**Progress Report**

**- Increment 1 -**

**Group #15**

# Team Members

* Brian Friedlander, bdf20, Bdf20
* Cooper Parmett, cmp20a, Cooperparmett
* Madison Dail, mrd20a, MadisonDail & OtterWolves (**note**: OtterWolves is when pushing from vscode)
* Chelsea Wang, cw20b, FloatingStory
* Richard Simpson, rjs19c, razskull
* **Project Title and Description**

Title: Florida State Pokeversity

Description: A 2D pokemon-like game with a mix of functionalities from different pokemon games using the FSU campus as the map. Game will include typical turn-based battles, exploration of a map, catching pokemon, and farming berries.

* **Accomplishments and overall project status during this increment**

Developed class named Entity which inherits from the Sprite class from the Pygame API. This Entity will be and has been already used as the base of any object that requires a sprite to be displayed in the game window at some point.

Developed class named Player, which inherits from Entity, allows for movement of Player sprite on screen depending on the user's keyboard arrow inputs(up to move up, etc.). Made player to spawn in middle of screen.

Developed class named NPC, inherits from Entity, when the Player is less than a tile away from the NPC and the user uses mouse to click on the NPC sprite then Dialog will be drawn onto the screen.

Developed class named Trainer, inherits from Entity, when Player moves from the left and is above the Trainer sprite then the Dialog will be drawn onto the screen.

Developed class named Dialog, which will render a rect for the dialog box and render text character by character onto the screen.

Developed class named Pokeball, which allows for the storage of Pokeballs in the database

Developed class named Items, which allows for the storage of Item entities in the database

Developed class named Pokemon, which allows for the storage of Pokemon entities in the database

Developed class named Stores, which allows for the storage of the Shop inventory in the database

Developed class named Moves, which allows for the storage of Pokemon’s moves in the database and to be used during battling

Developed Battle functionality which will allow users to battle each other. (Unfinished so far)

Developed Menu and buttons that allow you to change between screens, new game screen created that will allow you to create a new game. Load games will let you choose which game to play from. Quit will quit the game for you.

Developed class named Block to spawn types of blocks/walls around that the player cannot walk through.

Developed class named Game which makes a Game object to to exist so it can be instantiated and run. Includes functions to create in-game features such as the map, events, draw, and new.

Added config file to hold global variables such as RGB colors, dimensions, and map layouts in the form of arrays..

* **Challenges, changes in the plan and scope of the project and things that went wrong during this increment**

Some challenges were encountered due how pygame expects its gameplay loop to look in order to get desired results, all challenges were dealt with by researching online about the topic and learning why an issue was occurring and adjusting it so that the issue is no longer there. A challenge encountered was understanding how entities in the pygame runtime loop were drawn and updated, through researching online we realized that there is a certain order to call the methods in order to generate the expected visual results. Another challenge would be understanding how Pygame expects the runtime loop to be to avoid unwanted visual glitches(caused by calling update() twice in one iteration of a loop), this was solved through research online and moving the fill call in the pygame runtime loop before any updates and draws are called.

Changed from Python Arcade API to Pygame API as we found that Pygame provided more fitting modules with how we want to develop our game, with support for intuitive map making and the sprite module to display entities during gameplay.

Changed from using Firebase to SQLite as a database to store information because we found that it was easier to attach the database to the Python code using SQLite.

What went wrong with this increment is that everyone has been doing their functionalities in separate branches and we have not fully managed to merge all of our developed code together. When we attempted to merge the code, there were a lot of bugs that will need to be fixed in the next increment.

A challenge we encountered was finding proper time to get a lot of work done for this project due to work for other classes. This made developing the battle function difficult because it requires a lot of different classes to be done and be made fully functional.

* **Team Member Contribution for this increment**
* Brian Friedlander

**Progress report**: Challenges and Project Status

**Requirements and design document**: Functional Requirements

**Implementation and testing document**: Programming language and Non-execution testing

**Source code**: Menu, battle features (pokemon, moves, type matchups)

**Video/presentation**: Battling Features video

* Cooper Parmett

**Progress report**: Plans for next increment, accomplishments

**Requirements and design document**: Overview, functional requirements, non functional requirements, operating environment

**Implementation and testing document**: functional testing

**Source code**: Entity class development, map design, config global variables.

**Video/presentation**: Classes such as Block. Mentioned config file and overall functionality. Not mentioned in the video presentation itself, but also worked on Game class and certain functions such as init, events, new, update, main, and createTilemap.

* Madison Dail

**Progress report**: Plans for next increment, team members, project title and description, Accomplishments during increment, challenges and changes in plan

**Requirements and design document**: Functional Requirements

**Implementation and testing document**: Non-execution based testing

**Source code**: Menus, map design, player spawning, and screen switching

**Video/presentation**:​​ Change from python arcade to pygame, demoed the menu, screen switching, and player spawning, discussed plans for next increment.

* Chelsea Wang

**Progress report**: Accomplishments during increment, challenges and changes in plan, plans for next increment

**Requirements and design document**: Assumptions and dependencies, Functional requirements, Non-Functional requirements, Use Case Diagram, Class Diagram, and Sequence Diagram

**Implementation and testing document**: Programming language, API and Database used, Execution-based Non-Functional Testing

**Source code**: Entity class development, player movement, and trainer/npc encounter and dialog.

**Video/presentation**: Overview, state of project, accomplishments, demo of player movement and trainer and npc encounter, why change from python arcade to pygame, my plans for next increment.

* Richard Simpson

**Progress report**: Accomplishments during increment, challenges and changes in plan, plans for next increment

**Requirements and design document**: Class and sequence diagrams

**Implementation and testing document**: Database used

**Source code**: Database creation and management

**Video/presentation**: Showing off classes made and test main file. Plans for next iteration

* **Plans for the next increment**

Add ability to pull information from database whenever

Add more map entities (NPCs, blocks, tall grass, etc)

Design different areas of map

Add battling functionality

Add ability to catch pokemon

Add farming where plants can grow.

Generalize the Dialog class.

Add more direction detection for Dialog box popup for Trainer entities.

Add collision detection.

* **Link to video**

<https://www.youtube.com/watch?v=E0iudpYg4ec>

\*Video was sped up a tiny amount for time.