UZAY MACERASI:
ALTIN AVI

İÇİNDEKİLER

ÖZET

"Uzay Macerası: Altın Avı," her seviyede heyecan verici bir oyun deneyimi sunan, Mario tarzı mekaniği ile uzay atmosferinde geçen bu serüven, kullanıcının becerilerini sınayarak eğlenceli bir deneyim vadediyor. Her seviyesi, artan zorluk seviyeleri ve çeşitli görevlerle dolu; neredeyse oyuncuyu bir uzay serüvenine davet edercesine tasarlanmıştır. Uzayın derinliklerinde geçen bu oyun, renkli grafikleri ve akıcı kontrol mekanikleriyle oyuncuları etkileyici bir atmosfer içinde buluşturur. "Uzay Macerası: Altın Avı," sadece eğlenceli bir oyun olmanın ötesinde, aynı zamanda oyunculara strateji ve hızlı düşünme yeteneklerini geliştirme fırsatı sunan bir platform macerasıdır.

Oyunun Temel Özellikleri:

Beş Bölüm: Oyun, beş farklı bölümden oluşmaktadır. Her bölüm, öncekine göre daha zorlu ve heyecan verici bir platform deneyimi sunar.

Altın Toplama: Kullanıcı, bölümlerde yer alan altınları toplayarak puan kazanır. Ancak, bir bölümü tamamlamak için bütün altınları toplamak zorundadır.

Anahtar ve Kapı: Her bölümün sonunda bir anahtar bulunur. Bu anahtarı almadan, bölümün sonundaki kapıyı açmak mümkün değildir. Kullanıcı, bu stratejik unsuru kullanarak bölümleri tamamlamak zorundadır.

Işın Kılıcı: Bu ışın kılıcına temas ettiğinde, kullanıcıları öldürebilir.

Kullanıcı Giriş: Kullanıcı adı ile kayıt olur ve oyuna başlar.

Süre: Bu oyunda, geçen süre sürekli olarak sayılır ve tamamladığınız her bölüm için veri tabanına kaydedilir; en hızlı sürede en çok altını toplamak için yarışın!

Score: Bu oyunda, topladığınız altın miktarı ve geçen süre ekranda gösterilir ve veri tabanına kaydedilir; en yüksek skoru elde etmek için mümkün olduğunca hızlı ve çok altın toplayın!

Zorluk Seviyeleri: Oyun, her seviyede artan bir zorluk seviyesi sunarak kullanıcının becerilerini geliştirmesine olanak tanır.

Eğlence: Renkli grafikler, eğlenceli müzikler ve sürükleyici oyun mekaniği ile kullanıcıya eğlenceli bir deneyim sunar.

Stratejik Düşünme: Altınları ve anahtarı toplarken, kullanıcı stratejik düşünmeye teşvik edilir.

PROJENİN AMACI

Oyunun temel amacı, oyunculara farklı bölümlerden oluşan bir oyun keyfi sunmaktır. Oyuncular, her bölümde platformlar arasında zıplayarak altınları toplamalı ve en sonunda anahtarı bulup, başladıkları yerdeki kapıya ulaşarak bölümü tamamlamalıdır. Ancak unutmayın, süre ve skor da önemlidir! Hızlı hareket eden ve daha fazla altın toplayan oyuncular, yüksek skorlar elde eder ve liderlik tablosunda üst sıralara çıkar. Oyunu tamamlamak için istenilen bütün koşulları sağlayarak, en hızlı ve en yüksek skorlu oyuncu olmak için mücadele edilir.

Oyun içeriği aşağıdaki gibidir.

Altınlar ve Anahtar = Oyuncu seviyedeki bütün altınları toplamadan ve anahtarı almadan bölümü bitiremez.

Karakter= Kullanıcı karakteri kontrol eder.

Platformlar= Kullanıcı platformların üzerinden zıplayarak altın ve anahtarı toplar.

Engeller= Kullanıcı bu engellere değdiği zaman oyunu durdurur.

5 Seviye= Oyunda git gide zorlaşan 5 farklı seviye vardır. Oyundaki genel amaç bu 5 seviyeyi başarılı bir şekilde tamamlayarak oyunu bitirmektir.

Kullanıcı Giriş: Yazılan kullanıcıyı kaydeder ve her girdiğinde o isimle giriş yaparsan kedi score puanını aştığın zaman o kullanıcı güncellenir.

Süre ve Score: Oynadığınız süre otomatik olarak sayılır ve topladığınız altınlarla birlikte skor tablosuna kaydedilir. Eğer ilk 5'e girdiyseniz, liderlik tablosunda adınızı görebilirsiniz.

