Script “Load” :

- **Method : “get_records_by_email”**

  .Added a condition to check whether the level is finished or not and changing

  the button's text to LOCKED.

  A box collider for all levels is created. *If the level is pressed it will be checked if it's

  unlocked and if it is

  finished , if this is true the level will be loaded , otherwise the icon will be changed to LOCKED.

  . Adding conditions to get the levels and put the value of the level in the right 3Dtext

  object.

  . Adding conditions to get the scores and out the value of the score in the right 3Dtext

  object.

  Script “

Scene “Level1” :

- moving the light source up and down and rotating the light source
- introducing the new tools (buttons) in level zero
    - in the game engine in both scenes (Level1 / Level2) you will find 4 buttons on the left of the rail
    - the right 2 buttons are responsible to move the player up and down
        - these 2 buttons are 2 3D cubed named Up - Down

        - each one has a related C# script UpMove - DownMove
    - the left 2 buttons are responsible to rotate the player
        - these 2 buttons are 2 3D cubed named R1 - R2
        - each one has a related C# script R1Move - R2Move
    - in each one of those scripts we handle the mouse events
        - onMouseDown → indicating that the mouse is pressed.
        - onMouseUp→ indicating that the mouse is released
        - the effect of any of those events is to set a boolean flag to true or false indication if the player should move up or down / rotate clockwise or anticlockwise
    - in the script of each button we handle the halo effect that appears for 5 secs and the text that describes the functionality of this button in the Update method which is called during each time frame
    - there is a 5th script “Player Script”
        - it contains 4 core methods that actually move the player and rotate it
        - in the update method of this script we check the boolean variables (the values of those flags when the onMouseDown and onMouseUp are executed)
        - if the player should move or rotate we call one of the 4 methods mentioned before
    - the 6th script is the one responsible of showing the description of each button in the practice mode

- Light beam was created as Line render whose points are decided according to the position and direction of the shooter object, as the light beam always starts from the center of the shooter and is directed to where the shooter is pointing.
- The collision detection between the light beam and target was done using a Raycast and giving the target object a tag indicating its type ("Target").
- The game ends when a collision is detected and the player can no longer move the source. A congratulating message and a button that can redirect to other scenes appear when level is over that is indicated by a variable called gameover that turns true when a collision between light beam and target occurs.
- A cylinder was used as obstacle. It had the tag "Obstacle" to be identified when light collides with it. It was equipped with a material having a black hole texture.
- Text was added to illustrate the job of the black hole.

Adding the game to the website:

- Uploaded the game by building all the scenes then moving it to the public folder and referencing it in the iframe in the homepage of the website.

The game consists of 8 levels
Level 1-2 → “The Light travels in a straight line”
Level 3-5 → “Reflection”
Level 6-8 → “Refraction”

All the scripts are implemented in C# and you can find them in LightGame/Assets/Scripts
All the unity scenes are in LightGame/Assets/Scenes

Level 1
Scene → Level1_MM.unity

1- ShooterScript_1.cs

2-UpMove_1.cs

3-DownMove_1.cs

4-R1Move_1.cs

5-R2Move_1.cs

Level 2
Scene → Level2_MM.unity

1- ShooterScript_2.cs

2-UpMove_2.cs

3-DownMove_2.cs

4-R1Move_2.cs

5-R2Move_2.cs

in the 1st script ShooterScript_1.cs / ShooterScript_2.cs

- Update()

        - this one is called in each time frame
        - we check the 4 boolean flags of the movement if one of them is true we call the method responsible of this movement
        - display the number of moves done so far
    - MoveDown()
        - decrease the position vector of the shooter
    - MoveUp()
        - increase the position vector of the shooter
    - RotateCW()
        - rotate clockwise
        - increase the rotation vector (around the X axis)
    - RotateCCW()
        - rotate counter clockwise
        - decrease the rotation vector (around the X axis)

in all the scripts from (2 to 5) you will find 2 methods :-

- OnMouseDown()
    - this one handle the event of pressing the button
    - it sets the boolean flag of the movement in (ShooterScript_1.cs) to True indicating that it should move/rotate
- OnMouseUp()
    - this one handles the event of releasing the button
    - it sets the boolean flag of the movement in (ShooterScript_1.cs) to False indicating that it should stop moving/rotating

Level 3-5
The Three scenes are alike.

