**09/07/2017**
**Roteiro - Workshop Interno de ScriptableObject**
**Responsável:** Vinícius Garcia
**Duração:** 2 horas

---

**Considerações Iniciais**

- Vai ser pesado, lotado de coisas que eles provavelmente nunca ouviram falar.
- Não é nenhum conhecimento arcano, as coisas são bastante simples. Os conceitos que são meio traiçoeiros.
- Basta que prestem atenção e recomendo anotarem para perguntar algo depois, caso algo lhes deixe curiosos.

**Motivação**

- Por que usar Scriptable Object?
    - Armazenamento de informação
        - Mostrar projeto com JSON (miçanga tycoon?)
    - Compartilha dados entre objetos
        - Memória (gameobject com 1 milhão de ints)
    - Salva permanentemente
        - Não precisa ficar dando reload
    - Visualização
- Coisas legais que podem ser feitas com Scriptable Object
    - VCS (Praticidade)
    - Noir (Versatilidade)
    - Cyrus (Customização)

**Implementação**

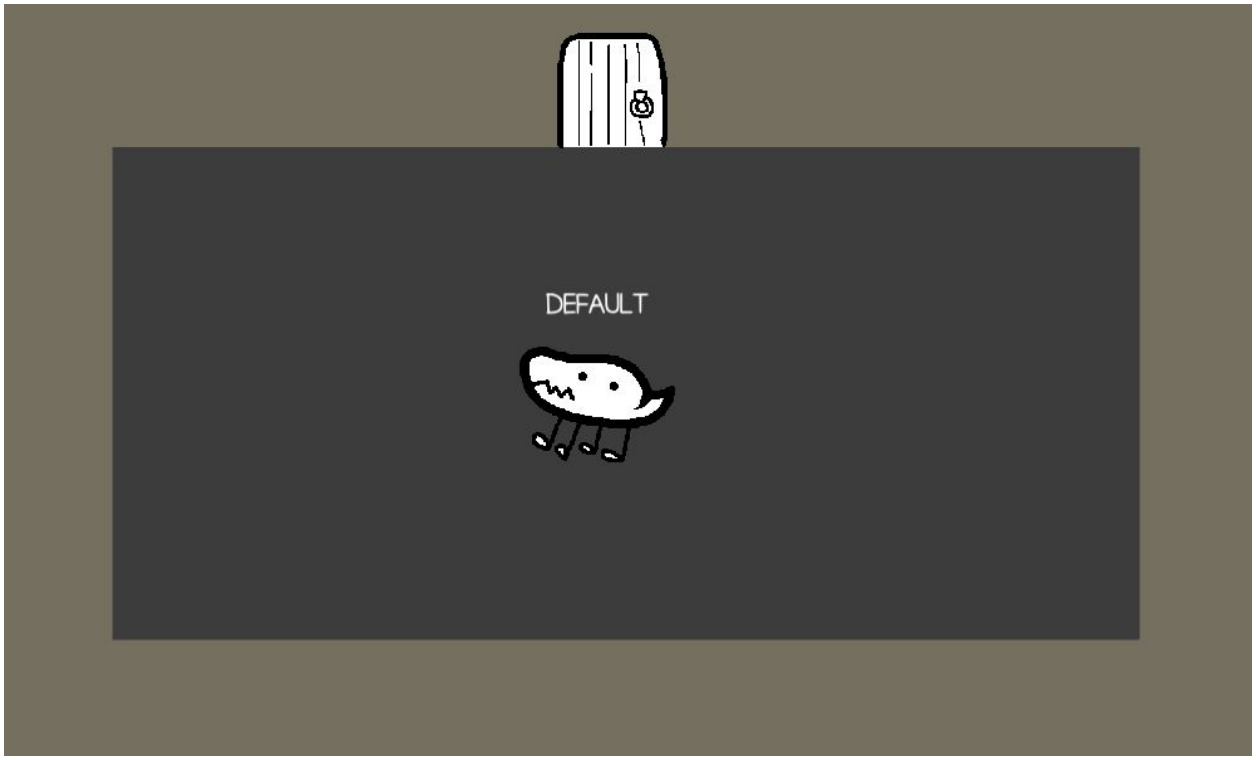
- ScriptableObject
    - Exemplo principal: inimigo que pode receber ScriptableObjects que ele dá parse. Atributos: sprite, velocidade, nome. (mais detalhes no final)
- Attributes Tips
    - [Header], [Range], [Tooltip], [ExecuteInEditMode]
- System.Serializable
    - Às vezes, você não quer um conjunto de dados 'individual' para cada dado. Nesses casos, é mais intuitivo preparar conjuntos de dados 'descartáveis'.
    - Exemplo: BUFF (possui uma lista de {`EFEITO, DURAÇÃO`}). Implementar!

