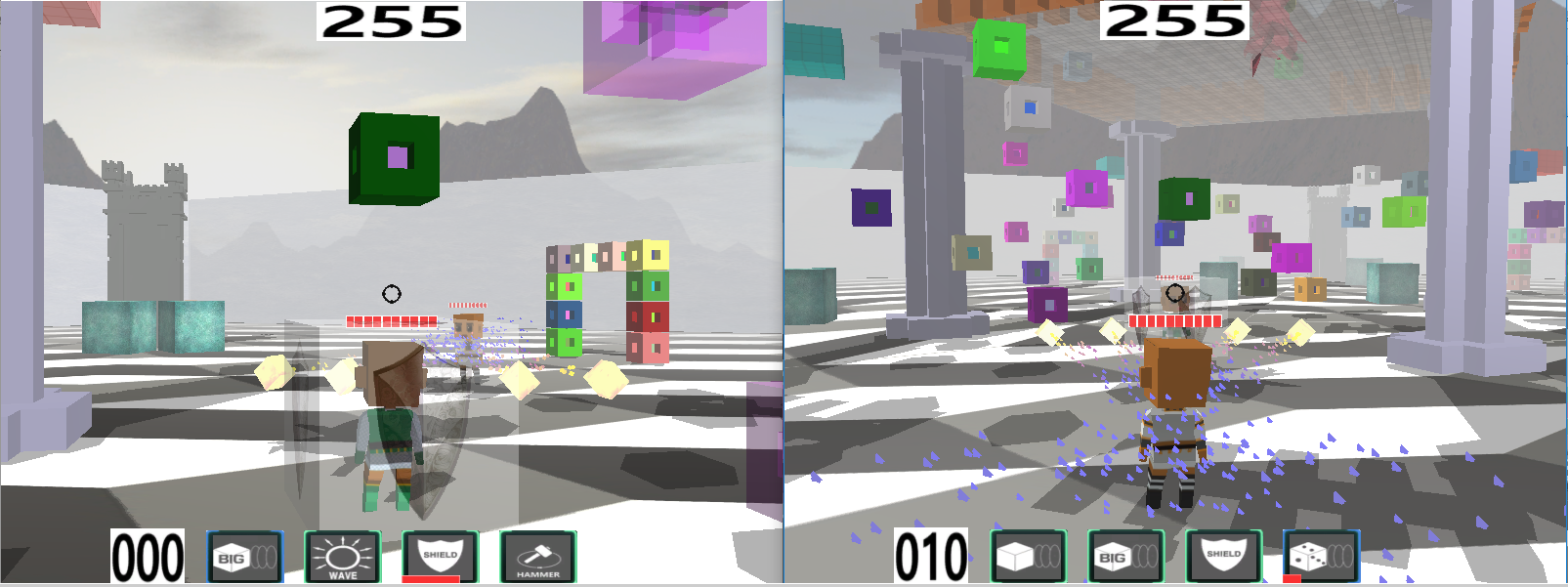
**[서버]**

**[1]. Boost asio와 멀티 스레드를 이용한 3인칭 슈팅 대전게임 [졸업작품] (2018년 제작)**

1. **기본 설명**

* **게임 제목: MagicCube**
* **깃 허브:** [**https://github.com/ChangminYoo/MagicWorld**](https://github.com/ChangminYoo/MagicWorld)
* **제작 도구: C++, C++ STL, Boost asio 라이브러리, DirectX12**
* **개발 환경: Visual studio 2017**
* **장르: 3인칭 슈팅 대전 게임**
* **플랫폼: PC**
* **제작 인원: 3명 (클라이언트 2, 서버 1)**
* **제작 기간: 약 12개월(기획 및 팀 조정 3개월, 제작 7개월, 추가 수정 2개월)**
* **역할 및 목표: boost asio를 이용한 서버 프레임워크 및 서버 통신을 위한 클라이언트 프레임워크 제작, 물리엔진 및 각종 게임 내 모든 컨텐츠 서버 클라이언트 간 동기화 . 비동기 입출력 IO인 Boost asio와 멀티 스레드를 이용한 대규모 접속 가능 서버 제작**
* **맵 크기: 300 x 300 (m2)**

