Gentleland

Isometric Dark Fantasy Documentation

Hello dear developer!

Thank you for acquiring this asset pack from the Unity Asset Store!

We are a growing art outsourcing agency and are very thankful for your support!
If you like this asset pack, please consider leaving a review on the Unity Asset Store or recommend us to your friends! This would help us greatly!

If you encounter problems of any kind, feel free to reach out!
We will help you further.

Jacky Martin
CEO - Gentleland
jacky@gentleland.net

Summary

Content listing

🏰Isometric Dark Fantasy is a package with 100+ sprites of buildings, decorations and 700+ background tiles. Perfect for 2D games such as 4x, tacticals, RPG or any other genre with an isometric perspective.
Like our package? Please let us a review and give us feedback.

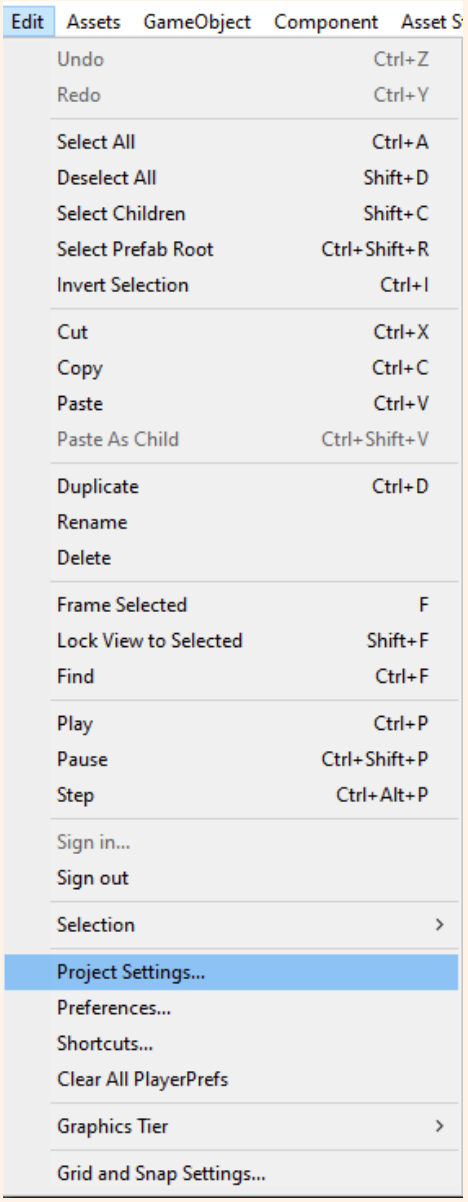
Included are:

- 25 sprite sheets with seamless isometric background tiles (cut into 128x64 pieces), setup to use with Unity's Tilemap systems.
- 1 sprite sheet with 4 different isometric selection borders.
- 17 preview images and 4 cover images.
- 3 fx sprites and 1 cloud sprite sheet containing 11 clouds.
- 7 UI sprites used in the sample.
- 117 unique isometric building and decorative element sprites with pivot and physics shape setup for isometric perspective :
    - 2 camps
    - 12 castles
    - 3 castle ruins
    - 3 crypts
    - 5 cursed mines
    - 2 dungeons
    - 3 forges
    - 6 hall of deities
    - 7 haunted lumber mill
    - 2 laboratory
    - 2 meat farms
    - 3 ramparts
    - 3 soul well
    - 6 summoning portals
    - 21 trees
    - 12 unholy quarry
    - 25 other buildings and decorative elements (bones, tombs, haunted ships, boat)
- 1 demo scene showing the sprites on a unicolor background
- 2 decorated example scenes showing the sprites on a video game environment
- a sample "city builder" project which shows with a small prototype what is possible to do with the assets.

All sprites are delivered on a transparent background as png files.

