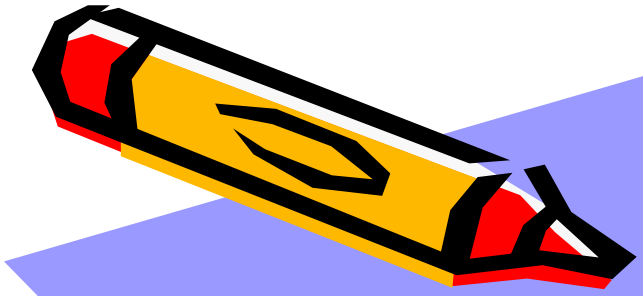


2006년 2학기 윈도우 게임 프로그래밍

제7강 트루타입폰트의 사용

이대현

한국산업기술대학교

오늘의 학습 내용

- 트루타입(True type) 폰트를 이용한 문자 출력
- 영문 폰트를 이용한 출력 방법
- 한글 폰트를 이용한 출력 방법

컴퓨터 폰트(Computer Font)

- 컴퓨터 폰트란?
  - 문자,기호 또는 각종 심볼들의 모양들로 구성된 파일. 아스키코드와 같은 코드를 통해서 필요한 모양들을 액세스할 수 있도록 구성됨.

- 폰트의 종류
  - 비트맵 폰트(Bitmap Font) – 문자별, 크기별로 폰트의 모양을 모두 일일이 비트맵 형태로 보관. 폰트 파일의 크기가 큼.
  - 윤곽선 폰트(Outline Fonts) – 폰트의 모양(윤곽선)을 벡터 방식으로 저장. 폰트 파일의 크기가 상대적으로 작음. Type1, Type2 및 Truetype 폰트 등이 있슴. 윈도우는 TrueType 폰트를 사용함.(C:₩WINDOWS₩Fonts 에 위치)
  - 스트로크 폰트(Stroke-based Fonts) – 문자의 모양을 여러 개의 선들의 연결로 구성하여 저장함.

SDL의 폰트를 이용한 문자 출력 과정

Step0: SDL_ttf 라이브러리를 포함.(SDL_ttf.h, SDL_ttf.lib, SDL_ttf.dll 사용)

↓

Step1: SDL_ttf 라이브러리 초기화(TTF_Init() 호출)

↓

Step2: 폰트의 로딩(TTF_OpenFont() 호출)

↓

Step3: 문자열을 렌더링하여 surface를 생성(TTF_RenderText_Blended() 등을 호출)

↓

Step4: 생성된 surface를 화면 surface로 블리팅

↓

Step5: 폰트의 해제(TTF_CloseFont() 호출)

↓

Step6: 라이브러리 종료(TTF_Quit() 호출)

Lecture07-01: 영문 폰트를 이용한 출력

Lecture07-01 프로젝트의 구성

- C++ 소스 파일들
  - gameengine.cpp – 실습 시간에 작성.
  - main.cpp
  - introstate.cpp
  - playstate.cpp – 실습 시간에 작성.
  - sprite.cpp
- C++ 헤더 파일들
  - gameengine.h
  - gamestate.h
  - introstate.h
  - playstate.h
  - sprite.h
- 이미지 파일들
  - background.bmp – 배경.
  - spritesheet.bmp – 스프라이트 이미지 모음.
  - HYNAMB.ttf – 폰트 파일. 한양나무 bold 체.

gameengine.cpp

```
void CGameEngine::Init(const char* title, int width, int height,
                                          int bpp, bool fullscreen)
{
    int flags = 0;
    SDL_Init(SDL_INIT_EVERYTHING);
    if ( fullscreen ) {
         flags = SDL_FULLSCREEN;
    }
    screen = SDL_SetVideoMode(width, height, bpp, flags);
    m_fullscreen = fullscreen;
    m_running = true;
    TTF_Init();
}

void CGameEngine::Cleanup()
{
    while ( !states.empty() ) {
         states.back()->Cleanup();
         states.pop_back();
    }
    if ( m_fullscreen ) {
         screen = SDL_SetVideoMode(640, 480, 0, 0);
    }
    TTF_Quit();
    SDL_Quit();
}
```


