Pokémon Battle Simulator

Group D:
Nicolás Sousa González , Alejandro Mateo Costa de Dios, Javier Bilbao Lima, Juan Pontón Rodríguez

Game description

- 1 Player VS AI
- Teams of 3 Pokémon
- Turn-based decisions
- 4 moves per Pokémon + the option to switch Pokémon

Objective: Deplete the health points of the entire opposing team

Damage modifiers

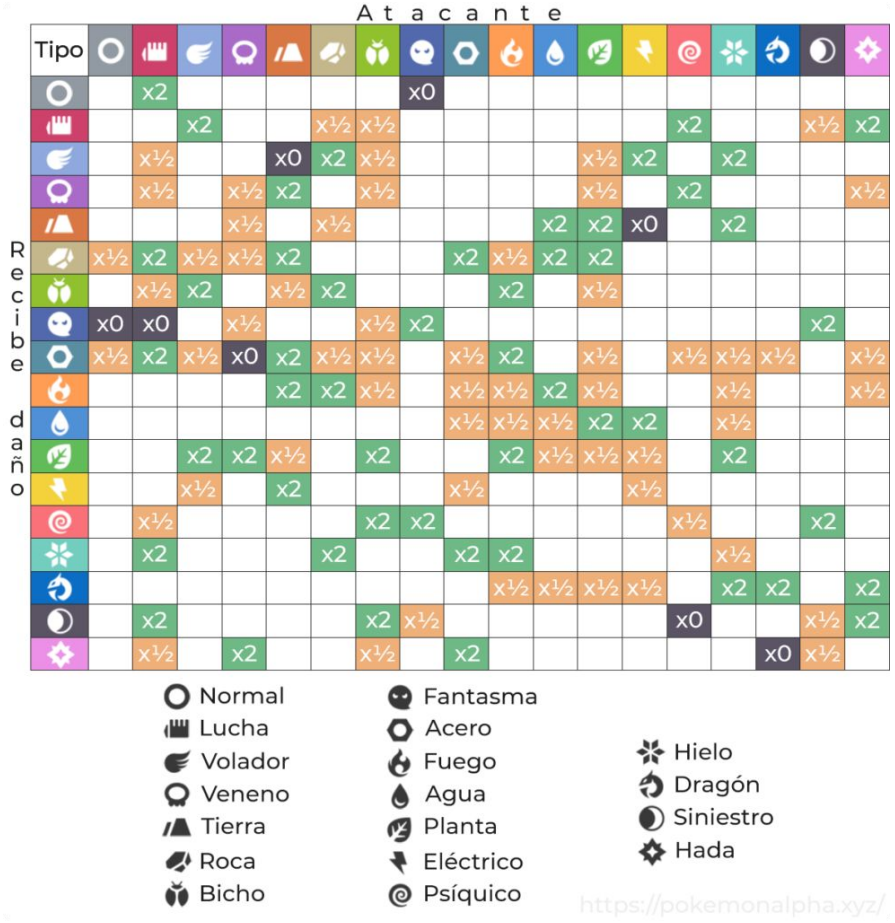
-Stats

-Movement power

-Type effectiveness

-Stab

-Critical damage

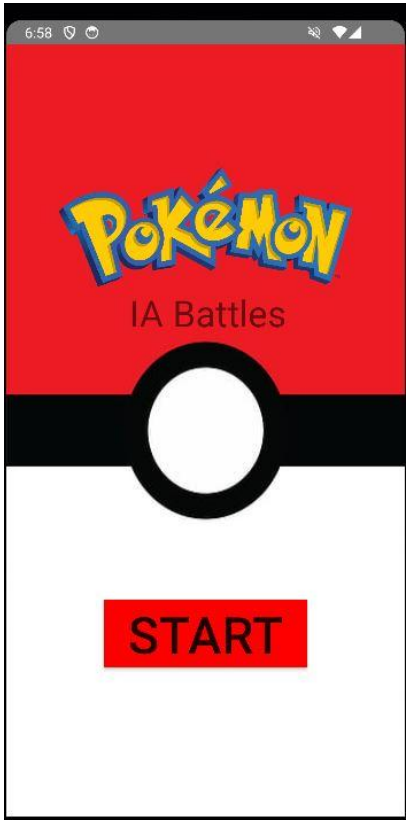


App Overview

Main Screen

