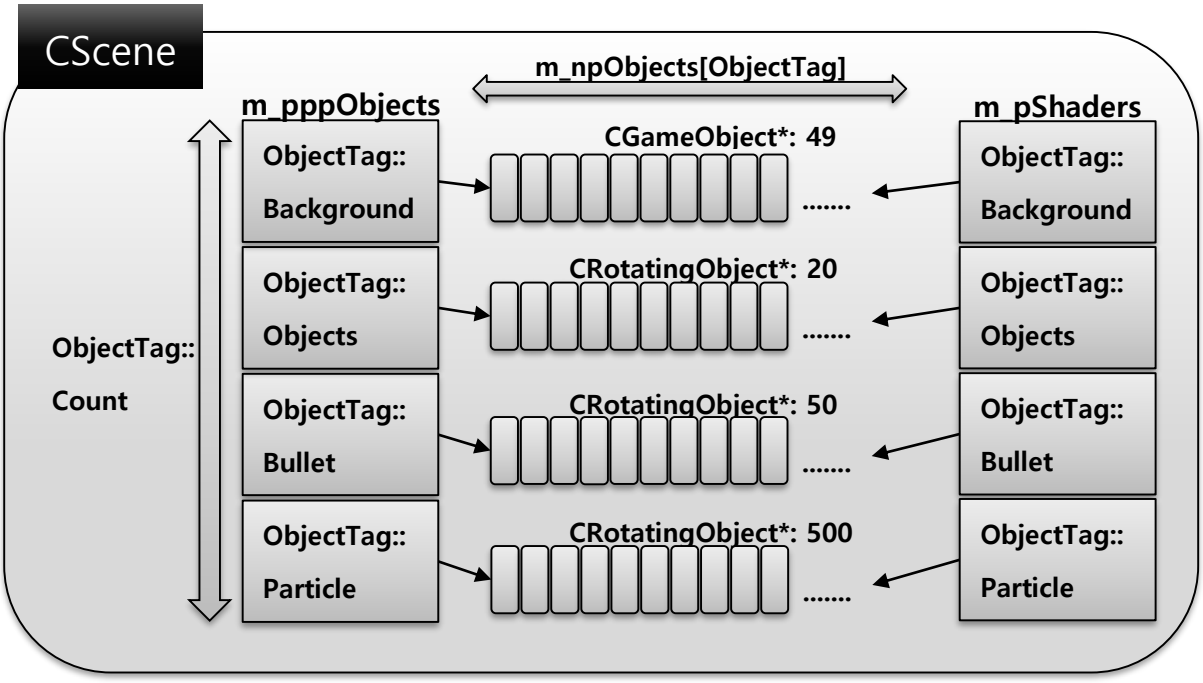
[과제 해결방법]

➢ [프로그램 요구사항]

1. 따라하기 15 의 지형 그리기 프로그램의 평평한 지형에 육면체로 미로를 짓고 p 를 누르면 플레이어가 미로의 입구로 이동한다.

- 가정
  ① 과제 03 은 따라하기 15 번 + 16, 17 번을 기반으로 진행한다.
  ② 미로의 크기는 20x20 블록이고 한 블록은 50^3 의 크기를 가지는 큐브 메쉬이다.
  ③ 플레이어 캐릭터와 미로의 지붕은 오브젝트셰이더 클래스를 사용하여 그렸고 다른 오브젝트들은 과제 02 와 비슷하게 그릴 때 오브젝트의 종류별로 인스턴싱셰이더 클래스를 사용한다.
  ④ 미로를 지을 때 미로 내부의 길 위치와 입구, 출구의 위치는 따로 저장해 두었다가 p 입력 및 적 오브젝트 생성할 경우 사용한다.

- 사용한 자료구조
  과제 02 와 마찬가지로 플레이어 캐릭터를 제외한 여러 오브젝트를 하나의 CGameObject 삼중포인터를 이용한 2 차원 포인터배열을 사용하여 관리한다.



