

Table des matières

[Etape 0 : Prérequis 4](#_Toc513044348)

[Etape 2 : Mise en place du projet 5](#_Toc513044349)

[Etape 3 : GUI (Graphical user Inteface) 6](#_Toc513044350)

[Créer une préfab 6](#_Toc513044351)

[Etape 4 : commençons à coder 7](#_Toc513044352)

[Vous allez commercer par TargetController. 7](#_Toc513044353)

[Passons à GameControler 9](#_Toc513044354)

[Bonus exemple de modification : 12](#_Toc513044355)

# Etape 0 : Prérequis

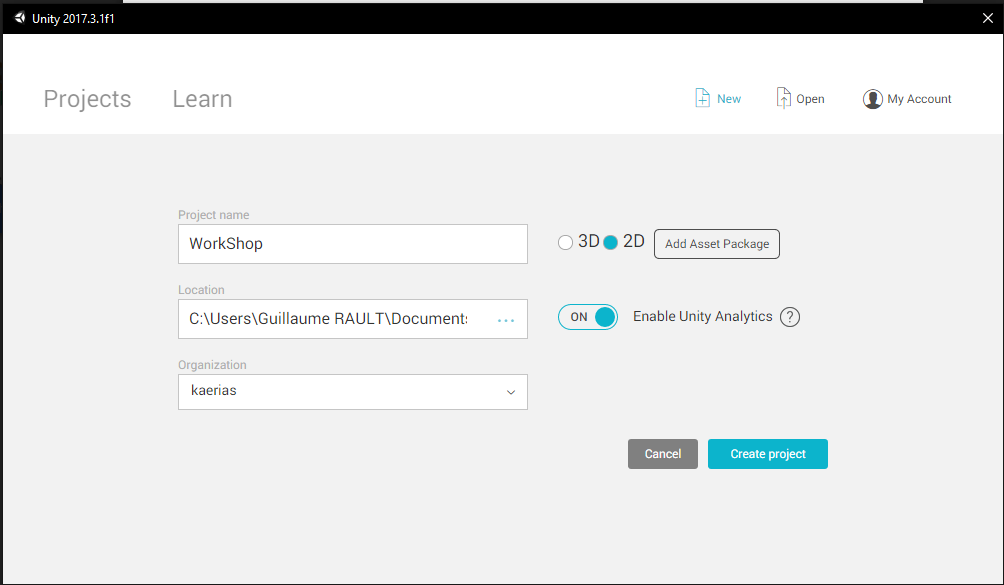
Tout d’abord, nous allons avoir besoin de Unity.

Pour ce qui est de la confection du code, un logiciel type Visual Studio/Monodevelop avec le module Unity installé sera préférable.

# Etape 2 : Mise en place du projet

Commençons par créer un nouveau projet.

- Nom du projet : WorkShop

- Choisissez « projet 2D »

Une fois Unity ouvert sur une nouvelle scène.

Créé 4 dossier dans votre dossier « Assets » :

* \_Save (le « \_ » et pour place votre dossier en haut de votre répertoire)
* Prefabs
* Scripts
* Sprites

Dans votre dossier « \_Save », créé un dossier « Game ».

Refaite de même dans le dossier « Scripts ».

Quand cela est fait, enregistré votre scène dans le dossier \_Save/Game.

# Etape 3 : GUI (Graphical user Inteface)

Avant de commencer à coder, vous allez devoir réaliser 2,3 manipulation sur votre interface pour bien commercer.

## Créer une préfab

Définition :

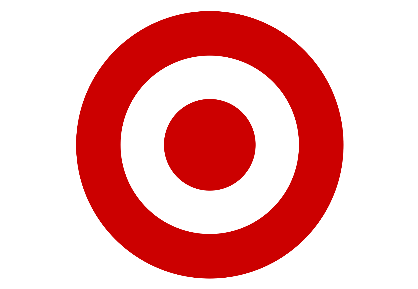
Les Prefabs sont des sortes de templates, des patrons d’objets qui peuvent être réutilisés pour créer d’autres objets similaires.

On les obtient à partir d’un objet modèle que l’on crée puis que l’on fait glisser de la fenêtre hiérarchie vers la fenêtre projet.

On peut ensuite par l’opération inverse créer à l’édition autant d’instances du Prefab que l’on veut.

Utilisation :

Vous allez devoir créé dans un premier temps une préfabs d’une image (.png,.jpg peux importe) que vous avez choisi pour représenter votre cible, personnellement j’ai choisi celle la :



Si votre Préfabs a été correctement réalisé, vous devriez avoir dans votre dossier « Prefabs » un objet comportement le nom de votre image.

Pour Finir sur le GUI, vous pouvez rajouter un arrière plan a votre jeu si l’envie vous en prend. (glisser une image de vos assets a votre hierarchy et faite correspondre votre image avec le champs de vision de votre caméra).

