구현

자율 과제

- 4월 16일 수요일까지
- SVN 에 프로젝트 파일과 문서(pdf) 올리기
- 문서 내용 :
  - 클래스 간의 상속 구조와 포함 관계에 대해 이해한 바를 자유로운 형식으로 정리
  - 새로 알게 된 C++ 문법 정리, 잘 이해가 안가는 부분이 있다면 구체적으로 적어 보기.
- 프로젝트 기능 구현:
  - 마우스 오른쪽 버튼을 클릭하면 적이 생성 됩니다. 이 때, 겹치지 않게 해주세요.
  - 플레이어 이동 중에 적 오브젝트와 겹치는 경우, 둘 다 파란 색으로 표시

---

studylog

```
m_GameObjectPtrTable에 하는 접근 하는 게 목표였다.
```

```
#250416
[Review]
bool  learning::Intersect(ColliderCircle const& lhs, ColliderCircle const&
rhs)
{
    return (rhs.center - lhs.center).LengthSquared() <= pow(lhs.radius +
rhs.radius, 2);
}

여깄다 써야 될 함수

"ColliderCircle " 이거 2개


m_EnemySpawnPos = { 0, 0 };

여깄다  lhs
```

```
아니네;;;

        //---
        ColliderCircle* circle = dynamic_cast<ColliderCircle*>
(m_GameObjectPtrTable[i]);
        Intersect(circle, circle);
        //---

왜 아닐까?
어제 캡슐 찔러 볼 떄도 그렇고, 오늘 읽어봐도 여기 밖에 넣을 때가 없어.
근데 안돼? 맞아. 근데 그 이유는 캡슐의 위치가 아니라 cpp를 몰라서야.


[Do]
과제 두개 모두 완료 ?
1. flag에 따라서 구현...
2.  circleCollider로 받아옴
```

코드 분석 및 정리

CreateEnemy() 함수

이 함수는 적(Enemy) 객체를 생성하는 역할, m_GameObjectPtrTable에는 0번째 원소는 플레이어, 1번째 원소 부터가 적이다.

1. **적 객체 생성 및 기본 설정**
   - 새 GameObject를 ObjectType::ENEMY로 생성 <- 이거 신기하다.
   - 이름을 "Enemy"로 설정
   - 스폰 위치, 속도, 충돌체(Collider) 설정
2. **충돌 검사 로직 (사용자 추가 부분)**
   - 현재 생성할 적의 CircleCollider 정보 설정
   - 기존 객체들과 겹치는지 확인하는 로직 구현
   - Intersect() 함수를 사용하여 원형 충돌체 간 겹침 검사
   - 겹치는 객체가 있으면 isValiable 플래그를 false로 설정
3. **객체 배치 또는 삭제**
   - 겹치지 않는 경우(isValiable이 true), 객체 테이블에 추가
   - 테이블이 가득 찼거나 겹침이, 발생한 경우 생성한 객체 삭제

UpdatePlayerInfo() 함수

이 함수는 플레이어의 정보를 업데이트하고 **충돌을 감지한다**:

1. **플레이어 객체 및 위치 참조**
   - 정적 플레이어 객체 참조 및 NULL 검사
   - 마우스 위치와 플레이어 위치 정보 획득
2. **충돌 검사 로직 (사용자 추가 부분)**
   - 플레이어의 ColliderCircle 설정
   - 모든 게임 객체를 순회하며 플레이어와의 충돌 검사
   - 충돌 발생 시 플레이어와 해당 객체의 IsOverlap 플래그를 true로 설정
   - 충돌이 없을 경우 IsOverlap 플래그를 false로 설정

GameObjectBase 클래스

이 클래스에서 bool isOverlap을 만들고 그것을 설정할 void SetIsOverlap(bool value)을 만들었다. 이제 이 bool값을 통해 어떤 색으로 랜더할지 결정한다.

구현된 요구사항

1. **적 생성 시 겹치지 않게 처리**
   - CreateEnemy() 함수에서 새로 생성되는 적이 기존 객체들과 겹치지 않도록 검사
   - 겹치는 경우 객체를 생성하지 않음
2. **플레이어와 적 충돌 시 색상 변경**
   - UpdatePlayerInfo() 함수에서 플레이어와 적의 충돌 감지
   - 충돌 시 IsOverlap 플래그를 설정하여 파란색으로 표시될 수 있도록 함

이 코드는 Intersect() 함수를 사용하여 두 원형 충돌체 간의 겹침을 감지하고, 적절한 객체 관리를 통해 게임의 기본적인 충돌 시스템을 구현했다.

주저리 주저리

코드를 읽다
대학 시절, 공부를 안 했기 때문에, 나쁜 습관이 많이 들어 있는 상태였다. 여러 이유 중 하나 gpt의 등장이다. 이로 코딩을 안하고 그것에 의존하는 습관이었다. 하지만 문제는 코드를 읽을 줄 모르는 바보가 되어 버린다는 점이다.
물론 콘솔 프로젝트를 거치며 영상이형의 철학, "gpt는 좋은 검색엔진이다"를 받아 들여 답을 받을 수는 없음을 깨달았다. 이제는 의존을 많이 줄였다.
하지만 아직 긴 코드를 읽는 것은 어렵다. 그게 바로 이 과제인 것 같다. 4월 15일은 'update 함수 내부를 모두 콕 콕 찔러 보며 어디가 맞을까... ' 하며 보냈다. 16일에 비로소 코드를 읽었다. 읽는다. 코드의 구조가 보인다? 절차 지향적으로 따라가 보니 읽을 수 있다에 가깝겠다. 객체 지향은 아직 보이지 않는

다.
아쉽다.