[Toy Ground(토이 그라운드)]

17 주	2020. 4. 18 ~ 2021. 4. 24	작성자	김영준

이번 주 한 일

[0] 공동

04.19 주간 회의

- 일주일간 계획공유

[1] 김영준(서버)

주간목표:

- Player Interface, Toy class 생성

04.19 유저의 상태를 처리하는 Player Interface 제작

```
class Object;

class Player {
public:
    Player();
    Player(const Player& rhs);
    Player& operator=(const Player& rhs)noexcept;
    virtual ~Player();

public:
    void Initialize();
    virtual void Reset();

    virtual bool Update(float elapsedTime) = 0;
```

```
protected:
    int m_id;       //=g_clients key val
    Object* m_cur;
    bool m_isEmpty;
    bool m_isReady;
    bool m_isMoveable;
    bool m_isDead;

    char m_idStr[MAX_STR_LEN];
    int m_mmr;

    bool m_keyW{ false };
    bool m_keyA{ false };
    bool m_keyS{ false };
    bool m_keyD{ false };
    bool m_keyJump{ false };
    bool m_animJump{ false };

    char m_prevAnimType;

    float m_fTimeElapsedJump{};
```

필요한 객체 종류를 동석이와 상의를 통해 결정하고
(시작 카운트 다운용 Moveable 등)
getter, setter 및 초기화 함수를 제작

04.20~21 게임서버 시험 준비

cache 관련 내용
WSABUF 관련 내용
volatile 키워드 등

volatile: 항상 메모리에 접근하여 변수를 확인
→ (volatile이 아니면 메모리를 레지스터에 등록하여 항상 확인하지 않을 수 있음)

C++ 복사 캡션 ...

```
volatile int * a; //volatile int를 가르키는 "포인터"
*a = 1; //최적화로 인한 오동작 없음
a = b; //최적화로 인한 메모리 접근 생략, 순서 변경(오동작 가능)

int * volatile a; //int를 가르키는 "volatile 포인터"
*a = 1; //최적화로 인한 메모리 접근 생략, 순서 변경(오동작 가능)
a = b; //최적화로 인한 오동작 없음
```

강의 내용을 다시 복기하며 시험을 준비함

04.22~23 Player가 조종하는 Toy 구현

```cpp
class Toy : public Player {
public:
    Toy();
    Toy(const Player& rhs) : Player(rhs) { Initialize(); }
    ~Toy() override;

    void Initialize();
    void Reset() override;
    bool Update(float elapsedTime) override;

public:
    void SetHP(const unsigned short& hp);
    unsigned short GetHP() const;

    void SetMaxHP(const unsigned short& max_hp);
    unsigned short GetMaxHP() const;

public:
    int GetAnimType() override;

private:
    unsigned short m_HP;
    unsigned short m_MaxHP;
};
```

```cpp
XMFLOAT3 look = m_cur->GetLook();
XMFLOAT3 up = m_cur->GetUp();

if (m_keyW) { vertical++; }
if (m_keyS) { vertical--; }
if (m_keyA) { horizontal++; }
if (m_keyD) { horizontal--; }
if (m_keyJump) {
    force = MathHelper::Add(force, up, 1.f);
    m_animJump = true;
    m_keyJump = false; //anim 끝나고 false 처리?
}
if (vertical != 0 || horizontal != 0) {
    force = MathHelper::Add(force, look, 1.f);
}

MathHelper::Normalize(force);
if (!MathHelper::IsZero(force)) {
    float curForceAmountXZ{ m_cur->GetForceAmountXZ() };
    float curForceAmountY{ m_cur->GetForceAmountY() };
    force.x *= curForceAmountXZ;
    force.y *= curForceAmountY;
    force.z *= curForceAmountXZ;

    m_cur->AddForce(force, elapsedTime, false);
}

return m_cur->Update(elapsedTime, true);
```

