Tutoriel pour l'utilisation du package *PackageCinema.sty*

Ludovic Ducobu

12 & 14 août 2016
Dernière modification : 29 juin 2017

Table des matières

1. Introduction

LaTeX est un outil formidable pour les maniaques de typographie. Il offre un grand nombre d'outils permettant de définir simplement [1] et de manière automatisée la mise en page d'un document.

Toute la puissance de LaTeX se révèle lorsque l'on prend le peine de se familiariser avec les *packages*. Les packages sont des fichiers annexes dans lesquels on peut définir une fois pour toute un certain nombre de commandes qui pourront être utilisées dans n'importe quel fichier "*.tex*" pour autant que l'on y appelle le package au moyen de la commande "\usepackage{NomDuPackage}" dans le préambule du document "*.tex*".

Dans ce tutoriel nous présentons les différents objets définit dans le package "*PackageCinema.sty*". Ce package à été développé par Ludovic Ducobu grâce aux conseils et aux indications d'Antoine Brandelet entre le 5 et le 14 août de l'année 2016 afin de donner les outils nécessaires à la rédaction du script d'un film avec LaTeX.

`Dernière modification du package : 29 juin 2017`

2. Installation et utilisation

2.1. Installation

Pour pouvoir utiliser le package, il suffit de disposer du fichier "*PackageCinema.sty*" et de le placer dans le même dossier que le fichier "*.tex*" dans lequel il est appelé [2]. Il suffit alors d'entrer

\usepackage{PackageCinema}

dans le préambule du document pour disposer de tous les outils du package.

2.2. Utilisation

Le package a été écrit pour que les commandes soient utilisées au sein d'un document de la classe "*report*" ou "*book*" il est donc vivement conseillé de n'utiliser le package que dans des documents commençant par

\documentclass{report}
ou
\documentclass{book}.

Le package a été écrit pour la rédaction de scripts en français et, par conséquent, "*PackageCinema.sty*" importe automatiquement les packages *inputenc* avec l'option "utf8"et *babel* avec l'option "français".

Les packages *amsmath*, *amsfonts*, *amssymb*, *amsthm*, *braket*, *graphicx* et *color* sont également importés pour les petits malins qui voudraient écrire des textes étranges remplis de symboles mathématiques et/ou avec des indications en couleur.

Pour des raisons techniques les packages *fancyhdr*, *etoolbox* et *refcount* sont également importés.

Dès lors, sauf demande spéciale, vous ne devriez avoir besoin d'importer que "*PackageCinema.sty*" pour la rédaction de votre script.

Le seul package dont vous pourriez avoir besoin et qui ne soit pas importé d'office lors de l'importation de "*PackageCinema.sty*" est le package *geometry* qui vous permet de redéfinir les marges de votre document. Ce package a volontairement été ignoré afin de laisser l'utilisateur libre de définir lui même les marges de son document.

1. Pour qui a des bases en programmation
2. Si LaTeX est correctement installé, on peut aussi le placer dans l'arborescence de LaTeX pour pouvoir l'appeler dans n'importe quel dossier. Il faut reconnaitre que placer le fichier dans le même dossier que le fichier "*.tex*" est une méthode d'installation en soi plus simple mais un peu "moins propre" sur le plan de la programmation. Cependant cette méthode fonctionnera toujours.

3. Les bases de *PackageCinema.sty*

Dans cette section, nous présentons les outils du package.

3.1. L'environnement *script*

Cet environnement est le plus important du package. C'est lui qui définit ce qui dans le fichier "*.tex*" est dans le script à proprement parler et ce qui ne l'est pas. Son utilisation est la même que pour tout environnement :

```
\begin{script}
    Le scénario du film...
\end{script}
```

L'ouverture d'un environnement *script* s'accompagnera automatiquement d'un saut de page.

À ce stade une remarque s'impose : Nous avons volontairement choisis de définir le script comme un environnement et non de définir une nouvelle classe de document. Le but de cette manœuvre était de faire en sorte que le package offre l'opportunité d'insérer un nombre arbitraire de scripts différentes au sein d'un seul fichier "*.tex*" [3].

3.2. La commande *titrescript*

ATTENTION : Cette commande est en fait une "sous-commande" de l'environnement *script*, elle ne peut donc être appelée qu'à l'intérieur dudit environnement !

