Технологии QNX и КПДА в России

Санкт-Петербург, 23 ноября 2016

«Применение 3D технологий для решения задачи визуализации информации в ЗОСРВ «Нейтрино»

Александр Молодцов, СВД Встраиваемые Системы

СВД Встраиваемые Системы

SWD Software

Содержание

- **Обзор библиотеки GL**

- Управление конфигурациями буфера кадров
- Создание 3D контекста и областей отображения
- Пример: рендеринг в Photon окружении

OpenGL (Open Graphics Library) — *спецификация, определяющая платформонезависимый программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику.*

Mesa — *реализация графического* **API OpenGL**. **Mesa** *ориентирована на обеспечение высокой производительности, в том числе за счёт использования аппаратного ускорения работы с графикой, поддерживаемого видеоадаптерами.*

GF3D — *расширение* **Graphics Framework API**, реализующее интерфейс между **OpenGL** и **GF**.

Функции **OpenGL** *и* **GF3D** *реализованы в библиотеке* **libGL.so**.

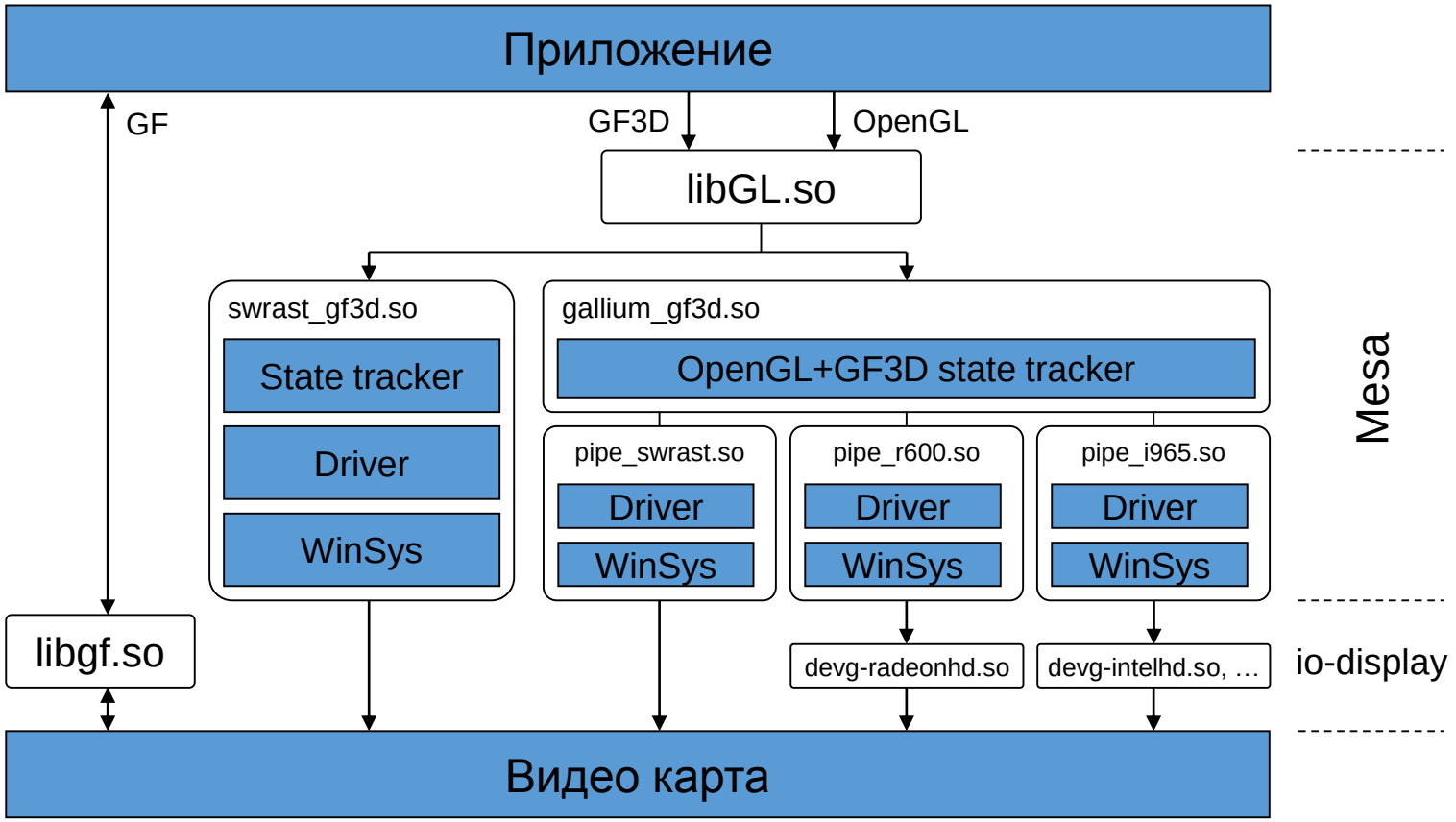
Конвейеры и драйверы **Mesa** *представлены в виде разделяемых библиотек, загрузка которых осуществляется автоматически библиотекой* **libGL.so**.

