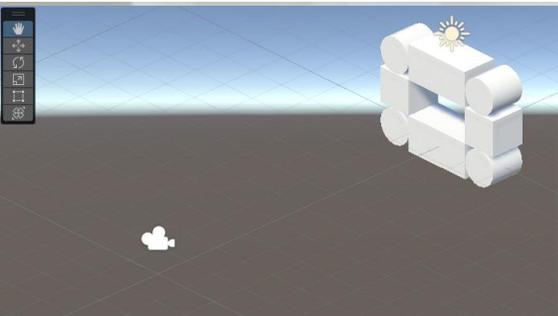


---

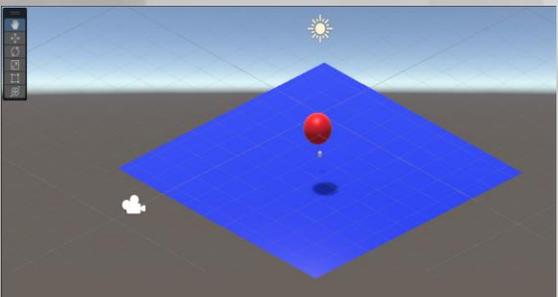
# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Ce qui a été construit lors de la séance du 04/10/2023



## Scène 1

- Découverte de l'environnement Unity
- Component **Transform**, notion d'**Empty Object**
- Construction et placement précis de **3D Objects** simples
- Hiérarchie entre les objets



## Scène 2

- Utilisation d'un objet construit avec un modeleur, format .FBX
- **Physicalisation**: Components **Renderer(Material)**, **Rigidbody**, **Sphere Collider**
- Importance de **isTrigger (Sphere Collider)** et de **isKinematic (RigidBody)**

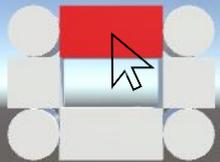
## Scène 1, 2 et 3

- Premiers codes C#, classe **MonoBehavior**, fonctions événementielles **Start()**, **Update()** et **OnTriggerEnter()**
- Affichage dans la console
- Accès aux propriétés des **GameObject** (name) et des **Component (Transform, Rigidbody)**

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Sélection d'un objet graphique à partir d'un ensemble d'objets graphiques et manipulation de l'objet sélectionné



## Etape 1 – Rendre visible un objet sélectionnable et afficher son nom dans la console

- Reprenez **scene1** et renommez-la en **scene4**
- **A FAIRE** rendre rouge 1 des 8 objets de la scène lorsque le curseur de la souris passe dessus, puis afficher dans la Console l'objet sélectionnable.

### ▪ COMMENT FAIRE?

- Créez un script **SelectionObjet.cs** placé sous tous les objets graphiques potentiellement sélectionnables, utilisant les fonctions prédéfinies **void OnMouseEnter()** et **void OnMouseExit()**
- Afin de contrôler la couleur d'un objet graphique, utilisez une variable **Renderer renderer** associée à votre **GameObject** graphique, afin de mettre à jour sa couleur: **GetComponent<Renderer>().material.color = Color.red** met à **red** le **Material** de l'objet qui porte **SelectionObjet.cs**.
- Appliquez le Script à tous les objets graphiques potentiellement **sélectionnables**

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Sélection d'un objet graphique à partir d'un ensemble d'objets graphiques et manipulation de l'objet sélectionné

```
public class SelectionObjet : MonoBehaviour
{
    private Color oldColor;
```



---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Sélection d'un objet graphique à partir d'un ensemble d'objets graphiques et manipulation de l'objet sélectionné

```
private void OnMouseEnter()  
{  
    oldColor = GetComponent<Renderer>().material.color; // On sauve la couleur courante de  
l'objet porté qui porte le script  
    GetComponent<Renderer>().material.color = Color.red; // La couleur de l'objet qui porte  
le script devient rouge  
    Debug.Log("[ " + transform.name + " SELECTIONNABLE]");  
}  
  
private void OnMouseExit()  
{  
    GetComponent<Renderer>().material.color = oldColor; // La couleur de l'objet qui porte  
le script devient ce qu'elle était avant que l'objet soit sélectionnable  
    Debug.Log("[ " + transform.name + " N'EST PLUS SELECTIONNABLE]");  
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Sélection d'un objet graphique à partir d'un ensemble d'objets graphiques

### Etape 2 – Sélectionner un objet par un clic-gauche de la souris et afficher son nom sur la Console

- **A FAIRE** Après le clic souris, afficher sur la Console le nom de l'objet sélectionné et désactiver la possibilité pour d'autres objets de devenir sélectionnables.
- **COMMENT FAIRE?**
  - Créez un script **GereSelection.cs**, à placer sur **GrosObjet**, utilisant les fonctions **Input.GetMouseButtonDown()**. **Input.GetMouseButtonDown(0)** renvoie **true** si appuie sur le bouton gauche de la souris, **false** sinon.
  - Créez 2 variables publiques de type **GameObject** **public** GameObject **objetSelectionne** et **public** GameObject **objetSelectionnable**;
  - Mettre à jour **objetSelectionnable** à partir des différents objets graphiques, dans **SelectionObjet**
    - **COMMENT?** Dans le script **SelectionObjet** porté par un objet graphique, utilisez une variable **GereSelection** **gs**: **gs = this.transform.parent.GetComponent<GereSelection>()** pointe vers **GereSelection** porté par GrosObjet. Puis accès à **gs.objetSelectionnable** et **gs.objetSelectionne**

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Sélection d'un objet graphique à partir d'un ensemble d'objets graphiques