Cette commande permet de définir le titre du scénario en cours. Elle est à utiliser à l'endroit où l'on veut que le titre s'affiche (il est cependant conseillé de mettre le titre dès l'ouverture de l'environnement *script*).

La commande prend évidement un argument : Le titre du script en cours. En entrant le code :

```
\titrescript{La nuit tous les chats sont gris}
Le texte...
```

on obtiendra dans le fichier compilé :

La nuit tous les chats sont gris

Le texte...

Notons que le fait de devoir utiliser une autre commande que \title pour définir le titre d'un script provient de notre souhait de permettre à l'utilisateur d'en inclure plusieurs au sein d'un même fichier "*.tex*". La commande \title définit le titre du document "*.tex*" complet tandis que *titrescript* définit localement le titre du script se trouvant à l'intérieur de l'environnement *script* en cours. Si vous incluez différents scripts au sein d'un même fichier "*.tex*", vous pourrez ainsi répéter l'utilisation de la commande *titrescript* dans chaque environnement *script* sans qu'il n'y ai de contradiction entre les différents scripts ni avec le titre du document "*.tex*" (qui pourrait être "Recueil de mes films à moi" [4]).

3. Pour les utilisateurs souhaitant regrouper plusieurs de leurs films au sein d'un même ouvrage tout en conservant une numérautation des pages cohérente, par exemple.

4. ou quelque chose d'approchant...

3.3. La commande *auteurscript*

À moins que vos ne teniez à rentrer dans le rang des illustres anonymes n'ayant pas signé leur chef-d'œuvre, il est toujours intéressant de pouvoir attribuer la paternité d'un texte. C'est là qu'intervient la commande *auteurscript*.

Cette commande prend un argument : le nom du (ou des) auteur(s) du scénario.

ATTENTION : 1. Contrairement à la commande *titrescript* qui est définie **localement** (c.à.d. uniquement à l'intérieur d'un environnement *script*), la commande *auteurscript* est définie **globalement** (autrement dit vous pouvez l'utiliser même hors de tout environnement *script*) ! Cela vous permet, si tous les scripts sont du (des) même(s) auteur(s), de mettre une seule commande *auteurscript* au début du document qui vaudra pour tous les scripts au sein du document "*.tex*". Si certains scripts sont d'un groupe d'auteurs différent des autres (vous avez écrit 30 scripts seuls mais 1 avec un ami), il vous suffit alors de rappeler la commande pour modifier le nom du ou des auteurs pour tous les scripts qui suivront (jusqu'à l'apparition éventuelle d'une nouvelle commande *auteurscript*).

2. Si vous voulez définir un nom d'auteur, il faut appeler la commande *auteurscript* **AVANT** la commande *titrescript*, autrement le nom de l'auteur ne s'affichera pas (ou pas correctement). Cela provient du fait que nous avons construit la commande *titrescript* pour qu'elle "combine les effets" de la commande *title* et de la commande *maketitle* pour un document "*.tex*". Autrement dit, lorsque vous utilisez la commande *titrescript*, vous définissez le titre de la pièce mais vous ordonnez aussi au package d'afficher les options de titre du script en cours (le titre et l'auteur). Si nous avons pris cette liberté, c'est simplement pour ne pas avoir à définir une commande "*maketitlescript*". Pourquoi faire avec trois commandes ce que l'on peut faire avec deux [5] ?

Le code

```
\auteurscript{moi}
\titrescript{La nuit tous les chats sont gris}
                    Le texte...
```

donnera après compilation :

La nuit tous les chats sont gris

moi

Le texte...

3.4. La commande *scene*

ATTENTION : Cette commande est en fait une "sous-commande" de l'environnement *script*, elle ne peut donc être appelée qu'à l'intérieur dudit environnement !

Cette commande permet de définir le début des différentes scènes d'un scénario donnée. Pour tout scénario qui se respecte, l'ouverture d'une scène doit s'accompagner de trois informations :

1. Le style de décors (intérieur, extérieur, intérieur et extérieur dans la même scène)
2. Le lieu où se déroule l'action (exemple : une forêt, la maison d'un des personnages, un ville, ...)

5. en faisant juste un peu attention à l'ordre dans lequel on les utilise...

