```
public class EnableObjectSelect : MonoBehaviour
{
	bool inRange;
	public float rotSpeed;

	float panX, panY;
	Vector3 baseRot;

	public Camera evidenceCam;
	Camera playerCam;

	// Use this for initialization
	void Start ()
	{

		playerCam = GameObject.FindGameObjectWithTag
		("PlayerCamera").GetComponent<Camera> ();
		evidenceCam.enabled = false;

		baseRot = transform.eulerAngles;

		panX = transform.eulerAngles.y;
		panY = transform.eulerAngles.x;

	}

	// Update is called once per frame
	void Update ()
	{

		if (inRange && Input.GetKeyDown (KeyCode.KeypadEnter)) {
			print ("Interacting With Things");
		}

		if (inRange) {

			if (Input.GetKey (KeyCode.Space))
			{
				evidenceCam.enabled = true;
				playerCam.enabled = false;

				if (gameObject.tag == "evidence") {

					transform.eulerAngles = new Vector3 (panY, panX,
					0);

					float rotX = Input.GetAxis ("Mouse X");
					float rotY = Input.GetAxis ("Mouse Y");

					panX += rotX * rotSpeed;
					panY -= rotY * rotSpeed;

				}
				}

			} else {
				playerCam.enabled = true;
				evidenceCam.enabled = false;

				if (gameObject.tag == "evidence") {

					transform.eulerAngles = baseRot;
					panX = transform.eulerAngles.y;
					panY = transform.eulerAngles.x;
				}
			}

	}

	void OnTriggerStay (Collider other)
	{

		if (other.gameObject.tag == "Player") {
			inRange = true;
		}
	}

	void OnTriggerExit (Collider other)
	{
		print ("Leave");
		inRange = false;
	}
}
```

What It Does:

Additional to the previous tutorial, this addition to the code will check the tag of a selected object. If the tag is “evidence”, the camera will change to the “evidence cam” and you will be able to use the mouse to rotate the object.

How To Do It:

- First add static variables to the top of your script (along with your “inRange” bool). The first one should be a public float called “rotSpeed”, you will use this later to set rotations on the object.
- Add two floats called “panX” and “panY”. These will be used later for object rotation.
- Add a Vector 3 called “baseRot”. You will use this to reset the rotation of the “evidence” object when you are done interacting with it.
- Set two cameras. One should be a public camera called “evidenceCam” and another should be a non-public camera called “playerCam”.
- Now set the various variables to their base states in void Start().
- To begin with set your playerCam variable to be your camera that is tagged “PlayerCamera” in your scene. This should look like this:
  playerCam = GameObject.FindGameObjectWithTag (“PlayerCamera”).GetComponent<Camera> ();
- Then you need to set your “evidenceCam” to false so that it is not active when the scene starts.
- Capturing the baseRot (base rotation) of an object is as simple as setting the baseRot variable to transform.eulerAngles;
- Doing the same as above for your panX and panY will set the x and y variables. The only difference in terms of code is that the transform.eulerAngles is followed by .x and .y respectively.

- Now that the variables all have a base state we can move to void Update() to have these affect the game.
- We need to set an if statement to check that the player model is both in range of the object and that the player is holding down the interact button (space). To do this we need to write: if (inRange){. This will check that the player is within range of the object's collider.
- We also need to write within the above if statement: if (Input.GetKey (KeyCode.Space)){. This checks that the player is pressing space.
- Within the above if statement we need to change the camera, so that if the player is within range of the "evidence" tagged object and is holding down space the camera is changed to the "evidenceCam". To do this we simply set the evidenceCam to = true and the playerCam to = false.
- Now we need to add another layer of if statement checking that the gameObject that is being interacted with has the "evidence" tag. To do this we use: if (gameObject.tag == "evidence"){. We use a double = so that we make sure the if statement knows that this means "the same as" so if the tag is the same as "evidence".
- Within this new layer of if statement we need to add our rotational transform code. To do this we set the transform with: transform.eulerAngles = new Vector3 (panY, PanX, 0);. This sets the rotational transform for the object to the variables panY and panX.
- Next we define a rotation. We add a float beneath the new Vector3 code that states that: rotX = Input.GetAxis ("Mouse X");. And do the same for rotY with "Mouse Y". What this does is state that the X and Y axis are set to the mouse's x and y positioning.
- Finally we add panX and panY definitions. Simply put, setting: panX += rotX * rotSpeed; allows the code to adjust the rotation of the object based on the rotation of the mouse. Doing this again with panY -= rotY * rotSpeed allows us to rotate the objects y axis in a non-inverted way with the mouse.
- Now we need to set a way to return to the player camera when we are not interacting with the object. After closing off the if statement chain above so that the if statement is closed up to the original if statement.
- We need to set an else statement. Within this else statement we set the player camera back to true and the evidence cam to false. Additionally, we add an if statement that says if the object is tagged as "evidence" then the transform.eulerAngles = baseRot. This resets the rotation, and by adding panX and panY = transform.eulerAngles.x and transform.eulerAngles.y respectively this resets the variables to their base state.
- Now we need to add a state that returns the player to the playerCamera if they move away from the object whilst still pressing space. Adding things like this is necessary to make sure that players are not punished for not doing everything exactly correctly.
- Closing off the if statements back down to the base if statement that checks whether the player is inRange. Then we add the else statement that checks whether the evidenceCam is active when the player is not in range and sets it to false and the playerCam to true.