Ilumisoft

Game Action System

Documentation

Overview

The Game Action System is a Unity plugin that allows game developers to easily set up gameplay logic directly within the editor. The system is composed of three key components: Game Action Trigger, Game Action, and Game Action Task. These components work together to provide a flexible and powerful system for creating complex gameplay interactions.

Installation

To install the package, follow these steps:

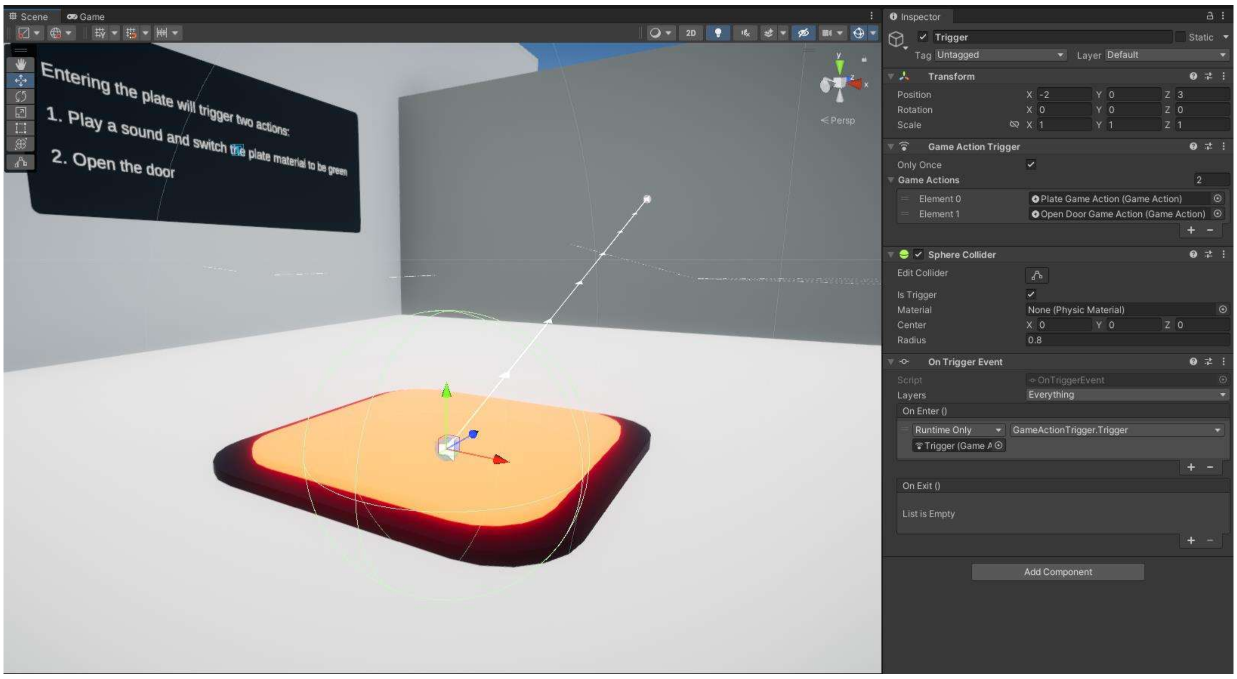
1. Add the plugin package from the Unity Asset Store to “My Assets”
2. Create or open a Unity project and download and install the plugin from the package manager
3. The plugin will be installed in your project and you will be able to access it through the Unity editor.

Components

The system is composed of three key components: Game Action Trigger, Game Action, and Game Action Task. These components work together to provide a flexible and powerful system for creating complex gameplay interactions.

Game Action Trigger

The Game Action Trigger is a component that allows you to trigger multiple Game Actions based on specific events. In the inspector, you can set whether the trigger should only fire once or if there should be a cooldown period before it can be triggered again. This allows you to create complex gameplay interactions that respond to player actions in real-time.