```
public class GereSelection : MonoBehaviour
{
    public GameObject objetSelectionne = null; // Pointe vers l'objet sélectionnable
    public GameObject objetSelectionnable = null; // Pointe vers l'objet sélectionné
    private GameObject[] tfils; // tableau contenant la liste des GameObject fils de
    GrosObjet
    private int nfils;
}
```



---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Sélection d'un objet graphique à partir d'un ensemble d'objets graphiques

```
// Initialisation du tableau tfils[], de objetSelectionnable, de objetSelectionne
void Start()
{
    objetSelectionne = null;
    objetSelectionnable = null;
    // Récupère tous les fils de GrosObjet
    Transform[] ts = this.gameObject.GetComponentInChildren<Transform>();
    if (ts == null)
        Debug.Log(this.gameObject.name + " n'a pas de fils");
    else
    {
        // Accède aux GameObject associé à chacun des Transform de ts et met à jour nfils
    }
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Sélection d'un objet graphique à partir d'un ensemble d'objets graphiques

```
else
{
    nfiles = ts.Length-1;
    Debug.Log(this.gameObject.name + " possède " + nfiles + " fils");
    tfils = new GameObject[nfiles];
    int i = 0;
    foreach (Transform t in ts)
    {
        if (t.name != this.gameObject.name) // GrosObjet est dans ts, donc ne pas le rajouter
        {
            tfils[i] = t.gameObject;
            i++;
        }
    }
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Sélection d'un objet graphique à partir d'un ensemble d'objets graphiques

```
void Update()
{
    if (
    ) // Appuie sur le bouton gauche de la souris et un objet est sélectionnable?
    {
        int indexSelectionne = objetGetIndex(objetSelectionnable.name); // index de l'objet
sélectionné dans tfils[]
        Debug.Log("*** " + objetSelectionnable.name + " est sélectionné");
        objetSelectionne = objetSelectionnable;
        tfils[indexSelectionne].GetComponent<SelectionObjet>().SetSelectionne(); // L'objet
est sélectionné
        objetSelectionnable = null;
    }
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Sélection d'un objet graphique à partir d'un ensemble d'objets graphiques

```
void Update()
{
    if (Input.GetMouseButtonDown(0) && (objetSelectionnable != null) && (objetSelectionne == null)) // Appuie sur le bouton gauche de la souris et un objet est sélectionnable?
    {
        int indexSelectionne = objetGetIndex(objetSelectionnable.name); // index de l'objet sélectionné dans tfils[]
        Debug.Log("*** " + objetSelectionnable.name + " est sélectionné");
        objetSelectionne = objetSelectionnable;
        tfils[indexSelectionne].GetComponent<SelectionObjet>().SetSelectionne(); // L'objet est sélectionné
        objetSelectionnable = null;
    }
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Sélection d'un objet graphique à partir d'un ensemble d'objets graphiques

```
// Pre: name est le nom d'un des objets fils de GrosObjet
// Post: index du GameObject de nom 'name' dans le tableau tfils[], ou -1 si name n'est
pas un fils de GrosObjet
private int objetGetIndex( string name)
{
    for (int i = 0; i < nfils; i++)
        if (name == tfils[i].name)
            return i;
    return -1; // Oups, erreur, name est introuvable, ce qui ne devrait pas arriver ...
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Sélection d'un objet graphique à partir d'un ensemble d'objets graphiques

```
public class SelectionObjet : MonoBehaviour
{
    private Color oldColor;
    private GereSelection gs;

    void Start()
    {
        gs = this.transform.parent.GetComponent<GereSelection>(); // gs pointe vers le Script
        GereSelection, porté par GrosObjet
    }
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Sélection d'un objet graphique à partir d'un ensemble d'objets

