ACTIVITY HANDBOOK

How to get started with
software development
for the OLPC XO

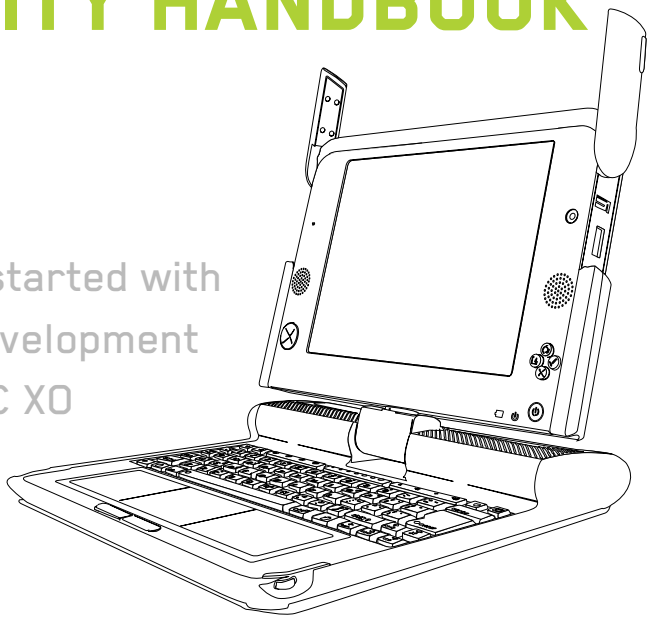


Table of Contents

Welcome to the Activity Handbook!

The purpose of this handbook is to provide you with all the information you need in order to get started with software development for the OLPC XO. We assume that all of you have used books such as „Teach Yourself C / Java / Visual Basic in 21 Days" and we want to provide a similar source of information for interested readers. The main difference here is that this handbook is not going to teach you the basics of programming. Neither are we going to teach you Python, the programming language we are mostly going to rely on, in this handbook. There are many other sources for learning these things and it would go far beyond the scope of this handbook to serve as an introduction to programming and / or Python. What we however do want to give you is a head start when it comes to writing software for the XO. To that end we will give you a detailed overview of both the hardware and software which is used in this remarkable machine.

Our motivation to write this handbook arose from our own desire to write code for the XO. We just did not know how to do it and the documentation we found on the internet was not as helpful as we had hoped. So we basically consider ourselves to be members of the target group that we envision this document to be useful for. We are writing the document we had hoped to find when we started out with XO software development ourselves. When we started working on this handbook in early October 2007 the documentation, especially when it comes to the software, of the XO left much to be desired. Realizing this OLPC had started looking for a „documentation lead" and various discussions took place in order to gather feedback on what documentation was needed. The OLPC Wiki contained several entries related to software development but there was no single coherent document that contained all the information a developer needs in order to get started with writing code. So while it was certainly doable to find that information it was by no means comfortable. Basically the barrier to entry was higher than it should have been.

When it comes to software development and especially documentation the process behind it is as important as the final result. So allow me to take a step back and describe how we currently envision this whole thing to work. As previously mentioned we feel that with documentation it is important to have a coherent piece of writing that readers can rely on. This thought lead us to the decision that a Wiki is not the best way of organizing the information the way we want to present it. We have

therefore settled for the DocBook format (http://en.wikipedia.org/wiki/Docbook) as the basis for this handbook. This decision of course entails that only a small number of people can actually actively edit and enhance the handbook. While this might seem like a limitation for some people we feel it is a necessary step in order to preserve the coherence we consider to be such an important characteristic of this handbook. We of course encourage a controlled collaboration process as 200 eyes spot more mistakes than 4 eyes and 100 brains know more than 2 brains. Additionally we are all restrained by the amount of time and energy we can invest into this project so we will never be able to provide a handbook that is 100% complete and finished. Therefore the handbook and all the information it contains is very much provided „as is" and you are free to do with it as long as you follow the rules imposed by the Creative Commons' „Attribution alone (CC-BY)" license that we are using for this project.

On a sidenote: At the end of many chapters you will find a section called „to do" which contains a list of things we still consider to be missing from the respective chapter. On the one hand this will serve as sort of a roadmap for ourselves but on the other hand we also actively encourage you to contribute missing pieces. Additionally we have provided a sort of chapter-framework that will still be relatively empty at the beginning. Again this is meant to help ourselves stay on course when writing the handbook. But again it should also enable people to start filling up the gaps by contributing (or simply suggesting) relevant documentation.

