

# LDraw Script Compiler (LDS) et LDS Shell

Mise à jour de la page : 29 octobre 2011.



[J.C. Tchang](#)

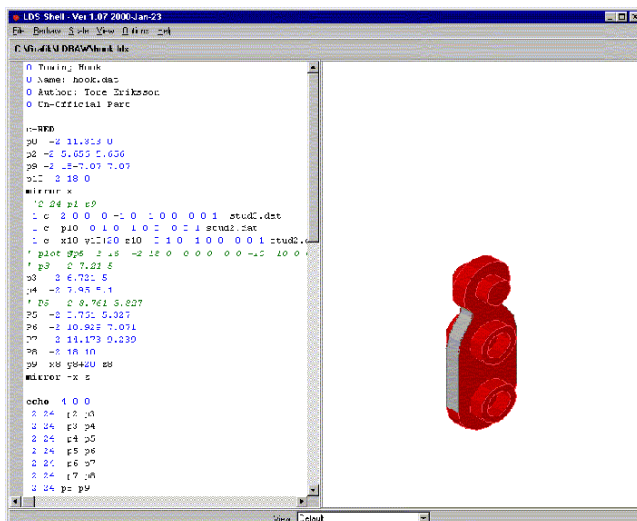
**LDraw Script Compiler** ou **LDS** est un programme permettant de générer des fichiers au format LDraw à partir d'un script de définition.

Imaginez que vous voulez faire un plaque de base classique 50x50. Cela nécessite 2500 tenons (studs), soit 2500 lignes de commande LDraw !

Inspiré du Basic, j'ai fait (*dixit Tore Eriksson*) un petit compilateur de script, qui peut manipuler de simples variables, des boucles FOR - TO - NEXT, et autres commandes utiles. Comme la plupart des pièces Lego sont symétriques, j'ai également ajouté quelques fonctions miroir (MIRROR).

C'est un programme fonctionnant sous DOS, ou dans une fenêtre "Invite de Commande" de Windows.

**LDS Shell** est une interface graphique sous Windows, facilitant l'utilisation de **LDS**.



## Navigation rapide

- [Téléchargement](#)
- [Installation](#)
- [Usage des programmes](#)
- [Menus de LDS Shell](#)
- [Visualisation avec LDView](#)
- [Langage Script de LDS](#)
- [Exemples de l'auteur du programme](#)
- [Exemples de J.C. Tchang](#)
- [Droits et copyright](#)

## Téléchargement

Pour utiliser LDS, il vous faut :

- Le programme DOS, dernière version : v0.64 du 28 octobre 2011.  
[Téléchargement LDS v0.64, 36 ko .RAR file.](#)
- La syntaxe du langage script de LDS :  
[LDraw Script Syntax Page](#) (trié par instructions et en anglais), ou aller plus bas dans ce manuel en français.
- Le programme servant d'interface graphique de Anders Isaksson :  
[Anders Isaksson's LDS Shell](#) (Semble plus disponible).

Vous pouvez aussi **télécharger sur ce site** les programmes et exemples dans un seul fichier :

- [lds064\\_complete.zip](#) (2.25 Mo).

Nota : La version actuelle (LDS v0.64) est compatible avec **Windows 7 - 64 bits**.

## Installation

Décompressez le fichier .ZIP ou .RAR avec votre utilitaire de compression préféré reconnaissant le format (par exemple Winrar, 7-zip, ...).

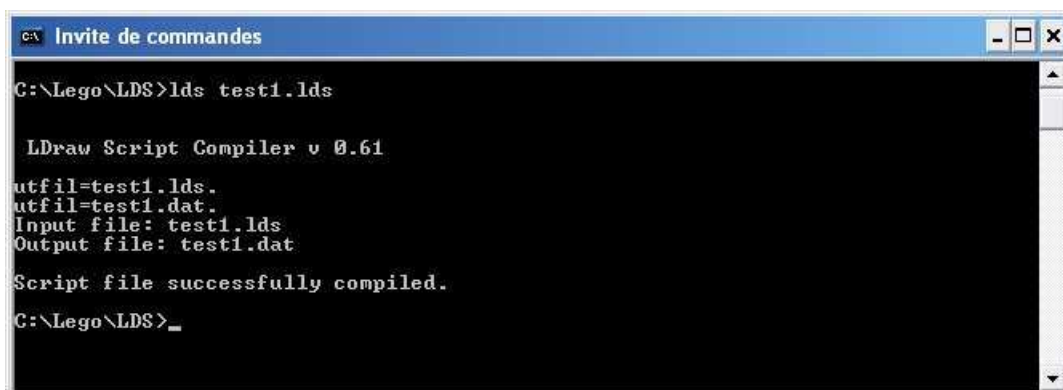
Puis mettre tous les fichiers dans un seul dossier de votre choix. Par exemple : [c:\lego\lds](#).

## Usage des programmes

### Usage de LDS

Dans une fenêtre "Invite de commandes", et dans le dossier contenant le programme et le fichier script, lancez la commande :

```
LDS[.exe] Infile [Outfile]
```



```
C:\Lego\LDS>lds test1.lds

LDraw Script Compiler v 0.61

utfil=test1.lds.
utfil=test1.dat.
Input file: test1.lds
Output file: test1.dat

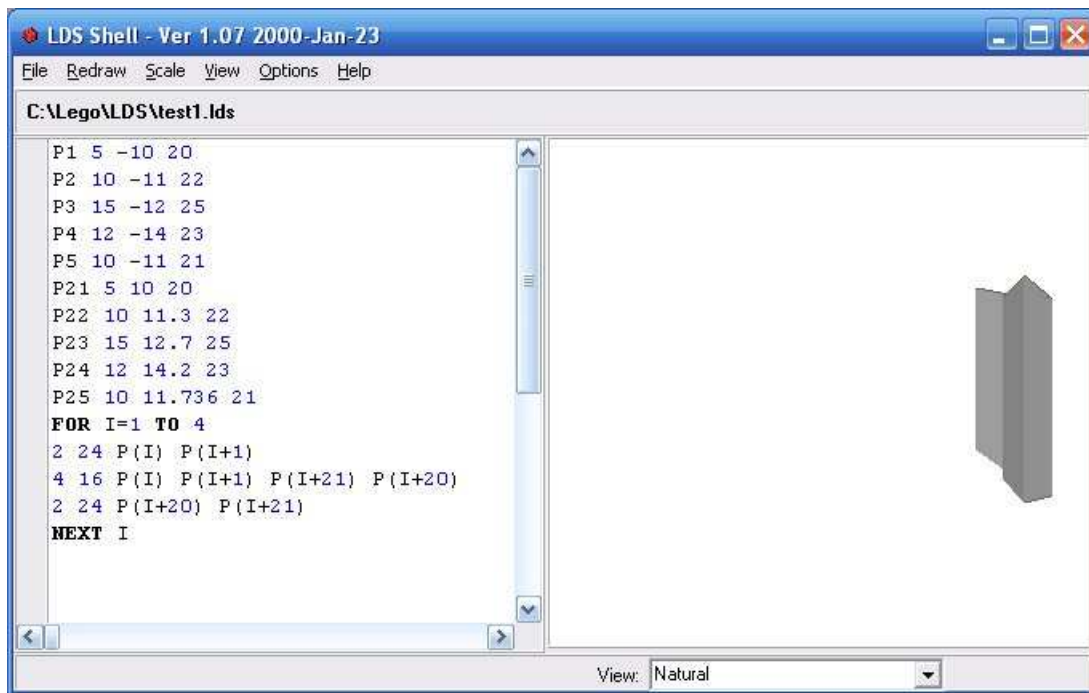
Script file successfully compiled.

C:\Lego\LDS>_
```

