Tutorial Factory

Tutorial Factory

Presented by HSVStudio

Tutorial Factory gives level designers and game makers an easy way to create customizable tutorials for their contents. It contains many customizable settings and multiple modules that it is capable of producing informative tutorial lessons.

Asset Store

https://assetstore.unity.com/packages/slug/219966

Getting Started

Please take a quick look at the video to see how Tutorial Factory works.

Release Notes

Version 1.1

New:

- Added Position Module to the built-in list. New module could spawn a display icon on top of target.
- Added effect prefab spawning for individual tutorial target. Every tutorial target could have extra prefab spawn on specified location.
- Added 'useRigidBodyTag' option for tag filtering in Trigger Config. If not use rigidbody for tag, it would use tag of collider/trigger's gameobject.
- Added new Position Module demo to the demo scene.

Bug Fix and Updates:

- Fixed Stage state and Tutorial state not being correctly initialized.
- Fixed Tutorial state name not updating on runtime if state changed.
- Fixed inspector error for module config.
- Updated inspector look for better displaying.

Guides: Setup Tutorial Factory

Follow our handy guides to get started on the Tutorial Factory as quickly as possible:

Guides

Create Tutorial Manager

ⓘ **Good to know:** Tutorial Manager can be created on runtime or through Editor menu bar

---

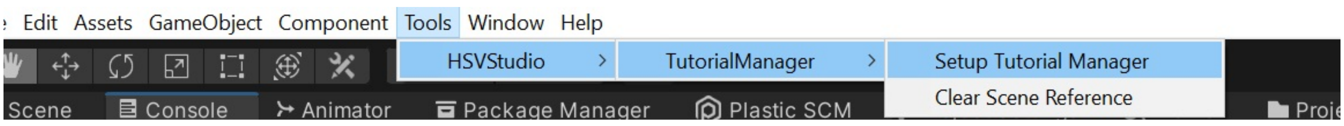
1. The basics

Tutorial Manager is the core component of Tutorial Factory. It contains all the stages and tutorials configurations that would be used at runtime. Tutorial Manager could be disable or deleted once you are done with the tutorials.

---

2. Creating Tutorial Manager Component

- Click on Tools -> HSVStudio -> TutorialManager -> Setup Tutorial Manager. This creates new Tutorial Manager gameobject if no existing Tutorial Manager is found. It will also setup and initialize Tutorial Manager's on scene references on creation.



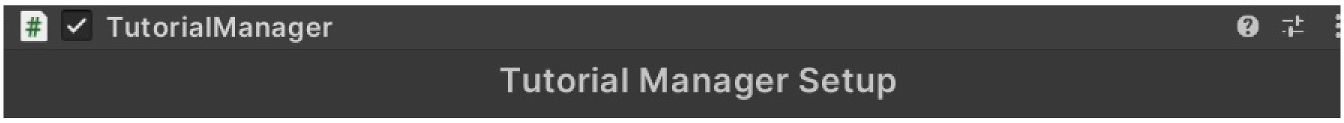
Picture 1. Create Tutorial Manager

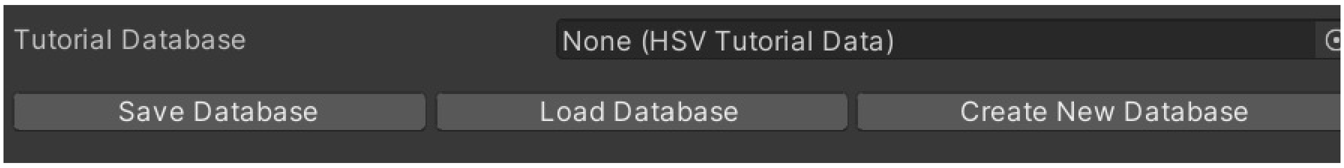
ⓘ **Tutorial Manager** component would add **PhysicsRaycaster** component onto Main Camera used by Tutorial Factory. **PhysicsRaycaster** is used to detect pointer event on scene object as seen through Main Camera. **Please remember to remove PhysicsRaycaster** if you find that it is not removed by Tutorial Manager on its removal.

2.1 Setup Tutorial Manager

2.1.1 Setup Database Asset

- 'Create New Database' button creates database resource assets.
- 'Load Database' button loads stored configuration to the existing Tutorial Manager.
- 'Save Database' button saves Tutorial Manager configurations to the asset resource.




Tutorial Database | None (HSV Tutorial Data)

Picture 2. Buttons for creating assets

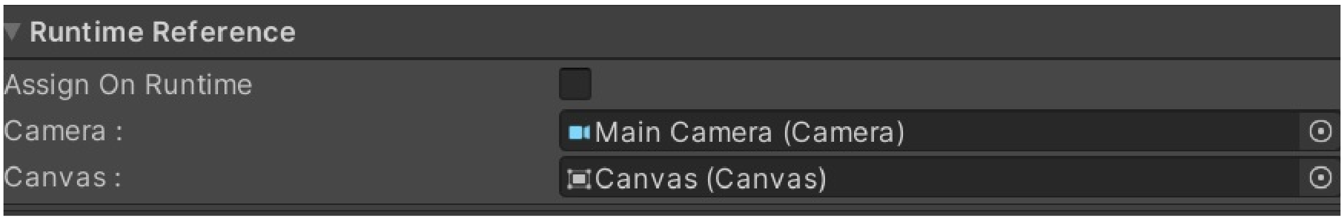
Database asset file is used to store users' level design information. Please be aware that scene objects are referenced using HSVExposeReferenceMgr component which is hidden on hierarchy. If you simple make a new scene and copy over scene objects over, and then load the database asset file, the scene object reference may not work for the Tutorial Objects.

You could save configuration on PlayMode, and then load back the configuration on Editor Mode.

2.1.2 Setting Up Reference

- Please setup references for Camera and Canvas field.

Camera would be used by Tutorial Manager to track scene objects. Canvas is used to position modules UI. If 'Assign On Runtime' is checked, Camera and Canvas would be searched on runtime if there are no reference assigned by user.


Runtime Reference
Assign On Runtime
Camera : Main Camera (Camera)
Canvas : Canvas (Canvas)

Picture 3. Runtime References

Create Stages

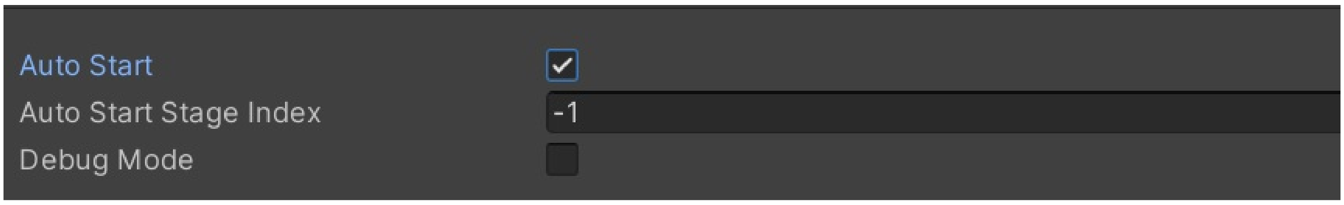
Stage is the fundamental of tutorial steps. You can add as many stage as you want. Each stage can have multiple Tutorial Objects and different configurations.

Please be aware that only one stage can be played at a time.

- **Auto Start**: Automatically starts Tutorial Stage when entering PlayMode.
- **Auto Start Stage Index**: Auto starting stage index. -1 starts from first element.

Stage Index starts from 1, and increment down the array. It is using array position plus one notation. Do not use index 0.

Debug Mode: Outputs Tutorial Manager running message to the console. You could use this to check the status of running tutorials.


Auto Start: ✓
Auto Start Stage Index: -1
Debug Mode: ☐

Picture 1. Stage Settings

1. Add Stages

Click on '+' button to add your first Stage Object.

- **Selected Stage Step**: Indicates which stage object is selected on Editor.
- **Current Tutorial Stage Step**: Current playing stage index of the Tutorial Manager.

This stage step is managed by Tutorial Manager to do play and stop actions. Users do not need to modify this.

- **Stage Object Name**: Playing Stage Object name.
- **Play State**: State of current playing stage.

Idle: The stage is not playing.

Start: Stage is started.

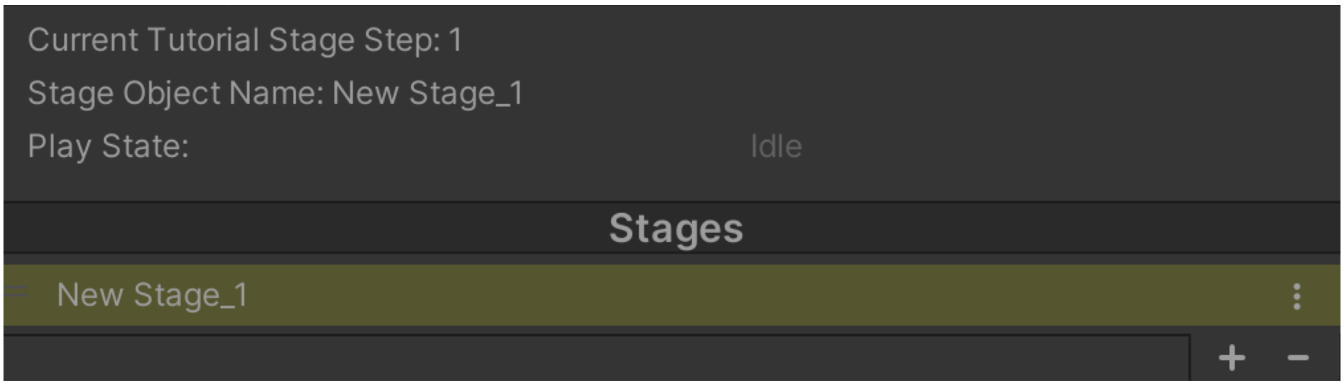
End: Stage is ended.

Remember that the name and state are only shown on PlayMode. It only indicates the status of stage object obtained using currentStageStep index. The stage selected on Editor does not change the currentStageStep index of Tutorial Manager.

You could delete selected Stage Object by clicking on '-' button.

The Stage Objects could be reordered on the list by dragging it up or down. This would change its playing index accordingly.

Selected Stage Step: 1 / 1


Current Tutorial Stage Step: 1
Stage Object Name: New Stage_1
Play State: Idle

Picture 2. Stage Objects

Now let's move on to the configuration of individual Stage Object.

Stage Object Configuration

Each Stage Object can have their own configuration. Tutorial Manager would only use its name and index to track the position of it in the list. All the configuration of Stage Object could be saved in the Database asset file which is described in # **setup-database-asset**

1. Stage Configuration

- **Stage Index**: Stage position in the list starting from 1.
- **Name**: Name of the Stage Object.

Both Index and Name could be used to find the Stage Object in the list. It is highly recommended to change default name to certain identifiable name.nt

Please be aware that stage names cannot be the same, as this would mess up searching.

