PROGRAMA MODIFICADO
Day at Races

Aluno: Wyctor Fogos da Rocha

Disciplina: Programação Orientada a Objeto

Professor: Danilo Cesar Azeredo Silva

TURMA: IEE7-2020/1

Principais modificações feitas:

- Obstáculo (que pode ser inserido na pista clicando no CheckBox);
- Animais pulam pra desviar do obstáculo;
- Foi Inserido um áudio (efeito 3D que pode ser melhor escutado com o uso de um HeadPhone) de corrida;
- Barra de progresso da corrida;
- Mensagem de qual animal ganhou no final;
- Mensagem perguntando ao usuário se o mesmo deseja repetir a corrida;
- Botão que direciona o usuário a página do GitHub para que o mesmo baixa os arquivos do jogo, se assim desejar.
- Imagem de ícone;

Diagrama de classes

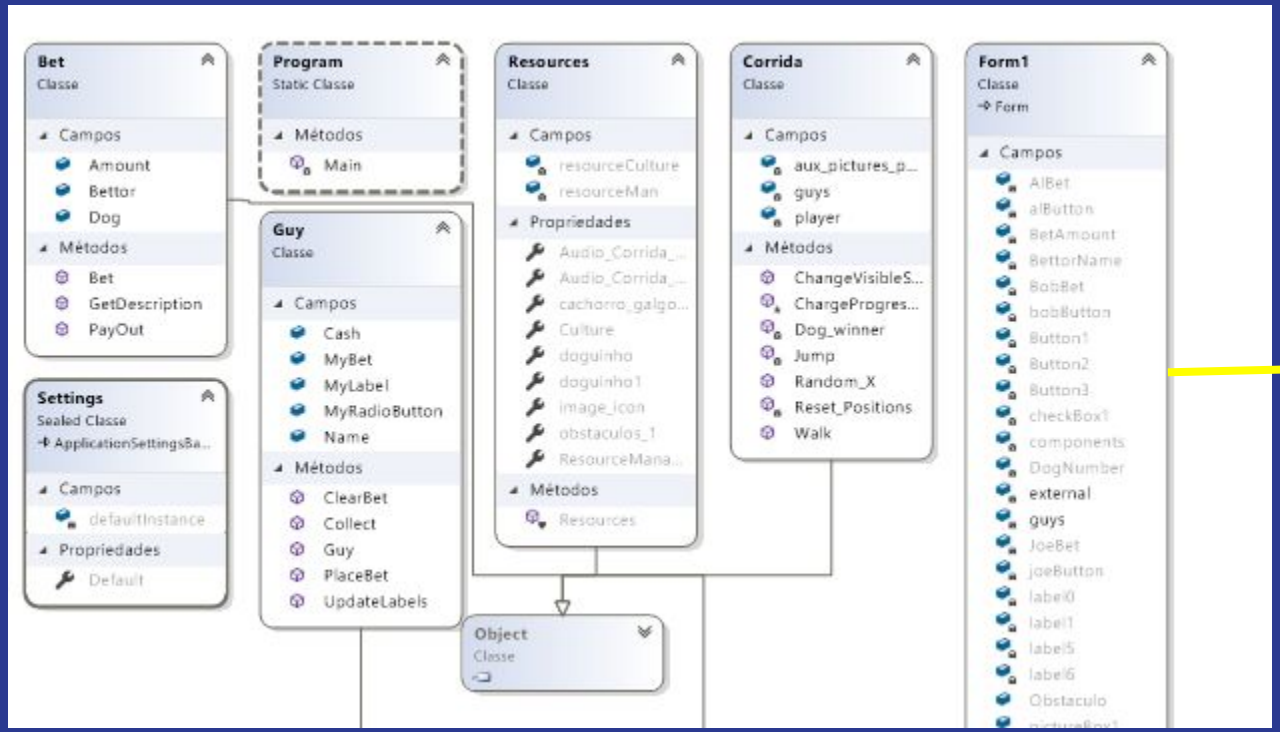
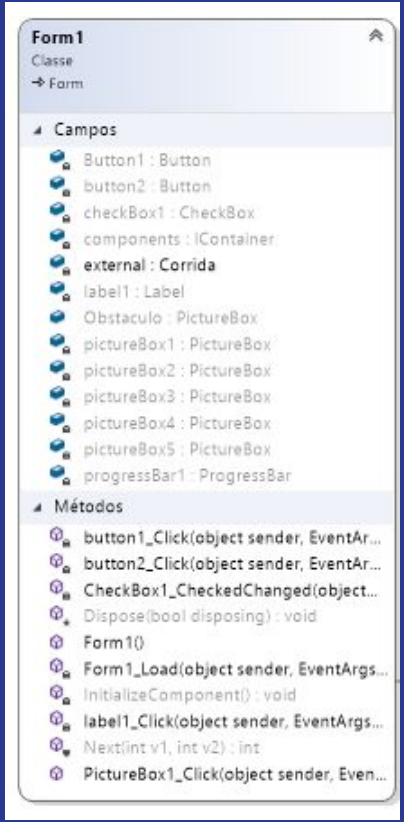