Exemple d'utilisation sous Dos.

Une autre façon de faire est de faire un glisser/déposer du fichier script (.lds) sur l'exécutable (lds.exe), dans l'explorateur Windows. Une fenêtre "Invite de commandes" s'ouvre montrant l'exécution. Tapez la touche "Entrée" pour fermer la fenêtre, et récupérer le résultat dans le fichier .DAT portant le nom du script.

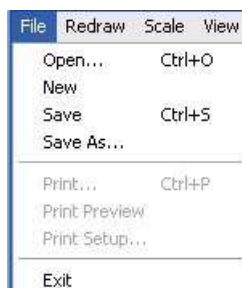
### Usage de LDS Shell



Avec LDS Shell, vous pouvez voir instantanément le fichier généré par le script en cliquant sur "**Redraw**", ou en appuyant sur la touche **F5** du clavier !

## Menus de LDS Shell

### File (Fichier)



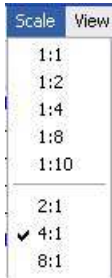
- **Open...** : Ouvre un fichier script existant .LDS, ou un fichier LDraw .DAT, ou un fichier texte .TXT.
- **New** : Ouvre un fichier vide tmp.lds.
- **Save** : Sauvegarde le fichier en cours, avec son nom et son dossier par défaut, au format .LDS pour le script, **et** au format .DAT pour le fichier converti en LDraw.  
Nota : En raison des limitations du programme au format DOS, ne pas mettre plus de 8 caractères dans le nom.
- **Save As...** : Sauvegarde le fichier en cours, avec un nom et un dossier à définir dans la boîte de dialogue qui s'ouvre.
- **Print** : Non implémenté.
- **Print Preview** : Non implémenté.
- **Print Setup...** : Non implémenté.
- **Exit** : Sortie du programme.

### Redraw (Redessine)

### Redraw

- **Redraw** : Met à jour le script dans la partie gauche, et regénère l'image de la partie droite à partir du script.

## Scale (Echelle)

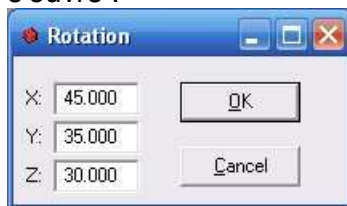


- **1:1** : Affiche l'image à l'échelle 1.
- **1:2 à 1:10** : Affiche l'image réduite de l'échelle 0.5 à 0.1.
- **2:1 à 8:1** : Affiche l'image agrandie à l'échelle 2, 4 ou 8.

## View (Visualisation)

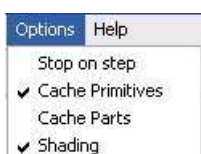


- **3D** : Affiche l'image en vue 3D.
- **Custom...** : Affiche l'image suivant les angles de rotation entrés dans la boîte de dialogue qui s'ouvre :



- **Front** : Affiche l'image vue de face.
- **Right** : Affiche l'image vue de droite.
- **Left** : Affiche l'image vue de gauche.
- **Back** : Affiche l'image vue de derrière.
- **Over** : Affiche l'image vue de dessus.
- **Under** : Affiche l'image vue de dessous.

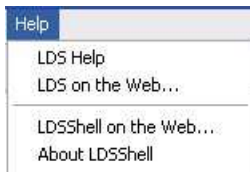
## Option (Options d'affichage)



- **Stop on step** : Si coché, fait une pause à chaque STEP.

- [Cache Primitives](#) : Si coché, ...
- [Cache Parts](#) : Si coché, ...
- [Shading](#) : Si coché, affiche l'image en vue ombrée.

## Help (Aide)



- [LDS Help](#) : Non implémenté.
- [LDS on the web...](#) : Ouvre une ancienne version de la page Internet LDS (si connecté).
- [LDSShell on the web...](#) : Ouvre la page Internet LDS Shell (page qui n'existe plus).
- [About LDSShell](#) : Affiche les copyrights du programme.

## Visualisation avec LDView

La fenêtre d'affichage de **LDS Shell** étant limitée en possibilités, il est possible d'utiliser le programme **LDView** pour visualiser le résultat du script en cours de création ou de modification.

Pour cela :

- Ecrire ou charger un script.
- Le sauvegarder, si ce n'était pas fait, sous un nom comme par exemple "test.lds" (menu : "File / Save As...").
- Lire le résultat du script, par exemple : "TEST.DAT" avec **LDView**.
- A chaque modification du script, sauvegarder la modification (menu : "File / Save").
- Sélectionner la fenêtre de **LDView**, pour qu'automatiquement elle soit mise à jour.

