HASAN FERDİ TURGUTLU TEKNOLOJİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ

YZM 3105 - YAZILIM SINAMA

ÖDEV 2 – Fare Oyunu

142805010 **ABDULLAH ÇALIŞKAN**

142805012 **YASİN İLKER MEHDER**

1. Labirent - Fare Problemi

Labirent içindeki farenin, peynire ulaşmasını sağlamak.
Kurallar:
a. Labirent içinde 1 fare ve 1 peynir olmak zorundadır
b. Labirent içinde, erişilemez alan olmamalıdır.

1.1. Amaç

Labirentimize manuel olarak satır ve sütun sayısı girildikten sonra istediğimiz yerlere duvar yerleştirilerek ve ardından fare ile peynirin konumu belirliyoruz. Belirlendikten sonra “başla” diyerek farenin peynire ulaşmasını sağlamak.

1.2. Girdiler

Satır	Integer	Labirent içerisinde, kaç satır olacak
Sütun	Integer	Labirent içerisinde, kaç sütun olacak
Fare	Bitmap	Labirent içerisinde 1 tane fare olmak zorundadır.
Peynir	Bitmap	Labirent içerisinde 1 tane peynir olmak zorundadır.
Duvar	Bitmap	İstege baglı olarak duvar engeli koyulabilir.

1.3. Mantığı

Kullanıcının Girdiği Satır- Sütun Sayısına göre , farenin hareket edeceği alan belirlenir.
Labirent oluştur butonuna basarak satır,sütun sayısını belirlediğimiz alan gelir Ve bu alanın çevresi otomatik olarak duvarlarla çevirilir .

Bu işlemleri labirent_olustur() fonksiyonu tarafından belirlenir. Satır Ve sütun sayıları hatalı girildiği zaman karşımıza hatalar çıkabilir.

- Satır veya sütun boş olamaz
- Satır ve Sütun bilgileri 0 veya daha küçük olamaz
- Ekran çözünürlüğünden dolayı max 11x15 lik bir alan açılabilir.

Labirentimiz oluştuktan sonra bu duvarlara hiçbir değişiklik yapılamaz. Bu işlemler gridMaze_CellMouseClick() fonksiyonu ile yapılır.
Faremizin Hareketleri Sağlamak için Duvarın değerini -9999- değeri atadık. Normal bir yola 0 değeri verdik. Peynirin değeri ise “-1” dir. Faremizin öncelik değeri peynirdir.

Alanımız oluştuktan sonra;

1- Manuel olarak ,orta taraf boş kalacak şekilde çevresini duvarlar eklersek karşımıza hata çıkacaktır: “Oluşturulan alanlardan birisine erişilemiyor. Alanı düzenle!”.

   Bu işlemleri DuvarKontrol() fonksiyonu ile yapıyoruz.

2- Fare ve Peynir eklendikten sonra Başla Butonuna Basarsak faremiz dolaşmaya başlayacaktır. Fakat fare ve peynir eklenmezse karşımıza hata çıkar .” Labirentte fare ve peynir yok. Fare ve Peynir ekleyin.”

   Bu işlemler btnStart_Click() fonksiyonu ile gerçekleşir.

3- Faremiz, Peynire ulaşırsa karşımıza şöyle bir mesaj çıkacaktır.” Fare, Peyniri buldu. Adım Sayısı :*-*-* ” ve ardından yeni oyuna geçilecektir.

Bu işlemler moveMouse() fonksiyonu ile gerçekleştirilir.

4- Faremizin Hareketlerini sağlamak için ilk önce üste gidiyor. Ve sırasıyla sol , sağ son olarak da alt yönüne geçiyor.
Bu işlemleri gerçekleştirmek için YonBelirle() fonksiyonu kullanılır.

5- Duvar,fare veya peynir seçeneklerden birini seçerek labirentimizi yapmaya başlıyoruz.
Bu işlemleri combo_to_bitmap() fonksiyonu ile yapıyoruz

6- Sol click yaptığımızda duvar,fare veya peynir ekleniyor. Sağ click yaptığımızda ise siliniyor.
Bu işlemleri gridMaze_CellMouseClick() fonksiyonu ile yapıyoruz.

