**연구개발과제 최종보고서**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **과제명** | 프로그래밍언어론 프로젝트 개발 | | | | | |
| **책 임 자** | **성 명** | 이상민 | | | | |
| **소 속** | 서울건국대학교 인터넷미디어공학부 | | | | |
| **학 년** | 3 | | **학 번** | 201111306 | |
| **개발기간** | 2015년 11월 8일 ～ 2015년 12월 8일 | | | | | |
| **참여학생** | **학번** | | **이름** | | | **전공** |
| 201311197 | | 김보하 | | | 인터넷미디어 |
| 201111419 | | 김진솔 | | | 인터넷미디어 |
| 201111305 | | 이동호 | | | 인터넷미디어 |
|  | |  | | |  |
|  | |  | | |  |
| 본 보고서를 프로그래밍언어론 프로젝트 개발과제의 최종보고서로 제출합니다.  (첨부) 프로그래밍언어론 프로젝트 최종보고서  2015 년 12월 8일  책임자 이상민    **담당 교수** 귀하 | | | | | | |

**CONTENT**

**1. 개요**

개발 목표

개발 배경 및 동기

프로젝트 주요 특징

조원 구성

**2. 개발 시스템 개요**

구현할 기능

동작 순서도

**3. 개발내용**

프로그램 시나리오 및 시스템 구성

각 시스템 별 상세 설명

**4. 구현 결과 및 테스팅**

실제 기능별 구현 내용

테스팅 결과 (실행화면)

**5. 부록**

소스코드 핵심 부분

프로그램 실행하기

프로젝트 참여현황 및 일정

건의사항

**1. 개요**

**1.1 개발 목표**

최종 목표: 2D 프로그래밍에 네트워크를 접목시켜 2인용 횡 스크롤 게임을 제작한다.

1. 키보드 조작

게임 내 캐릭터를 움직여 맵의 오브젝트와 상호작용이 가능하게 한다.

게임 내 블록을 움직여 맵 자체와 상호작용이 가능하게 한다.

1. 횡 스크롤

왼쪽에서 오른쪽으로 이동하는 맵 상에서 캐릭터가 게임을 지속한다.

1. 네트워크

1개의 서버에 2개의 클라이언트가 접속하여 게임을 진행한다.

서버를 통해 현재 캐릭터의 위치와 맵에 대한 정보를 실시간으로 주고받는다.

서버를 통해 현재 테트리스의 위치와 맵에 대한 정보를 실시간으로 주고받는다.

1. 다중 역할

Player 1은 캐릭터를 움직여 맵을 진행한다.

Player 2는 Player 1의 원활한 진행을 위해 블록을 쌓는다.

**1.2 개발 배경 및 동기**

Python 언어에 관한 강의를 들으면서 이 언어를 사용한 게임을 만들고 싶었다. 객체 지향적이면서 사람이 익히기 편한 고급 언어인 Python을 이용하면 어렵지 않게 2D 프로그래밍이 가능할 것으로 보았다.

사람이 흥미를 느끼는 것에는 친근감이 큰 요소로 작용한다. 최근 중국의 게임 물량 공세와 더불어 쏟아져 나오는 수 많은 게임들 중 살아남는 게임은 몇 되지 않는다. 이런 와중에 우리가 만들 게임이 살아남기 위해서는 익숙하지 않은 개념의 게임보다는 **쉬운 조작 방법**과 **친숙한 플레이 방식**의 게임이 적합하다.

게임이 주목 받기 위해서는 **첫 인상**이 중요하다. 누구나 쉽게 게임을 다운받고 삭제할 수 있는 현 시점에서 유저를 붙잡으려면 긍정적인 인상을 심어야 한다. 라이트 유저가 갈수록 많아지면서 게임은 갈수록 쉬워져야 하고, 그로부터 얻는 행복은 갈수록 커져야 한다. 그렇게 하기 위해 우리는 이동, 점프(혹은 선택)만을 위한 몇 개 안 되는 조작 키와 누구나 해봤을 법한 테트리스 방식을 조합하여 게임을 만들기로 하였다.

플레이스토어, 앱 스토어에 등록되는 게임의 종류는 매우 다양하지만 한 가지 공통점을 가지고 있다. 높은 순위의 게임들은 모두 2인 이상이 같이 플레이 가능한 **멀티게임**이라는 점이다. 혼자서 하는 게임이라고 할지라도 대부분 업적, 랭킹 시스템을 통해 다른 사람들과 경쟁이 가능한 구도로 만들어야 성공의 지름길이 된다.

우리가 만들 게임이 지금은 Visual Studio를 사용하여 컴퓨터로 실행되지만 앱 게임으로 만드는 것은 플랫폼의 변화일 뿐이지, 개발 환경에 한계를 두는 것은 옳지 않다. 즉, 시장의 흐름을 따라 **네트워크 게임**을 만드는 것은 중요하다.

**1.3 프로젝트 주요 특징**

프로그램 “쌓아라 뛰어라”는 지금까지는 없었던 새로운 형태인 “**협동-런** 게임”이다. 한 플레이어는 계속해서 달리는 **런 플레이어**가 된다. 이 플레이어가 오래 살아남을수록 높은 점수를 얻을 수 있다. 다른 플레이어는 런 플레이어가 죽지 않고 달릴 수 있게 도움을 주는 **협동 플레이어** 역할을 맡는다.

런 플레이어는 항상 도움을 받지 않으면 진행이 불가능한 상황에 놓인다. 이러한 상황을 부여함으로써 위기 대처 능력을 발휘하여 **극복**해나가는 과정에서 **재미**가 추구되며, 색다름을 제공한다. 또한 같은 상황에 처해도 매번 다르게 대처를 할 수 있다는 점과 어떠한 상황이 주어질 지 모른다는 점에서 **지속적인 게임 플레이**를 보장한다.

이 프로그램의 클라이언트는 2D Game Programming에 적합화된 **PyGame** 라이브러리를 사용하여 만들어졌으며, PyGame 내부의 **PodSixNet** 라이브러리를 사용하여 서버가 제작되었다.

**1.4 조원 구성**



**2. 개발 시스템 개요**

**2.1 구현 기능**

* **클라이언트**

1. 맵 데이터로부터 값을 읽어 들여 맵 생성
2. 캐릭터와 맵 간의 충돌체크
3. 아이템 획득 및 사용
4. 스테이지별 난이도 설정
5. 스테이지 목표
6. 게임 클리어, 게임 오버

* **네트워크(2인용)**

1. 서버 생성

* Player 0, Player 1 모두 클라이언트로 서버에 접속
* 포트 번호와 클라이언트의 소켓 및 채널 획득
* 채널을 통하여 유저간에 실시간 통신

