비전엔진과 프라우드넷을 이용해서 대규모 멀티플레이 구축하기

㈜넷텐션 배현직

초청을 수락해 버렸다!

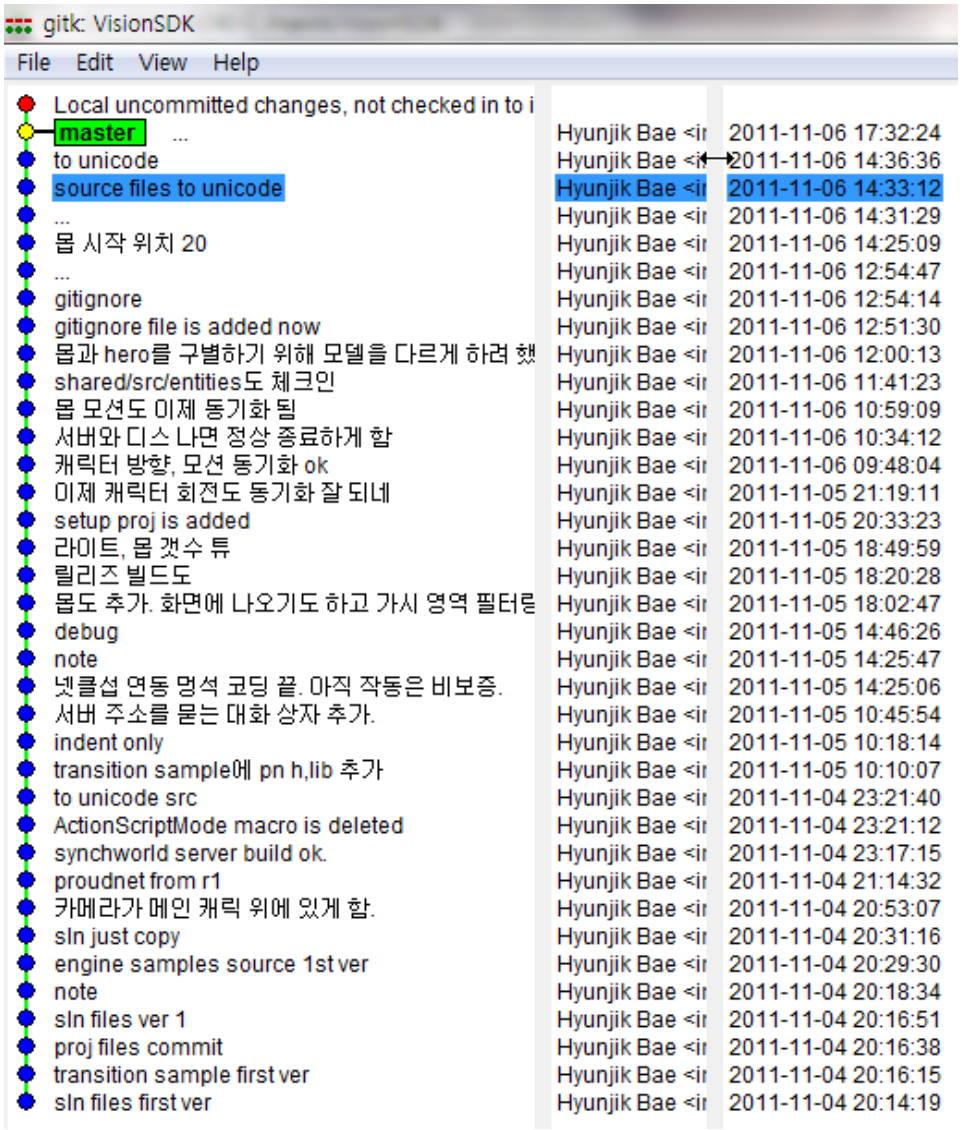
비전 엔진 쓸 줄도 모르는데.....-_-;;;

벼락치기로 만들 수 있는 목표

- 서버에 접속
- 서버에서 내 캐릭터가 돌아다니기
- 다른 캐릭터와 같이 돌아다니기
- 몹들과 같이 돌아다니기
- 여기까지만. 더 욕심 부리지 말자.
  (그럴 시간도 없음)

- 이 모두를 서버 개발자 입장에서 다루어보자

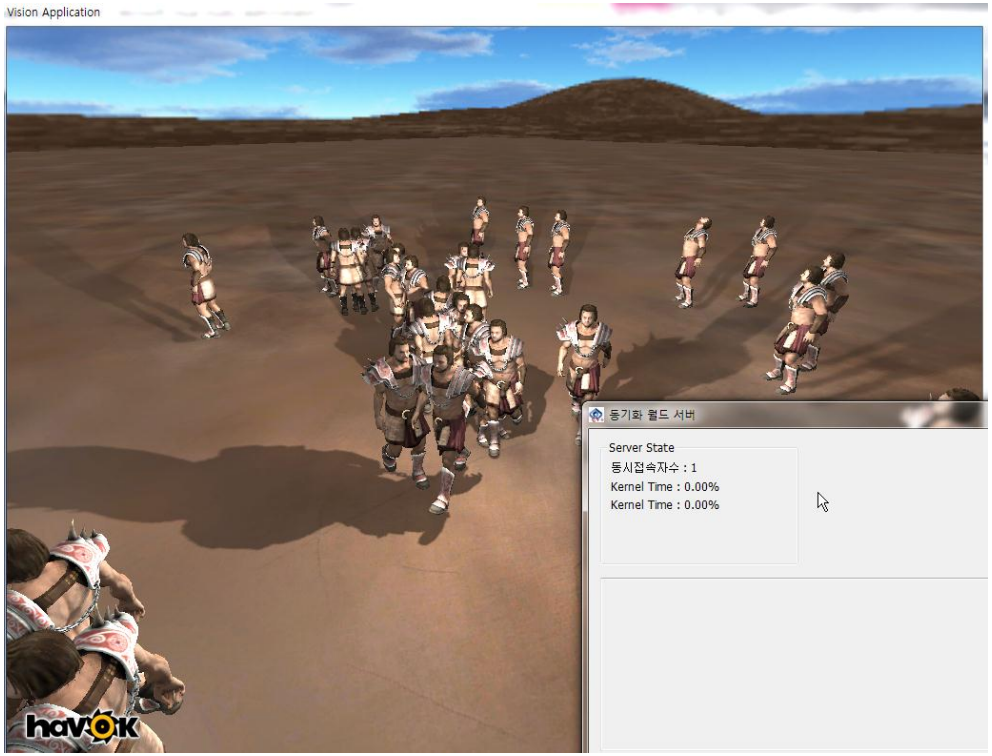
그리고 벼락치기



비전 엔진 공부에 하루
데모 앱 개발에 이틀

많이 부실하지만

- 게다가 몹들이 트위스트 추면서 이동하는 버그
  - 시간 부족으로 해결 못하고 강연 준비 -_-



- 그래도 MMO 스펙 돋음
- 기초적인 부분의 상세적인 구현 예

아아 용두사미 아아

해볼건 다 해봅니다

뭐 어쨌거나 시작해보죠 ㅋㅋㅋ

비전 엔진 초기화

```
VISION_INIT
{
  // setup directories, does nothing on platforms other than iOS,
  // pass true if you want load from the documents directory
  VISION_SET_DIRECTORIES(false);

  // Create an application
  spApp = new VisSampleApp();
  spApp->LoadVisionEnginePlugin();

  // Init the application
  if (!spApp->InitSample("Maps₩₩ViewerMap" /*DataDir*/, "ViewerMap"
/*SampleScene*/ ))
      return false;

  return true;
}
```

로딩 끝난 직후

```
VISION_SAMPLEAPP_AFTER_LOADING
{
  spApp->SetShowFrameRate(true);
  // Create a mouse controlled camera (optionally with gravity)
  VisBaseEntity_cl *pCamera = spApp->EnableMouseCamera();
}
```

메인 루프 & 렌더링

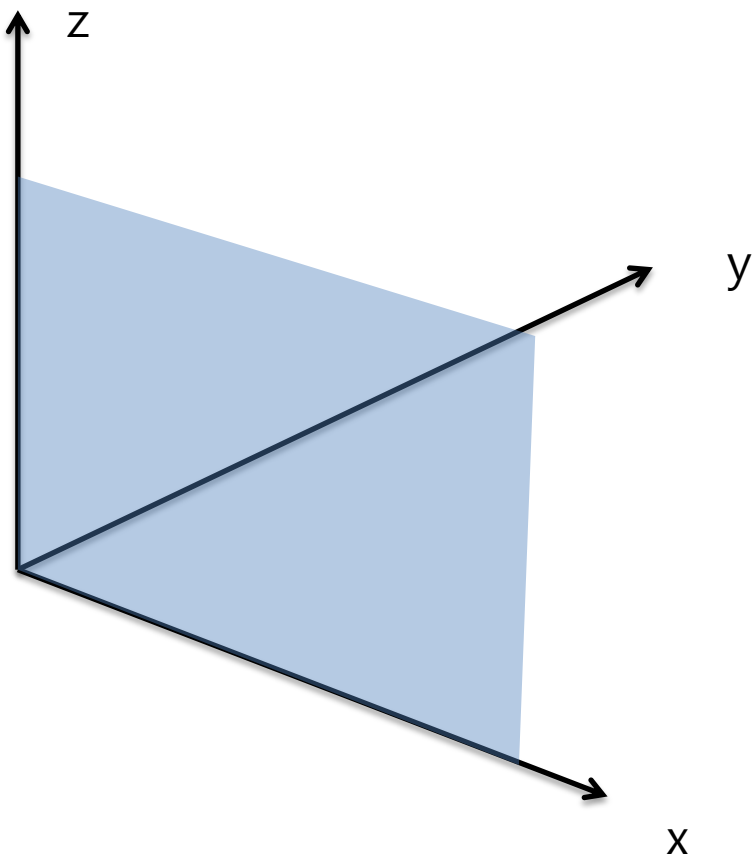
```
VISION_SAMPLEAPP_RUN
{
  return spApp->Run();
}
```