GÖREV DAĞILIMI

Her ne kadar görev dağılımı yapsak da, her aşamada birbirimize yardım ederek ilerledik. Bazı bölümlerde bazılarımızın daha fazla görev aldığı durumlar oldu, ancak kodlamanın her aşamasını birlikte gerçekleştirdik. Muhammed Rıdvan Beyiş tasarım ve kullanıcı girişi, Nisa Usta level design'da, Şimal Bülbül ise genel işleyiş alanında daha ağırlıklı çalıştı. Veri tabanı içeriği Şimal ve Nisa tarafından oluşturuldu. Skor ve süre sistemi ise hep birlikte geliştirildi. Ancak, dediğimiz gibi, kod aşamasında her bir satırda birlikteydik; araştırma yaparken birbirimize destek olduk ve yazma sürecinde birlikte hareket ettik.

PROJE NASIL BİR PROBLEMİ ÇÖZDÜ

Projemiz, karmaşık olmayan temel bir oyun içeriğine sahip olduğu için zaman geçirmek için idealdir. Hem küçük çocuklar hem de vakit geçirmek isteyen yetişkinler için uygundur. Sıkılan veya eğlenmek isteyen herkesin keyifli vakit geçirmesini sağlar. Ayrıca, liderlik tablosunda isminizi görmek için daha fazla rekabet etme isteğinizi tetikleyerek, oyunun tadını çıkarmak için bir motivasyon kaynağı sağlar.

PROJEMİZDE TAMAMLADIĞIMIZ HEDEFLER

Projeye başladığımızda kafamızdaki planladığımız oyunla, şu an geliştirdiğimiz oyun hemen hemen aynı özelliklere sahip. Eksik kalan veya yapamadığımız bir şey olmadı. Kafamızdaki oyunu tam olarak çalışarak, emek vererek ve paylaşım yaparak başarıyla tamamladık. İlk hedefimiz karakterin hareketlerini ve arkaplanı kontrol etmekti ve bunu başardık. Sonrasında altın toplama ve anahtar ile ilgili fonksiyonları tamamlayarak projeyi geliştirdik. Her seviyede ayrı zorluk ekleyerek hayalimizdeki projeye bir adım daha yaklaştık ve sonuç olarak tam aklımızdaki gibi bir proje ortaya çıkardık.

ÖNCEKİ DÖNEMİN DEVAM PROJESİ

Bu projemiz önceki dönem projemizin devamıdır. Ek olarak veri tabanı üzerinden yapılan işlemlerle ilgili güncellemeler yaptık. Artık her yapılan işlem veri tabanına ekleniyor ve kullanıcı girişi ile kullanıcı isimlerini kaydettik. Oyun başladıktan sonra süre ve skorun kaydedilmesini sağladık. Bu sayede kullanıcılar artık oyunda kaydedilen ilerlemeleri ve skorlarını takip edebilirler.

EKSİK KALAN HEDEFLER

"Nasıl Oynanır" adında bir sekme eklemeyi planladık. Bu sekmede, oyunu öğreten bir video ve görsel içerikli dokümanlar bulunacaktı. Ancak, video ve görselleri veritabanına yükleme işlemini istediğimiz gibi gerçekleştiremedik ve proje dosyasına eklemek zorunda kaldık. Bu da projeye fazladan yük getirdi. Bu nedenle, projenin performansını etkilememek için bu işlevi çıkarmaya karar verdik. Gelecekte, video ve görselleri optimize ederek bu işlevi yeniden eklemeyi düşünebiliriz.

PROJENİN GELECEĞİ

Şu anlık bu projeyi aklımızdaki gibi tamamlamış olsak da, geliştirmeye devam etmek hepimizin içinden geçen bir düşünce. İleride eklemeyi planladığımız şeyler arasında kullanıcıya karakter seçimi, nasıl oynanır izleme gibi özellikler bulunuyor. Bu eklemelerle oyun deneyimini daha zengin ve kişiselleştirilebilir hale getirmeyi hedefliyoruz.

PROJENİN YAPIM AŞAMALARI

Araştırma Süreci

Araştırma sürecinde daha önce ne Windows Forms ne de oyun geliştirme alanında çalışmadığımız için, kendi projemize başlamadan önce bir alıştırma projesi yapmak istedik ve bu amaçla Flappy Bird oyununu seçtik. Hem yabancı hem Türk kaynaklardan bu konuyla ilgili çeşitli araştırmalar yaparak, videolar izleyerek kodlamanın mantığını öğrendik. Ardından, yaparken öğrendiklerimizin üzerine ekleyerek öğrenmeye devam edip kendi projemizi geliştirmeye başladık.

Planlama ve Tasarım Süreci

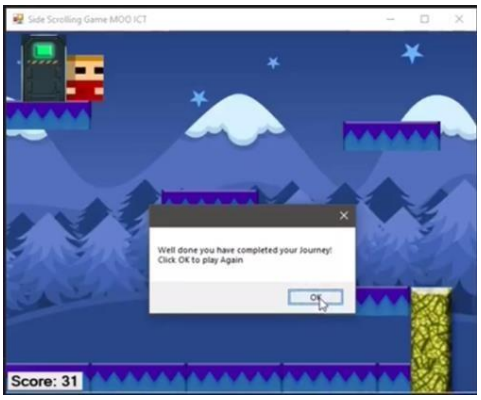
Planlama ve tasarım sürecinde ilk olarak, oyunumuzu nasıl bir konseptle geliştireceğimizi detaylıca tartıştık. Kararımız, oyunun uzay temasında olması yönündeydi. Bu temaya uygun olarak arkaplan, karakter, platform, altın ve kapı tasarımlarını geliştirmeye başladık. Her bir öğe için dikkatlice düşünülmüş ve uyumlu tasarımlar oluşturarak oyunun görsel kimliğini belirledik.

