Réunion

Objet

Cette réunion a pour objectif de clarifier les doutes et incompréhension de Théo dans l'utilisation de Visual Studio, UE4 et AirSim. Charles, qui travaille depuis un bout de temps avec ces outils pour la navigation autonome de drone, a accepté de participer à une visio conférence pour apporter son aide.

Traces

Un **CR écrit** sera produit à l'issu de la réunion.

Contenu de la réunion

Visual Studio

Description

Microsoft Visual Studio est un IDE permettant l'édition et l'exécution de code dans un environnement géré.

Charles travaillant essentiellement sous Linux maintenant ne connait pas beaucoup Visual Studio et ses connaissances en l'outil sont limitées.

Questions

```
1. Quand un build un projet .sln on compile des _binaries_ qu'il faut
déplacer dans notre projet UE4 en tant que plugin. Mais quand on veut
exécuter un script via API (ex: DroneShell), on 'démarre le debogage' de
'solutions'.
Quelles sont les différences entre les binaries, les solutions ? Comment
communiquent-ils ?
```

Une 'solution' semble être un dossier composé de fichiers sources et de tout ce qu'il faut pour pouvoir compiler son projet. Qu'il le soit ou non, Visual Studio le voit de la même manière et permet de l'exécuter.

Les solutions, une fois construites, communiquent avec la simulation via le protocol RPC.

```
2. Au lancement de la solution PythonClient, j'ai des échec de lancement
pour cause de fichiers python absents (ex:
PythonClient\computer_vision\character_control.py).
Une idée de pourquoi ?
```

Peut-être qu'à cause des environnements Python il y a confusion entre le paquet `airsim` et le projet Airsim issu du dépôt, mais dans tous les cas on peut simplement appeler les scripts manuellement au terminal, ça

marche.

Informations diverses

Charles a instinctivement commencé à manipuler UE4 sous Windows avant de passer sous Linux en faisant tourner l'éditeur via Wine.

UE4

Description

UE4 est un moteur de jeu permettant la modélisation de mondes réalistes, assujettis aux lois de la physique et de la lumière que l'on connait. Son éditeur est utile dans nos projets respectifs à Charles et moi pour offrir un monde dans lequel naviguer en drone simulé et un grand nombre de scènes forestières proches de la réalité.

Il est important de préciser que l'orsque l'on parlera d' "images 360" dans cette partie, on fait référence à des images equirectangulaires construites à partir de *cubemaps* qui sont une représentation de l'espace cubique avec 6 vues perspectives que l'on reconstruit ensuite sur une sphère puis en vue equirectangulaire.

Questions

```
1. J'ai du mal à comprendre la console de l'éditeur. Via le plugin python
on peut exécuter des commandes ou fichiers python sous réserve qu'on ai
importé les package nécessaires.
Mais à quoi correspond la console de base 'cmd' ?
```

Le mode 'cmd' du Terminal (*Console log*) permet uniquement d'appeler des fonctions de l'éditeur d'UE4, ce n'est pas un terminal comme on peut l'entendre au niveau OS.

```
2. Je voudrais mieux comprendre l'architecturation des assets, pour un
arbre donné par ex.
    a) Chaque arbre est-il identifiable individuellement ou par groupe ?
```

On a un nombre limité d'arbres différents dans Redwood qui servent à une génération procédurale de la forêt. Par conséquent on ne peut pas identifier chaque arbre individuellement (problème analogue à celui de Dao).

```
    b) Peut-on attribuer une texture random à chaque arbre individuel ?
```

On peut attribuer une texture à chaque prototype d'arbre, donc on peut imaginer plaquer une couleur unie et éteindre les effets de lumière dans la simulation pour obtenir naturellement un 'monde label'. On peut également enlever le feuillage des arbres en paramétrant les *mesh* pour ne garder que les troncs..

Informations diverses

La segmentation et la profondeur exraite par Charles correspond à celle des mini fenêtres de AirSim où la végétation est d'une seule couleur (il est également possible de le faire sur des images 360). En suivant un tuto du MIT, Charles a développé un modèle qui segmente bien sur des img equirectangulaires et donne des résultats acceptables sur des images réelles.

AirSim

Description

AirSim est un plugin à l'éditeur d'UE4 développé par Microsoft, inspiré d'un autre plugin plus ancien UnrealCV, et permettant de piloter un drone simulé ainsi que de bénéficier de certaines applications de Computer Vision telles que la segmentation, la capture stéréo etc...

Questions

```
1. Bien qu'on ne place dans notre projet UE4 que les binaries issus du
build, il reste dans le dépôt les fichiers source et leurs exécutables. Le
projet UE4, lorsque joué ('Play'), devient le server et les exécutables
(API) deviennent le client. Le client peut envoyer des requêtes au serveur
pour commander le drone. Correct ?
```

Oui, que cela soit la simulation de l'éditeur ou celle du *package* de l'environnement UE4 qui ne requiert pas l'éditeur.