	(update속 키 입력 처리) player를 부모로 받는 Toy class 구현 다른 특징은 키입력 및 체력 수치를 가지고 처리할 수 있게 함 **[2] 김동석(클라이언트)** **주간 목표 :** **- 애니메이션 컨트롤러(50%)** **04.18 카메라 오류 수정** **- 1인칭 카메라 오류 수정** 1인칭 시점에서 pitch값이 360도 회전이 되는 것을 수정했다. 그 외에 다른 시점에서 자연스럽지 않게 화면이 이동하는 부분을 자연스럽게 회전하도록 수정했다. **04.19 ~ 04.20 애니메이션 컨트롤러** **- 애니메이션 강의 복습(컨트롤러 부분)** 애니메이션 강의 부분을 복습하고 정리했다. **- Skinned Mesh 추가** 계층이 있는 모델을 다룰 수 있는 SkinnedMesh를 추가했고 SkinnedData 파일에서 애니메이션 시작시간과 종료시간, 키프레임을 관리할 수 있도록 했다. 키프레임 사이는 보간을 통해서 구하도록 만들고 있다. **- 애니메이션 컨트롤러** 애니메이션 모델을 Mesh, Skeleton, Anim 파일로 나누어서 관리하는데 Mesh와 Skeleton을 먼저 불러와 캐릭터에 적용하고 각각의 Animation들을 Anim 파일 여러 개 나누어 저장한다. 애니메이션 컨트롤러는 이렇게 분리되어있는 애니메이션들을 캐릭터의 상태(Idle. Running, Jump 등)에 따라서 매칭시켜주는 클래스이다. **- 어려운 부분** 3월 달에 개발해놓은 애니메이션 관련한 것은 하나의 모델에 하나의 애니메이션이 적용된 FBX 파일을 불러와 그대로 그리면 됐는데 이제는 한 캐릭터가 여러 개의 애니메이션을 해야하다 보니까 이를 분리시키고 자연스럽게 연결하도록 하는 부분이 어렵다. 강의 복습과 구글링으로 최적의 방법을 찾아서 개발해야겠다.

	04.21 ~ 04.22 중간 시험 준비 **04.23 FBX Converter 수정** **- Mesh, Skeleton, Anim** 위에서 개발하고 있는 애니메이션 컨트롤러에 적용하기 위해서 하나의 파일에 Mesh, Skeleton, Anim의 정보를 한번에 담는 것이 아니라 분리해서 저장하도록 수정하고 있다. Toy_Ground의 캐릭터는 같은 스켈레톤을 공유하기 때문에 이 방식으로 수정하면 2개의 캐릭터 외에도 다른 캐릭터의 메쉬를 씌우기도 편할 것이다. -> 파일을 읽어오는 부분에서 오류가 발생하는데 수정 중이다. **04.24 애니메이션 컨트롤러** **- 애니메이션 컨트롤러 개발(50%)** Mesh, Skeleton, Anim 값을 가지고 있는 테스트 캐릭터를 기준으로 개발 중이다. 캐릭터에 적용한 텍스쳐 파일을 dds 파일로 만들어 추가하고 Skeleton과 Anim 파일의 값을 읽어오는 부분을 제작했다. -> 파이프라인에 SkinnedMesh 관련한 부분을 추가하고 hlsl 파일을 수정해야 한다.
다음 주 할 일	**[0] 공동** **04.26 주간 회의 (18주차)** **안건 : 일일계획 공유** **[1] 김영준 (서버)** 주간 목표 - Network 연결 해결 **[2] 김동석 (클라이언트)** 주간 목표 - 애니메이션 프레임워크에 추가 - FBX Converter 수정
문제점	**[1] 김영준 (서버)** - 클라이언트 개발을 하면서도 항상 필요했던 구현을 서버에서 진행하면서 서버 안에서 처리해야하는 로직에 대한 구현을 했다, 유니티의 input에서 아이디어를 얻어서 상,하/좌,우 와 같이 동시에 눌리면 처

	리하지 않는 부분을 구현했다, 중간 평가 이전에 더 빠르게 구현해 나가겠습니다. - 학교에서 확진자가 늘어나며 대면시험으로 공지했던 많은 과목들이 비대면 보고서로 중간고사 형태가 변경되면서 다음주까지 제출해야 할 보고서가 갑자기 많아졌다. 일요일 안에 모든 보고서를 처리하고 졸업작품 진행에 차질 없도록 진행하겠습니다. **[2] 김동석 (클라이언트)** - 중간시험 절반 이상을 이번 주에 봤다. 다음 주에 보는 시험들은 이번 주에 미리 정리해 두어서 다음 주에는 프로젝트에 시간을 많이 쓸 수 있을 것 같다. - 다음 주에 서버와 클라이언트를 붙이기 위해 프로젝트를 좀 다듬어야겠다. - 중간발표 시간이 다가올수록 조급한 마음이 자꾸 드는데 평정심을 가지고 계획한대로 차근차근해야겠다.

[추가]