开发者指南

CANONICAL | ubuntu

物联网图形应用与 Ubuntu Frame

Ubuntu Frame

关于Ubuntu Frame的说明

Ubuntu Frame 是嵌入式显示器的实现基础。它能帮助开发者将应用程序可靠、安全、便捷地部署到信息亭式、物联网设备或数字标牌解决方案之中。使用Ubuntu Frame，开发者选择或开发的图形应用程序便可以获得一个全屏窗口、一套专用的窗口行为、实现触动、键盘和鼠标输入，无需再借助其他特定的硬件、屏幕键盘等等。

如果配合Ubuntu Core一起使用，Ubuntu Frame可以提供在边缘设备上安全部署并维护图形应用程序所需的所有基础设施。一方面，Ubuntu Core能最大限度提高应用程序的性能和安全性；另一方面，Ubuntu Frame能与所有支持snap格式的Linux操作系统兼容。

本文将向您介绍如何部署支持Wayland协议的图形应用程序，以便同时使用Ubuntu Frame和Ubuntu Core。本指南可帮助开发人员创建信息亭、数字标牌解决方案、信息娱乐系统、物联网设备或任何其他需要在屏幕上展示图形界面的应用程序。

本指南包括以下内容：

1. 如何设置打包和部署应用程序所需的桌面工具及桌面环境
2. 如何测试您的电脑是否可以同时运行应用程序和Ubuntu Frame
3. 常见问题解答
4. 将应用程序打包为snap软件包并测试其是否能在您的电脑上运行
5. 将snap软件包应用于物联网设备，并在该设备上进行测试

如需了解如何安装预置应用程序，如wpe-webkit-mir-kiosk、mir- kiosk-kodi或Scummvm，请参照官方安装说明及配置指南。

注：本指南不涉及如何使用支持Wayland协议的工具包（数量过多）来构建应用程序。本文也不涉及如何打包基于X11的应用程序到Ubuntu Core上运行（尽管这是可能实现的）。此外，本文将不会介绍如何将您的snap软件包上传到snap商店，或使用预配置的snap软件包构建自定义的Ubuntu Core映像。相关内容详见snapcraft.io/docs。

如果您此前未接触过Ubuntu Core，我们建议您阅读《新手指南》关于如何构建自定义的Ubuntu Core映像的信息详见snapcraft文档。

要求

在创建图形应用程序时，开发人员往往会使用不同的工具和流程。为清晰起见，我们在本指南中假设您目前已有一个支持Wayland协议的应用程序，并且您可以在安装Linux系统的电脑上对其进行测试。

您也可以在容器中或在其他电脑上进行测试（前提是snapd工具和X转发功能正常）。但是，目前暂不涉及上述操作。

后续步骤可能需要用到Ubuntu One账号。登录该账号后，您可以在Launchpad账户上启用“远程构建”功能，并在Snap Store上发布图形应用程序。

设置测试环境

Ubuntu Frame为开发人员提供了一个可以在开发环境中模拟最终应用程序外观和响应情况的工具。因此，开发者无需在目标设备上运行应用程序就可以进行首次设计和可用性迭代。

打开一个终端窗口并输入：

```
sudo snap install ubuntu-frame
```

Snapcraft是一个用于构建snap软件包的命令行工具。用户可用Snapcraft创建应用程序或软件包，然后将其发布到Snap Store。

在同一个终端窗口输入：

```
sudo snap install snapcraft --classic
```

若尚未安装Git，正好现在安装（在Ubuntu上，使用命令“**sudo apt install git**”即可）。

检查应用程序是否能与Ubuntu Frame同时运行

应用程序与Ubuntu Frame同时运行以及在snap软件包中运行时，可能出现问题。为避免混淆，我们建议您在将应用程序打包成snap软件包之前，先用Ubuntu Frame测试一下。在这个环节，您应该测试应用程序，看看是否会出现常见问题，并学习如何解决这些问题。

