

Maréchal Anthony  
Maréchal Benoît

# Projet IHM

## Création de cartes en tout genre



Créa 2K6

# Sommaire

## Introduction

### I. Cahier des charges

1. Choix parmi les sujets
2. Description des fonctionnalités choisis
  - 2.1. Convivialité et intuitivité
  - 2.2. Les fonctionnalités de bases
  - 2.3. Les fonctionnalités avancées

### II. Analyse

1. Modélisation
  - 1.1. Gestion des objets
  - 1.2. Gestion des Ressources images
  - 1.3. L'architecture générale
2. Messages gérés et enchaînement des traitements
  - 2.1. Gestion des événements souris
  - 2.2. Gestion des événements clavier
  - 2.3. Gestion des événements de menu
3. Descriptions des ressources
  - 3.1. Boite permettant l'insertion de ressources images
  - 3.2. Boite de dialogue permettant l'édition de texte
  - 3.3. Boite de dialogue permettant l'insertion de texte prédefinie

### III. Programmation

1. Méthode de dessin de CTexte
2. La gestion du Format de la carte et l'affichage des éléments dans la vue
3. La méthode listeSousDomaine de la classe CLesImagesRessources
4. L'impression
5. Le déplacement dans CCrea2K6

### IV. Problèmes rencontrés

## Conclusion

Annexe 1 : Manuel utilisateur

Annexe 2 : Etude de l'offre existante

# Introduction

Le module d'interface homme machine du semestre cinq aboutit cette année sur un projet devant être réalisé avec Visual C++ et les MFC (Microsoft Foundation Class). Nous avions le choix parmi de nombreux sujets pour réaliser ce projet. Ce rapport a donc pour but de préciser quel est le sujet que nous avons choisi de réaliser, puis de fournir une analyse afin d'expliquer le fonctionnement de l'application ainsi que de donner les détails concernant la programmation avec les MFC.

Pour effectuer ce travail nous nous sommes, répartis les tâches de manière à ce qu'une seule personne travail en même temps sur le projet, afin d'augmenter la facilité d'implémentation des différentes fonctionnalités et pour éviter des "copier/coller" généralement inefficace dans un projet sous Visual C++.

Nous allons, dans un premier temps, étudier quelles sont les fonctionnalités que le sujet nous impose d'insérer dans l'application, grâce à l'élaboration du cahier des charges. Ensuite, la partie Analyse expliquera les choix de modélisation des classes et l'enchaînement des événements nécessaire à l'implémentation des fonctionnalités retenus ainsi que les ressources créées pour assurer une interaction conviviale. La dernière partie sera consacrée à l'explication des classes spécifiques à l'application, ainsi qu'aux algorithmes intéressants et à la description des gestionnaires de messages.

En annexe vous trouverez une étude de l'offre existante en matière de création de cartes, qui nous a permis de retenir quelles étaient les meilleures solutions pour réunir convivialité, efficacité et ergonomie au sein de notre programme. Finalement un dossier utilisateur clôturera ce rapport ce qui permettra une prise en main rapide du programme.

## I. Cahier des charges

### 1. Choix parmi les sujets

Parmi les nombreux sujets que nous pouvions choisir, nous avons retenu le Sujet cinq : *Création de carte de vœux, de visite ou de billet de spectacle*. Nous l'avons modifié, pour en réalité permettre la création de carte de vœux, de visite, d'invitation ou de carte postale. Nous avons préféré ce sujet, aux autres pour plusieurs raisons. Tout d'abord par rapport à sa cohérence avec les notions que nous avons étudiées lors du module d'IHM, il nous a en effet permis de revoir et d'approfondir les concepts abordés en cours. Ensuite il s'agit de raisons plus personnelles. Passionné par la création et habitué des logiciels d'infographies ainsi que des sites web proposant la création d'e-cartes, ce sujet a été l'occasion de voir les coulisses de la conception de telles applications. Enfin nous n'avons pas choisi un sujet sur la création d'un jeu vidéo car nous en avions déjà réalisé un lors de notre première année.

### 2. Description des fonctionnalités choisis

Nous allons voir dans cette partie les différentes fonctionnalités que nous avons retenues pour la conception de notre application.

## 2.1. Convivialité et intuitivité

Comme pour tous nos projets d'IHM nous attachons une part importante à la convivialité et à l'intuitivité. Pour cela nous avons suivi les cinq directives suivantes :

- *Simplicité et respect des standards.* Nous avons repris toutes les « ficèles » habituelles des logiciels d'infographies sous Windows. Comme par exemple la sélection des objets de la vue par un premier clique, puis leurs déplacement avec un glisser / déposer. L'interface générale étant générée par l'architecture SDI, ce qui nous a permis de respecter les standards Windows.
- *Ergonomie*, avec la gestion d'un menu contextuel en fonction de l'objet sous la souris lors d'un clique droit, la création d'une barre d'outil proposant des raccourcis pour l'ajout de textes et d'images, ainsi que la conception de boîtes de dialogues offrant des fonctionnalités avancées d'édition de texte et d'ajouts de nouvelles images.
- *Efficacité et rapidité*, avec l'ajout de raccourcis claviers comme le déplacement des objets avec les flèches directionnelles ou la barre d'espace pour la mise en arrière plan couramment sélectionné.
- *Assistance*, grâce à l'aide disponible dans le menu ‘?’ et la définition systématique d'un commentaire dans la barre de statut lorsque le pointeur de la souris est au dessus d'une actions de menu, ou d'un raccourci de la barre d'outil.
- *Esthétisme*, avec une interface claire, concise et agréable à l'œil, comme vous pourrez le juger sur les screenshots réalisés dans le manuel utilisateur et dans la description des ressources.

## 2.2. Les fonctionnalités de bases

La création de différentes cartes impliquent d'implémenter plusieurs fonctionnalités de bases elles qui ont donc été implémentées en premier lieu dans notre programme, il s'agit de :

- L'ajout d'images et de textes
- La mise en forme du texte
- La gestion des déplacements
- La sélection d'un objet et sa mise en arrière plan
- La définition d'une couleur d'arrière plan

## 2.3. Les fonctionnalités avancées

Pour faire du programme une application complète de création de cartes il faut bien plus que ces simples fonctionnalités de bases. Nous avons donc développé d'autres fonctionnalités beaucoup plus avancées comme :

- L'apport de *ressources* (images et polices prédefinies) classés selon des thématiques définies et facilement insérable dans la carte.
- La gestion de *l'enregistrement* pour pouvoir sauvegarder une carte en cours de création et la finir plus tard.
- La gestion de *l'impression* de la carte, primordiale dans un logiciel de conception de cartes.

- La gestion de différents *formats* de cartes, avec une interface adaptée en conséquence. Seul les éléments visibles dans le rectangle, représentant le format de la carte doivent être dessinés à l'écran.
- La création d'un affichage, dans le coin supérieur gauche de la vue, des *informations* sur l'élément couramment sélectionné.

## 2.4. Comparaison avec l'application créé en Travaux Pratique : Sketcher

Nous sommes conscient qu'à première vu certaines fonctionnalités peuvent sembler être une copie simple de celle développée dans Sketcher, cependant il n'en n'est rien. En effet pour respecter les règles de convivialité nous avons dû repenser la plupart des concepts de Sketcher. Nous verrons plus en détails ces différences qui concernent l'aspect technique du programme, dans la partie Programmation.

# II. Analyse

Nous allons traiter dans cette partie la manière dont nous avons choisi de modéliser et d'implémenter les fonctionnalités citées dans la partie I. Pour réaliser cette analyse nous avons utilisé diverses sources, avec tout d'abord nos connaissances en Programmation Orientée Objet<sup>1</sup> et plus récemment, grâce aux connaissances acquises par le module "Modélisation".

## 1. Modélisation

Afin de représenter aux mieux les éléments textes, images et ressources disponibles nous avons définis différentes classes. Nous étudierons dans un premier temps les classes *CImg*, *CTexte* et *CEntite* qui représentent les éléments manipulables dans la vue, puis on verra comment les différentes ressources d'images (de fond et les imagettes) sont gérées grâce aux classes *CRessources* et *CLesRessources*. Enfin nous expliquerons comment les informations nécessaires aux fonctionnement de l'application sont gérées par l'architecture Document / Vue.

