Cyber Security Education

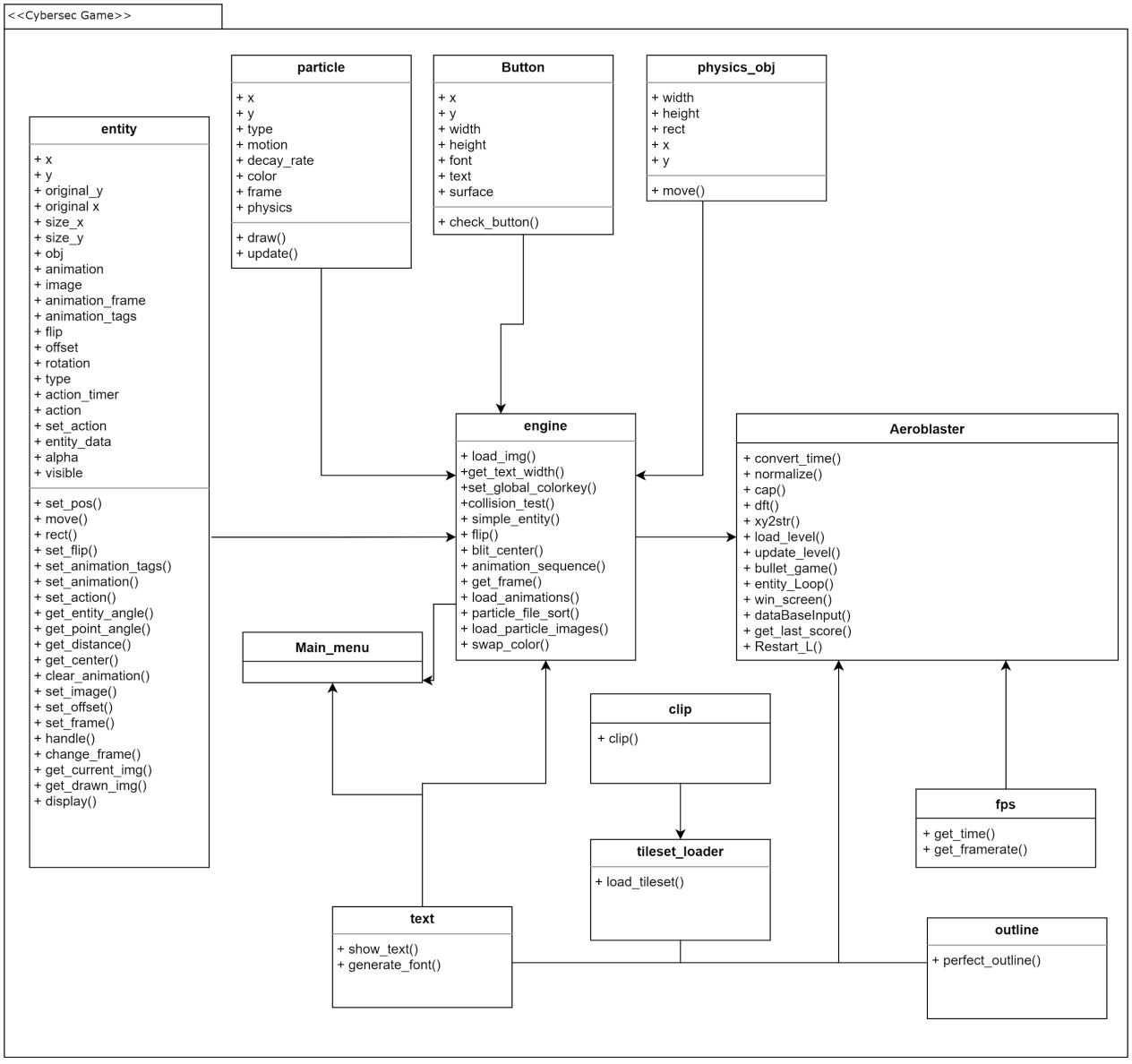
Design Document

By Liana Villafuerte, Bruce Wagner, Kyle Stead, and Hussein Okasha

System Architecture:

Once we had finished the project, we redid our original system architecture to include all functions, variables, and interactions between classes. Figure 1 below shows the final product of the code's architecture and structure.

Figure 1



outline
This class gives anything passed into it a colored outline.

- The perfect_outline method returns a provided image with an outline added around the image.

text
This class is responsible for working with text in regards to displaying it on screen.

