Easy Collider Editor - 6.3

Table of Contents

Quick Start Guide

Collider Creation

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

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. Press V to select vertices on the mesh, and B to select any point on the mesh.
    a. Enable mouse selection (in preferences) and left click to select vertices, and right click to select points.
    b. Hold A or CTRL before clicking to only add vertices.
    c. Hold S or ALT before clicking 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 to select vertices as well.
    a. Hold CTRL while clicking the grow or grow last buttons to grow until no more vertices can be selected with that button.
8. Click an icon representing the kind of collider you want to create.
    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 a collider from the current preview.
    c. Or double tap 1-7 to create each type of collider.
9. When done creating colliders, click the Finish Currently Selected GameObject button.

VHACD

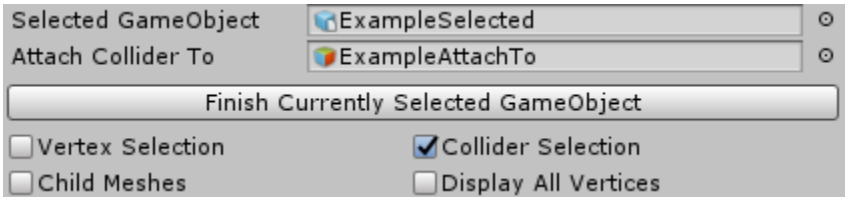
1. Open the editor window from Window > Easy Collider Editor
2. Drag a GameObject from the scene into the Selected GameObject field.
3. Click the VHACD button.
4. A preview will be drawn that shows the approximate result at the current VHACD settings.
5. Adjust VHACD settings until the preview shows 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.

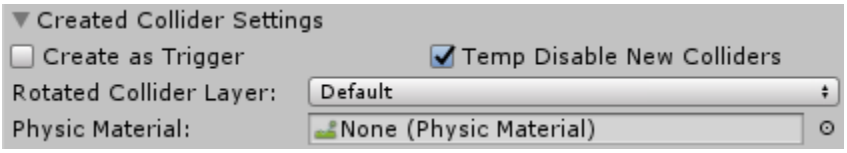
When using the asset, you will notice that some of the buttons appear to be disabled. Hovering over the disabled buttons will give you a tooltip on why the button is currently disabled. Buttons that require certain things to work are disabled and displayed differently when those conditions are not met, 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.
- **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. **It is particularly important to use this before closing unity, or before entering or exiting prefab isolation mode.** If this button is not used, components may be left on objects instead of being removed.
- **Vertex Selection**: Enables selection of points and vertices in the scene view.
- **Collider Selection:** Enables collider selection in the scene view
- **Child Meshes:** When enabled, points and vertices on meshes from the Selected GameObject's children are also able to be selected
- **Display All Vertices:** Displays all selectable vertices. Generally used to make sure meshes are being detected properly, or that the shader or gizmos are functioning correctly.

Created Collider Settings

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



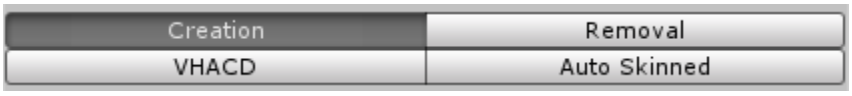
.

- **Create as Trigger:** When enabled, creates the collider with the IsTrigger property checked.

- **Temp Disable New Colliders:** When enabled, temporarily disables colliders created with this asset. This makes vertex selection easier when creating multiple colliders on the same object. Colliders are automatically enabled when the Finish Currently Selected GameObject button is used.
- **Rotated Collider Layer:** The layer to set on the gameobjects created for rotated colliders. This option only displays if the Rotated Collider on Selected GameObject Layer toggle in preferences is disabled.
- **Physic Material:** Any created collider's physic material property gets set to this value when it is created.

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:** Displays the Vertex Selection Tools, and the Collider Creation Tools. Going into this section automatically enables vertex selection and disables collider selection.
- **Removal:** Displays the Collider Removal Tools. Going into this section automatically enables collider selection and disables vertex selection.
- **VHACD:** Displays the UI used for generating convex hulls using VHACD.
- **Auto Skinned:** Displays the UI used for automatically generating colliders along the bones of a skinned mesh.

Vertex Selection Tools

This section of the UI contains several tools that make it easier to quickly select vertices. These tools will appear in both the creation tab, and the VHACD tab.

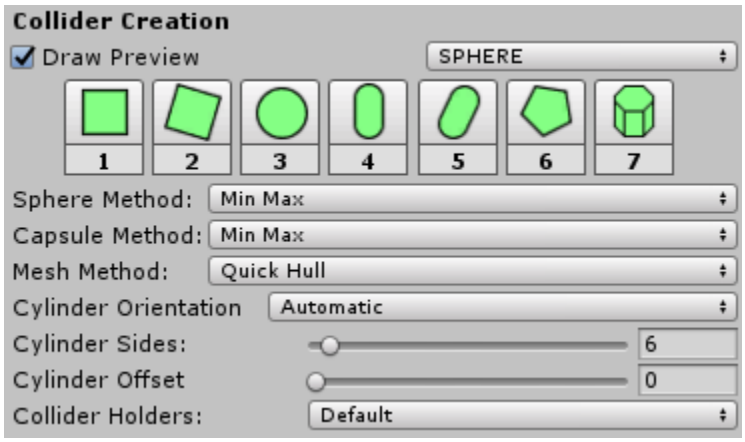
- **Snaps:** These buttons let you change between the vertex snapping to only add, only remove, or add and remove vertices. These buttons are automatically switched between

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

- **Clear:** Deselects all currently selected vertices.
- **Grow:** Expands the selected vertices along triangle edges of all selected vertices. If you are holding CTRL while clicking this button, it grows the selection until no more vertices are added.
- **Grow Last:** Expands the selected vertices along triangle edges of only the last selected vertices. If you are holding CTRL while clicking this button, it grows the selection from the last selected vertices until no more vertices are added.
- **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.

Collider Creation Tab

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



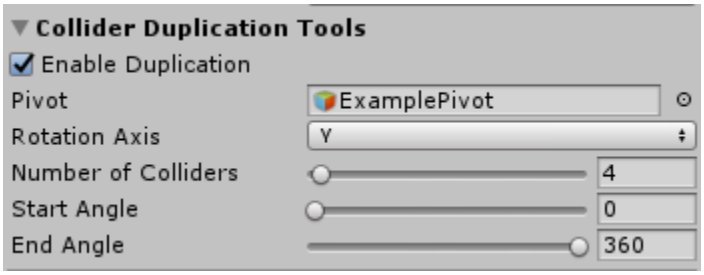
- **Draw Preview:** When enabled, draws a preview of the collider type set in the box to the right.
- **Icons:** In order from left to right.
    - **Create Box Collider:** Creates a box collider from the selected points.
    - **Create Rotated Box Collider:** Creates a gameobject that is aligned with the first 3 points selected, attaches it to the Attach Collider To gameobject, and then creates a box collider on the created object.

    - **Create Sphere Collider:** Creates a sphere collider using the algorithm specified by the Sphere Method field.
    - **Create Capsule Collider:** Creates a capsule collider using the algorithm specified by the Capsule Method field.
    - **Create Rotated Capsule Collider:** Creates a gameobject that is aligned with the first 3 points selected, attaches it to the Attach Collider To gameobject, and then creates a capsule collider on the created object.
    - **Create Convex Mesh Collider:** Creates mesh from all the points that are selected, saves it in the location specified in preferences. This mesh is then used to generate a convex mesh collider, and is attached to the Attach Collider To gameobject.
    - **Create Cylinder Collider:** Creates a convex mesh collider similar to the create convex mesh collider button, except creates the mesh in the shape of a cylinder using the number of sides specified below.
- **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.
- **Cylinder Orientation:** The axis on which to orient the cylinder created. Automatic aligns the cylinder's height with the largest axis. The Local X, Y, and Z options align the cylinder with the Attach To object's local X, Y, or Z axis respectively
- **Cylinder Sides:** The number of sides on a created cylinder collider..
- **Cylinder Offset:** The amount in degrees to offset the cylinder around it's height axis. Useful for when a cylinder-shaped object was rotated during mesh creation.
- **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.

Collider 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 similar to the vertex selection buttons.

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

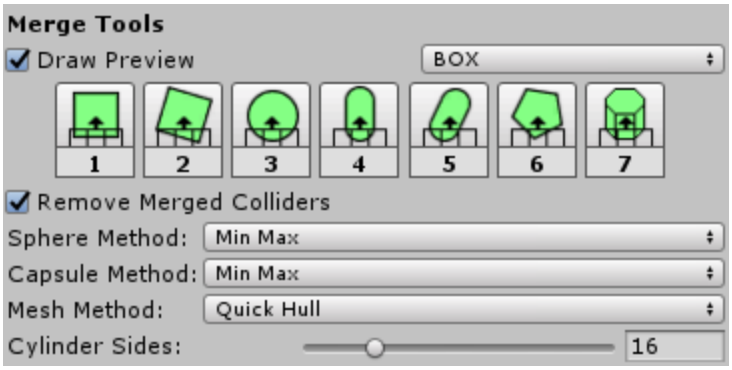
Collider Removal

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

- **Remove Selected Collider(s):** Removes the currently selected colliders.
- **Remove all Colliders:** Removes all colliders on the Selected GameObject, the Attach To gameobject, and their children.

Collider Merging

This section contains the options for merging colliders.

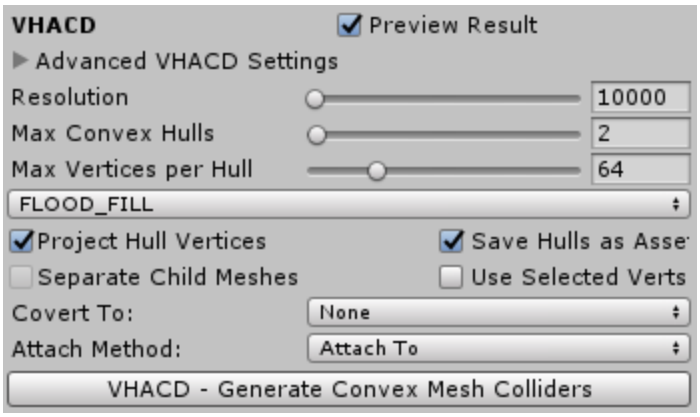


Most of the UI in this section functions exactly the same as the tools specified in the Collider Creation Tools section above. Except instead of creating colliders from vertices, the colliders selected are merged into the collider specified.

- **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.

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. If this is disabled, meshes won’t be saved. When enabled, you can create convex

meshes on prefabs or with VHACD and use the colliders across multiple scenes and objects.

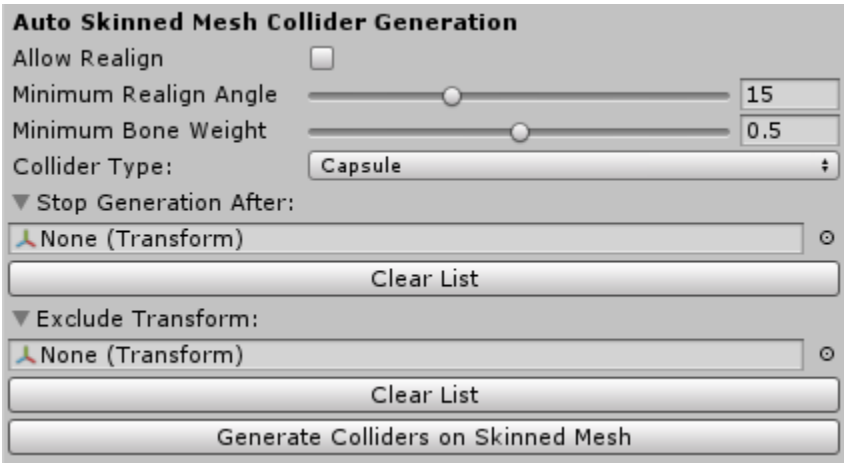
- **Separate Child Meshes:** When enabled, each child mesh will be processed separately using and have the resulting convex hulls attached to the child's transform. Allows you to generate convex hulls on multiple objects with a shared parent using the same parameters.
- **Use Selected Verts:** This option, when combined with vertex selection, 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. This can be useful to automatically generate primitive colliders instead of mesh colliders. Unfortunately, the control over VHACDs output is limited to the parameters used, so precise results are unlikely.
- **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.
- **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.