- **Start Object Index**: Autostart Tutorial Object index starting from 1. Value -1 would start Tutorial Object list from first element.

Stage Index starts from 1, and increment down the array. It is using array position plus one notation. Do not use index 0.

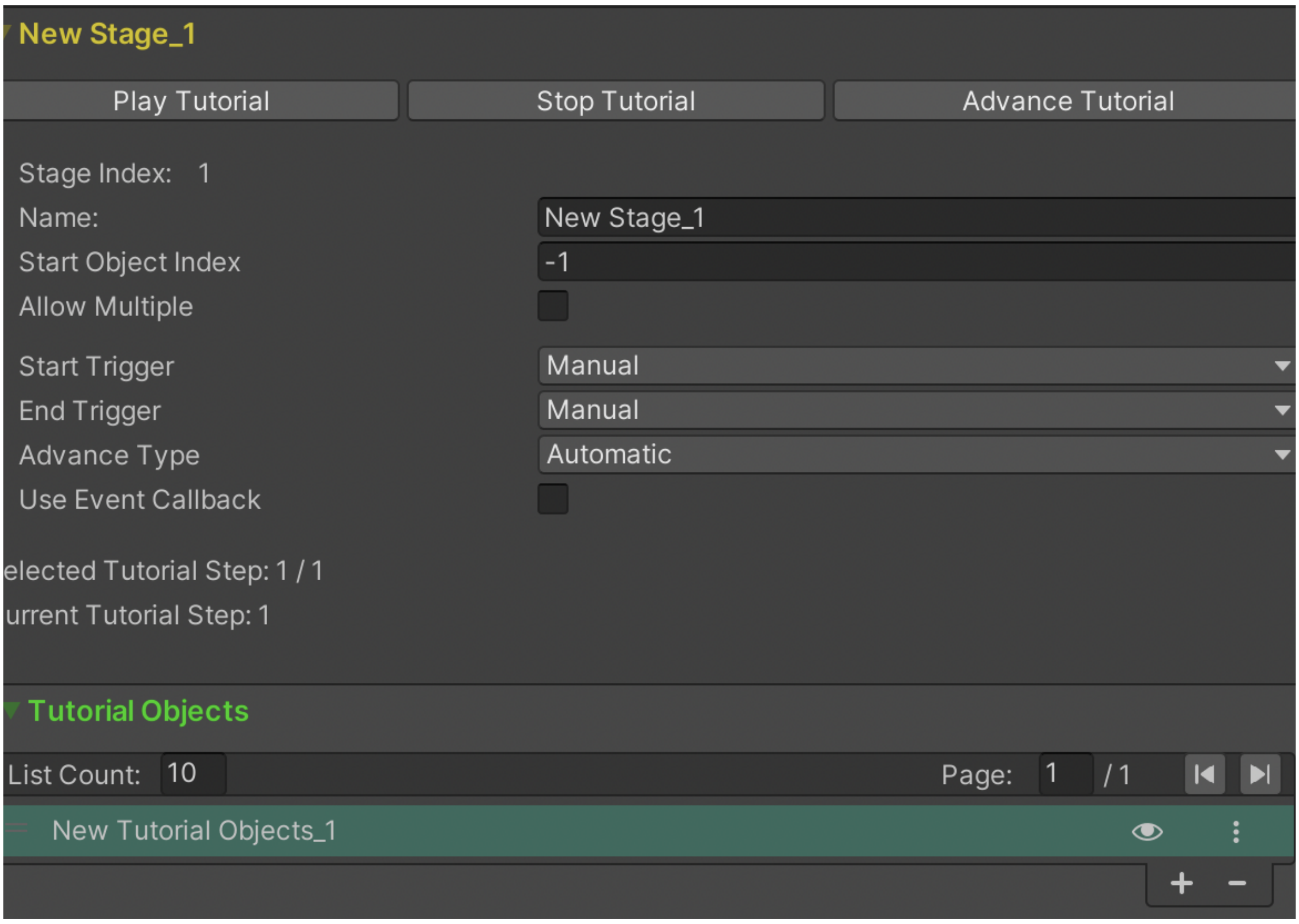
- **Allow Multiple**: Should allow multiple Tutorial Objects running for the current stage.

Use this option cautiously, as it may cause unwanted behaviour with improper settings. This will move current tutorial step to the most recent playing Tutorial Objects. It is recommended to set Advance Type to None in order to use this option.

- **Seletected Tutorial Step**: Current selected Tutorial Object on the Editor.
- **Current Tutorial Step**: Current tracking index used by Stage Object.
- **Tutorial Object**: Playing Tutorial Object name.
- **Play State**: State of current playing Tutorial Object.

1. Idle: Tutorial is not playing.
2. Start: Tutorial is started.
3. End: Tutorial is ended.

Remember that the name and state are only shown on PlayMode. It only indicates the status of Tutorial object obtained using currentStep index. The Tutorial Object selected on Editor does not change the currentStep index of Stage Object.


New Stage_1

Play Tutorial | Stop Tutorial | Advance Tutorial

Stage Index: 1
Name: New Stage_1
Start Object Index -1
Allow Multiple
Start Trigger Manual
End Trigger Manual
Advance Type Automatic
Use Event Callback

elected Tutorial Step: 1 / 1
urrent Tutorial Step: 1

Tutorial Objects
List Count: 10 Page: 1 / 1
New Tutorial Objects_1

Picture 1. Stage Object Configuration

2. Trigger Configuration

Stage Object can be started/ended using different types of Trigger Configuration.

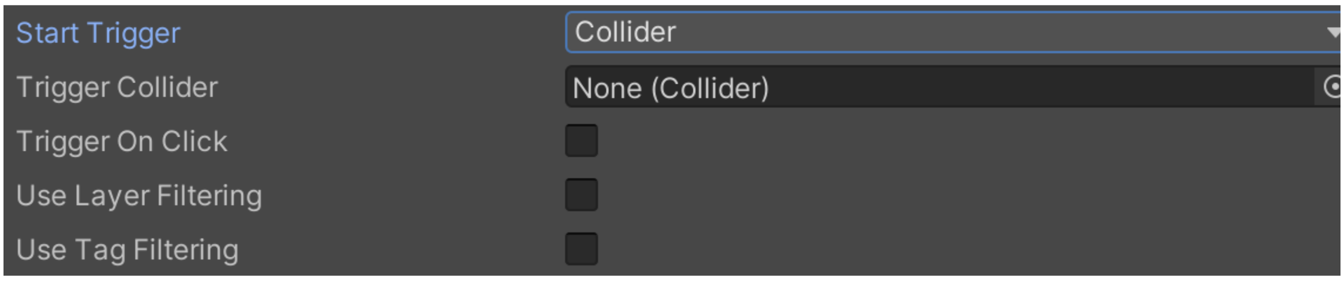
Manuel

The stage object would be started/ended manually would through auto advance from previous stage.

Collider

TriggerObject component would be added to the referenced collider during initialization. In Start Trigger, TriggerObject would be removed when stage starts. TriggerObject would call PlayStage method when the colliding conditions is met. In End Trigger, TriggerObject would be removed when stage ends. TriggerObject would call StopStage method when the colliding conditions is met

- **Trigger Collider**: Target collider component to detect collision.
- **Trigger On Click**: Should the stage be triggered through mouse clicking.
- **Use Layer Filtering**: Should layer filtering be used for collision detection. Please specify target layer if check.
- **Use Tag Filtering**: Should tag filtering be used for collision detection. Please specify target tag if check.



Start Trigger	Collider
Trigger Collider	None (Collider)
Trigger On Click	☐
Use Layer Filtering	☐
Use Tag Filtering	☐

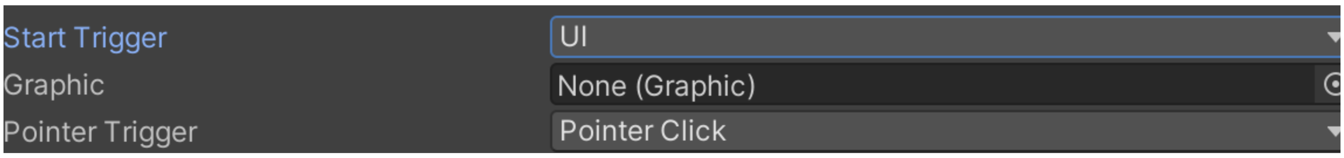
Picture 2. Collider Trigger

UI

TriggerObject component would be added to the referenced UI element during initialization. In Start Trigger, TriggerObject would be removed when stage starts. TriggerObject would call PlayStage method when the colliding conditions is met. In End Trigger, TriggerObject would be removed when stage ends. TriggerObject would call StopStage method when the colliding conditions is met

- **Graphic:** Target UI element to detect clicking event.
- **Pointer Trigger:** Types of UI detection.

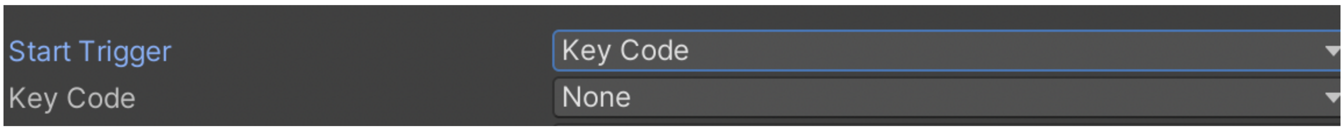
> **Pointer Click**: Trigger event when clicked.
>
> **Pointer Down**: Trigger event when mouse down or touch down on UI.
>
> **Pointer Up**: Trigger event when mouse up or touch up on UI.
>
> **Pointer Enter**: Trigger event when mouse enter or touch enter on UI.
>
> **Pointer Exit**: Trigger event when mouse exit or touch exit on UI.


Start Trigger: UI
Graphic: None (Graphic)
Pointer Trigger: Pointer Click

Picture 3. UI Trigger

KeyCode

Triggers start/end event when specified key is pressed.


Start Trigger: Key Code
Key Code: None

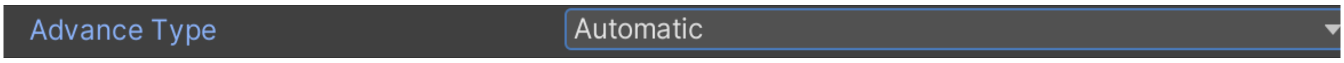
Picture 4. KeyCode

3. Advance Type

Stage Objects could advance to next stage using different types of advancing logics.

- **None**: Do nothing.
- **Automatic**: Plays next Stage Object when current playing stage is stopped.
- **Index**: Plays Stage Object with specified index when current playing stage is stopped.
- **Name**: Plays Stage Object with specified name when current playing stage is stopped.

