Ya3dag

Other tips

What you always wanted to know about Ya3dag.

Based on V1.51 release of August 17, 2014

Edition history

14.11.2004 RR: First edition.
16.12.2004 RR: Picture of the level during loading.
11.01.2005 RR: TravelOverland Level description reworked.
24.08.2008 RR: Translation to English.
24.01.2012 RR: Updated based on V1.35 release of February 1, 2012.
19.08.2012 RR: Updated based on V1.40 release of August 19, 2012.
10.02.2013 RR: Updated based on V1.41 release of February 12, 2013.
30.03.2013 RR: Updated based on V1.42 release of June 30, 2013.
17.08.2014 RR: Updated based on V1.51 release of August 17, 2014.

Table of contents

1. Introduction

This document describes some of Ya3dag things for which any other documentation is not the right place.

One goal of Ya3dag is to add new content to the game easily.

To get access to game scripts and documentation files from the project, unzip the game data in **Ya3dag\BaseQ2\Q2T_BaseQ2.pkz** and **Ya3dag\RRGame\Q2T_RRGame.pkz**. Rename the **.pkz** file extension to **.zip** and unzip it into **Ya3dag\BaseQ2** respectively **Ya3dag\RRGame**.

2. Games

Ya3dag offers the possibility to add a game (or modification).

Here are the rules to add such a game:

- Create a subdirectory for your game. Let the name start with 'Mod' to make clear this is a game directory.
- Add the file GameConfiguration.txt (see later for description).
  NOTE: This file may not be inside a pack file (.pkz or .zip).
- Add the file GameImage.jpg. This is used as preview image in Ya3dags 'Games' dialog.
  NOTE: This file may not be inside a pack file (.pkz or .zip).
- Add the file default.cfg. This file specifies initial keyboard layout, display settings, intro file (alias d1) and initial map (alias newgame).
- The into file is placed in the pics\Intro directory and must have the file extension cfg. It is a plain text file and can be edited with notepad. Take a look at the existing files to learn about the intro scripting language.
- Add game related data.

Here are the rules to switch between installed games:

- A game has it's own subdirectory
- This directory must hold the files
  GameConfiguration.txt
  GameImage.jpg
- Ya3dags main menu entry 'Games' allows switching between the games. This entry is only offered if there are two or more games existing.
- There must be one game existing for Ya3dag to startup.
- If there is only one games/modifications at startup, this one will be selected for playing.

2.1 About GameConfiguration.txt

The file GameConfiguration.txt specifies this settings for a game:

- Section [GameDescription]
  Setup game name and description. This information is used for Ya3dags 'Games' dialog. Text formatting as described in the documentation Ya3dag_Scripting_Language.pdf, chapter ‘Text attributes for dialog texts‘ is usable here. Until now, there is no multi-language support for this text settings.

