TP2 - Unity

Pour ce TP, vous devez déposer un compte rendu avec de captures d'écrans et vos personnalisations dans Moodle et fournir un lien avec le code de votre projet.

Ouvrez votre projet et puis le prefab des CatBots. Ajoutez un composant BoxCollider et cochez la case isTrigger pour pouvoir détecter les collisions. Nous allons donner à Ethan le pouvoir de détruire un CatBot en l'approchant. Pour cela, créez un nouveau script à ajouter à votre prefab :

```
using UnityEngine;

public class CatBehaviour : MonoBehaviour
{

    void Start() {
    }

    void Update() {

    }

    void OnTriggerEnter(Collider other) {   // OnCollisionEnter
        if (other.tag == "Player") {
            Destroy(gameObject);
        }
    }
}
```

Pour donner un retour à l'utilisateur, vous allez ajouter un bruitage. Ajoutez un composant ***Component > Audio > Audio Source*** à votre GameObject World. Glissez ensuite le bruit choisi vers le champs AudioClip de l'objet. Enlevez l'option Play on Awake pour empêcher le son de se déclencher automatiquement.

Modifiez votre script CatBehaviour pour récupérer le composant AudioSource créé :

```
        collisionSound = GameObject.Find("World").GetComponent<AudioSource>() ;
```

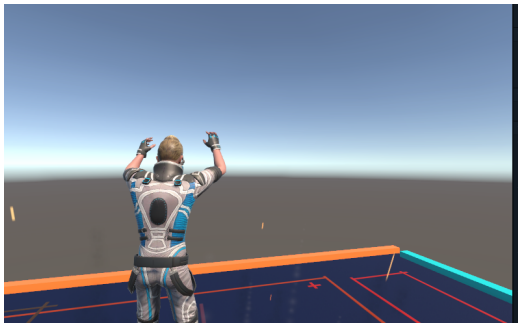
Lors de la collision, activez le son avant de détruire l'objet :

```
        if (collisionSound) {
                collisionSound.Play();
        }
```

Pour avoir un retour plus spectaculaire, nous allons utiliser un système de particules. Créez une variable GameObject fx en tant que variable « exposée » et puis créez une instance au moment de la détection de la collision :

```
            Instantiate(fx, transform.position, Quaternion.identity);
```

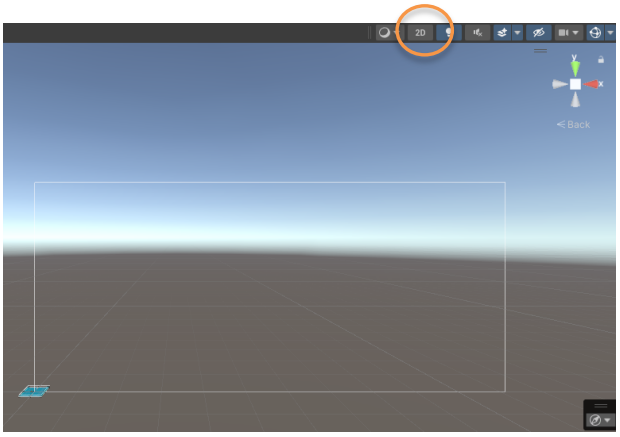
Pour la variable fx, vous pouvez utiliser un des effets disponibles dans **Standard Assets > Particle Systems > Prefabs**. J'ai utilisé l'effet Explosion.



