**Example: Adding a Boot Screen**

This document contains instructions for a hypothetical case where a new screen may be created for a user to choose between the Stalk Pushing test and Camera testing modes at device startup.

Note: The code contained in this file is the only instance of the code and is not found any where else. It was written without any testing but should provide an overview of the process and illustrate the steps clearly.

A new, simplified, software flowchart is depicted in Figure 1.

Graphical user interface, application, Teams

Description automatically generated

Figure 1. A simplified flowchart depicting the hypothetical case where a new boot screen is added that prompts the user to select between the software they would like to use.

Step 1: Reference Kivy Screen Management to become familiar with the steps for adding a screen to the device.

Step 2: Find an existing screen with a similar layout to the desired boot screen to be created. In this case, the Exit Screen will work nicely.

Step 3: Duplicate the ExitScreen.py and ExitScreen.kv files and rename them to be BootScreen.py and BootScreen.kv

Step 4: Add or modify the following lines of code to Granusoft/src/main.kv:

Around line 9, after the comments, add the following line of code:

1. #:import BootScreen view.screens.main.BootScreen

This will import the new BootScreen into the kivy screen manager. Another line of code will need to be added around line 48, after the #Root comment:

1. BootScreen:

This will add the BootScreen to the kivy screen manager. Finally, modify line 46 which previously read:

1. current: ‘main\_screen’ # Start with the main screen

The edited line should read as:

1. current: ‘boot\_screen’ # Start with the new boot screen

This change will modify which screen will be shown when the GUI boots up, now it will boot up to the BootScreen that has just been created.

Step 5: Open BootScreen.py and adjust the code as follows:

Adjust the comments in line 2 to an appropriate description of the new screen layout.

Change lines 9 and 11 to read:

1. Builder.load\_file('view/screens/main/BootScreen.kv')
3. class BootScreen(BaseScreen):

These changes will account for the name of our new boot screen and correctly load the appropriate kivy file.

Since the Exit Screen had a few custom python functions that are called when buttons are pressed, which are declared under the newly renamed BootScreen class, we can delete those lines of code. Since our new Boot Screen has no need for custom python functions, we will just need to add the python keyword pass. Line 12 should read as follows:

1. pass

And there should be no additional lines after line 12 and should look like this:

1. """
2. Two buttons to select from: Push Tests and Camera Tests. A user prompt is displayed as a title.
3. """
5. from kivy.lang import Builder
6. import os
7. from view.BaseScreen import BaseScreen
9. Builder.load\_file('view/screens/main/BootScreen.kv')
11. class BootScreen(BaseScreen):
12. pass

BootScreen.py is now finished and the only thing left to do is adjust BootScreen.kv

Step 6: Open BootScreen.kv and modify the code in the following manor:

Change the name of the kivy node and name element to match the new boot screen name. The first two lines should read as:

1. <BootScreen>
2. name: 'boot\_screen'

Be sure that the kivy node (name between ‘<’ & ’>’) matches the python class name and the name in line 2 matches the adjusted text in line 46 of the main.kv file.

Step 7: Continuing in BootScreen.kv:

Now the layout of the screen will be adjusted to have 2 side buttons and a label for the user to know what is wanted. Begin by replacing the text nested under GranuSideArea: (beginning on line 5 and proceeding through line 18) so that there are 4 new lines that read as follows:

1. GranuSideButton:
2. GranuNone:
3. GranuNone:
4. GranuSideButton:

This will replace the 4 buttons that are utilized on the Exit Screen with the elements to be used in the new screen–a button at the top of the screen, 2 empty slots, and another button at the bottom of the screen.

In order to give the new button layout functionality and labels, add a text label and on\_release label under the GranuSideButton widgets. The code will look like this:

1. GranuSideButton:
2. text: 'Stalk\nPushing Test'
3. on\_release: root.move\_to(‘main\_screen’) # Move to the main menu for Push Tests
4. GranuNone:
5. GranuNone:
6. GranuSideButton:
7. text: 'Camera Testing'
8. on\_release: root.move\_to(‘cam\_main\_screen’) # Move to the main menu for Camera Tests

The text label will be displayed in the GUI on the button. And the on\_release label will tell the GUI which screen to navigate to when the button is pressed. (The ‘\n’ text in place of a space on line 6 acts like the return key and is used to help display the text on the button in a more convenient way).

Step 8: Still working in BootScreen.kv:

The last step is to write a prompt for the user. Under GranuContent:, add the following lines:

1. GranuContent:
2. GranuTitle:
3. text: 'Which type of testing will be done today?'

This will create a text field where the user is told their decision is wanted.

All in all, the completed BootScreen.kv file should look like:

1. <BootScreen>
2. name: 'boot\_screen'
3. GranuContainer:
4. GranuSideArea:
5. GranuSideButton:
6. text: 'Stalk\nPushing Test'
7. on\_release: root.move\_to(‘main\_screen’) # Move to the main menu for Push Tests
8. GranuNone:
9. GranuNone:
10. GranuSideButton:
11. text: 'Camera Testing'
12. on\_release: root.move\_to(‘cam\_main\_screen’) # Move to the main menu for Camera Tests
13. GranuContent:
14. GranuTitle:
15. text: 'Which type of testing will be done today?'