****

CSE1142 Project

Project Name : O . G . I

Authors:

IDs : 150117036 – 150117064

Names : Muhammed Enes – Yunus Emre

Surnames : Aktürk – Ertunç

CSE1142: Computer Programming II, Spring 2019

Instructor : Sanem Arslan YILMAZ

Date Submitted: May 11, 2019

**-PROBLEM DEFINITION**

The name of our game is OGI Our game’s aim is creating a path that leads the ball by using sliding tiles. The player is expected to form a path using pipes given as sliding tiles. When each tile is in accurate position, which means level is solved , the ball located at starter tile starts to move towards end tile by following path.

The game has some specific rules. For example, if you are in the first level and can’t solve it yet, you cannot play other levels. Another important rule is that tiles cannot move diagonally. Additionally, some of the tiles are fixed, which means they cannot move.

**-IMPLEMENTATION DETAILS**

|  |  |
| --- | --- |
| OGI | |
| +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  + | stagesArray : ArrayList<CreateBoxes>  stagePassArray : ArrayList<CreateBoxes>  hasPassed : ArrayList<Boolean>  pathList : ArrayList<Double>  bp , gp , pane : Pane  newClassTile, basıldı, bırakıldı : CreateBoxes  hasFinished,level1Passed,level2Passed,level3Passed,level4Passed,level5Passed,level6Passed:Boolean  WIDTH , HEIGHT , pointX , pointY : Double  menu, play, options, exit, music, voice , language, back, time, move, yes, no, nextLevel, oldLevel : Polygon  gameName, playT, optionsT, exitT, languageT, backT, yesT, noT, nextLevelT, oldLevelT : Text  playStr, optionsStr, exitStr, languageStr, backStr, yesStr, noStr, exit2Str, nextLevelStr, oldLevelStr, moveMessage, moveStr, timeMessage : String  bölüm, hamleSayısı, basılan, bırakılan, nerdeyim, nerdeydim, secondsPassedint, secondsPassed : int  moveLabel, secondsLabel : Label  timer : Timer  siw, siw2 : ImageView  ball : Circle  path2 : Path  arcTo : ArcTo  pt : PathTransition  mediaMusic , mediaVoice : Media  mpMusic , mpVoice : MediaPlayer  sliderMusic , sliderVolume : Slider  timerTask : TimerTask |
| +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  + | menuOlustur() : void  playOlustur() : void  optionsOlustur() : void  exitOlustur() : void  level1Scene() : void  level2Scene() : void  level3Scene() : void  level4Scene() : void  level5Scene() : void  level6Scene() : void  levelPassedScene() : void  paneOluştur() : Pane  oynatma() : void  createArc() : void  pathOynat() : void  kontrolEt() : void  kontroleDevamEt() : void  TrString() : void  EnString() : void  dilDeğiş() : void  start(Stage primaryStage) : void |

● In **menuOlustur** method, polygons and shapes belonging this menu are created. These features are same for "**playOlustur**", "**optionsOlustur**" and "**exitOlustur**" methods.

● In **level1Scene** and other methods for level creating, txt file for each level is read by scanner and objects are created from **CreateBoxes** class by using information coming from txt files. Each object is added into **stagesArray** array list. Then **paneOlustur** method is called and game screen is created.

●In **levelPassedScene** method, each tile is taken from **stagesArray** and placed on screen .According to the coordinates, "*nextLevel*" and "*back*" polygons are created. The level we are at is determined. According to the this, if *nextLevel* polygon is clicked, next level is loaded. If *back* polygon is clicked, menu scene is loaded. In addition to this, this method prints how many moves are done and how much time is passed.

●In **paneOlustur** method, each tile taken from **stagesArray** is placed on screen and polygons in **levelScene** method are placed as well. This method is called everytime when tile is moved. And screen is updated. As a last thing, **oynatma** method is called in this method.

●In **oynatma** method, tiles in **stagesArray** is arranged by looking *canMove* property. If tile can move, the location in the array is rearranged and **paneOlustur** method is called again.

●In **createArc** method , box type of each appropriate tile in **stagePassArray** is determined and all of them is checked in switch-case. Then, pointX and pointY coordinates of tiles are determined and they are added into *path*. After all these things **pathOynat** method is called.

