make.tol de Ink.Watercolor

Programa constructor del sitio web del dominio inkwatercolor.com dedicado a contenidos de arte creados con tintas y acuarelas y con otras diferentes técnicas. Los contenidos que emplea para crearlo son: a) registros de obras de arte, en 2 idiomas, castellano e inglés, que se organizan en en un directorio, que se denomina agenda y b) ficheros en formato JPG con imágenes de las obras de arte en diversos tanaños (pequeños, más grandes, DIN A4 y DIN A3). // El directorio de agenda puede contener varios ficheros, en especial, contiene un fichero por cada galería de arte, de esta forma, hay un fichero para: a) la galería llamada chppho, que es la principal, y que está dedicada a obras de pintura automática realizadas en tintas y acuarelas, b) la galería llamada fleurs, que está dedicada a las Flores del Mal de Charles Baudelaire, hay una pintura para cada uno de sus poemas y c) la galería llamada inarmy, que está dedicada a esquemas y apuntes rápidos de la vida castrense.

Este programa para Ink Watercolor: a) Se clasifica de metaprogramación porque se ha escrito código en lenguaje de programación TOL que escribe, a su vez, código en lenguaje de Html para las páginas del sitio web, código TOL que representa los resultados de los análisis estadísticos y código en lenguaje Javascript para la gestión de contenidos artísticos. b) Se clasifica de Hipertexto porque el resultado final más importante es la generación de más de un millar de páginas de hipertexto. c) Se clasifica de arte porque el objetivo es la difusión de contenidos artísticos por internet.

La información sobre las 3 galerías de arte de este sitio web inkwatercolor.com se organiza en un fichero para cada galería, cada fichero contiene de decenas a centenas de registros, tantos registros como obras de arte tiene la galería. // Cada registro de estos ficheros tiene atributos para describir el título, el autor, las técnicas, materiales empleados, los colores, la fecha de realización, la calidad de la obra, etc. // Ha de notarse que este esquema de agenda es el precursos de las estructuras de agendas de posts que se emplean que otros sitios web generados con TOL.

Este programa para la creación del sitio web inkwatercolor.com es de los primeros desarrollados en TOL para la creación de páginas web y conserva mucha de su programación original que tiene las caracteríscas siguientes: a) La creación de las páginas Html se realiza en fases, primero construyendo semillas de páginas web (templeates) a partir de semillas de trozos comunes de páginas web. Por ejemplo, con semillas de títulos, de menús, de estructuras para cuadros, etc. se crean semillas (templates) para páginas que han de albergar cuadros, unas para inglés otras para castellano, unas para 1 cuadro, para 4 cuadros, para listados de cuadros, etc. b) Tiene funcionalidad para crea de forma automática: índices, como listados de enlaces, de ilustraciones, pero los índices principales del sitio web se tienen que crear de forma manual. c) Tiene funcionalidad para rellenar automáticamente la descripción y la lista de palabras clave de cada página web y las crea de una forma personalizada para cada página dependiendo del cuadro o de los cuadros que alberga (campos meta), de forma que la descripción de una página es un resumen da la descripción de sus cuadros, pero las descripciones y las listas de palabras clave que genera no son especialmente largas. d) Delega en Javascript todas aquellas funcionalidades para las que, en el momento de su desarrollo, Javascript parecía más adecuado.

Las funciones principales de este programa para la creación de inkwatercolor.com y que son accesibles a través de sus diferentes opciones, son: a) Crear las páginas de las obras de arte de todas las galerías, el contenido fundamental de estas páginas es un cuadro y su descripción, utiliza unas imágenes que se llaman zoom. b) Crear páginas con las obras de arte de cada galería agrupadas de 4 en 4, estas páginas contienen unas imágenes menores que se llaman tiny, obsérvese que este uso de imágenes de menor tamaño era frecuente cuando el ancho de banda disponible en internet era mucho más limitado. c) Crear listados de índice a obras agrupados de 16 en 16 con enlace a cada una de las obras. d) Secuenciar los enlaces entre todas las obras, por ejemplo, entre las páginas en inglés y castellano de una misma obra, entre cada obra y algunas de las que le siguen o le anteceden, etc. Para la creación de estos enlaces se emplean los números únicos que identifica a haca cuadro en cada garería. e) Crear la homepage del sitio web inkwatercolor.com. f) Crear las páginas por defecto (los index.html) para cada uno de sus directorios y subdirectorios y ha de tenerse en cuenta que en el contenido de estos index.html se da prioridad a la última galería que se haya ejecutado. g) Crear una estructura de pinturas e información sobre ella para ser utilizadas desde Javascript. h) Crear el mapa del sitio web sitemap.xml y ficheros de mandatos para la transmision por FTP, la programación de estas 2 funcionalidades es, en este caso, dependiente de este sitio web y es por lo que sus ficheros TOL están en el directorio de funciones de aplicación (app) y no en el directorio de funciones comunes (cmm).

El sitio web inkwatercolor.com se organiza en una estructura de directorios algo más compleja que otros sitios web desarrollados con TOL. // Este sitio web tiene un directorio para cada una de sus galerías de arte, que a su vez tiene subdirectorios para: a) sus páginas en inglés (page), b) sus páginas en castellano (pags), c) las semillas de páginas, templates, en inglés (sede), d) las semillas de páginas castellano (seds), e) un subdirectorio para imágenes (image) que, a su vez, tiene subdirectorios, por ejemplo, para imágenes pequeñas (tiny), imágenes más grandes (zoom), imágenes limpias y procesadas (clean), escaneados originales de las obras (scan) imágenes den formatos DIN A3 y DIN A4 (da3 y da4). // Además de un directorio para contenidos comunes a todas las galerías con subdirectorios para: a) estilo de páginas web (css), b) imágenes comunes en formato GIF (gif), c) partes de templates para componer semillas para páginas webs (seed) y d) código en lenguaje Javascript (src).

Los comentarios del código de este programa están realizados utilizando unas veces el español no acentuado dentro del código y otras veces el inglés. Se ha comprobado el funcionamiento de este programa para las versiones de Tol 1.1.5, 1.1.6 y 2.0.1. Con la version 1.1.1 da problemas, por ejemplo, por el uso que se hace del tercer parámetro de la función de texto TextReplace(texto, tabla, numero de ciclos), ya que la versión de Tol 1.1.1 no se contemplaba el número de ciclos.

Árbol de ficheros

Ink.Watercolor construye las páginas del sitio web inkwatercolor.com

- ← make.tol proceso principal de generación de contenidos del sitio web
- tol directorios de código fuente en lenguaje de programación Tol
  - cmm funciones comunes de textos, fechas, conjuntos, ficheros, etc.
    - ← txt.tol funciones de manejo de textos

- ← set.tol funciones de manejo de conjuntos
- ← tab.tol funciones de tablas como set of sets
- ← ser.tol funciones de series temporales
- ← fil.tol funciones de gestión de ficheros
- ← zip.tol compresor de ficheros en línea de mandatos
- ← apa.tol proceso de ficheros de log de Apache
- ← dir.tol funciones de gestión de directorios

app funciones especificas de Ink.Watercolor

- ← pdb.tol funciones de la base de datos de pinturas
- ← pag.tol funciones para generar páginas web Html
- ← sed.tol semillas, templates, de páginas web Html
- ← ftp.tol funciones para generar mandatos para hacer Ftp
- ← xml.tol funciones históricas para sitemaps en Xml
- ← alc.tol alchemy para la transformación de imágenes
- ← ink.tol funciones auxiliares de InkWatercolor

← inc.tol inclusión de los ficheros Tol básicos y de aplicación

agenda directorio destinado a albergar los ficheros de agendas de posts

- ← chpphodb01.txt ejemplo con las 4 primeras obras artísticas que se incluyeron

web directorio destinado a las paginas web generadas automáticamente

common directorio de recursos comunes a todas las galerías

css directorio para ficheros de estilo

- ← common.css fichero de estilo para las paginas Html del sitio web

seed trozos de código Html para construir templates

- ← strseed.htm estructura básica de página Html de inkwatercolor.com
- ← pi1lowseed.htm estructura para albergar información sobre una obra
- ← pi4cntseed.htm estructura para albergar 4 pinturas en una página

src directorio para ficheros javascript

- ← fleursdtspanish.js código Javascript generado automáticamente por TOL
- ← inarmydtspanish.js código Javascript generado automáticamente por TOL

chppho directorio para la galería principal de Inkwatercolor

- → index.html ejemplo de página Html generada automáticamente

download directorio para material extra para descargas

- → chppho0129.jpg ejemplo de imagen de obra artística para descargar
- → chppho0143.jpg ejemplo de imagen de obra artística para descargar

→ sitemap.xml mapa del sitio web generado en Xml de forma automática

history archivo de registro histórico del programa Ink.Watercolor

→ 20151202.inkwatercolor.picture1.jpg ejemplo de página generada automáticamente para 1 pintura

→ 20151202.inkwatercolor.picture4.jpg ejemplo de página generada automáticamente para 4 pinturas

→ 20151202.fleurs.picture1.jpg ejemplo de página web de las flores del mal con 1 pintura

→ 20151202.fleurs.picture4.jpg ejemplo de página web de las flores del mal con 4 pinturas

→ ink_watercolor.pdf documento resumen de funciones del programa de generación Html

Declaraciones

Inclusiones

- `Set allInc`
  Inclusion de las funciones comunes y de aplicacion.

Constantes

- `TimeSet TmsMth`
  Conjunto temporal de los meses, llamado mensual.
- `TimeSet TmsWee`
  Conjunto temporal de las semanas, llamado semanal.
- `TimeSet TmsDay`
  Conjunto temporal de los meses, llamado mensual.

Proceso

- `Text ctrExe`
  Funcion que hay que realizarn en el make.
- `Text PagGal`
  Galeria para la que se ha de realizar la funcion.
- `Real makHlp`
  Es cierto si se ha visualizado la ayuda.
- `Real makSed`
  Build seeds files.
- `Real makJpg`
  Build jpgs files.
- `Real makSt_`
  Build statistics and control information.
- `Real makWeb`

Build web pages.

- `Real makUpd`
  Pages with last updates.
- `Real makXsm`
  Build XML sitemap.
- `Real makFtp`
  Build FTP command files.
- `Real makJbd`
  Building javascript picture database.

Set allInc

```
////////////////////////////////////////////////////////////////////////////
Set  allInc = Include("tol/inc.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Inclusion de las funciones comunes y de aplicacion.", allInc);
////////////////////////////////////////////////////////////////////////////
```

Estructuras de datos

```
Struct PdbSt
{
  Text picCla,
  Text picSer,
  Real picWid,
  Real picHei,
  Date endDte,
  Date updDte,
  Real picNum,
  Text picTit,
  Real wrkTim,
  Real wrkQly,
  Set  papTec,
  Set  papCol,
  Set  bckTec,
  Set  bckCol,
  Set  forTec,
  Set  forCol,
  Text picAut,
  Text picOwn,
  Text jpgFil,
  Real picPrn,
  Text picRem
};


Struct CtrSt
{
  Text picNum,
  Real picPrn
};
```

Constantes

TimeSet TmsMth

```
////////////////////////////////////////////////////////////////////////
TimeSet TmsMth = D(1);
////////////////////////////////////////////////////////////////////////
PutDescription("Conjunto temporal de los meses, llamado mensual.", TmsMth);
////////////////////////////////////////////////////////////////////////
```

TimeSet TmsWee

```
////////////////////////////////////////////////////////////////////////
TimeSet TmsWee = WD(1);
////////////////////////////////////////////////////////////////////////
PutDescription("Conjunto temporal de las semanas, llamado semanal.", TmsWee);
////////////////////////////////////////////////////////////////////////
```

TimeSet TmsDay

```
////////////////////////////////////////////////////////////////////////
TimeSet TmsDay = C;
////////////////////////////////////////////////////////////////////////
PutDescription("Conjunto temporal de los meses, llamado mensual.", TmsDay);
////////////////////////////////////////////////////////////////////////
```

Text ctrExe

```
////////////////////////////////////////////////////////////////////////
Text ctrExe = If(Not(ObjectExist("Text","ctrBat")),  "web",
               If(ctrBat=="",                        "hlp",
                                                     ToLower(ctrBat)));
////////////////////////////////////////////////////////////////////////
PutDescription("Funcion que hay que realizarn en el make.", ctrExe);
////////////////////////////////////////////////////////////////////////
```

Text PagGal

```
////////////////////////////////////////////////////////////////////////
Text PagGal = If(Not(ObjectExist("Text","ctrGal")),  "chppho",
               If(ctrGal=="",                        "chppho",
               If(!(ToLower(ctrGal) <: [["chppho";
                                         "fleurs";
                                         "inarmy"]]), "chppho",
                                                     ToLower(ctrGal))));
////////////////////////////////////////////////////////////////////////
PutDescription("Galeria para la que se ha de realizar la funcion.", PagGal);
////////////////////////////////////////////////////////////////////////
```

Real makHlp

```
////////////////////////////////////////////////////////////////////////
Real makHlp = If(ctrExe!="hlp", FALSE,
{
  Text WriteLn(PdbSep+"\nhelp:"+
    "\nUsage: make [OPTION] [GALLERY]"+
    "\nBuilds inkwatercolor.com site"+
    "\nOPTION"+
    "\n  sed: build seeds files"+
    "\n  jpg: build jpegs and zip files"+
    "\n  st0: build fast stats"+
    "\n  st1: build stat plus painting counts (slow)"+
```

```
    "\n  web: build html pages"+
    "\n  upd: build updated html pages"+
    "\n  xsm: build sitemap.xml"+
    "\n  ftp: build ftp files (go to ftp dir and run manually chppho.bat)"+
    "\n  jdb: build javascript pictures database for forms and budgets"+
    "\n  hlp: view this help"+
    "\n  all: do all usual works (sed+web+upd+xsm)"+
    "\nGALLERY"+
    "\n  chppho: ink & watercolor"+
    "\n  fleurs: flowers of evil"+
    "\n  inarmy: ink in the army");
  TRUE
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Es cierto si se ha visualizado la ayuda.", makHlp);
////////////////////////////////////////////////////////////////////////////
```

Real makSed

```
////////////////////////////////////////////////////////////////////////////
Real makSed = If(And(ctrExe!="all",ctrExe!="sed"), FALSE,
{
  SedMake(TRUE) +
  SedMake(FALSE)
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Build seeds files.", makSed);
////////////////////////////////////////////////////////////////////////////
```

Real makJpg

```
////////////////////////////////////////////////////////////////////////////
Real makJpg = If(ctrExe!="jpg", FALSE,
{
//Text extImg = If(PagGal=="chppho", "*.tif", "*.jpg"); Old
  Text extImg = "*.jpg";

  Text WriteLn(PdbSep+"\nbuilding jpgs...");
  Set  chpSet = DirFiles("web/"+PagGal+"/image/clean", extImg, TRUE, TRUE);

  Set  chpCic = EvalSet(chpSet, Real(Text clsTif)
  {
    Text WriteLn(clsTif+":");
    Text clsJpg = Replace(clsTif, ".tif",     ".jpg");
    Text tinJpg = Replace(clsJpg, "/clean/", "/tiny/");
    Text zooJpg = Replace(clsJpg, "/clean/", "/zoom/");
    Text da5Jpg = Replace(clsJpg, "/clean/", "/da5/");
    Text da4Jpg = Replace(clsJpg, "/clean/", "/da4/");
    Text da4Zip = Replace(da4Jpg, ".jpg",     ".zip");
    Text da3Jpg = Replace(clsJpg, "/clean/", "/da3/");
    Text da3Zip = Replace(da3Jpg, ".jpg",     ".zip");

    Real tinDon = FilMake(clsTif, tinJpg, Real(Text inp, Text out)
                          { AlcImg2JpgDpi(inp, out,  400,  600, 0) });

    Real zooDon = FilMake(clsTif, zooJpg, Real(Text inp, Text out)
                          { AlcImg2JpgDpi(inp, out,  800, 1200, 0) });

//  Real zipDon = FilMakeUpdate(da4Jpg, da4Zip, ZipAdd);

    Real da5Don = If(! DirExist("web/"+PagGal+"/image/da5"), FALSE,
                     FilMake(clsTif, da5Jpg, Real(Text inp, Text out)
                             { AlcImg2JpgDpi(inp, out,  519,  719,  72) }));
//                           { AlcImg2JpgDpi(inp, out, 1539, 2244, 300) }));

    Real da4Don = If(! DirExist("web/"+PagGal+"/image/da4"), FALSE,
                     FilMake(clsTif, da4Jpg, Real(Text inp, Text out)
                             { AlcImg2JpgDpi(inp, out, 2244, 3272, 300) }));
```

```
    Real da3Don = If(! DirExist("web/"+PagGal+"/image/da3"), FALSE,
                     FilMake(clsTif, da3Jpg, Real(Text inp, Text out)
                          { AlcImg2JpgDpi(inp, out, 3272, 4724, 300) }));

    tinDon+zooDon+da4Don+da3Don //+zipDon
  });
  SetSum(chpCic)
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Build jpgs files.", makJpg);
////////////////////////////////////////////////////////////////////////////
```

Real makSt_

```
////////////////////////////////////////////////////////////////////////////
Real makSt_ = If(Sub(ctrExe,1,2)!="st", FALSE,
{
  Text WriteLn(PdbSep+"\nOnly Apache logs statistics...");
  InkLogsStats(20)
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Build statistics and control information.", makSt_);
////////////////////////////////////////////////////////////////////////////
```

