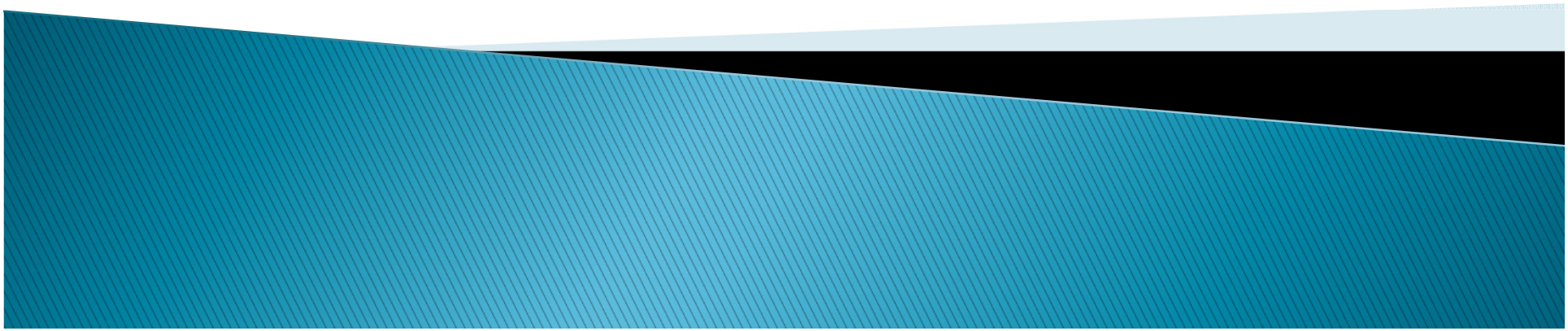
Unity Scripting III

Esteban Walter Gonzalez Clua
Instituto de Computação – UFF
esteban@ic.uff.br



Exemplo 3 – FPS

Abrir Projeto FPS

1 – Abrir Projeto FPS

2 – Tirar Tudo da área de Hierarquia

3 – Carregar modelagem de cena:
Objects/ mainLevelMesh/mainLevelMesh → mainLevelMesh

4 – Certificar-se que a opção “Meshes have colliders” está habilitada

Adicionar camera de FPS

1 – Acrescentar Standard Assets →Prefabs → First Person Controller

2 – Executar

Adicionar uma arma

[Game Objects são compostos por Componentes]

1 - Game Object->Create Empty

2 – Nomear para bazuca (este será o placeholder para a bazuca em si)

3 – Selecionar a camera e centralizar na tela

4 – Selecionar a Bazuca e usar o comando:
Game Object → Move to view

5 – Posicionar a Bazuca próximo ao que seria a mão

6 – Fazer a Bazuca filho da camera, na hierarquia

7 - Executar

Criar objeto Missil

[O que são Prefabs?]

1 – Assets→Create →Prefab e nomear para missil

2 – Criar o missil: GameObject→ Create Other→ Sphere

3 – Arrastar a esfera da visão de hierarquia para a de projeto, sobre o prefab missil

4 – apagar a esfera da hierarquia

Código para a Bazuca

1 – Assets→Create →Java Script

2 – Renomear para lancadorMissil

3 – Criar uma pasta em projeto chamada Scripts de armas

[Edit→Preferences then selecting the External Script Editor box]

4 – Mover o prefab Missil para a pasta Scripts de armas

Código para a Bazuca

5 – Transcrever o seguinte código:

```
var projectile : Rigidbody;
var speed = 20;

function Update ()
{
        if (Input.GetButtonDown ("Fire1"))
        {
        var instantiatedProjectile : Rigidbody = Instantiate
        (projectile, transform.position,transform.rotation);

        instantiatedProjectile.velocity =
        transform.TransformDirection(Vector3 (0,0,speed));

        Physics.IgnoreCollision(instantiatedProjectile.collider,
        transform.root.collider);
        }
}
```

O que faz o código?

```
var projectile : Rigidbody;
```

Ao declarar como global pode-se definir este objeto pelo GUI da Unity

O que faz o código?

```
if (Input.GetButtonDown ("Fire1"))
```

Detecta quando o botão Fire1 foi pressionado. Este botão está mapeado em Edit→ Project Settings → Input

O que faz o código?

```
var instantiatedProjectile : Rigidbody = Instantiate
(projectile, transform.position,transform.rotation);
```

Cria uma instância do tipo projétil. A função Instantiate cria uma instância de um objeto definido pelo primeiro parametro. O segundo e terceiro parâmetro definem a translação e a rotação.

Tipo RigidBody possui atributos físicos

Transform.position e transform.rotation correspondem a atributos do objeto que possui o script, no caso a bazuca, portanto o projetil iniciará da mesma posicao da bazuca

O que faz o código?

```
instantiatedProjectile.velocity =
transform.TransformDirection(Vector3 (0,0,speed));
```

Atribui-se uma velocidade ao projétil. A velocidade está para o eixo Z, ver no game object porque.

Velocity é um atributo para todos os Rigidbody

Para atribuir a velocidade ela deve estar em coordenadas globais (há duas coordenadas: globais e locais). Neste caso, a função TransformDirection converte o vetor local em global

O que faz o código?

```
Physics.IgnoreCollision(instantiatedProjectile.collider,
transform.root.collider);
```

Desabilita a colisão

Código para a Bazuca

1 – Arrastar o script para a bazuca, na hierarquia

2 – É necessário associar um projetil para o script.

3 – Colocar o componente de Rigidbody ao missil:
Components→Physics→RigidBody

4 – Selecionar a bazuca e no atributo Projectil arrastar o missil, que está na área de projeto

5 – Executar...

Explosão para o Tiro...

1 – Criar um novo Script: Assets→Create →Java Script

2 – Renomar para projetil

3 – Colocar no folder de armas

Código para a Bazuca

4 – Transcrever o seguinte código:

```
var explosion : GameObject;

function OnCollisionEnter (collision : Collision)
{
        var contact : ContactPoint = collision.contacts[0];
        var rotation = Quaternion.FromToRotation(Vector3.up,
                contact.normal);

        var instantiatedExplosion : GameObject = Instantiate
                (explosion, contact.point,rotation);

        Destroy(gameObject);

}
```

O que faz o código?

```
function OnCollisionEnter (collision : Collision)
```

Sempre que um Game Object que possui o componente Rigidbody e sofrer colisão irá executar está função.

O que faz o código?

```
var contact : ContactPoint = collision.contacts[0];
```

A classe Collision possui uma série de atributos, uma delas é o ponto de contato onde ocorreu a colisão.

O que faz o código?

```
var rotation = Quaternion.FromToRotation(Vector3.up,
      contact.normal);
```

Queremos copiar o eixo da normal da superficie da colisao para o vetor y (up), de forma que a parte de “cima” da explosão aponte para a mesma direção da normal do choque.

O que faz o código?

```
var instantiatedExplosion : GameObject = Instantiate
        (explosion, contact.point,rotation);
```

Aqui usamos a função Instatiate para instanciar uma explosão, da mesma forma como foi feito anteriormente. Qual a explosão será definido posteriormente pelo usuário.

O que faz o código?

```
Destroy(gameObject);
```

Queremos que o GameObject (missil) desapareça, depois da colisão.

Explosão para o Tiro...

1 – Colocar o script projetil no missil

2 – Criar um sistema de particulas de explosão ou usar um padrão: Standard Assets→Particles→Explosion

3 – Associar a explosão ao missil

Significado para o Tiro...

1 – Criar um novo script e chamar de explosao

Código para a Bazuca

2 – Transcrever o seguinte código:

```
var explosionTime = 1.0;

function Start() {
        Destroy (gameObject, explosionTime);
}
```

Código para a Bazuca

2 – Transcrever o seguinte código:

```
var explosionTime = 1.0;

function Start() {
        Destroy (gameObject, explosionTime);
}
```

A função start é executada sempre que um game object é instanciado

Aqui o objeto será destruido após a passagem de um tempo especificado.

Código para a Bazuca

3 – Selecionar o game object explosão e associar o script de explosao ao mesmo:
Components→Scripts→Explosion

GUIs - HUD

1 – Selecionar uma imagem que sirva de alvo, por exemplo: Textures→aim

2 – Criar um Game Object para a textura em questão: GameObject→Create Other →GUI Texture

3 – Fazer os ajustes necessários.

Som na caixa...

1 – Selecionar o Game Object Explosion

2 – Acrescentar o componente Component→Audio→Audio Source

3 – Colocar algum arquivo de áudio, por exemplo:
Sounds→RocketLauncherImpact

Um pouco de Física

Diferenças entre as funções:

Update() – DeltaTime variável
FixedUpdate() – DeltaTime fixo, definido em Edit→Project Settings→Time

Um pouco de Física

Pretendemos agora:

-Atirar um missil

- Verificar se houve choque e se há ao redor outros rigidbodies

- Transmitir para cada rigidbody a força da explosão

Física básica

```
var explosionTime = 1.0;
var explosionRadius = 5.0;
var explosionPower = 2000.0;

function Start ()
{
        Destroy (gameObject, explosionTime);
        var colliders : Collider[] = Physics.OverlapSphere(transform.position,explosionRadius);

        for (var hit in colliders) {
                if (hit.rigidbody)
                {
                        hit.rigidbody.AddExplosionForce(explosionPower,
                                transform.position, explosionRadius);
                }
        }
        if (particleEmitter)
        {
                particleEmitter.emit = true;
                yield WaitForSeconds(0.5);
                particleEmitter.emit = false;
        }
}
```

Física

```
var colliders : Collider[] = Physics.OverlapSphere(transform.position,explosionRadius);
```

Esta chamada retorna uma lista de todos os objetos de colisão que ocorrem dentro de uma esfera centrada em position (do objeto instanciado) a com raio explosionRadius

Física

```
for (var hit in colliders) {
                        if (hit.rigidbody)
```

Para cada rigidbody da lista de colisão...

Física

```
hit.rigidbody.AddExplosionForce(explosionPower,  transform.position, explosionRadius);
```

Aplica-se uma força para cada rigidbody com intensidade explosionPower, na posição onde o missil atingiu, com intensidade proporcional a distância.

Física

```
if (particleEmitter)
        {
                particleEmitter.emit = true;
                yield WaitForSeconds(0.5);
                particleEmitter.emit = false;
        }
}
```

Caso o objeto instanciado seja um emissor de particulas, ativa-se o mesmo e deixa-se por 0,5 segundos.

Conceito de **Yeld**

Mutliplas Armas

1 – Criar um Game Object vazio e chamar de Armas

2 – Fazer Armas filho da camera FPS

3 – Associar o script de múltiplas armas a este Game Object

4 – O código abaixo garante que a arma não faça disparos mais rápidos que o reloadTime e controla o número de tiros.

Multiplas Armas

```
function Awake () {
        SelectWeapon(0);
}

function Update () {
        if (Input.GetButton ("Fire1"))
                BroadcastMessage("Fire");
        if (Input.GetKeyDown("1"))
        {
                SelectWeapon(0);
        }
        else if (Input.GetKeyDown("2"))
        {
                SelectWeapon(1);
        }
}
```

Multiplas Armas

```
function SelectWeapon (index : int) {
        for (var i=0;i<transform.childCount;i++)
        {
                if (i == index)
                        transform.GetChild(i).gameObject.SetActiveRecursively(true);
                // Deactivate all other weapons
                else
                transform.GetChild(i).gameObject.SetActiveRecursively(false);
        }
}
```

Multiplas Armas

```
function Awake () {
        SelectWeapon(0);
}
```

Arma 0 é definido como a arma padrão

Multiplas Armas

```
function Update () {
        if (Input.GetButton ("Fire1"))
                BroadcastMessage("Fire");
        if (Input.GetKeyDown("1"))
        {
                SelectWeapon(0);
        }
        else if (Input.GetKeyDown("2"))
        {
                SelectWeapon(1);
        }
}
```

Leitura do teclado e seleciona a arma

Multiplas Armas

```
function SelectWeapon (index : int) {
        for (var i=0;i<transform.childCount;i++)
        {
                if (i == index)

        transform.GetChild(i).gameObject.SetActiveRecursively(true);
                // Deactivate all other weapons
                else
                transform.GetChild(i).gameObject.SetActiveRecursively(false);
        }
}
```

Ativa propriamente a arma desejada

Rocket Launcher

1 – Criar um Game Object vazio e chamar de RocketLauncher

2 – posicionar adequdamente, de acordo com o player

3 – fazer o RocketLauncher Filho de Armas

4 – Associar ao RocketLauncher o modelo Objects/weapson/rocketLauncher.

5 – Ajustar a posicao do modelo adequadamente

6 – Adicionar o seguinte script ao RocketLauncher Game Object

Rocket Launcher

```
var projectile : Rigidbody;
var initialSpeed = 20.0;
var reloadTime = 0.5;
var ammoCount = 20;
private var lastShot = -10.0;
function Fire ()
{
        // Did the time exceed the reload time?
        if (Time.time > reloadTime + lastShot && ammoCount > 0)
        {
        // create a new projectile, use the same position and rotation as
the
        // Launcher.
                var instantiatedProjectile : Rigidbody = Instantiate
(projectile,
                        transform.position, transform.rotation);
```

Rocket Launcher

```
// Give it an initial forward velocity. The direction is along the z-axis of
// the missile launcher's transform.
instantiatedProjectile.velocity = transform.TransformDirection(Vector3 (0,
0,
initialSpeed));

// Ignore collisions between the missile and the character controller

Physics.IgnoreCollision(instantiatedProjectile.collider,
transform.root.collider);
lastShot = Time.time;
ammoCount--;
}
}
```

Rocket

1 – Arrastar o Modelo Objects/weapons/rocket para a cena

2 – Colocar o seguinte script para o Rocket:

Rocket

```
// The reference to the explosion prefab
var explosion : GameObject;
var timeOut = 3.0;

// Kill the rocket after a while automatically
function Start () {
        Invoke("Kill", timeOut);
}

function OnCollisionEnter (collision : Collision) {
        // Instantiate explosion at the impact point and rotate the explosion
        // so that the y-axis faces along the surface normal
        var contact : ContactPoint = collision.contacts[0];
        var rotation = Quaternion.FromToRotation(Vector3.up,
contact.normal);
        Instantiate (explosion, contact.point, rotation);
        // And kill ourselves
        Kill ();
}
```

Rocket

```
function Kill ()
{
	// Se houver um filho que é emissor de particula, manda parar
	var emitter : ParticleEmitter=
GetComponentInChildren(ParticleEmitter);
	if (emitter)
		emitter.emit = false;
	// É necessário destacar os filhos (no caso o sistema de
particulas)
	// para que não desapareçam quando o projetil desaparecer
	transform.DetachChildren();

	// Destroi o projetil
	Destroy(gameObject);
}
// Para garantir que deve estar presente o componente de Rigidbody
@script RequireComponent (Rigidbody)
```

Rocket

3 – Criar Um Box Collider para o Rocket e fazer este box maior que o rocket em si

Fumaça do Foguete

1 – Criar um Game Object->Create Other->Particle System

2 – Colocar as segiuntes configurações:

Ellipsoid x, y, z = 0,1
Rnd Velocity = 0,1 (3 eixos)
Particle Emitter min e max size = 0,5
number of particles emitted = 100 (max e min)
Arrastar Particle Effects> Smoke para o sistema
Particle Animator section → cada eixo = 0,5
size grow= 3
habilitar autodestruct

3 – Fazer o sistema de particulas filho do Rocket

4 – Pocisionar adequadamente na parte de traz do foguete

Criar Prefab do Rocket

1 – Criar um prefab vazio e arrastar o Rocket para ele

Explosao ao foguete

1 – Arrastar o Standard Assets> Particles > small Explosion para o campo de explosão do Rocket

2 – Criar o segiunte Script para a explosão:

Explosão

```
var explosionRadius = 5.0;
var explosionPower = 10.0;
var explosionDamage = 100.0;
var explosionTime = 1.0;

function Start () {
        var explosionPosition = transform.position;

        // Busca a lista de objetos que caem dentro do raio de colisão
        var colliders : Collider[] = Physics.OverlapSphere (explosionPosition,
                                                    explosionRadius);
        for (var hit in colliders) {
                if (!hit)
                continue;
```

Explosão

```
// Aplica uma força para estes objetos…

if (hit.rigidbody)
{
        hit.rigidbody.AddExplosionForce(explosionPower, explosionPosition,
                                        explosionRadius, 3.0);
        var closestPoint = hit.rigidbody.ClosestPointOnBounds(explosionPosition);
        var distance = Vector3.Distance(closestPoint, explosionPosition);

        // The hit points we apply decrease with distance from the hit point
        var hitPoints = 1.0 - Mathf.Clamp01(distance / explosionRadius);
        hitPoints *= explosionDamage;

        // Avisa ao Rigidbody em questão quanto dano sofrerá
        hit.rigidbody.SendMessageUpwards("ApplyDamage", hitPoints,
        SendMessageOptions.DontRequireReceiver);
}

}
```

Explosão

```
// stop emitting ?
if (particleEmitter) {
        particleEmitter.emit = true;
        yield WaitForSeconds(0.5);
        particleEmitter.emit = false;
}

// destroy the explosion
Destroy (gameObject, explosionTime);
}
```

Machine Gun

1 – Criar Game Object vazio e colocar sobre as Armas

2 – Arrastar o Modelo Objects > Weapons > Machine Gun para este Game Object

3 – Associar o Script de Machine Gun a este Game Object:

Machine Gun

```
var range = 100.0;
var fireRate = 0.05;
var force = 10.0;
var damage = 5.0;
var bulletsPerClip = 40;
var clips = 20;
var reloadTime = 0.5;
private var hitParticles : ParticleEmitter;
var muzzleFlash : Renderer;
private var bulletsLeft : int = 0;
private var nextFireTime = 0.0;
private var m_LastFrameShot = -1;
```

Machine Gun

```
// Desliga as particulas

function Start ()
{
	hitParticles = GetComponentInChildren(ParticleEmitter);
	// We don't want to emit particles all the time, only when we hit something.
	if (hitParticles)
		hitParticles.emit = false;
	bulletsLeft = bulletsPerClip;
}
```

Machine Gun

```
// Late Update é executado depois do Update. No caso, o Update está em Armas

function LateUpdate()
{
        if (muzzleFlash)
        {
                // We shot this frame, enable the muzzle flash
                if (m_LastFrameShot == Time.frameCount)
                {
                        muzzleFlash.transform.localRotation = Quaternion.AxisAngle
                                (Vector3.forward, Random.value);
                        muzzleFlash.enabled = true;
                }

                // We didn't, disable the muzzle flash
                else
                {
                        muzzleFlash.enabled = false;
                        enabled = false;
                }
        }
}
```

Machine Gun

```
// Calcula se pode atirar, baseado na taxa de tiros e numero de tiros
function Fire ()
{
	if (bulletsLeft == 0)
		return;
	// If there is more than one bullet between the last and this frame
	// Reset the nextFireTime

	if (Time.time - fireRate > nextFireTime)
		nextFireTime = Time.time - Time.deltaTime;
	// Keep firing until we used up the fire time
	while( nextFireTime < Time.time && bulletsLeft != 0)
	{
		FireOneShot();
		nextFireTime += fireRate;
	}
}
```

Machine Gun

```
// Verifica se o tiro atinge um rigidbody e inflinge um dano e uma força
function FireOneShot ()
{
	var direction = transform.TransformDirection(Vector3.forward);
	var hit : RaycastHit;

	// Did we hit anything?
	if (Physics.Raycast (transform.position, direction, hit, range))
	{
		// Apply a force to the rigidbody we hit
		if (hit.rigidbody)
			hit.rigidbody.AddForceAtPosition(force * direction, hit.point);
		// Place the particle system for spawing out of place where we hit the
		// surface! And spawn a couple of particles

		if (hitParticles)
		{
			hitParticles.transform.position = hit.point;
			hitParticles.transform.rotation =
				Quaternion.FromToRotation (Vector3.up,hit.normal);
			hitParticles.Emit();
		}
```

Machine Gun

```
function Reload () {
        // Wait for reload time first - then add more bullets!
        yield WaitForSeconds(reloadTime);
        // We have a clip left reload
        if (clips > 0)
        {
                clips--;
                bulletsLeft = bulletsPerClip;
        }
}

function GetBulletsLeft () {
        return bulletsLeft;
}
```

Machine Gun

```
                // Send a damage message to the hit object
                hit.collider.SendMessageUpwards("ApplyDamage", damage,
                SendMessageOptions.DontRequireReceiver);
}

bulletsLeft--;
// Register that we shot this frame,
//// fsor amtehat the LateUpdate function enabled the muzzleflash renderer for one

m_LastFrameShot = Time.frameCount;
enabled = true;

// Reload gun in reload Time
if (bulletsLeft == 0)
                StartCoroutine("Reload");
}
```

Machine Gun

4 – Associar o Muzzle Flash (filho do Machine Gun) ao parametro do script.

5 – Associar o prefab Standard Assets > Particles > Sparks como filho do machineGun

Hit Points

1 – Criar o seguinte Script:

Hit Point

```
var hitPoints = 100.0;
var detonationDelay = 0.0;
var explosion : Transform;
var deadReplacement : Rigidbody;

// Aplica o dano
function ApplyDamage (damage : float)
{
        // We already have less than 0 hitpoints, maybe we got killed already?
        if (hitPoints <= 0.0)
                return;
        hitPoints -= damage;
        if (hitPoints <= 0.0)
                Invoke("DelayedDetonate", detonationDelay);
}

function DelayedDetonate ()
{
        BroadcastMessage ("Detonate");
}
```

Hit Point

```
function Detonate ()
{
        // Destroy ourselves
        Destroy(gameObject);
        // Create the explosion
        if (explosion)
                Instantiate (explosion, transform.position, transform.rotation);
        // If we have a dead barrel then replace ourselves with it!

        if (deadReplacement)
        {
                var dead : Rigidbody = Instantiate(deadReplacement, transform.position,
                                                transform.rotation);
                // For better effect we assign the same velocity to the exploded barrel
                dead.rigidbody.velocity = rigidbody.velocity;
                dead.angularVelocity = rigidbody.angularVelocity;
        }
}

// We require the barrel to be a rigidbody, so that it can do nice physics
@script RequireComponent (Rigidbody)
```

Hit Points

1 – Arrastar a cena Objects>CrateAdnBarrel>Barrel

2 – Adicionar componente de Rigidbody e Box Collider

3 – Associar o script anterior (de destruição)

4 – Associar uma explosão ao parametro explosion

5 – Criar um prefab Barril e colocar o barril anterior sobre este

6 – Tirar o Barril original e colocar diversos barris na cena, a partir do prefab Barril

Barril Velho

1 – Criar outro prefab Chamado Barril Velho

2 – Associar o mesmo modelo de barril Objects>crateAndBarrel>Barrel

3 – Selecionar o Barril Velho no prefab

4 – Selecionar Mesh Renderer > Element 0 > barrel1

5 – Duplicar o material (CTRL + D) e renomear para barril_envelhecido

6 – Modificar as cores e parametros deste novo material

7 – Associar este novo material ao prefab barrilvelho

8 – Adicionar Rigidbody e Boxcollider ao barril velho

9 – Associar este barril ao parametro dead replacement do Barril

SkyBox

1 – Criar ou editar um skybox

2 – Selecionar Edit → Rendering Settings → Skybox e arrastar o skybox

Arma Automática

1 – Arrastar o Object > Weapopns > SentryGun para a cena

2 – Acrescentar componentes de rigidbody e Box Collider. Fazer o Box Collider alto, para que desequilibre com facilidade.

3 – Associar o script de receber danos (Hit Point) a arma

4 – Associar uma explosao a variavel explode

5 – Associar o seguinte script ao sentryGunRotateY :

Arma Automática

```
var attackRange = 20.0;
var target : Transform;

// Verifica se há um alvo associado a arma
function Start () {
        if (target == null && GameObject.FindWithTag("Player"))
                target = GameObject.FindWithTag("Player").transform;
}
```

Arma Automática

```
function Update () {
        if (target == null)
                return;
        if (!CanSeeTarget ())
                return;
        // Rotaciona para apontar ao player
        var targetPoint = target.position;
        var targetRotation = Quaternion.LookRotation
                (targetPoint -transform.position,Vector3.up);
        transform.rotation = Quaternion.Slerp(transform.rotation,
                                                targetRotation,Time.deltaTime *
2.0);

// Se o ângulo for menor que 10 graus, começa a atirar…
var forward = transform.TransformDirection(Vector3.forward);
var targetDir = target.position - transform.position;
if (Vector3.AngleBetween(forward, targetDir) * Mathf.Rad2Deg < 10.0)
        SendMessage("Fire");
}
```

Arma Automática

```
// Função que verifica  se está na mira
function CanSeeTarget () : boolean
{
        if (Vector3.Distance(transform.position, target.position) >
attackRange)
                return false;
        var hit : RaycastHit;
        if (Physics.Linecast (transform.position, target.position, hit))
                return hit.transform == target;
        return false;
}
```

Arma Automática

1 – Arrastar o player para o target

2 – Associar alguma explosão ao explosion do SentryGun

3 – Associar algum modelo ao dead replacement

4 – Associar o MachineGun script ao sentryGunRotateY

5 – Associar ao parametro Muzzle Flash o muzzle flash filho do Sentry Gun

6 – Selecionar o Muzzle Flash e ativar o shader Particle > Additive

Waypoints

1 – Criar um Game Object vazio

2 – Nomeá-lo para waypoint

3 – Associá-lo com o script weaponscripts > autowaypoint

4 – Duplicar mais 2 waypoints e criar um triangulo

5 – Clicar em update waypoint (direita do script) e verificar se o trajeto está “limpo”

