Pong

Vamos programar!!

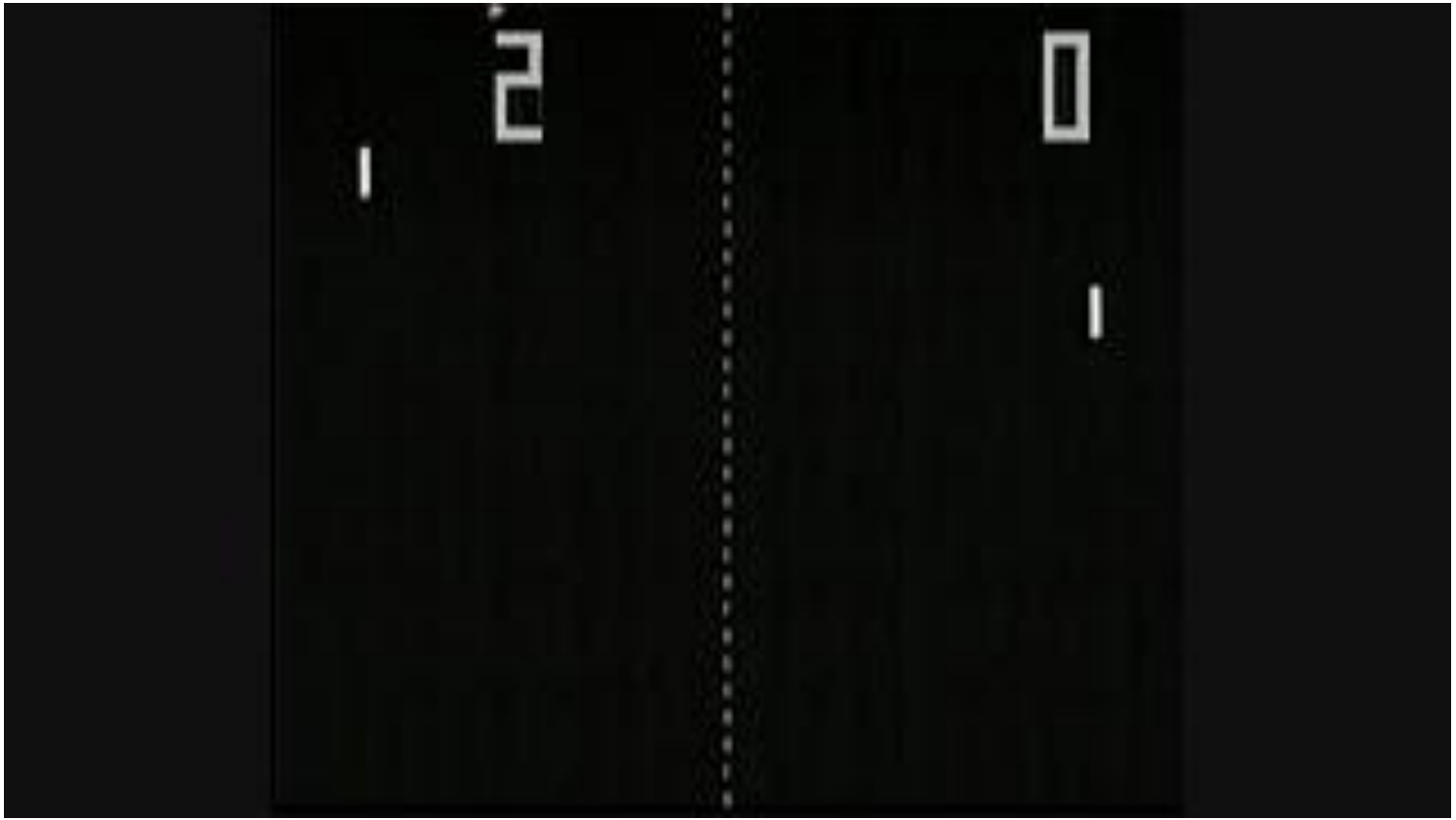
O que é o pong?

Um dos primeiros jogos eletrônicos desenvolvidos

lançado pela primeira vez em 1972 pela empresa americana Atari.

O jogo simula um jogo de tênis de mesa, em que dois jogadores controlam cada um uma raquete na tela, movendo-a para cima e para baixo, com o objetivo de rebater uma bola de um lado para o outro da tela, evitando que ela ultrapasse a linha de fundo.

O que é o pong?



O que é o pong?

