Driver-Inter Ltd.

D3DBase9 User Guide

Или как пользоваться программой, любезно предоставленной преподавателем

Владимир Беляев

15.09.2006

Введение

Представленная программа может быть разделена на две логические части: библиотеку и приложение, использующую эту библиотеку. При использовании рекомендуется по возможности не модифицировать библиотеку (это поможет в будущем быстро и просто заменять ее на более современные версии). Изменениям (или полной замене) подлежит только приложение (main.cpp, myApp.*).

Функциональность

Настоящая версия D3DBase создает и удаляет Direct3D9 и Direct3DDevice9. В качестве самого простого render действия выступает очистка target buffer'а некоторым цветом.

Приложение (myApp) обрабатывает mouse move (with left button pressed), mouse wheel и кнопки Left, Right, Up, Down, +, -. Эти кнопки предназначены для вращения камеры, но пока что используются для изменения цвета очистки target buffer'а.

Приложение также считает FPS (frames per second) и выводит его в caption окна.

Использование

Библиотека состоит из трех классов.

cglApp

Главный – **cglApp** – представляет собой класс базового приложения реализующий:

- Создание/удаление окна
- Создание/удаление cglD3D
- Базовую обработку input'а
- Расчет и вывод FPS
- cglD3D::beginRender() - cglD3D::clear() - cglD3D:: endRender() calls

Использовать **cglApp** предполагается через наследование от него с реализацией или перекрытием виртуальных методов:

`virtual bool processInput(unsigned int nMsg, int wParam, long lParam)` – в этот метод передаются windows сообщения. Метод базового класса обрабатывает сообщение ESC pressed. Метод возвращает true, когда решает, что приложению пора завершаться.

`virtual void update()` в этом методе следует обрабатывать изменения в сцене. Метод базового класса update'ит timer и FPS.

`virtual void renderInternal()` – в этом методе следует рендерить сцену.

`virtual char const *getWindowText()` – этот метод возвращает название приложения. Если вы хотите вывести название, отличающееся от "D3D Labs Basic App." – метод следует перекрыть и вернуть `char const*` на свою строчку.

Пользователю cglApp также доступны protected поля:

`m_hWnd` – HWND окна

`m_hInstance` – HINSTANCE приложения

`m_nClearColor` – цвет очистки frame buffer'а

`m_pD3D` – указатель на класс D3D

`m_timer` - `cglTimer;` таймер

Пример наследования от **cglApp** можно обнаружить в myApp.cpp, h.

cglD3D

Перейдем к рассмотрению класса **cglD3D**, указатель на который доступен наследникам **cglApp.** На настоящий момент **cglD3D** является простейшей оберткой вокруг IDirect3D9 и IDirect3DDevice9. **cglD3D** на конструкторе создает IDirect3D9 и IDirect3DDevice9, на деструкторе – убивает их.

`cglD3D::beginRender()` == IDirect3DDevice9::BeginScene()

`cglD3D::endRender()` == IDirect3DDevice9::EndScene() + IDirect3DDevice9::Present()

`cglD3D::clear()` == IDirect3DDevice9::Clear()

`cglD3D::getDevice()` позволяет получить IDirect3DDevice9*, который нужен для rendering'a.

cglTimer

Ну и напоследок посмотрим, что такое **cglTimer**. По факту это таймер с высоким разрешением, использующий виндозные QueryPerformanceFrequency(), QueryPerformanceCounter().

`float cglTimer::getDelta() const` – возвращает время (в секундах), прошедшее между двумя последними вызовами `cglTimer::update`(), который вызывается в `cglApp::update`(). Следовательно, можно считать, что это время, прошедшее между предыдущим и настоящим кадром.

`float cglTimer::getTime() const` – возвращает время (в секундах), прошедшее со старта приложения (если точнее – с создания `cglTimer`).