|  |  |  |
| --- | --- | --- |
| 이름 | 백성빈 | 손주은 |
| 학번 | 20170680 | 20170702 |
| 학과 | 컴퓨터공학과 | 컴퓨터공학과 |
| ID | sungbin0515 | jeson |

CSED 451 Computer Graphics—Assn2

**Assn 2: 피카츄 배구**

ASSN 1에서 만들었던 2D 피카츄 배구 게임을 개선하는 것을 목표로 한다.

# 개발 환경:

**IDE:** Visual Studio 2017 community

**Compile mode (VS 2017):** Release x86

**Dependency**: OpenGL / GLEW 2.1.0 / Freeglut 3.0.0

# 요구 사항

## Ball

* 공 가장자리 안쪽을 따라 전기가 흐르는 것을 표현한다. (전기는 공의 내부에서 이동한다.)
* 아래 예시를 따르거나 다른 2레벨 이상의 계층 구조를 사용한다. (선을 이용하여 그린다.)
* 전류가 자연스럽게 흐르는 것처럼 표현하지 않아도 된다.

## Cloud

* 좌우로 이동하며 모양이 변화하는 구름을 만든다.
* 2개 이상의 구름을 만든다.
* (예시) 큰 적색 원을 기준으로 실선 하늘색 원들은 고정되어 있으며, 작은 적색 원을 기준으로 점선 하늘색 원들이 주어진 각도 내에서 회전할 수 있다.
* 예시를 따르지 않아도 좋지만, 계층구조를 가지도록 구현해야 한다.

## Character

* 귀와 꼬리를 표현한다. (꼬리는 2단계 이상일 것)
* 좌우로 움직일 땐 귀와 꼬리가 고정되어 있는다.
* 공과 충돌 시 귀와 꼬리 각 축에 왕복 회전을 주어 떨림을 표현한다.
* 꼬리와 귀는 플레이어의 위치(왼쪽)에만 구현해도 무방하다.
* 충돌 검사는 과제 1과 마찬가지로 꼬리와 귀를 무시한 경계 박스를 이용해 구현한다.

## Restart

* 게임을 재시작하는 기능 추가

# **프로그램 설계**

Assignment 1번를 기준으로 새로 추가된 기능 및 구조에 대해 작성하였다.

1. **Anchor ~~**
2. **ChildList ~~**
3. **Unit shape class 추가**

구름이나 꼬리와 같이 draw할 때 하나 이상의 도형이 필요한 경우에 반복적인 일을 최소화할 수 있도록 UnitTri, UnitCircle 등의 unit 도형 class를 추가하였다.

* 1. UnitTri: position과 size에 부합하는 직각삼각형을 그리는 클래스이다.
  2. UnitCircle: position과 radius에 따라 원을 그리는 클래스이다.

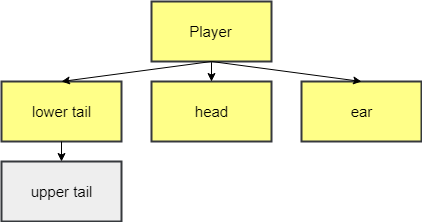
1. **공의 전기**
   1. 공은 기존의 움직임을 유지하고 그와 별개로 원의 중심을 기준으로 회전하는 전기가 있어야 한다.
2. **구름**
   1. 각 구름은 전체적으로 좌우로 움직여야 하고 object 내에서도 날개의 퍼덕거림 같은 움직임이 있어야 한다.
   2. Assn 요구조건에서 첨부된 예시의 구름과 최대한 비슷하게 표현했다. Cloud class내에서는 정삼각형의 세 꼭지점을 중심으로 하는 원의 위치는 상대적으로 고정되어 있고 나머지 양 옆의 ‘날개’ 구름들은 time frame마다 직선 경로를 따로 왕복 translation을 하게 된다.
3. **꼬리와 귀**
   1. 꼬리와 귀 모두 한 개 또는 두개의 UnitTri instance를 이용해 나타낼 수 있다. UnitTri 는 기준점을 중심으로 하는 직각삼각형을 그리는데 공과 충돌 시 이 기준점을 중심으로 rotate 해야한다.
   2. 꼬리를 2단 계층 구조로 나타내기 위해 아래와 같이 lower tail(노란색)을 player의 child list에 추가하고, upper tail(검정색)을 lower tail의 child list에 추가했다.
4. **재시작 기능**

Figure Player 의 계층 구조

‘R’ 또는 ‘r’을 누르면 게임 점수와 공의 위치 등이 초기화됨으로써 게임이 다시 시작된다.