**[게임 플레이 화면]**

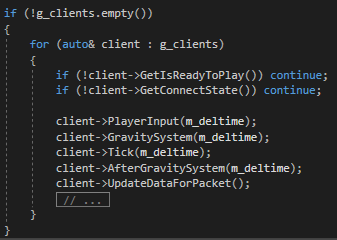
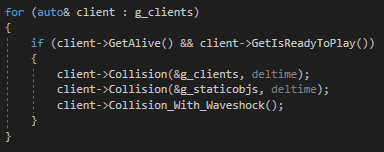
1. **세부 내용 설명**

* **게임 목표: 대전 스테이지에서 제한 시간 (2분 30초) 안에 스킬을 사용하여 적 플레이어를 처치하거나 2층에 존재하는 몬스터를 처치하여 가장 많은 점수를 획득하여 1등을 하는 것이 최종 목표입니다.**
* **게임 내용: 최대 5명이 개인전으로 진행되는 3인칭 슈팅 대전 게임입니다. 플레이어는 4개의 고유 스킬을 지닌 10종류의 캐릭터 중 하나를 선택하여 게임을 플레이 할 수 있습니다. 맵에는 물리엔진이 적용된 오브젝트, 움직이는 오브젝트, 움직이지 않는 설치형 오브젝트가 존재하며 1층과 2층으로 나뉘어져 구성되어 있습니다. 2층에는 몬스터 NPC가 존재하며 플레이어는 플레이어를 처치하거나 몬스터를 처치하여 자신의 게임 스코어를 올릴 수 있습니다. 스코어가 가장 높은 1등 플레이어에게는 게임에 유리한 버프 효과가 적용되며 스테이지 제한 시 모두 지나면 게임이 종료됩니다.**
* **내부 구현: main 스레드에서 서버 접속 이전에 진행되는 작업 과정인 현재 컴퓨터의 CPU 개수 체크, 맵 및 맵에 존재하는 오브젝트 데이터 설정, 몬스터 npc 데이터 설정을 진행합니다. 현재 이 프로그램에서 작동되는 스레드는 main 스레드를 포함하여 총 9개이며, 클라이언트와 서버 간 송수신되는 패킷 데이터를 처리하는 6개의 스레드, 물리엔진 및 충돌처리를 담당하는 1개의 스레드 그리고 타이머에 따라서 주기적인 데이터처리를 하는 1개의 스레드로 구성되어 있습니다.**

**위에서 언급한 초기화작업 완료 후 boost asio의 async\_accept를 이용하는 Accept 함수 부분에 접근하여 클라이언트와의 접속처리를 한 뒤 Callback되는 handler 부분에서 서버에서 관리 할 클라이언트 부분을 설정하고 boost asio의 async\_read\_some을 이용하는 RecvPacket함수 호출을 한 뒤 해당 비동기 IO에 의해 Callback 되는 handler 부분에서 클라이언트부터 오는 패킷 데이터를 재조립하여 수신된 패킷을 처리하는 부분인 ProcessPacket 함수를 처리합니다.**

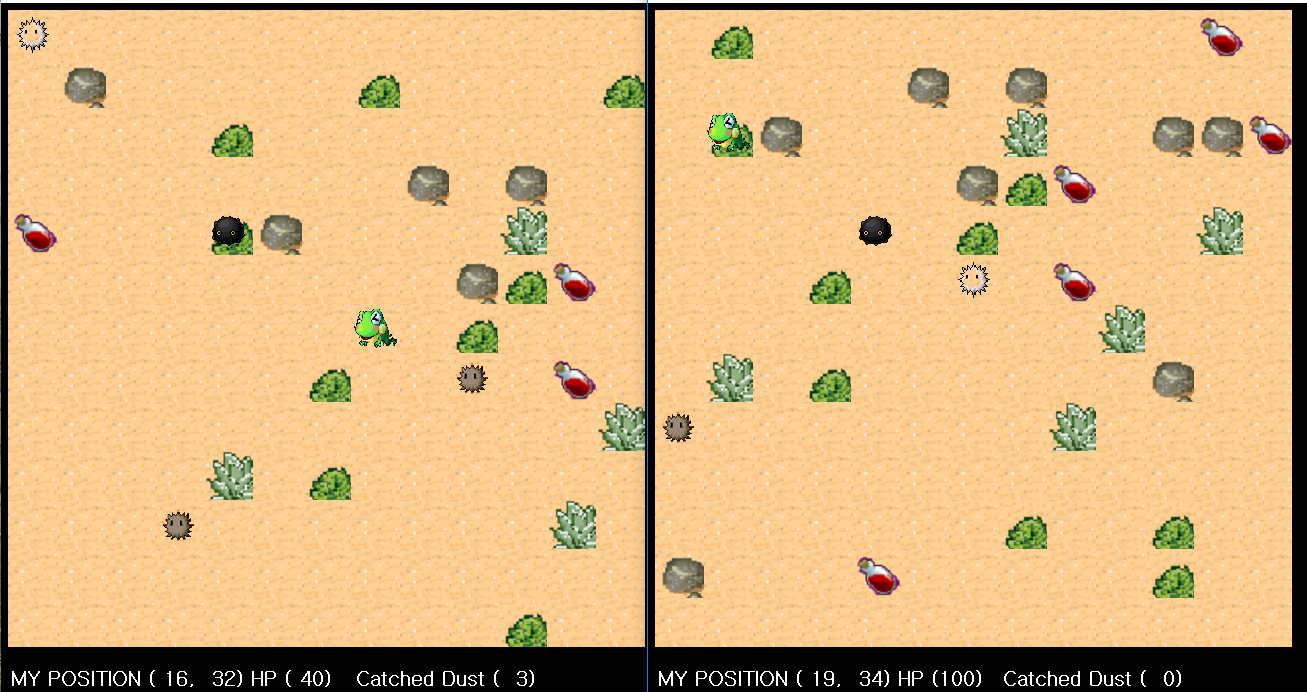
**[전체 서버의 구조를 나타낼 수 있는 Flow Chart, 혹은 UML 비슷한 도표]**

