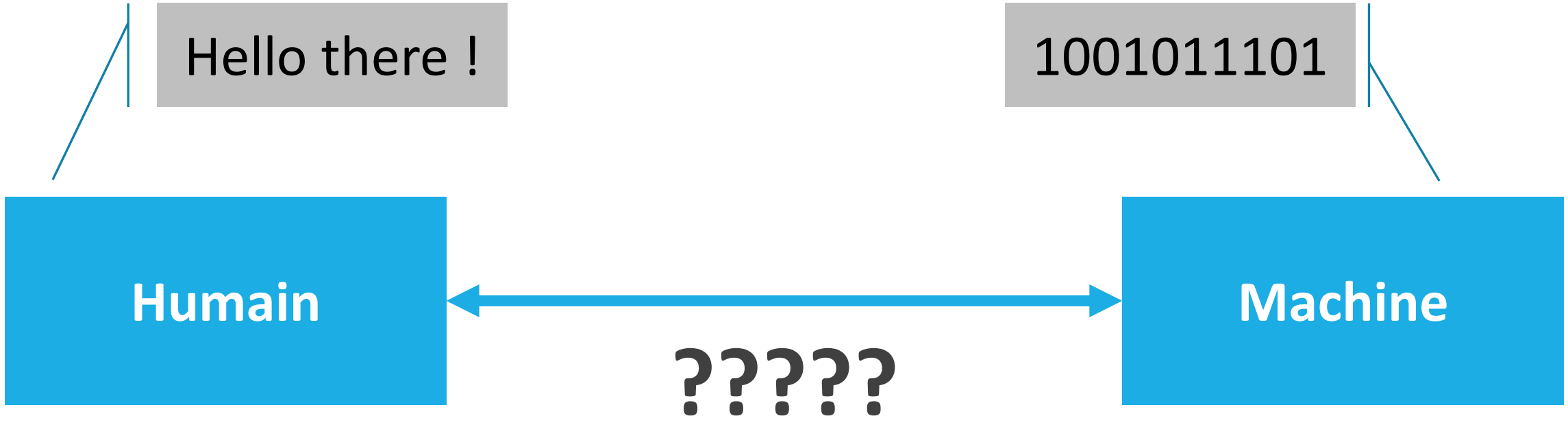
Créer des U.I. dans Unity

C’est quoi « U.I » ?

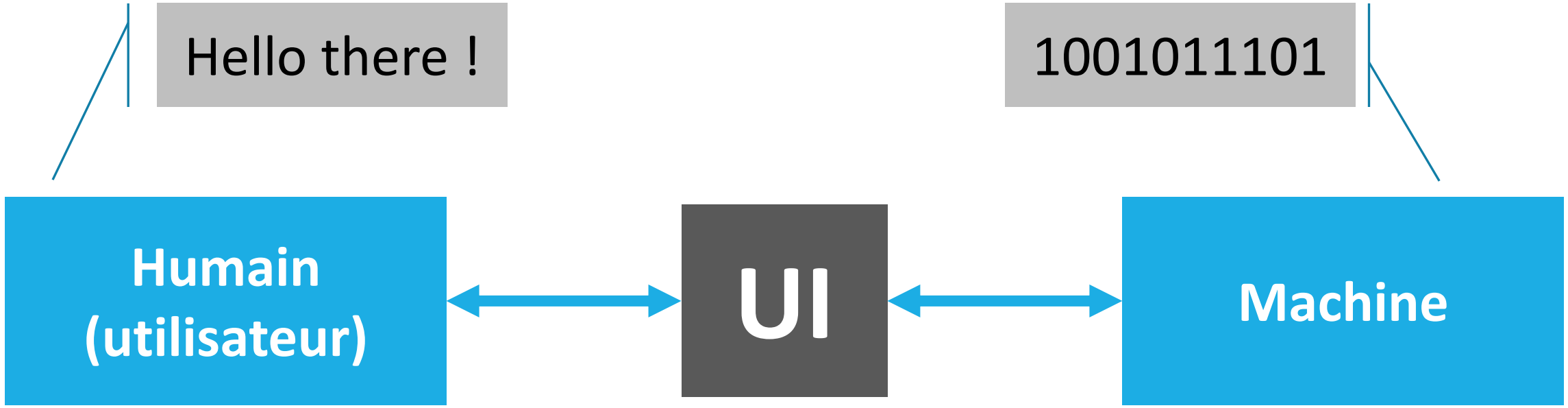
C’est quoi « U.I. » ?

User **I**nterface

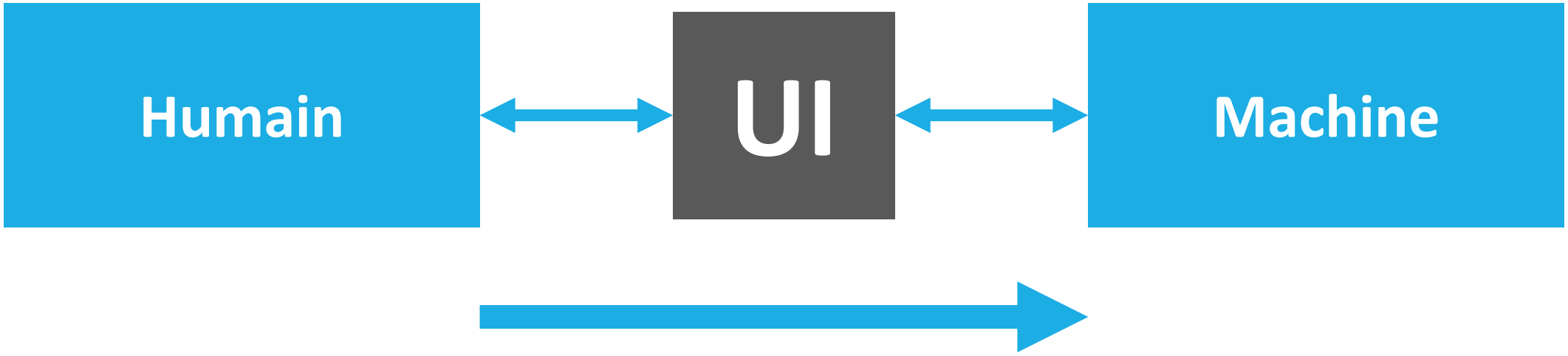
C’est quoi « U.I. » ?



