MOCAH Team

Table of contents

Contact : mathieu.muratet@lip6.fr

MOCAH Team

FYFY - FamilY For unitY

FYFY is an Entity Component System (ECS) made for Unity. See API documentation for details.

What FYFY is?

FYFY was designed for teaching purposes to help beginners to understand data-driven simulation and to play with ECS mechanisms. FYFY is focused on the modularity interest of ECS. We try to make it easy to use and well-integrated into the Unity editor.

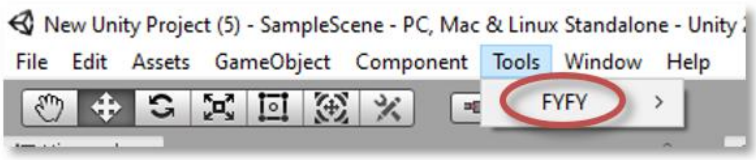
What FYFY is not?

FYFY doesn't focus on optimization. If you are interested in high performances games use the native ECS of Unity instead: https://learn.unity.com/tutorial/entity-component-system#

Short Introduction to FYFY

How to add FYFY to your project?

1. Drag and drop dll and xml files of the latest release in your Asset folder.
2. Create the MainLoop with the FYFY menu


New Unity Project (5) - SampleScene - PC, Mac & Linux Standalone - Unity
File
Edit
Assets
GameObject
Component
Tools
Window
Help
FYFY

3. Now you can begin to code your systems and play with ECS, enjoy!

What are FYFY entities?

Each Unity GameObject is processed as an entity in the FYFY context.

What are FYFY components?

A FYFY component extends MonoBehaviour and contains only public data (no functions):

```
using UnityEngine;

public class FYFYComponentExample : MonoBehaviour {
    // Advice: FYFY component aims to contain only public members
    public int firstValue;
    public GameObject secondValue;
    public string thirdValue;
}
```

What are FYFY systems?

A FYFY system extends FSystem:

```
using UnityEngine;
using FYFY;

public class FYFYSystemExemple : FSystem {

    public static FYFYSystemExemple instance;

    public FYFYSystemExemple() {
        instance = this;
    }

    // Use this to init system before the first onProcess call
    protected override void onStart(){
    }

    // Use this to update member variables when system pause.
    protected override void onPause(int currentFrame) {
    }

    // Use this to update member variables when system resume.
    protected override void onResume(int currentFrame){
    }

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

A FYFY system uses families to get entities to process on. In the following example, the system builds a family that filters all game objects that contains the `FYFYComponentExample` component and increments their `firstValue` member:

```
using UnityEngine;
using FYFY;
```

```
public class FYFYSystemExample : FSystem {

    private Family f_familyExample = FamilyManager.getFamily(new
AllOfComponents(FYFYComponentExample));

    // Use to process your families.
    protected override void onProcess(int familiesUpdateCount) {
        foreach (GameObject go in f_familyExample)
        {
            go.GetComponent<FYFYComponentExample>().firstValue++;
        }
    }
}
```

What are families?

Family is the mechanism that enables systems to access entities (GameObjects). Families are built by the FamilyManager with the help of a set of matchers:

Matchers on components

- AllOfComponents enables to filter GameObjects that contain all of specified components.

```
// Define a family that selects all GameObjects that contain Move and RandomTarget
components
Family myFamily = FamilyManager.getFamily(new AllOfComponents(typeof(Move),
typeof(RandomTarget)));
```

- AnyOfComponents enables to filter GameObjects that contain at least one of the specified components.

```
// Define a family that selects all GameObjects that contain Move or RandomTarget
components (or both)
Family myFamily = FamilyManager.getFamily(new AnyOfComponents(typeof(Move),
typeof(RandomTarget)));
```

- NoneOfComponents enables to filter GameObjects that don't contain the specified components.

```
// Define a family that selects all GameObjects that don't contain Move and
RandomTarget components
Family myFamily = FamilyManager.getFamily(new NoneOfComponents(typeof(Move),
typeof(RandomTarget)));
```

Matchers on layers

- AnyOfLayers enables to filter GameObjects that are part of the specified layers.

```
// Define a family that selects all GameObjects that are part of layer 5, 6 or 7
```

```
Family myFamily = FamilyManager.getFamily(new AnyOfLayers(5, 6, 7));
```

- NoneOfLayers enables to filter GameObjects that are not part of the specified layers.

```
// Define a family that selects all GameObjects that are not part of layer 5, 6 or
7
Family myFamily = FamilyManager.getFamily(new NoneOfLayers(5, 6, 7));
```

Matchers on tags

- AnyOfTags enables to filter GameObjects that are associated to one of the specified tags.

```
// Define a family that selects all GameObjects tagged "virus" or "bactery"
Family myFamily = FamilyManager.getFamily(new AnyOfTags("virus", "bactery"));
```

- NoneOfTags enables to filter GameObjects that are not associated to one of the specified tags.