●In **pathOynat** method, the path created is added into *pathTransition* which is formed before. Then the ball(circle) as a node is added into this *pathTransition* and animation is played. If it is clicked on the screen, **levelPassedScene** is called and finish scene is loaded.

●In **kontrolEt** method, starterTile is found out by searching in **stagesArray** and index of this tile in **stagesArray** is loaded into variable named "*nerdeyim*". Then, **kontroleDevamEt** method is called.

●In **kontroleDevamEt** method, variable "*nerdeyim*" is found in **stagesArray** and its direction is determined by getting direction property of this object. According to the this, variable "*nerdeyim*" is incremented. (ex : If it is vertical, forth index in array after nerdeyim, is analyzed.). While stagesArray is analyzed, each passed object is added into **stagePassArray**. If object is "free" or "none", the solution is not correct that's why **stagePassArray** is cleared. If index of "nerdeyim" is equal to the index of end tile that means the solution is correct, and boolean variable of that level is turned into true. After all of these operations , **createArc** method is called.

●In **TrString** and **EnString** methods, string expressions are converted into Turkish or English.

●In **dilDegis** method, string is checked and if it is Englısh ,**TrString** method is called. If it is Turkish , **EnString** method is called.

|  |  |
| --- | --- |
| CreateBoxes | |
| +  +  +  +  + | index : int  kutuTipi : String  yönü : String  canMove : boolean  iw : ImageView |
| +  +  +  +  +  +  +  +  +  +  + | CreateBoxes(int index, String kutuTipi, String yönü)  kutuOluştur(String kutuTipi , String yönü) : void  setIndex(int index) : void  getIndex() : int  setKutuTipi(String kutuTipi) : void  getKutuTipi() : String  setYönü(String yönü) : void  getYönü() : String  setIw(ImageView iw) : void  getIw() : ImageView  setCanMove(boolean canMove) : void |

●In **kutuOlustur** method, the file including the images of the tiles is read by scanner. Then, imageview named as "*ImaView*" is created by using parameters which are "*kutuTipi*" and "*yönü*" . After that, size of the image is adjusted.

We started our project by naming our game. Then, we thought about how it should look like and made a decision to choose one of the designs we did. In general, mechanism of the game is mentioned above.

Necessary features that are expected to include in the game were added except for one. When you click the tile and drag it, it does not drag where the mouse is. It just changes the position with tile where the mouse is released .

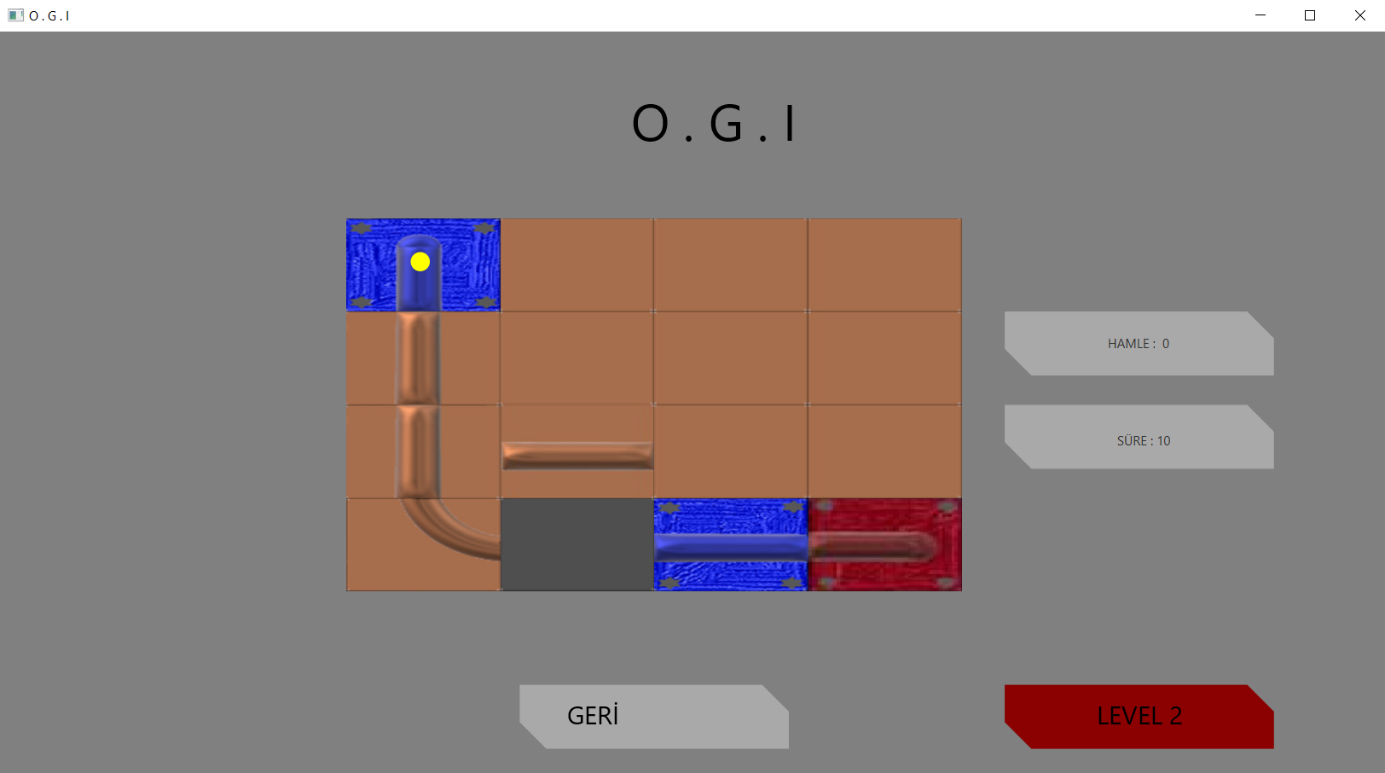
We have encountered with a lof of problems during the game’s implementation. When we began to build our game, switching between the scenes was a big problem for us. Another problem was rearranging the size of the screen. That is , adding resize property made us have a trouble with nodes. Creating a path was so difficult. In addition to all of these, checking the solution of level was the most difficult part for us because there must be something like artificial intelligence. In spite of all of these problems we created our game succesfully.

