

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Artifact Information** | | | | | | | | |
| **Artifact ID** | | **Artifact Title** | | | | | | |
| KSS-001 | | Kivy Screen Summary | | | | | | |
| **Capstone Team** | | | | **Revision** | | **Artifact Date** | | |
| Capstone Team 27 - Granustem | | | | 1.0 | | Apr 13, 2019 | | |
| **Prepared by** | | | | **Checked by** | | | | |
| Ben Alexander | | | |  | | | | |
| **Revision History** | | | | | | | | |
| **Revision #** | **Date** | | **Prepared by** | | **Checked by** | | **Description** | **Approved by** |
| 1.0 | Apr 13, 2018 | | Ben Alexander | |  | | Initial Version | Reese Bastian |

1. Purpose

The purpose of this artifact is to describe how screens work within Kivy and our program, and to describe how to add screens down the road.

2. Documentation

A comprehensive set of documentation for the Kivy Screen Manager can be found with the following link: <https://kivy.org/doc/stable/api-kivy.uix.screenmanager.html>

3. Basic Screen Summary

Screens generally have two parts: a kivy (.kv) layout file and a python (.py) functionality file. Within our program, each screen has kivy file and a python file of the same name (i.e. Settings.kv and Settings.py). While kivy files focus mainly on layout and python files focus mainly on functionality, they are not completely separate. For example, there are ways of adding layout details from the python file, and there are ways to reference variables and functions from the kivy file. To understand this more thoroughly, read the documentation.

Generally, transitioning from one screen to another involves pressing a button that has an on\_release event associated to it within the kivy file. The on\_release event generally indicates what screen the program will move to. Take a look at this line of code from Granusoft/src/view/screens/main/MainScreen.kv:

on\_release:

root.move\_to('settings\_screen')

This line specifies that we will move to the screen whose name is ‘settings\_screen’. Each kivy file must have a ‘name’ attribute written directly below the opening kivy tag, as these ids are how we transition between screens. For example, look at this code from Granusoft/src/view/screens/main/SettingsScreen.kv:

<SettingsScreen>:

name: 'settings\_screen'

This line is what names the SettingsScreen kivy and python files as ‘settings\_screen’. This is why the on\_release event in the MainScreen.kv file actually transitions the program to the SettingsScreen.

4. Screen Manager Summary

The Screen Manager Kivy module is what is used to handle all of the screens of the programs. Therefore, the Screen Manager must be aware of all screens and screen names in order to transition successfully between them all. Notice the following lines of code within GranuSoft/src/main.py:

class GranuScreenManager(ScreenManager):

pass

class MainApp(App):

def build(self):

sm = GranuScreenManager(transition=FadeTransition(duration=0.1))

return sm

These lines of code create a Screen Manager immediately when the app begins. Specifically, we’ve specified the Screen Manager as a GranuScreenManager. This is done in order to specify what a GranuScreenManager is (within the main.kv file) and include all desired screens.

Take a look at main.kv, now. Notice all of the import statements that are pulling in the files for all screens involved in the program. Also notice that below the ‘<GranuScreenManager>:’ line, we’ve specified each of the Screens that exist in the program.

5. Steps to Define New Screens

Adding a new screen is fairly simple, but the steps involved are very specific. Make sure to follow these carefully. Examples of following this process can be found in the two example files: *Example\_Adding the Camera Module* and *Example\_Adding a Boot Screen*.

1. Create new Screen kivy and python files of the same name, just different extensions (i.e. NewScreen.kv and NewScreen.py). It is probably easiest to just copy these files from another Screen already in the program.
2. Ensure that the python file is importing the following libraries:

from kivy.lang import Builder

from view.BaseScreen import BaseScreen

1. Make sure that the name of the python file class is equal to the name of the python file, passing in BaseScreen as a parameter (i.e. class NewScreen(BaseScreen)).
2. Make sure there is a line outside of the class declaration that looks like this:

Builder.load\_file('view/screens/main/NewScreen.kv')

Make sure that the file path that is passed is the correct file path to the new Screen’s kivy file.

1. The outermost node in the kivy file must use the same name as the python class name (i.e. <NewScreen>).
2. Just below the line described above, in step #5, make sure to include the name attribute, with a name that you will use to access the Screen from other Screen kivy files.
3. Within the Granusoft/src/main.kv file, import from the new Screen files, as follows:

#:import NewScreen view.screens.main.NewScreen

Make sure that the file path correctly leads to the python and kivy files for the new Screen

1. Add the new Screen into the GranuScreenManager kivy node (i.e. NewScreen:)

These steps should accurately create new screens and add them to the Screen Manager. As you follow these steps, make sure to have all of the related files opened, as it will be easier to follow these steps when looking at the files.

Also note that, when errors occur when trying to create new screens, you may hit errors that are not as descriptive as you’d like. Often, if a new Screen fails to be added to the Screen Manager for whatever reason (incorrect libraries, syntax errors, etc.), the bash will just say that the Screen in question doesn’t exist. Keep in mind that this may mean that you didn’t correctly create the Screen and add it to the Screen Manager, but it may also simply mean that there are errors within the Screen files themselves that you need to go resolve. For this reason, it is beneficial to have test functions within each Screen python file in order to ensure everything is correct within it.