The best way to communicate with us (especially when you're interested in contributing to the activity handbook) is by sending an e-mail to handbook@olpcaustria.org. We very much value your feedback and will do our best to reply to your comments, questions, suggestions and rants as soon as possible.

Last but not least you will always find the latest version of the Activity Handbook over on http://www.olpcaustria.org/mediawiki/index.php/Activity_handbook.

Happy coding!

Christoph Derndorfer
Daniel Jahre

OLPC Austria (www.olpcaustria.org/), Vienna, January 2007

1 Introduction to Sugar

Chances are that if you are reading this handbook you have already spent considerable time reading about the OLPC project in general and the XO laptop and its hard- and software in more detail. Apart from the predictable question about the missing hand crank one of the first things everyone who sees an XO notes is the user interface. A user interface that is very different from what most of us are used to. No start button. No clock on the lower right corner of the screen. No virtual desktops. No list of running applications in a taskbar. No widgets. Very different indeed!

So, let us take a minute to get familiar with some of the characteristics of the Sugar desktop. The first question here is: What exactly is Sugar? In a nutshell Sugar is a GUI (short for „graphical user interface"), which is written in the Python programming language and runs on top of the X Window System and the Matchbox Window Manager (http://projects.o-hand.com/matchbox). A less technical description might sound like this:

„Sugar is a unique approach to making computers more accessible to a broader range of people."

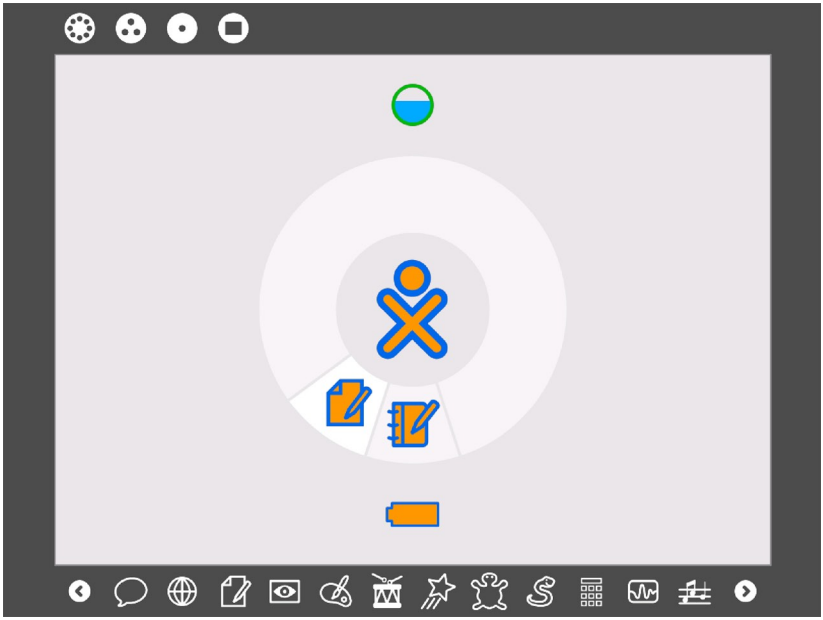
If you take a step back and think about the way we interact with computers then many words will come to mind. But few people will think of the term „intuitive". The simple fact is that our interaction with computers is highly artificial and things only really start to feel natural once you have spent a lot of time doing them a certain way. Also the names, symbols and abstractions we use are often less than self-explanatory. To give you a concrete example let us look at the way data is stored on computers. The reason why we are dealing with „files" and „folders" is quite historical in that it shows for what environment a lot of the early software was written: offices. It is not like this is the most complex thing to understand but at the same time it also is not that easy to crasp it. Especially not for a 6 year-old child living in the Peruvian Andes or Nigerian savanna. It must be possible to come up with an easier and more universal approach when it comes to how people interact with computers. This challenge sits at the very core of the OLPC human interface guidelines which is a definite must-read for anyone interested in developing software for the XO. In fact we would go as far as arguing that the article is a must-read for anyone who is interested in the challenges and requirements that lie at the heart of making computers more accessible, especially in a broader cultural context. Reading that text one realizes that in terms of innovation Sugar is one of the best examples of

the „thinking outside the box" approach that OLPC has taken throughout the whole project.