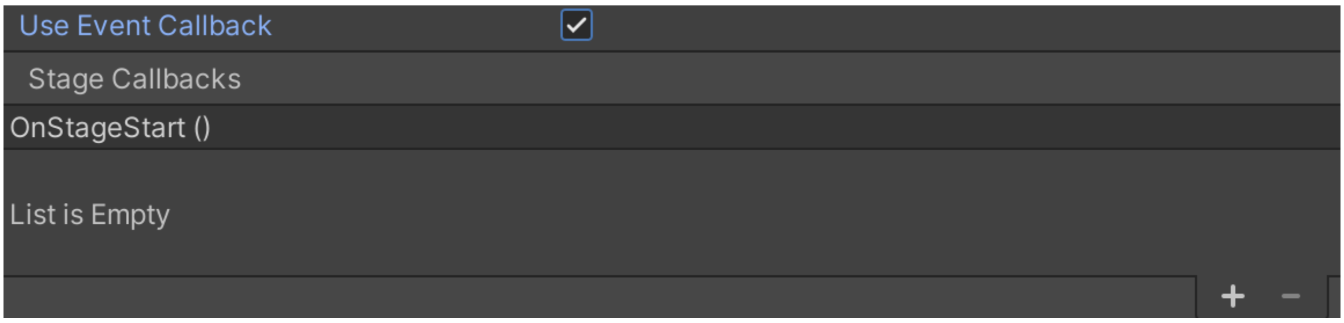
> Use different combination of advance configs could accomplish complicated scenarios.

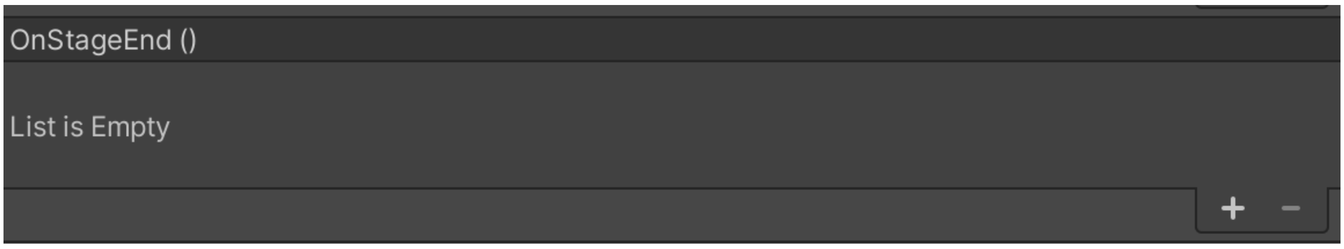

Advance Type: Automatic

Picture 5. Advance Type

4. Event Callback

- **Use Event Callback:** Should use events callback on stage playing.
- **OnStageStart**: Event would be called when stage starts playing
- **OnStageEnd**: Event would be called when stage ends.


Use Event Callback ✓
Stage Callbacks
OnStageStart ()
List is Empty


OnStageEnd ()

List is Empty

Picture 6. Event Callback

Tutorials Objects

ⓘ Tutorial Objects are the core of the Stage Object . You can add as many Tutorial Objects as you want. Tutorial Objects would use various Modules to achieve various scenarios.

Tutorial Object has certain similar configuration as Stage Object, such as Index, Name, Start Trigger, End Trigger and so on.

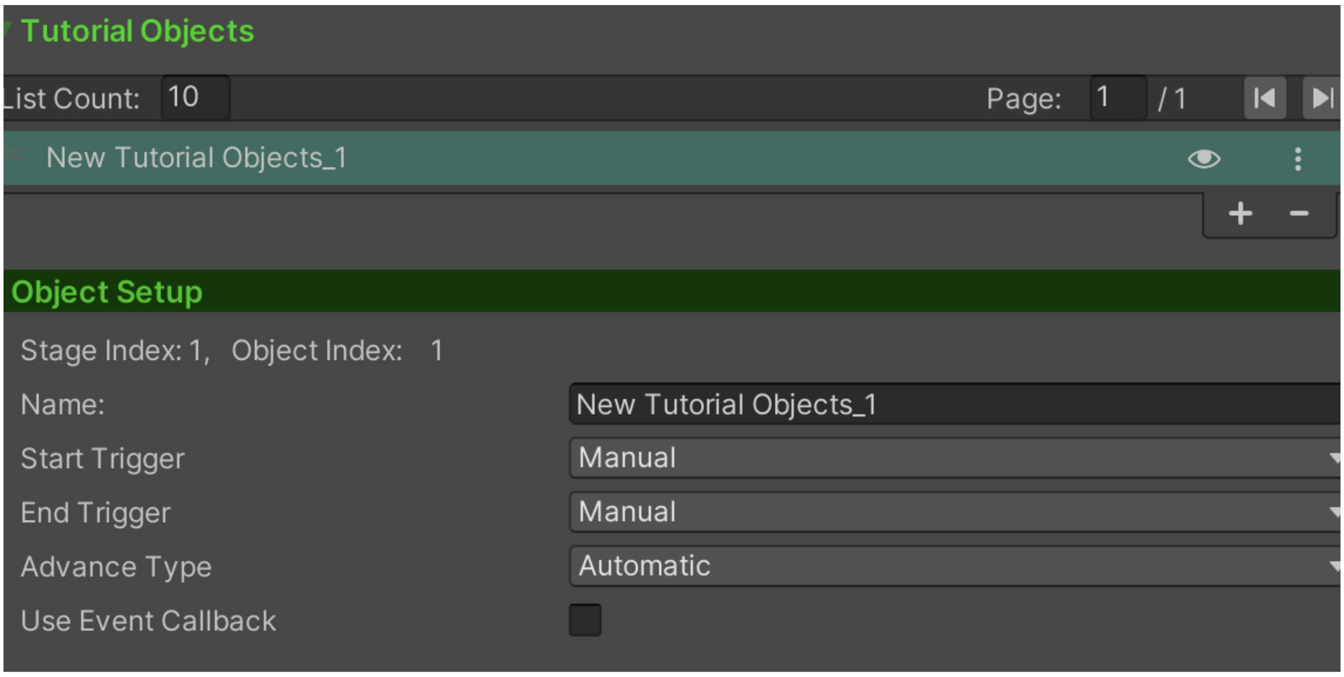
---

1. Add Tutorial Object

Click on '+' button to add your first Tutorial Object.

ⓘ You could delete selected Tutorial Object by clicking on '-' button.

The Tutorial Objects could be reordered on the list by dragging it up or down. This would change its playing index accordingly.


Tutorial Objects

List Count: 10   Page: 1 / 1

New Tutorial Objects_1

Object Setup

Stage Index: 1, Object Index: 1

Name:	New Tutorial Objects_1
Start Trigger	Manual
End Trigger	Manual
Advance Type	Automatic
Use Event Callback	☐

Picture 1. Tutorial Objects Configuration

2. Tutorial Object Configuration

- **Stage Index**: Index of the stage, the Tutorial Object belongs to.
- **Object Index**: Tutorial Object position in the list starting from 1.
- **Name**: Name of the Tutorial Object.

> ⓘ Both Index and Name could be used to find the Tutorial Object in the list.

2.1 Trigger Configuration

Tutorial Object can be started using different types of Trigger Configurations. Please refer to **# triggerconfiguration**

2.2 Advance Type

Tutorial Objects could advance to next tutorial using different types of advancing logics. Please refer to **# advancetype**

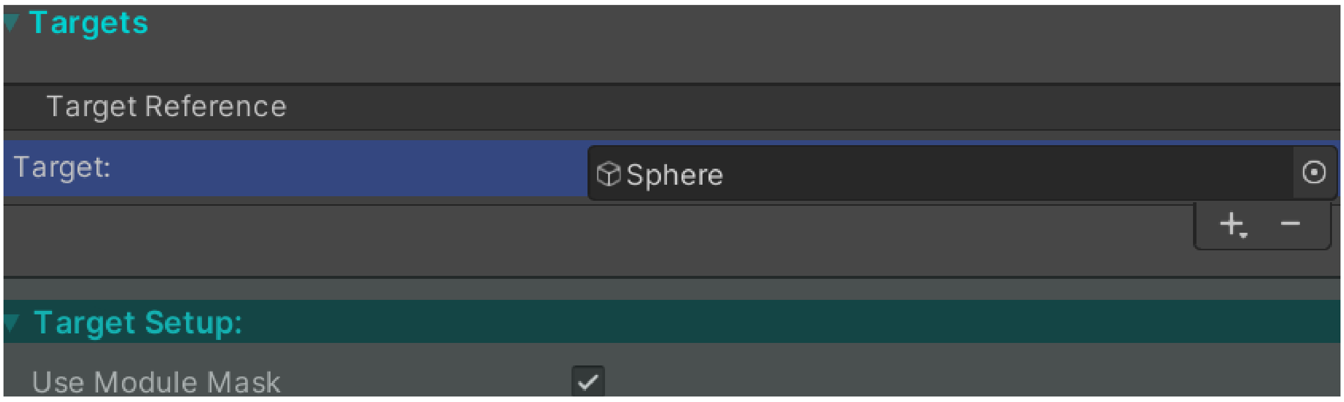
2.3 Event Callback

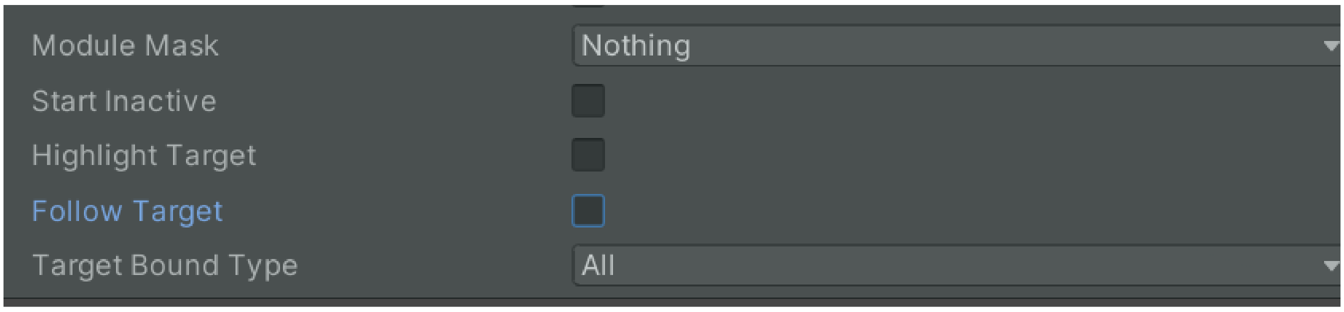
Please refer to **# 4. Event Callback**

3. Targets

Targets contains a list of objects which are used by Modules to track. Modules could track scene objects, prefabs, UI elements and so on.

> ⓘ Please feel free to try out different combinations of targets for the Tutorial Objects.


Targets
Target Reference
Target: Sphere
Target Setup:
Use Module Mask ✓


Module Mask | Nothing
Start Inactive | ☐
Highlight Target | ☐
Follow Target | ☐
Target Bound Type | All

Picture 2. Target Configuration

- **Use Module Mask**: Should selected target mask out modules.
- **Module Mask**: Layermask of all the preconfigured modules types. Only selected modules would be used for the target.
- **Start Inactive**: Target would be disable at start.
- **Highlight Target**: Target would be highlighted using OutlineController.

