레이디버그
추진 계획서
<2018년도 네트워크게임프로그래밍 텀프로젝트>

팀원: 차지환[2016184041] 정민지[2016184034] 고송지[2016180001]

학과: 게임공학과, 엔터테인먼트 컴퓨팅 학과

과목: 네트워크게임프로그래밍

목차

4. High-level 디자인

A. *서버 클라이언트 관계도*

B. *서버 설명*

C. *클라이언트 설명*

5. Low-level 디자인

A. *서버*

B. *클라이언트*

C. *프로토콜*

D. *오류 처리*

E. *동기화*

6. 팀원 별 역할 분담

A. *전체 차트*

B. *개발 순서*

7. 개발 인정

1. 개요

A. 애플리케이션 소개

i. 게임에 대해 간략한 소개를 한다.

ii. 플레이 방법과 조작키를 설명한다.

B. 개발 환경

i. 사용하는 플랫폼과 SDK, API 를 소개한다.

ii. 개발 목표와 사용 프로토콜을 정의한다.

C. High-level 디자인

i. 클라이언트와 서버의 큰 윤곽을 나타낸다.

ii.. 서버와 클라이언트 사이의 관계도를 통해 각각의 역할을 명확히 표현한다.

D. Low-level 디자인

i. 서버 구현을 상세하게 구성, 설명한다.

ii. 클라이언트의 서버통신 원리와 과정을 상세하게 구성, 설명한다.

iii. 애플리케이션 내의 오류 처리와 동기화에 관해 정리한다.

E. 팀원 별 역할 분담

i. 각 팀원의 역할을 기입한다.

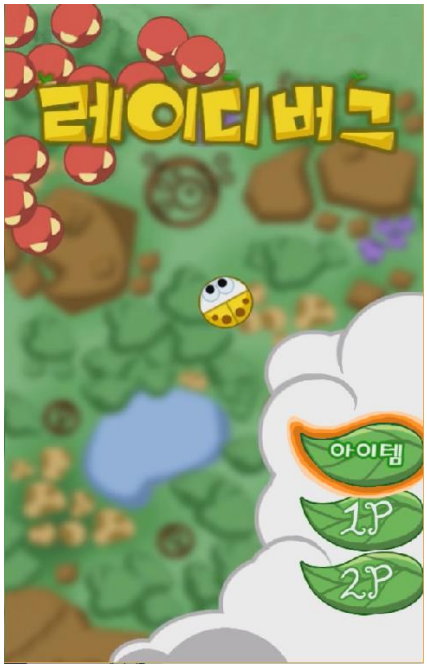
ii. PM 을 선정하고, 팀원의 역할에 따른 개발 순서를 표현한다.

F. 개발 일정

i. 전체 개발 일정을 기입하고, 완료 상태를 점검한다.

2. 애플리케이션 소개

A. 게임 소개



기존에 있던 스마트폰 게임 레이디버그는 싱글 게임으로 스마트폰을 기울여 레이디버그를 조작하는 게임이다. 아이템은 총 10개가 있으며 플레이 타임 동안 아이템을 통해 버그를 피하고 오래 버티는 것이 목적이다.

B. 아이템 소개

1. 꽃잎 폭탄

아이템을 먹은 그 자리에서 꽃잎이 뱅글뱅글 돌며 적을 차단한다.

2. 콩벌레

벽에 최대 5번까지 튕기며 화면 내 적들을 처리한다.

3. 코스모스

코스모스 여러 개가 일렬로 위로 올라가며 화면을 깨끗하게 청소해준다. 하지만 코스모스가 올라오는데 시간이 필요하므로 레이디버그가 위험하다면 아이템을 먹은 후 아래로 빠지는 요령이 필요하다.

4. 분신

레이디버그의 분신이 나타나 위로 올라가며 적들이 레이디버그 대신 이 분신이 있는 쪽을 쫓아간다. 하지만 이미 화면에 나온 적들이 아닌 새로 나오는 적들에게 적용되기 때문에 조심해야 한다.

