Visual Programming Semester Project

Project by: **Muhammad Anas Baig**

01-134152-037

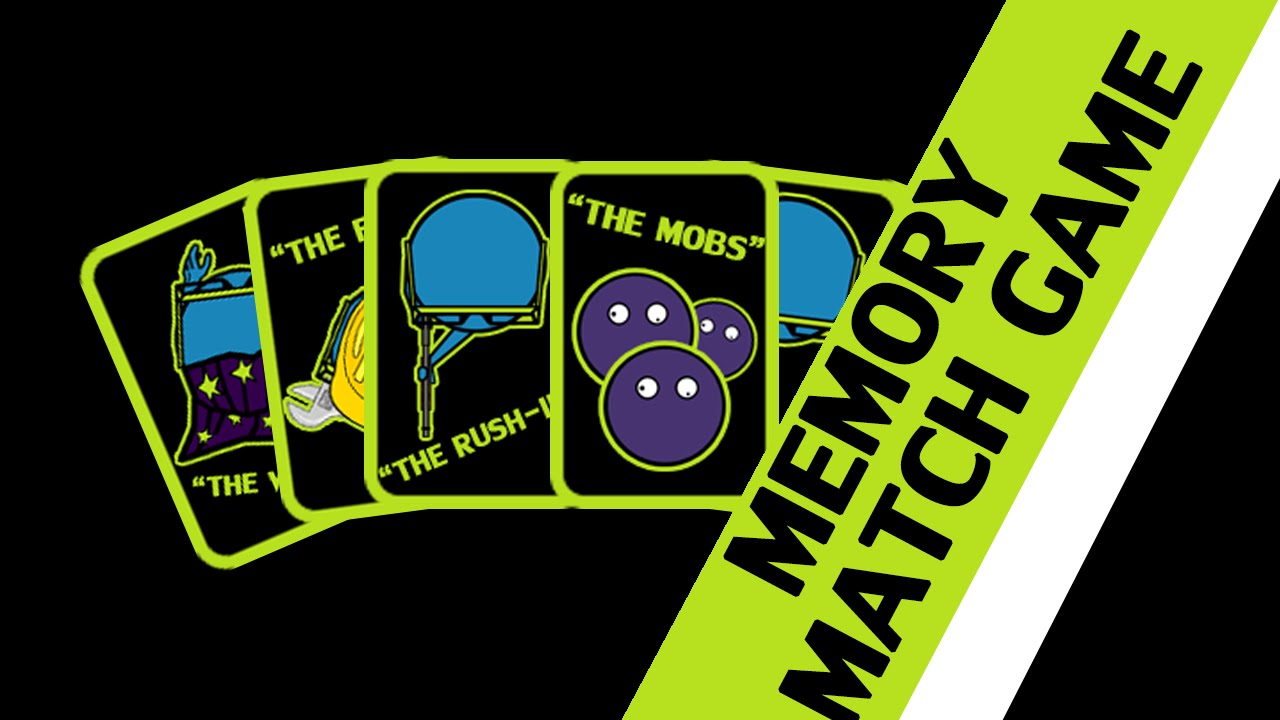
BS(CS)-5A



[Document subtitle]