### 1.1. Gestion des objets

#### 1.1.1. La classe *CEntite*

Pour pouvoir manipuler les divers éléments dans l'application et les regrouper nous avons définie une classe mère *CEntite*. Cette classe possède deux classes filles : *CImg* et *CTexte*. La classe *CEntite* dispose de deux attributs qui sont un *rectangle englobant* (utile pour la sélection d'un élément) de type *CRect* et un attribut *type* de type *WORD* qui spécifie si la classe héritière est un texte ou une image (utile pour l'affichage des informations sur l'objet couramment sélectionné ou encore pour l'affichage du menu contextuel adapté lors d'un clique bouton droit sur l'élément).

---

<sup>1</sup> Nos connaissances de la programmation Orienté Objet reposent essentiellement sur les deux projets en Java que nous avons réalisés lors de notre première année avec un jeu vidéo, et en deuxième année avec la création d'un moteur de recherche sur internet.

Afin de récupérer ses deux attributs, CEntite définit deux méthodes (les *Getters*). Elle possède également les méthodes virtuelles de déplacement, de sérialisation, et de dessin qui sont implémentées dans les classes filles.

### 1.1.2 La classe CTexte

Pour représenter un texte, la classe CTexte possède quand à elle les attributs suivants :

- Une chaîne de caractère (*CString*) qui contient le texte saisi par l'utilisateur.
- Une structure décrivant la couleur (*COLORREF*).
- Une structure décrivant la police (*LOGFONT*).
- Un entier signé (*UINT*) spécifiant la position du texte par rapport au rectangle englobant.

En ce qui concerne les méthodes, les classes CTexte, tout comme la classe CImg, implémentent les méthodes virtuelles de CEntite, à savoir :

- Une méthode de dessin « Draw » pour représenter dans la vue les objets CTexte.
- Une méthode « DrawEnclosingRect » pour dessiner le rectangle englobant lorsque l'élément est sélectionné.
- Une méthode « Move » pour le déplacement de l'élément
- Une méthode « Serialize » pour l'enregistrement

### 1.1.3 La classe CImg

La classe CImg, quant à elle, permet de représenter une image à l'écran. Elle possède trois attributs :

- Un objet *CPicture*, qui permet l'importation d'image selon les formats BMP, JPG et GIF dans une vue. Il s'agit d'une classe libre récupérée sur Internet.
- Un *CString* contenant le chemin du fichier à représenter.
- Un *CPoint* qui nous sert à connaître la position de l'image.

Pour ce qui est des méthodes, la classe CImg implémente les mêmes que celles définies dans CTexte. Son constructeur prend en paramètre un *CPoint* pour la position et un *CString* pour le chemin du fichier.

Voici la représentation de la hiérarchie de ces classes.

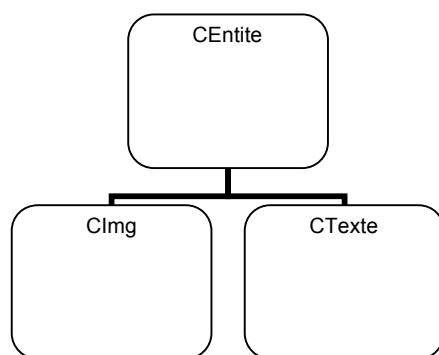


Schéma 1 : Hiérarchie des classes CEntite, CImg et CTexte

## 1.2. Gestion des Ressources images

### 1.2.1. La classe CImageRessource

Que ce soit sous forme d'imagettes ou d'images de fond. Créo2K6 possède un grand nombre de ressources images (150 images). Afin de pouvoir les classer par catégories puis par thèmes et permettre leur sélection par des boîtes de dialogues nous avons définie une classe CImageRessources. Cette classe décrit une image selon trois attributs de type *CString* :

- Un *Domaine*, qui indique à quelles catégories de format de cartes l'image appartient. Pour une image de fond par exemple cet attribut peut prendre les valeurs « vœux », « visite », « postal » ou « invitation ».
- Un *Sous-domaine*, qui indique à quels thèmes l'image appartient. Pour une imagette cet attribut peut prendre les valeurs « Branché » « Sobre » « Symboles » etc.
- Un nom de fichier, qui indique le nom exact du fichier à insérer.

Nous pouvons remarquer que cette classe aurait aussi bien pu s'appeler d'une manière plus générale CRessources, puisqu'elle peut être utilisée par n'importe quelles ressources classées selon un maximum de trois paramètres.

Grâce à cette classe, lorsqu'on désire ajouter l'image sélectionnée, il nous suffit d'importer l'image (grâce à la classe CPicture) depuis le chemin «/ressources/Domaine/Sous-domaine/fichier».

### 1.2.2. La classe CLesImagesRessources

Pour gérer un ensemble de ressources, nous avons définie une classe CLesImagesRessources qui possède une liste d'objet CImageRessource. Cette classe définit les deux méthodes suivantes :

- *listeSousDomaine*, qui prend en paramètre un nom de domaine, et un pointeur sur « ListBox », ce qui lui permet de lister, dans la liste box, toutes les images appartenant au domaine passé en paramètre.
- *listeFichier*, qui prend cette fois en paramètre un Sous domaine et un pointeur sur « ListBox ». Elle permet de lister, dans une seconde liste box, les fichiers appartenant à un sous domaine précis.

Nous pouvons ici faire la même remarque que sur la classe CImageRessource en ce qui concerne sa réutilisabilité pour la gestion de ressources autre que des images.

## 1.3. L'architecture générale

Notre logiciel repose sur une interface SDI, avec une architecture Document / Vue. Nous avons répartis les informations nécessaires à la réalisation de cartes et aux fonctionnements de l'application de la manière suivante : toutes les informations décrivant la carte en cours de réalisation ont été intégrées à la classe CCrea2K6Doc alors que toutes les informations utiles pour afficher les éléments de la vue ont été stockées dans la classe CCrea2K6View. Voyons ensemble de quels attributs il s'agit exactement.

### 1.3.1 La classe CCrea2K6Doc

Le but de cette classe est la gestion des informations de la carte en cours de réalisation elle possède donc les attributs suivants :

- Une *CSize* donnant le format de la carte.
- Une structure de type *COLORREF* contenant la couleur de fond choisi.
- Un objet de type *CImg* contenant l'image de fond sélectionnée.
- Une liste de pointeur sur *CEntite* pour afficher l'ensemble des entités ajoutées à la carte.
- Un *CPoint*, pour retrouver l'origine de la carte. Cette origine changeant en effet à chaque fois que l'on sélectionne un format de carte différent.
- Un *CString*, pour connaître l'emplacement du répertoire de l'application.

La plupart des événements de Crée2K6 se traduisant directement par la modification de la vue, très peu d'événements sont gérés dans cette classe puisqu'ils demandent systématiquement un rafraîchissement de la vue. Nous avons donc préférés les gérer dans la classe *CCrea2K6View*, c'est pourquoi cette classe ne définit que les méthodes relatives à la gestion de la liste d'entité (ajout, suppression, mise en arrière plan) et la méthode *Serialize* pour l'enregistrement de la carte.

### 1.3.2 la classe CCrea2K6View

Les informations utiles à la représentation des éléments dans la vue, sont gérées dans la classe *CCrea2K6View*. Ces attributs sont donc :

- Un *CPoint*, contenant la position du curseur de la souris.
- Un booléen, pour savoir si nous sommes en mode déplacement ou non.
- Un pointeur sur *CEntite* pour connaître l'élément sélectionné.

Hormis les méthodes gérant les événements que nous allons voir en détail dans la partie suivante, la classe *CCréa2K6View* possède les méthodes suivantes :

- La méthode *MoveEntite* qui gère le déplacement d'une entité. Celle-ci n'étant pas basée sur le même principe que la méthode de déplacement de Sketcher, nous l'étudierons donc plus en détail dans la partie Programmation.
- La méthode *SelectEntite* qui récupère l'entité dont le rectangle englobant est sous le curseur de la souris.
- Les méthodes de gestion de l'impression, *OnBeginPrinting*, *OnPreparePrinting*, *OnPrint* et *OnEndPrinting*.

Voyons à présent les événements gérés par *CCrea2K6View*.

## 2. Messages gérés et enchaînement des traitements

La classe *CCrea2K6* gère un grand nombre d'événements. C'est pourquoi nous allons les regrouper selon les catégories : Gestion des événements souris, Gestion des événements clavier, Gestion des événements menus.

## *2.1. Gestion des évènements souris*

Afin de manipuler les différents éléments de l'application de la manière la plus simple, nous écoutons les évènements souris suivants :

- Le clique bouton gauche, pour la sélection d'un élément.
- Le clique bouton droit, pour le menu contextuel selon l'objet sélectionné
- Le double clique gauche, pour faire apparaître l'édition de texte lorsque l'objet sélectionné est un texte.

