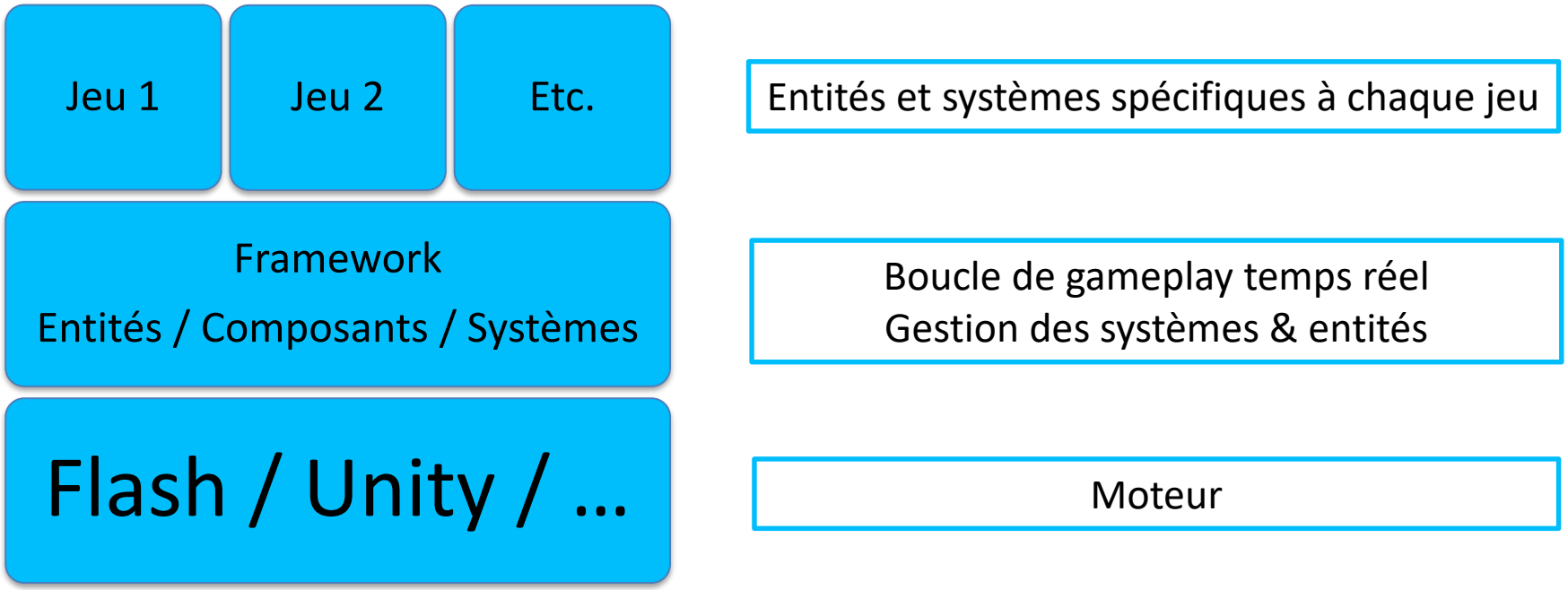
# Introduction à FYFY

FamilY For unitY

mathieu.muratet@lip6.fr

Gamagora 02/10/2020

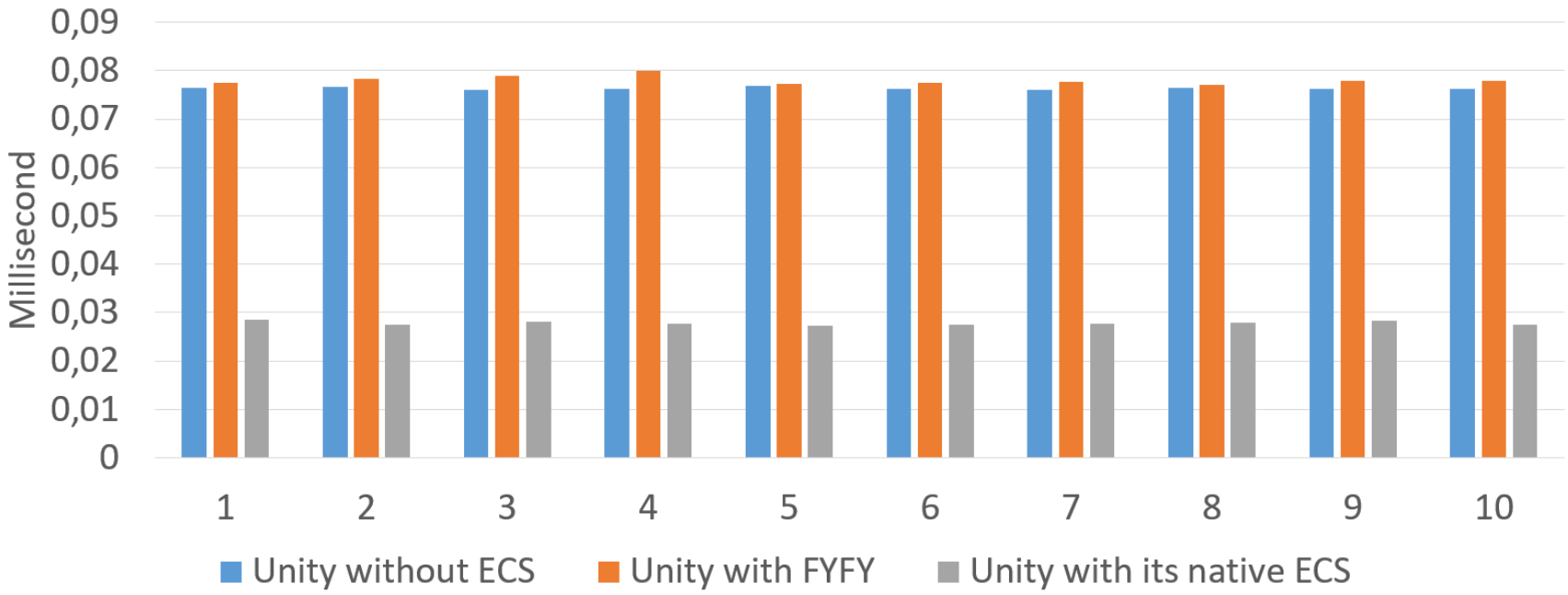
# Retour sur l'architecture générale d'ECS