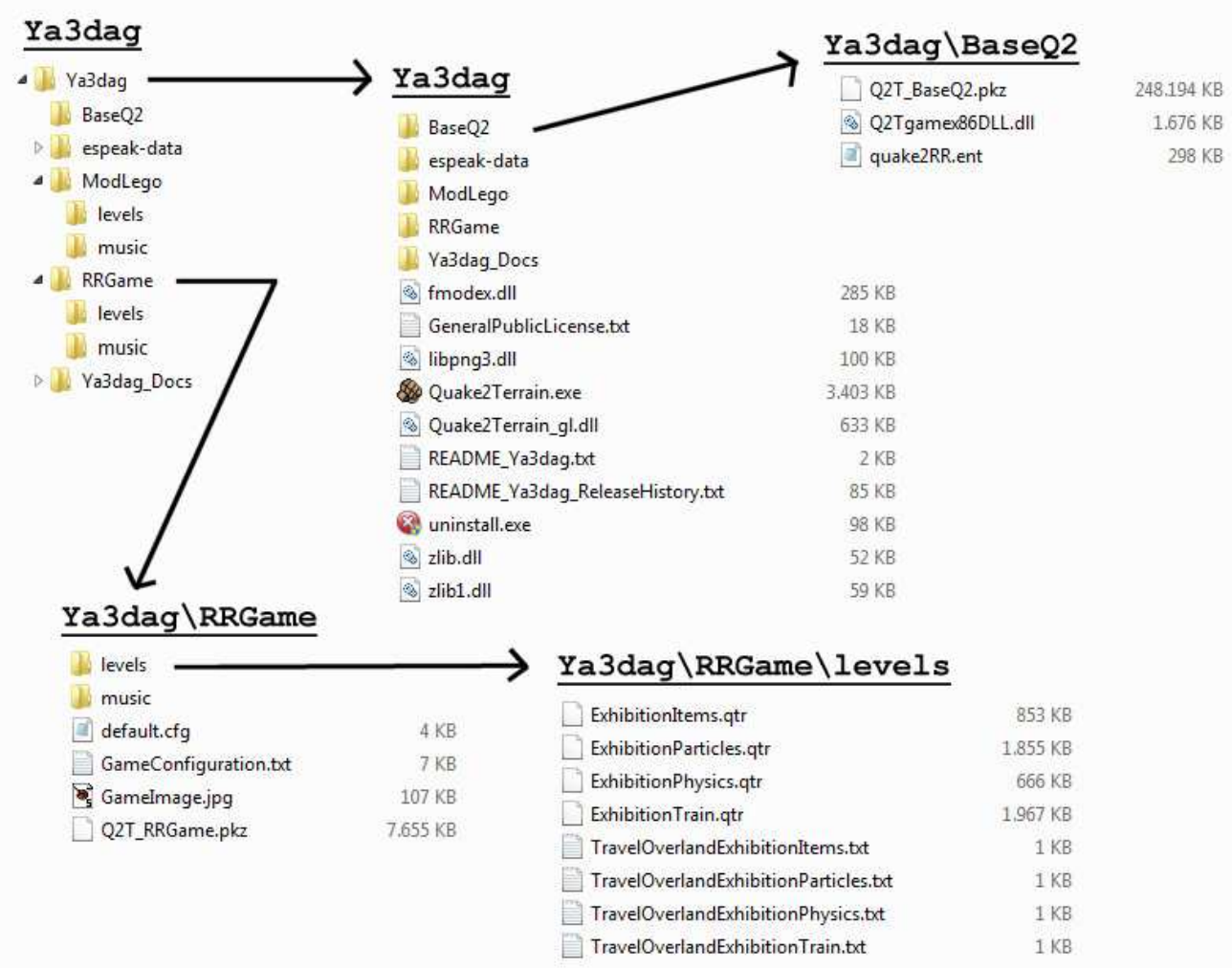
- Section [GameDLL]
  The setting **single_statusbar** defines the layout of the single player status bar (see ‚HUD language‘ elsewhere in this manual). Other settings define the position, frames and background images for the in-game dialogs or messages and the iBag menu.

- Section [MenuBanner]
  A background image for the header of each menu dialog is defined here. Also the size and color of a dialog header is specified here.

- Section [MainMenu]
  Things for the main menu. Name and position for up to three background images. Enable the network (or multiplayer) dialog. Position, size and background images for the main menu entries.

Take a look at some ‘GameConfiguration.txt‘ files to learn more.

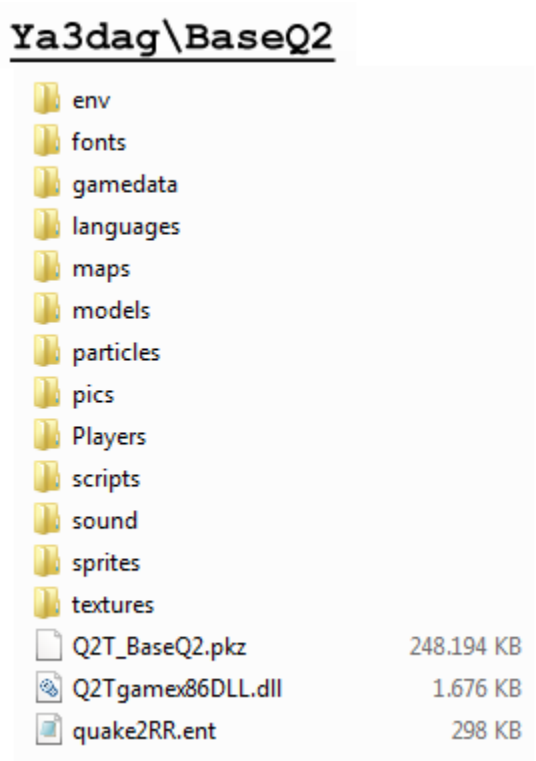
2.2 Directories and important files



Above you see some explorer screenshots of a Ya3dag installation.

