What it the Planet?

Com Unity 5

Denis Ken

Humberto Lino

Março 2017

O que fazer

Enunciado

Desenvolver um jogo em 3D para celular utilizando o motor Unity. A construção do jogo deve seguir o GDD (Game Design Document) do semestre passado (2016/2).

https://github.com/humbertodias/unity-projeto-integrador-extra/blob/master/doc/gdd.pdf

Requisitos

Grupos de até 4 alunos;

- O aluno deve desenvolver a fase de um jogo utilizando a UNITY;
- Deve descrever as Regras do Jogo;
- Deve ter vitória e derrota;
- Pelo menos 2 desafios;
- Menu de Start, Pause, Derrota, Vitória;
- Áudio;
- Todos os elementos do jogo devem ser desenvolvidos pelo aluno.

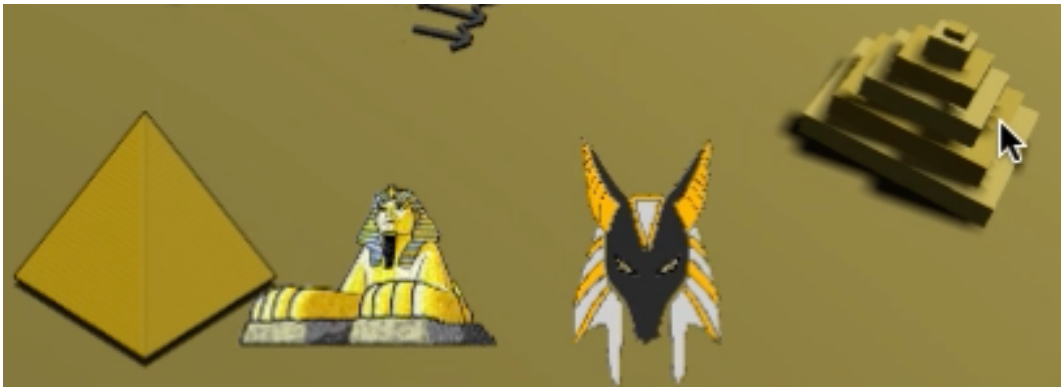
Como Jogar

Você deve destruir a torre inimiga e seus lacaios. Para isso, deve instanciar suas torres, selecionar seus lacaios e ordenar que ataquem os inimigos.

1. Instanciar

Clique na carta e clique novamente na posição do terreno que facilite o ataque ao inimigo.

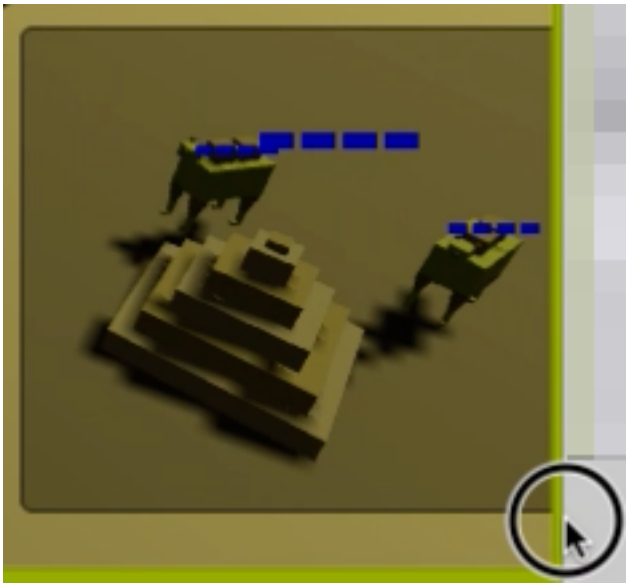
1.1. Instanciado



Para instanciar lacaios, basta clicar na torre instanciada no cenário.

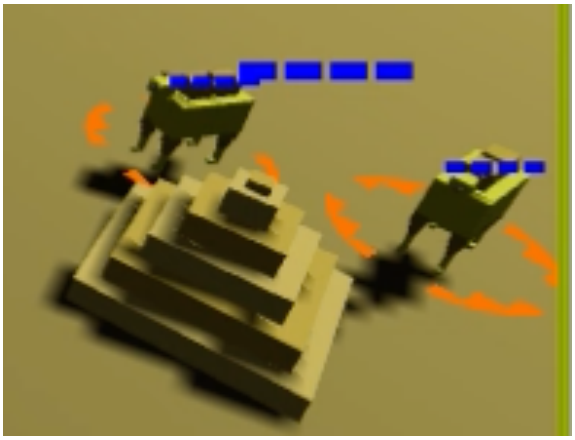
Cada torre possui um lacaio específica com características únicas e que dependendo do momento da batalha, serão melhores combatente.

