Jogo do Punch-Out

Matheus Endlich Silveira - 202305392

20 de outubro de 2024

Matheus Endlich Silveira - 202305392

Sumário

1 Introdução

O jogo é inspirado no clássico *Punch-Out* do Nintendinho. No entanto, devido às limitações de código e do projeto, ele apresenta apenas um inimigo, que se repete ao final de cada combate. As teclas do teclado são essenciais para a jogabilidade.

2 Problemas Iniciais

- Ao apertar o botão de soco, ele acerta múltiplos golpes de forma contínua, causando danos até o fim do combate.

- Apenas 1 inimigo disponível, com código mal organizado e difícil de adaptar para adicionar novos inimigos.

- Ausência de um juiz para julgar o combate, que poderia passar uma sensação de luta mais realista, como no jogo original.

- Não há tela de início ou *game over*, dificultando a definição do progresso do jogador.

- Não há benefício em continuar jogando, resultando em um ciclo infinito de lutas contra o mesmo inimigo.

- O inimigo recebe dano sem aparecer na tela, não demonstrando visivelmente que está presente.

- Não existiam funções básicas que abstraíssem ações importantes, como terminar o jogo ou carregar a página corretamente no início.

3 Mudanças e Novas Implementações

- Correção do botão de soco:

  - Foi criada uma variável booleana para controlar os ataques. Quando o botão de soco é pressionado, a variável se torna verdadeira, permitindo contabilizar o golpe apenas uma vez por clique. Isso impede que o soco cause dano continuamente.

```
if (attackPerformed == false)
{
    if (e.KeyCode == Keys.Left)
    {
        player.Image = Properties.Resources.boxer_left_punch;
        Player.PlayerBlock = false;
        if (player.Bounds.IntersectsWith(boxer.Bounds) && Enemy.PlayerBlock == false)
        {
            if (lutadorDaRodada == 0)
            {
                boxer.Image = Properties.Resources.enemy_hit;
                Enemy.PlayerHealth -= 5 + Player.DanoExtra;
            }
            else {
                boxer.Image = Properties.Resources.enemy2_hit;
                Enemy.PlayerHealth -= 5 + Player.DanoExtra;
            }
        }
        attackPerformed = true;
    }
```

Figura 1 – Exemplo da variável booleana.

- Adição de mais inimigos e reestruturação do código:

  – A lógica do código foi modificada para permitir a inclusão de múltiplos inimigos. Os parâmetros de HP do jogador e do inimigo foram movidos para classes apropriadas.

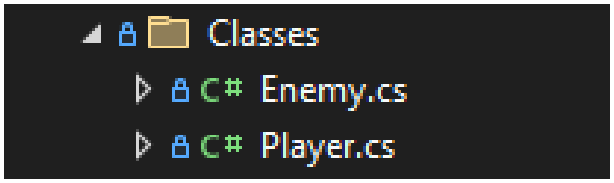

Figura 2 – Novas classes.

  – Agora, o jogo sorteia qual inimigo o jogador enfrentará, entre *Tough Rob* e *King Hippo*. A interface também foi ajustada para exibir o nome do inimigo corretamente.

Figura 3 – Nomes atualizados na interface.

  – As imagens dos inimigos foram adicionadas ao arquivo `resurse.resx`, incluindo sprites que mudam conforme o inimigo sofre dano ou morre.

Nome	Nome do Arquivo	Valor Neutro	Tipo
$this.BackgroundImage	Vitoria.resx	(Recurso Inserido)	Bitmap
background	Resources.resx	background.jpg	Bitmap
boxer-block	Resources.resx	boxer-block.png	Bitmap
boxer-left-punch	Resources.resx	boxer-left-punch.png	Bitmap
boxer-right-punch	Resources.resx	boxer-right-punch.png	Bitmap
boxer-stand	Resources.resx	boxer-stand.png	Bitmap
BoxerAttackTimer.TrayLocation	Form1.resx	Recurso Inserido (tipo: Point)	Point
BoxerMoveTimer.TrayLocation	Form1.resx	Recurso Inserido (tipo: Point)	Point
enemy-block	Resources.resx	enemy-block.png	Bitmap
enemy-death	Resources.resx	enemy-death.png	Bitmap
enemy-hit	Resources.resx	enemy-hit.png	Bitmap
enemy-punch1	Resources.resx	enemy-punch1.png	Bitmap
enemy-punch2	Resources.resx	enemy-punch2.png	Bitmap
enemy-stand	Resources.resx	enemy-stand.png	Bitmap
enemy2-block	Resources.resx	enemy2-block.png	Bitmap
enemy2-deach	Resources.resx	enemy2-deach.png	Bitmap
enemy2-hit	Resources.resx	enemy2-hit.png	Bitmap
enemy2-punch	Resources.resx	enemy2-punch1.png	Bitmap
enemy2-punch1	Resources.resx	enemy2-punch2.png	Bitmap
enemy2-stand	Resources.resx	enemy2-stand.png	Bitmap
Mario_Start	Resources.resx	mario_start.png	Bitmap
Mario_Win	Resources.resx	mario_end.png	Bitmap