7- Fare ve peynir artarda 2 defa eklenmeyecek
Bu işlemleri farekoyuldu ve peynirkoyuldu bool veri tipi ile belirliyoruz.

Başla Butonuna Basarak Faremiz harekete geçiyor ve ne kadar adım attıysa hedefe ulaşıldığında ekrana yazdırılıyor. Tekrar Oyna Butonuna basarak yeni oyun penceresi ekrana geliyor.

1.4. Program Kodu : .Net 4.5.2 – Microsoft Windows

```
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 yazilim_sinama_odev2
{
   public partial class Form1 : Form
   {
      Bitmap imgYol;
      Bitmap imgDuvar;
      Bitmap imgFare;
      Bitmap imgPeynir;
      Bitmap imgExperienceYol;
      bool fareKoyuldu = false;
      bool peynirKoyuldu = false;
      bool labirentOlusturuldu = false;
      int fareRow, fareCol;
      int satir, sutun;
      int gecilen_hucre = 0;
      public Form1()
      {
         InitializeComponent();
      }

      public void labirent_olustur(int satir, int sutun)
      {
         gridMaze.Rows.Clear();
         gridMaze.Columns.Clear();
         for (int i = 0; i < sutun; i++)
         {
            DataGridViewImageColumn imageColumn = new DataGridViewImageColumn();
            imageColumn.Width = 40;
            gridMaze.Columns.Add(imageColumn);
         }
         for (int i = 0; i < satir; i++)
         {
            gridMaze.Rows.Add();
         }
```

