Tutorial Three: Dropping an item

Game Programming Project

By Mariana Neiva Santos Silva

What you'll learn

In this Tutorial you will learn how to drag an item in a 2D game in Unity.

Programs used

UNITY

Game Engine

VISUAL STUDIO

Code Editor

ADOBE STOCK

Stock images

What you should already know:

1 A basic understanding of **Unity;**

2 Basic understanding of **C#**

2 Have followed tutorial 1 and 2

Let’s beginning!

Steps

timeline

SETTING UP THE SCENE

CREATING THE SLOT

CODING THE SCRIPT

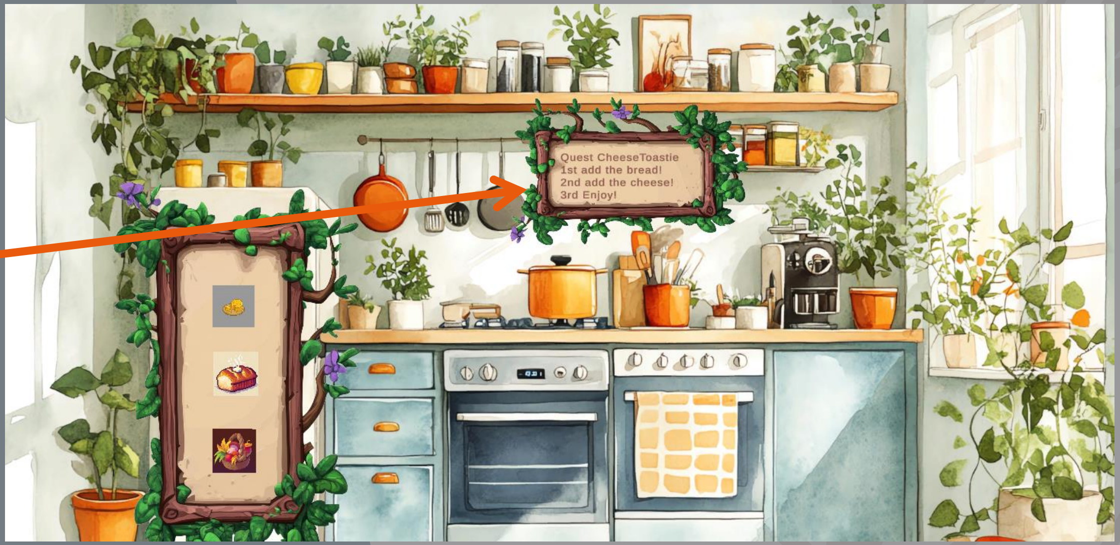
CONNECTING SCRIPT TO THE ITEMS

TESTING

Step 1: Setting the scene

THE SCENE:

- For this tutorial we will be using the same scene and script from the last tutorial.
- If you want to add anything extra to your scene now is the time!
- For example, I added a Quest board, with the skills I showed in the previous tutorials



Quest CheeseToastie
1st add the bread!
2nd add the cheese!
3rd Enjoy!

timeline

SETTING UP THE SCENE

CREATING THE PREFABS

CODING THE SCRIPT

CONNECTING SCRIPT TO THE ITEMS

TESTING

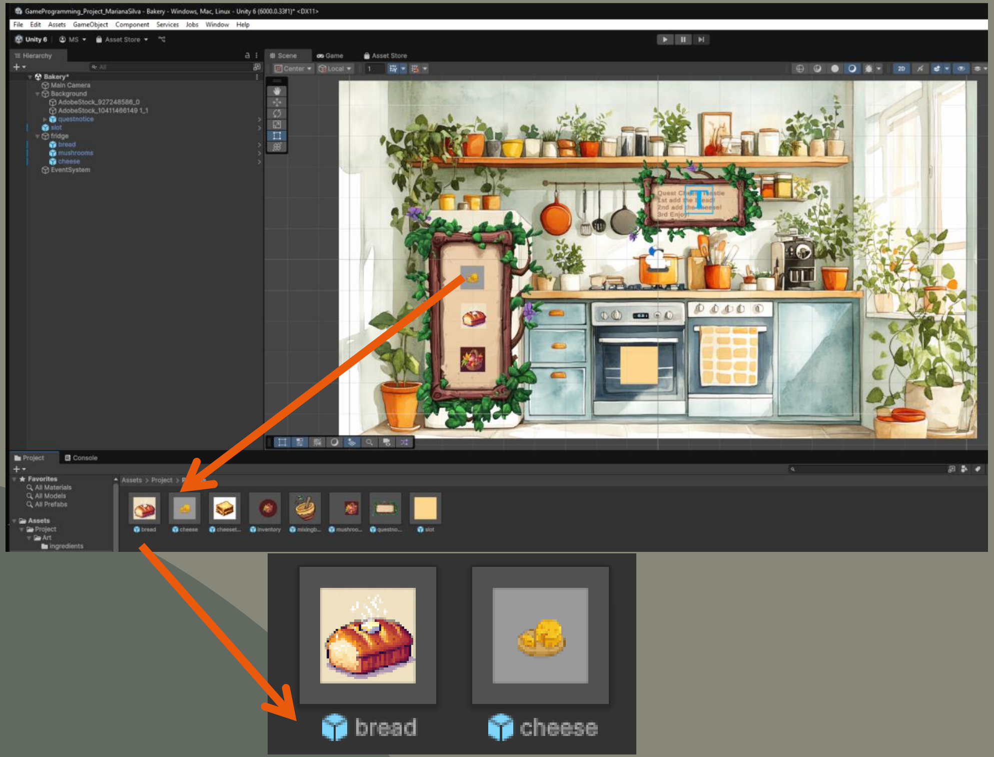
Step 2: Creating the Prefabs:

WHAT IS A PREFAB?

