

C++ 프로그래밍 및 실습

리소스 관리 모듈

진척 보고서 #1

제출일자: 2023.11.25

제출자명: 김기태

제출자학번: 185482

1. 프로젝트 목표

1) 배경 및 필요성

개인적으로 렌파이라는 툴을 이용한 비주얼 노벨 개발 작업을 진행 중에 있다. 작업을 진행할 때 관리해야 할 리소스가 너무나도 많아 불편한 점이 많이 있었기 때문에 이를 보완하기 위한 **실사용 목적으로 개발**한다. 실제로 현재 개발중인 버전 기준으로 리소스 정의 코드 + 시나리오 스크립트가 함께 들어 있어서 만 줄 가량을 뛰어넘는 스크립트 문서의 분량 때문에 그때그때 필요한 부분을 제때 찾기에 불편하고 시간이 소요되었기 때문에 스크립트 문서를 분할하여 각 리소스 관리 부분을 담당하는 텍스트 파일을 따로 만들 필요성이 생겼기 때문이다. 또한 개발 중인 리소스의 용량이 4GB를 넘는 문제 때문에 완성 후 배포 시에 필요 없는 더미 데이터들을 식별하여 삭제하여서 용량을 최소화시켜야만 할 필요가 생겼기 때문이기도 하다.

2) 프로젝트 목표

첫째로는 외부 파일에 리소스 정의에 사용된 코드들을 자동적으로 저장해내는 것이다. 렌파이의 스크립트 기반인 'rpy' 확장자의 텍스트 파일로 저장하도록 한다. 따라서 보유 리소스의 내용이 변동되어도 단 한 번의 툴 사용만으로 모든 리소스의 정의 내역이 현재 파일과 정확히 연동되도록 할 것이다.

둘째로는 외부 텍스트 파일을 읽어 들여서 해당 텍스트 파일에서 사용된 파일과 사용되지 않은 파일을 찾아내서 명단화하는 것이다. 여기서 읽기에 사용할 파일은 script.rpy로 이 파일의 텍스트를 읽어 들여서 만일 사용한 적이 있는 리소스라면 사용된 리소스의 목록을 담은 텍스트 파일에 분류하고, 사용한 적이 없는 리소스면 사용되지 않은 리소스 목록을 담은 텍스트 파일로 분류한다.

3) 차별점

기존에 리소스 정의 및 정리 용도로 사용하던 파이썬 기반 코드들은 사용자가 직접 스크립트 파일에 복사 붙여넣기를 해 주어야만 한다는 단점이 있었다. 따라서 기존 코드와는 달리 **C++로 재설계**함과 동시에 외부 파일에 결과값을 저장할 수 있도록 하며, 어떤 환경에서도 실행되도록 절대 경로를 상대경로로 변경하고, 기존 코드들이 배경음악, 사운드 이펙트, 스탠딩 그래픽, 배경 그래픽 등으로 나뉘어져 있었던 것을 한 파일 안에 통합환경으로 구현할 계획이다.

2. 기능 계획

1) 리소스 정의 및 외부 파일에 자동 저장

- 각자 서식에 맞는 리소스 정의를 개별 파일에 저장하고, 리소스 정의 파일을 각자 파일에 맞는 audio.rpy, scg.rpy, bg.rpy등의 파일에 저장한다. 종류별로 따로 따로 저장함으로써 스크립트의 가독성을 높이고 관리를 더 쉽게 할 수 있다.

2) 사용되지 않은 리소스 식별 기능

사용되지 않은 리소스를 스크립트 파일에서 동일 문자열의 존재 여부를 확인하여 단 하나라도 존재하면 존재하는 리소스 목록에 등재하고, 존재하지 않으면 존재하지 않는 리소스 목록에 등재하여 txt파일로 내보내는 것이다. 이는 테스트 버전 내보내기 및 개발 완료 단계에서 더미 데이터의 최소화를 통한 저장공간 절약에 도움이 될 것이다.

3. 진척사항

1) 기능 구현

(1) 리소스 자동 정의(4개 : BGM, SE, SCG, BG)

- 입출력

특정 경로에 위치하고 있는 파일들이 입력값이며, 출력값으로 bgm.rpy, se.rpy, scg.rpy,

bg.rpy로 4개의 파일이 만들어진다.