2. Selecionar



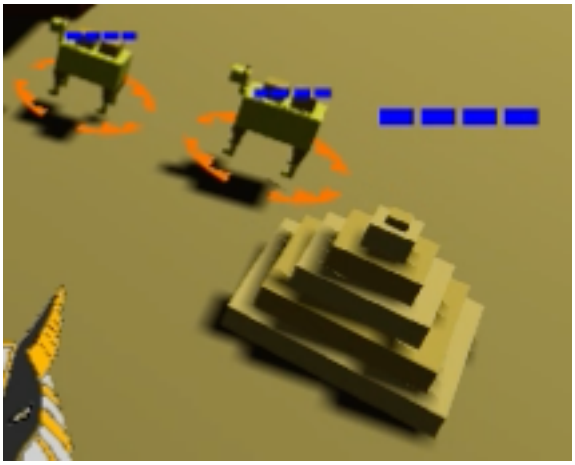
Selecione com o botão esquerdo mouse ou um dedo no dispositivo de tela sensível ao toque.

2.1. Selecionados



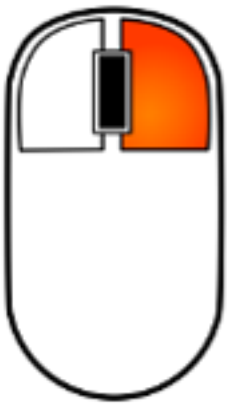
Perceba que os lacaios selecionados possuem um circulo para diferencia-los.

3. Atacar



Para atacar, clique no ponto mais próximo de seu adversário ou com dois dedos em uma tela de toque.

