

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

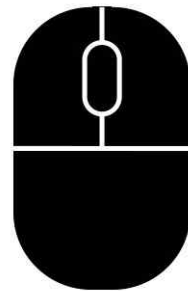
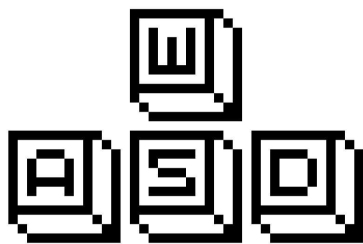
1. 게임기획

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. 플레이어 간 협력을 통하여 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

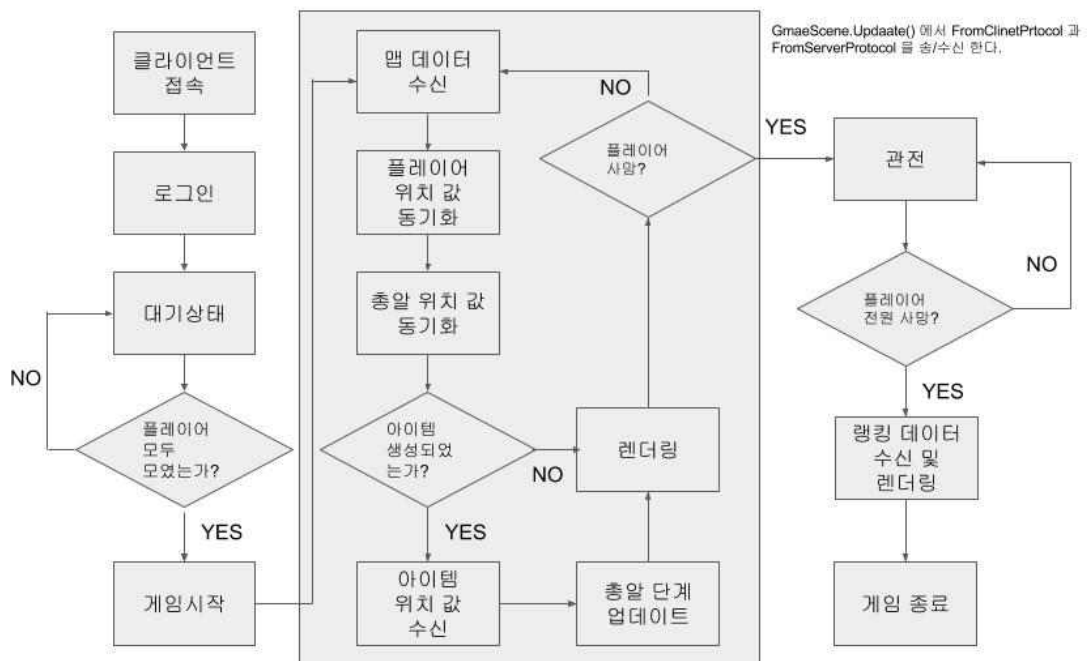
- 아이템은 총 2가지 종류가 존재하며 블록을 부셨을 때 일정 확률로 출현합니다.
- 아이템은 출현과 동시에 블록과 같은 속도로 떨어지며 플레이어와 충돌하였을 시 아이템 종류에 따른 이로운 효과를 획득 플레이어에게 줍니다.
- 한번 획득한 효과는 해당 게임이 플레이어가 사망할 때까지 지속됩니다.

2. 게임구조

a. High - level 디자인

· Client Flowchart

- 클라이언트 프로그램의 흐름도는 아래와 같습니다.



로그인 : Player 의 ID를 정해준다. 로그인 정보를 FromClientProtocol의 PlayerInfo에 담아 서버로 송신한다. WaitScene.Initialize()에서 처리한다.

대기상태 : WaitScene.Update()에서 처리한다. recv()를 호출하는 상태이다.

게임시작 : recv() 에 FromServerProtocol을 받으면 게임시작.

