Temporizadores – Unity 3d

Prof. Me. Hélio Esperidião

Temporizador

- Em outras palavras, um temporizador é um dispositivo capaz de medir o tempo, que pode ser usado para controlar um evento ou processo.

- No contexto de jogos imagine um bônus de velocidade de movimento que dura 2s ou um bônus de velocidade de pulo que dure 10 segundos.

Time.deltaTime

- No Unity, deltaTime é uma propriedade da classe Time que representa o tempo em segundos que levou para concluir o último quadro (frame).
- É frequentemente usado em jogos e aplicações gráficas para garantir que a movimentação, animações e outras operações baseadas em tempo sejam independentes da taxa de quadros (frame rate).
- Isso significa que, independentemente de quantos quadros por segundo o jogo está rodando, os objetos no jogo se moverão a uma velocidade constante.

Configuração inicial

- Losango:
  - Component >> Physic2D>>PolygonCollider2D
  - Is Trigger : True.
  - Tag: tipo1
- Triangulo
  - Component >> Physic2D>>PolygonCollider2D
  - Is Trigger : True.
  - Tag: tipo2.
- Quadrado - Personagem:
  - Component >> Physic2D>>Box Collider 2D
  - Component >> Physic2D>>RigidBody 2D

Atributos

```
float Vx;
float Vy;
float VelocidadeAndar;
float VelocidadePular;
float ContadorPulos;
float TotalPulos;
float DirecaoHorizontal;
Rigidbody2D CorpoRigido;
PolygonCollider2D Colisor;
bool BonusVelocidadeAndar;
bool BonusVelocidadePular;
float TempoBonusVelocidadeAndar;
float TempoBonusVelocidadePular;
SpriteRenderer Renderer;
```

Variável para contar se o bônus está ativo ou se o bônus está inativo.

Variável utilizada para contar o tempo decorrido de uma ação.

void Start ()

Inicializa as variáveis

```
void Start () {
 TempoBonusVelocidadeAndar= 2.0f;  // 2 segundos
 TempoBonusVelocidadePular= 10.0f; // 10 segundos
 BonusVelocidadeAndar = false;
 BonusVelocidadePular = false;
 TotalPulos = 2;
 VelocidadeAndar = 5;
 VelocidadePular    = 5;
 CorpoRigido = GetComponent<Rigidbody2D> ();
 Colisor = GetComponent<PolygonCollider2D> ();
 Renderer = GetComponent<SpriteRenderer>();
 CorpoRigido.freezeRotation = true;
}
```

void Update ()

```
void Update () {
    MovimentoHorizontalFlip ();
    PuloDuplo();
    VerificarBonusAndar();
    VerificarBonusPular();
}
```

void OnTriggerEnter2D()

```
void OnTriggerEnter2D(Collider2D objetoTriggerTocado){
        string tagTocadaTrigger = objetoTriggerTocado.gameObject.tag;
        if (tagTocadaTrigger == "tipo1") {
            print ("OnTriggerEnter2D: " + tagTocadaTrigger);
            BonusVelocidadeAndar = true;
            VelocidadeAndar = 25;
            Destroy (objetoTriggerTocado.gameObject);
        }
        if (tagTocadaTrigger == "tipo2") {
            print ("OnTriggerEnter2D: " + tagTocadaTrigger);
            BonusVelocidadePular = true;
            VelocidadePular = 15;
            Destroy (objetoTriggerTocado.gameObject);
        }
    }
```

Quando o personagem toca em algo que proporciona Bonus de velocidade a intensidade do movimento muda
E a variável bônus ativado recebe true;

void VerificarBonusAndar()

```
void VerificarBonusAndar(){
    if (BonusVelocidadeAndar == true) { // se o bonus for atividado
        //se o tempo de bônus for positivo
        if (TempoBonusVelocidadeAndar >= 0) {
            //Time.deltaTime: conta o tempo entre os frames
            //Diminui o tempo entre os frames do tempo total do bonus.
            TempoBonusVelocidadeAndar = TempoBonusVelocidadeAndar - Time.deltaTime;
            print ("Tempo Bonus Andar: " + TempoBonusVelocidadeAndar);
        } else {
            //cai no else se o tempo for menor que zero
            BonusVelocidadeAndar = false; //desativa o bonus
            TempoBonusVelocidadeAndar = 2; //retorna o tempo inicial
            VelocidadeAndar = 5;//retorna a velocidade inicial.
        }
    }
}
```

Como o método update é executado a cada frame
Não seria possível simplesmente contar o tempo por meio da troca de frames, pois computadores mais eficientes trocam mais frames.

