TUTORIAL SYSTEM

CREATE TUTORIALS EASILY

NINESOFT

CONTENTS

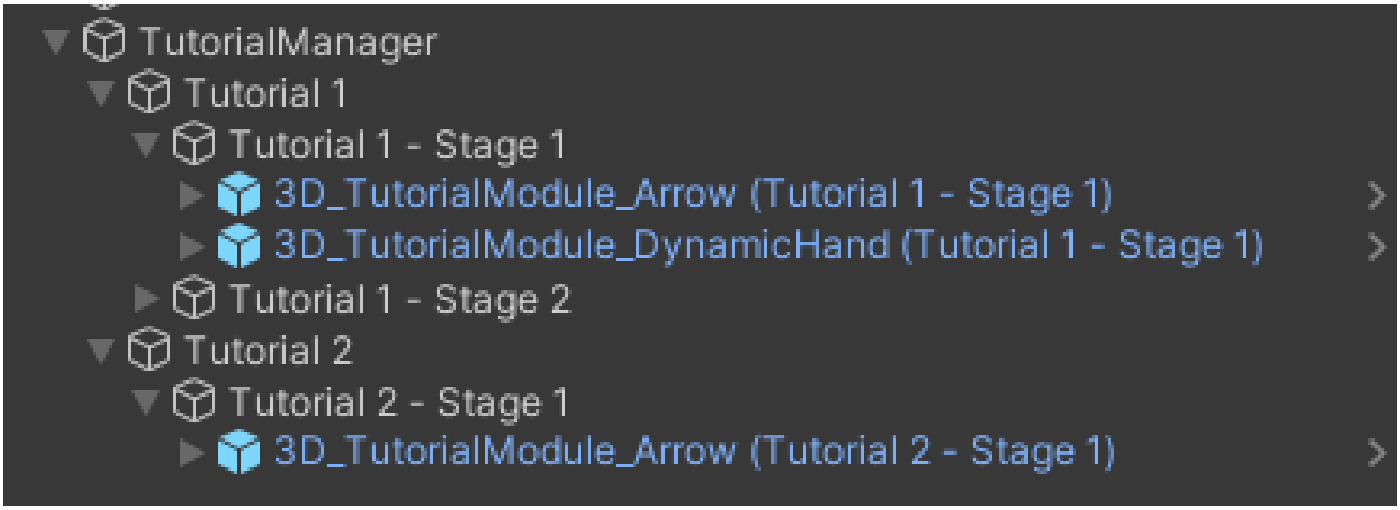
GET STARTED

>> Using this asset you will be able to prepare tutorials for your games in an easy way. You can prepare simple tutorials for your hyper casual, casual, FPS, TPS, puzzle or other types of games.

This asset; consists of Manager, Tutorials, Stages and Modules.

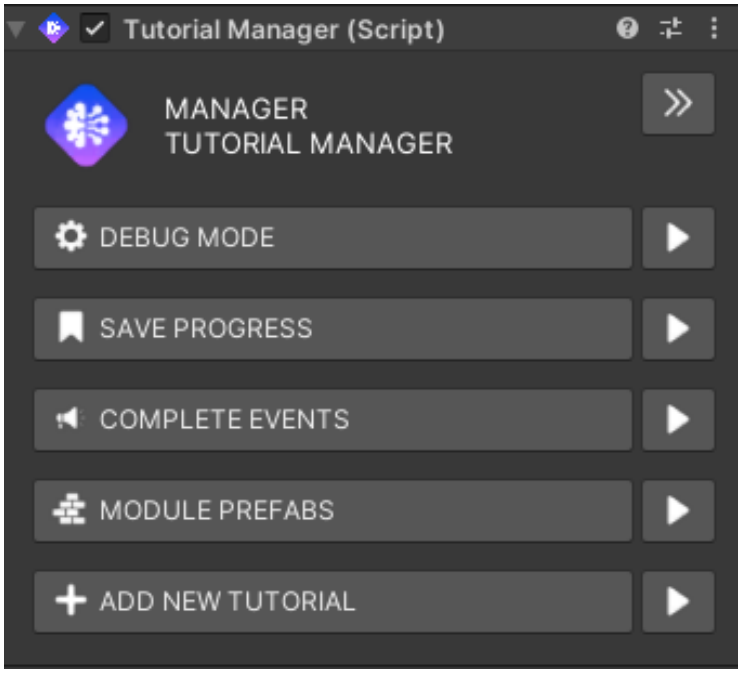
>> A TutorialManager contains Tutorials as child-objects and a Tutorial contains Stages as child-objects. A Stage contains Modules as child-objects (or as UI-objects)

Example hierarchy:



TUTORIAL MANAGER

>> Tutorial Manager is the main component that manages the whole process.

