[과제 해결방법]

* **[프로그램 요구사항]**

1. **끊임없이 이어지는 복도 형식의 벽을 구현하고 플레이어는 그 복도에서 앞으로 이동한다.**

* 가정

1. 끊임없이 이어지는 복도를 구현하기 위해 플레이어가 앞으로 이동했을 경우 복도를 구성하는 벽의 position을 z축 위에 늘어놓고 플레이어 기체의 z좌표보다 일정 거리 더 뒤에 있는 벽을 맨 앞으로 이동시킨다.
2. 벽 객체는 한번 생성한 후 프로그램이 종료될때까지 추가적인 생성 및 소멸이 일어나지 않는다.
3. 플레이어는 – z축 방향으로 이동할 수 없다.

* 사용한 자료구조

여러 개의 벽 오브젝트를 이용해 복도를 만들 때, 처음에 한 번 생성하고 프로그램이 종료될 때까지 제거되지 않으므로 STL의 벡터 컨테이너를 사용하여 복도를 구현하였다.

* 구현

1. 벽을 구성하고 있는 선이 화면 중앙으로 모이는 것을 줄이기 위해 CWallMesh의 생성자에서 벽을 구성하고 있는 사각형의 개수를 두배가 되도록 하드 코딩 되어있던 코드를 for문을 사용하여 정리하였습니다.

**소스코드: ModelData.cpp – CwallMesh:: CWallMesh() – 149 line**

1. 49개의 길이가 10.0f인 복도 벽을 플레이어 위치에 하나, 24개의 벽을 각각 플레이어의 앞, 뒤로 겹치지 않게 늘어놓았습니다.

**소스코드: MainScene.cpp - CMainScene::BuildObjects() - 715 line**

1. 매 프레임마다 가장 뒤에 있는 벽의 z좌표가 플레이어 기체의 z 좌표에서 플레이어의 뒤로 늘어놓았던 복도의 길이인 240.0f보다 더 멀어졌는지 확인하고 거리가 240.0f보다 더 멀다면 해당 벽의 위치를 복도의 총 길이인 490.0f만큼 앞으로 이동시켰습니다.

**소스코드: MainScene.cpp - CMainScene::Update() - 546 line**

1. 플레이어가 이동할 때 이동하는 방향 벡터의 z값이 음수일 경우 z값을 0.0f로 초기화하고 정규화한 이동 벡터를 사용하여 플레이어가 뒤로 이동할 수 없도록 하였습니다.

**소스코드: Player.cpp - CPlayer::Move() - 78 line**

1. **플레이어 기체를 포함한 모든 오브젝트들은 복도 밖으로 벗어나서는 안되며 복도 내부에 존재하는 오브젝트는 색깔 별로 앞뒤로만 이동, 좌우로만 이동, 위아래로만 이동, 자유롭게 이동의 네 종류가 있고 무조건 회전 및 이동을 해야한다.**

* 가정

1. 오브젝트의 종류는 색깔로 구별한다.
2. 빨간색 오브젝트는 x축으로만 이동할 수 있다.
3. 초록색 오브젝트는 y축으로만 이동할 수 있다.
4. 파란색 오브젝트는 z축으로만 이동할 수 있다.
5. 하얀색 오브젝트는 자유롭게 이동할 수 있다.
6. 빨간색, 초록색, 파란색 오브젝트는 벽 및 다른 오브젝트와 충돌 시 이동 방향이 정 반대로 바뀐다.
7. 하얀색 오브젝트는 빨간색, 초록색, 파란색 오브젝트와 충돌 시 충돌 반대 방향으로 튕긴다.
8. 하얀색 오브젝트는 벽과 충돌 시 반사된다.
9. 빨간색, 초록색, 파란색, 하얀색 오브젝트는 임의의 축으로 회전하고 임의의 속도로 이동한다.
10. 플레이어 기체를 제외한 모든 오브젝트들은 복도의 z축 범위를 벗어나면 위치와 오브젝트의 종류가 재설정된다.
11. 플레이어 기체는 벽을 뚫고 복도에서 나가지 못한다.

* 사용한 자료구조

빨간색, 초록색, 파란색, 하얀색 오브젝트들은 복도와 마찬가지로 한 번 생성된 후 프로그램이 종료될 때까지 제거되지 않으므로 STL의 벡터 컨테이너를 사용하였다.

* 구현

1. 큐브 오브젝트의 색깔을 오브젝트의 종류를 구분하는 tag로 사용하여 switch case로 객체 생성, 이동 방향 벡터 설정 및 충돌 체크를 종류별로 분할하였습니다.

