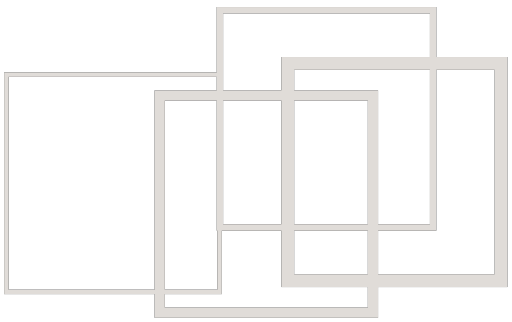
15 septembre 2015

Écrire autrement : de l'écriture graphique à l'écriture computationnelle

@s

http://aswemay.fr/co/010020.html



Stéphane Crozat

Table des matières

Préambule

L'écriture numérique est une écriture graphique, elle procède du signe interprété par l'humain ; mais l'écriture numérique est aussi une écriture computationnelle, qui procède du calcul effectué par l'ordinateur.

Les questions posées par l'écriture numérique le sont à l'ingénieur qui conçoit les instruments d'écriture et de lecture. Quand se reposer sur le graphique, l'héritage culturel, les conditions de l'interprétation ? Quand se reposer sur le computationnel, le traitement, l'automatisation ? Comment articuler les deux dimensions ? Mes ces questions sont aussi - surtout - posées au citoyen, qui écrit, qui lit. Pourquoi verser dans le computationnel ? Est-il possible de faire autrement ? Comment y résister ? Pourquoi, par exemple, s'encombrer du WYSIWYM quand le WYSIWYG paraît si satisfaisant ?

Les outils que nous utilisons pour écrire instrumentent notre communication, notre pensée. Nous avons un intérêt direct à chercher à les comprendre en tant qu'instruments, pour s'en émanciper, éventuellement les maîtriser, au moins n'en être pas totalement esclaves. C'est l'idée du développement d'une littératie numérique.

Ces questions essentielles sont d'ordre méthodologique, c'est à dire que l'enjeu n'est pas de décider ce qu'il convient de faire, où placer les curseurs, défendre l'un contre l'autre. L'enjeu est de poser - et de reposer - ces questions. C'est ce que je nous invite à faire à travers cette série d'articles autour de l'écriture numérique.

Vous pensiez peut être n'en a avoir rien à calculer, et pourtant...

I L'écriture numérique : une inscription intentionnelle codée avec un logiciel pour un ordinateur

L'écriture est une inscription intentionnelle

L'écriture est un enregistrement intentionnel de signes sur un support pérenne en vue de transmettre un contenu.

Définition originelle de l'écriture (graphique)

L'écriture désigne originellement la représentation graphique d'une langue, par la spatialisation sur un support pérenne des sons ou des mots sous la forme de signes. Deux notions sont donc constitutives de l'écriture : la notion de langue qui fixe l'intentionnalité d'exprimer un contenu ; la notion de pérennité du support qui sous-tend la transmission différée de ce contenu.

Définition étendue de l'écriture (multimédia)

Initialement réservé à l'écriture alphabétique et idéographique, le terme a depuis été étendu à d'autres formes d'expressions : écriture cinématographique, radiophonique, voire photographique. Elle devient écriture multimédia dès lors que le support permet de juxtaposer différentes forme sémiotiques, et que la question de l'articulation entre ces différentes formes - l'intersémiotisation - se pose.

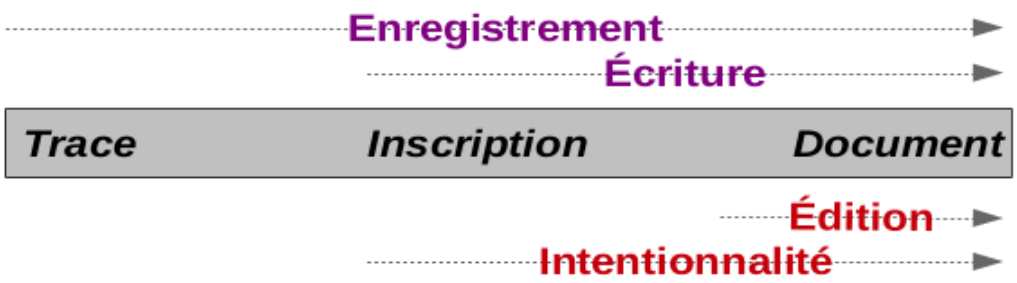
Définition absolue de l'écriture

Certains auteurs admettent un sens encore plus étendu de l'écriture, à tout ce qui fait trace. J'appellerai absolue cette définition, au sens de sans limite ; et je ne l'adopterai pas, car elle revient à supprimer l'intentionnalité de l'écriture (ce qui est contraire à mon hypothèse terminologique de départ).

Trace, inscription, document

On pourra en corollaire définir les concepts associés de trace, inscription et document.

- Une trace est un enregistrement (et non une écriture) non intentionnel de signes (par exemple, une caméra de surveillance enregistre des traces).
- Une inscription est le résultat d'une écriture ; une inscription est donc produite intentionnellement en vue de tenir un propos (par exemple, une vidéo personnelle de vacances cadrée, montée...)
- Un document est une inscription établie dans un contexte éditorial (par exemple, un film commercial distribué sur DVD)



L'écriture numérique est une inscription codée avec un logiciel pour un ordinateur

Le concept d'écriture numérique renvoie en fait à trois composantes du numérique :

1. l'écriture numérique est ce qui est produit avec un dispositif numérique de production des signes (un ordinateur et un logiciel d'écriture) ;
2. l'écriture numérique est ce qui produit des inscriptions qui sont numériques (un code binaire) ;
3. l'écriture numérique est ce qui produit des signes destinés à être interprétés via un dispositif numérique de lecture (un ordinateur et un logiciel interactif).

Écrire avec un logiciel

Toute écriture numérique procède de l'utilisation d'un ordinateur et d'un logiciel d'écriture. Ce dispositif peut tendre à masquer sa dimension numérique, pour simuler un dispositif graphique, comme une machine à écrire, une photocopieuse, une caméra ; ou bien intégrer sa dimension numérique en proposant explicitement des fonctions de calcul : copier/coller, chercher/remplacer, génération automatique d'index...

Fonctions caractéristiques des logiciels d'écriture numérique

Écrire du code

Toute écriture numérique conduit à la production de code qui représente le contenu sous forme de nombres. Cette inscription numérique pourrait n'être envisagée que comme intermédiaire, si la visée de l'écriture était la production exclusive et *one shot* (d'un trait) d'un support matériel graphique, typiquement une impression sur papier ; mais il faudrait, pour se rapprocher de cette situation idéale, qu'il n'y ait pas d'enregistrement de la production ni au cours de l'écriture, ni pour une révision ultérieure (ce qui n'est plus envisager en pratique). Or dès lors qu'il y a enregistrement du code, il y a une gestion informatisée de ce code (j'entends ici enregistrement au sens de stockage dans mémoire secondaire, l'inscription est nécessairement en mémoire primaire, mais l'on peut considérer que son caractère temporaire ne procède pas d'un enregistrement). Donc de fait, dans la majorité des cas l'inscription numérique aura une vie en tant que telle, elle sera gérée (diffusé, copié, manipulée...) en tant que code par les utilisateurs et les systèmes.

Fonctions caractéristiques des logiciels de gestion du contenu numérique

Écrire pour l'ordinateur

Enfin, si l'écriture peut se confiner à la réalisation d'un support graphique imprimé sur papier, elle tend de plus en plus à viser la lecture sur écran, c'est à dire la lecture via l'usage d'un ordinateur, et donc de ses périphériques d'entrée (souris, clavier, écran...) et de sortie (écran, haut parleurs...). Cette écriture marque une troisième rupture en intégrant dans l'acte de lecture des fonctions du numérique : la manipulation du texte via la souris (ou l'écran tactile) et une iconographie, la scénarisation non linéaire, le multimédia, l'extension du texte par les commentaires...

Fonctions caractéristiques des logiciels de publication et de lecture numérique

II L'écriture numérique est une programmation informatique

La dualité écriture-programmation de l'écriture numérique

Toute écriture numérique procède techniquement d'une programmation ; le résultat de l'acte d'écriture avec un ordinateur est un code informatique dont l'exécution produit des signes sur un terminal de lecture qui donnent lieu à des interprétations.

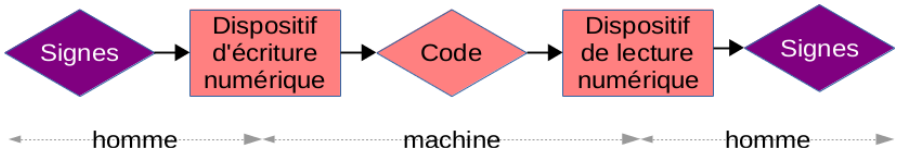
L'écriture numérique et la programmation informatique semblent relever de deux pratiques différentes : l'écriture numérique consisterait à coder des données, les signes que l'on code et que l'on souhaite voir restitués "tels quels" ; tandis que la programmation consisterait à coder les calculs qui manipulent ces signes, ce qui serait de l'ordre du traitement.

Mais cette séparation est fragile, d'abord au niveau logique de l'informatique où la frontière entre données et traitements est poreuses ; mais également au niveau sémiotique de l'écriture où la frontière entre écriture de contenus et écriture de programmes est également difficile à établir systématiquement dans le contexte du numérique.

Ce qu'il reste de la séparation entre écriture et programmation se cristallise dans le degré de conscience et d'intentionnalité de programmer que l'acte d'écrire inclut. Écrire relève d'une programmation, mais celle-ci peut être plus ou moins prégnante sur celle là ; et la maîtrise de l'articulation entre le niveau logique de la programmation et le niveau sémiotique de l'écriture est un enjeu majeur de l'écriture numérique.

Rappel : Écriture numérique et codage de signes

L'écriture consiste en l'inscription intentionnelle de signes en vue d'une transmission et d'une interprétation par un être humain. L'écriture numérique se caractérise par le fait que les inscriptions sont codées via un dispositif d'écriture numérique et décodées via un dispositif de lecture numérique.



1 Les programmes sont des données, les données sont des programmes

Fragilité théorique de la séparation entre données et instructions

La rupture fondamentale apportée par l'ordinateur, qui le distingue des automates qui le précédent est l'indifférenciation du programme et de la donnée, ou dit autrement, le fait de considérer un programme comme une donnée.

« The computer is based on a fixed hardware platform, capable of executing a fixed repertoire of instructions. At the same time, these instructions can be used and combined like building blocks, yielding abitrarily sophisticated programs. Moreover, the logic of the program is not embeded in the hardware, as it was in mechanical computers predating 1930. Instead, the program's code is stored and manipulated in the computer memory, *just like data*, becoming what is known as "software". (Nisan and Schocken, 2005) [fr] »

Séparation entre données et traitements au niveau du langage machine

Si on étudie le fonctionnement bas niveau d'un ordinateur on peut considérer d'un certain point de vue qu'il y a d'un côté le programme qui calcule, et de l'autre les données qui sont calculées ; d'un côté le microprocesseur, de l'autre la mémoire. Il y a bien dans la machine des séquences binaires qui donnent des ordres (les instructions) et d'autres qui paramètrent ces ordres (les données). Ces séquences peuvent être séparées au sein de mémoires différentes dans la machine, pour des raisons pratiques, mais ce n'est pas nécessaire.

Simulation d'une exécution de langage machine avec deux séquences binaires

Cette simulation a été réalisée avec *CPU Emulator, Version 2.5* (). Elle se base sur un ordinateur avec deux mémoires séparées, une pour le programme (ROM) et une pour les données (RAM).

La machine de Turing universelle

L'exemple précédent illustre une machine de Turing, avec d'un côté un programme et de l'autre les données impactées par le programme. Mais on peut également le représenter sous la forme d'une machine de Turing universelle, avec une seule mémoire comportant le programme et les données. Il n'y a plus qu'une seule séquence numérique dont la lecture et l'écriture entraîne des modifications des états mémoires, et par conséquent des périphériques de sortie, comme l'écran. Données et programmes sont alors confondus.

Simulation d'une exécution de langage machine avec une seule séquence binaire

Soit une machine composée d'un microprocesseur et d'une mémoire de 60 bits (5 fois 16 bits) et doté du langage machine Hack (Nisan and Schocken, 2005).

Soit la séquence binaire de 60 bits codant quatre instructions sur 16 bits en langage machine et stockant le résultat sur les 16 derniers bits :

0000 0000 0000 0100 1111 1100 1001 1000 0000 0000 0000 0000 1110 0011 0000 0001 0000 0000 0000 0010

Adresse	Langage machine	Assembleur	Signification
0	0000 0000 0000 0100	@4	Charge la valeur 4 (0100) dans le registre A
1	1111 1100 1001 1000	MD=M-1	Retire 1 à la valeur stockée à l'adresse mémoire stockée dans le registre A (donc à l'adresse 4) et stocke le résultat à la même adresse ainsi que dans le registre D
2	0000 0000 0000 0000	@0	Charge la valeur 0 dans le registre A
3	1110 0011 0000 0001	D;JGT	Si ce qui est stocké dans le registre D est supérieur à 0 alors se repositionne sur l'instruction située à l'adresse mémoire stockée dans le registre A (ici retourne à la première instruction située à l'adresse 0), sinon avance à l'instruction suivante
4	0000 0000 0000 0010	@2	Charge la valeur 2 (0010) dans le registre A

Décodage de la séquence à l'état initial

Bien que le décodage montre des parties qui codent des instructions et d'autres des données, l'exécution de la séquence montre l'indifférenciation binaire entre programme et donnée.

L'exécution va ici modifier l'état des 2 derniers bits de la mémoire (la valeur sera décrémentée deux fois pour finir à zéro) :

0000 0000 0000 0100 1111 1100 1001 1000 0000 0000 0000 0000 1110 0011 0000 0001 0000 0000 0000 0000

2 Données et traitements du point de vue informatique

Fragilité technique de la séparation entre données et traitements

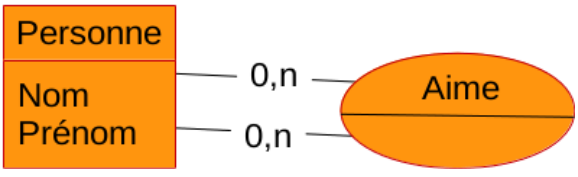
En conception informatique, on a tendance à privilégier la séparation entre données et programmes, notamment afin de pouvoir réutiliser les mêmes programmes pour des données différentes, ou inversement d'exploiter les mêmes données avec des traitements différents pour différents besoins fonctionnels.

Les méthodes de conception repose historiquement sur ce principe, comme la méthode MERISE qui sépare la conception entre le modèle des données et le modèle des traitements. En programmation orientée objet, les classes permettent de séparer les attributs (données) des méthodes (traitements). On retrouve également cette tendance au sein des architectures *n-tier* qui séparent les données stockées sur des serveurs de bases de données et les traitements implémentés au sein de serveurs applicatifs.

Néanmoins, au sein de certains paradigmes de programmation, cette posture n'est pas toujours aussi évidente. Par exemple en Lisp (programmation fonctionnelle) tout est fonction, donc les fonctions savent manipuler des fonctions, permettant la modification dynamique des programmes. De même en JavaScript (programmation orientée prototype), données et fonctions sont des objets dynamiques (variables). Ainsi les données peuvent être traitées comme des instructions et les instructions comme des données.

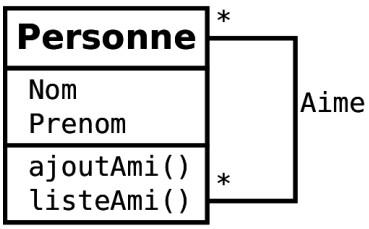
Par ailleurs, il existe des programme qui sont leurs propres données d'entrée, qui s'exécutent toujours de la même façon (les programmes d'amorçage) ; ou des programmes dont le résultat (la donnée de sortie) est un autre programme (c'est ce que fait un compilateur).

Exemple : Entité-Association (MERISE)



Un modèle conceptuel de données dans le formalisme MERISE ne représente que les données, les traitements sont représentés dans un autre modèle.

Exemple : Diagramme de classes (UML)



Un modèle conceptuel de données dans le formalisme objet d'UML représente les données (attributs) ainsi que les traitements principaux (méthodes).

Exemple : Classes Java

```java
public class Personne {
    //Attributs de la classe Personne
    String nom;
    String prenom;
    ArrayList<Personne> amis = new ArrayList<Personne>();
    //Méthodes de la classe Personne
    void ajoutAmi (Personne pAmi) {
        this.amis.add(pAmi);
    }
    String listeAmis() {
        String vAmis = "";
        for (Personne a : this.amis) {
            vAmis+=a.nom;
        }
        return vAmis;
    }
}
```

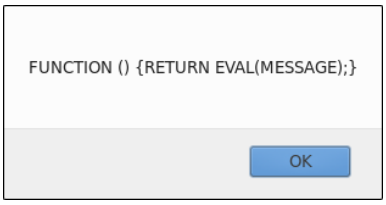
Un langage orienté objet comme Java procède de la programmation de classes qui contiennent des attributs (données) et des méthodes associées (traitements).