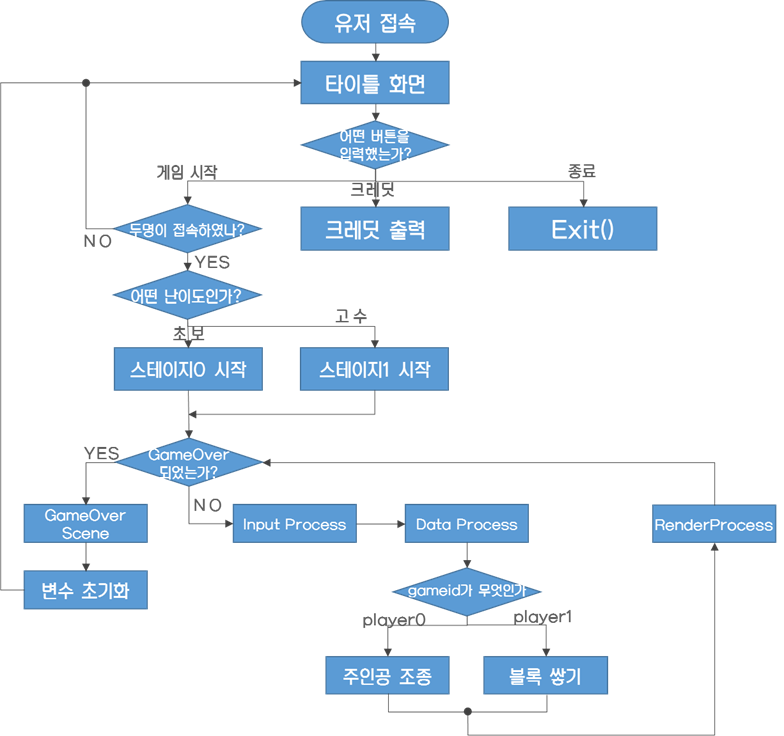
1. 데이터 전송

* Player 0 캐릭터의 위치를 서버로 전송
* Player 1 블록의 위치를 서버로 전송
* 상대 Player에게 데이터 전송

1. 동기화

* 상대 Player의 데이터를 적용
* 맵 상황 동기화

**2.2 동작 순서도**



**3. 개발내용**

**3.1 프로그램 동작 시나리오 및 시스템 흐름**

* 유저 접속1

클라이언트 프로그램을 이용하여 접속을 한다. 클라이언트 프로그램은 시작하자마자 NetworkListener 객체를 생성하여 서버와 통신한다. 서버는 accept상태로 대기하고 있다가 클라이언트로부터 연결 요청이 들어오면 클라이언트와의 통신 매개체인 채널을 서버에 저장해둔다.

* 유저 접속2

클라이언트 프로그램을 이용하여 접속을 하였을 때, 자신이 두번째에 접속한 유저라면 게임을 시작할 준비가 된것이다. 그러므로 저장해둔 채널정보를 이용하여 첫번째 유저와 두번째 유저에게 유저가 모두 연결되었다는 메시지와 유저에게 gameid를 부여하여 전송한다.

* 유저 접속2-2

클라이언트가 접속하고 서버에 접속한 후로, 유저가 모두 연결되었다는 메시지가 온다면 타이틀 화면 하단에 유저 접속 표시등을 초록불로 킨다. 그리고 나의 gameid를 부여받고, gameid에 따라 player0와 player1으로 나눈다. 만약에 자신이 player0라면 게임 시작 버튼을 누를 수 있도록 활성화한다.

* 유저 접속3

클라이언트 프로그램을 이용하여 접속을 하였을 때, 접속 가능한 최대 인원을 넘어서면 접속한 클라이언트에게 overflow 메시지를 보내, 인원이 가득 차서 접속할 수 없다는 메시지와 함께 클라이언트를 종료시킨다.

* 게임 시작

(유저가 두명 접속한 상태일 때) 타이틀 화면에서 게임시작 버튼이 활성화 되었다면, 원하는 난이도를 선택하여 게임을 시작시킨다.(STATE변수 값 수정) 그리고 게임을 시작하였다는 것을 서버에게 보낸다. 이때 어떤 난이도를 선택하였는지도 함께 보낸다.

* 게임 시작2

서버는 대기하고 있다가, player0로부터 게임시작 메시지를 받으면 player1에게 게임이 시작되었다는 메시지를 게임 난이도와 함께 전송한다. 게임 시작 메시지를 받은 유저 player1는 STATE변수 값 수정을 통해 게임이 시작하였다는 것을 기록한다.

* 게임 플레이

게임이 시작된 두 유저는 gameid에 따라 캐릭터를 조종할 지, 블록을 움직여 맵을 만들지 결정된다. (gameid==0 : 캐릭터 조종; gameid==1: 블록 조종;)

스테이지가 시작되고 화면은 자동으로 우측으로 이동된다. 그러므로 캐릭터는 우측으로 꾸준히 이동하며 앞으로 나아가야만 한다.

* 게임 조작1

player0는 캐릭터를 조종시키며 상황에 맞게 점프키와 이동키를 이용하여 게임을 진행시키는 역할이다. 이때 캐릭터 위치정보와 점프정보는 클라이언트 tick에 맞추어 실시간으로 서버에 전송된다. 이 정보를 받은 서버는 블록의 조종을 맡은 player1에게 전송되며, player0가 조종한대로 캐릭터를 위치를 이동시킨다. 이는 두 유저간의 화면 동기화를 구현한다.

추가적으로 player0로 인해 캐릭터의 방향을 바꾸거나와 같은 새로운 움직임 조종을 입력받았을 경우, 캐릭터가 어떠한 움직임을 수행하였는지 서버에 전송한다. 이는 player1에게 다시 전송되며, 이 정보를 통하여 캐릭터의 그래픽 리소스를 상황에 맞게 수정해주고 애니메이션을 줌으로써 자연스러운 캐릭터 모습을 연출한다.

* 게임 조작2

player1는 블록을 조종시키며 캐릭터가 이동할 수 있도록, 블록을 쌓아 길을 만들어주는 역할이다. 캐릭터가 올바른 길로 갈 수 있도록 하는 것이 목표이다.

방향키를 이용하여 블록을 이동시켰을 경우 블록의 위치정보와 회전정보를 서버에 전송하고 이 정보는 캐릭터를 조종하는 player0에게 전송되며, 블록을 player1이 조종한대로 이동시킴으로써 화면 동기화를 구현한다. 이때 블록 이동 이벤트를 넘기는 것이 아니라 블록의 포지션을 넘김으로써 이벤트가 씹히는 현상이 발생하더라도 두 유저간의 싱크가 맞도록 구현하였다.

추가적으로 블록이 맵에 닿을경우 블록을 맵 변수에 넣어 캐릭터가 밟고 이동할 수 있도록 한다. (이 또한 서버를 통해 상대에게 전송된다.)

* 게임 오버

캐릭터가 움직이다가 좌측 화면 밖으로 나가거나 바닥으로 빠질경우, 그리고 적에게 부딪칠경우 GAMEOVER된다. 이러한 게임오버 체크는 캐릭터를 조종하는 player0만 수행하며, 두 유저가 동시에 게임오버 화면을 볼 수 있도록 서버를 통해 player0과 player1 둘다에게 정보를 전송한다. 게임오버 이벤트를 받은 클라이언트는 STATE변수 값을 수정하여 Gameover Scene을 보여준다. 해당 Scene에는 Score가 보여진다.

* 게임 오버 후

사용하였던 변수들을 초기화하고 타이틀화면으로 이동한다.

