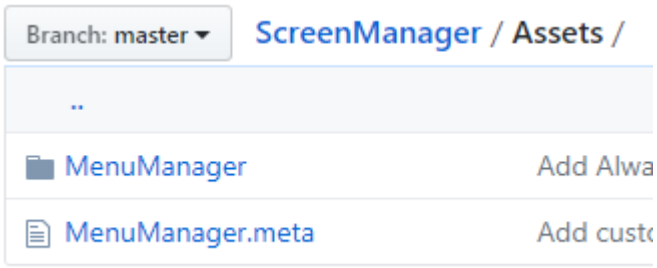
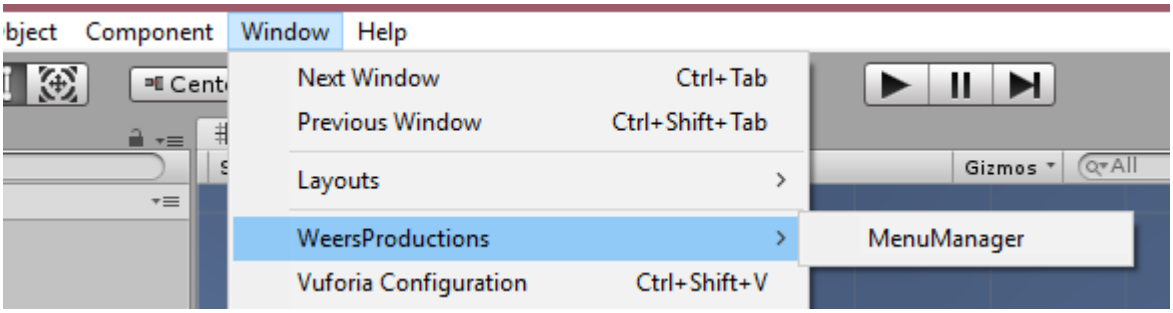
Content

Installation

1. Clone the git repository
2. Copy the ScreenManager/Assets/MenuManager folder to your current project

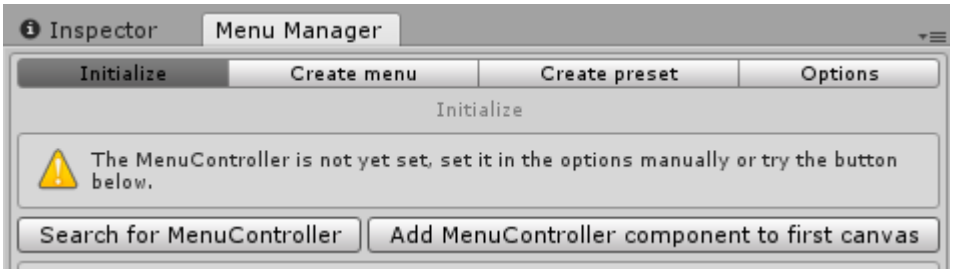


3. In your Unity project, a new menu-item should be added to the 'Window' dropdown menu.

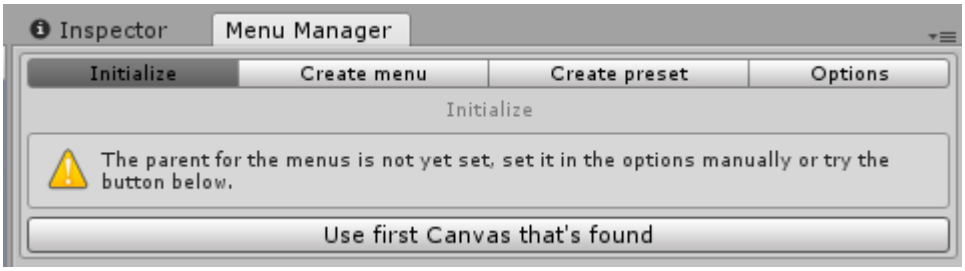


Set-up

1. Open the MenuManager window (Window→WeersProductions→MenuManager)
2. Open the 'Initialize tab'
3. Since the MenuManager doesn't know how to communicate with the current scene that is open, we need to tell it where to find the MenuController component. This is a component that should be attached to the Canvas that will be used for the menus.



