Easy Collider Editor - 6.18.0

Table of Contents

Quick Start Guide

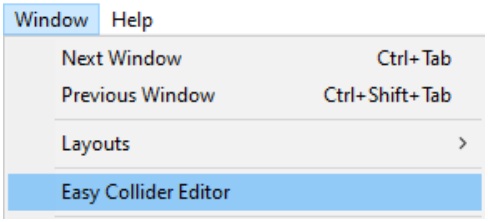
Collider Creation

Just want to jump in and make some colliders? Here's a simple guide:

Be sure to read it all as it contains some shortcuts that will speed up your workflow!

Some users have found the main video that is shown on the asset store helpful as well: https://www.youtube.com/watch?v=-69gLsVgUO8

1. Open the editor window from Window > Easy Collider Editor



2. Drag a GameObject from the scene into the Selected GameObject field.

3. Click the Creation button.

4. Move your mouse over the mesh in the scene view.
5. **Left click** or press V to select vertices on the mesh, and **right click** or press B to select any non-vertex point on the surface of the mesh beneath the mouse.
    a. Hold A or CTRL before selecting to only add vertices.
    b. Hold S or ALT before selecting to only remove vertices.
6. **Click and drag** to select or deselect multiple vertices at once.
    a. Hold A or CTRL after dragging to only select vertices in the box.
    b. Hold S or ALT after dragging to only deselect vertices in the box.
7. Use the grow, invert, clear, grow last and ring buttons to select vertices as well.
    a. Hold CTRL while clicking the grow or grow last button grows the selection until no more vertices can be selected with that button.
8. Click an icon representing the kind of collider you want to create.

Image shows all colliders that can be created along with the shortcuts underneath. The one highlighted in blue (in this case the box collider) is the one being drawn by the preview.

a. Press the **1-7** keys on your keyboard or numpad to change the current preview that is being drawn.
b. Press the ` or ~ button to create the collider currently being drawn by the preview.
c. Or **double tap 1-7** on the keyboard to create each type of collider.

9. When done creating colliders, click the Finish Currently Selected GameObject

VHACD

1. Open the editor window from Window > Easy Collider Editor
2. Drag a GameObject from the scene into the Selected GameObject field.
3. Click the VHACD button.
4. A preview will be drawn that shows the approximate result at the current VHACD settings.
5. Adjust VHACD settings until the preview shows a good result.
   For more in depth information and tips on generating convex hulls with VHACD check out the longer VHACD section in this documentation.
6. Click the VHACD - Generate Convex Mesh Colliders button.
7. When done, click the Finish Currently Selected GameObject button.

Editor Window UI

General UI

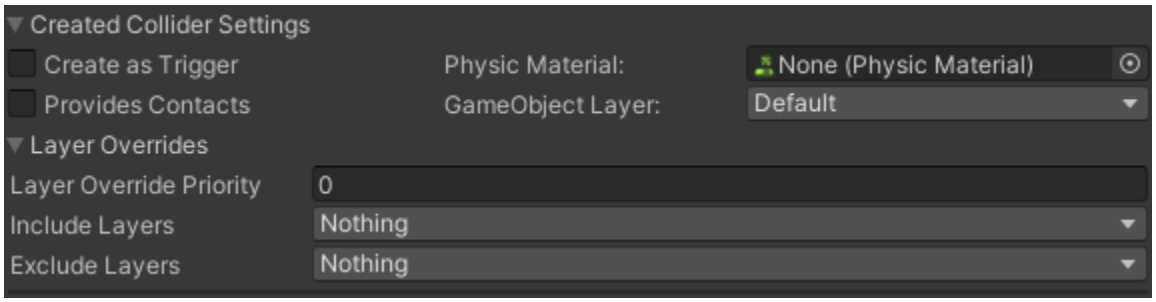
In general, hovering over the text of the UI when the window is in focus will provide tooltips that describe the options as well.

When using the asset, you will notice that some of the buttons appear to be disabled. Hovering over the disabled buttons will give you a tooltip on why the button is currently disabled. Buttons that require certain things to work are disabled and displayed differently when those conditions are not met, and enabled when they are met.

- **Selected GameObject:** the gameobject you wish to be able to select vertices from, or the parent of all the gameobjects you wish to select vertices from.
    - **Select From Scene Button:** Located beside the selected gameobject field is a button that automatically puts the object you currently have selected in the scene view into the Selected Object field.
- **Attach Collider To:** the gameobject you wish the created collider to be attached to. This gameobject is also used when calculating the collider, as it uses this gameobjects local space. This is especially useful for things like creating colliders on a skinned mesh, as you can select the whole mesh, but attach the colliders to each individual bone.
- **Finish Currently Selected GameObject:** Finishes editing on the Selected GameObject. This ensures proper cleaning up of added components required for this asset**.**
- **Vertex Selection**: Enables selection of points and vertices in the scene view. This is automatically changed as you change tabs.
- **Collider Selection:** Enables collider selection in the scene view. This is automatically changed as you change tabs.
- **Child Meshes:** Allows vertices from meshes on the Selected GameObject's children to be selected.
- **Display All Vertices:** Displays all selectable vertices.

Created Collider Settings

This section contains the settings that apply to all colliders that are created using this asset.



.

- **Create as Trigger:** Creates the collider with the IsTrigger property checked..
- **Physic Material**: Physic material to apply to each created collider.

- **GameObject Layer:** The layer to set on the gameobjects created to hold any kind of colliders (rotated colliders, child collider holders, etc). This option only displays if the Copy Object Parent Layer toggle in preferences is disabled.

Unity 2022.2+ Options (These are options added to colliders after this version of unity) Additional information on what these actually do on colliders when changed from the default can be found within the Unity manual.

- **Provides Contacts:** Controls the provides contacts option on generated colliders
- **Layer Override Priority:** Controls the layer override priority property on generated colliders
- **Include Layers:** Controls the Include Layers property on generated colliders
- **Exclude Layers:** Controls the Exclude Layers property on generated colliders.

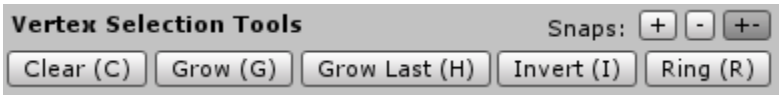
Toolbar

This section of the UI is the toolbar that lets you easily switch between all different options that Easy Collider Editor provides. Click each button to change to the UI for those tools. For more information on the UI Displayed in each section, see the other UI sections below.

- **Creation:** This section is for general collider creation using vertex selection.
- **Remove/Merge:** This section allows you to select colliders, and either remove them or merge them into a different kind of collider.
- **VHACD:** This section is for using VHACD to create convex mesh colliders.
- **Auto Skinned:** This section is used for creating colliders for skinned meshes without having to use vertex selection.

Vertex Selection Tools

This section of the UI contains several tools that make it easier to quickly select vertices.



- **Snaps:** These buttons let you change between the vertex snapping to only add, only remove, or add and remove vertices. These buttons are automatically switched between

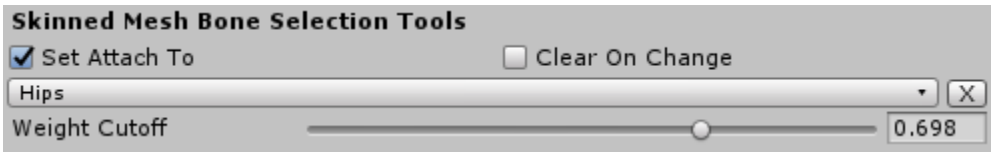
with the CTRL or ALT keys as well as the keys for Only Add and Only Remove set in the preferences.

Each button will display the shortcut beside the button label if the vertex tools shortcuts option is enabled.

- **Clear:** Deselects all currently selected vertices.
- **Grow:** Expands the selected vertices along triangle edges of all selected vertices. CTRL+Click grows until the selection can no longer grow.
- **Grow Last:** Expands the selected vertices along triangle edges of only the last selected vertices. CTRL + Click grows last until the selection can no longer grow.
- **Invert:** All selected vertices get unselected, and all unselected get selected.
- **Ring:** Attempts to select vertices around the mesh by looping around the mesh following triangle edges.

Skinned Mesh Bone Selection Tools

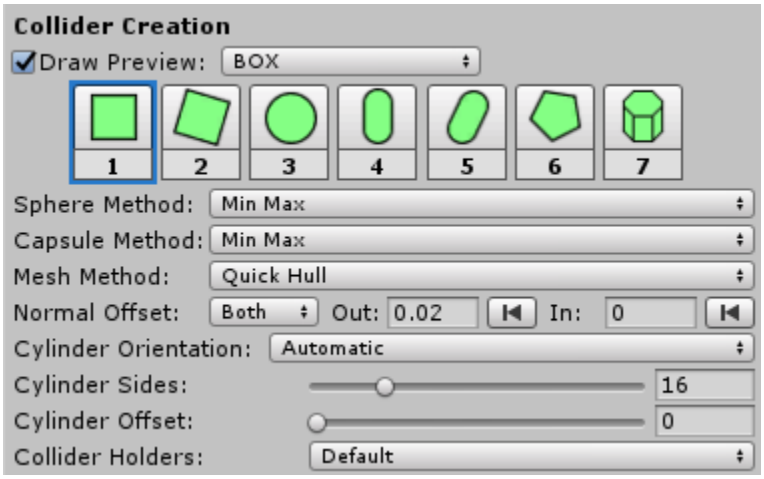
This section of the UI contains several tools that make it easier to work with and quickly select vertices for skinned mesh renderers.



- **Set Attach To:** When enabled, will automatically set the Attach To field to the selected bone.
- **Clear On Change**: When enabled, all previously selected vertices will be cleared when the selected bone changes.
- **Bone Dropdown:** A dropdown list that lets you change the selected bone from the detected skinned mesh renderers of the currently selected gameobject.
- **Weight Cutoff:** When selecting a bone or adjusting the weight cutoff, vertices that are weighted above the value for the selected bone will be selected.

Collider Creation Tab

This foldout section of the UI contains the tools used to create colliders from selected points.



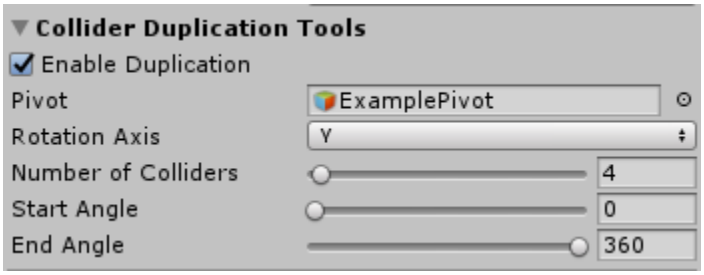
- **Draw Preview:** When enabled, draws a preview of the collider type set in the box to the right. The button that creates the collider that matches the current preview is also highlighted in blue.
- **Icons:** 1. Box, 2. Rotated Box, 3.Sphere, 4.Capsule, 5.Rotated Capsule, 6.Convex Mesh, 7. Cylinder
    - **Shortcuts:** The numbers below each collider is the shortcut on either the keypad or the numpad to change the preview to that type. The shortcut can also be pressed twice quickly to create a collider from the preview. Additionally, there is a shortcut to create a collider from the current preview (default ` or ~ key).
- **Sphere Method:** The algorithm to use when generating a sphere collider.
- **Capsule Method:** The algorithm to use when generating a capsule collider.
- **Mesh Method:** The algorithm to use when generating a mesh collider.
- **Normal Offset:** Allows extrusion of the selected vertices along it's normals. This option can be helpful to create colliders that while aligned properly, are larger or smaller than the underlying mesh.
    - **Out:** Extrusion is in the direction of the vertex normal.
    - **In:** Extrusion is in the opposite direction of the vertex normal.
    - **Both:** Allows for extrusion in both directions.
- **Cylinder Orientation:** The local axis on which to orient the cylinder created. Automatic aligns the cylinder's height with the largest axis. Cylinder orientation can also apply to capsules if the setting in Preferences is enabled. When this preference is enabled, the capsule visually does not change (unlike cylinders), and instead the capsule is rotated so

that it aligns along the specified direction. An extra transform to hold the rotated capsule collider is created if required.

- **Cylinder Sides:** The number of sides on a cylinder collider..
- **Cylinder Offset:** The amount in degrees to offset the cylinder around it's height axis.
- **Collider Holders:** Options to specify if you want empty child gameobjects to be created to hold colliders.
    - **Default:** Only creates child gameobjects for rotated colliders
    - **Once:** Creates a child gameobject once and uses it for all other colliders. Rotated colliders will also be children of this gameobject. If you rename the child gameobject while still creating colliders, another will be created.
    - **Always:** Creates a child gameobject every time you create a new collider.

Collider Duplication Tools

This foldout section of the UI contains the tools to duplicate and rotate colliders during creation.



- **Enable Duplication:** Enables and disables collider duplication.
- **Pivot:** This lets you select a different transform as the pivot point, allowing you to adjust the location you want to rotate around.
- **Rotation Axis:** The axis to rotate the collider around when duplicating.
- **Number of Colliders:** The number of colliders to create around the rotation axis.
- **Start Angle:** Angle to start duplication at.
- **End Angle:** Angle to end duplication at.

Remove / Merge Tab

This section contains the tools for both removing colliders, and merging colliders together.

Collider Selection Tools

This section only contains two buttons, to clear and invert. They function similarly to the vertex selection buttons.

- **Clear:** Deselects all currently selected colliders.
- **Invert:** All selected colliders get unselected, and all unselected colliders get selected.

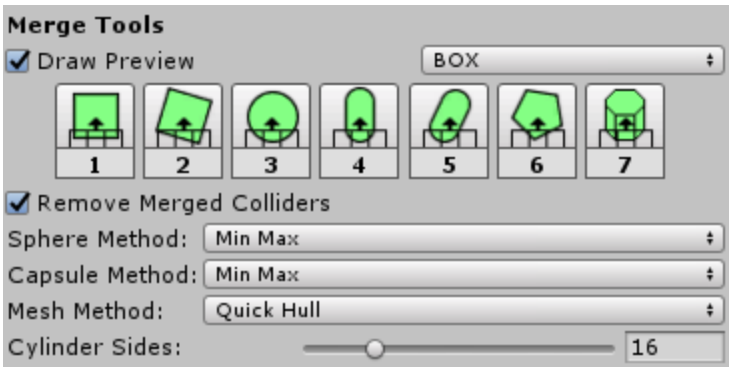
Collider Tools

This section only contains two buttons, to remove selected colliders, and remove all colliders.

- **Remove Selected:** Removes the currently selected colliders.
- **Remove All:** Removes all colliders on the Selected GameObject, the Attach To gameobject, and their children (if include child meshes is enabled)
- **Edit:** Starts editing the collider by removing it and converting it to vertices. Automatically switches you to the Creation tab.
- **Select Collider Vertices:** Similar to the edit button, but the collider is not removed before editing. This is useful if you want to use vertices from a collider to create a smooth connection.

Collider Merging

This section contains the options for merging colliders.

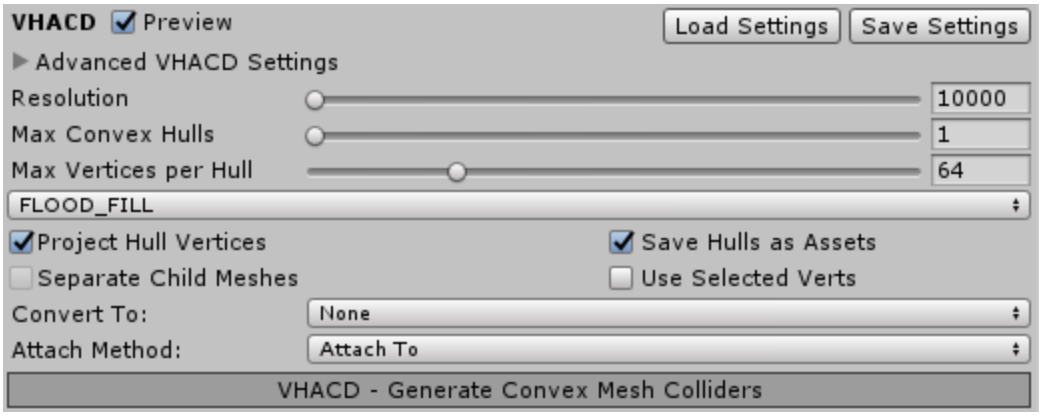


Most of the UI in this section functions exactly the same as the tools specified in the Collider Creation Tools section above. Except instead of creating colliders from vertices, the colliders selected are merged into the collider specified.

- **Remove Merged Colliders:** When this option is enabled, the colliders that are selected to be merged are removed after the merged collider is calculated.

VHACD Tab

This section contains options for generating convex mesh colliders using VHACD.



- **Preview Result:** Enabling this toggle causes VHACD to automatically recalculate and display a preview in the scene as you adjust parameters.
- **Load Settings:** Allows you to load any previously saved VHACD settings, click the button and locate the saved settings to load them.
- **Save Settings:** Allows you to save the current VHACD settings. Click the button and save them in your project to be loaded in any future sessions.

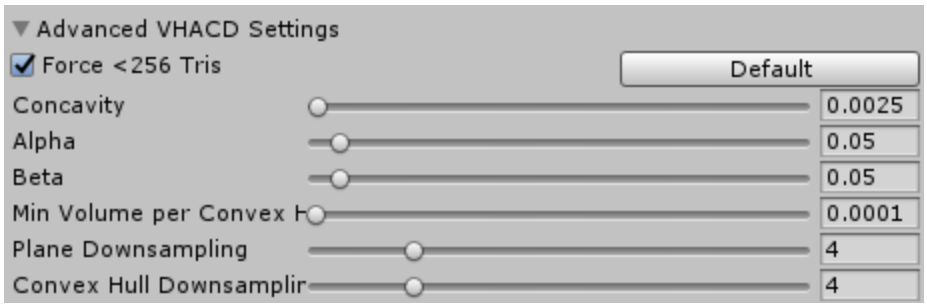
Basic Settings

- **Resolution:** Maximum number of voxels generated during the voxelization stage. With just the basic settings, this has a maximum value of 128k. With the advanced settings expanded, this can be increased up to 64 million. Increasing resolution to a high value will significantly increase calculation time.
- **Max Convex Hulls:** The maximum number of convex hulls to produce.
- **Max Vertices Per Hull**: The maximum number of vertices used per convex hull. With the basic settings this has a maximum value of 255. With advanced settings expanded, this value can be increased to 1024.
- **Fill Mode:** The fillmode used to determine what is 'inside' or 'outside' the mesh.
    - FLOOD_FILL: This is the default and uses a flood fill, and the one you should use in most cases.
    - SURFACE_ONLY: Only considers the surface as inside. Most useful for creating hollow objects.
    - RAYCAST_FILL: Uses raycasting to determine the inside from outside. Primarily useful for objects with holes.
- **Project Hull Vertices:** When enabled, projects the vertices of the convex hull back onto the source mesh, increasing accuracy.
- **Save Hulls as Assets:** When enabled, convex hull meshes from VHACD are saved as assets. This is important if you don't want to lose the collider when reloading.
- **Separate Child Meshes:** When enabled, each child mesh will be processed separately Allows you to generate convex hulls on multiple objects with a shared parent using the same parameters.
- **Use Selected Verts:** Allows VHACD to be run only on the selected vertices. When this option is enabled, vertex selection will automatically be enabled as well, and the vertex selection tools will work just as in the creation tab, and the UI will display above the VHACD settings.
- **Convert To:** Selecting box, capsule, or sphere in this dropdown takes the results of the VHACD calculation, and converts each convex-mesh collider into box, capsule, or sphere colliders.
- **Attach Method:**
    - **Attach To:** The default option, all convex mesh colliders are attached to the object in the Attach To field.

- **Child Object:** All convex mesh colliders are attached to a single child that is created as a child of the Attach To object.
- **Individual Child Objects:** Each convex mesh collider is attached to its own gameobject that is a child of a common parent which itself is a child of the Attach To Object.

- **Per Mesh:** This toggle gets displayed when Separate Child Meshes is also enabled. It allows you to specify that the attach method applies per mesh that is being run through VHACD. This allows you to attach the colliders that are generated from each mesh to the gameobjects the meshes are on, as opposed to all on the base Attach To object. With this option enabled, it's like you're individually running each gameobject a mesh is found on through VHACD at the same settings.
- **VHACD - Generate Convex Mesh Colliders:** Clicking this button begins the VHACD calculation. A progress bar will then be shown. If a new calculation is started before the current one is finished, the unfinished calculation will be discarded and a new one will begin.

Advanced VHACD Settings

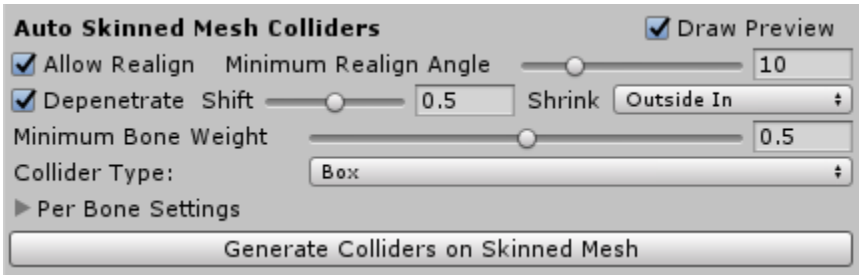


- **Force <256 Tris:** When enabled, allows VHACD to recalculate convex hulls with a lower number of vertices until under 256 triangles is reached. Convex Mesh Colliders with greater than 255 triangles can generate errors in some versions of unity.
- **Default:** Clicking resets the all VHACD settings to default values.
- **Concavity:** Maximum concavity.
- **Alpha:** Controls the bias toward clipping along symmetry planes.
- **Beta:** Controls the bias toward clipping along revolution axes.
- **Min Volume Per Convex Hull:** Controls the adaptive sampling of the generated convex-hulls.

- **Plane Downsampling:** Controls the granularity of the search for the best clipping plane.
- **Convex Hull Downsampling:** Controls the precision of convex hull generation process during the clipping plane selection stage.

Skinned Mesh Collider Generation

This section contains the tools used for automatically generating colliders along a skinned mesh bone chain. The options for creating colliders as triggers, and physic material at the top of the window also apply to the generated colliders.

