**Object->InGameElements->Character:End Step**

if(global.isHost && hp<=0) {

var assistant;

assistant = secondToLastDamageDealer;

if (lastDamageDealer and instance\_exists(lastDamageDealer))

if (lastDamageDealer.object)

if (lastDamageDealer.object.healer)

assistant = lastDamageDealer.object.healer;

if player.object\_index == BotPlayer

{

with player

{

BotInit()// Reset everything after death

}

}

//Added Code

if player.object\_index == MyPlayer

{

with player

{

MyBotInit()// Reset everything after death

}

}

//End Added Code

sendEventPlayerDeath(player, lastDamageDealer, assistant, lastDamageSource);

doEventPlayerDeath(player, lastDamageDealer, assistant, lastDamageSource);

with(GameServer) {

ServerBalanceTeams();

}

exit;

}

if(hp>maxHp) {

hp=maxHp;

}

if(((aimDirection+270) mod 360)>180) {

image\_xscale=1;

currentWeapon.image\_xscale=1;

currentWeapon.image\_angle = aimDirection;

} else {

image\_xscale=-1;

currentWeapon.image\_xscale=-1;

currentWeapon.image\_angle = aimDirection+180;

}

currentWeapon.x=round(x);

currentWeapon.y=round(y);

// Limit people to the area of the room to prevent the

// "Falling through the floors" issue.

if(x<0) {

x=0;

}

if(x>map\_width()){

x = map\_width();

}

if(y<0) {

y = 0;

}

if(y>map\_height()){

y = map\_height();

}

if(!place\_free(x,y+1))

moveStatus=0

// Cloak

if (cloak) {

cloakAlpha = max(cloakAlpha - 0.05, 0);

} else {

cloakAlpha = min(cloakAlpha + 0.05, 1);

}

// Taunts

if (taunting)

{

tauntindex += tauntspeed\*0.1;

if tauntindex >= tauntend taunting=false;

}

//sandvich

if (omnomnomnom)

{

omnomnomnomindex += 0.25;

image\_xscale=xscale;

if hp <= maxHp hp += 0.4;

if omnomnomnomindex >= omnomnomnomend omnomnomnom=false;

}

//for things polling whether the character is on a medcabinet

onCabinet = place\_meeting(x, y, HealingCabinet);

// Last x/y position for death cam if player is dead

player.lastKnownx=x;

player.lastKnowny=y;

// Record the distance since the last node.

if lastNode != -1

{

distanceTraveled += abs(hspeed)

}

**Object->Map Elements->Gamemode-specific->Generator->Generator:Create**

maxHp = 2100;

hp = maxHp;

maxShieldHp = 300;

shieldHp = maxShieldHp;

maxShieldResistance = 0.75;

minShieldResistance = 0.25;

shieldResistance = maxShieldResistance;

isShieldRegenerating = false;

regenerationBuffer = 3 \* 30;

ubered=0;

lastDamageDealer = -1;

lastDamageSource = -1;

smokeCount = 0;

image\_speed = 0.3;

blastRadius = 400;

shieldIndex = 0;

//Added Code

player=-1;

//End Added Code

**Objects->Menus->HostOptionsController:User Event 10**

//Server respawn time calculator. Converts each second to a frame. (read: multiply by 30)

//Added Code

global.sqldump=file\_text\_open\_write(working\_directory + "\sqldump"+string(global.mybotgameplay)+".txt");

global.mybotgameplay+=1

//End Added Code

ini\_open("gg2.ini")

ini\_write\_real("MyBots", "GamePlay", global.mybotgameplay)

ini\_close()

if (global.Server\_RespawntimeSec == 0)

{

global.Server\_Respawntime = 1;

}

else

{

global.Server\_Respawntime = global.Server\_RespawntimeSec \* 30;

}

global.gameServer = instance\_create(0,0,GameServer);

**Objects->Networking->GameServer:Destroy**

if(global.useLobbyServer) {

sendLobbyUnreg();

}

ds\_list\_destroy(global.players);

socket\_destroy(global.tcpListener);

socket\_destroy(global.serverSocket);

buffer\_destroy(serverId);

if global.changedNodeMap

{

writeNodesToFile()

}

with(Player)

instance\_destroy();

with (BotPlayer)

instance\_destroy();

//Added Code

with (MyPlayer)

instance\_destroy();

//End Added Code

with(JoiningPlayer)

instance\_destroy();

with(PlayerControl)

instance\_destroy();

room\_goto\_fix(Menu);

**Objects->Projectiles->Rocket:Collision with Generator**

if fade==false {

if(team != other.team) {

other.alarm[0] = other.regenerationBuffer;

other.isShieldRegenerating = false;

//allow overkill to be applied directly to the target

if (hitDamage > other.shieldHp) {

other.hp -= hitDamage - other.shieldHp;

other.hp -= other.shieldHp \* other.shieldResistance;

other.shieldHp = 0;

} else {

other.hp -= hitDamage \* other.shieldResistance;

other.shieldHp -= hitDamage;

}

//Added Code

other.lastDamageDealer=ownerPlayer;

//End Added Code

characterHit = -1;

event\_user(5);

}

} else instance\_destroy();

**Objects->Projectiles->Shot:Collision with Generator**

if(other.team != team) {

other.alarm[0] = other.regenerationBuffer;

other.isShieldRegenerating = false;

//allow overkill to be applied directly to the target

if (hitDamage > other.shieldHp) {

other.hp -= hitDamage - other.shieldHp;

other.hp -= other.shieldHp \* other.shieldResistance;

other.shieldHp = 0;

} else {

other.hp -= hitDamage \* other.shieldResistance;

other.shieldHp -= hitDamage;

}

//Added Code

other.lastDamageDealer=ownerPlayer;

//End Added Code

instance\_destroy();

}

**Objects->Projectiles->BurningProjectile:Collision with Generator**

if (other.team != team) {

other.alarm[0] = other.regenerationBuffer;

other.isShieldRegenerating = false;

//allow overkill to be applied directly to the target

if (hitDamage > other.shieldHp) {

other.hp -= hitDamage - other.shieldHp;

other.hp -= other.shieldHp \* other.shieldResistance;

other.shieldHp = 0;

