Cours 2b : programmation des interfaces graphiques

Anastasia.Bezerianos@lri.fr

(partie de la présentation basée sur des transparents de Michel Beaudouin-Lafon)

système interactif vs. système algorithmique

système algorithmique (fermé) :

- lit des entrées, calcule, produit un résultat
- il y a un état final

système interactif (ouvert) :

- évènements provenant de l'extérieur
- boucle infinie, non déterministe

interfaces graphiques

l'interaction graphique : les entrées sont spécifiées directement à partir des sorties

périphérique d'entrée spécifie dans une commande une position à l'écran qui désigne un objet précédemment affiché par le système (cette désignation directe est appelée *pointage*). Elle est familière dans le monde physique, donc le succès de ces interfaces

problème

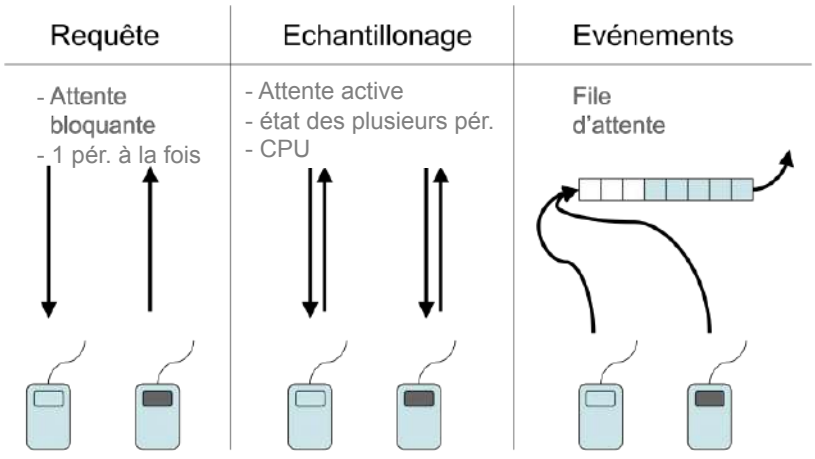
- nous avons appris à programmer des algorithmes (la partie "calcul")
- la plupart des langages de programmation (C, C++, Java, Lisp, Scheme, Ada, Pascal, Fortran, Cobol, ...) sont conçus pour écrire des algorithmes, pas des systèmes interactifs

problème

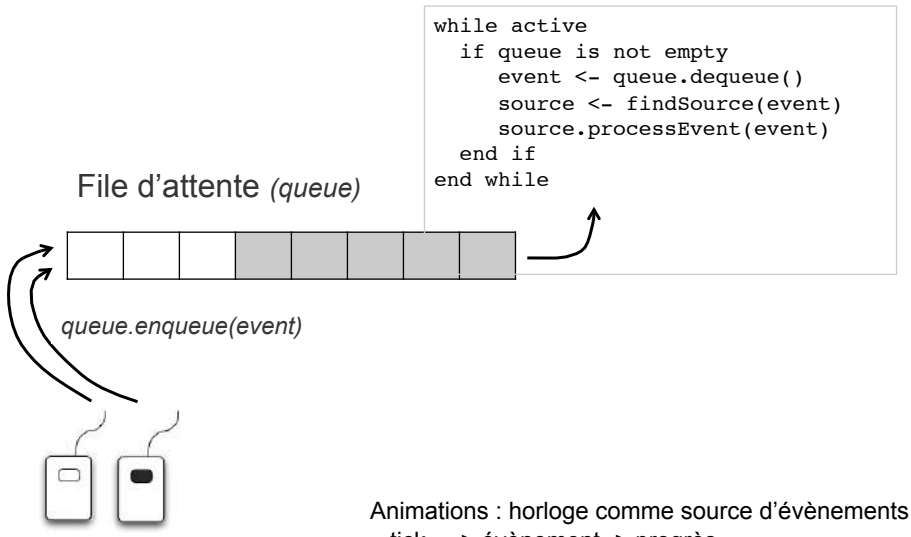
entée/sortie des langues algorithmiques

- instructions de sortie (`print, put, send,…`) pour envoyer des données aux périphériques
- instructions de lecture (`read, get, receive, …`) pour lire l'état ou changement d'états de périphériques d'entrée, du façon bloquante

comment gérer les entrées

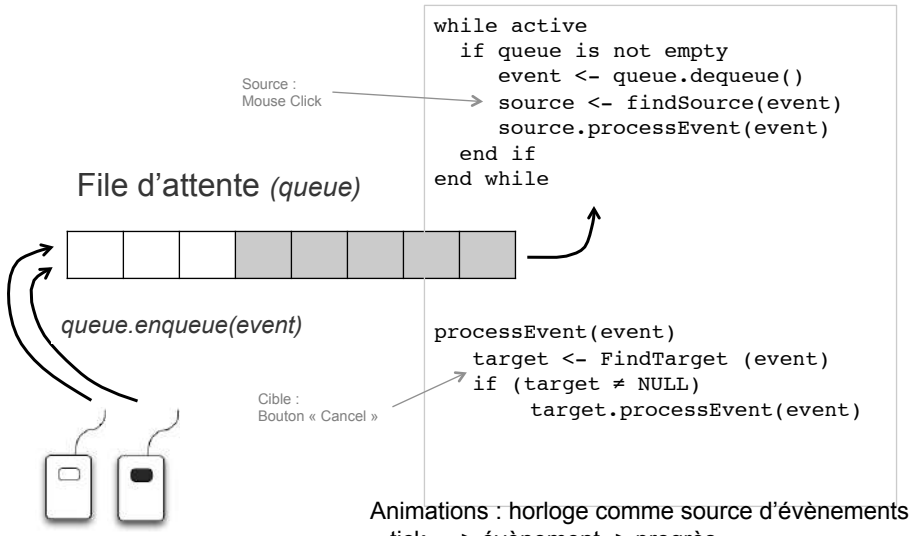


programmation événementielle



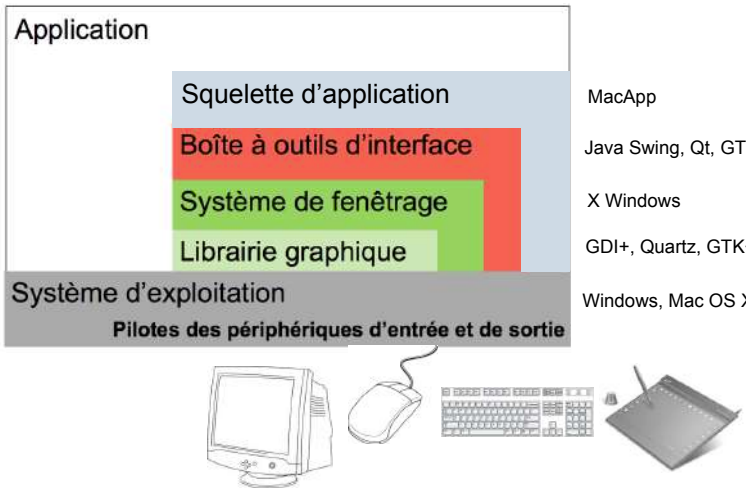
Animations : horloge comme source d'évènements
« tick » -> évènement -> progrès

programmation événementielle

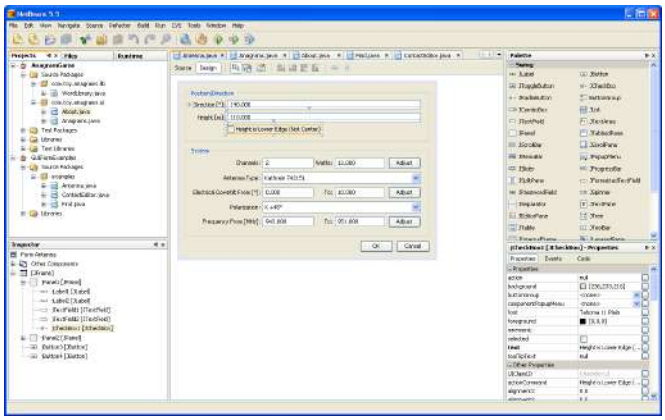


Animations : horloge comme source d'évènements
« tick » -> évènement -> progrès

couches logicielles



constructeurs d'interface



Exemples : MS Visual Studio (C++, C#, etc.), NetBeans (Java), Interface Builder (ObjectiveC)