# ECS et Unity

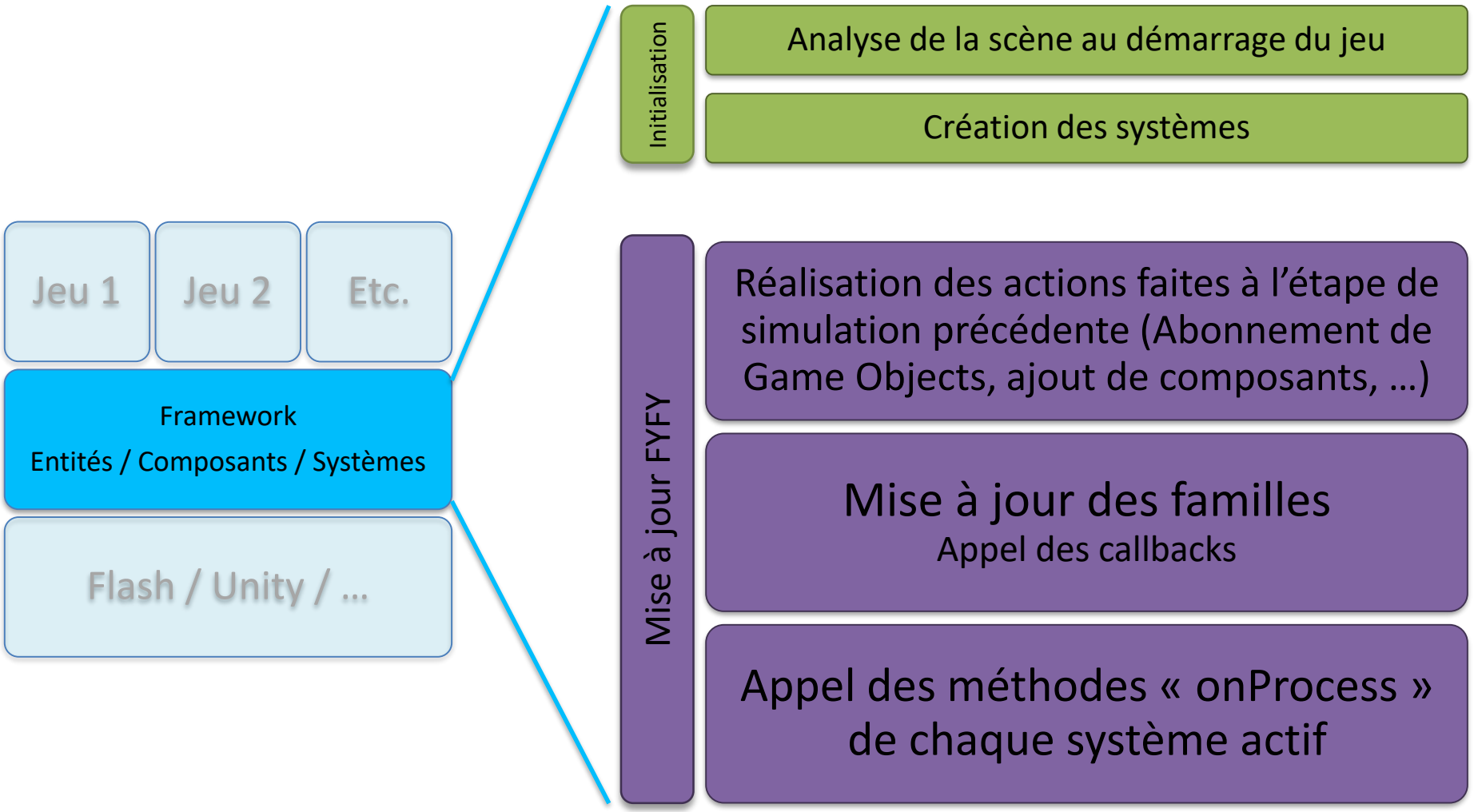
- Unity intègre un ECS (Job System, Burst Compiler…)
  - Optimisation des performances :
    - Approche classique : 18 500 GameObject à 28 Fps
    - Approche ECS : 156 000 Entities à 30 Fps
  - https://unity3d.com/fr/learn/tutorials/topics/scripting/introduction-ecs?playlist=17117
  - Difficulté : Avancée
  - Expérimental
- Pourquoi FYFY ?
  - Rendre plus accessible l'ECS
  - Intégration à l'environnement Unity plus « classique »
  - Contrepartie : pas plus performant que l'approche classique
  - Fournit un cadre méthodologique de conception