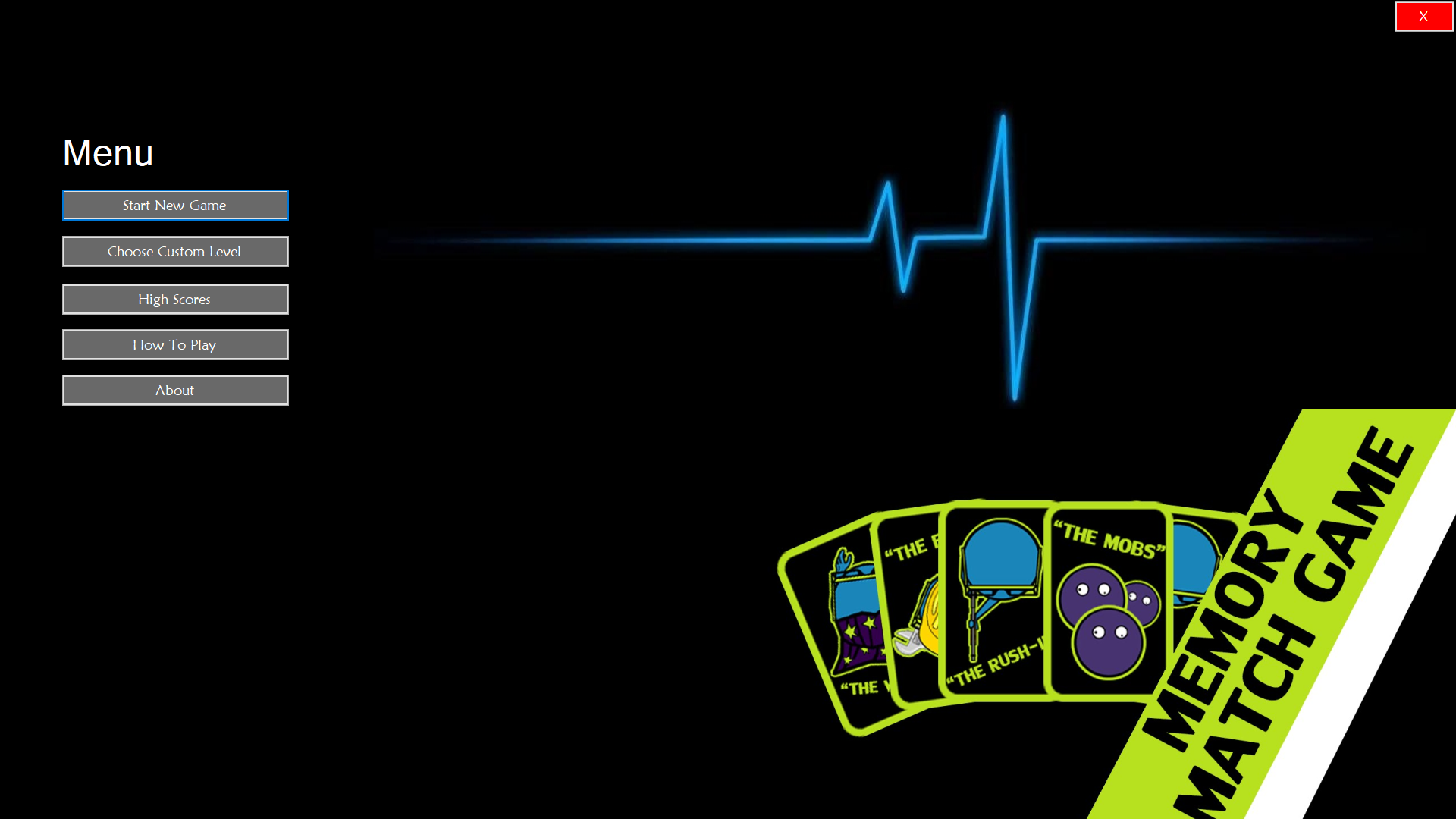
December 11, 2017

Title: **Memory Matching Game**

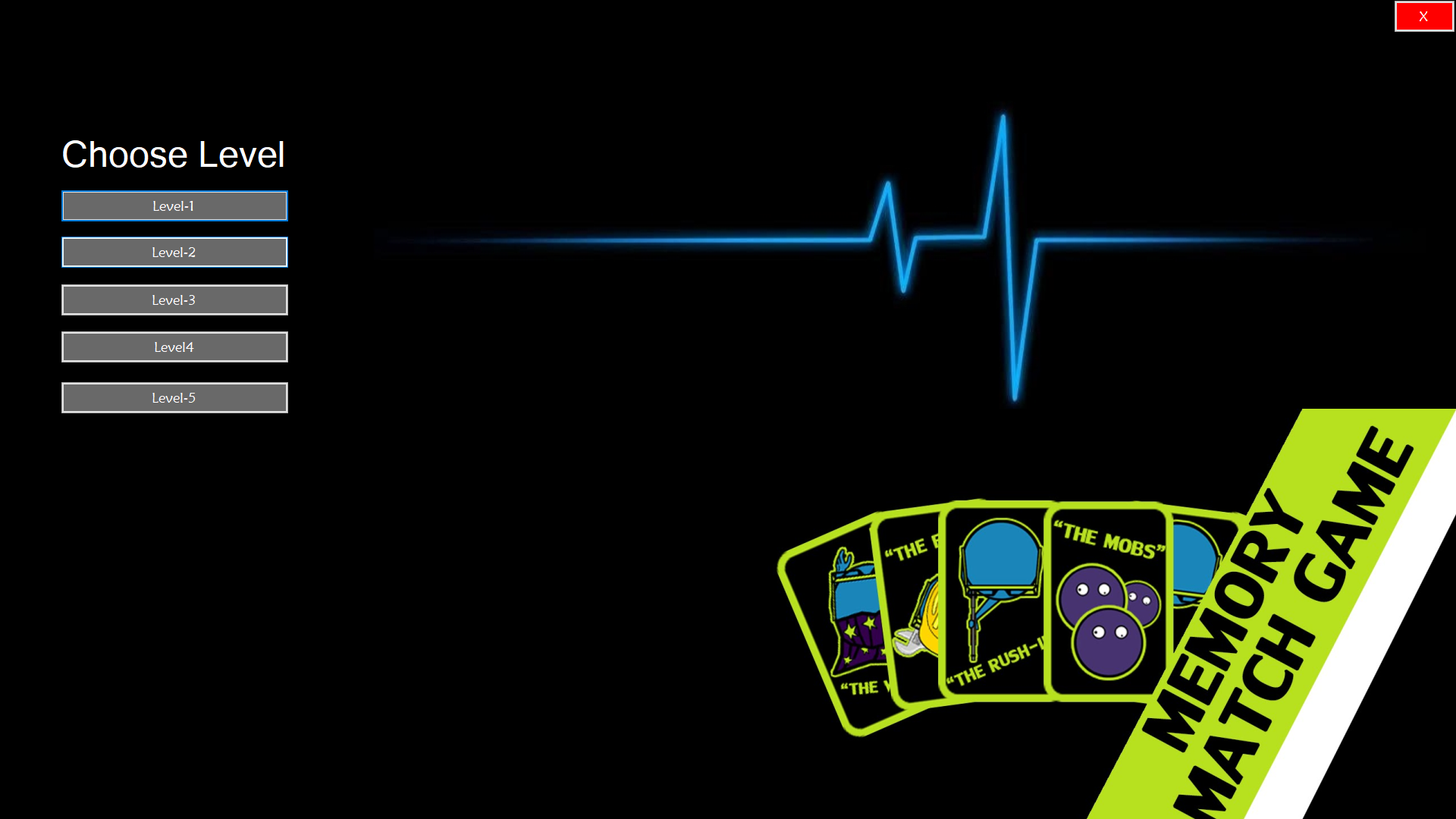


**Screenshots**

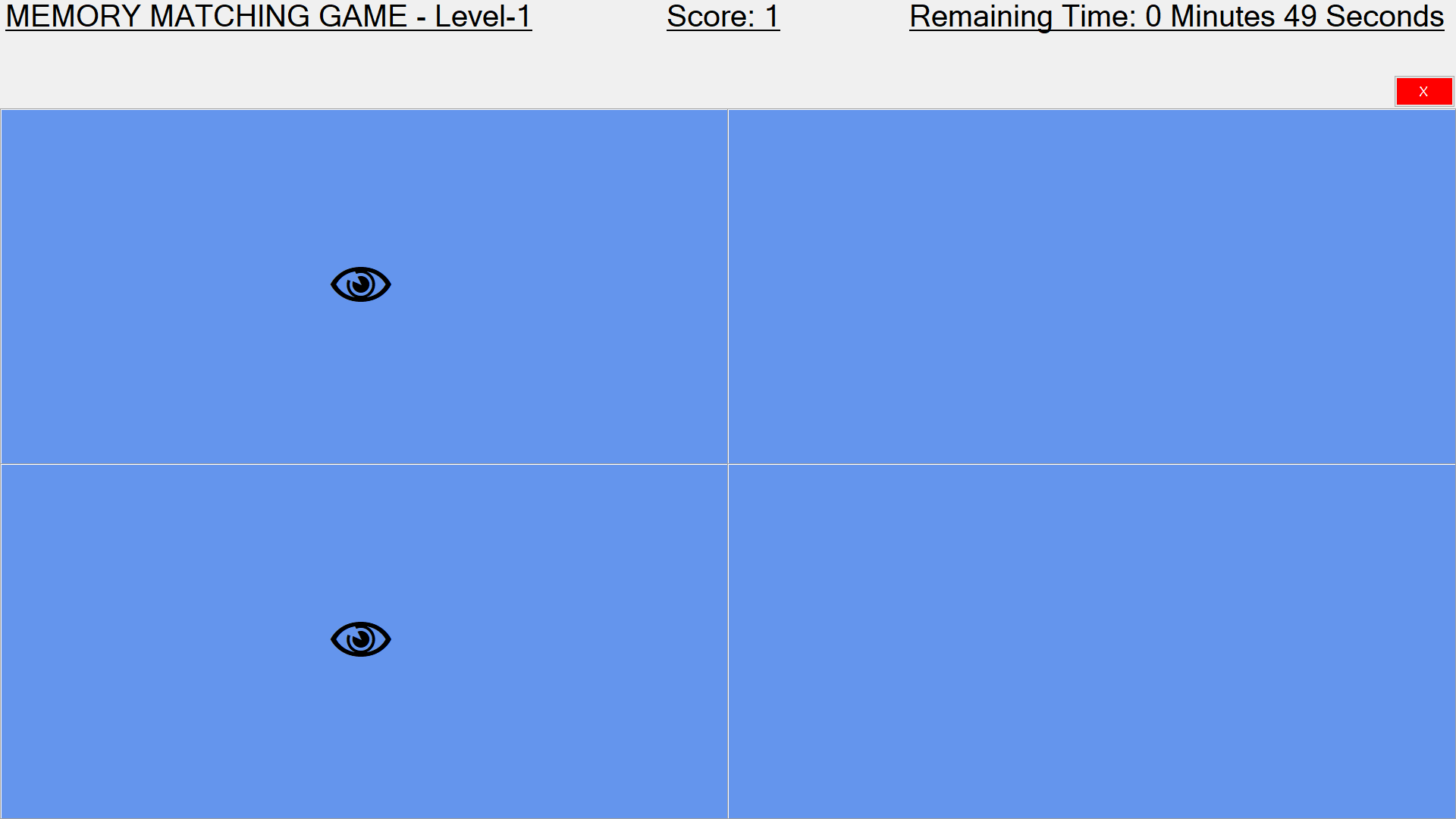
**Main Menu:**



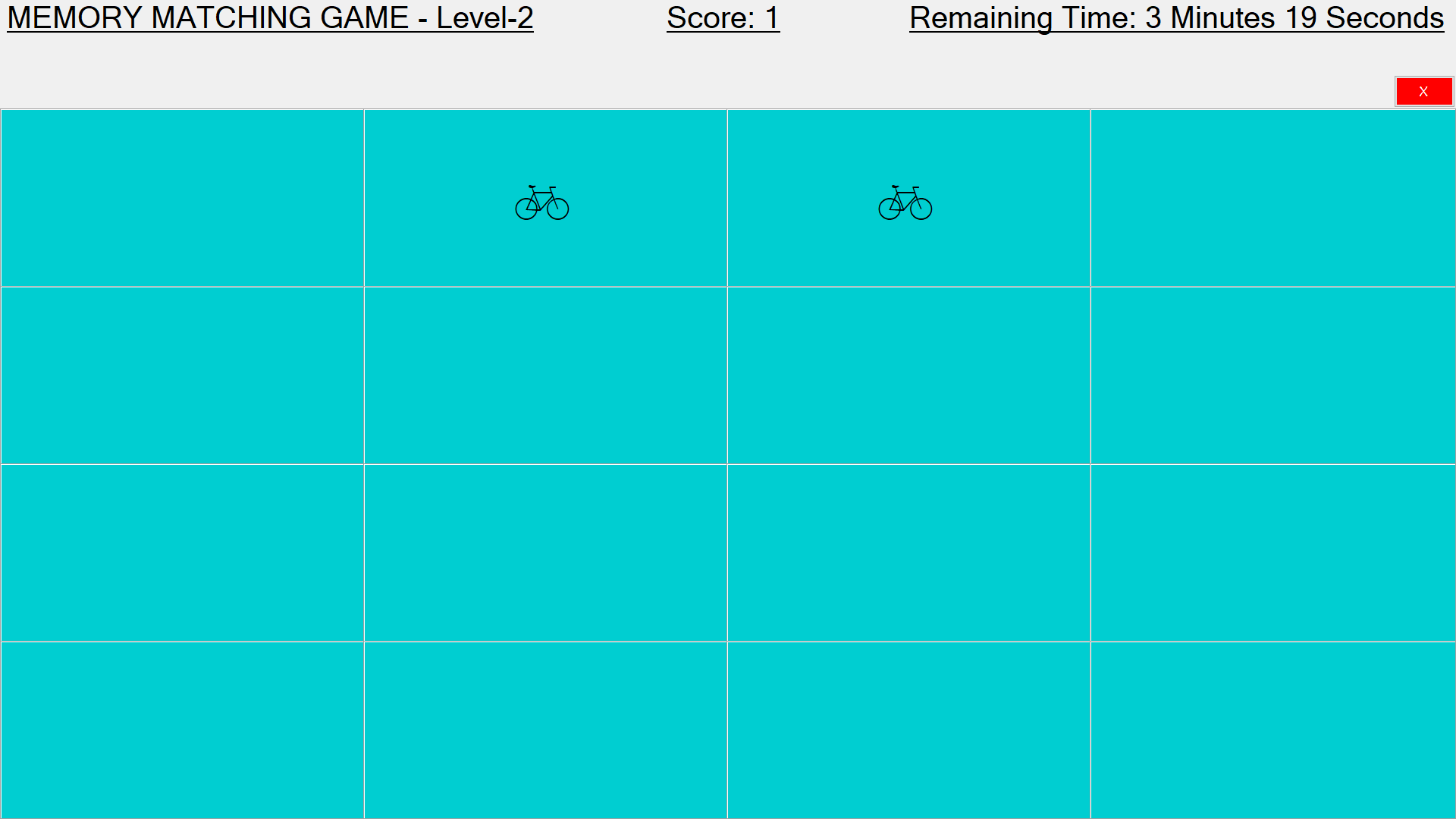
**Choose Custom Level:**



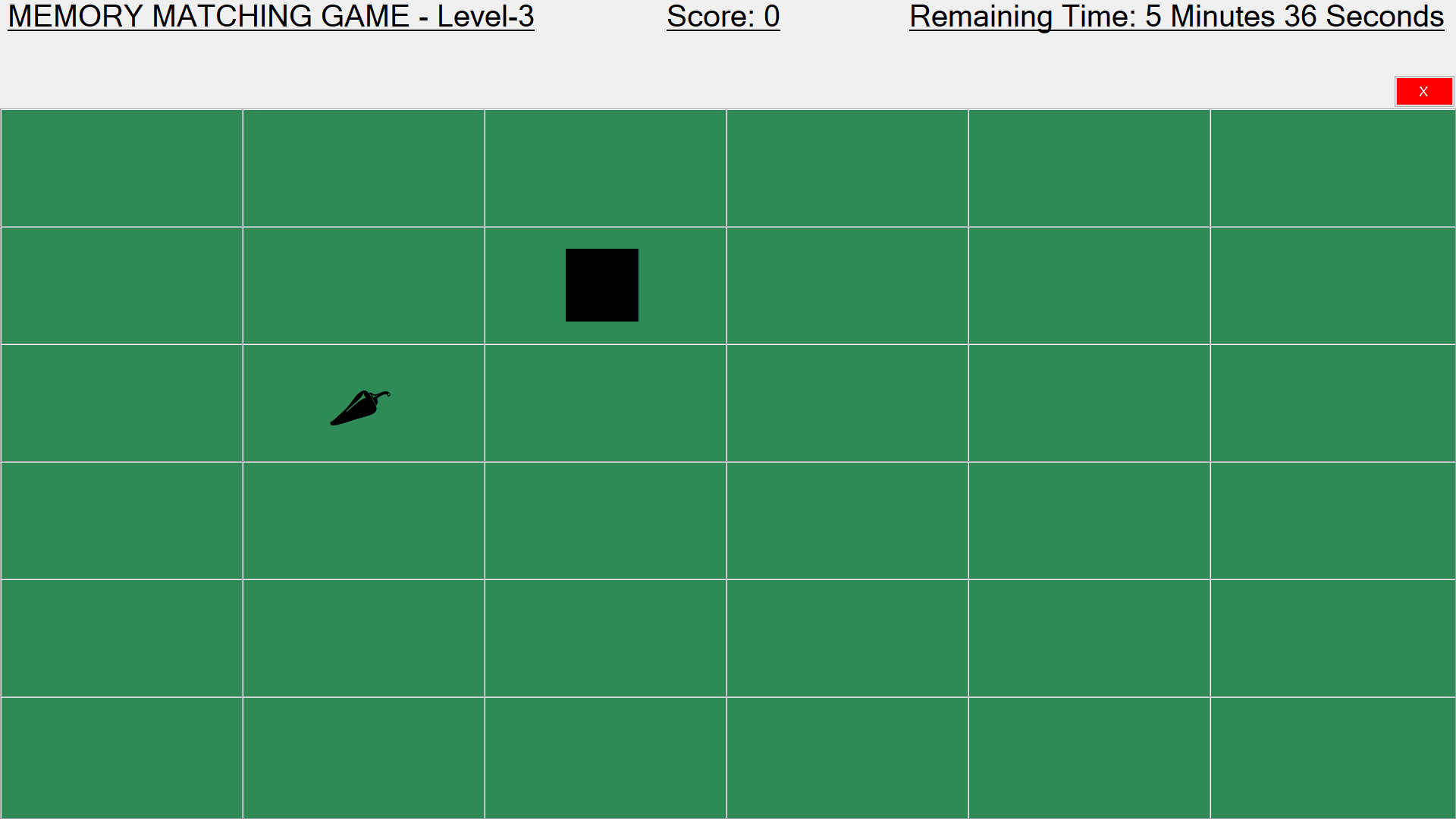
**Level-1:**



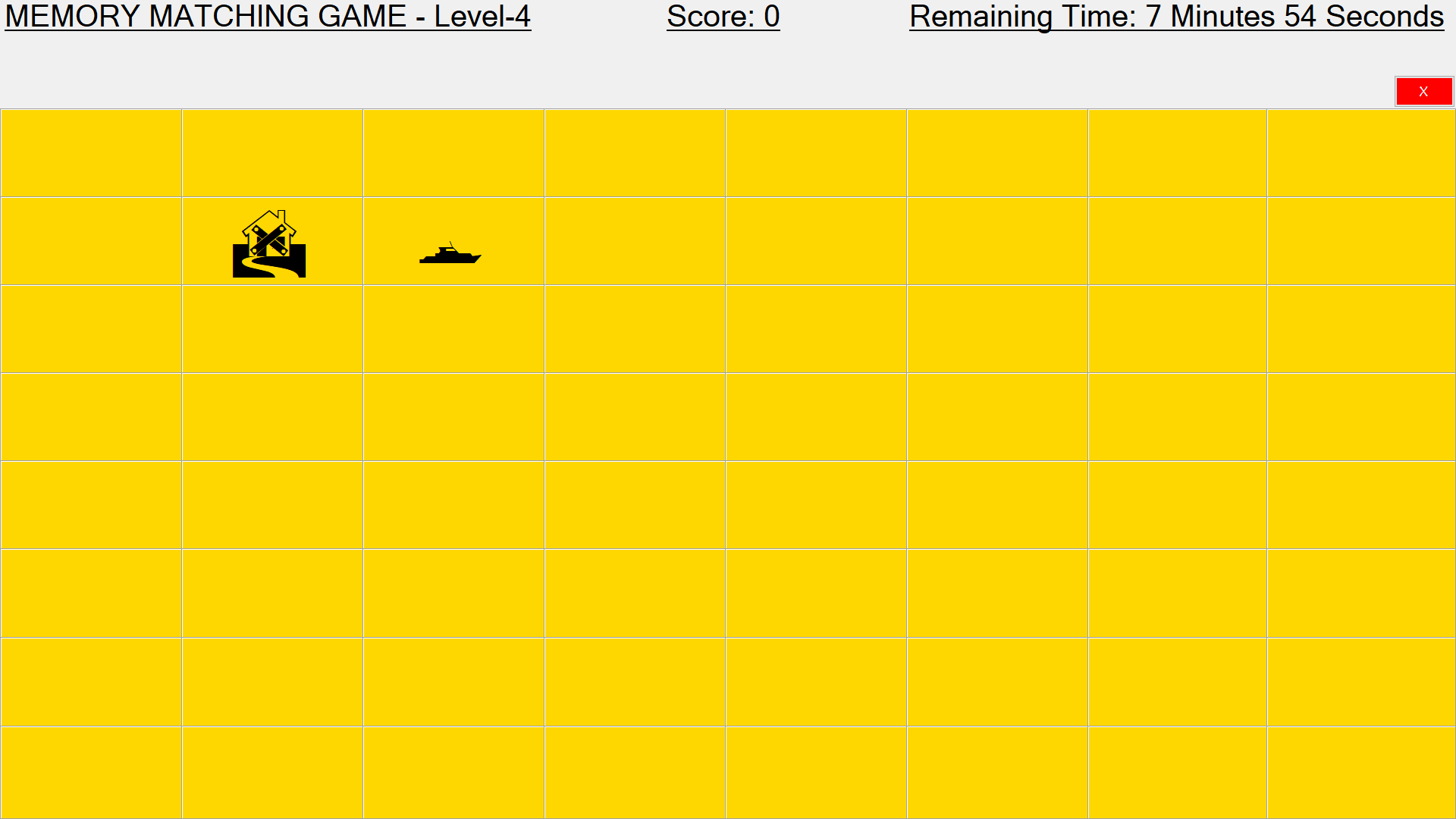
**Level-2:**



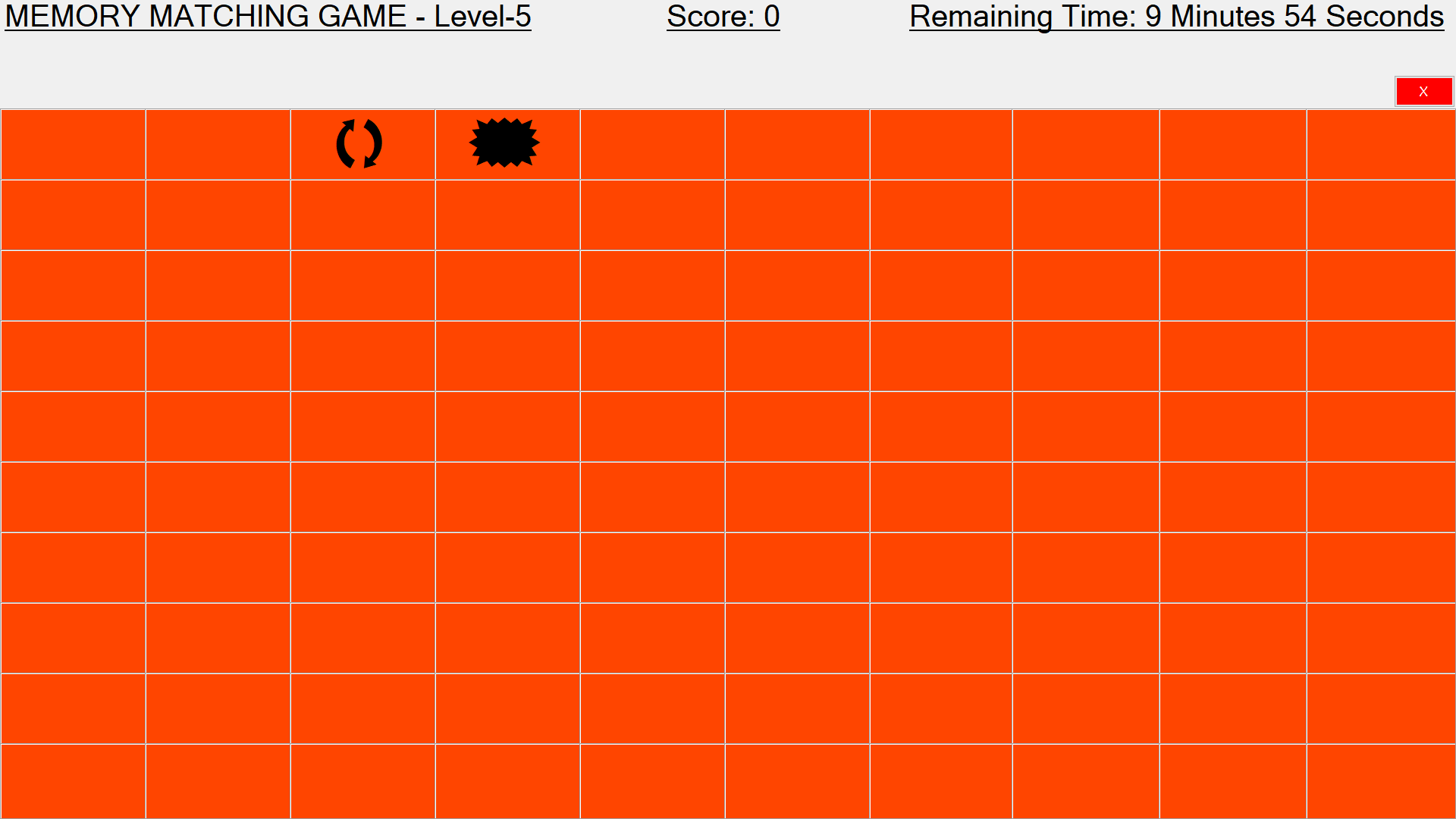
**Level-3:**



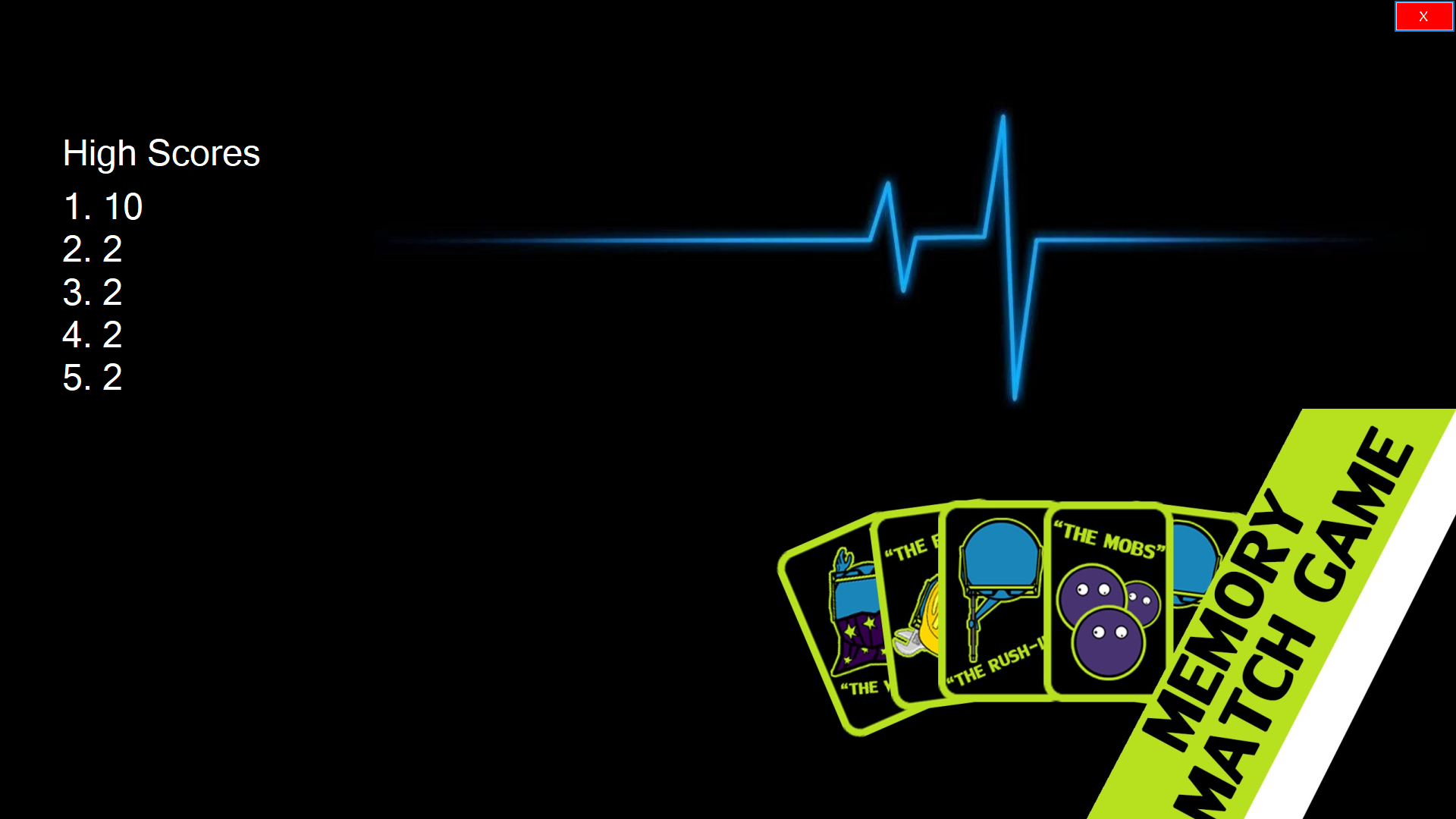
**Level-4:**



**Level-5:**



**High Score:**



**How To Play:**



**About:**



**Source Code**

**Menu.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

using System.Collections;

namespace MatchingGame

{

public partial class Menu : Form

{

public Menu()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

Lvl1\_2x2 f = new Lvl1\_2x2(0);

//this.Hide();

f.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

CustomLevel f = new CustomLevel();

//this.Hide();

f.Show();

}

private void button6\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void button3\_Click(object sender, EventArgs e)

{

Scores s = new Scores();

s.Show();

}

public int[] readScoreFile()

{

StreamReader readPlayerFile = new StreamReader("scores.txt"); //reading file to list

int[] a = new int[5] { 0, 0, 0, 0, 0 };

int i = 0;

while (!readPlayerFile.EndOfStream) //reading file till end

{

a[i] = int.Parse(readPlayerFile.ReadLine());

i++;

}

readPlayerFile.Close();

return a;

}

public void writeScoreFile(int []a)

{

StreamWriter writePlayerFile = new StreamWriter("scores.txt"); //writing the score file

writePlayerFile.WriteLine(a[0]);

writePlayerFile.WriteLine(a[1]);

writePlayerFile.WriteLine(a[2]);

writePlayerFile.WriteLine(a[3]);

writePlayerFile.WriteLine(a[4]);

writePlayerFile.Close();

}

private void button4\_Click(object sender, EventArgs e)

