Unity

Platformer 2D: Adicionando som

créditos imagens flaticon.com

Adicionando som

Adicionando som

O som desempenha um papel fundamental na criação de experiências imersivas em jogos e aplicativos desenvolvidos na Unity. Na Unity, o som é tratado como um componente importante para enriquecer a interação do usuário com o ambiente virtual. Aqui está uma breve explicação sobre como o som funciona na Unity.

Componente de Áudio (Audio Component):
O som é gerenciado através do componente de áudio da Unity. Este componente pode ser anexado a objetos dentro do seu cenário virtual, como personagens, objetos, ou até mesmo câmeras.

Adicionando som

Clips de Áudio (Audio Clips)
Para reproduzir sons, você precisa criar ou importar arquivos de áudio, que são conhecidos como "clips de áudio". Esses clips contêm os dados de áudio que a Unity usará para reproduzir sons específicos. Eles podem variar desde efeitos sonoros simples até músicas de fundo complexas.

Reprodução de Som (Sound Playback)
O componente de áudio pode ser configurado para reproduzir um ou vários clips de áudio em resposta a eventos específicos, como um jogador pressionando um botão, uma colisão acontecendo, ou até mesmo em um loop constante.

Adicionando som

Posicionamento Espacial (Spatial Positioning)
A Unity suporta posicionamento espacial de som, o que significa que o som pode ser ajustado para parecer que está vindo de uma direção específica ou de uma posição no espaço virtual. Isso é fundamental para criar uma sensação de profundidade e realismo em ambientes 3D.

Mixagem de Áudio (Audio Mixing)
A Unity oferece um sistema de mixagem de áudio que permite ajustar o volume, a reverberação e outros efeitos sonoros em tempo real. Isso é útil para criar uma trilha sonora equilibrada e para ajustar a ambientação sonora de acordo com o ambiente do jogo.

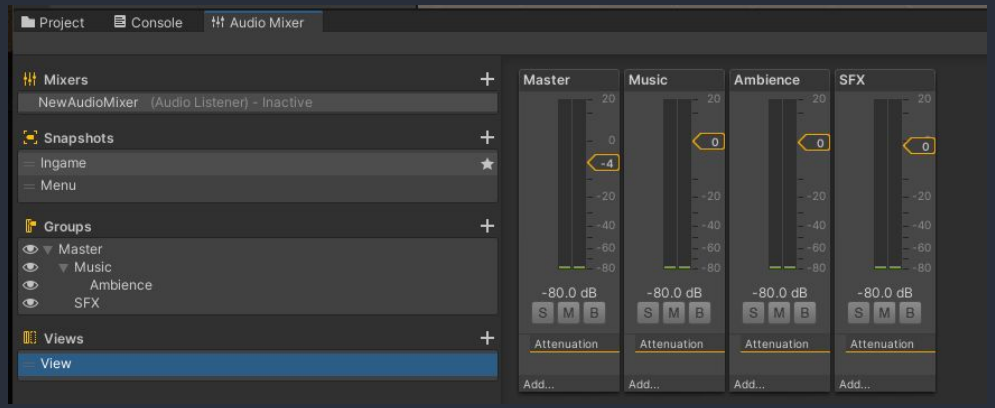
Adicionando som

Efeitos Sonoros (Sound Effects)
Além da reprodução básica de áudio, a Unity também suporta a aplicação de efeitos sonoros em tempo real, como eco, equalização e filtragem para dar mais profundidade e realismo aos sons.

Eventos de Áudio (Audio Events)
A Unity permite que você crie eventos de áudio para gerenciar a reprodução de sons de maneira mais dinâmica e reativa. Isso é útil para sincronizar efeitos sonoros com eventos do jogo.

créditos imagens flaticon.com

Audio mixer

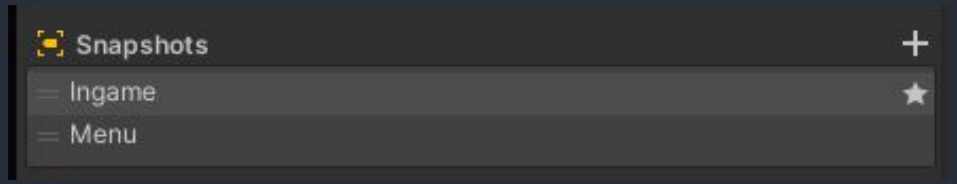


O Audio Mixer é uma ferramenta essencial para controlar e ajustar em tempo real a mixagem de áudio em jogos e aplicativos. Ele permite controlar volume, panorâmica, efeitos sonoros, grupos de áudio e automatização de parâmetros. Além disso, oferece suporte para efeitos de áudio em tempo real, curvas de automatização para controle dinâmico, snapshots para configurações específicas e é compatível com várias plataformas. O Audio Mixer desempenha um papel crucial na criação de uma experiência sonora imersiva e de alta qualidade para os usuários.

