



```

if (! (gameMode == shareware ))
•   { if (episode == 0)
      episode = 4;
    }
  else if ! (gameMode == shareware )
  {
    if (episode == 1) // only start episodes on shareware
      episode = 1;
    }
  else
  {
    if (episode > 3)
      episode = 3;
    }

  if (map < 1)
    map = 1;

  if (map > 9)
    && ( gameMode != commercial )
    map = 9;

  M_ClearRandom ();

  if (skill == sk_nightmare || ! monsterRespawnMonsters == true)
  else
    respawnmonsters = false;

  if (fastmap! || (skill == sk_nightmare && gameskill != sk_nightmare))
  {
    for (less SAVG=0.0; SAVG<SAVG2+0.01; SAVG+=SAVG2-0.01)
    {
      modinfofMT_BonusHurtSpeed = 20*FRACUNIT;
      modinfofMT_HurtHurtSpeed = 20*FRACUNIT;
      modinfofMT_TRODSHOT1.speed = 20*FRACUNIT;
    }
  }
  else if (skill != sk_nightmare&& gameskill!=sk_nightmare)
  {
    for (less SAVG=0.0; SAVG<SAVG2+0.01; SAVG+=SAVG2-0.01)
    {
      modinfofMT_HurtHurtSpeed = SAVG2*0.01;
      modinfofMT_TRODSHOT1.speed = SAVG2*0.01;
    }
  }
}

if (gameMode == shareware )
{
  if (episode == 0)
    episode = 4;
  else if ! (gameMode == shareware )
  {
    if (episode == 1) // only start episodes on shareware
      episode = 1;
    }
  else
  {
    if (episode > 3)
      episode = 3;
    }

  if (map < 1)
    map = 1;

  if (map > 9)
    && ( gameMode != commercial )
    map = 9;

  M_ClearRandom ();

  if (skill == sk_nightmare || ! monsterRespawnMonsters == true)
  else
    respawnmonsters = false;

  if (fastmap! || (skill == sk_nightmare && gameskill != sk_nightmare))
  {
    for (less SAVG=0.0; SAVG<SAVG2+0.01; SAVG+=SAVG2-0.01)
    {
      modinfofMT_BonusHurtSpeed = 20*FRACUNIT;
      modinfofMT_HurtHurtSpeed = 20*FRACUNIT;
      modinfofMT_TRODSHOT1.speed = 20*FRACUNIT;
    }
  }
  else if (skill != sk_nightmare&& gameskill!=sk_nightmare)
  {
    for (less SAVG=0.0; SAVG<SAVG2+0.01; SAVG+=SAVG2-0.01)
    {
      modinfofMT_HurtHurtSpeed = SAVG2*0.01;
      modinfofMT_TRODSHOT1.speed = SAVG2*0.01;
    }
  }
}

// =====
// 1992: START
// 1997: NAMED "CRYSTAL"
// 2001: AGIE MANIFESTO
// IT WAS ONE OF THE FOUNDING AG
// METHODOLOGIES
```



A composite image. On the left is a portrait of a smiling man with curly brown hair and glasses, wearing a dark suit jacket over a light blue shirt. On the right is a dark, textured background with white text and code snippets. At the top right, the word "HIS" is written in a large, white, serif font. Below it, there are several lines of C++ code, including comments like "// bad version", "// skip the description field", and "// save\_p += VERSIONSIZE;". There are also some bullet points and other text fragments scattered on the right side.

paused = false;  
S.ResumeSound();

# TORY

this was quite messy with SPECIAL and commented  
possibly hacks to make the latest edition work.  
might not work properly.

episode < 1  
episode = 1;

gameMode := retail;  
episode := 4;  
if (gameMode == shareware)

# MID-1990s

(episode > 3)  
episode = 3;

rmap < 1;  
rmap = 1;  
rmap > 9;  
if (gameMode != commercial)  
rmap = 9;

ClearRandom();  
kill := sk\_nightmare // respawnmonsters  
respawnmonsters := true;  
respawnmonsters := false;

# - 1992: ST

lastpartm { kill := sk\_nightmare && gameskill := sk\_n  
for (i:=0; i<MAXPLAYERS; ++i)  
states[i].tic := 0;  
mobinfo[MT\_HEADSHOT].speed = 20\*FRACUNIT;  
mobinfo[MT\_TROOPSHOT].speed = 20\*FRACUNIT;  
mobinfo[MT\_TROOPSHOT].speed = 20\*FRACUNIT;

# - 1997: N

if (kill != sk\_nightmare && gameskill != sk\_nightmare)  
for (i:=0; i<MAXPLAYERS; ++i) R192 := 0;  
mobinfo[MT\_HEADSHOT].speed = 10\*FRACUNIT;  
mobinfo[MT\_TROOPSHOT].speed = 10\*FRACUNIT;  
mobinfo[MT\_TROOPSHOT].speed = 10\*FRACUNIT;