Real makWeb

```
////////////////////////////////////////////////////////////////////////////
Real makWeb = If(And(ctrExe!="all",ctrExe!="web"), FALSE,
{
  Real eng = TRUE;

  Text WriteLn(PdbSep+"\nbuilding web...");

// index.html

  Text WriteLn(PdbSep+"\nbuilding index.html from index.htm...");
  Text inxHtm = ReadFile(PagSee(eng)+"/index.htm");
  Text inxHt0 = Replace(inxHtm,"../../","");
  Text inxHt1 = Replace(inxHtm,"../../","../");
  Text inxHt2 = inxHtm;
  Text inxHt3 = Replace(inxHtm,"../../","../../../");


  Text d00 = "web";
  Text WriteFile(d00+"/index.html",inxHt0); // Nivel 0
  Set  dir001 = GetDir(d00)[2];
  Set  cic001 = EvalSet(dir001, Real(Text nam)
  {
    Text d01 = d00+"/"+nam;
    Text WriteFile(d01+"/index.html",inxHt1);

    Set  dir002 = GetDir(d01)[2];
    Set  cic002 = EvalSet(dir002, Real(Text nam)
    {
      Text d02 = d01+"/"+nam;
      Text WriteFile(d02+"/index.html",inxHt2);

      Set  dir003 = GetDir(d02)[2];
      Set  cic003 = EvalSet(dir003, Real(Text nam)
      {
        Text d03 = d02+"/"+nam;
        Text WriteFile(d03+"/index.html",inxHt3);
```

```
        TRUE
      });
      TRUE
    });
    TRUE
  });

// indice.html

  Text WriteLn(PdbSep+"\nbuilding indice.html from spanish index.htm...");
  Text indHtm = ReadFile(PagSee(!eng)+"/index.htm"); // !eng -> spanish
  Text indHt0 = Replace(indHtm,"../../","");

  Text WriteFile(dOO+"/indice.html",indHt0); // Nivel 0


// inf*.htm

  Text WriteLn(PdbSep+"\nbuilding inf*.htm...");
  Set  ctrInc = Include("ctr/ctr.tol");

  Text inpFil = "agenda/"+PagGal+"db01.txt";
  Set  inpBst = IncludeBST("ctr/img."+PagGal+".bst");
  Set  inpPdb = PdbRead(inpFil, inpBst);


// Gallery

  Set  selSad =
  {
    Select(inpPdb, Real(Set picReg)
           { FileExist("web/"+PagGal+"/image/zoom/"+picReg->jpgFil) })
  };

  Text keySad = "sad"; // set all drawings
  Set  yeaEng = PagYearTab( eng, selSad, keySad);
  Set  yeaSpa = PagYearTab(!eng, selSad, keySad);

  Set  infEng = DirFiles("web/chppho/sede", "inf*", TRUE, TRUE);
  Set  cicEng = EvalSet(infEng, Real(Text infPth)
                        { InkPageInfo( eng, infPth, yeaEng) });

  Set  infSpa = DirFiles("web/chppho/seds", "inf*", TRUE, TRUE);
  Set  cicSpa = EvalSet(infSpa, Real(Text infPth)
                        { InkPageInfo(!eng, infPth, yeaSpa) });

  Text WriteLn(PdbSep+"\nabsolute pages...");
  Text thkFilEng = "web/chppho/page/inffeethk.htm";
  Text thkHtmEng = ReadFile(thkFilEng);
  Text WriteFile(thkFilEng, Replace(thkHtmEng, "../../",
                                    "http://www.inkwatercolor.com/"));
  Text thkFilSpa = "web/chppho/pags/inffeethk.htm";
  Text thkHtmSpa = ReadFile(thkFilSpa);
  Text WriteFile(thkFilSpa, Replace(thkHtmSpa, "../../",
                                    "http://www.inkwatercolor.com/"));


  Text WriteLn(PdbSep+"\nbuilding sad*.htm...");
  Real sadEng = PagSel( eng, selSad, keySad, yeaEng, PagSee( eng));
  Real sadSpa = PagSel(!eng, selSad, keySad, yeaSpa, PagSee(!eng));

  Real comXid = FilCopy("../common/dir.html", "web/comxidir.html", TRUE);

  sadEng+sadSpa+comXid
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Build web pages.", makWeb);
////////////////////////////////////////////////////////////////////////////
```

Real makUpd

```
////////////////////////////////////////////////////////////////////////////
Real makUpd = If(And(ctrExe!="all",ctrExe!="upd"), FALSE,
{
  Real eng = TRUE;

  Text WriteLn(PdbSep+"\nbuilding last updated...");

  Text inpFil = "agenda/"+PagGal+"db01.txt";
  Set  inpBst = IncludeBST("ctr/img."+PagGal+".bst");
  Set  inpPdb = PdbRead(inpFil, inpBst);
  Set  selUpd = PdbLastUpdates(inpPdb, 32);

  Text keyUpd = "upd"; // set last updates
  Set  yeaEng = PagYearTab( eng, selUpd, keyUpd);
  Set  yeaSpa = PagYearTab(!eng, selUpd, keyUpd);

  Real updEng = PagSel( eng, selUpd, keyUpd, yeaEng, PagSee( eng));
  Real updSpa = PagSel(!eng, selUpd, keyUpd, yeaSpa, PagSee(!eng));

  updEng+updSpa
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Pages with last updates.", makUpd);
////////////////////////////////////////////////////////////////////////////
```

Real makXsm

```
////////////////////////////////////////////////////////////////////////////
Real makXsm = If(And(ctrExe!="all",ctrExe!="xsm"), FALSE,
{
  Text WriteLn(PdbSep+"\nbuilding xml site map...");
  XsmDir("web/sitemap.xml","web")
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Build XML sitemap.", makXsm);
////////////////////////////////////////////////////////////////////////////
```

Real makFtp

```
////////////////////////////////////////////////////////////////////////////
Real makFtp = If(ctrExe!="ftp", FALSE,
{
  Text absPth = Replace(GetSourcePath(ctrExe),"/make.tol","");     // Absoluto
  Text locPth = absPth+"/web";                  // Ha de ser una ruta absoluta

  Text WriteLn(PdbSep+"\nbuilding "+locPth+" ftp...");

// Real FtpRecovery("ftp", "recover.log", "recover");

  FtpAll(locPth)
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Build FTP command files.", makFtp);
////////////////////////////////////////////////////////////////////////////
```

Real makJbd

```
////////////////////////////////////////////////////////////////////////////
Real makJbd = If(ctrExe!="jdb", FALSE,
{
  Real eng = TRUE;

  Text WriteLn(PdbSep+"\nbuilding javascript picture database...");
```

```
  Text inpFil = "agenda/"+PagGal+"db01.txt";
  Set  inpBst = IncludeBST("ctr/img."+PagGal+".bst");
  Set  inpPdb = PdbRead(inpFil, inpBst);
  Text outDir = "web/common/src"; // Output directory
  Text pthSee = "web/common/seed/jdbitem.js"; // Seed file

  Real jdbEng = PagJavascriptDB( eng, inpPdb, outDir, pthSee);
  Real jdbSpa = PagJavascriptDB(!eng, inpPdb, outDir, pthSee);

  jdbEng+jdbSpa
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Building javascript picture database.", makJbd);
////////////////////////////////////////////////////////////////////////////
```

txt.tol de Ink.Watercolor

Text functions.

Declaraciones

Funciones

- `Date Txt2Dte(Text txt, Text fmt)`
  Returns a date from a text using some formats. dd/mmm/yyyy | 07/Aug/2003 -> y2003m08d07 If the format is unknown returns the UnknownDate.
- `Real Txt2Month(Text txt)`
  Returns a month number from the english, spanish, portuguese or french month names (using this order). If several months names match then returns the first one. If none month match then returns 0.
- `Set Txt2Set(Text txt, Set sep, Real cmp)`
  Returns a set of texts like TOL function Tokenizer(). Can use a set of separators of any length. Assumes that there are not Char(1) inside the text. If argument sep is the Empty set then assumes ; as default separator. If argument cmp is true then apply the Compact() function. For example: Txt2Set(' a / b -- c / d / e / f ', [['/','--']], TRUE) returns [['a','b','c','d','e','f']]
- `Text TxtBetween2Tag(Text txt, Text tagIni, Text tagEnd, Real cmp)`
  Returns a subtext of text between the first ocurrence of tagIni and tagEnd. If tagIni or tagEnd does not ocurr then returns the empty text. If cmp argument is true then apply the Compact() function to the returning text. For example: TxtBetween2Tag('a b [[c]] d [[e]] f', '[[', ']]', TRUE) returns a 'c'. There are other version of this function with more functionalities.
- `Text TxtInside2Tag(Text txt, Text tagIni, Text tagEnd)`
  Returns all the texts between 2 tags (tagIni and tagEnd) in txt. For example: <aaa(::)bbb(---)ccc>, <(>, <)> -> <::--->.
- `Text TxtOutside2Tag(Text txtInp, Text tagIni, Text tagEnd)`
  Returns all the texts outside 2 tags (tagIni and tagEnd) in txt. For example: <aaa(::)bbb(:::)ccc>, <(>, <)> -> <aaabbbccc>.
- `Text TxtOutHtmTag(Text htmTxt)`
  Returns all text outside the Html tags.
- `Text TxtOutHtmScr(Text htmTxt)`
  Returns all text outside the Html tags and outside the scripts.
- `Text TxtRand(Real len)`
  Returns a random text with len chars from A to Z. Use Min() y Max() because in some old versions of TOL the Rand() functions sometimes returns numbers out of range.
- `Set TxtTokenizer(Text txtInp, Text tagBrk)`

Returns a set breaking the input text txtInp by the token tagBrk. The lenght of the token tagBrk can be 1 or more characters. This function use the Tol function Tokenizer() that breaks by only one character. This function assume that txtInp do not contain the character 7 (bell).

Date Txt2Dte()

```
////////////////////////////////////////////////////////////////////////////
Date Txt2Dte(Text txt, // Text
             Text fmt) // Format
////////////////////////////////////////////////////////////////////////////
{
  Case
  (
    fmt=="dd/mmm/yyyy",
    { //  123456789.1
      Real day = Eval     (Sub(txt, 1, 2)+"; ");
      Real yea = Eval     (Sub(txt, 8,11)+"; ");
      Real mth = Txt2Month(Sub(txt, 4, 6));
      If(Not(mth), UnknownDate, YMD(yea, mth, day))
    },
    TRUE, UnknownDate
  )
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a date from a text using some formats.
dd/mmm/yyyy | 07/Aug/2003 -> y2003m08d07
If the format is unknown returns the UnknownDate.",
Txt2Dte);
////////////////////////////////////////////////////////////////////////////
```

Real Txt2Month()

```
////////////////////////////////////////////////////////////////////////////
Real Txt2Month(Text txt) // Text 3 or more letter
////////////////////////////////////////////////////////////////////////////
{
  Set engSet = SetTxtBeginWith(SetEnglishMonth, txt, FALSE);
  If(GT(Card(engSet), 0), engSet[1],
  {
    Set spaSet = SetTxtBeginWith(SetSpanishMonth, txt, FALSE);
    If(GT(Card(spaSet), 0), spaSet[1],
    {
      Set porSet = SetTxtBeginWith(SetPortugueseMonth, txt, FALSE);
      If(GT(Card(porSet), 0), porSet[1],
      {
        Set freSet = SetTxtBeginWith(SetFrenchMonth, txt, FALSE);
        If(GT(Card(freSet), 0), freSet[1], 0)
      })
    })
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a month number from the english, spanish, portuguese or french month
names (using this order).
If several months names match then returns the first one.
If none month match then returns 0.",
Txt2Month);
////////////////////////////////////////////////////////////////////////////
```

Set Txt2Set()

```
////////////////////////////////////////////////////////////////////////////
Set Txt2Set(Text txt, // Text
            Set  sep, // Set of separators
            Real cmp) // If true apply the Compact() function
////////////////////////////////////////////////////////////////////////////
{
  Text sepUni = Char(1); // Unique separator with only one character
  Set  sepTab = If(EQ(Card(sep),0), [[ [[";", sepUni ]] ]],
                   EvalSet(sep, Set(Text s) { [[s, sepUni]] }));
  Text txtRep = ReplaceTable(txt, sepTab);
  Set  setTok = Tokenizer(txtRep, sepUni);
  If(cmp, EvalSet(setTok, Text(Text txt) { Compact(txt) }), setTok)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a set of texts like TOL function Tokenizer().
Can use a set of separators of any length.
Assumes that there are not Char(1) inside the text.
If argument sep is the Empty set then assumes ; as default separator.
If argument cmp is true then apply the Compact() function.
For example: Txt2Set(' a / b -- c / d / e / f ', [['/','--']], TRUE)
returns [['a','b','c','d','e','f']]",
Txt2Set);
////////////////////////////////////////////////////////////////////////////
```

Text TxtBetween2Tag()

```
////////////////////////////////////////////////////////////////////////////
Text TxtBetween2Tag(Text txt,    // Text
                    Text tagIni, // Initial tag
                    Text tagEnd, // End tag
                    Real cmp)    // If true apply the Compact() function
////////////////////////////////////////////////////////////////////////////
{
  Real posIni = TextFind(txt, tagIni);
  Text result =
    If(LE(posIni,0), "",
          {
            Real lenIni = TextLength(tagIni);
            Real posSub = posIni + lenIni;
            Real posEnd = TextFind(txt, tagEnd, posSub);
            Text subTxt = If(LE(posEnd,0),
                             Sub(txt,posSub,TextLength(txt)),
                             Sub(txt,posSub,posEnd-1));
            subTxt
          });
  If(cmp, Compact(result), result)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a subtext of text between the first ocurrence of tagIni and tagEnd.
If tagIni or tagEnd does not ocurr then returns the empty text.
If cmp argument is true then apply the Compact() function to the returning
text.
For example: TxtBetween2Tag('a b [[ c ]] d [[ e ]] f', '[[', ']]', TRUE)
returns a 'c'.
There are other version of this function with more functionalities.",
TxtBetween2Tag);
////////////////////////////////////////////////////////////////////////////
```

Text TxtInside2Tag()

```
////////////////////////////////////////////////////////////////////////////
Text TxtInside2Tag(Text txt, Text tagIni, Text tagEnd)
//
////////////////////////////////////////////////////////////////////////////
```

```
{
  Real posIni = TextFind(txt, tagIni);
  Text result = If(LE(posIni,0), "",
  {
    Real lenIni = TextLength(tagIni);
    Real posSub = posIni + lenIni;
    Real posEnd = TextFind(txt, tagEnd, posSub);
    If(And(EQ(posIni,1),LE(posEnd,0)), txt,
    If(And(GT(posIni,1),LE(posEnd,0)), Sub(txt,posIni, TextLength(txt)),
       Sub(txt,posIni,posEnd+TextLength(tagEnd)-1)+ // Recursion
       TxtInside2Tag(Sub(txt, posEnd+TextLength(tagEnd), TextLength(txt)),
                     tagIni, tagEnd)))
  });
  ReplaceTable(result, [[ [[tagIni, ""]], [[tagEnd, ""]] ]])
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns all the texts between 2 tags (tagIni and tagEnd) in txt.
For example: <aaa(::)bbb(---)ccc>, <(>, <)> -> <::--->.",
TxtInside2Tag);
////////////////////////////////////////////////////////////////////////////
```

Text TxtOutside2Tag()

```
////////////////////////////////////////////////////////////////////////////
Text TxtOutside2Tag(Text txtInp, // Input text
                    Text tagIni, // Initial tag
                    Text tagEnd) // End tag)
////////////////////////////////////////////////////////////////////////////
{
  Set  txtSet = TxtTokenizer(tagIni + tagEnd + txtInp, tagIni);
  Set  txtCic = EvalSet(txtSet, Text(Text txtTok)
  { TxtBetween2Tag(txtTok + tagIni, tagEnd, tagIni, FALSE) });
  SetSum(txtCic)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns all the texts outside 2 tags (tagIni and tagEnd) in txt.
For example: <aaa(::)bbb(:::)ccc>, <(>, <)> -> <aaabbbccc>.",
TxtOutside2Tag);
////////////////////////////////////////////////////////////////////////////
```

Text TxtOutHtmTag()

```
////////////////////////////////////////////////////////////////////////////
Text TxtOutHtmTag(Text htmTxt)
////////////////////////////////////////////////////////////////////////////
{ TxtOutside2Tag(htmTxt, "<", ">") };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns all text outside the Html tags.",
TxtOutHtmTag);
////////////////////////////////////////////////////////////////////////////
```

Text TxtOutHtmScr()

```
////////////////////////////////////////////////////////////////////////////
Text TxtOutHtmScr(Text htmTxt)
////////////////////////////////////////////////////////////////////////////
{ TxtOutHtmTag(TxtOutside2Tag(htmTxt, "<script", "</script>")) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns all text outside the Html tags and outside the scripts.",
TxtOutHtmScr);
////////////////////////////////////////////////////////////////////////////
```

Text TxtRand()

```
////////////////////////////////////////////////////////////////////////////
Text TxtRand(Real len) // Random text length
////////////////////////////////////////////////////////////////////////////
{ SetSum(For(1,len,Text(Real t) { Char(Min(90,Max(65,Rand(64,91)))) })) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a random text with len chars from A to Z.
Use Min() y Max() because in some old versions of TOL the Rand() functions
sometimes returns numbers out of range.",
TxtRand);
////////////////////////////////////////////////////////////////////////////
```

Set TxtTokenizer()

```
////////////////////////////////////////////////////////////////////////////
Set TxtTokenizer(Text txtInp, // Texto de entrada
                 Text tagBrk) // Tag por el que se corta
////////////////////////////////////////////////////////////////////////////
{ Tokenizer(Replace(txtInp, tagBrk, Char(7)), Char(7)) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a set breaking the input text txtInp by the token tagBrk.
The lenght of the token tagBrk can be 1 or more characters. This function use
the Tol function Tokenizer() that breaks by only one character.
This function assume that txtInp do not contain the character 7 (bell).",
TxtTokenizer);
////////////////////////////////////////////////////////////////////////////
```

set.tol de Ink.Watercolor

Set functions.

Declaraciones

Constantes

- `Set SetSpanishMonth`
  Set of spanish months names.
- `Set SetSpanishWeekday`
  Set of spanish weekdays names.
- `Set SetEnglishMonth`
  Set of english months names.
- `Set SetEnglishWeekday`
  Set of english weekdays names.
- `Set SetPortugueseMonth`
  Set of portuguese months names.
- `Set SetFrenchMonth`
  Set of french months names.