Exemple : Architecture 3-tier

Une architecture 3-tier sépare le stockage des données et le traitement des données au sein de deux serveurs distincts.

Exemple : Données et fonctions en JavaScript

// La fonction dialogue est déclarée comme une donnée var debut = "hello"; var fin = "world"; var dialogue = function() {return debut + ' ' + fin;}; alert(dialogue());	
//La donnée message est utilisée comme une instruction de concaténation var debut = "hello"; var fin = "world"; var message = "debut + ' ' + fin"; var dialogue = function() {return eval(message);}; alert(dialogue());	

//La fonction dialogue peut être manipulée comme une donnée
var debut = "hello";
var fin = "world";
var message = "debut + ' ' + fin";
var dialogue = function() {return eval(message);};
dialogue = dialogue.toString().toUpperCase();
alert(dialogue);
	

// La donnée message peut être manipulée pour modifier l'instruction qu'il représente var debut = "hello"; var fin = "world"; var message = "debut + ' ' + fin"; var dialogue = function() {return eval(message);}; message = message + ".toUpperCase()"; alert(dialogue());	

3 Écrire avec la conscience de programmer

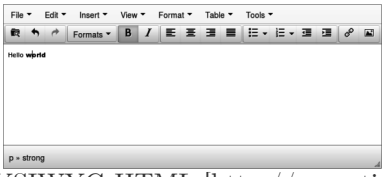
Programmation explicite et programmation implicite

La programmation informatique a pour finalité la production de calculs. Mais tous les programmes informatiques qui interagissent avec des êtres humains le font par l'intermédiaire de signes qu'ils produisent et qui permettent l'interprétation du message par l'humain. Un tel acte de programmation remplit donc la même fonction qu'un acte d'écriture.

L'écriture numérique a de son côté pour finalité la production de signes. Mais elle ne peut le faire que par l'intermédiaire des calculs opérés par la machine. Elle peut procéder pour cela d'un codage logique d'instructions informatiques, c'est à dire d'une programmation explicite, ; ou bien de manipulations de si haut niveau qu'elles masquent le caractère programmatique de l'acte (on parlera d'une programmation implicite).

L'écriture numérique telle qu'elle est proposée via les dispositifs de type traitement de texte relève essentiellement d'une programmation implicite. Mais la programmation d'une macro au sein d'un tel logiciel devient un acte de programmation explicite.

Exemple : production de signes par programmation explicite et par programmation implicite

Éditeur WYSYWYW	 Éditeur WYSIWYG HTML [http://www.tinymce.com]	cf. hw-html.eWeb
Écriture HTML	<html> <body> Hello world </body> </html>	cf. hw-html.eWeb

Écriture JavaScript	<html>
	<body> </body>
	<script type="text/javascript">
	var p = document.getElementsByTagName("p")[0]
	var s = document.createElement("strong");
	var h = document.createTextNode("Hello ");
	var w = document.createTextNode("world");
	p.appendChild(h);
	p.appendChild(s);
	s.appendChild(w);
	</script>
	</html>
		cf. hw-js.eWeb

Programmation de bas niveau et programmation haut niveau

Tout acte de programmation par un être humain se fait par l'intermédiaire de signes qu'il manipule et qui produisent les séquences binaires en machine. La manipulation de ces signes peut se faire avec des langages de plus ou moins bas niveau, c'est à dire plus ou moins proche du langage élémentaire de la machine. Les langages de plus bas niveau sont les assembleurs qui correspondent quasiment aux instructions binaires, puis les langages procéduraux (Basic, Fortran, C, Pascal) qui restent logiquement proche du fonctionnement de l'ordinateur. Il existe ensuite des langages de plus haut niveau, qui s'éloignent de la machine, pour correspondre à des modes d'expression plus abstraits : les langages fonctionnels (Lisp), logique (Prolog), mathématiques (Matlab), orientés objets (Java), orientés données (SQL)... Il existe également des modes de programmation graphique, c'est à dire que le code est produit par la manipulation d'une interface graphique via la souris, par exemple le langage orienté données QBE, les logiciels de création multimédia (Director), les environnements graphiques de développement d'interfaces (WindowBuilder d'Eclipse), les logiciels d'ETL (Pentaho Data Integration)...

4 Écrire avec l'intention de programmer

Fragilité sémiotique de la séparation entre contenus statiques et comportements dynamiques

Du point de vue de l'auteur qui écrit, on peut considérer en première approximation qu'il y a deux catégories d'actes d'écriture numérique : les actes dont la fonction est de coder des signes (je viens d'encoder "je viens d'encoder") et ceux dont la fonction est de coder un programme qui va générer des signes et de l'interaction (par exemple associer un lien hypertexte à un mot). Cette différence se manifeste *grosso modo* d'un point de vue fonctionnel : d'un côté j'entre les signes qui seront restitués, le calcul mobilisé est implicite, il procède essentiellement du stockage et de la restitution ; de l'autre j'ajoute à ces signes des instructions qui vont

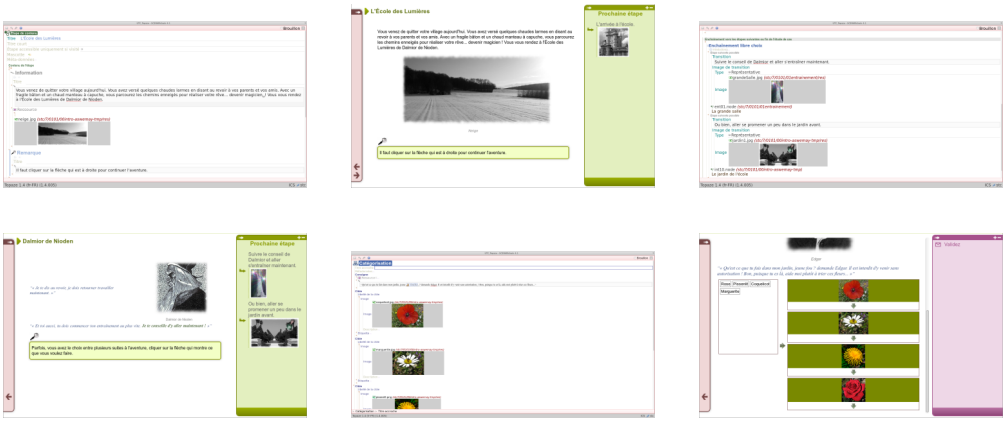
commander du calcul, car je programme explicitement un comportement.

La question porte donc un curseur sémiotique-logique, mais il n'y a pas de frontière - la plupart des écritures numériques renvoient à des modes de codages sémiotique et logique et à des codes à vocation sémiotique et logique. Et le développement du numérique (ses tropismes), tend à rendre la zone de flou de plus en plus large. Dès qu'on n'écrit plus strictement pour l'impression, mais pour l'écran, on programme des interactions, même mineures, comme l'apparition d'une barre de défilement qui dépend de la longueur d'un texte, ou le plan interactif d'un document PDF dont on aura préalablement marqué les niveaux de titre.

Exemple : contenus statiques et de comportements dynamiques

Cette illustration est issue d'un usage de la chaîne éditoriale Scenari/Topaze. Elle montre un entremêlement entre écriture de contenu statique et programmation de comportements dynamiques, typique d'usages avancés du support numérique.

Le contenu complet peut être consulté ici : .



5 Conclusion : intentionnalité et conscience programmatique de l'écriture numérique

L'écriture numérique doit être comprise comme composition entre le monde du sémiotique et le monde du logique. Donc l'enjeu n'est pas d'opposer écriture et programmation, mais de caractériser l'acte d'écriture numérique par la conscience et l'intention qu'a ou non son auteur de programmer.

Les dispositifs d'écriture numérique, et par conséquent les pratiques d'écriture numérique, jouent sur ces deux paramètres pour proposer différents modes d'écriture. D'un côté, ceux qui s'inscrivent largement dans le paradigme du graphique, par l'intermédiaire du sémiotique, mobilisé pour produire les signes et les transporter de façon traditionnelle. De l'autre, ceux qui s'inscrivent plus profondément dans le paradigme du computationnel, par l'intermédiaire du calcul explicitement utilisé pour coder des signes et des comportements.

	Intention de coder des contenus statiques	Intention de coder des comportements dynamiques
Codage par programmation implicite	Sémiotique Sémiotique	Sémiotique Logique
Codage par programmation explicite	Logique Sémiotique	Logique Logique

La dualité écriture-programmation de l'écriture numérique est donc fondamentale : la notion d'écriture renvoie au signe, qui ne peut se passer de calcul dans le contexte du numérique ; la notion de programmation, c'est l'inverse, elle renvoie au calcul qui ne peut se passer de signe dès lors qu'il vise une interprétation humaine.

III Écriture numérique traditionnelle et écriture numérique computationnelle

Toute écriture procède d'une programmation par nécessité, écrire sur support numérique c'est programmer l'apparition de signes sur un support de lecture. En revanche, l'écriture peut procéder ou non d'une programmation par *intention*, c'est à dire exploiter ou non la logique du calcul au niveau sémiotique.

Lorsqu'elle adresse un support statique, comme le papier imprimé, l'écriture est graphique en intention et reste inscrite dans le paradigme traditionnel. Dès lors qu'elle adresse un support dynamique, tel qu'un écran d'ordinateur, l'écriture peut alors s'inscrire dans un autre paradigme, que j'appellerai computationnel, en ce sens qu'il est fondé par l'ordinateur comme support d'inscription, et qu'il relève de la raison computationnelle, c'est à dire d'une façon de penser avec le calcul que permet l'ordinateur. L'écriture computationnelle a pour intention de programmer dynamiquement l'apparition des signes par l'exécution d'algorithmes et en fonction d'actions du lecteur.

Les questions sont alors de savoir pourquoi et comment conserver une écriture traditionnelle, et pourquoi et comment mobiliser une écriture computationnelle. Questions qui se déclinent notamment au travers de l'instrumentation des logiciels d'écriture qui renvoie plutôt au paradigme du graphique (WYSYWYG) ou au paradigme du computationnel (WYSIWYM).

1 L'écriture qui veut imprimer

Le paradigme traditionnel

Une écriture numérique est traditionnelle si elle cherche d'abord à préserver les propriétés de l'écriture graphique, si elle n'est numérique que par obligation, happée malgré elle par l'universalité du numérique. Son enjeu est d'abord d'être capable de se reposer sur les compétences - d'écriture et de lecture - acquises dans le contexte du graphique. L'écriture numérique traditionnelle va nécessairement profiter de certaines fonctions spécifiques, mais de façon limitée, et essentiellement orientées vers le stockage et la circulation du contenu, plutôt que vers la production et la mise en forme des signes eux mêmes. Cette résistance aux fonctions du numérique est essentielle dans la mesure où c'est la condition pour faire l'économie de la réinterrogation des modes d'interprétation. Elle doit tellement ressembler à l'écriture graphique non numérique que les mécanismes d'interprétation semblent fonctionner sans solution de continuité. On peut citer comme exemple l'écriture de documents PDF destinés à être imprimés, ou de document Epub à destination de liseuses passives (c'est à dire non ou très faiblement interactives).

Pourquoi conserver l'écriture traditionnelle ?

Conserver la possibilité d'une écriture traditionnelle sur support numérique permet de s'inscrire dans les habitudes établies de ce que l'on a appris à l'école, de ce que l'on sait avoir du sens. L'écriture traditionnelle est *facile*, c'est ce qui nous paraît naturel, puisque c'est *déjà* acquis. L'écriture traditionnelle est *fiable*, c'est un mode de communication codifié *a priori* que nous maîtrisons, pour lequel notre potentiel de contrôle, de prédiction, est important.

L'enjeu principal est de fait de celui de l'efficacité : l'écriture traditionnelle, est une écriture qui fonctionne d'emblée.

Problématique de l'écriture traditionnelle : assumer et protéger le statique

Le risque principal de l'écriture traditionnelle n'est pas sa disparation, la forme livresque (graphique, linéaire, non interactive) est un mode de communication de l'information puissant en soi (même si cette forme passe d'un statut très largement majoritaire à un statut minoritaire). Le risque principal est plutôt de ne pas assumer ou de mal en contrôler les limites et de conduire à de mauvais hybrides, qui se revendiqueraient

encore du graphique, tout en ayant trop transigé avec les fonctions numériques. Les conditions interprétatives du graphique ne s'appliqueraient déjà plus, sans que l'on s'en rende compte, entraînant une perte de sens. Cette "sortie" involontaire du graphique peut être due à une pression mal contrôlée de la technique, il y donc une nécessité de lutter en permanence contre les tropismes du numérique. Elle peut également être due à un défaut d'analyse, de celui qui chercherait les avantages du numérique sans en accepter les transformations, pensant que l'écriture n'a pas besoin d'être repensée, que "l'ordinateur n'est qu'un outil".

2 L'écriture qui veut programmer

Le paradigme computationnel

L'écriture numérique est computationnelle lorsque elle cherche d'abord à profiter des fonctions du numérique, que ce soit pour améliorer la productivité (de l'écriture ou de la lecture) ou pour élaborer des formes d'expression différentes de celles disponibles dans le monde graphique. Deux écritures s'opposent donc, dont la frontière serait plutôt simple à tracer a priori : toute écriture qui sort du graphique, du linéaire, du non interactif, tomberait de fait dans le computationnel. Ce serait la condition pour que l'écriture traditionnelle puisse se revendiquer de l'héritage quasiment intacte de l'écriture graphique. Sous l'influence des tropismes du numérique - de ce qu'ils suscitent en terme d'interactivité, de multimédia, d'archivage, de connexion, de diffusion - on observe un développement de formes multiples d'écritures computationnelles.

Pourquoi mobiliser une écriture computationnelle ?

La première raison de s'inscrire dans l'écriture computationnelle est peut être qu'il est difficile de faire autrement, sous la pression des tropismes du numérique, qui se manifestent sous plusieurs formes.

- Une *pression technologique* fait évoluer les supports de restitution - l'exemple des dispositifs portables, *smartphones* et tablettes, est particulièrement prégnant - et ces nouveaux supports peuvent imposer de fait de nouvelles logiques de lecture. Par ailleurs les modes d'accès en réseau font également pressions sur les modes d'écriture ; l'écriture collaborative synchrone ou asynchrone étant un exemple significatif. Or pour écrire à plusieurs, en réseau, il est nécessaire de mobiliser la dimension computationnelle de l'écriture, pour échanger des commentaires, comparer des versions...

  Exemples : Glose[p. 39*] – Instantanéité[p. 37*] – Polymorphisme[p. 42*] – Publication[p. 38*]

- Une *pression organisationnelle* liée à la numérisation des métiers conduit également petit à petit à s'inscrire dans une écriture computationnelle. On pourra citer comme exemple la production des métadonnées et la mise en place de *workflows* d'écriture, la nécessité de respecter des règles opérationnelles d'interopérabilité ou légales d'accessibilité.

  Exemples : Accessibilité[p. 39*] – Génération[p. 35*] – Interopérabilité[p. 43*] – Métadonnées[p. 37*]

- Une *pression personnelle* se manifeste également au niveau individuel : l'envie de bien faire ou de faire comme les autres et le sentiment de pouvoir faire "mieux" avec le numérique.

  Exemples : Hypertextualisation[p. 40*] – Interactivité[p. 41*] – Multimédia[p. 42*] – Scénarisation[p. 43*]

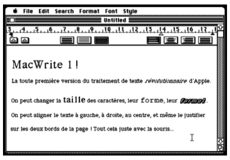
Problématique de l'écriture computationnelle : le nouveau et le multiple du dynamique

Le premier problème posé par l'écriture computationnelle est qu'il remet en cause les processus interprétatifs établis. Ces nouveaux processus n'ont pas été, pour l'essentiel, appris à l'école (légitimement restée ancrée dans la sécurité de la tradition graphique), ils doivent donc être inventés par et pour ceux qui écrivent, et par et pour ceux qui lisent. Les canons de l'écriture graphique s'ancrent dans une tradition qui se compte en siècles (encyclopédies, essais, romans, journaux, ouvrages techniques, documentations, articles scientifiques...), tandis que ceux de l'écriture computationnelle sont en invention, et se comptent encore en années, dans les meilleurs de cas.

