

Projet d'infographie

Projet final

Nom du projet : TARDIS HDRP

Github : <https://github.com/LoisonYo/TARDIS>

Proposé par : Yohann Loison

Classe : INF3dlm-b

Étudiants :

- Loison Yohann
- Ugo Crucy
- Laissue Luca

1. Introduction

Doctor Who est une série britannique de science-fiction diffusée depuis 1963. Cette série raconte l'histoire du Docteur, qui voyage à travers le temps et l'espace à bord de son vaisseau spatio-temporelle appelé le TARDIS.

Notre projet a pour but de redonner vie au TARDIS à l'aide de Blender et du *High Definition Render Pipeline* de Unity.

2. Objectifs

2.1. Objectifs principaux

- Animation des rotors centraux
- Animation des lumières murales
- Etat du TARDIS (à l'arrêt, en vol)
- Effet sonores
- Musique de fond

2.2. Objectifs secondaires

- Plus grand à l'intérieur
- Déplacement de la boîte

3. Attribution des tâches

Yohann Loison

- Modèle Blender
- Scripts
 - Traveling de la caméra

- Mouvement (Rotors et lumières)

Ugo Crucy

- Shader Graphs
- Matériaux

Luca Laissue

- Placement des lumières

4. Fonctionnement

4.1. Structure du projet

Deux dossiers d'assets sont importants dans notre projet. Le dossier Scenes et le dossier TARDIS.

Le dossier Scenes contient la seule et unique scène du projet. C'est cette scène nommée "Principale" que se trouve notre projet.

Le dossier TARDIS contient tout ce qui a un rapport avec le TARDIS (prefabs, scripts, matériaux, etc...).

4.2. Utilisation

Comme notre projet est une cinématique. Il suffit de sélectionner la scène "Principale" et de la lancer. Nous conseillons d'activer le son pour profiter de la musique ainsi que des effets sonores de la cinématique.

5. Points intéressants

5.1. Modèle Blender

5.1.1. Intérieur

Le modèle du TARDIS vient d'un projet personnel réalisé durant une semaine de vacances. Nous sommes repartis de ce modèle et l'avons grandement amélioré.



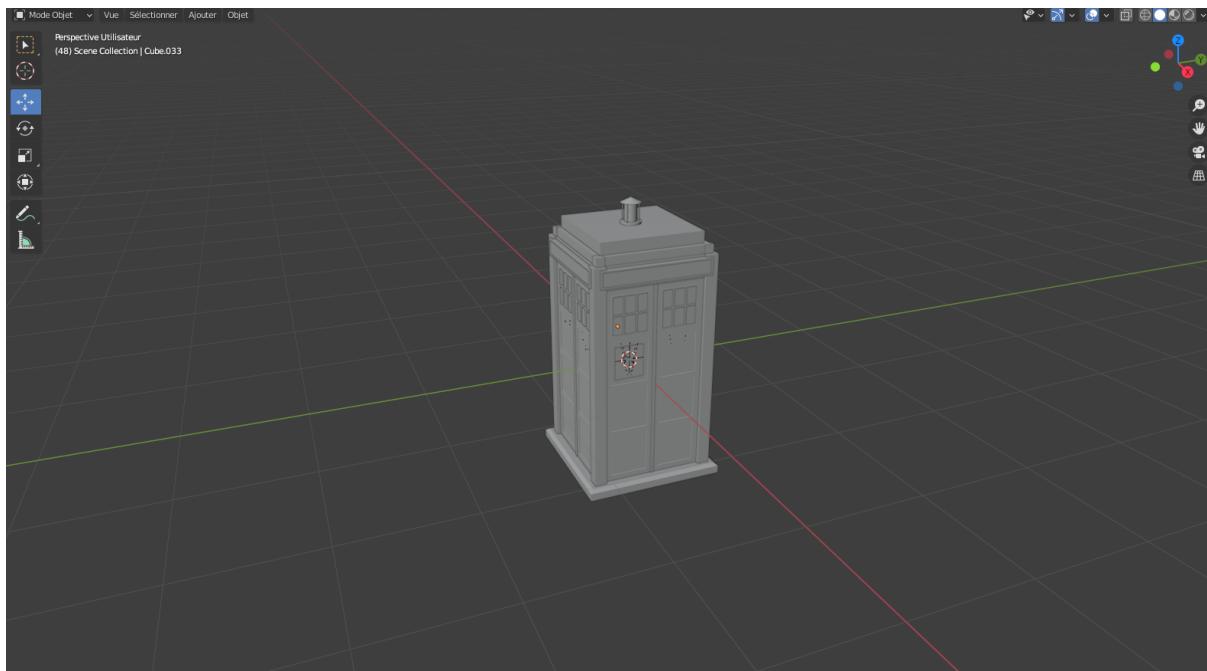
Modèle de l'intérieur du TARDIS sous Blender