{

MessageBox.Show(" Game Name: MEMORY MATCHING GAME. \n Developer Name: Muhammad Anas Baig. \n University: Bahria University, Islamabad. \n Date Created: 10 December, 2017 12:24 AM", "About");

}

private void button5\_Click(object sender, EventArgs e)

{

MessageBox.Show("Help:\n 1. Select 'Choose Custom Level' OR 'Start New Game' to start the Game.\n 2. Click on each of the boxes and try to memorize shown pictures. \n 3. Click two same pictures consecutively to make it still. \n 4. Complete all the boxes. \n 5. After each Level you will be moved to next advanced Level. ", "How To Play");

}

}

}

**CustomLevel.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MatchingGame

{

public partial class CustomLevel : Form

{

public CustomLevel()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

Lvl1\_2x2 f = new Lvl1\_2x2(0);

//this.Hide();

f.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

Lvl2\_4x4 f = new Lvl2\_4x4(0);

//this.Hide();

f.Show();

}

private void button3\_Click(object sender, EventArgs e)

{

Lvl3\_6x6 f = new Lvl3\_6x6(0);

//this.Hide();

f.Show();

}

private void button4\_Click(object sender, EventArgs e)

{

Lvl4\_8x8 f = new Lvl4\_8x8(0);

//this.Hide();

f.Show();

}

private void button5\_Click(object sender, EventArgs e)

{

Lvl5\_10x10 f = new Lvl5\_10x10(0);

//this.Hide();

f.Show();

}

private void button6\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**Lvl1\_2x2.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MatchingGame

{

public partial class Lvl1\_2x2 : Form

{

static int time = 60;

static int i = 0;

// firstClicked points to the first Label control

// that the player clicks, but it will be null

// if the player hasn't clicked a label yet.

Label firstClicked = null;

// secondClicked points to the second Label control

