LogiCube

UNITY

Objectifs

Créer un jeu “voxel”

Objectifs

Créer un jeu “voxel”

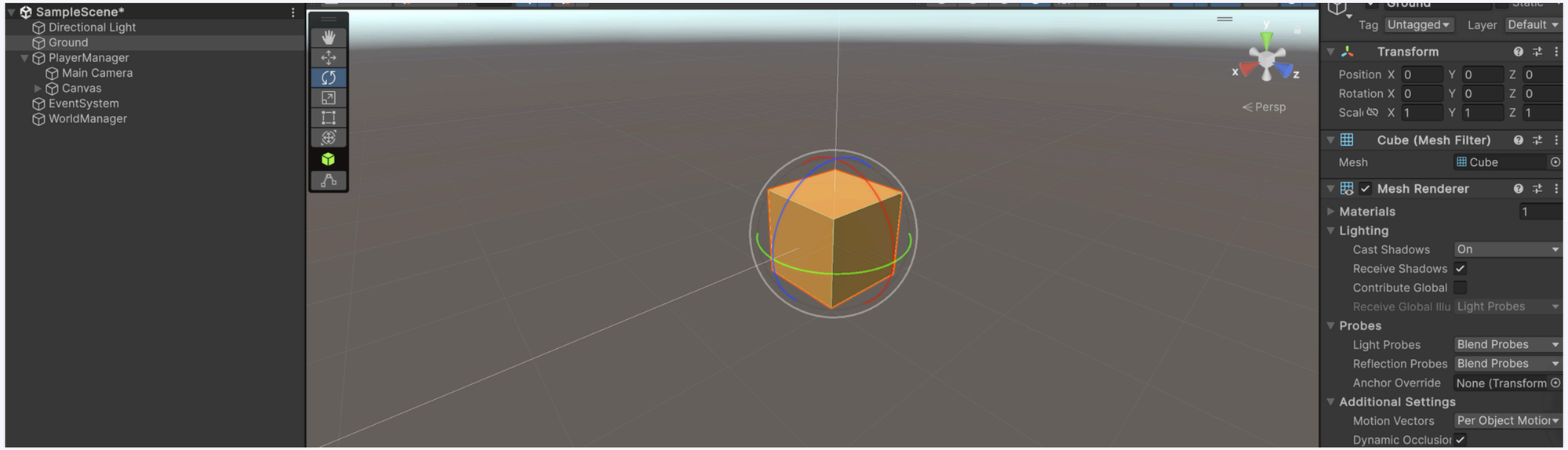
Objectifs

Créer un jeu “voxel”

1. **Se déplacer**
2. **Créer un crosshair**
3. **Placer un bloc**
4. **Détruire un bloc**
5. **(Bonus) Sauvegarder le monde**

