Realidade Aumenta
Primeiro Jogo – JOGO DA BOLINA

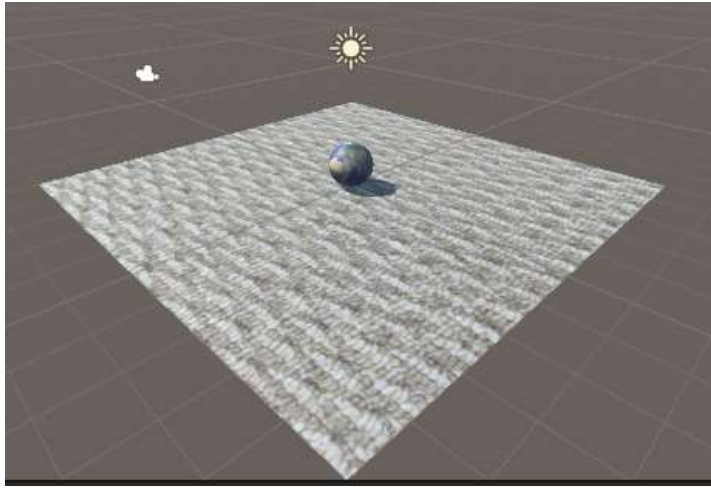
Prof. Rone Ilídio da Silva

Organize a Cena

- Jogo no qual a bolinha é controlada pelo teclado e coleta itens que estão sobre o plano
- Crie um novo projeto chamado Jogo da Bolinha
- Insira:
  - Um plano (GameObject→3D Object→Plane)
  - Uma esfera (GameObject→3D Object→Sphere)

Organize a Cena

- Posicione a bolinha acima do plano: X=0, Y=1 e Z=0 (para ficar acima do plano)
- Coloque texturas na bolinha e no plano
  - Plano: carpete
  - Esfera: planeta terra



Crie um Script Para a Movimentação

- Crie uma pasta chamada Script
- Crie um script dentro dela com o nome MovimentaBolinha (sem espaço)
- Associe esse script à esfera (arrasta ele até a esfera)
- Modifique a classe MovimentaBolinha para:

```
using UnityEngine;
public class MovimentaBolinha : MonoBehaviour
{
    private Rigidbody rb;
    void Start()
    {
        rb = this.GetComponent<Rigidbody>();
    }
    void Update()
    {
        Vector3 move = new
        Vector3(Input.GetAxis("Horizontal"),0,Input.GetAxis("Vertical"));
        rb.AddForce(move);
    }
}
```

Execute o jogo e controle a bolinha com as teclas direcionais teclado

Script MovimetaBolinha

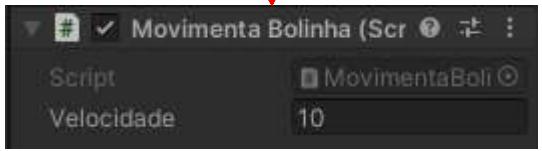
- rb = this.GetComponent<Rigidbody>();
  - O atributo rb recebe o Rigidbody da espera
- GetComponent<Tipo>()
  - Retorna o objeto correspondente a um dos componentes do GameObject
  - “Tipo”: define o tipo do objeto retornado
- Input.GetAxis("Horizontal")
  - Retorna 1 se o direciona para cima está pressionado
  - Retorna -1 se o direciona para baixo está pressionado
- Input.GetAxis(“Vertical")
  - Retorna 1 se o direciona para direita está pressionado
  - Retorna -1 se o direciona para esquerda está pressionado
- Vector3D → classe que define uma direção, utilizado para aplicar forças
- rb.AddForce(move); → aplica uma força em uma direção

Alterando a Velocidade da Bolinha

- Crie um atributo float e público chamado velocidade na classe MovimentaBolinha
- O atributo aparecerá no *Inspector*
- Multiplique o objeto *move* pela velocidade.
- O código modificado ficará da seguinte forma

```
public class MovimentaBolinha : MonoBehaviour
{
    private Rigidbody rb;
    public float velocidade = 1;
    void Start()
    {
        rb = this.GetComponent<Rigidbody>();
    }
    void FixedUpdate()
    {
        Vector3 move = new
            Vector3(Input.GetAxis("Horizontal"),0,Input.GetAxis("Vertical"));
        rb.AddForce(move * velocidade);
    }
}
```