Заголовочный файл:

GL/glqnx.h

Переменные окружения:

GF3D_VERBOSE	*устанавливает уровень подробности отладочного вывода*
GF3D_DRIVER_PATH	*устанавливает путь поиска драйверов*
	по умолчанию: /opt/mesa3d/<архитектура>/lib/mesa
GF3D_DRIVER	*принудительно устанавливает загружаемый драйвер*


Приложение
GF
GF3D
OpenGL
libGL.so
swrast_gf3d.so
State tracker
Driver
WinSys
gallium_gf3d.so
OpenGL+GF3D state tracker
pipe_swrast.so
Driver
WinSys
pipe_r600.so
Driver
WinSys
pipe_i965.so
Driver
WinSys
Mesa
libgf.so
devg-radeonhd.so
devg-intelhd.so, …
io-display
Видео карта

Содержание

- Обзор библиотеки GL

- **Управление конфигурациями буфера кадров**

- Создание 3D контекста и областей отображения
- Пример: рендеринг в Photon окружении

Получение списка поддерживаемых конфигураций

Конфигурация – *внутренняя структура расширения* **GF3D**, *определяющая параметры фреймбуфера будущего 3D контекста. Выбор конфигурации напрямую затрагивает функциональные возможности* **Mesa**.

gf3d_config_t * gf3d_get_configs(gf_dev_t gdev, int * nelements);

Параметры:

gdev	*дескриптор устройства (см. функцию* **gf_dev_attach()***)*
nelements	*количество элементов в результирующем массиве*

Функция возвращает список всех доступных конфигураций для данного устройства. Функция выделяет блок памяти, который может быть освобожден вручную при помощи **free()**. *Если* **gdev** *равен* **NULL**, *функция вернет* **NULL**, *передача заведомо некорректного* **gdev** *с большой вероятностью приведет к аварийному завершению процесса или сбоям в работе* **Mesa**.

gf3d_config_t → gf3d_context_t → gf3d_drawable_t → gf_surface_t

Получение списка предпочтительных конфигураций

gf3d_config_t * gf3d_choose_config(gf_dev_t gdev, const int * attrib_list, int * nelements);

Параметры:

gdev	*дескриптор устройства (см. функцию* **gf_dev_attach()***)*
attrib_list	*массив пар атрибут/значение, последний атрибут должен быть равен* **GF3D_NONE** *или* **0***.*
nelements	*количество элементов в результирующем массиве*