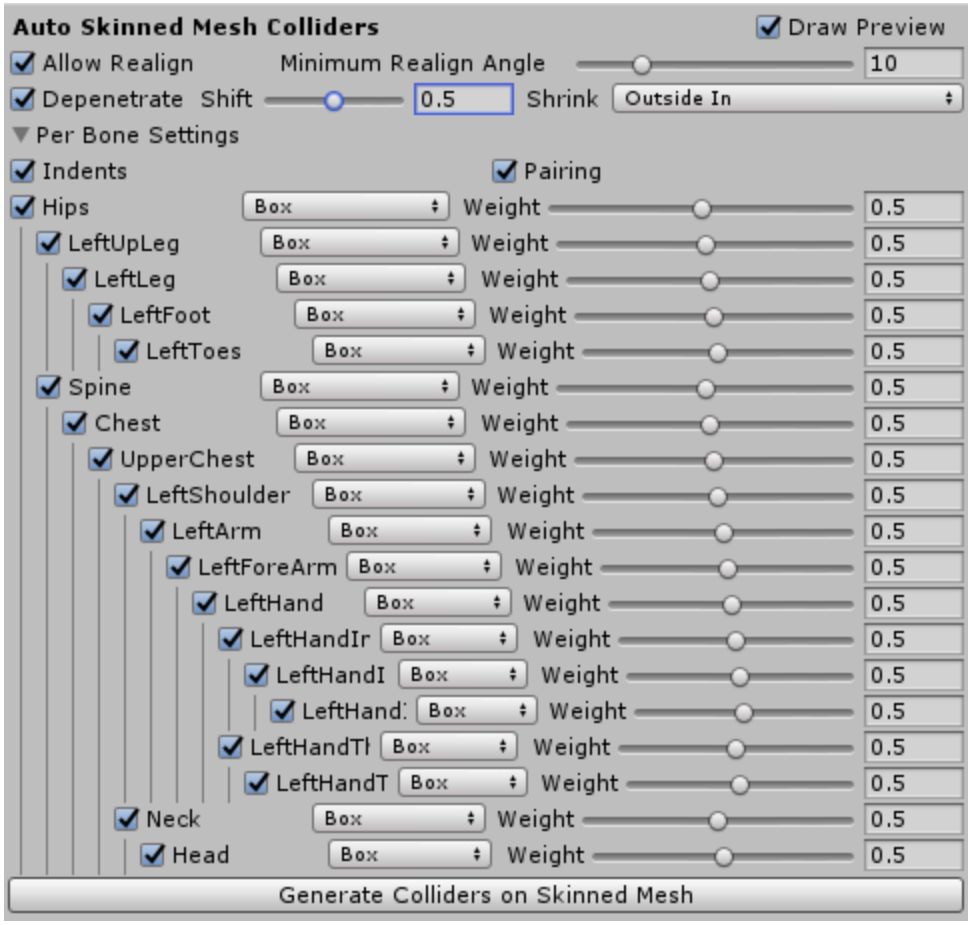


Basic Settings:

- **Draw Preview:** Toggles the preview.
- **Allow Realign:** Allows a child transform to be created to hold colliders when a bone with one child is not properly aligned.
- **Minimum Realign Angle:** When allow realign is enabled, this option sets the angle at which bones will be realigned. If the minimum angle between all the bones axis' and the next bone in the chain is larger than this value, a child transform will be created and aligned with the next bone to hold the generated collider.
- **Depenetrate:** Enables iterative shrinking and shifting to prevent collider overlap in the result.
- **Shift/Shrink Slider:** Controls the amount of shift vs shrink that is done on each iteration when trying to depenetrate colliders. A value of 0 means only shifting will be used.
- **Depenetration Order Dropdown:** Controls the order in which the colliders are processed when depenetrating.
    - **In Order:** As they appear in the Hierarchy.
    - **Reverse**: Opposite order they appear in the Hierarchy.

    - **Inside Out:** Processed from the root outwards. Ie: Starting at the hips, to the arm, to the fingers.
    - **Outside In:** Ends processed before the inner bones. Ie: Starting at the fingers, to the arm, to the hips.

- **Minimum Bone Weight:** The minimum weight a bone must have on a vertex for that vertex to be included in the calculation for that bone's collider.
- **Collider Type:** The type of collider to create when automatically generating colliders.
    - When the collider type is set to Convex_Mesh the following option becomes available:
        - **Force<256 Triangles:** Enabling this option iteratively welds vertices together to get a result that is under 255 triangles. Preventing errors in some versions of unity.
- **Generate Colliders on Skinned Mesh:** When clicked, creates and attaches the colliders on the skinned meshes' bones.

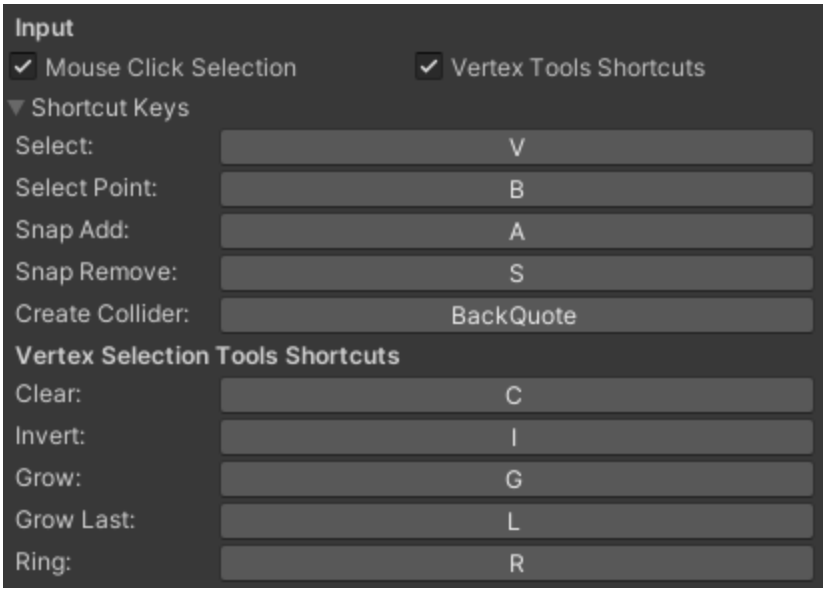


Per Bone Settings: expanding this fold out will show something similar to the image above, and allows each bone to be enabled/disabled, have a different type of collider, and different weights. Check out the Auto Generated Skinned Mesh Colliders section for more information and tips when using this feature. Holding CTRL while adjusting a parameter will also adjust the parameters of all children.

- **Indents:** Indents the UI similar to the Unity hierarchy, making it easier to find the bone you want to adjust.
- **Pairing:** Toggles automatic bone pairing. Paired bones only display one bone in the UI but adjustments to parameters affect both.

Preferences UI

This is a foldout that can be used to display all the various preferences options.

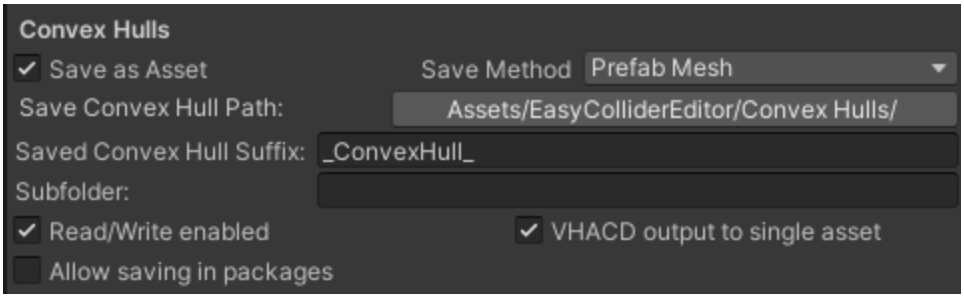


- **Enable Mouse Click Selection:** When this is enabled, you can use the left click to select vertices, and right click to select points. The normal buttons to select also work in this mode.
- **Vertex Tools Shortcuts:** You can use this option to enable or disable the vertex selection tools shortcuts if you wish.

Shortcut Keys: To change a key, press the button and then press a key on the keyboard.

- **Select:** Key to press to select/deselect a vertex or collider.
- **Select Point:** Key to press to select a non-vertex point on the mesh.

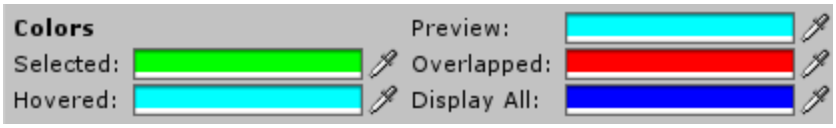
- **Snap Add:** When this key is held down vertices will only be selected. The **CTRL** key also provides the same functionality.
- **Snap Remove:** When this key is held down, vertices will only be deselected. The **ALT** key also provides the same functionality.
- **Create Collider:** Key to press to create a collider using the currently displayed preview. Remember you can also double tap the 1-6 keys to create colliders as well.
- **Vertex Selection Tools Shortcuts:** Using these keys all provide the same functionality that the buttons under Vertex Selection tools in the creation tab do. They're just shortcuts so you don't have to click a button.



Convex Hulls:

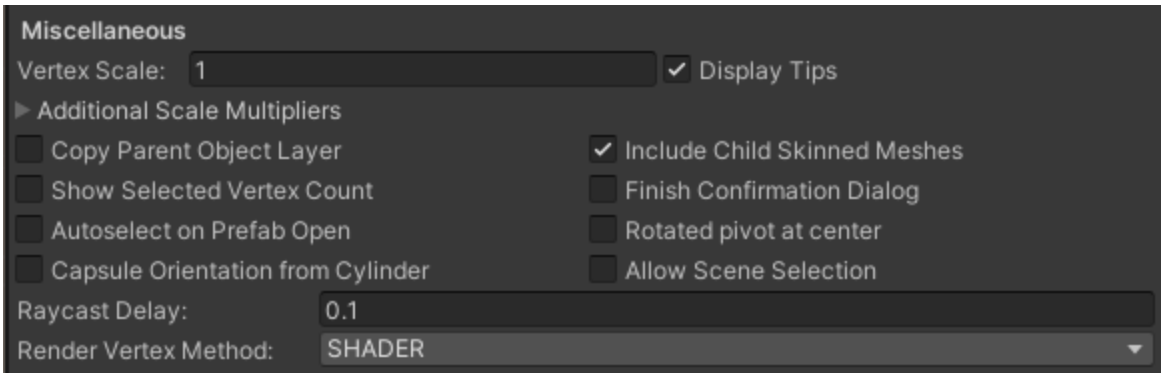
- **Save as Asset:** When enabled, meshes from creating convex mesh colliders or generating convex hulls using VHACD are saved as assets. Important to leave enabled if you want colliders to continue to function across multiple scenes, objects, and reloads.
- **Save Method:** Controls how the search for the location to save the mesh colliders' asset file is done. All save methods eventually fallback to saving in the folder specified, then a final fallback folder in the assets file if it fails to find a valid location.
    - **Prefab:** Saves in the location of the associated prefab of the selected gameobject.
    - **Mesh:** Saves in the location of the associated mesh of the selected gameobject.
    - **PrefabMesh:** Tries to save in the location of the prefab, then the location of the mesh.
    - **MeshPrefab:** Tries to save in the location of the mesh, then the location of the prefab.

    - **Folder:** Always saves in the folder specified by Save CH Path.
- **Save CH Path:** This folder is used to save the mesh collider's mesh, when trying to find the path for the save method fails, or when save method is set to folder.
- **Saved CH Suffix:** This is the suffix added onto saved mesh assets in addition to numbers.
- **Sub Folder:** The subfolder to save in when using anything other than the Folder save method. Ie: If the prefab is found in Assets/Environments/Prefabs, the convex mesh colliders created would be saved in Assets/Environments/Prefabs/SubFolderName
- **Read/Write Enabled:** Toggles read/write enabled on or off on the saved mesh assets used for convex mesh colliders.
- **VHACD output to single asset:** When enabled and using VHACD, each time you use the VHACD-Generate Convex Mesh Colliders button in the VHACD tab, the meshes generated will be combined into a single .asset file as opposed to each mesh having its own .asset file, reducing the number of files created.
- **Allow saving in packages:** This allows the files for VHACD meshes to be directly saved into folders that are within /Packages. Keep in mind that the packages folders do not actually exist within your project's directory. If the actual source package folders are not correctly shared with your team as well as your project, these generated mesh collider files will not exist in their projects and will be lost. A warning will be shown when this is enabled to remind users of this as well.



Colors:

- **Preview:** The color used to draw preview colliders.
- **Selected:** The color used to draw vertices and colliders that are currently selected.
- **Overlapped:** The color used to draw vertices and colliders that are currently selected and hovered.
- **Hovered:** The color used to draw vertices and colliders that are currently hovered and can be selected.
- **Display All:** The color used to draw vertices when the Display All Vertices toggle is enabled.



Miscellaneous:

- **Vertex Scale:** The scale of all vertices that are drawn. If the vertices are being drawn too large or too small for your mesh, try adjusting this value.
- **Display Tips:** Displays helpful tips for common issues at the bottom of the window.
- **Additional Vertex Scales:** Allows you to adjust the scaling of different vertex lists using a multiplier. For more information on what each multiplier is for, see the section on Colors above.
- **Copy Parent Object Layer:** Any gameobject that is created to hold colliders (like rotated colliders, or other collider holders) can be set to automatically copy the parent Attach To Object's layer. If this is disabled, under the Created Collider Settings foldout, there is an option to specify a specific layer.
- **Include Child Skinned Meshes:** Includes skinned meshes on children when Child Meshes is enabled.
- **Show Selected Vertex Count:** When enabled, displays the amount of selected vertices between the **Vertex Selection Tools** label, and the **Snaps** UI.
- **Finish Confirmation Dialog:** When this is enabled, a confirmation dialog is displayed if you click the finish button while still having vertices or colliders selected. A helpful reminder that the collider that is previewed wasn't created yet.
- **Auto Select on Prefab Open:** When this is enabled, and you enter into editing a prefab with the Easy Collider Editor window already open, the root object is automatically selected as the selected gameobject.
- **Rotated pivot at center:** When this is enabled, the pivot point of gameobjects that are created to hold rotated colliders is located at the center of the rotated collider as opposed to the pivot point of the parent object.

- **Capsule Orientation from Cylinder:** When enabled, the cylinder orientation dropdown also affects capsules. Unlike cylinders, the capsule visually will not change, and instead will be aligned along the specified axis. An extra gameobject is created to hold the rotated capsule collider if required.
- **Allow Scene Selection:** When enabled, you can use left mouse click to select objects in the scene view when you also currently have a gameobject selected in the Selected Gameobject Field. Works well with the “Select From Scene View” button located beside the Selected Gameobject Field. Disabled by default as colliders are not visible if a different object is selected. When enabled only the Selected Object, Attach To Object, and any child objects of those can be selected.
- **Raycast Delay:** The delay to use when raycasting, and updating the box select vertices.
- **Render Vertex Method:** The method to use to render selected, hovered, and overlapped vertices. The SHADER method is significantly faster, but requires your system to be able to use compute buffers. See https://docs.unity3d.com/Manual/class-ComputeShader.html for more details.
    - If the Render Vertex Method is **GIZMOS** the following options are available:
    - **Draw Gizmos:** Allows you to toggle on and off the drawing of gizmos. As gizmos are extremely slow compared to a shader, they can cause significant slowdown when a large amount of points are selected.
    - **Gizmo Type:** The type of gizmo to draw, either Spheres or Cubes.
    - **Use Fixed Gizmo Scale:** When enabled, gizmos use a fixed screen size.
- **Reset Preferences to Default:** Resets all the preferences above to their default values.

Selecting Vertices, Points, and Colliders

Easy Collider Editor contains multiple ways to select points on your mesh. All vertex selection methods can be undone/redone by the usual undo/redo shortcuts and buttons in unity.

All selections update and display in colors set in the preferences options. If you find the boxes are too big for your mesh, try scaling them down using the vertex scale option.

By default:

- Vertices, points, and colliders that will be selected are colored in light blue
- Vertices, points, and colliders that will be deselected are colored in red
- Vertices, points, and colliders that are currently selected are colored in green

Vertex / Collider Selection

As you hover over the selected mesh, different vertices will highlight, pressing the Select Key (Default is V) will select this vertex. If you enable the mouse selection, left click can also be used.

This process works in the same way with collider selection.

Point Selection

Sometimes it can be very useful to be able to select arbitrary points on the surface of the mesh so the colliders can include certain areas where vertices aren't located. Point selection is done using the Point Select Key (Default is B) and selects the point underneath the mouse. With mouse selection enabled in preferences, you can also use the right mouse button.

Vertex Snaps

Holding down the Snap Add key (or CTRL) or the Snap Remove key (or ALT) can also help with vertex selection. When the Snap Add key is held, you can only select vertices that aren't already selected, so you don't have to worry about accidentally removing vertices. When the Snap Remove key is held, you can only deselect vertices.

Snaps can be particularly useful to deselect points that are not vertices.

Drag Selection

Clicking and dragging in the scene will start a box selection. This allows you to select, or deselect lots of vertices all at once.

Additionally, holding down the vertex snaps keys after starting a drag select allows you to use vertex snaps as well. For example: holding CTRL will only add to the selected vertices, while ALT will only remove already selected vertices.

Drag selection does not work for selecting colliders.

Vertex Selection Tools

There are several buttons here that get enabled and disabled as vertices are selected.

Clear

This button allows you to quickly deselect all currently selected vertices in case you made a mistake, and don't want to go through multiple undo operations.

Grow

Once at least one vertex is selected, this button allows you to grow the selected vertices outwards from the currently selected ones. This button expands outwards along edges for all currently selected vertices. Hold down CTRL while clicking this button to grow until no more vertices can be selected with this button.

This option can be useful if you have a large complex mesh that has sections that are not welded together.

Grow Last

This button functions similar to the Grow Selected Vertices button. The difference is that this button only grows from the vertex, or vertices that were selected with the last operation. Hold down CTRL while clicking this button to grow until no more vertices can be selected with this button.

Invert

This button deselects all currently selected vertices, and selects all currently unselected vertices.

Ring

Uses the last 2 vertices selected to attempt to create a ring of selected vertices around an object.

Collider Creation

Trust me when I say that after a little bit of use, you'll be able to create colliders very quickly. Be sure to use the shortcuts (1-6) to switch between collider previews, use the ~ key to create the preview (or double tap 1-6), and play around a little to get used to how the vertices that are selected translate into the preview.

- The preview can be changed between each collider type by pressing the number on the keyboard that matches the number in the UI.

- The fastest way to create colliders is by using the Create Collider Key (defaut ~) or double tapping the same number used to change the preview. Ie: Quickly pressing 1 twice on the keyboard will create a box collider.
- Capsules, and spheres both have multiple algorithms available to them that can be changed using the dropdown. The method that is set by default (Min Max) generally works the best in most cases.

Be sure to check out the examples section for demonstrations of vertex selection for creating all kinds of colliders.

Boxes

- Normal box colliders: select all the vertices in whichever order you wish for box colliders, and they will all be included within the box.
- Rotated boxes use the first 3 points that are selected for it's orientation. Once you've played around with the preview enabled and selecting the first 3 points differently, creating rotated box colliders will be easy as well.

Capsules

- Min Max Methods: The order of selection of points with min-max methods is not important. The radius and diameter methods add additional height (based on radius, or diameter) to the capsule collider during generation.
- Best Fit Method: This one is trickier to get working properly, so I'd suggest using Min-Max methods where possible. The first 2 points selected define the height of the inner cylinder portion of the capsule collider. The other vertices are used to calculate the radius of the capsule. The easiest way to use this method is to select the first 2 points for the proper height, then use the ring select method on another 2 vertices to create the radius.

Spheres

- For all sphere methods, select points in whatever order you wish.
- I'd recommend using the Min-Max method as it's the easiest to use. Just select everything you want included in the sphere.
- The Best Fit method calculates a sphere where the points selected most closely fit the surface of a sphere. This is most useful when you have a surface where there is a portion

of a sphere on the surface of a mesh. Selecting more points generally makes this algorithm more accurate.

- The Distance method can be slow with large amounts of points selected. In these cases it falls back to a less accurate calculation. The fallback calculation will not be used unless an extreme amount of points are selected. I find this method most useful on objects that aren't exactly spherical.

Rotated Colliders

- Rotated boxes use the first 3 points that are selected for it's orientation. Once you've played around with the preview enabled and selecting the first 3 points differently, creating rotated box colliders will be easy as well.

- Rotated capsule colliders use the first 2 points that are selected for it's orientation. Again, playing around with the preview enabled and selecting different points helps you quickly understand.

- Rotated box colliders are aligned along an axis defined by the first 3 points that are selected. The easiest way to select appropriate points is to select 2 vertices along a long straight edge of a triangle that aligns with the collider you want. Ideally, the third point should be the other point of this triangle. Otherwise, the third point would lie along a flat plane, or one side of a box, that also aligns with the rotation of the collider. See the examples for more details.

- If the Attach Collider To field is a different object than the Selected GameObject, you may not need to use rotated colliders. If the gameobject you're attaching to has an axis that aligns with the direction of the collider you wish to create, you can create regular colliders. This is likely the case when creating a collider for a bone's transform on skinned meshes. If the bones are aligned correctly with the geometry, regular colliders can be attached to the bone itself.

Mesh Colliders

- There are 2 methods that creating mesh colliders from vertices can use.

    - **QuickHull** is generally the best method to use. It produces a cleaner mesh that could potentially be used for other things. The result of quickhull could also be used as a simplified static mesh collider for your object.

    - **MessyHull** produces a mesh that simply uses all the vertices you've selected and leaves the actual calculation of the convex hull to the mesh collider component's

internals. The actual mesh of the MessyHull, as the name implies, is messy and not usable for anything else other than a source mesh to a convex mesh collider.

- Mesh Colliders require at least 4 points to be selected. Additionally, all 4 points must not be collinear (lay along a single straight line) or helpful errors will be displayed. QuickHull also requires that the points must not all be coplanar (lay all on the same plane). **If you wish to make a convex mesh collider for a flat surface, be sure to use vertex normal offsets to give the collider a little bit of depth.**

- All points selected will be used to create the mesh used in the mesh collider. The more points you select, the more complex the resulting mesh collider will be. **It is best to select as few vertices as possible that still allow you to maintain the desired shape and accuracy.**

- If you select a significant number of points, you may actually get errors from unity itself once the collider is created. These errors come from physx, and occur because the convex mesh collider has too many vertices, triangles, or all points are coplanar.

- Meshes created as convex mesh colliders will be saved if the option is enabled in preferences. Various options exist for where to save the mesh asset. If Save at Selected's Path is enabled it will try to save in the same the folder the Selected GameObject is saved in. Otherwise it will save in the folder specified in the preferences. If neither path exists, it will save the mesh in the same folder as the preferences file (In the EasyColliderEditor folder).

- If you don't want to manually create convex mesh colliders by selecting points, try out the VHACD section of Easy Collider Editor. Keep in mind that you have significantly less control over the output of VHACD because it's controllable only by the parameters. I've added options like using selected vertices in VHACD to give back some of this control. See the VHACD section in this documentation for more information.

Cylinders

- Cylinder colliders are just convex-mesh colliders that are shaped like cylinders.

- The number of sides of the cylinder can be set using the slider.

- Cylinders can have the axis they are aligned with by changing the cylinder orientation option. Automatic aligns with the longest axis, and Local X, Y and Z align the cylinder's height with the Attach To object's local X, Y, and Z axis respectively.

- Cylinders can also be rotated around it's height axis using the cylinder offset slider. This can be used to align the sides of the cylinder with the mesh.