3. Le moment de la journée où la scène se déroule.

La commande *scene* prend donc trois arguments correspondant respectivement à ces trois informations.

ATTENTION : Afin d'uniformiser au maximum le rendu final, le contenu du premier argument de la commande est réglementé ! ! !

Explication : Le premier argument de la commande sert à définir si l'action se situe en intérieur, en extérieur ou "un peu des deux", les règles typographiques font que cela doit se stipuler sous une forme bien précise : "INT." pour intérieur, "EXT." pour extérieur et "INT./EXT." pour le dernier cas. afin de vous éviter d'oublier de ci de là un point, de noter "Inter" pour intérieur, ... le premier argument de la commande *scene* ne peut prendre que trois valeurs : "i", "e" ou "ie". En donnant l'un de ces trois arguments à la commande *scene*, le package saura automatiquement ce qu'il doit générer lors de la compilation entre "INT.", "EXT." et "INT./EXT." . Si vous placez n'importe quoi d'autre en tant que premier argument, le package vous signalera que votre argument n'est pas valable en plaçant [ARG.1 INVALIDE] là où le style de décors aurait dû apparaitre.

Pour le troisième argument la situation est similaire – mais malgré tout un peu différente. Le moment de la journée où l'action à lieu peut-être soit le matin, soit en plein jour, soit le soir, soit de nuit cependant dans certains cas, on peut utiliser ce troisième argument pour donner "l'ambiance générale de la scène" [6]. Afin de s'adapter à ces exigences, le troisième argument de la commande n'est pas réglementé MAIS, comme dans la majorité des cas on préférera les indications jour, nuit, matin ou soir (plus sobres), nous avons implémenté quatre raccourcis : "j", "n" ,"m" et "s" correspondant respectivement à jour, nuit, matin et soir. En entrant simplement l'une de ces quatre lettres en guise de troisième argument, le package générera automatiquement le mot souhaité vous évitant ainsi de faire une faute de frappe en écrivant "nuit" pour la trentième fois. Si vous placez quoique ce soit d'autre en tant que troisième argument, le package le restituera tel quel après compilation.

Le deuxième argument quant à lui n'est pas réglementée et ne comporte aucun raccourci ; le nom du lieu où se déroule l'action pouvant être à peu près n'importe quoi "forêt", "maison de John Doe", "mairie", ...

Lors de l'appel de cette commande via

```
\scene{ie}{forêt}{j}
le texte de la scène...
```

vous obtiendrez après compilation :

1 INT./EXT. FORÊT – JOUR

le texte de la scène...

Lorsque cette commande est utilisée plusieurs fois de suite, les scènes se numérotent automatiquement. Le code :

```
\scene{i}{forêt}{j}
le texte de la scène 1...
\scene{int}{maison}{aux premières lueurs de l'aube}
le texte de la scène 2...
\scene{e}{rue}{n}
le texte de la scène 3...
```

donnera après compilation :

6. Par exemple, pour marquer le fait que la scène à lieu dans un environnement opaque et stressant on pourra préférer écrire quelque chose comme "aube brumeuse" au lieu de "matin".

1 INT. FORÊT – JOUR

le texte de la scène 1...
2 [ARG.1 INVALIDE] MAISON – AUX PREMIÈRES LUEURS DE L'AUBE

le texte de la scène 2...
3 EXT. RUE – NUIT

le texte de la scène 3...

3.5. La commande *replique*

ATTENTION : Cette commande est en fait une "sous-commande" de l'environnement *script*, elle ne peut donc être appelée qu'à l'intérieur dudit environnement !

Pouvoir numéroter les différentes scènes de son texte c'est bien mais pouvoir remplir lesdites scènes, c'est mieux. La commande *replique* permet d'avoir un style uniforme pour l'encodage des répliques des personnages tout au long du scénario.

Deux arguments obligatoires

Cette commande prend, par défaut, deux arguments : Le nom du personnage qui parle et sa réplique. Par exemple, le code :

```
\replique{Marc}{Bonjour.}
\replique{Mathieu}{Salut !}
\replique{Marc}{Beau temps aujourd'hui...}
```

donnera :

MARC

Bonjour.

MATHIEU

Salut !

