

"Crazy Rusher Game" Brief Documentation

Team name: "Spiderman"

Team members:

- Viktoriya Stoyanova (viktoriya_st)
- Desislava Ivanova (desislavaiv)
- Zdravka Sandoval (Zdravka_Sandova)
- Nikolay Nikolov (nikolay.n)
- Teodora Tikova (teodoratikova)

TFS Repository:

- <https://arreatface.visualstudio.com/>

Project explanation:

Game plot

"Crazy Rusher Game" consists of a matrix, dungeon maze, known as "The maze", which is being uninhabited by swarm of evil creatures, in short known as "The evil creatures".





It is generally accepted, that peace and happiness always reign over the forces of evil, so our main character "Smiley" is being teleported to the dungeon, to destroy the swarm, before they turn the maze into an infectious hive, and populate the world with bloodthirsty dummies.

It stands to reason that the goal of the game is to destroy the evil creatures, faster and make higher score than other players.

Game code

Game code is composed using several classes and main *CrazyRusher.cs* file, which runs the game engine.

GameField.cs:

-  Initializes the main matrix, which represents the maze;
-  *PlayerCoordinates()* sets the player's coordinates onto the matrix;
-  *SetMatrixContent()* fills the matrix with its inner and outer walls;
-  *PrintMatrix()* prints the matrix and its content on the console;

- ✚ *SetConsoleDimensions()* defines the main matrix's dimensions;
- ✚ *DrawBorderLines()* is used also in the welcome screen of the game, and in the gameplay. It separates the game field with the status field;

GameID.cs:

- ✚ *PrintInitialPicture()* reads external files and prints their content on the console. The external files contain the game's welcome logo;
- ✚ *GetPlayerName()* gets the player's nickname;
- ✚ *PrintGameName()* prints the game logo while gameplay process;

Player.cs:

- ✚ *SetPlayerCoordinates()* sets the player char's coordinates in the matrix and places it init;
- ✚ *MovePlayer()* handles the pressed buttons, which can be up arrow, down arrow, left and right arrow for movement, and space bar for shooting;

Enemy.cs:

The main purpose of the class is to generate the coordinates of the enemies and print them on the main game field;

- ✚ *SetEnemyCoordinates()* uses a Random class to generate the 'row' and 'col' values of the current enemy, every time the method has been called. Afterwards it adds the values into a list, which contains all of the enemy's coordinates and places the enemy's char on the game field;
- ✚ *MoveEnemies()* uses again the Random class to generate an integer, which represents the direction, which the enemy will go to. It deletes the enemy's old position and places its new on the game field;

Collisions.cs:

- ✚ *EnemyShot()* handles collisions between enemy and player's arrow:
 - the enemy disappears;
 - *GameScores.Collision Scores()* is called with boolean value "true";
- ✚ *EatenByEnemy()* handles collisions between enemy and player:
 - The enemy disappears;
 - *GameScores.Collision Scores()* is called with boolean value "false".

GameScores.cs

✚ *CollisionScores()* method calls all other methods considering different game situations:

- ✓ if the player shoots an enemy, then he takes 1000 scores;
- ✓ if the player is caught by an enemy, then he loses a life and an enemy dies;
- ✓ if lives are 0 or enemies are 0:
 - *GameOver()* method is called:
 - clears the console;
 - sets the background color red or blue;
 - writes "Game Over" or "You won!";
 - *CalculateTimeElapsed* method is called:
 - takes the start and the finish time in seconds and calculates their difference;
 - *WriteHiScoresHistory()* method is called:
 - Calculates scores by the following formula:
$$\underline{\text{Math.Abs(scores + (scores - elapsed time))};}$$
 - Writes the scores in text file;
 - Displays all time scores from the file;