

Table of contents

[FYFY - FamilY For unitY 2](#_Toc94521761)

[What FYFY is? 2](#_Toc94521762)

[What FYFY is not? 2](#_Toc94521763)

[Short Introduction to FYFY 2](#_Toc94521764)

[How to add FYFY to your project? 2](#_Toc94521765)

[What are FYFY entities? 2](#_Toc94521766)

[What are FYFY components? 3](#_Toc94521767)

[What are FYFY systems? 3](#_Toc94521768)

[What are families? 4](#_Toc94521769)

[Matchers on components 4](#_Toc94521770)

[Matchers on layers 4](#_Toc94521771)

[Matchers on tags 5](#_Toc94521772)

[Matchers on properties 5](#_Toc94521773)

[Matchers combination 6](#_Toc94521774)

[Family listeners 6](#_Toc94521775)

[How to integrate a system to my project? 6](#_Toc94521776)

[GameObjectManager 7](#_Toc94521777)

[Access systems’ fields and methods in Inspector 8](#_Toc94521778)

**Contact :** mathieu.muratet@lip6.fr

# FYFY - FamilY For unitY

FYFY is an Entity Component System (ECS) made for Unity. ECS software architecture is interesting for two main reasons: optimization and modularity.

## 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.

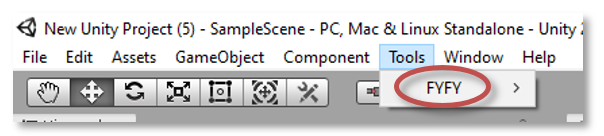
## What FYFY is not?

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#](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

[](https://github.com/Mocahteam/FYFY/blob/master/docs/CreateMainLoop.PNG)

1. 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 familyExample = FamilyManager.getFamily(new AllOfComponents(FYFYComponentExample));

// Use to process your families.

protected override void onProcess(int familiesUpdateCount) {

foreach (GameObject go in 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){

...

}

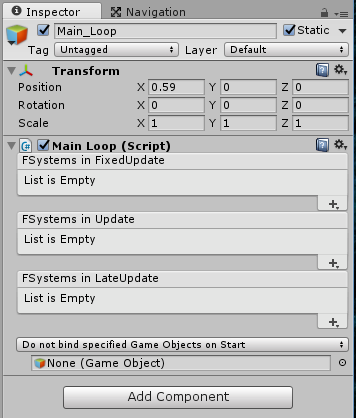
void onGameObjectExit (int removingGoInstanceId){

...

}

## 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).



## 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;

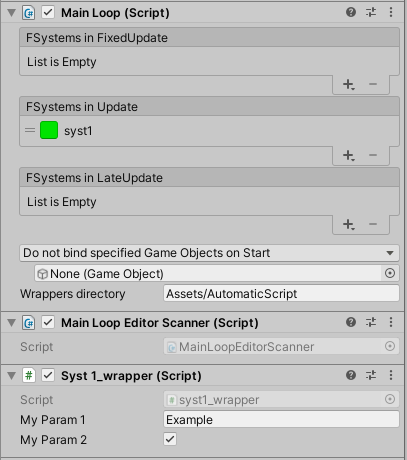
public class Syst1 : FSystem {

public string myParam1;

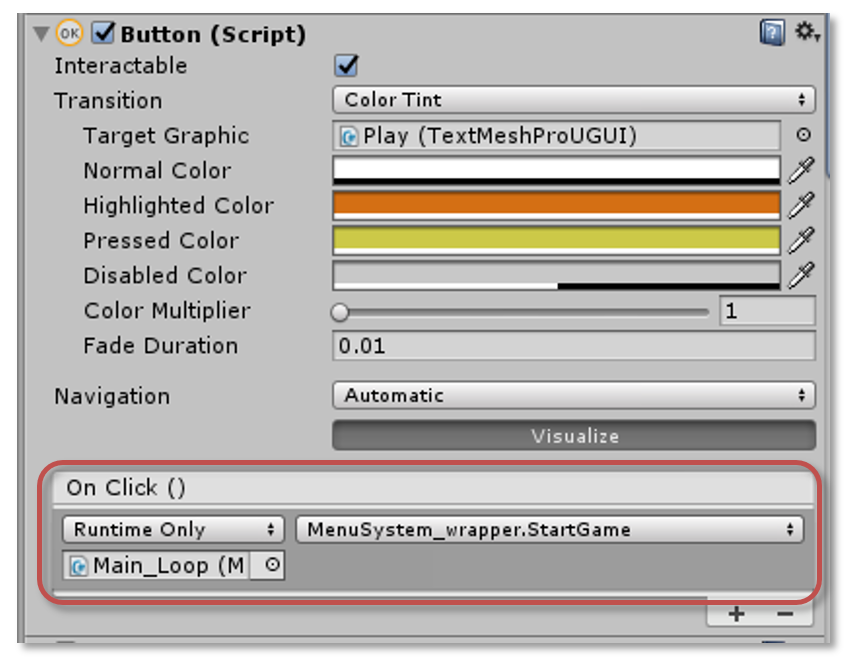
public bool myParam2;

...

}



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.

[](https://github.com/Mocahteam/FYFY/blob/master/docs/UI_Event.PNG)