**2020년 1학기 컴퓨터공학 설계 및 실험1 기말 프로젝트 보고서**

컴퓨터공학 20172141 김미소

**1. 프로젝트 목표**

이전에 실습했던 waterfall 프로젝트를 창의적으로 응용하여 새로운 프로젝트를 진행한다. waterfall 프로젝트를 응용하여 핀볼(pinball) 게임을 직접 제작/구현한다.

**2. 변수에 대한 설명**

|  |  |
| --- | --- |
| struct Ball {  float x;  float y;  float vx;  float vy;  }; | 핀볼 게임의 공의 위치(x, y), 얼마나 움직일 것인지(vx, vy)에 대한 정보를 담는 구조체이다. |
| struct Bar {  int START\_X;  int START\_Y;  int END\_X;  int END\_Y;  float slope = 0;  }; | 핀볼 게임의 막대에 대한 정보를 담는 구조체이다. ofDrawLine 함수로 그릴 것이기 때문에 시작점의 x, y좌표와 끝 점의 x, y 좌표가 필요하다. 그리고 기울기까지 구조체 멤버에 포함된다. |
| struct obj {  int x, y;  }; | input파일을 통해 입력되어 생성되는 오브젝트에 대한 정보를 담는 구조체이다. 어느 좌표에 오브젝트를 그릴 것인지에 대한 정보가 담겨있다. |
| typedef struct \_Node {  char name[NAMELEN];  int score;  struct \_Node \*link;  }Node; | 랭킹 시스템 구현을 위해 랭킹 리스트를 만들 때 사용할 구조체이다. 랭킹 리스트는 연결리스트로 구현되며 각 노드의 멤버들이 이 3가지로 구성되어 있다. |
| // 플래그 변수  int draw\_flag : ‘d’ 키를 눌렀을 때 활성화된다. draw함수를 실행할 수 있게 해준다.  int load\_flag : ‘l’키를 눌러 파일을 로드했을 때 활성화된다.  int s\_pressed : ‘s’ 키를 눌렀을 때 활성화된다. 공을 그리는데 사용된다.  int left\_pressed : ‘z’ 키를 눌렀을 때 활성화된다. 왼쪽 막대를 위로 올리는 데에 사용되는 플래그이다.  int right\_pressed : ‘/’ 키를 눌렀을 때 활성화된다. 오른쪽 막대를 위로 올리는 데에 사용되는 플래그이다.  int die\_flag : 게임이 오버되었을 때 활성화된다. | |
| int num\_of\_obj : input파일로부터 읽어 들인 오브젝트의 개수를 나타낸다.  Ball whiteBall : 핀볼 게임의 공 변수이다.  Bar right\_bar : 핀볼 게임의 오른쪽 막대 변수이다.  Bar left\_bar : 핀볼 게임의 왼쪽 막대 변수이다.  obj \*objlist : input파일로부터 읽어 들인 오브젝트들을 저장할 구조체 배열이다. | |
| int score = 0; : 핀볼 게임 점수에 대한 변수이다. 0으로 초기화 되어있다.  int ranknum = 0; : 랭킹에 등록된 사람이 몇 명인지 알려주는 변수이다.  int from\_X : 공이 현재 위치에 있기 바로 이전에 어떤 위치에 있었는지 알려주는 변수이다. 이 변수를 선언한 이유는 공이 막대에 의해 튕겨져 나갔을 때, ↙방향에서 ↗방향으로 또는 ↘방향에서 ↖방향으로 튕겨 나가는 것이 부자연스럽다고 판단되어 선언하였다. 따라서 ↙방향에서 ↖방향으로 또는 ↘방향에서 ↗방향으로 튕겨져 나가는 것을 의도하기 위하여 선언하였다.  Node \*rankhead = NULL; : 랭킹 리스트를 연결리스트로 구현하기 위하여 선언한 노드 포인터 변수이다. | |

**3. 자료구조 및 알고리즘**

**[자료구조]**

핀볼 게임을 하는 데에 필요한 공, 막대, 오브젝트들은 모두 구조체를 이용하여 구현하였으며 랭킹 시스템은 복잡도를 고려하여 연결 리스트로 구현하였다.

