Глава 9. Шрифты

В ходе игры нам часто требуется показать пользователю какую-либо текстовую информацию. В этой главе мы рассмотрим три способа отображения текста, поддерживаемые Direct3D. Каждый способ иллюстрируется примером программы, находящимся на сайте этой книги и в сопроводительных файлах.

Цели

- Узнать, как можно визуализировать текст с помощью интерфейса ID3DXFont.
- Узнать, как можно визуализировать текст с помощью класса CD3DFont.
- Узнать, как вычислить частоту кадров приложения.
- Узнать, как создать и визуализировать трехмерный текст с помощью функции D3DXCreateText.

9.1. ID3DXFont

Библиотека D3DX предоставляет интерфейс ID3DXFont, который можно использовать для отображения текста в Direct3D-приложениях. Этот интерфейс использует для отображения текста GDI и его применение может значительно уменьшить быстродействие приложения. Однако, поскольку ID3DXFont использует GDI, он поддерживает сложные шрифты и форматирование.

9.1.1. **Создание** ID3DXFont

Для создания интерфейса ID3DXFont может использоваться функция D3DXCreateFontIndirect.

```
HRESULT D3DXCreateFontIndirect(
    LPDIRECT3DDEVICE9 pDevice, // устройство, связанное со шрифтом
    CONST LOGFONT* pLogFont,   // структура LOGFONT, описывающая шрифт
    LPD3DXFONT* ppFont         // возвращает созданный шрифт
);
```

Приведенный ниже фрагмент кода показывает использование этой функции:

```
LOGFONT lf;
ZeroMemory(&lf, sizeof(LOGFONT));
lf.lfHeight    = 25;    // в логических единицах
lf.lfWidth     = 12;    // в логических единицах
lf.lfWeight    = 500;   // насыщенность,
                        // диапазон 0(тонкий) - 1000(жирный)
lf.lfItalic    = false;
lf.lfUnderline = false;
lf.lfStrikeOut = false;
lf.lfCharSet   = DEFAULT_CHARSET;
strcpy(lf.lfFaceName, "Times New Roman"); // гарнитура шрифта
ID3DXFont* font = 0;
D3DXCreateFontIndirect(Device, &lf, &font);
```

Обратите внимание, что сперва требуется заполнить структуру LOGFONT, которая описывает параметры создаваемого шрифта.

Для получения указателя на интерфейс ID3DXFont вы можете также воспользоваться функцией D3DXCreateFont.

9.1.2. **Рисование текста**

После того, как мы получили указатель на интерфейс ID3DXFont, рисование текста осуществляется простым вызовом метода ID3DXFont::DrawText.

```
INT ID3DXFont::DrawText(
    LPCSTR pString,
    INT Count,
    LPRECT pRect,
    DWORD Format,
    D3DCOLOR Color
);
```

- pString — Указатель на отображаемую строку текста.
- Count — Количество отображаемых символов строки. Если строка завершается нулевым символом можно указать –1, чтобы строка отображалась вся.
- pRect — Указатель на структуру RECT, определяющую область экрана в которой будет отображаться текст.
- Format — Необязательные флаги, определяющие форматирование выводимого текста; их описание находится в документации к SDK.
- Color — Цвет текста.

Вот пример использования метода:

```
Font->DrawText(
    "Hello World",    // Выводимая строка
    -1,               // Строка завершается нулевым символом
    &rect,            // Прямоугольная область для рисования строки
    DT_TOP | DT_LEFT, // Рисуем в верхнем левом углу области
    0xff000000);      // Черный цвет
```

9.1.3. **Вычисление частоты кадров**

Примеры приложений к этой главе ID3DXFont и CFont вычисляют и отображают количество визуализируемых за секунду кадров (FPS). В этом разделе мы покажем как вычисляется FPS. Сперва мы объявляем три глобальных переменных:

```
DWORD FrameCnt;    // Количество выведенных кадров
float TimeElapsed; // Прошедшее время
float FPS;         // Частота визуализации кадров
```

Мы вычисляем FPS каждую секунду; это дает нам достоверное среднее значение. Кроме того, мы храним вычисленное значение частоты кадров в течение одной секунды, что дает достаточно времени, чтобы прочитать его перед очередным изменением.