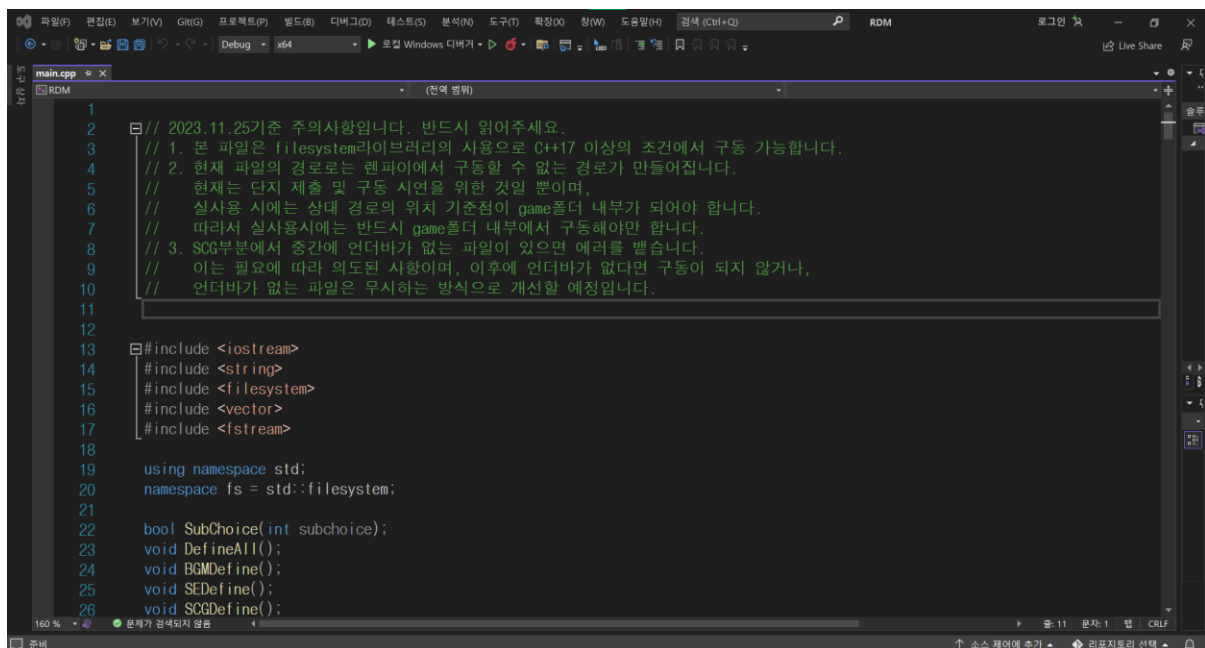
- 설명

Filesystem 라이브러리를 이용하여 파일의 경로와 제목을 끊어 오고, 그 파일의 경로를 벡터 배열을 이용하여 for문을 통해 저장한다. 경로는 그대로 저장하지만, 제목의 경우에는 렌파이가 인식할 수 없는 공백과 하이픈을 언더바로 대체하는 가공 작업을 한 차례 거친다. 그 뒤에 오디오, SCG, BG규격에 맞추어 각기 다른 유형으로 파일에 쓴다.

- 적용된 배운 내용 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)

조건문, 반복문, 함수, 벡터 배열 등이 사용된다.

