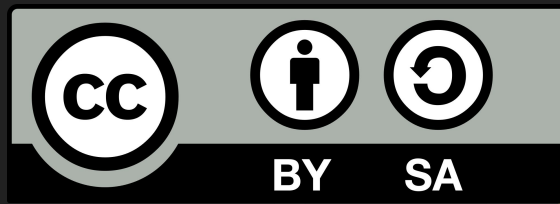


Introdução aos Scriptable Objects

Leonardo Tórtoro Pereira (leonardop@usp.br)



Este material é uma criação do
Time de Ensino de Desenvolvimento de Jogos
Eletrônicos (TEDJE)

Filiado ao grupo de cultura e extensão
Fellowship of the Game (FoG), vinculado ao
ICMC - USP

Este material possui licença CC By-SA. Mais informações em:
<https://creativecommons.org/licenses/by-sa/4.0/legalcode>



Objetivos

- Definir scriptable objects
- Mostrar suas principais vantagens e desvantagens
- Apresentar algumas de suas principais utilidades
- Apresentar exemplos práticos

O que é Scriptable
Objects?

Scriptable Objects - É de comer?

- Não... Mas dá pra colocar as propriedades da comida do seu jogo :)
- É uma classe serializável da Unity
- Armazena grandes quantidades de dados compartilhados independentemente das instâncias de script
- Pode ser salva como .asset

Scriptable Objects - Onde usar?

- Arquivos de configuração de jogo
- Inventários
- Atributos de Inimigos
- Coleção de Áudio

Scriptable Objects - Como usar?

- Variáveis
- Eventos
- Sistemas

Scriptable Objects - Por que usar?

- Facilita gerenciamento de alterações e depuração
- Comunicação flexível entre diferentes sistemas do jogo
- Mais fácil de adaptar e alterar, e reutilizar componentes
- Pode ser uma boa substituição a singletons

Singletons?

Singletons?

```
void Awake()  
{  
    if (instance == null)  
    {  
        instance = this;  
        DontDestroyOnLoad(gameObject);  
        audioSrc = GetComponent<AudioSource>();  
        ...  
    }  
    else if (instance != this)  
    {  
        Destroy(gameObject);  
    }  
}
```

Singletons?

→ Em algum outro código...

```
Player.instance.keys.Clear();  
Player.instance.usedKeys.Clear();  
Player.instance.AdjustCamera(map.startX, map.startY);  
Player.instance.SetRoom(map.startX, map.startY);
```

Singletons - Vantagens

- É fácil de fazer
- Acessa qualquer coisa importante em qualquer classe
- Estado persistente (resiste a mudança de cenas)
- Fácil de entender
- Padrão consistente
- Fácil de planejar
- ***Só funciona bem pra projetos bem pequenos e que você não vai mais mudar depois de pronto**

Singletons - Desvantagens

- Conexões “rígidas”
 - ◆ Não é modular
- Não aceita polimorfismo
- Não é “testável”
- Pesadelo de dependências
 - ◆ Desista de paralelizar
- Uma única instância

Lembra disso?

→ Em algum outro código...

```
Player.instance.keys.Clear();  
Player.instance.usedKeys.Clear();  
Player.instance.AdjustCamera(map.startX, map.startY);  
Player.instance.SetRoom(map.startX, map.startY);
```

Lembra disso?

→ Em algum outro código...

```
Player.instance.keys.Clear();  
Player.instance.usedKeys.Clear();  
Player.instance.AdjustCamera(map.startX, map.startY);  
Player.instance.SetRoom(map.startX, map.startY);
```

→ Estava dentro de outro Singleton...

Gerenciadores Singletons - Objetivos

- Rastrear todos os inimigos de uma cena
- Entender o jogador
- Enviar comandos

Gerenciadores Singletons - Problemas

- Race Conditions
 - ◆ Problemas de null reference e de paralelismo
- Singleton rígido
 - ◆ Difícil de testar
- Só 1 (ou muitos...)

Voltando aos 50s

Scriptable Objects - Como usar?