} else {

other.hp -= hitDamage \* other.shieldResistance;

other.shieldHp -= hitDamage;

}

//Added Code

other.lastDamageDealer=ownerPlayer;

//End Added Code

instance\_destroy()

}

**Objects->Projectiles->StabMask:Collision with Generator**

if (other.team != team) {

other.alarm[0] = other.regenerationBuffer;

other.isShieldRegenerating = false;

//allow overkill to be applied directly to the target

if (hitDamage/2 > other.shieldHp) {

other.hp -= hitDamage/2 - other.shieldHp;

other.hp -= other.shieldHp \* other.shieldResistance;

other.shieldHp = 0;

} else {

other.hp -= hitDamage/2 \* other.shieldResistance;

other.shieldHp -= hitDamage/2;

}

//Added Code

other.lastDamageDealer=ownerPlayer;

//End Added Code

instance\_destroy();

}

**Objects->Projectiles->Needle:Collision with Generator**

if(other.team != team) {

other.alarm[0] = other.regenerationBuffer;

other.isShieldRegenerating = false;

//allow overkill to be applied directly to the target

if (hitDamage > other.shieldHp) {

other.hp -= hitDamage - other.shieldHp;

other.hp -= other.shieldHp \* other.shieldResistance;

other.shieldHp = 0;

} else {

other.hp -= hitDamage \* other.shieldResistance;

other.shieldHp -= hitDamage;

}

//Added Code

other.lastDamageDealer=ownerPlayer;

//End Added Code

instance\_destroy();

}

**Scripts->GameServer->GameServerBeginStep**

if(serverbalance != 0)

balancecounter+=1;

// Register with Lobby Server every 30 seconds

if(global.useLobbyServer and (frame mod 900)==0)

sendLobbyRegistration();

frame += 1;

buffer\_clear(global.sendBuffer);

global.runningMapDownloads = 0;

global.mapBytesRemainingInStep = global.mapdownloadLimitBps/room\_speed;

with(JoiningPlayer)

if(state==STATE\_CLIENT\_DOWNLOADING)

global.runningMapDownloads += 1;

acceptJoiningPlayer();

with(JoiningPlayer)

serviceJoiningPlayer();

// Service all players

var i;

for(i=0; i<ds\_list\_size(global.players); i+=1)

{

var player;

player = ds\_list\_find\_value(global.players, i);

if player.object\_index == BotPlayer

{

if player.destroy

{

removePlayer(player);

ServerPlayerLeave(i);

i-=1;

}

else if player.object != -1

{

with player

{

getBotInput()

with object

{

event\_user(1)

}

}

}

continue

}

//Added Code

if player.object\_index == MyPlayer

{

if player.destroy

{

removePlayer(player);

ServerPlayerLeave(i);

i-=1;

}

else if player.object != -1

{

with player

{

getMyBotInput()

with object

{

event\_user(1)

}

}

}

continue

}

//End Added Code

if(socket\_has\_error(player.socket))

{

if global.botMode == 2 and player.team != TEAM\_SPECTATOR

{

with BotPlayer

{

destroy = 1// Destroy one bot. Just one.

break;

}

}

else if global.botMode == 4

{

CreateBot()

}

removePlayer(player);

ServerPlayerLeave(i);

i-=1;

}

else

processClientCommands(player, i);

}

if(syncTimer == 1 || ((frame mod 3600)==0) || global.setupTimer == 180)

{

serializeState(CAPS\_UPDATE, global.sendBuffer);

syncTimer = 0;

}

if((frame mod 7) == 0)

serializeState(QUICK\_UPDATE, global.sendBuffer);

else

serializeState(INPUTSTATE, global.sendBuffer);

if(impendingMapChange > 0)

impendingMapChange -= 1; // countdown until a map change

if(global.winners != -1 and !global.mapchanging)

{

if(global.winners == TEAM\_RED and global.currentMapArea < global.totalMapAreas)

{

global.nextMap = global.currentMap;

global.currentMapArea += 1;

}

else

{

global.currentMapIndex += 1;

global.currentMapArea = 1;

if(global.currentMapIndex == ds\_list\_size(global.map\_rotation))

global.currentMapIndex = 0;

global.nextMap = ds\_list\_find\_value(global.map\_rotation, global.currentMapIndex);

}

global.mapchanging = true;

impendingMapChange = 300; // in 300 frames (ten seconds), we'll do a map change

write\_ubyte(global.sendBuffer, MAP\_END);

write\_ubyte(global.sendBuffer, string\_length(global.nextMap));

write\_string(global.sendBuffer, global.nextMap);

write\_ubyte(global.sendBuffer, global.winners);

write\_ubyte(global.sendBuffer, global.currentMapArea);

if(!instance\_exists(ScoreTableController))

instance\_create(0,0,ScoreTableController);

instance\_create(0,0,WinBanner);

}

// if map change timer hits 0, do a map change

if(impendingMapChange == 0)

{

if global.changedNodeMap

{

writeNodesToFile()

global.changedNodeMap = 0

}

global.mapchanging = false;

global.currentMap = global.nextMap;

if(file\_exists("Maps/" + global.currentMap + ".png"))

{ // if this is an external map, get the md5 and url for the map

global.currentMapMD5 = CustomMapGetMapMD5(global.currentMap);

room\_goto\_fix(CustomMapRoom);

}

else

{ // internal map, so at the very least, MD5 must be blank

global.currentMapMD5 = "";

if(gotoInternalMapRoom(global.currentMap) != 0)

{

show\_message("Error:#Map " + global.currentMap + " is not in maps folder, and it is not a valid internal map.#Exiting.");

game\_end();

}

}

ServerChangeMap(global.currentMap, global.currentMapMD5, global.sendBuffer);

impendingMapChange = -1;

with(Player)

{

if(global.currentMapArea == 1)