## Langage Script de LDS

### Contenu

- [CONSTANTES](#)
- [VARIABLES et TABLEAUX](#)
  - [ADRESSE INDIRECTE](#)
- [LIGNES LDRAW](#)
- [COMMANDES ou FONCTIONS](#)
  - [\\*\\*\\*\\*.DAT](#)
  - [COPY](#)
  - [CROSS et NOCROSS](#)
  - [ECHO ou EXTRUDE](#)
  - [END](#)
  - [END IF ou ENDIF](#)
  - [FILL et NOFILL](#)
  - [FOR - TO - STEP](#)
  - [GOSUB et RETURN](#)
  - [GOTO ou GO TO](#)
  - [IF - THEN - ELSE](#)
  - [LET](#)
  - [LINE](#)

- MIRROR
- NEXT
- PIPE
- PLOT
- POINT
- PRINT ou ?
- REM ou '
- RING
- SEGM ou SEGMENT
- WHILE et WEND

## CONSTANTES

Les noms des constantes définissent un certain nombre de numéros de couleurs au standard LDraw. Elles sont maintenant définies dans un fichier éditable **ldsconst.lst** :

BLACK = 0 (noir)  
 BLUE = 1 (bleu)  
 GREEN = 2 (vert)  
 DARKCYAN = 3 (bleu cyan foncé)  
 RED = 4 (rouge)  
 MAGENTA = 5 (Magenta ou rose foncé)  
 BROWN = 6 (marron)  
 GRAY ou GREY ou LIGHTGRAY ou LIGHTGREY = 7 (gris clair)  
 DARKGRAY ou DARKGREY = 8 (gris foncé)  
 LIGHTBLUE = 9 (bleu clair)  
 LIGHTGREEN = 10 (vert clair)  
 CYAN ou LIGHTCYAN = 11 (bleu cyan)  
 LIGHTRED = 12 (rouge clair)  
 PINK = 13 (rose)  
 YELLOW = 14 (jaune)  
 WHITE = 15 (blanc)  
 CLEARBLUE = 33 (transparent bleu)  
 CLEARGREEN = 34 (transparent vert)  
 CLEARRED = 36 (transparent rouge)  
 CLEARLIGHTBLUE = 41 (transparent bleu clair)  
 CLEARYELLOW = 46 (transparent jaune)  
 CLEAR ou TRANSPARENT = 47 (transparent blanc)  
 GOLD = 334 (or)  
 TAN ou SAND = 382 (jaune sable)  
 CHROME ou SILVER = 383 (chrome ou argent)  
 MINTGREEN = 431 (vert menthe)  
 ORANGE = 462 (orange)  
 ELECTRIC = 494 (jaune des contacts électriques)  
 LIGHTYELLOW = 495 (jaune clair)

Autres constantes du fichier ldsconst.lst :

PI = 3.14159

## VARIABLES et TABLEAUX

### Variable simple

A-Z

Nota : les noms composés de plusieurs lettres ne sont pas autorisés.

### Variable de point 3D

X0-X99, Y0-Y99, Z0-Z99

Pour les coordonnées X, Y, ou Z des points.

P0-P99

Pour les points en coordonnées 3D, avec :

P0 est X0 Y0 Z0

P1 est X1 Y1 Z1

...

P99 est X99 Y99 Z99.

Nota : P sans nombre accolé est une variable simple, sans aucun rapport avec les autres variables simples X Y Z.

Nota : La fonction PLOT n'est pour l'instant pas capable d'utiliser les variables.

## ADRESSE INDIRECTE

X(i), Y(i), Z(i)

P(i)

où (i) définit le contenu d'une variable ou le résultat d'un calcul.

Exemple :

```
p1 5 7 9
p2 6 8 10
p3 11 12 13
for i=1 to 3
  p(i+10) = z(i) y(i) x(i) * 1 1 -1
  2 24 p(i) p(i+10)
next i
```

donne :

```
2 24 5 7 9 9 7 -5
2 24 6 8 10 10 8 -6
2 24 11 12 13 13 12 -11
```

Nota : La fonction PRINT n'est pour l'instant pas capable d'utiliser les adresses indirectes.

## LIGNES LDRAW

Crée directement une ligne de commande LDraw de type 0 à 5.

Les remarques et méta-commandes sont copiées telles quelles.

Les variables éventuelles sont remplacées par leurs valeurs.

Exemples :

```
0 // Exemple de tous les types de lignes
0 // Nota les variables seront remplacées par les valeurs déclarées avant
1 16 x y 0 1 0 0 0 1 0 0 0 1 finger1.dat
2 24 p12 p62
3 16 p1 p6 p3
4 16 p1 p11 p16 p6
5 24 p10 p11 p16 p17
```

## COMMANDES ou FONCTIONS

### \*\*\*\*.DAT

Crée une ligne de commande de primitive \*\*\*\*.dat à la position donnée, et en orientation par défaut.

Format :

\*\*\*\*.dat [couleur] x y z

La couleur par défaut est 16

**Exemple 1 :**

```
stud.dat 80 0 40
```

**donne :**

```
1 16 80 0 40 1 0 0 0 1 0 0 0 1 STUD.DAT
```

**Exemple 2 :**

```
3001.dat BLUE 0 0 0
```

**donne :**

```
1 1 0 0 0 1 0 0 0 1 0 0 0 1 3001.dat
```

**COPY**

Copie un point 3D.

**Format :**

```
COPY Var1 Var2
```

**Exemple :**

```
COPY P1 P2
```

Nota : Equivalent à  $X2=X1$   $Y2=Y1$   $Z2=Z1$ .

**CROSS et NOCROSS**

Paramètres de la commande [LINE](#).

Détermine quand une ligne de bord (2 24...) entre une commande ECHO POINT et sa ligne echo doit être dessinée.

Si CROSS est mis en dehors du bloc de commande LINE, il est valide globalement.

Si CROSS est mis dans le bloc de commande LINE, il est valide localement dans ce bloc, c'est-à-dire jusqu'à la commande LINE END.

Si CROSS est mis à la fin d'une commande POINT, il l'affecte uniquement.

**Exemple :**

```
ECHO 0 20 0
CROSS
LINE
  POINT 100 100 0
  POINT 0 200 0 NOCROSS
LINE END
```

**donne :**

```
2 24 0 100 0 0 120 0 - Première ligne de bord (CROSSed) partant du premier point.
2 24 0 100 0 0 200 0 - Première ligne entre les 2 premiers points de la commande LINE.
2 24 0 120 0 0 220 0 - Seconde ligne en "ECHO" de la commande LINE.
4 16 0 100 0 0 200 0 0 220 0 0 120 0 - Création d'un polygone entre les lignes en
"ECHO" (Voir FILL qui est ON par défaut).
(la seconde ligne de bord n'est pas créée (NOCROSS)).
```

La valeur pour CROSS est ON.

Passer en mode CROSS remet à zéro le mode COND.

**ECHO ou EXTRUDE**

Transforme des lignes et bords (edge) 2D en 3D.