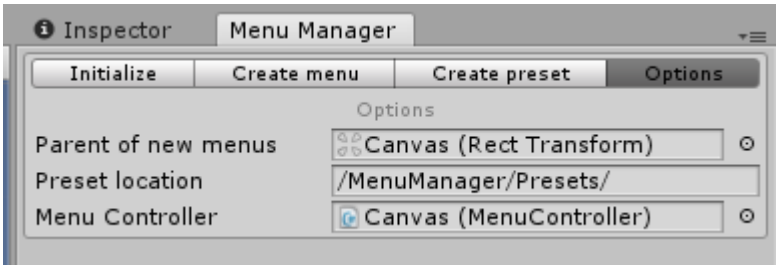
4. If you have a MenuController component on your canvas already, just click 'search for MenuController'. If you don't have one, just let the tool add it and click 'Add MenuController component to first canvas'.
5. The warning message about the MenuController should disappear.
6. A warning message will say that no parent for the menus is set:



7. If you have multiple Canvasses or are doing other fancy stuff, set the Parent of the menus manually in the ‘Options’ tab. Most users will not have this and can simply click ‘Use first canvas that’s found’.
8. The error message should disappear.
9. Before creating any menus, it might be smart to quickly go to the options tab and check the settings.

Options

1. Go to the ‘Options’ tab



Parent of new menus

Whenever menus are spawned and instantiated, this RectTransform will be used as the parent. This way you can organize your scene.

Only if you know what you’re doing you should set this manually, most of the times the ‘Initialize’ tab will fix any problems.

Preset location

This is the path to the preset files. Most of the times the default value should be fine.

Menu Controller

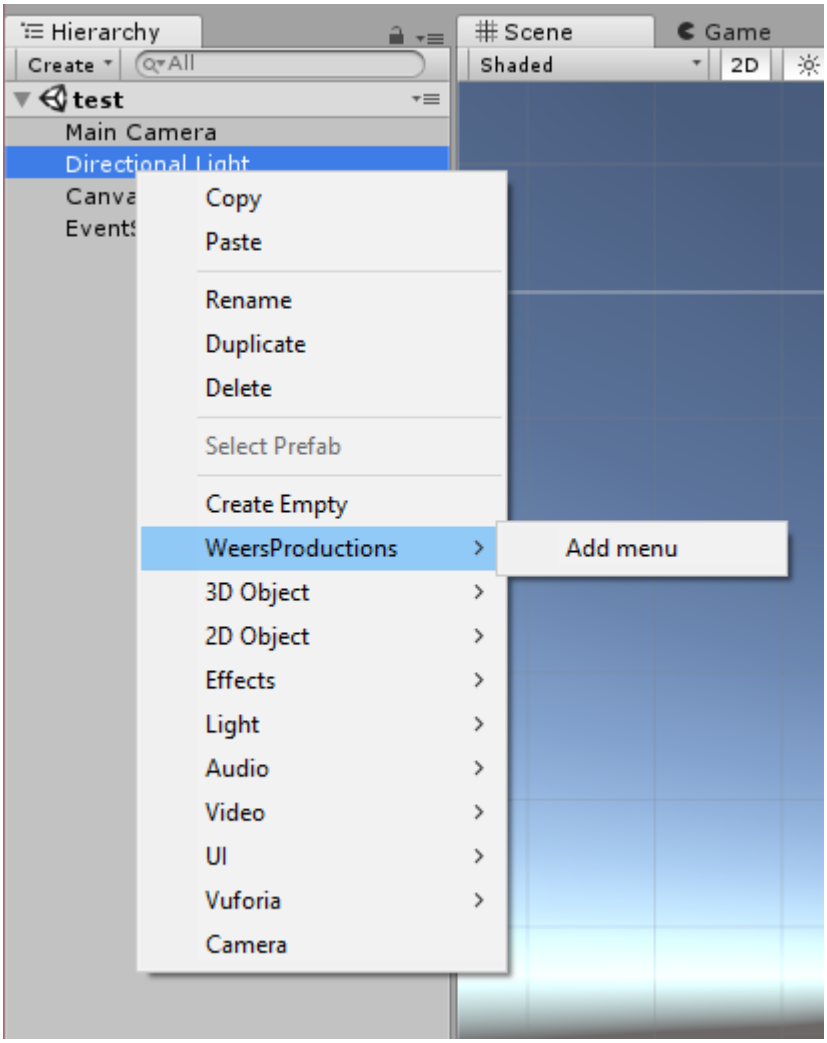
The core of the system, a reference to the MenuController component on your canvas in the scene.
You can just set this with the quick fix buttons in the ‘Initialize’ tab.

