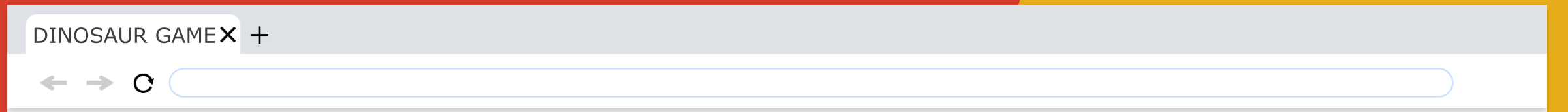


C++ PROJECT

DINOSAUR GAME

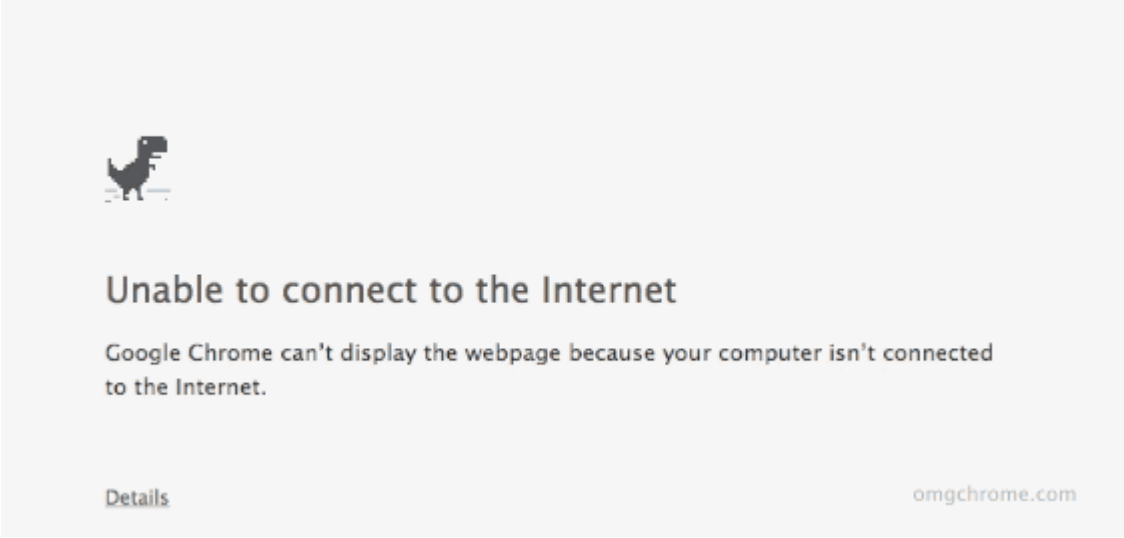
201704034 민문기

DINOSAUR GAME ✕ +

https://C++ PROJECT/주제

주제: 공룡게임

공룡 달리기, 또는 **공룡 게임**(Dinosaur Game)은 크롬에서 네트워크 연결이 끊겼을 때 나오는 공룡을 클릭하거나 스페이스바를 누르면 간단한 미니게임을 할 수 있는 이스터 에그가 있다.



강의 때 배운 것과 OOP를 중점으로 공룡 게임을 최대한 비슷하게 구현

DINOSAUR GAME ✕ +

https://C++ PROJECT/게임 이해

게임 이해하기

인터넷 없음 오류 화면에서 스페이스 바를 누르거나 터치를 통해 게임을 시작할 수 있다.

공룡이 앞으로 달려가며 앞에서 다가오는 장애물을 피하기 위해 점프

장애물을 피해 최대한 오래 피하는 킬링타임 게임.

DINOSAUR GAME

https://C++ PROJECT/구현할 기능

구현할 기능

1. 움직이는 공룡
(움직이는 모션, 점프)

2. 다가오는 장애물
(장애물 생성 및 이동)

3. 충돌판정

4. 게임 기능

DINOSAUR GAME

https://C++ PROJECT/기능 나누기

기능 나누기

-게임 관리
    -게임이 작동하는 기능을 담당
-엔티티
    -공룡과 장애물을 담당
-키입력
    -키보드 입력을 담당
-커서위치 설정
    -콘솔 창의 커서위치를 관리
-점수판
    -게임 중,게임 후에 점수를 알려줌

DINOSAUR GAME

https://C++ PROJECT/엔티티 구현

엔티티 구현

1. 움직이는 공룡
(움직이는 모션, 점프)