One vitally important concept that we have to explain at this point is that in the XO context there are not „applications" but rather „activities". The best explanation for why this is important can be found in the human interface guidelines document mentioned above:

„The laptop focuses children around „activities." This is more than a new naming convention; it represents an intrinsic quality of the learning experience we hope the children will have when using the laptop. Activities are distinct from applications in their foci—collaboration and expression—and their implementation—journaling and iteration."

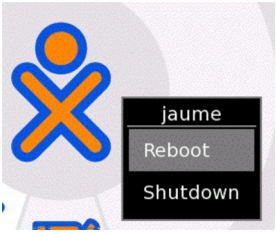
Now we are going to look at some of the Sugar elements in more detail. (Please note that these screenshots were taken with software builds 605 and 624 so things might look slightly different by the time you're reading this.)

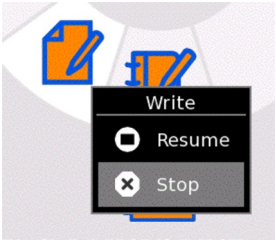


What you are looking at on the screenshot above is the so-called „home view" of

the Sugar desktop. The home view is best compared to the regular desktop that most of us look at when we start our computers. In the center of the screen you find the XO icon in the child's custom colours. The ring around it contains icons for all the activities which are currently running on the machine. Above that ring you can find the network status indicator while below it you have the battery status indicator. On the lower edge of screen you can see a scrollable list of all the activities which are installed on the laptop. The icons on the upper edge represent the 4 different views that are available in Sugar.

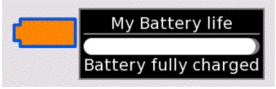
If you move across the XO icon in the middle the user's name is displayed. After a second of hovering above the icon you are presented with a menu where you can select to shutdown or reboot the machine.





Moving the cursor above an activity icon displays its name before more options pop up after another second. The default choice is of course resume so if you right-click on the icon then you will be taken directly into the activity.

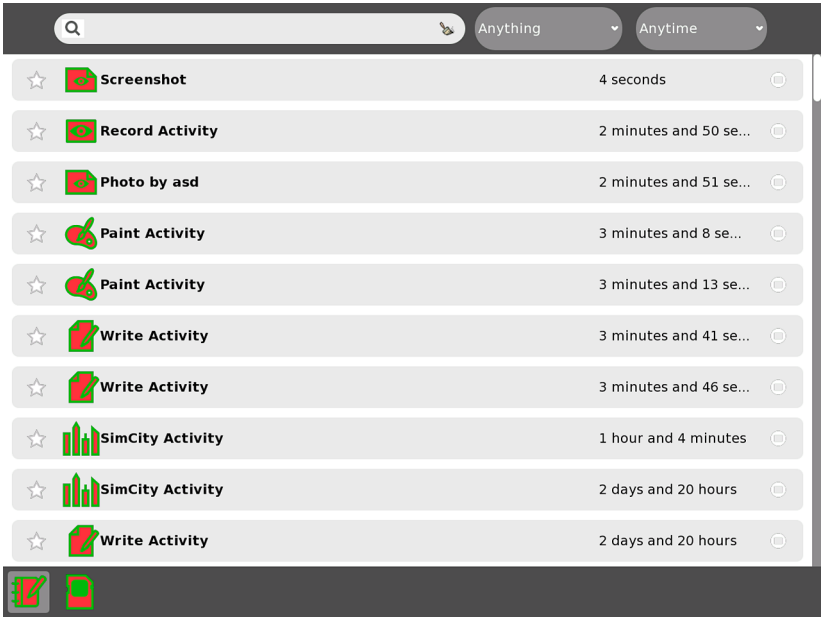
The same thing is true for the network status indicator. Moving across it reveals the SSID name of the network that you are currently connected to. After another second you get more information such as which channel is being used and of course also the option to disconnect from the network. The cool thing is that the icon also indicates the signal strength of the network connection, so if the circle is full then the signal is very good.



The battery icon also works in a very similar fashion as it roughly indicates the battery's current status and hovering over it reveals more details.