{

stats[KILLS] = 0;

stats[DEATHS] = 0;

stats[CAPS] = 0;

stats[ASSISTS] = 0;

stats[DESTRUCTION] = 0;

stats[STABS] = 0;

stats[HEALING] = 0;

stats[DEFENSES] = 0;

stats[INVULNS] = 0;

stats[BONUS] = 0;

stats[DOMINATIONS] = 0;

stats[REVENGE] = 0;

stats[POINTS] = 0;

roundStats[KILLS] = 0;

roundStats[DEATHS] = 0;

roundStats[CAPS] = 0;

roundStats[ASSISTS] = 0;

roundStats[DESTRUCTION] = 0;

roundStats[STABS] = 0;

roundStats[HEALING] = 0;

roundStats[DEFENSES] = 0;

roundStats[INVULNS] = 0;

roundStats[BONUS] = 0;

roundStats[DOMINATIONS] = 0;

roundStats[REVENGE] = 0;

roundStats[POINTS] = 0;

team = TEAM\_SPECTATOR;

}

timesChangedCapLimit = 0;

alarm[5]=1;

}

if global.botMode == 1 or global.botMode == 2

{

global.botChosenTeam = choose(TEAM\_RED, TEAM\_BLUE)

}

with(BotPlayer)

{

if(global.currentMapArea == 1)

{

if global.botMode == 2

{

removePlayer(id)

}

else

{

stats[KILLS] = 0;

stats[DEATHS] = 0;

stats[CAPS] = 0;

stats[ASSISTS] = 0;

stats[DESTRUCTION] = 0;

stats[STABS] = 0;

stats[HEALING] = 0;

stats[DEFENSES] = 0;

stats[INVULNS] = 0;

stats[BONUS] = 0;

stats[DOMINATIONS] = 0;

stats[REVENGE] = 0;

stats[POINTS] = 0;

roundStats[KILLS] = 0;

roundStats[DEATHS] = 0;

roundStats[CAPS] = 0;

roundStats[ASSISTS] = 0;

roundStats[DESTRUCTION] = 0;

roundStats[STABS] = 0;

roundStats[HEALING] = 0;

roundStats[DEFENSES] = 0;

roundStats[INVULNS] = 0;

roundStats[BONUS] = 0;

roundStats[DOMINATIONS] = 0;

roundStats[REVENGE] = 0;

roundStats[POINTS] = 0;

team = GetBotTeam()

ServerPlayerChangeteam(ds\_list\_find\_index(global.players, id), team, global.eventBuffer)

BotInit()

}

}

timesChangedCapLimit = 0;

alarm[5]=1;

}

}

var i;

for(i=1; i<ds\_list\_size(global.players); i+=1)

{

var player;

player = ds\_list\_find\_value(global.players, i);

if player.object\_index == BotPlayer

{

continue;

}

//Added Code

if player.object\_index == MyPlayer

{

continue;

}

//End Added Code

write\_buffer(player.socket, global.eventBuffer);

write\_buffer(player.socket, global.sendBuffer);

socket\_send(player.socket);

}

buffer\_clear(global.eventBuffer);

**Scripts->GameServer->processClientCommands**

var player, playerId, commandLimitRemaining;

player = argument0;

playerId = argument1;

// To prevent players from flooding the server, limit the number of commands to process per step and player.

commandLimitRemaining = 10;

with(player) {

if(!variable\_local\_exists("commandReceiveState")) {

// 0: waiting for command byte.

// 1: waiting for command data length (1 byte)

// 2: waiting for command data.

commandReceiveState = 0;

commandReceiveExpectedBytes = 1;

commandReceiveCommand = 0;

}

}

while(commandLimitRemaining > 0) {

var socket;

socket = player.socket;

if(!tcp\_receive(socket, player.commandReceiveExpectedBytes)) {

return 0;

}

switch(player.commandReceiveState)

{

case 0:

player.commandReceiveCommand = read\_ubyte(socket);

switch(commandBytes[player.commandReceiveCommand]) {

case commandBytesInvalidCommand:

// Invalid byte received. Wait for another command byte.

break;

case commandBytesPrefixLength1:

player.commandReceiveState = 1;

player.commandReceiveExpectedBytes = 1;

break;

default:

player.commandReceiveState = 2;

player.commandReceiveExpectedBytes = commandBytes[player.commandReceiveCommand];

break;

}

break;

case 1:

player.commandReceiveState = 2;

player.commandReceiveExpectedBytes = read\_ubyte(socket);

break;

case 2:

player.commandReceiveState = 0;

player.commandReceiveExpectedBytes = 1;

commandLimitRemaining -= 1;

switch(player.commandReceiveCommand)

{

case PLAYER\_LEAVE:

socket\_destroy(player.socket);

player.socket = -1;

if global.botMode == 2

{

with BotPlayer

{

destroy = 1

break;

}

}

else if global.botMode == 3

{

CreateBot()

}

break;

case PLAYER\_CHANGECLASS:

var class;

class = read\_ubyte(socket);

if(getCharacterObject(player.team, class) != -1)

{

if(player.object != -1)

{

with(player.object)

{

if (collision\_point(x,y,SpawnRoom,0,0) < 0)

{

if (lastDamageDealer == -1 || lastDamageDealer == player)

{

sendEventPlayerDeath(player, player, noone, BID\_FAREWELL);

doEventPlayerDeath(player, player, noone, BID\_FAREWELL);

}

else

{

var assistant;

assistant = secondToLastDamageDealer;

if (lastDamageDealer.object != -1)

if (lastDamageDealer.object.healer != -1)

assistant = lastDamageDealer.object.healer;

sendEventPlayerDeath(player, lastDamageDealer, assistant, FINISHED\_OFF);

doEventPlayerDeath(player, lastDamageDealer, assistant, FINISHED\_OFF);

}

}

else

instance\_destroy();

}

}

else if(player.alarm[5]<=0)

player.alarm[5] = 1;

player.class = class;

ServerPlayerChangeclass(playerId, player.class, global.sendBuffer);

}

break;

case PLAYER\_CHANGETEAM:

var newTeam, balance, redSuperiority;

newTeam = read\_ubyte(socket);

if global.botMode == 1 or global.botMode == 2

{

balance = global.botChosenTeam// Don't let humans go there

}

else

{

redSuperiority = 0 //calculate which team is bigger

with(Player)

{

if(team == TEAM\_RED)

redSuperiority += 1;

else if(team == TEAM\_BLUE)

redSuperiority -= 1;

}

with(BotPlayer)