Le modèle a entièrement été réalisé à la main. Aucun modèle 3D n'a été utilisé. Seules quelques images et vidéos sur internet ont été utilisées comme inspiration pour réaliser ce modèle.

Ce modèle est le fruit d'une première utilisation de Blender. Le résultat n'est pas parfait. En effet, certaines pièces (sur la console) ne sortent pas comme elle devrait sous Unity. Le résultat final est, à notre avis, plus que convenable pour une première utilisation de Blender.

5.1.2. Extérieur

En plus du modèle de l'intérieur, nous avons réalisé un modèle pour l'extérieur du TARDIS. Ce modèle avait pour but d'être utilisé pour réaliser les objectifs secondaires, mais il n'a jamais été utilisé par manque de temps.



Modèle de l'extérieur du TARDIS sous Blender

Ce modèle se trouve, tout de même, dans les prefabs du projet Unity.

5.2. Etat du TARDIS

Comme vous l'aurez peut-être remarqué dans la vidéo cinématique, le TARDIS possède plusieurs états : éteint, en vol, atterri.

Pour que chaque script puisse savoir dans quel état le TARDIS se trouve, il est nécessaire qu'ils aient tous accès à une variable pour connaître cet état. Pour régler ce problème, nous avons implémenté un Singleton nommé "TardisSingleton". Ce singleton contient un enum de l'état actuel du TARDIS comme attribut public.

Le script "TardisEventManager" va changer l'état du TARDIS après un laps de temps spécifique. Pour réaliser cela, nous avons utilisé une coroutine qui va attendre un temps spécifique et qui va modifier le singleton avec le nouvel état.

5.3. Traveling de la caméra

Comme notre projet est une cinématique, il est logique que la caméra ait plusieurs plans différents. Pour réaliser ça, nous avons principalement deux scripts C# : "CameraManager" et "CameraMovement".

"CameraMovement" est une classe virtuelle C# représentant un plan caméra. Chaque plan caméra va donc hérité de cette classe virtuelle et implémenter à sa façon les méthodes suivante :

- *bool IsFinished()*

Cette méthode retourne *True* si le plan caméra est terminé et *False* dans le cas inverse.

- *void Next()*

Cette méthode est appelée à chaque Update par le script “CameraManager”. Elle va appliquer le prochain “pas” du déplacement de la caméra.

- *CameraMovement GetNextMovement()*

Retourne le prochain plan caméra.

Grâce à cette approche la code du script “CameraController” est énormément simplifié :

```
public class CameraManager : MonoBehaviour
{
    private CameraMovement current;

    void Start()
    {
        current = new DoorMovement(this.gameObject);
    }

    void Update()
    {
        if (current.IsFinished())
            current = current.GetNextMovement();
        else
            current.Next();
    }
}
```

Le contrôleur appelle simplement la méthode “Next” de son instance “CameraMovement”. Dans le cas où le plan est terminé, le controller va simplement remplacer le plan actuel par le prochain plan caméra.