3.1. Atacando



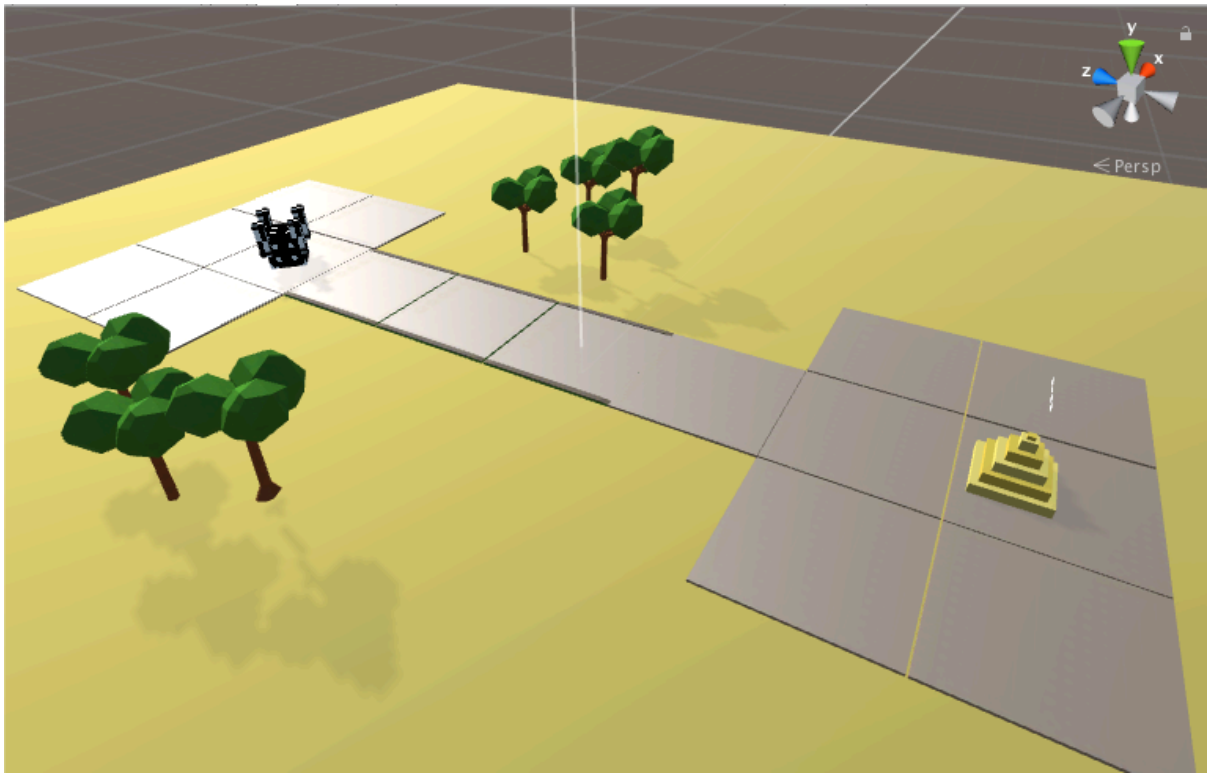
Two Finger Tap

Como foi feito

Tecnologias

Utilizamos o motor **Unity 5.5.2f1** para construção do jogo.
Blender 2.78c e **MagicaVoxel** para criação de cenário/personagens.

Cenário



Os cenários de batalha são basicamente compostos por um **Player** (Pirâmide/ Esfinge) um terreno ou plataforma com NavMesh indicando o caminho permitido e **Inimigos** (Torre Romana) com inteligência artificial para destruir o jogador.

HUD

Cartas

No canto inferior esquerdo o jogador encontrará as cartas que pode jogar no campo de batalha.

Contadores

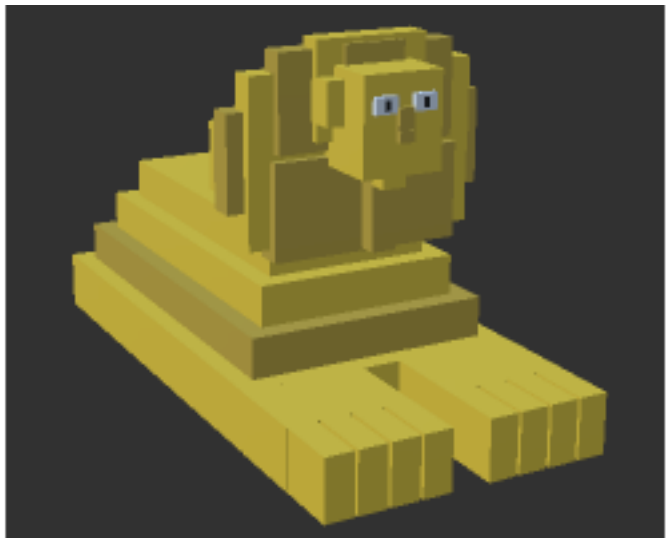
O contador de cor azul contabiliza o total de itens do Player, já o vermelho do Inimigo/NPC.

Player

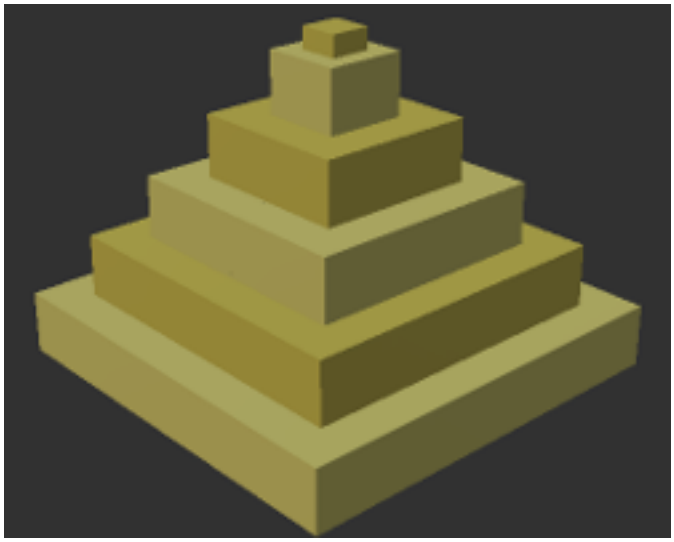
Torre

Todas as torres possuem a seguinte hierarquia **Corpo**, **HealthBar**:

Esfinge

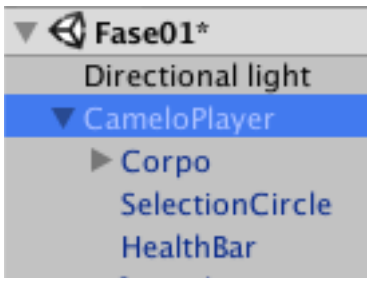


Pirâmide



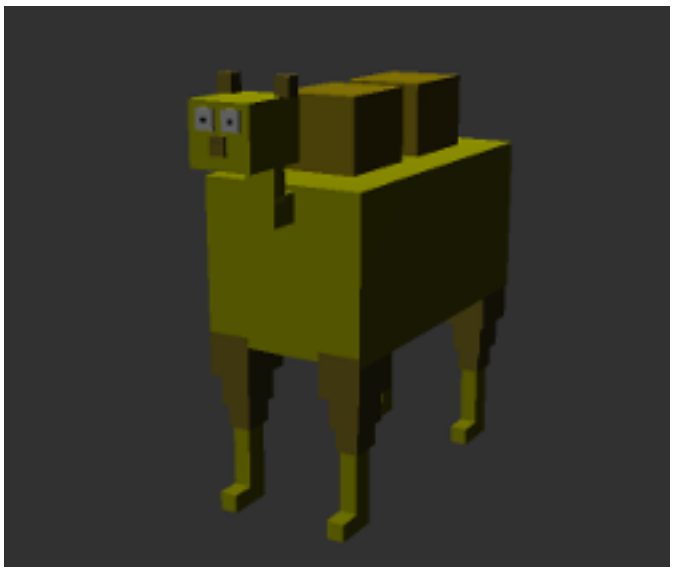
Lacaios

Todo lacaio possui: **Corpo**, **SelectionCircle** e **HealthBar**



Cactus

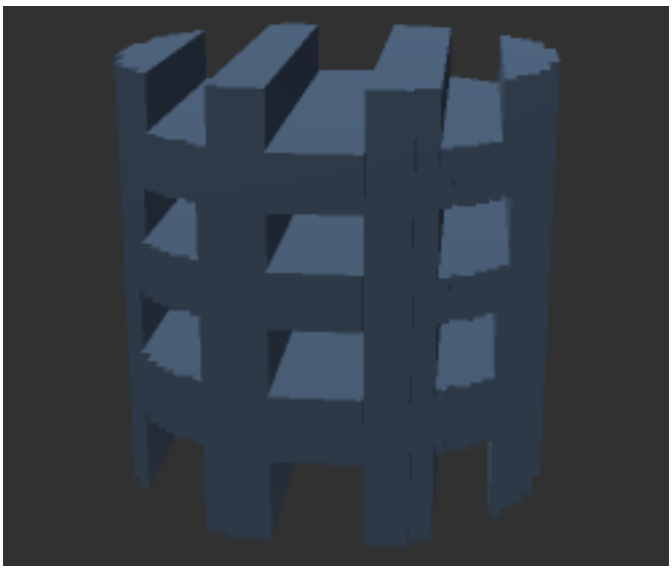
Camelo



Inimigos/NPC

Torre

Coliseu



Toda torre possui a capacidade de instanciar lacaios que atacam o inimigo (**prefab**).

```
public class UnitSpawner : MonoBehaviour {
    // prefab
    public GameObject unit;
    public float spawnRange = 1.5f;

    public void spawn() {
        // inicia nova animação
        GetComponent<AnimationSinus>().toggle();

        // cria uma nova unidade em uma posição aleatória
        Vector3 pos = transform.position;
        float x = pos.x + Random.Range(-1.0f, 1.0f) * spawnRange;
        float y = pos.y;
        float z = pos.z + Random.Range(-1.0f, 1.0f) * spawnRange;
        float angle = Random.Range(0.0f, 360.0f);
        Instantiate(unit, new Vector3(x, y, z),
                                Quaternion.Euler(0.0f, angle, 0.0f));
    }
}
```

Como funciona

1. Player

O Player se movimento de acordo com a mesh de navegação. Primeiro o selecionamos com um dedo ou botão esquerdo do Mouse e por fim o ponto destino com dois dedos/Botão direito do mouse.

