

2017년도 2학기

네트워크 게임 프로그래밍

< Term Project 추진 계획서 >

PM 2012181024 윤정훈

2013184002 김균환

2014184015 박지희

INDEX

1. 게임기획

- a. 게임소개
- b. 조작법
- c. 게임설정

2. 게임구조

- a. High - Level 디자인
- b. Low - Level 디자인

3. 개발계획

- a. 개발환경
- b. 팀원 별 역할분담
- c. 개발일정

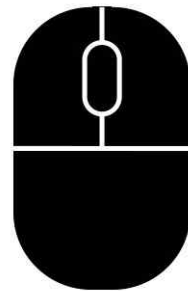
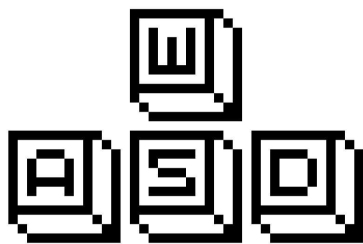
1. 게임기획

a. 게임 소개

- 게임 이름
: BreakBlocks
- 게임 장르
: 2D 종스크롤 슈팅게임
- 상세설명
: OpenGL을 이용한 2D 종스크롤 슈팅게임으로 최대 3인 플레이를 지원합니다. 각 플레이어들은 각자의 비행기 객체를 가지며 플레이어의 입력키에 따라 상하좌우를 이동하며 총알을 발사합니다. 화면상에서 장애물이 끝없이 내려오며 시간이 지남에 따라 게임 진행속도가 빨라집니다. 플레이어는 장애물과 충돌할 때 죽게 되고 모든 플레이어가 사망하면 게임이 종료됩니다. 장애물 충돌을 피하고 최대한 오랜 시간동안 살아남는 것이 게임의 목적입니다.

b. 조작법




- 플레이어의 조작은 기본적으로 키보드와 마우스를 이용합니다.
상세한 조작법은 아래와 같습니다.



키	설명
WASD	플레이어 이동
마우스 왼쪽 클릭	총알 발사

c. 게임설정

· 플레이어

	플레이어 1	플레이어 2	플레이어 3
이미지			
초기위치	($-X_{Max}/2$, $-Y_{Max}/10$)	(0 , $-Y_{Max}/10$)	($X_{Max}/2$, $-Y_{Max}/10$)
설명	각 플레이어의 스테이터스는 같으며 외형의 RGB값에만 차이를 갖습니다.		

- 플레이어의 스테이터스는 공격력(총알의 강화단계)만 존재합니다.
- 플레이어는 블록과 충돌하였을 시 즉시 사망합니다.

· 블록

	블록 1	블록 2	블록 3	블록 4	블록 5
이미지					
HP	1	2	5	10	15

- 블록은 화면에 최대 가로 5개 세로 7개가 존재합니다.
- 블록은 플레이어의 총알을 맞을 때마다 색깔이 벌어지고 HP가 0이 되면 삭제됩니다.
- 블록배치에 대한 정보는 파일에 따로 저장하여 게임 시작 시 클라이언트에서 불러들입니다.

· 총알

	총알 1단계	총알 2단계	총알 3단계
이미지			
공격력	1	2	3

- 총알 강화는 아이템에 의하여 이루어지며 최대 3단계까지 강화가 가능합니다.
- 총알이 강화 될 때마다 공격력이 1씩 증가됩니다.
- 왼쪽 클릭 시 플레이어를 기준으로 총알이 생성되며 총알이 블록에 맞거나 화면을 벗어나면 객체가 삭제됩니다.