// that the player clicks.

Label secondClicked = null;

// Use this Random object to choose random icons for the squares.

Random random = new Random();

// Each of these letters is an interesting icon

// in the Webdings font,

// and each icon appears twice in this list.

List<string> icons = new List<string>()

{

"!", "!", "N", "N"

};

/// <summary>

/// Assign each icon from the list of icons to a random square

/// </summary>

private void AssignIconsToSquares()

{

// The TableLayoutPanel has 16 labels,

// and the icon list has 16 icons,

// so an icon is pulled at random from the list

// and added to each label.

foreach (Control control in tableLayoutPanel1.Controls)

{

Label iconLabel = control as Label;

if (iconLabel != null)

{

int randomNumber = random.Next(icons.Count);

iconLabel.Text = icons[randomNumber];

iconLabel.ForeColor = iconLabel.BackColor;

icons.RemoveAt(randomNumber);

}

}

}

public Lvl1\_2x2(int i1)

{

InitializeComponent();

i = i1;

timer2.Start();

AssignIconsToSquares();

}

/// <summary>

/// Every label's Click event is handled by this event handler.

/// </summary>

/// <param name="sender">The label that was clicked.</param>

/// <param name="e"></param>

private void label\_Click(object sender, EventArgs e)

{

// The timer is only on after two non-matching

// icons have been shown to the player,

// so ignore any clicks if the timer is running

if (timer1.Enabled == true)

return;

Label clickedLabel = sender as Label;

if (clickedLabel != null)

{

// If the clicked label is black, the player clicked

// an icon that's already been revealed --

// ignore the click.

if (clickedLabel.ForeColor == Color.Black)

// All done - leave the if statements.

return;

// If firstClicked is null, this is the first icon

// in the pair that the player clicked,