Функция возвращает список всех конфигураций, удовлетворяющих массиву атрибутов. Список конфигураций отсортирован от наиболее к наименее подходящим. Функция выделяет блок памяти, который может быть освобожден вручную при помощи **free()**. **NULL** *возвращается если* **gdev** *равен* **NULL** *или список атрибутов не соответствует ни одной конфигурации. Если часть/все атрибуты опущены, используются значения по умолчанию (список атрибутов представлен далее). Передача заведомо некорректного* **gdev** *с большой вероятностью приведет к аварийному терминированию процесса или сбоям в работе* **Mesa***.*

gf3d_config_t → gf3d_context_t → gf3d_drawable_t → gf_surface_t

Получение значений атрибутов для определенной конфигурации

int gf3d_ get_config_attrib(gf3d_config_t config, int attribute, int *value);

Параметры:

config	*элемент списка конфигураций, полученного ранее*
attribute	*идентификатор атрибута (список атрибутов представлен далее)*
value	*текущее значение*

Функция возвращает значение атрибута указанной конфигурации. Необходимо помнить, что вызов **gf3d_choose_config***() не изменяет значения атрибутов конфигураций и используется только для поиска наиболее подходящих. При успешном завершении возвращается* **0***, в противном случае возвращается код ошибки:* **GF3D_NO_EXTENSION** *(данное устройство не поддерживает расширение* **GF3D***) или* **GF3D_BAD_ATTRIBUTE** *(указан некорректный атрибут или* **config** *содержит адрес* **NULL***).*

gf3d_config_t → gf3d_context_t → gf3d_drawable_t → gf_surface_t

Список поддерживаемых атрибутов конфигураций

Атрибут конфигурации – *характерный для данной конфигурации параметр, напрямую определяющий функциональность библиотеки* **Mesa**. *При выборе конфигурации одни атрибуты определяются точными значениями, другие могут представлять верхнюю или нижнюю границу разрешенных значений.*

GF3D_RENDER_TYPE	*поддерживаемые режимы рендеринга в виде битовой маски. Маска может содержать следующие биты:* **GF3D_RGBA_BIT** *(режим по умолчанию) и* **GF3D_COLOR_INDEX_BIT** *(индексированные цвета).*
GF3D_BUFFER_SIZE	*нижняя граница глубины цвета для индексированного цвета (если не установлен бит* **GF3D_COLOR_INDEX_BIT** *атрибут игнорируется). Неотрицательное целое число, значение по умолчанию* **0**.
GF3D_SAMPLE_BUFFER	*отключение (***0***, по умолчанию) / включение (не* **0***) мультисэмплинга.*

gf3d_config_t → gf3d_context_t → gf3d_drawable_t → gf_surface_t

Список поддерживаемых атрибутов конфигураций

GF3D_RED_SIZE
GF3D_GREEN_SIZE
GF3D_BLUE_SIZE
GF3D_ALPHA_SIZE

нижняя граница покомпонентной глубины цвета. Если для любого из этих атрибутов указано значение **0** *или* **GF3D_DONT_CARE**, *то при поиске конфигурации данный атрибут учитываться не будет. Неотрицательные целые числа, значение по умолчанию* **0**.

GF3D_DEPTH_SIZE

нижняя граница разрядности значений элементов в буфере глубины. Если значение равно **0**, *при поиске предпочтительными считаются конфигурации без буфера глубины. Неотрицательное целое число, значение по умолчанию* **0**.

GF3D_STENCIL_SIZE

нижняя граница разрядности значений элементов в буфере шаблона. Предпочтительными считаются конфигурации с наименьшим буфером шаблона. Если значение равно **0**, *предпочтительными являются конфигурации без буфера шаблона. Неотрицательное число, значение по умолчанию* **0**.

gf3d_config_t → gf3d_context_t → gf3d_drawable_t → gf_surface_t

Список поддерживаемых атрибутов конфигураций

GF3D_ACCUM_RED_SIZE
GF3D_ACCUM_GREEN_SIZE
GF3D_ACCUM_BLUE_SIZE
GF3D_ACCUM_ALPHA_SIZE

нижняя граница покомпонентной глубины цвета буфера аккумулятора. Если для любого из этих атрибутов указано значение **0**, *то предпочтительными считаются конфигурации без буфера аккумулятора, в противном случае осуществляется поиск конфигурации с максимальным размером аккумулятора. Неотрицательные целые числа, значение по умолчанию* **0**.

