한국산업기술대학교 게임공학과

포트폴리오

클라이언트 프로그래머 부분

박진수(Park Jin Soo)

목차

1. 알약전사(PillsFighter) 2

기본 설명 2

세부 설명 3

제작하면서 어려웠던 점 5

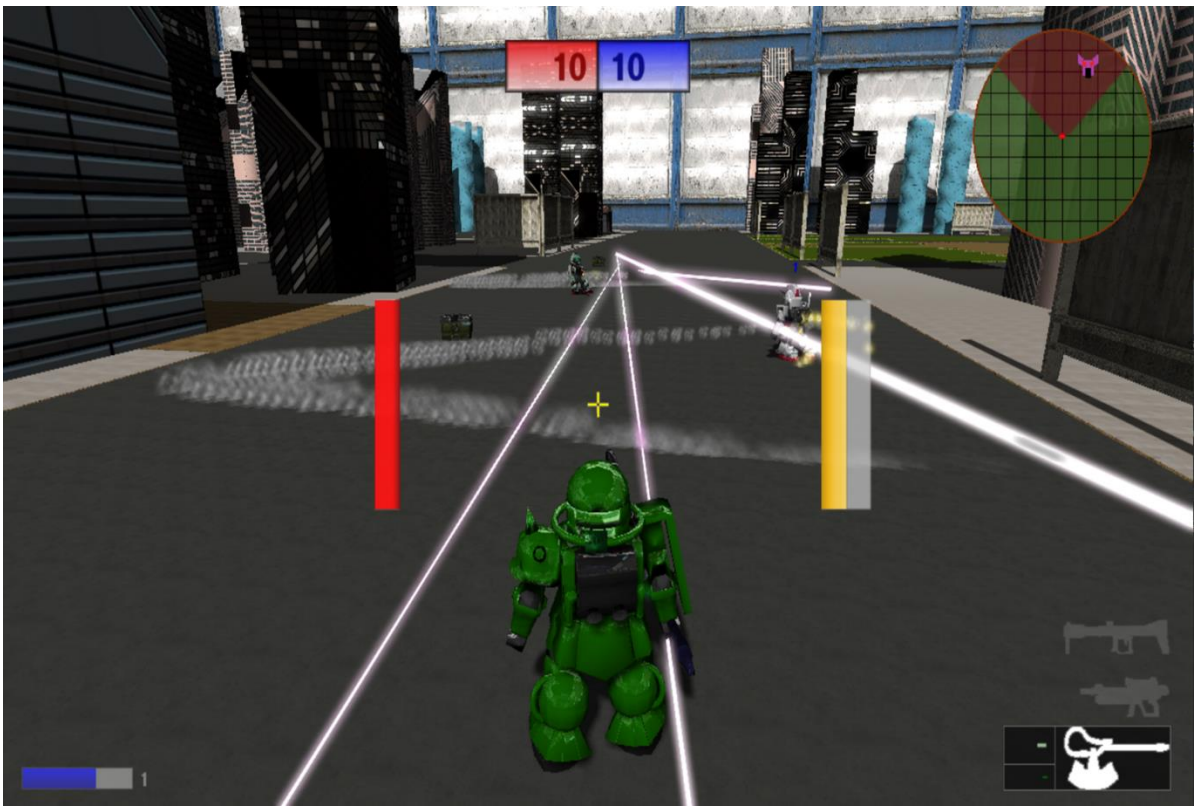
2. 오래버티기 7

기본 설명 7

3. 바리케이트 8

기본 설명 8

1. 알약전사(PillsFighter)



게임 이미지

기본 설명

게임 장르: 3D 3 인칭 멀티 대전 액션 게임

영상 링크: https://www.youtube.com/watch?v=dLpYjj6L7iM

프로젝트 소스: https://github.com/pjs0305/PortFolio

사용 언어: C++11

개발 환경: Visual Studio 2017, DirectX12, FMOD

제작 인원: 3 명

역할: 메인 클라이언트 (DirectX12 를 이용한 프레임워크 구축 및 게임 시스템, 다양한 그래픽 효과 구현)

제작 기간: 약 11 개월 소요

게임 내용: 로비에서 방을 만들어 인원을 모아 팀과 캐릭터를 정한 후, 게임이 시작되면 각 캐릭터마다 주어진 무기를 사용하여 상대팀의 점수를 0 으로 만들 때까지 싸우는 방식.