- Variáveis
- Eventos
- Sistemas

Scriptable Objects - Variáveis

```
[CreateAssetMenu]
public class FloatVariable : ScriptableObject
{
    public float Value;

    public void SetValue(float value)
    {
        Value = value;
    }
    ...
    public void ApplyChange(float amount)
    {
        Value += amount;
    }
}
```

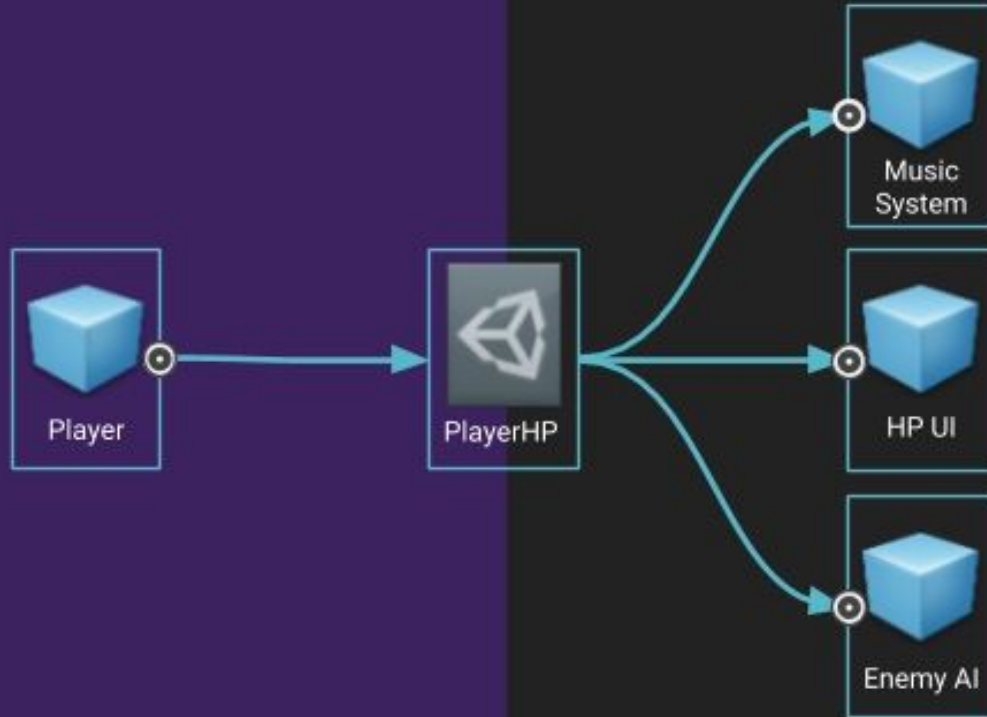
Scriptable Objects - Variáveis

- Vida
- Dano
- Dados de tempo

Scriptable Objects - Variáveis

```
public class DamageDealer : MonoBehaviour
{
    public FloatVariable DamageAmount;
}
```

Scriptable Objects - Variáveis



Demo - Unity

<https://github.com/roboryantron/Unite2017>

Cena: HP 1

Scriptable Objects - Como usar?