**Bibliografia**


- https://www.youtube.com/watch?v=VBA1QCoEAX4
- https://docs.unity3d.com/Manual/class-ScriptableObject.html
- https://www.youtube.com/watch?v=ltZbTYO0Mnw

**Seção de Dúvidas**

---




**Flow de Implementação do Projeto-Exemplo**

1. Apresentar projeto atual
    1.1. Inimigo que anda aleatoriamente (não nos focaremos no código dele andando). Em resumo: ele escolhe um ponto e vai até ele. A distância máxima desse ponto é o atributo `distance` e a velocidade com a qual ele vai até esse ponto é o atributo `speed`.
    1.2. Mostrar que mexer nas variáveis realmente altera o comportamento do bicho.
    1.3. Mostrar que a modificação dessas variáveis é perdida quando o jogo é fechado.

2. Problema: quero uma forma mais fácil de editar certas características do bicho.
    2.1. Não vou me focar em justificar o uso do ScriptableObject - em um projeto pequeno como esse, não faz muita diferença.
    2.2. A motivação já foi dada na seção (longa) anterior.

3. Apresentar: vou criar duas coisas
    3.1. `EnemyData` - armazenará as características da criatura que quero mudar. Vai ser o ScriptableObject.
    3.2. `EnemyDataHandler` - será responsável por intermediar a comunicação entre `Enemy` e `EnemyData`. Eu poderia fazer isso dentro de `Enemy` mas quero manter

encapsulado para não confundir as pessoas com a lógica interna da classe `Enemy` (movimentação).

4. Efetivamente criar e programar as duas classes.
    4.1. `EnemyData`
        4.1.1. Deve possuir os campos `name, color, speed, distance`.
    4.2. `EnemyDataHandler`
        4.2.1. Recebe `name`, troca o nome da label.
        4.2.2. Recebe `color`, troca a cor do bicho.
        4.2.3. Recebe `speed`, troca a velocidade do bicho.
        4.2.4. Recebe `distance`, troca a distância máxima de movimentação do bicho em um certo ciclo.

5. Apresentar o resultado do uso de ScriptableObject
    5.1. Funciona!
    5.2. Alterar em *runtime* mantém as modificações quando o jogo fecha.
        5.2.1. Reforçar que está alterando um arquivo.
    5.3. Botar velocidade muito alta para o bicho sair voando.
        5.3.1. Reforçar que podemos manter as modificações. Se eu der Play, o bicho sai voando novamente.
    5.4. Criar outra instância de ScriptableObject diferente e mudar de uma para a outra em *runtime*.

6. Apresentar o uso de Attributes, utilizando o ScriptableObject como exemplo
    6.1. [Header]
        6.1.1. Separar em *Enemy Appearances* e *Enemy Movement*
    6.2. [Tooltip]
        6.2.1. Botar uma descrição em `speed` e `distance`.
    6.3. [Range]
        6.3.1. Botar um limite [0f, 5f] em `speed` e `distance`.

7. Apresentar uma palhinha de CustomEditor
    7.1. Criar classe `EnemyDataHandlerCustomEditor`.
    7.2. Botar um botão chamado *Generate Color*, que gera uma cor aleatória para `EnemyData`.

8. [ExecuteInEditMode]
    8.1. Botar em `EnemyDataHandler` para atualizar em tempo real, mostrando que faz as alterações serem mostradas mesmo no Editor.

9. Apresentar a motivação para [System.Serializable]
    9.1. Um exemplo bem diferente do que estávamos vendo até agora, mas ainda assim um exemplo comum. Vou pedir a capacidade de abstração de cada um.

9.2. Às vezes, você não quer um conjunto de dados ‘individual’ para cada dado. Nesses casos, é mais intuitivo preparar conjuntos de dados ‘descartáveis’.
9.3. Exemplo: uma Skill. Essa Skill possui uma lista de efeitos, sendo que cada efeito tem uma duração. Como você faria isso? Perguntar pras pessoas.

10. Criar a classe `SkillData`, que é um ScriptableObject
    10.1. Criar subclasse `Effect`, com o float `duration` e o Enum `type` (que pode ser *BUFF* ou *DEBUFF*).
    10.2. Fazer uma lista de `Effect` e mostrar que não aparece no Inspector.
    10.3. Solução não-ideal: criar duas listas, uma de floats e outra de types.
        10.3.1. Não preciso nem dizer que isso é péssimo! Vai ter que verificar manualmente que as duas listas estão sincronizadas.
        10.3.2. Deus me livre se quiser mudar a ordem das coisas.
        10.3.3. Ou se quiser adicionar um novo atributo. Outra lista.
    10.4. Solução não-ideal: transformar Effect em um ScriptableObject e mostrar que funciona.
        10.4.1. É uma classe tão pequenininha, e não ‘identifica’ nada. Qual nome você botaria?
        10.4.2. Você vai ter que ter um arquivo prum Buff com cada duração que surgir, e o fluxo vai ser: criar arquivo, editar ele e depois passar pro ScriptableObject da skill. Olha quanto tempo perdido e perfeito pra se perder.
    10.5. Solução: [System.Serializable]. Simples assim!
        10.5.1. Meio feio, mas podemos também fazer um CustomEditor bonitinho para ele.

11. Apresentar CustomEditor pro `SkillData`.
    11.1. Já está pronto, basta descomentar o cabeçalho.

12. Solucionar dúvidas finais.