Kitchen Configurator Design Document

This kitchen configurator is being designed for a major home improvement company for use in its showrooms. Each showroom will have several kiosks at which clients can create a custom kitchen layout and easily order all of the parts needed based on their design. Most of the showrooms already have these kiosks in place, but they are several years old and have lower-end machines. Despite this, the configurator is expected to run smoothly and feature high-quality lighting and realistic models and materials.

Scale

All kitchen components were modeled according to their real-world specifications. This accurate scale should be maintained in the configurator to ensure that kitchens can be created as the client has designed. As a point of reference, the kitchen island is 0.962 meters tall at the top of its granite surface and the tall cabinet is 2.59 meters.

Naming Conventions

In Unity, all model groups should be named using title case. For example, a small cabinet should be named "Small Cabinet." Note that no underscores or camelCase should be used, as the name of the group will be displayed on the user interface when the object is selected. Child objects of these groups should follow the same naming convention.

There are a total of 14 meshes for this configurator:

- Kitchen*
- Cabinet End Cap
- Counter End Cap
- Island
- Cabinet Sink
- Refrigerator Cabinet
- Dishwasher
- Left Raised Cabinet
- Center Raised Cabinet
- Right Raised Cabinet
- Corner Cabinet
- Double Cabinet
- Single Cabinet
- Tall Cabinet

*This is the room mesh within which all other models will appear.

No animations have been created for these models, so any that might appear on them should be removed.

Lightmap UVs

No lightmap UVs were generated for the meshes in Maya, so these need to be auto generated in Unity on asset import.

Textures and Materials

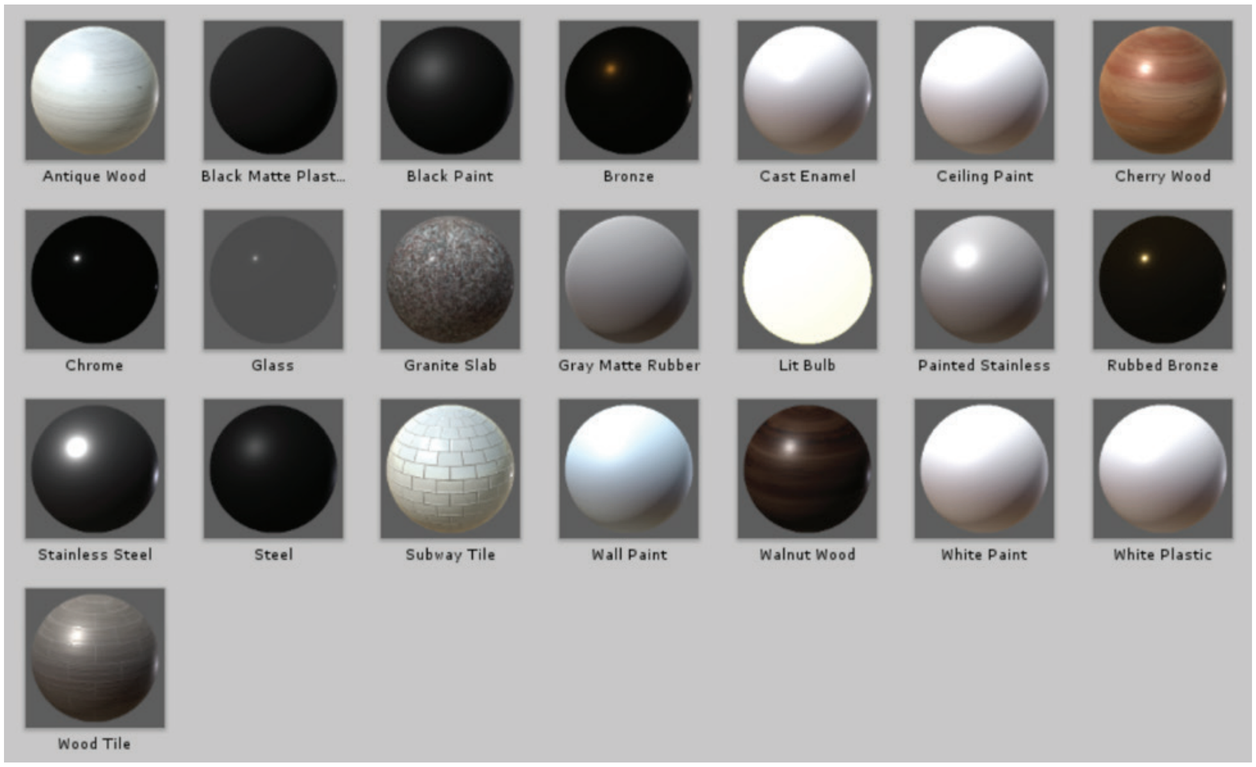
The artist team has exported all meshes with media embedded, so they must be extracted at import. Additional optional textures have been included in the file transfer, and materials for these must be created in Unity.

