SWD Software

Специализированный семинар
«День Технологий QNX»

СВД Встраиваемые Системы

Андрей Докучаев, СВД Встраиваемые Системы

Особенности разработки OpenGL ES приложений в QNX

Краткое содержание

- **Разработка OpenGL ES приложений в ОСРВ QNX Neutrino с использованием программного интерфейса EGL**
- Различия программных интерфейсов OpenGL ES версий 1.x и 2.0
- Использование Graphics Framework для создания композиции
  - Создание композиции с использованием Chroma ключа
  - Настройка окна обзора
- Пример графического приложения

Назначение EGL

EGL – интерфейс между OpenGL ES / OpenVG и графической подсистемой ОС.
Стандарт разрабатывается консорциумом Khronos с 2002 года.

Сферы ответственности EGL:

1. **Управление поверхностями в видео памяти;**
2. **Создание графических контекстов;**
3. **Обеспечение синхронизации при рендеринге;**
4. **Обеспечивает эффективный обмен данными между различными API, например, между видео подсистемой OpenMAX AL и графическим окружением OpenGL ES.**

KHRONOS GROUP
CONNECTING SOFTWARE TO SILICON

EGL

Совместимость стандартов

Стандарт EGL 1.2

1. display_id (аргумент функции eglGetDisplay()) соответствует описателю устройства библиотеки GF: gf_device_t handle.
2. Совместим с технологией QNX Graphics Framework.
3. Поддерживает OpenGL ES 1.x.

Стандарт EGL 1.4

1. аргумент display_id является независимым целочисленным идентификатором.
2. Совместимость с QNX GF не полная.
3. Поддерживает OpenGL ES 2.0.

Библиотеки QNX

libEGL.so.1 – EGL 1.4
libGLES_CL.so.1 – EGL 1.2 и OpenGL ES 1.0 – минималистичный профиль
libGLES_CM.so.1 – EGL 1.2 и OpenGL ES 1.0 – стандартный профиль

Основы использования EGL

Инициализация интерфейса

```
egl_disp = eglGetDisplay( display_id );
if ( egl_disp == EGL_NO_DISPLAY ) {
    egl_perror( "eglGetDisplay" );
    ...
}
```

Выбор дисплея для отрисовки сцены

```
if ( eglInitialize( egl_disp, NULL, NULL ) != EGL_TRUE ) {
    egl_perror( "eglInitialize" );
    ...
}
```

Инициализация интерфейса

Основы использования EGL

Управление поверхностями

Создание отображаемой поверхности

Слой

```
egl_surf = eglCreateWindowSurface( egl_disp, egl_conf, layer, (EGLint*)&surf_attr );
if (egl_surf == EGL_NO_SURFACE ) egl_perror( "eglCreateWindowSurface" );
```

Создание контекста

```
egl_ctx = eglCreateContext( egl_disp, egl_conf, EGL_NO_CONTEXT, (EGLint*)&attr );
if ( egl_ctx == EGL_NO_CONTEXT ) egl_perror( "eglCreateContext" );
```

Выбор поверхности в качестве активной

```
if ( eglMakeCurrent(egl_disp, egl_surf, egl_surf, egl_ctx) != EGL_TRUE )
    egl_perror( "eglMakeCurrent" );
```

Ассоциированный контекст

Основы использования EGL

Синхронизация буферов

Устанавливается режим синхронизации сцены с буфером кадров

Если интервал сброса буферов равен 0 сцена будет отображаться асинхронно - по мере завершения итераций рендеринга

```
if ( eglSwapInterval(egl_disp, interval) != EGL_TRUE ) egl_perror( "eglSwapInterval" );

while ( 1 ) {
    ...

    eglSwapBuffers( egl_disp, egl_surf );
}
```

Сброс буферов, ассоциированных с поверхностью egl_surf. В зависимости от движка рендеринга будет вызвана либо функция glFlush(), либо vgFlush().

