Различия между Maemo 4 и 5

Under construction

Maemo 5 с точки зрения пользователя

Изменения в Maemo 5 связаны с переходом к Hildon 2.2. стилю пользовательского интерфейса.Теперь интерфейс стал более оптимизирован для сенсорного управления. В связи с этим увеличился размер элементов пользовательского интерфейса, тем их размер приближен к размеру пальца. Это в свою очередь означает, что количество видимых элементов сокращается. В Maemo 5 максимальное количество элементов в меню - 10, в то время, как в Maemo 4 наибольший или полный набор функциональных возможностей приложения, организованный в иерархическом меню. В Maemo 5 никакой иерархичности нет, меню представления содержит только те команды, которые относятся к данному представлению (не прикладное меню). Исключение составляет корневое меню, которое дополнительно содержит пункты для прикладных параметров настройки. Удален фокус - навигация клавиатуры не поддерживается в UI вообще. Отсутствует “затемнение”, если элемент неактивен, то он должен исчезать. Удалены вкладки в диалогах. Если документ закрывается, то он сразу сохраняется, не спрашивая подтверждения. Меню стало стековым. Уменьшено число панелей инструментов и контекстно-зависимых меню. Изменения на примере почтового клиента:
http://wiki.maemo.org/Documentation/Maemo_5_Developer_Guide/Porting_Software/Redesigning_From_Maemo_4_to_Maemo_5#Re-design_of_Modest_email

Maemo 5 с точки зрения разработчика

Рассматривается интерфейс и модули определения географического положения

Основные компоненты графического интерфейса пользователя

Сюда входят: C library, Xlib, Glib, GDK, GDK, Pango, ATK, GTK+, Hildon. Коротко о них.

http://wiki.maemo.org/Legacy_Maemo_5_Documentation/Graphical_UI_Tutorial/Introduction

Основные подсистемы пользовательского интерфейса: hildon-desktop - OpenGL graphics API

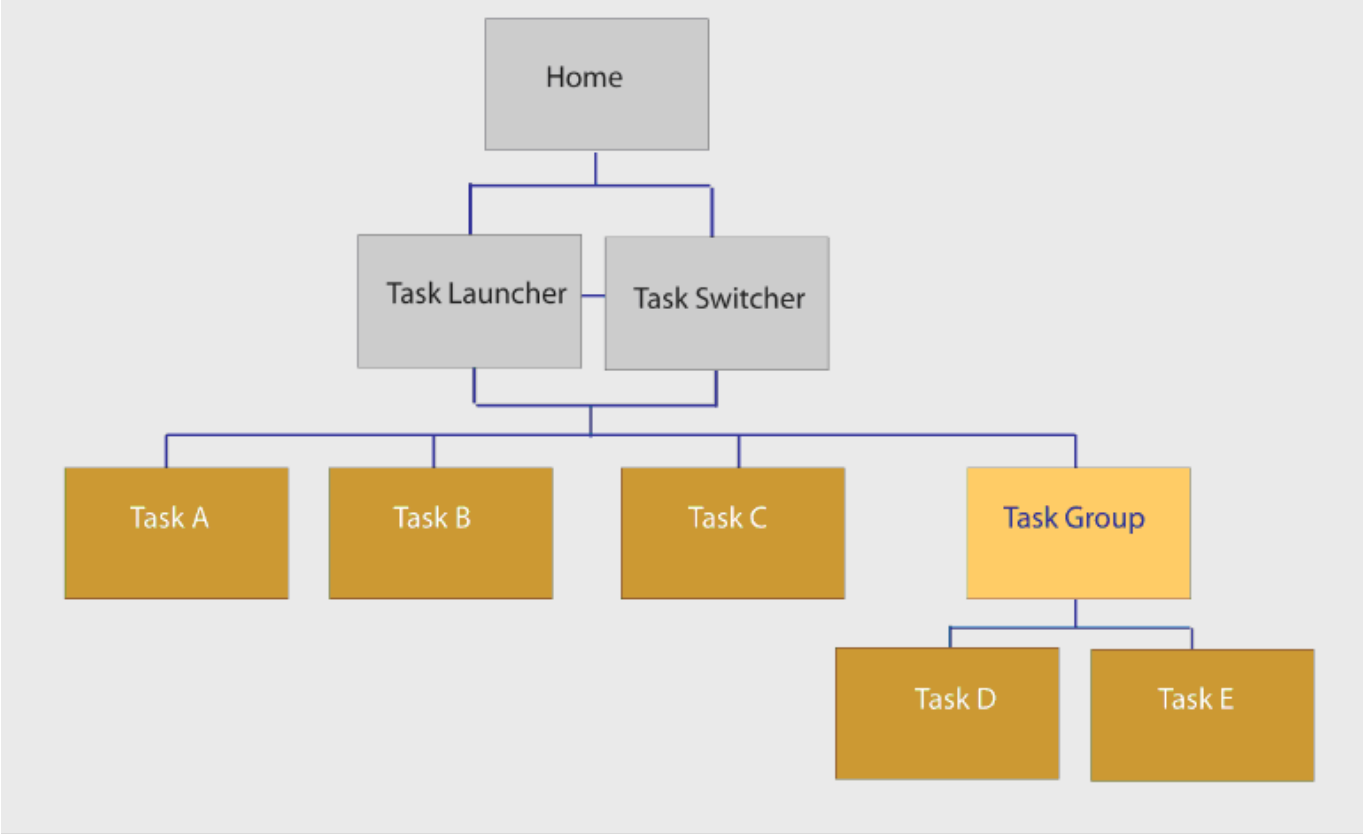
hildon-home - Home widget API and loading, notification service and plugins, background image and loading

