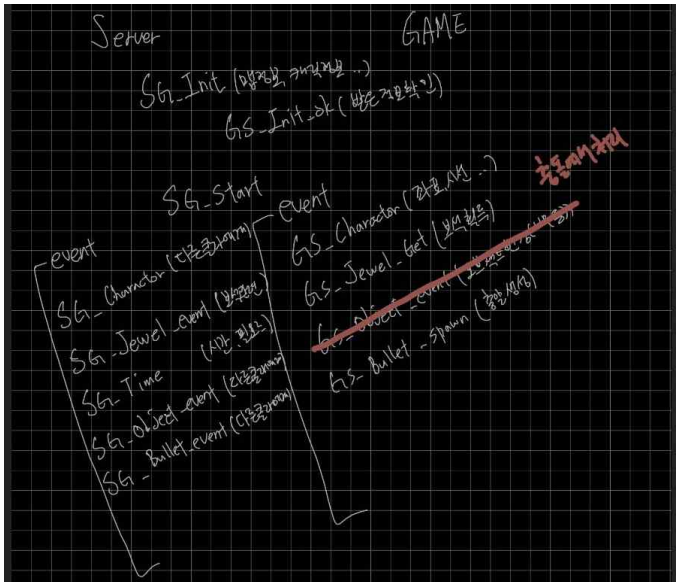


[Toy Ground(토이 그라운드)]

10 주	2020. 2. 28 ~ 2021. 3. 6	작성자	김동석
이번 주 한 일	<p>[0] 공동</p> <p>03.01 주간 회의</p> <ul style="list-style-type: none"> - 일주일간 계획공유 - 인게임 패킷 관련 회의 <p>[1] 김영준(서버)</p> <p>주간목표:</p> <ul style="list-style-type: none"> - 다중 클라이언트 문제 해결 - 게임 서버 프로그래밍 강의 수강 <p>03.01~03 DB공부</p> <p>sql을 통한 데이터 삽입 삭제 변경 등 VS와 연동하는 부분을 주제로 공부했다. (개강하면서 집중이 흐려져 진도를 많이 나가지 못함)</p> <p>03.04 인 게임 패킷 결정</p> <p>인 게임 패킷을 결정하고 패킷을 주고 받는 구조에 대해서 짜기 시작함</p> <p>03.04 인 게임 구현 시작</p> 		

추후 더미클라이언트에 실제로 연동하면서 생기는 문제에 대해서 해결, 추가 하며 클라이언트와 실제로 연동하기 전까지 수정을 거칠 예정

[2] 김동석(클라이언트)

주간 목표 :

- 프레임워크, 렌더러 부분 계속해서 다듬기
- 모델에 텍스처 씹우기
- 맵 오브젝트 배치

03.02 싱글톤 패턴, 템플릿 관련 공부

- 싱글톤 패턴

싱글톤 패턴이란?

→전역 변수를 사용하지 않고 객체를 하나만 생성하도록 하며, 생성된 객체를 어디에서든지 참조할 수 있도록 하는 패턴

생성자가 여러 차례 호출되더라도 실제로 생성되는 객체는 하나고 최초 생성 이후에 호출된 생성자는 최초에 생성한 객체를 반환한다.

싱글톤 패턴을 쓰는 이유

→고정된 메모리 영역을 얻으면서 한 번의 new로 인스턴스를 사용하기 때문에 메모리 낭비를 방지하기 쉽고 싱글톤으로 만들어진 인스턴스는 전역 인스턴스이기 때문에 다른 클래스의 인스턴스들이 데이터를 공유하기 쉽다.

싱글톤 패턴의 문제점

→싱글톤 인스턴스가 너무 많은 일을 하거나 많은 데이터를 공유시킬 경우 다른 클래스의 인스턴스들 간에 결합도가 높아져 “개방-폐쇄 원칙”을 위배하게 된다(=객체 지향 설계 원칙에 어긋남).

멀티쓰레드 환경에서 동기화 처리를 안하면 인스턴스가 두 개가 생성되는 경우가 발생할 수 있다.

참고: <https://jeong-pro.tistory.com/86>

싱글톤 패턴 사용(class 생성)

1. new가 한 번만 호출되도록 생성자가 외부에서 호출되지 않도록 생성자를 private로 선언
2. 자기 자신에 대한 인스턴스를 하나 만들어 외부에 제공
3. new가 호출되기 전이면 인스턴스 메서드는 호출할 수 없다.