- To put it simply a prefab is an asset you can re-use.
- One thing to not is that if you want to change a prefab you need to go to the original prefab and modify it. This way all prefabs will be changed.
- However, if you change a prefab. that is in a scene you are only changing that version of the prefab.

CREATING A PREFAB.

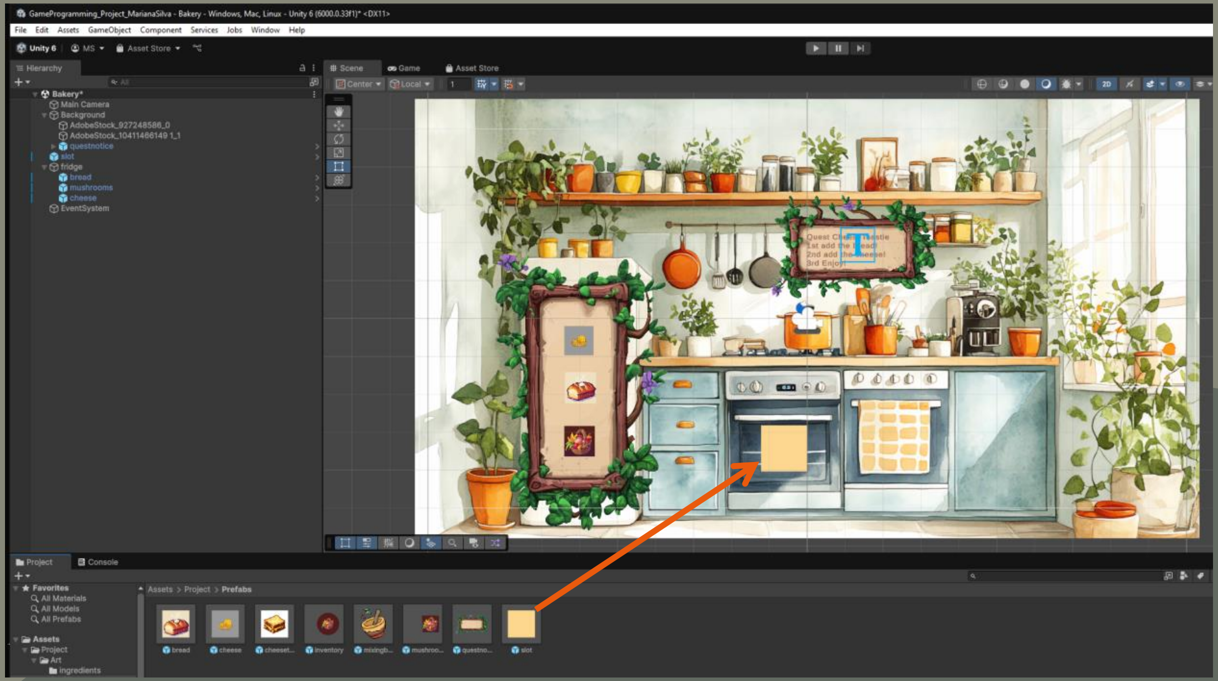
- Creating a prefab is very simple, whatever asset you want to make re-usable and drag it on to the project.
- Ideally you should drag it to a folder called Prefabs! That's why we created that folder in the first tutorial.
- It will them appear with a light blue cube next to it in the project, inspector and hierarchy.



Step 2: Creating the Prefabs:

THE SLOT

- Now drag the slot prefab into the scene.


GameProgramming_Project_MarianaSilva - Bakery - Windows, Mac, Linux - Unity 6 (6000.0.33f1)* <DX11>
File Edit Assets GameObject Component Services Jobs Window Help
Hierarchy
Bakery*
Main Camera
Background
questnotice
slot
fridge
bread
mushrooms
cheese
EventSystem
Scene
Game
Asset Store
Project
Console
Assets > Project > Prefabs
bread
cheese
inventory
slot