Selecione a esfera e altere o valor da velocidade aqui



Atenção: o método Update() foi alterado para FixedUpdate()

Método FixedUpdate

- Método Update(): chamado a cada frame
- Método FixedUpdate: chamado 50 vezes por segundo (valor configurável em Edit > Settings > Time > Fixed Timestep)
- Utilizado quando há movimentação
- Obs: Neste caso, o Update também funcionaria

Movimentação da Câmera

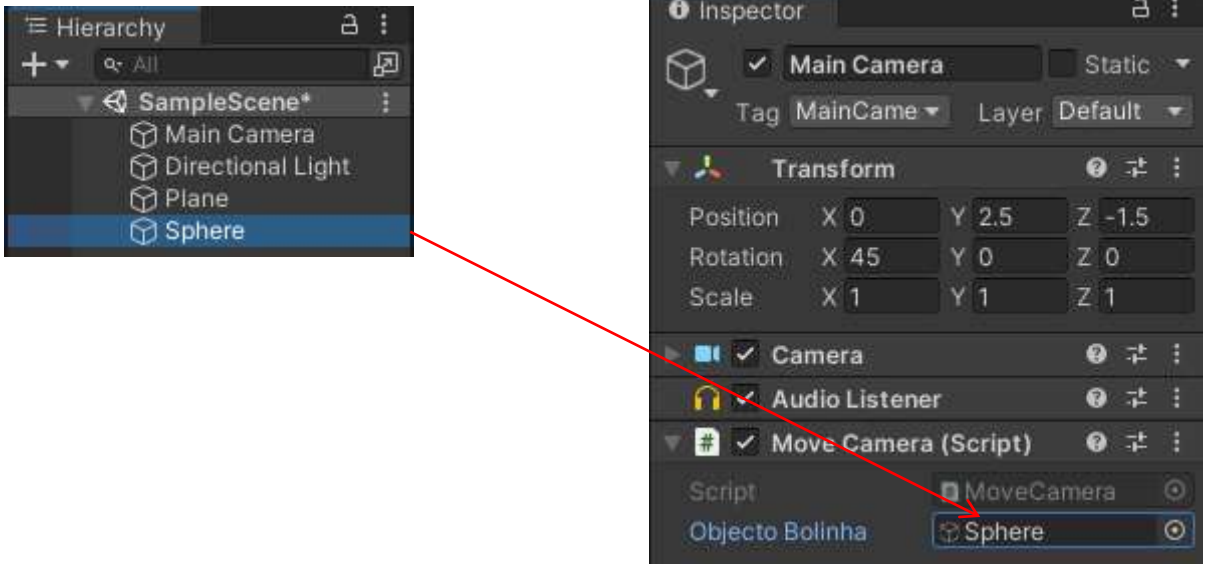
- A câmera acompanhará o movimento da bolinha
- Crie um novo script chamado MoveCamera
- Associe esse script à *mainCamera* (arraste o script da guia *Project* até *mainCamera* em *Hierarchy*)
- Crie o seguinte código:

Script MoveCamera

```
using UnityEngine;
public class MoveCamera : MonoBehaviour
{
    public GameObject objectoBolinha;
    private Vector3 diferenca;
     void Start() {
        diferenca = transform.position - objectoBolinha.transform.position;
    }
    void Update()    {
        transform.position = objectoBolinha.transform.position + diferenca;
    }
}
```

Movimentação da Câmera

- Atenção: associe a esfera ao objetoBolinha que foi declarado em MoveCamera
  - Arraste a esfera (que está em Hierarchy) até objetoBolinha (que está em Inspector/Move Camera/objetoBolinha quanto a câmera é selecionada)



