eBook Gratuit

APPRENEZ postscript

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#postscript

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: postscript

It is an unofficial and free postscript ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official postscript.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec PostScript

Remarques

PostScript est un langage de script dynamique, basé sur une pile, à typage dynamique, à typage dynamique, avec des primitives intégrées pour générer des images de rendu à partir de descriptions vectorielles. PostScript utilise le même "modèle d'image Adobe" que le format de fichier PDF.

PostScript est utilisé comme format de sortie par de nombreux programmes car il est conçu pour être facilement généré par une machine.

Comme le LISP, PostScript est *homoiconique* et le code et les données partagent la même représentation. Les procédures peuvent prendre les procédures comme des données et des procédures de rendement en tant que résultats, en se prêtant également aux techniques de la *programmation concaténative* .

Examples

Installation ou configuration

Les authentiques interpréteurs Adobe PostScript sont disponibles dans les imprimantes haut de gamme, le produit Display PostScript (DPS) et le produit Acrobat Distiller. En tant qu'auteurs du standard, ces produits sont considérés comme "l'implémentation standard" dans le but de décrire les différences entre les implémentations PostScript.

L'interface Standard de l'interpréteur définie dans le PLRM est le *flux de programme* qui peut être du texte ou du binaire, selon les détails du canal sous-jacent ou du système d'exploitation / contrôleur. Acrobat Distiller dispose d'une interface graphique utilisateur permettant de sélectionner le programme postscript en entrée et de rendre sa sortie en format pdf. Distiller a également un support limité pour utiliser le flux de texte de sortie pour signaler les erreurs et autres sorties de programme. GSView fournit une interface graphique similaire pour un workflow similaire en utilisant Ghostscript comme interpréteur.

Ghostscript et Xpost fonctionnent tous deux en mode ligne de commande. Le fichier programme postscript à exécuter peut être mentionné sur la ligne de commande (`gs program.ps` ou `xpost program.ps`) qui ouvrira une fenêtre graphique pour afficher la sortie graphique. Les options peuvent être utilisées pour rendre les graphiques ailleurs, comme un fichier disque, ou supprimer entièrement les graphiques et utiliser postscript comme un langage de script texte.

Les différents interprètes ont chacun leurs propres instructions d’installation et de configuration et il serait inutile de les reproduire ici (et risquerait d’être obsolètes).

Interprètes PostScript disponibles gratuitement

- Ghostscript est disponible pour toutes les principales plates-formes et distributions Linux, sous forme source ou binaire, sous licence GNU ou sous d'autres accords de licence avec les auteurs, le logiciel Artifex . Ghostscript implémente la norme PostScript 3 complète.

- Xpost est disponible sous forme de source pour toutes les principales plates-formes, sous la licence BSD-3-clause. Il implémente la norme de niveau 1 avec certaines extensions de niveau 2 et certaines extensions DPS.

Description générale de PostScript

PostScript est un langage de programmation général complet de Turing, conçu et développé par Adobe Systems. Beaucoup des idées qui ont fleuri dans PostScript ont été cultivées dans des projets pour Xerox et Evans & Sutherland.

Historiquement, sa principale application réelle est un *langage de description de page* ou, dans son fichier EPS à une seule page, un langage de description d'image vectoriel. Il est de type dynamique, à portée dynamique et basé sur une pile, ce qui conduit à une syntaxe essentiellement polonaise inversée.

Il existe trois versions principales de PostScript.

1. **PostScript Level 1** - il a été mis sur le marché en 1984 en tant que système d'exploitation résident de l'imprimante laser Apple LaserWriter, inaugurant l'ère de publication assistée par ordinateur.
2. **PostScript Level 2** - publié en 1991, il contenait plusieurs améliorations importantes au niveau 1, notamment la prise en charge de la décompression des images, la séparation in-RIP, les dictionnaires à croissance automatique, le nettoyage des données, les
3. **PostScript 3** - la version la plus récente et peut-être la plus largement adoptée a été publiée en 1997. Elle contient également plusieurs améliorations d'importation par rapport au niveau 2, telles que Smooth Shading. Le terme «niveau» a été supprimé.

