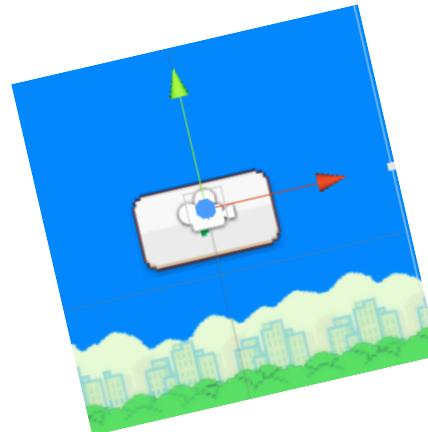
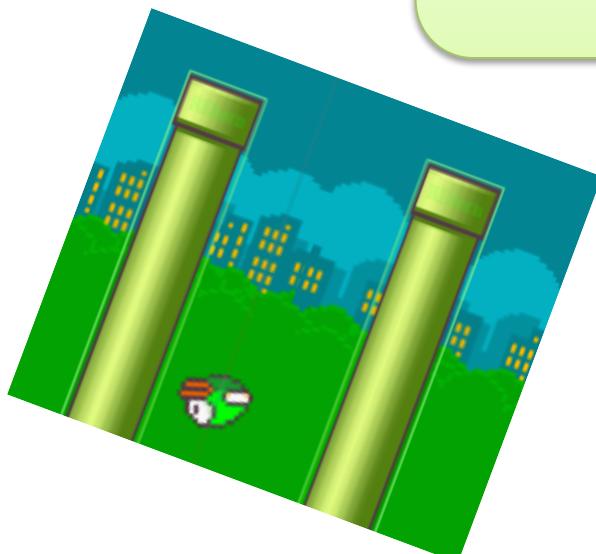


flapTheBird  
Unity – Cross développement  
TP

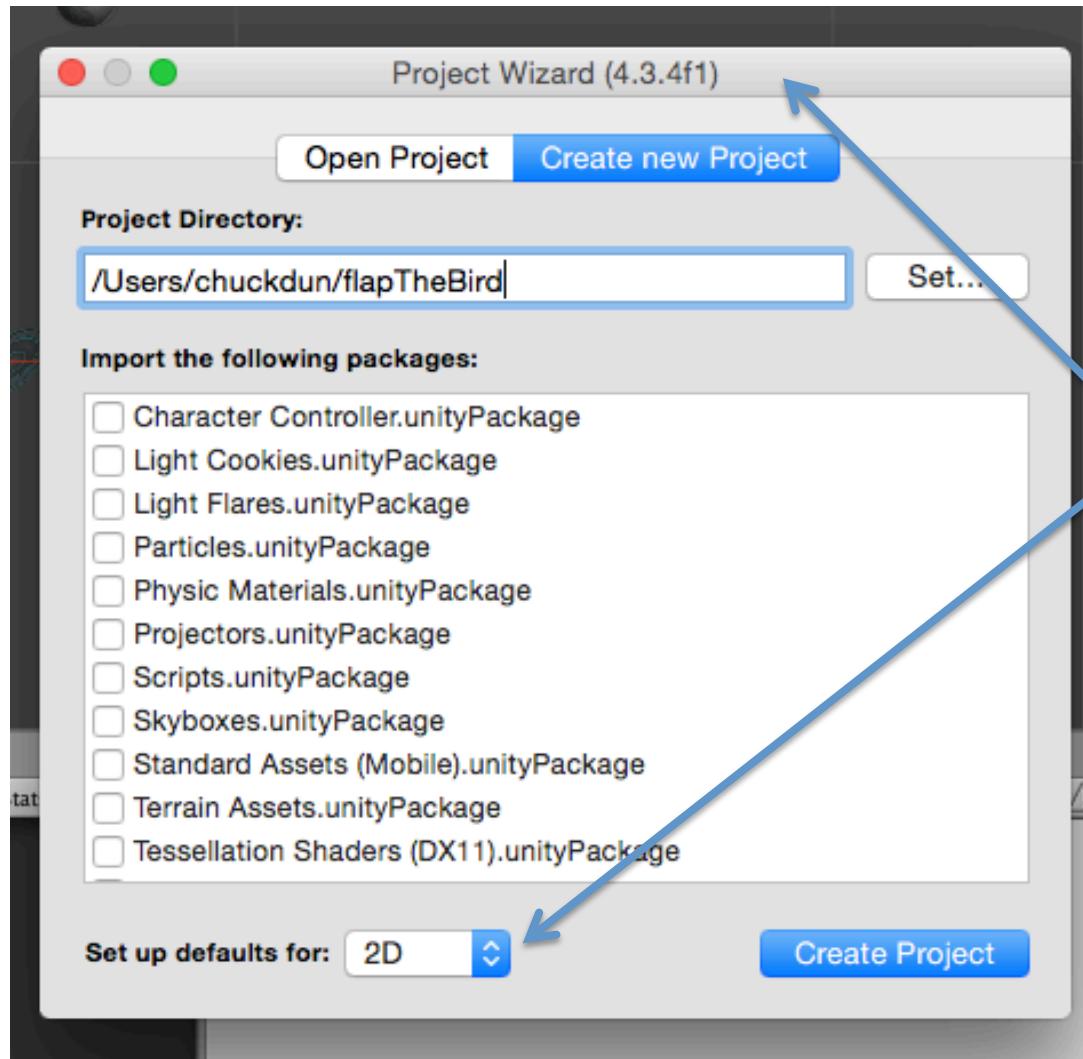


Roland Garnier  
José Martins  
Gérard Rozsavolgyi

Scene 2  
Le menu

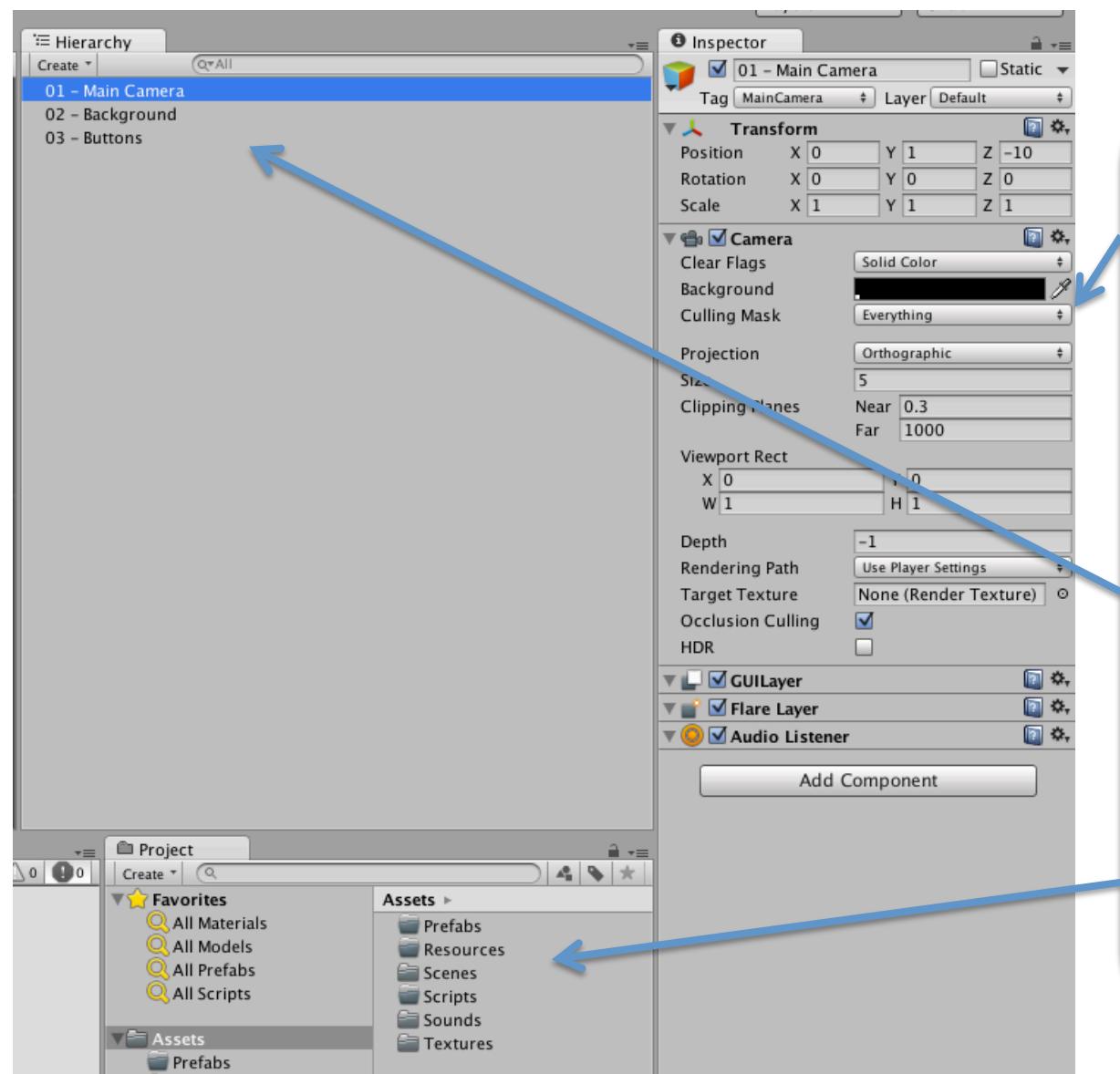
(La scene 1 arrive plus tard ...)

## Etape 1 – Créer un projet en 2D



Créer un nouveau projet  
Passer en mode 2D

## Etape 2 – Préparer la scène



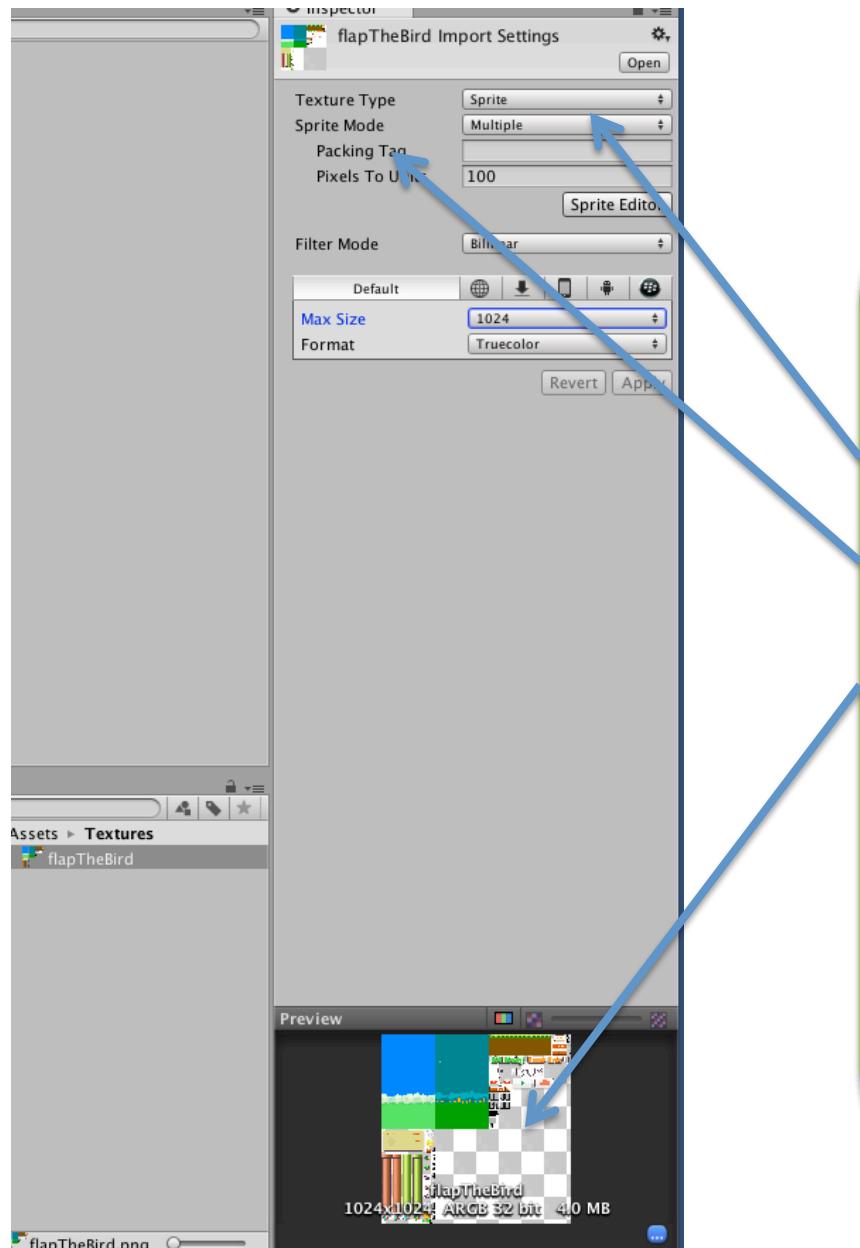
Passer la caméra en orthographic

Passer le clear flags en solid color et choisir une couleur de fond

Créer les gameObjects vides suivants (l'objet caméra existe déjà, il est par défaut dans la création d'une scene, renommer le)