L'INTERFACE

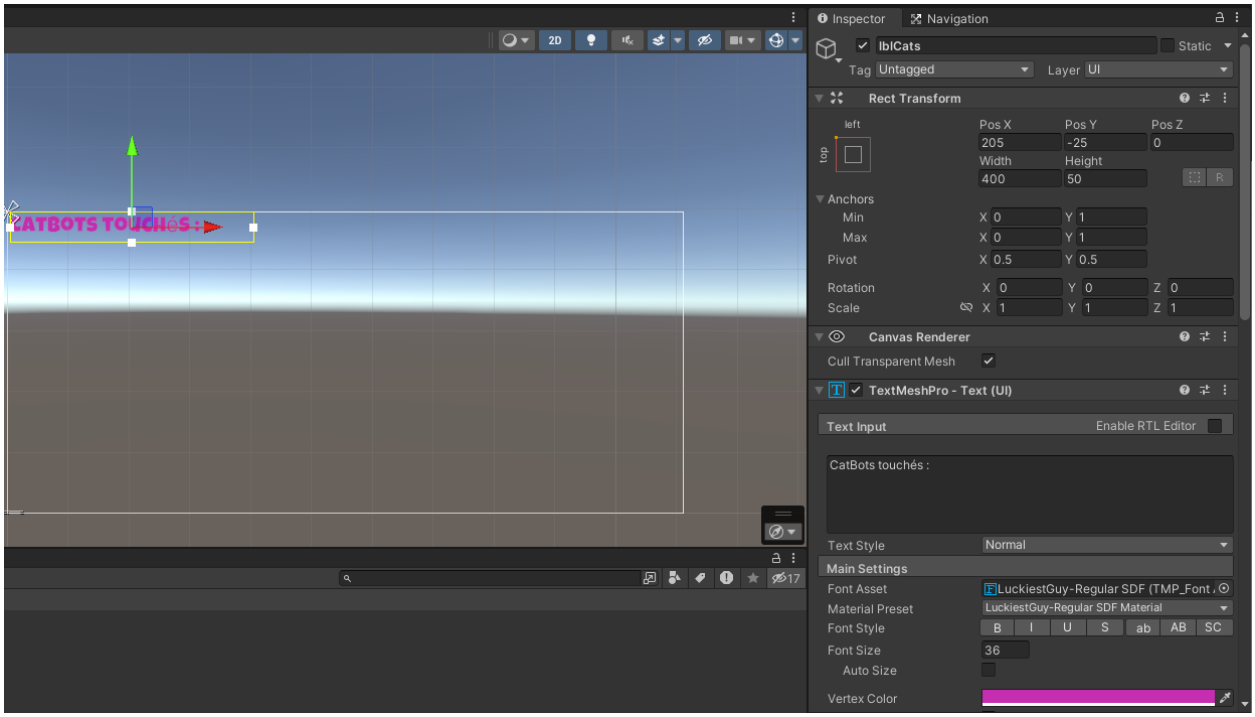
Maintenant nous allons compter les CatBots touchés par le joueur et afficher cette information sur l'écran. Pour créer l'interface, il est nécessaire d'ajouter un canvas à la scène : menu ***GameObject > UI***. Le canvas apparaitra

comme un très long rectangle à coté de votre scène. Pour travailler plus facilement avec les éléments de l'interface, utilisez le bouton **2D**.



Créez un objet Texte ***GameObject > UI > Text MeshPro*** et placez-le en haut à gauche de votre canvas (des éléments TMP Essentials seront à importer). Nommez-le lblCats.

Modifiez le **Font asset** en choisissant une police et modifiez la taille si nécessaire. Pour utiliser une police de votre choix, allez dans le menu ***Window > TextMeshPro > FontAssetCreator***. Glissez le font choisi et cliquez sur GenerateFontAtlas. Enregistrez le FontAsset et glissez-le dans le FontAsset du TextMeshPro.



Créez un script à ajouter à votre objet World avec le code ci-dessous, permettant de tenir le compte de CatBots détruits et de modifier le texte à afficher :

```
using TMPro;
using UnityEngine;

public class UIBehaviour : MonoBehaviour {
    TMP_Text headText;
    int nbCats = 0;

    void Start()  {
        headText = GameObject.Find("lblCats").GetComponent<TMPro.TMP_Text>();
    }
```

```
    void Update()  {
    }

    public void AddHit() {
        nbHeads++;
        headText.text = "BobHeads: " + nbHeads;
    }
}
```

Modifiez votre script CatBehaviour avec les lignes suivantes :

```
public GameObject worldObject;

Au début de la fonction Start :
        worldObject = GameObject.Find("World");

Dans la fonction OnTriggerEnter :
        worldObject.SendMessage("AddHit");
```

Ces lignes font le lien entre la détection des collisions et l'affichage dans l'UI.

