Grid Controller

Version 1.x

Grid Controller is a grid-based first-person movement controller that's simple to set up yet robust and versatile. Its features include:

- Real-time or turn-based movement
- Smooth movement or instant snapping to grid positions
- Support for inclines, declines, and crouching
- Free look camera
- Minimap
- Surface system for footstep sounds and barrier bounce sounds
- Triggers and interaction with doors, levers, stairs, scene portals, etc.
- Input system independent: use with built-in input, Input System package, Rewired, etc.
- Audio system independent: use with built-in audio, etc.
- Complete, clean, and well-documented C# source code.

How To Get Help

We're here to help! If you get stuck or have any questions, please contact us any time at support@pixelcrushers.com or visit https://pixelcrushers.com.

We do our very best to reply to all emails within 24 hours. If you haven't received a reply within 24 hours, please check your spam folder.

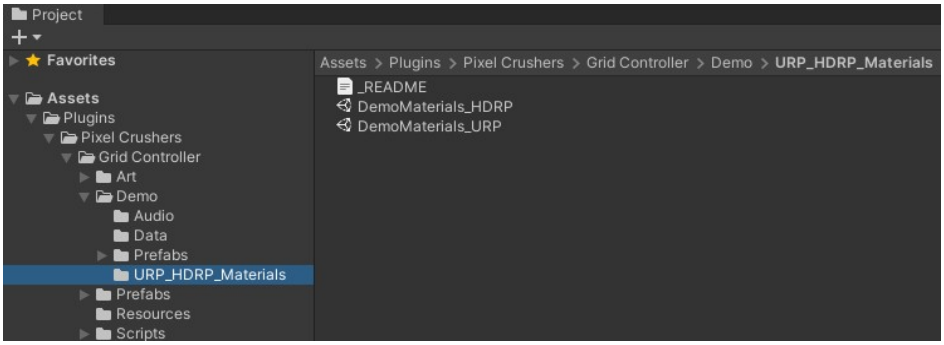
Grid Controller
Copyright © Pixel Crushers. All rights reserved.

Table of Contents

Quick Start

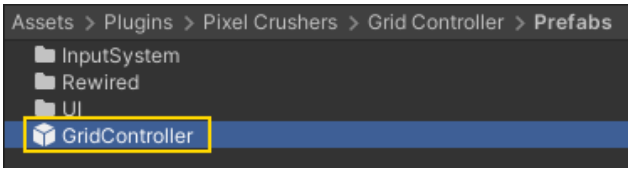
Demo Scene and URP/HDRP

The demo scene's materials are configured for Unity's built-in rendering pipeline. If your project uses URP or HDRP, before playing the demo import the corresponding unitypackage from the URP_HDRP_Materials subfolder.

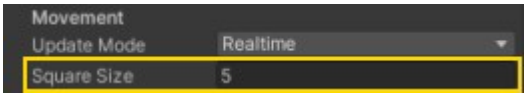


Use Grid Controller Prefab

The quickest way to set up your scene with Grid Controller is to drop the **GridController** prefab into your scene. This is your "player character" that will move around the map.

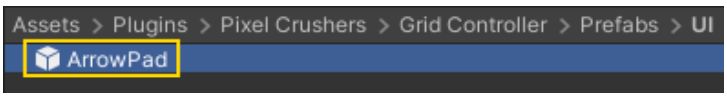


Set the **Grid Controller** component's **Square Size** to match the size of an individual square in your scene. For example, if a tile in your scene's map is 5x5 units, set Square Size to 5.



Click the **Grid Controller Input** component's **Check Input Settings** button.

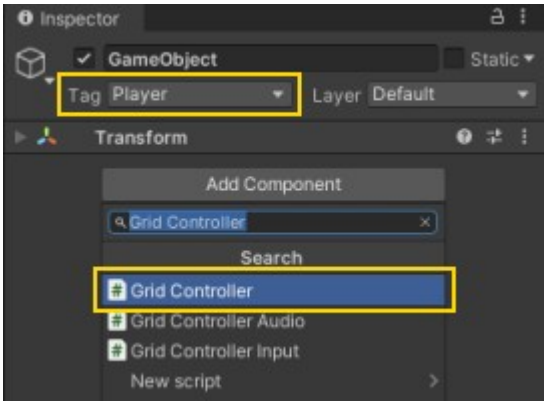
If you want to use onscreen UI buttons, add the **ArrowPad** prefab to your scene.