O delta Time conta o tempo entre os frames
Com isso não importa a quantidade de Frames por segundo e sim o tempo que passa entre um frame e outro

void VerificarBonusPular()

```
void VerificarBonusPular(){
        //controle do tempo do bonus de pular mais rápido
        // se o bonus for atividado
        if (BonusVelocidadePular == true) {
            //se o tempo de bonus for positivo
            //Diminui o tempo entre os frames do tempo total do bonus.
            if (TempoBonusVelocidadePular >= 0) {
                TempoBonusVelocidadePular = TempoBonusVelocidadePular - Time.deltaTime;
                print ("Tempo Bonus Pular: " + TempoBonusVelocidadePular);
            } else {
                //cai no else se o tempo for menor que zero
                BonusVelocidadePular = false; //desativa o bonus
                TempoBonusVelocidadePular = 10; //retorna o tempo inicial
                VelocidadePular = 5;    //retorna a velocidade inicial.
            }
        }
    }
```

Como o método update é executado a cada frame
Não seria possível simplesmente contar o tempo por meio da troca de frames, pois computadores mais eficientes trocam mais frames.

O delta Time conta o tempo entre os frames
Com isso não importa a quantidade de Frames por segundo e sim o tempo que passa entre um frame e outro

void OnCollisionEnter2D()

```
void OnCollisionEnter2D(Collision2D objetoTocado)
    {
        ContadorPulos = 0;

    }
```

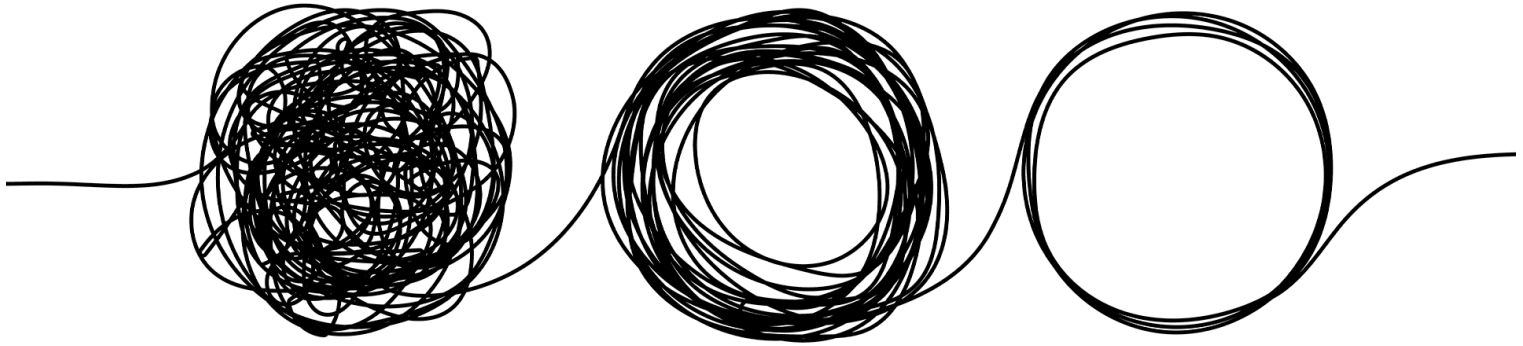
void PuloDuplo()

```
void PuloDuplo(){
    //verdadeiro(true) se o botão de Jump foi pressionado caso contrário igual a false
    bool apertou = Input.GetButtonDown ("Jump");
    // esse bloco só é executado se for apertado o botão de Jump
    // e se o personagem ContadorPulos < 2
    if (apertou == true && ContadorPulos < TotalPulos ) {
        ContadorPulos = ContadorPulos+1;
        Vx = CorpoRigido.velocity.x;
        Vy = VelocidadePular;
        Vector2 pulo = new Vector2 (Vx, Vy);
        CorpoRigido.velocity = pulo;
    }
}
```

void MovimentoHorizontalFlip()

```
void MovimentoHorizontalFlip(){
        DirecaoHorizontal = Input.GetAxis ("Horizontal");
        // Gera uma nova velocidade em x
        Vx = VelocidadeAndar * DirecaoHorizontal;
        //Recupera a velocidade em y que o personagem já possui
        Vy = CorpoRigido.velocity.y;
        //Cria um vetor de velocidade com os valores da velocidade em x e da velocidade em y (vx e vy)
        Vector2 andar = new Vector2 (Vx, Vy);
        // O vetor de velocidade é adicionado a velocidade do corpo rígido do personagem
        CorpoRigido.velocity = andar;
        if (DirecaoHorizontal < 0) {
            Renderer.flipX = true;
        }else if(DirecaoHorizontal > 0){
            Renderer.flipX = false;
        }
    }
```

Simplificando



SIMPLIFICANDO TUDO

involker

- O método Invoke pode chamar (invocar) um outro método dentro do script em um determinado tempo através do seu nome que deve ser passado como uma string.
- Exemplo:
  - `Invoke ("metodo", daquiQuantoTempo);`

```
void metodo(){
    //fazer alguma coisas depois que o tempo passar.
}
```

Vantagens e desvantagens

- Vantagem
  - Menos linhas de código.
  - Mais simples de entender.
- Desvantagem
  - Não é possível apresentar a contagem regressiva do tempo.
  - Menos controle do programador.