MARC

Beau temps aujourd'hui ...

Un argument optionnel

Il est parfois intéressant de spécifier l'intention d'un personnage au début de sa réplique au moyen d'une didascalie. Comme, dans un scénario, le seul endroit où l'on peut utiliser une didascalie est à l'ouverture d'une réplique, nous avons incorporé l'ajout d'une didascalie comme un argument optionnel de la commande *replique*. Comme tout argument optionnel, celui-ci se spécifie en premier et entre crochets [7]. Par exemple, le code :

7. et non entre accolades comme un argument obligatoire

```
\replique[Sur un ton maussade]{Marc}{Bonjour.}
\replique{Mathieu}{Salut !}
\replique{Marc}{Beau temps aujourd'hui...}
```

donnera :

MARC
(Sur un ton maussade)
Bonjour.

MATHIEU
Salut !

MARC
Beau temps aujourd'hui ...

après compilation.

ASTUCE : En définissant dans le préambule des commandes \Marc et \Mathieu comme

```
\newcommand{\Marc}[2][]{\replique[#1]{Marc}{#2}}
\newcommand{\Mathieu}[2][]{\replique[#1]{Mathieu}{#2}}
```

vous gagnerez du temps et de la clarté lors de la rédaction puisque vous pourrez écrire

```
\Marc[Sur un ton maussade]{Bonjour.}
\Mathieu{Salut !}
\Marc{Beau temps aujourd'hui...}
```

pour obtenir le résultat ci-dessus après compilation.

INSISTONS sur la présence de crochets **VIDES**[8] à côté du nombre d'arguments dans la définition des commandes \Marc et \Mathieu ci-dessus : Ces crochets signifient pour LaTeX "*la commande que je suis en train de définir possède deux arguments dont l'un*[9] *est optionnel. Sa valeur pas défaut est ...*" Le fait que les crochets soient vides signifie que, par défaut, l'argument optionnel est la chaine de caractère vide. Il est très important pour le package qu'il en soit ainsi ! En effet, la commande *replique* fait une distinction entre le cas où l'argument optionnel est vide et celui où il ne l'est pas afin d'ajuster la mise en page. Lorsque l'argument optionnel est vide, le package sait qu'il n'y a pas de didascalie et ne fait rien. En revanche, le package interprète toute chaine de caractère non vide comme la présence d'une didascalie. Dans ce cas de figure, le package placera le contenu de l'argument optionnel sur une ligne à part et entre parenthèses après compilation. Il est donc primordial, pour que les commandes \Marc et \Mathieu fonctionnent comme la commande *replique*, que la valeur par défaut de l'argument optionnel de ces commandes soit la même que celle de la commande *replique* : la chaine de caractère vide ! ! !

REMARQUE : La règle veut que lorsqu'une réplique dépasse sur plusieurs pages, "[...]" s'affiche au bas de la page et que le nom du personnage soit suivit de "[SUITE]" en haut de la page suivante. "*PackageCinema.sty*" génèrera cela pour vous automatiquement chaque fois que ce sera nécessaire cependant, comme pour la table des matières, il faut deux compilations pour que tout s'affiche correctement. "*PackageCinema.sty*" à besoin d'une première compilation pour "repèrer les répliques qui dépassent". Lors de la seconde compilation, l'information précédement stockée[10] est "restituée".

8. sans espace les séparant
9. ce sera toujours le premier
10. Cette information est stockée dans un fichier annexe portant l'extension "*.cinema*".

3.6. La commande *action*

Maintenant que vous êtes à même de faire parler vos personnages et de fixer leur intention au moyen d'une didascalie bien choisie, vous souhaiteriez surement pouvoir décrire leurs déplacements. La commande *action* est là pour ça !

Cette commande prend un argument : la description d'une action (ou de plusieurs actions consécutives). Le code :

```
\scene{ie}{forêt}{j}
\action{Et là le monsieur il se lève et puis il va prendre sa tasse de café et même qu'il la bois.}
```

donnera après compilation un texte aligné à droite avec le début de la scène :

1 INT./EXT. FORÊT – JOUR

Et là le monsieur il se lève et puis il va prendre sa tasse de café et même qu'il la bois.