timeline

SETTING UP THE SCENE

CREATING THE PREFABS

CODING THE SCRIPT

CONNECTING SCRIPT TO THE ITEMS

TESTING

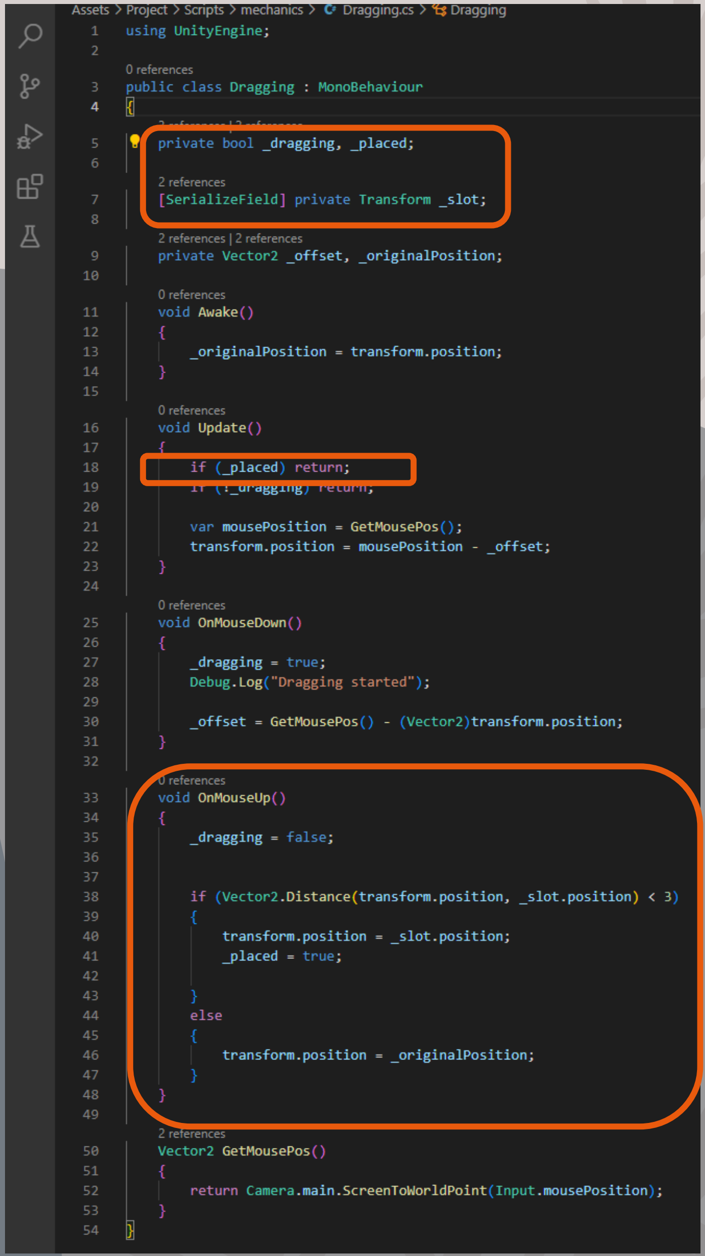
Step 3: The Script

ADDING TO THE SCRIPT

- We will be adding onto the script from the last tutorial as highlighted.
- If you change the name of the script in Unity, the script will not update itself.
- My script is called "Dragging" if I want to change it to DraggingAndDropping I would need to change it in Unity and in the script in the line "***`public class Dragging : MonoBehaviour`***".
- It would now be "***`public class DraggingAndDropping : MonoBehaviour`***".

THE SCRIPT

- This script will allow you to drag and drop an object around in unity. It won't allow you to do anything else. We will cover that in the next tutorial.


Assets > Project > Scripts > mechanics > Dragging.cs > Dragging

```csharp
using UnityEngine;

0 references
public class Dragging : MonoBehaviour
{
    private bool _dragging, _placed;

    2 references
    [SerializeField] private Transform _slot;

    2 references | 2 references
    private Vector2 _offset, _originalPosition;

    0 references
    void Awake()
    {
        _originalPosition = transform.position;
    }

    0 references
    void Update()
    {
        if (_placed) return;
        if (!_dragging) return;

        var mousePosition = GetMousePos();
        transform.position = mousePosition - _offset;
    }

    0 references
    void OnMouseDown()
    {
        _dragging = true;
        Debug.Log("Dragging started");

        _offset = GetMousePos() - (Vector2)transform.position;
    }

    0 references
    void OnMouseUp()
    {
        _dragging = false;


        if (Vector2.Distance(transform.position, _slot.position) < 3)
        {
            transform.position = _slot.position;
            _placed = true;

        }
        else
        {
            transform.position = _originalPosition;
        }
    }

    2 references
    Vector2 GetMousePos()
    {
        return Camera.main.ScreenToWorldPoint(Input.mousePosition);
    }
}
```

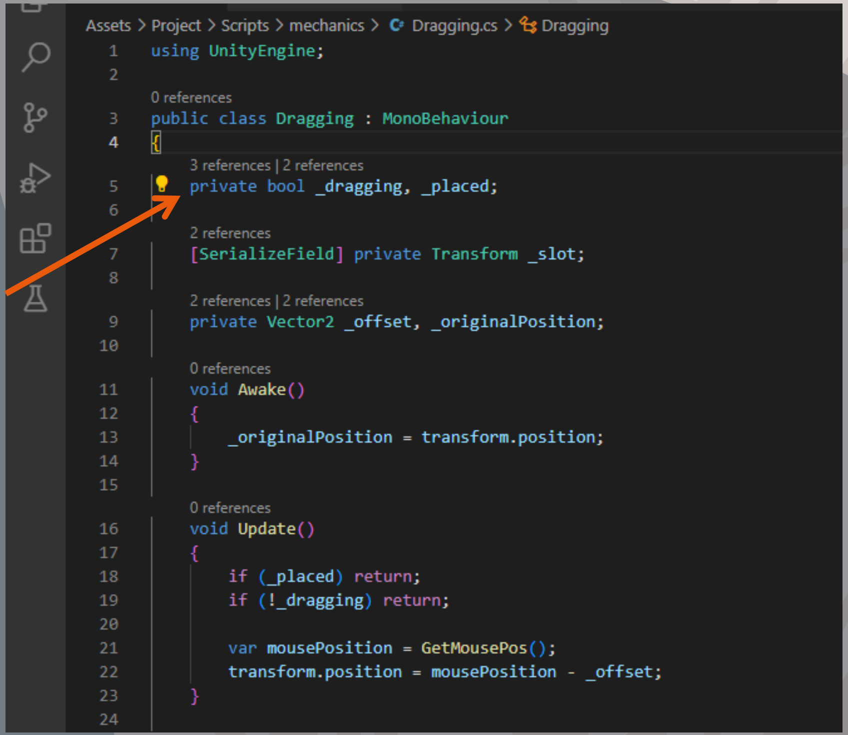
But why?!

