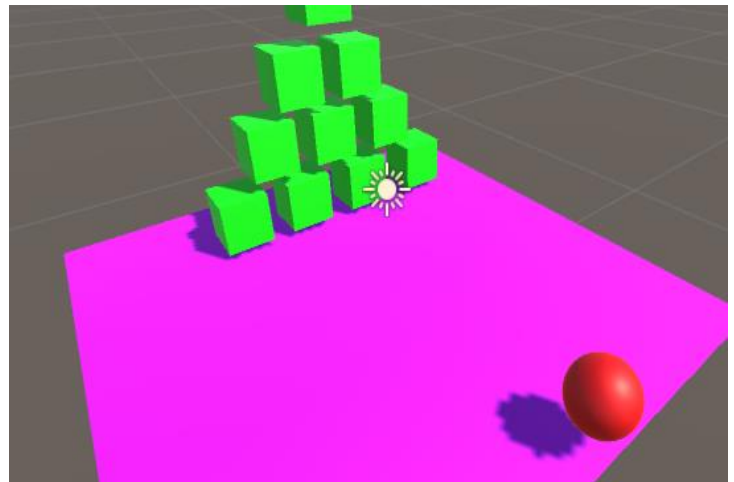
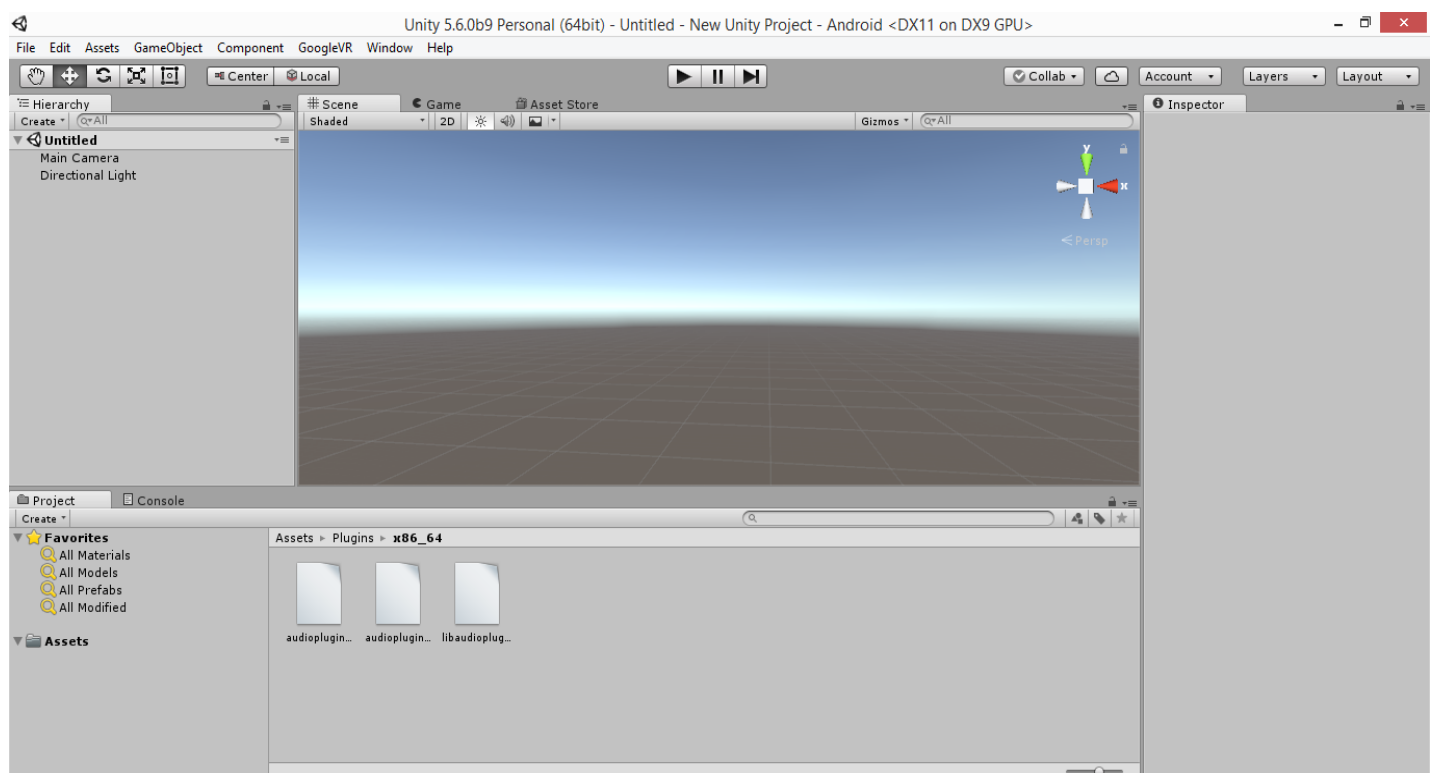


