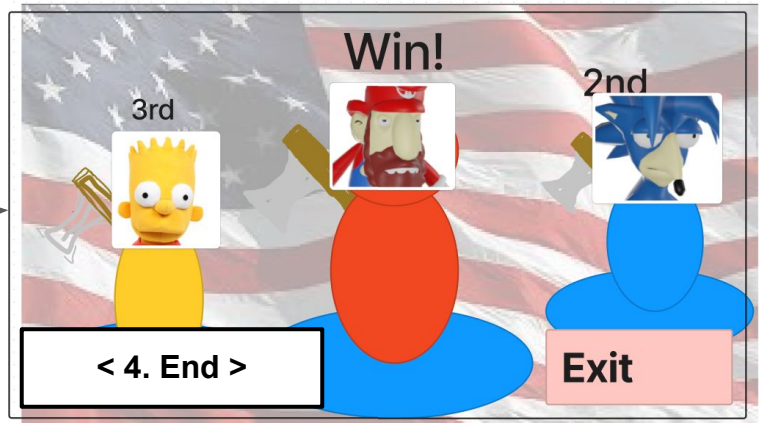
FIGHT VOTE

2019182028 이시영
2021180045 황유림
2021182022 엄미영

목차

01. 게임 개요

장르: 3인칭 배틀로얄 / 플레이 타임: 15분 / 플레이 인원: 3명

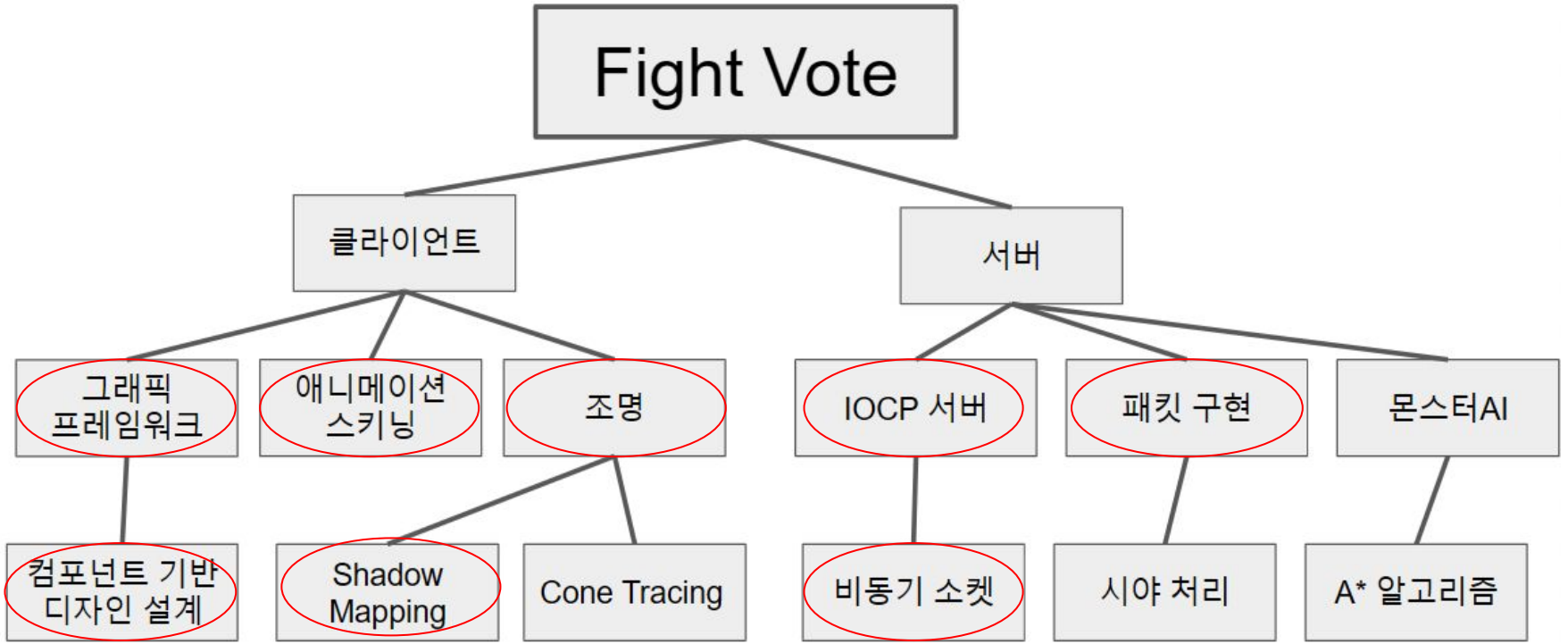


02. 게임 조작

이동		상호작용	
W	전면 이동	F	줍기/교환
A	좌측 이동	마우스	
S	후방 이동	오른쪽	조준
D	우측 이동	왼쪽	UI 클릭

03. 기술 요소와 중점 연구 분야

: 완료



04. 구성원 역할 분담

황유림

- Shadow mapping
- 컴포넌트 설계
- Descriptor Heap Manager