All materials should be physically based and should appropriately reflect the material that they're representing. Some details may be fine-tuned in Unity if necessary to improve the appearance.

As with the models, the names of the materials will be visible in the UI, and therefore must be named appropriately.

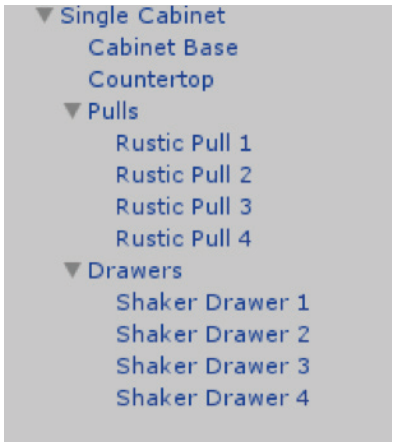
There are a total of 22 materials:

- Antique Wood
- Black Matte Plastic (Metallic: 0, Roughness: 1)
- Black Paint (Smoothness: 0.463)
- Bronze (Color: 1C1101FF, Roughness: 0.261)
- Cast Enamel (Metallic: 0, Smoothness: 0.738)
- Ceiling Paint (Smoothness: 0.4)
- Cherry Wood
- Chrome (Metallic: 1, Smoothness: 1)
- Glass (Transparent, Alpha: 0, Metallic: 0, Smoothness: 1)
- Granite Slab
- Gray Matte Rubber
- Lit Bulb (Albedo: 5C5205FF, Emission color: FDFFA3, Intensity: 1.2)
- Painted Stainless (Metallic: 0.146, Smoothness: 0.585)
- Rubbed Bronze
- Stainless Steel (Metallic: 0.85978, Roughness: 0.33)
- Steel (Metallic: 0.506, 0.44)
- Subway Tile
- Wall Paint (Smoothness: 0.463)
- Walnut Wood
- White Paint (Smoothness: 0.463)
- White Plastic (Smoothness: 0.615
- Wood Tile (Normal map: 0.5)



Prefabs

The programmers have created a script that allows all of the configurable components of a selected mesh to be displayed on a UI. In order to achieve this effect, all objects must be grouped and tagged appropriately. All mesh groups should have a top-level parent object that is an empty GameObject. Meshes should be grouped beneath the empty GameObject. Similar meshes, such as doors or drawers, should be grouped together beneath another empty Gamebject. See below for an example:



```
▼ Single Cabinet
    Cabinet Base
    Countertop
  ▼ Pulls
      Rustic Pull 1
      Rustic Pull 2
      Rustic Pull 3
      Rustic Pull 4
  ▼ Drawers
      Shaker Drawer 1
      Shaker Drawer 2
      Shaker Drawer 3
      Shaker Drawer 4
```

- The top level parent group should be tagged as Kitchen
- The countertops should be tagged as **CounterTop**
- The drawer group should be tagged as **Drawers**
- The pulls group should be tagged as Pulls
- The doors group should be tagged as **Doors**
- The knob group should be tagged as **Knobs**
- Appliances should be tagged as Appliance
- Base objects (such as cabinet bases) should be tagged as **Base**

