Unity - TP3

Les pieces d'or

Ouvrez votre scène Terrain. L'objectif dans ce TP sera de collecter des objets, d'utiliser des sons, d'augmenter la difficulté du deuxième niveau et de réaliser l'interface utilisateur. Les objets à collecter seront des pièces d'or, mais vous pouvez utiliser un autre objet de votre choix.

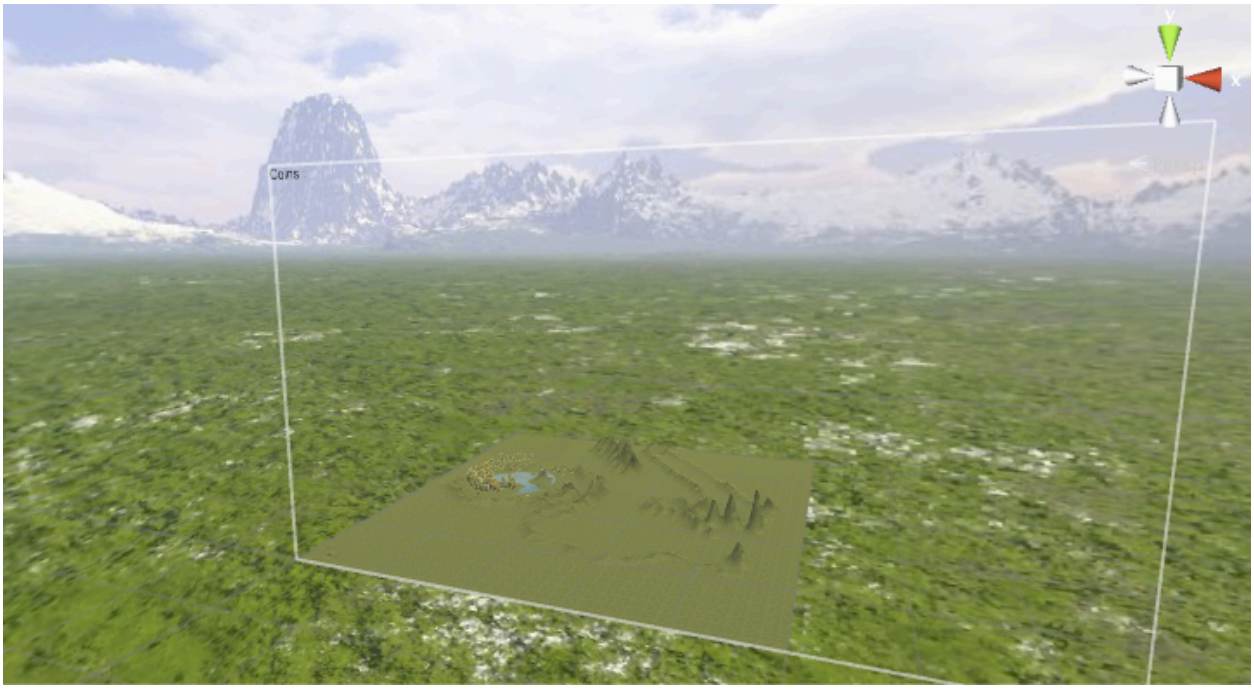
Pour la géométrie des pièces, créez un cylindre et modifiez-le pour qu'il ait la forme d'une pièce. Puis, collez la texture de votre préférence sur le cylindre (quelques textures sont disponibles dans les ressources du TP). Créez un Prefab « Coin » et glissez le cylindre vers lui. Ajoutez à votre projet le fichier coinSound.wav. En gardant le prefab Coin sélectionné, créez ensuite un composant ***Component > Audio > Audio Source***. Glissez le fichier coinSound.wav vers le champ AudioClip. Enlevez l'option Play on Awake.

