* **Project description:** This project is called “fireboy and watergirl v.112”. The game is based on the online game “Fireboy and Watergirl”, which is essentially a game where players control characters, get over obstacles, and get to the destination, but with more or different features added onto the original game, and with a new mode: “maze mode” added.
* **Competitive analysis:** On top of creating traps that could kill the character, generating gems that player can collect, and elevators that requires the push of buttons, I am planning to add randomly generated monsters to this game, which could chase players automatically (AI based on backtracking). And I am also planning to add a maze mode, where players need to control the characters in a maze to find the key and the exit
* **Structural Plan:** I have so far created five files: Main, background, character, and obstacle, and spritesheet. The Main file contains the core MVC of this game, calling functions that have already be written in other subfiles:

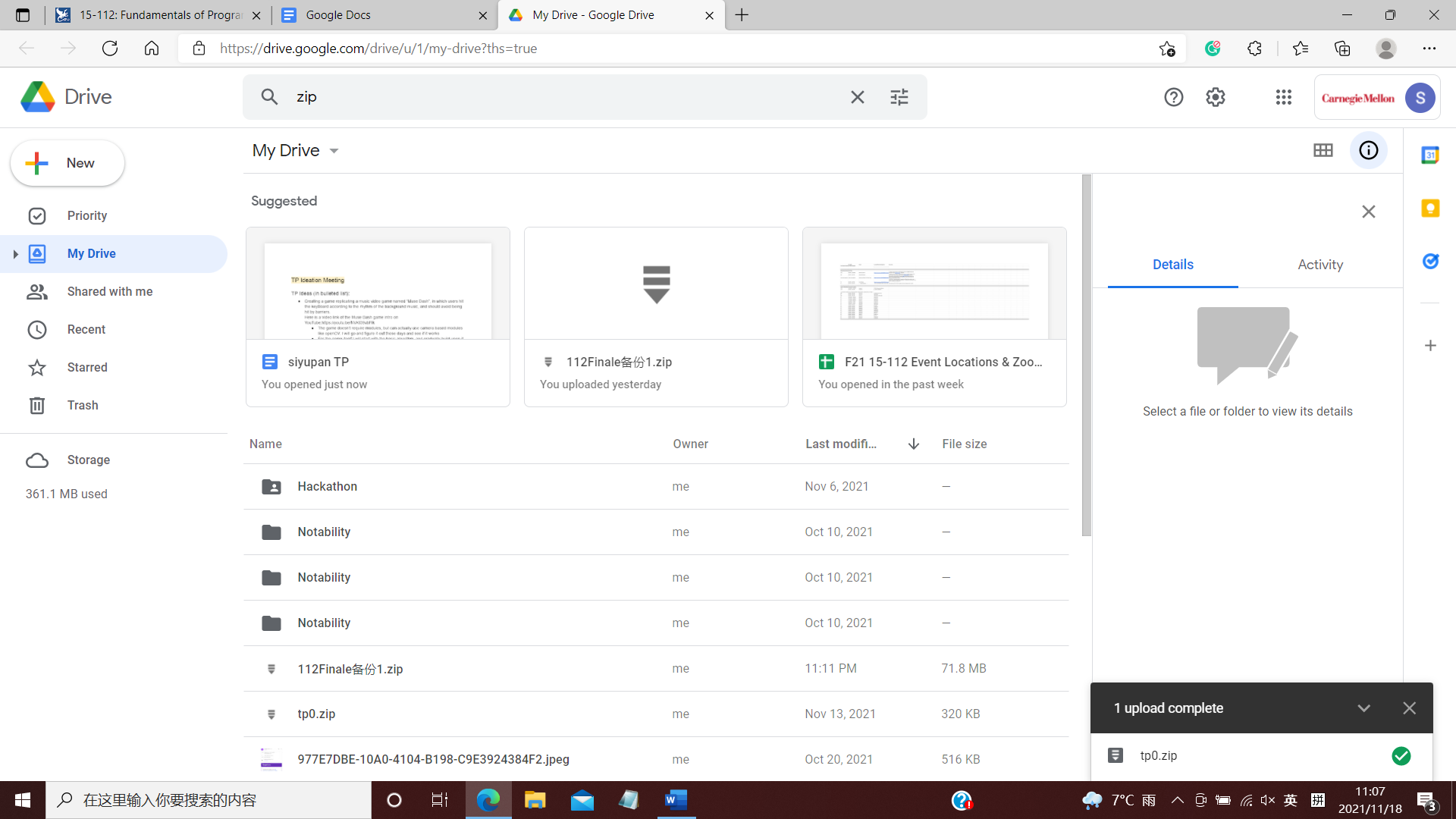
The background file contains three different classes of three different terrains of the game: the Floor, the Ceiling, the Ramp, and the Wall class.

The character file contains the Character class with multiple attributes, a lot of helper functions that detects key inputs or could be put under timerfired in Main.

The spriteSheet file contains the SpriteSheet class and multiple functions of chopping different preliminary spritesheets.

The obstacle file contains functions and classes of traps that could kill the players

* **Algorithmic Plan:** I have already completed graphics of characters(spritesheets), which is a very tricky part of the project. Apart from that, the trickiest parts would be writing the monster AI to allow monsters to chase players automatically; in addition, I could add competition mode where players need to solve the maze and collect the key; the map will be generated randomly, and there will be path suggestions. Both of those two things require backtracking and deep algorithmic thinking, and thus most challenging.
* The **timeline** would be: complete a preliminary replication of the original game by TP1, add monster generation and tracking features, and random maze generation by TP2.
* **Version Control Plan:** I am putting different versions of my code in google doc, just in case I lost my computer or something.



* Module list: I may use pygame for sound effect, but it depends on the time.

TP2 UPDATE:

* Project description: a Fireboy and Watergirl game with 2 modes, one kind of replicates the original game with different features, and the other is a newly designed “maze mod”"
* Competitive analysis: this game has more features than the original game, including movable platform, mechanical structures, and player-tracking cannonballs. And the maze mode is also a completely new mode, which could also be pretty fun for player to play.
* Structural Plan: now I have 7 files:

The Main contains the main file of this program. The main function consists of MVP with different modes.

The background file contains three different classes of three different terrains of the game: the Floor, the Ceiling, and the Wall class.

The character file contains the Character class with multiple attributes, a lot of helper functions that detects key inputs or could be put under timerfired in Main.

The spriteSheet file contains the SpriteSheet class and multiple functions of chopping different preliminary spritesheets.

The obstacle file contains functions and classes of traps that could kill the players.

The maze file contains the maze class for Maze mode

The mazechr file contains the class of characters in Maze mode

* Algorithmic plan: Update: random maze generation is implemented by using Prim algorithm, and path suggestion is also implemented by using back-tracking (DFS).
* Timeline: MVP requirement has been achieved. The completed game could be finished by TP3
* Version control plan: google drive backup is updated.

TP3 UPDATE：

* AI in maze mode added
* Different level choices in maze mode available now
* Test mode in normal mode implemented
* UI improved