**Format :**

```
ECHO X Y Z | EXTRUDE X Y Z
...définitions de lignes ou bords
ECHO OFF | EXTRUDE OFF
```

Chaque ligne LDraw est insérée tel quelle, plus une ligne décalée suivant XYZ, et une facette quadrangulaire (type 4) entre les deux lignes.



Chaque "edge" LDraw est insérée tel quelle, plus un "edge" décalée suivant XYZ, et un cylindre entre les deux.

Exemple :

```
ECHO 0 24 0
2 24 100 0 0 200 0 0
1 16 40 0 40 10 0 0 0 0 0 10 4-4edge.dat
ECHO OFF
```

donne :

```
2 24 100 0 0 200 0 0
2 24 100 24 0 200 24 0
4 16 100 0 0 100 24 0 200 24 0 200 0 0
1 16 40 0 40 10 0 0 0 0 0 10 4-4edge.dat
1 16 40 24 40 10 0 0 0 0 0 10 4-4edge.dat
1 16 40 0 40 10 0 0 0 24 0 0 10 4-4cylind.dat
```

## END

Termine l'interprétation du script. En fin de script, cette commande est optionnelle. Généralement utilisé pour séparer les sous-programmes du script principal.

Format :

```
[END]
```

Exemple :

```
END
```

## END IF ou ENDIF

Termine le bloc de lignes de script d'une commande IF-THEN[-ELSE].

Format :

```
END IF | ENDIF
```

Exemple :

```
ENDIF
```

## FILL et NOFILL

Paramètres de la commande LINE.

Détermine quand l'espace entre une ligne mise en ECHO et cet echo doit être rempli par un quadrilatère (Filled).

Si FILL est utilisé en dehors du mode LINE, cela est fait globalement.

Si FILL est utilisé dans le mode LINE, cela est fait localement, jusqu'à rencontrer un LINE END.

La valeur par défaut de FILL est ON.

Exemple :

```
ECHO 0 20 0
FILL
NOCROSS
LINE
POINT 100 100 0
POINT 0 200 0
LINE END
```

## FOR - TO - STEP

Définition de boucle de commande, de Val1 à Val2 avec un pas de Val3.

Format :

```
FOR Var=Val1 TO Val2 [STEP Val3]
```

**Exemples :**

```
for i=1 to 15
for x=-200 to 200 step 20
for x=10 to 1 step -1
```

Nota : La fin de la boucle est défini par la commande [NEXT](#).

**GOSUB et RETURN**

Définissent l'appel et la fin d'un sous-programme dans le script. Le label définit le nom du sous-programme.

**Format :**

```
GOSUB [:]<label>
....
RETURN
```

**Exemple :**

```
for i=1 to 5
  gosub :hello
next i
end

:hello
  print 0 "Hello";
  gosub world
  return

:WORLD
  print "World."
  return
```

**GOTO ou GO TO**

Définit un saut de lignes dans le script. Le label définit l'endroit où aller. Nota : Cette commande est à éviter, car cela rend difficile la mise au point du script. Préférez la commande GOSUB.

**Format :**

```
GOTO [:]<label>
```

**Exemple :**

```
:DACAPO
PRINT "0 Ne pas utiliser à la maison!"
GOTO DACAPO
```

**IF - THEN - ELSE**

Exécute les lignes de script de façon conditionnelle.

**Format :**

```
IF condition THEN [...]
[...]
ELSE [...]
[...]
END IF | ENDIF
```

**Exemple 1 :**

```
if i>10 then i=1
```

**Exemple 2 :**

```
if a=20 then goto :FINISH
```

**Exemple 3 :**

```

if i>10 then
  i=1
else
  i=i+1
end if

```

Exemple 4 NON autorisé :

```
if i>10 then i=1 else i=i+1
```

Pourquoi ? parce que ELSE doit être sur une ligne de script séparée, comme dans l'exemple 3.

Voir également END IF, et GOTO.

## LET

Affecte une valeur ou le résultat d'un calcul à une variable. Nota : le nom de la fonction "LET" est optionnel.

Format :

```
[LET] Variable = Calcul
[LET] Variable = X Y Z
```

Exemple 1 :

```
LET A=0
```

Exemple 2 :

```
b=a*2
```

Exemple 3 :

```
b = 2*a
```

Exemple 4 :

```
c=-.5/3
```

Exemple 5 :

```
p15 = 100 24 60
```

Exemple 6 NON autorisé :

```
b = 2 * a
```

Pourquoi ? parce que "2 \* a" sera interprété comme trois arguments.

## LINE

Génère une ou succession de lignes entre chaque point défini et le suivant.

Format :

```
LINE [CROSS | NOCROSS] [FILL | NOFILL] [END | OFF]
```

Exemple :

```

LINE
POINT 10 20 30
POINT 40 50 60
70 80 90
LINE END

```

donne :

```

2 24 10 20 30 40 50 60
2 24 40 50 60 70 80 90

```

## MIRROR

Symétrise les éléments qui suivent dans le script (en plus de l'élément non symétrisé).

Nota : La symétrie tient compte de leur sens BFC.

Format :

```

MIRROR [+]X | [+]Y | [+]Z
MIRROR -X | -Y | -Z
MIRROR 0 | OFF

```

où :

```

MIRROR +X o MIRROR X
Début de symétrie en X
MIRROR -X
Fin de symétrie en X
MIRROR 0 o MIRROR OFF
Fin de toutes les commandes de symétrie.

```

Exemples, avec résultat sous LDView :

```

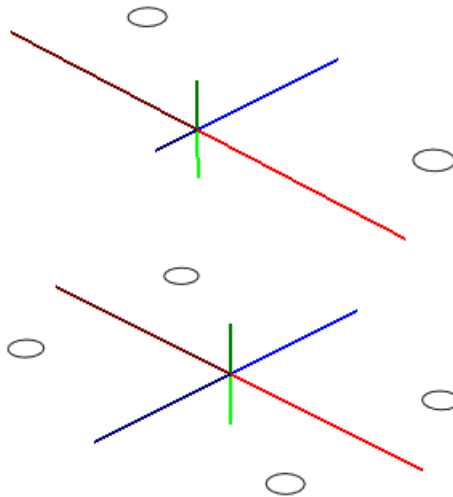
MIRROR X
4-4edge.dat 16 10 0 6
MIRROR OFF

```

```

MIRROR X
MIRROR Z
4-4edge.dat 16 10 0 6
MIRROR OFF

```



## NEXT

Définition de fin d'un bloc d'une boîte.