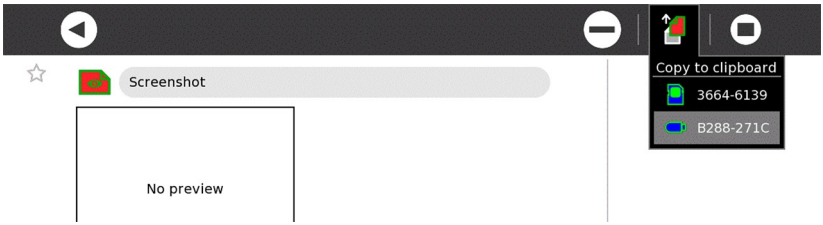
The toolbar on the bottom contains icons for all the currently installed activities plus there are scrollbuttons to gain access to more activities. The icons are pretty self-explanatory, from left to right the activities installed on this X0 are: chat, browse (internet browser), write (text processor), record (audio, video and still recording), paint, TamTamjam (one of three music / sound focused activities), Etoys (educational programming language), Turtle Art (drawing application similar to logo), Pippy (Python interpreter), calculate, measure and TamTamedit.

I am not going into details about what views are and how they work (we're going to talk more about that later). What is important to know at this point is that Sugar basically offers 4 different perspectives of the computer and you can always and easily switch between them.

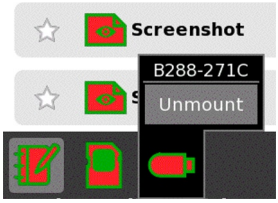


What you see on the screenshot above is one of Sugar's most important features, the so-called „Journal". The Journal is a unique and innovative way of accessing and storing data. Instead of ‚saving' and ‚opening' ‚files' the Journal allows you to

resume past activities. This again shows that the traditional seperation between „programs“ and „files“ doesn‘t really exist in Sugar. I‘m not going to go into more details at this point so please refer to the Journal entry on wiki.laptop.org (http://wiki.laptop.org/go/OLPC_Human_Interface_Guidelines/The_Laptop_Experience/The_Journal) for further information.



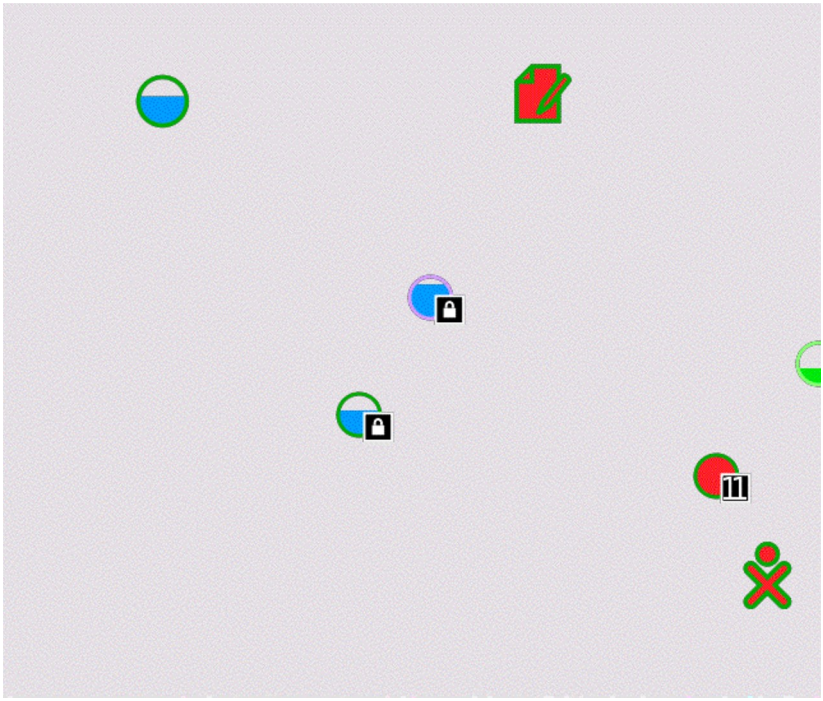
A concept that has however made it into Sugar is copy-paste. Once you select a Journal entry you have the possibility to copy it to your clipboard (it then shows up on the left side of Sugar‘s frame) or any removable storage device that you may have attached (in my case either an SD card or a USB thumbdrive).



When it comes to removable media you can also easily access them within the Journal. Unmounting these devices is also as easy as hoovering over it and waiting for the „unmount“ option to appear.

