Different Steps to Create a Framework using Component:-

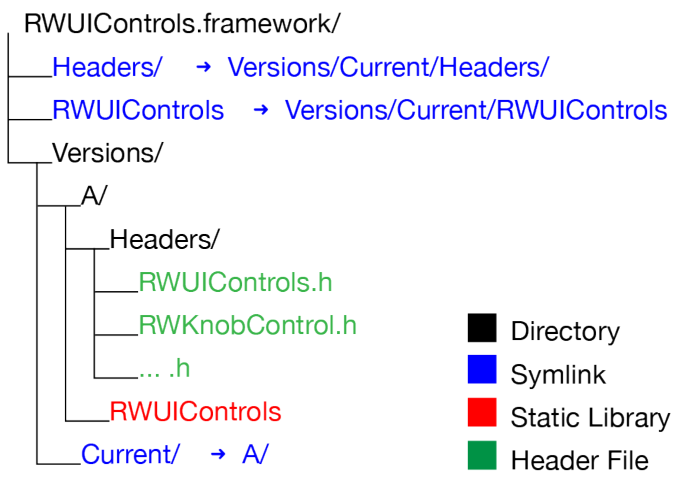
Step 1- Create a static library (Component)

Step 2- Add **Copy Headers** in build phases (This will collect the public header files and put them into the framework.)

Step 3- Configuring Build Settings,

Dead Code Stripping – Set this to NO
Strip Debug Symbols During Copy – Set this to NO for all configurations
Strip Style – Set this to Non-Global Symbols

Step 4- Framework Structure,



Note:- RWUIControls => CameraComponent

Step 5- Choose the Build Phases tab and add a new script by selecting **Editor/Add Build Phase/Add Run Script Build Phase,** Rename the script by double clicking on the panel title Run Script and replace it with Build Framework.

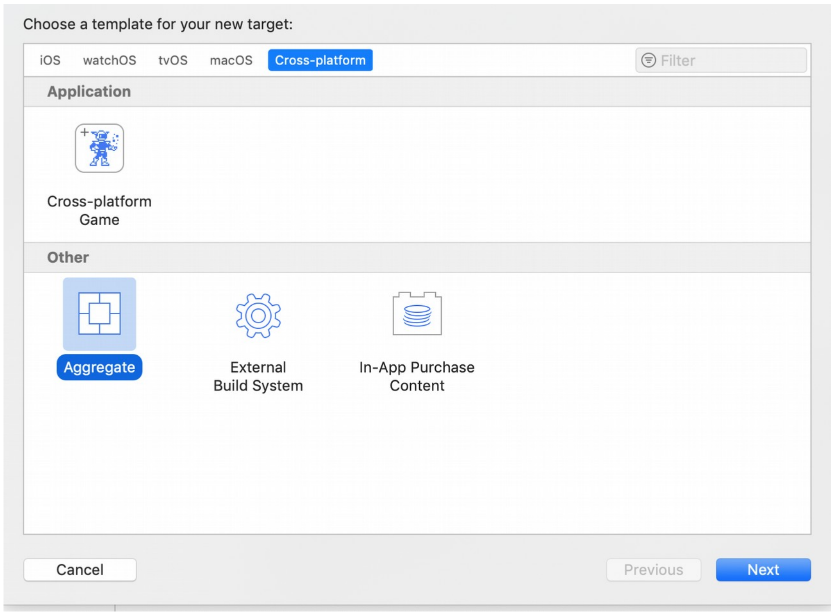
Paste the following Bash script into the script field:

```
set -e

export FRAMEWORK_LOCN="${BUILT_PRODUCTS_DIR}/${PRODUCT_NAME}.framework"

# Create the path to the real Headers die
mkdir -p "${FRAMEWORK_LOCN}/Versions/A/Headers"

# Create the required symlinks
/bin/ln -sfh A "${FRAMEWORK_LOCN}/Versions/Current"
/bin/ln -sfh Versions/Current/Headers "${FRAMEWORK_LOCN}/Headers"
/bin/ln -sfh "Versions/Current/${PRODUCT_NAME}" \
        "${FRAMEWORK_LOCN}/${PRODUCT_NAME}"

# Copy the public headers into the framework
/bin/cp -a "${TARGET_BUILD_DIR}/${PUBLIC_HEADERS_FOLDER_PATH}/" \
"${FRAMEWORK_LOCN}/Versions/A/Headers"
```

This script first creates **CameraComponent.framework/Versions/A/Headers** directory before then creating the three symbolic links required for a framework:

Versions/Current => A
Headers => Versions/Current/Headers
CameraComponent => Versions/Current/ CameraComponent

Step 6- The framework will be created using a new target in the CameraComponent project. To create it, select the CameraComponent project in the Project Navigator and then click the Add Target button shown below the existing targets.



Click Next and name the target Framework.

