

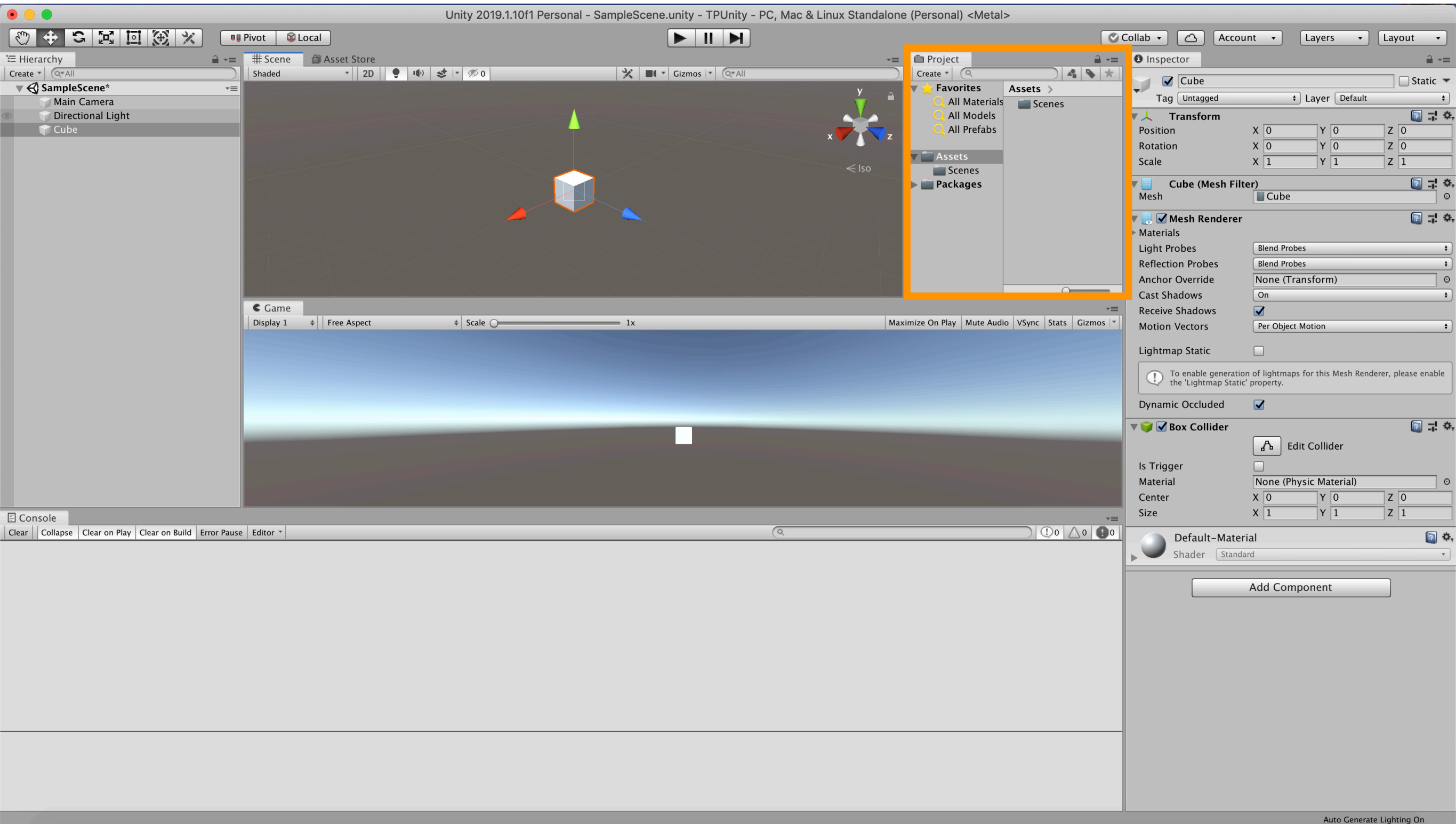
TP Unity

Master 2 Androïde

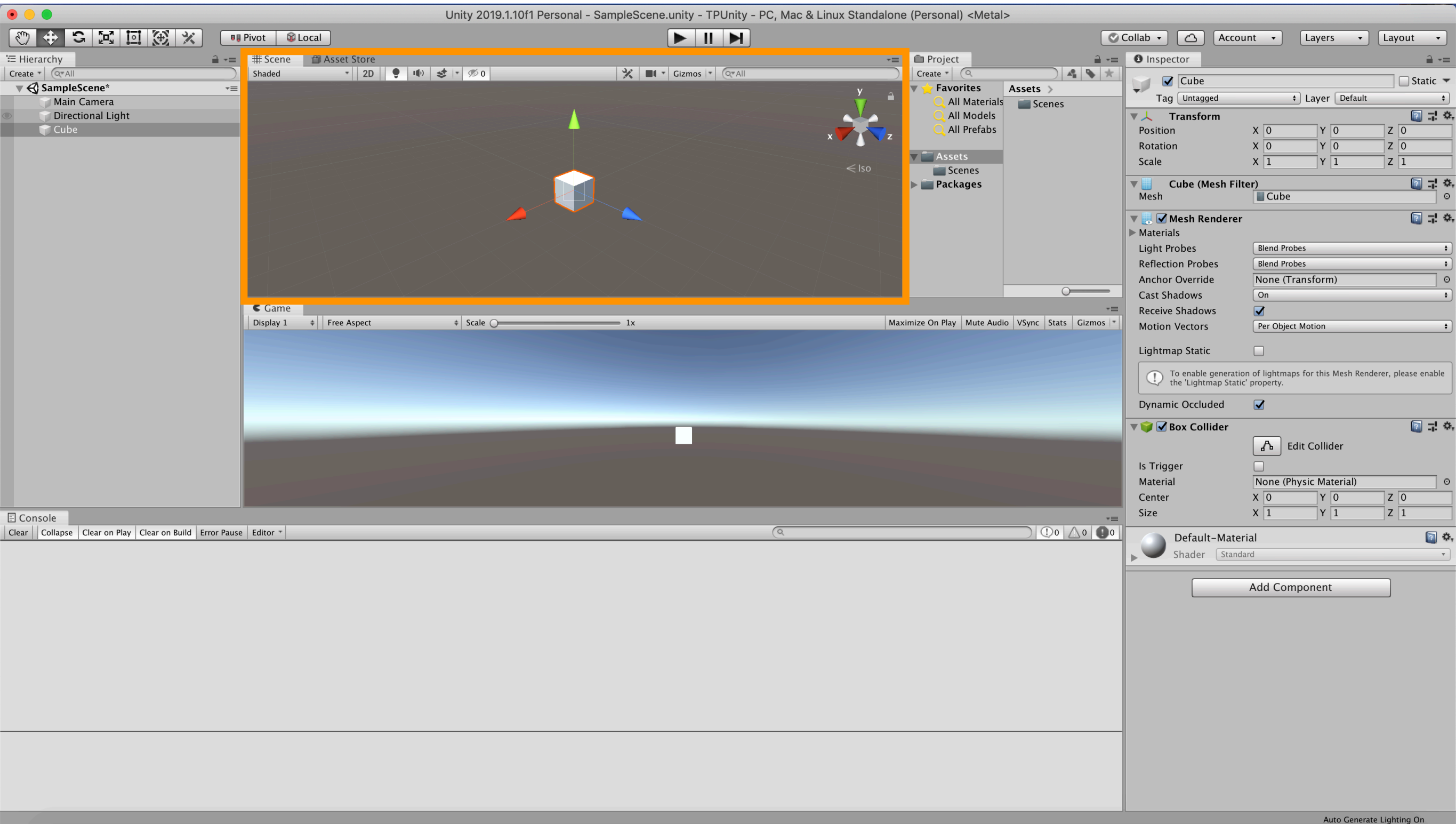
UE EVHI - Environnements virtuels hautement interactifs