Format :

```
NEXT [Var]
```

Nota : La variable d'index est optionnelle, et est ignorée par le compilateur.

Exemples :

```

NEXT
Next i

```

## PLOT

Génère dans le script les coordonnées des points d'un cercle ou portion de cercle, et les insère à la suite dans le fichier .LDS. L'insertion se fait sur une commande d'un menu "Redraw".

Format :

```
PLOT @Pnn [sectors totalsectors] x y z a b c d e f g h i
```

où :

sectors : Nombre de secteurs. Par défaut 16.

totalsectors : Nombre total de secteurs sur 360°. Par défaut 16.

x y z : Position du centre du cercle.

a b c d e f g h i : Matrice d'orientation du cercle.

Exemple :

```
PLOT @P20 4 16 0 0 0 10 0 0 0 0 10 0 0 0
```

après un Redraw, donne dans le fichier .LDS :

```

'PLOT @P20 4 16 0 0 0 10 0 0 0 0 10 0 0 0
P20 10 0 0
P21 9.239 3.827 0
P22 7.071 7.071 0

```

```
P23  3.827 9.239 0
P24  0 10 0
```

On peut ajouter les lignes suivantes pour créer les 4 lignes formant un 1/4 de cercle :

```
FOR i=20 to 23
  2 24 P(i) P(i+1)
NEXT i
```

Nota : La fonction PLOT n'est pas capable d'utiliser des variables.

## PIPE

Format :

```
PIPE [sectors totalsectors] [colour]
```

Maille un cercle ou portion de cercle en tuyau (pipe).

Valeurs par défaut : sectors=16, totalsectors=16, colour=16

Voir également [SEGMENT](#).

Nota : *Semble non fonctionnel.*

## POINT

Définit les coordonnées d'un point dans le script.

Format :

```
[POINT] x y z
```

Le nom de la commande est optionnelle uniquement dans un bloc [LINE](#).

Exemple 1 :

```
ECHO 0 10 0
POINT 1 2 3
```

donne (en mode CROSS) :

```
2 24 1 2 3 1 12 3
```

Exemple 2 : NON autorisé

```
LINE
...
LINE OFF
ECHO 0 10 0
1 2 3
```

Pourquoi ? Parce que "POINT" est optionnel uniquement à l'intérieur d'un bloc [LINE](#).

## PRINT ou ?

Affiche (génère) un message texte, ou affiche le contenu d'une variable ou d'une constante.

Format :

```
PRINT | ? ["<text>" | variable | constante] [;]
```

Le caractère ; annule le passage à la ligne dans le fichier résultat.

Nota : Ce point-virgule est permis uniquement comme dernier caractère dans une ligne de commande PRINT.

Exemple :

```
P1 11 22 33.33
PRINT "0 TRANSLATE" p1
3001.dat RED 100 0 0
PRINT "0 TRANSLATE";
PRINT "OFF"
```

donne :

```
0 TRANSLATE 11 22 33.33
```

```
1 4 100 0 0 1 0 0 0 1 0 0 0 1 3001.dat
0 TRANSLATE OFF
```

Avec **LDraw 1.5 et version plus récente - et non avec LDraw !** - cela est interprété comme :

```
1 4 111 22 33.33 1 0 0 0 1 0 0 0 1 3001.dat
```

**Nota :** PRINT est actuellement non affiché (ECHOed), et non symétrisé (MIRRORed), mais peut-être cela devrait l'être ?

## REM ou '

Insère une ligne de commentaire dans le script. Elle est ignorée par le compilateur.

Format :

```
REM .... | ' ....
```

Exemple :

```
REM Définition ergot
```

## RING

Génère un ensemble de facettes formant un rectangle inscrivant un anneau, ou portion d'anneau circulaire ou elliptique.

Format :

```
RING [segments total_segments] color1 color2 color3 x y z
ai ci di fi gi ii ao co do fo go io
```

**Nota :** Cette commande est sur deux lignes.

où :

segments : Nombre de segments ou secteurs.

total\_segments : Nombre total de segments ou secteurs sur 360°. Les valeurs acceptées sont 16 pour les cercles standards et 48 pour les cercles en haute résolution.

Ces 2 valeurs sont conjointement optionnelles, et si elles sont omises leur valeur est : 16/16

color1 : Couleur du cercle intérieur. Si la couleur est négative, alors il n'est pas généré.

color2 : Couleur de l'anneau.

color3 : Couleur du complément de cercle extérieur. Si la couleur est négative, alors il n'est pas généré.

x y z : Position du centre des cercle et anneau.

ai ci di fi gi ii : Matrice du cercle intérieur, où ai=rayon en X, et ii=rayon en Z.

ao co do fo go io : Matrice du cercle extérieur, où ao=rayon en X, et io=rayon en Z.

Exemple :

```
RING 8 16 14 0 14 100 0 0
7 0 0 0 0 5 9 0 0 0 0 7
```

donne le dessin suivant :



**Nota :** les matrices ne comportent que 6 valeurs au lieu de 9, car cette commande ne travaillant que sur des éléments 2D, les "**valeurs Y**" de la matrice ont été omises, c'est-à-dire, les 2ème, 5ème, et 7ème arguments de la matrice de rotation qui sont toujours égaux à 0.

Dans l'exemple qui précède, les 2 champs jaunes pourraient être créés par 2 lignes de commande LDraw :

```
1 14 100 0 0 7 0 0 0 0 0 0 5 2-4disc.dat
1 14 100 0 0 9 0 0 0 0 0 0 7 2-4ndis.dat
```

```
( "valeurs Y"      ^      ^      ^      )
```

... où l'on voit en rouge les paramètres omis de la matrice.

L'espace compris entre les parties jaunes, est un anneau noir composé de quadrilatères sans équivalent à une seule ligne de commande LDraw "2-4ring...".

## SEGM ou SEGMENT

Format :

```
SEGM | SEGMENT  x y z  a b c  d e f  g h i
```

Génère une coupe dans la fonction [PIPE](#).

Nota : *Semble non fonctionnel.*

## WHILE et WEND

Définit une boucle conditionnelle, ou la condition est en début de boucle, ce qui permet d'avoir la possibilité d'une boucle non exécutée en fonction de la valeur de la variable.

Nota : Une boucle [FOR - TO - STEP](#) est exécutée au moins une fois.

