Coding Club

Unity3D

Bienvenue dans **Unity3D !**

A travers divers exercices, nous apprendrons à nous servir d'Unity3D, merveilleux outil de développement dans le domaine du jeu vidéo.

Unity3D ?

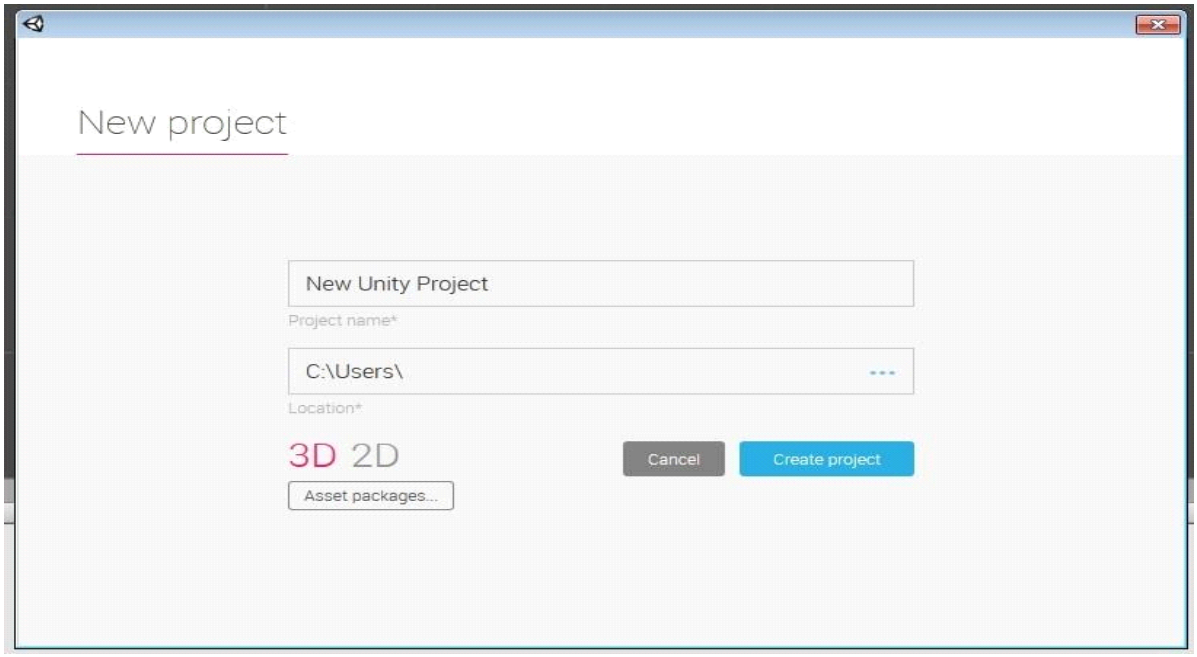
Il s'agit de ce qu'on appelle une «*Game engine*», soit un outil qui intègre un moteur graphique (rendu d'image) mais aussi un moteur physique (gravité, collisions), et qui fait ainsi gagner un temps précieux au développeur, qui n'ont pas à les coder (cela prend des années...).

A la portée de tout public, c'est un très bon outil pour développer son premier jeu et se familiariser avec cet univers.

SOMMAIRE :

1 . Lancer Unity3D

Commençons pas créer un nouveau projet.
- Nom du projet : Angry Club
- Choisissez projet 2D

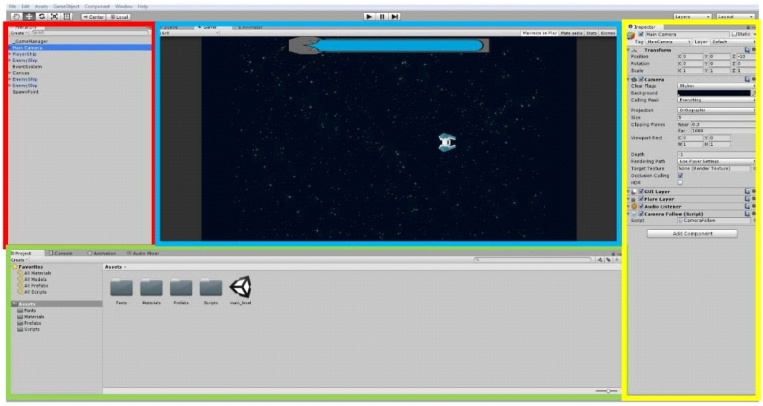


Une fois Unity ouvert sur une nouvelle scène.
- Sauvegardez (file → *save scene as*)