O jogo foi um grande sucesso na época, e ajudou a popularizar a indústria de jogos eletrônicos.

É considerado um marco na história dos jogos eletrônicos, sendo um dos primeiros a utilizar gráficos simples e a incorporar sons digitais.

Hoje em dia, existem muitas variações e versões do jogo Pong, disponíveis em diferentes plataformas e dispositivos.

Regras

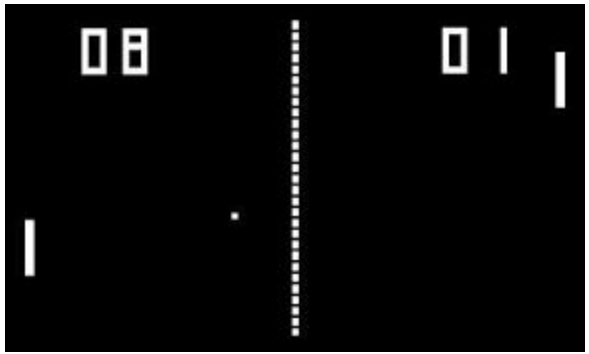
1. O jogo é jogado por dois jogadores, cada um controlando uma raquete no lado direito ou esquerdo da tela.
2. O objetivo do jogo é rebater a bola de um lado para o outro da tela, evitando que ela ultrapasse a linha de fundo da raquete do jogador.
3. Os jogadores ganham um ponto sempre que o adversário deixa a bola passar por ele e ultrapassa a linha de fundo.
4. O primeiro jogador a atingir um determinado número de pontos (geralmente 10 ou 11) é o vencedor.
5. A bola começa no centro da tela e é lançada por um dos jogadores. Ela se move em linha reta e rebate nas paredes e nas raquetes.
6. Os jogadores controlam a raquete movendo-a para cima e para baixo na tela, usando as teclas ou o mouse.
7. O jogador que toca na bola por último antes dela ultrapassar a linha de fundo do adversário ganha o ponto.
8. O jogo continua até que um jogador atinja o número de pontos necessários para ganhar. Se houver um empate, o jogo pode continuar até que um jogador ganhe com dois pontos de vantagem.

Para desenvolvimento do Pong, vamos utilizar a plataforma de desenvolvimento de jogos **Unity**

- Uma das plataformas mais populares para o desenvolvimento de jogos
- Permite criar jogos para uma ampla variedade de plataformas, incluindo PC, consoles, dispositivos móveis e realidade virtual.
- A plataforma oferece recursos avançados:
  - gráficos em 3D
  - física realista
  - inteligência artificial
  - Animação
  - Áudio
- Possui uma grande comunidade de desenvolvedores que compartilham conhecimento e recursos em fóruns e grupos online:
  - Uma ampla variedade de tutoriais
  - Documentação.

O quê mais vamos precisar?

- Raquetes
- Bola
- Limites do ambiente
- Pontuação



1 - Preparando o ambiente

Baixe as imagens que serão utilizadas no seu projeto: Disponivel no moodle.

No **Unity**, crie um novo projeto (template: 2D).

Dê um nome para o seu projeto (ex: Pong)

Adicione as imagens na pasta **assets**.

Na aba **hierarchy**, clique em **main camera** e altere o parâmetro **size** para 3.

Preparando as palhetas

- Arraste a imagem da raquete para a cena
- Em **Inspector**, selecione tag de **player**.
- Determine a posição xyz para (-4,0,0) e escala para (0.5,1.5,1)
- Click em **Add Component**, em seguida em **Physics 2D** e adicione:
  - **Box collider 2D** -> Permite que a bola tenha colisão com a raquete.
  - **Rigidbody 2D** -> Permite que o jogador movimente a raquete.
- No componente **Rigidbody 2D**, click em **body type** e selecione **kinematic.**
- Em seguida vamos adicionar um script para mover nossa raquete.
  - Click em **Add Component**
  - em seguida em **New Script**,
  - determine um nome de **PlayerControls** e
  - click em **Create and Add**.