Краткое содержание

- Разработка OpenGL ES приложений в ОСРВ QNX Neutrino с использованием программного интерфейса EGL
- **Различия программных интерфейсов OpenGL ES версий 1.x и 2.0**
- Использование Graphics Framework для создания композиции
  - Создание композиции с использованием Chroma ключа
  - Настройка окна обзора
- Пример графического приложения

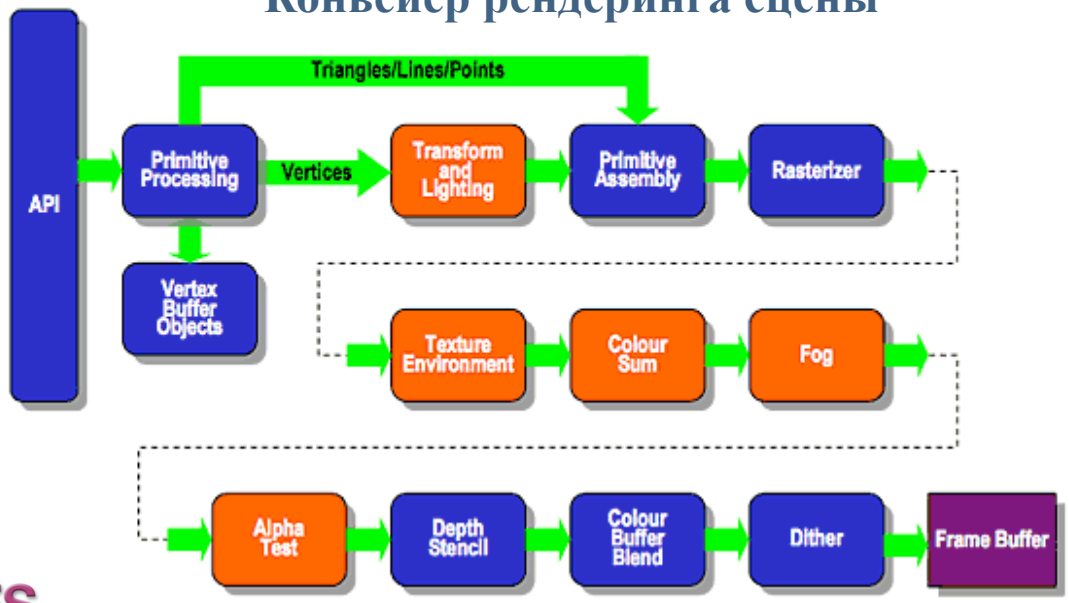
Стандарт OpenGL ES 1.x

Стандарт OpenGL ES 1.0 основан на OpenGL 1.3, при этом существенная часть функционала оригинала по архитектурным соображениям отсутствует.

Например, не поддерживается:

1. **моделирование объектов на основе прямоугольников и полигонов;**
2. **генерация координат текстур;**
3. **буфер аккумулирования;**
4. **работа с битовыми картами;**
5. **3D текстуры;**
6. **контейнер glBegin()…glEnd().**