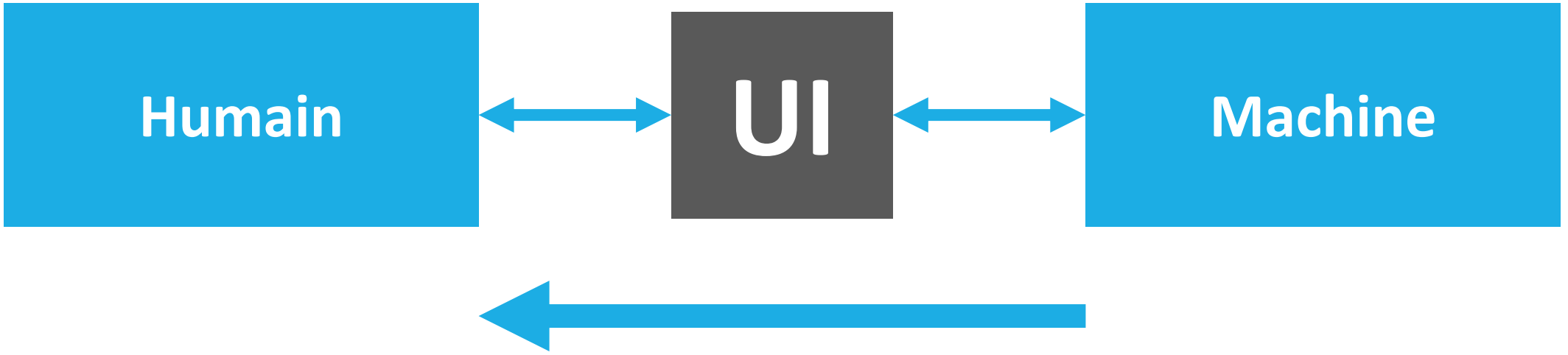
C’est quoi « U.I. » ?



C’est quoi « U.I. » ?

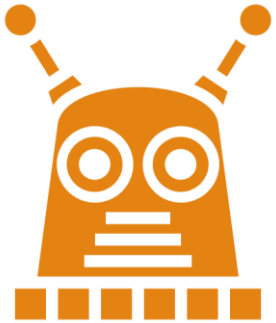


C’est quoi « U.I. » ?



C’est important ?
Pourquoi ?

C’est important ? Pourquoi ?



* Je mange des humains

Créer une UI dans Unity

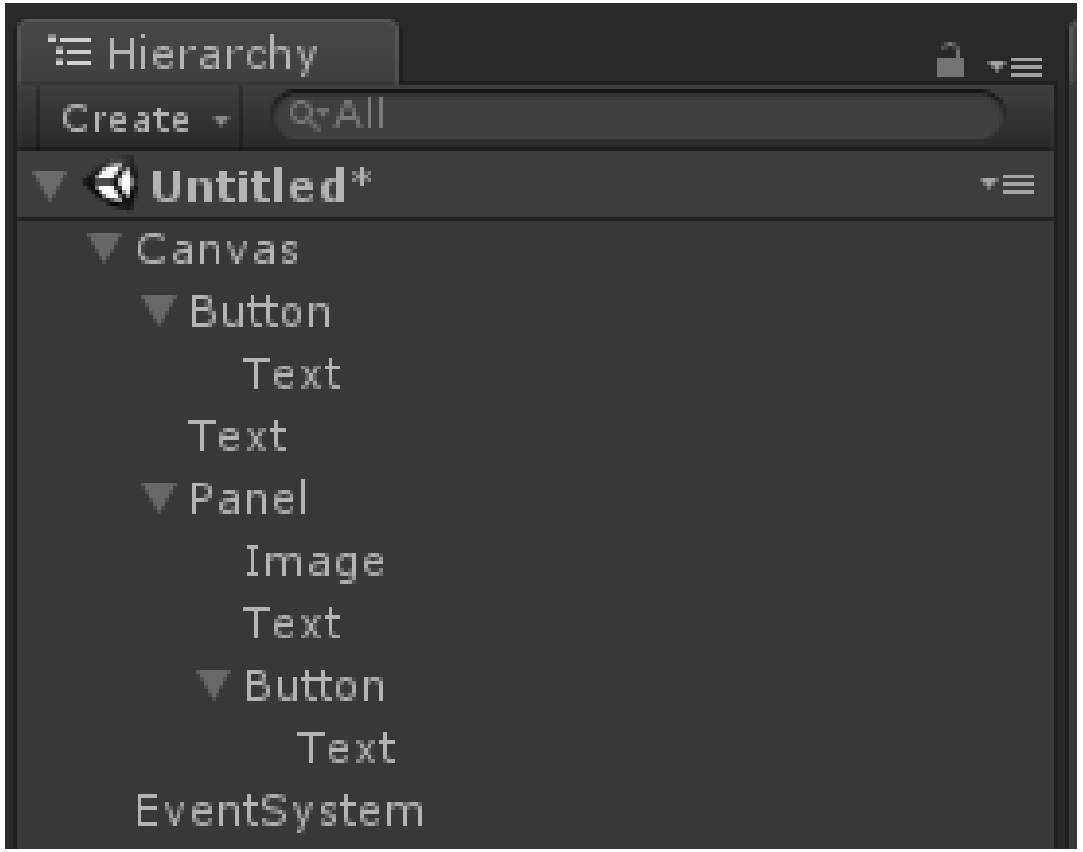
Le composant Canvas

C’est un espace de dessin pour les éléments de « UI »

Utilisation du Canvas

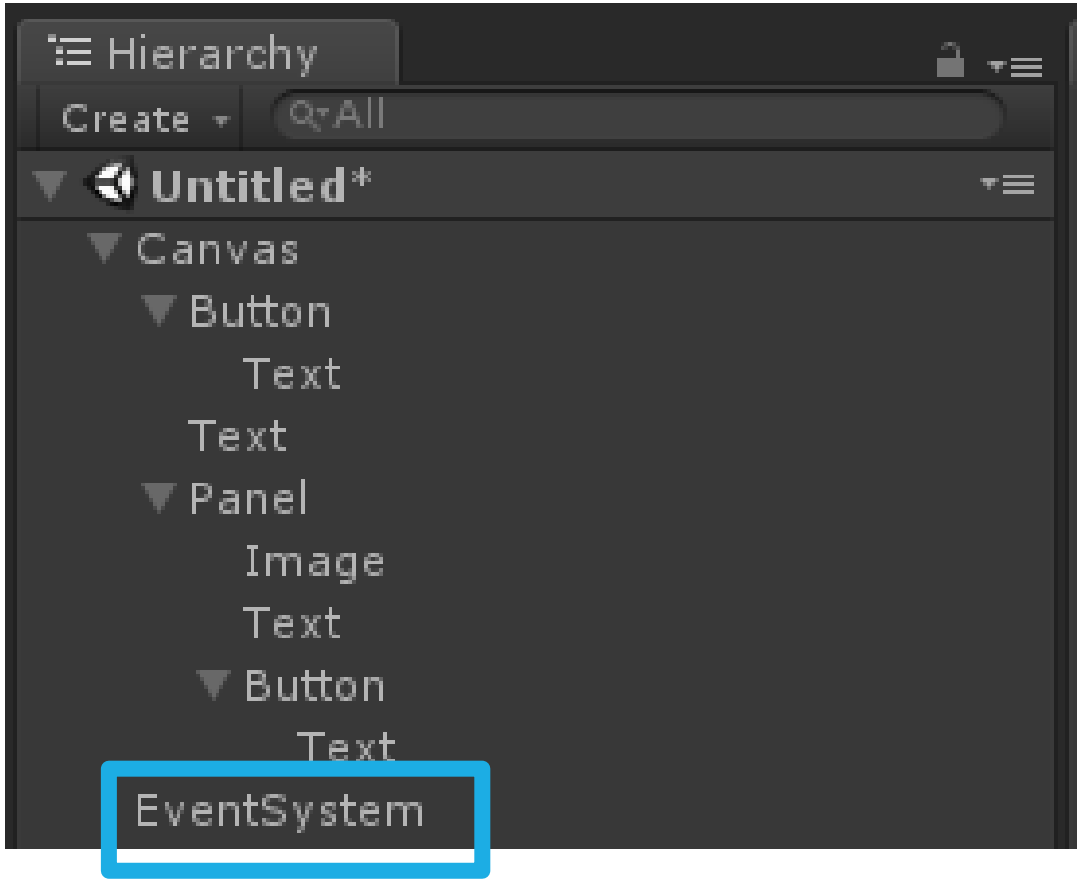
Vous **DEVEZ** utiliser le système de parenté

