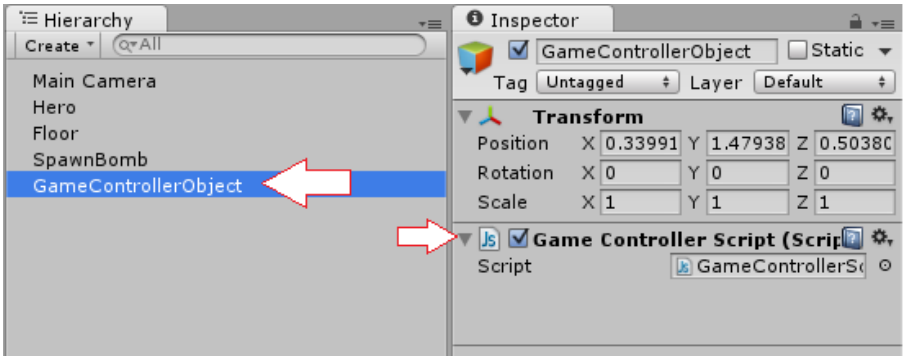
LABORATÓRIO: CRIANDO O JOGO MEGAMAN 2D

Parte-3

1. Vamos inserir o HUD (Head-Up Display) no nosso jogo. Nele exibiremos os pontos marcados pelo jogador e a saúde do herói.

2. Primeiro vamos criar um objeto vazio que servirá para controlarmos o estado do jogo, onde controlaremos os pontos e a saúde. Crie um objeto vazio e chame-o de *GameControllerObject*. Em seguida, crie um script com o nome de *GameControllerScript* e adicione-o ao objeto *GameControllerObject*.



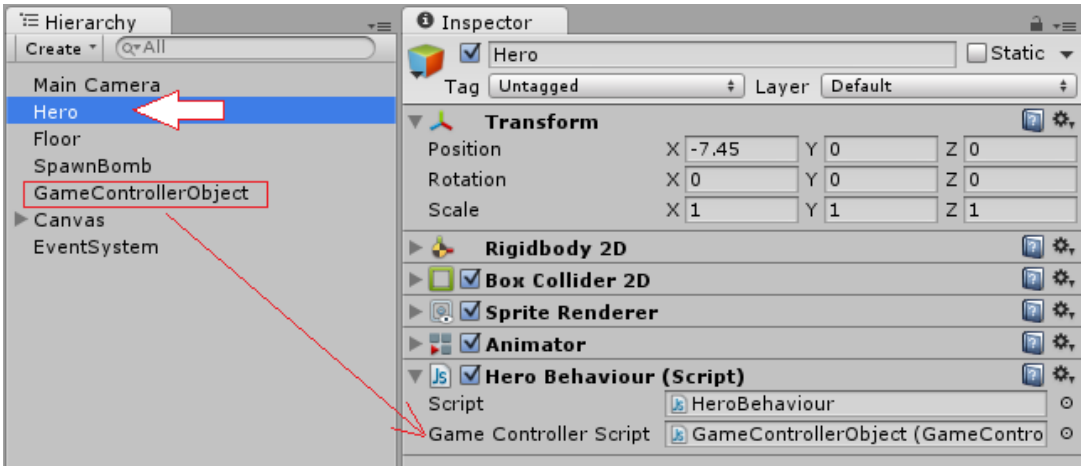
3. Abra o script criado e insira nele o código a seguir. Inicialmente definimos variáveis para controlar os pontos e a saúde do jogador. Em seguida, definimos referências aos objetos de UI, sendo um *Text* e um *Slider*, os quais irão exibir os dados sobre pontos e saúde. Criamos a variável *jogadorParado* para podermos parar a movimentação e criação das bombas quando o herói estiver no estado "*Parado*". Por fim, foram adicionadas duas funções, sendo uma para adicionar os pontos e outra para remover saúde do herói e, quando a saúde zerar, iremos exibir a cena de "*Game Over*". Perceba que iremos armazenar os pontos marcados pelo usuário através da classe *PlayerPrefs* – isso é necessário para passarmos informações através das cenas do nosso jogo (neste caso entre as cenas "*jogo*" e "*gameOver*").

```
//variaveis para controlar os pontos e a saude
private var pontos: int = 0;
private var saude:  int = 100;

//Objetos de UI
public var  pontosUI: UnityEngine.UI.Text;
public var  saudeUI:  UnityEngine.UI.Slider;

//indica quando o jogador estah no estado "Parado",
//serah usada para parar as movimentaçoes dos objetos
public var jogadorParado: boolean = true;

function AdicionarPontos(){
    pontos ++;
    pontosUI.text = "Pontos: " + pontos;
}

function RemoverSaude(){
    saude -= 20;
    saudeUI.value = saude;
    //se zerar a saude finaliza o jogo
    if (saude == 0){
        jogadorParado = true;
        PlayerPrefs.SetInt("pontos", pontos);
        Application.LoadLevel("gameOver");
    }
}
```