L'interface peut vous sembler chargée, mais rappelez-vous qu'il s'agit d'un outil complexe et très complet.

CODING CLUB

2 . L'éditeur



Par défaut, votre environnement devrait ressembler au suivant (si ce n'est pas le cas, cliquez sur le bouton «Layout» en haut à droite et sélectionnez «Default »

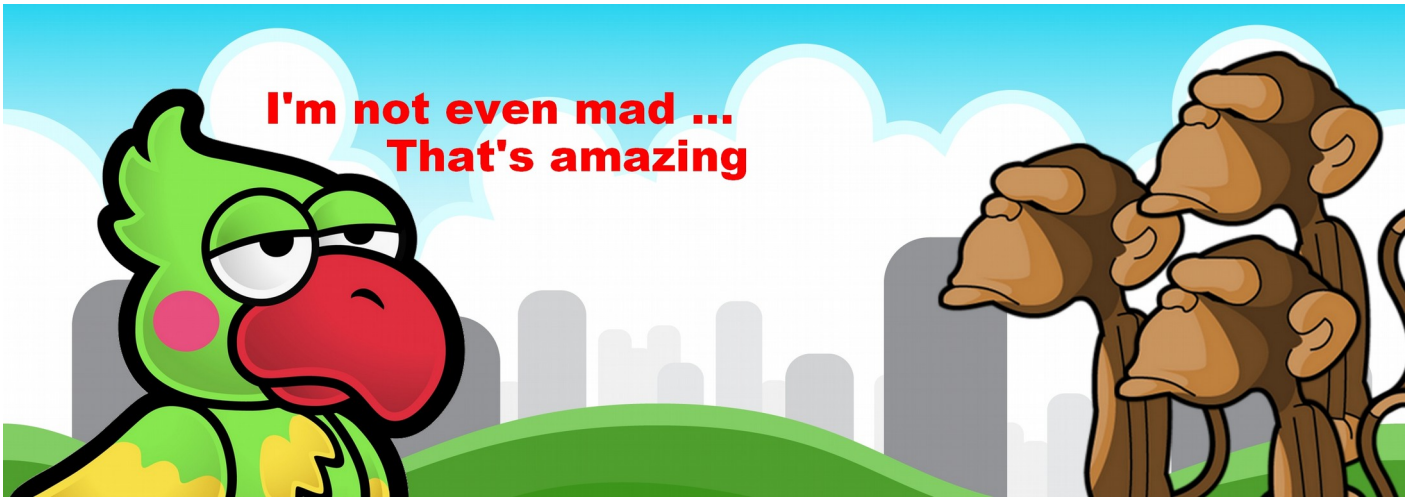
Rouge:
La fenêtre nommée «Hiérarchie», qui contient l'intégralité des objets présents dans votre scène.
Vert:
La fenêtre nommés «Projet», qui contient tous les Assets et les Préfabs que vous avez sauvegardés, et bien plus encore.
Jaune:
L'inspecteur
, qui vous donnera toutes les caractéristiques d'un objet sélectionné , et la possibilité de les modifier.
Bleu:
la fenêtre scène, qui donne un aperçu de votre scène courante.

Les déplacements dans la scène.

La souris est votre meilleur ami.
Sélectionnez la petite main en haut à gauche, ou appuyer sur 'Q'.

Molette: zoom / de-zoom.
Clic gauche (maintenir) : déplacer la caméra.
Clic droit : pivoter la caméra.

Sujet : Angry Club



Le Background :

Nous allons commencer par créer le « *background* » (fond), de notre scène.
Pour que l'image s'adapte correctement aux différentes résolutions d'écran nous allons utiliser un *Canvas*.

