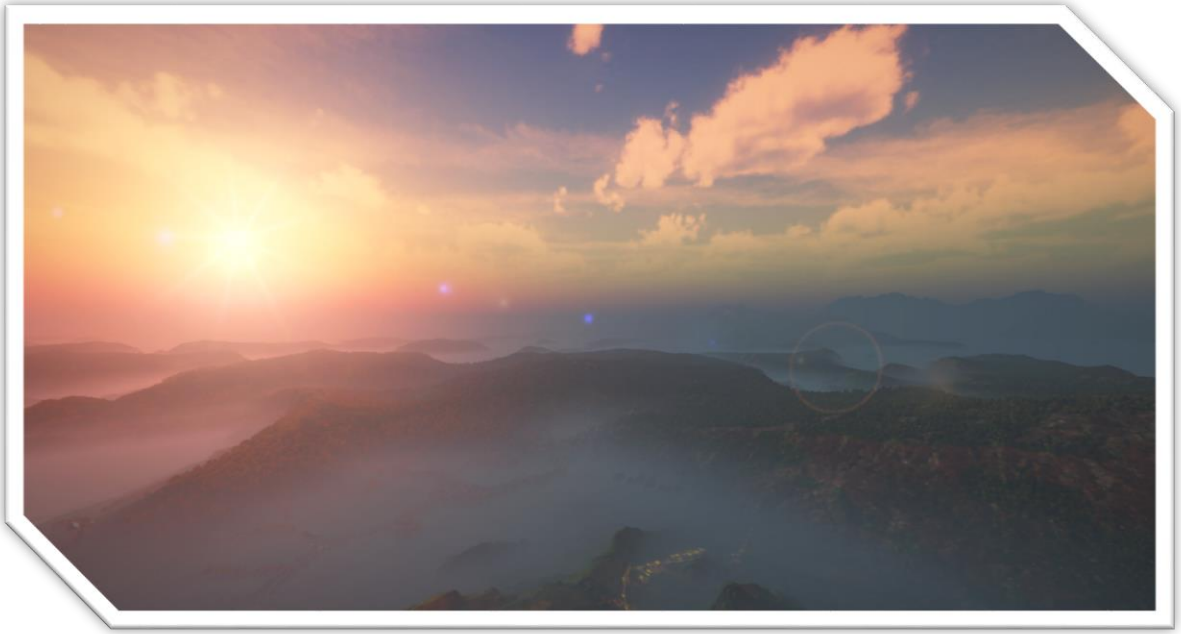


Azure[Sky]

Dynamic Skybox

Document Version 6.0.1

English is not my native language, so I apologize for possible grammar errors.

This documentation will not address each property one by one because they are many and the names are self-explanatory. You can also hover the mouse over each property that will show a tooltip about the functionality of that property.

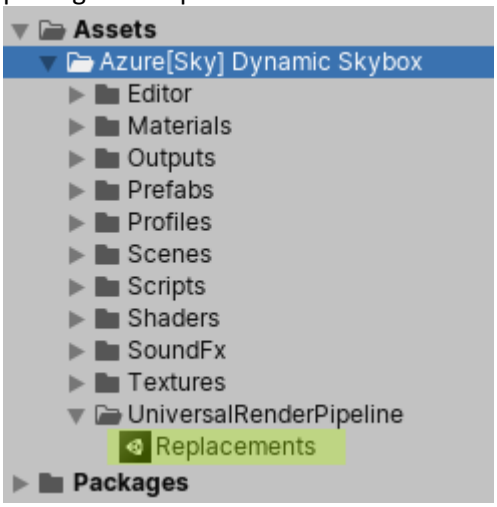
First Steps

- It is always a good practice to save a backup of your project or create an empty project to test the asset before importing it into the main project.

- The Anti-Aliasing somewhat affects (blur) the depth buffer texture and consequently generates artifacts when using the fog scattering effect. The fog scattering effect is depth-dependent, therefore, it is recommended to disable anti-aliasing in the project settings and use the AA from the post processing stack system instead.