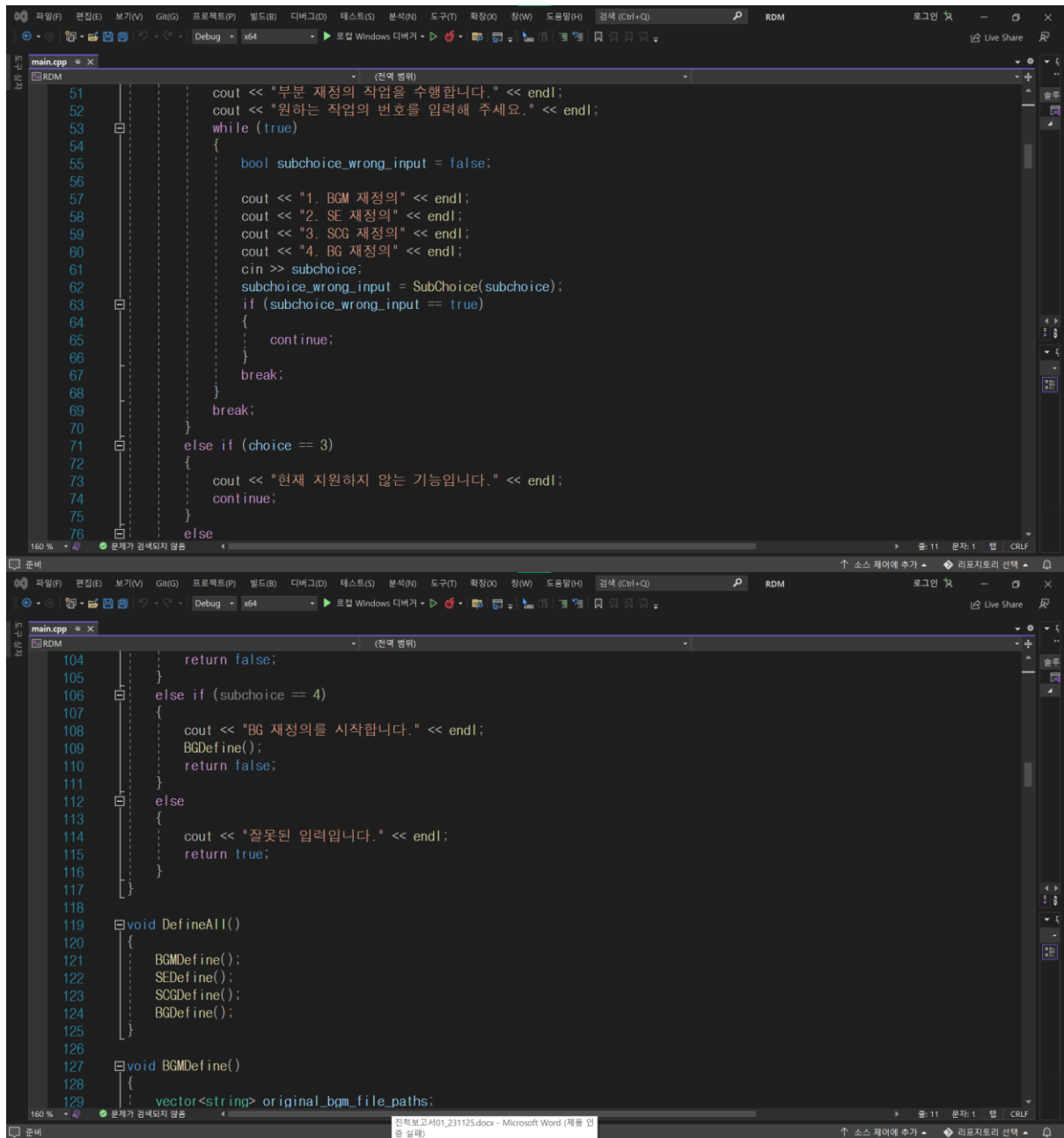
- 코드 스크린샷



```
1 // 2023.11.25기준 주의사항입니다. 반드시 읽어주세요.
2 // 1. 본 파일은 filesystem라이브러리의 사용으로 C++17 이상의 조건에서 구동 가능합니다.
3 // 2. 현재 파일의 경로로는 렌파이에서 구동할 수 없는 경로가 만들어집니다.
4 // 현재는 단지 제출 및 구동 시연을 위한 것일 뿐이며,
5 // 실사용 시에는 상대 경로의 위치 기준점이 game폴더 내부가 되어야 합니다.
6 // 따라서 실사용시에는 반드시 game폴더 내부에서 구동해야만 합니다.
7 // 3. SCG부분에서 중간에 언더바가 없는 파일이 있으면 에러를 뱉습니다.
8 // 이는 필요에 따라 의도된 사항이며, 이후에 언더바가 없다면 구동이 되지 않거나,
9 // 언더바가 없는 파일은 무시하는 방식으로 개선할 예정입니다.
10
11
12
13 #include <iostream>
14 #include <string>
15 #include <filesystem>
16 #include <vector>
17 #include <fstream>
18
19 using namespace std;
20 namespace fs = std::filesystem;
21
22 bool SubChoice(int subchoice);
23 void DefineAll();
24 void BGMDefine();
25 void SEDefine();
26 void SCGDefine();
```

