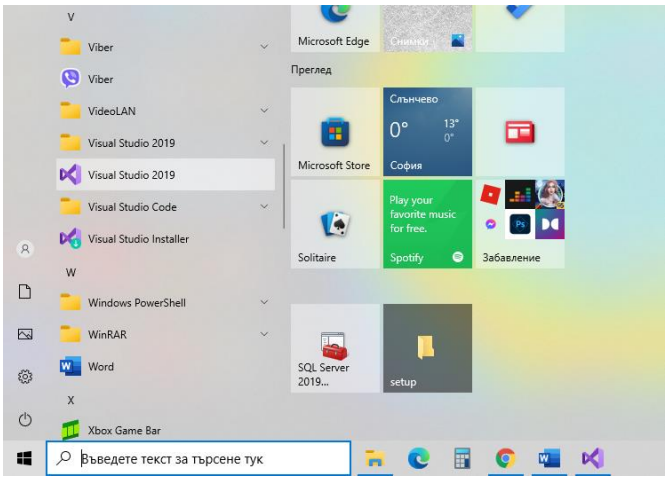
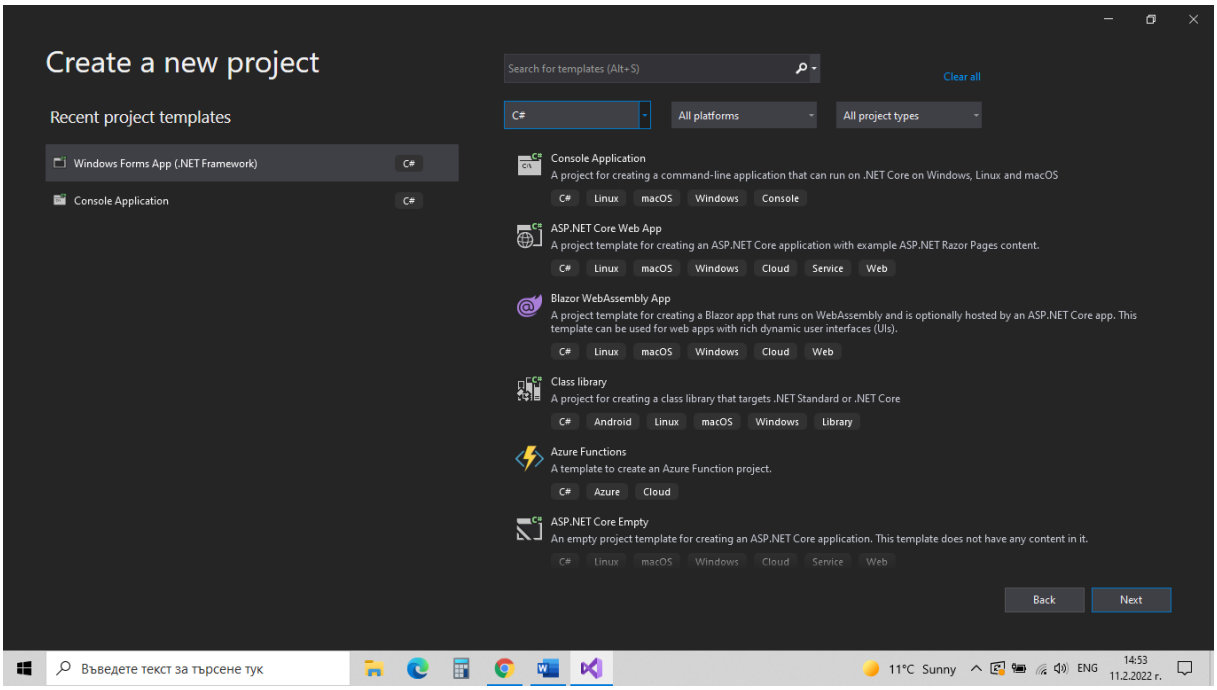
FWINDOWS FORMS UYGULAMASI VE C# KULLANARAK FLAPPY BIRD OYUNU YAPMAK

Adım 1: Yeni bir proje oluşturmak

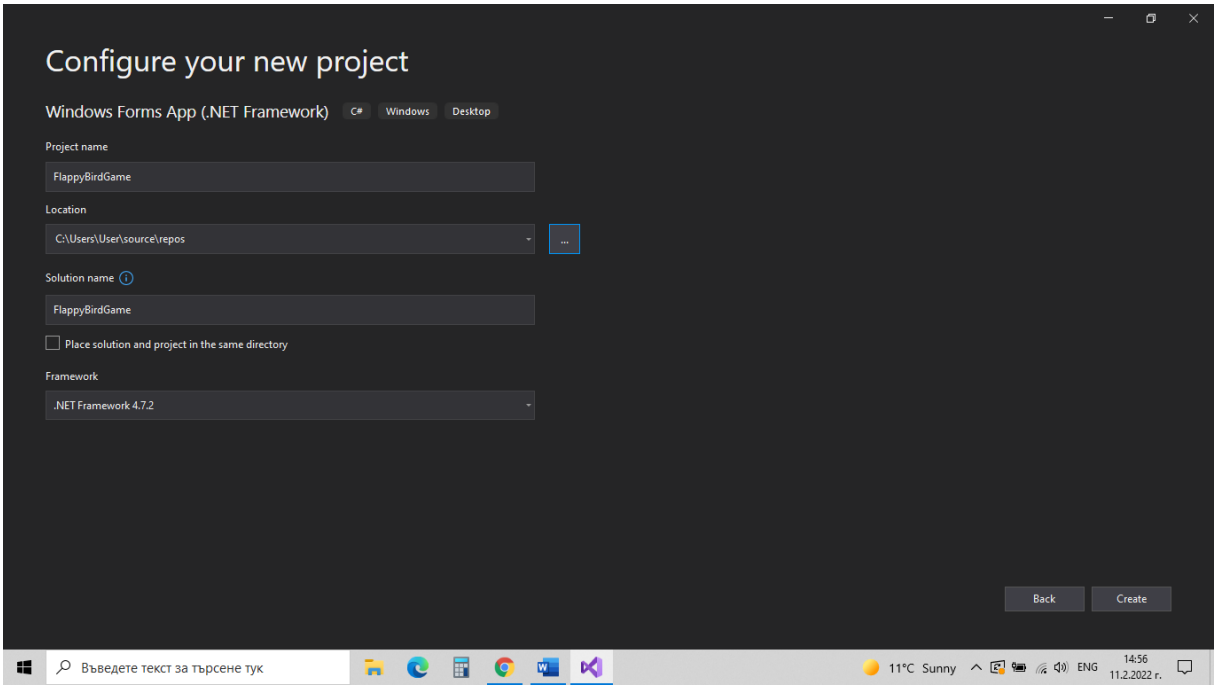
Visual Studio 2019 Geliştirme ortamını başlatın



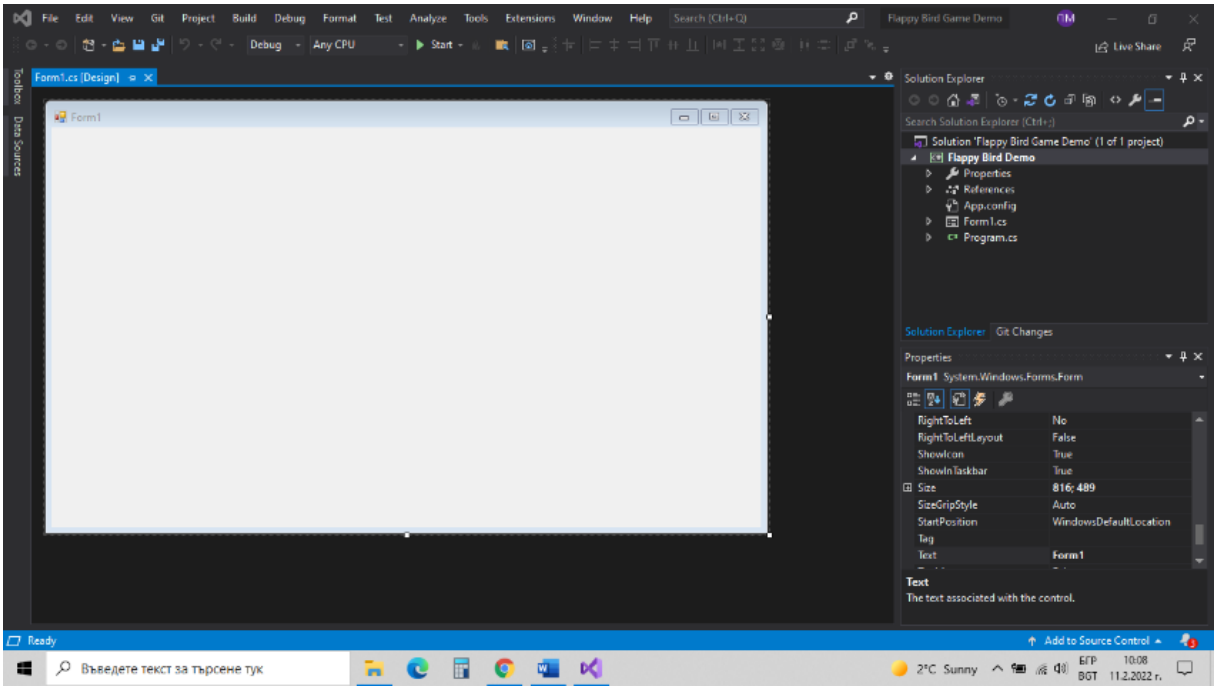
Yeni proje oluştur, Windows Forms Uygulaması (.NET Framework) ve #C'yi seçin.



