Ergonomie
notions et mise en œuvre avec Swing

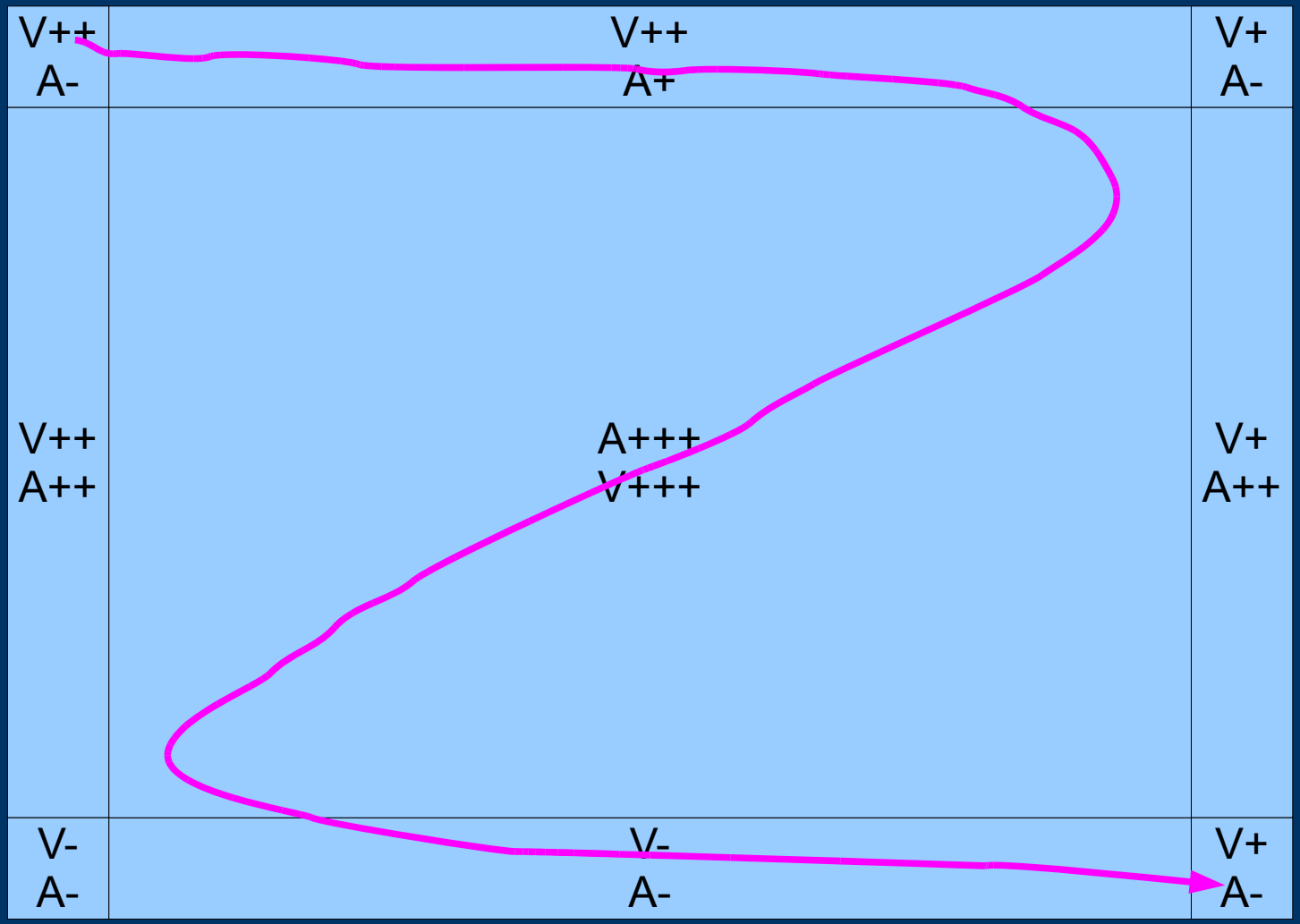
B. Mermet
Licence 3 – Université du Havre
2010