- 구현

① Maze 구조체를 만들고 typedef enum 으로 미로의 출입구와 길, 벽을 구분하고 멤버 변수로 XMFLOAT3 2 개, XMFLOAT3 배열 하나와 int 형 배열 길이를 나타내는 변수를 두고 씬에서 Maze 구조체를 멤버 변수로 두고 BuildObjects 에서 ObjectTag::Background 에 해당하는 미로의 벽들을 만들 때 XMFLOAT3 변수 두 개에 각각 출입구 위치를 저장하고 구조체가 가지고있는 미로 데이터에서 미로의 길 부분의 블록 개수를 세서 배열 길이 변수에 저장하고 변수 값 만큼의 길이를 가진 XMFLOAT3 배열을 동적할당하여 이 배열에 길 블록의 중심 위치를 전부 저장하였다.

소스코드: Scene.cpp - CScene::BuildObjects(), stdafx.h – struct Maze

② 키보드 p 키를 누르면 플레이어의 위치를 저장해 두었던 미로의 입구 위치로 setposition()을 사용해 이동시켰다.

소스코드: Scene.cpp - CScene::ProcessInput()

③ 미로를 만들 때 큐브 메쉬의 크기는 미로 안에서 카메라가 가리는 것을 막기 위해 카메라 offset 과 같은 50 이며 Maze 구조체이 있는 데이터에 따라 인스턴싱을 사용하여 벽 오브젝트들을 생성하였고 일반 CObjectShader 를 사용하여 세로 50, 너비, 길이 1100 인 거대한 지붕을 만들어 미로 위에다 미로 벽의 윗부분과 살짝 겹치게 올려놓았다.

소스코드: Scene.cpp - CScene::BuildObjects()

④ 미로를 그릴 때 따라하기에 있는 것처럼 높이 맵에 따른 지형의 법선 벡터에 따라 벽을 회전시키지 않고 오브젝트의 색은 과제 02 과 마찬가지로 CInstancingShader::UpdateShaderVariables()에서 오브젝트의 m_Tag 값에 따라 더해지는 색을 다르게 하여 오브젝트가 원하는 색을 가지도록 하였다.

소스코드: Scene.cpp - CScene::BuildObjects()

Shaders.cpp - CInstancingShader::UpdateShaderVariables()

2. 미로 안에 적이 10 체 이상 있으며, 적 오브젝트는 플레이가 접근하면 플레이를 향해 이동한다. 플레이어가 총알을 발사하여 맞추면 파티클이 생성되면서 적이 사라진다. 적과 플레이어 캐릭터가 부딪히면 미로의 입구부터 다시 시작한다. 적에게 총알을 발사할 때 피킹을 사용한다.

- 가정
  ① 적 오브젝트는 총 30 체가 있으며 메쉬는 구형이고 붉은색이다.
  ② 적 오브젝트는 플레이어와의 거리가 150 보다 작으면 플레이어를 향해 움직인다.
  ③ 적 오브젝트는 임의의 방향으로 지속적으로 회전하며 벽에 충돌하면 벽 바깥쪽으로 튕겨난다..
  ④ 플레이어가 총알을 발사하고 적 오브젝트가 맞아서 파티클이 터지면서 사라지고 다시 생성되는 것은 과제 02 와 똑같다.
  ⑤ 적 오브젝트는 미로를 만들 때 길 위치를 저장해놓은 배열 길이로 랜덤 모듈러 연산을 통해 미로의 길 어딘가에서 생성 및 재생성된다.

- 구현
  ① 적 오브젝트와 총알, 적 오브젝트와 플레이어 캐릭터의 충돌처리 알고리즘은 과제 02 와 동일하며 사용하는 함수도 같고, 적 오브젝트의 바인딩박스는 따라하기에서 설정된 대로 사용하면 벽이랑 충돌했을 때 구보다 바인딩박스가 더 커서 어색하므로 구보다 작게 바꾸었다.
  소스코드: Scene.cpp - CScene::PhysicsProcessing()

  ② 매 프레임마다 적 오브젝트의 y 위치값을 지형의 높이 맵에 따라 업데이트 시켜주고 플레이어와 오브젝트의 위치를 빼서 벡터를 생성하고 xmvector3length 함수를 사용하여 길이를 구하고 길이가 150 보다 작으면 미리 생성한 벡터를 정규화하여 오브젝트의 이동방향으로 설정하고

길이가 150 보다 크면 오브젝트의 이동 방향 벡터를 영 벡터로 만들어 움직이지 않도록 한다.

