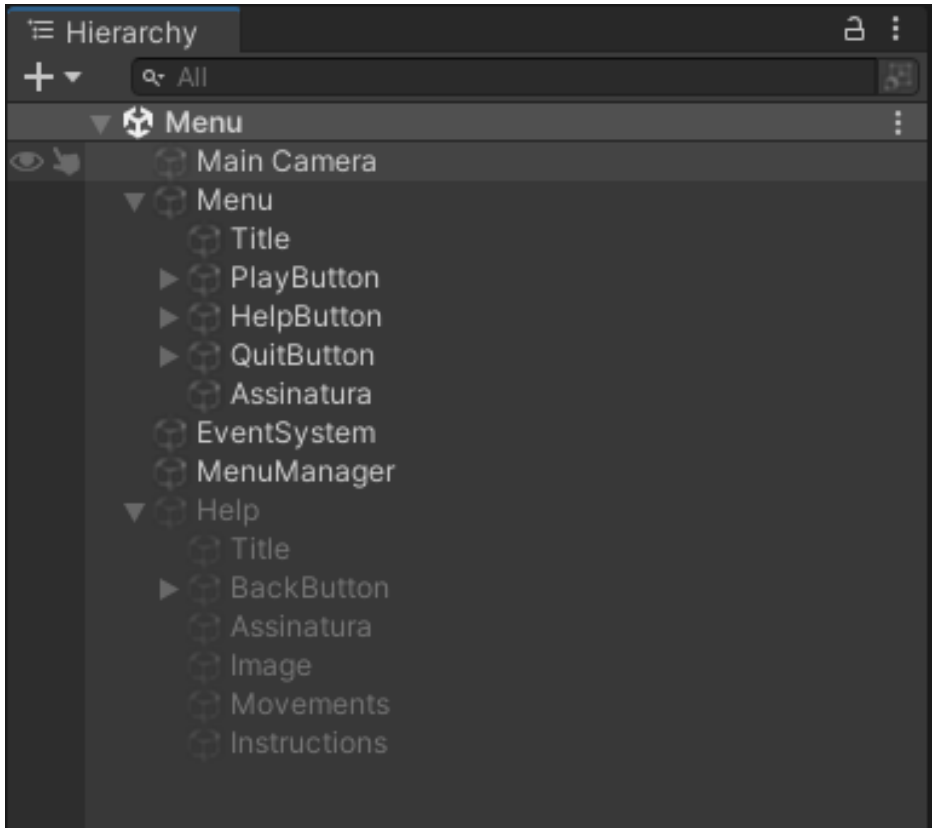
Pong pela Unity - 02/01/2023

Cenas - Menu



Main Camera : Background definido para preto.
Menu (Canvas) : Menu principal

- ***Title (Texto)*** : Título do jogo;
- ***PlayButton*** : Botão para o início do jogo, muda para a cena "Game";
- ***HelpButton*** : Desativa o "Menu" e ativa "Help";
- ***QuitButton*** : Fecha o jogo;
- ***Assinatura*** *(Texto)*.

MenuManager (Objeto vazio) : Possui o script "MenuManager"
Help (Canvas) : Tela com informações dos controles para o jogador;

