

네트워크 게임 프로그래밍 텀프로젝트 계획서 총책임이 난투

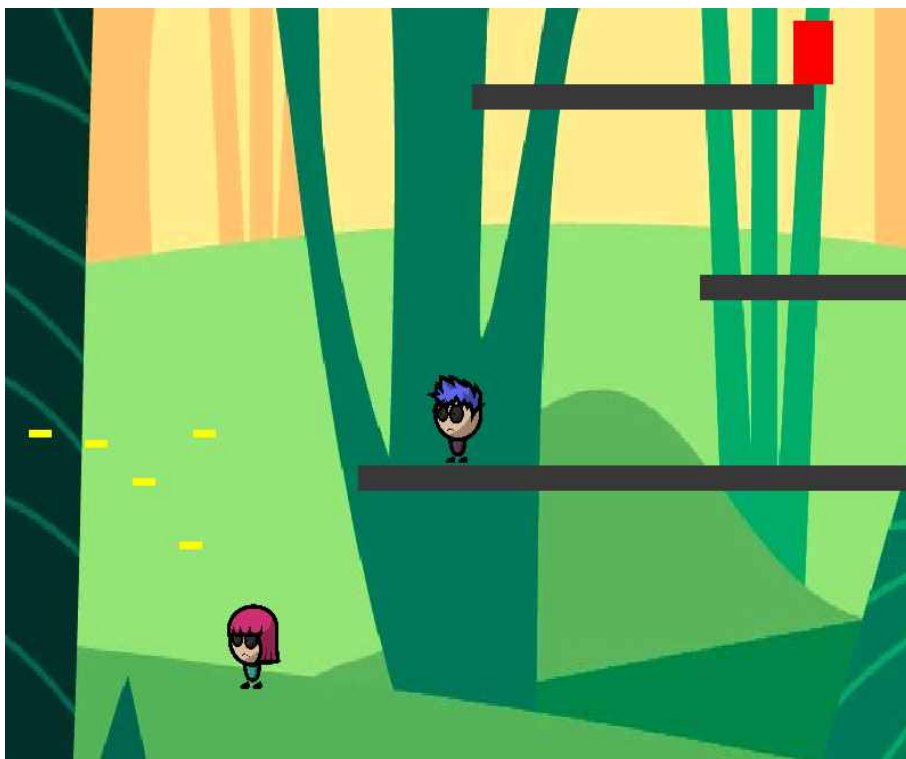
목차

- ▶ 클라이언트 기획
- ▶ 개발 환경
- ▶ High-Level Design
- ▶ Low-Level Design
- ▶ 역할 분담
- ▶ 개발 일정