```
private void OnMouseEnter()  
{  
    oldColor = GetComponent<Renderer>().material.color; // On sauve la couleur courante de  
l'objet porté qui porte le script  
    GetComponent<Renderer>().material.color = Color.red; // La couleur de l'objet qui porte  
le script devient rouge  
    gs.objetSelectionnable = transform.gameObject; // Met l'information de l'objet  
graphique sélectionnable dans GereSelection, porté par GrosObjet  
    Debug.Log("[ " + transform.name + " SELECTIONNABLE]");  
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Sélection d'un objet graphique à partir d'un ensemble d'objets graphiques

```
// L'objet qui porte le script reprend sa couleur initiale
private void OnMouseExit()
{
    GetComponent<Renderer>().material.color = oldColor;
    gs.objetSelectionnable = null;
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Manipuler un objet graphique et le faire tomber

**Etape 3 – Manipuler l'objet sélectionné avec la souris lorsque le bouton gauche est enfoncé et faire tomber l'objet lorsque le bouton gauche est relâché.**

- **COMMENT FAIRE?** Voir avec les éléments suivants pour le tracking de la souris dans l'Environnement Virtuel
  - `Input.mousePosition.x` et `Input.mousePosition.y`
  - `Camera.main.ScreenToWorldPoint()`
  - `Camera.main.nearClipPlane`
  - Calcul de la vitesse de déplacement de la souris à partir **gain** \*  $(\text{pos}(t) - \text{pos}(t-1)) * \text{Delta}(t)$
  - `Delta(t)` se calcule à partir de **Time.deltaTime**, durée entre deux frames graphiques
- Pour faire tomber l'objet sélectionné, physicalisez tous les objets graphiques, le booléen **isKinematic** du **Component Rigidbody** des objets graphiques est à fixer à **true** au Run et à **false** pour l'objet qui doit chuter...
- ❑ **A FAIRE**
  - Rajoutez le **Component Rigidbody** et cochez le booléen **isKinematic** pour tous les objets graphiques
  - Complétez **SelectionObjets** et **GereSelection** en conséquence

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Manipuler un objet graphique et le faire tomber

```
public class GereSelection : MonoBehaviour
{
    public float gain = 0.15F; // Gain à mettre à jour si besoin dans le Game, pour
l'ajustement de la position de l'objet manipulé par rapport au curseur de la souris
    public GameObject objetSelectionne = null; // Pointe vers l'objet sélectionnable
    public GameObject objetSelectionnable = null; // Pointe vers l'objet sélectionné
    private GameObject[] tfils; // tableau contenant la liste des GameObject fils de
GrosObjet
    private int nfiles;
    private Vector3 mp_past; // Position 3d associé au curseur de la souris dans
l'environnement virtuel, à l'instant t-1
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Manipuler un objet graphique et le faire tomber

```
void Update()
{
    if (Input.GetMouseButtonDown(0) && (objetSelectionnable != null) && (objetSelectionne == null)) // Appuie sur le bouton gauche de la souris et un objet est selectionnable?
    {
        int indexSelectionne = objetGetIndex(objetSelectionnable.name);
        Debug.Log("*** " + objetSelectionnable.name + " est sélectionné");
        objetSelectionne = objetSelectionnable;
        tfils[indexSelectionne].GetComponent<SelectionObjet>().SetSelectionne(); // L'objet est sélectionné
        objetSelectionnable = null;
        mp_past = Camera.main.ScreenToWorldPoint(
            new Vector3(Input.mousePosition.x,
                Input.mousePosition.y,
                Camera.main.nearClipPlane));
    }
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Manipuler un objet graphique et le faire tomber

