Chapitre 6
Utilisation des assets graphiques et audio

1. La caméra

1.1 Présentation

La caméra sert à capturer et à afficher le monde au joueur. Cet élément est un objet essentiel d'une scène et doit être défini avec précaution car c'est lui qui est en première ligne sur l'aspect général de votre jeu. Le nombre de caméras n'est pas limité mais les performances peuvent baisser si ce nombre devient important. Elles n'ont pas d'ordre de rendu défini et peuvent afficher l'intégralité de la scène ou seulement une partie.

1.2 Paramétrage de l'affichage

Dans l'Inspector on retrouve de nombreuses propriétés permettant de modeler la caméra selon ses besoins.

1.2.1 Affichage des objets

– **Clear Flags** : cette propriété permet de paramétrer l'affichage des parties de l'écran sans affichage. Cela est utile lors de l'utilisation de plusieurs caméras en même temps afin d'afficher un type d'objet sur la première caméra et un autre type d'objet sur la seconde caméra.

Le principe est simple, chaque caméra enregistre les informations de couleur et de profondeur quand il rend la vue. Les parties de l'écran n'étant pas renseignées, sans information de couleur donc, afficheront par défaut une **Skybox**. Lors de l'utilisation de plusieurs caméras, chacune d'entre elles enregistre ses propres informations en mémoire pour ensuite les combiner lors de la génération de la vue.

  - **Skybox** : c'est la valeur par défaut. Chaque portion vide de l'écran affichera la Skybox attachée à celle-ci.
  - **Solid Color** : chaque portion vide de l'écran affichera la couleur définie dans la propriété **Background**.
  - **Depth Only** : cette valeur permet de rendre un objet sans prendre en compte sa position dans l'espace et donc sa profondeur. Elle est particulièrement utilisée dans les FPS (*First Person Shooter*) afin d'afficher une arme par exemple.
  - **Don't Clear** : cette valeur un peu spéciale est rarement utilisée sans être associée à un shader. Elle fait en sorte de ne pas effacer les informations du buffer de rendu de la vue ce qui fait qu'à chaque cycle, le rendu de la vue se fait par-dessus l'ancien.
- **Background** : définit la couleur à utiliser pour le rendu des zones de l'écran vide une fois tous les éléments de la scène affichés et sans qu'aucune Skybox ne soit paramétrée.
- **Culling Mask** : définit les layers à prendre en compte ou non, lors du rendu de la caméra. Les objets de la scène peuvent être affectés à un layer via l'Inspector.

1.2.2 Vision et projection

- **Projection** : permet de changer la perspective des objets de la scène.
  - **Perspective** : la caméra affichera les objets de la scène en gardant la perspective.
  - **Orthographic** : la caméra affichera les objets de la scène uniformément sans prendre en compte la perspective.
- **Size** : quand la propriété **Orthographic** est sélectionnée, définit la taille de la fenêtre de la caméra.

- **Field of view** : quand la propriété **Perspective** est sélectionnée, définit la taille de l'angle de vue de la caméra, en dégrés selon l'axe Y.
- **Clipping Planes** : définit la distance de début et de fin de rendu de la caméra. Les objets en dehors de cette zone seront ignorés.
  - **Near** : le point le plus proche par rapport à la caméra où l'affichage se produira.
  - **Far** : le point le plus loin par rapport à la caméra où l'affichage se produira.
  - **Normalized View Port Rect** : définit la position et la taille où sera affiché le rendu de la caméra à sur l'écran. Les valeurs sont définies en *Screen Coordinates* (coordonnées d'écran) avec des valeurs allant de 0 à 1.
  - **X** : définit la position horizontale où la caméra sera rendue à l'écran.
  - **Y** : définit la position verticale où la caméra sera rendue à l'écran.
  - **W** : définit la largeur de la caméra à l'écran.
  - **H** : définit la hauteur de la caméra à l'écran.
- **Depth** : définit la position de la caméra dans l'ordre d'affichage. Les caméras ayant une valeur élevée seront affichées au-dessus des caméras ayant une valeur basse.
- **Rendering Path** : liste des options afin de définir la méthode de rendu qui sera utilisée par la caméra.
  - **Use Player Settings** : la caméra utilisera les **Rendering Path** définis dans les **Player Settings**.
  - **Vertex Lit** : tous les objets de la caméra seront rendus comme des objets Vertex-Lit.
  - **Forward** : tous les objets seront rendus avec seulement une passe par matériaux.
  - **Deferred Lighting** : tous les objets seront rendus une seule fois sans lumière puis le rendu de l'ensemble des objets avec la lumière sera fait à la fin du cycle de rendu.
- **Target Texture** : définit une Render Texture (texture de rendu) qui contient le rendu de la vue de la caméra. En renseignant cette texture, le calcul du rendu de la caméra sera désactivé.
- **HDR** : active le rendu *High Dynamic Range* pour la caméra.

