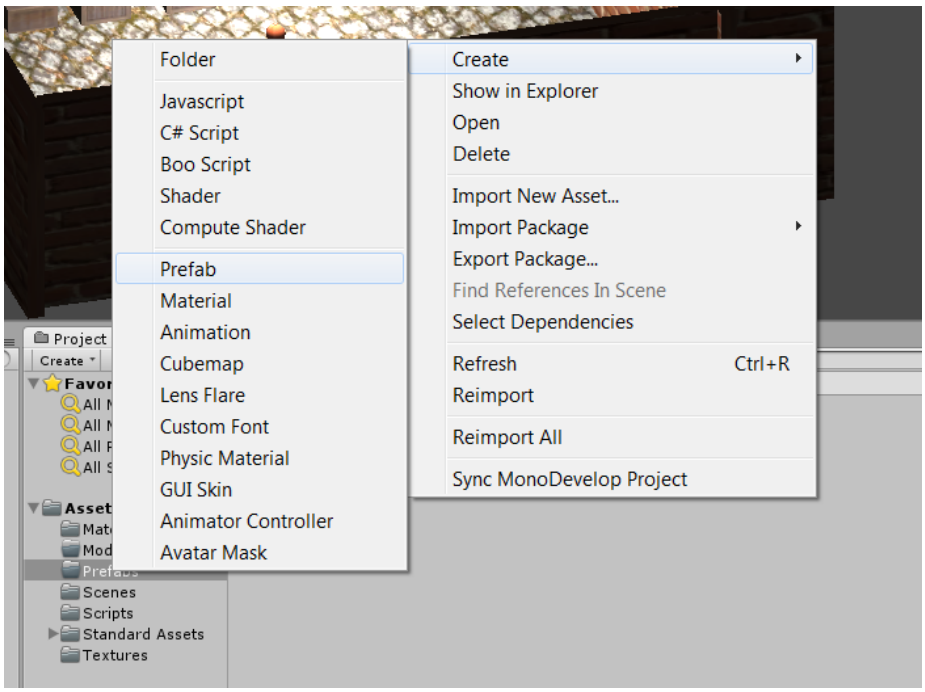
TD2 : Animation, son et particules

October 4, 2013

1 Prefabs

Durant le TD1 vous avez eu l'occasion de créer un certain nombres d'éléments pouvant se déplacer et interagir entre eux. Si vous souhaitez conserver votre travail sur ces éléments, Unity propose d'utiliser des *Prefabs*. Il s'agit seulement de structures dans lesquelles vous pouvez mémoriser un ou plusieurs objets, avec leurs configurations et leurs scripts. Pour en créer un, faites un clic droit dans votre fenêtre *Projet* ou *Assets puis Create et enfin Prefab*. Un objet grisé apparaît.



Si vous voulez enregistrer des objets dans ce prefab, faites les glisser de votre fenêtre *Hierarchy/Projet*. L'icône va alors changer de forme. Vous pour-

rez dorénavant réutiliser ce prefab dans n'importe quelle scène de votre projet. L'avantage est que si vous modifiez le *Prefab*, cette modification s'appliquera à toutes les instances dans toutes les scènes du projet !

Créer un prefab à partir de la balle sur laquelle vous aviez ajouté des scripts lors du TD précédent.

2 Instantiation d'un objet

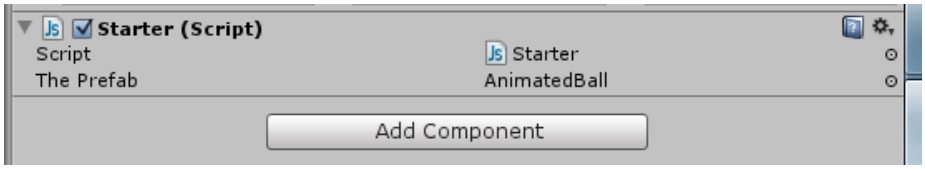
Sauvegardez votre ancienne scène et créez en une nouvelle. Créez un objet vide en faisant *GameObject puis Create Empty*. Nommez cet objet *StartPoint* et mettez le au milieu de la scène, en hauteur. Ce sera notre point de départ de la balle du joueur ! Vous pouvez utiliser les champs de saisie dans l'inspector pour positionner l'objet.

