

ÖDEV-1

Yazılım Sınama – YZM 3105



Mert Oğuz DALAK – 182803037

Ömer Furkan BİLGE – 182803052

İÇERİK

* Uygulamanın Amacı
* Girdiler
* Uygulamanın Mantığı
* Programın Kodları
* Ekran Çıktıları
* Test Caseler

Uygulamanın Amacı

Bu uygulamanın amacı, dizilen taşları en kısa yoldan ana taş üstünde biriktirip oyunu kazanmaktır.

Girdiler

Hedef taş, birinci oyuncu taşı, ikinci oyuncu taşı, engel

Uygulamanın Mantığı

Uygulamanın mantığı,kullanıcı taşları dizdikten sonra hamle sayısı bitmeden taşları ana taşta biriktirip oyunu kazanmaya çalışmaktır.

Programın Kodları

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Security.AccessControl;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TasToplama

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

int n, x;

int r = 0;

int player1\_win, player2\_win;

int[,] piece\_num = new int[10, 10];

PictureBox[,] P;

int target\_locx, target\_locy;

int distance\_x, distance\_y;

int board\_x1, board\_y1;

private void Form1\_Load(object sender, EventArgs e)

{

n = 10;

P = new PictureBox[n, n];

int left = 2, top = 2;

Color[] colors = new Color[] { Color.White, Color.Black };

btn\_target.Enabled = false;

btn\_wall.Enabled = false;

btn\_player1.Enabled = false;

btn\_player2.Enabled = false;

for (int i = 0; i < n; i++)

{

left = 2;

if (i % 2 == 0) { colors[0] = Color.White; colors[1] = Color.Black; }

else { colors[0] = Color.Black; colors[1] = Color.White; }

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

{

P[j, i] = new PictureBox();

P[j, i].BackColor = colors[(j % 2 == 0) ? 1 : 0];

P[j, i].Location = new Point(left, top);

P[j, i].Size = new Size(70, 70);

left += 70;

P[j, i].Name = j + " " + i;

P[j, i].SizeMode = PictureBoxSizeMode.CenterImage;

P[j, i].MouseClick += (sender2, e2) =>

{

int board\_x, board\_y;

PictureBox p = sender2 as PictureBox;

if (r == 1)

{

if (p.Image != null)

{

board\_x = Calculator(p.Location.X);

board\_y = Calculator(p.Location.Y);

if (P[board\_x, board\_y].Name == "player1" || P[board\_x, board\_y].Name == "player2")

{

board\_x1 = board\_x;

board\_y1 = board\_y;

Cleaner();

if (board\_x > 0)

{

if (P[board\_x - 1, board\_y].Image == null)

{

P[board\_x - 1, board\_y].Image = Properties.Resources.selection;

P[board\_x - 1, board\_y].Name = "selector";

}

else if (P[board\_x - 1, board\_y].Name == "target")

{

P[board\_x - 1, board\_y].Name = "target2";

}

}

if (board\_x < 9)

{

if (P[board\_x + 1, board\_y].Image == null)

{

P[board\_x + 1, board\_y].Image = Properties.Resources.selection;

P[board\_x + 1, board\_y].Name = "selector";

}

else if (P[board\_x + 1, board\_y].Name == "target")

{

P[board\_x + 1, board\_y].Name = "target2";

}

}

if ( board\_y > 0)

{

if (P[board\_x, board\_y - 1].Image == null)

{

P[board\_x, board\_y - 1].Image = Properties.Resources.selection;

P[board\_x, board\_y - 1].Name = "selector";

}

else if (P[board\_x, board\_y - 1].Name == "target")

{

P[board\_x, board\_y - 1].Name = "target2";

}

}

if (board\_y < 9 )

{

if (P[board\_x, board\_y + 1].Image == null)

{

P[board\_x, board\_y + 1].Image = Properties.Resources.selection;

P[board\_x, board\_y + 1].Name = "selector";

}

else if (P[board\_x, board\_y + 1].Name == "target")

{

P[board\_x, board\_y + 1].Name = "target2";

}

}

}

if (p.Name == "selector")

{

board\_x = Calculator(p.Location.X);

board\_y = Calculator(p.Location.Y);