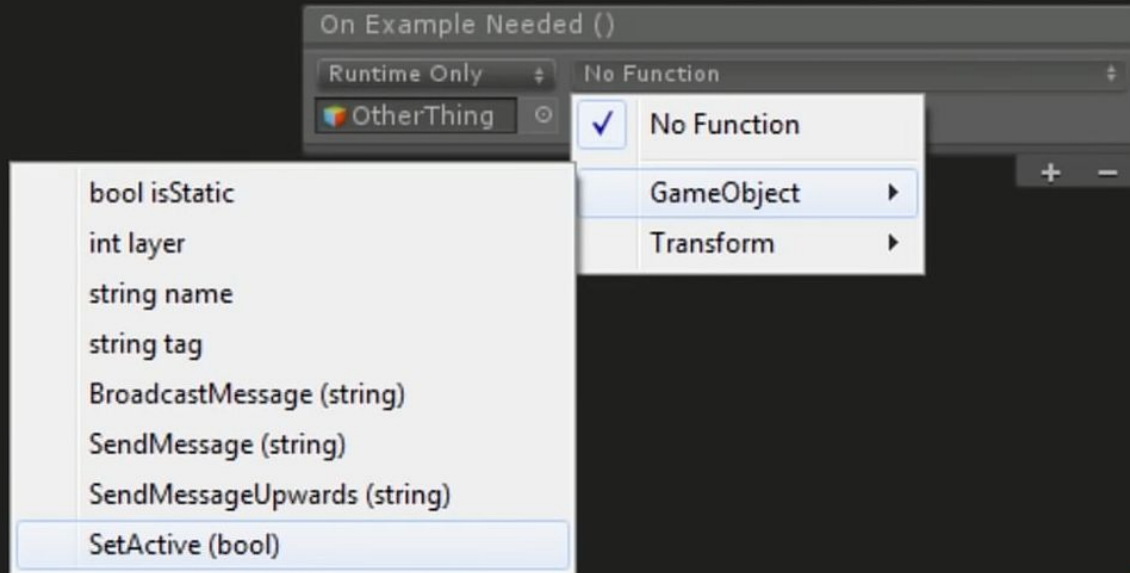
- Variáveis
- Eventos
- Sistemas

Scriptable Objects - Eventos

- Modularizam sistemas
 - ◆ Reúso em outros projetos
 - ◆ Isola prefabs
 - Responde a um evento, não importa de onde vem
 - ◆ Só executa quando precisa
 - Não precisa ficar com “ifs” checando variável
 - ◆ Fácil de debuggar (se você sabe bem)

Scriptable Objects - Eventos

UnityEvent



Scriptable Objects - Eventos

- Pode colocar no editor
- Menos código, não precisa assumir que o código está em algum componente específico
- Pode passar argumentos
 - ◆ `Extend UnityEvent<T>`
 - ◆ Estáticos ou Dinâmicos

Scriptable Objects - Eventos

→ Problemas

- ◆ Ligações rígidas
 - Botão precisa de referência “hard” do que ele responde
- ◆ Serialização limitada
- ◆ Alocação de lixo
 - Não criar evento chamado todo *frame*

Scriptable Objects - Eventos

```
public class GameEvent : ScriptableObject
{
    private readonly List<GameEventListener> eventListeners =
        new List<GameEventListener>();

    public void Raise()
    {
        for(int i = eventListeners.Count - 1; i >= 0; i--)
            eventListeners[i].OnEventRaised();
    }
    public void RegisterListener(GameEventListener listener)
    {
        if (!eventListeners.Contains(listener))
            eventListeners.Add(listener);
    }
    public void UnregisterListener(GameEventListener listener)...
}
```

Scriptable Objects - Eventos

```
public class GameEventListener : MonoBehaviour
{
    public GameEvent Event; //Pode ser referenciado
    public UnityEvent Response; //Serializa a chamada de função como resposta

    private void OnEnable()
    { Event.RegisterListener(this);}

    private void OnDisable()
    { Event.UnregisterListener(this);}

    public void OnEventRaised()
    {
        Response.Invoke();
    }
}
```

Demo - Unity

<https://github.com/roboryantron/Unite2017>

Cena: HP 2*****

Extras

Scriptable Objects - Runtime Sets

- Solucionar problema de gerenciadores singletons
- *Runtime Sets*
- Manter controle de uma lista de objetos na cena
- Evita *race conditions*
- Mais flexível que tags da Unity e singleton

Scriptable Objects - Runtime Sets

```
public abstract class RuntimeSet<T> : ScriptableObject
{
    public List<T> Items = new List<T>();

    public void Add(T thing)
    {
        if (!Items.Contains(thing))
            Items.Add(thing);
    }

    public void Remove(T thing)
    {
        if (Items.Contains(thing))
            Items.Remove(thing);
    }
}
```

Scriptable Objects - Runtime Sets

- Um sistema que sabe todas as entidades de um RTS
 - ◆ Unidades, construções...
 - ◆ Pode adicionar um novo conjunto no fim do projeto
- Renderizadores para efeitos especiais
 - ◆ Script de renderização precisa saber de todos os efeitos
 - ◆ Câmera lia todos os itens de renderização adicionados
- Itens em um mapa

Demo - Unity

<https://github.com/roboryantron/Unite2017>

Cena: Sets

Scriptable Objects - Enums

- Precisa mudar no código
- Difícil de remover/reordenar
- Não podem ter dados adicionais

Demo - Unity

<https://github.com/roboryantron/Unite2017>

Cena: Elements

Voltando ao trio...

Scriptable Objects - Como usar?