```
else if (Input.GetMouseButton(0) && (objetSelectionnable == null) && (objetSelectionne != null)) // Appuie sur le bouton gauche de la souris et un objet est sélectionné?
{
    int indexSelectionne = objetGetIndex(objetSelectionne.name);
    Vector3 mp = Camera.main.ScreenToWorldPoint( // Position 3d associé au curseur de la
souris dans l'environnement virtuel, à l'instant t
        new Vector3(Input.mousePosition.x,
                    Input.mousePosition.y,
                    Camera.main.nearClipPlane));
    Debug.Log("Durée entre deux frames graphiques (s): " + Time.deltaTime);
    Vector3 delta = (mp - mp_past) / (Time.deltaTime); // Vitesse de déplacement de
l'avatar porté par la souris dans le monde virtuel
    mp_past = mp;
    objetSelectionne.transform.position += gain * delta;
    Debug.Log("*** " + objetSelectionne.name + " est à la position (" + mp.x.ToString() +
", " + mp.y + ", " + mp.z + ")");}
```

---

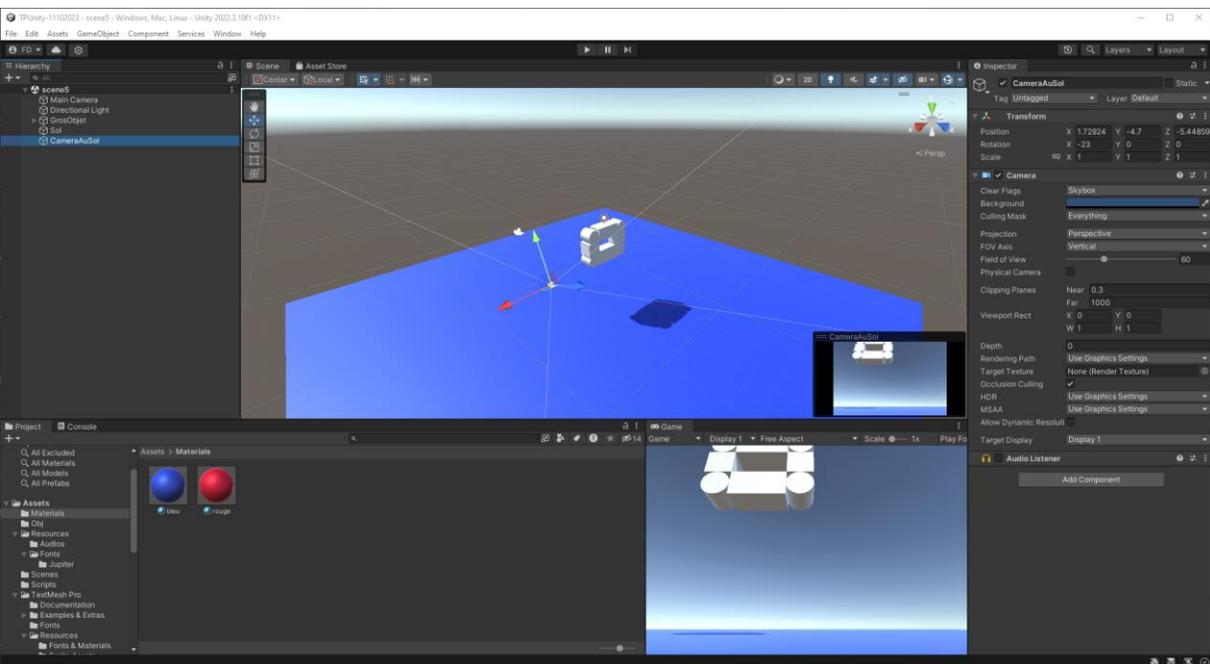
# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Manipuler un objet graphique et le faire tomber

```
else if (Input.GetMouseButtonUp(0) && (objetSelectionne != null)) // Relache le bouton
gauche de la souris et un objet est sélectionné?
{
    int indexSelectionne = objetGetIndex(objetSelectionne.name);
    Debug.Log("*** " + objetSelectionne.name + " est relaché.");
    objetSelectionne.transform.GetComponent<Rigidbody>().isKinematic = false; // L'objet
est à nouveau physicalisé et il chute
    objetSelectionne = null;
}
```

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Gérer la collision avec Sol et l'alternance des deux cameras



Etape 4 – Ajout d'un Sol et d'une caméra au Sol, gestion de la collision avec Sol et de l'alternance des deux cameras

- Copiez *scene4* en *scene5*
- Créez dans *scene5* un 3D objet de type **Plane**, nommé *Sol*, de 4mX4m, centré sur 0 sur le plan (x,z), à y=-5 mètres et appliquez lui le **Material bleu**
- Créez une caméra que vous nommerez **CameraAuSol** avec les coordonnées suivantes: **position (1.7, -4.7, -5.4)**, **orientation (-23,0,0)**, puis décochez **AudioListener**

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Gérer la collision avec Sol et l'alternance des deux cameras

### Règle d'alternance des caméras

- Au Run, **CameraAuSol** est désactivée (.enabled=false), donc **MainCamera** est activée
- Lorsqu'un objet graphique tombe vers **Sol**, **CameraAuSol** est activée (.enabled=true), et **MainCamera** est désactivée
- Lorsqu'un objet graphique intersecte **Sol**, **CameraAuSol** est désactivée (.enabled=false), donc **MainCamera** est activée

### Règle d'arrêt de la chute d'un objet graphique

- Un objet graphique arrête de chuter (.isKinematic devient **true**) lorsqu'il intersecte Sol

### A FAIRE

- Modifier **GereSelection** et **SelectionObjets** en conséquence

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Gérer la collision avec Sol et l'alternance des deux cameras