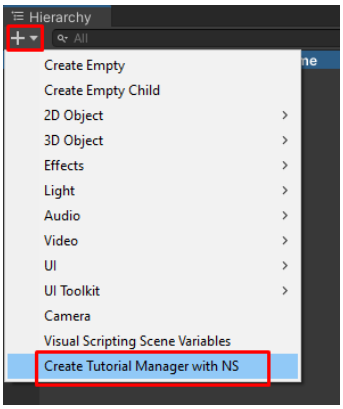


You can use Tutorial Manager multiple times within the same project.

Be sure to use the Tutorial Manager only once in the same level or the same scenes.

For example, make sure you only have 1 TutorialManager in your scene.

ADD A TUTORIAL MANAGER



>> To add a TutorialManager to your scene, right-click hierarchy and select the "Create Tutorial Manager" option.

DEBUG MODE

Debug Mode: If you turn this option off, it will turn off all Tutorial System related Debug.Log events in the unity console.

SAVE PROGRESS

Do not show again once the tutorials are complete: If you select this option, the training will not restart again after completion.

(example: the player has completed the tutorial, he will not encounter the tutorial again when he opens the game again)

Save Tutorial Progress: If you select this option, the players last remaining stage is saved. (example: player closed the game in tutorial 1 - stage 3, when he reopens the game it will continue from tutorial 1 - stage 3)

Save Key: It is the key used to record the progress of the tutorial. It is created automatically and there is no need to change it. (saves are stored in the player-prefs system, a save key is required so that saves do not overlap when using multiple Tutorial Managers)

COMPLETE EVENTS

On All Tutorials Completed: Once all the tutorials in TutorialManager are complete, you can add the events you want to happen here.

Use this code to find out if all tutorials have been completed:

```
if(TutorialManager.Instance.AllTutorialsCompleted)
{
        // YOUR CODE
}
```

MODULE PREFABS

>> The prefabs of the Tutorial Modules are kept here. When you want to add a Tutorial Module to a stage, the data here is referenced. There is no need to make any changes in this section.

ADD NEW TUTORIAL

Add New Tutorial: Click on the "Add New Tutorial" button to create a new tutorial

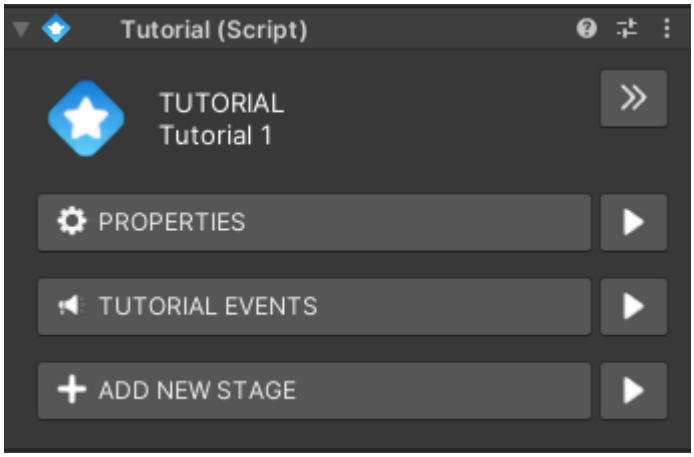
SKIP TUTORIAL (v1.2.0)

You can use the following codes to skip a tutorial

```
TutorialManager.Instance.SkipAllTutorials(); // Skips all tutorials.
TutorialManager.Instance.SkipCurrentTutorial(); // Skips the current tutorial.
```

TUTORIAL

>> There are many stages in the tutorial and it controls all these stages.