**3.2 각 시스템 별 상세 설명**

**3.2.1 클래스**

**myMap**

* 멤버 변수

Type: 해당 맵 타일의 종류(비어있는지, 블록인지에 대한 정보)

x, y: 해당 맵 타일의 픽셀 좌표

* 멤버 함수

|  |  |
| --- | --- |
| 클래스 명 | \_\_init\_\_ |
| 함수 전달인자 |  |
| 함수 역할 | 생성자 |
| 1. 클래스의 멤버 변수들 초기화 | |

**myEnemy**

* 멤버 변수

x, y: 해당 enemy의 픽셀 좌표

speed: 해당 enemy의 속도

dir: 해당 enemy의 이동 방향

img: 해당 enemy의 이미지

width: 해당 enemy의 가로길이

height: 해당 enemy의 세로 길이

fall: 해당 enemy의 추락여부 boolean값 저장

bPop: 해당 enemy의 pop여부 boolean값 저장

* 멤버 함수

|  |  |
| --- | --- |
| 클래스 명 | \_\_init\_\_ |
| 함수 전달인자 |  |
| 함수 역할 | 생성자 |
| 1. 클래스의 멤버 변수들 초기화 | |

**myCoin**

* 멤버 변수

x, y: 해당 coin의 픽셀 좌표

img: 해당 coin의 이미지

width: 해당 coin의 가로길이

height: 해당 coin의 세로 길이

bPop: 해당 coin의 pop여부 boolean값 저장

* 멤버 함수

|  |  |
| --- | --- |
| 클래스 명 | \_\_init\_\_ |
| 함수 전달인자 |  |
| 함수 역할 | 생성자 |
| 1. 클래스의 멤버 변수들 초기화 | |

**3.2.2 함수**

1. initProcess()

|  |  |
| --- | --- |
| 함수 명 | initMap |
| 함수 전달인자 | txt |
| 함수 역할 | 맵 초기화 |
| 1. txt를 받아 open하여 맵 정보 입력 2. 맵 정보에 따라 맵 리스트 초기화 | |

|  |  |
| --- | --- |
| 함수 명 | initEnemy |
| 함수 전달인자 | x, y |
| 함수 역할 | 적 초기화 |
| 1. x, y 맵의 인덱스를 입력 받아 해당하는 픽셀 좌표로 변환 2. 변환한 좌표 및 기타 myEnemy class의 변수를 초기화하여 m\_Enemy 리스트에 append | |

|  |  |
| --- | --- |
| 함수 명 | initCoin |
| 함수 전달인자 | x ,y |
| 함수 역할 | 코인 초기화 |
| 1. x, y 맵의 인덱스를 입력 받아 해당하는 픽셀 좌표로 변환 2. 변환한 좌표 및 기타 myCoin class의 변수를 초기화하여 m\_Coin 리스트에 append | |

1. inputProcess()

|  |  |
| --- | --- |
| 함수 명 | checkDown |
| 함수 전달인자 |  |
| 함수 역할 | 테트리스 블록 아래 체크 |
| 1. 떨어지고 있는 테트리스 블록의 각 블록들의 아래를 체크 2. 범위를 벗어나거나 비어있지 않으면 False 반환, 아니면 True 반환 | |
| 함수 명 | **checkLeftRight** |
| 함수 전달인자 |  |
| 함수 역할 | 테트리스 블록 양옆 체크 |
| 1. 떨어지고 있는 테트리스 블록의 각 블록들의 양옆을 체크 2. 범위를 벗어나거나 비어있지 않으면 False 반환, 아니면 True 반환 | |

|  |  |
| --- | --- |
| 함수 명 | fullDown |
| 함수 전달인자 |  |
| 함수 역할 | 테트리스 블록의 도착지점 체크 |
| 1. 떨어지고 있는 테트리스 블록의 각 블록들의 도착지점을 체크 2. 적당한 위치에 테트리스 블록들을 맵으로 변환 후 return | |

1. dataProcess()

|  |  |
| --- | --- |
| 함수 명 | resetMap |
| 함수 전달인자 |  |
| 함수 역할 | 스크린의 이동에 따라 후반부 Map추가 |
| 1. 사라진 기존의 맵 정보를 맵 리스트에서 pop 2. 새로 추가할 부분 text파일에서 읽어와 맵 리스트를 extend하여 각 인자에 초기화 | |

|  |  |
| --- | --- |
| 함수 명 | moveComponents |
| 함수 전달인자 |  |
| 함수 역할 | 스크린의 모든 components 이동 |
| 1. 일정 시간마다 스크린 상에 존재하는 모든 components의 x좌표를 왼쪽으로 이동 | |

|  |  |
| --- | --- |
| 함수 명 | netTetris |
| 함수 전달인자 |  |
| 함수 역할 | 새로운 테트리스 블록 생성 |
| 1. 새로운 테트리스 Dictionary 생성하여 생성한 테트리스 정보 서버에 전송 2. 생성한 테트리스 Dictionary 반환 | |

|  |  |
| --- | --- |
| 함수 명 | isBlocked |
| 함수 전달인자 |  |
| 함수 역할 | 떨어지는 테트리스의 다음 위치 막혀있는지 체크 |
| 1. 떨어지는 테트리스의 블록 인덱스를 맵의 인덱스로 변환 2. y좌표 인덱스+1의 위치에 맵에 블록 체크    1. 블록이 있으면 True 반환    2. 블록이 없으면 False 반환 | |

|  |  |
| --- | --- |
| 함수 명 | setOnMap |
| 함수 전달인자 |  |
| 함수 역할 | 떨어지는 테트리스 블록을 맵으로 변환 |
| 1. 떨어지는 테트리스의 블록 인덱스를 맵의 인덱스로 변환 2. 해당 맵의 인덱스에 테트리스 블록의 정보 저장하여 반영 3. 저장한 정보 서버로 전송 | |

|  |  |
| --- | --- |
| 함수 명 | convertBlockIdxToMapIdx |
| 함수 전달인자 | x, y, tetris |
| 함수 역할 | 테트리스 블록 인덱스를 맵의 인덱스로 변환 |
| 1. 테트리스 Dictionary에 저장되어 있는 x와 y인덱스를 맵의 인덱스로 변환 2. 변환된 맵의 인덱스 반환 | |

|  |  |
| --- | --- |
| 함수 명 | convertMapIdxToPixel |
| 함수 전달인자 | mapx, mapy |
| 함수 역할 | 맵의 인덱스를 화면 좌표로 변환 |
| 1. 맵의 x인덱스와 y인덱스를 각각 화면의 x, y좌표로 변환 2. 변환된 x, y좌표 반환 | |

|  |  |
| --- | --- |
| 함수 명 | convertPixelToMapIdx |
| 함수 전달인자 | x, y |
| 함수 역할 | 화면의 좌표를 맵의 인덱스로 변환 |
| 1. 맵 리스트를 순회하며 입력 받은 x, y좌표에 맞는 부분 탐색 2. 찾은 맵의 인덱스 반환 | |

|  |  |
| --- | --- |
| 함수 명 | collisionUp |
| 함수 전달인자 |  |
| 함수 역할 | 캐릭터의 위 부분과 맵의 충돌 체크 |
| 1. 캐릭터의 머리부분의 Rect와 전체 맵 타일의 Rect를 순회하며 충돌 체크    1. 맵에 블록이 있는 부분 중 충돌한 맵이 있으면 해당 인덱스 반환 2. 떨어지고 있는 테트리스 블록들의 Rect와 캐릭터의 머리부분의 Rect와 충돌체크    1. 충돌한 부분이 있으면 해당 인덱스 반환    2. 없으면 충돌한 부분이 없다는 정보 반환(음수 반환) | |

