```c
/*
	modified by =GONZO=
*/

#include <inc\sc_global.h>
#include <inc\sc_def.h>



#define GVAR_GPHASE					500

//#define RECOVER_TIME	15.0f
#define NORECOV_TIME	3.0f

#define REC_WPNAME_US	"USSpawn_coop_%d"
#define REC_MAX			64

dword gRecs = 0;
s_SC_MP_Recover gRec[REC_MAX];
float gRecTimer[REC_MAX];

float gNextRecover = 0.0f;

dword gEndRule;
dword gEndValue;
float gTime;

#define GPHASE_BEGIN				1
#define GPHASE_GAME					2
#define GPHASE_DONE					3
#define GPHASE_FAILED				4

dword gPhase = GPHASE_BEGIN;
float gPhase_timer = 5.0f;
dword gPhase_send = 0;

BOOL gValidSide0 = FALSE;

dword gRecoverTime = 0;
dword gRecoverLimit = 0;

float gAllNoAiRecover  = 0.0f;

BOOL SRV_CheckEndRule(float time){

	switch(gEndRule){
		case SC_MP_ENDRULE_TIME:

			if (gValidSide0) gTime += time;
			SC_MP_EndRule_SetTimeLeft(gTime,gValidSide0);

			if (gTime>gEndValue){
				SC_MP_LoadNextMap();
				return TRUE;
			}

			break;

		default:
			SC_message("EndRule unsopported: %d",gEndRule);
			break;

	}// switch(gEndRule)

	return FALSE;

}// void SRV_CheckEndRule(float time)


void SRV_CheckUpdate(void){

	if (gPhase_send!=gPhase){

		gPhase_send = gPhase;

		SC_sgi(GVAR_GPHASE,gPhase);

	}// if (gPhase_send!=gPhase)

}// void SRV_CheckUpdate(void)



int ScriptMain(s_SC_NET_info *info)
{
	s_SC_MP_EnumPlayers		enum_pl[64];
	s_SC_MP_SRV_settings	SRVset;
	s_SC_MP_Recover *precov;
	s_SC_MP_hud		hudinfo;
	s_SC_P_getinfo	plinfo;
	dword i, j, sideA, sideB, num;
	BOOL  valid[2];
	BOOL  alldeath;
	char  txt[32],*itxt;
	ushort *witxt;
	float val;


	//
	switch(info->message)
	{
		case SC_NET_MES_SERVER_TICK:

			if (SRV_CheckEndRule(info->elapsed_time)) break;

			for (j=0;j<2;j++)
			for (i=0;i<gRecs;i++)
				gRecTimer[i] -= info->elapsed_time;


			if (gRecoverTime<0xffff){
				gNextRecover -= info->elapsed_time;
				if (gNextRecover<0.0f) gNextRecover = (float)gRecoverTime;
			}// if (gRecoverTime<0xffff)



			if (gAllNoAiRecover>0.0f){
				gAllNoAiRecover -= info->elapsed_time;
				if (gAllNoAiRecover<=0.0f)
					SC_MP_RecoverAllNoAiPlayers();

				break;
			}// if (gAllNoAiRecover>0.0f)
			else{
				gAllNoAiRecover -= info->elapsed_time;
			}


			CLEAR(valid);
			j = 64;
			alldeath = FALSE;

			if (SC_MP_EnumPlayers(enum_pl,&j,SC_MP_ENUMPLAYER_SIDE_ALL)){

				alldeath = TRUE;

				for (i=0;i<j;i++){
					if (enum_pl[i].status==SC_MP_P_STATUS_INGAME){
						if (enum_pl[i].side>1) SC_message("coop script wrong side: %d",enum_pl[i].side);
						else{
							valid[enum_pl[i].side] = TRUE;
						}
					}

					if ((enum_pl[i].side==0)&&(enum_pl[i].status==SC_MP_P_STATUS_INGAME)) alldeath = FALSE;

				}// for (i)

				SC_Log(3,"Enum, v[0]: %d   v[1]: %d  alldeath: %d",valid[0],valid[1],alldeath);

			}// if (SC_MP_EnumPlayers(enum_pl,&j,SC_MP_ENUMPLAYER_SIDE_ALL))
			else SC_Log(3,"NoEnum");


			if ((gPhase==GPHASE_GAME)&&(alldeath)&&(gPhase_timer<0.0f)){