PROPERTIES

Tutorial Name:

Tutorial Name :)

TUTORIAL EVENTS

On This Tutorial Started: Events that will happen when the tutorial starts

On This Tutorial Completed: The events that will take place when all the stages are completed and the Tutorial ends:

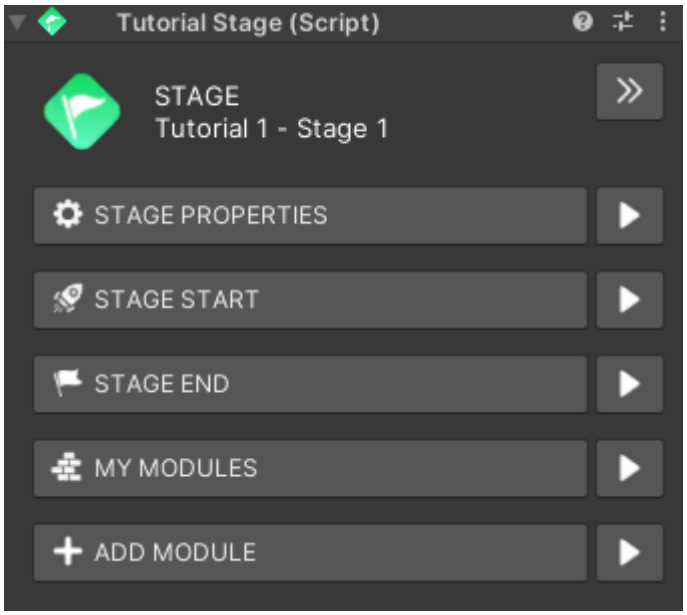
ADD NEW STAGE

Add New Stage: Click on the "Add New Stage" button to create a new stage

STAGE

>> Stages are the smallest building block of the tutorial.

It is recommended that you create a new stage for each step.



STAGE PROPERTIES

Tutorial Stage Name:

Tutorial Stage Name :)

STAGE START

>> Conditions and events required for the start of the stage

Start Delay: Start delay time for this stage

Start Trigger: The trigger* required for the start of the stage

On Stage Start: Events that will happen when the stage starts

STAGE END

>> Conditions and events necessary for the end of the stage

End Delay: End delay time for this stage

End Trigger: The trigger* required for the end of the stage

On Stage End: Events that will happen when the stage ends

START & END TRIGGERS

- AUTOMATIC RUN: The stage starts or ends automatically when it is its turn.

- COLLISION: The stage starts or ends when an object with the specified tag hits your collider

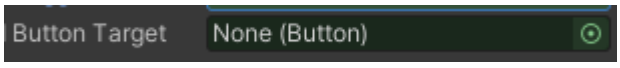
Collider Type: Choose Collider 3D or 2D according to your collider

Your Collider: Your collider (example: player collider). Drag the your collider and drop it into this field

Collision Tag: Target colliders tag (example: “Enemy”)

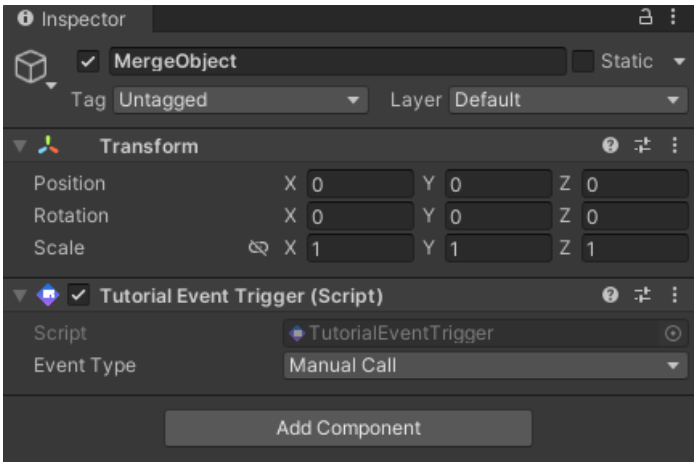
Is Trigger: Select this option for ‘On Trigger’ events. uncheck this option for ‘On Collision’ events.