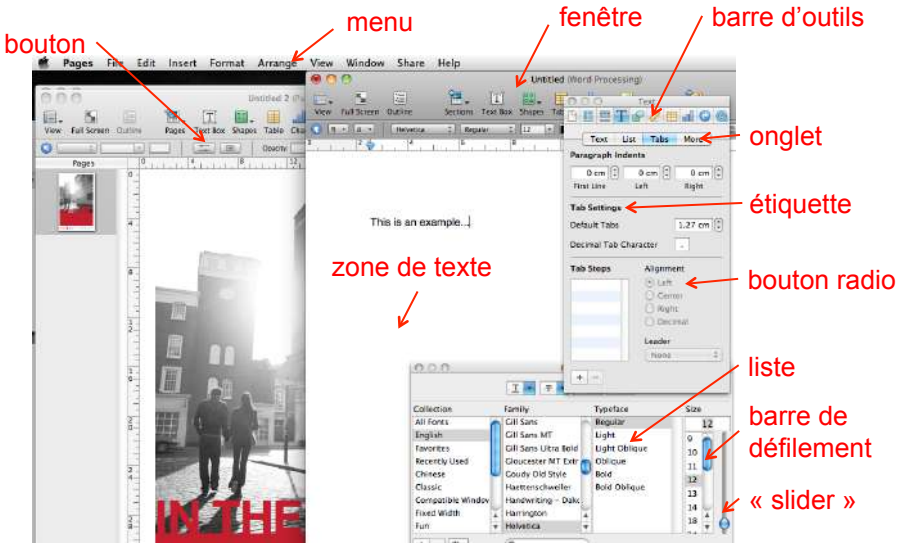
boîte à outils d'interface

bibliothèque d'objets interactifs (les « widgets ») que l'on assemble pour construire l'interface

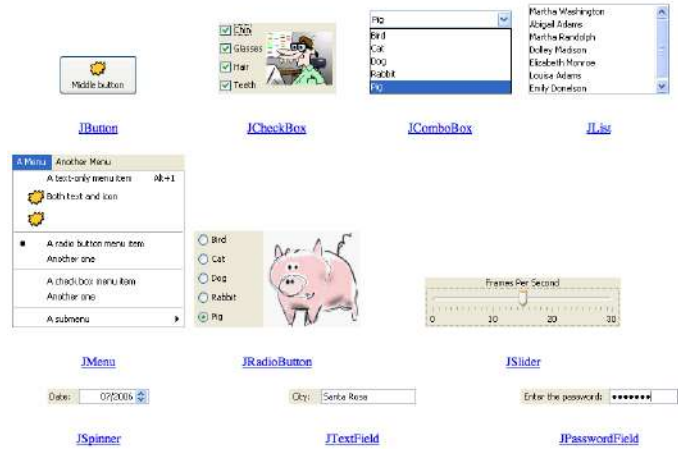
fonctionnalités pour faciliter la programmation d'applications graphiques interactives (et gérer les entrées)

Windows : MFC, Windows Forms (.NET)
Mac OS X : Cocoa
Unix/Linux : Motif, QT, GTK+
Multiplateforme : Java AWT/Swing

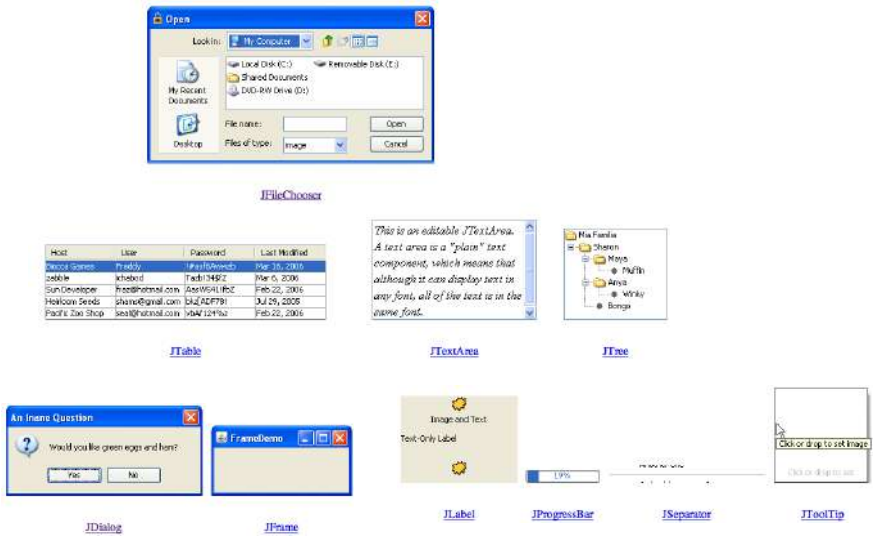
les « widgets »



les widgets de Swing



les widgets de Swing



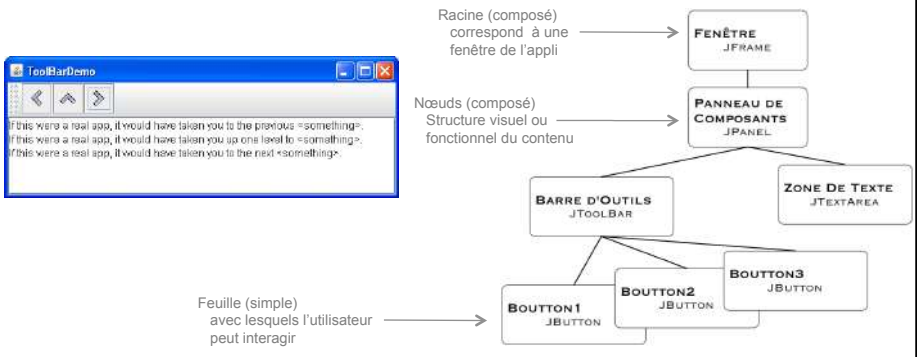
arbre des widgets

- widgets « simples »
  - buttons, barres de défilement, …
- widgets « composés »
  - Distinés à contenir d'autres widgets (simples ou composés)
  - Boites de dialogue, menus, …

arbre des widgets

représentation hiérarchique de la structure des widgets

- un composant ne peut appartenir qu'à un seul « container »



facettes d'un widget

présentation
- apparence graphique

comportement
- réactions aux actions de l'utilisateur

interfaces d'application :
notifications de changement d'état

Bouton:
cadre avec un nom à l'intérieur
« enfoncement » ou inversion vidéo lorsque l'on clique dessus
grisé quand non-disponible
+ fonction appelée lorsque le bouton est cliqué

facettes d'un widget

présentation
- apparence graphique

comportement
- réactions aux actions de l'utilisateur

interfaces d'application :
notifications de changement d'état
- fonctions de rappel (« callbacks ») (Swing)
- variables actives (Tcl/Tk)
- envoi de message (Qt)

fonctions de rappel

Enregistrement lors de la création du widget

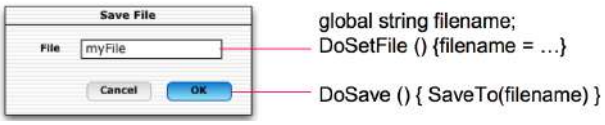
Appel lors l'activation du widget

fonctions de rappel

Problème : spaghetti des callbacks

Partage d'état entre plusieurs callbacks par:

- variables globales
  - Trop dans une application réelle
- arbre des widgets : la fonction de rappel est appelée en lui passant le widget qui l'a déclenché
  - Fragile si l'on change la structure, insuffisante pour d'autres données pas associés aux widgets
- « jeton » (token) : donnée enregistrée avec la callback, passée automatiquement au moment de l'appel



fonctions de rappel

```
/* fonction de rappel */
void DoSave (Widget w, void* data) {
     /* récupérer le nom de fichier */
      filename = (char**) data;
     /* appeler la fonction de l'application */
     SaveTo (filename);
     /* fermer la boîte de dialogue */
     CloseWindow (getParent(getParent(w)));
}

/* programme principal */
main () {
     /* variable contenant le nom du fichier */
     char* filename = "";
     …
     /* créer le widgets et lui associer sa callback */
     ok = CreateButton (....);
     RegisterCallback (ok, DoSave, (void*) &filename);
     …
     /* boucle de traitement des événements */
     MainLoop ();
}
```

« event listeners » (Java)

variante des callebacks adaptée au Java:

methods de type AddListener spécifient non pas une fonction de callback, mais un objet (le *listener*)

lorsque le widget change d'état, il déclenche une méthode prédéfinie du *listener* (par exemple *actionPerformed*)

« event listeners » (Java)

- Un composant (widget) qui crée des événements est appelé source
- Le source délègue le traitement de l'événement au listener
- Un listener doit s'inscrire auprès du composant source des événements qu'il veut traiter.

- Un événement peut provenir :
  - du clavier, un clique souris, un passage de la souris,..

- A chaque type d'événement, une classe (existante)
- A chaque type d'événement, son listener (à faire)

« event listeners » (Java)

```
public class ClickListener implements ActionListener
{
    public void actionPerformed(ActionEvent e){
           JButton button = (JButton)e.getSource();
           …
    }
}

…
ClickListener listener = new ClickListener();
JButton button = new JButton(''Click me'');
button.addActionListener(listener);
…
```

« event listeners » (Java)

Anonymous Inner classes

"new <nom-de-classe> () { <corps> }"

cette construction fait deux choses :

- elle crée une nouvelle classe, sans nom, qui est une sous-classe de <nom-de-classe> définie par <corps>
- elle crée une instance (unique) de cette nouvelle classe et retourne sa valeur

cette class a accès aux variables et méthodes de la class dans la quelle elle est définie

« event listeners » (Java)

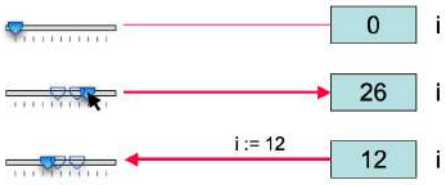
Anonymous Inner classes

```
…
button.addActionListener(new ActionListener(){
          public void actionPerformed(ActionEvent e){
                …
          }
});
…
panel.addMouseListener(new MouseAdapter(){
          public void mouseClicked(MouseEvent e){
                …
          }
});
```

Fonctions et évènements prédéfinis

variables actives

lien bi-directionnel entre une variable d'état du widget et une variable de l'application



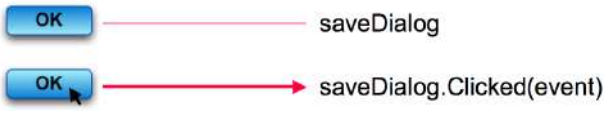

```
main (){
   int i = 0;
   …
   /* widget */
   nc = CreateSlider (…);
   /* var active */
   SetIntegerActiveVariable (nc, &i);
   …
}
```

problèmes

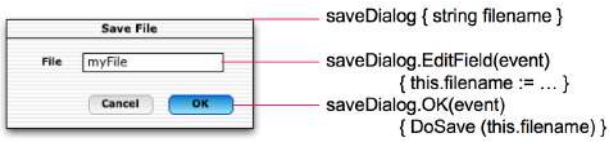
- limité aux types simples
- lien de retour peut être coûteux

envoi de message

association d'un objet à un widget et de méthodes de l'objet aux changements d'état

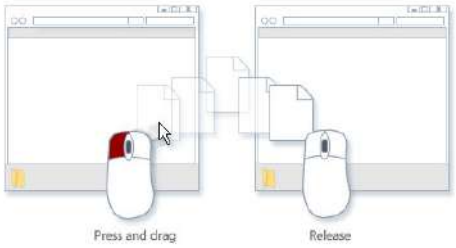


meilleure encapsulation



« drag-and-drop »

Quels sont les « widgets » affectés ?
Quels sont les événements?

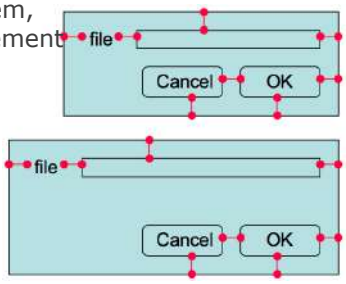


Exercice : comment décrire cette interaction avec un « event listener » ?

placement de widgets

Boîtes à outilles contrôlent le placement des widgets :

- il faut être indépendant de la taille des widgets
  (menu au moins égale à son plus large item, en changement de taille la barre de défilement et le texte s'ajustent)
- gestionnaires de géométrie, dans le widgets composés



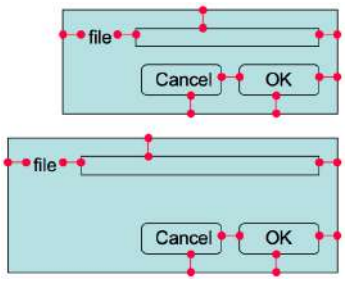
placement de widgets

règles générales

- imbrication géométrique d'un widget fils dans son parent
- contrôle par le parent du placement de ses fils

algorithme de placement

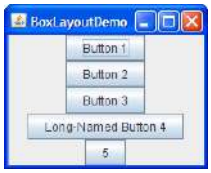
- taille naturelle de chaque fils
- taille et position finales imposées par le parent
- contraintes : grille, formulaire, etc.