5. 블라인드

일정시간 동안 검정 블록이 내려왔다 올라가며 플레이어의 시야를 가리는 방해 아이템

C. 플레이 방법 및 조작 설명

1. lobby

서버의 ip주소를 입력하여 로비에 접속한다. 모두 접속이 되었을 시 게임이 시작된다.

2. play

게임이 시작되면 두 플레이어는 방향키 (←→↑↓) 를 이용해 이동하며 서로 협력한다. 아이템을 얻어 오래 살아 점수를 획득하면 되는 게임이다.

3. game over

게임이 종료되면 다시 로비로 돌아간다.

3. 개발환경

A. 플랫폼

PC, 아케이드

B. SDK, API

마이크로소프트 비주얼 스튜디오 2017 사용.

윈도우 API

C. 개발 목표

i. 최대 2 명의 플레이어가 서버에 접속하여 게임 준비 버튼을 누르고,

호스트가 게임을 시작하도록 구현한다.

ii. 플레이어는 랜덤으로 출몰하는 아이템들을 사용하여 최대한

긴 시간 동안 버그와의 충돌을 피해 살아 남는다.

iii. 두 명의 플레이어가 전부 사망할 시, 메인 메뉴로 돌아간다.

D. 사용 프로토콜

서버와 통신하는 데이터가 여러 종류가 있기 때문에 신뢰성 확보가 필요

하고, 때에 따라서 전체 또는 일부 클라이언트에게 데이터를 송신해야

하기 때문에 멀티 쓰레드 환경이 적합하다 판단.

TCP/IP 프로토콜을 사용한다.

서버와 클라이언트는 초당 30 번(2 프레임에 한번씩[서버 기준])

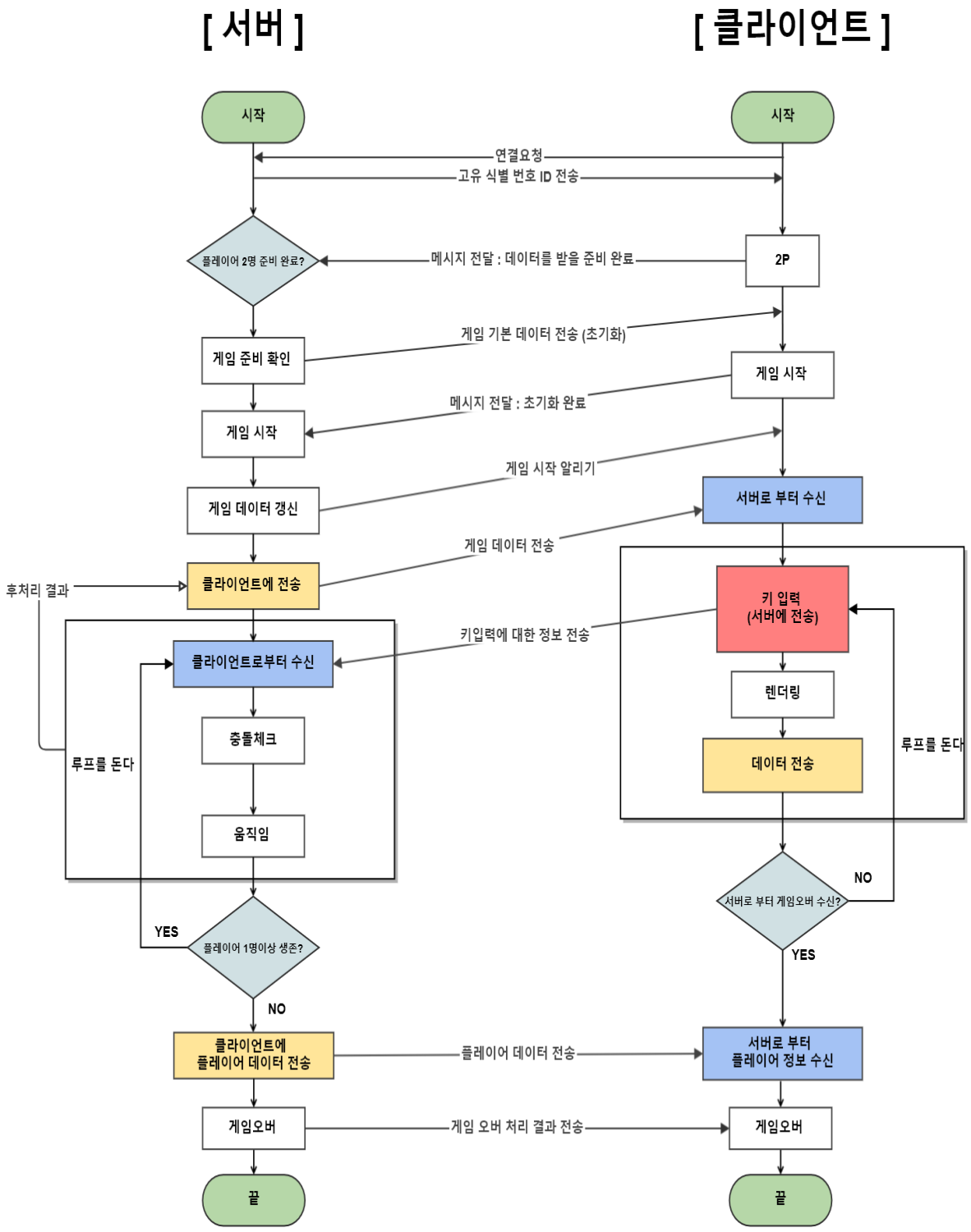
클라이언트의 모든 정보를 주고 받는다.