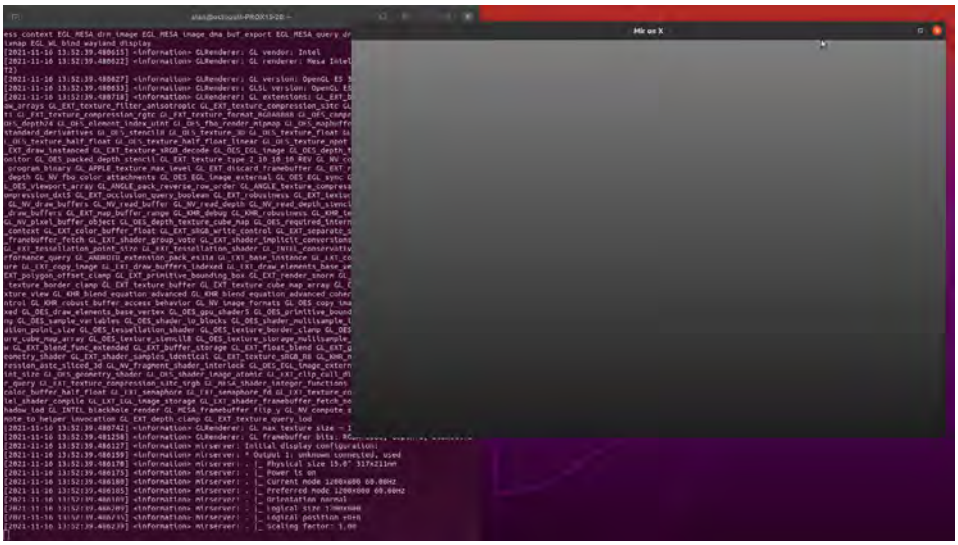
激活Ubuntu Frame模拟器

在将应用程序连接到Ubuntu Frame时，有一点非常关键，那就是这种连接是由“**WAYLAND_DISPLAY**”环境变量控制的。你需要将应用程序和Ubuntu Frame的环境变量值设置为相同的值，因为您的桌面环境可能使用的是基于Wayland协议（此环境下，默认环境变量为**wayland-0**），因此应将其设置为**wayland-99**。

在终端窗口（本节和下一节你将使用到该窗口）中输入：

```
export WAYLAND_DISPLAY=wayland-99
ubuntu-frame&
```

此时，您会看到一个包含灰色渐变背景的“Mir on X”窗口。Mir是一个库，而Ubuntu Frame是基于Mir的。在窗口中测试应用程序之前，您应该使用“Mir on X”模拟器。



在Ubuntu Frame模拟器上测试应用程序

您可以使用Electron、Flutter、Qt或任何其他工具包或编程语言来开发你的图形应用程序，但目前没有任何途径可检查这些工具包或编程语言。因此，本文中，用于例举的应用程序均由GTK、QT和SDL2创建。

这里，我们以游戏应用程序为例，如Mastermind、Neverputt和Bomber。选择这些应用程序是因为它们易于安装，无需完整的桌面会话即可运行。您也可以使用信息亭应用程序、行业GUI应用程序、智能冰箱GUI应用程序、数字标牌来进行测试。

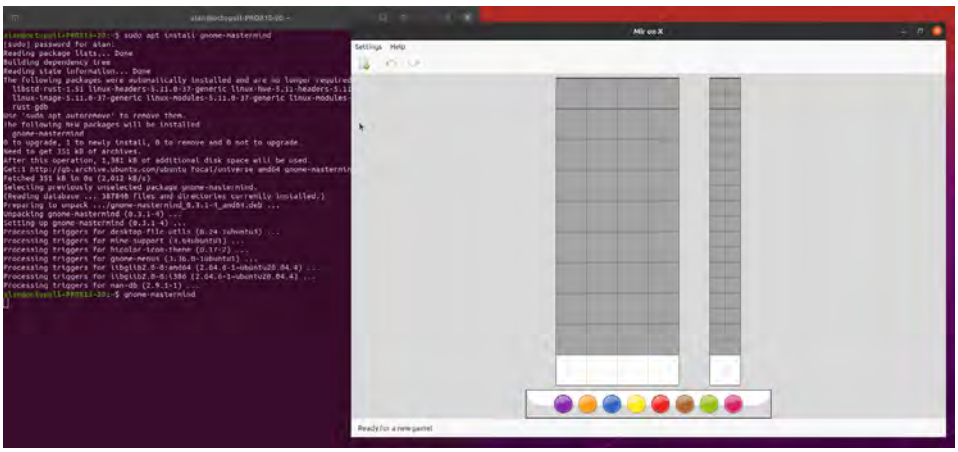
因此，第一步是下载并运行此类应用程序。如果在“Mir on X”中能看到该应用，则说明该应用程序可以在Ubuntu Frame中运行。

接下来，仍在这个终端窗口，输入：

使用GTK创建应用程序的示例： Mastermind

```
sudo apt install gnome-mastermind
gnome-mastermind
```