5.4. Gestion des lumières

La gestion de nos lumières se fait dans le script “LightController”. Ce dernier va simplement activer et / ou désactiver les lumières en fonction de l’état actuel du Singleton.

Chaque lumière est dans un “groupe” de lumière. Par exemple, les lumières du rotor (lumières oranges centrales) font partie du groupe “orangeRotorLight”. Afin de facilement retrouver toutes les lumières d’un groupe, nous avons mis un tag sur chaque lumières. Ainsi, nous pouvons utiliser la méthode : GameObject.FindGameObjectsWithTag("tag").

6. Problèmes rencontrés

6.1. Conversion du modèle

Lors de l'exportation du modèle Blender vers Unity, une infime partie des objets / parties du TARDIS ne sont plus placées correctement dans la scène (Position, rotation et même la forme).

Pour régler la majorité des cas, la parenté entre deux objets du modèle Blender cause ce problème. Pour régler le problème, nous avons simplement enlevé la parenté entre les objets puis nous avons re-exporter le modèle vers Unity.

Malheureusement, cette technique ne marche pas à tous les coups et il n'est donc pas possible d'avoir un rendu parfait. Pour une prochaine utilisation de Blender, il serait peut-être plus judicieux de séparer le modèle en plusieurs petites parties et assembler le toutes directement dans Unity.

6.2. Création de l'exécutable

Lors de la création de l'exécutable, Unity modifie l'ordre des objets dans la scène. Cela pose problème car nos lumières murales (celles qui tournent) fonctionnent en fonction de leurs ordres dans la scène. Comme l'ordre est chamboulé lors de la création de l'exécutable, nos lumières ne fonctionnent pas correctement.

A ce jour, nous n'avons pas encore trouvé de solution. Il est donc recommandé de regarder la vidéo directement pour un rendu le plus proche possible de ce qu'on a réalisé.