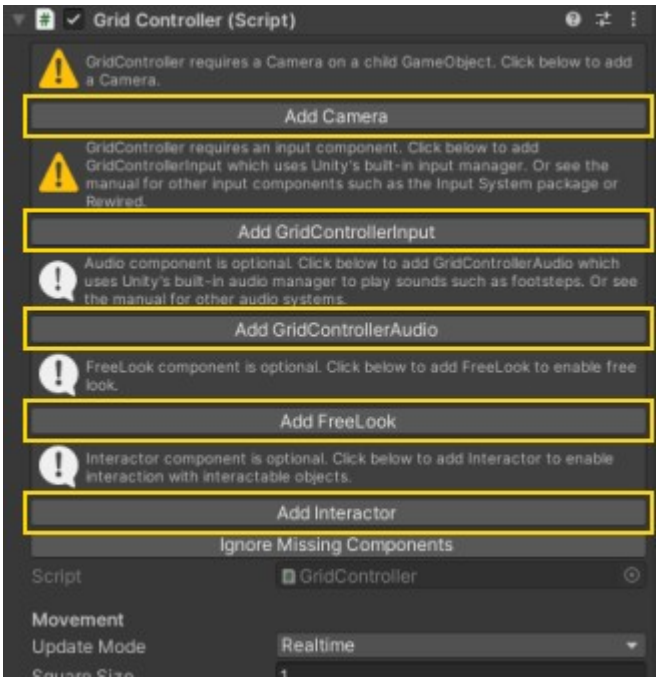
Manually Configure Grid Controller

If you prefer to manually configure a Grid Controller:

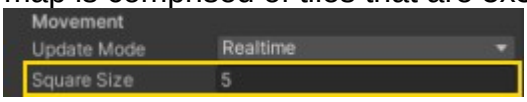
1. Add a **Grid Controller** component to your player GameObject. Grid Controller doesn't require a collider. Tag your GameObject as **Player**.



   Click the **Add** buttons to add the auxiliary components that Grid Controller uses. This will add components that use Unity's built-in input manager and audio manager. To use a different input or audio system such as Rewired or FMOD, see Input Systems and Audio Systems.

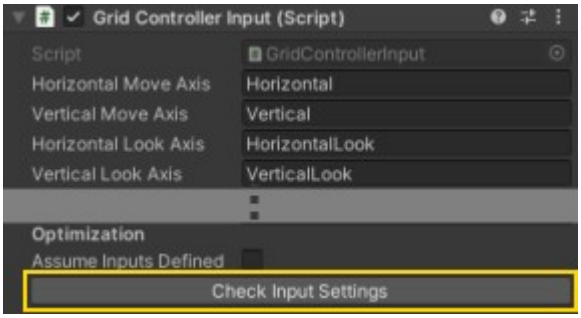


2. Set **Square Size** to size of a single square in your scene. For example, if your scene's map is comprised of tiles that are 5x5 units, set Square Size to 5.



3. If you want the controller to register trigger collisions (e.g., OnTriggerEnter), add a collider such as a Capsule Collider.

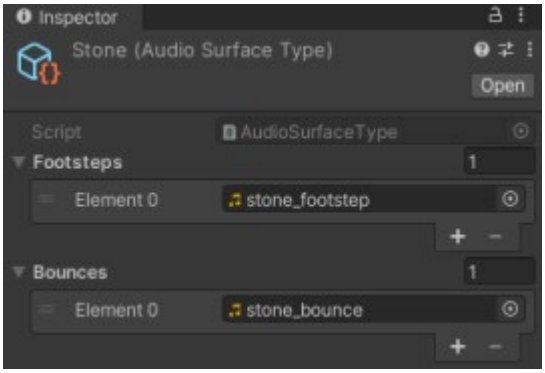
4. If you added the Grid Controller Input component, click **Check Input Settings** to add any missing input definitions to the project's Input Manager. You can inspect the input definitions with menu item *Edit > Project Settings > Input Manager*.



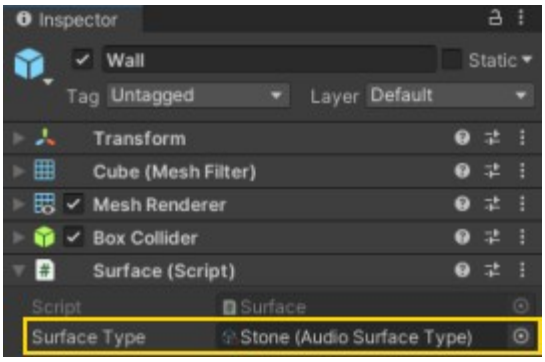
Once you've confirmed that all inputs are defined, you can tick **Assume Inputs Defined**. This will eliminate the runtime overhead of handling undefined inputs.

5. *Optional*: If you added the Grid Controller Audio component, assign Audio Sources. If you leave the Audio Sources unassigned, Grid Audio Controller will create them automatically at runtime. However, you can create them at design time if you want to assign audio mixer groups.

6. Optional: In the Project view, create **Audio Surface Type** assets (menu item *Assets > Create > Grid Controller > Audio Surface Type*) for each type of surface, and assign audio clips to play when the controller steps on or bounces against the surface.

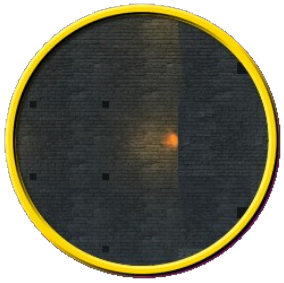


Then add **Surface** components to your scene's floors, walls, and other movement barriers such as doors, and assign the corresponding Audio Surface Type assets.