Les éléments de UI **DOIVENT** être enfants d'un Canvas



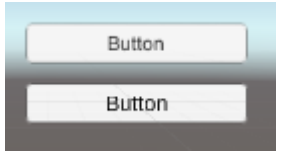
Interagir avec les éléments de UI

Un **EventSystem** est nécessaire pour interagir avec les éléments d'un Canvas (comme des boutons)

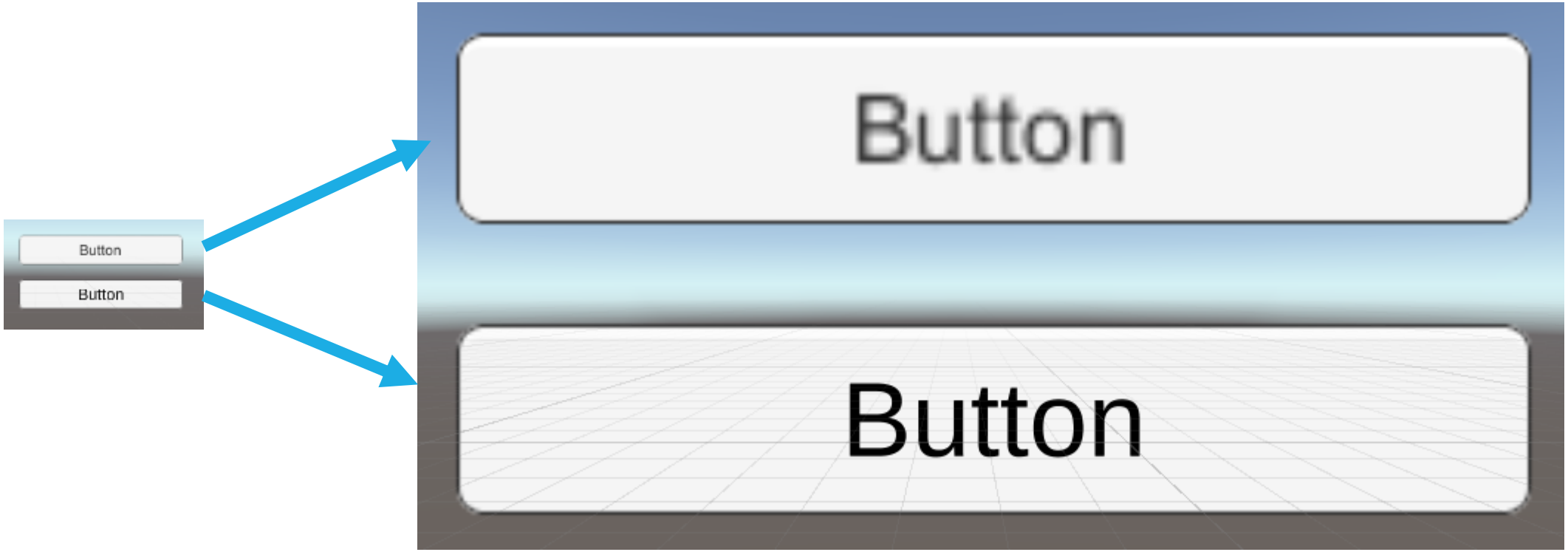


Textes avancés

TextMeshPro FTW



TextMeshPro FTW



TextMeshPro FTW

Composant Text par défaut

Composant TextMeshPro

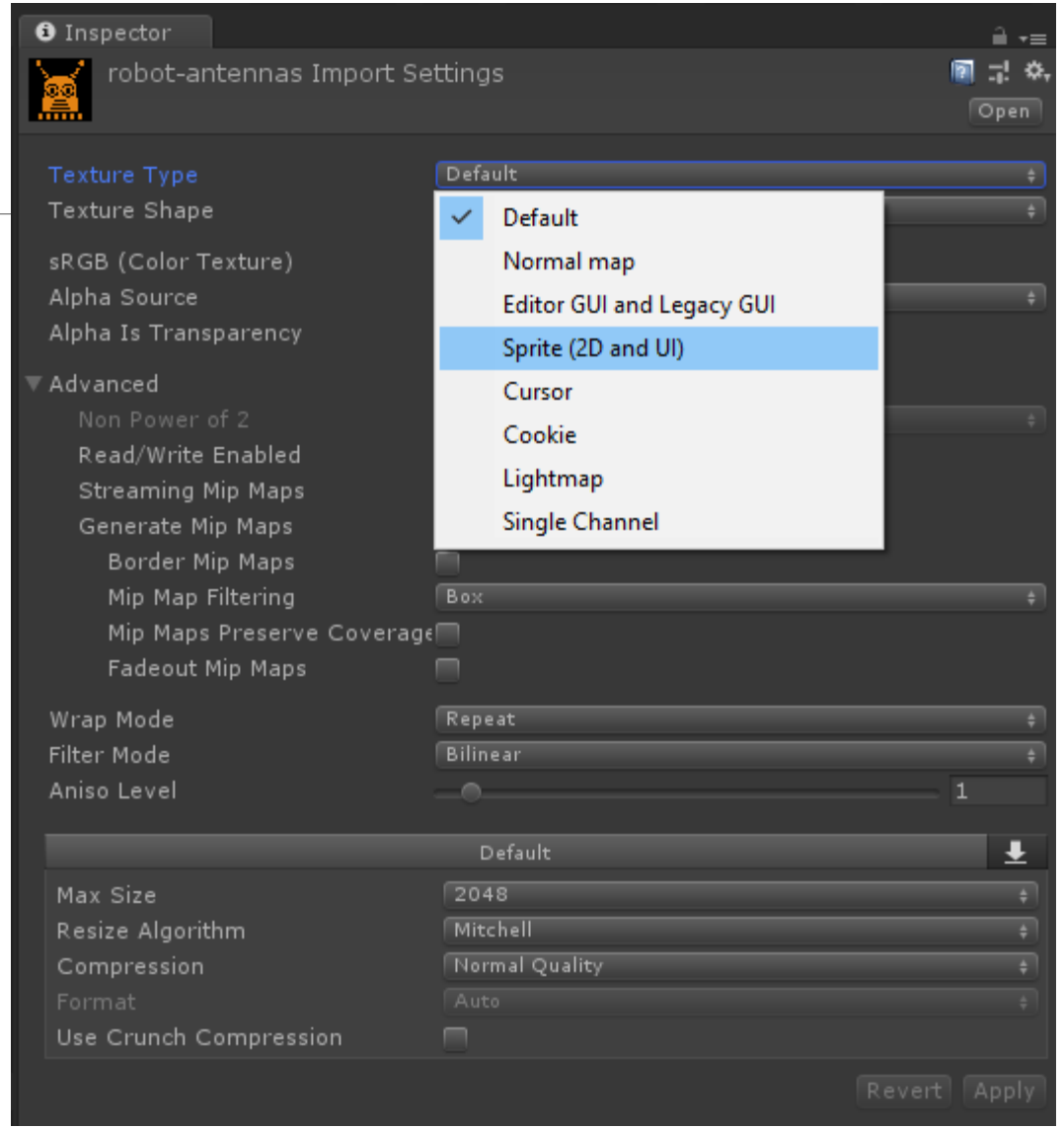
Personnaliser sa UI

Personnaliser sa UI

- Utilisez vos propres images / sprites
- Créez vos propres composants de UI en faisant de la composition de composants existants
- Ecrivez vos propres composants de UI

