Setting up the Crystal Switch Mechanic 2D

Some notes:

The blocks do not care what they are called or if they are nested in other GameObjects. They just listen to the switch for a change.

All blocks in a scene will be affected by the switch that governs them. Doesn't matter if they are on screen or not.

Creating a Switch

Create an empty *GameObject* in the hierarchy and give it a name. I chose "*CrystalSwitch*" for the purpose of this guide. The name is unimportant.

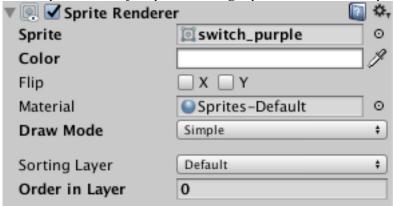
Attach the Crystal Switch.cs file to Switch GameObject. The script will auto-add all the required components.

Configuring the Switch

SpriteRenderer

In the *SpriteRenderer* component, drag a sprite to the sprite field. Or you could use and icon if you do not want the image displayed.

You may have to adjust your Sorting layers or Order in Layer if things do not show up.



BoxCollider2D

The *BoxCollider2D* will auto-check "Is Trigger" to be true from the script.

You can safely collapse this component, unless you need to edit the colliders shape or boundaries.

Crystal Switch Script

On/Off Sprites

For the purpose of this guide I named the variables the colors of the switches.

In this case Blue and Purple.

Drag a sprite you want to be your "Purple State" to the *Sprite Purple* field.

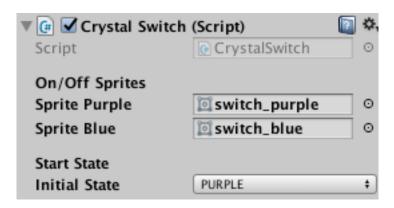
Drag a blue one to the "Sprite Blue" field.

Start State

What "color" should be active?

Active means the blocks are in the "up" state and their box colliders are on.

You can change the color names in the scripts' enum. Just be sure to use refactoring.



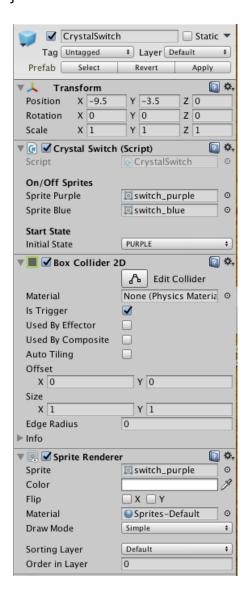
What's going on?

Currently the switch is looking for collision from a GameObject with the "Player" tag.

You can add more tags or change the tag, or have no tag at all, by editing the

```
OnTrigger2D method in the CyrstalSwitch.cs
```

```
private void OnTriggerEnter2D(Collider2D other) {
    if (other.gameObject.CompareTag("Player")) {
        SwapStates();
        TriggerEvent(EVENT_CRYSTAL_SWITCH_NOTIFICATION);
        //add soundfx here if you want.
    }
}
```



Setting up the Blocks

Create an empty GameObject name it Purple Block. The name is unimportant.

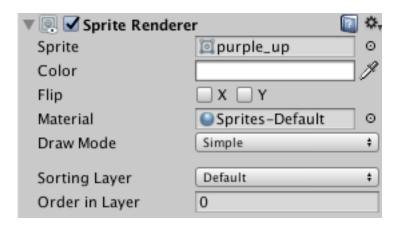
Drag the CrystalBlock.cs to the GameObject. The script will auto-add the appropriate required components.

Configure the Block

SpriteRenderer

In the *SpriteRenderer* component, drag a sprite that corresponds to your block to the sprite field. Or you could use and icon if you do not want the image displayed.

You may have to adjust your Sorting layers or Order in Layer if things do not show up.



You may already know you want the "Purple" blocks to be active when you start the game, you should choose the "up" sprite. Though it is not necessary, either the up or down sprite will do. It is your choice how you want to see it in the editor.

BoxCollider2D

The *BoxCollider2D* is turned **on** and **off** by the script. Ensure that "Is Trigger" is **NOT** checked. (Unless your game calls for it).

You can safely collapse this component if you do not need to edit the colliders shape or boundaries.

Crystal Block Script BLOCK SETUP

Select what color block this is. This will let the block know if it should activate or deactivate when the same color switch is active.

The script is smart enough to know that if this is a "blue" block and you have "purple" as the start state, blue will start in the deactivated state when the game is played.

Chose what sprites should be displayed in the **up** and **down** states.

THE SWITCH

Drag the "Switch" GameObject to the "Governing Switch" field. This lets the block know which switch to listen for changes to.

Drag this completed block *GameObject* to the prefabs folder for later reuse.

Repeat these steps for the other color block.

Here is a screen shot of how both blocks should be set up. 1 Inspector ✓ BlockPurple Static 🔻 Static 🔻 ✓ BlockBlue Tag Untagged Layer Default Tag Untagged Layer Default Prefab Select Revert Apply Prefab Select Revert Transform Transform Position X -3.9340 Y 2.00395 Z 0 Position X 5.06595 Y 2.00395 Z 0 Rotation X 0 Y 0 Z 0 Rotation X 0 Y 0 Z 0 X 1 Z 1 Y 1 Scale X 1 Y 1 Z 1 Scale 🔻 👰 🗹 Sprite Renderer <u>□</u> \$, Sprite Renderer Sprite purple_up 0 Sprite ii blue_up 0 Color Color Flip Flip Material Sprites-Default 0 0 Material Sprites-Default Draw Mode Simple Draw Mode Simple Sorting Layer Default + Sorting Laver Default 0 Order in Laver Order in Layer ▼ 🕝 🗹 Cystal Block (Script) ▼ 🕝 🗹 Cystal Block (Script) **€** CystalBlock Script **€** CystalBlock Script 0 **BLOCK SETUP BLOCK SETUP** This Block Color PURPLE + This Block Color BLUE + Sprite Block Up 0 purple_up Sprite Block Up ii blue_up 0 Sprite Block Down ii purple_down Sprite Block Down ii blue_down 0 THE SWITCH THE SWITCH 0 **Governing Switch ©** CrystalSwitch Governing Switch **©** CrystalSwitch 0 ▼ 🔳 🗹 Box Collider 2D <u>□</u> \$, 🔻 🔳 🗹 Box Collider 2D ₼ Edit Collider A Edit Collider None (Physics Materia 0 Material None (Physics Materia O Material Is Trigger Is Trigger Used By Effector Used By Effector Used By Composite Used By Composite Auto Tiling Auto Tilina Offset Offset X 0 Y 0 Y 0 Size X 1 X 1 Edge Radius Edge Radius Info

Now that your prefabs and switch is setup, you can now duplicate the blocks and create your barriers!

Here is an example Hierarchy.

I put all my Purple blocks in a purple block *GameObject* and blue in a blue *GameObject*. It isn't necessary. I did it for organization.

