TD2 : Présentation Unity et contrôle basique d'un objet

Kevin Wagrez

September 26, 2013

1 Présentation de l'interface Unity

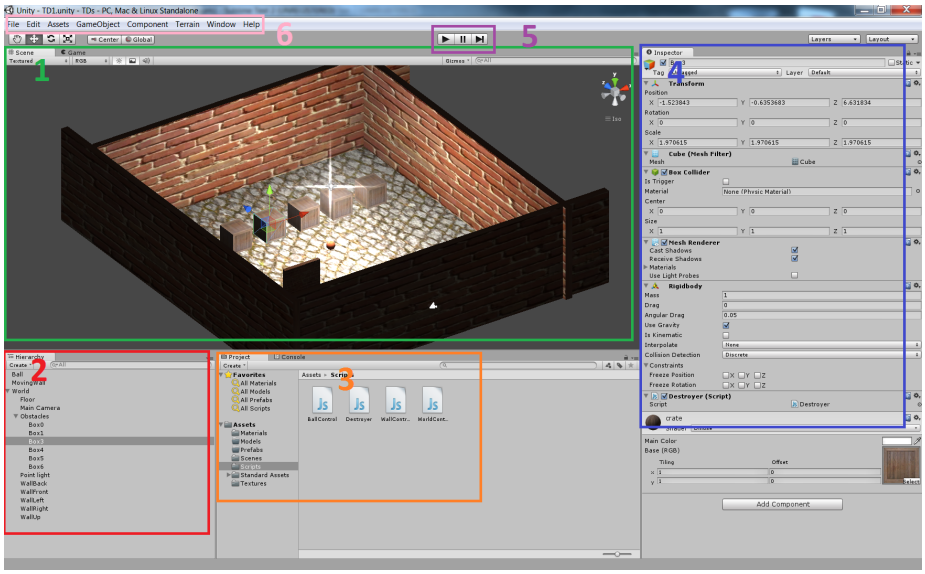
Unity est un moteur d'édition d'applications interactives très usité dans le jeu vidéo ou en réalité virtuelle.

Les éléments importants ici:

- Scene/Game (1): Elle affiche la scène soit d'un point de vue éditeur soit du point de vue de la caméra qui sera utilisée lors du lancement de l'application.
- Hierarchy (2): Elle affiche l'inventaire des éléments de la scène.
- Project (3): Elle affiche l'ensemble des dossiers et ressources du répertoire où se situe le projet.
- Console (3): Elle affiche les possibles erreurs de compilation.
- Inspector (4): Elle affiche les propriétés de l'objet sélectionné, que ce soit un élément de la scène ou un élément externe (texture, son...).
- Contrôle de l'application (5): Permet de lancer, stopper ou arrêter l'application.
- La barre de menu (6): Permet d'accéder à la plupart des options. Deux qui nous intéresserons dans ce TD sont GameObject et Component. Le premier permet la création d'éléments tandis que le second permet d'enrichir un élément existant.

2 Physique d'un objet

Nous allons commencer par créer deux objets simples: un sol et une balle. Allez dans *GameObject* dans le menu, *Create Other* puis *Plane*. Positionnez ce sol de façon à ce qu'il soit bien vu par la caméra. Ensuite, suivez la même démarche



pour créer une Sphere. Positionnez là au dessus du sol.

Pour ajouter de la physique à un objet, il suffit de lui ajouter un composant nommé *rigidbody*. C'est un composant qui permet de gérer les forces appliquées sur l'objet ainsi que la gravité. Sélectionnez la balle et faites soit *Add Component* en bas de *l'Inspector* (4) soit *Component* (6) dans la barre de menu en haut. Faites *Physics* puis *RigidBody*.

Lancez la scène grâce au bouton *Play*. Miracle, la balle tombe ! Histoire de mieux voir cela, créez un *GameObject, Point Light*. Déplacez là où vous voulez. Remarquez que dans l'inspector du Point Light vous pouvez modifier le *range* pour qu'elle éclaire plus.

