# The invasion of Pokémon

## By: “team name doesn’t matter”

Team members:

Nghia Duc Hong – 292119 – duc.hong@tuni.fi

Duy Anh Vu – 294381 – duy.a.vu@tuni.fi

### 1. Introduction

The invasion of Pokémon is a project given by TIE-02408. The game is implemented by an existing system in which buses and passengers and bus stops are provided by course staff. The work for team is to implement our own game rule, GUI, and how player and Pokémon reacts with our own city.

Our team has implemented the minimum, intermediate requirements and top-grade requirements, as well as some additional features and some extra work which qualifies bonus points. Section 2 describes the software’s architecture and classes. Section 3 describes the implementation of extra features and extra work. Section 4 mentions to workload division. Section 5 is details of gameplay, section 6 contains some known issues, and section 7 is the conclusion of documentation.

All additional documentations such as game rules and backstory are located in Documentation folder. Doxygen is also there.

### 2. Software architecture

The game is implemented by course staff and our team. All the implementations by the course side are located in the namespace CourseSide. Our implement use namespaces: Model, Ui, Utils, and Stats.

**Project Structure of classes implemented by course staff:**

- Actor:
  - Nysse: bus model. This is a vehicle running along the roads of Tampere.
  - Passenger: passenger model. Passengers can enter stops and from there, they can get on the buses.
  - Stop: represents bus stops. Passengers will need to get inside the stop before they can get on a bus.
- Core
  - Logic: game logic engine. This contains all logic operations regarding buses and passengers and stops as well as timing.

    - Location: represent the locations of all the items.
- Error
    - GameError: exception when something goes wrong in the game.
    - InitError: exception when something goes wrong in the initialization.
- OfflineReader: read offline data related to buses and stops.

**Project structure of classes implemented by our team:**

- Model:
    - Pokemon : one of our actors: Pokémon. It hides inside a pokemon ball, which is shown on the city map.
    - Player: main player who can move around the city by user's control.
    - Character: This stores the graphic item for any actor on the map. For instance, buses, bus stops, passengers, and pokemon balls use this class to have something to show on the city map.
- Ui:
    - MainWindow: our gameplay main window.
    - Dialog: represent instruction dialog and adjustment of game setting.
- Stats:
    - Statistics: represent real-time number of buses, passengers, and scores.

Resource: - Instruction text;

- Actor avatars: for bus, passenger, stop, pokemon, pokemon ball, and main player

