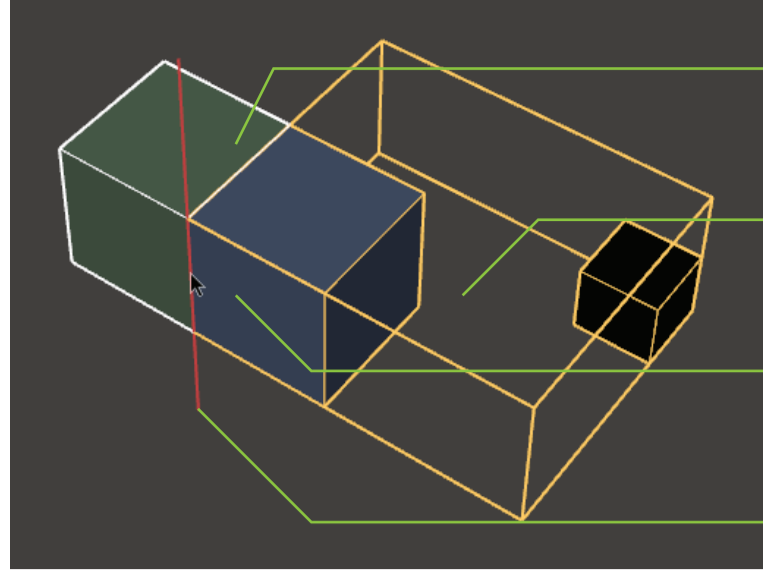


# PickHelper

## A Visual Guide

The PickHelper class in the SketchUp Ruby API can be confusing. The names of its methods isn't immediately obvious. This chart will attempt to break down the structure of the class for a better understanding of the data you get.

### The Test Scene



- C1 #<Sketchup::ComponentInstance:0xe5fe91c>
- G1 #<Sketchup::Group:0xe67e054> (Parent to Group2)
- G2 #<Sketchup::Group:0xe67d67c> (Child of Group1)
- E1 #<Sketchup::Edge:0xe5ff5ec> (Ungrouped - in current context)

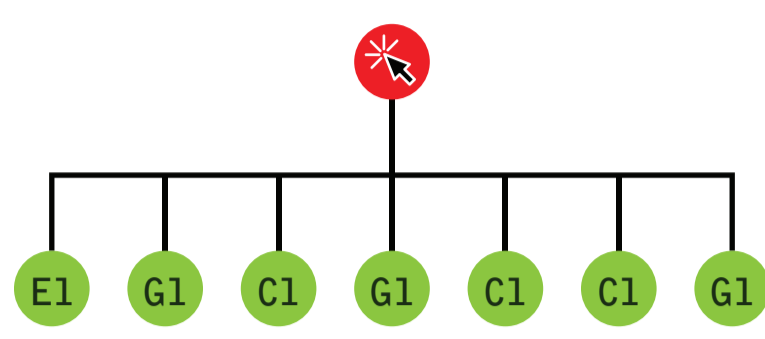
### The Pick

When you perform a pick the pickhelper resolves a tree of possible entities that fit the pick.

Lets see what kind of data we get when we pick a point where two groups and a lone edge all overlap with entities.

### PickHelper.all\_picked

```
pickhelper.all_picked
[
  #<Sketchup::Edge:0xe5ff5ec>,
  #<Sketchup::Group:0xe67e054>,
  #<Sketchup::ComponentInstance:0xe5fe91c>,
  #<Sketchup::Group:0xe67e054>,
  #<Sketchup::ComponentInstance:0xe5fe91c>,
  #<Sketchup::ComponentInstance:0xe5fe91c>,
  #<Sketchup::Group:0xe67e054>,
  #<Sketchup::Group:0xe67e054>
]
```



### PickHelper.all\_picked

The first surprise with the pickhelper is that PickHelper.all\_picked returns an array that might contain duplicates.

As in our example we're getting three references each to our groups.

### So what is going on here?

The PickHelper will also look for the entities inside the groups as possible picks. A group is after all just an abstract concept to organize entities in logical sets.