hildon-status-menu - X clipboard selection management

Hildon Input Method - Localized text input UI Control Panel - Control Panel applet and loading Startup Wizard - first boot and system time setting Startup scripts - AF startup scripts RFS and CUD framework - Reset factory settings and clear user data

Hildon - изменения в Maemo5

Navigation architecture



Вид рабочего стола при отображении окна приложений:

A-Кнопка переключения задач. Нажатие на нее приводит к переходу приложения в фоновый режим и отображению доступных для выбора других запущенных приложений. B- Кнопка статуса области. Предоставляет информацию о статусе устройства или приложения. C- Кнопка для закрытия приложений. При этом, если текущее окно приложения является подвидом, то вместо кнопки “закрыть” отображается кнопка “назад”. И кнопка назад закрывает окно, не закрывая приложения, а возвращая к предыдущему виду. D- Название области. Показывает идентификатор текущей задачи, обычно имя текущей задачи. F- Область приложения.

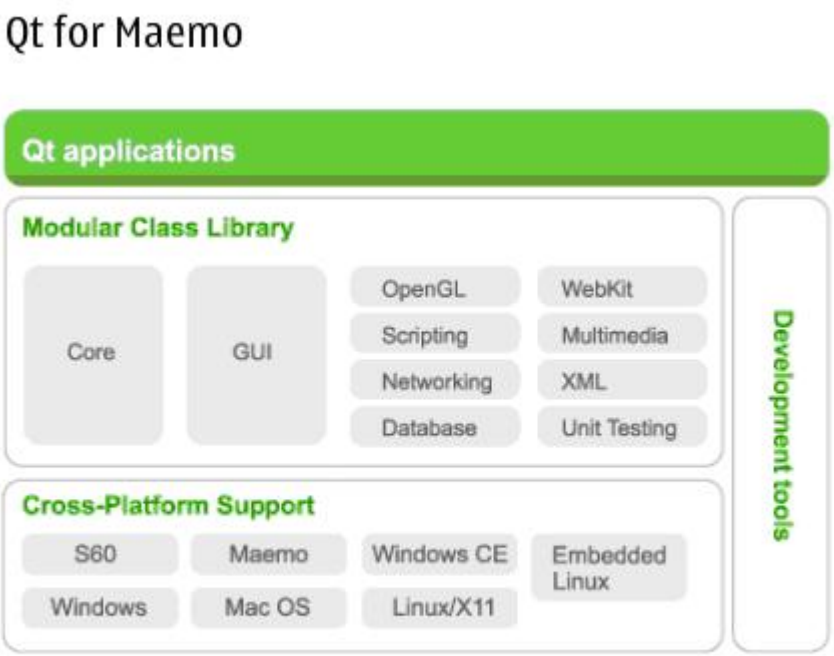
При работе приложения в полноэкранном режиме используется только область приложения. Если приложение имеет панель виджетов, то она она отображается в самой нижней части области приложения, как в нормальном, так и в полноэкранном режиме.