Elodie Bouzbib - 19/09/2019

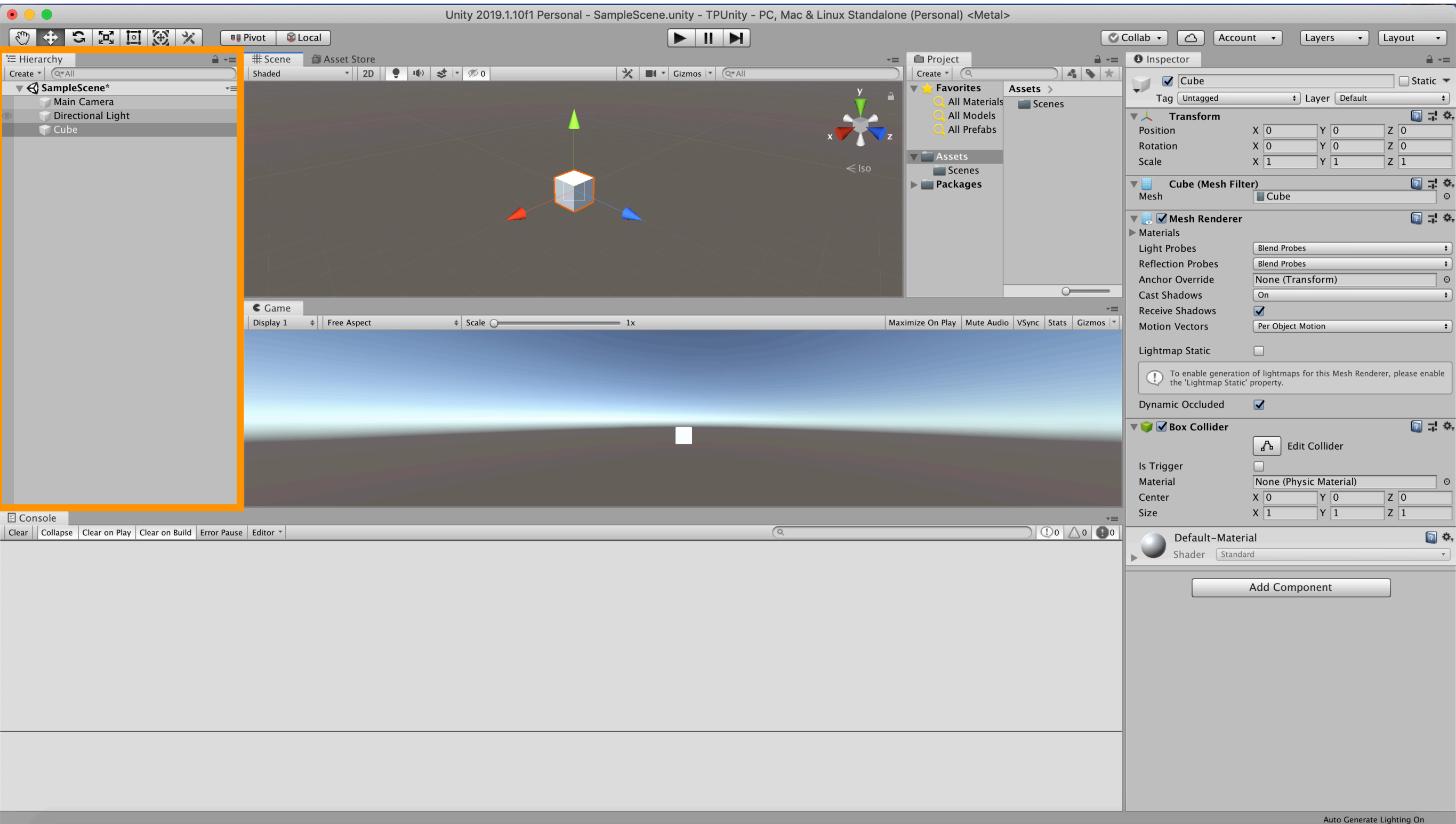
Interface



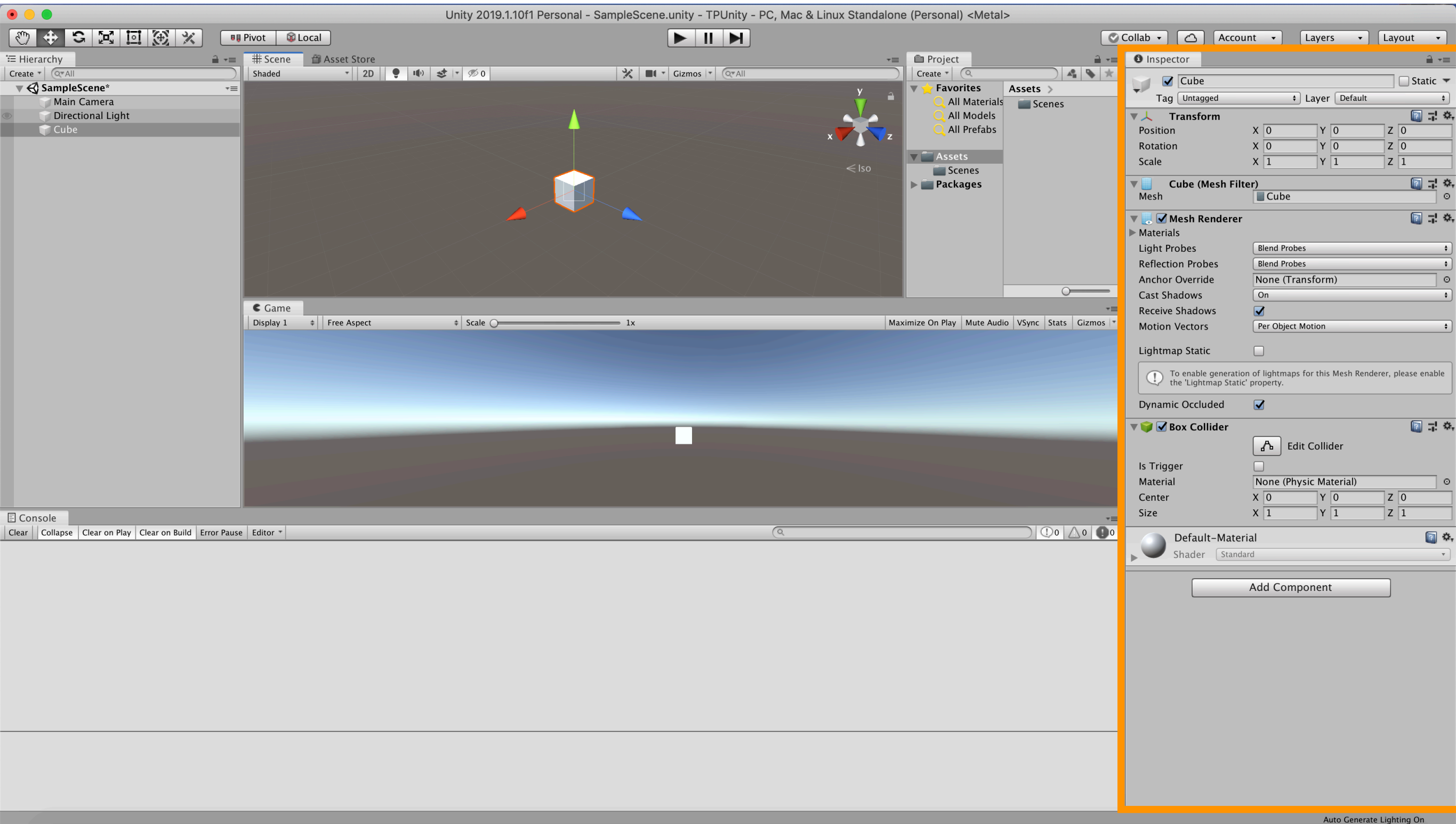
Interface



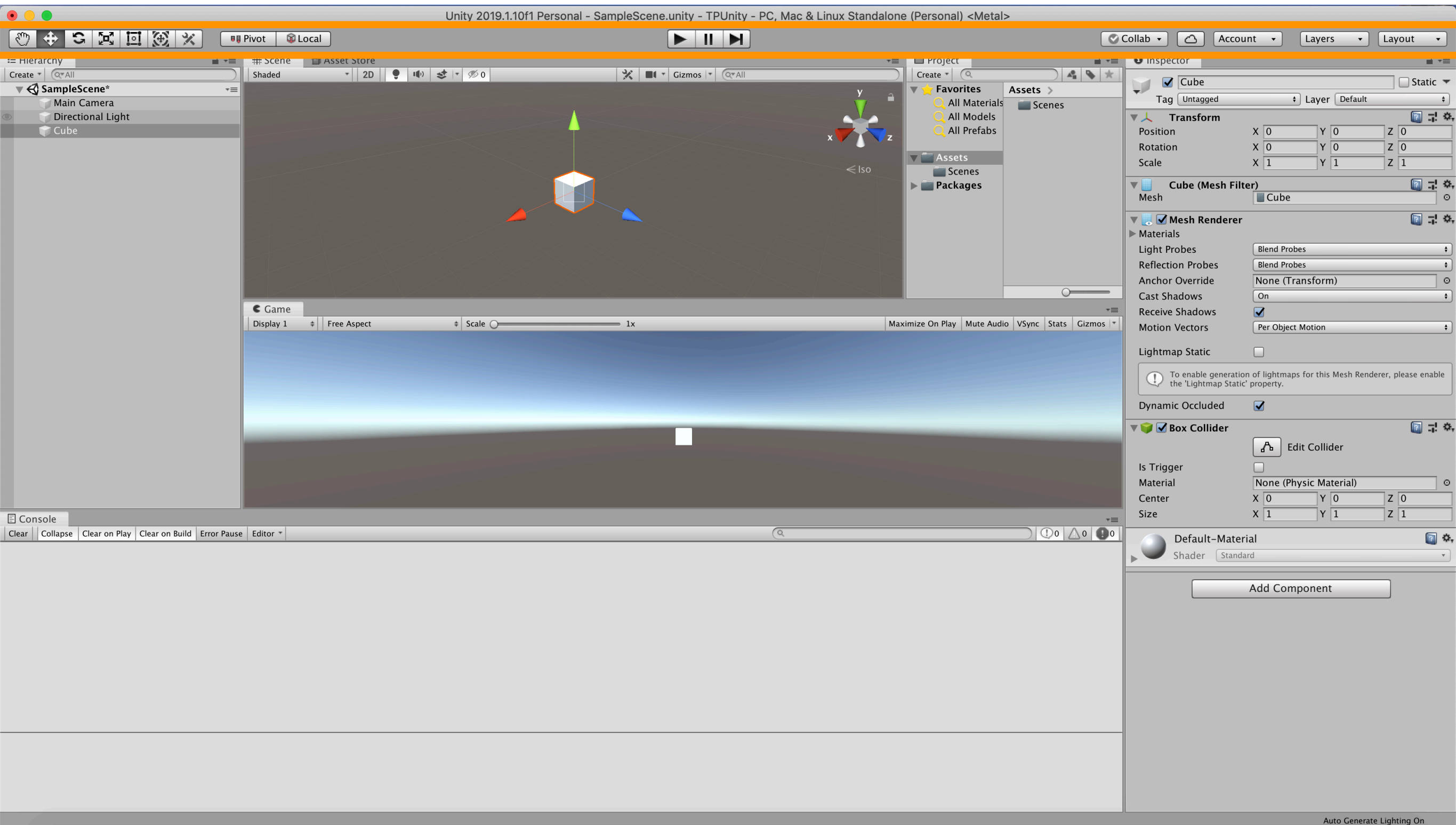
Interface



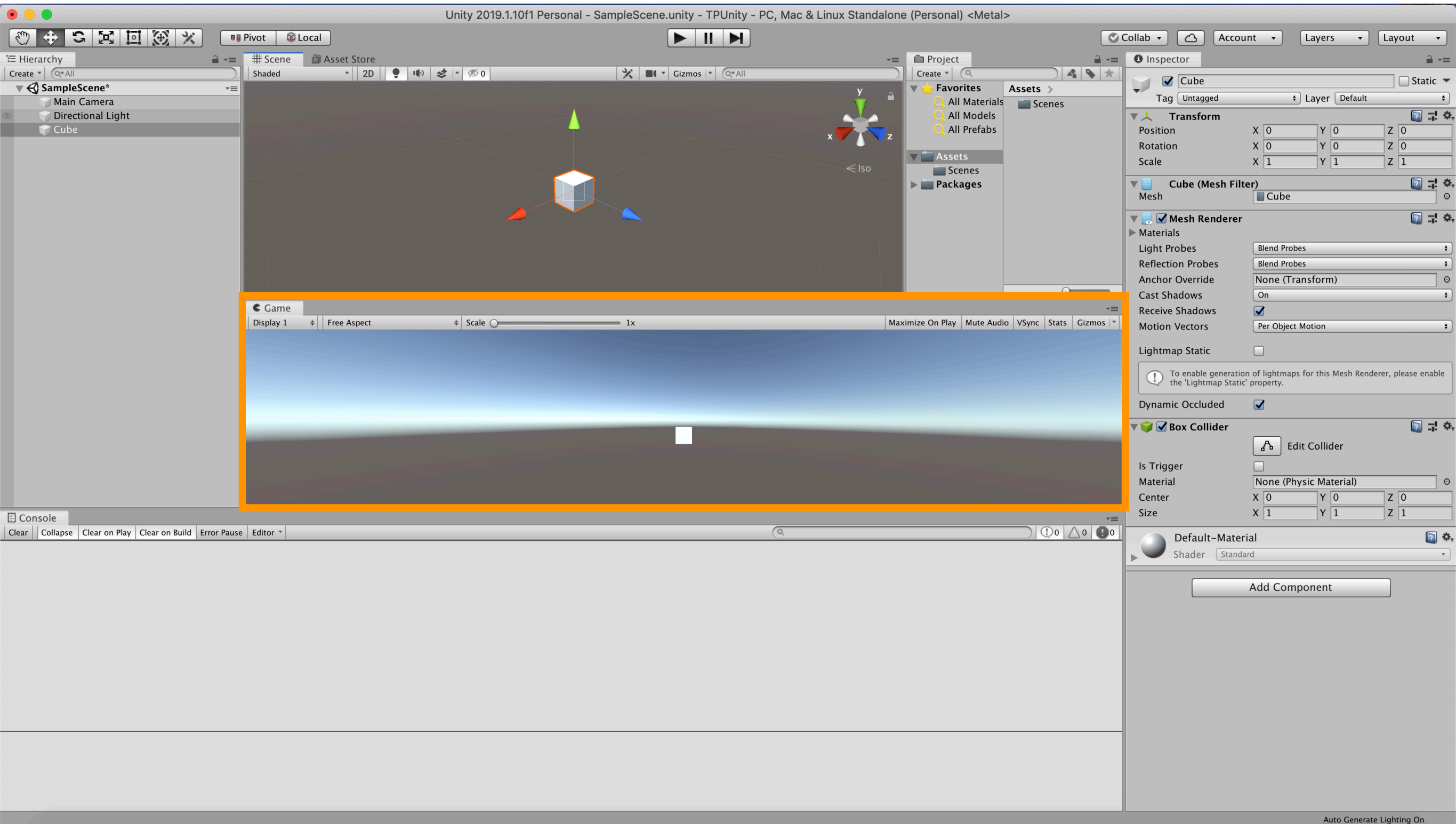
Interface



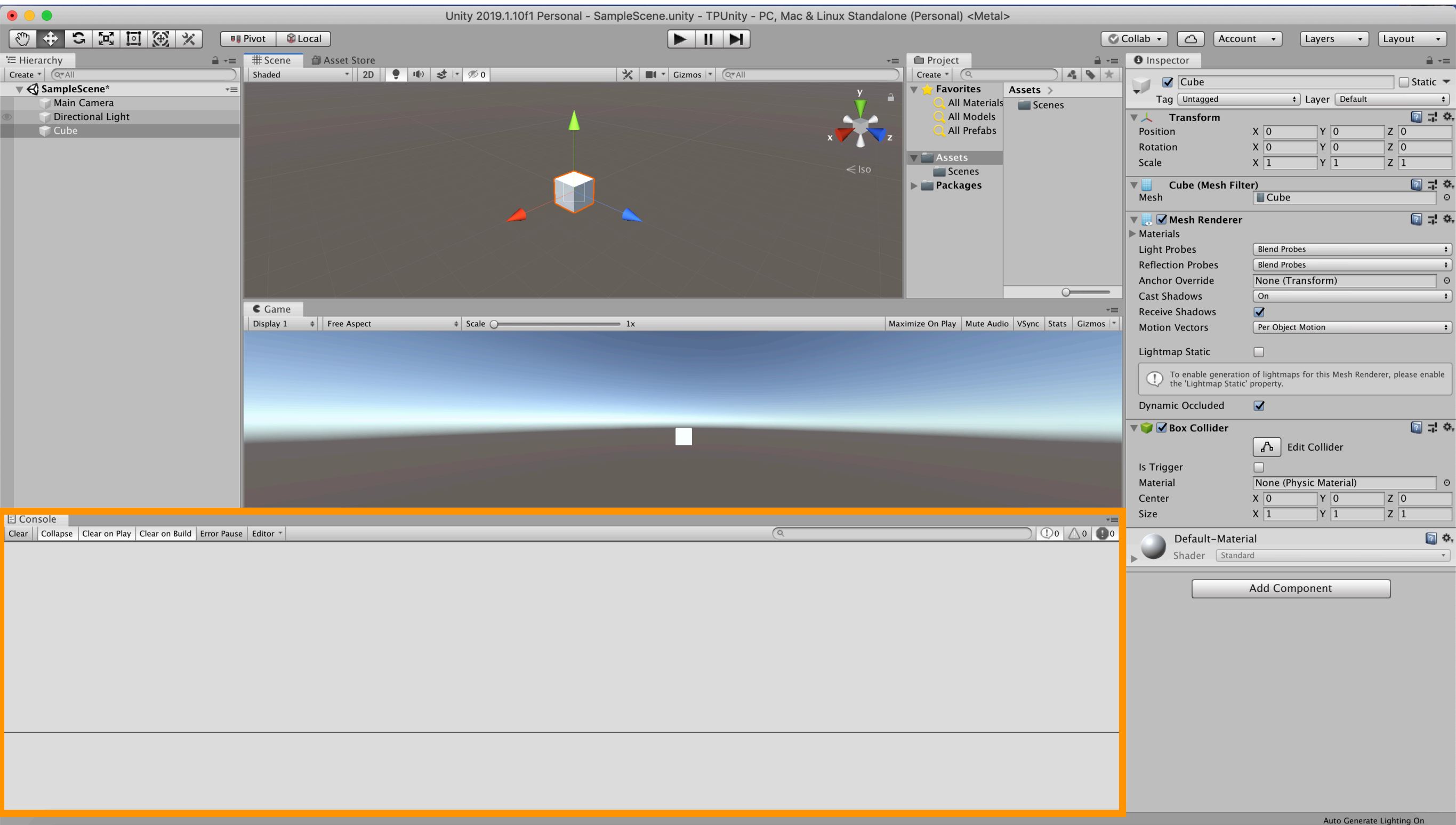
Interface



Interface



Interface



Mini-Projet pour 17/10

- Un jeu avec un personnage.
- L'utilisateur a une barre de vie.
- Des tokens de vie peuvent être rajoutés dans le jeu. Si l'utilisateur en prend, sa couleur se modifie quelques secondes. Sa barre de vie remonte.