종료

```
VISION_DEINIT
{
  // Deinit the application
  spApp->DeInitSample();
  spApp = NULL;
  return true;
}
```

팁: 이런 매크로를 안 쓰고
그냥 WinMain에서 비전 엔진을 초기화해서 다루는 것도 가능

비전 엔진 좌표계



스크린 안쪽

스크린 바깥쪽

엔티티 로딩

```
VISION_SAMPLEAPP_AFTER_LOADING
{
  // Create a mouse controlled camera
  spApp->EnableMouseCamera();

  // Create an entity of class VisBaseEntity_cl infront of the
camera with a model
  pEntity = Vision::Game.CreateEntity("VisBaseEntity_cl",
VisVector_cl(250,0,-120), "Models₩₩Warrior₩₩Warrior.model");
  pEntity->SetCastShadows(true);
}
```

엔티티 움직이기 &렌더링

```
VISION_SAMPLEAPP_RUN
{
  // Run the main loop of the application until we quit
  if (spApp->Run())
  {
    //Rotate the entity 20 degrees per second using this simulation
frame's time delta
    float fTime = Vision::GetTimer()->GetTimeDifference();
    pEntity->IncOrientation( VisVector_cl(20.f * fTime,0,0) );
    return true;
  }
  return false;
}
```

커스텀 엔티티(모델 데이터와 엔티티별 코드 결합하기)

```
//A custom entity class that rotates the entity
class MyEntity_cl : public VisBaseEntity_cl
{
public:

  MyEntity_cl();
  ~MyEntity_cl();

  VOVERRIDE void InitFunction();
  VOVERRIDE void ThinkFunction();

  V_DECLARE_SERIAL_DLLEXP( MyEntity_cl, DECLSPEC_DLLEXPORT );
  IMPLEMENT_OBJ_CLASS(MyEntity_cl)

  float RotateSpeed;  //rotation speed (degrees per second)
};
```

```
V_IMPLEMENT_SERIAL( MyEntity_cl, VisBaseEntity_cl /*parent class*/ , 0,
&g_MyModule );

// *** The variable table is used by the plugin system of the engine
and editor.
START_VAR_TABLE(MyEntity_cl,VisBaseEntity_cl,"Rotating entity",0,
NULL )
  DEFINE_VAR_FLOAT(MyEntity_cl, RotateSpeed, "Rotation speed (degrees
per second)", "10.0", 0, 0);
END_VAR_TABLE
```

```
//Module to register our classes with
DECLARE_THIS_MODULE( g_MyModule, MAKE_VERSION(MY_VERSION_MAJOR,
MY_VERSION_MINOR), "Engine Sample", "Havok", "Vision Engine Sample",
NULL );
```

```
VISION_INIT
{
  VISION_SET_DIRECTORIES(false);
  // ***************************************************
  // Register the module so the engine knows about the classes
  // ***************************************************
  Vision::RegisterModule(&g_MyModule);

  // Create an application
  spApp = new VisSampleApp();
```

이러한 방식으로 로컬 캐릭터와 리모트 캐릭터를 구현하자

로컬 캐릭터

- 키보드 입력에 따라 이동
- 지형 충돌 반응

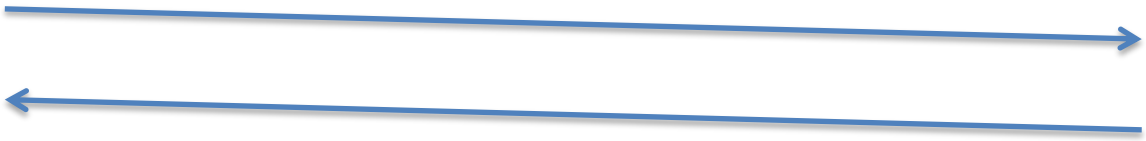
리모트 캐릭터

- 네트워크 메시지 수신에 따라 이동
- 지형 충돌 반응 안함 (샘플이니까)

프라우드넷 클라-서버 연결

1.
static CNetServer.Create
CNetServer.Start

2.
static CNetClient.Create
CNetClient.Connect



클라이언트

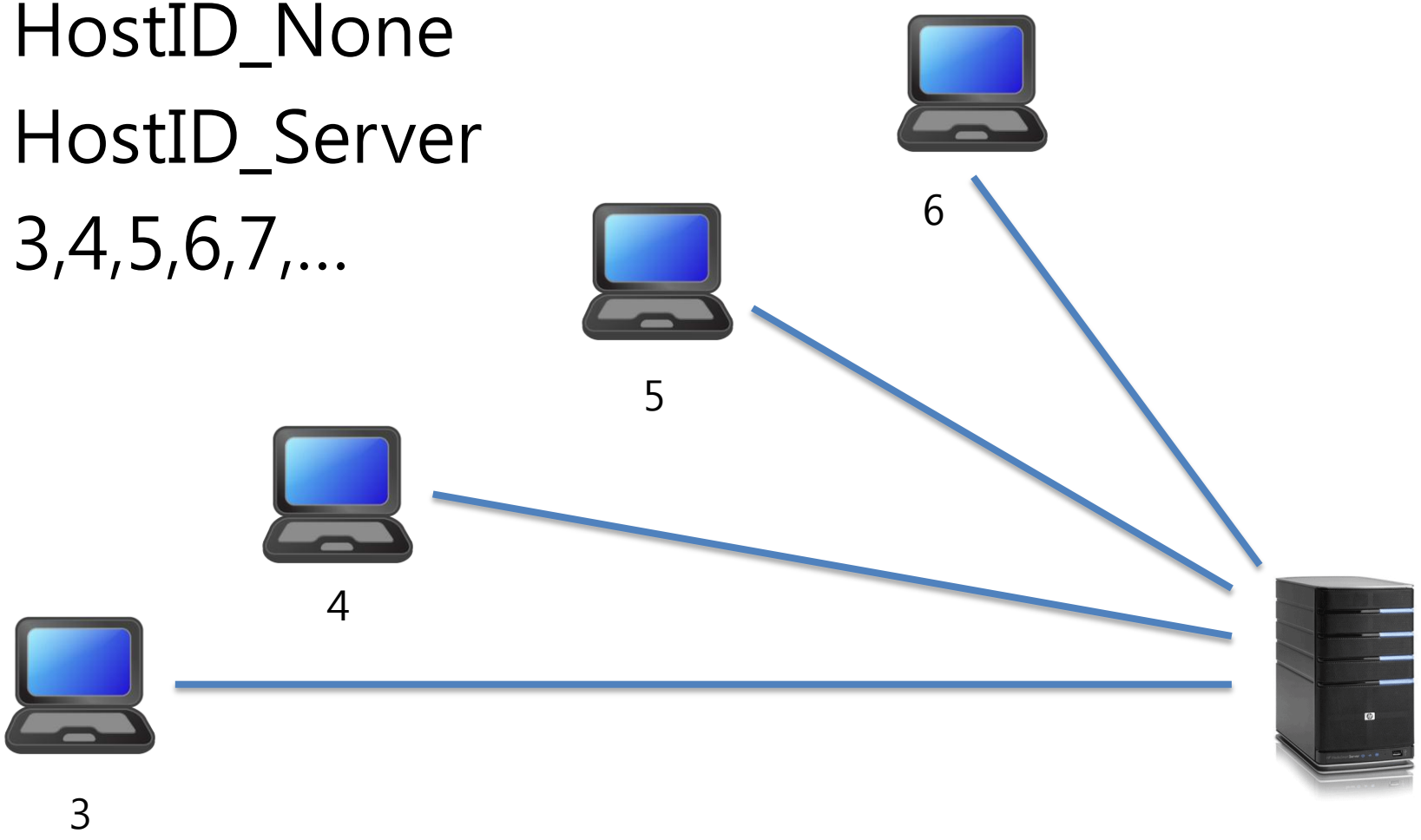
서버

4. OnJoinServerComplete 이벤트
(인자를 통해 성사여부 확인)

3. OnClientJoin 이벤트

HostID

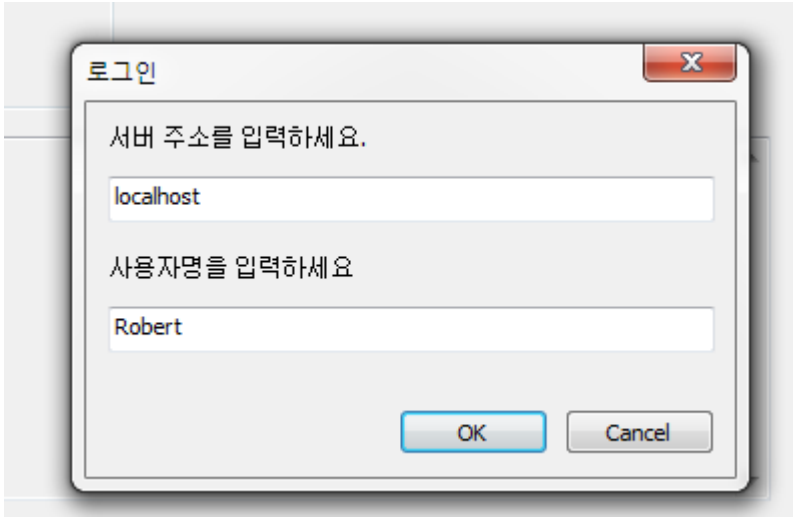
- HostID_None
- HostID_Server
- 3,4,5,6,7,…



클라 시작시 서버 주소 묻기

```
VISION_INIT
{
  <중략>
  // 서버 주소를 묻는 대화 상자
  if(::DialogBox( GetModuleHandle(NULL),
MAKEINTRESOURCE(IDD_LOGIN), NULL, LoginDlgProc) != IDOK)
  {
    <중략>
  }
```

FM대로 게임 개발을 한다면, 이렇게 하지 말고 게임용 GUI를 써야 하겠죠.