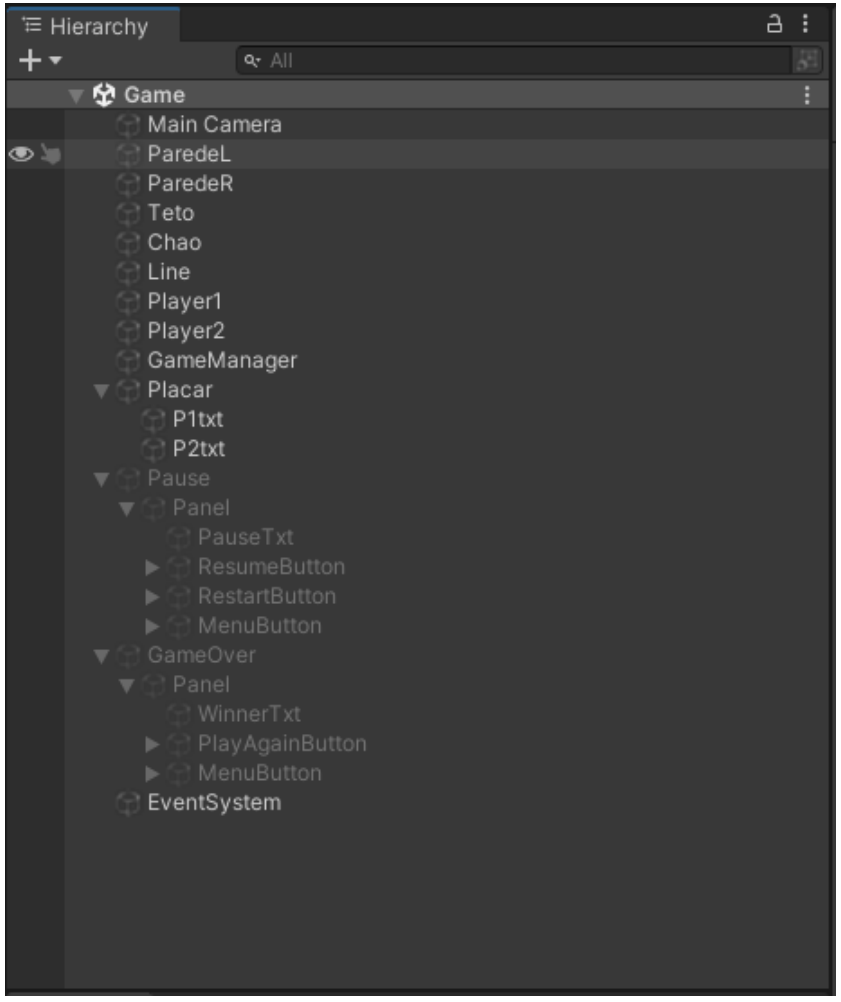
- ***Title (Texto)***;
- ***BackButton*** : Desativa "Help" e ativa "Menu";
- ***Assinatura (Texto)***;
- ***Image (Imagem);***
- ***Movements (Texto);***
- ***Instructions (Texto);***

Cenas - Menu

Menu (Canvas)

Help (Canvas)

