ZFont 클래스 사용 가이드

개요

`ZFont`는 DirectX 11 기반의 텍스트 렌더링 클래스입니다. DirectXTK의 `SpriteFont`와 `SpriteBatch`를 사용하여 UTF-8 텍스트(한글 포함)를 화면에 렌더링합니다.

특징

✅ **DirectXTK 기반** - SpriteFont 및 SpriteBatch 사용
✅ **UTF-8 완벽 지원** - 한글, 중국어, 일본어 등 모든 유니코드 문자
✅ **동적 크기 조절** - 런타임에 폰트 스케일 자동 계산
✅ **외부 SpriteBatch 지원** - 다른 렌더링과 배칭 가능
✅ **색상 및 포맷 지원** - 정렬, 색상, 그림자 효과

클래스 구조

멤버 변수

```
private:
    DirectX::SpriteFont* m_pSpriteFont;    // DirectXTK SpriteFont
    DirectX::SpriteBatch* m_pSpriteBatch;  // DirectXTK SpriteBatch (내부용)
    long m_Size;                           // 요청된 폰트 크기

    std::string m_Name;                    // 폰트 이름 (.spritefont 파일명)
    int m_iSize;                           // 크기 (비교용)
    BOOL m_bBold;                          // 볼드 여부
    BOOL m_bItalic;                        // 이탤릭 여부
```

API 문서

생성자 및 소멸자

```
ZFont();
~ZFont();
```

사용 예:

```
ZFont* pFont = new ZFont();
// ... 사용 ...
delete pFont;
```

폰트 생성

`BOOL Create(ZGraphics& gfx, const char* SpriteFontFile, long Size, BOOL Bold, BOOL Italic)`

`.spritefont` 파일을 로드하여 폰트를 생성합니다.

매개변수:

- `gfx` : ZGraphics 참조 (DirectX 장치)
- `SpriteFontFile` : `.spritefont` 파일 경로
- `Size` : 원하는 폰트 크기 (픽셀)

- `Bold` : 볼드 여부 (현재는 메타데이터용)
- `Italic` : 이탤릭 여부 (현재는 메타데이터용)

반환값:

- `TRUE` : 성공
- `FALSE` : 실패 (파일 없음, 로드 오류)

사용 예:

```
ZFont font;

// 기본 폰트 생성
if (font.Create(gfx, "./Data/Fonts/Arial_16.spritefont", 16, FALSE, FALSE))
{
    std::cout << "폰트 로드 성공!" << std::endl;
}

// 큰 볼드 폰트
ZFont titleFont;
titleFont.Create(gfx, "./Data/Fonts/Impact_24.spritefont", 24, TRUE, FALSE);

// 한글 폰트
ZFont koreanFont;
koreanFont.Create(gfx, "./Data/Fonts/NanumGothic_12.spritefont", 12, FALSE, FALSE);
```

주의사항:

- `.spritefont` 파일은 DirectXTK의 `MakeSpriteFont.exe`로 생성해야 함
- `Size` 매개변수는 스케일 계산에 사용됨
- 실제 폰트 크기는 `.spritefont` 파일에 미리 정의되어 있음

폰트 해제

`BOOL Free()`

폰트 리소스를 해제합니다.

사용 예:

```
font.Free();
```

텍스트 렌더링

`BOOL FastPrint(long XPos, long YPos, const char* text, DirectX::SpriteBatch* externalBatch = nullptr)`

빠른 텍스트 출력 (흰색, 왼쪽 정렬)

매개변수:

- `XPos`, `YPos` : 화면 좌표 (픽셀)
- `text` : UTF-8 텍스트
- `externalBatch` : 외부 SpriteBatch (nullptr이면 내부 사용)

사용 예:

```
// 내부 SpriteBatch 사용
font.FastPrint(100, 100, "Hello World");
```

```
font.FastPrint(100, 120, "안녕하세요");

// 외부 SpriteBatch 사용 (ZGUI에서)
DirectX::SpriteBatch* pBatch = ...;
pBatch->Begin();
font.FastPrint(50, 50, "Text 1", pBatch);
font.FastPrint(50, 70, "Text 2", pBatch);
pBatch->End();
```

`BOOL PrintLine(long XPos, long YPos, long Width, DirectX::XMFLOAT4 Color, const char* text, DirectX::SpriteBatch* externalBatch = nullptr)`

색상 지정 텍스트 출력

매개변수:

- `XPos`, `YPos`: 화면 좌표
- `Width`: 폭 (현재 미사용, 향후 확장용)
- `Color`: RGBA 색상 (0.0 ~ 1.0)
- `text`: UTF-8 텍스트
- `externalBatch`: 외부 SpriteBatch

