1. 길이 1000m의 직육면체 공간

전체 직육면체의 길이와 높이인 fWidthd와 fHeight를 인자로 받아와서 절반으로 나눈 값을 기준으로 6면을 만들었습니다. 몇 개의 작은 직육면체로 나눌건지도 정해줘서 평면 여러 개를 만들었습니다.

직육면체 공간을 못 벗어나게 하는 건 void CGameFramework::ProcessInput()에서 했습니다. 단순한 if문으로 플레이어의 위치가 직육면체를 넘었는지 체크 후 넘었으면 위치를 살짝 바꾸는 식으로 만들었습니다. 보스에 너무 가까이 가지 못하도록 z는 400까지로 제한했습니다.

2. 보스

보스는 커다란 빨간색 오브젝트이고 당시 플레이어 위치로 총알을 쏩니다. 플레이어가 3번 맞추면 게임에서 승리하고 재시작됩니다. 보스는 플레이어 클래스의 자식클래스로 플레이어클래스는 총알을 쏘는 Fire 함수를 갖고 있는데 Fire가 호출 될 때마다 std::vector<CBullet\*>형인 m\_bullets 에 Bullet객체를 넣어줬습니다. 총알 클래스인 Bullet은 CGameObject을 상속받은 클래스입니다. 기본적인 바운딩박스와 위치, 크기, 회전 값을 맴버변수로 갖습니다. 추가로 totalMovement라는 변수를 갖는데 전체 이동거리를 저장해 놓는 변수입니다. 이 변수로 총알이 얼마나 움직였는지 확인 후 isDead(1000)이렇게 맴버함수를 호출하면 totalMovement가 1000이 넘었을때만 true를 반환하게 해서 없어져야 되는지 체크했습니다. GameFramework의 FrameAdvance함수에서 시간을 체크 후 1초마다 Fire 함수를 호출하도록 했습니다. 보스의 총알을 플레이어와 충돌체크해 부딪혔다면 플레이어의 체력을 1 깎았습니다.

3. 적

기본적으로 적은 CScene 함수의 Animate함수 호출시마다 처리했습니다. Color값을 배열형태로 저장 후 rand함수를 통해 무작위로 인덱스를 뽑았습니다. 플레이어와 z값 기준 100 앞에 스폰되게 만들었습니다. 생성 당시 x,y 좌표는 랜덤함수로 -10~10 범위내에서 뽑아내었고 회전값과 방향도 0~1사이 실수값으로 랜덤하게 뽑았습니다. 생성한 오브젝트는 CScene 클래스에서 관리하는 m\_objects라는 벡터에 넣어두었고 게임이 재시작되거나 총알에 죽을 때 벡터에서 지워줬습니다.

플레이어를 따라가게 하는 부분은 m\_objects[i] ->setDirection()함수에 (플레이어 – 현제위치)를 계산한 벡터값을 넘겨줘서 매번 플레이어 위치로 이동하게 했습니다. 바로 따라가면 안되고 일정시간 후에 따라가도록 하는 부분은 오브젝트마다 m\_spawnTime 변수를 두어 시간을 제다가 2초가 넘었을 때만 플레이어 방향으로 날아가게 했습니다.

Red색깔을 가진 적은 따로 시간을 제서 10초마다 만들어줬고 역시 위치, 회전, 방향 랜덤으로 줬습니다. 기본 오브젝트와 같은 m\_objects 벡터에 넣어줬습니다.

CheckObjectByWallCollisions()함수에서 오브젝트와 벽간의 충돌체크를 했습니다. 벽의 6개의 평면과의 충돌체크를 해서 Disjoint이거나 Intersects 일 때 현재 방향의 반사벡터를 구해서 부딪힌 오브젝트를 그쪽으로 움직이게 했습니다.

CheckObjectByObjectCollisions()함수에서 플레이어나 총알과의 충돌체크를 처리했습니다. Scene은 플레이어 객체에 대한 포인터를 갖고 있으므로 플레이어 객체에 m\_bullets벡터에 접근해 for루프를 돌면서 총알의 바운딩박스와 오브젝트의 바운딩박스가 교차했는지를 체크했습니다. 교차했다면 오브젝트를 CExplosiveObject로 캐스팅하고 m\_bBlowingUp 변수를 참으로 바꿔줘서 Animate시 터지도록 만들었습니다. Bullet은 alive변수를 false로 해 플레이어 업데이트 시 사라지도록 만들었습니다. 추가로 빨간 적을 무찔렀을 때 아이템을 가져야 하므로 교차한 오브젝트의 색이 빨간색인지 판단해 m\_pPlayer의 m\_item의 수치를 1 증가시켰습니다.

4. 플레이어

게임프레임워크에서 처리하는 부분

ProcessInput함수에서 키 입력을 체크 합니다. 입력된 키에 따라 상수로 방향을 입력하는데 or연산을 통해 하나 이상의 키입력도 처리할 수 있습니다. 상수는 Player의 Move함수로 전해져 해당방향으로 움직이게 합니다. 마우스 좌표의 예전 값과 이번 값의 차이를 구해 5로 나눈 값으로 플레이어의 회전정도를 구합니다.

Ctrl을 누르면 플레이어의 함수의 Fire함수를 불러 총알을 발사합니다. 보스가 총알 발사하는 원리와 똑같이 m\_bulets 벡터에 총알을 생성해서 넣어줍니다. 총알은 플레이어가 바라보는 방향으로 날아갑니다. 별도의 총알갯수제한은 두지 않았고 일정거리가 지나면 없어집니다. 플레이어가 아이템이 있다면 Z키로 누르면 아이템을 사용합니다. 게임 난이도 때문에 모든 적을 폭파시켰습니다.