Créer la hiérarchie de dossier

## Etape 3 – Importer la spritesheet



Importer la planche de sprite  
flapTheBird.pnd dans le dossier texture

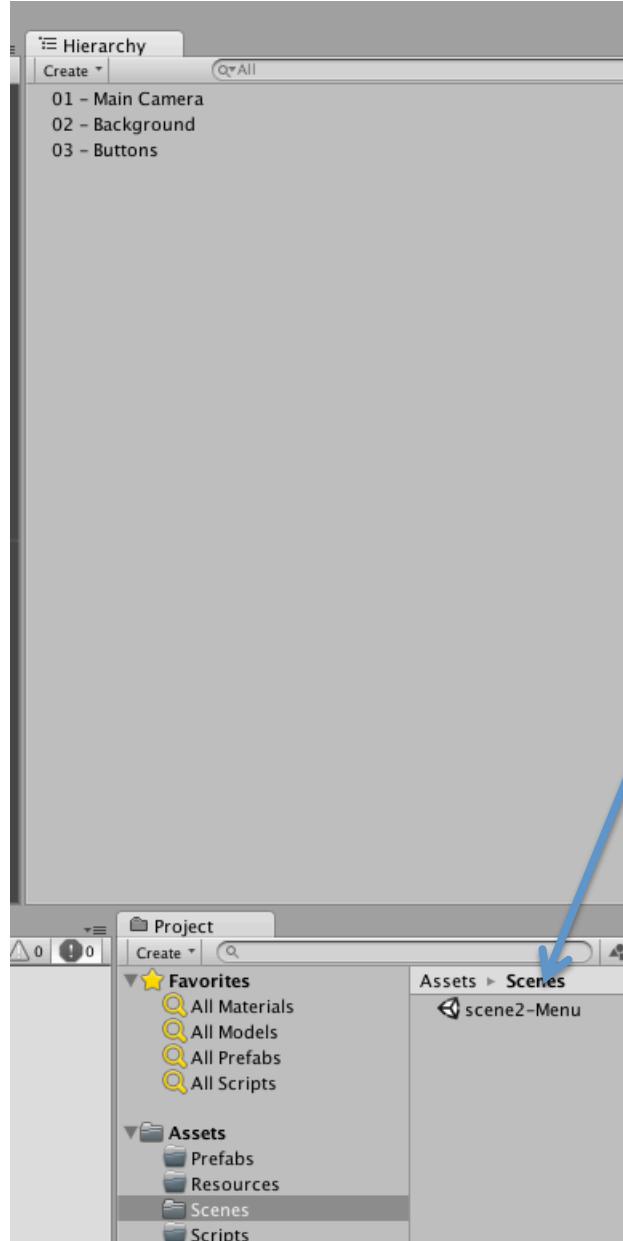
Changer le texture type en sprite

Changer le sprite mode en multiple

La transparence doit être active dans la  
texture, vérifier ici que le quadrillage  
blanc et gris est présent

Importer le son touchButton.wav dans le  
dossier Sounds

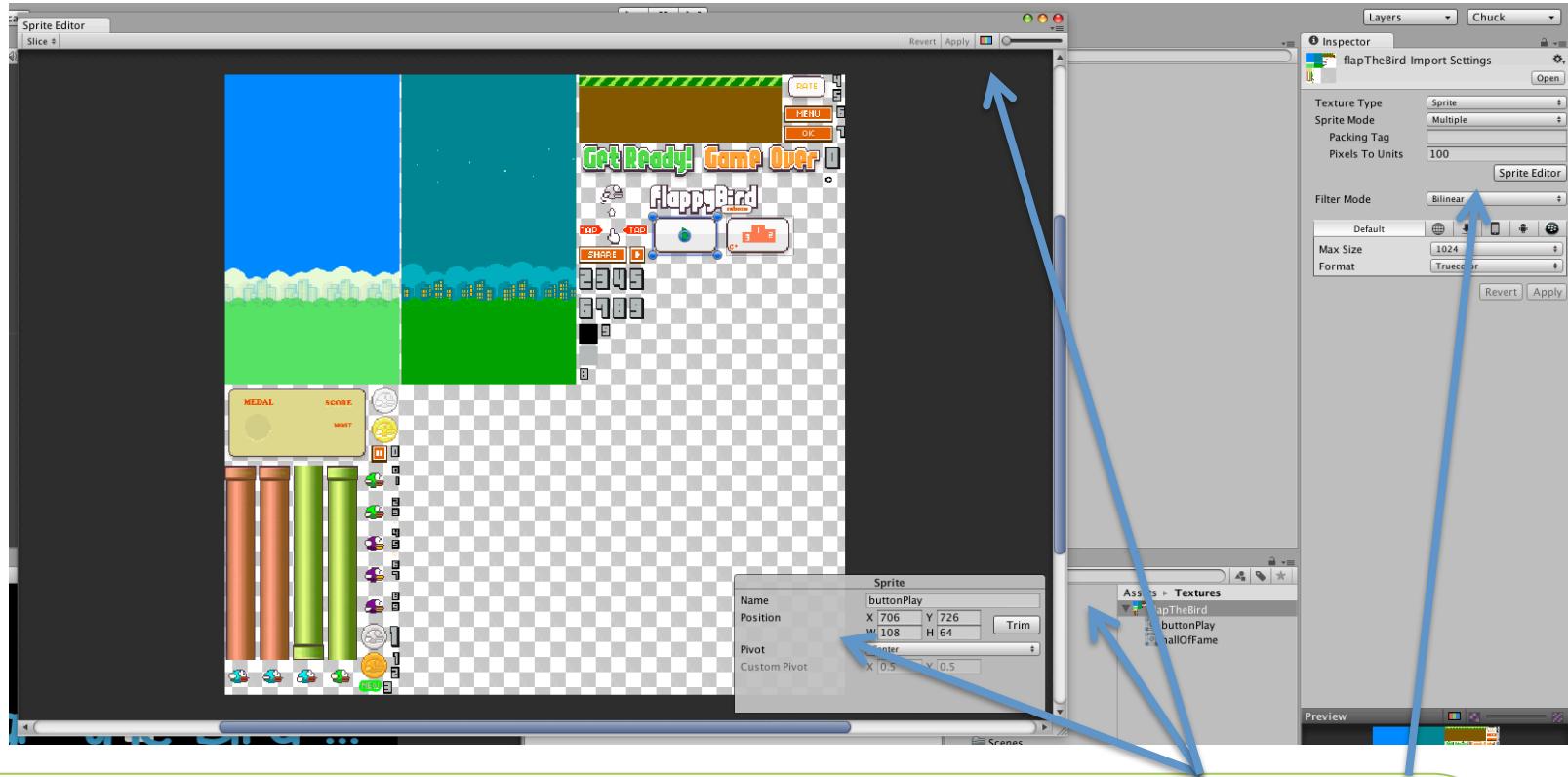
## Etape 4 – Sauver la scene 2



Sauvegarder la scene sous le nom  
scene2-Menu dans le dossier Scenes

Pourquoi l'avoir nommé scene2 ?  
Plus tard la scene1 sera créée sous la  
forme d'un splash screen.  
Elle va permettre de créer (en plus de  
l'affichage d'une image) les singltons  
nécessaires au jeu

## Etape 5 – Créer une sprite texture



Sélectionner l'image dans le dossier texture

Cliquer sur sprite editor

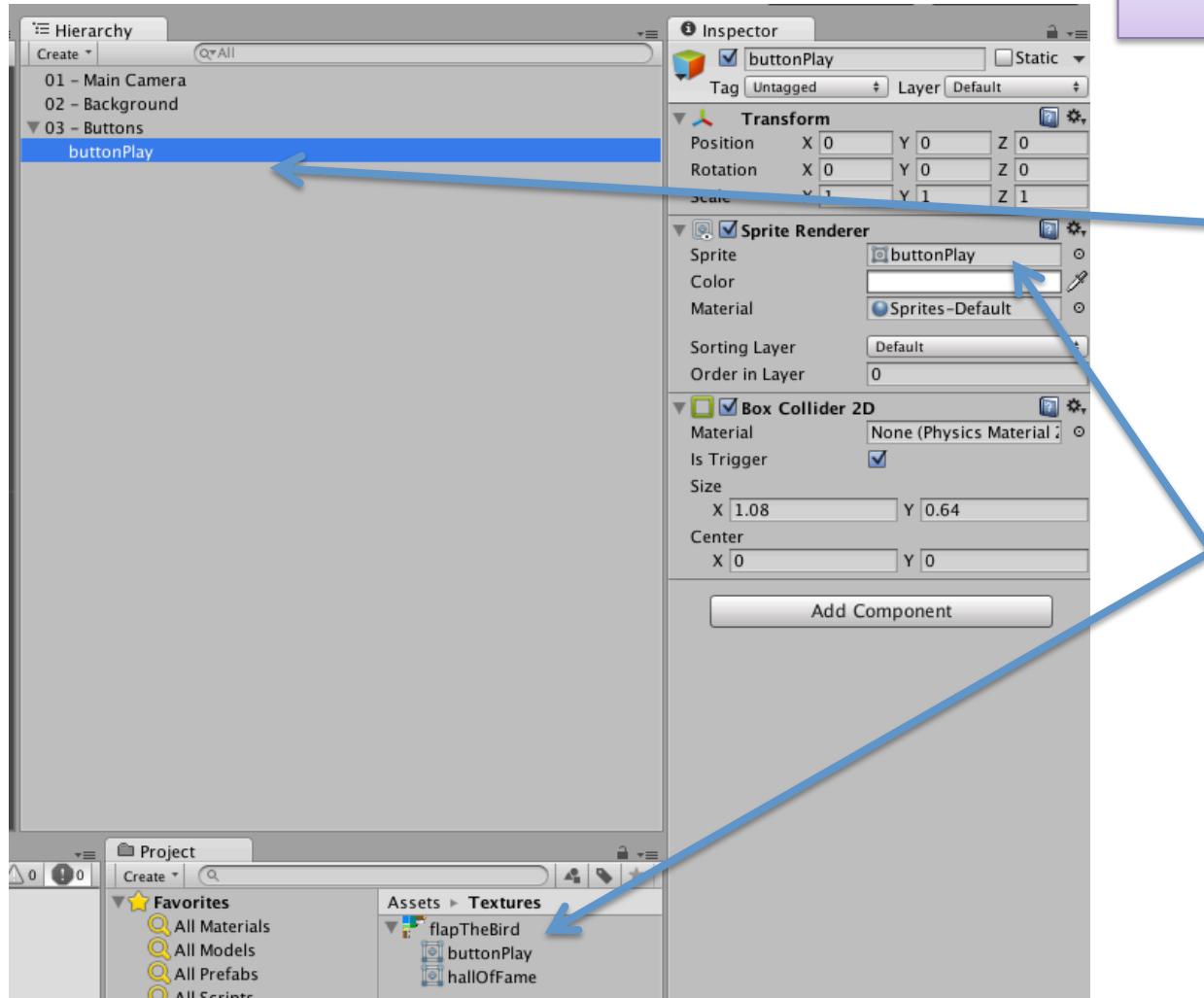
Créer un sprite dans l'éditeur en le détourant