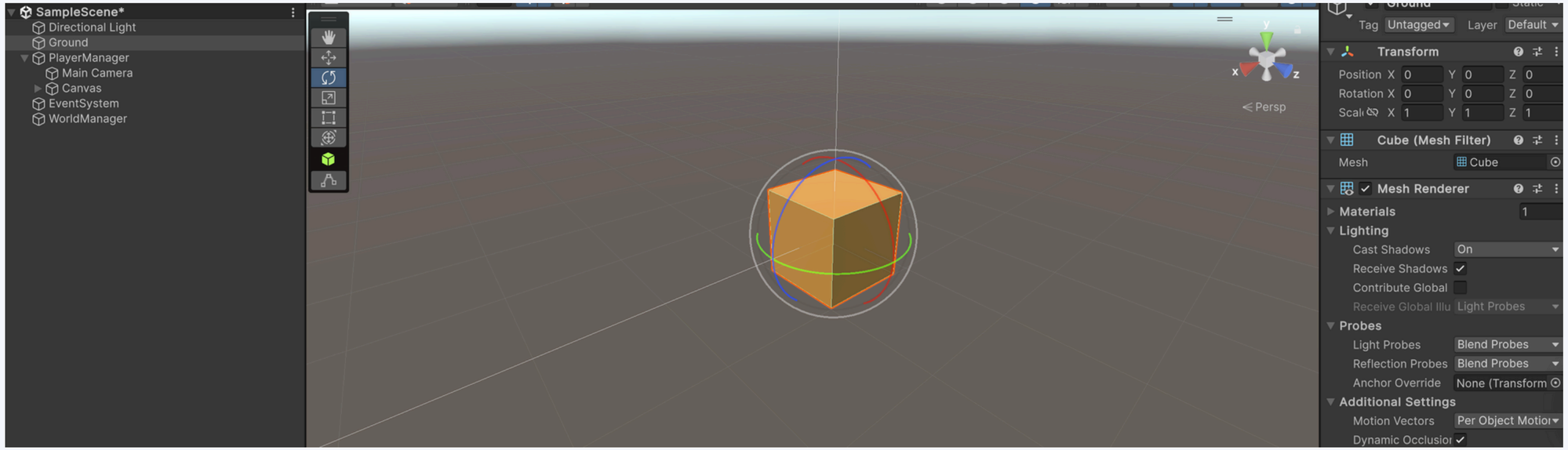
Base du projet

Créez un Cube en position 0,0,0 de taille 1



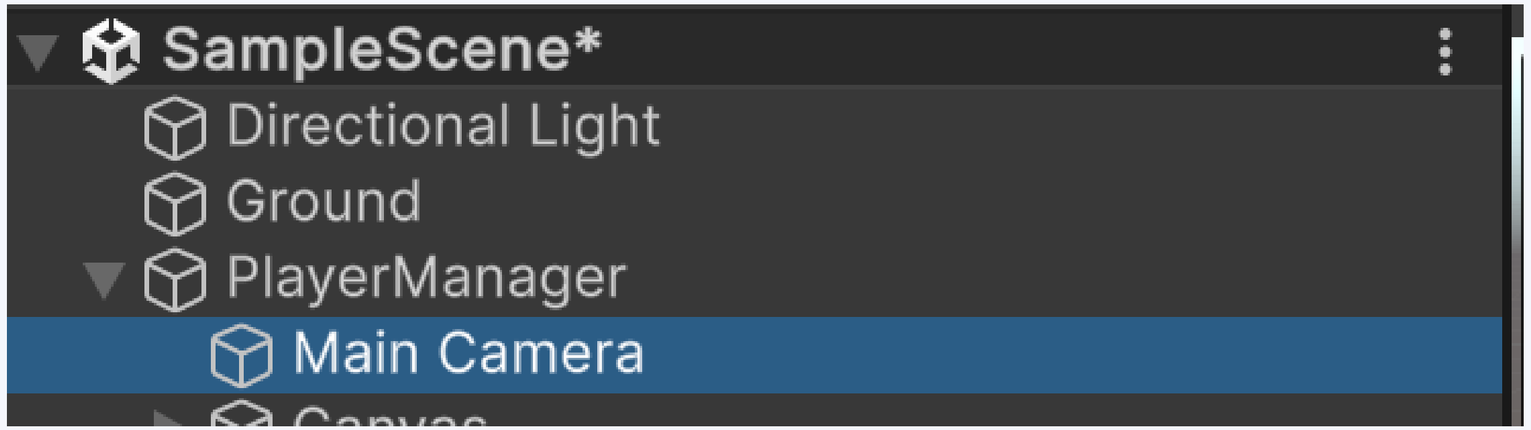
Base du projet

Il vas nous servir de base pour construire



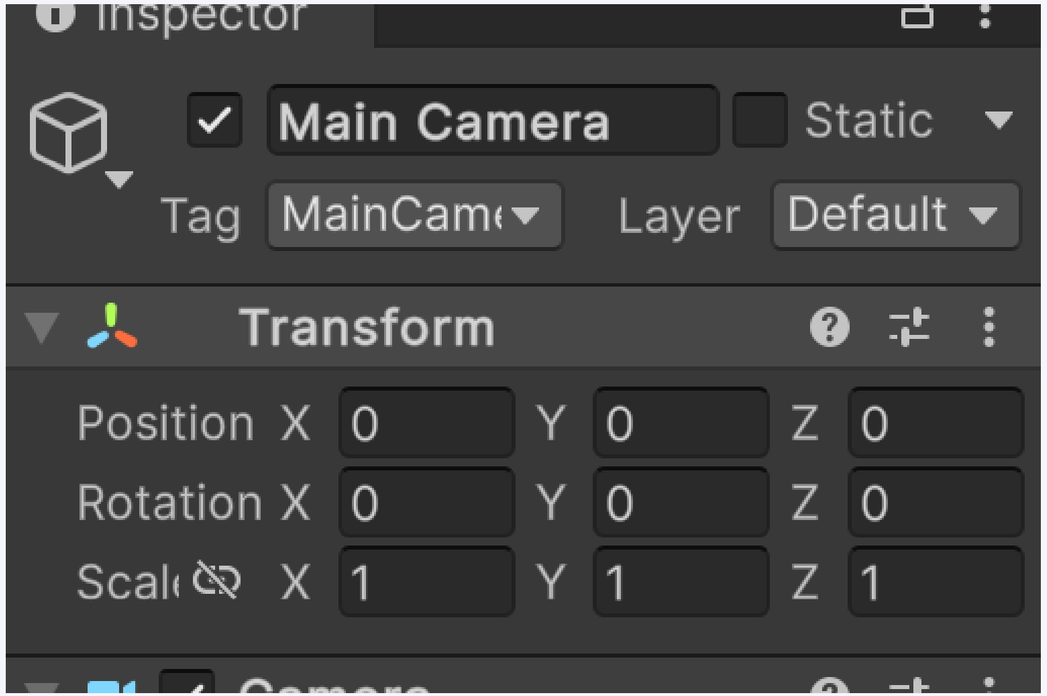
Base du projet

Créez un EmptyObject “PlayerManager” et glissez la caméra dans l’objet



Base du projet

Vérifiez que la caméra est en 0 0 0 aussi



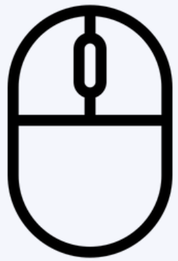
Se déplacer

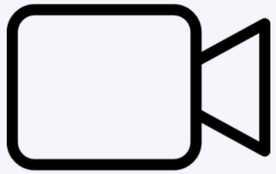
Bouger sur l’axe x et z

Monter

Descendre

Se déplacer





La camera suis la souris

Se déplacer

Ne codez pas encore, lisez juste les prochaines slides

Se déplacer

Ces deux variables float contrôle la vitesse de déplacement et celle de la souris

```
public float moveSpeed = 10f;    // Movement speed
public float lookSpeed = 2f;     // Mouse look speed
```

