VR Shooter Kit 1.3

Hello and thanks for your purchase of **VR Shooter Kit**.

If you have any question, please let me know **james@unity3dninja.com**

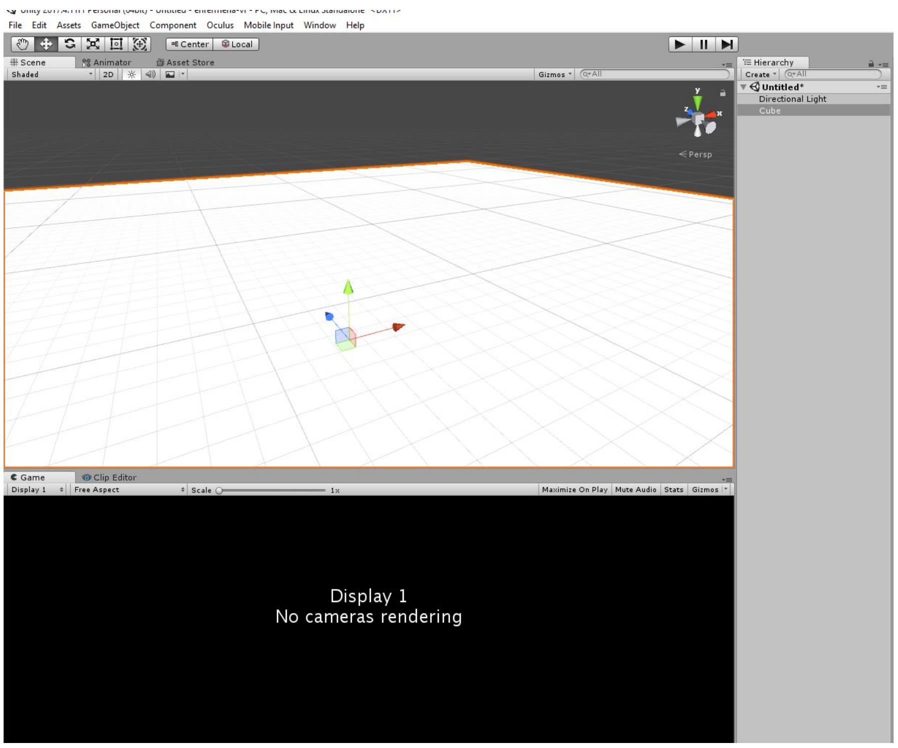
VR Shooter Kit is an asset that facilitates the development of **VR** games, with lots of scripts and examples, so you can build your own games.

All the code included is commented so if you want to understand how it works you just have to open any script and start reading a bit, although in this document I will try to explain the most relevant parts of it.

Quick Start

It is best to start playing with the demo scene, so you can get a little familiar with this asset.

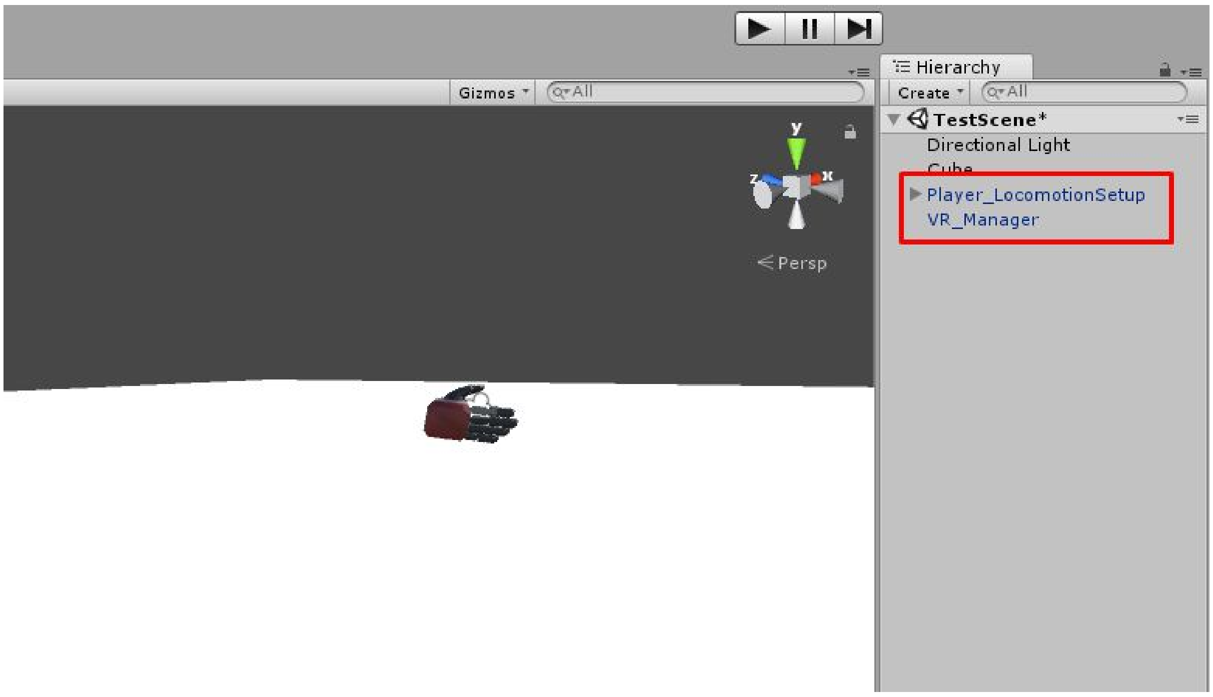
If you want to create your own game, you just have to create a new scene and make sure there is no any camera.