P[board\_x, board\_y].Name = P[board\_x1, board\_y1].Name;

P[board\_x1, board\_y1].Name = board\_x1 + " " + board\_y1;

if (P[board\_x, board\_y].Name == "player1")

{

p.Image = Properties.Resources.blue\_player;

P[board\_x1, board\_y1].Image = null;

Distance\_Printer(board\_x1, board\_y1, p);

}

else if (P[board\_x, board\_y].Name == "player2")

{

p.Image = Properties.Resources.brown\_player;

P[board\_x1, board\_y1].Image = null;

Distance\_Printer(board\_x1, board\_y1, p);

}

Cleaner();

}

else if (p.Name == "target2")

{

board\_x = Calculator(p.Location.X);

board\_y = Calculator(p.Location.Y);

if (P[board\_x1, board\_y1].Name == "player1")

{

player1\_win--;

if (player1\_win == 0)

{

lbl\_whowin.Text = " Player 1 WON! CONGRATULATIONS! ";

timer1.Enabled = true;

btn\_start.Enabled = false;

G.Enabled = false;

}

}

else if (P[board\_x1, board\_y1].Name == "player2")

{

player2\_win--;

if (player2\_win == 0)

{

lbl\_whowin.Text = " Player 2 WON! CONGRATULATIONS! ";

timer1.Enabled = true;

btn\_start.Enabled = false;

G.Enabled = false;

}

}

P[board\_x1, board\_y1].Name = board\_x1 + " " + board\_y1;

P[board\_x1, board\_y1].Image = null;

p.Name = "target";

Cleaner();

}

}

}

else

{

if (p.Image == null)

{

int c;

switch (x)

{

case 1:

if (Convert.ToInt32(lbl\_target\_counter.Text) > 0)

{

c = Convert.ToInt32(lbl\_target\_counter.Text);

c--;

if (c == 0)

{

btn\_target.Enabled = false;

btn\_wall.Enabled = true;

}

lbl\_target\_counter.Text = Convert.ToString(c);

board\_x = Calculator(p.Location.X);

board\_y = Calculator(p.Location.Y);

P[board\_x, board\_y].Name = "target";

p.Image = Properties.Resources.colored\_checker;

target\_locx = p.Location.X;

target\_locy = p.Location.Y;

}

break;

case 2:

if (Convert.ToInt32(lbl\_wall\_counter.Text) > 0)

{

c = Convert.ToInt32(lbl\_wall\_counter.Text);

c--;

if (c == 0)

{

btn\_wall.Enabled = false;

btn\_player1.Enabled = true;

}

lbl\_wall\_counter.Text = Convert.ToString(c);

board\_x = Calculator(p.Location.X);

board\_y = Calculator(p.Location.Y);

P[board\_x, board\_y].Name = "wall";

p.Image = Properties.Resources.wall;

}

break;

case 3:

if (Convert.ToInt32(lbl\_player1\_counter.Text) > 0)

{

c = Convert.ToInt32(lbl\_player1\_counter.Text);

c--;

if (c == 0)

{

btn\_player1.Enabled = false;

btn\_player2.Enabled = true;

if (rdb\_multiplayer.Checked == true)

{

btn\_start.Enabled = false;

}

else

{

btn\_start.Enabled = true;

}

}

lbl\_player1\_counter.Text = Convert.ToString(c);

board\_x = Calculator(p.Location.X);

board\_y = Calculator(p.Location.Y);

P[board\_x, board\_y].Name = "player1";

p.Image = Properties.Resources.blue\_player;

p.Paint += new PaintEventHandler((sender3, e3) =>

{

distance\_x = p.Location.X - target\_locx;

distance\_y = p.Location.Y - target\_locy;

piece\_num[board\_x, board\_y] = distance\_calculator(distance\_x, distance\_y);

string text = Convert.ToString(distance\_calculator(distance\_x, distance\_y));

SizeF textSize = e3.Graphics.MeasureString(text, Font);

PointF locationToDraw = new PointF();

locationToDraw.X = (p.Width / 2) - (textSize.Width / 2);

locationToDraw.Y = (p.Height / 2) - (textSize.Height / 2);

e3.Graphics.DrawString(text, Font, Brushes.Red, locationToDraw);