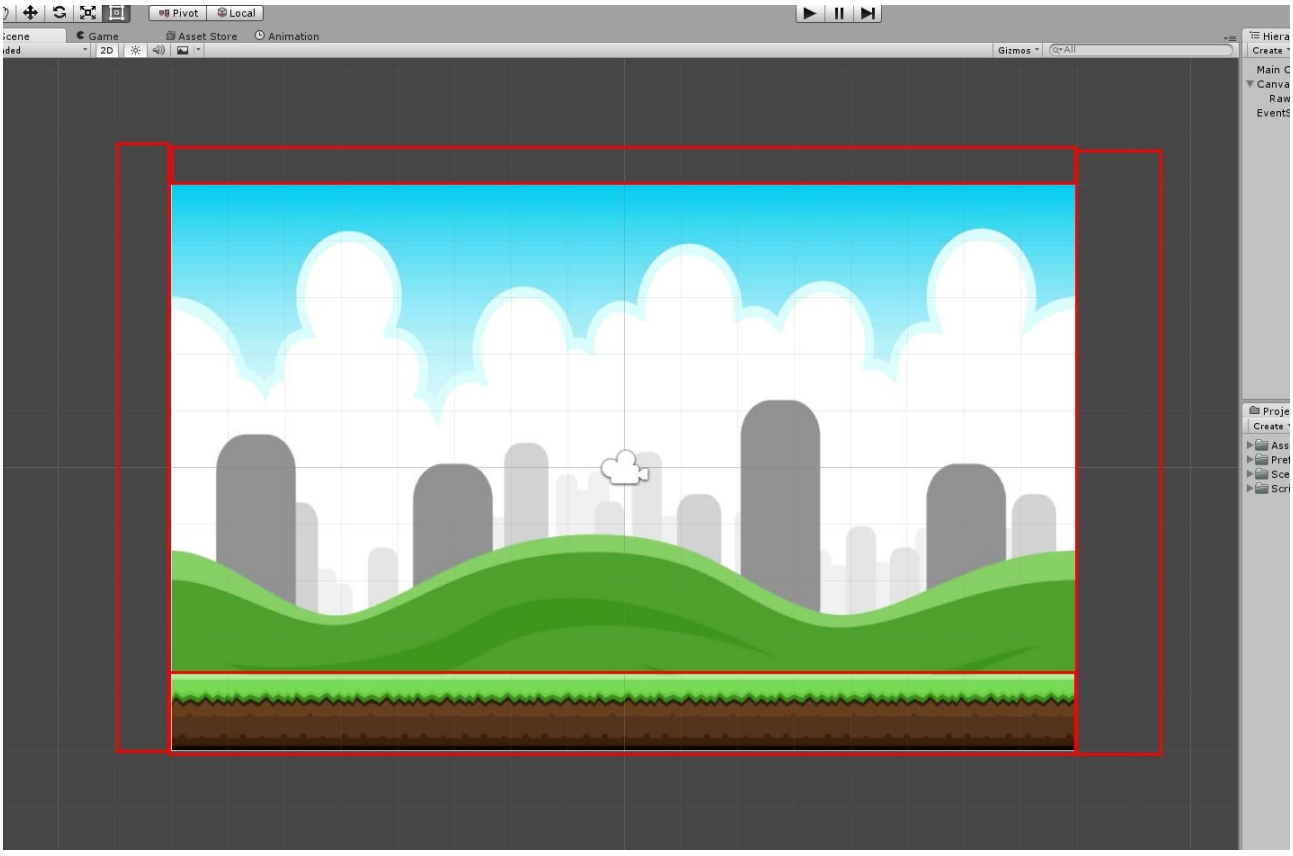
- **Créez un *Canvas***
- **Dans l'*inspector*, dans le composant *Canvas* du *GameObject* de type *Canvas* que vous venez de créer ; Séléctionnez le *render mode* et passez le en « *screen space – Camera* ».**
  Cela permettra d'adapter la taille de ce *canvas* en fonction de la caméra et donc de la vue du jeu.
- **Dans *render camera* vous devez maintenant spécifier la caméra**
- **Dans le composant *Canvas scaler*, dans le *UI scale Mode*, sélectionnez : *Scale with screen size*.**
  Cela adaptera les composants du *canvas* à la taille de l'écran.
- **Crée maintenant un *GameObject* de type *UI > RawImage*.**
  Appliquez lui la texture de l'image de fond fournie.
  Redimensionnez l'image de manière à ce qu'elle remplisse le *canvas*.