**[알고리즘]**

게임의 전체적인 흐름을 설명하겠다. user defined function에 대한 자세한 설명은 4번 항목에서 하겠다. 보통 창의적이거나 복잡한 계산은 user defined function에서 수행하였기 때문이다.

1) ‘l’ 키를 눌러 load\_flag 활성화한다.

2) processOpenFileSelection 함수를 통해 파일 로드 후 오브젝트 정보를 읽는다.

3) ‘d’ 키를 눌러 draw\_flag를 활성화하여 draw함수를 실행한다.

4) vx, vy만큼 포지션이 업데이트 되어 draw된다. 이 때, 가장자리나 오브젝트에 부딪히게 되면 vx나 vy값의 부호를 바꿔서 반대 방향으로 튕겨져 나갈 수 있도록 한다. 그리고 어디에 부딪혔는지에 따라 점수를 알맞게 증가시킨다.

5) getReflection 함수를 통해 막대에 부딪혔을 때 방향 조절과 속도 조절을 수행한다. 그리고 점수를 증가시킨다.

6) 4~5단계를 반복하여 게임이 오버될 때까지 진행한다.

7) 게임이 오버되면 ‘r’ 키를 눌러 랭킹 파일에 점수를 등록한다. 랭킹 파일은 프로젝트 폴더의 bin폴더 안에 저장된다.

8) 랭킹을 등록하고 난 후나 ‘q’ 키를 눌러 프로그램을 종료할 때는 메모리를 할당 해제한 후 프로그램을 종료한다.

**4. 함수에 대한 설명**

**1) processOpenFileSelection 함수**

|  |
| --- |
| void ofApp::processOpenFileSelection(ofFileDialogResult openFileResult) {  //Path to the comma delimited file    string fileName = openFileResult.getName();  ofFile file(fileName);    if( !file.exists()) cout << "Target file does not exists." << endl;  else cout << "We found the target file." << endl;    ofBuffer buffer(file);  int obj\_idx = 0;  // Read file line by line  for (ofBuffer::Line it = buffer.getLines().begin(), end = buffer.getLines().end(); it != end; ++it) {  string line = \*it;    // Split line into strings  vector<string> words = ofSplitString(line, " ");    if( words.size() == 1){  // 오브젝트의 개수를 받아옵니다.  num\_of\_obj = atoi(words[0].c\_str());  cout << "The number of object is: " << num\_of\_obj << endl;  objlist = (obj \*)malloc(sizeof(obj)\*num\_of\_obj); // 오브젝트의 개수만큼 오브젝트 리스트를 만듭니다. (구조체배열)  }  else if (words.size() >= 2){  int x1,y1;  // 오브젝트의 x, y위치를 읽어와서 오브젝트 리스트에 기록합니다.  x1 = atoi(words[0].c\_str());  y1 = atoi(words[1].c\_str());  objlist[obj\_idx].x = x1;  objlist[obj\_idx++].y = y1;  } // End of else if.  } // End of for-loop (Read file line by line).  } |

input 파일을 읽어오는 함수이다. input 파일 안에는 오브젝트의 개수, 각 오브젝트의 x위치, y위치가 담겨있다. ofBuffer 함수로 파일을 줄 단위로 받아오며 띄어쓰기를 단위로 word를 받아온다. words.size()가 1인 경우에는 오브젝트의 개수를 나타내는 것일 것이므로 num\_of\_obj 변수에 개수를 저장하고 그 개수만큼 구조체 배열 objlist를 할당한다. words.size()가 2 이상인 경우에는 x, y 위치에 대한 정보일 것이므로 각 정보를 받아 임시적으로 변수에 저장한 후 objlist 배열의 멤버에 최종적으로 저장한다. 파일 입력을 다 받은 후에는 함수가 종료된다.