Créez et ajoutez à cet objet un nouveau script Javascript. Nous allons spécifier une variable publique qui contiendra la référence vers l'objet à créer. Nous allons ensutie utiliser la commande *Instantiate* qui prend en paramètres la référence du *GameObject* à créer ainsi que ses positions et orientations.

```
1 // Public variable that will hold the reference of the object to
      create
  var thePrefab : GameObject;
3
  function Start () {
5   // Create in the scene a new instance of the object
    var instance : GameObject = Instantiate(thePrefab, transform.
      position, transform.rotation);
7 }
```

../Content/TD2/Scripts/Starter.js

Faites glisser votre *Prefab* vers la variable publique référencé dans le composant dans la fenêtre *inspector* de notre objet de départ. Puis lancez l'application. Voilà, simple non ?



3 Le son du rebond

Notre balle rebondit dans le silence le plus total. Cherchez sur internet un son de balle (ou n'importe quel autre son), court si possible et de format soit MP3

soit WAV (Unity peut en gérer d'autres éventuellement). Notre objectif est de lier ce son à la balle et de l'activer à chaque collision !

Créez un composant *Audio source* pour votre balle. Modifiez le script de contrôle de la balle en ajoutant la fonction qui récupère l'événement de collision (TD1). Ajoutez également une variable publique qui contiendra l'élément sonore que l'on souhaite jouer. Souvenez-vous, il faut modifier le *Prefab* et non une instance dans la scène car sinon vos modifications ne s'appliqueront qu'à cette instance.

```
1  // Contains the audio clip to play
   public var soundEffect: AudioClip;
3
   // Starts audio clip at collision !
5  // It needs a component audio source i nthe GameObject of course
   function OnCollisionEnter(collideEvent:Collision)
7  {
     audio.PlayOneShot(soundEffect);
9    // Or, an alternative
     // AudioSource.PlayClipAtPoint(myClip, transform.position);
11 }
```

../Content/TD2/Scripts/BallControl.js

Lancez l'application, et voilà le travail. Il existe une autre approche qui n'utilise pas de source audio rattachée à l'objet. Pour cela, au lieu d'utiliser le composant et sa fonction *PlayOneShot*, il faut utiliser la fonction *PlayClipAtPoint* de la classe *AudioSource* (et non l'instance) avec en deuxième argument la position du composant *transform* de l'objet. Cela créera temporairement une source audio dans la sècne qui jouera le clip avant de disparaître.

4 Liens physiques

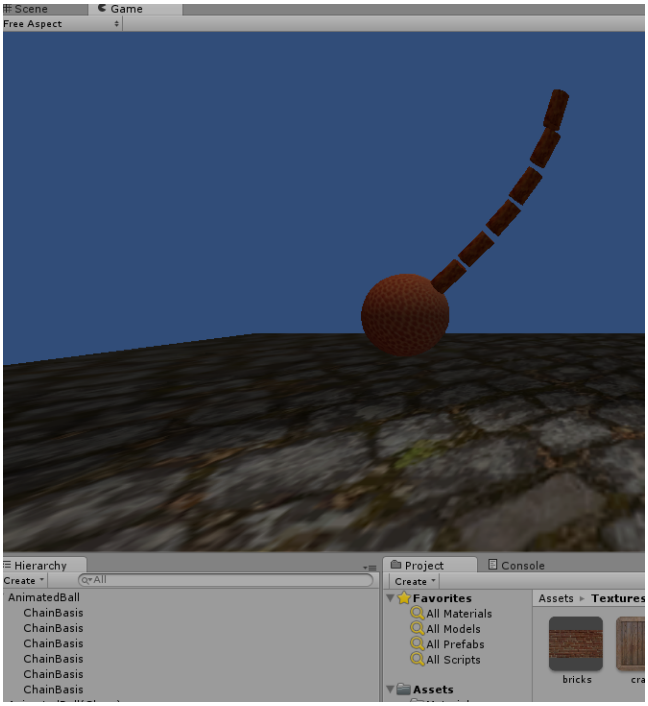
Notre balle est un peu trop libre, il serait peut être utile de la tenir en laisse. Pour cela, nous avons besoin de créer des *joints* (je sais à quoi vous pensez). Rajoutez une instance de votre balle dans la scène, ca sera plus simple pour cela. Créez ensuite un petit cylindre que vous positionnerez de façon à ce qu'il soit légéremment enfoncé dans le ballon. Ajoutez un *rigibody* à ce cylindre.