Ex:

```
public class MoveByPlayer : MonoBehaviour {
      // Atualiza a cada frame
      void Update () {
          // Botão direito esta seleciona?
        if (Input.GetMouseButtonDown(1) && isSelected()) {
            // Find out where the user clicked in the 3D world
            Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
            RaycastHit hit;
            if (Physics.Raycast(ray, out hit)) {
                GetComponent<UnityEngine.AI.NavMeshAgent>().destination =
hit.point;
            }
        }
      }

      // Descobre se esta selecionado (se a seleção do circulo esta
visivel)
    bool isSelected() {
        MeshRenderer[] children = GetComponentsInChildren<MeshRenderer>();
        foreach (MeshRenderer r in children) {
            if (r.gameObject.name == "SelectionCircle" && r.enabled) {
                return true;
            }
        }
        return false;
    }
}
```

2. NPC

O Inimigo se movimento de acordo com a mesh de navegação através do Agente.

Ex:

```
public class MoveByNpc : MonoBehaviour {

      // A cada frame
      void Update () {
        // pega intervalo de ataque
        float range = GetComponent<Attack>().range;

        // encontra inimigos
        string enemyTag = GetComponent<Attack>().enemyTag;
        GameObject[] units = GameObject.FindGameObjectsWithTag(enemyTag);

        // se tem algum atacando? então, não tem que atacar
        foreach (GameObject g in units) {
            // ainda vido?
            if (g != null) {
                if (Vector3.Distance(transform.position,
g.transform.position) <= range) {
                    return;
                }
            }
        }

        // ja esta se movendo? então não faça nada
        if (GetComponent<UnityEngine.AI.NavMeshAgent>().hasPath) {
            return;
        }


        // pega um alvo aleatório (se houver algum)
        if (units.Length > 0) {
            int index = Random.Range(0, units.Length);
            GameObject u = units[index];

            // ainda tem vida?
            if (u != null) {
                // mova proximo o suficiente para atacar
                Vector3 pos = transform.position;
                Vector3 target = u.transform.position;

                Vector3 dir = target - pos;
                dir = dir.normalized;
                Vector3 dest = pos + dir * (Vector3.Distance(target, pos)
- range);

                // fala para o agente da navmesh ir até lá

GetComponent<UnityEngine.AI.NavMeshAgent>().SetDestination(dest);

            }
        }
      }
}
```

3. Menu

O menu é iterativo e contém animações de transformação na propriedade escala.

4. Volume/Mudo

A função com audio e sem, fez uso do controle AudioMixer, que ao exportar a variável VolumeMasterMixer, nos possibilitou o controle através do script abaixo.

```
public void SetSoundOnOff(){
	if(soundButtonIsPlaying){
		// mute mixer
		masterMixer.SetFloat("VolumeMasterMixer",-80f);
		soundButtonIsPlaying = !soundButtonIsPlaying;
		soundButton.GetComponent<Image>().sprite = soundButtonOff;
	}  else {
		// normal volume
		masterMixer.SetFloat("VolumeMasterMixer",0f);
		soundButtonIsPlaying = !soundButtonIsPlaying;
		soundButton.GetComponent<Image>().sprite = soundButtonOn;
	}
}
```

5. Pause

A ação pausar, altera o atributo timeScale da classe Time. Conforme o trecho a seguir.

```
public void SetPauseOnOff(){
        if(pauseButtonIsPaused){
                // pause
                Time.timeScale = 0f;
                pauseButtonIsPaused = !pauseButtonIsPaused;
                pauseButton.GetComponent<Image>().sprite = pauseButtonOff;
        }  else {
                // unpause
                Time.timeScale = 1f;
                pauseButtonIsPaused = !pauseButtonIsPaused;
                pauseButton.GetComponent<Image>().sprite = pauseButtonOn;
        }
}
```