Par ailleurs, cette écriture est multiple et complexe et chaque composante (interaction, multimédia, glose...) fait appel à des dimensions interprétatives différentes. Il y a deux axes qui résonnent entre eux et se combinent pour former une matrice des possibles : un axe des fonctions d'écriture qui portent le champ du possible et un axe des types documentaires - des genres - qui définissent des structures générales d'écriture qui s'établissent par rapport à la poursuite d'une finalité (Crozat et al. 2012).

3 Instrumentation des écritures : du WYSIWYG et du WYSYWYM

La plupart des systèmes d'écriture numérique modernes se fondent sur le principe du WYSIWYG, « What you see is what you get », ou littéralement en français : « Ce que vous voyez est ce que vous obtenez ». Cette approche vise, en résumé, à permettre à l'utilisateur de créer un document tout en en composant le *rendu final*, comme dans sur un support traditionnel non numérique. Les premiers logiciels WYSIWYG datent des années 1970, mais leur démocratisation est associée aux premiers Macintosh en 1984 et aux traitements de texte MacWrite (1984) puis Word (1989).



p. 43*

MacWrite, le premier traitement de texte WYSIWYG (Source :)

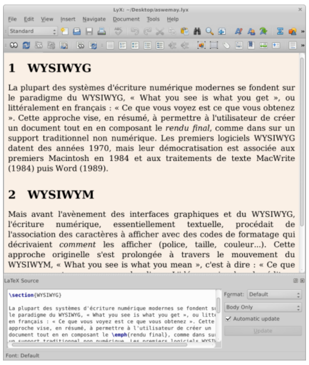
Mais avant l'avènement des interfaces graphiques et du WYSIWYG, l'écriture numérique, essentiellement textuelle, procède de l'association des caractères à afficher avec des codes de formatage qui décrivaient *comment* les afficher (police, taille, couleur...). Par exemple sous WordStar les mots en gras doivent être placés entre deux balises *^B*.

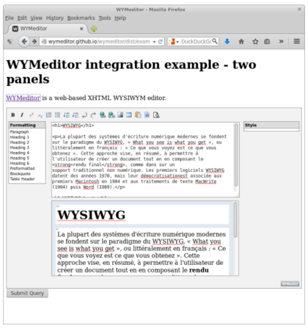
p. 44*

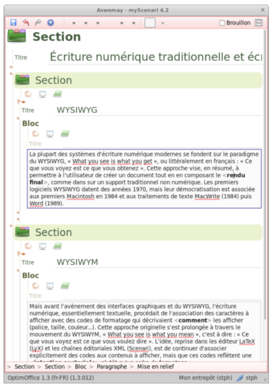
WordStar, un traitement de texte avant le WYSIWYG (Source :)

Cette approche originelle s'est prolongée à travers le principe WYSIWYM, « What you see is what you mean », c'est à dire : « Ce que vous voyez est ce que vous voulez dire », mobilisée dans les éditeurs LaTeX (LyX), HTML (WYMeditor) et les chaînes éditoriales XML (Scenari). L'idée est de continuer d'associer explicitement des codes aux contenus à afficher, mais d'une part que ces codes reflètent une *intention auctoriale* plutôt qu'un ordre de formatage, et d'autre part de mobiliser une mise en forme graphique adaptée au processus d'écriture (qui n'est pas le rendu final).

Éditeurs WYSIWYM : Lyx, WYMeditor, Scenari







Le WYSYWYG

L'instrumentation d'une écriture traditionnelle doit avoir pour objectif premier de masquer la dimension computationnelle de l'écriture. L'auteur qui écrit n'a pas besoin de savoir qu'il programme, puisqu'il ne *veut pas* programmer ; il est donc plus efficace de le maintenir dans un environnement qui simule l'ordre graphique qu'il maîtrise.

C'est sur cette hypothèse que reposent les fondamentaux du WYSIWYG : l'ordinateur simule une page blanche sur l'écran qui permet de se projeter directement dans l'imprimé qui résultera de l'acte d'écriture (comme le propose la traduction du terme WYSIWYG en français par « tel écran, tel écrit »). Les actes de programmation sont réalisés via des manipulations sémiotiques (cliquer sur une icône avec la souris) et provoquent immédiatement une manifestation sémiotique à l'écran qui préfigure un résultat équivalent sur l'imprimé (mettre en mot en italique).

Le WYSIWYM

L'approche WYSIWYM vise à proposer à l'auteur une interface de création de contenus qui se détache de la mise en forme finale, contrairement au WYSIWYG, et intègre explicitement des actes de programmation. L'écriture WYSIWYM consiste d'une part à produire *directement* une partie des signes, comme dans une approche WYSIWYG ; et d'autre part à *programmer* d'autres signes qui seront générés algorithmiquement par la machine, ainsi que les rendus et les comportements dynamique de l'interface de lecture. L'auteur déclare à l'aide de l'environnement d'écriture ce qu'il souhaite obtenir afin que la machine se charge de calculer une forme qui corresponde à ses intentions.

« Ainsi l'utilisateur ne met plus de mots en gras, mais spécifie qu'ils sont importants. Il ne crée plus d'animation pour marquer l'arrivée d'un bloc dans une diapositive, mais il précise que ce bloc en est la conclusion. Il ne crée plus le menu d'un site web, mais déclare sa structure. Et c'est le programme informatique qui se chargera, lors d'une phase que l'on appellera publication, de mettre en gras les mots importants, d'appliquer des animations aux blocs de conclusion ou de générer des menus interactifs. (Crozat, 2007) »

Le WYSIWYM permet d'instrumenter l'écriture computationnelle en préservant un compromis entre ouverture des possibilités calculatoires via une programmation déclarative de haut niveau et maintien d'une inscription graphique propre à l'écriture.

	Intention de coder des contenus statiques	Intention de coder des comportements dynamiques
Codage par programmation implicite	Sémiotique Sémiotique WYSIWYG (traditionnel)	Sémiotique Logique WYSIWYM (computationnel)
Codage par programmation explicite	Logique Sémiotique WYSIWYM (computationnel)	Logique Logique WYSIWYM (computationnel)

IV Étude de cas d'écriture computationnelle

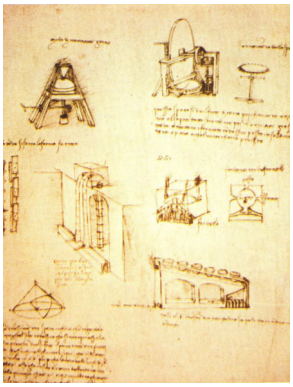
Dans cette partie nous présentons quelques cas d'écriture computationnelle typique. Ces cas sont illustrés au travers d'exemples issus de la chaîne éditoriale Scenari.

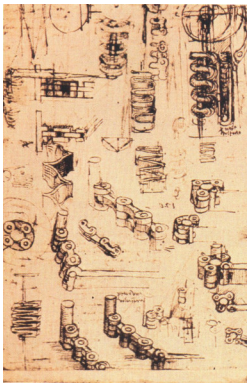
1 Insérer une image dans un texte (multimédia)

Un cas apparemment trivial

La plupart des écritures numériques mêlent le texte alphabétique et les images, l'universalité du support y conduisant rapidement. Les traitements de texte permettent ainsi simplement d'ajouter une image. Pour cet exemple nous prendrons un texte sans importance au sein duquel nous insérerons des images issues de plans de Léonard de Vinci (Cianchi, 1984).

Trois dessins de Léonard de Vinci





Traitement de texte WYSIWYG (écriture traditionnelle)

Dans un traitement texte traditionnel, l'insertion de l'image se fera graphiquement de façon à obtenir le rendu désiré. On jouera sur les paramètres graphiques tels que :

- la taille de l'image (redimensionnement) ;
- le positionnement de l'image relativement au flux de caractères : avant, après, à droite, à gauche, entourée par... ;
- la rotation de l'image.

L'on voit sur les trois exemples qui suivent que l'approche WYSIWYG fonctionne parfaitement bien, tant que c'est une impression sur papier qui est visée.

 (cf. 1.pdf)

 (cf. 2.pdf)

 (cf. 3.pdf)

Lecture sur écran

En revanche, si l'on projette ces trois exemples sur des supports dynamiques de lecture, de nombreuses critiques peuvent être formulées, par exemple :

- Pour le dessin *Polyèdre* :

  Sur un petit écran (*smartphone*), le choix d'un vis à vis entre le texte et l'image n'est pas opportun, on préférera un bout à bout classique.

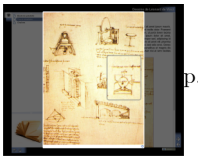
- Pour le dessin *Four* :

  Sur un écran d'ordinateur, on pourra fournir une image très haute résolution permettant d'explorer les plus petits détails de cette image. Mais pour éviter un temps de chargement long ou un encombrement spatial de la page, on préférera sûrement afficher d'abord une réduction de l'image qui permettra d'activer l'image haute résolution sur une action de l'utilisateur. On pourra aussi dans ce cas ajouter une loupe permettant d'accéder à la lecture des notes manuscrites sur le schéma.

- Pour le dessin *Chaîne* :

  Sur un écran d'ordinateur le choix d'avoir pivoté l'image en mode paysage (qui a permis d'optimiser la mise en page sur papier) n'est plus adapté, l'écran ne pouvant se tourner aussi facilement qu'une feuille imprimée d'une part, et la possibilité de défilement dynamique annulant la limite verticale de la page d'autre part.

 p. 44*

 p. 45*

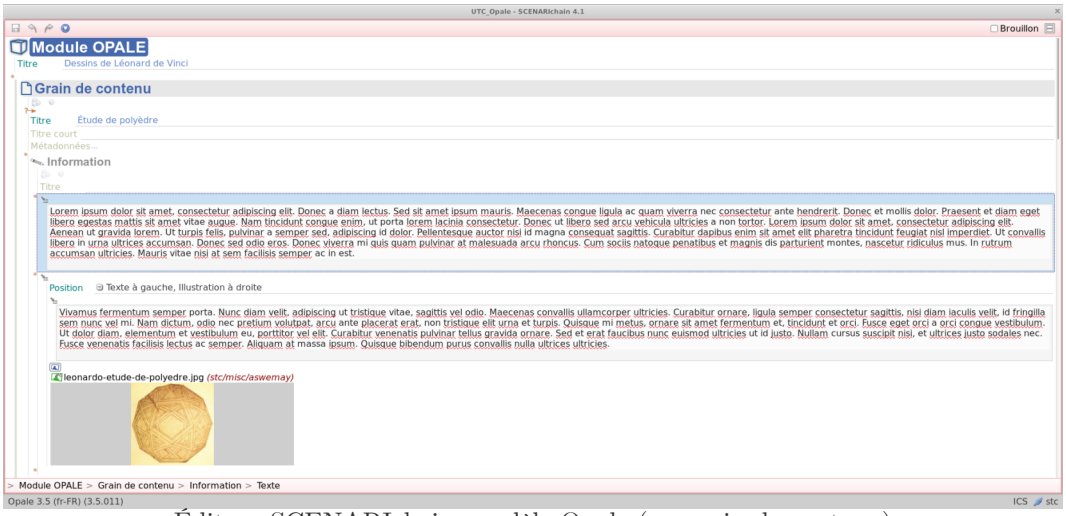
 p. 46*

Chaîne éditoriale WYSIWYM (écriture computationnelle)

Ce que ces exemples nous montrent c'est d'une part que le polymorphisme et le WYSIWYG sont en contradiction intrinsèque, puisque, par définition, si ce que l'on voit est ce qu'on obtient, on ne peut voir, donc obtenir, qu'une seule configuration graphique, alors que chaque support de restitution requiert des configurations graphiques différentes (ici, la taille des images, leur alignement, leur orientation).

On voit également que l'écriture traditionnelle ne peut pas prendre en charge l'interaction du support de restitution. Il n'est en effet pas possible de systématiser la gestion interactive des images : disposer d'une miniature et d'une version haute résolution ou d'une loupe dépend du contenu de l'image (présence de détails, de caractères textuels) et de l'intention de l'auteur, et pas seulement de ses propriétés graphiques. Un auteur pourra décider que le dessin *Polyèdre* est destiné à une vue d'ensemble et donc qu'une fonction de zoom sera contre-productive, alors que rien ne permet de le détecter automatiquement. Une action explicite de paramétrage par l'auteur est donc nécessaire pour activer les bonnes fonctions de rendu et d'interaction.

Il est donc nécessaire de passer à une écriture computationnelle, qui intègre une part de programmation explicite dès lors qu'un écran est visé, et a fortiori plusieurs types d'écrans.



Éditeur SCENARIchain, modèle Opale (un grain de contenu)

Sur cet écran on visualise une saisie WYSIWYM du dessin *Polyèdre*. La déclaration de la position de l'image par rapport au texte n'est plus graphique, elle est programmée par l'auteur. L'auteur mobilise une fonction qui prend en entrée un texte, une image et un paramètre de positionnement ; cette fonction engendrera un comportement à la publication, comme par exemple : mettre en vis à vis sur un écran large, et mettre à la suite sinon. L'explicitation de la programmation ouvre un champ d'expression beaucoup plus large (d'autres fonctions, d'autres rendus) que ce que permettait le positionnement graphique WYSIWYG.

Éditeur SCENARIchain, modèle Opale (une image)

Sur le second écran on visualise le paramétrage de l'affichage de l'image correspondant au dessin *Four*. L'auteur en fixe une taille relativement à une taille de police standard d'un texte de référence. Ainsi la taille de l'image n'est plus strictement graphique, exprimée en pixels ou en centimètres, elle devient un paramètre abstrait contrôlé par l'auteur, qui ainsi opère sur le calcul qui sera exécuté sur l'image à la publication. D'autres paramètres peuvent être mis à sa disposition, d'ordre graphique (la possibilité d'une rotation de l'image en cas de besoin d'optimisation d'espace de page), ou d'ordre interactif (l'activation d'une fonction de zoom ou de loupe).

Cette couche de programmation explicite apportée par le WYSIWYM permet à l'auteur d'élargir l'expression de son intentionnalité au delà des seules caractéristiques graphiques de son écriture devenue computationnelle, pour étendre et exploiter l'algorithme qui contrôle le rendu dynamique du texte sur un écran d'ordinateur.

V Écriture WYSIWYM : "What you see is what you mean"

Dès lors que son écriture procède du computationnelle, l'auteur doit avoir conscience qu'il programme afin d'en garder le contrôle. En cela le principe du WYSIWYG ne peut pas s'appliquer à l'écriture computationnelle, puisque ce dernier vise précisément à masquer la dimension calculatoire du numérique pour simuler un ordre graphique. D'un autre côté le recours à un langage de programmation de bas niveau n'est pas non plus souhaitable, car il relève d'une pratique d'informaticien, qui requiert des compétences spécifiques (concepts, méthodes, langages...) et qui interfère avec la pratique d'auteur.

Le WYSIWYM a été pensé pour permettre un compromis entre la posture essentiellement sémiotique du WYSIWYG et celle essentiellement logique de la programmation informatique.

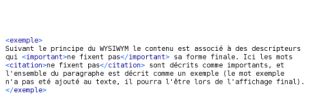
1 Principe du WYSIWYM

Le WYSIWYM consiste à associer aux contenus que l'on écrit des *descripteurs* qui les caractérisent pour paramétrer ce qu'un ensemble de programmes doit faire de ces contenus lors de leur restitution pour la lecture.

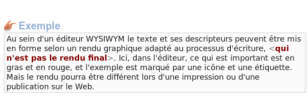
Toute écriture numérique procède d'un codage (inscription de l'information à transmettre sous la forme de chiffres binaires) et toute lecture d'un décodage (transformation des chiffres binaires enregistrés en signes interprétables). Dans le cadre du WYSIWYM l'idée est que ce codage ne se fait pas sur une dimension exclusivement sémiotique (comme pour le WYSIWYG), mais également sur une dimension calculatoire. Ce qui est codé, c'est à la fois du signe et du calcul. L'enjeu est de permettre au programme de lecture une plus grande variabilité dans la restitution du contenu pour mieux remplir des fonctions computationnelles comme le polymorphisme, la paramétrisation ou l'interactivité.

Exemple WYSIWYG et WYSIWYM

Exemple : Suivant le principe du WYSIWYG le contenu et sa mise en forme sont **confondus**. Dans cet exemple, le mot « Exemple » est souligné, le mot « confondus » est en gras, le paragraphe est justifié et ajouté sur trois lignes pour une page de format A4.



```
<exemple>
Suivant le principe du WYSIWYM le contenu est associé à des descripteurs
qui <important>ne fixent pas</important> sa forme finale. Ici les mots
<citation>ne fixent pas</citation> sont décrits comme importants, et
l'ensemble du paragraphe est décrit comme un exemple (le mot exemple
n'a pas été ajouté au texte, il pourra l'être lors de l'affichage final).
</exemple>
```



Exemple

Au sein d'un éditeur WYSIWYM le texte et ses descripteurs peuvent être mis en forme selon un rendu graphique adapté au processus d'écriture, <**qui n'est pas le rendu final**>. Ici, dans l'éditeur, ce qui est important est en gras et en rouge, et l'exemple est marqué par une icône et une étiquette. Mais le rendu pourra être différent lors d'une impression ou d'une publication sur le Web.