Audio mixer

Snapshots

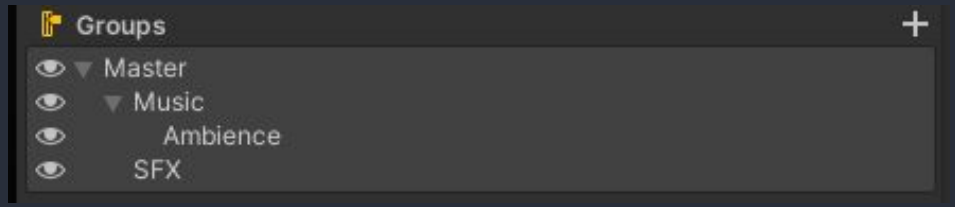
Snapshots são configurações predefinidas de mixagem de áudio no Audio Mixer da Unity. Eles permitem que você salve diferentes estados de mixagem para posteriormente alternar entre eles durante a execução do jogo. Isso é especialmente útil para criar transições suaves entre diferentes configurações de áudio, como alternar entre uma mixagem de áudio para uma cena de ação intensa e outra para uma cena mais tranquila. Os snapshots podem incluir ajustes para volumes, efeitos, curvas de automatização e outras configurações relacionadas ao áudio. Ao criar snapshots, você pode controlar a atmosfera e a qualidade do áudio em tempo real, tornando a experiência do jogador mais imersiva e envolvente.



Audio mixer

Groups

Os grupos de áudio no Audio Mixer da Unity são categorias organizacionais que permitem agrupar sons semelhantes e aplicar ajustes de mixagem a eles de forma coletiva. Esses grupos facilitam o controle e a manipulação de sons relacionados, tornando a mixagem de áudio mais eficiente e organizada. Por exemplo, você pode criar grupos para sons de efeitos sonoros, trilhas sonoras ou vozes dos personagens. Cada grupo pode ter suas próprias configurações de volume, panorâmica, efeitos sonoros e curvas de automatização, permitindo personalizar a mixagem de forma granular para diferentes categorias de sons. Isso é fundamental para alcançar uma experiência sonora equilibrada e imersiva em seu jogo ou aplicativo, garantindo que cada tipo de som seja tratado de acordo com suas necessidades específicas de mixagem.



créditos imagens flaticon.com

Audio mixer

Fantasy SFX

Colocando som

Vamos colocar som ao coletar moedas agora.

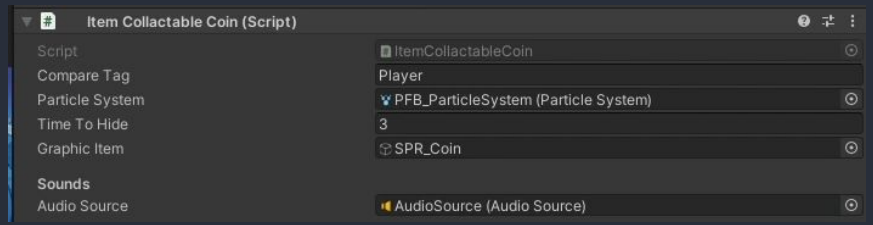
Colocando som

Vamos ao script ItemCollactableBase e vamos adicionar o seguinte:

```
[Header("Sounds")]
public AudioSource audioSource;


protected virtual void OnCollect()
{
    if (particleSystem != null) particleSystem.Play();
    if (audioSource != null) audioSource.Play();
}
```

Colocando som

Abrir o prefab PFB_Coin, e criar dentro dele um gameobject com nome de AudioSource. Adicione também o audio source neste objeto. Mude o som, para o som de coletar moedas que quiser, e adicione a referência ao IttemCollactableCoin.



