ioquake eye candy changes

- include:
  - smoke
  - blood
  - gibs
  - simple python bot basic movements

Smoke

- firstly note that in `qcommon/q_shared.h` the definition

```
#define GORE
```

- all gore related changes should be bound by:

```
#if defined(GORE)
/* modified code */
#else
/* original code */
#endif
```


changes to cgame/cg_particles.c

```
#if defined(GORE)
#   define PARTICLE_GRAVITY        10
#   define MAX_PARTICLES    1024 * 8 * 128
#else
#   define PARTICLE_GRAVITY        40
#   define MAX_PARTICLES    1024 * 8
#endif
```


changes to cgame/cg_weapons.c

- the function rocket trail is altered to yield
  - twice the density of smoke
  - to allow grenades to smoke even if stationary

```
#if defined(GORE)
        step = 25;
#else
        step = 50;
#endif
```

changes to cgame/cg_weapons.c

```
#if defined(GORE)
    // grenades smoke even if stationary :-)
#else
    // if object (e.g. grenade) is stationary, don't toss
    if ( es->pos.trType == TR_STATIONARY ) {
        ent->trailTime = cg.time;
        return;
    }
#endif
```


changes to cgame/cg_weapons.c

```
#if defined(GORE)
    if ( es->pos.trType == TR_STATIONARY ) {
       smoke = CG_SmokePuff( lastPos, up,
                             wi->trailRadius,
                             1, 1, 1, 0.33f,
                             wi->wiTrailTime,
                             t,
                             0,
                             0,
                             cgs.media.smokePuffShader );
       // use the optimized local entity add
       smoke->leType = LE_SCALE_FADE;
       return;
    }
#endif
```


cgame/cg_marks.c

- handles particles and their dissipation

```
#if defined(GORE)
#   define              PARTICLE_GRAVITY        10
#   define              MAX_PARTICLES   1024 * 8 * 128
#else
#   define              PARTICLE_GRAVITY        40
#   define              MAX_PARTICLES   1024
#endif
```


function CG_ParticleDust

```
void CG_ParticleDust (centity_t *cent, vec3_t origin, vec

etc

#if defined(GORE)
  if (length)
    p->endtime = cg.time + (4500*10) + (crandom() * 3500)
  else
    p->endtime = cg.time + (750*10) + (crandom() * 500);
#else
  if (length)
    p->endtime = cg.time + 4500 + (crandom() * 3500);
  else
    p->endtime = cg.time + 750 + (crandom() * 500);
#endif

etc
```

Adding per model gibs

- files altered
  - q3_ui/ui_local.h
  - q3_ui/ui_players.c

playerInfo_t extended in q3_ui/ui_local.h

```
#if 1
        qhandle_t               gibSkull;
        qhandle_t               gibBrain;
        qhandle_t               gibAbdomen;
        qhandle_t               gibArm;
        qhandle_t               gibFist;
        qhandle_t               gibFoot;
        qhandle_t               gibForearm;
        qhandle_t               gibChest;
        qhandle_t               gibIntestine;
        qhandle_t               gibLeg;
        int                     nSkulls;
        int                     nBrains;
        int                     nAbdomens;
        int                     nArms;
        int                     nFists;
        int                     nFeet;
        int                     nForearms;
        int                     nChests;
        int                     nIntestines;
        int                     nLegs;
#endif
```


playerInfo_t extended in q3_ui/ui_local.h

- notice that this is not switchable on/off at compile time by the GORE definition
- as it would probably mean it would be incompatible with other clients built without GORE
  - in general it is safer **not** to make data structures compile time switchable
    - any mistake could prove very very costly to debug

q3_ui/ui_players.c

```
/*
==========================
UI_RegisterClientModelname
==========================
*/
qboolean UI_RegisterClientModelname (playerInfo_t *pi,
                                     const char *modelSki
                                     const char *headMode
                                     const char *teamName
{
  char modelName[MAX_QPATH];
  char skinName[MAX_QPATH];
  char headModelName[MAX_QPATH];
  char headSkinName[MAX_QPATH];
  char filename[MAX_QPATH];
  char *slash;

  Com_Printf ("Inside UI_RegisterClientModelname");
  pi->torsoModel = 0;
  pi->headModel = 0;
```

q3_ui/ui_players.c

```
if (! modelSkinName[0])
   return qfalse;

Q_strncpyz (modelName, modelSkinName, sizeof(modelName)

slash = strchr (modelName, '/');
if (!slash) {
  // modelName did not include a skin name
  Q_strncpyz (skinName, "default", sizeof(skinName));
} else {
  Q_strncpyz (skinName, slash+1, sizeof(skinName));
  *slash = '\0';
}
```


q3_ui/ui_players.c

```
Q_strncpyz (headModelName, headModelSkinName, sizeof (h
slash = strchr (headModelName, '/');
if (!slash)
  // modelName did not include a skin name
  Q_strncpyz (headSkinName, "default", sizeof (skinName
else {
  Q_strncpyz (headSkinName, slash + 1, sizeof (skinName
      *slash = '\0';
}
```