Display 1
No cameras rendering

I have created a cube that will serve as the ground, then you must drag a prefab for the player, you can choose **Player_LocomotionSetup** or **Player_TeleportSetup**, from the **VRShooterKit/ Prefabs / PlayerSetup folder.**

Then you just have to drag the **VR_Manager** from **VRShooterKit/ Prefabs / Main.**



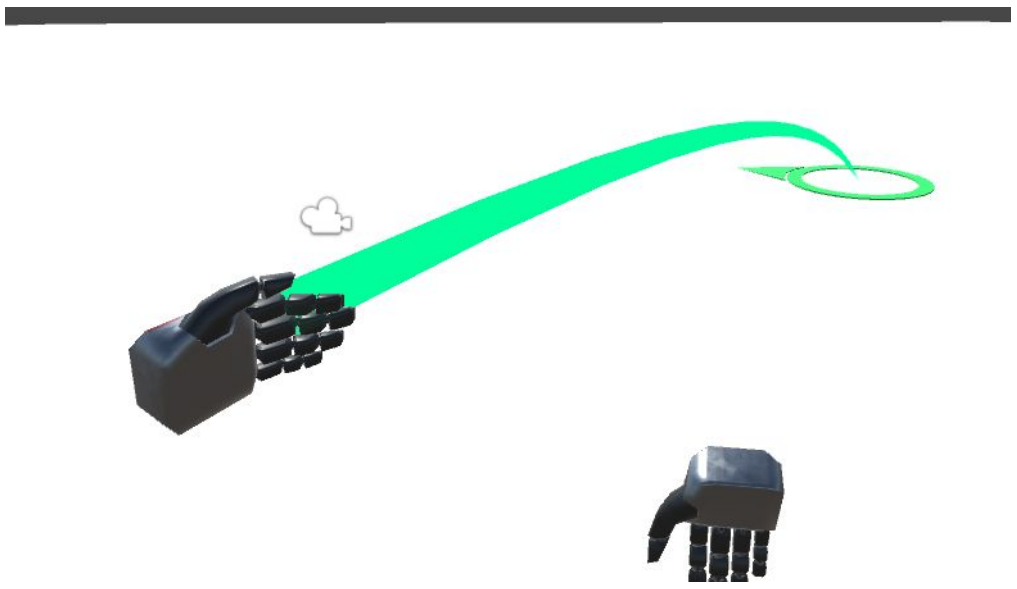
And we are ready.

Player_LocomotionSetup Controls

The player locomotion setup, use the OVRPlayerController.cs of the Oculus Integration, so the controls here are the standards, with the left control you can move around and with the right you can turn, with the trigger button on the left control you can run.