Proje adını ve Solution adını yazın. istersen yerini değiştirtirebilirsiniz

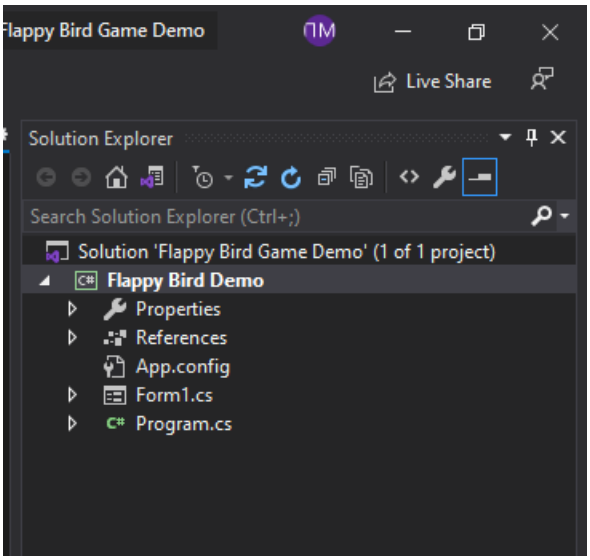


Ana ekranınızda oluşan • **Form1**



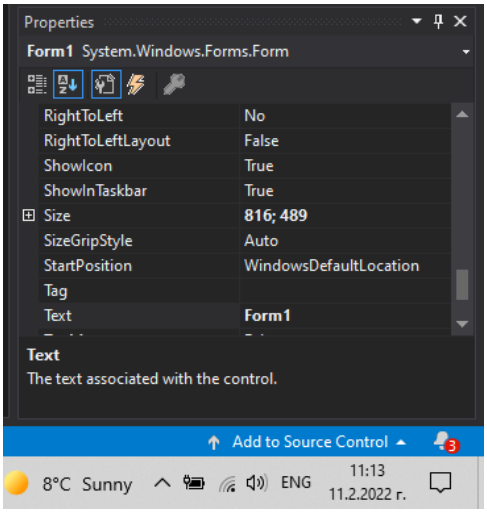
Pencerenin ortasında oyun alanımızı temsil edecek Form1 var. Bu form, bazı özelliklere sahip, önceden oluşturulmuş bir nesnedir ve üzerinde bazı eylemler gerçekleştirilebilir

- **Solution Explorer**



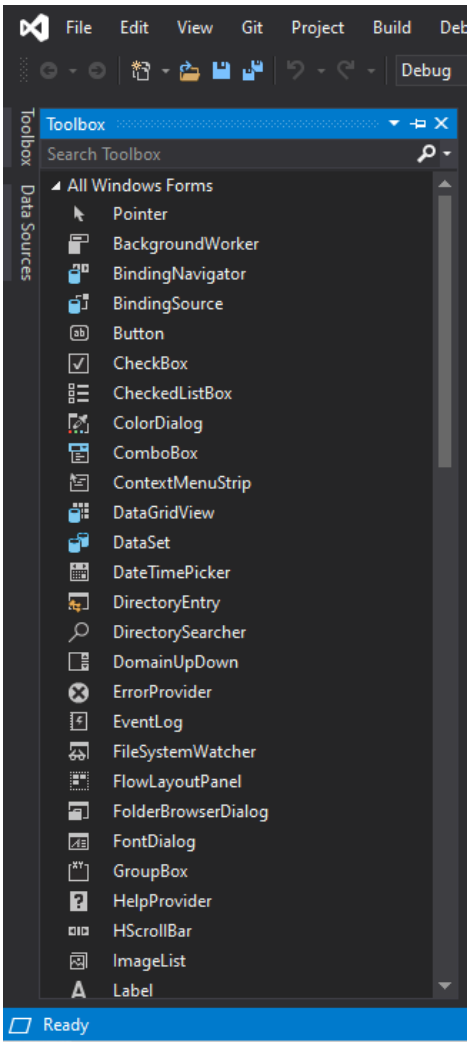
Sağ tarafta, Solution Explorer'da topladığımız projemizin tüm dosyaları var. Projemizin birkaç dosyadan oluştuğunu görebiliyoruz. Şimdi bunlardan ikisi ile ilgileniyoruz – Form1.cs ve Program.cs. Form1.cs tasarım görünümünde projemizin nasıl görüneceğini görebilir ve üzerine elemanlar ekleyebiliriz. Program.cs dosyasında program kodunu görebiliriz ve onu da yazabiliriz.

- **Git Changes**



Burada iki önemli sekme var – özellikler ve olaylar. Git Changes'de her nesnenin özelliklerini görebilir ve ayarlayabiliriz. Eventler, nesneyle ne yapacağımızı kod yazmamıza izin verir.