- BUTTON CLICK: A button must be clicked for the stage to start or end. Drag the target button and drop it into this field



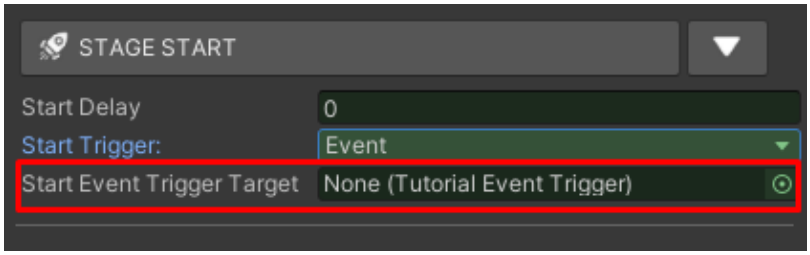
- EVENT: Allows the stage to start or end when there is any event

>> Event Example 1: You have a merge game and you want the stage to start or end when 2 objects are merged



Add a TutorialEventTrigger script to your game object(or create empty game object and add Tutorial Event Trigger script)

Drag the game object containing the Tutorial Event Trigger script and drop it into this field



Define a Tutorial EventTrigger variable in your own code and assign the game object containing the TutorialEventTrigger script to the variable that you created in your own code (drag and drop in inspector)

```
using NINESOFT.TUTORIAL_SYSTEM; // ← DON'T FORGET TO ADD THİS
...
public TutorialEventTrigger myEventTrigger;
public void Merged()  // YOUR GAME FUNCTION
{
        myEventTrigger.CallMyEvent();
        //my other merge codes..
}
```

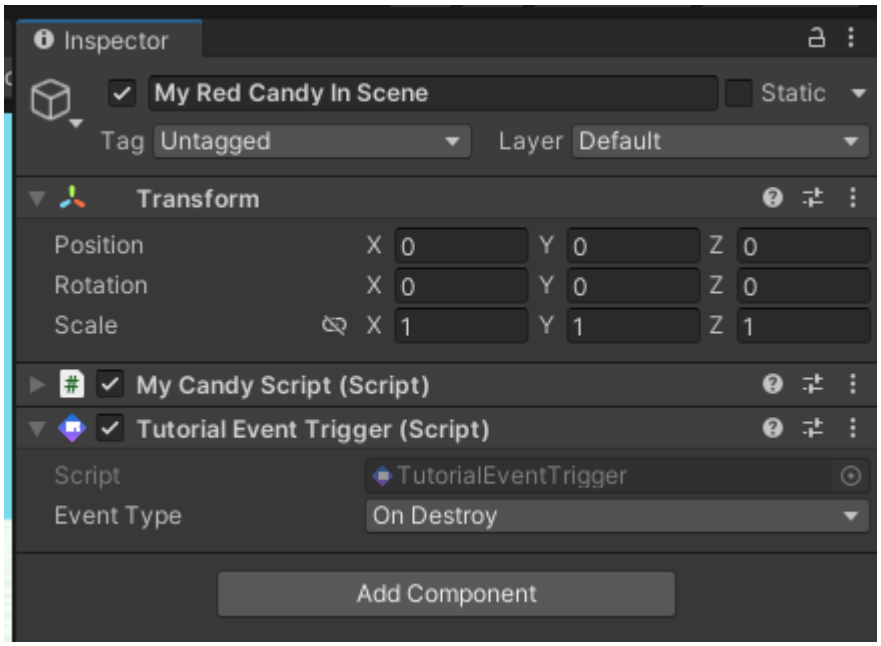
Thus, when the merge function runs, TutorialEventTrigger will be triggered.

>> Event Example 2: You have a match 3 game. When you match 3 identical candies in this game, the candies are broken.

You want the scene to end when the red candy breaks.

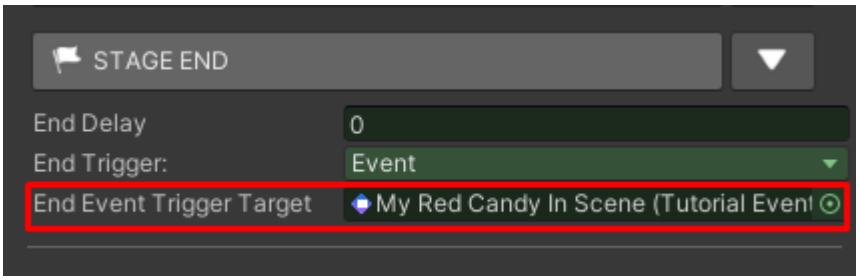
Suppose the candy is destroyed after the candy in the game is broken..

Add the Tutorial Event Trigger script to your red candy game object.