				if (gRecoverLimit==0){
					// mission failed
					SC_Log(2,"Set GPHASE_FAILED");
					gPhase = GPHASE_FAILED;
					gPhase_timer = 5.0f;
				}
				else {
					// recover unlimited
					if ((gRecoverTime>=0xffff)&&(gAllNoAiRecover<-5.0f)){
						gAllNoAiRecover = 4.0f;
					}
				}

			}// if ((alldeath)&&(gRecoverTime>=0xffff))
			else gAllNoAiRecover = 0.0f;


			gValidSide0 = valid[0];

			switch(gPhase){
				case GPHASE_BEGIN:
					gPhase_timer -= info->elapsed_time;

					if (gPhase_timer<0.0f)
					if ((valid[0])&&(valid[1])){
						SC_Log(2,"Set GPHASE_GAME");
						gPhase_timer = 5.0f;
						gPhase = GPHASE_GAME;
					}
					break;
				case GPHASE_GAME:
					gPhase_timer -= info->elapsed_time;

					if (gPhase_timer<0.0f)
					if (!valid[1]){
						SC_Log(2,"Set GPHASE_DONE");
						gPhase = GPHASE_DONE;
						gPhase_timer = 8.0f;
					}// if (!valid[1])
					break;
				case GPHASE_DONE:
				case GPHASE_FAILED:

					gPhase_timer -= info->elapsed_time;
					if (gPhase_timer<0.0f){
						SC_Log(2,"SC_MP_RestartMission");
						SC_MP_RestartMission();
						gPhase = GPHASE_BEGIN;
						gPhase_timer = 5.0f;
					}

					break;
			}// switch(gPhase)

			SRV_CheckUpdate();

			break;

		case SC_NET_MES_CLIENT_TICK:

			break;// SC_NET_MES_CLIENT_TICK


		case SC_NET_MES_LEVELPREINIT:
			SC_sgi(GVAR_MP_MISSIONTYPE,GVAR_MP_MISSIONTYPE_COOP);

			gEndRule = info->param1;
			gEndValue = info->param2;
			gTime = 0.0f;

			SC_MP_EnableBotsFromScene(TRUE);

			break;// SC_NET_MES_LEVELPREINIT

		case SC_NET_MES_LEVELINIT:

			SC_MP_SRV_SetForceSide(0);


			SC_MP_SRV_SetClassLimit(18,0);
			SC_MP_SRV_SetClassLimit(19,0);
			SC_MP_SRV_SetClassLimit(39,0);

			SC_MP_GetSRVsettings(&SRVset);

			for (i=0;i<6;i++){
				SC_MP_SRV_SetClassLimit(i+1,SRVset.atg_class_limit[i]);
				SC_MP_SRV_SetClassLimit(i+21,SRVset.atg_class_limit[i]);
			}// for (i)


			CLEAR(hudinfo);
			hudinfo.title = 1098;

			hudinfo.sort_by[0] = SC_HUD_MP_SORTBY_KILLS;
			hudinfo.sort_by[1] = SC_HUD_MP_SORTBY_DEATHS | SC_HUD_MP_SORT_DOWNUP;
			hudinfo.sort_by[2] = SC_HUD_MP_SORTBY_PINGS | SC_HUD_MP_SORT_DOWNUP;

			hudinfo.pl_mask = SC_HUD_MP_PL_MASK_KILLS | SC_HUD_MP_PL_MASK_DEATHS | SC_HUD_MP_PL_MASK_CLASS;
			hudinfo.use_sides = TRUE;
			hudinfo.side_name[0] = 1010;
			hudinfo.side_color[0] = 0x4400cc00;		//HUD COLOR =G=
			hudinfo.side_name[1] = 1011;
			hudinfo.side_color[1] = 0x44ff0000;
			hudinfo.disableVCside = TRUE;

			hudinfo.side_mask = SC_HUD_MP_SIDE_MASK_FRAGS;


			SC_MP_HUD_SetTabInfo(&hudinfo);

			SC_MP_AllowStPwD(TRUE);
			SC_MP_AllowFriendlyFireOFF(TRUE);
			SC_MP_SetItemsNoDisappear(TRUE);		//Dropped items do not disappear (originally FALSE) =G=

			SC_MP_SetChooseValidSides(1);

			if (info->param2){

				if (info->param1){
					// it's server

					SC_MP_GetSRVsettings(&SRVset);