Tasarım sürecinin ardından, kodlamaya geçerek oyunun temel mekaniklerini oluşturduk. Karakterin hareketi, altınları toplama, kapıyı açma gibi temel oyun özelliklerini programladık. Daha sonra veri tabanı araştırarak süre, skor ve kullanıcı girişi gibi özellikleri ekledik. Bu süreçte, daha önce belirlediğimiz uzay temasını oyun mekaniklerine başarılı bir şekilde entegre ettik.

PROJE ORJİNAL MİDİR?

Projemiz, konsept olarak daha önce benzerleri yapılmış bir projeye dayanmaktadır. Ancak, biz bu yapıya yeni özellikler ve bakış açıları ekleyerek projeyi ileriye taşıdık. Örnek aldığımız projenin linki buradadır:(https://www.youtube.com/watch?v=yw_sn9DonH4&t=1159s&ab_channel=MooICT).

Bu projeyi temel alarak, tasarımsal ve mantıksal birçok değişiklik yaptık ve orijinalinden oldukça farklılaştırdık. Yaptığımız geliştirmeler arasında arayüz kullanımı, veritabanı entegrasyonu, bölüm tasarımı, karakter ve çevre tasarımı, oyun mekanikleri (süre sayacı, skor sınırlamaları) gibi alanlarda önemli ilerlemeler bulunuyor. Bu sayede, bahsi geçen projeyi daha ileri seviyelere taşıdık.

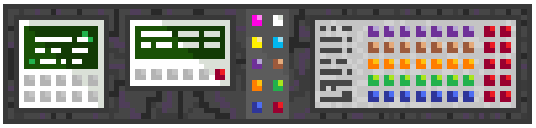


NEDEN TERCİH EDİLMELİDİR?

Tamamen kendi tasarımımız olan bir yapı oluşturmak istedik. Oyunun her tarafında elle yaptığımız çizimler bulunmaktadır. Türdaşlarına göre daha gelişmiş bir arayüz tasarımı olması, oyunumuzu tercih edilebilir kılan öncelikli sebeplerden biridir. Veritabanı bağlantılarıyla oluşturduğumuz sıralama mantığı, rekabetçi duyguları uyandırarak oyuncular arasında sosyal etkinlikler yaratabilme imkanı sunmaktadır. Bu özellik, oyunumuzu diğerlerinden ayıran ve tercih edilebilirliğini artıran önemli bir faktördür.

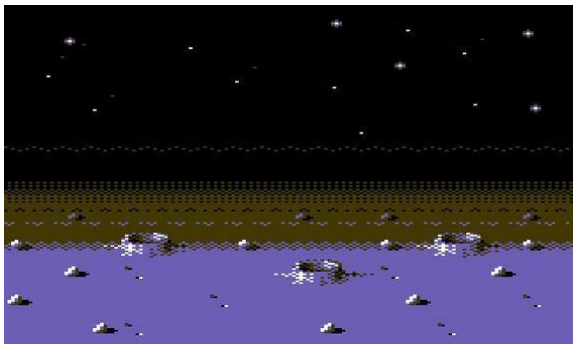
PLATFORMLAR

Şekil 2 : Platform1



Şekil 1 :Platform2

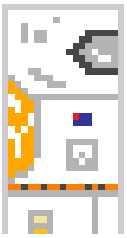
ARKAPLANLAR



Şekil 3 :Arka Plan2

Şekil 4 :Arka Plan1

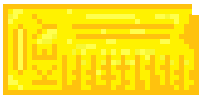
KARAKTER



Şekil 5 :Karakter

ENGELLER, COİN ve ANAHTAR

Şekil 6 : Engel



Şekil 7 :Anahtar

Şekil 8 : Coin

KULLANICI ANASAYFA

KULLANICI ANASAYFA

Kullanıcı adı girişi: Kullanıcıya bir kullanıcı adı girmesi için bir metin kutusu sunulur. Bu ad, veritabanına kaydedilir.

Oyun başlatma: Kullanıcı adı girişi yapıldıktan sonra, oyunun başladığını belirten bir mesaj görüntülenir ve oyun ekranı açılır.

Oyun yapımcılarına erişim: Kullanıcılar, oyunun yapımcıları hakkında bilgi almak için bu seçeneği kullanabilirler. Bu seçeneğe tıklandığında, yapımcıların bilgilerini içeren bir form açılır.

Skor tablosuna erişim: Kullanıcılar, oyunun skor tablosunu görmek için bu seçeneği kullanabilirler. Bu seçeneğe tıklandığında, skor tablosunu gösteren bir form açılır.