- Des étoiles bonus peuvent rendre l'utilisateur invincible aux ennemis pendant 10secondes.

Mini-Projet pour 17/10

- 5 niveaux
 - Entrer dans une salle, pousser un bouton. Le bouton ouvre une porte. Entrer niveau 2.
 - Un ennemi est en mouvement devant la seconde porte. Il faut lui lancer un missile pour le détruire. Puis pousser le bouton. Entrer niveau 3.
 - Un ennemi poursuit l'utilisateur. Le bouton de sortie est dans un isoloir avec une porte ouverte. Il faut pousser le bouton. La porte se ferme. Entrer le niveau 4.
 - L'environnement possède des trous. Si l'utilisateur tombe il perd de la vie. Arriver jusqu'au bouton. Pousser le bouton. Entrer le niveau 5.
 - L'utilisateur arrive devant des portes battantes. Il entre. Il n'y a pas de lumière mais un ajusteur de lumière rayonne. Il faut allumer la lumière. Quand la lumière est pleine, un gros ennemi avec une barre de vie se déplace. Il faut donc le toucher plusieurs fois. Si on le tue, on a gagné.

Mini-Projet pour 17/10

- On doit pouvoir quitter le jeu à tout instant.
- On doit utiliser le clavier et la souris pour se déplacer, ainsi que pour tirer.
- Les ennemis doivent changer de couleur quand on les touche. L'utilisateur aussi quand il gagne des étoiles ou de la vie.
- Les portes changent de couleur quand on les active.
- Si l'utilisateur perd, on doit voir "Jouer encore ?", "Oui", "Non".
- Si l'utilisateur gagne, on voit un "Félicitations". Le jeu se quitte.
- On doit pouvoir Build le jeu.
- Un Git doit être disponible.

