[Toy Ground(토이 그라운드)]

18 주	2020. 4. 25 ~ 2021. 5. 1	작성자	김동석

이번 주 한 일

[0] 공동

04.26 주간 회의

- 일주일간 계획공유

[1] 김영준(서버)

주간목표:

- Battle Server 완성, 버그 확인용 Dummy 보강

-

04.26, 29 battle server WorkerThread ProcPacket 완성

```
void WorkerThread::ProcThread() {
    while (true) {
        DWORD num_byte;
        ULONG key;
        PULONG p_key = &key;
        WSAOVERLAPPED* p_over;

        bool bSuccess = GetQueuedCompletionStatus(SR::g_iocp, &num_byte, (PULONG_PTR)p_key, &p_over, INFINITE);
        if (!bSuccess) {
            int err = WSAGetLastError();
            BattleServer::GetInstance()->error_display("GQCS ERROR", err);
        }

        SOCKET clientSocket;
        int evkey = static_cast<int>(key);
        if (evkey == EVENT_KEY);
        else if (evkey == LOBBY_SERVER_KEY) {
#ifdef LOG_ON
            std::cout << "MatchMakingServerKey Return\n";
#endif
            clientSocket = SR::g_clients[key]->m_s->GetSocket();
        }
        else {
            if (SR::g_clients[key]->m_s = nullptr) {
#ifdef LOG_ON
                std::cout << key << "- socket was nullptr\n";
#endif
                continue;
            }
            clientSocket = SR::g_clients[key]->m_s->GetSocket();
            if (num_byte == 0) {
                //disconnect
                DisconnectClient(key, clientSocket);
                continue;
            }
        }
        EX_OVER* ex_over = reinterpret_cast<EX_OVER*>(p_over);

        switch (ex_over->ev_type) {
```

```
48      switch (ex_over->ev_type) {
49      case EV_RECV: { ... }
98      case EV_SEND: { ... }
102     case EV_UPDATE: { ... }
108     case EV_TICK: { ... }
115     case EV_FLUSH_MSG: { ... }
120     case EV_MOVE_ENABLE: { ... }
126     case EV_MAKE_MOVE_DISABLE: { ... }
132     case EV_UPDATE_DB: { ... }
137     case EV_RESET_ROOM: { ... }
143     default: {
144         std::cout << "worker > unknown event type" << ex_over->ev_type << std::endl;
145     }
146     }
147 }
```

```
case EV_RECV: {
    cout << "recv\n";
    char* buf = SR::g_clients[key]->m_recv_over.net_buf;
    unsigned int psize = SR::g_clients[key]->curr_packet_size;
    unsigned int pr_size = SR::g_clients[key]->prev_packet_data;
    message msg{ -1 };
    while (num_byte > 0) {
        if (psize == 0) psize = (BYTE)buf[0];
        if (num_byte + pr_size >= psize) { //패킷완성
            unsigned char p[MAX_BUFFER];
            memcpy(p, SR::g_clients[key]->packet_buf, pr_size);
            memcpy(p + pr_size, buf, psize - pr_size);
            msg = ProcPacket(key, p);
            if (msg.type != NO_MSG) {
                int room_id = SR::g_clients[key]->room_id;
                SR::g_rooms[room_id]->PushMsg(msg);
            }

            num_byte -= psize - pr_size;
            buf += psize - pr_size;
            psize = 0;
            pr_size = 0;
        }
        else {
            ATOMIC::g_clients_lock.lock();
            memcpy(SR::g_clients[key]->packet_buf + pr_size, buf, num_byte);
            ATOMIC::g_clients_lock.unlock();
            pr_size += num_byte;
            num_byte = 0;
        }
    }
    SR::g_clients[key]->curr_packet_size = psize;
    SR::g_clients[key]->prev_packet_data = pr_size;

    DWORD flags = 0;
    ZeroMemory(&ex_over->over, sizeof(WSAOVERLAPPED));
    ZeroMemory(&msg, sizeof(message));
    ex_over->ev_type = EV_RECV;

    int ret = WSARecv(clientSocket, ex_over->wsabuf, 1, 0, &flags, &ex_over->over, 0);
    if (ret == SOCKET_ERROR) {
        int err_no = WSAGetLastError();
        if (err_no != ERROR_IO_PENDING) {
            BattleServer::GetInstance()->error_display("WSARecv Error", err_no);
            while (true);
```