Конвейер рендеринга сцены



KHRONOS GROUP
CONNECTING SOFTWARE TO SILICON

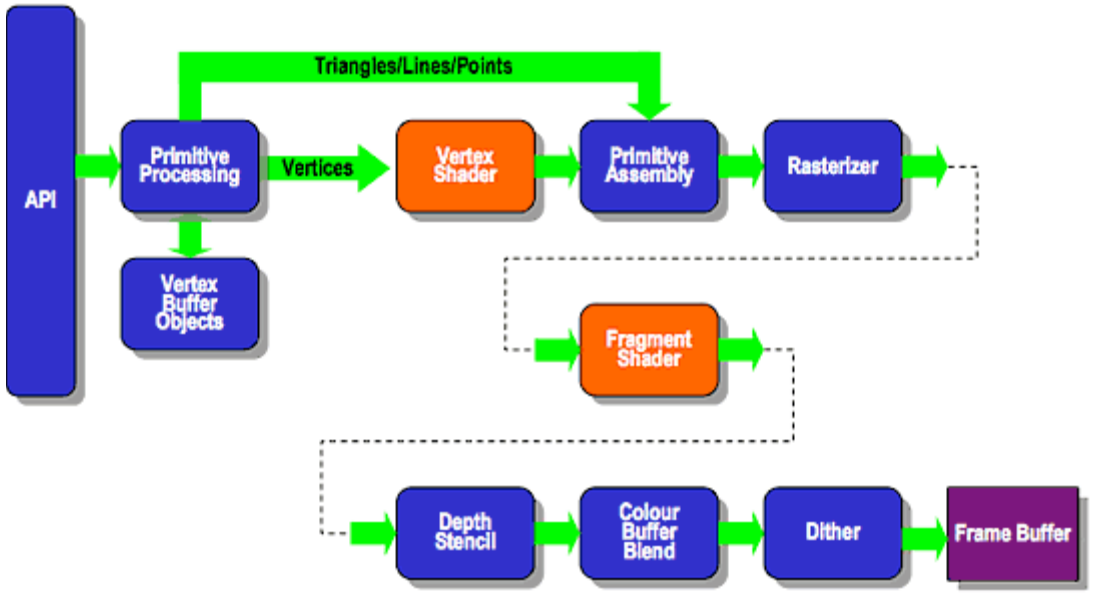
OpenGL|ES

Различия программных интерфейсов OpenGL ES версий 1.x и 2.0

Стандарт OpenGL ES 2.0

Стандарт опубликован в марте 2007 и базируется на спецификации OpenGL 2.0.

Конвейер рендеринга OpenGL ES 2.0



Особенности OpenGL ES 2.0:

1. отсутствует обратная совместимость с OpenGL (до OpenGL 4.1) и OpenGL ES 1.x;
2. поддержан язык GLSL, из API исключен весь легко реализуемый с помощью шейдеров функционал;
3. упрощен конвейер рендеринга (стадии рендеринга программируются шейдерами).

KHRONOS GROUP
Connecting Software to Silicon
OpenGL|ES

Краткое содержание

- Разработка OpenGL ES приложений в ОСРВ QNX Neutrino с использованием программного интерфейса EGL
- Различия программных интерфейсов OpenGL ES версий 1.x и 2.0
- **Использование Graphics Framework для создания композиции**
  - **Создание композиции с использованием Chroma ключа**
  - **Настройка окна обзора**
- Пример графического приложения

Создание композиции с использованием Chroma ключа

Изображение на черном фоне (слой №1). Chroma ключ не настроен.

Осуществление композиции изображений на различных слоях с применением Chroma ключа

Изображение на черном фоне (слой №1), Photon Micro GUI (слой №0). Для слоя №1 настроен Chroma ключ.

Инструментарий?

OpenGL ES 1.x → gf3d.h

OpenGL ES 2.0 → ?

Использование Chroma ключа в OpenGL ES 2.0 при помощи библиотеки GF

```
gf_chroma_t    chroma;
chroma.color0 = 0xff0000;
chroma.mode = GF_CHROMA_OP_NO_DRAW | GF_CHROMA_OP_SRC_MATCH;

while ( 1 ) {
   ...
   eglSwapBuffers( … );
   …
   gf_layer_set_chroma( layer, &chroma );
   gf_layer_update( layer, GF_LAYER_UPDATE_NO_WAIT_VSYNC );
}
```