To make our game better and different from others' , we added some extra features to it. We designed the game screens simple and easy to make it more understandable as much as possible so that the user can reach wherever he wants like menu screen, previous level and so on. We added music into the game . You can adjust the sound level of the music or just close it. Additionally, we added timer which starts when the level is started to determine how much time is passed. To check the correctness of mechanism of our game, we added a new level created by us. As a last thing, it can be the most important one, we added language option in the game to reach everyone wishing to play our game.

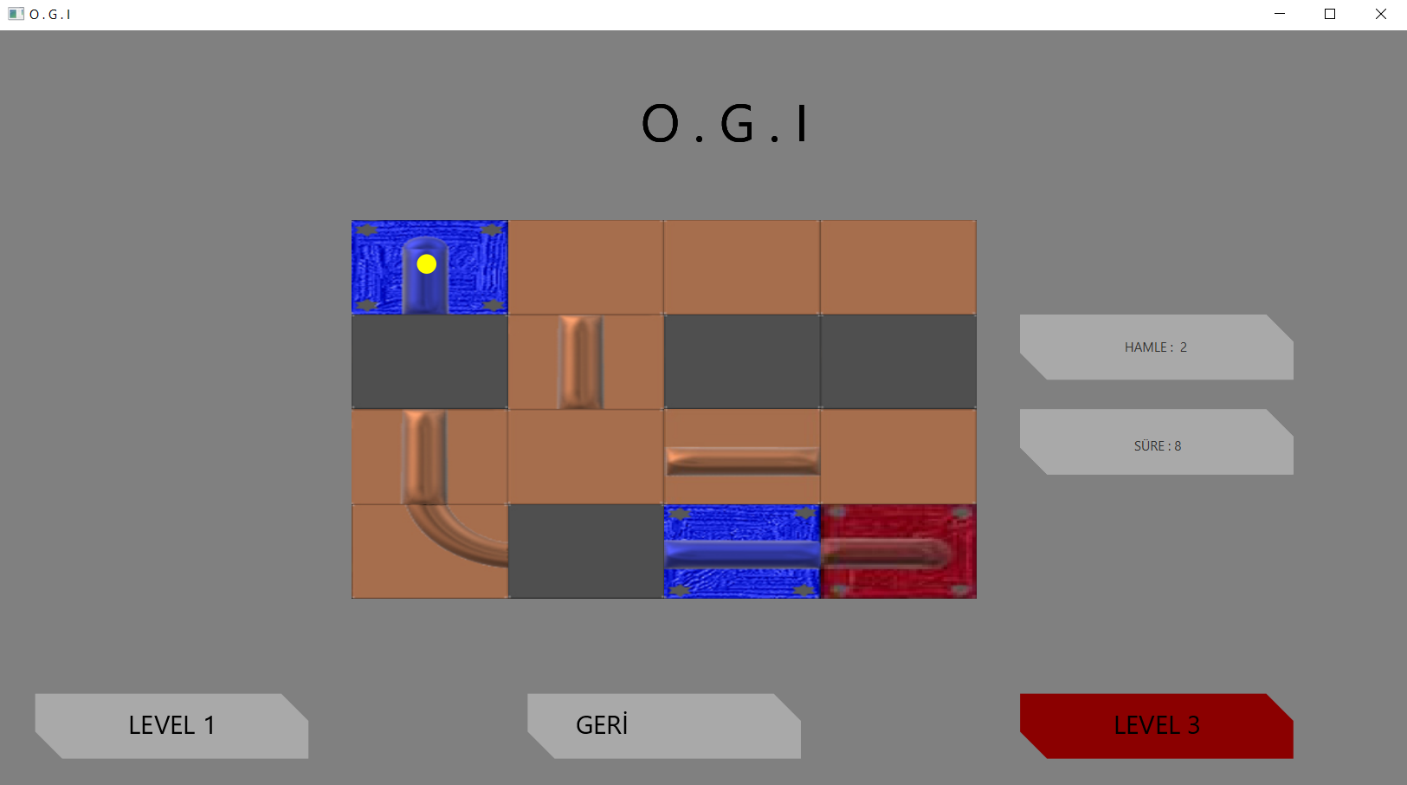
**-TEST CASES**

On the right of each level screen you can see the number of moves and the passed time. As you can see, next level and back level buttons are located at the right and left corner. As a last thing back button is located at the bottom of the screen. Apparance of each level generally looks like that.

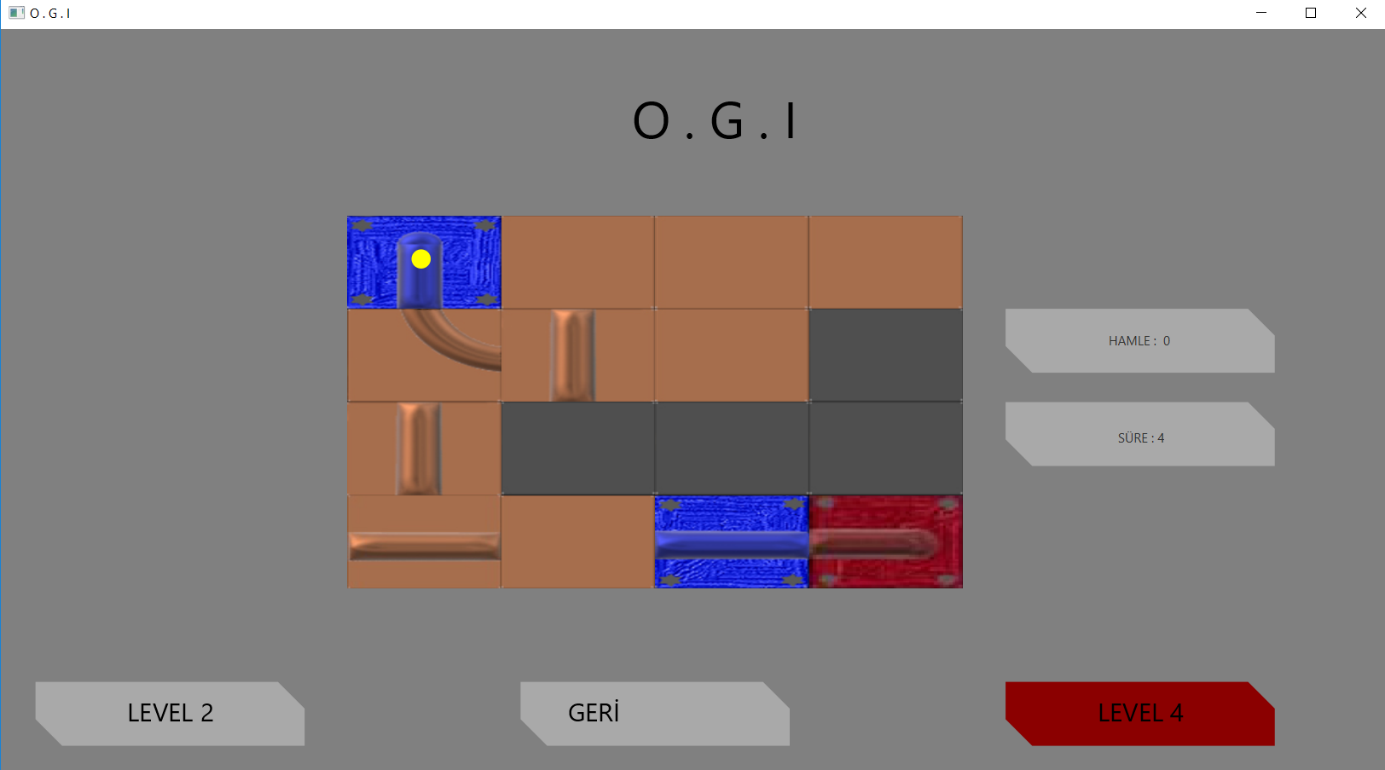
\*Level 1 :

