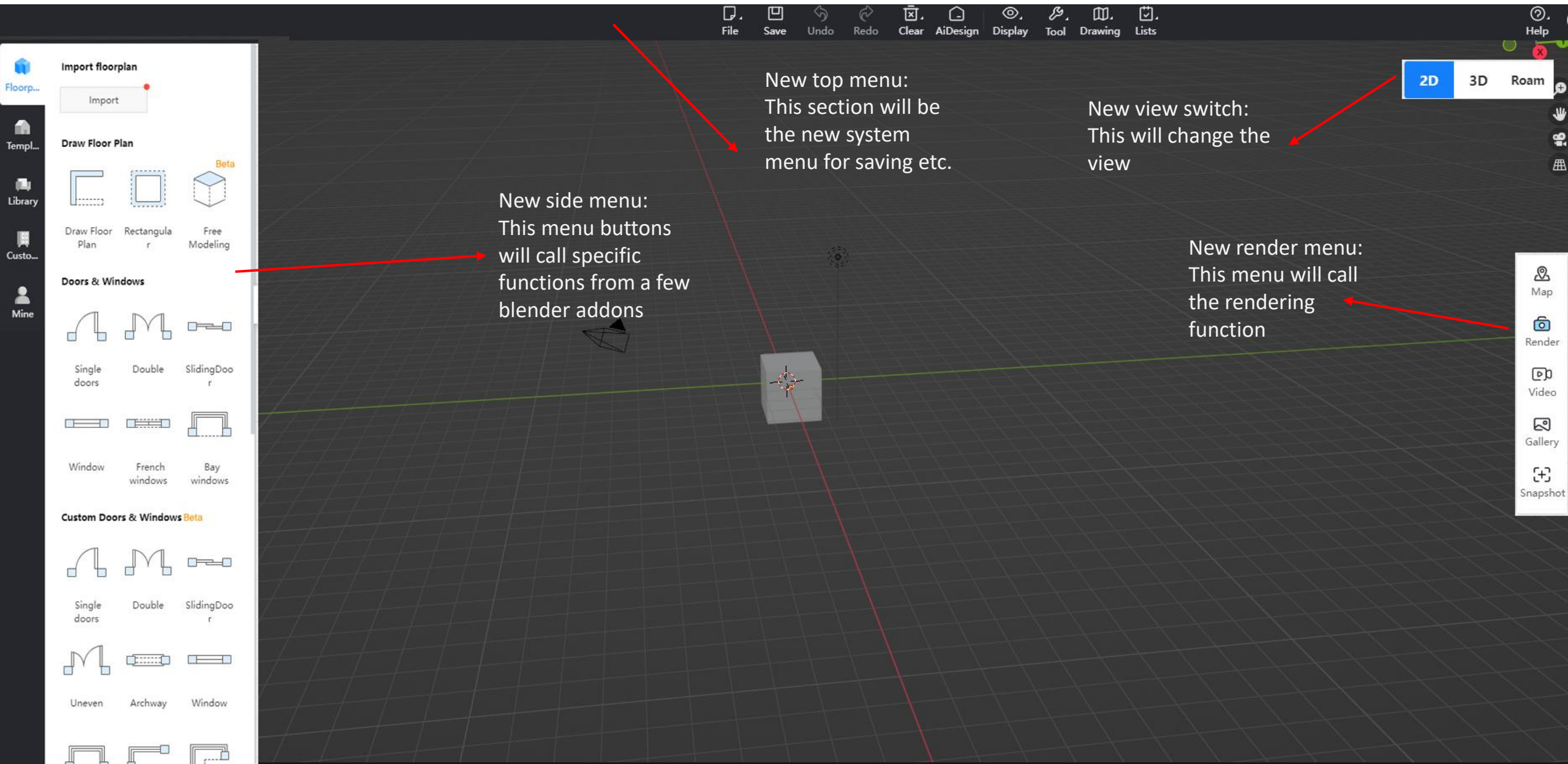


# Sample ui look



*Describe the steps in order to change the blender source code. I need to hide every system and addon menus on the screen. Only the 3d viewport will be shown. There will be a custom new ui for the blender. The new ui will be shown over the 3d view port in the new version of Blender. The new ui will have sections which calls some functions from various addons in Blender. Than i need to compile my own version of Blender.*