**소스코드: MainScene.cpp - CMainScene::BuildObjects() - 741 line**

1. 빨간색, 초록색, 파란색, 하얀색 오브젝트의 충돌 체크 판단 방식은 software renderer 샘플 코드의 방식을 그대로 사용하였습니다.

**소스코드: MainScene.cpp - CMainScene:: Animate() – 168, 342 line**

1. 플레이어 캐릭터를 제외한 오브젝트끼리의 충돌 시 빨간색, 초록색, 파란색 오브젝트들은 이동 방향 벡터의 각 축의 값의 부호만 바뀌었고 하얀색 오브젝트는 자신의 중심위치에서 충돌한 오브젝트의 중심위치의 값을 뺀 벡터로 이동 방향 벡터가 바뀝니다.

**소스코드: MainScene.cpp - CMainScene:: Animate() – 363 line**

1. 하얀색 오브젝트가 벽과 충돌했을 경우 이동 방향 벡터가 벽의 반사 벡터로 바뀌며 복도의 z축 범위에서 오브젝트가 벗어났을 경우 해당 오브젝트는 임의의 종류와 위치로 재설정 됩니다.

**소스코드: MainScene.cpp - CMainScene:: Animate() – 168 line, ::Update() - 575 line**

1. 플레이어 기체와 벽을 제외한 색깔 별 오브젝트들의 회전 축과 속도는 rand()함수를 사용하여 임의로 설정했습니다.

**소스코드: MainScene.cpp - CMainScene::BuildObjects() - 737 line**

1. 플레이어 기체가 벽과 충돌하면 벽의 법선 벡터 방향으로 플레이어 기체의 이동속도만큼 밀려나서 벽을 뚫을 수 없습니다.

**소스코드: MainScene.cpp - CMainScene:: Animate() - 291 line**

1. **플레이어는 탄환을 플레이어가 바라보는 방향으로 발사할 수 있으며 탄환과 복도 내부의 오브젝트가 충돌하면 50개 이상의 오브젝트를 사용한 파티클 이펙트가 나타난다.**

* 가정

1. 특정 키보드 입력이 들어오면 탄환이 발사된다.
2. 탄환이 발사되는 방향은 카메라의 look 벡터 방향이다.
3. 탄환은 다른 오브젝트와 같은 방식으로 벽 및 오브젝트들에 충돌체크한다.
4. 탄환이 벽과 충돌할 경우 제거된다.
5. 탄환이 벽 이외의 오브젝트에 충돌할 경우 충돌한 오브젝트의 위치와 종류가 임의로 변경되고 충돌한 오브젝트의 위치에서 임의의 색깔과 이동 방향 벡터를 가진 작은 파티클 오브젝트가 50개 생성된다.
6. 파티클 오브젝트는 생성 시 설정된 방향과 속도로 짧은 시간 동안 이동한 후 제거된다.
7. 파티클 오브젝트는 벽을 제외한 다른 오브젝트들과 충돌하지 않는다.
8. 파티클 오브젝트는 벽과 충돌하면 반사된다.

* 사용한 자료구조

탄환과 파티클 오브젝트들은 객체의 삽입, 삭제가 자유로운 STL의 LIST 컨테이너를 사용하였다.

* 구현

1. 스페이스 바를 누르면 플레이어 기체의 중심 위치에서 카메라의 Look벡터 방향으로 임의의 축으로 회전하며 이동하는 보라색의 탄환을 생성하였습니다.

**소스코드: MainSceneInputProcessor.cpp -CMainSceneInputProcessor::ProcessInput() - 16 line**

1. 플레이어 기체를 생성할 때 CMainScene의 멤버 변수인 BulletList의 포인터를 받아서 플레이어의 Input함수가 스페이스 바 키 입력과 함께 호출되었을 때 BulletList에 탄환 오브젝트를 push할 수 있도록 하였습니다.

**소스코드: MainScene.cpp - CMainScene::BuildObjects() - 781 line**

1. 탄환 오브젝트들의 충돌 체크 판단 방식은 software renderer 샘플 코드의 방식을 그대로 사용하였습니다.

**소스코드: MainScene.cpp - CMainScene:: Animate() – 36, 419 line**

1. 탄환이 벽 및 오브젝트들과 충돌하거나 복도의 z축 범위에서 벗어났을 경우 BulletList에서 erase()하였습니다.