As previously mentioned collaboration also sits at the heart of the OLPC project. Therefore every actitivity should allow for children to easily collaborate. Collaborating is as easy as going to the „activity“ tab within an activity and selecting „Share with: My Neighborhood“. This will lead to the activity that you‘re sharing (in this

example it's write) to show up on everyone else's Neighborhood View:



Clicking on that activity-icon in their neighborhood will connect other users to you and within seconds you're able to collaborate on whatever activity you're using.

You can find some more information about how to use the OLPC XO laptop at www.laptop.org/en/laptop/start/index.shtml

Now that you have an idea about some of the core elements of Sugar let's take the next step and write our first activity, shall we?

2. Preparation

This chapter gives you an overview of what you need to do before you're able to dive into activity-development. We're going to look into the different options you have for development such as emulation, Live-CDs, sugar-jhbuild or on the actual XO laptops. Get, set, ready, go..

Emulation

Currently emulating an XO is the recommended way for both demo'ing and developing software running on the XO laptops. As always there are some limitations related to emulation which can range from cosmetic details such a the wrong font-size, to the lack of audio-support to core-issues such as the obvious lack of a mesh-network and camera. Evaluating the speed of execution of an actual piece of code is also hardly possible when you're using an emulation. Regardless of what emulation you're using there are some common factors that you need to consider when setting up your environment. Probably one of the most important ones is deciding how much memory you're going to allow your emulation to use. The current OLPC XO-1 hardware comes with 256MB DDR-RAM so this is the recommended setting for a relatively close simulation of the performance you can expect on the OLPC XO. In general there are several different routes that you can take, below you can find a list of your options depending on the operating system that you're running:

Linux

We assume that most developers running Linux will have some experience with emulation so we're not going to go into details here. The best way to emulate Sugar on a Linux machine is to use QEMU, alternatively you can also rely on solutions such as VirtualBox or Vmware. Please see the Windows XP emulation section below for more details.

Mac OS X

Due to the fact that we don't have access to a machine running Mac OS X we can't comment on its capabilities when it comes to Sugar emulation. If you have any experience with this then please contact us at handbook@olpcaustria.org.

Windows XP

At the time of writing (January 2008) there are three different ways to emulate an XO environment on a Windows XP host-machine. In general emulation on Windows

XP works rather painlessly however still more information and feedback is required about emulating Sugar under Windows Vista. While we have three sub-sections below there are actually two main approaches to emulation, QEMU and VirtualBox/ VMware:

QEMU on Windows

Links	http://www.h7.dion.ne.jp/~qemu-win/ http://wiki.laptop.org/go/QEMU
Description	We ran into some issues with QEMU on Windows which we haven't been able to solve yet. We will update this section once we have more information.

VirtualBox

Links	http://www.virtualbox.org/ http://dev.laptop.org/pub/virtualbox/ http://wiki.laptop.org/go/VirtualBox
Description	VirtualBox is a popular emulator that runs under both Linux and Windows. Unfortunately we experienced some issues when testing VirtualBox (under Windows XP) with some of the pre-converted images from dev.laptop.org/pub/virtualbox/. Build 649 or Build 655 (Ship 2) for example would go into a constant reboot cycle and never managed to succesfully load Sugar. Based on this experience we would currently recommend using VMware (see below). However we would appreciate any comments and experiences about using VirtualBox for Sugar emulation. So please e-mail us at handbook@olpcaustria.org and also post your findings on http://wiki.laptop.org/go/VirtualBox.

VMware

Links	http://www.vmware.com/ http://dev.laptop.org/pub/virtualbox/ http://wiki.laptop.org/go/VMWare
Description	The easiest way to get started with an XO emulation is to download a pre-converted image from dev.laptop.org/pub/virtualbox/ and get the latest version of the free VMware Player (http://www.vmware.com/). After verifying that the download worked flawlessly by comparing the md5-checksum with the published md5 value you can unpack the .zip file and start the virtual-machine by double-clicking on the *.vmx file.