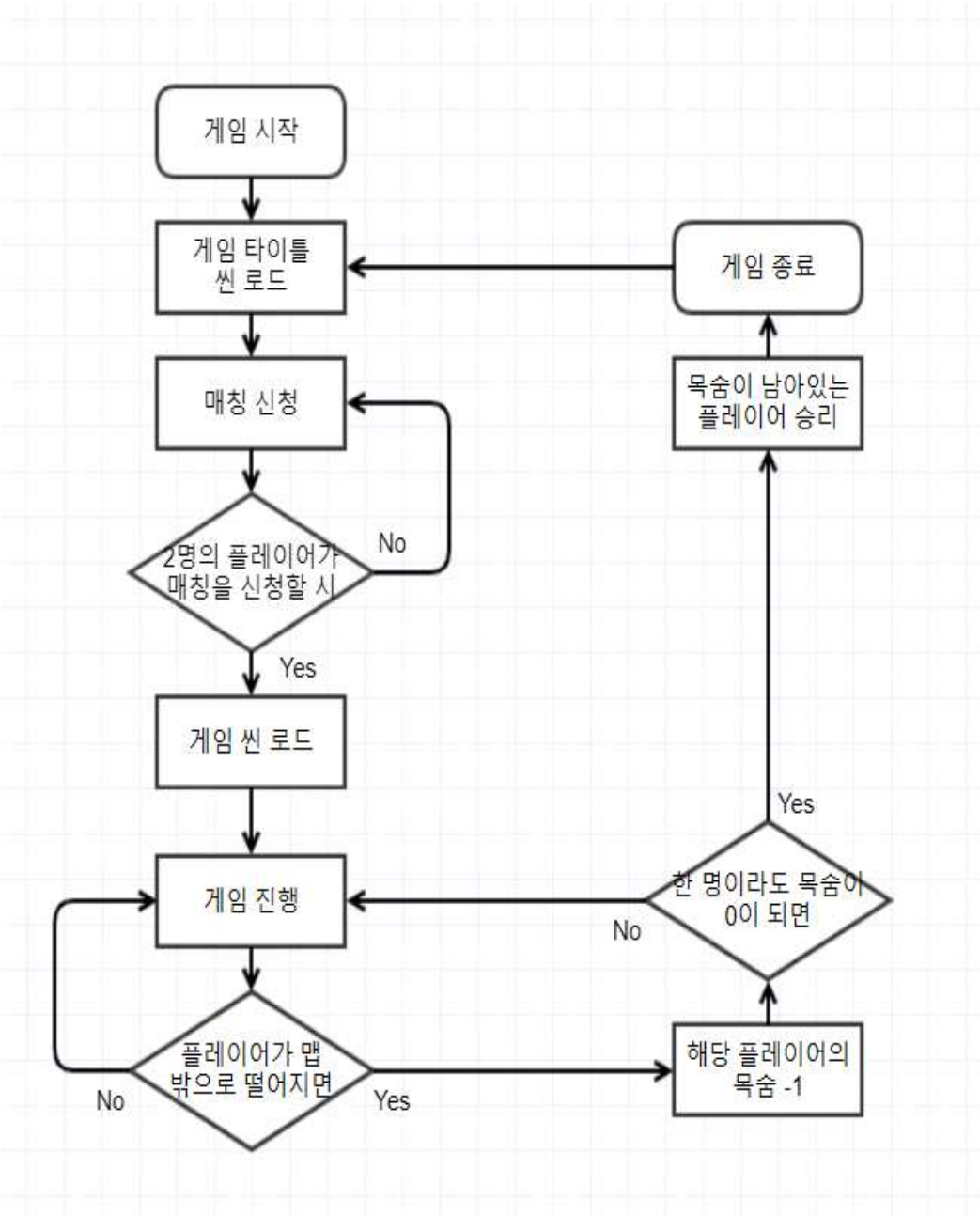
클라이언트 기획

- ▶ 게임 이름: 총잡이 난투(Gun Mayham 모작)
- ▶ 교과목: 네트워크 게임 프로그래밍을 위한 자체제작 게임
- ▶ 게임 장르: 2D Platform Shooting
- ▶ 게임 소개: 적을 총으로 격추시켜 맵밖으로 밀어내서 승리를 쟁취
- ▶ 조작키
 - ▷ 이동: 방향키
 - ▷ 총 발사: A
- ▶ 게임 목표
 - ▷ 모든 플레이어는 중력의 영향을 받아 맵과 충돌 중이 아닌 경우 아래로 떨어짐
 - ▷ 1:1 전투로, 플레이어는 다른 플레이어를 총으로 격추시켜 맵 밖으로 밀어내야 함
 - ▷ 각 플레이어는 3의 체력이 있고, 맵 밖으로 떨어질 경우 체력이 -1
 - ▷ 맵 밖으로 떨어진 후 1초 뒤 공중에서 부활 후 맵 위로 떨어짐
 - ▷ 현재 사용중인 총을 다른 총으로 바꿔주는 아이템이 주기적으로 공중에서 생성
 - ▷ 한 명의 플레이어라도 체력이 0이 되면 해당 게임은 종료

