

**3D게임프로그래밍2(01)**

**과제1 설명**

**제출일: 2023.11.05**

**전공: 게임공학과**

**학번: 2019184020**

**성명: 윤은지**

**목차**

1. 조작법(키)에 대한 설명
2. 플레이어
3. 적 헬리콥터
4. 모델+터레인 프로젝트에 빌보드 프로젝트 통합
5. 모델+터레인+빌보드 프로젝트에 물 프로젝트 통합
6. 충돌처리
7. 적 헬리콥터의 이동
8. 플레이어 가속
9. 폭발 애니메이션과 점수판
10. 실행 화면
11. **조작법(키)에 대한 설명**

1)플레이어

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

w : 앞으로 이동

s : 뒤로 이동

a : 왼쪽으로 이동

d : 오른쪽으로 이동

x : 위로 이동

c : 아래로 이동

SHIFT : 가속

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

마우스 우클릭 : 공격

2)적 헬리콥터

텍스트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

UP : 앞으로 이동

DOWN : 뒤로 이동

LEFT : 왼쪽으로 이동

RIGHT : 오른쪽으로 이동

ENTER : 위로 이동

SHIFT : 아래로 이동

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

ESC : 종료

CTRL : 전체 화면으로 전환

또한 Release/x64 모드를 사용하여 프로젝트를 빌드하였습니다.

1. **모델+터레인 프로젝트에 빌보드 프로젝트 통합**

Shader.h에 다음 클래스들을 추가하고, Shader.cpp에 다음 클래스들의 구현부를 추가하였습니다.

class CBillboardObjectsShader : public CObjectsShader

class CMultiSpriteObjectsShader : public CObjectsShader

class CTexturedShader : public CShader

Scene.cpp의 다음 함수에서



텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

폭발 애니메이션을 재생하기 위한 객체와 풀, 나무, 꽃을 출력하기 위한 객체를 빌드했습니다.

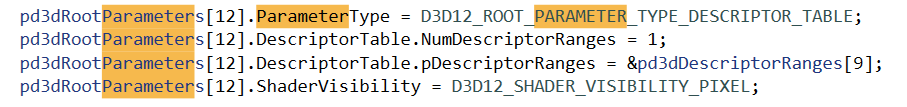
다음 함수에



이 부분을 추가하였습니다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명



Shader.cpp의 다음 함수에서  
void CBillboardObjectsShader::BuildObjects(ID3D12Device\* pd3dDevice, ID3D12GraphicsCommandList\* pd3dCommandList, void\* pContext)

이 부분을 수정하였습니다.

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

그리고 다음 함수들을 추가하였습니다.

D3D12\_SHADER\_BYTECODE CBillboardObjectsShader::CreateVertexShader()

D3D12\_SHADER\_BYTECODE CBillboardObjectsShader::CreatePixelShader()

D3D12\_INPUT\_LAYOUT\_DESC CBillboardObjectsShader::CreateInputLayout()

Object.h에 다음 클래스들을 추가하고, Object.cpp에 다음 클래스들의 구현부를 추가하였습니다.

class CGrassObject : public CGameObject

class CMultiSpriteObject : public CGameObject

Shaders.hlsl에 다음을 추가하였습니다.











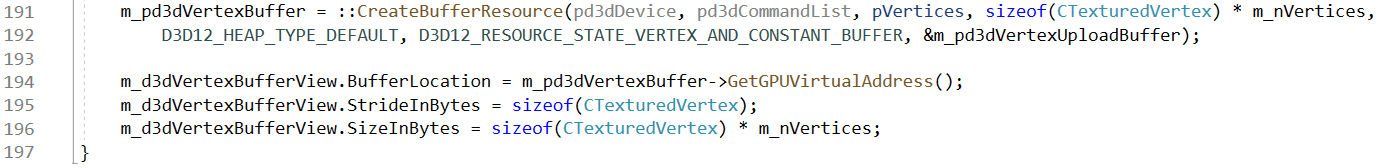
Mesh.h에 다음 클래스를 추가하고, Mesh.cpp에 다음 클래스의 구현부를 추가하였습니다.

class CVertex

Mesh.cpp의 다음 함수에서



이 부분을 수정하였습니다.