```
public class SelectionObjet : MonoBehaviour
{
    private Color oldColor;
    private GereSelection gs;
    public Camera cameraAuSol;
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Gérer la collision avec Sol et l'alternance des deux cameras

```
// Dans SelectionObjet
// Gestion de la collision de l'objet portant le script avec l'objet de name 'Sol'
private void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.name == "Sol")
    {
        transform.GetComponent<Rigidbody>().isKinematic = true; // On arrête la
        physicalisation de l'objet
        GetComponent<Renderer>().material.color = oldColor;
        cameraAuSol.enabled = false; // La caméra au sol est désactivée après la chute
    }
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Gérer la collision avec Sol et l'alternance des deux cameras

```
// Dans GereSelection, Update()
else if (Input.GetMouseButtonUp(0) && (objetSelectionne != null)) // Relache le bouton
gauche de la souris et un objet est sélectionné?
{
    Debug.Log("*** " + objetSelectionne.name + " est relaché.");
    objetSelectionne.transform.GetComponent<Rigidbody>().isKinematic = false; // L'objet
graphique est à nouveau physicalisé et il chute
    objetSelectionne = null;
    cameraAuSol.enabled = true; // La caméra au sol est activée pendant la chute
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Gestion d'une machine à états associée aux objets graphiques

### Etape 5 – Création d'une machine à états pour les objets graphiques

- Chaque objet graphique est associé à un état qui se trouve dans la liste EtatsObjets, parmi:
  - **Idle**, **Sélectionnable**, **Sélectionné**, **Manipulé**, **EnChute**, **AuSol**
- L'état initial de chaque objet graphique est Idle
- Transitions entre états pour un objet graphique**
  - **Idle** vers **Sélectionnable** lorsque le curseur de la souris intersecte l'objet graphique
  - **Sélectionnable** vers **Idle** lorsque le curseur de la souris n'intersecte plus l'objet graphique qui était dans l'état **Sélectionnable**
  - **Sélectionnable** vers **Sélectionné** lorsque le bouton gauche de la souris est appuyé pour la première fois
  - **Sélectionné** vers **Manipulé** lorsque le bouton gauche de la souris reste
  - **Manipulé** vers **EnChute** lorsque le bouton gauche de la souris est relâché
  - **EnChute** vers **AuSol** lorsque l'objet graphique intersecte **Sol**
- A FAIRE**
  - Complétez **SelectionObjets** en conséquence

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Gestion d'une machine à états associée aux objets graphiques

```
// dans l'état Selectionnable
public void SetSelectionnable()
{
    etatObjets = EtatsObjets.Sélectionnable;
}
// dans l'état Selectionne
public void SetSelectionne()
{
    etatObjets = EtatsObjets.Sélectionné;
}
// dans l'état EnChute
public void SetEnChute()
{
    etatObjets = EtatsObjets.EnChute;
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Gestion d'une machine à états associée aux objets graphiques

```
public class SelectionObjet : MonoBehaviour
{
    private Color oldColor;
    private GereSelection gs;
    public enum EtatsObjets { Idle, Sélectionnable, Sélectionné, Manipulé, EnChute, TombéAuSol };
    public EtatsObjets etatObjets;
    public Camera cameraAuSol;

    // Met à jour etatObjet
    // dans l'état Idle
    public void SetIdle()
    {
        etatObjets = EtatsObjets.Idle;
    }
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Gestion d'une machine à états associée aux objets graphiques

```
// dans l'état Selectionnable
public void SetSelectionnable()
{
    etatObjets = EtatsObjets.Sélectionnable;
}
// dans l'état Selectionne
public void SetSelectionne()
{
    etatObjets = EtatsObjets.Sélectionné;
}
// dans l'état EnChute
public void SetEnChute()
{
    etatObjets = EtatsObjets.EnChute;
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Gestion d'une machine à états associée aux objets graphiques