Les limites :

Maintenant il va falloir créer des limites à votre scène de façon à ce qu'aucun objet puisse sortir de l'écran.
Pour ce faire :

- **Créez un *GameObject* de type 2D → *Sprite.***
- **Désactivez son composant *Sprite renderer* dans l'inspecteur.**
- **Ajoutez lui un composant *Box Collider 2D.***
- **Taggez ce *GameObject* avec un *tag* spécial. Ex : « Limites » ou « Boundaries », à vous de voir.**
- **Dupliquez ce *GameObject* et placez les pour qu'il définissent les limites du terrain.**

Ex :



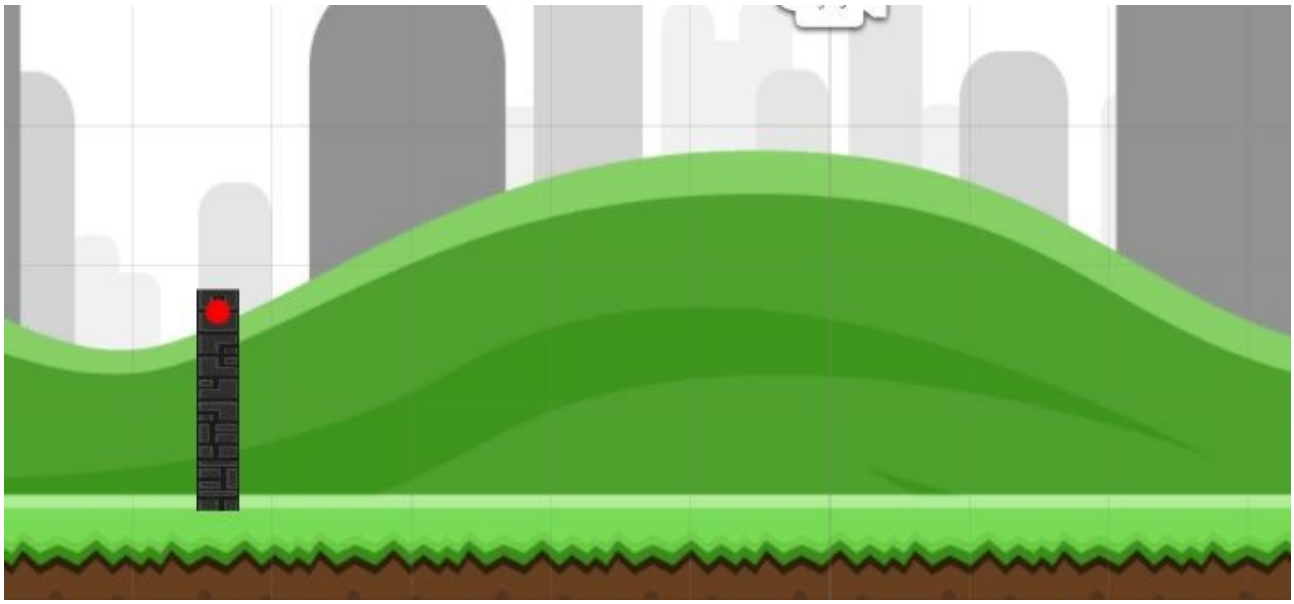
Astuce : ***Regrouper les GameObject de ce genre dans un GameObject vide est une bonne pratique pour la clarté du projet.***

Le Lanceur :

Nous allons maintenant créer le lanceur d'oiseaux.