**소스코드: MainScene.cpp - CMainScene::Update() - 558 line, ::Animate() – 36, 419 line**

1. 탄환이 빨간색, 초록색, 파란색, 하얀색 오브젝트들과 충돌했을 경우 충돌한 오브젝트의 중심위치에서 파티클 오브젝트를 생성하였습니다.

**소스코드: MainScene.cpp - CMainScene:: Animate() –429 line,**

1. 파티클 오브젝트를 생성할 때 생성 위치와 개수를 인자로 받는 함수를 사용했으며 생성 개수는 디폴트로 50개를 설정하였고 이동 방향 및 회전 축은 -1000~1000의 랜덤 값을 XMFLOAT3에 저장한 후 정규화하여 임의의 값으로 설정하였습니다.

**소스코드: MainScene.cpp - CMainScene:: BuildParticles() – 803 line,**

1. 파티클 오브젝트는 생성할 때 샘플 코드에서 사용되고 있지 않은 m\_fMovingRange 변수를 이용하여 생성할 때 초 단위의 시간을 설정해주고 매 프레임마다 fTimeElapsed만큼 값을 빼 줘서 값이 0보다 작아지면 오브젝트를 삭제하였습니다.

**소스코드: MainScene.cpp - CMainScene::Update() - 624 line**

1. 파티클 오브젝트가 벽과 충돌했을 때 반사되는 방식은 software renderer 샘플 코드를 그대로 사용하였습니다.

**소스코드: MainScene.cpp - CMainScene:: Animate() – 92 line**

1. **복도 내부의 오브젝트들과 플레이어 기체가 충돌하면 게임오버가 된다.**

* 가정

1. 플레이어 기체가 오브젝트와 충돌 할 경우 게임 오버가 되고 플레이어 기체의 중심 위치에서 파티클 오브젝트들이 생성된다.
2. 게임 오버가 된 후 일정 시간이 지나면 플레이어 기체는 죽었던 z좌표의 (x, y)가 (0, 0)인 지점에서 되살아난다.

* 구현

1. 플레이어 기체가 빨간색, 초록색, 파란색, 하얀색 오브젝트들과 충돌하는 방식은 오브젝트들끼리 충돌하는 방식과 같습니다.

**소스코드: MainScene.cpp - CMainScene:: Animate() – 357 line**

1. 플레이어 기체가 오브젝트들과 충돌했을 경우 게임 오버 bool 변수가 true가 되며 충돌한 위치에서 파티클 오브젝트가 300개 생성됩니다.

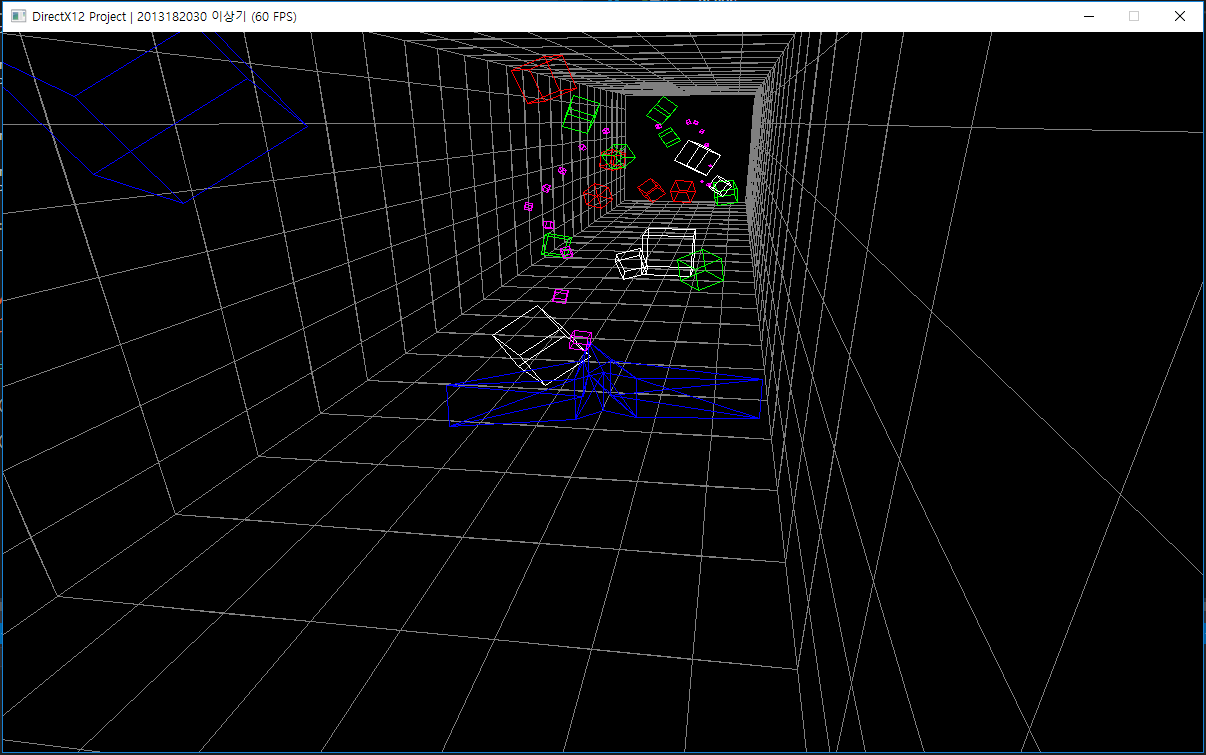
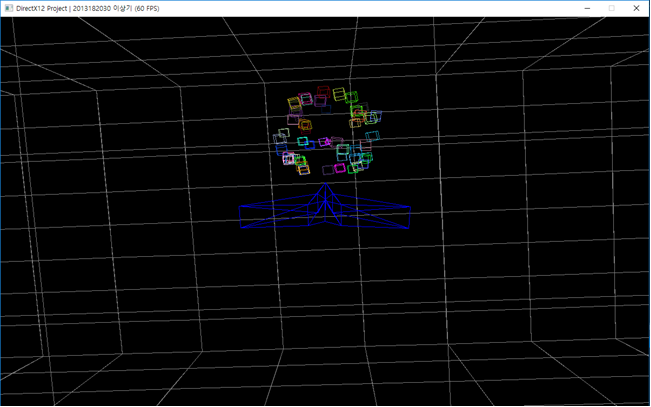
**소스코드: MainScene.cpp - CMainScene:: Animate() – 359 line**