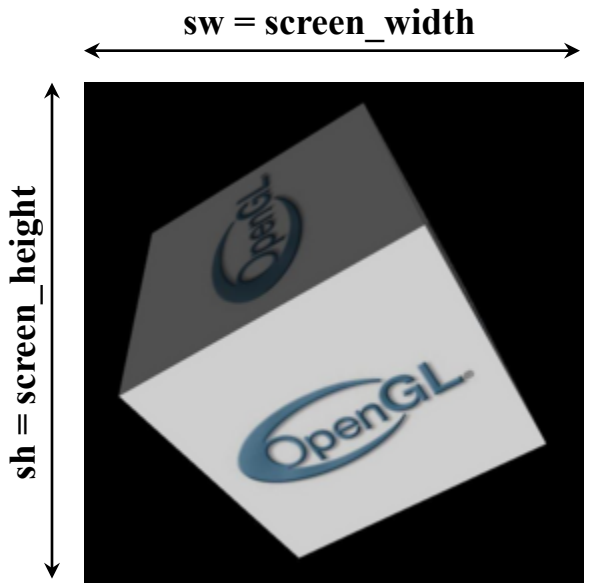
Маска цвета

Маска операции:
1. **исключать пиксели, чей цвет равен color0;**
2. **анализировать цвет изображения-источника (layer).**

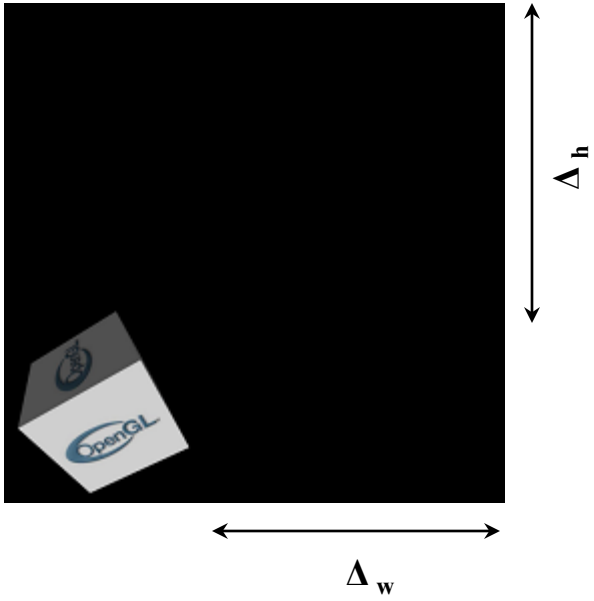
Корректировка Chroma ключа будет осуществляться при каждой отрисовке сцены

Настройка окна обзора

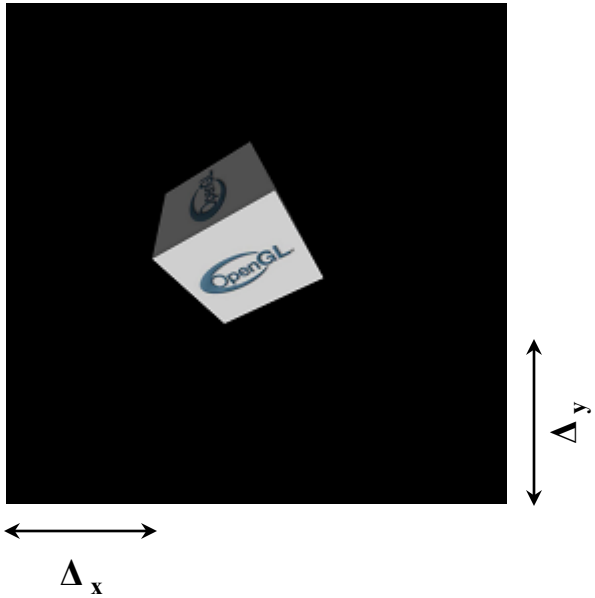
Вариант 1: настройка средствами OpenGL ES



glViewport(0, 0, sw, sh);



glViewport(0, 0, sw - Δ_w, sh - Δ_h);