# **시연**

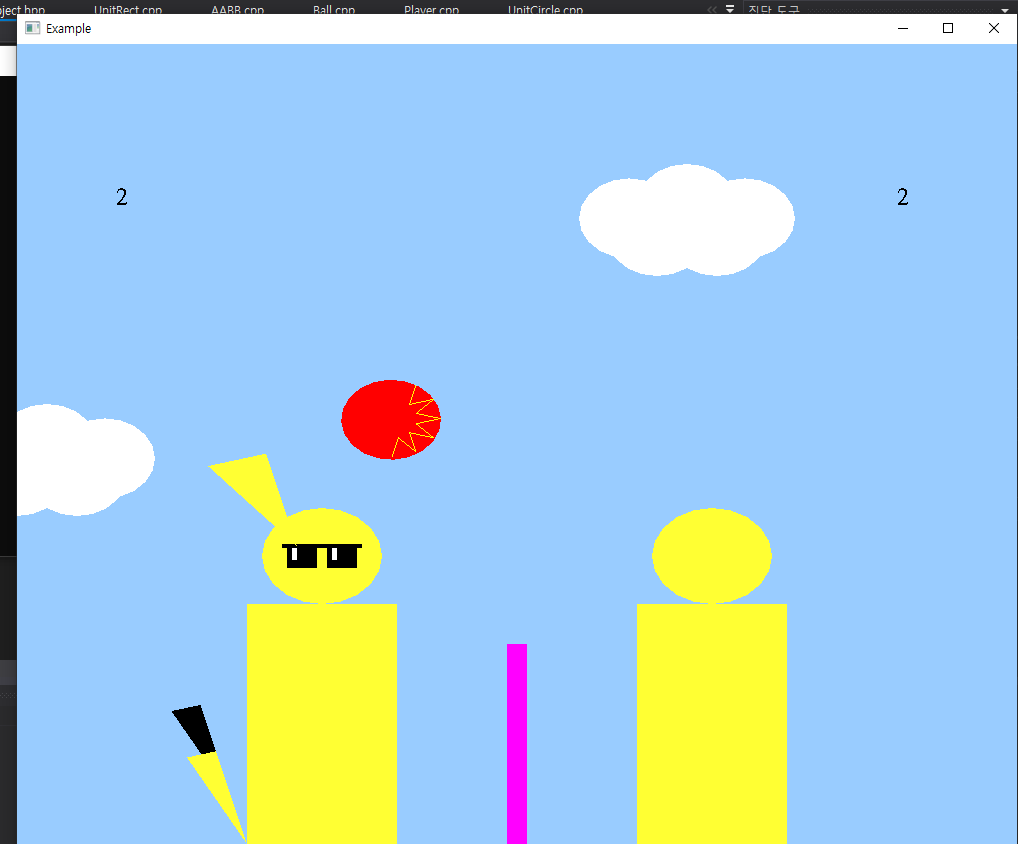


Figure 2 게임 화면이다. 구름, 꼬리, 귀, (공에 흐르는) 전기, 선글라스 등이 추가된 것을 확인할 수 있다.