세부 설명

1. 시스템 구조:

클라이언트 실행 시 DirectX12 를 사용하기 위한 장치 생성, 메모리 할당 등 초기화 작업을 한 후 타이틀을 생성함. 타이틀, 로비, 룸 등을 Scene 클래스로 분류하였고 Scene 을 생성할 때 해당 Scene 에서 사용하는 리소스를 생성하도록 했음. 해당 Scene 에서 사용할 리소스들을 생성한 후 루프를 돌며 1 프레임마다 오브젝트에 시간을 적용시켜 Animate, 충돌 처리 등을 처리한 후 화면에 렌더링하도록 하였음.

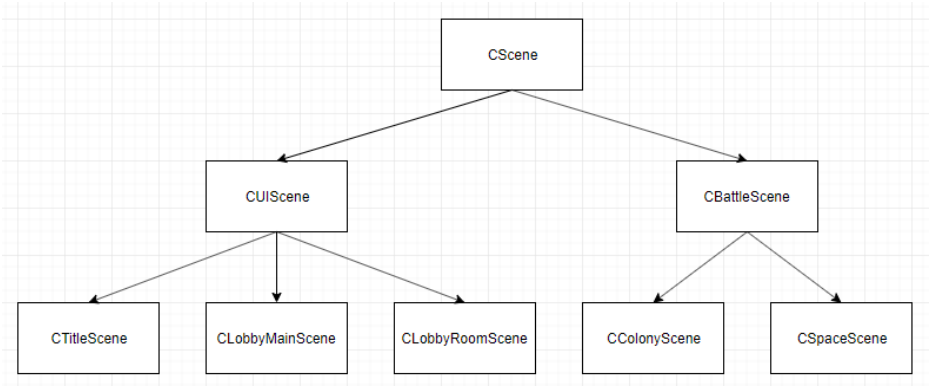
2. 클라이언트 상세 설명

A. GameFramework 클래스

DirectX12 디바이스, 명령 리스트, 스왑 체인 등 렌더링하기 위해 필요한 기본 자원들을 생성, 마우스, 키보드 입력 처리, 프레임 처리 등 프로그램의 기본 뼈대가 되는 클래스.

B. Scene 클래스

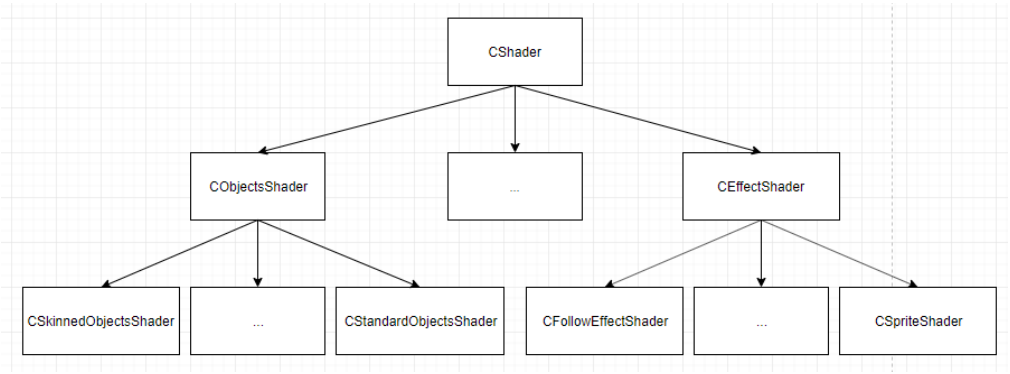
루트 시그니처, Shader 클래스, 조명, 지형 등 장면을 렌더링하기 위해 필요한 모든 것들을 관리하는 클래스.



