***Final Project OOP2 תש"פ***

Authors:   
Orel Aviad – 305575003  
Rachel Wolfe – 337816656

Name of the Game:  
Fireman Sam Saves the World from Corona

Explanation of the Program/Game:  
The program introduces an educational game for children to learn if there happens to be a global pandemic and they are unable to continue school as usual. The player in the game (Fireman Mickey) must smash his way through wooden doors and shoot down the Corona with alcogel spray and put out at least 15 fires to get to the Fire station successfully and complete the level. If he arrives at the fire station before all the lights are lit, he will not be able to enter it. The way to light up all 5 lights, is by answering questions. Every time he puts out 3 fires, a multiple-choice question will pop up on the screen in front of him. If answered correctly, a light at the top of the screen and on top of the fire house will turn on. However, if answered incorrectly, neither will light up and the same question will pop up again the next time 3 fires are put out, until the player correctly answers the question. If the player has put out all the fire in the level and not successfully answered all the required questions, the player will lose the game. However as soon as he has correctly answered the 5 questions within the given level, he can put out as many fires as he would like, and no more questions will pop up until he continues through the station to the next level. There are three levels to the game and when the player finishes them all, he wins the game!

Design of Program:  
The different objects used in the program are as follows-  
 \*Game Object – sets the resources for sprites/textures for all objects in the game  
 \*Moving Object – this holds the direction for movement and the sprites  
 \*Static Object – this holds the sprites and their scale  
 \*Controller – this is where all the technical changes happen for the game as it runs  
 \*Board – the game board is built and dealt with here  
 \*Menu – the main menu and all its actions are handled here  
 \*Toolbar – the statistics of the game throughout playing and the setting are set here  
 \*Question – the pop up questions that occur during the game are set and kept here  
  
The player, enemies and their shots are all types of moving objects. Likewise, the fire, hydrants, station and both types of walls (breakable and not) are static objects. They all are types of game objects.

Files Added by us:  
"Controller.h", "Controller.cpp"- controls the game in almost all aspects  
"Menu.h","Menu.cpp" - the settings and movement for the menu  
"GameObject.h", " GameObject.cpp" – all objects come from here  
"StaticObject.h", "StaticObject.cpp" – all immobile objects come from here  
"MovingObject.h", "MovingObject.cpp" - all mobile objects come from here  
"ShootObject.h", "ShootObject.cpp" - all shootable objects come from here  
"WaterShoot.h", "WaterShoot.cpp" - the water shots to kill the fire enemy or fires  
"FireShoot.h", "FireShoot.cpp" – the fire shots to kill the player  
"Question.h", "Question.cpp" – the process of dealing with the questions  
"Board.h", "Board.cpp" – all relevant board functionalities  
"Player.h", "Player.cpp" – the player of the game and his movements  
"CollisionObjects.h", "CollisionObjects.cpp" – instances of all objects possible collisions  
"Hydrant.h", "Hydrant.cpp" – refillable station for the water shooter  
"Fire.h", "Fire.cpp" – plain flaming fire  
"FireEnemy.h", "FireEnemy.cpp" – smart enemy chasers  
"Enemy.h", "Enemy.cpp" – all enemy types come from here  
"Corona.h", "Corona.cpp" – 2 way directional enemy  
"Wall.h", " Wall.h" – unbreakable wall  
"FireStation.h", "FireStation.cpp" – the way to finish the level   
"Toolbar.h", "Toolbar.cpp" – all toolbar stats  
"WoodWall.h", "WoodWall.cpp" – breakable wall  
"AlcogelShoot.h", " AlcogelShoot.cpp" – alcogel shot to kill the corona  
"SingleLoatData.h", "SingleLoatData.h" – singleton resource

Data Structures Used:  
Vectors, lists, map and basic array

Algorithms Worth Mentioning:  
The fire enemy tracks where the player is located and moves towards him by determining which direction is not blocked, if his path is blocked, he paces back and forth until he finds a direction that he can go

Known Bugs:  
None that we know of 😊

Other comments:  
Things that we changed since the presentation:  
 The questions now are read from a file  
 The enemy algorithm is fully functioning  
 We added to the toolbar the lights that show how many questions were answered

Other others:  
 Thanks! And to paraphrase Steve Jobs "Stay ***healthy***, stay positive"