**TEAM "TRANSFORMERS"**

Team:

Yana Georgieva - [yangra](https://telerikacademy.com/Users/yangra)

Stefan Petrov - [stefan\_petrov](https://telerikacademy.com/Users/stefan_petrov)

Deyan Peev - [didunt](https://telerikacademy.com/Users/didunt)

Ivan Kashukeev - [jquery](https://telerikacademy.com/Users/jquery)

Nikola Nenov - [nikolavn](https://telerikacademy.com/Users/nikolavn)

***Game describtion:***

The goal of the game is to cross a labyrinth in a preset time while shooting enemies and destroying weak walls on the way. The player uses the arrow keys for navigating the ship and spacebar for shooting walls and enemies. There are six different labyrinth maps included in the game and increasing level of difficulty. The increasing level of difficulty means that the number of enemies increases with every next level and the time for reaching the exit decreases. Enemies don’t shoot but the player loses a life if he bumps into an enemy. The exit of the labyrinth that should be rached is marked with an asterix (“\*”), if the player succeeds in reaching the end of the labyrinth in less time than preset he is sent directly to the next random map labyrint – next level.

***Main game specifications:***

* *Setting the console size and parameters*
* *Animate the beginning of the game*
* *Main menu with 3 options*
* *Start game*
* *Set difficulty*
* *Quit*
* *Initializing a level*
  + *Creating the ship and setting a beginning and exit points*
  + *Reading random labyrinth from a text file*
  + *Printing the labyrinth on the console*
  + *Generating enemies on random places in the labyrinth*
* *Moving the player’s ship*
* *Changing ship icon according to its direction*
* *Playing background music*
* *Shooting with the ship*
* *Detecting colision of player with enemy*
* *Detecting collision of bullet with enemy*
* *Detecting collison of bullet with wall*
* *Moving the enemies in random directions*
* *Timer which counts down a preset time*
* *Calcultaing score*
* *Keeping track of player’s lives*
* *Different difficulty levels*
* *Increasing the difficulty when a level is passed*
* *Printing scoreboard with score, time left, lives left and level of difficulty*
* *Printing a GAME OVER message and audio signal it*
* *Printing YOU WON message when level passed and audio signal it*

***URL of the repository:***

<https://yangra.visualstudio.com/DefaultCollection/CSharp%20Console%20Game>

username: brambazluk@gmail.com

password: teamTrans4m

The logic of the game is separated in 9 different classes.

***Class Enemy***

* **Variables**

***List<List<int>>*** *enemies*- collects all generated enemies that are not killed

***int positionX*** – X coordinate of the enemy

***int positionY*** – Y coordiante of the enemy

***char enemyIcon*** – the icon of the enemy

* **Constructors**

***Enemy(int posX, int posY, char icon)*** – constructs a new enemy by given coordinates and icon

* **Methods**

***void MoveEnemy()*** – perambulates the list of enemies, deletes the icon from the last position that it was and prints it on the next position that enemy moved to

***int[] GetRandomDirection(int posX, int posY)*** *–* returns a random change in direction of an enemy on its X and Y coordinates in the form of int array

***void GenerateNewEnemy(int enemyGeneratedCount****) –* generates a preset amount of random enemies using the methods *GetRandomPosition()* and *GetRandomIcon().* Adds the generated enemies to the list of enemies and prints them in the labyrinth.

***List<int> GetRandomPosition()*** *–* returns a random coordinates of an empty space in the labyrinth in the form of list of ints

***char GetRandomIcon()*** *–* returns a random char from a previously specifiedarray of chars

***void CollisionEnemyPlayer(Ship player)*** – checks if player bumped in an enemy and if so signals this to him, returns him at the start of the game and reduces his lives count with one

***void DeleteEnemy(int enemy)*** *–* deletes an enemy from the list and adds points to score

***Class Engine***

* **Variables**

***int fieldOffsetX –*** sets the offset of play field from the left side of console

***int fieldOffsetY –*** sets the offset of play field from the top side of console

***int fieldSizeX –*** sets the size of play field on X

***int fieldSizeY –*** sets the size of play field on Y

***public static int gameInfoX -*** sets the X coordinate of the upper left corner of scoreboard

***public static int gameInfoY -*** sets the Y coordinate of the upper left corner of scoreboard

***int enemiesLeft –*** keeps the count of enenmies left from previous level if it exists

***bool isGameOver –*** keeps the state of the game is it over

***bool isGameWon –*** keeps the state of thegame is it won at the end

***int endPointX –*** sets X coordinate of the labyrinth’s exit

***int endPointY –*** sets Y coordinate of the labyrinth’s exit

***object flag –*** object for resolving the conflicts between threads

***Ship userShip*** – the player ship

***Timer timer*** – the timer of the game

***int randomLabyrinthNum –*** keeps the random number of the labyrinth

***Random rdn*** *–* a random generator for random numbers

* **Methods**

***void PrintOnConsole(int posX, int posY, char character, ConsoleColor color = ConsoleColor.Gray) –*** prints a specified symbol on specified coordinates on console with a specified color

