



# **LOG8715**

## **TP1: Conception ECS**

**Version 2.0**

Responsable: Olivier Gendreau

Chargé de laboratoire: Louis-Philippe Lafontaine-Bédard

Auteurs: Samuel Bellomo et Louis-Philippe Lafontaine-Bédard

# Introduction

L'objectif de ce laboratoire est d'implémenter les bases du modèle ECS et de comprendre comment appliquer ces principes dans un contexte de simulation de jeu physique. Vous aurez donc quelques fonctionnalités à implémenter pour tester les interactions entre vos systèmes.

Le travail se fera en équipe de deux et sera divisé en deux parties: une partie sur l'implémentation d'ECS et une partie rapport.

Pour la partie implémentation, l'engin Unity sera utilisé pour ses fonctions de base, telles que l'affichage de graphiques, la compilation, la configuration, la lecture d'assets, etc. 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.

Vous ne devriez pas avoir besoin de toucher à l'éditeur à part pour tester votre jeu.

L'implémentation doit être faite totalement en C#.

À part les bibliothèques utilitaires d'Unity (comme Mathf, Time, Input, Screen, etc.) vous ne pourrez pas utiliser les classes d'entités, de GameObject et de physique d'Unity. La liste de classes interdites inclut MonoBehaviour, Component, Physics, Transform, etc. En cas de doute, envoyer un courriel au chargé de laboratoire ou demander lors des séances.

Ce laboratoire vise à vous faire explorer l'architecture ECS pour sa maintenabilité. Vous n'aurez donc pas à l'optimiser pour exécuter des milliers d'entités, mais le projet doit quand même exécuter avec un minimum de 30FPS.

# Travail à accomplir

Les travaux seront faits en équipe de deux.

## Partie 1 - 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`

### Fonctionnalités

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

- Il y aura deux types de cercles:
  - Des cercles dynamiques qui ont une vitesse et se déplacent à l'écran.
  - Des cercles statiques qui ne bougent pas.
    - Un système devra rendre le  $\frac{1}{4}$  des cercles statiques avant le début de la simulation
- À chaque détection de collision entre deux cercles, la taille des entités dynamiques devra être réduite de moitié. Les cercles statiques ne changeront pas.
- Lors d'un contact, la vitesse des entités sera inversée (pas besoin d'avoir une vraie restitution physique). Nous supposons une restitution parfaite, vous n'avez donc pas besoin de prendre en compte l'amortissement.
- Une fois en dessous de la taille minimum indiquée dans la config, ces entités n'auront plus de collisions et se passeront à travers l'une l'autre comme des fantômes.

- Lorsque l'entité entre en collision avec un bord de l'écran, en plus de rebondir son état devra se remettre à celui d'origine. Ceci inclut sa taille et sa capacité à entrer en collision avec les autres cercles.
- Les entités devront avoir une couleur qui change selon leur type.
  - Les entités statiques devront être rouges.
  - Les entités dynamiques avec collisions devront être bleues.
  - Les entités dynamiques sans collisions devront être vertes.
- En appuyant sur la touche espace, l'état de la simulation revient en arrière de 2 secondes (précision au frame près) avec un cooldown de 2 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 5x plus vite dans la moitié supérieure de l'écran. Il devrait donc y avoir plusieurs *frames* de simulation calculée à ce moment-là. (On parle ici d'effectuer 5x plus d'itérations de simulation, et non d'augmenter la vitesse des cercles)

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 exécuter à 30+ fps). 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'**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.

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 manager 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*. Vous devez y ajouter vos systèmes. 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 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 en charge 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.

On dit qu'une image vaut 1000 mots, heureusement pour vous, je ne compte pas les images dans le max de mots :) vous êtes donc libre d'ajouter des diagrammes pour rendre le rapport plus clair.

## Évaluation (/15000)

Implémentation	
Respect des requis	5000
Clarté, maintenabilité, respect du modèle ECS	5000
Rapport	
Description (contenu, pertinence et clarté)	5000
Langue	-4000
Général	
Mauvais format (remise et rapport)	-4000
Retards	Perte de points exponentielle. Jour 1: -2000 Jour 2: -4000 Jour 3: -8000 <b>0 après 3 jours de retard</b>

# Remise

Votre dossier de remise devrait contenir votre projet Unity et votre rapport PDF. Le nom du dossier devrait contenir les matricules des coéquipiers.

**LOG8715\_TP1\_matricule1\_matricule2.zip**

|\_\_ rapport.pdf

|\_\_ ProjetUnity

    |\_\_ Assets/...

    |\_\_ ProjectSettings/...

    |\_\_ ....

*Ne pas inclure le dossier "Library" et les autres 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 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 10 février, soit une semaine après la deuxième séance de laboratoire du TP1.