Système de Gestion des Ressources

Moteur de Jeu SFML

Vue d'ensemble du Resource Manager

Le ResourceManager est implémenté selon le pattern Singleton, ce qui signifie qu'il n'existe qu'une seule instance accessible globalement dans tout le moteur. Ce choix est judicieux car il centralise la gestion des ressources et évite les chargements redondants.

Caractéristiques principales

1. **Pattern Singleton** : Une seule instance accessible via `GetInstance()`
2. **Gestion de la mémoire** : Suivi de l'utilisation mémoire (`maxMemory` et `currentMemory`)
3. **Mapping de ressources** : Stockage des ressources dans une map avec leur chemin comme clé
4. **Support SFML** : Gestion des textures, sons et polices SFML
5. **Comptage de références** : Système qui permet de savoir quand une ressource n'est plus utilisée

Structure des classes

Le système s'articule autour des classes suivantes:

ResourceManager

```
- static ResourceManager* instance
- Map<string,IResource> resources
- ResourceFactory* factory
- size_t maxMemory
- size_t currentMemory
+ static GetInstance() ResourceManager*
+ GetTexture(string path) sf::Texture*
+ GetSound(string path) sf::SoundBuffer*
+ GetFont(string path) sf::Font*
+ UnloadResource(string path) void
+ UnloadUnusedResources() void
+ GetMemoryUsage() size_t
```

IResource (Interface abstraite)

```
- string path
- size_t size
- bool isLoaded
- int referenceCount
+ Load()* bool
+ Unload()* void
+ GetSize()* size_t
```

```
+ IsLoaded() bool
+ AddReference() void
+ RemoveReference() void
```

Classes d'implémentation

- **SFMLTexture** : Gère les textures
- **SFMLSoundBuffer** : Gère les sons
- **SFMLFont** : Gère les polices

Diagramme de classes

```
ResourceManager o-- IResource : manages
IResource <|-- SFMLTexture : extends
IResource <|-- SFMLSoundBuffer : extends
IResource <|-- SFMLFont : extends
```

Fonctionnement

1. Chargement à la demande

Les ressources sont chargées uniquement lorsqu'elles sont demandées via les méthodes comme `GetTexture()`, `GetSound()` ou `GetFont()`.

2. Processus de chargement

- Vérification si la ressource existe déjà en mémoire
- Si non, création d'une nouvelle instance du type de ressource approprié
- Chargement effectif via la méthode `Load()` de la ressource
- Incrémentation du compteur de références

3. Système de référencement

- Chaque fois qu'un objet utilise une ressource, son compteur est incrémenté (`AddReference()`)
- Quand l'objet n'utilise plus la ressource, le compteur est décrémenté (`RemoveReference()`)
- Une ressource avec un compteur à zéro peut être déchargée par `UnloadUnusedResources()`

4. Gestion de la mémoire

- Suivi de l'utilisation mémoire totale
- Possibilité de définir une limite maximale (`maxMemory`)
- Déchargement automatique possible quand la limite est atteinte

Intégration avec les autres systèmes

Le ResourceManager est utilisé principalement par:

1. **SFMLRenderer** pour charger les textures et polices

2. **SFMLAudioManager** pour charger les sons et musiques
3. **RenderComponent** indirectement pour accéder aux textures
4. **AudioSource** indirectement pour accéder aux sons

Points forts de cette implémentation

1. **Efficacité mémoire** : Les ressources ne sont chargées qu'une seule fois, même si utilisées à plusieurs endroits
2. **Gestion des références** : Permet de libérer automatiquement les ressources non utilisées
3. **Centralisation** : Un point d'accès unique pour toutes les ressources
4. **Abstraction** : L'interface IResource permet d'étendre facilement le système à d'autres types de ressources

Considérations d'implémentation

Pour implémenter ce système efficacement :

- Veiller à bien gérer les compteurs de références
- S'assurer que chaque composant qui utilise une ressource appelle correctement `AddReference()` et `RemoveReference()`
- Intégrer correctement le ResourceManager avec la classe Window et les systèmes de rendu et audio
- Implémenter une stratégie de déchargement intelligente pour les ressources non utilisées

Exemple de flux d'utilisation typique

1. Un GameObject avec un RenderComponent demande une texture
2. Le RenderComponent contacte le ResourceManager
3. Le ResourceManager vérifie si la texture est déjà chargée :
    - Si oui, il incrémente le compteur de références et retourne la texture
    - Si non, il crée une nouvelle SFMLTexture, la charge et l'ajoute à la map
4. Quand le GameObject est détruit, le RenderComponent informe le ResourceManager
5. Le ResourceManager décrémente le compteur de références
6. Lors d'un appel à `UnloadUnusedResources()`, toutes les ressources avec un compteur à 0 sont déchargées

Ce système optimise l'utilisation de la mémoire tout en simplifiant la gestion des ressources par les développeurs du jeu.