Maintenant ajoutez un composant *physics puis fixed joint* à la balle. Ce composant demande à être connecté à un autre rigidbody. Faites glisser le cylindre sur la variable pour connecter balle et cylindre. Voilà, les deux objets sont liés et le cylindre influence la physique de la balle. Mais l'objectif était une chaîne !

Créez un second cylindre (vous pouvez dupliquer avec Ctrl+D le premier) et créez/remplacez un composant *Hinge joints*ur le premier cylindre. Faites glisser le second cylindre sur la variable pour le relier. Répétez l'opération plusieurs

fois pour avoir une chaine conséquente.

Les *fixed joint* créent des liaisons fortes entre les objets (de la même manière que la hiérarchie). Les *hinge joint* permettent de créer des liaisons pivot (un degré de liberté).



Le fait de laisser le dernier cylindre non rattaché à un corps fait que celui-ci est attaché au monde. C'est à dire que votre balle va pendouiller, suspendue par ce dernier lien avec le monde.

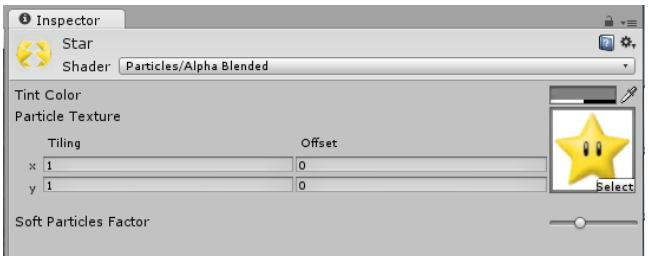
5 Détruire avec feux d'artifices

Notre balle ainsi chaînée est une formidable arme de destruction. Cependant, c'est assez peu spectaculaire. Il faut ajouter quelques effets pour en mettre plein la vue. Nous allons donc utiliser des systèmes de particules !

Ajouter dans la scène un objet qui pourra être détruit par la balle attachée à la chaîne. Mettez la balle et la chaîne de façon à ce que la gravité la force à tomber sur l'objet et le détruire. Vérifiez que la boule détruit bien l'objet. Maintenant nous allons nous atteller à créer l'effet d'explosion.

Tout d'abord recherchez sur Google une texture (petite) d'une étoile ou d'une petite étincelle. Un exemple ici est l'étoile de Mario. Notez qu'il est important que l'image ait un canal Alpha. Le canal alpha détermine la transparence des pixel de l'image. Les format GIF et PNG supportent ce canal (et non pas les JPG). Si ce n'est pas le cas, utilisez Gimp/Photoshop pour la modifier.

Nous cherchons à créer le matériau qui sera utilisé par la génération de particules. Chaque particule utilisera cette texture. Créez un dossier matériaux, si ce n'est déjà fait, et créez un nouveau matériau. Faites glisser votre texture vers la petite case prévue à cet effet. Modifiez le shader en sélectionnant le menu déroulant et utilisez *Particles/Alpha blended*.