```
// Define a family that selects all GameObjects that are not tagged "virus" and
"bactery"
Family myFamily = FamilyManager.getFamily(new NoneOfTags("virus", "bactery"));
```

Matchers on properties

Four properties are currently managed by FYFY. These properties are defined inside `PropertyMatcher` enum: ACTIVE_SELF, ACTIVE_IN_HIERARCHY, HAS_PARENT and HAS_CHILD.

- AllOfProperties enables to filter GameObjects that check all of specified properties.

```
// Define a family that selects all GameObjects that are active in hierarchy and
has a parent
Family myFamily = FamilyManager.getFamily(new
AllOfProperties(PropertyMatcher.PROPERTY.ACTIVE_IN_HIERARCHY,
                    PropertyMatcher.PROPERTY.HAS_PARENT));
```

- AnyOfProperties enables to filter GameObjects that check at least one of the specified properties.

```
// Define a family that selects all GameObjects that are active in hierarchy or
has a parent
Family myFamily = FamilyManager.getFamily(new
AnyOfProperties(PropertyMatcher.PROPERTY.ACTIVE_IN_HIERARCHY,
                    PropertyMatcher.PROPERTY.HAS_PARENT));
```

- NoneOfProperties enables to filter GameObjects that do not check all of specified properties.

```
// Define a family that selects all GameObjects that are not active in hierarchy
and has not a parent
Family myFamily = FamilyManager.getFamily(new
NoneOfProperties(PropertyMatcher.PROPERTY.ACTIVE_IN_HIERARCHY,
                      PropertyMatcher.PROPERTY.HAS_PARENT));
```

Matchers combination

It is possible to combine matcher to build the system appropriate filter. For instance, following family will contain all GameObjects active in hierarchy with components Move and RandomTarget and not Velocity, and part of layer 3 or 7

```
Family myFamily = FamilyManager.getFamily(
    new AllOfProperties(PropertyMatcher.PROPERTY.ACTIVE_IN_HIERARCHY),
    new AllOfComponents(typeof(Move), typeof(RandomTarget)),
    new NoneOfComponents(typeof(Velocity)),
    new AnyOfLayers(3, 7));
```

Family listeners

Because game simulation is dynamic some GameObjects enters and leaves families at runtime. You can define listener on families to catch these events.

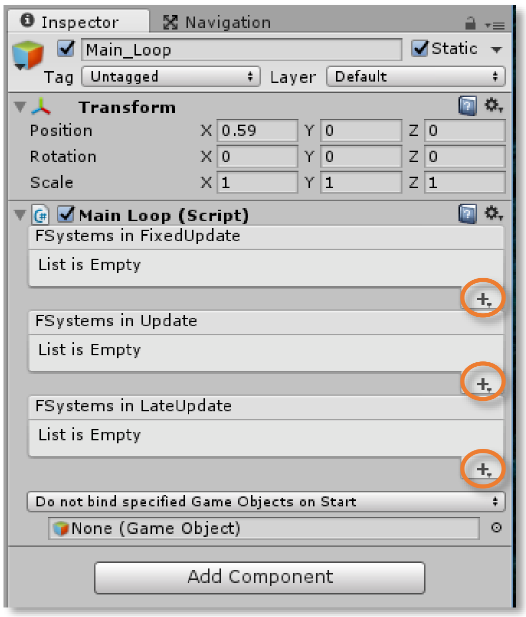
```
// Family definition
Family myFamily = FamilyManager.getFamily (...);

// Use this to init system before the first onProcess call
protected override void onStart(){
    // Add listener to call back when a GameObject enters into this family
    myFamily.addEntryCallback (onGameObjectEnter);
    // Add listener to call back when a GameObject leaves this family
    myFamily.addExitCallback (onGameObjectExit);
}

void onGameObjectEnter (GameObject addingGo){
    ...
}
void onGameObjectExit (int removingGoInstanceId){
    ...
}
```

How to integrate a system to my project?

Use MainLoop inspector to add your systems to one of the three execution contexts (FixedUpdate, Update or LateUpdate).


Inspector | Navigation

Main_Loop | Static

Tag Untagged | Layer Default

Transform

	X	Y	Z
Position	0.59	0	0
Rotation	0	0	0
Scale	1	1	1

Main Loop (Script)

FSystems in FixedUpdate
List is Empty

FSystems in Update
List is Empty

FSystems in LateUpdate
List is Empty

Do not bind specified Game Objects on Start

None (Game Object)

Add Component

GameObjectManager

The GameObjectManager enables FYFY to keep synchronized with Unity simulation. If you choose to experiment FYFY you have to replace some of the classic Unity calls with overloaded GameObjectManager calls:

```
GameObject go = Instantiate(prefab);
// Bind each new GameObject to FYFY
GameObjectManager.bind(go);

// Unbind all GameObject before destroying it
GameObjectManager.unbind(go);
Destroy(go);