· 아이템

	공격력UP	총알UP
이미지		
효과	공격력 증가	발사 총알 수 증가

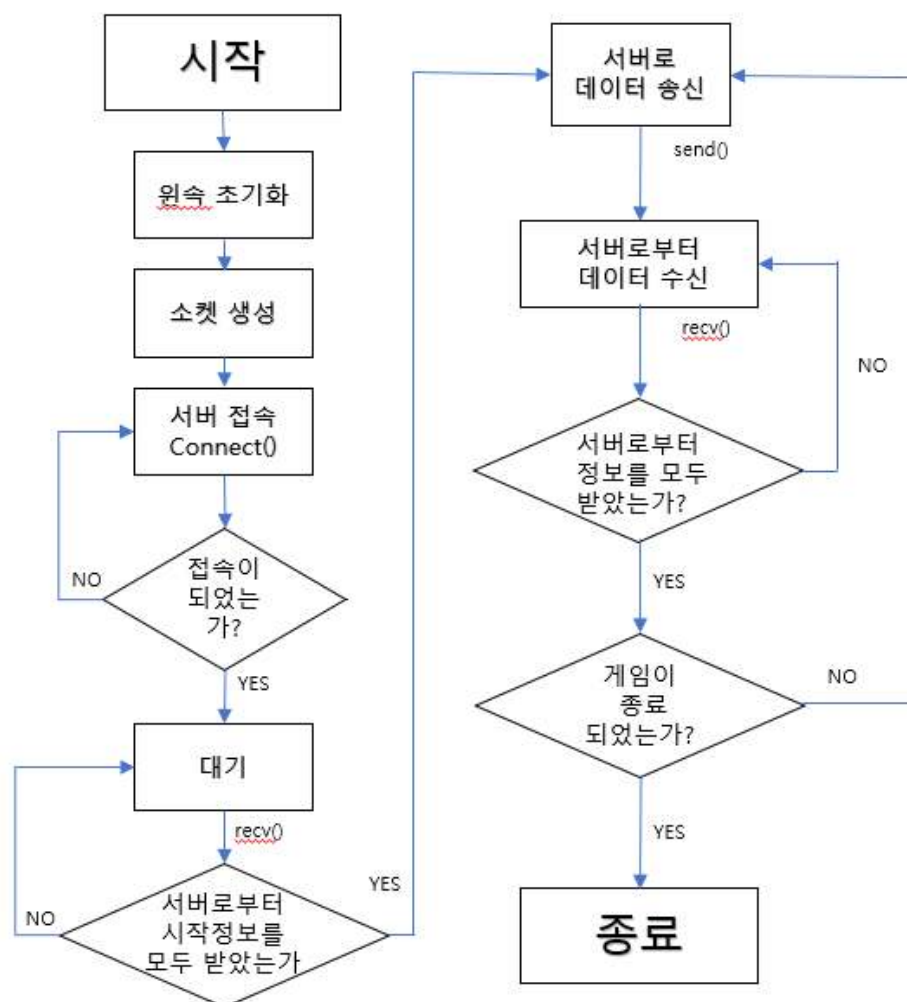
- 아이템은 총 2가지 종류가 존재하며 블록을 부셨을 때 일정 확률로 출현합니다.
- 아이템은 출현과 동시에 블록과 같은 속도로 떨어지며 플레이어와 충돌하였을 시 아이템 종류에 따른 이로운 효과를 획득 플레이어에게 줍니다.
- 한번 획득한 효과는 해당 게임이 플레이어가 사망할 때까지 지속됩니다.