2. 다가오는 장애물
(장애물 생성 및 이동)

DINOSAUR GAME ✕ +

https://C++ PROJECT/코드/Setting.h

Setting

```
const int KEY_ESC = 27;          // ESC 키
const int KEY_SPACE = 32;        // SpaceBar 키

const int MAX_JUMP = 6;          // 최대 점프 높이
const int Y_BASE = 20 ;          // 공룡의 초기 Y축 위치
const int Y_COLLISION = 4;       // Y축의 충돌 기준 위치
const int TREE_COLLISION = 7;    // 나무가 공룡과 충돌 가
const int TREE_START = 50;       // 나무가 생성되는 위치
const int TREE_END = -6;         // 나무가 사라지는 위치
const int SLEEP_TIME = 35;       // 게임 갱신 주기
const int TREE_BOTTOM_Y = 20;    //나무의 초기 Y축 위치
```

게임 구현에 필요한 상수들을 한곳에 모아서선언

DINOSAUR GAME ✕ +

https://C++ PROJECT/코드/Entity.h

Entity

```
class Entity
{
protected:
    int _X = 0;
    int _Y=0;
public:
    void setX(int X);
    void setY(int Y);
    int X() { return _X; }
    int Y() { return _Y; }
    virtual void drawEntity();
};
```

공룡과 장애물을 생성할 때 쓰일 기본적인 틀

DINOSAUR GAME ✕ +

https://C++ PROJECT/코드/DinoEntity.h

DinoEntity

```
class DinoEntity : public Entity
{
public:
    bool _leg=true;
public:
    DinoEntity();
    void drawEntity();
};
```

Entity 클래스를 상속 받아 DinoEntity 클래스를 생성

DINOSAUR GAME

https://C++ PROJECT/코드/DinoEntity.cpp

DinoEntity

```cpp
DinoEntity::DinoEntity() {
    _X = 0;
    _Y = 0;
}

void DinoEntity::drawEntity()
{   //대충 모양만 나중에 수정
    Game_manager* gm = new Game_manager;
    gm->gotoXY(0, Y_BASE - _Y); // 공룡 그리기, 커서 위치 변경
    std::cout << "        ■■■■ ₩n";
    std::cout << "       ■ ■■■₩n";
    std::cout << "      ■■■■■₩n";
    std::cout << "■     ■■      ₩n";
    std::cout << "■    ■■■■  ₩n";
    std::cout << "■■   ■■■    ₩n";
    std::cout << " ■■■■■■■■ ₩n";
    std::cout << " ■■■■■■   ₩n";
    std::cout << "  ■■■■■   ₩n";
    std::cout << "    ■■■■   ₩n";
    std::cout << "     ■■■    ₩n";

    if (_leg) {
        std::cout << "  ■   ■■₩n";
        std::cout << "  ■■   ₩n";
        _leg = false;
    }
    else {
        std::cout << "  ■■ ■ ₩n";
        std::cout << "       ■■ ₩n";

        _leg = true;
    }
    delete gm;
}
```

이전에 엔티티 클래스의 가상함수 drawEntity()를 재정의 함

공룡을 그리고 bool을 이용해서 다리가 움직이도록 보이게 함

DINOSAUR GAME ✕ +

https://C++ PROJECT/코드/TreeEntity.h

TreeEntity

```
class TreeEntity :public Entity
{
protected:
    int _tree=1;
public:
    TreeEntity();
    void setTree();
    void drawEntity();
};
```