참고:

<https://gmlwjd9405.github.io/2018/07/06/singleton-pattern.html>

- 템플릿

템플릿이란?

→ 함수나 클래스를 개별적으로 다시 작성하지 않아도, 여러 자료형으로 사용할 수 있도록 만들어 놓은 틀

함수 템플릿

→ 함수를 만들어 낼 때, 함수의 기능은 명확하지만 자료형은 모호하게 두는 것.

예시 1) 템플릿 없이 오버로딩 사용

```
1 int sum(int a, int b){
2     return a + b;
3 }
4 double sum(double a, double b){
5     return a + b;
6 }
```

예시 2) 템플릿 사용

```
1 template <typename T>
2 T sum(T a, T b){
3     return a + b;
4 }
```

클래스 템플릿

→ 클래스 내부의 멤버 변수의 타입에 대해서 템플릿으로 선언할 때 사용, 선언시 유의할 점은 멤버 함수(변수)를 클래스 외부에서 선언할 때 템플릿 선언을 다시 해주어야 한다.

- 클래스에 적용할 싱글톤 패턴 생성

생성자를 portected로 생성

자기 자신에 대한 인스턴스를 외부로 제공하는 함수 GetApp()

멤버변수인 m_pApp을 클래스 외부에서 선언할 때 템플릿 선언을 한번 더 해줌

이를 통해서 클래스를 관리할 예정

```

1  #pragma once
2  #include "pch.h"
3
4  template <typename Type>
5  class TemplateSingleton
6  {
7  protected:
8      TemplateSingleton() {};
9      virtual ~TemplateSingleton() {};
10
11 private:
12     static Type* m_pApp;
13
14 public:
15     static Type* GetApp()
16     {
17         if (m_pApp == nullptr)
18             m_pApp = new Type;
19         return m_pApp;
20     };
21
22     static void DestroyApp()
23     {
24         SAFE_DELETE_PTR(m_pApp);
25     };
26 };
27
28 template <typename Type> Type* TemplateSingleton<Type>::m_pApp = nullptr;

```

[Singleton.h]

참고: <https://blockdmask.tistory.com/43>
<https://blockdmask.tistory.com/45>

03.03 싱글톤 패턴 적용 프레임워크 수정

- 관련 공부 내용

try...catch 문법

1. 먼저 try { ... }안의 코드가 실행된다.
2. 에러가 없다면 try안의 마지막 줄까지 실행되고, catch 블록은 건너 뜀다.
3. 에러가 있다면, try안 코드의 실행이 중단되고, catch(err) 블록으로 제어흐름이 넘어간다.

참고: <https://ko.javascript.info/try-catch>

DXException - d3Util.h

DirectX 프로그램 작성에 있어서 발생하는 에러를 새로운 Window에 출력하는 역할

참고: <https://redchicken.tistory.com/280>

Microsoft::WRL::ComPtr

COM 객체를 위한 스마트 포인터이다. 즉, 범위를 벗어난 ComPtr 인스턴스는 해당 COM 객체를 자동으로 Release 시켜준다.

참고:

<http://blog.naver.com/PostView.nhn?blogId=websearch&logNo=221166428102&parentCategoryNo=&categoryNo=137&viewDate=&isShufflePopularPosts=false&from=postView>

<https://docs.microsoft.com/ko-kr/cpp/cppcx/wrl/how-to-create-a-classic-com-component-using-wrl?view=msvc-160>

스마트 포인터(unique_ptr)

자원을 확보하는 방법

1. new로 확보한 자원을 포인터로 저장
2. 포인터를 이용하여 자원에 access
3. 사용이 끝나면 delete로 반환

위의 방법은 실수로 자원을 두 번 반환하거나 자원을 반환하지 않을 때 사용 가능한 메모리가 줄어들고 비정상적인 종료의 위험이 있을 수 있다.

→이를 해결하는 방법이 스마트 포인터(unique_ptr), 프로그래머가 언제 자원을 해제할지 고민하지 않아도 된다. unique_ptr은 자신이 소멸될 때 저장한 자원의 소멸자를 호출하기 때문

Ex) int* p = new int;

unique_ptr<int> p { new int };

→주의할 점

unique_ptr은 자원을 독점하기 때문에 p를 복사 할 수 없다. 따라서 이동 문법을 사용하여 소유권을 넘겨야 한다.