# Benchmark

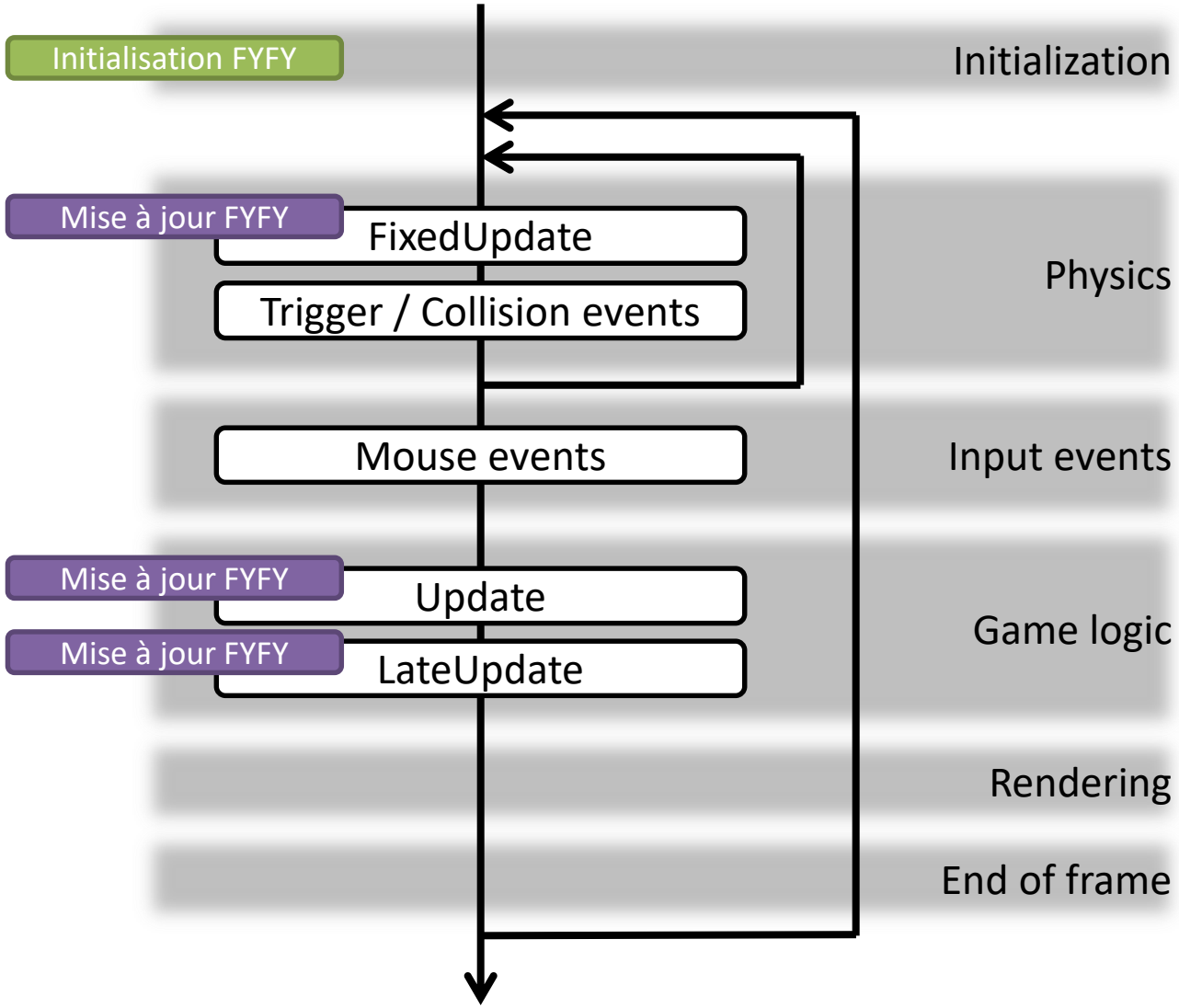



- Temps moyen pour calculer une frame (10 000 cubes en rotation)