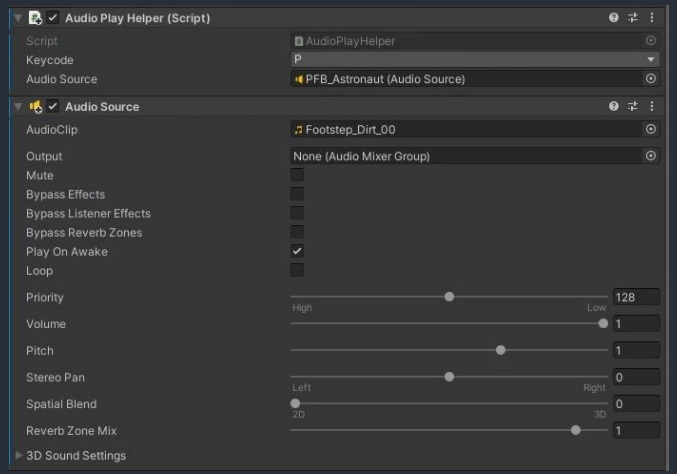
Colocando som de passos

Vamos criar um script que nos ajude a tocar o som dentro de uma animação. Crie um script com nome de AudioPlayerHelper. Adicione o seguinte:

```
public AudioSource audioSource;


public void Play()
{
    audioSource.Play();
}
```

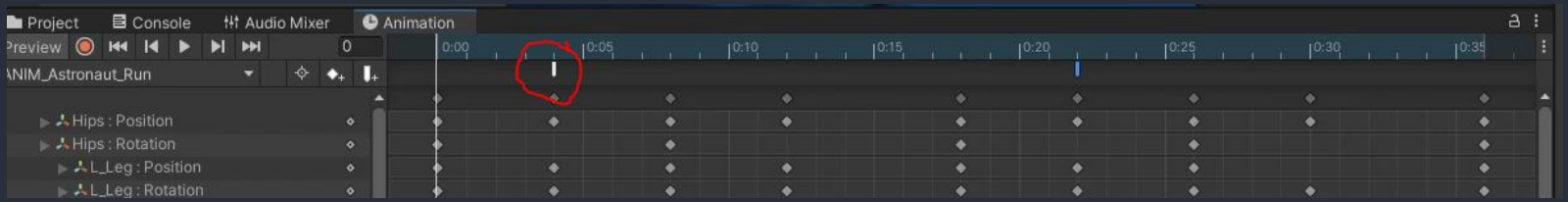
Colocando som de passos

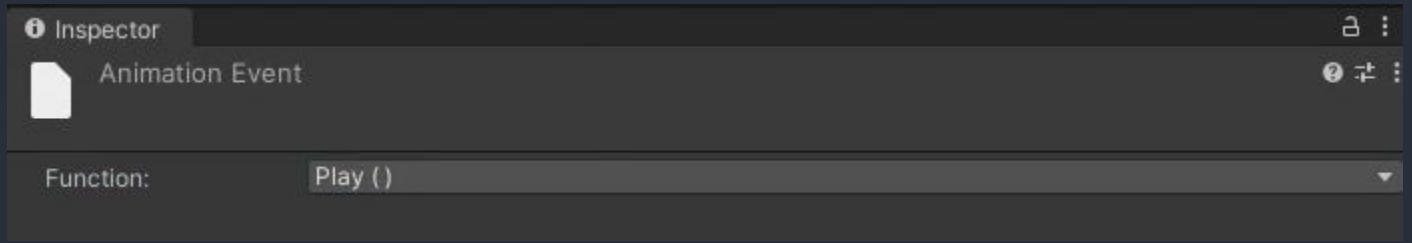
Abre o PFB_Astronaut, e adicione o AudioPlayerHelper e o AudioSource nele. Coloque o Audioclip o audio source correto, e atualize a referência do AudioSource no AudioPlayerHelper.



Colocando som de passos

Adicione um evento na animação, e chame a função de Play().