Collider Duplication

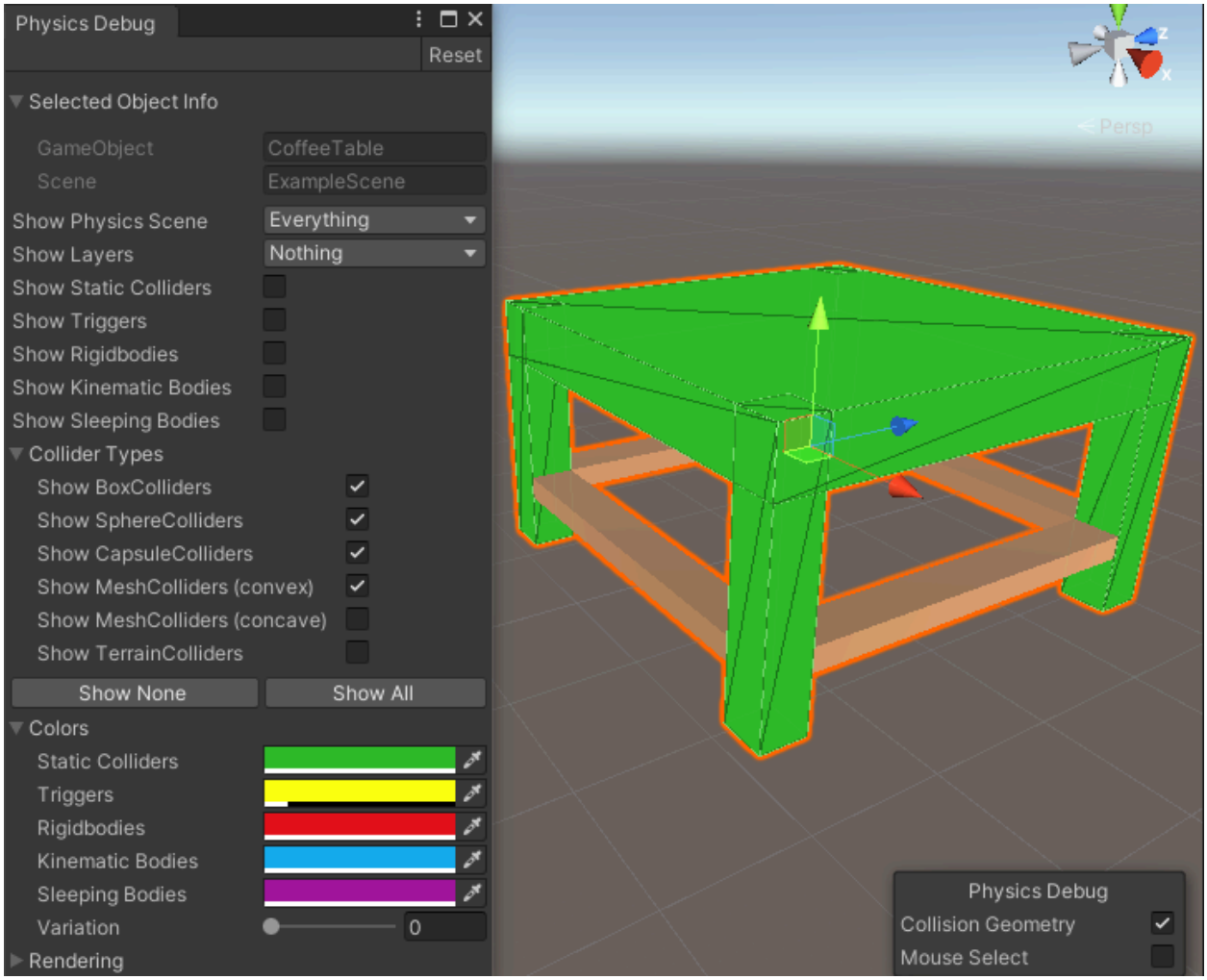
- Collider duplication allows you to select vertices to create one collider, and duplicate and rotate around an axis to quickly create multiple colliders.
- It is particularly useful for creating primitive colliders around the edges of a circular object like a cup or a bowl.
- Remember that you can change the pivot object to another object, and the colliders will still end up attached to the Attach To object. You can create an empty gameobject to temporarily use, and position it how you would like. The preview will be updated as you move the pivot around in the scene.

Using the Physics Debugger

- The physics debugger is a built-in unity tool and not a part of this asset, but using it alongside this asset can be helpful to determine what areas of the mesh are covered by colliders and which are not.
- The physics debugger can be opened by going to Window -> Analysis -> Physics Debugger.
- There are a couple of settings that changing can make using the physics debugger much easier, as by default it shows everything: Clicking the “Show None” button, and then enabling box, sphere, capsules, and convex mesh colliders will provide the best experience when working along with Easy Collider Editor. This asset uses static mesh colliders for vertex selection, so that option should be left unchecked.

  In the screenshot below you can see the options I typically use when using the physics debugger alongside my asset. It provides a much clearer visibility of what parts of the mesh are covered by colliders and which are not. Changing the color and alpha of the Static Colliders option will allow you to also see the mesh underneath as expected.

  For more information about the unity physics debugger, see the official documentation here: https://docs.unity3d.com/Manual/PhysicsDebugVisualization.html



Collider Creation - Examples

The following examples have been updated to use the preview functionality that is now built-in to all aspects of this asset. An extra script was used to thicken the drawing of the resulting colliders so they appear better in the images as well. **Only the first few examples show the actual result, otherwise only the preview is shown.**

Creating Colliders:

1. **The preview being drawn is the same as the button highlighted in blue. In the above image, this is the box collider.**
2. **There are 3 ways to create the collider that is currently being drawn by the preview:**

a. **Click the highlighted button**
b. **Press the ~ or ` key on the keyboard**
c. **Double tap the number of the keyboard that matches the collider. In this case the number would be 1.**

3. **The preview can be switched between different colliders in two ways:**
    a. **Pressing the number of the keyboard that matches the collider once. To change to a capsule collider preview, we would press the number 4.**
    b. **Manually changing the value in the dropdown menu beside the Draw Preview toggle.**

BE SURE TO CHECK OUT THE QUICK START GUIDE AT THE TOP OF THIS DOCUMENTATION

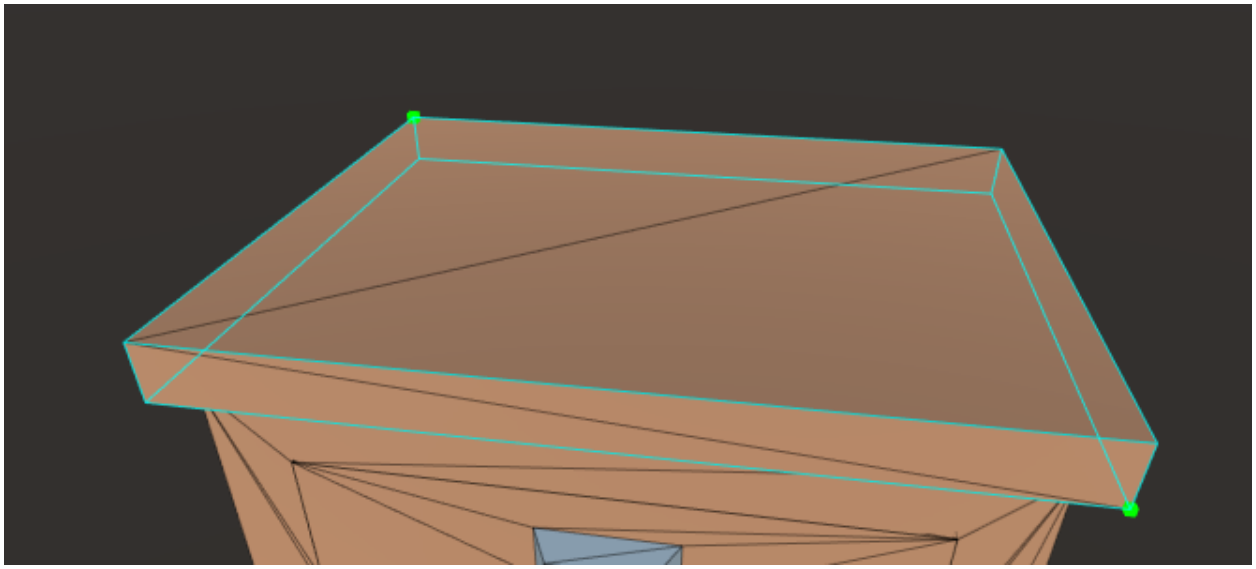
The following examples should be used as a demonstration on how to select points to generate the colliders you are looking to achieve. An important part to remember is that a small change in vertices selected can have a large impact on the resulting collider.

Note that even though some examples use static meshes, and others use skinned meshes, the concepts in each example apply to both. Skinned meshes are also used in several examples to show how you don't always need to use a rotated collider if the joint you are attaching the collider to is properly oriented.

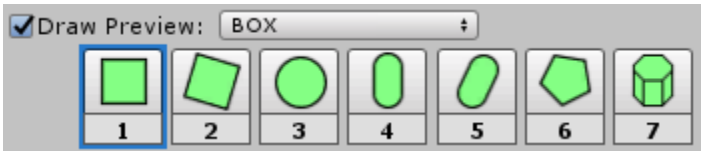
If you have any thoughts on how to make the following examples more clear please contact me at pmurph.software@gmail.com

Boxes

Example #1: Box on a Static Mesh

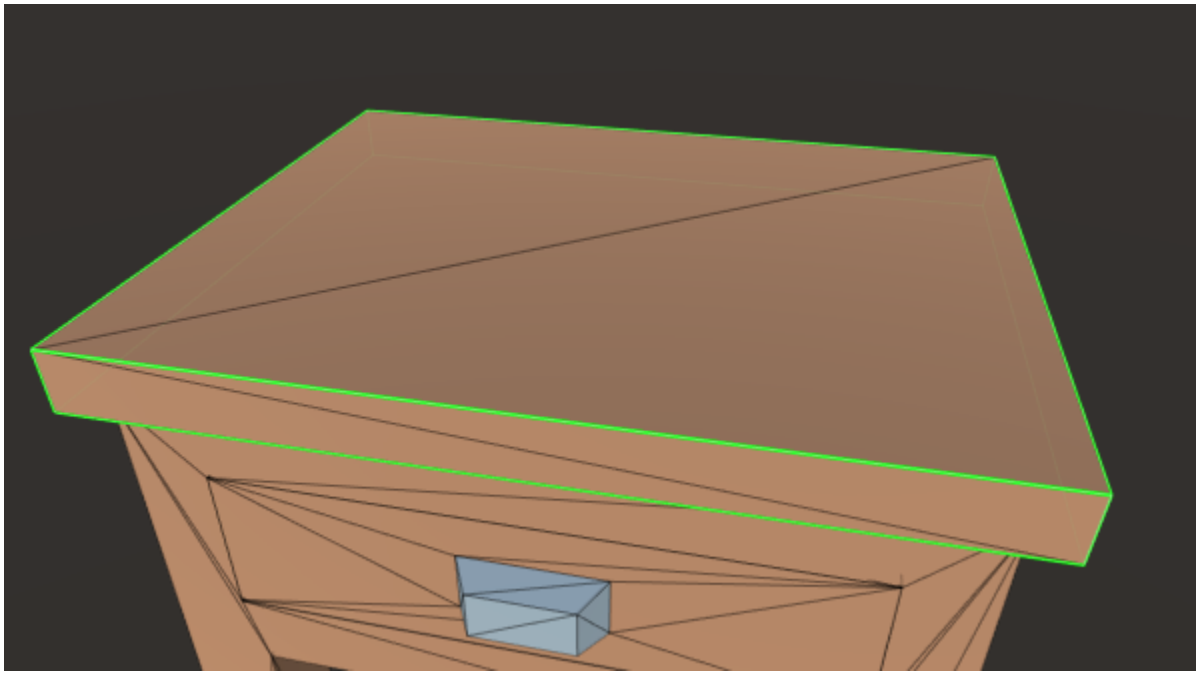


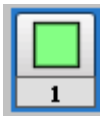
In this example we are creating a box on a simple static mesh of a cabinet. We have only selected two vertices; the corners of the box we wish to create. Note that we could select as many points as we wish, and the box collider generated would contain all of these points.



The preview is currently drawing the box collider that would be created. This preview can be changed to other colliders by pressing the number below the appropriate collider button in the UI on the keyboard. If you wanted to see what a sphere collider would look like with the vertices selected, pressing the 2 key on the keyboard would preview a sphere collider.

The preview that is currently being drawn can also be changed by the dropdown menu, and is also the button that is highlighted in the blue.

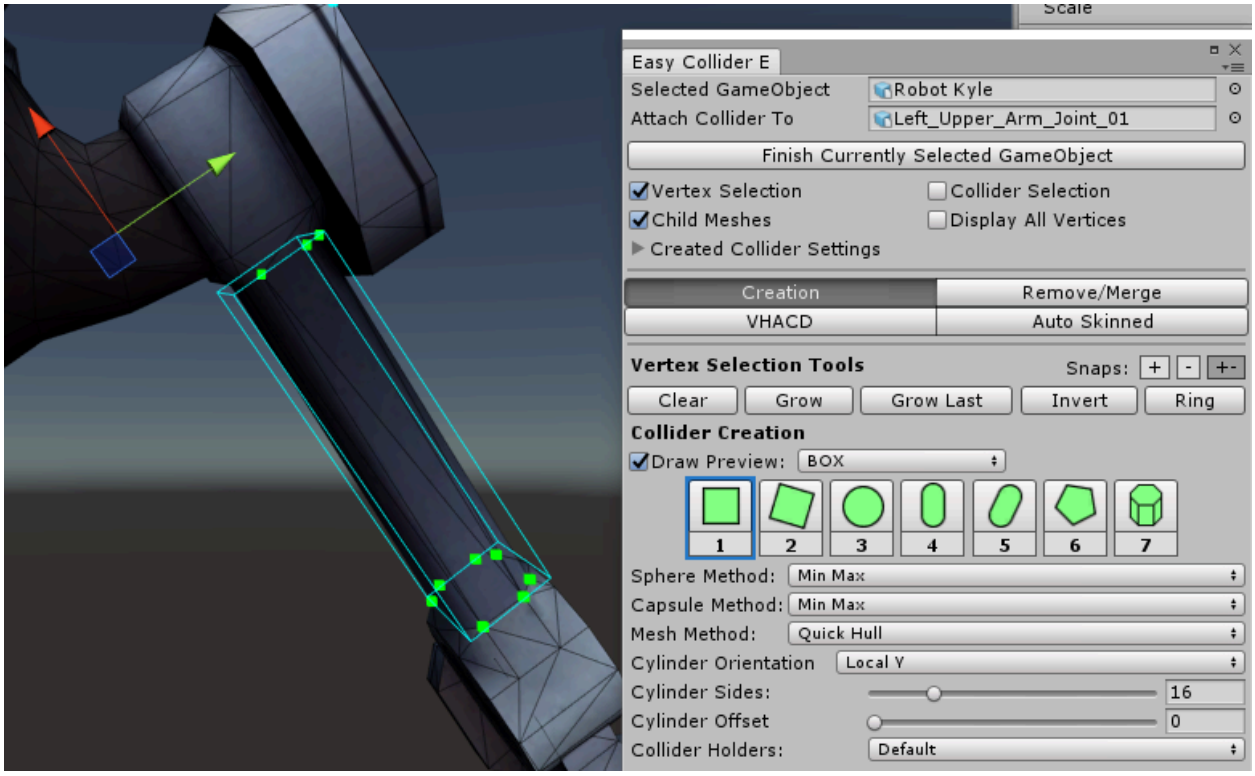


Shown above is the result of creating the box collider. This can be done in multiple ways: by clicking the appropriate button 

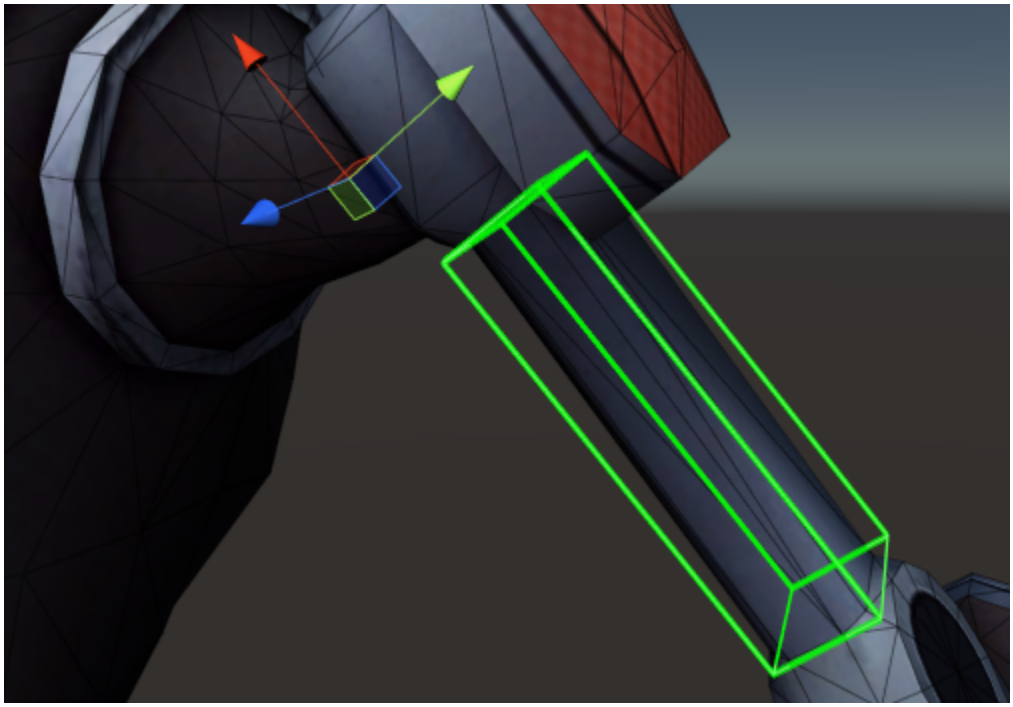
with your mouse, pressing the ` or ~ key on the keyboard (editable in the preferences) or double tapping on the keyboard the number below the button (in this case 1).

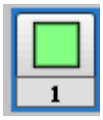
Example #2: Box on a Skinned Mesh

In this example we're going to add box colliders to the upper arm arm of a skinned mesh. In this case we will use Unity's free Space Robot Kyle asset.



We have selected the root object of the whole mesh, Robot Kyle, while making sure that the Child Meshes toggle is checked. The Attach Collider To field was changed to the Left_Upper_Arm_Joint_01. Notice how the axis of this joint aligns with the boxes we wish to create. In properly created skinned meshes, a good alignment is usually the case. This means we can use regular colliders instead of rotated colliders.



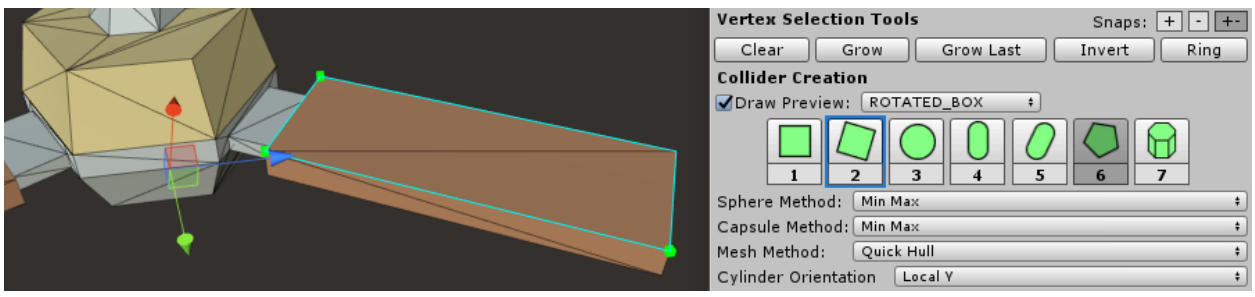
Shown above is the result of creating the box collider. This can be done in multiple ways: by clicking the appropriate button 

with your mouse, pressing the ` or ~ key on the keyboard (editable in the preferences) or double tapping on the keyboard the number below the button (in this case 1).

Note that if the joint was not aligned correctly we would have to use rotated colliders. We would still want to set the Attach Collider To field to the same joint as the properly aligned one, as this transform rotates with the arm during animations.

Rotated Boxes

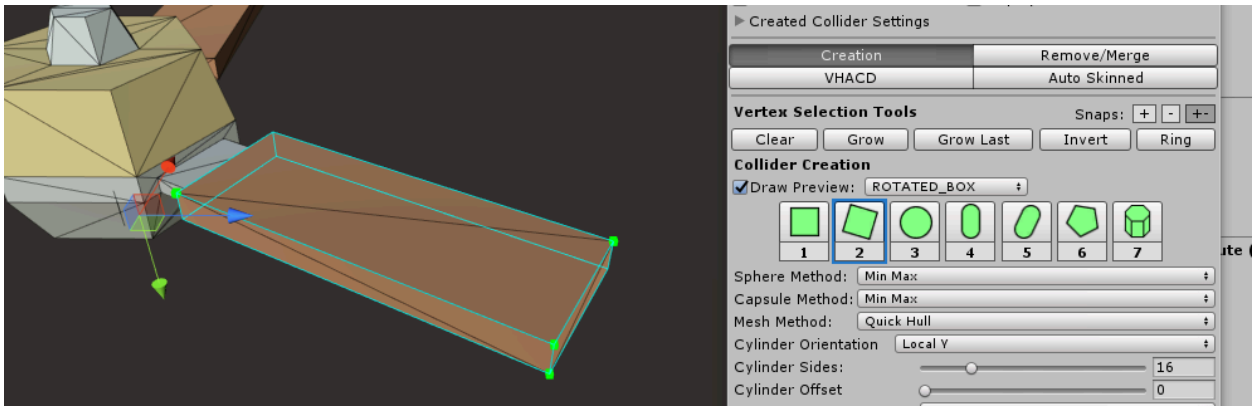
Example #1: Rotated Box on a Static Mesh



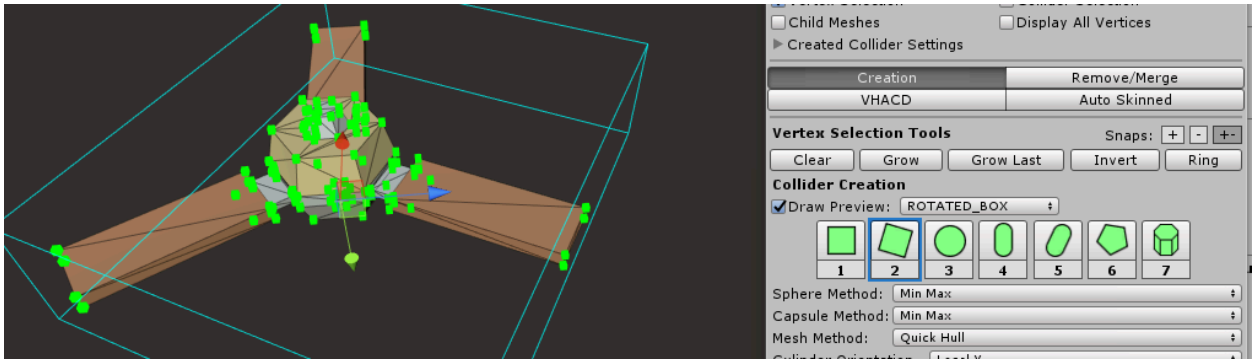
In this example of the ceiling fan, you can see that the fan blade does not align with any of the transform's axis. This makes it a good candidate for a rotated collider. In this case, I am looking to create a box collider that contains the whole angled section of the object.

