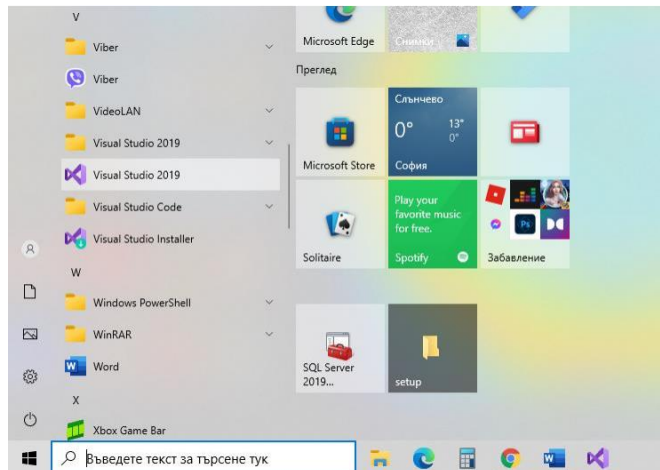


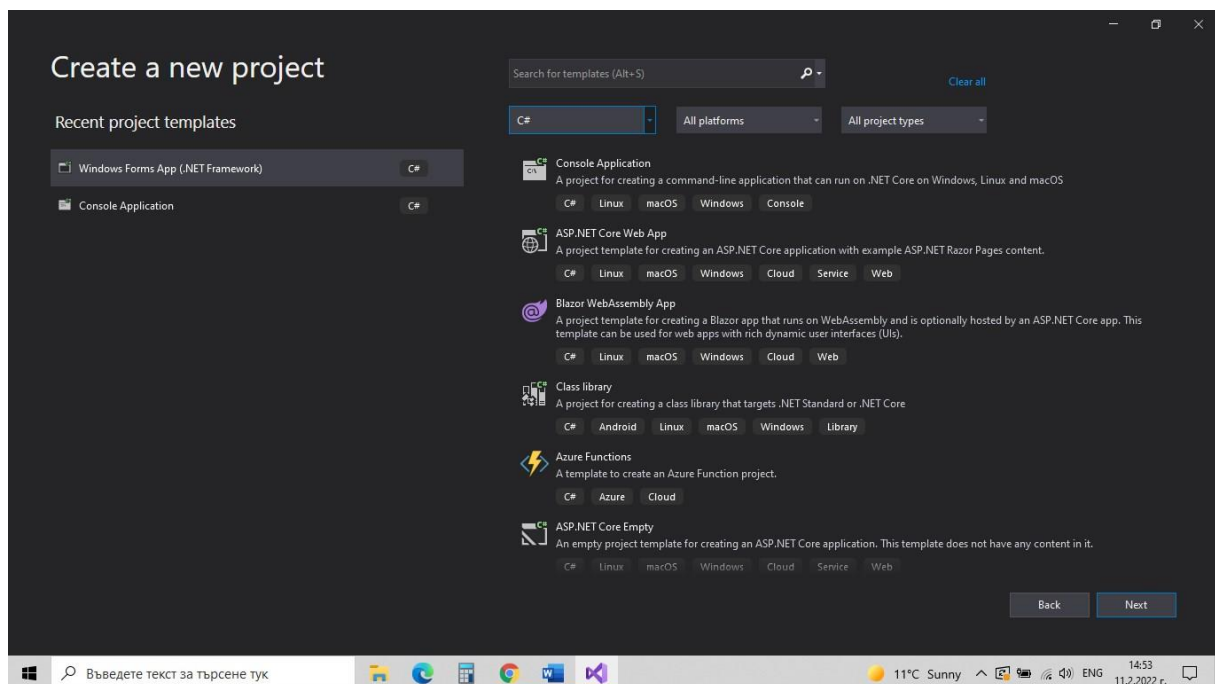
# JOGO FLAPPY BIRD USANDO WINDOWS FORMS APP AND C#

## 1º Passo: criar um projeto novo

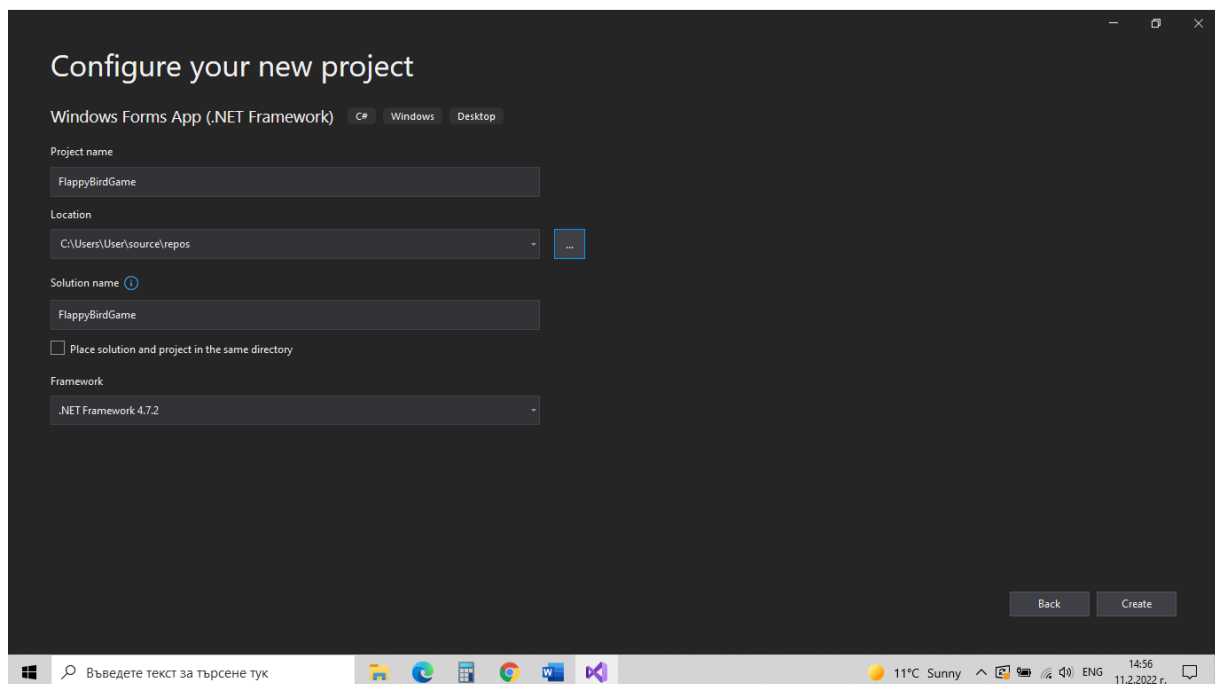
Iniciar o **Visual Studio 2019**.