OutlineController highlights out scene objects. It only works for Renderers. It supports Builtin and URP render pipeline.

- **Follow Target**: Should modules follow target when playing Tutorial Object.
- **Target Bound Type**: Types used to calculate target bounds.

None: Do nothing with target.

All: All transform renderers including target's children would be calculated.

Recttransform: Recttransform's screen rect would be calculated.

Normally, All type would be enough for scene mesh objects, Recttransform for UI elements.

---

4. Modules

Modules are the main display function of Tutorial Objects. There are several built-in modules each contains different configurations. Two categories of modules are used in this package, including TargetModule and TimeModule. More modules would be added in the future update.

Modules consist of two types of class. HSVModuleConfig class contains various configuration of modules. HSVTutorialModule class utilizes the configuration stored in HSVModuleConfig class for displaying when Tutorial Object starts playing. Remember to subclass both when creating custom modules.

> ⓘ Each module config has its corresponding Module component. Please use correct module prefab for the configuration.

4.1 Module Configuration

- **Module Prefab**: Module prefab spawned on runtime. Would be added under Canvas.

> ⓘ Make sure HSVTutorialModule component or its inherited module component is added to the module prefab when creating new one.

- **Mask Prefab**: Prefab spawned by module for each target. It is mainly used for display purpose.

> ⓘ Make sure HSVUIMask component is added to the mask prefab when creating new one.

- **Override Sprite**: Overrides mask prefab original sprite when module starts.
- **Override Mask Color**: Should the mask prefab sprite color be overridden.
- **Color**: Color of overridden sprite.
- **Instant Transition**: Should module UI appear instantly or use fading in/out animation.
- **Smooth Value**: Transition smoothing value.
- **Fade Time**: Elapsed time of animation transition.
- **Custom Transition**: Custom transition of module prefabs.

> ⓘ With custom transition, Tutorial Module would just enable/disable mask prefab when playing. You could use your own transition tool to do the action such as DOTWeen.

> ⓘ Please remember to call EndModule() method in Tutorial Module component when you are done with transition as this would set module playing status to PlayState.End.

4.2 Target Module

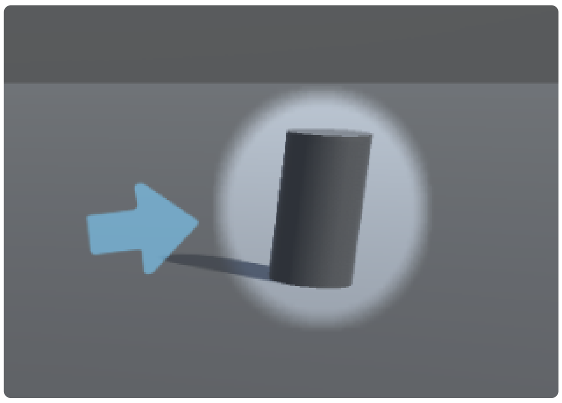
Modules used to track targets on the screen.

> ⓘ You need Targets added to the target list in order to use Target Module.

- **Override Follow Target**: Should module override following target configuration in # **3. Targets**
- **Follow Target**: Should module follow target when overriding Target's Config.
- **Allow No Target**: Should module still run when there is no target(This is used by Popup Module).

4.2.1 Arrow Module

Subclass of Target module. Modules spawn an arrow for each target. The arrow could be positioned manually or use built-in auto position calculation.


Picture 3. Arrow Module

- **Arrow Size**: Display arrow rect size.
- **Arrow Sprite Direction**: Arrow pointing direction in the sprite.

> The original pointing direction in the sprite. (1, 0, 0) is pointing right on the screen. Please specify this when using auto rotation.

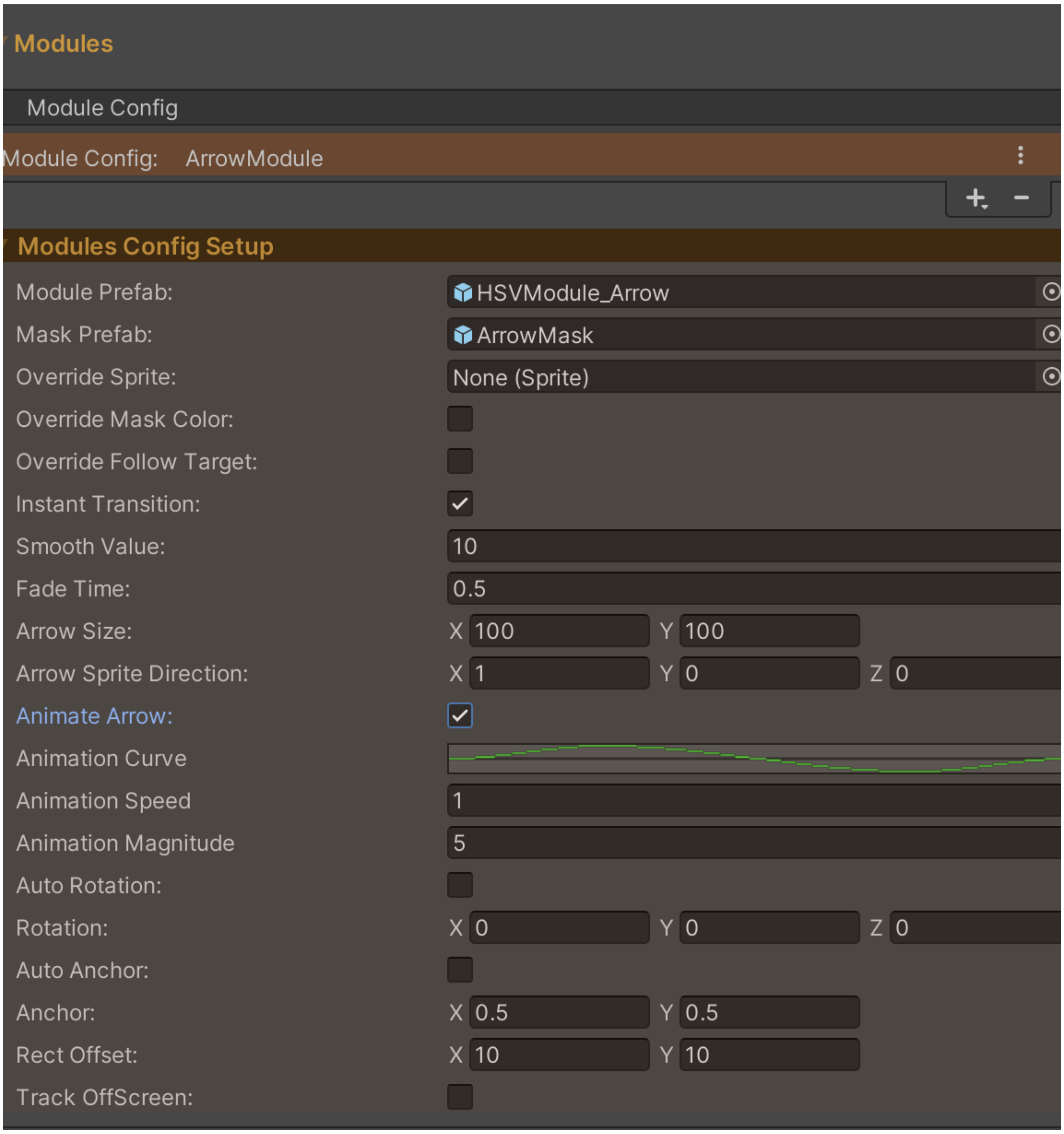
- **Animate Arrow**: Should arrow animate on screen. It would use animation curve to position itself.
- **Animation Curve**: Displacement curve.
- **Animation Speed**: Frequency of displacement.
- **Animation Magnitude**: Displacement magnitude.

- **Auto Rotation**: Should arrow auto rotates itself towards target.
- **Rotation**: If auto rotation is not check, arrow rotates towards this value.
- **Auto Anchor**: Should arrow automatically position around the target with respect to pivot point.
- **Pivot**: Screen point used to auto anchor arrow around target.
- **Anchor**: If not Auto Anchor, arrow will use this to position itself around target

> Anchor point(0, 0) is at top left corner around target, (1, 1) is at bottom right corner around Target.

- **Rect Offset**: Offset displacement from the calculated Target's display rect size.

- **Track OffScreen**: Should arrow continue pointing towards target when it is out of screen.
- **Screen Offset**: Offset distance from edge of the screen when target is out of screen. 0 is at the edge of screen while 1 is at center of screen.



Modules

Module Config

Module Config: ArrowModule

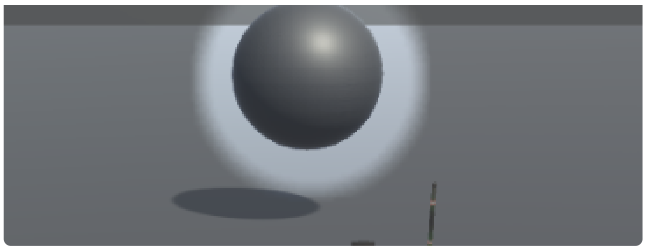
Modules Config Setup

Module Prefab:	HSVModule_Arrow
Mask Prefab:	ArrowMask
Override Sprite:	None (Sprite)
Override Mask Color:	☐
Override Follow Target:	☐
Instant Transition:	✓
Smooth Value:	10
Fade Time:	0.5
Arrow Size:	X 100 Y 100
Arrow Sprite Direction:	X 1 Y 0 Z 0
Animate Arrow:	✓
Animation Curve	
Animation Speed	1
Animation Magnitude	5
Auto Rotation:	☐
Rotation:	X 0 Y 0 Z 0
Auto Anchor:	☐
Anchor:	X 0.5 Y 0.5
Rect Offset:	X 10 Y 10
Track OffScreen:	☐