De plus, indépendamment d'où est localisé le dépôt AirSim, un fichier de config .json est initialisé par défaut au chemin `Documents/airsim/settings.json` et permet de *preset* certains modes comme le mode ComputerVision ainsi que d'autres éléments de la simulation. Les informations contenues dans ce fichier viennent *overwrite* le monde tel qu'il est dans l'éditeur avant le lancement de la simulation. Par exemple, on pourrait via ce fichier, définir des caméras ou des drones supplémentaires sans les ajouter de manière permanente au projet UE4, qui ne seraient donc présents que pendant l'exécution de la simulation.

```
2. Les API C++ possèdent un terminal attendant des instructions ce qui est
pratique pour une utilisation on-the-fly mais pas les scripts python.
Est-ce que néanmoins c'est techniquement possible ?
```

(?) Charles a développé des scripts sans interaction clavier à ma connaissance.

```
3. Les API C++ ne me permettent pas de faire autre chose que de me déplacer
sur z (la hauteur), que je les lance à la main ou via Visual Studio.
Une idée de pourquoi ?
```

Pas de réponse, Charles s'est concentré sur l'API Python. Mais peut-être que Rida pourrait me renseigner.

```
3. Peut-on capturer pendant pilotage manuel du drone ?
```

On peut sûrement *run* un script en arrière-plan sur une longue période de temps pour qu'il capture pendant qu'on pilote mais la configuration de la manette, d'après Charles, est très complexe à mettre en oeuvre.

Pour ses captures, Charles exécute un script ordonnant au drone de choisir des coordonnées aléatoires sur une grille de 200x200 auxquelles se déplacer, mais cela fait que des fois le drone passe à travers les textures ou le sol ou bien se cogne contre les éléments 'solides'. La collision n'arrête cependant pas la simulation : le drone peut traverser l'élément solide (via la commande `setvehiculepose`) ou s'y cogner (via la commande `movebyvelocity`).

Informations diverses

- Des fois il faut rajouter `.join()` à la fin de requêtes API pour les déplacements (cela remplit une pile d'action). Charles *freeze* aussi la simulation quand il prend ses *screens* 360 car cela prend du temps.
- Par package ou par l'editeur, AirSim fonctionne pareillement, pas besoin de modifier les ports et les IP.
- Il est impossible de prendre des captures directement en equirectangulaire il faut passer par de la perspective (cf. partie description de la section UE4). Dans l'API : utiliser `ImageRequest(NUM_CAMERA, ...)`.
- Rida, lui, a modifié les fichiers C++ et peut être en mesure de me renseigner si j'ai des questions.
- On peut simuler des captures stéréo en décalant la caméra du drone au moment des appels de capture par l'API ou bien choisir 2 des 4 caméras présentes par défaut sur le drone (*top*, *bottom*, *right*, *left*). Pour modifier ces caméras par défaut, il est possible de passer par les fichiers C++ ou directement par l'editor.
- Un objet *playerstart* dans le monde fera démarrer le mode *Computer Vision* dessus. Il est d'ailleurs recommandé de ne garder que celui-ci comme *start*.
- Pour éviter les problèmes de reconstruction au niveau des lumières il faut mieux simplifier l'éclairage en désactivant les propriétés suivantes dans les *project settings* :
    - *auto-exposure*
    - *atmospheric fog*
    - *lens flare*
    - les aberrations de lentille et autres effets dans *post-process volume*
- Les images equirectangulaires sont construites à partir de *cubemaps* mais attention avec les fonctions `ImageRequest` appelant le type `DepthPerspective`, il faut veiller à renseigner les aux booléens d'`ImageRequest` derrière le type.
- Quelques références sur lesquelles Charles s'est basé :

    - https://pypi.org/project/cube2sphere/ : marche via Blender. Ce que charles fera : définir un tableau possédant des valeurs de passage de coordonnées cubiques à sphériques car les images sont toutes de même taille, car le calcul du passage de coorodnnées est coûteux.
    - https://github.com/PaulMakesStuff/Cubemaps-Equirectangular-DualFishEye

- Les fichiers `.pfm` de charles comportent les informations de profondeurs, ce n'est pas un collage *side by side* d'une vue stéréo.

- Dans `ImageRequest` en fonction du numéro de la caméra (0 à 4) on aura les cam *top*, *bottom*, *right* ou *left*.

- L'équipe de Nice possède des réseaux d'entraînement sur des images équirectangulaires qui adaptent la forme des masques de convolution pour avoir des approximations de perspectives plans qui collent aux déformations.

Actions

- Guillaume doit faire ces acquisitions mais pas encore. Charles va prochainement prendre des nouvelles de l'avancement de cette tâche.

- Théo va demander à Pascal si on peut partager le rapport de Dao à l'équipe de Nice car la segmentation des arbres par Dao intéresse Charles.

- Si Théo arrive à modifier les caméras de base pour les placer à l'endroit souhaiter, que cela soit par l'éditeur ou AirSim, il en informera Charles. Cepndant il faudra peut-être re-créer un drone permanent dans l'éditeur et modifier le fichier `config.json`.
  [Tâche non prioritaire]