Nommer le buttonPlay

Avant de quitter valider en appuyant sur apply

Sous la texture flapTheBird une texture buttonPlay est maintenant présente

## Etape 6 – Créer un sprite



Créer un gameObject de type sprite

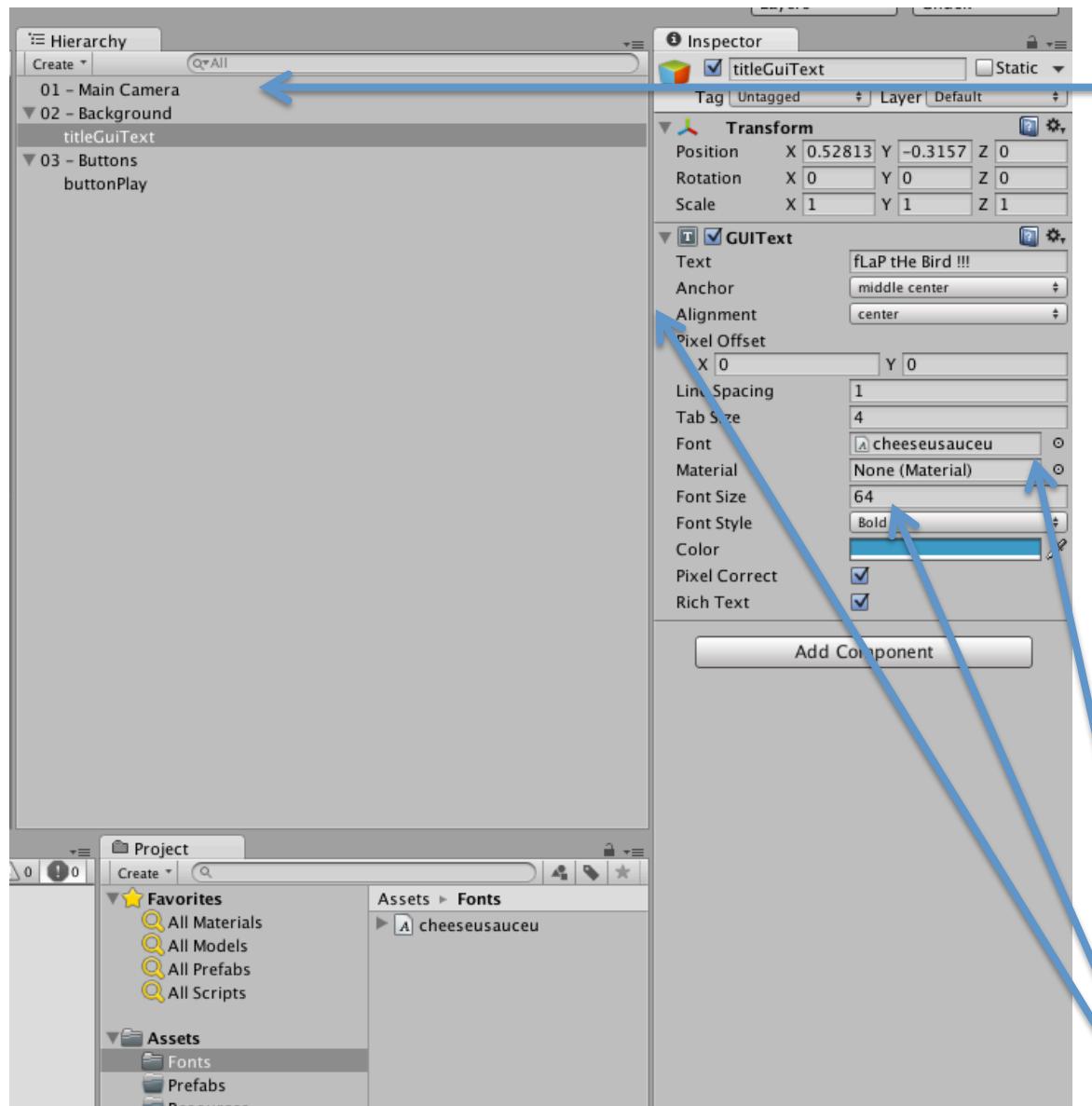
Nommer le buttonPlay

Associer la texture buttonPlay à ce sprite

Ajouter un composant boxcollider2D avec un trigger activé

Glisser le dans le gameObject vide  
02 - Buttons

## Etape 7 – Créer une GuiText (1)



Créer un gameObject de type  
GUIText

Nommer le titleGUIText

Créer un nouveau dossier Font  
dans l'onglet Project

Importer la font  
cheeseusauceu.ttf dans ce  
dossier

Dans l'inspector de  
titleGuitext sélectionner la  
nouvelle Font importée

Changer le texte et mettre les  
mêmes valeurs que sur cette  
image pour le Anchor/  
Alignment /Size

## Etape 7 – Créer une GuiText (2)

Avec la nouvelle version de Unity l'object Guitext n'est plus disponible sous cette forme

Pour le récréer:

Créer un gameObject vide

Ajouter un composant Uitext en utilisant AddComponent

L'accès à l'object GuiText est maintenant possible comme avec l'ancienne version

## Etape 7 – Créer une GuiText(3)



Le résultat de l'ensemble de ces étapes dans la fenêtre Game

## Etape 8 – Positionner le tout à l'écran (1)

Comment garder le sprite buttonPlay au centre de l'écran ?

En mettant ses coordonnées dans son Transform.position sous l'éditeur à

X = 0

Y = 0

Z = 0

Zoomer le bouton play en jouant sur le Transform.scale sous l'éditeur

X = 2

Y = 2

Z = 2

Ce type de facteur de zoom est peu recommandé, le dessin double sa taille d'origine et des artefacts peuvent apparaître avec certaines textures. Il est plus usuel de prendre une texture de grande taille et de la réduire.

Essayer un zoom plus grand pour voir apparaître ces artefacts.

## Etape 8 – Positionner le tout à l'écran (2)

Comment garder le texte titleGuiText en haut et au centre de l'écran ?

En mettant ses coordonnées dans son Transform.position sous l'éditeur à

X = 0.52

Y = 0

Z = 0

Pourquoi ?

Encore une fois Unity se joue des unités de mesure et le gameObject guiText complexifie encore davantage le tout car son ancre (anchor) peut être déplacée comme un sprite

Jouer avec les coordonnées du Tranform.position pour voir le résultat

Aide: Pour remettre la texture au centre remettre ces coordonnées:

X = 0

Y = 0.25

Z = 0

## Etape 9 – Déetecter l'appui sur le bouton Play

Création d'un script touchButtonPlay:

- dans un premier temps il affichera un message dans la console lors de sa sélection par l'utilisateur
- dans un second temps il lancera la scene3 qui sera la scène du jeu

Ce script sera attaché au gameObject buttonPlay

Attention, penser à gérer l'appui avec une souris et l'appui sur un écran tactile:

```
if (Input.touchCount > 0) {  
    ...  
}  
else if (Input.GetMouseButton(0)){  
    ...  
}
```

## Etape 10 – Changement de resolution et d'orientation (1)

Dans l'onglet Game, créer un nouvelle résolution:

640 \* 1136

Nommer la Vertical

Que se passe-t-il ?



## Etape 10 – Changement de resolution et d'orientation (2)

Première solution:

Adapter la taille du texte, changer la valeur de `FontSize` dans le `GameObject titleGuiText`  
Cette solution est-elle viable ?

Comment adapter automatiquement la taille du texte à l'écran ?

Créer un script `getScreenSize`, ce script:

- récupère la taille de l'écran:
- calcule un ratio par rapport à cette taille
- attacher le au `gameObject titleGuiText`

Le `GameObject titleGuiText` prend en compte ce ratio pour définir la taille de sa police  
La taille de la police doit être changée dynamiquement ...

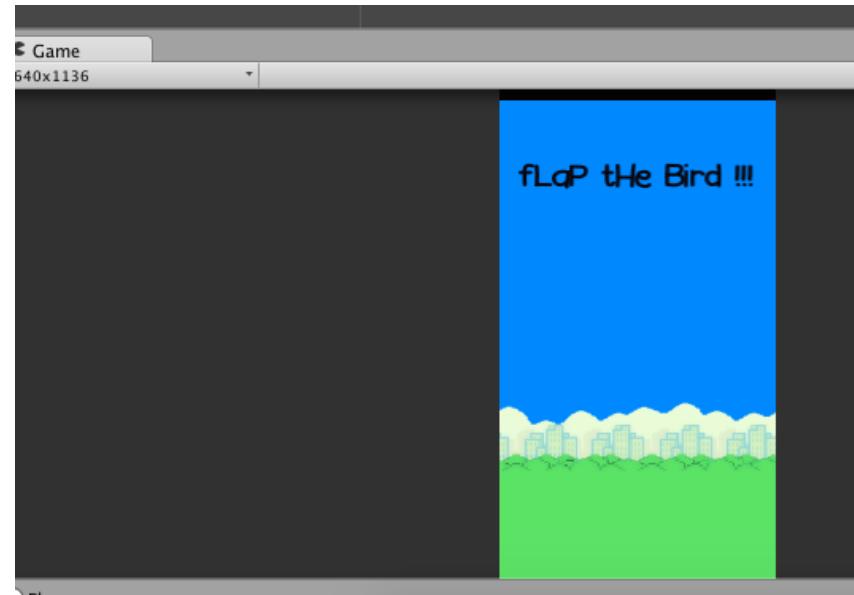
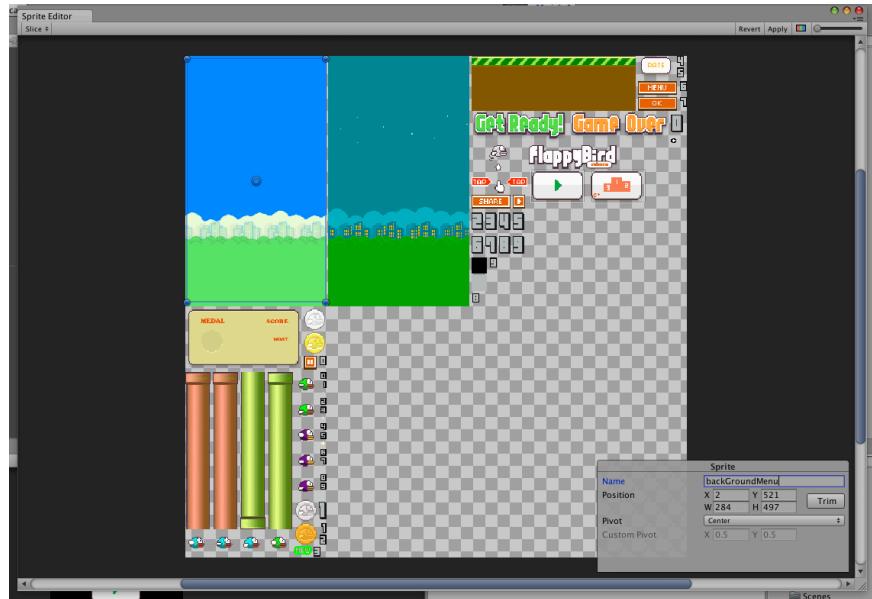
Astuce: `Screen.width` ... `Screen.height`

## Etape 11 – Ajouter un fond en mouvement et en boucle (1)

Dans le dossier Textures:

- créer une nouvelle texture de sprite en utilisant l'éditeur de sprite
- détourer le fond d'écran en haut à gauche de la spriteSheet
- nommer la backGroundMenu
- créer un nouveau gameObject backGround1 de type sprite avec cette texture, ajouter lui un rigidbody2D, cocher isKinematic et glisser le ensuite dans 02-BackGround
- ajuster le scale rate du sprite pour qu'il occupe tout l'écran

Où est passé le bouton play ?..



## Etape 11 – Ajouter un fond en mouvement et en boucle (2)

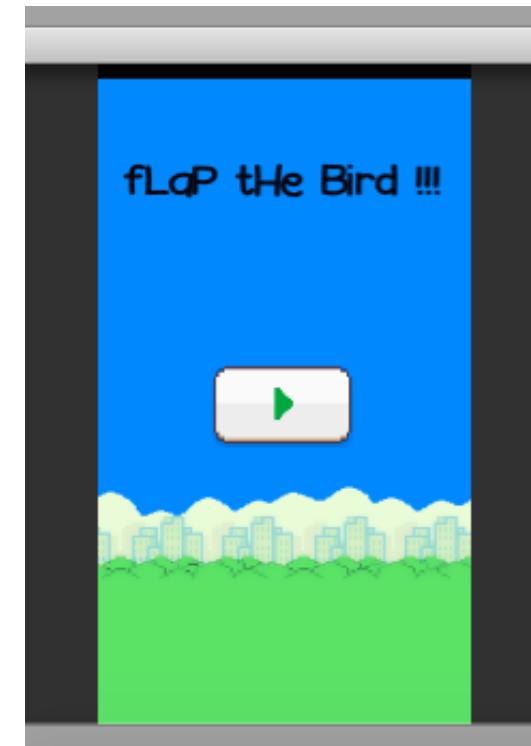
Le bouton play a disparu ... Pourquoi ?