Funciones

- `Text Set2TxtFormat(Anything val)`
  Internal function, do not call it. Only for the use of Set2Txt() function.
- `Text Set2Txt(Set valSet, Text iniTxt, Text endTxt, Text sepTxt, Text sepLst, Text txtDet, Text datFmt, Text dteDet, Text dteFmt)`
  Returns a text like a list with all the elements of valSet converted in a text format (elements types: Text, Real or Date). Arguments: - iniTxt initial text, for example: '(', '[[', '', etc. - endTxt end text, for example ')', ']]', '', etc. - sepTxt elements separator, for example '; ', ',', '', etc. - sepLst two last elements separator, for example, ' & ', ' and ', etc., you can specify the same as sepTxt - txtDet text delimiters, for exmple, quotes for TOL, single quote for SQL, nothing, etc. - datFmt real numbers format, for explame, '%.0lf' for integers, if none then uses the default TOL real number format. - dteDet date delimiters, for example, simgle quote for SQL. - dteFmt date format, for example, '%c%Y%m%d', if none then uses the default TOL dates format. Only works with TOL types Text, Real or Date, when find a Set type then works in a recursive way with the same arguments.
- `Text Set2TxtKeyword(Set valSet, Real minChr, Real a2zOrd)`
  Returns a text list with all the elements of valSet converted in a text format with commas like a keywords list, ordered and without repetitions. Remove all word with LE(TextLength(), minChr). For example Set2TxtKeyword([['cc, bb & dd',1,'dd ee',2,'ff',1,'aa']]) returns: 1.0, 2.0, aa, bb, cc, dd, ee, ff.
- `Text Set2TxtCommaAmp(Set valSet)`

Return a text list with all the elements of valSet converted in a text format with commas and & in Html style. For example Set2TxtCommaAmp([['a','b','c','d']]) returns: a, b, c & d -> in html -> a, b, c & d.

- `Set SetFirstN(Set set, Real num)`
  Returns a subset with the first num elements. If the set does not have num elements returns the set.

- `Set SetFirstNByFun(Set set, Real num, Code funSor)`
  Returns a subset with the first num elements of a set ordered by funSor. If the set does not have num elements returns the set ordered by funSor.

- `Serie Set2Ser(Set dteVal, TimeSet serTms)`
  Returns a serie in serTms time set using the dates and values of dteVal. The First() and the Last() date of the serie are the minimun and the maximum dates of the table dteVal. The date in dteVal cauld be orderder or disordered. If the table is Empty then return the a zero constant serie. For the dates without value the value zero is assumed. For the dates with several values all values are added. The dates in dteVal table belong to C timeSet. For example: Serie ser008 = Set2Ser([[[[y2004m01d03, 3]], // Only odd days [[y2004m01d07, 3]], [[y2004m01d07, 4]], [[y2004m01d01, 1]], [[y2004m01d05, 5]]]], C); returns the serie: C; ser008; 2004/01/01; 1.000000; 2004/01/02; 0.000000; 2004/01/03; 3.000000; 2004/01/04; 0.000000; 2004/01/05; 5.000000; 2004/01/06; 0.000000; 2004/01/07; 7.000000;

- `Text Set2TxtTol(Set valSet, Text datFmt)`
  Return a text like a TOL list with all the elements of valSet converted in a text format. For example Set2TxtTol([['text',1.0,y2003m12d31]], '%.0lf') returns: [['text', 1, y2003m12d31]].

- `Text Set2TxtSql(Set valSet, Text datFmt)`
  Return a text like a SQL list with all the elements of valSet converted in a text format. For example Set2TxtSql([['text',1.0,y2003m12d31]], '%.0lf') returns: (´text´, 1, ´2003/12/31´).

- `Set SetInclude(Set filSet)`
  Returns a one level set with the result of the inclusion of a set of TOL files. The returning set looks like all the definitions will be declared in the same (virtually single) TOL file.

- `Set SetTxtBeginWith(Set set, Text txt, Real cas)`
  Returns a set of set positions where the texts of the set begin with the argument txt. If none match then returns the empty set. If the argument cas is true then work in a case sensitive way.

- `Set SetTxtCount(Set set, Real noNull)`
  Returns a table with the differents texts insider a set a count of theirs ocurrences. If noNull is true the empty texts are not counted. For example SetTxtCount([['aa','bb','aa','cc','','dd','bb']], TRUE) returns: 'aa', 2 'bb', 2 'cc', 1 'dd', 1 If there are not texts different than empty text retunrs Empty

- `Real SetTxtFindFirst(Set set, Text txt, Real cas)`
  Returns the position of the first text element of set equal to txt. If none match then returns 0. If the argument cas is true then work in a case sensitive way.

- `Text SetReplaceTable(Text txt, Set set, Real pos)`
  As ReplaceTable() but with only one loop and in strict order.

Constantes

Set SetSpanishMonth

```
////////////////////////////////////////////////////////////////////////////
Set SetSpanishMonth =
  [[ "Enero", "Febrero", "Marzo",      "Abril",   "Mayo",      "Junio",
     "Julio", "Agosto",  "Septiembre", "Octubre", "Noviembre", "Diciembre" ]];
////////////////////////////////////////////////////////////////////////////
PutDescription("Set of spanish months names.", SetSpanishMonth);
////////////////////////////////////////////////////////////////////////////
```

Set SetSpanishWeekday

```
////////////////////////////////////////////////////////////////////////////
Set SetSpanishWeekday =
  [[ "Lunes",   "Martes", "Miercoles", "Jueves",
     "Viernes", "Sabado", "Domingo" ]];
////////////////////////////////////////////////////////////////////////////
PutDescription("Set of spanish weekdays names.", SetSpanishWeekday);
////////////////////////////////////////////////////////////////////////////
```

Set SetEnglishMonth

```
////////////////////////////////////////////////////////////////////////////
Set SetEnglishMonth =
  [[ "January", "February", "March",     "April",   "May",      "June",
     "July",    "August",   "September", "October", "November", "December"  ]]
////////////////////////////////////////////////////////////////////////////
PutDescription("Set of english months names.", SetEnglishMonth);
////////////////////////////////////////////////////////////////////////////
```

Set SetEnglishWeekday

```
////////////////////////////////////////////////////////////////////////////
Set SetEnglishWeekday =
  [[ "Monday", "Tuesday", "Wednesday", "Thursday",
     "Friday", "Saturday", "Sunday" ]];
////////////////////////////////////////////////////////////////////////////
PutDescription("Set of english weekdays names.", SetEnglishWeekday);
////////////////////////////////////////////////////////////////////////////
```

Set SetPortugueseMonth

```
////////////////////////////////////////////////////////////////////////////
Set SetPortugueseMonth =
  [[ "Janeiro", "Fevereiro", "Março",    "Abril",   "Maio",     "Junho",
     "Julho",   "Agosto",    "Setembro", "Outubro", "Novembro", "Dezembro"  ]]
////////////////////////////////////////////////////////////////////////////
PutDescription("Set of portuguese months names.", SetPortugueseMonth);
////////////////////////////////////////////////////////////////////////////
```

Set SetFrenchMonth

```
////////////////////////////////////////////////////////////////////////////
```

```
Set SetFrenchMonth =
  [[ "Janvier", "Février", "Mars",      "Avril",   "Mai",      "Juin",
     "Juillet", "Août",    "Septembre", "Octobre", "Novembre", "Décembre"  ]];
////////////////////////////////////////////////////////////////////////////
PutDescription("Set of french months names.", SetFrenchMonth);
////////////////////////////////////////////////////////////////////////////
```

Text Set2TxtFormat()

```
////////////////////////////////////////////////////////////////////////////
Text Set2TxtFormat(Anything val)
////////////////////////////////////////////////////////////////////////////
{
  Text gra = Grammar(val);
  If(gra=="Text", txtDet+val+txtDet,
  If(gra=="Real", If(datFmt=="", FormatReal(val),
                                 FormatReal(val,datFmt)),
  If(gra=="Date", dteDet+If(dteFmt=="", FormatDate(val),
                                        FormatDate(val,dteFmt))+dteDet,
  If(gra=="Set",  Set2Txt(val,    iniTxt, endTxt,
                          sepTxt, sepLst,
                          txtDet, datFmt, dteDet, dteFmt),
                  "Not valid grammar"))))
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Internal function, do not call it.
Only for the use of Set2Txt() function.",
Set2TxtFormat);
////////////////////////////////////////////////////////////////////////////
```

Text Set2Txt()

```
////////////////////////////////////////////////////////////////////////////
Text Set2Txt(Set  valSet, // Set of elements
             Text iniTxt, // Initial text for list
             Text endTxt, // End text for list
             Text sepTxt, // Element separators
             Text sepLst, // 2 last elements separator
             Text txtDet, // Delimiter for texts
             Text datFmt, // Format for real numbers
             Text dteDet, // Delimiter for dates
             Text dteFmt) // Format for dates
////////////////////////////////////////////////////////////////////////////
{
  Real card = Card(valSet);
  Text body =
    If(EQ(card,0), "",
    If(EQ(card,1), Set2TxtFormat(valSet[1]),
    If(EQ(card,2), Set2TxtFormat(valSet[1])+sepLst+Set2TxtFormat(valSet[2]),
    {
      Set txtVal = For(2,card,Text(Real p)
      {
        If(EQ(p,card),sepLst,sepTxt) + Set2TxtFormat(valSet[p])
      });
      Set2TxtFormat(valSet[1]) + BinGroup("+",txtVal)
    })));
  iniTxt+body+endTxt
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a text like a list with all the elements of valSet converted in a
text format (elements types: Text, Real or Date).
Arguments:
 - iniTxt initial text, for example: '(', '[[', '', etc.
 - endTxt end text, for example ')', ']]', '', etc.
 - sepTxt elements separator, for example '; ', ', ', '', etc.
 - sepLst two last elements separator, for example, ' & ', ' and ', etc.,
```

```
    you can specify the same as sepTxt
  - txtDet text delimiters, for exmple, quotes for TOL, single quote for
    SQL, nothing, etc.
  - datFmt real numbers format, for explame, '%.0lf' for integers, if none
    then uses the default TOL real number format.
  - dteDet date delimiters, for example, simgle quote for SQL.
  - dteFmt date format, for example, '%c%Y%m%d', if none
    then uses the default TOL dates format.
Only works with TOL types Text, Real or Date, when find a Set type then
works in a recursive way with the same arguments.",
Set2Txt);
////////////////////////////////////////////////////////////////////////////
```

Text Set2TxtKeyword()

```
////////////////////////////////////////////////////////////////////////////
Text Set2TxtKeyword(Set  valSet, // Set of elements
                    Real minChr, // Minimum number of chars
                    Real a2zOrd) // If true ordered
////////////////////////////////////////////////////////////////////////////
{
  // Convert all to text
  Text lstTxt = ToLower(Set2Txt(valSet,"", "", " ", " ", "", "", "", ""));
  Text lstCls = Compact(ReplaceTable(TxtOutHtmScr(lstTxt),
    [[ [[Char(34), " "]],
       [["'",      " "]],
       [[". ",     " "]],
       [[":",      " "]],
       [[";",      " "]],
       [[",",      " "]],
       [["[",      " "]],
       [["]",      " "]],
       [["(",      " "]],
       [[")",      " "]],
       [["/",      " "]],
       [["¡",      " "]],
       [["!",      " "]],
       [["¿",      " "]],
       [["?",      " "]],
       [["&amp",   " "]],
       [[" none ", " "]],
       [[" the ",  " "]],
       [[" and ",  " "]],
       [[" or ",   " "]],
       [[" & ",    " "]],
       [[" y ",    " "]],
       [[" o ",    " "]]
    ]]));
  Set  setTxt = Tokenizer(lstCls, " "); // Obtain a new set
  Set  setUni = Unique(setTxt); // Without repetitions
  Set  setSel = Select(setUni, Real(Text a) { GT(TextLength(a), minChr) });
  Set  setSor = If(! a2zOrd, setSel,
                   Sort(setSel, Real(Text a, Text b)
                     { Compare(ToLower(a),ToLower(b)) })); // Order
  Set2Txt(setSor, "", "", ", ", ", ", "", "", "", "")
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a text list with all the elements of valSet converted in a
text format with commas like a keywords list, ordered and without repetitions.
Remove all word with LE(TextLength(), minChr).
For example Set2TxtKeyword([['cc, bb & dd',1,'dd ee',2,'ff',1,'aa']]) returns:
 1.0, 2.0,  aa, bb, cc, dd, ee, ff.",
Set2TxtKeyword);
////////////////////////////////////////////////////////////////////////////
```

Text Set2TxtCommaAmp()

```
////////////////////////////////////////////////////////////////////////
Text Set2TxtCommaAmp(Set  valSet) // Set of elements
////////////////////////////////////////////////////////////////////////
{ Set2Txt(valSet, "", "", ", ", " &amp;amp; ", "", "", "", "") };
////////////////////////////////////////////////////////////////////////
PutDescription(
"Return a text list with all the elements of valSet converted in a
text format with commas and & in Html style.
For example Set2TxtCommaAmp([['a','b','c','d']]) returns:
 a, b, c &amp;amp; d -> in html -> a, b, c & d.",
Set2TxtCommaAmp);
////////////////////////////////////////////////////////////////////////
```

Set SetFirstN()

```
////////////////////////////////////////////////////////////////////////
Set SetFirstN(Set set, Real num)
////////////////////////////////////////////////////////////////////////
{ For(1, Min(Card(set), num), Anything(Real pos) { set[pos] }) };
////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a subset with the first num elements.
If the set does not have num elements returns the set.",
SetFirstN);
////////////////////////////////////////////////////////////////////////
```

Set SetFirstNByFun()

```
////////////////////////////////////////////////////////////////////////
Set SetFirstNByFun(Set set, Real num, Code funSor)
////////////////////////////////////////////////////////////////////////
{ SetFirstN(Sort(set, funSor), num) };
////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a subset with the first num elements of a set ordered by funSor.
If the set does not have num elements returns the set ordered by funSor.",
SetFirstNByFun);
////////////////////////////////////////////////////////////////////////
```

Serie Set2Ser()

```
////////////////////////////////////////////////////////////////////////
Serie Set2Ser(Set     dteVal, // Set table with dates and values
              TimeSet serTms) // Time set
////////////////////////////////////////////////////////////////////////
{
  If(EQ(Card(dteVal),0), CalInd(W,serTms), // Constant zero
  {
    // Order by date
    Set   tabSor = Sort(dteVal, Real(Set a, Set b) { Compare(a[1],b[1]) });

    Date  fstDte = tabSor[           1][1];
    Date  lstDte = tabSor[Card(tabSor)][1];

    Set   pulSet = EvalSet(tabSor, Serie(Set row) // For all rows in dteVal
    {
//    Date  dte = row[1]; // Date
//    Real  val = row[2]; // Value
```

```
      Pulse(row[1],C)*row[2]
    });

    DatCh(SubSer(SetSum(pulSet), fstDte, lstDte),serTms,SumS)
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a serie in serTms time set using the dates and values of dteVal.
The First() and the Last() date of the serie are the minimun and the maximum
dates of the table dteVal.
The date in dteVal cauld be orderder or disordered.
If the table is Empty then return the a zero constant serie.
For the dates without value the value zero is assumed.
For the dates with several values all values are added.
The dates in dteVal table belong to C timeSet.
For example:
Serie ser008 = Set2Ser([[ [[y2004m01d03, 3]], // Only odd days
                         [[y2004m01d07, 3]],
                         [[y2004m01d07, 4]],
                         [[y2004m01d01, 1]],
                         [[y2004m01d05, 5]] ]], C);

returns the serie:
C; ser008;
2004/01/01; 1.000000;
2004/01/02; 0.000000;
2004/01/03; 3.000000;
2004/01/04; 0.000000;
2004/01/05; 5.000000;
2004/01/06; 0.000000;
2004/01/07; 7.000000;",
Set2Ser);
////////////////////////////////////////////////////////////////////////////
```

Text Set2TxtTol()

```
////////////////////////////////////////////////////////////////////////////
Text Set2TxtTol(Set  valSet, // Set of elements
                Text datFmt) // Format for real numbers
////////////////////////////////////////////////////////////////////////////
{
  Text iniBra = "[[ ";
  Text endBra = " ]]";
  Text tolLst = Set2Txt(valSet, iniBra, endBra, ", ", ", ",
                                Char(34), datFmt, "", "%cy%Ym%md%d");
  Replace(tolLst, iniBra+endBra, "Empty")
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Return a text like a TOL list with all the elements of valSet converted in a
text format.
For example Set2TxtTol([['text',1.0,y2003m12d31]], '%.0lf') returns:
 [[ 'text', 1, y2003m12d31 ]].",
Set2TxtTol);
////////////////////////////////////////////////////////////////////////////
```

Text Set2TxtSql()

```
////////////////////////////////////////////////////////////////////////////
Text Set2TxtSql(Set  valSet, // Set of elements
                Text datFmt) // Format for real numbers
////////////////////////////////////////////////////////////////////////////
{
  Set2Txt(valSet,
              "(", ")", ", ", ", ", "'", datFmt, "'", "%c%Y/%m/%d")
};
////////////////////////////////////////////////////////////////////////////
```

```
PutDescription(
"Return a text like a SQL list with all the elements of valSet converted in a
text format.
For example Set2TxtSql([['text',1.0,y2003m12d31]], '%.0lf') returns:
 (´text´, 1, ´2003/12/31´).",
Set2TxtSql);
////////////////////////////////////////////////////////////////////////////
```

Set SetInclude()

```
////////////////////////////////////////////////////////////////////////////
Set SetInclude(Set filSet) // Set of TOL files paths
////////////////////////////////////////////////////////////////////////////
{
  // Select existing files
  Set filSel = Select(filSet, Real(Text filPth) { FileExist(filPth) });

  If(EQ(Card(filSel),0), Empty,               // None
  If(EQ(Card(filSel),1), Include(filSel[1]), // Only one file
  { // Include existing files
    Set setInc = EvalSet(filSel, Set(Text filPth) { Include(filPth) });
    BinGroup("<<", setInc) // Group to one level all definitions
  }))
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a one level set with the result of the inclusion of a set of TOL
files.
The returning set looks like all the definitions will be declared in the same
(virtually single) TOL file.",
SetInclude);
////////////////////////////////////////////////////////////////////////////
```