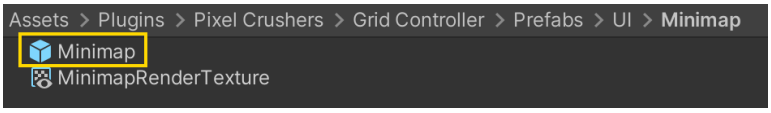


7. *Optional*: In the Project view, create **Audio Clip Info** assets for door opening and closing sounds, assign audio clips to them, and assign them to Door components on your doors. The demo's Prefabs folder contains preconfigured example doors.

Minimap



To add the optional minimap, add the Minimap prefab to your scene:



Since the minimap connects to the GridController GameObject, this should be a regular scene object, not a Don't Destroy On Load GameObject.

Components

Grid Controller

The **Grid Controller** component controls movement.

Properties

Property	Function
Update Mode	Operate in real-time or turn-based. In real-time mode, input is processed immediately. In turn-based mode, the most recent input is recorded, and it is only processed when you call the `ProcessTurn()` method.
Square Size	Size of a single square (e.g., tile) in the scene. Grid Controller moves one square at a time.
Grid Offset	Offset grid by this distance (e.g., if it's not perfectly aligned to (0,0)). Only used by Enforce Grid Position.
Enforce Grid Position	After moving, ensure controller is in center of square.
Rotation Duration	Turn left/right over this duration in seconds. Not used if Snap Movement is ticked.
Movement Duration	Move to new square over this duration in seconds. Not used if Snap Movement is ticked.
Snap Movement	Jump to new position instead of transitioning smoothly from current position to new position. Will still observe Movement Duration.
Enable Head Bob	Adjust camera to simulate head bob when moving. Ignored when Snap Movement is ticked.
Head Bob Focus Distance	When bobbing head, point camera to a fixed point this far ahead.
Head Bobs Per Square	Perform this many bobs with each movement into a new square.
Head Bob Height	Height of head bob.
Allow Crouch	Allow player to enter and exit crouch.
Crouch Scale Y	When crouched, move camera down to this proportion of original (standing) camera height.
Crouch Transition Duration	Smoothly transition camera from current height to new height over this duration in seconds. Not used if Snap Movement is ticked.
Crouching Footstep Volume	Reduce footstep volume to this proportion of original volume when crouching.
Boxcast Height	To determine if direction of movement is blocked, run boxcast centered this high from player's origin.

Boxcast Extents	Size of boxcast.
Wall Layer Mask	If boxcast hits object on this layer mask, register as hit with wall or other movement-blocking barrier.
Floor Layer Mask	Grid Controller raycasts down to detect floors on this layer mask to keep the controller grounded and identify surfaces for footstep sounds.
Bounce Amount	When hitting a wall, bounce in the direction of movement and back by this proportion of a full square size.
Bounce Backward Amount	When hitting a wall while moving backward, bounce in the direction of movement and back by this proportion of a full square size.

Events

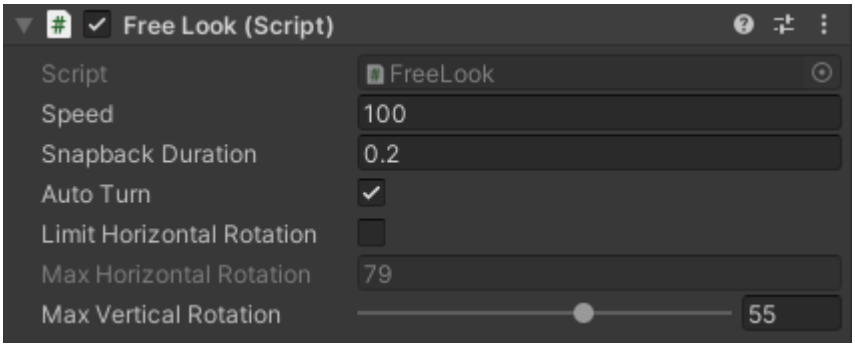
The **Grid Controller** component provides these C# events. You can also configure handlers for these events in the inspector by adding a **Grid Controller Events** component to the Grid Controller.

C# Event	Description
ExitingPosition	Invoked prior to leaving a square.
EnteredPosition	Invoked after entering a new square.
Turning	Invoked prior to rotating in place.
Turned	Invoked after rotating in place.
Strafed	Invoked after side-stepping into a new square.
Crouched	Invoked after entering the crouched state.
Uncrouched	Invoked after exiting the crouched state.
Blocked	Invoked when trying to move into a square that’s blocked (e.g., by a wall).

Free Look

The **Free Look** component allows the player to freely rotate the camera. This component is optional. If you don't want the player to free look, you don't have to add it.

If free looking beyond 80 degrees horizontally, Free Look will turn the Grid Controller's heading by 90 degrees unless Auto Turn is unticked.



Properties

Property	Function
Speed	Free look rotation sensitivity. Higher values rotate faster.
Snapback Duration	Duration in seconds to snap back to original orientation when exiting free look.
Auto Turn	Automatically rotate controller 90 degrees if looking more than 80 degrees to side.
Limit Horizontal Rotation	Restrict left/right rotation to Max Horizontal Rotation.
Max Horizontal Rotation	Max left/right free look angle. Only observed if Limit Horizontal Rotation is ticked.
Max Vertical Rotation	Max up/down free look angle.
Limit Horizontal Rotation	Use Max Horizontal Rotation.