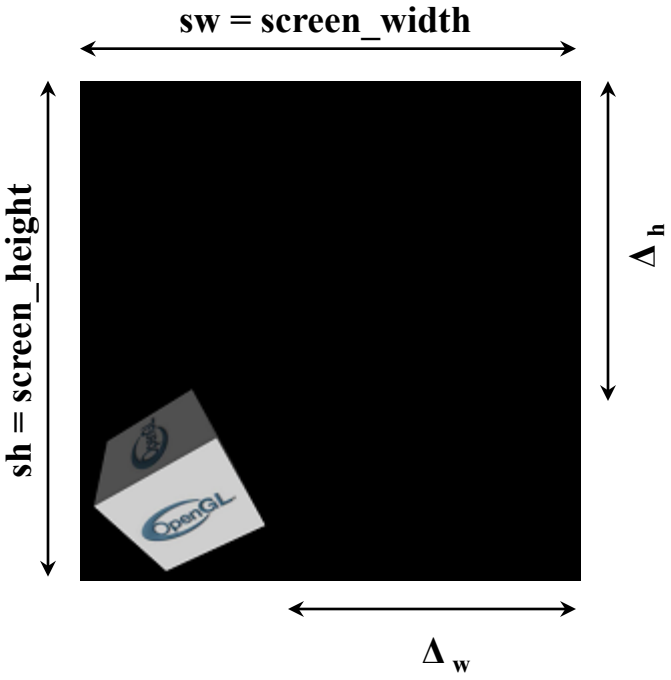


glViewport(Δ_x, Δ_y, sw - Δ_w, sh - Δ_h);

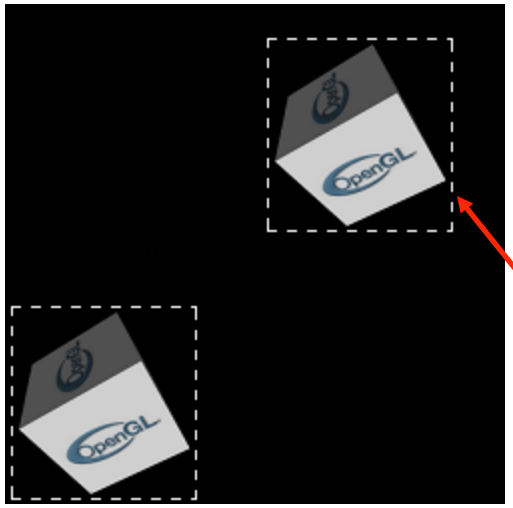
Использование Graphics Framework для создания композиции

Настройка окна обзора

Вариант 2: создание окна обзора на уровне слоя средствами Graphics Framework



glViewport(0, 0, sw - Δ w , sh - Δ h);



Окно приемник (отображаемая слоем область)

Окно источник (характеризует графический объект)

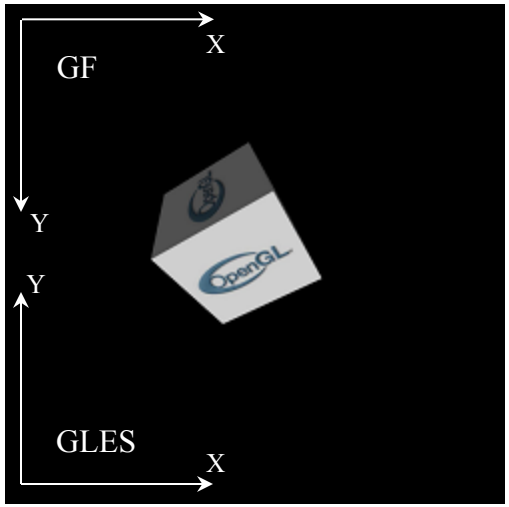
Настройка окна обзора

Вариант 2: создание окна обзора на уровне слоя средствами Graphics Framework

Средства GF API для работы с окном обзора:

```
gf_layer_set_src_viewport( layer, x0, y0, x1, y1 );
gf_layer_set_dst_viewport( layer, x0, y0, x1, y1 );
```

Важно! В общем случае недопустимо задавать пересекающиеся в видео памяти области, характеризующие окно источник и окно приемник. Нарушение данной рекомендации в ряде случаев может привести к аппаратному сбою контроллера.



