**Game: Save The King**

Students info:

name: Michael Basov ID: 315223156

name: Gal Ein-Dor ID: 209070671

Task description:

In this task we built a game called save the king. in this game you can add levels as .txt files in resources directory

and the game will read the levels, there are 3 pre-built levels in the game.

The purpose of the game is to bring the king to his throne.

The game contains the following characters:

King - needs to go back and sit on his throne, represented by char K

Mage - can put out fires that are on the board, represented by char M

Warrior - can kill orgs that drop keys, represented by char W

Thief - can pick up keys and open gates with them, represented by char T

Besides the characters, there are few objects across the board:

Throne - the king needs to sit on it, represented by char @

Fire - blocks the way of any character, the mage can put it out, represented by char \*

Org - powerful creature that can be easily be killed by the warrior, once dead, drops a key, represented by char !

Key - dropped after an org is killed, can be picked by the Thief, represented by char F

Gate - can be opened with a key, represented by char #

Wall - blocks every character in the game, represented by char =

Gnome - moving around the map and blocking the movement of other characters

Gift - can be collected by any playable character, there are 3 types of gifts: add time, decrease time, remove gnomes from map.

the game is time based, each level has a timer, if the time runs out the game ends.

The game was built using the SFML library.

List of files:

GameObject.cpp - an abstract class that has methods and members for each elemnt in the game

MovingObject.cpp - an abstract class that has methods and members for each moving object

StaticObject.cpp - an abstract class that has methods and members for each static object

King.ccp - contains the functions and the members used to control the King

Mage.ccp - contains the functions and the members used to control the Mage

Warrior.ccp - contains the functions and the members used to control the Warrior

Thief.ccp - contains the functions and the members used to control the Thief

Gnome.cpp - contains the functions and the members used to control the Gnomes

Fire.cpp - handles collision with characters and creates a Fire object

Gate.cpp - handles collision with characters and creates a Gate object

Key.cpp - handles collision with characters and creates a Key object

Ogre.cpp - handles collision with characters and creates a Ogre object

Teleport.cpp - handles collision with characters and creates a Teleport object

Throne.cpp - handles collision with characters and creates a Throne object

Wall.cpp - handles collision with characters and creates a Wall object

Gift.cpp - handles collision with characters and creates a Gift object

TimeGift.cpp - inherits from Gift and adds time to the game

BadTimeGift.cpp - inherits from Gift and decreases time

RemoveGnomesGift.cpp - inherits from Gift and removes gnomes from level

Controller.cc - controls the on-going of the game, controls the characters and the board, every thing happens in the game controls by this class

Board.cc - writes and displays the board, in charge of every change and update to the board that happens during the game

Button.cpp - creates and manages buttons across the game

Recources.cpp - a singleton used to load textures, sounds and more features

Caption.cpp - uses to put captions and text in the game

Menu.cpp - shows a menu screen when game starts, shows help, start game or finishing the program

main.cc - creates a Controller object and launches the game

// every .cpp file has a .h file with the same name //

levelList.txt - stores the names of the other levels in order to load them when needed

levelOne.txt - stores the first level board

levelTwo.txt - stores the second level board

levelThree.txt - stores the third level board

there are .png files for every object in the game, sounds for events in the game.

How to add new level:

The program uses .txt files to read from the design of the level, to create level u can upload a .txt file to resources directory

and add the name of the file to the list of CMakeList in resources and add it to the "LevelList.txt" file

Main databases:

We used vectors and unique pointers, one vector for moving objects and other for static objects

Special Algorithms:

We use the whole screen for the game, we left some space at the bottom for text, each level will be presented on the whole screen and the objects size and board size is calculated and changes with the level ratio.

Known bugs:

Other:

UML:

