**Todo objeto agrupador tem que aplicar reset no transform, se isso não for feito tem que tirar todos os objetos de dentro e restar a transform e colocar de volta!**

Game Engine: é um motor de jogo ontem contém toda a parte de desgin gráfico, compiladores para implementação do lógica de programação, compatibilidades com diferentes sistemas operacionais...etc

# Unity:

-Linguages de programação: C#, Javascript, BOO

-Deloy Multiplataforma: desenvolver para windows, linux, mac, android, samsung TV, IOS, Playstation, facebook games, Web...etc

- Unity é degraça se você não tiver fazendo 100k$ por ano.

# - Softwares Para Criar/Projetar representações 3D

## Blender

Software completo gratuito para criar representações 3D.

## MakeHuman

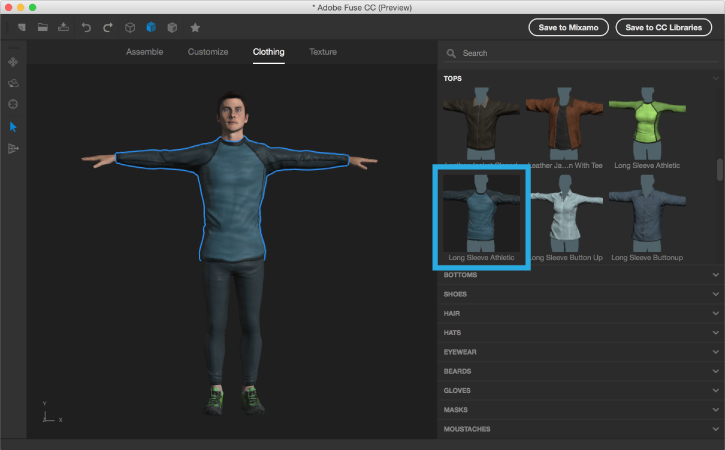
Software Pronto Para Arrastar e criar um humano **(Super Fácil para utilizar)**

## Image result for make human Substance Painter

Software Permite Pintar objeto 3D para criar Texturas e Materiais.

## Adobe Fuse

Software Pronto Para Arrastar e **criar um humano** com muito mais recursos para fazer. **DÁ PARA ARRASTAR E CRIAR ZOMBIES, HUMANOS, MUTANTES...ETC.**



# -Criar novo projeto:

New -> definir Nome, Path, 3D, -> (Optional) Adicionar packages do Unity assets

**-Criar Pastas:**

-Scenes

-Materials

-Scripts

-Editor (se for colocar scripts para rodar somente no unity)

**-Salvar Cenas em Assets/Scenes**

No arquivo do projeto de Scenes -> botão direito -> create -> Scene

# -Interface Unity

**-Scene:** cena ontem podemos visualizer a camera, as luzes e o nosso visual do jogo.

**-Game:** visualização do jogo como produto final.

**-Project tab:** aba que contém todos os arquivos gráficos e scripts do jogo.

**-Hierarchy:** hierarquia dos objetos (objeto dentro do outro, como texto do botão). Serve também para agrupar Objetos dentro de um objeto vazio

**-Inspector:** propridades do objeto selecionado.

**-Console:** local onde podemos ver informações de problemas, avisos e erros de compilação dos gráficos e scripts.

**-Layout default:** Windows -> Layouts -> Default

**-Busca**: Pode procurar por o nome do objeto ou por tipo do objeto no projeto ou no Asset store

# -Cena

É uma fase (level) de um jogo, onde pode ter varias cenas em um único jogo. O menu inicial é um cena propria.Quando aparece a tela de loading é por que está carregando outra cena.

**-Seleção dos objetos:** botão esquerdo para selecionar vários objetos ou clica em um objeto.

**-Mover orientação da camera:** mantém botão direitodo mouse apertado e mover o mouse.

**-Mover na cena:** mantém o scroll apertado e mover o mouse.  
**-Andar na cena:** mantém o botão direito o mouse e apertar w,s,a,d (Modo primeira pessoa), tecla ‘E’ e ‘Q’ sobem e descem a camera.

**-Mover a camera em volta de um centro: alt** + mantém botão esquerdo apertado + mexer com o mouse.

# Objetos da Cena



-**Mover Objeto**: selecionando o objeto apretando nele -> arrastar nos eixos dele (as setas).

-**Rotacionar o Objeto**: Clicar no objeto -> inspector -> transform -> Clicar na letra e arrastar

**-Posição e Escala:** // //

**-Eixos**: Y: Vertical (Cima/Baixo)

X: Horizontal (direita esquera)

Z: Profundidade

**-Perspectiva**: Item em baixo da orientação dos eixos

Isometrica: menos noção da profundidade.

**-Criar Novo Objeto:** Hierarquia -> Botão direito -> Objeto 3D -> Cubo

**-Propridadaes Básicas do Objeto:** Hierarquia -> Botão direito ao objeto -> pode renomear/duplicar/deletar o objeto.

**-Agrupar Objetos:** Hierarquia -> Botão direito -> Create Empty -> Arrastar objetos

-Os objetos agrupados são relátivos ao objeto pai

-Os objetos como cubos podem ser filhos de outos cubos

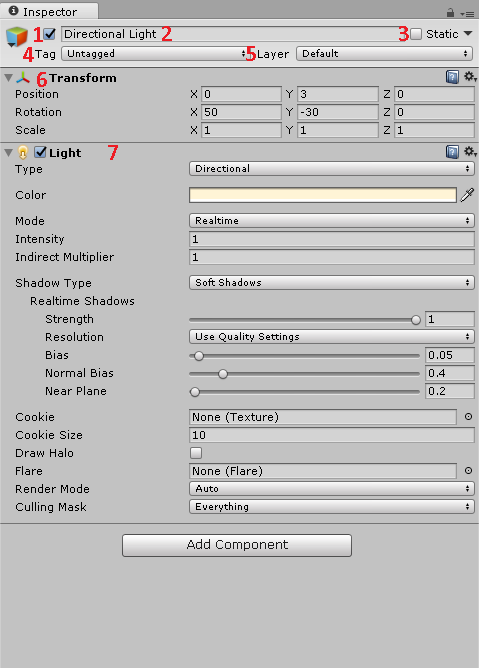
**-Colocar Labels nos objetos e cenas para identificação e agrupamento:**

- No arquivo do projeto -> selecionar objeto -> na imagem do projeto  -> clica e definir label. Onde podem ser procurados pela barra de busca

-As Buscas podem Ser salvas na aba de favoritos 

-**Material do objeto**: Para definir o material do objeto (cor/texture do objeto) podemos arrastar o material em cima do objeto, ou arrastando para cima do Elemento 0 do inspector, ou clicar na bolinha do Element 0 do inspector.

# Inspector



1. Vincular ou desvinular objeto da cena: opção para habilitar ou desabilitar objeto
2. Alterar o Nome do objeto
3. Definir se objeto será estático ou não
4. Tag
5. Layer
6. Transformação: Posição, Rotação e escala do objeto

Rotacionar,posicionar ou escalar o Objeto: Clicar na letra e arrastar

1. Propridades do objeto: como colisão, luz, material...etc

# Caracter

- O caracter tem que ser filho da camera, onde movendo a camera move o caracter.

# Planos e Materiais

-Na pasta de Materiais -> Botão direito -> create -> material -> definir cor.

# Luzes Direcionais

- Não da para mudar a posição ou a escala da luz, só da para rotacionar para simular por de sol

- **diminuir intensidade da sombra**: directional Light -> inspector -> Strength

# Standard Assets

São Bibliotecas que contém recursos gráficos e som e scripts padrão dos caracteres, efeitos de som de passos, Ambientes, Textures, Controles (Joystick Android), veiculos ou outro pacote já baixado da internet....

-Adicionar um character: Assets -> Import Packages -> Character

# Canvas

- Canvas é um espaço abstrato (invisivel ao usuario), onde representa a tela do usuario, onde pode ser de qualquer tamanho dependendo da tela do usuário.Na canvas (tela) é colocado os botões, controles,menus...etc

- A posição dos objetos da Canvas podem ser relátivos a tela, onde pode ser definido na Anchor Present.

**-Como Trabalhar com canvas**:

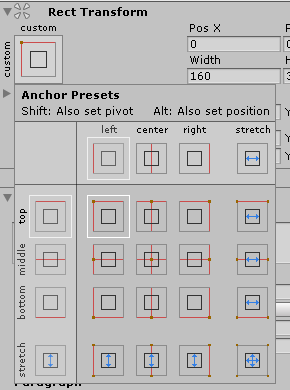
1. Inserir uma interface**(Canvas)** para saber a posição X e Y do usuário, onde que é a tela fixa do usuario, onde podemos colocar o texto do score fixa na tela....

Hierarquia -> Botão direito ->UI -> canvas

-Definir a escala padrão (nas escalas menores que resize automaticamente):Canvas -> Inspector -> Canvas Scaler -> UI scale Mode: Scale with Screen size -> Reference Resolution: 1920 x 1080

-Canvas -> Botao direito -> UI -> Text ->Inspector -> Vertical Overflow: overFlow (**permitir fonte maior que o tamanho da caixa do texto**) -> Horizontal Overflow: Overflow ->Font size: 60

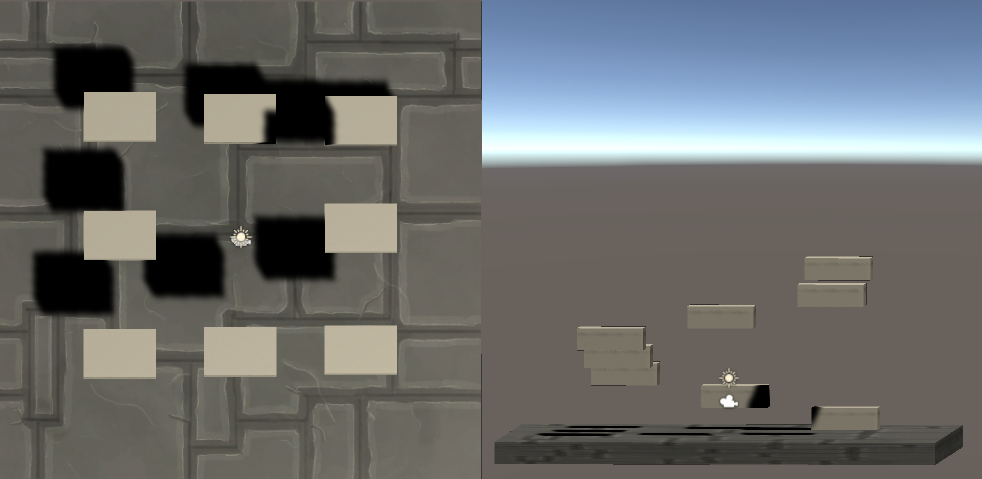
-Colocar o texto em uma posição relativa e fixa ao Canvas (a tela): Segurar Shit -> clicar na posição desejada -> Segurar CTRL -> clicar na mesma posção desejada



- Setar a margim do texto: PosX e PosY

# Criar Simples jogo

1. Criar Chão : Criar um cubo nomear: chão, colocar na posição 0,0,0 e dar 20 de scale no x e z
2. Colocar Material para o Cubo de chão
3. Deletar a Camera principal e adicionar um character(**Uma cena normalmente tem 1 camera**): Assets -> Import Packages -> Character
4. Implementar o caracter: standard Assets -> Characters -> First Person Character ->Prefabs(**objeto e script prefabricado e pronto**) -> RigidBodyFPSController -> Arrastar para Herarquia
5. Criar Objetos da Cena:- Criar Objeto vazio para agrupamento de objetos (objetos de cada andar é igual) e colocar a posição do objeto vazio no 0,0,0

 -Começar criando os cubos tamanho3,1,3 e a cada cubo aumentado o Y de Cada Cubo até formar um quadrado (no posicionamento primeiro aumenta Ye X, depois aumento Y e Z)

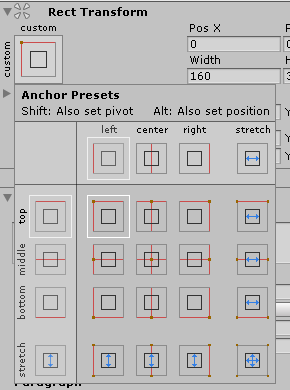
1. Duplicar o agrupamento do andar para formar outro andar posicionando o agrupamento no Y igual da posição Y do ultimo cubo do agrupamento anterior.
2. Inserir uma interface**(Canvas)** para saber a posição X e Y do usuário, onde que é a tela fixa do usuario, onde podemos colocar o texto do score fixa na tela....

Hierarquia -> Botão direito ->UI -> canvas

-Definir a escala padrão (nas escalas menores que resize automaticamente):Canvas -> Inspector -> Canvas Scaler -> UI scale Mode: Scale with Screen size -> Reference Resolution: 1920 x 1080

-Canvas -> Botao direito -> UI -> Text ->Inspector -> Vertical Overflow: overFlow (**permitir fonte maior que o tamanho da caixa do texto**) -> Horizontal Overflow: Overflow ->Font size: 60

-Colocar o texto em uma posição relativa e fixa ao Canvas (a tela): Segurar Shit -> clicar na posição desejada -> Segurar CTRL -> clicar na mesma posção desejada



- Setar a margim do texto: PosX e PosY



- (Opcional) podemos definir render mode, se a canvas vai ser em fixo em cima da camera (ou de todos os objetos (default)) ou vai ser embaixo da camera, este tipo de camera só movél pode ser no chão, parede ou qualquer outro objeto.

# FORMATOS

FBX: OBJETOS QUE SUPORTAM ANIMAÇÕES

OBJ: OBJETOS QUE NÃO SUPORTAM ANIMAÇÕES

# Material e Textures

- A texture is an image. Straight out of paint, gimp, photoshop ect.

- A material is the details of what an object should look like. Most materials take textures as a parameter. There are many cases where a material takes multiple textueres. For example one texture might define colour, another transparency, another normals and so forth.

**- Quando Altera o material, altera o objeto que tem o material.**

**- Criar Material**: Project -> Material(Criar pasta) -> Botão Direito -> Create -> Material

**- Definir Textura do Material**: Arrastar textura para Albedo ->Tiling -> (Definir Tamanho da texture para repetição)

Para Procurar por texturas (imagem preparada para repetição) basta procurar por o termo: **seamless**

## Normal Map

O Normal Map é um tipo especial de texturas para criar relevos/desenção no material ou no terreno.

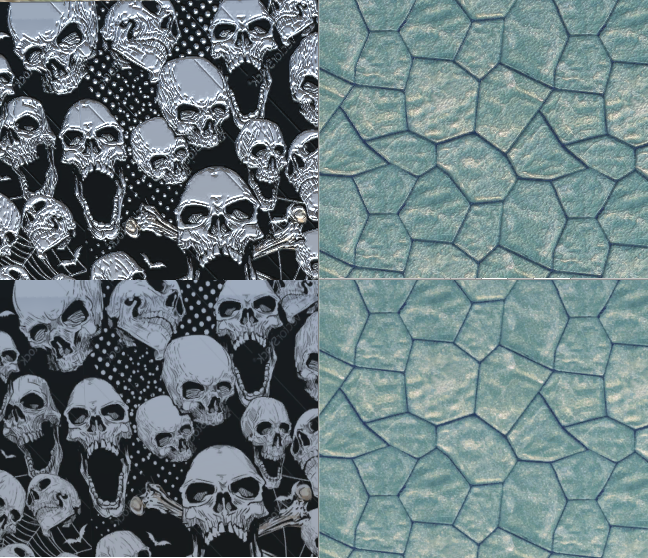
**Normal map da relevos nas texturas de chão, em texturas das casas, no mato (Requere MUITO PROCESSAMENTO, POR ISSO TEM QUE COLOCAR SCROLL PARA CONTROLAR A QUALIDADE).**

**- Podemos Criar normal automaticamente no unity apartir da texturas: Only as particulas pretas são desenção e as particulas brancas são relevo. Algumas texturas tem que fazer manualmente o normalmap por causa dos cores**

### Metodo 1

**Criar Normal map apartir da textura: TexturaDesejada -> Crtl+ D (Duplicar) -> Renomear NormalMapTextura -> Texture Type = normal Map -> Check Create from GrayScale -> Diminuir Bumpiness -> apply**

**-> Material -> arrastar normalMap criado para Nomal map -> Dimuir/aumentar smoothness&Material & normalMap.**

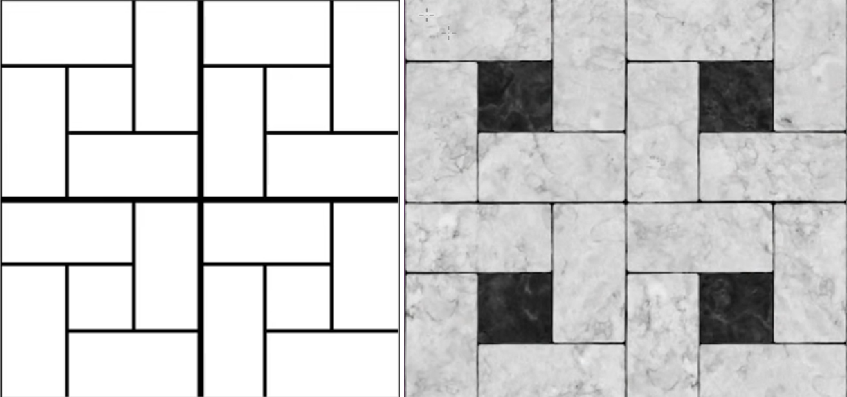
****

### Metodo 2

<http://cpetry.github.io/NormalMap-Online/>

### Metodo 3 (Melhor Resultado)

Criar normalmente o normalMap no Photoshop (Desnhar linhas pretas onde quer ter desenção)



# Animações

Nas animações basta mover colocar a posição inicial e final do movimeto que faz um movimento Linear.

## Criar Animação

1. Criar a Pasta Animations
2. Unity Menu -> Windows -> Animation -> Animator
3. Selecionar Objeto que Será Animado -> Animation -> Create -> Apertar no pontoVermelho (Record) -> Selecionar Part do objeto que será animada -> Nas Transformações Desejadas (Position) -> botão direito -> Add key -> Colocar o Linha do tempo no Momento desejado (1 segundo) -> mover para a posição desejada(posição final) -> play (para observar) -> Remover de todas as Animações Loop Time (Uncheck)



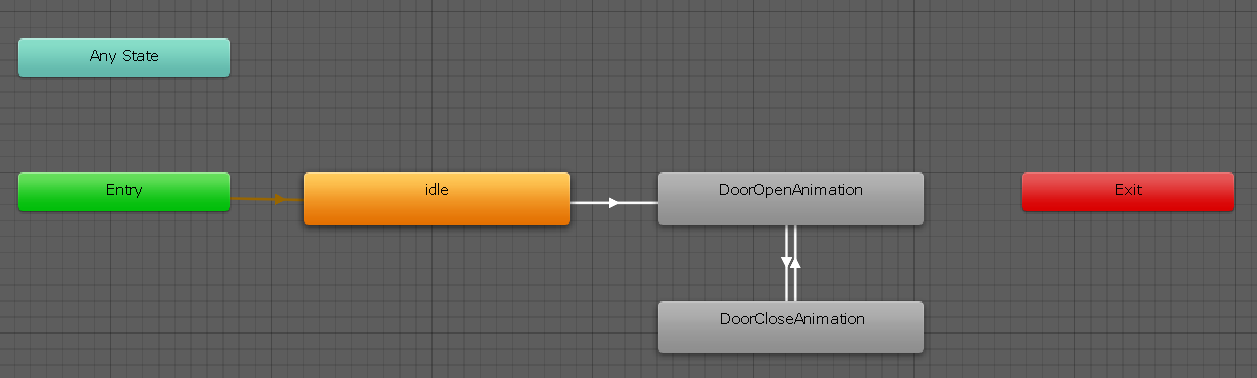
## Criar Animator Controller

Animator Controller é Controlador de animações onde podemos definir cadeias de animações e colocar para se reproduzidos em serie. Podemos também criar um trigger para cada animação que será desparada no script.

**Para criar Animator Controller(Normalmente Criado Automaticamente):**

-> Na pasta Animations -> Botão direito -> create -> Animator controller -> Arrastar Animações para o animator -> Botão direito -> Create State -> Empty (Criar Animação Default(Parado com estado atual)) -> mudar a animação para ‘idle’(nome padrão) -> Definir Animação Principal (Botão Direito) -> Set as Layer Default State.

-> Botão direito no bloco ‘Idle’ -> make Transition -> Proxima animação (AbrirPorta) -> make transition -> Proxima Transition(Fechar porta)



-> Na aba de parameters -> adicionar(parametros de condições) bool,trigger... -> nomear -> na Linha de transção -> Conditions -> Adicionar parametro da condição (Para ser executada). -> Uncheck had Exit Time



-> Objeto que será animado -> Add component -> Animator -> Arrastar animator controller

## Executar Animação C#

- Para executar frame basta seta o flag que deixa a condição de transição da animação verdadeira. **Para o trigger não precisa resetar o parametro da condição, outros tipos precisam restar**.

animatorComponent = GetComponent<Animator>();

animatorComponent.SetTrigger("openDoor"); //Flag para executar Animação: DoorOpenAnimation

# Scripts

-Normalmente é criado o script no objeto que vai atualizar (objeto que vai fazer ação como Texts de contagem de score).

## GameObject

Simbolizar um variavel de qualquer objeto na sena que pode ser inicializado no GUI com public, **ou obtido via o tag (não é recomendado)**

## void Start ()

Metodo executado somente 1 vezes na instânciação do objeto

## void Update ()

Metodo chamado 1 vez a cada frame

## void LateUpdate ()

Metodo chamado 1 vez a cada frame depois que o metodo update() foi chamado e todos os calculos do update já foram feitos, **normalmente é utilizado quando o player move no update() e todos os movimentos e rotações da camera são feitos na LaterUpdate()**

**- Criar um Script**: No objeto -> inspector -> Add Component -> escreveNomeDoScript -> Create And Add -> Arrastar para Pasta de scripts

//MonoBehavior: Propridades do objetos que são exibidos no Inspector

using UnityEngine.UI; //Imporar Biblioteca de UI dos objetos unity

public class MostrarAltura : MonoBehaviour {

public Transform transformPersonagem;//Propridade MonoBehavior da personagem (Public para ser visivel para outras classes)

//Metodo executado somente 1 vezes na instânciação do objeto

void Start () {

}

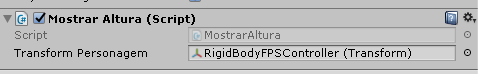
// Metodo chamado 1 vez a cada frame

void Update () {

}

}

- No lugar do transform Arrastar o Objeto que terá o transform



# Objetos Unity (Game Object)

**Todo Objeto Precisa Ter Tag. É um tag de referência para outros objetos poder achar o objeto.**

## Camera Automática

A camera automatica é recomendavél por que já tem os scripts prontos e otimizados para seguir o player e não ultrapassar os paredes, definir velocidades...etc.

A camera automatica é um asset do stander Assets do unity pode ser importado via: Assets -> Import package Cameras.

- Colocar Tag no Player .

-**Colocar prefab da camera MultiProposeCameraRig**

**-Free look camera: pode mover o mouse**

**-MultiProposeCameraRig: a camera é movida automaticamente (não é usado o mouse)**

## Camera Manual (Não Recomendavel)

A camerá pode ser filha do player mas vai rodar junto com o player e as vezes essa não é a melhor opção.

1. Colocar Tag no Player e na camera
2. Script:

public class CameraFollow : MonoBehaviour {  
    GameObject player;        *//jogador*  
  
    void Start () {            *// Use this for initialization*  
        player = GameObject.FindGameObjectWithTag("Player");    *//Achar o jogador com o tag já atribuido*  
    }  
  
    void LateUpdate () {     *//Metodo Chamado 1 vez a cada frame depois do metodo update(), onde é executado depois de terminar todos os calculos movimentos do jogador no update() para atualizar a camera no LaterUpdate()*  
        transform.position = player.transform.position; *//transform para o objeto que tem esse script attached (Camera)*  
    }  
}

1. Ajustar a camera

## Terrain

FREE ASSET TERRAIN TOOLKIT 2017 TEM MUITOS RECURSOS E MELHORIAS

No unity podemos criar um terreno onde terá relevações(montanhas...) ou buracos(rios, buracos, Dungeon..etc).

### Elevação

Basta clicar no rais do terreno e elevar o plano

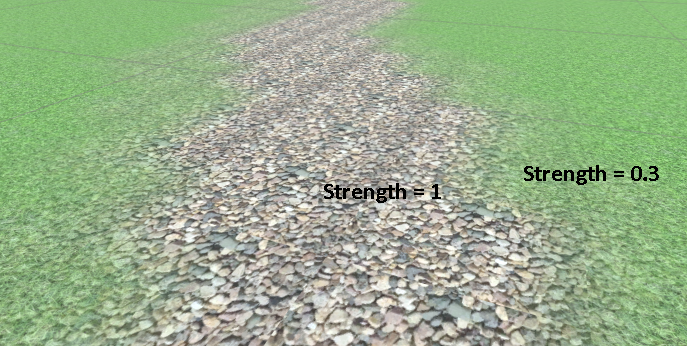
### [descensão](https://www.antonimos.com.br/descensao/)

basta clica no elevação e segurar Shift e descer o plano. ATENÇÃO: NÃO PODE TER DESCENSÃO ABAIXO DO PLANO DO TERRENO. Por isso:

1. Mover o centro do terreno para 0,0,0 (adicionando cubo no 0,0,0 e arrastando o centro para o cubo)
2. Abaixar a posição do terreno no eixo Y -100
3. Elevar todo o plano 100 em terrain -> Paint Height -> Height = 100 -> Flatten

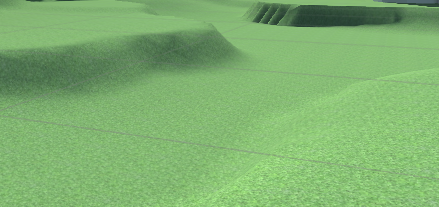
### Definir Caminho no chão

1. Download imagem de mato e de pedras e colocar no Assets
2. Terrain -> Paint Texture -> Edit Texture -> Add Texture -> Add. (fazer o mesmo para colocar a imagem de padras)
3. Mudar a escala em X e Y para ficar melhor visualmente para o jogador
4. Pintar caminho com o mouse com Strength = 1
5. Pintar o relevo do caminho(em volta do caminho) com strength = 0.3



### Desenhar Rio

1. Depois de evelar o chão: Terrain -> Paint Height -> 98 (isso significa 2 para baixo por que o terreno já está no 100) -> desenhar o rio
2. Colocar uma camada de agua



### Desenhar Montanha

1. Elevar o chão com ‘Paint Height‘ (110 de altura)
2. Passar ‘Raise Terrain’ para deixar a montaha pontuda
3. Repetir outras camadas atrás da montanha com altura maior
4. No inicio da primeira montanha ajudar com altura do plano.
5. **Passa Smooth se precisar**



### Importar e Exportar Terrno (Height Map)

Height map é um mapa que determina a altura dos pontos no terrno (Evelação/Desenção). Podemos importar/exportar as alturas do Terrno ou obter da internet um height map.

**PODEMOS UTILIZAR SITES DA INTERNET QUE DA PARA OBTER HEIGHT MAP DE UM CIDADE.**

[**https://terrain.party/**](https://terrain.party/)

### Desenhar Grama

**ATENÇÃO: DESENHAR SOMENTE UM POUCO DE GRAMA SE NÃO O JOGO FICA MUITO PESADO!**

Para desenhar grama basta importar Enviroment do Stander Assets **OU PODEMOS BAIXAR QUALQUER IMAGEM DE MATO COM FUNDO TRANSPARENTE E MARCAR NA IMAGEM A PROPRIDADE (ALTA IS TRANSPARENT).**

-> Adicionar Terreno -> Paint Detalhes -> Edit Details -> Add grass Texture -> GrassFrond01Albe (Definir altura) -> Desenhar com Strength 0.0625 -> Opacity 0.023 -> Desenhar(**COM CLICKS E NÃO ARRASTA**) -> Tirar um pouco da Grama (pq quando desenha e unity coloca muito) Shift + Apertar Mouse

**Ideia colocar Brush grande com menor Strength e Opacity para areas Grandes**

Essa Grama é uma imagem 2D com animação + Billboard do terreno (Segue a câmera)



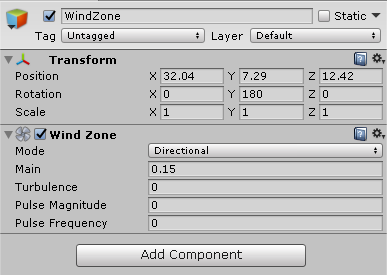
### Desenhar Arvores

É muito Fundamentar desenhar arvores em jogos por que o ambiente fica muito mais bonito.

Para desenhar grama basta importar Enviroment do Stander Assets

-> Adicionar Terreno -> Paint Tress -> Edit Tress -> Add Tress -> Broadleaf\_Desktop -> Definir altura da arvore(CHECK Random) + range muito baixo -> Posicionar uma arvore e testar se ficou bom a altra (não atrapalha a visão)

### Adicionar Vento

O vento é utilizado para mover as arvores e folhagem para dar animação e vida para o jogo.

Create -> 3D Object -> WindZone

Existe 2 tipo de vento:

**Direcional:** Não import a posição do vento que vai aplicar para todas as arvores da cena. NOTA QUE TURBULENCE, PULSE MAGNITUDE, FREQUENCY TODOS SÃO ZERADOS PARA NÃO FAZER UMA TEMPESTADE.

Spherical: Vai fazer o Vento somente em Volta do WindZone e o vento tem direção diferente depende da posição da arvore em relação do Vento.

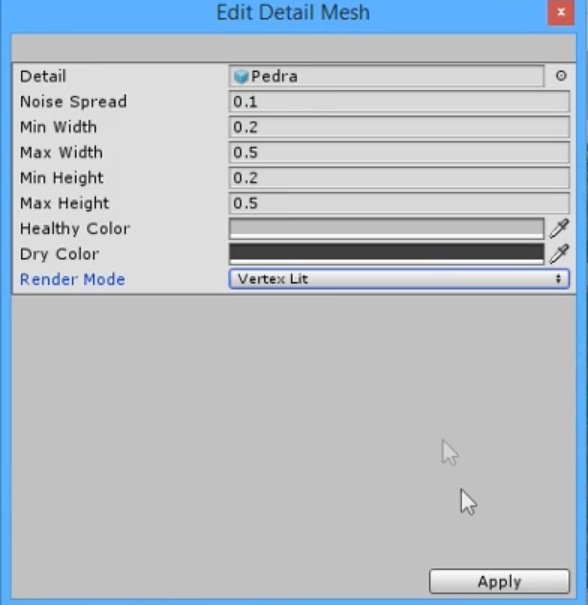
### Adicionar Agua

Para adicionar um plano de agua basta importar Enviroment do Stander Assets.

Depois de abaixar o terreno 100 e colocar a altura do terreno 100 e aplicar desenção no terreno ->

-> Standard Assets -> Enviroment -> Water -> Water -> Prefab -> Water.

EXISTE NO STANDARD ASSETS WATER COM ONDAS PARA CRIAR UM MAR

Podemos também adicionar um asset de water como: AquasWater

### Adicionar Pedra

Para adicionar pedras no terreno tem que ter um prefab de pedra.

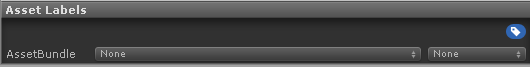
-> terrain -> Paint Details -> Edit Details -> Add Detail Mesh

-> Adicionar Prefab 0> Configurar parametros conforme a imagem do lado -> **Render Mode = Vertex Lit (não se mexer quando tem vento)**

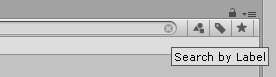
## Prefabs

Os prefabs são pacotes que contém objetos criados. **Tudo precisa ser prefab.**

- Os prefabs podem ser marcados nos labels(PREFAB -> CANTO INFERIOR ->Main), então **É IMPORTANTE CRIAR UM LABEL DE PREFABS BASICOS PARA CADA CENA (PLAYER, LIFE, TERRAIN, CAMERA, SUN, CANVAS...)**



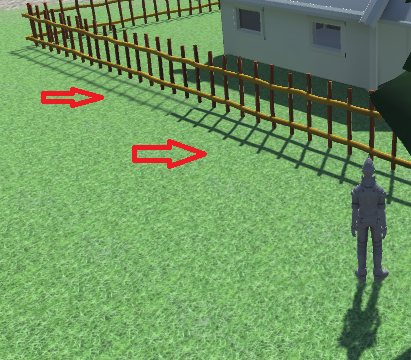
- Para acessar todos os prefabs marcados como main, basta abrir label na aba do projeto:



**IMPORTANTE: QUANDO ATUALIZAR PREFAB(SUBSITUIR) JÁ UTILIZADO NA CENA, SE A ESCALA NÃO FOR 1,1,1 ENTÃO VAI DESCONFIGURAR A ESCALA E VAI PRECISAR REFAZER A ESCALA.**

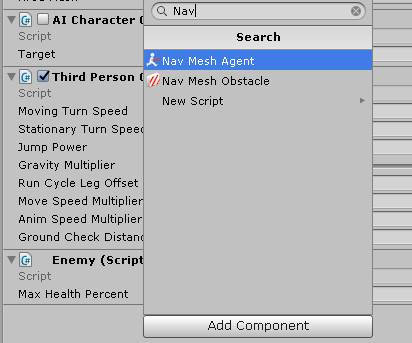
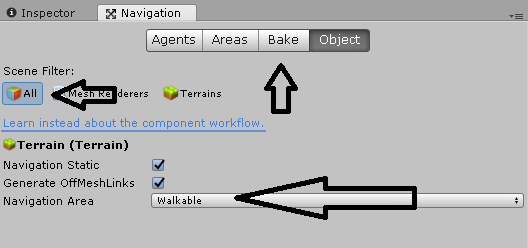
## Shadows (Sombras)

Se dar problema que o jogador está vendo que um lado a sombra está mais forte do que outro lado (mais distante) só basta alterar a distância da sombra na: unity -> edit -> Project Settings -> Quality -> Shadows -> Shadow Distance = 470



## Path Find ( Path Mesh)

Apartir da mesh é utilizado o algoritimo do path finder para encontar o caminho mais curto dentro da mesh path.

Window -> Navigation -> All -> Navigation Area = Walkable (**Layer criado e definido o chão e todo objeto que o jogador pode andar**) -> Bake -> Bake

-> Enemy(AI) -> Add component -> Nav Mesh -> Check para ativar component Nav Mesh -> Configurar Paramtero UI Stopping distance = 2 unidades (parar quando estiver perto do jogador 2 metros) -> Rigid Body -> isKinematic (configurar se o jogador pode arrastar o enemy)

-> Enemy(AI) -> AI Character Script(UI) -> Adicionar o jogador como parametro UI para o script (vai seguir o jogador)

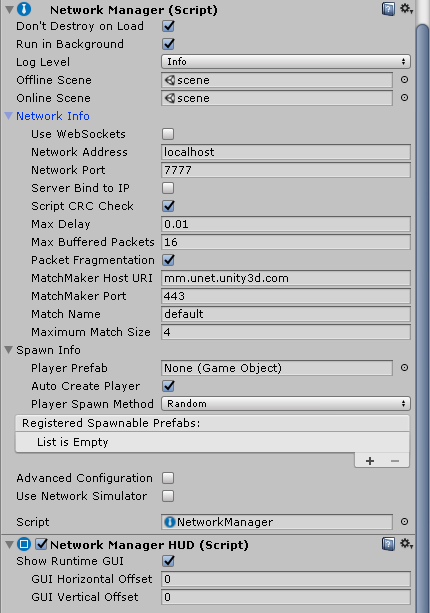
-> Enemy(AI) -> Editar Enemy script para ativar AI Character quando o jogador está presente no raio

IMPORTANTE: QUANDO UM OBJETO (COMO PONTO) ESTÁ ACIMA DO PLANO WALKABLE NÃO FUNCIONA O PATH FINDER POR ISSO TEM QUE ABAIXAR A PONTE.

Alterar o path finder altera o arquivo NavMesh.asset (arquivo que guarda o path finder)

# Unity Network

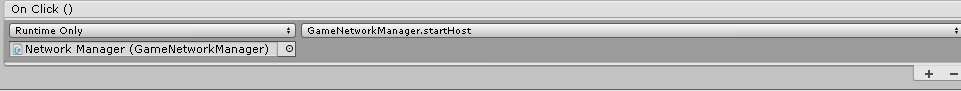
Modo Rápido(Não Usual):

1. ****File -> Build Settings -> Adicionar cena no Build
2. Criar prefab do player (arrastar player para arquivo do projeto)
3. Adicionar para o player Componente: ‘NetworkIdentity’
4. Criar um Empty Object e colocar o seguinte:
5. Adicionar Componte ‘**Network Manager**’ -> Adicionar como parametro a cena no ‘**Online Scence & Offline Scence’ -> Adicionar o prefab do player na propridade “Spawn Info -> Player Prefab”**
6. Adicionar Componente ‘**Network Manager HUD’ (INTERFACE PRONTA COM CANVAS PARA CONECTAR E DESCONECTAR)**
7. Remover o Player da Cena por que vai Spawn quando o jogo começar.

### Processo de Criação

Modo Avançado:

1. File -> Build Settings -> Adicionar cena no Build
2. Criar prefab do player (arrastar player para arquivo do projeto)
3. Adicionar para o player Componente: ‘NetworkIdentity’
4. Remover o Player da Cena por que vai Spawn quando o jogo começar.
5. Criar um Novo Script de GameNetworkManager (Escreve os metodos)
6. Criar um Empty Object(Colocar Nome: Network Manager) e colocar o seguinte:
7. Adicionar Script “GameNetwork Manager” -> Adicionar como parametro a cena no ‘**Online Scence & Offline Scence’ -> Adicionar o prefab do player na propridade “Spawn Info -> Player Prefab”**
8. Criar Um botão: Botão Direito -> UI -> Button -> Alterar o texto para “Start Host”
9. Botão Criado -> Onclick -> Add -> Arrastar Objeto Empty criado (Network Manager) para Object -> Escolher metodo Executado (startHost())



1. Adicionar em todos os objetos que tem ação (transformação geometrica, Ex: monstros, objetos que movem) um componente NetworkTransform (quando o objeto mover em um dos clientes vai atualizar isso para todos).
2. Adicionar ao Script do jogador que é responsável por Movimento do jogador:

using UnityEngine.Networking;

public class NetworkPlayer : NetworkBehaviour // para Usar isLocalPlayer

**//Colocar isso no Update, LateUpdate(), FixedUpdate() se existem**

if (!isLocalPlayer) //Se o movimento não é deste jogador então retorna

{

return;

}

1. Adicionar ao player NetworkTransform. Network Identity -> Check Local Authority (Cada jogador tem seu player e o servidor não tem autoridade a mexer com player. Cada player tem sua camera)
2. Metodos e variaveis compartilhados na rede (vida, attacks...etc). Onde podemos Criar variaveis no servidor sincronizadas com os clientes Como nome, vida, propridades de cada cliente. **Podemos Também criar metodos nos clientes que são sincronizados com o servidor e atualizados em todos os clientes, Onde começam com Cmd (CmdAtualizaNomeDoJogado(), CmdMudarVidaDoJogador()...)**

public class NetworkPlayerProperties : NetworkBehaviour {

//SyncVar: Variavel do servidor sincronizado com todos os clientes (Quando o variavel muda, vai atualizar no servidor. é uma variavel readOnly nos clientes e da para mudar com Metodo Command)

//hook="onNameChanged": É o metodo chamado nos clientes quando o variavel do servidor é atualizada

[SyncVar (hook="OnChangePlayerName")]

public string playerName = "player1";

public TextMesh txtPlayerName; //Só Está Funcionando como public e não acha com getComponentInChildern

//Server Methods

[Command]//Indicar que o metodo é chamado no Servidor e espalhado em todos os clientes para chamar também (Atualizar nome de um dos jogadores)

public void CmdChangePlayerName(string newName) //Para o metodo de comando é obrigatorio o metodo começar com Cmd..

{ //(Cliente Autenticado pode atualizar seus Variavel SyncVar no Servidor )

playerName = newName;

GetComponentInChildren<TextMesh>().text = playerName; //Aualizar o nome a cada frame criado

}

//Cliente Methods

public override void OnStartClient() //Quando o cliente é incializado ele precisa obter os variaveis do servidor (Processo de sincronização)

{

base.OnStartClient();

OnChangePlayerName(playerName); //Chamar o metodo hook que sincroniza o variavel do servidor para atualizar no cliente atual

}

public void OnChangePlayerName(string newName) //Metodo com hook criado que é chamado quando o SyncVar é atualizado

{

playerName = newName;

txtPlayerName.text = playerName; //Aualizar o nome a cada frame criado

}

//GUI - Botão em cada Cliente

private void OnGUI() //OnGUI é chamada varias vezes por frame para verificar eventos na interface gráfica

{

if (isLocalPlayer) { //TEM QUE VERIFICAR PRIMEIRO SE O USUARIO ESTÁ MEXENDO COM A SUA INTERFACE GRÁFICA

//Criar novo botão com evento

if (GUI.Button(new Rect(25, 15, 100, 25), "Mudar Nome")) //Para os jogadores (Clones de prefab) tem que colocar botão dentro da instância para mudar os valores desta instancia e não do prefab

{

CmdChangePlayerName("Nome1232");

}

}

}

void Start () {

}

void Update()

{

if (isLocalPlayer)

{

txtPlayerName.text = playerName; //Aualizar o nome a cada frame criado

}

}

}

1. **Criar Spawm Point**: Spawm point é ponto que vai criar jogadores/monstros.