Set SetTxtBeginWith()

```
////////////////////////////////////////////////////////////////////////////
Set SetTxtBeginWith(Set  set, // Set of texts
                    Text txt, // Text to find
                    Real cas) // If TRUE then case sensitive
////////////////////////////////////////////////////////////////////////////
{
  Text low = If(cas, txt, ToLower(txt));
  Real car = Card(set);
  Set  cic = For(1, car, Real(Real pos)
  {
    Text beg = If(cas, set[pos], ToLower(set[pos]));
    If(TextBeginWith(beg, low), pos, 0)
  });
  Select(cic, Real(Real pos) { GT(pos,0) })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a set of set positions where the texts of the set begin with the
argument txt.
If none match then returns the empty set.
If the argument cas is true then work in a case sensitive way.",
SetTxtBeginWith);
////////////////////////////////////////////////////////////////////////////
```

Set SetTxtCount()

```
////////////////////////////////////////////////////////////////////////////
Set SetTxtCount(Set  set,    // Set of texts
                Real noNull) // If true the empty texts are not counted
////////////////////////////////////////////////////////////////////////////
```

```
{
  // Selects text different than empty text
  Set  sel = If(noNull, Select(set, Real(Text txt) { txt!="" }), set);

  Real emp = EQ(Card(sel),0); // True if there aren't texts
  If(emp, Empty,
  {
    // Classify by text
    Set  cla = Classify(sel, Real(Text a, Text b) { Compare(a,b) });
    EvalSet(cla, Set(Set ele)
    {
      Text ide = ele[1];    // The first ocurrence
      Real num = Card(ele); // Number of ocurrences
      [[ide, num]]
    })
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a table with the differents texts insider a set a count of theirs
ocurrences.
If noNull is true the empty texts are not counted.
For example SetTxtCount([['aa','bb','aa','cc','','dd','bb']], TRUE) returns:
  'aa', 2
  'bb', 2
  'cc', 1
  'dd', 1
If there are not texts different than empty text retunrs Empty",
SetTxtCount);
////////////////////////////////////////////////////////////////////////////
```

Real SetTxtFindFirst()

```
////////////////////////////////////////////////////////////////////////////
Real SetTxtFindFirst(Set  set, // Set of texts
                     Text txt, // Text to find
                     Real cas) // If TRUE then case sensitive
////////////////////////////////////////////////////////////////////////////
{
  Text low = If(cas, txt, ToLower(txt));
  Real car = Card(set);
  Set  cic = For(1, car, Real(Real pos)
  {
    Text beg = If(cas, set[pos], ToLower(set[pos]));
    If(beg==low, pos, 0)
  });
  Set  sel = Select(cic, Real(Real pos) { GT(pos,0) });
  If(EQ(Card(sel),0), 0, sel[1])
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the position of the first text element of set equal to txt.
If none match then returns 0.
If the argument cas is true then work in a case sensitive way.",
SetTxtFindFirst);
////////////////////////////////////////////////////////////////////////////
```

Text SetReplaceTable()

```
////////////////////////////////////////////////////////////////////////////
Text SetReplaceTable(Text txt,
                     Set  set,
                     Real pos) // From set[pos] toset[Card(set)]
////////////////////////////////////////////////////////////////////////////
{
  If(GT(pos, Card(set)), txt, // end recursion
     SetReplaceTable(Replace(txt, set[pos][1], set[pos][2]), set, pos+1))
```

```
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"As ReplaceTable() but with only one loop and in strict order.",
SetReplaceTable);
////////////////////////////////////////////////////////////////////////////
```

tab.tol de Ink.Watercolor

Table functions (set of set in tabular form).

Declaraciones

ser.tol de Ink.Watercolor

Time series functions.

Declaraciones

Funciones

- `Serie SerIntegrate(Serie ser)`
  Resturn the integration of the serie ser with initial value 0.

Serie SerIntegrate()

```
////////////////////////////////////////////////////////////////////////////
Serie SerIntegrate(Serie ser)
////////////////////////////////////////////////////////////////////////////
{ DifEq(1/(1-B), ser, 0) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Resturn the integration of the serie ser with initial value 0.",
SerIntegrate);
////////////////////////////////////////////////////////////////////////////
```

fil.tol de Ink.Watercolor

File functions.

Declaraciones

zip.tol de Ink.Watercolor

WinZip functions WinZip Command Line Copyright WinZip Computing, Inc.

Declaraciones

Funciones

- `Real ZipAdd(Text inpPth, Text outPth)`
  Add inpPth file to outPth zip file with maximum compression.

Constantes

```
Text ZipPth = "c:/ARCHIV~1/WinZip/wzzip"; // Executable path
```

Real ZipAdd()

```
////////////////////////////////////////////////////////////////////////////
Real ZipAdd(Text inpPth, // Input file path
            Text outPth) // Output zip file path
////////////////////////////////////////////////////////////////////////////
{
  Text zipDos = Replace(ZipPth,"/","\\")+" ";
  Text inpDos = Replace(inpPth,"/","\\");
  Text outDos = Replace(outPth,"/","\\");

  Real sysExe = System(zipDos+                  // Executable
                       "-a -ex "+               // Options add maximum
                       outDos+" "+              // Output zip file path
                       inpDos);                 // Input file path
  sysExe
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Add inpPth file to outPth zip file with maximum compression.",
ZipAdd);
////////////////////////////////////////////////////////////////////////////
```

apa.tol de Ink.Watercolor

Apache log file functions

Declaraciones

Funciones

- `Set ApaBetween2Tag(Set linSet, Text tagIni, Text tagEnd)`
  Returns a table with different text that appear between 2 tags. For each text there are a line [[text, number of ocurrences (Real)]].
- `Set ApaBetween2TagFile(Text logPth, Text tagIni, Text tagEnd)`
  Returns a table with different text that appear between 2 tags inside a log file. For each text there are a line [[text, number of ocurrences (Real)]].
- `Set ApaLineSet(Text logPth)`
  Returns a set with all lines of a log file. The empty lines are removed.
- `Set ApaUrl(Set linSet)`
  Returns a table with different urls of a set of log lines. For each url there are a line [[url (Text), number of ocurrences (Real)]].
- `Set ApaUrlFile(Text logPth)`
  Returns a table with different urls of a log file. For each url there are a line [[url (Text), number of ocurrences (Real)]].
- `Real ApaSplitFile(Text logPth, Text outPth, Text prefix, Text filErr)`
  Split an Apache log file (logPth) y several files by date and returns the number lines processed. Each output file has the name: <outPth> + / + <prefix> + YYYYMMDD + .log The error line are stored at filErr.

Set ApaBetween2Tag()

```
////////////////////////////////////////////////////////////////////////////
Set ApaBetween2Tag(Set  linSet, // Set of log lines,
                   Text tagIni, // Initial tag
                   Text tagEnd) // End tag
////////////////////////////////////////////////////////////////////////////
{
  Set  set = EvalSet(linSet, Text(Text lin)
             { TxtBetween2Tag(lin, tagIni, tagEnd, TRUE) });
  SetTxtCount(set, TRUE)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a table with different text that appear between 2 tags.
For each text there are a line [[ text, number of ocurrences (Real)]].",
ApaBetween2Tag);
////////////////////////////////////////////////////////////////////////////
```

Set ApaBetween2TagFile()

```
////////////////////////////////////////////////////////////////////////////
Set ApaBetween2TagFile(Text logPth, // Input file path
```

```
                    Text tagIni, // Initial tag
                    Text tagEnd) // End tag
////////////////////////////////////////////////////////////////////////////
{
  Set  linSet = ApaLineSet(logPth); // Read log file
  ApaBetween2Tag(linSet, tagIni, tagEnd)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a table with different text that appear between 2 tags inside a log
file.
For each text there are a line [[ text, number of ocurrences (Real)]].",
ApaBetween2TagFile);
////////////////////////////////////////////////////////////////////////////
```

Set ApaLineSet()

```
////////////////////////////////////////////////////////////////////////////
Set ApaLineSet(Text logPth) // Input file path
////////////////////////////////////////////////////////////////////////////
{
  If(Not(FileExist(logPth)), Empty,
  {
    Text logTxt = ReadFile(logPth);
    Set  linSet = Tokenizer(logTxt,"\n");
    Select(linSet, Real(Text lin) { lin != "" })
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a set with all lines of a log file.
The empty lines are removed.",
ApaLineSet);
////////////////////////////////////////////////////////////////////////////
```

Set ApaUrl()

```
////////////////////////////////////////////////////////////////////////////
Set ApaUrl(Set linSet) // Set of log lines
////////////////////////////////////////////////////////////////////////////
{
  Set urlSet = EvalSet(linSet, Text(Text lin)
               { Compact(Sub(lin,1,TextFind(lin," - - ["))) });
  SetTxtCount(urlSet, TRUE) // Count different urls that aren't null
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a table with different urls of a set of log lines.
For each url there are a line [[ url (Text), number of ocurrences (Real)]].",
ApaUrl);
////////////////////////////////////////////////////////////////////////////
```

Set ApaUrlFile()

```
////////////////////////////////////////////////////////////////////////////
Set ApaUrlFile(Text logPth) // Input file path
////////////////////////////////////////////////////////////////////////////
{
  Set  linSet = ApaLineSet(logPth); // Read log file
  ApaUrl(linSet)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a table with different urls of a log file.
For each url there are a line [[ url (Text), number of ocurrences (Real)]].",
```

```
ApaUrlFile);
////////////////////////////////////////////////////////////////////////////
```

Real ApaSplitFile()

```
////////////////////////////////////////////////////////////////////////////
Real ApaSplitFile(Text logPth, // Input file path
                  Text outPth, // Output path
                  Text prefix, // File prefix
                  Text filErr) // Error file
////////////////////////////////////////////////////////////////////////////
{
  FilSplit(logPth, TRUE, Text(Text lin)
  {
    Text dteTxt = TxtBetween2Tag(lin, "[", "]", TRUE);
    If(dteTxt=="", filErr,
    {
      Date dte = Txt2Dte(dteTxt, "dd/mmm/yyyy");
      If(dte==UnknownDate, filErr,
         outPth+"/"+prefix+FormatDate(dte,"%c%Y%m%d")+".log")
    })
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Split an Apache log file (logPth) y several files by date and returns the
number lines processed.
Each output file has the name: <outPth> + / + <prefix> + YYYYMMDD + .log
The error line are stored at filErr.",
ApaUrlFile);
////////////////////////////////////////////////////////////////////////////
```

dir.tol de Ink.Watercolor

Directories functions.

Declaraciones

Funciones

- `Set DirFiles(Text dir, Text filPattern, Real toLower, Real casSen)`
  Returns a set of files paths with all the files inside dir (not in its subdirectories) that theirs names match with file pattern filPattern. If toLower is true all names are change to lower case. If casSen is true then the match is case sensitive. The power of pattern matching is the same than TOL funcion TextMatch().
- `Set DirAll(Text dir, Text filPattern, Real toLower, Real casSen)`
  Returns a set of files paths with all the files inside dir and all of its subdirectories that theirs names match with file pattern filPattern. If toLower is true all names are change to lower case. If casSen is true then the match is case sensitive. The power of pattern matching is the same than TOL funcion TextMatch().
- `Real DirUsage(Text dir, Text filPattern, Real casSen)`
  Returns the sum in bytes of the size of the set of files inside dir and all of its subdirectories that theirs names match with filPattern. Example, usage in bytes of all gif files inside a web directory at any level: Real gifSiz = DirUsage(<web>,<*.gif>, FALSE);

Set DirFiles()

```
////////////////////////////////////////////////////////////////////////////
Set DirFiles(Text dir,        // Directory path
             Text filPattern, // File names pattern
             Real toLower,    // If true all names to lower case
             Real casSen)     // If true case sensitive in file names
////////////////////////////////////////////////////////////////////////////
{
  If(Not(DirExist(dir)), Empty,
  {
    Set  getDir = GetDir(dir);
    Set  filSet = getDir[1];

    Set  filFnd = EvalSet(filSet, Text(Text fil)
    {
      If(TextMatch(fil,filPattern,casSen),
         If(toLower, ToLower(dir+"/"+fil), dir+"/"+fil),
         "")
    });

    Set filSel = Select(filFnd, Real(Text fil) { fil!="" });

    filSel
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a set of files paths with all the files inside dir (not in its
subdirectories) that theirs names match with file pattern filPattern.
If toLower is true all names are change to lower case.
If casSen is true then the match is case sensitive.
The power of pattern matching is the same than TOL funcion TextMatch().",
DirFiles);
////////////////////////////////////////////////////////////////////////////
```

Set DirAll()

```
////////////////////////////////////////////////////////////////////////////
Set DirAll(Text dir,        // Directory path
           Text filPattern, // File names pattern
           Real toLower,    // If true all names to lower case
           Real casSen)     // If true case sensitive in file names
////////////////////////////////////////////////////////////////////////////
{
  If(Not(DirExist(dir)), Empty,
  {
    Set  getDir = GetDir(dir);
    Set  filSet = getDir[1];
    Set  dirSet = getDir[2];

    Set  filFnd = EvalSet(filSet, Text(Text fil)
    {
      If(TextMatch(fil,filPattern,casSen),
         If(toLower, ToLower(dir+"/"+fil), dir+"/"+fil),
         "")
    });

    Set filSel = Select(filFnd, Real(Text fil) { fil!="" });

    Set  dirFnd = EvalSet(dirSet, Set(Text subDir)
      { DirAll(dir+"/"+subDir, filPattern, toLower, casSen) });

    Real dirCar   = Card(dirFnd);
    If(EQ(dirCar,0), filSel,
    If(EQ(dirCar,1), filSel<<dirFnd[1],
                     filSel<<BinGroup("+",dirFnd)))
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a set of files paths with all the files inside dir and all of its
subdirectories that theirs names match with file pattern filPattern.
If toLower is true all names are change to lower case.
If casSen is true then the match is case sensitive.
The power of pattern matching is the same than TOL funcion TextMatch().",
DirAll);
////////////////////////////////////////////////////////////////////////////
```

Real DirUsage()

```
////////////////////////////////////////////////////////////////////////////
Real DirUsage(Text dir,        // Directory path
              Text filPattern, // File names pattern
              Real casSen)     // If true case sensitive in file names
////////////////////////////////////////////////////////////////////////////
{
  Set  filSet = DirAll(dir, filPattern, FALSE, casSen);
  If(EQ(Card(filSet),0), 0, SetSum(EvalSet(filSet, FileBytes)))
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the sum in bytes of the size of the set of files inside dir and all
of its subdirectories that theirs names match with filPattern.
Example, usage in bytes of all gif files inside a web directory at any level:
  Real gifSiz = DirUsage(<web>,<*.gif>, FALSE);",
DirUsage);
////////////////////////////////////////////////////////////////////////////
```

pdb.tol de Ink.Watercolor

Pictures database functions. Attributes: Pic.Cla) La clase a la que pertenece la obra de arte, este atributo debería ser coincidente con la galería. Pic.Ser) La serie, dentro de la clase anterior, a la que pertenece la obra. Pic.Wid) El ancho de la pintura en milímetros. Pic.Hei) El alto de la pintura en milímetros. End.Dte) Fecha de terminación del cuadro especificada en lenguaje TOL. Upd.Dte) Fecha de última actualización de los datos de registro del cuadro, fecha que se especificada en TOL. Pic.Num) Número único de identificación del cuadro dentro de su galería, este número actua como clave única de la obra dentro de su galería y es el que se usa para dar nombre a sus páginas web y crear los enlaces entre las diferentes obras de arte. Pic.Tit) Título del cuadro que puede especificarse en inglés y en castellano como { English | Castellano } y que puede contener partes relevantes o principales y accesorias como { rest (main) | resto (principal) }. Wrk.Tim) Número de horas de trabajo empleadas por el artista para completar su obra. Wrk.Qly) Indicador de calidad de la obra, pudiendo emplearse ?, el desconocido en TOL, para aquellas obras no calificadas o pendientes de calificación. Pap.Tec) Técnica de fabricación del soporte, por ejemplo, del papel. Pap.Col) Color o colores de fabricación del soporte, por ejemplo, el color del papel, si son varios colores se separar por punto y coma. Bck.Tec) Técnicas y materiales empleados para crear la base de la obra artística. Bck.Col) Color o colores empleados para crear la base de la obra artística, si son varios colores se separan por punto y coma, por ejemplo, orange; yellow. For.Tec) Técnicas y materiales empleado para crear la obra artística. For.Col) Color o colores empleados para crear la obra artística, si hay más de un color se separan por punto y coma y si es un solo color no se emplea el punto y coma, por ejemplo, black. Pic.Aut) Nombre completo del artista autor de la obra. Pic.Own) Nombre del titular propietario actual de la obra, ya sea persona física o jurídica, o el nombre del último titular conocido o un texto genérico de pertenencia a una colección. Pic.Prn) Número de reproducciones de la obra artística. Pic.Rem) Comentarios que pueden ser en inglés y en castellano y que, a diferencia de los comentarios de las agendas de otros sitios web, si se publican.

Declaraciones

Constantes

- `Text PdbSep`
  Horizontal line between registers.

Funciones

- `Set PdbRead(Text inpFil, Set inpBst)`
  Reads and returns a pictures database.
- `Set PdbRenum(Set inpSet)`
  Returns a pictures database where the picNum is the position.
- `Set PdbLastUpdates(Set inpSet, Real maxNum)`
  Returns the last updates.
- `Real PdbRenamePic(Set picSet, Text dirPth)`
  Not useful now.
- `Real PdbInit(Text outFil, Real maxPic)`

Not used now. This function creates an initial pictures databse.