Colocando som de passos

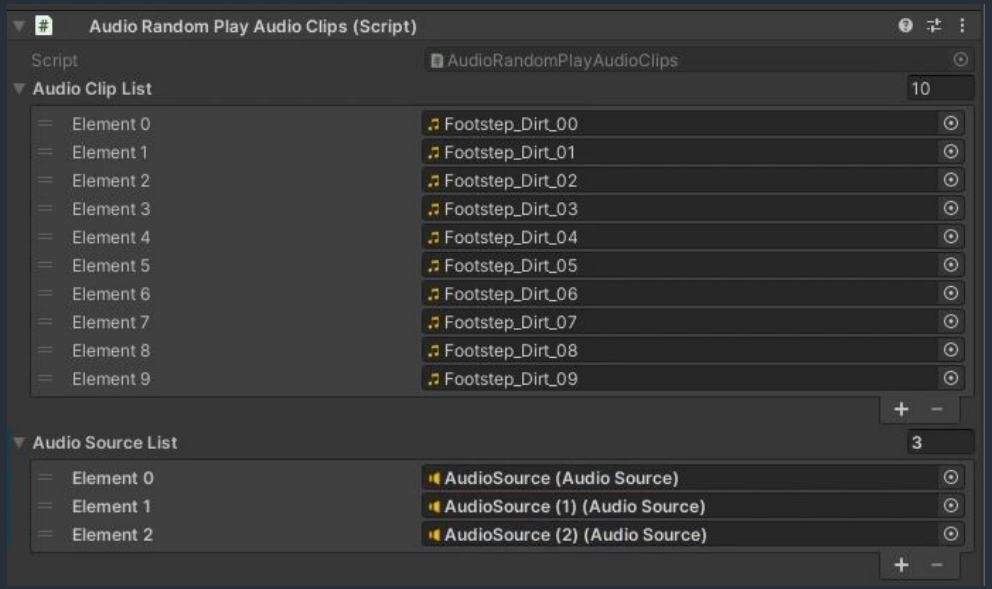
Vamos agora criar uma lista para tocar nossos passos.

Crie um novo script chamado AudioRandomPlayAudioClips. Adicione o seguinte

```
public List<AudioClip> audioClipList;

  public List<AudioSource> audioSourceList;

  private int _index = 0;

  public void PlayRandom()
  {
      if (_index >= audioSourceList.Count) _index = 0;

      var audioSource = audioSourceList[_index];


      audioSource.clip = audioClipList[Random.Range(0, audioClipList.Count)];
      audioSource.Play();

      _index++;
  }
```

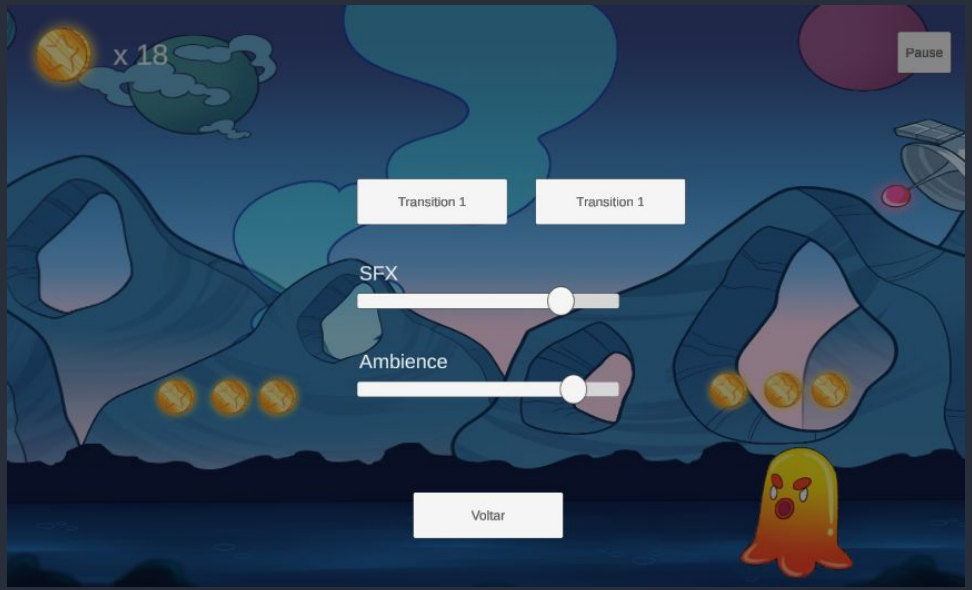
Colocando som de passos

Agora vamos adicionar uma lista de audio clips, na lista AudioClipList. Crie alguns audio source, como mostrado em aula, e atualize as referências também. Atualize também a chamada do método no evento da animação.