# Boucle de simulation de FYFY

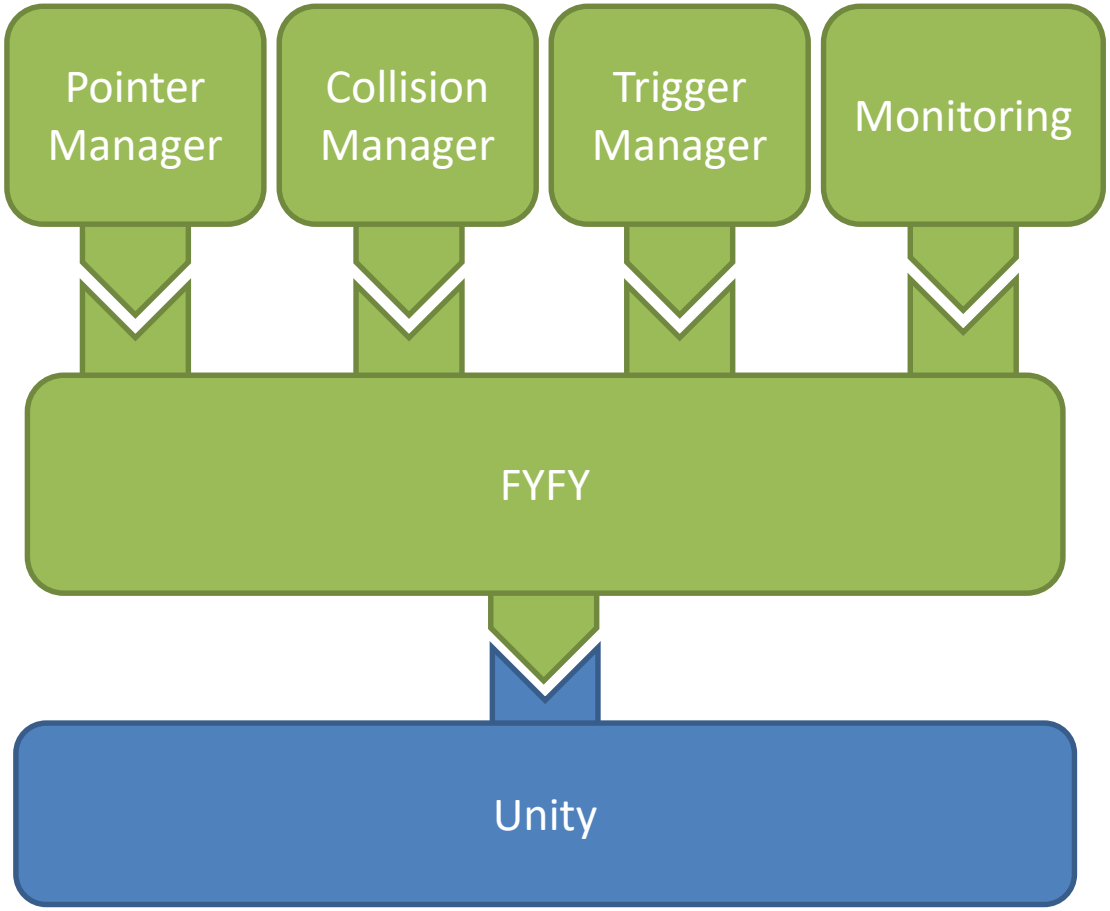


# Unity flowchart et intégration FYFY




https://docs.unity3d.com/uploads/Main/monobehaviour_flowchart.svg

# Architecture de FYFY



# FYFY API

- Création d'un composant

```
using UnityEngine;

public class Move : MonoBehavior {
    // Advice: FYFY component aims to contain only public
    // members (according to Entity-Component-System paradigm).
    public float speed = 2.5f;
}
```