#### – ChatGPT answers

- To change the Blender source code, including hiding system and addon menus and creating a custom UI, you'll need to follow these general steps:
  1. Download the Blender source code: You can download the Blender source code from the Blender website (<https://www.blender.org/download/source-code/>). Or (<https://github.com/blender/blender>) Make sure to download the version of Blender that you want to modify.
  2. Install the necessary software: You'll need to install a number of software packages in order to compile Blender. Blender recommends using Visual Studio or Xcode for Windows or Mac respectively. You'll also need to install CMake and Git.
  3. Modify the Blender code: You'll need to modify the Blender code to hide system and addon menus and create a custom UI. This will require knowledge of C++ programming and the Blender API.
  4. Compile Blender: Once you've made your changes to the Blender code, you'll need to compile it. This process will create a new executable file that includes your modifications.
  5. Test the new version of Blender: Once you've compiled your modified version of Blender, you'll need to test it to make sure everything works as expected.
- Here are some more detailed steps for hiding menus and creating a custom UI:
  1. Open the Blender source code in your code editor.
  2. Locate the source code files that define the menu and UI components that you want to hide or modify. These will likely be located in the "interface" folder of the source code.
  3. Modify the code to hide the menus and UI elements. This will likely involve setting the visibility of certain UI elements to false, or removing them entirely from the code.
  4. Define your new UI in a separate file. This will likely be a .cpp file located in the "interface" folder of the source code.
  5. Use the Blender API to call the necessary functions from the various addons in Blender. This will allow you to integrate the addon functionality into your new UI.
  6. Compile your modified version of Blender. Use CMake to configure the build settings, and then use your chosen compiler to build the executable.
  7. Test your modified version of Blender to ensure that everything works as expected.

*Please explain in detail for which sections of the source code to modify in order to achieve what i want. Make them in bullet points so that i can prepare a documentation out of it. I will give the documentation to a developer.*

ChatGPT answers

- here are the steps for modifying the Blender source code to achieve your specific goal:
1. Hide system and addon menus:
    - Open the source code file "interface\_region\_menu.c" located in the "source/blender/editors/interface" folder.
    - Locate the function "interface\_region\_menu\_draw()" and modify the code to hide the system and addon menus. This can be done by commenting out the code that draws these menus or setting their visibility to false.
    - Recompile Blender.
  2. Show only the 3D viewport:
    - Open the source code file "interface\_templates.c" located in the "source/blender/editors/interface" folder.
    - Locate the function "WM\_window\_ui\_draw()" and modify the code to show only the 3D viewport. This can be done by setting the visibility of all other UI elements to false or removing them entirely from the code.
    - Recompile Blender.
  3. Create a custom UI:
    - Create a new source code file in the "source/blender/editors/interface" folder for your custom UI. Give it a unique name, such as "my\_custom\_ui.c".
    - Define your custom UI in the new source code file using the Blender UI API. This can include creating new panels, buttons, and other UI elements.
    - Add code to call the necessary functions from various addons in Blender. You can do this using the Python API or by directly calling C functions from the addons.
    - Add your custom UI to Blender's UI layout. You can do this by modifying the code in the "WM\_window\_ui\_draw()" function as described in step 2.
    - Recompile Blender.