- The show_text method is responsible for rendering provided text onto the game window.
- The generate_font method returns data that determines how the rendered text looks on screen.

fps
fps is in charge of keeping track of time and frames per second information.

- The get_time method returns how much time has passed since the last tick.
- The get_framerate method returns how many frames are being displayed per second.

tileset_loader
tileset_loader is responsible for loading the tiles that are used for the levels.

- The load_tileset method takes a file path to the image of the tileset and splits the image into separate tiles that it then returns.

clip
The clip class is used by the tileset loader to help get the separate tilemap images.

- The clip method returns an individual tile from a main image with provided coordinates.

physics_obj
The physics_obj class is responsible for detecting collisions between objects.

- The move method checks to see if an object can be moved to a specific spot and moves it there if it can.

Button
The Button class is used to place buttons on the screen which are used for the user interface.

- The check_button method returns a boolean that determines whether the mouse has clicked the button or not.

particle
The particle class is used to draw particle effects on the screen.

- The draw method is used to place the initial effect.

- The update method is used to update the appearance of the effect based on how much time has passed.

entity
The entity class is responsible for storing information about the game objects.

- The set_pos method is used to set the position of an entity to a specific location.
- The move method will attempt to move the entity and takes into account collisions.
- The rect method creates a rectangle.
- The set_flip method is used to flip the entity during the game.
- The set_animation_tags method is used to apply tags to the animation.
- The set_animation method is used to add the animation sequence to the entity.
- The set_action method is used to bind an animation to a specific action.
- The get_entity_angle method is used to find the angle between two entities.
- The get_point_angle method is used to get the angle between the entity and a point in space.
- The get_distance method is used to find the distance between an entity and a point in space.
- The get_center method returns the center of the entity.
- The clear_animation method is used to reset and stop the animation.
- The set_image method is used to give the entity an image.
- The set_offset method is used to offset the entity.
- The set_frame method is used to set a specific animation frame.
- The handle method is used to advance the animation by one frame.
- The change_frame method is used to advance the animation by a specified number of frames.
- The get_current_img method is used to return the image of how the entity looks at that point in time.
- The get_drawn_img method is used to get the image that is going to be rendered to the screen.
- The display method is used to render the provided image onto the screen.

engine
The engine class is used to help the game run and reduce the number of tasks that the Aeroblaster and Main_menu classes have to handle.

- The load_img method is used to load any images that the rest of the game needs.
- The get_text_width method returns how many pixels the text is across.
- The set_global_colorkey method is used for setting the color key for the game.
- The collision_test method is used to check if any game objects are colliding.
- The simple_entity method is used to create a basic entity.
- The flip method is used to flip the image.
- The blit_center method is used to put an image at the center of the coordinates.
- The animation_sequence method is used to clean up the animation sequence.
- The get_frame method is used to get an entry from the animation database.
- The load_animations method is used to load animation information from a file.

- The particle_file_sort method is used to order the particle effects.
- The load_particle_images method is used to load the particle images from files.
- The swap_color method is used to change the colors on an image.

Aeroblaster

This is the main game, it is responsible for the actual functions and code in the game as well as the vulnerabilities. It contains the following methods.

- The convert_time method is used to convert the time displayed from milliseconds to minutes and seconds
- The normalize method is used to calculate the player's velocity when they jump or have a recoil from the gun
- The cap method is used to calculate the maximum speed for the player's movement either falling or while walking
- The dtf method is used to calculate the maximum speed of different entities such as the bullets or the player's velocity. This method works alongside cap and normalize methods.
- The xy2str method is used by the program to convert the level map from strings as it is stored to actual positions in the game.
- The load_level method is responsible for generating and loading in the map from the raw .txt string format to a set of positions which then have tiles and cores added to them.
- The update_level method is used to clear the screen and get it ready to read and generate the next level
- The bullet_game method is responsible for calculating and drawing the graphics for both the turret bullets and the player's bullets.
- The entity_loop method is responsible for scanning through the list of entities within the map and making sure they work properly. This method is the brains of the logic behind the turrets as well as the graphics for the turrets and the cores in the map.
- The win_screen method is responsible for displaying the final win screen when the players completes the game.
- The dataBaseInput method is responsible for generating and adding the player's scores into the scores.db file through an SQL query.
- The get_last_score method is used to check if the player completed the first level within the required time or not, it achieves this via an SQL query.
- The rest of the class is a large while loop that runs through the game, the pause menu, and everything else until the player wins or quits, level 1 is shown in Figure 1.