새로운 오브젝트 생성은 별도로 서버가 관리 하고

모든 클라이언트에 전송한다.

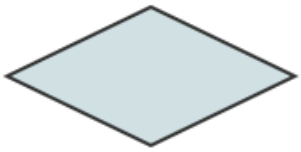
4. High-Level

#서버와 클라이언트 관계도



#플로우 차트 설명

시작과 끝을 알림



조건을 검사

데이터를 전송

데이터를 수신

루프 박스

#서버 플로우 차트

1. 게임을 실행시키고 게임을 시작 하기 전 클라이언트의 연결 요청을 기다린다.

2. 연결이 완료되면 클라이언트에 클라이언트 식별을 위한 고유 ID를 전송한다.

3. 플레이어 2명의 준비가 완료 되었는지 확인한다.

4. 플레이어 2명 모두 준비가 확인이 되면 클라이언트에 게임의 기본 데이터를 전송(초기화) 한다.

5. 클라이언트에서 초기화 완료 메시지가 들어오면 게임을 시작한다.

6. 게임 데이터를 갱신하면서 클라이언트에 게임 시작을 알린다.

7. 처음에는 초기화된 정보를 클라이언트에 전송을 하고, 키 입력에 따라서 클라이언트가 전송한 정보를 바탕으로 루프를 돌기 시작한다.

8. 키 입력 정보를 받아 충돌체크와 움직임을 검사한 후 후처리 결과를 다시 클라이언트에 전송한다.

9. 위 작업을 플레이어의 생존에 따라(플레이어 1명 이상 생존) 루프를 돌 것인지 검사한다.

10. 플레이어가 1명 이상 생존 하지 않았을 경우 플레이어의 죽음 데이터 정보를 클라이언트에 전송한다.

11. 게임오버 처리를 위하여 게임오버 데이터를 클라이언트에 전송하며 게임을 끝낸다.

12. 게임을 마치면 다시 타이틀로 돌아간다.

#클라이언트 플로우 차트

1. 게임을 시작하기 전 서버에 연결 요청을 시도한다.

2. 클라이언트가 서버로부터 식별 ID를 받은 후, 플레이어 2명이 게임을 시작하기 위해서 데이터를 받을 준비가 되었다는 메시지를 서버에 보낸다.

3. 클라이언트가 서버로부터 게임 기본 데이터를 받아 초기화 한다.

4. 데이터 초기화 작업이 완료되면 서버에 초기화 완료 메시지를 보낸다.

5. 서버에서 게임시작을 알려주면 그 때부터 기본 데이터를 토대로 게임을 시작한다.

6. 키가 눌려지면 클라이언트에서 키 입력에 대한 정보를 서버에 보낸다.

7. 서버가 검사를 마치고 후처리 결과를 보내면 클라이언트가 받아 렌더링(Rendering) 작업을 한다.