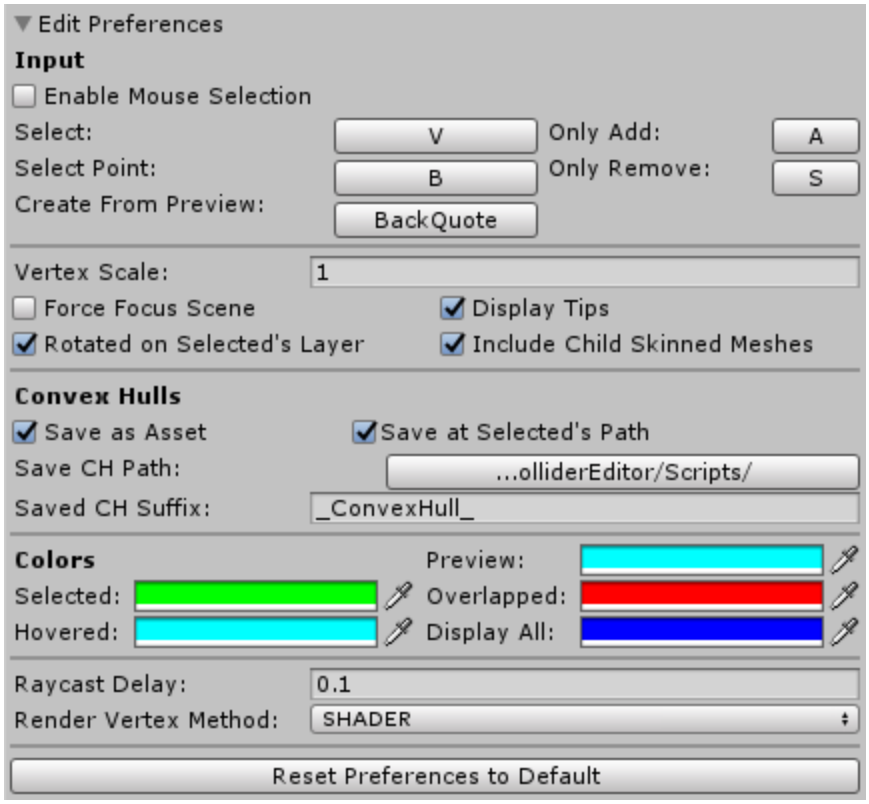
- **Allow Realign:** Allows a child transform to be created to hold colliders when a bone with one child is not properly aligned.
- **Minimum Realign Angle:** When allow realign is enabled, this option sets the angle at which bones will be realigned. If the minimum angle between all the bones axis' and 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.
- **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.
- **Stop Generation After:** Transforms in this list get a collider generated for them, but transforms of bones after are not included.
- **Exclude Transform:** Transforms in this list are specifically excluded from having a collider generated on them.

Finish Currently Selected GameObject

Finishes editing on the Selected GameObject. This ensures proper cleaning up of added components required for this asset. **It is particularly important to use this before closing unity, or before entering or exiting prefab isolation mode.** If this button is not used, components may be left on objects instead of being removed.

Preferences UI

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



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

- **Enable Mouse 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.
- **Select:** Key to press to select/deselect a vertex or collider when vertex or collider selection is enabled.
- **Select Point:** Key to press to select a non-vertex point on the mesh.
- **Only Add:** When this key is held down vertices will only be selected. This works for both single vertex selection (it snaps only to vertices you can add), and drag selection. The CTRL key also provides the same functionality.
- **Only Remove:** When this key is held down, vertices will only be deselected. This works for both single vertex selection (it snaps to the closest vertex you can deselect), and drag selection. The ALT key also provides the same functionality.

- **Create From Preview:** 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 that way as well.

Common Options:

- **Vertex Scale:** This multiplier is applied to all types of vertex displays. If the vertices are being drawn too large or too small for your mesh, try adjusting this value.
- **Force Focus Scene:** This option is no longer necessary and will soon be removed. Smart focus has been added to automatically focus the scene after certain buttons and toggles are clicked. This option force focuses the scene view when either vertex, or collider selection is enabled. Allows selection without having to click into the scene view first. Can make it difficult to move, or change values in other inspectors.
- **Display Tips:** When enabled, displays helpful tips for common issues at the bottom of the window.
- **Rotated Collider on Selected GameObject Layer:** When enabled, rotated colliders automatically get created on the same layer as the Selected GameObject's layer.
- **Include Child Skinned Meshes:** Includes skinned meshes on children when Child Meshes is enabled.

Convex Hull Settings:

- **Save as Asset:** When enabled, meshes from creating convex mesh colliders or generating convex hulls using VHACD are saved as assets. If this is disabled, meshes won't be saved. This means that only that instance of the object in that scene will be able to have that mesh collider. When enabled, you can create convex meshes on prefabs or with VHACD and use the colliders across multiple scenes and objects.
- **Save at Selected's Path:** When creating a convex mesh collider or using VHACD, attempt to save the mesh at the same location in the project as the selected gameobject's path. This option overrides the Save CH Path below.
- **Save CH Path:** This folder is used to save the mesh collider's mesh if Save Convex Hull at Selected GameObject's Path is disabled, or trying to find the path of the Selected GameObject fails.
- **Saved CH Suffix:** This is the suffix added onto saved mesh assets for use as source meshes for convex mesh colliders.

Colors:

- **Preview:** This is the color used to draw creation preview colliders when Draw Preview is enabled.
- **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. If these vertices are selected, they will be deselected.
- **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.

Other options

- **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 will select this vertex. If you enable the mouse selection, left click can also be used.

This process works in the same way when collider selection is enabled. The Select Key selects the current highlighted collider, clicking the same collider again like in vertex selection deselects that collider. This is the only method to select colliders that is currently supported.

Point Selection

This method allows you to select not just vertices, but any point on the mesh. Sometimes it can be very useful to be able to select arbitrary points on the mesh so the colliders only go up to certain areas where vertices aren't located. Point selection is done using the Point Select Key. With mouse selection enabled in preferences, you can also use the right mouse button.

Vertex Snaps

Holding down the Only Add (or CTRL) or the Only Remove key (or ALT) can also help with vertex selection. When the Only Add key is held, selecting vertices will only select vertices that can still be added, so you don't have to worry about accidentally removing vertices you want selected. When the Only Remove key is held, selecting vertices will do the opposite and only deselect vertices. When Only Remove is being held both the vertex select and point select keys or mouse buttons will snap to both vertices and arbitrary points that are currently selected.

Drag Selection

With vertex selection enabled, clicking and dragging in the scene will start a box selection. This allows you to select, or deselect lots of vertices all at once. The colors shown over the vertices represent the same things they do in normal vertex selection.

Additionally, holding down certain keys after starting a drag select allows you to only select (Only Add key, or CTRL) or only deselect (Only Remove key, or ALT) vertices within the box.

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 appears which 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. You can quickly select whole sections of a complex object if it's not welded by selecting a single vertex and CTRL + clicking the grow button.

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. This can be helpful if there's only a few vertices you wish to not include in the collider generation.

Ring

This button appears once at least 2 vertices are selected. It attempts to follow triangle edges around an object, creating a ring of vertices. Once it reaches one of the vertices it has already selected, it ends.

Collider Creation

To create colliders select vertices or points on the mesh, and then click the appropriate button. 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.

Alternatively, you can enable the Draw Preview option and set the dropdown beside it to the type of colliders you will be creating. In addition to the dropdown on the UI, you can quickly change between previews by pressing the numbers 1-6 on your keyboard or number pad to change the preview to a different type. Double tapping the key will create a collider from that preview. The ` or ~ button on your keyboard will also create a collider that is the same type as the preview without the need for double tapping.

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

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

Boxes

- Select all the vertices in whichever order you wish for box colliders, and they will all be included within the box.

Capsules

- Best Fit Method: 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.
- 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.

Spheres

- For all sphere methods, select points in whatever order you wish.
- 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.

Rotated Colliders

- 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. If this is not possible, try to visualize a line that connects the first two points. This line should align with the collider you are creating. 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.

- With the preview option set to show a rotated box collider, it makes it easier to visualize the plane of rotation the collider will lay on when selecting the first 3 points. After trying it out a few times, you'll quickly be able to identify where to best pick the first 3 points when a rotated collider is needed.

- Rotated capsule colliders are aligned based on the first two points selected. The third point is included, but does not really matter. This is because as you rotate a capsule collider around it's height axis, the world-space area it covers does not change like a box collider.

- 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 a specific collider 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, by actually calculating a convex hull, 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).
- Points can be selected in any quantity greater than 4, and in any order that you wish. 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. This is why it is important to think about which vertices you want to include in the mesh collider.
- This method will create a new mesh and save it (if the option to save hulls is enabled in preferences) in the folder set in preferences. It will try to find the Selected GameObject's path if that option is enabled, and fall back to the specified save folder if it is not enabled or the path could not be found. If neither path exists, it will save the mesh in the same folder as the preferences file.
- 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 with VHACD you may need to use more convex mesh colliders to get the same accuracy as manually created colliders.

Cylinders

- Cylinder Colliders are a recent addition to Easy Collider Editor that allows you to more easily create a cylinder shaped collider.
- Cylinder Colliders only require 2 vertices to be selected.
- Keep in mind that a cylinder collider is really just a convex mesh collider, so it's better to keep the number of sides as low as possible.
- The number of sides of the cylinder can be set using the slider. This slider is limited to 3 and 64 to stay below the limit at which physx can generate errors in some versions of unity.

- 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 is particularly useful for cylinder-shaped objects that were rotated in this way during mesh creation in an external program.

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. This is helpful when the pivot point, or the pivots rotation does not line up with what you want to rotate around. You can create an empty gameobject to temporarily use, and position it how you would like. The preview will be updated as you move and rotate and adjust the pivot's position.
- If you would like to see more duplication options, please let me know!

Collider Creation - Examples

The examples in this section were created with an earlier version of Easy Collider Editor where the previewer was not yet added. With the preview option that is now included, it's even easier to create the colliders that you need. Additionally the UI in some of the following examples may be out of date, but the process of each is still valid.

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. If the collider created does not match the vertices like you expected, hopefully these examples will better explain why this is the case. The improved support for undo/redo allows you to generate a collider, undo and select more, or different points very quickly.

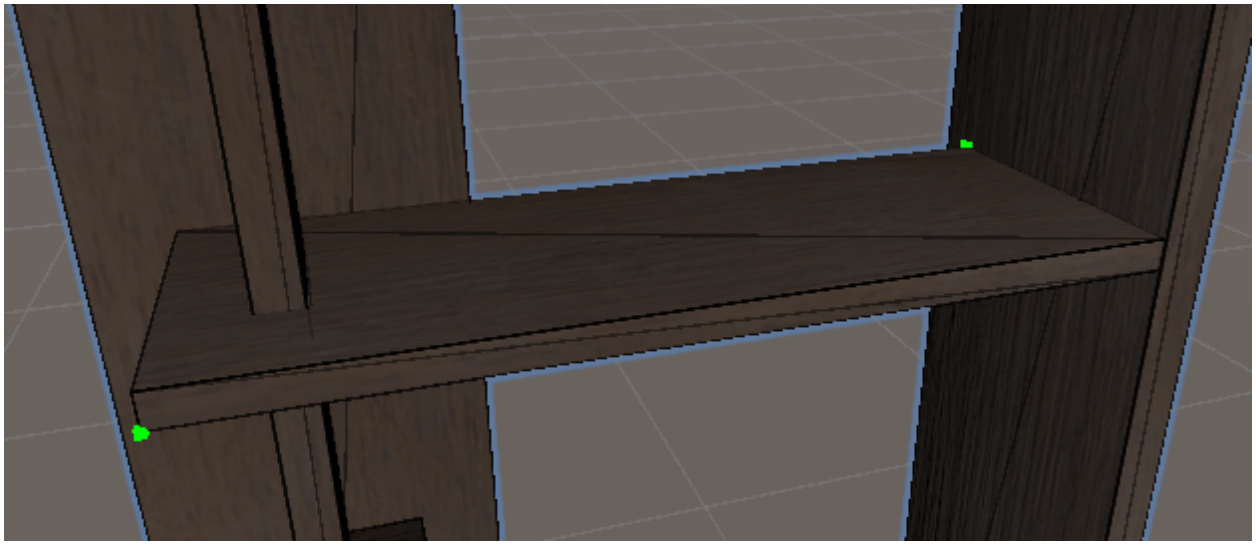
In the examples, I have set Unity's Editor shading mode to "Shaded Wireframe" to make it easier to see the mesh and it's vertices.

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.