**ProcessPacket 함수에서는 넘어온 패킷 데이터에 설정된 패킷 타입에 따라서 데이터를 처리합니다. 로비 씬, 씬 변환, 플레이어의 기본 동작(이동, 회전, 점프, 애니메이션, 공격) 및 스킬, 주기적인 패킷 처리(변화된 플레이어 정보, 공격 정보, 스킬 정보, NPC 정보, 움직이는 맵 오브젝트 및 물리효과가 적용된 오브젝트 정보)에 대한 데이터 처리를 진행합니다.**

****

**[ 모든 오브젝트에 적용되는 물리효과 및 충돌처리를 하는 PhysicsUpdate 함수]**

**이 부분에서 게임 내 모든 오브젝트(플레이어 오브젝트, 플레이어 공격에 의해 생성된 불렛 오브젝트, 맵 오브젝트, NPC 오브젝트)에 적용되는 중력, 가속도 등에 대한 물리효과처리 및 Rigidbody 오브젝트에 설정된 물리 값들에 따른 물리효과 처리를 진행합니다. 이 외에 게임 내에서 진행되는 모든 충돌처리를 진행합니다. 이 부분에서 FSM이 적용된 몬스터 NPC의 데이터 또한 업데이트 됩니다. 몬스터 NPC는 FSM방식으로 구현되어 자신과의 일정거리에 플레이어가 접근하면 대상으로 이동하여 공격을 하고 그 위치를 벗어나면 다시 제자리로 돌아옵니다.**