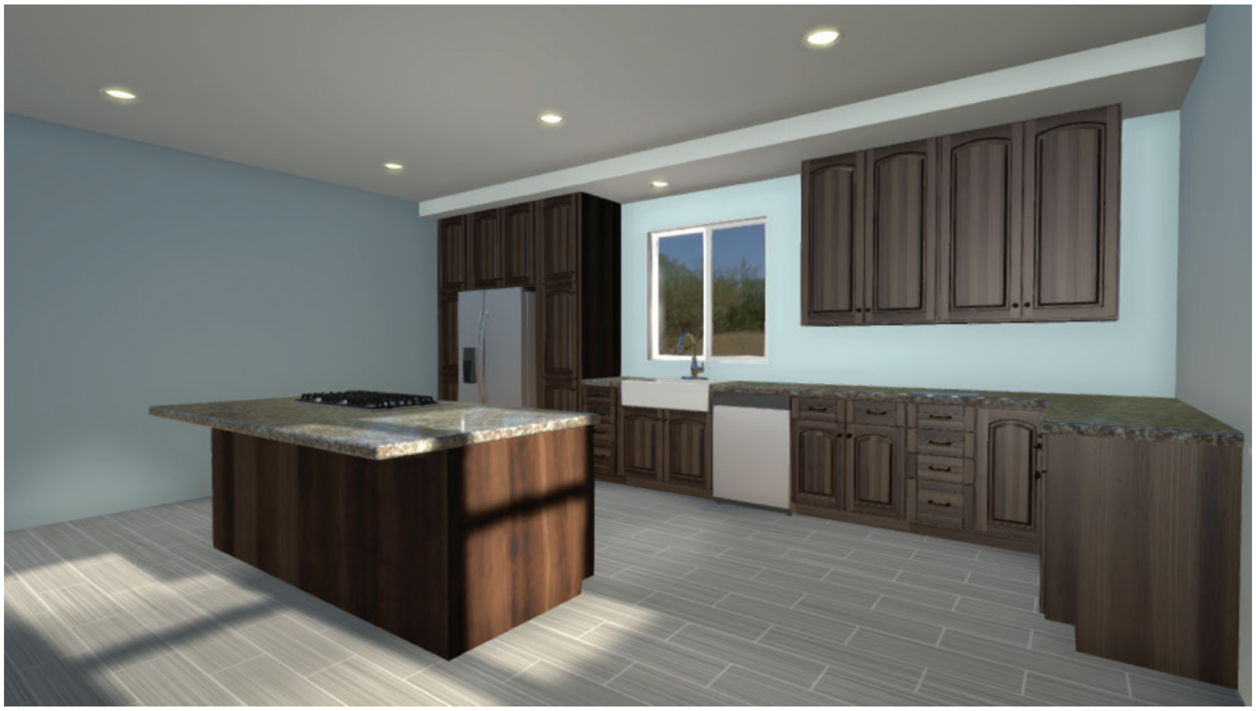
Countertops will be only a single mesh and do not need to be grouped with any other objects besides the parent object.

If there is more than one base object, including end caps, these objects should be grouped together.

Appliances should not be given their own subgroup.

Lighting

This application should use linear color space. The scene should be brightly light and inviting, with sunlight streaming through the larger of the two windows and shining in the center of the room where the kitchen island begins by default. The overhead lighting should be on and spilling light onto the kitchen floor.



The objects in the scene can be moved in the final application, so none should be set to static, except the kitchen itself. Because the objects should still receive lighting information, light probes should be applied.

A post-processing effect should be added to the scene, primarily to add Ambient Occlusion and some light bloom to the ceiling lights. As a final touch, a subtle Vignette effect should be added as well.

User Interface

The User Interface should appear in a fixed position, overlaying the middle of the right side of the screen. The UI should remain a consistent size regardless of screen aspect ratio. There should be a very slight transparency to the background image, but not so much to become distracting.