2. 게임구조

a. High - level 디자인

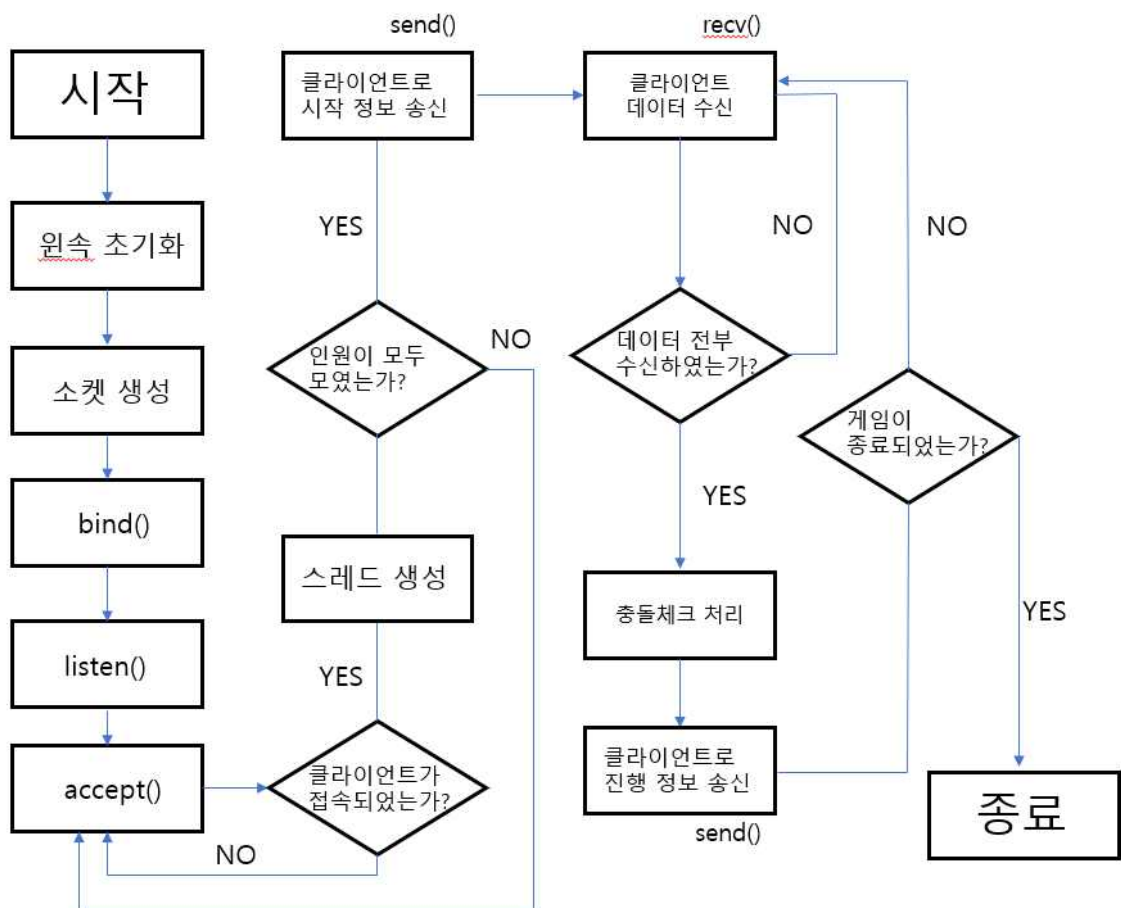
· Client Flowchart

- 클라이언트 프로그램의 흐름도는 아래와 같습니다.



· Server Flowchart

- 서버 프로그램의 흐름도는 아래와 같습니다.



b. Low - level 디자인

- 프로토콜 정의

```
class StartProtocol
```

```
// 시작할 때 서버에서 넘겨줄 패킷 구조체
```

```
int PlayerNumber      : 각 플레이어를 구분하는 변수  
Vec2 P1Pos;           : 플레이어1의 시작위치  
Vec2 P2Pos;           : 플레이어2의 시작위치  
Vec2 P3Pos;           : 플레이어3의 시작위치
```

```
class ServerMainProtocol
```

```
// 서버에서 클라이언트로 넘겨줄 구조체
```

```
int GameStateInfo      : 현재 게임의 상황을 알려주는 변수  
int BulletCnt          : 화면의 총알 개수  
int BlockCnt           : 화면의 블록 개수  
int ItemCnt            : 화면의 아이템 개수  
Player PlayerInfo[3]   : 플레이어의 정보  
Bullet BulletInfo[BulletCnt] : 현재 화면에 그려지는 총알의 정보  
Block BlockInfo[BlockCnt] : 현재 화면에 그려지는 블록의 정보  
Item ItemInfo[ItemCnt] : 현재 화면에 그려지는 아이템의 정보
```

```
class ClientMainProtocol
```

```
// 클라이언트에서 서버로 넘겨줄 구조체
```

```
int Type               : 플레이어를 구분하는 정보  
int PlayerType        : 총알을 구분하는 정보  
int BulletCnt          : 총알의 개수  
Player PlayerInfo     : 플레이어의 정보  
Bullet BulletInfo[BulletCnt] : 총알의 정보
```

```
class Object
```

```
// Player, Bullet, Block, Item 을 자식 클래스로 가지는 부모 클래스
```

```
int Type               : 객체들의 타입  
int Size               : 객체의 크기  
bool IsDead            : 객체의 존재유무  
Vec2 Pos               : 객체들의 좌표
```



```
class Player
    int Type :: Object           : 플레이어의 번호
    int Power                   : 플레이어의 공격력(총알 타입)
```

```
class Bullet
  int Tpye :: Object      : 총알의 특성(강화 단계)
  int PlayerBullet        : 총알을 구분하는 정보
```

```
class Item
    int Tpye :: Object          : 아이템의 특성
```

```
class Block
    int Type :: Object           : 블록의 특성
```

```
class Vec2
    int x, y
```