C. Shader 클래스

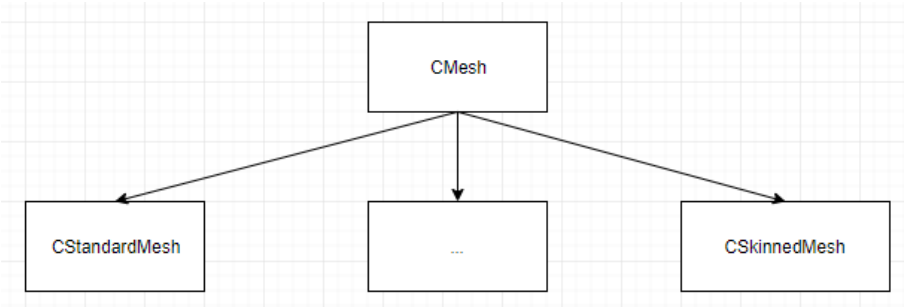
렌더링 파이프라인을 생성, 삭제, 명령 리스트에 Set 을 담당하는 클래스. 파이프라인 종류에 따라 세분화하였으며 배치 처리를 위해 멤버 변수로

파이프라인 뿐 아니라 해당 파이프라인을 통해 렌더링 되는 오브젝트들을 관리하는 클래스도 있음.



D. **Mesh 클래스**

정점 버퍼 생성, 삭제, 파이프라인에 정점 버퍼 연결을 담당하는 클래스. 어떻게 정범 버퍼를 생성, 파이프라인에 연결하는지에 따라 클래스를 세분화함.



E. **Texture 클래스**

.dds 파일을 읽어 텍스쳐 리소스를 생성, 렌더링 시 루트 시그니처에 연결하는 클래스.

F. **Material 클래스**

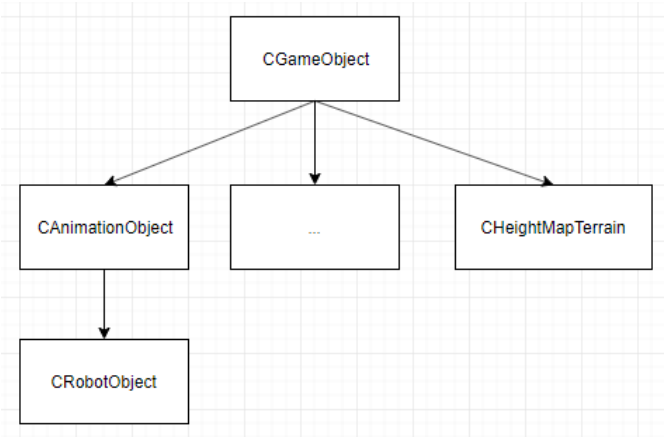
모델 파일을 읽어 재질 정보와 텍스쳐 파일 정보를 멤버 변수로 저장하고 렌더링 시 재질 정보를 상수 버퍼에 저장, Texture 클래스를 통해 텍스쳐 리소스를 연결하는 클래스.

G. **Model 클래스**

하나의 모델을 렌더링하기 위한 정보를 갖고 있는 클래스. Mesh, Material 클래스를 멤버 변수로 갖고 있으며, 모델 중에서도 관절을 표현해야 하는 모델도 있기 때문에 트리 구조 형태로 구현하였음.

H. **Object 클래스**

Model 클래스, 월드 변환 행렬, HP, 속도 등 하나의 오브젝트가 표현되기 위해 필요한 변수들을 멤버 변수로 갖는 클래스. 오브젝트 종류에 따라 세분화함.



I. **Camera 클래스**

투영 변환 행렬, 카메라 변환 행렬 등 게임에서 카메라와 같은 역할을 하는 클래스. 기본적으로 Player 를 따라다니게 하였음.

제작하면서 어려웠던 점.