Regarder le transform.position.z du buttonPlay, il vaut 0

Regarder le transform.position.z du backGround1, il vaut 0

Ils partagent le même Z order (ordre d'affichage en profondeur) et donc le dernier arrivé a toujours raison ...

Changer la valeur du transform.position.z du buttonPlay en la passant à -1, le bouton réapparaît



A screenshot of the Unity interface showing the Hierarchy and Inspector panels. The Hierarchy panel shows a scene structure with three main groups: "01 - Main Camera", "02 - Background" containing "backGround1" and "titleGuiText", and "03 - Buttons" containing "buttonPlay". The "buttonPlay" object is currently selected, highlighted with a blue selection bar at the bottom of the hierarchy list. The Inspector panel to the right shows details for "buttonPlay": it is a "buttonPlay" game object with "Static" unchecked, "Untagged" tag, and "Default" layer. Under the "Transform" section, the "Position" is set to X: 0, Y: 0, Z: -1. The "Sprite Renderer" component is also visible at the bottom.

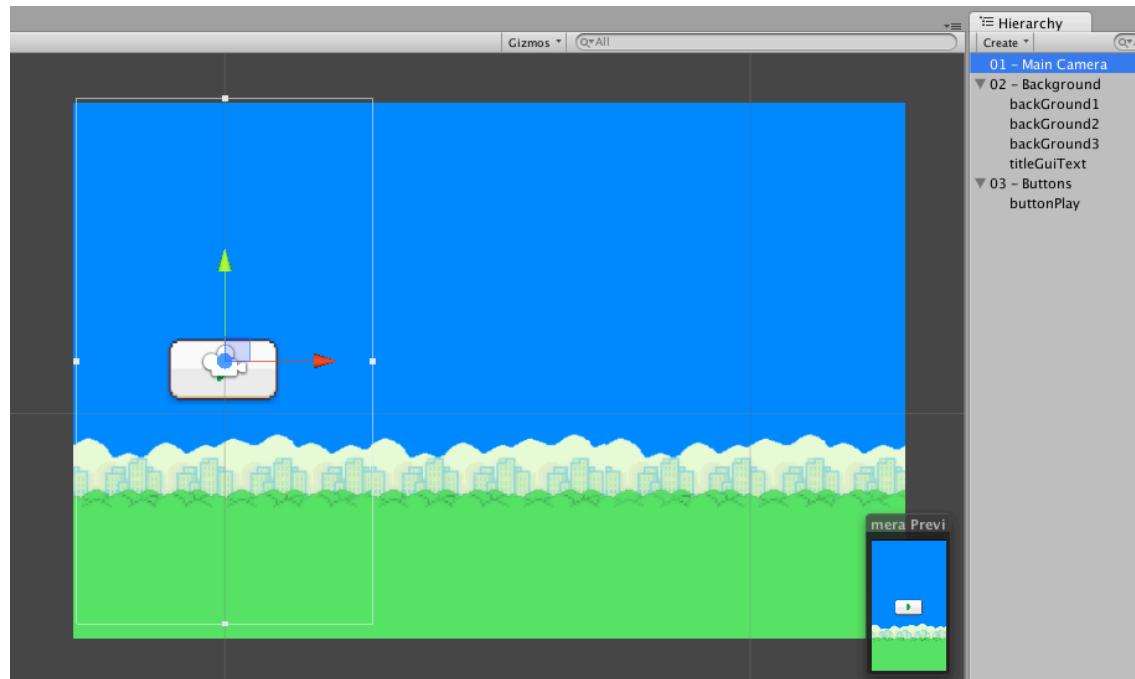
## Etape 11 – Ajouter un fond en mouvement et en boucle (3)

Créer 2 copies du gameObject backGround1

Renommer les backGround2 et backGround3

Positionner les à droite de backGround1 pour former un fond continu plus grand que la zone de caméra dans l'éditeur de scène

Créer un script (moveBk) qui scrolle les 3 sprites vers la gauche et qui les recrée à droite de l'écran ... Mais où exactement ?



## Etape 11 – Ajouter un fond en mouvement et en boucle (4)

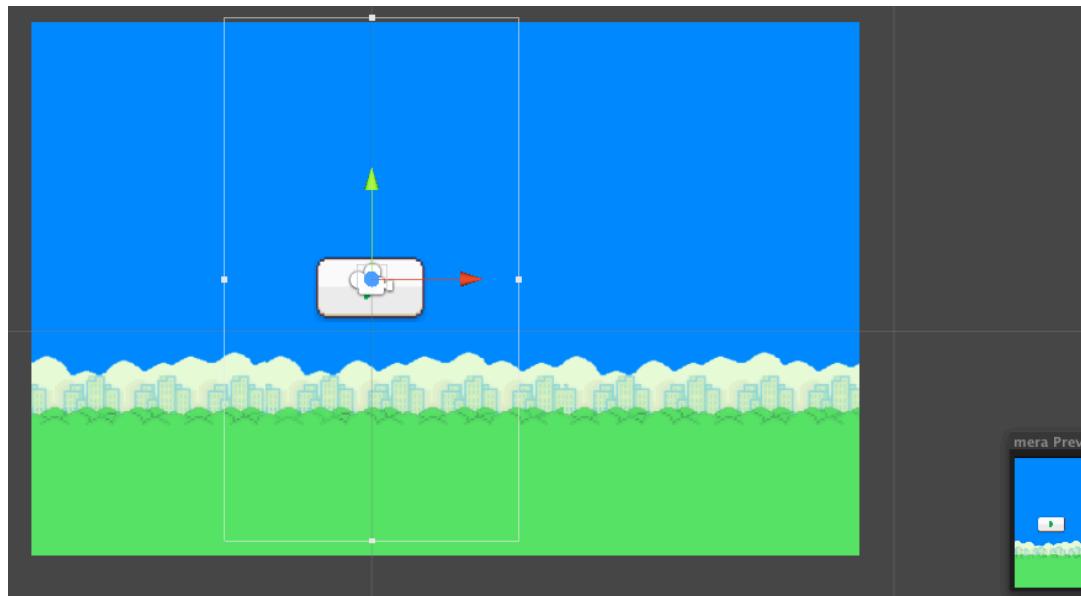
Regarder la position en X de backGround3 ...

Quand background1 va sortir de l'écran à gauche, il va falloir lui donner la position d'origine en X de backGround3, il prendra ainsi sa place ... et la boucle est bouclée ...

Pour réaliser cela dans le script, déclarer une variable public de type float

Renseigner sa valeur en indiquant la valeur en X de backGround3

Quand le sprite sort de l'écran il prend maintenant la valeur en X de backGround3, ses valeurs Y et Z restent inchangées.



## Etape 11 – Ajouter un fond en mouvement et en boucle (5)

Voici la méthode update du script moveBk, avec la variable public positionRestartX qui est de type float et qui est renseignée avec la valeur du transform.position.x de background3

```
void Update()
{
    rigidbody2D.velocity = movement;
    siz.x = gameObject.GetComponent<SpriteRenderer> ().bounds.size.x;
    siz.y = gameObject.GetComponent<SpriteRenderer> ().bounds.size.y;

    // Si le sprite sort de l'écran à gauche
    // Recalcul d'une nouvelle position en Y comprise dans les limites autorisées de l'écran
    // Repositionnement en X à la position d'origine de bacGround3

    if (transform.position.x < leftBottomCameraBorder.x - (siz.x / 2))
    {
        transform.position = new Vector3(positionRestartX, transform.position.y, transform.position.z);
    }
}
```

## Etape 12 – Améliorations possibles pour ce menu

Dans le cadre du projet il est possible pour ce menu:

- D'ajouter un bouton qui ouvre une page internet présentant le jeu
- De faire bouger le titre du jeu à l'écran
- D'ajouter des étoiles ou des nuages dans le ciel qui bougent à différentes vitesses
- D'animer les boutons en changeant la texture du sprite à la volée ou en les faisant bouger quand l'utilisateur appuie dessus
- Etc ...

## Scene 3 Le jeu

## Etape 13 – Création de la scene 3, le jeu en live

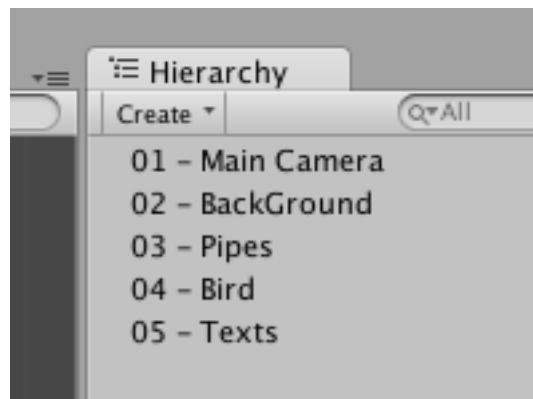
Créer une nouvelle scene dans le menu file/new scene

Sauver la sous le nom scene3-Game dans le répertoire Scenes

Créer 4 gameObject vides et nommer les comme dans l'image suivante

Penser à renommer la caméra

Sauver le tout (file/Save project)

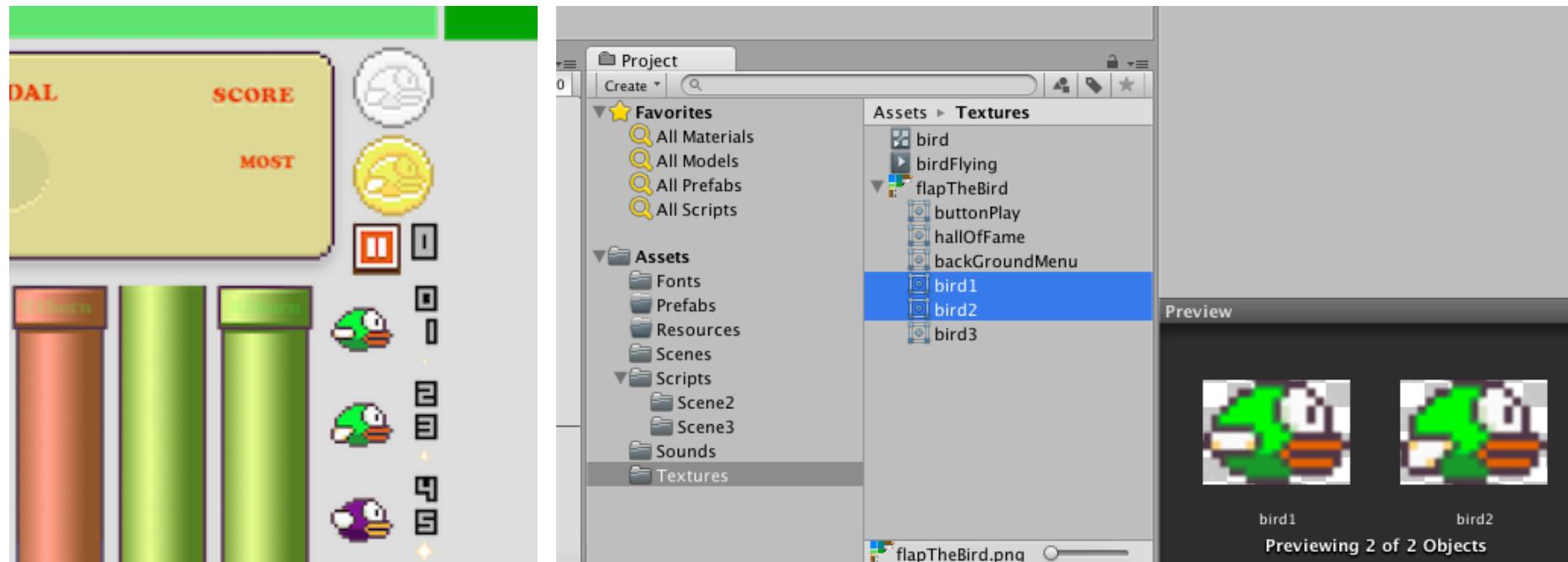


## Etape 13 – Crédit de la texture de l'oiseau

Créer 2 textures de type sprite à partir de la planche de sprite et l'éditeur de sprite