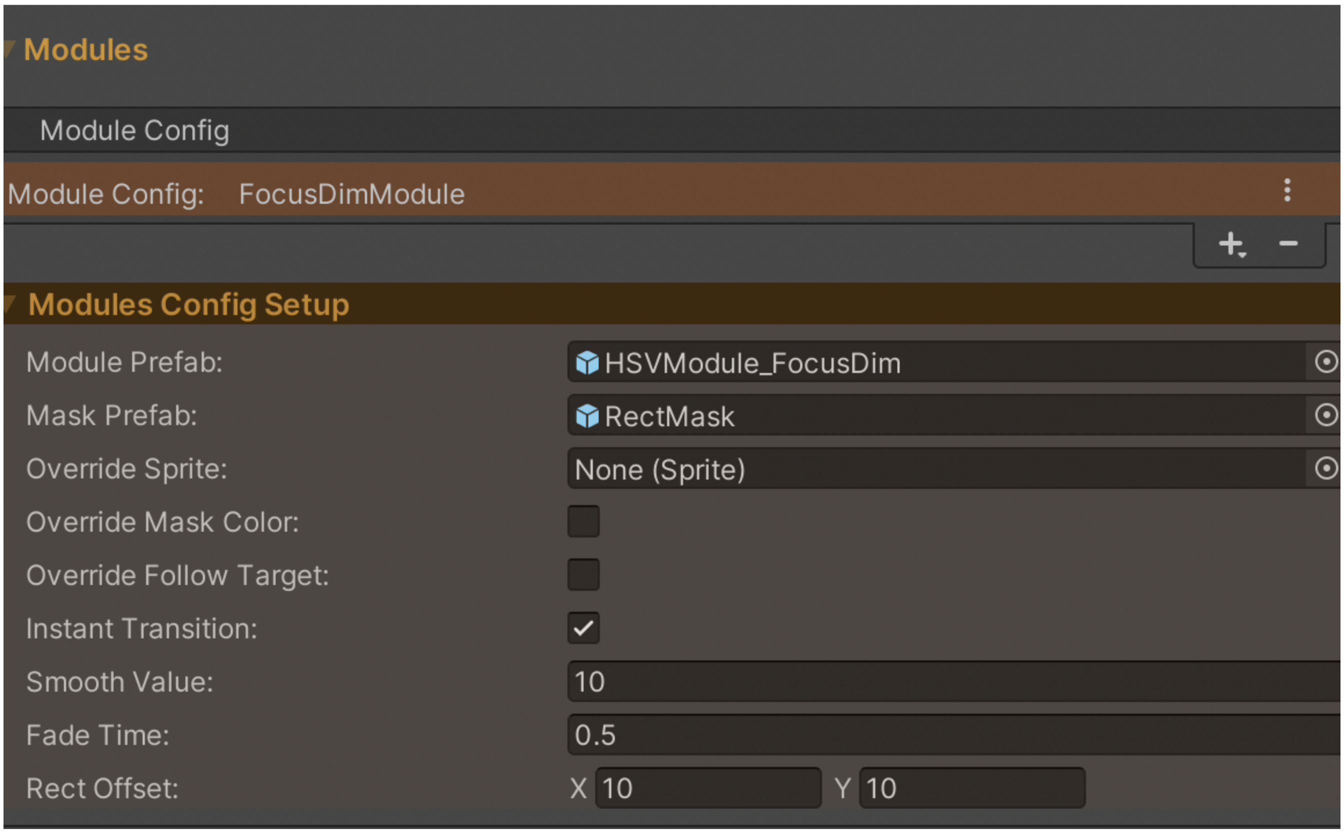
Picture 4. Arrow Module Configuration

4.2.2 Focus Dim Module

Subclass of Target Module. Module would create a screen dimming effect to preset color while masking out Target area.


Picture 5. Focus Dim Module

- **Rect Offset**: Offset displacement from the calculated Target's display rect size. Position value increases mask display size.



Modules

Module Config

Module Config: FocusDimModule

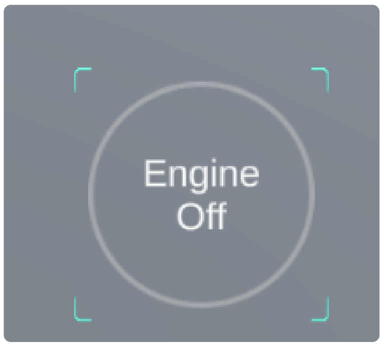
Modules Config Setup

Module Prefab:	HSVModule_FocusDim
Mask Prefab:	RectMask
Override Sprite:	None (Sprite)
Override Mask Color:	☐
Override Follow Target:	☐
Instant Transition:	✓
Smooth Value:	10
Fade Time:	0.5
Rect Offset:	X 10 Y 10

Picture 6. Focus Dim Module

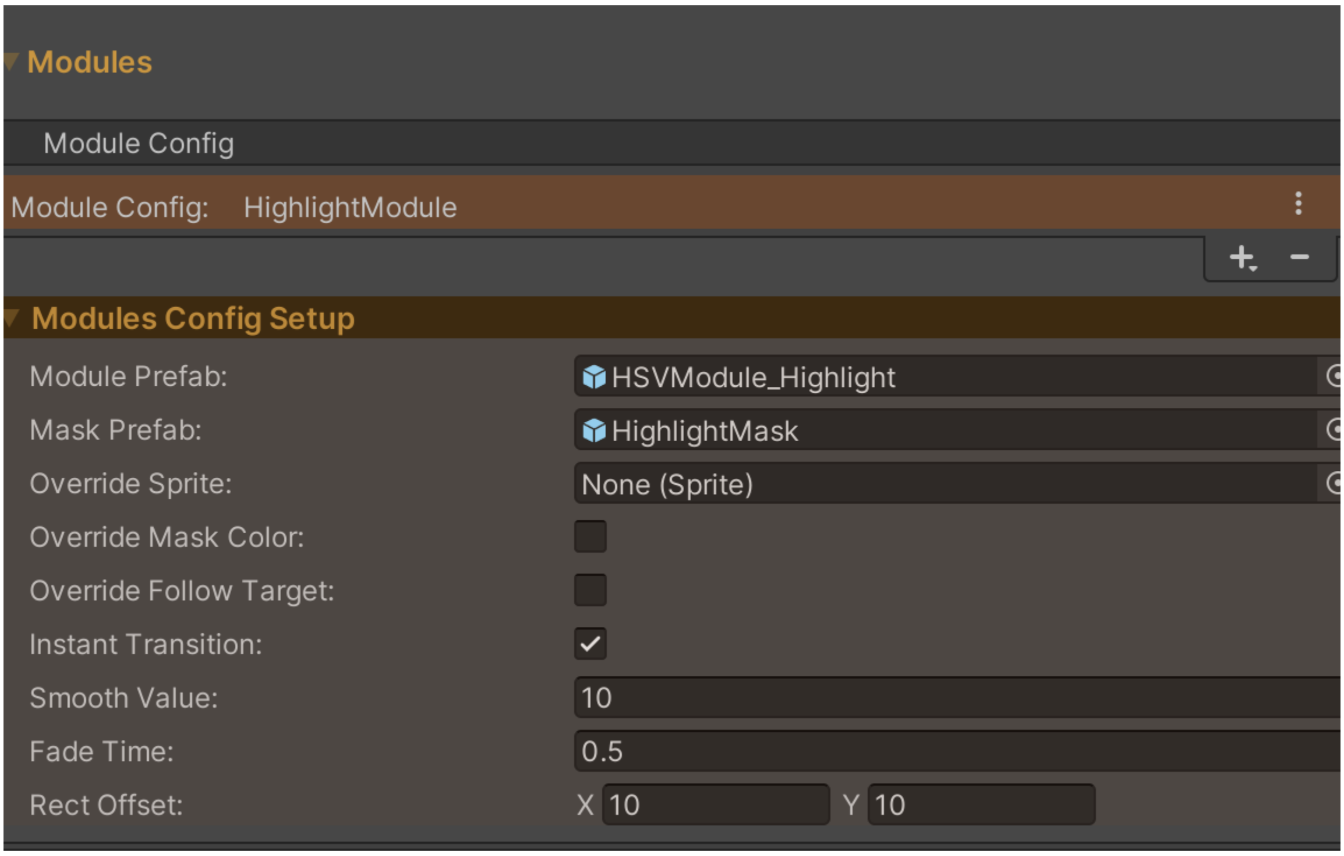
4.2.3 Highlight Module

Subclass of Target Module. Module creates highlighting sprite on the Target position.



Picture 7. Highlight Module

- **Rect Offset**: Offset displacement from the calculated Target's display rect size. Position value increases mask display size.


Modules

Module Config

Module Config: HighlightModule

Modules Config Setup

Module Prefab: HSVModule_Highlight

Mask Prefab: HighlightMask

Override Sprite: None (Sprite)

Override Mask Color: ☐

Override Follow Target: ☐

Instant Transition: ✓

Smooth Value: 10

Fade Time: 0.5

Rect Offset: X 10 Y 10

Picture 8. Highlight Module

4.2.4 Popup Module

Subclass of Target Module. Module creates displaying text on screen around Target.

- **Allow No Target**: Should module continue working without a target. If check, it is similar to InfoDisplay Module.
- **Auto Anchor**: Should arrow automatically position around the target with respect to pivot point.
- **Pivot**: Screen point used to auto anchor arrow around target.
- **Anchor**: If not Auto Anchor, arrow will use this to position itself around target

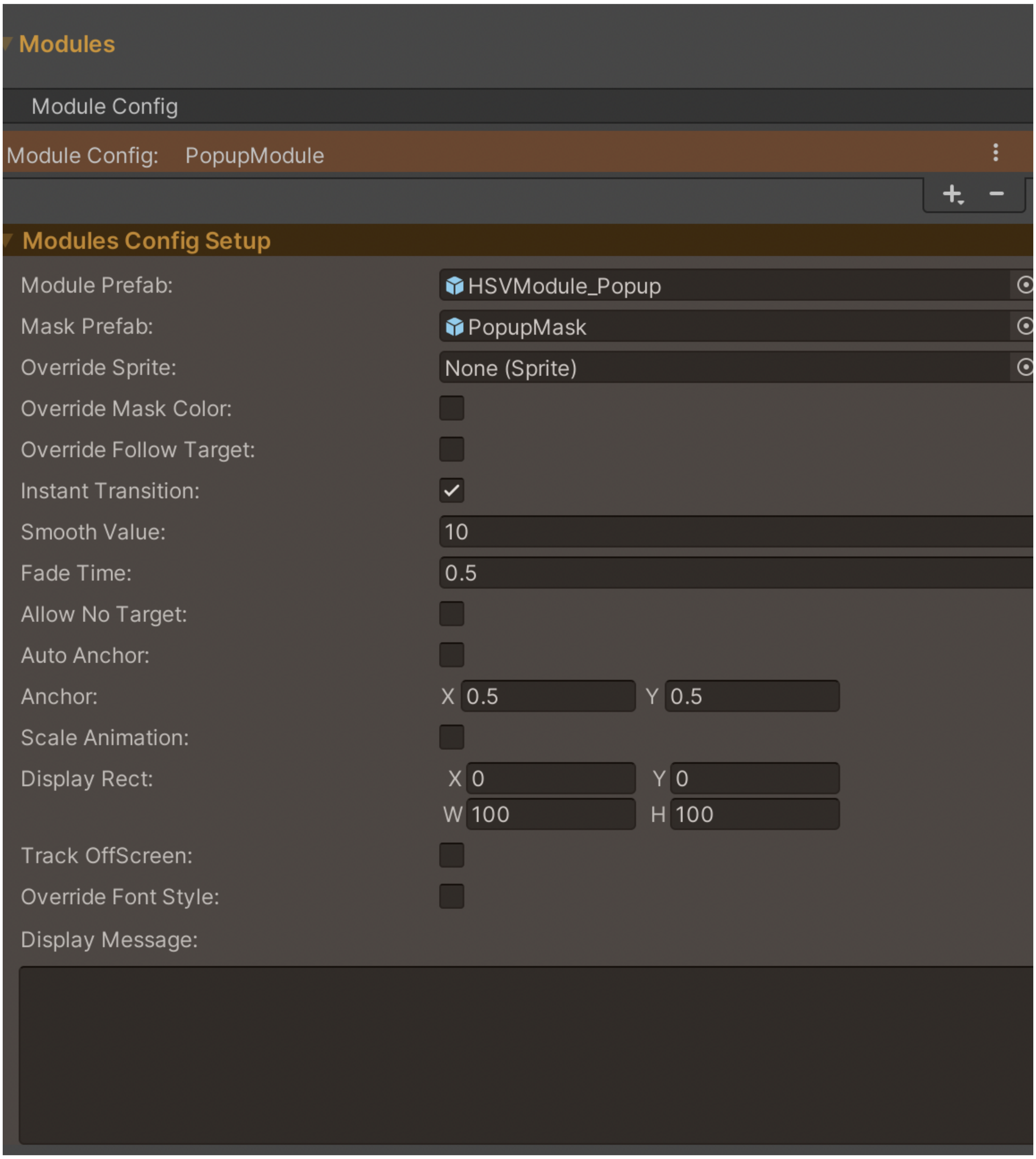
> Anchor point(0, 0) is at top left corner around target, (1, 1) is at bottom right corner around Target.