Introduction

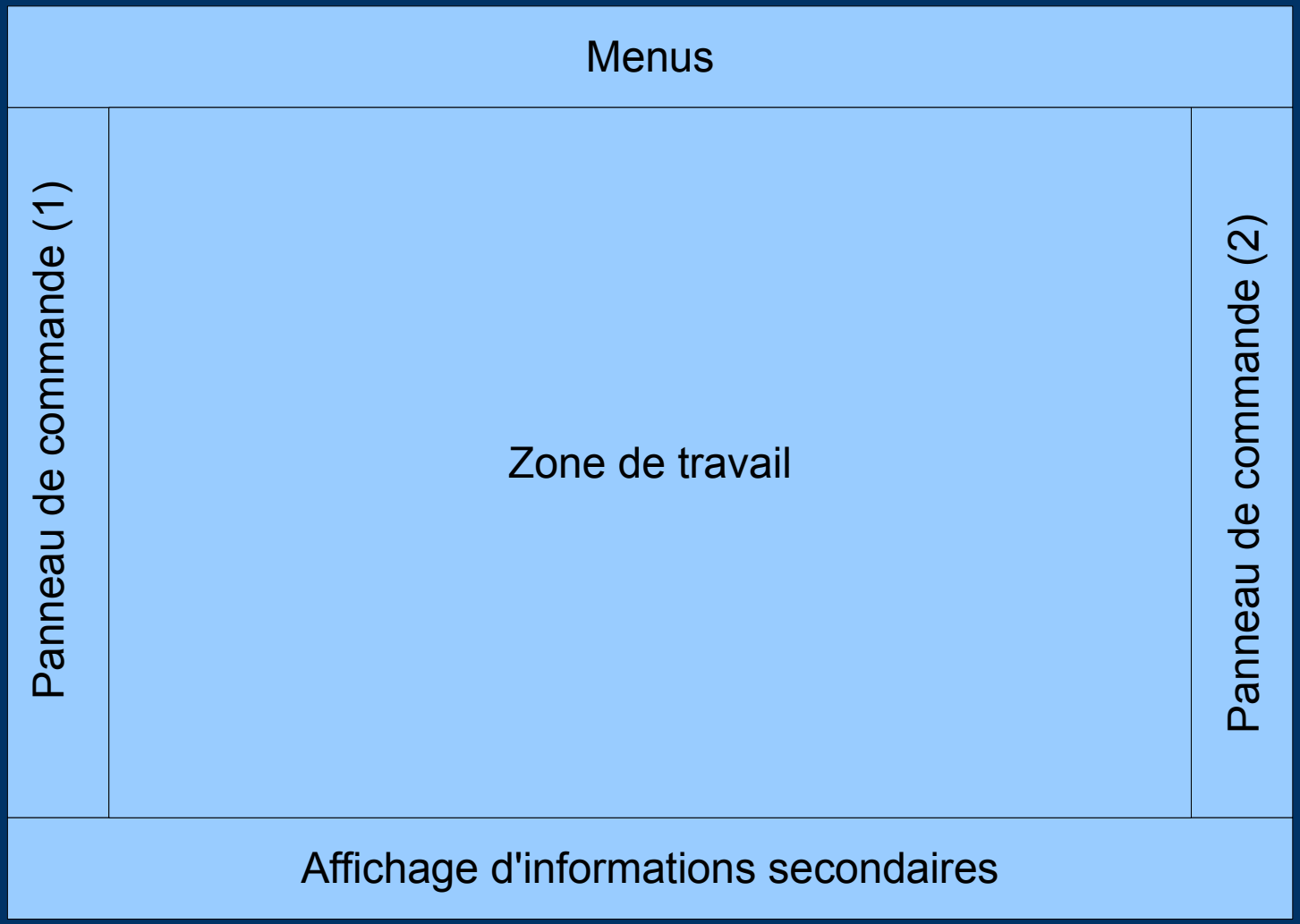
Normes et documents de référence

- http://www.oracle.com/technetwork/java/jlf-135985.html

Accessibilité et visibilité
Principes



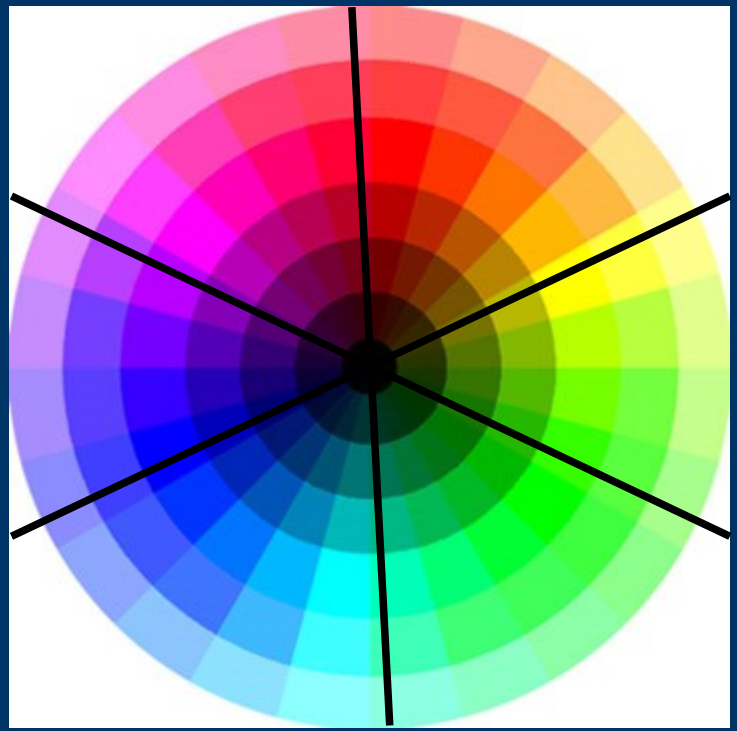
Accessibilité et visibilité
Conséquences



