1. Classe `ThreeCore`

A `ThreeCore` é a classe que gerencia o núcleo da cena 3D, como o ambiente, a câmera e o renderizador. Ela também garante que apenas uma instância de `ThreeCore` seja criada, usando o padrão **Singleton**. Isso significa que, mesmo se você tentar criar várias instâncias dessa classe, você só terá uma instância em toda a aplicação.

- **Construtor (`constructor`)**:
  - **Cena 3D**: A cena é onde todos os objetos 3D são colocados e renderizados.
  - **Câmera**: A câmera é responsável por exibir o que está na cena. Aqui, usamos uma câmera de perspectiva, que simula como vemos o mundo em 3D, com distâncias variáveis.
  - **Renderizador**: É o motor que desenha a cena 3D na tela. O `alpha: true` é utilizado para que o fundo da cena tenha transparência, o que pode ser útil para integrações com outras camadas (como a integração com um jogo 2D).
  - **Singleton**: `ThreeCore.instance = this` garante que apenas uma instância da classe seja criada. Se já houver uma instância, ele retorna a instância existente.
- **Métodos**:
  - `init(container)` : Inicializa a cena, adiciona o renderizador ao container da página HTML e começa o loop de animação.
  - `animate()` : Este método é chamado repetidamente através de `requestAnimationFrame` , o que permite a animação contínua. Ele renderiza a cena a cada quadro.

2. Classe `ThreeFactory`

A `ThreeFactory` é responsável por criar os objetos 3D fundamentais, como luzes, cubos e câmeras. Esse é um exemplo de **Abstract Factory**, já que ela centraliza a criação desses objetos.

- **Métodos**:
  - `createLight()` : Cria uma luz pontual (uma luz que emite de um ponto em todas as direções).
  - `createCube()` : Cria um cubo 3D com uma cor verde usando `BoxGeometry` (geometria do cubo) e `MeshStandardMaterial` (material com sombreamento adequado para renderização realista).
  - `createCamera()` : Cria uma câmera de perspectiva.

O padrão **Abstract Factory** ajuda a abstrair a criação de diferentes tipos de objetos 3D sem que o código do cliente precise saber como esses objetos são criados internamente.

3. Classe `EntityBuilder`

A `EntityBuilder` é uma implementação do padrão **Builder**. Ela permite construir entidades compostas de forma mais flexível, adicionando diferentes componentes a uma única entidade, como malhas (meshes) e posições.

- **Métodos**:
  - `addMesh(mesh)` : Adiciona um `mesh` (malha 3D, como um cubo) à entidade.
  - `setPosition(x, y, z)` : Define a posição da entidade no espaço 3D.

- `build()` : Finaliza a construção e retorna a entidade 3D.

O objetivo da classe `EntityBuilder` é proporcionar uma maneira fácil de criar objetos mais complexos, ao adicionar diferentes componentes (como malhas) e personalizar atributos de forma fluida.

4. Classe `PrototypeFactory`

A `PrototypeFactory` segue o padrão **Prototype**. Ela permite registrar e clonar objetos 3D, o que é útil quando você deseja criar cópias de objetos sem precisar recriá-los do zero, economizando recursos e tempo de processamento.

- **Métodos**:
  - `register(name, object3D)` : Registra um objeto 3D com um nome para que ele possa ser clonado mais tarde.
  - `clone(name)` : Clona um objeto registrado anteriormente, retornando uma cópia exata dele.

5. Classe `Game`

A classe `Game` utiliza a `ThreeFactory` para criar objetos de forma simplificada. Ao chamar o método `Game.create(type)` , ele cria um objeto do tipo especificado (como `cube` , `camera` , ou `light`).

- **Método** `create(type)` :
  - Cria os objetos 3D usando a `ThreeFactory` .
  - O `type` pode ser `cube` , `camera` ou `light` , e o método retorna o objeto correspondente.
  - Se um tipo desconhecido for passado, o código lança um erro.

Resumo de como funciona:

- `ThreeCore` gerencia o ciclo de vida da cena, a câmera e o renderizador.
- `ThreeFactory` cria objetos 3D, como luzes, cubos e câmeras.
- `EntityBuilder` ajuda a construir entidades 3D compostas (como adicionar malhas e definir posições).
- `PrototypeFactory` permite clonar objetos registrados.
- `Game` é a interface simplificada para criar objetos e interagir com a cena 3D de maneira fácil e rápida.