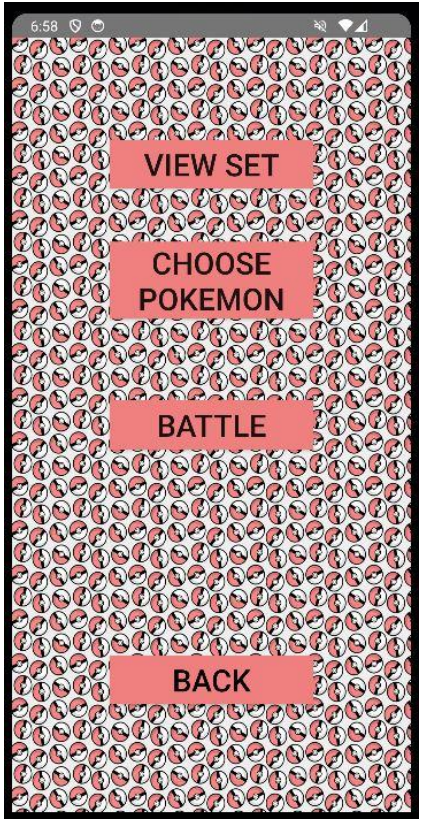
Welcomes you to the app with a great song!

Press start and enjoy.



Options Screen

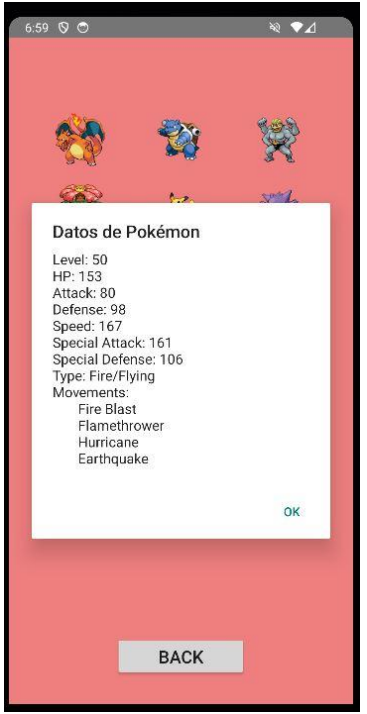
Shows the different possibilities that the app can offer to the user.



Set Screen

Allows the user to see the set of Pokémon available in the battles!

You can consult each Pokémon stats and moves anytime to plan your games.



Choose Screen

Instead of fighting with a randomly selected team, you can choose your 3 Pokémon before starting the battle.

Choose carefully!

