
mondoDX™

mondoDX™

CANTIL Framework v1.3

15/Agosto/2020

Cantil Mantra

Organização + Simplicidade + Clareza + Componentização Adequada = Eficiência e Robustes

Componentes

mondoDX™

Estrutura de Código

Padrão de Código

Componentes

Componentes

mondoDX™

Estrutura de Código

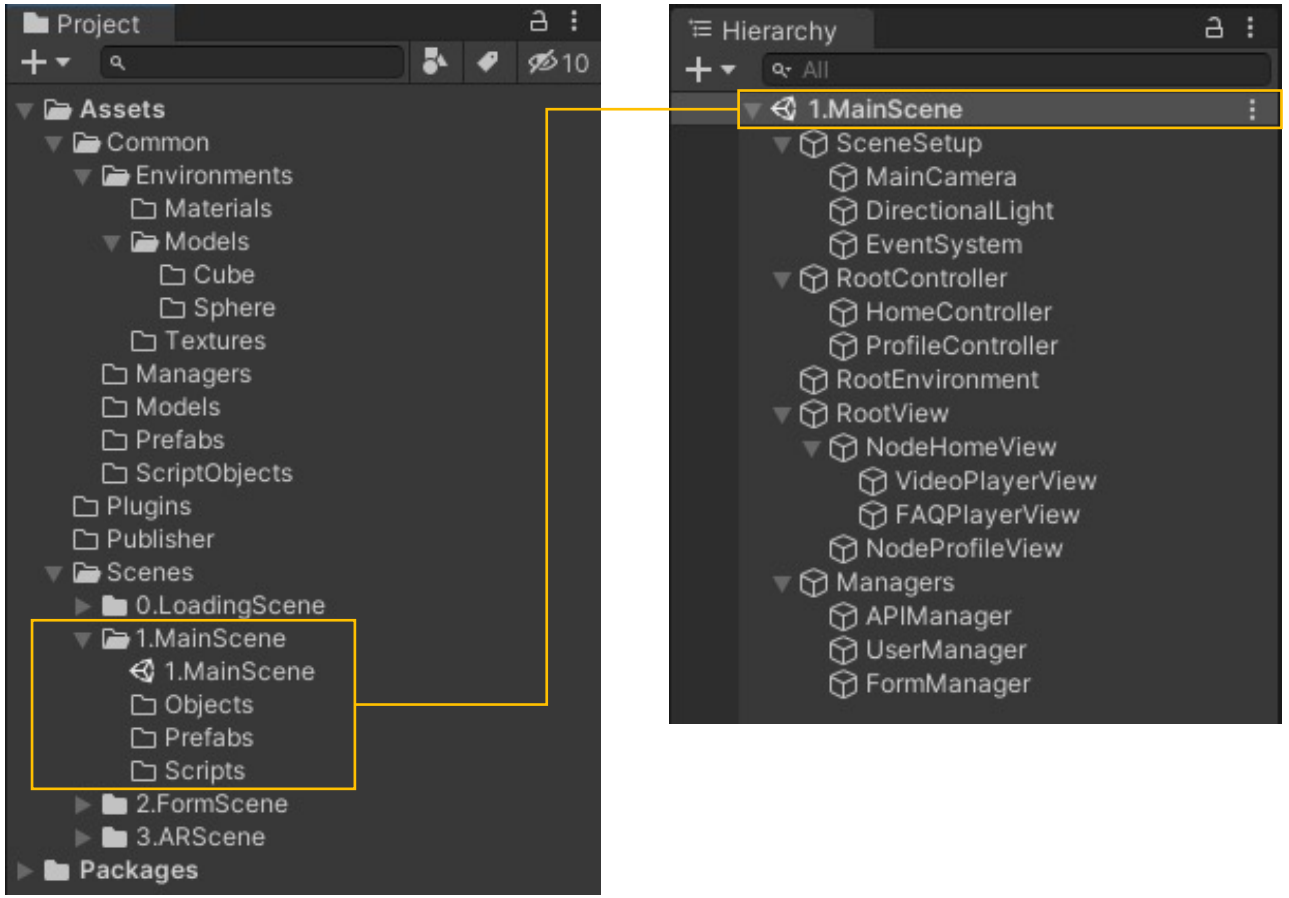
Padrão de Código

Componentes

Estrutura de Código

Cantil é um Micro-framework para estruturar aplicações móveis (mobile apps) geradas em ambientes híbridos de código visual e escrito (visual e scrippting) baseado em scripts, como os Unity3D engine.