Cenas - Game

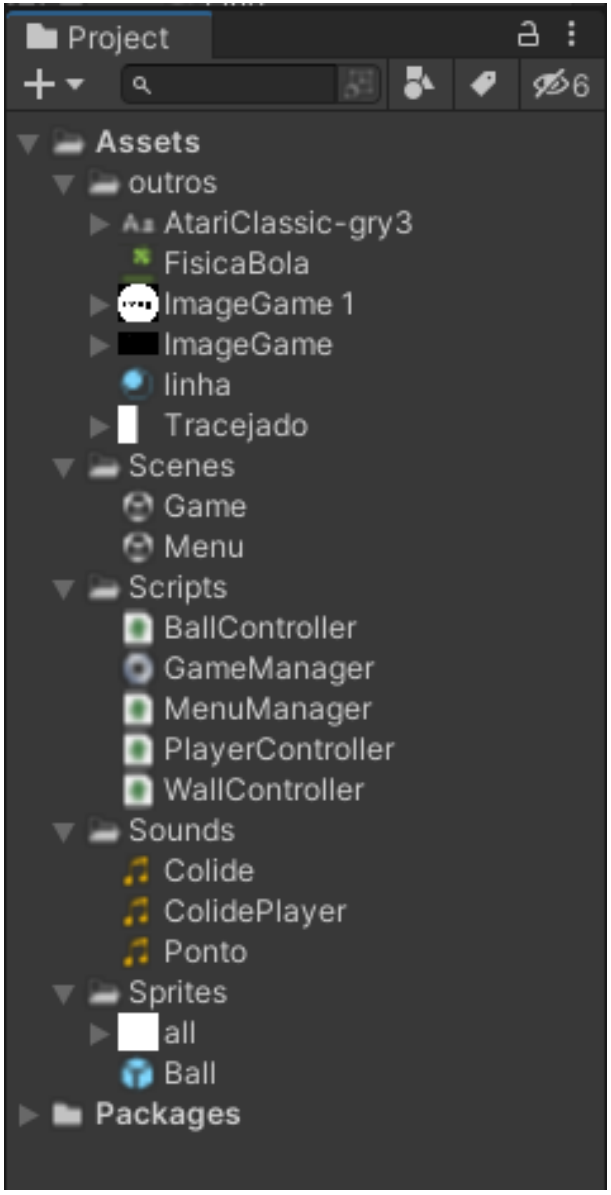


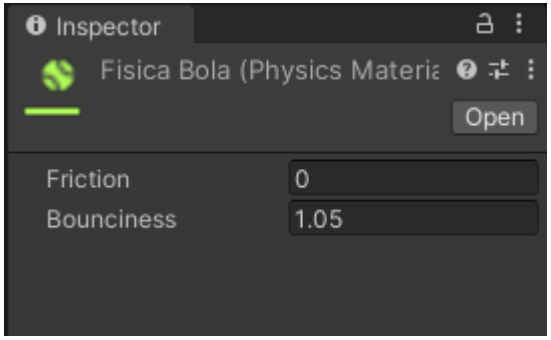
Main Camera : Background definido para preto.
GameManager : Objeto vazio com o script “GameManager”
ParedeL, ParedeR, Teto, Chao e **Line** - Componentes do Cenário.
Player1 e **Player2** - Paddle dos jogadores;
Placar (Canvas) - Pontuação de ambos, sempre ativo;
Pause (Canvas) - Pode ser ativada e desativada com a tecla “esc”;

GameOver (Canvas) - Aparece assim que um dos jogadores completa 11 pontos;

***Mostro as configurações do Inspector no Video.**

Projeto





Outros

- ***Fonte Pixelada;***
- ***FiscaBola***
  - *Friction* : 0 para não grudar em nada;
  - *Bounciness* : 1.05, assim sempre que ela colidir com algo ela volta com uma força um pouco maior que a inicial, ou seja, sua velocidade aumenta aos pouquinhos.
- ***ImageGame 1*** e ***ImageGame*** : Imagens para o ícone do jogo e a tela Help.
- ***Linha*** : Material para dar o tracejado de "Line" na cena.
- ***Tracejado (Imagem)*** : Colocado no material "Linha"

Scripts : Usados na programação.

Sounds : Sons de colisão da bolinha.

Sprites : "all" foi usado para criar os objetos dos jogadores, a bola e as paredes. "Ball" é o objeto da bolinha já configurado, mas fora de cena. Assim podemos definir quando e onde surgir durante o jogo.

***Mostro as configurações do Inspector no Vídeo.**

Scripts - MenuManager

É o mais simples dos scripts, possui uma função para iniciar o jogo e uma para sair.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class MenuManager : MonoBehaviour
{
    public void NextScene(){
        //Ir para a tela do jogo
        SceneManager.LoadScene("Game", LoadSceneMode.Single);
        //Tempo correndo normalmente
        Time.timeScale = 1;
    }
    public void Quit(){
        //Sair do jogo
        Application.Quit();
        Debug.Log(" Quit game!");
    }
}
```

Scripts - BallController

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BallController : MonoBehaviour
{
    //Variável para acessar o componente de Rigidbody deste objeto
    private Rigidbody2D rigi;

    //Variável para acessar o componente de Áudio
    private AudioSource som;

    //Velocidade inicial da bolinha
    private float speed = 300.0f;

    //Getters and Setters
    public float Speed{
        get{ return speed; }
        set{ speed = value; }
    }
```