Event별로 처리할 수 있게 구현, 기존 코드는 뼈대만 작성하고 내용이 비어있었지만 해당 내용 전부 구현 완료

04.27~28 쉐이더 프로그래밍 시험 준비 및 시험

강의 내용을 다시 복기하며 시험 준비함

04.30 디버깅용 Dummy 제작 마무리

```
void SendBattleLoginPacket(int id);
void SendJoinPacket(int id, int room_no);
void SendReadyPacket(int id);
void SendGameStartPacket(int id);
```

```
99  void Dummy::ProcessPacket(int id, unsigned char packet[])
100 {
101     switch (packet[1]) {
102     case LC_LOGIN_OK: { ... }
110     case LC_LOGIN_FAIL: cout << "login_fail\n"; break;
111     case LC_USERINFO: { ... }
121     case BC_AUTO_ACCEPT_OK: { ... }
129     case LC_MATCHSTART: { ... }
140     case BC_PLAYER_ROT: break;
141     case BC_PLAYER_POS: break;
142     case BC_JOIN_OK: { ... }
147     case BC_JOIN_FAIL: cout << "join fail\n"; break;
148     case BC_AUTO_ACCEPT_FAIL: cout << "auto accept fail\n"; break;
149     case BC_ROOM_ENTERED: cout << "room entered\n"; break;
150     case BC_NEW_ROOM_HOST: cout << "new room host\n"; break;
151     case BC_LEFT_TIME: cout << "left time\n"; break;
152     case BC_READY: cout << "ready\n"; break;
153     case BC_GAME_START: cout << "start\n"; break;
154     case BC_GAME_START_AVAILABLE: { ... }
158     case BC_UPDATED_USER_INFO: { ... }
162     default: { ... }
166     }
```

추가된 패킷 처리 추가 및 함수 추가, 패킷 이동 함수 추가

[2] 김동석(클라이언트)

주간 목표 :

- 애니메이션 컨트롤러

04.25 쉐이더 수정

- .hlsl 파일 수정

쉐이더에 계층이 있는 모델을 처리할 때 그냥 평범한 오브젝트와 다르게 처리할 수 있도록 SKINNED define을 따로 설정해주었다.
Bone의 정보를 받아와 메쉬와 연결해주는 부분

- RootSignature 수정

5번째에 Bone의 정보를 담을 수 있는 ConstantBufferView 생성
Skinned Model을 읽어올 때 Bone의 정보를 전달해준다.

- Pipeline 수정

Skinned Model을 관리할 g_SkinnedPSO 파이프라인 생성

04.26 테스트 캐릭터 불러오기

- BuildSkinnedModel()

Mesh, Bone로 이루어진 테스트 캐릭터를 불러오는 함수이다.
테스트 캐릭터를 사용하는 이유는 실제 ToyGround에서 사용할 에셋인 Cowboy에 적용되는 애니메이션을 완벽하게 다 찾지 못해서 애니메이션 컨트롤러를 제작하는 동안에는 Running, Idle, Jump 애니메이션을 적용 가능한 테스트 캐릭터를 사용해서 테스트할 것이다.

- BuildSkinnedModelAnimation()

.Anim 파일로부터 캐릭터에 적용할 Animation 정보를 불러온다.