Scripts et Behaviour

- Changer la couleur de l'élément

```
using UnityEngine;
using System.Collections;

public class ExampleBehaviourScript : MonoBehaviour
{
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.R))
        {
            GetComponent<Renderer>().material.color = Color.red;
        }
        if (Input.GetKeyDown(KeyCode.G))
        {
            GetComponent<Renderer>().material.color = Color.green;
        }
        if (Input.GetKeyDown(KeyCode.B))
        {
            GetComponent<Renderer>().material.color = Color.blue;
        }
    }
}
```

Variables et Fonctions

- Multiplier par deux et regarder dans la console

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MultiplyWhenPress : MonoBehaviour
{
    public float myFloat 5.0f;
    private float multipliedFloat;

    void Update(){
        if(GetKeyDown(KeyCode.Space)){
            multipliedFloat = MultiplyByTwo(myFloat);
            Debug.Log("Multiplied Float = " + multipliedFloat);
        }
    }

    float MultiplyByTwo(float number){
        float ret;
        ret = number * 2;
        return ret;
    }
}
```

Boucles

- Pas de boucle while
- Boucles for et boucle if OK.

Modificateur d'accès

- Créer un code :
 - On a une tasse de café à 85 degrés. Tant que le café n'est pas en dessous de 40 degrés, le Player ne peut pas boire.
 - Un code Player, avec un booléen qui `Debug.Log("Je peux boire");`
 - Un code café, avec les températures qui baissent quand on appuie sur `KeyCode.Space` et qui enclenche un booléen "tempOK";

Différences entre Awake(), Start(), Update(), FixedUpdate(), LateUpdate()

- Faire un Debug.Log dans chacun des cas pour comprendre leurs différences.
- Faire un Debug.Log quand l'objet n'est pas enabled.

```
using UnityEngine;
using System.Collections;

public class UpdateAndFixedUpdate : MonoBehaviour
{
    void Awake ()
    {
        Debug.Log("Awake called.");
    }

    void Start ()
    {
        Debug.Log("Start called.");
    }

    void FixedUpdate ()
    {
        Debug.Log("FixedUpdate time : " + Time.deltaTime);
    }

    void Update ()
    {
        Debug.Log("Update time : " + Time.deltaTime);
    }
}
```

Bibliothèque Mathf

- Tout ce qui concerne le calcul, on utilise Mathf.
 - Exemple : `Mathf.Abs()`, `Mathf.Sqrt()`...
- Pour les vecteurs, on peut directement utiliser `Vector3` (ou `Vector2`)
 - Exemple : `Vector3.magnitude`, `Vector3.Dot(VectorA, VectorB)`...

Différence entre enable et Activate

- Enable/Disable le composant "Light".
- Deactivate/Reactive l'objet "unAutreObjet"

```
public class EnableComponents : MonoBehaviour
{
    private Light myLight;

    void Start ()
    {
        myLight = GetComponent<Light>();
    }

    void Update ()
    {
        if(Input.GetKeyUp(KeyCode.Space))
        {
            myLight.enabled = !myLight.enabled;
            GameObject.Find("unAutreObjet").SetActive(false);
        }
    }
}
```

Translate & Rotate

- On doit multiplier par Time.deltaTime pour déplacer en m/s plutôt qu'en m/frame.

```
using UnityEngine;
using System.Collections;

public class TransformFunctions : MonoBehaviour
{
    public float moveSpeed = 10f;
    public float turnSpeed = 50f;

    void Update ()
    {
        if(Input.GetKey(KeyCode.UpArrow))
            transform.Translate(Vector3.forward * moveSpeed * Time.deltaTime);

        if(Input.GetKey(KeyCode.DownArrow))
            transform.Translate(-Vector3.forward * moveSpeed * Time.deltaTime);

        if(Input.GetKey(KeyCode.LeftArrow))
            transform.Rotate(Vector3.up, -turnSpeed * Time.deltaTime);

        if(Input.GetKey(KeyCode.RightArrow))
            transform.Rotate(Vector3.up, turnSpeed * Time.deltaTime);
    }
}
```

LookAt

- Sert à orienter le "forward" d'un gameObject.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TPUnity : MonoBehaviour
{
    public Transform target;

    void Update ()
    {
        transform.LookAt(target);
    }
}
```

Linear Interpolation : Lerp

- La fonction Lerp sert à trouver une valeur qui est à un pourcentage entre deux valeurs.
- Fonctionne à la fois en `Mathf.Lerp`, `Vector3.Lerp`, `Color.Lerp`.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TPUntity : MonoBehaviour
{
    public Transform target;

    void Update ()
    {
        GetComponent<Light>().intensity = Mathf.Lerp(GetComponent<Light>().intensity, 8f, 0.5f);
    }
    // Fonctionne aussi avec 0.5f * Time.deltaTime par exemple