서버 시작 코드

```
// 서버를 시작한다.
CStartServerParameter p1;
p1.m_localNicAddr = CA2W(localNicAddr);
p1.m_protocolVersion = g_Version;
p1.m_tcpPort = g_ServerPort;
p1.m_allowServerAsP2PGroupMember = true; // 서버도  P2P GroupMember로 추가를 허락한다.
p1.m_timerCallbackInterval = 10;    // OnTick 기능을 사용하기 위한 함수
srv->Start(p1);
m_server = srv;
```

클라의 서버 연결 코드

```
CNetConnectionParam param;

param.m_protocolVersion = g_Version;
param.m_serverPort = g_ServerPort;
param.m_serverIP = g_World->m_serverAddr;

return m_netClient->Connect(param);
```

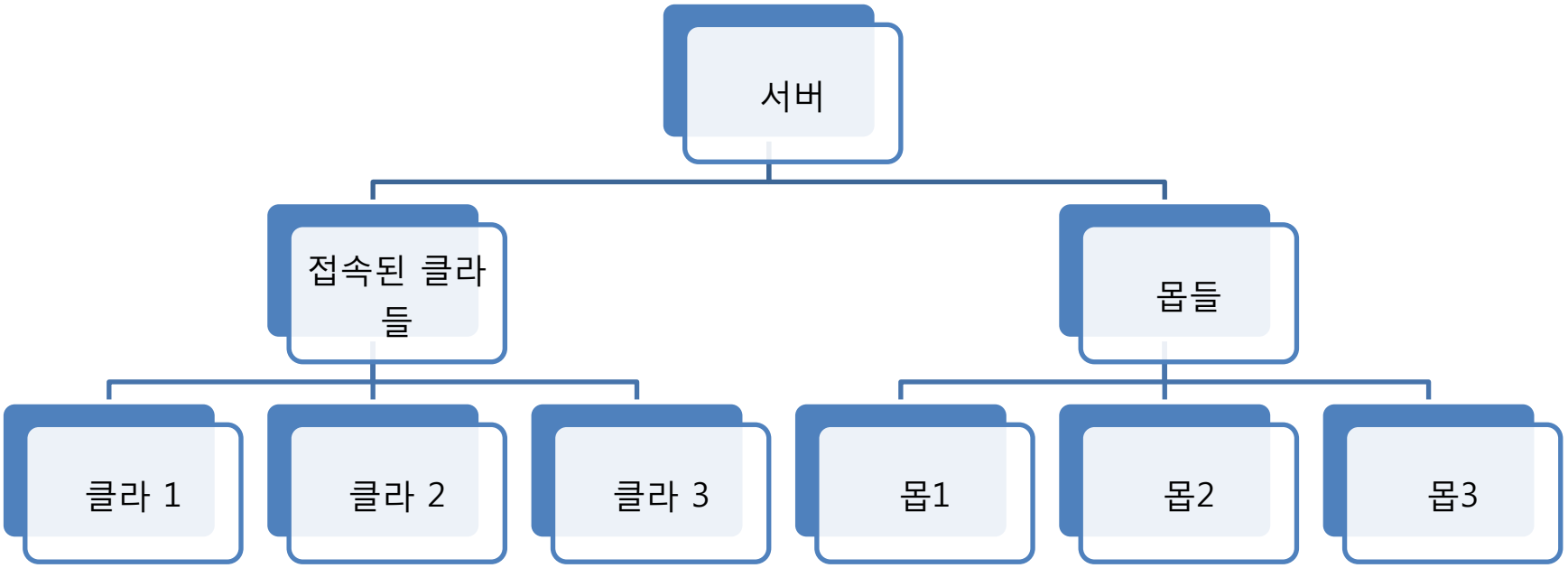
연결 결과 처리 코드(클라)

```
// 서버와의 연결 시도 후 연결이 성공했는지 실패했는지 결과가 나오는 핸들러
void CWorld::OnJoinServerComplete( ErrorInfo *info, const ByteArray
&replyFromServer )
{
  if (info->m_errorType != ErrorType_Ok)
  {
    ::MessageBox(NULL, CW2T(info->ToString()), _T("Error"), MB_ICONHAND|MB_OK);
    g_World->m_keepRunning = false;
  }
  else
  {
    // Setup the character
    VisVector_cl characterOrigin( 200.f, -200.f, 0.f );
    m_localHero = (TransitionCharacter_cl *)
Vision::Game.CreateEntity( "KeyControlledTransitionCharacter_cl",
characterOrigin );
    m_localHero->SetOrientation( 180.f, 0.f, 0.f );
    m_localHero->SetEntityKey("TransitionCharacterEntity");
```

연결 성사 처리 코드 (서버)

```
// 클라이언트가 접속을 들어왔을 때의 핸들러
void CSynchWorldServerDlg::OnClientJoin( CNetClientInfo *clientInfo )
{
  CriticalSectionLock lock(m_cs, true);

  // 클라이언트 객체 생성
  CRemoteClientPtr_S rc = CRemoteClientPtr_S(new
CRemoteClient_S(this));
  rc->m_hostID = clientInfo->m_HostID;

  m_remoteClients[rc->m_hostID] = rc;
}
```

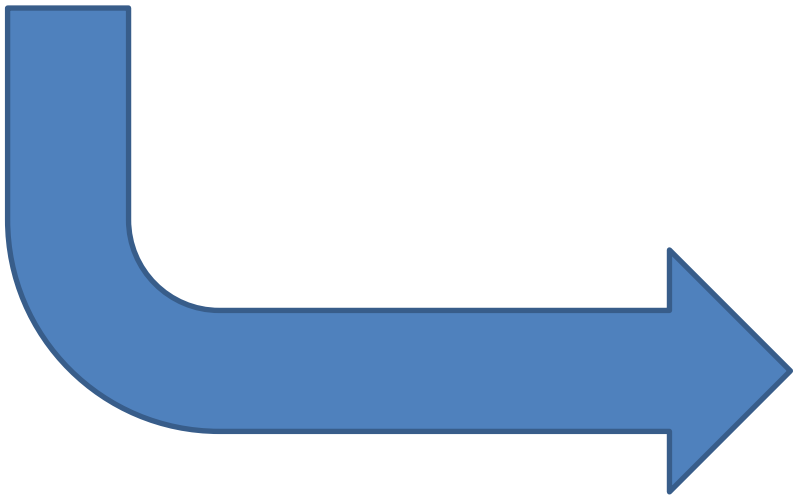
서버 내 데이터 구조



메시지 송수신 vs. RMI

```
Knight_Move([in] int id,[in] float x,[in] float y,[in] float z);
Knight_Attack([in] int id,[in] int target,[in] int damage);
```

