021, https://backlinko.com/steam-users.
4. "*Tkinter - Python Interface to TCL/TK.*" Python Documentation, https://docs.python.org/3/library/tkinter.html.
5. "*[2019 update] Creating an Alphabetized Text List of Your Steam Game Library.*" Steam Community. (n.d.). March 7, 2023, from https://steamcommunity.com/sharedfiles/filedetails/?id=444067719.
6. YouTube. (2019, July 6). "*How to automatically categorize your Steam Library with Depressurizer.*" YouTube. March 7, 2023, from https://www.youtube.com/watch?v=GIhsBkbvYTY.