Uygulamayı kapatma: Kullanıcılar, uygulamayı kapatmak için bu seçeneği kullanabilirler. Bu seçeneğe tıklandığında, uygulama sonlandırılır.

Kodun genel amacı, bir oyun uygulamasının ana sayfasını oluşturmak ve kullanıcıların bu sayfa üzerinden oyunu başlatmalarını, yapımcı bilgilerini görüntülemelerini, skor tablosunu incelemelerini ve uygulamayı kapatmalarını sağlamaktır. Bu işlevler, kullanıcı dostu bir arayüz sağlayarak oyun deneyimini geliştirmeyi amaçlar.

	OyuncuID	Ad
1	1	rid
2	2	Rıdvan
3	1002	sadgfas
4	1003	nisa

Buda kullanıcı girişi yapıldıktan sonra oyuncuları kaydedildiğini gösteren veri tabanı tablosu.

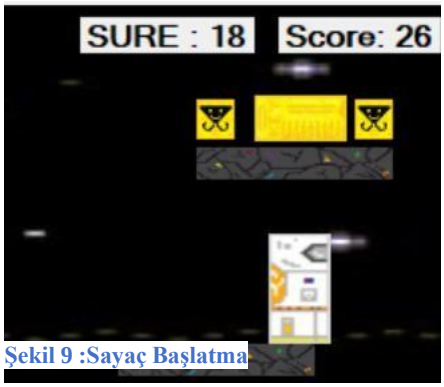

```
private void button3_Click(object sender, EventArgs e)
{if (string.IsNullOrEmpty(veri.name))
   { MessageBox.Show("Lütfen bir kullanıcı adı girin.");
return; // Metodu burada sonlandırın, çünkü kullanıcı adı girilmemiş.}
string conString = "Data Source=DESKTOP-65RMS34;Initial
Catalog=gameProject222;Integrated Security=True";
   SqlConnection connect = new SqlConnection(conString);
   Try { connect.Open();
SqlCommand cmd = new SqlCommand($"INSERT INTO Oyuncular(Ad) SELECT
'{veri.name}' WHERE NOT EXISTS (SELECT 1 FROM Oyuncular WHERE Ad =
'{veri.name}');", connect);,cmd.Parameters.AddWithValue("@ad",
veri.name);cmd.ExecuteNonQuery();
       // Kullanıcı adını labela atayalım
       label1.Text = "" + veri.name;
       adgetir.ad= veri.name; }
   catch (Exception ex) { MessageBox.Show("Hata: " + ex.Message); }
   finally{connect.Close();  }
   Form1 f = new Form1();
   f.isimYaz();
   f.Show();
   this.Hide();}
```

Kullanıcının girdiği adı veritabanına ekler ve ana oyuna geçiş yapar. Öncelikle, kullanıcının bir ad girip girmediğini kontrol eder; eğer ad girilmemişse, bir uyarı mesajı gösterir ve işlem sonlanır. Kullanıcı adı girilmişse, veritabanına bağlanarak bu adı Oyuncular tablosuna ekler. Eğer bu ad zaten veritabanında mevcutsa, yeniden ekleme yapılmaz. Kullanıcı adı başarılı bir şekilde eklenirse, bu ad bir etikete (label) ve statik bir değişkene atanır. Herhangi bir hata oluşursa, bir hata mesajı gösterilir. İşlemin sonunda, veritabanı bağlantısı kapatılır ve yeni bir form (Form1) açılarak kullanıcının adı bu formda gösterilir.

OYUN İÇERİK KODLARI

Oyun içerisinde kullanılacak değişkenleri ve değerlerini tanımlıyoruz. Her bir değişkenin ne amaçla kullanılacağı, programın mantığına ve ihtiyaçlarına bağlı olarak belirlenir. Bu değişkenler, karakterin hareketleri, zıplama hızı, skor, karakter hızı, arkaplan hızı gibi oyunun çeşitli özelliklerini kontrol etmek için kullanılabilir.

```
bool solaGit, sagaGit, zipla, anahtarial;
int zıplamaHizi = 10;
int force = 8;
int score = 0;
int karakterHizi = 10;
int arkaplanHizi = 8;
```

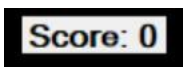


Şekil 9 :Sayaç Başlatma

Form yüklendiğinde kullanıcıya oyun kurallarını gösteren ve sayaç başlatan bir başlangıç işlemini gerçekleştirir.

```
private void Form1_Load(object sender, EventArgs e)
{MessageBox.Show("1 - Anahtarı almadan kapıyı
açamazsın.\n\r2 - Bütün coinleri toplamadan kapı
açıkmaz.","Kurallar");
    label4.Text = veri.name ;
    Sayaç.Start();
}
```

SCORE GÜNCELLEME



Şekil 10 :Score Puanı

```
txtScore.Text = "Score: " +
score;
```

Her döngüde güncel skoru ekranda gösterir.