## *2.2. Gestion des évènements clavier*

Pour améliorer l'efficacité et la rapidité de la conception d'une carte nous écoutons le clavier avec la gestion :

- Des flèches multidirectionnelles, qui déplace d'un pixel l'objet sélectionné dans la bonne direction, afin d'améliorer le positionnement des objets.
- La barre d'espace, qui met en arrière plan l'objet sélectionné.

## *2.3. Gestion des évènements de menu*

L'application propose différentes options de menu et icônes pour réaliser l'ensemble des actions possibles, avec :

- Les options qui n'appel pas de boites de dialogues, comme l'option Format qui change directement (dans CCrea2K6Doc) le format de la carte et actualise la vue, ou encore l'option Insertion > Texte qui ajoute un texte dans la vue.
- Les options faisant appel aux boites de dialogues, comme l'édition de textes ou l'ajout de textes prédéfinis, d'images de fond et d'imagettes. Notés que pour ces deux derniers dialogues il a été nécessaire de passer en paramètre un pointeur sur la vue afin que le boutons « Appliquer » affiche dans la vue l'image sélectionné depuis la boite de dialogue.

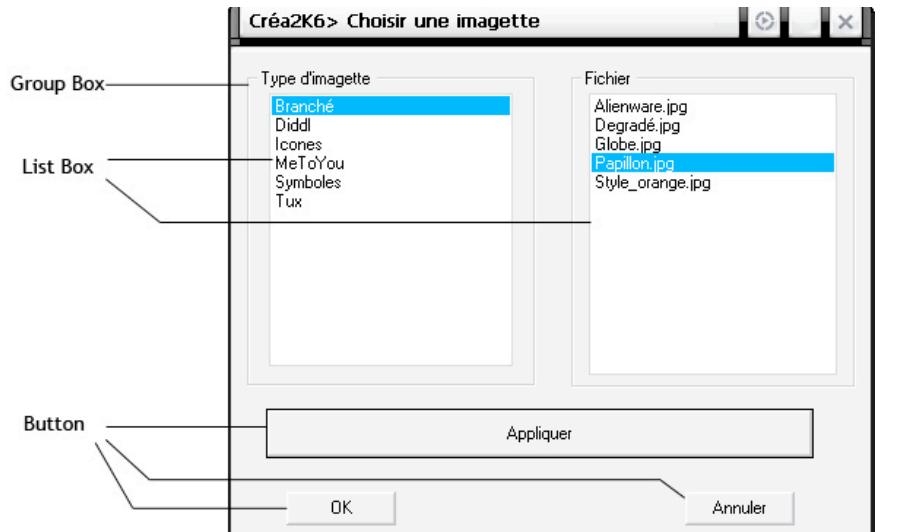
Voyons maintenant plus en détails les différentes boites de dialogues.

## **3. Descriptions des ressources**

Pour ajouter des ressources, ou éditer un texte nous avons crées différentes boites de dialogue que nous vous proposons d'étudié ici.

### *3.1. Boite permettant l'insertion de ressources images*

Nous avons créé deux boites de dialogue similaire pour l'ajout d'image de fond ou d'imagette. Elles sont composées essentiellement de deux « List Box » qui permettent la sélection d'image.

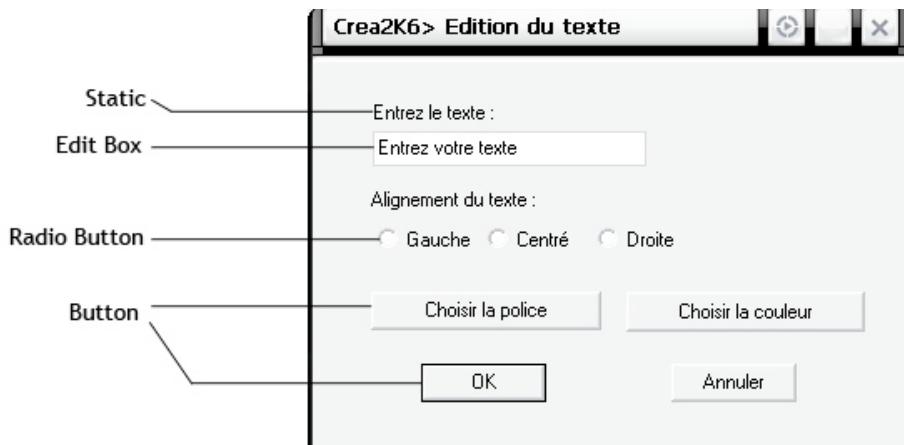


*Boîte de dialogue permettant la sélection d'imagettes*

Pour gérer cette boîte nous avons créé la classe *CImagetteDlg*, qui possède une variable *CString (m\_theme)* qui correspond au type d'imagette sélectionné (sur le screenshot ce thème est « Branché »), une variable *CString (m\_fichier)* contient quant à elle le fichier sélectionné, enfin cette classe construite dans *CCrea2K6View* avec en paramètre *this* possède un pointeur sur la vue pour permettre l'utilisation du bouton appliquer.

### 3.2 Boîte de dialogue permettant l'édition de texte

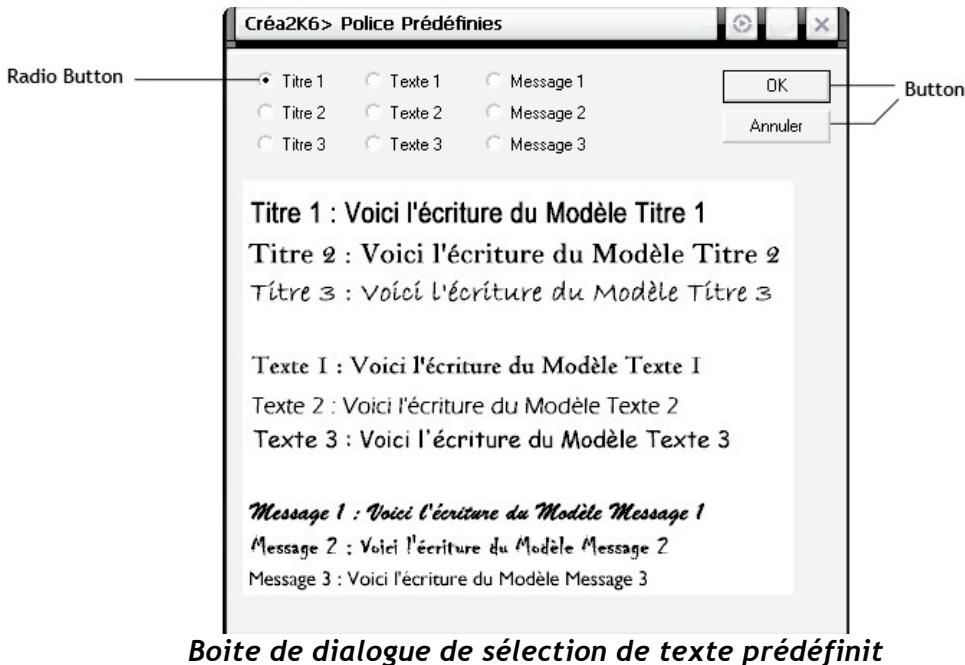
Lorsque l'utilisateur double clique sur un texte, le logiciel lance la boîte de dialogue permettant de l'éditer. C'est la classe héritière de *CDialog*, *CTexteDlg* qui gère cette boîte de dialogue, et définit les variables de mise en forme d'un objet *CTexte* à savoir le contenu, la police (Logfont), l'alignement par rapport au rectangle englobant (UINT) et la couleur.



*Boîte de dialogue d'édition de texte*

### 3.3. Boîte de dialogue permettant l'insertion de texte prédefinie

Crea2K6 offre la possibilité d'ajouter des textes pré formatés, pour cela nous avons conçus la boîte de dialogue suivante :



*Boîte de dialogue de sélection de texte prédéfini*

C'est la classe `CFontPredefDlg` qui gère ce dialogue définissant un entier nommé *police* lors d'un clique sur un bouton radio. La fonction de `CCrea2K6View` récupérant la valeur de cette police crée alors la structure `LOGFONT` correspondante au modèle d'écriture sélectionné, et l'insert dans la vue.

### III. Programmation

Ayant abordé la structure du programme, les classes spécifiques et la description des gestionnaires de messages dans la partie Analyse, la partie programmation va s'attarder sur les différents concepts intéressants mis en œuvre afin d'implémenter les fonctionnalités désirées de l'application.