```
// dans l'état Manipule
public void SetManipule()
{
    etatObjets = EtatsObjets.Manipulé;
}
// dans l'état TombeAuSol
public void SetTombeAuSol()
{
    etatObjets = EtatsObjets.TombéAuSol;
}
void Start()
{
    gs = this.transform.parent.GetComponent<GereSelection>();
    SetIdle(); // Etat initial associé à l'objet graphique
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Gestion d'une machine à états associée aux objets graphiques

```
// Post: l'objet courant est dans l'état Sélectionnable et sa la couleur devient rouge
//
private void OnMouseEnter()
{
    if (etatObjets == EtatsObjets.Idle)
    {
        oldColor = GetComponent<Renderer>().material.color;
        GetComponent<Renderer>().material.color = Color.red;
        gs.objetSélectionnable = transform.gameObject;
        Debug.Log("[ " + transform.name + " SELECTIONNABLE");
        etatObjets = EtatsObjets.Sélectionnable; // L'état associé est
    }
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Gestion d'une machine à états associée aux objets graphiques

```
// L'objet qui porte le script reprend sa couleur initiale et l'état associé devient Idle
private void OnMouseExit()
{
    if (etatObjets == EtatsObjets.Sélectionnable)
    {
        GetComponent<Renderer>().material.color = oldColor;
        gs.objetSelectionnable = null;
        etatObjets = EtatsObjets.Idle;
    }
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Gestion d'une machine à états associée aux objets graphiques

```
// Post: l'objet courant est dans l'état Sélectionnable et sa la couleur devient rouge
//
private void OnMouseEnter()
{
    if (etatObjets == EtatsObjets.Idle)
    {
        oldColor = GetComponent<Renderer>().material.color; // On sauve la couleur
courante de l'objet porté qui porte le script
        GetComponent<Renderer>().material.color = Color.red; // La couleur de l'objet
qui porte le script devient rouge
        gs.objetSelectionnable = transform.gameObject; // Met l'information de l'objet
graphique sélectionnable dans GereSelection, porté par GrosObjet
        Debug.Log("[ " + transform.name + " SELECTIONNABLE]");
        etatObjets = EtatsObjets.Sélectionnable; // L'état associé est Selectionnable
    }
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Gestion d'une machine à états associée aux objets graphiques

```
// L'objet qui porte le script reprend sa couleur initiale et l'état associé devient Idle
private void OnMouseExit()
{
    if (etatObjets == EtatsObjets.Sélectionnable)
    {
        GetComponent<Renderer>().material.color = oldColor;
        gs.objetSelectionnable = null;
        etatObjets = EtatsObjets.Idle;
    }
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Gestion de la partie

### Etape 6 – Gestion de la partie - Etat de terminaison

- Créez un booleen **partieTerminee**, placé dans **GereSelection**
- **partieTerminee** vaut **true** ssi les 8 objets graphiques sont associés à l'état **TombéAuSol**

#### A FAIRE

- Complétez **GereSelection** en conséquence

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

## Gestion de la partie

```
public class GereSelection : MonoBehaviour
{
    public Camera cameraAuSol;
    public bool partieTerminee; // Vaut true ssi tous les objets fils de GrosObjet sont
    dans l'état TombéAuSol
    public float gain = 0.15F;
    public GameObject objetSelectionne = null;
    public GameObject objetSelectionnable = null;
    private GameObject[] tfils;
    private int nfils;
    private Vector3 mp_past;

    void Start()
    {
        partieTerminee = false;
    }
    ...
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

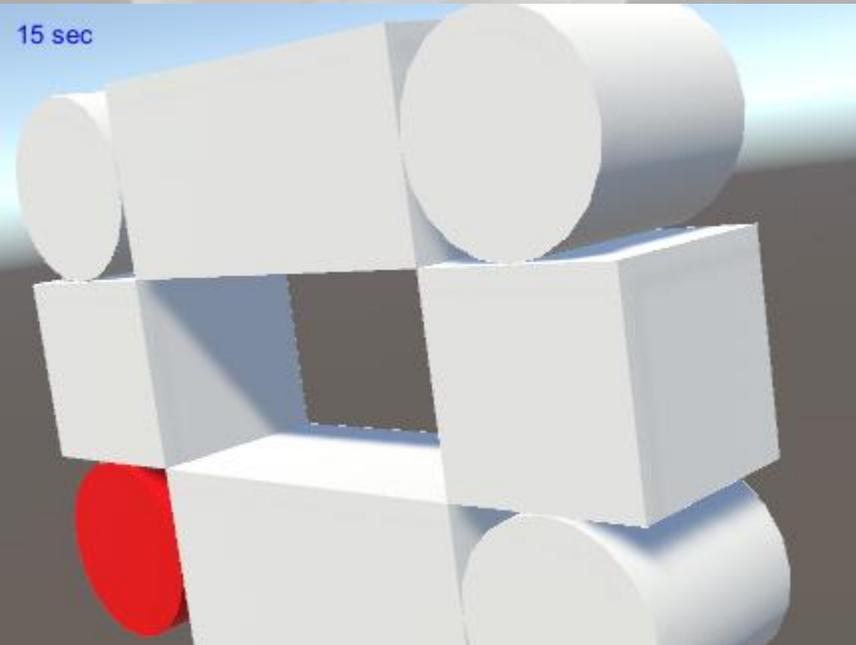
## Gestion de la partie