Step 7- To ensure the static library builds whenever this new framework target is created, you need to add a dependency on the static library target. Select the **Framework** target in the library project and add a **dependency** in the Build Phases tab. Expand the Target Dependencies panel, click the + and choose the **CameraComponent** static library.

Step 8- Create a new Run Script build phase by selecting the Build Phases tab of the **Framework target**, and clicking **Editor/Add Build Phase/Add Run Script Build Phase,** Change the name of the script by double clicking on Run Script. This time name it **MultiPlatform Build.**

Paste the following Bash script into the script text box:

```
set -e

# If we're already inside this script then die
```

```
if [ -n "$RW_MULTIPLATFORM_BUILD_IN_PROGRESS" ]; then
  exit 0
fi
export RW_MULTIPLATFORM_BUILD_IN_PROGRESS=1

RW_FRAMEWORK_NAME=${PROJECT_NAME}
RW_INPUT_STATIC_LIB="lib${PROJECT_NAME}.a"
RW_FRAMEWORK_LOCATION="${BUILT_PRODUCTS_DIR}/${RW_FRAMEWORK_NAME}.framework"

function build_static_library {
    # Will rebuild the static library as specified
    #     build_static_library sdk
    xcrun xcodebuild -project "${PROJECT_FILE_PATH}" \
                 -target "${TARGET_NAME}" \
                 -configuration "${CONFIGURATION}" \
                 -sdk "${1}" \
                 ONLY_ACTIVE_ARCH=NO \
                 BUILD_DIR="${BUILD_DIR}" \
                 OBJROOT="${OBJROOT}/DependentBuilds" \
                 BUILD_ROOT="${BUILD_ROOT}" \
                 SYMROOT="${SYMROOT}" $ACTION
}

function make_fat_library {
    # Will smash 2 static libs together
    #     make_fat_library in1 in2 out
    xcrun lipo -create "${1}" "${2}" -output "${3}"
}

# 1 - Extract the platform (iphoneos/iphonesimulator) from the SDK name
if [[ "$SDK_NAME" =~ ([A-Za-z]+) ]]; then
  RW_SDK_PLATFORM=${BASH_REMATCH[1]}
else
  echo "Could not find platform name from SDK_NAME: $SDK_NAME"
  exit 1
fi

# 2 - Extract the version from the SDK
if [[ "$SDK_NAME" =~ ([0-9]+.*$) ]]; then
  RW_SDK_VERSION=${BASH_REMATCH[1]}
else
  echo "Could not find sdk version from SDK_NAME: $SDK_NAME"
  exit 1
fi

# 3 - Determine the other platform
if [ "$RW_SDK_PLATFORM" == "iphoneos" ]; then
  RW_OTHER_PLATFORM=iphonesimulator
else
  RW_OTHER_PLATFORM=iphoneos
fi

# 4 - Find the build directory
if [[ "$BUILT_PRODUCTS_DIR" =~ (.*)$RW_SDK_PLATFORM$ ]]; then
  RW_OTHER_BUILT_PRODUCTS_DIR="${BASH_REMATCH[1]}${RW_OTHER_PLATFORM}"
else
  echo "Could not find other platform build directory."
  exit 1
fi
```

```
# Build the other platform.
build_static_library "${RW_OTHER_PLATFORM}${RW_SDK_VERSION}"

# If we're currently building for iphonesimulator, then need to rebuild
#   to ensure that we get both i386 and x86_64
if [ "$RW_SDK_PLATFORM" == "iphonesimulator" ]; then
    build_static_library "${SDK_NAME}"
fi

# Join the 2 static libs into 1 and push into the .framework
make_fat_library "${BUILT_PRODUCTS_DIR}/${RW_INPUT_STATIC_LIB}" \
                 "${RW_OTHER_BUILT_PRODUCTS_DIR}/${RW_INPUT_STATIC_LIB}" \
                 "${RW_FRAMEWORK_LOCATION}/Versions/A/${RW_FRAMEWORK_NAME}"



# Ensure that the framework is present in both platform's build directories
cp -a "${RW_FRAMEWORK_LOCATION}/Versions/A/${RW_FRAMEWORK_NAME}" \
     "${RW_OTHER_BUILT_PRODUCTS_DIR}/${RW_FRAMEWORK_NAME}.framework/Versions/A/$
{RW_FRAMEWORK_NAME}"

# Copy the framework to the user's desktop
ditto "${RW_FRAMEWORK_LOCATION}" "${HOME}/Desktop/${RW_FRAMEWORK_NAME}.framework"


# Copy the resources bundle to the user's desktop
ditto "${BUILT_PRODUCTS_DIR}/${RW_FRAMEWORK_NAME}.bundle" \
     "${HOME}/Desktop/${RW_FRAMEWORK_NAME}.bundle"
```

Step 9- Select the Framework aggregate scheme, and press cmd+B to build the framework, This will build and place a **CameraComponent.framework** on desktop.