Bien que PostScript soit généralement utilisé comme langage de description de page - et donc implémenté dans de nombreuses imprimantes pour générer des images raster - il peut également être utilisé à d'autres fins. Comme une calculatrice rapide inversée avec des noms d'opérateurs plus mémorables que `bc` . En tant que format de sortie généré par un autre programme (généralement dans une autre langue).

Bien que les fichiers PostScript soient généralement des fichiers ASCII à 7 bits, il existe plusieurs types de codage binaire décrits dans la norme de niveau 2. Etant programmable, un programme peut implémenter lui-même son propre schéma de codage complexe. Il existe un concours international de PostScript obscurci, un peu moins actif que le C.

Références en ligne

- Pages d'index de la documentation Adobe:
  https://www.adobe.com/products/postscript/resources.html
  http://www.adobe.com/devnet/postscript.html
  http://www.adobe.com/devnet/font.html

- Manuel de référence du langage PostScript, 3ed - Le standard PostScript 3. (7.41MB pdf) (Supplément , Errata)
- Manuel de référence du langage PostScript, 2ed - La norme PostScript Niveau 2. (inclut l'affichage de la documentation PostScript.) (3.29MB pdf)
- Tutoriel et livre de recettes Postscript - Le livre bleu. (847KB pdf)
- PostScript Language Program Design - Le livre vert. (911KB pdf)
- Penser en post - scriptum - Par l'auteur du livre vert et le tutoriel du livre bleu. (826KB pdf)
- Conventions de structuration des documents de langage PostScript Spécification 3.0 (521KB pdf)
- Format de police Adobe Type 1 (444 Ko pdf)
- Spécification du format de fichier PostScript encapsulé 3.0 (185 Ko pdf)
- PostScript Printer Description Spécification du format de fichier 4.3 (186KB pdf) (Mise à jour)
- Résoudre les erreurs PostScript - Conseils de débogage. (158Ko html)
- Acumen Journal - Archive des articles de programmation Postscript et PDF. (répertoire HTML des fichiers PDF compressés)
- Illustrations mathématiques: Manuel de géométrie et post - scriptum - par Bill Casselman. (répertoire HTML des chapitres pdf et téléchargements de code)
- Thread avec plusieurs implémentations d'algorithmes de tri (archive Usenet)
- Pages du gourou de Don Lancaster
- Utilisation directe d'Anastigmatix du langage Postscript
- Débogueur pas à pas open-source pour code Postscript

FAQ

- FAQ Usenet (vers 1995)
- FAQ PostScript de Wikibooks
- Questions SO PostScript triées par les vues les plus fréquemment consultées

Livres

- Manuel de référence du langage Postscript, 1ed, 1985. Recommandé pour sa petite taille et son index opérateur simple à partir des pages de résumé (absent des éditions ultérieures).

- PostScript du monde réel. Chapitres de divers auteurs sur divers sujets, y compris une excellente couverture des demi-teintes.

Espaces de noms locaux pour les fonctions

Postscript est un *langage* dynamique ou un *langage LISP 1* . Mais il fournit les outils pour implémenter des variables locales dans les procédures et d'autres effets nécessaires pour implémenter des algorithmes.

Pour les noms locaux dans une procédure, créez un nouveau dictionnaire au début et affichez-le à la fin.

```
/myproc {
    10 dict begin
    %... useful code ...
    end
 } def
```

Vous pouvez également combiner cela avec un raccourci pour définir les arguments de la fonction en tant que variables.

```
% a b c  myproc  result
/myproc {
    10 dict begin
    {/c /b /a} {exch def} forall
    %... useful code yielding result ...
    end
 } def
```

Si vous devez mettre à jour une variable * "global" * alors que le dictionnaire local est en haut, utilisez `store` au lieu de `def` .