Entity클래스를 상속받은 TreeEntity 클래스를 생성

DINOSAUR GAME ✕ +

https://C++ PROJECT/코드/TreeEntity.cpp

TreeEntity

```cpp
TreeEntity::TreeEntity(){
    _X = 0;
    _Y = 0;
}

void TreeEntity::setTree() {
    srand((unsigned int)time(NULL));
    _tree = rand()%3;
}

void TreeEntity::drawEntity()
{
    Game_manager* gm = new Game_manager;
    if (_tree==0) {
        gm->gotoXY(_X, TREE_BOTTOM_Y);  //나무 그리기, 커서 위치 변경
        std::cout << "    ■₩n";
        gm->gotoXY(_X, TREE_BOTTOM_Y + 1);//한 칸씩 아래로
        std::cout << "  ■■■₩n";
        gm->gotoXY(_X, TREE_BOTTOM_Y + 2);
        std::cout << " ■■■■₩n";
        gm->gotoXY(_X, TREE_BOTTOM_Y + 3);
        std::cout << "    ■₩n";
        delete gm;
    }
    else if(_tree==1) {
        gm->gotoXY(_X, TREE_BOTTOM_Y);  //나무 그리기, 커서 위치 변경
        std::cout << "  ■■■₩n";
        gm->gotoXY(_X, TREE_BOTTOM_Y + 1);//한 칸씩 아래로
        std::cout << "  ■■■₩n";
        gm->gotoXY(_X, TREE_BOTTOM_Y + 2);
        std::cout << "    ■₩n";
        gm->gotoXY(_X, TREE_BOTTOM_Y + 3);
        std::cout << "    ■₩n";
        delete gm;
    }
    else {
        gm->gotoXY(_X, TREE_BOTTOM_Y);  //나무 그리기, 커서 위치 변경
        std::cout << "      ■₩n";
        gm->gotoXY(_X, TREE_BOTTOM_Y + 1);//한 칸씩 아래로
        std::cout << "  ■   ■₩n";
        gm->gotoXY(_X, TREE_BOTTOM_Y + 2);
        std::cout << "  ■■■₩n";
        gm->gotoXY(_X, TREE_BOTTOM_Y + 3);
        std::cout << "     ■₩n";
        delete gm;
    }
}
```

setTree()

rand함수로 난수를 생성

drawEntity()

이전에 생성한 난수에 따른 나무를 생성

DINOSAUR GAME ✕ +

https://C++ PROJECT/게임 기능 구현

게임 기능 구현

3. 충돌판정

4. 게임 기능

DINOSAUR GAME

https://C++ PROJECT/코드/Game_manager.h

Game_manager

```
class Game_manager
{
public:
    int _Score = 0;             // 점수
    int _CurKey = -1;           // 현재 눌린 키
    bool _IsJumpping = false;   // 현재 공룡이 점프 중인가
    bool _maxHeight = false;    // 현재 공룡이 점프하여 최고 지점에 올랐는가
    bool _Collision = false;    // 현재 공룡이 나무와 충돌하였는가
public:
    void Init();
    int GetKeyDown();
    void CursorSetting();
    void gotoXY(int X, int Y);
    void drawScore(int score);
    void drawEnd(int score);
    void GameStart();
};
```

콘솔창의 초기 설정
키 입력
커서 초기 셋팅
커서 좌표 이동
점수판
게임 종료
게임 시작

DINOSAUR GAME

https://C++ PROJECT/코드/Game_manager.cpp

Game_manager

```cpp
void Game_manager::Init() {
    system("mode con:cols=100 lines=25");
    system("title Dino");
}

int Game_manager::GetKeyDown() {
    if (_kbhit() != 0)
        return _getch();
    return 0;
}

void Game_manager::CursorSetting()
{
    CONSOLE_CURSOR_INFO cursorInfo = { 0, };
    cursorInfo.dwSize = 1;
    cursorInfo.bVisible = FALSE;
    SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cursorInfo);
}
```

Init()

콘솔 창의 크기를 조정 25줄 100칸

GetKeyDown()

키보드의 입력이 있는지 없는지를 판단
_kbhit() 키보드가 눌렸다면 0이외의 값을 리턴
_getch()입력받은 아스키코드를 정수형으로 리턴

CursorSetting()

CONSOLE_CURSOR_INFO라는 구조체를 선언
콘솔의 커서가 보이지 않도록 변경

DINOSAUR GAME ✕ +

https://C++ PROJECT/코드/Game_manager.cpp

Game_manager

```cpp
void Game_manager::gotoXY(int X,int Y) {
    COORD cursorPos = { (SHORT)X, (SHORT)Y };
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), cursorPos);
}

void Game_manager::drawScore(int score) {
    gotoXY(0, 0);
    std::cout << "Score: " << score << "Wn";
}

void Game_manager::drawEnd(int score) {
    std::cout << "WnWnWnWnWn";
    std::cout << "WtWtWtWtWtGame OverWn";
    std::cout << "WtWtWtWtWtScore: " << _Score << "WnWnWnWnWnWn";
    system("pause");
}
```

gotoXY(int x,int y)
콘솔의 커서를 해당 좌표로 이동시키기

drawScore(int score)
현재 점수를 표시

DrawEnd(int score)
게임이 끝난후 최종 점수를 표시

DINOSAUR GAME ✕ +

https://C++ PROJECT/코드/Game_manager.cpp

Game_manager

```
void Game_manager::GameStart() {
    DinoEntity* dino = new DinoEntity;
    TreeEntity* tree = new TreeEntity;

    while (true)
    {
        // 키 입력 확인
        _CurKey = GetKeyDown();
        switch (_CurKey)
        {
        case KEY_ESC:       // ESC 키
            return;
        case KEY_SPACE:     // SPACE BAR 키
            _IsJumpping = true;
            break;
        default:
            break;
        }
```