인게임 스크린 샷

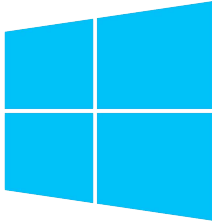


클라이언트 흐름 플로우차트

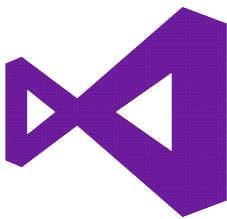


개발 환경

- ▶ 운영체제: Windows



- ▶ 컴파일러: Visual Studio



- ▶ 클라이언트 라이브러리: SFML



- ▶ 통신 프로토콜: TCP/IP



- ▶ 형상관리 프로그램: Git Hub



High-Level Design

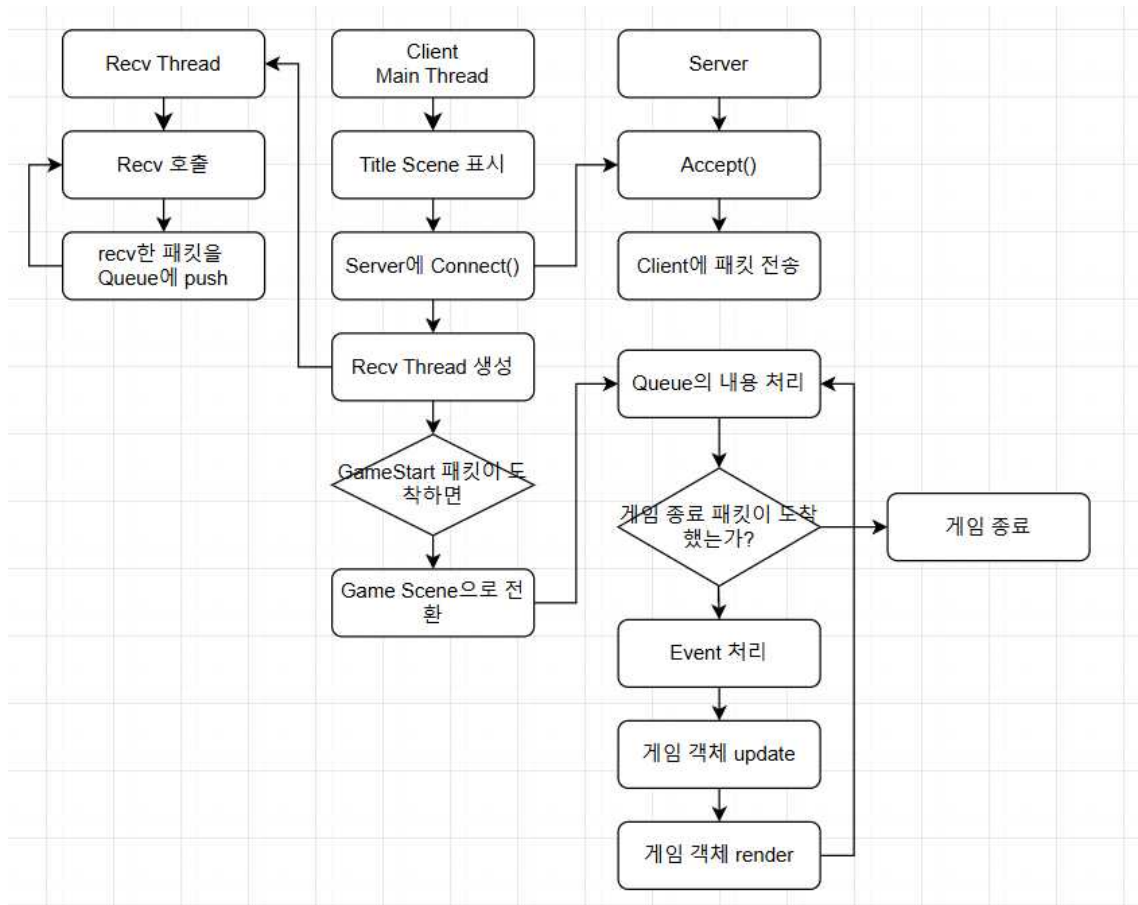
서버 구조

1. 서버 실행
 - ▶ 클라이언트의 연결 대기: accept
2. 매칭
 - ▶ 매칭을 시도하는 플레이어가 2명이면 매칭을 잡고 매칭 완료
3. 충돌 처리
 - ▶ 플레이어 캐릭터와 총알과의 충돌 처리
 - ▶ 플레이어 캐릭터와 아이템과의 충돌 처리
 - ▶ 플레이어 캐릭터와 플랫폼(게임 맵)과의 충돌처리
4. 클라이언트로 패킷 전송
 - ▶ 각 플레이어간 필요한 정보를 클라이언트로 전송