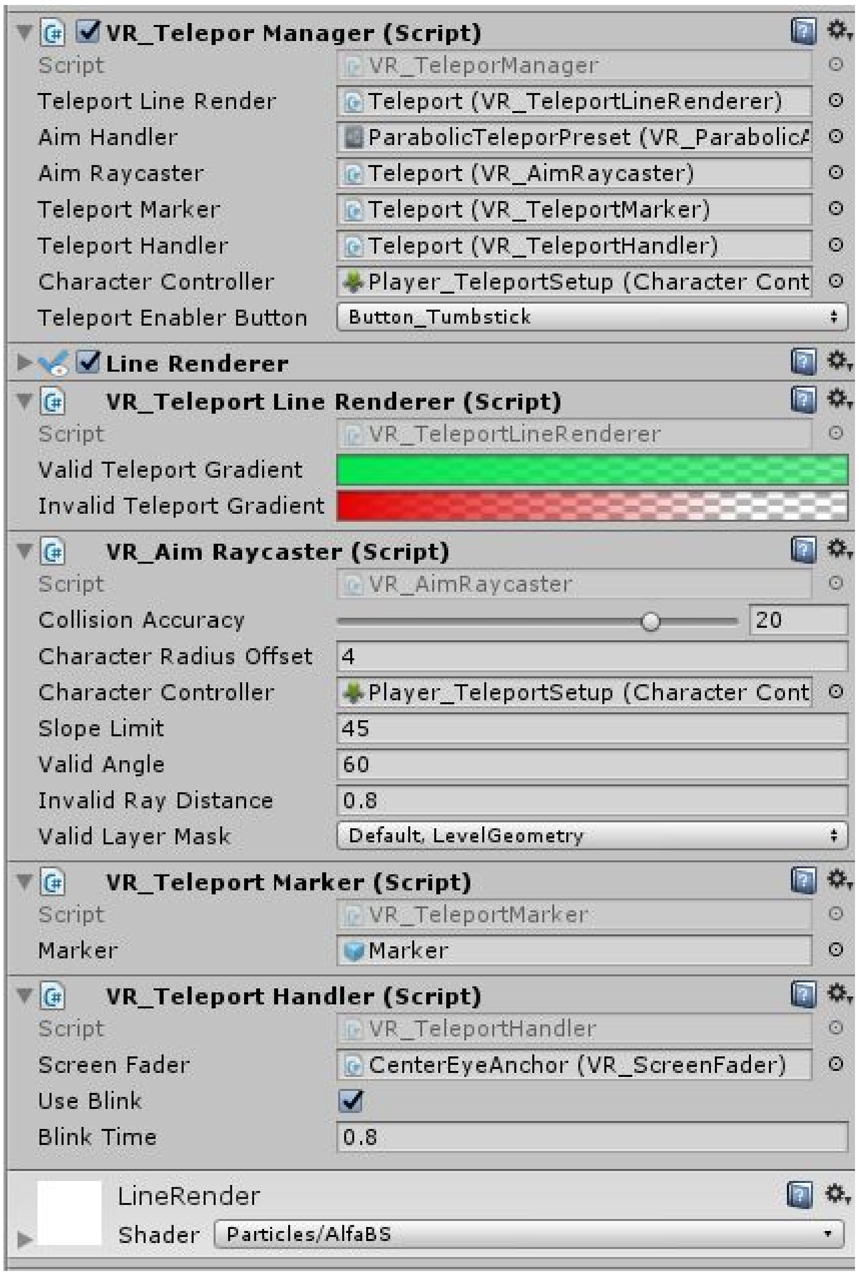
Controles Player_TeleportSetup

The player teleport setup, use the teleport mechanics that you could see in games like robo recall, you can teleport with any control just move the joystick, and you can see the line along with the marker, which indicates your final position, to complete the movement is enough to bring the joystick to its initial position.



You can move the joystick around to indicate the rotation you want, and move your hand to indicate the position.

The teleportation system is quite configurable and can be easily extended, just select the teleport object to see all the options we have.



VR_Telepor Manager (Script)

Script	VR_TeleporManager
Teleport Line Render	Teleport (VR_TeleportLineRenderer)
Aim Handler	ParabolicTeleporPreset (VR_Parabolic
Aim Raycaster	Teleport (VR_AimRaycaster)
Teleport Marker	Teleport (VR_TeleportMarker)
Teleport Handler	Teleport (VR_TeleportHandler)
Character Controller	Player_TeleportSetup (Character Cont
Teleport Enabler Button	Button_Tumbstick

Line Renderer

VR_Teleport Line Renderer (Script)

Script	VR_TeleportLineRenderer
Valid Teleport Gradient	🟩
Invalid Teleport Gradient	🟥

VR_Aim Raycaster (Script)

Script	VR_AimRaycaster
Collision Accuracy	20
Character Radius Offset	4
Character Controller	Player_TeleportSetup (Character Cont
Slope Limit	45
Valid Angle	60
Invalid Ray Distance	0.8
Valid Layer Mask	Default, LevelGeometry

VR_Teleport Marker (Script)

Script	VR_TeleportMarker
Marker	Marker

VR_Teleport Handler (Script)

Script	VR_TeleportHandler
Screen Fader	CenterEyeAnchor (VR_ScreenFader)
Use Blink	✓
Blink Time	0.8

LineRender

Shader	Particles/AlfaBS

VR_Grabbable.cs

Perhaps the most important script in this asset, as you probably know you can add this script to any object that the player can take.

bool SetJointSettings: indicates if we want to configure the joint used to grab objects, a **FixedJoint** is currently used, enable the **JointBreakTorque** and **JointBreakForce** fields.