Charte de couleurs

Principes généraux

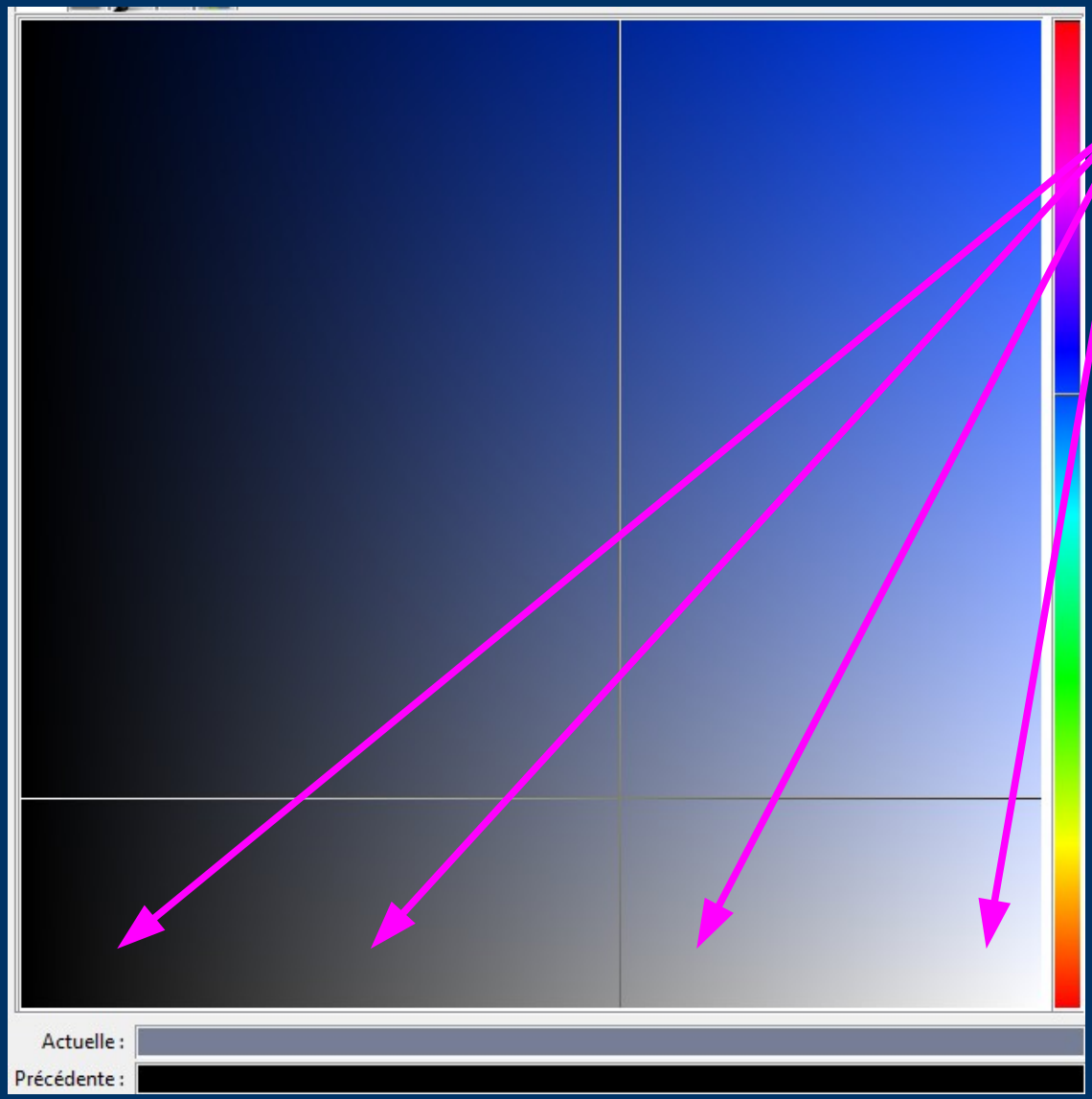
- Le bleu est plus visible en périphérie qu'au centre
  - Privilégier un cadre bleuté
  - Éviter le bleu pour les petites surfaces (texte par exemple) au centre
- Choisir des couleurs bien distinctes



Conseil généraux

- Limiter le nombre de couleurs (7 ± 2) et se tenir à celles choisies
- Eviter les couleurs trop soutenues, surtout en couleur de fond
- Eviter les couleurs trop proches
- Assigner un rôle précis à chaque couleur et le respecter dans toutes les fenêtres
  - Distinguer couleur=fonction et couleur=état
  - Tenir compte des codes « métier »

Couleur de fond (et grandes surfaces)



Choisir une couleur peu saturée (≈ gris)

- Avantages
  - Fatigue peu
  - Neutre
  - Permet de varier les autres couleurs
- Inconvénients
  - Triste
  - « absorbe » les autres couleurs

Principes du JL&F

- Principes
  - Une couleur primaire déclinée en 3 niveaux (foncé, clair, plus clair) : P1, P2, P3
  - Une couleur secondaire déclinée également en 3 niveaux (foncé, clair, plus clair) : S1, S2, S3
  - Une couleur "noire" (pour le texte) : N
  - Une couleur "blanche" (pour le fond des zones de texte) : B
- Mise en œuvre
  - Couleur primaire : bleu "triste" (gris bleuté)
  - Couleur secondaire : gris
- Choix de différenciation
  - Inverser les couleurs claires et foncées

Utilisations dans les textes

- Texte
  - Système (type "labels") : P1
  - Sélection : P3
  - Invalide : S2
  - Saisie et textes de "contrôles" (boutons, menus) : N
- Fond de texte
  - Champs de texte non-editable : S3
  - Zones de saisie : B

Autres utilisations

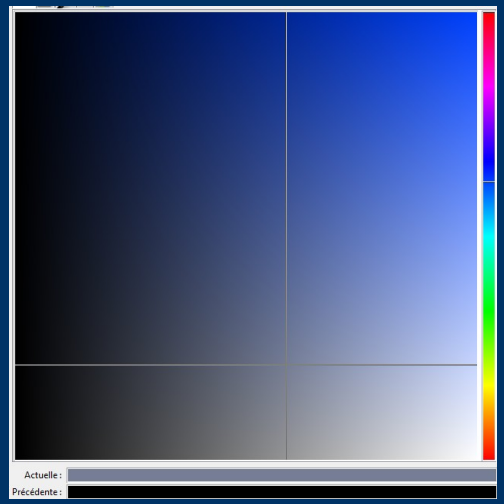
- Couleur de fond des fenêtres : S3
- Mise en évidence d'un élément sélectionné (menus et options de menu, focus clavier) : P2
- Grandes zones colorées (barre de menu) : P3
- Cadres internes des fenêtres actives : P1
- Cadres internes des fenêtres inactives : S2