8. 클라이언트가 서버로부터 게임오버 데이터를 전송 받기 전이라면 위 작업을 계속해서 진행한다(loop).

9. 서버로부터 게임오버 데이터를 받았다면 플레이어 2명의 죽음을 확인한다.

10. 서버가 게임오버 처리를 한 후 그 결과 데이터를 클라이언트가 받으면 게임을 끝낸다.

11. 게임을 마치면 다시 타이틀로 돌아간다.

5. Low Level

#서버

PLAYER		
타입	변수	설명
float	x	오브젝트의 x좌표
float	y	오브젝트의 y좌표
int	count	몇번째 오브젝트인가
int	state	0=죽어있는 상태? 1=살아있는 상태
int	collisionWithWho	어떤 오브젝트와 충돌했는가
BUG		
타입	변수	설명
float	x	버그의 x좌표
float	y	버그의 y좌표
float	x_move	버그가 이동할 x거리
float	y_move	버그가 이동할 y거리
ITEM		
타입	변수	설명
float	x	버그의 x좌표
float	y	버그의 y좌표
float	x_move	버그가 이동할 x거리
float	y_move	버그가 이동할 y거리
Session		
타입	변수	설명
SOCKET	socket	클라이언트 소켓
SOCKADDR_IN	sock_addr	클라이언트 주소
ID	client_id	클라이언트 식별 ID

CLASS		
NetworkManager : 멤버변수		
타입	변수	설명
SOCKET	m_socket	연결을 받아들이는 소켓
HANDLD	m_event;	동기화를 위한 임계영역
int	m_playerCount	현재 플레이어 수, ID 부여하기 위해 사용하기도함
Player[]	m_players	플레이어들을 저장 하기 위한 객체 배열
Bug[]	m_bugs	버그 정보들을 모아놓은 배열
Item[]	m_items	아이템 정보들을 모아 놓은 배열
std::thread	m_thread	클라이언트 접속 후 할당 되는 쓰레드를 저장한 배열

NetworkManager : 멤버함수			
반환 타입	메소드 명	인자	설명
int	Initialize()	X	소켓과 윈속 라이브러리 초기화 등
int	Finalize()	x	소켓과 윈속 라이브러리 종료처리
bool	AcceptLoop()	X	연결을 받아들이는 루프를 반복함 연결 후 스레드를 할당해줌

전역 함수			
void	WorkerThread( )	Session*	thread를 사용할 함수
void	ProcessPacket( )	char*	처리 함수
int	recvn()	char*, int	recvn 함수

#클라이언트

구조체 설명 MOVE		
타입	변수	설명
float	x	오브젝트의 x좌표
float	y	오브젝트의 y좌표
int	count	몇 번째 오브젝트인가
int	state	0=죽어있는 상태? 1=살아있는 상태
int	collisionWithWho	어떤 오브젝트와 충돌했는가
구조체 설명 BUG		
타입	변수	설명
float	x	버그의 x좌표
float	y	버그의 y좌표
float	x_move	버그가 이동할 x거리
float	y_move	버그가 이동할 y거리

변수 설명		
타입	변수	설명
bool	item_flowerBomb_Flag[10]	아이템을 먹으면 플래그를 1로 올린다
int	item_flowerBomb_Frame[10]	스프라이트 이미지 프레임 카운트
int	item_flowerBomb_count	현재 배열 몇번째까지 아이템이 나왔는지 관리
bool	item_interrupt_Flag[10]	방해 아이템을 먹으면 플래그를 1로 올린다
int	item_interrupt_Frame[10]	스프라이트 이미지 프레임 카운트
int	item_interrupt_count	현재 배열 몇번째까지 아이템이 나왔는지 관리
bool	item_beanWorm_Flag[10]	아이템을 먹으면 플래그를 1로 올린다
int	item_beanWorm_Frame[10]	스프라이트 이미지 프레임 카운트
int	item_beanWorm_count	현재 배열 몇번째까지 아이템이 나왔는지 관리
bool	item_cosmos_Flag[10]	아이템을 먹으면 플래그를 1로 올린다
int	item_cosmos_Frame[10]	스프라이트 이미지 프레임 카운트