Bonjour exemple mondial

Sélectionnez une police et une taille de police, sélectionnez l'emplacement, `show` chaîne.

```
%!PS
/Palatino-Roman 20 selectfont
300 400 moveto
(Hello, World!) show
showpage
```

Notes et pièges courants:

- Échec de la définition d'une police (résultant en une absence de texte ou une police par défaut (moche))
- Utiliser `findfont` et `setfont` mais en oubliant `scalefont` entre les deux. L'utilisation de `selectfont` niveau 2 évite ce problème et est plus concis.
- Ne pas définir un point avec `moveto` ou définir le point en dehors de la page. Pour les États-

Unis, le papier 8.5x11 fait 792x612 points de repère. Il est donc facile de se rappeler à peu près 800x600 (mais une odeur plus courte et plus large). Donc, `300 400` est à peu près le centre de la page (peu élevé, peu gauche).

- Oublier d'appeler `showpage` . Si vous prévisualisez un programme ps avec `gs` et qu'il ne se termine pas par `showpage` , `gs` peut afficher une image pour vous. Et pourtant, le fichier échouera mystérieusement lorsque vous essayez de convertir en pdf ou autre chose.

Programme scolaire

Lisez la documentation dans cet ordre pour apprendre facilement postscript:

1. Excellent tutoriel de Paul Bourke: http://paulbourke.net/dataformats/postscript/
2. Blue Book, premier semestre, le tutoriel officiel original: http://www-cdf.fnal.gov/offline/PostScript/BLUEBOOK.PDF
3. Livre vert, comment utiliser efficacement PostScript: http://www-cdf.fnal.gov/offline/PostScript/GREENBK.PDF
4. Penser en Postscript, "Nuff a dit: http://wwwcdf.pd.infn.it/localdoc/tips.pdf
5. Illustrations mathématiques . Commencez petit, construisez grand. Les maths derrière les courbes de Bezier. L'algorithme de découpage de polygones Hodgman-Sutherland. Transformations affines et transformations *non linéaires* du chemin. Dessin 3D et dégradé de Gouraud. De la préface:

   Lequel [des nombreux outils pour aider à produire des graphiques mathématiques] à choisir implique apparemment un compromis entre simplicité et qualité, dans lequel la plupart vont vers ce qu'ils perçoivent comme étant la simplicité. La vérité est que le compromis est inutile - une fois que l'on a fait un petit investissement initial d'effort, la meilleure chose à faire dans la plupart des situations est d'écrire un programme dans le langage de programmation graphique PostScript. La qualité de la sortie d'un programme PostScript est pratiquement illimitée et, à mesure que l'on acquiert de l'expérience, les difficultés d'utilisation du langage diminuent rapidement. La complexité apparente de la production de figures simples par programmation en PostScript, comme j'espère que ce livre le démontrera, est en grande partie une illusion. Et la quantité de travail nécessaire pour produire des chiffres plus compliqués ne sera généralement ni plus ni moins que ce qui est nécessaire.

Lire Démarrer avec PostScript en ligne: https://riptutorial.com/fr/postscript/topic/5616/demarrer-avec-postscript

Chapitre 2: La construction du chemin

Examples

Dessiner (décrire) un polygone

Cet exemple tente de reproduire le comportement des opérateurs de construction de chemin intégrés tels que `arc` .

S'il y a un point actuel, `poly` trace d'abord une ligne vers (x, y) + (r, 0), sinon il commence par se déplacer vers ce point.

Au lieu de `gsave` ... `grestore` (qui a pour effet indésirable de supprimer les modifications que nous souhaitons), il enregistre une copie de la matrice de transformation actuelle (CTM) telle qu'elle existe au démarrage de la fonction.