Figura 4 – Sprites adicionados no `resurse.resx`.

– Condições `if` foram inseridas no código para definir quais sprites serão carregados durante o combate.

```
if (lutadorDaRodada == 0)
{
    musicPath = Path.Combine(Application.StartupPath, @"Musics\Battle.wav");
}
else {
    musicPath = Path.Combine(Application.StartupPath, @"Musics\Battle2.wav");
    boxer.Image = Properties.Resources.enemy2_stand;
    label2.Text = "king-hippo   ";
}
```

Figura 5 – Exemplo de condição `if`.

– Cada personagem tem sua própria música tema, melhorando a imersão do jogador.

Figura 6 – Novo inimigo implementado.

- Implementação de um juiz:

  – Um sprite do Mario foi colocado no canto da arena. Ele reage ao andamento do combate, comemorando e até anunciando o K.O. em voz alta.

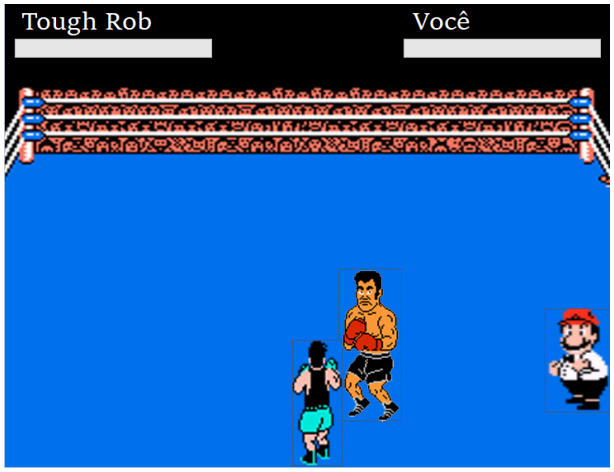

Figura 7 – Juiz no jogo.

- Tela de início e *game over*:
  - Uma tela de início foi adicionada, com a música clássica do jogo original. Também foram criadas telas de vitória e *game over*.

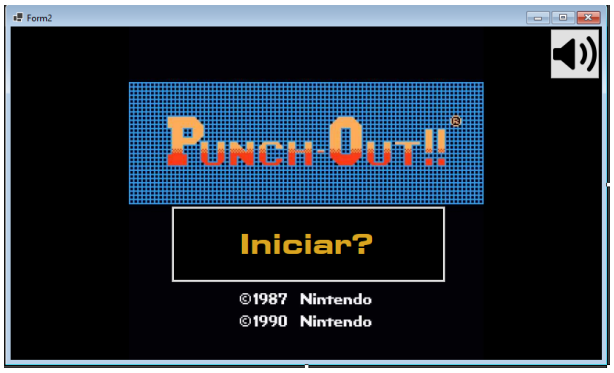

Figura 8 – Tela de introdução.

Figura 9 – Tela de vitória.

- Adição de uma academia:
  - Após cada luta, o jogador é levado para uma academia, onde pode gastar o dinheiro ganho para melhorar suas habilidades ou atrapalhar o inimigo.

Figura 10 – Academia implementada.

- Correção da exibição do inimigo na tela:
  - Agora, o inimigo aparece corretamente na tela e seu sprite muda conforme ele recebe dano ou morre.
- Funções de controle e abstração:
  - Foram adicionadas funções para controlar o carregamento do jogo e finalizar a partida, como a função `EndTheGame`.

```
1 referência
private void Form1_Load(object sender, EventArgs e)...
```

Figura 11 – Exemplo de função.