|  |  |
| --- | --- |
| 함수 명 | collisionBlockDown |
| 함수 전달인자 |  |
| 함수 역할 | 캐릭터의 아래부분과 맵의 충돌 체크 |
| 1. 캐릭터의 아래부분의 Rect와 전체 맵 타일의 Rect를 순회하며 충돌 체크    1. 맵에 블록이 있는 부분 중 충돌한 맵이 있으면 해당 맵의 위로 캐릭터 위치 고정 및 True 반환 2. 떨어지고 있는 테트리스 블록들의 Rect와 캐릭터의 아래부분의 Rect와 충돌체크    1. 충돌한 부분이 있으면 해당 인덱스 반환    2. 없으면 충돌한 부분이 없다는 정보 반환(음수 반환) | |

|  |  |
| --- | --- |
| 함수 명 | checkFallingTetris |
| 함수 전달인자 | topX, topY, faceX, faceY, legX, legY |
| 함수 역할 | 캐릭터와 떨어지고 있는 테트리스와 충돌 체크 |
| 1. 떨어지고 있는 테트리스 블록의 인덱스들과 인자로 입력 받은 인덱스가 같은지 비교    1. 같은 인덱스가 있으면 True반환    2. 없으면 False반환 | |

|  |  |
| --- | --- |
| 함수 명 | moveFez |
| 함수 전달인자 |  |
| 함수 역할 | 캐릭터의 좌우 이동 |
| 1. 캐릭터의 좌우 이동 flag가 True인지 검사    1. True이면 움직일 위치에 맵의 블록이나 테트리스의 블록이 있는지 검사    2. 움직일 수 있으면 캐릭터의 좌표 이동 | |

|  |  |
| --- | --- |
| 함수 명 | setFezPos |
| 함수 전달인자 | x, y, jump |
| 함수 역할 | 캐릭터의 위치 조정(테트리스 플레이어의 경우) |
| 1. 전달인자로 받은 x, y, jump 데이터를 캐릭터 정보 Dictionary의 데이터로 설정 | |

|  |  |
| --- | --- |
| 함수 명 | checkEnemyFez |
| 함수 전달인자 |  |
| 함수 역할 | enemy들과 캐릭터의 충돌 검사 |
| 1. Enemy 리스트에 저장된 enemy의 Rect와 캐릭터의 Rect의 충돌 검사   1.1 충돌하면 게임 종료 action 서버에 전송 | |

|  |  |
| --- | --- |
| 함수 명 | checkGameOver |
| 함수 전달인자 |  |
| 함수 역할 | 게임종료 체크 |
| 1. 캐릭터의 x좌표가 스크린 왼쪽으로 y좌표가 스크린 아래쪽으로 벗어나는지 체크   1.1 벗어나면 게임 종료 action 서버에 전송 | |

|  |  |
| --- | --- |
| 함수 명 | moveEnemy |
| 함수 전달인자 |  |
| 함수 역할 | Enemy 움직임 |
| 1. Enemy 리스트에 저장된 각 방향 체크    1. 방향에 따라 먼저 x좌표를 움직였을 때 블록이 있는지 체크       1. 블록이 있으면 방향 좌우 변환       2. 블록이 없으면 그 좌표로 이동    2. Enemy의 좌표가 스크린을 벗어났는지 체크   1.2.1 벗어난 Enemy정보 리스트에서 pop | |

|  |  |
| --- | --- |
| 함수 명 | fallEnemy |
| 함수 전달인자 |  |
| 함수 역할 | Enemy의 추락 |
| 1. 리스트에 저장된 Enemy들의 바닥 충돌 검사   1.1 바닥이 없으면 추락시켜 좌표 이동 | |

|  |  |
| --- | --- |
| 함수 명 | checkLRFallingTetris |
| 함수 전달인자 | ex, ey |
| 함수 역할 | 떨어지고 있는 테트리스 블록과 Enemy의 충돌 검사 |
| 1. 떨어지는 테트리스의 블록 인덱스를 맵의 인덱스로 변환 2. 테트리스 블록의 인덱스와 입력 받은 ex, ey(Enemy의 인덱스)와 겹치는지 검사    1. 겹치면 True 반환    2. 그 외 경우 False 반환 | |

|  |  |
| --- | --- |
| 함수 명 | collisionDownEnemy |
| 함수 전달인자 |  |
| 함수 역할 | Enemy의 바닥 충돌 검사 |
| 1. Enemy 리스트에 저장된 Enemy들의 픽셀 좌표를 맵의 인덱스로 변환 2. Enemy의 아래부분과 맵 블록 및 테트리스 블록이 충돌하는지 검사    1. 충돌하면 해당 Enemy의 fall 멤버변수 False로 reset 및 Enemy의 y좌표를 충돌한 부분으로 set    2. 충돌하지 않으면 해당 Enemy의 fall 멤버변수 True로 set | |

|  |  |
| --- | --- |
| 함수 명 | enemyImage |
| 함수 전달인자 |  |
| 함수 역할 | Enemy들의 이미지 애니메이션 |
| 1. Enemy들의 이미지 스프라이트 변환 | |

|  |  |
| --- | --- |
| 함수 명 | coinImage |
| 함수 전달인자 |  |
| 함수 역할 | Coin들의 이미지 애니메이션 |
| 1. Coin들의 이미지 스프라이트 변환 | |
| 함수 명 | coinCheck |
| 함수 전달인자 |  |
| 함수 역할 | 캐릭터와 coin의 충돌 체크 |
| 1. Coin들의 Rect중 캐릭터의 Rect와 충돌하는 게 있는 지 검사    1. 충돌하는 게 있으면 count변수 1 증가    2. 충돌한 Coin의 bPop 멤버 변수 True로 set(coinPop()함수에서 한꺼번에 pop) 2. Count 변수 반환 | |

|  |  |
| --- | --- |
| 함수 명 | fallFez |
| 함수 전달인자 |  |
| 함수 역할 | 페즈의 낙하 |
| 1. 페즈가 밟을 수 있는 땅이 없는 경우 Y좌표를 감소 2. 가속도를 주어 점점 빠르게 낙하 | |

|  |  |
| --- | --- |
| 함수 명 | jumpFez |
| 함수 전달인자 |  |
| 함수 역할 | 페즈의 점프 |
| 1. 점프가 가능한 상태일 때 페즈의 Y좌표를 증가 2. 최고점에 도달하거나 블록에 닿으면 가속도에 따라 낙하 | |

|  |  |
| --- | --- |
| 함수 명 | checkGameover |
| 함수 전달인자 |  |
| 함수 역할 | 게임오버 조건 체크 |
| 1. 페즈의 X좌표가 화면 왼쪽으로 밀릴 경우 2. 페즈의 Y좌표가 화면 아래쪽으로 떨어질 경우 게임오버 메시지를 서버로 전송 | |

