A black and white image of letters

Description automatically generated

By Vedran Muftic

October 2024

**Hustle City - Documentation**Contains spoilers!

1. Gameplay
2. Development
3. Bugs, glitches and patches for future releases
4. Concluding thoughts
5. Gameplay

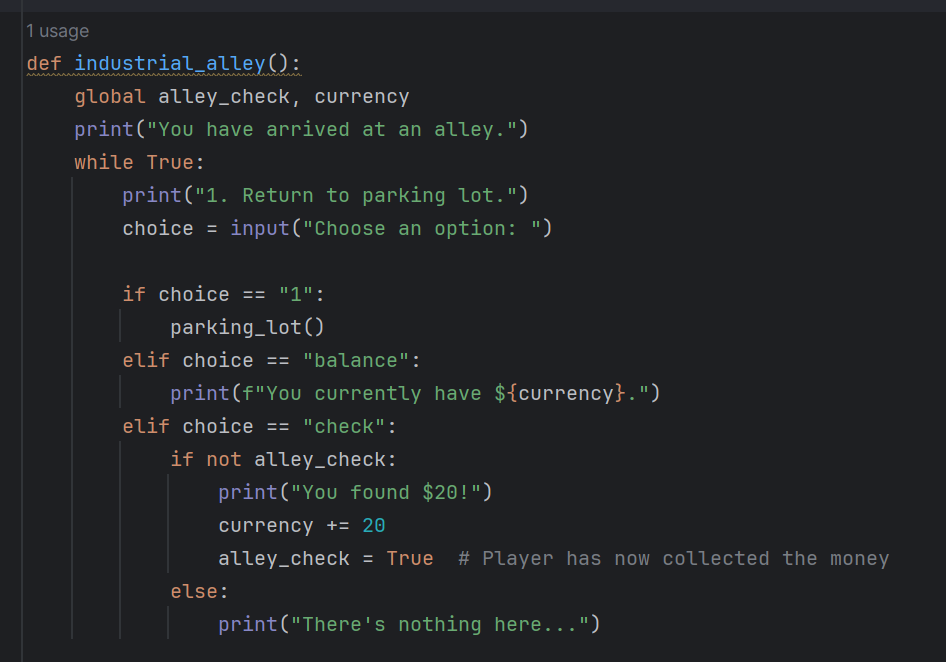
Hustle city is a text-based adventure game where you input commands to move around and interact with the environment. The goal of the game is to accumulate $10,000 by hustling, doing various forms of tasks such as investing, gambling, working at a café and scavenging. The game has an open world so the player can go explore and find clues about where to find money. You can find clues by talking to your contacts; however the game has some red herrings. For example, the pond and the newspaper do absolutely nothing and two of the contacts are quite useless.

