개요 | 제출 | 편집 | 코딩 결과

실습 104 - prototype

제출 마감일: 2023-06-09 23:59
업로드 가능한 파일 수: 2
제출 방식: 개인

설명

새로운 객체를 생성하는 비용이 비싸고 자원이 많이 소요된다면 프로토타입 패턴의 사용을 고려해 볼 수 있습니다.

예를 들어, 게임의 캐릭터나 몬스터 객체를 생성할 때 서버에 접속해서 관련 정보 (캐릭터 타입, 능력치, 아이템 등) 를 읽은 후, 필요한 데이터를 모두 설정하여 객체를 생성하는 과정은 비용이 클 수 있습니다. 또한, 특정 순간에 대량의 몬스터 객체를 생성해야 하는 경우 역시도 일반적인 방법으로는 생성 비용이 많이 듭니다.

이런 경우에 먼저 캐릭터 객체나 몬스터 객체를 처음으로 생성한 후, 다음부터는 이를 복제하여 새로운 객체를 생성하는 방법을 시도해 볼 수 있습니다.

다음은 clone 메서드를 이용하여 기존 객체를 복제하여 새로운 객체를 생성하는 코드입니다.

```
class ExpensiveObject : public Prototype {
    // Assume that the constructor makes a costly database query
public:
    ExpensiveObject() {
        // Expensive database query...
    }
    std::unique_ptr<Prototype> clone() const override {
        return std::make_unique<ExpensiveObject>(*this);
    }
};

// Instead of creating a new object...
ExpensiveObject expensive;
//ExpensiveObject expensive2;
```

```
// ...we can clone an existing one.
auto cheapCopy = expensive.clone();
```

또한, 유사한 객체를 자주 만들고 삭제하는 경우에도 매번 새 개체를 만드는 것보다 기존 객체를 복제하는 것이 더 효율적일 수 있습니다.

```
ExpensiveObject expensiveObject;
for (int i = 0; i < 1000; i++) {
    auto object = expensiveObject.clone();
    // Use the object...
}
```

문제

You're tasked with simulating a simple game environment where various types of monsters can spawn. The details of these monsters are stored in external files and are quite complex to load, making the creation of each monster an expensive operation in terms of time and computational resources.

Your task is to design and implement a C++ program which creates these monsters using the clone() method, significantly reducing the cost of creating each new monster.

<참고자료>

```
//MonsterTest.cpp

// class Monster
// class Dragon : public Monster
// class Goblin : public Monster

std::map<std::string, std::unique_ptr<Monster>> monsterRegistry;

void loadMonsters() {
    // Read monster information from file (this is the costly operation)
    // Here, we just pretend to do it and create a Dragon and a Goblin
```

```
    // Here, we just pretend to do it and create a Dragon and a Goblin

    monsterRegistry["Dragon"] = std::make_unique<Dragon>();
    monsterRegistry["Goblin"] = std::make_unique<Goblin>();
}

std::unique_ptr<Monster> spawnMonster(const std::string& type) {
    // implement your code
}

int main() {
    loadMonsters(); // load monster prototypes

    std::unique_ptr<Monster> dragon1 = spawnMonster("Dragon"); // clone from prototype
    dragon1->roar();

    std::unique_ptr<Monster> goblin1 = spawnMonster("Goblin"); // clone from prototype
    goblin1->roar();

    return 0;
}
```

입력

없음

출력

Dragon roars!

Goblin roars!

제출파일

MonsterTest.cpp
104.csv

VPL