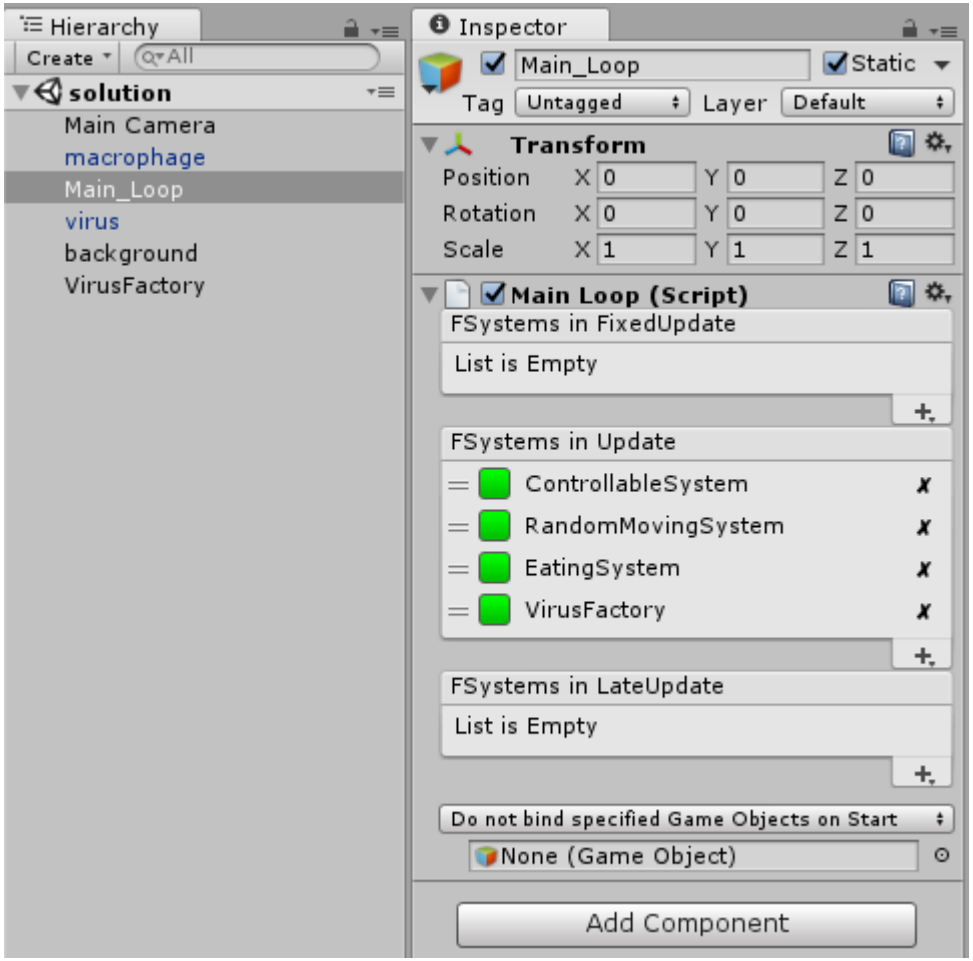
# FYFY API

- Création d'un système

```
using UnityEngine;
using FYFY;

public class ExempleSystem : FSystem {
    // Use this to update member variables when system pause.
    // Advice: avoid to update your families inside this function.
    protected override void onPause(int currentFrame) {
    }

    // Use this to update member variables when system resume.
    // Advice: avoid to update your families inside this function.
    protected override void onResume(int currentFrame){
    }

    // Use to process your families.
    protected override void onProcess(int familiesUpdateCount) {
    }
}
```

# FYFY API

- Main_Loop



# FYFY API

- GameObjectManager
  - Abonner un GameObject

```
GameObjectManager.bind (go);
```

  - Ajouter dynamiquement un Composant

```
GameObjectManager.addComponent<SphereCollider> (go, new { radius = 2f });
```

  - Supprimer dynamiquement un Composant

```
GameObjectManager.removeComponent<SphereCollider> (go);
```

  - Désabonner un GameObject

```
GameObjectManager.unbind (go);
```

# FYFY API

- GameObjectManager
  - (Dés)Activer un GameObject

```
GameObjectManager.setGameObjectState (go, true);
```

  - Changer le parent d'un GameObject

```
GameObjectManager.setGameObjectParent (go, parent_go, true);
```

  - Changer le layer d'un GameObject

```
GameObjectManager.setGameObjectLayer (go, 5);
```

  - Changer le tag d'un GameObject

```
GameObjectManager.setGameObjectTag (go, "newTag");
```

# FYFY API

- GameObjectManager
  - Changer de scène

```
GameObjectManager.loadScene ("nomDeLaSceneACharger");
```

  - Cas du DontDestroyOnLoad

```
GameObjectManager.dontDestroyOnLoadAndRebind (go);