1. **사용하기 힘든 그래픽 라이브러리**

   아직 DirectX12 에 대해 정리가 되지 않은 상태에서 프로그래밍을 시작했기 때문에 시스템 하나하나를 구현할 때마다 시행착오를 거치는데 그 때마다 포기하고 싶은 생각이나 스트레스 때문에 많이 힘들었습니다. 그래도 참고 견디면서 하다 보니 원하는 결과가 나왔을 때에는 나쁘지만은 않다는 생각을 했습니다.

2. **벡터와 행렬 등 수학적 지식 부족**

   오브젝트를 회전하거나 원하는 카메라 연출을 하기 위해서 벡터나 행렬에 대해 잘 알아야 하는데 그런 지식이 많이 부족했기 때문에 원하는 답을 얻을 때까지 식을 바꿔가며 결과를 확인하는 등 여러 시행착오를 거쳐야 했습니다.

3. **새로운 지식 습득**

   그림자 매핑, 환경 매핑, 모션 블러, 글로우 이펙트 등 게임에서 많이 봐 왔지만 어떻게 구현하는지 한 번도 생각해보지 않았던 것들을 프로젝트에 추가하기 위해 인터넷을 찾아보기도 하고 관련된 책을 사서 공부도 해봤지만 쉽지가 않았습니다. 특히 DirectX12 로 구현해야 되기 때문에 관련 정보가 많이 부족했고 리소스를

활용하는 데에 있어서 많이 힘들었습니다. 그래도 다른 문제들을 해결했던 것과 같이 끈기 있게 시도하며 원하는 결과가 나올 때까지 시행착오를 거친 덕분에 원하는 결과도 나올 수 있었고 무엇보다도 다행히 그래픽 효과 예제들은 쉐이더 코드들이 다 제공되어 있어서 정해진 시간 내에 예제 코드를 어느 정도 이해하고 프로젝트에 적용하는 데 큰 어려움이 없었고 DirectX12 를 다루는 것도 익숙해질 수 있었다고 생각합니다.

```
m_pComputeShader->SetMotionBlurPipelineState(pd3dCommandList);

pd3dCommandList->SetComputeRoot32BitConstants(COMPUTE_ROOT_PARAMETER_INDEX_MOTION_BLUR_INFO, 16, &m_xmf4x4PrevViewProjection, 0);

XMFLOAT4X4 xmf4x4Inverse = Matrix4x4::Inverse(m_xmf4x4CurrViewProjection);
pd3dCommandList->SetComputeRoot32BitConstants(COMPUTE_ROOT_PARAMETER_INDEX_MOTION_BLUR_INFO, 16, &xmf4x4Inverse, 16);
pd3dCommandList->SetComputeRoot32BitConstants(COMPUTE_ROOT_PARAMETER_INDEX_MOTION_BLUR_INFO, 1, &nWidth, 32);
pd3dCommandList->SetComputeRoot32BitConstants(COMPUTE_ROOT_PARAMETER_INDEX_MOTION_BLUR_INFO, 1, &nHeight, 33);
pd3dCommandList->SetComputeRoot32BitConstants(COMPUTE_ROOT_PARAMETER_INDEX_MOTION_BLUR_INFO, 1, &m_fFPS, 34);

pd3dCommandList->SetComputeRootDescriptorTable(COMPUTE_ROOT_PARAMETER_INDEX_DEPTH, m_d3dSrvDepthStencilGPUHandle);
pd3dCommandList->SetComputeRootDescriptorTable(COMPUTE_ROOT_PARAMETER_INDEX_MASK, m_d3dSrvMaskTextureGPUHandle);
pd3dCommandList->SetComputeRootDescriptorTable(COMPUTE_ROOT_PARAMETER_INDEX_INPUT_A, m_d3dSrvOffScreenGPUHandle);
pd3dCommandList->SetComputeRootDescriptorTable(COMPUTE_ROOT_PARAMETER_INDEX_OUTPUT, m_d3dUavMotionBlurScreenGPUHandle);
pd3dCommandList->Dispatch(nWidth, nHeight, 1);
```

```
[numthreads(8, 8, 1)]
void MotionBlurCS(int3 vDispatchThreadID : SV_DispatchThreadID)
{
    float zOverW = gtxtDepth[vDispatchThreadID.xy];
    float mask = gtxtMask[vDispatchThreadID.xy].r;

    if (mask > 0.5f)
    {
        gtxtRWOutput[vDispatchThreadID.xy] = gtxtInputA[vDispatchThreadID.xy];
        return;
    }

    float2 oriTex;
    oriTex.x = float(vDispatchThreadID.x) / float(gnWidth);
    oriTex.y = float(vDispatchThreadID.y) / float(gnHeight);

    float4 H = float4(oriTex.x * 2.0f - 1.0f, (1.0f - oriTex.y) * 2.0f - 1.0f, zOverW, 1.0f);
    float4 D = mul(gmtxInverseViewProjection, H);
    float4 worldPos = D / D.w;
    float4 currPos = H;
    float4 prevPos = mul(gmtxPrevViewProjection, worldPos);
    prevPos /= prevPos.w;
    float2 velocity = (currPos - prevPos).xy * 1.5f;

    float dt = T / fFPS;
    velocity *= dt;
    float2 du = velocity / SAMPLES;
    float2 texcoord = oriTex + du;

    float4 color = float4(0.0f, 0.0f, 0.0f, 1.0f);

    int nSamples = 0;

    for (int i = 1; i < SAMPLES; ++i, texcoord -= du)
    {
        if (texcoord.x >= 1.0f) break;
        if (texcoord.y >= 1.0f) break;
        if (texcoord.x <= 0.0f) break;
        if (texcoord.y <= 0.0f) break;

        int2 nIndex = int2(texcoord.x * gnWidth, texcoord.y * gnHeight);

        if (gtxtDepth[nIndex].r - zOverW > 0.1f) continue;
        mask = gtxtMask[nIndex].r;
        if (mask > 0.5f) continue;

        float4 currColor = gtxtInputA[nIndex];

        color += currColor;
        nSamples++;
    }
```