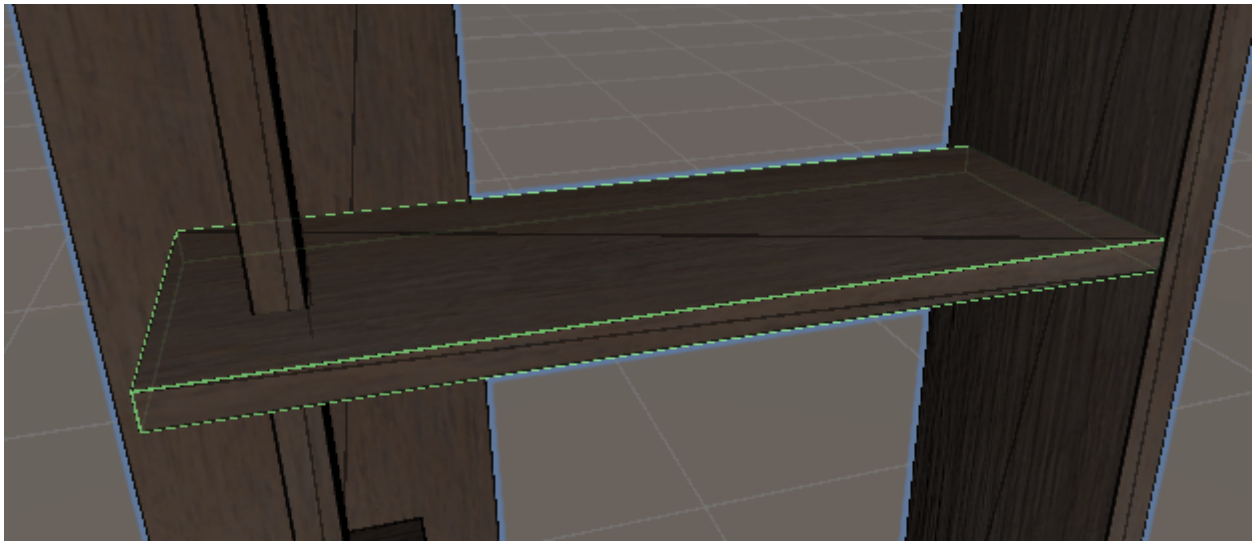
If you have any thoughts on how to make the following examples more clear please contact me at pmurph.software@gmail.com

Boxes

Example #1: Box on a Static Mesh



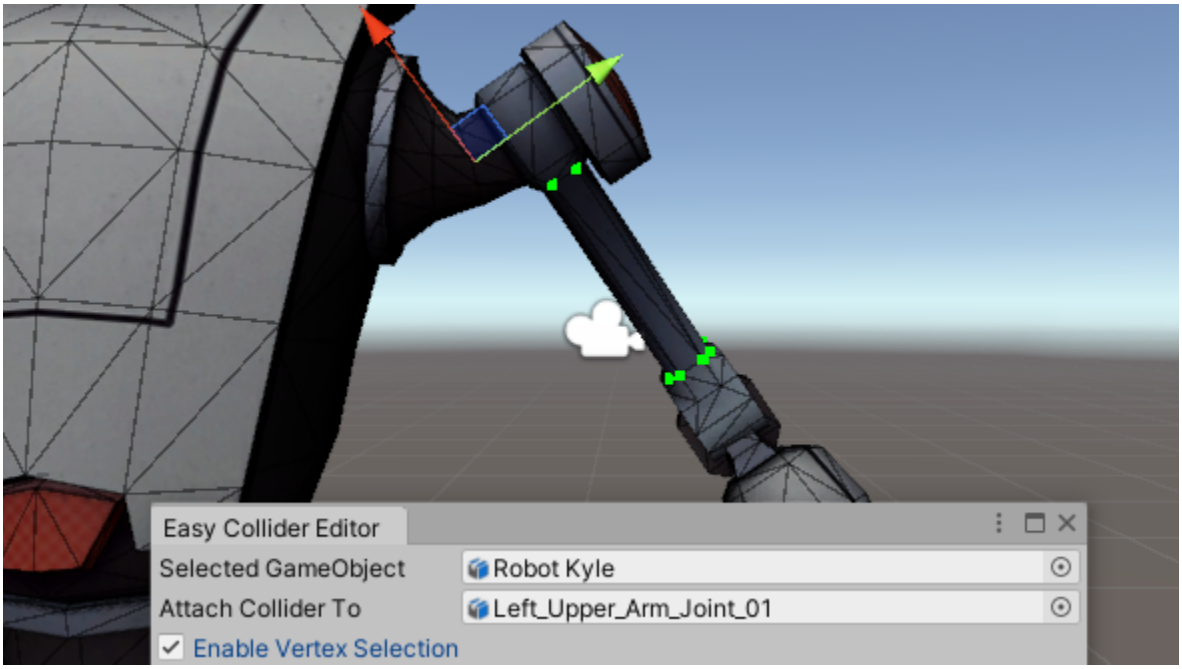
In this example we are creating a box on a simple static mesh of a shelf. We have only selected two vertices; the corners of the box we wish to create. Note that we could select as many points as we wish, and the box collider generated would contain all of these points.



After clicking the Create Box Collider button, a box collider is created that contains all of the points selected.

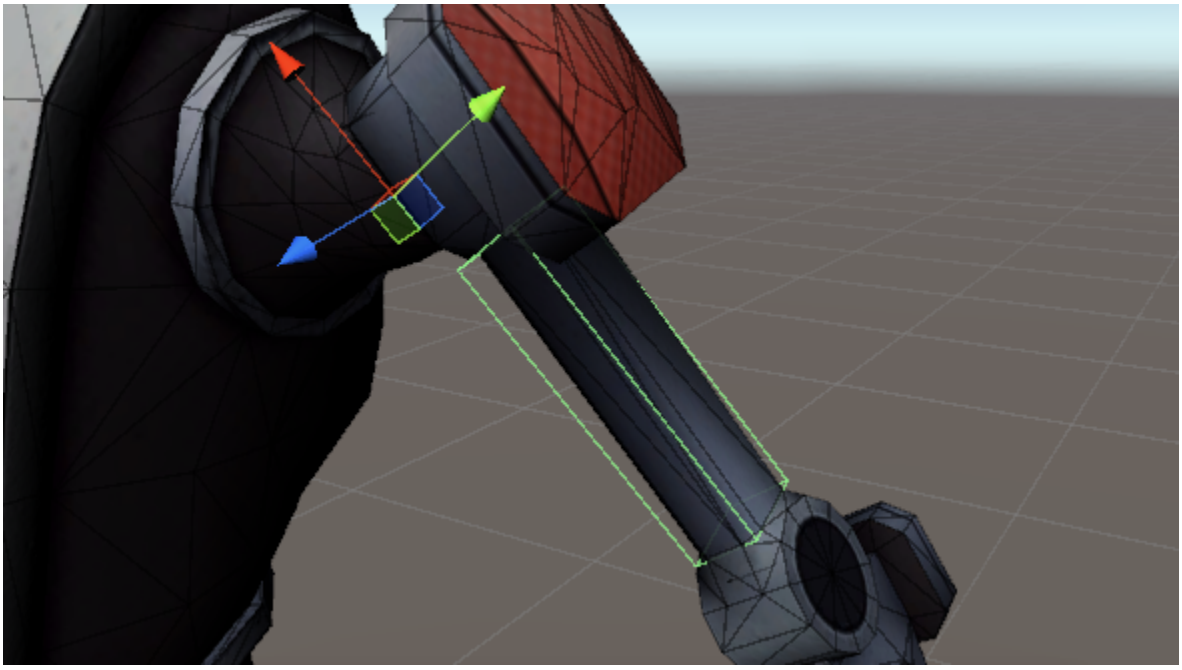
Example #2: Box on a Skinned Mesh

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.

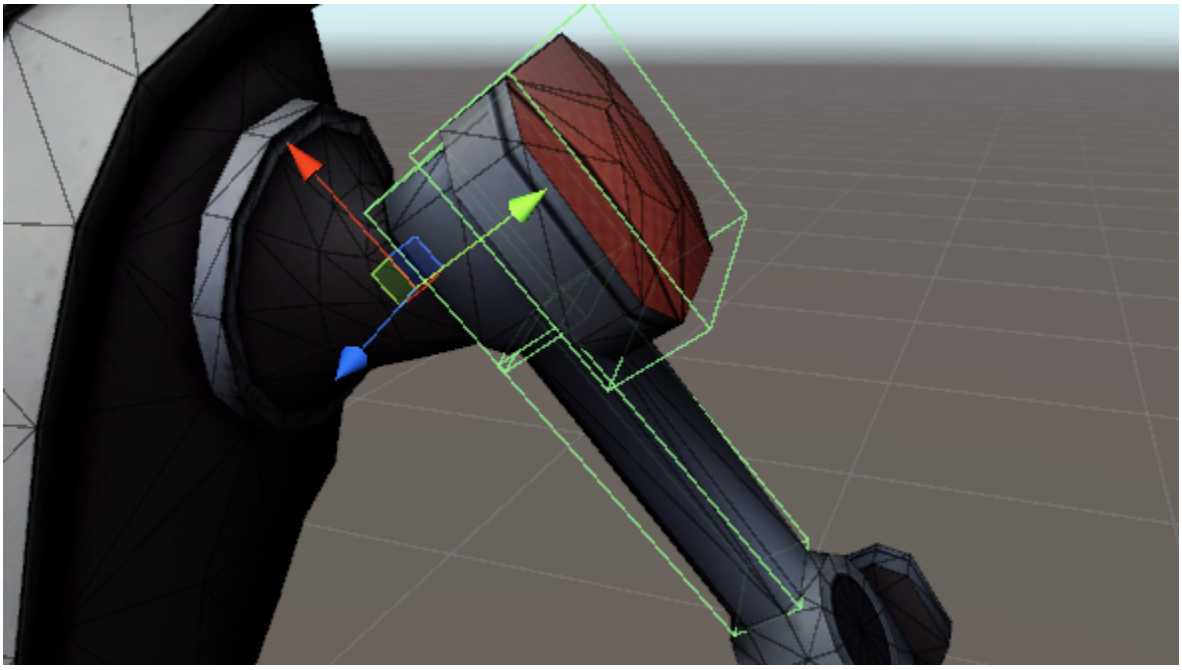


We have selected the root object of the whole mesh, Robot Kyle, while making sure that the Include Child Meshes toggle is checked. The Attach Collider To field was changed to the Left_Upper_Arm_Joint_01. Notice how the axis of this joint aligns with the boxes we wish to create. In properly created skinned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

In this image we have also selected some vertices on the skinned mesh to create a box collider from.



This box collider was created with the Create Box Collider button. Notice how it is aligned with the bone and encapsulates all the points we selected.



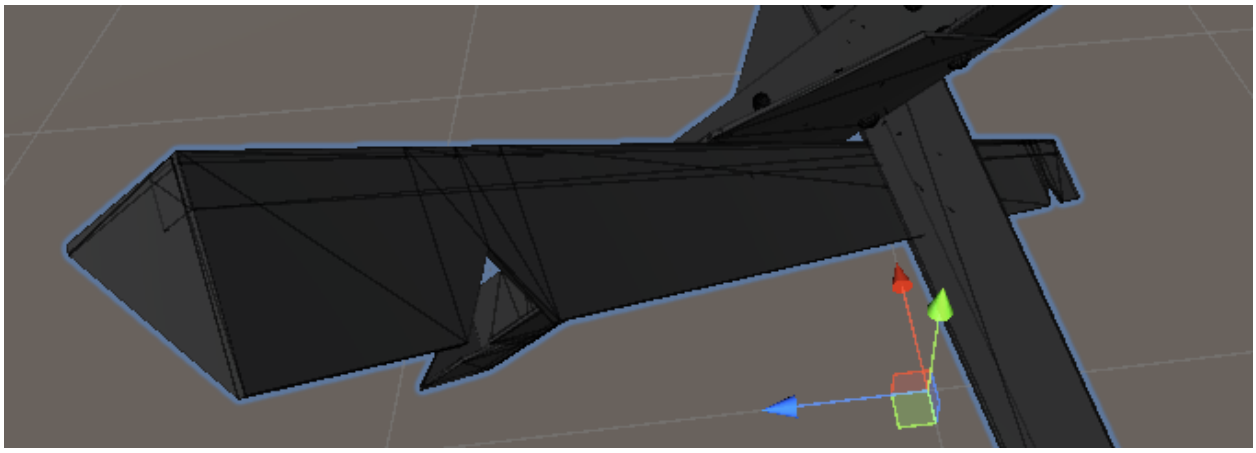
After creating 2 more box colliders, we have created very appropriate box colliders for the left upper arm of Robot Kyle.

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

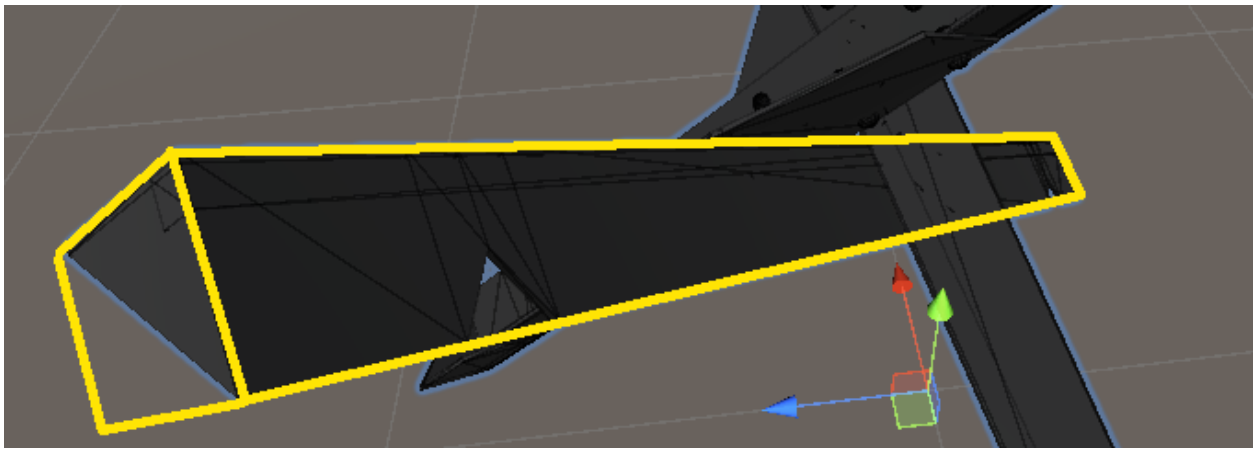
transform rotates with the arm during animations, which would rotate the colliders attached to it in the same way.

