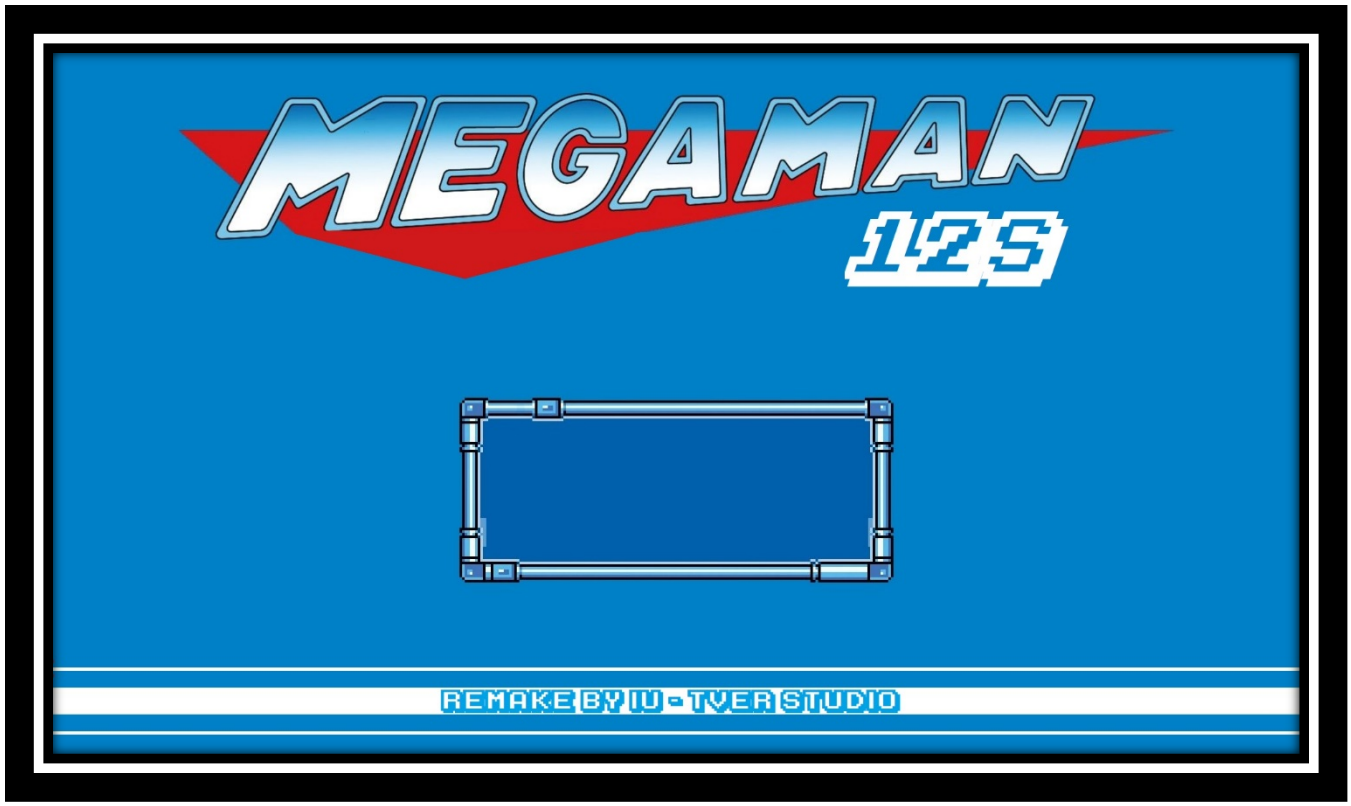


OOP GAME REPORT

Game project based on the Object - Oriented Principle

Members:

1. Lê Nguyễn Bình Nguyên – ITITIU19169
2. Lê Triệu Long – ITDSIU19044
3. Đặng Nguyễn Nam Anh – ITITIU19076

Demo day
4/1/202

I. INTRODUCTION:

The project is about a remake version of the game: Megaman which is a famous game in 1990s. Our project will simplify many features with the help of Unity.

This report will show you the brief and overview of our game, there are totally 6 parts:

- Section I: Introduction
- Section II: Github link
- Section III: Game rules
- Section IV: Class diagram
- Section V: Subjective assessment
- Section VI: Future improvements

II. Github link:

Click here to move on to our GitHub link which will help you understand clearly about our game

III. Game rules:

You will play a role: "*Megaman*". Player will use 4 buttons in computer's keyboard to control the character:

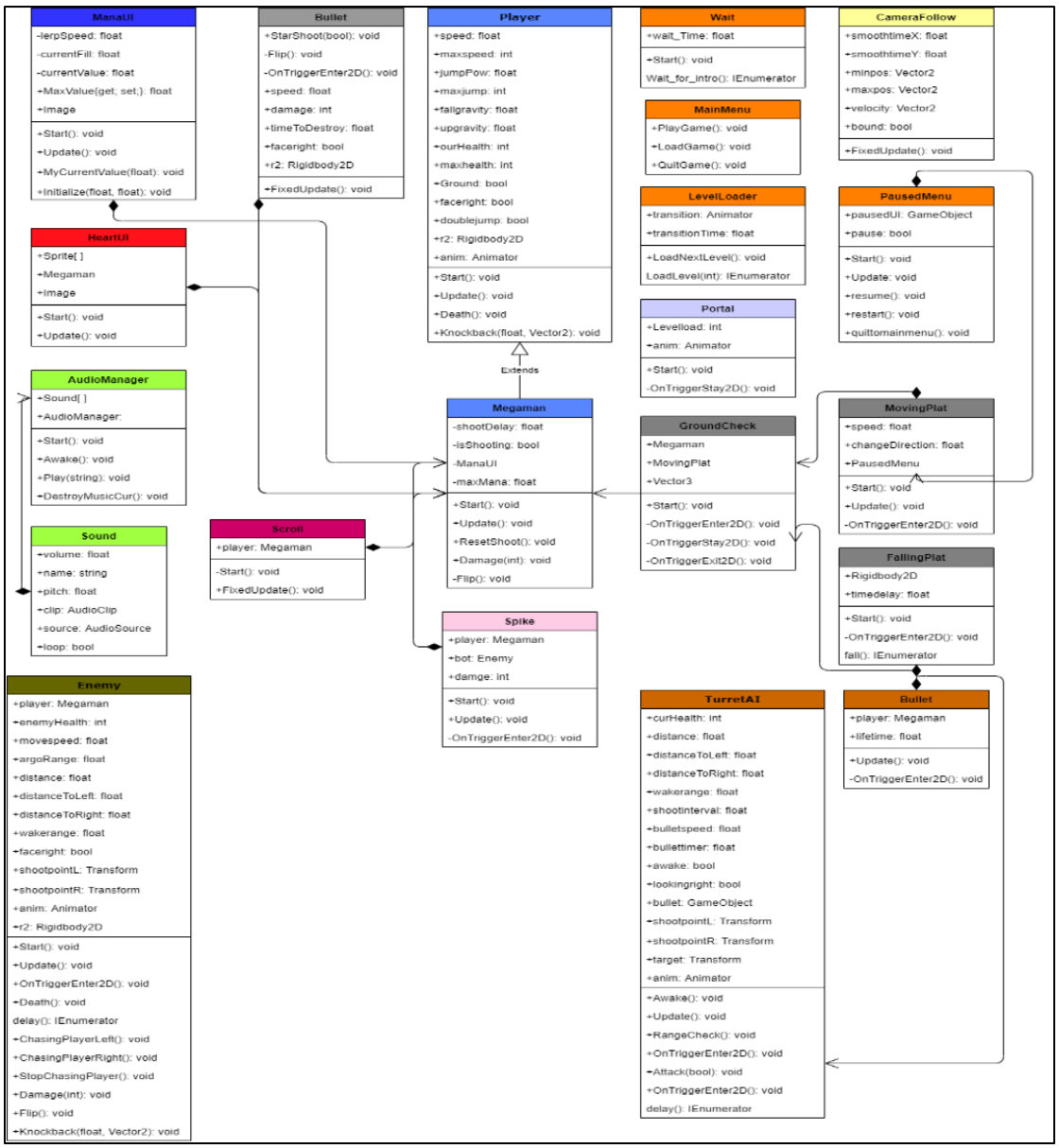
- **SPACE**: to jump
- **LEFT ARROW**: to turn left
- **RIGHT ARROW**: to turn right
- **E:** to attack the obstacles by Megaman's bullet
- **R:** to refill the Megaman's bullet
- **F:** to refill the Megaman's health

Player's main missions are overcome the obstacles by attacking or dodging them to set the highest record. You must finish it to move on to the next stage and the difficulty of game will increase with each stage.

Player will start with 10 HP and 5 MANA (which corresponding with each Megaman's bullets) it will be deducted when attacked by obstacles, for detail:

- Spike: deduct 1 HP
- Turret's bullets: deduct 1 HP/each bullet
- Enemy: deduct 1 HP
- Falling off the ground: counted as death

IV. Class Diagram:



V. Subjective Assessment

1. **Strength:**

- Uniform access principle:

+ Getter/Setter: Let the project have the continuity of the theory of Uniform Access, controls for the important properties that we often use and require exact values. Besides, using these two methods makes code maintenance easier.

- Inheritance: inherit and reuse properties and methods based on the old class. Besides, inheritance is exploited to minimize the boilerplate codes.

2. **Mistakes**:

- Overusing tools of Unity Engine: Unity already has a lot of useful tools for making game, so we did not apply much learned knowledge such as GUI, Abstract/Interface Class, …

- Shortages of Design Pattern: due to the simple game we made, the lack of Design Patterns can still make the game work properly, but this does not response the rules of the OOP principles, which can cause the difficulties of expansion for the code.

VI. Future Improvement

As this is the first time we develop a game, there are a lot of mistakes and shortages. To make this game perfectly, we think these criteria can improve our problems:

1. **Expand the game:** In our game, there is only one scene for players' gameplay, so that in the future, we tend to expand our game with many rounds and levels to improve the players'

experience. Besides, we want to create a Boss in the end of each round to create the difficulties and stimulate players.

2. **Create save station:** There are many rounds to play, so we must have some save stations to make players reassured about their records. The mechanism is that players will walk through over the save station to save the game. There are two new classes: SaveManger and CheckPoint, which will support us to create it. Besides, we use GUI to make a friendly client for players to access “LOAD GAME”.
3. **Improve the code:** We will apply more Design Pattern and OOP features such as Abstract Class and Interface. Besides, due to the existence of some small bugs in our game such as attack bugs and jump bugs which we have not figured out the solutions.