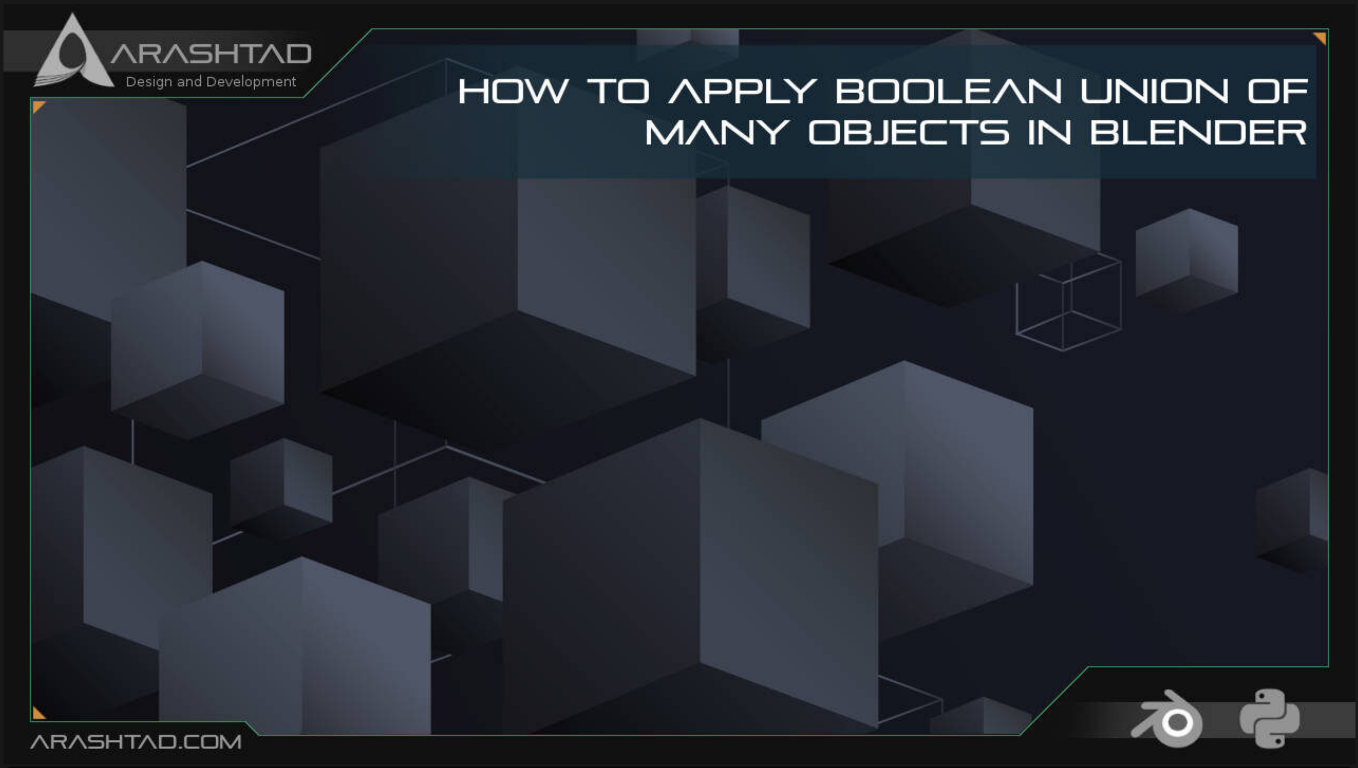
How to Apply Boolean Union of Many Objects in Blender