Format :

```
WHILE condition
```

```
....
```

```
WEND
```

Exemple :

```
let i=3
while i<7
  print 0 "Hello"
  i=i+1
wend
```

donne :

```
0 Hello
0 Hello
0 Hello
0 Hello
```

## Exemples de l'auteur du programme

### Créer des quadrilatères entre 2 chemins de lignes

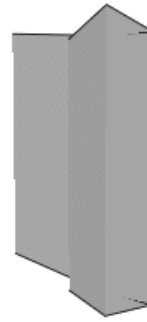
Par exemple, vous définissez les 5 points des lignes du sommet, P1 à P5, les 5 points des lignes secondaires, P21 à P25, puis écrivez un simple script :

```
P1 5 -10 20
P2 10 -11 22
P3 15 -12 25
P4 12 -14 23
P5 10 -11 21
P21 5 10 20
P22 10 11.3 22
P23 15 12.7 25
P24 12 14.2 23
P25 10 11.736 21
FOR I=1 TO 4
2 24 P(I) P(I+1)
4 16 P(I) P(I+1) P(I+21) P(I+20)
2 24 P(I+20) P(I+21)
NEXT I
```

Puis vous lancez le script à travers LDS, et vous obtenez le code LDraw.

Crée un fichier .DAT contenant :

```
2 24 5 -10 20 10 -11 22
4 16 5 -10 20 10 -11 22 10 11.3 22 5 10 20
2 24 5 10 20 10 11.3 22
2 24 10 -11 22 15 -12 25
4 16 10 -11 22 15 -12 25 15 12.7 25 10 11.3 22
2 24 10 11.3 22 15 12.7 25
2 24 15 -12 25 12 -14 23
4 16 15 -12 25 12 -14 23 12 14.2 23 15 12.7 25
2 24 15 12.7 25 12 14.2 23
2 24 12 -14 23 10 -11 21
4 16 12 -14 23 10 -11 21 10 11.736 21 12 14.2 23
2 24 12 14.2 23 10 11.736 21
```



Ce fichier contient 5 groupes de lignes et quadrilatères définis par le Script.

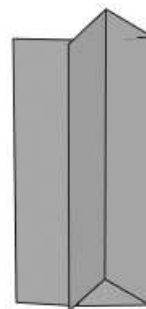
### Créer des quadrilatères suivant 1 chemin de lignes

Par exemple, pour extruder des lignes 2D en 3D avec une hauteur constante de 24 LDU en Y, vous reprenez les 5 points de l'exemple précédent P1 à P5, et écrivez le script :

```
P1 5 -10 20
P2 10 -11 22
P3 15 -12 25
P4 12 -14 23
P5 10 -11 21
EXTRUDE 0 24 0
LINE
P1
P2
P3
P4
P5
LINE END
EXTRUDE OFF
```

Crée un fichier .DAT contenant :

```
2 24 5 -10 20 5 14 20
2 24 10 -11 22 10 13 22
2 24 5 -10 20 10 -11 22
2 24 5 14 20 10 13 22
4 16 5 -10 20 10 -11 22 10 13 22 5 14 20
2 24 15 -12 25 15 12 25
2 24 10 -11 22 15 -12 25
2 24 10 13 22 15 12 25
4 16 10 -11 22 15 -12 25 15 12 25 10 13 22
2 24 12 -14 23 12 10 23
2 24 15 -12 25 12 -14 23
2 24 15 12 25 12 10 23
4 16 15 -12 25 12 -14 23 12 10 23 15 12 25
2 24 10 -11 21 10 13 21
2 24 12 -14 23 10 -11 21
2 24 12 10 23 10 13 21
4 16 12 -14 23 10 -11 21 10 13 21 12 10 23
```

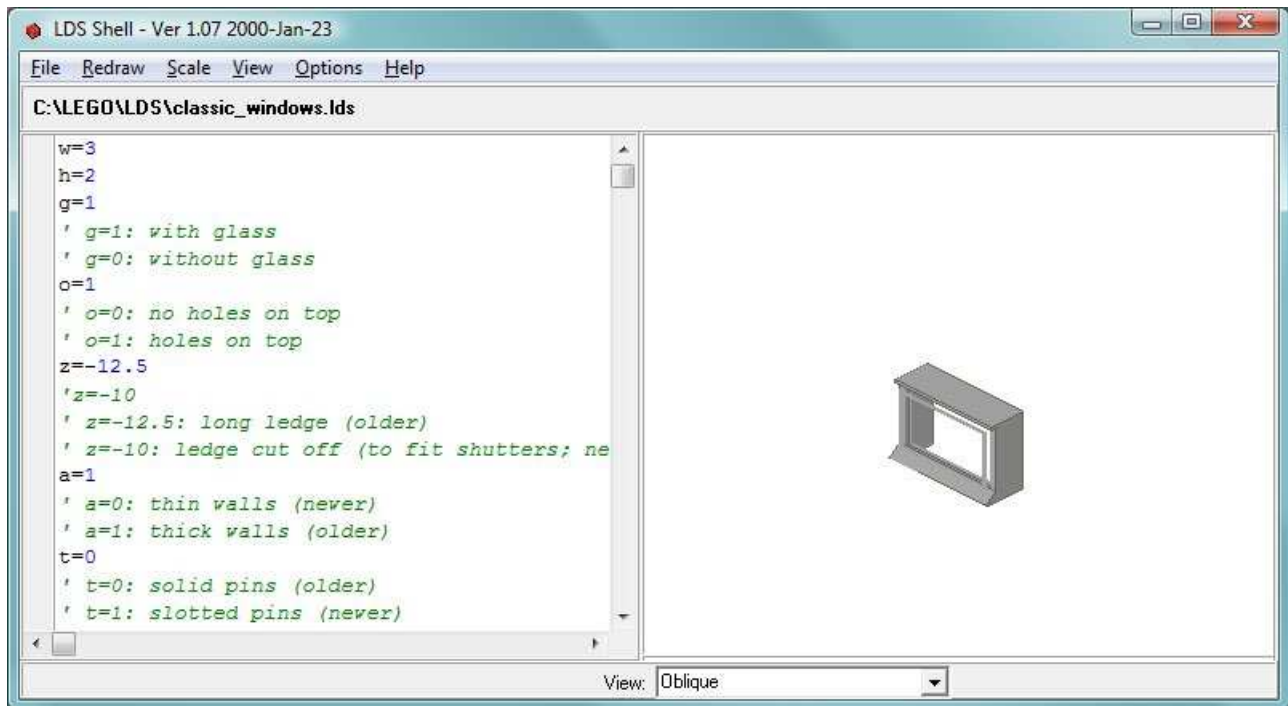


Ce fichier crée 4 surfaces et les lignes de bord correspondantes.