{

if(team == TEAM\_RED)

redSuperiority += 1;

else if(team == TEAM\_BLUE)

redSuperiority -= 1;

}

//Added Code

with(MyPlayer)

{

if(team == TEAM\_RED)

redSuperiority += 1;

else if(team == TEAM\_BLUE)

redSuperiority -= 1;

}

//End Added Code

if(redSuperiority > 0)

balance = TEAM\_RED;

else if(redSuperiority < 0)

balance = TEAM\_BLUE;

else

balance = -1;

}

if(balance != newTeam)

{

if(getCharacterObject(newTeam, player.class) != -1 or newTeam==TEAM\_SPECTATOR)

{

if(player.object != -1)

{

with(player.object)

{

if (lastDamageDealer == -1 || lastDamageDealer == player)

{

sendEventPlayerDeath(player, player, noone, BID\_FAREWELL);

doEventPlayerDeath(player, player, noone, BID\_FAREWELL);

}

else

{

var assistant;

assistant = secondToLastDamageDealer;

if (lastDamageDealer.object != -1)

if (lastDamageDealer.object.healer != -1)

assistant = lastDamageDealer.object.healer;

sendEventPlayerDeath(player, lastDamageDealer, assistant, FINISHED\_OFF);

doEventPlayerDeath(player, lastDamageDealer, assistant, FINISHED\_OFF);

}

}

player.alarm[5] = global.Server\_Respawntime;

}

else if(player.alarm[5]<=0)

player.alarm[5] = 1;

if player.team != newTeam and global.botMode == 2

{

if newTeam == TEAM\_SPECTATOR

{

with BotPlayer

{

destroy = 1

break;

}

}

else

{

CreateBot()

}

}

player.team = newTeam;

ServerPlayerChangeteam(playerId, player.team, global.sendBuffer);

}

}

break;

case CHAT\_BUBBLE:

var bubbleImage;

bubbleImage = read\_ubyte(socket);

if(global.aFirst) {

bubbleImage = 0;

}

write\_ubyte(global.sendBuffer, CHAT\_BUBBLE);

write\_ubyte(global.sendBuffer, playerId);

write\_ubyte(global.sendBuffer, bubbleImage);

setChatBubble(player, bubbleImage);

break;

case BUILD\_SENTRY:

if(player.object != -1)

{

if(player.class == CLASS\_ENGINEER

and collision\_circle(player.object.x, player.object.y, 50, Sentry, false, true) < 0

and player.object.nutsNBolts == 100 and (collision\_point(player.object.x,player.object.y,SpawnRoom,0,0) < 0)

and player.sentry == -1 and !player.object.onCabinet)

{

buildSentry(player);

write\_ubyte(global.sendBuffer, BUILD\_SENTRY);

write\_ubyte(global.sendBuffer, playerId);

}

}

break;

case DESTROY\_SENTRY:

if(player.sentry != -1) {

with(player.sentry) {

instance\_destroy();

}

}

player.sentry = -1;

break;

case DROP\_INTEL:

if(player.object != -1) {

write\_ubyte(global.sendBuffer, DROP\_INTEL);

write\_ubyte(global.sendBuffer, playerId);

with player.object event\_user(5);

}

break;

case OMNOMNOMNOM:

if(player.object != -1) {

if(player.humiliated == 0

&& player.object.taunting==false

&& player.object.omnomnomnom==false

&& player.class==CLASS\_HEAVY) {

write\_ubyte(global.sendBuffer, OMNOMNOMNOM);

write\_ubyte(global.sendBuffer, playerId);

with(player.object) {

omnomnomnom=true;

if player.team == TEAM\_RED {

omnomnomnomindex=0;

omnomnomnomend=31;

} else if player.team==TEAM\_BLUE {

omnomnomnomindex=32;

omnomnomnomend=63;

}

xscale=image\_xscale;

}

}

}

break;

case TOGGLE\_ZOOM:

if player.object != -1 {

if player.class == CLASS\_SNIPER {

write\_ubyte(global.sendBuffer, TOGGLE\_ZOOM);

write\_ubyte(global.sendBuffer, playerId);

toggleZoom(player.object);

}

}

break;

case PLAYER\_CHANGENAME:

var nameLength;

nameLength = socket\_receivebuffer\_size(socket);

if(nameLength > MAX\_PLAYERNAME\_LENGTH)

{

write\_ubyte(player.socket, KICK);

write\_ubyte(player.socket, KICK\_NAME);

socket\_destroy(player.socket);

player.socket = -1;

}

else

{

with(player)

{

if(variable\_local\_exists("lastNamechange"))

if(current\_time - lastNamechange < 1000)

break;

lastNamechange = current\_time;

name = read\_string(socket, nameLength);

if(string\_count("#",name) > 0)

{

name = "I <3 Bacon";

}

write\_ubyte(global.sendBuffer, PLAYER\_CHANGENAME);

write\_ubyte(global.sendBuffer, playerId);

write\_ubyte(global.sendBuffer, string\_length(name));

write\_string(global.sendBuffer, name);

}

}

break;

case INPUTSTATE:

if(player.object != -1)

{

with(player.object)

{

keyState = read\_ubyte(socket);

netAimDirection = read\_ushort(socket);

aimDirection = netAimDirection\*360/65536;

event\_user(1);

}

}

break;

case I\_AM\_A\_HAXXY\_WINNER:

write\_ubyte(socket, HAXXY\_CHALLENGE\_CODE);

player.challenge = "";

repeat(16)

player.challenge += chr(irandom\_range(1,255));

write\_string(socket, player.challenge);

break;

case HAXXY\_CHALLENGE\_RESPONSE:

var answer, i, challengeSent;

with(player)

challengeSent = variable\_local\_exists("challenge");

if(!challengeSent)

break;

answer = "";

for(i=1;i<=16;i+=1)

answer += chr(read\_ubyte(socket) ^ ord(string\_char\_at(player.challenge, i)));

if(HAXXY\_PUBLIC\_KEY==md5(answer)) {

player.isHaxxyWinner = true;

} else {

socket\_destroy\_abortive(player.socket);

player.socket = -1;

}

break;

}

break;

}

}

**Scripts->GameServer->sendLobbyRegistration**

var noOfPlayers;