Usei “Start” em todos os scripts, por padrão. Considerei colocar o “Awake”, mas poderia dar problemas futuros, principalmente na bolinha que não existe assim que o jogo é iniciado.

```
    // Start is called before the first frame update
    //Assim que a bolinha é gerada na cena...
    public void Start()
    {
        //Inicializando as variáveis
        rigi = GetComponent<Rigidbody2D>();
        som = GetComponent<AudioSource>();

        //Definindo movimento
        MovimentoBola();
    }
```

“MovimentoBola” possui os comandos para definir a direção que a bolinha segue assim que é gerada e para dar o impulso inicial.

```
  public void MovimentoBola(){

      //Variáveis para o eixo X e Y da Unity
      float x, y;
      //Direção definida (magnitude de x e y)
      Vector2 magnitude;
      /*X - bolinha sempre começa indo para o player que levou um gol. No início da partida vai para P1
      * "Check" é uma variável do script "GameManager"
      * Ela informa se o último jogador que marcou um ponto foi P1 (false) ou P2 (true)
      */
      if(GameManager.gameManager.Check == false){
          x = 1.0f;
      }else{
          x = -1.0f;
      }

      /*Random.value - Escolhe um valor aleatório entre 0 e 1
      * Se > 0.5 - bolinha vai cima
      * Se < 0.5 - bolinha vai para baixo
      * Random.Range(?, ?) - Define um valor aleatório entre os números informados
      */
      /*Y - Define aleatoriamente se ela vai para cima ou para baixo do centro da tela
      * O ângulo que ela segue também é aleatório
      */
      if(Random.value > 0.5f){
          y = Random.Range(1.0f, 0.1f);
      }else{
          y = Random.Range(-1.0f, -0.1f);
      }

      //Guardando a direção definida e dando o impulso inicial
      magnitude = new Vector2(x, y);
      rigi.AddForce(magnitude * Speed);
  }
```

Sempre que a bolinha detectar uma colisão com o Teto ou Chão, ela reproduz o som de colisão do Pong mesmo.
Também coloquei um comando para mudar um pouquinho o ângulo que ela segue quando é rebatida por qualquer superfície, evitando que ela fique reta indo de um lado para o outro.

```
    //Função ativada sempre que uma colisão é detectada, update desnecessário
    public void OnCollisionEnter2D(Collision2D collision2D){

        //Caso este objeto(bola) colida com o teto ou chão(tag "PdTocar"), reproduz som
        if(collision2D.gameObject.tag == "PdTocar"){
            som.Play();
        }

        /*Sempre que colide com alguma coisa, muda um pouquinho de direção
        * O quanto muda é definido aleatoriamente, mas são sempre valores baixos
        * Impede que ela fique presa em um loop com um mesmo movimento, indo reta de um lado pro outro
        */
        rigi.velocity += new Vector2(Random.Range(0.1f, -0.1f), Random.Range(0.1f, -0.1f));
    }
}
```

Scripts - PlayerControler

Usei somente um script para o controle dos dois jogadores, pois a maior parte dos comandos seriam os mesmos com apenas pequenas alterações. Diferenciei eles pela tag de objeto definida na Unity.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour
{
    //Variável para acessar o componente de Rigidbody deste objeto
    private Rigidbody2D rigi;

    //Variável para acessar o componente de BoxCollider2D
    private BoxCollider2D colide;

    //Variável para acessar o componente de Áudio
    private AudioSource som;

    //Velocidade dos jogadores e direção do movimento
    private float speed = 10f;
    private float direction;

    //Getters and Setters
    public float Speed{
        get{ return speed; }
        set{ speed = value; }
    }

    public float Direction{
        get{ return direction; }
        set{ direction = value; }
    }
```

Start só inicializa os componentes do objeto. Percebi que não usei “colide” no final.