- **Scale Animation**: Should popup rect scale in/out gradually. Animation would use Smooth Value to do transition.
- **Display Rect**: Size of the displaying rect
- **Track OffScreen**: Should arrow continue pointing towards target when it is out of screen.
- **Screen Offset**: Offset distance from edge of the screen when target is out of screen. 0 is at the edge of screen while 1 is at center of screen.
-

- **Override Font Style**: Should module overrides prefab font style.

> Popup Module is using Unity built-in TextMesh Pro to display text message. This option would override some display text configurations.

- **Display Message**: Text message displayed.



Modules

Module Config

Module Config: PopupModule

Modules Config Setup

Module Prefab:	HSVModule_Popup
Mask Prefab:	PopupMask
Override Sprite:	None (Sprite)
Override Mask Color:	☐
Override Follow Target:	☐
Instant Transition:	✓
Smooth Value:	10
Fade Time:	0.5
Allow No Target:	☐
Auto Anchor:	☐
Anchor:	X 0.5 Y 0.5
Scale Animation:	☐
Display Rect:	X 0 Y 0 W 100 H 100
Track OffScreen:	☐
Override Font Style:	☐
Display Message:	

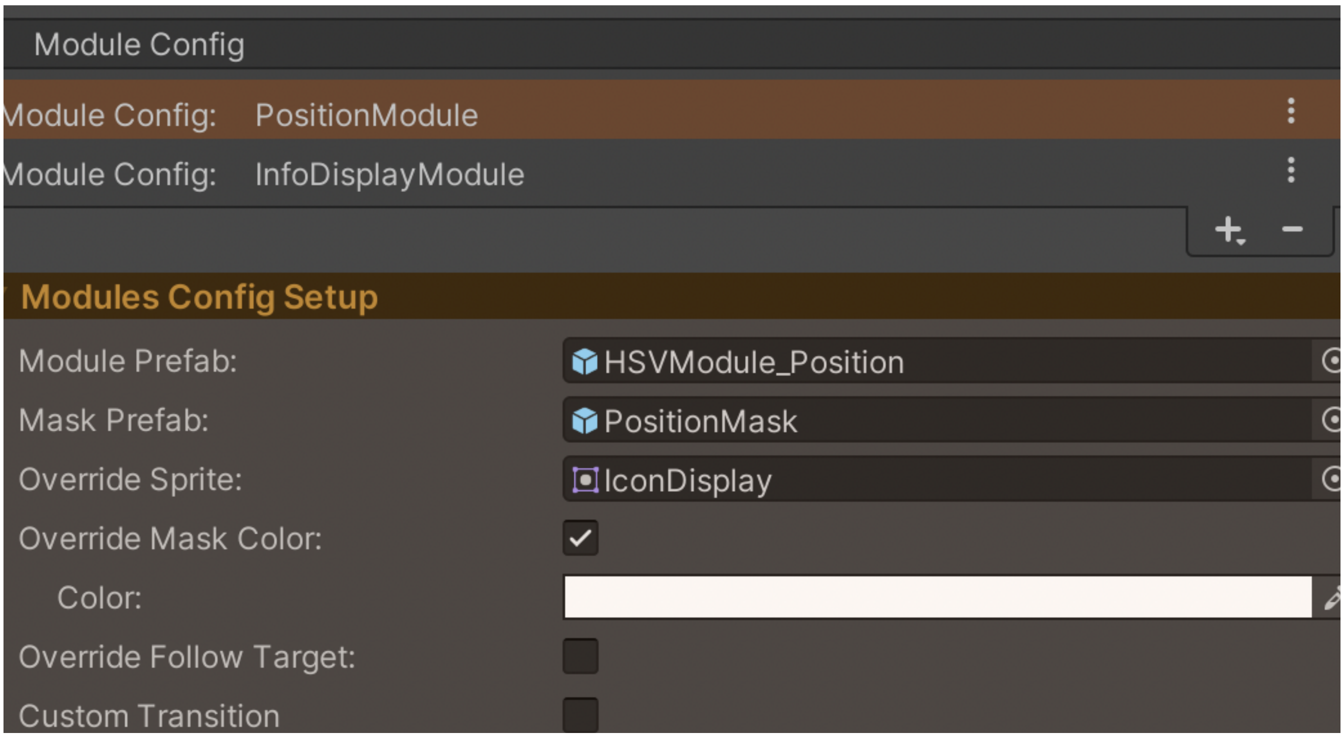
Picture 9. Popup Module Configuration

4.2.5 Position Module

Subclass of Target Module. Module creates displaying icon on top of Target.

Picture 10. Position Module

- **Position Offset:** Display offset from target's world location.
- **Mask Size:** Icon display rect size.
- **Display Arrow:** Should display arrow around icon when icon is far from target.
- **Arrow Sprite:** Overriding arrow sprite when display arrow enabled.
- **Arrow Color:** Arrow display color.
- **Arrow Size:** Arrow sprite rect size.
- **Arrow Sprite Direction:** Original arrow sprite pointing direction. (0, 1, 0) means pointing upward.
- **Arrow Offset:** Offset distance from display icon.
- **Track OffScreen**: Should Position Module continue tracking target when it is out of screen.
- **Screen Offset**: Offset distance from edge of the screen when target is out of screen. 0 is at the edge of screen while 1 is at center of screen. This is also used as screen bound to determine if target is out of screen.


Module Config

Module Config: PositionModule

Module Config: InfoDisplayModule

Modules Config Setup

Module Prefab: HSVModule_Position

Mask Prefab: PositionMask

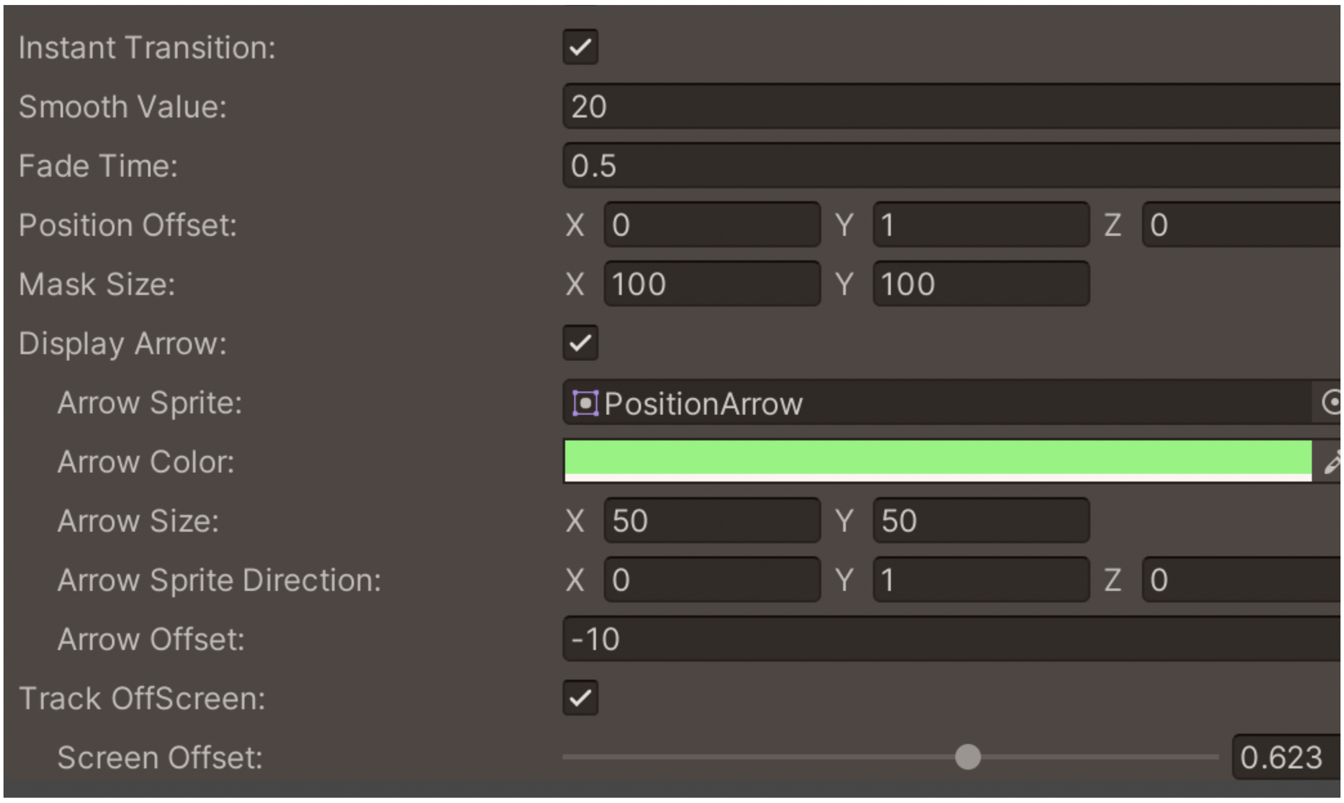
Override Sprite: IconDisplay

Override Mask Color: ✓

Color:

Override Follow Target:

Custom Transition



Instant Transition: ✓
Smooth Value: 20
Fade Time: 0.5
Position Offset: X 0 Y 1 Z 0
Mask Size: X 100 Y 100
Display Arrow: ✓
Arrow Sprite: PositionArrow
Arrow Color: 🟩
Arrow Size: X 50 Y 50
Arrow Sprite Direction: X 0 Y 1 Z 0
Arrow Offset: -10
Track OffScreen: ✓
Screen Offset: 0.623

Picture 11. Position Module Configuration

4.3 Time Module

Module runs without target. Such as displaying information text when there is no target highlighting, or do a time delay on tutorial playing.