//Editted Code

noOfPlayers = ds\_list\_size(global.players)-instance\_number(BotPlayer)-instance\_number(MyPlayer);

//End Editted Code

if(global.dedicatedMode)

noOfPlayers -= 1;

var lobbyBuffer;

lobbyBuffer = buffer\_create();

set\_little\_endian(lobbyBuffer, false);

parseUuid("b5dae2e8-424f-9ed0-0fcb-8c21c7ca1352", lobbyBuffer); // Message Type "register"

write\_buffer(lobbyBuffer, GameServer.serverId);

write\_buffer(lobbyBuffer, global.gg2lobbyId);

write\_ubyte(lobbyBuffer, 0); // TCP

write\_ushort(lobbyBuffer, global.hostingPort);

write\_ushort(lobbyBuffer, global.playerLimit);

write\_ushort(lobbyBuffer, noOfPlayers);

write\_ushort(lobbyBuffer, instance\_number(BotPlayer)); // Number of bots

//Added Code

write\_ushort(lobbyBuffer, instance\_number(MyPlayer)); // Number of mybots

//End Added Code

if(global.serverPassword != "")

write\_ushort(lobbyBuffer, 1);

else

write\_ushort(lobbyBuffer, 0);

write\_ushort(lobbyBuffer, 7); // Number of Key/Value pairs that follow

writeKeyValue(lobbyBuffer, "name", global.serverName);

writeKeyValue(lobbyBuffer, "game", "Tempest Bot - Gang Garrison 2");

writeKeyValue(lobbyBuffer, "game\_short", "TempestBot");

writeKeyValue(lobbyBuffer, "game\_ver", GAME\_VERSION\_STRING);

writeKeyValue(lobbyBuffer, "game\_url", "http://www.ganggarrison.com/forums/index.php?topic=25580.msg800919#msg800919");

writeKeyValue(lobbyBuffer, "map", global.currentMap);

write\_ubyte(lobbyBuffer, string\_length("protocol\_id"));

write\_string(lobbyBuffer, "protocol\_id");

write\_ushort(lobbyBuffer, 16);

write\_buffer(lobbyBuffer, global.protocolUuid);

udp\_send(lobbyBuffer, LOBBY\_SERVER\_HOST, LOBBY\_SERVER\_PORT);

buffer\_destroy(lobbyBuffer);

**Script->basicRoomSetup**

room\_caption = global.currentMap;

global.startedGame = true;

global.totalMapAreas = 1+instance\_number(NextAreaO);

if global.totalMapAreas > 1 {

global.area[1] = 0;

for(i=2;i<=global.totalMapAreas;i+=1) {

global.area[i] = instance\_find(NextAreaO,i-2).y;

}

if global.currentMapArea == 1 {

with all if y > global.area[2] instance\_destroy();

}

else if global.currentMapArea < global.totalMapAreas {

with all if (y > global.area[global.currentMapArea+1] || y < global.area[global.currentMapArea]) && y > 0 instance\_destroy();

}

else if global.currentMapArea == global.totalMapAreas {

with all if y < global.area[global.currentMapArea] && y > 0 instance\_destroy();

}

}

offloadSpawnPoints();

with(Player) {

canSpawn = 1;

humiliated = 0;

}

with(BotPlayer) {

canSpawn = 1;

humiliated = 0;

}

//Added Code

with(MyPlayer) {

canSpawn = 1;

humiliated = 0;

}

//End Added Code

if instance\_exists(IntelligenceBaseBlue) || instance\_exists(IntelligenceBaseRed) || instance\_exists(IntelligenceRed) || instance\_exists(IntelligenceBlue) instance\_create(0,0,ScorePanel);

else if instance\_exists(GeneratorBlue) || instance\_exists(GeneratorRed) {

instance\_create(0,0,GeneratorHUD);

} else if instance\_exists(ArenaControlPoint) {

instance\_create(0,0,ArenaHUD);

if ArenaHUD.roundStart == 0 with Player canSpawn = 0;

}else if instance\_exists(KothControlPoint) {

instance\_create(0,0,KothHUD);

}else if instance\_exists(KothRedControlPoint) && instance\_exists(KothBlueControlPoint) {

instance\_create(0,0,DKothHUD);

} else if instance\_exists(ControlPoint) {

with ControlPoint event\_user(0);

instance\_create(0,0,ControlPointHUD);

}

instance\_create(0,0,TeamSelectController);

if !instance\_exists(KillLog) instance\_create(0,0,KillLog);

sound\_stop\_all();

if(global.ingameMusic) {

AudioControlPlaySong(global.IngameMusic, true);

}

instance\_create(map\_width()/2,map\_height()/2,Spectator);

global.redCaps = 0;

global.blueCaps = 0;

global.winners = -1;

LoadBotNodes()

**Script->game\_init**

{

instance\_create(0,0,RoomChangeObserver);

set\_little\_endian\_global(true);

if file\_exists("game\_errors.log") file\_delete("game\_errors.log");

var customMapRotationFile;

//import wav files for music

global.MenuMusic=sound\_add(choose("Music/menumusic1.wav","Music/menumusic2.wav","Music/menumusic3.wav","Music/menumusic4.wav","Music/menumusic5.wav","Music/menumusic6.wav"), 1, true);

global.IngameMusic=sound\_add("Music/ingamemusic.wav", 1, true);

global.FaucetMusic=sound\_add("Music/faucetmusic.wav", 1, true);

if(global.MenuMusic != -1)

sound\_volume(global.MenuMusic, 0.8);

if(global.IngameMusic != -1)

sound\_volume(global.IngameMusic, 0.8);

if(global.FaucetMusic != -1)

sound\_volume(global.FaucetMusic, 0.8);

global.sendBuffer = buffer\_create();

global.eventBuffer = buffer\_create();

global.tempBuffer = buffer\_create();

global.HudCheck = false;

global.map\_rotation = ds\_list\_create();

global.CustomMapCollisionSprite = -1;

global.botNameCounter = 1

//Added Code

global.mybotNameCounter = 1

//End Added Code

global.changedNodeMap = 0

window\_set\_region\_scale(-1, false);

ini\_open("gg2.ini");