계산 쉐이더로 구현한 카메라 기반 모션 블러

(상) DirectX12 코드

계산 쉐이더를 사용하여 모션 블러 작업을 하기 위해 파이프라인 상태, 상수, 리소스를 Set 하는 코드

(좌) 쉐이더 코드(HLSL)

이전 프레임의 뷰프로젝션 행렬, 현재 프레임의 뷰프로젝션 행렬을 이용하여 Velocity 를 구한 후 픽셀을 Sampling.

2. 오래버티기

기본 설명

게임 장르: 2D 슈팅

영상 링크: -

프로젝트 소스: -

사용 언어: C

개발 환경: Visual Studio 2013, WinAPI

제작 인원: 2 명

역할: 클라이언트

제작 기간: 약 1 ~ 2 개월 소요

게임 내용: 최대한 오래 버티는 목적의 게임으로 시작 시 기본 무기가 주어지고 키보드 조작을 통해 적을 쓰러뜨려 무기, 스킬, 포션 등을 얻어 생명을 연장시킬 수 있다.

3. 바리케이트

기본 설명

게임 장르: 2D 디펜스

영상 링크: -

프로젝트 소스: -

사용 언어: Python

개발 환경: PyScripter, Pico2D

제작 인원: 1 명

역할: 클라이언트

제작 기간: 약 1 ~ 2 개월 소요

게임 내용: 성지키기 게임을 모티브로한 게임으로 오로지 마우스만을 사용하여 적을 공격하고 적을 처치하여 얻은 점수로 무기 구입, 업그레이드가 가능하다. 또한 적 무리에 껴서 일정 시간마다 보급품을 배달하는 아군 오브젝트가 있고 아군 오브젝트가 기지에 도착하면 지원 사격 유닛 혹은 기지 수리를 해주는 아군이 추가된다. 바리케이트(기지)는 총 3 단계까지 있으며 적의 공격으로 파괴될 때마다 바리케이트는 더욱 견고해진다. 적은 시간이 지날수록 강해지며 유닛도 다양하게 출현한다.