#### 1. Méthode de dessin de la classe de base `CTexte`

Héritière de `CEntite`, `CTexte` est une classe de base de l'application. Elle permet de stocker toutes les informations utiles de mise en forme et de position d'un texte. Cette classe définit une méthode *Draw* qui lui permet d'afficher dans la vue : le texte voulu, dans le format voulu et à la position désirée. Les étapes de cet algorithme sont les suivantes :

- Création d'un objet `Font` pour contenir la police.
- Initialisation de cet objet `Font` avec l'attribut de `CTexte` contenant la structure `LOGFONT`.
- Sélection de la police dans le contexte de périphérique donné en paramètre de la fonction.
- Sélection de la couleur avec l'attribut de `CTexte` contenant une structure `COLORREF` dans le contexte de périphérique.
- Grâce à la méthode *GetTextExtent* du contexte de périphérique on récupère la taille du rectangle qui va englober notre texte.
- On met à jour la taille du rectangle englobant du texte avec la taille que va faire le nouveau texte.
- On définit le fond du texte sur Transparent.

- Affichage du texte dans le rectangle englobant et aligné selon le *nFormat* choisi, grâce à la méthode *DrawText* du contexte de périphérique.
- Restitution des anciennes valeurs du contexte de périphérique.

Voyons maintenant son algorithme en C++ :

```
void CTexte::Draw(CDC *pDC) {
    // Cr ation de la font
    CFont font;

    // On initialise la font avec notre LOGFONT perso
    font.CreateFontIndirect(&m_lf);

    // On selectionne la font dans le contexte de p eriph rique
    CFont * pOldFont = pDC->SelectObject(&font);

    // On d efinie la couleur du texte
    COLORREF OldColor = pDC->SetTextColor(m_couleur);

    // On r ecupere la taille que va faire le texte avec le nouveau contenu
    CSize size=pDC->GetTextExtent(m_contenu);

    // Mise a jour de la taille du rectangle englobant
    CPoint& pointHautGauche = m_EnclosingRect.TopLeft();
    CPoint& pointBasDroit = pointHautGauche+size;

    m_EnclosingRect = CRect(pointHautGauche,pointBasDroit);

    // Fond du texte transparent
    pDC->SetBkMode(TRANSPARENT);
    // Affichage du texte dans le rectangle englobant et selon le nFormat d efini
    pDC->DrawText(m_contenu,&m_EnclosingRect,m_nFormat);

    // Restitution de l'ancienne police dans le CDC
    pDC->SelectObject(pOldFont);
    pDC->SelectObject(&OldColor);
}
```

## 2. La gestion du Format de la carte et affichage des  l ments dans la vue

Afin de repr senter les cartes dans la vue, nous avons d  g rer le format de la carte. Pour cela la classe *CCrea2K6Doc* poss de un objet *CSize* qui d termine la taille de la carte et un *CPoint* (*m\_Origine*) qui d fini le coin haut gauche du format de la carte. Cette origine est d termin e   chaque fois que l'utilisateur change le format de la carte. Pour la calculer nous r cup rons la taille courante de la vue, et la taille du format de la carte.

Son algorithme en C++ est le suivant :

```
// r ecup ration de la taille de la vue
pView->GetClientRect(m_RectVue);

// Calcul de la nouvelle position de l'origine
m_Origine=CPoint( (m_RectVue.Width()/2) - (m_Format.cx/2), (m_RectVue.Height()/2)
- (m_Format.cy/2) );
```

Pour ce qui est de l'affichage de la carte, c'est la m thode *OnDraw* de la classe *CCrea2K6View* qui s'en charge. Nous allons voir ici quelles sont les principales  tapes de cette m thode :

- R cup ration d'un pointeur sur Document pour conna tre les diverses informations sur la carte

- Limitation de la zone de dessin au format de la carte (grâce à la méthode *IntersectClipRect* du contexte de périphérique)
- Dessin du rectangle représentant le format de la carte
- Dessin de l'image de fond sélectionnée par l'utilisateur
- Dessin des entités présentes dans la liste d'entités stockées dans CCrea2K6Doc
- Affichage du cadre d'information sur l'élément sélectionné

Cet algorithme, ce traduit de la façon suivante en C++ :

```
/*** Méthode d'affichage de la carte ***/
void CCrea2K6View::OnDraw(CDC* pDC)
{
// Récupération d'un pointeur sur doc
    CCrea2K6Doc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    /** Définition du nouveau format **/

    // Limitation de la zone de dessin au format de la carte
    pDC->IntersectClipRect(CRect(pDoc->m_Origine,pDoc->m_Origine+pDoc->m_Format));

    // Dessin du rectangle délimitant le format
    CPen aPen;
    aPen.CreatePen(PS_SOLID,1,RGB(160,160,160));
    CPen* pOldPen = pDC->SelectObject(&aPen);
    CBrush brush(pDoc->m_couleurFond);
    CBrush* pOldBrush = pDC->SelectObject(&brush);
    pDC->Rectangle(CRect(pDoc->m_Origine,pDoc->m_Origine+pDoc->m_Format));
    pDC->SelectObject(pOldPen);
    pDC->SelectObject(pOldBrush);

    // Dessin de l'image de fond
    pDoc->m_imgFond->Draw(pDC);

    // Dessin des entités de la liste à afficher
    POSITION aPos=pDoc->GetListHeadPosition();
    CEntite* pEntite=NULL;

    // Boucle sur tous les éléments de la liste
    while(aPos) {
        pEntite=pDoc->GetNext(aPos);

        // Vérifie si le rectangle englobant de l'entité à dessiner est
        // entièrement dans la vue
        if(pDC->RectVisible(pEntite->GetBoundRect()))
            pEntite->Draw(pDC,m_pSelected);

    }
    // Affichage du cadre d'information sur l'élément sélectionné
    if(m_pSelected)
        AfficheInfo(pDC);
}
```

### 3. La méthode *listeSousDomaine* de la classe *CLesImagesRessources*

Comme nous l'avons déjà évoqué dans la partie modélisation, nous avons défini deux classes pour la gestion des ressources. La classe *CLesImagesRessources* gère une liste de *CImageRessource* et permet grâce aux fonctions *listeSousDomaine* et *listeFichier* d'afficher les images classées par domaines et sous domaines. La méthode *listeSousDomaine*, affiche donc dans une

« list box » les sous domaines classés selon le domaine donné en paramètre de la méthode. Voyons son algorithme :

- On supprime les entrées déjà présentes dans la « list box ».
- On parcourt la liste d'objet CImageRessources.
- Si le domaine de l'objet CImageRessource courant correspond au domaine demandé en paramètre, et si il s'agit d'un sous Domaine qui n'a pas déjà été ajouté dans la liste. On ajoute ce nouveau Sous Domaine.

L'algorithme correspondant en C++ :

```
void CLesImagesRessources::listeSousDomaine(CListBox*pBox, CString domaine)
{
    // Suppression de ce qu'il y avait dans la list box
    pBox->ResetContent();

    // Déclaration de la liste de sous domaine déjà ajouté à la liste
    CStringList temp;

    // Déclaration de la variable POSITION pour la parcourt de la liste
    POSITION aPos ;
    aPos = GetHeadPosition();

    // Déclaration du pointeur sur l'objet CImageRessource courant
    CImageRessource* ir ;

    // Parcourt de la liste de CImageRessouce
    while(aPos) {
        ir = GetNext(aPos);

        // Déclaration et initialisation du booléen qui donne le résultat de // la
        // recherche du sous domaine
        int dejaVue=à ;

        // Si le domaine de l'objet courant est le même que celui demandé
        if (ir->m_domaine==domaine) {

            // On recherche si ce n'est pas un sous domaine déjà
            // ajouté
            POSITION aPos2;
            aPos2 = temp.GetHeadPosition();
            CString s;
            while(aPos2){
                s=temp.GetNext(aPos2);
                if(s==ir->m_sousDomaine)
                    dejaVue=1;
            }
            // Si ce n'est pas un sous domaine déjà ajouté
            if(!dejaVue) {
                // On l'ajoute à la list box et à la liste de sous
                // domaine déjà ajouté
                pBox->AddString(ir->m_sousDomaine);
                temp.AddHead(ir->m_sousDomaine);
            }
        }
    }
}
```

#### 4. L'impression

Créa2K6 définit également les méthodes d'impressions de cartes. Pour cela nous avons implémenté les méthodes OnPreparePrinting, OnPrepareDC, OnPrint et OnEndPrinting, ainsi que la classe CPrintdata stockant les informations utiles à l'impression. Les diverses informations sur l'origine de la carte ou bien encore sa taille étant déjà contenu dans CCrea2K6Doc, nous n'avons pas eu ce genre de calcul

à implémenter. L'impression d'une carte s'effectuant sur une seule page. Nous n'avons également pas eu besoin de calculer le nombre de page total. Voyons ensemble le détail de l'algorithme de la méthode OnPrint.

