Pawell, Auston

  https://github.com/Takoyaki131/Pokemon-Battle-Project

Design Document for Pokemon Battle Project

Contents

[Proposal 2](#_Toc116058081)

[motivation 2](#_Toc116058082)

[Challenges 2](#_Toc116058083)

[Goal 2](#_Toc116058084)

[Plan & Approach 2](#_Toc116058085)

[Pokémon Mechanics 3](#_Toc116058086)

[Type Bonus 3](#_Toc116058087)

[Accuracy 3](#_Toc116058088)

[Pokémon Battle Statistic Moves 3](#_Toc116058089)

[Status Effects 3](#_Toc116058090)

[Key Method Logic / Pseudocode 3](#_Toc116058091)

[Classes, Handlers and Utilities 3](#_Toc116058092)

[Battle Logic 4](#_Toc116058093)

[Turn() 4](#_Toc116058094)

[useMove() 4](#_Toc116058095)

[UML Diagrams 5](#_Toc116058096)

[Trainer 5](#_Toc116058097)

[Move Tree 5](#_Toc116058098)

[Interfaces 6](#_Toc116058099)

[References 6](#_Toc116058100)

# Proposal

## motivation

This is a personal project based on the famous game franchise Pokémon. More specifically, this program will attempt to recreate the inner workings of how a traditional Pokémon battle is calculated and performed. Mechanics like damage taken by poison, attack reduction when burned and other things are reference through the Pokémon wiki but are still recreated in the program using original code. Without any proper experience in game coding or professional code outside school projects, this project will serve as a introduction for me to somewhat think like a game developer and give me an opportunity to problem solve on a much larger scale that what was experience in my early programming courses.

## Challenges

The biggest challenge will be maintaining and a building a project with a size that I don’t have experience in. Many projects that I have made have been in class where the project can be built and testing within a month or so. With this project however, an end result isn’t something I am coding towards. It will be more like an continuous ongoing project as I find different ways to improve on the project either by adding new features or refining parts of code based on what I am currently learning while in school.

## Goal

The goal for this project is to build a application that can somewhat simulate a pokemon battle with its core features and mechanics. It will begin on the console and will later shift to some basic GUI layout.

## Plan & Approach

The plan is built into three major phases on the project. The first is to design all the necessary class and their relationships. This will be a key phase as it define the scalabilty of the project meaning the ability to not only add more pokemon and moves but when new mechanics are added to the game, there isn’t a lot of backlog on progress due to redefining that code to fit the new mechanic. The second phase is to build out the core of the game, meaning including all the fundamental mechanics of a pokemon battle such as type bonus, switching out pokemon, etc. This will most likely be the longest step as there are many key features that define what goes on in a pokemon battle. The last phase will be to convert from console to a GUI layout. This will be beneficial as the users can feel more comfortable in battle by pressing buttons instead of entering string inputs. The shift to GUI will free up a display for developers to perform easier debugging and testing as well. During the whole process, there will moments in production where the focus will be on adding pokemon and moves along the way.

# Pokémon Mechanics

## Type Bonus

Type bonus is a multiplier that is applied to a pokemon when it attacks and is calculated by the relationship between the move type and the pokemon receiving the move. Also within in type bonus is what is known as Same Type Attack Bonus (STAB) which increases the moves effect is the type of the move matches one of the types of the pokemon using it.

Ex:

* Pokémon: Pikachu (Electric) + Move: Thunderbolt (Electric) = STAB
* Pokémon: Pikachu (Electric) + Move: Tackle (Normal) = ~~STAB~~

## Accuracy

## Pokémon Battle Statistic Moves

## Status Effects

* Poison (Gen 1)
  + inflicts damage equal to 1/16 of maximum HP every turn
* Burn
  + Inflicts damage equal to 1/16 of maximum HP every turn
  + Reduction of physical attack power by 50%
* Freeze
  + Unable to use a move
  + 20% chance each turn to thaw out
* Paralysis (Gen 1)
  + Reduction of speed stat by 25%
  + 25% chance to be unable to user a move
* Sleep
  + Last randomly between 1 to 7 turns
  + Unable to user a move
* Flinch
  + Prevents a Pokémon from attacking second

# Key Method Logic / Pseudocode

## Classes, Handlers and Utilities

While classes will hold the fundamental functions of a class such as accessors and mutators, a separate class called the Handler class will function as the object that will allow the user to fully interact with the object such as view specific attributes and editing those attributes. The handler will also have the responsibility of saving updated date either in the form of serialized objects or updating text files. Utility files will hold functions related to that class or help in setting up the handler classes.

EX: TrainerUtilities.java will have a function that will focus on helping the user select a trainer to edit and will return that Trainer object which the TrainerHandler can accept and begin to either edit or use in battle.

## Battle Logic

### Turn()

The program’s event order for each individual turn in battle.

**Step 1**. Update the Pokémon based on previous status condition – **preTurn()**

* If the Pokémon fainted pre turn, then prompt for a swap

**Step 2**. Get each Trainer’s move for the turn (Mid Turn)

Each trainer gets 3 main options to choose each turn

* Pokémon attack
* Pokémon swap
* Use item on Pokémon

**Step 3**. Determine turn course of action

Each move has 4 potential outcomes

* Each trainer swaps out their pokemon
* Left Trainer switches / Right Pokémon attacks
  + Make the switch of the Left, then execute attack function using the Right Pokémon
* Right Trainer switches / Left Pokémon attacks
  + Make the switch of the Right, then execute attack function using the Left Pokémon
* Both Trainers attack
  + Who attacks first is determined by Pokémon’s speed statistic
  + Execute the attack function twice based on who goes first

**Step 4.** Determine the battle status of the game – **postTurn()**

There are 3 endings to a turn

* No Pokémon faints – go to next turn
* A Pokémon faints but the trainer has more Pokémon available
* A Pokémon faints and the trainer has no more Pokémon available.
  + This ends the battle completely

### useMove()

Function logic order

1. Check if there is enough PP (Power points to use the move)
   1. If not, return (turn is forfeited)
2. Determine if the Pokémon attacking can use move despite status condition, **canUseMove**()
3. Determine if the move will hit the target (accuracy check)
4. Calculate the damage
5. Determine the type of move
   1. Damage moves
      1. Determine the type multiplier
      2. Determine the damage
      3. Determine the attack type (Physical or Special)
         1. Physical modifier
            1. Flinch
   2. StatModifierMoves
      1. Check if the max change has been reached
         1. Perform the move
6. Check is move can apply status
   1. Check if a status is already applied
      1. Get the move chance and calculate success
7. End Function

# UML Diagrams

## Trainer

## Move Tree

* Move (Abstract)
  + AttackMove
    - PhysicalAttack
    - SpecialAttack
  + StatModifierMove
  + WeatherMove
  + StatusMove

## Interfaces

* Move Package
  + Target\_self
  + AccuracyModifier
  + EvasionModifier
* Statistic\_modifier Package
  + \*All statModifiers have four versions (MinusOne, MinusTwo, PlusOne, PlusTwo)
  + Attack
  + Defense
  + SpecialAttack  
    SpecialDefense
  + Speed
* Status\_moves Package
  + ApplyBurn
  + ApplyConfusion
  + ApplyFlinch
  + ApplyFrozen
  + ApplyParalyze
  + ApplyPoison
  + ApplySleep

# References