4. Agora abra o script *HeroBehaviour* e insira nele uma variável para armazenar uma referência ao script *GameControllerScript*. Precisamos dessa referência para podermos ter acesso às funções e variáveis do script *GameControllerScript*. Faça a mesma coisa nos scripts *BombBehaviour* e *SpawnBombBehaviour*.

```
//referencia ao script que controla o status do jogo
public var gameControllerScript: GameControllerScript;
```

5. Agora volte à IDE do Unity, selecione o objeto *Hero* na janela *Hierarchy* e arraste e solte o objeto *GameControllerObject* para a propriedade "*Game Controller Script*" do seu componente de script. Faça a mesma coisa no script do objeto *SpawnBomb*.



6. Agora abra o script *HeroBehaviour* e altere a função *OnCollisionEnter2D()* como exibido a seguir. Aqui iremos chamar a função *RemoverSaude()* sempre que o herói colidir com uma das bombas.

```
//Esta funçao eh executada quando o player toca no piso
function OnCollisionEnter2D(coll: Collision2D) {
    if ( coll.gameObject.CompareTag("Bomba") ){
        Destroy(coll.gameObject);
        gameControllerScript.RemoverSaude();
    }
    else if ( coll.gameObject.CompareTag("Piso") ){
        noPiso = true;
        pulando = false;
    }
}
```

7. Altere também neste script o seguinte bloco de código da função *Update()*, conforme exibido. Aqui modificamos a variável *jogadorParado* de acordo com o estado do herói.

```
//quando o jogador deixar pressionada a tecla "SETA PARA DIREITA"
//atribui TRUE para o parametro "Correndo" do animator
if (Input.GetKeyDown("right") ){
    if (noPiso){
        animator.SetBool("Correndo", true); //animaçao "Run"
        gameControllerScript.jogadorParado = false;
    }
}
//quando o jogador liberar a tecla "SETA PARA DIREITA"
//atribui FALSE para o parametro "Correndo" do animator
else if (Input.GetKeyUp("right") ){
    animator.SetBool("Correndo", false); //animaçao "Idle"
    gameControllerScript.jogadorParado = true;
}
```

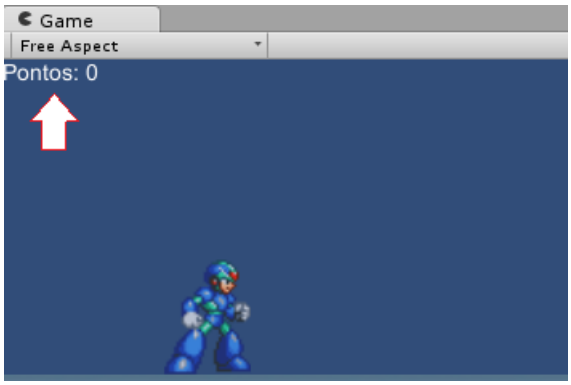
8. Agora abra o script *BombBehaviour* e altere o seguinte bloco de código da função *Update()*, conforme exibido. Aqui fazemos uma verificação para saber se o herói está parado e, caso esteja, não será executada a sua movimentação para a esquerda. Por fim, chamamos a função *AdicionarPontos()* sempre que uma bomba ultrapassar o limite de visão da cena e for destruída.

```
function Update () {
    //se o jogador estiver parado nao movimenta as bombas
    if ( ! gameControllerScript.jogadorParado){

        //obtem o valor da posiçao X do objeto Bomb
        var x = gameObject.transform.position.x;

        //se a posiçao X for maior que -15, move-a para a direita
        //fazendo o calculo utilizando o Time.deltaTime
        if (x > -15){
            x += -1 * velocidade * Time.deltaTime;
            gameObject.transform.position =
                new Vector3(x, gameObject.transform.position.y, gameObject.tr
        }
        else {
            //destroi o objeto bomba porque ele saiu da visao da camera
            Destroy(gameObject);
            gameControllerScript.AdicionarPontos();
        }
    }
}
```