Итак, каждый кадр мы увеличиваем значение переменной FrameCnt и прибавляем к переменной TimeElapsed время, прошедшее с вывода предыдущего кадра:

```
FrameCnt++;
TimeElapsed += timeDelta;
```

где timeDelta — это время, прошедшее между двумя кадрами.

После того, как пройдет одна секунда, мы вычисляем частоту кадров по следующей формуле:

```
FPS = (float)FrameCnt / TimeElapsed;
```

Затем мы обнуляем переменные FrameCnt и TimeElapsed и начинаем вычисление среднего значения частоты кадров для следующей секунды. Вот как выглядит весь код вместе:

```
void CalcFPS(float timeDelta)
{
    FrameCnt++;
    TimeElapsed += timeDelta;

    if(TimeElapsed >= 1.0f)
    {
        FPS = (float)FrameCnt / TimeElapsed;
        TimeElapsed = 0.0f;
        FrameCnt = 0;
    }
}
```

9.2. CD3DFont

DirectX SDK предоставляет полезный вспомогательный код, который находится а папке \Samples\C++\Common корневого каталога DXSDK. Среди этого кода есть класс CD3DFont, который отображает текст используя текстурированные треугольники и Direct3D. Поскольку CD3DFont использует для визуализации Direct3D, а не GDI, он работает гораздо быстрее, чем ID3DXFont. Однако, в отличие от ID3DXFont, CD3DFont не поддерживает сложные шрифты и форматирование. Если вам нужна скорость и достаточно простых шрифтов, класс CD3DFont — это ваш выбор.

Чтобы использовать класс CD3DFont необходимо добавить к приложению файлы d3dfont.h, d3dfont.cpp, d3dutil.h, d3dutil.cpp, dxutil.h и dxutil.cpp. Эти файлы находятся в папках Include и Src, которые расположены в ранее упоминавшейся папке Common.

9.2.1. **Создание экземпляра** CD3DFont

Экземпляр CD3DFont создается также как обычный объект C++ с помощью следующего конструктора:

```
CD3DFont(
    const TCHAR* strFontName,
    DWORD dwHeight,
    DWORD dwFlags=0L
);
```

- **§** strFontName — Завершающаяся нулем строка, задающая имя гарнитуры шрифта.
- **§** dwHeight — Высота шрифта.
- **§** dwFlags — Необязательные дополнительные флаги; параметру можно присвоить 0 или использовать произвольную комбинацию флагов D3DFONT_BOLD, D3DFONT_ITALIC, D3DFONT_ZENABLE.

После создания объекта CD3DFont для инициализации шрифта мы должны вызвать следующие методы (в указанном порядке):

```
Font = new CD3DFont("Times New Roman", 16, 0); // создание экземпляра
Font->InitDeviceObjects(Device);
Font->RestoreDeviceObjects();
```

9.2.2. **Рисование текста**

Теперь, когда мы создали и инициализировали объект CD3DFont, можно нарисовать какой-нибудь текст. Рисование текста выполняет следующий метод:

```
HRESULT CD3DFont::DrawText(
    FLOAT x,
    FLOAT y,
    DWORD dwColor,
    const TCHAR* strText,
    DWORD dwFlags=0L
);
```

- **§** x — координата x в экранном пространстве с которой начинается рисование текста.
- **§** y — координата y в экранном пространстве с которой начинается рисование текста.
- **§** dwColor — Цвет текста.
- **§** strText — Указатель на рисуемую строку.
- **§** dwFlags — Необязательные флаги визуализации; можете присвоить этому параметру 0 или указать произвольную комбинацию флагов D3DFONT_CENTERED, D3DFONT_TWOSIDED, D3DFONT_FILTERED.

Пример использования метода:

```
Font->DrawText(20, 20, 0xff000000, "Hello, World");
```

9.2.3. **Очистка**

Перед удалением объекта CD3DFont необходимо вызвать ряд процедур очистки, как показано в приведенном ниже фрагменте кода:

```
Font->InvalidateDeviceObjects();
Font->DeleteDeviceObjects();
delete Font;
```

9.3. D3DXCreateText

Функция D3DXCreateText создает трехмерную сетку, представляющую строку текста. На рис. 9.1 показана такая трехмерная сетка, отображаемая приложением FontMesh3D, которое находится в сопроводительных файлах к данной главе.