Maintenant, dans le Collider de l'objet Coin, cliquez sur la case isTrigger pour l'activer. Puis, créez un nouveau script CoinBehaviour. Il nous permettra de « désactiver » la pièce une fois qu'elle aura été trouvée :

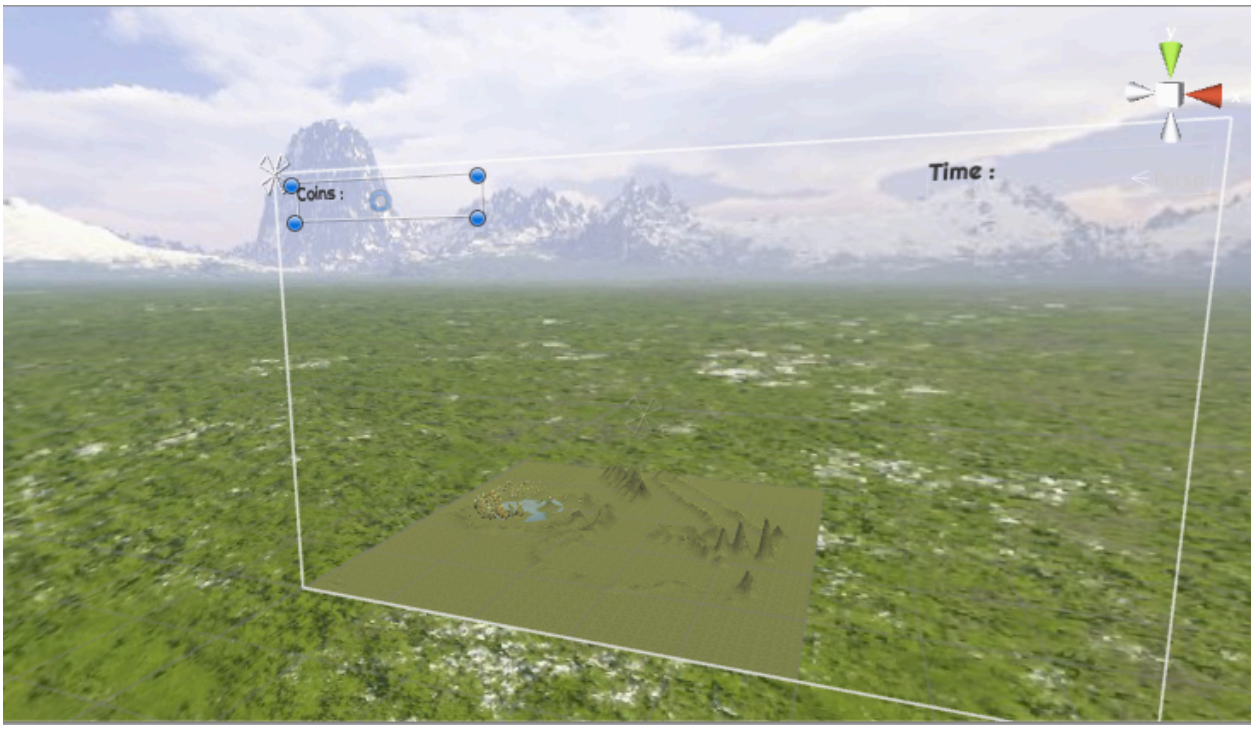
```
var worldObject : GameObject;
var aud: AudioSource;

function Start() {
   worldObject = GameObject.Find("World");
   aud = gameObject.GetComponent.<AudioSource>();
}

function OnTriggerEnter( other : Collider ) {
     worldObject.SendMessage("AddCoin");

     GetComponent.<Renderer>().enabled = false;
     GetComponent.<Collider>().enabled = false;
     if ( aud ) {
        aud.Play();
     }
}
```

Liez votre script à l'objet Coin. Ajoutez plusieurs pièces à votre scène. Nous devons maintenant compter les pièces et afficher cette information sur l'écran.