키보드 입력을 받습니다

입력은 ESC, SPACEBAR 두 가지

DINOSAUR GAME ✕ +

https://C++ PROJECT/코드/Game_manager.cpp

Game_manager

```
// 점프 중인가
if (_IsJumpping)
{
    // 최고 지점에 이르지 않았다면
    if (Dinoy < MAX_JUMP && _maxHeight == false)
        Dinoy++;
    // 최고 지점에 도달 후 점프가 끝났다면
    else if (_maxHeight && Dinoy == 0)
    {
        _maxHeight = false;
        _IsJumpping = false;
    }
    // 최고 지점에 도달 후라면 (중력을 표현)
    else if (_maxHeight)
        Dinoy--;
    // 최고 지점에 도달했다면
    else if (Dinoy == MAX_JUMP)
        _maxHeight = true;
}
// 점프 중이 아니라면
else
{
    if (Dinoy > 0)
        Dinoy--;
}
```

점프

점프를 한다면 우선 만약 공룡의 Y좌표가 최대 지점에 도달하지 않았다면 Y좌표를 상승시킵니다.

점프가 끝이 났다면
최대높이와 점프 상태를 모두 False로 만들어 점프를 할수 있는 상태로 변경 합니다.

최고 지점에 도달 후라면 공룡의 Y좌표를 내립니다.

공룡의 좌표가 최고 지점에 다다랐다면 최고 높이를 true로 변경합니다

DINOSAUR GAME ✕ +

https://C++ PROJECT/코드/Game_manager.cpp

Game_manager

```
// 나무 위치 관련
treeX -= 2;
if (treeX <= -2 ) {
    treeX = TREE_START;
    tree->setTree();
}
//충돌 관련
//나무의 X위치가 충돌 가능 X위치라면
if (treeX < TREE_COLLISION)
{
    // 공룡의 Y위치가 충돌 가능 위치이고
    // 나무의 X위치가 충돌 가능 위치라면
    if (Dinoy  < Y_COLLISION && treeX > TREE_END + 1)
        _Collision = true;
}
```

나무 엔티티 조작

나무의 X좌표를 계속 줄임
X좌표가 -2이하가 되면 X좌표를 초기화
나무의 종류를 선택

충동 판정
나무의 X좌표 위치가 충돌 가능 위치이고,
공룡의 Y좌표 위치가 충돌 가능 위치이면 충돌 판정

DINOSAUR GAME ✕ +

https://C++ PROJECT/코드/Game_manager.cpp

Game_manager

```
//엔티티 그리기
tree->setX(treeX);
tree->drawEntity();
dino->drawEntity();

Sleep(SLEEP_TIME);
system("cls");

// 충돌 시 게임 오버
if (_Collision)
{
    // 점수 출력
    drawEnd(_Score);
    return ;
}
// 충돌 상태가 아닐 때는 점수 증가
else
{
    _Score += 1;
    drawScore(_Score);
}
```

나무 엔티티의 좌표를 설정
나무의 엔티티를 그리기
공룡엔티티를 그리기

콘솔창을 sleep_time초 후 새로 고침

충돌 했다면 게임 종료 그렇지 않으면 스코어를 올리고 계속 진행

DINOSAUR GAME ×

https://C++ PROJECT/코드/Main.cpp/

Main

```cpp
#include<iostream>
#include<Windows.h>
#include<conio.h>
#include"Entity.h"
#include"Game_manager.h"
#include"setting.h"
using namespace std;

int main(void) {
    Game_manager gm;
    gm.Init();
    gm.CursorSetting();
    gm.GameStart();
}
```