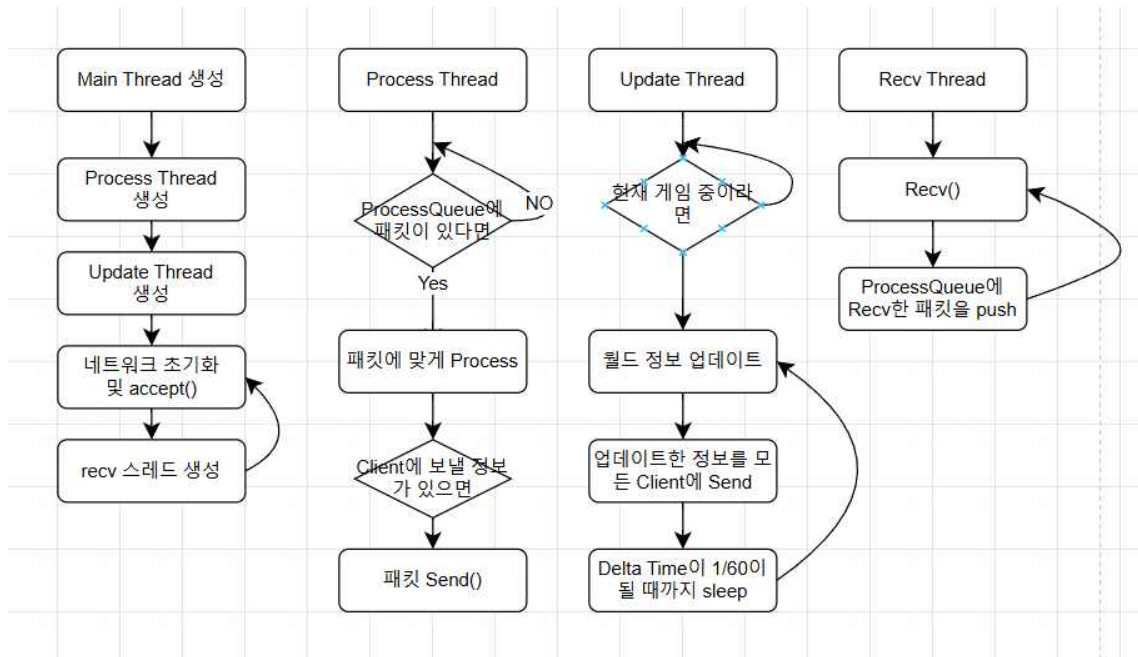
클라이언트 구조

1. 클라이언트 실행
 - ▶ 클라이언트 실행과 동시에 서버에 연결: Connect
 - ▶ 게임 타이틀에서 매칭 신청 버튼을 눌러 매칭 시작
2. 매칭 완료
 - ▶ 매칭이 완료되었다는 알림창을 3초 띄운 후 게임 시작
3. 플레이어 이동
 - ▶ 클라이언트에서 플레이어의 이동을 계산한 후 서버로 전송
4. 서버로 패킷 전송
 - ▶ 각 플레이어간 필요한 정보를 서버로 전송

클라이언트 - 서버 통신 플로우 차트



서버 구조 플로우 차트



Low-Level Design

클라이언트 to 서버

구조체	역할
<pre>struct BASE_PACKET { uint8_t size; uint8_t id; };</pre>	패킷에 공통된 헤더로 패킷이 가질 size와 id를 알려준다
<pre>struct CS_MATCHCATCH_PACKET : public BASE_PACKET { };</pre>	플레이어가 매칭을 잡으려고 한다는 것을 알려주는 패킷
<pre>struct CS_MOVE_PACKET : public BASE_PACKET { uint32_t p_id; float posX, posY; bool dir; bool moving; };</pre>	플레이어 위치와 방향, 움직이는지 아닌지 를 알려주는 패킷 (1/60초 마다 보낼 예정)
<pre>struct CS_FIRE_PACKET : public BASE_PACKET { uint32_t b_id; float posX, posY; bool dir; uint32_t type; std::chrono::milliseconds fire_t; };</pre>	해당 위치에서 총알이 발사되었음을 알리 는 패킷, 발사 위치 및 방향, 총알의 타입 등 정보를 포함 chrono의 시간은 총알의 위치 보간을 위해 사용할 예정 (TM에서)