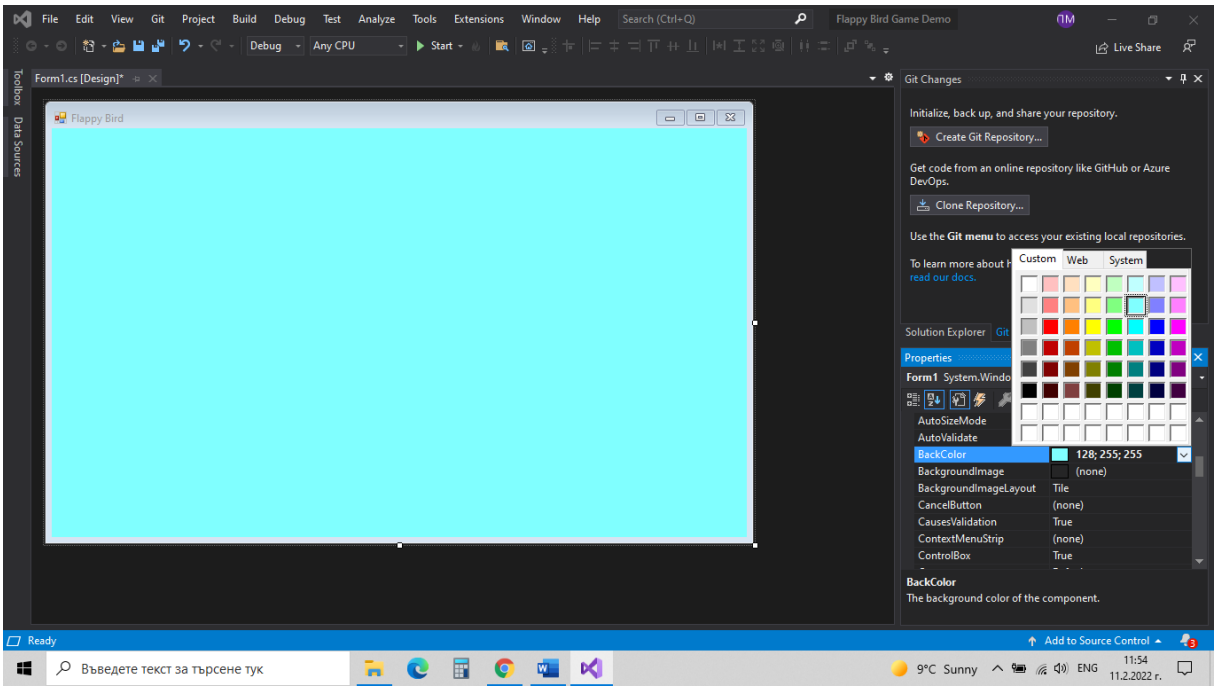
- Toolbox



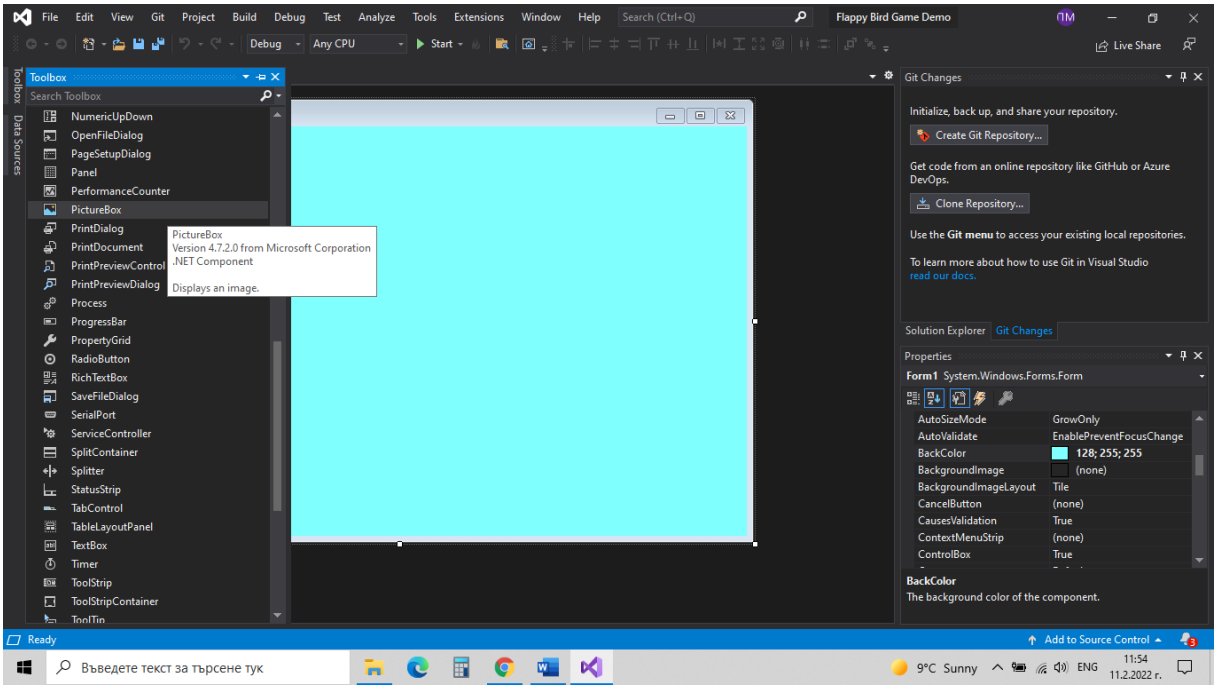
Solda bulunan araç kutusu, kullanmamız gereken tüm nesneleri seçmemizi ve seçmemizi sağlar.

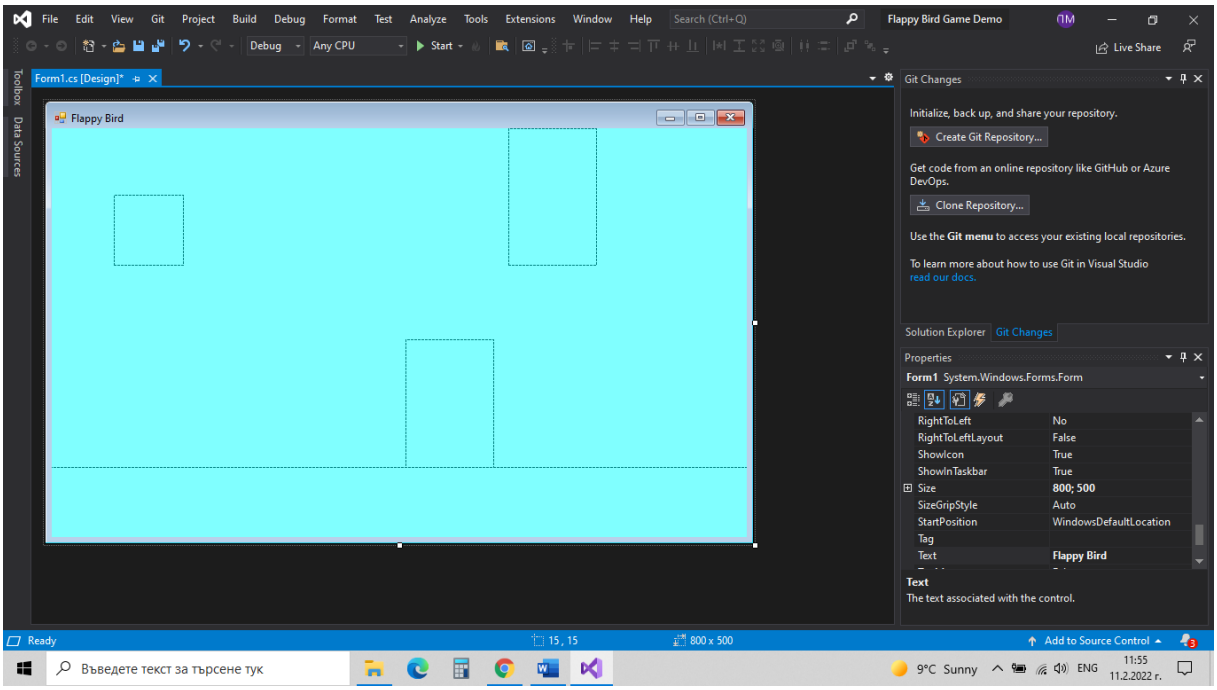
Adım 2: Form1'in özelliklerini ayarlama:

- (Name): gameField
- Boyut: 800;500
- Metin: Flappy Bird
- ArkaRenk: Mavinin bir tonu