**2) getReflection 함수**

|  |
| --- |
| void ofApp::getReflection() {  float y;  // 공이 왼쪽 막대와 닿는지 확인합니다.  if (whiteBall.x <= left\_bar.END\_X && whiteBall.x >= left\_bar.START\_X) {  left\_bar.slope = (float)(left\_bar.END\_Y - left\_bar.START\_Y) / (left\_bar.END\_X - left\_bar.START\_X); // 왼쪽 막대의 기울기  y = left\_bar.slope \* (whiteBall.x - left\_bar.START\_X) + left\_bar.START\_Y;  // 공의 x위치를 이용해서 막대에 닿았을 때의 y좌표를 계산합니다.  if (y < whiteBall.y && !die\_flag) {  // 게임이 오버되지 않았고 공이 튕겨져야 할 때  whiteBall.y = y;  if (!left\_pressed) {  // 키를 눌러 올려치지 않았을 때는 속도가 느려지고 점수는 10점만 받습니다.  whiteBall.vy = -whiteBall.vy \* 0.8;  score += 10;  }  else {  // 키를 눌러 올려 쳤을 때는 속도는 빨라지고 점수를 300점을 얻습니다.  if(from\_X <= whiteBall.x) // 방향 조절을 위한 조건  whiteBall.vx = -whiteBall.vx;  whiteBall.vy = -whiteBall.vy \* (1 + abs(left\_bar.slope));  score += 300;  }  }  }  // 공이 오른쪽 막대와 닿는지 확인합니다.  if (whiteBall.x <= right\_bar.START\_X && whiteBall.x >= right\_bar.END\_X) {  right\_bar.slope = (float)(right\_bar.END\_Y - right\_bar.START\_Y) / (right\_bar.END\_X - right\_bar.START\_X); // 오른쪽 막대의 기울기  y = right\_bar.slope \* (whiteBall.x - right\_bar.START\_X) + right\_bar.START\_Y; // 공의 x위치를 이용해서 막대에 닿았을 때의 y좌표를 계산합니다.  if (y < whiteBall.y && !die\_flag) {  // 게임이 오버되지 않았고 공이 튕겨져야 할 때  whiteBall.y = y;  if (!right\_pressed) {  // 키를 눌러 올려치지 않았을 때는 속도가 느려지고 점수는 10점만 받습니다.  whiteBall.vy = -whiteBall.vy \* 0.8;  score += 10;  }  else {  // 키를 눌러 올려 쳤을 때는 속도는 빨라지고 점수를 300점을 얻습니다.  if (from\_X >= whiteBall.x) // 방향 조절을 위한 조건  whiteBall.vx = -whiteBall.vx;  whiteBall.vy = -whiteBall.vy \* (1 + abs(right\_bar.slope));  score += 300;  }  }  }  } |

공이 막대에 닿았는지 확인하고 닿았다면 반사될 방향과 속도를 조절하는 함수이다. 우선 왼쪽, 오른쪽 막대의 시작점 좌표와 끝점 좌표를 이용하여 막대의 기울기를 구하여야 한다. 이 기울기를 이용하여 막대의 직선의 방정식을 세운다. y라는 변수가 하는 역할은 그 직선의 방정식을 통하여 현재 x위치에서 막대 위에서의 y좌표를 구하는 역할을 한다. 이 y값과 whiteBall.y를 비교하여 막대에 부딪혀 튕겨져야 하는지 비교한다. 왼쪽 오른쪽 막대의 pressed 정보를 활용하여 left\_pressed나 right\_pressed가 활성화되었을 때에는 반대 방향으로 속도를 증가시켜서 공을 튕긴다. pressed가 활성화되지 않았을 때는 반대 방향으로 속도를 20% 늦춰서 공을 튕긴다.

**3) ReadRank 함수**

|  |
| --- |
| void ofApp::ReadRank() {  FILE \*fp = fopen("rank.txt", "r"); // rank.txt 파일을 읽습니다.  int tmpscore;  if (fp == NULL) { // 파일이 없다면 생성합니다.  ranknum = 0;  fp = fopen("rank.txt", "w");  fprintf(fp, "%d", 0);  }  else {  // 파일이 존재한다면 랭크 리스트를 읽어 연결리스트를 생성합니다.  fscanf(fp, "%d", &ranknum);  int i = 0;  for (i = 0; i < ranknum; i++) {  Node \*newnode = NULL;  newnode = (Node \*)malloc(sizeof(Node));  fscanf(fp, "%s %d", newnode->name, &tmpscore);  newnode->score = tmpscore;  newnode->link = NULL;  Node \*tmp;  if (rankhead == NULL) {  rankhead = newnode;  }  else {  tmp = rankhead;  while (tmp->link != NULL) {  tmp = tmp->link;  }  tmp->link = newnode;  }  }  }  fclose(fp);  } |