Se déplacer

CursorLockMode permet de bloquer le curseur quand nous sommes en jeu

```
void Start()
{
    // Lock and hide the cursor
    Cursor.lockState = CursorLockMode.Locked;
    Cursor.visible = false;
}
```

Se déplacer

Ici on récupère les données de la souris et on tourne le personnage de cette manière

```
float mouseX = Input.GetAxis("Mouse X") * lookSpeed;
float mouseY = Input.GetAxis("Mouse Y") * lookSpeed;
transform.Rotate(-mouseY, mouseX, 0);
```

Se déplacer

On créer deux vecteur pour droite-gauche et avant-arrière

```
// Horizontal movement (relative to world y-axis)
Vector3 forward = new Vector3(transform.forward.x, 0, transform.forward.z).normalized;
Vector3 right = new Vector3(transform.right.x, 0, transform.right.z).normalized;

Vector3 move = Vector3.zero;
if (Input.GetKey(KeyCode.W)) move += forward;
if (Input.GetKey(KeyCode.S)) move -= forward;
if (Input.GetKey(KeyCode.A)) move -= right;
if (Input.GetKey(KeyCode.D)) move += right;
if (Input.GetKey(KeyCode.Space)) move += Vector3.up;        // Move up
if (Input.GetKey(KeyCode.LeftShift)) move -= Vector3.up;    // Move down

transform.position += move * moveSpeed * Time.deltaTime;
```

Se déplacer

Puis en fonction des touches on modifie ses vecteurs

```
// Horizontal movement (relative to world y-axis)
Vector3 forward = new Vector3(transform.forward.x, 0, transform.forward.z).normalized;
Vector3 right = new Vector3(transform.right.x, 0, transform.right.z).normalized;

Vector3 move = Vector3.zero;
if (Input.GetKey(KeyCode.W)) move += forward;
if (Input.GetKey(KeyCode.S)) move -= forward;
if (Input.GetKey(KeyCode.A)) move -= right;
if (Input.GetKey(KeyCode.D)) move += right;
if (Input.GetKey(KeyCode.Space)) move += Vector3.up;        // Move up
if (Input.GetKey(KeyCode.LeftShift)) move -= Vector3.up;    // Move down

transform.position += move * moveSpeed * Time.deltaTime;
```

Se déplacer

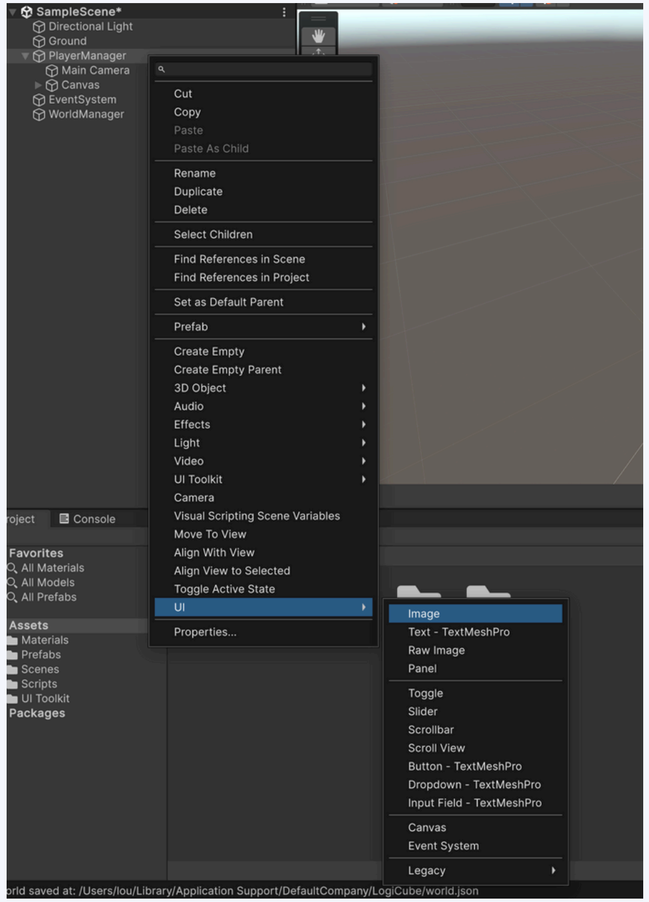
Tout le code est sur GitHub. Créez un nouveau Script FlyCamera et copiez collez le code

Glissez le Script sur le PlayerManager et essayez de vous déplacer !