1.3 La vue Frustum

1.3.1 Définition

Le *Frustum* (tronc) désigne un solide, souvent de type pyramidal ou cylindrique, délimité par deux plans parallèles. Dans Unity, cela se prête parfaitement pour définir le champ de vision de la caméra prenant en compte la perspective des objets.

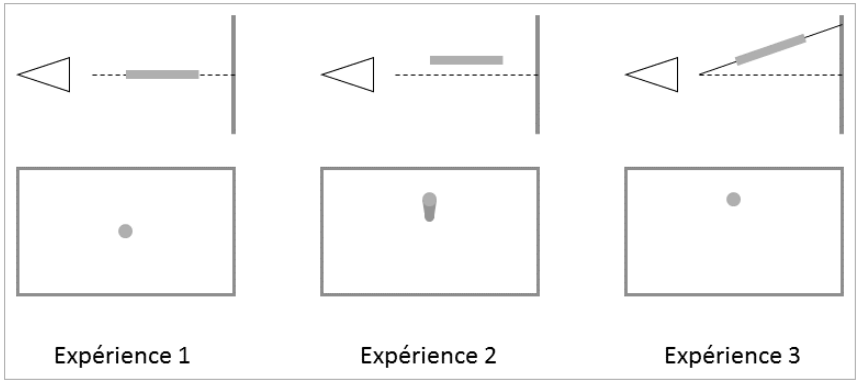
1.3.2 Expérimentations

Pour bien comprendre le principe, nous allons faire trois expériences :

1 Placez-vous devant un écran et positionnez les yeux au centre de celui-ci. Positionnez maintenant un stylo perpendiculairement à l'écran et au centre de celui-ci. Qu'est-ce que l'on remarque ? On ne voit qu'une partie du stylo, son dos, qui représente un cercle.

2 Toujours devant l'écran, positionnez les yeux au centre de celui-ci. Placez le stylo perpendiculairement à l'écran et au centre de celui-ci et déplacez-le de quelques centimètres vers le haut. Qu'est-ce que l'on remarque ? On voit toujours le dos du stylo mais aussi son corps. Le tout forme une sorte de trapèze.

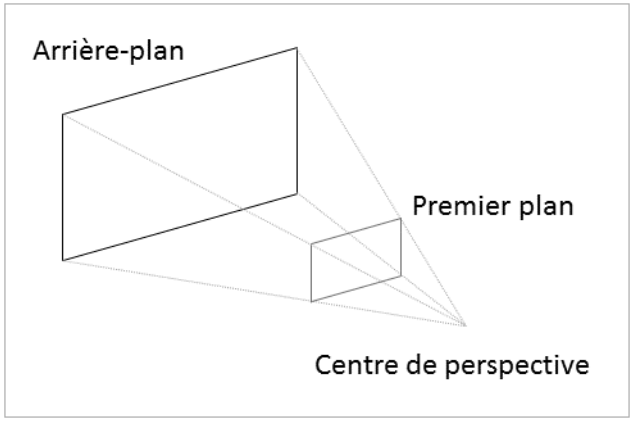
3 Inclinez maintenant le stylo de façon à ce que l'extrémité la plus proche de l'écran remonte vers le haut de l'écran jusqu'à ne plus voir qu'un cercle (comme dans l'expérience 1).