- **Créez un *Sprite* et faite le ressembler à quelque chose qui pourrait se rapprocher d'un lance pierre (ou autre, laissez parler votre imagination) en utilisant le *spritesheet* fourni.**
- **Créez également un *GameObject* vide que vous placerez au niveau de l'accroche de « l'élastique », et que vous mettrez en objet fils de l'objet lance-pierre.**

Ex :



L'oiseau :

Nous allons maintenant passer à la création de notre arme de destruction massive ... Le perroquet !

A vrai dire, nous vous avons fournis d'autres animaux, mais selon la rumeur le perroquet et l'animal le plus aérodynamique !!!

bref...

- **Créez un GameObject de type Sprite.**
- **Choisissez l'animal de votre choix pour le visuel. (surtout pas un singe, il sont du coté obscur!)**
- **Votre projectile animalier aura besoin de plusieurs composants :**
  - **➔ Un *rigidbody 2D, kinematic avec un angular drag de 2.5.***
  - **➔ Un *Circle collider 2D.***
  - **➔ Un *Line renderer*.**

Les Singes :

Dans la jungle, le pire ennemi des perroquets ronds est bien entendu ... Les singes carrés !!! Ces créatures simiesques aux angles droits sont perfides et faux !

Ils sont composés de :

- **➔ Un *rigidbody 2D.***
- **➔ Un *Box collider 2D.***
- **➔ *Taggez* les « Ennemi » ou bien « Vicieux singe joufflu » à vous de voir !**

Faites en un *préfab.*

Les blocs :

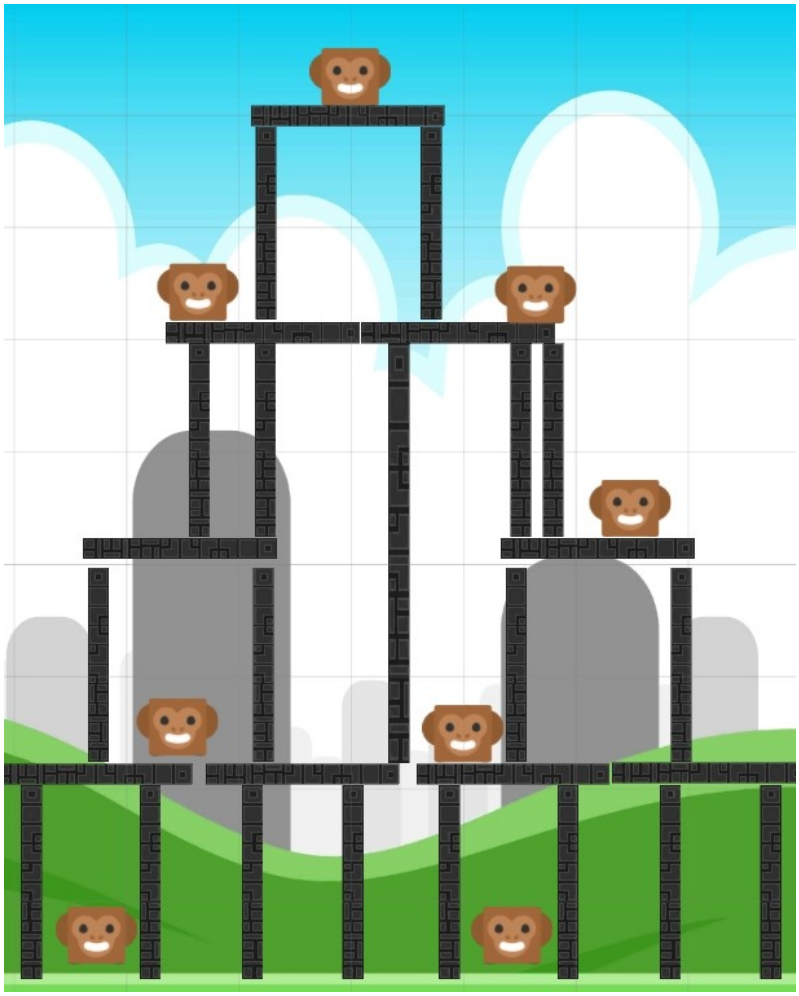
Il va falloir créer les blocs pour les structures singes.
Créez un *sprite* et choisissez le visuel que vous voulez dans le *spritesheet* fourni.