Entering the plate will trigger two actions:
1. Play a sound and switch the plate material to be green
2. Open the door

Trigger ✓ — Static
Tag: Untagged — Layer: Default

Transform

	X	Y	Z
Position	-2	0	3
Rotation	0	0	0
Scale	1	1	1

Game Action Trigger

Only Once: ✓
Game Actions: 2

Element 0	Plate Game Action (Game Action)
Element 1	Open Door Game Action (Game Action)

Sphere Collider ✓

Edit Collider	
Is Trigger	✓
Material	None (Physic Material)
Center	X 0, Y 0, Z 0
Radius	0.8

On Trigger Event

Script	OnTriggerEvent
Layers	Everything

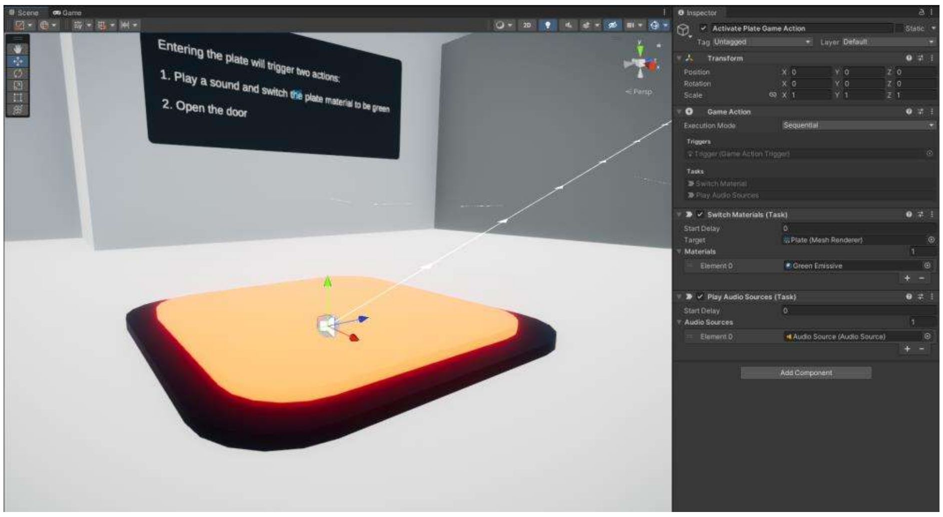
On Enter ()
Runtime Only — GameActionTrigger.Trigger
Trigger (Game A

On Exit ()
List is Empty

To add a Game Action Trigger to your scene, click ***Game Object->Game Action System->Game Action Trigger.*** You can drag and drop a Game Action to the Game Actions list to assign it to the trigger.

Game Action

The Game Action component specifies a specific gameplay action in your game. A Game Action performs a list of tasks, such as playing timelines, enabling game objects, or other custom tasks that you can define. In the inspector, you can specify whether the tasks should be executed in sequential or parallel mode. In sequential mode, tasks are executed one by one, with each task waiting until the previous task has been completed. In parallel mode, all tasks are executed simultaneously.



Entering the plate will trigger two actions:
1. Play a sound and switch the plate material to be green
2. Open the door

Activate Plate Game Action
Tag Untagged   Layer Default

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

Game Action
Execution Mode: Sequential
Triggers
Trigger (Game Action Trigger)
Tasks
Switch Material
Play Audio Sources

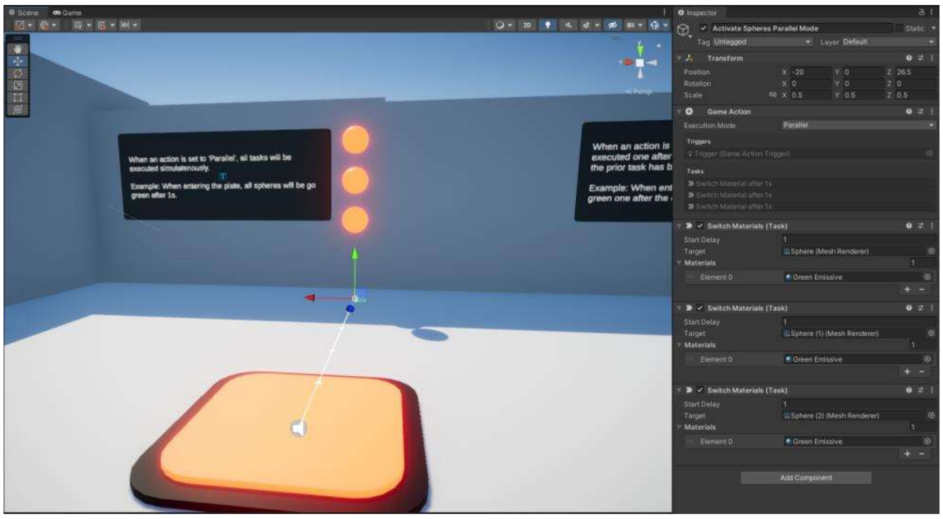
Switch Materials (Task)
Start Delay: 0
Target: Plate (Mesh Renderer)
Materials: 1
Element 0: Green Emissive

Play Audio Sources (Task)
Start Delay: 0
Audio Sources: 1
Element 0: Audio Source (Audio Source)

To add a Game Action to your scene, click ***Game Object->Game Action System->Game Action.***

Game Action Task

A Game Action Task is a specific task that can be performed by a Game Action. Each task can have a delay defined in the inspector that will be awaited before the task is executed. You can also create your own custom tasks by implementing the Game Action Task interface. This allows you to create highly customizable tasks that are specific to your game's unique requirements.


