**Multiplayer Assembly Package  
Tutorials  
Pre-Game Lobby**

A pre-game lobby is pretty much what it sounds like, a lobby in which players are placed in prior to the game’s start. This usually contains information regarding each individual player and the game settings. This example will show you a basic pre-game lobby setup containing these features.

Inside the Tutorial Files folder you will see the gui file for the pre-game lobby, place this in your gui folder and execute it in the scripts/client/init.cs file.

There are two other files in there, preGameLobby\_client.cs and preGameLobby\_server.cs, place them in their respective script folder and execute those files as well.

These will give your client access to push the lobby and receive update messages from the server, and on the server side, the server will send info regarding all of the players and the game settings down.

There are a few edits that need to be made in order for the lobby to function properly.

The first of these files is: core/scripts/server/server.cs. Scroll down to function createServer and change the contents to read as follows:

function createServer(%serverType, %level) {

$Server::Session++;

if (%level $= "") {

error("createServer(): level name unspecified");

return false;

}

%level = makeRelativePath(%level, getWorkingDirectory());

destroyServer();

$missionSequence = 0;

$Server::PlayerCount = 0;

$Server::ServerType = %serverType;

$Server::LoadFailMsg = "";

$Physics::isSinglePlayer = true;

// Setup for multi-player, the network must have been

// initialized before now.

if (%serverType $= "MultiPlayer") {

$Physics::isSinglePlayer = false;

echo("Starting multiplayer mode");

// Make sure the network port is set to the correct pref.

portInit($Pref::Server::Port);

allowConnections(true);

if ($pref::Net::DisplayOnMaster !$= "Never") {

schedule(0, 0, startHeartbeat);

}

// Create the ServerGroup that will persist for the lifetime of the server.

new SimGroup(ServerGroup);

// Load up any core datablocks

exec("core/art/datablocks/datablockExec.cs");

// Let the game initialize some things now that the

// the server has been created

onServerCreated();

$Server::InPreGame = true;

$Server::LevelSelection = %level;

$Server::isMultiplayerServer = 1;

//Phantom139: Don't load just yet!

//loadMission(%level, true);

}

else {

// Create the ServerGroup that will persist for the lifetime of the server.

new SimGroup(ServerGroup);

// Load up any core datablocks

exec("core/art/datablocks/datablockExec.cs");

// Let the game initialize some things now that the

// the server has been created

onServerCreated();

$Server::isMultiplayerServer = 0;

//Phantom139: We load if it's single player

loadMission(%level, true);

}

return true;

}

This will have all multiplayer servers start in a pre-game state without loading the mission while the single player games will just launch directly. Now we need to tell our server that when it is in the pre-game state to forward players to our lobby.

Open scripts/server/gameCore.cs and modify the last portion of function GameConnection::onConnect to read as follows:

//

echo("CADD: "@ %client @" "@ %client.getAddress());

//Phantom139: Check for Pre-Game

if($Server::InPreGame) {

PreGameDetails.addToPreGameLobby(%client);

}

else {

// If the mission is running, go ahead download it to the client

if ($missionRunning) {

%client.loadMission();

}

else if ($Server::LoadFailMsg !$= "") {

messageClient(%client, 'MsgLoadFailed', $Server::LoadFailMsg);

}

}

$Server::PlayerCount++;

And right below it in the onDrop function, add this block of code:

if($Server::InPreGame) {

PreGameDetails.removeFromPreGameLobby(%client);

}

The last thing we need to do is add functioning to our game itself to forward back to the pre-game lobby after the game ends.

Scroll down to GameCore::cycleGame and modify the function as follows:

function GameCore::cycleGame(%game)

{

if (%game.allowCycling)

{

// Cycle to the next mission

cycleGame();

}

else

{

//Phantom139: We don’t want to kill the entire server, push back to lobby:

if($Server::isMultiplayerServer) {

returnToLobbyPostGame();

for (%clientIndex = 0; %clientIndex < ClientGroup.getCount(); %clientIndex++) {

%cl = ClientGroup.getObject(%clientIndex);

commandToClient(%cl, ‘ReturnLobby’);

}

else {

// We're done with the whole game

endMission();

// Destroy server to remove all connected clients after they've seen the

// end game GUI.

schedule($Game::EndGamePause \* 1000, 0, "gameCoreDestroyServer", $Server::Session);

}

}

}

And that covers this setup. As you can see, while this is a pretty basic setup, there is room for improvement. A challenge in most cases would be to update the lobby to support administrative option setup. I recommend looking at the start game code to get a general starting point for that functioning.