Robos com IA

1 – Arrastar para a cena o robotartwork > Robot para a cena e posicionar acima do chao

2 – Colocar ao robo os scripts weaponsScripts > AI e AI Animation

3 – Modificar os parametros ca capsula envolvente (componente Character Controler) para que fique maior que o robo

4 – Colocar no alvo da IA o Player

Tiro do Robo

1 – Criar um game object vazio e chamar de gun_spawn

2 – Colcar como filho do robo

3 – Resetar sua posição (transorm > reset)

4 – Mover em Z para que fique ligeiramente a frente

5 – Adicionar o Rocket Launcher Script ao gun_spawn e o rocket ao parametro projectile

6 – Criar um prefab para este robo e apagar o original

7 – Criar um inimigo baseado no prefab

Dano ao Robo

1 – Ver qual a explosao associada ao Rocket que está sendo usado pelo player

2 – Selecionar esta explosao e trocar o scritp dela pelo script Explosion Advanced

3 – Colocar ao Robo o script CharacterDamage

Ragdolls

1 – Salvar a cena (não o projeto)

2 – Criar uma nova cena e chamar de ragdoll

3 – Criar e escalar um cubo para ser um piso

4 – Arrastar o robo para a cena e posicionar acima do cubo

5 – Remover o componente de animação do robo

6 – Associar um Ragdoll: Game Object → Create Other → Ragdoll

Ragdolls

7 – Fazer as seguintes associações:
rootHandle > root
Upleg_L > Left Hip
Lowleg_L > Left Knee
Heel_L > Left Foot
[Idem para perna direita]

upArm_L > Left Arm
Elbow_L > Left Elbow
upArm_R > Right Arm
Elbow_R > Right Elbow
Spine3 > Middle Spine
Head > Head

Mass = 4

Ragdolls

8 – Clicar em create

9 – Criar um prefab roboMorto e associar o robo com ragdoll a ele

10 – Abrir novamente a fase e no campo dead replacement do robo usar o roboMorto

Sons

1 – Arrastar o Audio clip machineGunSingleShotLoopable para o gameobject machine Gun

2 – Desabilitar Play on Awake

3 – Selecionar o missil que é usado no Rocket Launcher e arrastar o audio clip RocketLAuncherFire

4 – Ajustar o rolloff value, para que o som decaia com mais força, na medida que o missil se afasta

GUI

1 – Selecionar a textura GUI > GUIOverlay

2 – Criar um Game Object the Gui: Game Object > Create Other > GUI Texture

3 – Colocar os seguintes parametros:
Transform: 0, 0, 0
Pixe Insert: 0, 0, 256, 128

GUI

1 – Selecionar a textura GUI > Health

2 – Criar um Game Object the Gui: Game Object > Create Other > GUI Texture

3 – Colocar os seguintes parametros:
Transform: 0, 0, 0
Pixe Insert: 37, 83, 148, 112.

GUI

1 – Selecionar game object -> create other -> Text

2 – Colocar os seguintes parametros:
Transform: 0, 0
pixel offset : 188,109

GUI

1 – Selecionar GUIRocket

2 – Criar um Game Object the Gui: Game Object > Create Other > GUI Texture

Adicionar o FPS Player ao Player Controller e preencher os campos de GUIs

Saúde

1 – Selecionar a textura GUI > healthbar

2 – Criar um Game Object: game object > Create Other > GUI Texture

3 – colocar os valores: transform: 0, 0, 0 e pixel insert: 37, 83, 148, 33

Machine Gun

1 – Criar um Game Object > Create other> GUI Text

2 – colocar os valores: transform: 0, 0, 0 e pixel offset 188, 109

Mais sons

1 – Colocar os sons que faltam ao script de FPS Player:

moanOfPainSmall, moanOfPainBig, playerDie

2 – Colocar 5 sons para passos: walk sound > Size = 5 e usar os 5 sons existentes

Luzes

1 – Criar uma luz direcional

2 – Ir para Edit > Project Settings > Tags

3 – Clicar na Layer 8 e escrever Lightmapped

4 – No culling da Luz direcional excluir o Layer Lightmapped

5 – Selecionar o mainLevelMesh e colocá-lo no Layer Lightmapped

+power.soft10

Tutorial de Corrida

1 – Abrir projeto Car
2 – Abrir cena “The Track”
3 – Colocar o carro (Models/Car/catamount) na área de projeto.
4 – Renomear para carro e colocar em algum lugar da pista
5 – Analisar a composição do carro
6 – No inspector do carro, colocar o carro na layer car
7 – Remover o componente de animação do carro

Colisão do carro

1 – Selecionar a sub-malha Collider Bottom, do carro
2 – Adicionar um mesh collider a este componente
3 – Selecionar o material car para este collider
4 – habilitar os parametros: Smooth Sphere Collisions e Convex
5 – Tirar o MeshRenderer e o Mesh Filter, para que não seja visivel
6 – Fazer o mesmo com Collider_Top

Sombras do carro

1 – Configurar o cast Shadows e Receive Shadows

Física do carro

1 – Atribuir Rigidbody ao carro
2 – Ajustar massa do carro
3 – Associar o script do carro
4 – Colocar em 2 rodas para a frente e 2 para trás
5 – Associar o DiscBrake do modelo para cada parametro