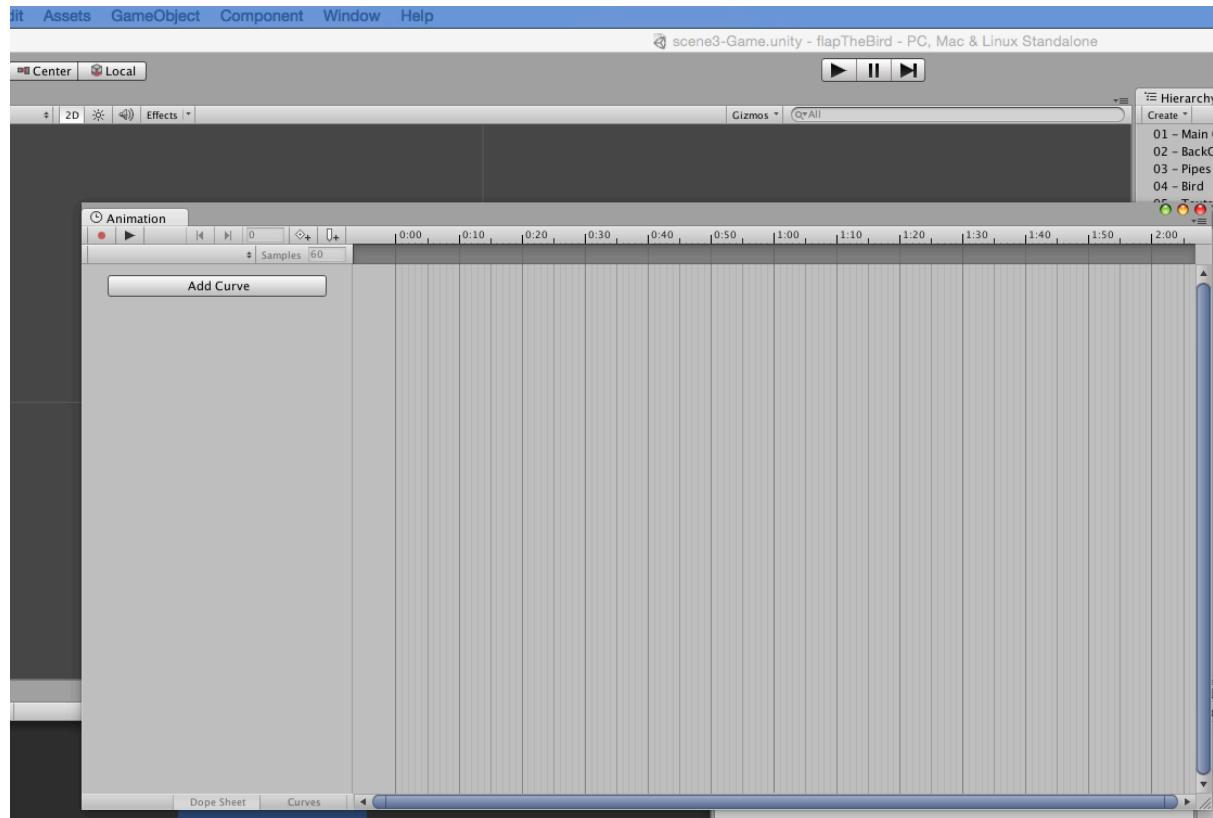
Nommer les:

- bird1
- bird2



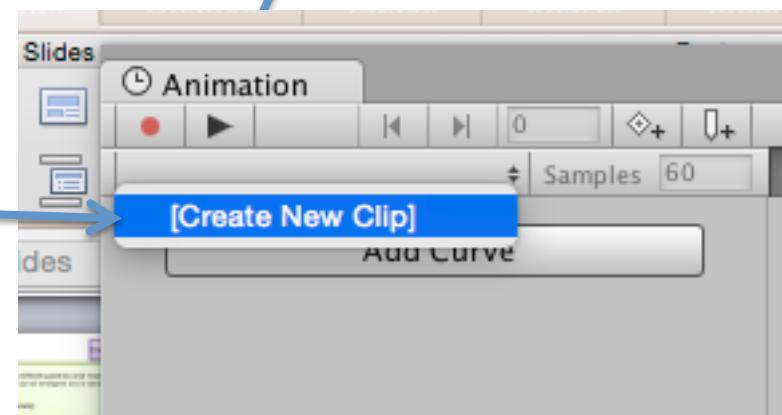
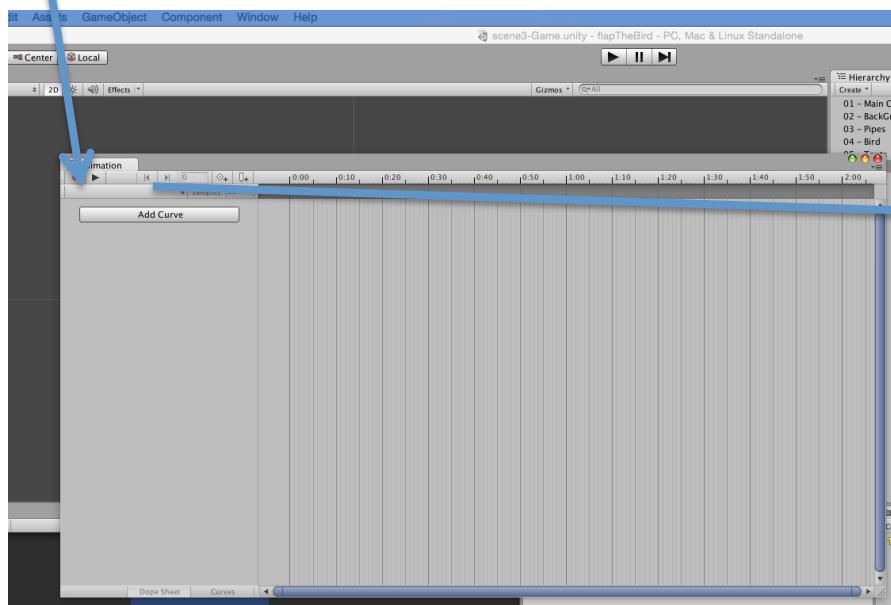
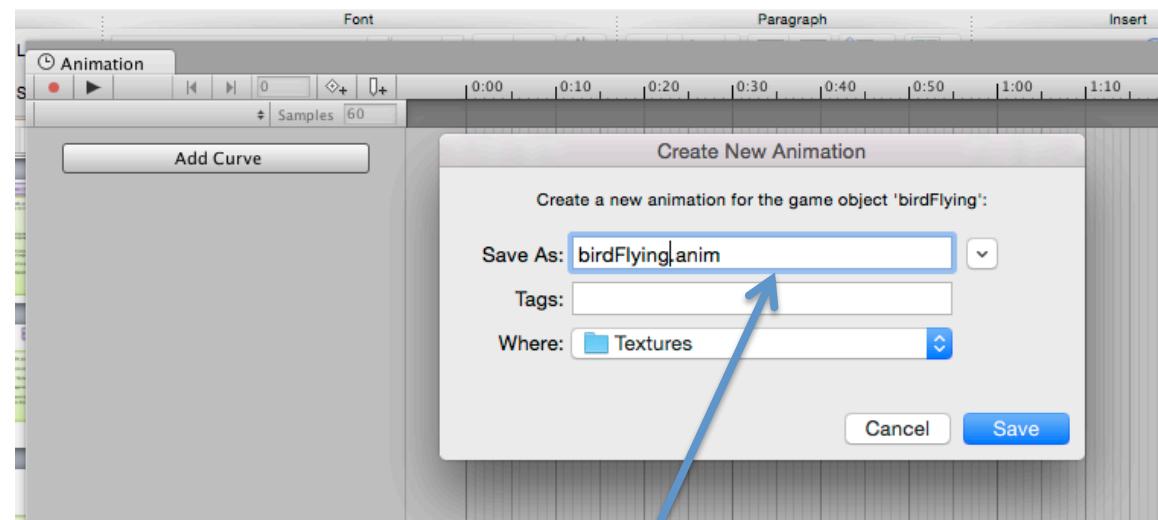
## Etape 13 – Crédit de la texture de l'oiseau

Créer un gameObject de type sprite2D choisir comme texture bird1, sélectionner le  
Dans le menu principal de Unity, Window/Animation  
L'éditeur d'animation s'ouvre



## Etape 14 – Animer l'oiseau (1)

Créer un clip d'animation  
Nommer le birdFlying



## Etape 14 – Animer l'oiseau (2)

Glisser déposer la texture bird1 au temps 0.00

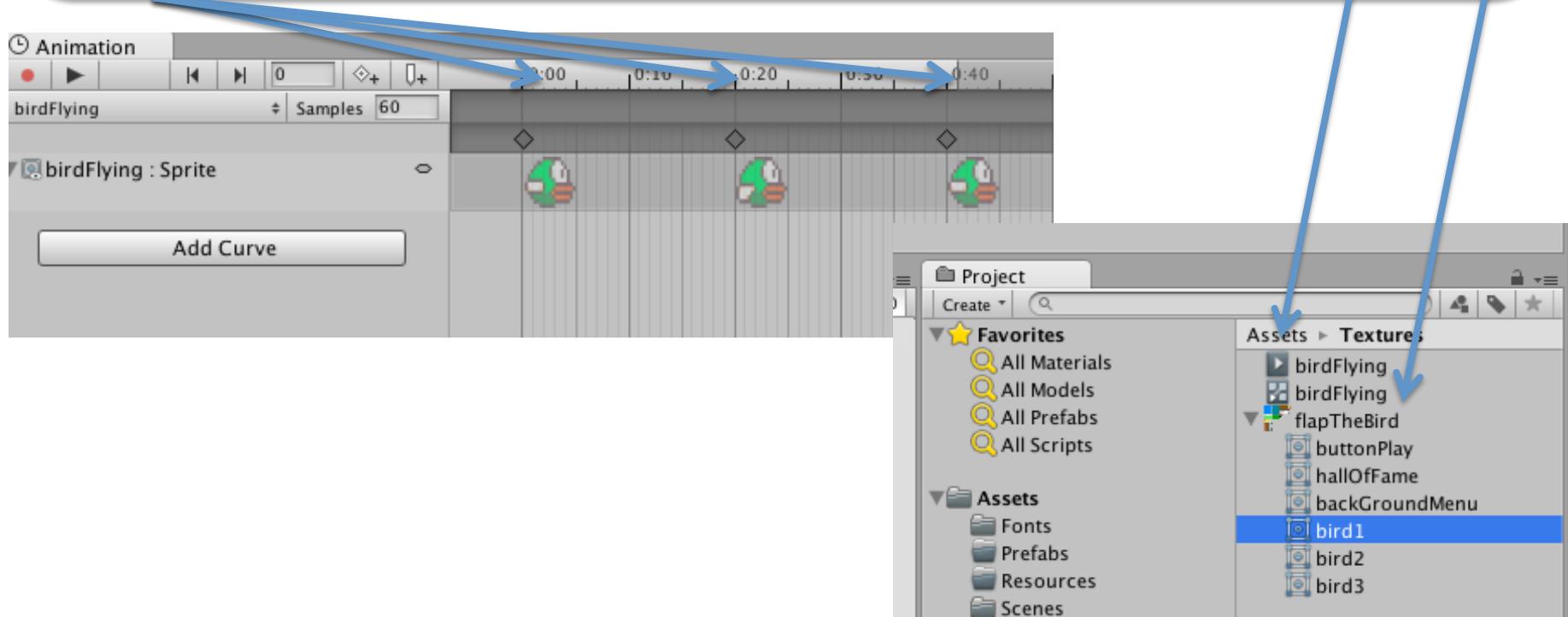
Glisser déposer la texture bird2 au temps 0.20

Glisser déposer la texture bird1 au temps 0.40

Appuyer sur play et la barre rouge de temps défile sur l'éditeur d'animation

Sortir de l'éditeur d'animation

Deux nouveaux objets se sont créés dans texture, un type animator et un objet de type controller