- Voxel Cone Tracing

엄미영

- Network framework 작성
- IOCP (서버-클라 연동)
- Packet 설계 & 구현
- Login, Add ,Move

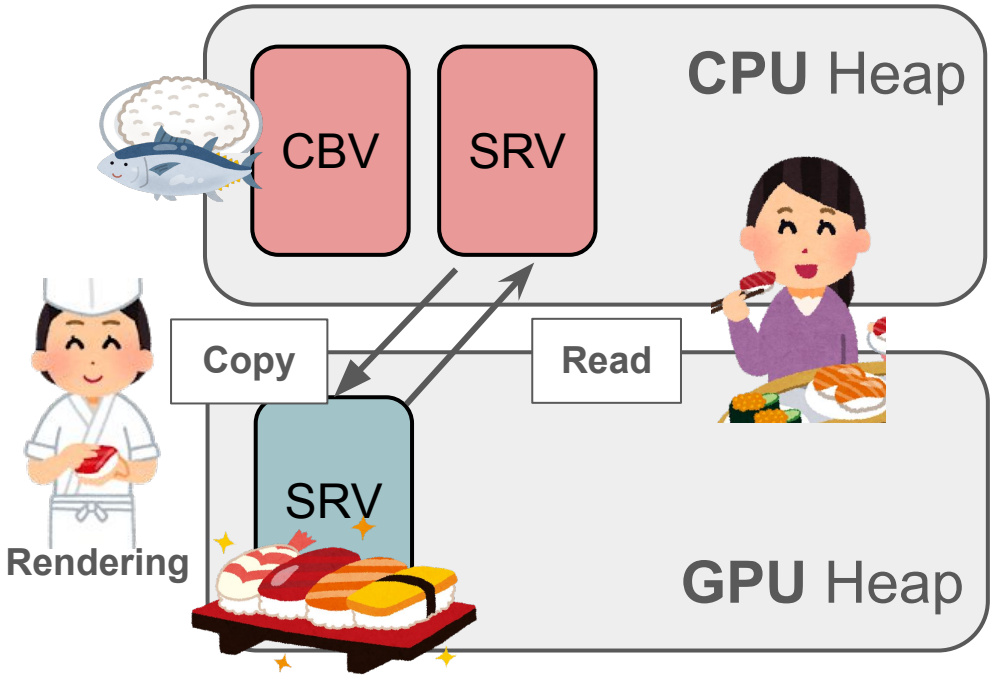
이시영

- Lighting
- Scene Load
- Fresnel Outline
- 에셋 수집/제작

05. 개발 내용

a. 클라이언트 - Descriptor Heap Manager

Descriptor Heap Manager

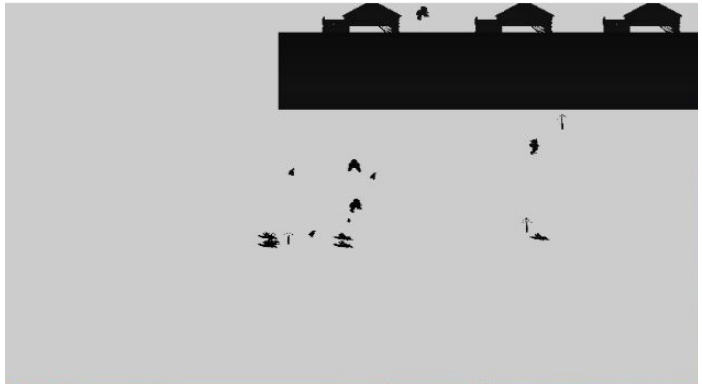


개발 이유: 렌더링 이후에 CPU에서 읽을 수 있는 리소스를 등록할 힙이 필요함.