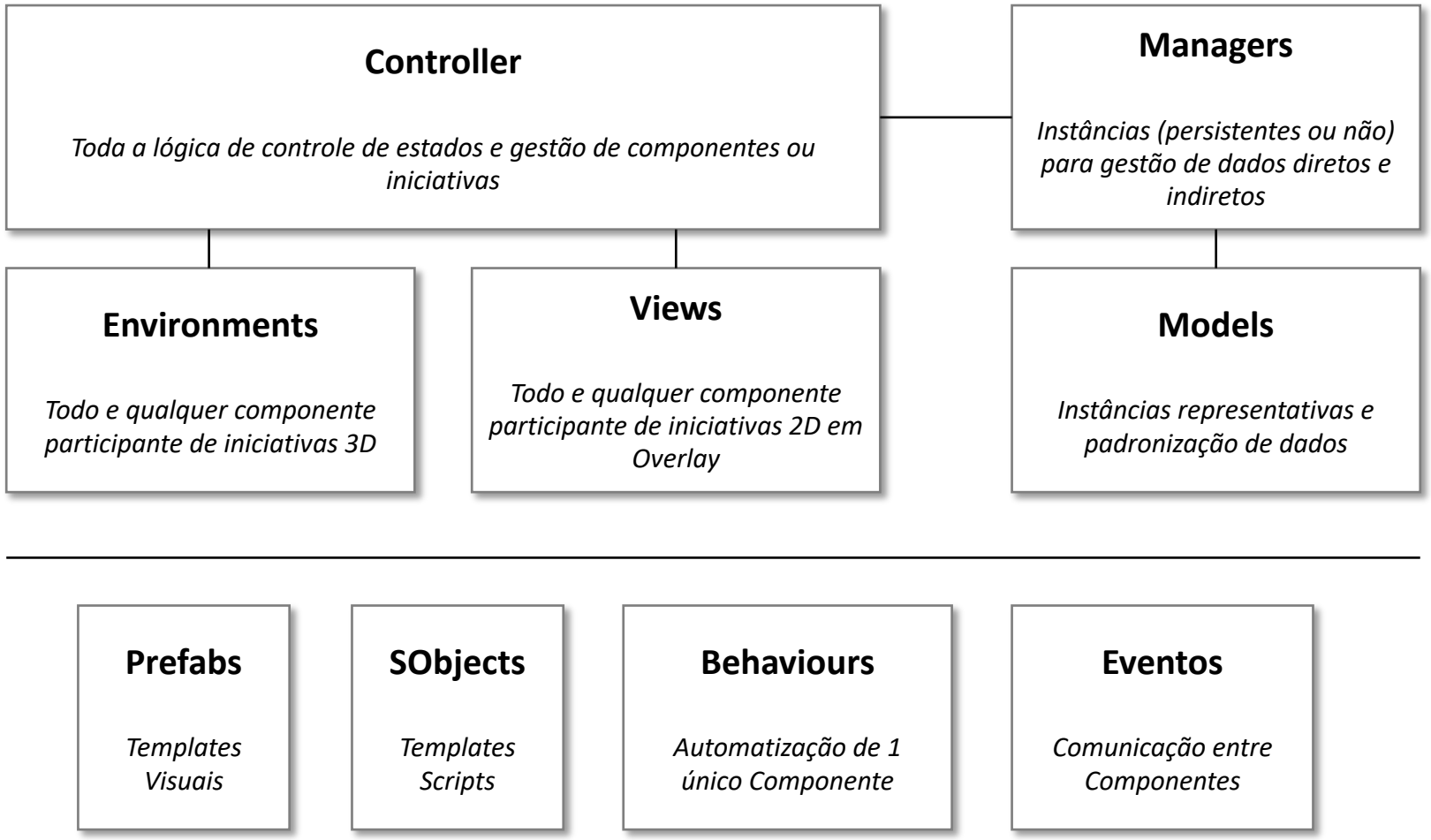
A estrutura de código idealizada precisa estar alinhada entre organização de scripts e hierarquia visual:



Estrutura de Código

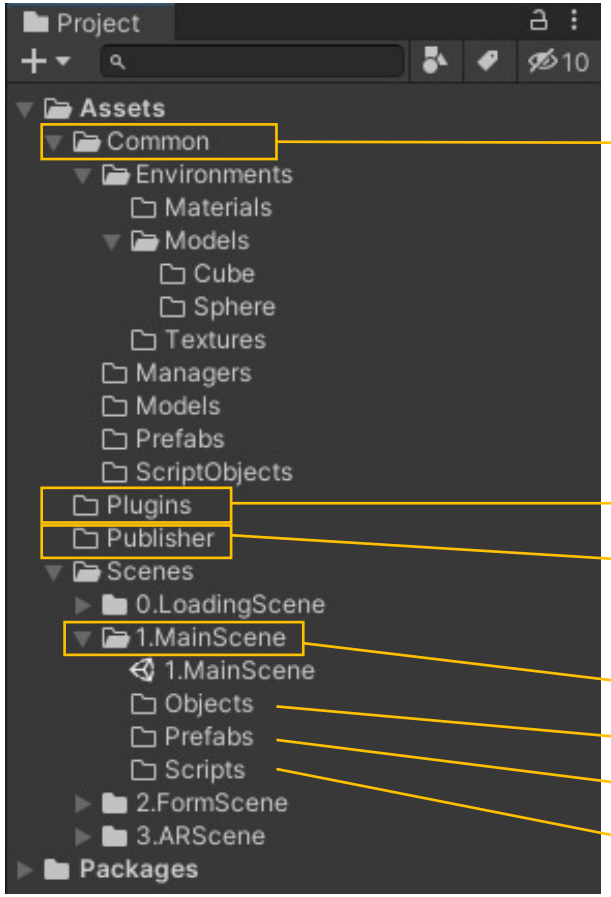
Nomenclatura

- **Iniciativas:** ambiente 3D, 2D ou UI Overlay
- **Componentes:** Objetos instanciados com responsabilidade definida, composto por mais de um elemento visual ou script
- **Elementos**: cada item existente e controlável seja visual ou script



Estrutura de Código

Organização de Scripts



Todos os itens compartilhados entre as várias cenas.

Bibliotecas/Pacotes de Terceiros não disponíveis via Package Manager

Assets relacionados a publicação do App

Todos os assets das cenas

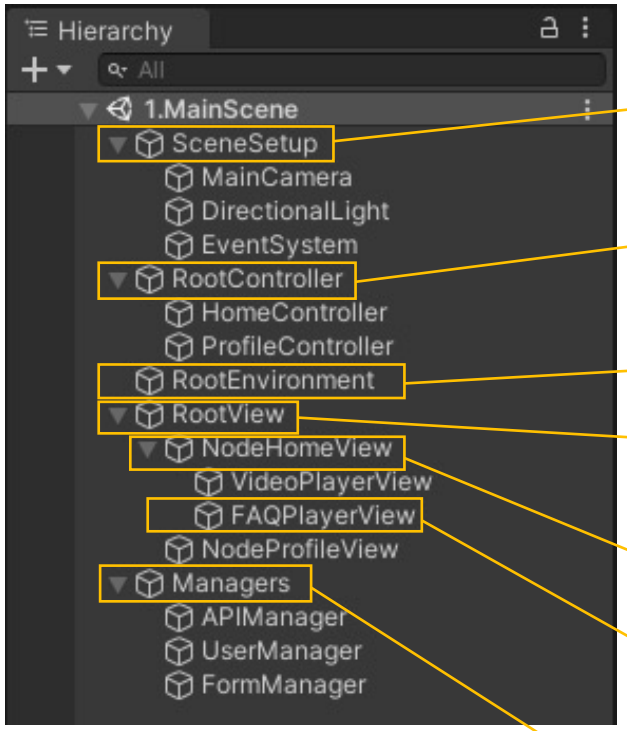
Objetos Template Scripts

Objetos Template Visuais

Scripts Monobehviours

Estrutura de Código

Organização de Scripts



Elementos Gerais da Cena

Agrupamento de Controllers.
Caso necessário pode exercer um controle master sobre os demais componentes (ex:sequência de execução)

Agrupamento de Environments.
Caso necessário pode exercer um controle master sobre os demais componentes