Format des images dans un Canvas

- Il faut changer le mode d'import
- Texture Type = Sprite
- Ne pas oublier d'appliquer les modifications



Créer des interfaces complexes

Créer des interfaces complexes

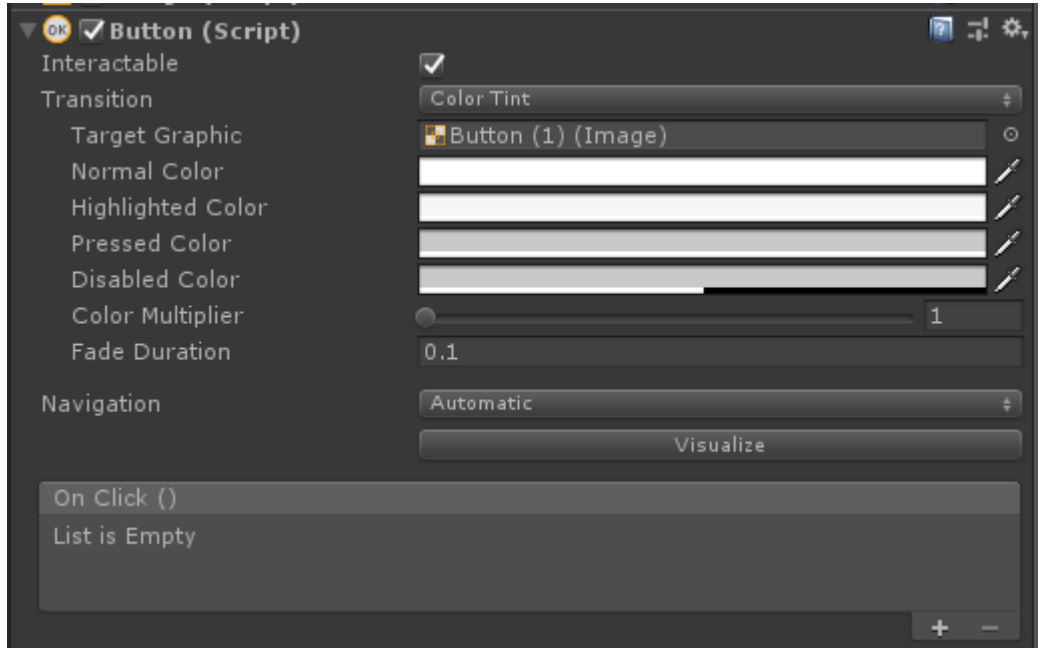
Un Panel est un composant de type « conteneur » avec une image de fond

Créer des interfaces complexes

Désactiver un GameObject dans la hiérarchie va masquer les éléments de UI attachés ainsi que tous ses enfants

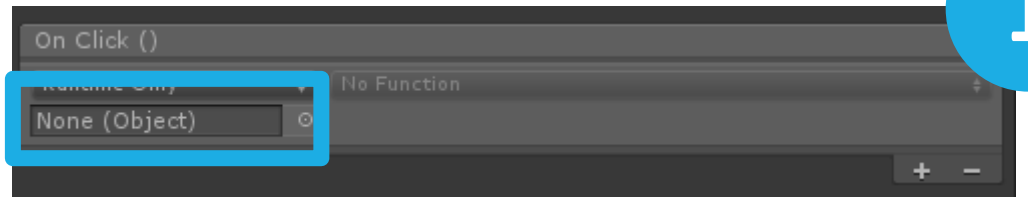
Afficher et masquer des éléments de UI sans code

Il est possible d'ajouter des actions sur un bouton

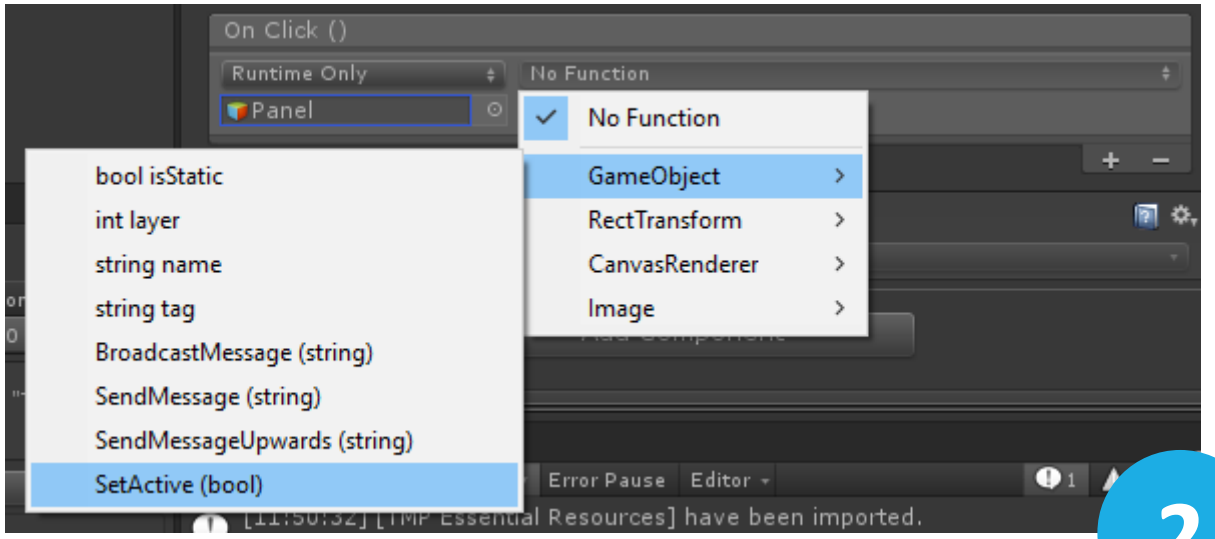


Afficher et masquer des éléments de UI sans code

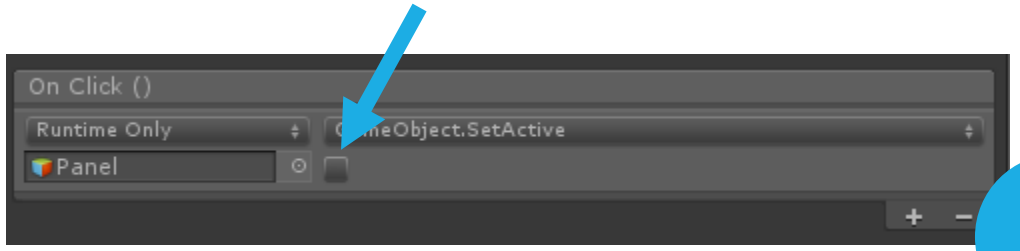
1. Selectionnez le GameObject cible
2. Sélectionnez la function / propriété à appeler / changer
3. Paramétrer les valeurs souhaitées



1



2



3

UI Avancée

Quelle est la différence entre ces supports ?