With regards to the settings of the virtual-machine we're using the following configuration:

Memory	256MB
Hard Disk	olpc-6xx.vmdk
Ethernet	Device status: Connect at power on (enabled) Network connection: custom (VMnet0 (default Bridged)
USB Controller	Automatically connect new USB devices to this virtual machine when it has focus. (enabled)
Audio	Device status: Connect at power on (enabled) Connection: Use default host sound adapter (selected)
Virtual Processors	Processors: Number of processors (One)

Live-CD

At the moment there are two different Live-CDs (three if you count a do-it-yourself version) available for testing Sugar without having to install anything on your hard-drive:

(1) Xubuntu Live-CD

Links	http://skolelinux.de/XO-LiveCD/
Description	This Live-CD is built around the popular Xubuntu distribution (www.xubuntu.org) and at the moment this the one we'd recommend to most people. After inserting the Live-CD and re-booting your computer the system comes up with a simple splash-screen where you have to select „Start or install Xubuntu" to continue. Then you simply wait for the desktop to come up, click on the „Applications" button in the top-left corner and select „Other" -> „Sugar Emulator". Voilà, Sugar starts and you can now type in your name, select a colour and explore it! Alternatively you can also logon to your Live-CD by clicking on the „Sessions" button on the login screen and selecting „2. Sugar" from the list. However this didn't work reliably at publication time. Other than that this approach has been proven to work quite well and we're currently not aware of any major issues regarding the Xubuntu Live-CD.

(2) LiveBackup Live-CD

Links	ftp://rohrmoser-engineering.de/pub/XO-LiveCD/ http://skolelinux.de/XO-LiveCD/ (http-mirror)
Description	This Live-CD was created with LiveBackup (http://livebackup.sourceforge.net/) and at publication time the latest version (XO-LiveCD_071206.iso) relies on Sugar build 625, therefore also significantly lagging behind the current latest build 650. Nevertheless it's quite an easy and convenient way to use Sugar. After inserting the Live-CD and re-booting your computer the system comes up with a simple splash-screen where you have to press „Enter" to continue. Afterwards it takes you straight to the screens where you can enter your name and colour. At publication time this Live-CD still suffered some two well-known bugs with Live-CDs: (a) small font size and (b) issues with the audio (mainly when using the TamTam activity). The number of included activities is also smaller than on the Xubuntu Live-CD mentioned above. Please also note the XO-LiveCD.pdf file which contains a short overview of the LiveBackup Live-CD.

(3) Pilgrim Fedora Live-CD [DIY]

Links	http://gregdek.livejournal.com/19843.html
Description	RedHat's Greg DeKonigsberg started a kickstart file to create a Live-CD image based on Fedora and built using the Fedora livecd-creator. We haven't tested this path up to now but in case you have please let us know how it went by e-mailing us at handbook@olpcaustria.org and adding the information to http://wiki.laptop.org/go/LiveCd.

sugar-jhbuild

The sugar-jhbuild skript is a python script that is using for building and running Sugar from source. It might have dependencies on your System that you have to install before you can use sugar-jhbuild. You should check the laptop.org Wiki if you have all the required packages on your Distribution or OS. The script is managed in a git repository. You need to have git installed on your system. You can download the script by typing

```
git-clone git://dev.laptop.org/sugar-jhbuild
```

Now you can change to the directory sugar-jhbuild by typing

```
cd sugar-jhbuild
```

Use git-pull to sync to the Repository. Now you are ready to use sugar-jhbuild. As a first step tell jh-build to download all the required source packages to download by using sugar-jhbuild's update command.

```
./sugar-jhbuild update
```

Use the build command to run the actual build.

```
./sugar-jhbuild build
```

sugar-jhbuild implicitly calls update on every run, so you can skip the update run in the first place but it might be better to have the huge download in one rush before start building. Sometimes some of sugar-jhbuild's builtin checks for required packages will tell you about missing requirements even if they are installed. If you are sure that you met all requirements you can switch that checks of by invoking the command as

```
./sugar-jhbuild build -x
```

The script runs interactively by default. This means if an error occurs the script will stop and asking you what to do next. If you don't like this behaviour because you don't have the time to attend the progress of the build all day then use the --no-interact switch to turn it off. Be aware that this might lead to a broken or incomplete build.

If everything built fine you can now start Sugar by typing

```
./sugar-jhbuild run
```

If you want to run multiple instance e.g. for testing collaboration you can set an environment variable like this

```
SUGAR_PROFILE=2 ./sugar-jhbuild run
```

This would run two instances of Sugar at the same time.