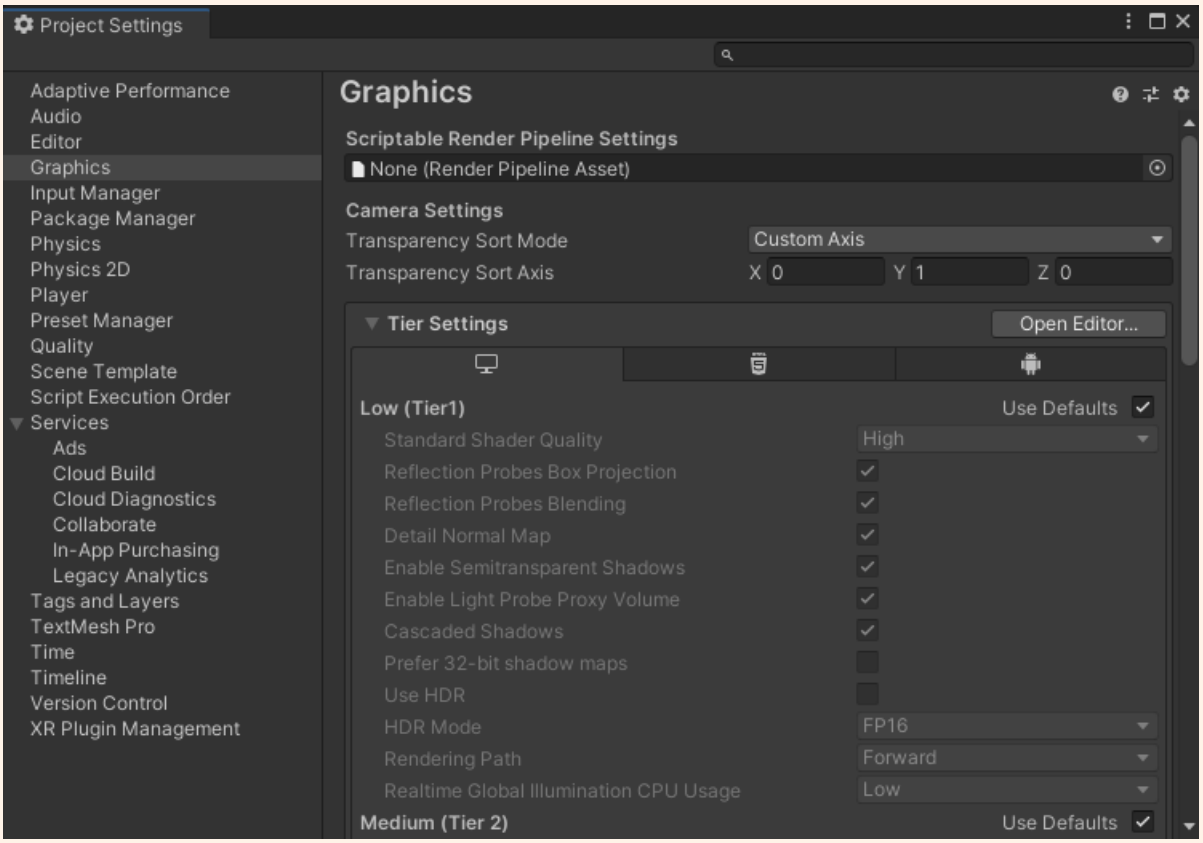
Isometric Ordering

To render isometric art in the correct order in Unity, we need to set up the transparency sort mode and sort axis correctly.
To do this, **Open up "Edit" and find "Project Settings".**



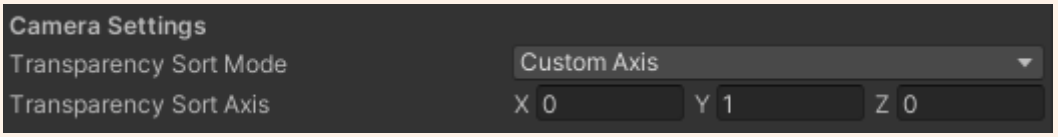
Edit → Project Settings

Once the Project settings window is opened **select the “Graphics”** panel**.**



Project Settings ⟶ Graphics

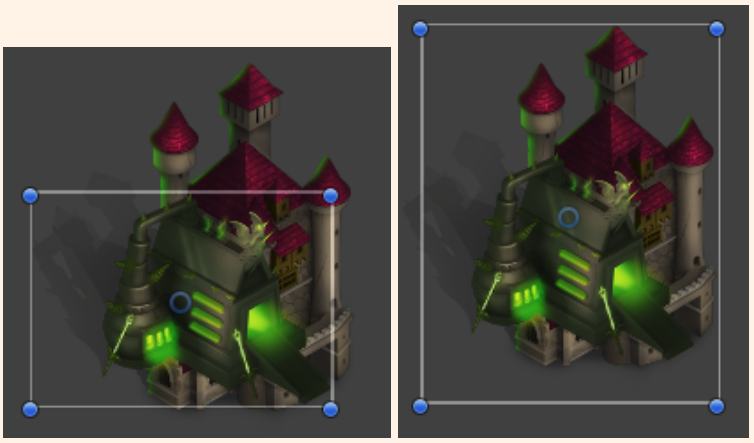
In the **camera settings**, you need to set the **“transparency sort mode” to “Custom Axis”** so that the axis defined below is used as a reference for sorting the sprites.
Then you need to set the sort axis to x=0 y=1 z=0 so that the sprites are sorted by the Y axis.



Graphics ⟶ Camera Settings

This allows the camera to sort sprites based on their Y-position.

In Unity, when you add a sprite to a scene, the position of the center of the sprite is used by default for comparison with other sprites. But in isometric, that's not what we want, because the center of the sprite may not be a relevant position.