Scenes → level3.unity/Level4.unity

1. Reflection_level_3.cs
2. rotateButtons.cs
3. AddingMirrors2.cs

script Reflection_level_3.cs

- update()
  -check if on the rotate booleans if they are true, the methods responsible for rotation are
  called
  -two methods detector() and setlightBea()

- RotateRight()

        - increase the angle of the lightbeam
        - call the method RotateLightbea() to apply the rotation

- RotateLeft()
    - decrease the angle of the lightBeam
    - call the method RotateLightBeam to apply rotation

- RotateLightBeam()
    - it calles two methods PointRotator() and adn setLightBeam()
- PointRotator()
    - vectors are calculated to calculate the correct rotation of every point in the line
- SetLightBeam()
    - apply what is calculated by the method PointRotator()

script rotateButtons.cs

- the script contains two methods :
    - OnMouseDown()
        - It detects if the button assigned to the script is clicked, it calls a method in
          script Reflection_level_3.cs that turns the boolean responsible for the rotation to true.
    - OnMouseUp()
        - It detects if the button assigned to the script is released, it calls a method
          in script Reflection_level_3.cs that turns the boolean responsible for the rotation to false.

script AddingMIrror 2 ()

- this script is assigned to a gameobject which is the mirror
- it is responsible for moving the mirror in the scene

Level 5-6
Scenes → Level5.Unity / Level6.unity

1- ShooterScript5.cs

2- GoLeftScript.cs

3- GoRightScript.cs

4- R1_Level5.cs

5-R2.Level5.cs

In the ist Script shhoterScript5.cs

- Update()
    - we check the boolean flags of the movements if one of them is true we call the method responsible the move the shooter
    - display the number of moves done so far
- RotateRight()
    - increase the rotation vector of the shooter if it is less than the maximum
    - call the method responsible to rotate and refract the light beam
- RotateLeft()
    - decrease the rotation vector of the shooter if it is less than the maximum
    - call the method responsible to rotate and refract the light beam
- RotateLightBeam()
    - this method calls 2 other methods
        - PointRotator ()
            - rotate the 1st half of the light beam
            - calculate the angle of refraction
            - rotate the 2nd part of the light beam using the angle of refraction
            - all these changes are done by changing the values of the start , middle and end point of the light beam
        - SetLightBeam ()
            - applies the changes done in the previous method on the light beam

Level 8

Topics → Reflection and Refraction

Scripts → Mirror_8.cs / Player_L3_3.cs / Movement_L3_3.cs

Player_L3_3

- we have 4 states for the level
    - state -1 → Introductory tip #1
    - state 0 → Introductory tip #2
    - state 1 → The player can play the game
    - state 2 → the game is
- movement(int x)
    - the integer x determines the type of movement
        - rotate ccw
        - rotate cw
        - stop rotating ccw

            - stop rotating cw
        - save the logs in the database
    - RotateLightBeam()
        - rotate the light beam and handle the reflection
            - rotate the upper part of the light beam
            - calculate the angle of refraction
            - set the second part of the light beam
    - detector()
      detects the collision with any other game object
        - Mirror
            - if the light beam hits the mirror we have to set a new light beam to simulate the light reflection
        - target
            - the game is over → change the state to 2
            - save the logs of the player in the DB

        - obstacle
            - the light beam ends at the hit point

Movement_L3_3.cs → handle the buttons' events (click and release)

Mirror+8.cs → to move the mirror left and right

Quiz 1
Topics → Light travels in a straight line
Scripts → SolutionChooser_Quiz1.cs / Quiz1.cs

Quiz1.cs
- save_answer()
    - save the selected answer in the DB
- Update()
    - we have 2 state
        - playing
        - the level is over

SolutionChooser_Quiz1.cs
- OnMouseDown()
    - set the selected answer field
    - move to the next state (the level is over)

Quiz 3

Topics → Light travels in a straight line
Scripts → SolutionChooser_Quiz3.cs / Quiz3.cs

Quiz3.cs

- save_answer()
  - save the selected answer in the DB
- Update()
  - we have 2 state
    - playing
    - the level is over

SolutionChooser_Quiz3.cs

- OnMouseDown()
  - set the selected answer field
  - move to the next state (the level is over)