사용 예:

```
// 빨간색 텍스트
DirectX::XMFLOAT4 red(1.0f, 0.0f, 0.0f, 1.0f);
font.PrintLine(100, 100, 200, red, "Red Text");

// 반투명 파란색
DirectX::XMFLOAT4 blue(0.0f, 0.0f, 1.0f, 0.5f);  // 50% 투명도
font.PrintLine(100, 120, 200, blue, "Blue Semi-Transparent");

// 흰색 (기본)
DirectX::XMFLOAT4 white(1.0f, 1.0f, 1.0f, 1.0f);
font.PrintLine(100, 140, 200, white, "한글 텍스트");
```

`BOOL PrintEx(long XPos, long YPos, long Width, long Height, DirectX::XMFLOAT4 Color, DWORD Format, const char* text, DirectX::SpriteBatch* externalBatch = nullptr)`

고급 텍스트 출력 (색상, 정렬, 포맷 지원)

매개변수:

- `XPos`, `YPos`: 화면 좌표
- `Width`, `Height`: 텍스트 영역 크기
- `Color`: RGBA 색상
- `Format`: 정렬 플래그 (DT_CENTER, DT_RIGHT, DT_VCENTER 등)
- `text`: UTF-8 텍스트
- `externalBatch`: 외부 SpriteBatch

Format 플래그:

- `DT_LEFT (0x00000000)`: 왼쪽 정렬 (기본값)
- `DT_CENTER (0x00000001)`: 가로 중앙 정렬
- `DT_RIGHT (0x00000002)`: 오른쪽 정렬
- `DT_VCENTER (0x00000004)`: 세로 중앙 정렬
- `DT_BOTTOM (0x00000008)`: 하단 정렬

사용 예:

```
// 중앙 정렬 텍스트
DirectX::XMFLOAT4 white(1.0f, 1.0f, 1.0f, 1.0f);
font.PrintEx(0, 0, 800, 100, white, DT_CENTER | DT_VCENTER, "Centered Text");

// 오른쪽 정렬
font.PrintEx(0, 0, 800, 50, white, DT_RIGHT, "Right Aligned");

// 하단 중앙 정렬
font.PrintEx(0, 500, 800, 100, white, DT_CENTER | DT_BOTTOM, "Bottom Center");

// 한글 + 색상 + 정렬
DirectX::XMFLOAT4 gold(1.0f, 0.84f, 0.0f, 1.0f);
font.PrintEx(100, 200, 400, 50, gold, DT_CENTER, "게임 제목");
```

정보 조회

`const char* GetName() const`

폰트 이름 반환 (.spritefont 파일명)

```
std::cout << "폰트 이름: " << font.GetName() << std::endl;
```

`int GetSize() const`

폰트 크기 반환

```
int size = font.GetSize();
std::cout << "폰트 크기: " << size << "px" << std::endl;
```

`BOOL IsBold() const`

볼드 여부 반환

```
if (font.IsBold())
{
    std::cout << "볼드 폰트" << std::endl;
}
```

`BOOL IsItalic() const`

이탤릭 여부 반환

```
if (font.IsItalic())
{
    std::cout << "이탤릭 폰트" << std::endl;
}
```

DirectXTK 객체 접근

`DirectX::SpriteFont* GetSpriteFont()`

내부 SpriteFont 객체 반환

```
DirectX::SpriteFont* pSpriteFont = font.GetSpriteFont();
if (pSpriteFont)
{
    // DirectXTK 메서드 직접 사용
```

```
    DirectX::XMVECTOR size = pSpriteFont->MeasureString(L"Text");
}
```

`DirectX::SpriteBatch* GetSpriteBatch()`

내부 SpriteBatch 객체 반환

```
DirectX::SpriteBatch* pBatch = font.GetSpriteBatch();
```

완전한 사용 예제

예제 1: 기본 텍스트 렌더링

```
#include "ZFont.h"
#include "ZGraphics.h"

class Game
{
private:
    ZFont m_Font;

public:
    BOOL Init(ZGraphics& gfx)
    {
        // 폰트 로드
        if (!m_Font.Create(gfx, "./Data/Fonts/Arial_16.spritefont", 16))
        {
            std::cerr << "폰트 로드 실패!" << std::endl;
            return FALSE;
        }

        return TRUE;
    }

    void Render()
    {
        // 간단한 텍스트 출력
        m_Font.FastPrint(10, 10, "FPS: 60");
        m_Font.FastPrint(10, 30, "Score: 1000");
        m_Font.FastPrint(10, 50, "안녕하세요!");
    }
};
```

