TX - STRESS
DOCUMENTATION TECHNIQUE

TABLE DES MATIERES

AVANT TOUT

1.1 INSTALLATION D'UNITY ET METTRE EN PLACE DE PROJET

1. Installer Unity version **5.0+** : https://store.unity.com/download

2. Télécharger le package Unity de projet sur : https://cloud.utc.fr/public.php?service=files&t=cba6a6cb321764ae0bbb3d7f1151a5ce

   ou télécharger le projet sur : https://github.com/yinsimei/TX_Stress_Environment_2016

Remarque : Bien que le package Unity contient tous les éléments essentiels, il ne contient pas la configuration du projet (« Project Settings »). Donc, en l'utilisant, on risque d'avoir des problèmes inattendus par rapport à ce qui est configuré dans le « Project Setting » (tag, input, lighting...etc.) Personnellement, je conseille d'importer le projet de github au lieu de package Unity pour le développement en suite.

3. Importer le package Unity / le projet

4. Aller dans *Asset/Scene* et ouvrir la scène StressEnvironment

Remarque : Le projet exige Unity d'une version supérieure à 5.0, sinon il y aura un crash pendant le chargement de fichiers

1.2 MISE EN PLACE DE LOGITECH G27

Afin d'utiliser *Logitech racing wheel G27*, il faut :

1. Mettre en place les périphériques en suivant le guide suivant : http://www.logitech.com/assets/47059/g27-racing-wheel-quickstart-guide.pdf

2. Installer le logiciel sur son PC : http://support.logitech.com/en_us/product/g27-racing-wheel

Remarque : L'asset « Logitech Gaming SDK [1]» a déjà été importé dans le projet mais seulement sous un environnement sous **x64**. Si vous travaillez sur un environnement sous x32, vous devrez importer l'asset dans votre projet Unity et supprimer les dossiers x64 dans *Logitech SDK/Lib/*. Il est cependant fortement déconseillé de faire cela car cela peut poser des problèmes. Il vaut mieux travailler sous un environnement **x64**.

[1] https://www.assetstore.unity3d.com/en/#!/content/6630

APERÇU DE CODE

Généralement, le code de ce projet peut être décomposé selon les parties suivantes :

1. **Contrôle de la voiture (input avec Logitech G27, HTC Vive à venir...)**
2. **Système d'évènements stressants (Mise en place et déclenchement des évènements)**

1. CONTROLE DE LA VOITURE

Tous les scripts concernant la voiture se situent dans le dossier : *Assets/Scripts/Car/*. Ce sont généralement des scripts provenant de l'Asset Standard de Unity « Vehicule » mais avec nos propres modifications.

1.1 INPUT AVEC LOGITECH G27

RECUPERATION D'INPUT

La récupération d'input de volant et pédales de Logitech G27 est faite dans le script : *Assets /Car/Scripts/CarUserControl.cs* grâce au Logitech Gaming SDK.

Toutes les informations concernant les volant et pédales peuvent être récupérées de la manière suivante :

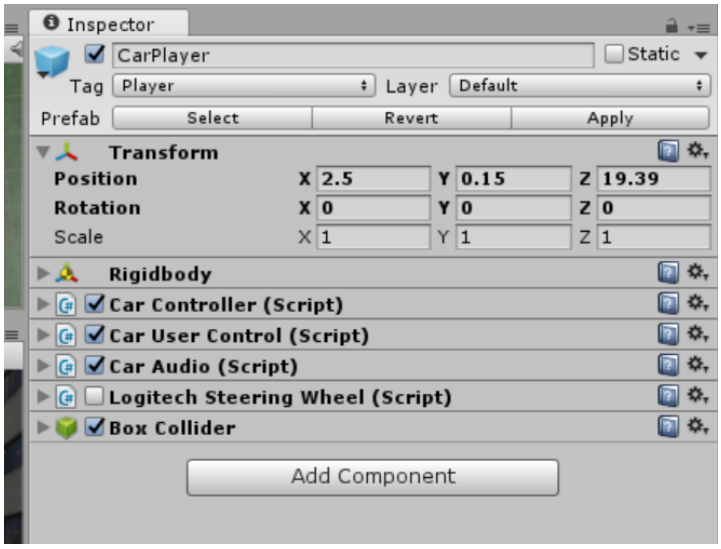
```
DIJOYSTATE2ENGINES rec = LogitechGSDK.LogiGetStateUnity(0);
```

On a utilisé des champs suivants de `rec`:

```
rec.lX;                  /* x-axis position            */
rec.lY;                  /* y-axis position            */
rec.lRz;                 /* z-axis rotation            */
```

Les valeurs retournées sont des entiers compris entre `short.MinValue et short.MaxValue.`

L'activation du script *LogitechSteeringWheel.cs* sur CarPlayer qui affiche tous les champs de rec, permet d'avoir une idée plus claire du fonctionnement générale.