Grid Controller Input

The **Grid Controller Input** component implements input using Unity's built-in input manager. It supports keyboard + mouse, joystick, and touch input.

If you want to use a different input system such as the Input System package or Rewired, see Input Systems.

Properties

Property	Function
Various Input Names	Names of input definitions in Unity's input manager. To automatically add inputs named in this component to the input manager, click **Check Input Settings**.
Recenter Input	When pressed during free look, snap back to the controller's original rotation.
Interact Input	Interactor component reads this input to activate interactables.
Input Repeat Speed	When an input axis is held, register presses at this frequency in

	seconds.
Input Dead Zone	Ignore axis values that are less than this threshold.
Interact Mouse Button	Interactor component checks this mouse button to activate interactables.
Free Look Mouse Button	Free Look component reads this input to enable free look.
Joystick Horizontal Axis	When pressed, switch to joystick mode. When mouse is used, switches to mouse mode.
Joystick Vertical Axis	When pressed, switch to joystick mode.
Assume Inputs Defined	Assume you've clicked **Check Input Settings** and inputs defined above are all defined in Input Manager. Eliminates overhead of checking for missing inputs.

Grid Controller Audio

The **Grid Controller Audio** component implements audio using Unity's built-in audio manager.

If you want to use a different audio system such as FMOD, see Audio Systems.

Properties

Property	Function
Footstep Audio Source	Audio Source to use for footstep sounds. If unassigned, Grid Audio Controller will create one at runtime.
Bounce Audio Source	Audio Source to use for bounce sounds when the controller tries to move against a barrier such as a wall. If unassigned, Grid Audio Controller will create one at runtime.
Door Audio Source	Audio Source to use for door open/close sounds. This should be on a child GameObject since Grid Audio Controller will move it to the position of the door when playing a sound. If unassigned, Grid Audio Controller will create an Audio Source on a child GameObject at runtime.

Interaction

Grid Controller includes an interaction system that you can use to interact with objects such as opening doors and pulling levers.

Interactor

The **Interactor** component responds to interaction input by raycasting for GameObjects with Interactable components in front of the controller or at the mouse position, depending on the type of input. If the raycast finds an Interactable, it calls its `Interact()` method.

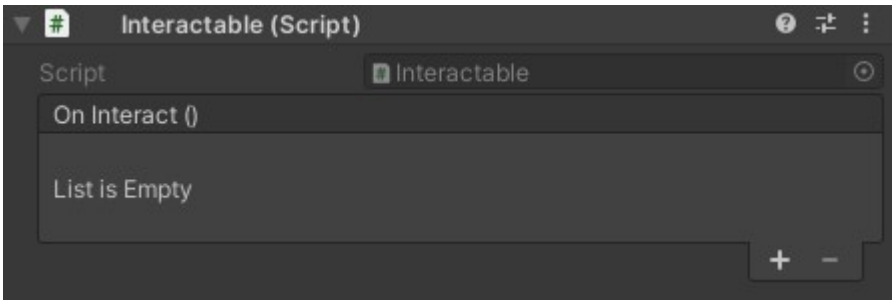
Properties

Property	Function
Interactable Layer Mask	Layer to check for Interactable objects.

Interactable

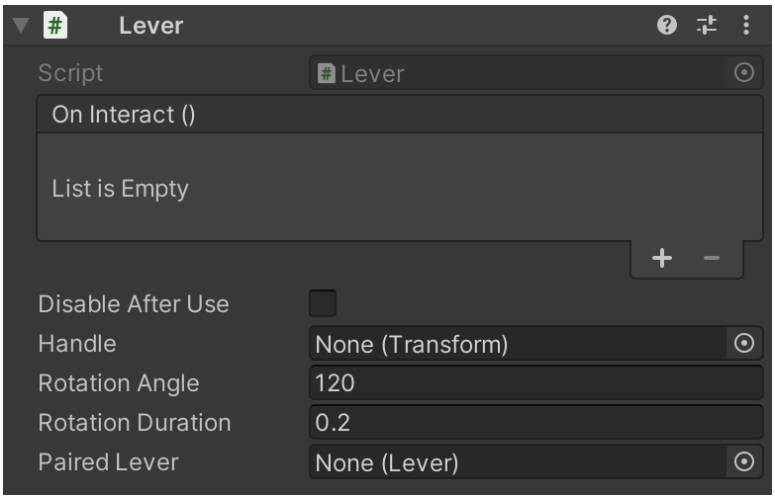
The **Interactable** component is the base class for interactable GameObjects. You can use it on its own (as the demo's InteractableDoor prefab does) or subclass it (as in the Lever class).

The Interactable class has an OnInteract() UnityEvent and an `Interacted` C# event.



Lever

The **Lever** component manages an interactable object with an animated handle. When activated, it rotates the handle in addition to invoking its interaction events.