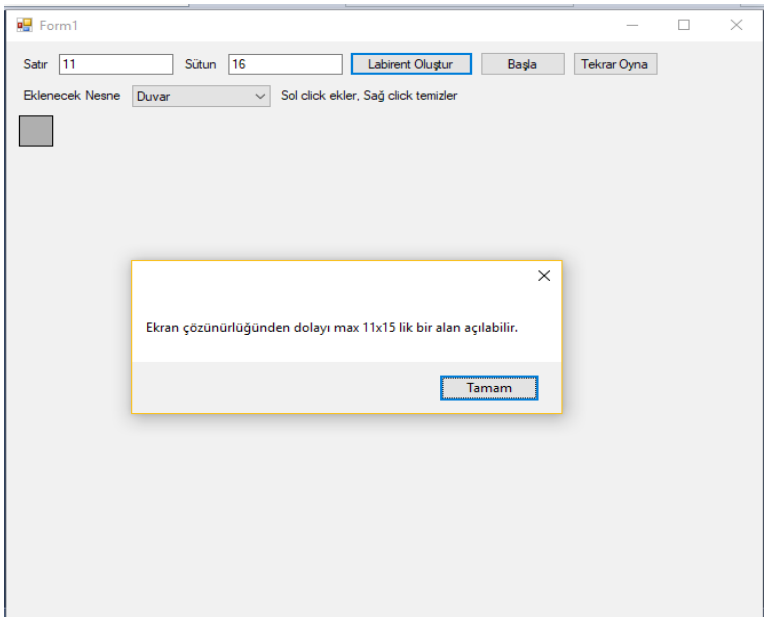
```
            // Hücrelerin arkaplan resimlerini ayarla.
            foreach (DataGridViewRow hucreSatiri in gridMaze.Rows)
            {
                hucreSatiri.Height = 40;
                foreach (DataGridViewCell hucre in hucreSatiri.Cells)
                {
                    hucre.Value = imgYol;
                    hucre.ToolTipText = "0";
                    if (hucre.ColumnIndex == 0 || hucre.ColumnIndex == sutun - 1 || hucre.RowIndex == 0 ||
hucre.RowIndex == satir -1)
                    {
                        hucre.Value = imgDuvar;
                        hucre.ToolTipText = "9999";
                    }
                }
            }
            gridMaze.ColumnHeadersVisible = false;
            gridMaze.RowHeadersVisible = false;
            gridMaze.AutoSize = true;
            labirentOlusturuldu = true;
            fareKoyuldu = false;
            peynirKoyuldu = false;
        }

        private void btnLabirentOlustur_Click(object sender, EventArgs e)
        {
            if(txtSatir.Text == "" || txtSutun.Text == "")
            {
                MessageBox.Show("Satır veya sütun boş olamaz");
                return;
            }
            satir = Convert.ToInt32(txtSatir.Text);
            sutun = Convert.ToInt32(txtSutun.Text);
            if (satir <= 0 || sutun <= 0)
            {
                MessageBox.Show("Satır ve Sütun bilgileri 0 veya daha küçük olamaz");
            }
            if (satir > 11 || sutun > 15)
            {
                MessageBox.Show("Ekran çözünürlüğünden dolayı max 11x15 lik bir alan açılabilir.");
                return;
            }
            labirent_olustur(satir, sutun);
            txtSatir.Enabled = false;
            txtSutun.Enabled = false;

        }
        private void Form1_Load(object sender, EventArgs e)
        {

            imgDuvar = new Bitmap("duvar.jpg");
            imgFare = new Bitmap("fare.jpg");
            imgPeynir = new Bitmap("peynir.jpg");
            imgYol = new Bitmap("yol.png");
            imgExperienceYol = new Bitmap("experience_yol.png");

            labirentOlusturuldu = false;
            fareKoyuldu = false;
            peynirKoyuldu = false;
            comboEklenecek.SelectedIndex = 0;
        }
        public Bitmap combo_to_bitmap()
        {
            if (comboEklenecek.Text == "Duvar")
                return imgDuvar;
            else if (comboEklenecek.Text == "Fare")