1.2 CONTROLE DE CAMERA

Le script *Assets/Scripts/MouseLookCamera.cs* permet de tourner la caméra selon la position de la souris sur l'écran, et donc ce que pointe l'utilisateur. Si, dans le futur, on veut travailler avec le HTC Vive, il serait nécessaire de renommer et modifier ce script.

2. SYSTEME D'EVENEMENTS STRESSANTS

Tous les scripts concernant le système d'événements stressants se situent dans le dossier suivant: *Assets/Scripts/StressEventSystem*.

2.1 COMMENT ÇA FONCTIONNE

2.1.1 STRESS EVENT MANAGER

Dans le Game Object StressEventManager, dans la scène, il y a un script singleton *StressEventManager.cs* qui permet de déclencher ou annuler un évènement via ses APIs :

```
StressEventManager.instance.StartEvent(eventName);.
```

```
StressEventManager.instance.EndEvent(eventName);.
```

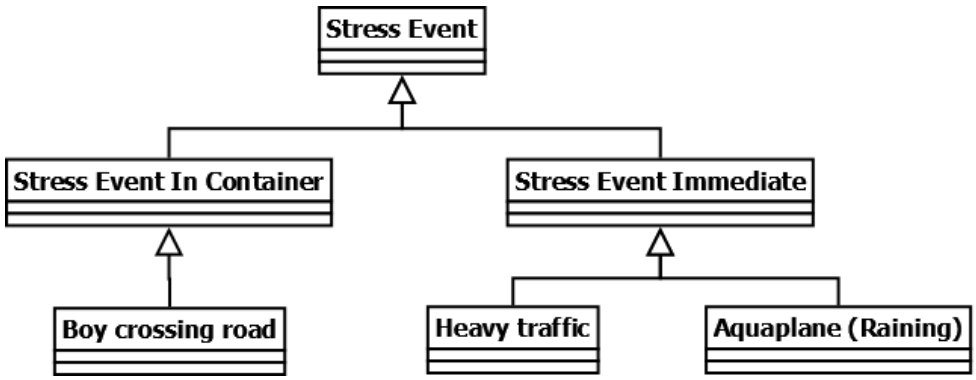
2.1.2 EVENEMENTS STRESSANTS

Il y a deux types d'événements stressants :

- Ceux qui démarrent immédiatement (par exemple : la pluie qui tombe) ;

- Ceux qui attendent la voiture dans une zone de « trigger » (déclenchement) pour démarrer (par exemple : le garçon qui traverse la rue quand la voiture s'approche)

Donc on a une hiérarchie de classes comme ci-dessous :



- `StressEvent` : Classe la plus haute au niveau hiérarchique qui représente généralement des évènements stressants. Elle est définie dans *StressEvent.cs.*

- `StressEvent_Immediate` : Cette classe représente des évènements du premier type qui peuvent démarrer tout de suite après l'appel de `StartEvent(string eventName)` de « StressEventManager « par l'utilisateur, afin de déclencher cet évènement. Cette classe est simple et définie dans le même script que `StressEvent` comme elle ne contient qu'un constructeur.

- `StressEvent_InContainer` : Cette classe représente des évènements du deuxième type qui attendent que la voiture s'approche pour démarrer. Cette classe est relativement compliquée et est définie dans le script *StressEvent_InContainer.cs*. Quand l'utilisateur apelle `StartEvent(string eventName)` de « StressEventManager » pour le déclencher, le « StressEventManager » va chercher le « StressEventContainer » qui est à la distance le plus raisonnable de la voiture de l'utilisateur, puis y insère cet évènement. Ensuite, cet évènement attend le rapprochement de la voiture pour démarrer.

- `StressEventContainer :` C'est un endroit où un évènement de type `StressEvent_InContainer` peut être inséré. Il en existe un très grand nombre dans la scène. Ils sont regroupés dans le Game Object *StressEventContainers* (fils de Game Object *StressEventManager*).

