[Toy Ground(토이 그라운드)]

13 주	2020. 3. 21 ~ 2021. 3. 27	작성자	김영준

이번 주 한 일

[0] 공동

03.22 주간 회의

- 일주일간 계획공유

[1] 김영준(서버)

주간목표:

- **공유자원 관리 클래스 구현**
- **BattleServer 구현**

*사진은 모두 notion을 통해 정리한 내용을 정리한 것을 캡처했습니다.

03.22~23 스레드 관리를 위한 클래스 구현

Thread Interface을 상속받는 Thread들, Thread Interface을 관리하는 ThreadMnr 클래스를 만들어 Thread을 관리할 수 있도록 구현, Effective C++ 의 상속 관련 부분을 읽은 내용을 사용하여 순수가상함수를 통해 Interface을 구현

▼ #34: interface의 상속과 구현을 위한 상속의 차이를 두자

```
class shape{
public:
  //순수 가상 함수
  virtual void draw() const = 0;
  //단순 가상 함수
  virtual void error(const std::string& msg);
  //비 가상 함수
  int objectID() const;
}
```

- public 상속의 종류: 함수 인터페이스(선언) / 함수 구현
- 순수 가상 함수: 함수의 인터페이스만 물려줌
  - 자식 class가 순수 가상 함수를 다시 선언해야 함
  - 부모 class가 선언만 하고 정의하지 않음
- 단순 가상함수: 함수의 인터페이스와 구현 상속
  - 자식 class에서 오버라이드 가능한 함수 구현부 제공
  - 기본구현이 자식 class마다 필요하거나 필요없는경우 순수 가상함수로 선언 후 기본 구현 부분을 protected에 비 가상 함수로 구현한뒤 사용
  - 기본 구현이 필요한 경우 자식 class의 순수가상함수 구현 부분에 비 가상 함수로 호출
  - 기본 구현이 필요없는 경우 자식 class의 순수가상함수 구현 부분에 따로 구현
- 비 가상 함수: 함수의 인터페이스와 필수적 구현 상속 → 클래스 파생에 관계없음

public 상속에서 파생 클래스는 항상 기본 클래스의 인터페이스를 전부 물려받음

순수 가상 함수는 인터페이스 상속만을 허용함

단순 가상 함수는 인터페이스 상속 + 기본 구현의 상속도 가능하도록 지정함

03.24 게임엔진1 팀 프로젝트 개발

03.25 BattleServer 변수를 사용하기 위한 Singleton 구현

```
template<class Handler>
class Singleton{
protected:
    Singleton() {};
    virtual ~Singleton(){};

public:
    static Handler* GetInstance(){
        if( nullptr == m_instance )
            m_instance = new Handler;
        return m_instance;
    }

    static void DeleteInstance(){
        delete  m_instance;
        m_instance = nullptr;
    }
private:
    static Handler* m_instance;
};

template<class Handler> Handler* Singleton<Handler>::m_instance = nullptr;
```

BattleServer에 정의해둔 SendPacket들과 Addr, Socket 정보를 각 Thread에서 다른 과정 없이 GetInstance을 통해 사용할 수 있도록 Singleton을 구현

03.26 BattleServer 구현(중)

정해둔 Packet들을 전송하기 위한 Send@@Pakcet 및 Accept, init 초기화 등 게임 서버에 필요한 내용들 구현 중, 이번주 안에 마무리하여 수요일날 교수님께 인게임 연동된 게임 보여드릴 수 있도록 노력하는 중

[2] 김동석(클라이언트)

주간 목표 :

- GameScene 마무리
- Object 관리자 만들기(ApplicationContext)

03.22 Shader 관련

- 오류 수정

	저번에 발생했던 에러를 해결하였다. 원인은 루트시그너쳐를 수정하면서 레지스터 값이 일치되지 않았었다. - **Lights.hlsl** 현재는 Direction Light만 사용하고 있지만 나중에 추가하기 쉽도록 spot light, point light 등을 추가했다. **03.24 ApplicationContext** - **오브젝트 관리** CreateProps, DisplayProps 등의 함수를 관리하고 오브젝트를 관리하는 역할을 한다. Map으로부터 오브젝트를 한 번에 불러오고(현재는 Map을 생성하지 않아서 테스트를 위해 몇 개의 오브젝트들을 임의로 위치 등을 지정해두었다) 동일한 메쉬를 가진 오브젝트를 불러올 때 같은 메쉬를 두 번 저장하지 않도록 관리할 수 있는 ItemsMap 배열을 추가하였다. **03.25 ~ 03.27 프레임워크 마무리** - **AssertsReference** Map으로부터 받아온 오브젝트의 FBX 파일로 접근해서 메쉬 정보를 받아오고 Matrial을 설정해준다. 저번에 추가해둔 GeometryGenerator를 이용해 기본적인 도형 등을 메쉬 배열에 추가해 놓는다.(바운딩 박스 등에서 사용할 예정) - **GamePlayScene** SceneManager를 통해 접근하여 위에서 생성해 두었던 오브젝트 정보들을 받아와 배치한다. Diretional Light를 사용하며 플레이어를 생성(현재 애니메이션이 구현이 안되어 있어 플레이어 메쉬는 육면체를 사용하고 있다.)하고 카메라를 연결한다. SceneController를 통해서 플레이어를 컨트롤하도록 하였다. - **프레임워크 마무리** 지금까지 나누어서 설계했던 것들을 서로서로 연결해서 실행되도록 마무리를 했다. 기존에 사용하던 프레임워크랑 3D게임프로그래밍 강의에서 제공한 예제들을 참고하였고 연결 과정에서 예상치 못한 오류들이 발생해서 인터넷 검색과 게임수학에서 교재로 사용하던 교재(게임 프로그래밍을 위한

	3차원 그래픽스)를 참고하여 오류를 수정하였다. 이번 주는 중간중간 기록해놨던 자료가 날라가서 많이 아쉽다(외장하드가 왜인지 모르겠지만 인식이 안된다...) 앞으로는 태블릿에 기록해두는 습관을 길러야겠다(적어두었던 것들이 너무 아깝다).
다음 주 할 일	**[0] 공동** **03.29 주간 회의 (14주차)** **안건 : 일일계획 공유** **교수님 피드백 수정 관련 회의** **[1] 김영준 (서버)** 주간 목표 - 인 게임 서버 구현 **[2] 김동석 (클라이언트)** 주간 목표 - 서버와의 연동 - Map 생성
문제점	**[1] 김영준 (서버)** - 이번 주는 Singleton, 순수가상함수 등 미리 공부했던 내용들을 사용할 수 있는 부분이 많아서 코드를 짜는 내내 즐거웠다, 딱히 막히는 점 없이 이대로 구현하여 수요일 교수님을 뵈러 가기 전에 인게임을 완성하는 것이 목표이다. - FAR* 관련 답변 감사합니다! **[2] 김동석 (클라이언트)** - 프레임워크가 마무리되었다. 생각했던 것보다 훨씬 시간을 많이 투자했는데 그만큼 무언가를 추가할 때 유연하게 대응할 수 있도록 짠 것 같아서 뿌듯하다(물론 앞으로 더 지켜봐야겠지만). - 서버 연동 완료 후 계속해서 스카이박스, 애니메이션 등을 추가해 나갈 예정이다.

[추가]