playstate.cpp (1)

```
void CPlayState::Init()
{
    player = new Sprite;

    SDL_Surface* temp = SDL_LoadBMP("background.bmp");
    bg = SDL_DisplayFormat(temp);
    SDL_FreeSurface(temp);

    font = TTF_OpenFont("HYNAMB.ttf", 30);
    SDL_Color color={0,0,0};
    msg1 = TTF_RenderText_Blended(font,"English Font Output Blended",color);
    msg2 = TTF_RenderText_Solid(font,"English Font Output Solid",color);

    SDL_ShowCursor(false);
    SDL_EnableKeyRepeat(10, SDL_DEFAULT_REPEAT_INTERVAL);
}

void CPlayState::Cleanup()
{
    delete player;
    TTF_CloseFont(font);
    SDL_FreeSurface(bg);
    SDL_FreeSurface(msg1);
    SDL_FreeSurface(msg2);
    SDL_ShowCursor(false);
}
```

playstate.cpp (2)

```
void CPlayState::Draw(CGameEngine* game)
{
  SDL_BlitSurface(bg, NULL, game->screen, NULL);
  player->draw();

  SDL_Rect rect1 = {40, 40, 0, 0};
  SDL_Rect rect2 = {40, 80, 0, 0};
  SDL_BlitSurface(msg1, NULL, game->screen, &rect1);
  SDL_BlitSurface(msg2, NULL, game->screen, &rect2);

  SDL_Flip(game->screen);
}
```


실행 화면

폰트 라이브러리의 초기화 및 종료

```
int TTF_Init(void);
int TTF_Quit(void);
```

```
void CGameEngine::Init(const char* title, int width, int height,
                                                    int bpp, bool fullscreen)
{
    int flags = 0;
    SDL_Init(SDL_INIT_EVERYTHING);
    if ( fullscreen ) {
          flags = SDL_FULLSCREEN;
    }
    screen = SDL_SetVideoMode(width, height, bpp, flags);
    m_fullscreen = fullscreen;
    m_running = true;
    TTF_Init();
}
```

- SDL_ttf 라이브러리의 모든 함수를 사용하기 위해서, 반드시 호출해야 함.
- 프로그램 종료시, TTF_Quit() 함수를 호출하여, 라이브러리 사용 종료를 알림. SDL_Quit() 이전에 사용

폰트의 로딩 및 해제

```
TTF_Font *TTF_OpenFont(const char *file,
                                    int ptsize);
void TTF_CloseFont(TTF_Font *font);
```