    // En math : Mathf.Lerp(3f, 5f, 0.5f) donnera 4;
    // En Vector3 : from = new Vector3(1f, 2f, 3f);
    // to = new Vector3(5f, 6f, 7f);
    // result = (4,5,6)
}
```

Destroy

- Fonctionne à la fois pour des objets et des composants.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TPUnit : MonoBehaviour
{
    public Transform target;

    public GameObject other;

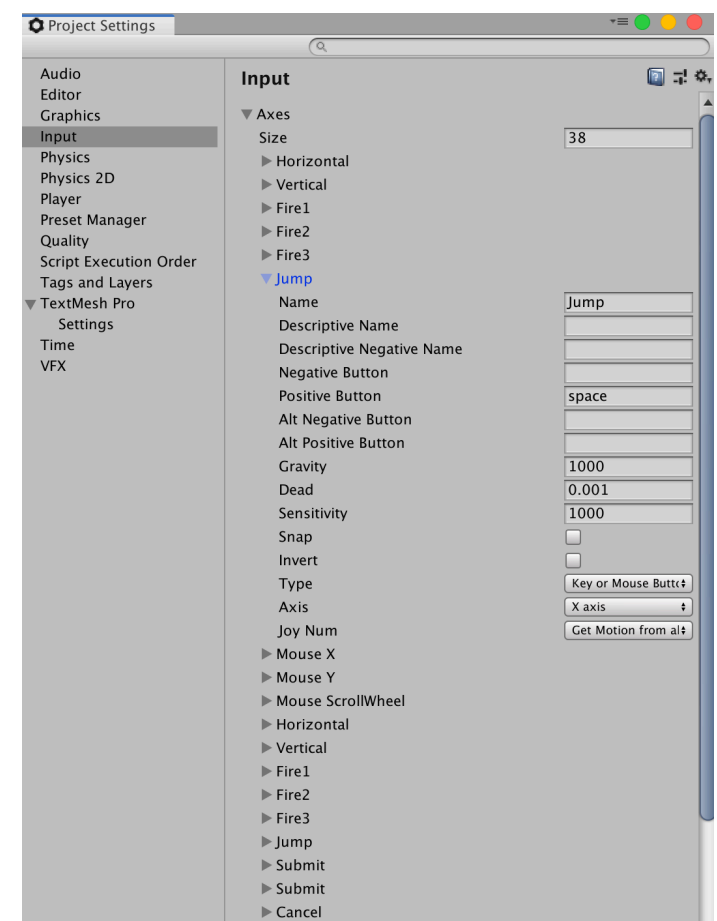
    void Update ()
    {
        if(Input.GetKey(KeyCode.Space))
        {
            Destroy(other);
        }
        if(Input.GetKey(KeyCode.A))
        {
            Destroy(other.GetComponent<MeshRenderer>());
        }
    }
}
```


GetButton, GetKey

- Buttons : Dans Edit, Project Settings, on trouve les Inputs.
- Trois états : GetButtonDown, GetButton, GetButtonUp
- GetKeyDown, GetKey, GetKeyUp

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TPUntity : MonoBehaviour
{
    // On a donc pareil pour :
    bool down = Input.GetKeyDown(KeyCode.Space);
    bool down = Input.GetButtonDown("Jump");
    // mais KeyDown est plus générique
}
```



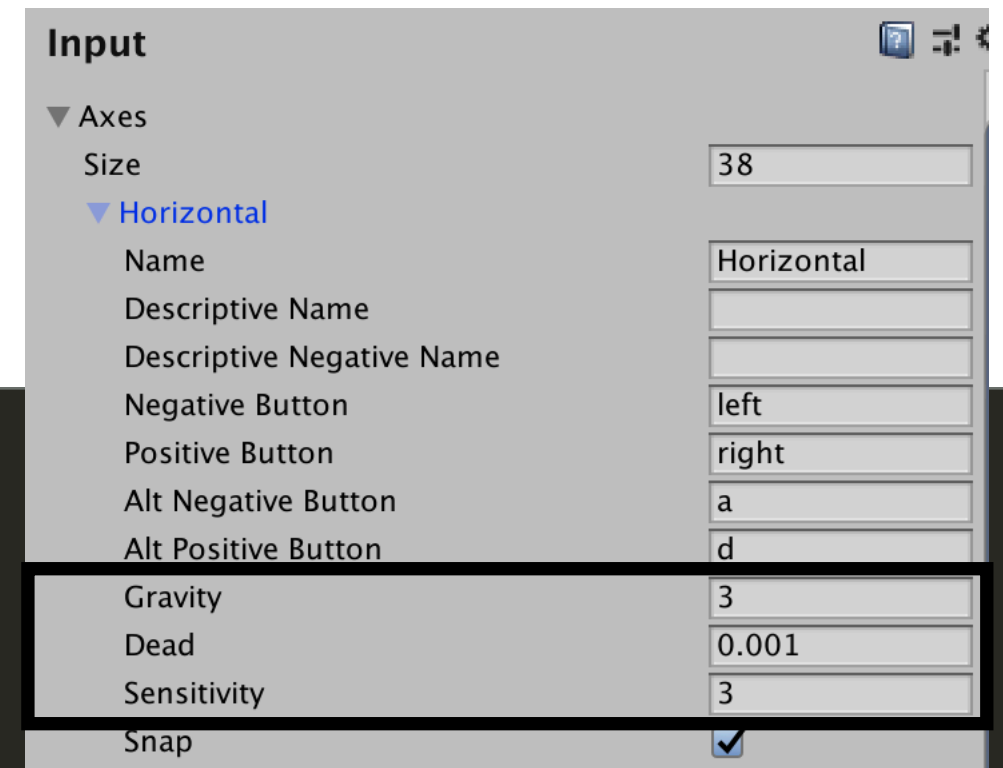
GetAxis

- Pareil que GetButton/GetKey mais ne renvoie pas un booléen.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class TPUnity : MonoBehaviour
{
    public float range;
    public Text textOutput;

    void Update ()
    {
        float h = Input.GetAxis("Horizontal");
        float v = Input.GetAxis("Vertical");
        float xPos = h * range;
        float yPos = v * range;
        transform.position = new Vector3(xPos, yPos, 0);
        textOutput.text = "Horizontal Value Returned: "+h.ToString("F2")+"\nVertical Value Returned: "+v.ToString("F2");
    }
}
```



The screenshot shows the Unity Inspector window with the 'Input' component selected. The 'Axes' section is expanded, showing the 'Horizontal' axis. The 'Size' is set to 38. The 'Name' is 'Horizontal'. The 'Descriptive Name' is empty. The 'Descriptive Negative Name' is empty. The 'Negative Button' is 'left'. The 'Positive Button' is 'right'. The 'Alt Negative Button' is 'a'. The 'Alt Positive Button' is 'd'. The 'Gravity' is 3. The 'Dead' is 0.001. The 'Sensitivity' is 3. The 'Snap' checkbox is checked.

