**Python Game Development**

**Through the Lens of a Data Analyst**

A computer on a table

Description automatically generated

**Disclaimer:** All characters and sprites are the property of their respective owners and are used here solely for learning purposes. I do not plan to sell, distribute, or make any money out of it.

**Introduction**

With the legal matters out of the way, let me share my journey into the fascinating world of game development. My passion for games began early on, and it was this passion that led me to pursue a bachelor’s degree in computer science. I enjoyed playing games, but I often felt that the games I loved were missing something. This drove me to tweak and implement my own changes in the open-sourced games I played, sparking my interest in game development.

Initially, my focus was on game design, but a pivotal moment came during my second to last year at university. I participated in a Data Hackathon (*Datathon*) that shifted my career trajectory towards Data Science. Despite this shift, my enthusiasm for games and programming never waned. I continued to work on small projects, but there was a significant challenge: my intensive focus on Python for data analytics caused me to lose touch with object-oriented programming.

Earlier this year, I decided it was time to revisit my game development skills. Although it is commonly believed that Python is not the best language for game development, I chose to use it because of my extensive experience with it over the past eight years. This decision proved to be fruitful. In this article, I will share the story of how I combined my data science expertise with my love for game development using Python.

**Why Python?**

I began my programming journey in high school with C++, customizing open-source games as mentioned earlier. During my bachelor's studies, I primarily learned Java. Python entered my life much later, when I was working as a Data Analyst for a major bank in Brazil. Initially, I wrote mostly SQL queries to extract data from our warehouse, but I quickly realized that further data cleaning and exploratory data analysis (EDA) were cumbersome. Around that time, I started learning Python and using Jupyter Notebooks in school, which revolutionized my workflow. Queries that once required "SELECT TOP 1" statements became more manageable and practical with segmented cells in Notebooks.

Fast forward a few years into my data science career and working mostly with Pandas Dataframes, I had forgotten most aspects of object-oriented programming, including classes, "self," and methods. However, I had become very comfortable with Python's syntax and structure, making it an ideal starting point. Despite Python not being among the top programming languages for game development, often criticized for being slower than C# or C++, I decided to pursue my hobby project with it, driven by my familiarity and confidence with the language.

During my initial research, I discovered **PyGame**, a key framework for game development in Python. A framework is a platform that provides a foundation of pre-written code and tools to simplify the development process. PyGame, released over 20 years ago, offers a plethora of documentation and tutorials online, including resources on YouTube, GitHub, and *StackOverflow*. Equipped with a Mac M1 and after completing a few test projects, I became convinced that Python was a viable choice for starting my game development journey.

A black rectangular object with white text

Description automatically generated

**Designing the Game**

Abc

**Resources**

Teenage Mutant Ninja Turtles: Shredder's Revenge

Marvel vs Capcom (1 & 2)

A cartoon character in blue and red

Description automatically generated

**Next Steps**

While it would be easy to focus on adding new characters, move sets, and jumping mechanics, one aspect that could significantly increase *replayability\** is procedural level generation.

*Replayability: the quality in a video game, music recording, etc. of being suitable for or worth playing more than once. Synonym: replay value.*

Currently, the game features only one level: the streets of New York City. My initial plan included three levels: the streets of New York, the sewers/subway tunnels, and Central Park. However, due to time constraints and other priorities, I couldn't complete these additional levels. Some early assets and sketches for these levels can be found on the project's **GitHub** page.

To minimize the workload required to create new challenges for players, **procedural generation of the map** is an effective solution. As described earlier, the map is composed of tiles representing different elements: the skies, foreground (city elements like cars, trees, poles), midground (streets), and background (buildings). The procedural generation method would randomly select a new set of these elements and dynamically add them to the edge of screen – this would ensure that every new game presents a fresh and different cityscape for players to explore. Additionally, a randomizer for enemy quantities, health packs, and boss encounters could further enhance the gameplay experience.

The key consideration for implementing this feature is ensuring that the background and midground elements match seamlessly. For example, if a building with a garage door is selected, the algorithm should pick a corresponding street tile, such as a lowered sidewalk with tire marks, to ensure visual coherence.

A screenshot of a video game

Description automatically generated

As illustrated in the example above, the algorithm must recognize the selected building and choose a matching street element to maintain consistency in the world’s environment.

**Conclusion**

Abc…