Charte de polices

Principes généraux

- Limiter le nombre de polices

  3 est un bon nombre
- Associer des rôles à chaque police
- Utiliser une police « sans-serif »
- Choisir un contraste important (V/L) entre le texte et le fond



Si fond « à gauche », texte « à droite »

Si fond « à droite », texte « à gauche »

Principes du JL&F

- 4 "styles"
  - Control : Boutons, cases à cocher, menus, étiquettes, titres de fenêtre
  - Small : raccourcis clavier, bulles d'aide
  - System : composants d'arbres
    - User : zones de saisie et tables
- Styles :
    - Control : 12 points, gras
    - Small : 10 points, normal
    - System & User : 12 points, normal
- Concrètement
    - Police "Dialog" pour tous les styles

Formatage du texte

- Casse
  - Texte en minuscules, initiale en majuscule
- Longueur des lignes
  - Éviter les lignes trop longues (> 80 caractères) et trop courtes (< 30 en double colonnes, < 50 en simple colonne)
  - Ne pas utiliser de justification totale
- Aérer le texte
  - Introduire des espaces supplémentaires entre les paragraphes

Textes
Consignes générales (1)

- Pour les actions, préférer des verbes à l'infinitif plutôt que des noms (« Afficher » au lieu de « Affichage »)
- Établir une bijection action/nom
- Éviter les abréviations ; si cela est impossible, unifier le processus d'abréviation
- Préférer des textes courts à des textes verbeux
- Éviter les formes passives et négatives
- Ordre dans le texte = ordre des opérations à effectuer (i.e. éviter le « faire ceci après avoir fait cela »)

Textes
Consignes générales (2)

- Messages optionnels = barre d'état
- Messages importants = boîte de dialogue modale
- S'adapter au vocabulaire et aux compétences de l'utilisateur

Disposition des éléments d'une boîte de dialogue

Disposition générale

- Regrouper les composants d'une boîte de dialogue par thème
- Adapter l'ordre de présentation à l'ordre d'importance ou d'utilisation (c.f. adresse)
- Préférer plusieurs fenêtres/onglets simples à une seule fenêtre très complexe
- Utiliser une « zone d'information » ou des bulles d'aide pour les explications plutôt que du texte dans la boîte de dialogue

Libellés

- Faire précéder chaque composant d'un libellé
- Choisir des libellés courts mais clairs
- Alignement horizontal
  - Si longueurs voisines

    Aligner les libellés entre eux, les composants entre eux
  - Si longueurs très différentes

    Coller les composants aux libellés et aligner les composants entre eux
- Alignement vertical
  - Aligner le haut du libellé avec le haut du composant
- Éviter les zones de saisies dans du texte (ordre dépend de la langue)

Ordre et espacement de base

- Ordre
  - Utiliser un ordre relatif à l'ordre de la langue (dans la suite du texte, pour simplifier, j'utilise un ordre "absolu" par rapport au français)
  - Mettre les boutons de commande en bas
  - Mettre les éléments principaux en haut
- Espacement de base
  - Utilise des multiples de 6 pixels (avec un -1 éventuel pour tenir compte des bords blancs en bas et à droite)
  - Par la suite, on appellera "espacement" 5/6 pixels, "double espacement" 11/12 pixels, triple espacement 17/18 pixels, etc.
  - Prévoir un double espacement autour du contenu de la boîte de dialogue

Boutons de commande

- Position
  - En bas
  - Alignés à droite
- Espacement
  - Triple espacement les séparant du reste
  - Simple espacement entre chacun des boutons
- Largeur

  Donner une largeur identique à tous les boutons, donc tenant compte du texte le plus long, qui dépend de la langue (!)

Utilisation des cadres avec titre

- À utiliser pour structurer...
- … mais à limiter car prend de la place
- À ne pas imbriquer
- Ne pas utiliser pour étiqueter un seul composant
- Pour un simple groupe de boutons radios, préférer un "Label"

Utilisation des composants (1)
Choix fermé

- Choix restreint à 2 ou 3 options
  - Boutons radio (JRadioButton)
- Choix multiple possible
  - Liste (JList)
- Place disponible limitée
  - Liste de sélection (JcomboBox)
- Si adéquat, fixer une valeur par défaut

Utilisation des composants (2)
Saisie d'une valeur numérique

- Si unité discutable
  - Proposer le choix de l'unité
- Utiliser un contrôle graphique facilite l'utilisation
  - JSlider, JScrollBar
- Doubler le contrôle graphique d'une zone de saisie pour permettre la précision/la sélection rapide
- Éventuellement, forcer l'alignement du contrôle graphique sur des valeurs données

Fenêtres de présentation

<< *Splash screen* >>

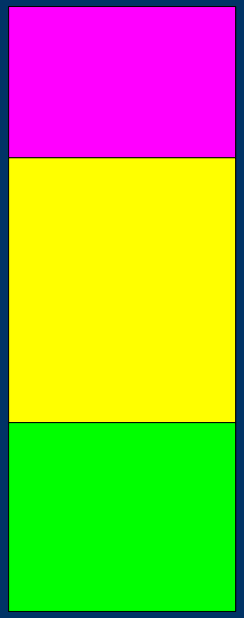
<< *A propos* >>

Menus

Types de menus

- Menus généraux
  - Doivent permettre de retrouver toutes les actions possibles à un moment donné
  - Accès lent
- Menus contextuels
  - Doivent permettre de retrouver les commandes essentielles applicables à un objet donné
  - Accès immédiat

Disposition des menus & palettes



Outils fréquemment utilisés au centre

Outils importants au début

Outils secondaires à la fin

Structure des menus

- Limiter le nombre d'éléments par menus
  - 10 pour un utilisateur débutant
  - 20 pour un utilisateur expérimenté
- Limiter la profondeur
  - Si possible, pas plus d'un niveau d'imbrication
- Préférer le « grisage » des options non disponibles à leur non-affichage
- Regrouper les items liés
- Séparer les items *destructeurs* (« supprimer », etc.)