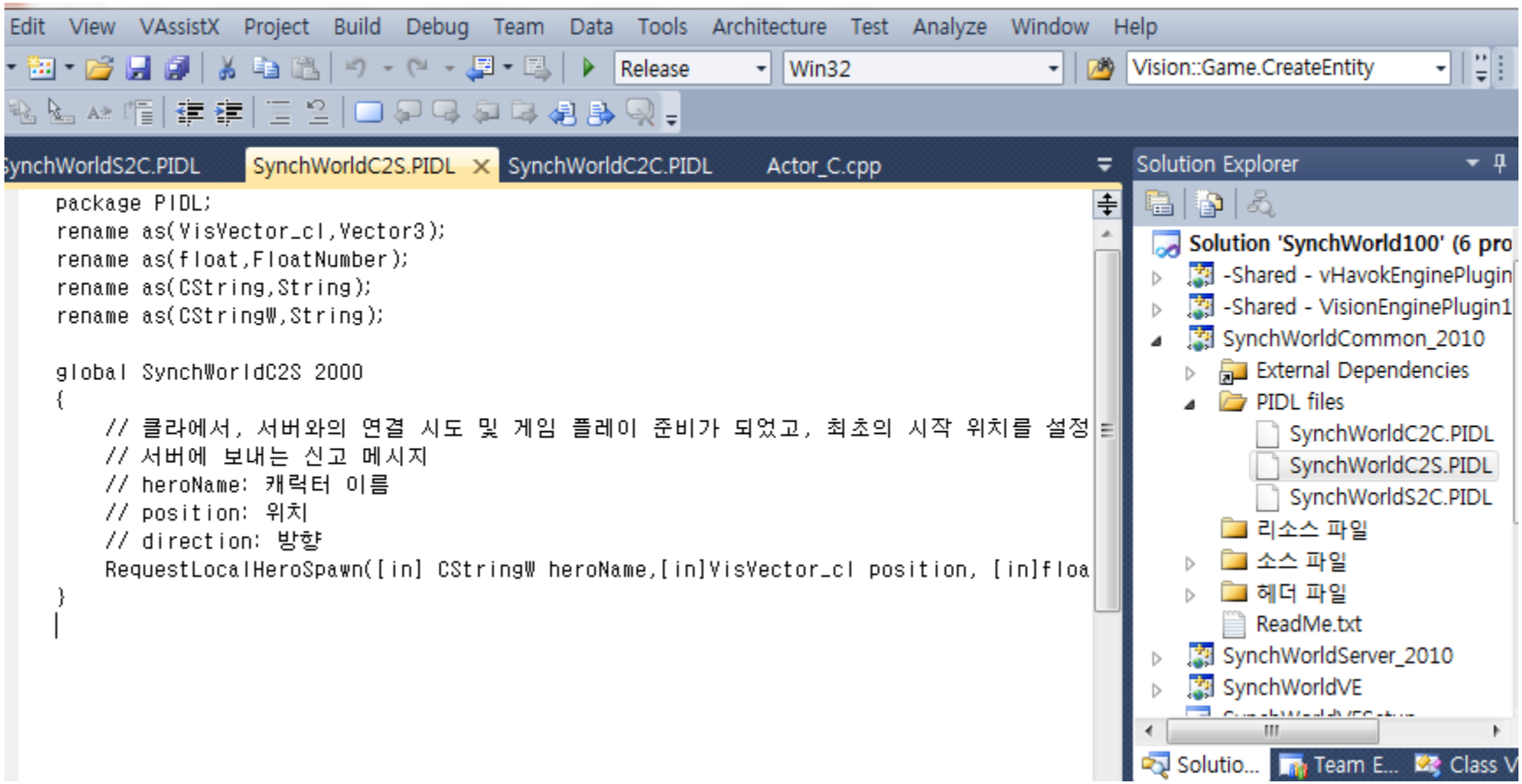
1) 이것을 컴파일하면



2) 이러한 송수신 루틴이 생성됨

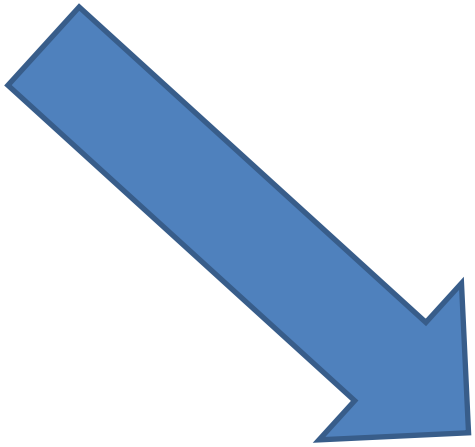
```
// A function which send a formatted message
void Knight_Attack(int id,int target,int damage)
{
    Message_Knight_Attack msg;
    msg.m_msgID=Message_Knight_Attack_ID;
    msg.m_id=id;
    msg.m_target=target;
    msg.m_damage=damage;
    Send(msg);
}

// Identified a received message and call an approp
void DoReceivedMessage(Message* msg)
{
    switch(msg->m_msgID)
    {
    case Message_Knight_Move_ID:
        {
            Message_Knight_Move* msg2=
                (Message_Knight_Move*)msg;

            Do_Knight_Move(
                msg2->m_id,
                msg2->m_x,
                msg2->m_y,
                msg2->m_z);
        }
        break;
        // ... cases for other message types
    case Message_Knight_Attack_ID:
        {
```

RMI 메시지 포맷 정의


Edit View VAssistX Project Build Debug Team Data Tools Architecture Test Analyze Window Help
Release
Win32
Vision::Game.CreateEntity
SynchWorldS2C.PIDL
SynchWorldC2S.PIDL
SynchWorldC2C.PIDL
Actor_C.cpp
package PIDL;
rename as(VisVector_cl,Vector3);
rename as(float,FloatNumber);
rename as(CString,String);
rename as(CStringW,String);
global SynchWorldC2S 2000
{
// 클라에서, 서버와의 연결 시도 및 게임 플레이 준비가 되었고, 최초의 시작 위치를 설정
// 서버에 보내는 신고 메시지
// heroName: 캐릭터 이름
// position: 위치
// direction: 방향
RequestLocalHeroSpawn([in] CStringW heroName,[in]VisVector_cl position, [in]floa
}
Solution Explorer
Solution 'SynchWorld100' (6 pro
-Shared - vHavokEnginePlugin
-Shared - VisionEnginePlugin1
SynchWorldCommon_2010
External Dependencies
PIDL files
SynchWorldC2C.PIDL
SynchWorldC2S.PIDL
SynchWorldS2C.PIDL
리소스 파일
소스 파일
헤더 파일
ReadMe.txt
SynchWorldServer_2010
SynchWorldVE
Solutio...
Team E...
Class V

다른 호스트에 있는 RMI를 호출하기

```
    // setup the camera
    m_pCamera->SetTarget( m_localHero );

    m_C2SProxy.RequestLocalHeroSpawn(HostID_Server, RmiContext::ReliableSend,
        m_userName,
        m_localHero->GetPosition(), m_localHero->GetOrientation().z);
}
```



RMI 호출 받기

```
// RMI 핸들러. 자세한 것은 PIDL 파일을 참고할 것.
DEFRMI_SynchWorldC2S_RequestLocalHeroSpawn(CSynchWorldServerDlg)
{
    CriticalSectionLock lock(m_cs, true);

    CRemoteClientPtr_S newRC = GetRemoteClient(remote);
    // 만약 '그런 클라 없음' 혹은 '이미 캐릭 생성을 노티받았음' 상태이면 무
    if (newRC == NULL || newRC->m_hero != NULL)
    {
        return true;
    }

    // 클라이언트가 모든게 준비가 됐으므로, 상태를 동기화할 캐릭 객체를 생
    CActorPtr_S newHero(new CActor_S(this));
    newHero->m_type = Hero;
    newHero->m_position = position;
    newHero->m_direction = direction;
    newHero->m_name = heroName;
```

서버와의 접속이 성공하면

- 내 캐릭터를 씬에 추가
- 카메라 설정
- 내 캐릭터 등장을 서버에 신고
  - 샘플이다 보니 캐릭 등장 판단을 클라에서 했지만, 실제 개발할 게임에서는 서버에서 판단해야 안전!

내 캐릭터를 씬에 추가

```
// 서버와의 연결 시도 후 연결이 성공했는지 실패했는지 결과가 나오는 핸들러
void CWorld::OnJoinServerComplete( ErrorInfo *info, const ByteArray &replyFromServer )
{
  if (info->m_errorType != ErrorType_Ok)
  { ... }
  else
  {
    // Setup the character
    VisVector_cl characterOrigin( 200.f, -200.f, 0.f );
    m_localHero = (TransitionCharacter_cl *) Vision::Game.CreateEntity( "KeyControlledTransitionCharacter_cl", characterOrigin );
    m_localHero->SetOrientation( 180.f, 0.f, 0.f );
    m_localHero->SetEntityKey("TransitionCharacterEntity");
```

카메라 설정

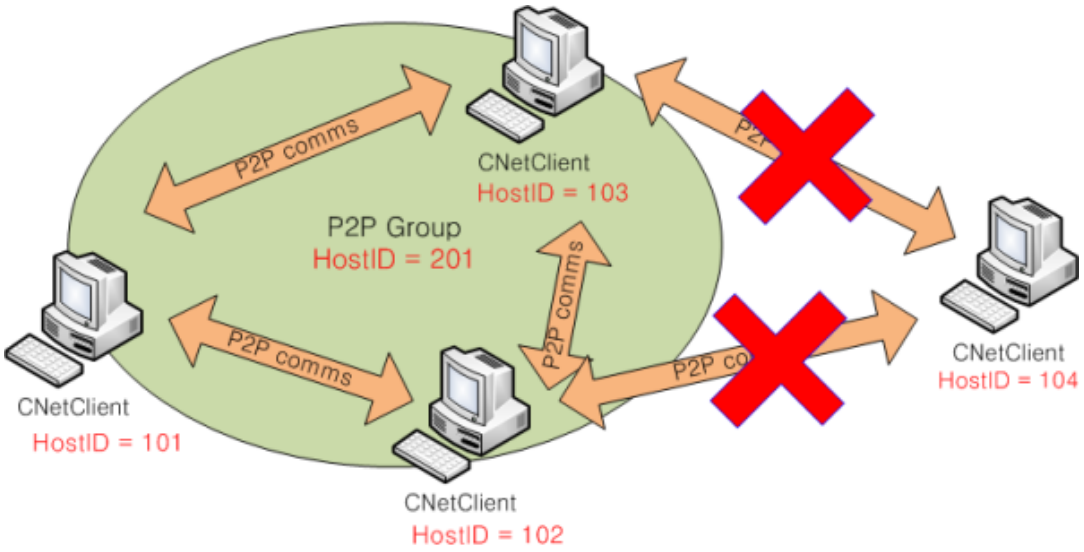
```
    // We want to keep the vertex position in a separate collision
buffer, so we don't need to skin the
    // model again for stencil shadows.
    m_localHero->GetAnimConfig()->SetFlags(m_localHero-
>GetAnimConfig()->GetFlags() | KEEP_EXTRA_COLLISIONBUFFER);

    // setup the camera
    m_pCamera->SetTarget( m_localHero );
```