q3_ui/ui_players.c

```
// load cmodels before models so filecache works

pi->legsModel = UI_doModelFile ("models/players/%s/lowe
                                "models/players/charact
                                modelName);
if (!pi->legsModel) {
  Com_Printf ("Failed to load model file %s\n", filenam
  return qfalse;
}

pi->torsoModel = UI_doModelFile ("models/players/%s/upp
                                 "models/players/charac
                                 modelName);
if (!pi->torsoModel) {
  Com_Printf ("Failed to load model file %s\n", filenam
  return qfalse;
}
```


q3_ui/ui_players.c

```
if (headModelName[0] == '*' )
  Com_sprintf( filename, sizeof (filename),
               "models/players/heads/%s/%s.md3",
               &headModelName[1], &headModelName[1] );
else
  Com_sprintf( filename, sizeof (filename),
               "models/players/%s/head.md3", headModelN

pi->headModel = trap_R_RegisterModel( filename );
if ( !pi->headModel && headModelName[0] != '*') {
  Com_sprintf( filename, sizeof( filename ),
     "models/players/heads/%s/%s.md3", headModelName, h
  pi->headModel = trap_R_RegisterModel( filename );
}

if (!pi->headModel) {
  Com_Printf( "Failed to load model file %s\n", filenam
  return qfalse;
}
```

q3_ui/ui_players.c

```
#if 1
    // Gaius
    UI_ParseBodyFile ("models/players/%s/%s.body", modelN
#endif
```


new function doModelFile

```
static qhandle_t UI_doModelFile (const char *preferredPat
                                 const char *defaultPath,
                                 const char *modelName)
{
  char filename[MAX_QPATH*2];
  qhandle_t q;

  Com_sprintf (filename, sizeof (filename), preferredPath
  q = trap_R_RegisterModel (filename);
  if (q == 0 && defaultPath != NULL) {
    Com_sprintf (filename, sizeof (filename), defaultPath
    q = trap_R_RegisterModel (filename);
  }
  return q;
}
```


new function doModelFile

```
/*
================
UI_ParseBodyFile
================
*/
void UI_ParseBodyFile (const char *fmt, const char *model
{
  char          *textPtr;
  int            len;
  char          *token;
  char           text[20000];
  char           filename[MAX_QPATH];
  fileHandle_t   f;

  UI_set_default_body_parts (pi);
  Com_sprintf (filename, sizeof (filename), fmt, modelNam
  Com_Printf ("About to read file: %s", filename);
```


new function doModelFile

```
  // load the file
  len = trap_FS_FOpenFile (filename, &f, FS_READ);
  if (len <= 0)
    return ;

  if (len >= (sizeof (text)-1)) {
    Com_Printf("File %s too long\n", filename);
    trap_FS_FCloseFile (f);
    return;
  }
  trap_FS_Read (text, len, f);
  text[len] = 0;
  trap_FS_FCloseFile(f);

  Com_Printf("About to parse file: %s", filename);

  // parse the text
  textPtr = &text[0];
```

new function doModelFile

```
  token = COM_Parse (&textPtr);
  if (UI_seen_token (token, "{")) {
    token = COM_Parse (&textPtr);
    while ((token != NULL) && (! UI_seen_token (token, "}
      if (UI_seen_value (&textPtr, &token, "BODY_NO_OF_SK
          UI_seen_value (&textPtr, &token, "BODY_NO_OF_BR
          UI_seen_value (&textPtr, &token, "BODY_NO_OF_AB
          UI_seen_value (&textPtr, &token, "BODY_NO_OF_AR
          UI_seen_value (&textPtr, &token, "BODY_NO_OF_FI
          UI_seen_value (&textPtr, &token, "BODY_NO_OF_FE
          UI_seen_value (&textPtr, &token, "BODY_NO_OF_FO
          UI_seen_value (&textPtr, &token, "BODY_NO_OF_CH
          UI_seen_value (&textPtr, &token, "BODY_NO_OF_IN
          UI_seen_value (&textPtr, &token, "BODY_NO_OF_LE
        ;
      else
        token = COM_Parse (&textPtr);
    }
  }
```


q3_ui/ui_players.c

```
static int UI_seen_token (char *token, const char *match)
{
  return (Q_stricmp (token, match) == 0);
}

static int UI_seen_value (char **textPtr, char **token,
                          const char *match, int *value)
{
  if (UI_seen_token (*token, match)) {
    *token = COM_Parse (textPtr);
    if (token != NULL)
      if (isdigit (**token)) {
        *value = atoi (*token);
        *token = COM_Parse (textPtr);
        return qtrue;
      }
  }
  return qfalse;
}
```


q3_ui/ui_players.c

```
/*
 *  UI_set_default_body_parts - sets the default for huma
 */

static void UI_set_default_body_parts (playerInfo_t *pi)
{
  pi->nSkulls = 1;
  pi->nBrains = 1;
  pi->nAbdomens = 1;
  pi->nArms = 2;
  pi->nFists = 2;
  pi->nFeet = 2;
  pi->nForearms = 2;
  pi->nChests = 1;
  pi->nIntestines = 1;
  pi->nLegs = 2;
}
```