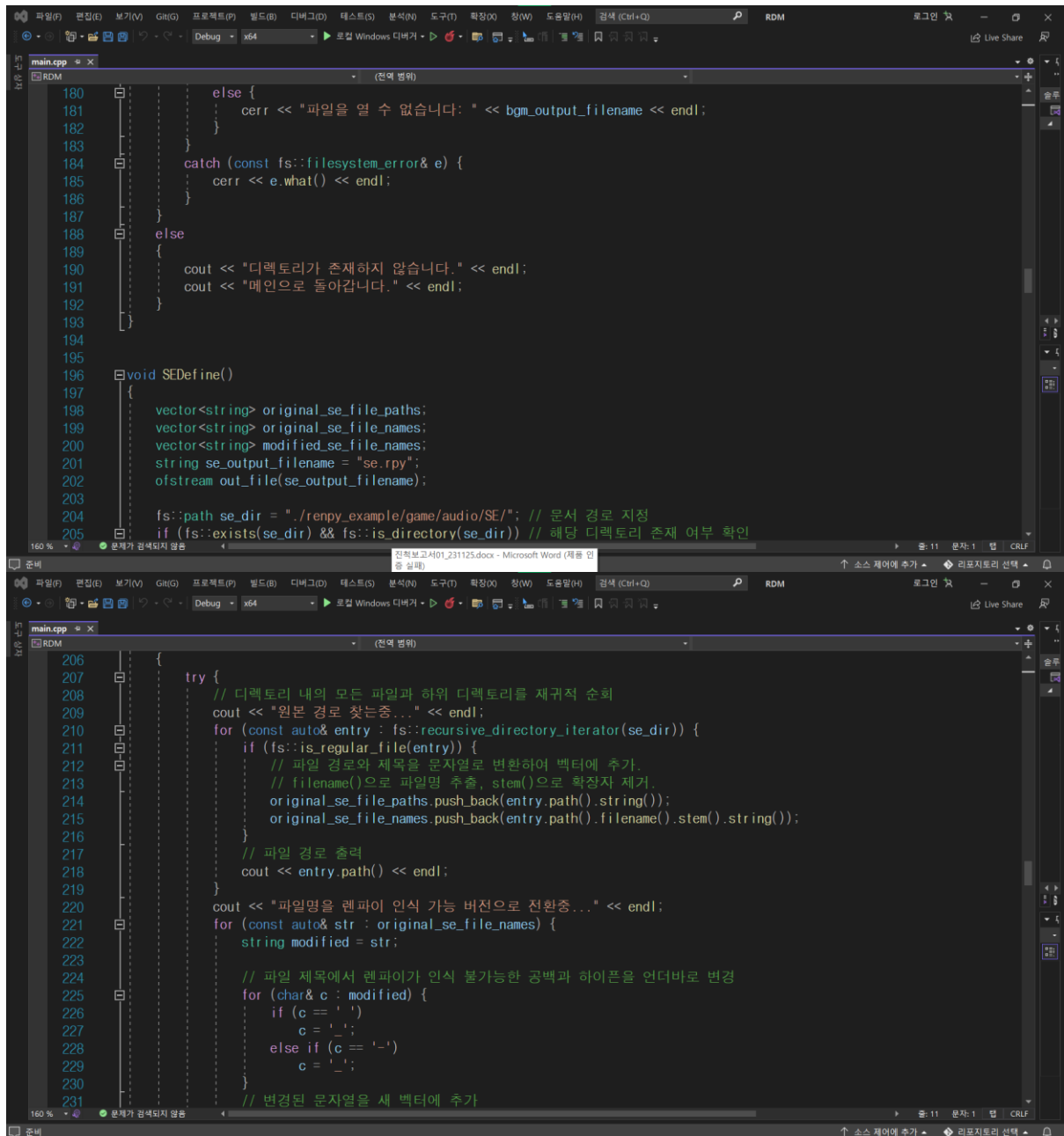
```
main.cpp x
RDM
26 void SCGDefine();
27 void BGMDefine();
28
29 int main()
30 {
31     int choice = 0;
32     int subchoice = 0;
33     cout << "렌파이 전용 리소스 관리 모듈입니다." << endl;
34     cout << "해야 할 작업의 번호를 입력해 주세요." << endl;
35
36     while (true)
37     {
38         cout << "1. 전체 재정의" << endl;
39         cout << "2. 부분 재정의" << endl;
40         cout << "3. 사용된 리소스 검출" << endl;
41         cin >> choice;
42
43         if (choice == 1)
44         {
45             cout << "전체 재정의 작업을 수행합니다." << endl;
46             DefineAll();
47             break;
48         }
49         else if (choice == 2)
50         {
51             cout << "부분 재정의 작업을 수행합니다." << endl;
52         }
53     }
54 }
```