// Replace all classic AddComponent calls
go.AddComponent<FYFYComponentExample>();
// with
GameObjectManager.addComponent<FYFYComponentExample>(go);

// Replace all component destruction
FYFYComponentExample fce = go.GetComponent<FYFYComponentExample>();
Destroy(fce);
// with
GameObjectManager.removeComponent<FYFYComponentExample>(go);

// Replace all changes on active state
go.SetActive(false);
// with
GameObjectManager.setGameObjectState(go, false);

// Replace all hierarchical updates
go.transform.SetParent(newParent, true);
// with
GameObjectManager.setGameObjectParent(go, newParent, true);
```

```
// Replace all scene loadind
UnityEngine.SceneManagement.SceneManager.LoadScene(sceneName);
// with
GameObjectManager.loadScene(sceneName);

// Replace all DontDestroyOnLoad calls
DontDestroyOnLoad(go);
// with
GameObjectManager.dontDestroyOnLoadAndRebind(go);
```

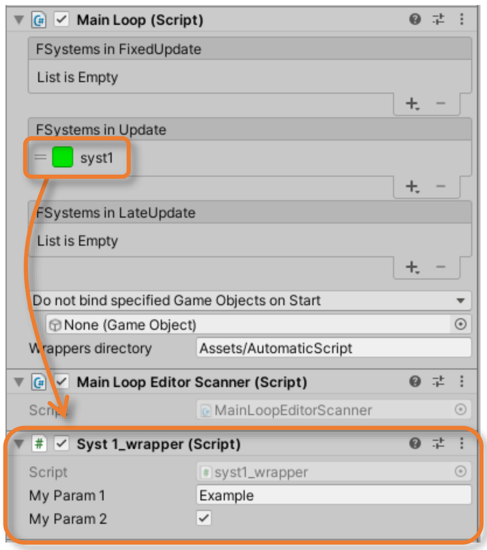
Access systems' fields and methods in Inspector

When you add a system to the MainLoop, FYFY scans your system and adds automatically a Monobehavior wrapper to your MainLoop. This wrapper exposes serializable fields of your system that enables you to initialize these fields via the Inspector.

Example of fields exposition:

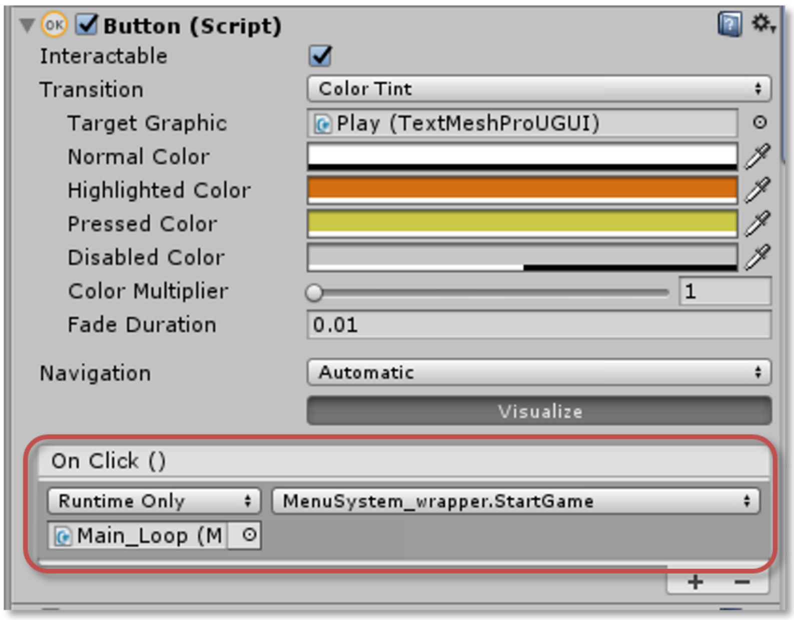
```
using FYFY;

public class Syst1 : FSystem {
    public string myParam1;
    public bool myParam2;
    ...
}
```



Main Loop (Script)
FSystems in FixedUpdate
List is Empty
FSystems in Update
syst1
FSystems in LateUpdate
List is Empty
Do not bind specified Game Objects on Start
None (Game Object)
Wrappers directory: Assets/AutomaticScript
Main Loop Editor Scanner (Script)
Script: MainLoopEditorScanner
Syst 1_wrapper (Script)
Script: syst1_wrapper
My Param 1: Example
My Param 2: ✓

If you define inside a system a void public function with a maximum of one parameter, FYFY give you access to this function through its wrapper of the MainLoop to link it on UI events.



Button (Script)

Interactable ✓

Transition: Color Tint

Target Graphic: Play (TextMeshProUGUI)

Normal Color

Highlighted Color

Pressed Color

Disabled Color

Color Multiplier: 1

Fade Duration: 0.01

Navigation: Automatic

On Click ()

Runtime Only — MenuSystem_wrapper.StartGame

Main_Loop (M