Les blocs doivent être composés de :

- ➔ **Un *Box Collider 2D.***
- ➔ **Un *rigidbody 2D.***
- ➔ **Etre *tagger* « Wall » ou « Structure » … A vous de voir.**

Faites en un *préfab*.

Ex :



CODING CLUB

4. Le Scripting

Le script de destruction :

Nous allons créer un script qui va permettre de d'identifier quand l'objet sur lequel est attaché ce script entre en collision avec un autre objet. Nous allons également déterminé la vitesse avec laquelle les objets entre en collision.

Commençons.

Créez un nouveau script et appelez le comme vous voulez, puis ouvrez le. (ex : '*CollisionManager*')

Vous devez rajouter une ligne en dessous de :

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

Rajoutez : « *using CodingClubAPI* »

Ensuite déclarez 2 variables éditable dans l'inspecteur (CAD public) un *float* correspondant au seuil de vitesse qui déterminera si l'objet et détruit ou pas. Et un *int* pour le nombre de points que cela rapportera au joueur de détruire cet objet.

Vous n'aurez pas besoin des méthodes *Start()* et *Update()*, en revanche vous allez utilisé la méthode *OnCollisionEnter2D* qui est appelée automatiquement quand l'objet entre en collision avec un autre objet. Voici son prototype :

void OnCollisionEnter2D(Collision2D) ;

A l'intérieur de cette méthode il faudra d'abord vérifier (avec une condition) le tag de l'objet avec lequel il est entré en collision et faire en sorte de n'agir que dans le cas ou l'objet en question n'est pas le sol.

Ex : `if (coll.gameObject.tag != "TOTO")`

Quand vous serez sur de vous, appelez la fonction suivante qui gérera la destruction de l'objet :

```
CodingClub.Destroy_on_collision(GameObject _this, Collision2D coll, float seuil, int points) ;
```

Voilà, votre script est prêt. Vous pouvez l'appliquer sur vos *préfabs* de blocs et de singes, en leur attribuant les valeurs de vôtre choix dans l'inspecteur.

Le script du Joueur :

(Rajoutez : *« using CodingClubAPI » comme plus haut.)*

Dans un premier temps vous allez avoir besoin de déclarer plusieurs variables.

1 public *GameObject* :
Pour le point d'encrage sur le lanceur.

3 private *bool* :
Appelez les respectivement, « follow », « played » et « drawline ». Vous comprendrez plus tard.

1 private *Vector2* :
Va vous servir à sauvegarder la position d'origine de l'oiseau.

1 private *Raycasthit2D* :
Nous l'utiliserons pour savoir ou le joueur clic.

C'est la qu'on entre dans le vif du sujet !

Dans la fonction *Start()*, nous allons commencer par assigner la position de départ au Vector2, ex :

```
oldPos = transform.position;
```

```
        drawline = true;
```

Nous attribuons l'état *true* à *drawline* car nous voulons qu'une ligne se trace entre l'oiseau et le lanceur.
Nous utiliserons ce *booléen* pour savoir quand afficher ou quand cacher cette ligne.

```
        played = false;
```

***played* est à *false* car nous n'avons pas encore jouer, nous l'utiliserons plus tard pour savoir quand faire réapparaître le joueur etc ...**

```
        follow = false;
```

Nous utiliserons follow pour savoir quand l'oiseau doit suivre le curseur de la souris ou pas.

Dans la fonction *Update()* :

- Il va vous falloir une condition qui vérifie l'état de drawline.
Si drawline est true alors il faut afficher la ligne et donc appeler la méthode :

CodingClub.Render_line() ;

En lui passant en paramètre le *gameobject* du *player* (this.gameObject) et le *GameObject* du point d'ancrage.

- Vous devez appeler la méthode CodingClub.Level_Reload()
En lui passant en paramètre la *string* correspondant au *tag* que vous avez assigné aux ennemis. Cette fonction recharge la scène dans le cas ou il n'y aurait plus d'ennemis.