직육면체의 공간에 부딪히면 색이 바뀌는 대신 원점으로 돌아옵니다. 충돌체크로 판단하건 if문으로 판단하건 오류가 있어 이렇게 설정했습니다.

CheckObjectByObjectCollisions()함수에서 플레이어의 바운딩 박스와 오브젝트가 교차했다면 플레이어의 체력을 1 줄이고 플레이어의 m\_invincibilityTime을 1로 설정했습니다. 일종의 무적시간으로 충돌 후 1초간은 충돌하더라도 체력을 깍지 않았습니다. 충돌 시마다 체력을 줄이면 30프레임이 나온다 했을 때 1초에 30의 체력이 줄기 때문입니다. 3번 부딪히면 게임이 재시작됩니다.

플레이어의 현재 z값과 오브젝트들의 z값을 비교해 플레이어가 크면 오브젝트가 뒤에 있는 경우이므로 오브젝트 색을 핑크색으로 바꿨습니다.

5. 픽킹

픽킹은 구현하려 했지만 제대로 구현되지는 않았습니다. PPT에 있는 화면좌표계->투영좌표계->카메라좌표계로 역변환한 광선을 구한 후 바운딩박스와의 충돌체크를 했지만 제대로 작동하지 않았습니다. 광선에 뷰행렬의 역행렬을 구해서 광선의 월드좌표를 구할려 했지만 제대로 되지 않았습니다.

6. 추가로 구현한 것.

플레이어가 부딪힌 오브젝트의 색깔로 플레이어가 변하게 했습니다. 보스의 총알을 플레이어를 따라 다니는 유도탄으로 했고 플레이어는 시작 시 아이템 하나를 가지고 시작합니다.

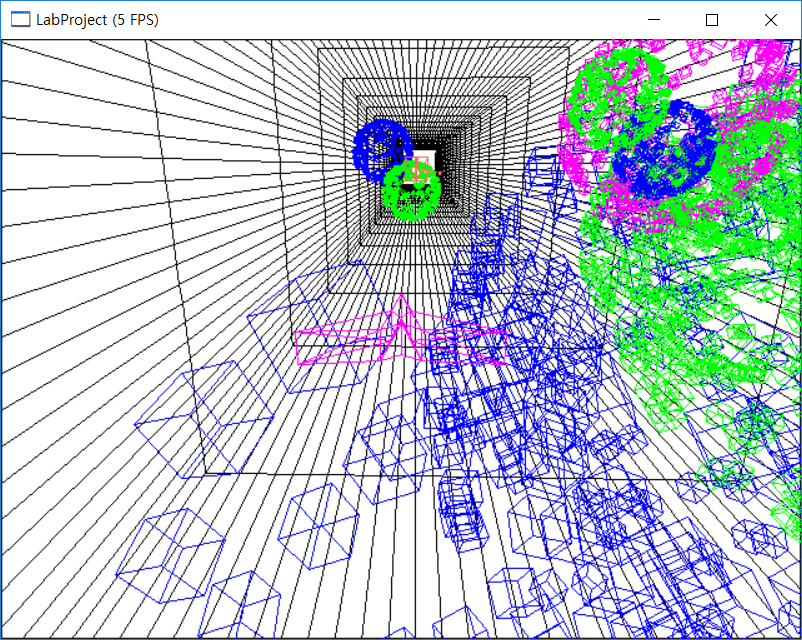
7. 조작법

화살표 - 전후좌우 이동

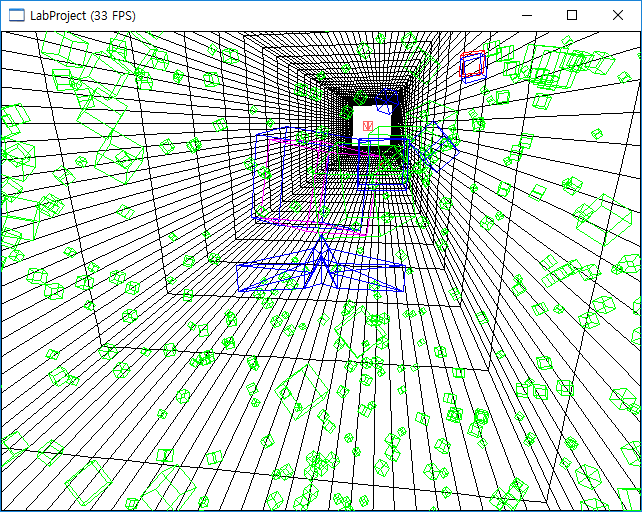
pageUp, pageDown – 상하 이동

왼쪽 ctrl – 총알 발사

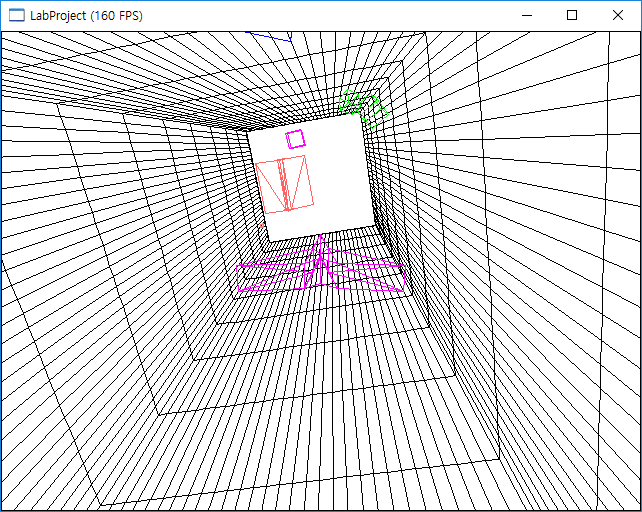
Z – 아이템 사용



=============아이템을 쓴 장면===============



=============적이 총알에 죽은 장면===============



==================보스======================