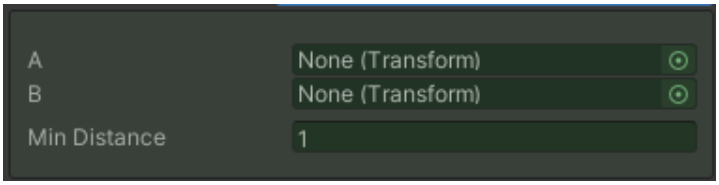
Select the onDestroy option as the Event Type.

Drag your red candy game object and drop it into the Trigger field inside the stage.



It's ready now! when your game object named "My Red Candy In Scene" is destroyed, the event will be triggered and the stage will end

- DISTANCE: If 2 objects get close enough to each other, the stage starts or ends.



A: The first object

B: The second object

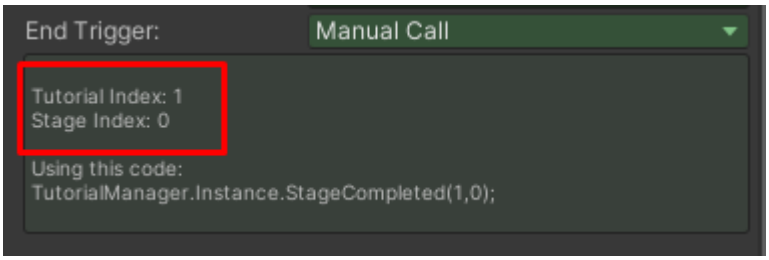
Min Distance: Approach distance

- MANUAL CALL: Select this option to start or end the stage with the code.

Sample Code: TutorialManager.Instance.StageStarted(tuorialIndex , stageIndex);

You need to send the tutorial index and the stage index as parameters in the code.

The tutorial index and the stage index are available in the information box at the bottom.



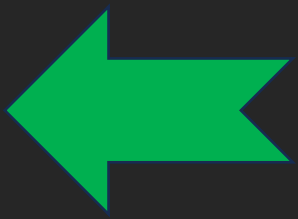

```
using NINESOFT.TUTORIAL_SYSTEM; // ← DON'T FORGET TO ADD THİS
...
public void MyFunc()  // YOUR GAME FUNCTION
{
        // if you want to start the stage, use this code:
        TutorialManager.Instance.StageStarted(tuorialIndex:0,stageIndex:0);

        // if you want to end the stage, use this code:
        TutorialManager.Instance.StageCompleted(tuorialIndex:0,stageIndex:0);
}
```

if you are going to manually call similar stages in the same function, use the “if” and “else if” methods

```
public void Merged()  // YOUR GAME FUNCTION
{       // Avoid writing one after the other!!!
        TutorialManager.Instance.StageCompleted(1, 0);
        TutorialManager.Instance.StageCompleted(1, 1);

        if(TutorialManager.Instance.StageCompleted(1, 0))
        {
        //if tutorial 1 is stage 0, the stage is completed,
        }else if(TutorialManager.Instance.StageCompleted(1, 1))
        {
        //or if tutorial 1 is stage 1, the training is completed
        }
}
```

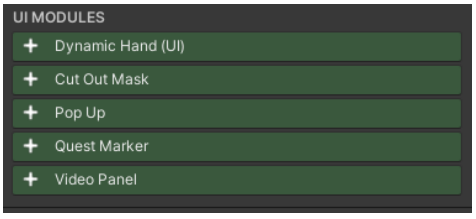


MY MODULES

>> It is a list of modules that depend on the stage. if you have manually added a module to the stage (by drag and drop method to the scene), you need to add that module to this list. If you have added a module from the "Add Module" section, the added module is automatically added to this list.

ADD MODULE

>> Click on the button of the module you want to add to the stage. the module will be created automatically.