3.7. Les commandes *transition* et *valplan*

Dans tout bon film, en plus d'être savamment découpée en scène, l'action est entrecoupée de différents types de transitions (noir, fondu au noir, coupure, ...).

On a également besoin de spécifier quelle sera la "valeur de plan" (si le plan est serré, large, ...).

Les règles typographiques font que ces deux d'informations, doivent être centrées à droite et en majuscule. C'est ce que font les commandes *transition* et *valplan*. Par exemple : le code

```
\transition{cut to}
\valplan{pl}
```

donnera :

CUT TO

PL

REMARQUE : En fait, pour les raisons énoncées ci-dessus les commandes *transition* et *valplan* font exactement le même travail. Nous avons choisis de "dédoubler" la commande afin de rendre le code que vous écrirez le plus agréable possible à relire. La règle typographique est la même mais, du point de vue du scénario, une transition n'est pas la même chose qu'une valeur de plan. L'existence de ces deux commandes portant des noms les plus explicites possibles permettra ainsi à votre code de "refléter au mieux" ce que vous voulez écrire.

3.8. La commande *musique*

D'une manière générale, il est déconseillé d'indiquer au sein d'un script la musique "extra-diégètique" [11] sauf dans le cas où elle poursuit un but précis. Pour ce cas ce figure, c'est sans grande originalité que nous avons créé la commande *musique* !

Cette commande prend un argument : ce que vous voulez dire à propos de la musique. (son titre, l'ambiance qu'elle amène,...)

Le code :

```
\musique{C'est de la musique...}
```

Produira après compilation :

MUSIQUE : *C'est de la musique...*

11. En gros, la musique que le spectateur entends mais pas les personnages.

3.9. La commande *CreePersonnage*

La commande

Cette commande a été développée pour sublimer l'utilisation de la commande *replique*. Comme nous l'avons remarqué un peu plus haut, la commande *replique*, bien que très intuitive, présente un "inconvénient" majeur : dans un texte donné, on est amené à faire parler à plusieurs reprises un certain nombre de personnages, toujours les mêmes (les protagonistes de la pièce). Le fait que la commande *replique* requière de manière systématique le nom du personnage qui s'apprête à parler est "désagréable" en ce sens qu'il est plus élégant de définir une commande qui ne prend qu'un argument (le texte de la réplique) et qui affiche automatiquement le nom du personnage choisis (cela évite de commettre une erreur d'inattention en recopiant le nom du personnage pour la nième fois). C'est ce que nous proposions en astuce dans le paragraphe décrivant la commande *replique*. Pour vous éviter d'avoir à faire cette étape par vous même et rendre votre code plus lisible (mais pas seulement !) nous avons développé la commande *CreePersonnage*.

Cette commande prend deux arguments et construit pour vous tous les outils nécessaires pour faire s'exprimer ou pour faire référence à un personnage donné. Le premier argument de la commande est le nom du personnage que l'on est en train de définir. Le second est le nom de la commande par laquelle on fera appel à ce personnage (ces deux arguments sont a priori différents, on peut avoir un personnage au nom très long que l'on souhaite appeler au moyen d'une commande plus courte).

Par exemple, en écrivant

```
\CreePersonnage{Mathieu}{mat}
```

on crée un personnage nommé Mathieu que l'on appellera au moyen de la commande \mat. Après avoir utilisé *CreePersonnage*, une commande a donc été créée (\mat dans notre exemple). Cette commande est particulière car elle peut prendre ZÉRO ARGUMENT OU UN ARGUMENT OBLIGATOIRE ET UN ARGUMENT OPTIONNEL.

Si on ne lui donne aucun argument, la commande considère que l'on est en train de faire référence au personnage (dans une action par exemple). Dans ce cas la commande se contentera d'afficher le nom du personnage.

Si on lui donne un argument, la commande considère que l'on est en train d'ouvrir une réplique du personnage concerné. Dans ce cas le comportement de la commande sera essentiellement celui de la commande *replique* à laquelle on aurait donné le nom du personnage et le texte de la réplique (l'argument donné à la commande générée par *CreePersonnage*). L'argument optionnel sert, comme pour la commande *replique*, à définir une didascalie.