Se déplacer

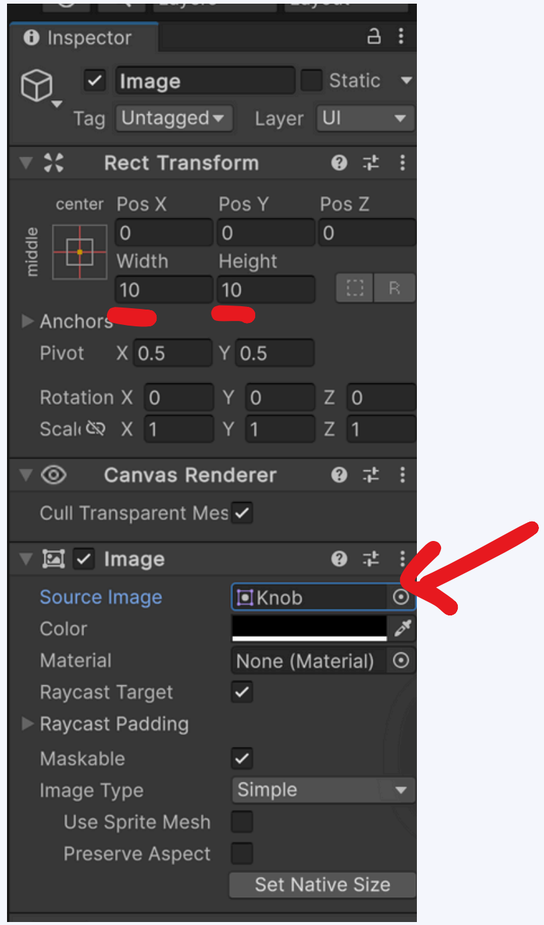
Le code sur GitHub à quelque lignes en plus pour éviter que le personage ne tourne sur lui même et permet de débloquer la souris en appuyant sur Escape

Créer un Crosshair



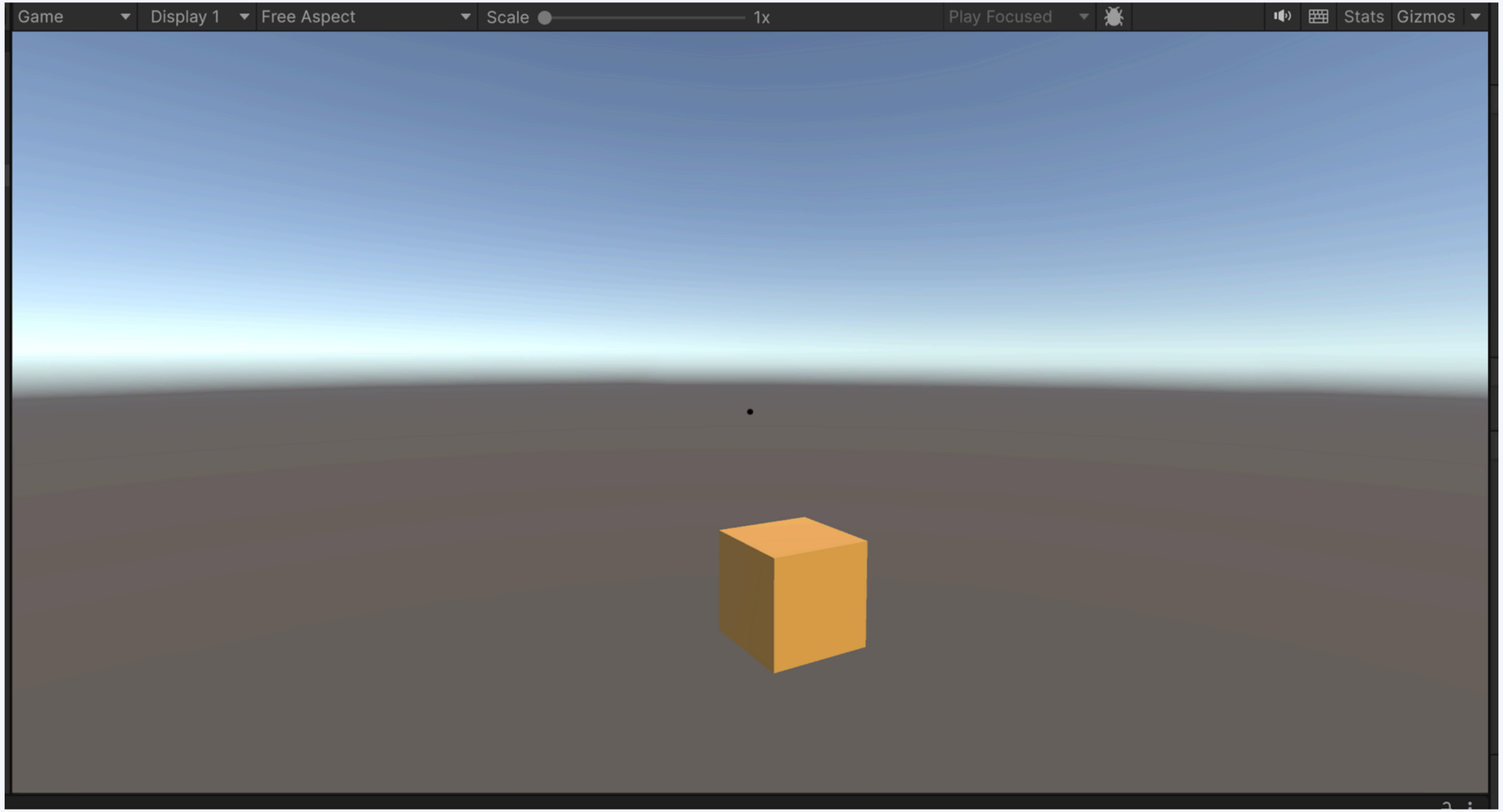
Dans le PlayerManager, créez un objet Ui → Image

Créer un Crosshair



Choisissez Knob dans les images (un cercle) avec une taille de 10 en Width et Height. Vous pouvez choisir la couleur

La base du projet est finie !