You might be asking why we are adding this to the same script.
After all it's something different!
Dragging and dropping are in the same action. A click of a mouse. We don't need nor should we make scripts for everything individually!
It is equally bad for a script to have too many things as it is to have a different script for every element.
The way I like to see this is you should create scripts by categories or drawers.
You wouldn't add your plates to your cutlery drawer at home or have a drawer for forks and another one for knifes – at least I hope you don't!

Step 3: Understanding the script

ADDING VARIABLES

- We will start by adding 2 variables!
- ***`private bool _dragging, _placed;`***
- First, we will add another private Boolean variable called "***`_placed`***" using the same logic as last time, we now want to see if the item is placed or not (true or false).
- The is no need to re write "***`private bool`***" in another line simply add a coma after the variable you already set in this case "***`_dragging`***".

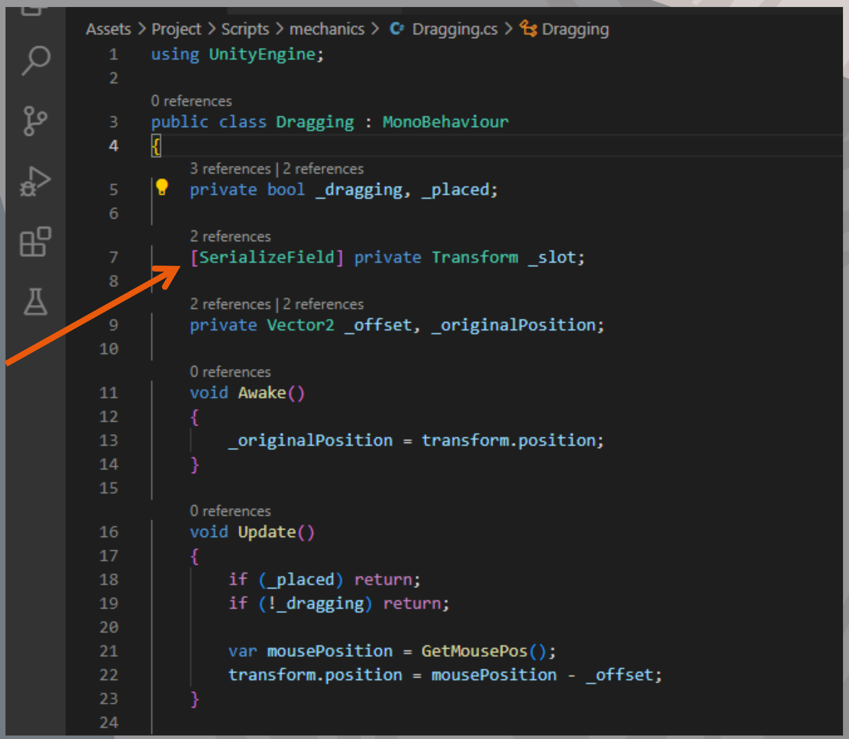

Assets > Project > Scripts > mechanics > Dragging.cs > Dragging

```
using UnityEngine;

0 references
public class Dragging : MonoBehaviour
{
    3 references | 2 references
    private bool _dragging, _placed;

    2 references
    [SerializeField] private Transform _slot;

    2 references | 2 references
    private Vector2 _offset, _originalPosition;

    0 references
    void Awake()
    {
        _originalPosition = transform.position;
    }

    0 references
    void Update()
    {
        if (_placed) return;
        if (!_dragging) return;

        var mousePosition = GetMousePos();
        transform.position = mousePosition - _offset;
    }

```

Step 3: Understanding the script

ADDING VARIABLES

- This line sets the second variable:
- *“[SerializeField] private Transform _slot;”*
- Let’s break it down:
  - While this variable is private “***[SerializeField]*** ” forces unity to “serialize” or in other words show us the private variable. However, it is important to be aware this doesn’t work for all variables.
  - “***Transform***” makes it possible for us to store the position of an asset.
  - Ergo by declaring the “***Transform _slot***” we are telling unity to store the position of whatever asset we add to the script in the inspector “***[SerializeField]***”.


Assets > Project > Scripts > mechanics > Dragging.cs > Dragging

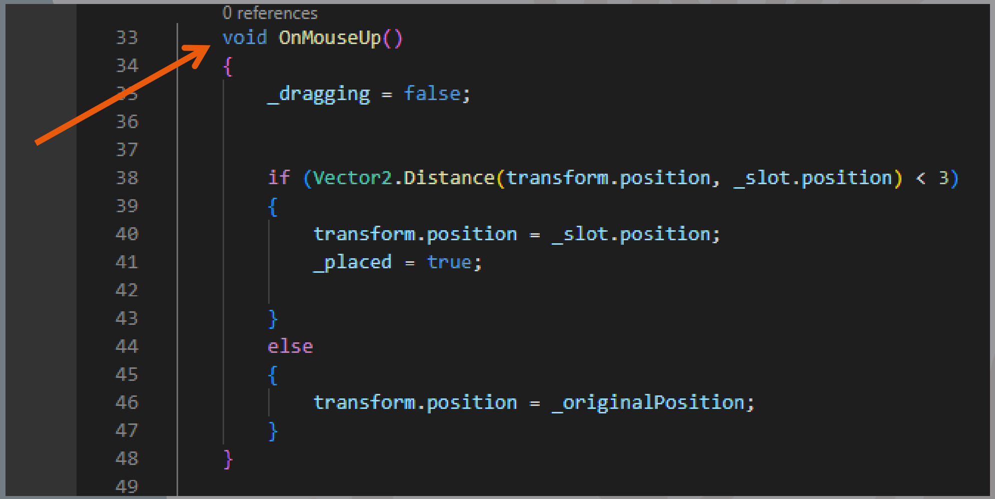
```
using UnityEngine;

0 references
public class Dragging : MonoBehaviour
{
    3 references | 2 references
    private bool _dragging, _placed;

    2 references
    [SerializeField] private Transform _slot;

    2 references | 2 references
    private Vector2 _offset, _originalPosition;

    0 references
    void Awake()
    {
        _originalPosition = transform.position;
    }

    0 references
    void Update()
    {
        if (_placed) return;
        if (!_dragging) return;

        var mousePosition = GetMousePos();
        transform.position = mousePosition - _offset;
    }