Escolha **criar um projeto, Windows Forms App (.NET Framework) e C#**

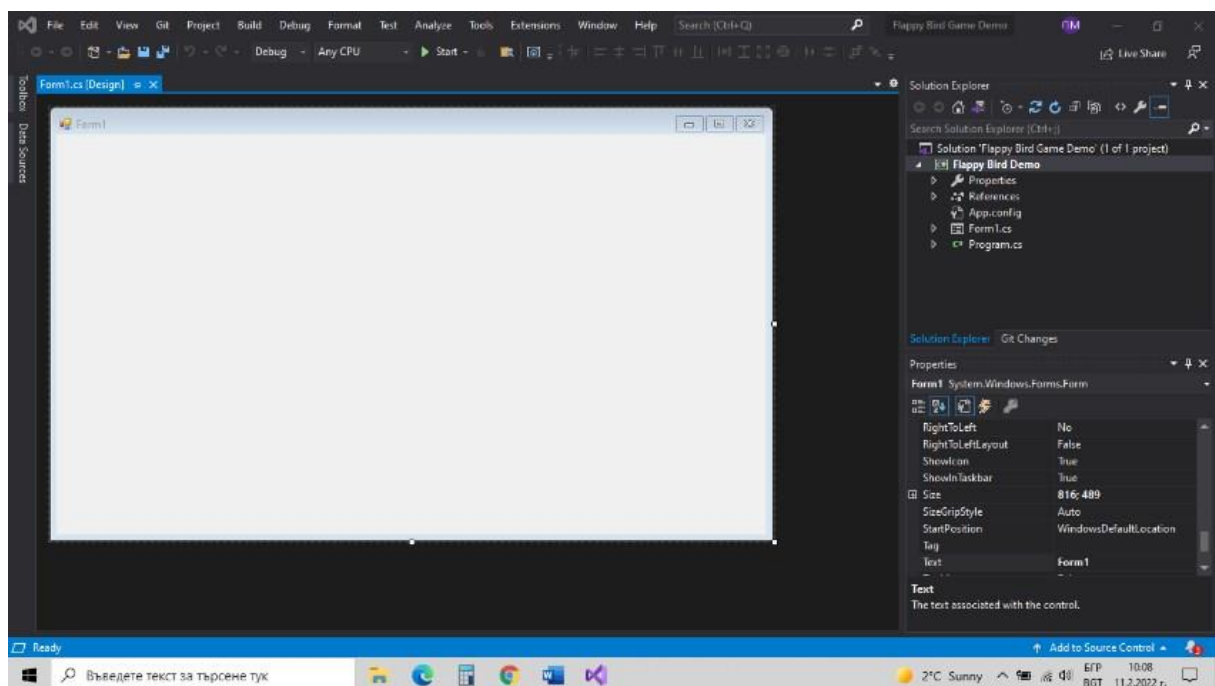


Escreva o nome do projeto e Solution name. Mude a localização se preferir.



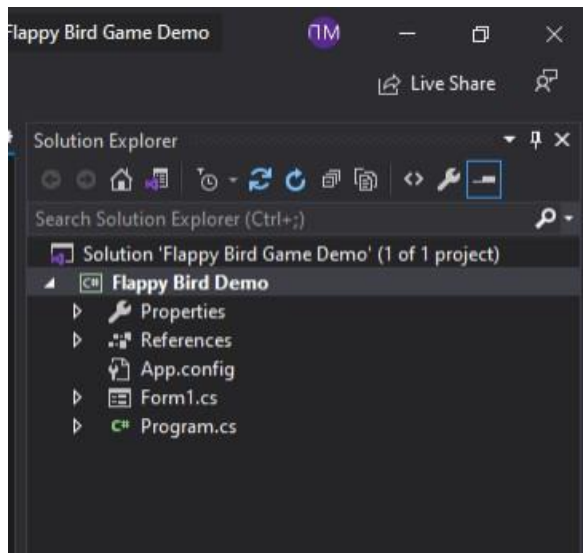
O que temos no ecrã principal?

- **Página 1**



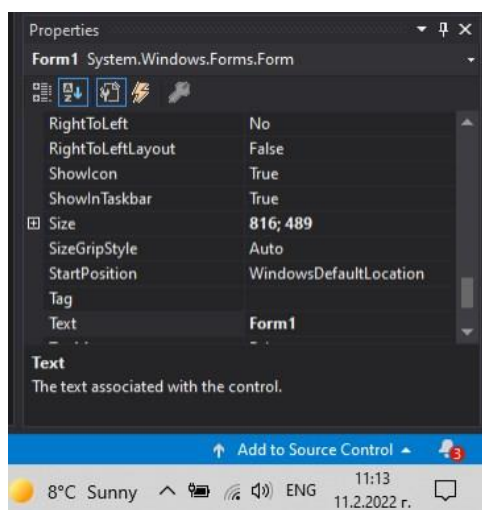
Temos o Form1 no centro da janela, que irá representar a nossa área de jogo. Este formulário é um objeto já criado que tem algumas propriedades, e algumas ações podem ser executadas sobre ele.

- **Solution Explorer**



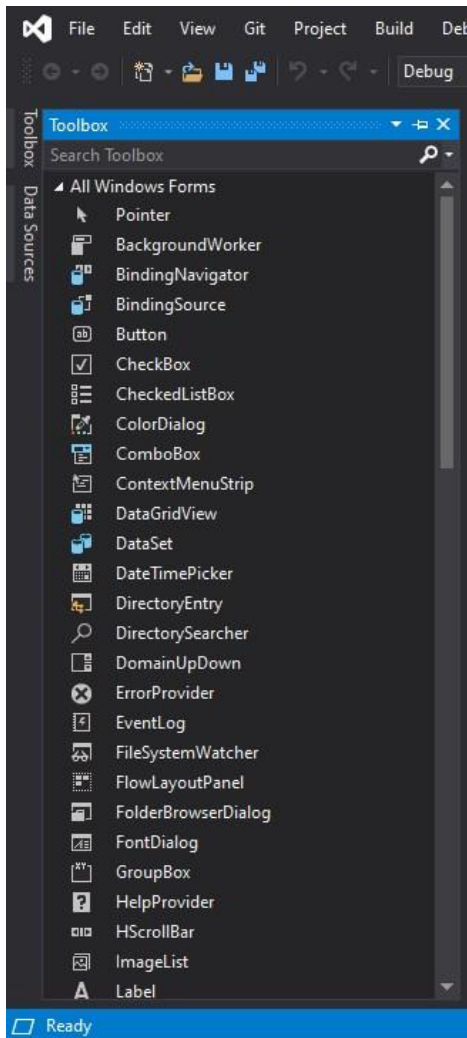
No lado direito temos todos os ficheiros do nosso projeto recolhido no Solution Explorer. Podemos ver que o nosso projeto consiste em vários ficheiros. Agora estamos interessados em dois deles - Form1.cs e Program.cs. Na visão de design do Form1.cs podemos ver como será o nosso projecto e podemos adicionar elementos sobre ele. No ficheiro Program.cs podemos ver o código do programa e também o podemos escrever.

- **Mudanças de Git**



Há aqui dois importantes separadores - **propriedades e eventos**. Podemos ver e definir as propriedades de cada objeto em Git Changes. Events permite-nos escrever o código do que fazer com o objeto.