KARAKTERİN YÜKSEKLİK AYARI

```
karakter.Top +=
zıplamaHizi;
```

Karakterin yukarı veya aşağı hareket etmesini sağlar. zıplamaHizi karakterin dikey hızını belirler.

KARAKTERİN YATAY HAREKETİ

```
if (solaGit == true && karakter.Left > 60)
{ karakter.Left -= karakterHizi; }
if (sagaGit == true && karakter.Left + (karakter.Width + 60) < this.ClientSize.Width)
{ karakter.Left += karakterHizi; }
```

Kullanıcı sola gitmek istediğinde (solaGit == true), karakterin sola hareket etmesini sağlar. Eğer karakter pencerenin sol kenarından çok uzak değilse (karakter.Left > 60). Sağ hareket (sagaGit == true) için benzer şekilde karakterin sağa hareket etmesini sağlar, ancak pencerenin sağ kenarına çok yaklaşmamasını kontrol eder (karakter.Left + (karakter.Width + 60) < this.ClientSize.Width).

ARKAPLAN VE OYUN ELEMANLARININ HAREKETİ

```
if (solaGit == true && arkaplan.Left < 0)
{ arkaplan.Left += arkaplanHizi;
    oyunHareketElements("forward");  }
if (sagaGit == true && arkaplan.Left > -935)
{ arkaplan.Left -= arkaplanHizi;
    oyunHareketElements("back");  }
```

Karakter sola hareket ederken arkaplanın sağa kaymasını (arkaplan.Left < 0) ve diğer oyun elemanlarının ileri hareket etmesini sağlar. Karakter sağa hareket ederken arkaplanın sola kaymasını (arkaplan.Left > -935) ve oyun elemanlarının geri hareket etmesini sağlar. oyunHareketElements metodu, oyun elemanlarının hareket yönünü kontrol eder.

ZIPLAMA KONTROLÜ

```
if (zipla == true)
{ zıplamaHizi = -12;
   force -= 1; }
else  { zıplamaHizi = 12; }
if (zipla == true && force < 0)
{ zipla = false; }
```

Karakterin zıplama hareketini ve yer çekimi etkisini kontrol eder. zipla true olduğunda, karakter yukarı doğru zıplar (zıplamaHizi = -12), zıplama gücü (force) azalır. zipla false olduğunda, karakter aşağı düşer (zıplamaHizi = 12). Zıplama gücü sıfırın altına düştüğünde, zıplama durur (zipla = false).

KARAKTERİN DÜŞMESİ

Şekil 11: Bölüm Sonu

```
if (karakter.Top + karakter.Height > this.ClientSize.Height)
{ GameTimer.Stop(); BitisEkraniForm f = new BitisEkraniForm(); f.Message("Bolum 1
İyi Denemeydi"); f.Time(gecenSure.ToString()); f.Score(score); f.Show(); this.Hide(); }
```

Karakter ekranın altına düşerse oyun durur ve bitiş ekranı gösterilir.

NESNELERLE ETKİLEŞİM

```
foreach (Control x in this.Controls)
{ // Platformlarla Etkileşim
    if (x is PictureBox && (string)x.Tag == "platform")
    { if (karakter.Bounds.IntersectsWith(x.Bounds) && zipla == false)
        { force = 8; karakter.Top = x.Top - karakter.Height; zıplamaHizi = 0;  } x.BringToFront();
}
    // Kılıçla Etkileşim
    if (x is PictureBox && (string)x.Tag == "kilic")
    { if (karakter.Bounds.IntersectsWith(x.Bounds))
{ GameTimer.Stop(); BitisEkraniForm f = new BitisEkraniForm(); f.Message("Bolum 1 İyi
Denemeydi"); f.Time(gecenSure.ToString()); f.Score(score); f.Show(); this.Hide(); }
x.BringToFront(); }
    // Altınlarla Etkileşim
    if (x is PictureBox && (string)x.Tag == "altın")
    {if (karakter.Bounds.IntersectsWith(x.Bounds) && x.Visible == true)
        {x.Visible = false;  score += 1; } }  }
```

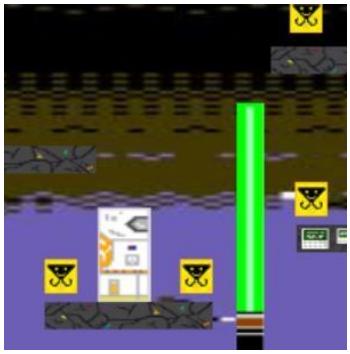
Bu bölümde, karakterin platformlar, kılıçlar ve altınlarla olan etkileşimleri kontrol edilir.

Platformlarla Etkileşim: Karakter bir platforma çarptığında, zıplama gücü sıfırlanır ve karakter platformun üstünde durur.

Kılıçla Etkileşim: Karakter kılıca çarptığında oyun durur ve bitiş ekranı gösterilir.

Altınlarla Etkileşim: Karakter altına çarptığında altın görünmez hale gelir ve skor artırılır.