(optional) float JointBreakTorque: Indicates the force that must be applied to break the joint.
(optional) float JointBreakForce: Indicates the force that must be applied to break the joint.

float InteractDistance: Indicates the distance at which this object can be taken.

bool PerfectGrab: indicates if the object uses perfect grab, the object will not move to the grap point, enable the **ShouldFly** and **AutoGrab** fields.

bool UsePerHadSettings: Indicate if we want to use different configurations for each hand, enable the **RightHandSettings** and **LeftHandSettings** fields.

VR_HandInteractSettings HandSettings: Global configuration for both hands.

- **Transform InteractPoint:** Point from which the object is grabbed.
- **Transform HighlightPoint:** Point at which it indicates that the hand can take the object, using the **VR_OutlineHighlight** or **VR_UIHighlight**.
- **Vector3 RotationOffset:** Indicates the offset of the rotation, when taking the object.
- **AnimationClip Animation:** animation for the hand when we take the object.
- **bool CanInteract:** Indicates if this hand can take the object.
- **bool HideHandOnGrab:** Indicates whether the hand should be hidden when taking the object.

(optional)VR_HandInteractSettings RightHandSettings: configuration for the right hand.

(optional)VR_HandInteractSettings LeftHandSettings: configuration for the lefthand.

bool ShouldFly: Indicates if when taking the object it should move towards the hand, enable the **GrabFlyTime** field.
(optional) float GrabFlyTime: Indicates the time it will take for the object to reach the hand.

(optional) bool AutoGrab: Indicates whether this object should be grabbed automatically, enable the **StartOnRightHand** and **StartOnLeftHand** fields.

(optional) button StartOnRightHand: Indicates whether the object should start in the right hand.

(optional) button StartOnLeftHand: Indicates whether the object should start in the left hand.

bool EnableColliderOnGrab: Indicates if at the time of grabbing the object, we want to have the collider activated.
VR_Button InteractButton: button to press to start the interaction.

int GrabLayer: Indicates the layer to use when an object is grabbed.
int UnGrabLayer: Indicates the layer to use when the object is released.

bool preserveKinematicState: Indicates if the kinematic state should change after dropping the object.

UnityEvent OnGrabStateChangeEvent: Event that is called when the object's grab status changes.

To create an object that can be grabbed, just add this script to any object.

VR_Weapon.cs

This script allows you to create weapons within the game.

enum ReloadMode ReloadMode: Indicates the way in which the reloading of weapons is processed.

- **ReloadMode.Physics:** Indicates that the weapon must be loaded by some gesture, such as the revolver in the example scene.
- **ReloadMode.Realistic:** Indicates that the weapon must be loaded by changing the weapon's magazine, for one that still has ammunition.
- **ReloadMode.UI:** Indicates that the weapon will reload automatically, showing a small bar indicating the time it will take to reload.
- **ReloadMode.InfiniteBullets:** Indicates that the weapon never needs to be reloaded.

(optional) WeaponUI WeaponUI: Component used to indicate the amount of ammo remaining, and the loading bar that is only used in **ReloadMode.UI.**

(optional) BarrelScript BarrelScript: Component used in the revolver reload animation, only required in **ReloadMode.Physics.**

(optional) float ReloadAngle: Hand angle to start a recharge, only used in **ReloadMode.Physics.**

(optional) enum WeaponTag WeaponTag: Enum that indicates what type of magazine this weapon can accept, only required in **ReloadMode.Realistic.**

(optional) Transform MagazineSnapPoint: Point at which the recharge is placed, only required in **ReloadMode.Realistic.**

(optional) int ClipSize: Charger size, only required in **ReloadMode.UI**.

(optional) float ReloadTime: time it takes to recharge, only required in **ReloadMode.UI.**

Transform ShootPoint: point from which the bullets fire.
Bullet BulletPrefab: bullet used by the weapon.
WeaponHammer WeaponHammer: component that simulates the use of a hammer in weapon.
ShellEjector ShellEjector: Component that handles the ejection of caps in the weapon.
float ShootRate: Indicates the time that must pass between each shot.
Isolutatic bool: indicates if the weapon is automatic.
float bulletSpeed: speed of the **bullet** when fired.
int HitLayer: indicates which layers the bullet can hit.
int MaxBulletBounceCount: indicates how many times a **bullet** can bounce on a surface.
float Dmg: indicates the damage produced by each bullet.