// so set firstClicked to the label that the player

// clicked, change its color to black, and return.

if (firstClicked == null)

{

firstClicked = clickedLabel;

firstClicked.ForeColor = Color.Black;

// All done - leave the if statements.

return;

}

// If the player gets this far, the timer isn't

// running and firstClicked isn't null,

// so this must be the second icon the player clicked

// Set its color to black.

secondClicked = clickedLabel;

secondClicked.ForeColor = Color.Black;

if (firstClicked.Text == secondClicked.Text)

{

i++;

linkLabel3.Text = "Score: " + i.ToString();

}

// Check to see if the player won.

CheckForWinner();

// If the player clicked two matching icons, keep them

// black and reset firstClicked and secondClicked

// so the player can click another icon.

if (firstClicked.Text == secondClicked.Text)

{

firstClicked = null;

secondClicked = null;

return;

}

// If the player gets this far, the player

// clicked two different icons, so start the

// timer (which will wait three quarters of

// a second, and then hide the icons).

timer1.Start();

}

}

/// <summary>

/// This timer is started when the player clicks

/// two icons that don't match,

/// so it counts three quarters of a second

/// and then turns itself off and hides both icons.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void timer1\_Tick(object sender, EventArgs e)

{

// Stop the timer.

timer1.Stop();

// Hide both icons.

firstClicked.ForeColor = firstClicked.BackColor;

secondClicked.ForeColor = secondClicked.BackColor;

// Reset firstClicked and secondClicked

// so the next time a label is

// clicked, the program knows it's the first click.

firstClicked = null;

secondClicked = null;

}

/// <summary>

/// Check every icon to see if it is matched, by

/// comparing its foreground color to its background color.