Constantes

Text PdbSep

```
////////////////////////////////////////////////////////////////////////
Text PdbSep = Repeat("_",78);
////////////////////////////////////////////////////////////////////////
PutDescription("Horizontal line between registers.", PdbSep);
////////////////////////////////////////////////////////////////////////
```

Set PdbRead()

```
////////////////////////////////////////////////////////////////////////
Set PdbRead(Text inpFil, Set inpBst)
////////////////////////////////////////////////////////////////////////
{
  Text WriteLn("Reading "+inpFil+"...");
  Text inpTxt = Replace(ReadFile(inpFil),PdbSep+"\n","·");
  Set  inpSet = Tokenizer(inpTxt,"·");
  Set  inpTab = EvalSet(inpSet, Set(Text inf)
  {
    Text picCla =         TxtBetween2Tag(inf,"<Pic.Cla>", "\n<", TRUE);
    Text picSer =         TxtBetween2Tag(inf,"<Pic.Ser>", "\n<", TRUE);
    Real picWid =    Eval(TxtBetween2Tag(inf,"<Pic.Wid>", "\n<", TRUE));
    Real picHei =    Eval(TxtBetween2Tag(inf,"<Pic.Hei>", "\n<", TRUE));
    Date endDte =    Eval(TxtBetween2Tag(inf,"<End.Dte>", "\n<", TRUE));
    Date updDte =    Eval(TxtBetween2Tag(inf,"<Upd.Dte>", "\n<", TRUE));
    Text picTxt =         TxtBetween2Tag(inf,"<Pic.Num>", "\n<", TRUE);
    Real picNum =    Eval(picTxt);
    Text picTit =         TxtBetween2Tag(inf,"<Pic.Tit>", "\n<", TRUE);

    Real wrkTim =    Eval(TxtBetween2Tag(inf,"<Wrk.Tim>", "\n<", TRUE));
    Real wrkQly =    Eval(TxtBetween2Tag(inf,"<Wrk.Qly>", "\n<", TRUE));
    Set  papTec = Txt2Set(TxtBetween2Tag(inf,"<Pap.Tec>", "\n<", TRUE), Empty,
    Set  papCol = Txt2Set(TxtBetween2Tag(inf,"<Pap.Col>", "\n<", TRUE), Empty,
    Set  bckTec = Txt2Set(TxtBetween2Tag(inf,"<Bck.Tec>", "\n<", TRUE), Empty,
    Set  bckCol = Txt2Set(TxtBetween2Tag(inf,"<Bck.Col>", "\n<", TRUE), Empty,
    Set  forTec = Txt2Set(TxtBetween2Tag(inf,"<For.Tec>", "\n<", TRUE), Empty,
    Set  forCol = Txt2Set(TxtBetween2Tag(inf,"<For.Col>", "\n<", TRUE), Empty,
    Text picAut =         TxtBetween2Tag(inf,"<Pic.Aut>", "\n<", TRUE);
    Text picOwn =         TxtBetween2Tag(inf,"<Pic.Own>", "\n<", TRUE);
    Text jpgFil =         PagGal+PagNumInt(picNum, TRUE)+".jpg";

    Set  picSel = Select(inpBst, Real(Set a) { a[1]==picTxt });
    Real picPrn = If(EQ(Card(picSel),1), picSel[1][2], 0);
    Text picRem =         TxtBetween2Tag(inf,"<Pic.Rem>", "\n<", TRUE);

    PdbSt(picCla,picSer,picWid,picHei,endDte,updDte,picNum,picTit,
          wrkTim,wrkQly,
          papTec,papCol,bckTec,bckCol,forTec,forCol,
          picAut,picOwn,jpgFil,picPrn,picRem)
  });
  Set inpSor = Sort(inpTab, Real(Set a, Set b)
               { Compare(a->picNum,b->picNum) });

  Text WriteLn("Readed "+FormatReal(Card(inpSor),"%.0lf")+" registers");

  inpSor
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Reads and returns a pictures database.",
PdbRead);
////////////////////////////////////////////////////////////////////////
```

Set PdbRenum()

```
////////////////////////////////////////////////////////////////////////////
Set PdbRenum(Set inpSet)
////////////////////////////////////////////////////////////////////////////
{
  Text WriteLn("Renumering database...");
  For(1, Card(inpSet), Set(Real pos)
  {
    Set reg = inpSet[pos];

    PdbSt(reg->picCla,
          reg->picSer,
          reg->picWid,
          reg->picHei,
          reg->endDte,
          reg->updDte,
          pos,         // Change picNum for the position
          reg->picTit,
          reg->wrkTim,
          reg->wrkQly,
          reg->papTec,
          reg->papCol,
          reg->bckTec,
          reg->bckCol,
          reg->forTec,
          reg->forCol,
          reg->picAut,
          reg->picOwn,
          reg->jpgFil,
          reg->picPrn,
          reg->picRem)
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a pictures database where the picNum is the position.",
PdbRenum);
////////////////////////////////////////////////////////////////////////////
```

Set PdbLastUpdates()

```
////////////////////////////////////////////////////////////////////////////
Set PdbLastUpdates(Set  inpSet,
                   Real maxNum)
////////////////////////////////////////////////////////////////////////////
{
  Text WriteLn("Last updates...");

  Set  selLst = SetFirstNByFun(inpPdb, maxNum, Real(Set a, Set b)
                { Compare(b->updDte,a->updDte) }); // Descending

  Set  sorLst = Sort(selLst, Real(Set a, Set b)
                { Compare(a->endDte,b->endDte) }); // Ascending

  PdbRenum(sorLst)  // Renumering
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the last updates.",
PdbLastUpdates);
////////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////////
Real Pag2CtrCsv(Set inpPdb) // Pictures database
////////////////////////////////////////////////////////////////////////////
```

```
{
  Text imgCsv = "ctr/img.csv"; // Control de visualizaciones en Excel
  Text WriteFile(imgCsv,"");

  Set  cicSet = For(1, maxReg, Real(Real numReg)
  {
    Set  picReg = inpPdb[numReg];
    Text AppendFile(imgCsv, FormatReal(picReg->picNum,"%.0lf")+";"+
                            Replace(picReg->picTit,";"," |")+";"+
                            dteTxt+";"+
                            FormatReal(picReg->picPrn,"%.0lf")+";\n");
    TRUE
  });
  Card(cicSet)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Writes a Csv file in control directory (ctr) with the number, title and
the number of visualizations for statistics.",
Pag2CtrCsv);
////////////////////////////////////////////////////////////////////////////
```

Real PdbRenamePic()

```
////////////////////////////////////////////////////////////////////////////
Real PdbRenamePic(Set picSet, Text dirPth)
////////////////////////////////////////////////////////////////////////////
{
  Set  filSet = GetDir(dirPth)[1];
  Set  renSet = For(1, Card(picSet), Real(Real picNum)
  {
    Set  picReg = picSet[picNum];
    Text filNew = picReg->jpgFil;
    Text filOld = filSet[picNum];
    FileRename(dirPth+"/"+filOld, dirPth+"/"+filNew)
  });
  Card(renSet)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Not useful now.",
PdbRenamePic);
////////////////////////////////////////////////////////////////////////////
```

Real PdbInit()

```
////////////////////////////////////////////////////////////////////////////
Real PdbInit(Text outFil, Real maxPic)
////////////////////////////////////////////////////////////////////////////
{
  Text txtSee =
    "<Pic.Cla> Chp\n"+
    "<Pic.Ser> Pho\n"+
    "<Pic.Wid> 310\n"+
    "<Pic.Hei> 460\n"+
    "<End.Dte> y1999m11d30\n"+
    "<Upd.Dte> _UPD_\n"+
    "<Pic.Num> _NUM_\n"+
    "<Pic.Tit> title\n"+
    "<Wrk.Tim> 8\n"+
    "<Wrk.Qly> 1.00\n"+
    "<Pap.Tec> Fabriano paper\n"+
    "<Pap.Col> \n"+
    "<Bck.Tec> watercolor\n"+
    "<Bck.Col> orange\n"+
    "<For.Tec> ink\n"+
    "<For.Col> black\n"+
    "<Pic.Aut> Antonio Salmeron\n"+
```

```
    "<Pic.Own> Antonio Salmeron\n"+
    "<Jpg.Fil> cphpho_NUM_.jpg\n"+
    "<Pic.Prn> _PRN_\n"+
    "<Pic.Rem> \n"+
    "<End>\n";

  If(FileExist(outFil), { WriteLn(outFil+" already exists"); 0 },
  {
    Text WriteFile(outFil, "");
    Set wriSet = For(1, maxPic, Real(Real numPic)
    {
      Date updDte = Succ(y2002m01d01, C, Round(Rand(3,200)));
      Text updTxt = FormatDate(updDte,"%cy%Ym%md%d");
      Text numTxt = FormatReal(numPic, "%04.0lf");
      Text prnTxt = FormatReal(Rand(0,50), "%.0lf");
      Text regTxt = ReplaceTable(txtSee,
                    [[ [["_NUM_", numTxt]],
                       [["_UPD_", updTxt]],
                       [["_PRN_", prnTxt]] ]]);
      Text regSep = If(EQ(numPic,maxPic), "", PdbSep);
      Text AppendFile(outFil, regTxt+regSep+"\n");
      TRUE
    });
    Card(wriSet)
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Not used now. This function creates an initial pictures databse.",
PdbInit);
////////////////////////////////////////////////////////////////////////////
```

pag.tol de Ink.Watercolor

Page functions.

Declaraciones

Funciones

- `Text PagPage(Real eng)`
  Returns the pages directory.
- `Text PagDir(Real eng)`
  Returns the pages full path.
- `Text PagSee(Real eng)`
  Returns the seeds directory.
- `Text PagSeePi1(Real eng)`
  Returns the page for zoom.
- `Text PagSeePi4(Real eng)`
  Returns the page for tinys.
- `Text PagSeeLst(Real eng)`
  Returns the page for list.
- `Text PagSeeInf(Real eng)`
  Returns the page for info.
- `Real PagIsBilingual(Text txt)`
  Returns true if the text is in english and spanish
- `Text PagLanguage(Real eng, Text txt)`
  If eng==true returns the english text, if eng=false the spanish text and if the text is not bilingual then return the text.
- `Text PagLongTitle(Real eng, Text tit)`
  Returns the long title, the full title without the () that are the tags of the sort title.
- `Text PagShortTitle(Real eng, Text tit)`
  Returns the long title, the title inside the () or [].
- `Text PagNumInt(Real picNum, Real zero)`
  Returns the number of a page in text format, if zero=true 0009 and if zero=false then 9.
- `Text PagNameInt(Text prefix, Real picNum)`
  Returns the name of a page giving a prefix and a picture number.
- `Text PagName(Text prefix, Set picReg)`
  Returns the name of a page giving a prefix and a picture register.
- `Real PagPicInLst(Real picNum)`
  Returns the number of list page (format of several pics per list page) in witch the picture picNum must be inserted.
- `Real PagPicInPag(Real picNum)`

Returns the number of page (format of several pics per page) in witch the picture picNum must be inserted.

- `Set PagPicFriends(Real ordReg, Set inpPdb)`
  Returns the set of pics that are in the same page.
- `Text PagLongTitFriends(Real eng, Real ordReg, Set inpPdb)`
  Returns a text title formed by union of the long titles of pics that are in the same page.
- `Text PagShortTitFriends(Real eng, Real ordReg, Set inpPdb)`
  Returns a text title formed by union of the short titles of pics that are in the same page.
- `Real PagFirstFriend(Real numPag, Set inpPdb)`
  Returns the number of the register that is the first (friend) a page of pictures.
- `Real PagFirstList(Real numPag, Set inpPdb)`
  Returns the number of the register that is the first of a list of pictures.
- `Text PagTechnique(Real eng, Set picReg, Real long)`
  Returns the pictore technique in a short or long way.
- `Set PagYearTab(Real eng, Set inpPdb, Text keySel)`
  Returns a replacement table for the years index.
- `Set PagYearMnu(Set picReg)`
  Returns a replacement tuble for a year of the years index.
- `Real PagCicle(Real eng, Set inpPdb, Text keySel, Set yeaRep, Text outDir, Text pthSee)`
  Creates and writes the html file for the cicle page of 4 pictures.
- `Real PagList(Real eng, Set inpPdb, Text keySel, Set yeaRep, Text outDir, Text pthSee)`
  Creates and writes the html file for the list page of 16 pictures.
- `Real PagSel(Real eng, Set inpPdb, Text keySel, Set yeaRep, Text pagSee)`
  Creates and writes html files for zoom, cicle and list pages.

Text PagPage()

```
////////////////////////////////////////////////////////////////////////////
Text PagPage(Real eng)
////////////////////////////////////////////////////////////////////////////
{ "pag"+If(eng, "e", "s") };
////////////////////////////////////////////////////////////////////////////
PutDescription("Returns the pages directory.", PagPage);
////////////////////////////////////////////////////////////////////////////
```

Text PagDir()

```
////////////////////////////////////////////////////////////////////////////
Text PagDir(Real eng)
////////////////////////////////////////////////////////////////////////////
{ "web/"+PagGal+"/"+PagPage(eng) };
////////////////////////////////////////////////////////////////////////////
PutDescription("Returns the pages full path.", PagDir);
////////////////////////////////////////////////////////////////////////////
```

Text PagSee()

```
////////////////////////////////////////////////////////////////////////
Text PagSee(Real eng)
////////////////////////////////////////////////////////////////////////
{ "web/"+PagGal+"/sed"+If(eng, "e", "s") };
////////////////////////////////////////////////////////////////////////
PutDescription("Returns the seeds directory.", PagSee);
////////////////////////////////////////////////////////////////////////
```

Text PagSeePi1()

```
////////////////////////////////////////////////////////////////////////
Text PagSeePi1(Real eng)
////////////////////////////////////////////////////////////////////////
{ "seedpi1"+If(eng, "eng", "spa")+".htm" };
////////////////////////////////////////////////////////////////////////
PutDescription("Returns the page for zoom.", PagSeePi1);
////////////////////////////////////////////////////////////////////////
```

Text PagSeePi4()

```
////////////////////////////////////////////////////////////////////////
Text PagSeePi4(Real eng)
////////////////////////////////////////////////////////////////////////
{ "seedpi4"+If(eng, "eng", "spa")+".htm" };
////////////////////////////////////////////////////////////////////////
PutDescription("Returns the page for tinys.", PagSeePi4);
////////////////////////////////////////////////////////////////////////
```

Text PagSeeLst()

```
////////////////////////////////////////////////////////////////////////
Text PagSeeLst(Real eng)
////////////////////////////////////////////////////////////////////////
{ "seedlst"+If(eng, "eng", "spa")+".htm" };
////////////////////////////////////////////////////////////////////////
PutDescription("Returns the page for list.", PagSeePi4);
////////////////////////////////////////////////////////////////////////
```

Text PagSeeInf()

```
////////////////////////////////////////////////////////////////////////
Text PagSeeInf(Real eng)
////////////////////////////////////////////////////////////////////////
{ "seedinf"+If(eng, "eng", "spa")+".htm" };
////////////////////////////////////////////////////////////////////////
PutDescription("Returns the page for info.", PagSeeInf);
////////////////////////////////////////////////////////////////////////
```

Real PagIsBilingual()

```
////////////////////////////////////////////////////////////////////////
Real PagIsBilingual(Text txt) // Text
////////////////////////////////////////////////////////////////////////
```

```
{ And(TextFind(txt,"{"), TextFind(txt,"|"), TextFind(txt,"}")) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns true if the text is in english and spanish",
PagIsBilingual);
////////////////////////////////////////////////////////////////////////////
```

Text PagLanguage()

```
////////////////////////////////////////////////////////////////////////////
Text PagLanguage(Real eng, // True if english
                 Text txt) // Text
////////////////////////////////////////////////////////////////////////////
{
  If(! PagIsBilingual(txt), txt,                // There aren't traduction
  If(eng, TxtBetween2Tag(txt, "{", "|", TRUE),  // English
          TxtBetween2Tag(txt, "|", "}", TRUE))) // Spanish
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"If eng==true returns the english text, if eng=false the spanish text and
if the text is not bilingual then return the text.",
PagLanguage);
////////////////////////////////////////////////////////////////////////////
```

Text PagLongTitle()

```
////////////////////////////////////////////////////////////////////////////
Text PagLongTitle(Real eng, // True if english
                  Text tit) // Tittle
////////////////////////////////////////////////////////////////////////////
{ ReplaceTable(PagLanguage(eng, tit), [[ [["(", ""]], [[")", ""]] ]]) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the long title, the full title without the () that are the tags of
the sort title.",
PagLongTitle);
////////////////////////////////////////////////////////////////////////////
```

Text PagShortTitle()

```
////////////////////////////////////////////////////////////////////////////
Text PagShortTitle(Real eng, // True if english
                   Text tit) // Tittle
////////////////////////////////////////////////////////////////////////////
{
  Text txt = PagLanguage(eng, tit);
  If(TextFind(txt, "("), TxtBetween2Tag(txt, "(", ")", TRUE), // ()
  If(TextFind(txt, "["), TxtBetween2Tag(txt, "[", "]", TRUE), // []
     txt))
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the long title, the title inside the ()  or [].",
PagShortTitle);
////////////////////////////////////////////////////////////////////////////
```

Text PagNumInt()

```
////////////////////////////////////////////////////////////////////////////
Text PagNumInt(Real picNum, Real zero)
////////////////////////////////////////////////////////////////////////////
```

```
{ If(zero, FormatReal(picNum, "%04.0lf"), FormatReal(picNum, "%.0lf")) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the number of a page in text format, if zero=true 0009 and if
zero=false then 9.",
PagNumInt);
////////////////////////////////////////////////////////////////////////////
```

Text PagNameInt()

```
////////////////////////////////////////////////////////////////////////////
Text PagNameInt(Text prefix, Real picNum)
////////////////////////////////////////////////////////////////////////////
{ prefix+PagNumInt(picNum, TRUE)+".htm" };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the name of a page giving a prefix and a picture number.",
PagNameInt);
////////////////////////////////////////////////////////////////////////////
```

Text PagName()

```
////////////////////////////////////////////////////////////////////////////
Text PagName(Text prefix, Set picReg)
////////////////////////////////////////////////////////////////////////////
{ PagNameInt(prefix, picReg->picNum) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the name of a page giving a prefix and a picture register.",
PagName);
////////////////////////////////////////////////////////////////////////////
```