Vamos editar nosso script de controle da raquete

```
public KeyCode moveUp = KeyCode.W;       // Move a raquete para cima
public KeyCode moveDown = KeyCode.S;     // Move a raquete para baixo
public float speed = 10.0f;              // Define a velocidade da raquete
public float boundY = 2.25f;             // Define os limites em Y
private Rigidbody2D rb2d;                // Define o corpo rigido 2D que representa a raquete
```

Vamos editar nosso script de controle da raquete

```
void Start () {
    rb2d = GetComponent<Rigidbody2D>();     // Inicializa a raquete
}
```

Vamos editar nosso script de controle da raquete

```
void Update () {

    var vel = rb2d.velocity;                // Acessa a velocidade da raquete
    if (Input.GetKey(moveUp)) {             // Velocidade da Raquete para ir para cima
        vel.y = speed;
    }
    else if (Input.GetKey(moveDown)) {      // Velocidade da Raquete para ir para cima
        vel.y = -speed;
    }
    else {
        vel.y = 0;                          // Velociade para manter a raquete parada
    }
    rb2d.velocity = vel;                    // Atualizada a velocidade da raquete

    var pos = transform.position;           // Acessa a Posição da raquete
    if (pos.y > boundY) {
        pos.y = boundY;                     // Corrige a posicao da raquete caso ele ultrapasse o limite superior
    }
    else if (pos.y < -boundY) {
        pos.y = -boundY;                    // Corrige a posicao da raquete caso ele ultrapasse o limite inferior
    }
    transform.position = pos;               // Atualiza a posição da raquete

}
```

Criando a segunda raquete

- Clique com o botão direito na primeira raquete na aba **Hierarchy**, em seguida selecione **Duplicate**.
- Renomeie sua segunda raquete.
- Determine a posição xyz para (4,0,0)
- Altere os controles para usar Up Arrow e Down Arrow em **Player Controls**

Adicionando a bola

- Arraste a imagem da bola para a cena
- Adicione um Tag para a bola, você pode fazer isso na aba **inspector** quando a bola está selecionada. Crie a Tag Ball e selecione esta Tag.
- Aplique a escala (0.5,0.5,1)
- Click em **Add Component**, em seguida em **Physics 2D** e adicione:
  - **Circle collider 2D**
  - **Rigidbody 2D**
- No **Circle collider 2D** altere o raio da bola para 0.23.
- Adicione o material **BallBounce**, arrastando ele para o **Circle collider 2D** da **bola.**
- Em **Rigidbody 2D** altere**:**.
  - **Mass** para 0.1
  - **Angular drag** para 0
  - **Gravity scale** para 0

Mais exemplos de Material podem ser encontrados em:
https://assetstore.unity.com/2d/textures-materials

Vamos agora criar um script para a bola

Selecione **Add Component**, em seguida **New Script**.

Dê um nome para o script de **BallControl**

Vamos agora criar um script para a bola

```
    private Rigidbody2D rb2d;                // Define o corpo rigido 2D que representa a bola
```

Vamos agora criar um script para a bola

```
// inicializa a bola randomicamente para esquerda ou direita
void GoBall(){
    float rand = Random.Range(0, 2);
    if(rand < 1){
        rb2d.AddForce(new Vector2(20, -15));
    } else {
        rb2d.AddForce(new Vector2(-20, -15));
    }
}

void Start () {
    rb2d = GetComponent<Rigidbody2D>(); // Inicializa o objeto bola
    Invoke("GoBall", 2);     // Chama a função GoBall após 2 segundos
}
```

Vamos agora criar um script para a bola

```
// Determina o comportamento da bola nas colisões com os Players (raquetes)
void OnCollisionEnter2D (Collision2D coll) {
    if(coll.collider.CompareTag("Player")){
        Vector2 vel;
        vel.x = rb2d.velocity.x;
        vel.y = (rb2d.velocity.y / 2) + (coll.collider.attachedRigidbody.velocity.y / 3);
        rb2d.velocity = vel;
    }
}
```

Vamos agora criar um script para a bola

```
// Reinicializa a posição e velocidade da bola
void ResetBall(){
    rb2d.velocity = Vector2.zero;
    transform.position = Vector2.zero;
}

// Reinicializa o jogo
void RestartGame(){
    ResetBall();
    Invoke("GoBall", 1);
}
```

Criando paredes

- Crie um novo objeto na aba **hierarchy**.
  - Clique com o botão direito em uma área vazia, em seguida **Create empty**.
  - Renomeie o novo objeto para **walls**
  - Posicione em (0,0,0)

- Crie cada uma das paredes no objeto **walls**.
  - Clique com o botão direito em **walls** e crie um novo objeto vazio.
  - Adicione **box collider 2d**
  - Copie 3x para ter todas as 4 paredes
  - Renomeie os objetos para
    - RightWall
    - LeftWall
    - TopWall
    - BottomWall