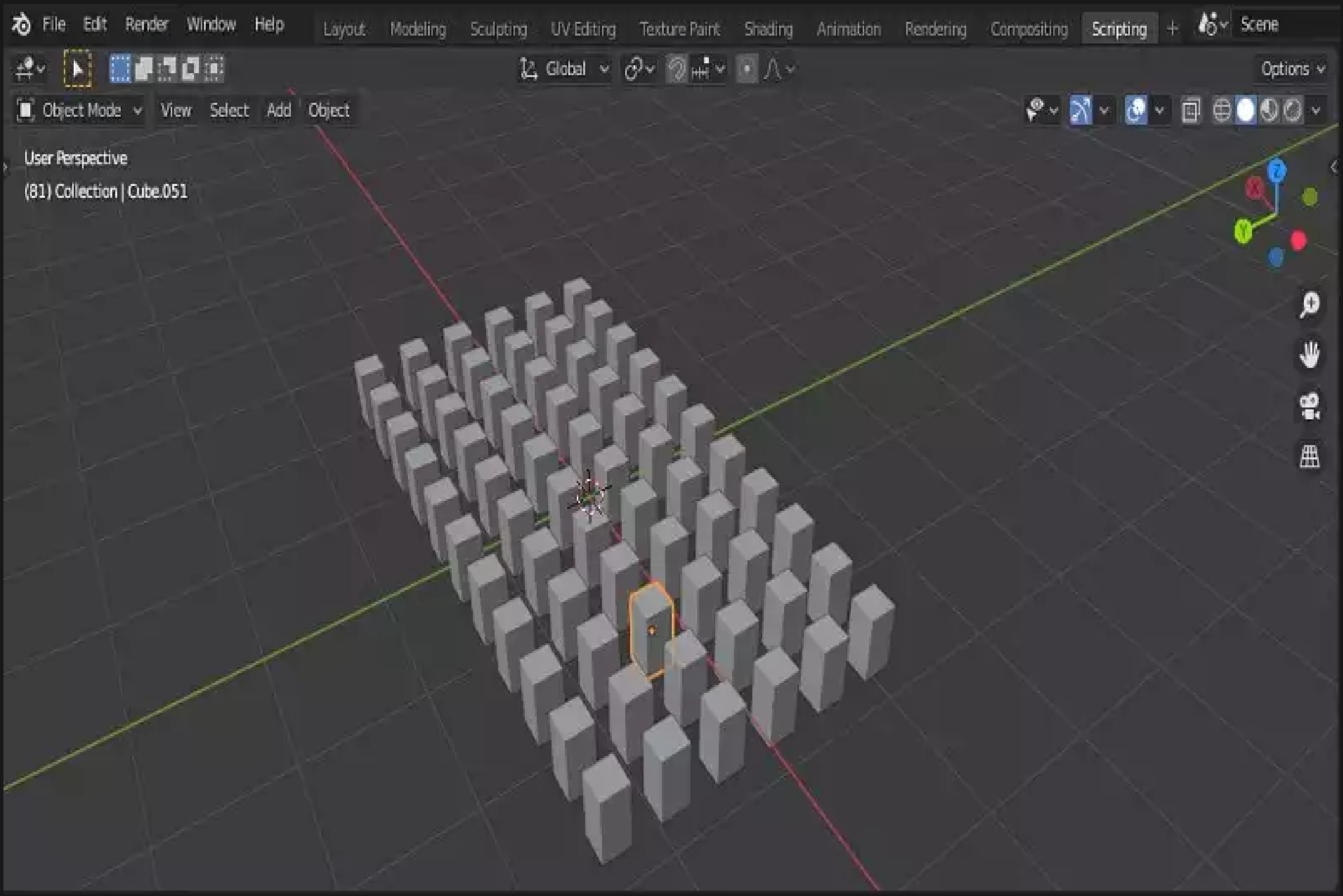
No comments



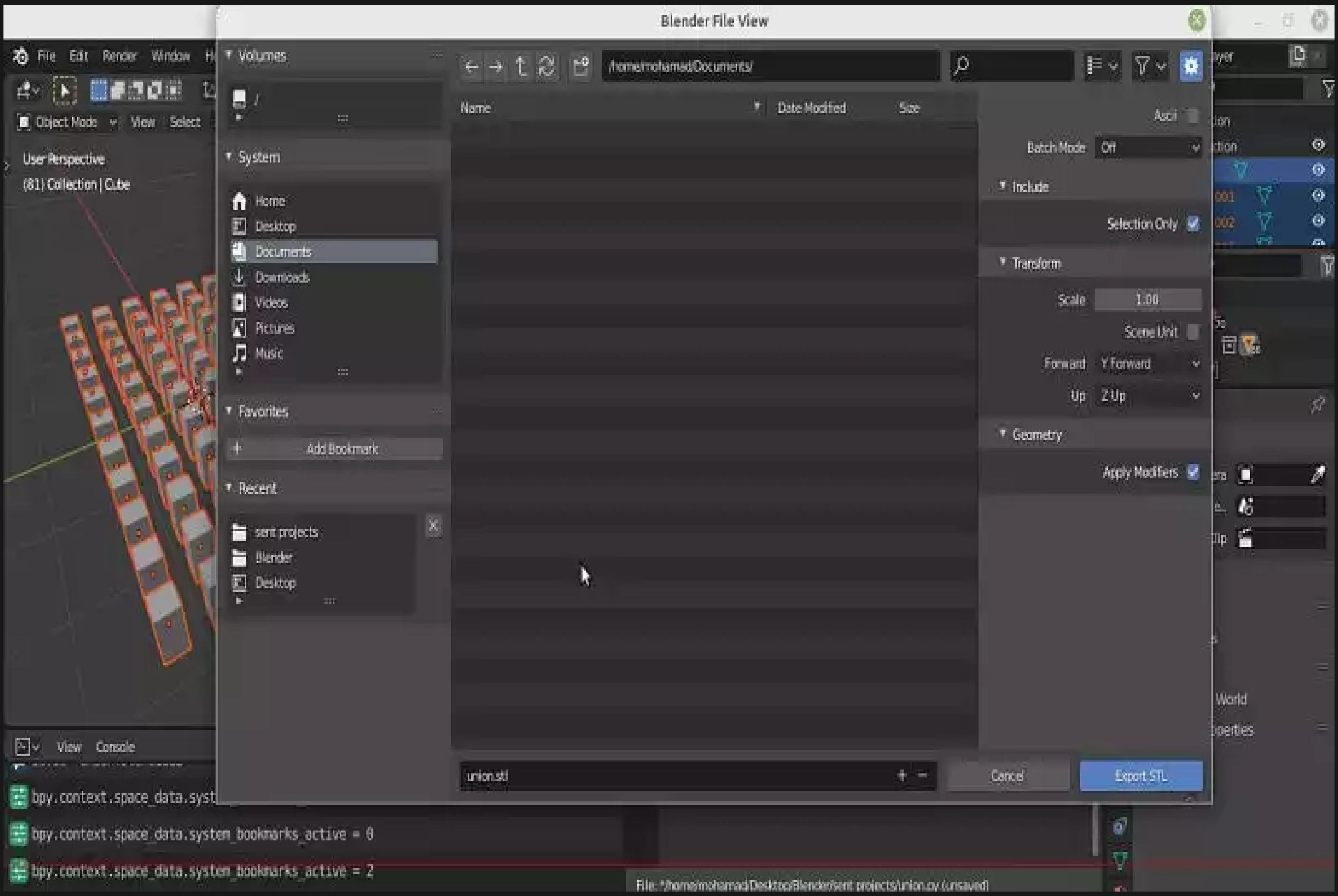
This tutorial covers most (if not all) of the problems concerning the Boolean union of multiple objects in Blender. If you have worked with a Boolean modifier in Blender more than just a try, you must have faced the circumstances in which the result of union would have been a non-manifold mesh or you have wanted to Boolean union a great number of objects and you have ended up with a simple modifier that only does the Boolean union only for a couple of objects that you wouldn't be sure if the result of the union would have non-manifold mesh or not. This article guides you with different ways to solve this issue.

Boolean Union of Multiple Objects in Blender

Suppose you have a lot of cubes that you want them all to merge into one object, if the number of cubes is n, you will have to apply the Boolean union modifier `n-1` times, which takes a lot of time. You may use the Python Blender API and do this job in a loop. This solution is OK until you face with the non-manifold result caused by the Boolean modifier. That is when the issue begins.



To solve the above problem manually (without any scripting), you can get benefit from a useful hack in Blender and which is exporting the objects altogether. To do this, all you have to do is to select all the objects and export them (do not forget to check the Selection Only box). If you import the exported file, you will see that all of the objects have merged into one object.



/home/mohamad/Documents/

Ascii
Batch Mode: Off
Selection Only: ✓
Scale: 1.00
Scene Unit
Forward: Y Forward
Up: Z Up
Apply Modifiers: ✓

union.stl

```
bpy.context.space_data.syst
bpy.context.space_data.system_bookmarks_active = 0
bpy.context.space_data.system_bookmarks_active = 2
```

File: */home/mohamad/Desktop/Blender/sent projects/union.py (unsaved)

There is also another way to do this in a more standard and sophisticated way and that is through scripting. The benefit of this method is that we can use it next to the rest of our code for more complex functionalities rather than just a Boolean union.