Creating a new menu

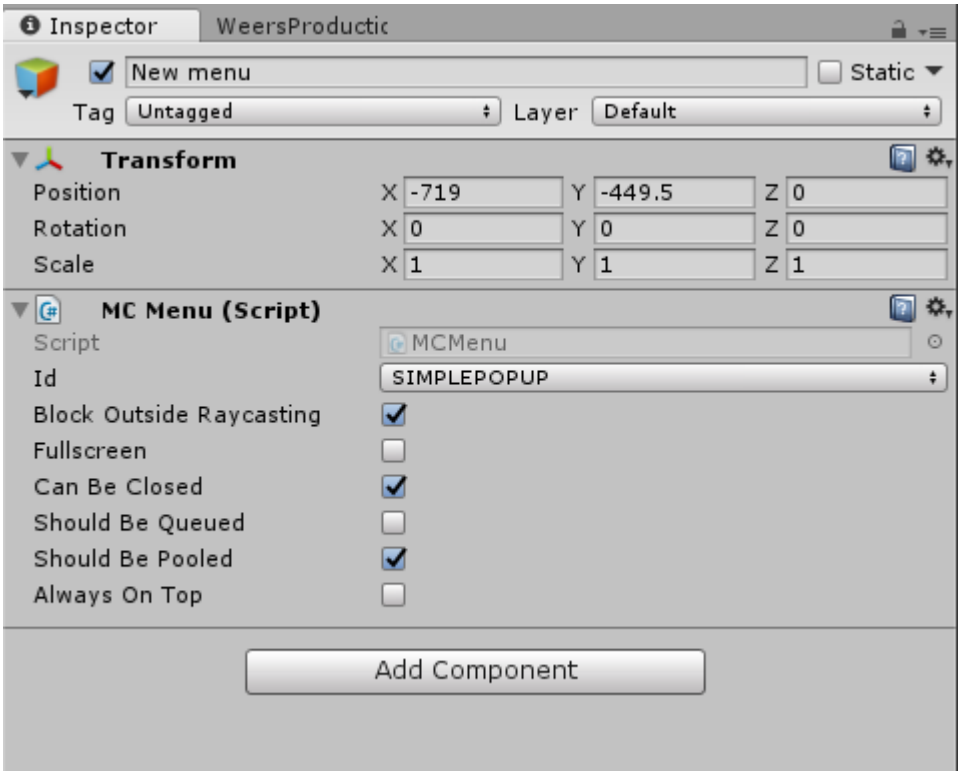
Creating a new menu, this is just the visual part. Go to ‘Adding custom logic’ steps for the logic.

Not using a preset

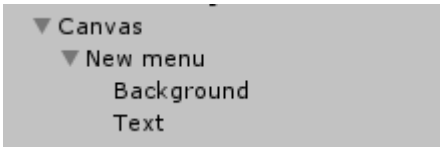
1. Right mouse click in the Hierarchy. It will create a ‘new menu’ GameObject with a MCMenu component.



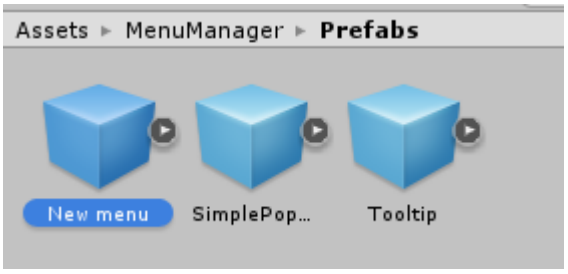
2. Clicking the ‘new menu’ object will show the MCMenu component. Set the properties for the menu.