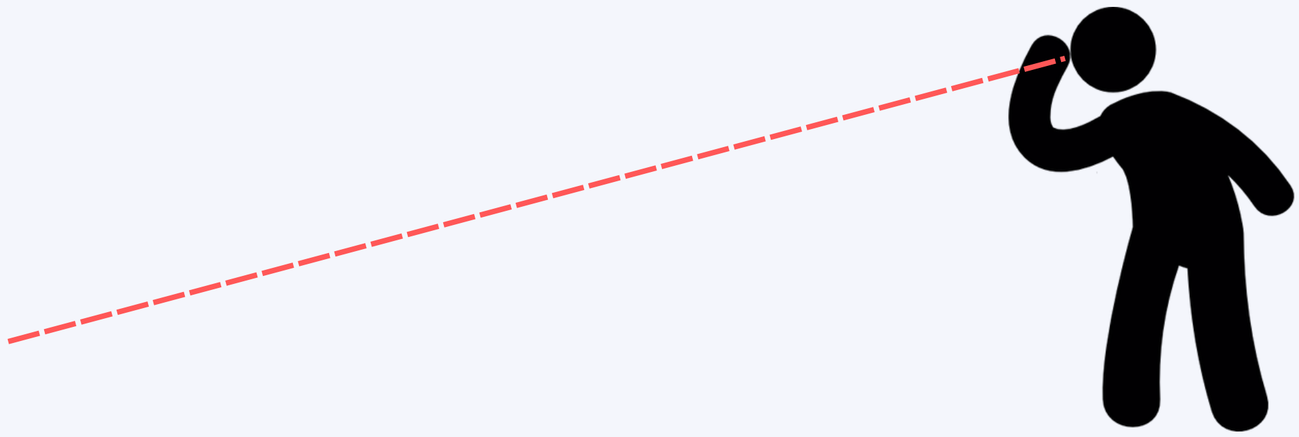


Placer des cubes !

Mais comment ? Et ou ?

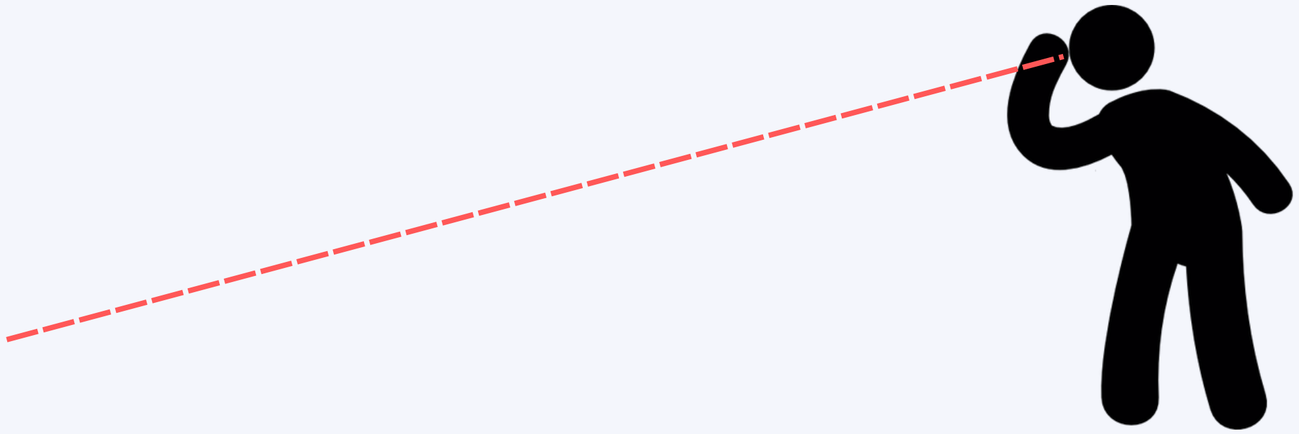
Placer des cubes !

Pour savoir ou le joueur vise, nous pouvons utiliser RayTrace !



Placer des cubes !

RayTrace permet de créer un rayon la ou le joueur regarde. On peut calculer la distance et avoir des informations sur l’objet



Placer des cubes !

Créez un scrip PlayerAction et appliquez le sur le joueur

Le code complet est aussi sur GitHub mais écrivez le bout par bout pour bien le comprendre

Placer des cubes !

On vas utiliser une fonction RayCast qui vas nous renvoyer un objet si il en a trouver un

```
// Raycast to get the first solid object hit
public RaycastHit? GetHit(float maxDistance = 100f)
{
    Ray ray = new Ray(transform.position, transform.forward);
    RaycastHit hit;

    if (Physics.Raycast(ray, out hit, maxDistance))
    {
        return hit; // Return the RaycastHit
    }

    return null; // Nothing hit
}
```

Placer des cubes !

Quand on appele la fonction, créer un nouveau “laser” avec la position et l’orientation du joueur

```
// Raycast to get the first solid object hit
public RaycastHit? GetHit(float maxDistance = 100f)
{
    Ray ray = new Ray(transform.position, transform.forward);
    RaycastHit hit;

    if (Physics.Raycast(ray, out hit, maxDistance))
    {
        return hit; // Return the RaycastHit
    }

    return null; // Nothing hit
}
```

Placer des cubes !

RayCastHit est un type de GameObject avec des informations en plus comme la distance ect.

```
// Raycast to get the first solid object hit
public RaycastHit? GetHit(float maxDistance = 100f)
{
    Ray ray = new Ray(transform.position, transform.forward);
    RaycastHit hit;

    if (Physics.Raycast(ray, out hit, maxDistance))
    {
        return hit; // Return the RaycastHit
    }

    return null; // Nothing hit
}
```

Placer des cubes !