Rotated Boxes

Example #1: Rotated Box on a Static Mesh



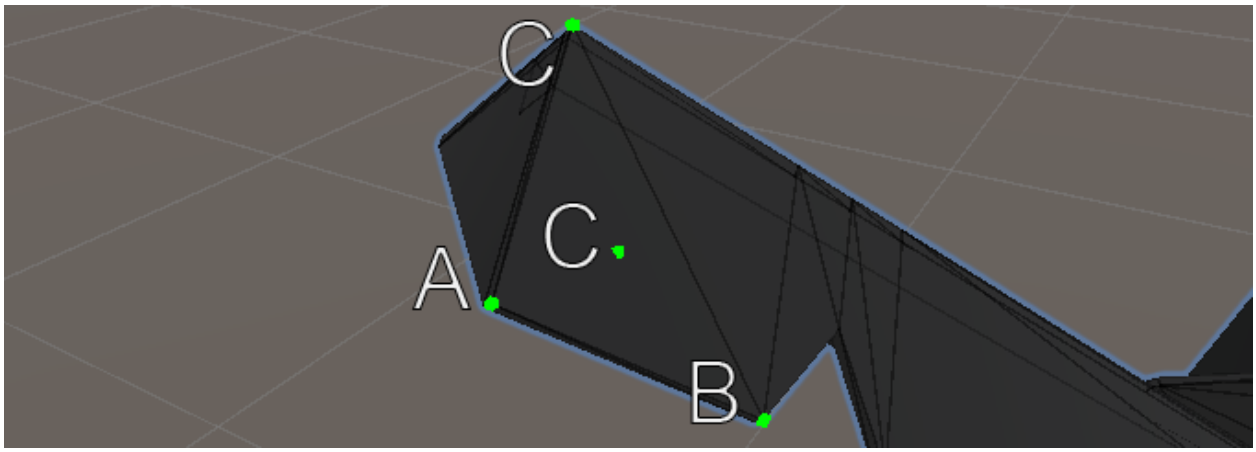
This is the object we are going to create a box collider on. Notice that it's pivot is aligned with the world space axis', but the geometry is not aligned with any particular axis. This makes it a good candidate for a rotated collider. In this case, I am looking to create a box collider that contains the whole angled section of the object.



In this image I have roughly outlined the rotated box collider I am looking to make. In this example I want to create a single box collider that contains the whole angled portion

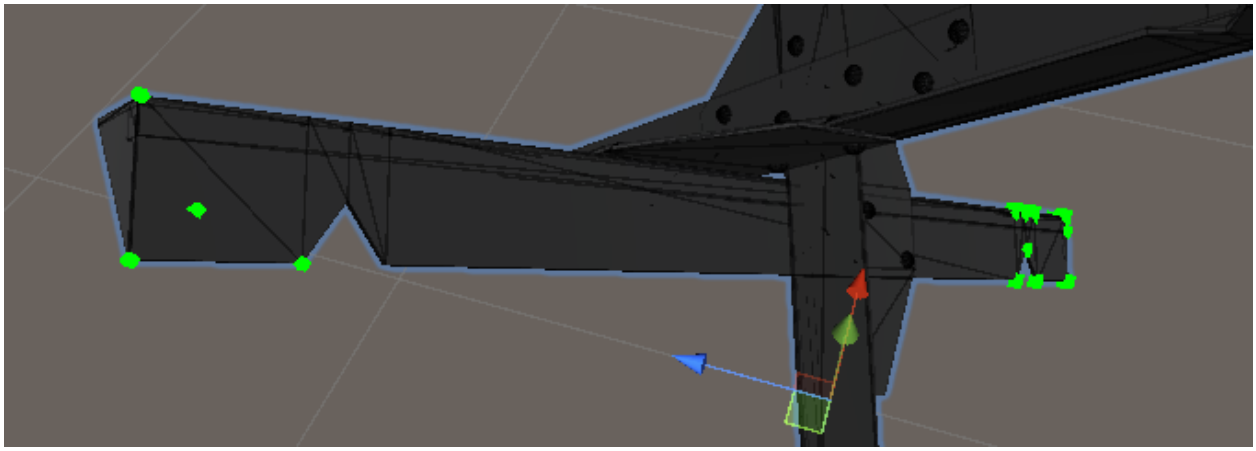
If I wanted, I could create several colliders, one for each side for more accuracy. In this case I would only need to create one rotated box collider. For the other colliders I could change the

Attach Collider To field to the first rotated box collider that was created, and create the other colliders as regular box colliders.

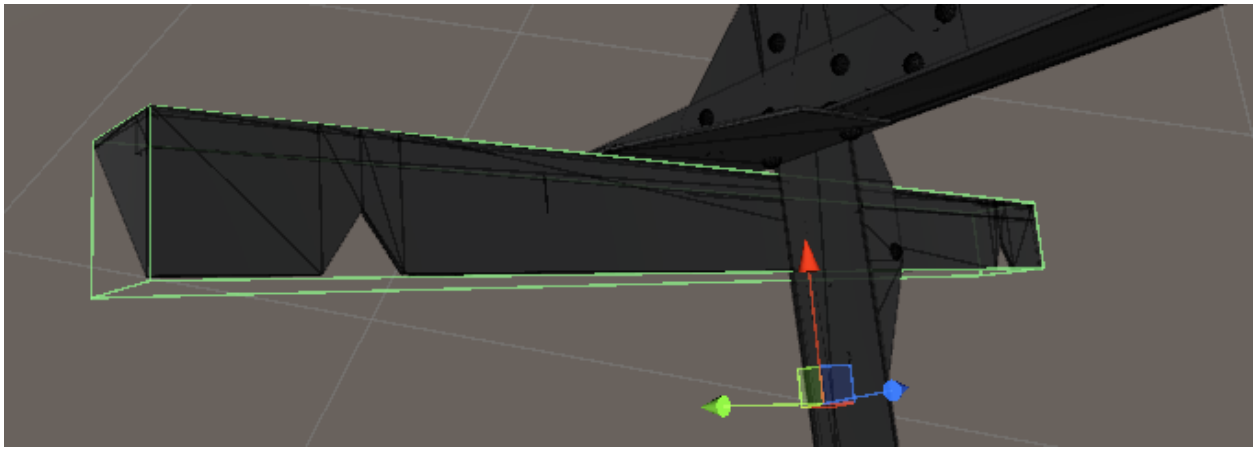


This shows the key points selected for a rotated box collider. A and B are the first two selected vertices. Notice how these two vertices align in the direction of the box collider we want to create. Note that A and B could be selected in either order.

Although there are two C's in the image above, either individual one would give the same result. Notice how when combined with A and B, either C would result in a face that would align with the resulting box collider.



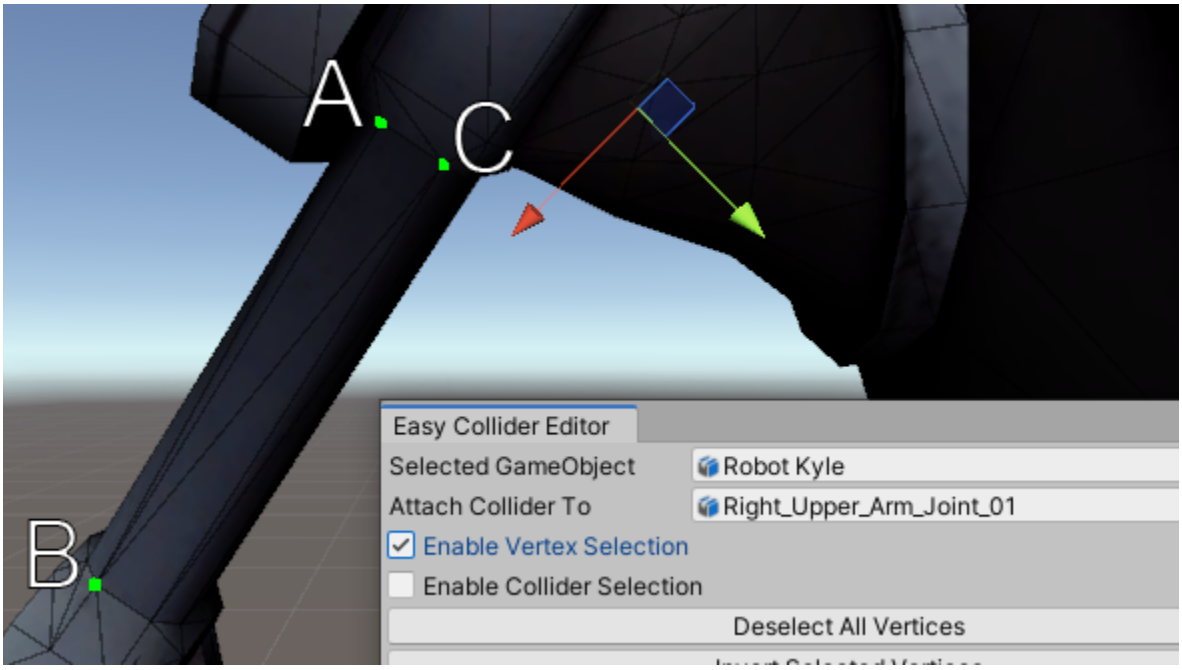
The vertices at the other end were selected very quickly by clicking and dragging the mouse to do a box selection. They are all inside the box that I want to create.



The resulting rotated box collider from the points selected above.

Example #2: Rotated Box on a Skinned Mesh

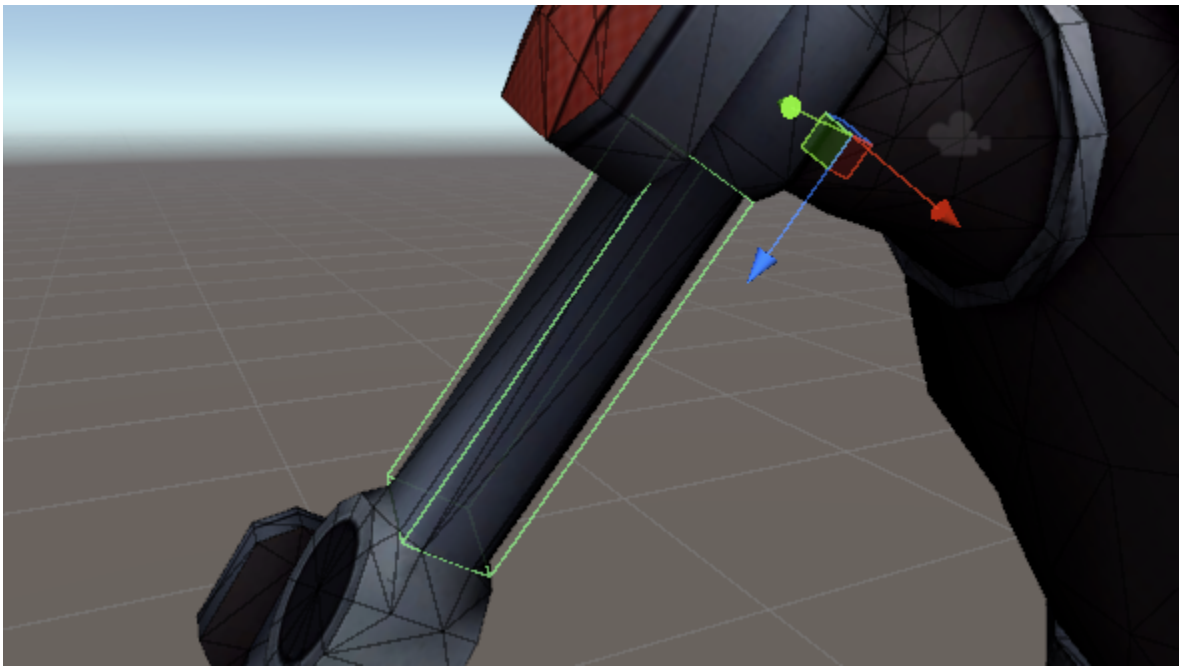
In this example, we are going to use the Right Upper Arm of Unity's free Space Robot Kyle asset.