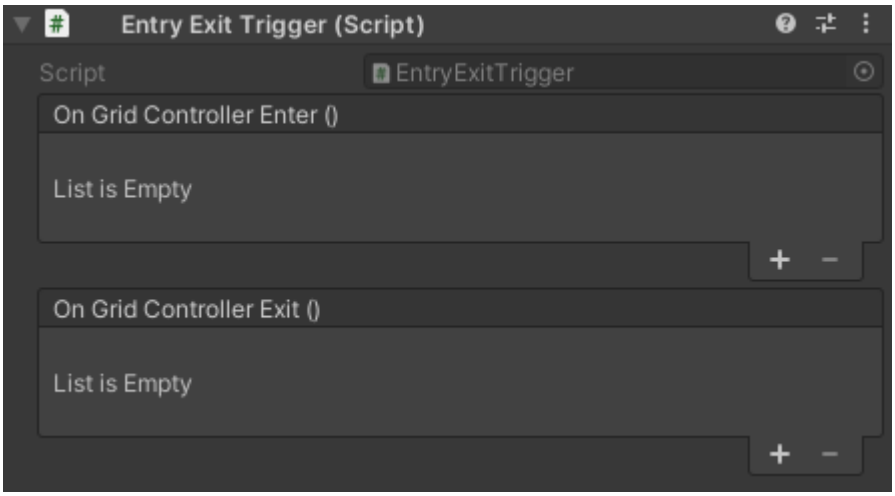


If you assign another lever to **Paired Lever**, the paired lever will also move when this lever moves. This is useful if you have connected levers on either side of a door.

Entry Exit Trigger

The **Entry Exit Trigger** component invokes events when the controller enters or exits its trigger collider. The controller must be tagged **Player** and have a Grid Controller component.

This component has UnityEvents and `Entered` and `Exited` C# events.

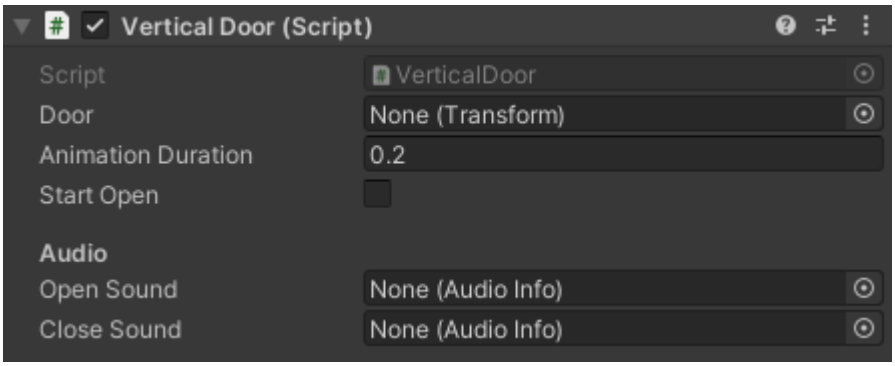


Slopes

Grid Controller detects floors by raycasting down from the top of the square to the distance specified by Floor Raycast Distance. A good distance is typically twice the Square Size. Grid Controllers allows traversal of inclines whose angles are at most Max Slope.

Vertical Door

The **Vertical Door** component controls a door or gate that rises into the ceiling when opened. To open the door, call the `Open()` method. To close it, call `Close()`. To alternate between open and closed, call `Toggle()`.

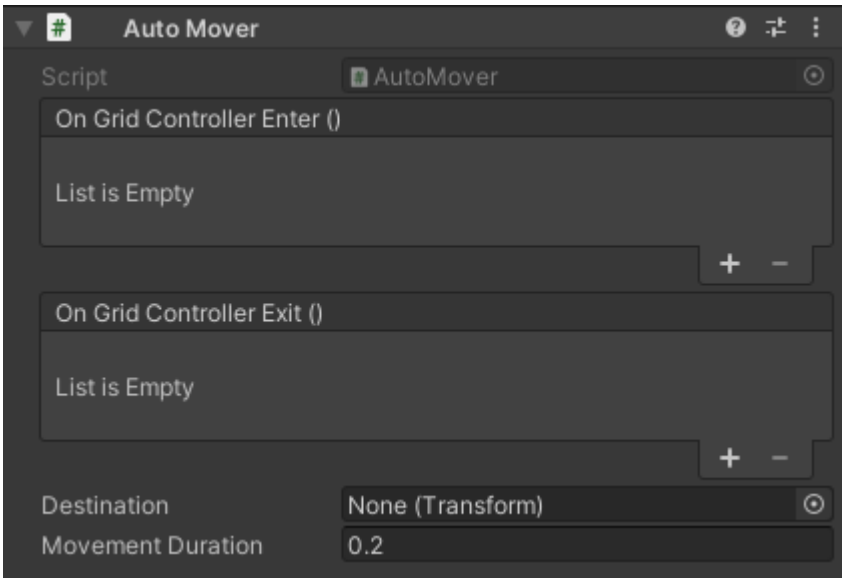


Properties

Property	Function
Door	Transform of child GameObject to move up into the ceiling.
Animation Duration	Duration in seconds to animate the door rising up or down.
Open & Close Sound	Audio assets to play when opening or closing. If using the Grid Controller Audio component, create and assign Audio Clip Info assets.