Criando paredes

Determine as posições e escalas para

- RightWall => (5.8, 0.0, 0.0) (1.0, 6.0, 1.0)
- LeftWall => (-5.8, 0.0, 0.0) (1.0, 6.0, 1.0)
- TopWall => (0.0, 3.5, 0.0) (10.7, 1.0, 1.0)
- BottomWall => (0.0, -3.5, 0.0) (10.7, 1.0, 1.0)

Adicionando a pontuação

Crie um novo objeto vazio

Renomeie para **Display**

Posicione em (0,0,0)

Adicione um novo script, chamado **GameManager**

Editando o script de pontuação

```
public static int PlayerScore1 = 0; // Pontuação do player 1
public static int PlayerScore2 = 0; // Pontuação do player 2

public GUISkin layout;              // Fonte do placar
GameObject theBall;                 // Referência ao objeto bola
```

Editando o script de pontuação

```
void Start () {
    theBall = GameObject.FindGameObjectWithTag("Ball"); // Busca a referência da bola
}
```

Editando o script de pontuação

```
// incrementa a potuação
public static void Score (string wallID) {
    if (wallID == "RightWall")
    {
        PlayerScore1++;
    } else
    {
        PlayerScore2++;
    }
}
```

Editando o script de pontuação

```
// Gerência da pontuação e fluxo do jogo
void OnGUI () {
    GUI.skin = layout;
    GUI.Label(new Rect(Screen.width / 2 - 150 - 12, 20, 100, 100), "" + PlayerScore1);
    GUI.Label(new Rect(Screen.width / 2 + 150 + 12, 20, 100, 100), "" + PlayerScore2);

    if (GUI.Button(new Rect(Screen.width / 2 - 60, 35, 120, 53), "RESTART"))
    {
        PlayerScore1 = 0;
        PlayerScore2 = 0;
        theBall.SendMessage("RestartGame", null, SendMessageOptions.RequireReceiver);
    }
    if (PlayerScore1 == 10)
    {
        GUI.Label(new Rect(Screen.width / 2 - 150, 200, 2000, 1000), "PLAYER ONE WINS");
        theBall.SendMessage("ResetBall", null, SendMessageOptions.RequireReceiver);
    } else if (PlayerScore2 == 10)
    {
        GUI.Label(new Rect(Screen.width / 2 - 150, 200, 2000, 1000), "PLAYER TWO WINS");
        theBall.SendMessage("ResetBall", null, SendMessageOptions.RequireReceiver);
    }
}
```

- Clique no objeto **Display** na aba **hierarchy** e em seguida selecione o **layout ScoreSkin.**
- Clique na **ScoreSkin** e você verá uma variável chamada **Font** na aba **Inspector**.
- Arraste a fonte **6809 chargen** para essa variável.
- Altere o tamanho da fonte em **label** -> **font size** para **50**

Vamos modificar a pontuação automaticamente.

- Clique em **LeftWall** na aba **hierarchy**
  - Crie um script novo e chame de **SideWalls**
- Clique em **RightWall** na aba **hierarchy**
  - Associe o objeto **RightWall ao** script **SideWalls**
  - **Add Component -> scripts -> SideWalls**
- Em ambos os objetos, no componente **Box collider 2d**, marque a opção **Is trigger**.

Vamos editar o script **SideWalls**

```
using UnityEngine;
using System.Collections;
public class SideWalls : MonoBehaviour {

    // Verifica colisões da bola nas paredes
    void OnTriggerEnter2D (Collider2D hitInfo) {
        if (hitInfo.tag == "Ball")
        {
            string wallName = transform.name;
            GameManager.Score(wallName);
            hitInfo.gameObject.SendMessage("RestartGame", null, SendMessageOptions.RequireReceiver);
        }
    }
}
```

Gerando um executável do seu jogo

Vá em **file**

Clique em **build and run**

Escolha a pasta e o nome do jogo

Clique em ok.

Exercício

Modifique o código do Pong de forma que:

- Duas bolas participam da partida
  - Tratamento de colisão entre as bolas
- Duas raquetes, sendo uma delas controlada pelo computador
  - Como implementar o controle automático da segunda raquete?
- Tenha efeitos sonoros
- Incremento na dificuldade
  - A partir de uma certa pontuação
  - A partir do tempo decorrido do jogo

Como deixar o jogo mais desafiante, mais interessante?

Referência

Todo o material utilizado nessa aula foi adaptado de:

- https://www.awesomeinc.org/tutorials/unity-pong/
- https://www.awesomeinc.org/assets/unity-pong-assets.zip