And now a brief description of some directories and important files:

- Ya3dag is the root directory. All other directories/files are below this root.
- Ya3dag\Quake2Terrain.exe is the game engine. Start this to execute Ya3dag. This is part of the source download.
- Ya3dag\Quake2Terrain_gl.dll is a .dll file used to display graphic. This is part of the source download.
- Ya3dag*.dll are some other .dll files needed to run Quake2Terrain.exe.
- Ya3dag\BaseQ2 is the base data directory. The data below this directory is common to all games.
- Ya3dag\BaseQ2\Q2Tgamex86DLL.dll is a so called game-dll. There can be one in each game directory. The one in BaseQ2 is the fallback.
- Ya3dag\BaseQ2\quake2RR.ent is a setup file for Ya3dags editor. There can be one in each game directory. The one in BaseQ2 is the fallback.
- Ya3dag\ModLego is one of Ya3dags games. The data below this directory is specific to this game.
- Ya3dag\ModLego\levels holds the map for this game. These files can (for now) not be put into a pack file.
- Ya3dag\ModLego\music holds the music for this game. These files can (for now) not be put into a pack file.
- Ya3dag\Ya3dag_Docs holds documentation.
- Pack files are zip archives. These files have the file extension pkz or zip. Please note that there is a tool named 'Ya3dag_minizip.exe' too build such archives from scratch.



This is an explorer screenshot of the Ya3dag\Baseq2 directory after unpacking the pack file 'Q2T_BaseQ2.pkz'. It shows the typical directories of a Ya3dag game. If you are used to Q2 games, most of them is known to you.

To access game data, the following search order is used:
1) game directory, file exists in the directory structure.

2) game directory, file exists in a pack file of the game directory.
3) BaseQ2 directory, file exists in the directory structure.
4) BaseQ2 directory, file exists in a pack file of the BaseQ2 directory.

Use the knowledge about the search order if you want to replace contents.

Now a brief description of this subdirectories:

- env, holds environment images.
- fonts, hold font files.
- gamedata, holds actor scripts, book files, recipes, …. See elsewhere in this manual.
- languages, multi-language support. The files inside here contains translations of texts.
- maps, holds map models. This are .bsp models. Larger buildings are made out of several small .bsp models. You can find bridges, cargo boxes, graves, walls and such things here. The ‘Qolle99‘ can be used to build new models.
- models, holds .md2 or .md3 models. This models are used for items, weapons, plants, animals, monsters, level decoration (skeletons, dead animals, vases and such things here), …
- particles, holds images used for game effects.
- pics, holds images for some game data.
- Players, holds player models together with skins and sound files.
- scripts, holds shader files. These files are compatible to Q3 shader files and describe surface effects of textures.
- sound, holds sounds for weapons, items, animals, environment, …
- sprites, holds images for some game data.
- textures, holds images for .bsp models and for terrain. The ‘Q2T‘ subdirectory holds data files specific for terrain.

Some other commonly used directories:

- level, holds ‘.qtr‘ files. This are Ya3dags level files. Use Ya3dags editor to make such files.
- levelshots, holds a loading image for each level.
- music, level music. Ya3dag supports .mp3 .ogg and .mid music files.

3. Gamedata

Gamedata can be found in the subdirectory 'gamedata' of each game. Besides scripts (see documentation Ya3dag_Scripting_Language.pdf), there are books, roles and recipes.

3.1 About books

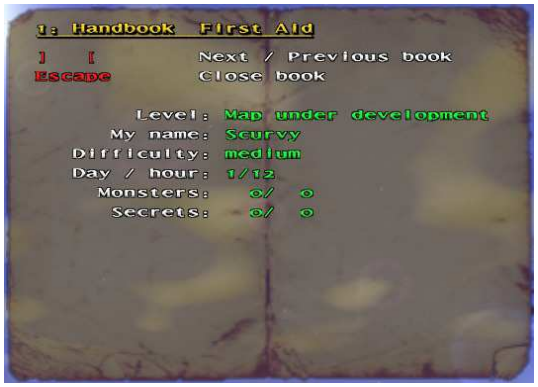
- Books will be picked up by the players or will be given to the player. Books will be stored in the inventory.

- Activate the book in the inventory or press the b key to read the books. Navigate through the books with the [,], Enter and Escape keys. Also the mouse wheel is usable here.

- Add books to the level with the editor action:
  Objects/Add object/Item/Other/Book

  Select Objects/Selected object
  With the property message you choose one of the book files from the sub-directory gamedata. Book file names begin with Book and have the file extension txt. The default for the property message is BookDefault.txt.