- Azure works in both color spaces, but it is recommended to switch the project to the linear color space, because all sample scenes have been set up in the linear color space and everything in Unity looks better when using the linear color space.

- If you are using the **Universal Render Pipeline**, you will need to import the "**Replacements.unitypackage**" located in the "**UniversalRenderPipeline**" folder. This sub-package will replace some files needed for the Azure compatibility with URP.



Getting Started

- To start using Azure, all you have to do is drag the **Azure Sky Controller** prefab into your scene, the skybox material should be automatically replaced by the Azure sky material.

- Remove any directional light from the scene because the prefab already comes with a directional light attached to it.

- If you are using the Standard Render Pipeline, you can add the fog scattering effect to any camera by dragging the **AzureSkyFogScattering.cs** script to the Inspector or by the menu: *Component>Azure[Sky]>Azure Fog Scattering.*

- The fog scattering distance is set by default to 2750 meters, you must adjust the fog distance *(at the "Fog Scattering" tab of each day profile)* to a value that best suits your needs.

- If you are using the Universal Render Pipeline, follow the instructions below to enable the fog scattering effect on your scene. In the Universal Render Pipeline, the fog effect is a Renderer Feature.

    - Add the **Azure Fog Scattering Feature** to one or more of your renderer data assets.

    - Select the camera you want enable the fog scattering effect and in the "**Renderer**" option choose the renderer data asset where you added the fog renderer feature. **Note:** If the renderer data asset you added the fog renderer feature is not available for selection on your camera renderer option, then you need to add this renderer data asset to the "**Renderer List**" of the **Scriptable Renderer Pipeline** in use by your project.

    - Also in the camera component set the "**Depth Texture**" option to "**On**".

The sky controller prefab consists of the following components:

- **Azure Sky Controller:** This component manages the overall system, is also responsible for taking care of some important features such as weather transitions, outputs, options, simple events and more.

- **Azure Time Controller:** This component is responsible for all calculations involving the date and time. It also controls the directions of the sun, moon and directional light.

- **Azure Event Controller:** This component manages the powerful event system of Azure. With the event system you can fire UnityEvents with a pre-set date and time and use it to control just about everything on your game that is time/date dependent. *You can safe remove this component for save performance if you do not intend to use this feature.*

- **Azure Reflection Controller:** This component controls the reflection probe included by default in the prefab. Note that real-time global illumination and reflections drastically affect performance. *You can safe remove this component and delete the reflection probe game object from prefab for save performance if you do not intend to use this feature.*

- **Azure Effects Controller:** This component manages all sounds, particles, wind and thunder effects available in the asset. *You can safe remove this component and delete the "Effects" game object from prefab for save performance if you do not intend to use this feature.*

Azure Sky Controller Component

References: On the References tab are attached all materials, transforms, shaders and other resources used by the sky system, there is not much to do on this tab.

Profiles: The Profiles tab is responsible for all things regarding the use of the day profiles such as global weather, local weather zones, and default day settings.

- **Default Day Profiles:** A profile list were you can add day profiles to work as a "regular calendar day" *(the default weather for that particular day)*.
    - A random profile from this list will be set to work as a default day profile every time a new day starts.
    - The game will start always using the first profile from this list.
    - If you want some date to start with a specific day profile, you can setup an Event Action to replace the current profile in use by the new profile just calling the public method: SetNewDayProfile(AzureSkyProfile profile).

- **Global Weather Profiles:** A list you can add as many day profiles as you want, you should use it to perform the global weather. Next to each profile in the list has a field to set the transition time in seconds for that profile and a button for quickly test in editor. To change the global weather by script, just call the public method: SetNewWeatherProfile(int index)
  Note: The global weathers are overrides by the local weather zones.

- **Local Weather Zones:** After creating the weather zones in the scene, you must add them to this list and arrange them according to the priority. For performance and organization reasons, the sky manager will process only the weather zones contained in the list.