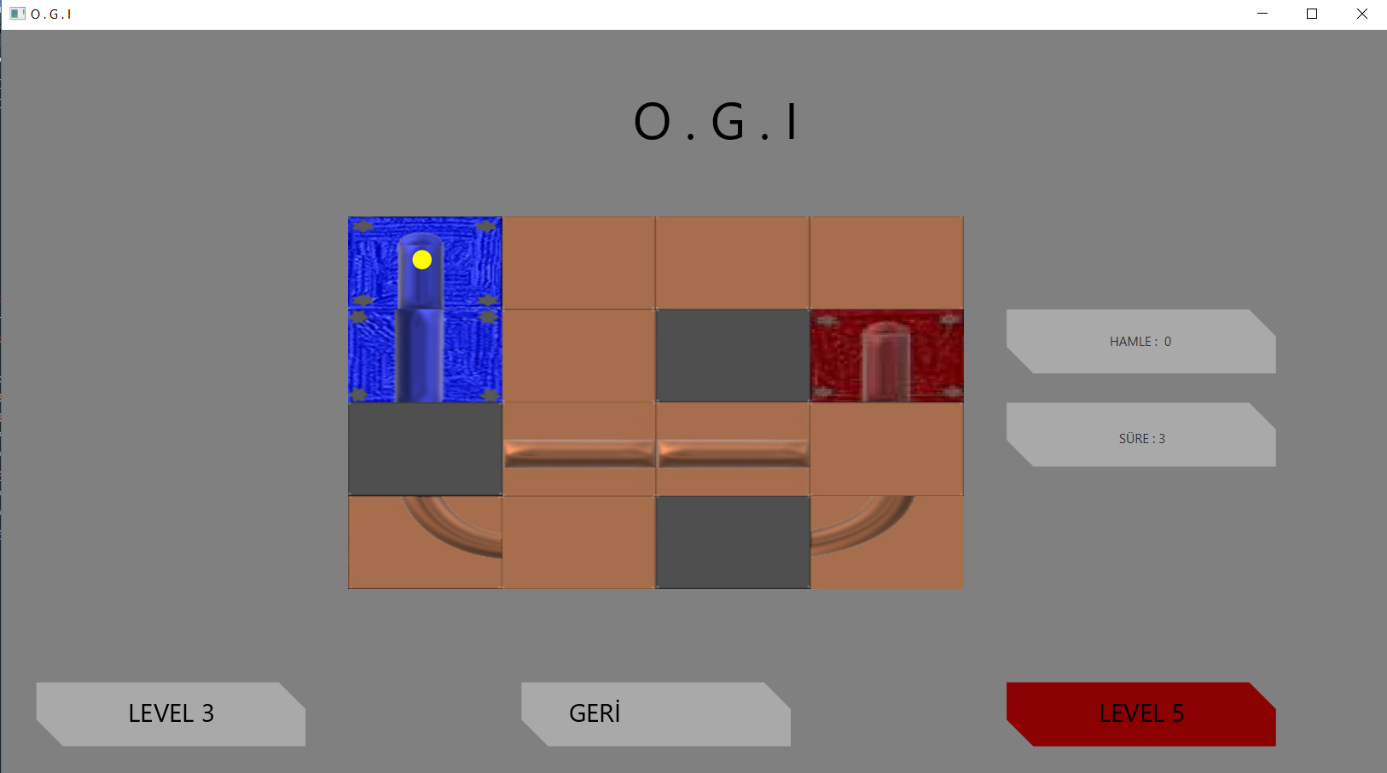


\*Level 2 :

