# Mobile VR Movement Pack V1.1

**Table of Contents:**

Introduction
    Video Tutorials
What is Mobile VR Movement Pack?
How To Use
    Autowalk
    Advanced Walk
    AutoFly
    Lookwalk
    NavMesh Movement:
    Bluetooth Controller Movement
    Waypoint System
Target Platforms
    Gear VR
    Google VR (Cardboard / Daydream)
Further Information

## Introduction

Thanks for the purchase and support! We are a community of VR devs, working together to create games, experiences, development tools, and tutorials in an effort to empower emerging VR developers worldwide. Join us here: https://www.youtube.com/nurfacegames/

### Video Tutorials

***Intro:*** https://www.youtube.com/watch?v=INpssimg588
***Tutorial:*** https://www.youtube.com/watch?v=a0F3eOVQ1bk
***GoogleVR Integration:*** https://www.youtube.com/watch?v=-qEqvIjRulU

For any questions or support, please email:
nurfacegames@gmail.com

## What is Mobile VR Movement Pack?

This is a collection of movement examples that work for mobile VR. The pack currently includes 7 demos: Waypoint System, NavMesh Movement, Bluetooth Controller Movement, Autowalk, AdvancedWalk, Autofly, and Lookwalk .

## How To Use

Each Camera Prefab includes a script called *'VRMouseLook'*. VRMouseLooks lets you easily rotate and tilt the camera inside of Unity Editor.

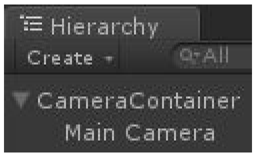
- Hold "**ALT**" to rotate the head.
- Hold "**CTRL**" to tilt the head.

Tags should be properly assigned to both the camera and the player. Please assign the following tags when doing custom development:

- The main VR camera has tag: **Main Camera**
- The VR player has tag: **Player**

In Unity VR development, you cannot move the camera directly. Instead, the camera must be a child of another GameObject, and changes to the position must be applied to the parent's Transform. With this in mind, remember this rule:

- **The VR Main Camera must always be a child of a root GameObject**



More info:
https://unity3d.com/learn/tutorials/topics/virtual-reality/movement-vr

## Autowalk

Tap button to start walking; Tap button to stop walking.

- Add script *VRAutowalk* & a *CharacterController* to the root player gameobject.
- Scene must have an *EventSytem* with a *GazeInputModule* above a *StandAloneInputModule*.
- **Scripts used:** VRAutowalk.cs

## Advanced Walk

Tap button to start & stop walking. Look up to jump. When looking at an interactive object, do not allow player to trigger movement.

- Add script VRAdvancedWalk & a *CharacterController* to the root player gameobject.
- Scene must have an *EventSytem* with a *GazeInputModule* above a *StandAloneInputModule*.
- **Scripts used:** VRAdvancedWalk.cs

## AutoFly

Tap button to start flying; Tap button to stop flying.

- Add script *VRAutofly* & a *CharacterController* to the root player gameobject.
- Scene must have an *EventSytem* with a *GazeInputModule* above a *StandAloneInputModule*.
- **Scripts used:** VRAutofly.cs

## Lookwalk

Look down to start walking; Looking back up to stop walking.

- Add script *VRLookWalk* & a *CharacterController* to the root player gameobject.
- Scene must have an *EventSytem* with a *GazeInputModule* above a *StandAloneInputModule*.
- **Scripts used:** VRLookwalk.cs

## NavMesh Movement:

If they player looks at a terrain, floor, or ground and taps an input button, the will move to the location they were looking at. How it works:

- A NavMesh is baked into all objects marked Navigation Static. See Window > Navigation. Objects that can be walked on (Terrain, ground mesh, etc) must have script *VRWalkableSurface* added and a collider component.
- Root Player Gameobject must have a *NavMesh Agent* component.
- Main Camera must have a *Physics Raycaster* component.
- Scene must have an *EventSytem* with a *GazeInputModule* above a *StandAloneInputModule*.