    - **Trigger:** The Transform used to perform the local weather zones blends. Set to null if you want to disable the weather zones feature to save performance.

Events: The Events tab has 3 simple events that should be very useful in a sky system:

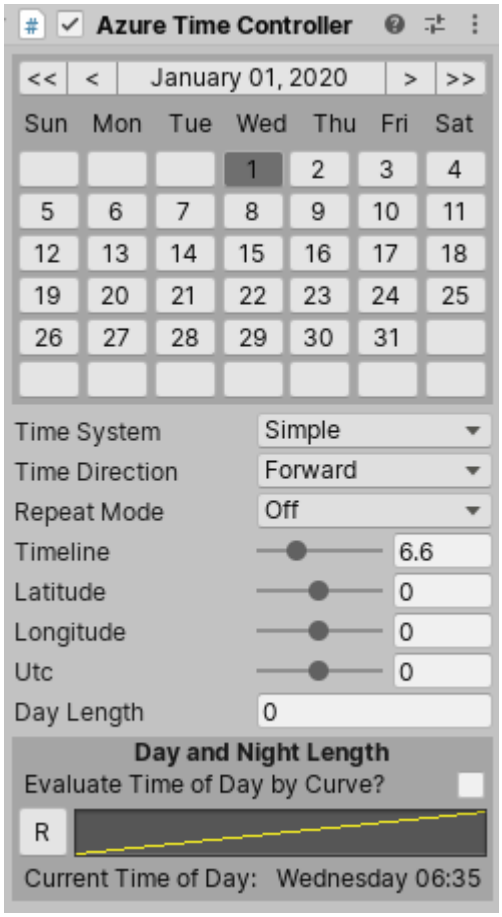
- OnMinuteChange(): Invoked when the minute changes.
- OnHourChange(): Invoked when the hour changes.
- OnDayChange(): Invoked when the day changes to the next day at midnight.

Options: On the Options tab, you can set things like the clouds mode, enable the day transition at midnight, set the position and color of the starfield cubemap.

If you set the “Day Transition Time” to a value different than zero, the sky system will automatically perform the day transition from the current day to the next when the day changes at midnight. Note that you will see the transition running only if the default weather is in use.

Outputs: On the Outputs tab, you can create extras properties to be set in each profile and extends the sky system functionalities. You can create an extra property to control for example the snow cover of your favorite terrain shader, this way you can set a different coverage value in each day profile. The sky system will perform all blends and weather transitions for this property and you will be able to access the resulting value to use and control all sorting of things in your game, as in this example send the output value to control the snow cover of your terrain shader.

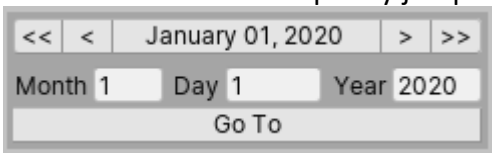
Azure Time Controller Component



- You can navigate through the calendar buttons to change the date.

- The date you set in the Inspector will be the date the sky system will start when you play the game.

- By clicking on the middle header button that displays the current date will show a hidden menu useful to quickly jump to a custom date.

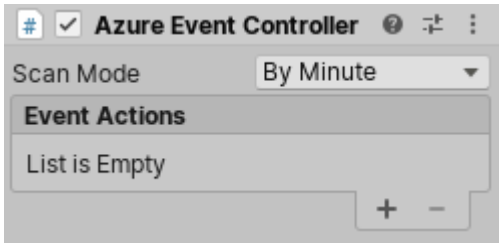


- Changing the time mode to "Realistic" will accurately set the position of the sun and moon in the sky based on the time, date and location *(latitude and longitude)*.

- Day Length is the day cycle duration in “minutes”. Set to zero if you want the time to be static.

- The “Current Time of Day” below the “Day and Night Length” curve, displays the timeline converted to “hours and minutes”. This is the time taken into account by the Event System.