9. Abra o script *SpawnBombBehaviour* e insira na função *Update()* a mesma verificação que fizemos no passo anterior. Com isso, caso o herói esteja parado, será interrompido o processo de criação das bombas. Insira também o código para atribuir ao objeto bomba criado a instância do componente *GameControllerScript* do objeto *SpawnBomb*.

```
function Update () {
    //se o jogador estiver parado nao cria as bombas
    if ( ! gameControllerScript.jogadorParado){

        //armazena o tempo decorrido
        tempoDecorrido += Time.deltaTime;
        //se o tempo decorrido for >= que o intervalo aleatorio de spawn
        if( tempoDecorrido >= intervaloSpawn) {
            //cria um objeto Bomb e posiciona-o junto com o SpawnBomb
            var obj: GameObject = Instantiate(spawnPrefab);
            obj.transform.position = gameObject.transform.position;

            //atribui o objeto GameControllerScript na bomba criada
            var prefabScript: BombaBehaviourScript = obj.GetComponent.<BombaBehaviourScript>();
            prefabScript.gameControllerScript = gameControllerScript;

            //zera o tempo decorrido e gera um novo intervalo aleatorio
            tempoDecorrido = 0;
            intervaloSpawn = Random.Range(intervaloMinSpawn, intervaloMaxSpawn);
        }
    }
}
```

10. Agora vamos criar os objetos de UI para a exibição dos pontos e da saúde do herói. Crie um objeto Text (*Create->UI->Text*) e chame-o de *PontosText*. Altere sua posição para que ele fique no canto superior esquerda da tela. Por fim, altere seu texto para "*Pontos: 0*" e sua cor para branca.



11. Agora insira um objeto *Slider* (*Create->UI->Slider*) e chame-o de *SaudeSlider*. Posicione-o no canto superior direito da tela. Altere sua propriedade *Min Value* para "*0*", *Max Value* para "*100*" e *Value* para "*100*".

    Na janela *Hierarchy*, expanda o objeto *Fill Area* do *Slider*, selecione o objeto *Fill* e altere sua cor para vermelho. Agora você terá que ajustar as posições dos objetos *Background* e *Fill Area* do *Slider* para que eles fiquem sobrepostos – estes objetos representam o fundo e o preenchimento do *Slider*.

    Insira um novo Text, chame-o de SaudeText e altera seu texto para "Saúde". Posicione no canto superior direito da tela e deixe-o à esquerda do Slider. Mude sua cor para amarelo.

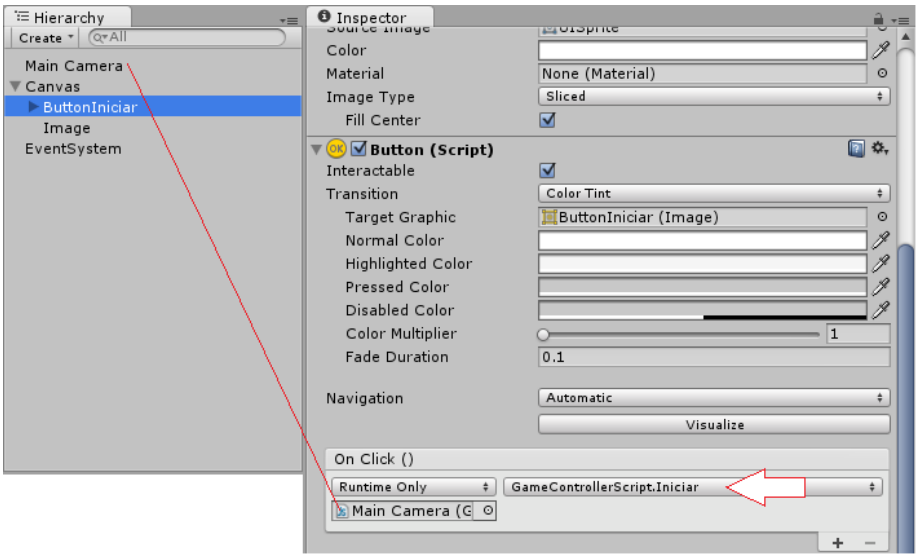
12. Agora vamos criar duas novas scenes no jogo. Chame a primeira de "*menu*" e a segunda de "*gamoOver*". Estas cenas serão usadas, respectivamente, como o menu inicial do jogo e a tela de *game over*.