->Criar objeto Vazio -> Add Component -> Network Identity -> Add Component -> Network Start Position -> Criar prefab do Objeto vazio -> Criar Quantos spawm point que for precisar(**Quando vai registrar spwm point só registra 1 prefab dos pontos).**

-> Network manager Object -> network manager script component -> Spawm player Method = Round Robin -> Add Spawn -> Arrastar o prefab.

1. **Adicionar Fire Actons**:

-> Adicionar um cubo para o jogador -> Nomear como FireSpawn -> Mover para posição onde vai atirar**(Colocar sem tocar no Colider do jogador)** -> desabilitar Mesh Render para tirar(se a posição não ficar boa, tem que habilitar para repositionar com melhor visualização)

->Prefab que vai atirar **(EX:Bala de Fogo que tem rigid body)** -> Adicionar Componente -> Network Identity -> Adicionar Componente -> Network Transform -> Network Send rate = 0(não vai ficar atualizando o posição, só vai enviar que lançou fogo) -> Transform Sync Mode = Rigid Body 3D

-> Utiltizar Script ‘FireControl’ com 2 parametros public (prefabs da Bala de fogo e local de spawn da bala)

1. **Sistema de Vidas (HealthBar & PlayerName)**:

Para o Jogador -> Criar Canvas -> Canvas Component -> Render Mode -> World Space -> Adicionar Script ‘FaceToCamera’ para o canvas -> Definir altura e largura do canvas para caber healthbar-> Arrastar o canvas para ficar em cima do usuario. -> (Se tem nome do usuario em cima da cabeça do jogador arrasta para Canvas e tirar faceTocamera script do texto para ficar somente na canvas)

-> Adicionar dentro do canvas um Silder -> Apagar subObjeto Handle Silde Area -> aplicar Scale 0.01 em X, 0.00179 em Y, 0.01 em Z -> altura e largura do silder = 100 -> SubObjeto background -> color -> vermelho -> Subobjeto Fill Area -> SubObjeto Fill -> Color Verde.

->HealthbarSilder -> Max Value = 100 -> Value = 100 **(TERÁ UMA PARTE VERMELHA)** -> Utilizar Ferramenta Rec Tool -> Diminuir o tamanho da parte vermelha -> Aplicar no prefab -> Aplicar no verde também.

-> Script de configuração está no NetworkPlayerProperity.cs

Prefab da tomba de morte -> Add Component -> Network Identity-> Adicionar o prefab no Spawm do Network Manager

1. **Selecionar Jogador Desejado**:

-> Para cada jogador tem que ter os scripts CONFIGURADOS “NetworkPlayerProperity.cs”, “FireControl.cs”, “PlayerController.cs” -> Network Identity com Local Player Authority -> Network Transform com Sync RigidBody 3D -> Collider Configurado e ajustado -> RigidBody com Freeze Rotation X,Y,Z -> Animator do Prefab

-> Adicionar Todos os Players Prefab no Network Manager(Registered Spawnable Prefabs) -> Utilizar networkManager que tem o codigo que de adicionar character escolhido pelo o cliente

**Funcionamento do script: Processo de escolher Character desejado ao conectar**

\* 1- Definir Handler no Servidor e no cliente para enviar e responder mensagem.

\* 2- Quando o Usuário connecta ao servidor, O servidor manda uma mensagem para o cliente escolher o prefab via OnServerAddPlayer, mas não vai completar a execução com base... até o usuário reponder com caracter desejado (não criar até ele escolher).

\* 3- O cliente ao conectar vai enviar o character escolhido (Handler).

\* 4- O cliente ao conectar vai responder a mensagem com o character escolhido (Handler).

\* 5- O Servidor vai receber a mensagem e vai definir o default player prefab e vai spawn o jogador.

1. Criar Lobby Scene:

-> Create -> Scence -> Build settings add Sence (Colocar como primário no topo)

->Arrastar Lobby/Prefab/Lobby Manager para o Scene -> Atruibuir Parametros GUI -> Lobby & Main Scene -> Max Players -> Max Players Per Connection -> Minimuim Players = 1 -> Lobby Player Prefab

### Network Info

- Network Manager Incialization Process:

onStartServer -> Register Handler onRespondPlayerPrefab() -> Esperar até um usuario conectar -> Quando o Cliente conecta OnClientConnect() -> Register Handler onRequestPlayerPrefab() -> Agora o Servidor vai desparar OnClientConnect() e vai mandar mensagem para o cliente escolher Prefab (Não vai chamar base.OnServerAddPlayer()) -> Cliente vai responder a mensagem enviada do servidor com prefab escolhido onRequestPlayerPrefab() -> Agora quando o servidor recebe a resposta, ele altera o Default prefab e chamar base.OnServerAddPlayer() para criar jogador.

- Os Variaveis principais do jogador Não podem está na maquina local, eles tem que estar no servidor onde sincroniza com todos os clientes. Por Isso Utilizar Variaveis SyncVar que são variaveis armazenadas no servidor e o cliente só tem direito de ler

- O cliente pode solicitar para o servidor atualizar seus variaveis atraves dos metodos [Command] (metodos começam com Cmd..). MAS SÓ PODE ALTERAR A INSTÂNCIA QUE TEM ALTORIDADE

- Quando o cliente é incializado ele precisa obter os variaveis do servidor (Processo de sincronização) atraves do metodo OnStartClient

- Os clientes sincronizam os variaveis do servidor atráves do hook. É um modificador do metodo que informa os clientes quando o variavel do servidor é alterado.

- Os Spawm points Criados só aparecem quando o servidor for iniciado

**-TODO QUE É DESTORIDO NO SERVIDOR É DESTOIDO NOS CLIENTES**

**- Metodo Mais eficente para criar ação (Atirar)**

1- Para atirar o cliente vai chamar o metodo do servidor Cmd..

2- O servidor vai chamar o metodo ClientRpc Rpc.. para criar um tiro em todos os clientes i nclusive no servidor

**- [SyncVar]:** only sync from Server --> Client. They never sync from Client --> Server. However they \*can\* be changed locally on the client and no errors will be thrown.  
**- [Command]:** send information from Client --> Server. It can only be used if the client owns the object and has authority.  
**- [ClientRPC]:** are just like command's but send information from Server --> all Client's  
**- [TargetRPC]:** is just like ClientRPC, but only sends to 1 client (not all clients).

-Nota-se: Cada interface do cliente (Aba do jogador) tem sua propria camera que somente é criada e visualizada pela interface do cliente. (Servidor não tem acesso para aumentar a eficiencia de processamento).

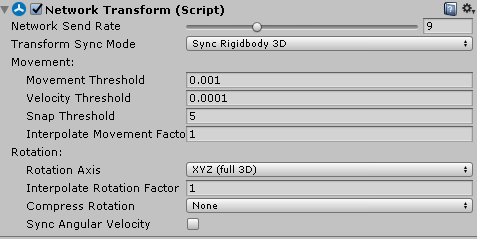
- O componente ‘Network Transform’ utiliza um algoritimo de interpolação onde vai prever a posição que o jogador estará no presente para não mostrar o atraso do envio de informações na rede.