Azure Event Controller Component



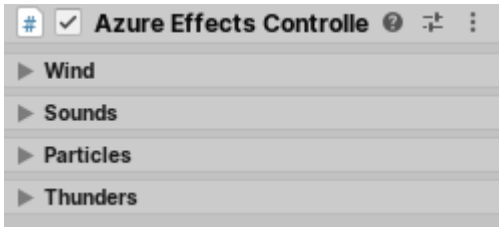
With **Azure Event System**, you can create as many "Event Actions" as you want to control all sorts of things in your game. You can use the event system to create or delete game objects at runtime, call methods in other scripts, and an infinity of things. The Event System is completely integrated with the time of day and Unity's Event System

- **Scan Mode:** As the Event Actions are performed according to the date and time, there is no need to waste performance by checking each event every frame, instead, this is done only when the time of day changes. So the scan mode sets when the sky manager will scan the event list to check if it needs to invoke some Event Action.

- **Event Action:** The list with all event actions.

- You can force the event system to ignore dates or times by setting the value -1 to the field you want the scanner to disregard when checking the event list. For example, you can set a real value only for the time fields and set -1 to date fields, this will cause the event to be called every day at that time.

Azure Effects Controller Component



Wind: On the Wind tab, there is a reference field to attach a Wind Zone and a float field to set the wind zone multiplier.

Sounds: On the Sounds tab, there is reference fields to attach or change all the sounds effects of the weather system.

Particles: On the Particles tab, there is refences of all the particles and materials used by the weather system. The most important thing here is the "**Follow Target**" field, here you must attach the transformation you want the particles to follow in the scene, usually the main camera is used here.

Thunders: On the Thunder tab, there is a list with 3 premade thunder effects and you can add more if you need. It is likely that the only thing you will need to change here is the position that each thunder will be instantiated in the scene.

To instantiate the thunder in the scene, simply press the "Test" button when using the Unity Editor or by calling the public method: InstantiateThunderEffect(int index) and pass as index parameter the element number of the thunder on the list.

Note:

- The instance of each thunder is automatically destroyed when the sound effect ends.
- The first premade thunder effect on the list is just the sound fx and the lightning, you can't see the "thunderbolts".

Creating a Day Profile

You can create a day profile by clicking on the menu: *Create>Azure[Sky] Dynamic Skybox>New Sky Profile*. This way you will create a profile without the proper configuration with all colors and gradients set to white. The best way is to select the "_DefaultSky" profile and click on the keyboard shortcut "Ctrl + D" to duplicate the profile and starting the new customization from there.

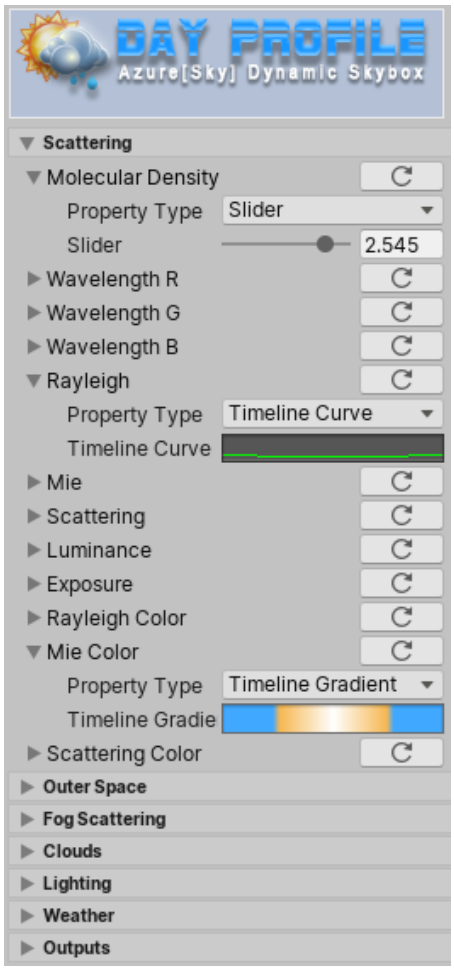
- To view your profile changes in real time while you customize it, the best way is to temporarily add the new profile to the first element of the *Default Day Profiles* list on the "Profile" tab of the Azure Sky Controller component.