Ordre des menus

- Si les menus suivants doivent figurer, les mettre dans cet ordre
  - Fichier
  - Edition
  - Format
  - Affichage
  - Aide
- Mettre les menus supplémentaires Après "Affichage"

Représentation des menus et des options

- Limiter les titres des menus à 1 mot (nom : Fichier, Affichage, etc.)
- Mettre des titres courts pour les options
- Mettre une majuscule aux noms de menus et d'options
- Définir un code mnémonique pour chaque menu et chaque option et un raccourci clavier pour les actions les plus fréquentes
- Rajouter une icône devant l'option si un bouton d'une barre d'outil permet également d'effectuer la même action

Points de suspension

- Cas d'utilisation
  - Si l'action nécessite des paramètres (à rentrer via une boîte de dialogue) pour être exécuté
    - Exemple : Enregister sous...
  - Mais pas si l'action consiste en la saisie d'options via une boîte de dialogue
    - Exemple : Mise en page
- Autre intérêt
  - Eviter les sous-sous-menus

Actions type "Bascule"

- Eviter les menus à intitulé changeant (problème d'ambiguïté)
- Utiliser plutôt
  - Des options de menus "case à cocher" Si 2 états
  - Des options de menus "boutons radio" Si plus de 2 états
  - Des séparateurs pour regrouper les boutons radio ou cases à cocher fonctionnant ensemble

Menus standards

Fichier	
Nouveau	Ctrl-N
Ouvrir...	Ctrl-O
Fermer	Ctrl-W
Enregistrer	Ctrl-S
Enregistrer sous...	
Mise en Page	
Imprimer	Ctrl-P
Quitter	Ctrl-Q

Edition	
Annuler	Ctrl-Z
Refaire	Ctrl-Y
Couper	Ctrl-X
Copier	Ctrl-C
Coller	Ctrl-V
Effacer	Suppr
Tout Sélectionner	Ctrl-A
Chercher...	Ctrl-F
Chercher le suivant	Ctrl-G

Aide	
Aide de *Logiciel*	F1
A propos de *Logiciel*	

Attente

Représenter une attente

- Motivation
- Moyens
  - Rien : si $t < 2s$
  - Curseur d'attente si $2s \leq t \leq 6s$
  - Barre de progrès si $t > 6s$

Icônes & « boutons graphiques »

Consignes générales

- Tailles standards
  - Icônes : 16 x 16 ou 32 x 32
  - Boutons graphiques : 16 x 16 ou 24 x 24
- Utiliser des dessins plus schématiques que réalistes
- N'utiliser la couleur que pour agrémenter une forme de base
- Utiliser le format GIF
- Unifier la représentation
  - 2D/3D
  - Choix d'un système unique de métaphore
- Choisir des symboles universels

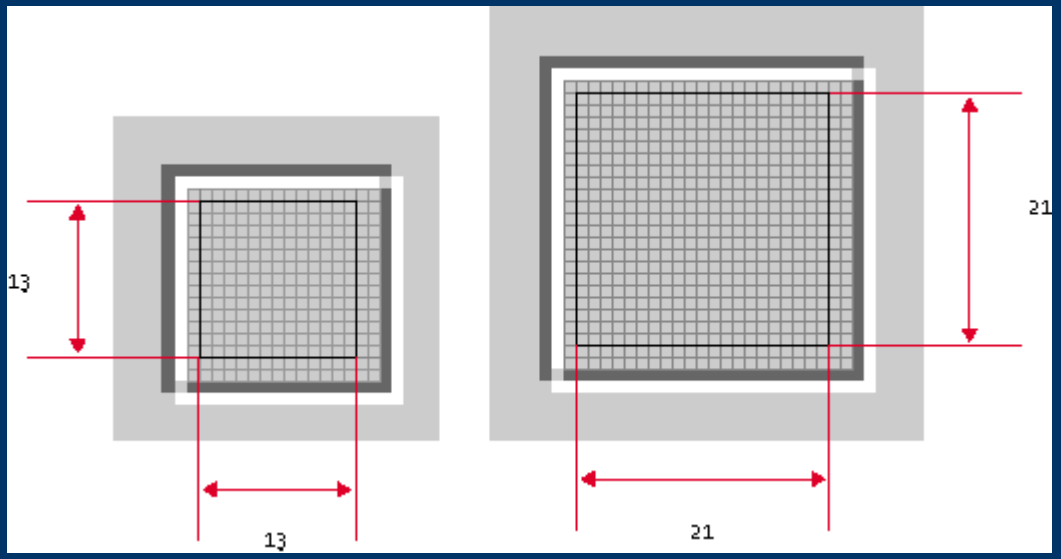
Dessin d'une icône

1. Tracer la forme de base en noir
2. Remplir avec une couleur de la charte graphique
3. Éclaircir le haut et la gauche des « intérieurs »
4. Rajouter quelques détails
5. Rajouter éventuellement un dégradé (du centre vers l'extérieur) pour imiter un reflet
6. Rajouter un motif type damier pour éviter les problèmes en cas de nombre de couleurs réduit
7. Définir le contour comme « transparent »
8. Tester le résultat

Dessin d'un bouton graphique

- Raisonner comme pour une icône
- Prévoir une zone de dessin réduite pour décoller le dessin de son cadre

exemple extrait de Java Look&Feel Design Guidelines vol. 1



Utilisation des « badges »

