Roasted Rooster Road

The game I decided to develop was influenced by Crossy Road, where a chicken attempts to cross an endless amount of roads, avoiding the speeding cars. However, given the freedom of recreating the game, I will implement a unique element of buffs that the player can attain. Within the program, this will translate to requiring a third class for managing different power-ups. While the original crossroad had a moving screen that adapted to player movements, this will most likely be extremely difficult to implement based on the limitations of time and content taught in class; it would require more classes for the roads and background and an algorithm for camera tracking. Instead, a static environment will be used with the objective of reaching the top of the screen. Upon reaching the border, the feature of different levels can be utilized, and the chicken will be teleported back to the start with the caveat of increased difficulty in the form of higher car spawn rates and car speeds. Furthermore, the movement of the chicken will be different to the typical holding of keys; instead of resembling the original "Crossy Road", only singular taps will be registered as movement by a preset amount of pixels in a direction.

Reflection

Originally I had difficulty implementing timers for the cooldowns of particular objects such as buffs and spawn rates, so I used time modules. However, ultimately I decided to challenge myself and whilst it took more lines of code, the game functioned fundamentally the same without the use of other definitions. Also, I had great difficulty creating the UML diagrams as I could not just copy down the code I had written; instead, it required thinking about the different parameters and data types. I was especially confused regarding images; I sought a teacher and was able to do it in the end. Another problem I encountered was the movement of the character, as continuously pressing a key would similarly continuously move the player. Therefore I created a new boolean variable that would keep track of the state of "events", rendering any further inputs of a key press futile unless a "key up" event is registered. Future iterations of this project could include the choice of different playable characters, similar to the original game. These characters could be attained through a gacha system which can be done by collecting coins or achieving high scores. This would undoubtedly make the game more enjoyable as it creates a purpose for the player to reach higher records and "farm" the coins for their favorite characters.

UML diagrams

Chicken
chicken: pygame.surface position: list(int, int) speed: int bufflist:list(str) advance:boolean level: str velocity:list(int,int) position:list(int,int)

setVelocity:(vx:int,vy:int)
setPosition:(x:int,y:int)
move()
draw()

Buff
buff: pygame.surface buffRect:list(int,int) name:str

__init__(name:str)
setPosition:(x:int,y:int)
draw()

Car
car: pygame.surface buffrect:list(int,int)

setPosition:(x:int,y:int)
draw()
move()
setVelocity:(vx:int,vy:int)