1. 게임 오버 bool 변수가 true일 경우 Process Input 함수를 호출하지 않으며 플레이어 기체를 그리지 않고 보이지 않는 기체와 오브젝트가 충돌했을 경우에 아무 일도 일어나지 않고 게임 오버 카운터 변수에 fTimeElapsed값이 누적됩니다.

**소스코드: MainScene.cpp - CMainScene:: Update() – 483 line**

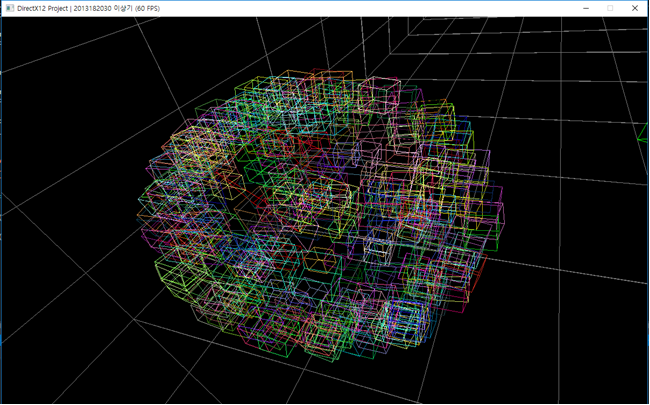
1. 게임 오버 카운터 변수 값이 2(초)보다 클 경우 플레이어 기체의 x, y위치를 0으로 설정하고 z위치의 +40.0f, -40.0f범위 안에 있는 큐브 오브젝트들의 위치를 플레이어 기체의 앞 100.0f 이상으로 재설정합니다.

**소스코드: MainScene.cpp - CMainScene:: Update() – 486 line**

* **[실행결과]**

**오브젝트 파괴**

**플레이 화면**



**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. |
| **R** | 카메라가 바라보는 방향의  **위쪽으로 상승**한다. |
| **F** | 카메라가 바라보는 방향의  **아래쪽으로 하강**한다. |
| **마우스**  **왼쪽 버튼 + 좌우 상하 이동** | **누른 상태**로 마우스를 움직이면  **카메라가 바라보는 방향이 마우스를 따라 움직인다.** |
| **마우스**  **오른쪽 버튼 + 좌우 이동** | **누른 상태**로 마우스를 움직이면  **카메라가 바라보는 방향이 z축으로 회전한다.** |
| **마우스**  **오른쪽 버튼 + 상하 이동** | **누른 상태**로 마우스를 움직이면  **카메라가 바라보는 방향이 마우스를 따라 움직인다.** |
| **SPACE** | 플레이어가 **탄환을 발사**한다. |
| **ESC** | **프로그램 종료** |