Scene.cpp - CScene::AnimateObjects()

③ 적 오브젝트를 생성 및 재생성할 때 씬에 멤버 변수로 만들어놓은 m_Maze 가 가지고 있는 미로의 길 위치정보 배열에 배열 길이 변수를 사용하여 배열 내부의 임의의 위치정보를 얻고, 이것을 SetPosition() 함수를 사용하여 위치를 설정해준다.

소스코드: Scene.cpp - CScene::ResetObjects()

④ 플레이어 캐릭터를 리셋할 때 m_Maze 구조체가 가지고 있는 미로의 입구 위치로 캐릭터를 이동시킨다.

Scene.cpp - CScene::AnimateObjects()

⑤ 피킹은 기본적으로 따라하기 17 과 같으며 마우스 왼쪽 버튼이 떼어졌을 때 피킹을 수행하여 오브젝트를 찾는다. 이때 모든 오브젝트를 찾는 것이 아니라 적 오브젝트만 찾게 되고 찾으면 총알을 한발 발사하고, 총알이 발사되면 찾은 오브젝트를 초기화 시킨다. 총알을 발사하지 않더라도 찾은 오브젝트가 총알에 맞으면 재생성하기 전에 오브젝트 포인터를 초기화한다.

소스코드: Scene.cpp - CScene::OnProcessMouseMessage()

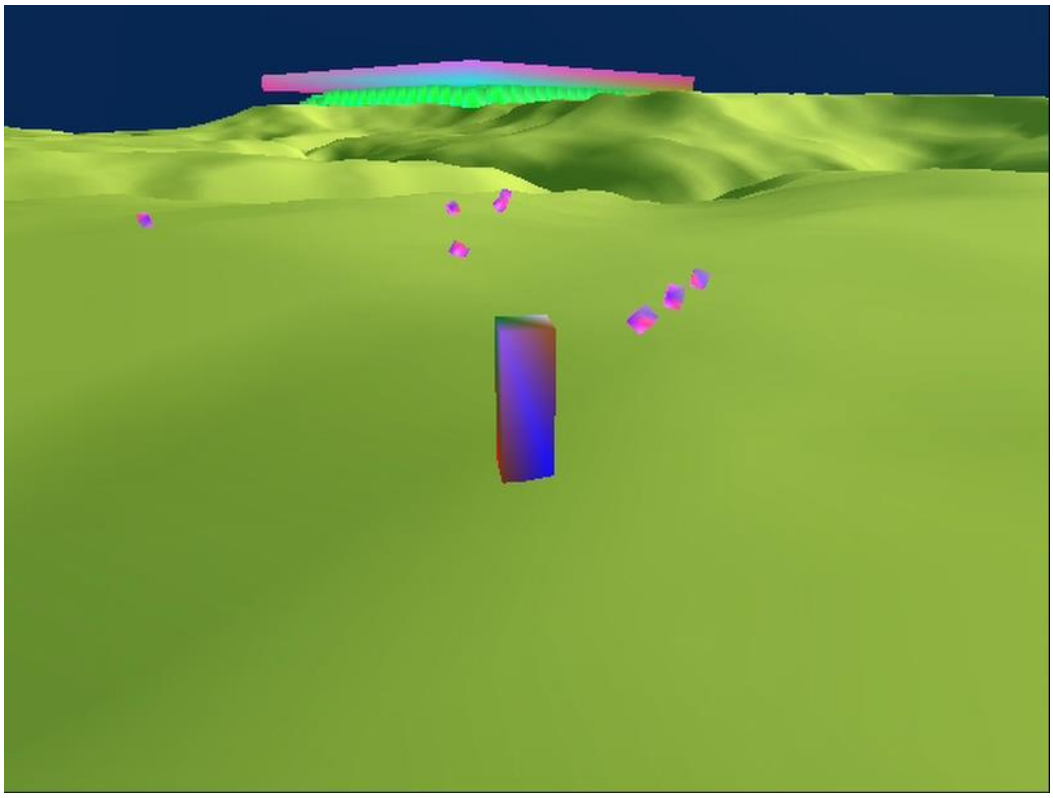
⑥ 총알 발사는 과제 02 와 같이 스페이스바로도 할 수 있다.