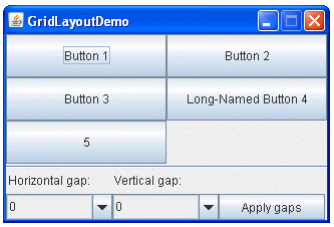
« layout managers » (Swing)

BorderLayout

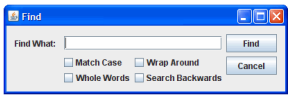
FlowLayout



BoxLayout



GridLayout



GroupLayout

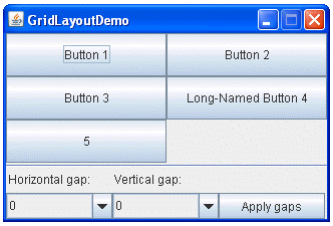
http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

« layout managers » (Swing)

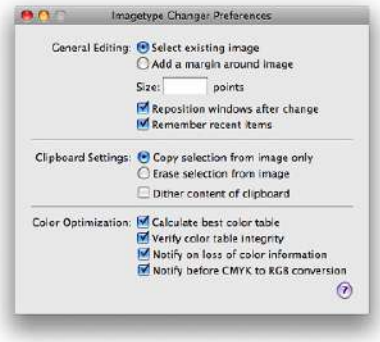
```
GridLayout gridLayout = new GridLayout(0,2);

JPanel gridPanel = new JPanel();
gridPanel.setLayout(gridLayout);

gridPanel.add(new JButton("Button 1"));
gridPanel.add(new JButton("Button 2"));
gridPanel.add(new JButton("Button 3"));
gridPanel.add(new JButton("Long-Named Button 4"));
gridPanel.add(new JButton("5"));
```

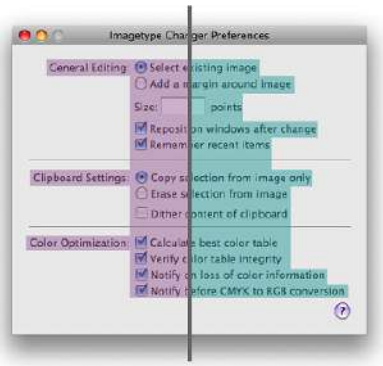


guides de placement (Mac OS X)



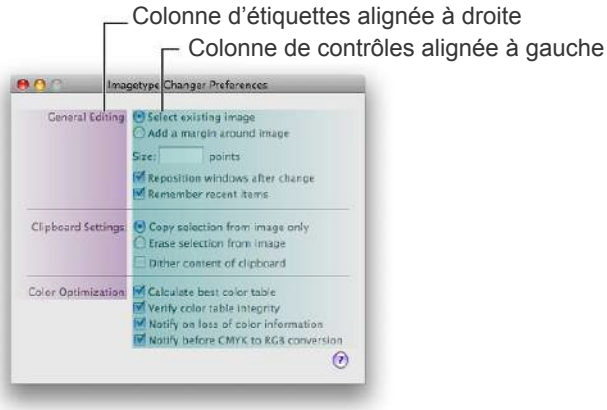
guides de placement (Mac OS X)

« **Center-equalization** » : équilibre visuelle du contenu d'un composant, à droite et à gauche de la médiane



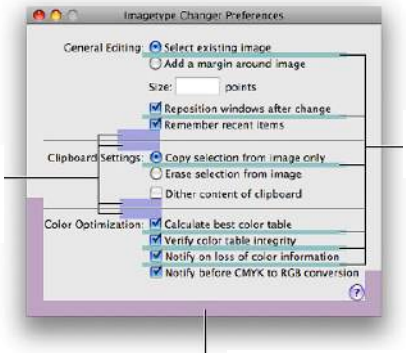
guides de placement (Mac OS X)

Alignement



guides de placement (Mac OS X)

Espacement



Même hauteur avant et après les lignes de séparation

Même espace entre des contrôles groupés

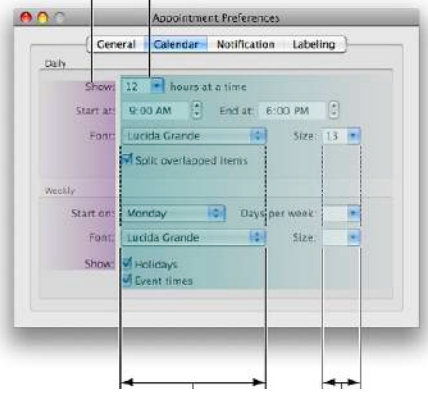
Même espace dans tous les trois côtés

guides de placement (Mac OS X)

Alignement et cohérence

Colonne d'étiquettes alignée à droite

Colonne de contrôles alignée à gauche

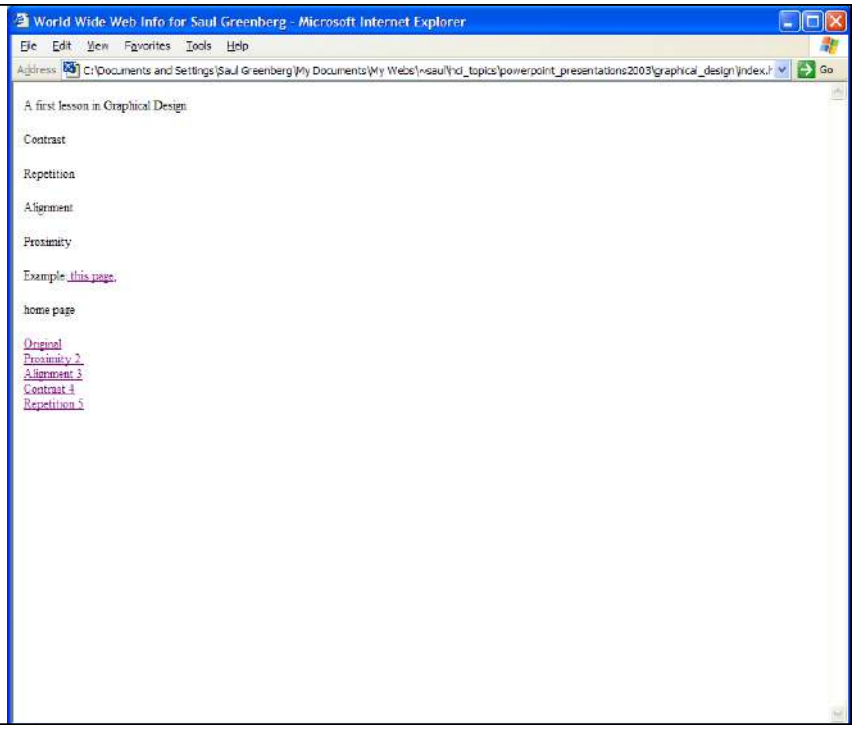


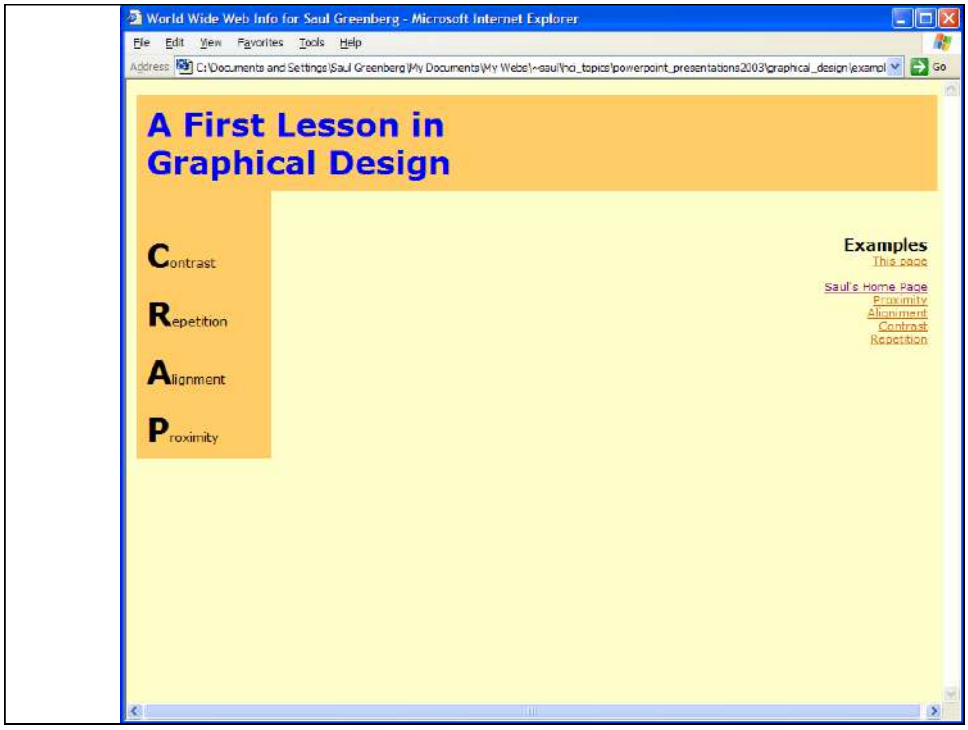
Cohérence entre les contrôles du même type