Анализ на основе таблицы
http://repository.maemo.org/stable/fremantle/4.1.2_vs_5.0_content_comparison.html
http://gitorious.org/fremantle-hildon-desktop

GTK - изменения в Maemo5

На основе таблицы
http://repository.maemo.org/stable/fremantle/4.1.2_vs_5.0_content_comparison.html

Qt



http://doc.trolltech.com/4.5/index.html

Пакет qt4-x11

http://packages.debian.org/ru/source/lenny/qt4-x11

Компиляция Qt приложений

http://wiki.maemo.org/Qt4Hildon#Overriding_the_Qt_Maemo_changes

Портирование QT приложений на Maemo5

http://wiki.maemo.org/Qt4Hildon#Overriding_the_Qt_Maemo_changes
http://wiki.maemo.org/Packaging_a_Qt_application

Сосуществование Qt и GTK+ на Maemo5

Новые Возможности написания интерфейса, предоставленные появлением Qt

Использование модулей для определения географического положения

Location framework в MAEMO4: liblocation

Liblocation — библиотека, состоящая из нескольких модулей и предоставляющая разработчику приложений для Maemo средства определения географического положения. Liblocation включает в себя модули: LocationGPSDevice, LocationGPSDControl, location-distance-utils, location-misc. Модуль LocationGPSDevice содержит набор типов данных и функций, позволяющих принимать информацию от демона местоположения (например, GYPSY), который в свою очередь взаимодействует с GPS устройством. Модуль LocationGPSDControl определяет объект, который позволяет управлять соединением с демоном местоположения. В модуле определены функции, информирующие демона об открытии или закрытии соединения с ним. Демон положения взаимодействует с устройством до тех пор, пока хотя бы одно приложение поддерживает с этим демоном соединение. Модуль location-distance-utils предоставляет разработчику средство определения расстояния между двумя точками на поверхности Земли. Модуль location-misc содержит набор дополнительных функций для работы с liblocation. На данный момент определяет одну функцию, которая осталась для совместимости и не рекомендуется для использования при разработке новых программ. Maemo Diablo включает liblocation версии 0.30. Реализации версий liblocation до 0.30 были основаны на взаимодействии с демоном местоположения GPSD. Следующая версия liblocation 0.90 основана на GYPSY и D-Bus.

Location framework в MAEMO5: gypsy vs gpsd

Maemo Diablo для работы с GPS устройствами использовало демон GPSD. Maemo Fremantle содержит liblication версии 0.99, которая вместо GPSD использует GYPSY. GYPSY был написан, чтобы исправить проколы, обнаруженные в GPSD. Одним и з проколов была проблема выделения памяти (allocating memory). Разработчики GPSD категорически не рекомендовали использовать функции malloc/free (http://gpsd.berlios.de/hacking.html#malloc). Кроме того, в GPSD клиенту необходимо открывать сокет для взаимодействия с сервером GPSD. Обмен информацией достаточно сложный. В GYPSY взаимодействие основано на использовании мощной сигнальной системы D-Bus. Существует ряд других причин, по которым предпочтительнее использовать GYPSY, а не GPSD.

Location framework в MAEMO5: liblocation, gypsy daemon, location daemon

Location framework в Maemo5 включает в себя компоненты liblocation, gypsy daemon, location

daemon, которые относятся к категории proprietary (Источник). Разработчикам предоставляется API для работы с этими компонентами.

Библиотеки нижнего уровня для определения геграфического положения в Maemo4

К библиотекам нижнего уровня для определения положения в Maemo Diablo относятся библиотеки: libgpsbt, libgpsmgr, gpsd daemon.

GPS BT (libgpsbt)

Application that needs GPS data can use this API to manage GPS daemon start and stop and to find out what serial port device (/dev/rfcommX) the GPS device is using. The API uses services provided by libgpsmgr that handles the GPS daemon start and stop.