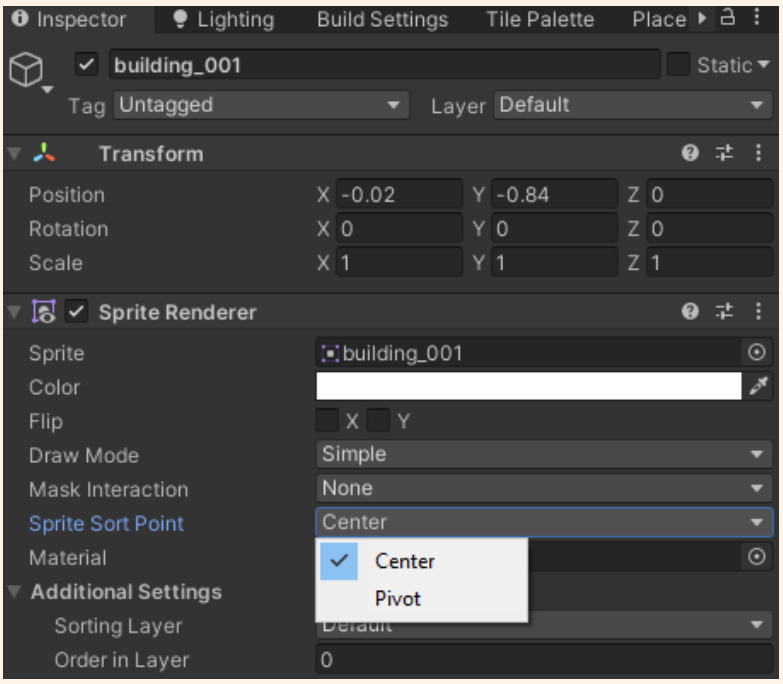


Example for sorting sprites.

For example, here the castle (in red) is lower than the forge (in green), so it should be drawn in front of it, but as you can see, the position used to compare these sprites is the center of the sprites (represented by the blue circle).

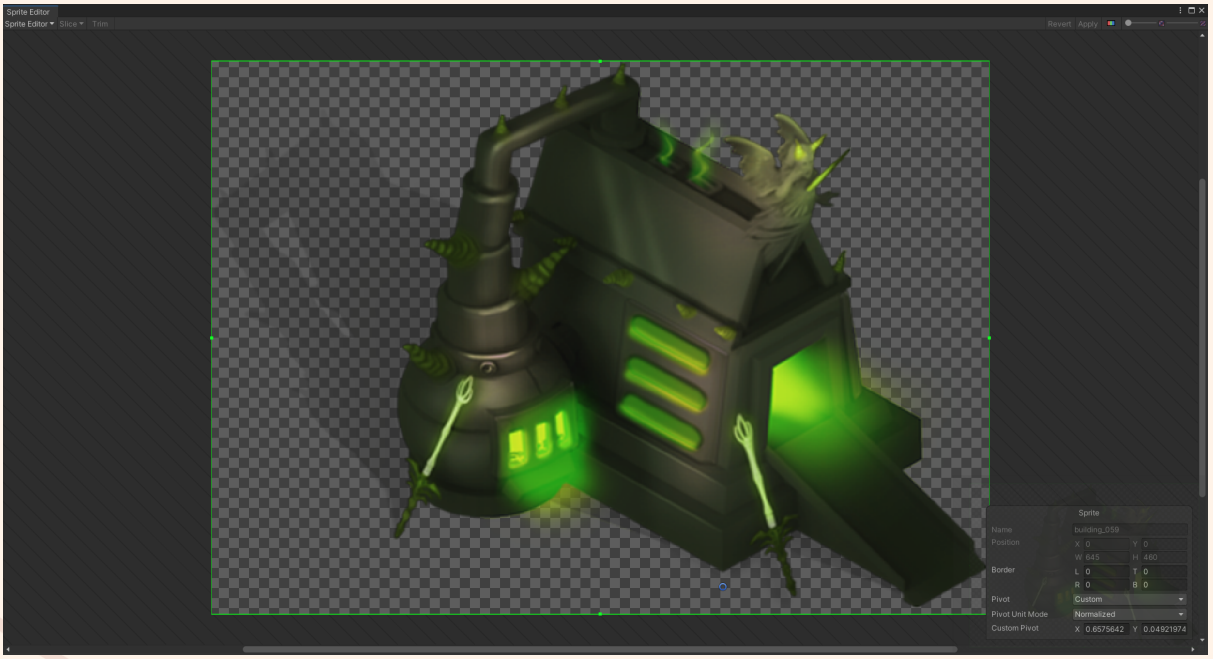
This is not what we want, but fortunately Unity allows us to use another position to sort the sprites, namely the **pivot** point.
You can select which position is used to sort the sprite in the Inspector panel on the sprite :
Inspector window → Renderer Component → Sprite Sort Point.

If you select **"Pivot"**, the pivot you can place manually will be used.



Inspector panel for sorting sprites

We have set all the sprites' pivots to work in most cases, but it might not be perfect for your case. You can change the pivot at any time using the sprite editor by moving the blue circle to the desired position.



Example of the forge in the Sprite Editor window

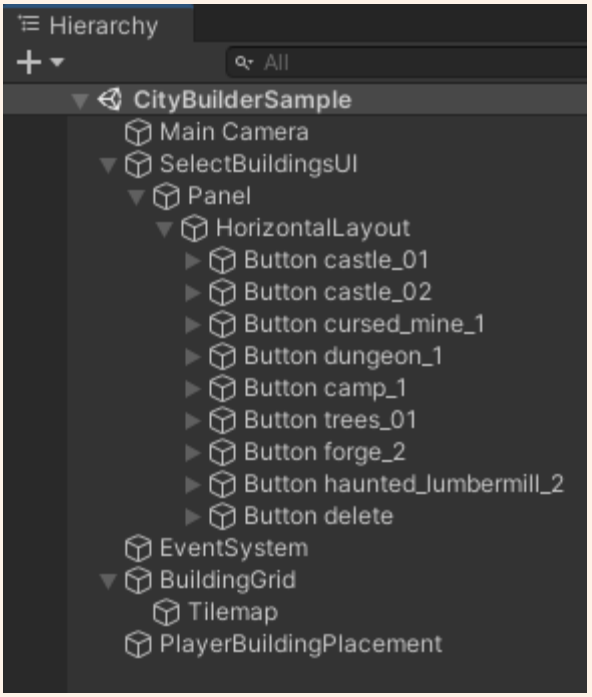
City Builder Sample

Screenshot of the CityBuilderSample Scene in Play Mode

As a sample we added you a small city builder feature!
To see the sample in action launch the **CityBuilderSample Scene** in **Assets/Gentleland/IsometricVampires/Samples/CityBuilder/Scenes**