I have selected 3 appropriate points that lay on the blade of the fan itself. These three points define the orientation that the rotated collider will have. The order of selection is important for rotated colliders. When selecting the first two points, try to think of aligning the edge of a box between the two points. When selecting the third point, try to imagine aligning the face of the box with all three points.

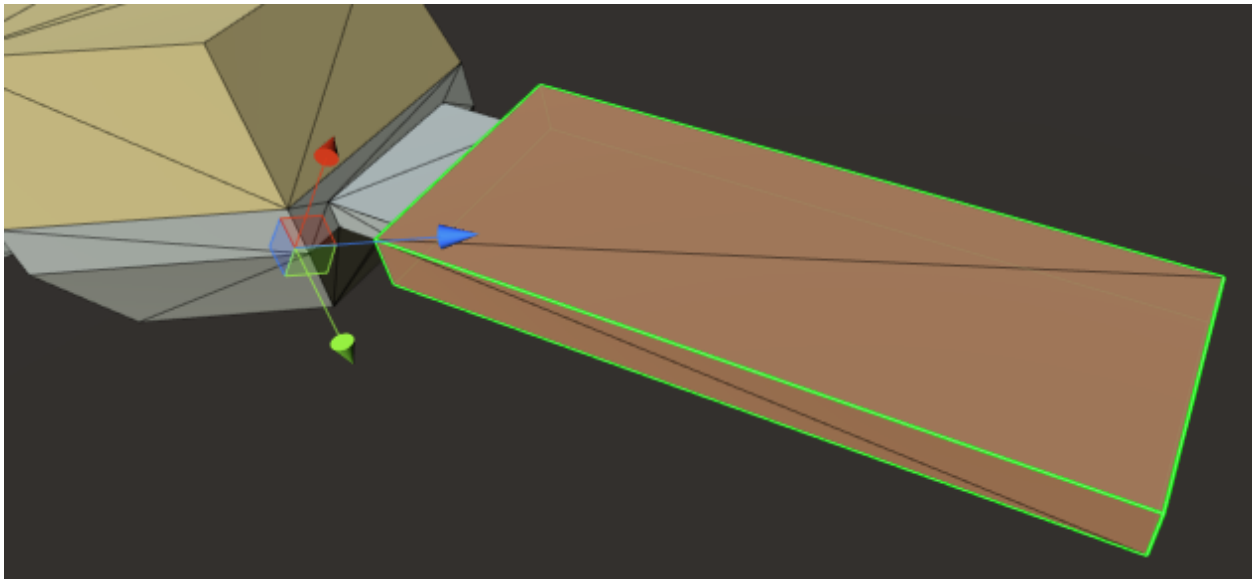
I suggest selecting two points and selecting various different vertices for the third point until you get the hang of it. Luckily the preview is there to make it a lot easier to visualize and understand what is going on.



After selecting the third point, any other point (including the original 3) is used to define the size of the box. As you can see above, selecting a single extra point shows us a preview of the box collider we need.



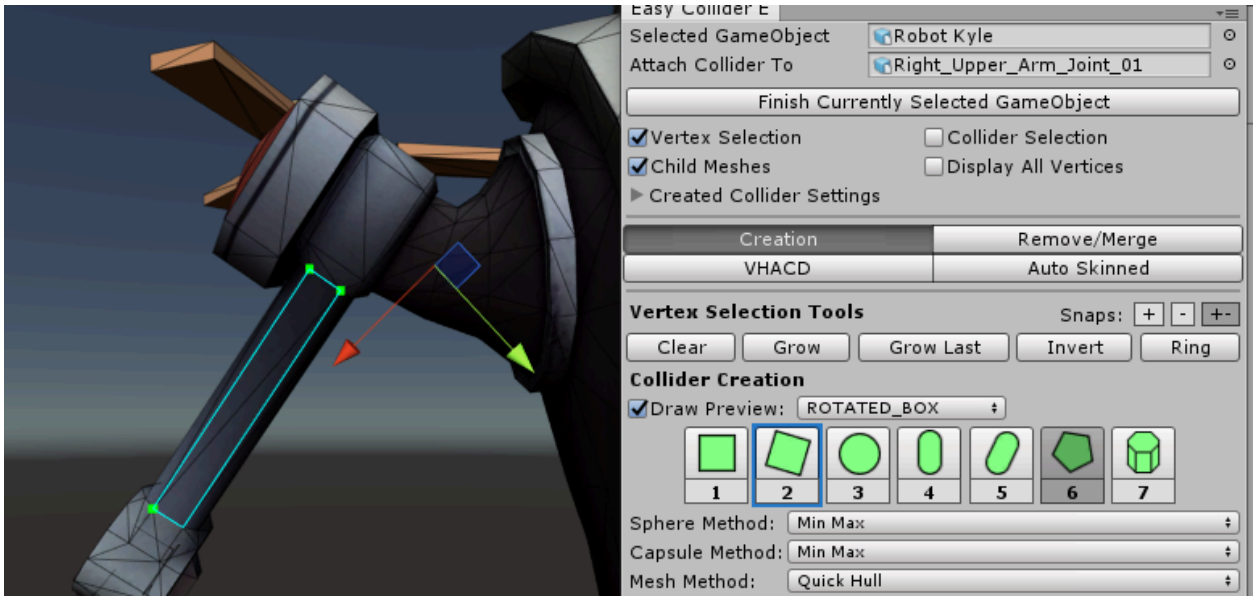
Keep in mind you can select as many points as you wis, and they will all align with the original 3 points.



Going back to the original 4 points we selected, the result of creating a rotated box collider is shown above.

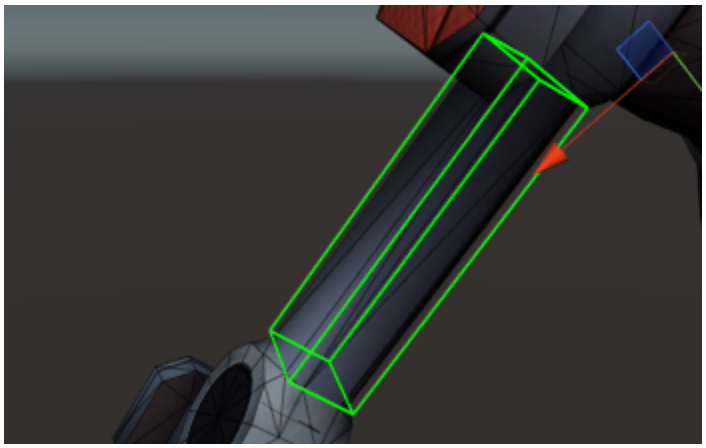
Example #2: Rotated Box on a Skinned Mesh

In this example, we are going to use the Right Upper Arm of Unity's free Space Robot Kyle asset.



We have selected the root of the skinned mesh, Robot Kyle, with the Attach Collider to set to the upper arm joint. As you can see in the image above, the box collider we want to create does not properly align with the arm's axis. This is a case where a rotated box collider would be needed.

Even though we are going to create a rotated box collider so it aligns properly, the Attach Collider To field was changed to the misaligned arm joint. This is needed because this is the transform that rotates when animations are playing, so we need to attach our rotated colliders to that as well.

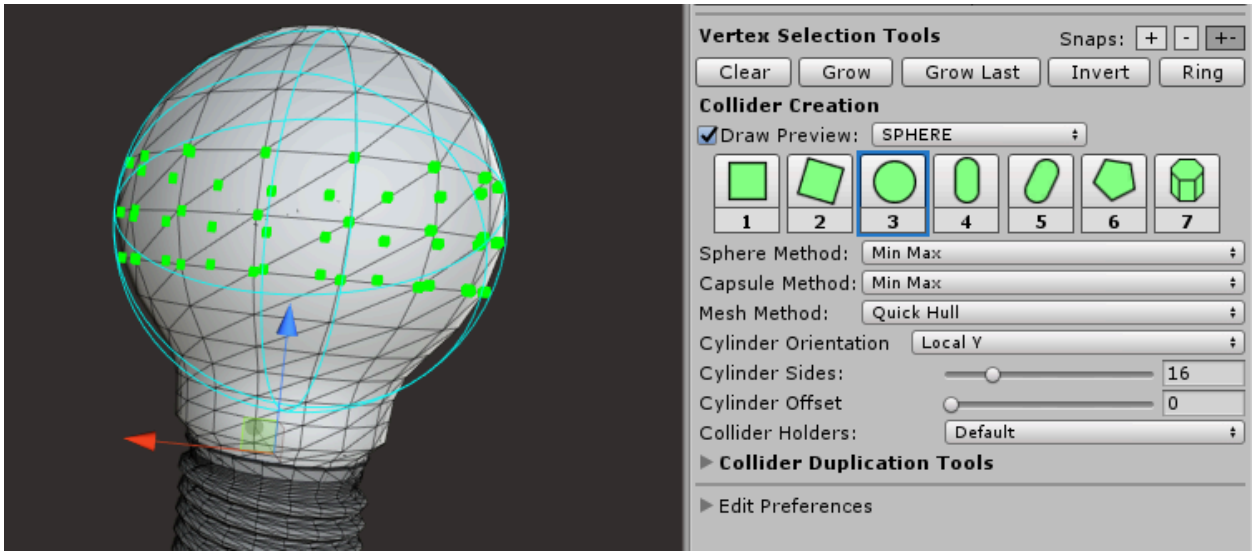


After selecting a few more vertices to create the size of the box, the collider can be created. Shown above is the result of creating the rotated box collider.This can be done in multiple ways: by clicking the appropriate button 

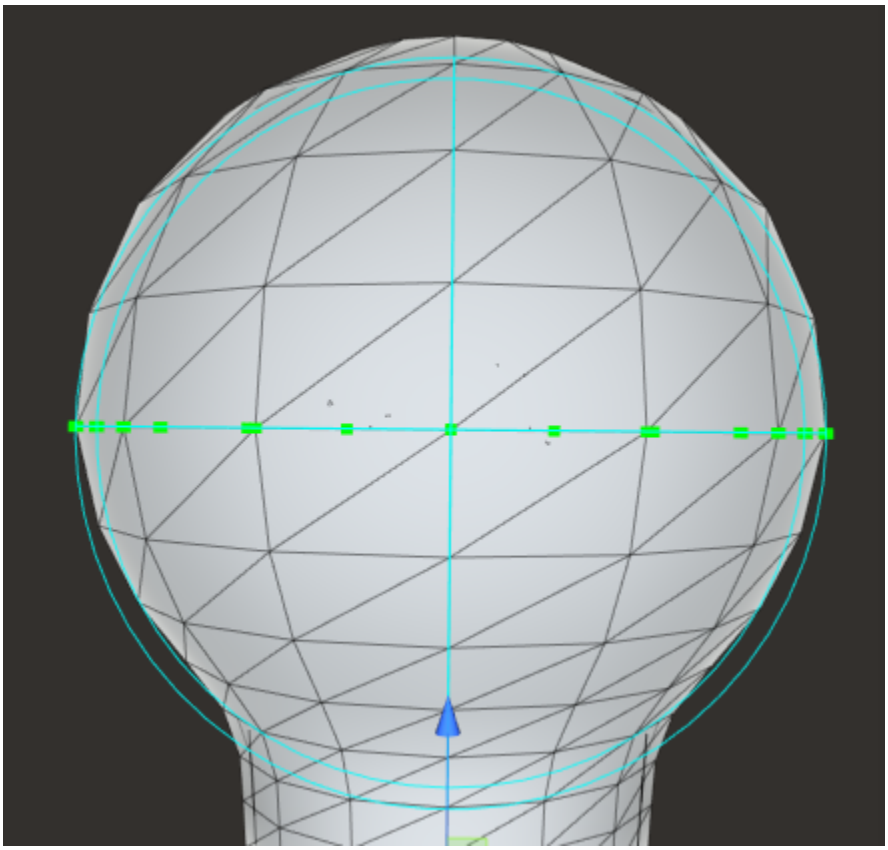
with your mouse, pressing the ` or ~ key on the keyboard (editable in the preferences) or double tapping on the keyboard the number below the button (in this case 2).

Spheres

Example #1: Min Max on a Static Mesh



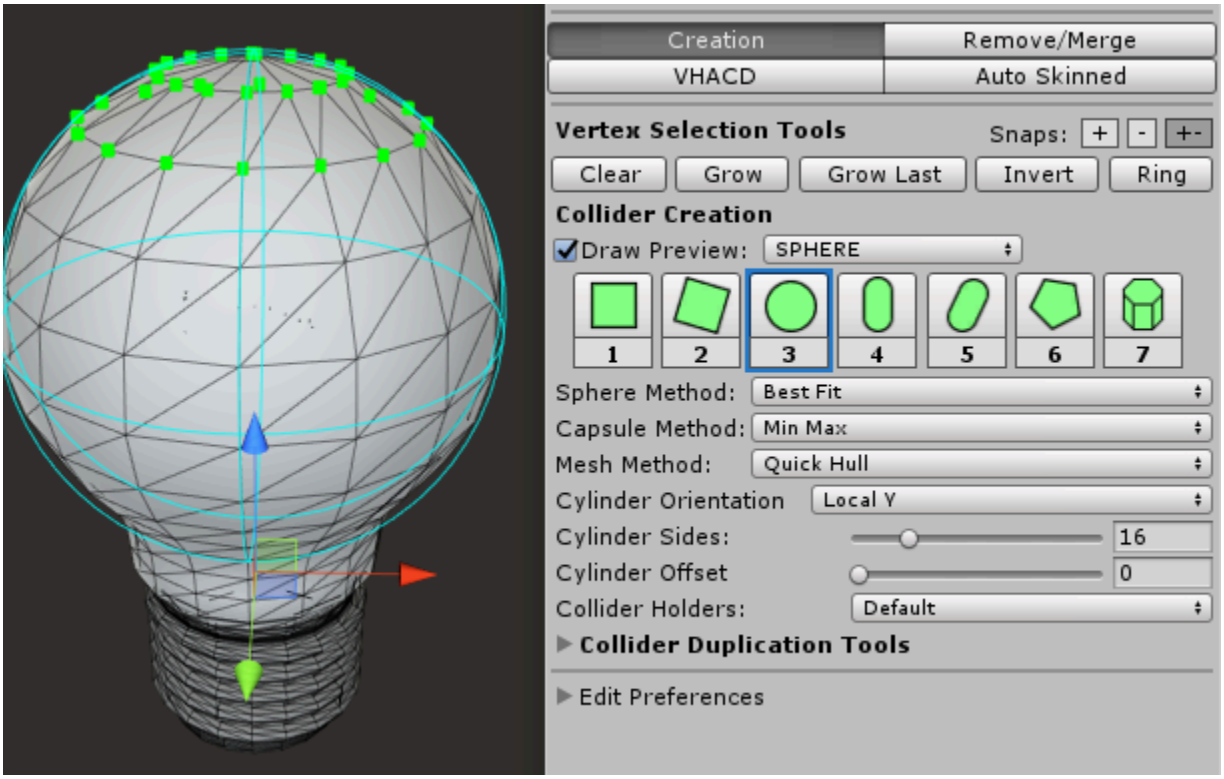
In this example we have selected the vertices in a ring around the center of the lightbulb since this is the area that the lightbulb is centered around. Although the difference is minor (see below), if we only selected the middle points, the sphere would be slightly offset because the vertices in the middle do not perfectly align with the center of the sphere we want.



In this example, selecting all the vertices with a box selection results in a sphere that is too large. This is because it is not a perfect sphere. Additionally, it is difficult to guess where the bottom of the sphere would be, given the shape of the mesh changes into more of a cylinder.

Example #2: Best Fit on a Static Mesh

This example uses the Best Fit method. As you can see in the image below, only some vertices of the sphere were selected. This was done by selecting the top vertex and using the Grow Selected Vertices button.

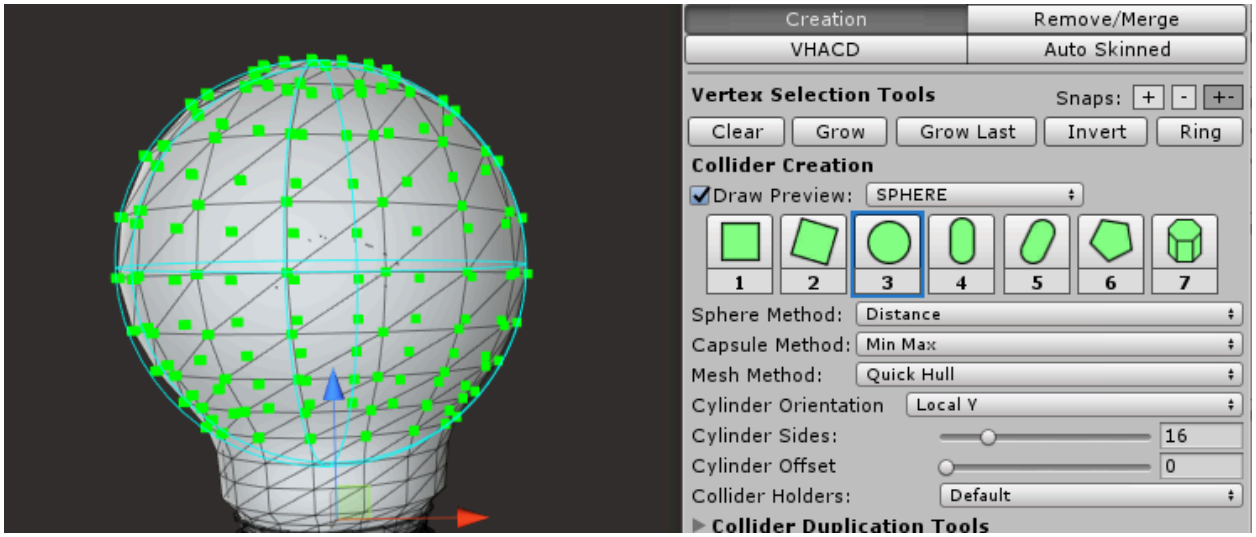


In the case of this lightbulb mesh, it was important I did not do a full box select and try the best fit method. With this lightbulb mesh, there is actually a filament inside with additional vertices. Selecting these vertices would have included them in calculating the sphere, and significantly altered the result.

As you can see, the vertices selected all lay on the surface of the sphere collider I want to create. This method calculates a sphere that has a surface that most closely matches the selected vertices. Generally adding more vertices that lay on this surface increases the accuracy of this calculation.

Example #3 Distance on a Static Mesh

In this example we have changed the Sphere Method to Distance. On the left we have used a simple box select to select all the vertices we want to use to calculate the sphere. The result is seen in the left image.

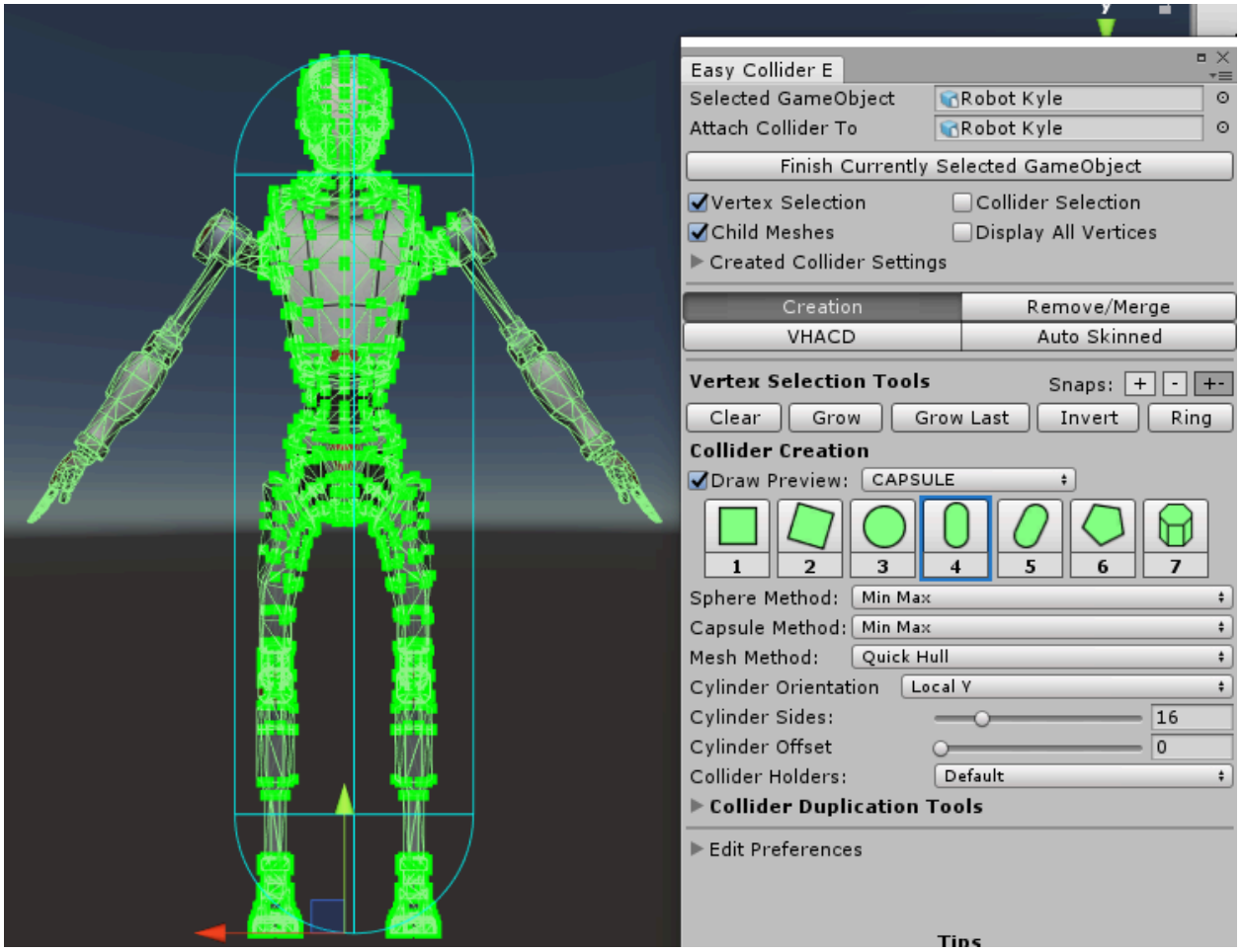


If this mesh had a very large amount of vertices, and they were all selected, this method would fall back to a less accurate but faster method which uses the average position of each vertex. In this case the distance method works pretty well as the furthest two points are on opposite ends of the sphere we want to create.

Capsules

Example #1: Min Max on a Skinned Mesh

Again we are going to use our friend Space Robot Kyle in this example. In this example we used a box select to quickly select a bunch of vertices on the characters mesh. Then, we made sure the Capsule Method property was set to Min Max.