1. **모델+터레인+빌보드 프로젝트에 물 프로젝트 통합**

Mesh.h에 다음 클래스들을 추가하고, Mesh.cpp에 다음 클래스들의 구현부를 추가하였습니다.

class CGridMesh : public CMesh

class CDiffusedVertex : public CVertex

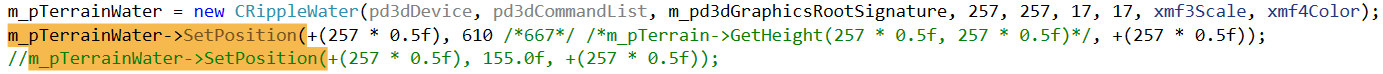
class CDiffusedTexturedVertex : public CDiffusedVertex

Object.h에 다음 클래스를 추가하고, Object.cpp에 다음 클래스의 구현부를 추가하였습니다.

class CRippleWater : public CGameObject

Scene.cpp의 다음 함수에서

void CScene::BuildObjects(ID3D12Device\* pd3dDevice, ID3D12GraphicsCommandList\* pd3dCommandList)



물 객체를 빌드하고, 위치를 설정했습니다.

Shader.h에 다음 클래스를 추가하고, Shader.cpp에 다음 클래스의 구현부를 추가하였습니다.

class CRippleWaterShader : public CTexturedShader

Shaders.hlsl에 다음을 추가하였습니다.

struct VS\_RIPPLE\_WATER\_INPUT

struct VS\_RIPPLE\_WATER\_OUTPUT

VS\_RIPPLE\_WATER\_OUTPUT VSRippleWater(VS\_RIPPLE\_WATER\_INPUT input)

float4 PSRippleWater(VS\_RIPPLE\_WATER\_OUTPUT input) : SV\_TARGET

다음 구조체에서

struct MATERIAL

float4 m\_cSpecular; //a = power

위 라인을 주석 처리하고

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

다음 변수들을 추가하였습니다.





텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

위 코드에서 gfCurrentTime을 gMaterial.gfCurrentTime로 바꿔주었습니다.

그리고 물의 위치를 변경하기 위해



다음과 같이 155.0f를 610.0f로 수정했습니다.

물을 출렁이게 하기 위해



위 라인을 주석 해제하고



위 라인을 주석 처리 했습니다.

물을 더 빠르게 흐르게 하기 위해



위 라인에서 0.00125f를 0.0825f로 수정하였습니다.

Shader.cpp의 다음 함수에



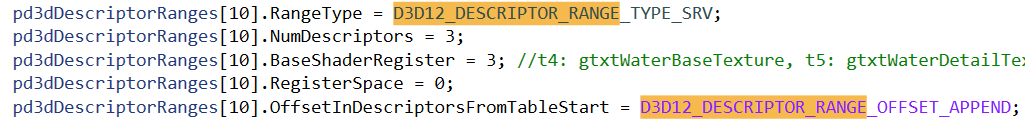
다음 코드를 추가하였습니다.



Scene.cpp의 다음 함수에



다음 코드를 추가하였습니다.



텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명



다음 함수에

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

다음 코드를 추가하였습니다.

위의 변수들은 GameFramework.cpp의 다음 함수에서

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

이렇게 값을 할당했습니다.

Object.cpp의 다음 함수를

void CMaterial::UpdateShaderVariables(ID3D12GraphicsCommandList\* pd3dCommandList)

다음과 같이 수정하였습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명



다음 함수는

void CGameObject::Render(ID3D12GraphicsCommandList\* pd3dCommandList, CCamera\* pCamera)

이 부분을 수정하였습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

다음 함수는

CRippleWater::CRippleWater(ID3D12Device\* pd3dDevice, ID3D12GraphicsCommandList\* pd3dCommandList, ID3D12RootSignature\* pd3dGraphicsRootSignature, int nWidth, int nLength, int nBlockWidth, int nBlockLength, XMFLOAT3 xmf3Scale, XMFLOAT4 xmf4Color) //: CGameObject(0)

이 부분을 수정하였습니다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

Mesh.cpp의 다음 함수의

CGridMesh::CGridMesh(ID3D12Device\* pd3dDevice, ID3D12GraphicsCommandList\* pd3dCommandList, int xStart, int zStart, int nWidth, int nLength, XMFLOAT3 xmf3Scale, XMFLOAT4 xmf4Color, void\* pContext) : CMesh(pd3dDevice, pd3dCommandList)

이 부분을 수정하였습니다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

그리고 이 부분을 수정하였습니다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**4. 충돌처리**

텍스트, 폰트, 번호, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. if(m\_pPlayer->attack) :

이 조건은 플레이어가 공격을 수행 중인지 여부를 확인합니다.

