README file for the final OOP Project :

**מגישים:**

יבגני בורקו: 309543247

אלכסיי בוטוב: 317552610

אלכסנדר בורד: 319263836

בן קידר: 208308817

================================================================================

General Description of program:

The Main purpose of the Project is to build a fully functional PC Game "Jill of the Jungle" using the "SFML" Graphic Library.

The game itself is made according to our teem decisions so it's none like the original

but it has some cool features and abilities in our oppinion, we hope you will enjoy it.

You start by choosing a male or a female player character.

The main purpose of the game is to find all keys to open the door for the next level

before the timer ends and clear all levels until the end, there are some ninja weapons at your disposal in order to survive in the harash environment of the jungle.

The program defines the following classes:

Alligator, Ant, Apple, Bee, Bird, Board, Collision, Controller, Diamond, DynamicObject, Floor, Frog, Gate, GemGate, Item, Key, KeyGate, Knife, Menu, Monster, NinjaStar, Object, Player, Singleton, Snake, SpecialWall, Spikes, StaticObject, StatusBar, Vine, Wall, Weapon.

Object derived classes <= DynamicObject, StaticObject.

StaticObject derived classes <= Floor, Gate, Item, SpecialWall, Spikes, Vine, Wall.

DynamicObject derived classes <= Weapon, Monster, Player.

Gate derived classes <= GemGate, KeyGate.

Item derived classes <= Apple, Key, Diamond.

Weapon derived classes <= Knife, NinjaStar.

Monster derived classes <= Alligator, Ant, Bee, Bird, Frog, Snake.

Non inharited classes: Board, Collision, Controller, Menu, Singleton, StatusBar.

================================================================================

Design

We start by running the main function , it then runs the menu which runs the main function that runs the game window. Once we are in the menu window we choose between 4 options: 1) running the game, 2) browsing game controls 3) getting help or 4) exiting the menu. after the "play" option is chosen the game opens another menu to choose between 2 player characters "male" or "female".

the function returns a boolean value "play" if the value returns as "false" than the window loop ends and the program stops running, if we choose "play" it returns true and passes it to the main loop which opens the game window, the first level is loaded from a file trough a function in Level\_Upload class and all the game stats are set and the game starts.

Once the player tries to collde with a wall, the movement is stopped, if it collides with a Diamond, apple or Preasent, the object is erased from the staticObject vector and the player stats are updeted. if the player "ate" the defined number of keys in the game file, the door to the next level opens. if there is a diamond gate that blocks the way, it dissapears once the player found a diamond. the level keys and diamonds are defined in the level file, the player's life value is a constant value set to 3 and it's decreased by 1 for each time the player got hit or a timer gets to 0.

if there is a timer defined in the level file, once the time value gets to 0 the level is reset as well and the digger life value is decreased by 1. the timer collor is changed to Red if the number of seconds left is less than 11.

The file levels also includes monsters and all obstacles, they are defined with different symbools. after the level files are being inserted to the game vector, each symbol gets into a switch case statement and defined by unique ptr to later be used in the game.

to win the player needs to eat all keys and diamonds and go on to the next level until he clears all of them before it's life value is set to 0, the victory window is oppened and the player can choose to quit or play again.

if the life value is set to 0 the game is over and a game over window is opened to choose if the player wants to play again or quit.

All image, sound and string files are stored in Singelton class using singelton design patern.

================================================================================

List of files

1) Menu.cpp – the purpose of this class is to have a menu in which the player can choose if he wants to play, get help, view the game controls or quit.

afterwards another window opens to choose a player for the game.

it runs in the main function and turns on the game.

2) Controller.cpp - runs the game after the play option and player character is chosen, runs all controls, buttons and all the game objects, responsible for jumping, firing and to activate other game classes as well, and also it runs all nesting classes.

3) Board.cpp– the purpose of this class is to have a game board in which all objects will appear and interact with each other, it uses the collision class multi methods to activate collisions between objects.

4) Singelton.cpp – it's purpose is to load the images and sound files.

5) Main.cpp - the main function, it defines an object of menu class and runs the run function from it.

6) Collision.cpp - the multi methods map class, stores all collision functions in a map

and instanciates them only once when used like in singelton.

7) Object - the abstract base class to store game objects and their functions for polymorphism.

8) StaticObject - a derived class from the abstract base class "Object " to store the static game objects such as "wall, floor, gate, etc.." and their functions for polymorphism.