- Vous aurez maintenant besoin d'une condition qui vérifie 2 choses.
La première, avez vous jouer ? (*played* est-il *true*?) et (&&)
« this.GetComponent<Rigidbody2D>().velocity == new Vector(0, 0) »
Traduction : Est ce que le *gameObject* du player est-il à l'arrêt ?

Le Drag'n drop :

(toujours dans Update())

Le principe va être le suivant, nous allons faire en sorte de pouvoir cliquer sur le perroquet et tant que nous ne lâchons pas le clic de la souris, le perroquet suit le curseur.
Une fois le clic relâcher, il partira dans la direction du vecteur entre le perroquet et le point d'encrage. Nous allons utilisé les booléens dans le but d'afficher ou non la ligne, replacer ou non le joueur etc ...

- Vous devez assigné à votre *private Raycasthit2D* le retour de la fonction : CodingClub.Mouse_pos()
Cela vous permet de récupérer l'objet sur lequel vous avez cliqué.

- Créez une condition qui va vérifier le retour de la fonction CodingClub.is_moving()
Fonction qui prend 2 paramètres, le *bool played* et le *gameObject* du *player* .
Si cette fonction renvoie true, cela veut dire que votre petit perroquet ne bouge plus et il faut donc le replacer.
Pour ce faire appelez la fonction CodingClub.restart_player()
Fonction qui prend 2 paramètres, le *gameObject* du *player* et votre *Vector2* qui correspond à la position de départ.
Nous allons maintenant remettre *drawline* à *true* et *played* à *false.*

- Il vous faut maintenant créer une condition complexe.
Vous devez vérifié que votre *Raycasthit2D* n'est pas *null*, que l'objet sur lequel vous avez cliqué est bien le *GameObject* du *player* et que vous n'avez pas encore joué.

A l'intérieur de cette condition, vous allez devoir détecter deux cas de figures.

Si le clic gauche de la souris est pressé, il faut passer *follow* à *true* ;
L'oiseau suivra donc le curseur.

Si le clic gauche et relâché il faut passer ***played*** à ***true***, ***follow*** à ***false*** et ***drawline*** à ***false***.
En gros, une fois le clic lâché, vous avez jouer (du coup ***played*** est ***true***), l'oiseau doit arrêter de suivre le curseur(***follow*** passe ***false***) et la ligne ne doit plus s'afficher (***drawline*** passe ***false***).

Astuce : Input.GetKeyDown / Input.GetKeyUp

Dans cette condition après avoir modifié les trois booléens vous devez encore appeler deux fonctions.

CodingClub.Shot(GameObject, GameObject) ;
Qui prend en paramètre le ***GameObject*** du ***player*** et le ***GameObject*** du point d'ancrage.
Elle sert à faire partir l'oiseau dans la bonne diréction.

CodingClub.Hide_line(GameObject) ;
Qui prend en paramètre le ***GameObject*** du ***player*** .
Elle sert à cacher la ligne (l'élastique).

Cela devrait ressembler à quelque chose dans ce genre la :

```
Si (Raycasthit2d n'est pas null && qu'il s'agit bien du joueur && je n'ai pas encore jouer )
{
        Si ( Je fais un clic gauche )
        {
                follow = true ;
        }

        Si ( Je relâche le clic gauche )
        {
                played = true ;
                follow = false ;
                drawline = false ;
                CodingClub.Shot(this.gameObject, ancrage.gameObject) ;
                Coding.Club.Hide_line(this.GameObject) ;
        }
}
```

Courage, dernière condition !!!

Dans l'Udpate() il manque encore une condition.

Si le joueur maintient le clic gauche et que follow est à true.
Alors l'oiseau doit suivre le curseur.

Et bien voila ... Tout est dis !

Astuce : Pour connaître la fonction qui permet de faire suivre la souris, vous pouvez aller voir le README du plugin CodingClubAPI.

Astuce' : Allez voir la documentation de Inpu.GetKey si vous avez du mal à trouver comment récupérer les inputs de la souris. (Keycode.Mouse0 pour le clic gauche;)).

Bonne chance !