int	item_cosmos_count	현재 배열 몇번째까지 아이템이 나왔는지 관리
bool	item_playerCopy_Flag[10]	아이템을 먹으면 플래그를 1 로 올린다
int	item_playerCopy_Frame[10]	스프라이트 이미지 프레임 카운트
int	item_playerCopy_count	현재 배열 몇번째까지 아이템이 나왔는지 관리
int	key	플레이어가 죽지 않는 무적상태로 만들어준다
Move	backGround	게임 배경이 흘러가는 작업을 위한 변수
Move	player	1p 에 해당하는 플레이어
Move	secondPlayer	2p 에 해당하는 플레이어
TCHAR	str[100]	점수 출력을 위한 변수
int	item_Drop_Timer	아이템을 일정 시간마다 생성시켜주기 위해 시간을 체크
Move	item_Drop[20]	생성된 아이템을 저장하는 배열
int	item_count	아이템이 몇개까지 생성되었는지 세는 변수
bool	gameover	값이 1 이면 게임오버인 상태
float	mx	마우스 x 좌표
float	my	마우스 y 좌표
int	score	획득한 점수
bool	pause	일시정지를 체크하는 변수
bool	start	게임시작을 체크하는 변수
int	menu_check	마우스 좌표에 따라 메뉴에서 선택한 메뉴를 담는 변수
int	over_check	게임오버가 됐을 때 게임을 다시 시작한다는 버튼을 눌렀는지 체크
int	itme_menu_check	메뉴에서 아이템 설명 창에 갔을 때 이전이나 다음 버튼을 눌렀는지 체크

#공통함수

함수설명			
리턴값	함수명	파라메터	설명
void	playerCollisionCheck	Move& player	플레이어와 버그가 충돌하는지 체크. 충돌한다면 플레이어 state = 0
void	collisionCheck	Move& itme	아이템과 버그가 충돌하는지 체크. 충돌한다면 버그 state = 0
int	isItemCollisionCheck	Move& item, Move& player	아이템과 플레이어가 충돌했는지 체크. 충돌했다면 return 1, 충돌하지 않았다면 return 0

int	checkMenu	int mx, int my	메인 화면에서 마우스 클릭 좌표에 따른 메뉴 체크
void	ITEM_1_flowerBomb	Move& item_1, int i	꽃 돌아가는 아이템
void	ITEM_2_interrupt	Move& item_2	플레이어들의 시야를 가리는 방해 아이템
void	ITME_6_beanWorm	Move& item_6, int i	콩벌레 튀기는 아이템
void	ITEM_8_cosmos	Move& player, Move& item_8, int i	코스모스 회전 아이템
void	ITEM_10_playerCopy	Move& player, Move& item_10, int i	플레이어 분신 생성 및 돌진 아이템

#프로토콜

Enum

서버 -> 클라이언트 패킷 종류	패킷 이름	패킷 설명
SC_ID_PUT_PACKET	ID 부여 패킷	접속하는 클라이언트한테 ID 부여하는 패킷
SC_GAME_INIT_INFO_PACKET	게임 초기화 패킷	게임 시작하기 전 접속한 유저에게 가지고 있는 게임 정보를 초기화 할 수 있도록 정보(오브젝트 위치랑)가 담긴 패킷
SC_GAME_START_PACKET	게임 시작 패킷	접속 한 유저들한테 게임이 시작했음을 알리는 패킷
SC_MOVE_PACKET	클라이언트의 이동 패킷	클라이언트 위치정보가 담긴 패킷
SC_USE_ITEM_PACKET	클라이언트 아이템 사용 패킷	클라이언트가 아이템과 충돌하여 아이템 사용함을 알리는 패킷
SC_DEAD_PACKET	클라이언트 사망 패킷	클라이언트가 버그와 닿아 사망했음을 알리는 패킷
SC_GAME_END	게임 종료 패킷	모든 유저가 사망하여 게임이 종료 되었음을 알리는 패킷.