**Pas de guégerre PC / Console svp. On sait tous que les PC sont meilleurs. Merci.*

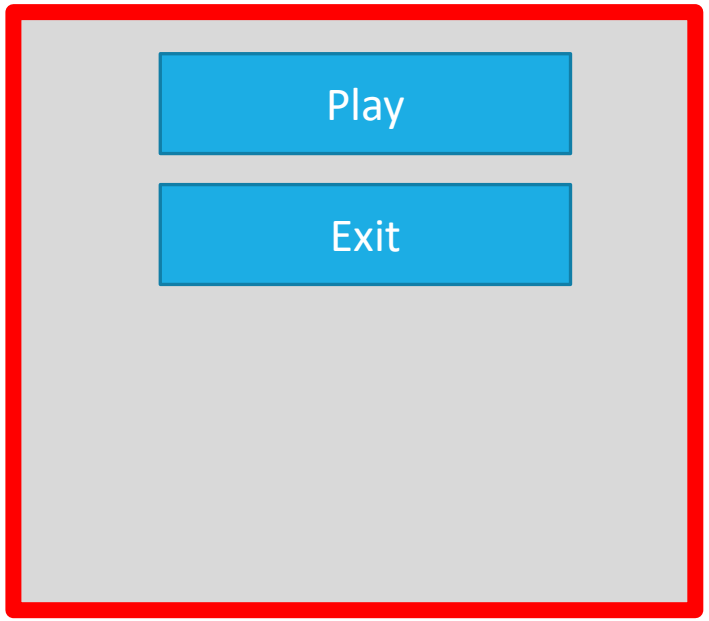
Le problème ?

LA TAILLE D'ECRAN

Il est communément admis que « c'set pas la taille qui compte », mais dans le cas d'un écran… si, c'est le cas 👀

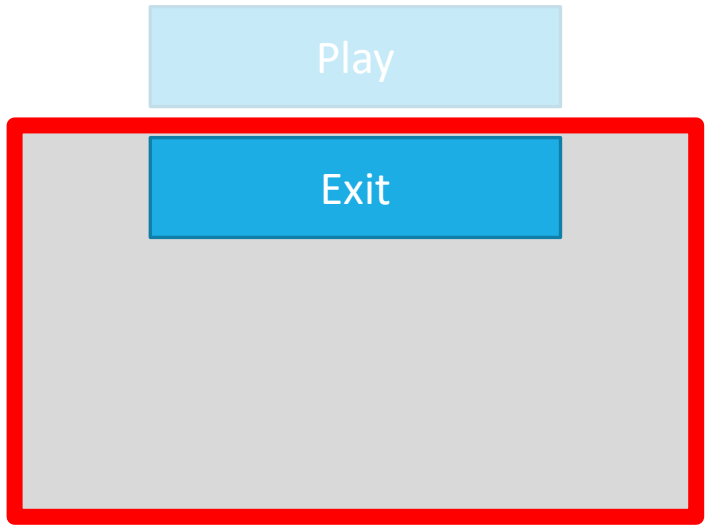
Pourquoi c’est un problème ?

Pourquoi c’est un problème ?



La zone rouge, c’est ce qui est vu sur l’écran du joueur

Pourquoi c’est un problème ?

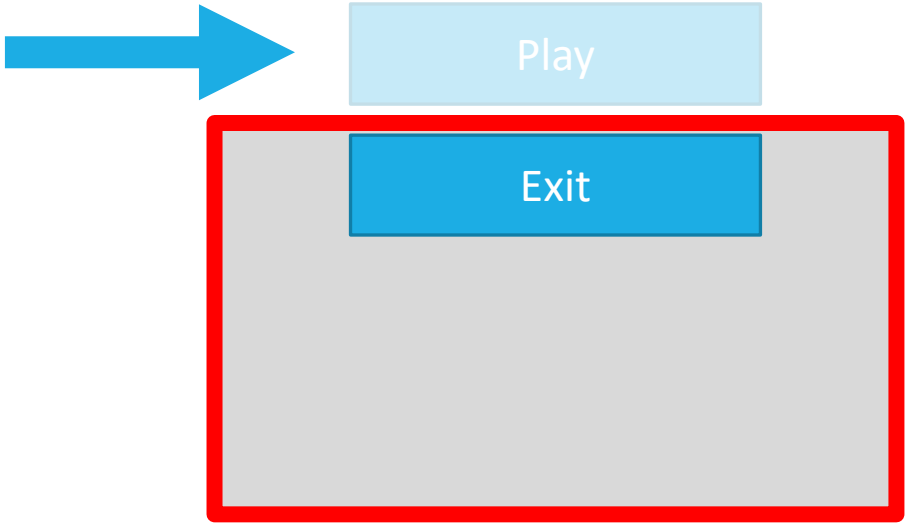


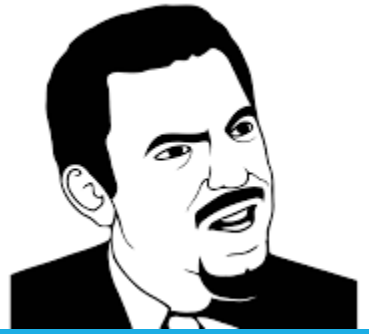
La zone rouge, c’est ce qui est vu sur l’écran du joueur

Pourquoi c’est un problème ?

Comment est-ce qu’on

clique sur ce bouton hein ?