Blob do carro

1 – Criar um GameObject vazio e colocar sobre o carro (hierarquia)
2 – Adicionar o components -> Rendering -> Projector
3 – Near Clip Plane = 0.1, Far Plane = 50 e Field of View = 30
4 – Colocar o Material Blob_Shadow
5 – Ignorar everything e depois incluir apenas a estrada
6 – Associar o script BlobShadowController
[analisar script]

Marcas do Pneu, som e efeitos visuais

1 – Colocar na cena o Prefabs -> VFX-> Sidemarks
2 – Associar o Script de som do carro ao carro, bem como os áudio correspondentes
3 – Associar os seguintes scripts ao carro: LightmapperObjectUV, CrashController, Generate2DReflections

Camera seguir o carro

1 – Colocar target da main camera como sendo o carro

Centro de Massa

1 – Criar um GameObject vazio e chamar de centro de massa
2 – Associar como filho do carro
3 – Ressetar suas transformacoes
4 – Posicionar em algum lugar diferente
5 – Colocar como parametro do script

Criar um Prefab do Carro

1 – Criar um prefab vazio e arrastar o carro para o mesmo
2 – Selecionar os seguintes elementos:
Prefabs/VFX: Directional_Light_Car_Road, Main_Camera, Skidmarks
cripts/Javascripts: SoundToggler.js
Pro Standard Assets/Image Based: ImageEffects.cs e ImageEffectsBase.cs

3 – Exportar e criar um novo projeto usando o mesmo carro

Animação em Malhas

1 – Importar um modelo 3D como animação
2 – Em Split Animation, nomear cada sequencia
Warp mode: Loop ou uma única vez
Loop: copia o primeiro frame para o final
Ping Pong: vai e vem...

Player Controller completo

1 – Criar um script chamado PlayerController
2 – Criar um Asset Player
3 – Criar um Character Controller

Player Controller

```
var rollSpeed = 6.0;
var fastRollSpeed = 2.0;
var jumpSpeed = 8.0;
var gravity = 20.0;
var rotateSpeed = 4.0;
var duckSpeed = .5;

private var moveDirection = Vector3.zero;
private var grounded : boolean = false;
private var moveHorz = 0.0;
private var normalHeight = 2.0;
private var duckHeight = 1.0;
private var rotateDirection = Vector3.zero;

var isControllable : boolean = true;

var controller : CharacterController ;
controller = GetComponent(CharacterController);
```

Player Controller

```
function FixedUpdate() {
        if(!isControllable)
                Input.ResetInputAxes();
        else{
```

Player Controller

```
        if (grounded) {

                moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0,
xis("Vertical"));
                moveDirection = transform.TransformDirection(moveDirection);
                moveDirection *= rollSpeed;

                moveHorz = Input.GetAxis("Horizontal");
                if (moveHorz > 0)                                   //right turn
                        rotateDirection = new Vector3(0, 1, 0);
                else if (moveHorz < 0)                              //left turn
                        rotateDirection = new Vector3(0, -1, 0);
                else                                                //not turning
                        rotateDirection = new Vector3 (0, 0, 0);


        }
```

Player Controller

```
moveDirection.y -= gravity * Time.deltaTime;
```

Player Controller

```
                var flags = controller.Move(moveDirection * Time.deltaTime);
                controller.transform.Rotate(rotateDirection * Time.deltaTime, rotateSpeed);
                grounded = ((flags & CollisionFlags.CollidedBelow) != 0 );
                }
}
```

Player Controller

@script AddComponentMenu("Player/Widget'sController")

Exemplo 3 – Movimento de camera

Movimento de Camera

```
var target : Transform;
var distance = 10.0;
var height = 5.0;

var heightDamping = 2.0;
var rotationDamping = 3.0;
var distanceDampingX = 0.5;
var distanceDampingZ = 0.2;

var camSpeed = 2.0;
var smoothed = true;
```

Movimento de Camera

```
function LateUpdate () {

        wantedRotationAngle = target.eulerAngles.y;
        wantedHeight = target.position.y + height;
        wantedDistanceZ = target.position.z - distance;
        wantedDistanceX = target.position.x - distance;

        currentRotationAngle = transform.eulerAngles.y;
        currentHeight = transform.position.y;
        currentDistanceZ = transform.position.z;
        currentDistanceX = transform.position.x;
```

Movimento de Camera

```
currentRotationAngle = Mathf.LerpAngle (currentRotationAngle, wantedRotationAngle, rotationDamping * Time.deltaTime);


currentHeight = Mathf.Lerp (currentHeight, wantedHeight, heightDamping * Time.deltaTime);
currentDistanceZ = Mathf.Lerp(currentDistanceZ, wantedDistanceZ, distanceDampingZ * Time.deltaTime);
currentDistanceX = Mathf.Lerp(currentDistanceX, wantedDistanceX, distanceDampingX * Time.deltaTime);

currentRotation = Quaternion.Euler (0, currentRotationAngle, 0);
```

Movimento de Camera

```
            transform.position -= currentRotation * Vector3.forward * distance ;
            transform.position.x = currentDistanceX;
            transform.position.z = currentDistanceZ;
            transform.position.y = currentHeight;

            transform.LookAt(target);
}
```