Si on touche un objet avce le “laser” jusqu’à une certaine distance, alors on renvoie l’objet touché

```
// Raycast to get the first solid object hit
public RaycastHit? GetHit(float maxDistance = 100f)
{
    Ray ray = new Ray(transform.position, transform.forward);
    RaycastHit hit;

    if (Physics.Raycast(ray, out hit, maxDistance))
    {
        return hit; // Return the RaycastHit
    }

    return null; // Nothing hit
}
```

Placer des cubes !

Si pas on ne renvoie null

```
// Raycast to get the first solid object hit
public RaycastHit? GetHit(float maxDistance = 100f)
{
    Ray ray = new Ray(transform.position, transform.forward);
    RaycastHit hit;

    if (Physics.Raycast(ray, out hit, maxDistance))
    {
        return hit; // Return the RaycastHit
    }

    return null; // Nothing hit
}
```

Placer des cubes !

Dans Update() on vas appeler notre fonction RayCast quand on utilise le clic gauche

```
if (Input.GetMouseButtonDown(0))
{
    RaycastHit? hit = GetHit();
    if (hit.HasValue)
    {
        Vector3 hitPosition = hit.point;
        Debug.Log(hitPosition);
    }
    else
    {
        Debug.Log("Nothing hit");
    }
}
```

Placer des cubes !

Testez si vous arrivez à avoir la position de votre cube. Il devrait apparaître dans la console

```
if (Input.GetMouseButtonDown(0))
{
    RaycastHit? hit = GetHit();
    if (hit.HasValue)
    {
        Vector3 hitPosition = hit.point;
        Debug.Log(hitPosition);
    }
    else
    {
        Debug.Log("Nothing hit");
    }
}
```

Placer des cubes !

Nous avons les coordonnées de la ou on clique. Mais nous ne savons pas ou placer notre bloque ET qu’il soit alligné .-.

Pour ça, nous allons faire appel à une autre fonction

Placer des cubes !

Ce début de code permet d’arrondir vers le bas les valeurs de notre position

```
// Get the voxel position for block placement (only adjacent face)
public Vector3 GetBlockPlacementPosition(RaycastHit hit)
{
    // Get the hit block position (floor to voxel)
    Vector3 hitBlockPos = new Vector3(
        Mathf.FloorToInt(hit.transform.position.x),
        Mathf.FloorToInt(hit.transform.position.y),
        Mathf.FloorToInt(hit.transform.position.z)
    );
```

Placer des cubes !

On regarde ensuite quel côté du bloc on a touché et on calcule la position du nouveau bloc

```
    // Determine which face was hit
    Vector3 normal = hit.normal;

    // Place the new block adjacent along the axis of the normal
    Vector3 placePos = hitBlockPos;

    if (Mathf.Abs(normal.x) > 0.5f) placePos.x += normal.x > 0 ? 1 : -1;
    else if (Mathf.Abs(normal.y) > 0.5f) placePos.y += normal.y > 0 ? 1 : -1;
    else if (Mathf.Abs(normal.z) > 0.5f) placePos.z += normal.z > 0 ? 1 : -1;

    return placePos;
}
```

Placer des cubes !

On regarde ensuite quel côté du bloc on a touché et on calcule la position du nouveau bloc

```
    // Determine which face was hit
    Vector3 normal = hit.normal;

    // Place the new block adjacent along the axis of the normal
    Vector3 placePos = hitBlockPos;

    if (Mathf.Abs(normal.x) > 0.5f) placePos.x += normal.x > 0 ? 1 : -1;
    else if (Mathf.Abs(normal.y) > 0.5f) placePos.y += normal.y > 0 ? 1 : -1;
    else if (Mathf.Abs(normal.z) > 0.5f) placePos.z += normal.z > 0 ? 1 : -1;

    return placePos;
}
```

Placer des cubes !

On peut enfin créer nos cubes. On récupère la position de ou vas être notre cube. Puis on le créer avec Instantiate

```
if (Input.GetMouseButtonDown(0))
{
    RaycastHit? hit = GetHit();
    if (hit.HasValue)
    {
        Vector3 blockPos = GetBlockPlacementPosition(hit.Value);
        Instantiate(blockPrefab, blockPos, Quaternion.identity);
        Debug.Log("Block placed at: " + blockPos);
    }
```

Placer des cubes !

Mais pour ça, il nous faut un cube (préfab) à placer :)