L’exemple des jeux sur console de salon

L’exemple des jeux sur console de salon

L’exemple des jeux sur console de salon

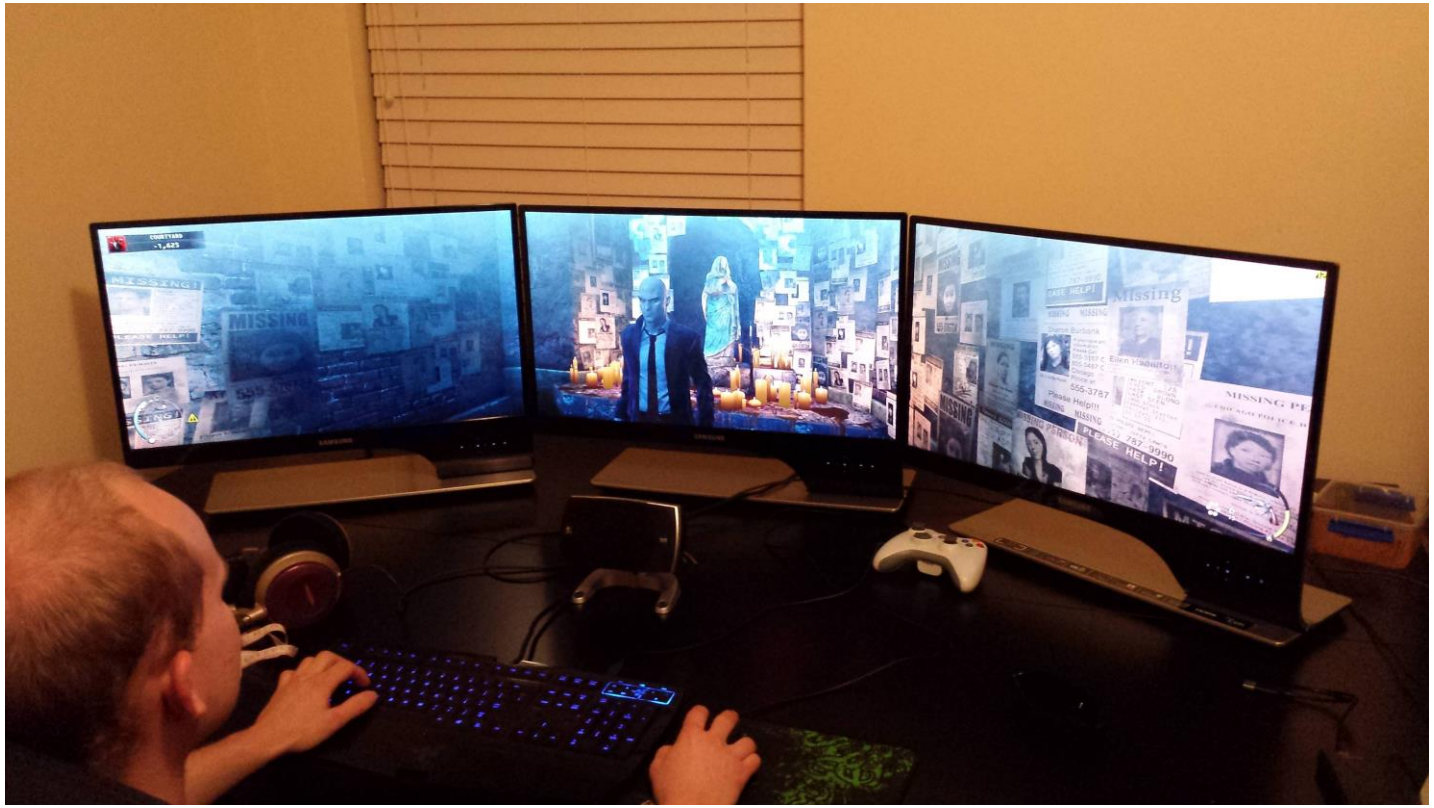
Zone de « Danger »

Zone « Safe »

Et ça, on en parle ?

5760x1080

#PCMasterRace



Comment gérer tout ça dans Unity ?

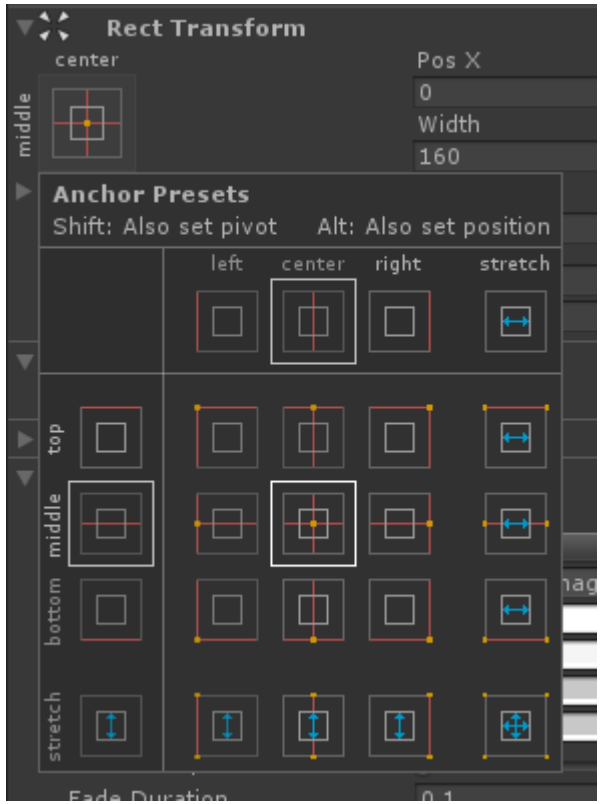
Ancres

+

Canvas Scaler

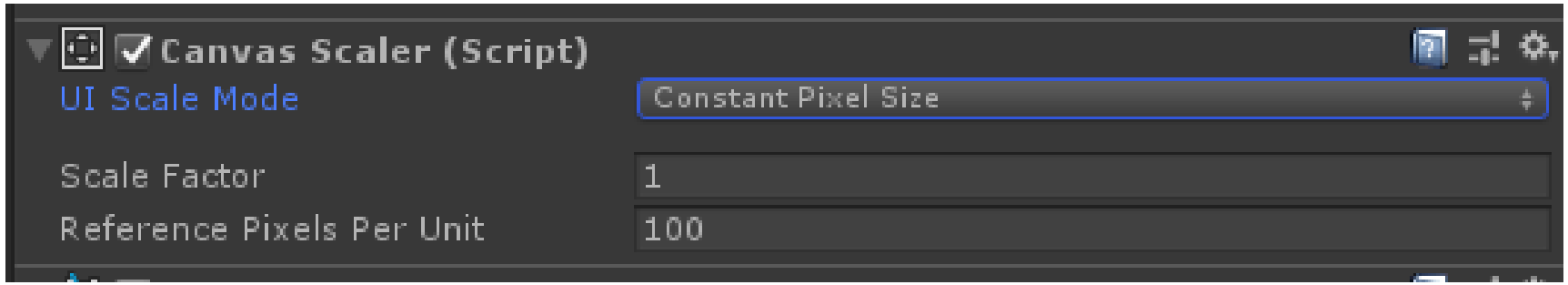
Les Ancres

- Avec les ancres, vous pouvez « ancrer » ou « attacher » un bord d'un composant UI à un bord de l'écran
- Les positions et coordonnées du composant de UI seront alors recalculées en fonction de la taille de l'écran*



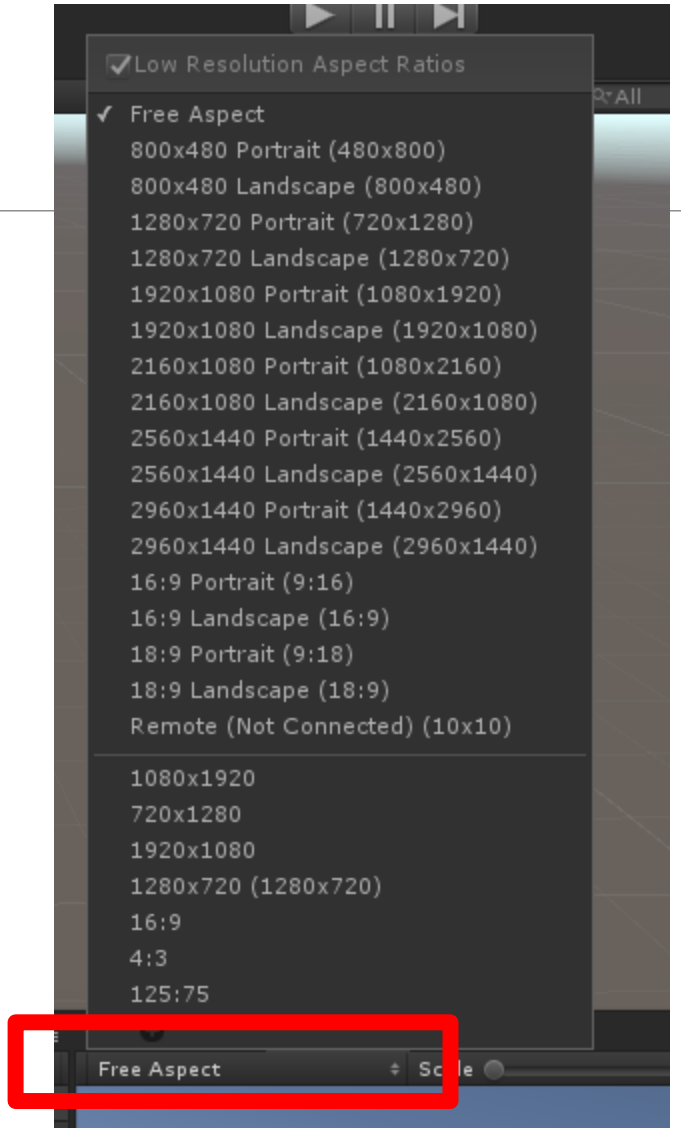
** Ou du paramétrage spécifique que vous aurez mis en place*

