REVUE DE CONCEPTION DU PBRADIUM

Matthias Roves, Hugo Rens et Jehan-Antoine Vayssade

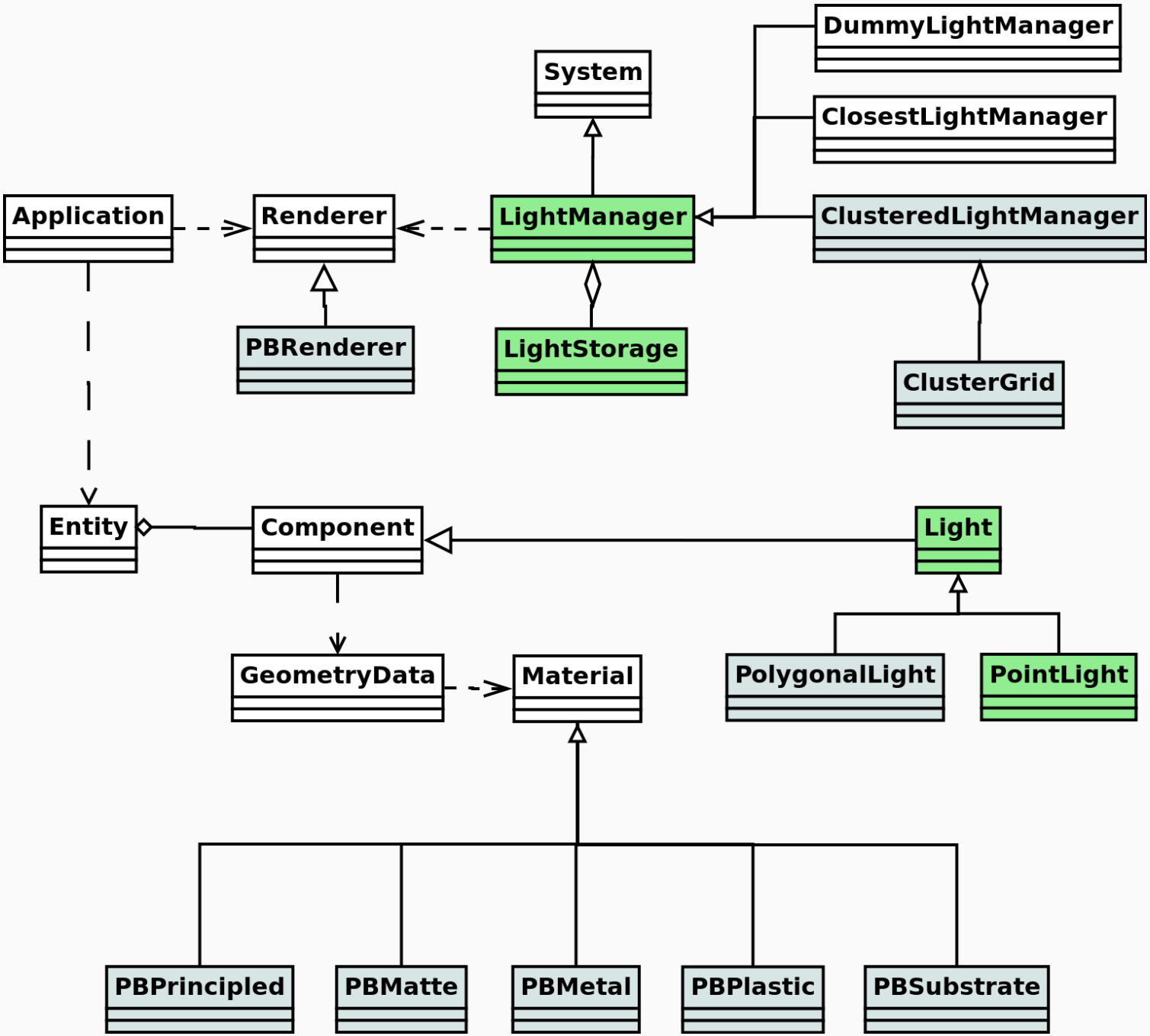
23 février 2018

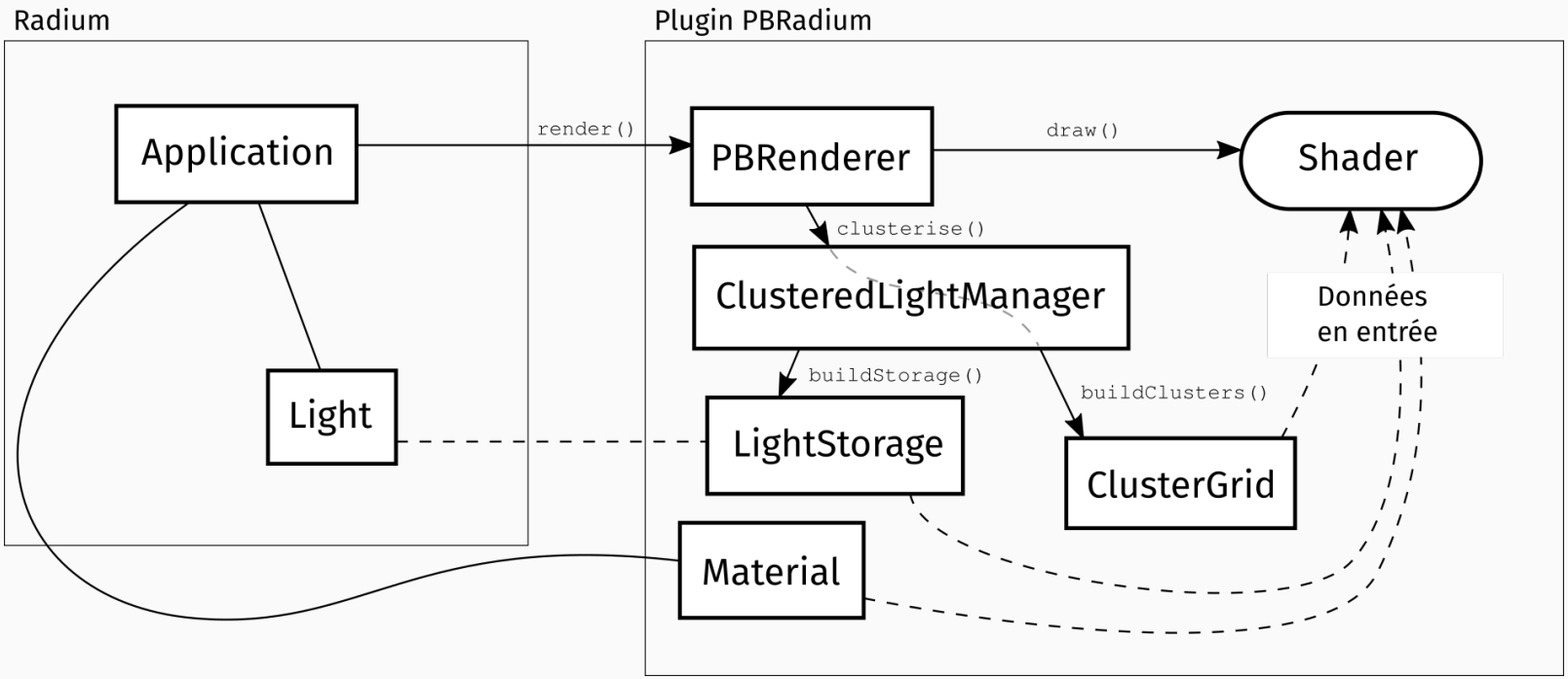
Université Paul Sabatier (Toulouse III)

PBRADIUM

- Implémentation des BSDF Disney
- Implémentation des sources indirectes nombreuses
- Implémentation des sources polygonales

VUE D'ENSEMBLE


DummyLightManager
System
ClosestLightManager
Application
Renderer
LightManager
ClusteredLightManager
PBRenderer
LightStorage
ClusterGrid
Entity
Component
Light
GeometryData
Material
PolygonalLight
PointLight
PBPrincipled
PBMatte
PBMetal
PBPlastic
PBSubstrate