- Example for a book file:




```
; BookFirstHelp.txt
;
; Handbook for the adventurer
;
; Max. 17 lines of the following with.
;12345678901234567890123456789012345678
;
; The first line is used as header line.
;
;12345678901234567890123456789012345678
Handbook  First Aid
^1#   +^r        Next / Previous book
^1Escape^r       Close book

        Level: ^2$Level
      My name: ^2$Name
   Difficulty: ^2$Skill
   Day / hour: ^2$DayNr/$HourNr
     Monsters: ^2$Monsters
      Secrets: ^2$Secrets
[End Of File]
```

Lines starting with ; are comment lines. The first line is as the book title. Lines can. A line can have a maximum of 37 characters. A book has up to 17

lines. Note the colored text (see Ya3dag_Scripting_Language.PDF, chapter „Text attributes for dialog texts“).

Books have their own set of $-variables:

$HourNr	Days since begin of the game.
$DayNr	Hour of the day.
$Skill	Easy, medium or hard.
$Name	The name of the player.
$Level	Name of the level.
$Monsters	Number of killed/total monsters in the level.
$Secrets	Number of found/total secrets in the level.

3.2 About rolls

- Rolls are activated when they are touched by the player.
- Rolls show its text only once on the screen.
- Add books to the level with the editor action:
  Objects/Add object/Item/Other/Roll

  Select Objects/Selected object
  With the property message you choose one of the roll files from the sub-directory gamedata. Roll file names begin with Roll and have the file extension txt. The default for the property message is RollDefault.txt.
- Use rolls for hints, tips or explanations.

Example for a roll file:


```
; RollDefault.txt

;
; Max. 24 lines.
; The first line is used as header line.
;
;xxxxxxxxxxxxxxxxxxxxxxxxxx
^5^yA roll
This roll is not
important for the
upcoming tasks.
[End Of File]
```

Lines starting with ; are comment lines. The first line is as the roll title. Lines can. A line can have a maximum of 27 characters. A book has up to 24

lines. Note the colored text (see Ya3dag_Scripting_Language.PDF, chapter „Text attributes for dialog texts“).

3.3 About recipes

- Recipes are Yad3dags method to manufacture new items. Recipes will be collected as well as other items. First, all ingredients of the recipe must be in possession of the player. After that, the new item can be manufactured. The availability of ingredients is displayed each time.

- Recipes will be picked up by the players or will be given to the player.

- Activate the iBag menu or press the r key to get the recipes. Navigate through the books with the [,], Enter and Escape keys. Also the mouse wheel is usable here.

- Add books to the level with the editor action:
  Objects/Add object/Item/Other/Recipe

  Select Objects/Selected object
  With the property message you choose one of the recipe files from the sub-directory gamedata. Recipe file names begin with Recipe and have the file extension txt. The default for the property message is RecipeDefault.txt.

Example for a recipe file:


```
; Recipe_PoisonFlask.txt
;
; * Per line
;   First is the amount of items produced/needed.
;   Second is the classname of an item (see file Items.txt).
;
; * The first line is the item(s) that is produced with the recipe.
;
; * The next lines are the needed ingredients (an ingredient per line).
;   In addition to the class name of an object also "Money", "Mana"
;   or "Health" can be used.
;
; * Empty lines are skipped.
;
; * Comments begin with a ';' character.
;
; * A Line starting with '[' ends the parsing of this file.
;

   1 item_p_flask       ; Poison flask

   1 item_e_flask       ; Empty flask
   3 item_Toadstool     ; Toadstool
```

[End Of File]

It's simple.
Only name the ingredients and the amount needed. See the file Items.txt for the class names of items.
Lines starting with ; are comment lines.

3.4 About Items.txt

The file Items.txt is the database of all items in the game. The most important thing is the class name of the item. This is used as reference to an item in actor scripts, recipes or editor data. In addition to the model also the item type, graphic gimmicks, weight, price and description is give here.

The initial place of this file is in BaseQ2/gamedata. Each game can have it's own incarnation. Simply put a modified copy of Item.txt in the gamedata directory of the game.

To use the items in the editor, enter them in the file 'quake2RR.ent'.

Take a look at this file to learn more.

4. Replacing or Adding Content

Any contents can be replaced. See also the chapter ‚directories and important files‘ to learn about the search order of files.