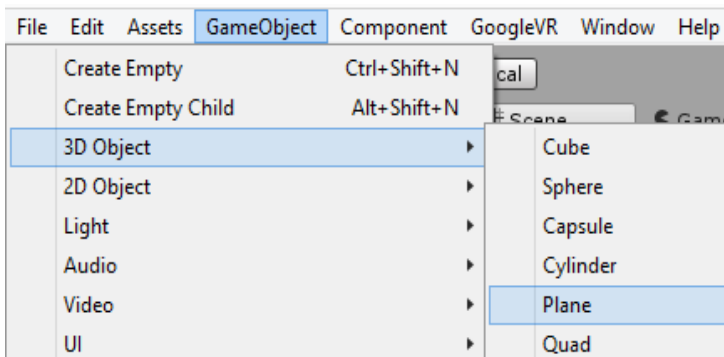
Ce chapitre sur Unity3D va commencer par deux fiches, la première où nous créerons un jeu qui sera exportable sur n'importe quel système d'exploitation (Windows, Android, navigateur internet, Facebook...) ensuite dans une seconde fiche, nous verrons comment l'exporter plus précisément sur Android (il faudra vraiment s'y prendre à l'avance...).

Le premier jeu sera un « chamboule tout » formé de cubes que nous essayerons de faire tomber grâce à une balle que nous dirigerons grâce au clavier ou à une manette pour la version windows ou directement en tactile pour la version Android.



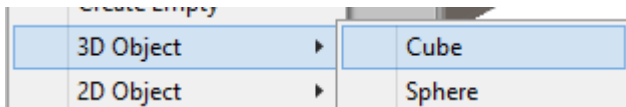
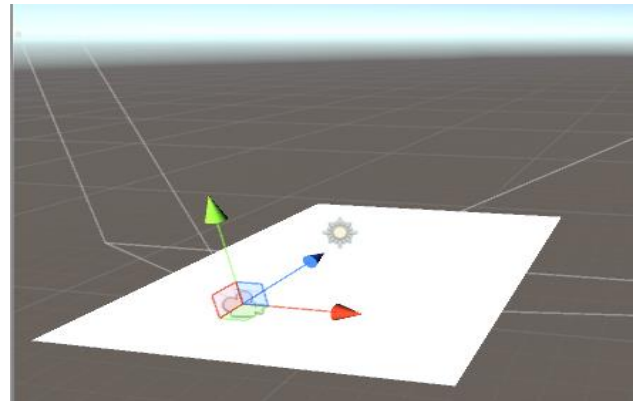
Commencez par ouvrir Unity puis créez un nouveau projet 3D avant d'arriver sur la fenêtre ci-dessous :



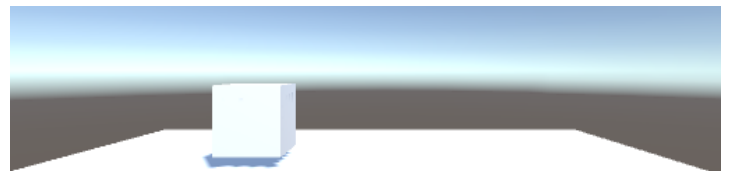


En réalisant un clic droit dans la fenêtre Hierarchy ou en allant dans le menu GameObject, vous allez insérer une surface plane pour pouvoir y poser notre chamboule tout.


Une fois la surface plane insérée, vous pouvez placer la Main Camera sur le dessus de manière à ce que vous puissiez voir correctement dans le Camera Preview.