Lumières murales corrects



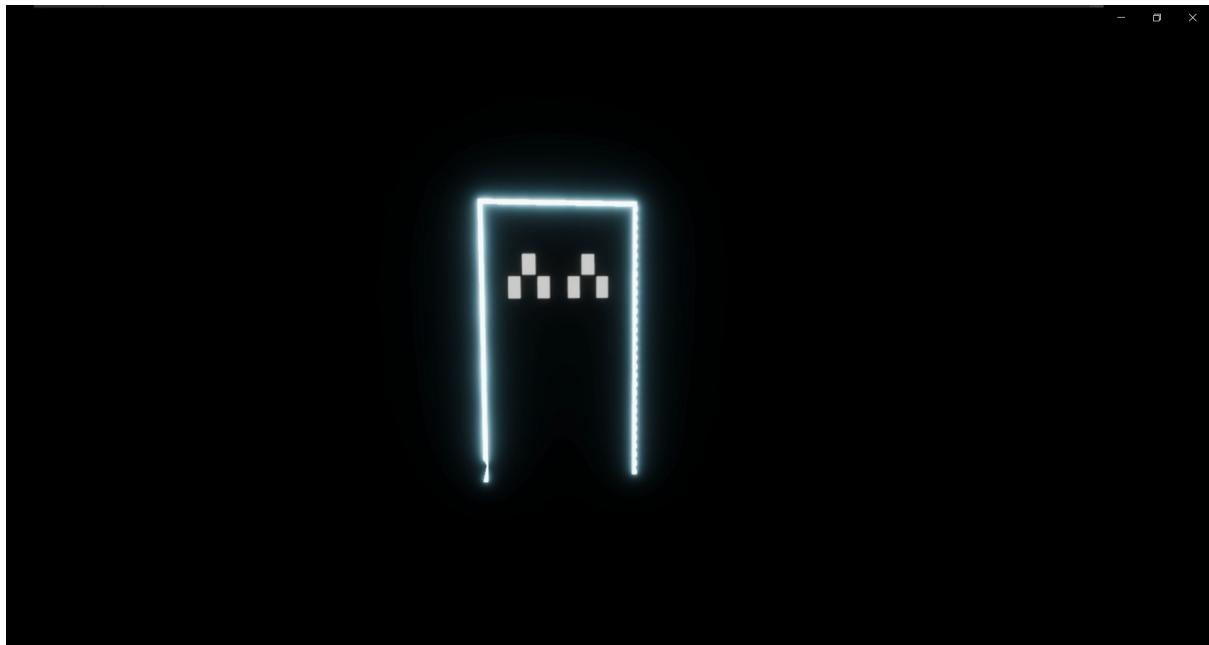
Lumières de l'exécutible

7. Conclusion

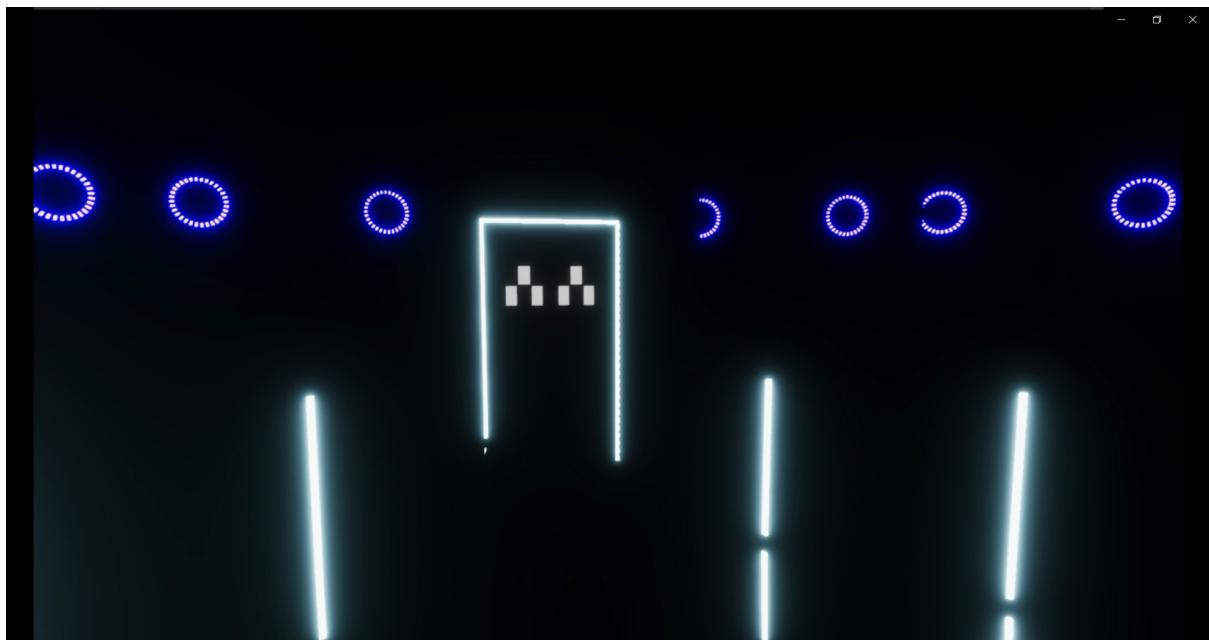
Pour conclure, nous avons réalisé tous les objectifs principaux que nous nous sommes fixés au début du projet. Nous avons une cinématique dans laquelle chaque aspect du projet est montré : Sound effects, Musique de fond, état du TARDIS (éteint, posé, en vol) et animations.

Pouvoir voir l'intérieur du TARDIS depuis l'extérieur serait un des objectifs secondaires à implémenter en priorité dans le futur. Avoir cet effet de "plus grand à l'intérieur" pourrait être une addition franchement très sympa au projet.

8. Screenshots



TARDIS éteint



Démarrage du TARDIS



TARDIS allumé



TARDIS en vol



Console du TARDIS