**Network Send Rate**: Quantas informações manda por segundo. **OBJETOS MUITO RÁPIDOS REQUEREM TAXA DE TRANSFERÊNCIA MAIOR.**

**Transform Sync Mode**: Componente que será sincronizado.

**Movement Threshold**: Quanto que é a distancia que o objeto pode mover para desparar uma mensagem de atualizar posição nos outros clientes. **OBJETOS MUITO RÁPIDOS REQUEREM MAIOR** **Movement Threshold & ROTATION THRESHOLD.**

Velocity Threshold: Quanto que é a distância para aplicar a correção de distância (interpolação) entre a network e local.



# Audio

Cada cena tem somente 1 ouvinte de audio (**Audio Listener**) que é a camera principal. Os objetos podem ter ‘**Audio Source’**.

# Gizmos

Metodo Utilizado em unity C# para debug gráfico onde pode construir elementos gráficos em tempo de execução. (Para Habilitar Gizmos tem que apertar no Gizmos)



**Usado para desenha linha do jogador até o alvo,**

private void OnDrawGizmos() //Metodo para construição de elementos gráficos (para Debugg) em tempo de execução (chamado cada frame)

{

Gizmos.color = Color.black; //Cor das linhas Gizmos

Gizmos.DrawLine(transform.position, currentClickTarget); //desnhar linha do jogador até o alvo

}

# Comandos

public Transform transformPersonagem;//Propridade MonoBehavior da personagem (Public para ser visivel para outras classes)

Debug.Log(transformPersonagem.position.y); //Imprimir no console do Unity a posição y da personagem

public Text alturaText; //Caixa de texto, depois pode arrastar qualquer text no UI

alturaText = GetComponent<Text>(); //Inicializar o text, depois pode arrastar qualquer text no UI

# Animation

- Todo objeto é feito animação em relação do posição 0,0,0 por isto que colocar dentro de um cubo onde a gente tira o mesh renderer e o box collider

- Existe 2 tipos de animations:

-Legacy Animation (Animação antiga): Utiliza componente Animation, quando clica na animação -> Inspector -> wrap mode -> (loop/once) definir quantas vezes deve reproduzir a animação.

Play C#: GetComponent<Animation>().Play("walk", PlayMode.StopAll);

-Animator Controller: É a animação mais utilizada onde podemos criar fluxogramas com condições para reprodução da animação.

# Build Project

File -> Build Settings -> Escolher Plataforma -> Escolher arquitetura x86/x64 -> Escolher se será verão de developer ou versão release -> Arrastar Cenas que vão ser incluidas -> Build -> Definir Path e nome

# Variaveis

**Public:** qualquer classe pode ver, e é mostrado no GUI para alterar a propridade

**Private**: somente a classe proprietária pode alterar, e não é mostrado no GUI

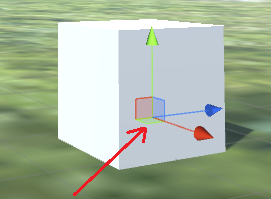
**SerializeField:** Privado mas a propriedade é mostrado no GUI

# Assets

- Importar um caracter: Unity-> Assets -> Import Package -> Character

Files -> Assets -> Caracters -> ThirdPersonCharacter ->Prefabs

# Dicas

- Quando Criar um novo projeto, crie um objeto vazio chamado ‘world’, e deixar o world como (-0.5) na posição do eixo Y (para conseguir deixar os objetos encima do plano)

- Quando Desejamos mover o objeto sem alterar no eixo Y podemos clicar no quadrado interior do objeto, no lado do eixo Y e mover.

-Para Obter o prefab “Prefabricate” um objeto qualquer da cena na pasta do projeto, basta arrastar a instância da Hieranchy para Pasta de Assets. **Importante: qualquer alteração feita na instância principal e criando Apply vai mudar em todas as instâncias do prefab! Se mudar pode Criar novo prefab que o anitgo não alterará**

- **No tempo de execução podemos alterar os valores** de posição para ver o melhor ponto de vista depois no componente transform -> copy component -> depois de execução -> componente transform -> paste component values

**- Para alinhar Objetos**: apertar v e definir angulo e arrastar para alinhar com outro objeto.

-Os bots IA do Ethan precisamter a propridade Material em: Max Fraction. **Isso para evitar que o IA escorrega ao tocar nele**

# Detectar A posição do Curso nas camadas permitidas

1. Utilizar a classe CameraRaycaster .cs onde utilizar raycaster (processo AND para detectar aonde o curso aponta).
2. Utilizar a calsse PlayerMovement.cs (Controle com Mouse)
3. Definir o numero de cada layer em classe com enum
4. Criar classe cursor.cs na Camera Arm

public class Cursor : MonoBehaviour {  
  
    CameraRaycaster cameraRayCaster;        *//Objeto para detectar em qual camada o mouse está (walkable, enemy,...etc)*  
  
    *// Use this for initialization*  
    void Start () {  
        cameraRayCaster = GetComponent<CameraRaycaster> ();        *//obter a cameraRaycaster por o tipo (Componente)*  
    }  
      
    *// Update is called once per frame*  
    void Update () {  
        print (cameraRayCaster.layerHit);        *//Printar em qual camada o cursor do mouse está*  
    }  
}

# Input Manager

Input manager já tem nomes prefinidos no unity para serem utilizados para detectar se uma ação foi invocada com um botão responsável pela a ação.

Horizontal, Vertical, Jump, Fire1, Fire2, Fire3, MouseX, MouseY

Input.GetButtonDown("Jump");

Input.GetAxis("Horizontal")

# Comandos C#

## Vector3

É um vetor Tridimensional

## Vector3.magnitude vs Vector3.normal

Vector3.magnitude: é o modulo do vetor onde vai obter a distancia entre o origem e o ponto

**Vector3.normal**: Vetor/Magnitude

## Distância Entre 2 Vetores

float DistanceBetweenEnemyAndPlayer = Vector3.Distance(player.transform.position, transform.position); //distancia entre o jogador e o inimigo

## Outros

Camera.main.transform.position = transform.position - transform.forward \* 10 + transform.up \* 3; //Mover a camera 10 unidades atrás do jogador e 3 unidades acima do jogador

Camera.main.transform.LookAt(transform.position); //Deixar a camera olhar para o jogador

Camera.main.transform.parent = transform; //Definir o jogador como o pai da camera

## RigidBody.IsKinematic

**Deixa outros objetos passar por este objeto mas, o objeto não cai mais e deixa animation travar**

# DLL

- Unity aceita DLL, mas para executar em Android ou para outras plataformas tem que ser feito com .net versão 2.0.

- Para desmontar dll podemos utilizar dotpeek exportar para projeto visual studio -> Visual Studio -> Botão direiro no projeto -> Propridades -> Versão .net -> 2.0 -> Tipo de saida -> Biblioteca de classes