- Pokemon data

- Leader board scores

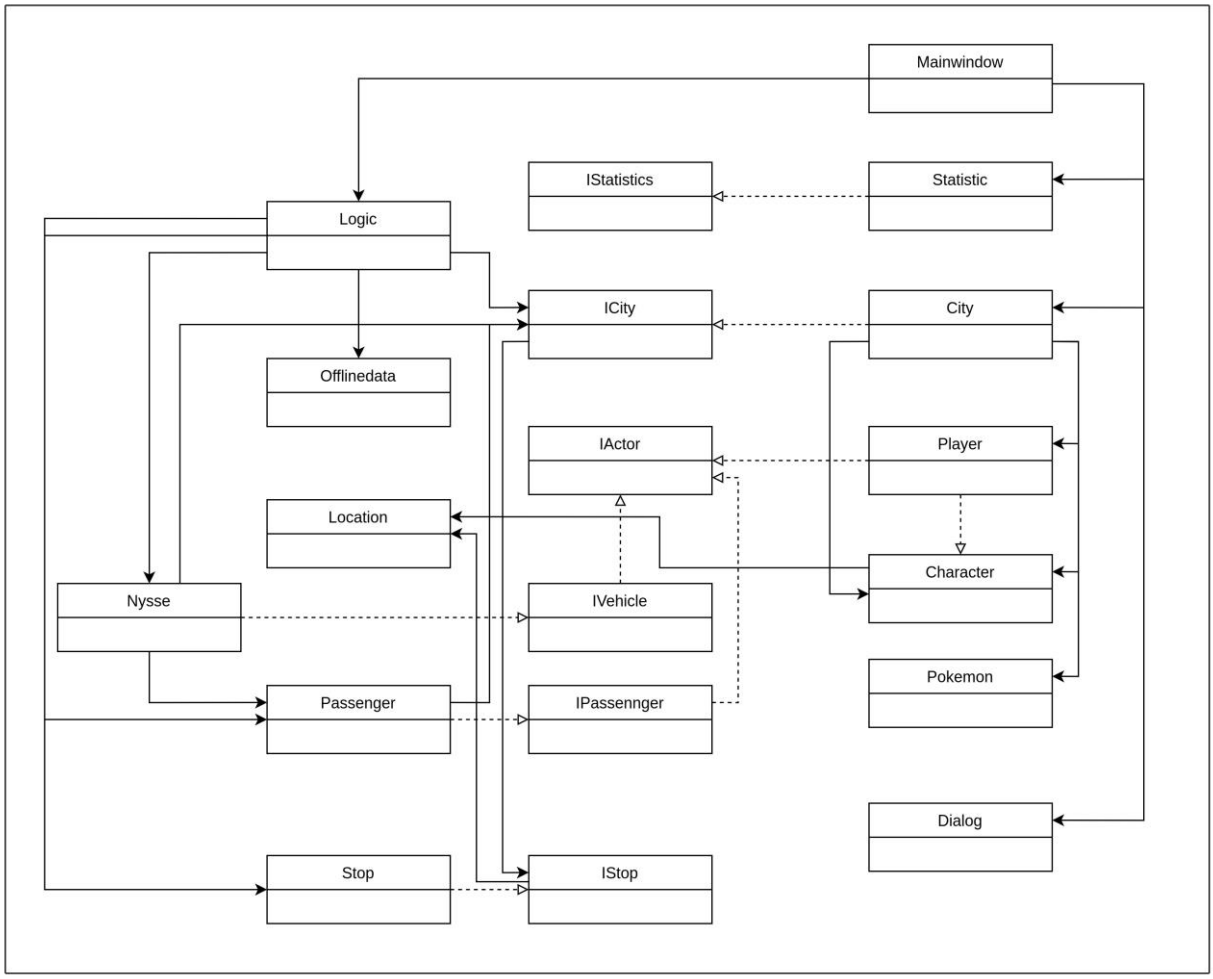


**Figure 1**: Class diagram

### 3. Extra features and works.

With the minimum and intermediate requirements, we have also implemented the following extra features:

- High score system: Display the top 10 scores of all time in the game. The game displays game high scores when the game is over for the user to compare their score with others. Also, users can click on Leaderboard button on mainwindow to open leaderboard anytime.
- Statistics are displayed in the mainwindow with real time update: Score and how many buses and passengers are in the city.
- Update player figure. Every time a rare Pokémon is collected (its score > rare index), the player would be regenerated back some energy. The energy bar would be updated as the player moves. The less a player uses the bus, the higher the energy consumption he has. This then leads to energy exhaustion.
- Multiplayer: another player can be played at the same time with the main player. Multiplayer must be set as “Yes” in game setting before the game starts.

- The figures can interact with others: 1. The player can go to a bus stop, 2. Waiting for a bus, 3. gets on the bus, 4. leaves the bus, 5. catch the pokemon when player move to pokemon ball location. The player can enter a stop when he/she is close enough to one stop. If any bus goes to this stop, the player would get on the bus automatically. Whenever the player presses space, the bus drops the player. Moreover, if the bus moves outside the map, the player would automatically leave the bus. When the player moves to a pokemon ball, he/she can catch this pokemon, adding this pokemon to his/her bag.
- Additional: We did not implement the scroll map but the larger map. The larger map needs some handling of player/buses/stops' location and conversion of some features. In addition, we also make an inventory for the player that the player can see how many and what type of Pokémon he caught.
- Passenger amount: If the player hovers the mouse on the bus/stop, the number of passengers in the bus/stop will be shown as a tooltip.
- Even screen update: This is implemented when the player is on the bus. The player location would be updated in real-time with the bus.
- Minimal screen update: The actors outside the map view will not be updated in the scene. This saves a lot of performance as refreshing scene is much slower than pure setting new location.
- Game story has been developed and created logically, with related available figures has been given by the course, the player is connected with those entities to make the game not to be boring but interesting with various types of Pokémon.

