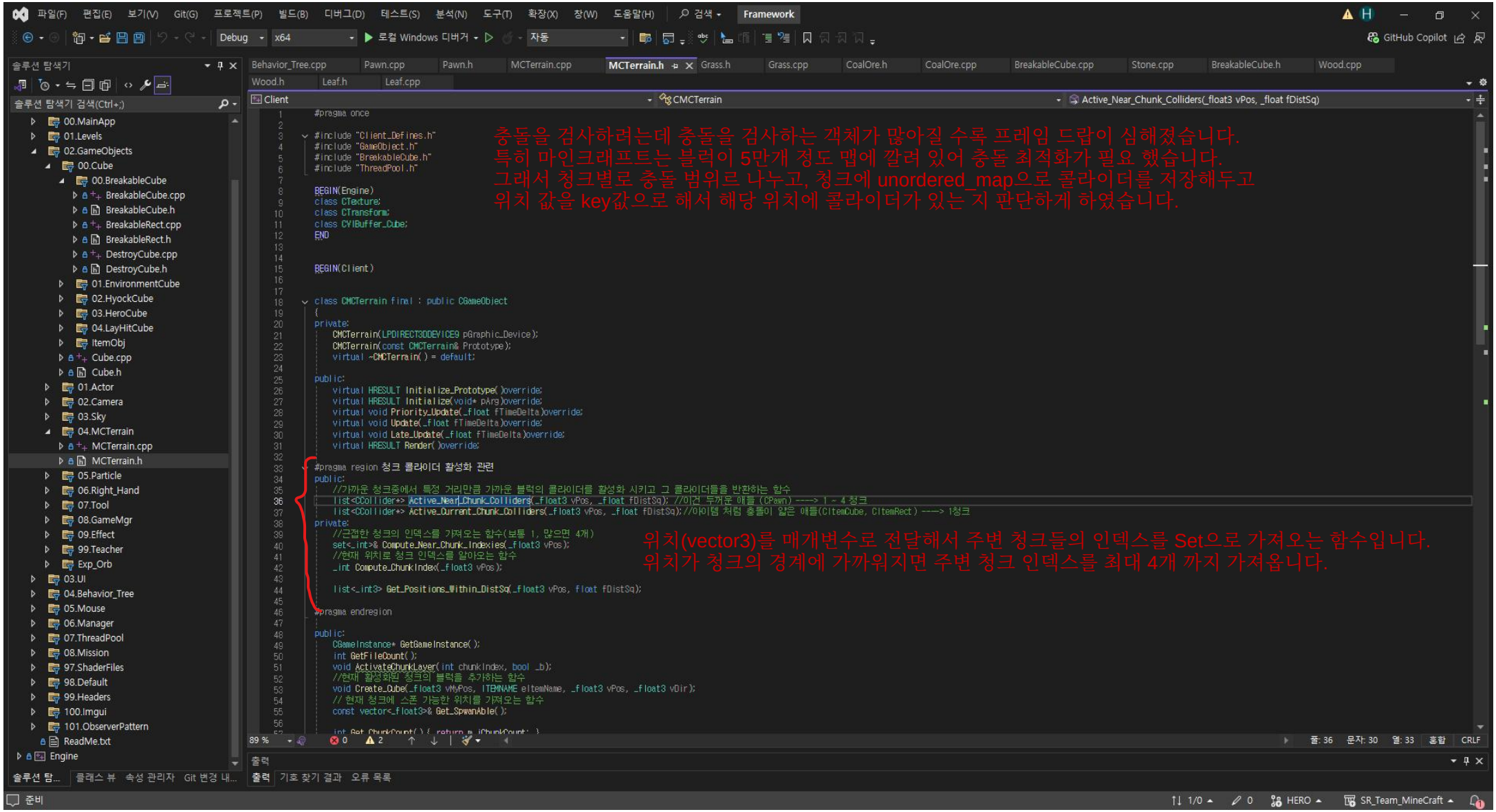
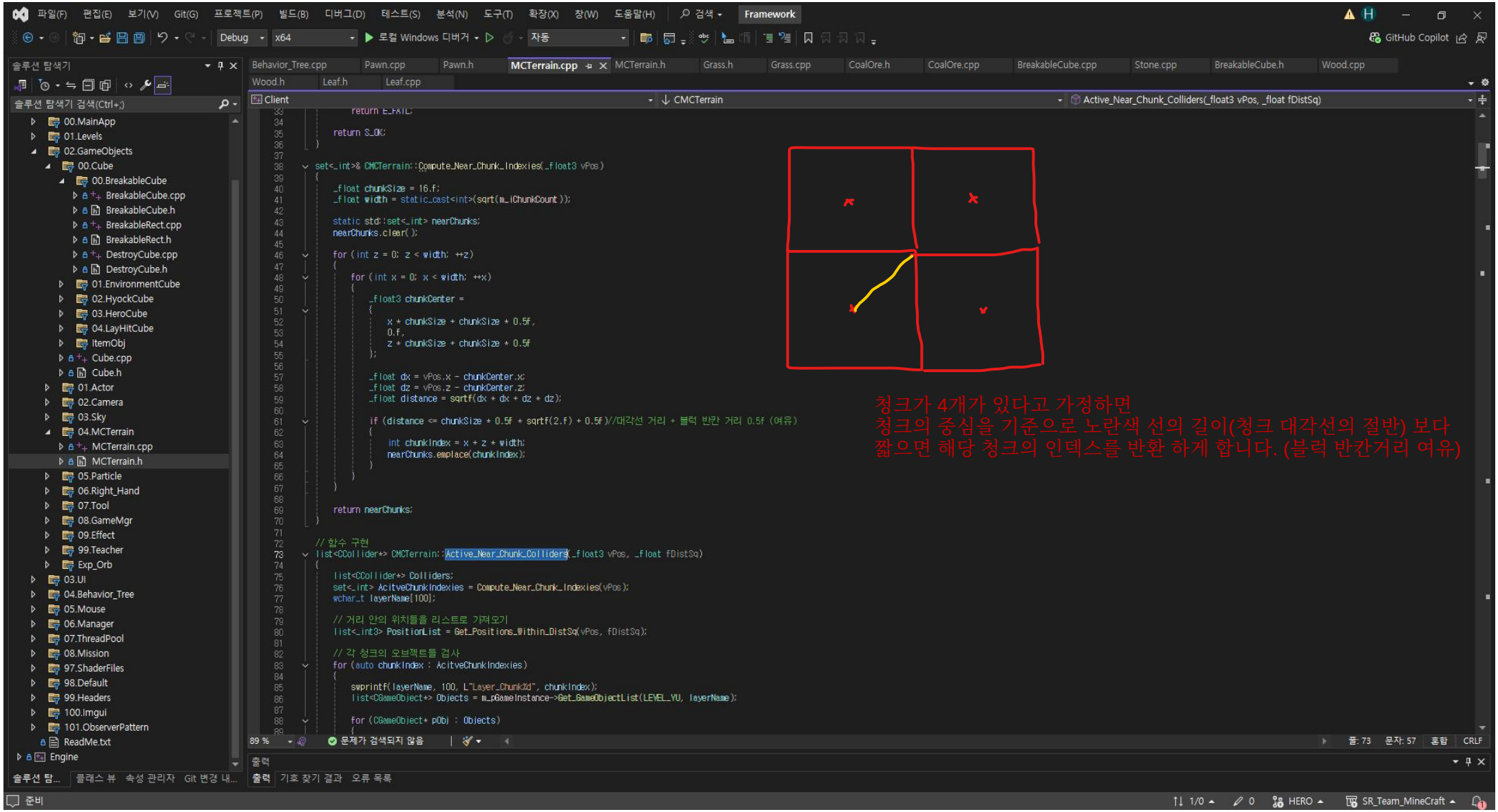
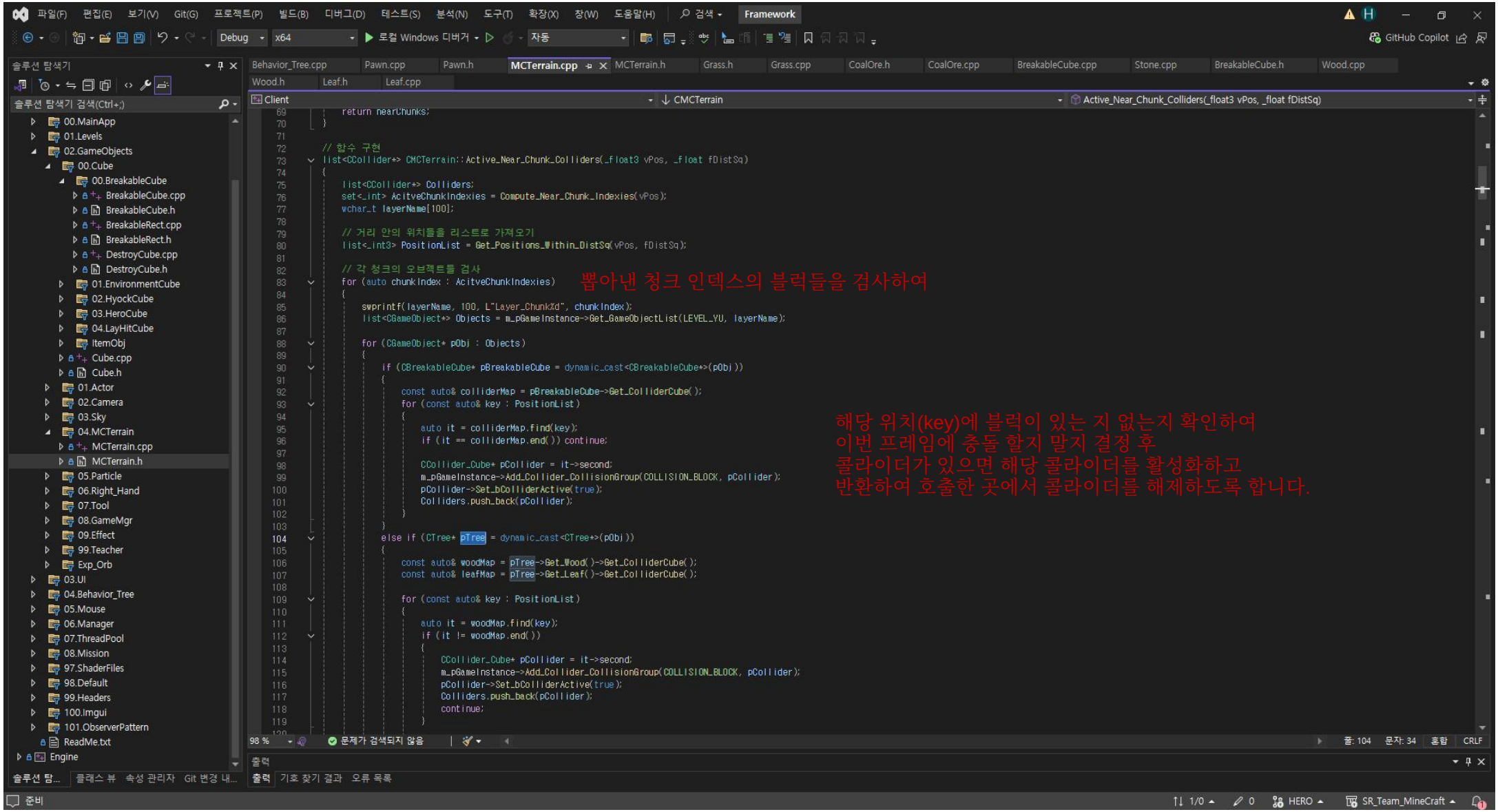