TIMER

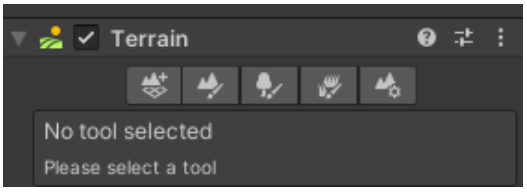
Vous allez ajouter le temps à votre jeu pour le rendre plus intéressant ! Créez un nouveau ***GameObject > UI > Text*** lblTime et placez-la à droite du canvas. Modifiez le script UIBehaviour ou créez un nouveau script pour réaliser le compte à rebours, et l'afficher dans l'interface. Voici une fonction utile :

```
    IEnumerator TimerTick() {
        while (currentTime > 0)
        {
        // attendre une seconde
        yield return new WaitForSeconds(1);
        currentTime--;
        timerText.text = "Time :" + currentTime.ToString();
        }
        // game over
        SceneManager.LoadScene("sceneCatBots");   // le nom de votre scene
    }
```

à appeler dans la fonction Start(), de la manière suivante : StartCoroutine(TimerTick()). Dans cette fonction, lorsque le temps est fini, la scène est rechargée, grâce à la dernière ligne.

CREATION D'UN TERRAIN

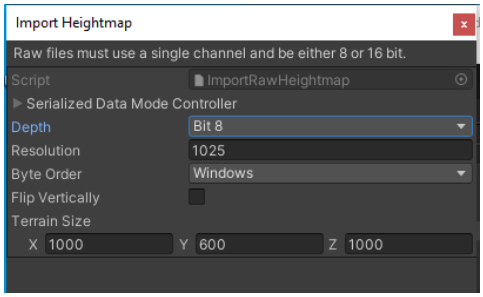
Intégrez à votre projet le package **Environment**, disponible dans le dossier du TP. Créez une nouvelle scène dans votre projet. Dans cette scène, créez un terrain grâce au menu ***Game Object > 3D Object >Terrain***. Le terrain est, par défaut, un maillage plat. Dans l'Inspector on trouvera les outils permettant de modifier le maillage :

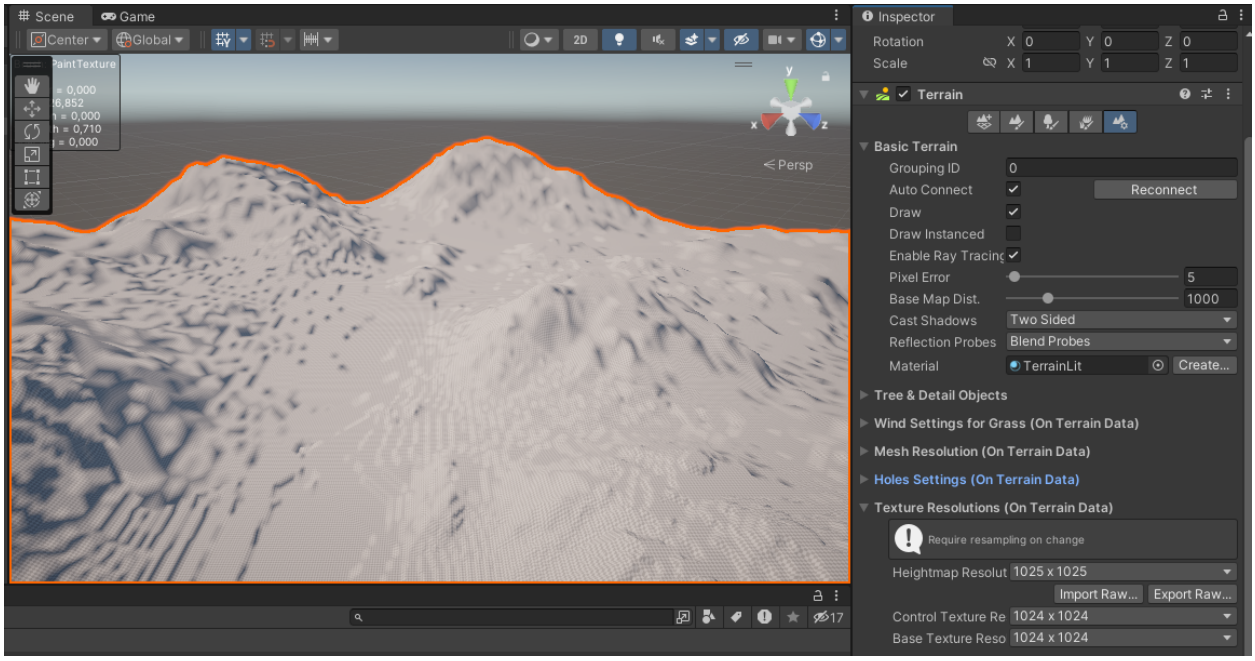


De gauche à droite :

