Universidade Vila Velha

Programação Orientada a Objeto II

Jogo Endless Runner

Guilherme Ferreira de Góes - 202305386

1. CRIAÇÃO DA TELA INICIAL

1.1. TELA PRINCIPAL

Um menu inicial foi criado para que o jogador possa escolher qual modo de jogo ele pretende jogar.

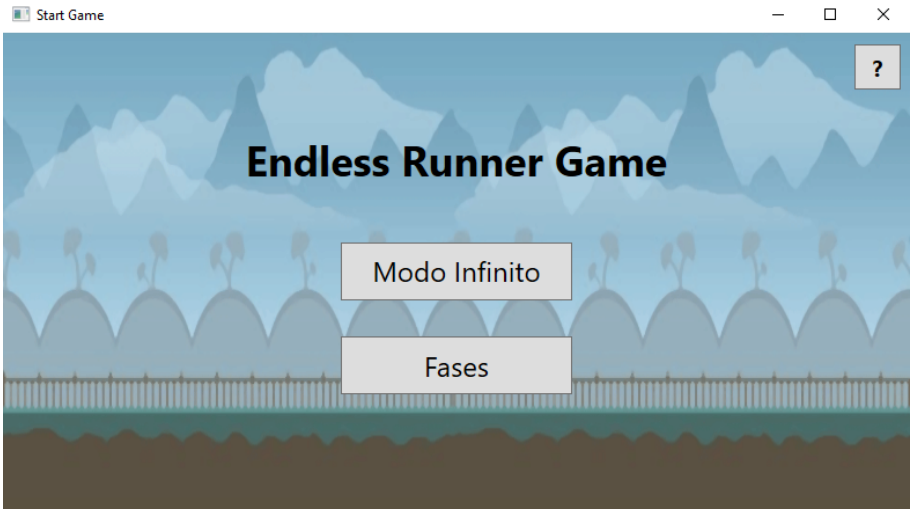

Figura 1 – Menu inicial

1.2. TELA DE AJUDA

Caso o jogador tenha alguma dúvida de como os modos de jogo funcionam ele pode clicar na "?"no canto superior direito para abrir uma tela de tutorial.

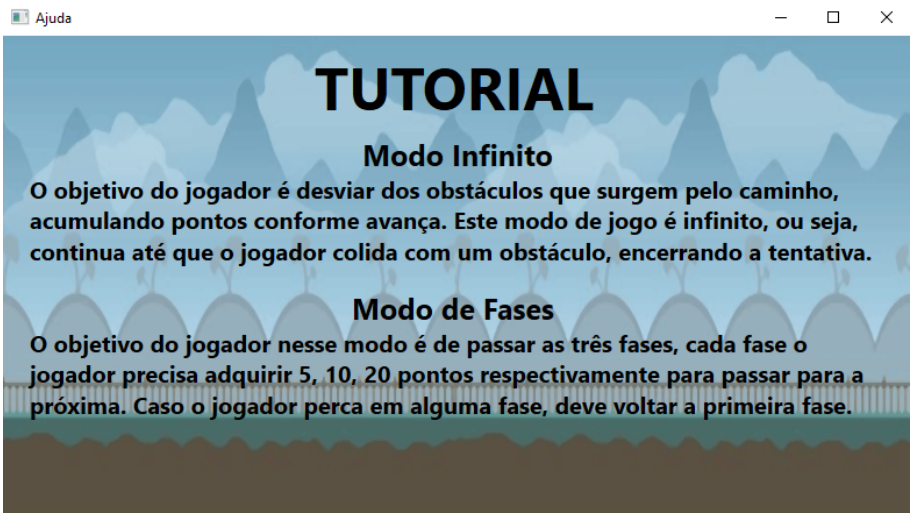

Figura 2 – Tela de Ajuda

No botão de ajuda, após a tela ser instanciada, ela é exibida utilizando o ShowDialog(); dessa forma o jogador terá que fecha-la para iniciar algum modo de jogo.

```
1 referência
private void button_Click(object sender, RoutedEventArgs e)
{
    Ajuda tutorialModos = new Ajuda();
    tutorialModos.ShowDialog();

}
```