Des symboles pour les hommes et pour les machines

Les descripteurs associés au contenu sont donc des symboles qui sont interprétés par l'auteur qui écrit pour exprimer une information relative au contenu qu'il écrit, pour préciser *ce qu'il veut dire*. Ce sont également des symboles reconnus par les logiciels de lecture ou de publication. Ces logiciels vont donc adapter la présentation du contenu sur le support de lecture afin de restituer un environnement sémiotique fonction du contenu et de sa description.

Les descripteurs concernent toutes les formes sémiotiques (textes, images...) et peuvent revêtir également plusieurs formes : textuelle, graphique, iconique... Un éditeur WYSIWYM est donc aussi un éditeur graphique, mais la forme qu'il adopte n'est pas contrainte par le respect de la forme finale qui sera publiée, et elle cherchera plutôt à optimiser l'acte d'écriture.

« Ainsi, avoir un éditeur qui met en forme en gras un texte qui a été déclaré comme "important" est simplement destiné à exploiter nos habitudes de présentation pour "faire sens" plus vite dans notre esprit : lire une balise <important> encadrant le texte est naturellement plus laborieux et surtout ne fait pas appel à nos "réflexes". De même pour la notion de paragraphe, de listes à puces, etc. Pourquoi ne pas exploiter la force de ces réflexes acquis ? Après, dans une publication donnée, que la puce de la liste soit un rond noir ou

un carré rouge pour mieux se fondre dans une charte graphique, quelle importance pour l'auteur ? Le sens est là ! (Spinelli, 2006) »

Exemple WYSIWYM versus programmation

DONNER UN EXEMPLE avec une interaction : WYSIWYG, WYSIWYM, codé (XSLT ou Js par exemple)

Cf exemple Exercice simplifié

Publication, polymorphisme, séparation fond/forme

indépendance vis à vis de la forme

logique de publication par l'éditeur dans plusieurs formats standard de lecture

ce sont plus souvent les éditeurs qui adaptent le contenu (principe de publication) que les logiciels de lecture qui s'adaptent

2 Une petite histoire du WYSIWYM

En 1999, alors que nous commençons à travailler sur ce qui deviendra Scenari, écrire sur un ordinateur, c'est écrire avec Word, *Notre Word*, comme d'autres auraient dit "Notre Ford". Tout le monde a appris à taper sur un clavier avec Word. Word est sur toutes les machines. Le Wordisme est une religion, vouloir faire autre chose est une hérésie. Nous sommes alors dans le meilleur des mondes, pourquoi en changer ?

« Our Ford himself did a great deal to shift the emphasis from truth and beauty to comfort and happiness [fr] (Huxley, 1932) »

Pour sacrifier un peu au confort donc, peut-être, aux habitudes certainement, pour rechercher de la « vérité » du numérique, de son potentiel computationnel, et même, finalement, retrouver une « beauté » perdue dans ces traitements de texte et éditeurs WYSIWYG qui modèlent encore tant de nos textes.

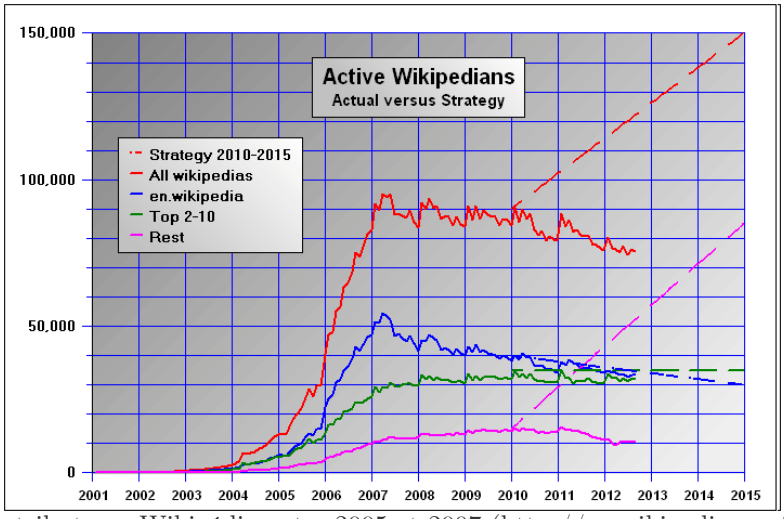
Le Wordisme et Scenari

Le terme WYSIWYM a été retrouvé par Sylvain Spinelli en 2006. Nous cherchions - quotidiennement à l'époque, un peu moins souvent maintenant - à expliquer les choix ergonomiques faits pour l'interface et la logique d'édition dans Scenari. Celle-ci se démarquait fortement de la logique dominante, pour ne pas dire exclusive, du WYSIWYG, pour ne pas dire de Word.

Une préoccupation récurrente de nombreux nouveaux et futurs utilisateurs de chaînes éditoriales Scenari était de chercher comment concilier les avantages de celui-ci - polymorphisme, réutilisation, contrôle de la rédaction et de la publication -avec *leur* Word - c'est à dire une visualisation ancrée dans l'impression, un fichier monolithique, et une liberté totale d'écriture et de mise en forme. L'enjeu était de mettre en évidence la contradiction de ce discours. Le WYSIWYM ne peut se fonder que sur une déconstruction du WYSIWYG, c'est à dire une démonstration de l'opposition entre son principe technique et ergonomique d'ordre graphique et la recherche de certaines fonctions d'ordre computationnel. Pour exister, le WYSIWYM doit d'abord tuer Word.

Le traitement de texte et l'écriture sur le Web au milieu des années 2000

OpenOffice commençait à peine à se faire une place (marqué en France par une adoption à l'assemblée nationale, l'armée, la gendarmerie en 2006 et 2007), Google Docs naissait tout juste (2006). L'édition Web était arrivée au début des années 2000 via les CMS (Drupal et SPIP en 2001, Wordpress en 2003, Joomla! en 2005) et puis les Wikis. Mais les pratiques étaient loin d'être ce qu'elles sont aujourd'hui. Les contributions Wikipédia par exemple se développent réellement entre 2005 et 2007. C'est aussi à cette période qu'émerge l'expression Web 2.0 et que commencent à se développer les pratiques de commentaires en ligne.



Explosion des contributeurs Wikipédia entre 2005 et 2007 (http://en.wikipedia.org/wiki/Wikipedia)

Naissance du terme WYSIWYM : de l'ingénierie des connaissances à l'ingénierie documentaire

La première formulation du terme WYSIWYM remonte à un article de Power, Scott et Evans (1998). Le terme est utilisé alors pour *What You See Is What You Meant*, dans un domaine assez différent. Les auteurs cherchent en effet une solution pour éditer des bases de connaissance exprimées sous la forme de réseaux conceptuels, et ils explorent pour cela des représentations sous des formes documentaires interactives manipulables par les utilisateurs de type expert métier, sans compétence en programmation ou en représentation des connaissances.

Leur objectif est donc de faire réaliser des programmes (des bases de connaissances) par des non informaticiens (des spécialistes métiers). Pour cela ils proposent à ces utilisateurs un texte interactif (*feedback text*) dont la manipulation (ajout/suppression de phrases) modifie le programme (ajout/suppression de connaissances) qui modifie à son tour le texte. Le WYSIWYM naît donc à l'origine d'une mobilisation de l'écriture pour la programmation, alors que le WYSIWYM dont nous parlons consiste au contraire à réintroduire de la programmation pour de l'écriture. Mais dans les deux cas il s'agit bien de mobiliser la dimension computationnelle en articulation avec la dimension graphique. Et de fait les auteurs sont amenés à explorer les mêmes questions que nous.

Ils proposent notamment une caractérisation des traitements de texte WYSIWYG. Ces éditeurs permettent d'intervenir directement au niveau des caractères (graphémique), indirectement au niveau graphique (on peut changer la présentation des caractères, mais pas dessiner librement sur la page) et pas du tout au niveau sémantique (qui n'est qu'interprétable par l'homme à partir des deux niveaux précédents). *Le WYSIWYM est alors défini comme le moyen pour les utilisateurs d'intervenir directement sur la couche sémantique*, par la formalisation de connaissances.

Outre cette hypothèse fondatrice, on retrouve des principes structurants comme :

- la génération de textes et de mises en formes à partir de formalisations informatiques et la production de formalisations à partir de manipulations graphiques ;
- le contrôle via des ancres d'interaction (l'utilisateur n'intervient plus au niveau du caractère comme dans un traitement de texte WYSIWYG, mais à certains points définis au sein de la structure) ;
- la modélisation (les outils sont d'emblée spécialisés pour un contexte métier, des procédures multilingues dans le cas présenté par les auteurs).

Ces travaux déboucheront sur l'application du WYSIWYM pour l'édition de document multimédia (Van Deemter and Power, 2000). L'idée est d'associer à la base représentant le contenu, une seconde base définissant sa forme et sa présentation. Le dernier pas est ainsi franchi pour passer de l'ingénierie des connaissances à l'ingénierie documentaire. Le premier usage du terme en ingénierie documentaire, accompagné de la suppression du *t* de *Meant*, est associé à l'éditeur LyX (Morère, 2001). LyX est un éditeur LaTeX qui s'inscrit dans le courant de l'écriture structurée (André et al., 1989). Il veut se démarquer de « la tradition périmée héritée de la machine à écrire (Morère, 2001) ». La rupture principale apportée est la (re)séparation des tâches d'écriture et d'édition, qu'avaient fusionnées les traitements de texte WYSIWYG. Dès lors « le travail de typographie sera pris en charge en majorité par l'ordinateur, non par l'auteur (Ibid.) ».

Développement des éditeurs WYSIWYM : l'écriture structurée

Le principe du WYSIWYM - associer explicitement des marqueurs aux contenus dans l'intention de programmer leur publication - remonte en fait aux origines des traitements de texte. En effet, avant le développement des interfaces graphiques dans les années 1980, les terminaux textuels ne permettaient par de mettre en forme dynamiquement le texte à l'écran ; la solution consistait alors à associer des caractères particuliers au texte pour programmer un rendu particulier lors de l'impression, par exemple, mettre une chaîne en gras ou en italique.

^YCeci est un texte en italique^Y

Fonction italique sous WordStar

L'écriture structurée - ou édition structurée - est une approche à la fois technique et méthodologique de l'écriture numérique dont on peut faire remonter l'origine aux technologies LaTeX (créé en 1982 sur la base du langage d'édition TeX, lui-même créé en 1978) et SGML (norme ISO depuis 1986, issue des travaux d'IBM initiés en 1979 avec GML). Tandis que le mouvement WYSIWYG cherche à dissoudre la structure du texte dans sa représentation graphique, l'écriture structurée cherche au contraire à conserver le principe du marquage explicite du contenu, grâce à un système de balisage qui serait à la fois intelligible par l'homme et par la machine.

\begin{document}	<document>
\section{L'édition structurée}	<section>L'édition structurée
Le principe du WYSIWYM...	<paragraph>Le principe du WYSIWYM...
\end{document}	</document>

Exemple de structure LaTeX (à gauche) et SGML (à droite)

La nouveauté introduite par l'écriture structurée est que les balises ne cherchent plus à programmer directement une mise en forme (gras, italique), mais à expliciter la structure du texte (titre, résumé, paragraphe, mots importants...) : ainsi l'auteur mobilise les balises pour formaliser l'organisation de son texte, et cette formalisation est ensuite exploitée par un algorithme pour réaliser des mises en forme. Cette forme d'écriture écarte donc le graphique pour s'inscrire plutôt dans une forme de programmation ; et elle est bien reçue comme telle par les auteurs, qui, dans l'ensemble, la rejette également. L'écriture structurée reste alors une pratique circonscrite à des niches professionnelles au sein desquelles les avantages de l'écriture structurée - le contrôle typiquement - est déterminant : LaTeX pour l'édition scientifique, SGML pour la documentation technique. Dans tous les cas l'écriture structurée reste à ce stade une pratique d'experts formés spécifiquement pour (rédacteurs techniques), ou de "logiciens" dont les modes de raisonnement et d'écriture sont compatibles *a priori* (informaticiens, mathématiciens...).

Nous pouvons alors redéfinir le WYSIWYM comme le mouvement qui a conduit à *réconcilier les pratiques d'écriture structurée avec les pratiques d'écriture ordinaire*. Côté SGML, une des premières initiatives est l'éditeur Grif (Quint, 1987), côté LaTeX, LyX (Morère, 2001). Dans tous les cas, l'approche consiste à réintroduire du graphique au dessus du programmatique lorsque c'était possible. Et par ailleurs, les notions de *modélisation* et de *spécialisation* s'affirment lorsque les systèmes se frottent à la complexité consistant à rendre le plus accessible possible la couche de formalisation, pour des auteurs et non des programmeurs.

« Tout cela rend improbable la définition d'un modèle unique pouvant décrire n'importe quel document à un niveau d'abstraction relativement élevé. On pourrait évidemment tenter de dresser une liste d'entités d'usage général, comme les chapitres, sections paragraphes, titres, etc... et ramener les entités exclues de cette liste à celles qui y figurent. Ainsi une introduction pourrait être assimilée à un chapitre, une clause de contrat à un paragraphe ou une section. Mais en voulant élargir le domaine d'utilisation de certaines entités privilégiées, on risque de les vider de leur sens et donc de réduire la puissance du modèle. (Quint, 1987, p95) »

« Il paraît impossible de dresser un inventaire exhaustif de toutes les entités nécessaires et suffisantes pour décrire n'importe quel document. Il semble tout aussi impossible de décrire une fois pour toutes l'organisation de ces entités dans un document. C'est pourquoi, l'approche que nous avons prise s'écarte de celle utilisée dans certains systèmes de traitement de documents structurés [...] où les entités et leur organisation sont spécifiées de façon définitive. Nous proposons plutôt un méta-modèle qui permet de décrire de nombreux modèles, chaque modèle ayant une utilisation limitée à une classe de document. (Ibid.) »

C'est dans cette filiation que s'inscrit Scenari, en profitant des avancées apportées par les technologies XML à partir de 1998.

Le WYSIWYM : un compromis entre graphique et logique, en contexte

Le WYSYWIM est historiquement la recherche d'un *compromis* entre manipulation graphique et programmation, *en fonction du contexte d'usage visé*. Selon les fonctions computationnelles visées et la

typologie des acteurs impliqués dans le projet d'écriture, le curseur entre logique et graphique sera renégocié. Mais l'hypothèse du WYSIWYG doit être abjurée : tout n'est pas soluble dans le graphique, il faudra programmer, un peu.

Dualité sémiotique-logique du WYSYWIM

Un éditeur WYSWYM peut être vu comme *un environnement de programmation de très haut niveau* (au sens informatique), c'est à dire où l'on manipule des structures logiques proches du monde sensible, ici proche des manifestations sémiotiques recherchées, où l'on favorise l'interprétation par l'humain sur le calcul par la machine. Et un éditeur WYSIWYM peut également être vu comme *un outil d'écriture traditionnelle augmenté de fonctions de programmation explicite.* Il reste essentiellement sémiotique tant cela est possible, il repose sur une logique de programmation déclarative de haut niveau de préférence à une logique de programmation algorithmique.

Ainsi les balises qui supportent la structuration du texte dans un éditeur WYSIWYM ont une valeur sémiotique, elles sont des méta-données (des informations sur le texte lui même), et elles sont d'ailleurs généralement restituées graphiquement à la publication (les mots importants seront mis en forme en italique ou en gras, les définitions seront préfixées d'une étiquette, d'une couleur ou d'une icône). Mais les balises ont également une valeur logique, elles sont des paramètres de programmation, des instructions.

Le WYSIWYG est (presque) mort, vive le WYSIWYM

Le WYSIWYG a constitué une étape fondamentale pour l'adoption de l'ordinateur comme outil d'écriture, en faisant la jonction entre le monde du graphique et le monde du computationnel. Mais il a aussi contribué à paupériser les pratiques éditoriales : d'une part en transférant les métiers de l'édition sur les auteurs, appauvrissant la qualité graphique des textes ; et d'autre part en prétendant adresser de façon uniforme la totalité des pratiques d'écriture, de la lettre personnelle mono-page aux documents professionnels de plusieurs milliers de pages. Il butte aujourd'hui sur de nombreux obstacles qui relèvent du computationnel : pluralité des supports, interactivité, accessibilité...

L'enjeu est aujourd'hui de dépasser cette étape, et le WYSIWYM constitue un candidat sérieux à la passation de pouvoir.