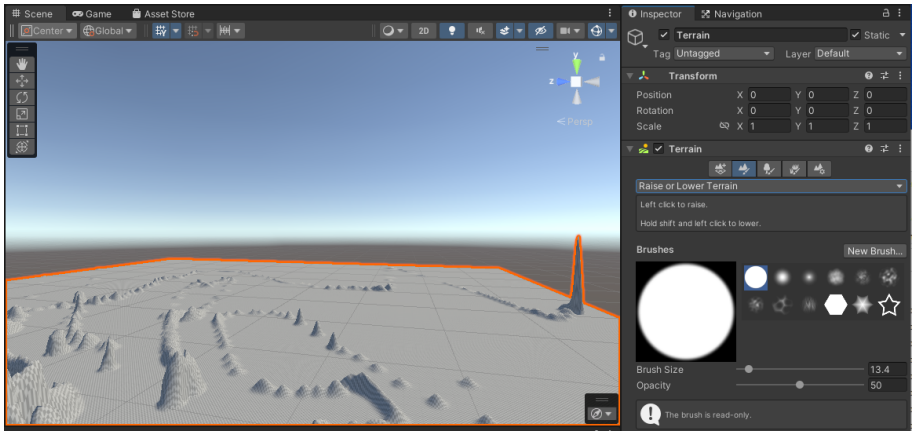
- Créer des terrains adjacents
- Sculpter et peindre le terrain
- Ajouter des arbres
- Ajouter des détails comme l'herbe, fleurs et roches
- Modifier la configuration du terrain

Vous pouvez créer votre terrain entièrement en « sculptant » grâce à l'outil 2, mais il est également possible de créer un terrain à partir d'un heightmap, une image en niveaux de gris où chaque pixel définit une hauteur, le noir étant la hauteur minimale et le blanc la hauteur maximale. Pour utiliser un map, importer le fichier .raw ou .data dans votre projet. Puis, cliquez dans le dernier bouton de la barre d'outils Terrain et ensuite dans le bouton **Import Raw** de la rubrique **TextureResolution**. La fenêtre d'import vous permet de choisir votre fichier et de rentrer les différentes informations :



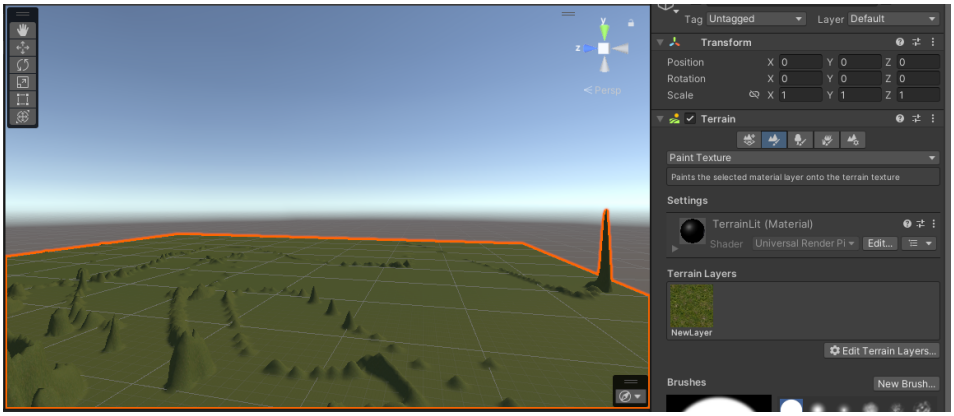


En partant d'une surface importée ou du terrain plat, le deuxième bouton vous donne donc accès aux outils de modification. Utilisez les différents modes come **Raise or Lower Terrain**, **Set Height** ou **Sculpt** pour sculpter votre terrain, en changeant les brosses, et en cliquant ou glissant sur une zone pour la modifier.