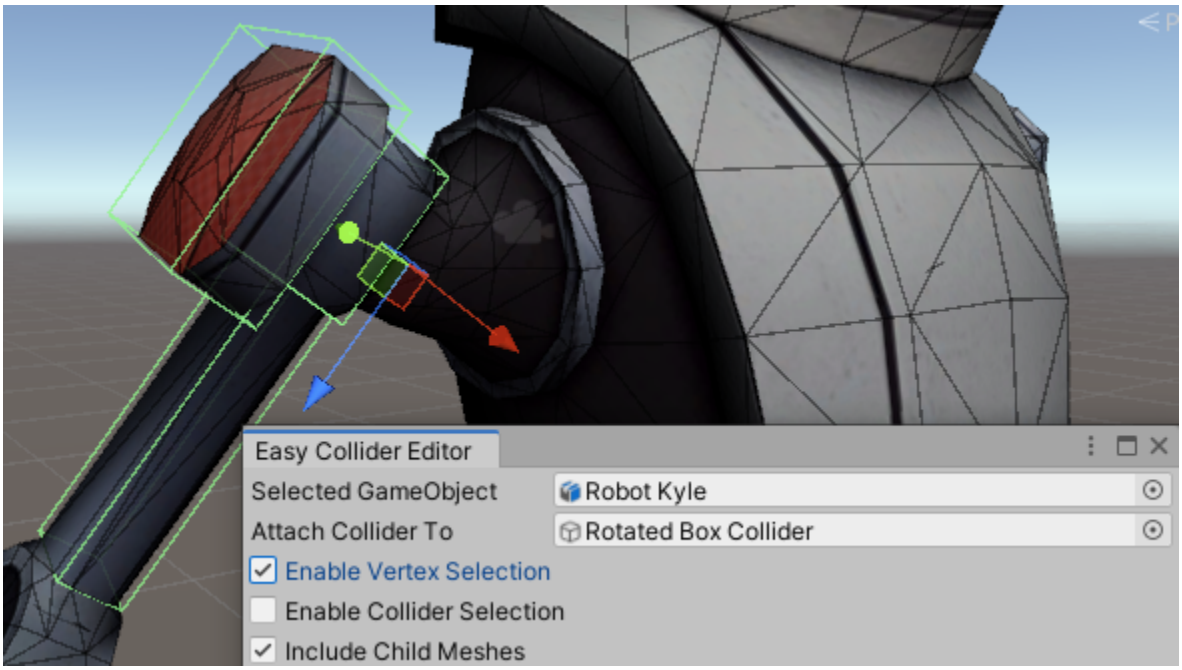
We have selected the root of the skinned mesh, Robot Kyle. As you can see in the image above, the box collider we want to create does not properly align with the arm's axis. This is a case where a rotated box collider would be needed.

Note that 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 that is the joint that rotates when animations are playing, so we need to attach our colliders to that bone.

In the image above, you can also see the three points selected to align the box we want to create. In this example the order they were selected was A, B, then C. The points selected in any of the following orders: A, B, C, or B, A, C, or A, C, B or C, A,B would all create a properly aligned box in this case. Selecting B, C (or C, B), then A, would create a box that is misaligned.



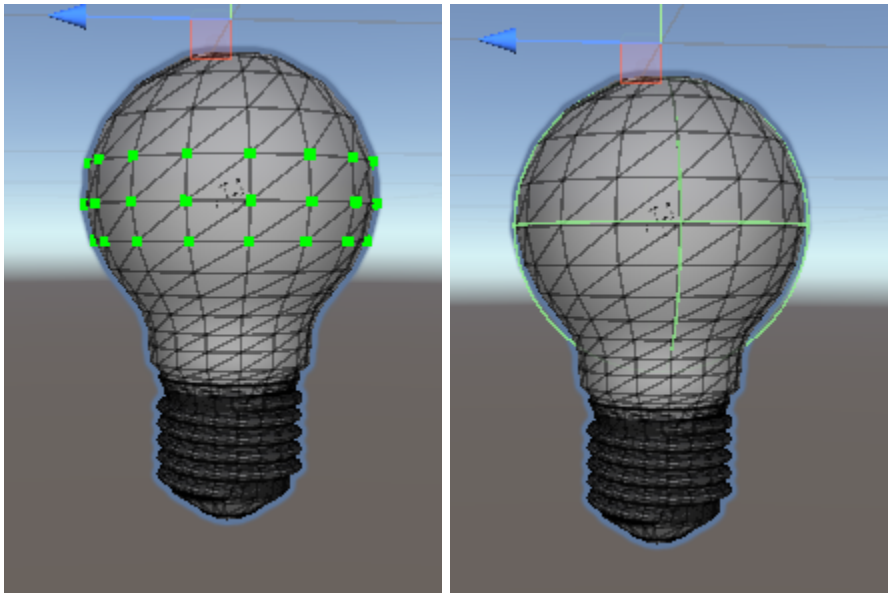
After selecting a few more vertices to create the size of the box, we clicked the Create Rotated Box Collider button. Notice how the box collider is aligned with the vertices we selected above.



Now that we've created one rotated collider for this joint, we can now use regular box colliders. Since the Rotated Box Collider object that contains the first rotated box collider is aligned properly, it can be used in the Attach Collider To field. After this is done, we have created two additional normal boxes by selecting vertices and clicking the Create Box Collider button.

Spheres

Example #1: Min Max on a Static Mesh

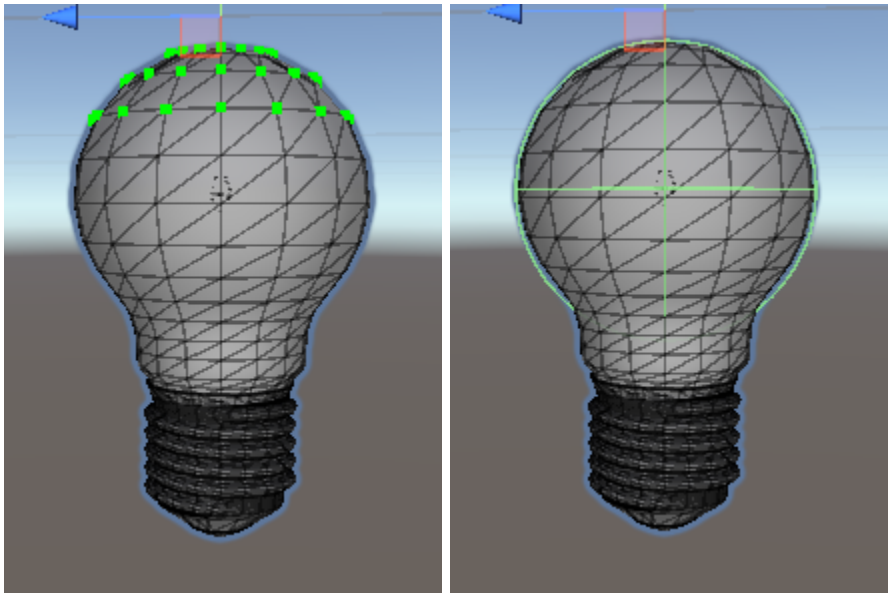


In this example the Sphere Method is set to Min Max. The vertices were selected around the center of the sphere using a ring selection. The result of creating the sphere collider is seen in the image on the right.

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. This is an example where changing the vertices selected changed the resulting sphere collider significantly.

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.

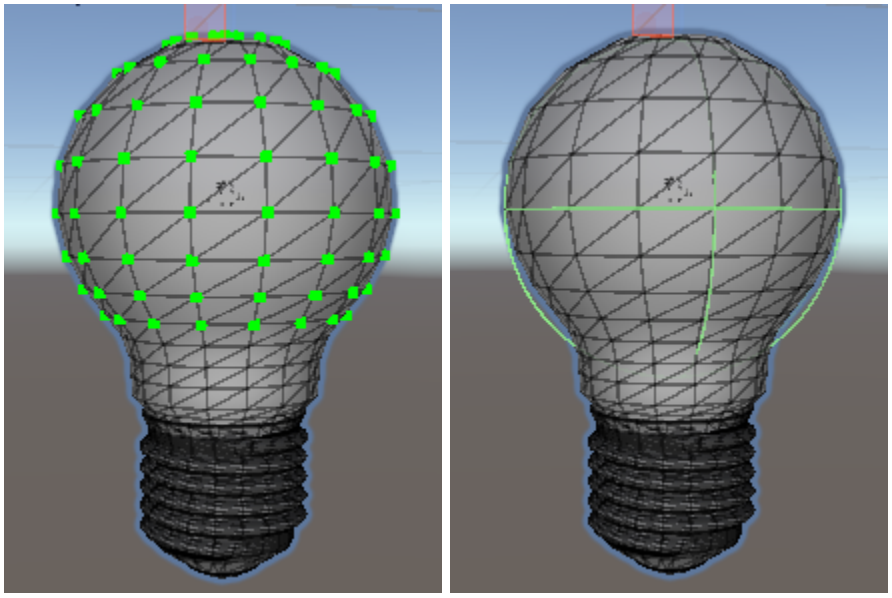


In the case of this lightbulb mesh, it was important I did not do a full box select and try the best fit method. With this lightbulb mesh, there is actually a filament inside with additional vertices. Selecting these vertices would have included them in calculating the sphere, and significantly altered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

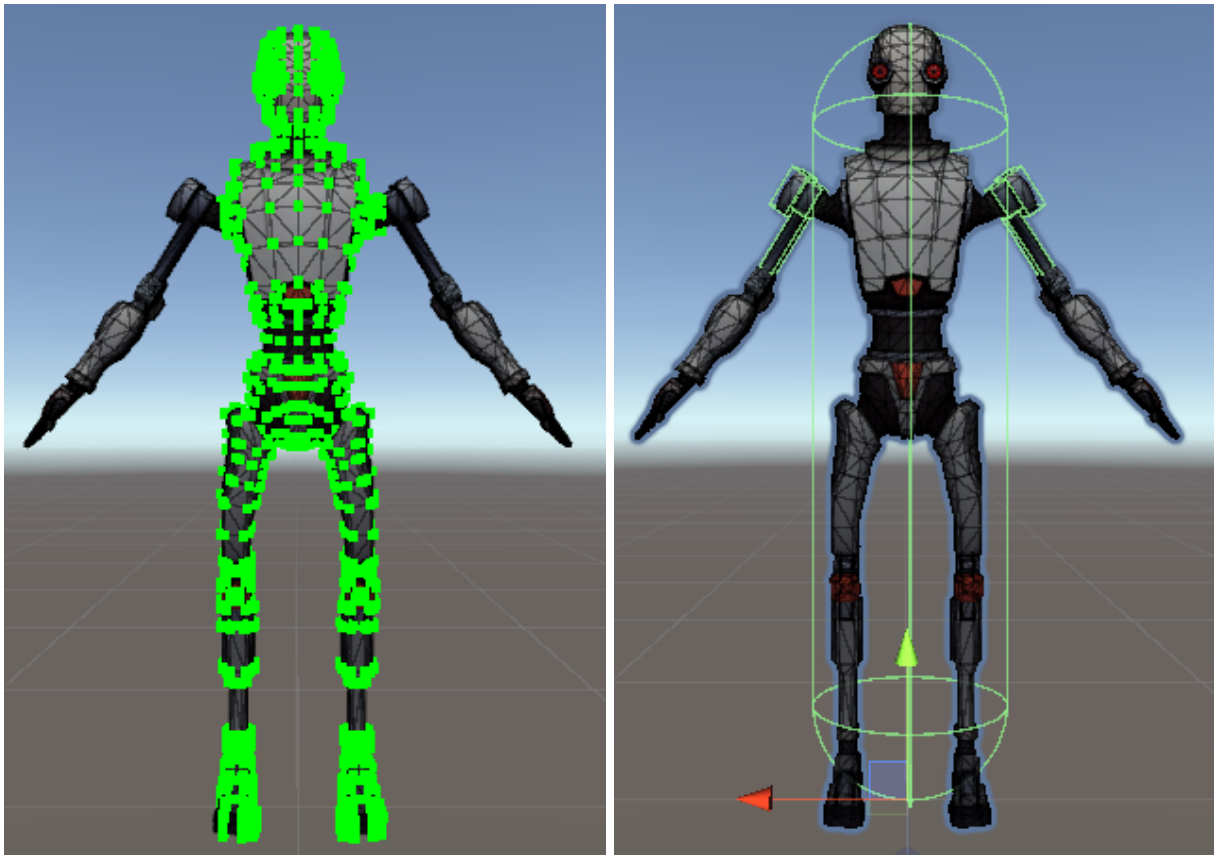


If this mesh had a very large amount of vertices, and they were all selected, this method would fall back to a less accurate but faster method which uses the average position of each vertex. In this case the distance method works pretty well as the furthest two points are on opposite ends of the sphere 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. After clicking the Create Capsule Collider button, the resulting capsule collider is seen in the image on the right.