Input	
▼ Axes	
Size	38
▼ Horizontal	
Name	Horizontal
Descriptive Name	
Descriptive Negative Name	
Negative Button	left
Positive Button	right
Alt Negative Button	a
Alt Positive Button	d
Gravity	3
Dead	0.001
Sensitivity	3
Snap	<input checked="" type="checkbox"/>

OnClick

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class TPUntity : MonoBehaviour
{
    void OnMouseDown ()
    {
        Debug.Log("Click");
    }
    // Autre exemple :
    // Il faut ajouter un rigidbody (fonctionne aussi sur les UI)
    // void OnMouseDown ()
    // {
    //     GetComponent<Rigidbody>().AddForce(-transform.forward * 500f);
    //     GetComponent<Rigidbody>().useGravity = true;
    // }
}
```

GetComponent

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class TPUntity : MonoBehaviour
{
    public GameObject otherGameObject;
    private AnotherScript anotherScript;

    void Awake ()
    {
        anotherScript = GetComponent<AnotherScript>();
    }

    void Start ()
    {
        Debug.Log("The player's score is " + anotherScript.playerScore);
    }
}
```

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class AnotherScript : MonoBehaviour
5  {
6      public int playerScore = 9001;
7  }
```

Time.deltaTime

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class TPUntity : MonoBehaviour
{
    public float speed = 8f;
    public float countdown = 3.0f;

    void Update ()
    {
        countdown -= Time.deltaTime;
        if(countdown <= 0.0f)
            light.enabled = true;

        if(Input.GetKey(KeyCode.RightArrow))
            transform.position += new Vector3(speed * Time.deltaTime, 0.0f, 0.0f);
    }
}
```

Classes

```
6 public class TPUnit : MonoBehaviour
7 {
8     public class Inventory : MonoBehaviour
9     {
10         public class Stuff
11         {
12             public int bullets;
13             public int grenades;
14             public int rockets;
15             public float fuel;
16
17             public Stuff(int bul, int gre, int roc)
18             {
19                 bullets = bul;
20                 grenades = gre;
21                 rockets = roc;
22             }
23
24             public Stuff(int bul, float fu)
25             {
26                 bullets = bul;
27                 fuel = fu;
28             }
29
30             // Constructor
31             public Stuff ()
32             {
33                 bullets = 1;
34                 grenades = 1;
35                 rockets = 1;
36             }
37         }
38
39         // Creating an Instance (an Object) of the Stuff class
40         public Stuff myStuff = new Stuff(50, 5, 5);
41
42         public Stuff myOtherStuff = new Stuff(50, 1.5f);
43
44         void Start()
45         {
46             Debug.Log(myStuff.bullets);
47         }
48     }
49 }
50 }
```

Instantiate

- Utile pour cloner des GameObjects et surtout des préfabs.
- Attention : Crée des clones. Penser à Destroy après un certain temps.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class TPUnit : MonoBehaviour
{
    public Rigidbody rocketPrefab;
    public Transform barrelEnd;

    void Update ()
    {
        if(Input.GetButtonDown("Fire1"))
        {
            Rigidbody rocketInstance;
            rocketInstance = Instantiate(rocketPrefab, barrelEnd.position, barrelEnd.rotation) as Rigidbody;
            rocketInstance.AddForce(barrelEnd.up * 5000);
        }
    }
}
```


Matrices

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class TPUnity : MonoBehaviour
{
    public GameObject[] players; // type d'array

    void Start ()
    {
        players = GameObject.FindGameObjectsWithTag("Player"); // !Jouer sur les tags!

        for(int i = 0; i < players.Length; i++)
        {
            Debug.Log("Player Number "+i+" is named "+players[i].name);
        }
    }
}
```

Invoke

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class TPUntity : MonoBehaviour
{
    public GameObject target;
    public GameObject target2;

    void Start()
    {
        Invoke ("SpawnObject", 2); // délai pour démarrer
        InvokeRepeating("SpawnObject2", 2, 1); // delai pour demarrer
        // puis delai pour répéter
    }

    void SpawnObject()
    {
        Instantiate(target, new Vector3(0, 2, 0), Quaternion.identity);
    }

    void SpawnObject2()
    {
        float x = Random.Range(-2.0f, 2.0f);
        float z = Random.Range(-2.0f, 2.0f);
        Instantiate(target2, new Vector3(x, 2, z), Quaternion.identity);
    }
}
```

Enumérations

```
public class TPUnity : MonoBehaviour
{
    public enum Direction
    {
        North,
        East,
        South,
        West
    };

    void Start ()
    {
        Direction myDirection;

        myDirection = Direction.North;
    }

    Direction ReverseDirection (Direction dir)
    {
        if(dir == Direction.North)
            dir = Direction.South;
        else if(dir == Direction.South)
            dir = Direction.North;
        else if(dir == Direction.East)
            dir = Direction.West;
        else if(dir == Direction.West)
            dir = Direction.East;

        return dir;
    }
}
```

Switch

- "Machine à état"

```
public class TPUntity : MonoBehaviour
{
    public int intelligence = 5;

    void Greet()
    {
        switch (intelligence)
        {
            case 5:
                print ("Why hello there good sir! Let me teach you about Trigonometry!");
                break;
            case 4:
                print ("Hello and good day!");
                break;
            case 3:
                print ("Whadya want?");
                break;
            case 2:
                print ("Grog SMASH!");
                break;
            case 1:
                print ("Ulg, glib, Pblblblblb");
                break;
            default:
                print ("Incorrect intelligence level.");
                break;
        }
    }
}
```

Quelques aides pour déclarer des variables

```
public class TPUntity : MonoBehaviour
{
    [RequireComponent( typeof( XXXX) )] // Associer un composant de force pour ne pas l'oublier

    [Range(0.0f, 280.0f)] // Avoir un slider dans son Inspector
    public float positionX;

    [SerializeField] // force Unity à serialiser un objet privé
    private bool content; // ie est dans l'Inspector ET la valeur est sauvée dans la scène
}
```