```

Step 3: Understanding the script

VOID ONMOUSEUP()

- We are creating a method that only runs when the player lets go of the mouse, therefore it will always run after "***void OnMouseDown()***" that we created on the previous tutorial.

- To start this uses a variable we had set on tutorial 2, "***_dragging***"
- This means that when this method starts running dragging is no longer true.

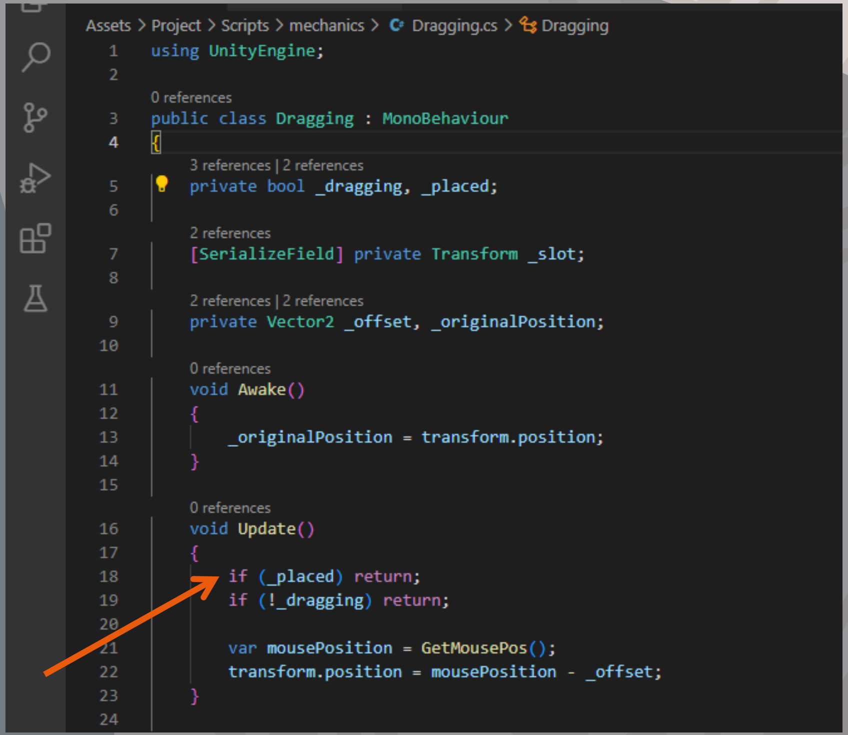


```
0 references
void OnMouseUp()
{
    _dragging = false;


    if (Vector2.Distance(transform.position, _slot.position) < 3)
    {
        transform.position = _slot.position;
        _placed = true;

    }
    else
    {
        transform.position = _originalPosition;
    }
}
```

Step 3: Understanding the script

_PLACED

- This stops the method the "***void Update()*** " from running any further if the variable "***_placed***" is true.
- If this does not happen this line will not do anything!
- We have this If statement because the method "***void Update()*** " does not need to un if the item was placed.



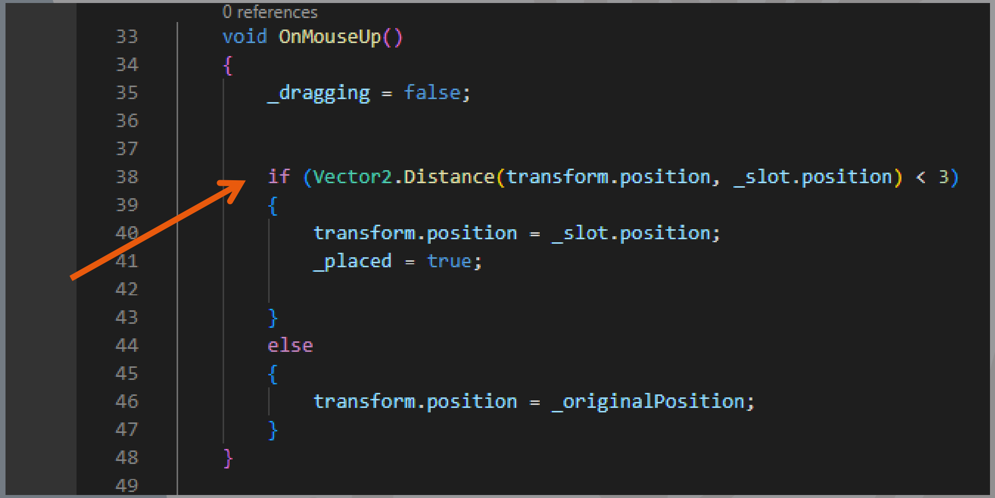
Assets > Project > Scripts > mechanics > Dragging.cs > Dragging

```
using UnityEngine;

0 references
public class Dragging : MonoBehaviour
{
    3 references | 2 references
    private bool _dragging, _placed;

    2 references
    [SerializeField] private Transform _slot;

    2 references | 2 references
    private Vector2 _offset, _originalPosition;

    0 references
    void Awake()
    {
        _originalPosition = transform.position;
    }

    0 references
    void Update()
    {
        if (_placed) return;
        if (!_dragging) return;

        var mousePosition = GetMousePos();
        transform.position = mousePosition - _offset;
    }
```