ANAHTAR VE KAPIYLA ETKİLEŞİM

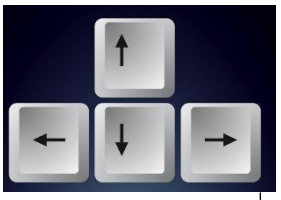


Şekil 12 :Nesne Etkileşim

```
if (karakter.Bounds.IntersectsWith(anahtar.Bounds))

{ anahtar.Visible = false; anahtarial = true; }

if (karakter.Bounds.IntersectsWith(kapı.Bounds) && anahtarial == true && score >= 30)

{  kapı.Image = GameProjectSon.Properties.Resources.door_open; GameTimer.Stop();

   BitisEkraniForm f = new BitisEkraniForm(); f.Message("1. Bölüm Bitti"); string
oyuncuadı = "";  oyuncuadı = AnaSayfaForm.adgetir.ad; Time.Second = gecenSure;

   string conString = "Data Source=DESKTOP-
65RMS34;InitialCatalog=gameProject222;Integrated Security=True";

   SqlConnection connect = new SqlConnection(conString); connect.Open();

   SqlCommand cmd2 = new SqlCommand($"select OyuncuID from Oyuncular where Ad =
'{oyuncuadı}'", connect);

   int id = Convert.ToInt32(cmd2.ExecuteScalar());

   SqlCommand cmd = new SqlCommand($"IF EXISTS (SELECT 1 FROM Score WHERE
OyuncuID = {id})\r\nBEGIN\r\n    UPDATE Score\r\n    SET ScorePuani = ${score}\r\n
WHERE OyuncuID = ${id};\r\nEND\r\nELSE\r\nBEGIN\r\n    INSERT INTO Score
(OyuncuID, ScorePuani)\r\n    VALUES (${id}, ${score});\r\nEND", connect);
cmd.ExecuteNonQuery();

   SqlCommand cmd3 = new SqlCommand($"IF EXISTS (SELECT 1 FROM Sure WHERE
OyuncuID = {id})\r\nBEGIN\r\n    UPDATE Sure\r\n    SET Sure = ${Time.Second}\r\n
WHERE OyuncuID = ${id};\r\nEND\r\nELSE\r\nBEGIN\r\n    INSERT INTO Sure
(OyuncuID, Sure)\r\n    VALUES (${id}, ${Time.Second});\r\nEND", connect);

   cmd3.ExecuteNonQuery();

   f.Time(GameProjectSon.Time.Second.ToString());  f.Score(score); f.Show();  this.Hide();}
```

Karakter anahtara çarptığında anahtar görünmez hale gelir ve anahtarial true olur. Karakter kapıya çarptığında ve gerekli koşullar sağlandığında (anahtar alınmış ve skor 30 veya daha fazla), kapı açılır ve oyun durdurulur. Sonrasında bitiş ekranı gösterilir ve oyuncu bilgileri veritabanına kaydedilir veya güncellenir.

```
private void KeyIsDown(object sender, KeyEventArgs e) //tuşa basıldığında
{
    if (e.KeyCode == Keys.Left || e.KeyCode == Keys.A) // sol için
    {
        solaGit = true;
    }
    if (e.KeyCode == Keys.Right || e.KeyCode == Keys.D) // sağ için
    {
        sagaGit = true;
    }
    if (e.KeyCode == Keys.Up && zipla == false || e.KeyCode == Keys.W && zipla == false
|| e.KeyCode == Keys.Space && zipla == false) // zıplama için
    {
        zipla = true;
    }
}
```



```
private void KeyIsUp(object sender, KeyEventArgs e) //tuşa
basılmadığında
{
    if (e.KeyCode == Keys.Left || e.KeyCode == Keys.A)
    { solaGit = false; }
    if (e.KeyCode == Keys.Right || e.KeyCode == Keys.D)
    { sagaGit = false; }
    if (zipla == true) // zıplama için
    { zipla = false;}
}
```

Tuş atamalarını yapmak için tanımladığımız “KeyUp” ve “KeyDown” nesnelerini çağırarak özellikler ekliyoruz. Bu kod, bir oyun veya uygulama içinde kullanıcının klavye tuşlarına tepki veren bir olay işleyicisi içerir. Kod, iki farklı olay işleyicisi içerir: KeyIsDown ve KeyIsUp. İlk fonksiyon, bir tuşa basıldığında çalışırken, ikinci fonksiyon bir tuş serbest bırakıldığında çalışır.

OYUN DURDURMA KODU

Bu kod parçası, bir Windows Forms uygulamasında oyunun baştan başlatılması için kullanılacak bir fonksiyonun temel yapısal tanımını içerir. Ancak, fonksiyon şu anda geçici olarak boş bırakılmıştır ve ilerleyen aşamalarda içeriği doldurulacaktır.