- You can create as many profiles as you want.

- The profiles can be shared with the team members.

Customizing Your Day Profile

The customization of the profiles is also very simple and has no secrets, you just need to move some sliders, adjust curves and gradients.



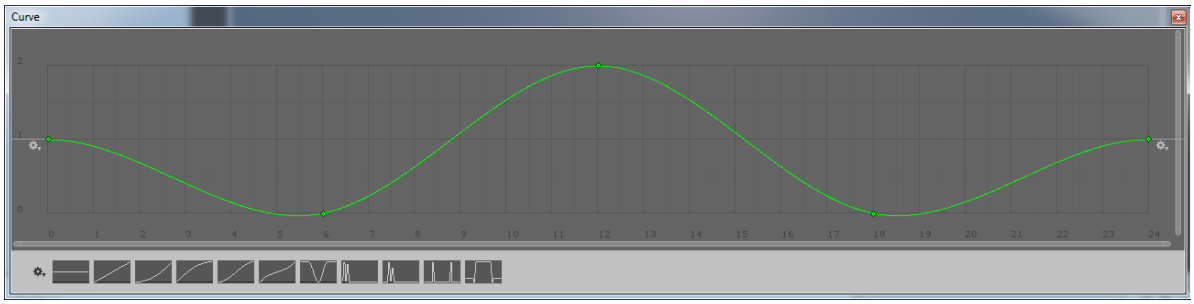
- You can fold or unfold each property to show the content and inside each property there is a popup to select the “Property Type”. In most cases, you will need to use only a slider or color field, but on some occasions, you will want the property to have different values depending on the time of day, so you can change the “Property Type” to curve or gradient and set different values for each time of day.

- There is also a button to reset the property.

How Do The Curves Work?

Curves allow each property to have dynamic values according to the time of day.

Timeline Curves: The customization of this curve is based on the timeline. When the time of day changes, the property value also dynamically changes to the value you set on the curve for that time of day. See in the screenshot below how the value of the property varies between 0 and 2 along the 24-hour day cycle.



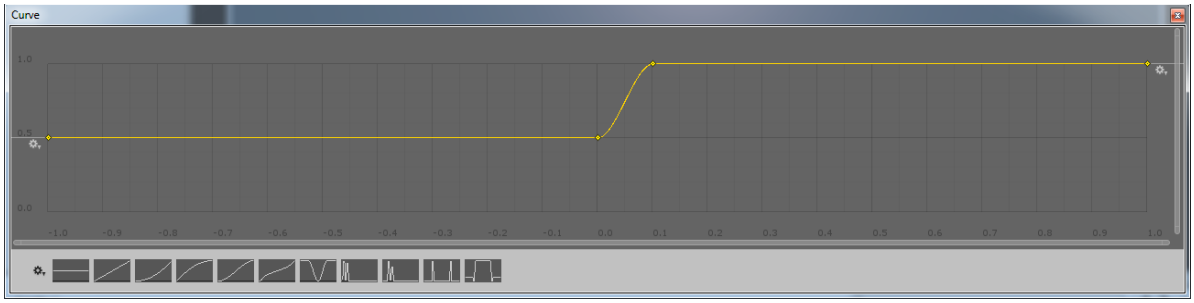
Sun Elevation Curves: The customization of this curve is based on the sun elevation in the sky. Let's suppose you customize a property using the curve mode based on the timeline and set, for example, the intensity of the directional light near zero at 20 o'clock taking into account that it is already night. If you are using realistic time mode, depending on the location and date, in some places in the world it can still be daytime at 8pm making the curve customization out of sync. In this case the curve should be customized based on the elevation of the sun in the sky, if the sun is above the horizon line, then it is daytime, otherwise, it is night. This curve type allows you to customize dynamic values depending on how high the sun is in the sky.