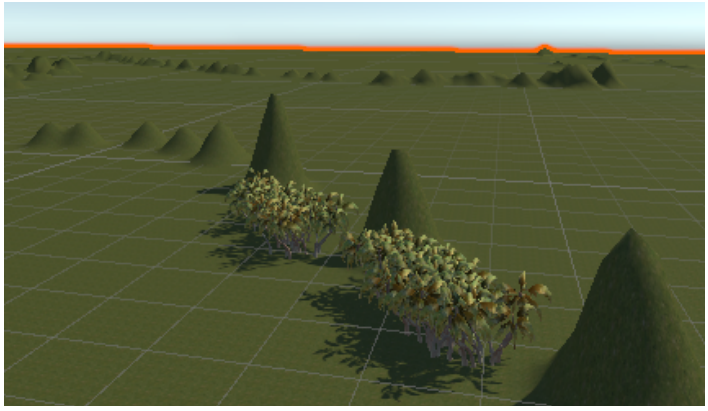


Appliquez ensuite une texture au terrain : Changez le mode du bouton à **Paint Texture** et puis ajoutez des **Terrains Layers : Edit Terrain Layers** puis **Create Layer**. Le premier **TerrainLayer** remplit le terrain avec

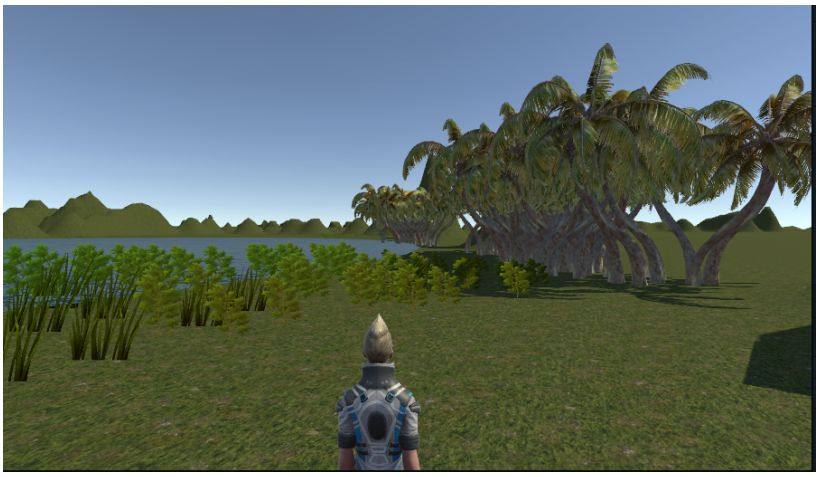
la texture choisie (j'ai utilisé « GrassHillAlbedo »). Les calques suivants permettent d'ajouter des détails mais l'utilisation de plusieurs calques peut nuire à la performance.



Pour ajouter des arbres, cliquez sur le troisième bouton **Paint Trees.** Ajoutez un prefab Arbre en cliquant sur le bouton **Edit Trees** et sélectionnez **Add Tree**. Puis, sélectionnez dans **Tree Prefab** un asset disponible (PalmTree dans l'exemple). Pour ajouter les colliders, activez l'option enable Tree Colliders dans l'Inspector.



Pour ajouter une zone d'eau, ajouter à votre scène un des prefabs disponibles dans le package **Standard Assets > Environment > Water (Basic)** : WaterBasicDaytime and WaterBasicNightime. L'eau est une « texture animée », elle ne déborde pas, vous ne pourrez pas créer des cascades par exemple. Vous pouvez améliorer l'effet en créant une bordure avec de l'herbe. Pour cela, cliquez sur le bouton **Paint Details** et puis sur **Edit Details > Add Grass Texture**. Modifiez la rubrique **Detail Texture**, la texture choisie représentera l'herbe.