/// If all of the icons are matched, the player wins.

/// </summary>

private void CheckForWinner()

{

// Go through all of the labels in the TableLayoutPanel,

// checking each one to see if its icon is matched.

foreach (Control control in tableLayoutPanel1.Controls)

{

Label iconLabel = control as Label;

if (iconLabel != null)

{

if (iconLabel.ForeColor == iconLabel.BackColor)

return;

}

}

// If the loop didn’t return, it didn't find

// any unmatched icons.

// That means the user won. Show a message and close the form.

MessageBox.Show("LEVEL-1 PASSED! You matched all the icons!", "Congratulations!");

Menu m = new Menu();

int[] a = m.readScoreFile();

bool check = false;

for (int j=0; j<5; j++)

{

if ((i > a[j]) && (check == false))

{

a[j] = i;

check = true;

}

}

m.writeScoreFile(a);

Lvl2\_4x4 f = new Lvl2\_4x4(i);

//this.Hide();

f.Show();

Close();

}

private void Form4\_Load(object sender, EventArgs e)

{

}

private void tableLayoutPanel1\_Paint(object sender, PaintEventArgs e)

{

}

private void timer2\_Tick(object sender, EventArgs e)

{

time--;

int minutes = time / 60;

int seconds = time - (minutes \* 60);

linkLabel1.Text = "Remaining Time: " + minutes.ToString() + " Minutes " + seconds.ToString() + " Seconds";

if (time == 0)

{

MessageBox.Show("Game Time Over");

this.Close();

}

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**Lvl2\_4x4.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MatchingGame

{

public partial class Lvl2\_4x4 : Form

{

static int time = 204;

static int i = 0;

// firstClicked points to the first Label control

// that the player clicks, but it will be null

// if the player hasn't clicked a label yet.

Label firstClicked = null;

// secondClicked points to the second Label control

// that the player clicks.

Label secondClicked = null;

// Use this Random object to choose random icons for the squares.

Random random = new Random();

// Each of these letters is an interesting icon

// in the Webdings font,

// and each icon appears twice in this list.

List<string> icons = new List<string>()

{

"!", "!", "N", "N", ",", ",", "k", "k",

"b", "b", "v", "v", "w", "w", "z", "z"

};

/// <summary>

/// Assign each icon from the list of icons to a random square

/// </summary>

private void AssignIconsToSquares()

{

// The TableLayoutPanel has 16 labels,

// and the icon list has 16 icons,

// so an icon is pulled at random from the list

// and added to each label.

foreach (Control control in tableLayoutPanel1.Controls)

{

Label iconLabel = control as Label;

if (iconLabel != null)

{

int randomNumber = random.Next(icons.Count);

iconLabel.Text = icons[randomNumber];

iconLabel.ForeColor = iconLabel.BackColor;

icons.RemoveAt(randomNumber);

}

}

}

//public Lvl2\_4x4()

//{

// InitializeComponent();

// timer2.Start();

// AssignIconsToSquares();

//}

public Lvl2\_4x4(int i1)

{

InitializeComponent();

i = i1;

timer2.Start();

AssignIconsToSquares();

}

/// <summary>

/// Every label's Click event is handled by this event handler.

/// </summary>

/// <param name="sender">The label that was clicked.</param>

/// <param name="e"></param>

private void label\_Click(object sender, EventArgs e)

{

// The timer is only on after two non-matching

// icons have been shown to the player,

// so ignore any clicks if the timer is running

if (timer1.Enabled == true)

return;

Label clickedLabel = sender as Label;

if (clickedLabel != null)

{

// If the clicked label is black, the player clicked

// an icon that's already been revealed --

// ignore the click.

if (clickedLabel.ForeColor == Color.Black)

// All done - leave the if statements.

return;

// If firstClicked is null, this is the first icon

// in the pair that the player clicked,

// so set firstClicked to the label that the player

// clicked, change its color to black, and return.

if (firstClicked == null)

{

firstClicked = clickedLabel;

firstClicked.ForeColor = Color.Black;

// All done - leave the if statements.

return;

}

// If the player gets this far, the timer isn't

// running and firstClicked isn't null,

// so this must be the second icon the player clicked

// Set its color to black.

secondClicked = clickedLabel;

secondClicked.ForeColor = Color.Black;

if (firstClicked.Text == secondClicked.Text)

{

i++;

linkLabel3.Text = "Score: " + i.ToString();

}

// Check to see if the player won.

CheckForWinner();

// If the player clicked two matching icons, keep them

// black and reset firstClicked and secondClicked

// so the player can click another icon.

if (firstClicked.Text == secondClicked.Text)

{

firstClicked = null;

secondClicked = null;

return;

}

// If the player gets this far, the player

// clicked two different icons, so start the

// timer (which will wait three quarters of

// a second, and then hide the icons).

timer1.Start();

}

}

/// <summary>

/// This timer is started when the player clicks

/// two icons that don't match,

/// so it counts three quarters of a second

/// and then turns itself off and hides both icons.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void timer1\_Tick(object sender, EventArgs e)

{

// Stop the timer.

timer1.Stop();

// Hide both icons.

firstClicked.ForeColor = firstClicked.BackColor;

secondClicked.ForeColor = secondClicked.BackColor;

// Reset firstClicked and secondClicked

// so the next time a label is

// clicked, the program knows it's the first click.

firstClicked = null;

secondClicked = null;

}

/// <summary>

/// Check every icon to see if it is matched, by

/// comparing its foreground color to its background color.

/// If all of the icons are matched, the player wins.

/// </summary>

private void CheckForWinner()

{

// Go through all of the labels in the TableLayoutPanel,

// checking each one to see if its icon is matched.

foreach (Control control in tableLayoutPanel1.Controls)

{

Label iconLabel = control as Label;

if (iconLabel != null)

{

if (iconLabel.ForeColor == iconLabel.BackColor)

return;

}

}

// If the loop didn’t return, it didn't find

// any unmatched icons.

// That means the user won. Show a message and close the form.

MessageBox.Show("LEVEL-2 PASSED! You matched all the icons!", "Congratulations!");

Menu m = new Menu();

int[] a = m.readScoreFile();

bool check = false;

for (int j = 0; j < 5; j++)

{

if ((i > a[j]) && (check == false))

{

a[j] = i;

check = true;

}

}

m.writeScoreFile(a);

Lvl3\_6x6 f = new Lvl3\_6x6(i);

//this.Hide();

f.Show();

Close();

}

private void timer2\_Tick(object sender, EventArgs e)

{

time--;

int minutes = time / 60;

int seconds = time - (minutes \* 60);

linkLabel1.Text = "Remaining Time: " + minutes.ToString() + " Minutes " + seconds.ToString() + " Seconds";

if (time == 0)

{

MessageBox.Show("Game Time Over");

this.Close();

}

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**Lvl3\_6x6.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MatchingGame

{

public partial class Lvl3\_6x6 : Form

{

static int time = 360;

static int i = 0;

// firstClicked points to the first Label control

// that the player clicks, but it will be null

// if the player hasn't clicked a label yet.

Label firstClicked = null;

// secondClicked points to the second Label control

// that the player clicks.

Label secondClicked = null;

// Use this Random object to choose random icons for the squares.

Random random = new Random();

// Each of these letters is an interesting icon

// in the Webdings font,

// and each icon appears twice in this list.