Step 3: Understanding the script

AN IF STATMENT

- We are using an If statement to either (if) place the item or (else) return it to its original position.
- Let's simplify!
- First, we declare our IF:
- "***if (Vector2.Distance(transform.position, _slot.position) < 3)***"
- "***Vector2.Distance***" This calculates the distance between 2 points in this case the points "***transform.position***" and "***_slot.position***":
  - "***transform.position***" position of item being dragged.
  - "***_slot.position***" this is a place that holds the slot's position
- "***< 3***" this makes it so that distance between the previous points need to be less than (<) 3 units of the slot.

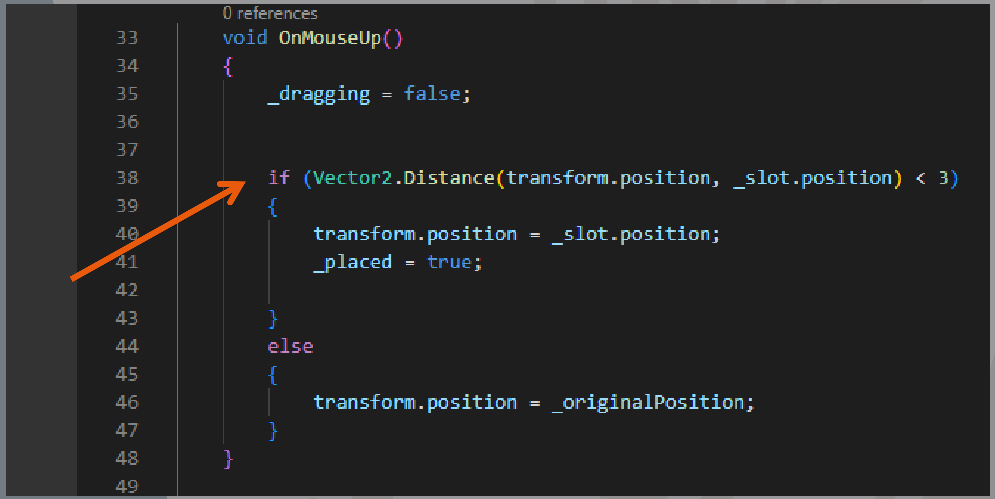


```
0 references
void OnMouseUp()
{
    _dragging = false;


    if (Vector2.Distance(transform.position, _slot.position) < 3)
    {
        transform.position = _slot.position;
        _placed = true;

    }
    else
    {
        transform.position = _originalPosition;
    }
}
```

Step 3: Understanding the script

AN IF STATMENT

- ***`transform.position = _slot.position;`***
- If the player successfully moves the item to the slot the position of the item ("***`transform.position`***") is now equal to the slot position ("***`_slot.position`***").
- In other words, the item drops to the slot if the player moves it close enough
- ***`_placed = true;`***
- We now declare that the variable "***`_placed`***" only ***`if`*** the requirements we set at met.

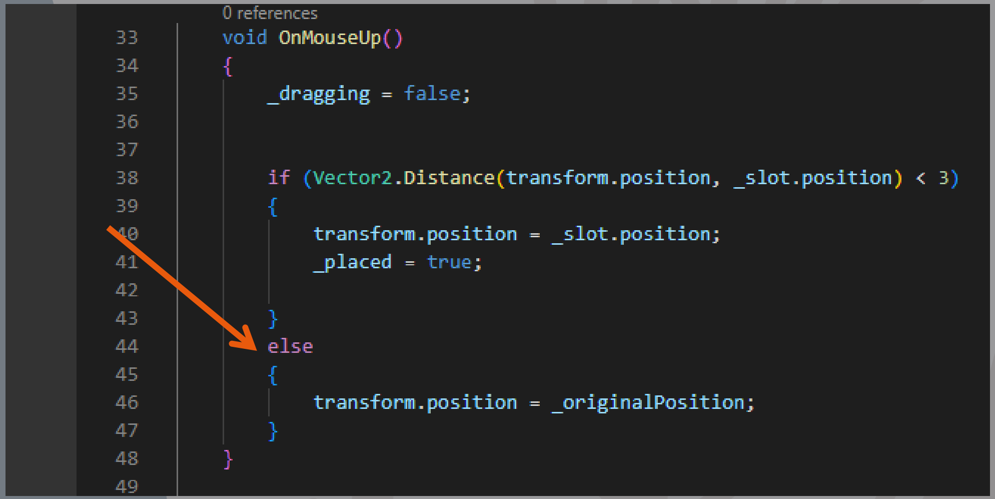


```
0 references
void OnMouseUp()
{
    _dragging = false;


    if (Vector2.Distance(transform.position, _slot.position) < 3)
    {
        transform.position = _slot.position;
        _placed = true;

    }
    else
    {
        transform.position = _originalPosition;
    }
}
```