Ubuntu packages

Given that the OLPX XO is built on a Fedora 7 base environment and uses many standard libraries (see a full list of software components at http://wiki.laptop.org/go/Software_components) it should theoretically be quite easy to run Sugar under a variety of different Linux distributions. Unfortunately this isn't the case at the moment even though we do expect to see a lot of progress in that area throughout the first half of 2008. For now your best bet is to use Ubuntu (see below for details) even though other distributions might work as well. Please let us know (handbook@olpcaustria.org) if you have any experience with running Sugar on other distributions so we can add it below!

Ubuntu (7.10 Gutsy Gibbon / 8.04 Hardy Heron)

Links	http://www.ubuntu.com http://wiki.laptop.org/go/Sugar_on_Ubuntu_Linux https://edge.launchpad.net/~jani/+archive
Description	If you're running Ubuntu (7.10 Gutsy Gibbon or 8.04 Hardy Heron) you can also use deb packages to get Sugar (plus several activities) directly on your system. Please follow the instructions available at http://wiki.laptop.org/go/Sugar_on_Ubuntu_Linux (please note that the wiki entry only mentions Gutsy support at the moment) and https://edge.launchpad.net/~jani/+archive which explain all the steps you need to take in order to get Sugar running on your Ubuntu system. Even with Christoph's Linux experience being somewhat limited he managed to get Sugar running on his laptop in less than 10 minutes, it's really that easy!

XO Laptop

At the time of writing these lines (January 2008) OLPC is working on a new way for people to request post-mass-production developer machines. Currently the number of laptops available to developers is still somewhat limited (this is subject to change though as more units are produced) but interested developers can directly apply at OLPC for an XO-1.

Please keep a close eye on
http://wiki.laptop.org/go/Developers_Program#How_to_apply_for_an_XO and
http://wiki.laptop.org/go/Contributors as more information becomes available.

Developers will have to rely on standard text-editors such as „vi" and „nano" for coding their activities. You can basically develop a complete activity by just using one of these programs. However we suggest that you use at least a half-decent graphics-editors for designing the .svg icons for your activity. Additionally you'll need an application such as „zip" for creating the .xo file which contains all the files required by the activity.

Chris from OLPC Austria has written up a nice page on how to customize and configure an XO:
www.olpcaustria.org/mediawiki/index.php/Xo_setup_user_guide

3. Sugar Basics

This chapter covers the basics of creating a software bundle for the XO, including a short code sample („hello world", what else?) to show you all the steps you need to take in order to write an activity.

First of all it's important to explain the concept behind these activity bundles. Here's what the description from the Activity Bundles (http://wiki.laptop.org/go/Activity_Bundles) entry on wiki.laptop.org has to say:

Every activity in the Sugar environment is packaged into a self-contained „bundle". The bundle contains all the resources and executable code (other than system-provided base libraries) which the activity needs to execute. Any resources or executable code that is not provided by the base system must be packaged within the bundle.

One of the main reasons why these self-contained bundles play such a vital role within the XO ecosystem is that children should easily be able to transfer activities they don't have yet to their own laptops.

Activities are meant to be shared between children. If a child doesn't have the activity, it is automatically transfered to the child when he or she joins the shared activity. Packaging activities in self-contained bundles allows easy sharing, installation, removal, and backup.

The package layout

Like most modern package formats the .xo package requires a specific directory layout and files that follow a certain naming scheme. The basic directory layout for our sample activity is shown below:

```
MyActivity.activity/
MyActivity.activity/MANIFEST
MyActivity.activity/MyActivity.py
MyActivity.activity/setup.py

MyActivity.activity/activity
MyActivity.activity/activity/activity.info
MyActivity.activity/activity/activity-myactivity.svg
```

```
MyActivity.activity/lib (additional libraries required
by the activity) [optional]
MyActivity.activity/locale (additional libraries
required for localization) [optional]
```

The package files in more detail

MANIFEST

This file contains a list of the files which are part of the Activity Bundle. (Make sure that you don't leave any blank lines at the end of the file!)
In our example the MANIFEST looks like this:

```
MyActivity.activity/HelloWorldActivity.py
MyActivity.activity/setup.py
MyActivity.activity/activity/activity.info
MyActivity.activity/activity/activity-helloworld.svg
```