Adım3: 4 nesne ekleme ve ayarlama pixtureBox

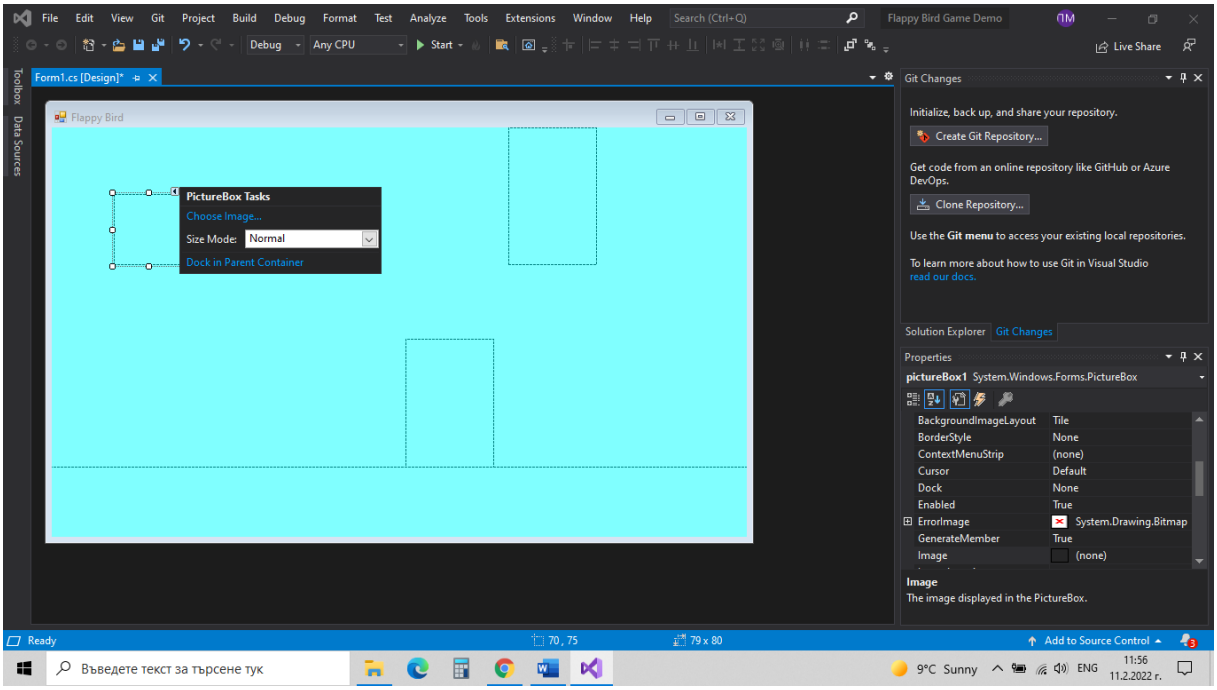


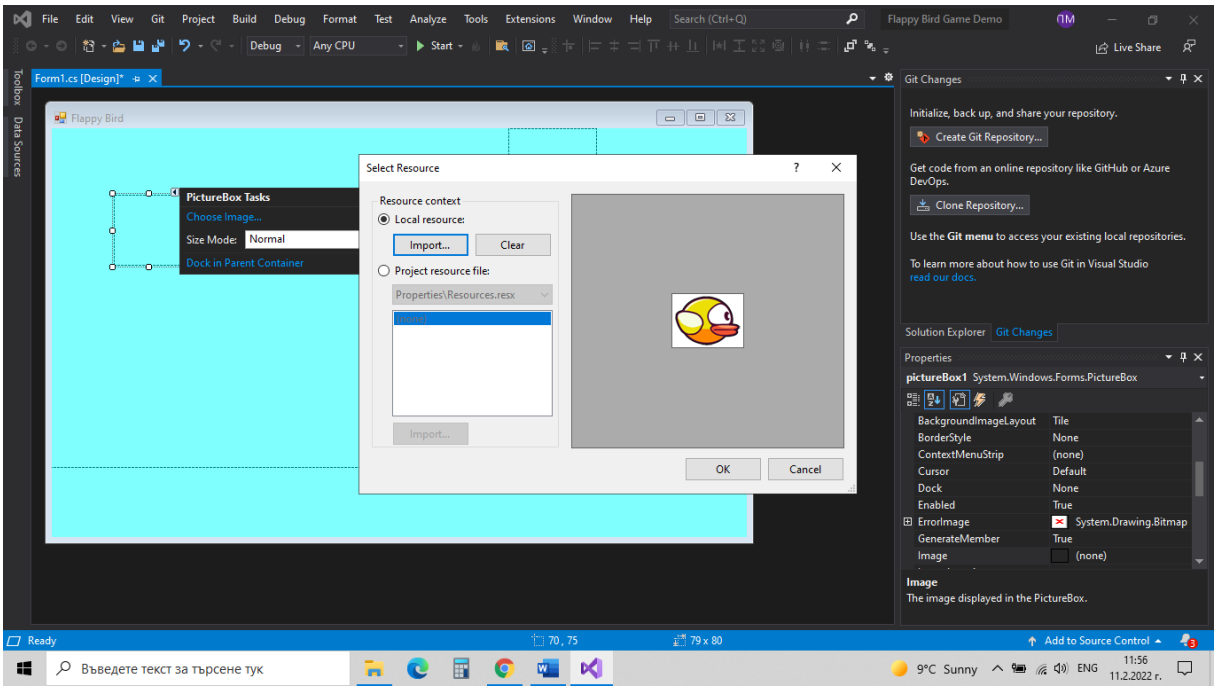


PictureBox1 kuş olacak, pictureBox2 bir boru olacak, pictureBox3 diğer boru olacak ve pictureBox4 zemin olacak.