Any contents may be added. Some new content is handled automatically, while other content somewhere else must be mentioned.

4.1 About images

Image formats supported: png, tga, jpg, pcx, wal, lim.
If there are images with the same base name but different file extensions, the precedence is in the order named above. So a tga file is used before a pcx file.

Lim is Ya3dags link-image data file format.

- Create a file with the extension '.lim' (Create a new notepad file with the help of the explorer).
- With notepad write in the name of an existing texture. If the filename of the texture is replaced with an asterik, the filename of the .lim file is used as image name.

```
 Example 2:
 * In directory 'textures\Q2T\tex\Wall' create the
   file 'misc_13_q1_ground1_7.lim'.
 * Write in the text
   textures/Quake/ground1_7.jpg

 Example 1:
 * In directory 'textures\Q2T\tex\Misc' create the
   file 'Wall_22c.lim'.
 * Write in the text
   textures/Wall/*
   to use the file textures/Wall/Wall_22c.jpg

 ==> In the Ya3dag terrain editor, the texture
     misc_13_q1_ground1_7 is listed, but used is
     the texture ground1_7.jpg from the quake
     directory. Note that also the material files
     _norm, _depth and _gloss are loaded too.

Benefits:
  * Terrain textures in the Q2T subdirectory can be linked
    to an other existing textures.
    Every texture can be used as terrain texture
    without wasting space on the hard disk.
  * Also material textures like _norm, _depth or _gloss
    can link to other files.
    Use this textures multiple times.
  * Link saves memory/texture space because textures
    can be multipled used but are loaded only once.
```

Texture layers.

Texture layers are handled automatically inside Ya3dag. The layer must just exist and Ya3dag adds them to the base texture.
Texture layers are handled for .bps model textures, terrain textures and .md2/.md3 model skins.
All texture layers are optional except the base texture (the others are NOT

loaded without it). The layers can have different file extensions. As example besides the base texture ‚metal1_4.jpg' in the BaseQ2\textures\Quake directory, there are the files ‚metal1_4_depth.jpg', ‚metal1_4_gloss.jpg' and ‚metal1_4_norm.jpg'.

Normal data has the name suffix "_norm" or "_normal". Can have alpha channel with height data for offsetmapping/reliefmapping.
This layer is usable for .bsp textures, terrain textures and terrain splatters.

Height data has the name suffix "_bump" or "_depth". This is not loaded if there is a normal data file with alpha channel. The height data is needed for offsetmapping/reliefmapping.
This layer is usable for .bsp textures, terrain textures and terrain splatters.

Gloss data has the name suffix "_gloss. This layer is used as specular color map (the color of reflected light).
This layer is usable for .bsp textures only (until now).

Glow data has the name suffix "_gloss", "_decal", "_glow", "_add" or "_blend". If the file has an alpha channel, the image layer is blended (by the alpha channel) else the image layer is added (is a glow effect).
This layer is usable for .bsp textures, terrain textures, terrain splatters and model skins.

_TextureData.cfg

This file is available in most textures directories. It specifies additional features of a texture like surface flags, contents, light value, animations and a scale factor. There is a column called ‚relief' to automatically generate height and normal data from a base image file.

Surface flags (like SURF_WARP) and contents (like CONTENTS_SOLID) should be known to Q2 mappers.
Qoole99 also looks at this data files.

5. Others

5.1 Notice of a level in the travel agency TravelOverland

Put a file ‚TravelOverlandXXXXXXXX.txt' in the subdirectory maps, in addition to the Level XXXXXXXX.qtr.

Following an example for the contents of this file:

```
;
; TravelOverlandWGFireland.txt
;
; Description of level for TravelOverland
;
;
; line 1: Name of level
; line 2: cost of a ticket
; line 3: minimum points player skills
; line 4: type of level: Single Multi
; line 5: short description of level
; line 6: Author
; line 7: spare, keep empty
; line 8: long description of level, lines until [EndOfFile]
;
;
;
WGFireland
600
Magic 30 nTreasures 5
Single
"&level_wgfireland_sname=Hadal"
R. Reinhard

;12345678901234567890123456789O
"&level_wgfireland_lname="
"A trip with claims. This is"
"something for the experienced"
"adventurer. Please check your"
"equipment before you begin"
"this journey."
[EndOfFile]
```

Comments on entries:

- Lines starting with ; are comment lines.
- **line 3: minimum points player skills**

  Here are skills of the player listed and the minimum values, which must have these skills so that the level is offered by the travel agency. All listed skills must have the named minimum number of points. If this line is empty the level is always offered by the travel agency (if type of level is Single). The skills are described in the manual ‚Ya3dag_Scripting_Language.PDF', chapter ‚PlayerSkills@...'.