List<string> icons = new List<string>()

{

"!", "!", "N", "N", ",", ",", "k", "k",

"b", "b", "v", "v", "w", "w", "z", "z",

"c", "c", "d", "d", "e", "e", "f", "f",

"g", "g", "h", "h", "j", "j", "k", "k",

"l", "l", "q", "q"

};

/// <summary>

/// Assign each icon from the list of icons to a random square

/// </summary>

private void AssignIconsToSquares()

{

// The TableLayoutPanel has 16 labels,

// and the icon list has 16 icons,

// so an icon is pulled at random from the list

// and added to each label.

foreach (Control control in tableLayoutPanel1.Controls)

{

Label iconLabel = control as Label;

if (iconLabel != null)

{

int randomNumber = random.Next(icons.Count);

iconLabel.Text = icons[randomNumber];

iconLabel.ForeColor = iconLabel.BackColor;

icons.RemoveAt(randomNumber);

}

}

}

//public Lvl3\_6x6()

//{

// InitializeComponent();

// timer2.Start();

// AssignIconsToSquares();

//}

public Lvl3\_6x6(int i2)

{

InitializeComponent();

i = i2;

timer2.Start();

AssignIconsToSquares();

}

/// <summary>

/// Every label's Click event is handled by this event handler.

/// </summary>

/// <param name="sender">The label that was clicked.</param>

/// <param name="e"></param>

private void label\_Click(object sender, EventArgs e)

{

// The timer is only on after two non-matching

// icons have been shown to the player,

// so ignore any clicks if the timer is running

if (timer1.Enabled == true)

return;

Label clickedLabel = sender as Label;

if (clickedLabel != null)

{

// If the clicked label is black, the player clicked

// an icon that's already been revealed --

// ignore the click.

if (clickedLabel.ForeColor == Color.Black)

// All done - leave the if statements.

return;

// If firstClicked is null, this is the first icon

// in the pair that the player clicked,

// so set firstClicked to the label that the player

// clicked, change its color to black, and return.

if (firstClicked == null)

{

firstClicked = clickedLabel;

firstClicked.ForeColor = Color.Black;

// All done - leave the if statements.

return;

}

// If the player gets this far, the timer isn't

// running and firstClicked isn't null,

// so this must be the second icon the player clicked

// Set its color to black.

secondClicked = clickedLabel;

secondClicked.ForeColor = Color.Black;

if (firstClicked.Text == secondClicked.Text)

{

i++;

linkLabel3.Text = "Score: " + i.ToString();

}

// Check to see if the player won.

CheckForWinner();

// If the player clicked two matching icons, keep them

// black and reset firstClicked and secondClicked

// so the player can click another icon.

if (firstClicked.Text == secondClicked.Text)

{

firstClicked = null;

secondClicked = null;

return;

}

// If the player gets this far, the player

// clicked two different icons, so start the

// timer (which will wait three quarters of

// a second, and then hide the icons).

timer1.Start();

}

}

/// <summary>

/// This timer is started when the player clicks

/// two icons that don't match,

/// so it counts three quarters of a second

/// and then turns itself off and hides both icons.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void timer1\_Tick(object sender, EventArgs e)

{

// Stop the timer.

timer1.Stop();

// Hide both icons.

firstClicked.ForeColor = firstClicked.BackColor;

secondClicked.ForeColor = secondClicked.BackColor;

// Reset firstClicked and secondClicked

// so the next time a label is

// clicked, the program knows it's the first click.

firstClicked = null;

secondClicked = null;

}

/// <summary>

/// Check every icon to see if it is matched, by

/// comparing its foreground color to its background color.

/// If all of the icons are matched, the player wins.

/// </summary>

private void CheckForWinner()

{

// Go through all of the labels in the TableLayoutPanel,

// checking each one to see if its icon is matched.

foreach (Control control in tableLayoutPanel1.Controls)

{

Label iconLabel = control as Label;

if (iconLabel != null)

{

if (iconLabel.ForeColor == iconLabel.BackColor)

return;

}

}

// If the loop didn’t return, it didn't find

// any unmatched icons.

// That means the user won. Show a message and close the form.

MessageBox.Show("LEVEL-3 PASSED! You matched all the icons!", "Congratulations!");

Menu m = new Menu();

int[] a = m.readScoreFile();

bool check = false;

for (int j = 0; j < 5; j++)

{

if ((i > a[j]) && (check == false))

{

a[j] = i;

check = true;

}

}

m.writeScoreFile(a);

Lvl4\_8x8 f = new Lvl4\_8x8(i);

//this.Hide();

f.Show();

Close();

}

private void timer2\_Tick(object sender, EventArgs e)

{

time--;

int minutes = time / 60;

int seconds = time - (minutes \* 60);

linkLabel1.Text = "Remaining Time: " + minutes.ToString() + " Minutes " + seconds.ToString() + " Seconds";

if (time == 0)

{

MessageBox.Show("Game Time Over");

this.Close();

}

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**Lvl4\_8x8.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MatchingGame

{

public partial class Lvl4\_8x8 : Form

{

static int time = 480;

static int i = 0;

// firstClicked points to the first Label control

// that the player clicks, but it will be null

// if the player hasn't clicked a label yet.

Label firstClicked = null;

// secondClicked points to the second Label control

// that the player clicks.

Label secondClicked = null;

// Use this Random object to choose random icons for the squares.

Random random = new Random();

// Each of these letters is an interesting icon

// in the Webdings font,

// and each icon appears twice in this list.

List<string> icons = new List<string>()

{

"!", "!", "N", "N", ",", ",", "k", "k",

"b", "b", "v", "v", "w", "w", "z", "z",

"c", "c", "d", "d", "e", "e", "f", "f",

"g", "g", "h", "h", "j", "j", "k", "k",

"l", "l", "q", "q", "r", "r", "s", "s",

"t", "t", "u", "u", "p", "p", "o", "o",

"y", "y", "\*", "\*", "A", "A", "B", "B",

"C", "C", "D", "D", "E", "E", "F", "F"

};

/// <summary>

/// Assign each icon from the list of icons to a random square

/// </summary>

private void AssignIconsToSquares()

{

// The TableLayoutPanel has 16 labels,

// and the icon list has 16 icons,

// so an icon is pulled at random from the list

// and added to each label.

foreach (Control control in tableLayoutPanel1.Controls)

{

Label iconLabel = control as Label;

if (iconLabel != null)

{

int randomNumber = random.Next(icons.Count);

iconLabel.Text = icons[randomNumber];

iconLabel.ForeColor = iconLabel.BackColor;

icons.RemoveAt(randomNumber);

}

}

}

//public Lvl4\_8x8()

//{

// InitializeComponent();

// timer2.Start();

// AssignIconsToSquares();

//}

public Lvl4\_8x8(int i3)

{

InitializeComponent();

i = i3;

timer2.Start();

AssignIconsToSquares();

}

/// <summary>

/// Every label's Click event is handled by this event handler.

/// </summary>

/// <param name="sender">The label that was clicked.</param>

/// <param name="e"></param>

private void label\_Click(object sender, EventArgs e)

{

// The timer is only on after two non-matching

// icons have been shown to the player,

// so ignore any clicks if the timer is running

if (timer1.Enabled == true)

return;

Label clickedLabel = sender as Label;

if (clickedLabel != null)

{

// If the clicked label is black, the player clicked

// an icon that's already been revealed --

// ignore the click.

if (clickedLabel.ForeColor == Color.Black)

// All done - leave the if statements.

return;

// If firstClicked is null, this is the first icon

// in the pair that the player clicked,

// so set firstClicked to the label that the player

// clicked, change its color to black, and return.

if (firstClicked == null)

{

firstClicked = clickedLabel;

firstClicked.ForeColor = Color.Black;

// All done - leave the if statements.

return;

}

// If the player gets this far, the timer isn't

// running and firstClicked isn't null,

// so this must be the second icon the player clicked

// Set its color to black.

secondClicked = clickedLabel;

secondClicked.ForeColor = Color.Black;

if (firstClicked.Text == secondClicked.Text)

{

i++;

linkLabel3.Text = "Score: " + i.ToString();

}

// Check to see if the player won.