Şekil 13: Oyundan

```
private void OyunuDurdur(object sender, FormClosedEventArgs e)
{
 Application.Exit();
}
```

HAREKET KODU

Şekil 14:Hareket Kodu

oyunHareketElements' bölümünün içerisini doldurmaya geçiyoruz. Bu kod, oyun içerisinde bulunan belirli türdeki PictureBox nesnelerini (platform, altın, anahtar, kapı, kılıç gibi) belirtilen yönde (ileri veya geri) hareket ettirmekten sorumludur.

```
private void oyunHareketElements(string direction)
{  foreach (Control x in this.Controls)
    {  if (x is PictureBox && (string)x.Tag == "platform" || x is PictureBox && (string)x.Tag
== "altın" || x is PictureBox && (string)x.Tag == "anahtar" || x is PictureBox &&
(string)x.Tag == "kapı" || x is PictureBox && (string)x.Tag == "kilic")
        {       if (direction == "back")
           { x.Left -= arkaplanHizi;  }
           if (direction == "forward")
           {  x.Left += arkaplanHizi;  }} }  }
```

SAYFA YÖNLENDİRME

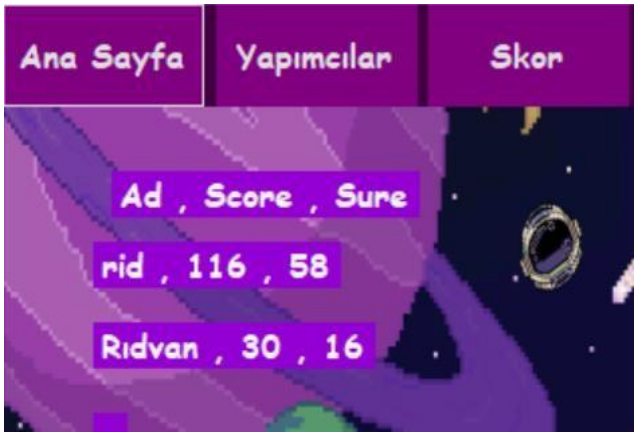
Level seçme bölümünde, her bir butona tıklandığında tasarladığımız formların yönlendirmelerini ekledik.

Bu kod, belirli butonlara tıklandığında belirli formları açma ve şu anki formu gizleme işlevini yerine getirir. Bu tip bir kod, genellikle kullanıcı arayüzünde farklı seviyelere veya bölümlere geçiş yapmak için kullanılır.

```
private void button3_Click(object sender,
EventArgs e)
{ Form1 f = new Form1();
   f.Show();
   this.Hide();  }

private void button5_Click(object sender,
EventArgs e)
{ Form2 f = new Form2();
    f.Show();
    this.Hide();  }
```

```
private void button6_Click(object sender,
EventArgs e)
{ Form3 f = new Form3();
 f.Show();
   this.Hide();  }

private void button4_Click(object sender,
EventArgs e)
{ Form4 f = new Form4();
   f.Show();
   this.Hide();  }

private void button7_Click(object sender,
EventArgs e)
{ Form5 f = new Form5();
   f.Show();
   this.Hide();  }
```

SCORE VERİ TABANINA KAYDETME



Şekil 15: Score Vt

```
private void lblScore_Click(object sender, EventArgs e)
 {string conString = "Data Source=DESKTOP-65RMS34;Initial
Catalog=gameProject222;Integrated Security=True";
    SqlConnection connect = new SqlConnection(conString);
    try
    {connect.Open();
SqlCommand cmd = new SqlCommand("INSERT INTO BolumDurumu (Skor)
VALUES (@skor)", connect);
        cmd.Parameters.AddWithValue("@skor", lblScore.Text);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Score başarıyla kaydedildi!");
        // Kullanıcı adını labela atayalım
        label2.Text = "Skor:  " + lblScore.Text;}
    catch (Exception ex)
    {MessageBox.Show("Hata: " + ex.Message);}
    finally
    {connect.Close();     }}
```

Bu kod, bir kullanıcı tıkladığında skoru bir veritabanına kaydeder. lblScore_Click metodu, bir etiketin (label) tıklanma olayını işler. İlk olarak, SQL Server veritabanına bağlantı kurar ve sonra BolumDurumu tablosuna, lblScore etiketinin metin değerini ekler. Başarılı bir ekleme durumunda, kullanıcıya bir mesaj gösterilir ve başka bir etiket (label2) güncellenir. Hata oluşursa, hata mesajı kullanıcıya gösterilir ve sonunda bağlantı kapatılır.