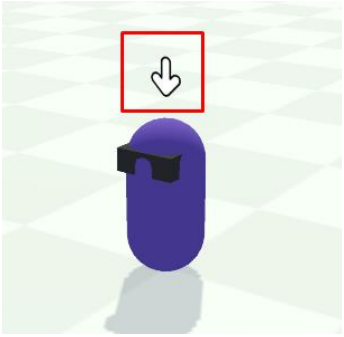


MODULES

>> To add a module to a stage, you can use the "Add module" section in the stage component. More than one module can be added to a stage

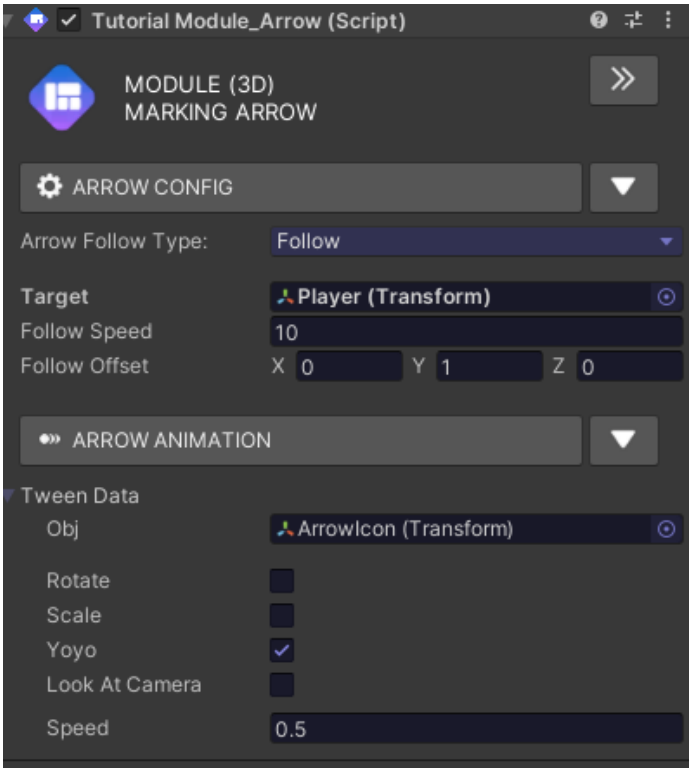
ARROW MODULE (3D)

>> Puts an animated arrow mark on a character or an object.



Arrow Follow Type: If you select the "Follow" option, the arrow will follow the target in real time.

If you select the "Static" option, the arrow remains fixed where you put it



Arrow Animation:

You can change the arrow animation from the "Arrow Animation" section

Rotate: The object constantly rotates around itself.

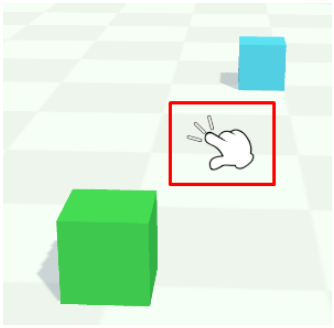
Scale: The object constantly grows and shrinks.

Yoyo: The object is constantly moving up and down.

Look At Camera: The object is constantly looking at the camera

DYNAMIC HAND MODULE (3D - UI)

>> In this module, a hand icon moves between the points



Points: The points where the hand icon will move

Point: Target point

Offset: (v1.1.0)

Offset for hand position

Hand Event Type: The event that the hand icon will perform when it comes to the target point

Speed: The movement speed of the han d icon

Waiting Time: The waiting time of the hand icon on any point

Wait Time For Replay: When one loop ends, the time it takes for the other loop to start.

TEXT 3D (3D)

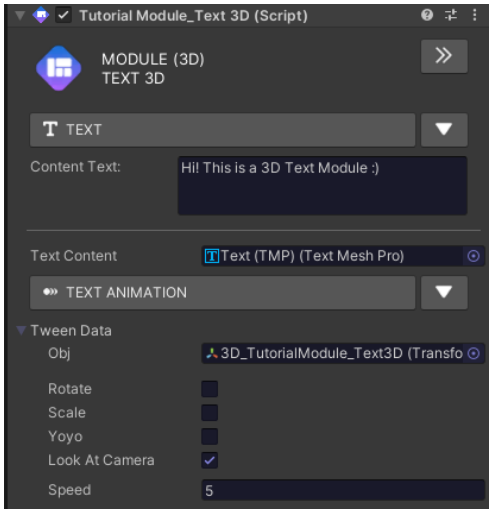
>> If you want to add text to the stage, you can use this module. (TextMeshPro is required)

Content Text: Your content text.

Text Animation:

You can change the text animation from the "Text Animation" section

Text, by default, constantly looks at the camera.