3 WYSIWYM et fonctions du numérique

Cette partie s'appuie essentiellement sur un article de Spinelli (2006) publié sur le wiki de Scenari afin d'expliquer les enjeux du WYSIWYM aux utilisateurs des chaînes éditoriales XML. J'ai croisé ses considérations avec les fonctions auxquelles elles renvoient.

Polymorphisme

Un premier apport du WYSIWYM est le polymorphisme. En ayant dégagé l'écriture du contenu de sa forme de restitution finale, il devient possible d'écrire pour plusieurs supports de publication *à la fois.*

« En WYSIWYG, on écrit pour un et un seul support majeur (une et une seule mise en forme papier ou un et un seul site web), les autres sont considérés comme secondaires et dégradés. Par exemple, lorsqu'un support papier et un support Web trouvent tous deux un usage "important", le WYSIWYG est logiquement impossible. (*Ibid.*) »

C'est parce que toute écriture s'ancre dans une forme que le WYSIWYG et le polymorphisme sont antinomiques. Si l'auteur a la promesse d'une inscription graphique fidèlement restituée, alors la promesse doit être tenue, la forme ne peut pas varier. En revanche, si les formes de lecture sont explicitement abstraites dans une forme d'écriture qui les représente, en quelque sorte, il devient tenable d'écrire dans une forme (auctoriale) pour d'autres formes (lectoriales).

Interactivité

L'écriture de contenus interactifs, implique une programmation de cette interactivité qui, fondamentalement, ne relève plus du graphique. Si la représentation graphique de programmes est dans une certaine mesure possible, elle ne relève en tous cas plus du WYSIWYG, puisque l'interaction ne peut être seulement vue, elle doit être exécutée.

« Le WYSIWYG ne permet pas la production de contenu dynamique et interactif. Comment éditer un contenu riche qui apparaît dans un "tooltip" ? Comment écrire un contenu de type "web-radio" c'est à dire dont l'apparition/disparition est pilotée par une ligne de temps ? Comment écrire un exercice interactif en précisant la ou les bonnes solutions, le mode du calcul de scoring, les différents feed-backs possibles en

fonction des réponses de l'apprenant ? (*Ibid.*) »

Le WYSIWYM permet donc de sortir du graphique pour intégrer la dimension computationnelle de l'interaction. Le WYSIWYG ne peut que proposer des formes simplistes d'interaction (si simple qu'elles se fondent dans leur forme graphique), et souvent nécessite une technicité importante pour forcer un logiciel qui n'est pas conçu a priori pour intégrer de l'interaction. On pourra citer par exemple les exercices interactifs de primaire "Les exOOOs d'Aleccor" (Cloarec, 2015), réalisés avec le traitement de texte OpenOffice Writer (exercices par ailleurs excellents en tant que contenu).

Paramétrisation

La paramétrisation implique par définition que le contenu affiché dans le logiciel n'est pas ce qui sera vu par le lecteur, puisque cette visualisation dépend justement de paramètres.

« En WYSIWYG, on écrit un seul contenu monolithique où tout est "vu" : cela ne permet pas d'écrire un contenu plus riche qui est ensuite filtré en fonction du contexte d'usage et du support désiré (exemple : contenu en slideshow présentiel, support apprenant, consignes tuteurs). Impossible donc de faire un contenu à profondeur variable (*Ibid.*) »

Les descripteurs du WYSIWYM sont par construction des paramètres. S'ils ont souvent une fonction graphique, ils peuvent également être interprétés par le programme de lecture pour gérer par exemple un affichage sélectif en fonction de l'usage visé. Le contenu ci-après peut ainsi être interprété comme une instruction de mise en forme (mettre en valeur la question et la solution) ou comme une instruction de paramétrage (afficher les questions, mais pas les solutions).

<exercice>

<question>Décrivez en français ce que représente ce diagramme.</question>

<solution>Pour chaque moteur on relève tous les défauts observés à des dates différentes. Pour chacun de ces défauts on note le kilométrage et on diagnostique la gravité.</solution>

</exercice>

Métadonnées

L'universalité du numérique permet la fusion du contenu avec ses métadonnées, que ce soit pour enrichir l'édition (identification, datation, géolocalisation...) ou faciliter la gestion (licences, mots-clés...).

« Sans aller jusqu'à du contenu à profondeur variable, il est très souvent nécessaire d'ajouter des informations non publiées : quelques metadonnées, des commentaires pour l'orateur/tuteur, etc. Sur le plan ergonomique, il est très difficile de faire un éditeur WYSIWYG qui permette ce genre de choses. Des solutions ont néanmoins été trouvées par les éditeurs, au coup par coup : fenêtres de propriétés, bulles de commentaires superposées, etc. Ces solutions sont très limitées et surtout dédiées à des usages prédéfinis car très difficile à rendre ergonomiquement acceptable (*Ibid.*). »

Un exemple caractéristique est celui de la gestion des légendes des tableaux ou images dans les traitements de texte. En général la légende peut être saisie dans une sur-fenêtre dédiée. Mais parce qu'elle *doit* être affichée en WYSIWYG, elle est ensuite matérialisée par un flux de caractères sous la ressource considérée, associée avec un libellé et une numérotation. Et ce flux de texte *doit* alors être modifiable graphiquement, directement, sans passer à nouveau par la fenêtre. Cela engendre rapidement des erreurs ou une complexité importante, alors que la fonction à rendre est triviale dès lors que l'on accepte que cette légende n'est pas dans le flux de texte principal (visible et modifiable) lors de l'écriture, ce qui ne l'empêchera pas de l'être à la publication. L'acceptation de cette désynchronisation est à la base du WYSIWYM.

Contrôle

Les logiciels WYSIWYG ont cherché à reproduire la liberté dont dispose un auteur avec une page blanche et un stylo ou une machine à écrire tout en la combinant avec la puissance dont dispose un éditeur avec du matériel d'imprimerie. Cette hypothèse fondatrice pose deux problèmes. Elle suppose que l'auteur a les compétences de l'éditeur dans la manipulation des outils de composition graphique, ce qui n'est pas le cas en général. Ensuite, elle suppose que l'auteur est libre dans le format de son écriture. Or, dans de nombreux écrits, comme les écrits professionnels, que l'on pourra appeler utilitaires (par opposition aux documents personnels ou artistiques), ce n'est pas la liberté ou la compétence graphique de l'auteur qui est recherchée, mais plutôt une standardisation et une optimisation des productions permettant d'en contrôler l'écriture, la lecture, la gestion. Or le numérique permet d'inscrire et d'automatiser cette logique de contrôle au sein même du processus d'écriture.

« L'approche WYSIWYG [...] met donc en avant la forme et pas la structuration du contenu. De ce fait, il est encore moins acceptable pour l'auteur que l'outil le contraigne ou le guide dans une démarche d'écriture[...]. En d'autres termes, l'édition WYSIWYG est étrangère aux concepts de structuration documentaire, modélisation, ligne éditoriale. *(Ibid.)* »

Si de nombreux logiciels de gestion électronique de documents implémentent une surcouche de gestion au dessus de logiciels WYSIWYG classiques, cela conduit à des difficultés en terme technique (contraindre a posteriori ce qui est libre a priori n'est pas simple) et en terme d'acceptation (comment accepter le contrôle alors que le champ du possible reste ouvert). De fait, ces outils se développent surtout que dans des contextes à la fois suffisamment réglementés pour contraindre les pratiques et suffisamment prospères pour supporter des coûts de développement informatique importants (pharmaceutique, aéronautique...).

Les alternatives WYSIWYM permettent au contraire d'imbriquer d'emblée la couche de contrôle et de gestion et la couche d'écriture, chaque fois que l'un interagit avec l'autre.

Transclusion

La transclusion est la possibilité pour un premier document informatique de référencer un second document (ou fragment de document) de façon à l'afficher comme s'il était inclus dans le premier. C'est ce qui se passe par exemple quand une page Web affiche une image ou une vidéo : la page référence l'image, qui peut être sur un autre serveur, et celle-ci est affichée à l'intérieur, comme si elle en faisait partie. Le premier bénéfice de la transclusion et de pouvoir réutiliser des ressources sans les recopier. Ces fonctions sont marginalement disponibles dans les outils bureautiques, pour permettre la gestion déportée des images par exemple, mais leur usage est à la fois complexe et peu fiable, ce qui fait que le copier/coller des parties réutilisées est la pratique dominante. Ce qui est copié est bien ici, je peux le voir et le manipuler comme ce que j'écris. Une fois encore le WYSIWYG fonctionne à condition de rapporter toutes les manipulations à la logique de l'écriture traditionnelle, de dissoudre le computationnel dans le graphique. Mais cette dissolution n'est pas sans perte.

« Avez-vous déjà essayé de créer une base de plusieurs documents Word et de les recombiner entre eux par des inclusions dynamiques ? Tous ceux qui s'y sont vraiment frottés vous diront que c'est une galère infernale ! En effet, vous ne pouvez pas vouloir à la fois maîtriser votre publication (gestion des hauteurs, des sauts de ligne et page pour du papier, gestion de la taille écran pour du web) et bénéficier du principe d'un contenu modifié à un seul endroit qui se répercute à tous les endroits où il est exploité. *(Ibid.)* »

Et aussi : multimédia, scénarisation, accessibilité, instantanéité, glose...

Ces réflexions permettent d'établir une première liste des principaux points de tension entre WYSIWYG et écriture computationnelle, qui sont levés par le WYSIWYM. Cette liste n'est pas exhaustive, on pourrait également citer :

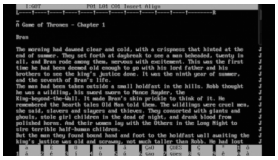
- le multimédia : l'intégration de contenus temporels ou de combinaisons de formes sémiotiques qui ne sont pas toujours présentes ensemble en même temps pose des problèmes de représentation spatiale ;
- la scénarisation : l'écriture de contenus suivant des parcours non linéaires (réseaux de grains pédagogiques, récit suivant plusieurs arbres possibles) s'inscrit et se lit mal, par définition, dans la linéarité de la page ;
- l'accessibilité : la gestion du handicap implique de considérer des éléments de formatage du contenu qui dépassent ce que l'on voit, pour se projeter dans ce que percevra un utilisateur via un dispositif de lecture adapté ;
- l'instantanéité : l'écriture collaborative synchrone fait apparaître des marqueurs (couleurs) qui permettent de différencier les contributions lors de l'écriture, mais ne sont pas publiées ;
- la glose : le contenu n'est pas plus la production d'un auteur unique, mais un amalgame de texte et de paratexte, dont l'agencement et la présentation ne relève plus du seul logiciel d'écriture ;
- ...

Finalement la plupart des fonctions du numérique, en tant qu'avatars du computationnel, font émerger des situations où le graphique du WYSIWYG montre ses limites, et où se construisent les dispositifs d'écriture WYSIWYM.

4 Limites et extensions du WYSWYM

Écriture "zéro computationnel"

George R. R. Martin, l'auteur de *A Game Of Thrones* (Le trône de fer) a expliqué dans une interview utiliser un ordinateur déconnecté sous DOS pour écrire sous WordStar 4.0, parce qu'il voulait un traitement de texte, qui ne fasse "rien d'autre", qui ne lui fournisse aucune assistance, aucune distraction, aucune interférence avec *son* écriture (Holly, 2014).

Le roman et l'écrivain restent ces figures de l'écriture prise dans son sens premier, la représentation d'une langue au moyen de signes, ici alphabétiques, sans dimension multimédia, ni computationnelle d'aucune sorte, avec une structure réduite au minimum, les paragraphes et les chapitres. Une écriture qui se moque du calcul, des lettres qui n'ont rien à calculer. Comme souvent les cas limites de ce genre nous renseignent sur nos pratiques ordinaires.	 Source :

Sylvain Spinelli écrivait en 2006 que le WYSIWYM relevait d'une « démarche inhabituelle [qui] nécessite un travail d'abstraction de la part de l'auteur qui en fonction des situations peut-être contre-productive, car inutile compte-tenu du type d'information à produire (simple, linéaire et statique) et de son exploitation (un seul support associé) ». L'ancrage de la dimension computationnelle dans la pratique de l'auteur exige de fait un effort supplémentaire, puisqu'il s'agit d'écrire tout en programmant. Or il existe des situations, comme celle de l'auteur de roman, qui ne cherchant pas à bénéficier de la dimension calculatoire, ne souhaitent pas en payer le prix. On remarquera que cette exigence conduit George R. R. Martin à rejeter les outils WYSIWYG modernes, déjà trop habités de calculs pouvant interférer avec son écriture. En alternative à l'usage de systèmes minimalistes ou de systèmes des années 1980, il me parait plus fécond de considérer la possibilité de configurer des environnements d'écriture adaptés, c'est à dire réduits à ce que l'auteur souhaite exprimer en terme de structure et de calcul, y compris presque rien donc.

WYSIWYM et environnements juste nécessaires

Nous proposons de considérer une application des environnements d'écriture WYSIWYM dans une logique de juste nécessaire, que l'on peut décliner selon deux hypothèses :

- un éditeur WYSWIYM doit être configuré *contextuellement* à une écriture de façon à minimiser la perturbation et l'apprentissage inhérents à la dimension computationnelle ;
- un éditeur WYSIWYM doit permettre de *sortir* du computationnel pour s'immerger pleinement dans l'écriture dès que c'est possible.

Spécialisation du WYSIWYM et modélisation de l'écriture

« L'approche WYSIWYG amène nécessairement l'auteur à se préoccuper de la forme et du résultat final. On connaît bien la surcharge cognitive et la perte de temps que cela entraîne, Powerpoint étant un must en la matière. (Spinelli, 2006) »

Si le WYSIWYG tend à plonger celui qui écrit dans un maelström de fonctions de mise en forme qui nuisent à son activité, le WYSIWYM porte en lui un risque du même ordre, en transférant du côté du calcul la charge qu'il a abandonnée du côté de la mise en forme. Or l'enjeu est de pouvoir se concentrer sur l'écriture et d'éviter de perdre du temps sur des manipulations contre-productives, qu'elles relèvent du graphique comme de la programmation. La *spécialisation* des éditeurs WYSIWYM permet de répondre à cet enjeu.

La spécialisation consiste à construire un éditeur spécifiquement pour adresser un contexte d'écriture donné : par exemple un éditeur de recette dans un restaurant sera différent d'un éditeur de documentation technique dans une industrie et d'un éditeur de cours dans une université. Par ailleurs, au sein de chaque contexte, les fonctions sont elles mêmes contextualisées : l'écriture d'un cours est différente de l'écriture d'un exercice, l'écriture d'une procédure est différente de la description d'une machine. Or pour réaliser un éditeur spécialisé de la sorte, il faut être capable d'en faire un *modèle*, c'est à dire d'en formaliser les fonctions, pour les anticiper et les rendre disponibles lorsqu'elles sont nécessaires, et, dans la mesure du possible, uniquement à ce moment là.

Les outils WYSYWYG peuvent difficilement s'inscrire dans cette logique car la mise en forme relève d'une pratique informelle difficilement modélisable. En revanche la dimension structurelle de l'écriture WYSIWYM est beaucoup plus facile à représenter formellement et à contrôler automatiquement. Il existe des outils WYSIWYM généralistes, comme LyX, des éditeurs HTML, ou XML basés sur des schémas standard comme Docbook, qui ne s'inscrivent pas non plus profondément dans cette logique de spécialisation. En revanche des outils comme Scenari sont totalement spécialisables.

« Connaissant le contexte métier de l'auteur, le modélisateur peut spécifier l'environnement d'édition juste nécessaire pour permettre à l'auteur de produire son information. De ce fait l'outil s'adapte à l'auteur et non l'inverse, et ceci est un des points fondamentaux qui rendent l'approche WYSIWYM réaliste (Spinelli, 2006). »

Exemples d'éditeurs spécialisés Scenari : Quick, Dila, Ifcam

Exemple Sc 3 éditeurs ()

Exemple Sc contexte dans un cours

La rapport investissement/bénéfice de la spécialisation

Tandis qu'un éditeur généraliste est prêt à l'emploi, un éditeur spécialisé nécessite une étape de modélisation préalable, qui a un coût. Plus un éditeur est généraliste, plus la charge cognitive du côté de l'auteur est élevée (étant donné un certain spectre fonctionnel) ; plus un éditeur est spécialisé plus l'investissement en terme de modélisation est élevé (ainsi que le coût de maintenance de cette spécialisation). Il y a donc un rapport entre cet investissement et le bénéfice en terme de réduction de la charge sur les auteurs qui doit être évalué chaque fois, avant de procéder à une spécialisation. Le bénéfice est évidemment à mesurer pour chaque écrit produit par chaque auteur. Ainsi un écrit correspondant à une structure graphique et/ou computationnelle originale, sera en général mois favorable à la spécialisation que des écrits qui obéissent à des schémas récurrents.