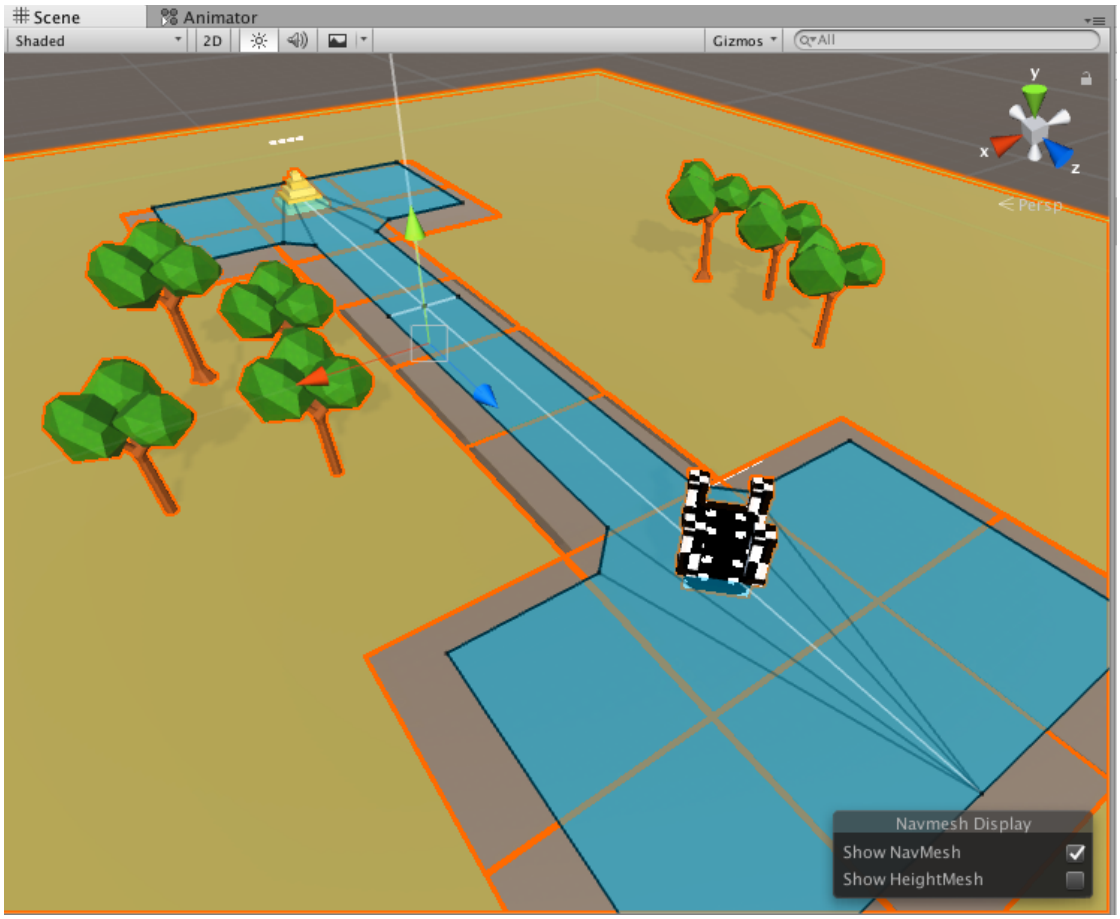
6. Vitória

O critério de vitória é simples. Basta que o contador a direita de cor **vermelha** esteja com valor **zero**. Pois, ele contabiliza o numero de inimigos em cena.