GPS Manager (libgpsmgr)

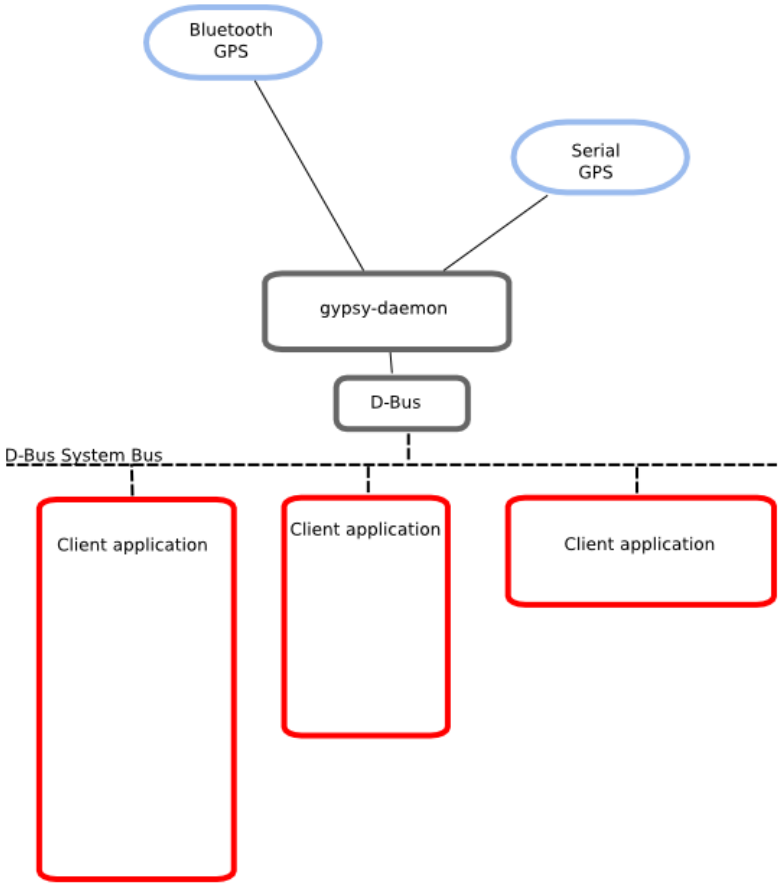
Application that needs GPS data can use this API to manage GPS daemon start and stop. This is needed in small handhelds where it is not good to have gpsd running all the time (because of battery and memory constraints) and to guarantee that only one gpsd process is running simultaneously even if multiple processes need the GPS data. The idea of the API is to allow multiple applications to use gpsd services but only when there is at least one application that needs GPS data. If no application is running, the gpsd is not started at all or gpsd stops itself automatically (to save memory and battery life).

Библиотеки нижнего уровня в Maemo 5.0 (Fremantle)

В MAEMO5 отсутствуют библиотеки нижнего уровня libgbsbt, libgpsmgr, gpsd daemon. Вместо этих библиотек в MAEMO5 предполагается использование liblocation либо API для обращения к gypsy daemon и к location daemon.

Gypsy daemon

Gypsy — это многоканальный GPS демон, обеспечивающий нескольким клиентам доступ к GPS информации, полученной с нескольких GPS источников параллельно. Gypsy использует D-Bus для оповещения клиентов, расположенных на системной шине, путем формирования сигналов об изменении GPS информации. При этом клиенты получают только те сигналы, на которые они зарегистрировались и не получают сигналы, которые им неинтересны. Таким образом Gypsy отлично подходит для устройств с низким энергопотребелением, так как приложения просыпаются только при действительной необходимости, тем самым сберегая энергию.



Методы и сигналы клиентов Gypsy разделены на следующий интерфейсы:

- Gypsy.Server: методы для управления функциями сервера Gypsy демона;
- Gypsy.Device: методы и сигналы, связанные с функциями GPS устройства;
- Gypsy.Postion: методы и сигналы, связанные с текущим GPS положением;
- Gypsy.Course: методы и сигналы, связанные с текущим курсом GPS;
- Gypsy.Accuracy: методы и сигналы, связанные с точностью GPS;
- Gypsy.Satellite: методы и сигналы, связанные со спутниками, которые видит GPS устройство.