float MinHitForce: minimum impact applied to the object hitting the bullet.
float MaxHitForce: maximum impact applied to the object hitting the bullet.
float Range: maximum distance the bullet can travel.
AudioClip ShootSound: Sound for shooting.
GameObject MuzzleFlash: Effect for shooting.
bool ParentMuzzleFlash: Indicates whether the shooting effect should be a child of the shootPoint.
bool DisableMuzzleWhileNoShooting: Indicates whether the muzzle effect should be deactivated while not firing.
VR_Button FireButton: button that starts the shooting action.

float MinRecoilPositionForce: Indicates the minimum force of movement, which the weapon exerts when firing.
float MaxRecoilPositionForce: Indicates the maximum force of movement, which the weapon exerts when firing.
float RecoilPositionLimit: maximum accumulated force of movement, which the weapon can exert.

float MinRecoilRotationForce: Indicates the minimum rotational force, which the weapon exerts when firing.
float MaxRecoilRotationForce: Indicates the maximum rotational force, which the weapon exerts when firing.
float RecoilAngleLimit: Maximum accumulated angle, which a weapon can exert when shooting.

bool UseSpread: Indicates whether the bullet should be fragmented, normally used in shotguns, enable the **MinSpreadCount**, **MaxSpreadCount**, **MinSpreadAngle** and **MaxSpreadAngle** fields

int MinSpreadCount: minimum number of fragments.
int MaxSpreadCount: maximum number of fragments.
float MinSpreadAngle: minimum angle that each fragment will have.
float MaxSpreadAngle: minimum angle that each fragment will have.

float PositionLerpSpeed: value applied to return to the initial position.
float RotationLerpSpeed: value applied to return to the initial rotation.

VR_Lever

float InteractDistance: Indicates the distance at which this object can be taken.
bool UsePerHadSettings: Indicates if we want to use different configurations for each hand, enable the RightHandSettings and LeftHandSettings fields.

VR_HandInteractSettings HandSettings: Global configuration for both hands.
- **Transform InteractPoint:** Point from which the object is grabbed.
- **Transform HighlightPoint:** Point at which it indicates that the hand can take the object, using the **VR_OutlineHighlight** or **VR_UIHighlight**.
- **Vector3 RotationOffset:** Indicates the offset of the rotation, when taking the object.
- **AnimationClip Animation:** animation for the hand when we take the object.
- **bool CanInteract:** Indicates if this hand can take the object.
- **bool HideHandOnGrab:** Indicates whether the hand should be hidden when taking the object.

(optional) VR_HandInteractSettings RightHandSettings: configuration for the right hand.

(optional) VR_HandInteractSettings LeftHandSettings: configuration for the left hand.

VR_Button InteractButton: button that must be pressed to start the interaction.

Transform TransformBase: lever base to ignore collisions, and get direction.

int SolverIteractions: Indicates the number of SolverIteractions, more information here.

bool ShouldBackToStartingPosition: Indicates whether the lever returns to its initial position.
float BackForce: force with which the lever returns to its initial position.

UnityEvent OnValueChange: Event that is called when the lever changes position, 0 indicates that the lever is in the initial position, 1 that the lever is in its final position.

UnityEvent OnGrabStateChangeEvent: Event that is called when the object's grap status changes.

VR_DropZone

enum DropZoneMode DropZoneMode: Indicates the way in which the VR_DropZone works.

- **VR_DropZoneMode.Collider:** the objects that come into contact with the collider will activate the dropzone.
- **VR_DropZoneMode.Distance:** objects that are in the range will activate the dropzone.

Transform DropPoint: Indicates the position and final rotation of the objects that enter the DropZone.

VR_Grabbable StartingDrop: indicates if there is an object initially in the DropZone.

Collider [] DropZoneColliderArray: Indicates the collider that the DropZone will use.
float DropRadius: Indicates the minimum distance at which objects can activate the DropZone.

bool ShouldFly: Indicates whether the object should move towards the DropPoint.
bool SyncronizePosition: Indicates whether the DropZone should modify the position of the objects.
bool SyncronizeRotation: Indicates whether the DropZone should modify the rotation of the objects.
bool UsePreview: Indicates if the DropZone will use a preview.

UnityEvent OnDropStateChange: Event called when an object is removed, or added to the DropZone.

VR_Button

List <Collider> IgnoreColliderList: Indicates that colliders should ignore the button.
Transform ObstacleCastPoint: Point from which this button should search for obstacles.
float ObstacleCastRadius: Radius in which you must search for obstacles.
float PressThreshold: distance that this button must travel to reach its final position.
float PressTime: Time it takes for this button to reach the pressed position.
Vector3 PressingDir: Direction to which the button is pressed.