1. XMFLOAT3 Cur\_LookVector = m\_pPlayer->GetLookVector(); :

플레이어의 시선(바라보는 방향)을 나타내는 벡터를 얻습니다. 이 방향은 플레이어가 총을 쏘는 방향을 나타냅니다.

1. XMFLOAT3 Cur\_Pos = m\_pPlayer->GetPosition(); :

플레이어의 현재 위치를 얻습니다.

1. XMVECTOR Bullet\_Origin = XMLoadFloat3(&Cur\_Pos); 및 XMVECTOR Bullet\_Direction = XMLoadFloat3(&Cur\_LookVector); :

Cur\_Pos와 Cur\_LookVector를 사용하여 총알의 시작 위치와 방향을 나타내는 XMVECTOR를 생성합니다.

1. for(int i{}; i<pObjectShader->m\_nObjects; ++i) :

게임 오브젝트를 순회하는 반복문을 시작합니다. pObjectShader 객체의 m\_nObjects 속성은 게임 오브젝트의 총 수를 나타냅니다.

1. float bullet\_monster\_distance =

Vector3::Length(Vector3::Subtract(pObjectsShader->obj[i]->aabb.Center, Cur\_Pos)); :

현재 적 헬리콥터와 플레이어 사이의 거리를 계산합니다. 이 거리는 적 헬리콥터와 총알 간의 충돌을 감지하는 데 사용됩니다.

1. if ( pObjectShader->obj[i]->aabb.Intersects(Bullet\_Origin, Bullet\_Direction, bullet\_monster\_distance)) :

적 헬리콥터의 경계 상자와 총알 간의 충돌을 검사합니다.

Intersects 메서드를 사용하여 충돌 여부를 확인하고, 충돌이 발생하면 아래 코드 블록이 실행됩니다.

1. pObjectsShader->obj.erase(pObjectsShader->obj.begin() + i); :

명중한 적 헬리콥터를 벡터에서 삭제합니다.

**5. 적 헬리콥터 이동**

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. obj.push\_back(m\_ppObjects[nObjects]); :

obj는 CObjectsShader 클래스의 멤버 변수로, 적 헬리콥터를 저장하는 벡터입니다. 마지막 줄의 이 코드는 생성한 적 헬리콥터를 obj에 추가합니다.

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

1. if (!obj.empty()) :

이 조건은 obj 벡터가 비어 있지 않은 경우, 즉 적 헬리콥터에서 렌더링 함수를 호출했을 경우에 작업을 수행합니다.

1. for (const auto& o : obj) :

obj 벡터에 저장된 각 헬리콥터에 대해 반복합니다. auto 키워드를 사용하여 범용 자료형을 사용하며, 범위 기반 for 루프를 사용하여 컨테이너를 순회합니다. o는 현재 순회 중인 헬리콥터를 나타냅니다.

1. o->Render2(pd3dCommandList, pCamera); :

Render2 메서드를 호출하여 적 헬리콥터를 화면에 렌더링합니다. 이 메서드는 pd3dCommandList와 카메라를 사용하여 적 헬리콥터를 그래픽스 파이프라인을 통해 렌더링합니다. 따라서 적 헬리콥터는 화면에 그려집니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

1. if(1==dist(gen)) ~ else if(6==dist(gen)) :

각 헬리콥터에 대해, 1부터 6까지의 난수를 생성하고, 그 값에 따라 다양한 이동을 적용합니다. 각 조건문은 난수가 특정 값과 일치할 경우에 해당 이동을 수행합니다. 예를 들어, if(1==dist(gen))은 난수가 1인 경우에 해당 헬리콥터를 위로 이동시킵니다. 나머지 경우에는 다른 방향으로 이동합니다.

o->MoveForward(+1.0f) : 오브젝트를 앞으로 이동시킵니다.

o->MoveForward(-1.0f) : 오브젝트를 뒤로 이동시킵니다.

o->MoveStrafe(-1.0f) : 오브젝트를 좌로 이동시킵니다.

o->MoveStrafe(+1.0f) : 오브젝트를 우로 이동시킵니다.

o->MoveUp(+1.0f) : 오브젝트를 상승시킵니다.

o->MoveUp(-1.0f) : 오브젝트를 하강시킵니다.