04.28 ~ 04.30 애니메이션 컨트롤러

- MoveCommand

SetAnimationKeyState()
캐릭터의 상태에 따른 Animation을 정해준다.
AnimationPlayerState
현재 플레이어의 애니메이션 상태
AnimationKeyState
플레이어가 앞으로 해야하는 애니메이션

- CommandCenter

애니메이션을 관리하는 Class
Jump 상태에서는 다른 애니메이션으로 바뀌지 못하도록 관리한다.
List에 애니메이션 명령들을 Push, Pop하며 관리한다.
기본적으로 Idle이 들어있다.

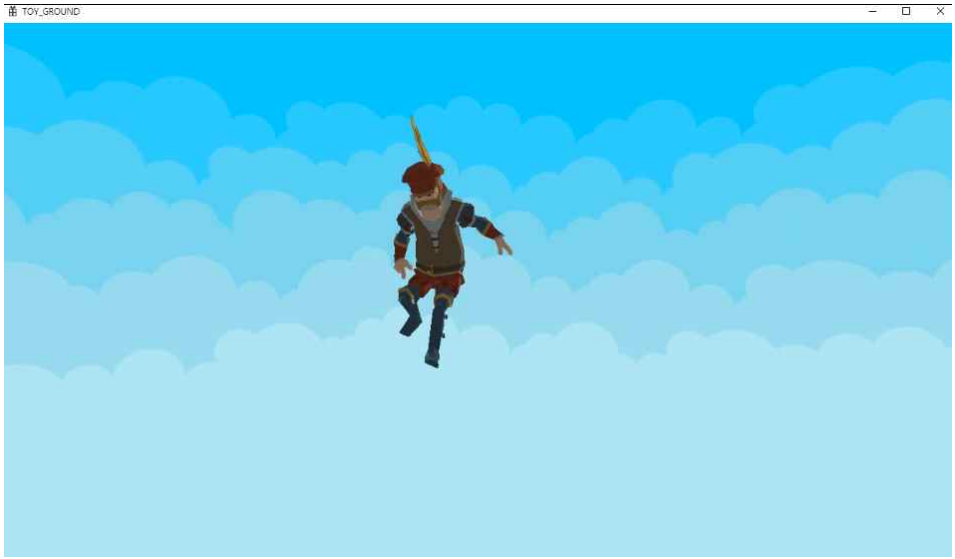
- GameController에 OnKeyPressed/OnKeyReleased 추가

OnKeyPressed
키입력에 따라 Command Center에 해당 애니메이션 Push
OnKeyReleased
키입력에 따라 Command Center에 해당 애니메이션 Pop

- 화면에 캐릭터가 그려지지 않는 오류 발생

애니메이션 컨트롤러 관련 코드들을 모두 작성한 후 실행을 했는데 캐릭터가 화면에 그려지지 않았다(거의 하루 헤맸다...)

시도 1: 저번에 오브젝트를 읽어올 때 .mesh 파일에서 파일경로에 띄어쓰기가 포함되어 있어 값이 한 칸씩 뒤로 밀려왔던 것처럼 제대로 읽어오지 못하고 있는걸까?
-> 콘솔창으로 출력해 확인해보니 제대로 읽어오고 있었다.
시도 2: hlsl 파일 내부에서 계산 실수가 있는게 아닐까?
-> 확인해보니 계산 실수는 없었다.
시도 3: 처음부터 직접 디버깅해보며 확인
->원인
AnimationController의 Update에서 BoneTransfroms 부분을 실시간으로 갱신해 주고 있지 않았었다.
->해결방안
관련 코드 추가



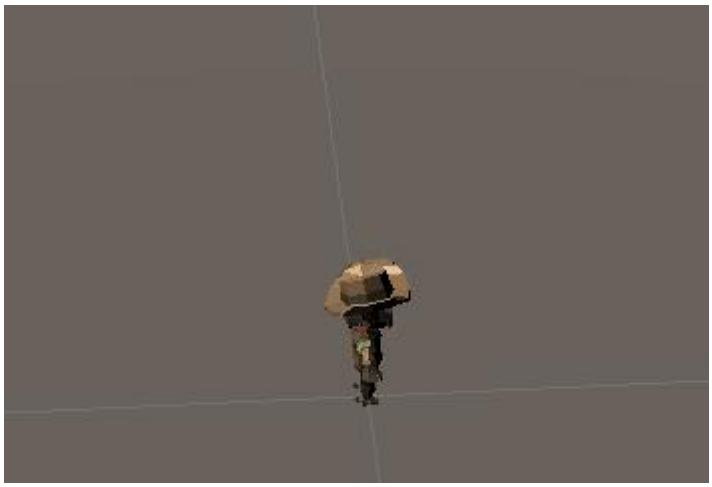
[테스트 캐릭터가 Jump하는 모습]