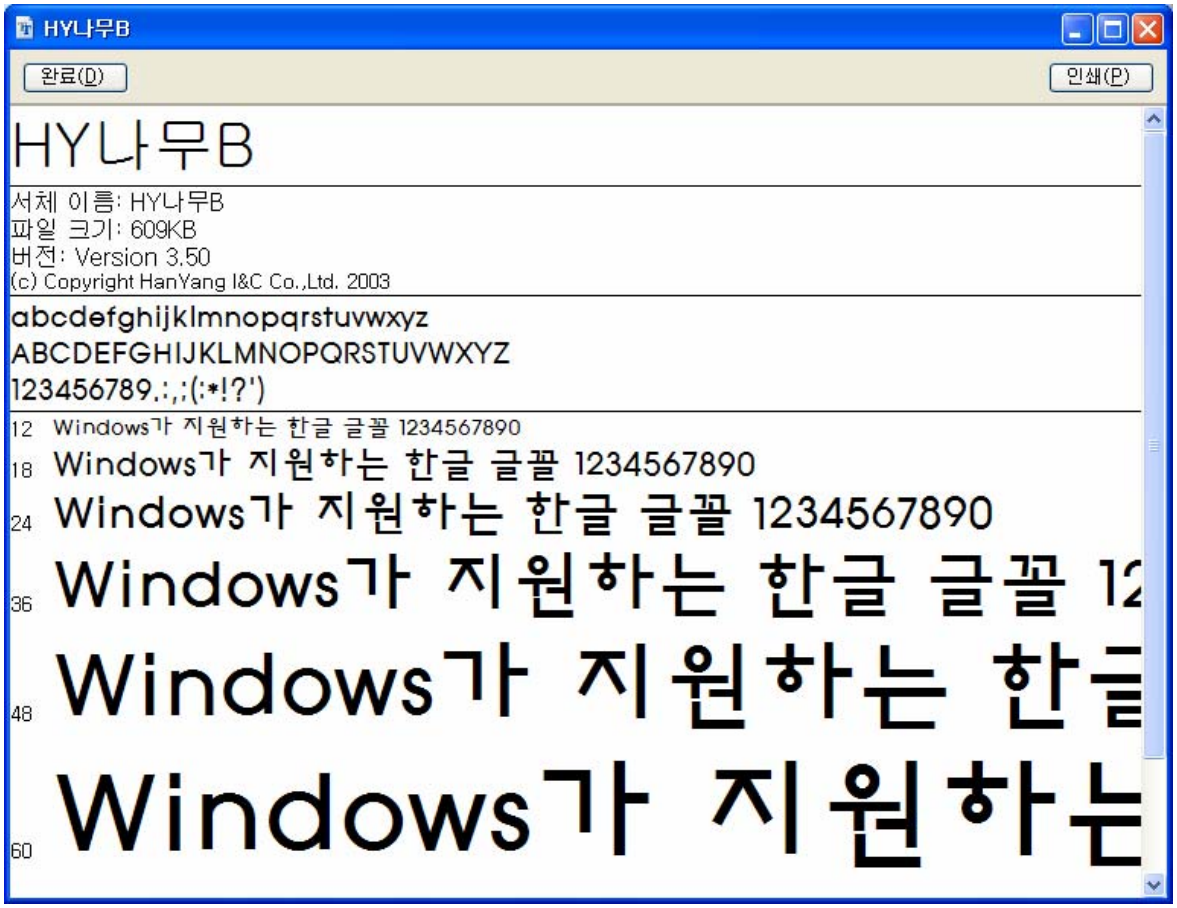
```
void CPlayState::Init()
{
    player = new Sprite;

    SDL_Surface* temp = SDL_LoadBMP("background.bmp");
    bg = SDL_DisplayFormat(temp);
    SDL_FreeSurface(temp);

    font = TTF_OpenFont("HYNAMB.ttf", 30);
```

- "HYNAMB.ttf" 트루타입폰트를 30 point의 크기로 로딩한다.

HYNAMB.TTF



문자열의 출력

```
SDL_Surface
*TTF_RenderText_Solid(TTF_Font *font,
const char *text, SDL_Color fg);

SDL_Surface
*TTF_RenderText_Blended(TTF_Font *font,
const char *text, SDL_Color fg)
```

• 폰트의 색상을 검정색으로 설정.

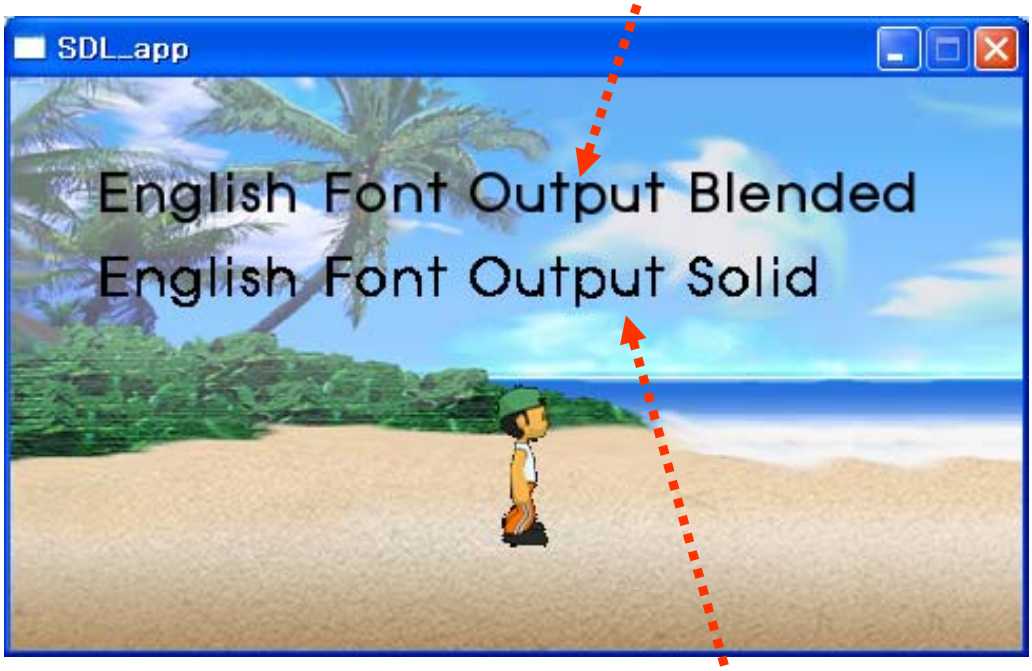
```
SDL_Color color={0,0,0};

msg1 = TTF_RenderText_Blended(font,"English Font Output Blended",color);

msg2 = TTF_RenderText_Solid(font,"English Font Output Solid",color);
```