GameScene.Update() 에서의 송/수신 :

*수신 : FromServerProtocol에 포함된 내용을 수신한다.

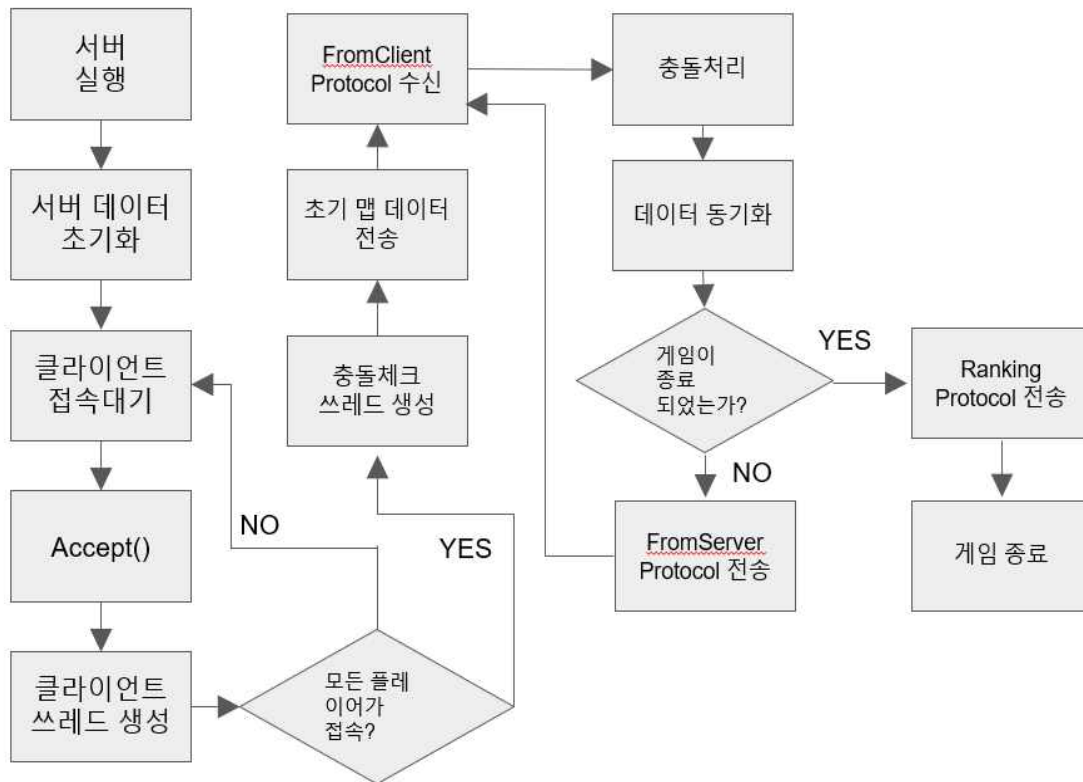
*동기화 : FromClientProtocol 과 FromServerProtocol에서 각각 송/수신 하며 값을 동기화 해준다.

렌더링 : 서버의 CollisionObject() 충돌체 체크 쓰레드에서 처리한 값까지 모든 데이터 업데이트를 완료하면 프레임의 끝에서 최종적으로 렌더링한다.

랭킹 데이터 수신 및 렌더링 : 서버로부터 RankingProtocol을 수신 받아 결과 창에 플레이어의 아이디와 시간을 표시한다.

· Server Flowchart

- 서버 프로그램의 흐름도는 아래와 같습니다.



서버 데이터 초기화 : 게임에 필요한 데이터(맵, 시간 등)를 초기화한다.

클라이언트 접속대기 : 3명의 클라이언트가 모두 접속할 때 까지 대기한다.

클라이언트 쓰레드 생성 : 클라이언트가 접속될 때마다 각각의 클라이언트를 관리하기 위한 쓰레드를 만든다. 이 후 클라이언트가 보낸 로그인 정보를 수신한다.