- On initialise l'objet CPrintData avec les informations utilisateur de la classe contenant les informations sur l'impression
- On récupère un pointeur sur document
- On affiche le titre du document en haut de la page
- On remplace l'origine de la fenêtre par l'origine de la carte
- Découpe du rectangle à imprimer, correspondant au format de la carte
- On affiche le document
- Enfin on restaure les anciennes valeurs du contexte de périphérique

```
void CCrea2K6View::OnPrint(CDC *pDC, CPrintInfo *pInfo)
{
    // On initialise l'objet CPrintData
    CPrintData* pPrintData = static_cast<CPrintData*>(pInfo->m_lpUserData);

    // Récupération d'un pointeur sur document
    CCrea2K6Doc * pDoc = GetDocument();

    // Affichage du titre du document
    pDC->SetTextAlign(TA_CENTER); // Centre le texte
    pDC->TextOut(pInfo->m_rectDraw.right/2, 20, pPrintData->m_DocTitle);

    // Remplace l'origine de la fenêtre par celle de la carte
    CPoint OldOrg = pDC->SetWindowOrg(pDoc->m_Origine.x, pDoc->m_Origine.y);

    // Définit un rectangle de découpage de la taille de la zone imprimée
    CRect rect (pDoc->m_Origine, pDoc->m_Origine+pDoc->m_Format);
    pDC->IntersectClipRect(rect);

    // On désélectionne l'objet sélectionné (sinon on voit son rectangle
    // englobant lors de l'impression
    m_pSelected=NULL;
    // Affiche le document

    OnDraw(pDC);

    // Restauration des anciennes valeurs du contexte de périphérique
    pDC->SelectClipRgn(NULL); // Supprime le rect. de découpage

    pDC->SetWindowOrg(OldOrg);
}
```

## 5. Le déplacement dans CCrea2K6

En ce qui concerne la gestion du déplacement des éléments nous n'avons pas réutilisé la technique de l'application Sketcher. En effet ce dernier utilise le mode R2\_NOTXORPEN pour effacer l'objet en cours de déplacement, alors que ce n'est pas possible avec les images. C'est pourquoi, pour effacer un élément nous réaffichons à chaque fois toute la liste d'éléments<sup>2</sup>. Ainsi l'algorithme en français de la méthode MoveEntite définie dans la classe CCrea2K6View est le suivant :

- On récupère un pointeur sur Document pour avoir accès à la liste d'entités.

---

<sup>2</sup> Le réaffichage de la liste d'entité à chaque déplacement d'un objet provoque cependant des clignotements intempestifs de la vue. La solution à ce problème réside dans la technique dite du « Double Buffering », qui consiste à dupliquer l'objet contexte de périphérique. Nous n'avons cependant pas réussi à implémenter cette technique dans notre application.

- On calcul la distance de déplacement à partir de la position actuelle de la souris et de sa position précédente.
- On supprime l'entité de la liste et on réaffiche la liste.
- On déplace l'entité grâce à sa méthode Move.
- On ajoute l'entité à la liste, et on la redessine par à sa méthode Draw.

Algorithme en langage C++ :

```
/** Méthode de déplacement d'une entité ***/
void CCrea2K6View::MoveEntite(CClientDC &aDC, const CPoint &point)
{
    // Récupération d'un pointeur sur doc
    CCrea2K6Doc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    // Calcul de la distance de déplacement
    CSize distance;
    distance= point - m_CursorPoint;
    m_CursorPoint=point;

    // S'il y a bien un objet sélectionné
    if (m_pSelected) {
        // On récupère la position de cet élément dans la liste
        POSITION aPos = pDoc->m_EntiteList.Find(m_pSelected);
        // On vérifie s'il a bien été trouvé
        if(aPos) {
            // On supprime l'élément
            pDoc->m_EntiteList.RemoveAt(aPos);
            // On rafraichi la vue
            Invalidate();
        }
        // Appel de la méthode de déplacement de l'entité
        m_pSelected->Move(distance);           // déplacement

        // Ajout de l'entité déplace dans la liste
        pDoc->AddEntite(m_pSelected);

        // Appel de la méthode de dessin de l'entité
        m_pSelected->Draw(&aDC,m_pSelected);
    }
}
```

## IV. Les problèmes rencontrés

La programmation avec les MFC sous Visual C++ est reconnu comme étant assez singulière par de nombreux programmeurs. Pour notre part nous devons avouer que le temps d'adaptation fût effectivement plus long que pour d'autres langages de programmation comme Java. L'élaboration de notre logiciel ne c'est pas déroulé sans problèmes. Le premier d'entre eux était pour nous la représentation du format de la carte. Ensuite nous avons dû trouver une méthode de déplacement des objets. Le technique implémenté entraînant de forts clignotements lorsque de nombreux éléments sont présents à l'écran nous avons essayé de résoudre le problème mais en vain.

Pour la gestion du bouton appliquer des boites de dialogues d'insertion d'images, il nous fallait un pointeur sur vue. L'appel des dialogues avec le paramètre *this* à entraîner beaucoup de problème dans les classes gérant les dialogues à cause de problèmes d'inclusions. La solution trouvée consiste à déplacer certaines inclusions générées par défaut dans CCrea2K6View.cpp dans son entête CCrea2K6View.h.

Pour la sérialisation, nous n'avons pas réussi à sérialiser le pointeur *IPicture*\* de la classe CPicture. La solution mise en place pour la sauvegarde des images a consisté à sauvegarder la liste de chemins (*CString*) respectifs des images présentes et de leur position.

Enfin, à l'heure où nous écrivons ses lignes, l'impression d'une carte s'effectue dans un format très petit.

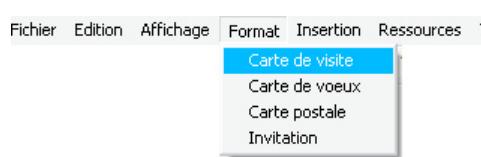
## V. Conclusion

La programmation d'un logiciel sous Visual C++ pour un débutant n'est pas chose aisée. Pourtant après un travail de longue haleine nous avons pu finir le projet dans les temps, même si nous en avons souvent douté lors de sa conception. Nous avons, en effet, pu mettre la quasi totalité des fonctionnalités que nous souhaitions implémenter. Pour cela nous nous sommes servis de toutes les infos dont on pouvait disposés, que ce soit les cours magistraux, ou l'aide des MFC ou encore les divers renseignements obtenus sur les forums, et ressources libres récupérées sur Internet. La conception de Créo2K6 nous a beaucoup fait progressé en nous faisant revoir, approfondir et compléter les différentes notions du cours, ce qui nous permet maintenant d'affronter les prochains examens avec un peu plus de sérénité.

## Annexe 1 : Manuel utilisateur

Nous allons maintenant voir ensemble comment utiliser les fonctionnalités de l'application, ainsi que les étapes de la conception d'une carte.

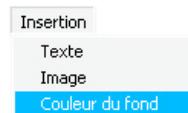
### 1<sup>ère</sup> Etape : Le choix du format



Lorsque l'application démarre, le choix du format par défaut est Carte de Vœux. Il peut être modifié en sélectionnant l'option Format puis en sélectionnant le format désiré (Carte de visite, Carte de vœux, Carte d'invitation ou Carte Postal).

### 2<sup>ème</sup> Etape : Le choix d'une couleur ou d'une image de fond

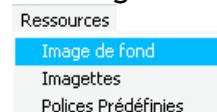
Pour le choix d'une couleur de fond, il suffit de sélectionner l'option « Insertion », puis de cliquez sur « Couleur de fond ».