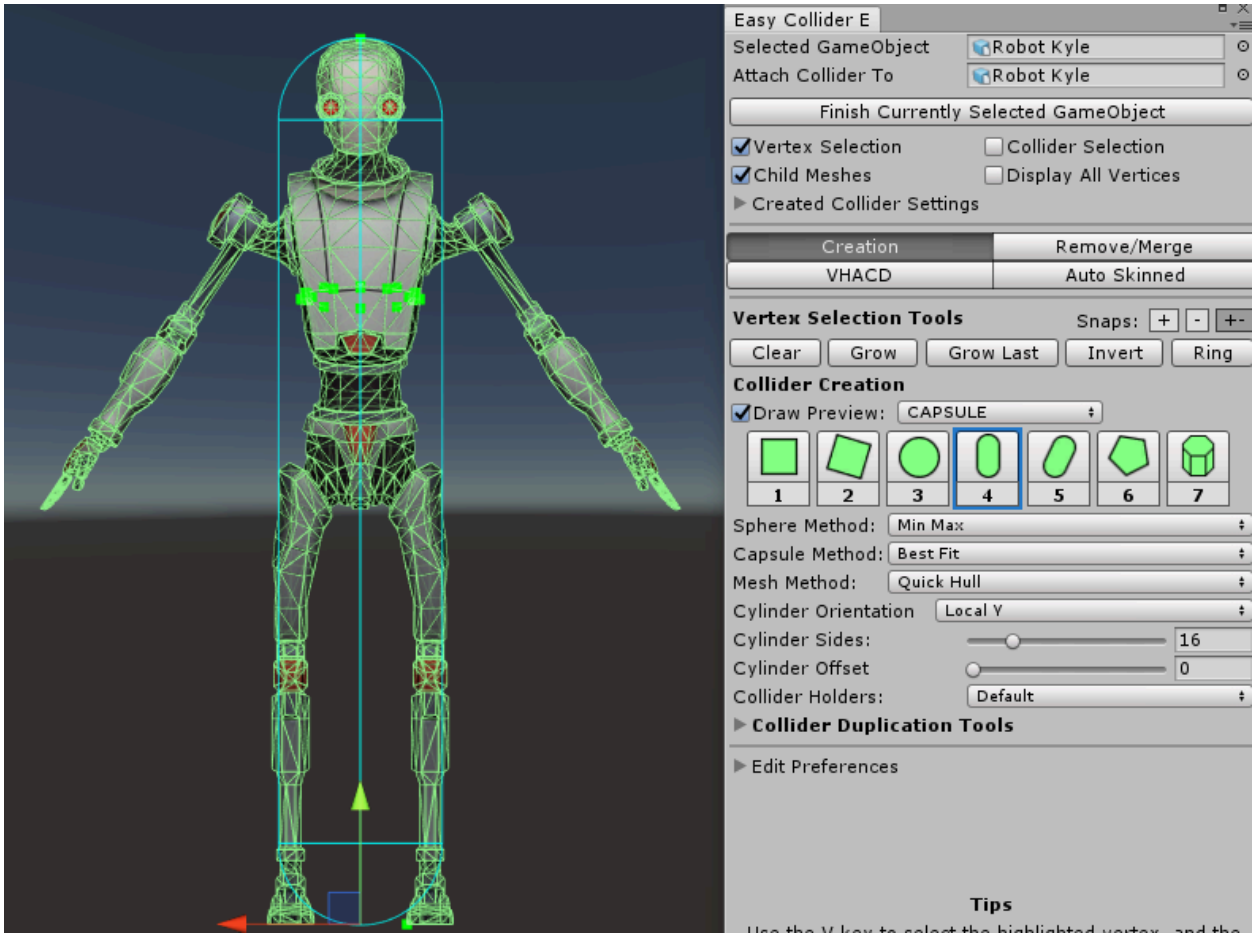


If you look at the preview you will notice that it does not contain all of the points that we selected. This is only the case at the top and bottom of the capsule collider, especially noticeable at the feet. In most cases this is actually what we want from a capsule collider.

Methods to add additional height are also available through the Min Max Plus Radius method, and the Min Max Plus Diameter method. The Min-Max plus Diameter method guarantees that all points selected will be inside the cylinder section of the capsule collider.

Example #2: Best Fit on a Skinned Mesh

In this example, we changed the Capsule Method field to Best Fit. In the Best Fit method, the first two points define the height of the capsule collider. In this case we selected a point on the foot, and one at the top of the head as seen in the image on the left.

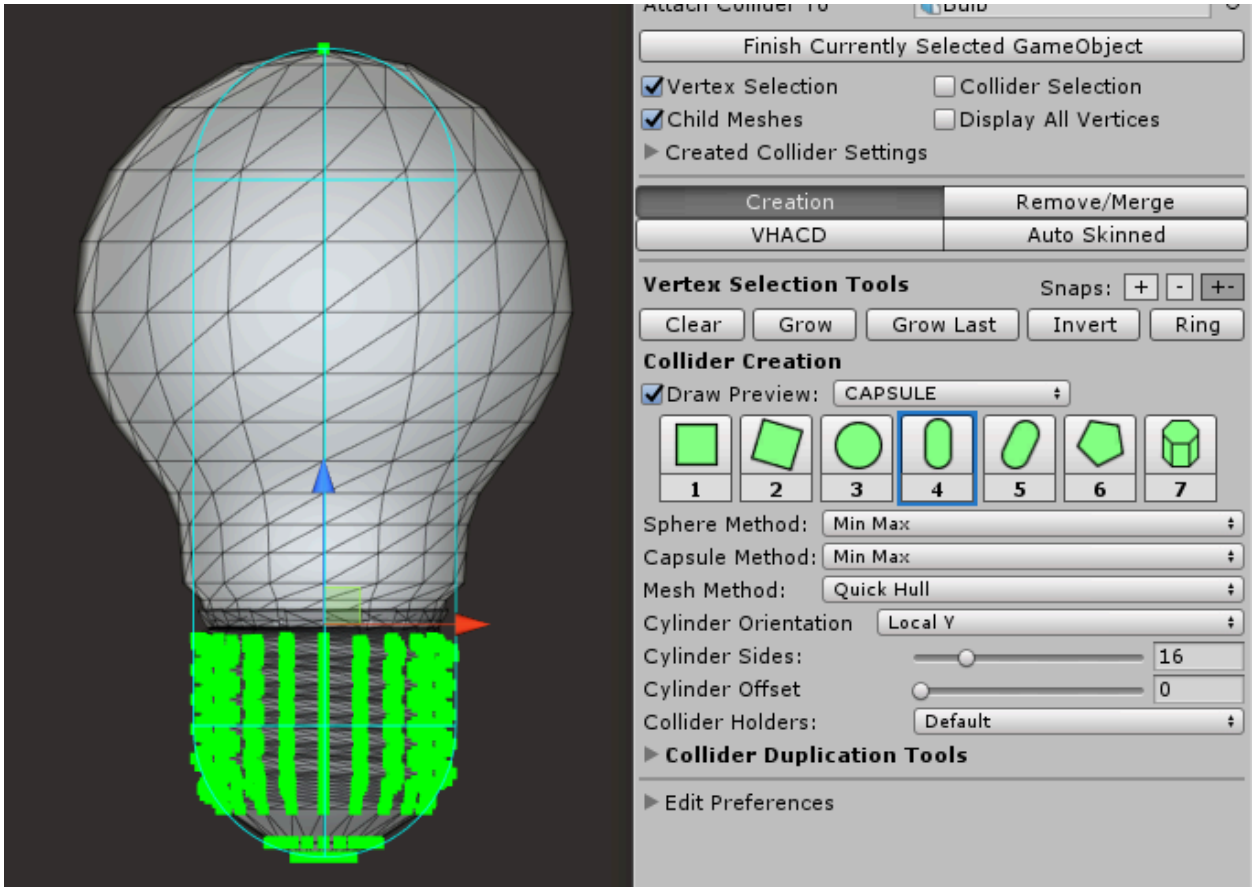


After this we selected two points in the mid section of the character, and used the Ring Select button to quickly get a set of vertices that would roughly lay on a sphere with a radius that matches the collider we want to create.

Note that after the first 2 points, the best fit capsule method uses the same calculation as the best fit sphere method. The first 2 points define the height of the capsule, while the remaining points are used to calculate the sphere that would best fit those points.

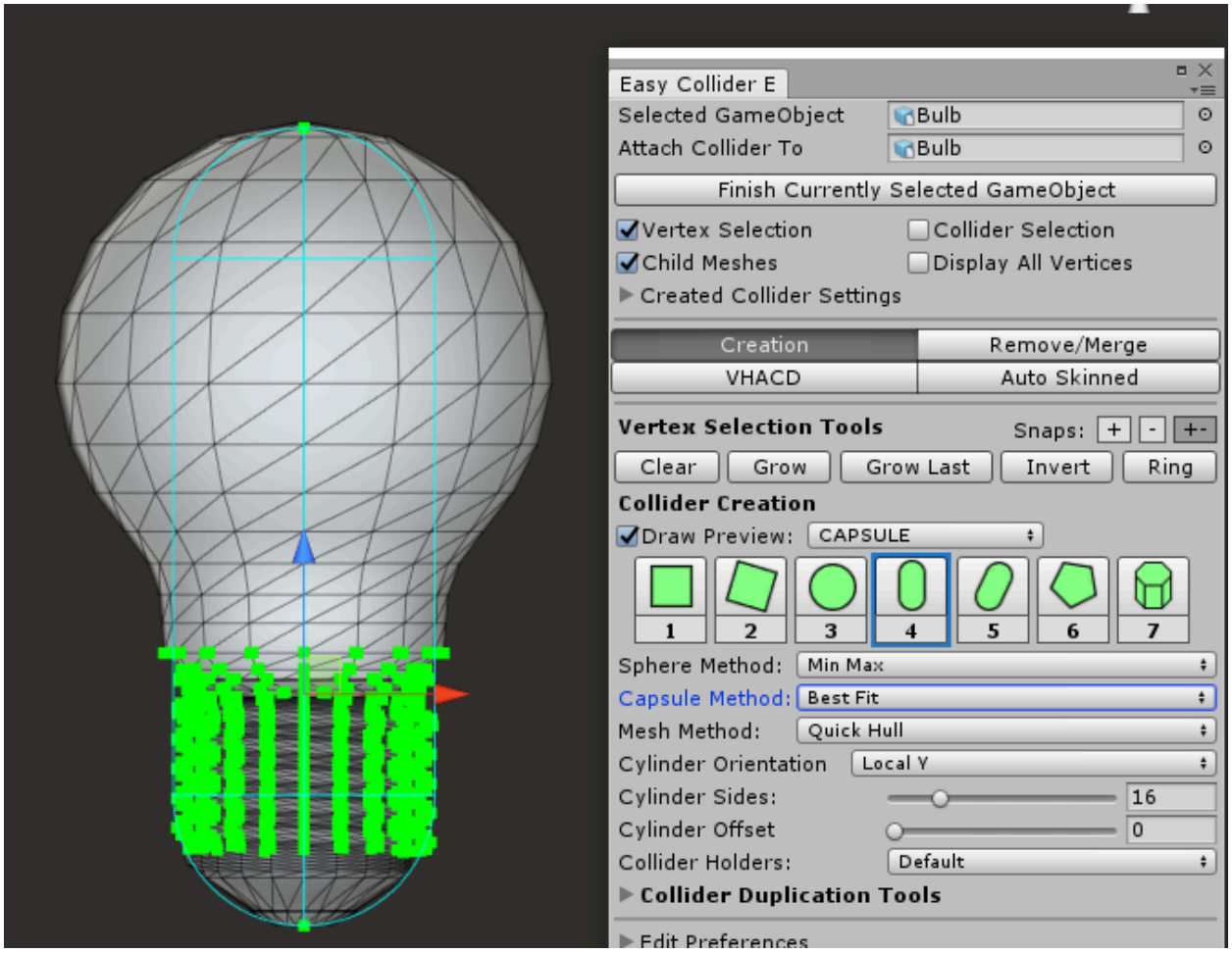
Example #3: Min Max on a Static Mesh

In this example we are looking to add a capsule collider to our lightbulb using the Min Max method. A box select was used to select all the points at the bottom of the lightbulb, and a single vertex was selected at the top of the bulb to make sure the capsule extends all the way up to it.



As you can see from the preview, this capsule would help correctly represent the full shape of the lightbulb in combination with a sphere collider.

Example #4: Best Fit on a Static Mesh



In this example we are trying to add a capsule collider to our lightbulb using the Best Fit method. The first two vertices selected define the height of the capsule, and were chosen to be the very bottom of the bulb, and the very top of the bulb.

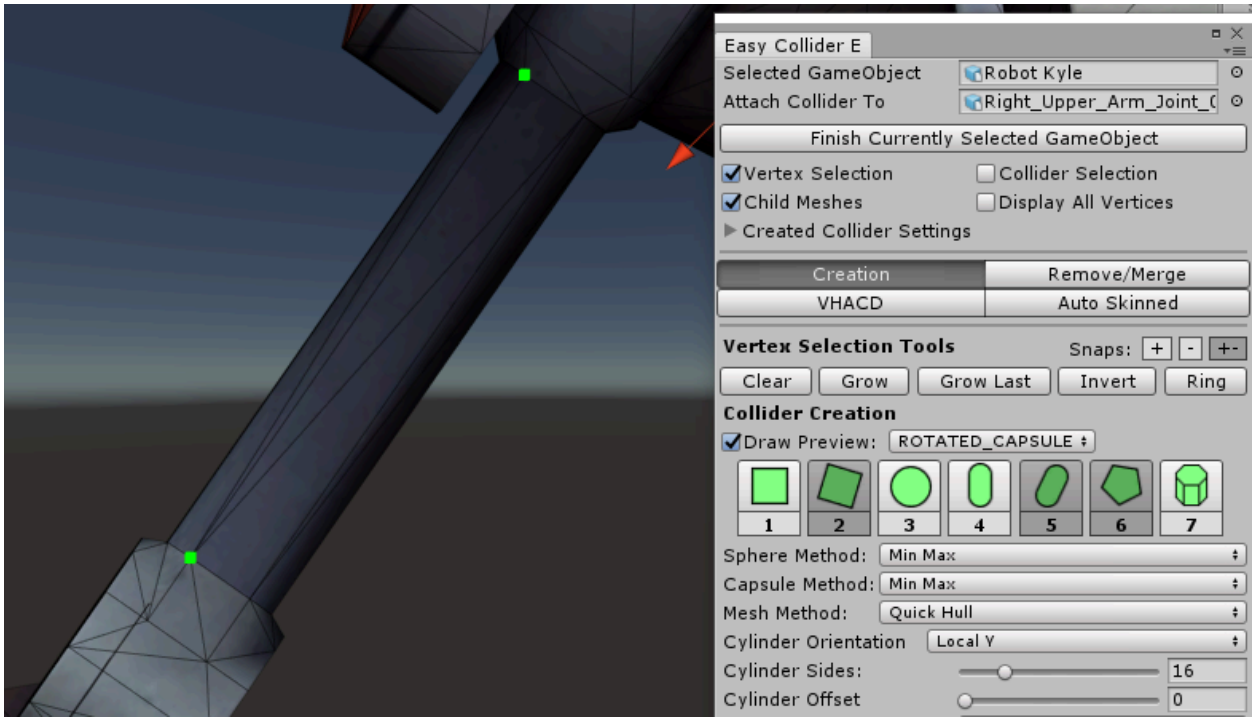
The other vertices selected should approximately lay on a sphere that has a radius similar to what we want our capsule collider to have. If we have selected a single ring around the bulb in any location, this would throw off the calculation significantly. This is because a single ring can be on a surface of a sphere of any radius equal to or larger than the ring itself. By selecting multiple rings, it allows the algorithm to better calculate the radius of a sphere that would best fit the points selected.

Rotated Capsules

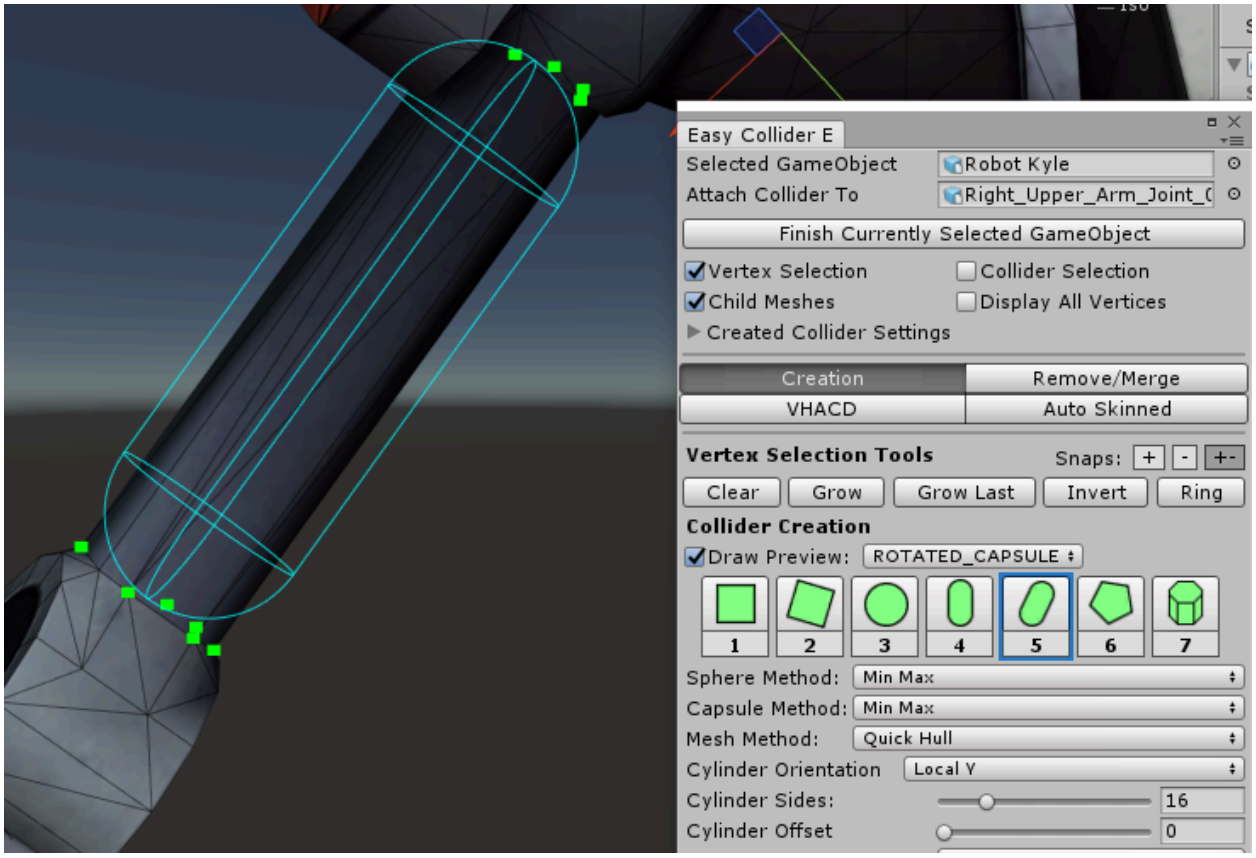
Example #1: Min Max on a Skinned Mesh

Again we're going to use our friend Robot Kyle. In this case we are again using the same right arm joint used in the rotated box collider example, except this time we're doing a rotated capsule. In this example we're also using the Min Max Capsule Method.

The idea of creating a rotated capsule is similar to that of creating a rotated box. Except you only need the first 2 points to define the rotation of the capsule as rotating a capsule around its height axis doesn't change the area it covers.



As you can see after selecting the two points, we don't have a preview yet. This is because to create a capsule collider you need at least 3 points selected, otherwise you just have a line! You can see in the image above that the two points chosen align with the arm of the mesh we want to create the capsule on.

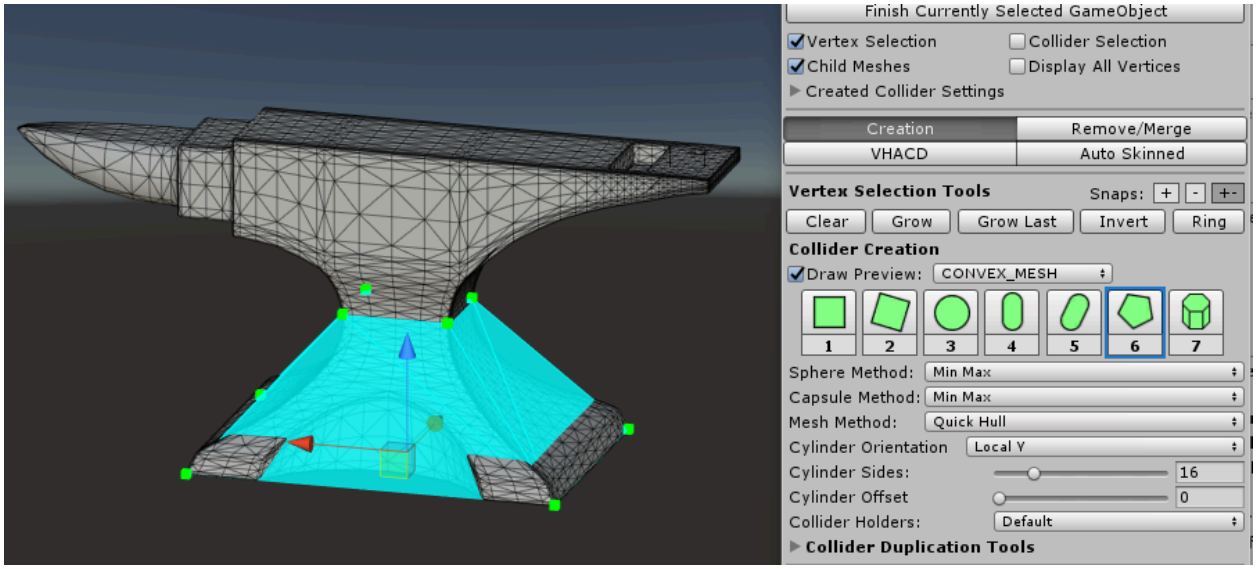


After selecting a couple more points around the arm, we have a preview that shows what the result will be.