            {
                if (fareKoyuldu == true)
                {
```

```
                MessageBox.Show("Labirentte zaten 1 fare var.");
                return null;
            }

            fareKoyuldu = true;
            return imgFare;
        }
        else
        {
            if (peynirKoyuldu == true)
            {
                MessageBox.Show("Labirentte zaten 1 peynir var.");
                return null;
            }
            peynirKoyuldu = true;
            return imgPeynir;
        }
    }
    private void gridMaze_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)
    {
        if (labirentOlusturuldu == true)
        {
            if (e.ColumnIndex == 0 || e.ColumnIndex == sutun - 1 || e.RowIndex == 0 || e.RowIndex ==
satir - 1)
            {
                MessageBox.Show("Bu alan üzerinde düzeltme yapamazsınız.");
                return;
            }
            Bitmap eklenecek_img = null;
            if (gridMaze.Rows[e.RowIndex].Cells[e.ColumnIndex].Value == imgFare)
                fareKoyuldu = false;
            else if (gridMaze.Rows[e.RowIndex].Cells[e.ColumnIndex].Value == imgPeynir)
                peynirKoyuldu = false;
            if (e.Button == MouseButtons.Left)
                eklenecek_img = combo_to_bitmap();
            else
            {
                eklenecek_img = imgYol;

            }
            if (eklenecek_img == null)
                return;
            gridMaze.Rows[e.RowIndex].Cells[e.ColumnIndex].Value = eklenecek_img;
            if (eklenecek_img == imgDuvar)
                gridMaze.Rows[e.RowIndex].Cells[e.ColumnIndex].ToolTipText = "9999";
            else if (eklenecek_img == imgYol)
                gridMaze.Rows[e.RowIndex].Cells[e.ColumnIndex].ToolTipText = "0";
            else if (eklenecek_img == imgPeynir)
                gridMaze.Rows[e.RowIndex].Cells[e.ColumnIndex].ToolTipText = "-1";
            else if (eklenecek_img == imgFare)
            {
                gridMaze.Rows[e.RowIndex].Cells[e.ColumnIndex].ToolTipText = "1";
                fareRow = e.RowIndex;
                fareCol = e.ColumnIndex;
            }
            DuvarKontrol(Convert.ToInt32(txtSatir.Text), Convert.ToInt32(txtSutun.Text));
        }
        else
            return;
    }



  public void DuvarKontrol(int satir, int sutun)
    {
        for (int i = 1; i < sutun; i++)
        {
            for (int j = 1; j < satir; j++)
            {
```

