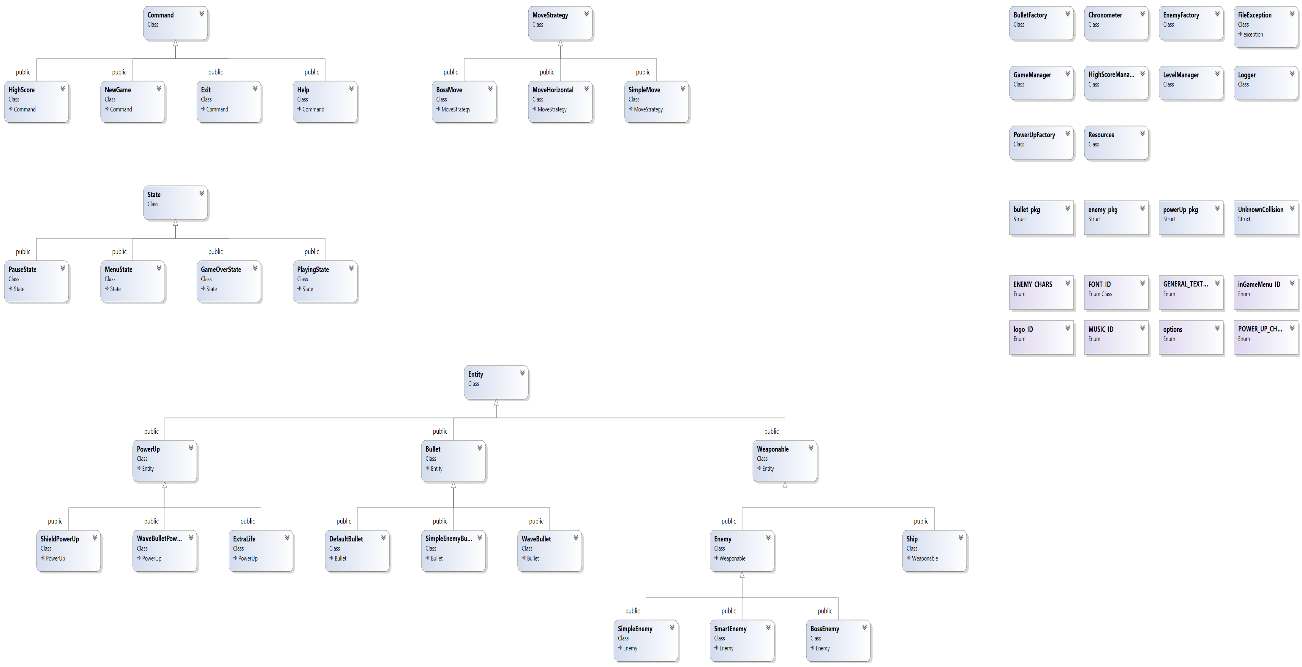
OOP 2- Final Project

Chicken Invaders

Ron Efraim 204661425

Sahar Itzhak 308389485



* Game Explanation:

You the user are a spaceship, you move left and right using the arrow keys,

And shoot using space. Your mortal enemies are weird looking squids and crabs.

The goal in the game is to get to the boss level, the evil squid monster.

The simple enemies move as a group (an army).

The boss squid has a lot more health and moves in a different way.

* Files and classes:

1. Main.cpp

The main file that runs the start of the GameManager, pushes a menu state.

1. Macros.h

The file holds all defines and consts, enums, structs as well as usings and includes used throughout the program.

1. Resources

The resources class is in charge of loading and holding all outer files,

Such as pictures, music’s and fonts.

It is a static Singleton class.

1. Chronometer:

A namespace which is an expansion of sf::clock, it has an addition of a pause and resume function which allows us to pause and resume the game clock when we pass between states.

1. Logger

A singleton static class, the class holds the logger file. The logger file will hold times of major things that happened throughout the program. such as Exceptions, collisions and more.

If the program has ever failed, the logger file will allow us to know the reason.

1. Bullet, Enemy, PowerUp factories

These classes are each in charge of creating objects.

The class holds a static map which holds a key matching to the proper objects and a pointer to a function which creates the object. Each object is in charge of registering itself to the factory.

1. Level Manager:

The level manager gets the vector of entities and updates them by using the outer level files. It creates the objects according to the char in the file using the factories.

1. HighScoreManager:

The manager is in charge of getting the file if it exists and creates a new one if it doesn’t. it adds all the existing high scores to a multimap.

1. GameManager:

The gameManager runs the main game loop.

It holds a stack of states like menu, help, highscore, pause, play

Generically it calls a set of functions for the state that’s on the top of the stack.

It has a push and pop function which are called when we want to move between game states.

1. MyExceptions:

Holds all the exceptions needed throughout the game, the exceptions inherit from std::exception.

1. **State**

State is an abstract class that represents possible game states.

1. Menu state:

The class inherits from state, represents the main menu state.

It is the first state pushed into the stack of states.

It holds a vector of pointers that inherit from command,

Help, start, exit, highscore.

1. playingState

This state is the playing state, it holds the vectors of entities and is in charge of all playing updates, like movement, shooting, collision, deletion of objects….

1. PauseState

The state pauses all game clocks…

1. GameOverState.

The state is in charge of ending the playing state.. cleaning everything getting users name and adding it to the multimap of high scores.

1. **Command:**

Abstract class that represents all buttons in games, has an execute function

1. Exit

Inherits from command, exits the game…. Writes high scores to highscore file.

1. Help:

Inherits from command, opens a help screen

1. HighScore:

Inherits from command, loads from multimap, makes sf::Texts and draws them on the window.

1. NewGame:

Pushes the playingState.

1. **MoveStrategy:**

Abstract class that represents the enemy’s movements.

1. SimpleMove

Inherits from MoveStrategy, is the default move for the simple enemies.

Moves them in an army motion.

23.HorizontalMove

Inherits from MoveStrategy, this move is given to the enemy that was just   
 teleported in front of the ship.

1. BossMove

Inherits from MoveStrategy and moves the boss

1. **Entity:**

Abstract class which represents a game object.

1. Bullet:

Abstract class represents a bullet

1. SimpleEnemyBullet:

Bullet for enemy

1. DefaultBullet

Default hero bullet

1. WaveBullet

Bullet for hero gotten after a powerup

1. **Weaponable**

Abstract class represents an enemy or hero

1. Ship

Inherits from weaponable, is the hero class

1. Enemy.

Inherits from weaponable, is abstract

1. SimpleEnemy

Inherits from enemy

1. smartEnemy

Inherits from enemy

1. BossEnemy

Inherits from enemy

26. **PowerUp**

Abstract class that inherits from entity.

27. Shield

28. WaveBullet

29. Life

* DataStructures:

1. Map:

All factories hold a map

1. Multimap:

For high score, is multimap because we want it sorted and allow the duplicate keys.

1. Stack:

GameManager holds stack of game states.

1. Vectors:

Entities, textures, sounds ……

* Bugs:

No knows bugs.