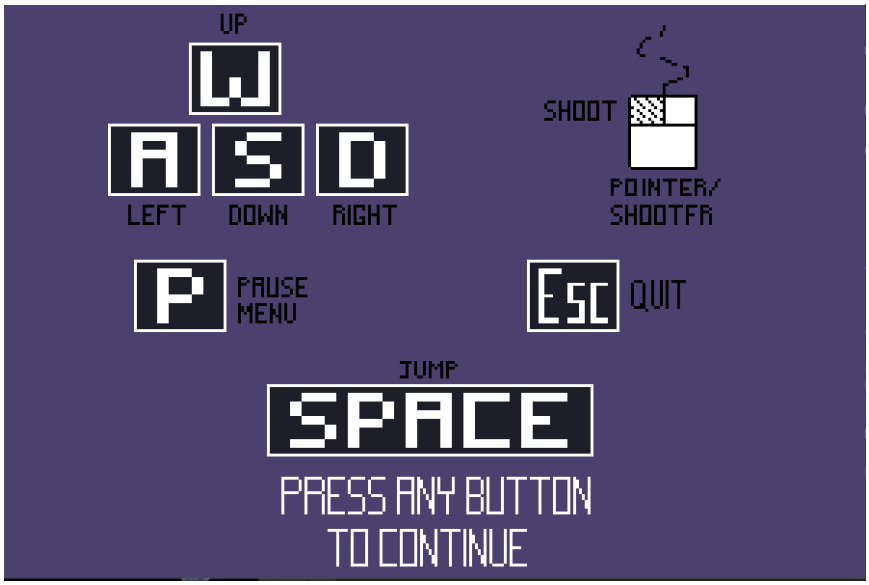
Figure 1

Main_menu

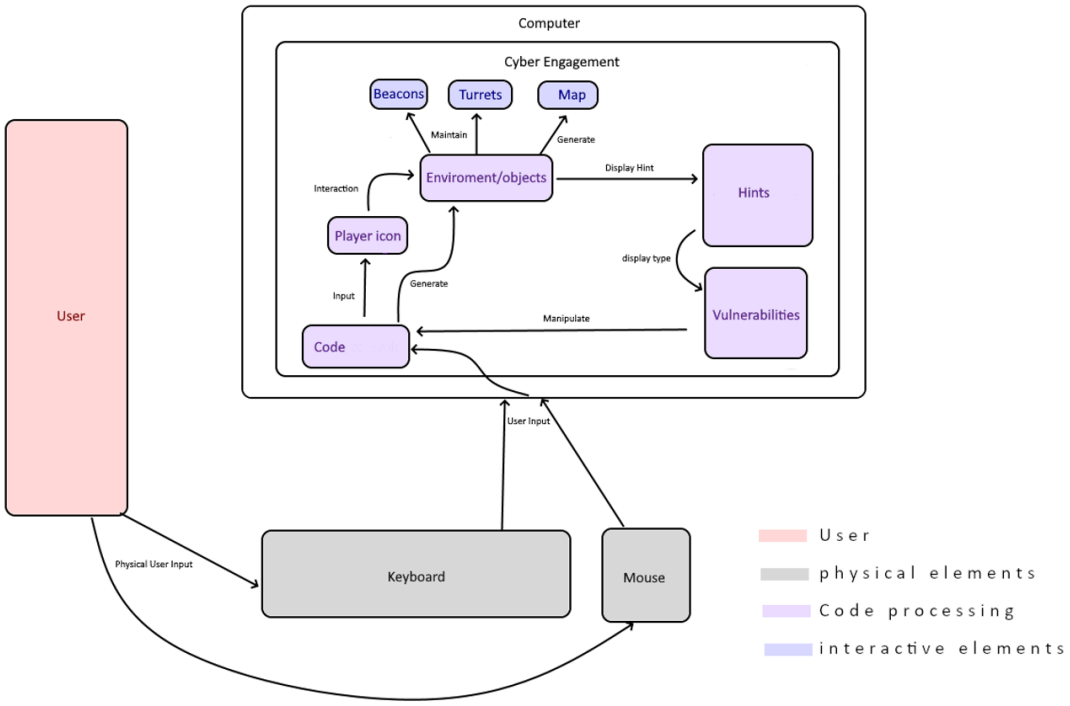
The main_menu class is responsible for the main menu of the game, the intro screen as shown in Figure 2, and sending the data that is put into the name and level text fields to the Aeroblaster class. It is also responsible for the controls menu as shown in figure 3.