  In addition, these names can be used:

nTreasures	Number of treasures found until now
nVisitsMaze	Number of visits to the labyrinth

- **line 4: type of level**

Single	This level is offered by the travel agency
Multi	This level is enumerated for multiplayer games

- **line 5: short description of level**

  The text in this line may not be more than 22 characters. The optional text **&level_wgfireland_sname=** is a multi-language reference. See on another place for this.

- **line 8: long description of level, lines until [EndOfFile]**

  Up to 6 lines of text. A single line can not be longer than 30 characters. ‚\n‘ can be used as line separator. The optional text **&level_wgfireland_lname=** is a multi-language reference. See on another place for this.

5.2 Time length of the day

The length of the day is 24 minutes for Ya3dag.

For testing the value of 24 can be changed as follows:

- Press the tilde key (~) or the (^) key to pull down the Console window.
- Type ‚DayLengthInMinutes‘ to get the current value.
- Type ‚DayLengthInMinutes XX‘ to set a the new value.

5.3 Displaying a level-image while loading

Level-images are displayed during the load a level. The offers by travel agency „TravelOverland“ also use these pictures.

- Put a jpg-image with the name of level in the subdirectory levelshots (for example: WGCastle1.jpg).

- Give the image a size of 732 * 576. If necessary, use an image-processing software for the task to resize the image.

- The F12 key stores a new screenshot in the subdirectory scrnshot. These screenshots are a good starting point for your level-images.

- Prepare the screen for screenshots
  - Press the tilde key (~) or the (^) key to pull down the Console window.
  - Type ‚hud‘ to toggle the Head-Up-Display or status-bar.
  - Type ‚fps‘ to toggle the ‚frames per second‘ display.
  - Type ‚noclip‘ to fly around in the level.
  - Press the tilde key (~) or the (^) key to remove the Console window.
  - Press the minus key on the number-block until you have the ego-view mode.

5.4 Support of multiple languages

Multi language reference looks like **„&tag=text“**. Virtually anywhere that a text is output, this reference can be used.
Tag is an identification used in language files. Text will be output if no such an identification is found in any language file.
See the subdirectory languages for language files. The subdirectories you see there are listed in the language selection of Ya3dag‘s Options/interface menu.
There is also a tool called **Ya3dag_LanguageExtract.exe** and **Ya3dag_LanguageInitial.bat** to extract language references from files and to build initial language files. See in the language subdirectory of a Ya3dag

source download.

5.5 HUD language

This tutorial will help you understand the HUD layout and control, in two steps:

- HUD macro language [file: cl_scrn.c, function SCR_ExecuteLayoutString()]
- STAT_* values [file: q_shared.h]

The layout of the HUD is described by a very simple macro language (see the setting ‚single_statusbar' in any of the ‚GameConfiguration.txt' files).
Also all in game dialogs, message boxes, book displays and the iBag menu use the HUD macro language to format the information for display on the screen.

Cursor positioning

For positioning a screen size of 640 * 480 units is used (whatever side fits better to the current screen resolution).
0,0 is the center of the screen with x going negative to the right and y going negative to the top.

- xl <value>
  X position from the left side of the physical screen
- xr <value>
  X position from the right side of the physical screen
- yb <value>
  Y position from the bottom of the physical screen
- yt <value>
  Y position from the top of the physical screen
- xv <value>
  X position from the left of a centered rectangle of size 320 * 240
- yv <value>
  X position from the top of a centered rectangle of size 320 * 240
- xc <value>
  X position relative to the center of the physical screen
- yc <value>
  Y position relative to the center of the physical screen
- xi <value>
  Increment X position
- yi <value>
  Increment Y position
- xs <value>
  X position from the last saved position
- ys <value>
  Y position from the last saved position
- xys
  Save current position

Drawing

A standard HUD character is 8 screen units wide and high (for example, the string "FPH" is 3 chars wide, 3 * 8 = 24 units).
Big HUD characters have a width of 16 and an height of 24 screen units.
In all cases <stat> is an integer representing the stat(us) message item to be presented (see later for ‚STAT_* values').
If not otherwise noted, the images are located inside the ‚pics' subdirectory.