此时，Ubuntu Frame的“Mir on X”窗口中应显示“Mastermind”游戏应用程序。



如果此时没有显示应用程序或看起来不正确，那么就应该现在进行修改，然后再将其打包成snap软件包。以下是一些可能遇到的问题的示例。

关闭Mastermind（**Ctrl-C**）并尝试下一个示例应用程序：

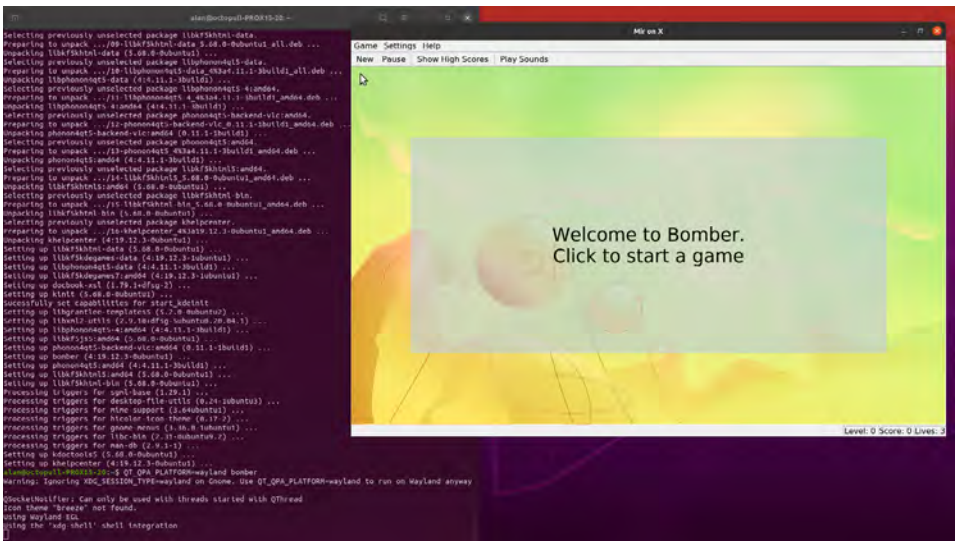
使用QT创建应用程序的示例：Bomber

```
sudo apt install bomber
bomber
```

正常情况下，Bomber应出现在灰色窗口中，而不是在Ubuntu Frame窗口中。问题就在于QT的默认协议不是Wayland协议，因此需要手动设置“**QT_QPA_PLATFORM** ”。关闭窗口，再尝试输入：

```
QT_QPA_PLATFORM=wayland bomber
```

此时，Ubuntu Frame的“Mir on X”窗口中会显示“Bomber”，如下图所示。



稍后，您应在snap recipe的“环境”中合并设置“**QT_QPA_PLATFORM**”。

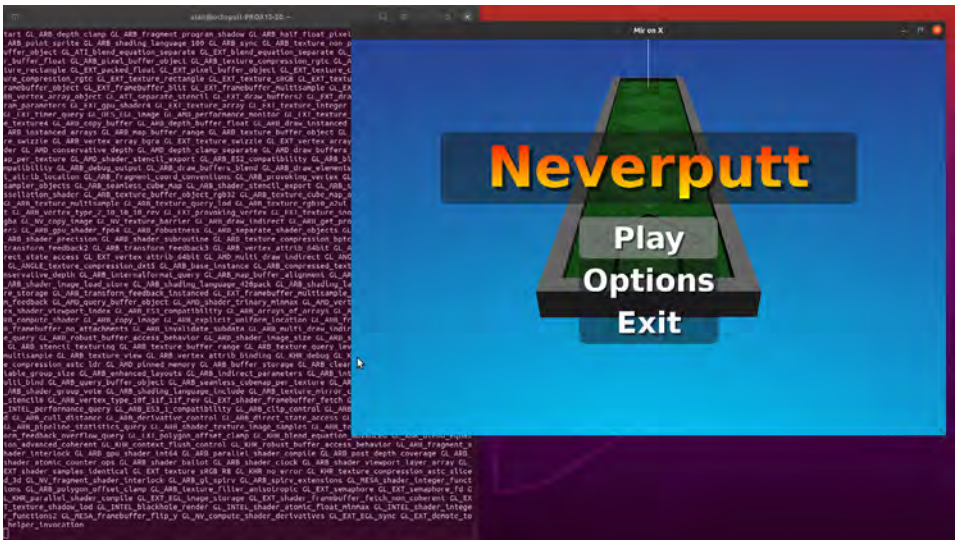
关闭此应用程序（**Ctrl-C**）并尝试下一个示例应用程序。

SDL2 Example: Neverputt

```
sudo apt install neverputt
SDL_VIDEODRIVER=wayland neverputt
```

此时，Ubuntu Frame的“Mir on X”窗口中应显示“Neverputt”游戏应用程序。需要注意的是，SDL2的默认协议不是Wayland，因此需要手动设置“**SDL_VIDEODRIVER**”。

您可能会遇到另一种问题：游戏应用程序在屏幕上显示不完全。这是因为应用程序不理解Ubuntu Frame的全屏显示指令。某些程序会出现这个问题。这时，开发者可以编辑“**~/.neverball/neverballrc** [sic] to say‘fullscreen 1’”，再重启游戏即可修复这个问题。（这个文件也同样适用于Neverball（游戏应用程序），但开发者可能需要先找出正确的配置文件。）现在，您将看到如下结果：