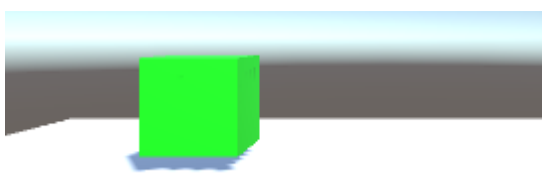
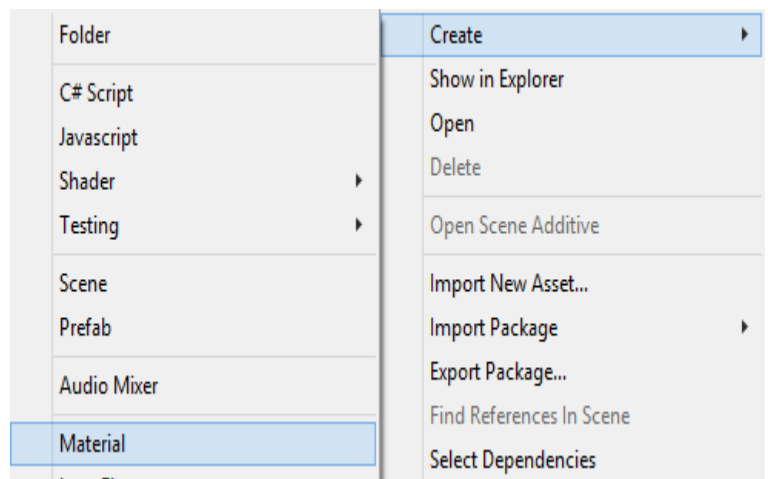


Vous pouvez à présent insérer un premier cube et le placer sur la surface dans l'axe de la caméra.

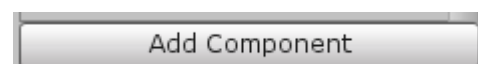


On peut le voir grâce à son ombre mais ce serait plus facile s'il possédait une couleur.

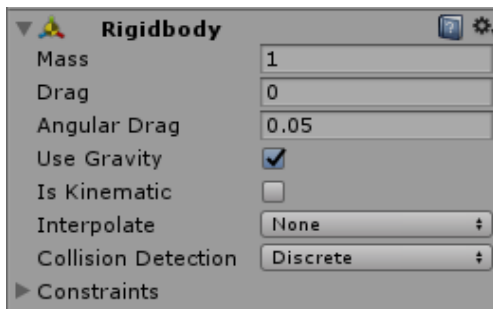
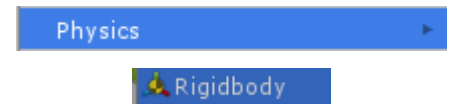
Pour cela, placez-vous dans la partie Assets de la fenêtre Project, réalisez un clic droit et choisissez Create puis Material. Donnez lui un nom simple mais reconnaissable comme Cube_Color par exemple. Vous verrez alors en haut à droite, dans la fenêtre Inspector, un petit rectangle blanc  En cliquant dessus, vous pourrez choisir une couleur et une fois cela réalisé, vous n'aurez qu'à faire glisser « Cube_Color » directement sur votre Cube soit dans la fenetre Hierarchy soit dans la fenêtre Scene.



Votre premier cube est presque prêt, il faut maintenant lui définir les propriétés « physiques ». Pour cela tout en bas de la fenêtre Inspector, cliquez sur :



Choisissez Physics puis Rigidbody. Une nouvelle partie apparaît dans la fenêtre. Vous pourrez régler la masse du cube et surtout si vous souhaitez que la gravité s’y applique.



Attention si vous cochez ou modifiez :

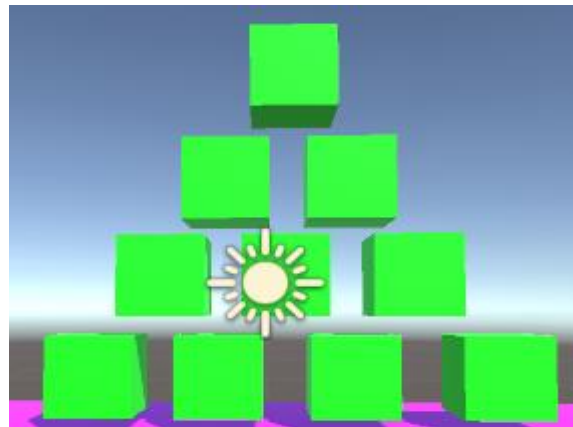
« Drag » (glisser) : Indique la résistance à l’air lorsque l’on pousse ou tire un objet.