Le Canvas Scaler



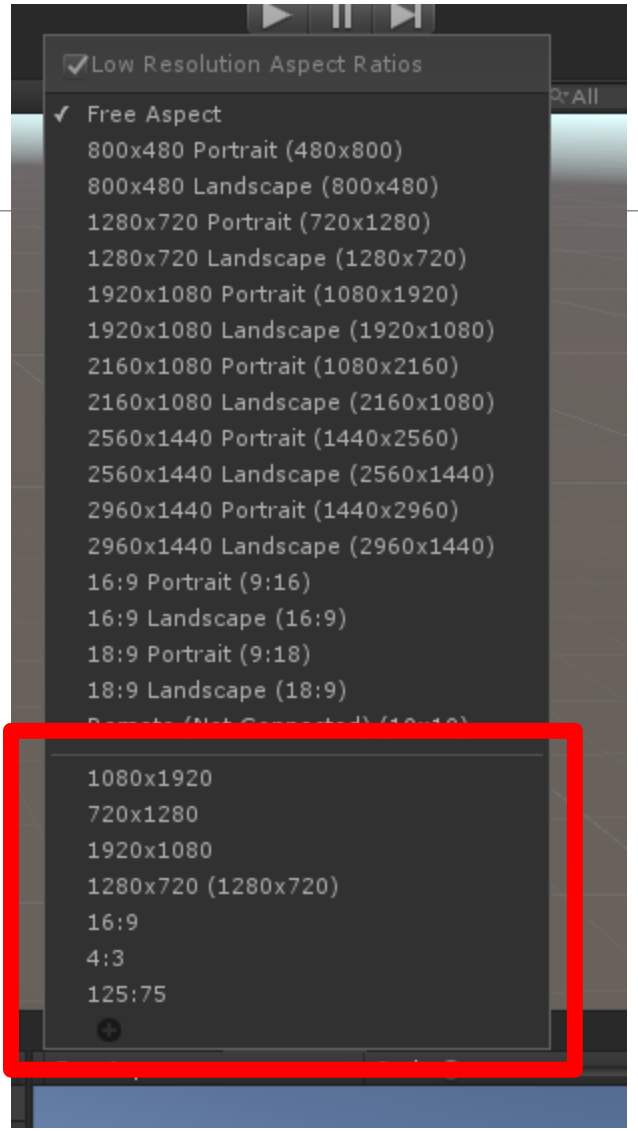
Comment tester ça ?

Forcez la résolution de la fenêtre Game à celle qui vous intéresse



Comment tester ça ?

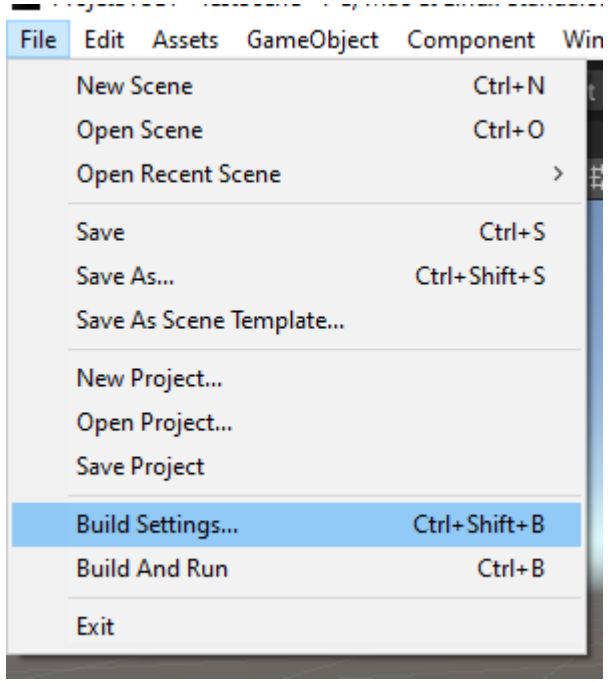
Vous pouvez enregistrer des résolutions custom si besoin