稍后，您应在snap recipe中添加“**SDL_VIDEODRIVER**”设置和“**neverballrc**”文件。现在，可以关闭此应用程序（**Ctrl-C**）。

将应用程序打包成snap软件包

现在，您已经知道如何确认应用程序是否能与Ubuntu Frame兼容，下一步就是使用snap打包工具，将应用程序打包至物联网设备。本节将展示如何将应用程序打包，以及过程中可能出现的问题和解决方法。
用Snap打包物联网图形

要想使用Ubuntu Core，应用程序就必须打包为snap软件包。您也可以利用OTA更新、自动回滚、增量更新、更新语义通道等。如果您不使用Ubuntu Core，而是Linux的其他形式，我们建议您使用snaps，以便利用相关的众多优势。

网上有很多关于在线打包snap软件包的资料，但介绍snapcraft打包工具或者Snap Store并非本指南的目的。因此，我们将重点介绍只针对物联网图形的打包工具。

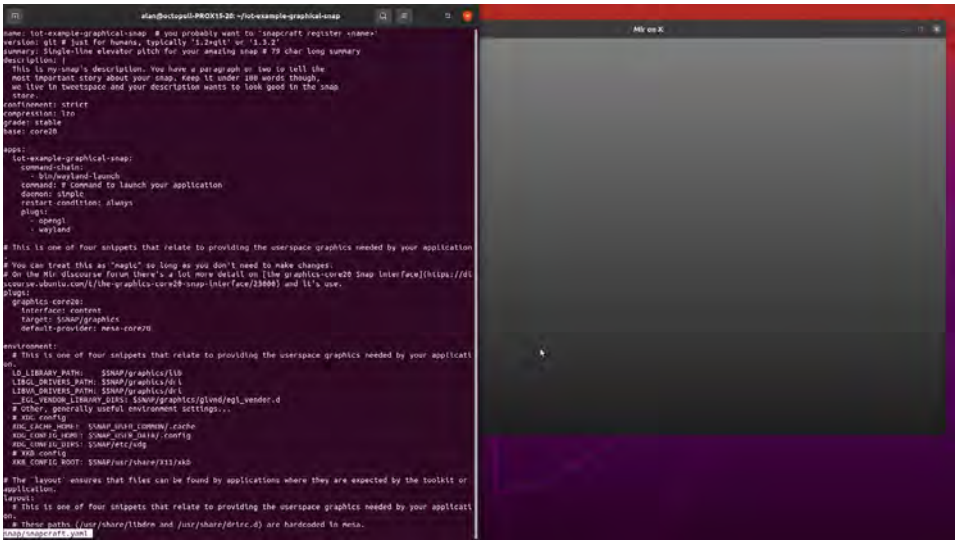
您在网上找到的大部分关于打包GUI应用程序的内容都是仅用于电脑端的打包程序。有些工具可能不适用于物联网设备，因为Ubuntu Core和Ubuntu服务器并不包括桌面安装的所有动作，而且snaps软件包需要作为Daemon程序（后台服务）运行，而不是在用户会话中。尤其需要注意的是，不必使用任何Snapcraft扩展文件，即便是这些扩展文件有助于编写与桌面环境相匹配的snap recipe（例如，使用正确的主题）。

没有扩展文件情况下，编写snap recipe的难度也不大。接下来，我们将对上一节中的各个示例应用程序进行逐一说明。

首先，您应该复制一个包含物联网图形通用Snapcraft recipe的软件仓库。在您已经打开的同一个窗口中（上一节开始时打开的窗口），输入：

```
git clone https://github.com/MirServer/iot-example-graphical-snap.git
cd iot-example-graphical-snap
```

这时候，如果您打开“snap/snapcraft.yaml”文件，可以看到一个物联网图形snap软件包的通用“snapcraft recipe”。就在这个“snapcraft recipe”中插入应用程序的打包说明。.yaml文件的格式如下：



本指南中每个示例（即分别使用GTK、Qt和SDL2创建的应用程序）的特定snapcraft recipe都在下列软件仓库的相应分支中。

```
$ git branch -a
* master
  remotes/origin/GTK3-mastermind
  remotes/origin/HEAD -> origin/master
  remotes/origin/Qt5-bomber
  remotes/origin/SDL2-neverputt
  remotes/origin/master
```

获得特定的snapcraft recipe之后就可以开始将示例应用程序打包为snap软件包了。

使用GTK创建应用程序的示例：Mastermind

切换到用GTK创建示例软件的分支。然后用下列**snapcraft**来构建snap软件包：

```
git checkout GTK3-mastermind
snapcraft
```