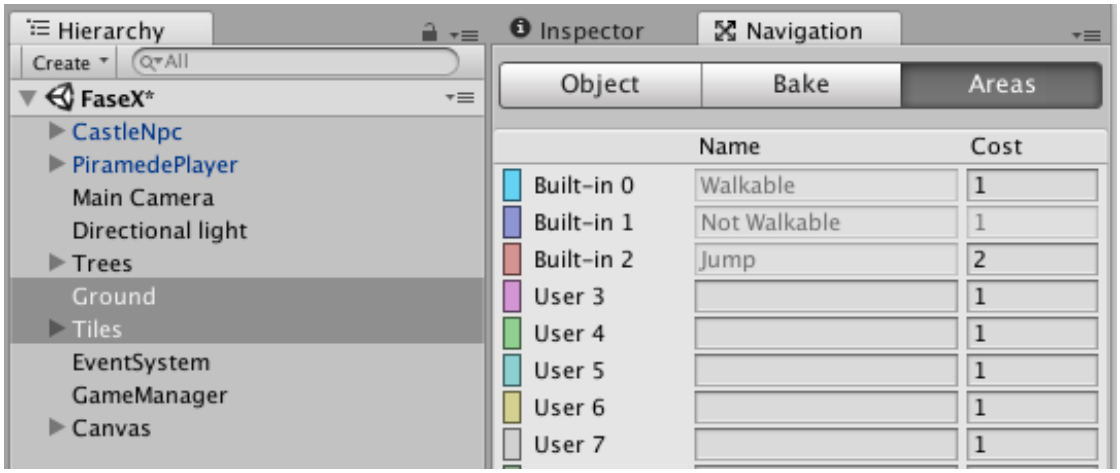
7. Derrota

O critério de **derrota** é simples. Basta que o contador de cor **azul** esteja com valor **zero**. Pois, ele contabiliza o número de aliados vivos em cena.

8. Geração de terreno com NavMesh

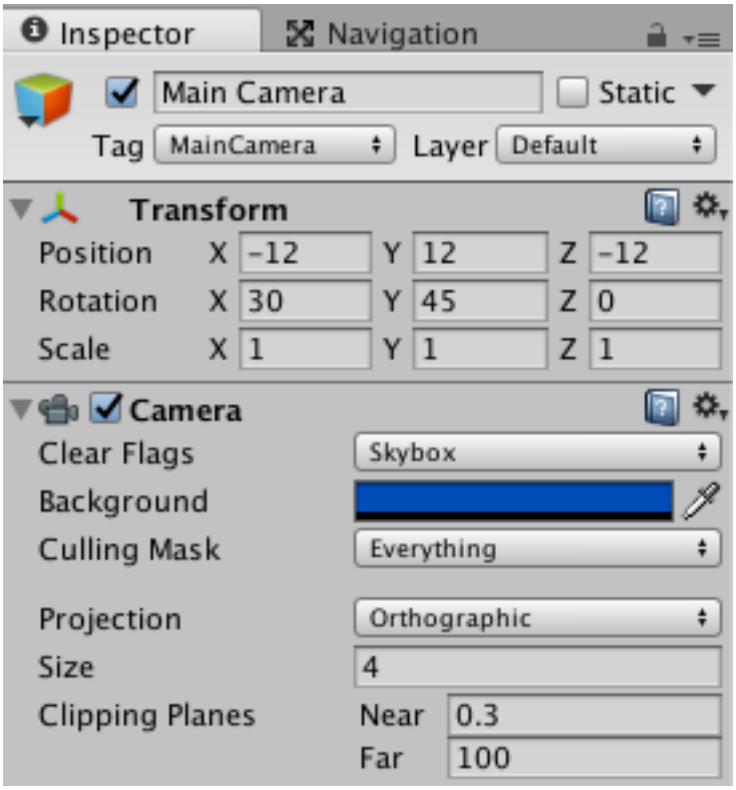


Para que o Player e NPC não trilhassem caminhos inválidos. Definimos em conjunto com o terrenos uma plataforma.



9. Controle de câmera

A camera foi possui a rotação de **30** graus no eixo **X** e **35** no eixo **Y**.