3. Design your menu as child objects of the ‘new menu’ object.

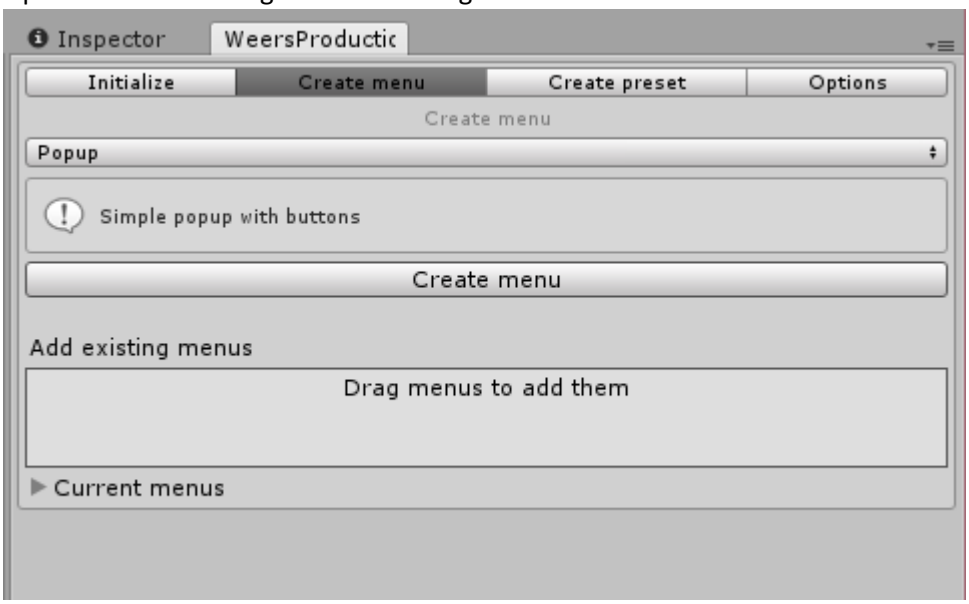


4. Create a prefab of the ‘new menu’. Simply drag the gameObject to a folder in your assets. The specific folder under ‘assets’ is not important (I just like to organize stuff :D).



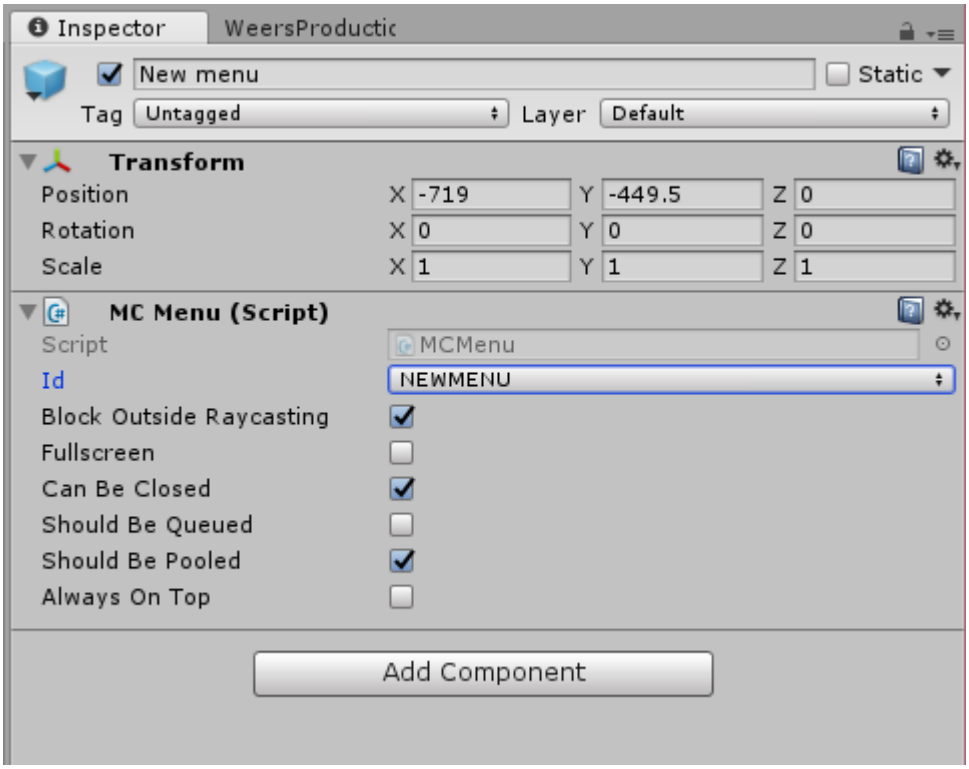
5. Delete the ‘new menu’ child gameobject in the scene. In our case this means the ‘Canvas’ object should not have any childs anymore.

6. Open the MenuManager window and go to the ‘create menu’ tab.



7. Drag your prefab from the ‘project’ window onto the ‘Drag menus to add them’ field.
8. The new menu is added to the ‘MenuController’ of the scene.
9. To show the menu you can now call:
```
MenuController.ShowMenu(MenuController.Menus.SIMPLEPOPUP);
```
10. Since your new menu is probably not a ‘SIMPLEPOPUP’, we will change that now. Open the ‘MenuController’ script.
11. You will notice an enum called ‘Menus’.
```
public class MenuController : MonoBehaviour
{
    /// <summary>
    /// Static references to the ids of all the menus and popups. Makes it easy to change ids.
    /// </summary>
    public enum Menus
    {
        UNDEFINED = -2,
        NONE = -1,
        SIMPLEPOPUP = 0,
        SIMPLETOOLTIP = 1
    }
```
12. Simply add a new entry in the enum with your own name, like this:
```
public class MenuController : MonoBehaviour
{
    /// <summary>
    /// Static references to the ids of all the menus and popups. Makes it easy to change ids.
    /// </summary>
    public enum Menus
    {
        UNDEFINED = -2,
        NONE = -1,
        SIMPLEPOPUP = 0,
        SIMPLETOOLTIP = 1,
        NEWMENU = 2
    }
```