Figura 3 – Botão de ajuda

2. CRIAÇÃO DE MODOS DE JOGO

2.1. MODO INFINITO

Esse modo é parecido com o jogo inicial, ele apenas chegará ao fim quando o jogador colidir com algum obstaculo. A verificação de que o jogador colidiu com algum obstáculo é feita da seguinte forma:

```
if (playerHitBox.IntersectsWith(obstacleHitBox))
{
    gameOver = true;
    gameTimer.Stop();
}

if (gameOver)
{
    obstacle.Stroke = Brushes.Black;
    obstacle.StrokeThickness = 1;

    player.Stroke = Brushes.Red;
    player.StrokeThickness = 1;

    player.Height = 99;
    player.Width = 67;
    scoreText.Content = $"Score: {score} Pressione Enter para jogar novamente!!";
}
```

Figura 4 – Colisão + GameOver

Caso seja detectado que o jogador colidiu com o obstáculo, a variável gameOver do tipo bool é atribuida com true e o jogo vai parar. Após isso a borda do jogador ficará vermelha e a borda do obstáculo preta, e será exebida a mensagem de derrota.

2.2. MODO DE FASES

Nesse modo o jogador terá que vencer 3 fases em sequência, precisando de 5, 10, 20 pontos respectivamente para vencer cada uma delas.

2.2.1. Verificação de Fase

A verificação de fase é feita pelo metódo ConferirFase(); é utilizado a variável faseAtual para verificar em qual fase o jogador está atualmente, quando ele conseguir os pontos necessário para a proxima fase faseCompletada será alterada para true.

```
1 referência
private void ConferirFase()
{
    if (faseAtual == 1 && score >= 5)
    {
        gameTimer.Stop();
        faseCompletada = true;
        scoreTextF.Content = $"Fase 1 completa! Enter para proxima fase!";
        score = 0;
    }
    else if (faseAtual == 2 && score >= 10)
    {
        gameTimer.Stop();
        faseCompletada = true;
        scoreTextF.Content = $"Fase 2 completa! Enter para para proxima fase!";
        score = 0;
    }
    else if (faseAtual == 3 && score >= 20)
    {
        gameWon = true;
        gameTimer.Stop();
        scoreTextF.Content = $"Você venceu! Pressione Enter para jogar novamente!!";
    }
}
```

Figura 5 – ConferirFase();

No momento que o jogador pressionar Enter para continuar, faseCompletada voltará a ser false, o jogo vai se iniciar e faseAtual será somado em 1.

```
if (e.Key == Key.Enter && faseCompletada)
{
    faseCompletada = false;
    gameTimer.Start();
    faseAtual += 1;
}
```

Figura 6 – alteração do faseCompletada

3. CRIAÇÃO DE NOVO UM OBSTÁCULO

3.1. OBSTÁCULO SUPERIOR

Um novo tipo de obstáculo foi criado para gerar uma maior diversidade na jogabilidade. Esse obstáculo o jogador deverá deslizar por baixo dele para conseguir passar.

.