Ensuite, il fait un `lineto` à chaque point suivant, ce qui en `lineto` à l'échelle et en tournant la matrice est toujours à (0,1). Enfin, il appelle `closepath` puis restaure la matrice enregistrée en tant que CTM.

```
% x y n radius  poly  -
% construct a path of a closed n-polygon
/poly {
    matrix currentmatrix 5 1 roll  % matrix x y n radius
    4 2 roll translate             % matrix n radius
    dup scale                      % matrix n
    360 1 index div exch           % matrix 360/n n
    0 1 {lineto currentpoint moveto}stopped{moveto}if   % start or re-start subpath
    {                              % matrix 360/n
        dup rotate                 % matrix 360/n
        0 1 lineto                 % matrix 360/n
    } repeat                % matrix 360/n
    pop                     % matrix
    closepath               % matrix
    setmatrix               %
} def
```

Itérer à travers un chemin

Cet extrait extrait le contenu du chemin actuel vers stdout. Il utilise la procédure ghostscript `=only` qui peut ne pas être disponible sur tous les interpréteurs. Une procédure équivalente sur les interpréteurs Adobe est appelée `=print` .

`pathforall` est un opérateur de boucle qui prend 4 corps de procédure comme arguments appelés pour les types spécifiques d'éléments de chemin, le résultat de `moveto` , `lineto` , `curveto` , `closepath` et tous les autres opérateurs de construction de chemin qui se résument à ces éléments.

```
{ exch =only ( ) print =only ( ) print /moveto =}
{ exch =only ( ) print =only ( ) print /lineto =}
```

```
{ 6 -2 roll exch =only ( ) print =only ( ) print
  4 2 roll exch =only ( ) print =only ( ) print
  exch =only ( ) print =only ( ) print /curveto =}
{ /closepath = }
pathforall
```

Papier graphique

```
/in {72 mul} def
/delta {1 in 10 div} def
/X 612 def
/Y 792 def
0 delta Y {
    0 1 index X exch  % i 0 X i
    moveto exch       % 0 i
    lineto
    stroke
} for
0 delta X {
    0 1 index Y  % i 0 i Y
    moveto       % i 0
    lineto
    stroke
} for
showpage
```

Lire La construction du chemin en ligne: https://riptutorial.com/fr/postscript/topic/6679/la-construction-du-chemin

Chapitre 3: La gestion des erreurs

Syntaxe

- { *-code-* } arrêté { *-error-* } { *-no-error-* } ifelse% erreur capture d'image
- $ error / errorname get% stackunderflow typecheck rangecheck etc
  $ error / newerror get% bool. mettre faux pour désactiver l'erreur
  $ error / ostack obtient% copy de la pile d'opérande au point d'erreur
- errordict / stackoverflow { *-additional-code-* / stackoverflow signalerror}
  % exécuter du code supplémentaire sur les types d'erreurs (ici, l'erreur / stackoverflow).

Remarques

La gestion des erreurs en postscript comporte deux niveaux. Cette dichotomie s'applique à la fois à la manière dont l'interprète gère les erreurs et aux moyens dont dispose l'utilisateur (le programmeur) pour contrôler le traitement.

Le niveau inférieur est un `stop ... stopped` structure de contrôle inhabituel `stop ... stopped` . `stopped` se comporte un peu comme une construction en boucle dans la mesure où il établit une marque sur la pile d'exécution à laquelle on peut accéder si l'opérateur de `exit` (pour une boucle) ou l'opérateur d' `stop` (pour un contexte `stopped`) est appelé. Contrairement à une construction en boucle, `stopped` génère un booléen sur la pile indiquant si l' `stop` été appelé (sinon, la procédure passée à `stopped` est connue pour s'être achevée).

Lorsqu'une erreur PostScript se produit, comme `stackunderflow` peut-être, l'interpréteur recherche le nom de l'erreur dans `errordict` qui vit dans `systemdict` . Si l'utilisateur n'a pas remplacé la procédure dans `errordict` , la procédure d'erreur par défaut prendra des instantanés de toute la pile et les placera dans `$error` , un autre dictionnaire dans `systemdict` . Enfin, la procédure par défaut appelle `stop` ce qui fait sortir le programme utilisateur de la pile exec et exécute la procédure d'impression d'erreur de l'interpréteur appelée `handleerror` en `errordict` .