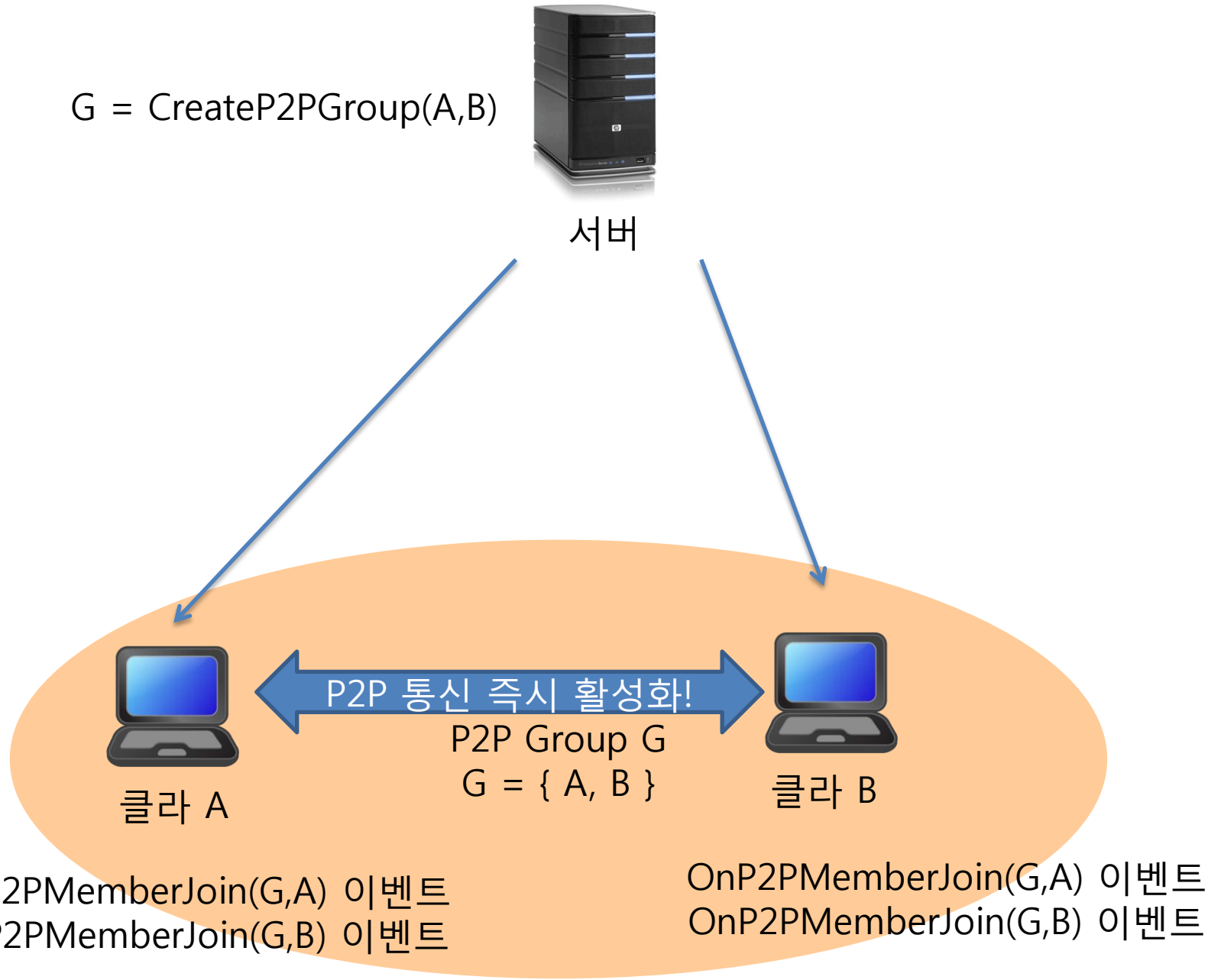
서버에 캐릭 등장을 신고

```
    m_C2SProxy.RequestLocalHeroSpawn(HostID_Server,
RmiContext::ReliableSend,
        m_userName,
        m_localHero->GetPosition(), m_localHero->GetOrientation().z);
```