« Angular Drag » (glisser par torsion) : Indique la résistance à l’air lorsque l’on applique une torsion à l’objet.

« Is Kinematic » la physique du jeu ne s’appliquera pas à cet objet.

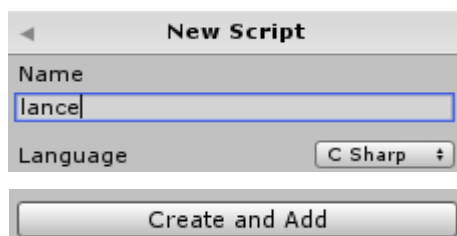
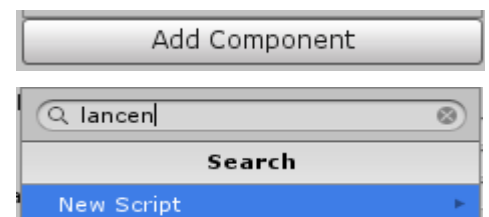
Si vous choisissez 0, sur la force considérée sera nulle.

Votre premier cube est prêt, vous pouvez le cloner et les déplacer de manière à obtenir la forme souhaitée.



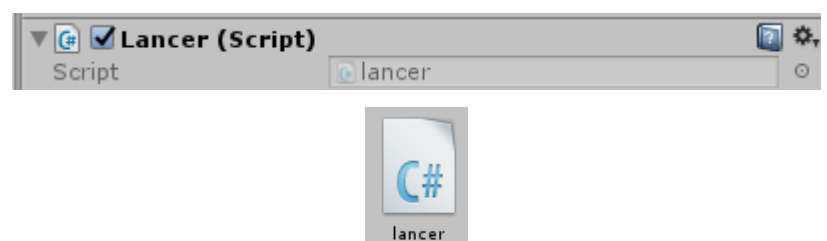
Passons à présent à la réalisation du « Chambleur », cette sphère qui détruira tout sur son passage. Cette étape est plus stratégique puisqu’elle va au-delà de placer une simple balle sur la surface. En effet, il va falloir réaliser un petit code pour expliquer à Unity quoi faire de cette balle.

Une fois la balle créée et placée, vous pouvez ajouter un composant (Add Component), tapez dans la barre de recherche le nom de votre script qui dirigera la balle puis cliquez sur « New Script ».



Validez le langage, ici nous préférons le C Sharp aussi noté C# mais si vous avez des notions de Javascript, vous pouvez les utiliser ici.

Une fois que vous cliquez sur « *Create and Add* », vous verrez apparaître un nouveau composant dans l’inspecteur de votre balle ainsi qu’un nouveau fichier dans la partie *Asset* de votre projet.



Pour lancer le logiciel qui vous permettra de modifier le script (*MonoDevelop*) il suffit de cliquer sur le nom de votre script dans la ligne Script. Une nouvelle fenêtre s'ouvre alors avec un début de code qu'il ne faudra pas supprimer mais juste agréementer suivant vos besoins.

```
No selection
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class lancen : MonoBehaviour {
6
7     // Use this for initialization
8     void Start () {
9
10    }
11
12    // Update is called once per frame
13    void Update () {
14
15    }
16 }
17
```

Si vous souhaitez définir des variables que vous pourrez modifier depuis l'*Inspector* de Unity, il vous faudra déclarer des variables publiques dans votre class :

```
public float speed=5f;
```

(exemple d'une variable décimale de vitesse)

Si vous souhaitez réaliser quelque chose au lancement de votre script, il vous faudra le placer dans la fonction (*void*) Start. Si par contre vous souhaitez réaliser une action, une vérification (...) à chaque image créée dans votre jeu, vous devrez la placer dans la fonction Update.