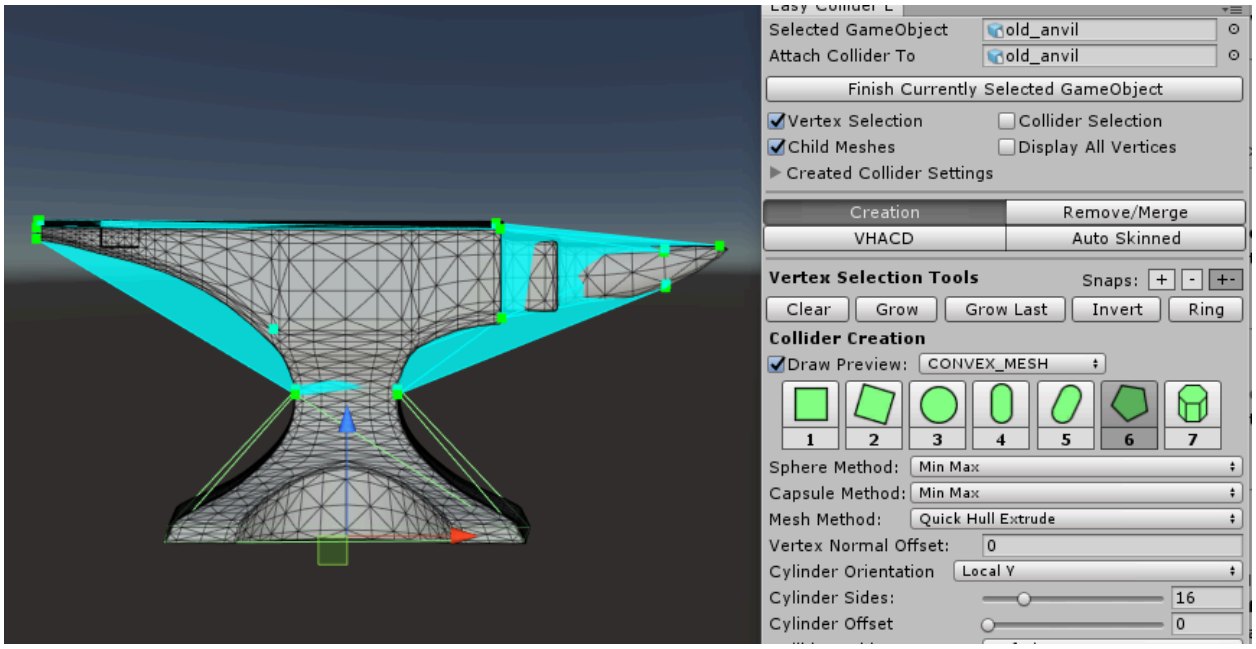
Mesh Colliders

Example #1: Static Mesh

In this example we are going to create several convex mesh colliders to better approximate the shape of an anvil.v



In the image above, 8 points were selected to create as simple a convex mesh as possible while still maintaining a rough outline of the mesh. More points could be selected to represent the shape of the mesh, but this would also increase the complexity of the resulting convex hull and thus reduce performance.

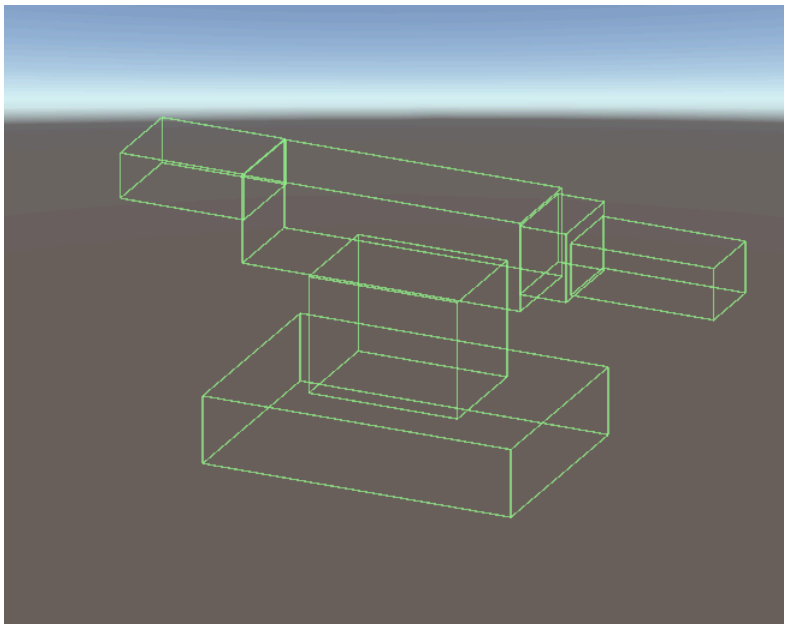


We then selected a minimal amount of points at the top of the anvil to roughly approximate the shape. Keep in mind that the result is used for a convex mesh collider, so the output will only be a convex mesh.

We want to select the least amount of points, as complex convex meshes are bad for performance.

Alternatively, if we don't want to create convex mesh colliders ourselves by manually selecting points, we can use the VHACD section of this asset. See the VHACD section of this documentation for more information.

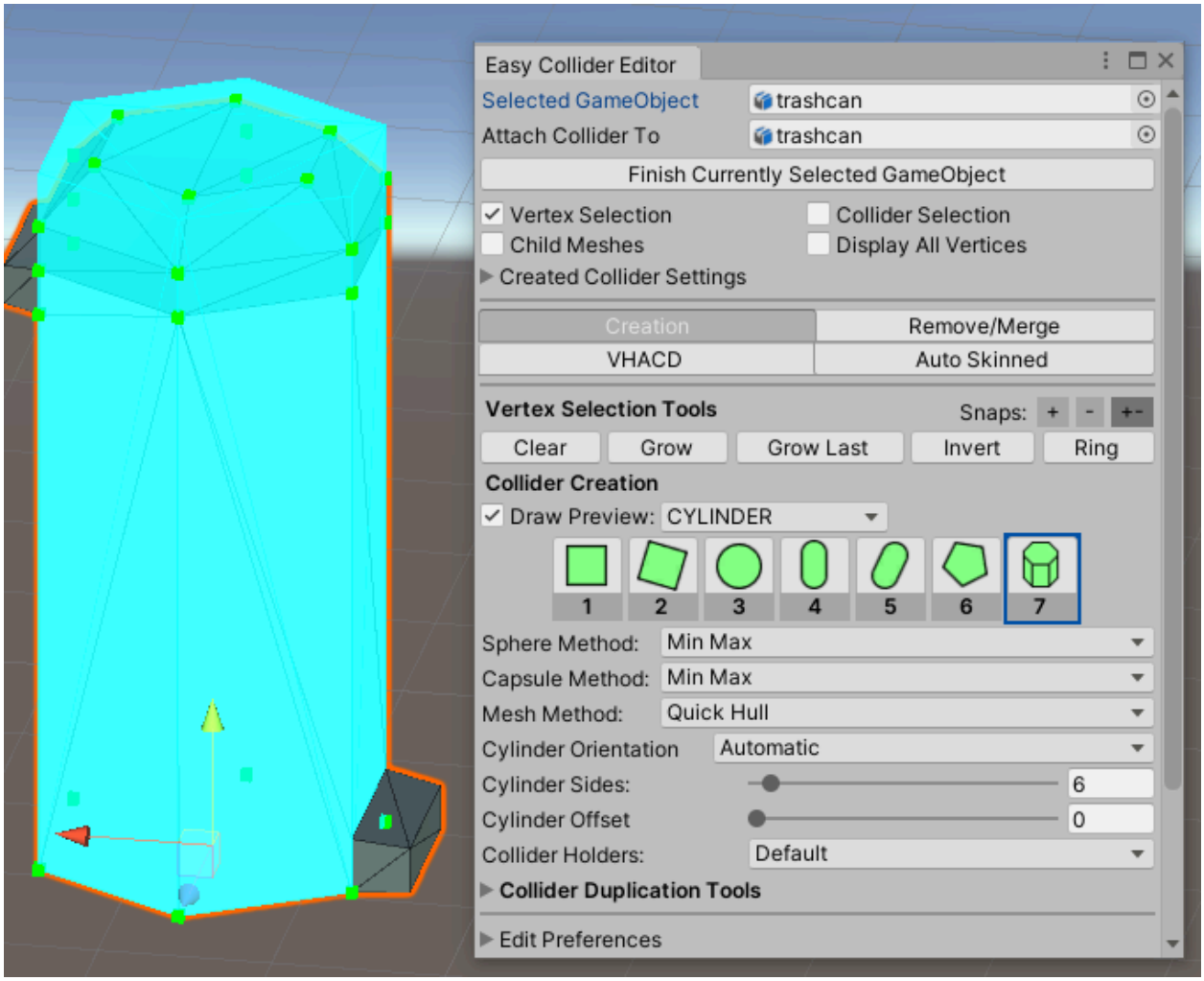
If this were not a demonstration, we should think about whether we really need mesh colliders to represent a mesh of this shape. As an example, the result of quickly using only box colliders on the anvil mesh is shown below. The approximation of the shape is good, and would perform better than an equivalent number of convex mesh colliders. Each game is different though, and it may be the case that using convex mesh colliders does not have any large effect on performance for your game.



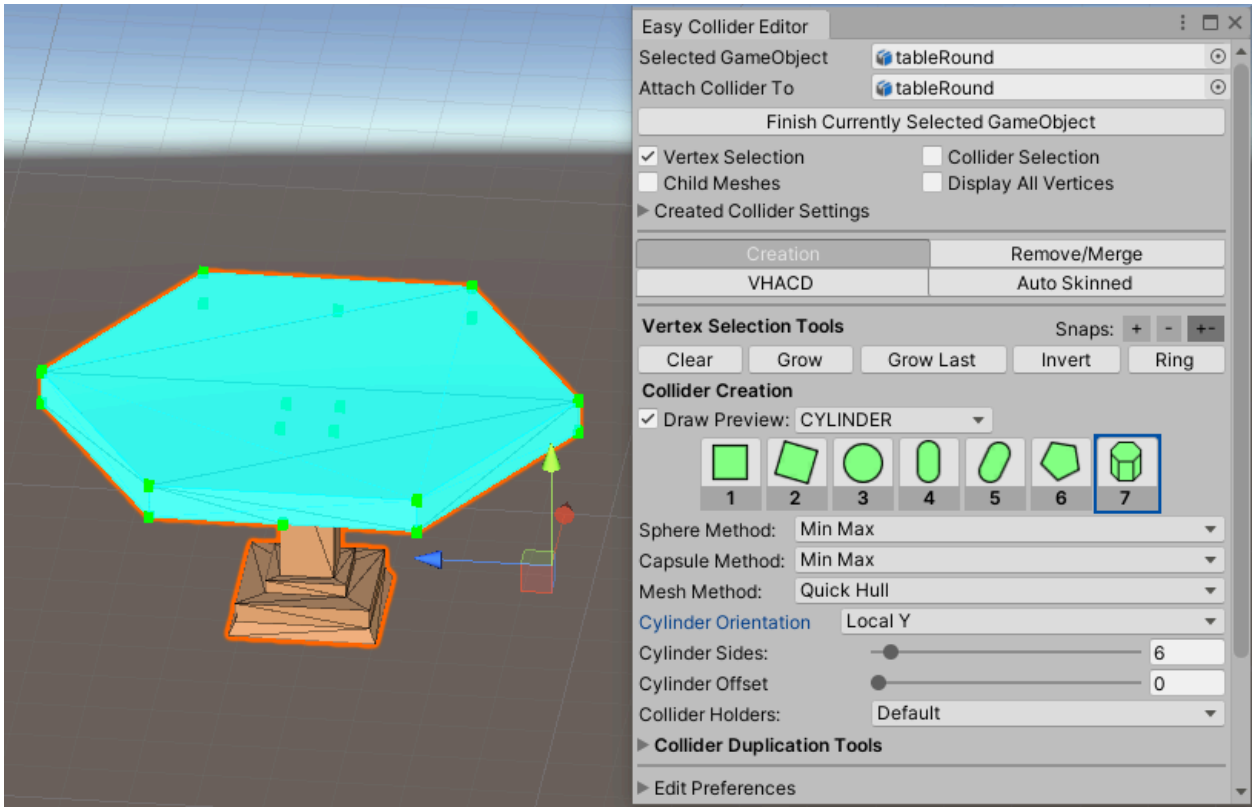
Cylinder Colliders

Example #1: Cylinder on static meshes

In this example we're going to create a cylinder shaped collider on a static mesh. One thing to keep in mind with cylinder colliders is they are simply convex mesh colliders in the shape of a cylinder. This means that they come with the performance drawbacks of convex mesh colliders when used excessively. However, each game is different and it may not matter for yours!



The trash-can in the image above is a great example of where a cylinder collider could potentially be used. We can quickly select as many vertices as we wish and they will all fit inside of the cylinder. Since the trashcan is taller than it is wide, the automatic cylinder orientation setting aligns the cylinder correctly.



In contrast creating a cylinder collider for the top of the table mesh shown above would normally result in a cylinder pointed in the wrong direction, as it is wider than it is tall. In this case, we have changed the cylinder orientation to Local Y to correctly orient the cylinder with the table's top.

If the number of sides wasn't the same as the default value of 6, we could increase or decrease the number of sides using the cylinder sides parameter.

If the sides of the table didn't align with the sides of the cylinder, we could adjust this with the cylinder offset parameter, spinning it around its axis until it aligns.

Merging Colliders

These tools function almost exactly the same as the tools for creating colliders, the only difference is that instead of selecting vertices, colliders are selected.

Using the only add or remove keys (or ctrl/alt) from preferences can let you select colliders that are inside of other colliders more easily. This can be used so that you don't have to try to adjust the scene's camera view to see correctly inside of the covering collider to select an internal one.

Merging colliders is done by selecting several colliders and merging them into a single collider. This can be helpful as you may only be able to identify which colliders could be simplified into a single larger collider after the creation process is completed.

One primary reason you may want to merge colliders is to reduce the number of convex mesh colliders in your scene. Although it is best to also limit the complexity of the convex mesh colliders by selecting fewer points, sometimes simply merging multiple convex mesh colliders into a single one works as well.

Additionally, since VHACD does not give super precise control over where convex mesh colliders are placed, it may create extra mesh colliders that are not necessarily needed. Some of them could likely be merged together to reduce the number of convex mesh colliders used without significantly affecting the actual accuracy of the colliders.

If you wish to merge colliders into a rotated collider, be sure to select the rotated collider first.

Editing Colliders

Editing colliders can be done through the Remove/Merge tab. Simply select a collider, and click the edit button. This will convert the collider to selected vertices, remove the collider, switch tabs to the creation tab, and finally automatically change the preview to the type of the first collider selected.

An important thing to remember is that when clicking the edit button, the selected colliders are removed. Although this is undoable, if you enter into editing collider(s) but don't create any colliders after selecting more points, the colliders will still be removed.

Once you've entered into editing a collider everything behaves just like normal collider creation. So be sure to read the Selecting Points and Vertices, and Collider Creation documentation for any questions you may have about that process.

Generating Convex Hulls - VHACD

VHACD is a method that can be used to automatically generate multiple convex mesh colliders. Unfortunately since VHACD is auto-magic, you can't specifically tell the process which parts of a mesh are important and which are not. This can lead to some frustration with meshes where VHACD does not work as well as you would like.

Fortunately, I have added some additional tools and have some helpful advice to make it easier to work with problem meshes. The "Use Selected Verts" toggle allows you to select vertices and

only put those through VHACD instead of the whole mesh at once. If you have any other ideas that might make VHACD easier for you to use, please let me know!

General Settings

Using VHACD for generating convex hulls uses the Selected GameObject and Attach To fields just like normal collider creation.

Resolution changes the amount of voxels that are used to calculate the convex mesh colliders. A higher resolution will generally be better at finding the bounds of the mesh. It is also the setting that when raised, increases the calculation time substantially. This is why it is limited to 128k without expanding the advanced settings. Although the maximum resolution is 64 million, using this value can take a very long time to complete.

Max Convex Hulls is the maximum number of convex hulls that will be generated for each mesh. A higher number of convex hulls will capture the shape of a complex mesh better than a lower number.

Max Vertices per Hull is the limit of vertices of each convex hull mesh that is created. Keep in mind that convex mesh colliders over 256 vertices or triangles can cause errors in some versions of unity.

FLOOD_FILL will generally give you the best results on all meshes where you are not using the advanced vhacd settings with a sufficiently high resolution.

There are 2 other fill modes available in VHACD: RAYCAST_FILL is a good alternative when your mesh contains holes as it uses raycasts to determine what is inside or outside the mesh. SURFACE_ONLY is useful when you want the convex hulls to cover only the surface of the object and not the interior, creating a hollow object using a high number of convex hulls.

I personally haven't had good results using any method other than FLOOD_FILL, but your results may vary.

Project Hull Vertices

Project Hull Vertices causes the resulting convex hull's vertices to lay on the surface of the source mesh instead of outside of it. This allows for increased accuracy in cases of lower resolution.

Save Hulls as Assets

Allows the meshes generated by VHACD to be saved in your project. If the meshes are not saved they will likely be lost on reload and the mesh colliders will not function.

Separate Child Meshes

This toggle can be used to generate convex hulls on multiple meshes at once using the same settings. When this option is enabled, the Attach Collider To field is ignored. Instead each mesh is individually run through convex hull generation. The resulting convex hulls from each generation are then attached to the transform of the mesh.

Use Selected Verts

With this toggle enabled, the vertex selection tools will also appear in the VHACD tab. This allows you to select vertices on the mesh directly. This allows you to portion out sections of a large and complex single mesh for VHACD.

Attach Methods

There are 3 attach methods available.

The default “Attach To” method creates all convex mesh colliders on the gameobject specified in the Attach To field.

The Child Object method attaches all colliders to a single gameobject as a child of the Attach To object.

Individual Child Objects creates a child object of the Attach To object, and then a child object for each individual convex mesh collider.

When “Separate Child Meshes” is enabled, these function the exact same. Except now, an additional option is displayed labeled “Per Mesh”. When this is enabled, the attach methods function similarly, except as each mesh is run through VHACD the Attach To object changes automatically to the GameObject the mesh is found on. This allows you to create colliders for child meshes and attach them directly to the objects those meshes are from. That way, if the child meshes move in the future, the colliders will move with them.

Advanced VHACD Settings

Advanced Settings

Expanding the Advanced VHACD Settings exposes all of the available parameters. Expanding the advanced settings also allows for a higher maximum resolution, and number of vertices. Adjusting these values is only recommended if you are not getting the results you are looking for with the normally exposed settings.

The button labeled default will reset the VHACD parameters back to their default settings.

Force <256 Triangles

When a convex hull is generated that has more than 256 triangles, the max number of vertices is lowered and the computation is done again. Only a maximum of 3 calculations is done before it allows over 256 triangles. This option can be disabled if you need more than 256 triangles, but it is not recommended.

Why is this done? Mesh Colliders with convex hulls that have more than 256 triangles generate errors in some versions of unity.

Default VHACD Settings

The explanation for the parameters is from the official repository of VHACD and gives the best explanation of the parameters that I can provide.

Parameter	Description	Range	Default
Concavity	Maximum concavity	0 - 1	0.0025
Alpha	Controls the bias toward clipping along symmetry planes	0 - 1	0.05
Beta	Controls the bias toward clipping along revolution axes	0 - 1	0.05
Min Volume Per Convex Hull	Controls the adaptive sampling of the generated convex-hulls	0 - 1	0.0001
Plane Downsampling	Controls the granularity of the search for the best clipping plane	1 - 16	4
Convex Hull Downsampling	Controls the precision of convex hull generation process during the clipping plane selection stage	1 - 16	4
Resolution	Maximum number of voxels generated during the voxelization stage	10k - 6400k	10000
Max Convex Hulls	Maximum number of controls hulls to produce	1 - 128	1
Max Vertices Per Hull	Maximum number of vertices used per convex hull.	4 - 1024	64

Saving and Loading VHACD Settings

Settings can be saved or loaded to preserve all of the parameters above between multiple sessions. This can be useful if you want to use the same settings when creating colliders across the majority of objects. Click the load settings button to load VHACD settings, and the save settings to save them for a future session.

If you overwrite the default settings, you can still revert to the default with the default button. You can then re-save the settings if you wish.

VHACD Preview

The preview should be automatically updated as the parameters for the calculation change.

One thing to note when using the VHACD preview is that the resolution is limited to a range of 10,000-128,000 (due to computation time). Differences in the result of final high resolution calculations and the preview calculation is usually not significant, especially if using project hull vertices.

Saving Convex Hulls

If the option to save hulls as assets is enabled, the convex hull meshes that are generated are saved as .asset files and used as the source mesh for the mesh colliders. It is highly recommended to save them as asset files, otherwise they will eventually be lost and the mesh collider component will not work.

The folder the asset files are saved in can be changed in preferences. There are several different save methods that can be used to tell the save location search where you want the files to be saved. The documentation of the preferences UI explains how these work.

When the Separate Child Meshes toggle is enabled along with the per mesh option, the save method “Mesh” will try to save each hull at each individual mesh’s path. For example: If you are generating colliders on multiple objects at once, the convex hull meshes would be saved in the same folder as the original source mesh, if it is found.

During preview no convex hulls are created or saved, only when you click the button to generate the colliders are the assets created and saved. Undoing the creation does not remove the assets that were created, and re-clicking the button to generate colliders creates additional asset files. To clean up the created assets, use the asset browser to find where your convex mesh assets are stored. Since they are ordered by name and saved in the same folder for each object this is relatively easy to do. I am looking into methods to reduce the need to manually clean up these asset files in some instances, so hopefully this will improve in the future.

To reduce the number of asset files generated, there is an option within Preferences that combines the meshes generated in a single use of the Generate Convex Mesh Colliders button into a single .asset file.

VHACD - Converting to Primitive Colliders

Currently this asset also supports converting the convex-mesh colliders of vhacd into primitive colliders (boxes, spheres, and capsules).

Conversion takes the convex-meshes that are output from VHACD and passes the vertices into the methods that create colliders. Because the output from VHACD can only be controlled by parameters, it is not possible in most cases to get precise results. However, converting to box colliders does allow you to more quickly create compound colliders than possible with the collider creation tools.

VHACD - General Tips

It can be occasionally frustrating working with VHACD to get it to correctly handle certain types of meshes. The following are tips that come from my experience when working with VHACD.

Objects like cups, bowls, mugs, or objects with holes can be difficult for VHACD to process. I recommend increasing the maximum number of convex hulls, and increasing the alpha and beta parameters.

Single meshes that are very large and complex are also difficult for VHACD to process accurately. Other than increasing the maximum number of convex hulls, there's not much that VHACD can do by default. I have added the Use Selected Verts toggle to help with these situations. With this toggle enabled, the normal vertex selection tools show up, and vertices and points can be selected as normal. This can be helpful to portion out sections of a mesh for vhacd to process.

Unfortunately, a lot of the usage of the advanced parameters comes from experience. Luckily, this asset uses the performance branch of vhacd, calculates in the background, and shows a preview as the parameters change. This can allow you to try out a variety of meshes quickly and play around with the parameters to get a little bit of a better understanding of what each does.

Sometimes, VHACD just does not meet your needs regardless of how much one fiddles with the parameters. In these cases, it's best to switch to the creation tab and try manually creating colliders. Convex mesh colliders can be created manually as well, and also have a preview drawn in creation mode to aid in the process.

VHACD - Performance Considerations & Tips

Convex mesh colliders are expensive compared to primitive colliders. If you're extremely worried about performance, the best option is to use no convex mesh colliders and only use primitive colliders. Otherwise, the next best option is to use as few convex mesh colliders as possible to achieve what you need to achieve.

Ultimately the performance impact of using convex mesh colliders depends on how many and the complexity of those colliders. Changing all your convex mesh colliders to compound primitive colliders can have no impact on performance as your performance issues are elsewhere. The best way to profile your game is to use Unity's profiler. More information can be found here: https://docs.unity3d.com/Manual/Profiler.html

If you find that you're using a lot of your frame time budget for physics calculations, and you're using a lot of convex mesh colliders, performance can be improved by reducing the number of convex mesh colliders. Selecting multiple colliders can be merged into a single convex mesh collider using the merge tab, which will likely increase performance. But this will also change the coverage and shape of the colliders.