Ex) void f(unique_ptr<Dog>);

f(move(p));

참고: 윤정현 교수님 STL 강의 자료,

https://en.cppreference.com/w/cpp/memory/unique_ptr/make_unique

연관 컨테이너(map)

map은 pair<key, value>를 원소로 저장하며 key를 기준으로 정렬하여 저장한다.

연관 컨테이너는 찾고자 하는 원소를 빨리 찾기 위해 사용한다.

시간복잡도: $O(\log_2 n)$

map은 어떤 자료형이라도 key로 사용할 수 있다.

	<p>Ex) map[key] = value; Light["Direction"]->Position = (0.f, 1.f, 0.f);</p> <pre>std::map<std::string, std::unique_ptr<Light>> m_Lights;</pre> <p>참고: 윤정현 교수님 STL 강의 자료, https://blockdmask.tistory.com/87</p> <p>03.04 ~ 03.06 프레임워크 내 렌더러 부분 분리(진행 중)</p> <p>- 프레임워크 분석</p> <p>현재 프레임워크 내에서 mesh, object, shader 등이 분리돼 있어 오브젝트를 하나 그리려면 여러 단계를 거쳐야 함 →이는 그래픽 파이프라인 구조를 공부하기에는 좋지만 오브젝트의 추가 및 제거가 어려움 →따라서 렌더러 부분을 분리하고 싱글톤 패턴을 적용하여 프레임워크를 다듬을 필요가 있음.</p> <p>- 렌더러 부분 분리 계획</p> <p>현재 프레임워크를 SceneManager, InputHandler, AssertsRefernce, ApplicationContext, GraphicsRenderer, GameCore 등으로 분리</p> <p>- 프레임워크, 렌더러 부분 계속해서 다듬기(50%)</p> <p>계속해서 진행 중, 3월 중순 안에 마무리해서 오브젝트를 배치하는 것이 목표이다.</p> <p>현재의 프레임워크가 싱글톤 패턴이 적용이 안돼있어 Class끼리 서로 맞물려 있는 경우가 많아서 분리하는데 어려움이 있다. →해결방법</p> <p>태블릿을 사용하여 전체적인 흐름을 정리하여 시간이 오래 걸리더라도 실수를 최대한 줄이려고 노력하고 있다.</p>
<p>다음 주</p> <p>할 일</p>	<p>[0] 공동</p> <p>03.08 주간 회의 (11주차)</p> <p>안건 : 일일계획 공유</p> <p>[1] 김영준 (서버)</p> <p>주간 목표</p> <p>- 인 게임 서버 구현</p> <p>[2] 김동석 (클라이언트)</p>

	<p>주간 목표</p> <ul style="list-style-type: none"> - 프레임워크 렌더러 부분 계속해서 다듬기 - 모델에 텍스처 씹우기
문제점	<p>[1] 김영준 (서버)</p> <ul style="list-style-type: none"> - 개강 이후 밀린 약속을 챙기면서 이번 주 시간을 많이 놓쳤다, 교수님께 졸업작품 보여드리기 전에 완성할 내용을 동석이와 회의 이후 확실히 정하고 해당 내용으로 동기부여를 받아 일정에 맞출 수 있도록 노력해야겠다. - 로비 씽 보다 클라이언트와 연동하여 보여드릴 수 있는 인 게임 연동을 먼저 하기로 했다. DB 공부하던 것을 중단하고, 주간회의를 통해 인 게임 패킷을 정하고 미리 더미 클라이언트를 통해 연동해야겠다. - 게임엔진 수업을 혼자 듣게 되어 무작위 팀원과 작업을 해야 해서 걱정이다, 유니티는 전에 다뤄봐서 걱정 없지만 모르는 사람과 해야 한다는 점이 걱정된다. <p>[2] 김동석 (클라이언트)</p> <ul style="list-style-type: none"> - 개강을 했다. 다행히 비대면 강의를 하는 과목들이 많아서 3월 한 달 동안은 월,화,수에 학교를 안가서 비교적 졸업작품을 진행하는데 시간을 계속 투자할 수 있게 돼 다행이다. - 현재 프레임워크를 다듬으면서 관련 내용들에 대해서 계속 공부 중인데 생각보다 시간이 오래 걸려서 걱정이다. 개발일정에서 계획한대로 최대한 3월달 안에 최대한 빨리 완성할 수 있도록 노력해야겠다.

[추가]