- Principe général

  Rajouter un symbole standardisé sur les icônes ou boutons graphiques pour mieux préciser leur fonction et placer ce symbole à une place standard (en bas à droite en général)

- Exemples

  sous-menu : 

  création d'un objet : 

  ajout d'un objet : 

- Remarque

  ne pas combiner les « badges » sur une même icone

Erreurs de l'utilisateur

Types d'erreur

- Erreurs d'intention

  L'utilisateur choisit délibérément une action, mais celle-ci ne permet pas d'atteindre le but visé

- Erreur d'exécution
  - Erreurs de perception

    L'utilisateur perçoit mal réel l'état du système
  - Erreurs de cognition

    L'utilisateur fait un mauvais raisonnement, par exemple car il n'a pas bien mémorisé une information affichée ailleurs
  - Erreurs de motricité

    L'utilisateur fait une manipulation accidentelle (clic à côté, appui fortuit sur une touche de fonction, etc.)

Prévenir les erreurs

- Erreurs d'intention
  - Fournir une aide en ligne de type « How to »
  - Guider l'utilisateur par des suggestions et des retours d'information
- Erreurs de perception
  - Travailler la mise en évidence
  - Accentuer le retour utilisateur, surtout sur les changements d'état
- Erreurs de cognition
  - Limiter la mémorisation (rappeler les informations utiles, utiliser des codes mnémoniques)
- Erreurs de motricité
  - Désactiver les commandes inutilisables
  - Agrandir la taille des objets
  - Permettre d'éviter les saisies au clavier

Signaler les erreurs

- Le message d'erreur doit
  - Décrire le problème
  - Décrire le(s) moyen(s) de le corriger
- Forme du message
  - Clair, sans ambiguïté
  - Adapté au niveau de l'utilisateur
  - Non culpabilisant

Permettre la réparation des erreurs

- Autoriser le retour arrière
- Autoriser l'interruption de tâches longues
- Faciliter l'accès à l'aide en ligne
- Ré-afficher la fenêtre de configuration permettant d'éviter que l'erreur ne se produise