});

}

break;

case 4:

if (Convert.ToInt32(lbl\_player2\_counter.Text) > 0)

{

c = Convert.ToInt32(lbl\_player2\_counter.Text);

c--;

if (c == 0)

{

btn\_player2.Enabled = false;

btn\_start.Enabled = true;

}

lbl\_player2\_counter.Text = Convert.ToString(c);

board\_x = Calculator(p.Location.X);

board\_y = Calculator(p.Location.Y);

P[board\_x, board\_y].Name = "player2";

p.Image = Properties.Resources.brown\_player;

p.Paint += new PaintEventHandler((sender4, e4) =>

{

distance\_x = p.Location.X - target\_locx;

distance\_y = p.Location.Y - target\_locy;

piece\_num[board\_x, board\_y] = distance\_calculator(distance\_x, distance\_y);

string text = Convert.ToString(distance\_calculator(distance\_x, distance\_y));

SizeF textSize = e4.Graphics.MeasureString(text, Font);

PointF locationToDraw = new PointF();

locationToDraw.X = (p.Width / 2) - (textSize.Width / 2);

locationToDraw.Y = (p.Height / 2) - (textSize.Height / 2);

e4.Graphics.DrawString(text, Font, Brushes.Red, locationToDraw);

});

}

break;

}

}

}

};

G.Controls.Add(P[j, i]);

}

top += 70;

}

}

public int distance\_calculator(int h, int p)

{

if (h < 0)

{

h = h \* -1;

}

if (p < 0)

{

p = p \* -1;

}

return (p + h) / 70;

}

private int Calculator(int x)

{

x = x - 2;

if (x == 0)

{

return 0;

}

else

return x / 70;

}

public void Cleaner()

{

int x1, y1;

for(x1 = 0; x1 < 10; x1++ )

{

for(y1 = 0; y1 < 10; y1++)

{

if(P[x1 , y1].Name == "selector")

{

P[x1, y1].Image = null;

}

if ( P[x1, y1].Name == "target2")

{

P[x1, y1].Name = "target";

}

}

}

}

/\*

private void Distance\_Cleaner (int x , int y)

{

P[x, y].Paint += new PaintEventHandler((sender3, e3) =>

{

string text = null;

SizeF textSize = e3.Graphics.MeasureString(text, Font);

PointF locationToDraw = new PointF();

locationToDraw.X = (P[x, y].Width / 2) - (textSize.Width / 2);

locationToDraw.Y = (P[x, y].Height / 2) - (textSize.Height / 2);

e3.Graphics.DrawString(text, Font, Brushes.Red, locationToDraw);

});

}

\*/

private void Distance\_Printer(int x, int y, PictureBox p)

{

p.Paint += new PaintEventHandler((sender3, e3) =>

{

string text = Convert.ToString(piece\_num[x, y] - 1);

piece\_num[Calculator(p.Location.X), Calculator(p.Location.Y)] = piece\_num[x, y] - 1;

if (piece\_num[Calculator(p.Location.X), Calculator(p.Location.Y)] == 0)

{

if (P[Calculator(p.Location.X), Calculator(p.Location.Y)].Name == "player1")

{

lbl\_whowin.Text = " Player 1 Lost. GAME OVER! Press Restart! ";

timer1.Enabled = true;

btn\_start.Enabled = false;

G.Enabled = false;

}

else if (P[Calculator(p.Location.X), Calculator(p.Location.Y)].Name == "player2")

{

lbl\_whowin.Text = " Player 2 Lost. GAME OVER! Press Restart! ";

timer1.Enabled = true;

btn\_start.Enabled = false;

G.Enabled = false;

}

}

SizeF textSize = e3.Graphics.MeasureString(text, Font);

PointF locationToDraw = new PointF();

locationToDraw.X = (p.Width / 2) - (textSize.Width / 2);

locationToDraw.Y = (p.Height / 2) - (textSize.Height / 2);

e3.Graphics.DrawString(text, Font, Brushes.Red, locationToDraw);

});