예제 2: 색상 및 정렬

```
void RenderUI(ZFont& font)
{
    DirectX::XMFLOAT4 white(1.0f, 1.0f, 1.0f, 1.0f);
    DirectX::XMFLOAT4 red(1.0f, 0.0f, 0.0f, 1.0f);
    DirectX::XMFLOAT4 green(0.0f, 1.0f, 0.0f, 1.0f);
    DirectX::XMFLOAT4 yellow(1.0f, 1.0f, 0.0f, 1.0f);

    // 제목 (중앙 정렬, 노란색)
    font.PrintEx(0, 50, 800, 100, yellow, DT_CENTER | DT_VCENTER,
                 "슈팅 게임");

    // 점수 (오른쪽 정렬, 흰색)
    font.PrintEx(600, 10, 200, 30, white, DT_RIGHT,
                 "Score: 1500");

    // HP (왼쪽 정렬, 녹색)
    font.PrintLine(10, 10, 100, green, "HP: 100/100");

    // 경고 메시지 (중앙 정렬, 빨간색)
```

```
    font.PrintEx(0, 400, 800, 50, red, DT_CENTER,
                 "경고: 적 접근 중!");
}
```

예제 3: ZGUI와 통합 (외부 SpriteBatch 사용)

```
// ZGUIDialog::DrawText 내부
BOOL ZGUIDialog::DrawText(std::string text, ZGUIElement* pElement,
                          RECT* prcDest, BOOL bShadow)
{
    RECT rcScreen = *prcDest;
    OffsetRect(&rcScreen, m_iX, m_iY);

    ZFont* pFont = GetFont(pElement->iFont);
    if (!pFont)
        return FALSE;

    // 그림자 (검은색, 1픽셀 오프셋)
    if (bShadow)
    {
        RECT rcShadow = rcScreen;
        OffsetRect(&rcShadow, 1, 1);

        DirectX::XMFLOAT4 shadowColor(0.0f, 0.0f, 0.0f,
                                      pElement->FontColor.Current.w);

        pFont->PrintEx(
            rcShadow.left, rcShadow.top,
            rcShadow.right - rcShadow.left,
            rcShadow.bottom - rcShadow.top,
            shadowColor, pElement->dwTextFormat, text.c_str(),
            _pSpriteBatch.get()  // ← 외부 SpriteBatch 전달
        );
    }

    // 본문 텍스트
    pFont->PrintEx(
        rcScreen.left, rcScreen.top,
        rcScreen.right - rcScreen.left,
        rcScreen.bottom - rcScreen.top,
        pElement->FontColor.Current, pElement->dwTextFormat, text.c_str(),
        _pSpriteBatch.get()  // ← 외부 SpriteBatch 전달
    );

    return TRUE;
}
```

예제 4: 여러 폰트 사용

```
class TextRenderer
{
private:
    ZFont m_TitleFont;    // 큰 폰트
    ZFont m_BodyFont;     // 일반 폰트
    ZFont m_SmallFont;    // 작은 폰트

public:
    BOOL Init(ZGraphics& gfx)
    {
        // 제목용 큰 폰트
        if (!m_TitleFont.Create(gfx, "./Data/Fonts/Impact_32.spritefont", 32))
            return FALSE;

        // 본문용 일반 폰트
        if (!m_BodyFont.Create(gfx, "./Data/Fonts/Arial_16.spritefont", 16))
            return FALSE;
```

```
    // 작은 폰트 (UI 등)
    if (!m_SmallFont.Create(gfx, "./Data/Fonts/Arial_12.spritefont", 12))
        return FALSE;

    return TRUE;
}

void RenderPage()
{
    DirectX::XMFLOAT4 white(1.0f, 1.0f, 1.0f, 1.0f);
    DirectX::XMFLOAT4 gray(0.7f, 0.7f, 0.7f, 1.0f);

    // 제목 (큰 폰트)
    m_TitleFont.PrintEx(0, 50, 800, 80, white, DT_CENTER | DT_VCENTER,
                        "게임 설명서");

    // 본문 (일반 폰트)
    m_BodyFont.FastPrint(100, 150, "1. 이동: WASD 키");
    m_BodyFont.FastPrint(100, 180, "2. 공격: 마우스 왼쪽 버튼");
    m_BodyFont.FastPrint(100, 210, "3. 점프: 스페이스바");

    // 하단 주석 (작은 폰트)
    m_SmallFont.PrintEx(0, 550, 800, 30, gray, DT_CENTER,
                        "© 2025 My Game Studio. All rights reserved.");
    }
};
```