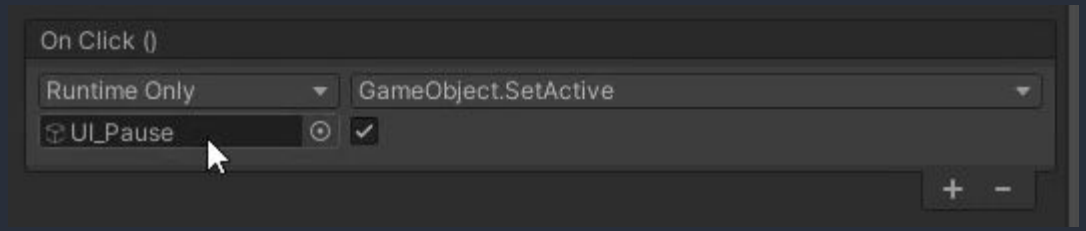
Menu de pause

Vamos adicionar algumas configurações de som no menu de pause. Crie um menu com 2 sliders, e 2 botões para transição de som, como na imagem abaixo.



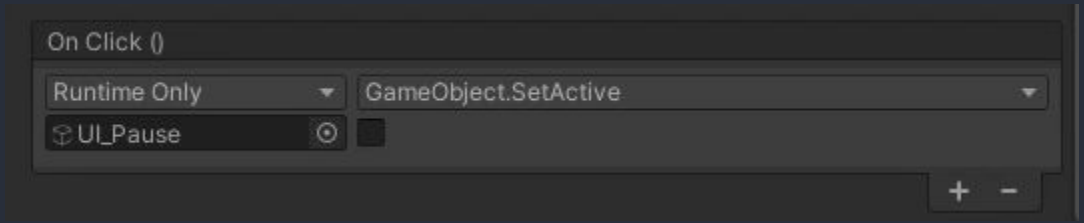
Menu de pause

Deixe o gameobject desligado por default. No botão de Pause, ligue a UI quando receber o evento de OnClick()



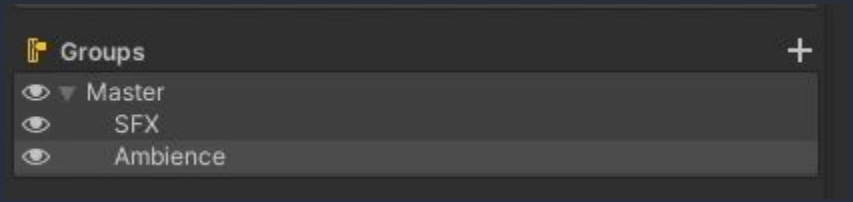
Menu de pause

E no botão voltar, dentro do menu de pause, desligue o menu todo o menu:



Menu de pause

Crie um AudioMixer, e monte dois grupos: SFX e Ambience.



Menu de pause

Em snapshots, crie 2 estados diferentes:

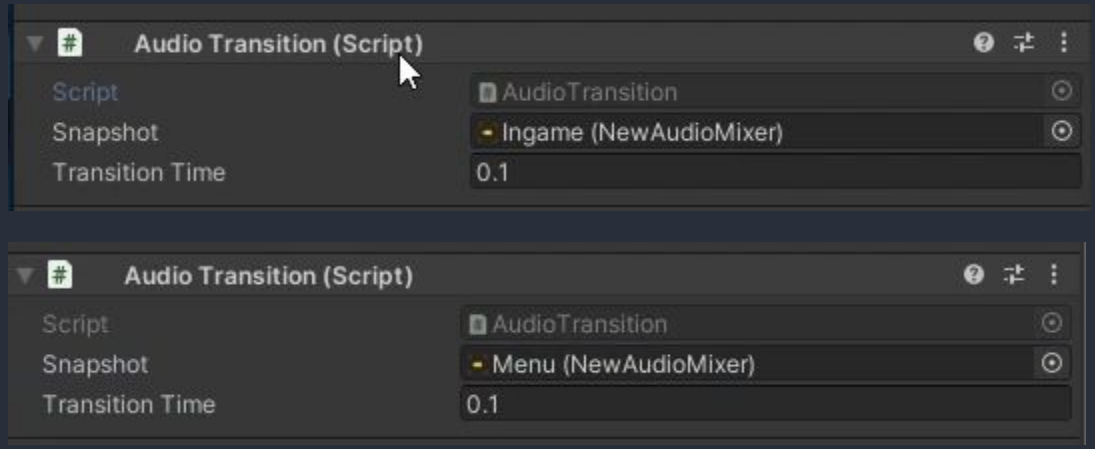
Menu de pause

Crie um script com o nome de AudioTransition. Adicione o seguinte código:

```
public AudioMixerSnapshot snapshot;
 public float transitionTime = .1f;

 public void MakeTransition()
 {
     snapshot.TransitionTo(transitionTime);
 }
```

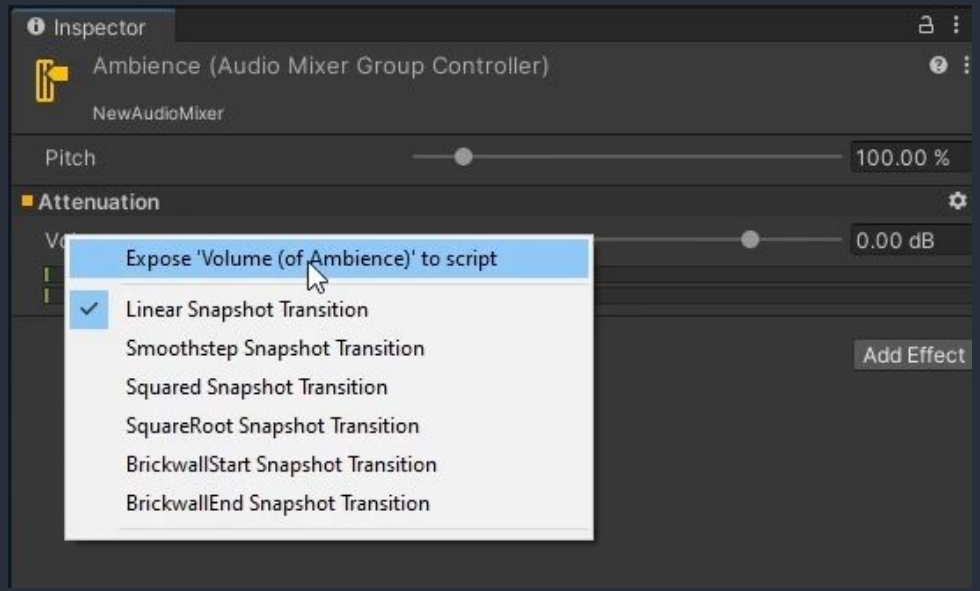
Menu de pause

Adicione o script AudioTransition nos dois botões de transição, e faça o setup de acordo com a imagem abaixo.