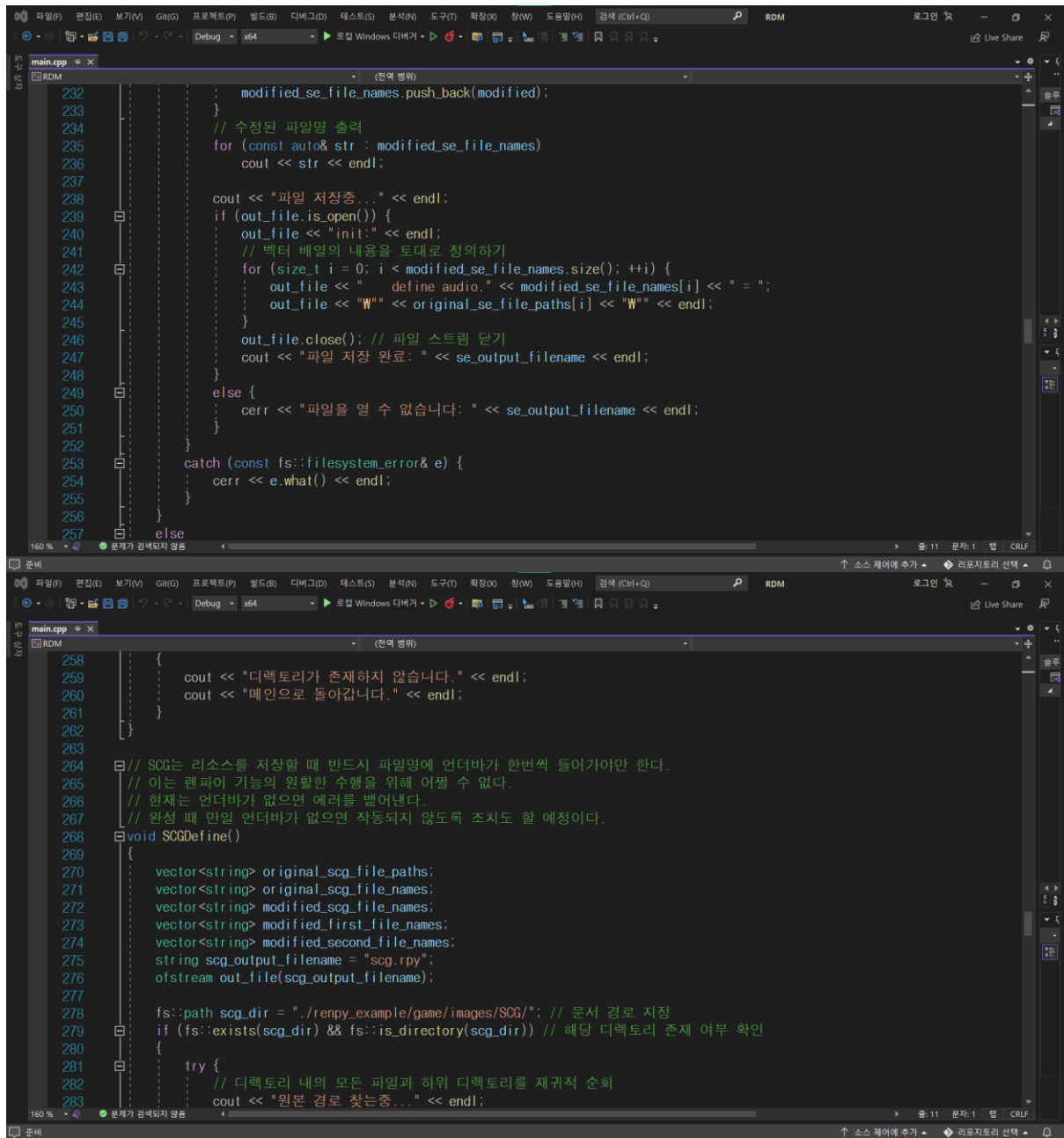
```
main.cpp x
RDM
78         cout << "잘못된 입력입니다. 다시 입력해 주세요." << endl;
79         continue;
80     }
81 }
82 return 0;
83
84
85
86 bool SubChoice(int subchoice)
87 {
88     if (subchoice == 1)
89     {
90         cout << "BGM 재정의를 시작합니다." << endl;
91         BGMDefine();
92         return false;
93     }
94     else if (subchoice == 2)
95     {
96         cout << "SE 재정의를 시작합니다." << endl;
97         SEDefine();
98         return false;
99     }
100     else if (subchoice == 3)
101     {
102         cout << "SCG 재정의를 시작합니다." << endl;
103         SCGDefine();
104     }
105 }
```



```
127 void BGMDefine()
128 {
129     vector<string> original_bgm_file_paths;
130     vector<string> original_bgm_file_names;
131     vector<string> modified_bgm_file_names;
132     string bgm_output_filename = "bgm.rpy";
133     ofstream out_file(bgm_output_filename);
134
135     fs::path bgm_dir = "../renpy_example/game/audio/BGM/"; // 문서 경로 지정
136     if (fs::exists(bgm_dir) && fs::is_directory(bgm_dir)) // 해당 디렉토리 존재 여부 확인
137     {
138         try {
139             // 디렉토리 내의 모든 파일과 하위 디렉토리를 재귀적 순회
140             cout << "원본 경로 찾는데..." << endl;
141             for (const auto& entry : fs::recursive_directory_iterator(bgm_dir)) {
142                 if (fs::is_regular_file(entry)) {
143                     // 파일 경로와 제목을 문자열로 변환하여 벡터에 추가.
144                     // filename()으로 파일명 추출, stem()으로 확장자 제거.
145                     original_bgm_file_paths.push_back(entry.path().string());
146                     original_bgm_file_names.push_back(entry.path().filename().stem().string());
147                 }
148                 // 파일 경로 출력
149                 cout << entry.path() << endl;
150             }
151             cout << "파일명을 렌파이 인식 가능 버전으로 전환중..." << endl;
152             for (const auto& str : original_bgm_file_names) {
153                 string modified = str;
154
155                 // 파일 제목에서 렌파이가 인식 불가능한 공백과 하이픈을 언더바로 변경
156                 for (char& c : modified) {
157                     if (c == ' ')
158                         c = '_';
159                     else if (c == '-')
160                         c = '_';
161                 }
162                 // 변경된 문자열을 새 벡터에 추가
163                 modified_bgm_file_names.push_back(modified);
164             }
165             // 수정된 파일명 출력
166             for (const auto& str : modified_bgm_file_names)
167                 cout << str << endl;
168
169             cout << "파일 저장중..." << endl;
170             if (out_file.is_open()) {
171                 out_file << "init:" << endl;
172                 // 벡터 배열의 내용을 토대로 정의하기
173                 for (size_t i = 0; i < modified_bgm_file_names.size(); ++i) {
174                     out_file << "    define audio." << modified_bgm_file_names[i] << " = ";
175                     out_file << "W" << original_bgm_file_paths[i] << "W" << endl;
176                 }
177                 out_file.close(); // 파일 스트림 닫기
178                 cout << "파일 저장 완료: " << bgm_output_filename << endl;

```





The image displays two screenshots of a C++ IDE, likely Visual Studio, showing the editing of a file named `main.cpp`. The IDE interface includes a menu bar at the top with options like File, Edit, View, Git, Project, Build, Debug, Test, Tools, Extensions, Window, Help, and Search. The status bar at the bottom indicates the current file is `main.cpp`, the build configuration is `Debug`, and the architecture is `x64`. The code is written in C++ and uses the `filesystem` library for directory traversal.