Voyons un peu différentes parties de codes qui nous servirons par la suite avant de les mettre en place au sein de notre code :

On définit la variable run comme étant un entier de valeur 0. Si le joueur venait à appuyer sur la touche Fire1, qui est d'origine dans Unity, le bouton gauche de la souris, la touche Ctrl de gauche, ou le fait d'appuyer sur l'écran tactile, alors celle-ci prendrait la valeur 1.

```
public int run = 0;
```

```
if (Input.GetButtonDown ("Fire1")) {
    run = 1;
}
```

Si la valeur de run est à 1 alors

transformer la position de la balle en y

```
if (run == 1) {
```

ajoutant un nouveau vecteur à 3 dimensions où x=0 , y=0 et où z est

```
transform.position += new Vector3 (0f,0f,speed*0.1f);
```

10% de la vitesse indiquée plus haut.

Pour finir en matière de code, voyons la seule différence qu'il y aura entre le programme que nous compilerons pour Windows et celui pour Android.

Pour Windows, on définit une variable locale h qui aura la valeur de l'axe horizontal du joystick ou des flèches directionnelles puis on applique une transformation suivant l'axe x pour que la balle aille vers la gauche ou vers la droite.

```
float h = Input.GetAxis("Horizontal");
```

```
transform.position += new Vector3 (h*0.1f,0f,0f);
```

Comme nous travaillerons pour Android dans la prochaine fiche, il est important de préciser dès maintenant la ligne de code à ajouter et qui nous permettra de connaître les modifications de l'accéléromètre du smartphone ou de la tablette utilisée.

```
transform.position += new Vector3 (Input.acceleration.x*0.1f, 0f, 0f);
```

Le code complet et commenté est présenté ci-dessous mais il est également téléchargeable à l'adresse :

[https://github.com/labomaillo1/Niveau2/blob/master/Unity Chamboule/lancer.cs](https://github.com/labomaillo1/Niveau2/blob/master/Unity%20Chamboule/lancer.cs)

Le code intégral :

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class lancer : MonoBehaviour {
    //Variable qui définit la vitesse de la balle
    public float speed=5f;
    //Variable qui définit si l'utilisateur a appuyé sur Fire1
    public int run = 0;
    //Variable qui regroupe les caractéristiques d'interaction de la sphere.
    public Rigidbody sp;
    //On définit tous les cubes

    // Use this for initialization
    void Start () {
        sp = GetComponent<Rigidbody>();
        transform.position = new Vector3 (0f, 1.8f, -4.66f);
    }

    void Update () {

        //Permet de déplacer la balle parallèlement à la camera
        float h = Input.GetAxis("Horizontal");
        transform.position += new Vector3 (h*0.1f , 0f , 0f);

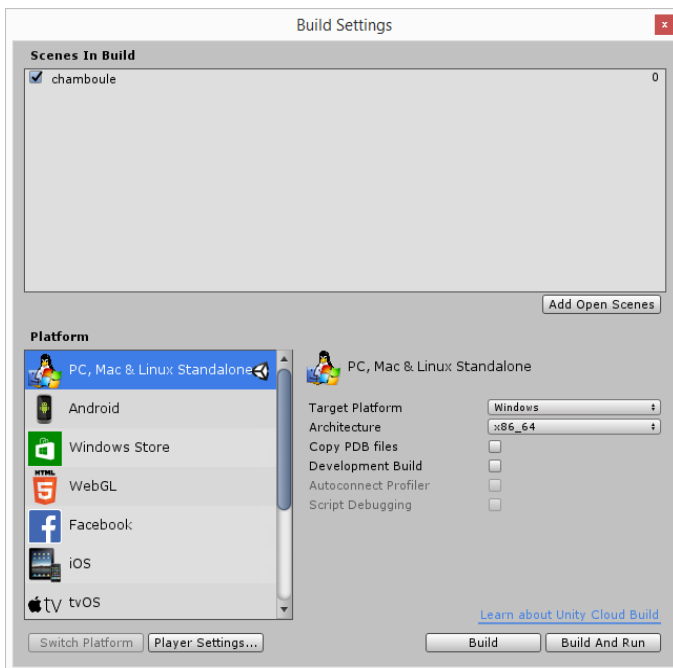
        //Pour l'exporter sur Android
        transform.position += new Vector3 (Input.acceleration.x*0.1f , 0f , 0f);

