LOG8715

TP3: Prédiction client

Version 2.0

Responsable: Olivier Gendreau

Chargé de laboratoire: Louis-Philippe Lafontaine-Bédard

Auteurs: Samuel Bellomo, Martin Paradis et Louis-Philippe Lafontaine-Bédard

Introduction

L’objectif de ce laboratoire est de comprendre les principes de bases de prédiction client. Pour ce faire, vous devrez implémenter une technique vue en cours sur une simulation simple synchronisée entre client-serveur. Vous aurez une implémentation client-serveur déjà fournie et devrez faire en sorte que même avec du délai, les clients et serveurs soient synchronisés.
Lorsqu’un des clients applique un input, les autres clients devront réconcilier leur état avec les données du serveur.
L’implémentation sera faite dans l’engin Unity avec le projet de base fourni sur Moodle.

Travail à accomplir

Le travail sera fait en équipe de deux.

Implémentation

Installation

Si ce n'est pas déjà fait, installez **Unity 2019.4.16f1** et ouvrez le projet pour la plateforme Windows. Veuillez utiliser la version d'archive disponible au site: https://unity3d.com/get-unity/download/archive

Assurez-vous d'utiliser la version spécifiée d'Unity, étant donné qu'une version différente demanderait une migration du projet. Des points seront enlevés si la version est différente.

Ouvrez la scène `Scenes/MainScene.unity`

Instructions pour faire fonctionner clients et serveurs

Vous devrez avoir une deuxième instance du jeu qui fonctionne en même temps que l'éditeur afin d'avoir un client et un serveur. Pour ce faire, vous devrez générer un « build ». Vous pourrez ensuite ouvrir parallèlement une ou plusieurs instances de votre build en plus de l'éditeur, pour en faire un serveur et un ou des clients.

Mode client et serveur

En démarrant la solution, vous avez deux options. Démarrez la simulation en tant que serveur ou client via l'interface utilisateur ou avec les raccourcis clavier (C et S)

Latence simulée

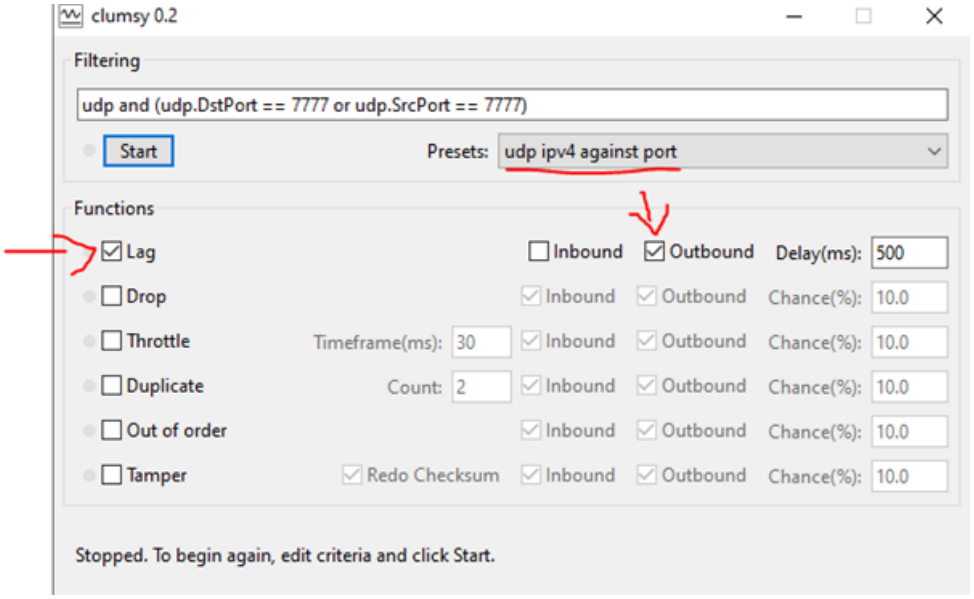
Pour évaluer votre système de prédiction client, une latence sera ajoutée. Lors de la correction, votre solution sera évaluée avec des latences variant de 0 à 1s.

Windows

Sur Windows, vous pouvez utiliser l'outil Clumsy pour générer cette latence simulée. Vous pouvez le télécharger ici : https://jagt.github.io/clumsy/download.html

Son utilisation nécessite des privilèges administrateurs, il ne pourra donc pas être utilisé sur les postes de Polytechnique.

Lorsque vous l'exécutez, utilisez les paramètres suivants :



Vous pourrez modifier la valeur du champ « Delay(m/s) » pour spécifier la latence à simuler en ms. Cliquez sur Start pour activer la latence simulée, puis stop pour la désactiver.

Linux:

Si vous utilisez Linux, vous pouvez utiliser netem pour l'ajout de latence simulée. Voir https://wiki.linuxfoundation.org/networking/netem, https://man7.org/linux/man-pages/man8/tc.8.html et https://man7.org/linux/man-pages/man8/tc-netem.8.html pour plus de détails.