Snapcraft是用来制作snap软件包的打包工具。本文不会详细探讨**Snapcraft**的所有可用选项。但是，为避免疑问，特此提醒您注意：首次使用snapcraft时，界面上会弹出以下对话框“需要设置系统以支持‘multipass’，是否立即设置？[y/N]：”，选择“是”即可。

几分钟后，snap软件包就创建好了，界面会显示如下的类似信息：

```
Snapped iot-example-graphical-snap_0+git.7c73b6a_amd64.snap
```

现在，您就可以安装并运行如下snap：

```
sudo snap install --dangerous *.snap
snap run iot-example-graphical-snap
```

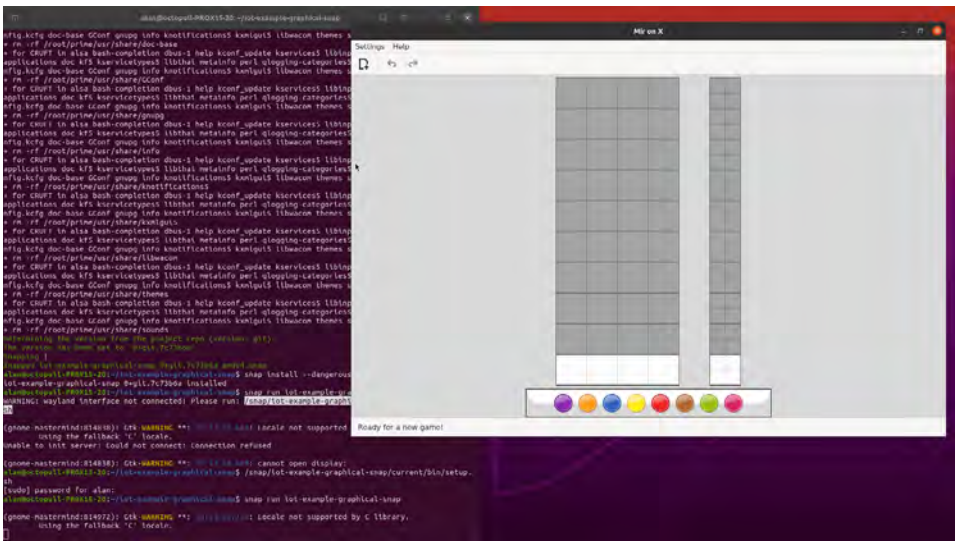
在首次安装了Ubuntu Frame之后运行snap软件包时，系统可能弹出如下警告信息：

```
WARNING: wayland interface not connected! Please run: /snap/iot-example-graphical-
snap/current/bin/setup.sh

(gnome-mastermind:814838): Gtk-WARNING **: 14:13:26.644: Locale not supported by C
library.
    Using the fallback ‘C’ locale.
Unable to init server: Could not connect: Connection refused

(gnome-mastermind:814838): Gtk-WARNING **: 14:13:26.649: cannot open display:
```

首个警告信息是解决问题的关键提示，这个信息是来自通用recipe中的一个脚本。创建snap的过程中（也就是说，上传snap到商店并获得必要的声明之前），连接snap使用的任何“接口”需要手动设置。根据提示信息，如下是一个辅助脚本。运行这个脚本，进行二次尝试：

```
/snap/iot-example-graphical-snap/current/bin/setup.sh
snap run iot-example-graphical-snap
```

此时，Ubuntu Frame的“Mir on X”窗口中应显示“Mastermind”游戏应用程序。



关闭此应用程序（**Ctrl-C**）并尝试下一个示例应用程序。

使用QT创建应用程序的示例：Bomber

为避免混淆，需删除前面的例子中创建的**.snap**文

```
件。 rm *.snap
```

现在，切回到“Qt **first-try**”示例分支，然后创建、安装并运行以下snap软件包：

```
git checkout Qt5-bomber-first-try
snapcraft
sudo snap install --dangerous *.snap
snap run iot-example-graphical-snap
```

如果仔细观察，您可能会发现名为“**Qt5-bomber-first-try**”的分支本不应该出现在此界面上。如下信息解释了你可能遇到问题：

```
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland
to run on Wayland anyway.
QSocketNotifier: Can only be used with threads started with QThread
“Session bus not found\nTo circumvent this problem try the following command
(with Linux and bash)\nexport $(dbus-launch)”
```