13. Abra a cena "menu" e insira um objeto *Button* e um *Image*. Defina o texto do botão como "*Iniciar*" e o *Source Image* da imagem para um dos sprites do herói.

14. Agora abra o script *GameControllerScript* e insira nele a seguinte função, a qual irá carregar a cena "jogo".

```
function Iniciar(){
    Application.LoadLevel("jogo");
}
```

15. Selecione o objeto *Main Camera* e adicione nele o script *GameControllerScript*. Agora selecione o botão “Iniciar” e adicione a *Main Camera* no seu componente “*Button (Script)*”. Em seguida, selecione a função *Iniciar()* do *GameControllerScript* – isso fará com que ela seja chamada quando o botão for clicado.



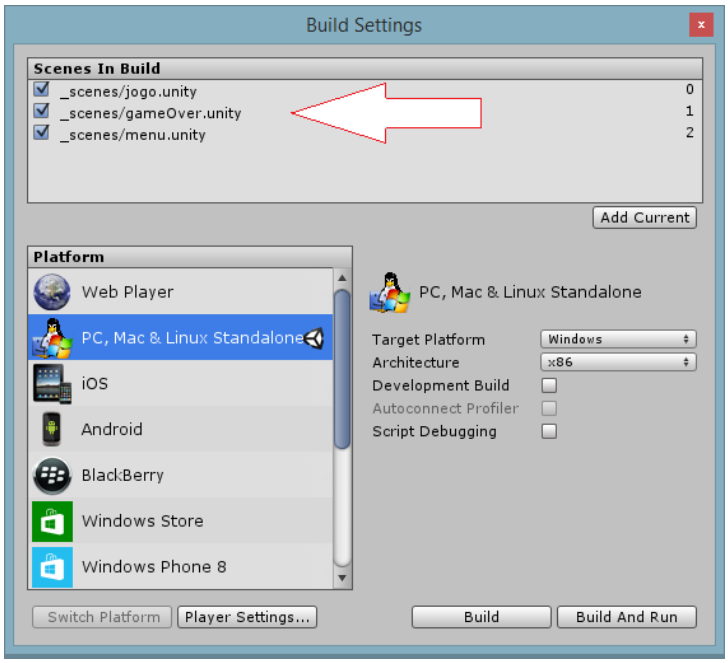
16. Agora abra a cena “*gameOver*” e configure-a como exibido na figura.

17. Crie um script chamado *GameOverScript* e insira-o na *Main Camera* desta cena. Escreva nele o seguinte código:

```
//referencia para o objeto Text
public var  pontosUI: UnityEngine.UI.Text;

function Start () {
    //exibe os pontos que foram armazenados no PlayerPrefs
    pontosUI.text = "Pontos: " + PlayerPrefs.GetInt("pontos");
}

function Reiniciar(){
    //reinicia o jogo
    Application.LoadLevel("jogo");
}
```

18. Associe o *Text* de pontos ao script. Configure o botão de forma que ao ser clicado seja chamada a função *Reiniciar()*.

19. Abra as configurações de *Build* (menu *File->Build Settings*) e insira todas as cenas dentro do campo "*Scenes In Build*". Isto é necessário para que o carregamento das cenas, através de *Application.LoadLevel()*, funcione corretamente. Depois apenas feche a tela, pois não é necessário "*fazer o build*" do jogo.



20. Agora abra a cena "*menu*" e execute o seu jogo.