The city builder is a small sample, where the player can select a building by clicking on the building icon at the bottom, and place the selected building by clicking on the ground. The player can also delete buildings by clicking on the cross that selects the delete mode and then clicking on a building or other environment piece to delete it.

Scene Hierarchy



Screenshot of the CityBuilderSample Scene Hierarchy

- ★ **MainCamera** is the Camera of the scene
- ★ **SelectBuildingUI** is the UI container with all the buttons
- ★ **EventSystem** is a unity builtin GameObject responsible for managing inputs for UI elements.
- ★ **BuildingGrid** is the grid on which the buildings are built. It also has a **Tilemap** child used for the ground.
- ★ **PlayerBuildingPlacment** is the GameObject responsible to process player inputs and do the correct actions resulting from it.

To see the isometric grid in the scene you must select the **BuildingGrid** GameObject.

Scripts Documentation

Building.cs

This class is the monobehaviour class that has to be on each building.
It has 2 member variables :

- ★ `public SpriteRenderer spriteRenderer` is a reference to the SpriteRenderer.
- ★ `public Vector2Int gridSize` is the size of a building on the grid (x and y axis).

BuildingGrid.cs

This class is a singleton and thus contains a static variable of its own type called instance.
`public static BuildingGrid instance`

It has 2 member variables :

- ★ `Grid grid` is a reference to the Grid (a built-in component from unity)
- ★ `Dictionary<Vector2Int,Building> tilesOccupiedBybuildings` is a dictionary of occupied tiles(coordinates) referencing the building which occupies them.

This class has 7 functions :

Awake

`void Awake()`
This function is a builtin function that is called to initialize MonoBehaviours.

AddBuildingAtGridPosition

`public void AddBuildingAtGridPosition(Vector2Int gridPos, Building building)`
This function adds a Building to the tilesOccupiedBybuildings dictionary for each grid position occupied by the building at the given position (using the gridSize)

IsBuildingAtGridPositionPossible

`public bool IsBuildingAtGridPositionPossible(Vector2Int gridPos, Building building)`
This function checks if it is possible to add a building at a given position by verifying if a tile in the grid size of the building is already occupied.

DeleteBuilding

`public void DeleteBuilding(Building building)`
This function delete all `Vector2Int, Building` pairs in the dictionary referencing the given building `tilesOccupiedBybuildings` and then delete the gameObject owning the given Building.

GetBuildingAtPos

`public Building GetBuildingAtPos(Vector2Int gridPos)`
This function returns the building referenced by the given position in the `tilesOccupiedBybuildings` dictionary. If there is no it returns null.

WorldToGridPosition

`public Vector2Int WorldToGridPosition(Vector3 pos)`
This function converts a world space position to grid space position.

GridToWorldTilePosition

`public Vector3 GridToWorldTilePosition(Vector2Int gridPos)`
This function converts a grid space position to world space position.

PlayerBuildingPlacement.cs

This class is the class responsible to process player inputs and do the correct actions resulting from it.
It has 3 member variables available in the Inspector window :

- ★ `Color deleteColor` is the color a building takes when the player has mouse over the building (in Delete mode).
- ★ `Color wrongPlacementColor` is the color a building takes when the mouse position is not a correct position for the selected building (in Place mode).
- ★ `Color goodPlacementColor` is the color a building takes when the mouse position is a correct position for the selected building (in Place mode).

It has 2 member variables to reference currently selected building :

- ★ `GameObject selectedBuildingPrefab` is the reference to the selected building's Prefab (in Place mode)..
- ★ `Building selectedBuildingInstance` is the reference to the selected building's Instance (in Place mode).

It also has 3 other members :

- ★ `BuildingPlacementMode mode` is the current mode (Place or Delete).
- ★ `Building mouseOverbuilding` is the reference to the last building with the mouse over it (in Delete mode).

- ★ `Color mouseOverbuildingColor` is the color of the last building with the mouse over it (in Delete mode).

It only has 2 functions :

Update
`void Update()`
This function is a Unity builtin function called each frame in which all the input processing is done.

SelectBuilding
`public void SelectBuilding(GameObject newBuildingPrefab)`

This function Selects the Place mode and the given Building which allows to place it with mouse.

SelectDeleteMode
`public void SelectDeleteMode()`
This function Selects the Delete mode which allow to delete buildings by clicking on them.

UISelectionMenu.cs

This class is the class responsible to show and hide the tree selection ui
It has 4 member variables :

- ★ `bool shown` keeps the state of the ui true if shown false elsewise.
- ★ `GameObject visuals` is the reference to the parent gameObject of the UI to show or hide..
- ★ `float ySpacing` is the space(in Y axis) between the button and the ui selection menu to show.