PickHelper keeps digging until it finds an entity that isn't a Group, ComponentInstance or Image - usually an edge or face.

So in our example we picked a point which matches seven entities:

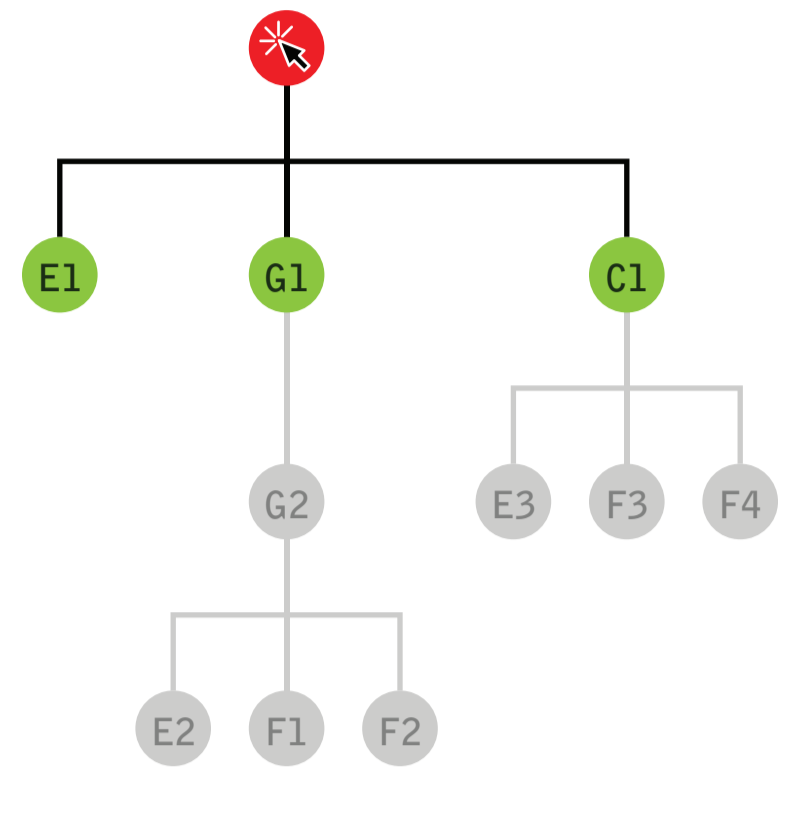
The obvious is E1 which is in the current context.

Then we have the corner edge (E2) and it's faces (F1 & F2) of G2, which is nested inside G1.

Finally we have the corner edge (E3) of C1 and it's two adjacent faces (F3 & F4).

So when we think of how the model hierarchy works we can imagine the pick result to be like the tree illustrated to the right here.

But why did we not get an array with just G1, G2 and E1? Why all the duplicates?



### The Model View of the Pick

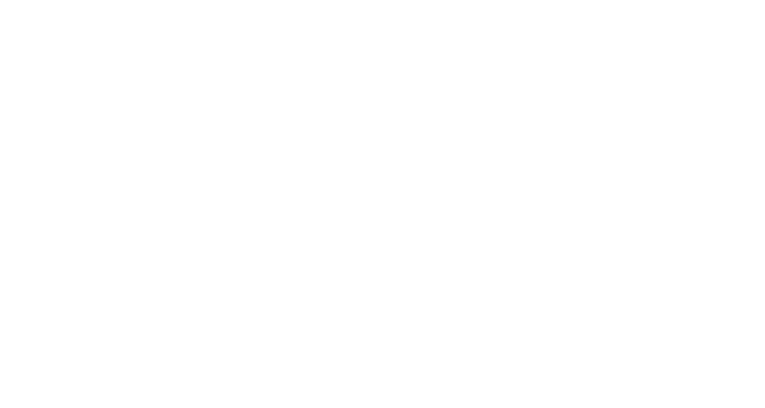
The reason is that PickHelper builds a tree where each branch is a roadmap back up to the root.

