[Documentação em Português-BR - Versão Final]

Eventus: O Hub de Comunicação Unificado

Versão: 1.0.0
Criado por: David Souza (twoBrackets)

1. Filosofia

Eventus é um sistema de comunicação profissional e centralizado, projetado para criar arquiteturas altamente desacopladas e dinâmicas em seu projeto Unity. Ele elimina a necessidade de referências diretas entre componentes, prevenindo o "código espaguete" e tornando seu projeto mais limpo, escalável e de fácil manutenção.

O Eventus combina com maestria dois padrões de comunicação essenciais em uma única e elegante API:

1. **Event Bus (Sistema de "Push"):** Para comunicação reativa, baseada em notificações. Componentes Publicam um evento (ex: OnPlayerCrashed) para anunciar que algo aconteceu, e qualquer componente interessado pode se Inscrever para reagir a ele. É ideal para ações e ocorrências únicas.
2. **Data Hub (Blackboard / Sistema de "Pull"):** Para estado compartilhado em tempo real. Componentes Escrevem dados (ex: velocidade atual da moto) em um "quadro-negro" central, e qualquer outro componente pode Ler esses dados a qualquer momento. É perfeito para valores que mudam a cada frame, como dados para a UI.

Todo o sistema é gerenciado através da intuitiva janela de editor **Eventus Hub**, dando a você controle visual total sobre os canais de comunicação do seu projeto, sem nunca precisar editar manualmente o código por trás.

2. Guia de Início Rápido

Coloque o Eventus para funcionar em seu projeto em menos de 5 minutos.

1. **Adicione o Eventus à sua Cena:**
   - Crie um GameObject vazio em sua cena principal ou de bootstrap e nomeie-o Eventus.
   - Anexe o script Eventus.cs a ele.
   - (Recomendado) Salve este objeto como um Prefab para uso fácil em outras cenas.
2. **Defina a Ordem de Execução de Scripts (Crucial!):**
   - Vá em Edit -> Project Settings -> Script Execution Order.
   - Clique no botão + e selecione o script Eventus.

- Defina seu valor para um número negativo, como -100. Isso garante que o Eventus esteja sempre pronto antes que qualquer outro script tente usá-lo, prevenindo condições de corrida.

3. **Abra o Eventus Hub:**
   - Vá no menu da Unity: Tools -> Eventus -> Open Window.
   - Esta janela é seu centro de comando para todos os canais de comunicação.
4. **Crie Seus Primeiros Canais:**
   - Navegue para a aba **"Channels"** no Eventus Hub.
   - Na seção inferior, insira um nome para seu novo canal (ex: PlayerHealth).
   - Selecione seus atributos:
     - **AsData:** Para dados que serão lidos/escritos (Read/Write).
     - **AsEvent:** Para notificações que serão publicadas/inscritas (Publish/Subscribe).
     - **DisallowNull:** (Apenas para AsData) Impede que valores null sejam escritos neste canal, garantindo a integridade dos dados.
   - Clique em "Add" para adicioná-lo à lista.
   - Clique em **"Save Changes"** para gerar o código e recompilar.
5. **Use em Seus Scripts!**

```
using EventusAsset;
using EventusAsset.Core;
using UnityEngine;

public class Player : MonoBehaviour
{
    private int _health = 100;

    void Start()
    {
        // Write initial health to the Data Hub
        Eventus.Write<int>(Channel.PlayerHealth, _health);

        // Subscribe to a damage event
        Eventus.Subscribe<int>(Channel.OnTakeDamage, TakeDamage);
    }

    private void OnDisable()
    {
        // Always unsubscribe to prevent memory leaks!
        Eventus.Unsubscribe<int>(Channel.OnTakeDamage, TakeDamage);
    }

    private void TakeDamage(int amount)
    {
        _health -= amount;
        // Update the value in the Data Hub for other systems to read
        Eventus.Write<int>(Channel.PlayerHealth, _health);
    }
}
```

3. A Janela de Editor Eventus Hub

Aba Home

Uma tela de boas-vindas com links rápidos para a documentação oficial em inglês e português.

Aba Channels

Este é o núcleo do gerenciamento do sistema.

- **Lista de Canais:** Exibe todos os canais definidos atualmente em ordem alfabética.
- **Atributos:** Cada canal mostra seus atributos ([Event], [Data], [No Null]) para referência rápida.
- **Busca:** Uma barra de busca poderosa que permite filtrar a lista por nome de canal ou por atributo (ex: digitar "event" mostrará todos os canais de evento).

- **Adicionar/Remover:** Adicione novos canais ou remova existentes facilmente.
- **Salvar Alterações:** Confirma suas mudanças, regerando o arquivo Channel.cs e recompilando seu projeto. Um aviso "Unsaved" aparecerá no título da janela se você tiver alterações pendentes.
- **Reverter Alterações:** Descarta quaisquer alterações não salvas que você tenha feito, restaurando a lista para seu último estado salvo.

4. Referência da API

Data Hub (Sistema Blackboard)

- Eventus.Write<T>(Channel dataKey, T value)
  - Escreve ou atualiza um valor no Data Hub.
- Eventus.Read<T>(Channel dataKey)
  - Lê um valor do tipo T do Data Hub. Retorna default(T) se a chave não for encontrada ou o tipo estiver incorreto.

Event Bus (Sistema de Notificação)

- Eventus.Publish(Channel type)
  - Dispara um evento sem parâmetros.
- Eventus.Publish<T>(Channel type, T data)
  - Dispara um evento com uma carga de dados do tipo T.
- Eventus.Subscribe(Channel type, Action listener)
  - Inscreve um método a um evento sem parâmetros.
- Eventus.Subscribe<T>(Channel type, Action<T> listener)
  - Inscreve um método a um evento com uma carga de dados do tipo T.
- Eventus.Unsubscribe(Channel type, Action listener) / Eventus.Unsubscribe<T>(Channel type, Action<T> listener)
  - **Crucial:** Desinscreve um método para prevenir vazamentos de memória. Normalmente deve ser chamado em OnDisable.