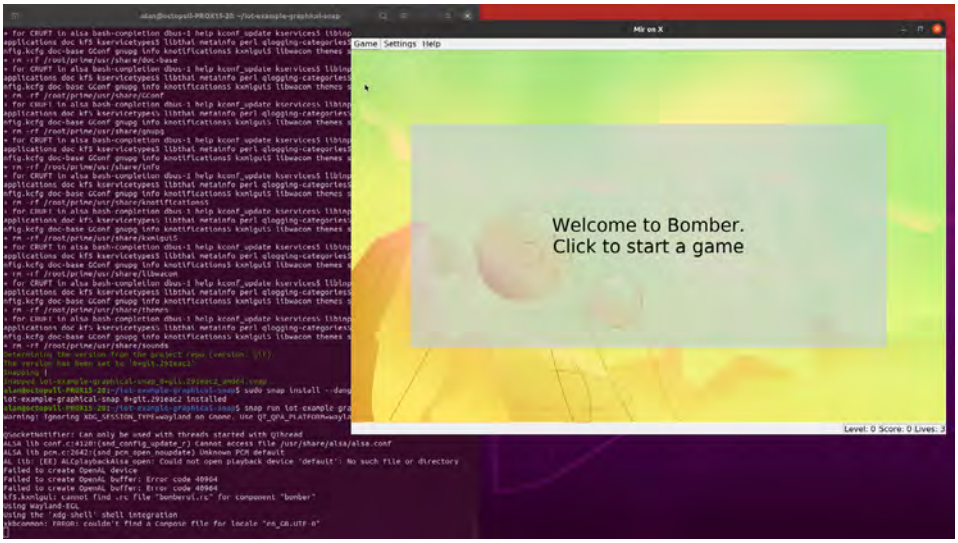
脚本之所以不能正常运行是因为Bomber应用程序需要DBus程序的“会话总线”。为解决这个问题，您可以使用snap软件包里面的“**dbus-run-session**”临时创建一个“会话总线”。比较不同的分支就可以清晰看到此过程。

```
git diff Qt5-bomber-first-try Qt5-bomber
```

现在，切回到Qt示例分支。然后创建、安装并运行以下snap软件包：

```
git checkout Qt5-bomber
snapcraft
sudo snap install --dangerous *.snap
snap run iot-example-graphical-snap
```

此时，Ubuntu Frame的“Mir on X”中应显示“Bomber”游戏应用程序。



关闭此应用程序（**Ctrl-C**）并尝试下一个示例应用程序。

使用SDL2创建应用程序的示例：Neverputt

为避免混淆，需删除前面的例子中创建的**.snap**文件。

```
rm *.snap
```

现在，切换到以SDL2创建应用程序示例软件的分支。然后创建、安装并运行以下snap软件包：

```
git checkout SDL2-neverputt
snapcraft
sudo snap install --dangerous *.snap
snap run iot-example-graphical-snap
```

此时，Ubuntu Frame的“Mir on X”窗口中应显示“Neverputt”游戏应用程序。

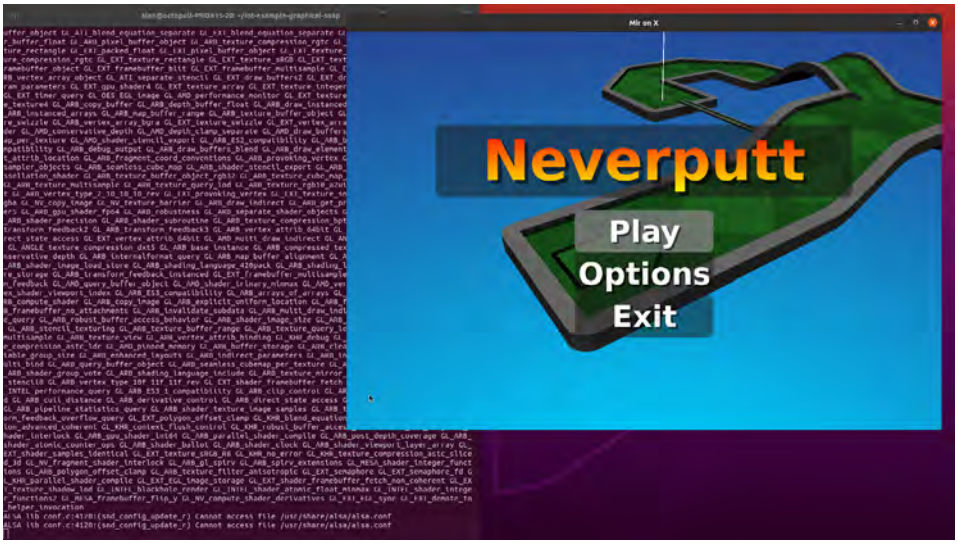
但运行时很可能会有警告信息弹出，这是因为Neverput使用了其他尚未连接的接口：

```
WARNING: hardware-observe interface not connected! Please run: /snap/iot-example-
graphical-snap/current/bin/setup.sh
WARNING: joystick interface not connected! Please run: /snap/iot-example-
graphical-snap/current/bin/setup.sh
ALSA lib conf.c:4120:(snd_config_update_r) Cannot access file /usr/share/alsa/alsa.
conf
Failure to initialize SDL (Could not initialize UDEV)
```