: x, y의 좌표

////////////////////////////////////

(1) High-Level 디자인

1. 프로토콜의 종류

- class CSceneMgr
- class CObject
- struct ObjInfo

(2) Low-Level 디자인

1. 프로토콜

```
enum ObjectType
    = { E_PLAYER, E_BULLET, E_BLOCK, E_ITEM, E_OBJ_END };
: 오브젝트의 종류

enum ePlayerType = { E_PLAYER_1, E_PLAYER_2, E_PLAYER3 };
: 플레이어의 번호

struct InitProtocol                                // accept()시 서버->클라
    ePlayerType eType;                            // 각 플레이어에게 할당될 번호
    vec2 vInitPos;                                // 시작 위치

struct ObjInfo
    vec2 vPos;                                    // 위치
    int iValue                                    // 각 오브젝트가 가지는 특성
    ObjectType eObjType                          // 오브젝트 종류

struct ObjInfo_from_Client                        // 클라 -> 서버
    ObjInfo tInfo                                // 오브젝트 정보
    ePlayerType ePlayerType; // 할당된 번호
```

```

***수정***struct ObjInfo_from_Seiver          // 서버 -> 클라
            ObjInfo tInfo                      // 오브젝트
struct vec2
            float fX;
            float fY;
struct vec3
            float fX;
            float fY;
            float fZ;
struct vec4
            float fX;
            float fY;
            float fZ;
            float fW;

```

2. 클라이언트

class CSceneMgr : 스테이트(state)를 관리하는 싱글톤 클래스. CScene 변수를 통해 현재 실행되는 스테이트의 갱신하고 그린다.

```

enum eSceneType
    = {E_TITLESCEINE, E_READYSCENE, E_GAMESECEINE}
CSceneMgr* GetInstance();
    처음 호출 시 멤버변수 m_pInstance == NULL 이라면 인스턴트
    객체를 생성하는 함수.
DestroyInstance();
    게임 종료시 객체를 삭제해 주는 함수.
Update();
    게임 내의 모든 오브젝트들을 갱신 시키는 함수.
Draw();
    게임 내의 모든 오브젝트들을 그리는 함수.
Change_Scene(eSceneType eType)
    각 씬에서 Change_Scene()이 호출되면 들어온 매개 변수에 의해
    씬이 변환된다.

CSceneMgr* m_pInstance = NULL;
    싱글톤으로 사용하기위해 자기 자신을 포인터로 가지고 있는 변수.
eSceneType m_eCurSceneType;
    현재 실행되고 있는 씬의 타입을 저장하는 변수.
CScene* m_pScene;
    현재 실행되고 있는 씬을 저장하는 포인터 변수.
list<CGameObject*> m_pObjList[E_OBJ_END];
    씬에 존재하는 객체들의 리스트

```

class CScene : 모든 씬의 부모가 되는 클래스

virtual void Initialize() = 0;

virtual void Update() = 0;

virtual void Draw() = 0;

class CWaitScene : public CScene

다른 플레이어를 기다리는 씬의 정보를 가지고 있는 클래스

virtual void Initialize();

connect() 호출하는 함수

virtual void Update();

다른 플레이어가 접속을 할 때까지 대기하고 있다. 모두 접속했으면 키 입력을 통해 CSceneMgr::Change_Scene()을 호출한다.

class CGameScene : public CScene

실제 게임이 진행되는 씬의 정보를 가지고 있는 클래스

virtual void Initialize();

맵의 블록과 다른 플레이어의 초기 정보를 초기화하는 함수.

virtual void Update();

실제 게임이 진행되는 함수. 키 입력을 통해 움직임,

class CObjMgr : 게임 내의 오브젝트들을 관리하는 클래스. 싱글톤으로 만들어 사용한다.