global.playerName = ini\_read\_string("Settings", "PlayerName", "Player");

if string\_count("#",global.playerName) > 0 global.playerName = "Player";

global.playerName = string\_copy(global.playerName, 0, min(string\_length(global.playerName), MAX\_PLAYERNAME\_LENGTH));

global.fullscreen = ini\_read\_real("Settings", "Fullscreen", 0);

global.useLobbyServer = ini\_read\_real("Settings", "UseLobby", 1);

global.hostingPort = ini\_read\_real("Settings", "HostingPort", 8190);

global.ingameMusic = ini\_read\_real("Settings", "IngameMusic", 1);

global.playerLimit = ini\_read\_real("Settings", "PlayerLimit", 10);

global.particles = ini\_read\_real("Settings", "Particles", PARTICLES\_NORMAL);

global.gibLevel = ini\_read\_real("Settings", "Gib Level", 3);

global.killCam = ini\_read\_real("Settings", "Kill Cam", 1);

global.monitorSync = ini\_read\_real("Settings", "Monitor Sync", 0);

if global.monitorSync == 1 set\_synchronization(true);

else set\_synchronization(false);

global.medicRadar = ini\_read\_real("Settings", "Healer Radar", 1);

global.showHealer = ini\_read\_real("Settings", "Show Healer", 1);

global.showHealing = ini\_read\_real("Settings", "Show Healing",1);

global.showHealthBar = ini\_read\_real("Settings", "Show Healthbar",0);

//user HUD settings

global.timerPos=ini\_read\_real("Settings","Timer Position", 0)

global.killLogPos=ini\_read\_real("Settings","Kill Log Position", 0)

global.kothHudPos=ini\_read\_real("Settings","KoTH HUD Position", 0)

global.clientPassword = "";

// for admin menu

customMapRotationFile = ini\_read\_string("Server", "MapRotation", "");

global.timeLimitMins = max(1, min(255, ini\_read\_real("Server", "Time Limit", 15)));

global.serverPassword = ini\_read\_string("Server", "Password", "");

global.mapRotationFile = customMapRotationFile;

global.dedicatedMode = ini\_read\_real("Server", "Dedicated", 0);

global.serverName = ini\_read\_string("Server", "ServerName", "My Server");

global.caplimit = max(1, min(255, ini\_read\_real("Server", "CapLimit", 5)));

global.caplimitBkup = global.caplimit;

global.autobalance = ini\_read\_real("Server", "AutoBalance",1);

global.Server\_RespawntimeSec = ini\_read\_real("Server", "Respawn Time", 5);

global.haxxyKey = ini\_read\_string("Haxxy", "SecretHaxxyKey", "");

// Bots

global.botNumber = ini\_read\_real("Bots", "Number (in total)", 0)

global.botMode = ini\_read\_real("Bots", "Fight mode", 0)

global.botNamePrefix = ini\_read\_string("Bots", "Bot Name Prefix", "")

global.botLearningMode = ini\_read\_real("Bots", "Bot learning", 0)

global.botClasses[CLASS\_SCOUT] = ini\_read\_real("Bots", "Runners enabled", 1)

global.botClasses[CLASS\_PYRO] = ini\_read\_real("Bots", "Firebugs enabled", 1)

global.botClasses[CLASS\_SOLDIER] = ini\_read\_real("Bots", "Rocketmen enabled", 1)

global.botClasses[CLASS\_HEAVY] = ini\_read\_real("Bots", "Overweights enabled", 1)

global.botClasses[CLASS\_MEDIC] = ini\_read\_real("Bots", "Healers enabled", 1)

global.botClasses[CLASS\_ENGINEER] = ini\_read\_real("Bots", "Constructors enabled", 1)

global.botClasses[CLASS\_SNIPER] = ini\_read\_real("Bots", "Riflemen enabled", 1)

global.botDebugMode = ini\_read\_real("Bots", "Debug Mode enabled", 0)

//Added Code

// MyBots

global.mybotNumber = ini\_read\_real("MyBots", "Number (in total)", 0)

global.mybotMode = ini\_read\_real("MyBots", "Team", 0)

global.mybotNamePrefix = ini\_read\_string("MyBots", "Bot Name Prefix", "")

global.mybotLearningMode = ini\_read\_real("MyBots", "Reinforcement learning", 1)

global.mybotClasses[CLASS\_SCOUT] = ini\_read\_real("MyBots", "Runners enabled", 1)

global.mybotClasses[CLASS\_PYRO] = ini\_read\_real("MyBots", "Firebugs enabled", 1)

global.mybotClasses[CLASS\_SOLDIER] = ini\_read\_real("MyBots", "Rocketmen enabled", 1)

global.mybotClasses[CLASS\_HEAVY] = ini\_read\_real("MyBots", "Overweights enabled", 1)

global.mybotClasses[CLASS\_MEDIC] = ini\_read\_real("MyBots", "Healers enabled", 1)

global.mybotClasses[CLASS\_ENGINEER] = ini\_read\_real("MyBots", "Constructors enabled", 1)

global.mybotClasses[CLASS\_SNIPER] = ini\_read\_real("MyBots", "Riflemen enabled", 1)

global.mybotgameplay = ini\_read\_real("MyBots", "GamePlay", 1)

//End Added Code

global.mapdownloadLimitBps = ini\_read\_real("Server", "Total bandwidth limit for map downloads in bytes per second", 50000);

global.currentMapArea=1;

global.totalMapAreas=1;

global.setupTimer=1800;

global.joinedServerName="";

ini\_write\_string("Settings", "PlayerName", global.playerName);

ini\_write\_real("Settings", "Fullscreen", global.fullscreen);

ini\_write\_real("Settings", "UseLobby", global.useLobbyServer);

ini\_write\_real("Settings", "HostingPort", global.hostingPort);

ini\_write\_real("Settings", "IngameMusic", global.ingameMusic);

ini\_write\_real("Settings", "PlayerLimit", global.playerLimit);

ini\_write\_real("Settings", "Particles", global.particles);

ini\_write\_real("Settings", "Gib Level", global.gibLevel);

ini\_write\_real("Settings", "Kill Cam", global.killCam);

ini\_write\_real("Settings", "Monitor Sync", global.monitorSync);