파일 포인터를 통해 rank.txt를 오픈한다. 만약 파일이 없어서 파일 포인터가 NULL인 경우에는 rank.txt 파일을 만들어서 랭크 개수 0을 파일 출력한 후 닫고 다시 ‘r’모드로 오픈한다. 파일 포인터가 NULL이 아닌 경우에는 랭크 개수 ranknum을 읽어들인 후 랭크 개수만큼 newnode를 할당하고 정보를 저장한다. 첫번째 랭크를 읽을 때는 rankhead가 NULL인 상태이므로 rankhead = newnode;처리를 한다. 그 이후부터는 맨 마지막 노드를 찾은 후 그 노드 뒤에 newnode를 연결시킨다.

**4) MakeRank 함수**

|  |
| --- |
| void ofApp::MakeRank(int score) {  ReadRank(); // rank.txt를 읽어 랭크 리스트를 받아옵니다.  Node \*newnode = NULL;  newnode = (Node \*)malloc(sizeof(Node));  printf("Your name: "); // 플레이어의 이름을 입력받고 이름과 점수를 랭크 리스트의 올바른 위치에 삽입합니다.  scanf("%s", newnode->name);  newnode->score = score;  newnode->link = NULL;  Node \*pre = rankhead, \*tmp = rankhead;  int flag = 0;  if (rankhead == NULL) { // 랭크 리스트가 없을 때  rankhead = newnode;  }  else { // 랭크 리스트가 존재하면 알맞은 위치를 찾아 삽입합니다.  if (newnode->score >= rankhead->score) {  newnode->link = rankhead;  rankhead = newnode;  }  else {  while (tmp->link != NULL) {  if (newnode->score >= tmp->score && tmp->link != NULL) {  pre->link = newnode;  newnode->link = tmp;  flag = 1;  break;  }  pre = tmp;  tmp = tmp->link;  }  if (flag == 0 && newnode->score <= tmp->score) {  tmp->link = newnode;  }  else if (flag == 0 && newnode->score > tmp->score) {  pre->link = newnode;  newnode->link = tmp;  }  }  }  ranknum++;  WriteRank(); // 저장한 랭크 리스트를 파일 출력합니다.  } |

랭크를 추가하는 함수이다. 우선적으로 ReadRank함수를 호출하여 기존 랭크를 읽어 랭크 리스트를 만든 후 사용자가 ‘r’키를 눌렀을 때 이름을 입력받는다. 플레이어의 이름을 입력 받은 후에는 newnode->name에 저장하고 파라미터로 전달받은 score는 newnode->score에 저장한다. 그리고 newnode->score와 랭크 리스트 노드들의 score의 대소비교를 통해 newnode가 삽입되어야 할 위치를 찾아 노드를 연결한다. 그 후에 ranknum을 증가시켜서 랭크 개수를 갱신하고 WriteRank함수를 호출하여 파일 출력할 수 있도록 한다.

**5) WriteRank 함수**

|  |
| --- |
| void ofApp::WriteRank() {  // rank.txt파일로 랭킹을 출력합니다.  FILE \*fp = fopen("rank.txt", "w");  int i = 0;  Node \*curr = rankhead;  fprintf(fp, "%d\n", ranknum);  for (i = 0; i < ranknum; i++) {  fprintf(fp, "%s %d\n", curr->name, curr->score);  curr = curr->link;  }  fclose(fp);  freeMemory(); // 종료하기 전에 메모리를 할당 해제합니다.  cout << "Your score has been recorded." << endl;  cout << "Dynamically allocated memory has been freed." << endl;  \_Exit(0);  } |

랭크 리스트를 파일 출력하는 함수이다. 파일 포인터를 통해 rank.txt파일을 ‘w’모드로 오픈한다. 노드 포인터를 새로 생성하여 랭크 리스트의 노드를 하나하나 살펴볼 수 있도록 한다. 랭크 개수만큼 다음 노드로 넘어가면서 fprintf를 통하여 이름과 점수를 출력한다. 모두 다 출력한 후에는 freeMemory를 통해 메모리를 할당 해제하고 점수가 기록되었음을 알리고 프로그램을 종료한다.