Radium
Plugin PBRadium
Application
render()
PBRenderer
draw()
Shader
clusterise()
ClusteredLightManager
Données
en entrée
buildStorage()
buildClusters()
Light
LightStorage
ClusterGrid
Material

1. Chargement
2. Création des LightManagers *(et LightStorages associés)*
3. Création de la ClusteredGrid
4. À chaque frame
   - 4.1 Z-prepass•
   - 4.2 Clustering•
   - 4.3 Préparation du LightManager direct
   - 4.4 Calcul de l'éclairage direct•
   - 4.5 Préparation du LightManager indirect
   - 4.6 Calcul de l'éclairage indirect•

Les étapes marquées d'un • se font sur **GPU**.

CLASSES ET STRUCTURES DE DONNÉES

Niveau Radium

LIGHT

- Interface de base des lumières
- Rattachée à une entité en tant que Component
  - Transformations
  - Affichage
- Base des PointLights et PolygonalLights

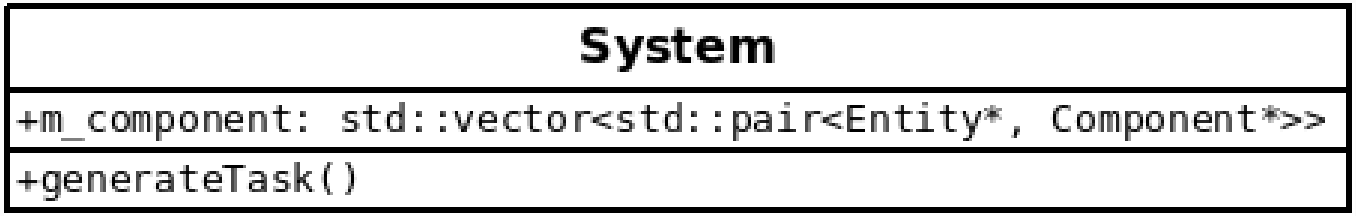


FIGURE 1 – Interface System du Radium

- Implémente l'interface System
- Gère un ensemble de lumières
- Utilise un LightStorage pour stocker
- Définit le transfert des sources au GPU

- Interface de stockage de sources
- Peut être plus ou complexe
  - Simple liste
  - Texture 3D

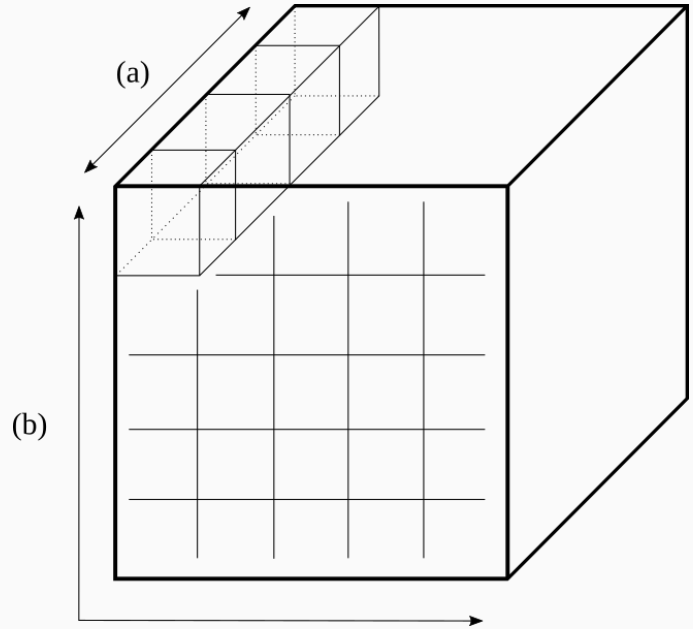


FIGURE 2 – Stockage des sources ponctuelles.

Niveau Plugin

PBRENDERER

- Implémente l'interface Renderer.
- Fait les différentes étapes de rendu.
- Conserve les différents framebuffers de rendu.
- Utilise un ensemble de LightManager.

ClusteredLightManager

- Implémente l'interface LightManager.
- Utilise la structure ClusterStorage.
- Filtre les lumières par rapport au cluster.