2.2 PROCEDURE DE MISE EN PLACE D'UN EVENEMENT STRESSANT

ETAPE 1 : CRÉATION D'ÉVÈNEMENT

STRESS EVENT_IMMEDIATE

- La mise en place d'un évènement de type `StressEvent_Immediate` est très simple. Il suffit de créer une classe qui hérite de `StressEvent_Immediate,` puis override les fonctions : `StartEvent()` et `EndEvent()`.

- Ajouter ce script dans le Game Object qui correspond.

Exemple : *Events/StressEvent_Aquaplane.cs*, *Events/ StressEvent_Traffic.cs*

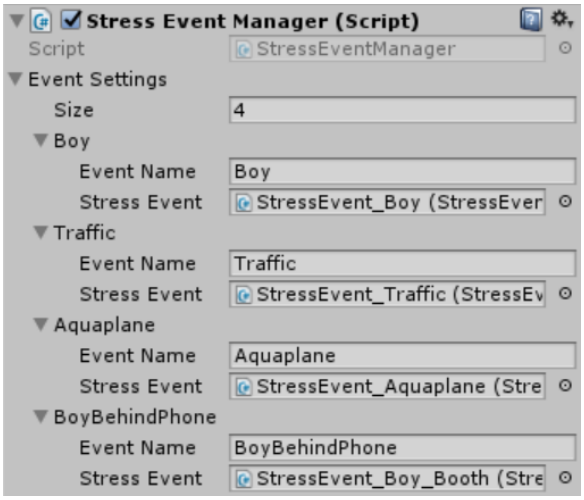
STRESS EVENT_INCONTAINER

- La mise en place d'un évènement de type `StressEvent_InContainer` est un peu différente. Il faut créer une classe qui hérite de `StressEvent_InContainer` et puis override les fonctions : `StartContainerEvent()` et `EndContainerEvent()`.

- Ajouter ce script dans le Game Object qui correspond.

Exemple : *Events/ StressEvent_Boy.cs*

ETAPE 2 : AJOUT D'EVENEMENT DANS STRESS EVENT MANAGER



1. Augmenter le paramètre « Size » dans « Event Setting » du script « StressEventManager », il y aura alors un nouvel évènement qui apparaît en bas.

2. Dans le champ « Event Name », donnez un nom **unique** à cet évènement, qui permettra de déclencher cet évènement par son nom.

3. Dans le champ « Stress Event », faire glisser le Game Object où se situe le script qu'on vient de créer pour l'y ajouter.

ETAPE 3 : DECLENCHEMENT D'UN EVENEMENT

Normalement, le déclenchement des évènements peut être faite n'importe où via les APIs de « StressEventManager » :

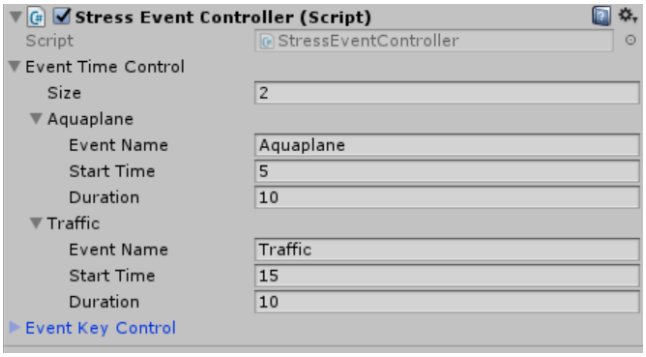
```
StressEventManager.instance.StartEvent(eventName);.
```

```
StressEventManager.instance.EndEvent(eventName);.
```

Nous avons écrit un « StressEventController.cs » qui est ajouté dans le Game Object StressEventManager.

On a mise en place deux genres de système du déclenchement/de l'annulation des évènements :

Lancement/annulation au bout d'un certain nombre de secondes :



Lancement/annulation par l'appuie sur une touche au clavier :

Des « Event Name » ici son ceux qu'on a mis dans le StressEventManager.

4. VIDEO DE TUTORIEL

Nous avons réalisé des vidéos « tutoriel » qui expliquent tout le fonctionnement du projet à partir de l'importation du package Unity jusqu'à la création et le déclenchement d'un évènement.

Playlist « TX Stress_Tutoriel » sur Youtube :
https://www.youtube.com/watch?v=grQB0huuOZ4&list=PL8BNwwCeojJ5fg-O7yxk2d2T56LCJHqyv