서버 to 클라이언트

구조체	역할
<pre>struct SC_MATCHMAKING_PACKET : public BASE_PACKET { bool succ; };</pre>	매치 메이킹 성공 실패 여부를 알리는 패킷
<pre>struct SC_MOVE_PACKET : public BASE_PACKET { uint32_t p_id; float posX, posY; };</pre>	플레이어 위치를 서버에서 클라이언트에 알려주는 패킷
<pre>struct SC_PLAYER_DAMAGE_PACKET : public BASE_PACKET { float damage; };</pre>	해당 플레이어가 총알에 맞아 데미지를 받았음을 알리는 패킷
<pre>struct SC_FIRE_PACKET : public BASE_PACKET { uint32_t b_id; float posX, posY; bool dir; uint32_t type; std::chrono::milliseconds fire_t; };</pre>	상대 클라의 총알 발사를 알려주는 패킷
<pre>struct SC_BULLET_REMOVE_PACKET : public BASE_PACKET { uint32_t p_id; uint32_t b_id; };</pre>	해당 총알이 서버에서 처리(소모) 되어 게임 씬에서 사라져야 함을 알리는 패킷
<pre>struct SC_ITEM_CREATE_PACKET : public BASE_PACKET { uint32_t i_id; float posX, posY; };</pre>	해당 위치에 아이템이 생성되었음을 알리는 패킷 (이미지 처리를 위해)
<pre>struct SC_ITEM_REMOVE_PACKET : public BASE_PACKET { uint32_t i_id; };</pre>	해당하는 아이템의 ID의 객체가 소모되어 게임 씬에서 사라져야 함을 알리는 패킷

<pre> struct SC_GUN_UPDATE_PACKET : public BASE_PACKET { uint32_t g_id; }; </pre>	<p>해당 클라이언트의 플레이어의 총기가 변경됨을 알리는 패킷</p>
<pre> struct SC_LIVE_UDPATE_PACKET : public BASE_PACKET { uint32_t player_id; }; </pre>	<p>해당 플레이어가 목숨이 감소했음을 알리는 패킷 (그게 클라이언트 플레이어 ID라면 사망 처리)</p>
<pre> struct SC_GAMEOVER_PACKET : public BASE_PACKET { }; </pre>	<p>게임 종료 조건이 만족되어 게임이 종료되었음을 알리는 패킷 (타이틀 씬으로 복귀)</p>

클래스 예상 구현

클라이언트

```
class NetworkManager {
private:
    Handle socket
    Handle thread
    Queue<Buffer[MAX_SIZE]> process_queue
    int ClientID
public:
    Connect()
    CreateRecvThread()
    NetworkManager()
    PushBuffer( buf )
    ProcessQueue()
    SendPacket( enum id )
    ::WorkerRecv()
    PushBuffer()
    Update()
    Draw()
}
```

```
class SceneManager {
private:
    Vector<Scene>
    Scene currentscene
public:
    ChangeScene()
    Update()
    Draw()
}
```

서버

```
class NetworkManager {
private:
    Handle socket
    Handle thread
    Queue<Buffer[MAX_SIZE]> process_queue
public:
    Init()
    CreateProcessThread()
    CreateUpdateThread()
    CreateRecvThread(HANDLE socket)
    NetworkManager()
    ~NetworkManager()PushBuffer( buf )
    GetQueue()
    SendPacket( enum id )
}

class World {
private:
    Object object
public:
    Update()
    ::WorkerProcess()
    ::WorkerUpdate()
    ::WorkerRecv()
}
```

공용

```
class TimerManager {
private:
    std::chrono::system_clock::time_point oldTime
    std::chrono::milliseconds deltaTime
public:
    void Init()
    void update()
    bool isSyncTime()
    std::chrono::milliseconds epochToMillis()
    std::chrono::milliseconds timeGap(std::chrono::milliseconds bullet_fire_t)
}
```

역할 분담

민경원

- ▶ 클라이언트 네트워크 기능 및 스레드 생성
- ▶ Scene Class 제작 및 객체 관리
- ▶ Server to Client Packet의 처리

김용주

- ▶ 서버 네트워크 기능 및 스레드 생성
- ▶ Client to Server Packet 송신 및 처리
- ▶ Title Scene 생성 및 전환
- ▶ UI 및 Animation 제작