```
// Start is called before the first frame update
    public void Start(){

        rigi = GetComponent<Rigidbody2D>();
        colide = GetComponent<BoxCollider2D>();
        som = GetComponent<AudioSource>();

    }
```

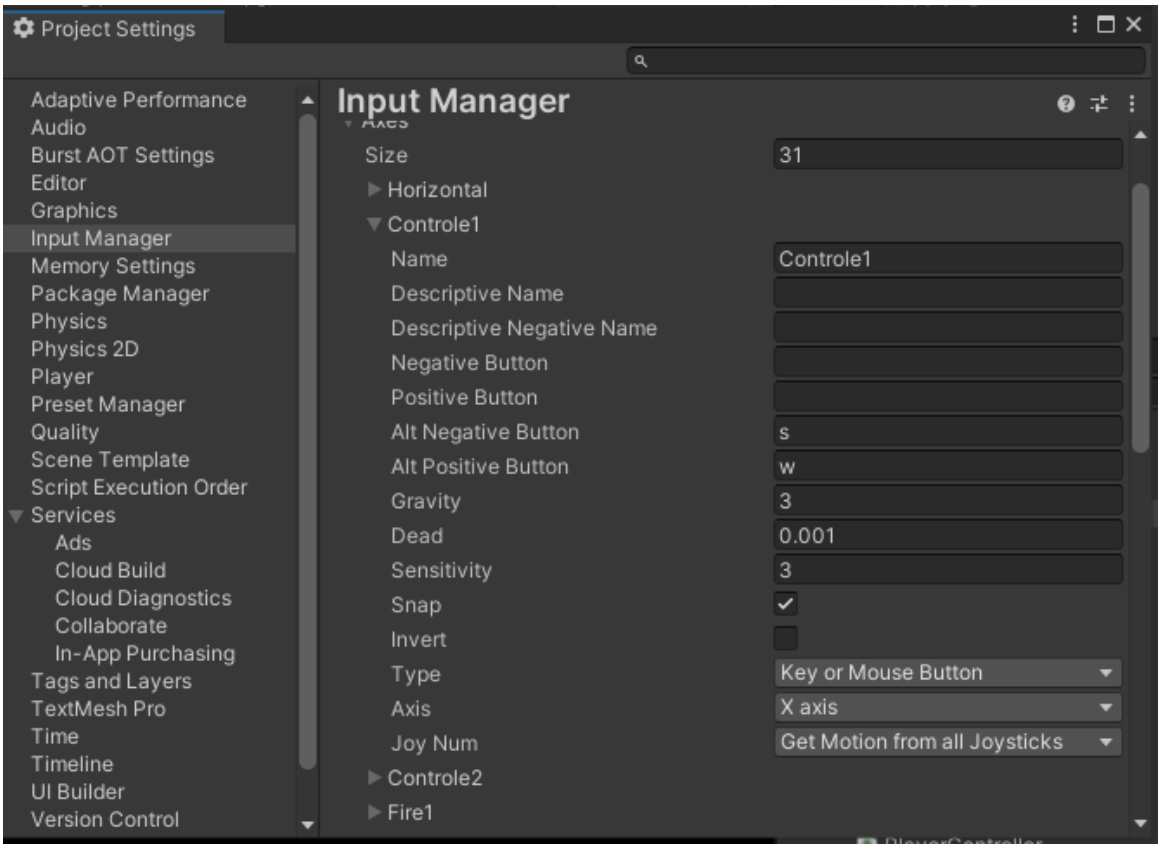
Update sempre confere os comandos dados pelos jogadores.
User FixedUpdate para o movimento em si por ter um tempo definido, o que evita bagunça na física do jogo.