CUT OUT MASK (UI)

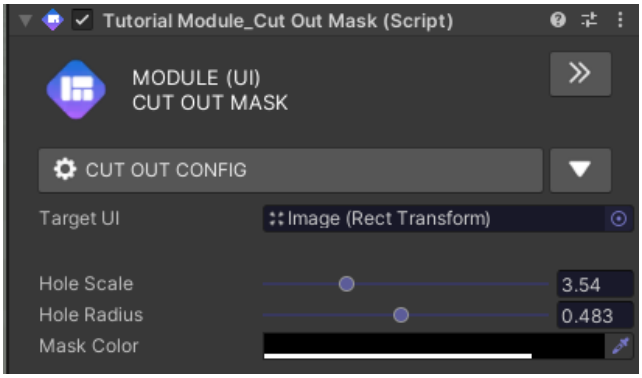
>> It is a module that allows you to focus on an element contained in the UI

Target UI: The object you want to focus on

Hole Scale: Hole width

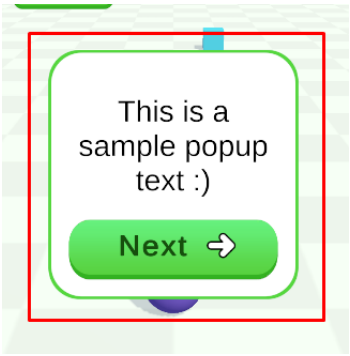
Hole Radius: Corner softness

Mask Color: Mask color



POPUP MODULE (UI)

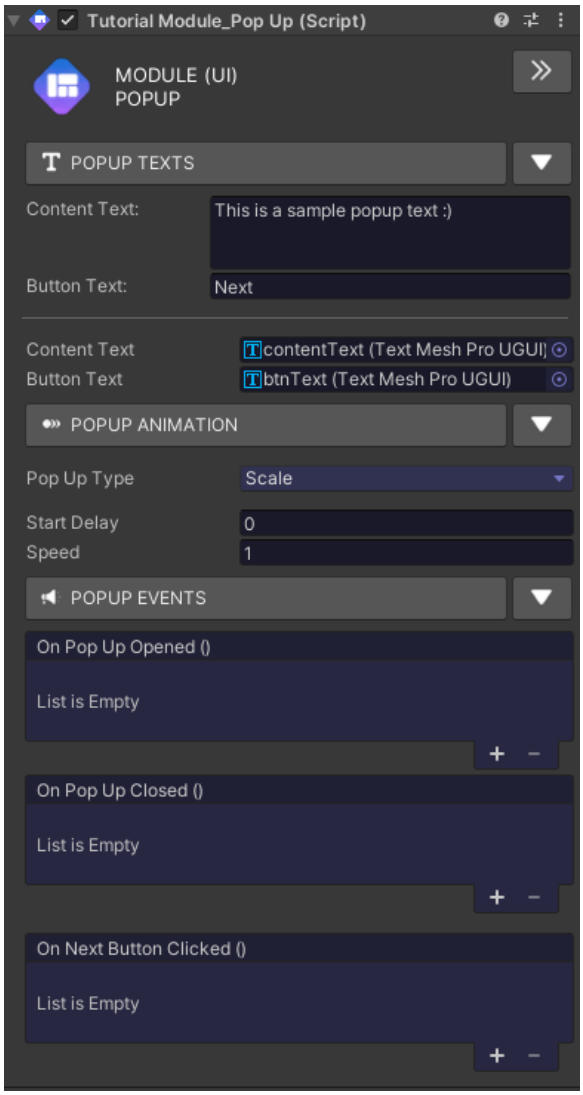
>> You can use the popup module to send information or messages to the player



POPUP TEXTS

Content Text: Popup content text

Button Text: Popup button text



POPUP ANIMATION

Pop Up Type: Popup animation type

Start delay: Popup animation start delay

Speed: Popup animation speed

POPUP EVENTS

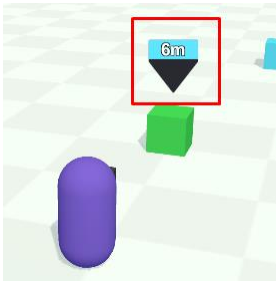
On Pop Up Opened: This event it works when all the animations are finished and the popup is opened

On Pop Up Closed: This event it works when all the animations are finished and the popup is closed

On Next Button Clicked: This event works when the Popup button is clicked

QUEST MARKER MODULE (UI)

>> You can use this module to mark a target.