3 Déplacer un objet

Maintenant nous voudrions pouvoir déplacer notre balle en utilisant le clavier. Pour cela, il est nécessaire de créer un script. Les scripts peuvent être écrit en utilisant les syntaxes Javascript, C# ou Boo (format de script propre à Unity). Créez dans votre projet un dossier *Scripts*. Faites un clic-droit dans le dossier (ou allez dans *Assets*, dans le menu) et faites *Create,Javascript*. Un fichier apparaît dans l'explorateur du Projet. Nommez le *BallControl*, histoire de clairement voir ce qu'il va faire.

Ce script est généré avec deux fonctions prédéfinies : Start() et Update(). *Start()* contient les instructions qui seront exécutées au démarrage de la scène. *Update()* contient les instructions qui seront exécutées en boucle. Nous allons

donc modifier la fonction *Update()* ! Tout d'abord, il faut ajouter le script à un élément de la scène. Faites glisser le script sur l'objet Sphere (que vous aurez renommé *Balle* peut être). Vous remarquez que le script apparaît en tant que Component ! Vous pouvez également ajouter le script en faisant *Add Component puis Scripts* et en choisissant votre script qui sera apparu dans la liste déroulante.

Le script peut avoir accès aux composants de l'objet qui le contient et c'est justement pour cela que nous lions le script de la balle avec l'objet Balle. Définissez dans le fichier une nouvelle fonction *UpdateKeyboardMovements()*.

Dans cette fonction, nous allons récupérer les touches clavier *left, right, up et down*. La classe *Input* donne accès à la plupart des interfaces. Pour chaque touche, nous souhaitons donner un mouvement à la balle. Nous ferons cela en appliquant une force. Ce déplacement aura l'avantage de gérer collisions et gravité.

```
  // Testing the keyboard events for arrows
  // And apply a force to the desired direction on the rigid body
  if(Input.GetKey("right"))
  {
    rigidbody.AddForce(Vector3.right*speed);
  }
  else if(Input.GetKey("left"))
  {
    rigidbody.AddForce(Vector3.left*speed);
  }
  else if(Input.GetKey("up"))
  {
    rigidbody.AddForce(Vector3.forward*speed);
  }
  else if(Input.GetKey("down"))
  {
    rigidbody.AddForce(Vector3.back*speed);
  }
```

../Content/TD1/Scripts/BallControl.js

Spécifiez auparavant une variable publique dans le script nommée *speed*. Elle vous permettra de modifier la puissance de la force appliquée. Une variable publique dans un script apparaît ensuite dans la fenêtre du composant.

```
// This variable allows control of the ball's speed
// It will appear in the component of the GameObject as editable
var speed: double = 5;
```

../Content/TD1/Scripts/BallControl.js

Voilà, sauvegardez et lancez l'application. Le projet compile automatiquement les scripts et vous avertira s'il y a une erreur. Le projet ne peut se lancer

en cas d'errreur, donc vous pouvez être sûr que rien ne vous échappera.

Une autre façon de déplacer un objet est de modifier la position de son composant *transform*. Créez un nouveau *GameObject* (un cube par exemple). Cependant, ne lui ajoutez pas de *rigidbody*. Créez un nouveau script que vous lui attacherez. Nous proposons cette fois comme contrôles les touches classiques *qsdz* pour la translation.

```
// Reference of translation must be specified. It is logical to specify
// the camera as reference !
if(Input.GetKey("d"))
{
  transform.Translate(Vector3.right*speed, Camera.main.transform);
}
else if(Input.GetKey("q"))
{
  transform.Translate(Vector3.left*speed, Camera.main.transform);
}
else if(Input.GetKey("z"))
{
  transform.Translate(Vector3.forward*speed, Camera.main.transform);
}
else if(Input.GetKey("s"))
{
  transform.Translate(Vector3.back*speed, Camera.main.transform);
}
else if(Input.GetKey("a"))
{
  transform.Translate(Vector3.up*speed, Camera.main.transform);
}
else if(Input.GetKey("e"))
{
  transform.Translate(Vector3.down*speed, Camera.main.transform);
}
else if(Input.GetKey("w"))
{
  transform.Rotate(Vector3.down);
}
else if(Input.GetKey("x"))
{
  transform.Rotate(Vector3.up);
}
}
```

../Content/TD1/Scripts/WallControl.js

A nouveau, sauvegardez puis lancez.