Donc, en utilisant toutes ces connaissances, vous pouvez *intercepter des* erreurs en encapsulant une section de code dans `{ ... } stopped` . Vous pouvez *relancer* une erreur en appelant `stop` . Vous pouvez déterminer le type d'erreur survenue avec `$error /errorname get` .

Vous pouvez également modifier le comportement par défaut pour un type d'erreur spécifique en remplaçant la procédure portant ce nom par `errordict` . Ou changez le format d'impression du rapport d'erreur en remplaçant `/handleerror` dans `errordict` .

Examples

Y a-t-il un point actuel?

Rendement `true` sur la pile si le point `currentpoint` s'exécute avec succès, ou `false` s'il signale une erreur `/nocurrentpoint` .

```
{currentpoint pop pop} stopped not  % bool
```

Séquence d'événements lorsqu'une erreur est signalée

La séquence d'une erreur est généralement:

1. L'erreur est déclenchée en recherchant le nom de l'erreur en `errordict` et en exécutant cette procédure.
2. la procédure d' `errordict` appelle l'appel de `signalerror` , en lui passant le nom de l'erreur.
3. `signalerror` prend des instantanés des piles, en enregistrant les instantanés dans `$error` , puis les appels `stop` .
4. `stop` ouvre la pile exec jusqu'au contexte clos le plus proche établi par l'opérateur arrêté.
5. Si le programme n'a pas établi son propre contexte arrêté pour détecter l'erreur, il sera attrapé par un niveau externe `stopped { errordict /handleerror get exec } if` le code de démarrage l'avait appelé pour placer l'ensemble du programme utilisateur.
6. `handleerror` utilise les informations de `$error` pour imprimer un rapport d'erreur.

Signaler (lancer) une erreur

La plupart des outils sont standardisés à l'exception du nom de l'opérateur pour lancer une erreur. Dans Adobe interpreters, il s'appelle `.error` . Dans ghostscript, cela s'appelle `signalerror` . Ainsi, avec cette ligne, vous pouvez utiliser `signalerror` dans le code postscript pour les interpréteurs Adobe, ghostscript ou xpost.

```
/.error where {pop /signalerror /.error load def} if
```

nom de commande errorname **signalerror** -
Prenez des instantanés de la pile en `$error` , puis `stop` .

Par exemple.

```
% my proc only accepts integer
/proc {
    dup type /integertype ne {
        /proc cvx /typecheck signalerror
    } if
    % ... rest of proc ...
} def
```

Attraper une erreur

Comme l'action finale du gestionnaire d'erreurs par défaut consiste à appeler `stop` , vous pouvez intercepter les erreurs des opérateurs en insérant du code dans une construction `{ ... } stopped` .

```
{
```

```
    0 array
    1 get
} stopped {
    $error /errorname get =
} if
```

va imprimer " `rangecheck` ", l'erreur signalée par `get` lorsque l'index est en dehors de la plage acceptable pour ce tableau.

Re-lancer des erreurs

Cet extrait implémente une procédure qui se comporte comme un opérateur de boucle PostScript. Si l'utilisateur `proc` appelle la `exit` , il attrape la `invalidexit` erreur de fixer le dictstack pour la `end` à la fin. Toute autre erreur, sauf `invalidexit` est renvoyée en appelant `stop` .

```
% array n proc  .  -
% Like `forall` but delivers length=n subsequences produced by getinterval
/fortuple { 4 dict begin
    0 {offset proc n arr} {exch def} forall
    /arr load length n idiv
    {
        {
            /arr load offset n getinterval
            [ /proc load currentdict end /begin cvx ] cvx exec
            /offset offset n add def
        } stopped {
            $error /errorname get /invalidexit eq
            { 1 dict begin exit }{ stop } ifelse
        } if
    } repeat
end
} def

%[ 0 1 10 {} for ] 3 {} fortuple pstack clear ()=
```

Lire La gestion des erreurs en ligne: https://riptutorial.com/fr/postscript/topic/6199/la-gestion-des-erreurs

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec PostScript	Community, Kurt Pfeifle, luser droog
2	La construction du chemin	luser droog
3	La gestion des erreurs	luser droog