장점: 여러개의 View를 만들고 CPU 힙에 넣어도, 그 중에 골라서 GPU 힙에 넣으면 되므로 부하가 적어짐

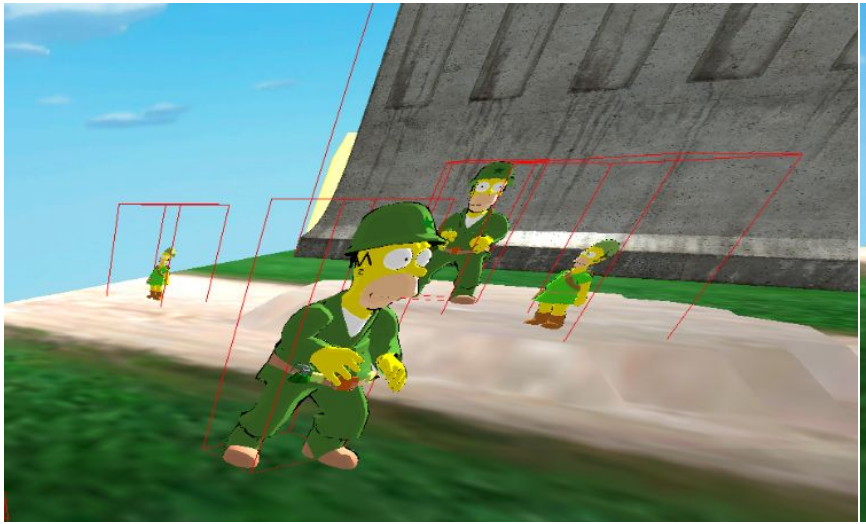
05. 개발 내용

a. 클라이언트 - Shadow Mapping

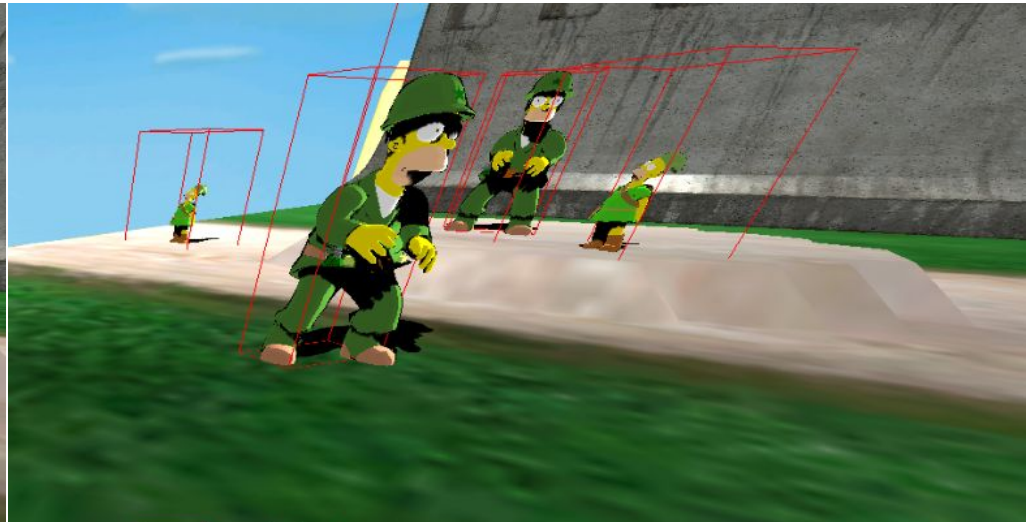


Depth Write Texture

Before

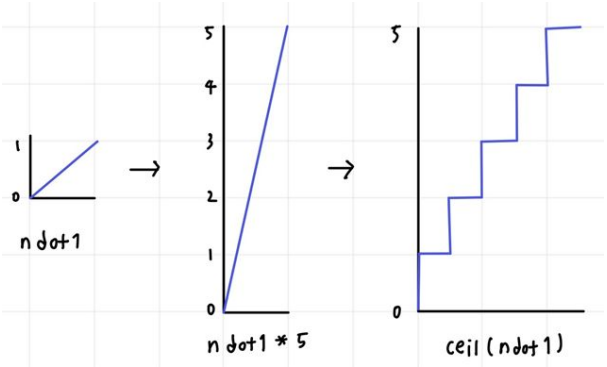


After



05. 개발 내용

a. 클라이언트 - Cartoon Rendering & Lighting & OutLine



Before

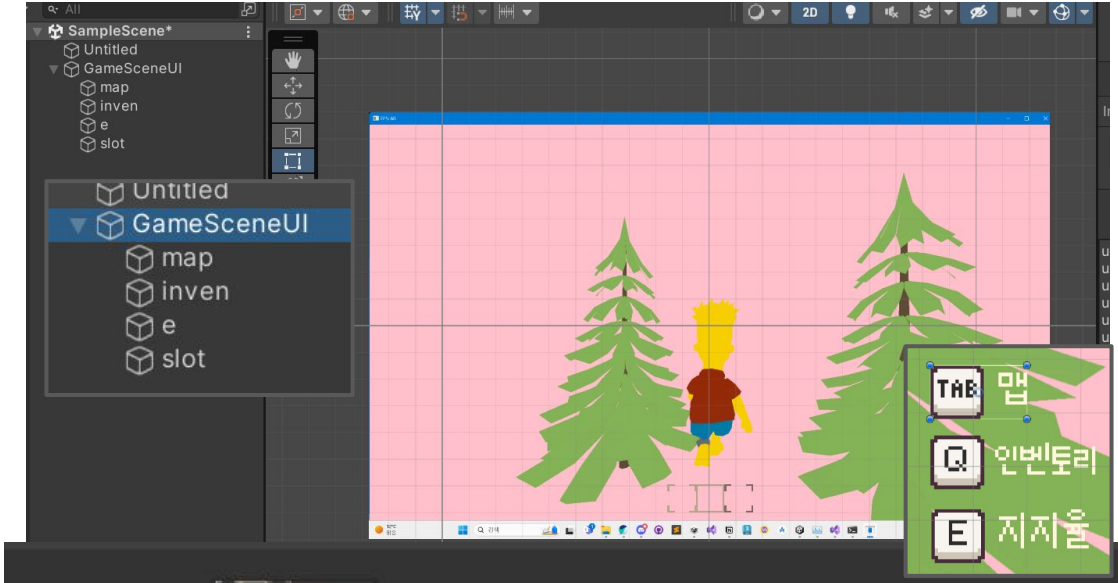
After

05. 개발 내용

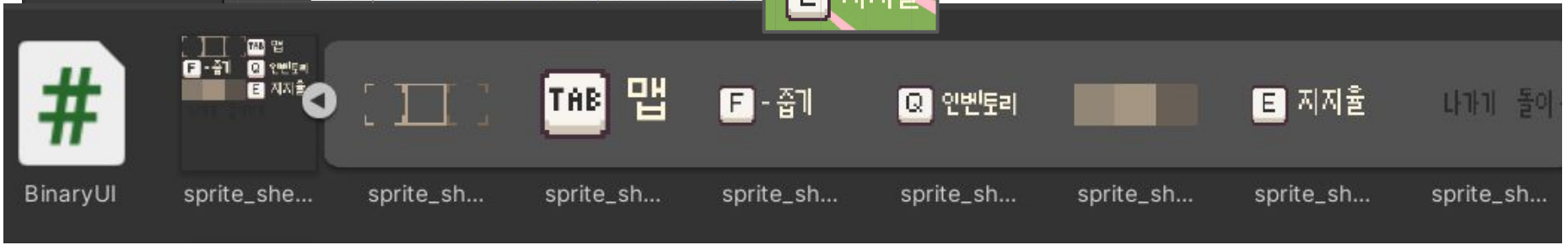
Unity2D로 UI Scene을 추출

a. 클라이언트 - UI 씬 로드

추출 스크립트 제작



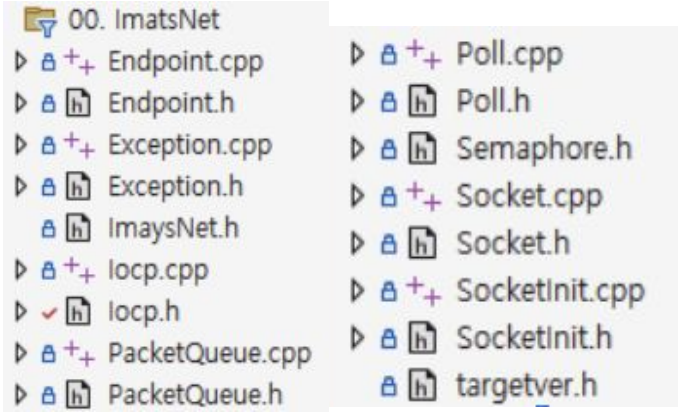

```
void Start()
{
    binaryWriter = new BinaryWriter(File.Open(string.Copy(gam

    for (int i = 0; i < transform.childCount; ++i)
    {
        WriteString("<UI>");
        WriteInteger("<ClassType>", m_classTypes[i]);
        WriteObjectName("<Name>", transform.GetChild(i));
        WriteInteger("<IsActive>", 1);
        WriteString("<RectTransform>");
        RectTransform rectTransform = transform.GetChild(i).G
        //Rect r = rectTransform.rect;
        Vector3[] vec3 = new Vector3[3];
        vec3[0] = rectTransform.localPosition;
        vec3[0].x /= ((float)25.5043 / 2);
        vec3[0].y /= ((float)15.99 / 2);
        vec3[1] = rectTransform.localRotation.eulerAngles;
        vec3[2] = rectTransform.localScale;
        WriteVectors(vec3);
        Vector2 vec2 = new Vector2(
            rectTransform.rect.width / (float)25.5043,
            rectTransform.rect.height / (float)15.99);
        WriteVector(vec2);
```