# Etape 4 : commençons à coder

Tout d’abord, nous allons commencer par créer 2 scripts :

* GameController
* TargetController

## Vous allez commercer par TargetController.

Crée 3 variable :

\_speed en float //valeur par défaut : -1

\_direction en int[] //valeur par défaut d’un int : -1

GameBoard en GameObject //utiliser le GameObject.Find sur l’objet « GameBoard »

Dans la fonction « start » initialiser vos variables.

void Start () {

GameBoard = ???;

\_direction = new int[2];

\_direction[0] = ???;

\_direction[1] = ???;

}

Créé 4 fonctions :

* void MoveTarget() ;
* void SetDirection() ;
* void Translated() ;
* private void OnMouseDown() ;

Après les avoir créés, revenez à votre fonction « Update »

Et dans le corps de cette fonction appelez la fonction « MoveTarget »

A ce stade là vos fonction Start et Update sont fini (au passage nous reviendrons sur la fonction « OnMouseDown à la fin du tuto).

Passons aux fonctions que vous avez créé récemment.

Le code de la fonction « SetDirection » étant peu utile je vous l’offre gracieusement.

void SetDirection()

{

if (transform.position.x <= -11 || \_direction[0] == -1)

\_direction[0] = 0;

else if (transform.position.x >= 11)

\_direction[0] = 1;

if (transform.position.y <= -5 || \_direction[1] == -1)

\_direction[1] = 0;

else if (transform.position.y >= 5)

\_direction[1] = 1;

}

Pour réaliser le MoveTarget, vous aurez 3 instructions à réaliser :

* \_speed = speed dans le script GameControllerqui se trouve dans « GameBoard »
* Appeler la fonction « SetDirection »
* Appeler la fonction « Translated »

void MoveTarget()

{

\_speed = ???.Speed;

???;

???;

}

Pour réaliser le MoveTarget, vous aurez 1 instructions à réaliser et à modifier légèrement pour les 3 autres :

* Vous devrez faire : « (la position de l’objet courant + un incrémenteur de 1 dans la direction choisi) \* la vitesse \* le deltaTime ».

void Translated()

{

if (\_direction[0] == 0)

transform.Translate(???);

else if (\_direction[0] == 1)

transform.Translate(???);

if (\_direction[1] == 0)

transform.Translate(???);

else if (\_direction[1] == 1)

transform.Translate(???);

}

## Passons à GameControler

Vous allez devoir d’en un premier temps créer 6 variables :

Bool :

\_endOfGame

\_isShooted

Float :

(public) \_speed

Int :

(public)\_life

GameObject :

(public) \_Target

\_CloneTarget

Dans la Fonction « Start »

Initialiser \_CloneTarget grâce la fonction « Instantiate »

Proto= public staticObject **Instantiate** ([Object](https://docs.unity3d.com/ScriptReference/Object.html) **original**, [Vector3](https://docs.unity3d.com/ScriptReference/Vector3.html) **position**, [Quaternion](https://docs.unity3d.com/ScriptReference/Quaternion.html) **rotation**);

Par défaut, life sera à 5, \_speed à 1, \_isShooted à false.

// Use this for initialization

void Start () {

\_cloneTarget = Instantiate(???) as GameObject;

\_cloneTarget.transform.position = new Vector3(-9, Random.Range(-3.5f, 3.5f), -1);

???  
 ???

???

}

Maintenant créé 2 fonctions :

public void TestShoot()

void IsEndOfGame()

Pour IsEndOfGame :

Si la vie est égale ou inférieure à 0, \_endOfGame est égale true, puis utiliser la fonction « Application.Quit() ».

Sinon \_EndOfGame est égale à false

Pour la fonction « TestShoot »

Vous devrez remplir le premier « if », si \_isShooted est faux et si le clic gauche de la souris est enfoncé.

Proto = Input.GetKeyDown(KeyCode key)) ;

Sinon

La position du gameobject « Target » et égale à une nouvelle position.

\_isShooted est égale à « false ».

J’affiche la vie pour voir si je touche bien la cible ou non quand je tire.

public void TestShoot()

{

If(???)

\_life -= 1;

else if (\_isShooted)

{

??? = new Vector3(-9, Random.Range(-3.5f, 3.5f), -1);

\_isShooted = ???;

}

print("life = " + \_life);

}

Maintenant que les fonctions principales sont faites. Il va falloir réaliser le « setter » de IsShooted et le « getter/setter » de Speed

Example :

Public bool Exemple

{

Set {

\_exemple = value ;

}

get {

return \_exemple ;

}

}

Une fois cela fait. On va pouvoir terminer de remplir la dernière fonction :

OnMouseDown(), ce trouvant dans le premier cripts.

Vous devrez faire en sorte que IsShooted se trouvant dans le deuxième script passe à true et que la variable \_speed se trouvant aussi dans le deuxième script gagne +1.

private void OnMouseDown()

{

??? = true;

??? += 1;

}

Tu as normalement terminé ce tutoriel, essaie de le modifier si tu le temps :D.

# Bonus exemple de modification :

Mettre une tache de sang quand tu détruits une cible ou un Song lorsque tu tires, mettre des animations, un menu, un bouton pause….