Pour créer l'interface, il est nécessaire d'ajouter un canvas à la scène. Il sera créé par défaut lorsque vous créez un premier élément d'UI (User Interface). Créez le canvas, il apparaitra comme un long rectangle à coté de votre scène. Cette configuration par défaut est donnée par la valeur **Screen Space Overlay** du **Render mode** :



Créez ensuite deux éléments UI Text. Ils se placent sur le canvas. Vous pouvez les déplacer (en cliquant-glissant sur leur milieu) ou changer leurs dimensions grâce aux poignées :



Nous allons créer maintenant l'objet « World ». Créez un nouveau GameObject vide et appelez-le World. Puis, créez un script, world.js avec le code suivant :

```
var coinsText : UI.Text;
var coins : int = 0;
var canvasObj : GameObject;
var child : Transform;

function Start () {
    canvasObj = GameObject.Find("Canvas");
    child =  canvasObj.transform.Find("CoinsText");
    coinsText = child.GetComponent(UI.Text);

}

function AddCoin () {
   coins++;
   coinsText.text = "COINS: " + coins;
}
```

Liez le script à l'objet World. Testez votre jeu.

TIMER

Nous allons ajouter maintenant le timer à notre jeu pour le rendre plus intéressant ! Créez un nouveau script Countdown.js associé à l'objet World :

```
public var allowedTime : int = 90;
var timerText : UI.Text;
var currentTime = allowedTime;
var canvasObj : GameObject;
var child : Transform;


function Start () {
    canvasObj = GameObject.Find("Canvas");
    child =  canvasObj.transform.Find("TimerText");
    timerText = child.GetComponent(UI.Text);
    TimerTick();
}

function TimerTick() {
    while(currentTime > 0) {
       //attendre 1 seconde
        yield WaitForSeconds(1);

        currentTime--;
        timerText.text = "Time :" + currentTime.ToString();
    }

    // game over and restart
    Application.LoadLevel("Terrain");
}
```

Comme pour le script précédent, nous récupérons le label grâce au canvas. Il est également possible de mesurer le temps utilisé au lieu de faire un compte à rebours. La fonction TimerTick contient une boucle qui s'arrête lorsque le compteur arrive à 0. La fonction

WaitForSeconds(…) arrête l'exécution pendant une seconde et puis, réduit le temps disponible et met à jour le texte à afficher.

DE LA CONTINUITE DU TEMPS

Vous vous êtes surement demandé comment faire pour que le Timer que nous avons implémenté fonctionne sur tout le jeu et non seulement pour un niveau. La réponse : les variables statiques !! (bon, c'est vrai, je ne vous apprends rien ;) pas besoin de deux points d'exclamation !!!)

Sur la scène terrain, créez un script définissant une classe avec des variables statiques :

```
public static class GameVariables {
	public var allowedTime: int = 90;
	public var currentTime = GameVariables.allowedTime;
	public var coins : int = 0;
}
```

Modifiez maintenant les scripts Countdown.js et World.js pour utiliser les variables statiques : GameVariables.coins, GameVariables.currentTime et GameVariables.allowedTime.

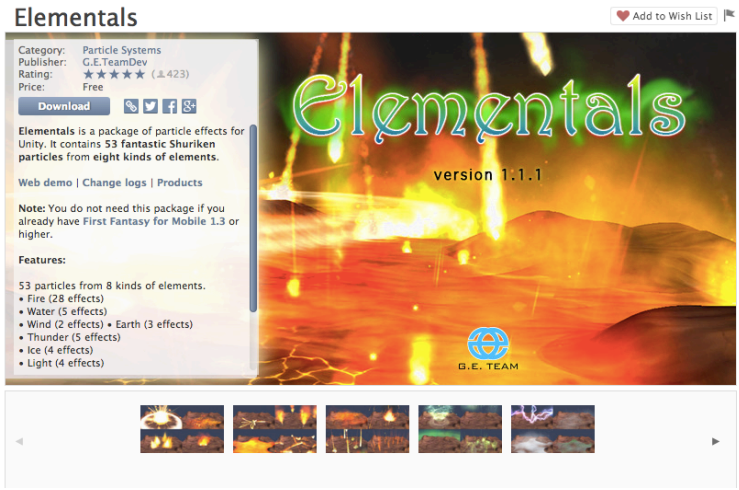
Maintenant :

- affichez un message différent lorsque le joueur a trouvé toutes les pièces.
- ne permettez pas au joueur de changer de niveau que lorsqu'il aura trouvé un nombre prédéfini de pièces
- créez des pièces différentes qui donnent plus de points ou qui en enlèvent !