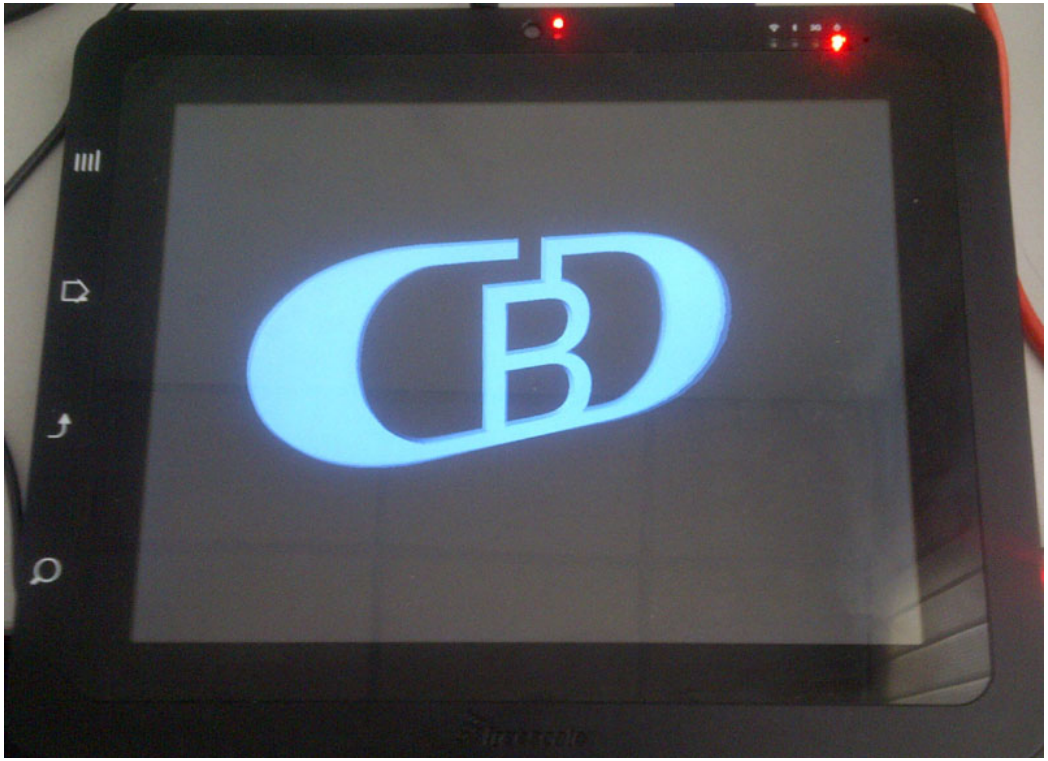
Краткое содержание

- Разработка OpenGL ES приложений в ОСРВ QNX Neutrino с использованием программного интерфейса EGL
- Различия программных интерфейсов OpenGL ES версий 1.x и 2.0
- Использование Graphics Framework для создания композиции
  - Создание композиции с использованием Chroma ключа
  - Настройка окна обзора
- **Пример графического приложения**

Простейшее графическое приложение

Этапы подготовки сцены :

1. инициализация интерфейса EGL, выбор слоя;
2. инициализация OpenGL ES;
3. загрузка полигональной модели в формате 3DS;
4. загрузка вершинного и фрагментарного шейдеров;
5. рендеринг сцены, содержащей вращающийся объект, с учетом синхронизации.



На базе имеющегося приложения осуществим интеграцию с окружением Photon…

Шаг 1. Масштабирование сцены

1.1. Масштабирование осуществляется средствами OpenGL ES…

```
glViewport( 0, 0, sw-Δw, sh-Δh );
```

1.2. Присоединение к приложению слоя средствами GF…

```
gf_layer_attach( ...,
    GF_LAYER_ATTACH_PASSIVE );
```

Внимание! Слой должен быть подключен именно пассивно, так как владельцем слоя является библиотека OpenGL ES.