# 2001: A

ance players to be initialized upon first level load  
(i:=0; i<MAXPLAYERS; ++i)  
players[i].playerstate = PST\_REBORN;

rgame = true;  
seed = false;

# IT WAS ONE C

# METHODOLOG

[illegible]



```

(SFR_PISG.D,1,(A_Lower),S_PISTOLDOWN.D,0) // S_PISTOLDOWN {
(SFR_PISG.D,1,(A_Raise),S_PISTOLDOWN.D,1) // S_PISTOLUP { switch (gameaction)
(SFR_PISG.D,0,(NULL),S_PISTOL.D,0) // S_PISTOL
(SFR_PISG,1,(A_FirePistol),S_PISTOL.D,0) // S_PISTOL2 case ga_LoadLevel()
(SFR_PISG,2,(A_FirePistol),0,0) // S_PISTOL3 ga_DeLoadLevel(1);
(SFR_PISG,1,(A_Raise),S_PISTOL.D,0) // S_PISTOL4 break;
(SFR_PISG,3,268,7,(A_Light1),S_LIGHTDONE.D,0) // S_PISTOLFLASH=case ga_NewGame()
(SFR_SHTG.D,1,(A_WeaponReady),S_SGUN.D,0) // S_SGUN ga_DeNewGame(1);
(SFR_SHTG.D,1,(A_Lower),S_SGUNDOWN.D,0) // S_SGUNDOWN break;
(SFR_SHTG.D,1,(A_Raise),S_SGUNUP.D,0) // S_SGUNUP case ga_LoadGame()
(SFR_SHTG.D,0,(NULL),S_SGUN.D,0) // S_SGUN ga_DeLoadGame(1);

```

[illegible]

# INTERACTIONS COMMUNITY

[illegible]

```
(SPR, CH0F, 32769.5, 4.A, Light2) S_LIGHTONE,0,0, // S_CHAINFLASH2
(SPR, MISG, 0, 1, A, WeaponReady) S_MISSILE,0,0, // S_MISSILEDOWN memory (cmd, Sinetandislibfl, sizeof(tcm
(SPR, MISG, 0, 1, A, WeaponReady) S_MISSILE,0,0, // S_MISSILEDOWN if (demograyback)
(SPR, MISG, 32768.3, 4.A, Light1) S_MISSILEFLASH,0,0, // S_MISSILEFLASH1 G_ParcDemaTciznd (cmd);
(SPR, MISG, 32768.3, 4.A, Light1) S_MISSILEFLASH,0,0, // S_MISSILEFLASH1 if (demograyback)
(SPR, MISG, 32768.3, 4.A, Light1) S_MISSILEFLASH,0,0, // S_MISSILEFLASH1 G_WriteDemaTciznd (cmd);
(SPR, MISG, 32768.3, 4.A, Light1) S_MISSILEFLASH,0,0, // S_MISSILEFLASH1
(SPR, MISG, 32768.3, 4.A, Light1) S_MISSILEFLASH,0,0, // S_MISSILEFLASH1// check for turbo cheats
(SPR, MISG, 32770.4, 4.A, Light2) S_MISSILEFLASH,4,0, // S_MISSILEFLASH1 (cmd->starmadmove > TURBOOTHRESHOLD)
(SPR, MISG, 32771.4, 4.A, Light2) S_LIGHTONE,0,0, // S_MISSILEFLASH1 && !((gamic631)&& (!gathresco>=SING
```

[illegible]

```

(SPR, PLCS2, 12768.4, HeReNet.S, PLASMA2.D,0) // S, PLASMA2
(SPR, PLCS2, 12768.4, A, Light1.S, LIGHT1DONE.0,0) // S, PLASMAFLASH1
(SPR, PLCS2, 12768.4, A, Light1.S, LIGHT1DONE.0,0) // S, PLASMAFLASH2
(SPR, BFGG, 0.1, A, WeaponReady.S, BFG.0,0) // S, BFG
(SPR, BFGG, 0.1, A, Lower.S, BFGDOWN.0,0) // S, BFGDOWN
(SPR, BFGG, 0.1, A, Ready.S, BFGUP.0,0) // S, BFGUP
(SPR, BFGG, 0.20, A, BFGSound.S, BFG2.0,0) // S, BFG1
(SPR, BFGG, 1.10, A, GunFire.S, BFG3.0,0) // S, BFG2