1. **플레이어 가속**

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. pKeysBuffer[0x10] & 0xF0 :

pKeysBuffer 배열에서 0x10 위치의 값과 0xF0을 비트 단위로 비교합니다. 이렇게 비트 AND 연산자를 사용하면 0x10 위치의 비트 중에서 0xF0과 비트 AND를 수행합니다.

1. pKeysBuffer[0x10] :

pKeysBuffer 배열에서 0x10 위치에 있는 값을 나타냅니다. 이 값은 키 입력 상태로 사용됩니다.

1. 0xF0 :

0xF0은 16진수 상수로, 이진으로 나타내면 11110000입니다. 이 값은 비트 마스크로 사용됩니다.

1. pKeyBuffer[0x10] & 0xF0 :

pKeyBuffer 배열에서 0x10 위치의 값과 0xF0을 비트 AND 연산하여 특정 비트 패턴을 확인합니다. 만약 pKeysBuffer[0x10]의 비트 중 상위 4비트가 모두 1로 설정되어 있을 때, 이 연산 결과는 0xF0이 됩니다.

1. dwDirection :

dwDirection은 비트 연산을 통해 조작할 상태 정보를 저장하는 변수로 사용됩니다.

1. dwDirection |= DIR\_RUN :

이 부분은 dwDirection에 DIR\_RUN 상수를 추가합니다. 비트 OR 연산자를 사용하여 DIR\_RUN의 비트 패턴을 dwDirection에 추가합니다. 이것은 특정 상태를 나타내며, DIR\_RUN이 상수일 때 이 비트를 설정하고 나머지 비트는 변경하지 않습니다.

1. dwDirection |= DIR\_RUN 실행 조건 :

조건 pKeysBuffer[0x10] & 0xF0의 결과가 0이 아니고, dwDirection이 0이 아닐 때 dwDirection에 DIR\_RUN을 추가합니다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. SetMaxVelocityXZ(25.5f);와 SetMaxVelocityY(20.0f);를 주석 처리 했습니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

1. direction 변수가 DIR\_DOWN 상수와 일치하는지 확인하여 오브젝트가 수직 방향으로 이동 중인지 확인합니다.
2. 만약 수직으로 이동 중이고 RUN 상태라면, 최대 Y 방향 속도를 150.0으로 설정합니다. 그렇지 않다면 50.0으로 설정합니다.
3. 만약 수직 이동 중이 아니라면, 좌우로 이동 중인지 확인합니다.
4. 만약 좌우로 이동 중이고 RUN 상태라면, 최대 수평 방향(XZ)속도를 150.0으로 설정합니다. 그렇지 않다면 50.5로 설정합니다.
5. **폭발 애니메이션과 점수판**

GameFramework.cpp의 다음 함수에서

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 마우스 우클릭 시

m\_pScene->pMultiSpriteObjectShader->m\_ppObjects[0]->m\_ppMaterials[0]->m\_pTexture->m\_bActive = true; 이 변수에 true를 할당했습니다.

1. 3부터 5까지의 난수를 생성해 3일 경우 B+ 점수판, 4일 경우 C+ 점수판, 5일 경우 A+ 점수판의 텍스쳐를 할당하게끔 했습니다.

Shader.h의 다음 클래스에

class CMultiSpriteObjectsShader : public CObjectsShader

다음 함수들을 추가하였습니다.

virtual D3D12\_SHADER\_BYTECODE CreateVertexShader();

virtual D3D12\_SHADER\_BYTECODE CreatePixelShader();

virtual D3D12\_INPUT\_LAYOUT\_DESC CreateInputLayout();

Shader.cpp의 다음 함수를



이렇게 수정하였습니다.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

위의 코드는 빌보드 스프라이트 애니메이션을 재생하는 부분입니다.

다음 함수는

void CMultiSpriteObjectsShader::Render(ID3D12GraphicsCommandList\* pd3dCommandList, CCamera\* pCamera)

이 부분을 수정하였습니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

j가 0, 1일 때는 폭발 애니메이션을 로드하고 j가 2일 때는 적 헬리콥터의 위치 정보를 갖고 있는 hit 변수를 이용하여 점수판을 로드 합니다.

**8. 실행 화면**

텍스트, 지도, 스크린샷, 항공기이(가) 표시된 사진

자동 생성된 설명