**OOP PROJECT REPORT**



**Our Group Members:**

* Võ Ngọc Duy Nguyên ITITIU19167
* Nguyễn Hữu Luân ITITIU19027
* Nguyễn Thiên Bảo ITITUN19014

DEMO DAY:

4/1/2021

1. **Introduction:**

In this report**,** we introduce our project which is inspired by the Megaman X series by Capcom. This is a well-known classic game. In this mini-game, we make our version in which main character will collect objects as known as health and escape of cave. During this task, he must dodge enemies, which take the shape of the machine bat, by his abilities. Hence, we will provide a brief and concise overview of our game through the following sections:

* Section II contains the link to our GitHub
* Section III is the tutorial about this game
* Section IV shows the design explanation
* Section V illustrates design patterns and principles we used
* Section VI is the conclusion and seft-evaluate

1. **GitHub repository:**

Here is our link GitHub: https://github.com/stackTsi/MegamanXDemoProject

1. **Game rules:**

Megaman X will appear at starting point and player will control him to collect total 6 heart for winning. During this task, enemies will appear and chase X. If collision happen, X dies and the game will be over.

# However, X will have abilities such as jump (X) and dash (Z) to run away and also combine both (Z+X), beside moving left (←) and moving right (→).

1. **Design explanation:**

To design this game, we have basic classes to represent the function of the game:

* **Class**: **Enemies** represent for bat machine and heart in this game, in this class we provided 4 functions for it:

**New**(): the initialize method, which used to add animation frames, position of enemy in the map,…

**Idle**(): idle state for enemy, always check if see player for every frame.

**Chase**(): if player is not found, the enemy will stand still. Once the player has been seen in the same screen, it will get the position of the player through the seesPlayer bool in the checkEnemyVision method in PlayState class and chase the player.

**Update**(elapsed:Float): called each frame of the game, used to play animation or update movement of enemy

* **Class**: **Gameover** have 3 functions

**Create**()**:** to create board which has “ Game Over” text in center and button play again under text.

**ClickPlay**(): switch Menu screen to game screen

**Update**(): goes through each frame and calls update on everything added.

* **Class:** HUD include background, heart counter and heart icon

**New**(): to create health counter, health icon and background scroll following player

**UpdateHUD**(): track task, background.

* **Class: Player** represents the character which we will control.

**New**()**:** provide new animations, hitbox setting, object’s physics.

**Update**(): call each frame and update on movement, animation,…

{

**UpdateMovement**(): it will update movement and animations according to counter when time elapsed

}

**Class: \_ MHCores :** represents the health cores for the player to collect.

**\_HUD:** the User Interface of the game.

For method actions and **Gameover** condition, we provide in class **PlayState**() and we also have class for main screen.

All class above was applied OOP features such as: encapsulation, abstraction, inheritance, polymorphism. For example, Many class above extend from **FlxState** **FlxSprite,** etc**.** from the HaxeFlixel’s framework.

Finally, the tools we used for this project:

* Languages: Haxe: this is an open source, high-level cross-platform programming language can build cross-platform applications targeting JavaScript, C++, C#, Java, Lua,PHP,…
* Game engine: HaxeFlixel
* Library: Haxelib
* Software framework: OpenFL
* Tile-map design: Ogmo Editor

1. **Design patterns/Principles:**

The following design patterns and principles are applied in this project, an explanation for their strengths is in Section V.

Fully implemented patterns/principles

* KISS philosophy
* Strategy design pattern

Partially implemented patterns/principles

* Defensive design
* Discoverability principle
* Don’t Repeat yourself principle
* Uniform access principle
* Minimal Agile model

1. **Evaluation:**

Overall evalution

Although this project seems basic with easy-accessible it used all of principles in OOP which are data encapsulation, abstraction, inheritance, polymorphism. Besides, it also has many classes which can provide interaction with some of the others and also has some methods, attributes in their class for supporting mechanic of game.

Moreover, because of using Haxe language it make this project become flexible to edit and develop in the future. The interface makes this game easy to play just after a few trying.

And finally, our code compliance designs patterns\principles so it will make us interact with others project, coders around the world.

Subject to improvement:

* Lack of deep mechanic and simple for long term playing
* Have some bugs during playing
* Using non popular languages

First issue, we are trying to recreate the same physics and conditions from the series Megaman X which is a super game project from Capcom, so we lack of experience, but in the future we will try to improve this game to completion with proving more mechanic, graphic works, stages and game rules.

Secondly, some bugs happen in the testing in demo process(hitbox observing-related issue), however we are fixing it and will have complete launch for this game soon.

And finally, although HaxeFlixel is a very flexible engine, but because of its unpopularity which can make it very difficult for beginners to approach.