CRAP

contraste, répétition, alignement, proximité

Major sources: Designing Visual Interfaces, Mullet & Sano, Prentice Hall / Robin Williams Non-Designers Design Book, Peachpit Press




World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address C:\Documents and Settings\Saul Greenberg\My Documents\My Webs\~saul\hci_topics\powerpoint_presentations2003\graphical_design\examp
Go
A First Lesson in Graphical Design
Contrast
Repetition
Alignment
Proximity
Examples
This page
Saul's Home Page
Proximity
Alignment
Contrast
Repetition


Good Design Is As Easy as 1-2-3
1. Learn the principles.
They're simpler than you might think.
2. Recognize when you're not using them.
Put it into words -- name the problem.
3. Apply the principles.
You'll be amazed.
Good design
is as easy as . . .
1 Learn the principles.
They're simpler than you might think.
2 Recognize when you're not using them.
Put it into words — name the problem.
3 Apply the principles.
You'll be amazed.

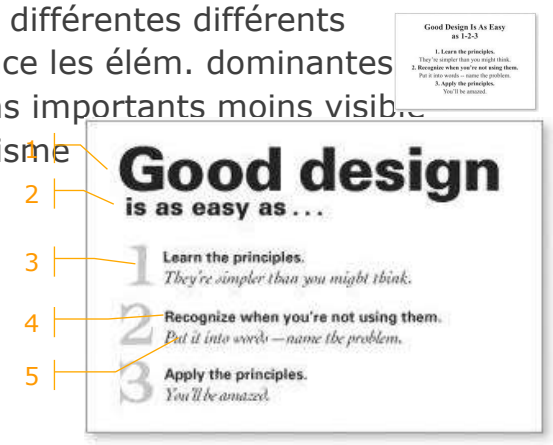
CRAP

- **C**ontraste
- **R**épétition
- **A**lignement
- **P**roximité

Robin Williams Non-Designers Design Book, Peachpit Press

CRAP

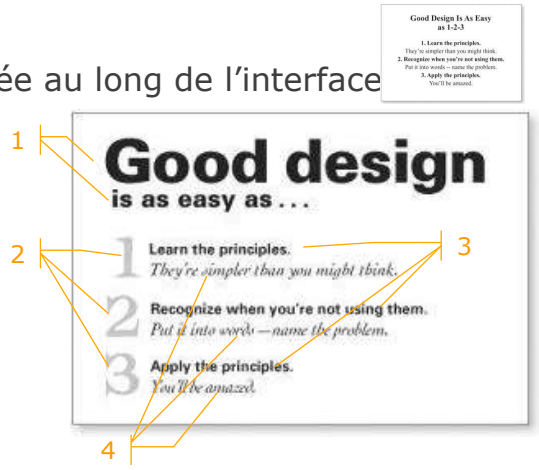
- **Contraste**
  - Faire des choses différentes différents
  - Maitre en évidence les élém. dominantes
  - Faire élém. moins importants moins visible
  - Créer un dynamisme
- **R**épétition
- **A**lignement
- **P**roximité



Robin Williams Non-Designers Design Book, Peachpit Press

CRAP

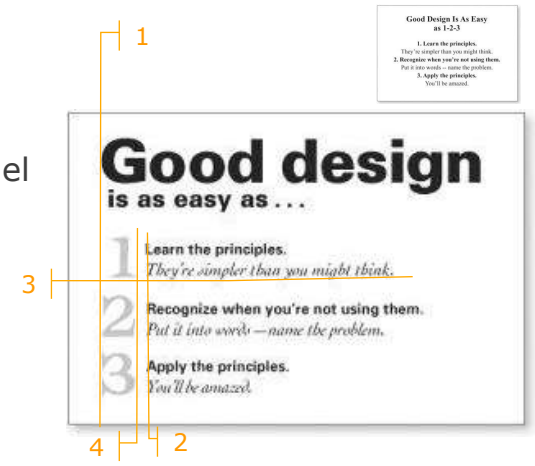
- **C**ontraste
- **Répétition**
  Conception repetée au long de l'interface
  Consistance
  Créer unité
- **A**lignement
- **P**roximité



Robin Williams Non-Designers Design Book, Peachpit Press

CRAP

- **C**ontraste
- **R**épétition
- **A**lignement
  Créer un flux visuel
  Connecter élém.
- **P**roximité

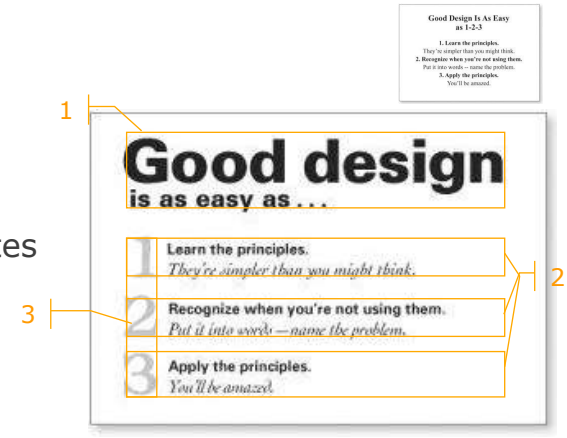


Robin Williams Non-Designers Design Book, Peachpit Press

CRAP

- **C**ontraste
- **R**épétition
- **A**lignement
- **P**roximité

  Groupes évidentes
  Indépendants
  séparées



Robin Williams Non-Designers Design Book, Peachpit Press

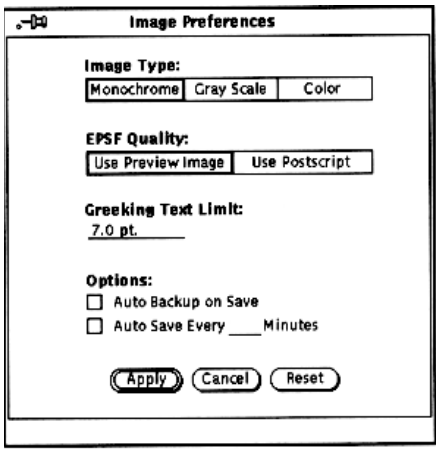
Qu'est-ce que tu vois d'abord?

- CRAP donne des indices sur la façon de lire le graphique

Robin Williams Non-Designers Design Book, Peachpit Press

Qu'est-ce que tu vois d'abord?

- Ici contraste mais proximité pas bien utilisé
  - Structure ambigüe
  - Groupes ambigües

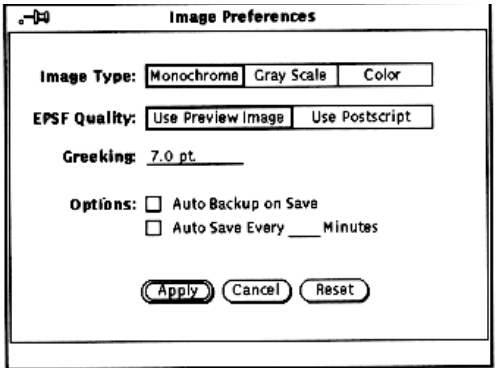


×

Robin Williams Non-Designers Design Book, Peachpit Press

Qu'est-ce que tu vois d'abord?