When the curve time is -1, it means that the sun is below the horizon line at 270°, in other words, it must be midnight...

When the curve time is 0, it means that the sun is at the horizon line at 0°, it must be sunset or sunrise.

When the curve time is 1, it means that the sun is above the horizon line at 90°, in other words, it must be noon.

See in the screenshot below how the curve is set to 1.0 at daytime and is set to 0.5 at night based on the sun elevation.

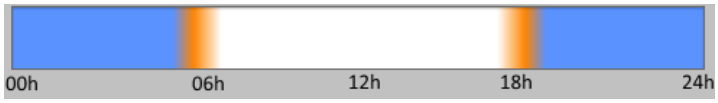


Moon Elevation Curves: The same as the sun elevation curves, but in this case it is based on the moon elevation in the sky.

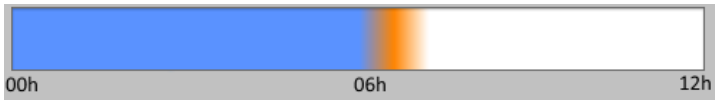
How Do The Gradients Work?

Gradients allow each color property to have dynamic values according to the time of day. The gradients work in the same way as the curves, but different from the curves it is not possible to visualize the time, but there is a trick that can help when customizing a gradient.

When the gradient is based on the timeline, the left side is referent to the 00h, the center is referent to the 12h and the right side is referent to the 24h. You can use the image below as a time reference.



When the gradient is based on the sun/moon elevation, the left side is referent to the 00h, the center is referent to the 06h and the right side is referent to the 12h. You can use the image below as a time reference.

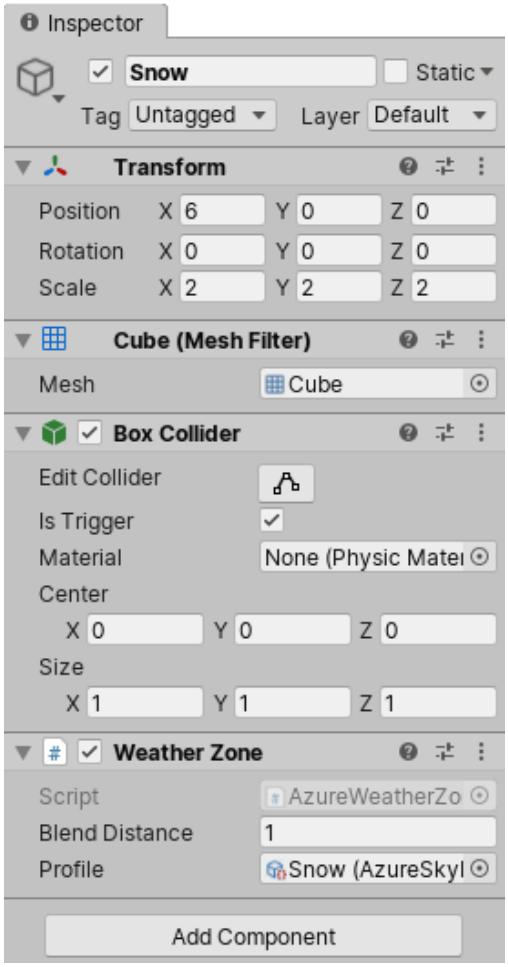


Weather Zones

Create a weather zone is very simple:

- Create an empty game object.
- Add a collider to it and check the “Is Trigger” option. It supports box collider, sphere collider, and mesh collider. Convex mesh is not supported.
- Add the weather zone component to it by dragging the **AzureSkyWeatherZone.cs** to the Inspector or by the menu: *Component>Azure[Sky]>Weather Zone*.