PickHelper.all\_picked is just a cross section of this tree - highlighted in green in the right illustration.

**In other words; each item in PickHelper.all\_picked is the first node in each branch.**

The first node in a branch is always an entity located in the current context. (model.active\_entities).

### The PickHelper Results Tree



### Accessing a Branch

```
pickhelper.count
7

pickhelper.path_at(1)
[
  #<Sketchup::Group:0xe67e054>,
  #<Sketchup::Group:0xe67d67c>,
  #<Sketchup::Edge:0xe5f5bd4>
]

# Iterate all pick-routes:
pickhelper.count.times { |index|
  branch = pickhelper.path_at( index )
  p branch
}

pickhelper.depth_at(1)
3
```

Each branch in the pick-tree can be accessed by its index.

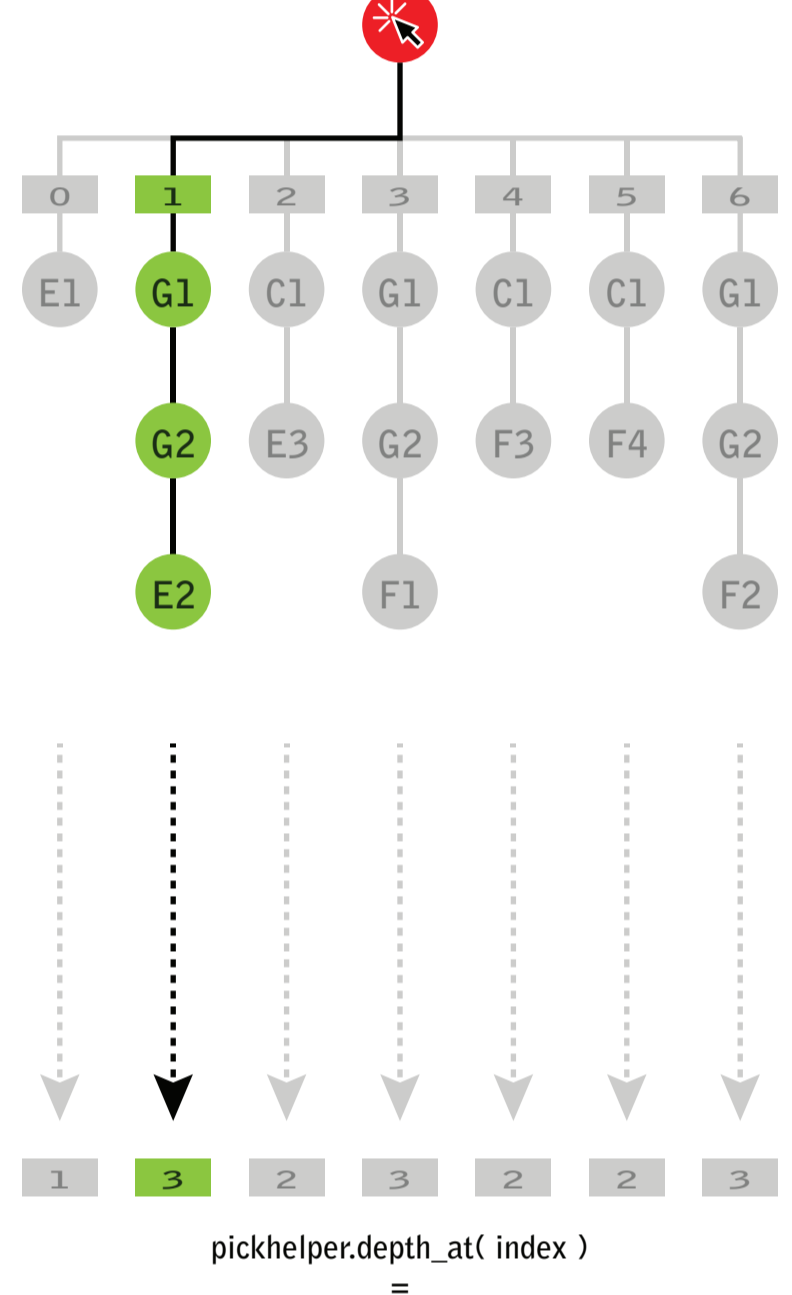
pickhelper.count will tell you how many different pick-routes there are.