## Etape 15 – Finir la création du sprite de l'oiseau (1)

Sélectionner le sprite birdFlying, dans l'inspector le composant Animator s'est ajouté

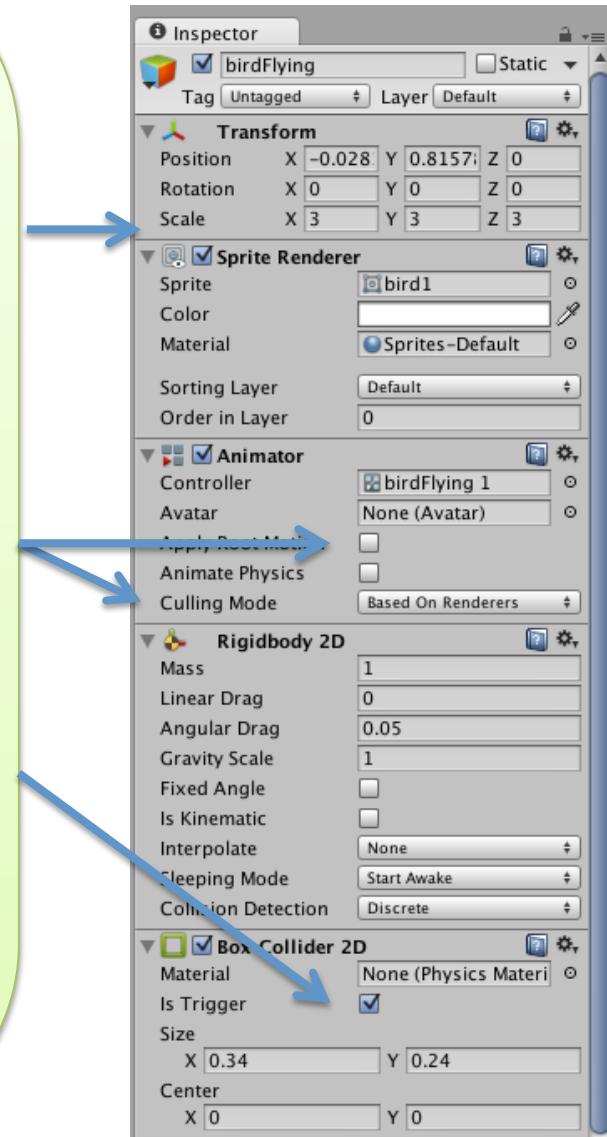
Changer dans le transform.Scale le facteur de zoom, passer le de 1 à 3

Appuyer sur play et le sprite de l'oiseau s'anime dans la fenêtre game, sortir du mode play

Ajouter lui un composant rigidbody2d et une box collision2d  
Dans le composant animator décocher ApplyRootMotion,  
passer le culling mode en Based on Renderers

Ne pas oublier de cocher Is Trigger dans le boxcollider2d

Glisser déposer le sprite birdFlying dans le gameObject 04-Bird



## Etape 15 – Finir la création du sprite de l'oiseau (2)

Créer un script touchAction qui a chaque appui sur la touche espace ou sur l'écran tactile fait subir à l'oiseau une impulsion vers le haut ...

Quel est le composant qu'il faut utiliser ici ?

Est ce que le composant rigidBody2D et l'utilisation de la velocity est suffisant ?

Pourquoi pour ce sprite la propriété de gravité a-t-elle été conservée ?

Penser à mettre les propriétés de vitesse utilisées en public pour trouver la bonne valeur en faisant les tests sous éditeur, le travail sera ainsi facilité

## Etape 16 – Création des obstacles - Pipes (2)

A l'aide de l'éditeur de sprites, créer deux textures avec les pipes vertes

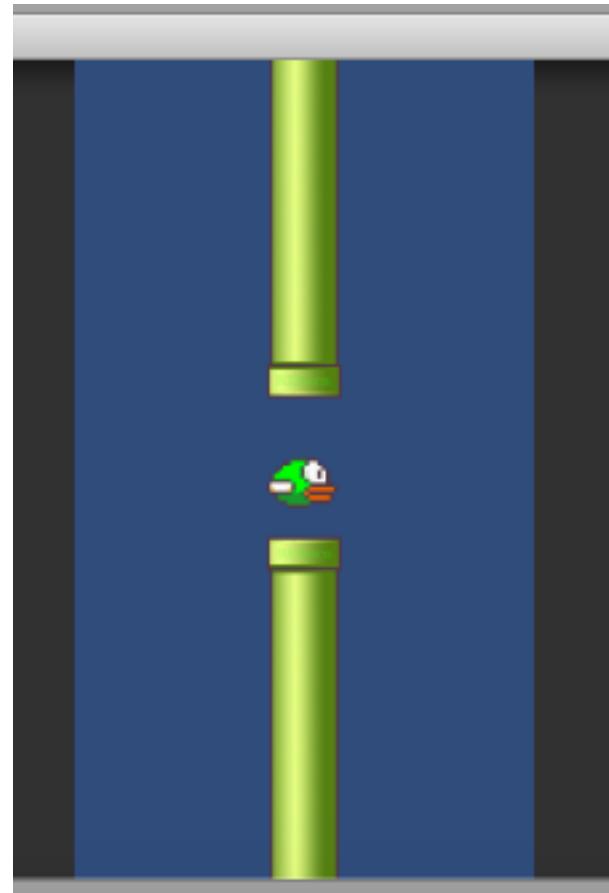
Créer deux gameObject de type sprite avec les textures créées (downPipe, upPipe)

Changer le facteur de zoom, passer le de 1 à 2 pour le X et de 1 à 3 pour le Y, de 1 à 2 pour le Z

Placer les au centre de l'écran

Ajouter à chacun de ces sprites les composants:  
rigidBody2d en activant la propriété is Kinematic (elle exclue le sprite des interactions avec la physique des autres composants)  
boxCollider2d

Créer un gameObject vide pipesPair1, déposer les deux sprites dans ce gameObject



## Etape 16 – Création des obstacles - Pipes (2)

Créer un script movePipes avec:

- une variable public movement de type Vector2
- deux variable public pipe1Up et pie1Down de type GameObject
- deux variables private pipe1UpOriginalTransform et pipe1DownOriginalTransform de type Transform

Ce script déplace les 2 pipes de droite vers la gauche, quand les pipes sortent de l'écran, ils sont replacés à la droite de celui-ci avec une nouvelle position en Y.

Dans le start sauver le transform de pipe1Up et pipe1Down dans pipe1UpOriginalTransform et pipe1DownOriginalTransform.

Quand les pipes sortent de l'écran, calculer la nouvelle position des pipes en utilisant pipe1UpOriginalTransform et pipe1DownOriginalTransform (uniquement pour la position en Y ...)

## Etape 16 – Création des obstacles - Pipes (2)

Attacher ce script à pipePair1, faire glisser downPipe1 et upPipe1 dans les variables public du script dans l'inspector

```
void Update()
{
    pipe1Up.rigidbody2D.velocity = movement; // Déplacement du pipe haut
    pipe1Down.rigidbody2D.velocity = movement; // Déplacement du pipe bas
    siz.x = pipe1Up.GetComponent<SpriteRenderer> ().bounds.size.x; // Récuperation de la taille d'un pipe
    siz.y = pipe1Up.GetComponent<SpriteRenderer> ().bounds.size.y; // Suffisant car ils ont la même taille
    // Le pipe est sorti de l'écran ? Si oui appel de la méthode moveToRightPipe
    if (pipe1Up.transform.position.x < leftBottomCameraBorder.x - (siz.x / 2)) moveToRightPipe();
}

void moveToRightPipe(){
    float randomY = Random.Range (1,4) - 2; // Tirage aléatoire d'un décalage en Y
    float posX = rightBottomCameraBorder.x + (siz.x / 2); // Calcul du X du bord droite de l'écran
    // Calcul du nouvel Y en reprenant la position Y d'origine du pipe, ici le downPipe1
    float posY = pipe1UpOriginalTransform.position.y + randomY;
    // Création du vector3 contenant la nouvelle position
    Vector3 tmpPos = new Vector3 (posX, posY, pipe1Up.transform.position.z);
    // Affectation de cette nouvelle position au transform du gameObject
    pipe1Up.transform.position = tmpPos;

    // Idem pour le second pipe
    posY = pipe1DownOriginalTransform.position.y + randomY;
    tmpPos = new Vector3 (posX, posY, pipe1Down.transform.position.z);
    pipe1Down.transform.position = tmpPos;
}
```

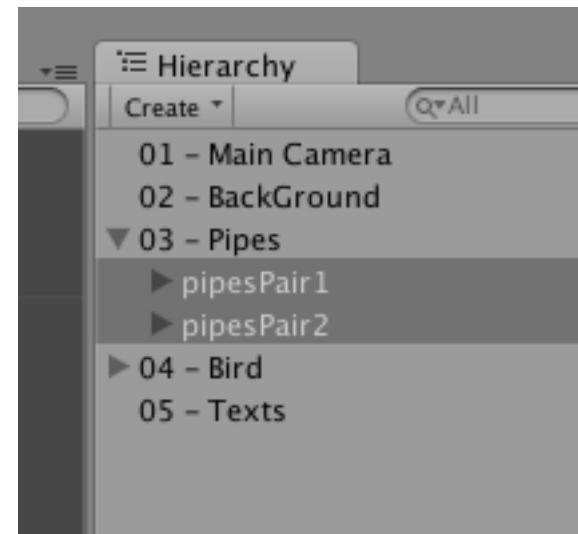
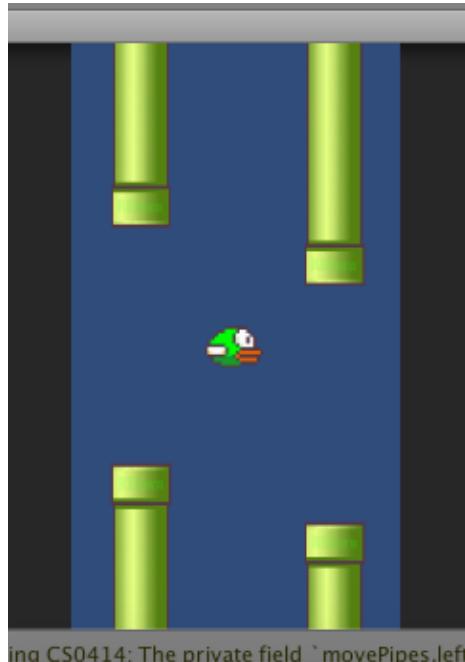
## Etape 16 – Création des obstacles - Pipes (3)

Copier pipersPair1 dans l'onglet Hierarchy et renommer la copie en pipesPair2

Placer le contre le bord droit de l'écran dans l'éditeur de Unity

Glisser les deux piperPair dans le gameObject vide 03-Pipes

Tester, régler les positions et la vitesse plus finement si nécessaire



## Etape 17 – Crédit d'un fond qui scroll

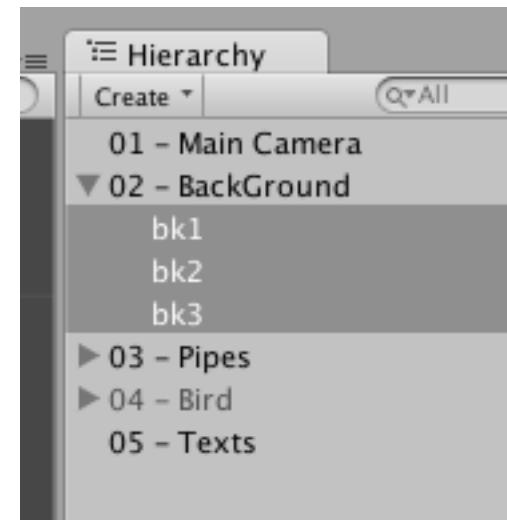
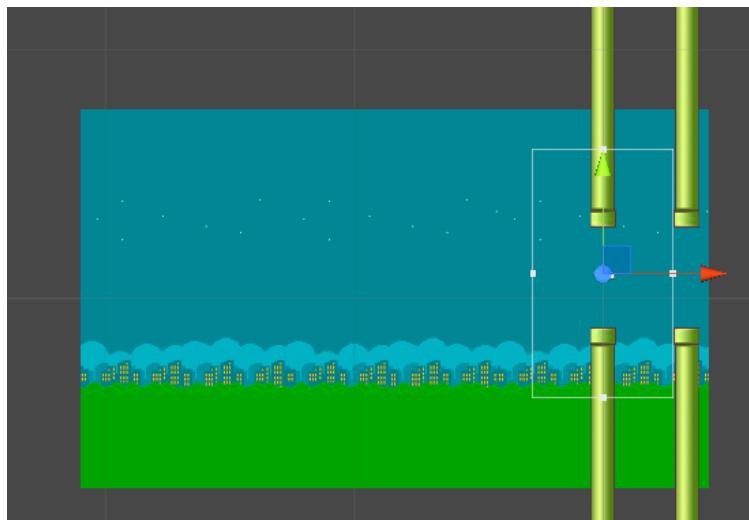
Créer une texture avec le fond nuit dans la planche de sprites

Créer 3 gameObject de type sprite nommés bk1, bk2, bk3, avec le composant rigidbody2d (is Kinematic coché), glisser les dans 02-Background

Poser les 3 sprites les uns après les autres (comme le fond du menu) mais cette fois en partant de la gauche

Créer un script (comme pour le fond du menu) moveBkR qui scroll chaque sprite de gauche vers la droite puis le repose à la place la plus à gauche.