Scene
Game
2D
Persp
When an action is set to 'Parallel', all tasks will be executed simulatenously.
Example: When entering the plate, all spheres will be go green after 1s.
When an action is
executed one after
the prior task has b
Example: When ent
green one after the
Inspector
Add Component

```
Activate Spheres Parallel Mode                    Static
Tag Untagged              Layer Default
Transform
Position        X -20    Y 0     Z 26.5
Rotation        X 0      Y 0     Z 0
Scale           X 0.5    Y 0.5   Z 0.5
Game Action
Execution Mode  Parallel
Triggers
  Trigger (Game Action Trigger)
Tasks
  Switch Material after 1s
  Switch Material after 1s
  Switch Material after 1s
Switch Materials (Task)
Start Delay     1
Target          Sphere (Mesh Renderer)
Materials                                  1
  Element 0     Green Emissive
Switch Materials (Task)
Start Delay     1
Target          Sphere (1) (Mesh Renderer)
Materials                                  1
  Element 0     Green Emissive
Switch Materials (Task)
Start Delay     1
Target          Sphere (2) (Mesh Renderer)
Materials                                  1
  Element 0     Green Emissive
```

Three tasks with a delay of 1 second. Each task will switch the material of a sphere to green. As the Game Action is set to Parallel, all tasks will be executed after 1 second.

To add a task to a Game Action, select the Game Action, click ***Add Component->Game Action System->Tasks*** and select a task from the list.

Creating custom tasks

You can easily write your own tasks. This allows you to create highly customizable tasks that are specific to your game's unique requirements.

Example: Log Message Task

In this example we will write a simple task that will log a message to the console. Create a new Script called **LogMessageTask.cs** and paste the following content:

```
using Ilumisoft.GameActionSystem;

using UnityEngine;


/// <summary>

/// Logs the given message when being executed

/// </summary>

[AddComponentMenu("Game  Action System/Tasks/Debug/Log  Message (Task)")]

public class LogMessageTask : GameActionTask

{

    /// <summary>

    /// A message that can be defined in the inspector

    /// </summary>

    public string message = string.Empty;


    /// <summary>

    /// This will be invoked when the task is executed

    /// </summary>
    /// <returns></returns>
    protected override StatusCode OnExecute()
    {
        // Log the message
        Debug.Log(message);

        // As the task is completed, when return the completed status code
        return StatusCode.Completed;
    }
}
```

Example: Wait Until Key Pressed Task

You can also write tasks that are running over multile frames before they are completed. The following task will be running until a specific key is pressed. This could be used for example on a sequential Game Action to delay a timeline from being played until the player pressed a key.

```
using Ilumisoft.GameActionSystem;

using UnityEngine;


/// <summary>

/// Logs the given message when being executed

/// </summary>

[AddComponentMenu("Game  Action System/Tasks/Debug/Log  Message (Task)")]

public class LogMessageTask : GameActionTask

{

    /// <summary>

    /// A message that can be defined in the inspector

    /// </summary>

    public string message = string.Empty;


    /// <summary>

    /// This will be invoked when the task is executed

    /// </summary>

    /// <returns></returns>

    protected override StatusCode OnExecute()

    {

        // Log the message
        Debug.Log(message);


        // As the task is completed, when return the completed status code

        return StatusCode.Completed;

    }

}
```

Demo

The Sample scene provided with the package (Plugins->Ilumisoft->Game Action System->Demo->Demo) gives an overview of the provided features and options. It contains a simple sample scene showing different scenarios created using the system.

When you enter playmode, you can control a sample character and interacting with the environment by entering plates. The info boxes in the scene will provide you with additional information.

On the following pages you will find a description of the core components of the system.

Support

If you like the project, please take a minute and give us a rating in the Asset Store. This really helps us to create and improve our Unity Assets.

If you encounter any problems or errors, please contact us via email:

support@ilumisoft.de