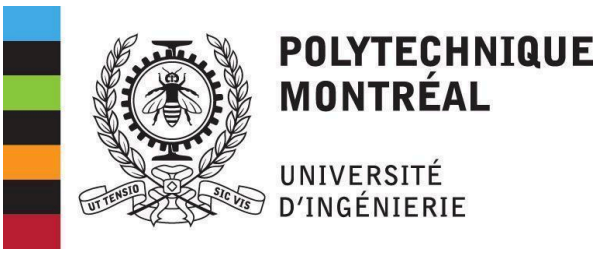

LOG8715

TP1: Conception ECS

Version 4.5

Responsable: Olivier Gendreau

Chargé de laboratoire: Taha Lazreg

Auteurs: Samuel Bellomo, Louis-Philippe Lafontaine-Bédard et Keven Chaussé

Introduction

Ce laboratoire vise à mettre en pratique le modèle ECS dans une simulation de jeu physique. Vous serez amenés à développer certaines fonctionnalités pour tester l'interaction entre les systèmes. Le travail se fera en **équipe de trois**, divisé en deux volets principaux: **l'implémentation** et la **rédaction d'un rapport**.

L'objectif est de vous faire explorer la maintenabilité de l'architecture ECS. Vous n'aurez donc pas à l'optimiser pour exécuter des milliers d'entités. Le projet doit quand même fonctionner à un minimum de 30FPS.

Environnement de développement

Moteur de jeu

Unity pour ses fonctions de base, telles que l'affichage de graphiques, la compilation, la configuration, sa lecture d'assets, etc. Vous ne toucherez pas à l'éditeur à part pour tester votre jeu.

Classes fournies

Quelques classes seront également fournies pour vous permettre d'interfacer votre code avec Unity. Ces classes, la configuration du projet et la scène Unity d'origine seront écrasées par les originaux lors de la correction. **Vous ne devez donc pas les modifier**.

Langage de programmation

L'implémentation doit être faite totalement en C#.

Restrictions

Vous êtes autorisés à utiliser les bibliothèques utilitaires d'Unity telles que Mathf, Time, Input, Screen, etc. L''utilisation des classes principales d'Unity liées aux entités, objets de jeu et physique est strictement interdite. Cela inclut, mais n'est pas limité à, MonoBehaviour, Component, Physic, Transform, etc. En cas de doute, contactez votre chargé de laboratoire, par courriel ou directement pendant les séances de laboratoire.

Travail à accomplir

Formation des équipes

Utilisez l'activité **Formation des équipes** sur moodle pour joindre un groupe. Vous devez avoir formé vos équipes avant la date limite qui est la date de remise du TP1.

Partie 1 - Implémentation

Installation

Si ce n'est pas déjà fait, installez **Unity 6000.0.65f1** et ouvrez le projet pour la plateforme Windows. Si cette version n'est pas disponible dans le launcher Unity Hub, 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.

Configuration et test de votre implémentation

- Modifier la configuration en cliquant sur le fichier de Configuration Config/Config.asset.
    - L'interface de configuration apparaît à droite de l'éditeur Unity dans l'inspecteur. Vous pouvez modifier les paramètres de simulation ainsi que les cercles à faire apparaître au lancement du jeu.
- Ouvrez la scène `Scenes/MainScene.unity` et appuyez sur le bouton Jouer. Cliquez à nouveau sur ce bouton pour arrêter votre jeu. 
- `Par eux-même, ces éléments de configuration ne font rien. Vous devez les utiliser dans votre implémentation. Vous pouvez récupérer les valeurs de la configuration en utilisant` **`ECSController.Instance.config`**

Fonctionnalités

Vous devrez faire évoluer des cercles à l'écran avec une architecture ECS. Les fonctionnalités suivantes devront être implémentées:

- Les cercles ont des tailles différentes représentées par un nombre entier.
- Certains cercles sont dynamiques. Ils ont une vitesse et se déplacent à l'écran.
- Certains cercles sont statiques et ne bougent pas. Ces cercles représentent des obstacles.
- Lors d'un contact entre deux cercles, plusieurs choses se produisent.
    - Les cercles rebondissent en utilisant la méthode CalculateCollision dans la classe utilitaire CollisionUtility. Celle-ci prend en paramètre la position, la vitesse et la taille des deux cercles et retourne leur position et vitesse résultante.
    - Si les cercles sont de tailles différentes, le cercle le plus gros augmente sa taille de 1 et le cercle le plus petit diminue sa taille de 1.
    - Si les cercles ont la même taille, leur taille ne change pas.
    - Une collision entre un cercle statique et dynamique n'affecte pas leur taille. Seul le cercle dynamique rebondit, le cercle statique ne bouge pas.
- Lorsqu'un cercle atteint la taille 0, il est détruit.
- Lorsqu'un cercle atteint une taille d'explosion définie dans la configuration, il explose en quatre cercles différents
    - Chacun des quatres cercles se déplace dans une diagonale différente. 
    - La taille des quatre cercles résultants est le quart du cercle initial. (Un cercle de taille 8 devient quatre cercles de taille 2)
    - La plus petite taille possible pour les quatre cercles est 1. (Un cercle de taille 3 devient quatre cercles de taille 1)
- Lorsqu'un cercle entre en collision avec un bord de l'écran, il rebondit.
- En dessous ou égale à une taille définie dans la configuration, un cercle dynamique peut devenir protégé après un certain nombre de collision avec des cercles de même taille. On compte le nombre de collision à partir du moment où le cercle est en dessous ou égale à une taille définie. Cette taille sera toujours plus petite que la taille d'explosion.
    - Le nombre de collisions avant de devenir protégé est définie dans la configuration.
    - Le cercle protégé ne peut pas changer de taille.
    - Les cercles plus petits qui entrent en collision avec le cercle protégé ne changent pas de taille.
    - Les cercles plus grands qui entrent en collision avec le cercle protégé diminuent leur taille de 1.
    - Le cercle cesse d'être protégé après un délai défini dans la configuration
    - Après avoir été protégé, un cercle ne peut pas redevenir protégé tant qu'un cooldown défini dans la configuration ne s'est pas écoulé
- Les cercles devront avoir une couleur qui change selon leur état (De la couleur plus prioritaire à la moins prioritaire).
    - Les cercles statiques au départ sont rouges
    - Les cercles dynamiques avec collision (autre cercle ou mur) sont verts
    - Les cercles dynamiques qui sont protégés sont blanc.
    - Les cercles dynamiques qui peuvent devenir protégés sont bleu pâle.

    - Les cercles dynamiques qui sont en cooldown après avoir été protégés sont jaune.
    - Les cercles dynamiques qui vont atteindre leur taille d'explosion à la prochaine collision avec un plus petit cercle sont orange
    - Les quatres cercles créés par un cercle qui explose sont rose à leur création
    - Les cercles dynamiques sans collision sont bleus foncé
    - Le changement de couleur se fait dès qu'il y a un changement d'état. (La méthode UpdateShapeColor de l'ECSController vous permet de changer la couleur d'un élément de jeu. Le système va automatiquement effectuer une transition entre les deux couleurs. Donc même si votre couleur ne s'affiche que pendant une frame (Comme dans le cas d'une collision), elle sera quand même visible)
- En appuyant sur la touche espace, l'état de la simulation revient en arrière de 3 secondes (précision au frame près) avec un cooldown de 3 secondes. L'état du cooldown doit apparaître dans la console de l'éditeur si la barre d'espace est appuyée avant la fin du cooldown (Debug.Log).
- Le temps passe 4x plus vite dans la moitié gauche de l'écran. Il devrait donc y avoir plusieurs frames de simulation calculée à ce moment-là. (On parle ici d'effectuer 4x plus d'itérations de simulation, et non d'augmenter la vitesse des cercles)
- L'utilisateur peut cliquer sur un cercle dynamique pour le faire exploser en quatre si sa taille est égale à 4 ou plus ou le faire disparaître si sa taille est inférieure à 4.

Vous devrez ajouter des systèmes et composants supplémentaires afin de faire fonctionner la solution.
Vos composants et systèmes devraient tous être dans les dossiers "Assets/Components" et "Assets/Systems".
La maintenabilité de votre architecture et le respect du modèle ECS seront évalués en plus de la fonctionnalité. La performance ne sera pas testée (le jeu doit quand même fonctionner à un minimum de 30FPS). Utilisez donc les recommandations et les règles vues en cours afin d'avoir une architecture maintenable d'ECS.
Aucun attribut business n'est permis dans les systèmes et aucune méthode business n'est permise dans les composants.