Real PagPicInLst()

```
////////////////////////////////////////////////////////////////////////////
Real PagPicInLst(Real picNum)
////////////////////////////////////////////////////////////////////////////
{ If(EQ(picNum%PagLst, 0), picNum/PagLst, 1+Floor(picNum/PagLst)) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the number of list page (format of several pics per list page) in
witch the picture picNum must be inserted.",
PagPicInLst);
////////////////////////////////////////////////////////////////////////////
```

Real PagPicInPag()

```
////////////////////////////////////////////////////////////////////////////
Real PagPicInPag(Real picNum)
////////////////////////////////////////////////////////////////////////////
{ If(EQ(picNum%PagPic, 0), picNum/PagPic, 1+Floor(picNum/PagPic)) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the number of page (format of several pics per page) in witch the
picture picNum must be inserted.",
PagPicInPag);
////////////////////////////////////////////////////////////////////////////
```

Set PagPicFriends()

```
////////////////////////////////////////////////////////////////////////////
Set PagPicFriends(Real ordReg, // A pic order number
                  Set  inpPdb) // Pictures database
////////////////////////////////////////////////////////////////////////////
{
  Real numPag = PagPicInPag(ordReg); // Find the page
  Real maxReg = Card(inpPdb);        // Maximum of pics
  Set  picRep = For(1, PagPic, Set(Real picPos)
  {
    Real numReg = ((numPag-1)*PagPic)+picPos;
    If(GT(numReg,maxReg), Empty, inpPdb[numReg])
  });
  Select(picRep, Real(Set reg) { Card(reg) }) // Not empty regs
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the set of pics that are in the same page.",
PagPicFriends);
////////////////////////////////////////////////////////////////////////////
```

Text PagLongTitFriends()

```
////////////////////////////////////////////////////////////////////////////
Text PagLongTitFriends(Real eng,    // English/Spanish
                       Real ordReg, // A pic order number
                       Set  inpPdb) // Pictures database
////////////////////////////////////////////////////////////////////////////
{
  Set frdSet = PagPicFriends(ordReg, inpPdb);
  Set titSet = EvalSet(frdSet, Text(Set frdReg)
               { PagLongTitle(eng, frdReg->picTit) });
  Set2TxtCommaAmp(titSet)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a text title formed by union of the long titles of pics that are in
the same page.",
PagLongTitFriends);
////////////////////////////////////////////////////////////////////////////
```

Text PagShortTitFriends()

```
////////////////////////////////////////////////////////////////////////////
Text PagShortTitFriends(Real eng,    // English/Spanish
                        Real ordReg, // A pic order number
                        Set  inpPdb) // Pictures database
////////////////////////////////////////////////////////////////////////////
{
  Set frdSet = PagPicFriends(ordReg, inpPdb);
  Set titSet = EvalSet(frdSet, Text(Set frdReg)
               { PagShortTitle(eng, frdReg->picTit) });
  Set2TxtCommaAmp(titSet)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a text title formed by union of the short titles of pics that are in
the same page.",
PagShortTitFriends);
////////////////////////////////////////////////////////////////////////////
```

Real PagFirstFriend()

```
////////////////////////////////////////////////////////////////////////////
Real PagFirstFriend(Real numPag, // Page number
```

```
                  Set  inpPdb) // Pictures database
////////////////////////////////////////////////////////////////////////////
{ ((numPag-1)*PagPic)+1 };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the number of the register that is the first (friend) a page of
pictures.",
PagFirstFriend);
////////////////////////////////////////////////////////////////////////////
```

Real PagFirstList()

```
////////////////////////////////////////////////////////////////////////////
Real PagFirstList(Real numPag, // Page number
                  Set  inpPdb) // Pictures database
////////////////////////////////////////////////////////////////////////////
{ ((numPag-1)*PagLst)+1 };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the number of the register that is the first of a list of pictures.",
PagFirstList);
////////////////////////////////////////////////////////////////////////////
```

Text PagTechnique()

```
////////////////////////////////////////////////////////////////////////////
Text PagTechnique(Real eng,    // English/Spanish
                  Set  picReg, // A pic register
                  Real long)   // If True then long technique
////////////////////////////////////////////////////////////////////////////
{
  Text traFun(Text txt, Real eng, Real fem)
  {
    If(eng, txt,
    {
      Set  tab = [[
      [["diffusion of alcohol fire", "difusión de fuego de alcohol"]],
      [["rubbed with alcohol",       "frotado con alcohol"]],
      [["Fabriano paper",            "papel Fabriano"]],
      [["marker's ink",              "tinta de rotulador"]],
      [["couché paper",              "papel couché"]],
      [["mixed media",               "técnicas mixtas"]],
      [["golden brown",              "marrón "+If(fem, "dorada", "dorado")]],
      [["aerograph",                 "aerógrafo"]],
      [["watercolor",                "acuarela"]],
      [["cotton",                    "algodón"]],
      [["orange",                    "naranja"]],
      [["marker",                    "rotulador"]],
      [["violet",                    "violeta"]],
      [["white",                     If(fem, "blanca", "blanco")]],
      [["yellow",                    If(fem, "amarilla", "amarillo")]],
      [["green",                     "verde"]],
      [["brown",                     "marrón"]],
      [["black",                     If(fem, "negra", "negro")]],
      [["grey",                      "gris"]],
      [["light blue",                "azul claro"]],
      [["blue",                      "azul"]],
      [["cyan",                      "cian"]],
      [["pink",                      "rosa"]],
      [["red",                       If(fem, "roja", "rojo")]],
      [["ink",                       "tinta"]], // pink ?
      [["and",                       "y"]],
      [["&amp;",                     "y"]] ]];
      SetReplaceTable(txt, tab, 1)   // Recursive secuential replacement
```

```
    })
  };

  Text tecCol(Set tecSet, Set colSet, Real fem)
  {
    If(tecSet[1]=="", "",
    {
      Text colTxt = Set2TxtCommaAmp(colSet);
      Text tecTxt = Set2TxtCommaAmp(tecSet);
      Text tecCol =
        If(!long, tecTxt,
        If(eng, colTxt+" "+tecTxt, tecTxt+" "+colTxt));
      traFun(Compact(tecCol), eng, fem)
    })
  };

  Text forTxt = tecCol(picReg->forTec, picReg->forCol, TRUE); // Femenine
  Text bckTxt = tecCol(picReg->bckTec, picReg->bckCol, TRUE); // Femenine
  Text papTxt = If(! long, "",
                   tecCol(picReg->papTec, picReg->papCol, FALSE)+" "+ // Mascu
                   "("+FormatReal(picReg->picWid,"%.0lf")+"<i> </i>x<i> </i>"+
                   FormatReal(picReg->picHei,"%.0lf")+"<i> </i>mms)");

  Text sepF2b = If(eng, " over ", " sobre ");
  Text sepB2p = If(eng, " on ",   " en ");

  Text tecTxt = Case(
    And(forTxt!="", bckTxt!="", papTxt!=""), forTxt+sepF2b+bckTxt+sepB2p+papTx
    And(forTxt=="", bckTxt!="", papTxt!=""),               bckTxt+sepB2p+papTx
    And(forTxt!="", bckTxt=="", papTxt!=""), forTxt              +sepB2p+papTx
    And(forTxt!="", bckTxt!="", papTxt==""), forTxt+sepF2b+bckTxt
    TRUE /* only 1 of the 3 */             , forTxt       +bckTxt       +papTx

  tecTxt
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the pictore technique in a short or long way.",
PagTechnique);
////////////////////////////////////////////////////////////////////////////
```

Set PagYearTab()

```
////////////////////////////////////////////////////////////////////////////
Set PagYearTab(Real eng,
               Set  inpPdb,
               Text keySel)
////////////////////////////////////////////////////////////////////////////
{
  Text pthOld = "../../chppho/"+PagPage(eng)+"/";
  Text pthGal = "../../"+PagGal+"/"+PagPage(eng)+"/";

  Set  claPdb = Classify(inpPdb, Real(Set a, Set b)
                { Compare(Year(a->endDte),Year(b->endDte)) });
  Set  fstPdb = EvalSet(claPdb, Set(Set cla)
  { Sort(cla,Real(Set a, Set b){ Compare(a->endDte,b->endDte) })[1] });

  Set  yeaRep = For(1980,2010, Set(Real yeaNum)
  {
    Set  yeaSet = Select(fstPdb, Real(Set a) { EQ(Year(a->endDte),yeaNum) });
    Real crdSet = Card(yeaSet);
    Text yeaTxt = FormatReal(yeaNum,"%.0lf");
    Text yeaOld = "<a href='"+pthOld+"AN."+yeaTxt+"'>"+yeaTxt+"</a>";
    Set  repTwo =
       If(NE(crdSet,1),
         SetOfText(yeaOld, // Si no hay cuadros para el año no pone nada
           ""),
         SetOfText(yeaOld, // Si hay cuadros para el año cambia el link
           "<a href='"+pthGal+PagName(keySel+"zoo", yeaSet[1])+"'>"+yeaTxt+"</

    // Cambiar comillas simples por dobles como todo el mundo en internet
    SetOfText(repTwo[1], Replace(repTwo[2], "'", Char(34)))
```

```
  })
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a replacement table for the years index.",
PagYearTab);
////////////////////////////////////////////////////////////////////////
```

Set PagYearMnu()

```
////////////////////////////////////////////////////////////////////////
Set PagYearMnu(Set picReg)
////////////////////////////////////////////////////////////////////////
{
  Text dte = FormatDate(picReg->endDte,"%c%Y/%m/%d");
  Text yea = Sub(dte,1,4);
  SetOfText("<td class='navAMn'><!-- "+yea+" -->",    // Marcar el año
            "<td class='navOMn'><!-- "+yea+" -->")    // en curso
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a replacement tuble for a year of the years index.",
PagYearMnu);
////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////
Set PagTranslation(Real eng,    // Language english/spanish
                   Text htmPth) // Html file
////////////////////////////////////////////////////////////////////////
{
  Text Q      = Char(34);
  Text notTra =
    "<img src="+Q+"../../common/gif/block.gif"+Q+" width='100%' />";
  Text hlpTxt = " ("+If(eng,"translate to Spanish",
                            "traducir al inglés")+")";

  Text yesTra(Text url, Text tit)
  {
    "<a href="+Q+url+Q+">"+
    "<img src="+Q+"../../common/gif/translsta.gif"+Q+
    " width='100%'"+
    " name="+Q+"traBut"+Q+
    " border='0' "+
    " alt="+Q+tit+hlpTxt+Q+" "+
    " onmouseover="+Q+"document.traBut.src='../../common/gif/translact.gif';"+
    " onmouseout ="+Q+"document.traBut.src='../../common/gif/translsta.gif';"+
    " /></a>"
  };

  Text traPth = Replace(htmPth, PagPage( eng), PagPage(!eng));

  If(! FileExist(traPth), [[ "TRA.COD", notTra ]],
  {
    Text traHtm = ReadFile(traPth);
    Text traUrl = Replace(htmPth, PagDir(eng), "../"+PagPage(!eng));
    Text traTit = TxtBetween2Tag(traHtm, "<title>", "</title>", TRUE);
    [[ "TRA.COD", yesTra(traUrl, traTit) ]]
  })
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns the Html code that link each page wit it translation.",
PagTranslation);
////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////
Set PagGalleryLabel(Real eng,     // Language english/spanish
                    Text keySel)  // Selector
////////////////////////////////////////////////////////////////////////
```

```
{
  [[
    SetOfText("GAL.NAM",
              If(PagGal=="inarmy", If(eng, "ink in the army",
                                           "tinta en la mili"),
              If(PagGal=="fleurs", If(eng, "the flowers of evil",
                                           "las flores del mal"),
                                   If(eng, "ink and watercolor",
                                           "tinta y acuarela")))),
    SetOfText("GAL.KEY",
              If(keySel=="sad", If(eng, "[all]",         "[completo]"),
              If(keySel=="upd", If(eng, "[new]",         "[nuevo]"),
              If(keySel=="inx", If(eng, "[index]",       "[índice]"),
              If(keySel=="inf", If(eng, "[information]", "[información]"),
                                If(eng, "[error]",       "[error]"))))))
  ]]
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Returns a pair of labels for an art gallery.",
PagGalleryLabel);
////////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////////
Real PagZoom(Real eng,     // Language english/spanish
             Set  inpPdb,  // Pictures database
             Text keySel,  // Selector
             Set  yeaRep,  // Year map
             Text outDir,  // Output directory
             Text pthSee)  // Seed file
////////////////////////////////////////////////////////////////////////////
{
  Text WriteLn("Zoom  for: "+keySel);

  Text zooSee = pthSee+"/"+PagSeePi1(eng);
  Text zooHtm = ReadFile(zooSee);
  Text notNow = "../../common/gif/notnowzoo.gif";
  Real maxReg = Card(inpPdb);

  Text iniFil = PagName(keySel+"zoo", inpPdb[1]);
  Text iniTit = PagLongTitle(eng, inpPdb[1]->picTit);

  Text endFil = PagName(keySel+"zoo", inpPdb[maxReg]);
  Text endTit = PagLongTitle(eng, inpPdb[maxReg]->picTit);

  Set  cicSet = For(1, maxReg, Real(Real numReg)
  {
    Set  picReg = inpPdb[numReg];
    Text lngTit = PagLongTitle(eng, picReg->picTit);
    Text numTxt = PagNumInt(picReg->picNum, TRUE);  // 0099
    Text numTxB = PagNumInt(picReg->picNum, FALSE); // 99
    Text dteTxt = FormatDate(picReg->endDte,"%c%Y/%m/%d");
    Text yeaTxt = Sub(dteTxt,1, 4);
    Text dteTxB = Replace(dteTxt,"/", "<i> </i>/<i> </i>"); /* very little */
    Text traRem = PagLanguage(eng, picReg->picRem);

    Text zooFil = PagName   (keySel+"zoo", picReg);
    Text zooPth = outDir+"/"+zooFil;

    Text zooJpg = "web/"+PagGal+"/image/zoom/"+picReg->jpgFil;

    // Realmente sobra ya que si no hay imagen no crea ni la pagina
    Real zooExi = FileExist(zooJpg);

    Text picKey = Set2TxtKeyword([[lngTit, traRem]] + // Keywords
                                 picReg->bckTec   +
                                 picReg->forTec,
                                 PagMinChr, TRUE);

    Text nxtFil = PagName(keySel+"zoo", inpPdb[Min(maxReg,numReg+1)]);
    Text nxtTit = PagLongTitle(eng, inpPdb[Min(maxReg,numReg+1) ]->picTit);

//  http://www.baudelaire.cz/works.html?aID=100&artID=2 Frances
//  http://www.baudelaire.cz/works.html?aID=200&artID=2 Ingles
```

```
//  until 102 That kind heart you were jealous of... ok
//  but for me Mists and Rains is the 103 and baudelaire.cz jump to 137

    Text cz.bau = If(PagGal!="fleurs", "",
    {
      Text cz.txt = "<br /><br />" +
                    If(eng, "Full poem at ", "Poema completo en ");
      Text cz.lan = If(eng, "200", "100");
      Text lnk.cz = cz.txt+
                    "<a href='http://www.baudelaire.cz/works.html?aID="+
                    cz.lan+"&amp;artID="+numTxB+"'>baudelaire.cz</a>";
      lnk.cz
    });

    Set repTab =
    [[
      SetOfText("WIN.TIT", lngTit),
      SetOfText("MET.DES", lngTit+" ("+dteTxt+")"),
      SetOfText("MET.KEY", picKey),
      SetOfText("PAG.TIT", lngTit),

      SetOfText("JPG.FIL", If(zooExi, "../image/zoom/"+picReg->jpgFil,
                                      notNow)),

      SetOfText("URL.INI", iniFil),
      SetOfText("URL.B10", PagName   (keySel+"zoo", inpPdb[Max(1,     numReg-1(
      SetOfText("URL.PRE", PagName   (keySel+"zoo", inpPdb[Max(1,     numReg-1)
      SetOfText("URL.CIC", PagNameInt(keySel+"cic", PagPicInPag(numReg))),
      SetOfText("URL.NXT", nxtFil),
      SetOfText("URL.F10", PagName   (keySel+"zoo", inpPdb[Min(maxReg,numReg+1(
      SetOfText("URL.END", endFil),

      SetOfText("TIT.INI", iniTit),
      SetOfText("TIT.B10", PagLongTitle(eng, inpPdb[Max(1,     numReg-10)]->pi
      SetOfText("TIT.PRE", PagLongTitle(eng, inpPdb[Max(1,     numReg-1) ]->pi
      SetOfText("TIT.CIC", PagShortTitFriends(eng, numReg, inpPdb)),
      SetOfText("TIT.NXT", nxtTit),
      SetOfText("TIT.F10", PagLongTitle(eng, inpPdb[Min(maxReg,numReg+10)]->pi
      SetOfText("TIT.END", endTit),


// Forms page free images and budgets
      SetOfText("URL.FRE", "../../chppho/"+If(eng, "page", "pags")+
                           "/inffeeimg.htm?pn="+numTxt), // Free image
      SetOfText("URL.BDG", "../../chppho/"+If(eng, "page", "pags")+
                           "/inffeebdg.htm?pn="+numTxt), // Budgets
      SetOfText("TIT.FRE", If(eng, "free image of ",
                                   "imagen gratis de ")+lngTit),
      SetOfText("TIT.BDG", If(eng, "price for the original painting ",
                                   "precio para la obra original ")+lngTit),

      SetOfText("LOW.001",
      {
        lngTit                                      +", "+
        If(eng, "painted in", "pintado en")         +" "+
        dteTxB                                      +" "+
        If(keySel!="sad", "",
           If(eng, "with number", "con número")  +" "+
           numTxt)
      }),

      SetOfText("LOW.002", Text PagTechnique(eng, picReg, TRUE)),

      SetOfText("LOW.003",
      {
        If(eng, "painting viewed", "pintura vista")+" "+
        FormatReal(picReg->picPrn,"%.0lf")          +" "+
        If(eng, "times", "veces")
      }),

      SetOfText("LOW.004",
      {
        If(eng, "next", "siguiente")+": "+
        "<a href="+Char(34)+nxtFil+Char(34)+">"+
                              nxtTit+"</a>"
```

```
      }),

      SetOfText("LOW.LST", traRem+cz.bau),

      PagTranslation(eng, zooPth),

      PagYearMnu(picReg)

    ]]<<
      PagGalleryLabel(eng, keySel);

    Text filWri = WriteFile(zooPth, ReplaceTable(zooHtm, repTab<<yeaRep));
    TRUE
  });
  Card(cicSet)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Creates and writes the html file for the zoom page of a picture.",
PagZoom);
////////////////////////////////////////////////////////////////////////////
```