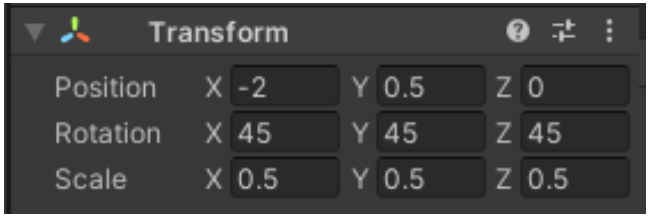
Execute o jogo

Modificando o Transform

- transform: dá acesso ao Transform do objeto associado ao script
- Pode-se alterar
  - transform.position
  - transform.rotation
  - transform.scale
- Todos são objetos do tipo Vector3D: possuem as propriedades x, y e z
  - Exemplo: transform.position.x=10

Inserindo Itens Com Movimento

- Insira um cubo
- Coloque uma textura nele
- Altere o *Transform* para



- Crie um script, associado ao cubo, chamado GiraCubo
- Insira o seguinte código:

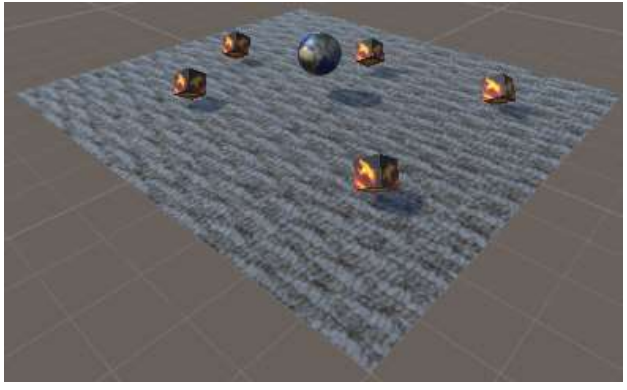
```
public class GiraCubo : MonoBehaviour
{
    void Start()
    {
    }
void Update()
    {
        transform.Rotate(new Vector3(0,0,90) * Time.deltaTime);
    }
}
```

Time.deltaTime = 1/(tempo entre o último frame e o atual)

Giro de 90º por segundo

Criando Prefabs

- Crie uma pasta chamada Prefabs abaixo de Assets
- Arraste o cubo que está em Hierarchy para dentro e Prefabs
- A partir da pasta Prefabs, arraste quantos cubos quiser para a cena
- Obs: cada cubo será criado com Y=0. Você pode posicionar um dos cubos, selecioná-lo e clicar em Control+D que equivale a copiar e colar. O novo cubo ficará sobre o anterior, é necessário mudá-lo de posição.

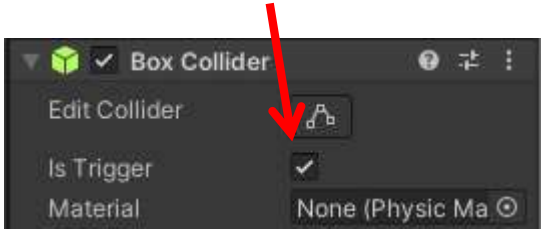


Tratando a Colisão

- A classe MonoBehaviour possui o método OnTriggerEnter() que é chamado pelo evento de colisão
- Esse método recebe um objeto do tipo Collider, o qual possui um atributo gameObject que corresponde ao objeto que colidiu.
- O método Destroy() de MonoBehavior remove da tela o gameObject passado como parâmetro
- Veja código a seguir

```
public class MovimentaBolinha : MonoBehaviour
{
    private Rigidbody rb;
    public float velocidade = 1;
    void Start()   {
        rb = this.GetComponent<Rigidbody>();
     }
    void Update() {
        Vector3 move = new
                              Vector3(Input.GetAxis("Horizontal"),0,Input.GetAxis("Vertical"));
        rb.AddForce(move * velocidade);
    }
    void OnTriggerEnter(Collider outro){
        Destroy(outro.gameObject);
    }
}
```