ScoreID	OyuncuID	ScorePu...
1	1	116
2	2	30
1002	1003	30
NULL	NULL	

Şekil 16:Score VT

```
private void scoreTablosu_Load_1(object sender, EventArgs e)
 {string conString = "Data Source=DESKTOP-65RMS34;Initial
Catalog=gameProject222;Integrated Security=True";
    SqlConnection connect = new SqlConnection(conString);
    connect.Open();
    List<string> list = new List<string>();
    SqlCommand cmd = new SqlCommand($"SELECT TOP 8 Oyuncular.Ad,
Score.ScorePuani, Sure.Sure\r\nFROM Oyuncular\r\nJOIN Score ON Oyuncular.OyuncuID =
Score.OyuncuID\r\nJOIN Sure ON Oyuncular.OyuncuID = Sure.OyuncuID\r\nORDER BY
Score.ScorePuani DESC, Sure.Sure ASC;", connect);
    DataTable dt = new DataTable();
    using (SqlDataAdapter adtr = new SqlDataAdapter(cmd))
    {adtr.Fill(dt);}
    if (dt.Rows.Count == 1)
    {label4.Text = $"{dt.Rows[0][0].ToString()} , {dt.Rows[0][1].ToString()} ,
{dt.Rows[0][2].ToString()}";}
    if (dt.Rows.Count == 2)
    {label4.Text = $"{dt.Rows[0][0].ToString()} , {dt.Rows[0][1].ToString()} ,
{dt.Rows[0][2].ToString()}";
label5.Text = $"{dt.Rows[1][0].ToString()} , {dt.Rows[1][1].ToString()} ,
{dt.Rows[1][2].ToString()}"; }……
```

Bu kod, `scoreTablosu` formu yüklendiğinde veritabanına bağlanarak en yüksek puana sahip ilk 8 oyuncunun adını, puanını ve süresini çekip ekranda gösterir. Veriler, SQL sorgusu ile `DataTable` nesnesine doldurulur ve ardından farklı etiketlere (`label4`, `label5`, vb.) atanarak kullanıcılara sunulur. Bu işlem, kullanıcıların oyun skorlarını görmesini sağlar.

YAPIMCILAR VERİ TABANINDAN ÇEKME

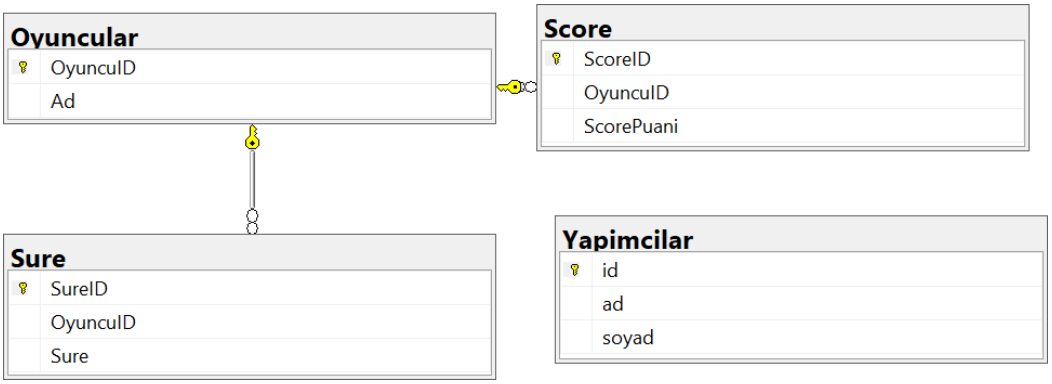
id	ad	soyad
1	Şimal	Bülbül
2	Nisa	Usta
3	Muham...	Beyiş

Şekil 17:Yapımcılar

```
private void listele()
 {string conString = "Data Source=DESKTOP-65RMS34;Initial
Catalog=gameProject222;Integrated Security=True";
    SqlConnection connect = new SqlConnection(conString);
    connect.Open();
    List<string> list = new List<string>();
    SqlCommand cmd = new SqlCommand($"SELECT yapimcilar.ad, yapimcilar.soyad
FROM yapimcilar", connect);
    DataTable dt = new DataTable();
    using (SqlDataAdapter adtr = new SqlDataAdapter(cmd))
    {adtr.Fill(dt);}
    label1.Text = $"{dt.Rows[0][0].ToString()} {dt.Rows[0][1].ToString()}";
    label2.Text = $"{dt.Rows[1][0].ToString()} {dt.Rows[1][1].ToString()}";
    label3.Text = $"{dt.Rows[2][0].ToString()} {dt.Rows[2][1].ToString()}"; }
```

Bu kod, `listele` metodu içinde bir SQL sorgusu çalıştırarak `yapimcilar` tablosundaki yapımcıların ad ve soyad bilgilerini çekip, sonuçları etiketler (`label1`, `label2`, `label3`) üzerinde gösterir. Kod, veritabanına bağlanır, verileri çeker ve bu verileri ekranda gösterir.

VERİ TABANI DİYAGRAM



Şekil 18 :Veri tabanı Diyagram

Kaynakça

C# Tutorial - Make a flappy bird game in windows form

https://www.youtube.com/watch?v=yUCCv-sFUDQ&t=540s&ab_channel=MooICT

Timer Control Example in C# - Windows Form Application

https://www.youtube.com/watch?v=HscX2kz72Zk&ab_channel=KhushalRajani

How to use Timer Control in Windows Forms C#

https://www.youtube.com/watch?v=iUFk9uMabKg&ab_channel=winforms