Figure 2

Figure 3



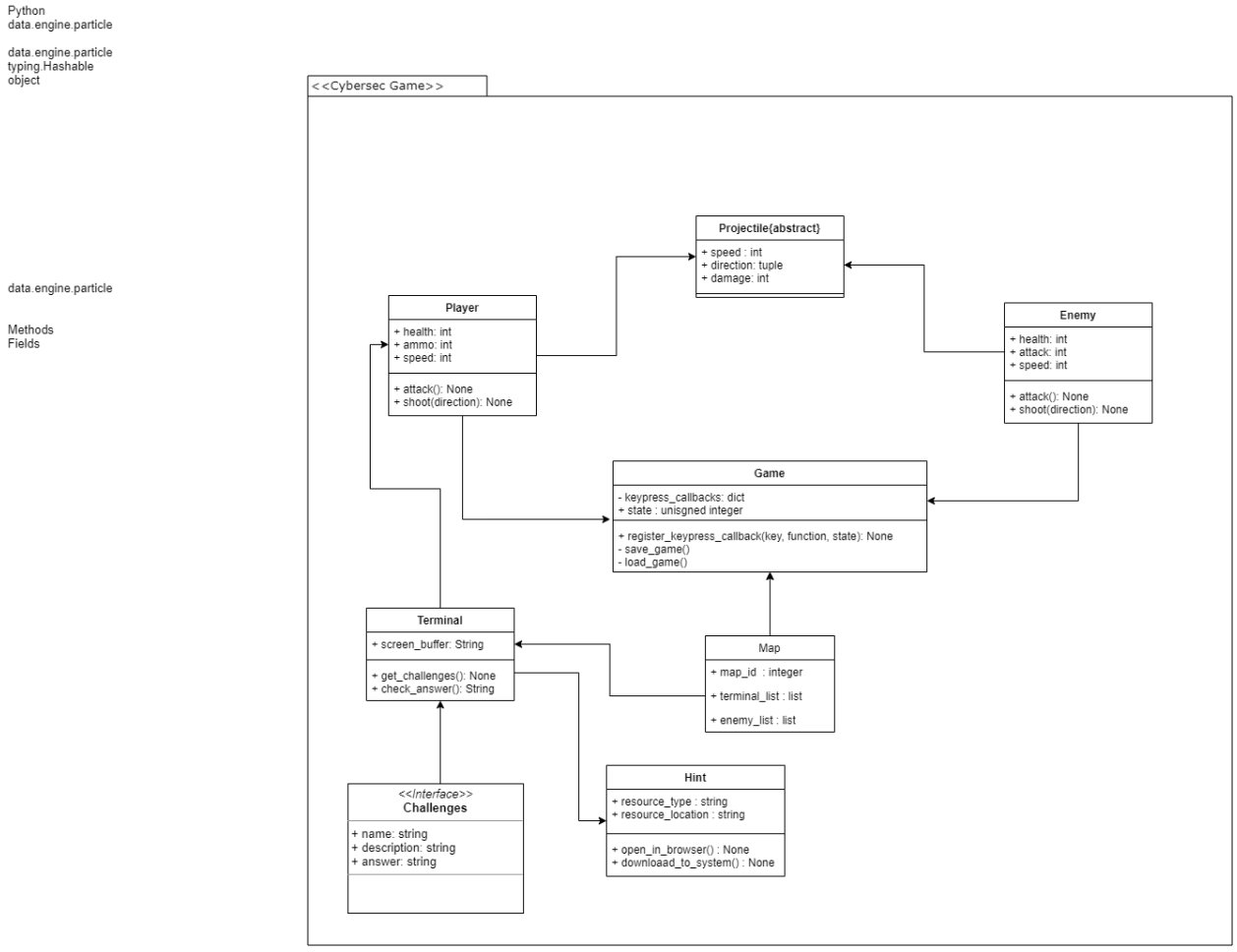
Final design diagram used:



Original Architecture design made:

As mentioned earlier, while making the program, our initial structure was based on an old format of code with different ideas. The original architecture was based on a top down dungeon crawl instead of the current sideways shooter style dungeons. Figure 5 below is the structure and original design.

Figure 5



Old Designs:

The game had to go through a couple design changes due to how it needed to flow better and be more understandable and user friendly. Figures 6 through 8 below shows the original designs for the main menu.

Figure 6

Figure 7

Figure 8