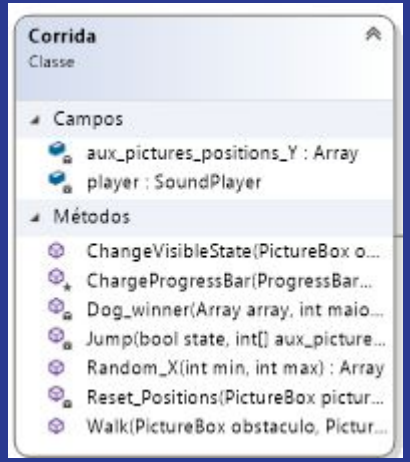
Diagrama de classes



Na parte dos campos, temos um objeto principal, ‘external’ que acessa os métodos da classe ‘Corrida’ em outro arquivo.

Pode se ver que há também métodos na classe ’Form1’, entretanto apenas 4 são utilizados, sendo eles: *Form1, button1_Click,button2_Click e CheckBox1_CheckedChanged*.

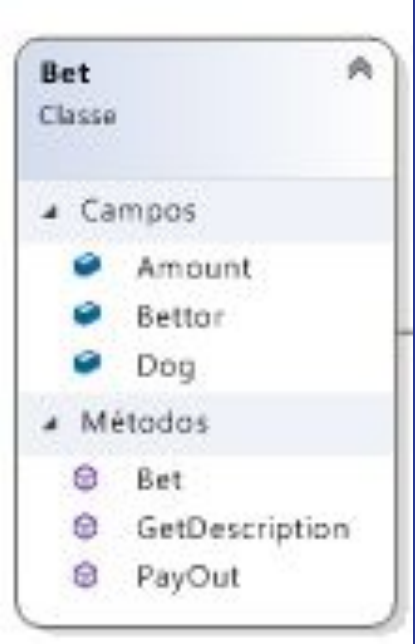
Diagrama de classes



Na classe Corrida, na parte dos campos, temos um objeto que toca o áudio da corrida e um array com as coordenadas iniciais das imagens para o caso do usuário queira repetir a corrida.

Nesta classe são utilizados 7 métodos, sendo eles: *ChangeVisibleState, ChargeProgressBar, Dog_winner, Jump, Random_X, Reset_Positions e Walk.*

Diagrama de classes



Na classe Bet, na parte dos campos, temos os campos com as características das apostas.

Nesta classe são utilizados 3 métodos, sendo eles: *Bet, GetDescription e Payout.*

Diagrama de classes

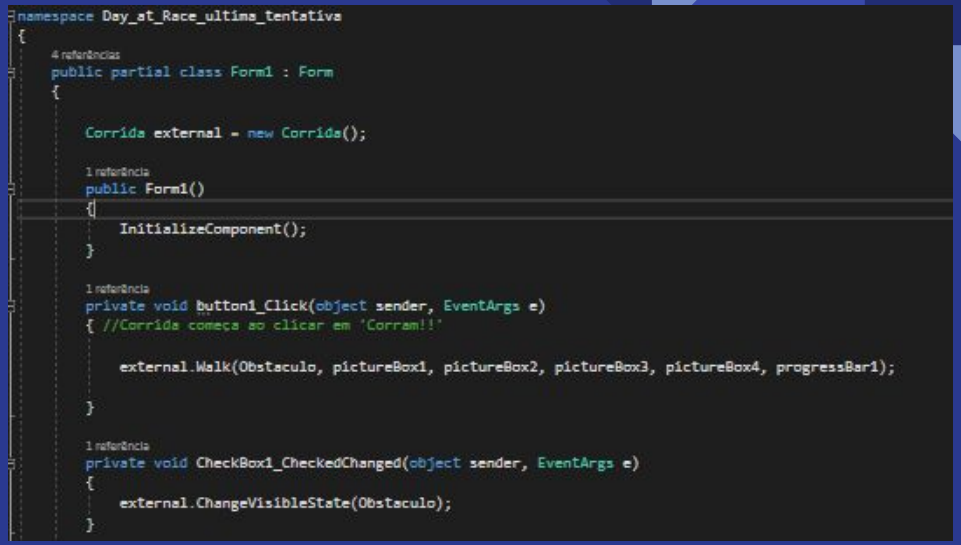
Na classe Guy, na parte dos campos, temos os campos com as características da pessoa que fará a aposta.