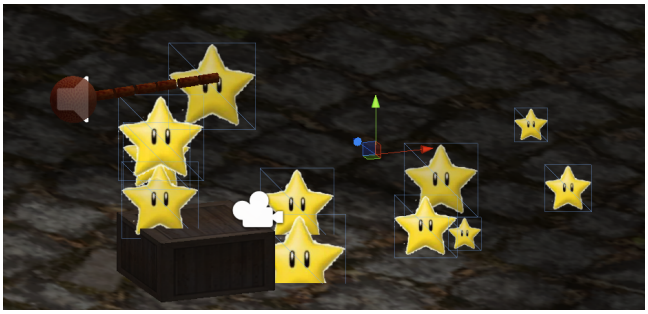


Maintenant, reste à créer l'objet générant ces particules. Il y a deux façons de faire, l'une étant basée sur une version récente d'Unity. Pour être sur que tout le monde puisse faire ce TD, nous allons décrire la méthode qui marchait même dans les versions antérieures. Créez un objet vide dans la scène et rajoutez lui 3 composants :

- Ellipsoid Particle Emitter
- Particle Animator
- Particle Renderer

Ils peuvent être trouvés dans composants puis *Effects*. Dans le *Particle Renderer*, vous verrez une variable dans laquelle placer le matériau que vous avez fabriqué. Vous remarquerez que durant l'édition de l'objet, les particules sont générées en boucle pour faire office de preview. Trifouillez les paramètres pour comprendre ce à quoi chacun correspond. Cependant, deux sont importants dans notre contexte:

- Cochez la case *One Shot* dans l'emitter
- Cochez la case *Autodestruct* dans l'animator

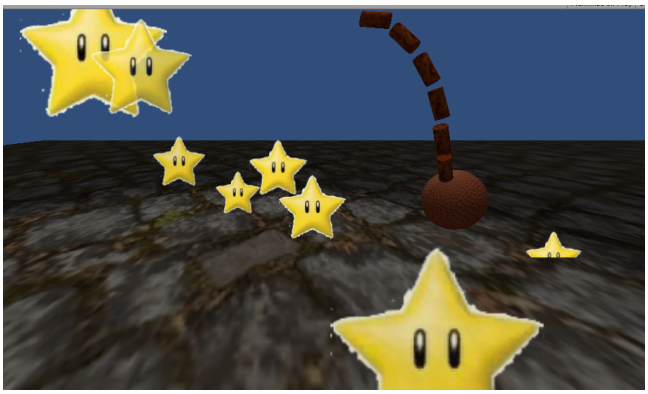


Ces deux options permettent à l'objet générateur de particules de lancer une seule fois ses particules avant de s'auto-détruire. C'est parfait pour notre cas. Cependant, il reste à pouvoir l'utiliser au moment opportun. Faites en un prefab et supprimez le de votre scène. Modifiez le script de destruction des objets pour y rajouter une variable publique prenant notre prefab de générateur de particules en entrée. Et lors de la destruction, vous n'avez plus qu'à instantier l'objet de la même façon que dans la partie 2.

```
1  public var specialEffect : GameObject;
2
3  function Start () {
4
5  }
6
7  function Update () {
8
9  }
10
11 function OnCollisionEnter(collideEvent : Collision)
12 {
13   if(collideEvent.gameObject.name == "Ball" )
14   {
15     // We destroy the object, but with class !
16     Instantiate(specialEffect, transform.position, transform.
       rotation);
17     Destroy(gameObject);
18   }
19 }
```

../Content/TD2/Scripts/Destroyer.js

Et voilà.



6 Exercice

Essayez d'utiliser votre balle à chaîne (ou Morgenstern/Fléau) pour détruire des objets rebondissants dans un espace clos. Rajoutez des bruits d'explosions, des particules et la destruction de ces objets lors d'un contact avec le fléau !