        //Verifie si on clique Fire1 (Ctrl de gauche ou bouton souris)
        if (Input.GetButtonDown ("Fire1")) {
            run = 1;
        }
        //Si fire1 a été appuyé on lance la balle.
        if (run == 1) {
            transform.position += new Vector3 (0f , 0f , speed*0.1f);
        }
        //Si la balle tombe on réinitialise le tout
        if (transform.position.y < 0) {
            run = 0;
            transform.position= new Vector3 (0f, 1.8f, -4.66f);
            sp.velocity = Vector3.zero;
            sp.angularVelocity = Vector3.zero;
        }
    }
}
```

Maintenant que votre jeu est prêt vous pouvez le lancer en cliquant sur le bouton Play situé au milieu de votre écran :



Nous allons pour finir exporter ce jeu pour pouvoir l'envoyer à vos camarades (et peut-être dans un futur proche réaliser un jeu en réseau ou en Bluetooth). Pour cela cliquez sur *File* en haut à gauche puis sur *Build & Settings*.



La fenêtre qui apparait vous permettra de l'exporter sous différents formats mais ne vous aventurez pas encore dans des formats compliqués comme Android puisqu'il vous sera demandé d'effectuer des téléchargements (~30Go...) et de paramétrer tout cela, ce sera l'objet de la prochaine fiche pour réaliser un jeu simple en réalité virtuelle (VR).

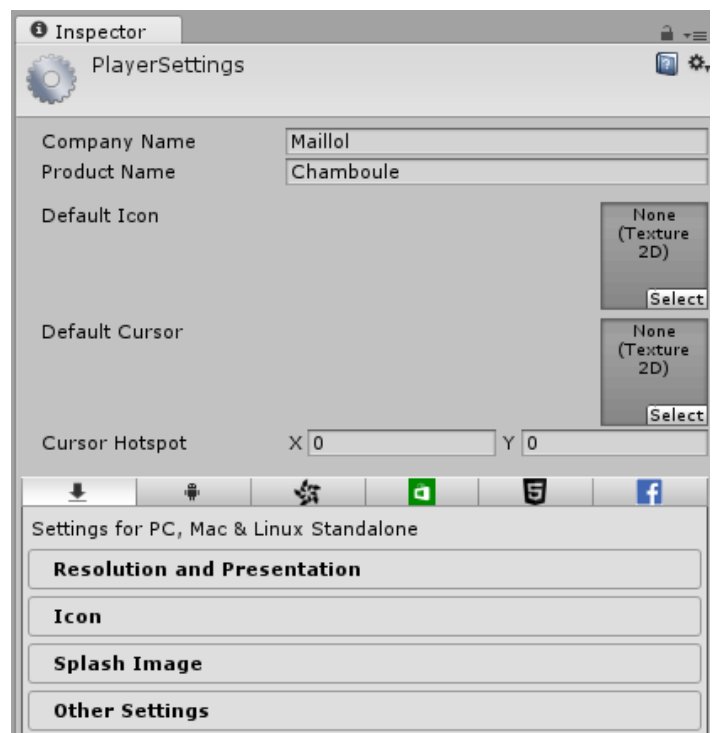
Normalement lorsque vous cliquez sur PC, Mac... vous pourrez choisir si vous l'exporter pour un système d'exploitation 32 (x86) ou 64 bits (x86_64 le plus courant aujourd'hui) mais il est important de paramétrer les informations avant de cliquer sur Build.

Pour cela cliquez sur Player Settings située en bas de la fenêtre afin de voir les informations dans l'Inspector. Vous pouvez choisir le nom de votre société, le nom du jeu, mais aussi une image comme icône, un curseur de souris, la résolution du jeu...

Si vous souhaitez l'exporter sur Mac ou sur Android, il ne faut surtout pas oublier de compléter la ligne

Bundle Identifier `com.Company.ProductName`

Sans quoi vous vous risquez à des erreurs de compilation lors de la création du fichier exécutable. Une fois que tout est prêt, vous pouvez vous lancer et cliquer sur le bouton Build. Choisissez l'emplacement du fichier et laissez travailler votre ordinateur.



Ca y est, vous êtes officiellement un développeur de jeu vidéo, à vous le nouveau blockbuster !