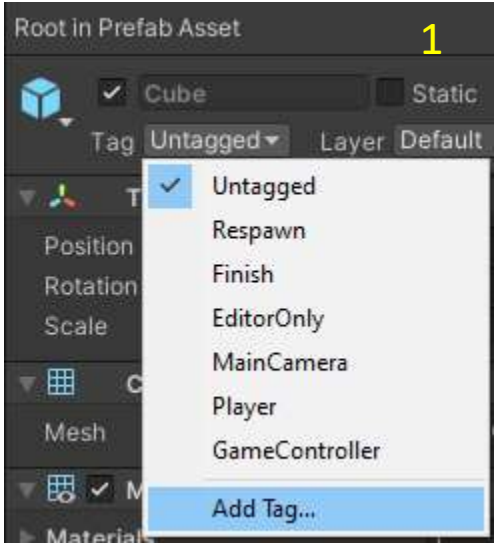
Importante: Selecione o Asset (na pasta Prefabs) do cubo em *Project* e marque a opção isTrigger que está em Box Collider.

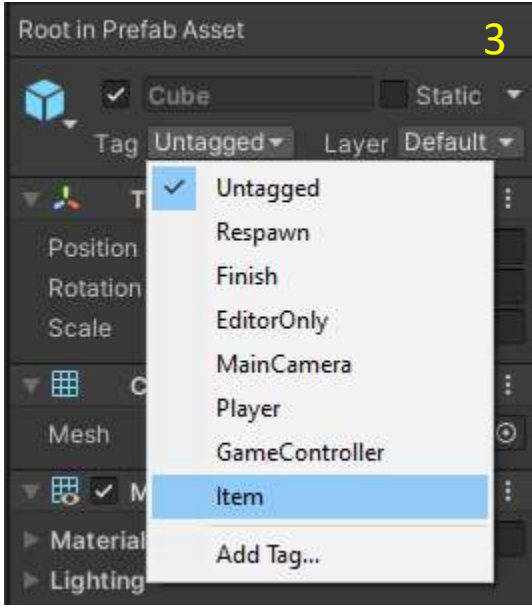


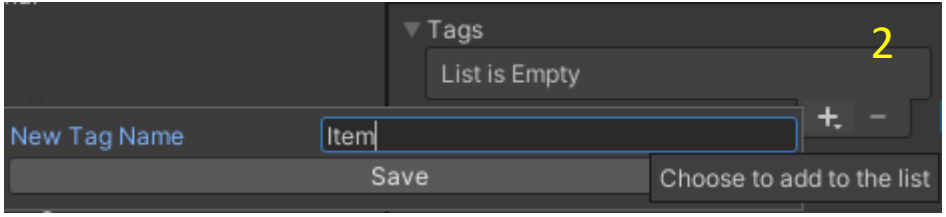
Execute o Jogo

Criando Marcações (tags) para os Objetos

- Selecione o Asset do cubo em Project
- Adicione uma nova tag chamada Item
- Associe o Asset do cubo à tag Item
- Modifique o código ...



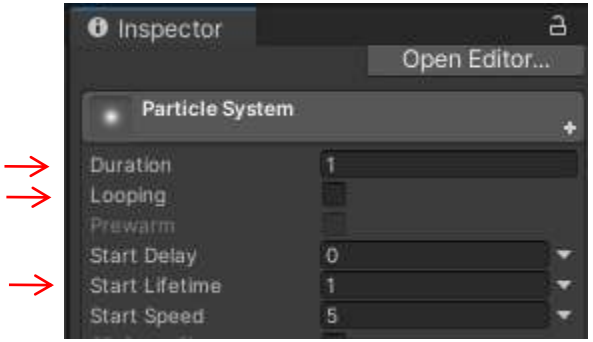




```
public class MovimentaBolinha : MonoBehaviour
{
    private Rigidbody rb;
    public float velocidade = 1;
    void Start(){
        rb = this.GetComponent<Rigidbody>();
    }
    void Update(){
        Vector3 move = new Vector3(Input.GetAxis("Horizontal"),0,Input.GetAxis("Vertical"));
        rb.AddForce(move * velocidade);
    }
    void OnTriggerEnter(Collider outro){
        if(outro.gameObject.CompareTag("Item")) {
            Destroy(outro.gameObject);
         }
    }
}
```