Step 3: Understanding the script

AN IF STATMENT

- "*else*"
- This mean that if the requirements are not meat the following will happen
- The item will return to the "" which we set in the previous tutorial.
- In other words, the item will move back to the starting position if the player doesn't move it close enough to the slot



```
0 references
void OnMouseUp()
{
    _dragging = false;


    if (Vector2.Distance(transform.position, _slot.position) < 3)
    {
        transform.position = _slot.position;
        _placed = true;

    }
    else
    {
        transform.position = _originalPosition;
    }
}
```

timeline

SETTING UP THE SCENE

CREATING THE PREFABS

CODING THE SCRIPT

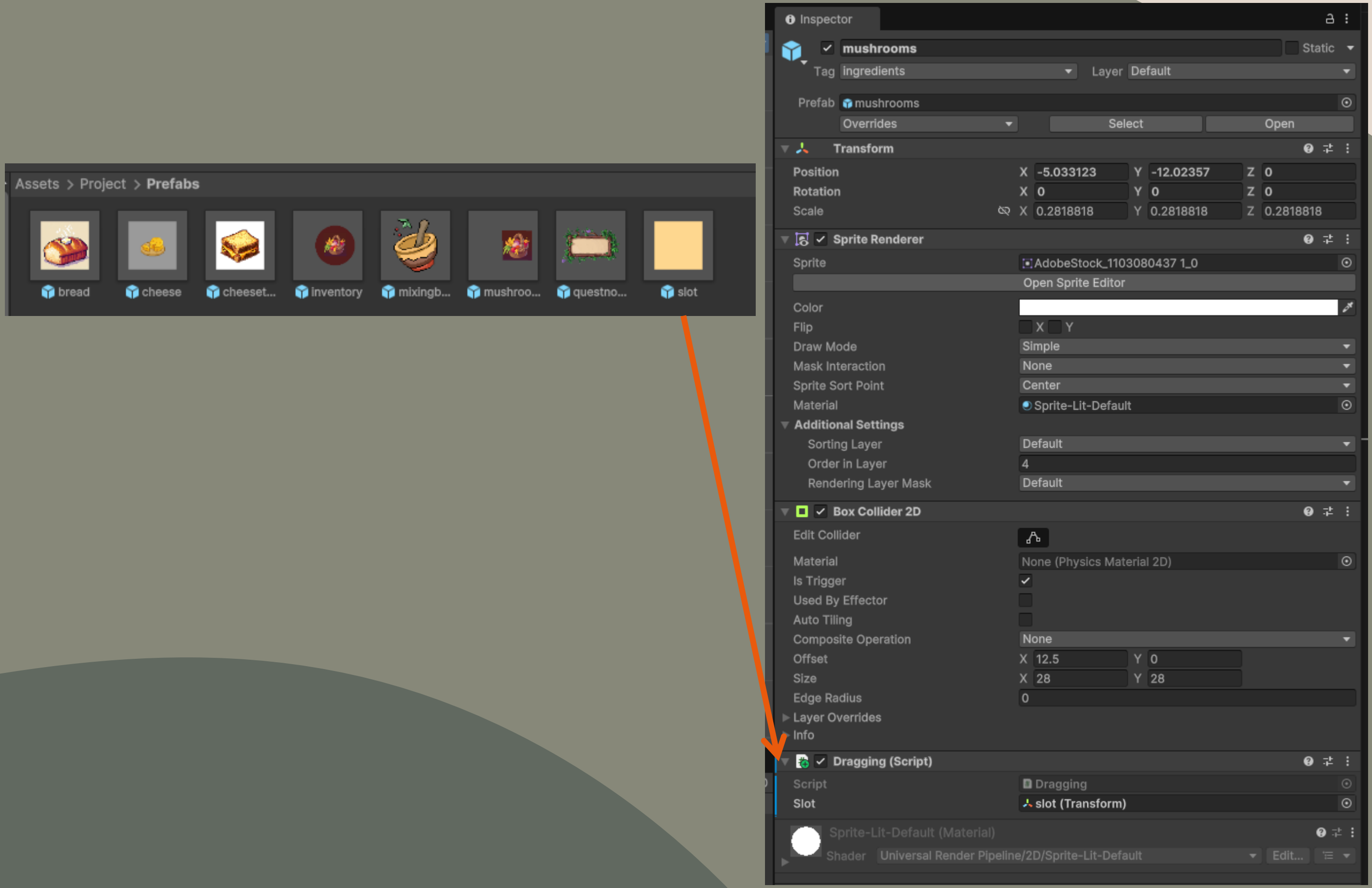
CONNECTING SCRIPT TO THE ITEMS

TESTING

Step 4: Connecting the script.

SLOT

- Finally, after saving your script.
- In the items that have the script:
  - Go to the project and drag the slot prefab we made previously to the Slot variable that we created that will be seen in the inspector.