### Créer la gamme des fenêtres "classic"

Ce long exemple permet de voir une application de création de toute une gamme de pièces LDraw. Il suffit de modifier les paramètres au début du script et de lancer la commande Redraw pour voir le résultat.

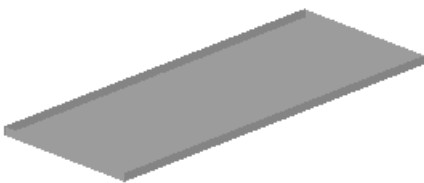




Vous pouvez télécharger le script : [class\\_w.lds](#) pour le tester et vous en servir d'exemple.

## Exemples de J.C. Tchang

### Créer la boîte englobante des autocollants rectangulaires

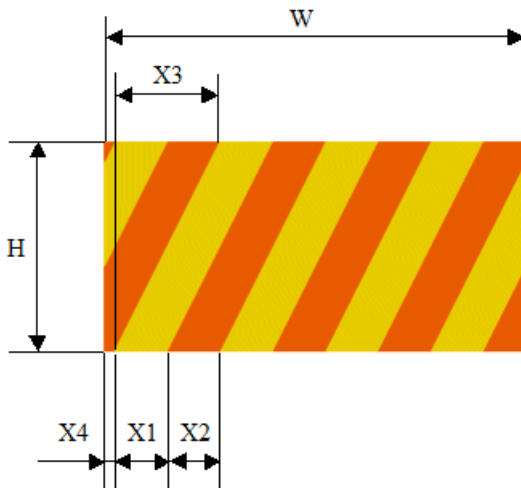


Utilisez ce script pour faire les 5 faces englobant les autocollants (Sticker). L'origine est au centre de la face inférieure.

Nota : Depuis 06-2009, il est préférable d'utiliser la primitive **box5-12.dat** pour les pièces LDraw, le script **Indice A** permet de créer cette boîte à la bonne dimension et la bonne position, tout en gardant l'ancienne possibilité.

Vous pouvez télécharger le script : [stk\\_box.lds](#), indice **A**, et vous en servir.

### Créer des autocollants à bandes

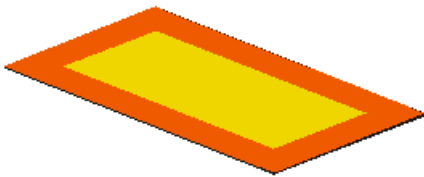


Utilisez ce script pour créer des autocollants (Sticker) à bandes alternées. L'origine est au centre de la face.

Nota : Ce script ne prend pas en compte tous les cas de figures. En particulier X3 doit être supérieur à X1 et X2.

Vous pouvez télécharger le script : [stk\\_band.lvs](#), indice **A** et vous en servir.

### Créer des autocollants à cadre

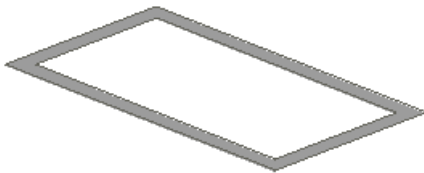


Utilisez ce script pour créer des autocollants (Sticker) à cadre.

Nota : Les valeurs du rectangle intérieur peuvent être nulles. Cela génère un autocollant monochrome.

Vous pouvez télécharger le script : [stk\\_frm1.lvs](#), indice **A** et vous en servir.

### Créer des cadres



Utilisez ce script pour créer des cadres. C'est-à-dire 4 faces quadrangulaires, souvent utilisé dans la conception des pièces, avec les 8 lignes de bords.

Nota : Les valeurs du rectangle intérieur peuvent avoir les mêmes valeurs que le rectangle extérieur. Cela génère uniquement les 4 lignes de bord.

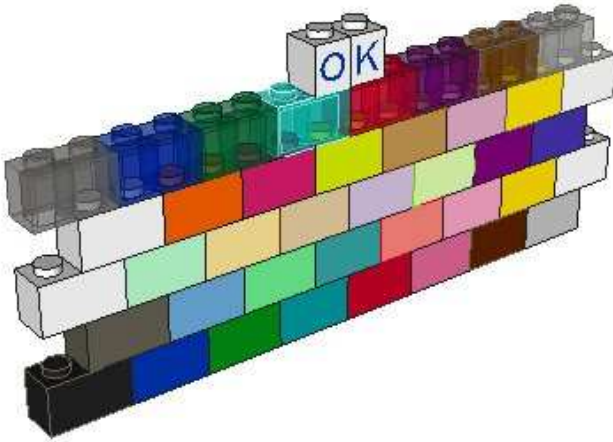
Vous pouvez télécharger le script : [stk\\_frm2.lvs](#) et vous en servir.

### Créer un mur de briques

Autre exemple de script, montrant qu'il est possible d'utiliser LDS pour créer des modèles. Cet exemple simple permet de construire un mur, de longueur et hauteur variables, avec des briques

1x2.

Nota : Le résultat ne s'affiche pas dans la fenêtre LDS Shell. Le niveau avec les briques "O" et "K" ne sont là que pour montrer un exemple de calcul.



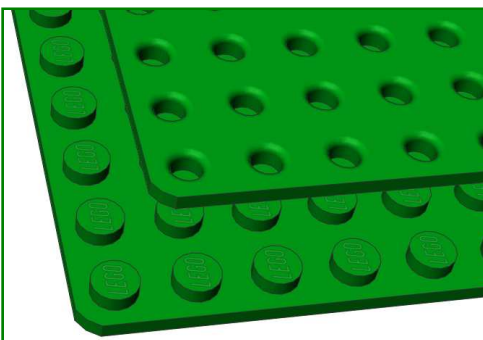
Vous pouvez télécharger le script : [wall.lds](#) et vous en servir.

### Créer une série de plaques de base (Baseplate)

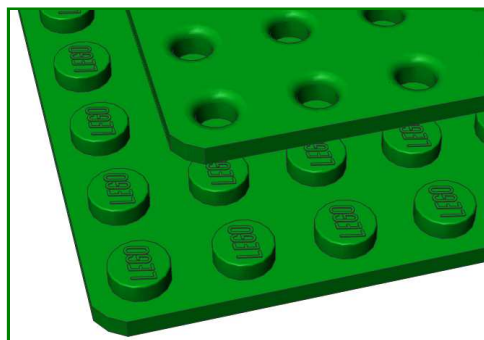
Autre exemple de script, pour créer la série des plaques de base comportant les évidements sous la plaque, pour ceux qui aiment voir ce qui se passe en dessous :-).