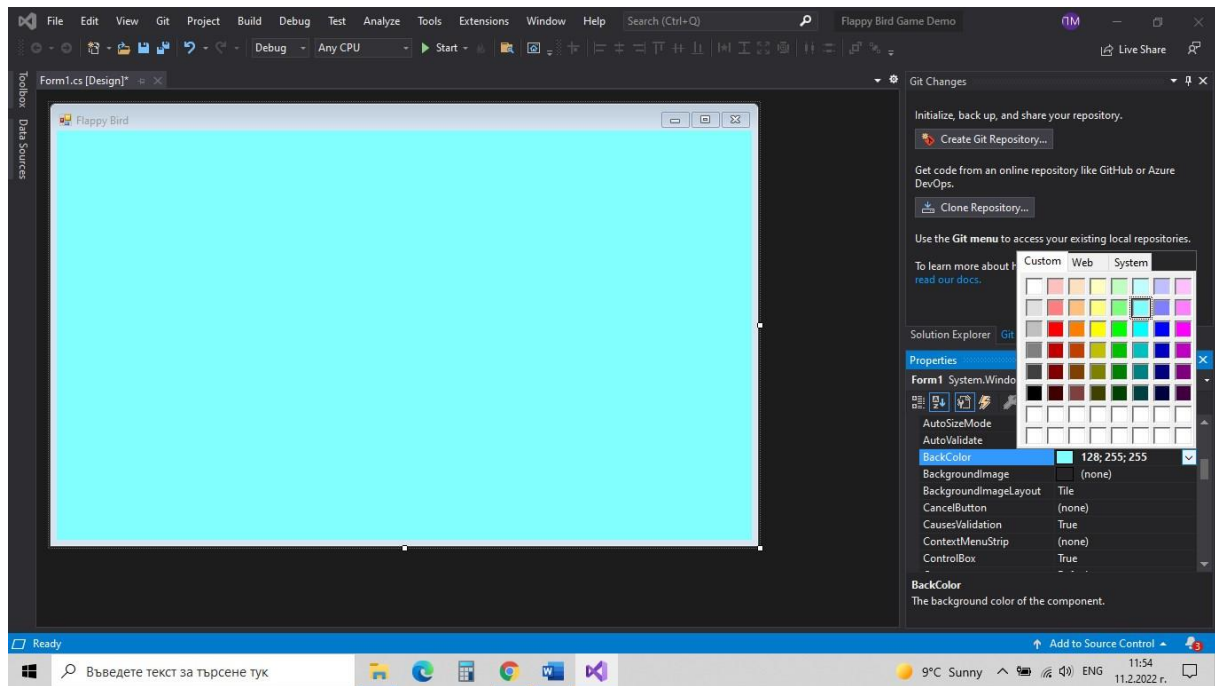
- Toolbox



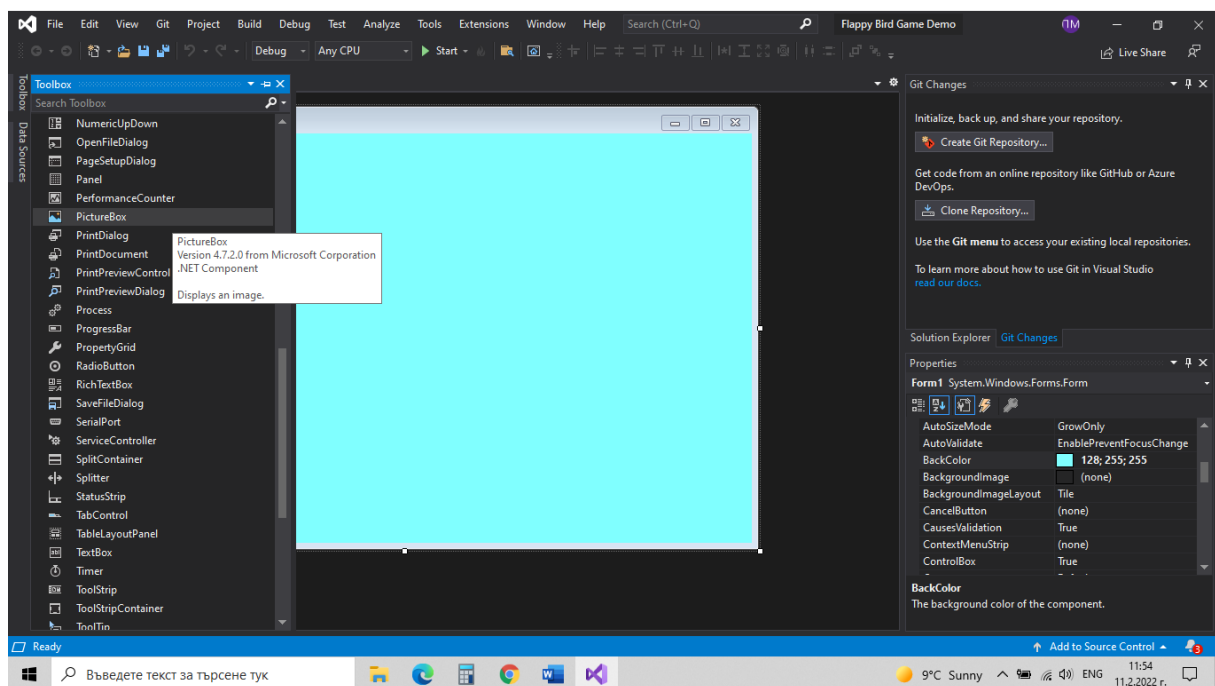
Localizada à esquerda, a caixa de ferramentas permite-nos seleccionar e escolher todos os objetos que precisamos de utilizar.

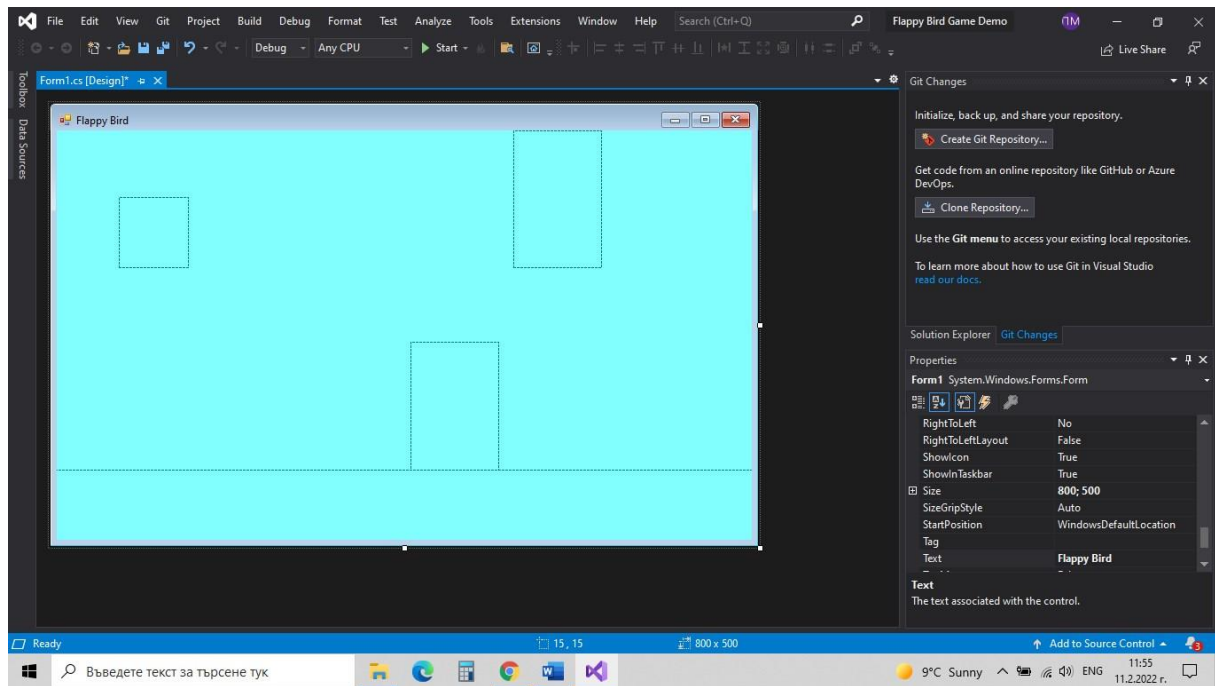
## 2º Passo: Propriedades de configuração do Formulário1:

- (Nome): gameField
- Tamanho: 800;500
- Texto: Flappy Bird
- BackColor: algum azul