UTF-8 처리

UTF-8 to UTF-16 변환

ZFont는 내부적으로 UTF-8 문자열을 UTF-16으로 변환합니다:

```
// ZFont.cpp 내부 (Utf8ToWstring 헬퍼 함수)
static std::wstring Utf8ToWstring(const char* utf8Str)
{
    if (utf8Str == nullptr || *utf8Str == '\0')
        return std::wstring();

    // Windows API 사용 (UTF-8 → UTF-16)
    int wideSize = MultiByteToWideChar(CP_UTF8, 0, utf8Str, -1, NULL, 0);
    if (wideSize <= 0)
        return std::wstring();

    std::wstring wideStr(wideSize - 1, 0);
    MultiByteToWideChar(CP_UTF8, 0, utf8Str, -1, &wideStr[0], wideSize);

    return wideStr;
}

// 사용 (PrintEx 내부)
std::wstring wText = Utf8ToWstring(text);
m_pSpriteFont->DrawString(batch, wText.c_str(), pos, colorVec, ...);
```

한글 사용 예제

```
// UTF-8 인코딩된 소스 파일
font.FastPrint(100, 100, "안녕하세요");
font.FastPrint(100, 120, "게임에 오신 것을 환영합니다!");
font.FastPrint(100, 140, "시작하려면 스페이스바를 누르세요");

// 변수에서 읽기
std::string playerName = "플레이어1";
std::string message = "안녕하세요, " + playerName + "님!";
font.FastPrint(100, 200, message.c_str());

// INI 파일에서 읽기 (ZInitFile)
```

```
ZInitFile iniFile;
iniFile.LoadIFT("./Data/Texts.ift");
std::string welcomeText = iniFile.GetValue("Korean", "WelcomeMessage");
font.FastPrint(100, 250, welcomeText.c_str());
```

폰트 크기 조절

자동 스케일 계산

ZFont는 요청된 크기와 `.spritefont` 파일의 기본 크기를 비교하여 자동으로 스케일을 계산합니다:

```
// ZFont::PrintEx 내부
float defaultSize = m_pSpriteFont->GetLineSpacing();  // .spritefont의 기본 크기
float scale = (m_Size > 0 && defaultSize > 0) ? (float)m_Size / defaultSize : 1.0f;

m_pSpriteFont->DrawString(batch, wText.c_str(), pos, colorVec, 0.0f, origin, scale);
```

사용 예제

```
// Arial_16.spritefont (16픽셀로 생성된 파일)을 24픽셀로 표시
ZFont font;
font.Create(gfx, "./Data/Fonts/Arial_16.spritefont", 24);  // scale = 24/16 = 1.5
font.FastPrint(100, 100, "This is 24px text");

// 같은 파일을 12픽셀로 표시
ZFont smallFont;
smallFont.Create(gfx, "./Data/Fonts/Arial_16.spritefont", 12);  // scale = 12/16 = 0.75
smallFont.FastPrint(100, 130, "This is 12px text");
```

주의사항:

- 너무 큰 스케일 (예: 10배)은 폰트가 뭉개질 수 있음
- 최적 품질을 위해서는 원하는 크기에 가까운 `.spritefont` 파일 생성 권장

.spritefont 파일 생성

DirectXTK MakeSpriteFont 사용

1. MakeSpriteFont.exe 다운로드

- DirectXTK 릴리스 페이지에서 다운로드
- 또는 소스에서 빌드

2. TrueType 폰트에서 생성

```
MakeSpriteFont.exe "Arial.ttf" Arial_16.spritefont /FontSize:16 /CharacterRegion:0x0020-0x007E
```

매개변수:

- `/FontSize:N`: 폰트 크기 (픽셀)
- `/CharacterRegion:X-Y`: 포함할 문자 범위 (유니코드)
- `/DefaultCharacter:X`: 기본 문자

3. 한글 포함

```
MakeSpriteFont.exe "NanumGothic.ttf" NanumGothic_16.spritefont /FontSize:16 ^
  /CharacterRegion:0x0020-0x007E ^
```

```
  /CharacterRegion:0xAC00-0xD7A3
```

한글 범위:

- `0xAC00-0xD7A3`: 한글 완성형 11,172자

4. 여러 언어 포함

```
MakeSpriteFont.exe "Arial.ttf" Arial_Multi_16.spritefont /FontSize:16 ^
  /CharacterRegion:0x0020-0x007E ^
  /CharacterRegion:0xAC00-0xD7A3 ^
  /CharacterRegion:0x4E00-0x9FFF
```

문자 범위:

- `0x0020-0x007E`: ASCII (영문, 숫자, 기호)
- `0xAC00-0xD7A3`: 한글
- `0x4E00-0x9FFF`: 중국어 (CJK 통합 한자)
- `0x3040-0x309F`: 일본어 히라가나
- `0x30A0-0x30FF`: 일본어 카타카나

성능 최적화

1. 외부 SpriteBatch 사용

```cpp
// ❌ 나쁜 방법 (매번 Begin/End)
for (int i = 0; i < 100; i++)
{
    font.FastPrint(10, 10 + i * 20, "Text");  // 내부 SpriteBatch 사용
    // → Begin/End가 100번 호출됨!
}

// ✅ 좋은 방법 (한 번만 Begin/End)
DirectX::SpriteBatch* pBatch = font.GetSpriteBatch();
pBatch->Begin();
for (int i = 0; i < 100; i++)
{
    font.FastPrint(10, 10 + i * 20, "Text", pBatch);
}
pBatch->End();
```

2. 문자열 캐싱

```cpp
// ❌ 나쁜 방법 (매 프레임 문자열 생성)
void Render()
{
    std::string text = "FPS: " + std::to_string(fps);
    font.FastPrint(10, 10, text.c_str());
}

// ✅ 좋은 방법 (변경 시에만 업데이트)
std::string m_FPSText;
int m_LastFPS = -1;

void Render()
{
    if (fps != m_LastFPS)
    {
        m_FPSText = "FPS: " + std::to_string(fps);
        m_LastFPS = fps;
    }
```

```
    font.FastPrint(10, 10, m_FPSText.c_str());
}
```

3. 폰트 재사용

```
// ✅ 폰트를 한 번만 로드하고 재사용
class FontManager
{
private:
    std::map<std::string, ZFont*> m_Fonts;

public:
    ZFont* GetFont(const std::string& name)
    {
        auto it = m_Fonts.find(name);
        if (it != m_Fonts.end())
            return it->second;

        // 폰트 로드 및 캐싱
        ZFont* pFont = new ZFont();
        if (pFont->Create(gfx, name.c_str(), 16))
        {
            m_Fonts[name] = pFont;
            return pFont;
        }

        delete pFont;
        return nullptr;
    }
};
```

문제 해결

문제: **텍스트가 표시되지 않음**

원인 1: SpriteBatch가 Begin()되지 않음
해결: 외부 SpriteBatch 사용 시 Begin()/End() 호출 확인

원인 2: 색상 알파값이 0
해결: `DirectX::XMFLOAT4(r, g, b, 1.0f)` (알파 = 1.0)

원인 3: 화면 밖에 렌더링
해결: 좌표 확인

문제: **한글이 깨짐**

원인: `.spritefont` 파일에 한글이 포함되지 않음
해결: MakeSpriteFont로 한글 범위 포함하여 재생성

```
MakeSpriteFont.exe "NanumGothic.ttf" Korean.spritefont /FontSize:16 ^
  /CharacterRegion:0xAC00-0xD7A3
```

문제: **폰트가 흐리거나 뭉개짐**

원인: 스케일이 너무 크거나 작음
해결: 원하는 크기에 가까운 `.spritefont` 파일 생성

```
// 16px 파일을 64px로 확대 (나쁨)
font.Create(gfx, "Arial_16.spritefont", 64);  // scale = 4.0 (흐림)

// 32px 파일을 64px로 확대 (좋음)
font.Create(gfx, "Arial_32.spritefont", 64);  // scale = 2.0 (괜찮음)
```

ZGUI와의 통합

ZFont는 ZGUI 시스템에서 ZGUIResource에 의해 관리됩니다:

```
// ZGUIResource.cpp
int ZGUIResource::AddFont(std::string faceName, int iSize, BOOL bBold, BOOL bItalic)
{
    ZFont* pFont = new ZFont();

    if (pFont->Create(*m_pGraphicsRef, faceName.c_str(), iSize, bBold, bItalic))
    {
        m_FontList.push_back(pFont);
        return (int)m_FontList.size() - 1;  // 인덱스 반환
    }

    delete pFont;
    return -1;
}

// ZGUIDialog.cpp
ZFont* ZGUIDialog::GetFont(int iResourceFontIndex)
{
    return m_pResourceRef->GetFont(iResourceFontIndex);
}
```

관련 클래스

- **ZGUIResource**: ZFont 인스턴스 관리
- **ZGUIDialog**: ZFont를 사용하여 텍스트 렌더링
- **ZGUILabel/Button/Image**: 텍스트를 포함하는 컨트롤

작성일: 2025-11-01
버전: 1.0
작성자: Cascade AI