Nesta classe são utilizados 3 métodos, sendo eles: Collect, Guy, Place*Bet, UpdateLabels e ClearBet.*

Código

Com o código iniciado, é criado um objeto para acessar (o *external*) os métodos na classe *Corrida*.

Após escolher com os sem obstáculo, ou seja, mudar a visibilidade da imagem ‘obstáculo’, o usuário clica no botão ‘Corram!’ que chama o método ‘Walk()’ na classe Corrida por meio do objeto external.



```
namespace Day_at_Race_ultima_tentativa
{
    4 referências
    public partial class Form1 : Form
    {

        Corrida external = new Corrida();

        1 referência
        public Form1()
        {
            InitializeComponent();
        }

        1 referência
        private void button1_Click(object sender, EventArgs e)
        { //Corrida começa ao clicar em 'Corram!!'

            external.Walk(Obstaculo, pictureBox1, pictureBox2, pictureBox3, pictureBox4, progressBar1);

        }

        1 referência
        private void CheckBox1_CheckedChanged(object sender, EventArgs e)
        {
            external.ChangeVisibleState(Obstaculo);
        }
```

Código

Com Walk chamado, suas variáveis são declaradas, inclusive as coordenadas iniciais das imagens, para o caso do usuário querer repetir no jogo.

```
public void Walk(PictureBox obstaculo, PictureBox pictureBox1, PictureBox pictureBox2, PictureBox pictureBox3, PictureBox pictureBox4, ProgressBar progressBar1)
{
    //Começa a tocar a música de fundo
    player.SoundLocation = Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location) + "\\Audio_Corrida_de_cavalo.wav";

    //Atualiza o valor da posição ('X') de cada imagem
    int maior_dist = 0;
    int[] pictures_positions_X;
    pictures_positions_X = new int[4];

    //Faixa de valores a serem escolhidos para fazer os cachorros correrem
    int X_max = 110;
    int X_min = 0;

    //Visibilidade do obstáculo
    bool state;

    // Obtem o valor a ser acrescido em 'X'
    int[] aux_array;
    aux_array = new int[4];

    // Obtem o valor a ser acrescido em 'Y' se precisarem pular
    int[] aux_pictures_positions_Y;
    aux_pictures_positions_Y = new int[4];

    // Caso o usuário desejar repetir no final do programa
    char repeat = 's';

    // Enquanto não chegarem na linha de chegada, continuam correndo
    while (repeat== 's')
    {
        //As posições iniciais de cada imagem são salvas
        int[,] posicoes_iniciais;
        posicoes_iniciais = new int[4, 2] { { 89, 145 }, { 89, 220 }, { 89, 294 }, { 89, 372 } };
        player.Play();
```

Código

Com corrida iniciada, são utilizado um array auxiliar para obter o novo valor da coordenada de ‘x’ que será o valor anterior mais um valor entregue pelo método *Random_X*. Após isso, é guardado o maior valor, que serve de parâmetro para saber quando a corrida acaba.

Assim as imagens vão para frente.

```
player.Play();
while (maior_dist <= 750)
{

    pictures_positions_X[0] = pictureBox1.Location.X + Convert.ToInt32(Random_X(X_min, X_max).GetValue(0));
    pictures_positions_X[1] = pictureBox2.Location.X + Convert.ToInt32(Random_X(X_min, X_max).GetValue(1));
    pictures_positions_X[2] = pictureBox3.Location.X + Convert.ToInt32(Random_X(X_min, X_max).GetValue(2));
    pictures_positions_X[3] = pictureBox4.Location.X + Convert.ToInt32(Random_X(X_min, X_max).GetValue(3));

    //Verifacação de quem está mais na frente
    maior_dist = pictures_positions_X[0];
    for (int i = 0; i < pictures_positions_X.Length; i++)
    {
        if (pictures_positions_X[i] > maior_dist)
        {
            maior_dist = pictures_positions_X[i];
        }
    }
```

Código

Caso o usuário tenha marcado a opção com obstáculo, o método *Jump* irá enviar os novos valores da coordenada ‘y’ de cada imagem, gerando o pulo.