### 3º Passo: Adicionar e definir 4 objetos pictureBox

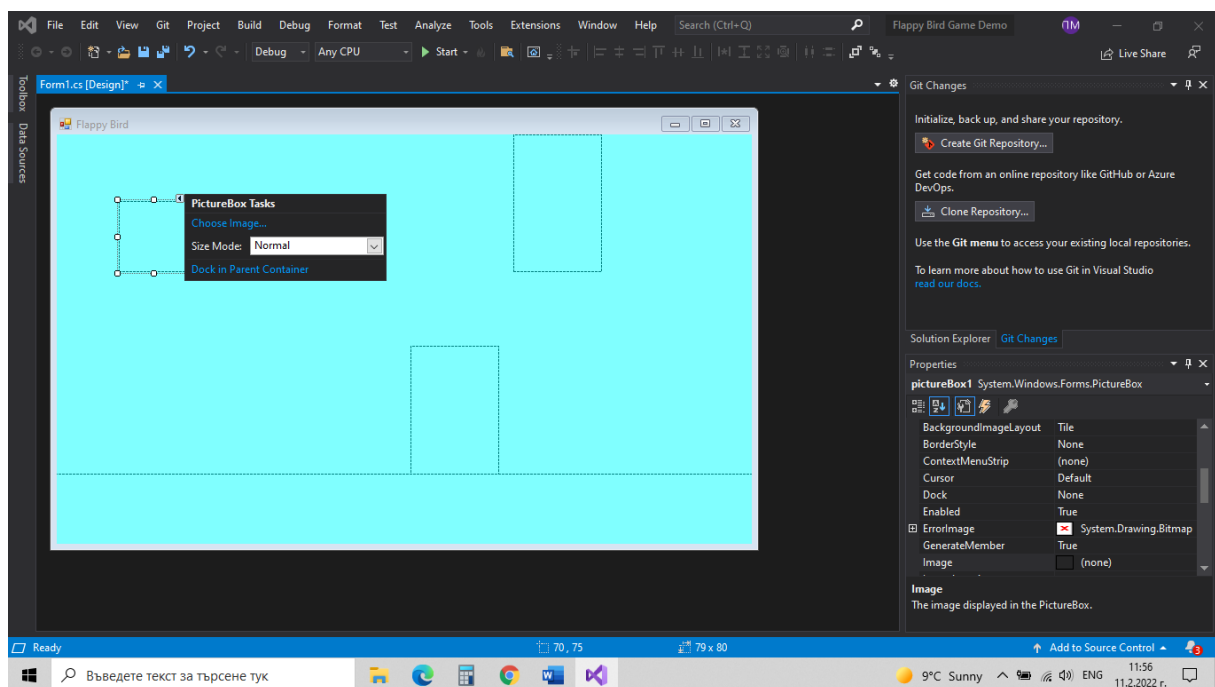


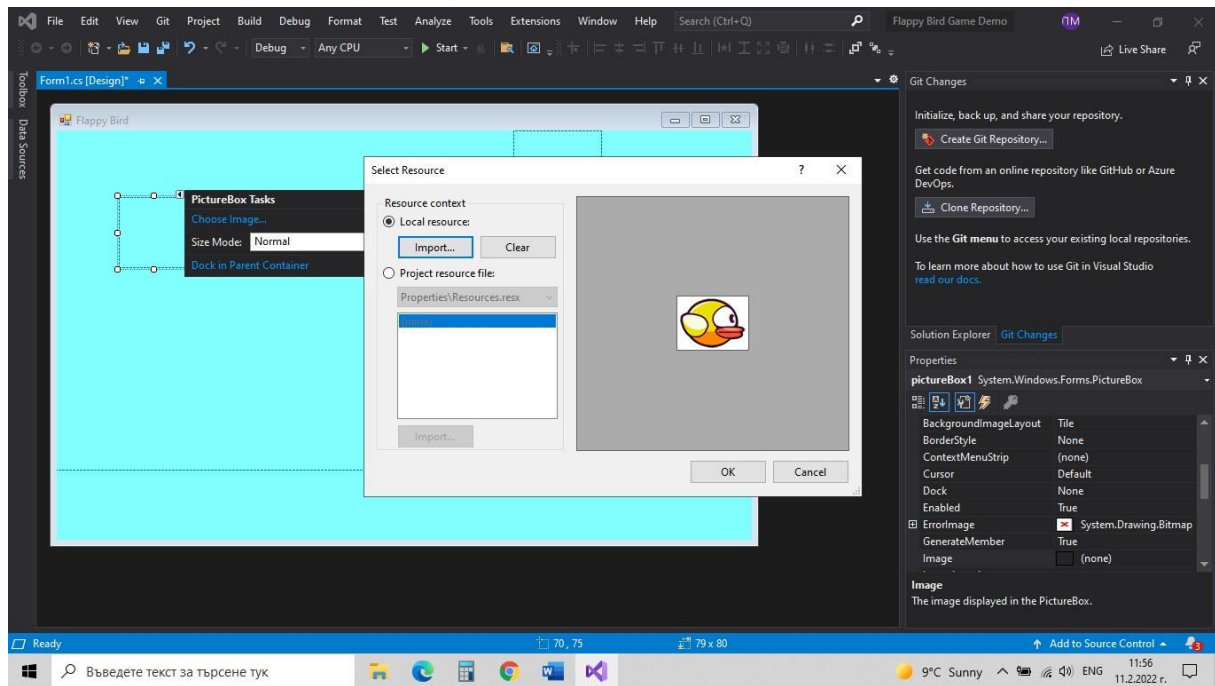


pictureBox1 será o pássaro, pictureBox2 será um tubo, pictureBox3 será o outro e pictureBox4 será o chão.

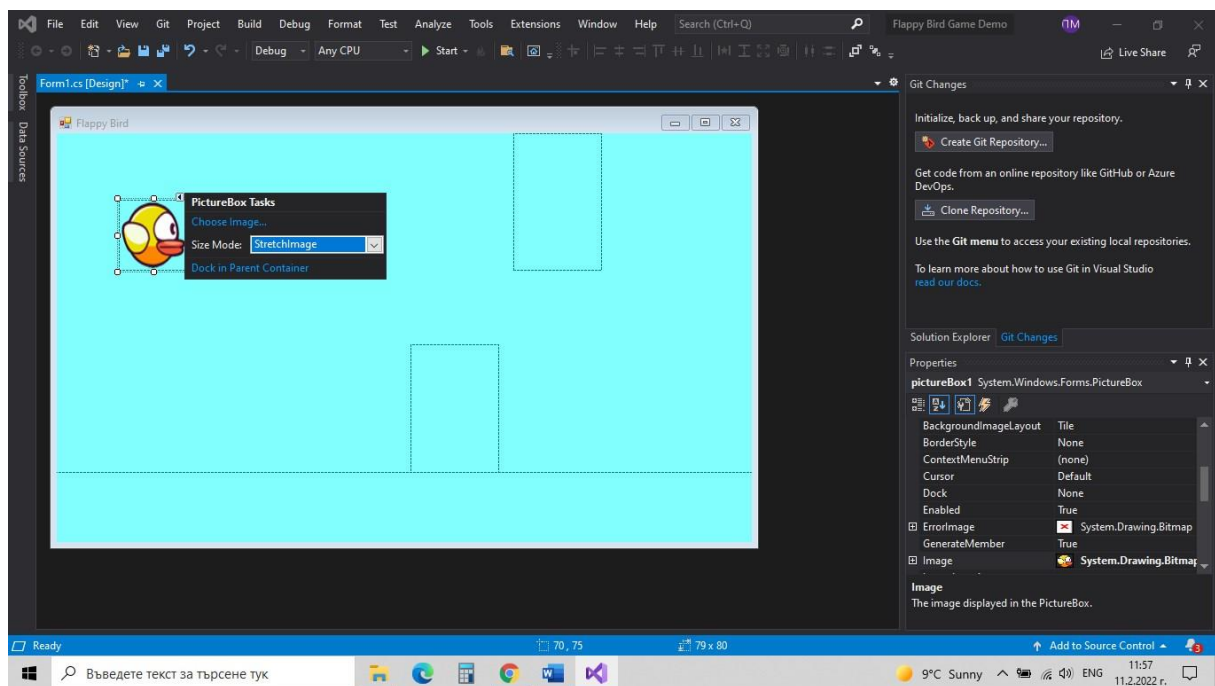
Configurações da pictureBox1:

- Escolher imagem -> Recurso Local -> Importar...