클라이언트 -> 서버 패킷 종류	패킷 이름	패킷 설명
CS_READY_PACKET	준비 패킷	접속 후 준비 버튼을 누르면 보내는 패킷.
CS_UNREADY_PACKET	준비 패킷	접속 후 준비를 취소를 서버에게 알리는 패킷.
CS_INIT_COMPLETE_PACKET	게임 준비완료 패킷	게임 초기화 패킷을 받아 준비를 초기화를 끝낸 뒤 보내는 패킷.
CS_MOVE_PACKET	이동 패킷	클라이언트에서 이동 이벤트 마다 이동할 좌표의 패킷을 보냄
CS_GAME_DISCONNECT	게임 종료	클라이언트가 종료 버튼을 눌러 게임을 종료함.

Struct

Packet

Sever to Client

CLASS		
NetworkManager : 멤버변수		
타입	변수	설명
SOCKET	m_socket	연결을 받아들이는 소켓
HANDLD[]	m_event;	동기화를 위한 이벤트 객체
int	m_playerCount	현재 플레이어 수, ID 부여하기 위해 사용하기도함
Player[]	m_players	플레이어들을 저장 하기 위한 객체 배열
Bug[]	m_bugs	버그 정보들을 모아놓은 배열
Item[]	m_items	아이템 정보들을 모아 놓은 배열
std::thread	m_thread	클라이언트 접속 후 할당 되는 쓰레드를 저장한 배열

NetworkManager : 멤버함수			
반환 타입	메소드 명	인자	설명
int	Initialize()	X	소켓과 윈속 라이브러리 초기화 등
int	Finalize()	x	소켓과 윈속 라이브러리 종료처리
bool	AcceptLoop()	X	연결을 받아들이는 루프를 반복함 연결 후 스레드를 할당해줌

전역 함수			
void	WorkerThread( )	Session*	thread를 사용할 함수
void	ProcessPacket( )	char*	처리 함수
int	recvn()	char*, int	recvn 함수

Client to Server

패킷 이름	길이	헤더	추가 내용		
준비 패킷	2	1	2		
	5	CS_READY_PACKET	0XFFFF		
준비 취소 패킷	2	1	2		
	5	CS_UNREADY_PACKET	0XFF00		
게임 준비완료 패킷	2	1	2		
	5	CS_INIT_COMPLETE_PACKET	0x00FF		
이동 패킷	2	1	1	2	2
	9	CS_MOVE_PACKET	ID	X	Y
게임 종료	2	1	2		
	5	CS_GAME_DISCONNECT	0x0000		

5. 팀원 별 역할분담

- 각 팀원의 역할을 표로 제시

- 세 사람의 협업을 위주로 각자 역할을 분담 하였음.

- 팀원 모두 네트워크 기능을 구현하도록 하였음.

- 모든 작업 내용은 Git으로 공유할 예정.

차지환	정민지	고송지
서버 클래스 설계, 서버 스레드 설계	게임 클라이언트 구조 개선	서버 스레드 설계, 서버 클래스 설계
패킷 설계	클라이언트 구조체 설계	서버 연산 설계
동기화 설계	추가기능 및 UI 제작	PM

6. 개발일정

2018 년도 네트워크게임프로그래밍 Term-Project
상세일정 계획표

1. 상세 계획표

월	주차	일	요일	작업자	업무내용	작업확인 (O, △, X)
11 월	1 주차	1	목	차지환		
				정민지		
				고송지		
		2	금	차지환	통신 프로토콜 제작	
				정민지	클라이언트 변수, 함수 수정	
				고송지	서버 Player 클래스 제작	
		3	토	차지환	통신 프로토콜 제작	
				정민지	클라이언트 변수, 함수 수정	
				고송지	서버 Bug 클래스 제작	