Pour créer l'effet vent, il faut ajouter un composant spécifique : ***Component > Miscellaneous > Wind Zone***. L'Inspector vous permet de configurer le comportement du vent. Le mode Directional applique le vent sur tout le terrain. Dans le mode Spherical, le vent souffle vers l'extérieur d'une sphère défini par la propriété Radius. Le vent souffle par pulsations (pulses) pour créer un effet plus naturel. Attention : cette fonctionnalité est consommatrice de ressources, vous pouvez la désactiver en mode test.

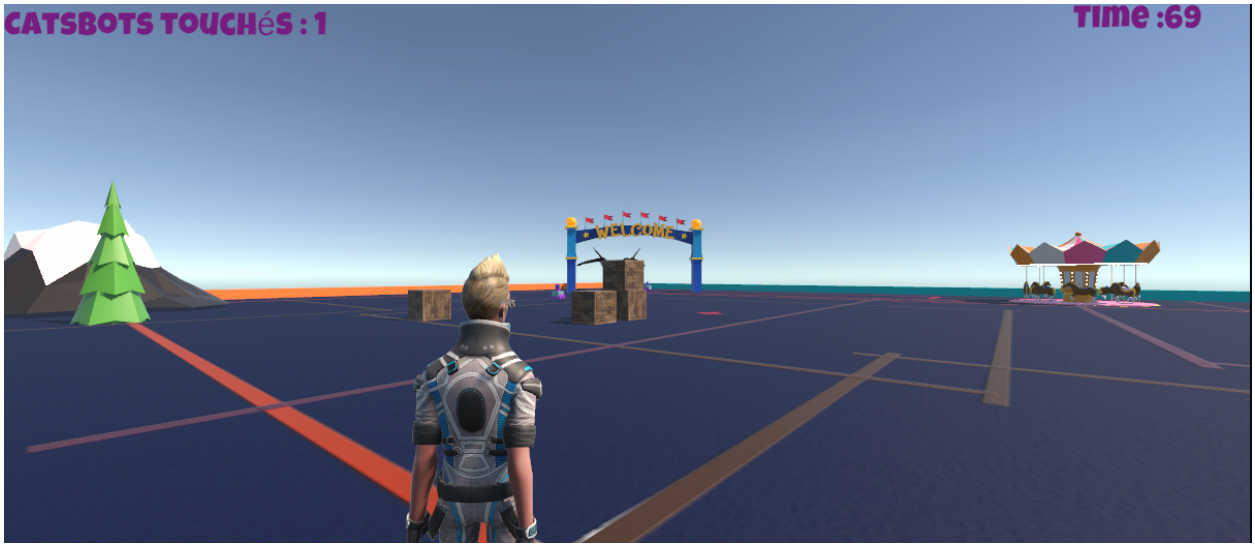
Pour finir l'apparence de la scène, ajoutez une skybox et le TPC.

DE LA CONTINUITE DU TEMPS

Vous vous demandez surement comment faire pour que le Timer que nous avons implémenté dans la première scène fonctionne sur tout le jeu et non seulement pour un niveau. La réponse : les variables statiques[1]. Créez un script définissant une classe avec des variables statiques :

```
public static class GameVariables {
        public static int allowedTime = 90;
        public static int currentTime = GameVariables.allowedTime;
}
```

En utilisant ces variables, créez maintenant les éléments nécessaires à la mise à jour et à l'affichage du timer dans la scène Terrain. Puis, dans la scène CatBots, modifiez le script contenant votre compte à rebours pour utiliser les variables statiques GameVariables.currentTime et GameVariables.allowedTime. J'ai ajouté à cette scène quelques objets du package County Fair[2] du dossier du TP.



CHALLENGE FINAL

Créez le lien entre les deux scènes du projet. Par exemple, créez un script qui charge la scène Terrain lorsque le TPC s'approche du dragon une fois qu'un certain nombre ou tous les CatBots sont détruits. Imaginez ensuite une « quête » pour la seconde scène.

Voilà ! Déposez votre compte rendu de TP et le lien pour le téléchargement du code dans l'espace Moodle.

[1] *Vous avez peut-être songé au Singleton…une recherche rapide vous montrera que le débat est ouvert !*

[2] *https://www.vrcreators.io/the-confident-vr-dev-series*