4.3.1 InfoDisplay Module

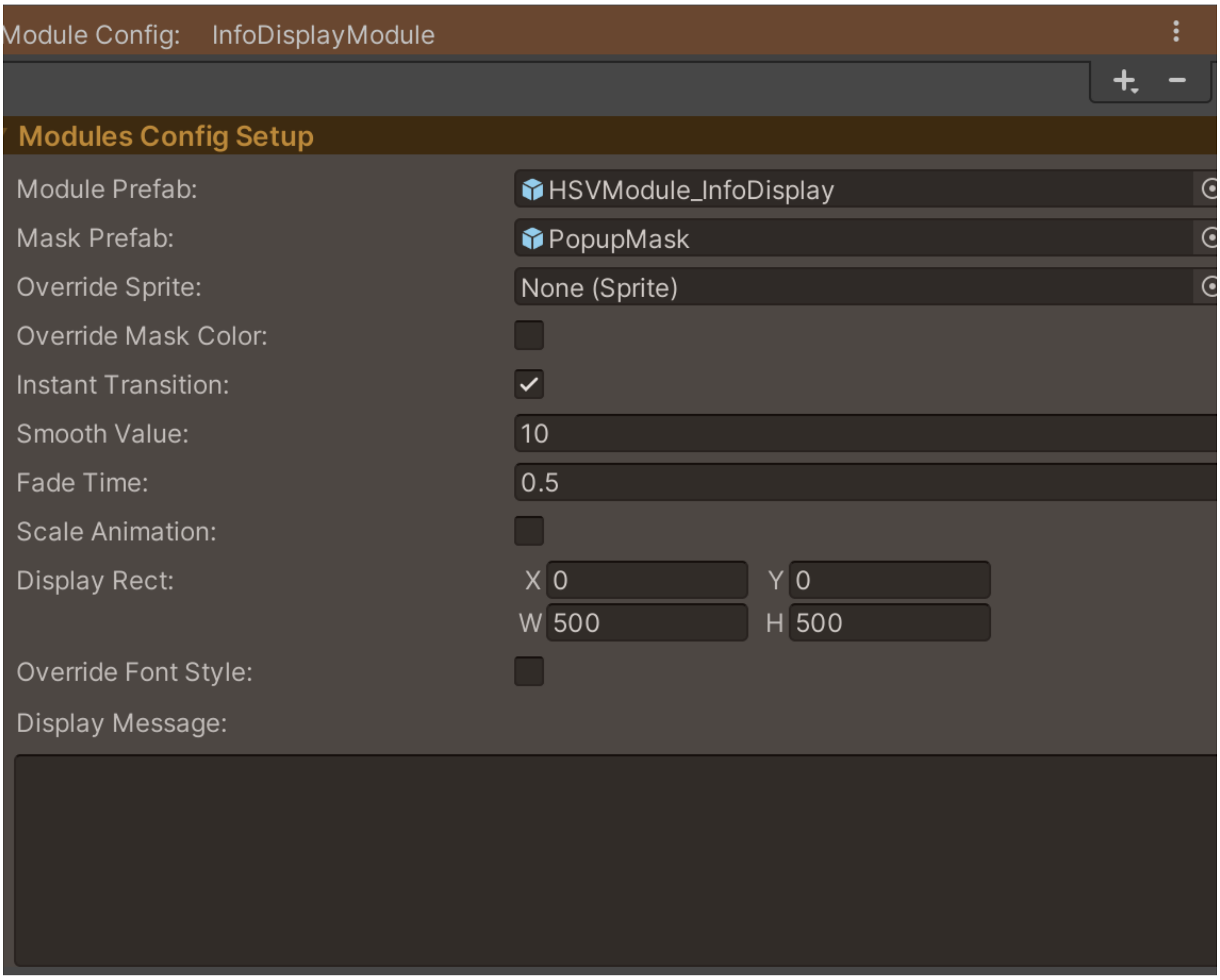
Subclass of Time Module. Similar to Popup module, this module is mainly used to display informative text message on screen.

- **Scale Animation**: Should popup rect scale in/out gradually. Animation would use Smooth Value to do transition.
- **Display Rect**: Size of the displaying rect.
- **Override Font Style**: Should module overrides prefab font style.

> ⓘ Popup Module is using Unity built-in TextMesh Pro to display text message. This option would override some display text configurations.

- **Display Message**: Text message displayed.

Modules
Module Config


Module Config: InfoDisplayModule

Modules Config Setup

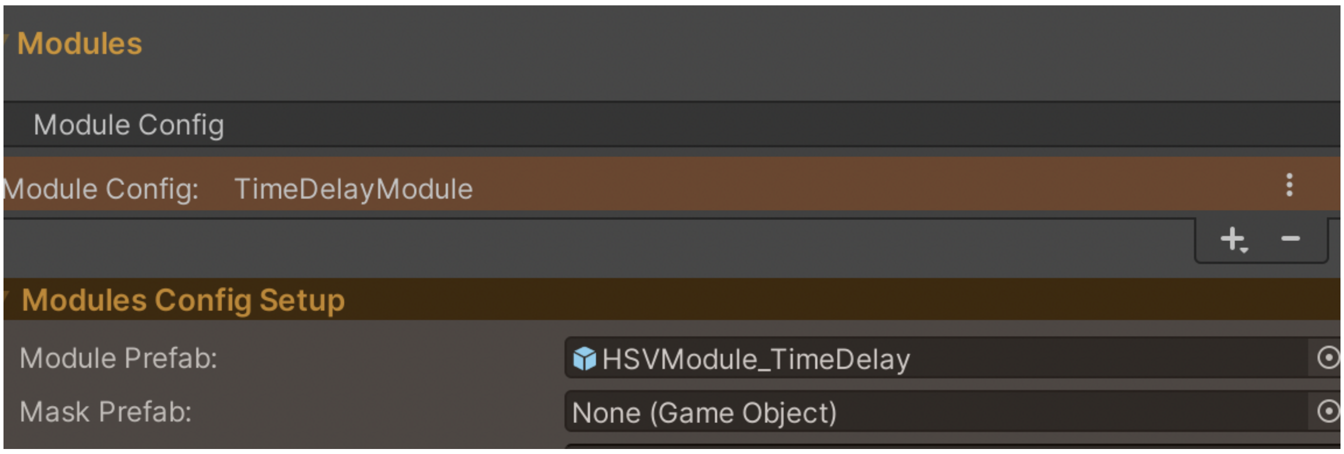
Module Prefab:	HSVModule_InfoDisplay
Mask Prefab:	PopupMask
Override Sprite:	None (Sprite)
Override Mask Color:	☐
Instant Transition:	✓
Smooth Value:	10
Fade Time:	0.5
Scale Animation:	☐
Display Rect:	X 0 Y 0 W 500 H 500
Override Font Style:	☐
Display Message:	

Picture 12. InfoDisplay Module Configuration

4.3.2 Time Delay Module

Subclass of Time Module. Module does not need mask prefab for displaying. Its main function is to delay next tutorial step by certain amount of time.

- **Delay Time**: Elapsed time of this module.
- **Auto End Tutorial**: Tutorial would automatically end when delay time elapsed. Otherwise, tutorial would be still in playing state and do nothing.

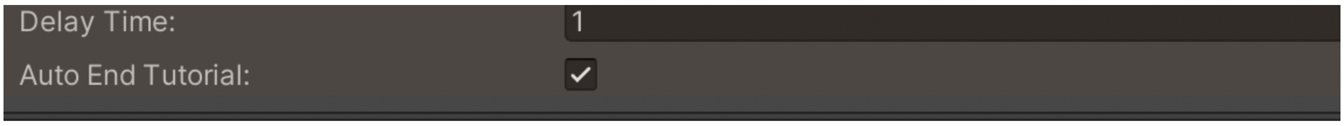

Modules

Module Config

Module Config: TimeDelayModule

Modules Config Setup

Module Prefab:	HSVModule_TimeDelay
Mask Prefab:	None (Game Object)


Delay Time: 1

Auto End Tutorial: ✓

Picture 13. Time Delay Module

4.4 Custom Module

You can subclass HSVModuleConfig class, or HSVTargetModuleConfig and HSVTimeModuleConfig for your own custom module.

> ⓘ Remember to create new custom modules subclassing HSVTutorialModule or HSVTargetModule and HSVTimeModule when creating new module configs class.

Utility Classes

Following are some utility classes that used in combination with Tutorial Manager.

---

1. HSVMainTimer

Runtime action running manager. Used by Tutorial Manager to perform time related actions.

---

2. HSVObjectPool

Runtime pooling for performance purpose.

---

3. HSVTargetTrigger

You could use this component to add scene object into Targets of Tutorial Object on runtime.

- **Stage Index**: Index of Stage Object that Tutorial Object belongs to.
- **Tutorial Index**: Index of Tutorial Object that this target belongs to.
- **Target Config**: The configuration of this target used by modules. Please refer to # **3. Targets**

---

4. HSVTriggerObject

Tutorial Manager would add this component to the Target objects when playing Stage/Tutorial Objects on runtime. This component is used to trigger Stage/Tutorial start or stop events.

- **Collider Config**: Collider config of Start/End Trigger of Stage/Tutorial Object.
- **Graphic Config**: Graphic config of Start/End Trigger of Stage/Tutorial Object.
- **IsStartTrigger**: Is this trigger object used to Start or End event.
- **IsStage**: Is this trigger object used by Stage or Tutorial Object.

API References

HSVTutorialManager

public void Setup();

Setup references of Canvas, Camera, Raycaster components for Tutorial Manager.

public void SetCamera(Camera camera);

Sets camera reference to 'camera'.