**pickhelper.path\_at( index ) will give you an array of entities for the given branch index.**

For each item on a branch you are removed one step from the current context.

pickhelper.depth\_at( index )

pickhelper.path\_at( index ).length



### Leaves

```
pickhelper.leaf_at(1)
#<Sketchup::Face:0xe5f5bd4>

pickhelper.element_at(1)
#<Sketchup::Group:0xe67e054>

pickhelper.transformation_at(1)
#<Geom::Transformation:0xe57a1bc>
```

PickHelper.leaf\_at( index ) will always return an entity from the current context - **the first item on the given branch.**

PickHelper.element\_at( index ) will always return an entity from the current context - **the first item on the given branch.**

PickHelper.transformation\_at( index ) returns the combined transformation for each of the containers between the current context and the leaf.

In our example it means it's the combinations of: G1.transformation \* G2.transformation

That gives us the transformation needed to converts the coordinates of E2 into the coordinates of the current context.

This method is one which name describes very well what it returns - once you know that the pickhelper returns a tree of branches.

**It's the last item on the branch** and it will be an edge, face, construction point or similar - not a group or component.

Note that an Image is an instance just like a Group or ComponentInstance. If you click on an Image in the current context the branch will be an array of the Image and the face inside the Image.

pickhelper.leaf\_at( index ) = pickhelper.path\_at( index ).last

### Best Picks

```
pickhelper.picked_edge
#<Sketchup::Edge:0xe5ff5ec>

pickhelper.picked_face
#<Sketchup::Face:0xe5f5bd4>

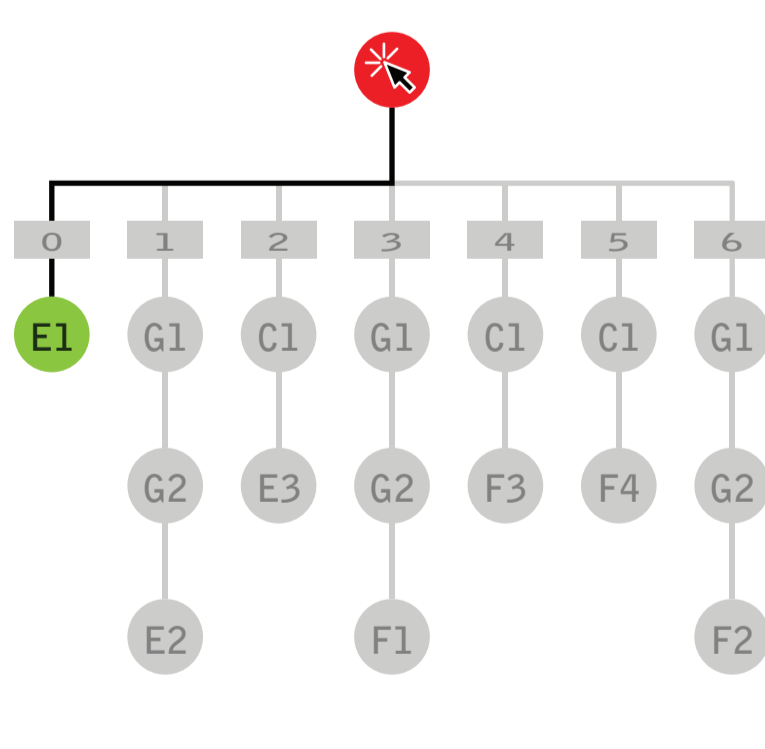
pickhelper.picked_element
nil
```

The order of the branches isn't completely random as one might initially think. **The path index relates to how good a pick SketchUp consider it - lower index means better pick.**