Blended VS. Solid

• Blended 모드:앤티알리아스된 부드러운 윤곽선의 표현.



• Solid 모드: 딱딱한 외곽선의 표현. 빠른 처리 속도.

실습

Lecture07-02:
한글 폰트를 이용한 출력

Lecture07-02 프로젝트의 구성

- C++ 소스 파일들
  - gameengine.cpp
  - main.cpp
  - introstate.cpp
  - playstate.cpp – 실습 시간에 작성.
  - sprite.cpp
  - han2unicode.cpp – 완성형 한글 코드를 유니코드로 변환.

- C++ 헤더 파일들
  - gameengine.h
  - gamestate.h
  - introstate.h
  - playstate.h
  - sprite.h

- 이미지 파일들
  - background.bmp – 배경.
  - spritesheet.bmp – 스프라이트 이미지 모음.
  - HYNAMB.ttf – 폰트 파일. 한양나무 bold 체.

playstate.cpp (1)

```
void CPlayState::Init()
{
  player = new Sprite;

  SDL_Surface* temp = SDL_LoadBMP("background.bmp");
  bg = SDL_DisplayFormat(temp);
  SDL_FreeSurface(temp);

  font = TTF_OpenFont("HYNAMB.ttf", 34);

  char *src1 = "한글 폰트 출력 Blended";
  char *src2 = "한글 폰트 출력 Solid";

  Uint16 dest1[50], dest2[50];

  han2unicode(src1, dest1);
  han2unicode(src2, dest2);

  SDL_Color color={0,0,0};
  msg1 = TTF_RenderUNICODE_Blended(font, dest1, color);
  msg2 = TTF_RenderUNICODE_Solid(font, dest2, color);

  SDL_ShowCursor(false);
  SDL_EnableKeyRepeat(10, SDL_DEFAULT_REPEAT_INTERVAL);
}
```

실행 결과

한글 폰트를 이용한 출력 방법

```
char *src1 = "한글 폰트 출력 Blended";
char *src2 = "한글 폰트 출력 Solid";
```

- 출력할 한글 문자열을 설정.
- 완성형코드로 저장됨.

```
Uint16 dest1[50], dest2[50];
```

- 유니코드로 변환된 문자열을 저장할 공간의 확보.
- 문자열의 길이만큼을 제대로 확보해야 함.

```
han2unicode(src1, dest1);
han2unicode(src2, dest2);
```

- 완성형문자열 src를, 유니코드문자열 dest로 변환함.

```
SDL_Color color={0,0,0};
msg1 = TTF_RenderUNICODE_Blended(font, dest1, color);
msg2 = TTF_RenderUNICODE_Solid(font, dest2, color);
```

- 유니코드 문자열을 렌더링함.

실습 과제 #6

- 한명의 캐릭터 애니메이션
  - 한명의 캐릭터가 화면에 출력됨.
  - 캐릭터는 방향키를 이용하여 상하좌우이동이 됨.
  - 화면의 상단에는 “SDL Game Programming” 이 항상 표시.
  - 캐릭터의 머리 위에는 캐릭터의 이름과 x,y 좌표가 항상 표시.