소스코드: Scene.cpp - CScene::ProcessInput()

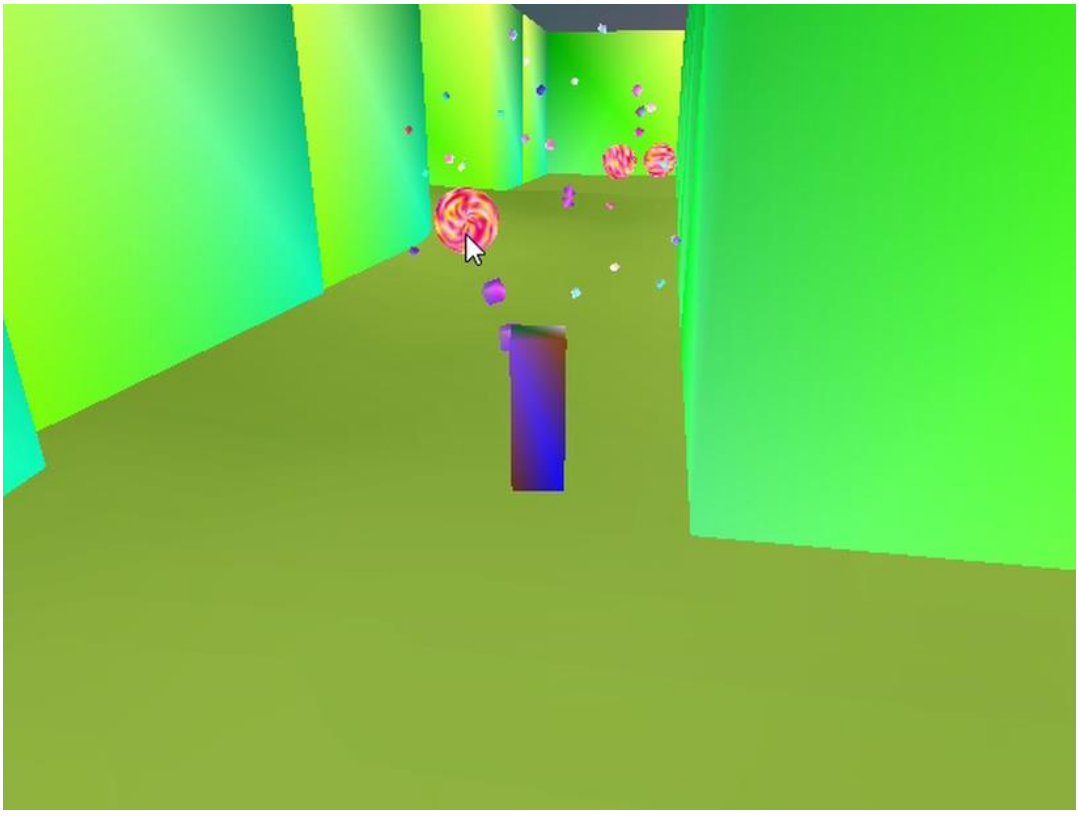
⑦ 파티클과 총알은 과제 02 와 같은 개수만큼 같은 방식으로 생성된다.

소스코드: Scene.cpp - CScene::BuildObjects()