** [2]. IOCP와 멀티 스레드를 이용한 2D 대전 게임 (2018년 제작)**

1. **기본 설명**

**- 게임 제목: Dust Eating**

**- 깃 허브: https://github.com/alclsTit/2018GameServer**

**- 제작 도구: C++11, Direct3D9, IOCP**

**- 개발 환경: Visual studio 2017**

**- 장르: 대전 게임**

**- 플랫폼: PC**

**- 제작 인원: 1명**

**- 제작 기간: 약 2개월 (기획 및 그래픽데이터(스프라이트, 이미지) 수집 1주, 제작 6주)**

**- 목표: 비동기 IO를 지원하는 IOCP 소켓모델과 멀티 스레드를 이용한 대규모 접속가능 서버 제작**

**- 역할: IOCP를 이용한 서버 프레임워크 제작, 멀티 스레드를 이용한 작업 분담처리, 시야처리, 충돌처리, 몬스터 NPC 구현, 맵 및 오브젝트 구현, DB 연동, LUA를 이용한 메시지 띄우기**

**- 맵 크기: 200 x 200 (pixel)**

**- 최대 동시 접속: 평균 1500명**

1. **세부 내용 설명**

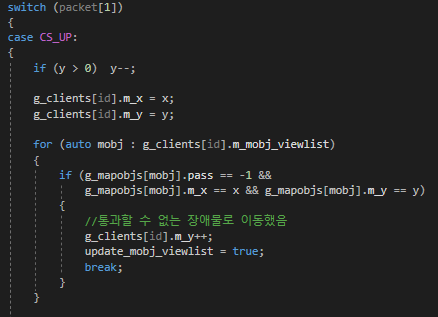
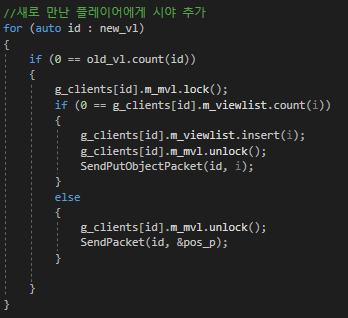
* **게임 목표: 체력을 유지하며 맵 상에 떠다니는 먼지 몬스터 NPC를 상대 플레이어 보다 더 많이 잡아먹어 더 많은 먼지를 없애는 것이 최종 목표입니다.**
* **게임 내용: DB에 저장된 로그인정보를 통해 게임에 접속하여 게임을 진행합니다. 맵 상에 존재하는 각종 장애물들을 피해서 떠다니는 몬스터 NPC를 잡아먹어야 합니다. 플레이어 자신은 검은 성게로 표시되며 상대방은 개구리 이미지로 표시되어 나타납니다. 2종류의 몬스터 NPC 중 한 종류는 플레이어를 공격해 피해를 입히며 플레이어는 맵 상의 체력 포션을 먹어가며 모든 체력이 소진되기 전까지 상대 플레이어보다 더 많은 먼지 몬스터 NPC를 잡아먹어야 합니다.**
* **내부 구현:**

**[1]. 스레드 할당 및 각 스레드에서 하는 작업**

**스레드는 총 8개를 사용하며 클라이언트의 접속 처리를 담당하는 accept\_thread는 1개, 클라이언트와 서버 에서 송수신되는 패킷 데이터 처리를 하는 부분인 worker\_thread 는 6개, 주기적인 행동을 하는 몬스터 NPC에 대한 데이터처리를 하는 부분인 timer\_thread 1개로 구성되어 있습니다.**

**플레이어가 움직일 때마다의 업데이트되는 시야, 플레이어 위치, 충돌 처리 , 1초 마다 랜덤하게 변화되는 위치 및 그에 따른 시야처리 등 지속적으로 업데이트 되는 데이터를 클라이언트와 송수신 해야 되는 많은 부하가 생길 수 있는 작업을 담당하는 worker\_thread에 가장 많은 스레드를 할당하였습니다.**

**[2]. 클라이언트에서 송신 된 패킷 데이터를 처리하는 ProcessPacket 부분 및 시야처리 부분**

**아래의 샘플 코드는 수업시간에 보여준 코드이며 10년간 졸업생들이 포트폴리오로 사용해서 많은 회사에서 이미 본 코드이고, 같은 코드를 제출한 학생들이 이번 연도에도 분명히 존재한다. 따라서, 빼던가, 재작성해서 사용할 것)**

**키보드 입력을 통해 캐릭터를 위로 움직였을 때 받아온 패킷 데이터를 처리하는 부분으로 서버에서 관리하는 해당 플레이어의 위치 업데이트, 아이템, 몬스터, 맵 오브젝트와의 충돌체크를 합니다. 플레이어의 위치가 업데이트 되었으므로 이를 이용하여 자신의 시야(7 x 7) 안에 플레이어 혹은 몬스터 NPC가 있는지 자신의 시야를 업데이트 합니다.**

**[3]. MSSQL 2017를 이용한 DB 연동**

**(샘플은 Stored Procedure 작성할 것 )**

**플레이어의 로그인, 위치, 먹은 먼지 몬스터 NPC의 수를 DB에 저장하여 서버와의 접속이 끊겨도 해당 정보가 손실되지 않도록 하였습니다. DB는 MSSQL 2017을 이용하였고 SQL 명령어를 직접 서버에서 작성하여 사용하는 방식이 아닌 Stored Procedure을 통해 SQL명령을 실행하는 더 효율적인 방식으로 DB연동을 하였습니다.**