Ces deux modes de déplacements sont complètement différents. Le mode physique consiste a appliquer des forces sur les objets pour induire un déplacement graphique. Outre la gestion des déplacements via les forces, le composant Rigid-Body apporte également la gestion des collisions à vos objets. Quand au mode

graphique qui consiste à utiliser le composant Transform, il modifie directement la matrice de pose de vos objets et ne permet pas la gestion des collisions.

Ici, vous remarquerez que l'objet ne gère pas les collisions mais il y a bien collision entre lui et la balle. C'est dû au fait que la mise à jour de la position du composant *transform* ne prend pas en compte les objets déjà présents à l'endroit où il se déplace. La collision est gérée uniquement pour les RigidBody! Cependant la manipulation des objets via le composant Transform offre un meilleur contrôle.

4 Gérer un événement basique

Nous voudrions que notre balle interagisse avec notre cube maintenant. Par exemple, nous pourrions vouloir qu'un contact entre le cube et la balle détruise cette dernière ! Le jeu serait ainsi que la balle fuie devant le cube. Pour cela, nous allons rajouter du contenu au script du cube. Définissez une fonction nommée *OnCollisionEnter*.

```
1  // This function will be called at each collision event.
   // It verifies if the gameobject which collided with the wall is
       named "Ball"
3  // In that case, it removes the ball from the scene
   function OnCollisionEnter(collideEvent : Collision)
5  {
     // Verify the name of the object included in the collision event
7    if(collideEvent.gameObject.name == "Ball")
     {
9      // Remvoes the object fro mthe scene
       Destroy(collideEvent.gameObject);
11   }
   }
```

../Content/TD1/Scripts/WallControl.js

A chaque collision du cube avec un autre objet, cette fonction sera automatiquement appellée. Il n'y a pas besoin de spécifier quoi que ce soit, cela se fait tout seul. Dans cette fonction, on veut accéder à l'objet qui est entré en collision et s'il s'agit de la balle, on le détruit.

Ces fonctions *event* répondent à des messages envoyé par les composants. Vous pouvez vous référez aux références des composants pour connaître la liste des messages et leur origines. (http://docs.unity3d.com/Documentation/ScriptReference/)

5 Gérer son projet

Voilà, nous avons une scène avec des objets physiques, contrôlés et interagissant entre eux. Si vous voulez vous pouvez ajouter de nouveaux objets et/ou enrichir ceux existants avec des textures. Sauvegardez votre scène dans un sous-dossier de votre projet (logiquement nommé *Scenes*). Essayez de garder une grande

rigueur dans l'organisation des ressources, scripts et objets importés dans votre projet car elles seront utilisées pour toutes les scènes de votre projet. Et ce sera un élément important dans l'évaluation du projet à la fin du semestre !

Si vous souhaitez ajouter des textures, vous pouvez directement les copier coller dans les dossiers, ou les faire glisser sur la fenêtre Unity. Pour appliquer une texture à un objet, sélectionnez la texture dans la fenêtre projet et faites la glisser sur l'objet. Une material est automatiquement créée avec votre texture en canal *Diffus* et appliquée à l'objet. Unity est magique, n'est-ce pas ?

6 Exercice

Pour voir si vous avez bien compris, complétez la scène en ajoutant plusieurs boîtes (ou autres objets) qui se détruisent lorsqu'ils sont touché par la balle ! Le but du jeu sera cette fois de détruire toutes les boîtes en évitant le cube !

7 Ressources externes

Si vous voulez vous former en avance à Unity (dans l'ordre décroissant au niveau pédagogique) :

- (FR) `http://www.djor-gri.com/blog/?p=652`
- (EN) `www.unity3dstudent.com/2010/10/beginner-b23-particle-systems/`
- (FR) `http://www.unity3d-france.com/unity/`
- (FR) `http://www.youtube.com/watch?v=GTxtJ7f4Tg0`

Pour les non-développeurs qui aimeraient un peu s'initier à la syntaxe *Javascript*:

- `http://fr.openclassrooms.com/informatique/cours/dynamisez-vos-sites-web-avec-javascrip`

Enfin, si vous cherchez des modèles 3D sur l'internet, quelques liens (nous vous rappellons qu'Unity peut presque tout importer comme format d'objet 3D):

- `http://archive3d.net/`.
- `http://animium.com`.
- `http://www.123dapp.com/Search/bug`.