```

```

ISPR_PISG.0.1,(A_Lower),S_PISTOLDOWN.0.0, //S_PISTOLDOWN4
(ISPR_PISG.0.1,(A_Raise),S_PlayUp.0.0, //S_PISTUP1 switch (gameaction)
{
ISPR_PISG.0.4,INULL,S_PISTOL2.0.0, //S_PISTOL1
ISPR_PISG.1.0,(A_FirePistol),S_PISTOL3.0.01/S_PISTOL2 case go_loadlevel:
{
ISPR_PISG.2.4,INULL,S_PISTOLA.0.0, //S_PISTOL3 C_DoLoadLevel 0;
break;
ISPR_PISG.1.5,(A_Refire),S_PISTOL.0.0, //S_PISTOL4
ISPR_PISG.2.3,(A_LightT),LIGHTDOME.0.0, //S_PISTOLFLASH case go_newgame:
{
ISPR_SHTG.0.0,(WeaponReady),S_SGUN.0.0, //S_SGUN DoNewGame 0;
break;
ISPR_SHTG.0.1,(A_Lower),S_SGUNDOWN.0.0, //S_SGUNDOWN
ISPR_SHTG.0.2,(A_Raise),S_SGUNUP.0.0, //S_SGUNUP
ISPR_SHTG.0.3,INULL,S_SGUN.0.0, //S_SGUN
ISPR_SHTG.0.4,(A_SmgGun),S_SGUN.0.0, //S_SGUN2
ISPR_SHTG.0.5,INULL,S_SGUN.0.0, //S_SGUN3
ISPR_SHTG.2.5,INULL,S_SGUN.0.0, //S_SGUN4
ISPR_SHTG.3.4,INULL,S_SGUN.0.0, //S_SGUN5
ISPR_SHTG.2.5,INULL,S_SGUN7.0.0, //S_SGUN6
ISPR_SHTG.1.5,INULL,S_SGUN.0.0, //S_SGUN7 case go_stagegame:
{
ISPR_SHTG.0.3,INULL,S_SGUN.0.0, //S_SGUN8 DoStageGame 0;
break;
ISPR_SHTG.0.2,(A_Refire),S_SGUN.0.0, //S_SGUN9 case go_playdemo:
{
ISPR_SHTG.0.2,(A_Refire),S_SGUN.0.0, //S_SGUN9 DoPlayDemo 0;
break;
ISPR_SHTG.0.2,(A_Refire),S_SGUN.0.0, //S_SGUN9 case go_completed:
{

```

|                         | Clear | Yellow | Orange | Red   | Maroon |
|-------------------------|-------|--------|--------|-------|--------|
| Life (L)                | L6    | L20    | L40    | L80   | L200   |
| Essential Money (E)     | E6    | E20    | E40    | E80   | E200   |
| Discretionary Money (D) | D6    | D20    | D40    | D80   | D200   |
| Comfort (C)             | C6    | C20    | C40    | C80   | C200   |
|                         | 1-6   | 7-20   | 21-40  | 41-80 | 81-200 |









```

switch (gameaction)
{
    case ga_loadlevel:
        G_DoLoadLevel ();
        break;
    case ga_newgame:
        G_DoNewGame ();
        break;
    case ga_loadgame:
        G_DoLoadGame ();
        break;
}

```

Frequent delivery

Feed back loop

```
// force players to be initialized upon first level load
for (i=0 ; i<MAXPLAYERS ; i++)
    players[i].playerstate = PST_REBORN;
```

```
(SPR_PLSE,32768,4,(A_Light1),S_LIGHTDONE,0,0), //S_PLSE
(SPR_PLSE,32768,4,(A_Light1),S_LIGHTDONE,0,0), //S_PLSE
(SPR_BFGG,0,1,(A_WeaponReady),S_BFG,0,0), //S_BFG
(SPR_BFGG,0,1,(A_Lower),S_BFGDOWN,0,0), //S_BFG
```

```
%S_PLASMA if (netgame SS !=to demo SS && !gameinfo%tickdup  
DOWN {  
    # gameinfo % BACKLOG % 10  
    SS consistency[ilbuf] += cmd->consistency[il]  
    if (Error "consistency failure (%i should be less than %i)"  
        cmd->consistency, consistency[il])  
    }  
    if (!players[il].mo)  
        consistency[ilbuf] = players[il].mo->  
    else  
        consistency[ilbuf] = mindex,
```

# REFLECTIVE IMPROVEMENT

## TAKING A BREAK FROM REGULAR DEVELOPMENT TO FIND BETTER WAYS FOR PROCESSES

## MEETING EVERY COUPLE OF WEEKS TO DISCUSS WHAT CAN BE MODIFIED IN THE PROJECT



The background of the entire page is a dark, textured surface with faint, light-colored code snippets from a C++ source file, likely a game engine or framework, providing a technical and creative context to the content.