**6) freeMemory 함수**

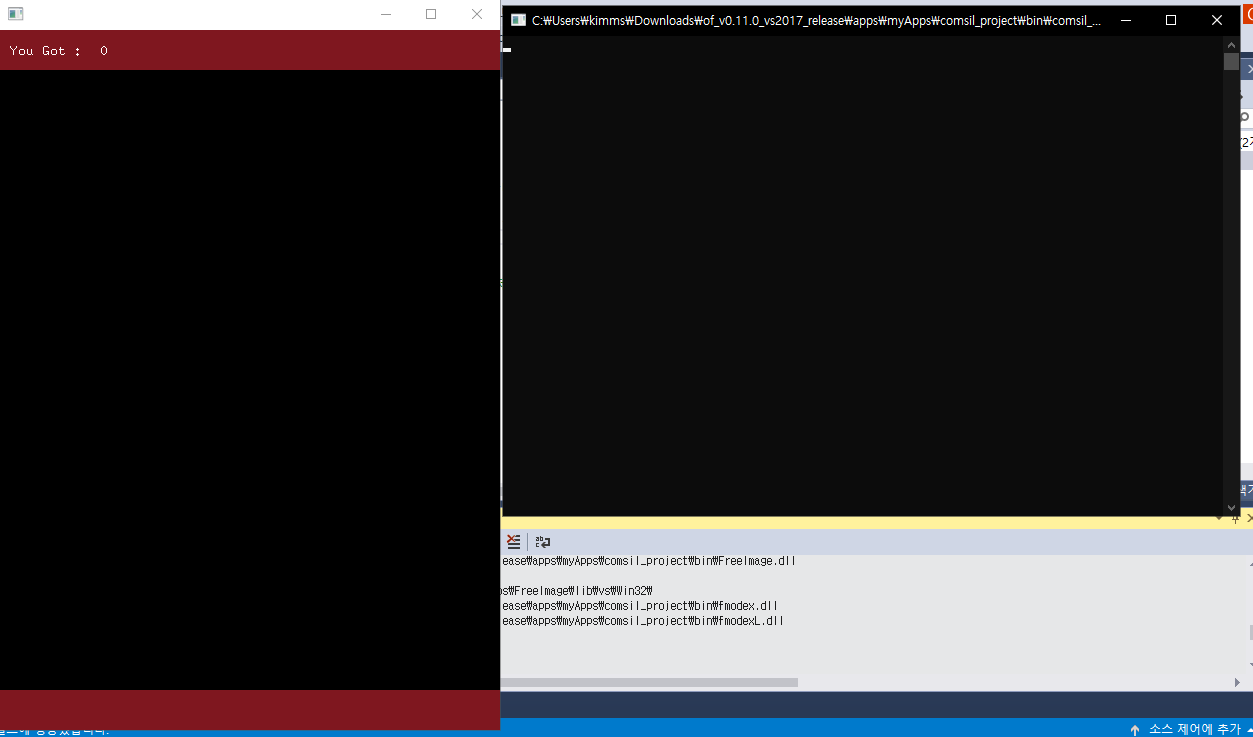
|  |
| --- |
| void ofApp::freeMemory() {  free(objlist); // 오브젝트 리스트(구조체 배열)를 해제합니다.  Node \*temp = rankhead; // 랭크 리스트를 해제합니다.  Node \*prev = temp;  if (rankhead) {  while (temp != NULL) {  temp = temp->link;  free(prev);  prev = temp;  }  }  } |

동적 할당한 메모리를 해제하는 함수이다. objlist는 1차원 구조체 배열이므로 단순히 free하고 랭크 리스트는 연결 리스트로 구현하였기 때문에 노드 포인터를 두개 선언하여 하나는 다음으로 넘어가는 용도로 사용하고 하나는 노드를 free하는 용도로 사용한다. 모든 노드를 할당한 후에는 함수가 종료된다.