```
// Pre: les éléments de tfils[] sont initialisés et portent le Script SelectionObjet
// Post: true si tous les objets de tfils[] sont associés à l'état TombéAuSol, false
sinon
public bool PartieTerminee()
{
    bool term = true;
    for (int i = 0; i < nfils; i++)
        term = term && (tfils[i].GetComponent<SelectionObjet>().etatObjets ==
SelectionObjet.EtatsObjets.TombéAuSol);
    return term;
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Affichage des informations du jeu grâce à `void OnGui(){}`



- ❑ **Etape 7 - Afficher le temps depuis le début du Run, en haut à gauche de l'écran, en bleu.**

Nous allons utiliser la fonction prédéfinie `void OnGUI(){}` dans *GereSelection*, dans laquelle nous calculerons le temps depuis le début du Run et nous l'afficherons grâce à un `GUI.Label(new Rect(10, 5, 150, 20), compteur_temps_str);`

Le temps est donné par `Time.time`

La couleur du texte est donnée par `GUI.contentColor = Color.blue`

- ❑ **A FAIRE**

- Complétez *GereSelection* en conséquence
-

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Affichage des informations du jeu grâce à void OnGui(){}

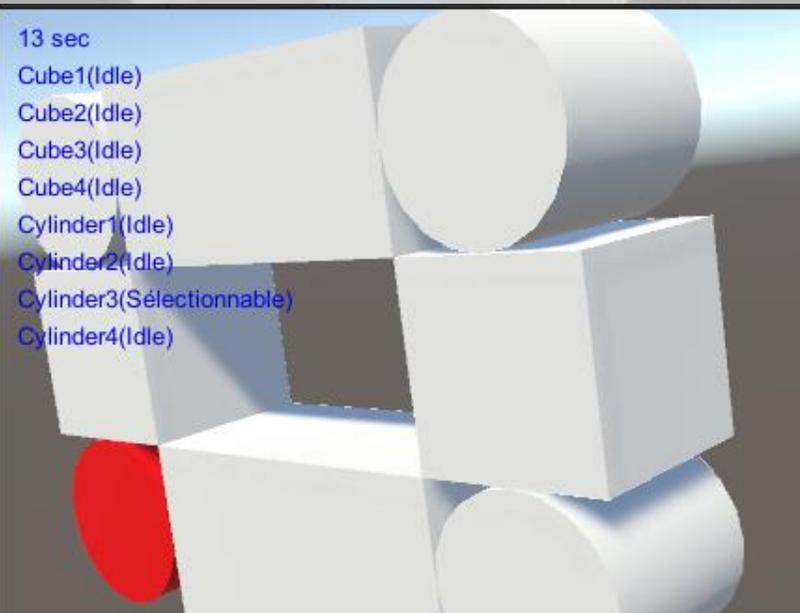
```
public class GereSelection : MonoBehaviour
{
...
    private string compteur_temps; // Temps depuis le début du Run, formaté en string,
se terminant par ' sec'
}

// Implémente ce qui noté sur l'écran du Game
// Exécuté à chaque frame graphique
void OnGUI() // Scène 5
{
    GUI.contentColor = Color.blue;
    compteur_temps = Time.time.ToString("0.") + " sec";
    GUI.Label(new Rect(10, 5, 150, 20), compteur_temps);
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Affichage des informations du jeu grâce à void OnGui(){}



- Etape 8 – Afficher le nom des objets, avec entre parenthèses leurs états respectifs, en couleur bleue
- A FAIRE
  - Complétez *GereSelection* OnGui(){} en conséquence

---

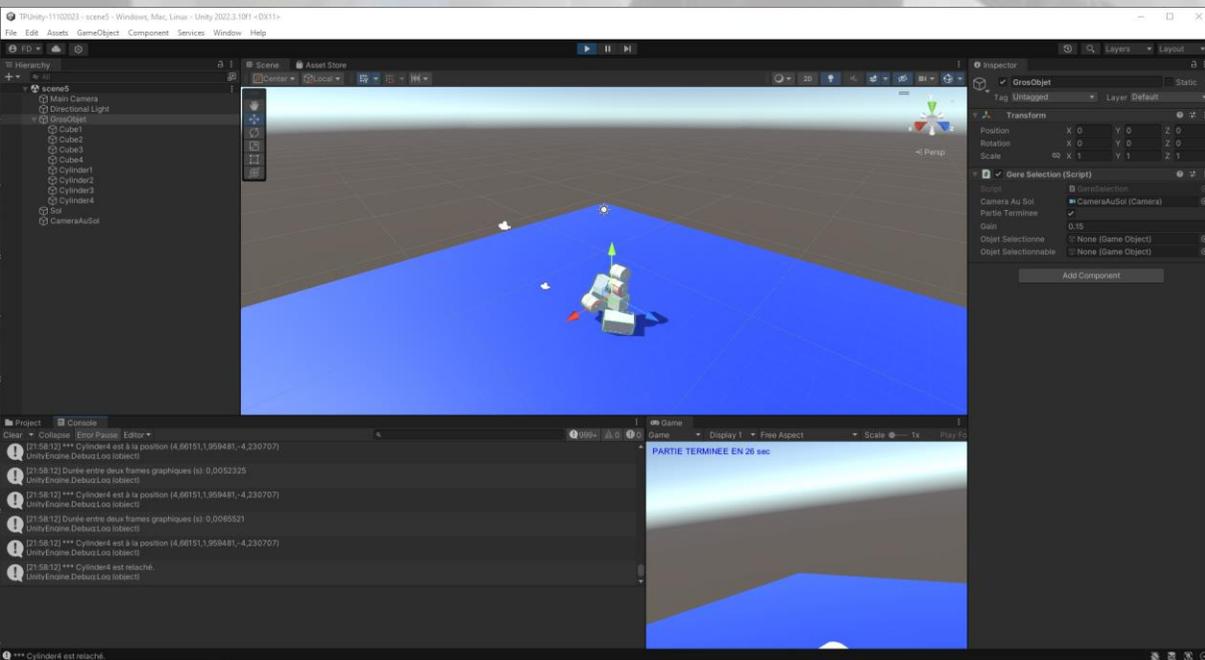
# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Affichage des informations du jeu grâce à void OnGui(){}