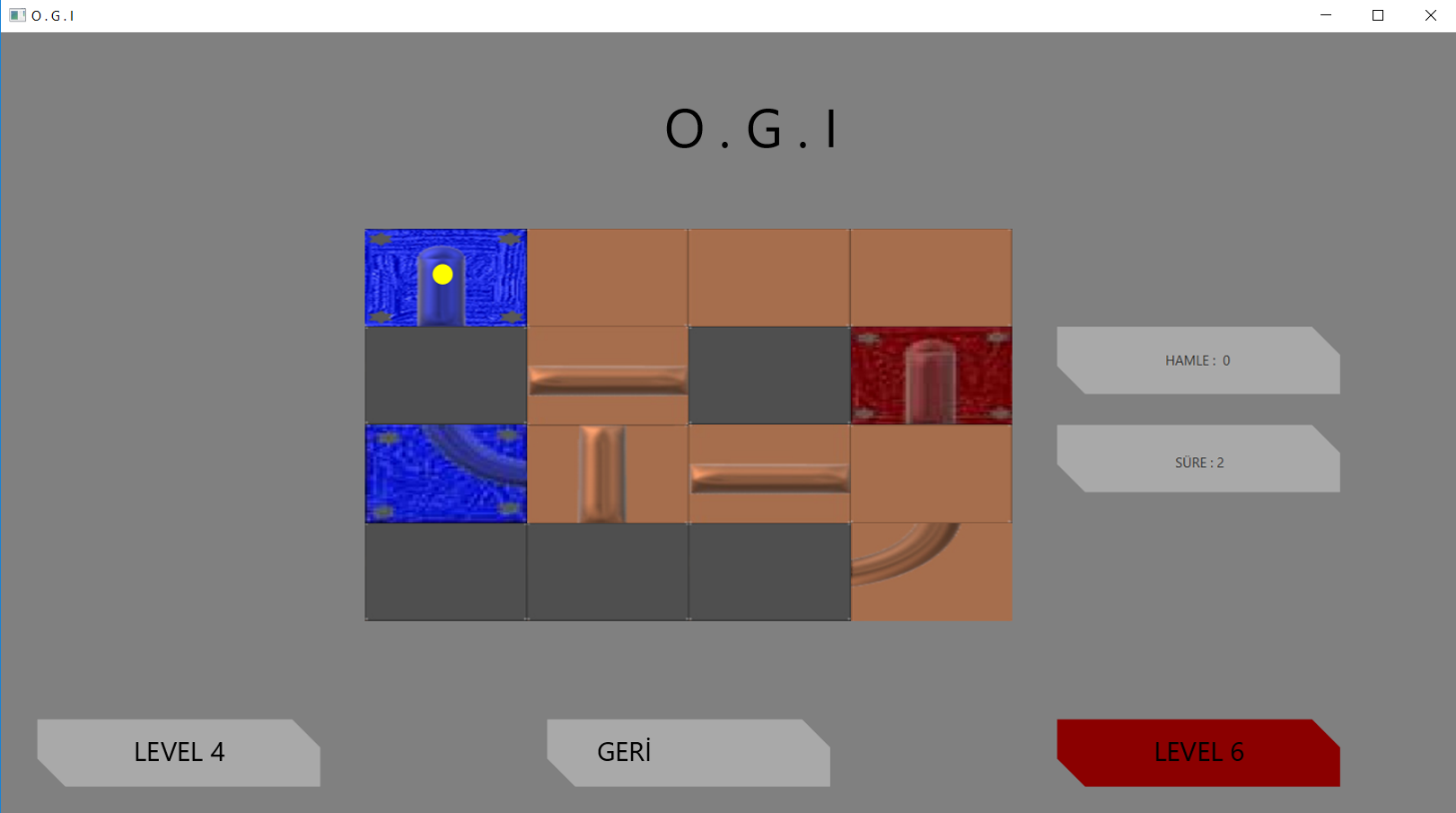


\*Level 3 :