同时，界面上还会出现要求运行设置脚本的警告信息。这是因为Neverputt 的snap软件包需要一些其他接口，而这些接口目前尚未连接。以SDL2创建的其他应用程序也可能会出现同样的情况。此时，您应该确定您的应用程序所需的接口并将它们添加到**snapcraft.yaml** recipe中。

运行设置脚本中尚未连接的接口，然后进行二次尝试：

```
/snap/iot-example-graphical-snap/current/bin/setup.sh
snap run iot-example-graphical-snap
```



关闭此应用程序（**Ctrl-C**），然后关闭Ubuntu Frame窗口。您的应用程序已经成功打包为snap软件包了。

打包您的应用程序

打包应用程序前，还需要解决许多问题，例如，哪些内容需要放在snap软件包中；运行环境需要如何配置；需要哪些接口。

也许，您能从我们给出的示例中得到一些启发。您可以使用“**git diff**”指令来查看每个示例中使用的独特配置指令，如下所示：

```
    git diff master SDL2-neverputt
```

此时，您可以看到本示例中的“**SDL_VIDEODRIVER**”设置和“**neverballrc**”文件。但篇幅限制，我们无法列举出所有可能性，也无法全面介绍每种工具。您还可以在Snapcraft网站上找到一些有用的文档和论坛帖子。

创建并安装用于边缘设备的应用程序

目前为止，我们已经在自己的电脑上测试了应用程序打包为snap软件包后是否能在Ubuntu Frame上运行。尽管Ubuntu Frame加速了应用程序的开发过程，但我们仍然需要考虑应用程序是否能在您的边缘设备上成功运行。许多边缘设备都不支持**amd64**架构，但此架构却是开发机器最为典型的架构。因此，本节将以SDL2创建Neverputt应用程序为例，介绍当设备采用不同架构时，如何创建并安装应用程序要边缘设备上，以及常见问题的解决办法。

利用Snapcraft远程构建工具

如下所示为创建snap软件包的最简单的办法：

```
    snapcraft remote-build
```

这个方法需要用Launchpad构建一个farm，从而逐一构建支持snap软件包的架构。此时，您需要一个Launchpad账户，并且该账户应支持将snap源代码上传到一个公共位置。

构建完成后，您可以使用scp命令，将.snap文件传输至物联网设备上，并使用“**--dangerous**”进行安装。

在本指南中，我们采用《Ubuntu Core：创建支持图形的虚拟机》中介绍方法来设置一个虚拟机。除了用于scp命令和ssh命令的地址外，设置方式和其他设备相同，但使用scp命令和ssh命令更容易截屏。

```
    scp -P 10022 *.snap <username>@localhost:~
    ssh -p 10022  <username>@localhost
    snap install ubuntu-frame
    snap install --dangerous *.snap
```