```
// Implémente ce qui noté sur l'écran du Game
// Exécuté à chaque frame graphique
void OnGUI() // Scène 5
{
    GUI.contentColor = Color.blue;
    compteur_temps = Time.time.ToString("0.") + " sec";
    GUI.Label(new Rect(10, 5, 150, 20), compteur_temps);
    for (int i = 0; i < nfils; i++)
    {
        GUI.Label(new Rect(10, 5 + (i + 1) * 20, 150, 20), tfils[i].name + "(" +
tfils[i].GetComponent<SelectionObjet>().etatObjets + ")");
    }
}
```

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Affichage des informations du jeu grâce à void OnGui(){}



- Etape 9 – Gestion de la fin de partie
- A FAIRE
  - Complétez *GereSelection* Update(){} et OnGui(){} en conséquence, en fonction de *partieTerminee*

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Affichage des informations du jeu grâce à void OnGui(){}

```
void Update()
{
    if (partieTerminee == false)
    {
        if (Input.GetMouseButtonDown(0) && (objetSelectionnable != null) &&
(objetSelectionne == null)) // Appuie sur le bouton gauche de la souris et un objet
est selectionnable?
        {
...
        }
...
    }
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

Affichage des informations du jeu grâce à void OnGui(){}

```
void OnGUI() // Scène 5
{
    GUI.contentColor = Color.blue;
    if (PartieTerminee() == false)
    {
        compteur_temps = Time.time.ToString("0.") + " sec";
        GUI.Label(new Rect(10, 5, 150, 20), compteur_temps);
        for (int i = 0; i < nfils; i++)
        {
            GUI.Label(new Rect(10, 5 + (i + 1) * 20, 150, 20), tfils[i].name + "(" +
tfils[i].GetComponent<SelectionObjet>().etatObjets + ")");
        }
    }
    else
    {
        GUI.Label(new Rect(10, 5, 250, 20), "PARTIE TERMINEE EN " + compteur_temps);
        partieTerminee = true;
    }
}
```

---

# TP UNITY N°2, MERCREDI 11 OCTOBRE 2023

53 sec  
Cube1(TombéAuSol)  
Cube2(Idle)  
Cube3(Idle)  
Cube4(Manipulé)  
Cylinder1(TombéAuSol)  
Cylinder2(Idle)  
Cylinder3(Idle)  
Cylinder4(Idle)



Fin du TP n°2