```
/*
DragRigidbodyUse.cs ver. 11.1.16 - wirted by ThunderWire Games * Script for Drag & Drop & Throw Objects & Draggable Door & PickupObjects
*/
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
[System.Serializable]
public class GrabObjectClass
{
public bool m_FreezeRotation;
public float m_PickupRange = 3f;
public float m_ThrowStrength = 50f;
public float m_distance = 3f;
public float m_maxDistanceGrab = 4f;
}
[System.Serializable]
public class ItemGrabClass
{
public bool m_FreezeRotation;
public float m_ItemPickupRange = 2f;
public float m_ItemThrow = 45f;
public float m_ItemDistance = 1f;
public float m_ItemMaxGrab = 2.5f;
}
[System.Serializable]
public class DoorGrabClass
{
public float m_DoorPickupRange = 2f;
public float m_DoorThrow = 10f;
public float m_DoorDistance = 2f;
public float m_DoorMaxGrab = 3f;
}
[System.Serializable]
public class TagsClass
{
```

```
public string m_InteractTag = "Interact";
public string m_InteractItemsTag = "InteractItem";
public string m_DoorsTag = "Door";
}
public class DragRigidbodyUse : MonoBehaviour
{
public Camera PlayerCamActualCamera;
public GameObject playerCam;
public string GrabButton = "Grab";
public string ThrowButton = "Throw";
public string UseButton = "Use";
public GrabObjectClass ObjectGrab = new GrabObjectClass();
public ItemGrabClass ItemGrab = new ItemGrabClass();
public DoorGrabClass DoorGrab = new DoorGrabClass();
public TagsClass Tags = new TagsClass();
private float PickupRange = 3f;
private float ThrowStrength = 50f;
private float distance = 3f;
private float maxDistanceGrab = 4f;
private Ray playerAim;
private GameObject objectHeld;
private bool isObjectHeld;
private bool tryPickupObject;

void Start()
{
isObjectHeld = false;
tryPickupObject = false;
objectHeld = null;
}
void FixedUpdate()
{

if (Input.GetButton(GrabButton))
{
if (!isObjectHeld)
{
tryPickObject();
tryPickupObject = true;
```

```
}
else
{
holdObject();
}
}
else if (isObjectHeld)
{
DropObject();
}
if (Input.GetButton(ThrowButton) && isObjectHeld)
{
isObjectHeld = false;
objectHeld.GetComponent<Rigidbody>().useGravity = true;
ThrowObject();
}
if (Input.GetButton(UseButton))
{
isObjectHeld = false;
tryPickObject();
tryPickupObject = false;
Use();
}
}
private void tryPickObject()
{
Ray playerAim = playerCam.GetComponent<Camera>().ViewportPointToRay(new
Vector3(0.5f, 0.5f, 0));
RaycastHit hit;
if (Physics.Raycast(playerAim, out hit, PickupRange))
{
objectHeld = hit.collider.gameObject;
if (hit.collider.tag == Tags.m_InteractTag && tryPickupObject)
{
isObjectHeld = true;
objectHeld.GetComponent<Rigidbody>().useGravity = false;
if (ObjectGrab.m_FreezeRotation)
{
objectHeld.GetComponent<Rigidbody>().freezeRotation = true;
```

```
}
if (ObjectGrab.m_FreezeRotation == false)
{
objectHeld.GetComponent<Rigidbody>().freezeRotation = false;
}
/**/
PickupRange = ObjectGrab.m_PickupRange;
ThrowStrength = ObjectGrab.m_ThrowStrength;
distance = ObjectGrab.m_distance;
maxDistanceGrab = ObjectGrab.m_maxDistanceGrab;
}
if (hit.collider.tag == Tags.m_InteractItemsTag && tryPickupObject)
{
isObjectHeld = true;
objectHeld.GetComponent<Rigidbody>().useGravity = true;
if (ItemGrab.m_FreezeRotation)
{
objectHeld.GetComponent<Rigidbody>().freezeRotation = true;
}
if (ItemGrab.m_FreezeRotation == false)
{
objectHeld.GetComponent<Rigidbody>().freezeRotation = false;
}
/**/
PickupRange = ItemGrab.m_ItemPickupRange;
ThrowStrength = ItemGrab.m_ItemThrow;
distance = ItemGrab.m_ItemDistance;
maxDistanceGrab = ItemGrab.m_ItemMaxGrab;
}
if (hit.collider.tag == Tags.m_DoorsTag && tryPickupObject)
{
isObjectHeld = true;
objectHeld.GetComponent<Rigidbody>().useGravity = true;
objectHeld.GetComponent<Rigidbody>().freezeRotation = false;
/**/
PickupRange = DoorGrab.m_DoorPickupRange;
ThrowStrength = DoorGrab.m_DoorThrow;
distance = DoorGrab.m_DoorDistance;
maxDistanceGrab = DoorGrab.m_DoorMaxGrab;
```

```
}
}
}
private void holdObject()
{
Ray playerAim = playerCam.GetComponent<Camera>().ViewportPointToRay(new
Vector3(0.5f, 0.5f, 0));
Vector3 nextPos = playerCam.transform.position + playerAim.direction * distance;
Vector3 currPos = objectHeld.transform.position;
objectHeld.GetComponent<Rigidbody>().velocity = (nextPos - currPos) * (10 -
objectHeld.GetComponent<Rigidbody>().mass);
if (Vector3.Distance(objectHeld.transform.position, playerCam.transform.position) >
maxDistanceGrab)
{
DropObject();
}
}
private void DropObject()
{
isObjectHeld = false;
tryPickupObject = false;
objectHeld.GetComponent<Rigidbody>().useGravity = true;
objectHeld.GetComponent<Rigidbody>().freezeRotation = false;
objectHeld = null;
}
private void ThrowObject()
{
objectHeld.GetComponent<Rigidbody>().AddForce(playerCam.transform.forward *
ThrowStrength);
objectHeld.GetComponent<Rigidbody>().freezeRotation = false;
objectHeld = null;
}
private void Use()
{
objectHeld.SendMessage("UseObject", SendMessageOptions.DontRequireReceiver); //Every
script attached to the PickupObject that has a UseObject function will be called.
objectHeld = null;
}
}
```