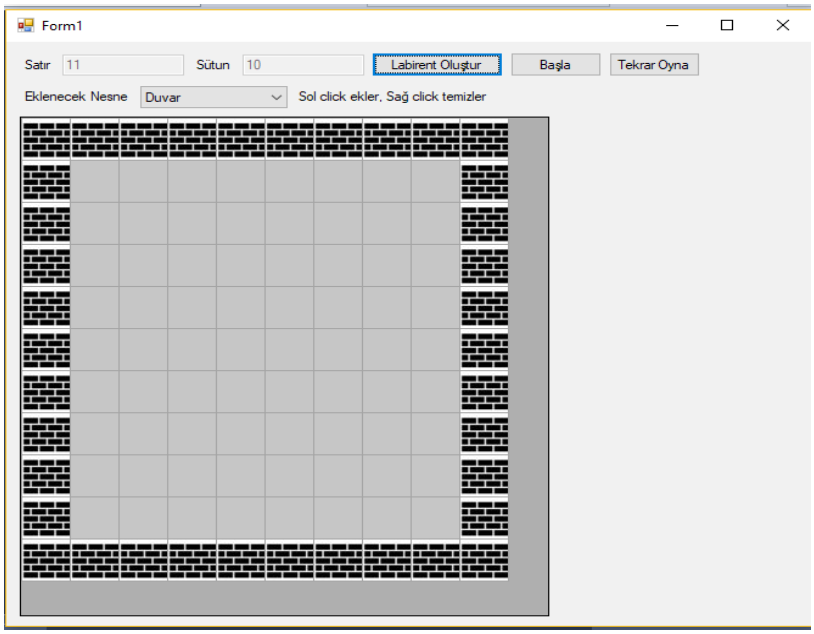
```
                if (gridMaze.Rows[j].Cells[i].ToolTipText == "0")
                {
                    if (gridMaze.Rows[j - 1].Cells[i].ToolTipText == "9999" && gridMaze.Rows[j +
1].Cells[i].ToolTipText == "9999" && gridMaze.Rows[j].Cells[i + 1].ToolTipText == "9999" &&
gridMaze.Rows[j].Cells[i - 1].ToolTipText == "9999")
                        MessageBox.Show("Oluşturulan alanlardan birisine erişilemiyor. Alanı düzenle!");
                }
            }
        }
    }


    private void btnStart_Click(object sender, EventArgs e)
    {
        if (fareKoyuldu == true && peynirKoyuldu == true)
        {
            btnLabirentOlustur.Enabled = false;
            timer1.Enabled = true;
            timer1.Start();

            // dolaşmaya başla
        }
        else
        {
            MessageBox.Show("Labirentte fare ve peynir yok. Fare ve Peynir ekleyin.");
        }
    }

    public void moveMouse(int new_row, int new_col)
    {
        gecilen_hucre++;
        if (gridMaze.Rows[new_row].Cells[new_col].ToolTipText == "-1")
        {
            timer1.Stop();
            timer1.Enabled = false;
            MessageBox.Show("Fare, Peyniri buldu. Adım Sayısı : " + gecilen_hucre.ToString());
            MessageBox.Show("Yeni oyun açılıyor.");
            Application.Restart();

        }
        int tool_tip = Convert.ToInt32(gridMaze.Rows[new_row].Cells[new_col].ToolTipText);
        tool_tip++;
        gridMaze.Rows[new_row].Cells[new_col].ToolTipText = tool_tip.ToString();
        gridMaze.Rows[new_row].Cells[new_col].Value = imgFare;
    }
    public void gecildi_olarak_isaretle()
    {
        gridMaze.Rows[fareRow].Cells[fareCol].Value = imgExperienceYol;
    }
    private void timer1_Tick(object sender, EventArgs e)
    {
        string yon = YonBelirle();
        if (yon == "sag")
        {
            gecildi_olarak_isaretle();
            moveMouse(fareRow, fareCol + 1);
            fareCol++;
        }
        else if (yon == "sol")
        {
            gecildi_olarak_isaretle();
            moveMouse(fareRow, fareCol - 1);
            fareCol--;
        }
        else if (yon == "alt")
        {
            gecildi_olarak_isaretle();
            moveMouse(fareRow + 1, fareCol);
            fareRow++;
```

```
            }
            else if (yon == "ust")
            {
                gecildi_olarak_isaretle();
                moveMouse(fareRow - 1, fareCol);
                fareRow--;
            }
            else
            {
                timer1.Stop();
                timer1.Enabled = false;
                MessageBox.Show("Error. Nereye gidecegimi bilmiyorum");
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Application.Restart();
        }

        public string YonBelirle()
        {

            int ust = 0, sol = 0, sag = 0, alt = 0;
            string yon = "";
            ust = Convert.ToInt32(gridMaze.Rows[fareRow - 1].Cells[fareCol].ToolTipText);
            alt = Convert.ToInt32(gridMaze.Rows[fareRow + 1].Cells[fareCol].ToolTipText);
            sol = Convert.ToInt32(gridMaze.Rows[fareRow].Cells[fareCol - 1].ToolTipText);
            sag = Convert.ToInt32(gridMaze.Rows[fareRow].Cells[fareCol + 1].ToolTipText);

            if (ust <= sol && ust <= sag && ust <= alt)
            {
                if (ust != 9999)
                    yon = "ust";
            }
            else if (sol <= ust && sol <= sag && sol <= alt)
            {
                if (sol != 9999)
                    yon = "sol";
            }
            else if (sag <= ust && sag <= sol && sag <= alt)
            {
                if (sag != 9999)
                    yon = "sag";
            }
            else if (alt <= ust && alt <= sol && alt <= ust)
            {
                if (alt != 9999)
                    yon = "alt";
            }
            return yon;
        }
    }
}
```