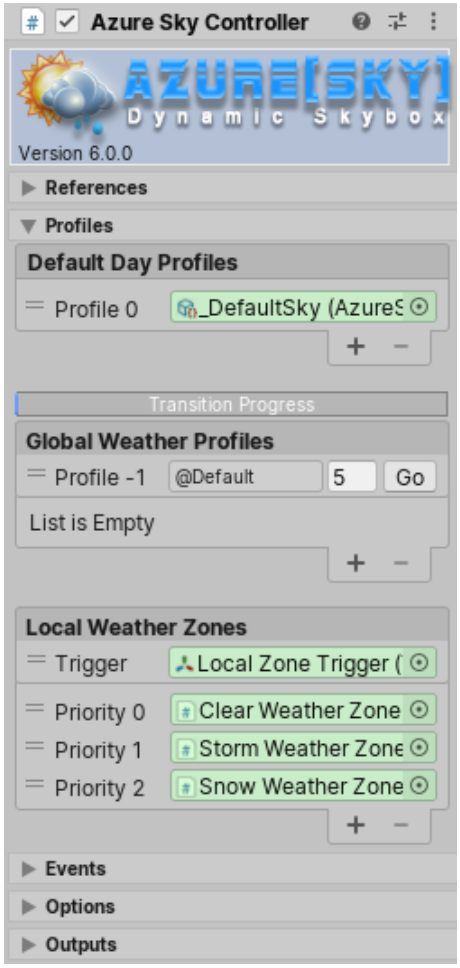
The Inspector of the game object should look like this.



Each weather zone component contains a field to place a day profile with the sky settings for that zone and also the option to set the blend distance.

You can now set the scale and location of the weather zone in your scene and also play with the “Blend Distance“ to find the best transition for that local weather zone.

It is also necessary to add this new weather zone to the "**Local Weather Zones**" list on the "**Profile**" tab of the “**Azure Sky Controller**” component and arrange according to the priority.



For performance, all the weather zones in the scene that is not attached in the “Local Weather Zones“ list will be ignored by the system.

Add the trigger transform to the “**Local Zone Trigger**”, usually the camera or player is used as a trigger. Setting the “local Zone Trigger” to null will disable the local weather zones (the global weather will still work).

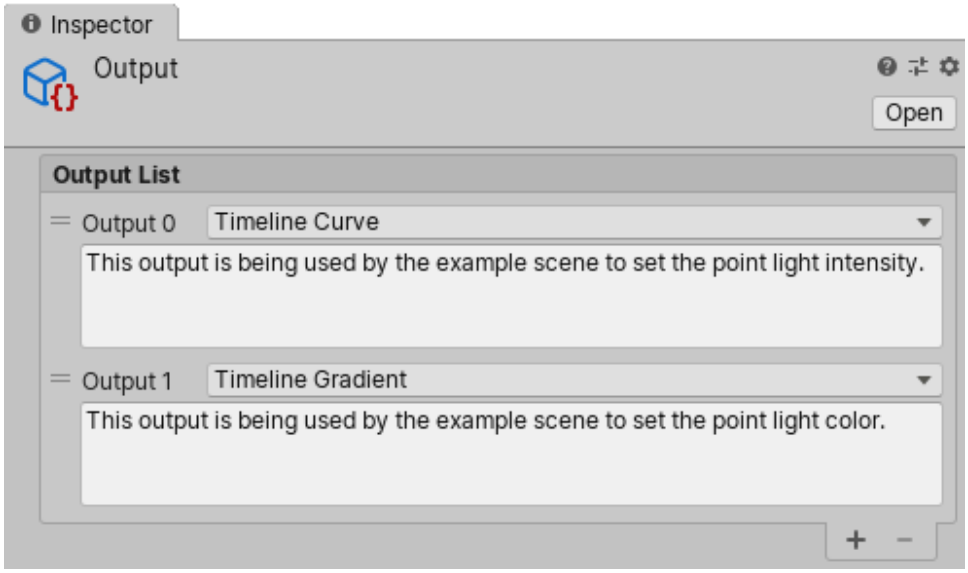
Now the weather zone system should already be working, when moving the trigger into the zone collider, the sky settings should automatically change to the setting defined in that zone.