Now imagine that we want to create a button that Boolean union any number of cubes at once without any need to export or import anything. The following code will serve this purpose for us:

IMPORTANT NOTE:

Remember that the scripts we are using here are related to Blender version 2.83 and if you are working with any other versions, it is probable that the scripts might differ a little bit, but we will show you ways to find the proper functions if there are any differences at all.

Python Scripts

The below scripts will create a panel inside which you can Boolean union a large number of objects that have been imported in Blender or designed by the user. Notice that the main purpose of the below code is to make you familiar with the procedure and for other Boolean unions of another object with different names and numbers you should modify the code a bit. However, the utility functions remain the same.

```
import  bpy

####################################################################
#####                Utility Functions
####################################################################

def make_custom_context(*object_names, base_context=None, mode=None):
    if base_context is not None:
        ctx = base_context
    else:
        ctx = {}

    if mode is not None:
        assert mode in ('OBJECT', 'EDIT'), "Wrong mode used"
        ctx['mode'] = mode

    objs = [get_object_by_name(obj_name) for obj_name in object_names]
    ctx['active_object'] = ctx['object'] = objs[0]
    ctx['selected_editable_objects'] = ctx['selected_objects'] = objs
    ctx['editable_objects'] = ctx['selectable_objects'] = ctx[
'visible_objects'] = objs
    return ctx

def makeUnionOpt(*object_names):
    ctx = bpy.context.copy()
    if object_names:
        ctx = make_custom_context(*object_names, base_context=ctx, mode=
'OBJECT')
    bpy.ops.object.join(ctx)  # mostly the same as export/import
 combination

def get_object_by_name(obj_name):
    assert obj_name in bpy.data.objects,
"Error getting object by name: {}".format(obj_name)
    obj = bpy.data.objects[obj_name]
    return obj

def deselect_objects():
    bpy.ops.object.select_all(action='DESELECT')

####################################################################
########             Main Panel
####################################################################

class MainPanel(bpy.types.Panel):
    bl_label = "Object Adder"
    bl_idname = "VIEW_PT_MainPanel"
    bl_space_type = 'VIEW_3D'
    bl_region_type = 'UI'
    bl_category = 'Design Automation'

    def draw(self, context):
        layout = self.layout
        layout.scale_y = 1.2

        row = layout.row()
        row.label(text= "Design Automation", icon= 'OBJECT_ORIGIN')
        row = layout.row()
        row.operator("wm_function.myop", text=
"Boolean Union of multiple cubes")

####################################################################
####                  Main UI ?Functions
####################################################################

class WM_Function_myOp(bpy.types.Operator):
    """Click to apply our customized function"""
    bl_label = "Our customized function"
    bl_idname = "wm_function.myop"

    n = bpy.props.IntProperty(name= "Enter the size of the Cube"
, default = 20)

    def execute(self, context):
        n = self.n
        cubes = ["Cube" for i in range(n)]
        #cubes[0] = "Cube"
        if(n >= 10 and n <= 99):
            for i in range (9):
                cubes[i+1] = "Cube.00%d"%(i+1)

            for i in range (n-10):
                cubes[i+10] = "Cube.0%d"%(i+10)

        elif(n < 10 and n >= 2):
            for i in range (9):
                cubes[i+1] = "Cube.00%d"%(i+1)

        cubes = tuple(cubes)
        makeUnionOpt(cubes)

        return {'FINISHED'}

    def invoke(self, context, event):
        return context.window_manager.invoke_props_dialog(self)

####################################################################
#####                         Register and Unregister
```