Exemple d'éditeur généraliste Scenari : Opale

Le modèle Scenari-Opale est le modèle le plus répandu dans la communauté Scenari. Or il est un contre exemple du point de vue de la spécialisation. En effet, le modèle fait l'union de ses nombreux contextes d'usage : cellules de rédaction universitaires, centres de formation professionnels, enseignants du primaire et secondaire, approches classiques expositives, approches plus appliquées, évaluation... Autant de milieux et fonctions qui le rendent de fait très généraliste. Il présente donc les mêmes avantages et défauts que les éditeurs WYSIWYM généralistes : il est prêt à l'emploi, mais il demande un investissement plus important en terme d'apprentissage et de charge cognitive à l'usage.

Inventer son écriture en écrivant : œuvres uniques, originalité, contextes artistiques

La modélisation implique une formalisation *préalable* à l'écriture, c'est à dire qu'elle suppose un modèle de structure identifiable qui pré-existe à l'écriture. Si cette hypothèse se vérifie pour la plupart des écrits ordinaires, qui répondent de fait à des codes et cadres déjà existants, il existe des cas où la structure s'invente en même tend que le contenu, par exemple dans le contexte de la littérature numérique ou du marketing. Dans un tel cadre la modélisation et la formalisation qu'elle impose sont contre-productives, puisqu'elles limitent ou ralentissent la créativité recherchée. De tels écrits mobilisent en général des éditeurs de *bas niveau graphique*, c'est à dire qui permettent d'intervenir à un niveau inférieur à celui du caractère, comme les outils de dessin vectoriel ; et/ou des outils de *bas niveau informatique* qui permettent de mobiliser des langages de programmation aux spectres très ouverts. Des technologies représentatives de ces modalités sont historiquement des outils comme Adobe Director puis Flash (et son langage ActionScript), et à présent HTML5 et la suite de technologies intégrées : SVG pour le vectoriel, Canvas pour le bitmap ou CSS et JavaScript pour la programmation.

Sortir du computationnel : environnement "zen"

Le WYSWIYM, en réintégrant la dimension computationnelle de l'écriture numérique, porte un risque de déplacement de l'attention sur la structuration, voire sur la programmation. Dans un environnement d'écriture Scenari, il est typiquement possible de se retrouver avec un contexte fonctionnel assez dense permettant de structurer, de fragmenter, de contrôler, de visualiser, de collaborer. Scenari permet de mettre le focus sur l'édition d'un fragment (une partie du texte), mais l'on pourrait souhaiter un environnement qui soit plus spécifiquement dédié à l'écriture du texte, avec un paramétrage différent en termes graphiques et une épuration fonctionnelle plus poussée. Du côté de la lecture, on retrouve cette approche dans la presse en ligne (interface zen du monde.fr) et au sein des liseuses électroniques.

Environnement d'écriture WYSIWYM "Zen"

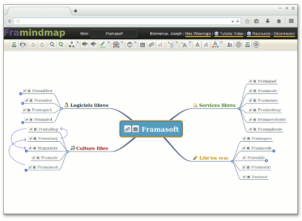
Le premier écran présente un environnement d'écriture WYSIWYM Scenari complet. Le second montre l'interface d'écriture que j'utilise majoritairement, qui se limite à l'écriture d'une sous-partie du texte. Le troisième serait une vue "Zen" - ou une vue "Martin" - très épurée, qui pourrait être activée pour maximiser la concentration sur l'écriture.



Des phases dans l'écriture : l'idéation

La structuration inhérente au WYSIWYM suppose une projection dans son écriture, qui est cohérente avec les phases les plus en aval d'un processus éditorial, lorsque le projet a déjà pris une certaine forme. En revanche, pour les phases plus en amont, cela peut se révéler contre-productif. La phase d'idéation typiquement consiste à stimuler l'émergence des idées au début d'un projet, tout en les organisant petit à petit. L'idéation est aujourd'hui par exemple instrumentée avec des outils de mind-mapping (Framamind) ou des tableaux blancs collaboratif (Scrumblr). Si nous travaillons pour le moment sur des passerelles possibles entre ces outils et les chaînes éditoriales WYSIWYM (Hdoc Converter), il sera certainement intéressant que ces dernières intègrent des environnements d'écriture permettant la malléabilité et la dé-structuration nécessaires à l'idéation, tout en s'intégrant avec les phases suivantes de structuration.

Outils d'idéation

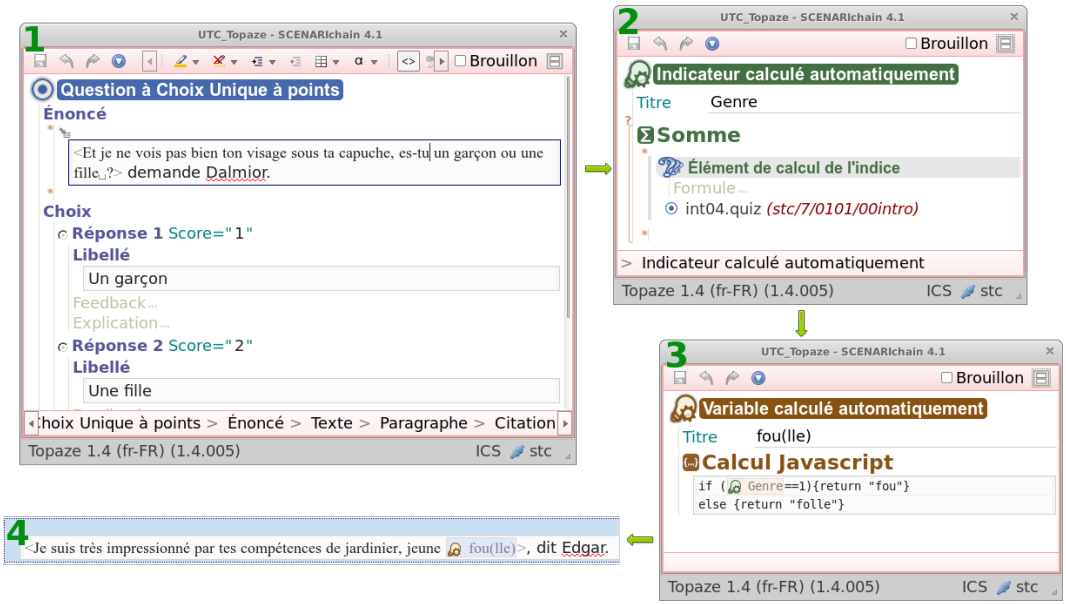


L'écriture essentiellement graphique

L'abstraction et la description de la structure est ce qui permet à la machine de calculer, la représentation graphique est ce qui permet à l'homme d'interpréter. Or il est des structures fondamentalement graphiques, pour lesquelles la recherche d'une abstraction ne fonctionne pas, c'est à dire que ce n'est pas efficiente (le rapport entre l'effort demandé à l'auteur et le gain en terme de manipulation n'est pas intéressant), voire n'est pas possible (l'auteur en cherchant à abstraire perd sa capacité d'expression). On citera deux exemples : les schémas et les tableaux. Pour ces deux cas typiquement graphiques il existe quelques situations pour lesquelles il est intéressant de proposer des éditions WYSIWYM, par exemple des schémas de procédures organisationnelles ou des tableaux de comptabilité. Mais en général les éditeurs WYSIWYM gagnent à repasser dans un mode purement WYSIWYG (dans Scenari cela se manifeste par l'intégration d'objets LibreOffice Calc et Draw, deux outils de dessin, vectoriel et tabulaire).

L'écriture essentiellement computationnelle

À l'inverse, on observe dans d'autres cas la nécessité de passer dans un mode de programmation de plus bas niveau. Certaines fonctions ne s'instancient que dans une couche de programmation plus explicite que celle habituellement mobilisée dans le WYSIWYM. Ainsi, pour programmer des scénarios complexes au sein de la chaîne éditoriale Topaze - multiplicité des chemins possibles, passages conditionnés, personnalisation des messages - il est nécessaire que l'auteur manipule des outils informatique, tels que des variables ou des conditions de type « si alors sinon ».



http://7.crzt.fr/0101

Dans cet exemple de type « livre dont vous êtes le héros », (1) un quiz permet de demander au lecteur s'il est un garçon ou une fille ; (2) puis d'instancier une variable « Genre » en fonction de la réponse à la valeur 1 (garçon) ou 2 (fille) ; (3) puis de calculer un texte en fonction de cette variable, ici le mot « fou » ou « folle » ; (4) pour enfin utiliser ce texte calculé : « Je suis impressionné par tes compétences de jardinier, jeune #fou(lle) ».

Conclusion : Pourquoi le WYSYWYG gagne encore ?

« Pour conclure : cela fait 20 ans maintenant que l'approche WYSIWYG est mise en oeuvre (avec des moyens financiers colossaux), aboutissant à une très grande maturité et stabilité technologique. Mais des nouveaux besoins massifs de gestion/diffusion de l'information combinés à l'avènement de l'interactivité et du multimédia mettent en évidence les limites intrinsèques de cette approche. (Spinelli, 2006) »

« le WYSIWYM est un concept émergent sur le quel nous avons encore peu de recul et peu de moyens ont été investis. Les précurseurs de cette approche sont les éditeurs historiques en ligne de commandes, mais la non exploitation de la force de la représentation graphique les cantonnent dans des milieux spécifiques où l'abstraction est reine : les math et l'informatique. Je crois profondément que WYSIWYM intégrant la dimension graphique est le ticket gagnant sur lequel il faut aujourd'hui investir ! (Ibid.) »

VI Notion de littératie numérique

Enjeux de l'écriture computationnelle : la littératie numérique ou la prolétarisation

Le dualisme écriture-programmation propre à l'écriture numérique repose la question de la littératie, c'est à dire des compétences de lecture et d'écriture qu'il est nécessaire de maîtriser pour manipuler l'information numérique.

Les conditions de l'écriture numérique : accéder au computationnel

Il faut toujours la même appréhension au niveau sémiotique, mais il faut donc en plus une appréhension du niveau calculatoire qui sous-tend le niveau sémiotique que l'on perçoit ; et il faut une appréhension de l'articulation entre les deux niveaux. Cette nécessité d'*appréhension* porte la nécessité d'une pratique réfléchie pour devenir intuitive, accessible à la pensée pour être contrôlée. Il faut comprendre pour anticiper ce que l'on produit, la nature de ce que l'on peut faire et de ce que l'on ne peut pas faire, de ce que ça va devenir, des opérations que ça peut subir.

La littératie numérique implique donc la possibilité d'appréhender le niveau calculatoire, le niveau sémiotique et l'articulation entre les deux.

Un exemple : Framapad

Prenons comme exemple l'écriture via un éditeur de texte très simple - en apparence au moins - comme Framapad, une instance d'Etherpad hébergée par Framasoft que l'on peut utiliser directement en ligne, sans identification préalable.

Au premier abord l'écriture semble essentiellement graphique, orientée vers une production de signes alphabétique, complétée par de la mise en forme basique (gras, italique). De fait les documents produits avec Framapad sont simples. Et pourtant, si l'on y regarde de plus près...

- Le simple fait de saisir une adresse web produit un lien cliquable, sans qu'il soit besoin de programmer cette interaction (c'est le cas aujourd'hui dans la majorité des éditeurs de texte). Se pose donc d'emblée la question du bon usage des ces liens, problématique bien connue de l'écriture numérique.
- Framapad est un logiciel en réseau, ce qui est écrit est donc sur un serveur distant, ce qui impacte la disponibilité de ce contenu (il faut une connexion Internet et que le serveur soit fonctionnel) ou sa confidentialité (le contenu est public par défaut).
- C'est un logiciel collaboratif synchrone, donc d'autres utilisateurs peuvent modifier le contenu, y compris pendant que j'écris.
- Par ailleurs tout ce qui est écrit dans Framapad est historisé automatiquement, et il est possible de rejouer la séquence d'écriture depuis le début, y compris ses fautes, ses hésitations, et bien sûr ces quelques mots géniaux que l'on avait cru perdus.

Ainsi donc, ce qui au premier abord pourrait sembler relever d'une écriture très traditionnelle, relève finalement largement aussi du computationnel. Cette analyse superficielle montre la nécessité de se questionner sur ce qui se passe au niveau logique du numérique. On notera que la superficialité de cette analyse fait ici son intérêt, elle montre que les fonctions computationnelles sont mal tapies, prêtes à surgir dès que l'on s'attarde un peu sur le chemin, facile à débusquer à qui les cherche un peu.

Littératie et technologie

Ce que je propose ici relève donc d'une démarche d'étude de la dimension technique des outils qui permettent de produire les textes, une démarche *technologique* au sens étymologique.

Pour bien écrire avec les outils numériques, ma thèse est qu'il faut comprendre le numérique dans sa dimension informatique : les algorithmes, les variables, le stockage, la transmission, la recherche, l'identification, la modélisation...

Ceci entraîne nécessairement une complexification et une abstraction des connaissances nécessaires à l'écriture, puisque l'on ajoute le niveau logique au niveau sémiotique, sans rien retrancher.

Des outils et des hommes

Une telle littératie numérique permet de comprendre les lois numériques qui gouvernent notre rapport au monde numérisé, comme les lois physiques gouvernent notre rapport au monde non numérisé. Elle permet de ne plus être le simple utilisateur de cette phrase érigée en bouclier : « je ne suis qu'un simple utilisateur ».

Analphabète numérique on est relégué au rang d'utilisateur de services, on est prolétarisé (Stiegler, 2006), c'est à dire que l'on est privé d'un savoir et d'un savoir-faire que l'on possédait avant, que l'on ne peut plus décider de ce que l'on fait de notre écriture numérisée, de nos relations numérisées au monde. On est servi, et asservi, par des logiciels qui commande à nos pratiques, des instruments qui nous instrumentent.

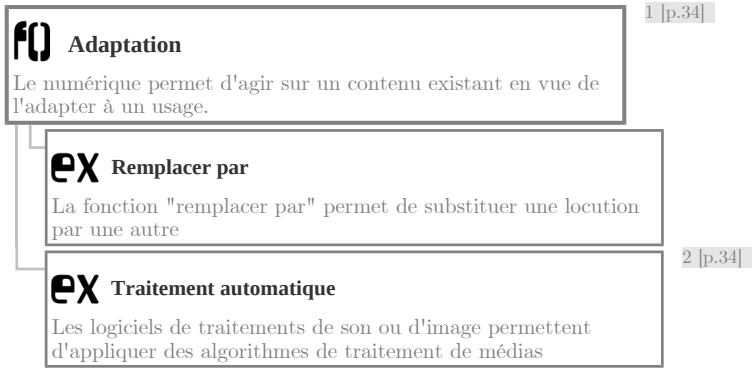
Le choix des outils que nous concevons et utilisons pour écrire est donc fondamental parce qu'ils conditionnent notre écriture, ils sont prescripteurs de possibles.

Le développement d'une littératie numérique nous permet de reprendre la main sur ces choix, de ne plus être simplement consommateur d'une informatique qui nous dépasse, de ne plus laisser uniquement le *marketing* décider de nos modes d'écriture à notre place.

Etherpad ou Scenari, Powerpoint ou Writer, Gmail ou Thunderbird, Facebook ou Twitter, et l'indénombrable légion des autres logiciels qui existent ou qui existeront, peuvent être vus, doivent être vus, comme des logiciels d'écriture *et* comme des langages de programmation, dont il faut alors connaître les arcanes pour garder la maîtrise de son écriture. Mais comme il est impossible d'apprendre et de réapprendre en permanence les spécificités techniques et fonctionnelles de chaque logiciel, il est nécessaire de se doter de fondamentaux théoriques et techniques qui relèvent du numérique en général et sont donc transportables d'un logiciel à un autre.

Annexes

Annexe 1



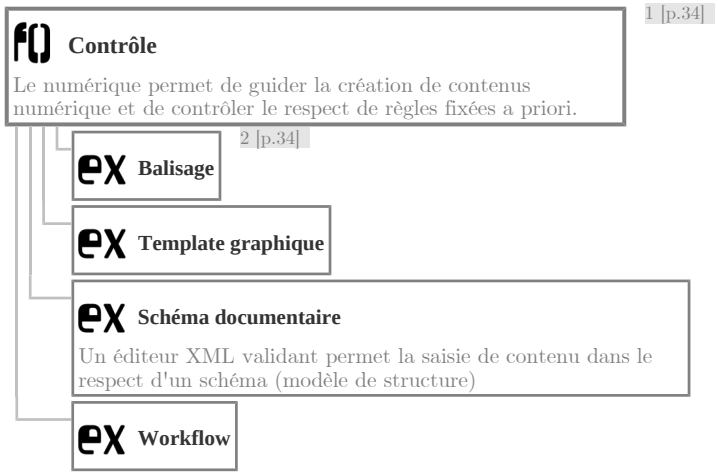
1 **Adaptation**

« Action d'agir sur une substance ou un produit en vue de le modifier et de l'adapter à un usage déterminé (TLFi). »

2 **Traitement automatique**

Suppression de bruit de fond pour un son, de flou pour une image, calcul de transition entre deux images pour une vidéo...