The first screenshot shows the code from line 284 to 309. It defines a recursive function `recursive_directory_iterator` to traverse a directory. The code uses `fs::recursive_directory_iterator` to iterate over files and directories. It checks if the entry is a regular file and then pushes the file path and name to vectors `original_scg_file_paths` and `original_scg_file_names`. It also checks for file permissions and modifies the file name if necessary.

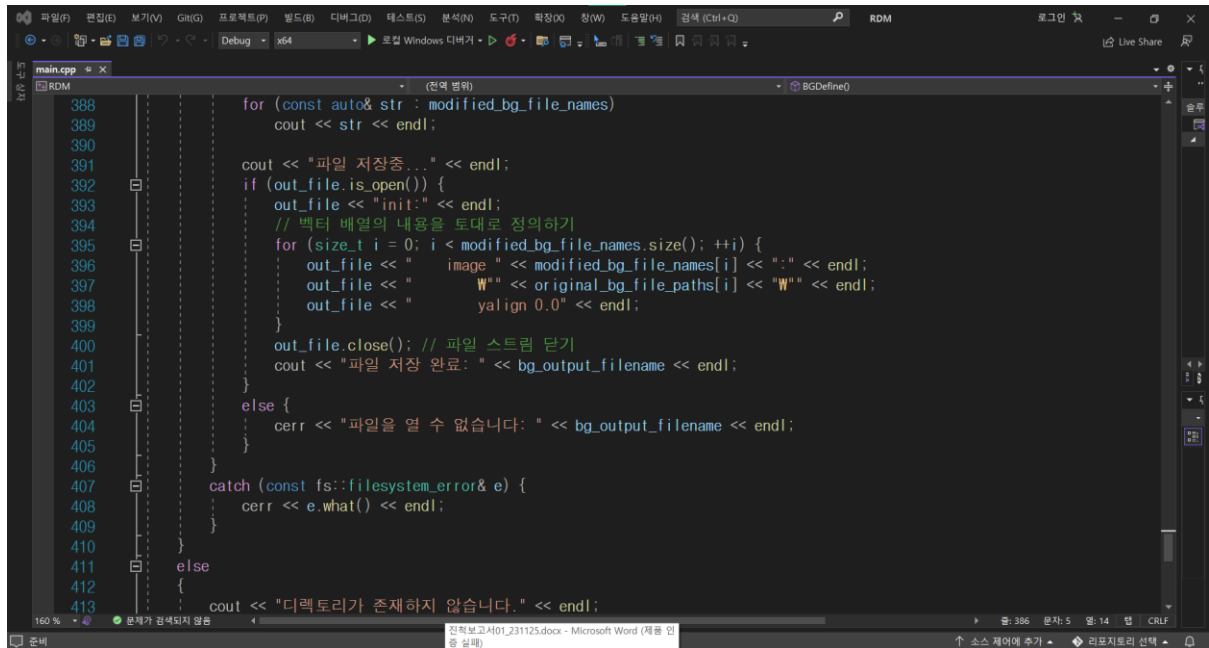
```
284 for (const auto& entry : fs::recursive_directory_iterator(scg_dir)) {
285     if (fs::is_regular_file(entry)) {
286         // 파일 경로와 제목을 문자열로 변환하여 벡터에 추가.
287         // filename()으로 파일명 추출, stem()으로 확장자 제거.
288         original_scg_file_paths.push_back(entry.path().string());
289         original_scg_file_names.push_back(entry.path().filename().stem().string());
290     }
291     // 파일 경로 출력
292     cout << entry.path() << endl;
293 }
294 cout << "파일명을 랜파이 인식 가능 버전으로 전환중..." << endl;
295 for (const auto& str : original_scg_file_names) {
296     string modified = str;
297
298     // 파일 제목에서 랜파이가 인식 불가능한 공백과 하이픈을 언더바로 변경
299     for (char& c : modified) {
300         if (c == ' ')
301             c = '_';
302         else if (c == '-')
303             c = '_';
304     }
305     // 변경된 문자열을 새 벡터에 추가
306     modified_scg_file_names.push_back(modified);
307 }
308
309 // 변경된 파일 제목 문자열을 첫 번째 언더바를 기준으로 절단
```