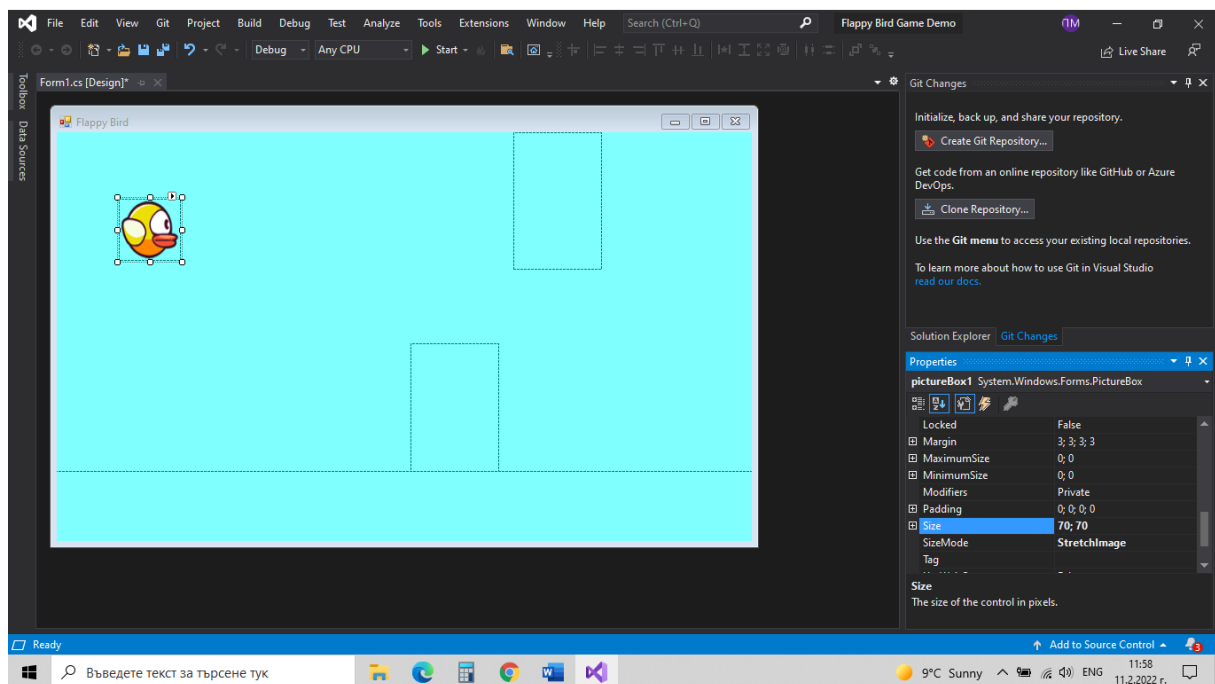


- Modo Tamanho: Imagem esticada

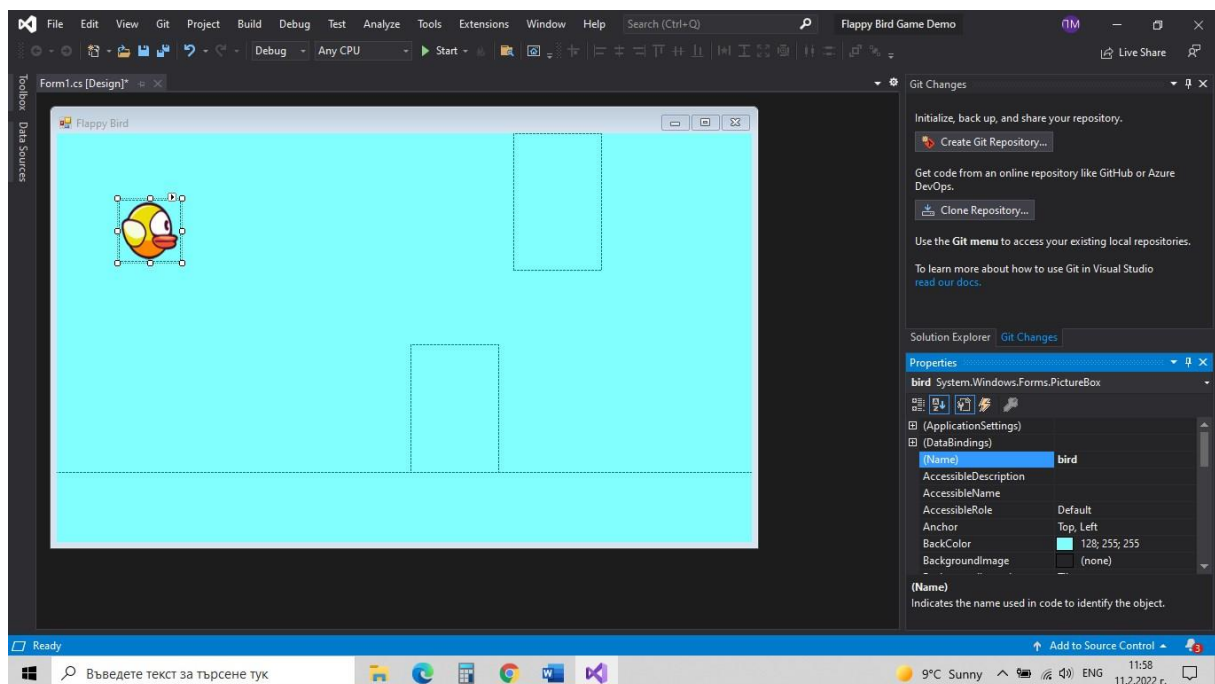




- Tamanho: 70;70



- (Nome): pássaro



Faça o mesmo cenário para as outras imagens:

Definições da pictureBox2:

- (Nome): pipeUp
- Modo Tamanho: Imagem esticada
- Tamanho: 90;140

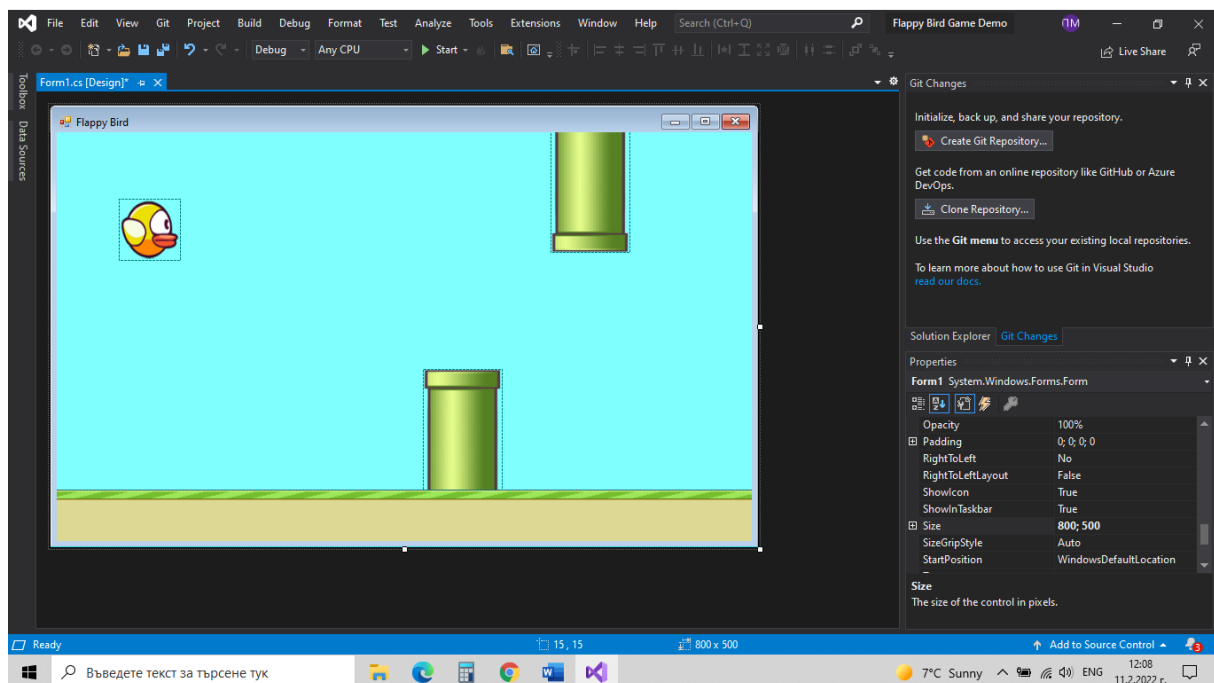


### Definições da pictureBox3:

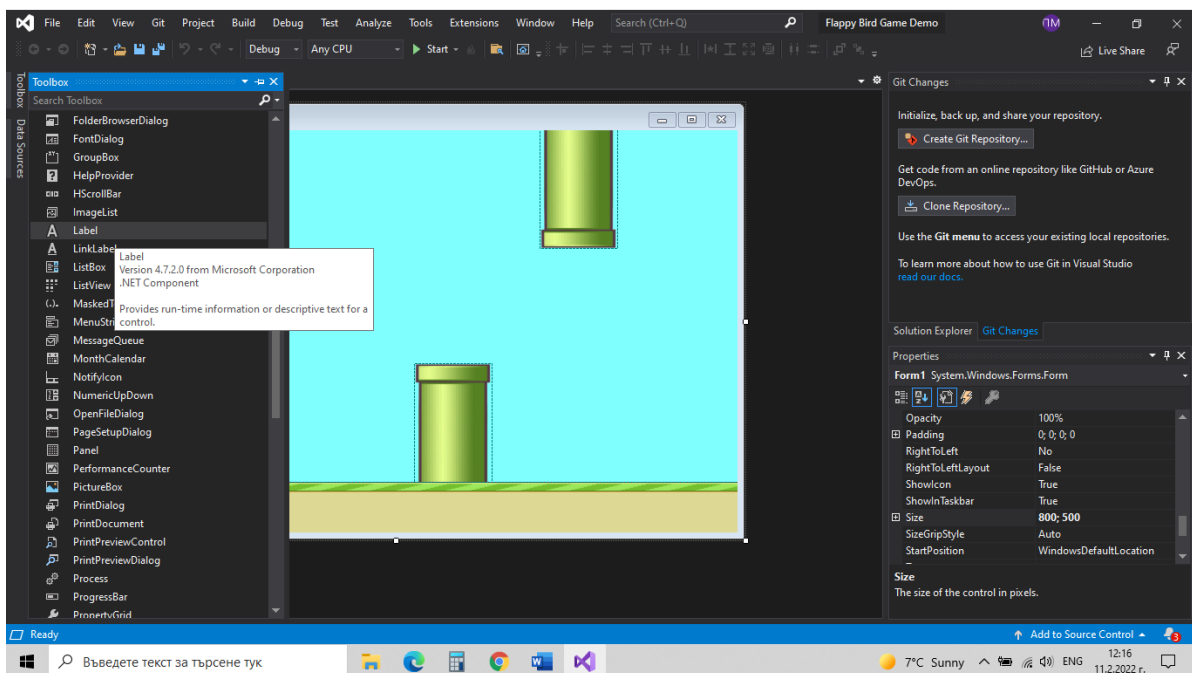
- (Nome): pipeDown
- Modo Tamanho: Imagem esticada
- Tamanho: 90;140

### Definições da pictureBox4:

- (Nome): ground
- Modo Tamanho: Imagem esticada
- Tamanho: 800;60

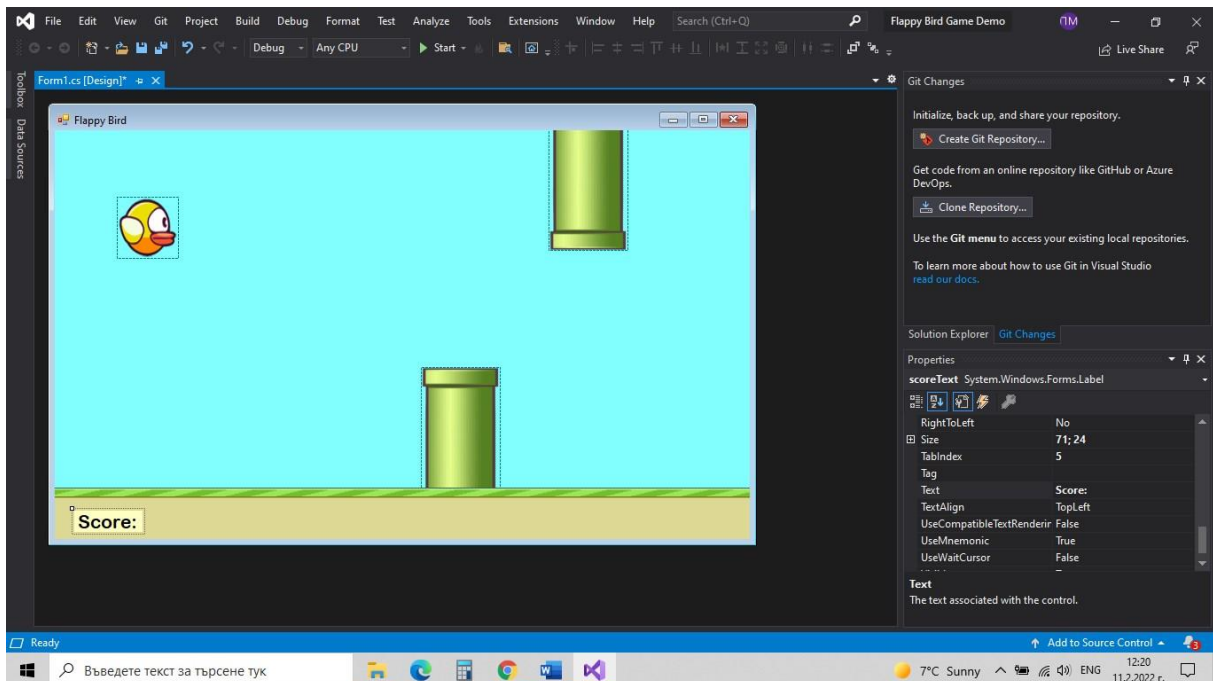


## 4º Passo: Acrescentar e definir uma etiqueta de objeto

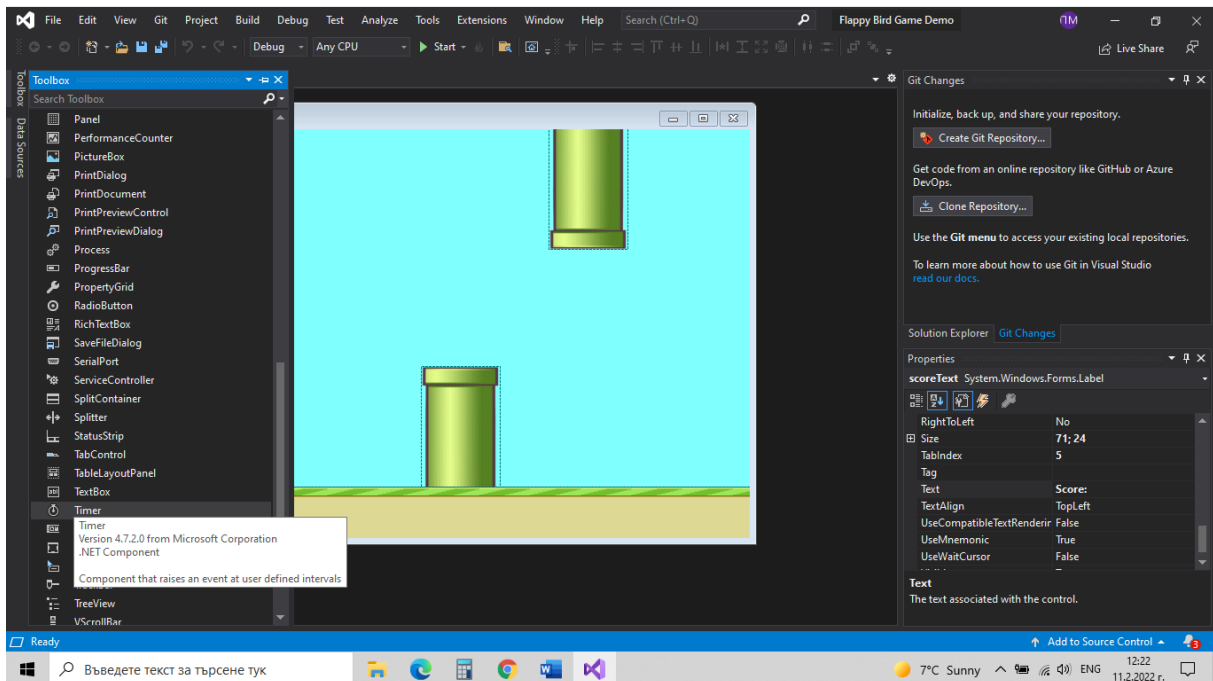


Definir a legenda 1:

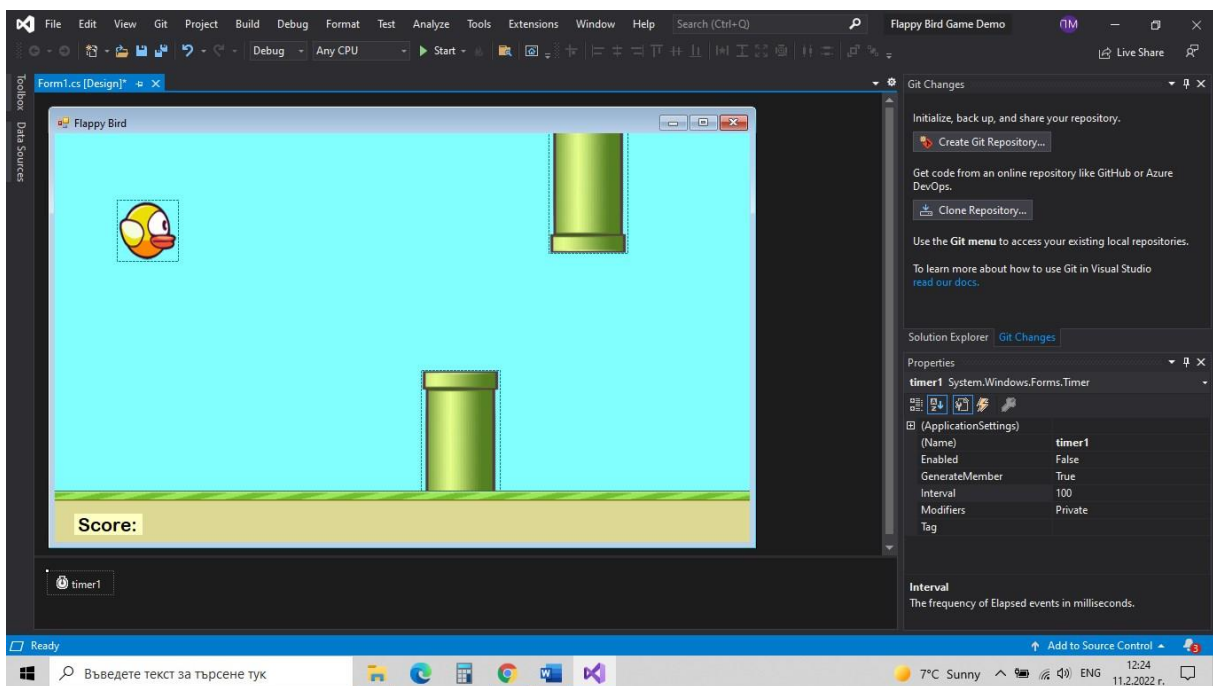
- (Nome): scoreText
- BackColor: similar to the ground
- Fonte: Arial; 16
- Texto: Pontuação



## 5º Passo: juntar um cronómetro



Isto cria um elemento que é invisível no campo de jogo.



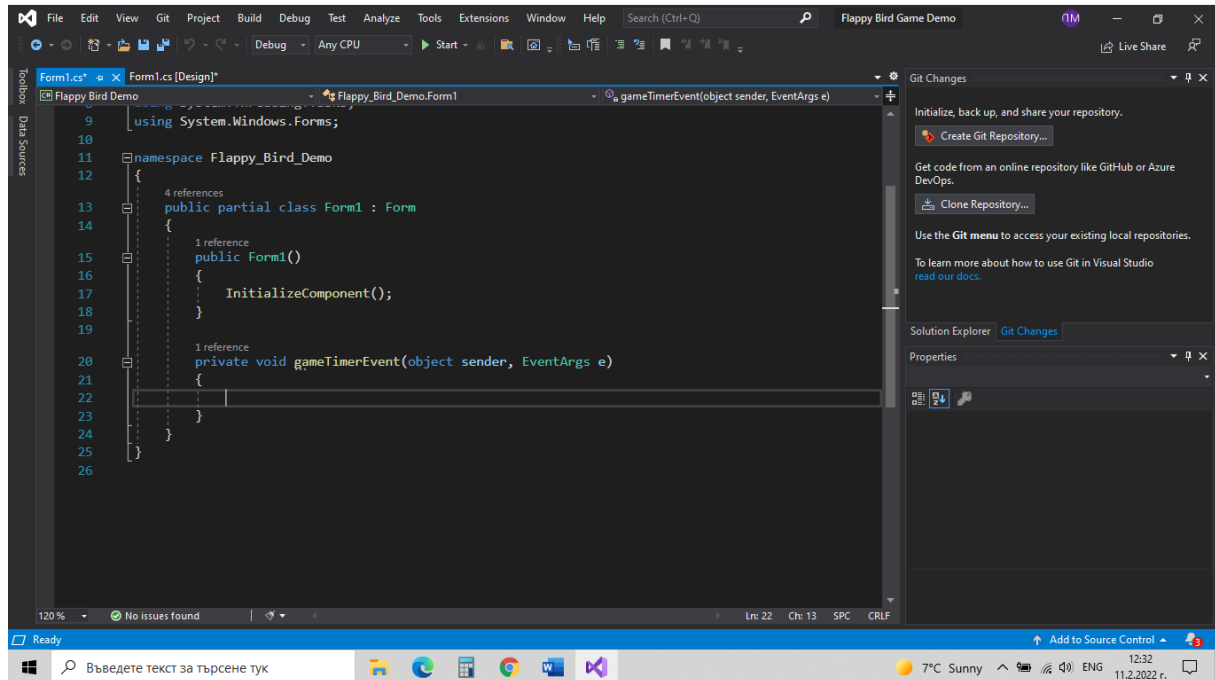
Ajuste do temporizador1:

- (Nome): gameTimer - Ativado: true
- Intervalo: 20

E nós criamos um evento deste elemento:

- Tick: gameTimerEvent

Quando clicamos neste evento, podemos ver o código do mesmo. Este evento cria um método:

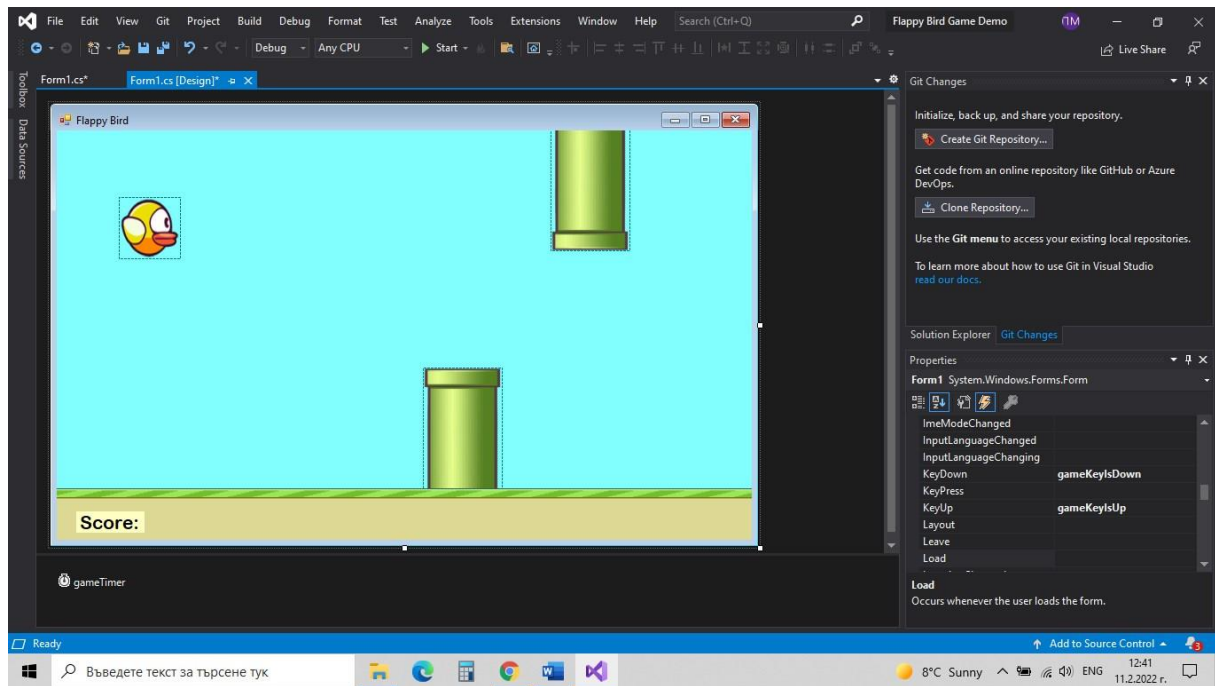


Neste método, iremos escrever código para descrever ações que acontecem a toda a hora.