충돌체크 쓰레드 생성 : 충돌체크를 담당하는 쓰레드를 생성한다. CollisionObject() 쓰레드 함수를 인자 값으로 넘긴다.

FromClientProtocol 수신 : 클라이언트의 정보를 받아 ServerObjectMgr내의 멤버변수에 저장한다.

충돌처리 : 충돌체크 쓰레드가 쓰레드함수 클라이언트로부터 수신한 데이터들을 반영하여 플레이어와 블록, 플레이어와 아이템, 블록과 총알간의 충돌처리를 진행한다.

데이터 동기화 : 충돌처리 작업으로 인한 결과 값을 공유자원인 ServerObjectMgr내의 멤버변수에 동기화해준다.

FromServerProtocol 전송 : 최종적으로 동기화된 게임 내의 오브젝트들의 정보와 시간을 각각의 클라이언트에 클라이언트 쓰레드를 통하여 보낸다.

RankingProtocol 전송 : 모든 플레이어가 사망 시 랭킹 정보를 보내준다.

b. Low - level 디자인

· 프로토콜 정의

class FromServerProtocol

// GameState->Update() 에서 게임의 전반적인 데이터를 서버에서 클라이언트로 매 프레임 send() 되는 구조체.

클라이언트의 RecvServerInfo() 에서 recv()된다.

// 서버는 클라이언트에 첫 FromServerProtocol을 보내며 게임시작을 알린다.

int GameStateInfo	: 현재 게임의 상황을 알려주는 변수
int BulletCnt	: 화면의 총알 개수
int BlockCnt	: 화면의 블록 개수
int ItemCnt	: 화면의 아이템 개수
float Time	: 흘러간 시간을 저장하는 변수
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 FromClientProtocol

// 각 클라이언트에서 충돌체크 및 동기화를 위하여 사용자 및 총알의 정보 서버로 넘겨 줄 때 사용되는 구조체.

서버의 RecvClientInfo() 에서 recv()된다.

String ID	: 플레이어의 아이디
int BulletCnt	: 총알의 개수
Player PlayerInfo	: 플레이어의 정보
Bullet BulletInfo[BulletCnt]	: 총알의 정보

class RankingProtocol

// 게임이 끝난 후 랭킹을 송/수신 할 때 사용되는 구조체

클라이언트에서 이 패킷을 받으면 정보를 출력하며 커넥트를 끊는다.

int RankCnt	: 랭킹에 기록된 랭킹 개수
String Name[RankCnt][3]	: 랭킹을 기록한 아이디 나열
Time time	: 랭킹에 기록된 시간

// 클라이언트와 서버 양쪽에서 사용될 객체 및 기타 클래스 정의한다.

```
class Object
// Player, Bullet, Block, Item 을 자식 클래스로 가지는 부모 클래스
    int Type          : 객체들의 타입
    bool IsDead       : 객체의 존재유무
    Vec2 Pos          : 객체들의 좌표
    Vec2 Size         : 객체의 크기

class Player
    String Name        : 플레이어의 ID
    int Type :: Object : 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
    float x, y : x, y의 좌표

class Time
    int minute,second : 랭킹에 사용될 시간
    float m_second   : 랭킹에 사용될 시간
```

· 클라이언트

class WaitScene : public Scene

클라이언트의 정보를 넘기고, 서버가 다른 플레이어의 접속을 비롯한 초기 정보를 보내주기 전까지 대기하는 클래스.

virtual void Initialize();

connect() 호출하는 함수. connect()를 호출하기 전 플레이어의 ID를 입력 받는다. 그후 서버에 접속하여 대기상태에 들어감을 알릴 수 있다.

virtual void Update();

다른 플레이어가 접속을 할 때까지 대기하고 있다. 게임을 시작할 정보를 받기위해(FromServerProtocol) recv()를 호출하고 있다. recv() 가 되면 모든 플레이어가 접속했다는 뜻이므로 씬을 넘겨준다.

class GameScene : public Scene

실제 게임이 진행되는 씬의 정보를 가지고 있는 클래스. 전반적인 게임 데이터의 송수신 이 본 클래스에서 이루어진다.

void InitStartInfo();

WaitScene에서 받았던 정보로 씬을 동기화 해준다.