Gypsy разделяет эти интерфейсы и не использует один большой для того, чтобы клиенты клиенты могли слушать только те сигналы, в которых они заинтересованы. Это позволяет клиентам оставаться в состоянии простоя более долгий период времени, а следовательно экономить энергию.

Для взаимодействия с Gypsy демоном Gypsy содержит библиотеку libgypsy на языке C, основанную на Gobject и являющуюся оболочкой D-Bus API, библиотека упрощает написание клиентов Gypsy демона. В тоже время для написания клиентов можно использовать любой другой язык, имеющий поддержку D-Bus.

Gypsy tutorial (пример простого GPS-клиента)

The libgypsy API documentation

Location daemon

Liblocation

Liblocation — библиотека, входящая в состав Location Framework и предназначенная для разработки приложений в Maemo, которые работают с географическим положением. Данная библиотека поддерживает внутреннее GPS устройство, bluetooth GPS устройсво, а также дополнительные методы определения географического положения, использующие сотовую связь. В Maemo5 liblocation основана на взаимодействии с Gypsy демоном с помощью D-Bus. Существует два способа взаимодействия с Gypsy демоном — это использование библиотеки libgypsy и прямое взаимодействие с демоном с помощью API D-Bus. Библиотека liblocation предлагает третий способ, обеспечивая собственный интерфейс взаимодейсвия с Gypsy демоном.

Модули liblocation

Для использования библиотеки liblocation необходимо включить в программу два заголовочный файла:

```
#include <location/location-gps-device.h>
#include <location/location-gpsd-control.h>
```

Объект LocationGPSDControl

LocationGPSDControl — объект типа GObject, позволяющий запускать и останавливать различные сервисы, устанавливать метод определения географического положения и интервал, а также получать информацию о различный ошибках.

Для создания объекта в библиотеке определена функция location_gpsd_control_get_default():

```
LocationGPSDControl *control = location_gpsd_control_get_default();
```

Для работы с объектом в библиотеке определены функции:

- location_gpsd_control_start () : Starts an active connection to Location server. In other words, by calling this function the application informs that it wants to use the location service.
- location_gpsd_control_stop () : Informs the location framework that the application is no longer interested about the current location. Location service is kept running as long as there is at least one application using it. Please note that LocationGPSDevice still sends data as long as the location framework is running.

Объект обладает следующими свойствами:

- “maincontext-pointer” (тип gpointer, write): устанавливает main context address;
- “preferred-interval” (тип gint, Read / Write): принимает значения >= -1, определяет интервал, через который будет определяться положение, по умолчанию -1;
- “preferred-method” (тип gint, Read / Write): принимает значения [0, 15], определяет метод,

по которому будет определяться положение, по умолчанию 0.

Объект может принимать следующие сигналы:

- Сигнал “error”
- Сигнал “error-verbose”
- Сигнал “gpsd-running”
- Сигнал “gpsd-stopped”

Более подробно в liblocation Reference Manual

Методы определения географического положения

- LOCATION_METHOD_USER_SELECTED: Liblocation выберет наилучший из возможных методов определения географического положения, опираясь на настройки в панели управления. Данный метод можно представлять как совокупность всех остальных: CWP+ACWP+GNSS+AGNSS. При отсутствии конкретных требований лучше всего выбирать этот метод.
- LOCATION_METHOD_CWP - Complementary Wireless Positioning: метод определяет координаты цента текущей страны с точностью по горизонтали равной радиусу страны (MCC fix (мобильный код страны)) или определяет координаты на основе используемой GSM станции. Позднее используется при наличии информации в кэше устройства, которая затем уточняется методом ACWP. Для метода CWP необходима SIM карта.
- LOCATION_METHOD_ACWP - Assisted Complementary Wireless Positioning: в определении географического положения участвует базовая сотовая станция, на которой зарегистрировано устройство. Для метода ACWP необходима SIM карта и сотовая сеть. Если сотовая сеть недоступна, то метод работает аналогично CWP. Приложение может определить MCC (мобильный код страны) прежде чем получит информацию о сотовой станции с внешнего сервера географического положения или если сеть недоступна.
- LOCATION_METHOD_GNSS - Global Navigation Satellite System: Метод использует GPS приемник. Как правило время первого определения положения значительно дольше, чем при использовании метода AGNSS. Ни SIM карта, ни сотовая сеть не нужны для этого метода. Данный метод может быть также использован в режиме «offline».
- LOCATION_METHOD_AGNSS - Assisted Global Navigation Satellite System Метод использует GPS приемник с вспомогательной информацией от внешнего сервера географического положения. Для данного метода необходимы SIM карта и сотовая сеть. Если сотовая сеть или SIM карта недоступны, то метод эквивалентен методу GNSSA.