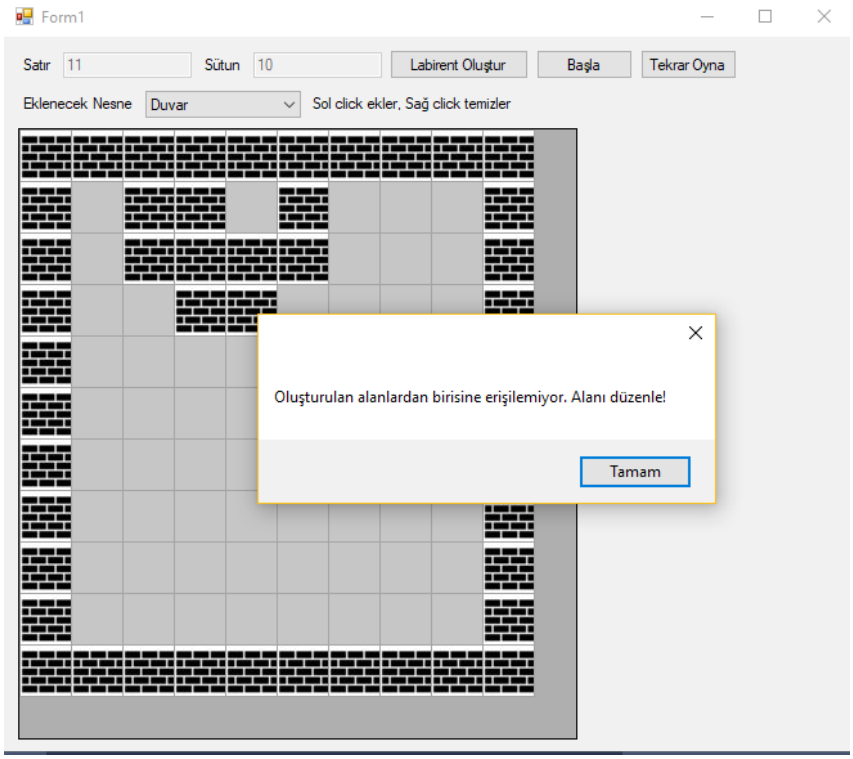
1.5. Ekran Çıktısı



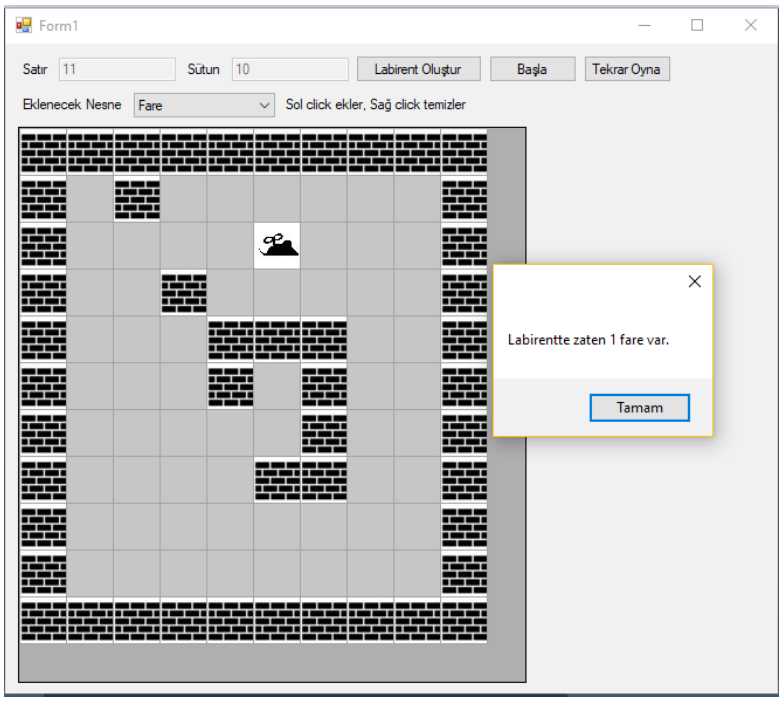
Ekran çözünürlüğünden dolayı oluşturulabilecek max. labirent boyutu.



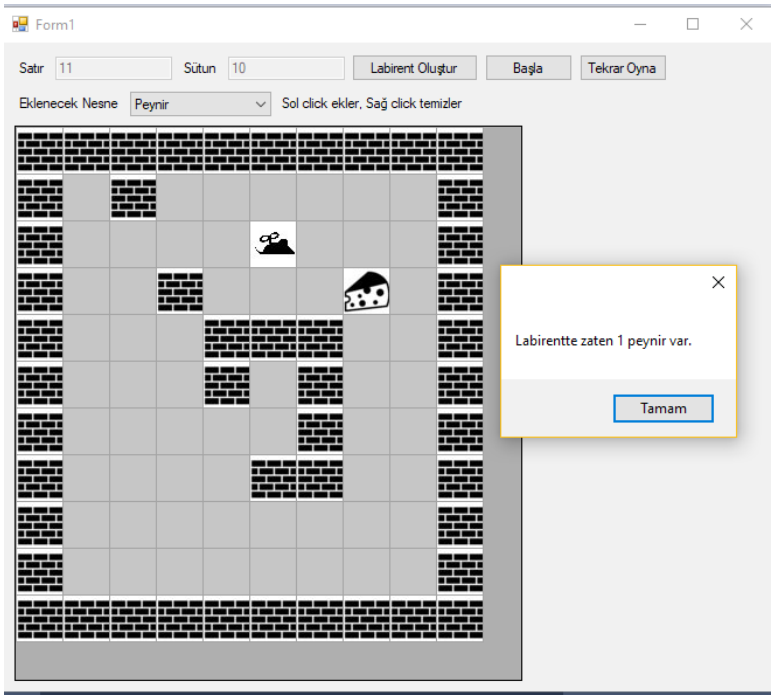
Labirentin oluşturulması.