Auto Mover

The **Auto Mover** component automatically moves the player from a trigger collider to a destination point. The Auto Mover's GameObject should have a trigger collider and a rigidbody set to Kinematic. An Auto Mover can be used in combination with a Scene Portal to create stairs to a new scene.

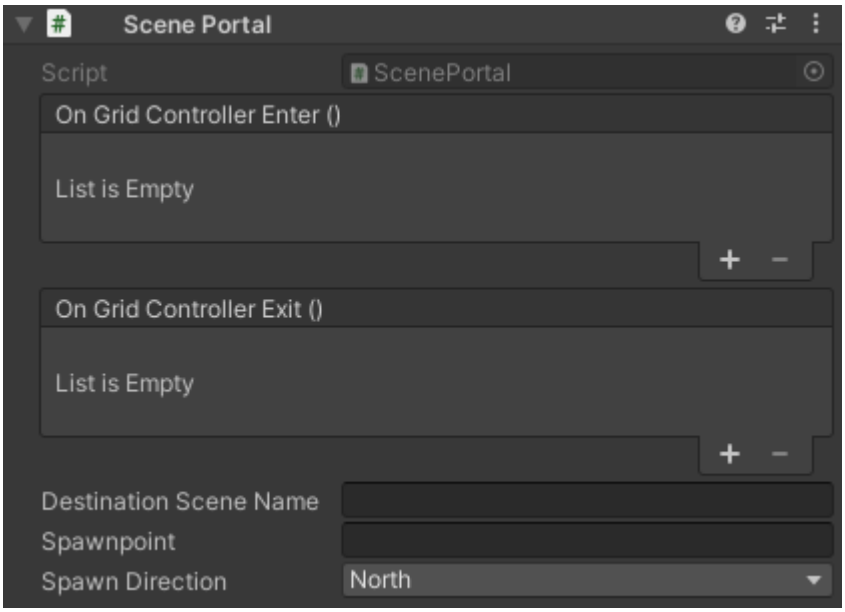


Properties

Property	Function
Destination	Position and rotation to move the player to.
Movement Duration	Duration in seconds to move the distance of one square.

Scene Portal

The **Scene Portal** component changes scenes when the player enters its trigger collider. The Scene Portal's GameObject should have a trigger collider and a rigidbody set to Kinematic. An Auto Mover can be used in combination with a Scene Portal to create stairs to a new scene.



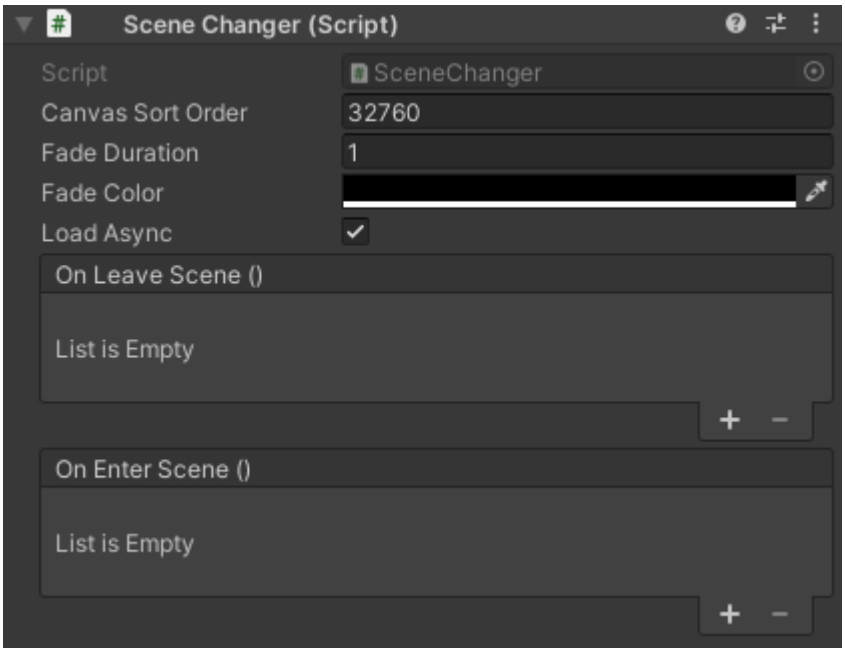
Properties

Property	Function
Destination Scene Name	Scene to load. Should be listed in build settings.
Spawnpoint	Name of GameObject in destination scene that player will match position of.
Spawn Direction	Direction that player will face in destination scene.

Scene Portals use the Scene Changer, described below.

Scene Changer

The **Scene Changer** is a *persistent singleton*. This means it survives scene changes and only allows one instance of itself to exist. The Scene Portal component will automatically create a Scene Changer at runtime if one doesn't already exist. However, if you want to customize the Scene Changer's settings, you can create an empty GameObject and add a Scene Changer component yourself.



Properties

Property	Function
Canvas Sort Order	Scene Changer fades out using a full-screen canvas. This specifies the canvas's sort order.
Fade Duration	Duration in seconds in which to fade in/out when changing scenes.
Fade Color	Color to fade to.
Load Async	Load using SceneManager.LoadSceneAsync. If unticked, uses SceneManager.LoadScene, which pauses the game while loading.