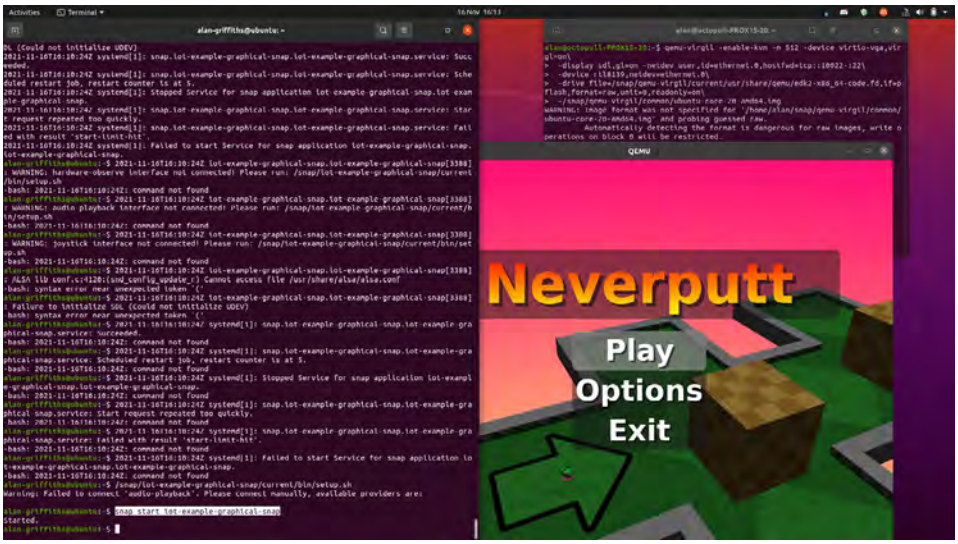
如下所示，（如果您一直按照步骤操作且操作无误）安装完毕后，界面上会显示Ubuntu Frame的灰色窗口，但不显示Neverputt的启动信息：

```
$ snap logs -n 30 iot-example-graphical-snap
...
2021-11-16T16:10:24Z iot-example-graphical-snap.iot-example-graphical-snap[3388]:
WARNING: hardware-observe interface not connected! Please run: /snap/iot-example-
graphical-snap/current/bin/setup.sh
2021-11-16T16:10:24Z iot-example-graphical-snap.iot-example-graphical-snap[3388]:
WARNING: audio-playback interface not connected! Please run: /snap/iot-example-
graphical-snap/current/bin/setup.sh
2021-11-16T16:10:24Z iot-example-graphical-snap.iot-example-graphical-snap[3388]:
WARNING: joystick interface not connected! Please run: /snap/iot-example-
graphical-snap/current/bin/setup.sh
2021-11-16T16:10:24Z iot-example-graphical-snap.iot-example-graphical-snap[3388]:
ALSA lib conf.c:4120:(snd_config_update_r) Cannot access file /usr/share/alsa/alsa.
conf
2021-11-16T16:10:24Z iot-example-graphical-snap.iot-example-graphical-snap[3388]:
Failure to initialize SDL (Could not initialize UDEV)
2021-11-16T16:10:24Z systemd[1]: snap.iot-example-graphical-snap.iot-example-
graphical-snap.service: Succeeded.
2021-11-16T16:10:24Z systemd[1]: snap.iot-example-graphical-snap.iot-example-
graphical-snap.service: Scheduled restart job, restart counter is at 5.
2021-11-16T16:10:24Z systemd[1]: Stopped Service for snap application iot-example-
graphical-snap.iot-example-graphical-snap.
2021-11-16T16:10:24Z systemd[1]: snap.iot-example-graphical-snap.iot-example-
graphical-snap.service: Start request repeated too quickly.
2021-11-16T16:10:24Z systemd[1]: snap.iot-example-graphical-snap.iot-example-
graphical-snap.service: Failed with result ‘start-limit-hit’.
2021-11-16T16:10:24Z systemd[1]: Failed to start Service for snap application iot-
example-graphical-snap.iot-example-graphical-snap.
...
```

其中所有的警告信息都表明：snap软件正在开发中，接口尚未自动连接。所以，您需要连接所有尚未连接的接口，并手动运行如下的Daemon程序：

```
/snap/iot-example-graphical-snap/current/bin/setup.sh
snap start iot-example-graphical-snap
```

现在，界面上会显示Neverputt已启动。



结论

本指南介绍了使用Ubuntu Frame部署图形应用程序的过程，包含从测试到具体部署的全部过程，涉及到如何设置电脑的桌面工具及桌面环境、如何测试某个应用程序是否能在Ubuntu Frame上运行、如何将应用程序打包为可用于物联网设备的snap软件包。

同时，本文也涵盖了各个环节可能遇到的常见问题以及解决办法，并展示了确保snap软件包在设备上运行所需的详细步骤。剩下的开发流程就和使用其他snap软件包进行开发相同了：只需要上传snap软件包到Snap Store，然后从Snap Store安装该软件包到设备上。还有几点需要注意：

现在，我们已经配置好了Snap接口，系统（重新）启动后，应用程序将自动运行。

把snap软件包上传到Snap Store后，您可以要求Snap Store声明，以便snap软件包可自动连接任何尚未连接的接口。另外，如果您想创建一个“Snap Appliance”，那么就可以连接“Gadget Snap”中的接口。

欲详细了解Ubuntu Frame的更多信息，请访问我们的网站。

你也可以考虑阅读以下材料：

- 如何在Ubuntu Core上运行Flutter应用程序
- 如何利用现有snap软件包构建网络信息亭
- 如何在Ubuntu Core上配置音频
- 如何在Ubuntu Frame中启用屏幕键盘

如需获得投放市场建议，敬请联系我们。

© Canonical Limited 2022. Ubuntu, Kubuntu, Canonical 及其相关徽标皆为Canonical Ltd.的注册商标。所有其他商标是其各自所有者的财产。本文档中提及的任何信息如有更改，恕不另行通知，Canonical将不对任何此类更改承担任何责任。

Canonical Limited，一家于英格兰和威尔士注册的公司，公司代码110334C，注册办事处：12-14 Finch Road, Douglas, Isle of Man, IM991TT 增值税注册号：GB 003 2322 47

CANONICAL ubuntu