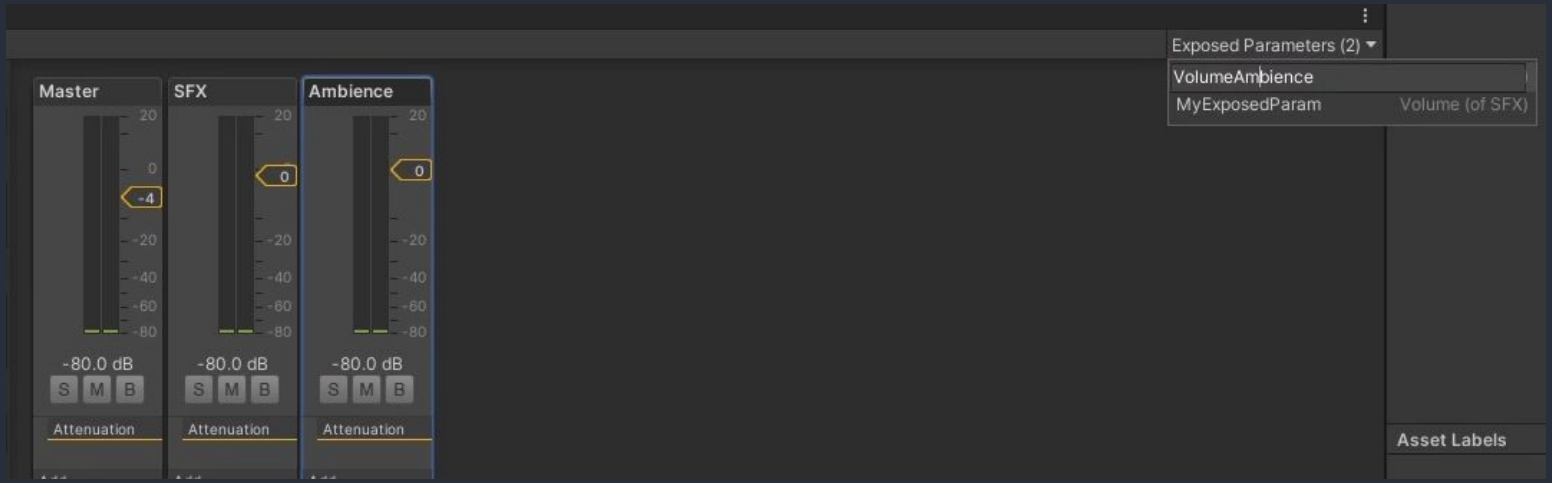
Menu de pause

No audio mixer, em cada grupo criado, exponha o valor **Volume**.



Menu de pause

Você pode ver os parâmetros expostos aqui:



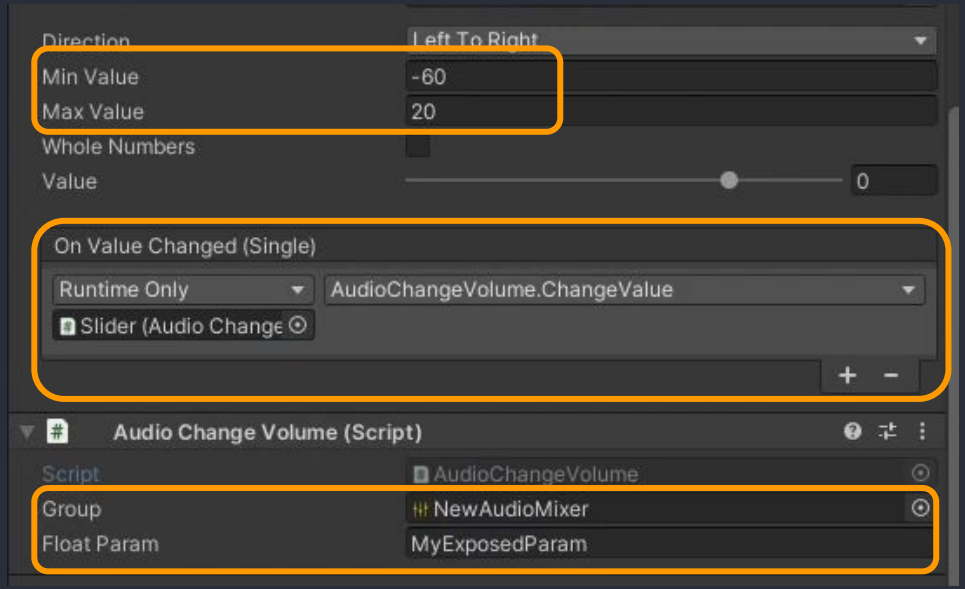
Menu de pause

Para fazer o setup desses valores em tempo real, crie um script com nome de AudioChangeVolume, e adicione o seguinte:

```
public AudioMixer group;
public string floatParam = "MyExposedParam";

public void ChangeValue(float f)
{
    group.SetFloat(floatParam, f);
}
```

Menu de pause

Em cada slider, adicione o script criado. Faça o setup de acordo com a configuração feita e necessária:



Transição ingame

Para fazermos uma transição dentro do jogo, crie um novo script com nome de AudioTriggerTransition. Então, adicione o código:

```
public AudioMixerSnapshot snapshot01;
public AudioMixerSnapshot snapshot02;

public string tagToCompare = "Player";

private void OnTriggerEnter2D(Collider2D collision)
{
    if(collision.transform.CompareTag(tagToCompare))
    {
        snapshot02.TransitionTo(.1f);
    }
}

private void OnTriggerExit2D(Collider2D collision)
{
    if (collision.transform.CompareTag(tagToCompare))
    {
        snapshot01.TransitionTo(.1f);
    }
}
```

Transição ingame

Crie um trigger com boxCollider2d, adicione o código criado, e faça os ajustes como mostrado em aula.