Real PagCicle()

```
////////////////////////////////////////////////////////////////////////////
Real PagCicle(Real eng,    // Language
              Set  inpPdb, // Picture database
              Text keySel, // Selector
              Set  yeaRep, // Year map
              Text outDir, // Output directory
              Text pthSee) // Seed path
////////////////////////////////////////////////////////////////////////////
{
  Text WriteLn("Cicle for: "+keySel);

  Text defNot = "../../common/gif/notnowtin.gif";

  Real maxReg = Card(inpPdb);
  Real maxPag = PagPicInPag(maxReg);

  Text cicSee = pthSee+"/"+PagSeePi4(eng);
  Text cicHtm = ReadFile(cicSee);

  Text iniFil = PagNameInt(keySel+"cic", 1);
  Text endFil = PagNameInt(keySel+"cic", maxPag);
  Text iniAlt = PagShortTitFriends(eng, 1,      inpPdb);
  Text endAlt = PagShortTitFriends(eng, maxReg, inpPdb);

  Set  cicSet = For(1, maxPag, Real(Real numPag)
  {
    // Movimientos de pagina adelante y atras
    Real b10Pag = Max(1,      numPag-10);
    Real prePag = Max(1,      numPag-1);
    Real nxtPag = Min(maxPag, numPag+1);
    Real f10Pag = Min(maxPag, numPag+10);

    Text b10Fil = PagNameInt(keySel+"cic", b10Pag);
    Text preFil = PagNameInt(keySel+"cic", prePag);
    Text nxtFil = PagNameInt(keySel+"cic", nxtPag);
    Text f10Fil = PagNameInt(keySel+"cic", f10Pag);

    // Años en curso a los que pertenecen las imagenes
    Real numFst = PagFirstFriend(numPag, inpPdb); // El primero marca año
    Set  picFst = inpPdb[numFst];                 // El registro
    Set  picLst = inpPdb[Min(maxReg,((numPag-1)*PagPic)+PagPic)]; // Ultimo
    Set  yeaMrk = If(EQ(Year(picFst->endDte),Year(picLst->endDte)),
    [[PagYearMnu(picFst)]],  // Marcar el año en curso
    [[PagYearMnu(picFst),    // Marcar el año ini en curso
      PagYearMnu(picLst)]]); // Marcar el año end en curso
```

```
    Set  picRep = For(1, PagPic, Set(Real picPos)
    {
      Real numReg = ((numPag-1)*PagPic)+picPos;
      Text pos3Tx = FormatReal(picPos,"%1.0lf");
      If(LE(numReg,maxReg),
      {

        Text jpgPth =
        If(FileExist("web/"+PagGal+"/image/tiny/"+inpPdb[numReg]->jpgFil),
           "../image/tiny/"+inpPdb[numReg]->jpgFil,
           defNot);

        [[
          SetOfText("ZOO."+pos3Tx, PagName(keySel+"zoo", inpPdb[numReg])),
          SetOfText("PIC."+pos3Tx, jpgPth),
          SetOfText("TIT."+pos3Tx, PagLongTitle(eng, inpPdb[numReg]->picTit)),
          SetOfText("TEC."+pos3Tx, "<b>"+PagTechnique(eng, inpPdb[numReg], FAL
        ]]
      },
      {
        [[
          SetOfText(Replace("<li><a href="+Char(34)+"ZOO._N_"+Char(34)+">TIT._
          SetOfText("ZOO."+pos3Tx, "#"), // The same page
          SetOfText("PIC."+pos3Tx, defNot),
          SetOfText("TIT."+pos3Tx, ""),
          SetOfText("TEC."+pos3Tx, "")

        ]]
      })
    });

//        SetOfText(Replace(defZoo,"_N_",pos3Tx), defNot),

    Text cicFil = PagNameInt(keySel+"cic", numPag);
    Text cicPth = outDir+"/"+cicFil;

    Text frdLng = PagLongTitFriends (eng, numFst, inpPdb);
    Text frdShr = PagShortTitFriends(eng, numFst, inpPdb);
    Text frdS80 = If(LE(TextLength(frdShr),PagTitMax), frdShr,
                  If(eng, "a free art gallery showcasing ink and watercolor pa
                          "una galería de arte libre con pinturas de tinta y a

    Set  keySet = EvalSet(PagPicFriends(numFst, inpPdb), Set(Set reg)
                  { reg->bckTec + reg->forTec });
    Text keyRep = Set2TxtKeyword(BinGroup("<<",keySet)+[[frdLng]], PagMinChr,

    Set repTab =
    [[
      SetOfText("WIN.TIT", frdShr),
      SetOfText("MET.DES", frdLng),
      SetOfText("MET.KEY", keyRep),
      SetOfText("PAG.TIT", frdS80),

      SetOfText("URL.INI", iniFil),
      SetOfText("URL.B10", b10Fil),
      SetOfText("URL.PRE", preFil),
      SetOfText("URL.LST", PagNameInt(keySel+"lst", PagPicInLst(numFst))),
      SetOfText("URL.NXT", nxtFil),
      SetOfText("URL.F10", f10Fil),
      SetOfText("URL.END", endFil),

      SetOfText("TIT.INI", iniAlt),
      SetOfText("TIT.B10", PagShortTitFriends(eng, PagFirstFriend(b10Pag, inpP
      SetOfText("TIT.PRE", PagShortTitFriends(eng, PagFirstFriend(prePag, inpP
      SetOfText("TIT.LST", If(eng, "gallery listing of paintings",
                                   "listado de cuadros de la galería")),
      SetOfText("TIT.NXT", PagShortTitFriends(eng, PagFirstFriend(nxtPag, inpP
      SetOfText("TIT.F10", PagShortTitFriends(eng, PagFirstFriend(f10Pag, inpP
      SetOfText("TIT.END", endAlt),

// Forms page free images and budgets (without any number)
      SetOfText("URL.FRE", "../../chppho/"+If(eng, "page", "pags")+
                           "/inffeeimg.htm"), // Free image
      SetOfText("URL.BDG", "../../chppho/"+If(eng, "page", "pags")+
```

```
                              "/inffeebdg.htm"), // Budgets
      SetOfText("TIT.FRE", If(eng, "ask for free images",      // Free image (
                                   "pedir imágenes gratis")),
      SetOfText("TIT.BDG", If(eng, "ask for the prices of the originals",
                                   "pedir precios para la obra original")),

      PagTranslation(eng, cicPth)
    ]] << BinGroup("<<",picRep)
       << yeaRep
       << yeaMrk
       << PagGalleryLabel(eng, keySel);

    Text filWri = WriteFile(cicPth, ReplaceTable(cicHtm, repTab));
    TRUE
  });
  Card(cicSet)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Creates and writes the html file for the cicle page of 4 pictures.",
PagCicle);
////////////////////////////////////////////////////////////////////////////
```

Real PagList()

```
////////////////////////////////////////////////////////////////////////////
Real PagList(Real eng,    // Language
             Set  inpPdb, // Picture database
             Text keySel, // Selector
             Set  yeaRep, // Year map
             Text outDir, // Output directory
             Text pthSee) // Seed path
////////////////////////////////////////////////////////////////////////////
{
  Text WriteLn("List  for: "+keySel);

  Real maxReg = Card(inpPdb);
  Real maxPag = PagPicInLst(maxReg);

  Text lstSee = pthSee+"/"+PagSeeLst(eng);
  Text lstHtm = ReadFile(lstSee);

  Text iniFil = PagNameInt(keySel+"lst", 1);
  Text endFil = PagNameInt(keySel+"lst", maxPag);

  Set  lstSet = For(1, maxPag, Real(Real numPag)
  {
    // Movimientos de pagina adelante y atras
    Real b10Pag = Max(1,      numPag-2);
    Real prePag = Max(1,      numPag-1);
    Real nxtPag = Min(maxPag, numPag+1);
    Real f10Pag = Min(maxPag, numPag+2);

    Text b10Fil = PagNameInt(keySel+"lst", b10Pag);
    Text preFil = PagNameInt(keySel+"lst", prePag);
    Text nxtFil = PagNameInt(keySel+"lst", nxtPag);
    Text f10Fil = PagNameInt(keySel+"lst", f10Pag);

    // Años en curso a los que pertenecen las imagenes
    Real numFst = PagFirstList(numPag, inpPdb);   // El primero marca año
    Set  picFst = inpPdb[numFst];                 // El registro
    Real numLst = Min(maxReg,((numPag-1)*PagLst)+PagLst); // Ultimo
    Set  picLst = inpPdb[numLst]; // Ultimo
    Set  yeaMrk = If(EQ(Year(picFst->endDte),Year(picLst->endDte)),
    [[PagYearMnu(picFst)]],  // Marcar el año en curso
    [[PagYearMnu(picFst),    // Marcar el año ini en curso
      PagYearMnu(picLst)]]); // Marcar el año end en curso

    Set  picRep = For(1, PagLst, Text(Real picPos)
    {
      Real numReg = ((numPag-1)*PagLst)+picPos;
      If(GT(numReg,maxReg), "",
```

```
    {
      Text url = PagName(keySel+"zoo", inpPdb[numReg]);
      Text tit = PagLongTitle(eng, inpPdb[numReg]->picTit);
      Text srt = PagShortTitle(eng, inpPdb[numReg]->picTit);
      Text lnk = "<a href="+Char(34)+url+Char(34)+">"+srt+"</a>";
      Text lin = "<li>"+Replace(tit, srt, lnk)+"</li>\n"
    })
  });


  Text lstFil = PagNameInt(keySel+"lst", numPag);
  Text lstPth = outDir+"/"+lstFil;

  Set  keyCic = For(1, PagLst, Text(Real picPos)
  {
    Real numReg = ((numPag-1)*PagLst)+picPos;
    If(GT(numReg,maxReg), "", PagLongTitle(eng, inpPdb[numReg]->picTit))
  });
  Text keyRep = Set2TxtKeyword(keyCic, 6, TRUE);  // Minimum 6 ordered
  Text keyTit = Set2TxtKeyword(keyCic, 7, FALSE); // Minimum 7 not ordered
  // Buscar la primera coma con blanco antes de la longitud maxima del titul
  Text pagTit = Sub(keyTit, 1, TextFind(keyTit, ", ", PagTitMax, 1, -1))+"..

  Set repTab =
  [[
    SetOfText("WIN.TIT", pagTit),
    SetOfText("MET.DES", pagTit),
    SetOfText("MET.KEY", keyRep),
    SetOfText("PAG.TIT", pagTit),


    SetOfText("URL.INI", iniFil),
    SetOfText("URL.B10", b10Fil),
    SetOfText("URL.PRE", preFil),
    SetOfText("ZOO.1",   PagName(keySel+"zoo", picFst)),
    SetOfText("URL.NXT", nxtFil),
    SetOfText("URL.F10", f10Fil),
    SetOfText("URL.END", endFil),

    SetOfText("TIT.INI", If(eng, "begin of gallery listing",
                                 "inicio del listado de la galería")),
    SetOfText("TIT.B10", "-2"),
    SetOfText("TIT.PRE", "previous"),
    SetOfText("TIT.ZOM", PagLongTitle(eng, picFst->picTit)),
    SetOfText("TIT.NXT", "next"),
    SetOfText("TIT.F10", "+2"),
    SetOfText("TIT.END", If(eng, "end of gallery listing",
                                 "final del listado de la galería")),

// Forms page free images and budgets (without any number)
    SetOfText("URL.FRE", "../../chppho/"+If(eng, "page", "pags")+
                         "/inffeeimg.htm"), // Free image
    SetOfText("URL.BDG", "../../chppho/"+If(eng, "page", "pags")+
                         "/inffeebdg.htm"), // Budgets
    SetOfText("TIT.FRE", If(eng, "ask for free images",      // Free image (
                                 "pedir imágenes gratis")),
    SetOfText("TIT.BDG", If(eng, "ask for the prices of the originals",
                                 "pedir precios para la obra original")),

//    SetOfText("PIC.MIN", FormatReal(numFst,"%.0lf")),
//    SetOfText("PIC.MAX", FormatReal(numLst,"%.0lf")),

    SetOfText("PIC.LST", BinGroup("+",picRep)),

    PagTranslation(eng, lstPth)
  ]] << yeaRep
     << yeaMrk
     << PagGalleryLabel(eng, keySel);

  Text filWri = WriteFile(lstPth, ReplaceTable(lstHtm, repTab));
  TRUE
 });
 Card(lstSet)
};
////////////////////////////////////////////////////////////////////////////
```

```
PutDescription(
"Creates and writes the html file for the list page of 16 pictures.",
PagCicle);
////////////////////////////////////////////////////////////////////////////
```

Real PagSel()

```
////////////////////////////////////////////////////////////////////////////
Real PagSel(Real eng,
            Set  inpPdb,
            Text keySel,
            Set  yeaRep,
            Text pagSee)
////////////////////////////////////////////////////////////////////////////
{
  Text WriteLn(PdbSep+"\nbuilding "+If(eng,"english","spanish")+" pages...");

  Real zoo = PagZoom (eng, inpPdb, keySel, yeaRep, PagDir(eng), pagSee);
  Real cic = PagCicle(eng, inpPdb, keySel, yeaRep, PagDir(eng), pagSee);
  Real lst = PagList (eng, inpPdb, keySel, yeaRep, PagDir(eng), pagSee);

  zoo+cic+lst
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Creates and writes html files for zoom, cicle and list pages.",
PagSel);
////////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////////
Real PagJavascriptDB(Real eng,     // Language english/spanish
                     Set  inpPdb,  // Pictures database
                     Text outDir,  // Output directory
                     Text pthSee)  // Seed file
////////////////////////////////////////////////////////////////////////////
{
  Text WriteLn("PagJavascriptDB: "+PagGal+" -> ");

  Text budJsc = ReadFile(pthSee);     // Javascript seed
  Text budPth = outDir+"/"+PagGal+If(eng,"dbenglish",        // Big DB eng
                                         "dbspanish")+".js"; // Big DB spa
  Text tudPth = outDir+"/"+PagGal+If(eng,"dtenglish",        // Tiny DB eng
                                         "dtspanish")+".js"; // Tiny DB spa
  Text WriteLn(" ->"+budPth);
  Text WriteLn(" ->"+tudPth);

  Text fstLin = "// Inkwatercolor pictures javascript database "+
                "PdbLan = "+If(eng, "'english'", "'spanish'")+";";
  Text budWri = WriteFile(budPth, fstLin+" (big)\n");
  Text tudWri = WriteFile(tudPth, fstLin+" (tiny)\n");

  Real maxReg = Card(inpPdb);

  Set  cicSet = For(1, maxReg, Real(Real numReg)
  {
    Set  picReg = inpPdb[numReg];

    Text numTxt = PagNumInt(picReg->picNum, TRUE);
    Text lngTit = PagLongTitle(eng, picReg->picTit);
    Text shrTit = PagShortTitle(eng, picReg->picTit);
    Text dteTxt = FormatDate(picReg->endDte,"%c%Y/%m/%d");
    Text dteFmt = If(eng, "painted in", "pintado en") +" "+dteTxt+" "+
                  If(eng, "with number", "con número")+" "+numTxt;
    Text picJpg = PagGal+"/image/tiny/"+picReg->jpgFil;
    Text lngTec = ReplaceTable(
                    PagTechnique(eng, picReg, TRUE); // Long technique
                    [[ [["&amp;", "&"]], [["<i> </i>", " "]] ]]); // No html
    Text shrTec = ReplaceTable(
                    PagTechnique(eng, picReg, FALSE); // Short technique
                    [[ [["&amp;", "&"]], [["<i> </i>", " "]] ]]); // No html
```

```
    Real priBas = 100 *
                  If(IsUnknown(picReg->wrkTim), 0, picReg->wrkTim) *
                  If(IsUnknown(picReg->wrkQly), 0, picReg->wrkQly);
    Text priTxt = FormatReal(priBas, "%.0lf");

    Set repTab =
    [[
      SetOfText("PIC.NUM", numTxt),
      SetOfText("PIC.DTE", dteFmt),
      SetOfText("PIC.JPG", picJpg),
      SetOfText("PRI.BAS", priTxt)
    ]];

    Set repBig =
    [[
      SetOfText("PIC.TIT", lngTit),
      SetOfText("PIC.TEC", lngTec)
    ]] << repTab;

    Set repTin =
    [[
      SetOfText("PIC.TIT", shrTit),
      SetOfText("PIC.TEC", shrTec)
    ]] << repTab;

    Text budApp = AppendFile(budPth, ReplaceTable(budJsc, repBig));
    Text tudApp = AppendFile(tudPth, ReplaceTable(budJsc, repTin));
    TRUE
  });
  Card(cicSet)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Creates and writes a Javascript database of the pictures.",
PagSel);
////////////////////////////////////////////////////////////////////////////
```

sed.tol de Ink.Watercolor

Funcion para realizar semillas.

Declaraciones

Funciones

- `Real SedMake(Real eng)`

  Creates and writes all the seed files (templates) for an art gallery.