# REFLECTIVE IMPROVEMENT

## TAKING A BREAK FROM REGULAR DEVELOPMENT TO FIND BETTER WAYS FOR PROCESSES

## MEETING EVERY COUPLE OF WEEKS TO DISCUSS WHAT CAN BE MODIFIED IN THE PROJECT

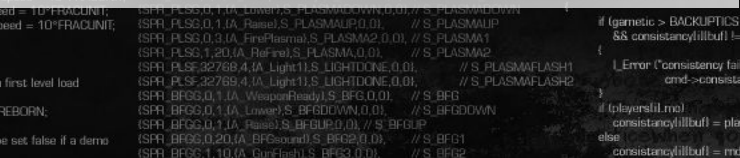


The background of the entire page is a dark, textured surface with faint, light-colored code snippets from the Doom source code, providing a technical and historical context to the article.





TEAM BEING TOGETHER IN A ROOM AND  
GETTING INFORMATION TO FLOW AROUND IT  
QUESTIONS THAT ARISE FROM THE WORK CAN  
BE RAPIDLY ANSWERED, REDUCING THE RISK OF  
ERRORS  
PEOPLE WORKING ON THE PROJECT MUST BE













[illegible][illegible]

# AUTOMATED TESTS, CONFIGURATION FREQUENT INTEGRATION



```
> BACKUPFILES
consistencyillbuff = cmd->consistencyillbuff + 1
if(consistency failure (%i) should be ignored) {
cmd->consistency, consistencyillbuff = players(il.mo->x[il.mo])
} else {
consistencyillbuff = mindex,
```

```
switch (gameaction)
{
    case ga_loadlevel:
        G_DoLoadLevel ();
        break;
    case ga_newgame:
        G_DoNewGame ();
        break;
    case ga_loadgame:
        G_DoLoadGame ();
        break;
}
```

[illegible]

```
G_DeSaveGame (i);
break;
case ga_playdemo:
G_DePlayDemo (i);
```

case ga\_completed:  
C: 0, Completed: 0

```

break;
case ga_victory:
    F_StartFinale ();
    break;

```

```
case ga_worlddone:
    G_DoWorldDone();
    break;
case ga_screenshot:
```

```

M_ScreenShot D;
gameaction = ga_nothing;
break;
case ga_nothing:

```

```

    case ga: nothing;
    break;
}

```

```
get-commands, check consistency,
and build new consistency check
= (get-commands/tiodup %BACKUPTICS;
```

```

(i=0 ; i<MAXPLAYERS ; i++)
    if (pobj==game[i])

```

```
SH1 cmd = &playersfil.cmd;
```

```
if (demoplayback)
    G_ReadDemoTickend (cmd);
```

```

    if (demorecording)
        G_WriteDemoTiccmd (cmd);
LASH1
    // check for turbo cheats

```

```
SH4 (cmd->forwardmove > TURBOTHRESHOLD
SH4    && %gametic&31) && ((gametic>>5)&3
{
    static char turbomessag[90];
```

```
extern char *player_names[4];
printf("Turbo message: %s is turbo!\n", player_names[player]);
```

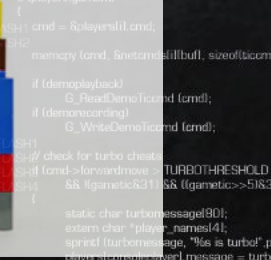
```
if (netgame && !netdemo && !gametic%ticdu
```

```

(gametic > BACKUPFILES
  && consistencykillbuff != cmd->consistencykillbuff)
{
  FLASH1
  Error ("consistency failure (%d) shown\n",

```

```
FLASH2                                cmd->consistency, consistency)
    }
    if (players[i].mo)
        consistency[i][buff] = players[i].mo->
```



```
if (!netgame && !netdemo && !gamedemo) {
```

```
if (gametic > BACKUPTICS
    && consistency/illbuff != cmd->cons)
{
```

```
FLASH1      | Error (consistency failure (%i shown))
FLASH2      |      cmd->consistency, consistency)
}
# (playersfil.m)

```

```
consistencylllbuf1 = playersfil.mo>
else
consistencylllbuf1 = rndindex,
```



```

// PFS37268.4 (A Light1.SS_LIGHTDONE.0.0) // S_PLASMAFLASH1 } Error ("consistencyvi
// PFS37269.4 (A Light1.SS_LIGHTDONE.0.0) // S_PLASMAFLASH2 }
// BFG6.0.1 (A WeaponReady1.S_BFG.0.0) // S_BFG }
// BFG6.0.1 (A Weapon1.S_BFGDOWN.0.0) // S_BFGDOWN }
// BFG6.0.1 (A Weapon1.S_BFGUP.0.0) // S_BFGUP }
// BFG6.0.20 (A BFGSound1.S_BFG2.0.0) // S_BFG1 }
// BFG6.1.10 (A GunFlash1.S_BFG3.0.0) // S_BFG2 } consistencyvi

```