Input Systems

Grid Controller reads input from a generic C# interface named `IGridControllerInput`. Grid Controller ships with implementations of this interface for Unity's built-in input manager, the Input System package, and Rewired.

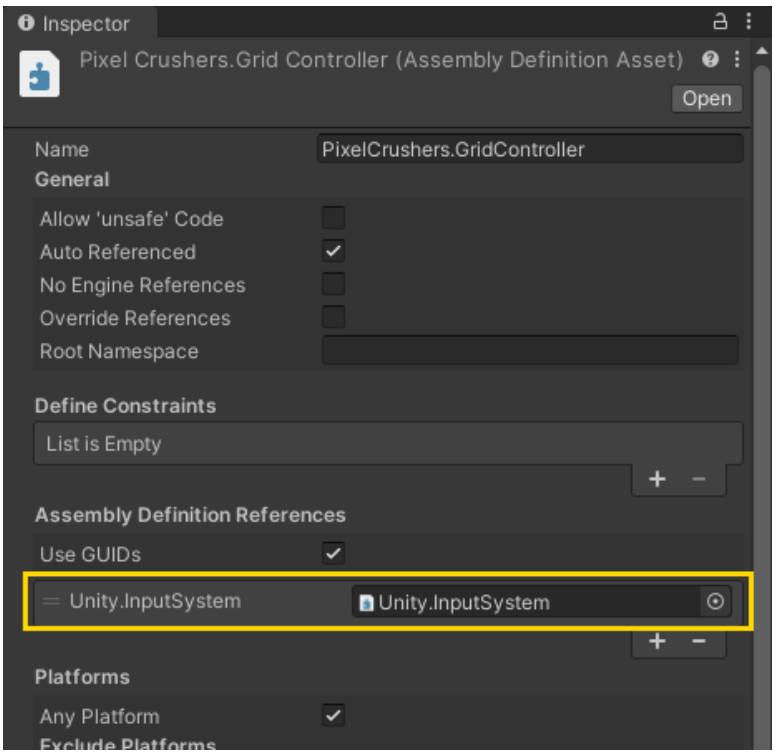
Input Manager

To use Unity's built-in input manager, add a Grid Controller Input component to the controller as described in the Quick Start and Components sections above.

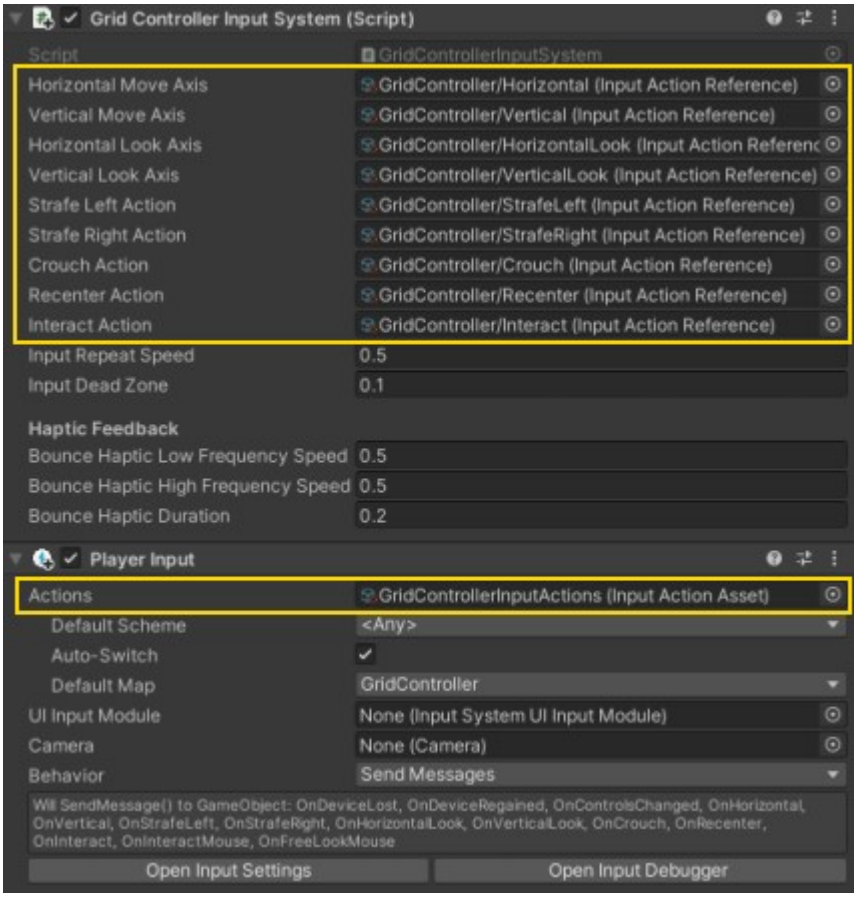
Input System Package

To use the **Input System** package:

1. Import the Input System package.
2. Inspect the PixelCrushers.GridController assembly definition asset, located in Grid Controller's Scripts folder. Add Unity.InputSystem to the Assembly Definition References and click Apply:



3. Remove the Grid Controller Input component from the Grid Controller GameObject, and add a Grid Controller Input System component in its place. This will also automatically add a Player Input component.