		4	일	차지환	통신 프로토콜 제작	
				정민지	클라이언트 변수, 함수 수정	
				고송지	서버 Item 클래스 제작	
		5	월	차지환	서버 gameSceneManager 클래스 제작	
				정민지	대기방 UI 제작	
				고송지	송신, 수신을 담당할 코드 작성	
		6	화	차지환	서버 gameSceneManager 클래스 제작	
				정민지	대기방(Lobby) 설계	
				고송지	송신, 수신을 담당할 코드 작성	
		7	수	차지환	일정회의 및 주간 개발 점검	
				정민지		
				고송지		
	2 주차	8	목	차지환		
				정민지		
				고송지		
		9	금	차지환	충돌함수(PlayerCollisionCheck) 구현	
				정민지	대기방(Lobby) 구현	
				고송지	아이템함수(item_1~item_5) 구현	
		10	토	차지환	충돌함수(BugCollisionCheck) 구현	
				정민지	대기방(Lobby) 구현	
				고송지	아이템함수(item_6~item_10) 구현	
		11	일	차지환	충돌함수(ItemCollisionCheck) 구현	
				정민지	클라이언트 Scene 변경 연동(Lobby->gameScene)	
				고송지	통신 테스트	
		12	월	차지환	서버 멀티스레드 설계	
				정민지		
				고송지		
		13	화	차지환	서버 멀티스레드 구현	
				정민지		
				고송지		

		14	수	차지환	일정회의 및 주간 개발 점검	
				정민지		
				고송지		
	3 주차	15	목	차지환		
				정민지		
				고송지		
		16	금	차지환	서버 멀티스레드 연동	
				정민지		
				고송지		
		17	토	차지환	통신테스트 및 서버 오류사항 수정	
				정민지	통신테스트 및 클라이언트 오류사항 수정	
				고송지	통신테스트 및 서버 오류사항 수정	
		18	일	차지환	접속처리 및 id 부여	
				정민지	클라이언트 초기화 작업 및 초기화 데이터 수신	
				고송지	서버 초기화 작업 및 초기화 데이터 송신	
		19	월	차지환	충돌체크 동기화	
				정민지	클라이언트와 대기방 연동	
				고송지	아이템 동기화	
		20	화	차지환	충돌체크 동기화	
				정민지	클라이언트와 대기방 연동	
				고송지	아이템 동기화	
		21	수	차지환	일정회의 및 주간 개발 점검	
				정민지		
				고송지		
	4 주차	22	목	차지환		
				정민지		
				고송지		
		23	금	차지환	버그 동기화	
				정민지	클라이언트 키입력 패킷 전송 처리	
				고송지	플레이어 동기화	

		24	토	차지환	사망 패킷 연동	
				정민지	클라이언트 플레이어 사망 처리	
				고송지	플레이어 사망 데이터 전송	
		25	일	차지환	서버 게임 스코어 데이터 처리	
				정민지	클라이언트 일시정지 및 게임종료 구현	
				고송지	서버 게임종료 데이터 전송	
		26	월	차지환	1 차 테스트 및 피드백	
				정민지		
				고송지		
		27	화	차지환	테스트 피드백 결과 수정	
				정민지	테스트 피드백 결과 수정	
				고송지	테스트 피드백 결과 수정	
		28	수	차지환	일정회의 및 주간 개발 점검	
				정민지		
				고송지		
	5 주차	29	목	차지환		
				정민지		
				고송지		
		30	금	차지환	예외상황 처리 (플레이어 탈주 등)	
				정민지		
				고송지		
12 월		1	토	차지환	2 차 테스트	
				정민지		
				고송지		
		2	일	차지환	수정된 리소스 서치	
				정민지	배경 사운드 및 이펙트 사운드 서치	
				고송지	배경 사운드 및 이펙트 사운드 서치	
		3	월	차지환	기말고사 준비	
				정민지		
				고송지		
		4	화	차지환	오류 수정 및 리팩터링(refactoring)	

			정민지		
			고송지		
	5	수	차지환	일정회의 및 주간 개발 점검	
			정민지		
			고송지		
	6	목	차지환		
			정민지		
			고송지		
	7	금	차지환	개발 중 추가 문서 종합	
			정민지		
			고송지		
6 주차	8	토	차지환	최종 테스트	
			정민지		
			고송지		
	9	일	차지환	최종 테스트 및 보고서 작성	
			정민지		
			고송지		
	10	월	차지환	최종 검수일	
			정민지		
			고송지		

레이디 버그_업무 계획표xlsx 첨부

- 매주 수요일은 일정회의 및 주간점검의 날

- 매주 목요일은 수업이 가장 많은 날로 휴식