```
/*Função chamada com um tempo definido (0.02)
    * Comando de controle de movimento (fisica do objeto) colocado aqui para evitar errinhos
    */
    public void FixedUpdate(){
        rigi.velocity = new Vector2(rigi.velocity.x, Direction * Speed);
    }

    // Update is called once per frame
    public void Update()
    {
        /*Obtendo entradas dos jogadores 1 e/ou 2
        * N esquecer - Edit -> Project Settings -> Input Manager
        * GetAxisRaw() usado no lugar de GetAxis() para um movimento menos fluido
        */
        if(this.gameObject.tag == "Play1"){
            Direction = Input.GetAxisRaw("Controle1");
        }else{
            Direction = Input.GetAxisRaw("Controle2");
        }

    }
```

Controle1 e Controle2 foram definidos com controles padrões, s/w e setinhas down/up.



Sempre que a bolinha colide com o player reproduz um som diferente do Teto e do Chão.

```
public void OnCollisionEnter2D(Collision2D collision2D){

    //Sempre que a bola colidir com esse objeto, reproduz som
    if(collision2D.gameObject.tag == "Bola"){
        som.Play();
    }
  }
}
```

Script - WallController

Usei também só um script para as duas paredes, da direita e da esquerda, diferenciando com as tags.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class WallController : MonoBehaviour
{
    //Variável para acessar o componente de Áudio
    private AudioSource som;

    // Start is called before the first frame update
    public void Start(){

        som = GetComponent<AudioSource>();
    }
```

Nessa função controlamos quem exatamente marcou o ponto. Pensei em deixar no script "GameManager", mas não fez muito sentido. Precisei buscar alguns elementos dele, pois é lá que os pontos e a manipulação do placar estão. Sempre que computar um ponto e caso nenhum dos jogadores tenha vencido ainda, reproduz um som de destruição e remove a bolinha da cena. Quando alguém chega aos 11 pontos, a bolinha não é mais destruída, fica pulando atrás da tela de Game Over.

*Check = false - P1 foi o último a marcar ponto
Check = true - P2 foi o último a marcar ponto

```
public void OnCollisionEnter2D(Collision2D collision2D){

        if(collision2D.gameObject.tag == "Bola"){

            //Se nenhum dos jogadores alcançou 11 pts ainda
            if((GameManager.gameManager.PontosP1 < 11) &&
(GameManager.gameManager.PontosP2 < 11)){

                //Ponto para P1
                if (this.gameObject.tag == "ParedeR"){
```

```
                GameManager.gameManager.Check = false;
                GameManager.gameManager.AtualizaPlacar();

            //Ponto para P2
            }else {
                GameManager.gameManager.Check = true;
                GameManager.gameManager.AtualizaPlacar();
            }

            som.Play();
            Destroy(collision2D.gameObject);
        }
    }
  }
}
```

Script - GameManager

Este script controla todas as outras funções do jogo, manipulação da UI, troca e reinício de cenas e guarda os pontos dos jogadores.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class GameManager : MonoBehaviour
{
    //Variável para acessar este script pelos outros
    public static GameManager gameManager;

    private Vector2 local;

    //Definidas como public para serem definidas pela Unity
    public GameObject ballPrefab;
    public GameObject pausePanel;
    public GameObject gameOverPanel;

    private int pontosP1;
    private int pontosP2;

    //Definidas como public para serem definidas pela Unity
    public Text PlacarP1;
    public Text PlacarP2;
    public Text Winner;

    //Usada para saber se P1(false) ou P2(true) fez o último ponto
    private bool check;

    //Getters and Setters
    public Vector2 Local{
        get{ return local; }
        set{ local = value; }
```

```
}

public int PontosP1{
    get{ return pontosP1; }
    set{ pontosP1 = value; }
}

public int PontosP2{
    get{ return pontosP2; }
    set{ pontosP2 = value; }
}

public bool Check{
    get{ return check; }
    set{ check = value; }
}
```

Assim que o jogo começa:

- Inicializamos a referência deste script utilizada pelos outros;
- Deixamos o cursor invisível;
- Zeramos os pontos e o placar dos dois jogadores;
- Geramos a bolinha no jogo após 1 segundo, assim os jogadores têm tempo para se preparar.