송승호

- ▶ Protocol 생성
- ▶ 서버 내 World 정보 관리
- ▶ Timer Class 제작
- ▶ Server to Client Packet의 송신

개발 일정

10 주차

날짜 팀원	월 (11/4)	화 (11/5)	수 (11/6)	목 (11/7)	금 (11/8)	토 (11/9)	일 (11/10)
민경원	Client Network Mgr 선언 및 정의	C_NM Init()	C_NM Connect ()	C_NM Push Buffer	::Worker Recv()	디버깅, 머지, 오류 수정, 부족한 코드 개선	
김용주	Server Network Manager 선언 및 정의	S_NM init()	S_NM Push Buffer (buf)	S_NM Get Queue()	S_NM Send Packet()		
송승호	Protocol Enum, 상수 정의	Protocol 패킷 선언 및 연동	World 클래스 정의	Object Class 선언 및 정의	Player 객체 구현		

11 주차

날짜 팀원	월 (11/4)	화 (11/5)	수 (11/6)	목 (11/7)	금 (11/8)	토 (11/9)	일 (11/10)
민경원	C_NM Create Recv Thread()	C_NM Process Queue()	SC_matc hmaking 처리	SC_move 처리	Scene 정의 및 선언	디버깅, 머지, 오류 수정, 부족한 코드 개선	
김용주	::Worker Process()	::Worker Update()	::Worker Recv()	Create Thread()	SC_move 보내기		
송승호	Bullet 객체 구현	Item 객체 구현	World Update 구현	World 객체 내 충돌 감지	객체 충돌 처리		

개발 일정

12 주차

날짜 팀원	월 (11/4)	화 (11/5)	수 (11/6)	목 (11/7)	금 (11/8)	토 (11/9)	일 (11/10)
민경원	기존 객체 구조 변경	Scene 에서 객체 관리	Scene update() Draw()	Scene Manager update() Draw()	Scene Manager Change Scene()	디버깅, 머지, 오류 수정, 부족한 코드 개선	
김용주	SC_matc hmaking 보내기	클라이 언트 Send() 구현	CS_MOV E 보내기	Server CS_MOV E 처리	CS_FIRE_ PACKET 보내기		
송승호	Timer Manager 정의 및 선언	TM init()	TM isSyncTi me()	TM epochTo Millis, timeGap	World에 Time 기능 적용		

13 주차

날짜 팀원	월 (11/4)	화 (11/5)	수 (11/6)	목 (11/7)	금 (11/8)	토 (11/9)	일 (11/10)
민경원	SC_FIRE_ 처리	SC_GUN_ UPDATE_ 처리	SC_LIVE_ UPDATE 처리	SC_ITEM _CREATE 처리	SC_ITEM _LOCATI ON 처리	디버깅, 머지, 오류 수정, 부족한 코드 개선	
김용주	Server CS_FIRE_ PACKET 처리	Client UI Class 정의 및 선언	UI Update 및 Draw 구현	플레이어 정보 UI 생성	타이틀 Scene 구현		
송승호	SC_FIRE_ PACKET 전송	SC_GUN_ UPDATE_ PACKET 전송	SC_LIVE_ UDPA TE_ PACKET 전송	SC_ITEM _CREATE 처리	SC_ITEM _LOCATI ON 전송		

개발 일정

14 주차

날짜 팀원	월 (11/4)	화 (11/5)	수 (11/6)	목 (11/7)	금 (11/8)	토 (11/9)	일 (11/10)
민경원	SC_ITEM_REMOVE 처리	SC_BULLET_REMOVE 처리	SC_PLAYER_DAMAGE 처리	SC_GAMEOVER 처리	게임 종료 구현	디버깅, 머지, 오류 수정, 부족한 코드 개선	
김용주	타이틀 Button UI 구현	Scene 전환 적 용	캐릭터 Animation 정의	캐릭터 Animation 구현	게임 종료 시 처리		
송승호	SC_ITEM_REMOVE 전송	SC_BULLET_REMOVE 전송	SC_PLAYER_DAMAGE 전송	SC_GAMEOVER 전송	게임 종료 시 처리		