- Stocke le résultat de l'opération de clustering.
- Transmet au GPU.
- Contient la liste d'indices de lumières par clusters.
- Pas une interface.

PolygonalLight

- Implémente l'interface Light.
- Utilise les propriété de l'entité (Mesh, couleur, transformation).
- Possède sa propre intensité.
- Se base sur un ensemble de textures pré-calculées.

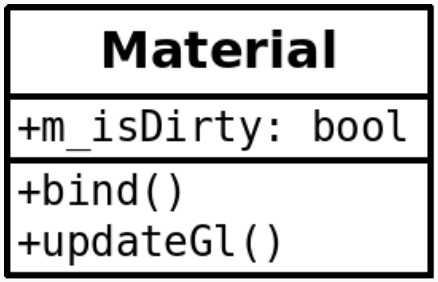


FIGURE 3 – Classe d'interface Material.

- Déjà existante dans le moteur
- Oblige la définition d'interaction avec le GPU

- Implémente l'interface Material
- S'enregistre auprès du Radium
- Gère ses textures et données de rendu

BSDF Disney	BSDF Plastique	BSDF Métallique	BSDF Matte	BSDF Substrate
Couleur diffuse	Couleur diffuse	Couleur diffuse	Couleur diffuse	Couleur diffuse
Couleur spéculaire	Couleur spéculaire	Coefficient d'absorption		Couleur Spéculaire
Rugosité	Rugosité	Rugosité	Rugosité	
Anisotropie		Rugosité selon $\vec{u}$		Rugosité selon $\vec{u}$
Indice de réflexion		Rugosité selon $\vec{v}$		Rugosité selon $\vec{v}$
Paramètre métallique				
Paramètre d'effet soie				
Couleur de l'effet soie				
Paramètre de Clearcoat				
Brillance du Clearcoat				

FIGURE 4 – Ensemble des paramètre de chaque matériaux

Shaders

- Alternative à ShaderProgramManager et ShaderConfiguration
- **Composer** des shaderprograms
- Représenter toutes les configurations possibles
- Éviter trop de branchements dans le GPU
- Avoir un code modulaire
- Contraint à une interface très codifiée

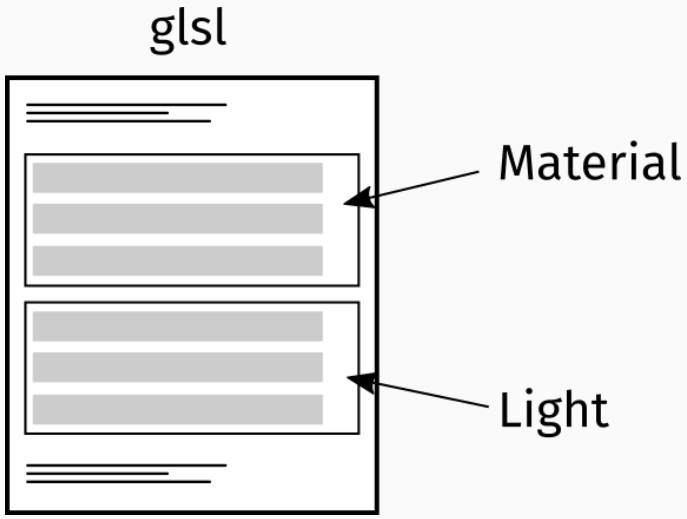


FIGURE 5 – Fragment shader

TESTS

Entrée

- Carte de profondeur
- Configuration de caméra

Sortie

- Liste des clusters

Entrée

- Jeu de lumières
- Configuration de caméra
- Cellule à accéder

Sortie

- Sous-ensemble de lumières

- Extraction des valeurs de l'éclairage
  - côté Radium (glsl $\Rightarrow$ texture)
  - côté PBRT (image)
- Comparaison de données calculées
- Intégré à la suite de tests
- Seule la partie Radium dans le temps du projet.