Le but de cette expérience est de montrer que pour tout point de l'image de rendu de la caméra correspond une ligne dans l'espace et qu'un seul point de cette ligne est visible dans cette image. Tout ce qui est derrière cette position sur cette ligne n'est pas affiché.

1.3.3 Limites d'affichage

Les bords extérieurs de l'image sont définis par les lignes divergentes qui correspondent aux coins de l'image. Si les lignes passant derrière la caméra étaient tracées, elles convergeraient toutes vers un seul point. Dans Unity, on appelle ce point le centre de perceptive et il est situé exactement aux coordonnées du `Transform` de la caméra. L'angle créé par les lignes partant des centres supérieurs et inférieurs de l'écran jusqu'au centre de perspective est appelé *Field Of View* (champ de vision), souvent abrégé FOV.



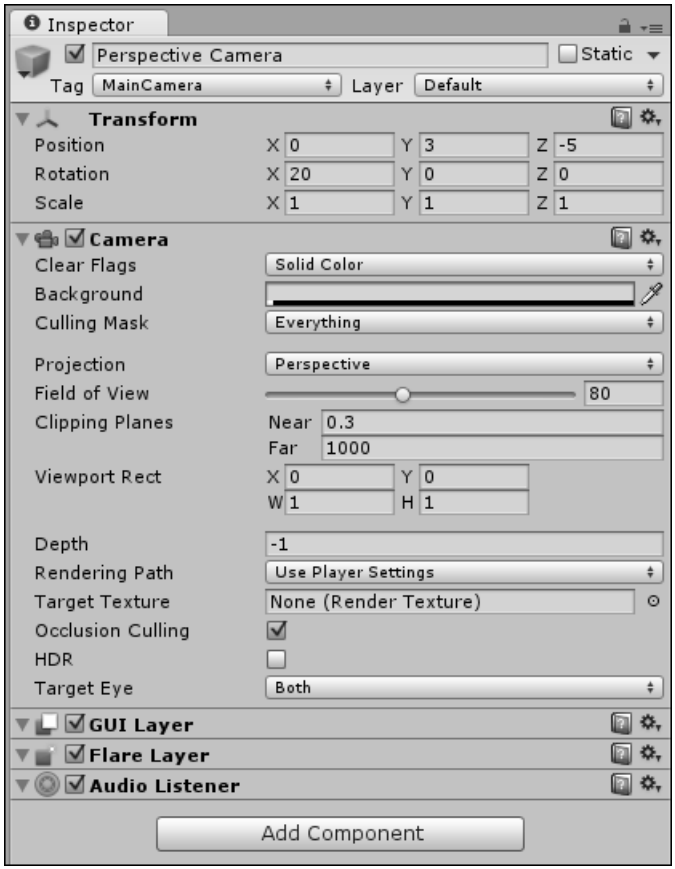
1.3.4 Différence entre la vue perspective et la vue orthographique

Afin de comprendre la différence entre les deux modes de projection, nous allons créer deux scènes avec les différentes vues possibles.

Création d'une scène ayant une caméra prenant en compte la perspective

- Dans le menu **File**, créez une nouvelle scène en cliquant sur le sous-menu **New Scene**.

- Dans la fenêtre **Hierarchy**, sélectionnez l'objet **Main Camera** et supprimez-le. Cliquez ensuite sur **Create - Create Empty**.
- Dans la fenêtre **Inspector**, cliquez sur **Add Component - Rendering - Camera**. Dans le composant **Camera** nouvellement créé, renommez l'objet **Perspective Camera**. Modifiez les propriétés **Projection** en vue **Perspective**, **Clear Flags** en **Solid Color**, **Field Of View** à **80** et **Background** avec votre couleur préférée. Modifiez la **Position** de la caméra en **(0, 3, -5)** et sa **Rotation** en **(20, 0, 0)**.



- Dans la fenêtre **Hierarchy**, sélectionnez **Create - 3D Object - Cube**.