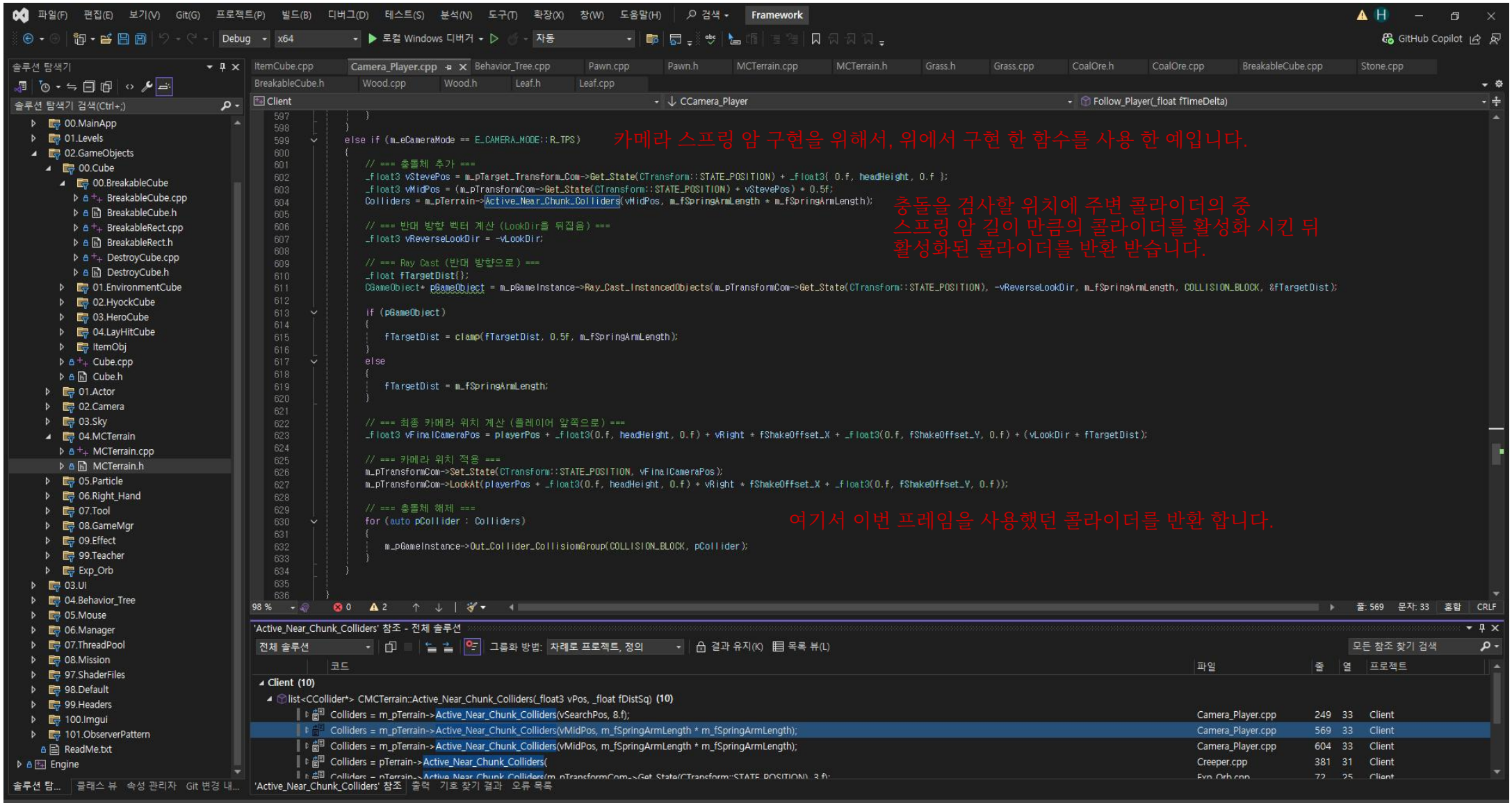

충돌을 검사하려는데 충돌을 검사하는 객체가 많아질 수록 프레임 드랍이 심해졌습니다.
특히 마인크래프트는 블럭이 5만개 정도 맵에 깔려 있어 충돌 최적화가 필요 했습니다.
그래서 청크별로 충돌 범위르 나누고, 청크에 unordered_map으로 콜라이더를 저장해두고
위치 값을 key값으로 해서 해당 위치에 콜라이더가 있는 지 판단하게 하였습니다.
위치(vector3)를 매개변수로 전달해서 주변 청크들의 인덱스를 Set으로 가져오는 함수입니다.
위치가 청크의 경계에 가까워지면 주변 청크 인덱스를 최대 4개 까지 가져옵니다.