Пример выбора метода:

```
g_object_set(G_OBJECT(control), "preferred-method", LOCATION_METHOD_GNSS |
LOCATION_METHOD_AGNSS, NULL);
```

Интервалы определения географического положения

Интервал между последующими определениями координат может равняться 1, 2, 5, 10, 20, 30, 60 и 120 секундам. В реальности интервал может значительно отличаться для обеспечения

производительности и в целях экономии энергии, он зависит от множества факторов (например частота проверки приложением изменения координат). Если несколько приложений установили различные интервалы, то выбирается наименьший. Пример задания интервала:

```
g_object_set(G_OBJECT(control), "preferred-interval", LOCATION_INTERVAL_60S,
NULL);
```

Объект LocationGPSDevice

LocaionGPSDevice — объект типа GObject, содержащий информацию о состоянии устройства и о текущем местоположении, если оно определено.

Объект может принимать сигнал «changed», который производится каждый раз, когда приходит сообщение об изменении координат.

Более подробно в liblocation Reference Manual

Пример работы с liblocation в Maemo.

Ниже приведен пример работы с liblocation в Maemo 5, который был взят с сайта maemo (http://wiki.maemo.org/Documentation/Maemo_5_Developer_Guide/Using_Connectivity_Components/Using_Location_API#Complete_example).

Here is a complete standalone example using liblocation. It starts location services after program is started, then when first fix arrives, prints it, stops services, and shutdowns.

```
 #include <location/location-gps-device.h>
 #include <location/location-gpsd-control.h>

 static void on_error(LocationGPSDControl *control, LocationGPSDControlError
error, gpointer data)
 {
    g_debug("location error: %d... quitting", error);
    g_main_loop_quit((GMainLoop *) data);
 }

 static void on_changed(LocationGPSDevice *device, gpointer data)
 {
    if (!device)
        return;

    if (device->fix) {
        if (device->fix->fields & LOCATION_GPS_DEVICE_LATLONG_SET) {
            g_debug("lat = %f, long = %f", device->fix->latitude,
device->fix->longitude);
            location_gpsd_control_stop((LocationGPSDControl *) data);
        }
    }
 }
```

```
 static void on_stop(LocationGPSDControl *control, gpointer data)
 {
    g_debug("quitting");
    g_main_loop_quit((GMainLoop *) data);
 }

 static gboolean start_location(gpointer data)
 {
    location_gpsd_control_start((LocationGPSDControl *) data);
    return FALSE;
 }

 int main(int argc, char *argv[])
 {
    LocationGPSDControl *control;
    LocationGPSDevice *device;
    GMainLoop *loop;

    g_type_init();

    loop = g_main_loop_new(NULL, FALSE);

    control = location_gpsd_control_get_default();
    device = g_object_new(LOCATION_TYPE_GPS_DEVICE, NULL);

    g_object_set(G_OBJECT(control),
        "preferred-method", LOCATION_METHOD_USER_SELECTED,
        "preferred-interval", LOCATION_INTERVAL_DEFAULT,
        NULL);

    g_signal_connect(control, "error-verbose", G_CALLBACK(on_error), loop);
    g_signal_connect(device, "changed", G_CALLBACK(on_changed), control);
    g_signal_connect(control, "gpsd-stopped", G_CALLBACK(on_stop), loop);

    g_idle_add(start_location, control);

    g_main_loop_run(loop);

    g_object_unref(device);
    g_object_unref(control);

    return 0;
 }
```

You can compile this example with following command:

```
 gcc -Wall `pkg-config --cflags glib-2.0, liblocation --libs glib-2.0,
liblocation` -o test test.c
```