ini\_write\_real("Settings", "Healer Radar", global.medicRadar);

ini\_write\_real("Settings", "Show Healer", global.showHealer);

ini\_write\_real("Settings", "Show Healing", global.showHealing);

ini\_write\_real("Settings", "Show Healthbar", global.showHealthBar);

ini\_write\_real("Settings","Timer Position", global.timerPos)

ini\_write\_real("Settings","Kill Log Position", global.killLogPos)

ini\_write\_real("Settings","KoTH HUD Position", global.kothHudPos)

ini\_write\_string("Server", "MapRotation", customMapRotationFile);

ini\_write\_real("Server", "Dedicated", global.dedicatedMode);

ini\_write\_string("Server", "ServerName", global.serverName);

ini\_write\_real("Server", "CapLimit", global.caplimit);

ini\_write\_real("Server", "AutoBalance", global.autobalance);

ini\_write\_real("Server", "Respawn Time", global.Server\_RespawntimeSec);

ini\_write\_real("Server", "Total bandwidth limit for map downloads in bytes per second", global.mapdownloadLimitBps);

ini\_write\_real("Server", "Time Limit", global.timeLimitMins);

ini\_write\_string("Server", "Password", global.serverPassword);

ini\_write\_string("Haxxy", "SecretHaxxyKey", global.haxxyKey);

ini\_write\_real("Bots", "Number (in total)", global.botNumber)

ini\_write\_real("Bots", "Fight mode", global.botMode)

ini\_write\_real("Bots", "Bot Learning", global.botLearningMode)

ini\_write\_string("Bots", "Bot Name Prefix", global.botNamePrefix)

ini\_write\_real("Bots", "Runners enabled", global.botClasses[CLASS\_SCOUT])

ini\_write\_real("Bots", "Firebugs enabled", global.botClasses[CLASS\_PYRO])

ini\_write\_real("Bots", "Rocketmen enabled", global.botClasses[CLASS\_SOLDIER])

ini\_write\_real("Bots", "Overweights enabled", global.botClasses[CLASS\_HEAVY])

ini\_write\_real("Bots", "Healers enabled", global.botClasses[CLASS\_MEDIC])

ini\_write\_real("Bots", "Constructors enabled", global.botClasses[CLASS\_ENGINEER])

ini\_write\_real("Bots", "Riflemen enabled", global.botClasses[CLASS\_SNIPER])

ini\_write\_real("Bots", "Debug Mode enabled", global.botDebugMode)

//screw the 0 index we will start with 1

//map\_truefort

maps[1] = ini\_read\_real("Maps", "ctf\_truefort", 1);

//map\_2dfort

maps[2] = ini\_read\_real("Maps", "ctf\_2dfort", 2);

//map\_conflict

maps[3] = ini\_read\_real("Maps", "ctf\_conflict", 3);

//map\_classicwell

maps[4] = ini\_read\_real("Maps", "ctf\_classicwell", 4);

//map\_waterway

maps[5] = ini\_read\_real("Maps", "ctf\_waterway", 5);

//map\_orange

maps[6] = ini\_read\_real("Maps", "ctf\_orange", 6);

//map\_dirtbowl

maps[7] = ini\_read\_real("Maps", "cp\_dirtbowl", 7);

//map\_egypt

maps[8] = ini\_read\_real("Maps", "cp\_egypt", 8);

//arena\_montane

maps[9] = ini\_read\_real("Maps", "arena\_montane", 9);

//arena\_lumberyard

maps[10] = ini\_read\_real("Maps", "arena\_lumberyard", 10);

//gen\_destroy

maps[11] = ini\_read\_real("Maps", "gen\_destroy", 11);

//koth\_valley

maps[12] = ini\_read\_real("Maps", "koth\_valley", 12);

//koth\_corinth

maps[13] = ini\_read\_real("Maps", "koth\_corinth", 13);

//koth\_harvest

maps[14] = ini\_read\_real("Maps", "koth\_harvest", 14);

//dkoth\_atalia

maps[15] = ini\_read\_real("Maps", "dkoth\_atalia", 15);

//dkoth\_sixties

maps[16] = ini\_read\_real("Maps", "dkoth\_sixties", 16);

//Server respawn time calculator. Converts each second to a frame. (read: multiply by 30 :hehe:)

if (global.Server\_RespawntimeSec == 0)

{

global.Server\_Respawntime = 1;

}

else

{

global.Server\_Respawntime = global.Server\_RespawntimeSec \* 30;

}

// I have to include this, or the client'll complain about an unknown variable.

global.mapchanging = false;

ini\_write\_real("Maps", "ctf\_truefort", maps[1]);

ini\_write\_real("Maps", "ctf\_2dfort", maps[2]);

ini\_write\_real("Maps", "ctf\_conflict", maps[3]);

ini\_write\_real("Maps", "ctf\_classicwell", maps[4]);

ini\_write\_real("Maps", "ctf\_waterway", maps[5]);

ini\_write\_real("Maps", "ctf\_orange", maps[6]);

ini\_write\_real("Maps", "cp\_dirtbowl", maps[7]);

ini\_write\_real("Maps", "cp\_egypt", maps[8]);

ini\_write\_real("Maps", "arena\_montane", maps[9]);

ini\_write\_real("Maps", "arena\_lumberyard", maps[10]);

ini\_write\_real("Maps", "gen\_destroy", maps[11]);

ini\_write\_real("Maps", "koth\_valley", maps[12]);

ini\_write\_real("Maps", "koth\_corinth", maps[13]);

ini\_write\_real("Maps", "koth\_harvest", maps[14]);

ini\_write\_real("Maps", "dkoth\_atalia", maps[15]);

ini\_write\_real("Maps", "dkoth\_sixties", maps[16]);

ini\_close();

// parse the protocol version UUID for later use

global.protocolUuid = buffer\_create();

parseUuid(PROTOCOL\_UUID, global.protocolUuid);

global.gg2lobbyId = buffer\_create();

parseUuid(GG2\_LOBBY\_UUID, global.gg2lobbyId);

var a, IPRaw, portRaw;

doubleCheck=0;

global.launchMap = "";

for(a = 1; a <= parameter\_count(); a += 1)