4. renderProcess()

|  |  |
| --- | --- |
| 함수 명 | drawBackGround |
| 함수 전달인자 |  |
| 함수 역할 | 배경화면 출력 |
| 1. 배경 이미지 출력 2. 배경 이미지 마지막 부분에 이어서 이미지 출력 | |

|  |  |
| --- | --- |
| 함수 명 | drawBoard |
| 함수 전달인자 |  |
| 함수 역할 | 게임 틀 출력 |
| 1. 게임이 진행되는 화면 바깥쪽의 틀 출력 | |

|  |  |
| --- | --- |
| 함수 명 | drawBound |
| 함수 전달인자 |  |
| 함수 역할 | 틀 바깥쪽의 화면 가림 |
| 1. 게임화면 상하좌우에 검은색 사각형 출력 | |

|  |  |
| --- | --- |
| 함수 명 | drawBox |
| 함수 전달인자 | y, x, color |
| 함수 역할 | 타일 출력 |
| 1. 맵 정보를 받아와 각 x, y 좌표에 해당하는 타일 출력 | |

|  |  |
| --- | --- |
| 함수 명 | drawMovingTetris |
| 함수 전달인자 |  |
| 함수 역할 | 테트리스 블록 출력 |
| 1. 현재 player1이 조작하고 있는 테트리스 블록 출력 | |

|  |  |
| --- | --- |
| 함수 명 | drawFez |
| 함수 전달인자 |  |
| 함수 역할 | 페즈 캐릭터 출력 |
| 1. 현재 페즈 캐릭터의 위치값 및 정보 체크 2. 해당 위치에 페즈 캐릭터 출력 | |

|  |  |
| --- | --- |
| 함수 명 | drawEnemy |
| 함수 전달인자 |  |
| 함수 역할 | 몬스터 출력 |
| 1. 몬스터 위치값 및 정보 체크 2. 해당 위치에 몬스터 이미지 출력 | |

|  |  |
| --- | --- |
| 함수 명 | drawCoin |
| 함수 전달인자 |  |
| 함수 역할 | 동전 출력 |
| 1. 맵에 있는 코인 위치값 체크 2. 해당 위치에 코인 이미지 출력 | |

|  |  |
| --- | --- |
| 함수 명 | drawScore |
| 함수 전달인자 |  |
| 함수 역할 | 점수 출력 |
| 1. 실시간으로 점수를 받아옴 2. 받아온 점수 출력 | |

|  |  |
| --- | --- |
| 함수 명 | imgSprite |
| 함수 전달인자 |  |
| 함수 역할 | 페즈 이미지 변환 |
| 1. 페즈 캐릭터의 이동상태 정보에 따라 해당하는 스프라이트를 변환 2. 결과적으로 자연스럽게 애니메이션이 적용되어 출력됨 | |

1. FrozenServer()

|  |  |
| --- | --- |
| 함수 명 | \_\_init\_\_ |
| 함수 전달인자 | host IpAddress, port Num |
| 함수 역할 | Server객체 생성 및 Accept |
| 1. 떨어지고 있는 테트리스 블록의 각 블록들의 도착지점을 체크 2. 적당한 위치에 테트리스 블록들을 맵으로 변환 후 return | |

|  |  |
| --- | --- |
| 함수 명 | Connected |
| 함수 전달인자 | channel |
| 함수 역할 | 클라이언트 채널정보를 저장 |
| 1. 입력된 채널정보를 변수에 저장한다. 2. 만약 기존에 입력된 채널정보가 있다면 이미 한명이 들어와있었다는 것이므로 채널정보를 저장하고 두 유저에게 모든 유저가 입장하였다고 전송 3. 만약 두명이상이 이미 접속해있었다면 overflow메시지 전송 | |

|  |  |
| --- | --- |
| 함수 명 | DelPlayer |
| 함수 전달인자 | player, id |
| 함수 역할 | 접속 해제한 player의 채널을 지운다. |
| 1. 접속 해제한 player의 정보를 보고 해당되는 채널을 지운다. 2. 만약 두명이 있던 가운데, 한명이 나간것이라면 남아있는 유저에게 누군가 나갔음을 전송한다. | |

|  |  |
| --- | --- |
| 함수 명 | GameStart |
| 함수 전달인자 | stage |
| 함수 역할 | 게임이 시작하였음을 알린다 |
| 1. 방장이 아닌 다른 플레이어에게 게임이 시작하였음을 알려준다. 2. 어떠한 난이도를 선택하였는지 같이 전송한다. | |

|  |  |
| --- | --- |
| 함수 명 | FezMove |
| 함수 전달인자 | move, turn |
| 함수 역할 | 받은 캐릭터의 움직임을 블록 컨트롤 유저에게 전달 |
| 1. 캐릭터 그래픽 및 애니메이션 구현에 사용되는 캐릭터 움직임 정보를 받는다. 2. 이를 블록을 컨트롤하는 유저에게 전달한다. | |

|  |  |
| --- | --- |
| 함수 명 | FezPos |
| 함수 전달인자 | x, y, jump |
| 함수 역할 | 받은 캐릭터의 Position을 블록 컨트롤 유저에게 전달 |
| 1. 캐릭터 동기화를 위한 캐릭터 위치 정보를 받는다. 2. 이를 블록을 컨트롤하는 유저에게 전달한다. | |

|  |  |
| --- | --- |
| 함수 명 | GameOver |
| 함수 전달인자 | score |
| 함수 역할 | GameOver되었다는 사실을 모든 유저에게 전달 및 점수 전달. |
| 1. GameOver되었으므로 모든 유저에게 이를 알린다. 2. 이와 함께 유저의 Score정보를 전달한다. | |

|  |  |
| --- | --- |
| 함수 명 | NewTetris |
| 함수 전달인자 | shape, color |
| 함수 역할 | 새롭게 생성된 블록정보를 받아서 전달. |
| 1. 새롭게 생성된 블록정보를 받는다. 2. 받은 블록정보를 캐릭터를 조종하는 유저에게 전달한다. | |

|  |  |
| --- | --- |
| 함수 명 | TetrisMove |
| 함수 전달인자 | act, what, value |
| 함수 역할 | 블록을 움직인 정보를 캐릭터 조종 유저에게 전달 |
| 1. 블록을 어떻게 얼마나 무엇으로 움직였는지에 대한 정보를 받는다. 2. 이를 캐릭터를 조종하는 유저에게 전달한다. | |

|  |  |
| --- | --- |
| 함수 명 | BlockOnMap |
| 함수 전달인자 | x, y |
| 함수 역할 | 맵이 된 블록을 캐릭터 조종 유저에게 전달한다. |
| 1. 바닥에 닿아 맵으로 변환될 블록이 어디에 저장되었는지 받는다. 2. 이를 캐릭터를 조종하는 유저에게 전달하여 캐릭터가 밟고 올라설 수 있도록 한다. | |

|  |  |
| --- | --- |
| 함수 명 | MoveComponents |
| 함수 전달인자 |  |
| 함수 역할 | 우측으로 화면이 이동되는것에 대한 정보를 보낸다. |
| 1. 우측으로 화면이 이동되고 있다고 유저에게 정보를 보낸다. | |

1. 외부 기능

|  |  |
| --- | --- |
| 함수 명 | Title |
| 함수 전달인자 |  |
| 함수 역할 | 게임 타이틀 화면 출력 |
| 1. 게임 타이틀 화면 출력 및 구름 이동 2. 게임시작 및 크레딧, 게임 종료 버튼 구현, 각 버튼 클릭시 각 기능 실행 | |

|  |  |
| --- | --- |
| 함수 명 | Credit |
| 함수 전달인자 |  |
| 함수 역할 | 크레딧 화면 출력 |
| 1. 크레딧 이미지 출력 2. 화면 클릭 시 타이틀로 이동 | |

|  |  |
| --- | --- |
| 함수 명 | Gameover |
| 함수 전달인자 |  |
| 함수 역할 | 게임오버 시 처리 |
| 1. 게임 오버 이미지 출력 2. 게임 끝나는 시점의 점수 출력 3. 약 2.9초간 배경음악이 나오고 타이틀 화면으로 이동 | |

**4. 구현 결과 및 테스팅**

**4.1 실제 기능별 구현 내용**

**클라이언트**

**맵 데이터로부터 값을 읽어 들여 맵 생성**

initMap 함수에 START\_MAP[fez['stage']] 전달인자로 txt를 불러와 m\_Map[i][j]에 저장한다.