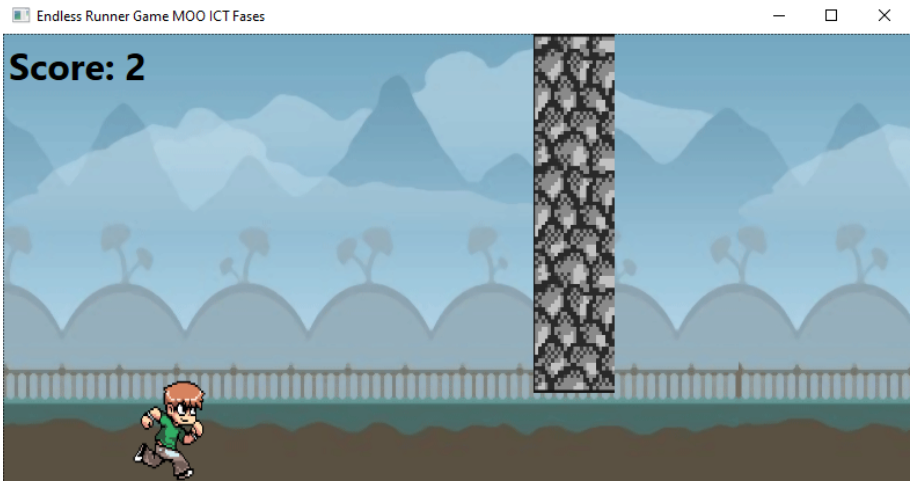

Figura 7 – Novo Obstáculo Adicionado

Com isso, a lógica de criação dos obstáculos foi alterada para gerar o novo obstáculo criado:

```
if (Canvas.GetLeft(obstacle) < -50)
{
    if (proxObstaculo == "normal")
    {
        obstacleSprite.ImageSource = new BitmapImage(new Uri("pack://application:,,,/images/obstacle.png"));
        obstacle.Height = 178;
        obstacle.Width = 50;
        Canvas.SetTop(obstacle, 310);
        proxObstaculo = "top";
    }
    else
    {
        obstacleSprite.ImageSource = new BitmapImage(new Uri("pack://application:,,,/images/obstacle2.png"));
        obstacle.Height = 310;
        obstacle.Width = 70;
        Canvas.SetTop(obstacle, 0);
        proxObstaculo = "normal";
    }

    Canvas.SetLeft(obstacle, 950);
    score += 1;
}
```

Figura 8 – Lógica de criação dos Obstáculos

A nova lógica é que os obstáculos serão gerados de forma alterada, vai ser utilizado o proxObstaculo para saber se é o "normal"(placa no chão) ou se é o "top"(obstaculo no topo da tela). Assim a altura, largura e textura do obstáculo será alterada e a sua posição também.

4. CRIAÇÃO DE NOVAS MECÂNICAS

4.1. DESLIZAR

Com a criação do novo obstáculo, foi necessário criar a mecânica de "deslizar"para passar por baixo do obstáculo. ela foi feita da seguinte maneira:

```
if (e.Key == Key.Down && !deslizando && jumping == false)
{
    deslizando = true;
    playerSprite.ImageSource = new BitmapImage(new Uri("pack://application:,,,/images/newRunner_12.gif"));
    player.Height = 67;
    player.Width = 99;
    Canvas.SetTop(player, Canvas.GetTop(player) + (99 - 67));
}
```

Figura 9 – Mecânica de Deslizar

Quando o jogador pressionar a seta para baixo no teclado, será feita uma verificação para garantir que ele nao esteja pulando enquanto faz essa nova ação. Além disso, vai ocorrer a aplicação do novo sprite durante o movimento e a alteração da altura e largura para que nao ocorra a colisão com a hitbox do obstaculo.