- stat_string <stat>
  Use <stat> value as index to one of the ‚configstrings‘ (whatever this is). This is used for the pickup-string of inventory icons.
- stat_num <stat>
  Use <stat> value and output as number string.
- stat_num2 <stat>
  Use <stat> value and output as number string with a minimum with of 3 characters.
- string <string>
  Output the <string>.
- s1 <string>
  This is the short form of ‚string‘.
- string2 <string>
  Output the <string> highlighted.
- s2 <string>
  This is the short form of ‚string2‘.
- cstring <string>
  Output the <string> relative to center of current x position.
- c1 <string>
  This is the short form of ‚cstring‘.
- cstring2 <string>
  Same as ‚cstring‘ but highlighted.
- c2 <string>
  This is the short form of ‚cstring2‘.
- cstring3 <string>
  Save as ‚cstring2‘ but with a frame around the text.
- c3 <string>
  This is the short form of ‚cstring3‘.
- num <fieldwidth> <stat>
  Use <stat> value and output as number string using big HUD characters. <fieldwidth> is the number of character to used (to keep it proper aligned).
- hnum
  Use the <stat> value for the player health and output with a field with of 3. Also colors the output on low values.
- anum
  Use the <stat> value for the player ammo and output with a field with of 3. Also colors the output on low values.
- rnum
  Use the <stat> value for the player armor and output with a field with of 3. Also colors the output on low values.
- barh <value> <MaxValue> <ID> <SizeFak>
  Output a horizontal bar. If <value> is negative, use it as <stat> index and pick up that value. <ID> identifies one of the ‚bics/bar/barh<ID>.png‘ images. The value is related to <MaxValue> and that part of the image is drawn.<SizeFak> scales the size of the bar image.
- barv <value> <MaxValue> <ID> <SizeFak>
  Same as ‚barh‘ but vertical an using images ‚bics/bar/barv<ID>.png‘.
- box <width> <height> <BGndImage>
  Draw a box (with characters from the current font) for <width> * <height> characters. <BGndImage> is the name for a background image.
- box2 <width> <height> <BGndImage> <FrameImage>
  Draw a box for <width> * <height> characters. <BGndImage> is the name for a background image. The frame is constructed from the image <FrameImage>.
- pic <stat>
  Draw a picture from a <stat> number. Use <stat> value as index to one of the ‚configstrings‘ (whatever this is). The string there is the name for a picture of an inventory icons. Size to draw is taken from the image.
- pics <size> <stat>
  Same as above but the square size is given.

- psn <size> <ImageNr>
  Draw a picture from a <ImageNr> number. Use <ImageNr> as index to one of the ‚configstrings' (whatever this is). The string there is the name for a picture of an inventory icons. A square size is given as argument.
- picn <image>
  Draw a picture from the string <image>. Size to draw is taken from the image.
- picsn <size> <image>
  Same as above but the square size is given.
- pics2n <width> <height> <image>
  Draw a picture from the string <image>. With and height for drawing are arguments.
- pican <image>
  Draw a transparent picture from the string <image>. Size to draw is taken from the image.
- picasn <size> <image>
  Same as above but the square size is given.
- picas2n <width> <height> <image>
  Draw a transparent picture from the string <image>. With and height for drawing are arguments
- pisk <size> <character>
  Draw a picture from a icon bind to the key <character>. A square size is given as argument.
- level <levelname> <width> <height> <Xoffset>
  Draw one of the <image> in the ‚levelshots' directory. Also a frame is drawn around the levelshot image.
- image <image> <width> <height> <Xoffset>
  Draw the <image>.
- compass
  Draw the compass. The size is 64 * 64 units.
- clock
  Draw the clock. The size is 32 * 32 units.
- map
  Draw the map. Size is about 148 * 148 units.
- client <Xpos> <Ypos> <ClientNr> <score> <ping> <time>
  Draw a deathmatch client block.
- ctf <Xpos> <Ypos> <ClientNr> <score> <ping>
  Draw a ctf client block.

Control

- if <stat>
  If <stat> value not zero then do the statements until the corresponding endif statement.
- endif
  End of an if block.

STAT_* values

There are 32 status-objects. The <stat> number is directly referring to an array each player-entity has; status-array or stats[]. It is defined in file: q_shared.h in the typedef struct player_state_t (at the very bottom of the file).