9) DynamicObject.cpp - a derived class from the abstract base class "Object " to store the static game objects such as "player, weapon, gate and monster" and their functions for polymorphism.

10) Player.cpp - a derived class from DynamicObject class, it defines the specific DynamicObject as Player and sets it's functions acordingly.

11) Monster.cpp - a derived class from DynamicObject class, it defines the specific DynamicObject as Monster and sets it's functions acordingly.

12) Alligator.cpp - a derived class from Monster class, it defines the specific Monster as Alligator and sets it's functions acordingly.

13) Ant.cpp - a derived class from Monster class, it defines the specific Monster as Ant and sets it's functions acordingly.

14) Bee.cpp - a derived class from Monster class, it defines the specific Monster as Bee and sets it's functions acordingly.

15) Bird.cpp - a derived class from Monster class, it defines the specific Monster as Bird and sets it's functions acordingly.

16) Frog.cpp - a derived class from Monster class, it defines the specific Monster as Frog and sets it's functions acordingly.

17) Snake.cpp - a derived class from Monster class, it defines the specific Monster as Snake and sets it's functions acordingly.

18) Weapon.cpp - a derived class from DynamicObject class, it defines the specific DynamicObject as Weapon and sets it's functions acordingly.

19) Knife.cpp - a derived class from Weapon class, it defines the specific Weapon as Knife and sets it's functions acordingly.

20) NinjaStar.cpp - a derived class from Weapon class, it defines the specific Weapon as NinjaStar and sets it's functions acordingly.

21) Apple.cpp - a derived class from Item class, it defines the specific Item as Apple and it's functions acordingly.

22) Diamond.cpp - a derived class from Item class, it defines the specific Item as Diamond and it's functions acordingly.

23) Floor.cpp - a derived class from StaticObject class, it defines the specific StaticObject as Floor and it's functions acordingly.

24) Gate.cpp - a derived class from StaticObject class, it defines the specific StaticObject as Gate and it's functions acordingly.

25) GemGate.cpp - a derived class from Gate class, it defines the specific Gate as GemGate and it's functions acordingly.

26) Item.cpp - a derived class from StaticObject class, it defines the specific StaticObject as Item and it's functions acordingly.

27) Key.cpp - a derived class from Item class, it defines the specific Item as Key and it's functions acordingly.

28) KeyGate.cpp - a derived class from Gate class, it defines the specific Gate as KeyGate and it's functions acordingly.

29) SpecialWall.cpp - a derived class from StaticObject class, it defines the specific StaticObject as SpecialWall and it's functions acordingly.

30) Spikes.cpp -a derived class from StaticObject class, it defines the specific StaticObject as Spikes and it's functions acordingly.

31) Vine.cpp - a derived class from StaticObject class, it defines the specific StaticObject as Vine and it's functions acordingly.

32) Wall.cpp - a derived class from StaticObject class, it defines the specific StaticObject as wall and it's functions acordingly.

33) Present.cpp - a derived class from StaticObject class, it defines the specific StaticObject as Present and it's functions acordingly.

34) StatusBar.cpp - the object that sets all sats of the level.

35) PresentLife.cpp - a derived class from Present class, it defines the specific Present as PresentLife and it's functions acordingly.

36) PresentScore.cpp - a derived class from Present class, it defines the specific Present as PresentScore and it's functions acordingly.

37) PresentShield.cpp - a derived class from Present class, it defines the specific Present as PresentShield and it's functions acordingly.

39) PresentTime.cpp - a derived class from Present class, it defines the specific Present as PresentTime and it's functions acordingly.

Also includes the headers for the classes above which is 39 more header files,

all the image and sound files

a font file

and 5 level files (from 1-5).

================================================================================

Data structures:

1) Controller: stores the player object and the vectors of Knives and NinjaStars instances, also it includes the SFML library into the game in addition it stores the game clock and the object of the game board.

2) Singelton: stores all the All image files and sound files In vectors, to draw on the game board.

3) Board: holds a matrix of the loaded file and the vectors of unique pointers to all the monsters in the level and all the level obsticles.

4) Collision: stores the map of collision functions to use in the board class for interaction between defferent objects.

5) StatusBar: holds the player status of the lives, keys, diamonds, player score, player lives.

================================================================================

Worth to mention algorithms:

Manhattan Algorithm for Smart monsters AI, used for Bee and Bird objects.

================================================================================

Known Bugs: ================================================================================

Other notes:

Must have a text file with a name of Level+(number from 1-5).txt otherwise returns an error because cmake needs a configured\_file.