GameScene클래스의 멤버함수인 Initailize()에서 호출된다.

void RecvServerInfo();

서버에서 넘어오는 패킷 구조체(FromServerProtocol)를 recv() 하는 함수다. GameScene->Update()에서 호출된다.

void SendClientInfo();

클라이언트에서 서버로 FromClientProtocol 구조체를 send() 하는 함수다.

GameScene->Update()에서 호출된다. 플레이어 본인의 HP가 0이 되면 '관전' 상태로 넘긴다. 플레이어 모두가 HP가 0이 되었나를 판단하여 GameOverScene 으로 넘긴다.

class GameOverScene : public Scene

게임이 끝나면 나오는 씬이다.

void RankingProcessing();

RankingProtocol을 recv() 하여 랭킹 데이터를 업데이트 해주고 통신을 종료한다.

· 서버

SOCKET ObjectSend - TCP

// FromServerProtocol 패킷을 보내기위한 소켓.

SOCKET ObjectRecv - TCP

// FromClientProtocol 패킷을 받기위한 소켓.

class ServerObjectMgr

// 서버에서 처리해야할 오브젝트를 관리하는 클래스, 공유자원으로 사용되는 클래스이다.

list<Player> PlayerList : 플레이어를 관리하는 리스트.

list<Bullet> BulletList : 총알을 관리하는 리스트.

list<Block> BlockList : 블럭을 관리하는 리스트.

list<Item> ItemList : 아이템을 관리하는 리스트.

Thread ClientDataProcessing()

// 게임 중의 데이터 프로토콜을 송/수신해주는 스레드 함수

공유자원인 ServerObjectMgr에 있는 데이터를 FromServerProtocol 에 실어보내준다. 또한 FromClientProtocol을 받아 공유자원인 ServerObjectMgr에 넣어 갱신해준다.

Thread CollisionObject()

// 충돌체크는 매순간 해야 하므로 ServerObjectMgr 의 값을 이용하여 매순간 충돌체크를 해준다. 충돌이 되었을 시 ServerObjectMgr 의 플레이어, 총알, 블럭, 아이템에 정보를 갱신시켜준다. 갱신시켜주며 플레이어의 체력도 0이 되었나 살펴 게임을 끝내는 정보도 판단해준다.

Thread 동기화 및 공유자원에 대한 이슈

// 공유자원인 Object(Player,Bullet,Block,Item)을 각 플레이어간 데이터 불일치를 막기 위하여 순서대로 자원에 접근해야 한다.

이벤트를 사용한다. 각 스레드는 순서를 구분하기 위해 자동 리셋 이벤트를 사용한다.

void GameOver()

// 게임이 끝났을 때 불리는 함수. RankingProtocol을 send() 해준다.

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
프레임워크 제작 메인	서버 제작 제작	클라이언트 제작 메인
클라이언트 제작 보조	서버 송수신 구현 제작	프레임워크 제작 보조
클라이언트 송수신 구현 보조	Thread 및 임계영역 설계	프로토콜 제작
충돌체크 구현	충돌체크 동기화	클라이언트 송수신 구현 메인