- 0 STAT_HEALTH_ICON
  Image nr of the health icon (see the ‚pic <stat>‘ statement).
- 1 STAT_HEALTH
  Players health value.
- 2 STAT_AMMO_ICON
  Image nr of the current ammo (see the ‚pic <stat>‘ statement).
- 3 STAT_AMMO
  Players ammo count.
- 4 STAT_ARMOR_ICON
  Image nr of the current armor (see the ‚pic <stat>‘ statement).
- 5 STAT_ARMOR
  Players armor amount.
- 6 STAT_SELECTED_ICON
  If > 0, image nr of current selected item (see the ‚pic <stat>‘ statement).
- 7 STAT_PICKUP_ICON
  If > 0, image nr of picked up item (see the ‚pic <stat>‘ statement). After an item pickup, this is reset to 0 after 3 seconds.
- 8 STAT_PICKUP_STRING
  Item description of picked up item, see STAT_PICKUP_ICON.
- 9 STAT_TIMER_ICON
  If > 0, image nr of ican performing a timed action (like quad damage powerup).
- 10 STAT_TIMER
  Count down timer, seconds to go. See STAT_TIMER_ICON.
- 11 STAT_HELPICON
  If > 0, image nr of help icon. This is not used by Ya3dag.
- 12 STAT_SELECTED_ITEM
  Item nr of current selected item.
- 13 STAT_LAYOUTS
  Save layout hints for drawing. It is used for deathmatch or or other score displays.
- 14 STAT_FRAGS
  Players amout of frags.
- 15 STAT_FLASHES
  Cleared each frame to zero. Flash numbers. 1 = health, 2 = armor.
- 16 STAT_CHASE
  If > 0, a player nr. Used by spectators. Only valid if STAT_SPECTATOR is > 0.
- 17 STAT_SPECTATOR
  If > 0, this player is spectating. See STAT_CHASE.
- 18 STAT_AIR
  Remaining air in percent, 0 for no air, 1 no more air, 100 have all air.
- 22 STAT_SPEED
  If > 0, the player drives a vehicle. This is an image nr of a speed indicator.
- 23 STAT_ZOOM
  If > 0, a zoom view is activated. This is an image nr of the zoom icon.
- 24 STAT_MONEY
  Players amount of money.
- 25 STAT_MANA
  Playera amount of mana.

- 26 STAT_SELECTED_WEAPON
  If > 0, image nr of the selected or active weapon.
- 27 STAT_SELECTED_ICONCOUNT
  Count of current selected item, see STAT_SELECTED_ICON.
- 28 STAT_SHOW_COMPASS
  If > 0, show the compass.
- 29 STAT_INVENT_CUR_WEIGHT
  Current weight of items in inventory.
- 30 STAT_INVENT_MAX_WEIGHT
  Max weight of items in inventory the player can carry.
- 31 STAT_SHOW_MAP
  If > 0, show the map.

5.6 Fonts

Ya3dag supports TTF and bitmap fonts. There can be a font for the console, the menu and the game/hud.
Character codes used are according to ISO/IEC 8859-1 (or Latin1).

ISO/IEC 8859-1

Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	not assigned															
1...																
2...	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8...	not assigned															
9...																
A...	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
B...	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C...	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D...	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E...	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F...	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

See http://en.wikipedia.org/wiki/Latin1 for more information.
The image above is from the German wikipedia.

TTF fonts

To speed up character display, Ya3dag renders TTF fonts in three sizes to a bitmap.

Bitmap fonts monospaced

The characters are organized in a 16 x 16 grid (see the image above). Please note that only the characters 0x20 to 0x7F and 0xA0 to 0xFF are used (hexadecimal notation).
The bitmaps can be of type .png, .tif or .pcx.

Bitmap fonts proportional spaced

Ya3dag supports output from the **BMFont** tool. This program will allow you to generate bitmap fonts from TrueType fonts. The application generates both image files and character descriptions (.fnt files) that can be read by Ya3dag.
The benefit of this program is the generation of characters with an outline.

See http://www.angelcode.com/products/bmfont/ for more.

Graphic character fonts

The codes 0x00 to 0x1F and 0x80 to 0x9F are used for graphic characters like checkbox or bars.
Only bitmaps are supported. Here the character are organized in a 8 x 8 grid.
The only exception is the font used for the editor. Here the graphic characters are integrated into the font.

Pixelated fonts

You do not like blurred characters. If the name of a bitmap font ends in the word **Pixel**, a nearest neighbor algorithm is used to get sharp characters.