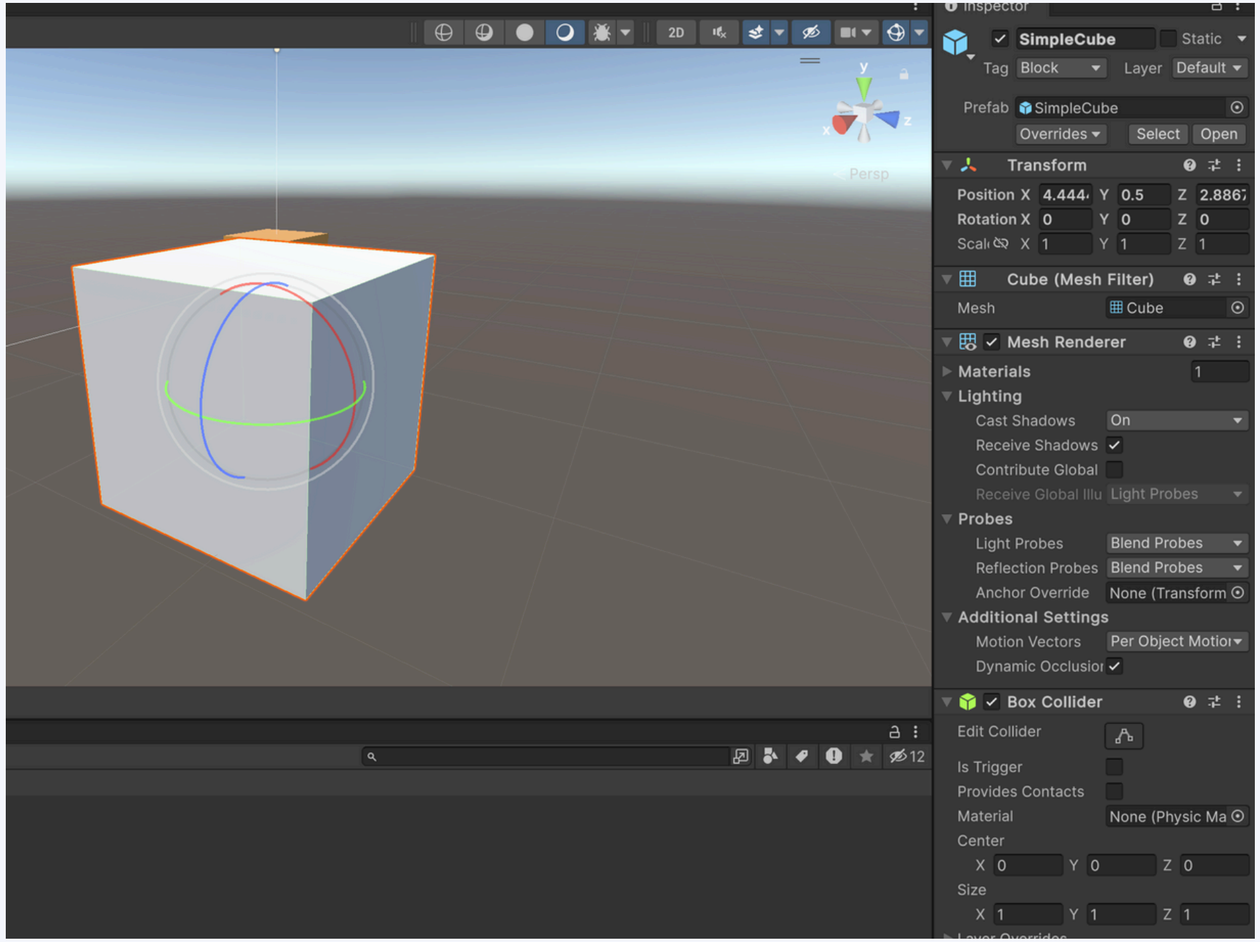
```
if (Input.GetMouseButtonDown(0))
{
    RaycastHit? hit = GetHit();
    if (hit.HasValue)
    {
        Vector3 blockPos = GetBlockPlacementPosition(hit.Value);
        Instantiate(blockPrefab, blockPos, Quaternion.identity);
        Debug.Log("Block placed at: " + blockPos);
    }
```

Placer des cubes !

Créez une nouvelle variable blockPrefab

```
public class PlayerAction : MonoBehaviour
{
    public GameObject blockPrefab;
```

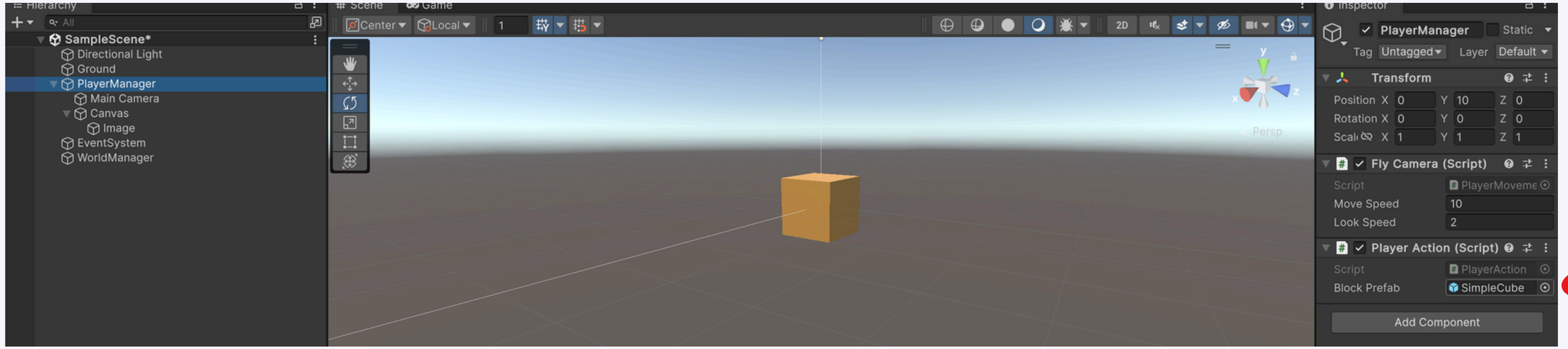
Placer des cubes !



Le préfab doit avoir le tag “Block” et doit être de taille 1. Vous pouvez ajouter un matériel ;)

Placer des cubes !