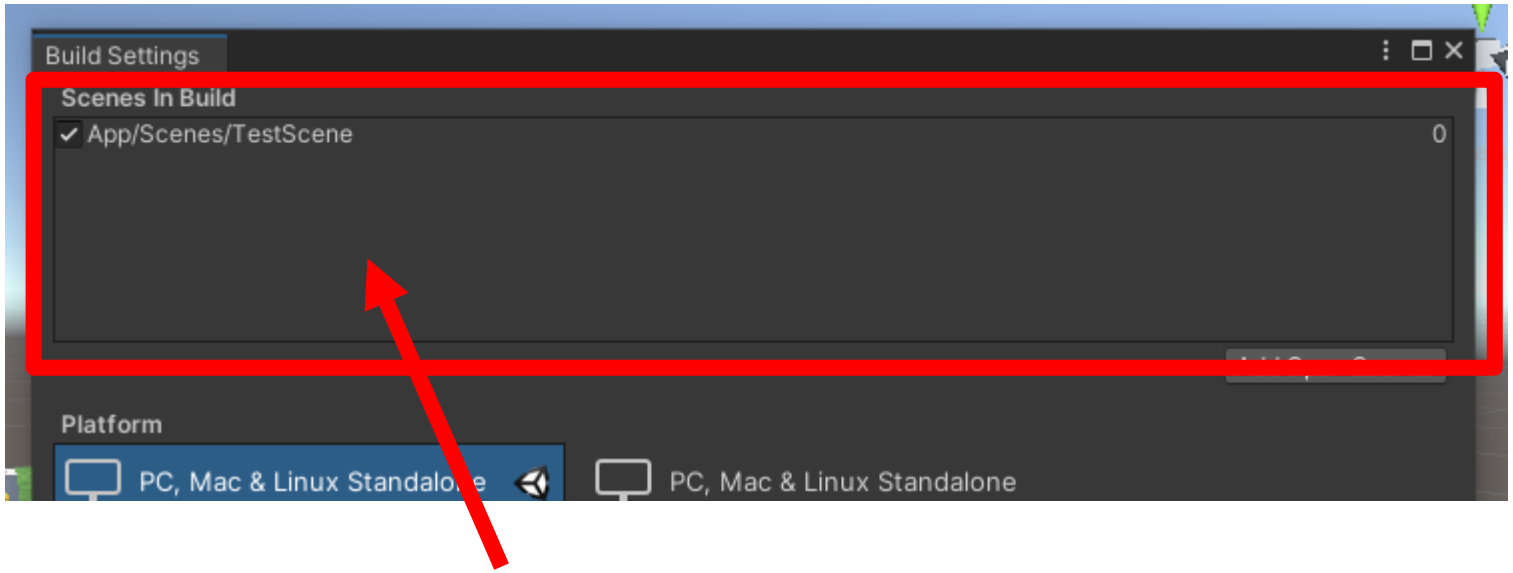


Gestion multi-scène

Changer de scène

Prérequis : déclarer les scènes dans la fenêtre de Build





Ajoutez **TOUTES** les scènes que vous comptez utiliser dans le projet

Changer de scène

- Unity met à disposition une classe pour gérer les chargements de scènes : le SceneManager

- Pour l'utiliser dans un script, il faut importer ce namespace :

```
using UnityEngine.SceneManagement;
```

- Ensuite, pour charger une scène, le code est très simple :

```
SceneManager.LoadScene("LeNomDeMaScene");
```

Changer de scène

```
SceneManager.LoadScene("LeNomDeMaScene");
```

Ce qui se passe :

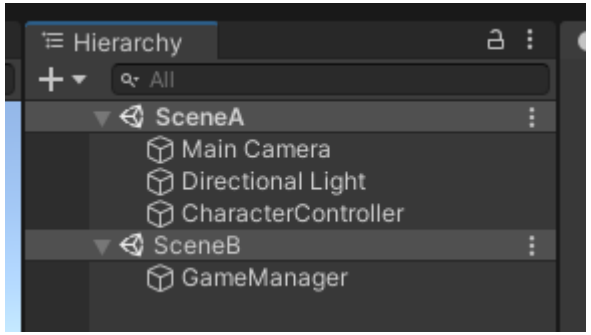
1. Déchargement de la scène courante de la mémoire
2. Chargement de la nouvelle scène en mémoire
3. … et c’est tout, en une ligne de code.

Usages avancés

Chargement additif

- Il est possible de charger plusieurs scènes en même temps
- Elles viennent « s'additionner » les unes sur les autres
- Vous pouvez virtuellement charger autant de scènes en même temps que vous voulez

➔ Voyez ça comme plusieurs calques photoshop



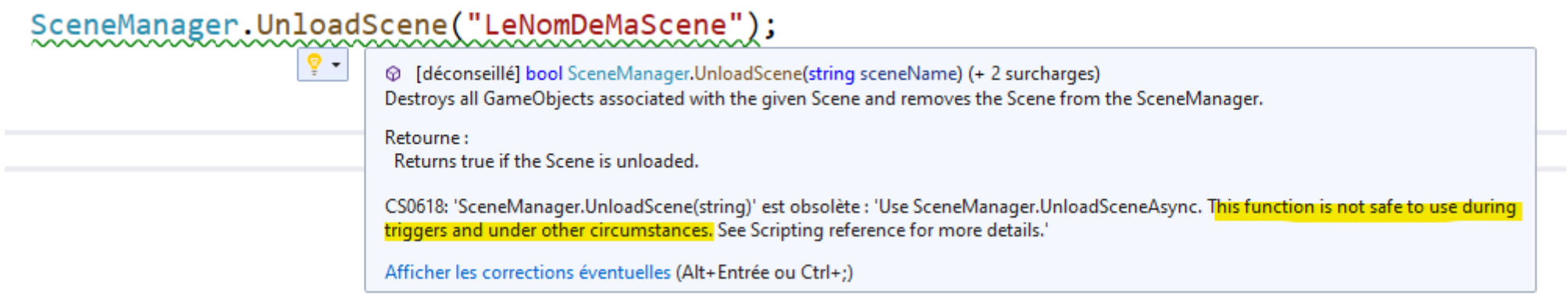
Chargement additif

- Pour charger une scène en mode additif :

```
SceneManager.LoadScene("LeNomDeMaScene", LoadSceneMode.Additive);
```

Chargement additif

- Pour décharger une scène en mode additif



Le déchargement de scènes en synchrone est déconseillé car cela entraine des bugs dans certains cas.

Chargement asynchrone

- Le chargement d'une scène est **bloquant**
- Lors d'un chargement, le moteur ne fait **QUE** charger la scène
- … pendant ce temps là, le joueur / utilisateur se tourne les pouces
- Les scènes qui ont beaucoup de contenu peuvent mettre un peu de temps à se charger.

➔ Solution : charger la scene en asynchrone

Chargement asynchrone

- Le principe est d'étaler le chargement de la scène sur plusieurs frames d'exécution du moteur :
  - L'application reste « utilisable » pendant le chargement
  - On peut afficher des animations pendant ce temps

Chargement asynchrone

- Charger une scène en asynchrone :

```
yield return SceneManager.LoadSceneAsync("LeNomDeMaScene");
yield return SceneManager.LoadSceneAsync("LeNomDeMaScene", LoadSceneMode.Additive);
```

- Décharger une scène en asynchrone :

```
yield return SceneManager.UnloadSceneAsync("LeNomDeMaScene");
```

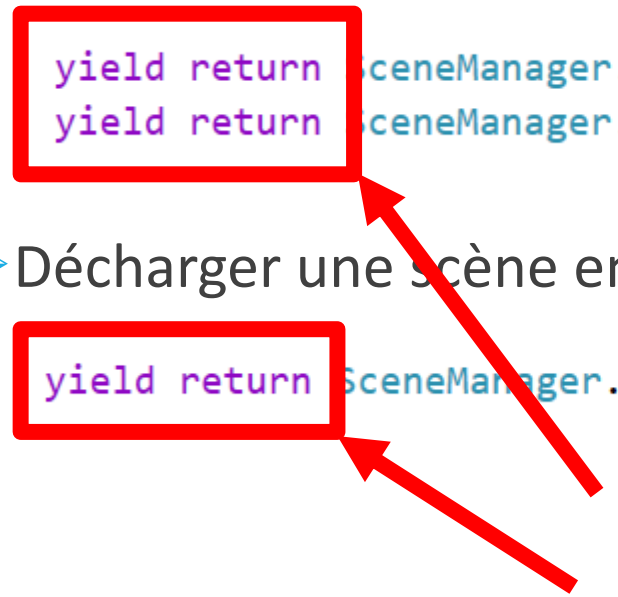
Chargement asynchrone

- Charger une scène en asynchrone :

```
yield return SceneManager.LoadSceneAsync("LeNomDeMaScene");
yield return SceneManager.LoadSceneAsync("LeNomDeMaScene", LoadSceneMode.Additive);
```

- Décharger une scène en asynchrone :

```
yield return SceneManager.UnloadSceneAsync("LeNomDeMaScene");
```



On reparlera de ça plus tard.

Pour l'instant n'utilisez pas le mode asynchrone ☺

GO