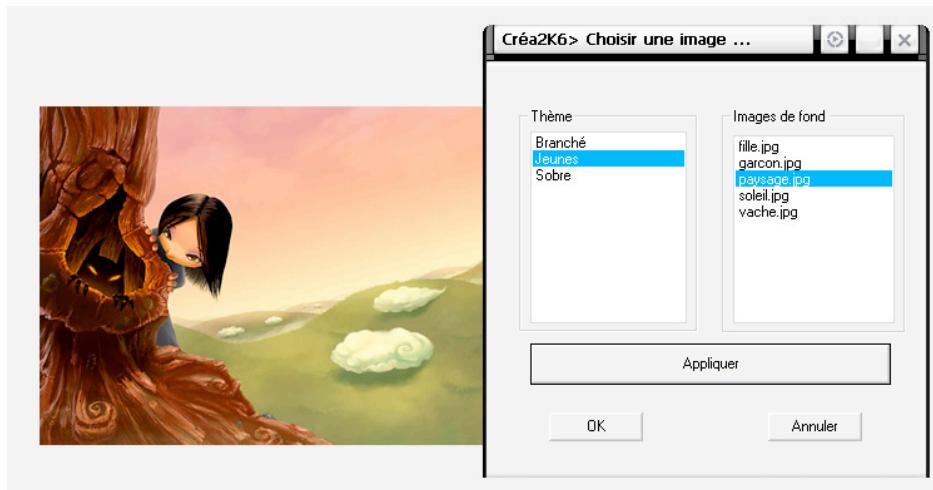
Une boite de dialogue s'ouvre alors et propose le choix de la couleur.



Pour le choix d'une image de fond, il suffit de sélectionner l'option « Ressources », puis de cliquez sur « Image de fond ».



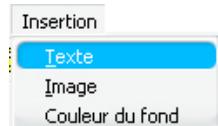
Une boite de dialogue apparaît alors et permet le choix entre les différents thèmes du format de la carte. Ici le format de la carte était Carte de visite, d'où les thèmes « Branché », « Sobre » et « Jeunes ». Il suffit donc de sélectionner son thème et le fichier que l'on veut insérer, puis de cliquer sur « Appliquer » pour voir à quoi la carte ressemble avec le nouveau fond, si cela vous convient : validez en cliquant sur « Ok ».



### 3<sup>ième</sup> Etape : L'ajout de textes

Insérons maintenant un texte. Il existe deux types d'insertion de textes, à savoir l'insertion de texte simple, ou l'insertion de texte prédéfini.

Pour insérer un texte simple faites Insertion, puis cliquez sur Texte.

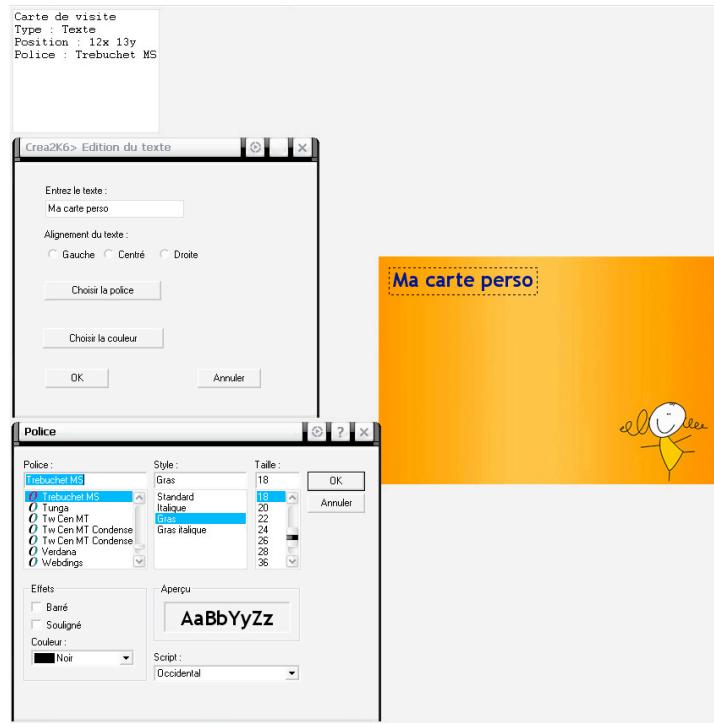


Un texte est affiché par défaut au centre de la carte. Pour le déplacer il vous suffit de cliquez dessus puis de faire un classique glisser / déposer à l'endroit où vous désirez le mettre.

Pour insérer un texte prédéfini, sélectionné l'option « Ressources », puis cliquez sur « Polices prédéfinies ». La boîte de dialogue de sélection de polices prédéfinies s'affiche alors. Sélectionnez la police que vous désirez et validez en cliquant sur « Ok ».

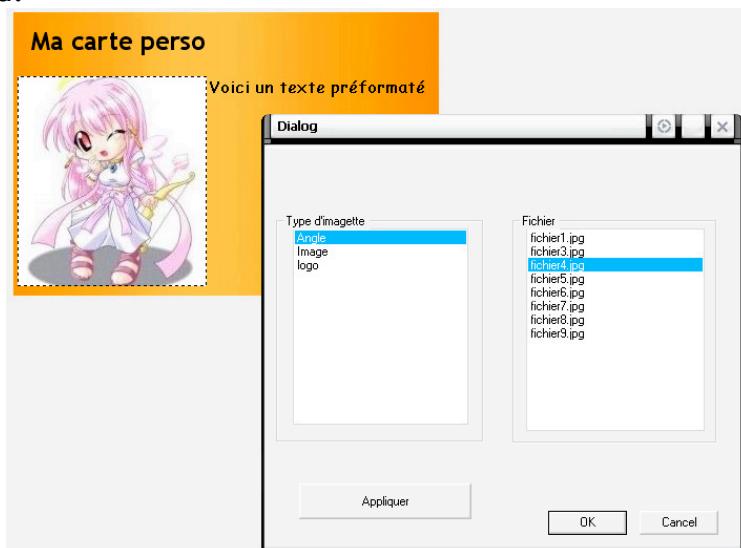


Maintenant que vous avez ajouté un texte, nous allons l'éditer. Pour cela il suffit de double cliquer sur le texte, ou alors de faire un clique bouton droit, puis de sélectionner « Editer le texte ». Une boîte de dialogue d'édition de texte apparaît et vous permet de configurer la mise en forme de votre texte. Cette boîte comporte les boutons « choix de la police » et « choix de la couleur » qui lancent de nouvelles boîtes de dialogues permettant de définir ce que vous souhaitez.



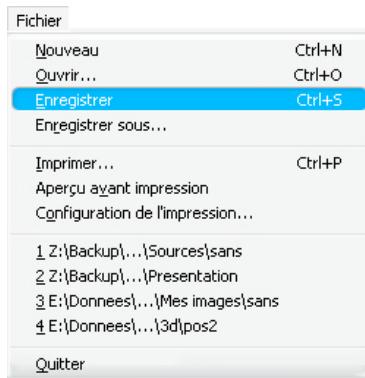
#### 4<sup>ième</sup> Etape : L'ajout d'imagettes

Pour ajouter une imagette sélectionnez l'option « Ressources » puis cliquez sur « Imagettes ». La boîte de dialogue de sélection d'imagette s'ouvre alors. Il vous suffit alors de sélectionner l'image que vous désirez de la même manière que pour l'image de fond.



## **5<sup>ième</sup> Etape : Enregistrer une carte**

Pour enregistrer une carte il suffit de sélectionner l'option de menu « Fichier » puis de cliquer sur « Enregistrer ».



## **6<sup>ième</sup> Etape : Impression d'une carte**

Pour imprimer une carte, sélectionner l'option de menu « Fichier » puis cliquer sur « Imprimer... ». Une boîte de dialogue s'affiche alors vous permettant de configurer divers paramètres, vous pouvez alors valider en cliquant par « Imprimer ».

## Annexe 2 : Etude de l'offre existante

Afin de faire le tour des fonctionnalités que propose ce genre d'application et des différentes interfaces qu'il existe, nous nous proposons de faire ici un rapide tour de l'offre existante. Il existe de très nombreux logiciels et sites Internet qui permettent de créer des cartes en tout genre, nous ne verrons ici que les plus connus. Le but de cette étude est essentiellement de voir ce qui existe actuellement sur le marché, mais également de permettre au lecteur de juger en connaissance de cause les fonctionnalités que nous avons implémentés ainsi que leur intégration dans l'interface, grâce à l'analyse et à la critique.

Nous verrons dans un premier temps les sites Internet, et dans un deuxième temps les différents programmes existant sur le marché.

### 1. Les sites Internet

Les fonctionnalités des sites Internet sont généralement beaucoup moins évoluées que celles des programmes, mais leur simplicité d'accès et leur convivialité grâce à leur interface web les font préférés des programmes spécialisés par le grand public. Aussi, leurs mises à jours régulières leurs permettent d'avoir un nombre toujours grandissant d'images, de modèles et propose des cartes en fonction de la période de l'année (comme des cartes déjà configurées pour souhaiter la nouvelle année par exemple). Pour notre application, l'intérêt d'étudier ces sites dédiés à la création de carte de vœux, réside dans l'analyse de leurs interfaces claires et ergonomiques.