Converting them using the merge tab to box colliders, sphere colliders, or capsule colliders where appropriate can help performance as well. Removing them all and manually creating the same primitive colliders also achieves the same result. Manually creating primitive colliders can take time, but gives you full control over the balance of collider accuracy and number of colliders.

Ultimately the best answer for how many convex mesh colliders is okay to use is impossible to answer. It's possible for your game to use them on everything and be fine, and it's also possible that you have to use none.

If you're wondering what I personally do in my projects, I use VHACD first as it's extremely fast to get accurate collisions up and running this way. If it's discovered that it's causing significant performance impacts, I remove the convex mesh colliders and manually create colliders.

Ultimately the importance and accuracy requirements of collision geometry differ from game to game, and the accuracy of the collision geometry depends on the amount of time spent creating it. Although the tools in this asset make it significantly faster compared to the built-in tools, it still can require some time to get accurate results.

VHACD - Examples

For simple meshes, it's much easier to play around with the VHACD settings and see the results of the changes almost immediately because of how VHACD works with the preview in this asset.

Because of this the current examples focus on larger, complex meshes although the same principles can be applied to simpler meshes / objects as well.

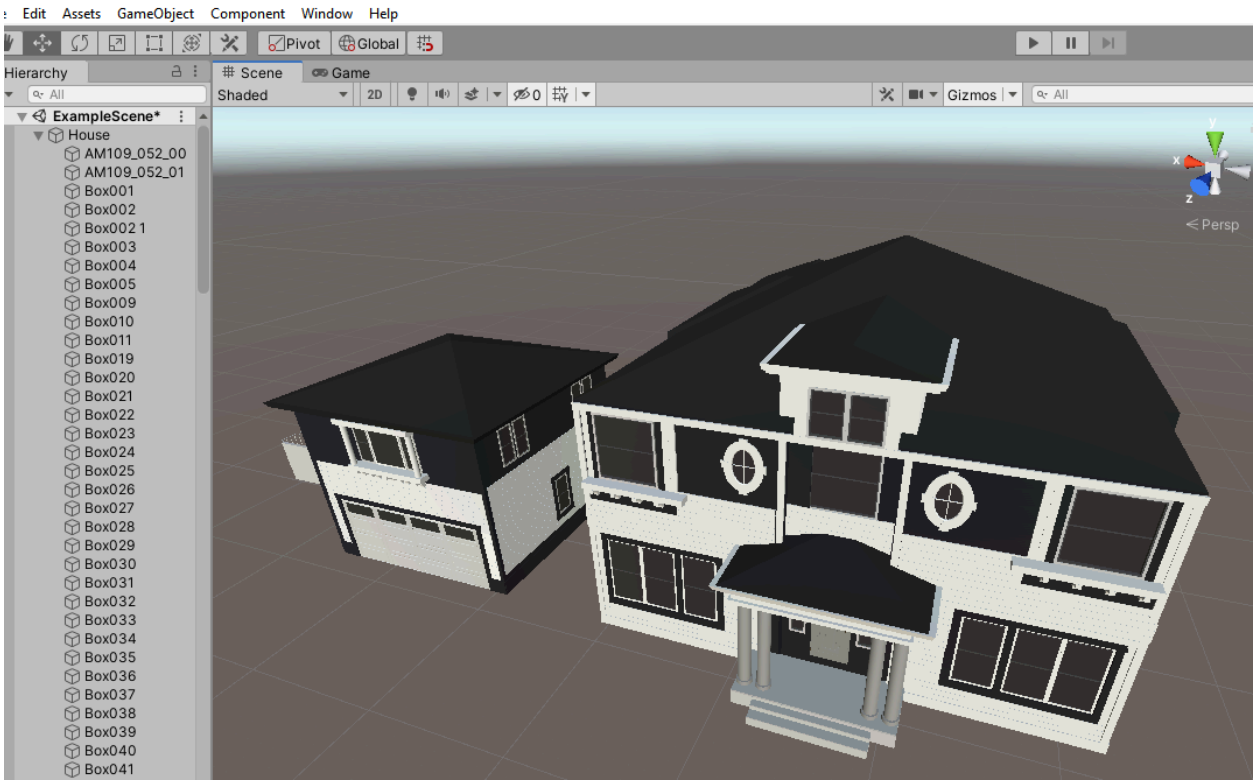
If you are using prefabs, remember that the best results will of course come from creating colliders for each prefab. Using prefabs and creating colliders on the prefabs themselves will generally give significantly better results. Additionally if everything is made of prefabs, when you create the collider on the prefab, it automatically gets applied to every instance of that prefab, making iteration significantly easier.

Since objects that use prefabs would likely be made up of smaller objects, the examples will focus on large complex objects that were for some reason not created with prefabs. This can easily be the case when the object was created together in an external program, and has not yet been separated and converted into prefabs.

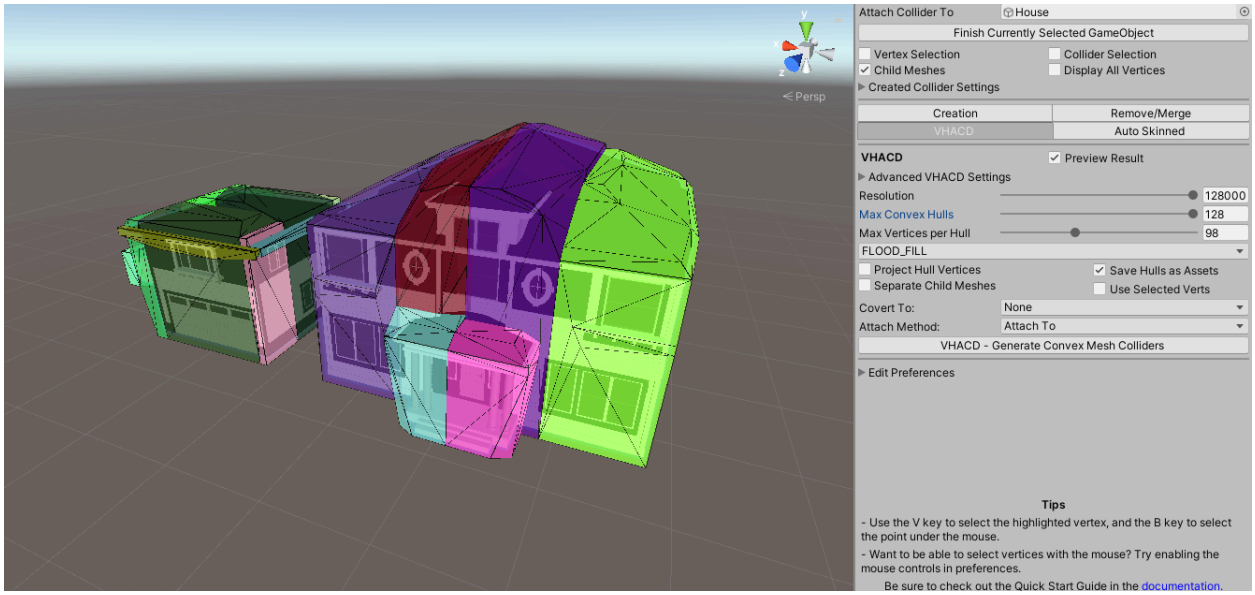
All of the example images have purposefully disabled the project hull vertices toggle so that the result is more visible on top of the mesh.

Large House Mesh

In this example, we're going to look at a large multi-building house mesh. In this case our object was unfortunately not created with prefabs.



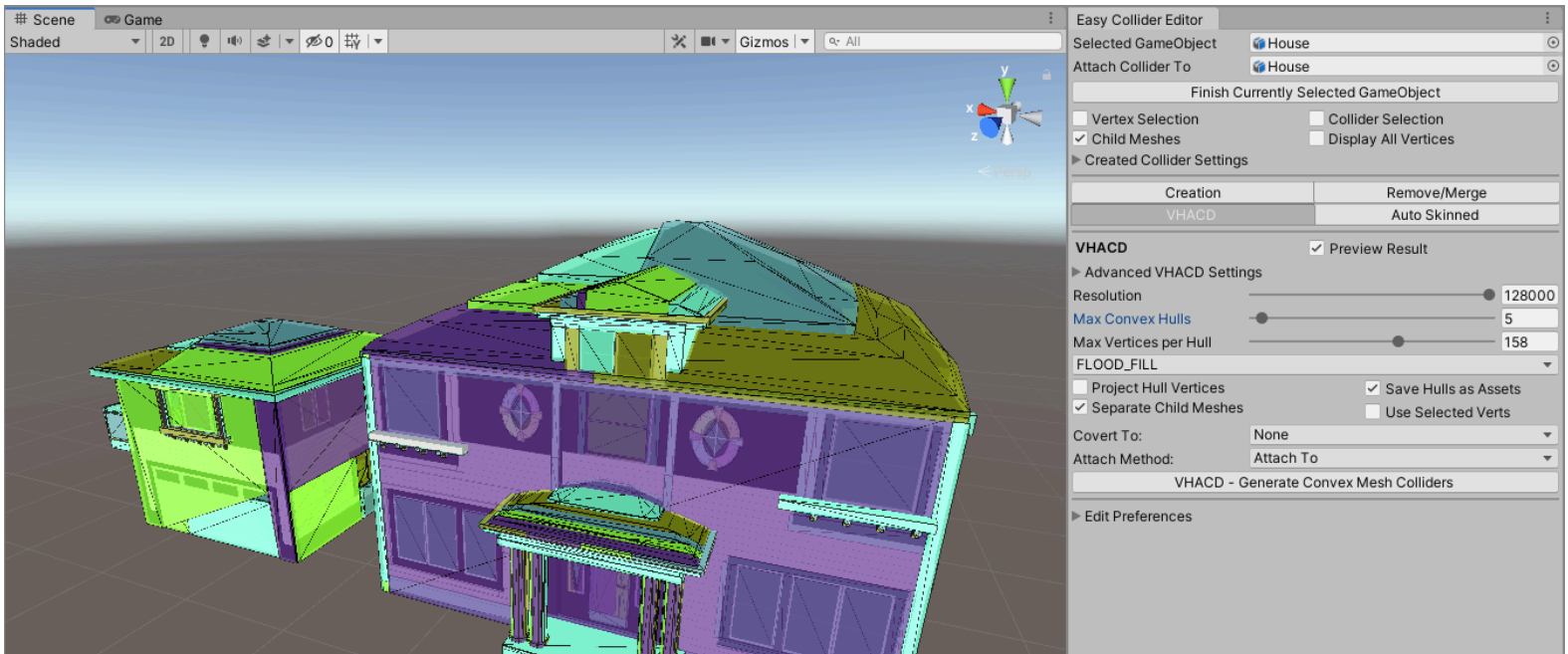
As you can see in the image above, this house is made from many small objects, none of which have been separated into prefabs.



In this image, we have enabled child meshes and increased the maximum number of convex hulls. As you can see, the result is quite poor! Luckily there are 2 additional options we can use to improve the results.

Large House Mesh - Separate Child Meshes

In this case, since the large house mesh is made up of many smaller meshes, we can use the separate child meshes toggle.



In the image above, you can see that the Separate Child Mesh toggle was enabled, and the max convex hulls value was decreased significantly. This is because each individual mesh is run through VHACD separately using the same settings. This result is significantly better than the previous one.

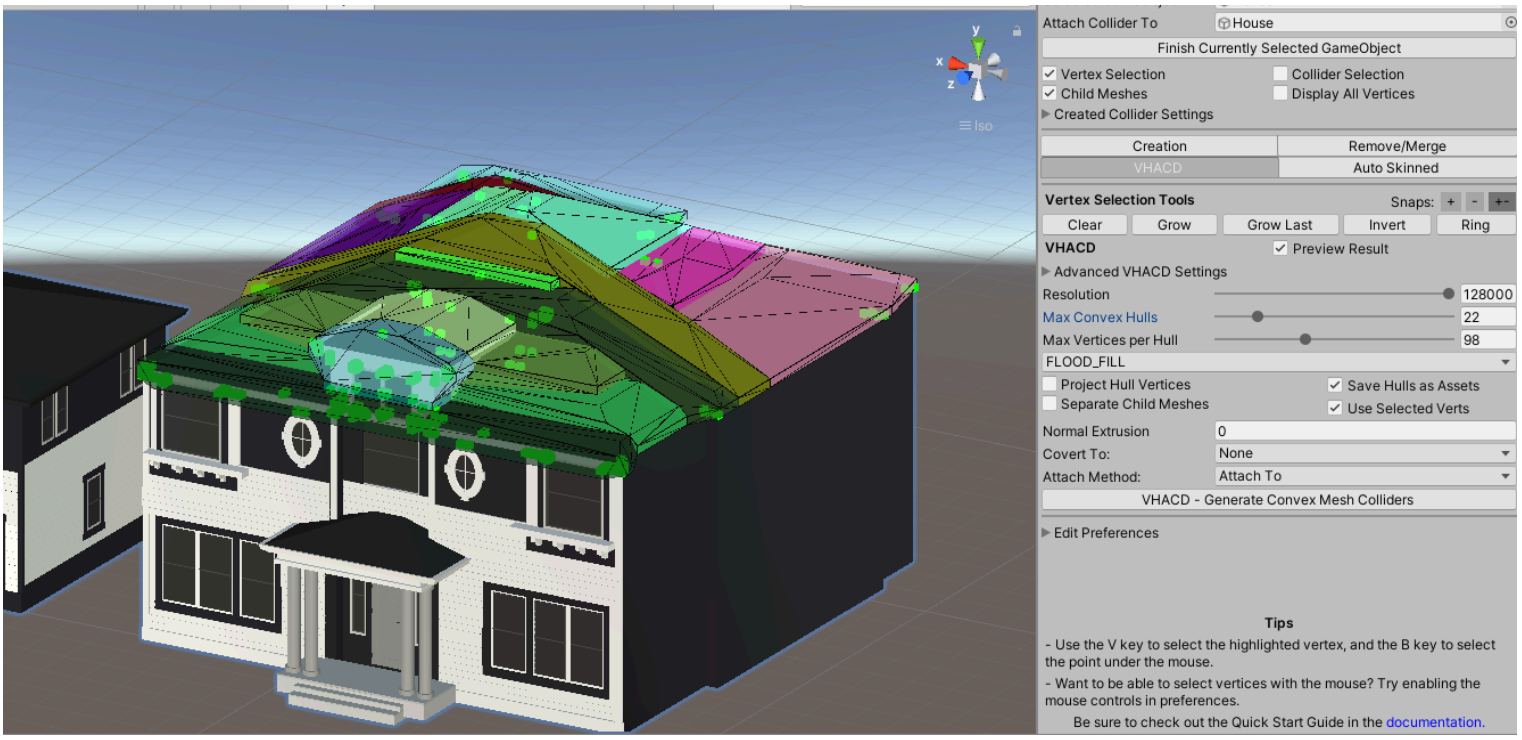
For better results, the next step would be to expand the advanced settings, increase the resolution, and let the collision generate on the 200+ meshes that are a part of this house over a longer period of time.

Large House Mesh - Use Selected Vertices

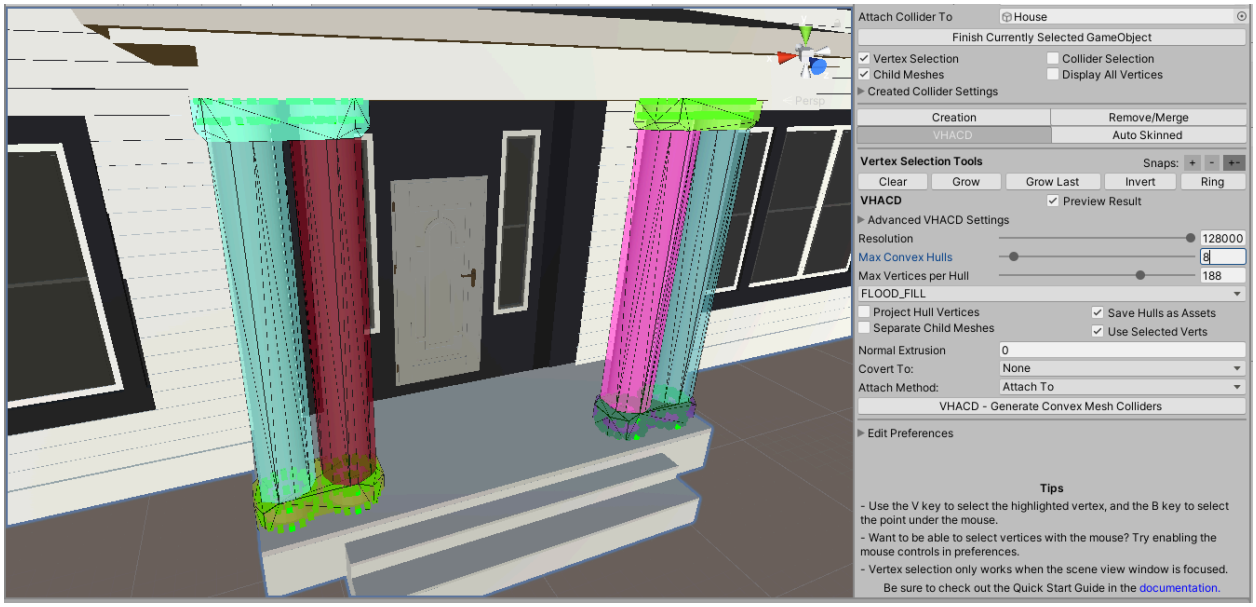
Even though the house is made up of many smaller meshes, in this example we're going to treat it as if it is all one single complex mesh.

With a large single complex mesh, the best solution is to enable the "Use Selected Verts" toggle. When this is enabled, the standard Vertex Selection Tools can be used. In this case, the box selection (click and drag) and the grow and grow last buttons will be very helpful.

The goal when using "Use Selected Verts" in VHACD is to portion out sections of a mesh for VHACD to handle separately. Allowing the smaller details to be picked up easily.



In the image above the top part of the house was selected by going into front view isometric mode and using box selection to quickly select the whole top part of the mesh.



In this image, vertex selection along with CTRL + clicking the grow / grow last buttons were used to quickly select all of the vertices of the pillars on the front of the house. This allows you to quickly select all the vertices of an object, providing collision detail in areas of a mesh that need it most.

Large House Mesh - Final Thoughts

In this case, since the object is really created from many smaller separated meshes, the best option to use is the separate child meshes toggle.

With the use selected vertices option, the most important selection tips I can give are to use isometric mode to select portions of the mesh, and CTRL+click the grow last button to quickly grow a single vertex selection to select all connected vertices in areas where you need more detail in the collision geometry.

If I were to use this object in a real project, the first step I would likely take would be to create prefabs of the reusable parts and create the colliders on those objects.

Depending on the size of the game, I would also try to avoid using convex mesh colliders as much as possible for performance reasons. I would likely create primitive colliders (boxes, capsules, spheres) for each section of the house using the creation tab of this asset. The only area I would consider using convex mesh colliders would likely be the roofs of the houses.

During the start of any project, I find it helpful to use VHACD to quickly generate approximate colliders that are “good enough”, and then over the course of a project create more finalized colliders on the prefabs that are being used.

Ultimately the importance and accuracy requirements of collision geometry differ from game to game, and the accuracy of the collision geometry depends on the amount of time spent creating it. Although the tools in this asset make it significantly faster compared to the built-in tools, it still can require some time to get accurate results.

Auto Generated Skinned Mesh Colliders

Auto generating skinned mesh colliders is a simple process. Simply set the Selected GameObject field to the gameobject that either has a skinned mesh renderer or has a skinned mesh renderer as a child.

Allow Realign / Minimum Realign Angle

The allow realign toggle allows transforms to be added as children that are better aligned with the bone chain than the transforms of the bones. Usually this is not the case, but sometimes you don't have the time to go back and fix a complex bone chain. The minimum realign angle is the angle the current bone's axis must be away from the next bone in the chain to be realigned. A minimum angle of 0 means that only perfectly aligned bones will not have child gameobjects added to hold the colliders.

Depenetration

Enabling this option creates colliders that do not overlap with one another.

It works by iteratively shrinking and shifting the resulting colliders while doing penetration tests. The amount of shrinking vs shifting can be controlled with the slider, while the order the bones are processed can be controlled with the dropdown.

Minimum Bone Weight

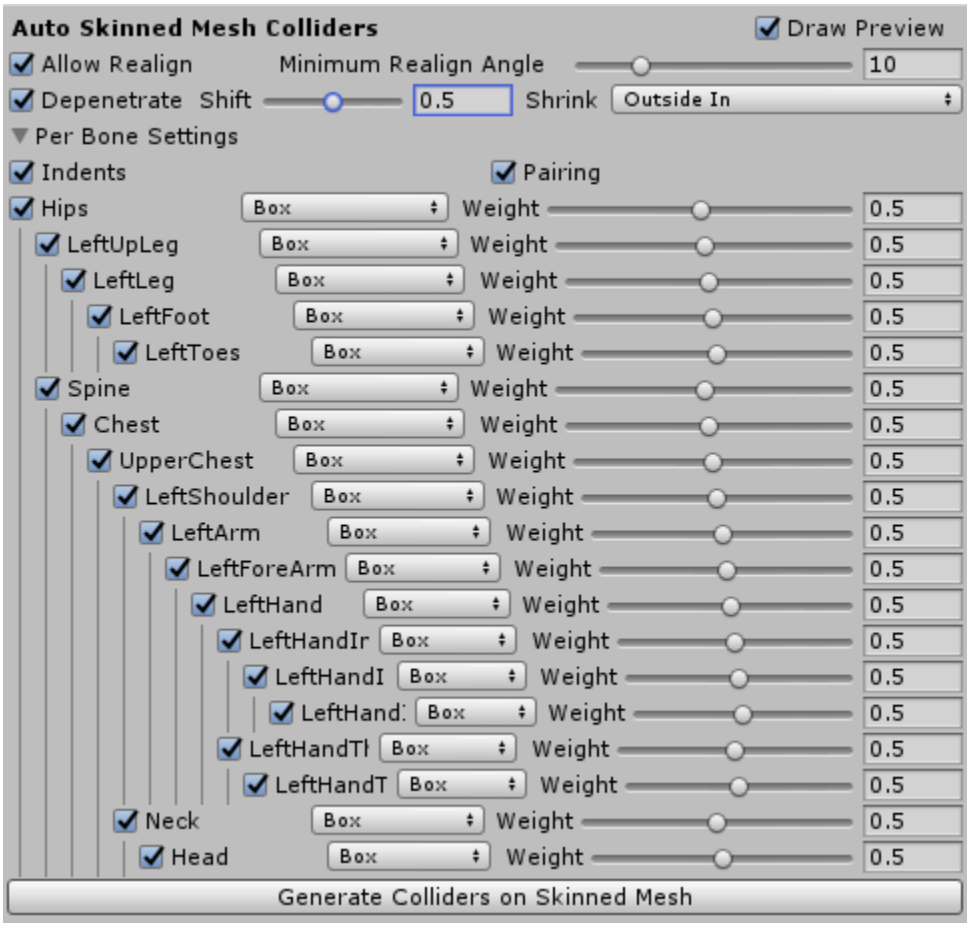
The minimum bone weight value is the minimum weight a vertex must have for a bone to be included in the collider calculations. If this value is set to 1, only vertices which are controlled by a single bone will be included in that bone's collider generation.