API fourni

L'**ECSController** vous permettra d'interfacer avec Unity. Il contient des méthodes permettant de créer et détruire des objets de type cercle dans la scène, ainsi que mettre à jour leurs positions, leurs tailles et leur couleur.
Afin de ne pas avoir à implémenter vous-même le tick d'un jeu, vous devrez modifier la méthode **GetListOfSystems** dans la classe **RegisterSystems** avec votre liste de systèmes. Cette liste sera utilisée pour enregistrer les systèmes dans le gestionnaire de systèmes fourni qui appellera la méthode UpdateSystem à intervalle régulier.
Vous n'avez pas à modifier les classes déjà fournies dans "Utility/", elles sont là afin d'aider à la sérialisation de la config.

Configuration

Un fichier de configuration vous est fourni dans *Assets/Config/Config.asset*. C'est ce fichier de configuration qui sera utilisé pour interagir avec votre simulation lors de la correction.

La simulation devra pouvoir être configurée des façons suivantes:

- Les paramètres de la simulation
    - Taille d'explosion
    - Taille de protection
    - Nombre de collisions pour la protection
    - Durée de la protection
    - Cooldown de protection
- Les entités à instancier:
    - Position initiale
    - Taille
    - Vitesse initiale
- Les systèmes à exécuter
    - La liste devra être mise à jour par votre équipe selon les systèmes que vous allez implémenter. Une case à cocher pourra être utilisée pour activer ou désactiver le système pendant l'exécution de la simulation.

La configuration utilisée sera la même pour tous les étudiants, mais sera définie lors de la correction. Vous devez donc vous assurer que votre simulation est assez fiable pour gérer différents cas d'utilisation.
La classe *Assets/Config/Config.cs* est utilisée pour définir le fichier asset et ne devrait pas être modifiée.

Partie 2 - Rapport

Format PDF, interligne 1.5, 12 pts
Max 500 mots, utilisation de diagrammes suggérée.
Lorsque vous serez dans l'industrie, vous aurez souvent à expliquer vos designs à des chefs techniques ou des directeurs techniques. Le temps d'un chef ou d'un directeur technique est précieux, il va souvent être responsable de grosses équipes avec plusieurs employés, vous devez donc vous pratiquer à faire des documents techniques assez détaillés pour être compréhensibles, mais assez brefs pour ne pas prendre trop de temps à lire. Une partie de la note va dépendre de cette consigne.

Description de votre projet (max 500 mots)

Description et explication sommaire de votre architecture. À noter qu'une explication implique une **justification** de vos choix. Vous **DEVEZ JUSTIFIER** vos choix. Si votre rapport n'est qu'une explication technique de votre architecture vous perdrez des points.
On dit qu'une image vaut 1000 mots, heureusement pour vous, les images ne sont pas comptées dans le max de mots :) vous êtes donc libre d'ajouter des diagrammes pour rendre le rapport plus clair.

Évaluation (/10)

Implémentation	
Respect des requis	4
Clarté, maintenabilité, respect du modèle ECS	3
Rapport	
Description (contenu, pertinence et clarté)	3
Langue	-2
Général	
Mauvais format (remise et rapport)	-2
Projet ne compile pas	-5
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 et votre rapport PDF. Le nom du dossier devrait contenir les matricules des coéquipiers. Vous perdrez des points si vous ne respectez pas cette structure, le nom des fichiers et le nom des dossiers.

LOG8715_TP1_matricule1_matricule2.zip
|___ rapport.pdf
|___ ProjetUnity
|____ Assets/…
|____ ProjectSettings/…
|____ Packages/…

*Pour votre ProjetUnity, incluez seulement les dossiers précisez ci-haut. Les autres dossiers sont des fichiers générés automatiquement par Unity et ne sont donc pas nécessaire pour la remise. Exemple des dossiers et fichiers à ne pas inclure: Library, Logs, obj, Temp, UserSettings, Assembly-CSharp.csproj, Assembly-CSharp-Editor.csproj, *.sln, etc.*
***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 TP1.zip HEAD
```

Pour créer votre archive zip.

Le projet ne devrait pas être très lourd, veuillez remettre le zip sur Moodle avant 23h55 le 08 février 2026, soit une semaine après la deuxième séance de laboratoire du TP1.