Real SedMake()

```
////////////////////////////////////////////////////////////////////////////
Real SedMake(Real eng)       // If true then english else spanish
////////////////////////////////////////////////////////////////////////////
{
  Text WriteLn(PdbSep+"\nbuilding "+If(eng,"english","spanish")+" seeds...");
  Text inpFil = "agenda/"+PagGal+"db01.txt";
  Set  inpBst = IncludeBST("ctr/img."+PagGal+".bst");
  Set  inpPdb = PdbRead(inpFil, inpBst);

  Set yeaAllTab =
  {
    Set  selSad = Select(inpPdb, Real(Set picReg)
                  { FileExist("web/"+PagGal+"/image/zoom/"+picReg->jpgFil) });
    PagYearTab(eng, selSad, "sad")
  };

  Text sedPth    = "web/common/seed/";
  Text sedStr    = ReadFile(sedPth+"strseed.htm"); // Structure
  Text sedPub    = ReadFile(sedPth+"pubseed.htm"); // Publicity

  Text sedButStr = Replace(
                     ReadFile(sedPth+"butstrseed.htm"), // Buttons structure
                     "URL.HOM", "http://www.inkwatercolor.com/"+
                                If(eng, "index.html",
                                        "indice.html"));
  Text sedButOt  = ReadFile(sedPth+"butzotseed.htm");   // Button zoom out
  Text sedButIn  = ReadFile(sedPth+"butzinseed.htm");   // Button zoom in
  Text sedButLs  = ReadFile(sedPth+"butlstseed.htm");   // Button list

  Text sedButZNo = Replace(sedButStr, "ZOM.COD", "");       // Buttoms no zoom
  Text sedButZOt = Replace(sedButStr, "ZOM.COD", sedButOt); // Buttoms zoom ou
  Text sedButZIn = Replace(sedButStr, "ZOM.COD", sedButIn); // Buttoms zoom in
  Text sedButLst = Replace(sedButStr, "ZOM.COD", sedButLs); // Buttoms list

  Text sedTxtCnt = ReadFile(sedPth+"txtcntseed.htm"); // Text contens
  Text sedTxtLow = ReadFile(sedPth+"txtlowseed.htm"); // Text low

  Text sedLstCnt = ReadFile(sedPth+"lstcntseed.htm"); // List of pictures
  Text sedLstLow = ReadFile(sedPth+"lstlowseed.htm"); // List low

  Text sedPi4Cnt = ReadFile(sedPth+"pi4cntseed.htm"); // Four pictures
  Text sedPi4Low = ReadFile(sedPth+"pi4lowseed.htm"); // Four pictures low are

  Text sedPi1Cnt = ReadFile(sedPth+"pi1cntseed.htm"); // One picture
  Text sedPi1Low = ReadFile(sedPth+"pi1lowseed.htm"); // One picture low area

//Text sedInxCnt = ReadFile(sedPth+"inxcntseed.htm");     // Home page
  Text sedInxCnt = ReadFile(sedPth+"inxcntrandseed.htm"); // Home page Random
  Text sedInxLow = ReadFile(sedPth+"inxlowseed.htm");     // Home page low are
```

```
Text sedMapYea =
{
  Text sed = ReadFile(sedPth+"mapyeaseed.htm"); // Years table
  If(eng, sed, ReplaceTable(sed, [[ [["by years",   "por años"     ]],
                                    [[PagPage(TRUE), PagPage(FALSE)]] ]]))
};
Text sedMapNot = ReadFile(sedPth+"mapnotseed.htm");


Set  repTab =
[[
  [[ "MET.AUT", If(eng, "ink and watercolor painter", // author
                        "pintor con tinta y acuarela") ]],

  [[ "MET.GEN", If(eng, "gallery of paintings builder",  // generator
                        "constructor de la galeria de pinturas") ]],

  [[ "MET.LAN", If(eng, "english", "spanish") ]],        // DC.language

  [[ "PAG.PUB", sedPub                        ]],        // Publicity

  [[ "ALT.HOM", If(eng, "ink and watercolor home page",
                        "página principal de tinta y acuarela") ]],

  [[ "ALT.SEA", If(eng, "search in inkwatercolor.com",
                        "buscar en inkwatercolor.com") ]],

  [[ "URL.MAI", If(eng, "../../chppho/page/inffeebck.htm",
                        "../../chppho/pags/inffeebck.htm") ]],

  [[ "ALT.MAI", If(eng, "send us your questions, suggestions and messages",
                        "envienos sus preguntas, sugerencias y mensajes") ]]

  [[ "URL.HLP", If(eng, "../../chppho/page/infhelp.htm",
                        "../../chppho/pags/infhelp.htm") ]],

  [[ "ALT.HLP", If(eng, "help", "ayuda") ]],

  [[ "PAG.GAL", PagGal ]],
  SetOfText("PIC.MIN", FormatReal(1,           "%.0lf")),
  SetOfText("PIC.MAX", FormatReal(Card(inpPdb),"%.0lf")),

  [[ "MNU.URL.001", If(eng, "../../chppho/page/inffreuse.htm",
                            "../../chppho/pags/inffreuse.htm") ]],
  [[ "MNU.LBL.001", If(eng, "free images", "imágenes gratis") ]],

  [[ "MNU.URL.002", If(eng, "../../chppho/page/inffeebck.htm",
                            "../../chppho/pags/inffeebck.htm") ]],
  [[ "MNU.LBL.002", If(eng, "author contact", "contactar autor") ]],

  [[ "MNU.URL.003", If(eng, "../../chppho/page/infdwnloa.htm",
                            "../../chppho/pags/infdwnloa.htm") ]],
  [[ "MNU.LBL.003", If(eng, "download", "descargas") ]],

  [[ "MNU.URL.004", "http://inkwatercolor.master.com/texis/master/search/?"]]
  [[ "MNU.LBL.004", If(eng, "search this web", "buscar aquí") ]],

  [[ "MNU.URL.005", If(eng, "../../chppho/page/infartlnk.htm",
                            "../../chppho/pags/infartlnk.htm") ]],
  [[ "MNU.LBL.005", If(eng, "art links", "enlaces de arte") ]],

  [[ "MNU.URL.006", If(eng, "../../chppho/page/infstatis.htm",
                            "../../chppho/pags/infstatis.htm") ]],
  [[ "MNU.LBL.006", If(eng, "statistics", "estadísticas") ]],

  [[ "MNU.URL.007", If(eng, "../../chppho/page/infsitmap.htm",
                            "../../chppho/pags/infsitmap.htm") ]],
  [[ "MNU.LBL.007", If(eng, "site map", "mapa del web") ]],

  [[ "MNU.URL.008", If(eng, "../../chppho/page/sadcic0001.htm",
                            "../../chppho/pags/sadcic0001.htm") ]],
  [[ "MNU.LBL.008", If(eng, "ink watercolor", "tinta acuarela")]],

  [[ "MNU.URL.009", If(eng, "../../fleurs/page/sadcic0001.htm",
```

```
                              "../../fleurs/pags/sadcic0001.htm") ]],
    [[ "MNU.LBL.009", If(eng, "flowers of evil", "flores del mal")]],

    [[ "MNU.URL.010", If(eng, "../../inarmy/page/sadcic0001.htm",
                              "../../inarmy/pags/sadcic0001.htm") ]],
    [[ "MNU.LBL.010", If(eng, "ink in the army", "tinta en la mili")]],

    [[ "MNU.URL.011", If(eng, "../../"+PagGal+"/page/updcic0001.htm",
                              "../../"+PagGal+"/pags/updcic0001.htm") ]],
    [[ "MNU.LBL.011", If(eng, "new paintings", "nuevos cuadros")]],

    // Delete empty menus entries
    [[ "<li><a href="+Char(34)+"MNU.URL.012"+Char(34)+
                              ">MNU.LBL.012</a></li>", "" ]],
    [[ "<li><a href="+Char(34)+"MNU.URL.013"+Char(34)+
                              ">MNU.LBL.013</a></li>", "" ]],
    [[ "<li><a href="+Char(34)+"MNU.URL.014"+Char(34)
                            +">MNU.LBL.014</a></li>", "" ]],

    [[ "<a href="+Char(34)+Char(34)+"></a>", "" ]]

  ]];


  Text pagInf = ReplaceTable(sedStr, repTab <<
  [[
    [["WIN.TIT", "_TIT_"]],
    [["MET.DES", "_TIT_"]],
    [["PAG.TIT", "_TIT_"]],
    [["PAG.BUT", If(eng, "<h2>ink and watercolor</h2>",
                         "<h2>tinta y acuarela</h2>")]],
    [["PAG.CNT", sedTxtCnt]],
    [["PAG.LOW", sedTxtLow]],
    [["PAG.MAP", sedMapYea]] // Page map
  ]]<<PagGalleryLabel(eng, "inf"));
  Text WriteFile(PagSee(eng)+"/"+PagSeeInf(eng), pagInf);

//  [["PAG.MAP", If(PagGal=="chppho", sedMapYea, sedMapNot) ]] If no map is wa

  Text pagLst = ReplaceTable(sedStr, repTab <<
  [[
    [["PAG.BUT", sedButZIn]],
    [["PAG.CNT", sedLstCnt]],
    [["PAG.LOW", sedLstLow]],
    [["PAG.MAP", sedMapYea ]] // Page map
  ]]);
  Text WriteFile(PagSee(eng)+"/"+PagSeeLst(eng), pagLst);

  Text pagPi4 = ReplaceTable(sedStr, repTab <<
  [[
    [["PAG.BUT", sedButLst]],
    [["PAG.CNT", sedPi4Cnt]],
    [["PAG.LOW", sedPi4Low]],
    [["PAG.MAP", sedMapYea ]] // Page map
  ]]);
  Text WriteFile(PagSee(eng)+"/"+PagSeePi4(eng), pagPi4);

  Text pagPi1 = ReplaceTable(sedStr, repTab <<
  [[
    [["PAG.BUT", sedButZOt]],
    [["PAG.CNT", sedPi1Cnt]],
    [["PAG.LOW", sedPi1Low]],
    [["PAG.MAP", sedMapYea ]] // Page map
  ]]);
  Text WriteFile(PagSee(eng)+"/"+PagSeePi1(eng), pagPi1);

  Text pagInx = ReplaceTable(sedStr, repTab << yeaAllTab <<
  [[
    [["WIN.TIT", If(eng, "ink watercolor: free art gallery",
                         "tinta acuarela: galería de arte libre")]],
    [["PAG.TIT", If(eng, "a free art gallery showcasing ink and watercolor pai
                         "una galería de arte libre con pinturas de tinta y ac
    [["MET.DES", If(eng, "a free art gallery showcasing ink and watercolor pai
                         "una galería de arte libre con pinturas de tinta y ac
    [["MET.KEY", If(eng, "ink, inks, watercolor, watercolors, watercolour, wat
```

```
                        "tinta, tintas, acuarela, acuarelas")]],
    [["PAG.BUT", "<h2>inkwatercolor.com</h2>"]],
    [["PAG.CNT", sedInxCnt]],
    [["PAG.LOW", sedInxLow]],
    [["LOW.PRO", If(eng, "the ink watercolor project is a free art gallery sho
                        "watercolor paintings, for private and non commercial
                        "el proyecto Ink Watercolor es una galería de arte li
                        "tinta y acuarela disponeble y abierta a todo el mund
    [["LOW.FRE", If(eng, "feel free to use any of the drawings on the site if
                        "private individual and your use is not commercial.",
                        "siéntete libre para utilizar todas la obras de este
                        "que sea para tu uso privado y no para uso comercial.
    [["LOW.INK", If(eng, "ink watercolor gallery listing",
                        "galería de tintas y acuarelas")]],
    [["LOW.FLE", If(eng, "flowers of evil gallery listing",
                        "galería de las flores del mal")]],

    [["PAG.MAP", sedMapYea]] // Page map

  ]]<<PagGalleryLabel(eng, "inx"));
  Text WriteFile(PagSee(eng)+"/index.htm", pagInx);

// Seed for feedback forms
  Text sedFrm = ReadFile(sedPth+"feefrm.htm"); // Common feedback forms
  Text htmFrm = ReplaceTable(sedFrm, repTab);
  Text WriteFile(PagSee(eng)+"/feefrm.htm", htmFrm);

  TRUE
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Creates and writes all the seed files (templates) for an art gallery.",
SedMake);
////////////////////////////////////////////////////////////////////////////
```

ftp.tol de Ink.Watercolor

Funciones para realizar ftp.

Declaraciones

xml.tol de Ink.Watercolor

Declaraciones

alc.tol de Ink.Watercolor

Alchemy functions Image Alchemy is Copyright of Handmade Software, Inc.

Declaraciones

Funciones

- `Real AlcImg2JpgDpi(Text inpPth, Text outPth, Real width, Real height, Real dpi)`

  Convert any image file to a jpeg file of width x heigth pixels and dpi dots per inch. If dpi is 0 doesn't use the dots per inch options.

Constantes

```
Text AlcPth = "bin/alchlong.exe"; // Executable path
```

Real AlcImg2JpgDpi()

```
////////////////////////////////////////////////////////////////////////////
Real AlcImg2JpgDpi(Text inpPth, // Input file path
                   Text outPth, // Output file path
                   Real width,  // Width
                   Real height, // Height
                   Real dpi)    // Dots per inch
////////////////////////////////////////////////////////////////////////////
{
  Text alcDos = Replace(AlcPth,"/","\\")+" ";
  Text inpDos = Replace(inpPth,"/","\\");
  Text outDos = Replace(outPth,"/","\\");
  Text widTxt = FormatReal(width,  "%.0lf");
  Text heiTxt = FormatReal(height, "%.0lf");
  Text widDpi = If(LE(dpi,0), "", "--X"+FormatReal(width /dpi,"%.3lf")+"i ");
  Text heiDpi = If(LE(dpi,0), "", "--Y"+FormatReal(height/dpi,"%.3lf")+"i ");

  Real sysExe = System(alcDos+                  // Executable
                       "-o -j "+                // Options
                       "-Xb"+widTxt+" "+        //  - width
                       "-Yb"+heiTxt+" "+        //  - height
                       widDpi+heiDpi+           //  - dots per inch
                       inpDos+" "+              // Input file path
                       outDos);                 // Output file path
```

```
  sysExe
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Convert any image file to a jpeg file of width x heigth pixels and dpi dots
per inch.
If dpi is 0 doesn't use the dots per inch options.",
AlcImg2JpgDpi);
////////////////////////////////////////////////////////////////////////////
```

ink.tol de Ink.Watercolor

Inkwatercolor functions

Declaraciones

inc.tol de Ink.Watercolor

Inclusion of common functions and application functions

Declaraciones

Inclusiones comunes

- `Set txtInc`
  Text functions.
- `Set setInc`
  Set functions.
- `Set tabInc`
  Table, set of sets, functions.
- `Set serInc`
  Time series functions.
- `Set filInc`
  File functions.
- `Set zipInc`
  WinZip command line functions.
- `Set apaInc`
  Apache log file functions.
- `Set dirInc`
  Directory functions.

Inclusiones de aplicación

- `Set pdbInc`
  Pictures database funtions.
- `Set pagInc`
  Page functions, to generate Html pages.
- `Set sedInc`
  Seed functions, to generate Html seed pages.
- `Set ftpInc`
  Ftp functions.
- `Set xsmInc`

Xsm site maps functions.

- Set alcInc

  Alchemy command line functions for images.

- Set inkInc

  Inkwatercolor special functions.

Set txtInc

```
////////////////////////////////////////////////////////////////////////////
Set  txtInc = Include("cmm/txt.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Text functions.", txtInc);
////////////////////////////////////////////////////////////////////////////
```

Set setInc

```
////////////////////////////////////////////////////////////////////////////
Set  setInc = Include("cmm/set.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Set functions.", setInc);
////////////////////////////////////////////////////////////////////////////
```

Set tabInc

```
////////////////////////////////////////////////////////////////////////////
Set  tabInc = Include("cmm/tab.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Table, set of sets, functions.", tabInc);
////////////////////////////////////////////////////////////////////////////
```

Set serInc

```
////////////////////////////////////////////////////////////////////////////
Set  serInc = Include("cmm/ser.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Time series functions.", serInc);
////////////////////////////////////////////////////////////////////////////
```

Set filInc

```
////////////////////////////////////////////////////////////////////////////
Set  filInc = Include("cmm/fil.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("File functions.", filInc);
////////////////////////////////////////////////////////////////////////////
```

Set zipInc

```
////////////////////////////////////////////////////////////////////////////
Set  zipInc = Include("cmm/zip.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("WinZip command line functions.", zipInc);
```

```
////////////////////////////////////////////////////////////////////////////
```

Set apaInc

```
////////////////////////////////////////////////////////////////////////////
Set  apaInc = Include("cmm/apa.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Apache log file functions.", apaInc);
////////////////////////////////////////////////////////////////////////////
```

Set dirInc

```
////////////////////////////////////////////////////////////////////////////
Set  dirInc = Include("cmm/dir.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Directory functions.", dirInc);
////////////////////////////////////////////////////////////////////////////
```

Set pdbInc

```
////////////////////////////////////////////////////////////////////////////
Set  pdbInc = Include("app/pdb.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Pictures database funtions.", pdbInc);
////////////////////////////////////////////////////////////////////////////
```

Set pagInc

```
////////////////////////////////////////////////////////////////////////////
Set  pagInc = Include("app/pag.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Page functions, to generate Html pages.", pagInc);
////////////////////////////////////////////////////////////////////////////
```

Set sedInc

```
////////////////////////////////////////////////////////////////////////////
Set  sedInc = Include("app/sed.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Seed functions, to generate Html seed pages.", sedInc);
////////////////////////////////////////////////////////////////////////////
```

Set ftpInc

```
////////////////////////////////////////////////////////////////////////////
Set  ftpInc = Include("app/ftp.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Ftp functions.", ftpInc);
////////////////////////////////////////////////////////////////////////////
```

Set xsmInc

```
////////////////////////////////////////////////////////////////////////////
Set  xsmInc = Include("app/xsm.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Xsm site maps functions.", xsmInc);
////////////////////////////////////////////////////////////////////////////
```

Set alcInc

```
////////////////////////////////////////////////////////////////////////////
Set  alcInc = Include("app/alc.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Alchemy command line functions for images.", alcInc);
////////////////////////////////////////////////////////////////////////////
```

Set inkInc

```
////////////////////////////////////////////////////////////////////////////
Set  inkInc = Include("app/ink.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Inkwatercolor special functions.", inkInc);
////////////////////////////////////////////////////////////////////////////
```