## *How to call addon functions in a new ui of blender?*

### Chat GPT Answers:

- To call addon functions in a new UI of Blender, you'll need to use the Python API or directly call C functions from the addons. Here are the general steps to do this:
  1. Find the Python module or C function that defines the addon functionality you want to use. You can usually find this information in the addon documentation or source code.
  2. Import the Python module or C header file for the addon in your custom UI code. This will give you access to the functions and classes defined in the addon.
  3. Call the necessary functions from the addon in your custom UI code. This may involve passing arguments to the function or creating instances of classes defined in the addon.
  4. Make sure the addon is enabled in Blender before using its functionality in your custom UI.
- Here's an example of how to call an addon function from a custom UI panel in Blender using the Python API:
- In this example, the "my\_addon\_function" function is called from the "my\_addon" addon, and the result is displayed in the UI using a label. Note that you'll need to replace "my\_addon" and "my\_addon\_function" with the actual names of the addon and function you want to use.

```
import bpy

class MyCustomPanel(bpy.types.Panel):
    bl_label = "My Custom Panel"
    bl_idname = "OBJECT_PT_my_panel"
    bl_space_type = 'VIEW_3D'
    bl_region_type = 'UI'

    def draw(self, context):
        layout = self.layout
        addon_prefs = context.preferences.addons['my_addon'].preferences

        # Call a function from the addon
        result = addon_prefs.my_addon_function(arg1, arg2)

        # Display the result in the UI
        layout.label(text=result)
```

*how to compile my own version of Blender? Explain in details and bullet points*

**Chat GPT Answers:**

Compiling your own version of Blender can be a complex task, but it can also be rewarding and provide you with more flexibility and customization options. Here are the steps to compile Blender:

1. Install the necessary tools:

- a. Install a C++ compiler, such as Visual Studio (for Windows), Xcode (for Mac), or GCC (for Linux).
- b. Install CMake, a cross-platform tool used to build software.
- c. Install Git, a version control system.

2. Clone the Blender source code:

- a. Open a command prompt/terminal and navigate to the directory where you want to store the Blender source code.
- b. Clone the Blender Git repository with the following command:

`git clone https://git.blender.org/blender.git`

3. Build Blender:

- a. Create a new directory inside the Blender source code directory called "build".
- b. Open CMake and set the source code directory to the Blender source code directory and the build directory to the newly created "build" directory.
- c. Click on "Configure" and select your C++ compiler.
- d. Once the configuration process is complete, click on "Generate".
- e. Open the generated project file with your C++ compiler.
- f. Build the project in your C++ compiler. This will create the executable file for Blender.

4. Test Blender:

- a. Navigate to the "build" directory and locate the "blender" executable file.
- b. Run the "blender" executable file and test it to ensure that it is functioning correctly.

Congratulations! You have successfully compiled your own version of Blender. You can now modify the source code and rebuild Blender to create a customized version of the software

# 1- Possible addons to use – home builder 3

The Home Builder 3 addon is an open-source asset library for Blender that focuses on simplifying the process of creating interior architectural spaces [\[2\]](#). Here are the steps to call functions of the Home Builder 3 addon from a new UI of Blender:

1.Install Home Builder 3: Since Blender 3 has implemented an asset browser, you can simply install Home Builder 3 as a standard addon to Blender [\[1\]](#).

2.Import the Home Builder 3 module: In your Python script, import the Home Builder 3 module using the following line of code: **import homebuilder3** [\[2\]](#).

3.Call the Home Builder 3 functions: To call a Home Builder 3 function, use the following syntax:

**homebuilder3.<function\_name>(<arguments>)**. Replace

**<function\_name>** with the name of the function you want to call, and **<arguments>** with any arguments that the function requires [\[2\]](#).

4.Integrate the Home Builder 3 functions into your new UI: Create a new panel or operator in your Blender UI. Add a button or other user interface element to trigger the Home Builder 3 function. In the button or other element's event handler, call the Home Builder 3 function using the syntax described in step 3.

Optionally, pass any necessary arguments to the Home Builder 3 function based on user input or other factors [\[2\]](#).

5.Test your integration: Run your Blender script and test the new UI elements you created. Verify that the Home Builder 3 functions are being called correctly and are behaving as expected. Debug any issues as necessary by reviewing error messages and checking your code against the Home Builder 3 documentation or source code [\[2\]](#).

By following these steps, you can integrate the Home Builder 3 addon's functionality into your own custom Blender UI.