- 애니메이션이 자연스럽게 연결되도록 수정
애니메이션 간의 연결(Running 애니메이션에서 Idle 애니메이션으로 변할 때)이 부자연스럽다. 팔을 올리고 있다가 키 입력을 중지하면 Idle로 돌아갈 때 팔이 순간이동한다.
->해결방안
보간을 사용. AnimationController에 현재 동작하고 있는 애니메이션(m_PlayerState)와 변경되어야 할 애니메이션(m_KeyState)를 이용해 STATE_JUMP_TO_IDLE 등으로 설정해주었다.
->결과
동작과 동작간의 자연스러운 연결이 된다.

05.01 Cowboy Asset

- Cowboy Asset 수정

ToyGround에서 사용할 Asset인 Cowboy를 Unity에서 수정했다.
전에 수정할 때는 게임세계의 크기와 일치 시켜주기 위해 Scale할 때 Bone의 위치는 그대로 있는데 Mesh의 크기만 줄어들어 팔다리가 엄청 길고 얇은 팔척귀신처럼 수정이 되었는데 그 부분을 제대로 줄어들도록 수정했다.



[Unity에서 제대로 줄어든 모습]

- **오류 발견**

화면 밖에서 마우스가 있다가 게임 실행창을 클릭하면 위아래가 반전되는 오류가 발생했다. 아마 화면을 이동할 때 마우스를 실행창의 중앙으로 강제로 이동 시킬 때 발생하는 오류인 것 같아서 수정 중이다.

[화면이 상하 반전이 된다]

	- **FBX Converter(수정 중)** 저번 주에 실행이 안되는 오류를 계속 수정 중이다. Converter를 수정 완료 하는대로 Unity를 다룰 줄 아는 영준이에게 Unity에서 Cowboy 캐릭터에 애니메이션이 자연스럽게 붙도록 하는 것을 배울 것이다. 현재 Running, Dance, Jump 등의 무료 에셋은 구했지만 Cowboy 캐릭터의 공격 모션인 쌍권총을 발사하는 애니메이션의 에셋은 구하지 못했기 때문에 직접 제작할 방법도 생각 중이다.
다음 주 할 일	**[0] 공동** **05.04 주간 회의 (19주차)** **안건 : 일일계획 공유** **교수님 면담 피드백 수정** **[1] 김영준 (서버)** 주간 목표 - Network 연결 해결 **[2] 김동석 (클라이언트)** 주간 목표 - FBX Converter 수정 - Cowboy 캐릭터 추가
문제점	**[1] 김영준 (서버)** - 매칭 시스템을 제외하면 계획했던 개발을 완료했다, 중간발표 전까지 연결과 매칭, 클라이언트와 연동을 완료할 수 있도록 노력하겠습니다. - 게임서버 시험을 준비하면서 매치메이킹을 배틀서버에서 처리하면 손해보는 것을 인지했음, 일정을 쪼개서 따로 매치메이킹 서버를 분리하여 구현할 수 있도록 노력하겠습니다. **[2] 김동석 (클라이언트)** - 애니메이션 컨트롤러를 완성하는데 거의 3주 정도 소모한 것 같다. 다행히 목표했던대로 4월 내로 마무리해서 다행이다. - 중간발표 전까지 좀 더 속도를 높여 준비해야겠다. 어려운 부분은 거의 다 완성했다. 파이팅!

[추가]