Initialize()

list를 초기화 해주는 함수.

Update()

list안의 객체들의 Update()를 호출한다

Draw()

list안의 객체들의 Draw()를 호출하여 그려주는 함수.

Release()

객체가 삭제될 때 list안의 객체를 모두 삭제시킨다.

list<CGameObject*> m_pObjList[E_OBJ_END];

오브젝트의 종류에 따라 나누어 관리하는 리스트.

class CGameObject : 모든 오브젝트의 부모가 되는 클래스

virtual void Initialize() = 0;

virtual void Update() = 0;

virtual void Draw() = 0;

ObjInfo tInfo; // 객체의 정보를 담고있는 변수.

```

class CPlayer : public CGameObject
    플레이어의 정보를 가지는 클래스.
    virtual Initialize();
        서버에서 받아온 정보(ePlayerType, e)
    virtual Update();
        키 입력을 통해 플레이어를 움직이는 함수.
        키 입력이 들어왔을 때 서버에 플레이어 정보를 send()
    virtual Draw();
    vec4 Get_BoundingBox();
        플레이어의 위치와 사이즈를 이용해
        vec4(minX, minY, maxX, maxY)로 반환해준다.

        vec2 m_vSize          // 플레이어의 사이즈
////////////////////////////////////

ServerStart()
//서버가 시작될때 실행되는 함수. 소켓의 생성과 바인딩을 해준다.

    원속 초기화
    SOCKET listen_socket 생성
    바인드
    클라이언트가 알맞게 접속했는지 확인
    스레드를 생성하고 스레드 매니저도 생성.
    게임을 초기화.

StartThread()
// 스레드를 생성하고 스레드 매니저를 생성해 관리할 수 있게끔 한다.

class ServerObjectMng
// 서버에서 처리해야할 오브젝트를 관리하는 클래스

    list<Player> PlayerList          : 플레이어를 관리하는 리스트
    list<Bullet> PlayerList          : 총알을 관리하는 리스트
    list<Block> PlayerList           : 블럭을 관리하는 리스트
    list<Item> PlayerList            : 플레이어를 관리하는 리스트
    list<Player> PlayerList          : 플레이어를 관리하는 리스트

    CollisionPlayerBullet()
    CollisionPlayerBlock()
    CollisionBlockBullet()

```

```
class ThreadMng
```

```
// 공유 자원 관리와 소켓관리를 위한 클래스
```

```
    SOCKET clientSocket[3]          : 클라이언트 소켓을 관리
```

```
// 동기화되어 관리될 자원들
```

```
    list<Player> PlayerList
```

```
    list<Bullet> BulletList
```

```
    list<Block> BlockList
```

```
    list<Item> ItemList
```

```
    list<Player> PlayerList
```

```
    ServerObjectMng ObMng
```

```
: 오브젝트간의 충돌을 검사할 오브젝트 매니저를 생성
```

```
    update()
```

```
: 오브젝트 매니저 내의 충돌을 호출하며 충돌검사를 실행
```

2. Server

3. 개발계획

a. 개발환경

	클라이언트	서버
IDE	MS Visual Studio 2017	
OS	Windows 10	
Language	Visual C++	
Feature	OpenGL	TCP

b. 팀원 별 역할분담

윤정훈	김균환	박지희

c. 개발일정

· 11월

	윤정훈		김균환		박지희	
	개발내용	비고	개발내용	비고	개발내용	비고
11일(토)						
12일(일)						
13일(월)						
14일(화)						
15일(수)						
16일(목)						
17일(금)						

18일(토)						
19일(일)						
20일(월)						
21일(화)						
22일(수)						
23일(목)						
24일(금)						
25일(토)						
26일(일)						
27일(월)						
28일(화)						
29일(수)						
30일(목)						

c. 개발일정

· 12월

	윤정훈		김균환		박지희	
	개발내용	비고	개발내용	비고	개발내용	비고
1일(토)						
2일(일)						
3일(월)						
4일(화)						
5일(수)						
6일(목)						
7일(금)						

8일(토)						
9일(일)						
10일(월)						
11일(화)						
12일(수)						
13일(목)						
14일(금)						