**[클라이언트]**

**[1]. Win32 API 및 C++을 이용한 2D 메이플스토리 모작 (2016년 제작)**

1. **기본 설명**

**- 게임 제목: Project M**

**- 깃 허브: -**

**- 제작 도구: C++, Win32 API**

**- 개발 환경: Visual studio 2013**

**- 장르: 2D RPG**

**- 플랫폼: PC**

**- 제작 인원: 1명**

**- 제작 기간: 약 3개월**

**- 목표: 어릴 적에 재미있게 플레이 해봤던 메이플 스토리를 직접 제작**

**- 역할: C++언어를 이용하여 사용할 클래스 구현 및 프레임워크 제작, 플레이어 구현(이동, 점프, 공격, 4종류의 스킬), 몬스터 구현(이동, 공격, 스킬 사용, AI, 리스폰), 아이템 구현(표창), 퀘스트 구현(단순한 텍스트 출력), 스테이지 전환 및 화면 이동 구현, 인터페이스 구현, BGM 추가**

**[2]. Python 을 이용한 2D 러닝 + 리듬게임 제작 (2016년 제작)**

1. **기본 설명**

**- 게임 제목: Run your beat**

**- 깃 허브: https://github.com/alclsTit/2D-Game-Programming**

**- 제작 도구: Python, Pico-2d(수업시간에 제공받은 모듈)**

**- 개발 환경: PyCharm**

**- 장르: 2D 러닝 + 리듬 게임**

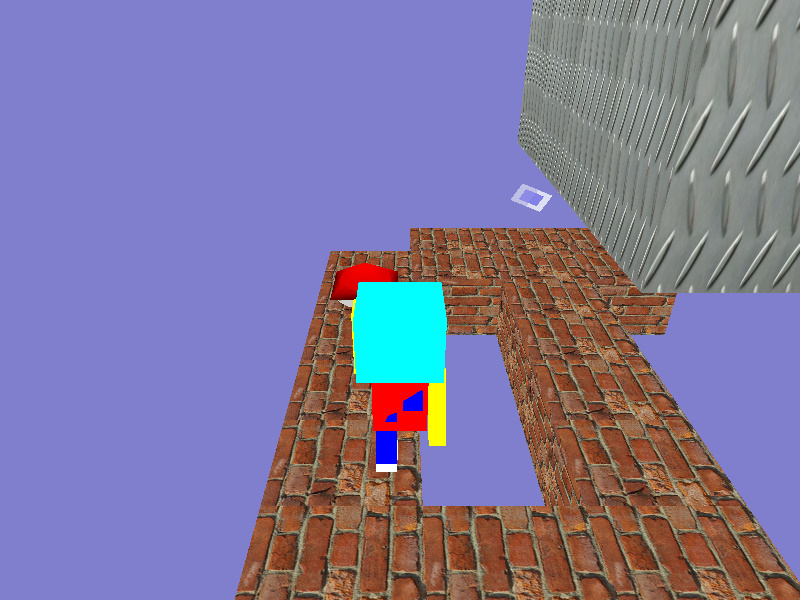
**- 플랫폼: PC**

**- 제작 인원: 1명**

**- 제작 기간: 2개월 ~ 3개월**

**- 목표: Python과 제공받은 모듈을 이용하여 러닝과 리듬을 합친 새로운 게임을 제작**

**- 역할: Python 모듈인 pico-2d를 이용하여 PNG 스프라이트, 이미지 출력. 스테이지 및 각종 오브젝트 구현, 플레이어 구현, 몬스터 NPC 구현, 리듬게임 요소 구현(상단에 표시되는 A,S,D,F, 폭탄, 황금돼지에 맞는 키를 음악에 맞게 눌러 스코어를 획득), 랭킹 시스템 구현, 로비 씬 구현(플레이할 노래 선택), 종료 씬 구현**