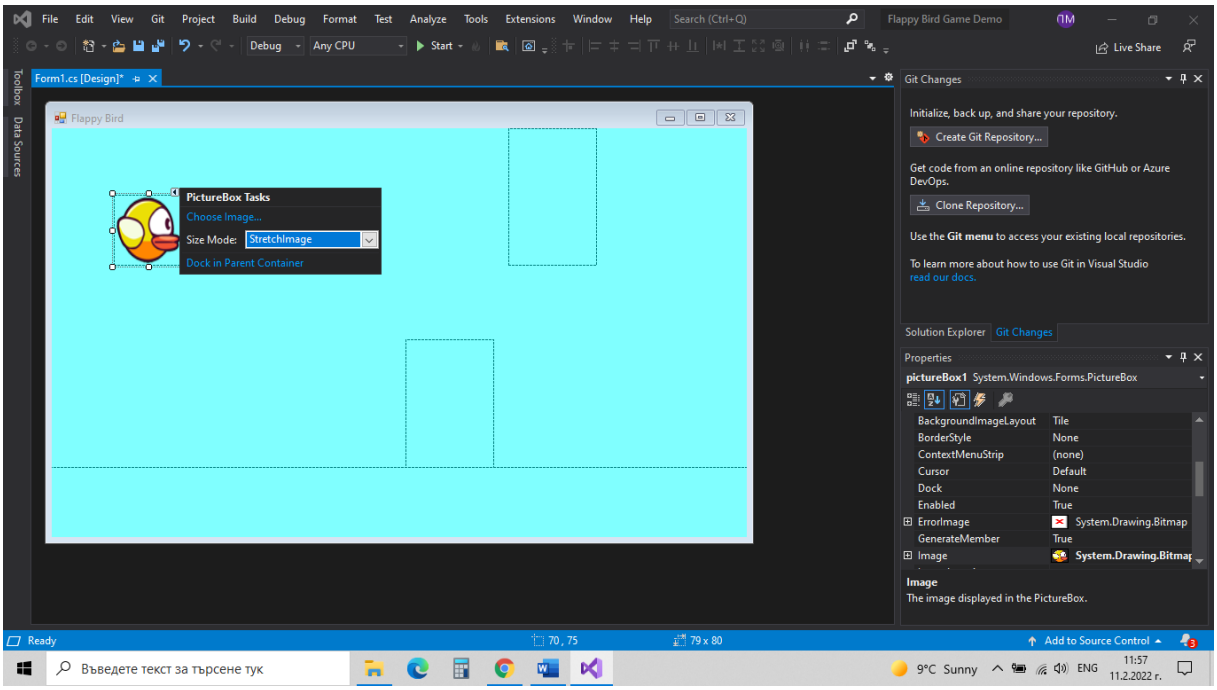
pictureBox1 ayarları

- Choose image -> Local Resource -> Import…

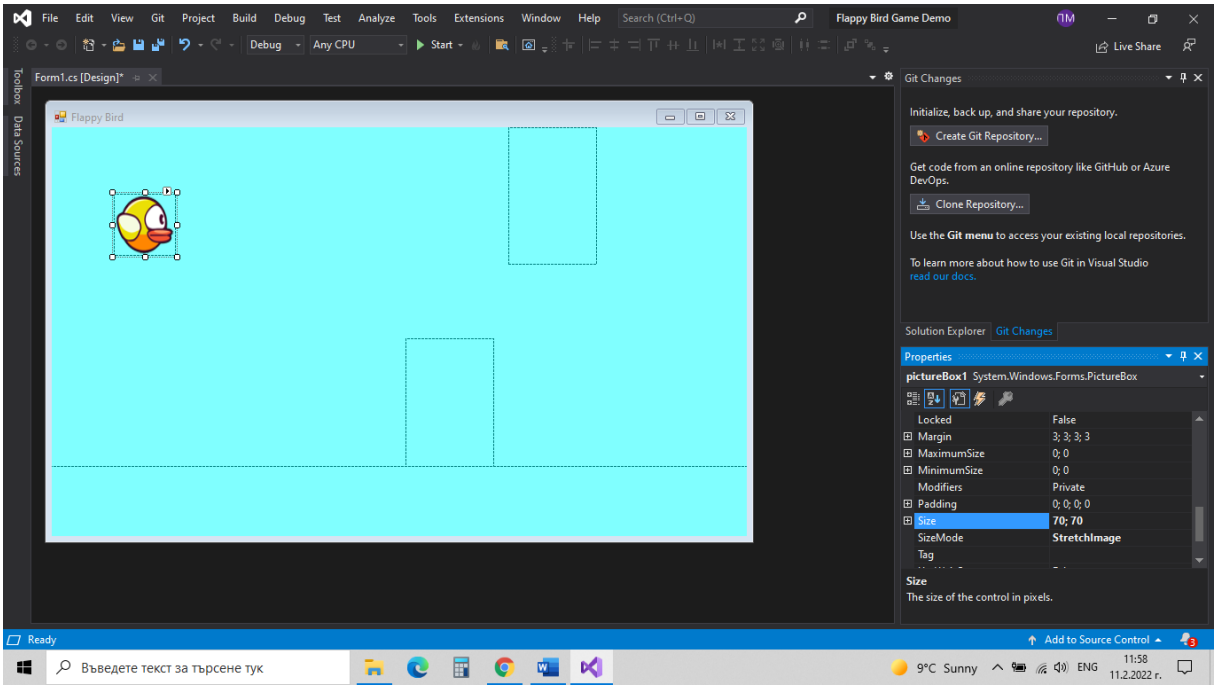




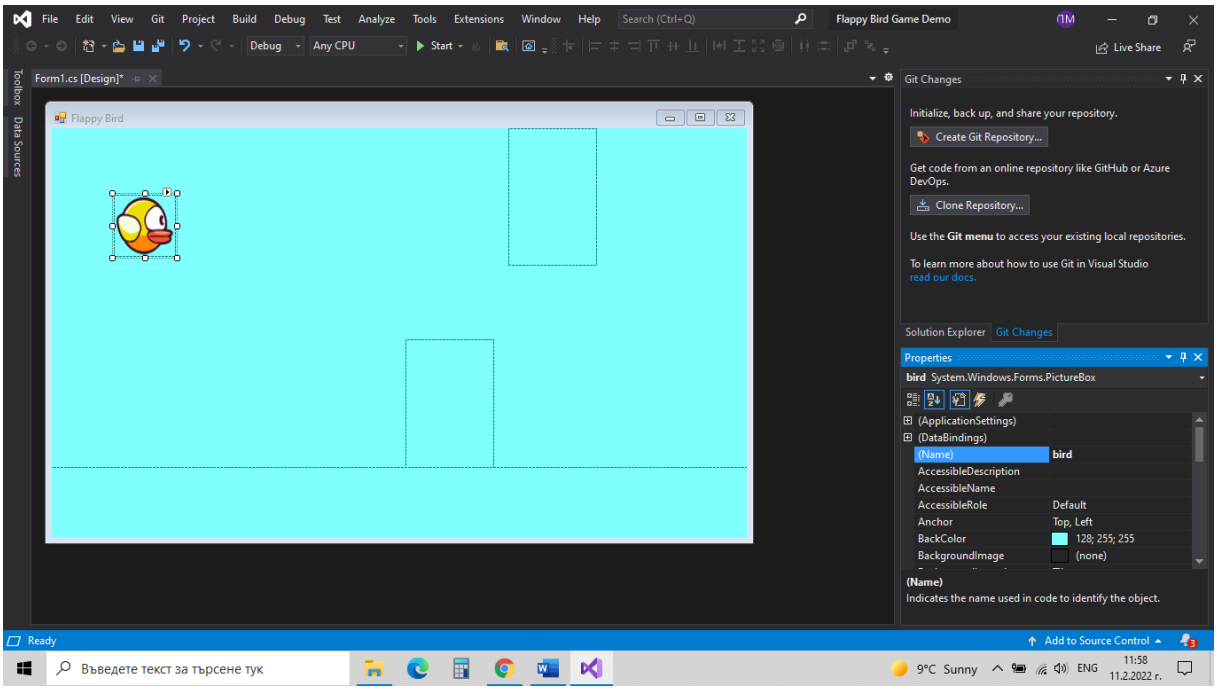
- Size Mode: StretchImage



- Boyut: 70;70



- (İsim): bird



Diğer resimler için de aynı ayarı yapın:

pictureBox2 ayarı

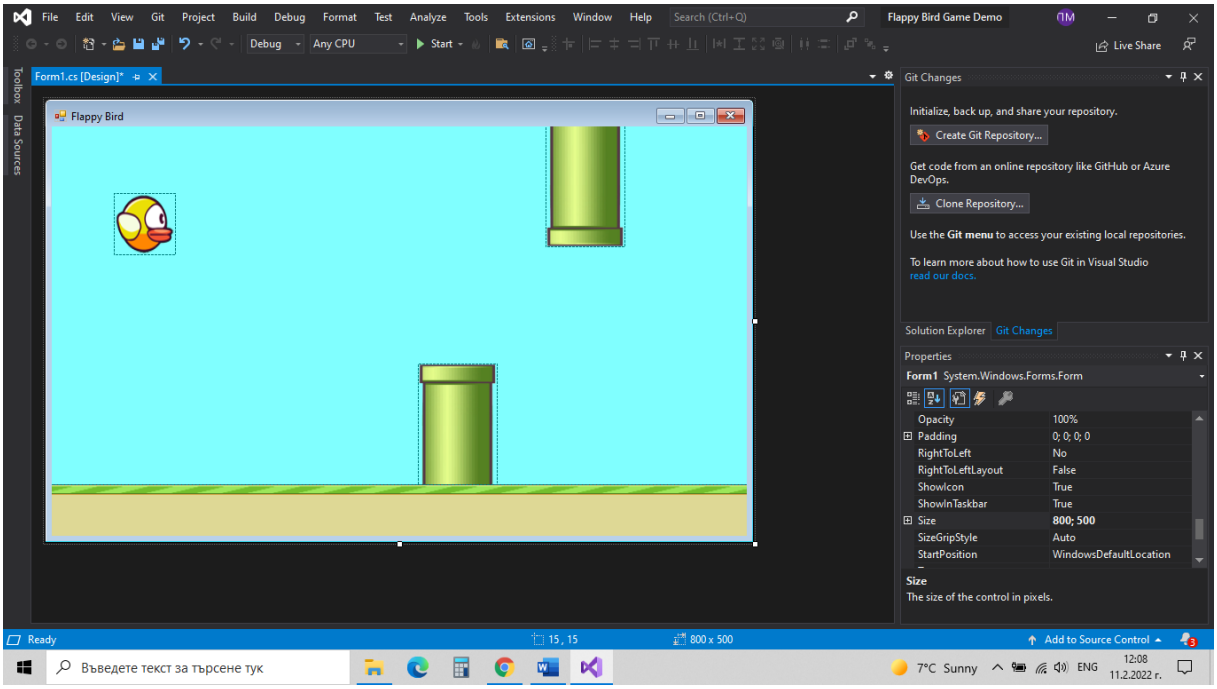
- (İsim): pipeUp
- Size Mode: StretchImage
- Boyut: 90;140