Colocando Movimento Suave

```
// Substituir a última linha pela função LookAtMe:

function LookAtMe(){
		if(smoothed)
		 {
				var camRotation = Quaternion.LookRotation(target.position - transform.position);
				transform.rotation = Quaternion.Slerp(transform.rotation, camRotation, Time.deltaTime *
camSpeed);
		}
		else{
				transform.LookAt(target);
		}
}
```

Status do Player (player_status.js)

```
// Energia e Vida
var health: float = 10.0;
var maxHealth: float= 10.0;
var energy: float = 10.0;
var maxEnergy: float = 10.0;
var energyUsageForTransform: float = 3.0;
var widgetBoostUsage :float = 5.0;

//Sound Effects----------------------------------------------------------------
var hitSound: AudioClip;
var deathSound: AudioClip;


//link com o player controller
playerController = GetComponent(player_controller) ;
var controller : CharacterController;
controller = GetComponent(CharacterController);
```

Status do Player (player_status.js)

```
//Helper Controller Functions----------------------------------------------------
function ApplyDamage(damage: float){

		health -= damage;

		//play hit sound if it exists
		if(hitSound){


		}
		//check health and call Die if need to
		if(health <= 0){
				health = 0; //for GUI
				Die();
		}

}
```

Status do Player (player_status.js)

```
function AddHealth(boost: float){
		//add health and set to min of (current health+boost) or health max
		health += boost;
		if(health >= maxHealth){
			health = maxHealth;
		}
		print("added health: " + health);
}

function AddEnergy(boost: float){
		//add energy and set to min of (current en + boost) or en max
		energy += boost;
		if(energy >= maxEnergy){
			energy = maxEnergy;
		}
		print("added energy: " + energy);
}
```

Status do Player (player_status.js)

```
function Die(){
		//play death sound if it exists
		if(deathSound){

		}
		print("dead!");
		HideCharacter();

		yield WaitForSeconds(1);

		//inserir Respawn aqui

		ShowCharacter();
		health = maxHealth;
}

function HideCharacter(){
		GameObject.Find("Body").GetComponent(SkinnedMeshRenderer).enabled = false;
		GameObject.Find("Wheels").GetComponent(SkinnedMeshRenderer).enabled = false;
		playerController.isControllable = false;

}

function ShowCharacter(){
		GameObject.Find("Body").GetComponent(SkinnedMeshRenderer).enabled = true;
		GameObject.Find("Wheels").GetComponent(SkinnedMeshRenderer).enabled = true;
		playerController.isControllable = true;

}
```

Acrescentar ao Player Controller

```
...
moveDirection = transform.TransformDirection(moveDirection);
moveDirection *= rollSpeed;

//novo
var playerStatus : player_status;
playerStatus = GetComponent (player_status);



//novo
controller.height = normalHeight;
controller.center.y = controller.height/2;
```

Acrescentar ao Player Controller

```
//Jump Controls
if (Input.GetButton ("Jump")) {
		moveDirection.y = jumpSpeed;
}

//Apply any Boosted Speed
if(Input.GetButton("Boost")){
		if(playerStatus){
				if(playerStatus.energy > 0)
				{
						moveDirection *= fastRollSpeed;
						playerStatus.energy -=
sage *Time.deltaTime;
				}
		}
}

//Duck the controller
if(Input.GetButton("Duck")){
		controller.height = duckHeight;
		controller.center.y = controller.height/2 + .25;
		moveDirection *= duckSpeed;
}
```

Renomear Fire1 e Fire2 para Boost e Duck

Incluindo a Animação

```
private var nextPlayIdle = 0.0;
var waitTime = 8.0;

function Start ()
{
		animation.Stop();
		animation.Play("idle");

}
```

Incluindo a Animação

```
//Jump Controls
if (Input.GetButton ("Jump")) {
                moveDirection.y = jumpSpeed;
                animation.CrossFade("jump", 0.2);
                nextPlayIdleTime = Time.time + waitTime;
}
```

Incluir em todas as ações:

```
animation.CrossFade ("duck"); nextPlayIdleTime = Time.time + waitTime;

animation.CrossFade (”boost"); nextPlayIdleTime = Time.time + waitTime;
```

Incluindo a Animação

```
// no fim do update:
                    if (Time.time >nextPlayIdle)
                    {
                                nextPlayIdle = Time.time + waitTime;
                                animation.CrossFade("idle");
                    }
```

Animation Editor

Abrir o editor de animação e criar algum movimento para o player

Animation Layers

```
animation[“anda”].layer = 1;
animation[“pula”].layer = 1;
animation.SyncLayers(1);

animation[“abaixa”].layer = 2;
....
```

Triggers

function OnTriggerEnter (collider: Collider)

Triggers

```
function OnTriggerEnter (collider: Collider)
```

```
function OnTriggerEnter (other: Collider)
{
    print("other.name: " + other.name);
    if ( other.name == "porta" )
        other.abre = true;
}
```

BroadcastMessage

BroadcastMessage (”funcao", parametro);

// OBS: somente manda para os filhos

Pegando um item

```
function OnTriggerEnter (other: Collider)
{
		var PlayerStatus : player_controller = other.GetComponent(player_controller);

		 if ( other.name == "widget" )
		 {
				PlayerStatus.armado=true;
				print(PlayerStatus.armado);
		 }
}
```