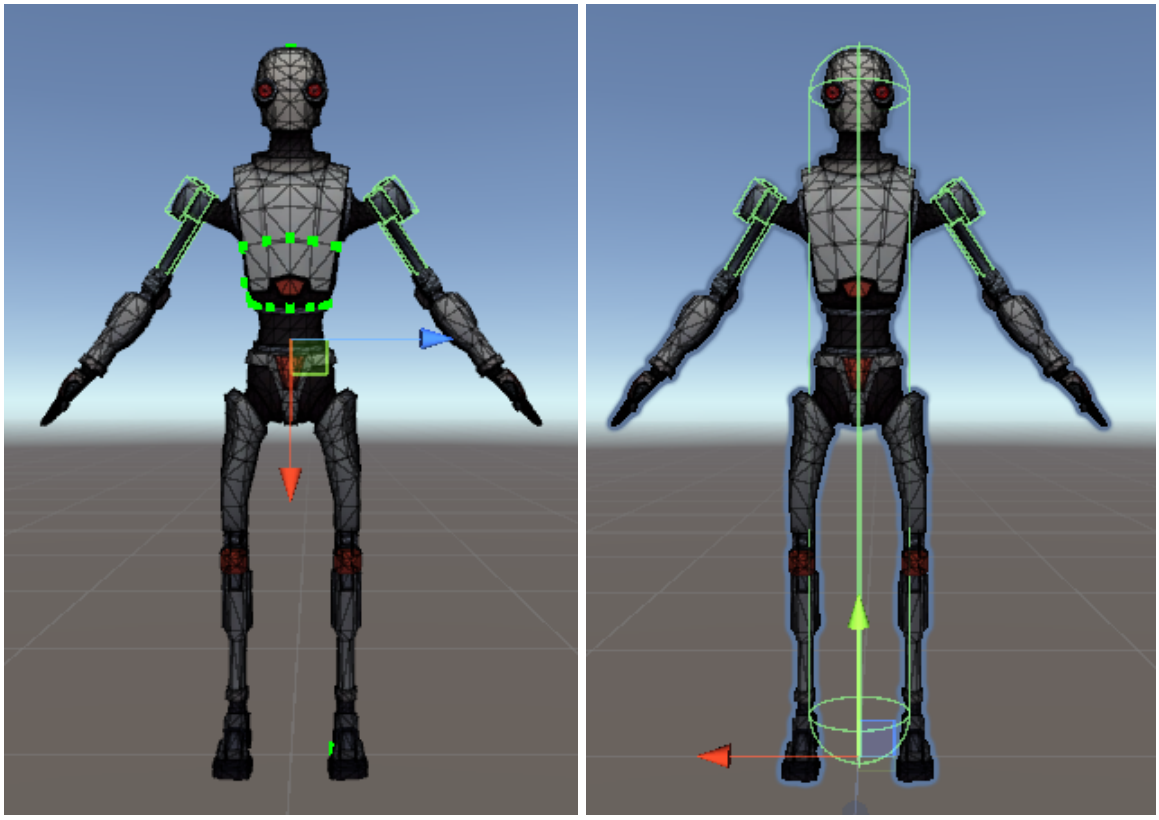


If you look at the generated collider you will notice that it does not contain all of the points that we selected. This is only the case at the top and bottom of the capsule collider, especially noticeable at the feet. In most cases this is actually what we want from a capsule collider.

Methods to add additional height are also available through the Min Max Plus Radius method, and the Min Max Plus Diameter method.

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.

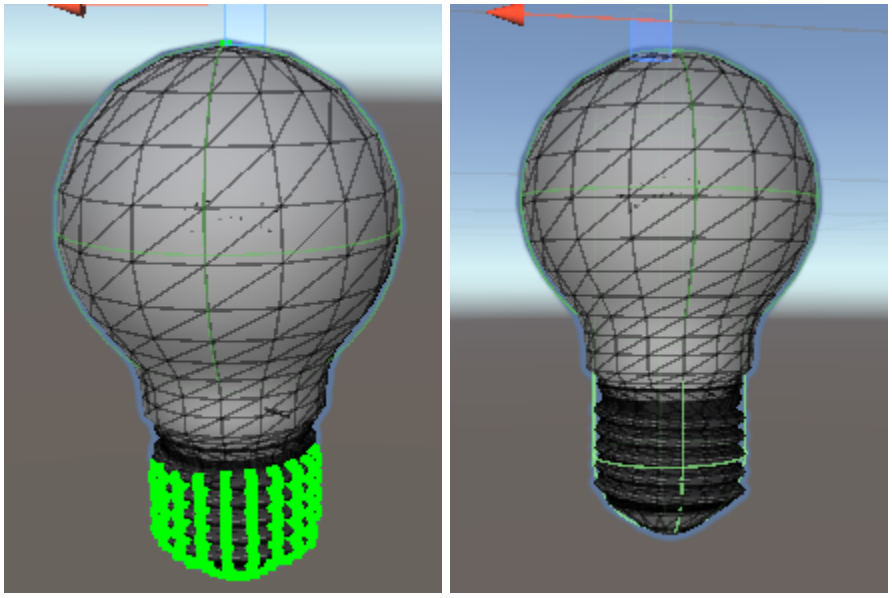


After this we selected two points in the mid section of the character, and used the Ring Select button to quickly get a set of vertices that would roughly lay on a sphere with a radius that matches the collider we want to create. The image on the right is the result of creating a capsule collider with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

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.

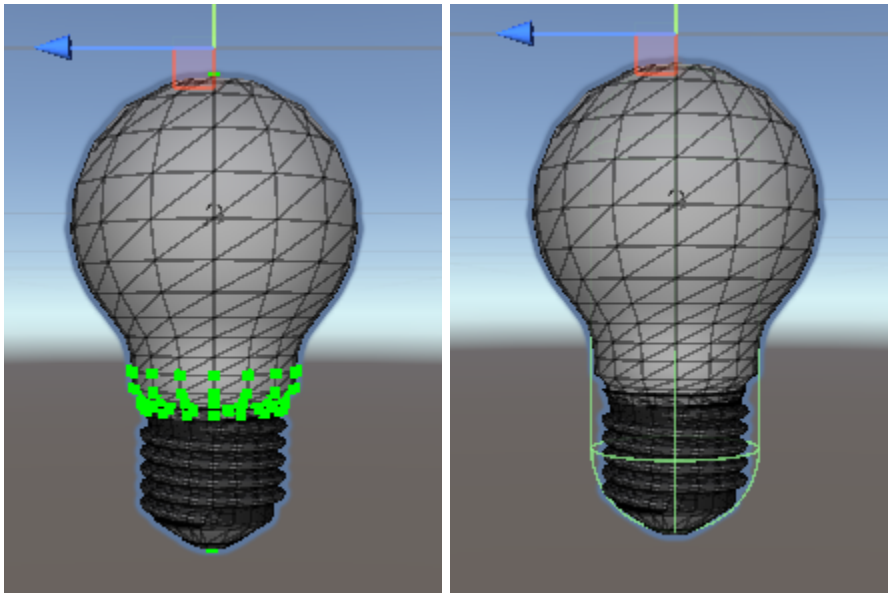
Example #3: Min Max on a Static Mesh

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.



The result of creating the capsule collider is seen on the right. It's hard to see in the image, but the capsule extends all the way up to the top portion of the lightbulb.

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 to define the height of the capsule, were chosen to be the very bottom of the bulb, and the very top of the bulb.

The other vertices selected 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

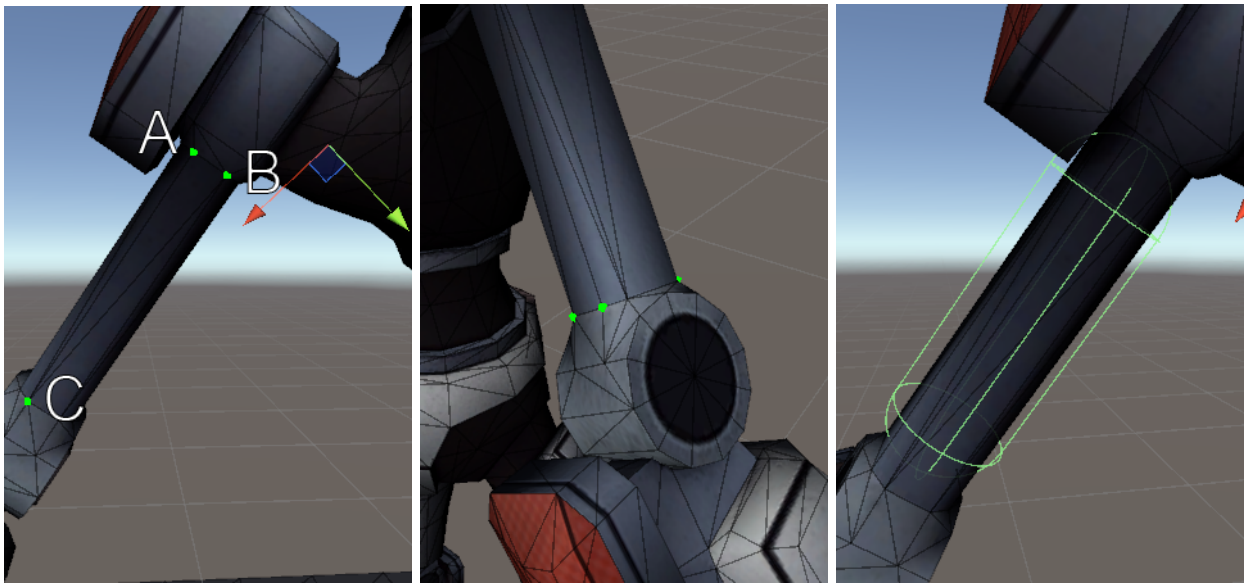
sphere of any radius equal to or larger than the ring itself. By selecting multiple rings, it allows the algorithm to better calculate the radius of a sphere that would best fit the points selected.

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. The first 3 points define the rotation of the collider. In this case, to show how a different selection order would still work, the points were selected in the order A, B, then C in the image on the left below.

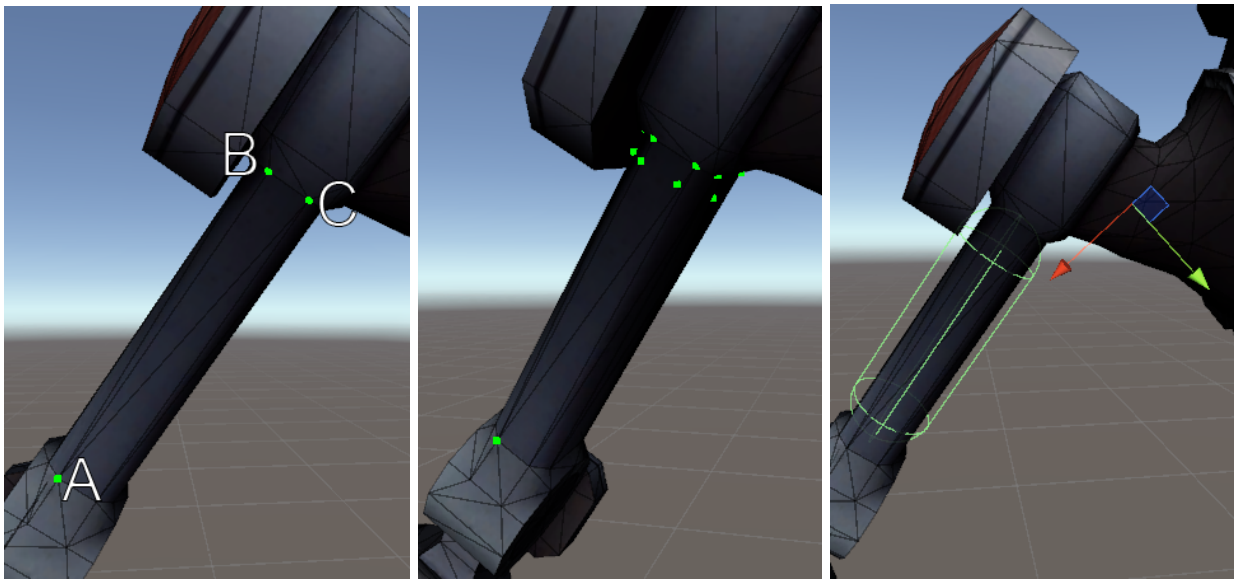


After selecting the three points, a couple more points were selected around the bottom portion of the arm. Although we could have used points around the top as well. The result of creating a rotated capsule collider using the Min Max Method with these vertices selected is shown on the right.

In both the Min Max method and the Best Fit method, the first three points are also included in the other calculations similar to the rotated box collider. This is something to keep in mind.

Example #2: Best Fit on a Skinned Mesh

Below you will see the same example arm, except this time the Best Fit Method was used.



The vertices on the left were selected in the order A, B, and C. The first two points, A and B, are used to calculate the rotation, and the height of the capsule. Since a capsule as it rotates around the height axis, doesn't actually change it's dimensions like a box collider does, the placement of point C is not all that important for defining the rotation.