configuração

- Azul:
  - Component >> Physic2D>>PolygonCollider2D
  - Is Trigger : True.
  - Tag: tipo3
- Vermelho
  - Component >> Physic2D>>PolygonCollider2D
  - Is Trigger : True.
  - Tag: tipo4.
- Quadrado - Personagem:
  - Component >> Physic2D>>Box Collider 2D
  - Component >> Physic2D>>RigidBody 2D

Atributos

```
float Vx;     //velocidade em x
float Vy;    //velocidade em y
float VelocidadeAndar;
float VelocidadePular;
float ContadorPulos;
float TotalPulos;
float DirecaoHorizontal;
//corpo rígido do elemento onde o script foi inserido
Rigidbody2D CorpoRigido;
SpriteRenderer Renderer;
float TempoBonusVelocidadeAndar;
float TempoBonusVelocidadePular;
```

void Start ()

```
void Start () {
    TempoBonusVelocidadeAndar= 2.0f; // 2 segundos
    TempoBonusVelocidadePular= 10.0f; // 10 segundos
    TotalPulos = 2;
    VelocidadeAndar = 5;
    VelocidadePular    = 5;
    //corpo rigido do elemento que possui o script
    CorpoRigido = GetComponent<Rigidbody2D> ();
    Renderer = GetComponent<SpriteRenderer>();
    CorpoRigido.freezeRotation = true;
    //impede que rotacione no eixo.
    CorpoRigido.gravityScale = 1;
}
```

void Update ()

```
void Update () {
    MovimentoHorizontalFlip ();
    PuloDuplo();
}
```

void OnTriggerEnter2D()

```
void OnTriggerEnter2D(Collider2D objetoTriggerTocado){
        string tagTocadaTrigger = objetoTriggerTocado.gameObject.tag;
        if (tagTocadaTrigger == "tipo3") {
            print ("OnTriggerEnter2D: " + tagTocadaTrigger);
            VelocidadeAndar = 25;
            //Invoke permite chamar um método depois de um determinado tempo
            //no exemplo o void DesativarBonusVelocidadeAndar() será chamado
            //depois do TempoBonusVelocidadeAndar;
            Invoke ("DesativarBonusVelocidadeAndar", TempoBonusVelocidadeAndar);
            Destroy (objetoTriggerTocado.gameObject);
        }
        if (tagTocadaTrigger == "tipo4") {
            print ("OnTriggerEnter2D: " + tagTocadaTrigger);
            VelocidadePular = 25;
            Invoke ("DesativarBonusVelocidadePular", TempoBonusVelocidadePular);
            Destroy (objetoTriggerTocado.gameObject);
        }

    }
```

void DesativarBonusVelocidadeAndar()/
void DesativarBonusVelocidadePular()

```
void DesativarBonusVelocidadeAndar(){
    VelocidadeAndar = 5;
    print ("Bônus de velocidade desativado");
}
void DesativarBonusVelocidadePular(){
    VelocidadePular = 5;
    print ("Bônus de velocidade de pular desativado");
}
```

```
Invoke ("DesativarBonusVelocidadeAndar", TempoBonusVelocidadeAndar);
```

```
Invoke ("DesativarBonusVelocidadePular", TempoBonusVelocidadePular);
```

void OnCollisionEnter2D()

```
void OnCollisionEnter2D(Collision2D objetoTocado)
    {
        ContadorPulos = 0;
        string tagObjetoTocado = objetoTocado.gameObject.tag;
        if (tagObjetoTocado == "tipo1") {
            print ("OnCollisionEnter2D: " + tagObjetoTocado);
            Destroy(objetoTocado.gameObject);
        }
    }
```

void MovimentoHorizontalFlip()

```
void MovimentoHorizontalFlip(){
    DirecaoHorizontal = Input.GetAxis ("Horizontal");
    // Gera uma nova velocidade em x
    Vx = VelocidadeAndar * DirecaoHorizontal;
    //Recupera a velocidade em y que o personagem já possui
    Vy = CorpoRigido.velocity.y;

    Vector2 andar = new Vector2 (Vx, Vy);

    CorpoRigido.velocity = andar;
    if (DirecaoHorizontal < 0) {
        Renderer.flipX = true;
    }else if(DirecaoHorizontal > 0){
        Renderer.flipX = false;
    }
}
```

void PuloDuplo()

```
void PuloDuplo(){
     bool apertou = Input.GetButtonDown ("Jump");
    // esse bloco só é executado se for apertado o botão de Jump
    // e se o personagem ContadorPulos < 2
    if (apertou == true && ContadorPulos < TotalPulos ) {
        ContadorPulos = ContadorPulos+1;
        Vx = CorpoRigido.velocity.x;
        Vy = VelocidadePular;
        Vector2 pulo = new Vector2 (Vx, Vy);
        CorpoRigido.velocity = pulo;
    }
}
```