It has a unique function :

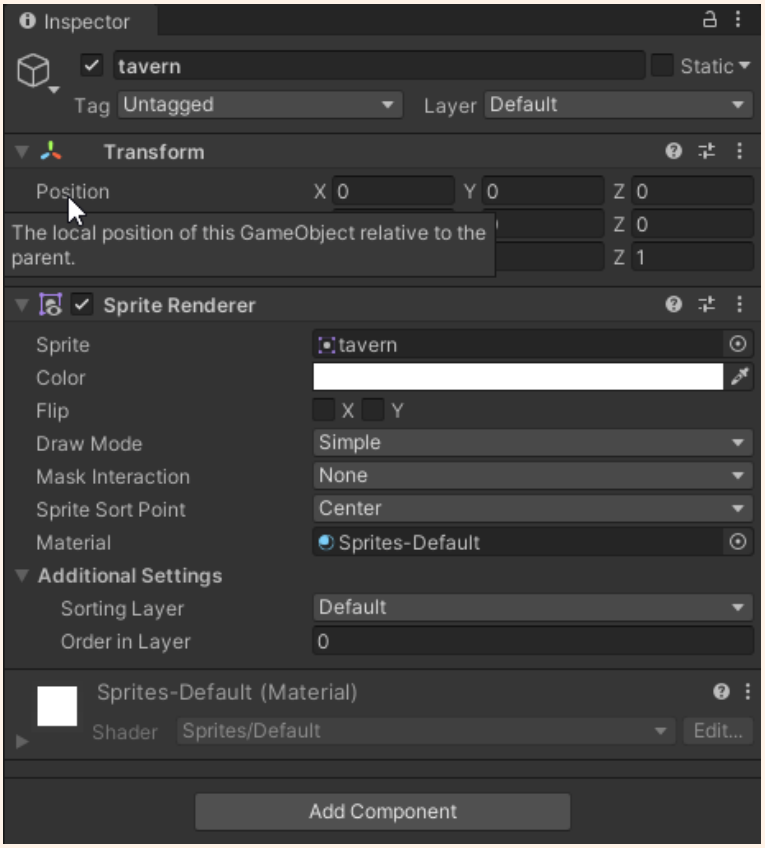
OnClick
`public void OnClick(Transform buttonTransform)`
This function is called when you click on the button which opens or closes the tree selection menu and shows or hides visuals accordingly.

Adding a new Building to the selection menu

To add a new building to the selection menu there are 2 things to do (In Edit Mode).

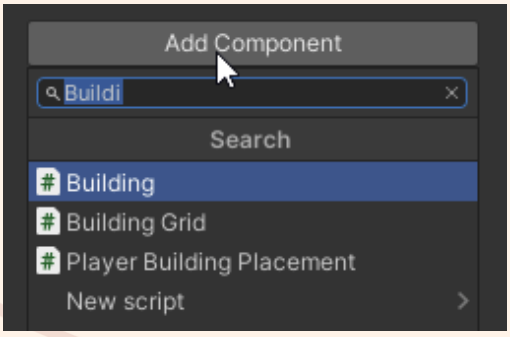
1. Create a building prefab for the new building:

To do so, drag the sprite of the building you want in the scene it should create a GameObject with a SpriteRenderer referencing the sprite you selected.



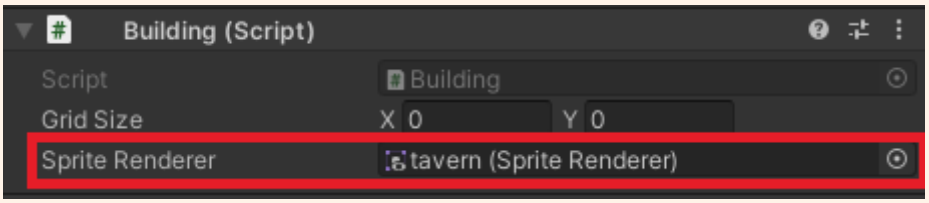
Inspector window of the sprite's GameObject

Then Add a Building Component To it by clicking **Add Component** in the Inspector window, type Building and click on **Building**.



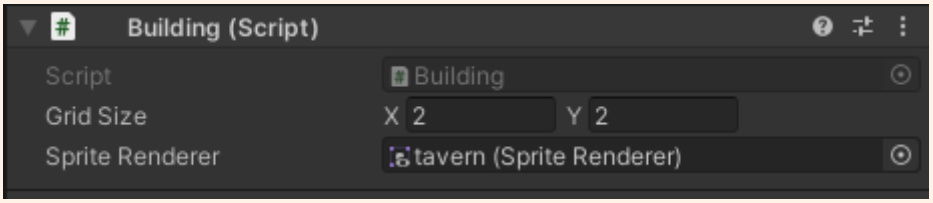
Add Component search field

Drag the **SpriteRenderer** to the Building Component to reference it.