Est-il possible de ne créer qu'un seul script pour scroller dans les deux sens avec une simple option ?..



## Etape 18 – Gestion des collisions (1)

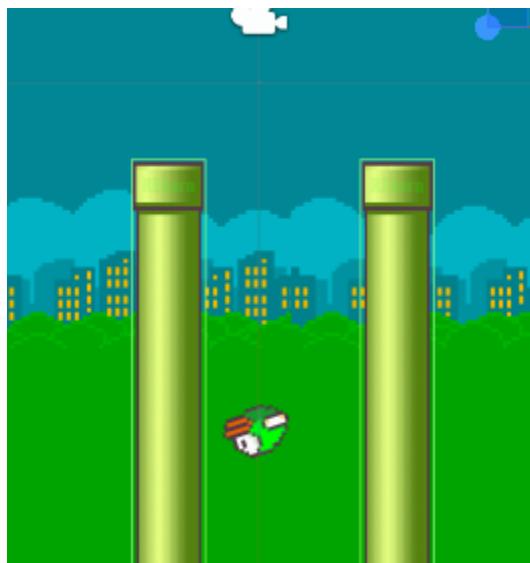
Créer un script gérant la collision de l'oiseau avec les pipes

Nommer le collideManagementBird

Créer un autre script endAction qui sera appelé quand une collision se produit. Celui-ci sera attaché à l'oiseau lorsqu'une collision sera détectée ...

endAction a pour rôle:

- de détruire le script touchAction pour couper toutes les interactions avec le jeu
- de faire tourner l'oiseau sur lui-même
- de tester qu'il sort du bas de l'écran
- de lancer la scène de fin



## Etape 18 – Gestion des collisions (2)

Le joueur marque 1 point à chaque fois qu'il passe entre deux pipes.

Comment gérer cela ?

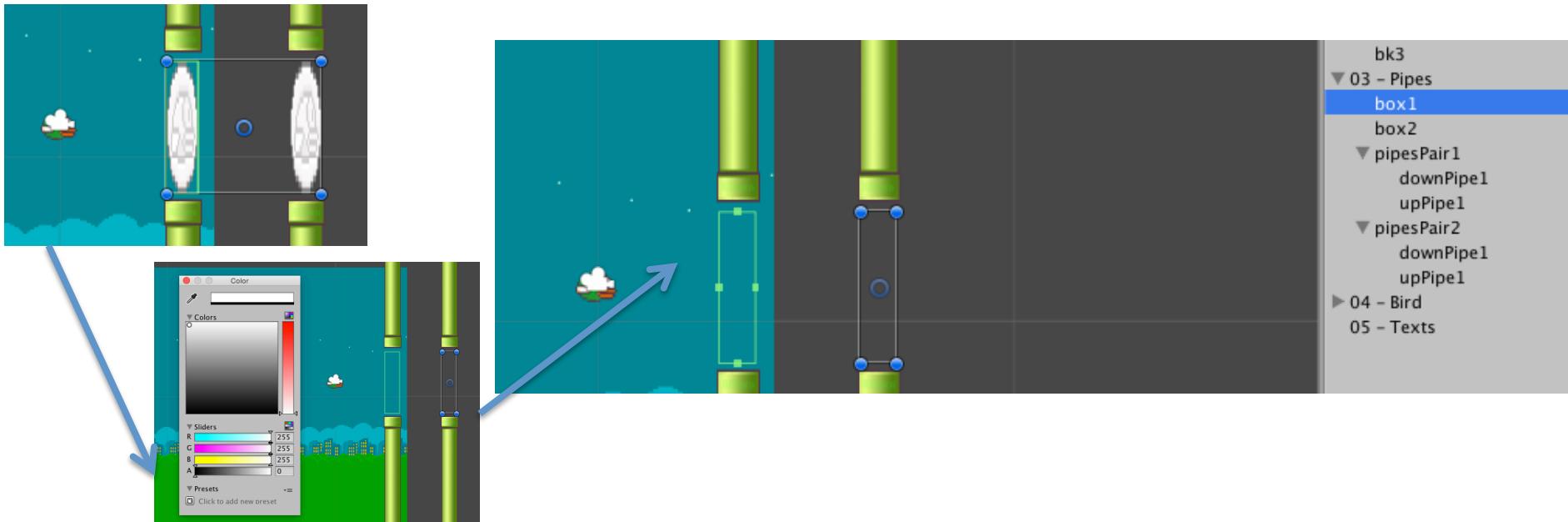
Créer un sprite avec une texture prise au hasard dans la planche de sprite

Placer le au centre de deux pipes, dans le sprite renderer, changer la valeur de A de 255 à 0, le sprite est maintenant transparent.

Ajouter lui les composants habituels rigidbody2d et boxcollider2d

Attacher un script de mouvement comme pour les pipes, faites boucler

Tester si l'oiseau rende en collision avec la hit box de ces sprites transparents, et incrémenter les points



## Etape 19 – Ajout du score (voir TD)

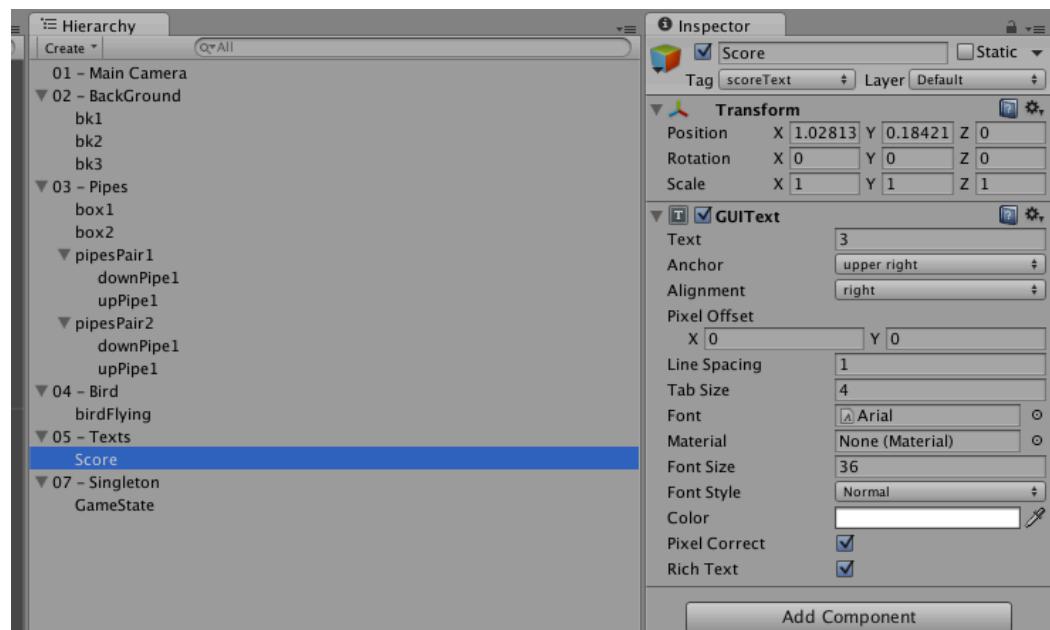
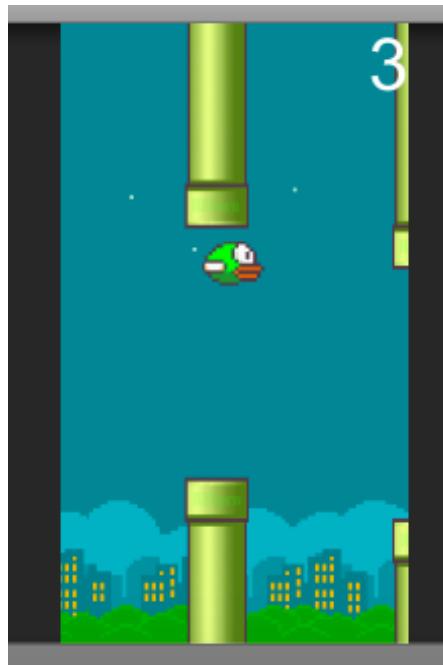
Créer un gameObject de type guiTExt (si non présent créer un gameObject vide et ajouter un composant de type UiText), nommer le Score, glisser le dans 05 – Texts

Placer le en haut à droite, penser à changer ses valeurs Anchor, size comme ci-dessous

Créer un script sous la forme d'un singleton, nommer le GameState (comme pour le TD)

Ajouter l'appel à la méthode addScore du singleton dans le script CollideManagementBird

Le score est maintenant visible et évolue à chaque fois que l'oiseau passe entre deux pipes.



## Etape 20 – Améliorations possibles pour le jeu

Dans le cadre du projet gérer obligatoirement:

- pendant le jeu l'oiseau ne doit jamais sortir de l'écran
- ajouter le son (comme vu en TD)

Dans le cadre du projet il est possible d'ajouter:

- L'inclinaison de l'oiseau quand il monte ou descend
- un fond qui change aléatoirement à chaque redémarrage de partie
- des nuages qui viennent s'jouter aux décors
- des bonus à ramasser par le joueur
- d'inverser le sens de défilement pendant la partie pour corser la difficulté
- etc ...

**Scene 4**  
**Game Over**

## Etape 21 – Ecran de fin ...

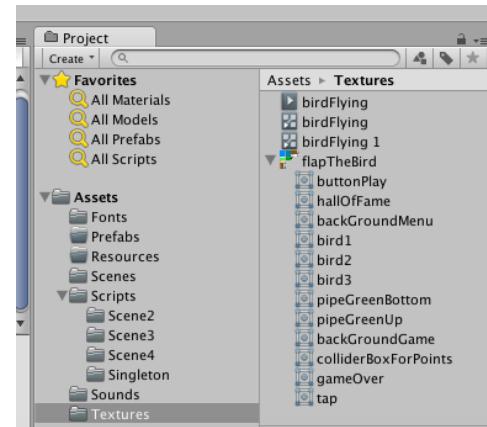
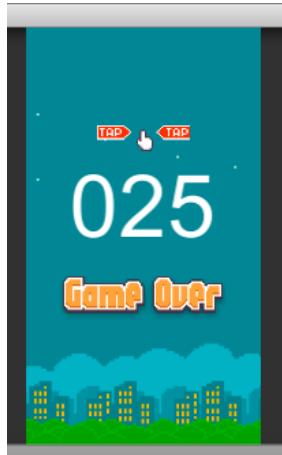
Créer une scène composée:

D'un fond qui scroll

D'un texte qui affiche le score obtenu par le joueur

Deux sprites (comme ci-dessous)

Quand le joueur appui sur espace, ou sur la souris, ou sur l'écran, il revient sur le menu d'origine



## Etape 20 – Améliorations possibles pour l'écran de fin

Dans le cadre du projet il est possible d'ajouter:

- Des sprites qui bougent
- Un texte qui s'anime
- Une sauvegarde du meilleur score ...
- Un partage sur FaceBook du score obtenu
- La possibilité de mettre son nom et de sauver le tout
- Etc ...