13. Don't forget to set the 'Id' field of the 'MCMenu' component of your new menu in the prefab.

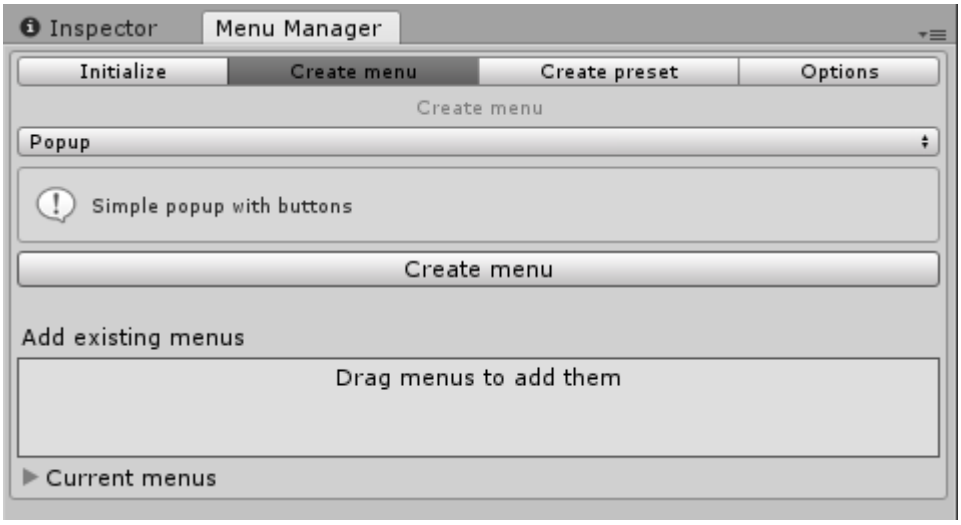


14. Now you can call:
```
MenuController.ShowMenu(MenuController.Menus.NEWMENU);
```

Using a preset

Assuming you already made a preset, if not, go to the 'Creating a new preset' steps.

1. Open the MenuManager window and open the 'create menu' tab

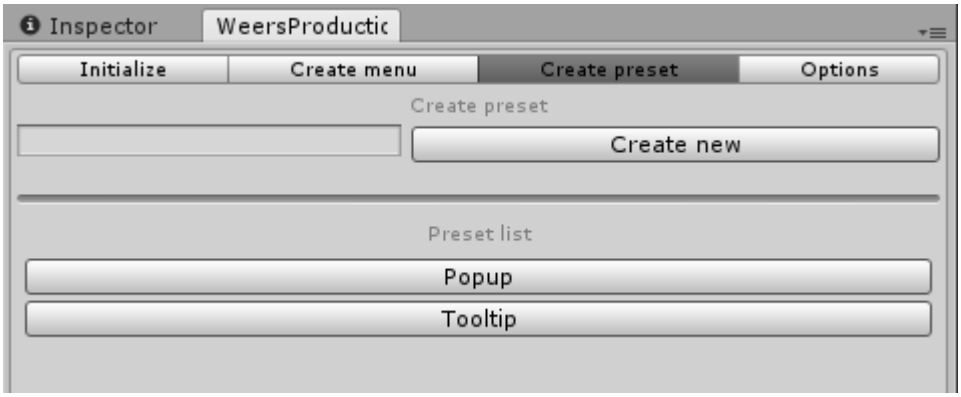


2. Select the preset you want to use, in this case I am using the 'Popup' preset.
3. Press 'Create menu'. It will instantiate a new menu.
4. Edit this menu how you like.
5. Continue from step 4 of the 'Creating a new menu' without using a preset.