**Navigation diagram:**

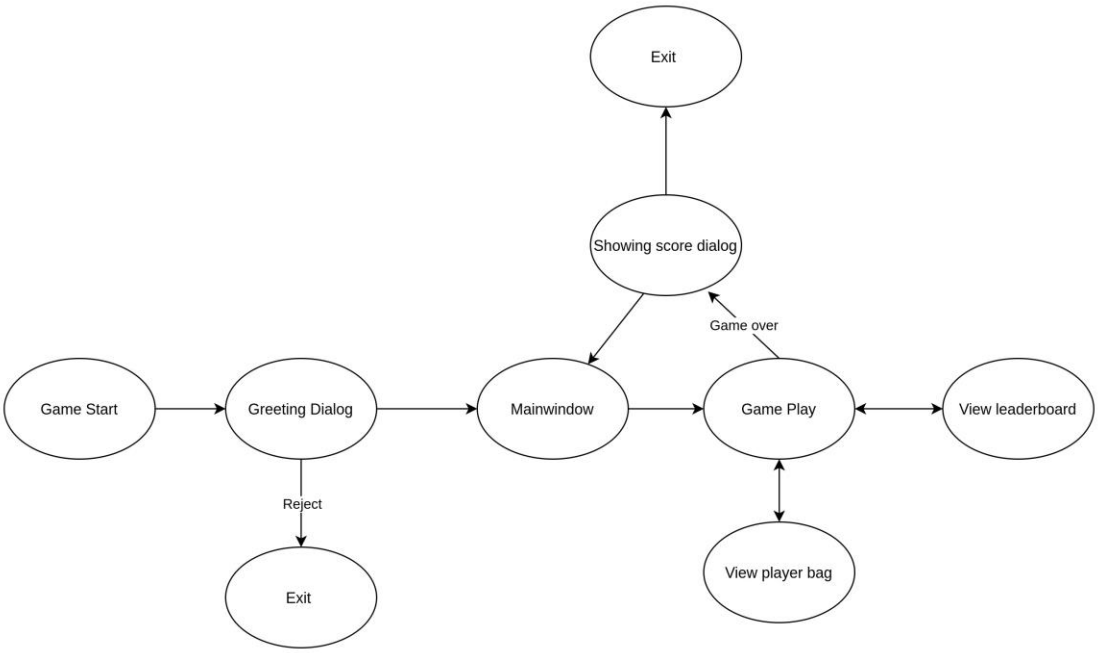

**Figure 2:** Navigation diagram

## 4. Workload division

In the beginning, we had meetings to discuss how the game is designed and investigate the code written by staff. Then we sketched some first designs on architecture and thought about how things would be implemented. Most work divisions are based on functionalities for the game.

Duc Hong:

- Make mainwindow UI, connecting signals and slots for classes
- Make Dialog UI, implement the game setting
- Find resource for Pokémon, read Pokémon data, and UI for showing dialog when a pokemon is collected.
- Make UI for Player Fuel bar
- Make player is movable with keyboard and can interact with buses and stops
- Make leaderboard list and display when the game is over.

Duy Anh Vu:

- Make buses/passengers move in real time.
- Implement Character class
- Convert the small map to larger rendering movable buses and stops on city map
- Handle catching pokemon process, add pokemon to the bag
- Build UI for player inventory
- Implement Statistics for the game
- Write Unit tests
- Handle exceptions

Both:

- Design logic and game flow.
- Implement City class
- Make data structure storing figures (players, buses, stops, balls)
- Fixing bugs and crashes
- Write software documentation.

## 5. Game Manual

**Game starts:**

The User are greeted with basic game story and clear instructions.