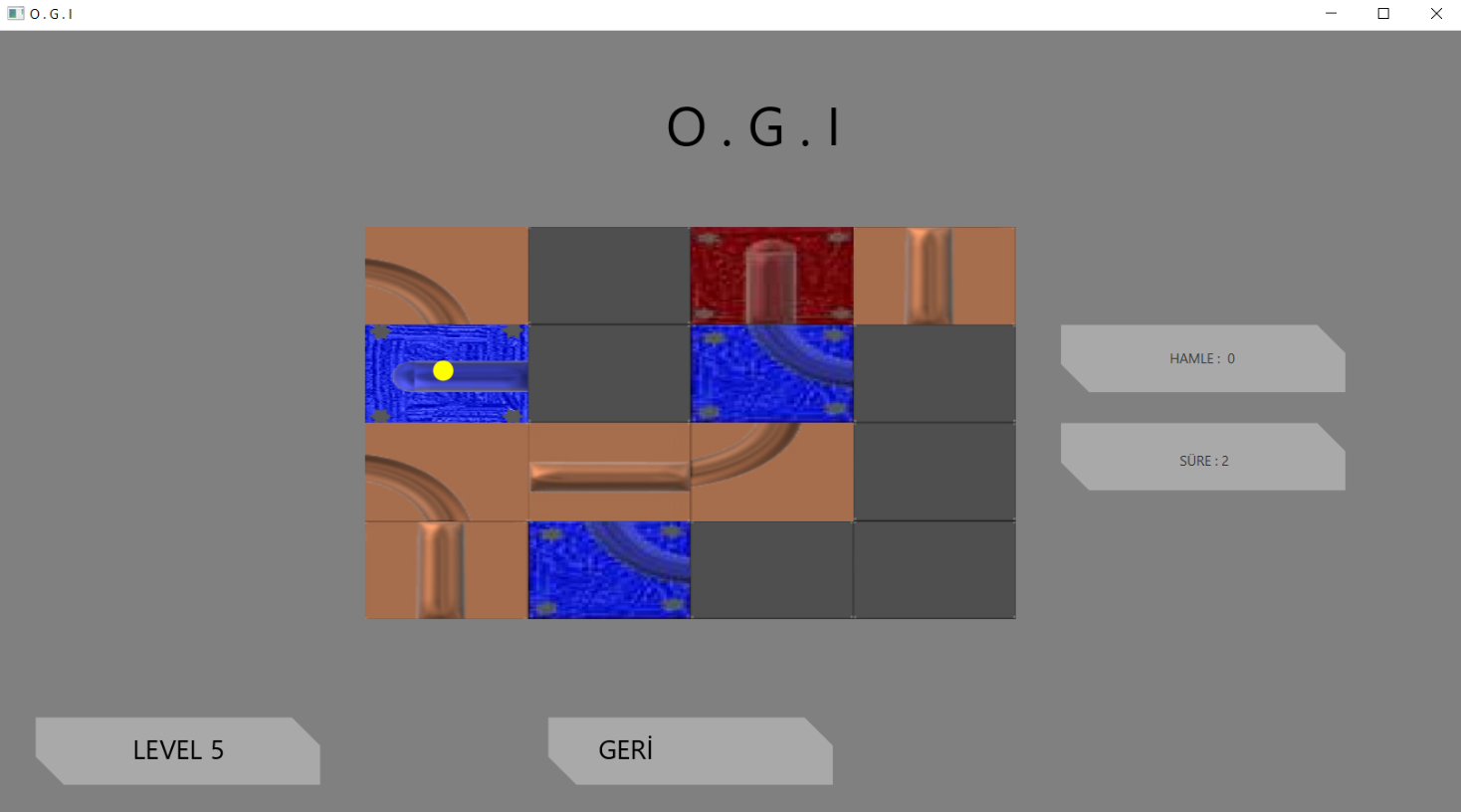


\*Level 4 : 

\* Level 5 :



\*Level 6 :



Level 6 is designed by us . This level of the game is some complicated than others.

\*\* End of the each level :



When the level is solved and user clicks the screen, this scene ,showing the number of moves and passed time, will be loaded.