GF3D_SAMPLES

нижняя граница предпочтительного числа сэмплов при мультисэмплировании. Конфигурации с наименьшим количеством сэмплов считаются предпочтительными. Неотрицательное целое число, значение по умолчанию **0**.

gf3d_config_t → gf3d_context_t → gf3d_drawable_t → gf_surface_t

Содержание

- Обзор библиотеки GL
- Управление конфигурациями буфера кадров

- **Создание 3D контекста и областей отображения**

- Пример: рендеринг в Photon окружении

Создание контекста

Контекст – *это окружение, в рамках которого работает машина состояний* **OpenGL** *и* **GF3D**.

gf3d_context_t gf3d_create_context(gf_dev_t gdev, gf3d_config_t config, gf3d_context_t share_context, const int *attrib_list);

Параметры:

gdev	*дескриптор устройства (см. функцию* **gf_dev_attach()***)*
config	*элемент списка конфигураций, полученного ранее*
share_context	*контекст с которым разделяются списки отображения или* **NULL**
attrib_list	*массив пар атрибут/значение , последний атрибут должен быть равен GF3D_NONE*

Функция создает новый контекст рендеринга и возвращает его дескриптор или **NULL** *в случае ошибки. Массив атрибутов позволяет задать желаемую версию* **OpenGL API** *посредством атрибутов* **GF3D_CONTEXT_MAJOR_VERSION** *и* **GF3D_CONTEXT_MIN**.

gf3d_config_t → **gf3d_context_t** → gf3d_drawable_t → gf_surface_t

Освобождение контекста

void gf3d_destroy_context(gf3d_context_t ctx);

Параметры:

ctx *контекст для освобождения*

Функция уничтожает выбранный контекст если он не используется в настоящий момент, в противном случае контекст будет уничтожен сразу же после освобождения (см. **gf3d_make_current()***.*

gf3d_context_t gf3d_get_current_context(void);

Функция возвращает текущий активный контекст. Если активный контекст отсутствует, будет возвращено значение **NULL***.*

Создание области отображения

Область отображения – *контейнер для передачи поверхности* **GF** *в библиотеку* **Mesa**.

gf3d_drawable_t gf3d_create_gf_drawable(gf_dev_t gdev, gf3d_config_t config, gf_surface_t *surfaces, int nsurfaces, int width, int height);

Предыдущее название: **gf3d_create_gf_target()**.

Параметры:

gdev	*дескриптор устройства (см. функцию* **gf_dev_attach()***)*
config	*элемент списка конфигураций, полученного ранее*
surfaces	*массив поверхностей, используемых для рендеринга*
nsurfaces	*количество используемых поверхностей (поддерживается только* **1***)*
width	*ширина области рендеринга (в настоящий момент не используется)*
height	*высота области рендеринга (в настоящий момент не используется)*

Функция возвращает дескриптор созданной области отображения на основе переданных поверхностей и требуемой конфигурации. В случае ошибки возвращается значение **NULL**.

gf3d_config_t → gf3d_context_t → **gf3d_drawable_t → gf_surface_t**

Освобождение области отображения

void gf3d_destroy_drawable(gf3d_drawable_t drawable);

Предыдущее название: **gf3d_destroy_target()**.

Параметры:

drawable *область отображения, подлежащая освобождению*

Функция уничтожает выбранную область отображения если она не используется в настоящий момент, в противном случае область будет уничтожена сразу же после освобождения (см. **gf3d_make_current()**).

gf3d_drawable_t gf3d_get_current_target(void);

Функция возвращает основную область отображения, выбранную через **gf3d_make_current()** *и* **NULL** *в противном случае.*

gf3d_config_t → gf3d_context_t → **gf3d_drawable_t → gf_surface_t**

Выбор текущего контекста рендеринга и основной области отображения

Основная область отображения – *поверхность, в которую* **Mesa** *выводит результирующее изображение. Также эту область отображения часто именуют* **target**.