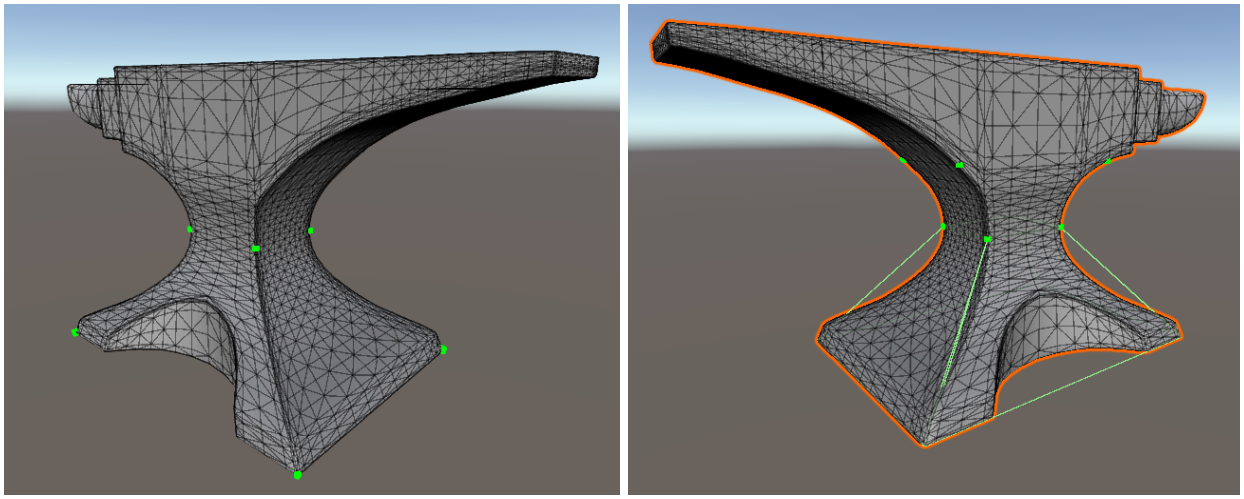
The image in the middle demonstrates how points around the upper arm joint were selected. These were selected since combined, they all lay on the surface of a sphere with a radius we are looking for. Note that general point selection was used to select points that also lay on the surface of the mesh, but aren't actual vertices of the mesh.

The result of creating a rotated collider with these is seen in the image on the right. If you wish to use the best fit method, the key point is to remember that point A and B define the height of the capsule, as well as the rotation of the capsule.

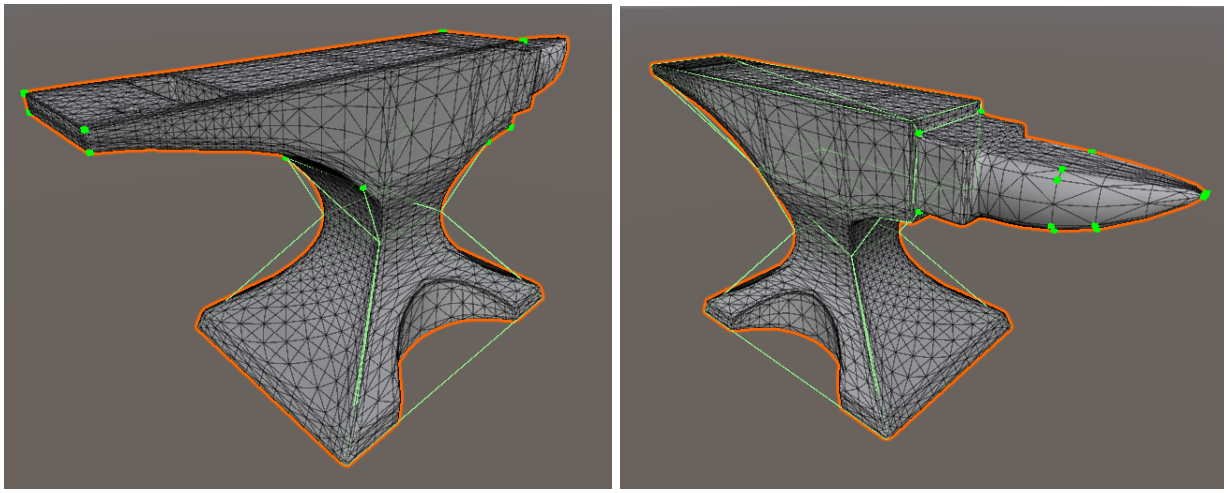
Mesh Colliders

Example #1: Static Mesh

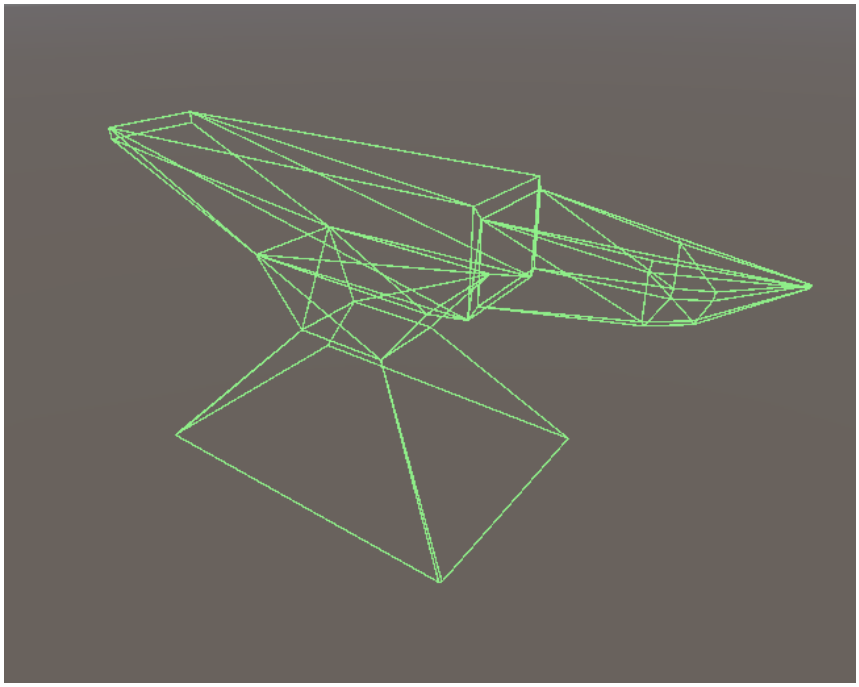
In this example we are going to create several convex mesh colliders to better approximate the shape of an anvil.



In the image on the left, 8 points were selected to create as simple a convex mesh as possible while still maintaining a rough outline of the mesh. In the image on the right you can see the result of creating a convex mesh with those points selected. Additionally, more points were selected to represent the general shape of the rest of the base of the anvil.

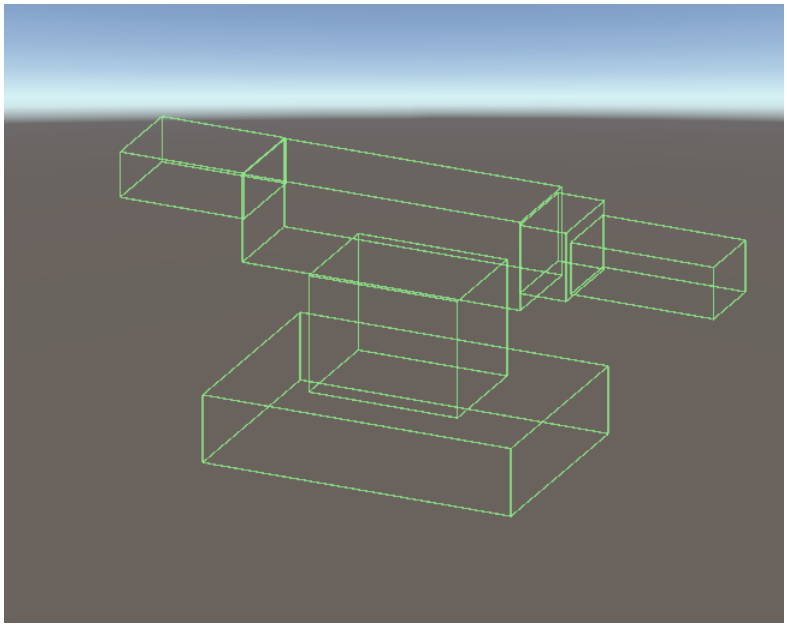


In the image on the left, the points were selected to contain the back half of the anvil. Again a low number of vertices were selected to contain the rough outline of the part of the mesh the mesh collider will contain. The image on the right followed the same process, several more points were selected along the curved surface to better approximate that section of the mesh.



Above is the final result of all of our created convex meshes. To me, this is a good approximation of the anvil mesh and will suit our purposes. If we find we want more accuracy at the base of the anvil, we could remove those two mesh colliders, and then create three mesh colliders to better represent that curved surface.

If this were not a demonstration, we should think about if 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.



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. Collider selection works similar to vertex selection, with the same shortcuts and functionality.

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 used 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.

One important thing to know when merging into a rotated collider is that it only has a different output from non-rotated colliders if the first collider selected is a rotated collider (or on a different transform from the object in the attach to field). So if you wish to merge colliders into a rotated collider, be sure to select the rotated collider first.

Generating Convex Hulls - VHACD

VHACD is an automatic convex hull method. Unfortunately this means that 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.

Generating convex hulls for a single mesh uses the Selected GameObject and Attach To fields just like normal collider creation. The resulting mesh collider can be attached to a different object

set in the attach to field. It also behaves similar to collider creation when Include Child Meshes is enabled. All of the child meshes are processed together, and the resulting convex hulls are all added to the same Attach To object. This behaviour can be changed using the Separate Child Meshes toggle.

General Settings

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 will take an **exceptionally long** time to complete. Additionally when project hull vertices toggle is enabled, the resolution is not nearly as important.

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. This value is limited to 128 without expanding into advanced options. Keep in mind that convex mesh colliders over 256 vertices or triangles generates errors in some versions of unity.

Fill Modes

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.

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. Generally, this allows for increased accuracy in cases of lower resolution. This can be seen quite easily by toggling it on and off and looking at the preview.

Save Hulls as Assets

When this toggle is enabled, the meshes generated by VHACD will be saved in your project. If the meshes are not saved, they will likely eventually be lost and the mesh colliders will not function properly. See the Saving Convex Hulls section below for more information.

Separate Child Meshes

This toggle can be used to generate convex hulls on multiple meshes at once using the same settings. This toggle changes the default behaviour of how meshes are processed. 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 attached to the transform of the mesh that was used in the calculation. Every calculation of VHACD with separate child meshes enabled will use the same parameters.

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. These vertices will be used to create a temporary mesh that is then passed to VHACD. This allows you to portion out sections of a large and complex single mesh.

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 creates a single gameobject as a child of the Attach To object and attaches all convex mesh colliders to that gameobject. Individual Child Objects creates a child on the Attach To Object, and then each convex mesh collider gets added to its own gameobject that is a child of that object.

Advanced VHACD Settings

Advanced Settings

Expanding the Advanced VHACD Settings exposes all of the available parameters. An explanation of what these parameters do can be found by hovering over the name of the parameter with the window focused. 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

An important option exposed in advanced vhacd settings is the Force <256 Tris setting. This option is enabled by default. 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 can 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 from physx in some versions of unity. In other versions the errors are simply hidden.

Default VHACD Settings

The explanation for the parameters is from the official repository of VHACD and gives the best explanation of the parameters than I can provide. The same descriptions are provided in the UI section of this documentation.

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

VHACD Preview

With the new VHACD preview option, it's even easier to change parameters and see the result without having to use undo/redos. Just enable the toggle and adjust parameters. The calculations will automatically run and display the result of the calculation in the scene.

One thing to note when using the VHACD preview is that the resolution is limited to a range of 10,000-128,000. This is because higher resolutions significantly impact calculation times. 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. The folder the asset files are saved in can be changed in preferences. There is also an option to save the meshes in the same folder as the selected gameobject. If it can not find the folder, it will save in the folder specified in preferences. If this option is disabled, the meshes are not saved, and instead only exist on that instance.

When the Separate Child Meshes toggle is enabled, the option in preferences "Save Convex Hull at Selected GameObject's path" will instead try to save each hull at each individual object's path. For example: If you are generating colliders on multiple prefabs at once, the convex hull meshes would be saved in the same folder as each individual prefab if it is found.

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.

Additionally, the current initial implementation only allows for using the Min Max algorithm for spheres and for capsules.

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, complex meshes 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.

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. Once this is done, the skinned mesh renderer is automatically detected and the button to generation colliders on skinned mesh is enabled.

Setting the collider type you want is easy with the drop down menu in the skinned mesh collider generation foldout. Keep in mind that the methods used for both sphere and capsule are always the min/max method. Additionally the settings for creating a collider as a trigger and setting a physic material apply to auto generated colliders as well.

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.

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.

There are also two important lists that you can use when generating skinned mesh colliders. Stop generation after is a list where items in the list get colliders generated on them, but items after do not. This is useful in cases like a head that has multiple bones on it for the eyes, hair, etc. If you stop at the head you won't get all the extra colliders generated on child bones. The exclude transform list prevents colliders being generated on any individual bone in the list.