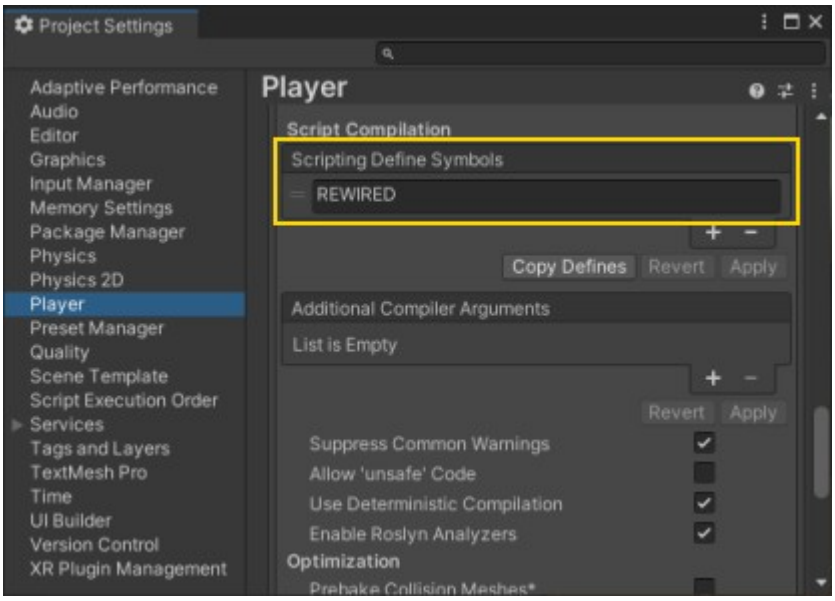
4. Assign the GridControllerInputActions asset to the Player Input component's **Actions** field. Assign the input actions to the Grid Controller Input System component as indicated in the image above. You can customize GridControllerInputActions' configuration. The only restriction is that the gamepad control scheme must be named `Gamepad`.

   Reminder: Use "`Gamepad`" as the name of the gamepad control scheme.

Rewired

To use Guavaman Enterprises' **Rewired**:

1. Import Rewired.
2. Select menu item *Edit* > *Project Settings*. In **Player > Other Settings**, add `REWIRED` to **Scripting Define Symbols**.



3. Remove the Grid Controller Input component from the Grid Controller GameObject, and add a **Grid Controller Rewired Input** component in its place.
4. Add the **GridControllerRewiredInputManager** prefab to your scene, or assign it to a Rewired Initializer component in your scene.

Other Input Systems

To use any other input system, implement `IGridControllerInput`.

Audio Systems

Grid Controller plays audio using a generic C# interface named `IGridControllerAudio`. Grid Controller ships with an implementation of this interface for Unity's built-in audio manager.

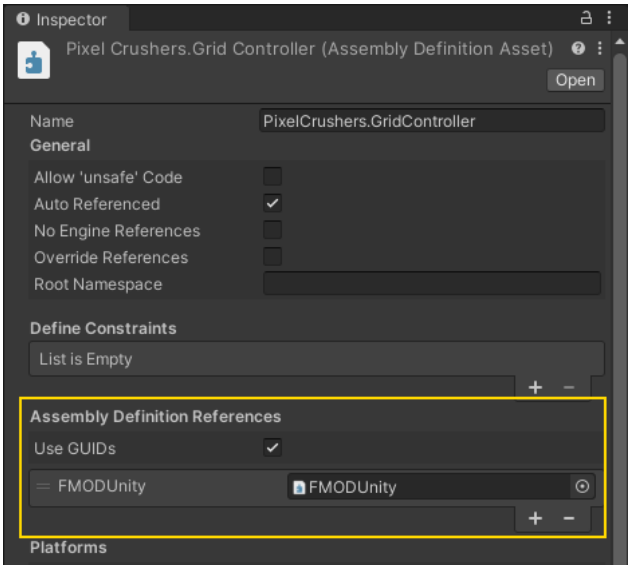
Audio Manager

To use Unity's built-in audio manager, add a Grid Controller Audio component to the controller as described in the Quick Start and Components sections above.

FMOD

To use Firelight Technologies' **FMOD**:

1. Import and configure FMOD for Unity.
2. Select menu item *Edit > Project Settings*. In **Player > Other Settings**, add `FMOD` to **Scripting Define Symbols**.
3. Inspect the PixelCrushers.GridController assembly definition file. Add FMODUnity to the **Assembly Definition References** and click **Apply**:



4. Remove the Grid Controller Audio component from the Grid Controller GameObject, and add a **Grid Controller FMOD Audio** component in its place. Assign your FMOD events to the component's properties.

5. Optional: Right-click in the Project view and select **Create > Grid Controller > FMOD Surface Type** to create surface types that play FMOD events when stepped on or run into. Then assign these surface types to your scene's Surfaces.

Other Audio Systems

To use any other input system, such as Wwise, implement `IGridControllerAudio`.

Scripting

Grid Controller comes with complete, documented source code.

You can access the scripting API online here:

Grid Controller Scripting Reference