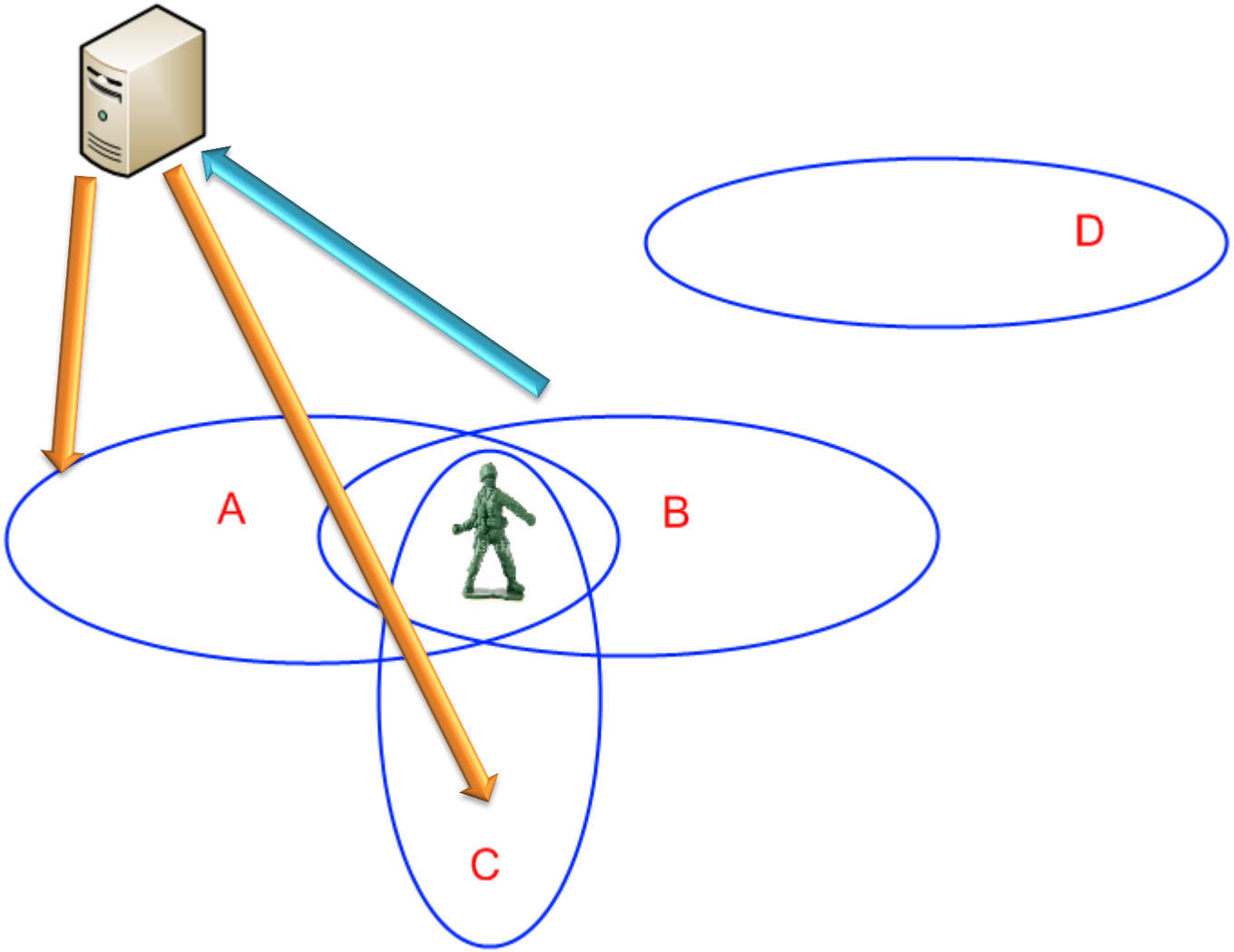
이제 움직이는 캐릭터를 네트웍으로 동기화해보자

P2P 통신과 P2P 그룹

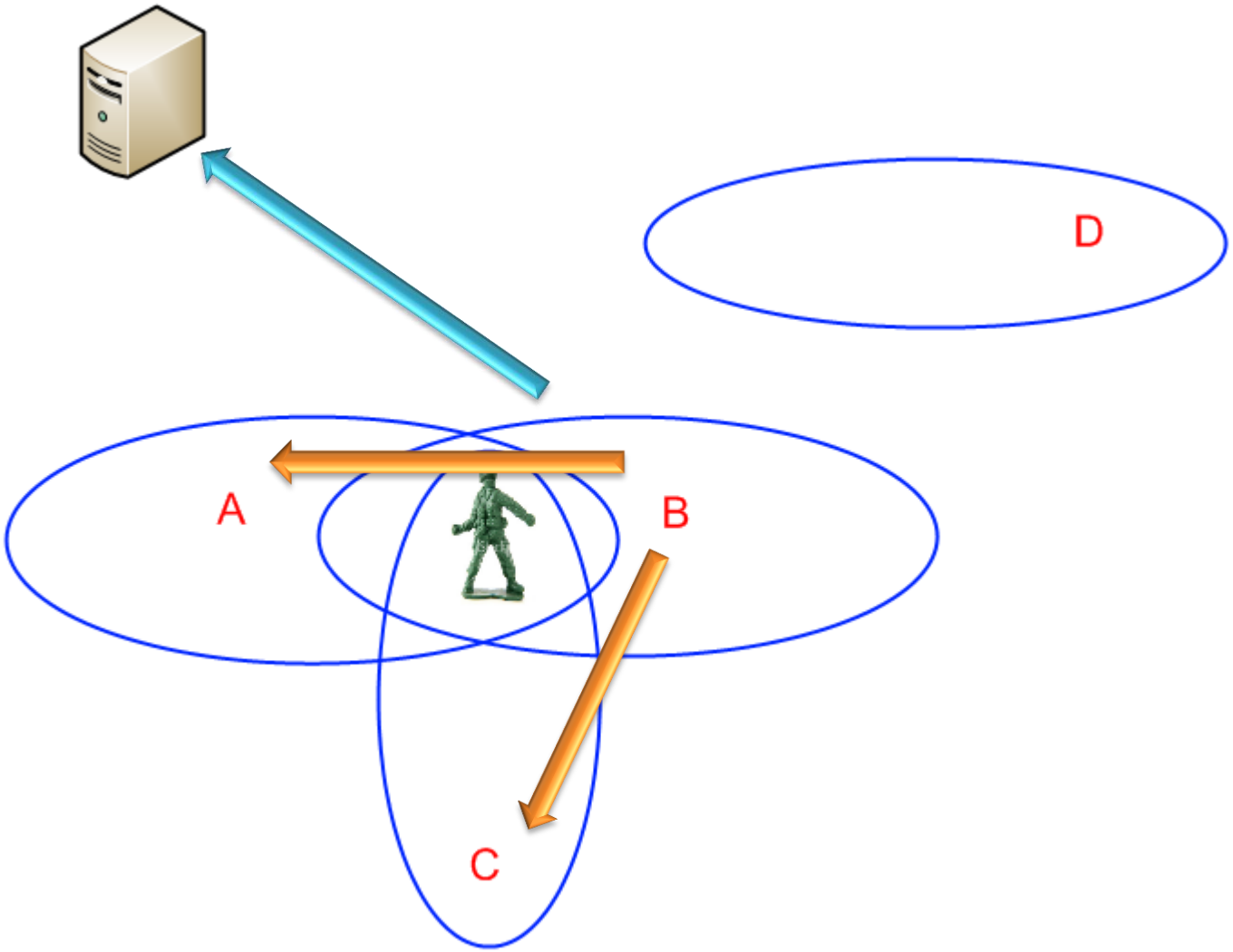



G = CreateP2PGroup(A,B)
서버
P2P 통신 즉시 활성화!
P2P Group G
G = { A, B }
클라 A
클라 B
OnP2PMemberJoin(G,A) 이벤트
OnP2PMemberJoin(G,B) 이벤트
OnP2PMemberJoin(G,A) 이벤트
OnP2PMemberJoin(G,B) 이벤트