On peut distinguer deux types de sites web qui permettent la création de cartes. Ceux spécialisés dans la création d'e-carte à envoyer par mail, et les imprimeurs. Les premiers, permettent de concevoir des e-mails personnalisés par des images, des textes, des sons et animations selon un thème choisi. Ces e-mails sont créé de façon dynamique (généralement par javascript, ou des langages côté serveur tel que php, jsp) depuis le site web, puis généré en html et/ou en flash puis envoyé au destinataire défini par l'internaute.

Il existe de très nombreux sites web proposant de tels services, voyons ici les principaux.

Yahoo : Même les moteurs de recherche permettent la création d'e-carte. Le service de yahoo est cependant très simpliste : seulement trois modèles de disposition proposés, cinq couleurs pour le texte et des thèmes à la qualité très aléatoire. La personnalisation se fait uniquement depuis un assistant qui ne permet pas d'intervenir directement sur la carte. Une particularité bienvenue cependant, est la planification de la date d'envoi de la carte, ainsi on peut à l'avance (fixé malheureusement à un mois maximum) créer les cartes de vœux, ou d'anniversaires de nos proches.

Lien : <http://fr.send.greetings.yahoo.com/>

L'internaute : Magasine web, L'internaute propose de nombreux services et n'échappe pas à un service d'e-carte. Ce dernier propose de très nombreux thèmes et sous thèmes, de quoi combler tous les goûts, mais la personnalisation de la carte y est très dépouillée : On peut seulement donner le titre, le message, et une signature

sans personnalisation ni de polices, ni de couleur ni de disposition. Remarquer que l'interface en est par conséquent très claire, et simple d'accès.

Lien : <http://www.linternaute.com/cartes/>

J'envoie une carte Rédigez votre message

Personnalisez votre carte : Veuillez remplir le formulaire ci-dessous et cliquez sur le bouton "VISUALISEZ VOTRE CARTE".

À : Nom E-mail

De : Nom E-mail

Nombre de destinataires : 1 [Extraire de mon carnet d'adresses Yahoo!](#)

Message: (4000 caractères max)

Options facultatives

Affichage de la carte :

Couleur / taille / choix de la police :

Fixer la date d'envoi :

Votre titre :

Votre message :

Signature (Nom et ou Formulation) :

couper le son

Envoyer immédiatement  Envoyer plus tard

Envoyez ma carte le : 29  12  2005  à 13h  Heure de Paris

VALIDER

Villeronce : Site à l'accès réservé, il propose cependant un service d'e-carte ouvert à tous. A la première approche Villeronce peut rebouter l'internaute, son interface n'utilisant visiblement pas les CSS, lui donne un coté très « site perso ». Mais cette dernière impression est vite estompé après avoir vu les dizaines de thèmes, et les nombreuses possibilités de personnalisation. En effet la création d'une e-carte passe par de nombreuse étapes, en commençant par le choix de l'image permet les milliers que propose le site et les dizaines d'effets (torsion, neige, bouillard, lumière) applicable dessus, puis l'internaute est amené à choisir le titre, les polices d'écritures, et sa couleur, la couleur de fond, le papier peint, le fond sonore, le nombre de destinataire, le contenu du message (avec smileys) et un modèle (parmis 5 prédéfinis). On regrettera qu'il n'y ai pas la possibilité d'envoi différé des cartes.

Lien : <http://cartes.villeronce.com/carte.htm>

Joliecarte : Site entièrement dédié à la création de carte, l'interface belle et claire et les thèmes très nombreux en font un site agréable à visiter. Quid des fonctionnalités ? Elles sont également très complètes avec les divers effets applicable sur toutes la carte, l'édition d'un faux timbre, et la saisie direct sur la carte du message avec formatage complet du texte. Joliecarte.com est selon nous le site le plus beau, le plus claire, et le plus simple de tous les sites d'e-carte. On regrettera qu'il oblige l'internaute à donner son adresse mail pour pouvoir accéder à la personnalisation d'une carte.

Lien : <http://www.joliecarte.com/>

**Indiquez le nombre de destinataires (facultatif) :** Développer tout | Réduire tout  
Passez directement à la suite si vous voulez envoyer cette carte à une seule personne. Sinon, indiquez le nombre de destinataires et cliquez sur le bouton « Continuer ».

La carte sera envoyée à 1 personne

**Entrez vos informations**  
Donnez les informations nécessaires à l'envoi de votre carte virtuelle. Assurez-vous que les adresses e-mail soient valides.

* Prénom du destinataire : <input type="text"/>	* e-Mail du destinataire : <input type="text"/>
* Votre prénom : <input type="text"/>	* Votre e-mail : <input type="text"/>
<input type="checkbox"/> Néanmoins mon prénom et ma adresse e-mail	

**Choisissez un style de carte (facultatif) :** Développer tout | Réduire tout  
Choisissez la couleur de la carte et le type de papier peint (modèle d'arrière-plan).

Couleur de la carte :



Interface toutes en longueur, voici l'aperçu de quelques fonctionnalités de cartes. [Villeronce.com](http://Villeronce.com). Le deuxième screenshot est la personnalisation d'une carte sur [Linternaute.com](http://Linternaute.com)

Doctissimo : Site sur la santé, doctissimo propose lui aussi un service de création d'e-carte dont l'interface n'a pas trop à envier à celle de Joliecarte.com. De plus il permet également l'envoi des e-cartes en différé, et à plusieurs destinataires. Dommage qu'il propose un nombre très limité d'images.

Lien : <http://cartes.doctissimo.fr/>

Cartepostalevirtuelle : Site entièrement dédié à la création de carte virtuelle. Il propose les fonctionnalités classiques pour la création d'e-carte, mais assez peu évolué : choix entre seulement trois modèles de dispositions, deux couleurs d'écritures, définition du titre, du message, possibilité d'envoi différé. Ce site peut cependant convenir à la plupart des internautes.

Lien : <http://www.cartepostalevirtuelle.com/>

**Rédaction du message**

Votre message

Joyeux Noël

Votre prénom

Votre nom

**Mise en forme**

Police: Times, Arial, Garamond, Verdana

Couleur du texte:

Disposition du texte: Droite, Gauche, Bas

**Paramètres d'envoi**

Votre adresse mail:   
Nombre de destinataires: 1  
Destinataire 1:   
A envoyer: Aujourd'hui  
 Je souhaite recevoir une copie de cette carte  
 Je souhaite recevoir un accusé de réception

**BONNE ANNEE**

Votre titre (30 lettres max.):  Le texte de votre carte apparaîtra comme ceci

Message:   
Format:  couleur de l'écrance  
 couleur du fond  
 couleur de l'image

Signature (20 lettres max.):

Nom du destinataire:   
Email du destinataire:   
Votre nom:   
Votre email:

Date d'envoi ?  
Votre carte sera envoyée immédiatement.

L'interface de personnalisation d'une e-carte de [Joliecarte.com](http://Joliecarte.com)

Le second type de site offrant la possibilité de création de carte sont ceux des imprimeurs. Généralement destiné aux professionnels, ces sites sont souvent très sobre et claire mais assez austère. Leur spécialité étant la création de carte de visite.

Sepieter : De tous les imprimeurs Sepieter offre le plus de service de personnalisation. Ainsi comme les autres sites web il propose de créer des cartes par interface web normale mais également par l'intermédiaire d'un assistant en flash ou l'utilisateur peut agir directement sur la carte qu'il est entrain de créer. Il défini alors son propre modèle, dommage qu'il n'y est que six polices prédéfinis et que la couleur du texte ne soit pas personnalisable.

Lien : <http://www.sepieter.com/>

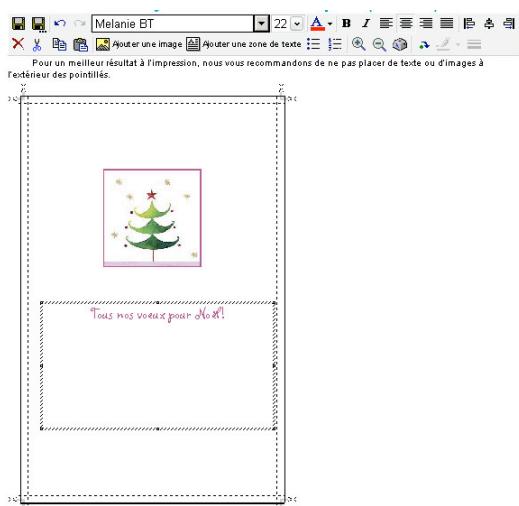
Ooprint : Grand imprimeur professionnel, ooprint.fr propose un très grand nombre de thèmes et de modèles, même si ceux-ci reste très classiques. Dommage que leurs services de personnalisation étaient hors service lors de la rédaction de ce rapport.