Sauvegardez votre scène et ouvrez la scène BoxBoy. Créez les éléments nécessaires à la mise à jour et à l'affichage du Timer.

LES SYSTEMES DE PARTICULES

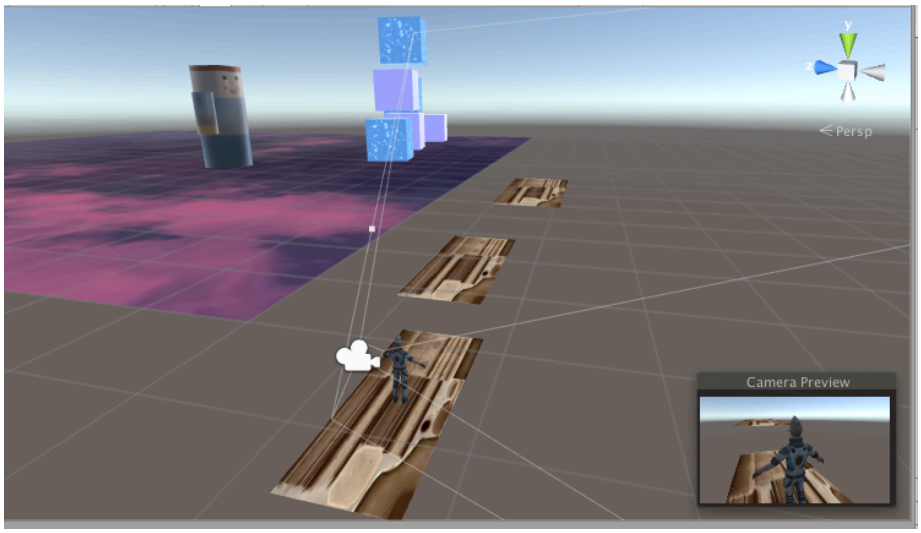
Les systèmes de particules permettent de créer des effets d'eau, vent, nuages, etc. Certains packages vous offrent des effets prédéfinis plutôt spectaculaires ! Dans la capture d'écran de la page précédente, un système de particules a été ajouté à la fontaine pour l'effet « magique ». J'ai utilisé l'effet Cyclone du package Elementals, disponible dans l'asset store. (Attention : le chat n'est pas un système de particules ☺)



Ouvrez votre scène BoxBoy. Ajoutez au prefab de la lampe un système de particules simple ***Component > Effects > ParticleSystem***. (http://docs.unity3d.com/Manual/class-ParticleSystem.html). Décochez l'option Looping pour que les particules s'envolent à tout jamais. Bien évidemment vous pouvez choisir un autre effet.

LES PLATEFORMES

Ouvrez votre scène BoxBoy. Nous allons donner du fil à retordre à notre personnage principal. Créez plusieurs plans séparés et modifiez leurs positions de manière à créer un escalier. Le personnage doit maintenant sauter pour arriver jusqu'à la plateforme où se trouve le boxboy magicien.



Si le personnage rate une marche, il tombe à l'infini. Pour pouvoir continuer le jeu en relançant le niveau au lieu de le relancer, nous allons créer un plan parallèle à la plateforme et des dimensions suffisantes pour contenir toutes les marches. Créez un objet GameObject vide en utilisant ***GameObject > Create Empty***. Ajoutez à cet objet un BoxCollider (***Add Component > Physics > Box Collider***), et activez l'option isTrigger. En associant au plan le script suivant, à chaque fois que le personnage tombera, il sera renvoyé au début du niveau :

```
function OnTriggerEnter( other : Collider ) {
  Application.LoadLevel("BoxBoy");
}
```

Il est temps maintenant de m'envoyer votre projet ☺ Envoyez le lien de téléchargement à nancy.rodriguez@lirmm.fr.

BON WEEK-END ET BONNES VACANCES !