Inserindo Um Particle System

- Particle System é um efeito
- Vamos inserir, configurar e transformá-lo em Prefab
- Insira em GameObject→Effect→Particle System
- Modifique de acordo com a figura
- Arraste o Particle System para a pasta Prefabs em Project
- Apague o Particle System que está em Hierarchy
- Modifique o código para ...



```
public class MovimentaBolinha : MonoBehaviour
{
    private Rigidbody rb;
    public float velocidade = 1;
    public GameObject particulas;
    void Start(){
        rb = this.GetComponent<Rigidbody>();
    }
    void Update(){
        Vector3 move = new Vector3(Input.GetAxis("Horizontal"),0,Input.GetAxis("Vertical"));
        rb.AddForce(move * velocidade);
    }
    void OnTriggerEnter(Collider outro){
        if(outro.gameObject.CompareTag("Item"))
            Instantiate(particulas, outro.gameObject.transform.position, Quaternion.identity);
            Destroy(outro.gameObject);
    }
}
```

Inserindo Um Particle System

- No Unity, associe o Asset Particle System ao atributo partícula da esfera

Execute o jogo

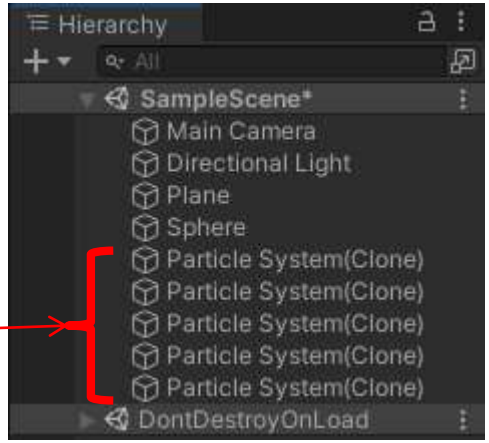
Gerenciamento de Memória

- Cada objeto na cena é um objeto da memória
- Um número grande de objetos pode comprometer o desempenho do jogo
- O método Destroy() remove o objeto da tela e da memória
- Todo objeto que não aparece deve ser destruído

Gerenciamento de Memória

- No jogo que estamos desenvolvendo, cada Particle System é instanciado e não é apagado
- Veja resultado após a execução do jogo

Objetos que ocupam memória sem aparecer na tela



Gerenciamento de Memória

- Para apagar um Particle System após ser exibido, crie o seguinte código e associe ao Prefab do Particle System:

```
public class DestroiObjeto : MonoBehaviour
{
    void Start()    {
        Invoke("ApagaObjeto",1.0f);
    }
    void ApagaObjeto()    {
        Destroy(this.gameObject);
    }
}
```

Chama o método ApagaObjeto após 1 segundo

Remove um objeto da tela e da memória

Gerenciamento de Memória

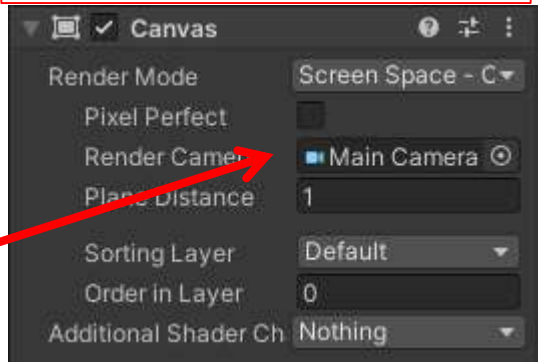
- Para inserir o script DestroiObjeto ao Asset o particlesystem (que está em Prefabs):
  - Selecione o prefab
  - Add Component: digite Destri
  - Escolha Destroi Objeto

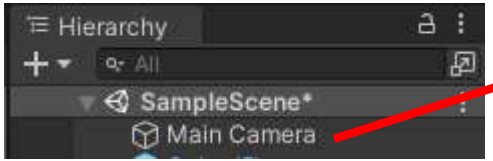
Execute o jogo e faça a esfera colidir com um cubo