Assets > Project > **Prefabs**

bread | cheese | cheeset... | inventory | mixingb... | mushroo... | questno... | slot

Inspector

mushrooms — Static
Tag: ingredients — Layer: Default
Prefab: mushrooms
Overrides | Select | Open

Transform

	X	Y	Z
Position	-5.033123	-12.02357	0
Rotation	0	0	0
Scale	0.2818818	0.2818818	0.2818818

Sprite Renderer

Sprite	AdobeStock_1103080437 1_0
	Open Sprite Editor
Color	
Flip	X Y
Draw Mode	Simple
Mask Interaction	None
Sprite Sort Point	Center
Material	Sprite-Lit-Default
Additional Settings	
Sorting Layer	Default
Order in Layer	4
Rendering Layer Mask	Default

Box Collider 2D

Edit Collider	
Material	None (Physics Material 2D)
Is Trigger	✓
Used By Effector	
Auto Tiling	
Composite Operation	None
Offset	X 12.5 Y 0
Size	X 28 Y 28
Edge Radius	0
Layer Overrides	
Info	

Dragging (Script)

Script	Dragging
Slot	slot (Transform)

Sprite-Lit-Default (Material)
Shader: Universal Render Pipeline/2D/Sprite-Lit-Default — Edit...

timeline

SETTING UP THE SCENE

CREATING THE ITEMS

CODING THE SCRIPT

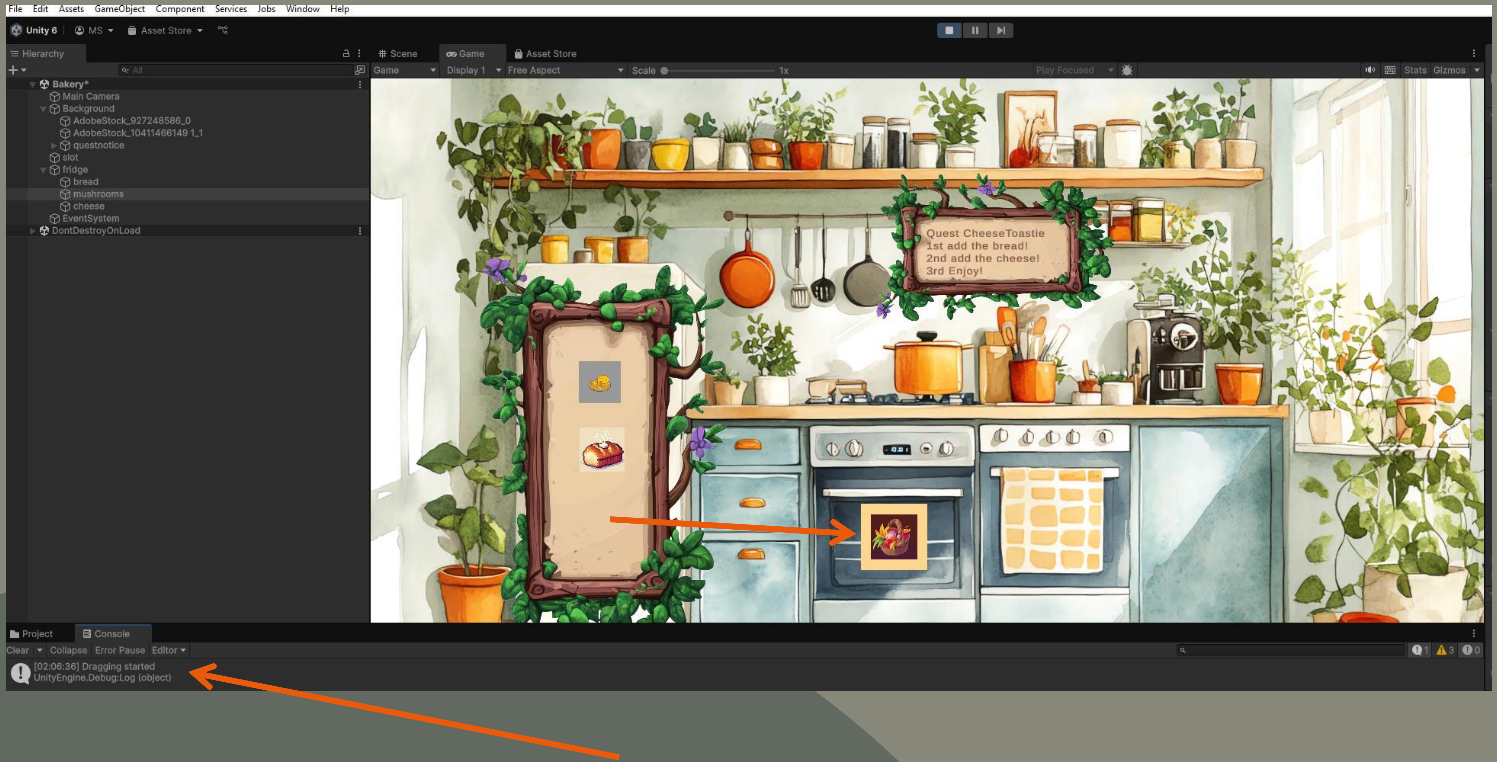
CONNECTING SCRIPT TO THE ITEMS

TESTING

Step 6: Testing.

PLAYING OUR GAME:

- Now let's test our work!
- If you click on an item with the script, it will follow your mouse, and you will now be able to drop it in the slot!
- Remember that when you drag it a log will still appear in the console it will read "Dragging started".



Quest CheeseToastie
1st add the bread!
2nd add the cheese!
3rd Enjoy!

```
[02:06:36] Dragging started
UnityEngine.Debug:Log (object)
```

timeline

SETTING UP THE SCENE

CREATING THE ITEMS

CODING THE SCRIPT

CONNECTING SCRIPT TO THE ITEMS

TESTING

Congratulations!

You now can drag and drop an item in unity!

Thank you