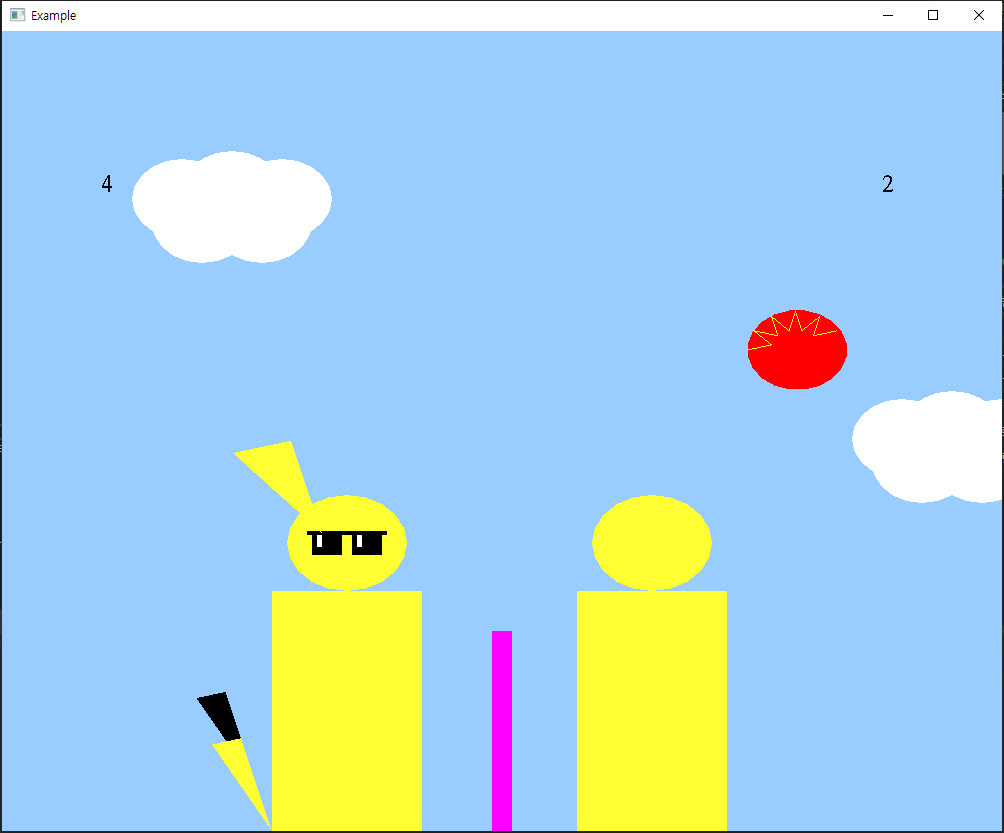


Figure 3 시간에 따라 구름과 전기의 위치와 모양이 변하는 것을 확인 할 수 있다.

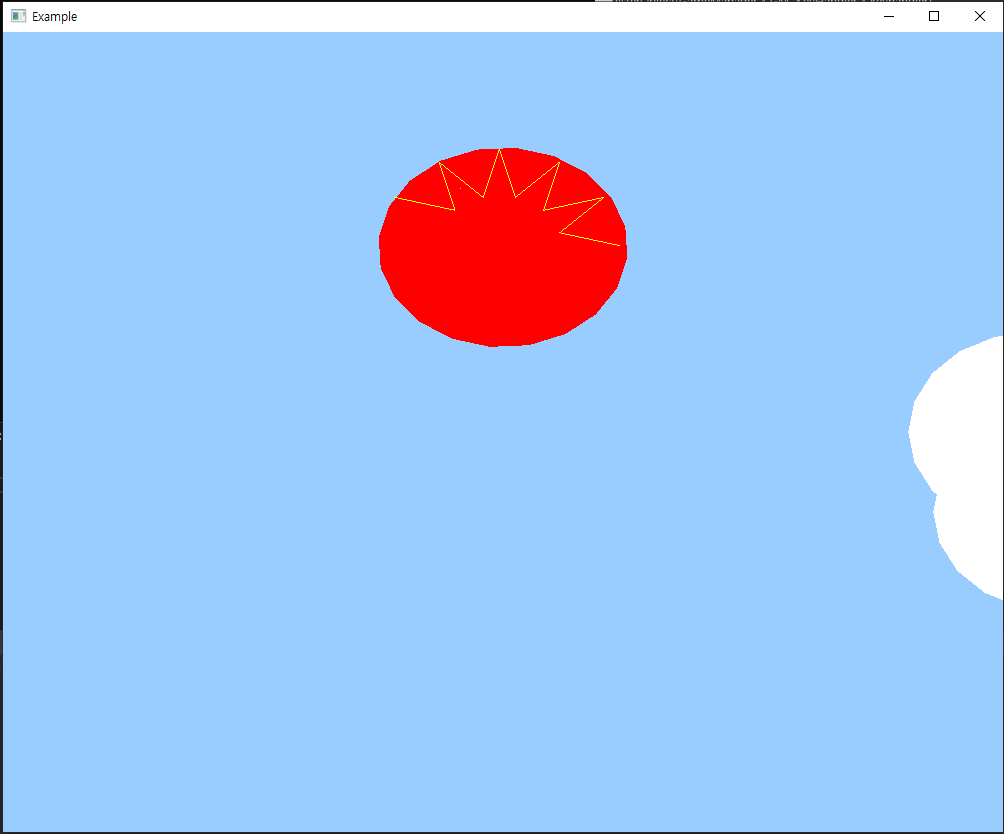


Figure 4 Zoom mode(‘x’키 눌렀을 때)에서 공의 중심을 기준으로 회전하는 전기의 모습이다.

# 토론

처음에는 귀와 꼬리 등의 회전 운동을 구현할 때 object마다 개별적으로 draw 함수 내에서 rotate을 했으나 원하는 대로의 회전을 구현하기 힘들었다. 하지만 Structure를 정리해서 회전하는 각도를 정해주는 부분과 draw하는 부분을 따로 나눔으로써 디버깅 작업을 손쉽게 할 수 있었고 원하는 회전을 나타낼 수 있었다.

# 결론

이번 과제에서는 기존의 Game Object에 hierarchy를 추가함으로써 전보다 더 다이내믹한 2D 애니메이션을 구현할 수 있었다. 물체 자체의 움직임 뿐만 아니라 상위 object를 기준으로 한 하위 object의 transformation을 나타낼 수 있었다.

# 개선 방향

내용내용내용

# 인용 자료