- Proximité (séparation gauche/droit)
  - Structure claire



✓

Robin Williams Non-Designers Design Book, Peachpit Press

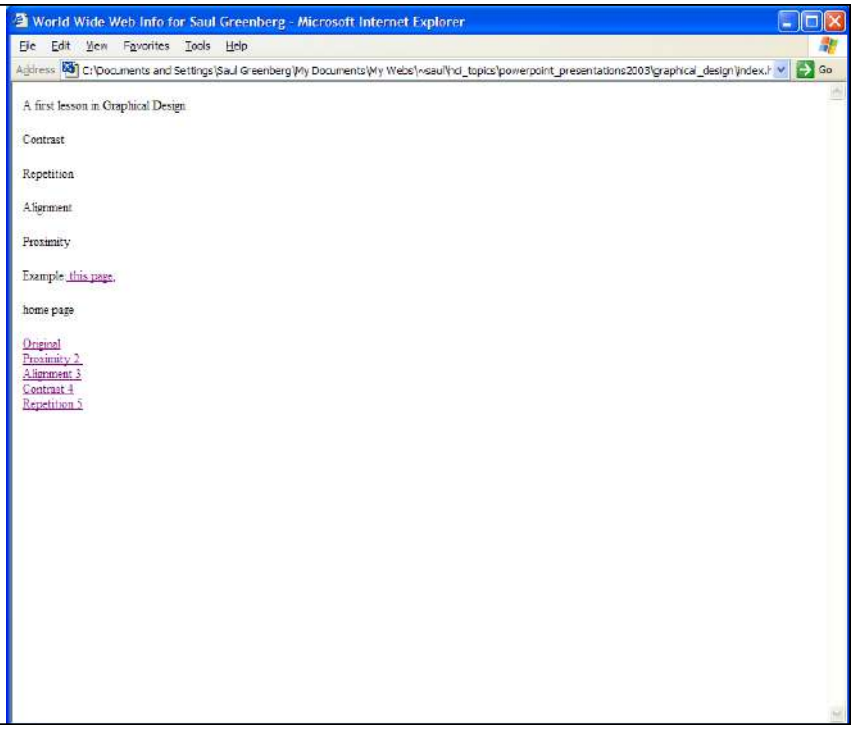
Qu'est-ce que tu vois d'abord?

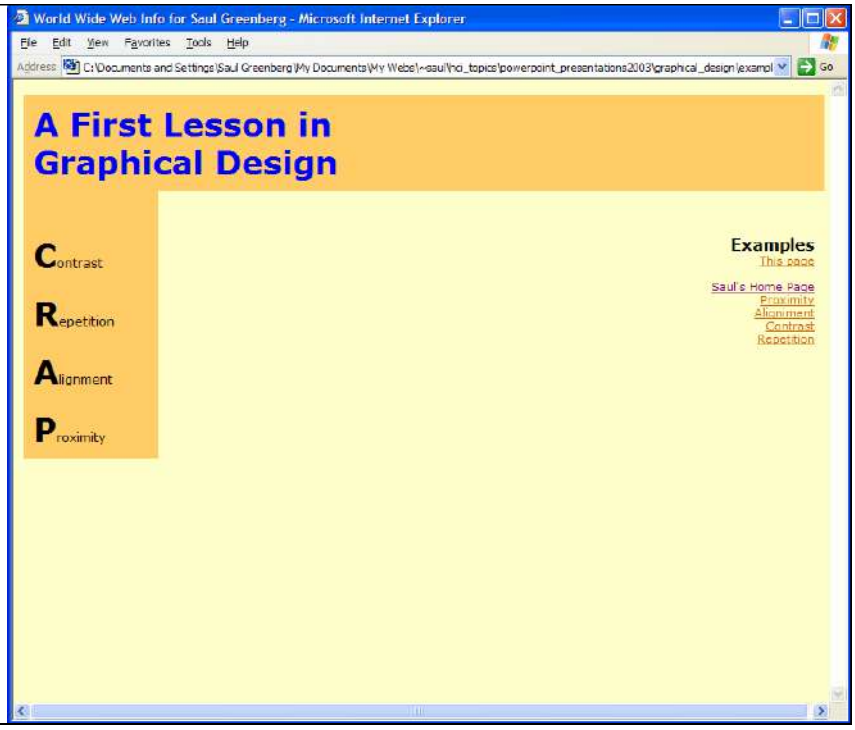
- La puissance de la proximité
  - Alignement
  - Structure explicite peut être améliorée
  - Espace blanc

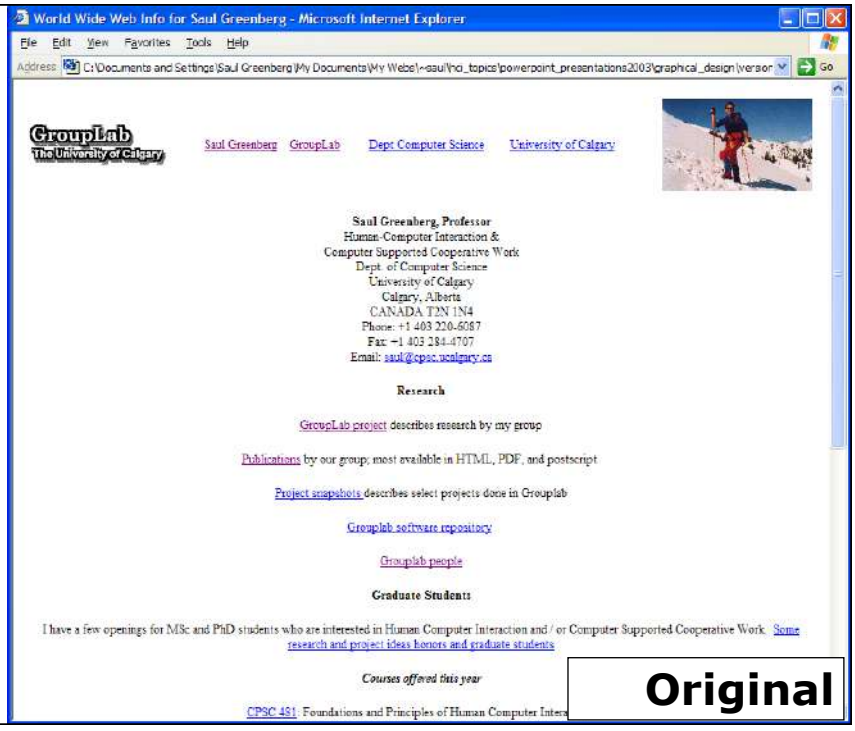
Mmmm:
Mmmm:
Mmmm:
Mmmm:
Mmmm:

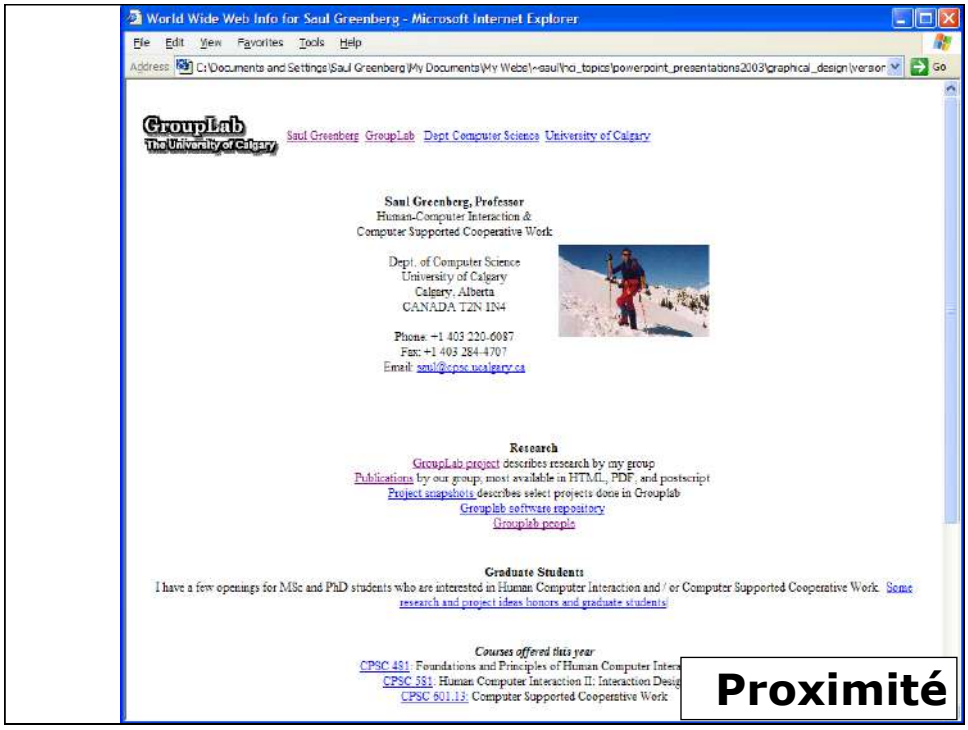
Mmmm:
Mmmm:
Mmmm:
Mmmm:
Mmmm:

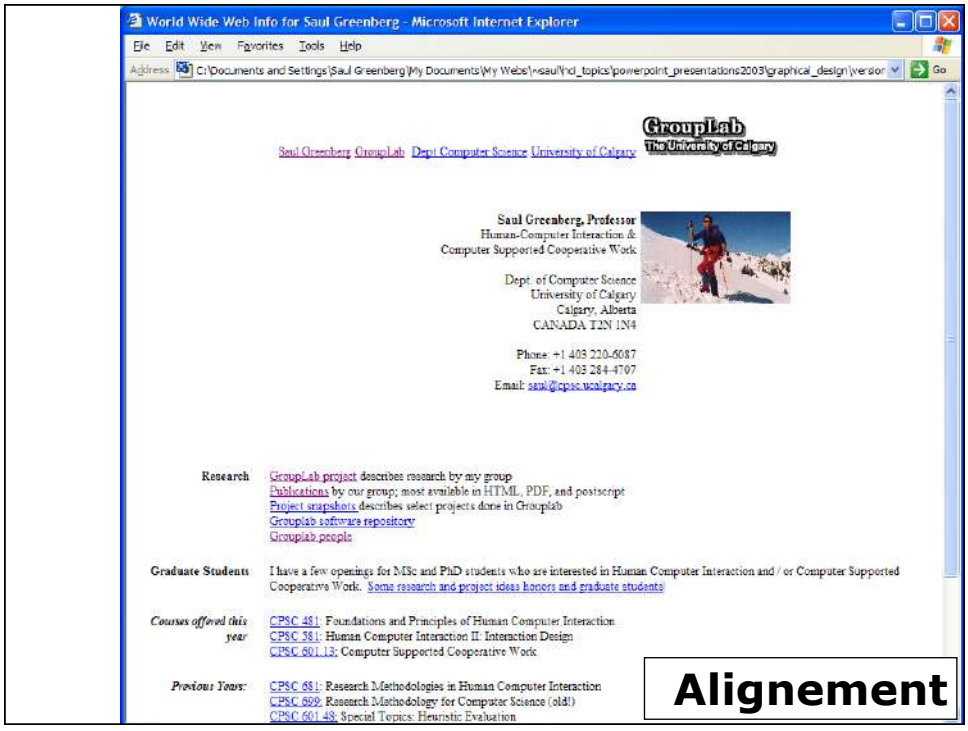
Mmmm:
Mmmm:
Mmmm:
Mmmm:
Mmmm:

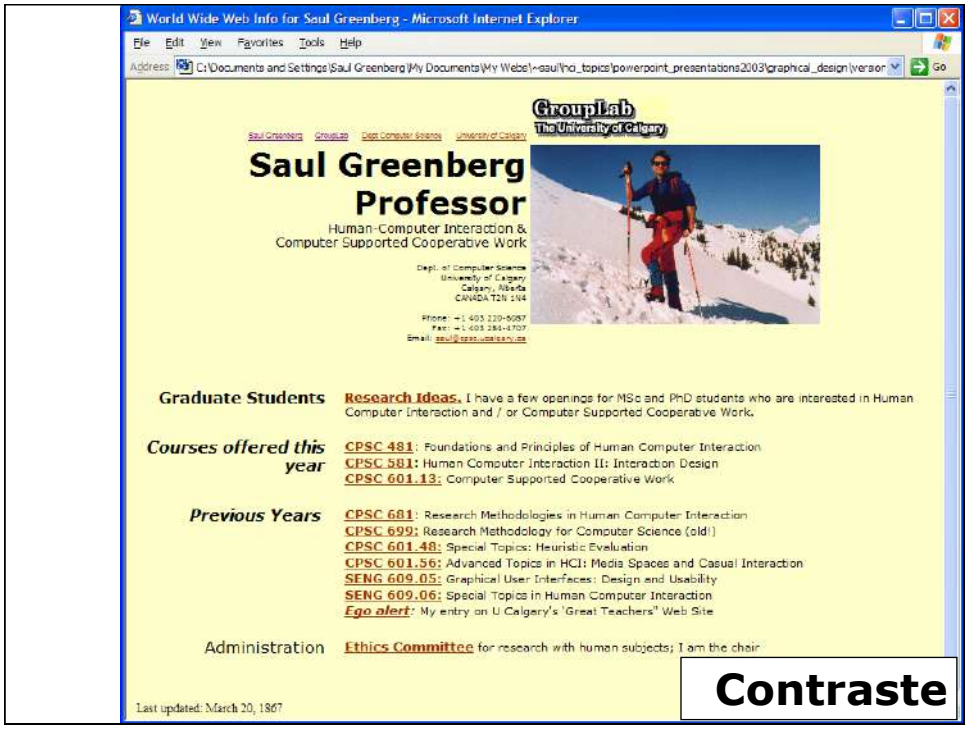


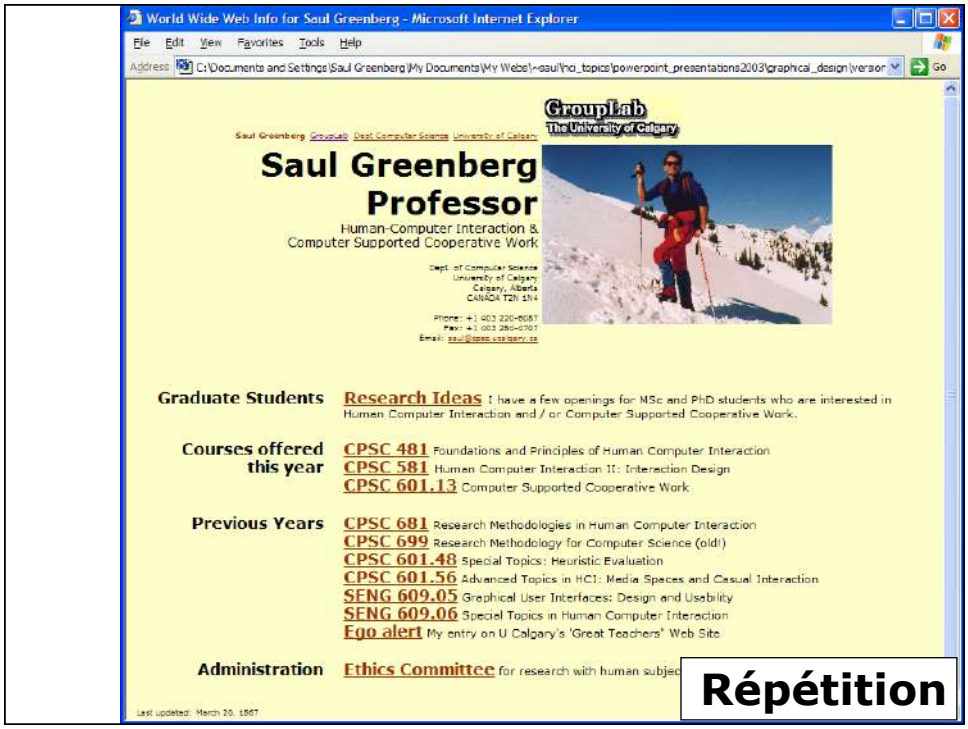

World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
File Edit View Favorites Tools Help
A First Lesson in Graphical Design
Contrast
Repetition
Alignment
Proximity
Examples
This page
Saul's Home Page
Proximity
Alignment
Contrast
Repetition


World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
File Edit View Favorites Tools Help
GroupLab
The University of Calgary
Saul Greenberg GroupLab Dept Computer Science University of Calgary
Saul Greenberg, Professor
Human-Computer Interaction &
Computer Supported Cooperative Work
Dept. of Computer Science
University of Calgary
Calgary, Alberta
CANADA T2N 1N4
Phone: +1 403 220-6087
Fax: +1 403 284-4707
Email: saul@cpsc.ucalgary.ca
Research
GroupLab project describes research by my group
Publications by our group, most available in HTML, PDF, and postscript
Project snapshots describes select projects done in Grouplab
Grouplab software repository
Grouplab people
Graduate Students
I have a few openings for MSc and PhD students who are interested in Human Computer Interaction and / or Computer Supported Cooperative Work. Some research and project ideas honors and graduate students
Courses offered this year
CPSC 481: Foundations and Principles of Human Computer Inter
Original


World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address C:\Documents and Settings\Saul Greenberg\My Documents\My Webs\~saul\hci_topics\powerpoint_presentations2003\graphical_design\version
Go
GroupLab The University of Calgary
Saul Greenberg GroupLab Dept Computer Science University of Calgary
Saul Greenberg, Professor
Human-Computer Interaction &
Computer Supported Cooperative Work
Dept. of Computer Science
University of Calgary
Calgary, Alberta
CANADA T2N 1N4
Phone: +1 403 220-6087
Fax: +1 403 284-4707
Email: saul@cpsc.ucalgary.ca
Research
GroupLab project describes research by my group
Publications by our group, most available in HTML, PDF, and postscript
Project snapshots describes select projects done in Grouplab
Grouplab software repository
Grouplab people
Graduate Students
I have a few openings for MSc and PhD students who are interested in Human Computer Interaction and / or Computer Supported Cooperative Work. Some research and project ideas honors and graduate students
Courses offered this year
CPSC 481: Foundations and Principles of Human Computer Inter
CPSC 581: Human Computer Interaction II: Interaction Desig
CPSC 601.13: Computer Supported Cooperative Work
Proximité

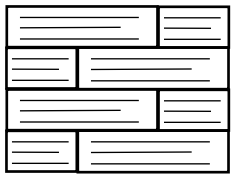

World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address C:\Documents and Settings\Saul Greenberg\My Documents\My Webs\~saul\hci_topics\powerpoint_presentations2003\graphical_design\version
Go
Saul Greenberg GroupLab Dept Computer Science University of Calgary
GroupLab The University of Calgary
Saul Greenberg, Professor
Human-Computer Interaction &
Computer Supported Cooperative Work
Dept. of Computer Science
University of Calgary
Calgary, Alberta
CANADA T2N 1N4
Phone: +1 403 220-6087
Fax: +1 403 284-4707
Email: saul@cpsc.ucalgary.ca
Research
GroupLab project describes research by my group
Publications by our group; most available in HTML, PDF, and postscript
Project snapshots describes select projects done in Grouplab
Grouplab software repository
Grouplab people
Graduate Students
I have a few openings for MSc and PhD students who are interested in Human Computer Interaction and / or Computer Supported Cooperative Work. Some research and project ideas honors and graduate students
Courses offered this year
CPSC 481: Foundations and Principles of Human Computer Interaction
CPSC 581: Human Computer Interaction II: Interaction Design
CPSC 601.13: Computer Supported Cooperative Work
Previous Years:
CPSC 681: Research Methodologies in Human Computer Interaction
CPSC 699: Research Methodology for Computer Science (old!)
CPSC 601.48: Special Topics: Heuristic Evaluation
Alignement

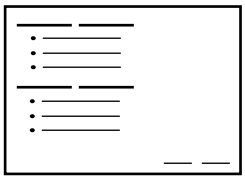

World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
File Edit View Favorites Tools Help
GroupLab
The University of Calgary
Saul Greenberg
Professor
Human-Computer Interaction &
Computer Supported Cooperative Work
Graduate Students
Research Ideas. I have a few openings for MSc and PhD students who are interested in Human Computer Interaction and / or Computer Supported Cooperative Work.
Courses offered this year
CPSC 481: Foundations and Principles of Human Computer Interaction
CPSC 581: Human Computer Interaction II: Interaction Design
CPSC 601.13: Computer Supported Cooperative Work
Previous Years
CPSC 681: Research Methodologies in Human Computer Interaction
CPSC 699: Research Methodology for Computer Science (old!)
CPSC 601.48: Special Topics: Heuristic Evaluation
CPSC 601.56: Advanced Topics in HCI: Media Spaces and Casual Interaction
SENG 609.05: Graphical User Interfaces: Design and Usability
SENG 609.06: Special Topics in Human Computer Interaction
Ego alert: My entry on U Calgary's 'Great Teachers' Web Site
Administration
Ethics Committee for research with human subjects; I am the chair
Last updated: March 20, 1867
Contraste


World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
File Edit View Favorites Tools Help
GroupLab
The University of Calgary
Saul Greenberg
Professor
Human-Computer Interaction &
Computer Supported Cooperative Work
Graduate Students
Research Ideas I have a few openings for MSc and PhD students who are interested in Human Computer Interaction and / or Computer Supported Cooperative Work.
Courses offered this year
CPSC 481 Foundations and Principles of Human Computer Interaction
CPSC 581 Human Computer Interaction II: Interaction Design
CPSC 601.13 Computer Supported Cooperative Work
Previous Years
CPSC 681 Research Methodologies in Human Computer Interaction
CPSC 699 Research Methodology for Computer Science (old!)
CPSC 601.48 Special Topics: Heuristic Evaluation
CPSC 601.56 Advanced Topics in HCI: Media Spaces and Casual Interaction
SENG 609.05 Graphical User Interfaces: Design and Usability
SENG 609.06 Special Topics in Human Computer Interaction
Ego alert My entry on U Calgary's 'Great Teachers' Web Site
Administration
Ethics Committee for research with human subjec
Last updated: March 20, 1867
Répétition

Qu'est-ce que tu vois d'abord?

- Boîtes ne montre pas la structure
  - Utilisez CRAP





Robin Williams Non-Designers Design Book, Peachpit Press