CheckForWinner();

// If the player clicked two matching icons, keep them

// black and reset firstClicked and secondClicked

// so the player can click another icon.

if (firstClicked.Text == secondClicked.Text)

{

firstClicked = null;

secondClicked = null;

return;

}

// If the player gets this far, the player

// clicked two different icons, so start the

// timer (which will wait three quarters of

// a second, and then hide the icons).

timer1.Start();

}

}

/// <summary>

/// This timer is started when the player clicks

/// two icons that don't match,

/// so it counts three quarters of a second

/// and then turns itself off and hides both icons.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void timer1\_Tick(object sender, EventArgs e)

{

// Stop the timer.

timer1.Stop();

// Hide both icons.

firstClicked.ForeColor = firstClicked.BackColor;

secondClicked.ForeColor = secondClicked.BackColor;

// Reset firstClicked and secondClicked

// so the next time a label is

// clicked, the program knows it's the first click.

firstClicked = null;

secondClicked = null;

}

/// <summary>

/// Check every icon to see if it is matched, by

/// comparing its foreground color to its background color.

/// If all of the icons are matched, the player wins.

/// </summary>

private void CheckForWinner()

{

// Go through all of the labels in the TableLayoutPanel,

// checking each one to see if its icon is matched.

foreach (Control control in tableLayoutPanel1.Controls)

{

Label iconLabel = control as Label;

if (iconLabel != null)

{

if (iconLabel.ForeColor == iconLabel.BackColor)

return;

}

}

// If the loop didn’t return, it didn't find

// any unmatched icons.

// That means the user won. Show a message and close the form.

MessageBox.Show("LEVEL-4 PASSED! You matched all the icons!", "Congratulations!");

Menu m = new Menu();

int[] a = m.readScoreFile();

bool check = false;

for (int j = 0; j < 5; j++)

{

if ((i > a[j]) && (check == false))

{

a[j] = i;

check = true;

}

}

m.writeScoreFile(a);

Lvl5\_10x10 f = new Lvl5\_10x10(i);

//this.Hide();

f.Show();

Close();

}

private void Form3\_Load(object sender, EventArgs e)

{

}

private void timer2\_Tick(object sender, EventArgs e)

{

time--;

int minutes = time / 60;

int seconds = time - (minutes \* 60);

linkLabel1.Text = "Remaining Time: " + minutes.ToString() + " Minutes " + seconds.ToString() + " Seconds";

if (time == 0)

{

MessageBox.Show("Game Time Over");

this.Close();

}

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**Lvl5\_10x10.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MatchingGame

{

public partial class Lvl5\_10x10 : Form

{

static int time = 600;

static int i = 0;

// firstClicked points to the first Label control

// that the player clicks, but it will be null

// if the player hasn't clicked a label yet.

Label firstClicked = null;

// secondClicked points to the second Label control

// that the player clicks.

Label secondClicked = null;

// Use this Random object to choose random icons for the squares.

Random random = new Random();

// Each of these letters is an interesting icon

// in the Webdings font,

// and each icon appears twice in this list.

List<string> icons = new List<string>()

{

"!", "!", "0", "0", ",", ",", "k", "k",

"b", "b", "v", "v", "w", "w", "z", "z",

"c", "c", "d", "d", "e", "e", "f", "f",

"g", "g", "h", "h", "j", "j", "k", "k",

"l", "l", "q", "q", "r", "r", "s", "s",

"t", "t", "u", "u", "p", "p", "o", "o",

"y", "y", "\*", "\*", "A", "A", "B", "B",

"C", "C", "D", "D", "E", "E", "F", "F",

"G", "G", "H", "H", "I", "I", "J", "J",

"K", "K", "L", "L", "M", "M", "N", "N",

"O", "O", "P", "P", "Q", "Q", "R", "R",

"S", "S", "T", "T", "U", "U", "V", "V",

"W", "W", "X", "X"

};

/// <summary>

/// Assign each icon from the list of icons to a random square

/// </summary>

private void AssignIconsToSquares()

{

// The TableLayoutPanel has 16 labels,

// and the icon list has 16 icons,

// so an icon is pulled at random from the list

// and added to each label.

foreach (Control control in tableLayoutPanel1.Controls)

{

Label iconLabel = control as Label;

if (iconLabel != null)

{

int randomNumber = random.Next(icons.Count);

iconLabel.Text = icons[randomNumber];

iconLabel.ForeColor = iconLabel.BackColor;

icons.RemoveAt(randomNumber);

}

}

}

//public Lvl5\_10x10()

//{

// InitializeComponent();

// timer2.Start();

// AssignIconsToSquares();

//}

public Lvl5\_10x10(int i4)

{

InitializeComponent();

i = i4;

timer2.Start();

AssignIconsToSquares();

}

/// <summary>

/// Every label's Click event is handled by this event handler.

/// </summary>

/// <param name="sender">The label that was clicked.</param>

/// <param name="e"></param>

private void label\_Click(object sender, EventArgs e)

{

// The timer is only on after two non-matching

// icons have been shown to the player,

// so ignore any clicks if the timer is running

if (timer1.Enabled == true)

return;

Label clickedLabel = sender as Label;

if (clickedLabel != null)

{

// If the clicked label is black, the player clicked

// an icon that's already been revealed --

// ignore the click.

if (clickedLabel.ForeColor == Color.Black)

// All done - leave the if statements.

return;

// If firstClicked is null, this is the first icon

// in the pair that the player clicked,

// so set firstClicked to the label that the player

// clicked, change its color to black, and return.

if (firstClicked == null)

{

firstClicked = clickedLabel;

firstClicked.ForeColor = Color.Black;

// All done - leave the if statements.

return;

}

// If the player gets this far, the timer isn't

// running and firstClicked isn't null,

// so this must be the second icon the player clicked

// Set its color to black.

secondClicked = clickedLabel;

secondClicked.ForeColor = Color.Black;

if (firstClicked.Text == secondClicked.Text)

{

i++;

linkLabel3.Text = "Score: " + i.ToString();

}

// Check to see if the player won.

CheckForWinner();

// If the player clicked two matching icons, keep them

// black and reset firstClicked and secondClicked

// so the player can click another icon.

if (firstClicked.Text == secondClicked.Text)

{

firstClicked = null;

secondClicked = null;

return;

}

// If the player gets this far, the player

// clicked two different icons, so start the

// timer (which will wait three quarters of

// a second, and then hide the icons).

timer1.Start();

}

}

/// <summary>

/// This timer is started when the player clicks

/// two icons that don't match,

/// so it counts three quarters of a second

/// and then turns itself off and hides both icons.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void timer1\_Tick(object sender, EventArgs e)

{

// Stop the timer.

timer1.Stop();

// Hide both icons.

firstClicked.ForeColor = firstClicked.BackColor;

secondClicked.ForeColor = secondClicked.BackColor;

// Reset firstClicked and secondClicked

// so the next time a label is

// clicked, the program knows it's the first click.

firstClicked = null;

secondClicked = null;

}

/// <summary>

/// Check every icon to see if it is matched, by

/// comparing its foreground color to its background color.

/// If all of the icons are matched, the player wins.

/// </summary>

private void CheckForWinner()

{

// Go through all of the labels in the TableLayoutPanel,

// checking each one to see if its icon is matched.

foreach (Control control in tableLayoutPanel1.Controls)

{

Label iconLabel = control as Label;

if (iconLabel != null)

{

if (iconLabel.ForeColor == iconLabel.BackColor)

return;

}

}

// If the loop didn’t return, it didn't find

// any unmatched icons.

// That means the user won. Show a message and close the form.

MessageBox.Show("You matched all the icons!", "Congratulations! GAME END");

Menu m = new Menu();

int[] a = m.readScoreFile();

bool check = false;

for (int j = 0; j < 5; j++)

{

if ((i > a[j]) && (check == false))

{

a[j] = i;

check = true;

}

}

m.writeScoreFile(a);

Close();

}

private void Form3\_Load(object sender, EventArgs e)

{

}

private void timer2\_Tick(object sender, EventArgs e)

{

time--;

int minutes = time / 60;

int seconds = time - (minutes \* 60);

linkLabel1.Text = "Remaining Time: " + minutes.ToString() + " Minutes " + seconds.ToString() + " Seconds";

if (time == 0)

{

MessageBox.Show("Game Time Over");

this.Close();

}

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

}

private void Lvl5\_10x10\_Load(object sender, EventArgs e)

{

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**Program.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MatchingGame

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Menu());

}

}

}