pictureBox3 ayarı

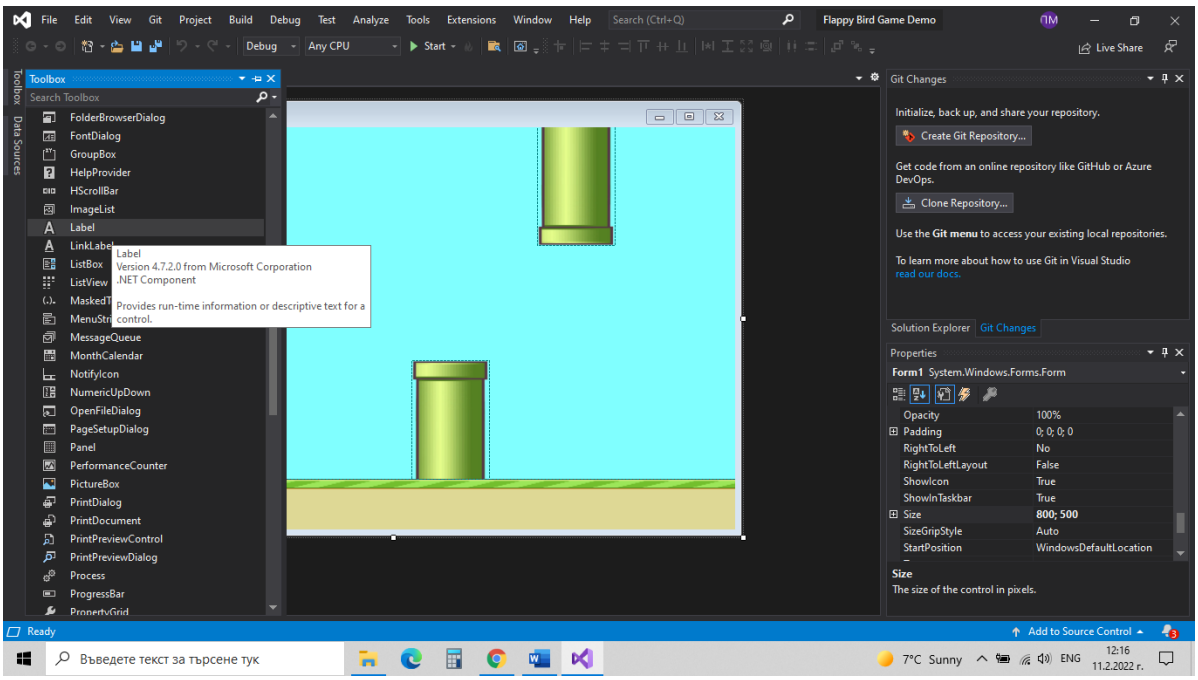
- (İsim): pipeDown
- Size Mode: StretchImage
- Boyut: 90;140

pictureBox4 ayarı

- (İsim): ground
- Size Mode: StretchImage
- Boyut: 800;60

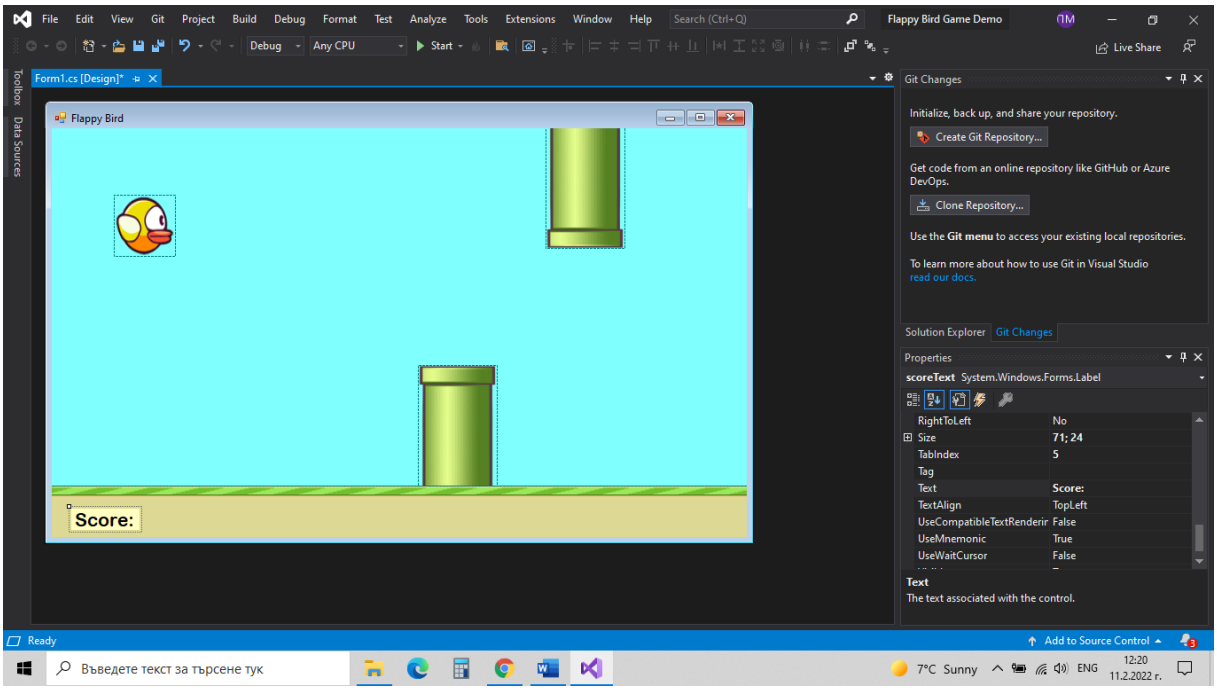


Ayar 4: Nesne Etiketi ekleme ve ayarlama

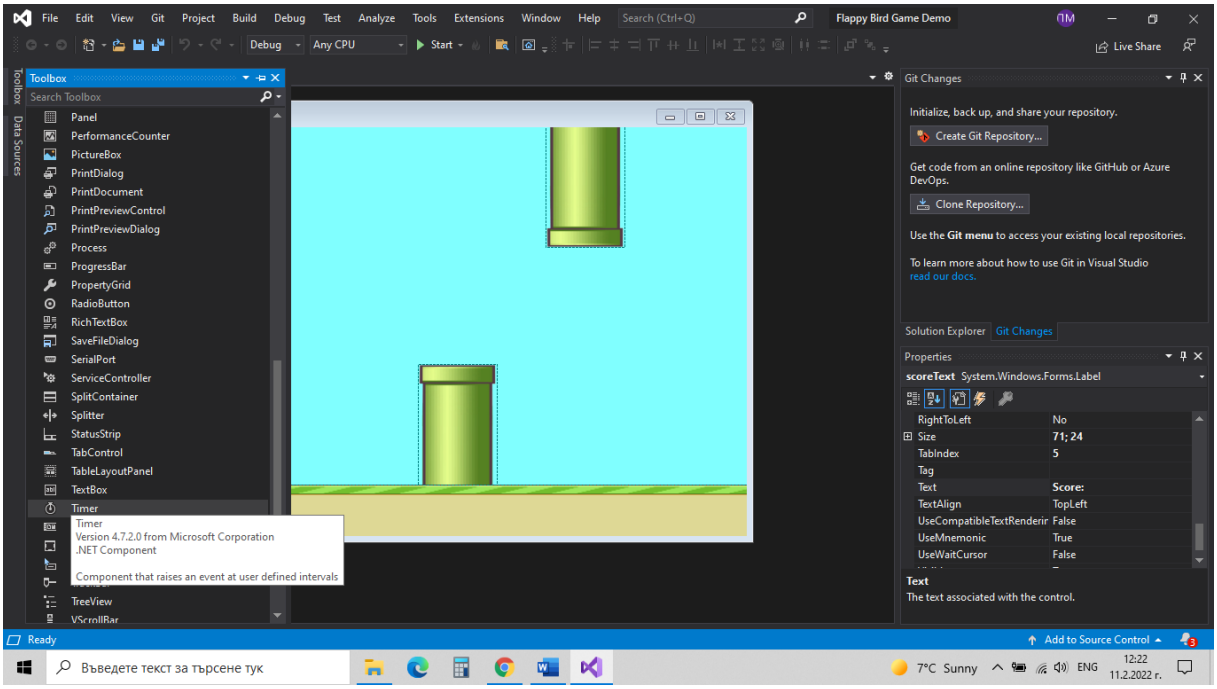


label1 ayarı:

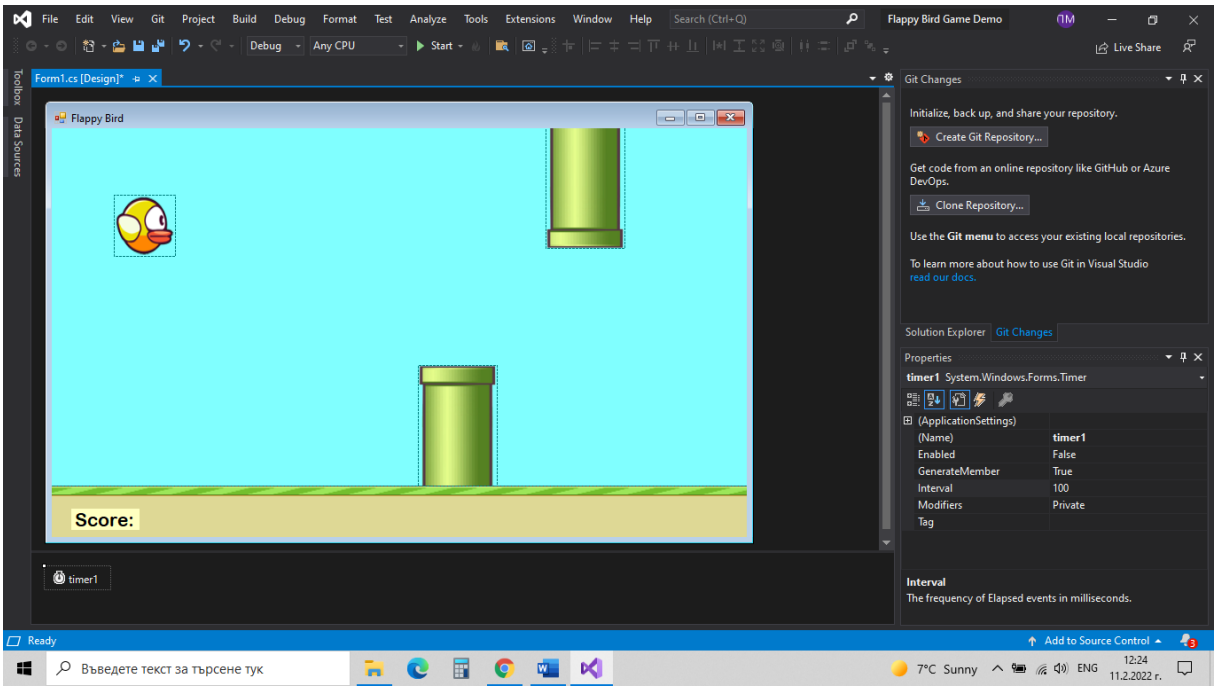
- (İsim): scoreText
- ArkaRenk: Zemin ile aynı -
  Font: Arial; 16
- Metin: Score



Adım 5: element zamanlayıcı ekleme



Bu, oyun alanında görünmeyen bir unsur yaratır.



timer1 ayarı

- (İsim): gameTimer -

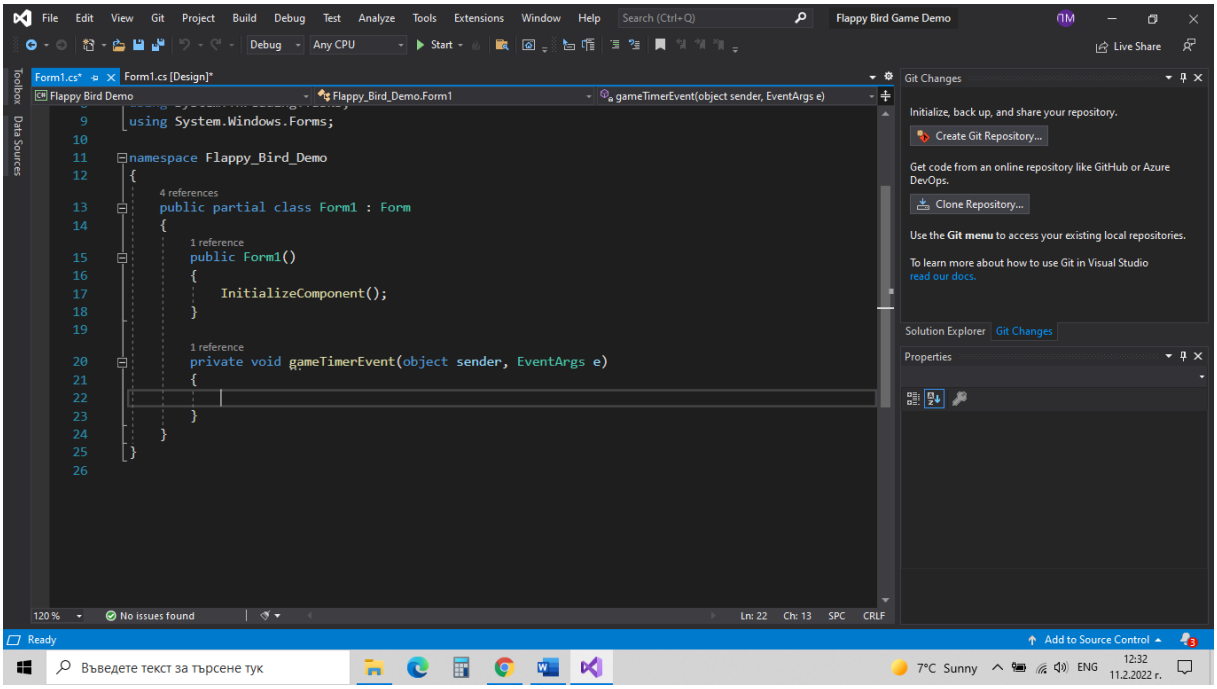
Enabled: true

- Interval: 20

Ve bu element için bir event oluşturuyoruz

- Tick: gameTimerEvent

Bu event'e tıkladığımızda kodunu görebiliriz. Bu event bir yöntem oluşturur:

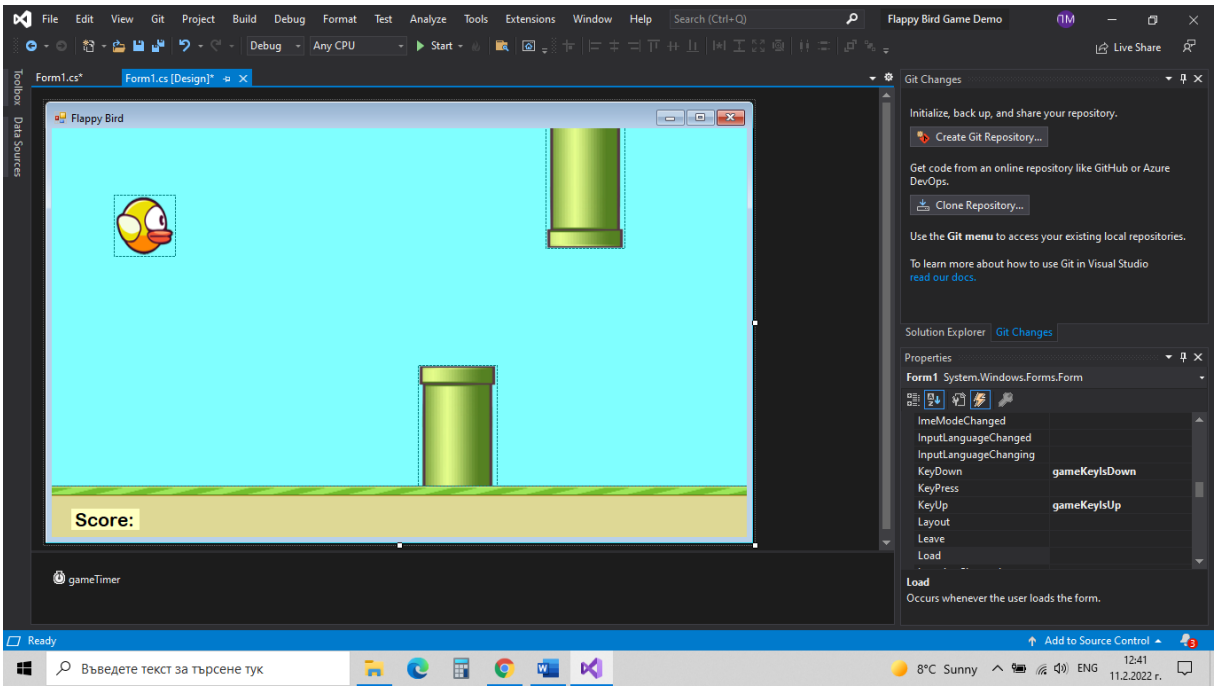


Bu yöntemde, her zaman gerçekleşen eylemleri tanımlamak için kod yazacağız.