## Scene 1

### Initialisation

## Etape 21 – Et la scene 1 ? Les initialisations ...

Créer une page avec une image de fond et un logo (2 sprites au total)

Créer un script nommer wait2Sec qui utilise la méthode Invoke. Elle permet de lancer une méthode après un laps de temps déterminé.

Ainsi la scene du menu sera lancé après 2 secondes, voici le update et la méthode en question:

```
void Update () {  
    Invoke ("goScene2",2.0f);  
}
```

```
void goScene2(){  
    Application.LoadLevel("Scene2-Menu");  
}
```

## Etape 21 – Et la scene 1 ? Les initialisations ...

Créer des gameObjects vides dans toutes les scenes et attachés les singlenton nécessaire au jeu, au moins deux, le GameState et le gestionnaire de son.

Attention ce transfert ne se fera pas sans modification du code de certaines méthodes des singlenton ... En effet il va falloir, par exemple, tester qu'une instance n'est déjà pas présente et le code du cours doit être aménagé.

Mettre à jour l'ensemble des lignes de code permettant le passage d'une scene à l'autre en utilisant:

À Application.LoadLevel(nom de la scene).

Penser à remettre le score à zéro à chaque passage par la scene 2

Maintenant il est temps de compiler pour la première fois le tout pour un site web ...

Compilation  
Pour le Web ...

## Etape 22 – Compilation pour le Web ... (1)

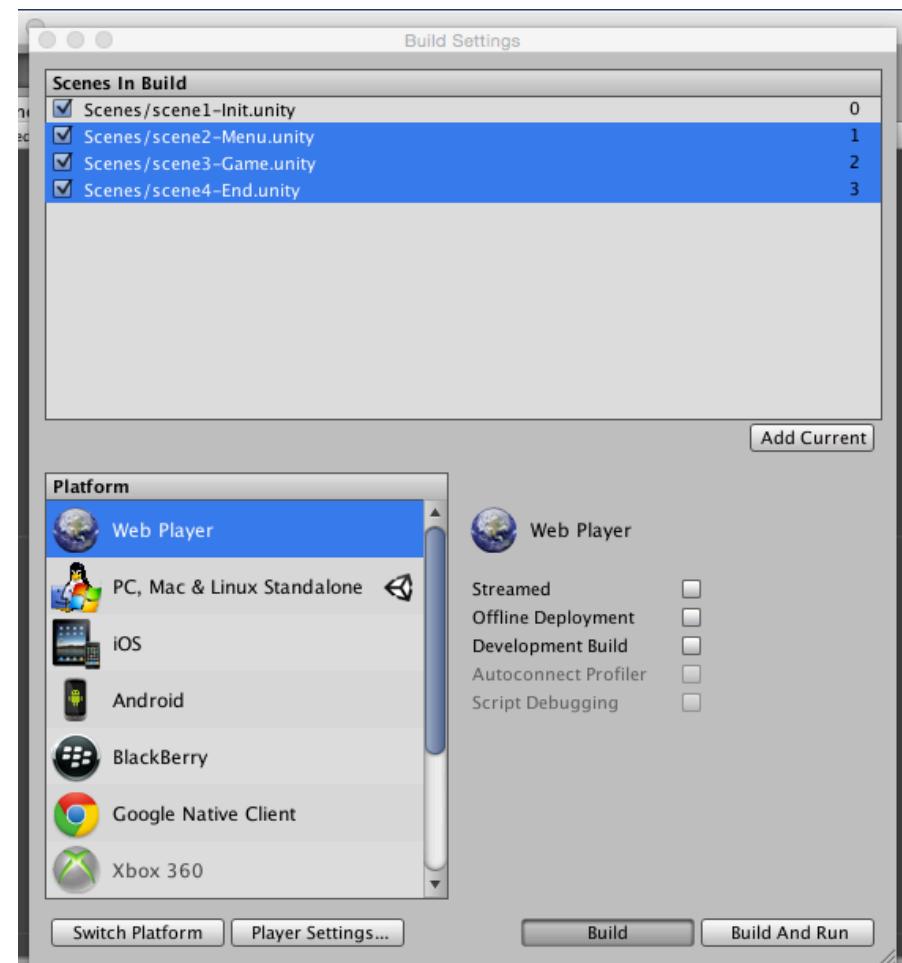
Aller dans File, sauver le tout.

Ensuite, File/BuildSettings/

Glisser déposer les scenes dans l'ordre de leur apparition dans le jeu

Choisir ensuite Web Player

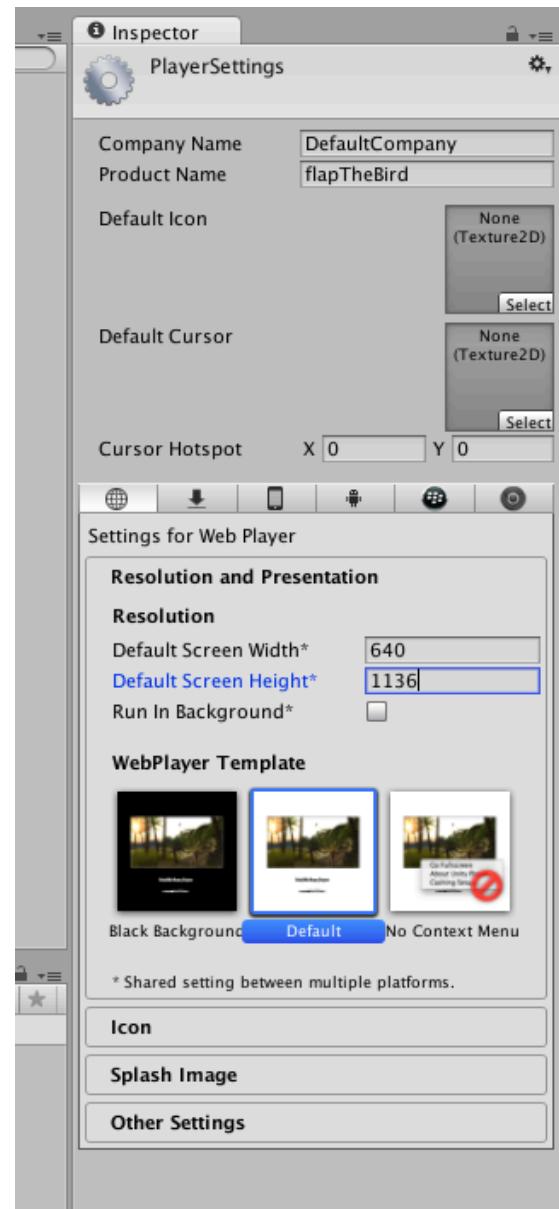
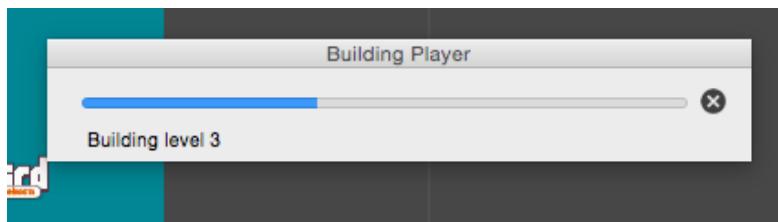
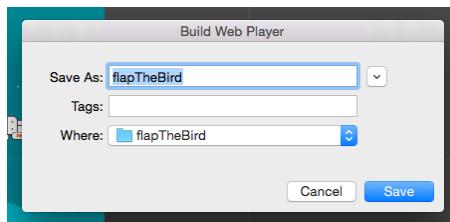
Aller dans player settings



## Etape 22 – Compilation pour le Web ... (2)

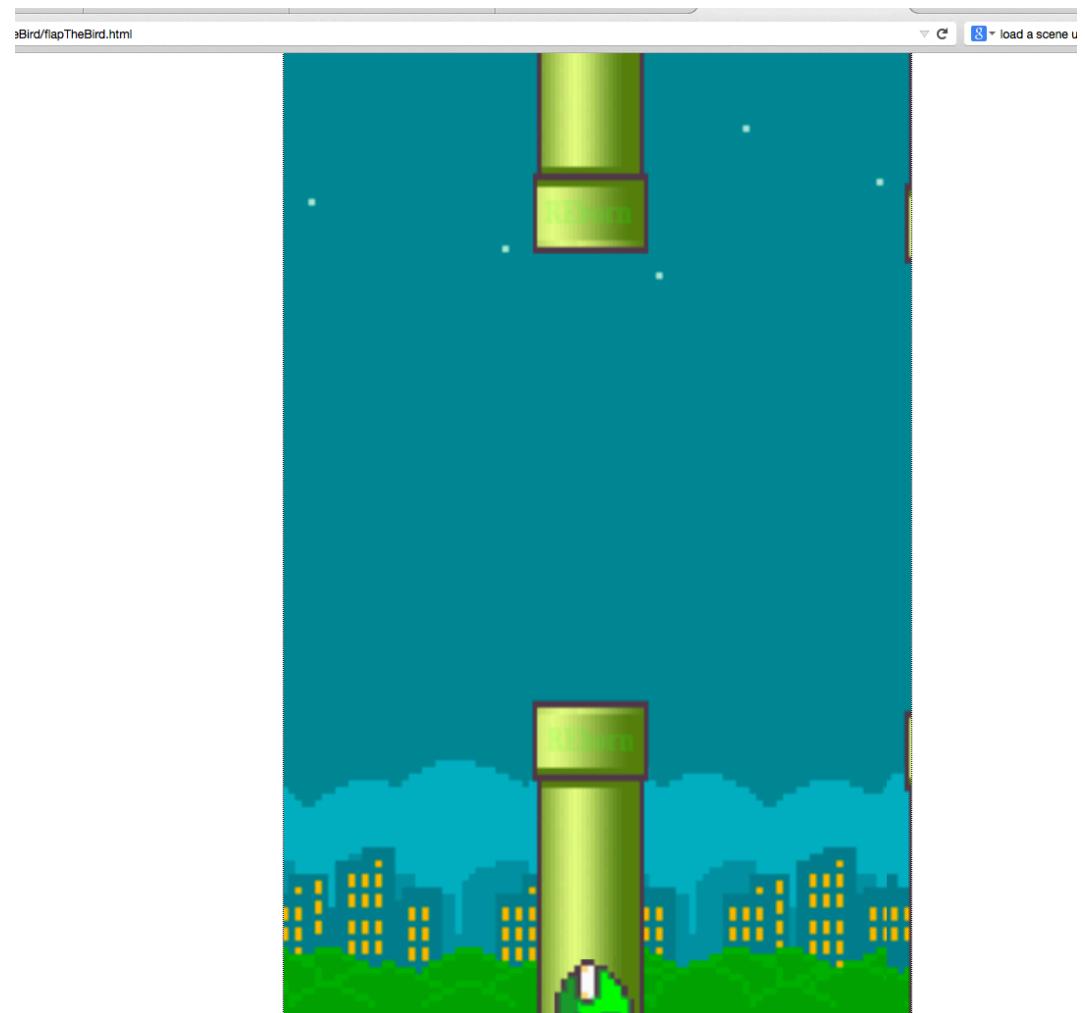
Dans l'inspector, changer la résolution comme sur l'image

Changer d'autres settings si désirer notamment le logo ou encore le fond de couleur de la page web qui va accueillir le jeu, enfin compiler



## Etape 22 – Compilation pour le Web ... (3)

Et voilà sous Firefox ...



Quelques erreurs à éviter:

Penser à transformer les textures en mode sprite

Vérifier que tous les singltons sont bien présents dans chaque scène

Détacher les scripts d'intéraction avec l'utilisateur quand ils ne sont plus nécessaire

Ranger l'ensemble des données (sons, scripts etc...) dans les dossiers adéquats ...

Respecter bien le nom des classes ou des méthodes, les erreurs de compilations les plus courantes sont souvent liées à l'inattention

Le jeu est jouable sous web à cette adresse:  
[chuckdune.free.fr/flapTheBird.html](http://chuckdune.free.fr/flapTheBird.html)

Ce projet est maintenant terminé.

Le plus important c'est d'apprendre, d'être curieux de tout, aussi ce TP ne doit pas être considéré comme une fin en soi mais bien comme le début de quelque chose.

L'imagination n'a pas de limite et les joueurs sont insatiables ...

Bon jeu et bon Unity ...