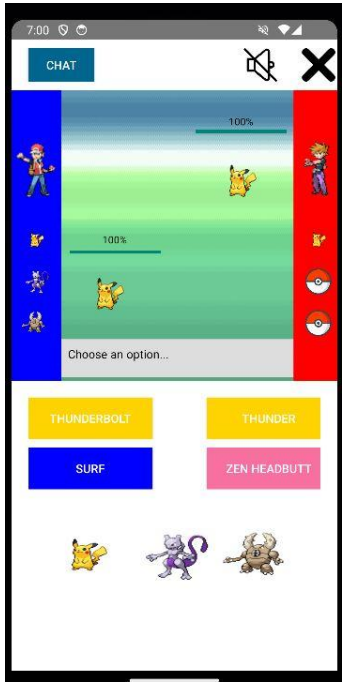
Battle Screen

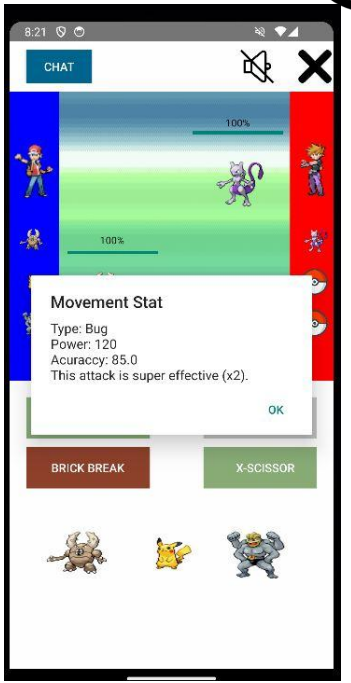
The Battle Starts!

Choose your move or switch depending on your opponent.

You can check your movements stats and grade of effectiveness against your rival (x0,x0.25,x0.5,x1,x2,x4)

You can also cancel the battle or mute the music anytime.





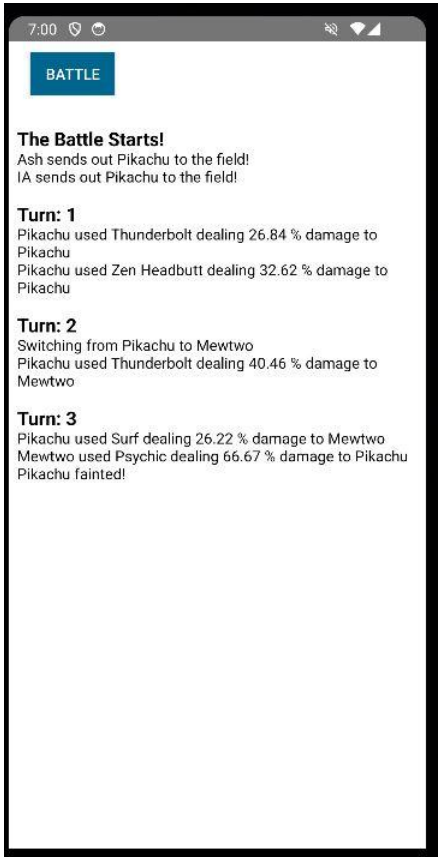
Chat Screen

Consult the turns of the current battle anytime in order to consult different relevant information:

-Damages done and received

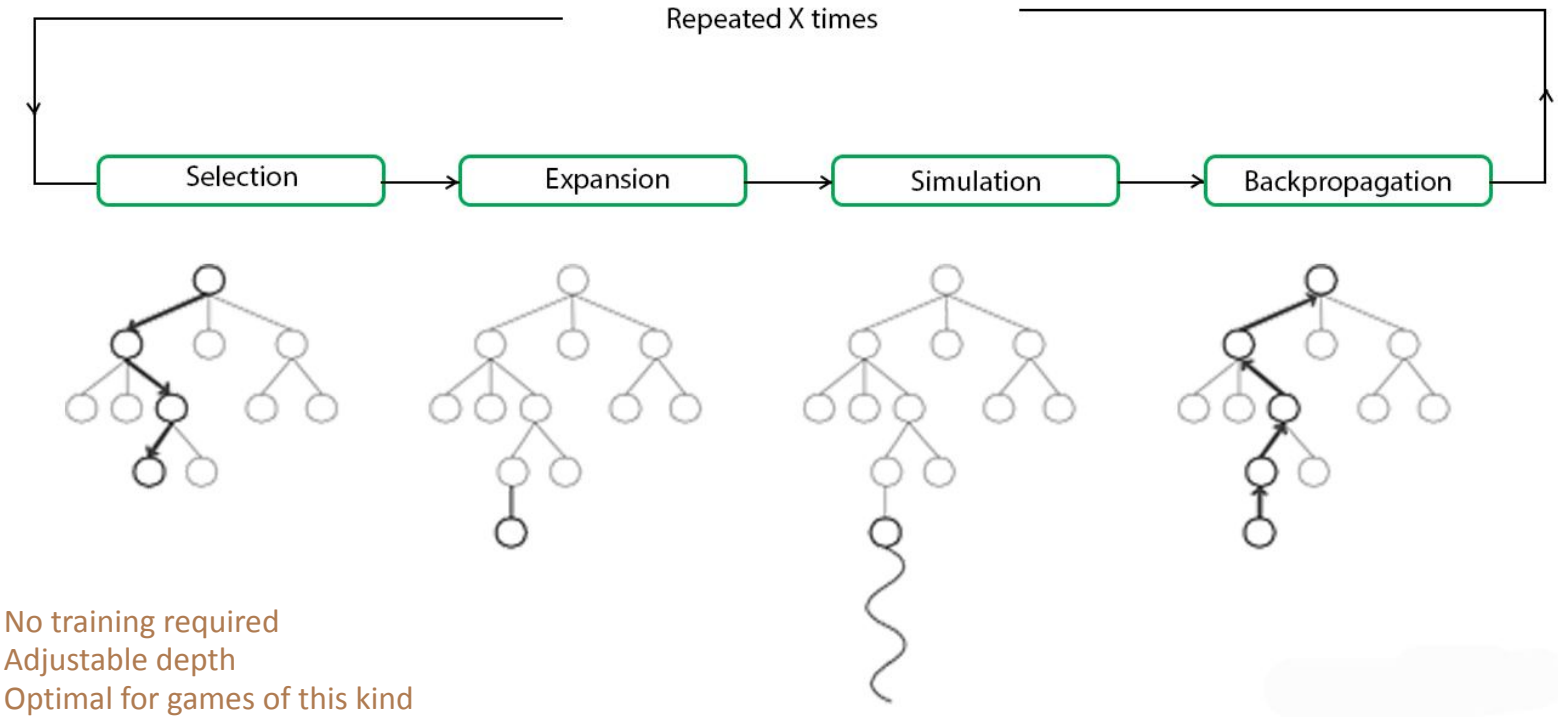
-Pokémon switched during the combat

-Pokémon fainted



AI Operation: MCTS

Monte Carlo Tree Search



- No training required
- Adjustable depth
- Optimal for games of this kind
- Intensive exploration

MCTS Implementation

Two main classes: MonteCarloTreeSearch and PokemonBattleState.

PokemonBattleState Class:

- Represents the game state, players, and turn information.
- Methods for simulating actions and score calculation.
- Internal methods handle Pokemon selection and effectiveness calculations to calculate the score of the node.

MCTS Implementation

Two main classes: MonteCarloTreeSearch and PokemonBattleState.

MonteCarloTreeSearch Class:

- Parameters set for simulations, repetition, and exploration.
- findBestMove method initiates tree building using MCTS.
- Node class internally represents nodes in the MCTS tree.
- Uses PokemonBattleState objects to create nodes and simulate the actions.

MCTS Implementation

Score

```
public double getScore(PokemonBattleState state) {
    double diffHealth = player2.getTeamHealth() - player1.getTeamHealth();
    double mediumPoints = 50 * ((double) ((state.getPlayer1().getRemainingPokemons() - player1.getRemainingPokemons()) - (state.getPlayer2().getRemainingPokemons(

    double battleOver = 0;
    if (Battle.isBattleOver(player1, player2) != 0)
        battleOver = Battle.isBattleOver(player1, player2) == 1 ? -200 : 200;
    double finalPoints = mediumPoints + battleOver + diffHealth;

    return finalPoints;
}
```

```
return score * node.state.getPlayer2().getCurrentPokemon().getMoves()[node.action].getAccuracy()/100.0;
```

MCTS Implementation

Overall Flow

- The app core reaches the action choosing moment and creates a MCTS.
- MCTS builds a tree by simulating gameplay.
- Nodes track state, visits, and scores.
- Tree traversal, expansion, simulation, and backpropagation implemented.
- AI selects the best move by selecting the best node child.
- The best child is selected based on score, visits, exploration and exploitation parameters.

Demo