05. 개발 내용

b. 서버 - IOCP

- Iocp/IocpEvents class 사용하여 구현
- main.cpp에서 사용
- 게임서버프로그래밍 저자의
  예제/연습 파일 활용하여
  서버 시간에 배운 내용으로 변경 후 사용




```
class Socket;
class IocpEvents;

// I/O Completion Port 객체.
class Iocp {
public:
    // 1회의 GetQueuedCompletionStatus이 최대한 꺼내올 수 있는 일의 갯수
    static const int MaxEventCount = 1000;

    Iocp(int threadCount);
    ~Iocp();

    void Add(Socket& socket, void* userPtr);

    HANDLE m_hIocp;
    int m_threadCount; // IOCP 생성시 및 소켓 추가시 계속 사용되는 값인지라...
    void Wait(IocpEvents &output, int timeoutMs);
};

// IOCP의 GetQueuedCompletionStatus로 받은 I/O 완료신호들
class IocpEvents
{
public:
    // GetQueuedCompletionStatus으로 꺼내온 이벤트들
    OVERLAPPED_ENTRY m_events[Iocp::MaxEventCount];
    int m_eventCount;
};
```

05. 개발 내용

b. 서버 - Packet

- 비동기 방식으로 send와 recv가 서로 기다리지 않고 처리됨

- 수신된 패킷을 재조립하여 사용

- 패킷 type에 따른 Recv처리 (PacketProcess 함수 구현)

- send 패킷은 PacketQueue class를 만들어 처리함

```
int remain_data = recv_buf_Length + remoteClient->m_tcpConnection.m_prev_remain;
while (remain_data > 0) {
    unsigned char packet_size = recv_buf[0];
    if (packet_size > remain_data) // // 남은 데이터가 현재 처리할 패킷 크기보다 적으면 잘
        break;

    //패킷 처리
    PacketProcess(remoteClient, recv_buf);

    //다음 패킷 이동, 남은 데이터 갱신
    recv_buf += packet_size;
    remain_data -= packet_size;
```

```
void PacketProcess(shared_ptr<RemoteClient>& _Client, char* _Packet)
{
    switch (_Packet[1]) {

    case PACKET_TYPE::P_CS_LOGIN_PACKET:
    {
        CS_LOGIN_PACKET* recv_packet = reinterpret_cast<CS_LOGIN_PACKET*>(_Packet);
        _Client->m_id = nextClientID++;
```

```
CS_WALK_ENTER_PACEKET send_packet;
send_packet.m_size = sizeof(CS_WALK_ENTER_PACEKET);
send_packet.m_type = PACKET_TYPE::P_CS_WALK_ENTER_PACKET;
PacketQueue::AddSendPacket(&send_packet);
```

06. 문제점 및 보완책

애니메이션 버그	터레인 텍스처 화질 저하
문제점: 애니메이션의 상/하체 분리로 인한 어색한 움직임 보완책: 애니메이션 관련 코드 수정 필요	문제점: 터레인 크기가 커서 텍스처의 화질 저하되어 보임 보완책: 디테일 텍스처로 보완할 예정
외곽선 보완 필요	**최적화 필요**
문제점: 로우 폴리로 인한 내적 계산 오류 발생 (외곽선 X) 보완책: 2Pass 기법을 이용해 보완할 예정	문제점: NPC 수 증가와 플레이어의 수많은 패킷으로 인한 랙 발생 보완책: 시야처리 적용할 예정

07. 향후 개발 일정

일정	서버 구현 계획		클라이언트 구현 계획		
5월	조작 패킷 & 충돌 작업 완료	시야처리	Voxel Cone Tracing	카툰 렌더링 마무리	메인씬 제작
6월	게임 콘텐츠 패킷 처리 (Scene따라)	AI구현 (A* 알고리즘)	로그인,로비,인게임, 게임종료 씬 구현	UI 컨텐츠 제작	게임 콘텐츠 제작
7월	동기화&최적화	기타 구현	최적화 작업	마무리 작업	후 처리

07. 데모 시연

참고 자료