GLboolean gf3d_make_current(gf3d_context_t ctx, gf3d_drawable_t target);

Параметры:

ctx	*выбранный контекст, устанавливаемый в качестве текущего*
target	*основная область отображения*

Функция изменяет активный контекст и его основную область отображения, все последующие **OpenGL** *вызовы будут использовать новый контекст.* ***Активный контекст рендеринга общий для всего процесса****. Запланированные предыдущим контекстом команды выполняются перед его освобождением. Первая установка контекста приводит его поле вывода (***viewport***) к полному размеру области отображения, последующие выводы этот параметр не изменяют. Передача* **NULL** *для обоих параметров переводит контекст и область отображения в разряд неиспользуемых. Возвращается* **GL_TRUE** *в случае успеха или* **GL_FALSE** *(текущий контекст не меняется).*

gf3d_config_t → **gf3d_context_t → gf3d_drawable_t → gf_surface_t**

Содержание

- Обзор библиотеки GL
- Управление конфигурациями буфера кадров
- Создание 3D контекста и областей отображения

- **Пример: рендеринг в Photon окружении**

Создание контекста GF3D

```
/* подключение к адаптеру */
gf_dev_attach( &gdev, GF_DEVICE_INDEX(0), NULL );

/* желаемые атрибуты конфигурации */
int conf_attrs[] = {  GF3D_RED_SIZE,     1,  GF3D_GREEN_SIZE,  1,
                      GF3D_BLUE_SIZE,    1,  GF3D_ALPHA_SIZE,  1,
                      GF3D_DEPTH_SIZE,   1,  GF3D_NONE };

/* поиск подходящей конфигурации */
configs = gf3d_choose_config( gdev, conf_attrs, &ncfgs );

/* создание контекста GF3D */
ctx = gf3d_create_context( gdev, configs[0], NULL, NULL );
```

Создание поверхности для рендеринга и активация контекста

```
/* создание GF поверхности */
gf_surface_create( &gf_surface, gdev, width, height, GF_FORMAT_BGRA8888, NULL,
                   GF_SURFACE_CREATE_3D_ACCESSIBLE |
                   GF_SURFACE_CREATE_SHAREABLE );

/* получение offscreen-контекста Photon для созданной поверхности */
ph_surf = PdCreateOffscreenContextGF( gf_surface );

/* создание области рисования GF3D для данной поверхности */
target = gf3d_create_gf_drawable( gdev, configs[0], &gf_surface, 1, -1, -1 );

/* Активация контекста (назначение контекста рендеринга и основной */
/* области отображения) */
gf3d_make_current( ctx, target );
```

Вывод изображения в виджет PtRaw

```
… /* команды рисования */

/* отправка команд на исполнение и ожидание окончания выполнения */
glFinish();
/* помечаем виджет для перерисовки */
PtDamageWidget( draw_area );

/* процедура рисования виджета PtRaw */
void raw_draw_f( PtWidget_t *widget, PhTile_t *damage ) {
        PhRect_t *dst_rect = PtGetCanvas( widget );
        PhRect_t   src_rect = { {0, 0}, {width - 1, height - 1} };       /* GF поверхность */
        PgContextBlit( ph_surf, &src_rect, NULL, dst_rect );
        /* отправка команд на исполнение и ожидание завершения */
        PgFFlush( Ph_DONE_DRAW );
        PgWaitHWIdle();
}
```

Спасибо за внимание

Александр Молодцов
Инженер-программист

(812) 346-89-56 доп. 104
support@kpda.ru

www.kpda.ru
www.swd.ru

СВД Встраиваемые Системы

SWD Software