10. Sonorização

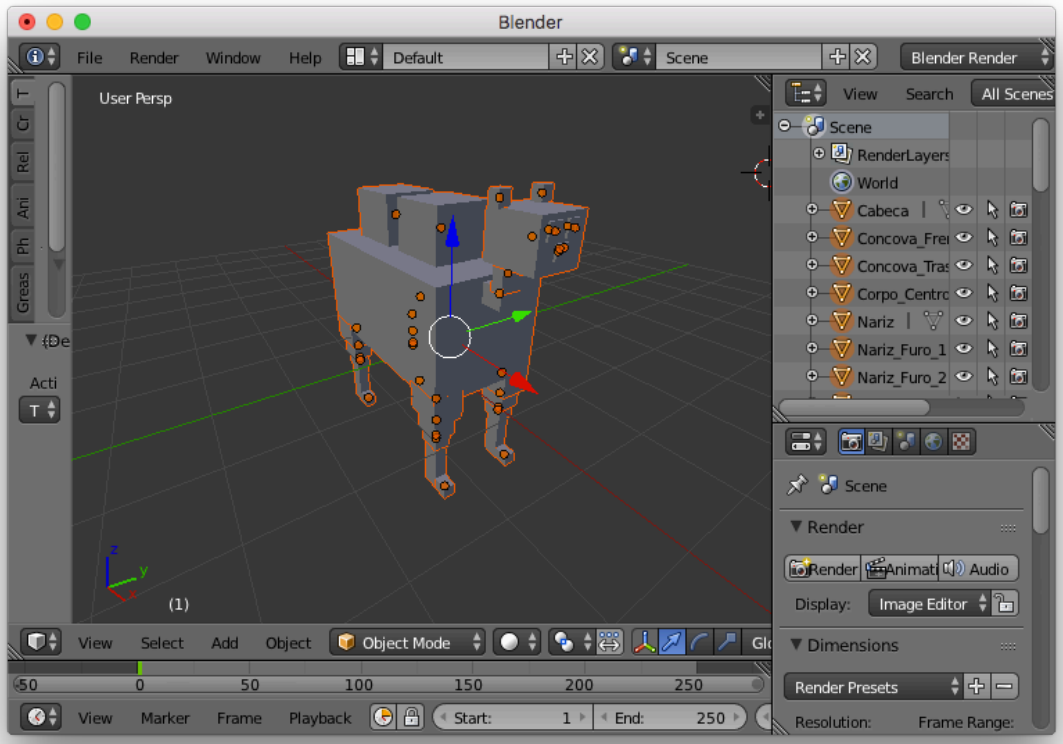
Utilizamos os efeitos sonoros do site www.freesound.org e para músicas de fundo algumas trilhas do filme **A múmia**.

Para o controle multiplo de som, utilizamos o componente **AudioMixer**.

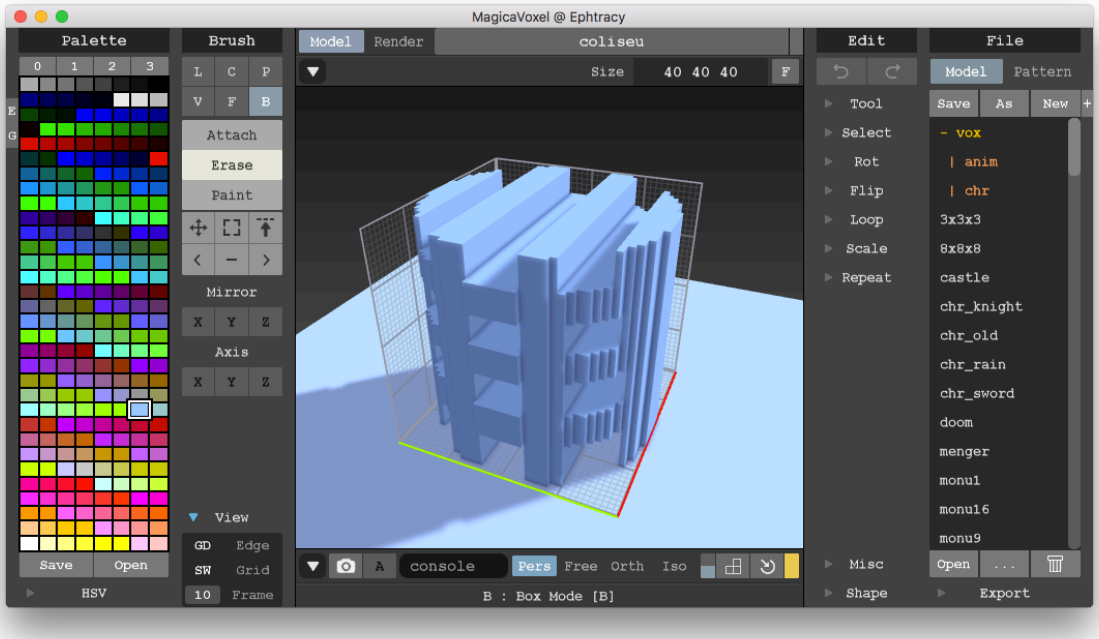
11. Modelagem

Utilizamos duas ferramentas para modelar os objetos do jogo:

Blender



Magica Voxel



Código Fonte

Disponível em https://github.com/humbertodias/unity-projeto-integrador-extra

Executáveis

Disponível para as plataformas: Windows, Mac OS e Android

em

https://github.com/humbertodias/unity-projeto-integrador-extra/releases

Referências

https://unity3d.com

https://www.blender.org

https://ephtracy.github.io

https://www.gimp.org