

Faculty of Engineering

Cairo University

**Data Structures and Algorithms**

**Castle Game**

**Simulation Program**

Team Members Sec ID

Yousif Gamal Abd El-Rahman 2 36

Yousif Muhammed 2 37

Khaled Galal Helmy 1 16

Muhammed Ayman Muhammed 2 12

**Main Functions:**

Battle::SimpleSimulator(): contains all functions that's responsible for the simulation to work

Battle::update(): updates the enemies which gone be drawn and also calls update in castle class

Castle::update(): updates the enemy lists in all 4 towers and distributes new enemies including high priority list

Castle::TowersAttack(): let active enemies attack its tower and decrease its health or pave more area depending on the enemy type

Castle::AttackEnemies(): let each tower attack its high priority list and decrease its health

-The battle ends win all the enemies are killed or the four towers are destroyed

In the four regions (A, B, C, D)

**Project Modules**

There are four main classes which runs the simulation and divides it between them

**Battle:** the main class like a simulation manager it contains a list to draw enemies called CEnemeisToDraw[300] which responsible to draw enemies and also contains a priority queue which contains all inactive enemies and when they become active it distributes them with update function and finally contains simulation function which runs the whole show as I might say and last but not least DisplaySound() functions to produce background sound

**Castle:** contains the four towers one for each region and works as a communication channel between class battle and class tower plus it's responsible to update tower lists calling function in it

**Tower:** contains a list for its active enemies and some utility functions to help him manage it also uses polymorphism on enemies to calc damage done to and also by them

**Enemy:** consists of a base class Enemy and derived class for each type (paver, fighter, shielded) and functions to calc damage done to and by towers and some info about each enemy like health, fire power, arrival time and so on

**Code innovation**

* Using priority queue so no sorting is needed when a tower is destroyed and its data is moved to another one also no need to order enemies in the input file according to their arrival
* Some code tweaks when moving an active list from one destroyed tower to another healthy ('D) one, so no sorting is needed
* Using only on active list and using a counter for the shielded type to help in calculating high priority enemies for less complexity
* Every class responsible for its data
* Using OOP as efficient as possible plus using polymorphism with enemies
* Playing appropriate music in the background to help improve and enhance the playing experience 😎
* Enhanced GUI to make the game more fun to play
* Basic controlling also for more fun the world is complicated enough "D
* Using camel case technique in coding to make it uniformed
* Putting instructive comments to help better understanding of the code