Par exemple, pour un personnage créé comme décrit ci-dessus, le code suivant

```
\mat[Perplexe]{C'est bien compliqué toutes ces commandes \dots}

\action{\mat se gratte la tête.}
\mat{Mais il faut avouer que c'est bien foutu !}
```

donnera après compilation

MATHIEU
(Perplexe)
C'est bien compliqué toutes ces commandes ...

MATHIEU se gratte la tête.

MATHIEU
Mais il faut avouer que c'est bien foutu !

L'AVANTAGE : L'avantage de cette manière de faire est que l'on définit une fois pour toute le nom du personnage. Plus de risque de se tromper en recopiant ET, si d'aventure vous décidiez de changer le nom d'un de vos personnages en cours d'écriture, vous n'auriez qu'à le changer une seule fois au niveau de la création du personnage pour qu'il soit modifié automatiquement dans tout le texte.

Et, fous de LaTeX que nous sommes, on ne s'arrête pas là !

En bonus !

Une "manie" assez répandue chez les acteurs consiste à compter le nombre de répliques du personnage dont ils ont hérité. Et, pour satisfaire cette curiosité dévorante, nos ancêtres en étaient réduits à compter manuellement. Mais grâce à la commande *CreePersonnage* ce temps-là est révolu !

En plus de la commande permettant de faire référence à un personnage donné, *CreePersonnage* crée un compteur, portant le même nom que la commande permettant de référer au personnage, qui comptera le nombre de répliques du personnage. Comme toujours avec les compteurs LaTeX, on affichera le nombre stocké dans le compteur en entrant \theNomDeLaCommande.

Dans notre exemple, on écrira \themat pour savoir combien de répliques ont été prononcées par Mathieu jusque là. En utilisant la commande à la fin du scénario, on aura ainsi naturellement accès au nombre total de répliques que Mathieu.

Et, tant qu'à y être, nous avons également prévu un compteur "totalrepliques" comptant le nombre total de répliques du scénario en cours. En tapant \thetotalrepliques, on aura donc accès au nombre de répliques tous personnages confondus jusqu'à l'endroit où la commande est appelée. En l'utilisant à la fin du script, ici encore, on connaitra le nombre total de répliques de sa prose.

REMARQUE : Pour plus de lisibilité de votre code, l'idéal est de mettre tous les appels à la commande *CreePersonnage* au même endroit, par exemple dans le préambule [12] de votre document "*.tex*".

CEPENDANT, si le document "*.tex*" que vous écrivez regroupe plusieurs scripts il vaut mieux mettre les appels à la commande *CreePersonnage* pour une pièce donnée juste avant de commencer votre script [13]. Cette manoeuvre permet de remettre à zéro les compteurs de répliques pour les personnages apparaissant dans plusieurs scénarios, et ce afin d'éviter d'obtenir un compte des répliques erroné. Dans notre exemple, il faudrait donc remettre \CreePersonnage{Mathieu}{mat} au début de chaque script où le personnage de Mathieu apparait même si le personnage avait déjà été créé précédemment. D'une certaine façon on peut considérer que ce nouvel appel à la commande *CreePersonnage* "réinitialise" le personnage.

Si vous souhaitez définir des compteurs indépendants pour les différentes apparitions d'un personnage au cours de différents scénarios (sans devoir le réinitialiser) il convient de définir des commandes d'appel au personnage différentes pour chaque script. Par exemple, utiliser

\CreePersonnage{Mathieu}{mat}

pour un premier script et

\CreePersonnage{Mathieu}{mat2}

pour un second. De cette façon, les compteurs "mat" et "mat2" seront indépendants mais l'appel à l'une où à l'autre des commandes permettra de faire référence à Mathieu.

N.B : Le compteur "totalrepliques" est automatiquement remis à zéro lors de l'ouverture d'un nouveau script. Il sert donc de compteur du nombre total de répliques pour le script en cours et non de compteur du nombre total de répliques tous scripts confondues si jamais votre document "*.tex*" en comporte plusieurs différents.

12. c.à.d. avant le \begin{document}
13. c.à.d. juste avant le \begin{script}

4. Les options

À ce stade du tutoriel, vous disposez de tous les outils nécessaires à la rédaction de votre scénario. Cependant, dans un soucis de flexibilité du package, quelques options de personnalisation du style de votre texte ont été implémentées. Ce sont ces options que nous présentons dans cette section.