Assim os animais correm até alcançarem a chegada.

Lembrando que o progresso da corrida é atualizada a barra de progresso.

```
    //Setando os novos valores das coordenadas
    // Caso haja obstáculo na pista de corrida, os cães pulam

    state = obstaculo.Visible;

    this.aux_pictures_positions_Y = Jump(state, aux_pictures_positions_Y, pictures_positions_X);

    //Mudando_posicao()
    pictureBox1.Location = new Point(pictures_positions_X[0], pictureBox1.Location.Y + aux_pictures_positions_Y[0]);
    pictureBox2.Location = new Point(pictures_positions_X[1], pictureBox2.Location.Y + aux_pictures_positions_Y[1]);
    pictureBox3.Location = new Point(pictures_positions_X[2], pictureBox3.Location.Y + aux_pictures_positions_Y[2]);
    pictureBox4.Location = new Point(pictures_positions_X[3], pictureBox4.Location.Y + aux_pictures_positions_Y[3]);

    //Carrega o progresso da corrida na Barra de progresso
    ChargeProgressBar(progressBar1, maior_dist);

    //Delay no tempo do loop
    int milliseconds = 750;
    System.Threading.Thread.Sleep(milliseconds);
}


ChargeProgressBar(progressBar1, maior_dist);
```

Código

É informado o animal vencedor e se o usuário deseja jogar novamente. Se sim, o método *Reset_Positions*, reseta as coordenadas e todas as variáveis necessárias.

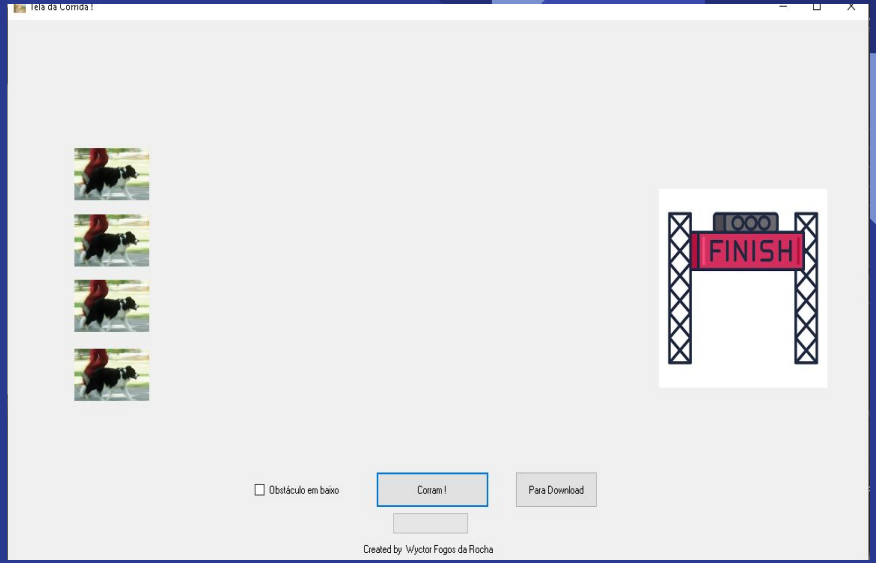
```
    //Após o final da corrida, é analisado o que
    Dog_winner(pictures_positions_X, maior_dist);

    if (MessageBox.Show("Deseja jogar novamente?", "Pergunta", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==DialogResult.Yes)
    {
        repeat = 's';
        Reset_Positions(pictureBox1,pictureBox2,pictureBox3,pictureBox4,posicoes_inicias,progressBar1, maior_dist);
    }
    repeat = 'n';
}
player.Stop();
```

Funcionamento

Ao lado temos a interface do usuário, dando-o opção de obstáculo ou não na corrida. Caso o usuário escolha com obstáculo, os animais irão pular.

No final será perguntado se o usuário irá querer jogar novamente.

Há um botão que leva o usuário direto ao GitHub, caso queira baixar o código.



Obrigado !

Wyctor Fogos da Rocha

wyctor.fogos@gmail.com / wyctor.fogos@hotmail.com

Link do jogo no GitHub:

https://github.com/wyctorfogos/Day_at_Races_Modificado_Wyctor_Fogos.git