}

private void btn\_target\_Click(object sender, EventArgs e)

{

x = 1;

}

private void btn\_wall\_Click(object sender, EventArgs e)

{

x = 2;

}

private void btn\_player1\_Click(object sender, EventArgs e)

{

x = 3;

}

private void btn\_player2\_Click(object sender, EventArgs e)

{

x = 4;

}

private void timer1\_Tick\_1(object sender, EventArgs e)

{

lbl\_whowin.Text = lbl\_whowin.Text.Substring(1) + lbl\_whowin.Text.Substring(0, 1);

}

private void btn\_start\_Click(object sender, EventArgs e)

{

r = 1;

btn\_start.Enabled = false;

}

private void btn\_replay\_Click(object sender, EventArgs e)

{

timer1.Enabled = false;

Application.Restart();

}

private void btn\_tasdiz\_Click(object sender, EventArgs e)

{

if (rdb\_singleplayer.Checked == true)

{

label4.Visible = false;

btn\_player2.Visible = false;

lbl\_player2\_counter.Visible = false;

btn\_target.Enabled = true;

groupBox3.Enabled = false;

}

if (rdb\_multiplayer.Checked == true)

{

label4.Visible = true;

btn\_player2.Visible = true;

lbl\_player2\_counter.Visible = true;

btn\_target.Enabled = true;

groupBox3.Enabled = false;

}

if(rdb\_level1.Checked == true)

{

lbl\_player1\_counter.Text = "6";

lbl\_player2\_counter.Text = "6";

lbl\_wall\_counter.Text = "3";

player1\_win = 6;

player2\_win = 6;

}

if(rdb\_level2.Checked == true)

{

lbl\_player1\_counter.Text = "7";

lbl\_player2\_counter.Text = "7";

lbl\_wall\_counter.Text = "4";

player1\_win = 7;

player2\_win = 7;