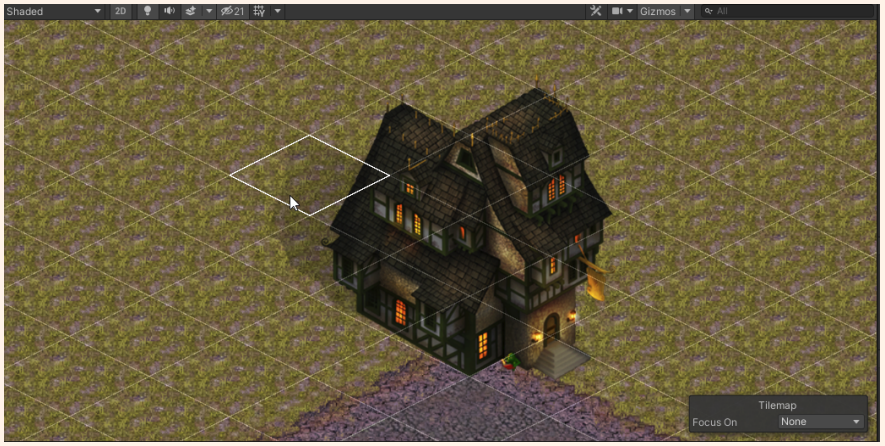
Building Component with Sprite Renderer reference

Set a **Grid Size** depending on the space the building takes on the grid.



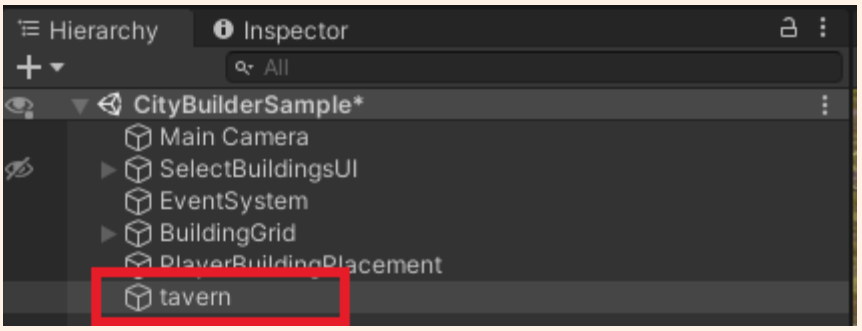
Building Component with Sprite Renderer reference and Grid Size

To help you set the correct size, place the gameobject in the scene and select the **BuildingGrid** to see it (you can always adjust it later).

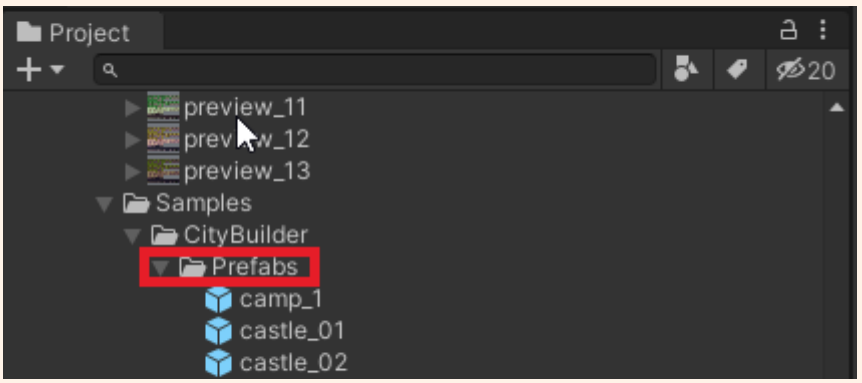


Scene View with the sprite placed in the scene and the BuildingGrid selected

Then you can drag the sprite's GameObject from the **Hierarchy** window to the **Project** window inside the Prefab folder so that it will create a prefab.

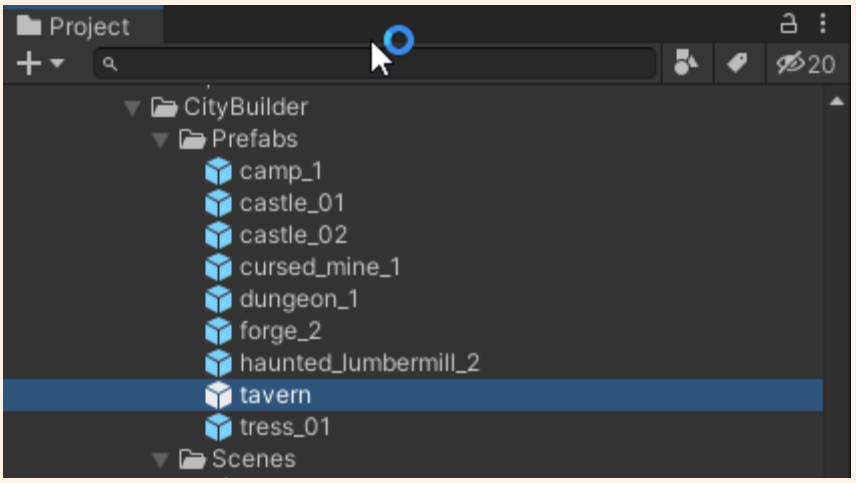


Hierarchy window with the sprite's GameObject



Project window with the Prefabs folder

It should create a **new file**.