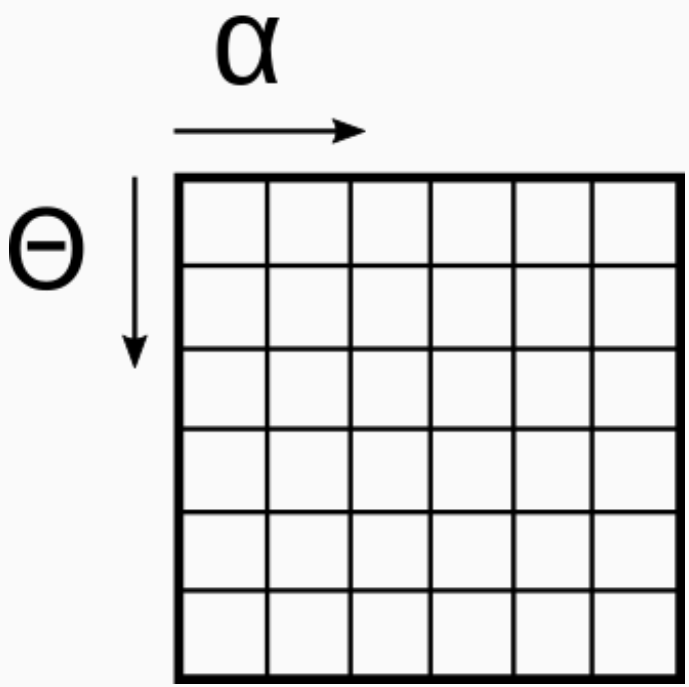


FIGURE 6 – Dimensions de la texture en sortie. α est la rugosité, Θ l'angle d'incidence.

- Extraction des résultats d'évaluation de la BRDF
  - côté Radium (glsl $\Rightarrow$ texture)
  - côté PBRT (image)
- Comparaison de données calculées
- Intégré à la suite de tests
- Seule la partie Radium dans le temps du projet.

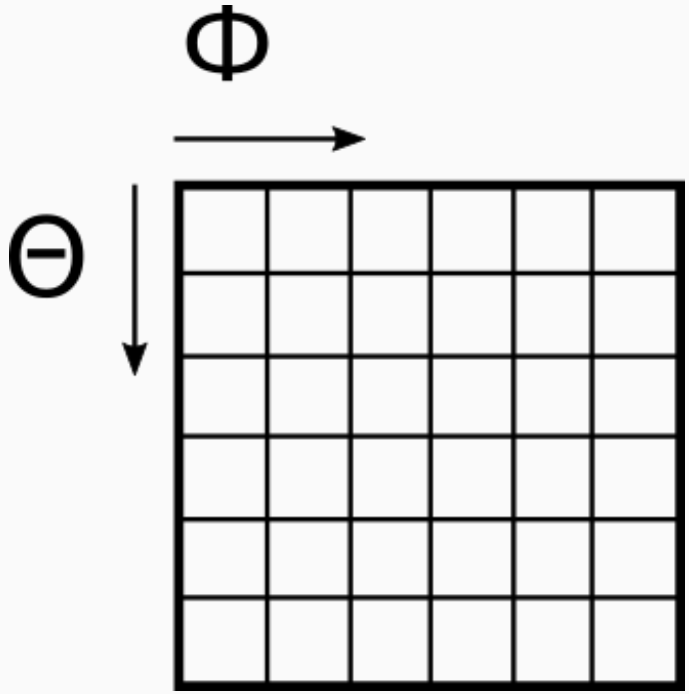


FIGURE 7 – Dimensions de la texture en sortie. Φ est l'angle de vue, Θ l'angle de la lumière incidente.