{

if (parameter\_string(a) == "-dedicated")

{

global.dedicatedMode = 1;

}

else if (parameter\_string(a) == "-server")

{

IPRaw = parameter\_string(a+1);

if (doubleCheck == 1)

{

doubleCheck = 2;

}

else

{

doubleCheck = 1;

}

}

else if (parameter\_string(a) == "-port")

{

portRaw = parameter\_string(a+1);

if (doubleCheck == 1)

{

doubleCheck = 2;

}

else

{

doubleCheck = 1;

}

}

else if (parameter\_string(a) == "-map")

{

global.launchMap = parameter\_string(a+1);

global.dedicatedMode = 1;

}

}

if (doubleCheck == 2)

{

global.serverPort = real(portRaw);

global.serverIP = IPRaw;

global.isHost = false;

instance\_create(0,0,Client);

}

global.customMapdesginated = 0;

// if the user defined a valid map rotation file, then load from there

if(customMapRotationFile != "" && file\_exists(customMapRotationFile) && global.launchMap == "") {

global.customMapdesginated = 1;

var fileHandle, i, mapname;

fileHandle = file\_text\_open\_read(customMapRotationFile);

for(i = 1; !file\_text\_eof(fileHandle); i += 1) {

mapname = file\_text\_read\_string(fileHandle);

// remove leading whitespace from the string

while(string\_char\_at(mapname, 0) == " " || string\_char\_at(mapname, 0) == chr(9)) { // while it starts with a space or tab

mapname = string\_delete(mapname, 0, 1); // delete that space or tab

}

if(mapname != "" && string\_char\_at(mapname, 0) != "#") { // if it's not blank and it's not a comment (starting with #)

ds\_list\_add(global.map\_rotation, mapname);

}

file\_text\_readln(fileHandle);

}

file\_text\_close(fileHandle);

}

else if (global.launchMap != "") && (global.dedicatedMode == 1)

{

ds\_list\_add(global.map\_rotation, global.launchMap);

}

else { // else load from the ini file Maps section

//Set up the map rotation stuff

var i, sort\_list;

sort\_list = ds\_list\_create();

for(i=1; i <= 16; i += 1) {

if(maps[i] != 0) ds\_list\_add(sort\_list, ((100\*maps[i])+i));

}

ds\_list\_sort(sort\_list, 1);

// translate the numbers back into the names they represent

for(i=0; i < ds\_list\_size(sort\_list); i += 1) {

switch(ds\_list\_find\_value(sort\_list, i) mod 100) {

case 1:

ds\_list\_add(global.map\_rotation, "ctf\_truefort");

break;

case 2:

ds\_list\_add(global.map\_rotation, "ctf\_2dfort");

break;

case 3:

ds\_list\_add(global.map\_rotation, "ctf\_conflict");

break;

case 4:

ds\_list\_add(global.map\_rotation, "ctf\_classicwell");

break;

case 5:

ds\_list\_add(global.map\_rotation, "ctf\_waterway");

break;

case 6:

ds\_list\_add(global.map\_rotation, "ctf\_orange");

break;

case 7:

ds\_list\_add(global.map\_rotation, "cp\_dirtbowl");

break;

case 8:

ds\_list\_add(global.map\_rotation, "cp\_egypt");

break;

case 9:

ds\_list\_add(global.map\_rotation, "arena\_montane");

break;

case 10:

ds\_list\_add(global.map\_rotation, "arena\_lumberyard");

break;

case 11:

ds\_list\_add(global.map\_rotation, "gen\_destroy");

break;

case 12:

ds\_list\_add(global.map\_rotation, "koth\_valley");

break;

case 13:

ds\_list\_add(global.map\_rotation, "koth\_corinth");

break;

case 14:

ds\_list\_add(global.map\_rotation, "koth\_harvest");

break;

case 15:

ds\_list\_add(global.map\_rotation, "dkoth\_atalia");

break;

case 16:

ds\_list\_add(global.map\_rotation, "dkoth\_sixties");

break;

}

}

ds\_list\_destroy(sort\_list);

}

window\_set\_fullscreen(global.fullscreen);

draw\_set\_font(fnt\_gg2);

cursor\_sprite = CrosshairS;

if(!directory\_exists(working\_directory + "\Maps")) directory\_create(working\_directory + "\Maps");

instance\_create(0, 0, AudioControl);

instance\_create(0, 0, SSControl);

// custom dialog box graphics

message\_background(popupBackgroundB);

message\_button(popupButtonS);

message\_text\_font("Century",9,c\_white,1);

message\_button\_font("Century",9,c\_white,1);

message\_input\_font("Century",9,c\_white,0);

//Key Mapping

ini\_open("controls.gg2");

global.jump = ini\_read\_real("Controls", "jump", ord("W"));

global.down = ini\_read\_real("Controls", "down", ord("S"));

global.left = ini\_read\_real("Controls", "left", ord("A"));

global.right = ini\_read\_real("Controls", "right", ord("D"));

global.attack = ini\_read\_real("Controls", "attack", MOUSE\_LEFT);

global.special = ini\_read\_real("Controls", "special", MOUSE\_RIGHT);

global.taunt = ini\_read\_real("Controls", "taunt", ord("F"));

global.chat1 = ini\_read\_real("Controls", "chat1", ord("Z"));

global.chat2 = ini\_read\_real("Controls", "chat2", ord("X"));

global.chat3 = ini\_read\_real("Controls", "chat3", ord("C"));

global.medic = ini\_read\_real("Controls", "medic", ord("E"));

global.drop = ini\_read\_real("Controls", "drop", ord("B"));

global.changeTeam = ini\_read\_real("Controls", "changeTeam", ord("N"));

global.changeClass = ini\_read\_real("Controls", "changeClass", ord("M"));

global.showScores = ini\_read\_real("Controls", "showScores", vk\_shift);

ini\_close();

calculateMonthAndDay();

if(global.dedicatedMode == 1) {

AudioControlToggleMute();

room\_goto\_fix(Menu);

}

//Added Code

global.move =0;

global.RL=ds\_list\_create();

global.sqlvalue[0]=0;

global.mytime=0;

global.mcmd=ds\_map\_create();

//End Added Code

}