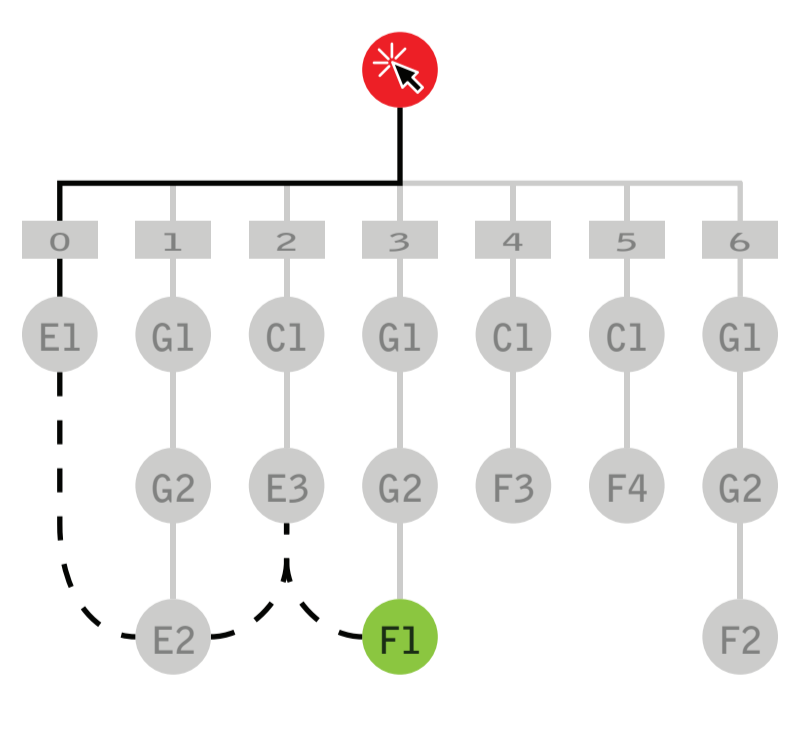
When you use pickhelper.picked\_edge it will return the first edge it finds in the paths available. The edge will always be a leaf.

Same rules applies to pickhelper.picked\_face and pickhelper.picked\_element.

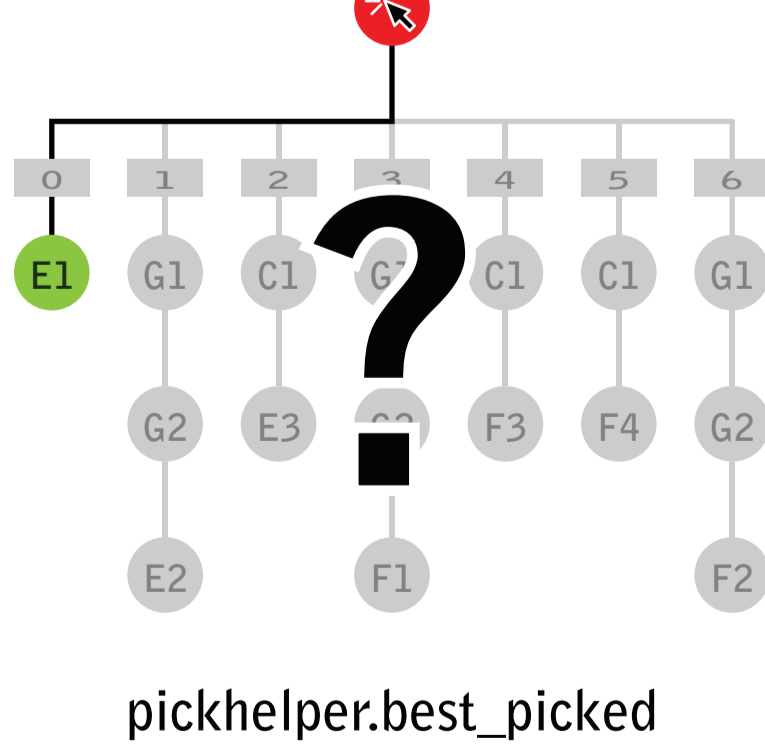
pickhelper.picked\_element returns an entity that isn't an edge, face, group, component or image.



pickhelper.picked\_edge



pickhelper.picked\_face



pickhelper.best\_picked

Quote from the API docs:

**The best\_picked method is used to retrieve the "best" entity picked (entity that you would have picked if you were using the select tool).**

This method does exactly what the description says, but exactly what rules it follows is unclear. The pick is not based on the lowest index.