```


Toujours utiliser le GameObjectManager pour les GameObjects synchronisés dans les familles

# FYFY API

- FamilyManager et Matchers (AllOf, AnyOf, NoneOf)

```
Family myFamily = FamilyManager.getFamily (
            new AllOfComponents (typeof (Move), typeof (RandomTarget)),
            new NoneOfComponents (typeof (Velocity)));
```

- Matchers possibles
  - Sur la présence ou non de composants
  - Sur l'appartenance ou pas à un layer
  - Sur l'association ou pas à un tag
  - Sur l'intégration ou pas de propriétés (ACTIVE_SELF, ACTIVE_IN_HIERARCHY, HAS_PARENT, HAS_CHILD)

# FYFY API

- Parcours d'une famille

```
for (GameObject go in myFamily){
    // Récupération des composants
    Move mv = go. GetComponent<Move> ();
    RandomTarget rt = go. GetComponent<RandomTarget> ();
    Velocity rt = go. GetComponent<Velocity> ();
    // Exploitation des composants
    ...
}
```

- Accès au premier élément d'une famille

```
GameObject go = myFamily.First();
```

# FYFY API

- Ajout d'écouteurs sur les familles

```
// Définition d'une famille
Family myFamily = FamilyManager.getFamily (...);
// Enregistrement d'un écouteur sur l'ajout d'un GameObject à la famille
myFamily.addEntryCallback (onAdd);
// Enregistrement d'un écouteur sur le retrait d'un GameObject de la famille
myFamily.addExitCallback (onRemove);


// Définition de l'écouteur de l'ajout
void onAdd (GameObject addingGo){
   // traitement du GameObject passé en paramètre arrivant dans la famille
   ...
}

// Définition de l'écouteur du retrait
void onRemove (int removingGOInstanceId){
   // traitement de l'identifiant d'instance du GameObject retiré de la
famille
   ...
}
```