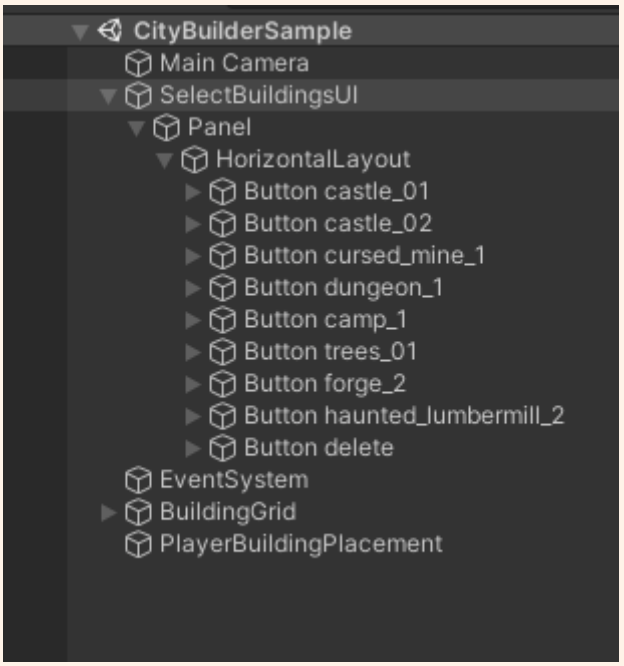


Project window with the new file

If so you created the prefab and can **delete** the sprite's GameObject from the scene.

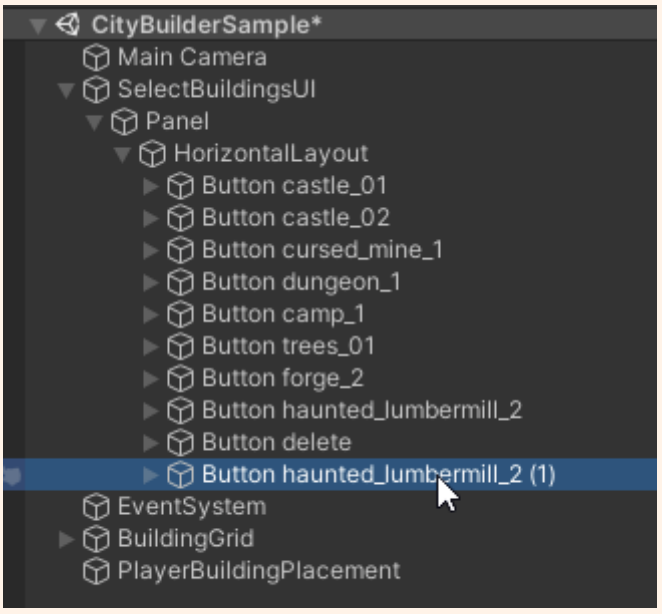
2. Create a button to select the prefab:

Now that we have our prefab we want to create a button in the User Interface to select it.
To do so, unfold the **SelectBuildingsUI** and his childs until you find the buttons



Hierarchy Window with the UI GameObjects unfold

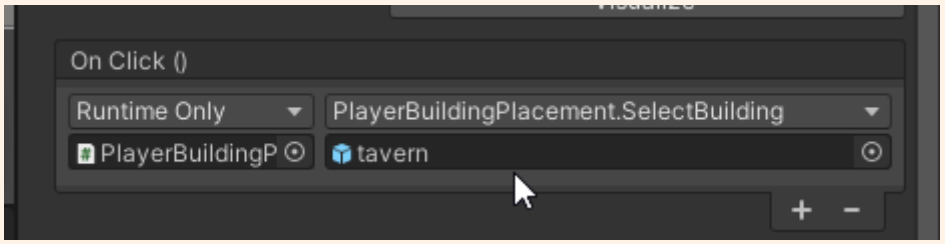
Then **duplicate** a button (but the delete one) by using Ctrl+D or right click -> Duplicate



Hierarchy window with the duplicated button

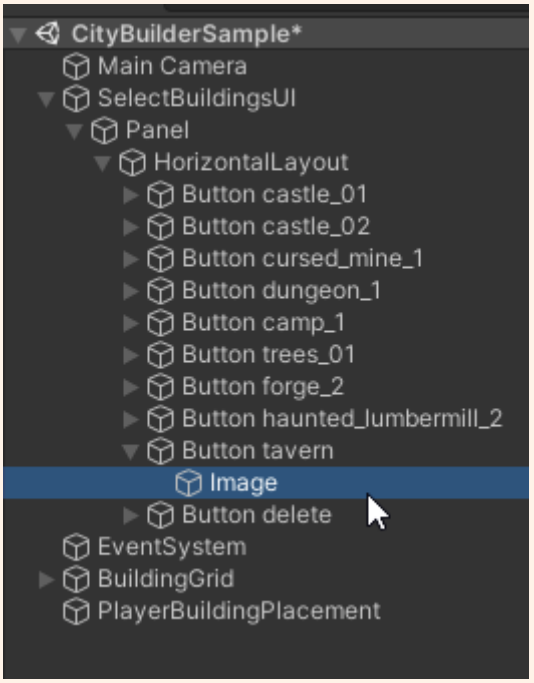
Once it is done you should have a new object move it so that Button delete is the last one.
You can rename the button you just created to have a correct name.

In the Inspector window (having the button selected) change the prefab inside the **OnClick** event to reference the one we just created.