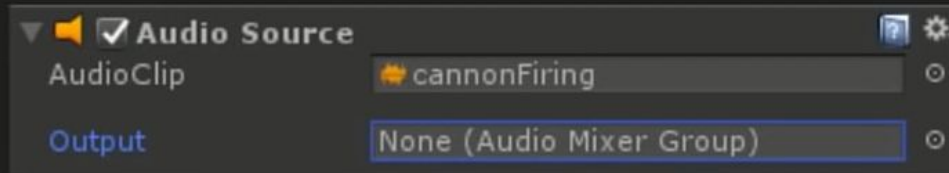
- Variáveis
- Eventos
- Sistemas

Scriptable Objects - Sistemas

- SOs podem ter funções dentro
 - ◆ Não é muito comum. Mas não é errado!
- Sistema é um Asset de SO no projeto
- Referência direta com o inspetor
- Não precisa procurar coisas em código
- Sem referências que só existem na cena
- Como AudioManager/AudioMixerGroup

Scriptable Objects - Sistemas

project



Scriptable Objects - Sistemas

→ Inventário

- ◆ SO: Master List
 - Todos os itens disponíveis
- ◆ 1 SO por item
- ◆ Busca que acontece na validação da Master List
 - Popula ela com todos os itens
- ◆ Usa diferentes inventários em diferentes cenas
 - Como num tutorial

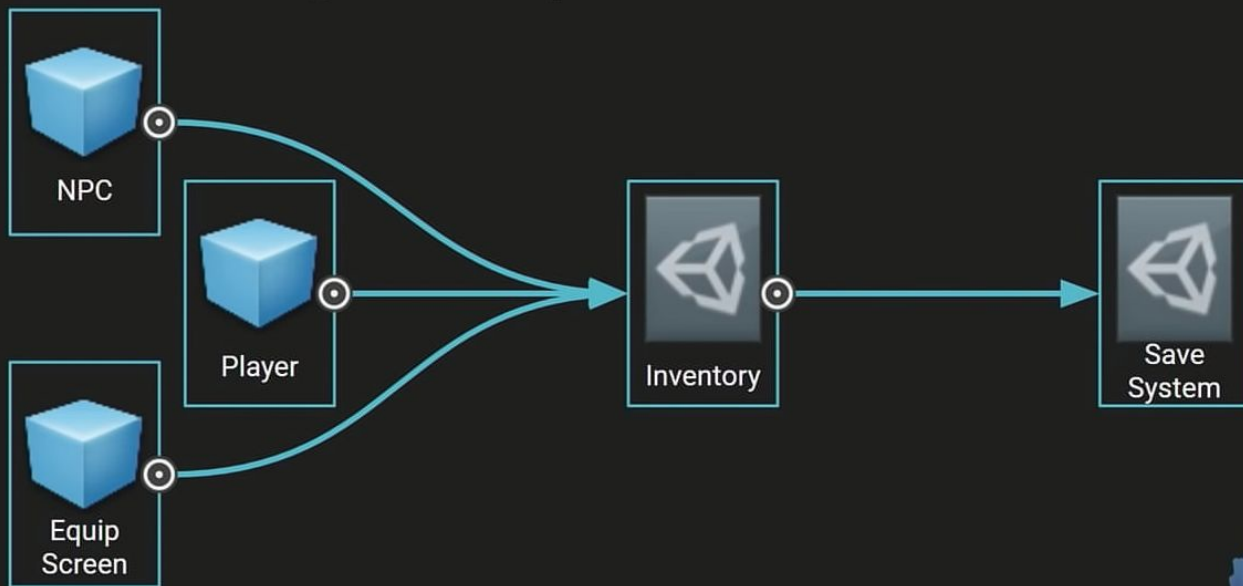
Scriptable Objects - Sistemas

→ Inventário

- ◆ SO: Master List
 - Todos os itens disponíveis
- ◆ 1 SO por item
- ◆ Busca que acontece na validação da Master List
 - Popula ela com todos os itens
- ◆ Usa diferentes inventários em diferentes cenas
 - Como num tutorial

Scriptable Objects - Sistemas

Inventory Example



Dúvidas?

Vamos a um exemplo
mais simples:

<https://www.raywenderlich.com/2826197-scriptableobject-tutorial-getting-started>

Referências

Referências

[1] Unite Austin 2017 - Game Architecture with Scriptable Objects, Ryan Hipple (Schell Games) -

https://www.youtube.com/watch?v=raQ3iHhE_Kk

[2] ScriptableObject Tutorial: Getting Started, Jeff Fisher & Ben MacKinnon -

<https://www.raywenderlich.com/2826197-scriptableobject-tutorial-getting-started>

[3] 3 maneiras legais de construir seu jogo com Objetos Scriptáveis, Unity3d -

<https://unity3d.com/pt/how-to/architect-with-scriptable-objects>

Extra:

Exemplo de um sistema de habilidades com SOs -

<https://learn.unity.com/tutorial/create-an-ability-system-with-scriptable-objects#>