Glissez le prefab sur le script dans le champ Block Prefab



Détruire des cubes

```
if (Input.GetMouseButtonDown(1))
{
    RaycastHit? hit = GetHit();
    if (hit.HasValue)
    {
        GameObject hitObject = hit.Value.collider.gameObject;

        if (hitObject.CompareTag("Block"))
        {
            Destroy(hitObject);
            Debug.Log("Block destroyed: " + hitObject.name);
        }
        else
        {
            Debug.Log("Hit object is not a block: " + hitObject.name);
        }
    }

    else
    {
        Debug.Log("Nothing hit");
    }
}
```

Même principe ! On regarde ce qu'on a touché quand on utilise le clic droit.
Puis si c'est un bloc on le détruit. Si pas on ne fait rien pour ne pas détruire notre plateforme de base

Sauvegarder votre monde

En Bonus, vous pouvez sauvegarder votre monde avec le script WorldManagerScript

Sauvegarder votre monde

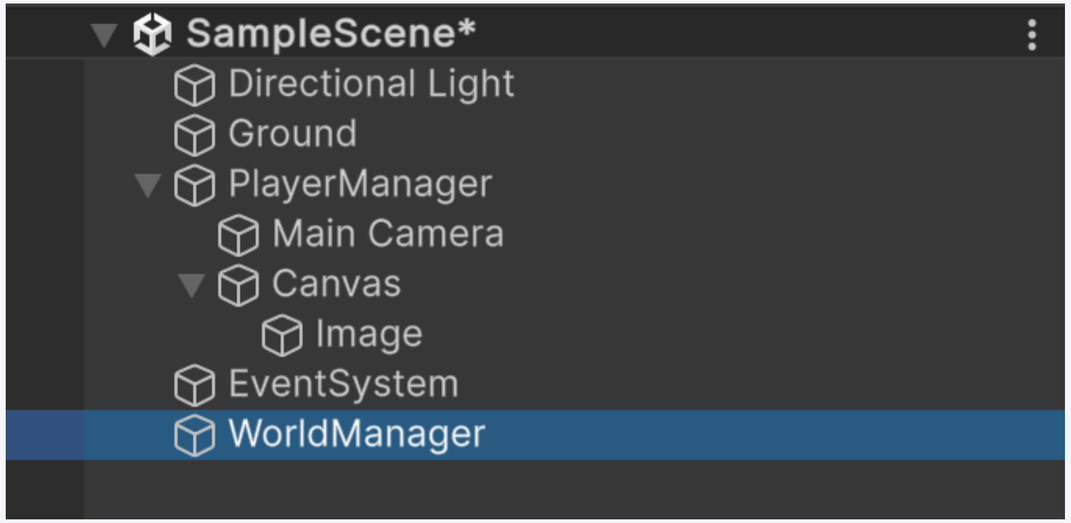
Le script sauvegarde la position de tout les blocs dans un fichier JSON

Sauvegarder votre monde

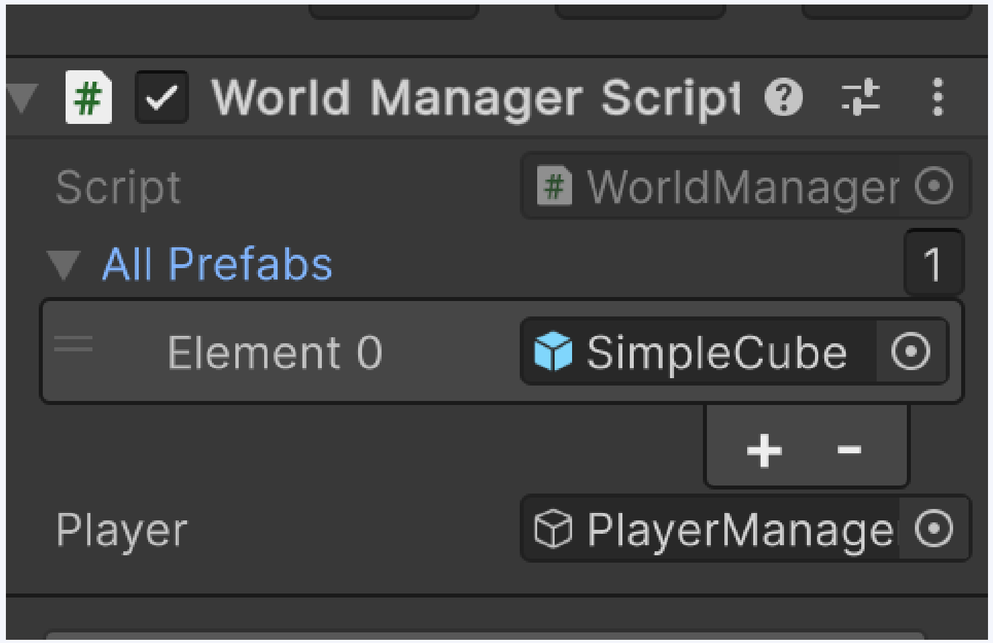
Créez un nouveau Script WorldManagerScript, et copiez collez le code depuis GitHub

Sauvegarder votre monde

Créez un objet vide “WorldManager”



Sauvegarder votre monde



Appuyez sur le + puis glissez votre Cube préfab dans All Prefab

Et pour finir, glissez le PlayerManager dans Player

Et après ?

Plusieurs blocs ?
Un inventaire ?
De la physique ?
Un monde généré ? :)