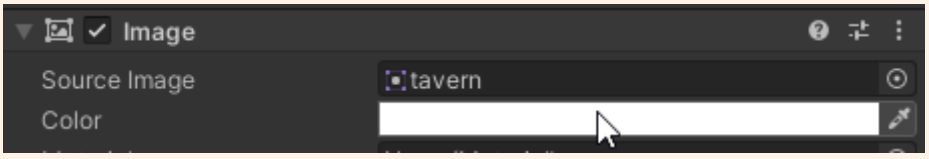
OnClick event in the Inspector window

Then unfold the button on the Hierarchy window and select the Image object inside it.



Hierarchy window with sub GameObject Image selected

You can now drag the correct sprite to the **Source Image** field in the Inspector window.

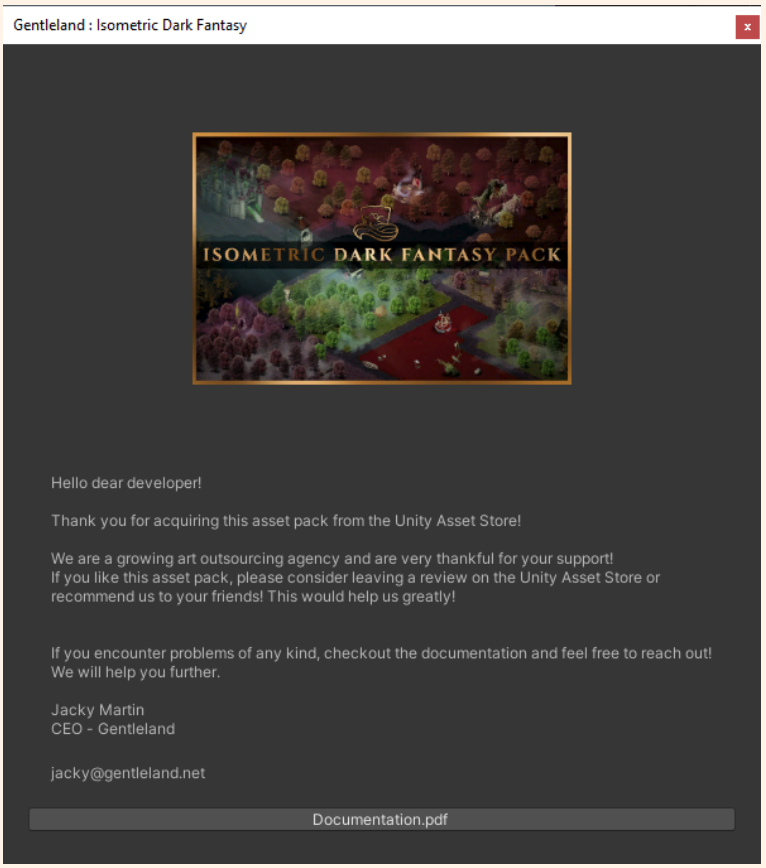


Inspector window with the correct Source Image

You're done now press play and try it out!

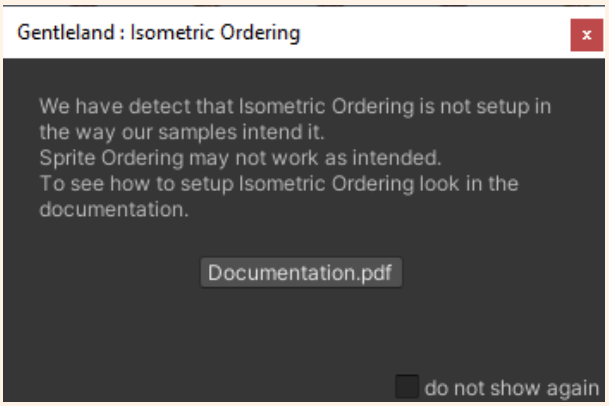
Gentleland Utils

As you will probably notice when you import the asset a window shows up.



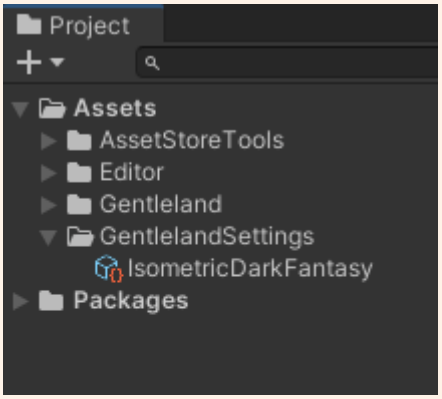
Welcome window

If your settings for sprite Rendering order are not in the way we put them another window should show up. If you don't care and don't want to see it again, you can tick do not show again, it will not check again whether the settings are correct or not.



Isometric Ordering window

At first load we create a folder called “GentlelandSettings” containing a file called “IsometricDarkFantasy.asset” which tracks whether it is the first time the asset is loaded or not and if you want that we check if the settings for Isometric Ordering are set up in the way it is intended for the sample and demo scenes to work.



Project window with the

If you delete this folder it will be recreated. If you don't want any of that you can delete the Utils folder under ‘Gentleland/IsometricDarkFantasy/Utils” and then delete the “GentlelandSettings" folder.