Interface Gráfica do Usuário

- Insira um componente GameObject→UI→Text TextMeshPro
  - O texto será filho de Canvas em Hierarchy
  - Canvas equivale à tela vista pelo usuário
- No Inspector, guia Text, mude o conteúdo do texto para “Score”
- Selecione Canvas e na guia Canvas altere:
  - Render Mode: Screen Space Camera
  - Render Camera: arraste a Main Camera
  - Plane Distance: 1

Obs: Não está na versão 6 do Unity





Dê 2 cliques sobre o Canvas para visualizá-lo

Interface Gráfica do Usuário

- O Canvas aparece na guia Scene como um retângulo banco → ajuste o Zoom para visualizá-lo
- O texto aparece no canto inferior esquerdo
  - Selecione o objeto Text em Hierarchy
  - Mude a cor na guia Text/Vertex Color

  (o amarelo realçar)

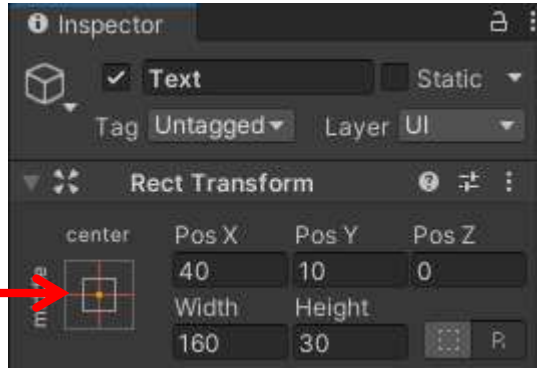
Interface Gráfica do Usuário

- O posicionamento do texto requer atenção
- Ao selecionar Text, é possível movimentar o texto em 3D, mas o resultado aparece em 2D
- Solução → movimentar no Inspector

Interface Gráfica do Usuário

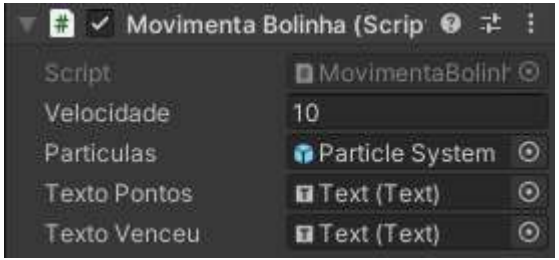
- Insira outro Text no Canvas
- Altere seu texto para vazio
- Aumente sua fonte, mas atenção:
  - A fonte está relacionada com o tamanho da caixa de texto, fontes muito grandes não aparecem na cena.
  - O tamanho da caixa de texto você altera em React Transform/Width e Height
- Posicione da seguinte forma:

Obs: altera a referência dos eixos



Interface Gráfica do Usuário

- Para atualizar o Score e controlar You Win!
  - Em MovimentaBolinha, crie os atributos:
    - pontos: inteiro
    - public TextMeshProUGUI textoPontos
    - public TextMeshProUGUI textoVenceu
  - Obs: importe TMPro (using TMPro;) para utilizar a classe TextMeshProUGUI
  - No Unity, selecione a esfera e associe o objeto Text dos pontos ao atributo textoPontos do script MovimentaBolinha
  - O mesmos para textoVenceu
- Veja o código modificado...



Código Final

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
public class MovimentaBolinha : MonoBehaviour
{
    private Rigidbody rb;
    public float velocidade = 1;
    public GameObject particulas;
    public TextMeshProUGUI textoPontos;
    public TextMeshProUGUI textoVenceu;
    private int pontos = 0;
```

```
    void Start(){
        rb = this.GetComponent<Rigidbody>();
        textoVenceu.text = "";
    }
    void Update(){
        Vector3 move = new Vector3(Input.GetAxis("Horizontal"),0,Input.GetAxis("Vertical"));
        rb.AddForce(move * velocidade);
    }
    void OnTriggerEnter(Collider outro){
        if(outro.gameObject.CompareTag("Item"))
            Instantiate(particulas, outro.gameObject.transform.position, Quaternion.identity);
            Destroy(outro.gameObject);
            pontos++;
            textoPontos.text = "Score: " + pontos;
            if(pontos>=5)
                textoVenceu.text="You Win!";
    }
}
```