- **Scripts used:** VRWalkableSurface.cs

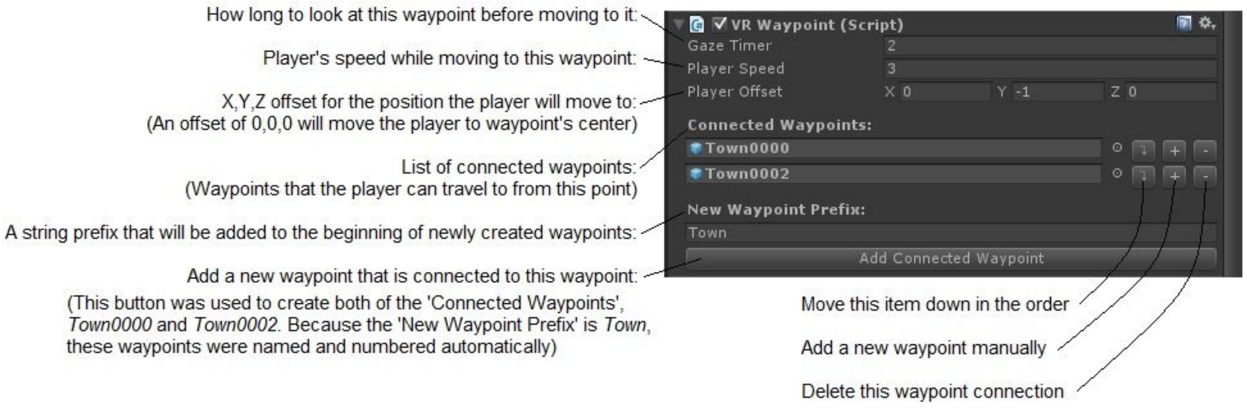
## Bluetooth Controller Movement

A Bluetooth controller MAY be used to move the player. This is not a comprehensive VR bluetooth controller and will not work with every Bluetooth controller. It should work with most basic brands. It takes input from the horizontal and vertical inputs. See *Edit* > *Project Settings* > *Input.*

- Add script *VRBluetoothController* & a *CharacterController* to the root player gameobject.
- Scene must have an *EventSytem* with a *GazeInputModule* above the
- **Scripts used:** VRBluetoothController.cs

## Waypoint System

They waypoints have a custom editor so that creating waypoint paths should be very easy. The waypoint system is explained here:

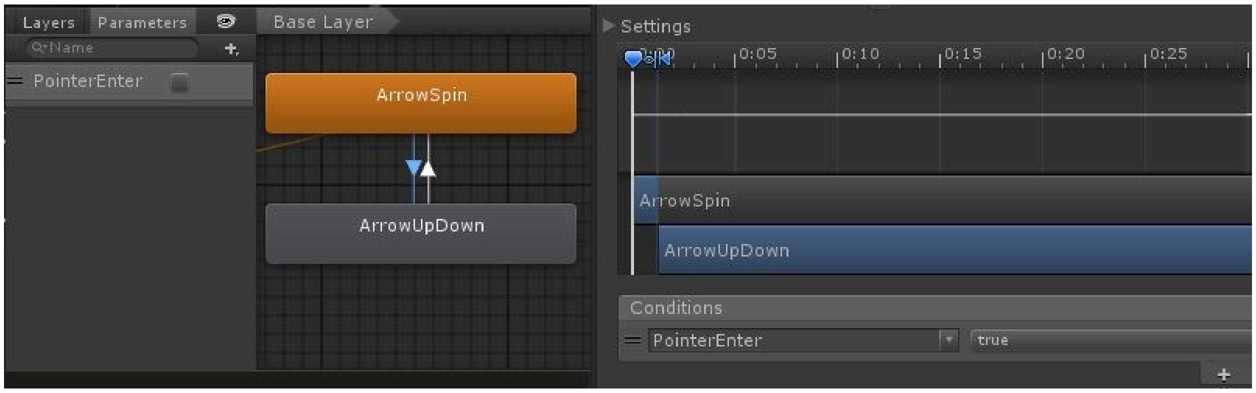


VR Waypoint (Script)
Gaze Timer 2
Player Speed 3
Player Offset X 0 Y -1 Z 0
Connected Waypoints:
Town0000
Town0002
New Waypoint Prefix:
Town
Add Connected Waypoint

How long to look at this waypoint before moving to it:
Player's speed while moving to this waypoint:
X,Y,Z offset for the position the player will move to:
(An offset of 0,0,0 will move the player to waypoint's center)
List of connected waypoints:
(Waypoints that the player can travel to from this point)
A string prefix that will be added to the beginning of newly created waypoints:
Add a new waypoint that is connected to this waypoint:
(This button was used to create both of the 'Connected Waypoints', *Town0000* and *Town0002*. Because the 'New Waypoint Prefix' is *Town*, these waypoints were named and numbered automatically)
Move this item down in the order
Add a new waypoint manually
Delete this waypoint connection

A Waypoint Prefab in included in the *VRMovementPack/Prefabs/* folder. A Waypoint requires the following components:

- A custom tag must be applied to the waypoint: **Waypoint**
- A *VRWaypoint.cs* script.
- A collider component (sphere is best).
- A visual component (arrow) which is a child GameObject of the Waypoint itself.
- The child may have an *Animator* component, and the code uses a boolean parameter on the animation controller named “PointerEnter” to toggle between animations:



- The Main Camera must have a *Physics Raycaster* component.
- Scene must have an *EventSytem* with a *GazeInputModule* above the *StandAloneInputModule*.
- **Scripts used:**
  Scripts\VRWaypoint.cs
  Scripts\Waypoint.cs
  Scripts\Editor\VRWaypointEditor.cs
  Scripts\Editor\WaypointDrawer.cs
  Scripts\Editor\WaypointList.cs

## Target Platforms

### Gear VR

This pack is ready to use with native Unity VR platforms such as GearVR. Simply select the Virtual Reality Supported option in the build settings and you are ready to build.