In our example it happened to be so because we only had one edge in current context while the rest were groups or components.

Had we removed the edge the best pick could be either C1 or G1.

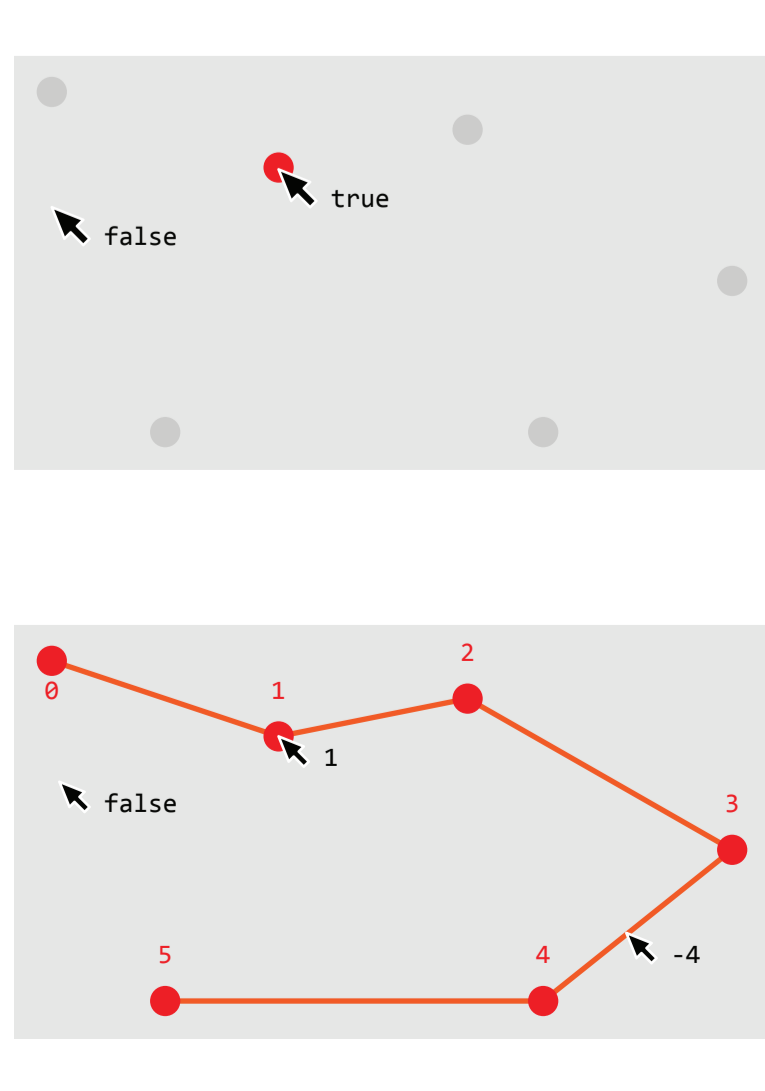
### Segments and Points

```
# Test a single point.
pickhelper.test_point( point, x, y )

# Testing multiple points faster.
pickhelper.in( x, y )
for point in points
  p pickhelper.test_point( point, x, y )
end

# Test a segment.
pickhelper.pick_segment( point, x, y )

# Testing multiple segments faster.
pickhelper.in( x, y )
for segment in segments
  p pickhelper.pick_segment( segment, x, y )
end
```



PickHelper serves many purposes, which is one of the reasons why it's many methods can be confusing. It's not immediately obvious which ones are related.

The last remaining methods of the PickHelper allows you to test a set of 3D points for picking - no entities of any kind.

These can be particularly useful when you create custom tools that draw virtual geometry in the screen for the user to interact with.