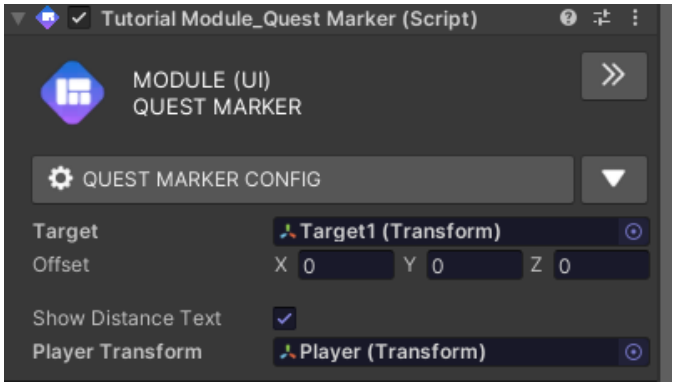


Target: The target object that will be marked

Offset: Offset of the marker to the target

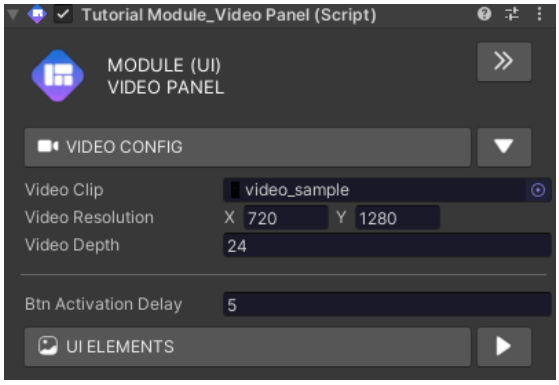
Show Distance Text: Shows the distance between the target object and the player transform.

Player Transform: The player required for the calculation of the distance is (default: Main Camera)



VIDEO PANEL MODULE (UI)

>> With this module you can play a video on canvas



Video Clip: Your video clip

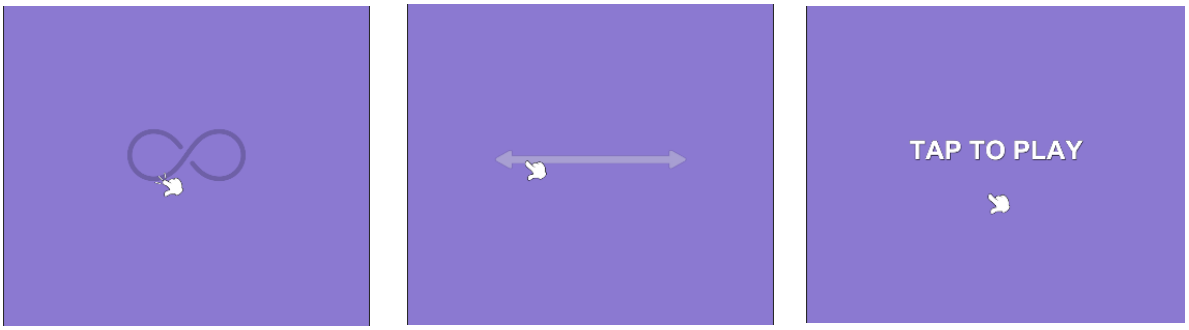
Video Resolution: Your video resolution

Video Depth: Your video depth (recommended: 24)

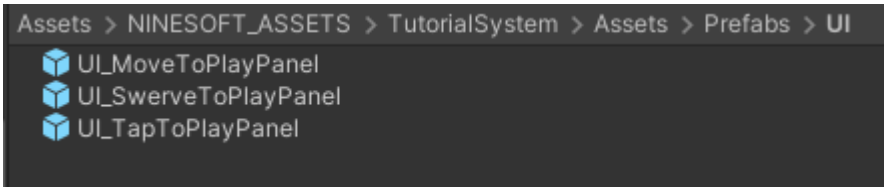
>> The video panel contains a popup module as a child object.

STATIC MODULES (UI)

Static modules have animations and play automatically. In hyper-casual or simple games, you can make static modules work constantly by putting them in the menu.



You can use static modules by dragging and dropping them directly onto the canvas. Static modules are located in the UI folder inside the Prefabs folder



[>> ASSET STORE <<]

[9ninesoft 9.blogspot.com]

[9ninesoft9@gmail.com]