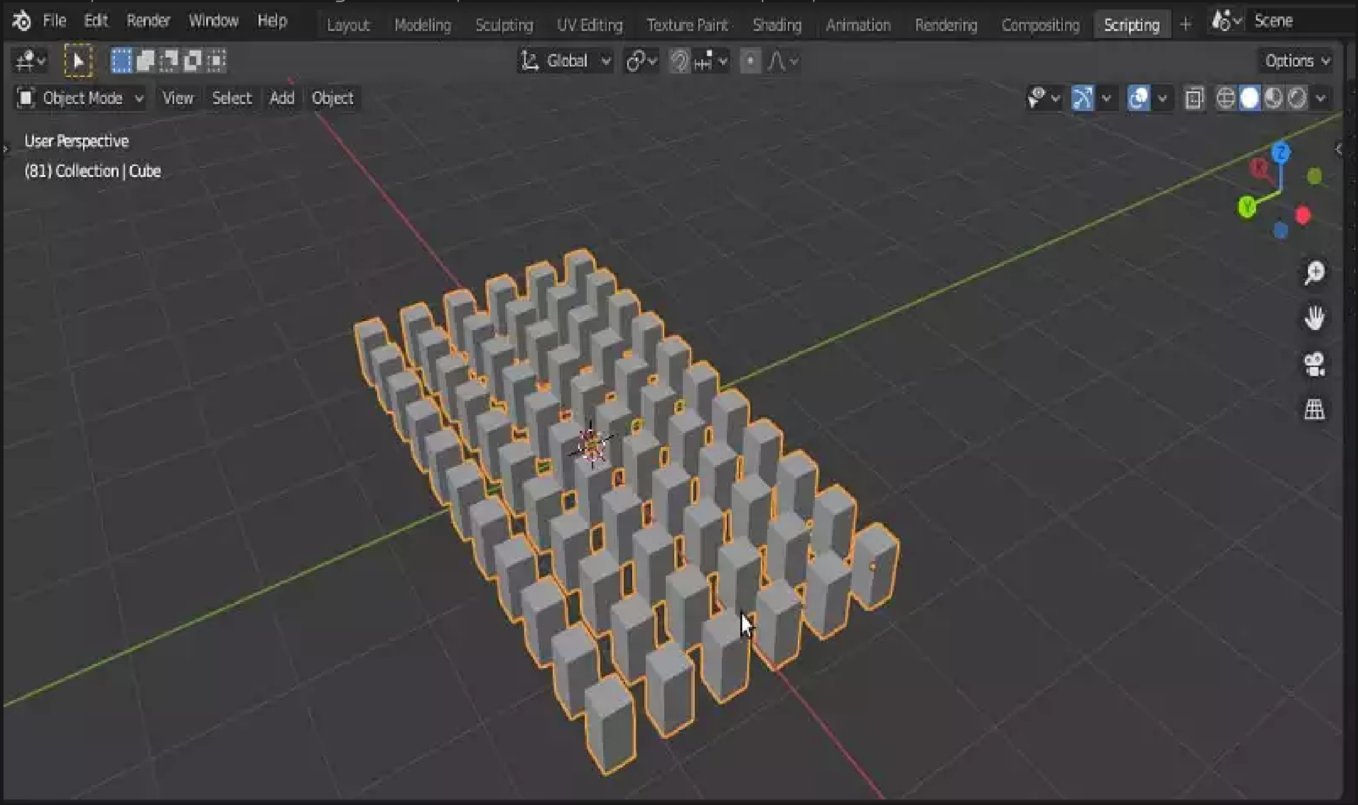
```
#####################################################################

def register():
    bpy.utils.register_class(MainPanel)
    bpy.utils.register_class(WM_Function_myOp)

def unregister():
    bpy.utils.unregister_class(MainPanel)
    bpy.utils.unregister_class(WM_Function_myOp)

if __name__ == "__main__":
    register()
```

And as you can see, we have got one object at the end with the help of just a button.


File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Scripting
Scene
Global
Options
Object Mode View Select Add Object
User Perspective
(81) Collection | Cube

Last Word

In this tutorial, we have managed to Boolean union a large number of objects using the Boolean union other than the one in the modifiers section. This type of union will join the objects and the good thing about this type of union is that the output will not have the non-manifold meshes especially when we have overlap between the objects. We have also provided a manual method so that you can avoid the scripts.

Join Arashtad Community

Follow Arashtad on Social Media

We provide variety of content, products, services, tools, tutorials, etc. Each social profile according to its features and purpose can cover only one or few parts of our updates. We can not upload our videos on SoundCloud or provide our eBooks on Youtube. So, for not missing any high quality original content that we provide on various social networks, make sure you follow us on as many social networks as you're active in. You can find out Arashtad's profiles on different social media services.

Get Even Closer!

Did you know that only one universal Arashtad account makes you able to log into all Arashtad network at once? Creating an Arashtad account is free. Why not to try it? Also, we have regular updates on our newsletter and feed entries. Use all these benefitial free features to get more involved with the community and enjoy the many products, services, tools, tutorials, etc. that we provide frequently.

SIGN UP NEWSLETTER RSS FEED