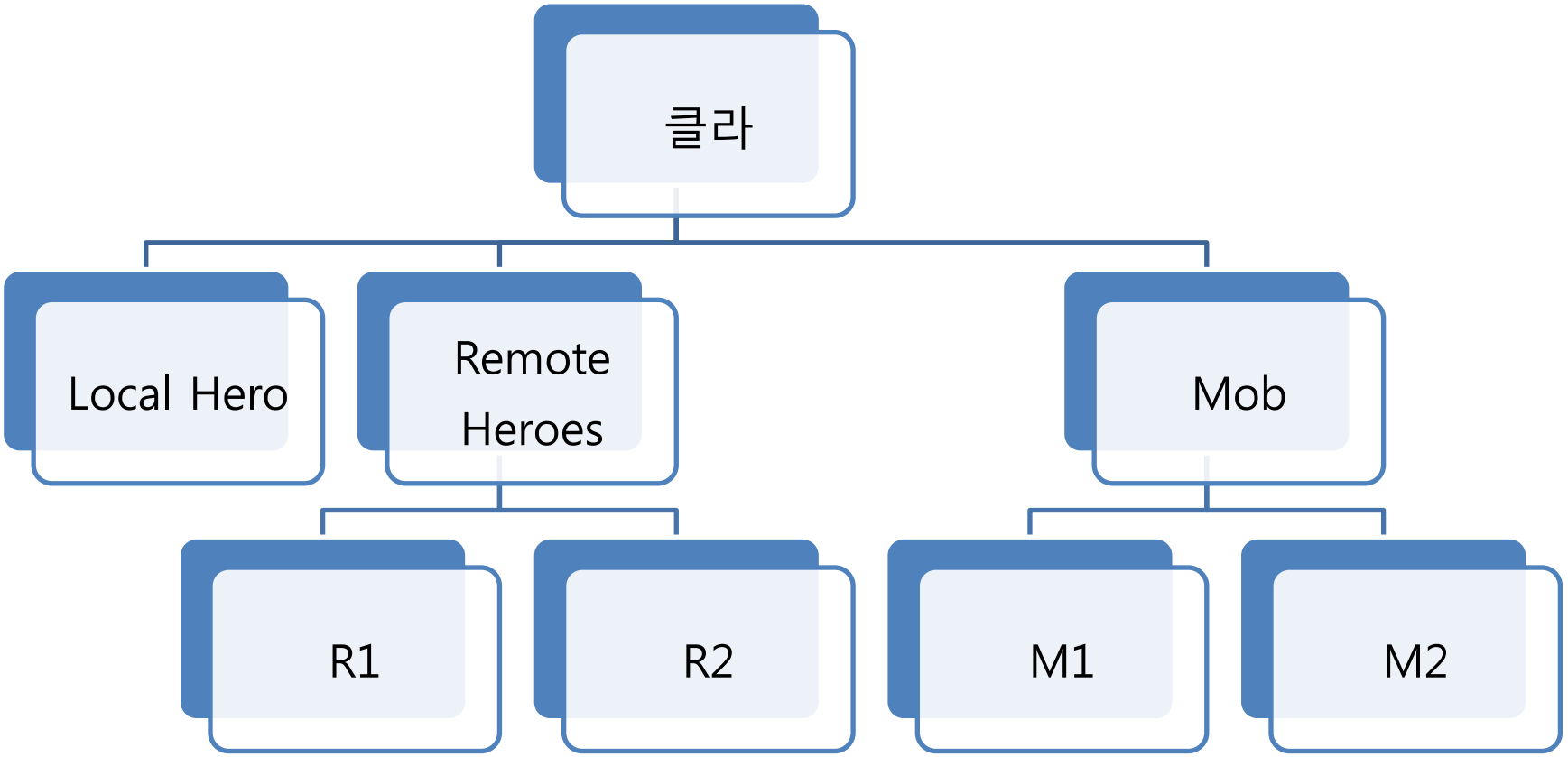
C/S만 하는 가시 영역 동기화



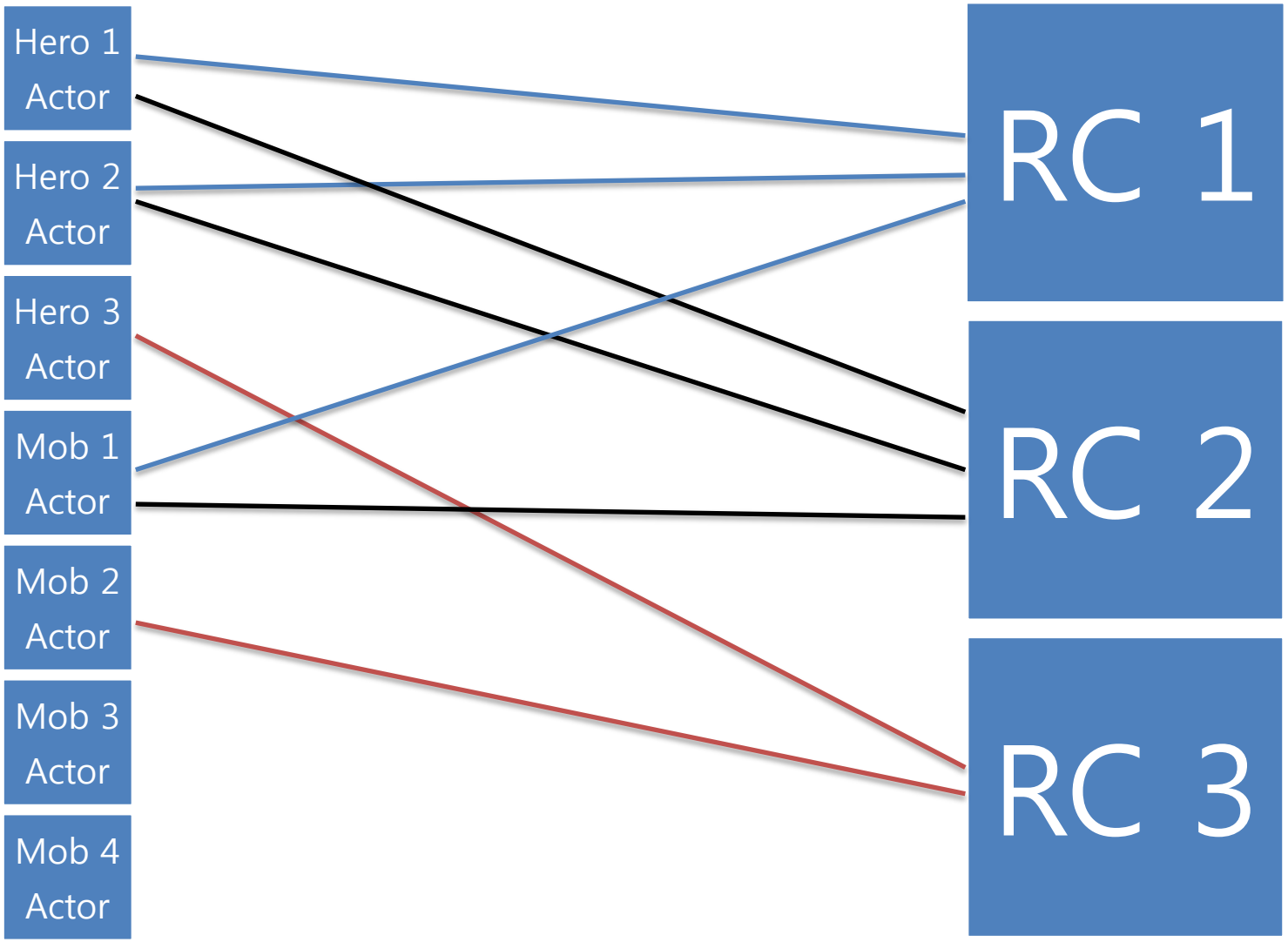
C/S & P2P 혼용 가시 영역 동기화



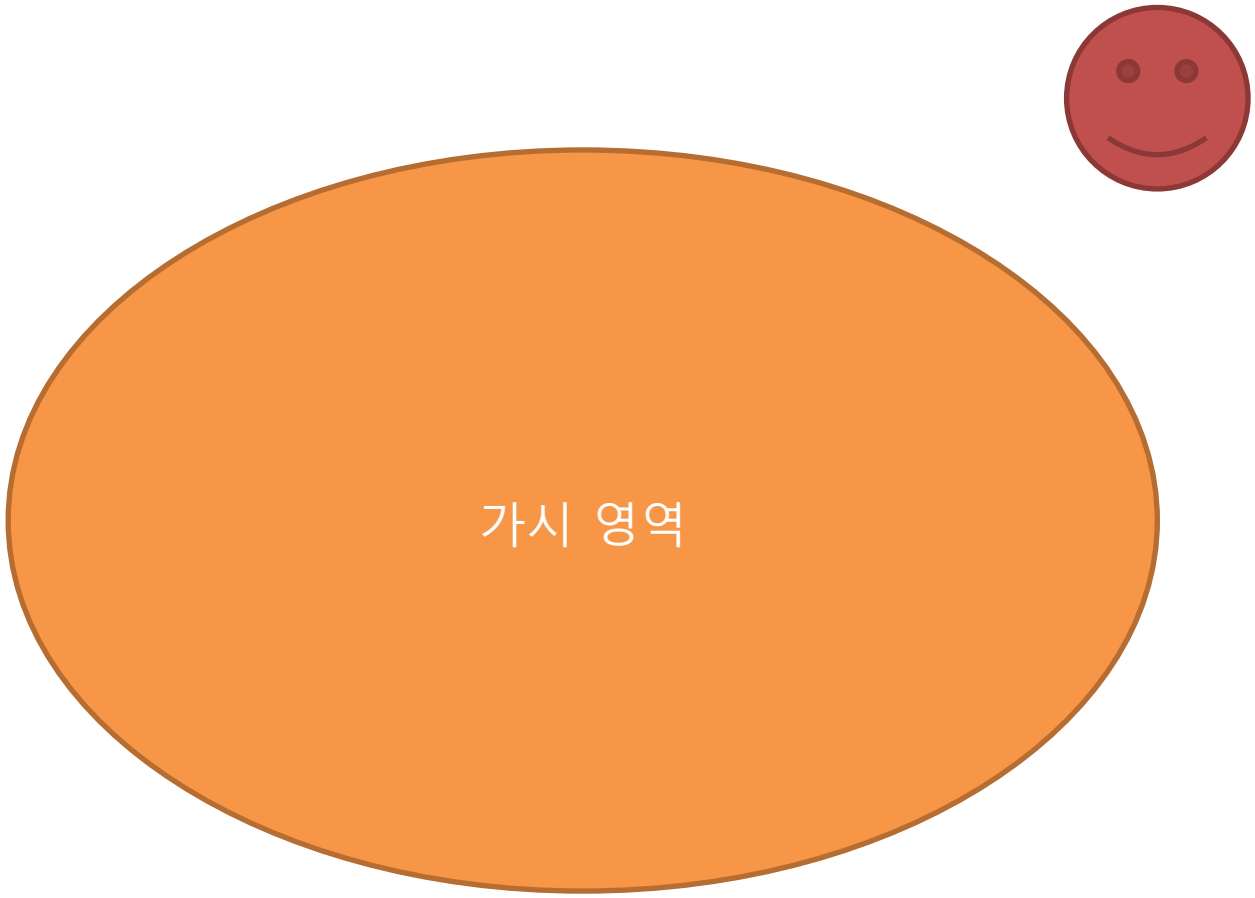
클라 내 데이터 구조



서버 내 데이터 구조


Hero 1 Actor
Hero 2 Actor
Hero 3 Actor
Mob 1 Actor
Mob 2 Actor
Mob 3 Actor
Mob 4 Actor
RC 1
RC 2
RC 3

가시 영역 검사 및 캐릭터 등장소멸



Appear

ShowState

ShowState

ShowState

Disappear

클라의 local hero 이동 처리

- 키 반응, 지형 반응
  - KeyControlledTransitionCharacter_cl 자체가 처리해주고 있음

귀차니즘이라고 불러도 좋다
있는 것 잘 쓰는 것도 지혜

Local hero 상태를 서버 및 P2P 전송하기

```
// 일정 간격으로 자신의 위치를 서버와 p2p group member에게 보낸다.
m_broadcastStateCoolTime -= elapsedTime;
if (m_broadcastStateCoolTime < 0)
{
  m_broadcastStateCoolTime = BroadcastLocalHeroStateInterval;

  // UniqueID를 지정해서 과량 송신하는 경우 최종적으로 결정된 위치만 보내도록 한다.
  // 즉 스로틀링 기법을 쓴다.
  RmiContext rctx = RmiContext::UnreliableSend;
```

Local hero 상태를 서버 및 P2P 전송하기

```
// P2P 그룹에는 서버도 포함되어 있는 상태이다. 따라서 P2P 그룹에 보내더라도 서버 또한 수신을 하게 된다.
        // 서버는 이를 받아서 가시영역 동기화 연산에 쓸 것이다.
        VisVector_cl pos = m_localHero->GetPosition();
        VisVector_cl ort = m_localHero->GetOrientation();

        int localheroMotion = StateNameToMotionIndex(m_localHero->GetStateMachine()->GetActiveState()->GetName());

        m_C2CProxy.P2P_LocalHero_Move(m_viewersGroupID, rctx, pos, VisVector_cl(0,0,0), EulerDegreeToRadian(ort.x), localheroMotion);
    }
```

Local hero 상태를 받아 처리하기 (클라)

```
// RMI 핸들러. 자세한 것은 PIDL 파일을 참고할 것.
DEFRMI_SynchWorldC2C_P2P_LocalHero_Move(CWorld)
{
  CActorPtr_C remoteHero = GetRemoteHeroByClientID(remote);
  if (remoteHero)
  {
    // 캐릭터의 위치,속도,방향 갱신한다.
    remoteHero->m_positionFollower.SetTargetPosition(position);
    remoteHero->m_positionFollower.SetTargetVelocity(velocity);

    remoteHero->m_directionFollower.SetTargetAngle(direction);

    // 모션
    remoteHero->m_motion = motion;
  }

  return true;
}
```

Local hero 상태를 받아 처리하기 (서버)

```
// 서버이지만 P2P_LocalHero_Move를 받아야 합니다. 가시 영역 필터링을 위해서입니다.
DEFRMI_SynchWorldC2C_P2P_LocalHero_Move(CSynchWorldServerDlg)
{
  CriticalSectionLock lock(m_cs, true);

  CRemoteClient_S* rc = GetRemoteClient(remote);
  if (!rc)
    return true;

  /* 캐릭의 상태를 업데이트한다. 하지만, 브로드캐스트는 하지 않는다.
  브로드캐스트는 P2P로 진행되기 때문이다. ProudNet은 P2P group 맺기 요청 직후
  즉시 P2P 통신이 가능하므로 (참고: ProudNet 소개 문서) 이런 방식이 안전하다. */
  if (rc->m_hero)
  {
    rc->m_hero->m_motion = motion;
    rc->m_hero->m_direction = direction;
    rc->m_hero->m_position = position;
    rc->m_hero->m_velocity = velocity;
  }

  return true;
```

```
	else
	{
		int delIndex = m_hero->m_viewers.FindByValue(otherRemote);
		if (delIndex >= 0)
		{
			// 목록에서 제거. P2P 그룹에서도 제거, 사라짐 신고 메시지 송신.
			m_hero->m_viewers[delIndex]->m_tangibles.RemoveOneByValue(m_hero);
			m_hero->m_viewers.RemoveAt(delIndex);

			m_owner->m_server->LeaveP2PGroup(otherRemote->m_hostID, m_hero-
>m_viewersGroupID);
			m_owner->m_S2CProxy.RemoteHero_Disappear(otherRemote->m_hostID,
RmiContext::ReliableSend, m_hostID);
		}
	}
```