4.1. L'option *suite*

Une règle typographique veut qu'en haut à gauche de chaque page d'un script (excepté la première) on trouve l'indication "[SUITE]" et qu'en bas à droite de chaque page (excepté la dernière) on trouve "[.../...]" afin de signaler que l'on est ni au début ni à la fin du scénario. Cette règle ayant été édictée, nous ne pouvions l'ignorer cependant, à notre sens, l'ajout de ces indications alourdi inutilement la mise en page.

Nous avons dès lors choisi de laisser à l'utilisateur le soin de décider s'il voulait ou non afficher ces informations. Par défaut, *PackageCinema.sty* ne les affiche pas[14] mais le package possède une option *suite* (notez que *Suite* ou *SUITE* fonctionnent également) permettant de forcer leur affichage.

Il s'agit ici d'une option de package, c.à.d. qu'il faut faire appel à cette option dès l'appel du package via le code :

\usepackage[suite]{PackageCinema}.

4.2. Options d'espacement des scènes

Comme nous l'avons mentionné ci-avant, lors de l'appel de la commande *scene*, des espacements sont ajoutés mais l'on continue sur la page en cours (pour autant qu'il reste suffisamment de place).

La commande *ActiveEspaceScene*

Pour ceux que cela dérangerait, pour ceux qui souhaitent espacer leur texte un maximum, pour ceux là[15] nous avons développé une commande très simple : *ActiveEspaceScene*. L'appel de cette commande changera le comportement du package de telle sorte que toutes les scènes (sauf la première afin d'éviter de se retrouver avec une page comportant uniquement le titre de la pièce) débuteront sur une nouvelle page.

La commande *ArreteEspaceScene*

Vous êtes heureux de pouvoir aérer votre texte seulement le sort s'acharne : Une de vos scènes se termine après seulement deux ou trois lignes sur une nouvelle page de telle sorte que que votre texte comporte une page presque vide (ce qui n'est pas génial du point de vue de l'économie de papier). Que faire ? Pas de panique, la commande *ArreteEspaceScene* a été faite pour vous !

Comme son nom le laisse présager, cette commande annule les effets de la commande *ActiveEspaceScene* pour tout le texte situé en dessous d'elle (à moins que l'on ne vienne réinsérer une commande *ActiveEspaceScene*) de telle sorte que toutes les scènes suivantes (jusqu'à un nouvel appel de la commande *ActiveEspaceScene*) apparaîtront sans qu'un saut de page soit effectué.

Notons que l'appel de la commande *ArreteEspaceScene* sans qu'une commande *ActiveEspaceScene* n'ai été appelée au préalable ne provoquera aucune erreur de compilation. Cela sera simplement sans effet.

La combinaison de ces deux commandes vous permettra ainsi de faire en sorte que votre code soit aéré juste ce qu'il faut sans devoir inscrire explicitement \pagebreak au début de 99 scènes sur 100.

14. Si vous envisagez d'envoyer votre scénario à un producteur ou à une maison de production quelconque il est malgré tout conseillé d'afficher ces informations.

15. et pour tous les autres

4.3. Option pour la table des matières

Maintenant que vous savez comment rédiger votre scénario et le personnaliser (presque) suivant la moindre de vos fantaisies, la seule chose sur laquelle vous pourriez encore vouloir de l'influence est la table des matières du document.

Par défaut, le titre de la pièce est affiché dans la table des matières ainsi que les éventuelles scènes qui structurent le texte.

La commande *EffaceSceneTableMatiere*

permet d'enlever la position des différentes scènes de la table des matières.

5. Conclusion

Ici s'achève le tutoriel pour l'utilisation du fihcier "*PackageCinema.sty*". Nous espérons sincèrement que les outils développés dans ce package vous seront autant utiles qu'agréables à utiliser.

Pour toute suggestion d'amélioration, remarque, bugg, mouvement d'humeur ou autre n'hésitez pas à nous contacter via l'adresse mail *ludoducobu@hotmail.com* en spécifiant en objet "*PackageCinema.sty* : l'objet de votre mail".

Il ne nous reste plus pour terminer ce tutoriel qu'à vous souhaiter une bonne route et de nombreuses créations artistiques avec "*PackageCinema.sty*" !!!