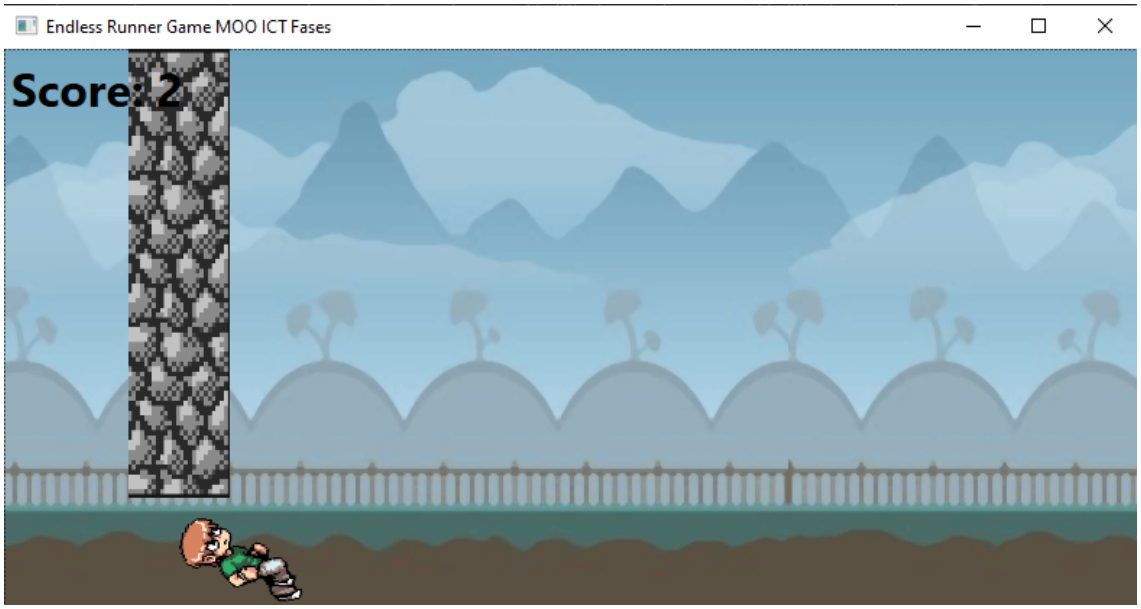

Figura 10 – Mecânica de Deslizar

4.2. PAUSE

Também foi implementada a função de "pausar"o jogo, quando o jogador pressionar a tecla "P"ocorrerá a pausa, e quando for pressionada novamente o jogo voltrá a prosseguir.

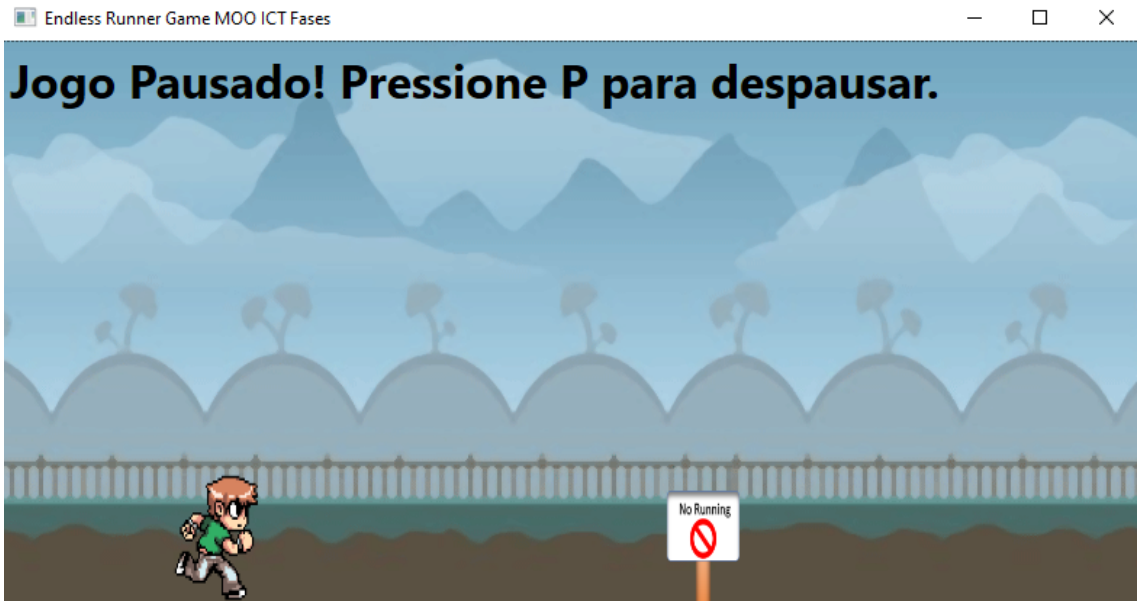

Figura 11 – Mecânica de Pausar