README file for OOP - Project :

**מגישים:**

יבגני בורקו: 309543247

אלכסיי בוטוב: 317552610

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

General Description of program:

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

The program defines the following 22 classes: Controller, Board, Menu, Multimedia\_Files, Game\_Status, Level\_Upload, Demon\_AI, Game\_Object, Obstacle, Entity, Diamond, Wall, Stone, Present, Digger, Demon, Simple\_Demon, Smart\_Demon, Time\_present, Stone\_present, Score\_present, Speed\_present.

Game\_Object derived classes <= Obstacle, Entity.

Obstacle derived classes <= Diamond, Wall, Stone, Present.

Entity derived classes <= Digger, Demon.

Demon derived classes <= Simple\_Demon, Smart\_Demon.

Present derived classes <= Time\_present, Stone\_present, Score\_present, Speed\_present.

The Monster\_Ai class also holds a struct "Node" object.

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

Design

We start by running the main function , it then runs the controller function that runs the menu window. Once we are in the menu window we choose between 2 options: 1) running the game or 2) exiting the menu. 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 digger tries to collde with a wall, the movement is stopped, if it collides with a Diamond, Stone or Present, the object is erased from the Obstacle vector and the player stats are updated. if the digger "ate" the allowed number of stones which is defined in the level file, the diggers life value is decreased by 1 and all the level objects are reset from the level file, the number of stones allowed to eat is reset as well.

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 color is changed to Red if the number of seconds left is less than 11.

The file levels also includes demons, the number of monsters on the level is set by the symbol "!" in the level file. then a random function is used to set the demons to become smart or random demons, while the random demons move to a random direction, a smart demon is using the bfs algorithm to follow the digger wherever it goes. if the path to the digger is blocked by a stone or a wall, the smart demon waits until the path to the digger is cleared.

to win the digger needs to eat all diamonds of all the levels before it's life value is set to 0, the victory window is opened 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 in which the player can choose if he wants to play again or quit.

All image, sound and string files are stored in Multimedia\_Files class using singleton design pattern.

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

List of files

1) Controller.cpp - runs the menu window and the main function, runs alongside the other classes as well, and also it runs all nesting classes.

2) Menu.cpp – the purpose of this class is to have a menu in which the player can choose if he wants to play or quit.

3) Board.cpp– the purpose of this class is to have all the working functions of the game and to be in touch with the other classes and to report to the controller.

4) Level\_Upload.cpp – it's purpose is to load the characters from the level file.

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

6) Diamond.cpp - a derived class from Obstacle class, it defines the specific obstacle as diamond and it's functions accordingly.

7) Demon.cpp - a derived class from Entity class, it defines the specific entity as demon and it's functions accordingly.

8) Entity.cpp - a derived class from game object class, it defines the specific game object as Entity and it's functions accordingly.

9) Game\_Object.cpp - is a base class for all the game objects.

10) Digger.cpp - a derived class from Entity class, it defines the specific Entity as digger and it's functions accordingly.

11) Game\_Status.cpp - the object set all the stats of the player.

12) Simple\_Demon.cpp - a derived class from demon class, it defines the specific demon as simple demon and it's functions accordingly.

13) Multimedia\_Files.cpp - a singleton class which contains all multimedia files.

14) Demon\_AI.cpp - the "brain" class behind the smart demon logic.

14) Smart\_Demon.cpp - a derived class from demon class, it defines the specific demon as smart demon and it's functions accordingly.

15) Obstacle.cpp - a derived class from game object class, it defines the specific game object as obstacle and it's functions accordingly.

16) Wall.cpp - a derived class from Obstacle class, it defines the specific obstacle as wall and it's functions accordingly.

17) Stone.cpp - a derived class from Obstacle class, it defines the specific obstacle as stone and it's functions accordingly.

18) Present.cpp - a derived class from Obstacle class, it defines the specific obstacle as Present and it's functions accordingly.

19) Time\_present.cpp - a derived class from Present class, it defines the specific Present as Time Present and it's functions accordingly.

20) Stone\_present.cpp -a derived class from Present class, it defines the specific Present as Stone Present and it's functions accordingly.

21) Score\_present.cpp -a derived class from Present class, it defines the specific Present as Score Present and it's functions accordingly.

22) Speed\_present.cpp - a derived class from Present class, it defines the specific Present as Speed Present and it's functions accordingly.

Also includes the headers for the classes above which is 22 more header files,

all the image and sound files

a font file

and 5 level files (from 1-5).

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

Data structures:

1) Controller: holds hpp functions using SFML, array of objects, insertion to files.

2) Multimedia\_Files: holds all the All image files, sound files, font and strings to draw on the game board.

3) Level\_Upload: holds a matrix of the loaded file.

4) Board: holds the vectors of unique pointers to all the demons , digger and all the obstacles in the level.

5) Game\_Status: holds the player status about the allowed number of stones, player score, player lives, player basic speed, player current speed, and speed update value from the present.

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

Worth to mention algorithms:

BFS Algorithm for Smart Demon AI

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

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

Other notes:

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

\*for example, if you want to add a new level file , replace one of the existing files :

level1.txt or level2.txt or level3.txt or level4.txt or level5.txt

with the exact same name.

\*\*there is a time function for each level, you must add an integer for regular timer or -1 if you don’t want any time in the level.

\*\*\*The movement of the digger can be stopped by pressing space.