파일(F) 편집(E) 보기(V) Git(G) 프로젝트(P) 빌드(B) 디버그(D) 테스트(S) 분석(N) 도구(T) 확장(X) 창(W) 도움말(H)
Framework
Debug x64 로컬 Windows 디버거 자동
GitHub Copilot
솔루션 탐색기
솔루션 탐색기 검색(Ctrl+;)
00.MainApp
01.Levels
02.GameObjects
00.Cube
00.BreakableCube
BreakableCube.cpp
BreakableCube.h
BreakableRect.cpp
BreakableRect.h
DestroyCube.cpp
DestroyCube.h
01.EnvironmentCube
02.HyockCube
03.HeroCube
04.LayHitCube
ItemObj
Cube.cpp
Cube.h
01.Actor
02.Camera
03.Sky
04.MCTerrain
MCTerrain.cpp
MCTerrain.h
05.Particle
06.Right_Hand
07.Tool
08.GameMgr
09.Effect
99.Teacher
Exp_Orb
03.UI
04.Behavior_Tree
05.Mouse
06.Manager
07.ThreadPool
08.Mission
97.ShaderFiles
98.Default
99.Headers
100.Imgui
101.ObserverPattern
ReadMe.txt
Engine
Behavior_Tree.cpp
Pawn.cpp
Pawn.h
MCTerrain.cpp
MCTerrain.h
Grass.h
Grass.cpp
CoalOre.h
CoalOre.cpp
BreakableCube.cpp
Stone.cpp
BreakableCube.h
Wood.cpp
Wood.h
Leaf.h
Leaf.cpp
Client
CMCTerrain
Active_Near_Chunk_Colliders(_float3 vPos, _float fDistSq)
return E_FAIL;
return S_OK;
}
set<_int>& CMCTerrain::Compute_Near_Chunk_Indexies(_float3 vPos)
{
_float chunkSize = 16.f;
_float width = static_cast<int>(sqrt(m_iChunkCount));
static std::set<_int> nearChunks;
nearChunks.clear();
for (int z = 0; z < width; ++z)
{
for (int x = 0; x < width; ++x)
{
_float3 chunkCenter =
{
x * chunkSize + chunkSize * 0.5f,
0.f,
z * chunkSize + chunkSize * 0.5f
};
_float dx = vPos.x - chunkCenter.x;
_float dz = vPos.z - chunkCenter.z;
_float distance = sqrtf(dx * dx + dz * dz);
if (distance <= chunkSize * 0.5f * sqrtf(2.f) + 0.5f)//대각선 거리 + 블럭 반칸 거리 0.5f (여유)
{
int chunkIndex = x + z * width;
nearChunks.emplace(chunkIndex);
}
}
}
return nearChunks;
}
// 함수 구현
list<CCollider*> CMCTerrain::Active_Near_Chunk_Colliders(_float3 vPos, _float fDistSq)
{
list<CCollider*> Colliders;
set<_int> AcitveChunkIndexies = Compute_Near_Chunk_Indexies(vPos);
wchar_t layerName[100];
// 거리 안의 위치들을 리스트로 가져오기
list<_int3> PositionList = Get_Positions_Within_DistSq(vPos, fDistSq);
// 각 청크의 오브젝트를 검사
for (auto chunkIndex : AcitveChunkIndexies)
{
swprintf(layerName, 100, L"Layer_Chunk%d", chunkIndex);
list<CGameObject*> Objects = m_pGameInstance->Get_GameObjectList(LEVEL_YU, layerName);
for (CGameObject* pObj : Objects)
청크가 4개가 있다고 가정하면
청크의 중심을 기준으로 노란색 선의 길이(청크 대각선의 절반) 보다
짧으면 해당 청크의 인덱스를 반환 하게 합니다. (블럭 반칸거리 여유)
89 %
문제가 검색되지 않음
줄: 73 문자: 57 혼합 CRLF
출력
출력 기호 찾기 결과 오류 목록
솔루션 탐... 클래스 뷰 속성 관리자 Git 변경 내...
준비
1/0
0
HERO
SR_Team_MineCraft


뽑아낸 청크 인덱스의 블럭들을 검사하여
해당 위치(key)에 블럭이 있는 지 없는지 확인하여
이번 프레임에 충돌 할지 말지 결정 후
콜라이더가 있으면 해당 콜라이더를 활성화하고
반환하여 호출한 곳에서 콜라이더를 해제하도록 합니다.


카메라 스프링 암 구현을 위해서, 위에서 구현 한 함수를 사용 한 예입니다.
충돌을 검사할 위치에 주변 콜라이더의 중
스프링 암 길이 만큼의 콜라이더를 활성화 시킨 뒤
활성화된 콜라이더를 반환 받습니다.
여기서 이번 프레임을 사용했던 콜라이더를 반환 합니다.