c. 개발일정

· 11월

	윤정훈		김군환		박지희	
	개발내용	비고	개발내용	비고	개발내용	비고
11일(토)	기획서 수정		기획서 수정		기획서 수정	
12일(일)	기획서 수정		기획서 수정		기획서 수정	
13일(월)	기획서 수정		기획서 수정		기획서 수정	
14일(화)	프레임워크 제작 OpenGL 프레임워크		시험 준비		시험 준비	
15일(수)	프레임워크 제작 OpenGL 프레임워크		기획서 수정		기획서 수정	
16일(목)	프레임워크 제작 OpenGL 프레임워크		FromServerProtocol 제작		프로토콜 제작 Object 프로토콜 제작	
17일(금)	프레임워크 제작 OpenGL 프레임워크		FromClientProtocol, RankingProtocol 제작		Object 프로토콜에 맞춰 Scene 설계	
18일(토)	Scene 클래스 제작		서버 초기화 함수 구현. (bind(), connect())		Scene 클래스 제작	
19일(일)	구현 내용 점검 및 미구현 내용 구현					
20일(월)	Object 클래스 제작		SceneObjectMgr 클래스 구현		WaitScene.Initialize() 구현 - connect() 및 초기화	
21일(화)	Object 클래스 제작		SceneObjectMgr 클래스 구현		WaitScene.update() 제작 - recv() 및 초기화	
22일(수)	Object 클래스 제작		서버,클라이언트 송/수신 테스트		서버,클라이언트 송/수신 테스트	
23일(목)	Object 클래스 제작 각 오브젝트들 업데이트 및 초기화		이벤트 및 쓰레드 설계		GameScene.Initializ e() 제작 - InitStartInfo() 제작	
24일(금)	SceneMgr 프레임워크 제작		이벤트 및 쓰레드 설계		GameScene.Update() 제작 - RecvServerInfo();	
25일(토)	SceneMgr 프레임워크 제작		이벤트 및 쓰레드 설계		GameScene.Update() 제작 - RecvServerInfo();	
26일(일)	구현 내용 점검 및 미구현 내용 구현					
27일(월)	GameScene.Update()		송/수신 하는 Thread		SceneMgr	

	제작 - void SendClientInfo();		ClientDataProcessing 함수 구현		프레임워크 실행 테스트.	
28일(화)	GameScene.Update() 제작 - void SendClientInfo();		송/수신 하는 Thread ClientDataProcessing 함수 구현		GameOverScene 제작 (렌더링)	
29일(수)	GameScene.Update() 내의 send()와 recv() 설계		송/수신 하는 Thread ClientDataProcessing 함수 구현		GameOverScene의 RankingProcessing() 제작(send())	
30일(목)	GameScene.Update() 내의 send()와 recv() 설계		이벤트 구현		GameOverScene의 RankingProcessing() 제작(send())	

· 12월

	윤정훈		김균환		박지희	
	개발내용	비고	개발내용	비고	개발내용	비고
1일(금)	GameScene.Update() 내의 send()와 recv() 마무리 및 테스트		이벤트 구현		게임 실행 테스트	
2일(토)	서버,클라이언트 동기화 테스트		클라이언트 쓰레드 동기화 테스트		서버,클라이언트 동기화 테스트	
3일(일)	동기화 테스트 마무리 및 구현 내용 점검 및 미구현 내용 구현					
4일(월)	충돌체크 함수 구현		충돌체크 함수 Thread 구현		충돌체크 함수 구현	
5일(화)	충돌체크 함수 구현		충돌체크 함수 Thread 구현		충돌체크 함수 구현	
6일(수)	충돌체크 동기화 테스트		충돌체크 동기화 테스트		충돌체크 동기화 테스트	
7일(목)	충돌체크 동기화 테스트		충돌체크 동기화 테스트		충돌체크 동기화 테스트	
8일(금)	충돌체크 동기화 테스트		충돌체크 동기화 테스트		충돌체크 동기화 테스트	
9일(토)	동기화 테스트 마무리 및 구현 내용 점검 및 테스트					
10일(일)	동기화 테스트 마무리 및 구현 내용 점검 및 테스트					
11일(월)	추가 요구사항 구현		추가 요구사항 구현		추가 요구사항 구현	
12일(화)	추가 요구사항 구현 및 테스트		추가 요구사항 구현 및 테스트		추가 요구사항 구현 및 테스트	
13일(수)	추가구현 내용 점검 및 테스트					
14일(목)						