**[3]. C++, OpenGL을 사용한 3인칭 3D 탐험 게임 제작 (2017년 제작)**

1. **기본 설명**

**- 게임 제목: Cube Mario**

**- 깃 허브: -**

**- 제작 도구: C++, OpenGL**

**- 개발 환경: Visual Studio 13**

**- 장르: 3D 액션 게임**

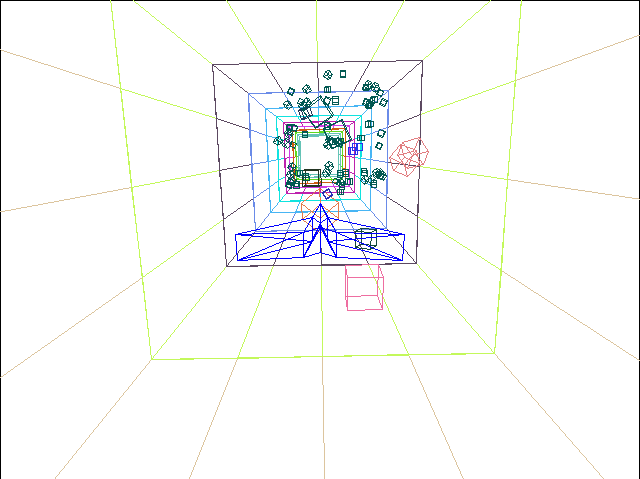
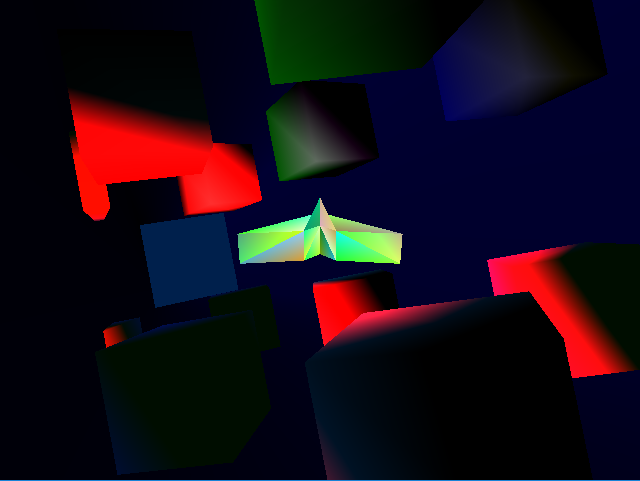
**- 플랫폼: PC**

**- 제작 인원: 2명**

**- 제작 기간: 2개월 ~ 3개월**

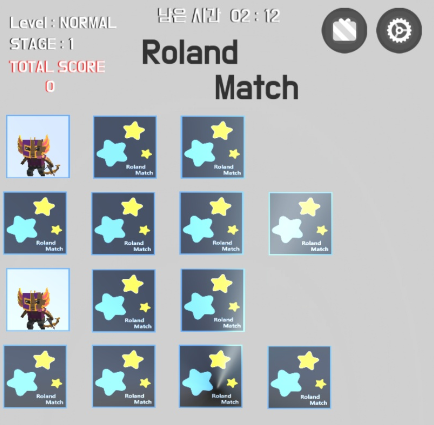
**- 목표: OpenGL을 이용하여 플레이 가능한 3D 게임 제작**

**- 역할: 플레이어 캐릭터 구현. 버섯 몬스터 구현 및 버섯과의 충돌처리 부분 구현. 캐릭터 사망 시 생성되는 큐브 형태의 파티클 구현. 스테이지 제작. 텍스쳐 수집 및 적용**

**[4]. DirectX 12을 사용한 기초적인 3D 게임 제작 지식 습득**

**3학년 1학기 때 3D 게임프로그래밍 수업을 들으면서 DirectX12를 이용한 3D게임 제작에 대한 기초적인 지식을 습득 할 수 있었습니다.**

**Direct3D 12 장치 생성부터, CPU와 GPU간 명령 대기열을 통한 데이터 업로드 및 업로드 된 데이터 사용. SwapChain을 이용한 전면 버퍼와 후면 버퍼 교환, 서술자 힙 생성 및 렌더 타겟 뷰, 깊이 스텐실 뷰 생성, 뷰 포트 설정, 파이프 라인 단계, 셰이더에 대한 지식, 조명, 텍스쳐 등 DirectX12이라는 그래픽라이브러리의 사용법에 대해서 전반적으로 배웠습니다.**

**[5]. Unity3D 엔진을 이용한 짝 맞추기 게임 제작 (2017년 제작)**

**해당 게임은 3학년 1학기 게임엔진 수업과 인터페이스 수업을 통해서 배운 Unity3D 엔진을 이용하여 8주간 진행 된 현장실습 기관에서 제작한 3D모델을 사용한 짝 맞추기 게임입니다.**

**게임 진행은 타이틀 화면에서 난이도(EASY, NORMAL, HARD)를 선택하고 각 난이도에 맞게 다양한 캐릭터가 표시된 카드들이 배치됩니다. 제한 시간 안에 같은 캐릭터 짝을 모두 맞추면 해당 스테이지에서 획득한 스코어를 보여주며 다음 스테이지로 이동합니다. 스테이지는 각 난이도 마다 50개이며 총 150개의 스테이지로 이루어져 있습니다.**