Collider Type

The dropdown can be used to change the type of colliders being generated for the skinned mesh.

When CONVEX_MESH is selected, an option appears to force < 256 triangles. This can be used in cases where the resulting colliders go over the triangle limit and generate errors in some versions of Unity.

Per Bone Settings:



Expanding the per bone settings makes it possible to have different vertex weights, and collider types on each bone. It also allows you to disable or enable generation on each bone.

The pairing toggle enables automatic pairing, and can be extremely helpful. Paired bones get hidden and automatically adjusted when the displayed parameters are adjusted. This is helpful for things with limbs where when adjusting one leg, you want to adjust both without having to manually copy values. There are some cases where bones are paired incorrectly, in which case you will want to disable bone pairing.

Holding control and adjusting any parameter will adjust that parameter on all of its children as well. This can be used to enable (or disable) generation on multiple bones, adjust the collider type, as well as the minimum vertex weights.

If your skinned mesh is currently imported with the Optimize Game Objects setting enabled you must disable this setting, generate colliders, then enable it again. The generated colliders will be correctly transferred to the exposed transforms. Only transforms that are exposed will retain the generated colliders.

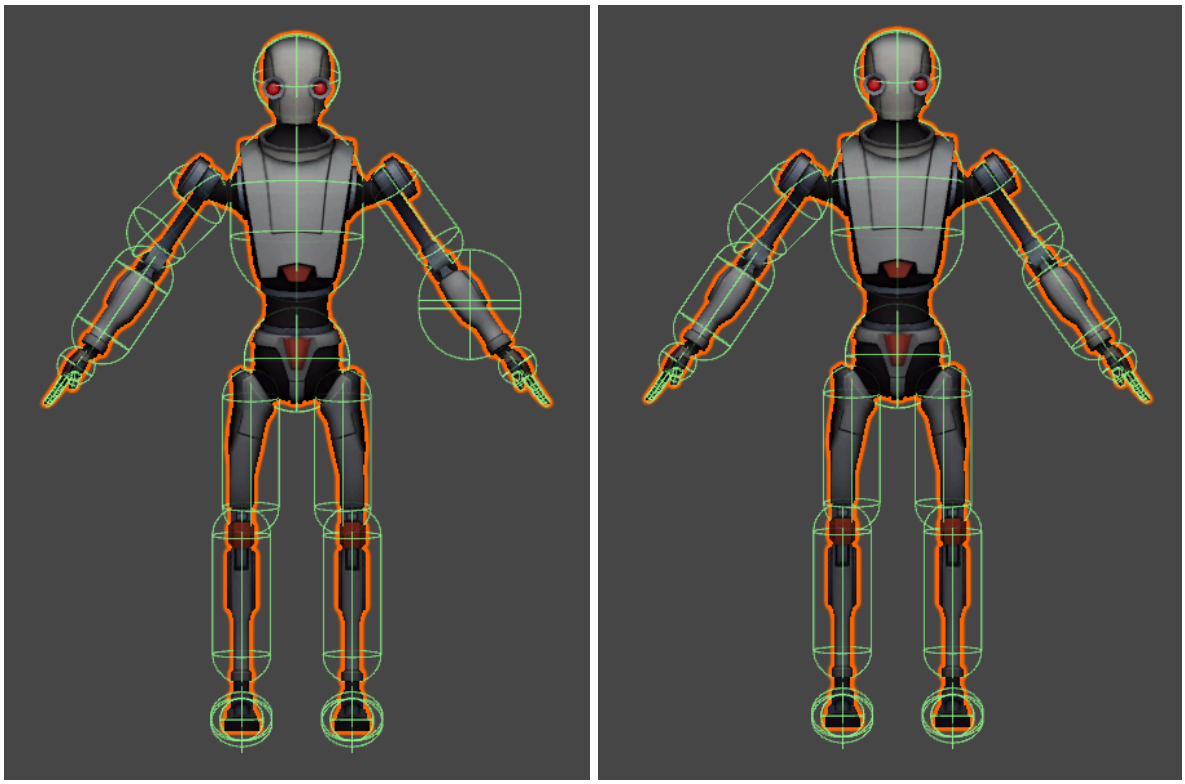
Auto Generated Skinned Mesh Colliders - Examples

These examples were created before previews were enabled in the auto skinned section of Easy Collider Editor. The information is still applicable, but it's much easier now to get a good result when you can see the result as you adjust parameters. Keep in mind for areas where the auto generation does not work, you can remove colliders and recreate them manually.

Example #1: Capsule Colliders on Robot Skinned Mesh

In this example we are going to automatically generate colliders on Space Robot Kyle. This is a good example as it will demonstrate why the allow realign toggle is important.

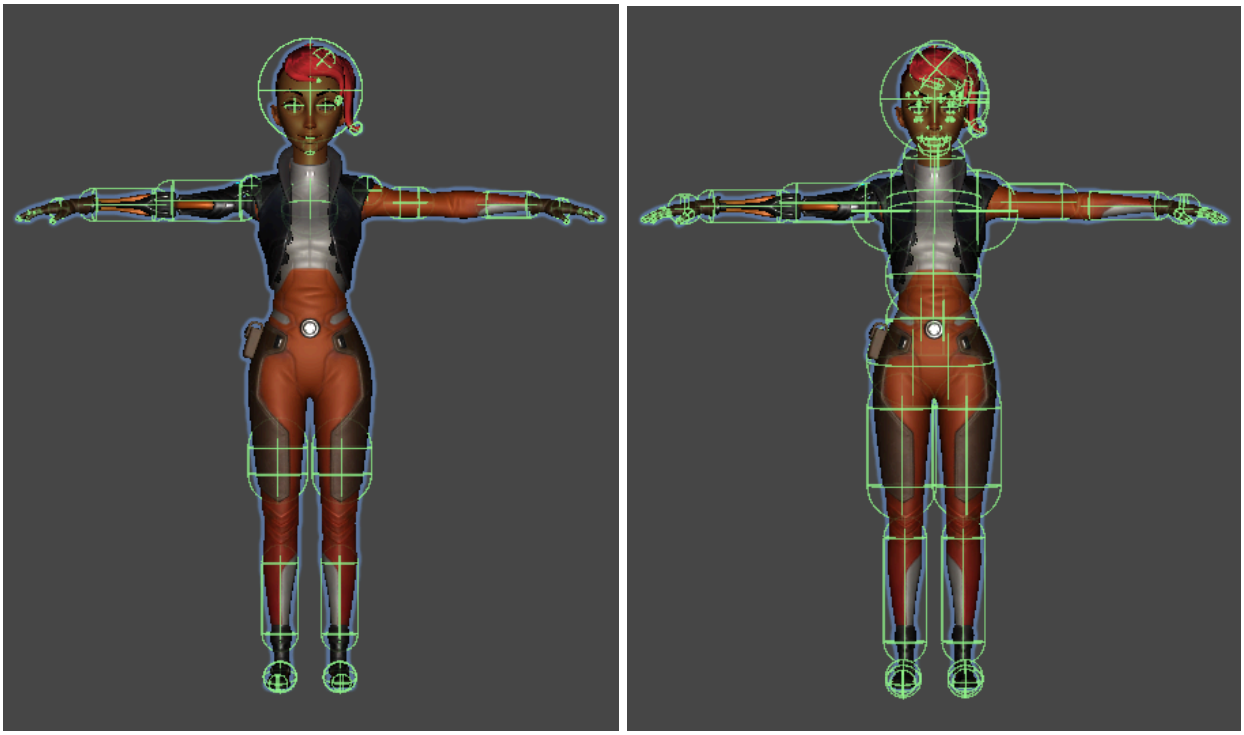
As you can see in the image on the left below, without allowing realignment, the capsule collider on the arm is completely misaligned with the actual mesh. This is because bones were not properly aligned during the mesh creation, skinning, and animation process. In the image on the right however, we have enabled realigning with a minimum angle of 15. As you can see the collider on the arm is significantly improved.



Example #2: Capsule Colliders on Human Skinned Mesh

In this example, we're going to use another free Unity Technologies asset to demonstrate the importance of the lists, and minimum bone weight.

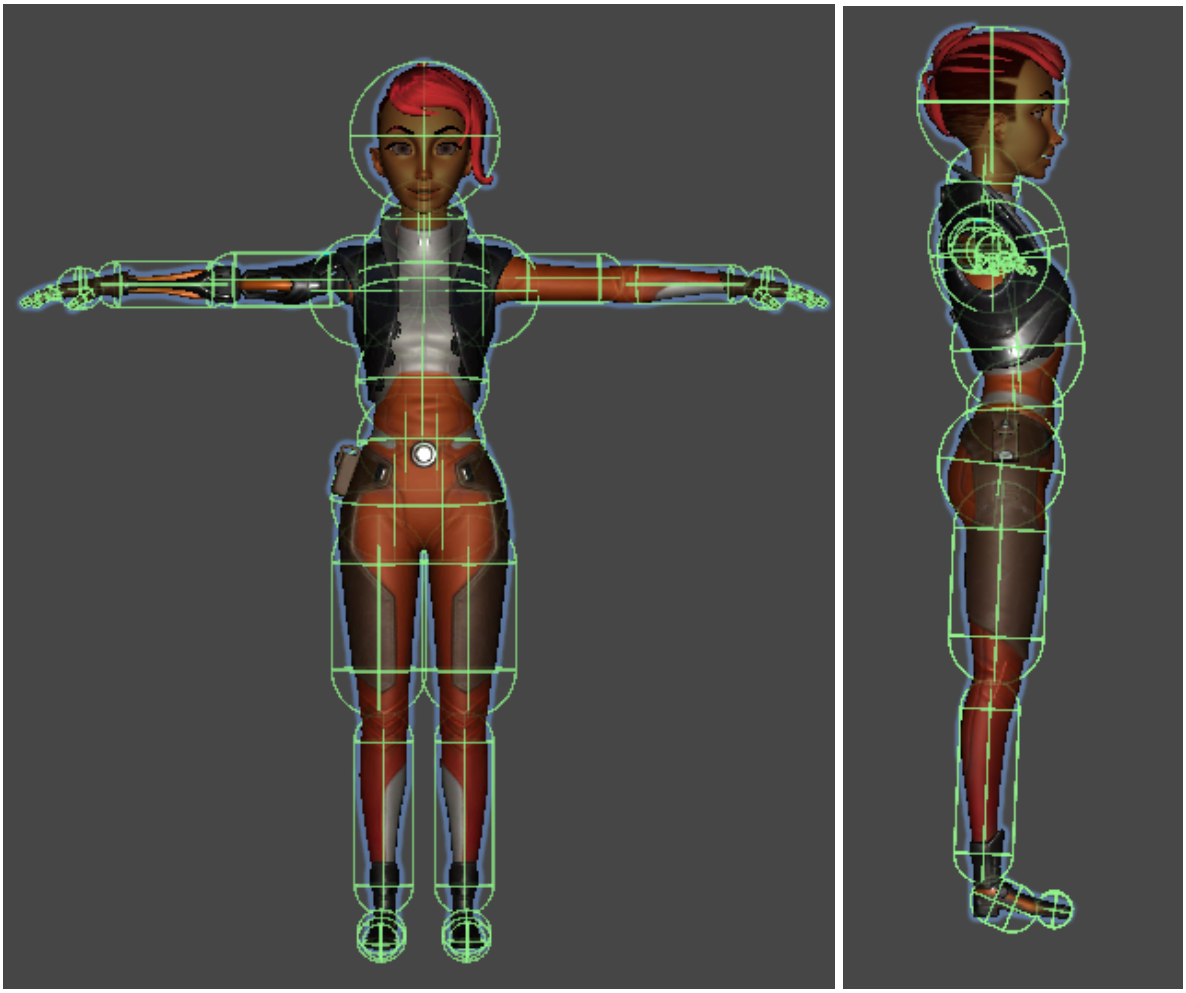
In the first image minimum bone weight was left at the value of 1, and capsules were generated. In this skinned mesh, most of the vertices are shared between bones. With a minimum bone weight of 1, only vertices that are controlled by a single bone were included in the calculation. The result of this is a very poor coverage of the colliders.



In the image on the right, the minimum bone weight value was changed to 0.5. This resulted in significantly better coverage of the mesh by the colliders.

There is still one issue, a lot of extra colliders are being generated on the head. This skinned mesh has lots of additional bones for the face and hair.

To fix this, we deselect any bones after the head from within the per bone settings dropdown. This will prevent colliders from being generated on those bones. To make this quicker, you can hold CTRL while you click the head toggle, and all the children will be untoggled as well. Then toggle the head back on and you are done.



The result of ignoring all the bones after the head is seen above. As you can see all of the extra colliders are removed. The result is a great coverage of automatically generated colliders.

Runtime Collider Creation

General

Note that you will need your own class that handles the selection and passing of the vertices.

The documentation of this section will continue to improve over several updates. Proper testing is still ongoing so I do not currently advertise the runtime usage of this asset. If you encounter any errors in runtime usage, be sure to let me know with as much detail as possible.

Runtime collider creation can be done by using the **EasyColliderCreator** class in the **ECE namespace**. This class exposes methods that can be provided with a list of worldspace, or local

space vertices. Intellisense should give you the documentation for each property and method as you use them.

The CalculateXCollider methods return an instance of EasyColliderData specific to the type of collider. These methods also have the CalculateXColliderLocal methods that only require local vertices to be passed. I recommend the use of these methods over the CreateX methods, as they allow more flexibility and don't require an additional properties class.

The CreateXCollider methods require world space vertices and a EasyColliderProperties object. They will create a collider of the specific type, using the specified method, add it to the properties' AttachTo object, set the specified collider properties, and return the collider. These are primarily for internal editor use, but I have modified them to work during runtime as well.

VHACD

Currently, runtime VHACD is **not** supported. I hope to add this functionality at some point in the future. If it is something you are interested in, please let me know!

Important Note on Prefab Isolation Mode

When using this asset with prefab isolation mode, be sure to enter prefab isolation mode before setting the Selected GameObject field. Once you are done editing the prefab, be sure to click the Finish Currently Selected GameObject button before exiting prefab isolation mode, otherwise components will be left on the prefab.

The correct process is as follows: Enter prefab isolation mode for the prefab you wish to edit, drag the gameobject from the hierarchy into the Selected GameObject field, create colliders, click Finish Currently Selected GameObject button, exit prefab isolation mode.

Other

FAQ

Below is a list of common questions or errors that can occur when using this asset. If you have any further questions please contact me.

1. **Vertex selection isn't working / I cant see the vertices I have selected.**
    - Make sure the Selected GameObject field is set to the gameobject you want to work with.
    - Try changing the Render Vertex Method in preferences from SHADER to GIZMOS

- If you are using GIZMOS, make sure gizmos are enabled in the scene view.
- Check the console for any warnings from this asset.
- If you are selecting a parent of a mesh, make sure Child Meshes is enabled
- If the child mesh is a skinned mesh, make sure Include Child Skinned Meshes is enabled in preferences.
- Try closing the Easy Collider Editor window and reopening, or clicking finish and reselecting the gameobject.
- Make sure that any parents of the mesh you have selected do not have a kinematic rigidbody, or select that parent and enable the child meshes toggle.
- If you have multiple scene view windows open side by side, try closing one of the windows.

2. **Using gizmos causes slowdown with lots of vertices selected, why can't I use the shader?**

    - The shader used to display vertices uses a compute buffer. This is only supported on some systems. More information about platforms where compute shaders work can be found here: https://docs.unity3d.com/Manual/class-ComputeShader.html

3. **The rotated colliders aren't being rotated correctly.**

    - When creating the rotated box colliders, the first 3 points selected define the rotation of the box collider. For a capsule collider the first 2 points define the rotation of the collider. These points are very critical when creating rotated colliders. I suggest selecting different vertices with the preview enabled, and set to rotated box colliders, until you better understand how the the orientation works

4. **Prefab isolation mode left components on the gameobject.**

    - To make sure components are not left on the gameobject, be sure to click the Finish Currently Selected GameObject button before leaving prefab isolation mode.

5. **Unity is giving me a warning that it could not create a convex hull because it has >256 polygons and is instead using the partial hull?**

    - In newer versions of unity, the inflate mesh option on mesh colliders has been removed. Since this removal, if the convex hull created from a mesh has >256 vertices physX will display a warning about having too many polygons.
    - Everything should still work as expected, but instead of this warning being hidden by unity, it is now displayed in the console.

- To fix this warning, consider making each convex hull simpler by selecting as few vertices as possible during collider creation.
- Additionally, multiple mesh colliders with less than 256 polygons can be created on the same object to solve this error.
- The example in this documentation of creating convex mesh colliders on a static anvil mesh may be helpful.

- If using auto skinned mesh colliders, be sure to enable the force<256 triangles toggle after changing the collider type to Convex_Mesh

6. **Rotated colliders don't work correctly after I scale my object / The preview collider is different from the actual result.**

    - This occurs due to scaling issues. Certain components do not fully support non-uniform scaling, and some don't support non uniform scaling combined with rotation. Unfortunately these components include sphere, capsule, and box colliders.
    - The preview uses the same data that the created collider uses, but unfortunately there are instances where the actual collider is different due to non uniform scaling especially when combined with parent/child rotations.
    - For more information about the limitations of non uniform scaling, see the section Limitations with Non-Uniform Scaling at: https://docs.unity3d.com/Manual/Transforms.html

7. **I'm getting errors about a mesh requiring read/write access?**

    This occurs when a mesh has the read/write option disabled in its import settings, and the mesh has a negative scaling with a mesh collider attached. To fix this issue enable read/write access in import settings or use a mesh with uniform non-negative scaling when generating colliders.

8. **I'm getting a DLL not found error for VHACD.**

    This likely occurs because of missing dependencies that are usually installed on development computers (generally these are installed when visual studio is installed along with unity).

    Installing the latest vc redistributable from microsoft should solve most of these issues: https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads.

Installing the windows 10 sdk may solve other issues: https://developer.microsoft.com/en-ca/windows/downloads/windows-10-sdk/

If you are still having issues a dependency walker can be run on VHACD.dll to identify which dependencies you may be missing. A third party tool that does this is located here: https://github.com/lucasg/Dependencies

9. **When updating to the newest version I get an error about ECE_VHACD.dll.**

   This error occurs because of the way unity loads dlls. If the update notes do not specifically mention that ECE_VHACD.dll has been modified, this is not an issue.

   If you notice an update is required that does mention ECE_VHACD.dll being modified, the best method for updating is to close Easy Collider Editor's window if it is open, save your project, close unity, re-open unity, and update the asset immediately upon entering the project again.

10. **VHACD is generating convex mesh colliders that don't fit how I would like them to fit / there are holes / other errors.**

    Unfortunately, there's not much I can do to help with particular issues relating to the resulting calculations from vhacd. Be sure to look at the VHACD section of this documentation and see which parameters may need to be adjusted, and other tips for problem meshes. Alternatively, vertex selection can be used to create the mesh colliders themselves for meshes where VHACD does not produce a satisfactory result.

11. **After running VHACD I'm getting errors about the source mesh and smooth surface regions?**

    Unfortunately, there's not anything specific I can do to fix this particular issue either. Generally this only happens rarely, and usually only when the Project Hull Vertices option is enabled. There are two options that can fix this issue. You can either disable this option and use a higher resolution to maintain the precision, or leave it enabled and change the other VHACD settings (max convex hulls, resolution) until convex hulls are generated without this issue.

12. **The VHACD results are vastly different from the preview!**

In this case, if the preview looks good, but the result looks malformed, it's likely a result of a very small scale on the underlying mesh data. Be sure to keep an eye on things like a parent object having a scale of 1, but the children with the mesh have a scale of 100, which can happen quite often in meshes exported from some modeling programs. In this case the actual mesh data is 100x smaller than appears. Since the preview is correct, you can identify that the actual

generated meshes are correct as well by clicking a generated collider's mesh field and looking at it in the preview in the inspector. In this case, it's an issue with Unity/Physx being unable to convert a very small mesh into a correctly scaled mesh collider. I'd highly recommend rescaling your objects to have a uniform scale of 1 in your 3d-modeling software of choice, as this can cause issues elsewhere as well.

Change Log

6.15.0:

- Add additional scaling options for vertices in preferences
- Mouse cursor now changes visually when using the add / remove from selection shortcuts
- Add API for opening an EasyColliderEditorWindow window from script and setting the selected gameobject.
- Add a collider post processing interface as well as a default implementation for people to use if they need to do anything custom after creating the different kinds of colliders. This can be done using the EasyColliderPostProcessor.cs script.
- Enabled collider post processing to be used at runtime. You can supply your own implementation at runtime using EasyColliderCreator.EasyColliderPostProcessor
- Fixed issue where remove/merge wouldn't allow selection of colliders on child objects that were created with easy collider editor

6.14.0:

- Add preference options for specifying a subfolder to save generated convex mesh colliders to allow users to more easily organize their collision meshes
- Fix issue where having another "Assets" in the asset path would cause issues saving a convex mesh collider

6.13.1:

- Add option to toggle read/write enabled on saved meshes used for convex mesh colliders
- Add a workaround to an input issue that only occurs in one version of unity.

6.13.0:

- Add support for arm/silicon mac.
- Rename and move OSX VHACD plugin to separate folder.
- Mac should no longer have to use the render gizmos method, as the shader should now correctly work with metal, which will significantly improve performance when lots of vertices are selected

- Add a button to Edit/Remove/Merge tab to only select the colliders vertices instead of removing and selecting as the edit button does.
- Fix issue where trigger colliders were not selectable in projects where query trigger interaction in physics settings would not collide with raycasts
- Fix issue where inverting selected colliders would not respect include child meshes, and would select the mesh collider used for vertex selection.
- Fix issue where clicking in empty space would select a null collider

6.11.1:

- Add documentation for saveable and loadable VHACD settings. Below VHACD Advanced settings table and in the UI Documentation.
- Fixed issue where objects with invalid characters in their name would cause an error.

6.11.0:

- Add saveable/loadable VHACD settings
- Add several different methods when picking where to save convex hulls in preferences
- Updated documentation with some convex mesh collider considerations.

6.10:

- Add additional shortcuts for all vertex selection tools
- Adjust drawing order of merge preview and colliders section for visibility
- Change default vertex selection to use mouse selection
- Update for ECE_VHACD.dll to fix an issue where an instance is not correctly cleaned/deleted.

Bug reports

If you experience any issues or bugs while using Easy Collider Editor, please contact me at pmurph.software@gmail.com with as much information as possible. Please include the version of unity you are using (ie 2019.3.7f1). It would also be helpful to describe what you were trying to do, what you expected to happen, and what actually happened. Images or videos are also very helpful if you are able to provide them.

Want more features? Or have ideas for other improvements? Let me know!

Over the development of this asset, I have added many features and improvements based on the requests from users. Without these requests, I don't know the types of new features or general improvements people wish to see. I love hearing what users would like to have added to this asset, as well as any improvements or clarifications that can be made to this documentation. Of

course I can’t guarantee all requests will be added to the asset, but if you have any features or improvements you would like to see, please contact me at pmurph.software@gmail.com