**캐릭터와 맵 간의 충돌체크**

collisionUp, collisionBlockDown 함수에 pygame.Rect로 캐릭터 크기의 직사각형 을 만들어 충돌체크한다.

**캐릭터와 블록 간의 충돌체크**

checkFallingTetrisUp, checkFallingTetrisDown 함수로부터 리턴 값으로 충돌 여부 를 알 수 있다.

**아이템 획득**

맵 상 coinImage 함수로 코인을 출력하고 coinCheck 함수로 캐릭터 직사각형과 코인이 충돌했는지 체크하고 coinPop 함수로 획득한 코인을 m\_Coin 배열에서 삭제한다.

**스테이지별 난이도 설정**

fez['stage'] 변수가 0이면 쉬운 모드, 1이면 고급 모드로 설정된다.

**스테이지 목표**

drawScore 함수로 점수를 계속해서 출력한다.

gameover 함수로 게임오버 시 획득한 최종 점수를 출력한다.

dataProcess 함수에서 시간에 따른 점수, 아이템 획득에 따른 점수를 누적시킨다.

**게임 오버**

checkEnemyFez 함수로 캐릭터와 적이 충돌했는지 검사해서 게임 오버시킨다.

checkGameOver 함수로 캐릭터가 맵에서 밀려나는 경우와 맵에서 떨어지는 경우 게임 오버시킨다.

**네트워크(2인용)**

**서버 생성**

Player 1, Player 2 모두 클라이언트로 서버에 접속

서버의 Connected 함수에서 currentIdx를 통해 서버에 몇 명이 접속해있는지 알 수 있다.

포트 번호와 클라이언트의 소켓 및 채널 획득

클라이언트에서 host, port번호를 직접 입력하여 연결할 server정보를 넘겨준다.

게임 생성

첫번째 플레이어가 접속하면 Game 오브젝트가 생성되며, 두번째 플레이어가 접속하면 GameStart버튼이 활성화된다.

**데이터 전송**

Player 1 캐릭터의 위치를 서버로 전송

Sendserver 함수에서 action값을 fezMove로 전송해주면 해당 위치의 x,y좌표를 서버의 Network\_fezPos에서 저장

Player 2 블록의 위치를 서버로 전송

Sendserver 함수에서 action값을 TetrisMove로 전송해주면 해당 블록의 우치, 모양 및 회전 여부를 서버의 Network\_tetrisMove에서 저장

상대 Player에게 데이터 전송

서버에서 FezPos 혹은 TetrisMove 함수를 호출하여 상대 플레이어에게 fezPos 값과 moveTetris값을 전달하여 클라이언트의 DataProcess에서 처리

**동기화**

상대 Player의 데이터를 적용

FezMove, FezPos 함수로 현재 캐릭터의 움직임을 동기화한다.

NewTetris, TetrisMove 함수로 현재 블록의 움직임을 동기화한다.

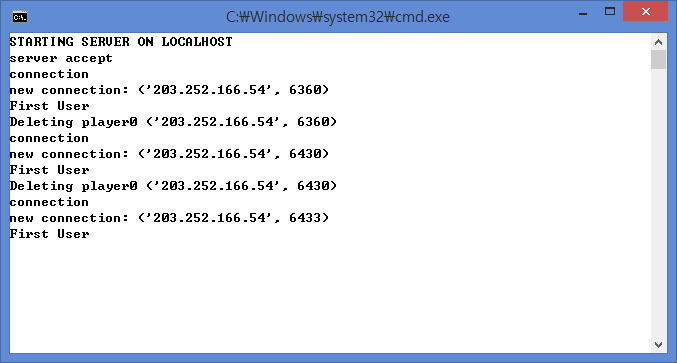
DelPlayer 함수로 사용자가 나갔을 때 변수를 초기화한다.

맵 상황 동기화

BlockOnMap 함수로 블록이 맵에 고정된 상황을 동기화한다.

MoveComponents 함수로 맵 이동 상황을 동기화한다.

**4.2 테스팅 결과 (실행화면)**



**< 서버 실행 화면 >**

서버 실행 시 'server accept' 라는 문구와 함께 서버가 준비되었음을 알려준다.

처음 접속한 유저의 ip 주소가 로그로 출력되고 'First User'로 선정한다.

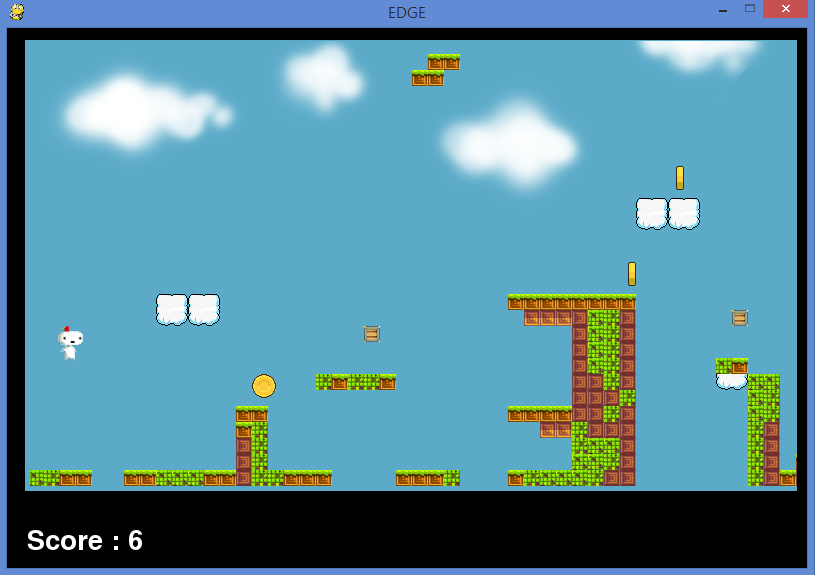
player가 접속을 해제하면 'Deleting player' 정보를 전달받는다.



**< 클라이언트 실행 화면 (타이틀) >**

Player 접속 상황에 따라 1, 2 접속 상태가 초록색 원, 회색 원으로 출력된다.

마우스 클릭으로 메뉴 선택 가능하다.