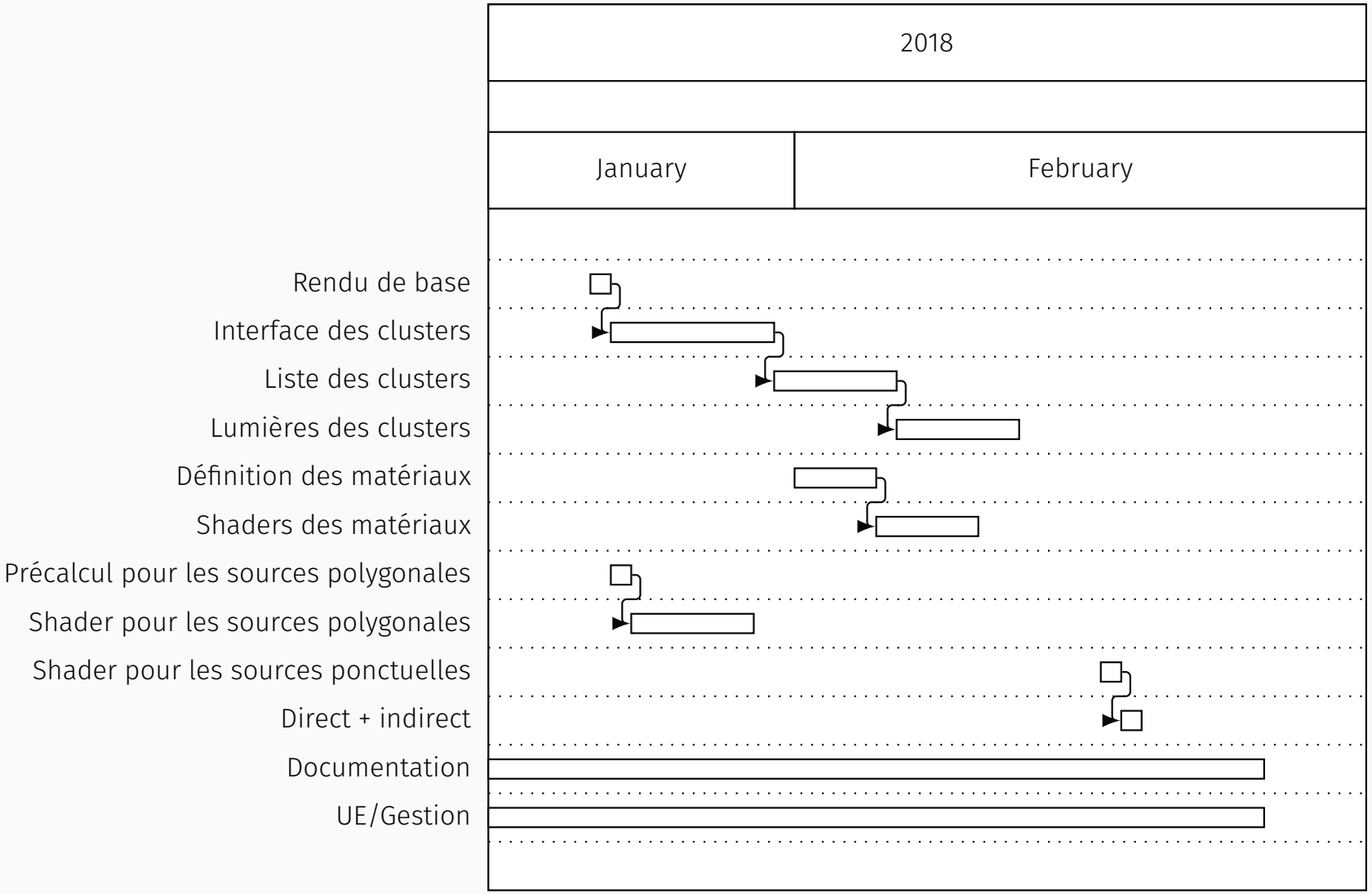
PLANNING

Changement

Les durées de certaines tâches ont été remaniées.

- La tâche ”Interface des clusters” est passée de 10 à 8 jours par homme.
- La tâche de ”précalcul pour les sources polygonales” s’est vue amplement simplifiée.

PLANNING


2018
January
February
Rendu de base
Interface des clusters
Liste des clusters
Lumières des clusters
Définition des matériaux
Shaders des matériaux
Précalcul pour les sources polygonales
Shader pour les sources polygonales
Shader pour les sources ponctuelles
Direct + indirect
Documentation
UE/Gestion

RISQUES

Dépassements des délais

- Réestimation des tâches
- Réorganisation

Autres risques

Peu d'évolution depuis la revue de spécifications.

- Non-conformité
- Modifications du Radium Engine

Questions?