***void PrintStringOnConsole(int posX, int posY, string line, ConsoleColor color = ConsoleColor.Gray) –*** prints a specified string on specified coordinates on console with a specified color

***void ShipMoveShoot(Ship player) –*** checks what key is pressed by the player and reacts accordingly with a movement of the ship or shooting

***void InitializeGame() –*** initializes a new game, sets beginning and end poins, initializes the timer, prints the labyrinth

***void ShootAndMove() –*** a repeating process of printing the player’s ship and bullets, the scoreboard, collision checks

***void MoveEnenmiesAndTimer() –*** a repeating process of decreasing the timer and moving the enemies

***void Music() –*** background music

***void Main() –*** the main loop of the gamerealized with three threads with ***InitializeGame()***, ***MoveEnenmiesAndTimer()*** and ***Music()*** threads are aborted when the game is over or won and a message is printed accordingly sending the player to next level or start menu

***Class GameData***

* **Variables**

***int playerLives –*** lives left to player

***int score –*** player’s score

***gameDifficulty –*** game difficulty

* **Methods**

***void ChangeDifficulty(int difficultyLevel) –*** sets the game difficulty according to the choice of the player in start menu

***void PrintScore() –*** prints the score in the scoreboard

***void PrintLives() –*** prints the lives left in the scoreboard

***PrintDifficulty() –*** prints the chosen difficulty in the scoreboard

***Class GameOver***

* **Variables**

***string[] gameOverArr –*** an array with the drawn text GAME OVER

***string[] gameWonArr–*** an array with the drawn text YOU WON

* **Methods**

***void PrintGameOver() –*** prints GAME OVER on the console and the current score

***void PrintGameWon()–***prints YOU WON on the console, the current score and how to go on the next level

***Class Labyrinth***

* **Variables**

***char[,] labyrinth –*** a char array with the symbols of the labyrinth

* **Methods**

***void ReadLabyrinth() –*** reads a random labyrinth from text file and transfers it in the labyrinth array

***void PrintLabyrinth() –*** prints the array on the console

***void ChangeChar(int positionX, int positionY, char change)*** –changes a char from the ***labyrinth*** array

***Class Ship***

* **Variables**

***int xPos –*** defines X coordinate of the ship

***int yPos –*** defines Y coordinate of the ship

***char shipIcon –*** defines the icon of the ship

***int shipDirection –***  defines the direction of the ship

* **Constructors**

***Ship(int xPos = 0, int yPos = 0, int shipDirection = 3, char shipIcon = '#') –*** constructs an ship with predefined coordinates direction and icon

* **Methods**

***void PrintShip() –*** prints ship with different icon according to the direction

***Class Shot***

* **Variables**

***List<List<int>> shots –*** a collection of all active shots

***int posX –*** defines X coordinate of the shot

***int posY –*** defines Y coordinate of the shot

***int direction –***  defines the direction of the shot

* **Constructors**

***Shot(Ship player) –*** constructs an shot according to the player’s ship coordinates and direction and adds it to the list of shots

* **Methods**

***void MoveBullets() –*** moves bullets according to their direction and reprints them

***void DeleteBullet(int bullet) –*** deletes a bullet from the list of shots

***void CollisionEnemy() –*** checks for collisions between shots and enemies

***void CollisionWall() –*** checks for collisions of shots with walls and reacts according to the type of wall

***Class StartMenu***

* **Variables**

***bool notSelected –*** keeps the state if there is selection made

***int cursorPositionLeft –*** defines the lposition of the cursor according to the left side

***int cursorPositionTop –*** defines the lposition of the cursor according to the top side

***int difficultyLevel –***  defines the level of difficulty

***string[] startMenu –***  a string array with drawn start menu

***string[] difficultyMenu –***  a string array with drawn difficulty level

* **Methods**

***void Animate() –*** slides the menu from the top side of the console

***void PrintMenu() –*** prints the string array ***startMenu***

***void SelectTask() –***  loops menus while there isn’t a selected option and if an option is chosen acts accordingly

***Class Timer***

* **Variables**

***int xPos –*** sets timer X position

***int yPos –*** sets timer Y position

***int secondsLeft –*** sets seconds left

***ConsoleColor color –***  sets the color of the timer

* **Constructors**

***Timer(int xPos = 0, int yPos = 0) –*** constructs a timer with predefined coordinates

* **Methods**

***void PrintTheTimer(int posX, int posY, ConsoleColor color) –*** prints the value of the timer on specified coordinates with specified color