Lien : <http://www.ooprint.fr/>

The screenshot shows the Sepieter website's card creation interface. On the left, there's a sidebar with options like 'Enveloppes', 'Pochettes', 'Têtes de lettres', 'Cartes de visite', and 'Cartes de correspondance'. Below this, it says 'Faites votre choix dans chacune des étapes suivantes et visualisez la création de votre produit personnalisé en temps réel.' There are sections for 'Impression' (Recto/Verso), 'Textes' (Ajouter une information, Police de caractère: Garamond, Stone Sans, etc.), 'Visuals' (Taille de la police, Couleur: Noir), and 'Gestion des zones' (recto, modifier, déplacer). A central panel shows a preview of a business card with fields for 'Prénom NOM' (Pierre Dupont) and 'Adresse1' (88 bis rue de la République, 62800 Péronne France). To the right, there are four examples of generated business cards labeled 'VOTRE SOCIETE' with details like 'Pierre Dupont' and contact info. Each card has a 'de détails sur ce modèle' link.

Vistaprint.fr : Très grand site de création de carte, il permet aussi bien de réalisé des cartes à imprimer que des cartes virtuelles. Vistaprint possède un nombre impressionnant d'images et de thèmes et leur module de personnalisation, quoique long à charger, permet une personnalisation très fine de la carte, avec comme chez Sepieter la possibilité d'agir directement sur la carte, pour choisir la disposition exacte que l'on souhaite.

Lien : <http://www.vistaprint.fr/>



*Ici, la personnalisation de l'intérieur d'une carte de voeux*

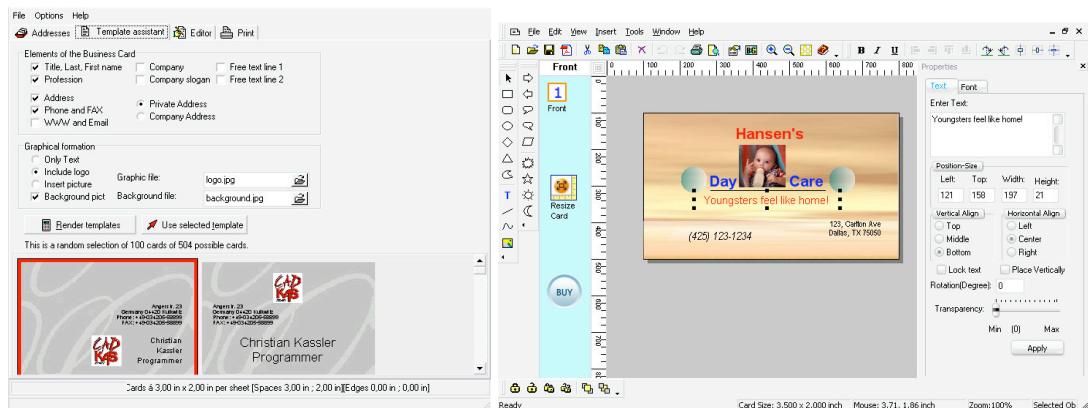
Comme on peut le voir il est difficile de trouver le site web parfait pour la création de cartes. Certains disposent de ressources très conséquente mais offre des fonctionnalités pauvre ou très limités, et inversement pour d'autre. Aussi, certains sites ont des interfaces trop peu conviviales pour le grand public.

## 2. Les programmes

Voyons à présent, les quelques programmes qui proposent la création de cartes. Il faut savoir qu'il s'agit souvent d'importants programmes de dessins qui proposent ce genre de fonction, mais on peut trouver quelques programmes plus modeste (mais payant) qui sont entièrement dédié à la création de cartes (de visite, de vœux ou d'étiquette).

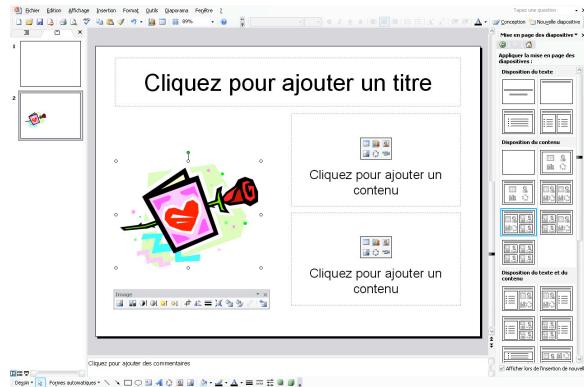
**Business Card Printery 3.5 :** Logiciel payant, il est spécialisé dans la création de cartes de visites. Ce programme propose une interface austère. Pour ce qui est des ressources, Business Card Printery possède simplement une dizaine d'images (visiblement issu des images que Windows possède, qui sont à mettre en mosaïque pour le fond d'écran) à insérer. Le bon point de Business Card Printery réside dans sa configuration de l'impression, avec la possibilité de définir le nombre de cartes à imprimer sur une même page avec visualisation en temps réel de la page à imprimer. Pourtant à sa version 3.5, ce logiciel, payant qui plus est, ne peut pas concurrencer les autres produits que nous verrons ci-après.

**Advanced Business Card Maker :** Plus complet et plus intitif, Advanced Business Card Maker est également spécialisé uniquement dans la création de carte de visite. La création d'une nouvelle carte commence toujours par le lancement de l'assistant qui permet de choisir son modèle parmi un large choix, ainsi qu'une image de fond et le format de la carte. Ensuite, il nous reste alors à personnaliser la carte sous l'éditeur. Ce dernier permet de sélectionner n'importe quel objet présent et affiche ses informations (de taille, de couleur, de position etc.) que l'on peut alors modifier comme il nous plaît. Advanced Business Card Maker est donc beaucoup plus avancé que Business Card Printery, mais étant comme ce dernier payant, il peut difficilement se faire une place face aux logiciels de dessins.

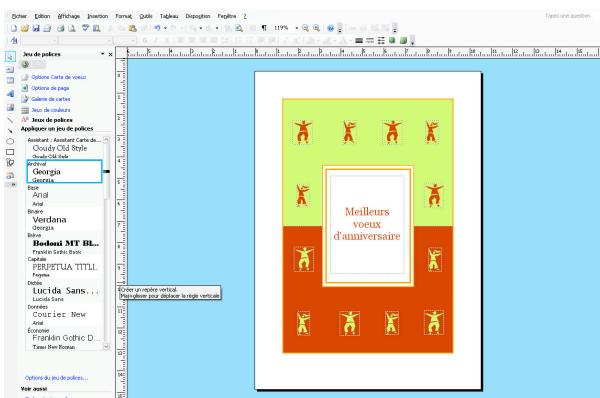


**Microsoft PowerPoint :** Spécialisé dans la création de diaporama, Power point convient également parfaitement à la création de carte en tout genre. Aux niveaux des ressources il dispose de milliers de clipart classés par catégories, et il est possible d'importer directement de nouveaux modèles depuis le site de microsoft. L'éditeur est une référence du genre, il suffit de choisir un modèle de disposition,

puis de cliquer sur les objets créer dans la vue pour ajouter, textes, images, graphiques etc. Les inconvénients que l'ont peut trouver sur PowerPoint est son manque de simplicité pour la création de cartes puisqu'il n'y a pas d'assistant permettant de faire cela.



**Microsoft Publisher :** Outils de composition pour l'impression Publisher est l'un des ténors de la création de cartes. Sur le plan des ressources il compte des centaines d'images et de modèles prédéfinis. Très complet, il possède toutes les fonctionnalités classiques et bien plus, tout en restant accessible pour tout public.



Comme on peut le voir, les programmes les plus connus et réalisé par les plus grand éditeur de logiciels sont bien plus pratique et complet que les petits logiciels dédiée à la création de cartes. Beaucoup d'autres, payant, sont ceux qui offrent les services les plus complets. On aurai pu également cité Microsoft Picture it !, FireWorks, et bien d'autres encore...

### 3. Bilan

Cette étude nous a été très bénéfique lors des divers choix que nous avons dû faire tout au long de la conception de notre programme. Elle nous a permis d'avoir dès le départ un recul suffisant. C'est pourquoi nous avons estimé nécessaire de faire part de cette étude dans le rapport, le lecteur pourra ainsi juger par lui-même de la qualité des choix que nous avons fait.