Ste6. adım: Form1 ana formunda iki olay ayarlama: Oyun klavye tuşları ile oynanacaktır. Bir tuşa basıldığında kuş havalanacaktır

Tuşa basılmadığında kuş düşmeye başlayacaktır. Bu nedenle, gameField ana formuna uygulanan iki olaya ihtiyacımız var:

- aşağı tuş : gameKeyIsDown -
  yukarı tuş : gameKeyIsUp

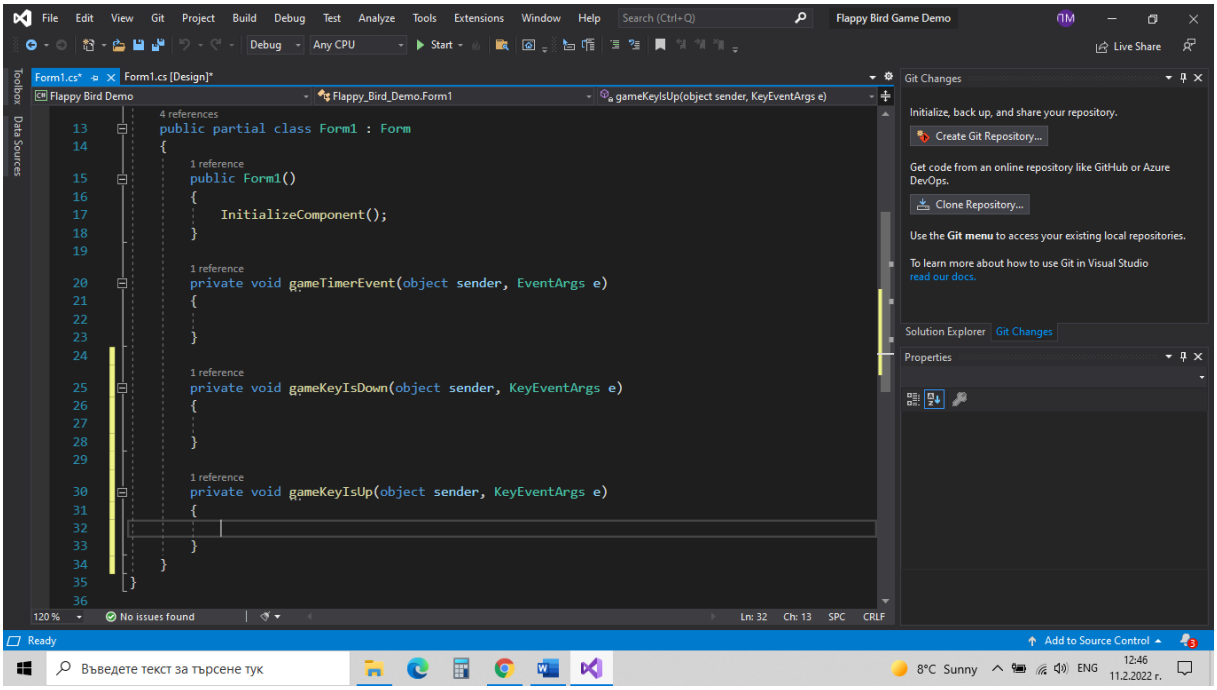


Bu olayları isimlendirerek onları yaratırız.

Peki, şimdi C# betiğinde ne var?

Form1 sınıfı ve int dört yöntemi oluşturuldu – Form1(), GameTimerEvent(), GameKeyIsDown(), GameKeyIsUp().

Buradan oyunumuzun kodunu yazmaya başlıyoruz..



Adım 7: değişkenleri oluşturmak

Başlangıçta üç değişkene ihtiyacımız var - biri boruların hızı için, biri kuşun düşeceği yerçekimi için ve biri de skor için. Her üç değişkene de tamsayılar başlangıç değerleri veriyoruz ve bunlar uygun değilse değiştirebiliyoruz. Sonuç, elbette, başlangıçta 0'dır.

```
int pipeSpeed = 8;
int gravity = 5;
int score = 0;
```

8. Adım: Kuşu aşağı hareket ettirmek

Kuş bir nesnedir. Nesnenin ve noktanın adını yazdığımızda her nesnenin özelliklerine ve yöntemine ulaşabiliriz. Örneğin, "kuş". Böylece kuş elementinin özelliklerine bakabilir ve ihtiyacımız olanı seçebiliriz. Aşağı hareket için Top özelliğine ihtiyacımız var. Bu özelliğin değerini artırarak, resmin üstü ile tarlanın üstü arasındaki mesafeyi artıracağız ve böylece kuş aşağı doğru hareket edecektir. Değişken yerçekimimiz var ve onunla birlikte mülkü artıracağız. Kuş her zaman aşağı doğru hareket ediyor, bu yüzden bu kodu GameTimerEvent() yöntemine yazmamız gerekiyor.

```
1 reference
private void gameTimerEvent(object sender, EventArgs e)
{
    bird.Top += gravity;
}
```

Adım 9: tuşa basıldığında kuşu yukarı ve tuşlara basılmadığında aşağı hareket ettirin

Özellik(property) azaltırsak kuş yukarı çıkar. Bu, yerçekimini negatif bir sayıya değiştirerek yapılabilir. Bu hareket bir tuşa basıldığında gerçekleştirilmelidir, bu nedenle kod gameKeyIsDown() yöntemine yazılacaktır. Anahtar yukarıdayken tekrar aşağı hareket etmeye başlamak için yerçekimini tekrar pozitif bir sayı olacak şekilde ayarlamamız gerekiyor ve bu gameKeyIsUp() yönteminde yazılmalıdır.

```
1 reference
private void gameKeyIsDown(object sender, KeyEventArgs e)
{
    gravity = -5;
}

1 reference
private void gameKeyIsUp(object sender, KeyEventArgs e)
{
    gravity = 5;
}
```

Adım 10: boruları sola hareket ettirmek

Borular ayrıca pipeUp ve pipeDown adlı nesnelerdir ve ayrıca özellikleri vardır. Sola doğru hareket edecekleri için onların sol özelliklerini (property) kullanacağız. Bu özellik, görüntünün sol tarafından alanın sol tarafına olan mesafeyi ayarlar. Bu mesafeyi azaltırsak borular sola doğru hareket edecektir. Ve elbette kodu gameTimerEvent() yöntemine yazacağız çünkü bu hareket her zaman oluyor.

```
1 reference
private void gameTimerEvent(object sender, EventArgs e)
{
    bird.Top += gravity;
    pipeDown.Left -= pipeSpeed;
    pipeUp.Left -= pipeSpeed;
}
```

Adım 11: boruların tekrar görünümü

Şimdi borular sola hareket ediyor ve oyun alanının dışına çıkıyor. Ekranın solunda tekrar görünmelerini sağlamak için soldan çıkıp çıkmadıklarını kontrol etmemiz gerekiyor. Sol tarafları 0'dan küçükse, onu daha büyük bir sayı yapacağız. gameTimerEvent() yönteminde tekrar yazılırsa, mantıksal koşullar için bir operatör kullanacağız.

```
if (pipeUp.Left < 0)
    pipeUp.Left = 900;
if (pipeDown.Left < 0)
    pipeDown.Left = 1300;
```

Adım 12: Bir çarpışma varsa oyunu sonlandırın

Şimdi tek bir şey yapacağımız kendi fonksiyonumuzu (yöntemimizi) yaratacağız - zamanlayıcıyı durdurmak. Zamanlayıcıyı durdurmak için yerleşik Stop() yöntemini kullanacağız.

```
0 references
private void endGame()
{
    gameTimer.Stop();
}
```

Şu anda hiçbir şey olmuyor çünkü kuş bir boruya ya da yere çarptığında bu yöntemi çağırmamız gerekiyor.

Bounds.IntersectsWith() çarpışma kontrolü için yerleşik yöntemi kullanacağız. Bu yöntem, doğru veya yanlış sonuçları verir, böylece bir if operatöründe mantıksal bir koşul olarak kullanabiliriz.

Çarpışma için üç kontrol yapacağız - bir boruyla, diğer boruyla ve zeminle. Bir çarpışma varsa, endGame() yöntemini çağırırız.

```
if (bird.Bounds.IntersectsWith(pipeDown.Bounds))
    endGame();

if (bird.Bounds.IntersectsWith(pipeUp.Bounds))
    endGame();

if (bird.Bounds.IntersectsWith(ground.Bounds))
    endGame();
```

Adım 13: noktaları saymas

.

```
if (pipeUp.Left < 0)
{
    pipeUp.Left = 900;
    score++;
}

if (pipeDown.Left < 0)
{
    pipeDown.Left = 1300;
    score++;
}
```

Skoru etikette görünür kılmak için kalır.

```
scoreText.Text = "Score: " + score.ToString();
```