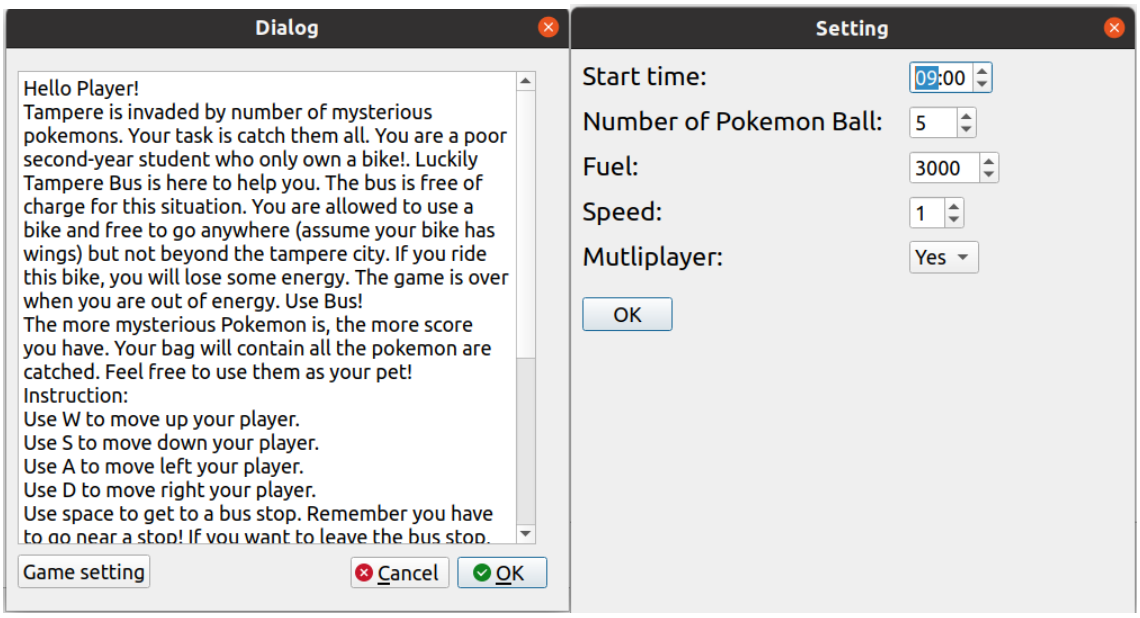


**Figure 3:** Gameplay greetings with instructions and settings.
(Note: this figure is extracted from own computer with QT:5.14.2)

If the player does not want to use default game setting, he can press the game setting, another window is opened, and player can set some game setting: The start time indicate what time at the moment when the player presses start. The number of Pokemon balls is how many balls there are available to catch on the map in one instant. The fuel is the sum of fuel both players have. Speed is how fast the players move on the map. Multiplayer option: Start with 1 player or 2 players.

After accepting the setting and game instruction, the player would be navigated to mainwindow.


**Figure 4:** Gameplay City.

When a pokemon is collected:

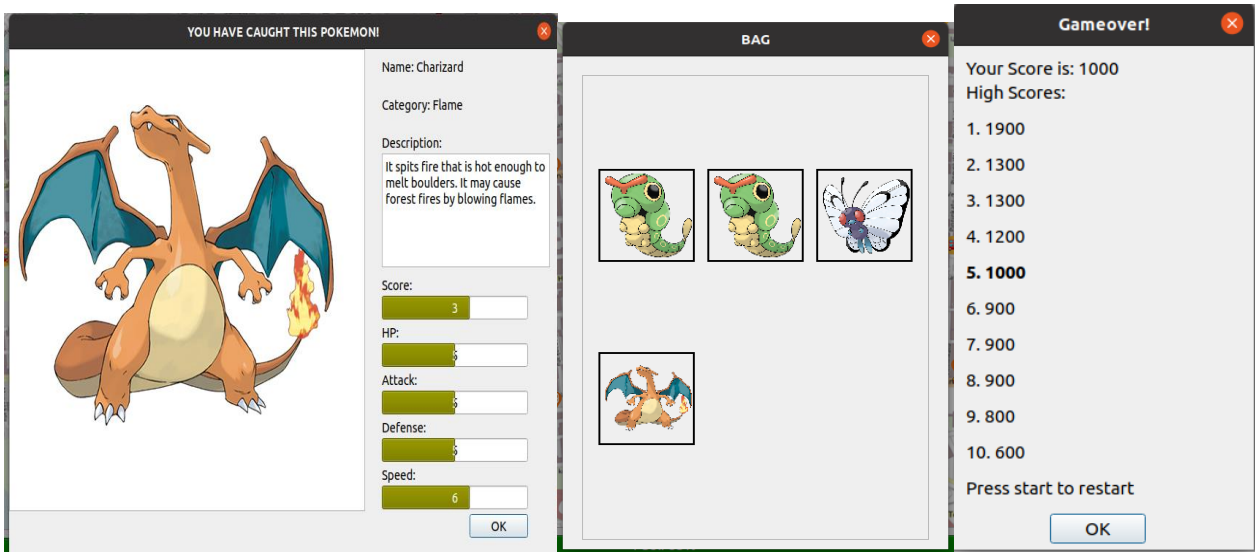


**Figure 5:** Pokemon Dialog

## 6. Known issues and notes

- The text font varies from different screen resolution and scaling. This is due to Qt does not automatically rescale, even QtCreator has problems with this. This means there are some cases that some characters of the labels are missing.
- The program forbids the player to go outside the map.
- The score file is stored inside Game/Text folder and reading file assumes the build directory to be set default. If you run the program in debug mode, please do not change the build directory. Otherwise, the top 10 feature would break.
- In multiplayer mode, Qt, for now, would not capture multiple key inputs. This mean, if one player holds a key, the other player cannot move. We suggest users to press and release the keys instead of holding them.
- Hovering might take 1 or 2 second for the tooltip to show. Follow the cursor to the running bus.

## 7. Conclusion

At the moment, there are no game bugs at the time writing documentation. Longer time of game testing may reveal bugs if there are any.

The game can be developed in the future such has player can fight the Pokémon when he meets the pokemon and so on. Due to logic provided by the course side and limited time, these are not implemented. There is a lot of room for improvement, but further improvement needs time and work.

The project is implemented by Design by contract technique that the course provides interface and our team use it.