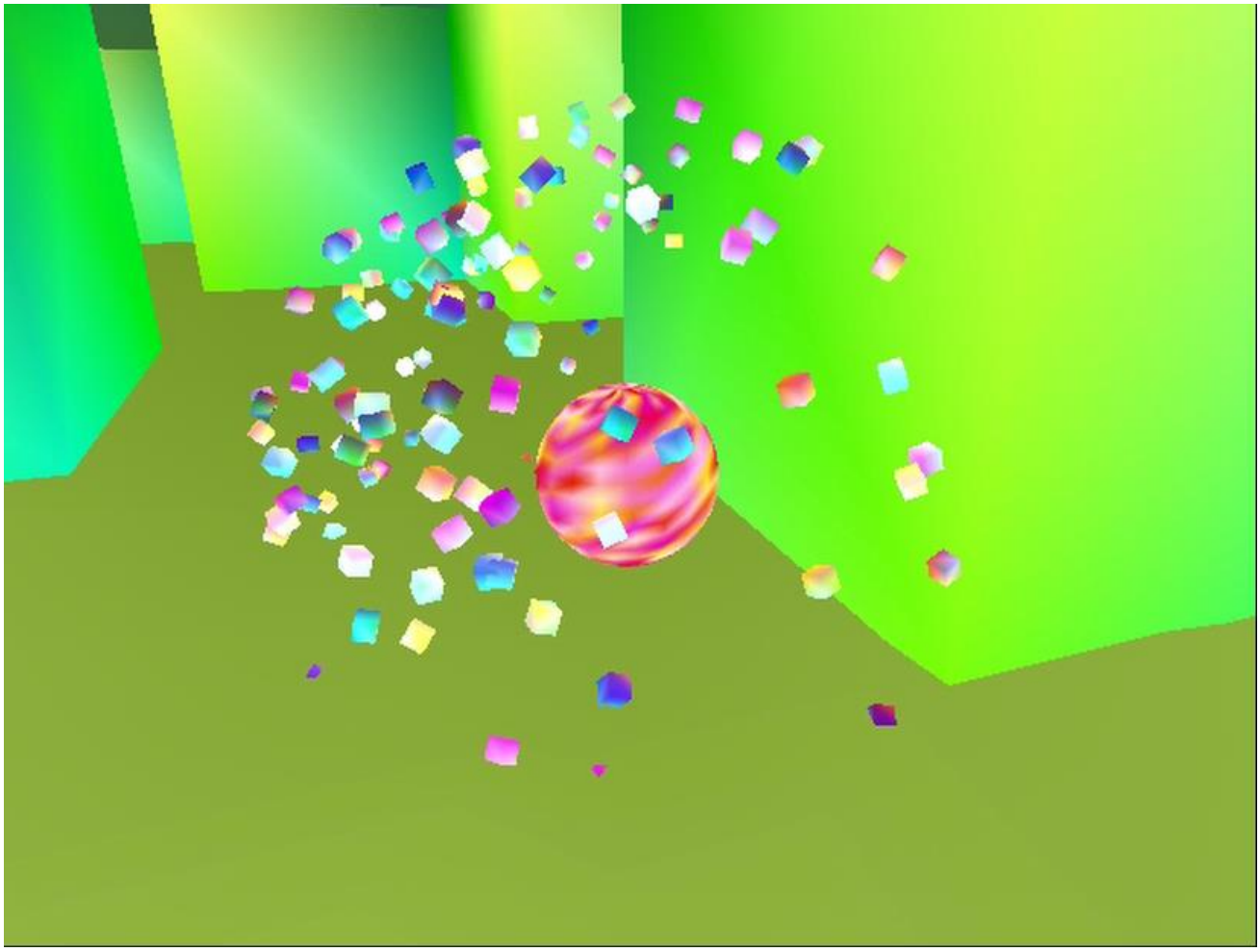
➢ [실행결과]



플레이 화면



오브젝트 파괴



게임 오버

[조작법]

키	설명
W	카메라가 바라보는 방향으로 **(앞으로) 이동**한다.
A	카메라가 바라보는 방향에서 **왼쪽으로 이동**한다.
S	카메라가 바라보는 방향의 반대로 **(뒤로) 이동**한다.
D	카메라가 바라보는 방향에서 **오른쪽으로 이동**한다.
마우스 왼쪽 버튼 떼기	마우스 커서가 적 오브젝트위에 있었다면 **적 오브젝트에게 탄환을 발사**한다.
P	플레이어의 위치를 **미로의 입구로 이동**한다.
마우스 왼쪽 버튼 + 좌우 상하 이동	**누른 상태**로 마우스를 움직이면 **카메라가 바라보는 방향이 마우스를 따라 움직인다.**
SPACE	플레이어가 카메라가 보는 방향으로 **탄환을 발사**한다.
ESC	**프로그램 종료**