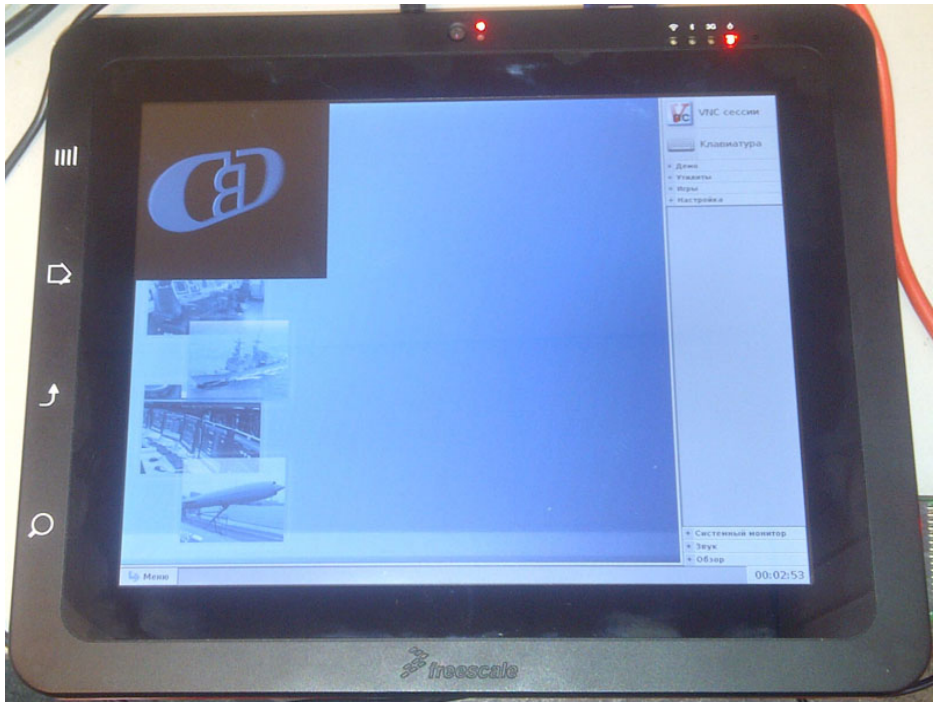
Шаг 2. Позиционирование объекта на экране

2.1. Создание окна обзора…

```
gf_layer_set_src_viewport( layer, 0,
  disp_info.yres-height, width-1,
  disp_info.yres-1 );
gf_layer_set_dst_viewport( layer,
  xpos, ypos, xpos+width-1,
  ypos+height-1);
```

2.2. Обновление конфигурации слоя в контексте основного цикла рендеринга сцены…

```
while ( 1 ) {
  ...
  eglSwapBuffers( … );
  ...
  gf_layer_update( layer,
    GF_LAYER_UPDATE_NO_WAIT_VSYNC );
}
```



Шаг 3. Использование Chroma ключа

3.1. Задание цвета фона, который будет считаться прозрачным…

```
glClearColor( 1.0, 0.0, 0.0, 1.0 );
glClear( GL_COLOR_BUFFER_BIT |
         GL_DEPTH_BUFFER_BIT);
```

Выбранный цвет не должен совпадать с каким-либо цветом, используемым в модели.

Шаг 3. Использование Chroma ключа

3.1. Настройка слоя в соответствии с желаемым Chroma ключем…

```
memset( &chroma, 0, sizeof chroma );
chroma.color0 = 0xff0000;
chroma.mode = GF_CHROMA_OP_NO_DRAW |
              GF_CHROMA_OP_SRC_MATCH;
...
while ( 1 ) {
  eglSwapBuffers( egl_disp,
                  egl_surf );
  ...
  gf_layer_set_chroma( layer,
                       &chroma );
}
```

Спасибо за внимание.

СВД Встраиваемые Системы

www.kpda.ru forum.kpda.ru

sales@kpda.ru support@kpda.ru

Центральный офис:

196066 Санкт-Петербург

Московский проспект, 212А

тел.: (812)373-41-17

факс:(812)373-19-07

Технический офис:

191014 Санкт-Петербург

ул.Госпитальная, д.3

тел./факс:(812)578-02-45