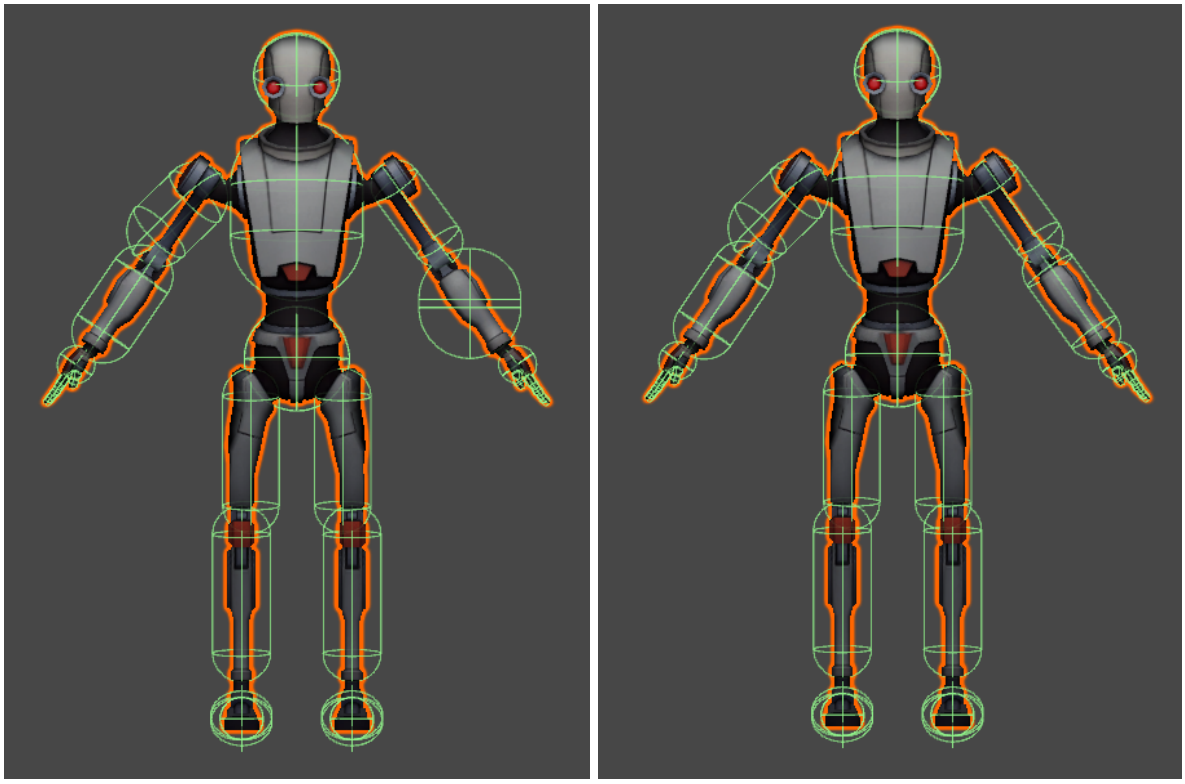
If your skinned mesh is currently imported with the Optimize Game Objects setting enabled you must disable this setting, generate colliders, then enable it again. The generated colliders will be correctly transferred to the exposed transforms. Only transforms that are exposed will retain the generated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

Auto Generated Skinned Mesh Colliders - Examples

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.

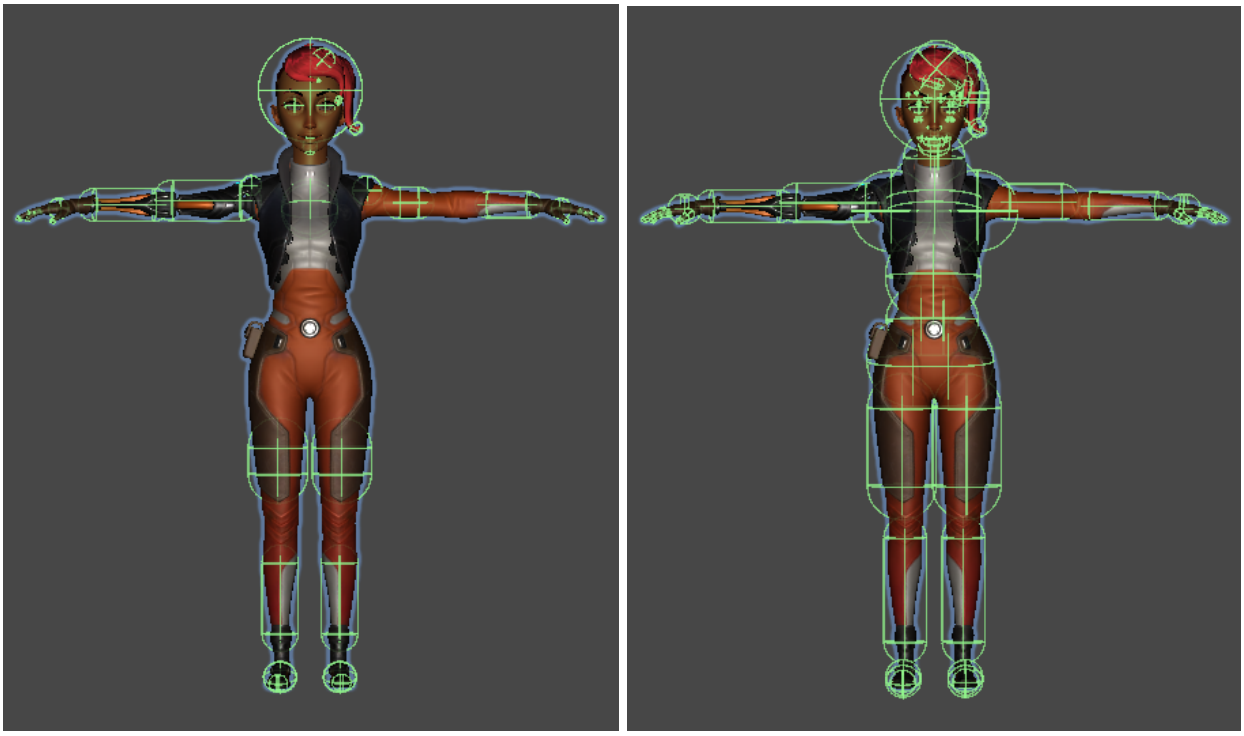
As you can see in the image on the left below, without allowing realignment, the capsule collider on the arm is completely misaligned with the actual mesh. This is because bones were not properly aligned during the mesh creation, skinning, and animation process. In the image on the right however, we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

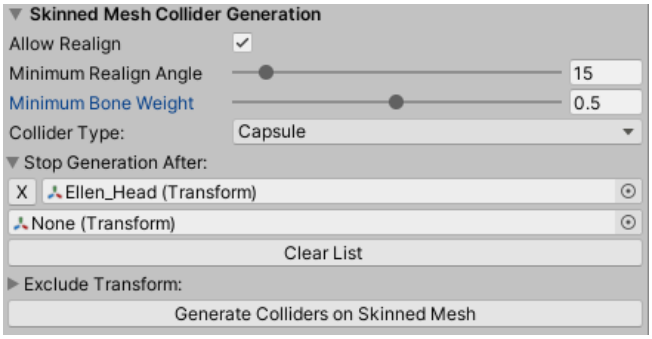
In this example, we're going to use another free Unity Technologies asset to demonstrate the importance of the lists, and minimum bone weight.

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.

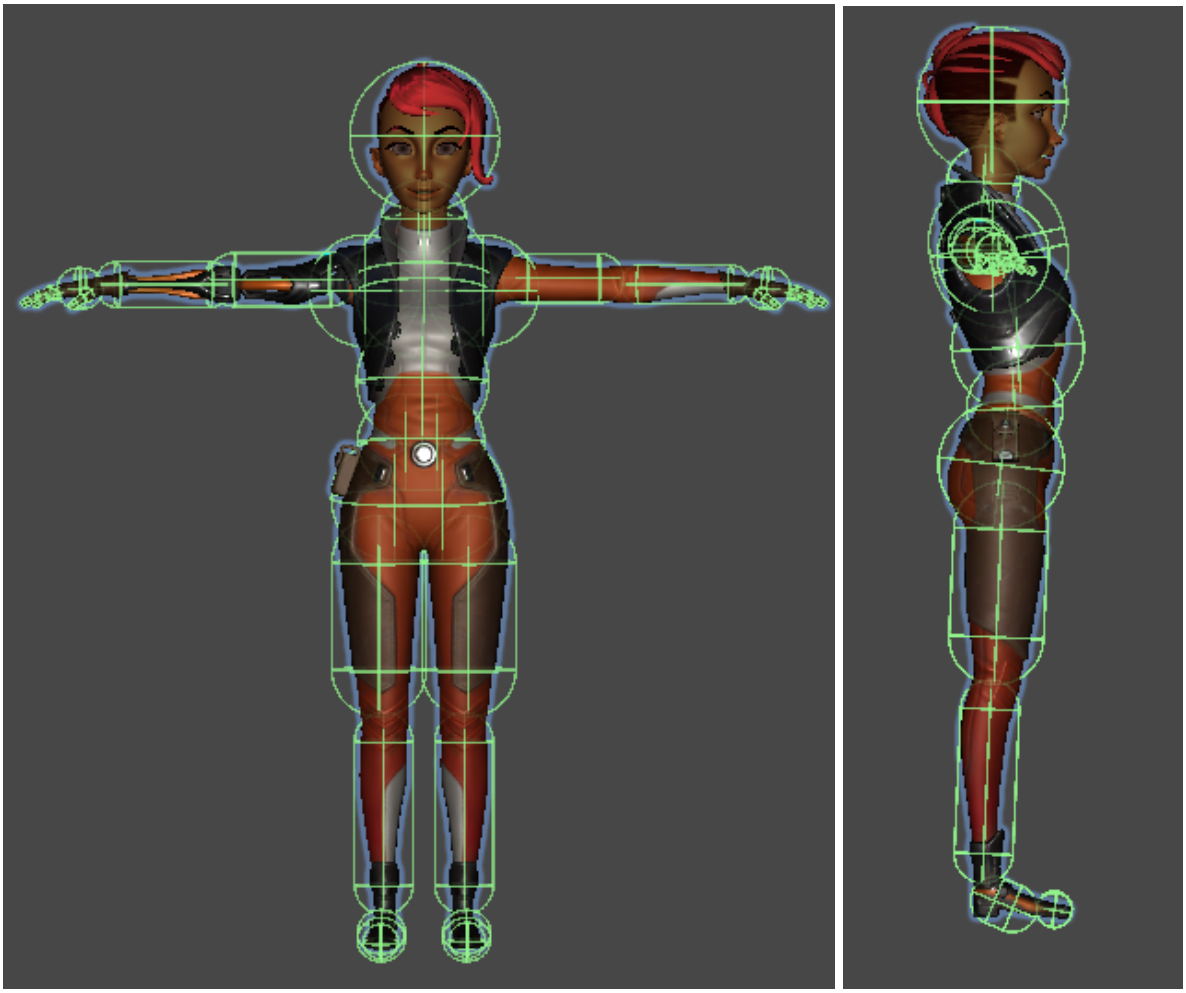


In the image on the right, the minimum bone weight value was changed to 0.5. This resulted in significantly better coverage of the mesh by the colliders.

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 add the head transform to the Stop Generation After list. This will allow us to generate a collider on the head, and ignore the rest of the bones. Note that even though allow realign is checked no extra transforms are generated with these settings as the bones in this skinned mesh are correctly aligned already.



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

The documentation of this section will continue to improve over several updates. In the 6.0 release, several classes have had their #IF (UNITY_EDITOR) directives removed to allow for run-time usage. 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.

Once completely implemented and tested, I will likely provide a second document specifically to cover the runtime API at some point in the future, but for now here is some general information:

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. This documentation will be continually improved over future updates.

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.

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

VHACD

Currently, runtime VHACD is **not** supported. I am currently working on getting it to a point where I am happy with the performance.

To give a little more information for those who are interested or awaiting this feature:

The current plan is to implement a runtime API that allows you to create multiple instances of VHACD that can each run their own calculations. This will be added along with a demo scene that demonstrates how it could be used.

This asset uses the performance branch of VHACD for in-editor usage. Unfortunately, run-time usage of that branch of VHACD currently has a bug that can cause the computation of the convex meshes to get stuck forever. I am still investigating the cause, but as I have more limited C++ knowledge and the complexity of VHACD, it will likely take some time (if I even am able) to solve.

I do currently have an implementation of the slower master branch of VHACD that functions in runtime and works without the bug of the performance branch. I am currently working on further testing to ensure this is the case, as well as finalization of the API and a demo scene that would be included in a released version in this asset.

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. **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

2. **When updating to the newest version I get an error about VHACD.dll.**

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

   If you notice an update is required that does mention 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.

3. **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 not to add, edit, or remove components from the SelectedGameobject and its children while you are selecting vertices.
    - If you did add, editor, or remove components, click the Finish Currently Selected GameObject button, then reselect the gameobject.
    - Try closing the Easy Collider Editor window and reopening, or clicking finish and reselecting the gameobject.

4. **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

5. **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. Look at the examples in the examples section carefully, and try to select a different set of points. Alternatively, enable the preview option and set it to ROTATED_BOX_COLLIDER or use the shortcut key "2" while in the creation tab to change to the rotated box collider preview.

6. **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.

7. **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.

8. **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

9. **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.

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.

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 are also 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