Большой урок. AppImage, 3D Models.

Данная инструкция предназначена специально для **Linux**.
В **Windows-е** может работать некорректно, криво или не работать вообще. Уже проверено. Не делайте этих ошибок.

Для работы нам понадобятся несколько утилит: **Blender**, **Kicad**, **FreeCad**, **Wings-3D**, **Appimagetool**.

Для корректной работы Blender-а нужен **opengl**.

Для чего же это всё таки надо?

Всё ради того, чтобы получить не просто голые бесцветные или кривые 3D модели для **EasyEda** и **KiCad**, но и раскрасить их в разные цвета. Да чтоб ещё цвета принял сам **Cad**.

В EasyEda используются модели преимущественно формата wrl. В KiCad же можно как step, так и stl и даже wrl.

Однако, не из всех форматов KiCad может видеть цвета.

В некоторых случаях придётся постараться. В идеале если это будет формат **step**. Но, если получится корректно конвертировать в **wrl** - тоже сойдет.

Делаем модель, например, в Blender-е. Чтобы все цвета корректно распознались - необходимо сначала в режиме "**Object Mode**" убрать выборку со всех объектов, а на вкладке "**Material Properties**" насоздовать необходимое количество материалов, в каждом из которых будет свой цвет.

Обратите внимание, что здесь нас интересует ТОЛЬКО цвета, и ничего больше.

Далее выделяем каждый объект, переходим в режим "**Edit Mode**" клавишей "Tab". Для быстрого выделения рекомендую сразу включить выделение по "Face", щелкнуть по одному из полигонов (не путать с ребром и вершиной), а затем чтобы выделить всё, что связано замкнуто с этим полигоном кнопку "L" без шифта. Для выделения всего объекта сразу кнопку «a». Ну или выделите необходимые области вручную с шифтом и CTRL-ом. Можно выделять инструментом «c»... Вообщем способ выделения существует не один. Затем на той же вкладке "**Material Properties**" добавляем новый материал и рядом чуть ниже "**Browse material to be linked**" выбираем один из созданых, а затем не зубудьте щелкнуть кнопку "**Assign**" чтобы применить метриал только для выделенных граней.

Всё, можем сохранять. Сейчас у нас "*.blend" файл. Экспортируем его в "Collada (Default)(*.dae)". Первый на всякий случай, чтобы не переделывать всю работу по нескольку раз.

Далее заходим во **FreeCad**. Открываем "*.dae)" файл. Теперь смотрим чтобы всё цвета были установлены правильно. Если нет, открываем верстак "Part" и для каждого меша изменяем цвет по своему усмотрению.

Обязательно сохраняем в файл самого Cad-а - **FCStd** и пересохраним в файл "Collada (*.dae)". Далее он пригодится.

Теперь пробуем эскортировать модель для **KiCad**. Либо мы установили плагин **KicadStepUpMod** через
Меню **Инструменты -> Менеджер дополнений**, либо установили его вручную.

А вот теперь самое интересное. Зачем было переупаковывать AppImage?

Если у вас **Debian**-подобная система или **Fedora** или **CentOS** - то просто закиньте клонированный репозиторий (обязательно с папкой **.git**) в папку: /usr/lib/FreeCad/Mod.

Если же у вас Arch-подобная система, то скорее всего библиотеки FreeCad и других Cad-ов могут быть не совместимы. Тогда можно просто скопировать AppImage файл, положить его например в домашнюю директорию, а затем скопировать desktop файл в /usr/share/applications/.

[Desktop Entry]

Name=FreeCad

GenericName=3D modeler

Comment=3D modeling

Keywords=3d;modeling;

Exec=/home/mikl/programs/FreeCad_Linux/FreeCAD_0.19-24291-Linux-Conda_glibc2.12-x86_64.AppImage

Icon=/home/mikl/programs/FreeCad_Linux/freecad-logo.png

Terminal=false

Type=Application

Categories=Graphics;3DGraphics;

MimeType=model/fcstd;

После этого можно конечно также зайти в менеджер дополнений и установить плагин, но при очередном обновлении сия затея просто слетает и плагин приходится устанавливать заново. Именно для этого мы и переупаковали AppImage во избежание косяков с обновлениям как системы, так и самой программы.

Заходим во **FreeCad** в верстак KicadStepUp, выделяем все меши, жмем кнопку с черной микросхемкой "Export 3D model to KiCad" и подтверждаем все действия.