The second screenshot shows the code from line 310 to 335. It continues the processing of the file names, splitting them into two parts based on the first underscore. It then iterates over the modified file names and prints them. Finally, it closes the output file and prints a completion message.

```
310 for (const auto& str : modified_scg_file_names) {
311     size_t underbar_cut = str.find("_"); // 첫 번째 언더바의 위치를 찾음
312     if (underbar_cut != std::string::npos) {
313         modified_first_file_names.push_back(str.substr(0, underbar_cut)); // 첫 번째 부분
314         modified_second_file_names.push_back(str.substr(underbar_cut + 1)); // 두 번째 부분
315     }
316 }
317
318 // 수정된 파일명 출력
319 for (const auto& str : modified_scg_file_names)
320     cout << str << endl;
321
322 cout << "파일 저장중..." << endl;
323 if (out_file.is_open()) {
324     out_file << "init:" << endl;
325     // 벡터 배열의 내용을 토대로 정의하기
326     for (size_t i = 0; i < modified_scg_file_names.size(); ++i) {
327         out_file << "    image " << modified_first_file_names[i] << modified_second_file_names[i] << ":" << endl;
328         out_file << "        W" << original_scg_file_paths[i] << "W" << endl;
329         out_file << "        yalign 0.0" << endl;
330     }
331     out_file.close(); // 파일 스트림 닫기
332     cout << "파일 저장 완료: " << scg_output_filename << endl;
333 }
334 else {
335     cerr << "파일을 열 수 없습니다: " << scg_output_filename << endl;
```

```
336 }
337 }
338 catch (const fs::filesystem_error& e) {
339     cerr << e.what() << endl;
340 }
341 }
342 else
343 {
344     cout << "디렉토리가 존재하지 않습니다." << endl;
345     cout << "메인으로 돌아갑니다." << endl;
346 }
347 }
348
349 void BGDefine()
350 {
351     vector<string> original_bg_file_paths;
352     vector<string> original_bg_file_names;
353     vector<string> modified_bg_file_names;
354     string bg_output_filename = "bg.rpy";
355     ofstream out_file(bg_output_filename);
356
357     fs::path bg_dir = "./renpy_example/game/images/BG/"; // 문서 경로 지정
358     if (fs::exists(bg_dir) && fs::is_directory(bg_dir)) // 해당 디렉토리 존재 여부 확인
359     {
360         try {
361             // 디렉토리 내의 모든 파일과 하위 디렉토리를 재귀적 순회
```

```
362 cout << "원본 경로 찾는중..." << endl;
363 for (const auto& entry : fs::recursive_directory_iterator(bg_dir)) {
364     if (fs::is_regular_file(entry)) {
365         // 파일 경로와 제목을 문자열로 변환하여 벡터에 추가.
366         // filename()으로 파일명 추출, stem()으로 확장자 제거.
367         original_bg_file_paths.push_back(entry.path().string());
368         original_bg_file_names.push_back(entry.path().filename().stem().string());
369     }
370     // 파일 경로 출력
371     cout << entry.path() << endl;
372 }
373 cout << "파일명을 랜파이 인식 가능 버전으로 전환중..." << endl;
374 for (const auto& str : original_bg_file_names) {
375     string modified = str;
376
377     // 파일 제목에서 랜파이가 인식 불가능한 공백과 하이픈을 언더바로 변경
378     for (char& c : modified) {
379         if (c == ' ')
380             c = '_';
381         else if (c == '-')
382             c = '_';
383     }
384     // 변경된 문자열을 새 벡터에 추가
385     modified_bg_file_names.push_back(modified);
386
387     // 수정된 파일명 출력
```



```
388 for (const auto& str : modified_bg_file_names)
389     cout << str << endl;
390
391 cout << "파일 저장중..." << endl;
392 if (out_file.is_open()) {
393     out_file << "init:" << endl;
394     // 벡터 배열의 내용을 토대로 정의하기
395     for (size_t i = 0; i < modified_bg_file_names.size(); ++i) {
396         out_file << "    image " << modified_bg_file_names[i] << ":" << endl;
397         out_file << "        W" << original_bg_file_paths[i] << "W" << endl;
398         out_file << "        yalign 0.0" << endl;
399     }
400     out_file.close(); // 파일 스트림 닫기
401     cout << "파일 저장 완료: " << bg_output_filename << endl;
402 }
403 else {
404     cerr << "파일을 열 수 없습니다: " << bg_output_filename << endl;
405 }
406
407 catch (const fs::filesystem_error& e) {
408     cerr << e.what() << endl;
409 }
410
411 else
412 {
413     cout << "디렉토리가 존재하지 않습니다." << endl;
414 }
```

