**OOP1 Project - Save the King**

Written by: Tali Kalev (טלי כלב), ID: 208629691 and Noga Levy (נוגה לוי), ID: 315260927

**General Description:**

In this project, we were asked to implement the game “Save the King” using C++ and the SMFL library. The goal of the game is to seat the King on his throne with the help of other movable players and their different set of skills. There are 4 levels in the game, each having a different level of difficulty and 4 movable players. The game is designed so that new elements and levels can be added with ease. The moment the King comes into contact with the throne, the level is won.

**Design:**

List of the Different Objects:

1. Players (King, Mage, Warrior, Thief)
2. Dwarfs
3. Static Objects (obstacles, bonuses, tools)
4. Board
5. Controller
6. Data Display
7. Audio
8. Timer
9. Menu
10. Resources

Descriptions of the Different Objects:

*Players*

The user is able to switch between four different players (order: King->Mage->Warrior->Thief) using the P key and move them across the board using the arrow keys. Only the King can sit on the throne. The Mage can put out fires and is the only player unable to use the teleports. The Warrior can kill the Orc. The Thief can pick up one key at a time and use them to open gates.

*Dwarfs*

These moving objects are additional obstacles the players must face. A collision with a dwarf blocks the players and they are unable to move. The dwarfs move horizontally across the board switching directions as they hit walls.

*Static Objects*

The game contains many different types of static objects. Some are obstacles that can be faced by some of the players (Wall, Gate, Fire, Orc). Some obstacles are tools used in order to win the game or remove obstacles (Throne, Teleport, Key). Other obstacles can be bonuses. These bonuses, when collided with, can add or take away aspects to the game in order to help or hinder the player in its journey to winning. The following types of bonuses are implemented in the game at random:

* Adding time to countdown timer
* Subtracting time to countdown timer
* Removing all the dwarfs on the board
* Adding a dwarf to the board
* Making the dwarfs faster
* Making the dwarfs slower

*(Note: if there is no timer on the level, none of the timer bonuses are generated)*

*Board*

The board object deals with all the aspects related to the board game. Reading the level from files, creating the objects on the board, removing the objects from the board, holding game data relevant to the players or their progress in the game. The game contains all the movable and unmovable objects on the board.

*Controller*

This object deals with all the aspects related to the polling of events, updating of data and interaction between different objects in the game. The controller contains the board, data display, audio, and menu.

*Data Display*

The data display is an object that appears at the bottom of the game window. It allows the user to keep up to date with certain game aspects (who is the active player, what is the current level, if the thief has a key, time left/passed), turn off the background music, restart the level and return to the main menu.

*Audio*

The audio in the game is controlled using the designated class Audio. It plays the background music that you can hear throughout the game as well as different sound effects when the player collides with different objects, wins or loses.

*Timer*

The timer object in this game either counts the time passed since the beginning of the level or the time left (depending on the attributes defined in the level file). This add a risk level to the game.

*Menu*

The menu is the screen that appears when the game starts. From the menu the user can start a new game, read the instructions on how to play the game or exit.

*Resources*

This is a singleton used to load all forms of media used throughout the game (audio, font, images).

**Level File Format:**

Each file’s name is Level?.txt (? = which level it is). The current game contains 4 level files. If one wishes to add more levels, they must change the const int value NUM\_OF\_LEVELS in the utilities.h header file to fit the number of total levels (including those added) and add the files to the folder resources/levels and resources/CMakeLists.txt.

For example, in resources/CMakeLists.txt:

configure\_file("levels/Level1.txt" ${CMAKE\_BINARY\_DIR} COPYONLY)

The first two integers in the file are the boards height and width values respectively. After the width value there is a newline and then immediately the game board is represented by the movable and unmovable objects using their designated symbols:

K - King

M - Mage

W - Warrior

T - Thief

^ - Dwarf

= - Wall

F - Key

! - Orc

@ - Throne

X - Portal

# - Gate

\* - Fire

$ - Bonus

Immediately after the board, separated by a newline, is the number of **pairs** of teleports and then the coordinates of the pairs of teleports on the board (Y value and then X value). Finally, the last value represents the time set for the level. It is either a positive integer representing the number of seconds on the countdown clock **or** -1 if there is no countdown timer set for this level.

*Additional Requirements for the Level File Format*

* All movable players **must** appear on the board once (King, Mage, Warrior, Thief)
* There must be at least one empty space to the right of each teleport
* Teleports must come in pairs
* If there are no teleport pairs on the board, the number of pairs must be input as 0

**List of Files Created:**

*Files dealing with movable object classes:*

* MovingObject.h, MovingObject.cpp - base class for all moving objects
* Player.h, Player.cpp - base class for all movable players
* King.h, King.cpp
* Mage.h, Mage.
* Warrior.h, Warrior.cpp
* Thief.h, Thief.cpp
* Dwarf.h, Dwarf.cpp

*Files dealing with static object classes:*

* StaticObject.h, StaticObject.cpp - base class for all static objects
* Gate.h, Gate.cpp
* Fire.h, Fire.cpp
* Orc.h, Orc.cpp
* Teleport.h, Teleport.cpp
* Bonus.h, Bonus.cpp - base class for all types of bonuses
* AddTimeBonus.h, AddTimeBonus.cpp
* SubTimeBonus.h, SubTimeBonus.cpp
* RmvDwarfsBonus.h, RmvDwarfsBonus.cpp
* FastDwarfBonus.h, FastDwarfBonus.cpp
* MoreDwarfsBonus.h, MoreDwarfsBonus.cpp
* SlowDwarfsBonus.h, SlowDwarfsBonus.cpp

*Other files:*

* Controller.h, Controller.cpp - deals with poll events, game updates and data transfers between objects
* Board.h, Board.cpp - deals with objects related to board, game data, file reading
* Menu.h, Menu.cpp - deals with the main menu that appears at the start and end of the game
* DataDisplay.h, DataDisplay.cpp - deals with the display and data appearing on the bottom of the game window
* Timer.h, Timer.cpp - deals with the countdown/timer of each level
* Audio.h, Audio.cpp - deal with playing music and sound effects
* GameObject.h, GameObject.cpp - base class for all static and moving objects on board
* Resources.h, Resources.cpp - singleton class that loads all necessary files for the game including images, audio and fonts
* utilities.h - contains names of different objects appearing on the board as well certain characteristics of these objects
* macros.h - contains frequently used values relating to window, sizes, speeds etc.

**Main Data Structures:**

Structure:

In the utilities.h file there is a struct Partners defined. This represents the “pairs” of teleports on the board.

Unique\_ptrs:

Three main vectors of unique\_ptrs are defined to hold the players, static objects and dwarfs of the game.

**Notable Algorithms:**

There are no notable algorithms implemented.

**Bugs:**

When playing the game with headphones, some audio files may not load. This is a bug that only occurs when wearing headphones and has been experienced by other SFML users (according to google research).

**Additional Notes:**

* It is important to leave the space to the right of the teleport empty in order to allow passing to it (this includes keeping it empty of players after they jumped to the teleport)
* After the player wins a level, they must press space in order to continue.
* Pressing the Escape button at any point that the window is open, will close the window.