[ЗДЕСЬ ОБЪЕМНЫЙ ТЕКСТ]

***Рис**. 9.1. Трехмерный текст, созданный функцией D3DXCreateText*

Прототип функции выглядит следующим образом:

```
HRESULT D3DXCreateText(
    LPDIRECT3DDEVICE9 pDevice,
    HDC hDC,
    LPCTSTR pText,
    FLOAT Deviation,
    FLOAT Extrusion,
    LPD3DXMESH* ppMesh,
    LPD3DXBUFFER* ppAdjacency,
    LPGLYPHMETRICSFLOAT pGlyphMetrics
);
```

В случае успешного завершения функция возвращает D3D_OK.

- pDevice — Устройство, связанное с сеткой.
- hDC — Дескриптор контекста устройства, содержащего описание шрифта, которое будет использоваться для генерации сетки.
- pText — Указатель на завершающуюся нулем строку с текстом, для которого будет создаваться сетка.
- Deviation — Максимальное хордальное отклонение от контуров шрифта TrueType. Значение должно быть больше или равно нулю. Когда значение равно нулю, хордальное отклонение будет равно одной проектной единице оригинального шрифта.
- Extrusion — Глубина шрифта, измеряемая вдоль отрицательного направления оси Z.
- ppMesh — Возвращает созданную сетку.
- ppAdjacency — Возвращает информацию о смежности для созданной сетки. Если она вам не нужна, укажите в данном параметре null.
- pGlyphMetrics — Указатель на массив структур LPGLYPHMETRICSFLOAT, содержащий данные метрик глифов. Каждый элемент массива содержит информацию о местоположении и ориентации соответствующего глифа в строке. Количество элементов массива должно соответствовать количеству символов в строке. Если вы не хотите связываться с метриками глифов, просто укажите 0.

Следующий фрагмент кода показывает как создать изображающую текст трехмерную сетку с помощью рассматриваемой функции.

```
// Получение дескриптора контекста устройства
HDC hdc = CreateCompatibleDC(0);
// Заполнение структуры LOGFONT, описывающей свойства шрифта
LOGFONT lf;
ZeroMemory(&lf, sizeof(LOGFONT));
lf.lfHeight    = 25;    // в логических единицах
lf.lfWidth     = 12;    // в логических единицах
lf.lfWeight    = 500;   // насыщенность,
                        // диапазон 0(тонкий) - 1000(жирный)
lf.lfItalic    = false;
lf.lfUnderline = false;
lf.lfStrikeOut = false;
lf.lfCharSet   = DEFAULT_CHARSET;
strcpy(lf.lfFaceName, "Times New Roman"); // гарнитура шрифта
// Создаем шрифт и выбираем его в контексте устройства
HFONT hFont;
HFONT hFontOld;
hFont = CreateFontIndirect(&lf);
hFontOld = (HFONT)SelectObject(hdc, hFont);
// Создаем представляющую текст трехмерную сетку
ID3DXMesh* Text = 0;
D3DXCreateText(_device, hdc, "Direct3D", 0.001f, 0.4f, &Text, 0, 0);
// Восстанавливаем бывший до этого шрифт и освобождаем ресурсы
SelectObject(hdc, hFontOld);
DeleteObject(hFont);
DeleteDC(hdc);
```

Теперь вы можете визуализировать трехмерную сетку просто вызвав метод сетки DrawSubset:

```
Text->DrawSubset(0);
```

9.4. **Итоги**

- Если вам необходима поддержка сложных шрифтов и форматирования, используйте для визуализации текста интерфейс ID3DXFont. Он использует при визуализации текста GDI и поэтому работает медленно.
- Для быстрой визуализации простого текста используйте класс CD3DFont. Он использует для визуализации текста текстурированные треугольники и Direct3D и поэтому работает гораздо быстрее, чем ID3DXFont.
- Чтобы создать трехмерную сетку, изображающую строку текста, воспользуйтесь функцией D3DXCreateText.