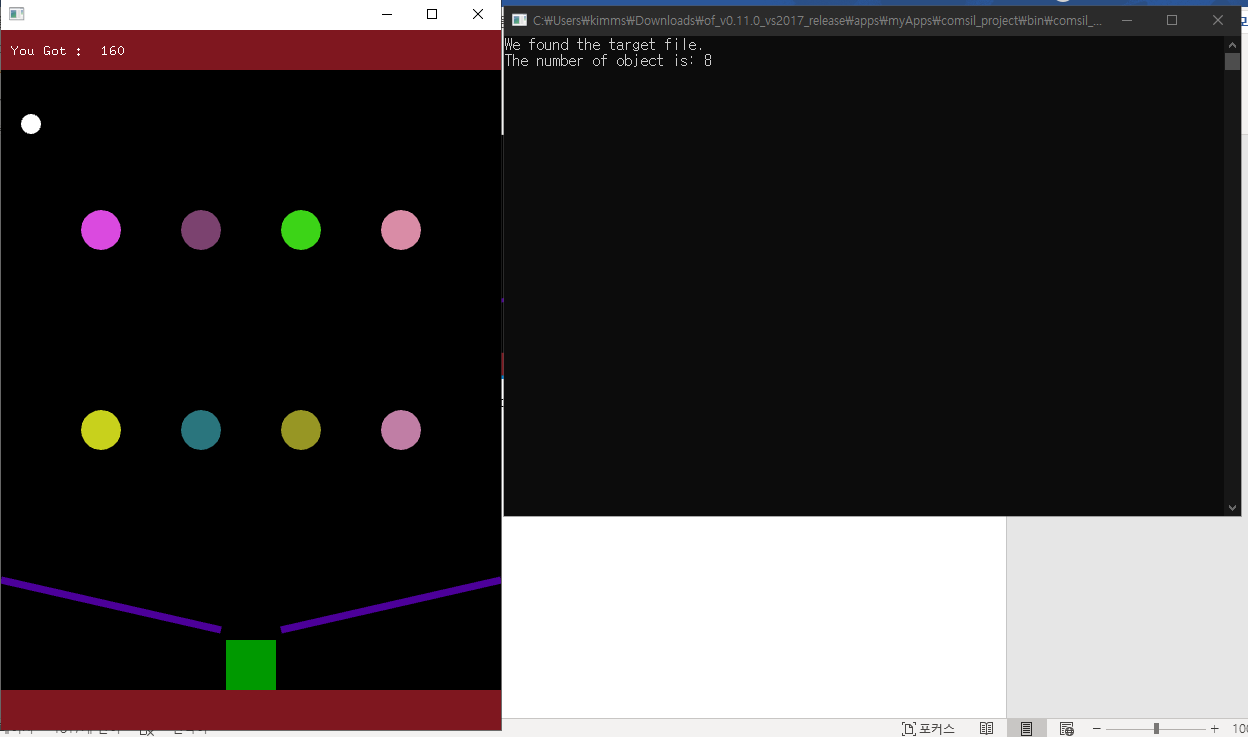
**5. 어떻게 창의적 구현을 하였는가?**

실험했던 waterfall이 오브젝트를 만났을 때 새로운 연산이 이뤄진다는 것을 이용하여 게임을 만들게 되었다. 오브젝트와의 상호작용을 생각했을 때 제일 먼저 떠오른 것이 핀볼 게임이었다. 따라서 핀볼 게임을 제작하게 되었다. 막대와 공의 상호작용은 직선의 방정식을 통해 구현하였으며 단순히 튕기기보다는 속도를 조절하여 난이도를 스스로 조절할 수 있게끔 구현하였다. 또 대부분의 게임들은 기본적으로 랭킹 시스템을 가지고 있으므로 게임이 끝났을 때 랭킹을 등록할 수 있도록 하였다. 이 때 랭킹시스템은 시간 복잡도를 고려하여 연결리스트로 구현할 수 있도록 하였다. 그리고 게임 속 오브젝트들은 input 파일을 사용자가 임의로 만들어서 플레이 할 수 있다. 이 프로젝트는 전반적으로 시간적으로나 공간적으로나 복잡하지 않은 알고리즘을 갖고 있다.