1. Development

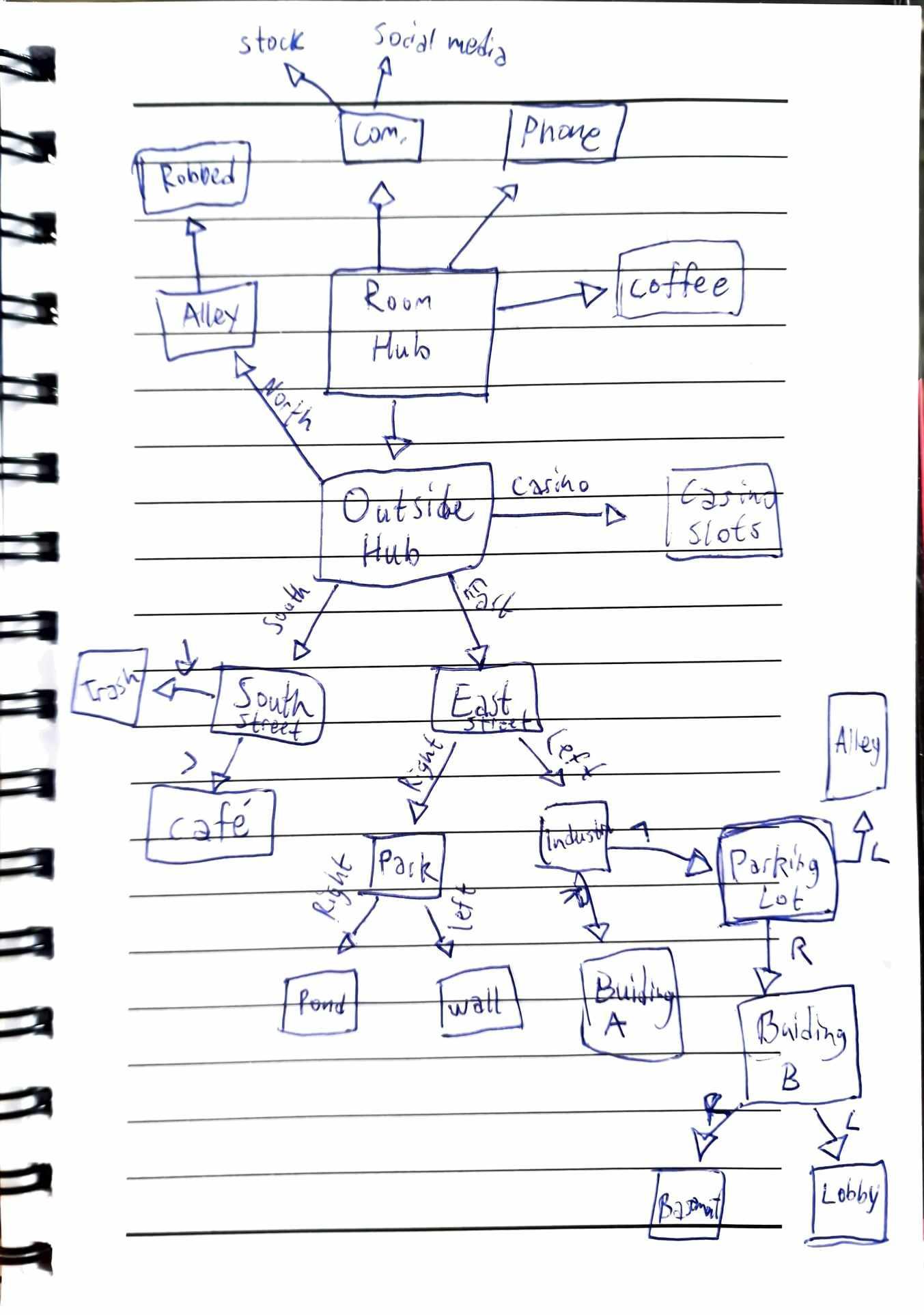
Hustle City is based on text-based games from the era of Apple II computers, such as The Oregon Trail and Déjà Vu. Unlike a lot of choose-your-own-adventure games, I wanted to make a forgiving game where you don’t lose the game if choosing the wrong option; you are relatively free to explore the game map to your liking.   
At the time of writing this, I had been learning Python for about 2 months, and this was my first big project. The idea came about after creating various small programs, such as creating a generic database which eventually became the social media option in the game as well as creating the slot machine mini game. The idea was to put all small projects together into one big code, which eventually became this game. The whole project took about two weeks to complete.

I am definitely not a huge fan of text-based point and click games but, by making one, I learned a lot about Python programming. The most challenging part of the game was learning how to work with global variables that affect each other. The most grueling part of the development was programming the chat bots. The chatbots are not AI and work in a very simple way; they have a pool of string responses and respond to certain words you type. I am sure there are a million ways I could have made these chat bots better, but this is the way I did it.

Another thing that I learned was how to look at the code through the point of a game designer. The biggest design choice for me was to choose how cryptic I wanted the game to be. I had to find a balance between not telling the player exactly what to do and making things explicit enough for the player to understand what they can do in the game. The game was made for players who are already familiar with the common actions of a text-based game, such as ‘take’, ‘check’ ‘look’. In the end it is up to the player to explore the map and find clues.



Global variables such as currency and boolean are frequently used in the game. In the example above, the boolean variable tracks if the player has completed certain tasks, in this case it tracks if the player has collected the money.

Original drawing of the game map.

1. Bugs, glitches and patches for future releases

* Losing money by asking Chester for stock tips can give you negative currency. Also, having negative currency and getting robbed in the North Alley will reset your currency to 0.
* If you lose all your money at the slot machine, you can still choose to play. If you do so the game will ask you for money to bet, but having no money, you cannot play and the game will continuously ask you to make a bet, effectively soft locking the game as there is no way to exit the slot machine game once you choose to bet.
* Checking the wall at the park will unlock Keith’s second stock tip. Getting the tip should reset the tip until you check the wall again however, due to a glitch the tip resets as soon as you cut the call with Keith.
* Initially, the pyfiglet library was used for a stylized ASCII title screen but did not work when exporting the game as an exe file.

1. Concluding thoughts

This was not a professional attempt at making a game, but merely a hobby project. I did not follow any game making or programming conventions, so I am painfully aware that my game is riddled with non-optimal spaghetti code alongside some vague documentation. This was a fun and educational project, and I will continue to program in Python as well as taking a crack at other programming languages such as C or 6502 assembly for NES homebrews. If I’m really dedicated (we’ll see!) I might tackle more complex architecture and learn x86 assembly. I will also continue to make updates to this game and might implement a proper GUI and some visuals.