World Space Crosshair is a Unity plugin that is exactly what the name describes: a crosshair that exists in the 3D world space with the rest of your game, rather than as a screen space GUI piece. This is necessary because 2D screen space crosshairs are completely unusable in VR. Of course, this 3D crosshair can be useful for non-VR games, too, if your project needs it.

As you'll see below, it has a very quick set up and lots of options to help customize your cursor. And like all of our plugins, we include the well-commented source code for free, too.

Check out the DemoScene in the Demo folder for an example implementation (some demo features are unavailable without Unity Pro, but the plugin itself does NOT require Unity Pro).

Planned Features

The following features are planned but not yet implemented for the World Space Crosshair plugin. Please let us know on the Unity forum thread if you would like to see additional features not listed here or if you would like us to prioritize certain features.

1) Optional boundaries for the crosshair, allowing the crosshair to prevent itself from traveling offscreen or outside a certain range.

2) UI activation capabilities, allowing the crosshair to interact with the standard Unity UI selectables such as buttons or check boxes. Note that this will likely require users to purchase our World Space Cursor plugin as well.

3) Custom inspector for the crosshair component, which will simplify the integration process further and suggest certain ranges for the values of the customization options.

4) Ignore layers, which will allow the crosshair to ignore certain layers. Would be useful for windows, transparent objects, UI, etc.

Quick Set-up guide

For an easy set-up, we have included two prefabs that can be used to immediately get your World Space Crosshair up and running in your project: the Crosshair Projector and the Independent Crosshair. After choosing the right prefab for your project, just drag it into your hierarchy and customize the options (see below for more info on what the customization options do).

1) The Crosshair Projector consists of two nested objects: the parent projector and the child crosshair. This prefab should be used when you don't have your own way of moving the crosshair around. The projector base will rotate around with the mouse by default, which will move the crosshair itself around as its child. For more info on how this works, please refer to CrosshairProjector.cs and the next section (Summary).

2) The Independent Crosshair is a single prefab that will allow you to place a 3D crosshair in your project without using the pre-made projector. Most people will want to use this option, but it requires a little more forethought and understanding of how the plugin works. The Independent Crosshair will need to be attached to a different rotating parent object, which means you must create this input yourself. Unfortunately, we can't give specific instructions for every use case, so you will need to integrate it into your specific input code yourself. A quick example helps illustrate one of the uses of this prefab: if you have a camera that rotates along with the mouse (as the Unity-provided character controller does), attaching the Independent Crosshair as a child of this camera will make it follow the camera's rotation - while still adjusting to the correct 3D depth. Important note: if you use an Independent Crosshair, make sure to set your Raycast Origin object carefully. This will usually be the parent object (weapon/camera/etc) of the Independent Crosshair.

Summary of how the World Space Crosshair works

Before customizing your World Space Crosshair, it's useful to know a little about how the plugin works. Essentially, a separate game object (called the "Raycast Origin") is used to cast a ray forward into the world in order to determine the correct depth for the crosshair. The crosshair itself will follow the impact point of the ray from the Raycast Origin, which means that the only input necessary to move the crosshair is to rotate this Raycast Origin. Of course, this can be done using the pre-made Crosshair Projector prefab as detailed above, or you can rotate this Raycast Origin on your own and attach the Independent Crosshair prefab as a child of this object if you wish.

The World Space Crosshair uses a special shader to make sure that the crosshair object is not obscured by world geometry, as well. This is the "TranspUnlitShaderZtestOff.shader" that is included with the plugin. If you want to use a different shader while still allowing the crosshair to be drawn on top of the world geometry, please contact support and we will be happy to help you edit your shader for this functionality.

Customization Options

The following describes the options available on the WorldCrosshair.cs component:

Raycast Origin: This is the object from which a ray will be cast into the world to determine the correct depth for the crosshair. Usually this is the parent object of the crosshair itself.

Max Sense Distance: This is the maximum distance that the raycast will extend for, which means anything beyond this distance will not be registered and the crosshair will stay at this distance. If this is set to <= 0, the main camera's (tag: "Main Camera") far clip plane distance will be used instead.

Update Interval: The amount of time in seconds between raycasts. If set to <= 0, it will default to raycasting as fast as possible. Normally this can be left at 0, but if you experience performance issues (likely due to a large number of physics interactions or colliders in the scene), you can increase this. The higher this number is, the more time there will be between raycasts, which means anything above very small numbers will have a noticeable delay on the crosshair's movement.

Use Fixed Update: If this is checked, the crosshair will update with the physics engine during the FixedUpdate call instead of during the regular Update call. Most will want to leave this unchecked unless using the crosshair to interact with the physics system in some capacity.

Enable Adaptive Scaling: Adaptive Scaling means that the crosshair will try to maintain the same relative size on screen regardless of the distance that it is from the camera. While this can be very helpful for VR/Oculus Rift games, it is mostly useless for non-VR games. You should experiment with this option on and off to see which one works best for you and your project, but it most likely should be left off unless you are working with VR applications. *Note that this variable cannot be changed at runtime without issues. For an example of how to properly change this setting without problems, see the demo included with the plugin.*

Stereoscopic Scale Tweak: This is another option that is mostly intended for VR/Oculus Rift projects. When the crosshair is very close to the player or camera, it can sometimes give the sensation that it is very small due to our stereoscopic sense of vision. This might be a weird feeling and it can be mitigated by increasing the size of the crosshair when it is close to the Raycast Origin. Turn this option on to enable this size reduction, and turn it off to disable it. The benefits of this tweak can be dependent on the unique player and might only be useful for some projects. Therefore, you should experiment with this in your project to see if helps. You may also wish to expose this option in your GUI so that players can choose for themselves.

Depth Offset: The crosshair will be offset by this distance from the raycast's point of collision with your world's geometry. For most cases, this should be left at 0, but if you need to tweak this option, it is exposed and can be changed at runtime. Positive numbers will mean that the crosshair is pushed farther away from the Raycast Origin and negative numbers will move it towards the Raycast Origin.

The crosshair texture: if you wish to use a custom texture for your crosshair, just edit the m_Crosshair material in the Images folder of the plugin. This will change all of the crosshairs in your project to the new texture. If you wish to use multiple crosshair textures, you can create multiple materials (one for each texture you would like to use). If you want, you can even create fancy effects like an animated crosshair with some creativity on your part. If you're interested in creating something like this and need help, feel free to contact us for guidance.

Special Thanks

A special thank you to eVRyday for allowing us to use his excellent stereoscopic scaling tweak in our plugin. He gets all the credit for this solution to the weird, VR-specific phenomenon. Make sure to check out his YouTube channel where he explores all sorts of topics in the burgeoning VR industry. eVRyday is a huge asset to the VR community, so please join us in supporting his work!

Support

As with all our assets, you can expect great, fast support through our website or by sending an email to support@makeorbreakgames.com. Please make sure to review World Space Cursor on the Unity Asset Store if you like it!