Game_manager객체를 생성

콘솔창 초기화 및 커서 설정
게임 시작

내용 슬라이드

https://C++ PROJECT/최종_결과물/

최종 결과물

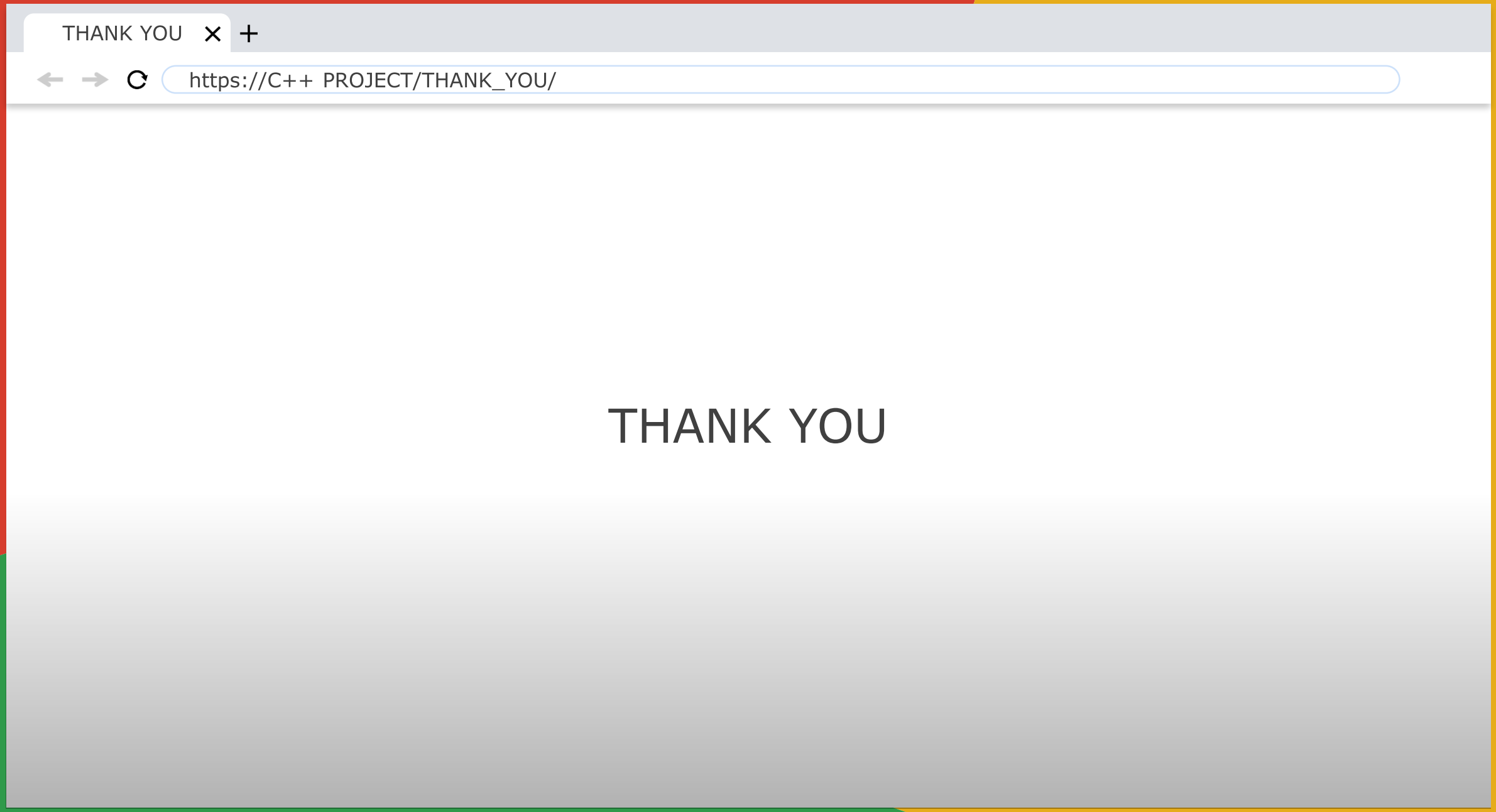
내용 슬라이드

https://C++ PROJECT/문제점

문제점

강의 때 배운 것과 OPP를 중점으로 개발하려 했지만 오히려 여러 기능을 억지로 끼워 맞춘 느낌이 들었다.

코드가 깔끔하지 못했고 앞에서 기획한 대로 구현하지 못한 것 같다.


THANK YOU
https://C++ PROJECT/THANK_YOU/
THANK YOU