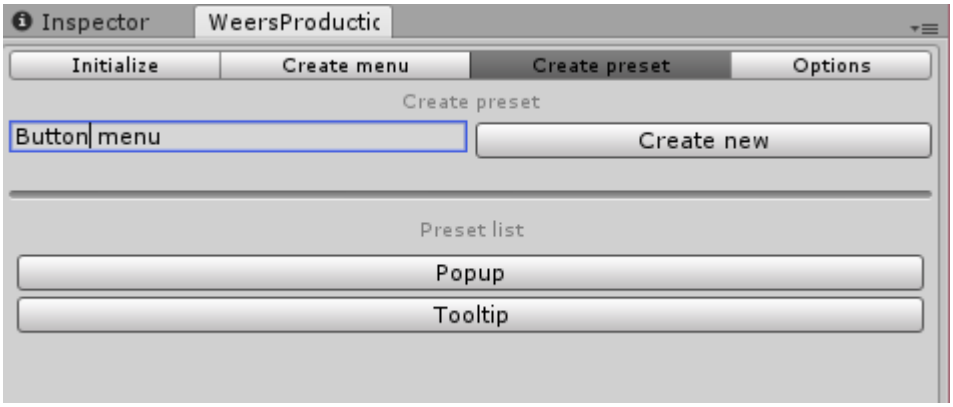
Creating a new preset

Since you might create quite a few menus that look very similar, presets can help speed up the production. Create one basic menu and every time you need to create a new menu use it as a starting point, instead of starting from scratch.

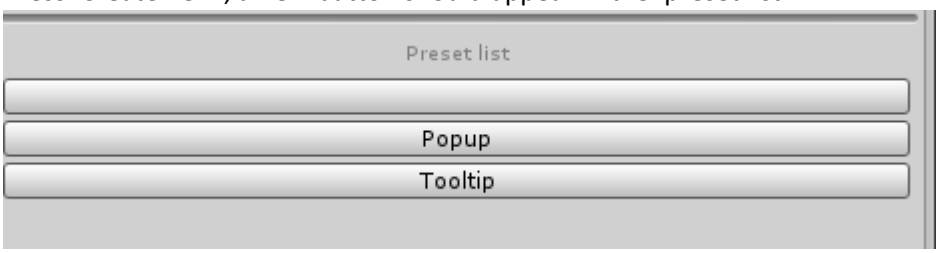
1. Open the MenuManager window and go to the ‘Create preset’ tab.



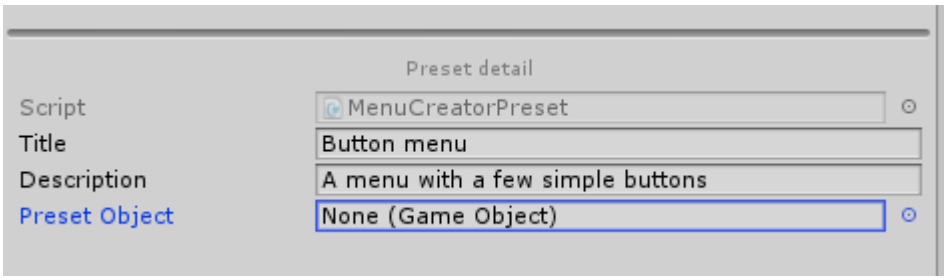
2. Enter a name for your preset, for example: button menu



3. Press ‘Create new’, a new button should appear in the ‘preset list’.



4. Click on it, a ‘preset detail’ focus should appear. Enter a title and optionally a description.



5. If you already made a basic UGUI object, make it a prefab and assign it to the ‘preset object’ property. Make sure the preset object contains a MCMenu (or inheritor of MCMenu) attached. Every time you want to create a new menu, it will instantiate this menu.

Adding custom logic

Instead of using the default 'MCMenu' component, you can customize and add more logic to menus when you create your own component and inherit 'MCMenu'.

```
public class MCSimpleTooltip : MCMenu
```

The rest of the steps to create a new menu are all the same. Just replace every 'MCMenu' with 'MCSimpleToolTip' (in this case).

You can override various functions to customize (to add animations, like fade in or out, for example).

The two most important functions are:

```
/// <summary>
/// Called when this menu should be shown, can be used for animations.
/// </summary>
/// <param name="data">Can be used to send extra data to the menu.</param>
public virtual void Show(object data)
```

```
/// <summary>
/// Called when the Menu is being hidden.
/// Only this menu will be hidden here, no popups or child menus are called in here.
/// </summary>
/// <param name="afterHidden">Should be called when the menu is actually hidden.
/// This means that when you have fade out animations, you should call this when the fade out is complete.</param>
protected virtual void OnHide(UnityAction afterHidden)
```

As you can see this is copied from the actual source code, so read the source code comments to see what else you can do!