Annexe 2



1 **Contrôle**

Un modèle est une représentation formelle du contenu, de sa structure et/ou de son contenu. Un modèle est calculable, interprétable par la machine, il permet d'écrire des programmes pour contrôler le contenu.

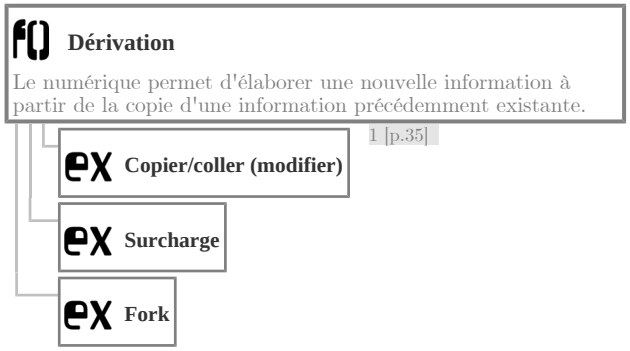
2 **Balisage**

Langage de présentation à balises (HTML, LaTeX)

Dessin vectoriel (SVG)

Publication multimédia (SMIL)

Annexe 3

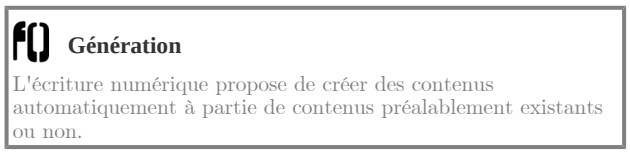


1

Copier/coller (modifier)

La fonction est devenu constitutive de la manipulation numérique : le copier/coller suivi de modifications afin de créer de l'original à partir du déjà existant.

Annexe 4



1 [p.35]

1

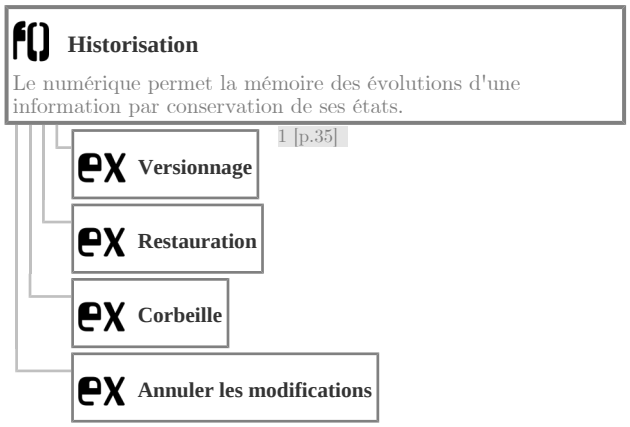
Génération

Synthèse d'une version courte d'un texte à partir d'un texte original.

Extraction d'une image à partir d'une vidéo, d'une icône à partir d'une image

Synthèse vocale à partir d'un texte

Annexe 5



1

Versionnage

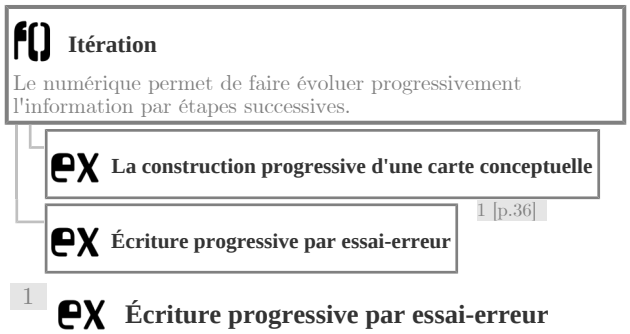
Le versioning des systèmes de GED permet de conserver (automatiquement ou manuellement) des copies (versions) correspondant à un état identifié d'un contenu, la version est une étape donnée de sa genèse

le versioning des systèmes de gestion de code informatique permet de conserver les programmes à différents états de leur développement (SVN, Git...)

L'historisation automatique des wikis (outils d'écriture collaborative asynchrone), permet de remonter à n'importe qu'elle sauvegarde antérieure d'un contenu

L'historisation automatique dans les outils d'écriture collaborative synchrone (Google Doc, Etherpad...) permet de remonter à n'importe quel état antérieur du contenu

Annexe 6



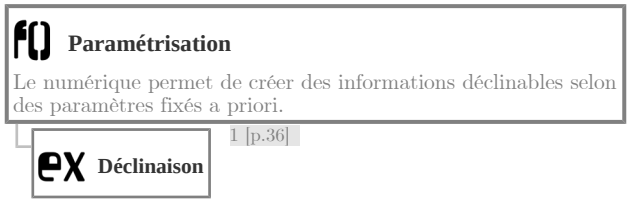
1

Écriture progressive par essai-erreur

Un processus rédactionnel "délinéarisé", au sens où les étapes de production n'ont plus à être réalisées dans un ordre donné (planification, rédaction, révision) : l'écrit est en permanence modifiable tant au niveau du contenu (suppressions, ajouts, déplacements,...) qu'à celui de sa mise en forme.

Déplacer des fragments, afin de réordonner un discours dynamiquement, au cours de l'écriture

Annexe 7



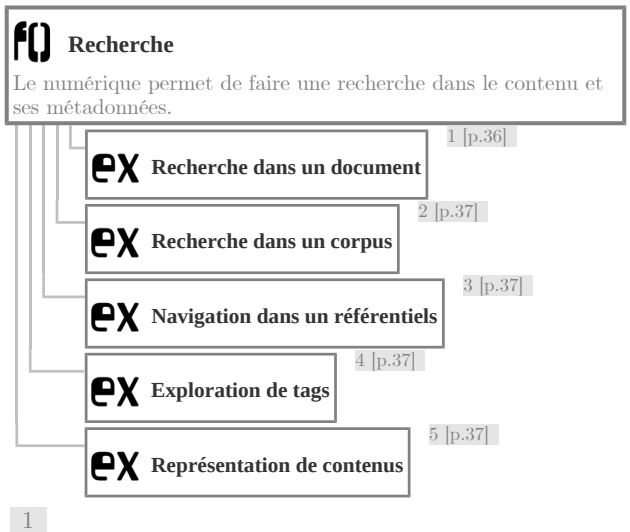
1

Déclinaison

Intégration de variables issues d'une base données dans une recette : « introduire *$quantité* grammes de sucre ... », avec $quantité une variable qui sera remplacée par une valeur numérique issue d'une autre source au moment de la publication

Image annotée avec des valeurs paramétrées pour gérer le multilinguisme

Annexe 8



1

Recherche dans un document

Rechercher dans la page (CTRL+F)

2 **Recherche dans un corpus**

Recherche par mots clés

Indexation automatique plein texte

commande grep

3 **Navigation dans un référentiels**

Catalogue

Terminologie

4 **Exploration de tags**

Folksonomies

5 **Représentation de contenus**

Ontologie

Annexe 9

Asynchronisme

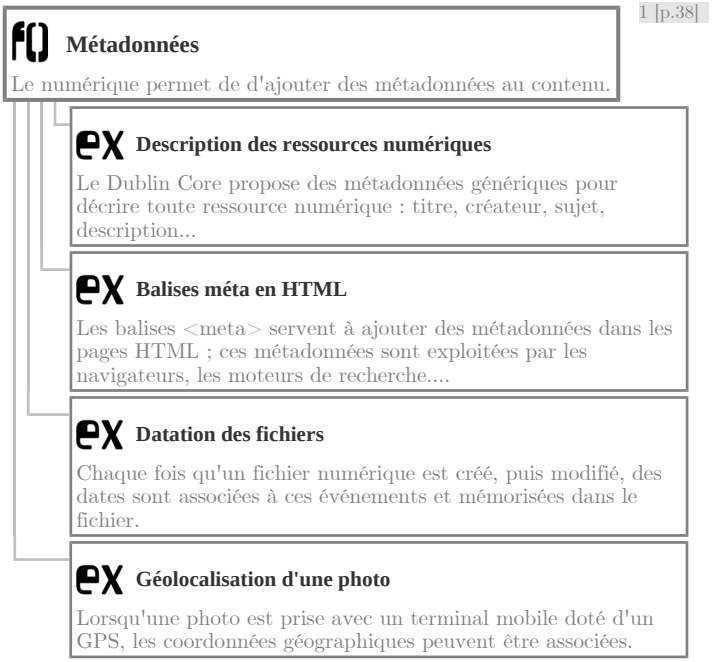
Le numérique permet à plusieurs personnes de travailler successivement sur un même objet, afin de le construire à plusieurs.

Annexe 10

Instantanéité

Le numérique permet à plusieurs personnes d'accéder et de modifier un même objet en même temps.

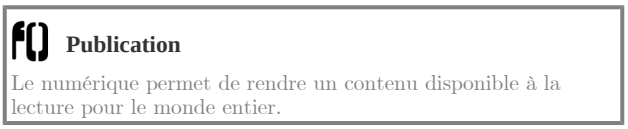
Annexe 11



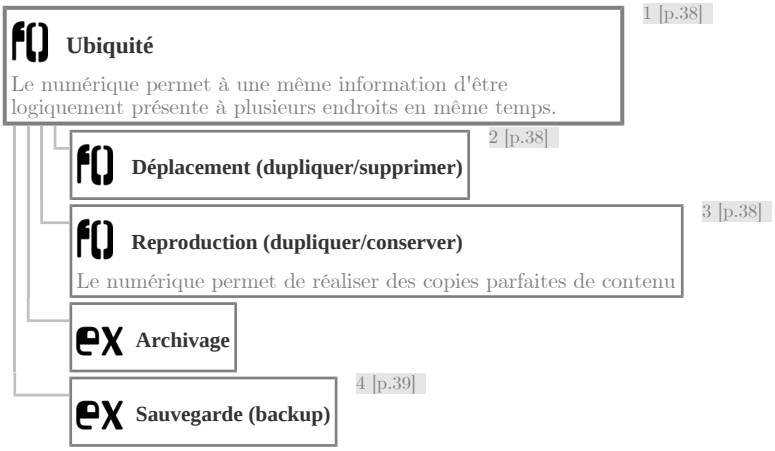
1 **Métadonnées**

Il est possible de stocker, dans un fichier, en plus du contenu lui même, des métadonnées relatives à ce contenu. Ces métadonnées sont ainsi disponibles pour toutes les applications en connaissant le format.

Annexe 12



Annexe 13



1 **Ubiquité**

L'original et la copie étant indifférenciés, il est logiquement possible de considérer qu'un même contenu se trouve en plusieurs endroits.

2 **Déplacement (dupliquer/supprimer)**

Déplacer des fichiers afin de les organiser

3

Reproduction (dupliquer/conserver)

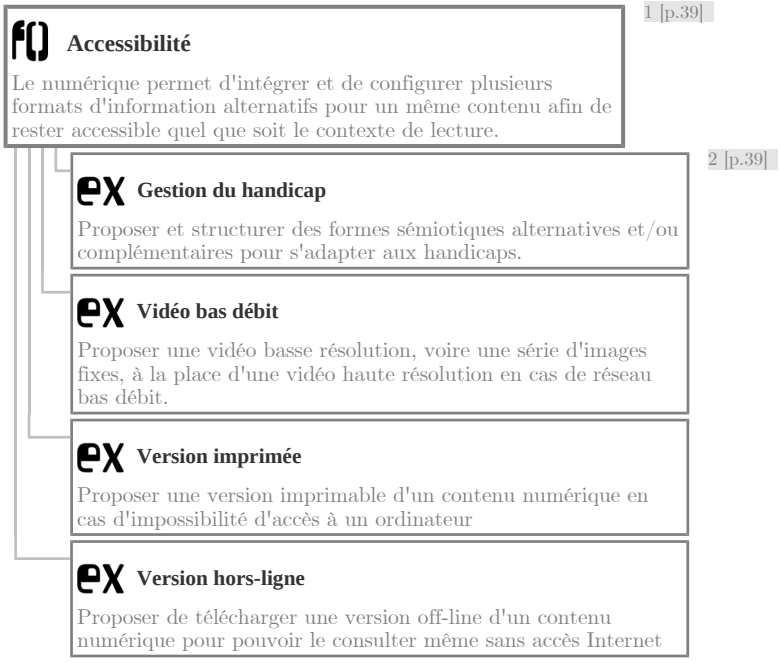
Tout contenu peut être dupliqué autant de fois qu'on le souhaite, sans effort, et sans pouvoir différencier une copie de l'original. La notion même d'original n'a pas de sens, autre que généalogique (à condition que cette mémoire d'antériorité soit conservée).

4 **Sauvegarde (backup)**

Fonction "sauvegarder sous"

Copier/coller un fichier sur un support de sauvegarde

Annexe 14



1 **Accessibilité**

L'accessibilité consiste à mettre les « contenus à la disposition de tous les individus, quelque soit leur matériel ou logiciel, leur langue maternelle, leur culture, leurs infrastructures réseau ou leurs aptitudes physiques ou mentales ». (d'après T. Berners Lee, cité par Sloïm et Crozat (2008)).

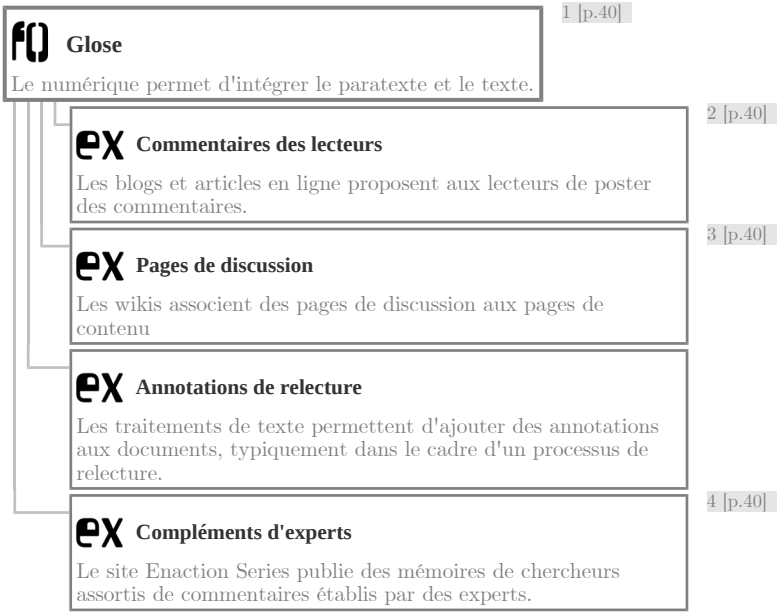
Module de sensibilisation : Exemples et contre exemples d'accessibilité des documents numériques

2 **Gestion du handicap**

Description textuelle d'une image (pour les malvoyants)

Vidéo associée à des sous-titres (pour les malentendants)

Annexe 15



1 **Glose**

Tout contenu numérique peut être augmenté de paratexte (produit par l'auteur lui-même, des co-auteurs, des critiques, des lecteurs...) et ce paratexte peut rester solidaire du contenu tout au long de son cycle de vie.

2 **Commentaires des lecteurs**

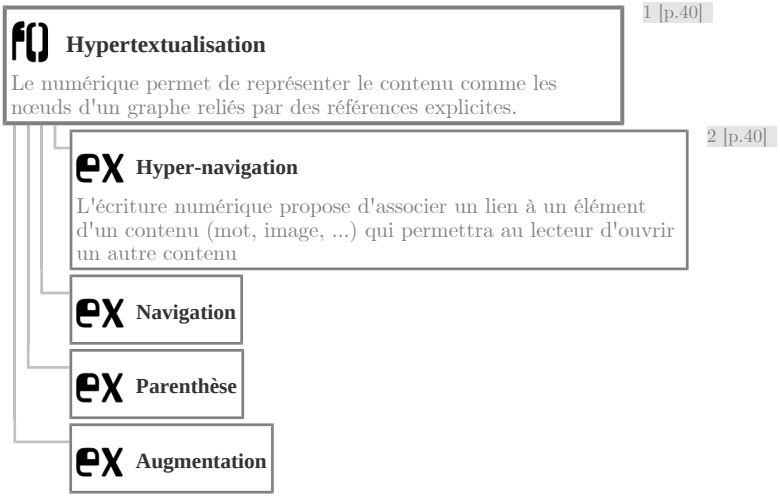
Le système Disqus permet d'associer des fils de commentaire à toute page Web

3 **Pages de discussion**

La page de discussion de la page Wikipédia du mot *document*.