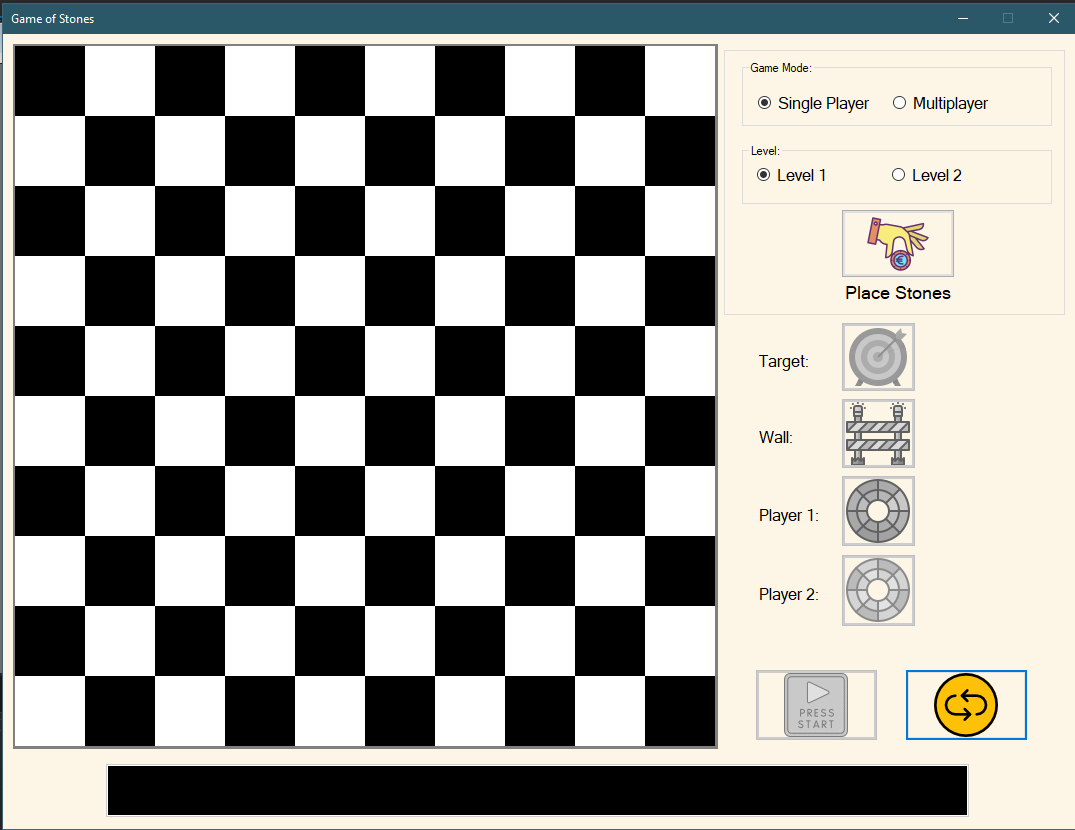
}

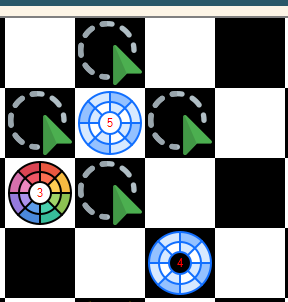
lbl\_target\_counter.Visible = true;

}

}

}

Ekran Çıktıları





Test Caseler

**Test Case 1.**

|  |  |
| --- | --- |
| Risk Level | Orta |
| Purpose | Bu test durumunda bütün taşların tahtaya yerleştirilmeden oyunun başlayıp başlamama durumu test edilmektedir. |
| Inputs | Left Click, Taşların Butonu, Başlama Butonu |
| Expected Outputs | Taşlar yerleştirilmeden oyunun başlamaması. |
| Pass Criterias | Taşlar yerleştirilince oyunun başlaması. |
| Fail Criterias | Taşların hepsi yerleştirilmeden oyunun başlaması. |
| Test Procedure | Test kullanıcısı; yazılımı, uygun cihazda çalıştırarak  belirtilen girdilerle testi gerçekleştirmelidir. Test işlemini tamamladıktan  sonra test sonucunu Pass/Fail olarak belirterek nedenleri ile birlikte raporlamalıdır. |

**- Taşların tahtaya yerleştirilmeden oyunun başlaması.**

**Test Case 2.**

|  |  |
| --- | --- |
| Risk Level | Düşük |
| Purpose | Bu test durumunda kullanıcı taşları oynattıktan sonra kalan hamle hesaplama fonksiyonunun doğru çalışıp çalışmadığı durumu test edilmektedir. |
| Inputs | Left Click |
| Expected Outputs | Hamle hesaplama algoritmasının sorunsuz çalışması |
| Pass Criterias | Hamlelerin yeni halleriyle sorunsuz bir şekilde o taşın üzerinde gösterilmesi |
| Fail Criterias | Bir önceki hamle sayılarının hareket edilen karenin içinden silinmemesi |
| Test Procedure | Test kullanıcısı; yazılımı, uygun cihazda çalıştırarak  belirtilen girdilerle testi gerçekleştirmelidir. Test işlemini tamamladıktan  sonra test sonucunu Pass/Fail olarak belirterek nedenleri ile birlikte raporlamalıdır. |

**- Hamle hesaplama işleminden sonra tahta üzerinde eski hamle sayısının silinmesi.**

**Test Case 3.**

|  |  |
| --- | --- |
| Risk Level | Yüksek |
| Purpose | Bu test durumunda oyunculardan birinin bütün taşlarının hedef taş üzerine gelince oyunun sonlanıp sonlanmayacağı test edilmektedir. |
| Inputs | Left Click, Taş Butonu, Hedef görseli |
| Expected Outputs | Oyuncu taşlarının hepsi hedefte birleştiğinde oyunun bitmesi. |
| Pass Criterias | Taşlar toplandığında oyunun sonlanması. |
| Fail Criterias | Oyunun sonlanmaması. |
| Test Procedure | Test kullanıcısı; yazılımı, uygun cihazda çalıştırarak  belirtilen girdilerle testi gerçekleştirmelidir. Test işlemini tamamladıktan  sonra test sonucunu Pass/Fail olarak belirterek nedenleri ile birlikte raporlamalıdır. |

**- Oyunculardan herhangi birinin oyunu olması gerektiği gibi bitirebilmesi.**