Tom&Jerry

written by : Asaf Hakmon , id = 205364409 and Guy Danin , id: 205372105

**Overview:**

A game of cat and mouse starring Tom and Jerry. At the beginning of the game, a menu is presented with four options: New Game, Help, Information about the Objects in the Game, and Exit.

In a new game, the first level opens. The player's objective is to collect all the cheese before time runs out, all while being pursued by cats.

Throughout the game, the player encounters scattered gifts such as extra lives, time extensions, cat deletion gifts, and freeze cat gifts.

There are also doors that will only open if the player collects keys scattered on the board.

Each level is unique, with its own number of cheeses and different types of gifts.

When a cat catches the player, a life is lost, and the player returns to the starting point of the level.

When the time for a level runs out, the level resets and starts from the beginning.

At any point in the game the user can press the ESC button and return to the menu

The goal is to complete all the levels.

**Files Structure:**

**Main.cpp:**

File for running the game.

**Menu.h/Menu.cpp:**

A class representing the game menu, providing access to starting a new game, getting help, details about objects in the game, and exiting the game.

**GameController.h/GameController.cpp:**

A class representing the management of the game flow against the board.

**Board.h/Board.cpp:**

A class representing the game board, managing all display on the board and presenting the status board synchronously with the status data and board data.

**Status.h/Status.cpp:**

A class representing the current status of the player, including the stage number, the number of lives, the player's accumulated points, the number of keys collected, and the remaining time. It also displays this data during the game on the status bar.

**Resources.h/Resources.cpp:**

A class managing all game resource files and loading them, including images and audio files.

**Tile.h/Tile.cpp:**

This files define a Tile class with a constructor, an assignment operator,

and methods to get/set the character and position of the tile.

**CharactersAndObjects.h/CharactersAndObjects.cpp:**

A class representing all inanimate objects in the game from which all objects inherit, except for the cat and mouse.

**Gift.h:**

A class representing game gifts of all types, including time gifts, life gifts, freeze gifts, and cat deletion gifts, from which time gifts, life gifts, freeze gifts, and cat deletion gifts inherit.

**TimeGift.h/TimeGift.cpp:**

A class representing time addition gifts and their management against the status and board.

**LifeGift.h/LifeGift.cpp:**

A class representing life gifts and their management against the status and board.

**FreezeGift.h/FreezeGift.cpp:**

A class representing freeze cat gifts and their management against the status and board.

**DeleteGift.h/DeleteGift.cpp:**

A class representing cat deletion gifts from the game and their management against the status and board.

**GiftEnum.h:**

A file representing all gift enums.

**Key.h/Key.cpp:**

A class representing keys in the game and their management against the board and status.

**Cheese.h/Cheese.cpp:**

A class representing cheese in the game and its management against the status and board.

**Space.h/Space.cpp:**

A class representing empty spaces on the board.

**Blocks.h/Blocks.cpp:**

A class representing walls and doors in the game, inheriting from CharactersAndObjects.

**Wall.h/Wall.cpp:**

A class representing game walls, which are the game's boundaries.

**Door.h/Door.cpp:**

A class representing doors in the game and their behaviour based on the number of keys in the status, as well as their management against the status and board.

**Mouse.h/Mouse.cpp:**

A class representing the mouse, which is the player, and responsible for its movement.

**Cat.h/Cat.cpp:**

A class representing cats in the game, including within it the array of cats, updating the cats, and responsible for their movement algorithm.

**Main Databases and their role :**

The main data structure in the program is the board, it stores all the objects of type CharctersAndObject in each of its cells which is of type Tile, keeps the correct ratio of the size of the distance between objects and thus saves running time in every movement.

We will go over the text file twice and after that we will not have to go over all the objects with every movement of the moving objects.

In addition, we can choose which step size we want.

**Noteworthy Algorithms:**

**Cat Movement Algorithm:**

The cat movement algorithm is designed to simulate the behaviour of cats within the game environment.

Cats in the game attempt to move strategically towards the mouse while navigating around obstacles

and avoiding collisions with other cats.

**Overview:**

**Initialization:** The cats are initially placed on the game board.

**Perceived Threats:** Each cat identifies potential threats, including walls, doors ('D'), and other cats.

**Prioritization:** Cats prioritize movement based on the relative position of the mouse and the perceived threats.

**Movement Decision Tree:**

If the mouse is vertically or horizontally aligned with a cat, the cat moves towards the mouse in that direction.

If obstacles or other cats are in the way, the cat assesses available paths.

Prioritization is based on avoiding obstacles and selecting the shortest path towards the mouse.

**Avoidance Behaviour:**

Cats avoid collisions with other cats by checking their positions and adjusting movement accordingly.

If another cat is detected in the same row or column, the cat evaluates alternative directions.

**Design : Explanation of different objects and their roles:**

**Menu Class:**

Role: Manages the game menu, facilitating access to various game functionalities such as starting a new game, accessing help, viewing object details, and exiting the game.

**GameController Class:**

Role: Oversees the flow of the game, coordinating interactions with the game board.

**Board Class:**

Role: Represents the game board, responsible for rendering all elements on the board and displaying status information in synchronization with the status and board data.

**Status Class:**

Role: Tracks the current player status including stage number, lives, points, keys collected, and remaining time. It also ensures the display of this data during gameplay through the status bar.

**Resources Class:**

Role: Manages the loading and handling of game resource files such as images and audio files.

**CharactersAndObjects Class:**

Role: Serves as the base class for all inanimate objects in the game, excluding the cat and mouse.

**Gift Classes (TimeGift, LifeGift, FreezeGift, DeleteGift):**

Role: Represent different types of gifts in the game, offering various benefits such as time addition, extra lives, freezing cats, and deleting cats. They manage their effects on the game status and board.

**Blocks Class:**

Role: Represents obstacles like walls and doors in the game, inheriting properties from the CharactersAndObjects class.

**Other Notes:**

During the game the player can always exit back to menu by pressing the ESC button

You can add as many stages as you want and in different sizes. The size of the objects will change relative to the size of the stage.