2) 테스트 결과

(1) 리소스 자동 정의

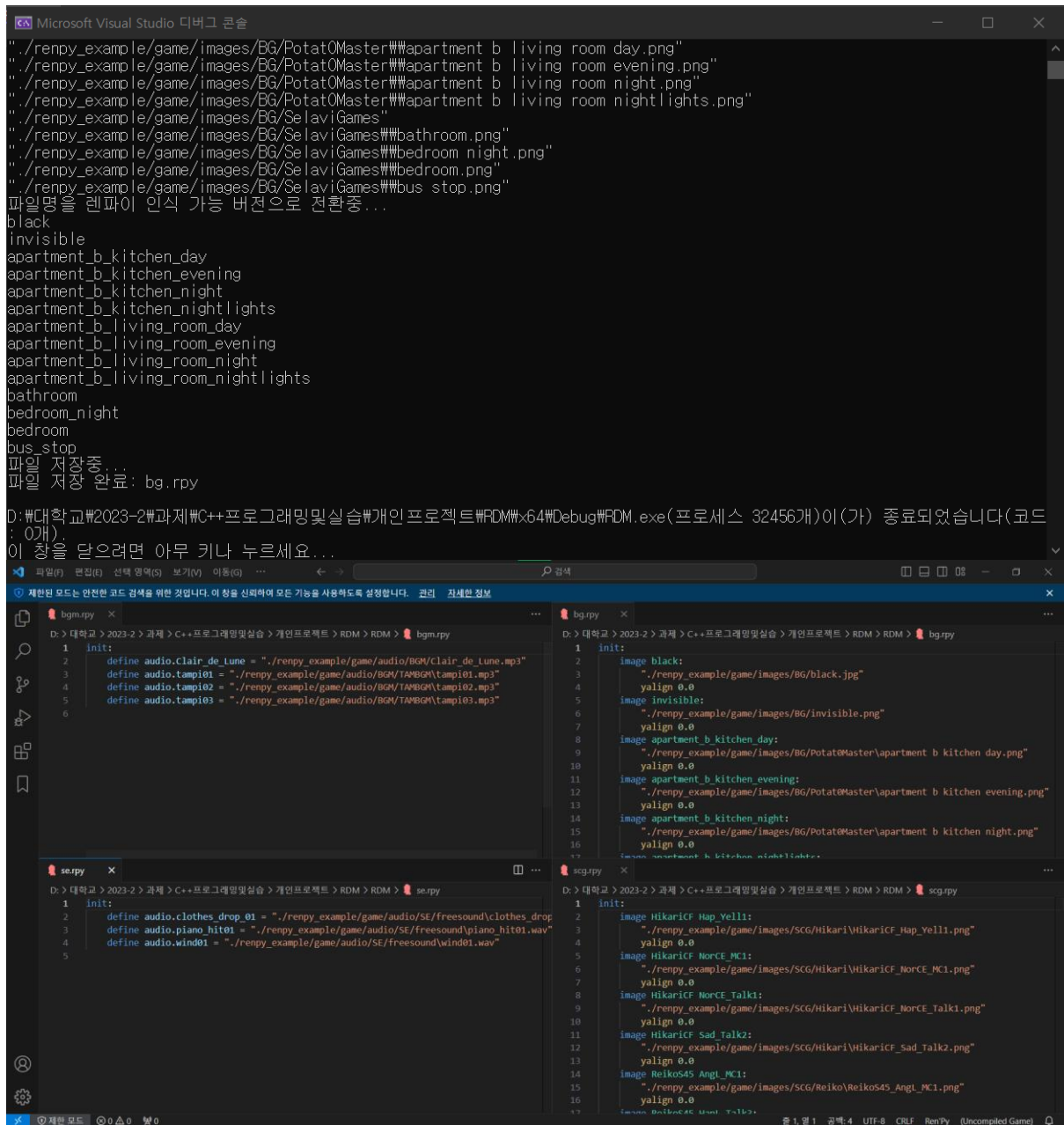
- 설명

리소스 자동 정의를 수행하면 콘솔창에는 리소스의 경로와 파일명들이 출력된다. 수행 뒤에는 bgm.rpy, se.rpy, scg.rpy, bg.rpy의 4개의 파일이 생겨난다.

- 테스트 결과 스크린샷

```
Microsoft Visual Studio 디버거 콘솔
렌파이 전용 리소스 관리 모듈입니다.
해야 할 작업의 번호를 입력해 주세요.
1. 전체 재정의
2. 부분 재정의
3. 사용된 리소스 검출
1
전체 재정의 작업을 수행합니다.
원본 경로 찾는중...
"./renpy_example/game/audio/BGM/Clair_de_Lune.mp3"
"./renpy_example/game/audio/BGM/TAMBGM"
"./renpy_example/game/audio/BGM/TAMBGM###tampi01.mp3"
"./renpy_example/game/audio/BGM/TAMBGM###tampi02.mp3"
"./renpy_example/game/audio/BGM/TAMBGM###tampi03.mp3"
파일명을 렌파이 인식 가능 버전으로 전환중...
Clair_de_Lune
tampi01
tampi02
tampi03
파일 저장중...
파일 저장 완료: bgm.rpy
원본 경로 찾는중...
"./renpy_example/game/audio/SE/freesound"
"./renpy_example/game/audio/SE/freesound###clothes_drop_01.mp3"
"./renpy_example/game/audio/SE/freesound###piano_hit01.wav"
"./renpy_example/game/audio/SE/freesound###wind01.wav"
파일명을 렌파이 인식 가능 버전으로 전환중...
clothes_drop_01
piano_hit01
wind01
파일 저장중...

D:\₩대학교₩2023-2₩과제₩C++ 프로그램및실습₩개인프로젝트₩RDM₩x64₩Debug₩RDM.exe
렌파이 전용 리소스 관리 모듈입니다.
해야 할 작업의 번호를 입력해 주세요.
1. 전체 재정의
2. 부분 재정의
3. 사용된 리소스 검출
```



4. 계획 대비 변경 사항

없음.

5. 프로젝트 일정

현재까지 리소스 정의 부분의 대부분은 완성되었다. 하지만 아직 보충할 부분이 있다고

판단하여, 완료 상태로 표시하지는 않는다.

업무	11/3	11/26	12/4	12/23
제안서 작성	완료			
기능1		진행 중		
기능2			----->	
디버그				----->