4 **Compléments d'experts**

Annexe 16



1 **Hypertextualisation**

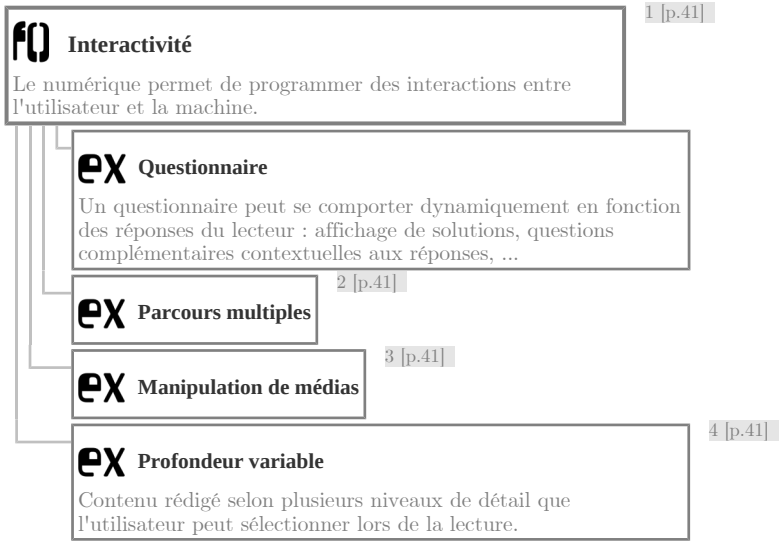
Chaque nœud du graphe est un contenu, qui dispose de références sortantes vers d'autres contenus et de références entrantes depuis d'autres contenus.

2 **Hyper-navigation**

La balise <a/> d'une page HTML permet de pointer un autre site Web

Il est possible de commander l'ouverture d'un second document dans un document bureautique via un clic sur un mot

Annexe 17



1 **Interactivité**

L'interactivité est une transformation dynamique de l'information disponible en réaction à des actions effectuées par l'utilisateur.

2 **Parcours multiples**

Un scénario de lecture peut s'adapter en fonction d'un niveau déclaré par le lecteur (novice, confirmé, ...)

3 **Manipulation de médias**

Faire tourner un objet en 3D

Se positionner sur un objet temporel (audio, vidéo)

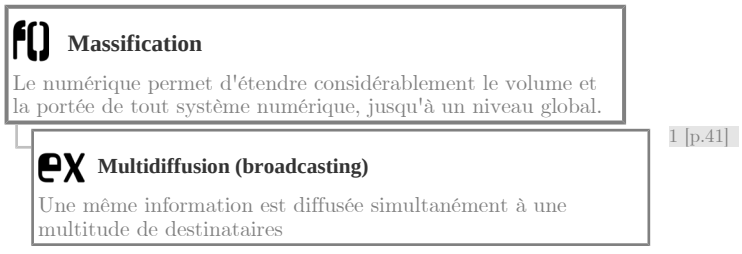
Zoomer une image

4 **Profondeur variable**

A0, A1 ... An, sont des contenus de plus en plus détaillés d'une même information et le système propose une navigation adéquate au lecteur pour passer de l'un à l'autre en fonction de l'intérêt porté au contenu

Vidéo épaisse : version résumée, montée, avec *rush* d'un même reportage, que l'utilisateur peut mobiliser en fonction de son intérêt à l'instant t.

Annexe 18

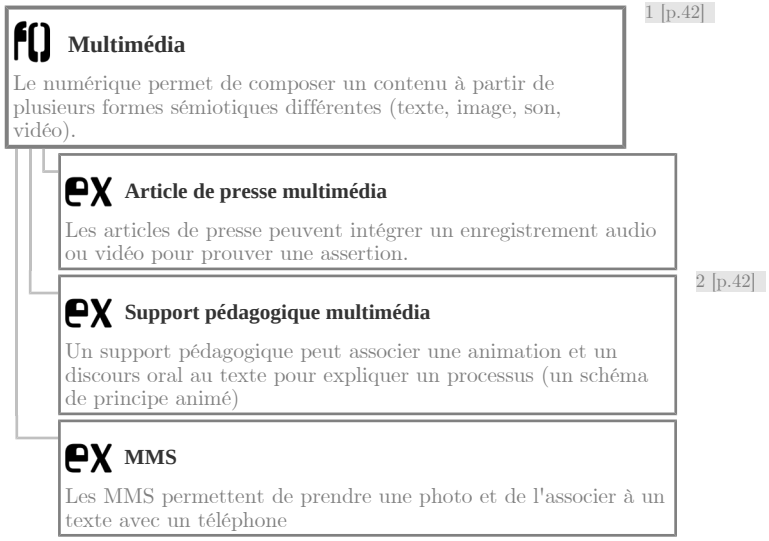


1 **Multidiffusion (broadcasting)**

La reproduction du contenu étant sans perte et sans coût, il est possible d'en fabriquer et d'en diffuser

des copies aussi nombreuses qu'on le souhaite.

Annexe 19



1 [p.42]

2 [p.42]

1 **Multimédia**

Le numérique permet d'associer sur un même support des médias de différente nature (texte, son, image, vidéo...) pour constituer un discours cohérent, mais hétérogène dans la forme. Les formes sémiotiques mobilisées peuvent être spatiales (texte, image), temporelles (son) et spatio-temporelles (vidéo).

2 **Support pédagogique multimédia**

La chaîne éditoriale Opale permet la création de contenus multimédia

Exemple de contenu textuel intégrant une animation

Exemple de contenu textuel intégrant une version "audio-book"

Exemple de conférence audiovisuelle enregistrée enrichie de titres, notices et mots clés.

Annexe 20

1 [p.42]

2 [p.43]

3 [p.43]

1 **Polymorphisme**

Le format du contenu tel qu'il est stocké diffère nécessairement de son format de présentation, car le format de stockage n'est pas pas directement intelligible par l'homme. Un calcul permet de construire une forme sémiotique pour l'appréhension humaine. Si l'on applique plusieurs calculs différents on obtient

autant de formes différentes de ce contenu.

2 **Feuilles de style**

Les traitements de texte permettent l'édition de feuilles de style

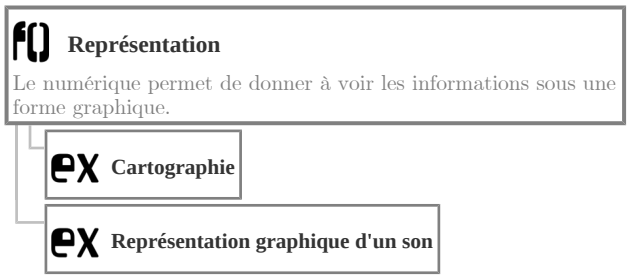
Le standard CSS permet d'associer des styles aux pages HTML

3 **Multisupports**

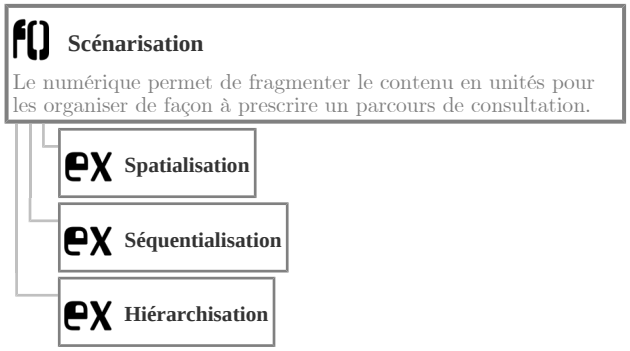
La langage XSL-XSLT permet de transformer un contenu XML en HTML pour le web ou en XSL-FO pour l'impression.

Principe des chaînes éditoriales

Annexe 21

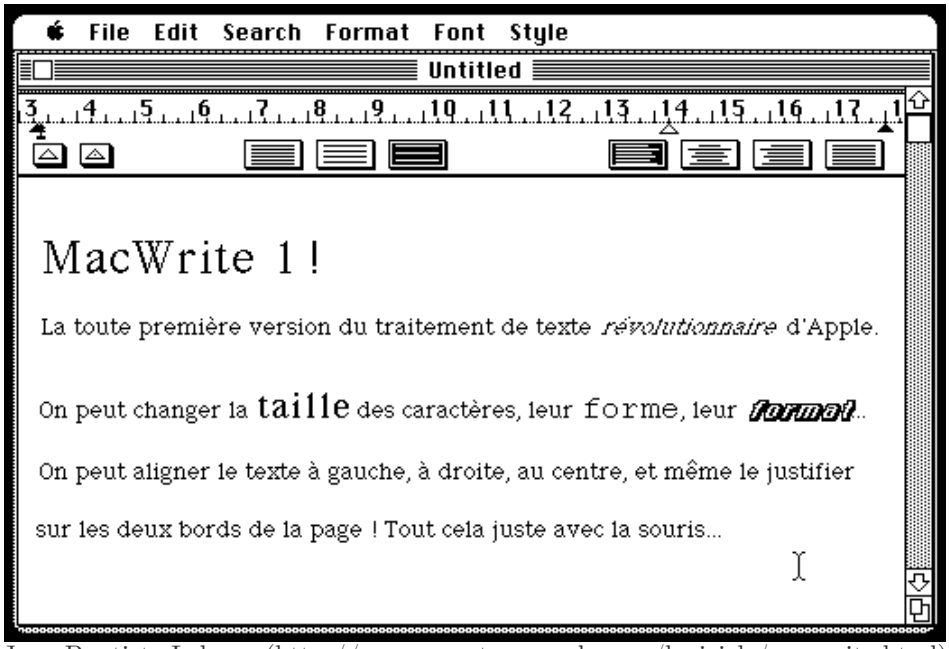


Annexe 22



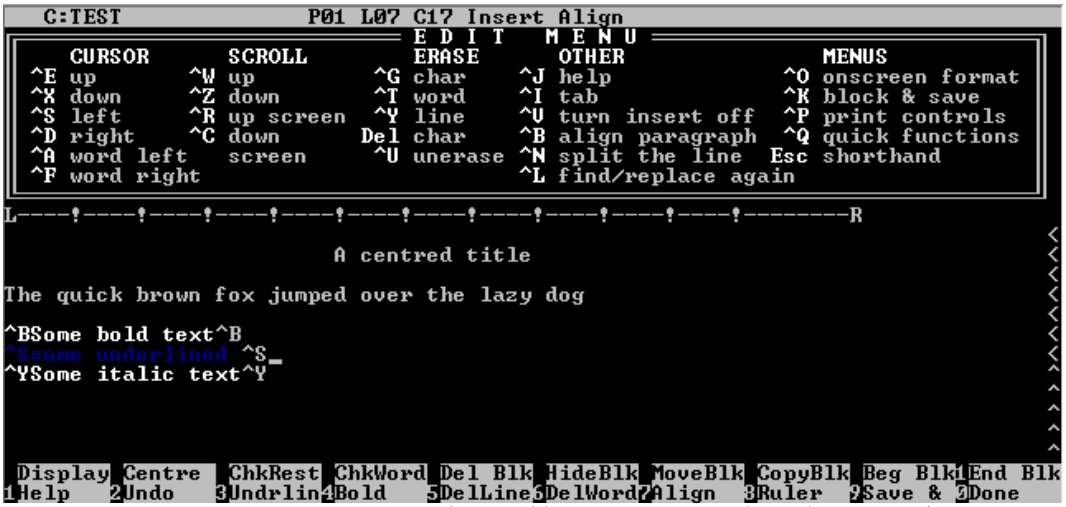
Annexe 23

Annexe 24



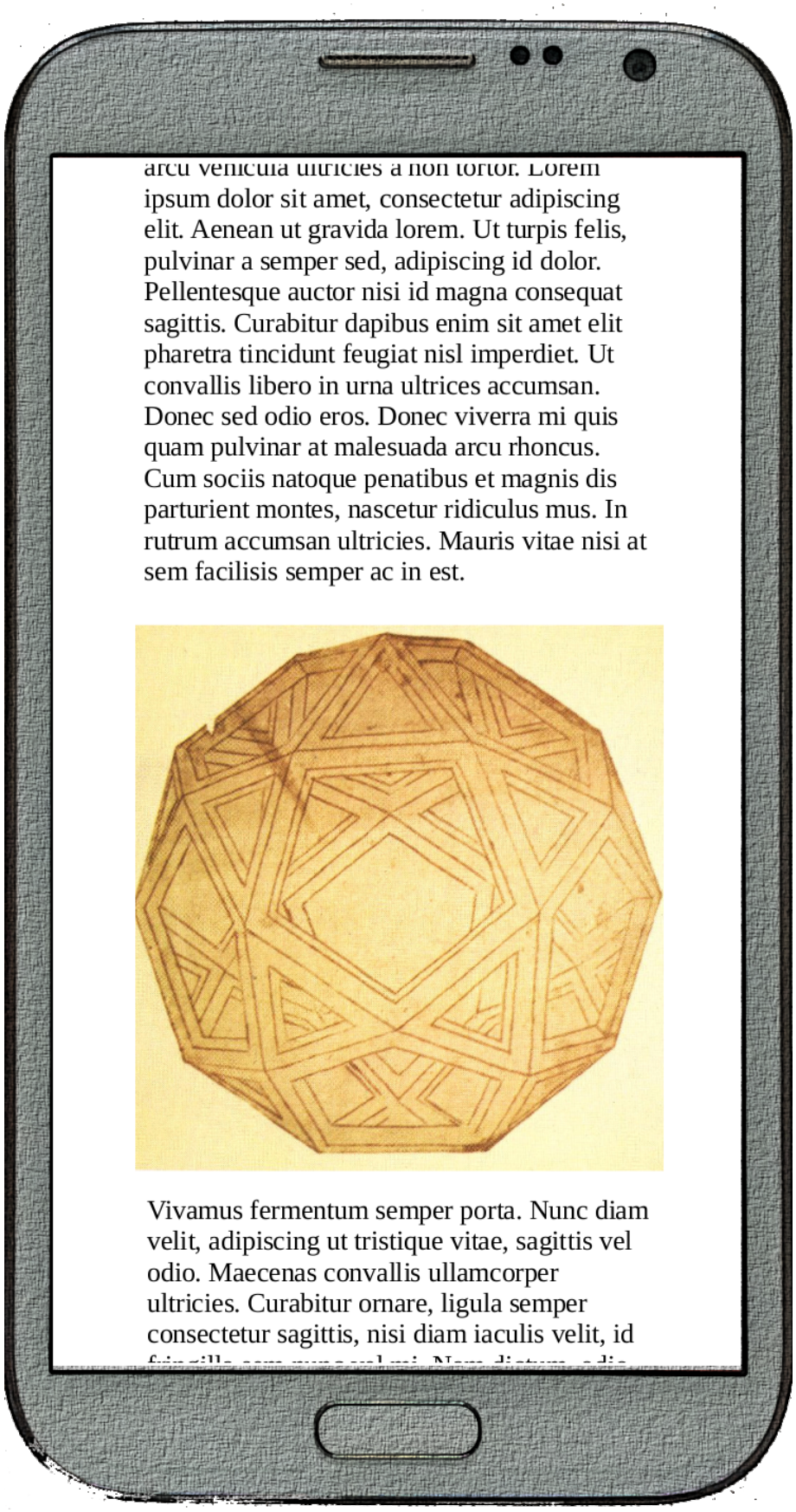
Jean-Baptiste Leheup (http://www.aventure-apple.com/logiciels/macwrite.html)

Annexe 25



Wordstar running in DOS (https://en.wikipedia.org/wiki/WordStar)

Annexe 26


arcu vehicula ultricies a non tortor. Lorem
ipsum dolor sit amet, consectetur adipiscing
elit. Aenean ut gravida lorem. Ut turpis felis,
pulvinar a semper sed, adipiscing id dolor.
Pellentesque auctor nisi id magna consequat
sagittis. Curabitur dapibus enim sit amet elit
pharetra tincidunt feugiat nisl imperdiet. Ut
convallis libero in urna ultrices accumsan.
Donec sed odio eros. Donec viverra mi quis
quam pulvinar at malesuada arcu rhoncus.
Cum sociis natoque penatibus et magnis dis
parturient montes, nascetur ridiculus mus. In
rutrum accumsan ultricies. Mauris vitae nisi at
sem facilisis semper ac in est.
Vivamus fermentum semper porta. Nunc diam
velit, adipiscing ut tristique vitae, sagittis vel
odio. Maecenas convallis ullamcorper
ultricies. Curabitur ornare, ligula semper
consectetur sagittis, nisi diam iaculis velit, id

Annexe 27

cf. Exemple2-HTML.eWeb

Annexe 28

cf. Exemple3-HTML.eWeb