## 6 Passo: fixando dois eventos no formulário principal Form1

O jogo será jogado com teclas de teclado. Quando uma tecla é premida, o pássaro levanta-se. Quando uma tecla não é premida, o pássaro cairá. Portanto, precisamos de dois eventos aplicados à forma principal do jogo Campo:

- KeyDown: gameKeyIsDown
- KeyUp: gameKeyIsUp

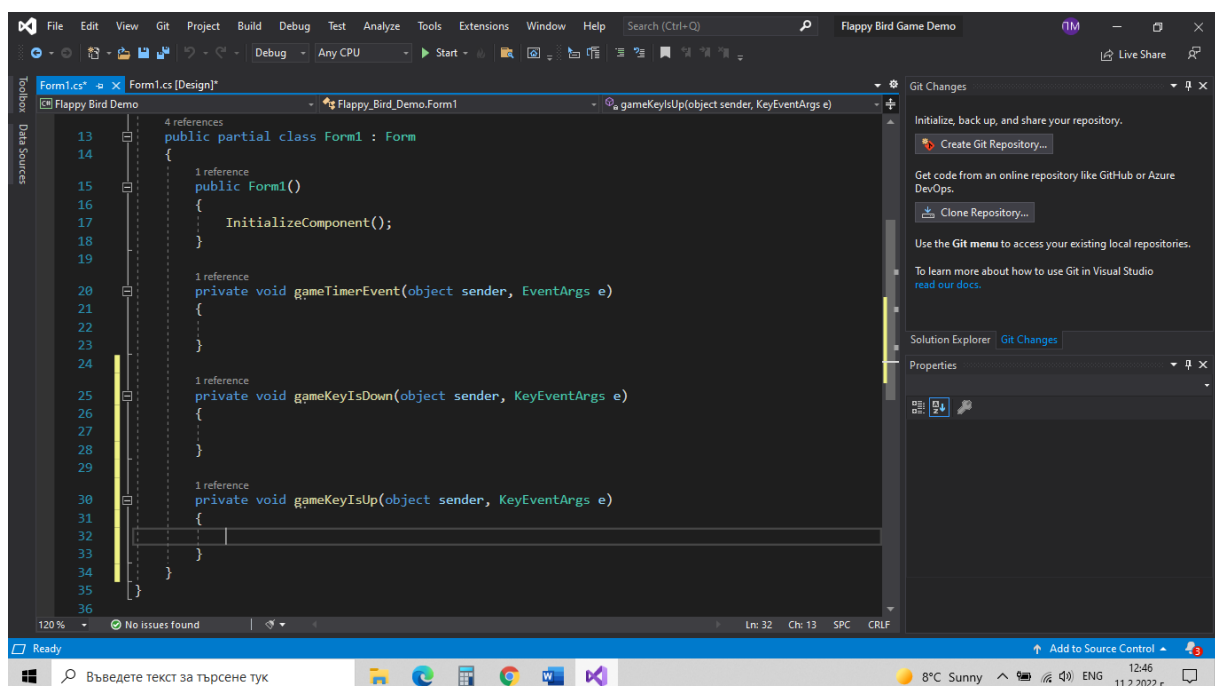


Ao nomear estes eventos criamo-los.

Então, o que é que temos agora no guião C#?

Criámos a classe Form1 e quatro métodos - Form1(), GameTimerEvent(), GameKeyIsDown(), GameKeyIsUp().

Aqui começamos a escrever o código do nosso jogo.



## 7º Passo: criar variáveis

No início precisamos de três variáveis - uma para a velocidade dos tubos, uma para a gravidade com que a ave irá descer e uma para a pontuação. Damos números inteiros dos valores iniciais às três variáveis, que podemos depois alterar se não forem apropriados. O resultado, claro, é 0 no início.

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

## 8º Passo: movendo o pássaro para baixo

O pássaro é um objeto. Podemos alcançar as propriedades e o método para cada objeto quando escrevemos o nome do objeto e do ponto. Por exemplo, "ave". Assim, podemos procurar as propriedades do elemento pássaro e escolher o que precisamos. Para a deslocação para baixo, precisamos da propriedade Topo. Aumentando o valor desta propriedade, vamos aumentar a distância entre o topo da imagem e o topo do campo e assim a ave vai mover-se para baixo. Temos uma gravidade variável e é com ela que vamos aumentar a propriedade. A ave está sempre a mover-se para baixo, pelo que temos de escrever este código no método GameTimerEvent().

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

## 9º Passo: movendo a ave para cima quando a tecla é premida e para baixo quando as teclas não são premidas

O pássaro irá subir se reduzirmos a sua propriedade Top. Isto pode ser feito através da alteração da gravidade para um número negativo. Este movimento deve ser realizado quando uma tecla é premida, pelo que o código será escrito no método gameKeyIsDown(). Para recomençar a descer quando a chave está em cima, precisamos de definir a gravidade para ser novamente um número positivo e isto deve ser escrito no método gameKeyIsUp().

```

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

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

```

### 10º Passo: mover os tubos para a direita

Os tubos são também objetos chamados pipeUp e pipeDown e têm também propriedades. Ao deslocarem-se para a esquerda, utilizaremos a sua propriedade Left. Esta propriedade define a distância do lado esquerdo da imagem para o lado esquerdo do campo. Se reduzirmos esta distância, os canos mover-se-ão para a esquerda. E, claro, escreveremos o código no método gameTimerEvent() porque este movimento acontece a toda a hora.

```

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

```

### 11º Passo: os tubos aparecem novamente

Agora os tubos movem-se para a esquerda e saem do campo de jogo. Para os fazer reaparecer à esquerda do ecrã, temos de verificar se saem pela esquerda. Se o seu lado esquerdo se tiver tornado inferior a 0, faremos um número maior. Utilizaremos um operador para condições lógicas, se escrito novamente no método gameTimerEvent().



```

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

```

## 12º Passo: terminar o jogo se houver uma colisão

Agora vamos criar a nossa própria função (método), na qual faremos apenas uma coisa - parar o temporizador. Para parar o temporizador, utilizaremos o método integrado Stop().

```

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

```

Nada está a acontecer neste momento, porque temos de chamar a este método quando a ave embate num cano ou no chão.

Utilizaremos o método incorporado para a verificação de colisão Bounds.IntersectsWith(). Este método dá resultados verdadeiros ou falsos para que possamos utilizá-lo como condição lógica num operador se.

Teremos três verificações de colisão - com uma tubagem, com a outra tubagem e com o solo. Se houver uma colisão, chamamos ao método endGame().

```

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

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

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

```

### 13º Passo: Contabilizar os pontos

Contaremos um ponto se algum dos tubos tiver saído pela esquerda. Já temos um tal cheque; resta-nos escrever nele que a pontuação aumentará em 1.

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

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

Resta tornar a pontuação visível.

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