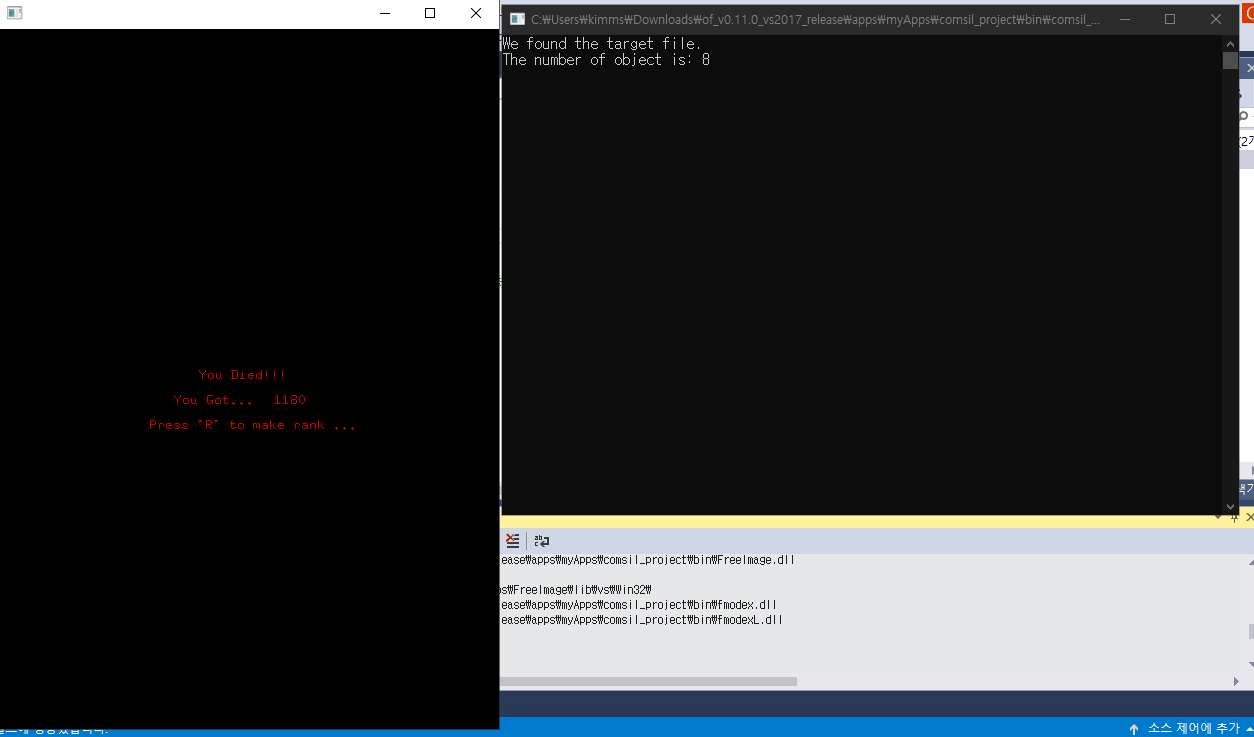
**6. 프로젝트 실행 결과**



<첫 실행 화면>



<플레이 화면>



<게임 오버 화면>

**7. 느낀 점 및 개선 사항**

이번 프로젝트를 하면서 확실하게 느낀 것은 스스로 처음부터 만드는 프로젝트여서 구현하는 내내 재미가 있었다는 것이다. 그리고 내가 기획한 기능이 제대로 구현되는 것들을 보면서 자신감과 자부심이 생겨나기도 했다. 그리고 구현하면서 부족한 점들도 많이 느끼게 되었고 그것들을 보완하는 기회를 가지게 되어서 좋았다. 이 프로젝트의 개선 사항은 완벽하게 모든 기능이 자연스럽지는 않다는 것이다. 튕겨져 나가는 것이 가끔씩은 부자연스러워 보인다. 이 사항을 개선하려 노력했지만 끝내 완벽하게 고쳐내지는 못했다. 그러나 이런 사항들을 개선하기 위해 계속해서 고민하면서 부족한 점도 느끼고 새롭게 알게 된 것들도 많아서 여러가지로 많은 것을 배운 프로젝트였다.

**8. 프로젝트 개발 환경**

코드 구현 : Visual Studio 2017

OpenFrameWorks 버전 0.11.0

Windows SDK 버전 : 10.0.17763.0