- More info: https://docs.unity3d.com/Manual/VROverview.html
- For GearVR, remember the device signature file: https://developer.oculus.com/osig/

### Google VR (Cardboard / Daydream)

GoogleVR will soon be a native Unity VR platform, at that point this pack will automatically work. For now, there are a couple extra steps to take when working with GoogleVR/Cardboard/Daydream.

**Step 1: Download Google VR 0.9 or Current Version**
Get the Google VR SDK 0.9 as we will use the '*GVRViewerMain*' prefab.
https://developers.google.com/vr/unity/download

**Step 2: Import Google VR SDK**

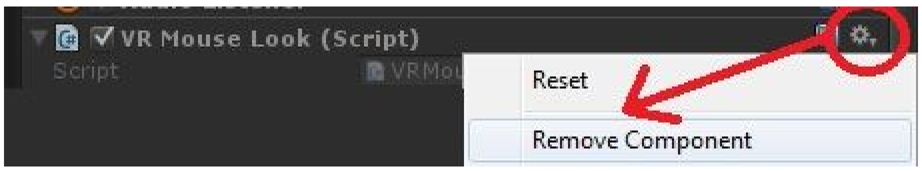
Import the Google VR SDK by opening the file ‘*GoogleVRForUnity.unitypackage’* from the previous download.

**Step 3: Import the Mobile VR Movement Pack**
Import this asset into the project and select all files.

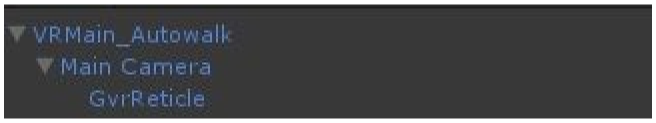
**Step 4: Open the Autowalk Demo Scene**
Navigate to ‘*VRMovementPack/Demos/DemoAutoWalk.unity’* and open the scene. We will use the Autowalk scene as an example.

**Step 5: Add *GVRViewMain* Prefab to the Scene**
Navigate to *‘GoogleVR/Prefabs/GVRViewerMain.prefab’* and drag this prefab into the scene.

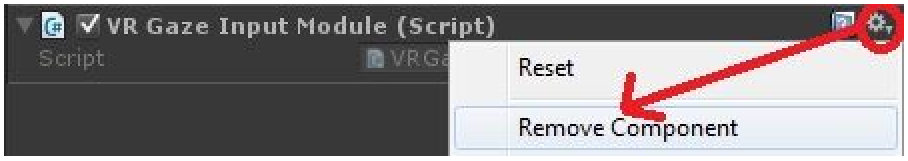
**Step 6: Remove *VRMouseLook* script from Main Camera**
Because GoogleVR rotates and tilts head with Alt and Ctrl, we remove the *VRMouseLook* script from the main camera:



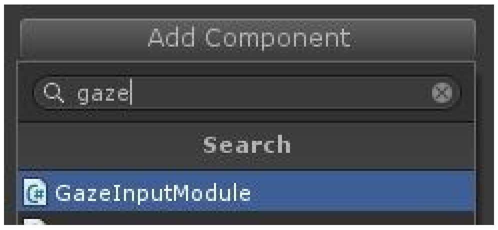
**Step 7: Add *GvrReticle* prefab to Main Camera**
Navigate to *‘GoogleVR/Prefabs/UI/GvrReticle.prefab’* and drag this prefab onto ‘*Main Camera*’:



**Step 8: Remove *VRGazeInputModule* Component**
The demos use a basic input module named *VRGazeInputModule*. Select the gameobject named *‘EventSystem’* and remove *VRGazeInputModule* from it:

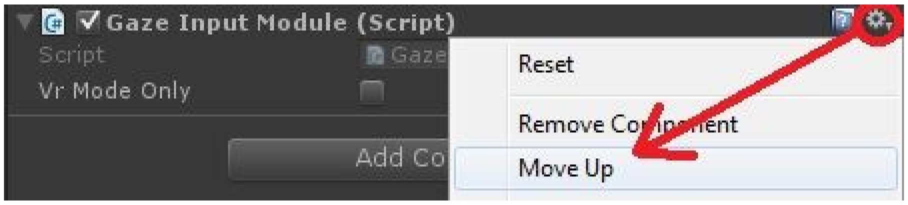


**Step 9: Add Google VR’s *GazeInputModule***
Now we want to add Google’s input module. Select ‘*Add Component*’ and add a ‘*GazeInputModule*’ (No ‘VR’ at the beginning of the name):



**Step 10: Move GazeInputModule Up**

Select the gear icon in the inspector on the GazeInputModule class and select '*Move Up*':



## Further Information

For a video tutorials related to this asset, please click here:
***Intro:*** https://www.youtube.com/watch?v=lNpssimg588
***Tutorial:*** https://www.youtube.com/watch?v=a0F3eOVQ1bk
***GoogleVR Integration:*** https://www.youtube.com/watch?v=-qEqvIjRuIU

Join our VR community here for VR tutorials and videos:
https://www.youtube.com/nurfacegames/

For any questions or support, please email:
nurfacegames@gmail.com