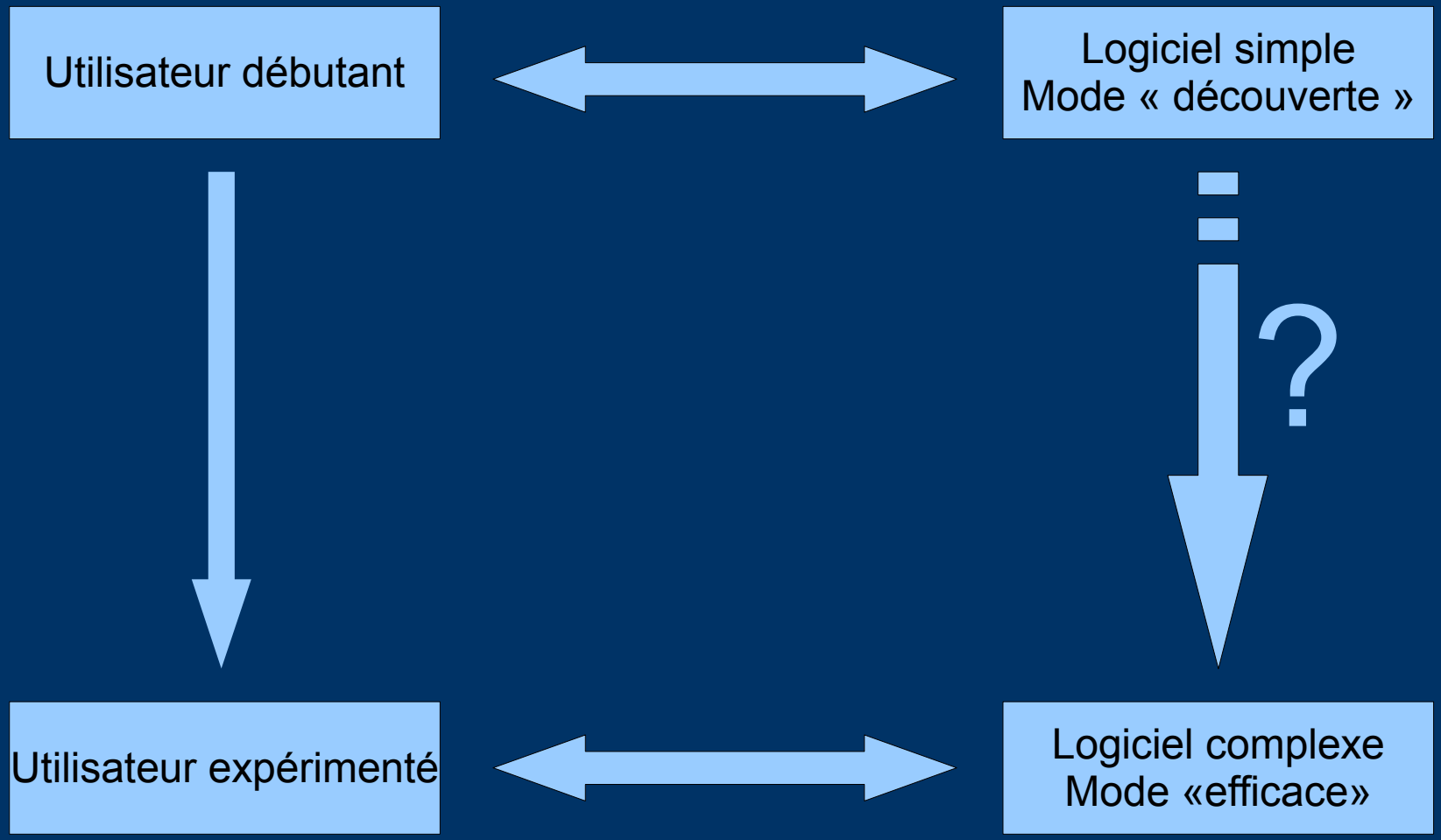
Aide en ligne

Formes d'aide

- Manuel utilisateur
  - (presque) jamais regardé
- Manuel de référence
  - peu regardé, souvent trop complexe, difficile de s'y retrouver
- « How to »
  - mode souhaité par l'utilisateur, mais trouve rarement la question qu'il se pose
- Bulles d'aide
  - très succinct, mais immédiat, au bon moment
- Astuce du jour
  - (force) l'exploration

Adaptation à l'utilisateur

Adéquation utilisateur/interface



Résoudre l'adaptation
Quelques pistes

- À l'initiative de l'utilisateur
  - Autoriser l'utilisateur à complexifier le logiciel en rajouter des items de menu par exemple
- Par suggestion à l'utilisateur
  - Affichage d' « astuces du jour »
  - Affichage systématique de bulles d'aide désactivable manuellement par l'utilisateur
- Automatiquement en se fondant sur
  - Le taux d'utilisation du logiciel
  - Le niveau des fonctionnalités utilisées
  - L'utilisation des accès accélérés
  - La consultation de l'aide en ligne

Internationalisation

Bruno MERMET
Septembre 2010

Externalisation des ressources

Introduction

- Définition

  Séparation les « ressources » (texte, images, sons, etc.) du code

- Intérêt

  Pouvoir « localiser » un logiciel sans devoir le recompiler

- Principe

  Les ressources, chacune identifiée par une clé (chaîne de caractères), sont regroupées (*ResourceBundle*) et leur valeur est

  - soit stockées dans un fichier texte (pour les ressources de type texte)
  - soit produites par une classe indépendante

Identification d'un ResourceBundle

- Identifiant de base
  - Un nom
- Localisation
  - Un suffixe pour la langue
  - (Un deuxième suffixe pour le pays)
    Différencier anglais britannique/américain par exemple
  - (Un troisième suffixe pour le système/navigateur)
- Exemples d'un même ensemble de ressources :
  - mesTextes, mesTextes_fr
  - mesTextes_fr_FR, mesTextes_fr_FR_UNIX
  - mesTextes_en_GB

Spécifier un lieu

- Principe
  - Créer une instance de la classe java.util.Locale
    - Locale(String langue)
    - Locale(String langue, String pays)
    - Locale(String langue, String pays, String variant)
- Chaînes références
  - Constantes dans Locale (Locale.FRENCH, Locale.FRANCE, etc.).
  - Normes ISO-639 et ISO-3166
- Spécifier la localisation par défaut

  public static void setDefault(Locale l)

Recherche de la version locale d'un ensemble de ressources

- Ordre
  - Du plus précis au moins précis
  - D'abord pour la localisation spécifiée, puis pour la localisation par défaut
- Exemple
  - Contexte
    - Localisation spécifiée : fr_FR_WIN
    - Localisation par défaut : en_GB
  - Ordre de recherche
    - mesTextes_fr_FR_WIN, mesTextes_fr_FR, mesTextes_fr
    - mesTextes_en_GB, mesTextes_en, mesTextes
- Conséquence

  Pour éviter une exception, toujours prévoir une version non suffixée

Utilisation des ressources

- Création de l'objet Locale :

  Locale ici = new Locale(*maLangue*, *monPays*);
- Obtention de l'ensemble de ressources

  ResourceBundle mesRessources =
  ResourceBundle.getBundle(*monEnsemble*, ici);
- Obtention d'une ressource

  String maRessource =
  mesRessources.getString(*identifiantDeMaRessource*);

  (ou getObject)
- Retrouver la description des ressources
  - Via un ClassLoader
  - Les « . » sont remplacés par des « / »
  - Les classes sont prioritaires sur les fichiers texte

Spécifier des ressources dans des fichiers textes

- Application

  Uniquement pour les chaînes de caractères !

- Fichiers
  - Un fichier par localisation d'ensemble de ressource
  - Nom : *nomEnsembleRessources.localisation*.properties
  - Exemples
    - mesTextes_fr_FR.properties
    - mesTextes.properties
    - mesTextes_gb.properties
- Structure d'un fichier
  - Une ligne par ressource :

    clef = valeur
  - Des commentaires sur des lignes commençant par un #

Spécifier des ressources dans des classes

Format des classes

- Nom : nomEnsembleRessources_localisation
- Hériter de ListResourceBundle
- Définir la méthode Object[][] getContents() renvoyant un tableau de tableaux [clé, valeur]

Conseils de structuration

- Préférer plusieurs ensembles de ressource à un seul gros
- Regrouper les ressources selon une structuration logique
- Hiérarchiser les ensembles de ressources pour mieux les structurer

Affichage « localisé » des nombres, des dates et des heures

Nombres

- Problématique
  - Nombres, sommes d'argent, pourcentage ne sont pas affichées de la même façon suivant les pays
- Solution
  - Utiliser la classe NumberFormat en la paramétrant par une instance de java.lang.Locale
- Exemple

```
Locale ici = new Locale(« fr », « FR »);
Double d = new Double(1234.56);
NumberFormat nf = NumberFormat.getNumberInstance(ici);
String sortie = nf.format(d);
```

- Prolongation
  - getCurrencyInstance, getPercentInstance()