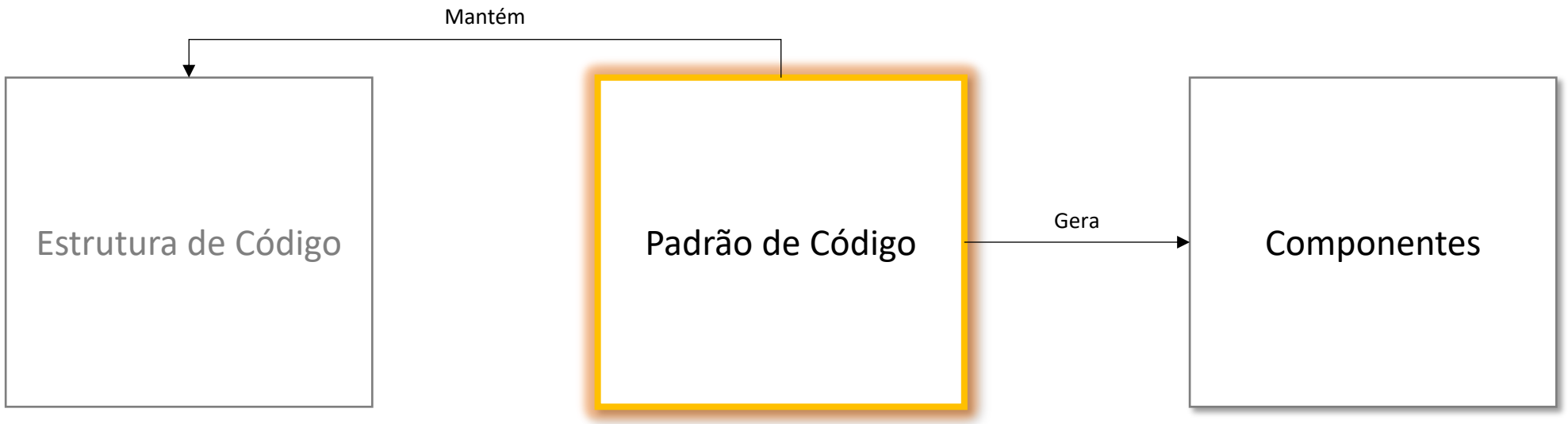
View Raíz (Root)

Todos os elementos filhos do Root recebem o prefixo ‘Node’

Todos os demais filhos de views não possuem nomenclatura definida

Agrupador de elementos Managers (persistentes ou não)

Componentes



Padrão de Código

1. Nomenclatura segue padrões C# e Unity já estabelecidas.
2. O conteúdo mínimo de um projeto é: SceneSetup, RootController, RootView, AppManager.
3. Um Script Behaviour precisa ter ao menos 1 método declarado e recomenda-se não deverá ultrapassar 120 linhas de código sem contabilizar enums ou descritivos e comentários.
4. Faça comentários que não explicam a técnica e sim a função e motivo de determinada estrutura. Seja amável, claro e objetivo. Se for óbvio, não escreva nada.
5. As funções sempre devem ter nomes significativos e claros sobre sua natureza
6. Todo evento compartilhado deverá iniciar com 'On'
7. **Cenas**
    1. As cenas são sempre numeradas e identificadas por enum no AppManager
    2. As cenas não são equivalentes a telas e devem ser definidas e limitadas a uma responsabilidade de comunicação e ação claras ao usuário
    3. Cenas derivadas podem ser criadas caso haja perda de performance em executar todas as responsabildiades em uma única cena
    4. Nunca deverá existir assets 'desligados' na cena. A inicialização deverá ocorrer sempre por código.
8. **Componentes**
    1. Cada componente deverá ter uma responsabilidade clara e auto-contida (quando possível)
    2. Todos os elementos pertencentes a um mesmo componente devem ser agrupados visualmente em uma mesma estrutura (quando possível)
    3. Evitar atribuir listeners, actions e eventos de forma visual
    4. Sempre agrupar elementos em hierarquia em forma visual
    5. Prefira *[SerializeField] private* ao invés de *public*
9. **Objetos Template Scripts ou Prefabs**
    1. Devem ser criados apenas quando surgirem situações de repetição de funções.
    2. Nunca prever ou premeditar uma situação de repetição, ela precisa ocorrer.

Componentes

Componentes

mondoDX™

- Scene Templates
- Rest API Manager
- Scene Switch
- Rest Resources Templates
- Model Manager
- View Switch
- Models Templates
- Logging Manager
- Storage Manager