Output System

The output system is a very useful feature that extends the sky system functionalities. For example: Do you need an extra property to control the wetness of your favorite terrain shader? No problem, just create a slider output and set the appropriate value in each profile, eg: In the rain profile, you set the value of this output to the maximum, in the sunny day profile you reduce the output value to zero. The sky manager will perform all blends and weather transitions to this output and store the result in a list where you can access with a simple script and send it to your terrain shader.

The output system is now a ScriptableObject and you can create as many output profiles as you want by the menu: *Create>Azure[Sky] Dynamic Skybox>New Output Profile*. With output profiles you can add in Azure extra properties to control anything in your game, such as making your favorite third party assets work with Azure.

In this example I created an output profile and added two extra properties:

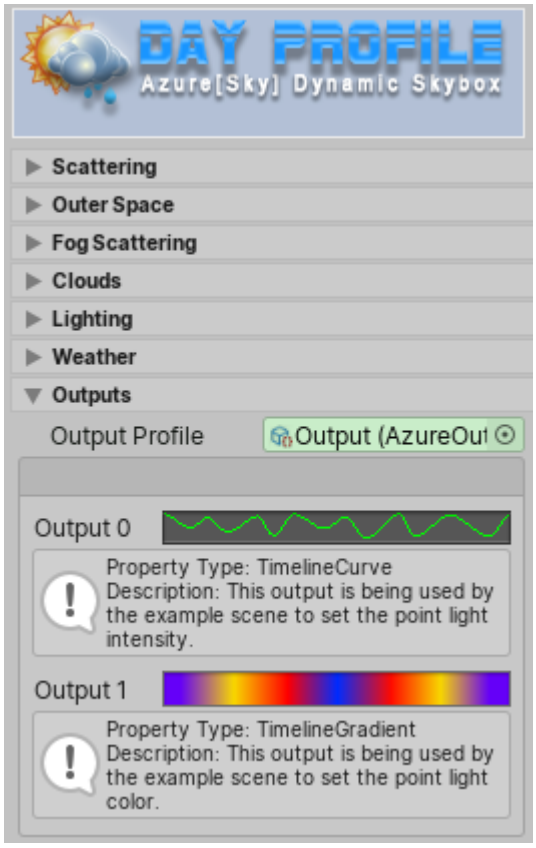


The first output (Output 0 in the list) was set to curve type based on the timeline that will return a different value depending on the time of day. This property will be used to control the intensity of a point light. As you can see from the screenshot above, you can write a text to help you remember why you created this extra property.

The second output (Output 1 in the list) was set to the gradient type based on the timeline that will return a different color depending on the time of day. This property will be used to control the point light color.

You must attach this output profile to the "**Output Profile**" field on the "**Outputs**" tab of the "**Sky Controller**" component.

You also need to attach this output profile into some day profile and customize it. You can plug this same output profile into other day profiles to set different values, the sky controller will manage the output system and perform the blend transition for each extra propertie when the weather changes.



In the last step you need to access the value of each output and use it for the purpose you want, in this example the outputs were created to control the properties of a point light. So this simple script below takes the value of each output and uses it to control the intensity and color of the point light.

```
1.  using UnityEngine;
2.  using UnityEngine.AzureSky;
3.
4.  public class AzureOutputExample : MonoBehaviour
5.  {
6.      public AzureSkyController azureSky;
7.      public Light myPointLight;
8.
9.      private void Update()
10.     {
11.         //Get the float value from the output 0
12.         myPointLight.intensity = azureSky.GetOutputFloatValue(0);
13.
14.         //Get the float value from the output 1
15.         myPointLight.color = azureSky.GetOutputColorValue(1);
16.     }
17. }
```

Note: If the output profile from the sky controller component do not match with the output profile from the day profiles, the output blend transitions will be aborted.