Par exemple, pour ajouter un délai de 100ms sur l'interface de rebouclage (loopback):

```
tc qdisc add dev lo root netem delay 100ms
```

Puis pour retirer ce délai:

```
tc qdisc del dev lo root netem
```

Fonctionnalités déjà existantes

La simulation contient déjà une implémentation de base.

Description de la simulation déjà existante:

- La simulation contient des carrés qui évoluent à l'écran.
- Les carrés se déplacent à l'écran selon leur vitesse
- Les carrés entrent en collision avec les bords de l'écran, mais pas entre eux.

    - Il n'y a pas de restitution au contact d'entités. Lors d'un contact, la vitesse des entités reste inchangée.
- Les entités sont instanciées par le serveur, qui a autorité sur elles
- Lorsqu'un client se connecte, une entité devient une entité « joueur » :
    - Elle prend l'apparence d'un cercle
    - Les entités « joueur » peuvent entrer en collision avec d'autres entités

API fourni

Le projet de base est implémenté en suivant le modèle ECS. Il utilise la librairie de haut niveau MLAPI pour la communication réseau. Voir https://mlapi.network/wiki/home/ pour la documentation.
L'**ECSManager** vous permettra d'interfacer avec Unity. Il contient des méthodes permettant de créer et détruire des entités à l'écran ainsi que mettre à jour leurs positions et leurs tailles.
Le **CustomNetworkManager** s'occupe de la gestion du réseau. Les paramètres sont déjà définis dans l'éditeur, vous n'avez pas à les modifier. Le code pour la gestion des messages de réplication est fourni. Lorsque vous ajouterez d'autres types de messages, vous pourrez vous baser sur cette structure.
Plusieurs systèmes et composants sont fournis. Vous pouvez les modifier et en ajouter d'autres.

Fonctionnalités à développer

Input

Un client peut utiliser les touches A, S, D et W pour déplacer l'entité joueur qui lui est attribuée. Ce déplacement doit être sous la forme d'une vitesse constante qui est appliquée selon la direction indiquée par les touches. Si aucune touche n'est appuyée, l'entité a une vitesse nulle.

Prédiction

Votre objectif est d'implémenter un système de prédiction et de réconciliation client à l'aide de l'extrapolation et de la prédiction d'inputs.
Un client devrait voir déplacer l'entité joueur en temps réel, peu importe la latence ajoutée, et interagir avec les entités prédites. Les entités devraient ensuite être réconciliées au besoin.

Ces deux systèmes doivent pouvoir être activés ou désactivés indépendamment à l'aide des champs "enableInputPrediction" et "enableDeadReckoning" de la configuration.
Veuillez prendre note qu'il n'est pas nécessaire d'implémenter les autres techniques vues en cours, telles que les techniques de fiabilité et d'interpolation. Seule la prédiction et la réconciliation client seront évaluées. C'est donc normal si une entité se téléporte parce que sa position a été corrigée.

Restrictions

Vous devez implémenter votre prédiction et réconciliation client vous-mêmes. Vous ne pouvez pas utiliser les solutions fournies par Unity.
Bien que le projet de base soit implémenté selon le modèle ECS, vous pouvez en déroger si vous jugez que c'est pertinent. Votre code doit tout de même rester clair et maintenable.

Configuration

Vous n'avez pas besoin de modifier le fichier de configuration. Il sera remplacé lors de la correction.

Rapport

Format PDF, interligne 1.5, 12 pts
Max 300 mots

1. Vous devrez rendre une courte discussion sur votre implémentation. À noter qu'une discussion implique de justifier vos points. Il vous est donc demandé de décrire votre implémentation, de la justifier et de discuter de la maintenabilité de votre solution.
2. La compensation de latence par « input prediction » prédit l'état du joueur du côté client, mais pas celui du reste des entités. L'extrapolation, quant à elle, sert à prédire l'état de la simulation en général, mais ne prend pas en compte les entrées du joueur. Quelles conséquences l'utilisation d'un de ces systèmes sans l'autre entraîne-t-il pour le client? Vous pouvez utiliser le présent TP comme exemple.

Évaluation (/15)

Implémentation	
Rapport	3
Respect des requis	4
Input prediction	4
Extrapolation	4
Manque de clarté/maintenabilité	-3
Projet ne compile pas	-10
Mauvais format de remise et langue	-3
Retards	Perte de points exponentielle. jour 1: -2 jour 2: -4 jour 3: -8 **0 après 3 jours de retard**

Remise

Votre dossier de remise devrait contenir votre projet Unity. Le nom du fichier zip devrait contenir les matricules des coéquipiers.

LOG8715_TP3_matricule1_matricule2.zip

|___ ProjetUnity

|____ Assets/…

|____ ProjectSettings/…

|____ ….

Ne pas inclure le dossier Library et autre fichiers générés. Pour une liste complète de fichiers à éviter, voir le .gitignore à

https://github.com/github/gitignore/blob/master/Unity.gitignore

Si votre projet utilise git, vous pouvez utiliser la commande git

```
git archive -o TP3.zip HEAD
```

Pour créer votre archive zip.

Le projet ne devrait pas être très lourd, veuillez remettre le zip sur Moodle avant la **date indiquée sur Moodle.**