**< Player 1 화면 >**

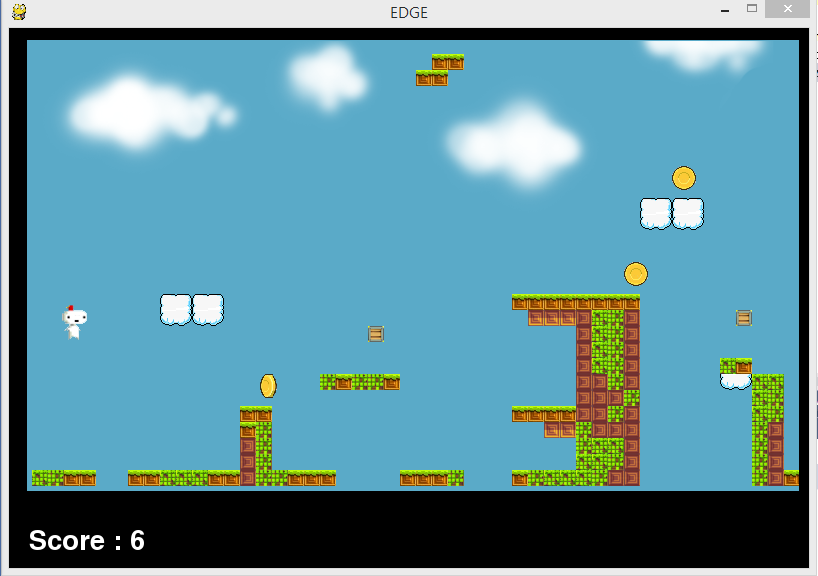
Player 1은 런 플레이어(캐릭터)가 된다.

Player 1은 w,a,d 키를 이용하여 캐릭터 이동이 가능하다.

캐릭터가 움직이는 맵 밖으로 밀려나거나, 맵 구멍에 빠지거나, 적에게 부딪히면 게임오버가 된다.

캐릭터가 오래 살아남아 진행할 수록 높은 점수를 받는다.

화면은 Player 2와 소켓 통신을 통해 동일한 화면을 볼 수 있다.



**< Player 2 화면 >**

Player 2는 협동 플레이어(블록)가 된다.

Player 2는 방향키와 space bar를 이용하여 블록을 이동하거나 땅으로 블록을 한번에 내린다.

캐릭터가 오래 살아남을 수 있도록 협동을 한다.

화면은 Player 1과 소켓 통신을 통해 동일한 화면을 볼 수 있다.

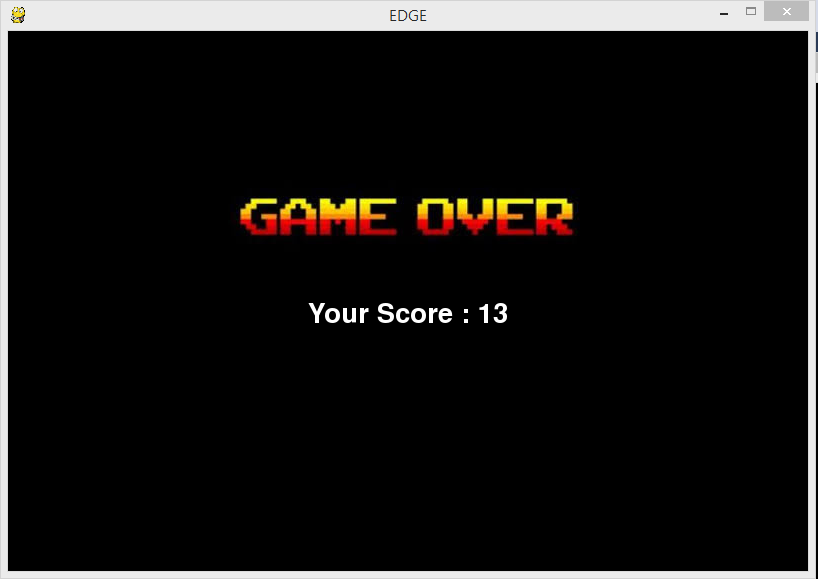


**< 게임 진행 화면 >**

게임 시작 시 맵은 계속해서 왼쪽으로 이동한다.

동전을 습득하면 Score 100을 획득하며 살아남아 있을 수록 시간에 따른 Score를 획득한다.

캐릭터가 게임오버 되는 조건에 의해 게임오버가 되면 타이틀 화면으로 돌아간다.



**< 게임 오버 화면 >**

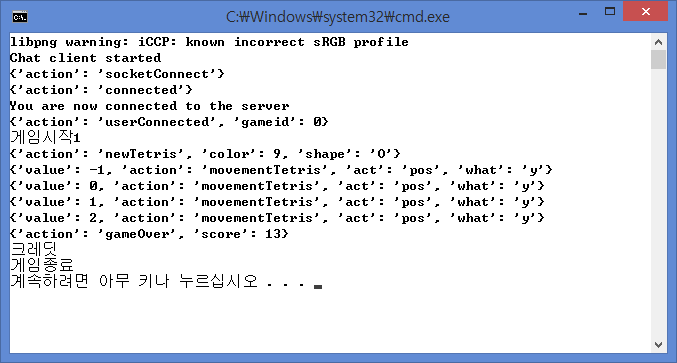
캐릭터가 게임오버 되면 출력되는 화면이다.

이 화면과 함께 재생되는 게임 오버 음악이 끝나면 타이틀 화면으로 돌아간다.



**< 크레딧 화면 >**

타이틀 화면에서 크레딧을 클릭하면 출력되는 화면이다.



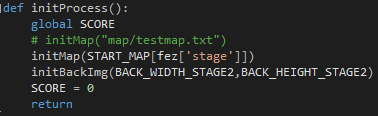
**< 클라이언트 로그 화면 >**

클라이언트가 어떤 행동을 하는 지에 따른 로그가 출력된다.

**5. 부록**

**5.1 소스코드 핵심 부분**

InitProcess()



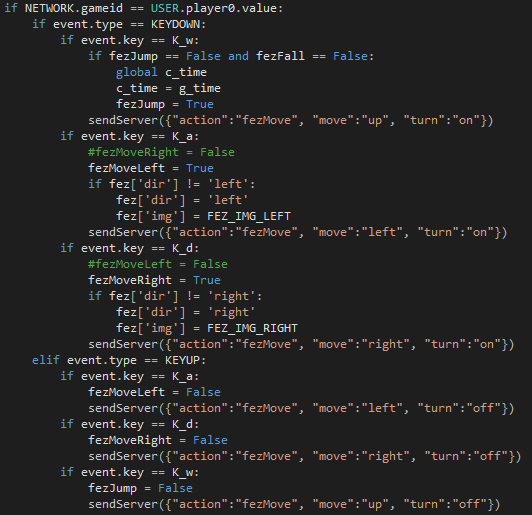
게임 시작 시 실행

1. 선택 스테이지에 따라 맵 생성 및 배경이미지 생성
2. 점수 0점으로 초기화

inputProcess()



게임 시작 후 키 입력을 실시간으로 처리



1. 먼저 접속한 플레이어일 경우(player0)

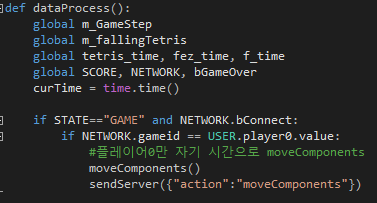
* 키보드 버튼이 눌리는 경우에 이벤트 호출
* w버튼 누를 시 캐릭터 점프
* a,d버튼 누를 시 캐릭터 좌우 이동
* a,d,w버튼 뗄 시 캐릭터 정지



1. 나중에 접속한 플레이어일 경우(player1)