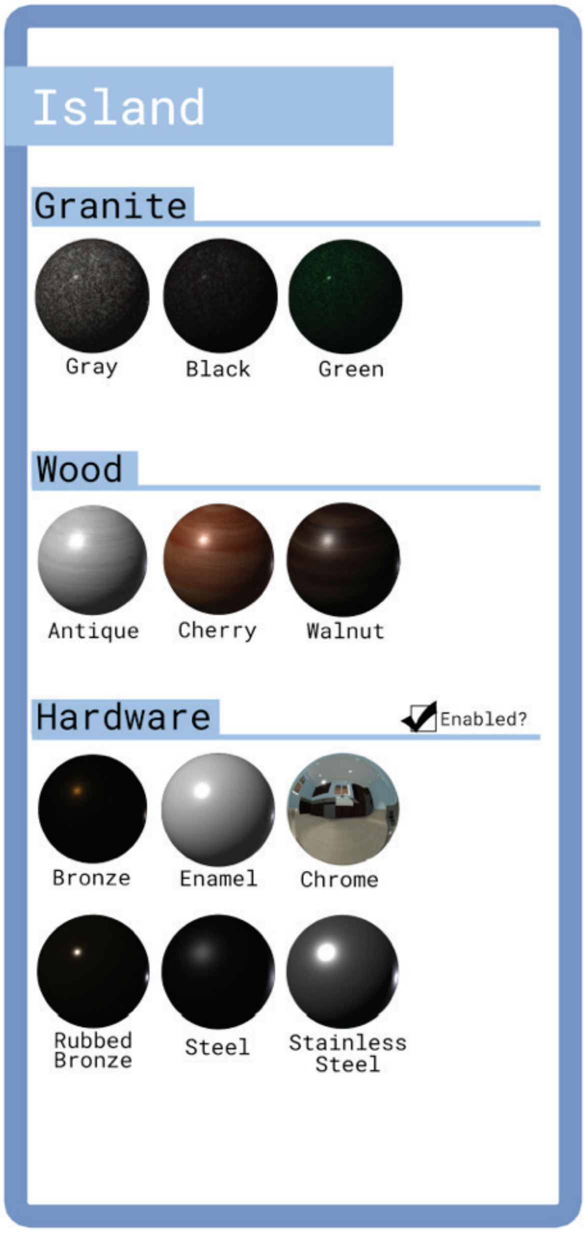
There are three dedicated sections: Granite, Wood, and Hardware, along with a header that identifies the object selected in the scene (this will be handled via script). In addition to being able to change hardware material, users have to option to disable it on the object.

The UI uses the RobotoMonoRegular font. To maintain a high level of text clarity, use TextMesh Pro.

Each material swatch should have its associated label directly below it. In the case of long names, two rows of text is acceptable. The material swatches should be consistently scaled across the UI and should have equal spacing between them.

Each material swatch should also be a button, with the OnClick() set to the UIManager SwapMat() method. The material associated with the swatch should be plugged into the parameter. Finally, each material swatch must also be tagged as either **Granite**, **Wood**, or **Metal**.

The background sprite should be 9-sliced so it can scale as needed to create the UI.



UI Scripts

There are a total of five scripts used for the UI.

1. CameraController: This script must be applied to the main camera. This allows the user to move around in the scene and manages the selection of the kitchen objects. The user can then right-click the mouse to rotate the camera and use the middle mouse button to move in the scene.

2. ObjectData: This script must be applied to all Prefabs in the scene. On awake, it automatically gathers data about the object types in the Prefabs and feeds this information to the UI when called.

3. UIManager: This script must be applied to the UI itself. It has one public variable for the object header, which will indicate the name of any object selected in the scene. This script must also be applied to all of the material swatch button OnClick() methods, calling on the SwapMat() method. This method takes a Material variable, and the material added to this parameter should match the swatch (e.g., the bronze swatch should have the bronze material applied).

The last two scripts are only used for the XR Prototype:

4. Locator: This script must be applied to an empty GameObject and added to the Prefabs in the position where you would like the WorldSpace UI to appear.

5. UI Placement: This script must be applied to the WorldSpace UI and works in conjunction with the UIManager. It has a public variable for the UIManager to be plugged into.

When working in XR design, uncomment line 28 in ObjectData.cs