Nota : Ce script peut générer toutes les dimensions de plaques possibles, mais en dehors des plaques courantes, les tenons sur le dessus ne sont pas toujours complètement générés. Dans ce cas seul un groupe de tenons est généré au centre, et il faut compléter manuellement. De même, l'orientation des tenons n'est pas toujours vérifié.

Les coins générés représentent le cas le plus courant (coin coupé, légèrement arrondi). Il existe malgré tout des plaques avec des coins à grands rayons, et pour certaines plaques récentes des coins à petits rayons, non prévus par ce script.



Exemple avec tenon "stud.dat" standard, sous LDView en mode substitution de primitives.

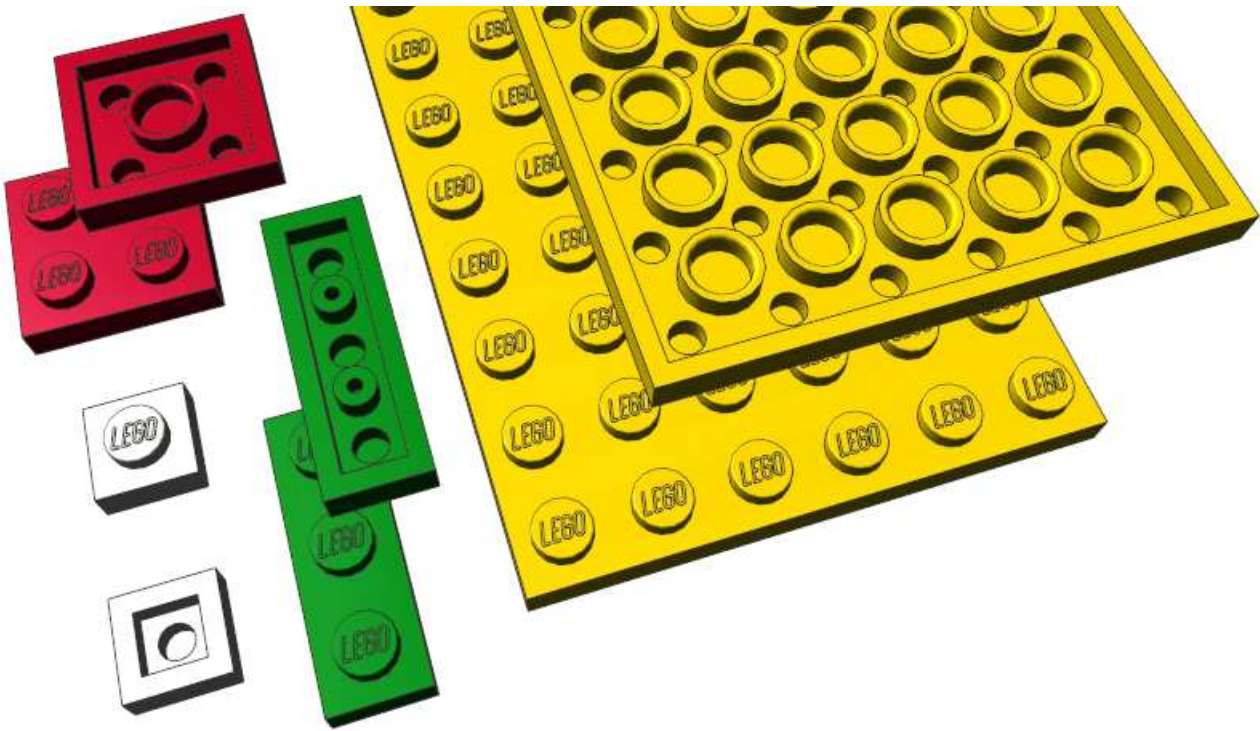


Exemple avec tenon "stud.dat" spécial, sous LDView, voir : [Tenons spéciaux avec logo filaire ou embossé.](#)

Vous pouvez télécharger le script : [bp\\_hu.lds](#) et vous en servir.

### Créer une série de plates en haute définition

Autre exemple de script, pour créer la série des plates comportant les évidements en-dessous.



Nota : Ce script peut générer toutes les dimensions de plats possibles, de 1 x 1, à N x N.

Vous pouvez télécharger le script : [plate\\_hd.lsd](#) et vous en servir.

Pour utiliser les fichiers générés, il faut avoir certaines primitives Non Officielles, que vous pouvez retrouver au chapitre : [Briques et plates Haute Définition \(HD\)](#), de ma page [Pièces LDraw de Haute Définition](#).

## Droits et copyright

### Historique LDS

(c) Tore Eriksson.

[Tore's Home Page](#) (Page de démarrage de Tore Eriksson).

- V0.64 : Version du 28 octobre 2011.
  - Ajouté support de ldsconst.lst, liste des constantes éditables par l'utilisateur.
- V0.62 : Version du 28 octobre 2011.
  - Recompilé avec Open Watcom IDE v1.9 pour supporter les systèmes 64 bits.
- V0.61 : Version du 14 mars 2006.
  - Ré-écriture des fonctions de symétrie et extrusion, pour tenir compte de la compliance BFC.
  - Génère automatiquement le fichier de sortie \*.dat si le seul argument fourni en entrée est le script, au lieu de générer une erreur et d'arrêter le programme.
  - Limite et formate les lignes générées pour une meilleur lisibilité.
- V0.31 : Version du 2 avril 2000.
- V0.29 : Version du 26 février 2000.

Développements à venir (*Tore Eriksson*) :

- ~~Création d'un fichier INI, pour que les utilisateurs puissent définir leur propre constantes de couleur. Actuellement les constantes sont codées dans le programme, et sont "très" hors d'âge.~~

- Fonctions COND / CONDITIONAL - NOCOND / NOCONDITIONAL ne sont actuellement pas supportées.

Souhaits (*J.C. Tchang*) :

- Ajouter les fonctions trigonométriques (sin, cos, tan, asin, ...) et racines carrés.
- Accepter les IF imbriqués.
- ... Lire un fichier et en extraire des valeurs.

## **Historique LDS Shell**

(c) Anders Isaksson.

- V1.07 : Version du 23 janvier 2000.

## **Bugs ou limitations de LDS Shell**

1. Les cercles et cylindres ne sont pas affichés !!!.

## **Ce manuel en français**

*Traduction et Adaptation : J.C. Tchang.*