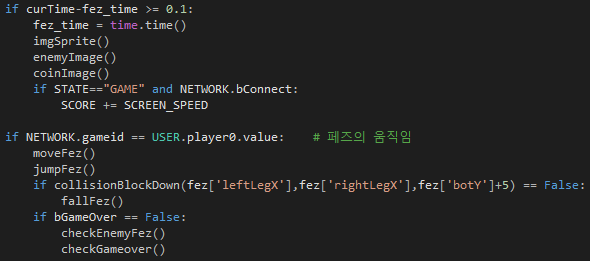
* 키보드 버튼이 눌리는 경우에 이벤트 호출
* 위쪽 방향키 누를 시 테트리스 블록 회전
* 좌우 방향키 누를 시 테트리스 블록 이동
* 아래 방향키 누를 시 테트리스 블록 한칸 아래로 이동
* 스페이스바 누를 시 테트리스 블록을 맵 가장 아래로 붙임

dataProcess()



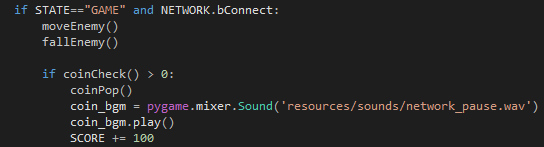
게임 실행 중 실시간으로 데이터를 처리

1. 시작 시, player0에서 전체 화면을 왼쪽으로 이동시키고 서버로 메시지 전송



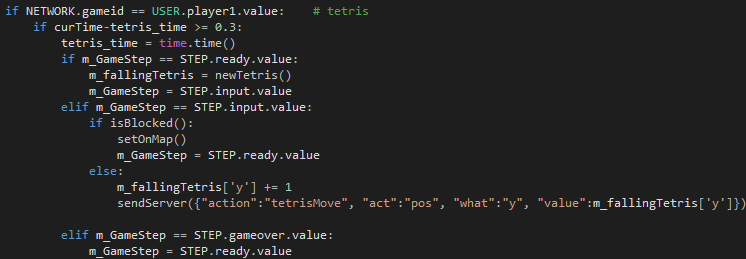
1. Player0의 경우

* 스크린 이동 횟수에 따라 점수를 증가
* 캐릭터 이동 시 캐릭터의 좌표값 처리
* 아래쪽에 블록이 없을 경우 캐릭터는 계속해서 하강
* 게임오버가 아니라면 게임오버 조건 체크



1. 게임중 몬스터 처리

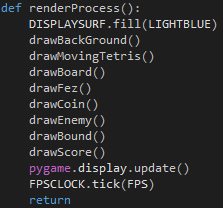
* 몬스터의 위치값 계속해서 이동
* 코인 획득 시 코인획득 bgm 재생 및 점수 100점 추가



1. Player1의 경우

* 테트리스 블록이 바닥에 떨어질 경우 맵으로 세팅
* 테트리스블록의 위치 실시간으로 서버로 전송

renderProcess()



데이터 처리 이후 처리된 데이터에 따라 모든 오브젝트 새로 그려줌

1. 하늘색 배경 처리
2. 배경이미지 출력
3. 테트리스 이동 가능 범위 출력
4. 게임 범위 출력
5. 캐릭터 출력
6. 코인 출력
7. 몬스터 출력
8. 프레임 출력
9. 점수 출력
10. 화면 업데이트

**5.2 프로그램 실행하기**

* 실행하기 전, 셋팅 및 설치 *(Python 3.4 버전 기준으로 작성되었습니다)*

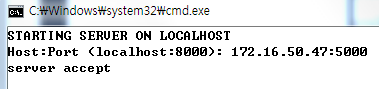
**Python 3.4** https://www.python.org/downloads/release/python-343/

**Pygame** http://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame

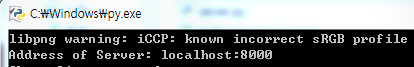
*“pygame‑1.9.2a0‑cp34‑none‑win32.whl"* 파일 **다운로드**

cmd창 /**python -m pip install pygame‑1.9.2a0‑cp34‑none‑win32.whl**

* 서버 실행하기
* ProjectFrozen\프로그래밍\FrozenTetris\PythonFrozen1 폴더 안의 server.py 실행
* 자신의IP**:**포트번호를 입력. 아무것도 입력하지 않고 엔터치면 자동으로 localhost:8000



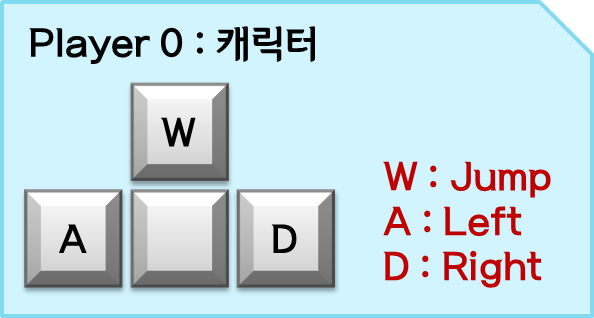
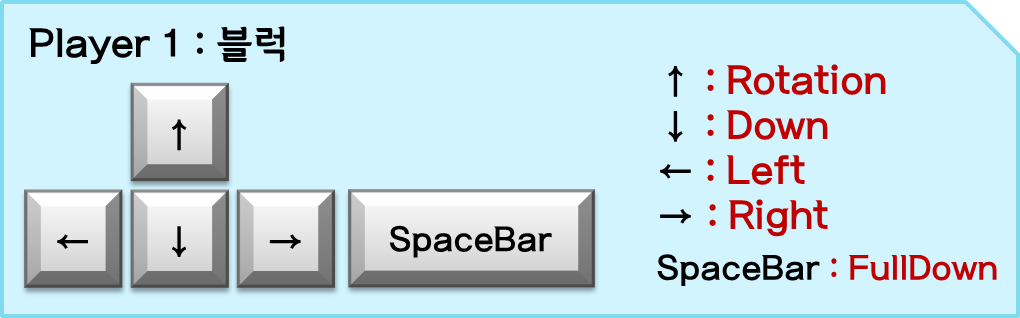
* 클라이언트 실행하기
* 서버의 IP:포트번호 입력. 아무것도 입력하지 않고 엔터치면 자동으로 localhost:8000



게임 플레이하기



게임 플레이하기 - 조작법

**5.3 사용 라이브러리 및 참고문헌**

|  |  |  |
| --- | --- | --- |
| **Pygame** | 윈도우기반 그래픽 출력 및 소리 출력 | http://www.pygame.org/hifi.html |
| **Pygame~PodSixNet** | 소켓통신기반 네트워크 서버 | http://pygame.org/project-PodSixNet-1069-.html  https://github.com/dexgecko/podsixnet\_python3.4.1 |

**5.4 프로젝트 참여현황 및 일정**

|  |  |
| --- | --- |
| 이름 | 프로젝트 참여 요소 |
| 김보하 | 배경, 스테이지, 타이틀화면 디자인, 네트워크, 소켓통신 구현 |
| 김진솔 | 프로토 타입, 테트리스, 화면 이동, 클라이언트, 버그 수정 |
| 이동호 | 스프라이트 출력, 점수UI, 소리, 클라이언트, 타이틀 |
| 이상민 | 캐릭터, 맵 디자인, 적 디자인, 발표, 버그 수정, 리소스 |