Stages API

public void PlayStage(int stageStep, bool autoStart = true);

public void PlayStage(string stageName, bool autoStart = true);

public void PlayStage(bool autoStart = true);

public void StopStage();

public HSVTutorialStage GetCurrentStageObject(int stageStep);

public HSVTutorialStage GetCurrentStageObject(string stageName);

Tutorial Object API

public void PlayTutorial(int step);

public void PlayTutorial(string name);

public void PlayTutorial();

public void PlayTutorial(int stageIndex, int tObjIndex);

public void StopTutorial(bool autoAdvance = true);

public void StopTutorial(int step, bool autoAdvance = true);

public void StopTutorial(string name, bool autoAdvance = true);

public void StopTutorial(int stageIndex, int tObjIndex, bool autoAdvance = true);

public void AdvanceTutorial();

public void HighlightTarget(HSVTutorialObject tObj, bool highlight);

public HSVTutorialObject GetCurrentTutorialObject(HSVTutorialStage stageObj, int step);

public HSVTutorialObject GetCurrentTutorialObject(HSVTutorialStage stageObj, string name);

public HSVTutorialObject GetCurrentTutorialObject(int stageIndex, int step);

public bool CheckAllTutorialObjectComplete();

Outline Effect Controller

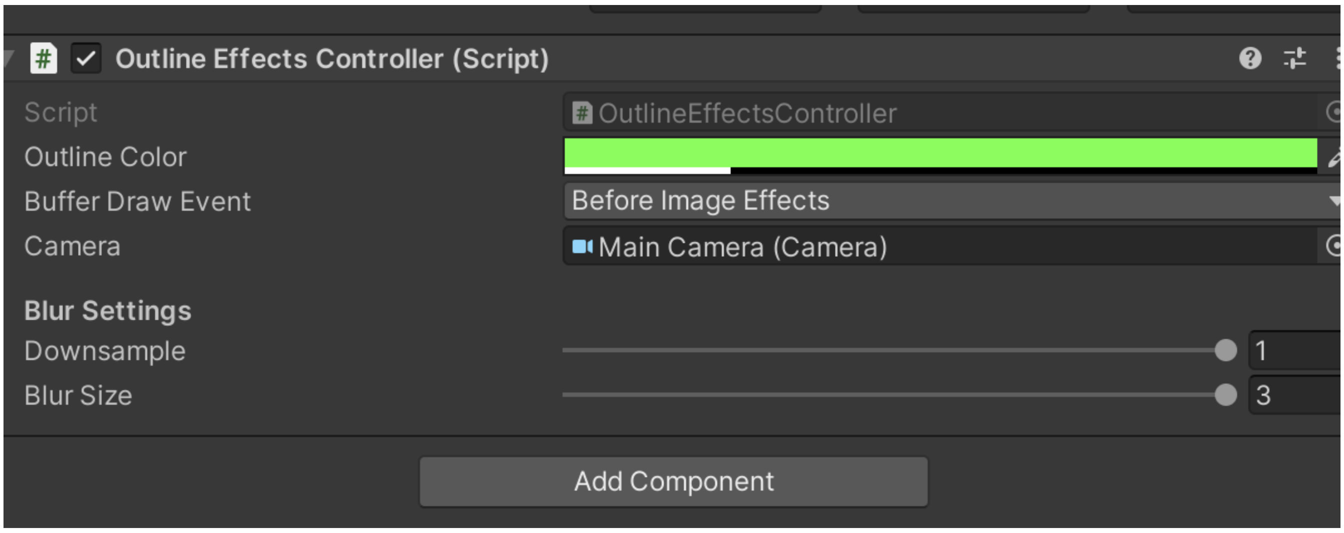
Outline effect is built-in highlight effect for Tutorial Factory. You could always use your own highlight effect and some other good assets, such as Highlight Plus from Kronnect.

> ⓘ Please use OutlineEffectControllerURP for the URP render pipeline. Remember to add Outline Render feature to the URP render pipeline asset.

Setup OutlineEffectController

1. Create a new GameObject and add OutlineEffectController component to the gameobject.
2.

2. Assign camera to Camera to have highlight effect.
3. Set Buffer Draw Event to 'Before Image Effects'.
4. Adjust Downsample and Blur Size if you want to have different look.
5. Adjust Color to your own choice. Remember to reduce alpha of color to show original object.
6. Now you should have outline effect by checking 'Highlight Target' on **# targets**.



Outline Effects Controller (Script)

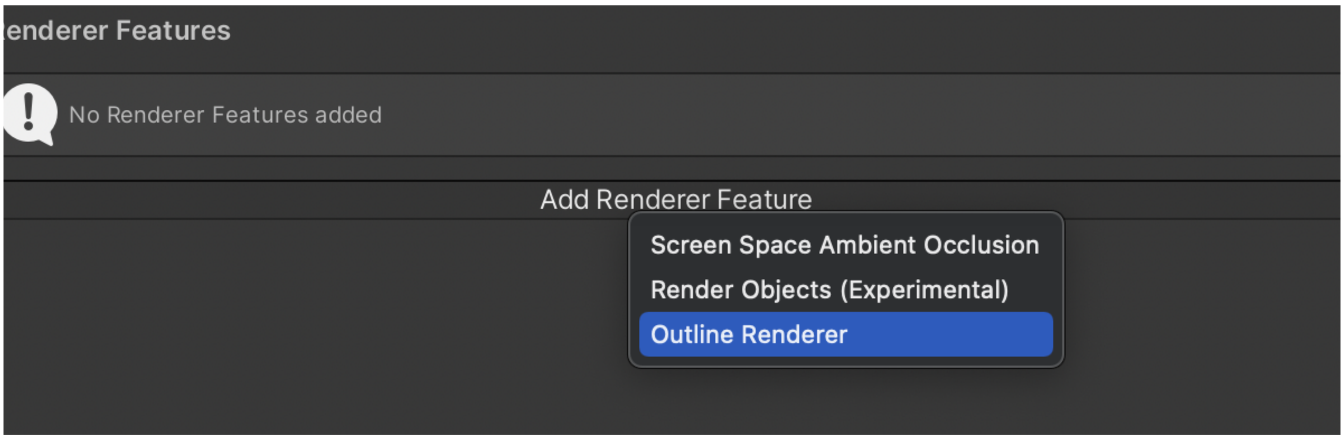
Script: OutlineEffectsController
Outline Color: 🟩
Buffer Draw Event: Before Image Effects
Camera: Main Camera (Camera)

Blur Settings
Downsample: 1
Blur Size: 3

Picture 1. Outline Effect Controller

Setup OutlineEffectController in URP

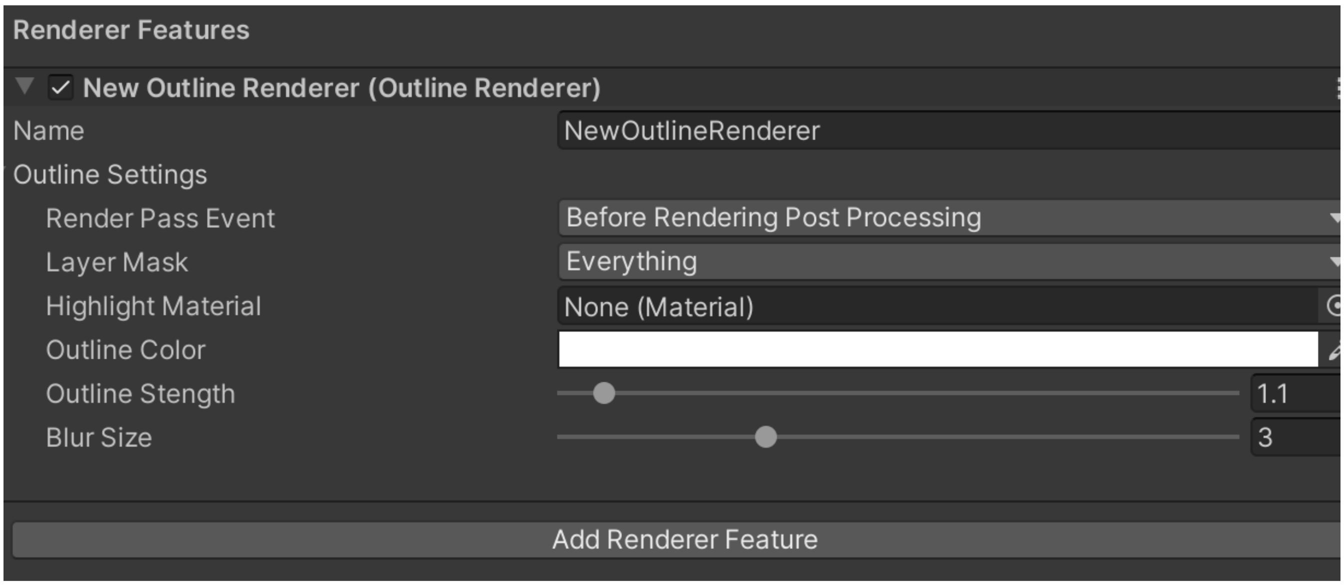
1. Unpack 'Tutorial Factory URP.unitypackage' in the package.
2. Add Outline Renderer Feature to the renderer asset data file(Picture 2).
3. Setup layer mask to include highlight objects' layers.
4. Assign Assets/HSVStudio Tutorial Factory/Materials/OutlineURP.mat material to the 'Highlight Material' field(Picture 4).
5. Change color to your own choice.
6. Adjust outline strength and blur size accordingly.
7. Now you should have Outline Effect in URP when 'Highlight Target' option is checked.



enderer Features

No Renderer Features added

Picture 2. Setup Outline Renderer URP


Renderer Features

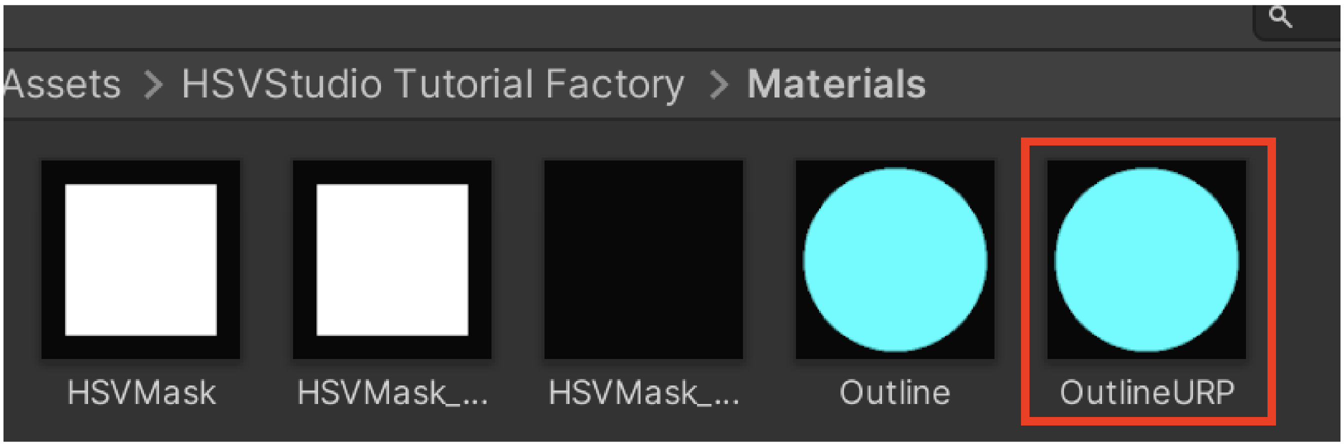
New Outline Renderer (Outline Renderer)

Name: NewOutlineRenderer

Outline Settings

- Render Pass Event: Before Rendering Post Processing
- Layer Mask: Everything
- Highlight Material: None (Material)
- Outline Color:
- Outline Stength: 1.1
- Blur Size: 3

Picture 3. Outline Renderer Setup


Assets > HSVStudio Tutorial Factory > Materials

Picture 4. Outline Material

Integrations

Tutorial Factory could be used in combination with other assets, such as DOTWeen for transition animation.

Please feel free to contact me for integration support.

Demo

Demo Scene

URP and Built-in demo scenes are included in the package. Please feel free to try them out. Demo scenes demonstrate how Tutorial Factory works.

Extras

Keyboard Shortcuts

In Editor Mode: Press 'q' for previewing Tutorial Objects UI. This would show how displaying modules look like on screen. However, animations of module would only work on PlayMode.