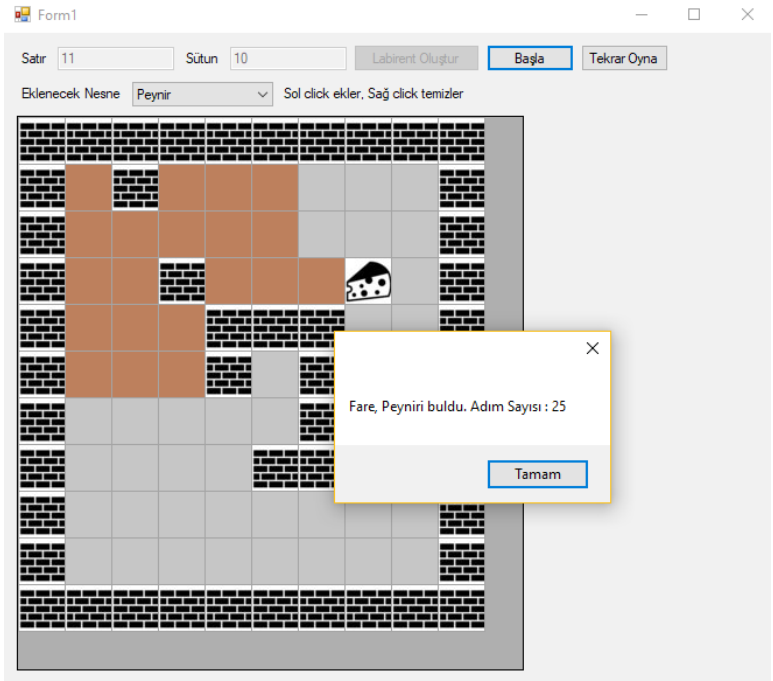
Oluşturulan alana, farenin ulaşamama durumunu kontrol ediyoruz.



Labirentte bir adet farenin oldugunu yenisinin eklenemeyecegini gösteriyoruz.



Labirentte zaten 1 peynirin olduğunu ve yenisinin eklenemeyecegini gösteriyoruz.



Farenin peyniri bulduğunu gösteriyoruz.

1.6. Test Case

a. Programın doğru sonuç ürettiğinin kontrol edilmesi

Risk	Yüksek
Amaç	Farenin, peyniri bulması.
Girdiler	Labirentin oluşturulduğunun bilgisi, fare ve peynirin koyulduğunun bilgisi
Beklenen Çıktılar	Farenin peyniri bulduğunu gösteren MessageBox sonucu
Test Geçiş Kriteri	Beklenen çıktının elde edilmesi.
Başarısızlık Kriteri	Farenin peyniri bulduğunu gösteren pencerenin gelmemesi veya farenin hareket etmemesi
Test Prosedürü	Test kullanıcısı; yazılımı, desteklenen sistem ve gerekliliklerini karşılayarak testi gerçekleştirmelidir. (C# - Visual Studio ve Windows 10)) Test işlemini tamamladıktan sonra test sonucunu, Pass/Fail olarak belirterek nedenleri ile birlikte raporlamalıdır

b. Satır-sütun değerinin beklenen aralıkta girilmemesinin testi

Risk	Yüksek
Amaç	Satır – sütun sayılarını girerken 0 yada 15 sayısının üstünde değer girdisi ile programın exception üretmesinin test edilmesi
Girdiler	0 yada 15 sayısının üstünde ki girdiler
Beklenen Çıktılar	MessageBox.Show("Ekran çözünürlüğünden dolayı max 11x15 lik bir alan açılabilir.");
Test Geçiş Kriteri	MessageBox.Show exceptionın elde edilmesi
Başarısızlık Kriteri	Programın exception üretmeme durumu
Test Prosedürü	Test kullanıcısı; yazılımı, desteklenen sistem ve gerekliliklerini karşılayarak testi gerçekleştirmelidir. (C# - Visual Studio ve Windows 10)) Test işlemini tamamladıktan sonra test sonucunu, Pass/Fail olarak belirterek nedenleri ile birlikte raporlamalıdır

c. Kişinin ulaşılamaz bir alan eklediğinde uyarı almasının testi

Risk	Yüksek
Amaç	Labirent üzerine duvar eklendiğinde, ulaşılamaz bir alan oluştuğunda uygulamanın bunu bildirmesinin test edilmesi.
Girdiler	Duvar eklenmesi.
Beklenen Çıktılar	MessageBox ile oluşturulan alanın erişilemez oldugunun bilgisi.
Test Geçiş Kriteri	Beklenen çıktının oluşması
Başarısızlık Kriteri	Beklenen çıktının oluşmamış olması.
Test Prosedürü	Test kullanıcısı; yazılımı, desteklenen sistem ve gerekliliklerini karşılayarak testi gerçekleştirmelidir. (C# - Visual Studio ve Windows 10)) Test işlemini tamamladıktan sonra test sonucunu, Pass/Fail olarak belirterek nedenleri ile birlikte raporlamalıdır