Dates et heures

- Utiliser les méthodes de la classe DateFormat :
  - DateFormat getDateInstance(int format, Locale ici)
  - DateFormat getTimeInstance(int format, Locale ici)
  - DateFormat getDateTimeInstance(int format, Locale ici)
- Format :
  - DateFormat.DEFAULT
  - DateFormat.SHORT
  - DateFormat.MEDIUM
  - DateFormat.LONG
  - DateFormat.FULL

Localisation des messages composés

Problématique

- Exemple
  - Soit un logiciel
    - demandant à un utilisateur
      - Son animal préféré (exemple : âne)
      - Sa couleur préférée (exemple : noir)
    - Et affichant une phrase de synthèse type :
      - Vous aimeriez avoir un âne noir
  - En anglais, traduire mot-à-mot donnerait
    - You would like to have a donkey black
  - Or la version correcte serait :
    - You would like to have a black donkey
- Le problème

  La phrase à afficher est paramétrée par des éléments eux-mêmes internationalisés
- Remarque : limiter ce genre de situation

Solution (1)

- Principe
  - Externationnaliser des phrases paramétrées
  - Utiliser la classe MessageFormat pour les concrétiser
- Concrètement
  - Dans la phrase, faire figurer les paramètres ainsi :
    - {numero} (pour les chaînes simples, par exemple)
    - {numero,typeFormat}
    - {numero,typeFormat,styleFormat}

typeFormat	StyleFormat possibles
number	integer, currency, percent
date	short, medium, long, full
time	short, medium, long, full

Solution (2)

- Application sur l'exemple
  - Fichier mesTextes_fr.properties
    - Chat = chat
    - Noir = noir
    - Synthese = Vous aimeriez avoir un petit {0} {1}.
  - Fichier mesTextes_en.properties
    - Chat = cat
    - Noir = black
    - Synthese = You would like to have a {1} {0}.

Solution (3)

- Création de la phrase de synthèse

```
Locale ici = new Locale(« fr », « FR »);

ResourceBundle ressources =
   ResourceBundle.getBundle(« mesTextes », ici);

Object [] parametres = {ressources.getString(« Chat »),
   ressources.getString(« Noir »)};

MessageFormat formateur = new MessageFormat(« »);

formateur.setLocale(ici);// pas indispensable ici où on n'a ni
   date ni nombre

formateur.applyPattern(ressources.getString(« Synthese »));

String sortie = formateur.format(parametres);
```

Localisation : gestion du pluriel

Exemple

- Exemple repris de http://download.oracle.com/javase/tutorial/i18n/format/choiceFormat.html
  - On veut afficher le nombre de fichiers trouvés sur un disque ; 3 cas :
    - *Il* n'y a pas de fichier *sur* monDisque
    - *Il* y a 1 fichier *sur* monDisque
    - *Il* y a 2 fichiers *sur* monDisque (ou plus)
  - Après avoir identifié les paramètres, on va stocker le motif et les différentes possibilités dans un ensemble de ressource, et utiliser un objet ChoiceFormat pour gérer ce qui dépend de la quantité
- Fichier de ressource mesTextes_fr_FR
  - Phrase = Il {0} sur {1}
  - Aucun = n'y a pas de fichier
  - Un = y a 1 fichier
  - Plusieurs = y a {2} fichiers

ChoiceFormat

Classe permettant un affichage conditionnel

- Soit le programme suivant :

```
Double[] limites = {0,2,6};
String [] textes = {« t1 », « t2 », « t3 »};
ChoiceFormat f = new ChoiceFormat(limites,textes);
For (int i = -1 ; i < 8 ; i++) {
        System.out.print(f.format(i)+ « »);
}
```

- L'affichage produit sera :

  t1 t1 t1 t2 t2 t2 t2 t3 t3

- Explications
  - Si limites[i] <= i < limites[i+1], on affiche textes[i]
  - Si i < limites[0], on affiche textes[0]
  - Si i > limites[limites.length-1], on affiche textes[textes.length-1]

Retour sur MessageFormat

- Rappel : utilisation vue de MessageFormat :

```
Object [] parametres = {new Double(1), « Chat », « Noir »};
MessageFormat formateur = new MessageFormat(« »);
formateur.setLocale(ici);
formateur.applyPattern(«Vous aimeriez avoir {2,number,integer} {0}
    {1} »);
String sortie = formateur.format(parametres);
```

- Mais on peut aussi écrire :

```
Object [] parametres = {new Double(1), « Chat », « Noir »};
MessageFormat formateur = new MessageFormat(« »);
formateur.setLocale(ici);
formateur.applyPattern(«Vous aimeriez avoir {2,} {0} {1} »);
Format [] f= {NumbertFormat.getIntegerInstance(ici),null,null};
formateur.setFormats(f);
String sortie = formateur.format(parametres);
```

MessageFormat et ChoiceFormat

- Récursivité

  Dans le cas où l'un des « formats » d'un paramètre d'un MessageFormat est ChoiceFormat, la substitution des paramètres est réappliquée une foit le ChoiceFormat appliqué

- Application

```
String disque = « monDisque »;//par exemple
int quantité = 1; //par exemple
String phrase = « Il {0} sur {1} »;
double[] limites = {0, 1, 2};
String[] debuts = {« n'y a pas de fichier », « y a un fichier », « y a {2}
    fichiers »};
ChoiceFormat cf = new ChoiceFormat(limites, debuts);
MessageFormat mf = new MessageFormat(« »);
mf.applyPattern(phrase);
Format [] f = {cf, null};
mf.setFormats(f);
System.out.println(mf.format({quantite,disque,quantite});
```