Если лог не выдал ошибок, значит всё прошло успешно и можно проверять модель через редактор посад.мест в KiCad.

Если выдал кучу ошибок и файлов **step** и **wrl** на выходе в папке с моделями нет - будем конвертировать вручную.

В логе может быть что-нибудь вроде "**Mesh has null shape**", но файлы всё равно появятся. Не пугайтесь раньше времени, возможно всё в порядке. Их просто надо проверить через редактор посад.мест KiCad-а.

Обратите внимание, что с масштабом своих моделей всегда придётся немного колдовать. В **EasyEda** только параметр "Scale x" и "Scale Y". В **KiCad** ещё и "Scale Z". Таким образом масштабировать модель можно более точно по сравнению с **EasyEda**.

Еси же не отображаются, отображаются криво, или нет цветов - тогда точно будем конвертировать вручную.

Вот для этого и нужен был формат "**Collada (*.dae)**". Он адекватно сохраняет все цвета, но к сожалению не воспринимается ни **EasyEda**, ни **KiCad-ом**.

А для конвертации будем использовать **Wings 3D**.

Скачиваем Wings-3D по ссылке с оф. реп-я.

Рекомендую именно версию **2.2.7.**

$ chmod +x ./wings-2.2.7-linux.bzip2.run

$./wings-2.2.7-linux.bzip2.run

Получаем директорию куда его распаковало. Desktop файл можете вытащить на рабочий стол. А вот копировать его в /usr/share/applications/ или перетаскивать куда либо директорию с программой **не рекомендую**. Она у вас скорее всего будет просто висеть, как у меня, и вы ничего не сможете сделать, кроме как выгрузить, т.е. убить её.

Запускаем и идем в **File -> Import Collada (.dae) ...**

Всё, можем экспортировать. **File -> Export -> VRML 2.0 (.wrl)**.

Выставляем параметры как на скриншоте, даём адекватное имя и нажимаем "**Save**".

Всё, можно проверять **wrl** модель через редактор посад.мест в KiCad.

Ну а на сегодня это всё. Моё дело маленькое - лишь заинтересовать вас.

С вами как всегда был Shadow.

Подписывайтесь на канал, ставьте лайки, комментируйте.

Всем Добра и Удачи!

Создание «.Appimage»

Для того, чтобы создать .Appimage нужна утилита под названием appimagetool.

Теперь можно переходить к структуре проекта.

Структура проекта обязательно должна выглядетьследующим образом.

Project.AppDir

---AppRun

---icon.png

---.DirIcon

---Project.desktop

---./usr/bin/execute-file

Название вашего проекта обязательно через точку: «.AppDir». Необязательно помещать исполняемый файл непосредственно в «Project.AppDir/usr/bin».

А вот .desktop файл обязательно должен быть в соответствии с названием проекта до «.AppDir».

Содержимое файла «Project.desktop», не забываем про иконку:

[Desktop Entry]

Version=1.0

Name=Project-Name

Exec=./AppRun

Executable file without extension and it's better that it was “AppRun” file, not “execute-file”.

Icon=Project-icon

Image without extension (.png, .jpg,.xpm…).

Type=Application

Categories=Utility;

Terminal=false

StartupNotify=true

NoDisplay=false

Comment=The comment

MimeType=application/x-extension-fcstd;

Содержимое файла «.DirIcon»:

icon.png

Размер иконки должен быть не более 64x64 пикселя.

«AppRun» по сути представляет собой скрипт, который как раз и запускает всё что будет внутри вашего «.Appimage».
Содержимое этого файла:

```
#!/usr/bin/env sh

HERE="$(dirname "$(readlink -f "${0}")")"

EXEC="${HERE}/usr/bin/execute-file"

exec "${EXEC}»
```

Теперь необходимо все файлы проекта сделать исполняемыми:

```
chmod +x ./Project.AppDir/usr/bin/execute-file
./Project.AppDir/AppRun ./Project.AppDir/Project.desktop
```

С помощью нехитрой команды создаем наш долгожданный «.Appimage»:

```
ARCH=x86_64 ./appimagetool-x86_64.AppImage Project.AppDir/
```

В качестве примера вы можете видеть на скриншоте создание такого пакета для одного из проектов на QT Creator, после работы утилиты CQtDeployer для полной переносимости данной программы на любой другой дистрибутив Linux.

Ну а сегодня на этом всё. Надеюсь я вас хоть немного заинтересовал.

Спасибо за внимание. Всем Удачи, до новых встреч, Пока-Пока!