```
// Start is called before the first frame update
void Start()
{
    if(gameManager == null){
        gameManager = this;
    }

    //Mouse invisível durante a partida
    Cursor.visible = false;

    PontosP1 = 0;
    PlacarP1.text = " " + PontosP1;
    PontosP2 = 0;
    PlacarP2.text = PontosP2 + " ";

    //Bolinha criada após um segundo do início
    Invoke("Spawn", 1.0f);
}
```

Update sempre verifica se a tela de Game Over está ligada, ou seja, se um dos jogadores já venceu o jogo. Caso desligada, chama “GameOver()” para verificar.
Também verifica se o jogador apertou “esc” para pausar/despausar o jogo. Caso ela já esteja ativa, volta ao jogo, caso contrário, para o tempo e volta com o mouse.
Deixei “Resume()” como uma função separada, assim ela pode ser usada pelo botão “Resume” da tela de Pause.

```
// Update is called once per frame
    void Update()
    {
        //Se a tela de Game Over estiver desligada
        if(gameOverPanel.activeSelf == false){

            GameOver();

            //Ao apertar esc
            if(Input.GetKeyDown(KeyCode.Escape)){

                //Se a tela de pause estiver ligada
                if(pausePanel.activeSelf){
                    ResumeGame();

                //Se não
                } else{
                    pausePanel.SetActive(true);
                    Cursor.visible = true;
                    Time.timeScale = 0;
                }
            }
        //Se não
        }else{
            Cursor.visible = true;
        }
    }
```

“Spawn()” cria a bolinha em uma posição aleatória no eixo Y, fazendo com que ela possa aparecer na parte de cima ou de baixo da tela.
"ballPrefab" guarda o objeto pré configurado da bolinha.

```
//Criar a bolinha em um lugar aleatório no eixo Y
    public void Spawn(){

        Local = new Vector2(0, Random.Range(3.0f, -3.0f));
        Instantiate(ballPrefab, Local, transform.rotation);
    }
```

Função para reiniciar a cena e voltar com o tempo normalmente (a tela de pause deixava congelado). Usada pelos botões “Restart” e “Play Again”.

```
//Recomeçar partida
    public void RestartGame(){
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
        Time.timeScale = 1;
        Cursor.visible = false;
    }
```

Função para voltar ao jogo no meio de uma partida. Usada pelo botão “Resume”.

```
//Voltar ao jogo
    public void ResumeGame(){

        pausePanel.SetActive(false);
        Time.timeScale = 1;
        Cursor.visible = false;
    }
```

Atualização dos pontos e do placar dos jogadores. Chamada no script WallController.
Bolinha é Respawned 2 segundos após um ponto ser contabilizada.

```
public void AtualizaPlacar(){

        if(Check == false){
            PontosP1++;
            PlacarP1.text = " " + PontosP1;
        } else{
            PontosP2++;
```

```
            PlacarP2.text = PontosP2 + " ";
        }
        //Bolinha volta 2 segundos após o ponto
        Invoke("Spawn", 2.0f);
    }
```

Verifica se um dos jogadores já marcou 11 pontos. Se sim, game over!

```
//Assim que um dos jogadores marca 11 pts
    public void GameOver(){

        if(PontosP1 >= 11){

            gameOverPanel.SetActive(true);
            Winner.text = "P1 WON!";

        }else if(PontosP2 >= 11){

            gameOverPanel.SetActive(true);
            Winner.text = "P2 WON!";
        }
    }
```

Volta ao menu principal. Usada pelo botão “Menu”

```
//Ir para a tela de menu
    public void NextScene(){
        SceneManager.LoadScene("Menu", LoadSceneMode.Single);
    }
```