HelloWorldActivity.py

This file contains the code of our activity.

```
from sugar.activity import activity
import logging

import sys, os
import gtk

class HelloWorldActivity(activity.Activity):
    def hello(self, widget, data=None):
        logging.info('Hello World')

    def __init__(self, handle):
        print "running activity init", handle
        activity.Activity.__init__(self, handle)
        print "activity running"

        self.set_title('Hello World')
```

```
        # Creates the Toolbox. It contains the
        # Activity Toolbar, which is the bar that appears
        # on every Sugar window and contains essential
        # functionalities, such as the 'Collaborate'
        # and 'Close' buttons.
        toolbox = activity.ActivityToolbox(self)
        self.set_toolbox(toolbox)
        toolbox.show()

        # Creates a new button
        # with the label "Hello World".
        self.button = gtk.Button("Hello World")

        # When the button receives the "clicked"
        # signal, it will call the function hello()
        # passing it None as its argument.  The hello()
        # function is defined above.
        self.button.connect("clicked", self.hello, None)

        # Set the button to be our canvas. The canvas
        # is the main section of every Sugar Window.
        # It fills all the area below the toolbox.
        self.set_canvas(self.button)

        # The final step is to display this newly
        # created widget.
        self.button.show()

        print "AT END OF THE CLASS"
```

setup.py

```
from sugar.activity import bundlebuilder
if __name__ == "__main__":
       bundlebuilder.start('HelloWorld')
```

activity.info

The activity.info file contains all important metadata for the package similar to a .spec file in a rpm package. The fileformat is kept simple, it looks like the .ini files from dos and Windows and are also specified on freedesktop.org

```
[Activity]
name = HelloWorld
service_name = com.olpcaustria.HelloWorldActivity
activity_version = 1
host_version = 1
bundle_id = org.olpcaustria.HelloWorldActivity
icon = activity-helloworld
class = HelloWorldActivity.HelloWorldActivity
mime_types = application/pdf;image/tiff
show_launcher = yes
```

Here's a more detailed break-down of the file:

`[Activity]:` The activity.info file must always start with „[Activity]", no other tag is allowed in the first line of the file!
`name:` The name of the activity - unless defined with an additional language code in []-brackets is assumed to be „en_US".
`activity_version:` A single positive integer, larger values are considered to be „newer" versions.
`host_version:` Again a single positive integer, which specifies the version of the Sugar environment which the activity is compatible with. (While writing this book no higher versions than 1 for Sugar are available.)
`bundle_id:` Activity bundle identifier which is also used as the default service type when sharing the activity.
`icon:` Specifies the name of the activity's icon which is expected to be in the base directory and has an .svg extension.
`class:` Defines which class to start upon the activity initialization. (You can either use the „class" or the „exec" line below to define that class.)
`mime_types:` Specifies which mime types (separated by semi colons) are supported by the activity, is used when downloading files from the web or for offering the activity when resuming something from the Journal. Don't use it unless your

activity really needs and supports these file-types.
`show_launcher:` (optional) If this tag is missing or set to „yes" the activity's icon will be shown in Sugar's launcher panel.

icon.svg

The activity icon has to be placed in the /activity sub-directory and should use the .SVG format (http://www.w3.org/Graphics/SVG/ - SVG is a a XML based language for two-dimensional vector graphics). Please take a look at the Sugar Icon Format (http://wiki.laptop.org/go/Sugar_Icon_Format) and the Icon section of the Human Interface Guidelines (http://wiki.laptop.org/go/OLPC_Human_Interface_Guidelines/The_Sugar_Interface/Icons) for more information about icon design.

Here's a sample icon produced by Daniel Jahre's activity-bundle-generator

```
<!--
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
 "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd" [
  <!ENTITY fill_color "#FFFFFF">
  <!ENTITY stroke_color "#000000">
]>
<svg xmlns=Chttp://www.w3.org/2000/svg" width="50"
height="50">
  <rect x="1" y="1" width="48" height="48"
  style="fill:&fill_color;;stroke:&stroke_color;;stroke-
width:2"/>
</svg>-->
```

xo bundle

All that's left to do at this point is to create the .xo bundle package. You can easily accomplish this by using a compression tool of your choice to make a zip-archive. Then you simply rename the .zip file to HelloWorld.xo and voila, you've just finished your first activity!

Localization issues

In the locale directory are subdirectories for each locale that the activity is localized for. The localization is done in files called activity.linfo. How localization is handled will be described further down the road.