클라에서 타 캐릭들의 위치 보정 및 렌더링

```
void CActor_C::FrameMove_RemoteActor( double elapsedTime )
{
  m_positionFollower.FrameMove(elapsedTime);
  m_directionFollower.FrameMove(elapsedTime);

  //Grab the remote values (here just from the other entity)
  VisVector_cl remotePosition;
  remotePosition.x = (float)m_positionFollower.GetFollowerPosition().x;
  remotePosition.y = (float)m_positionFollower.GetFollowerPosition().y;
  remotePosition.z = (float)m_positionFollower.GetFollowerPosition().z;

  VisVector_cl remoteOrientation =
VisVector_cl((float)RadianToEulerDegree(m_directionFollower.GetFollowe
rAngle()), 0, 0);
```

```
//first update position and orientation
  m_transitionCharacter->SetPosition(remotePosition);
  m_transitionCharacter->SetOrientation(remoteOrientation);

  // 모션
  m_transitionCharacter->GetStateMachine()-
>SetState(MotionIndexToStateName(m_motion));
}
```

서버에서 가시 영역 검사 및 P2P 갱신

```
        if (m_owner->IsActorVisibleToRemote(m_hero, otherRemote))
        {
          if (m_hero->m_viewers.FindByValue(otherRemote) < 0)
          {
            // 목록에 추가, P2P 그룹에도 추가, 출현 신고 메시지 송신.
            m_hero->m_viewers.Add(otherRemote);
            otherRemote->m_tangibles.Add(m_hero);

            m_owner->m_server->JoinP2PGroup(otherRemote->m_hostID, m_hero-
>m_viewersGroupID, ByteArray());
            m_owner->m_S2CProxy.RemoteHero_Appear(otherRemote->m_hostID,
RmiContext::ReliableSend,
              m_hero->GetID(), m_hostID, m_hero->m_name, m_hero->m_position,
m_hero->m_velocity, m_hero->m_direction, m_hero->m_motion);
          }
        }
```

이에 따라 처리되는 타 캐릭터 등장 및 소멸

```
// RMI 핸들러. 자세한 것은 PIDL 파일을 참고할 것.
DEFRMI_SynchWorldS2C_RemoteHero_Appear(CWorld)
{
  // 자기 캐릭터에 대해서 날아온 것은 무시한다.
  if (remote != m_netClient->GetLocalHostID())
  {
    // 타 캐릭터에 대한 인스턴스를 찾거나 만든다.
    CActorPtr_C remoteHero = GetRemoteHeroByClientID(heroOwnerID);
    if (!remoteHero)
    {
      remoteHero = CActorPtr_C(new CActor_C(Hero, heroID, position));
      remoteHero->m_ownerID = heroOwnerID;
      remoteHero->m_name = heroName;
      m_remoteHeroes.Add(remoteHero->m_ownerID, remoteHero);
    }
```

```
// 초기 캐릭터 값을 세팅한다.
    remoteHero->m_positionFollower.SetTargetPosition(position);
    remoteHero->m_positionFollower.SetTargetVelocity(velocity);
    remoteHero->m_directionFollower.SetTargetAngle(direction);

    // 이것도 설정해야 한다. 왜냐하면 바로 갓 등장한 것이기 때문이다.
    remoteHero->m_positionFollower.EnableAutoFollowDuration(true);

    remoteHero->m_positionFollower.SetFollowerPosition(position);
    remoteHero->m_positionFollower.SetFollowerVelocity(velocity);
```

```
// 각도에 대한 값도 세팅한다.
    remoteHero->m_directionFollower.SetFollowerAngleVelocity(D3DX_PI /
180 * 330);
    remoteHero->m_directionFollower.SetFollowerAngle(direction);
    remoteHero->m_directionFollower.SetTargetAngle(direction);

    // 모션
    remoteHero->m_motion = motion;
  }
  return true;
}
```

CActor_C.ctor

```
CActor_C::CActor_C(ActorType type, int actorID, const VisVector_cl
&characterOrigin)
{
  m_actorID = actorID;

  // Setup the synced character
  m_transitionCharacter = (TransitionCharacter_cl *)
Vision::Game.CreateEntity( GetEntityClassNameFromType(type),
characterOrigin );
  m_transitionCharacter->SetOrientation( 180.f, 0.f, 0.f );
  m_transitionCharacter-
>SetEntityKey("TransitionCharacterEntity_synced");
```

```
// 아직 다른 종류의 캐릭터를 로딩하는 방법을 모르므로, 몹은 좀 더 덩치만 키우자. 무섭게.
  if(type == Mob)
  {
    m_transitionCharacter->SetScaling(m_transitionCharacter->GetScaling() * 1.3f);
  }

  // We want to keep the vertex position in a separate collision buffer, so we don't need to skin the
  // model again for stencil shadows.
  m_transitionCharacter->GetAnimConfig()->SetFlags(m_transitionCharacter->GetAnimConfig()->GetFlags() | KEEP_EXTRA_COLLISIONBUFFER);
}
```

몹 추가

- 서버에서 몹 AI
  - 가까이 있는 HERO를 찾아 좀비처럼 천천히 다가가기
- 몹 상태를 서버에서 계속 전달

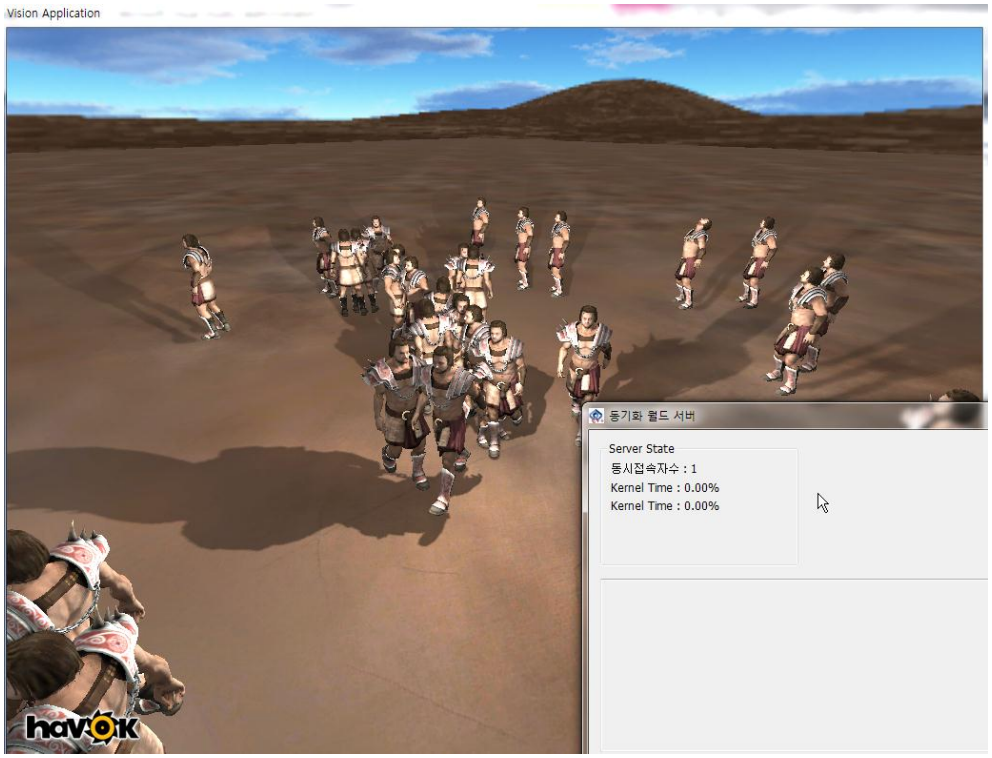
몹 AI

```
// 가장 가까운 hero를 찾아 추격한다. 단, 너무 멀면 추격 안함.
CActor_S* hero = m_owner->GetNearstHeroInRange(m_position, ViewportRadius * 0.4);
if(hero != NULL)
{
  VisVector_cl moveTo = hero->m_position - m_position;
  VisVector_cl moveTo2 = moveTo;
  moveTo.Normalize();
  moveTo *= 50 * float(elapsedTime); // 이동 속도
  m_position += moveTo;

  // 방향
  m_direction = atan2f(moveTo2.y, moveTo2.x) + D3DX_PI;
  if(m_direction > 2* D3DX_PI)
  {
    m_direction -=2*D3DX_PI;
  }

  // 모션
  m_motion = Walk;
}
```

몹 이동

```
// 일정 시간마다 자신의 상태를 멀티플레이 동기화한다.
CFastArray<HostID> sendToList;
sendToList.SetCapacity(m_viewers.Count);

for (int i=0;i<m_viewers.Count;i++)
{
  CRemoteClient_S* sendTo = m_viewers[i];
  sendToList.Add(sendTo->m_hostID);
}

m_owner->m_S2CProxy.Mob_ShowState(sendToList.GetData(),
sendToList.GetCount(), RmiContext::UnreliableSend, GetID(), m_position,
m_direction, m_motion);

m_sendSelfStateCoolTime = BroadcastLocalHeroStateInterval;
```

Show me ~~the money~~ the demo



서버 한대당 최대 동접 1~2만
몹 1000마리

아직 못 끝내 아쉬운 점

- 너무 뒤늦게 강연 준비
  (다들 일이 많아서 위임도 불가... 개발자 증원중! 입사지원하삼!)

- 몹이 이상한 방향으로 회전해 버리는 버그
  - 비전 엔진 사용법 이해 부족 탓
- 로그온, 디비 캐시, 분산 서버 풀세트 미장착
- 보안 개선 (서버에서 캐릭터 이동 판단의 공정성 검사 등)
- 쩌는 잉여력을 못 보여줘서 아쉽
  - 어제밤(kgc 화요일!)에 리니지 이터널 기사 읽고 감동... 아 저런거 넣고 싶은데...

- 나중에 잘 정리해서 공개하겠음

Twitter @imays

감사합니다!