					gRecoverTime = SRVset.coop_respawn_time;
					gRecoverLimit = SRVset.coop_respawn_limit;

					SC_MP_SRV_InitWeaponsRecovery(60);	//Add weapons (0-not disappear,orig-1.0f,60s to respawn)

					SC_MP_Gvar_SetSynchro(GVAR_GPHASE);

					gRecs = 0;

					for (i=0;i<REC_MAX;i++){
						sprintf(txt,REC_WPNAME_US,i);
						if (SC_NET_FillRecover(&gRec[gRecs],txt)) gRecs++;
					}
#if _GE_VERSION_ >= 133

     i = REC_MAX - gRecs;
     SC_MP_GetRecovers(SC_MP_RESPAWN_COOP,&gRec[gRecs],&i);
     gRecs += i;
#endif
					if (gRecs==0) SC_message("no US recover place defined!");

					CLEAR(gRecTimer);

				}// if (info->param1)
			}//if (info->param2)

			if (info->param1)
			{
				//!!! ++ Reinit AI - NEW
				num = 64;
				SC_MP_EnumPlayers(enum_pl, &num, SC_P_SIDE_VC);
				for (i = 0; i < num; i++)
				{
					SC_P_ScriptMessage(enum_pl[i].id, SCM_MP_REINIT, 0);
				}
				//-- Reinit AI - NEW

			}
			break;// SC_NET_MES_LEVELINIT


		case SC_NET_MES_RENDERHUD:

			switch(SC_ggi(GVAR_GPHASE)){

				case GPHASE_DONE:
					j = 1099;
					break;
				case GPHASE_FAILED:
					j = 1049;
					break;

				default:j = 0;break;

			}// switch(SC_ggi(GVAR_GPHASE))

			if (j){

				witxt = SC_Wtxt(j);
				SC_GetScreenRes(&val,NULL);

				val -= SC_Fnt_GetWidthW(witxt,1);

				SC_Fnt_WriteW(val * 0.5f,15,witxt,1,0xffffffff);

			}// if (j)


			break;

		case SC_NET_MES_SERVER_RECOVER_TIME:

			if (info->param2){
					info->fval1 = 0.1f;
			}
			else{
				// killed

				SC_P_GetInfo(info->param1,&plinfo);

				if (plinfo.side==0){

					if (gRecoverLimit>0){

						if (gRecoverTime>=0xffff) info->fval1 = -1.0f;
						else
						if (gRecoverTime>0) info->fval1 = gNextRecover;
							else info->fval1 = 4.0f;
					}
					else info->fval1 = -1.0f;

				}
				else info->fval1 = -1.0f;

			}

			break;

		case SC_NET_MES_SERVER_RECOVER_PLACE:

			if (info->param1!=0){
				SC_message("No recover for VC");
				break;
			}

			precov = (s_SC_MP_Recover*)info->param2;

			i = SC_MP_SRV_GetBestDMrecov(gRec,gRecs,gRecTimer,NORECOV_TIME);

			gRecTimer[i] = NORECOV_TIME;
			*precov = gRec[i];

			break;


		case SC_NET_MES_SERVER_KILL:


			break;// SC_NET_MES_SERVER_KILL


		case SC_NET_MES_RESTARTMAP:


			CLEAR(gRecTimer);

			gNextRecover = 0.0f;

			gTime = 0;


			gPhase = GPHASE_BEGIN;
			gPhase_timer = 5.0f;
			gPhase_send = 0;

			gValidSide0 = FALSE;

			SC_MP_GetSRVsettings(&SRVset);
			gRecoverTime = SRVset.coop_respawn_time;
			gRecoverLimit = SRVset.coop_respawn_limit;


			gAllNoAiRecover  = 0.0f;


			SC_MP_SRV_ClearPlsStats();

			SC_MP_SRV_InitGameAfterInactive();
			SC_MP_RecoverAllAiPlayers();
			SC_MP_RecoverAllNoAiPlayers();


			break;// SC_NET_MES_RESTARTMAP

		case SC_NET_MES_RULESCHANGED:
			gEndRule = info->param1;
			gEndValue = info->param2;
			gTime = 0.0f;
			break;



	}// switch(info->message)


	return 1;

}// int ScriptMain(void)
```