GROUPKIT
Groupware toolKit

Cédric LÉVY-BENCHETON

Plan

1. Fonctionnement de GroupKit
2. Les concepts coopératifs utilisés
3. En pratique
4. Pour conclure

Partie 1

- Fonctionnement de GroupKit

3/29

Introduction

- Logiciel mono-utilisateur existant
- On veut le rendre coopératif
  - C'est le même logiciel
  - Pas envie de programmer la « couche coopérative »

GroupKit !

4/29

GroupKit

- Toolkit écrit en Tcl/Tk
  - Multi-plateforme
  - RPC pour communiquer : totalement transparent
- Sessions et conférences
  - Différents groupes d'utilisateurs et programmes
- Primitives
  - Liaison entre utilisateurs
  - Commencent par `gk_`
- Widgets
  - Visualisation multi-utilisateurs
- Créé en 1992, développé jusqu'en 1998
  - Université de Calgary [Mark Roseman, Saul Greenberg]
- Relancé en 2003 (projet SourceForge)

Les concepts de GroupKit

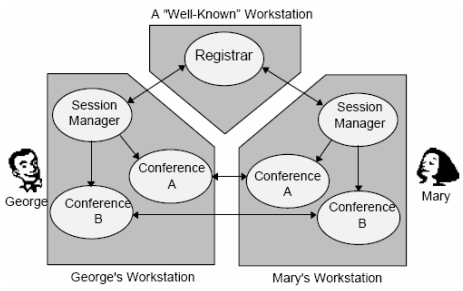
- Registrar
  - Le serveur qui gère une session et ses conférences
  - Connu de tous les participants
- Gestionnaire de session
  - Connexion au registrar
  - Lancement de conférences
- Applications de conférence
  - Chaque application est liée à une conférence
  - Tableau blanc, éditeur de texte, etc…

Environnement partagé

- Modèle
  - Structure de données à afficher
- Vue
  - Représentation des données (affichage à l'écran)
  - Wysiwis
    - Strict ou Relâché (barre de défilement)
- Contrôleur
  - L'utilisateur utilise le contrôleur pour modifier le modèle

Une session GroupKit

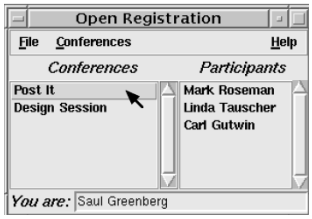
- George et Mary participent
  - A la même session
  - A 2 conférences en même temps



Exemple de session GroupKit – © Roseman, Greenberg

Conférence

- Définition
  - Une application coopérative
  - Et des utilisateurs
- Plusieurs événements associés
  - Rejoindre
  - Quitter
  - Sauvegarder la session
  - Mettre à jour
  - Informer un nouvel arrivant tardif de l'état actuel
- Primitives pour chaque événement



Rejoindre une conférence

Les primitives GroupKit

- Facile et rapide à utiliser
  - Moins d'une journée pour créer une application coopérative
- Accès à la « couche coopérative »
- Gestion et coordination
  - Des utilisateurs
  - De l'environnement

Si j'arrive en retard…

- Récupération de l'état actuel
  - Tout le monde est notifié de mon arrivée
  - Je demande une copie de l'état actuel à un utilisateur, choisi au hasard (primitive `gk_toUserNum`)
  - Il me l'envoie
  - Et c'est tout !
- Mais ça doit être autorisé !
  - Création de `gk_bind updateEntrant { Code à exécuter }`

Partie 2

- Les concepts coopératifs utilisés

Awareness

- Utilisateur identifié par son nom
  - Modification possible dans le fichier de config
  - Pas d'authentification
- Chaque utilisateur possède un numéro d'identifiant
  - Utilisé par GroupKit
- Possibilité d'obtenir des infos sur les utilisateurs
  - Savoir ce qu'il fait en temps réel
  - Connaître l'auteur d'une modification
  - ⇒ Utilisation des widgets !

Gestion du contenu

- Edition du document
  - Chaque utilisateur dispose de la version complète du document en local
  - Mais, pas de droits ! Si un utilisateur peut faire quelque chose, tout le monde peut le faire
- Wysiwis
  - Strict ou relâché ? Décision du programmeur
  - Notion de sélection : simple ou multiple ? Idem
- Contenu lié à un environnement
  - `gk_newenv` *`environnement`*
- Le dernier utilisateur qui quitte décide soit de…
  - Sauver la conférence et son contenu, stocké en mémoire sur le registrar
  - Détruire la conférence et perdre tout le contenu

Coordination

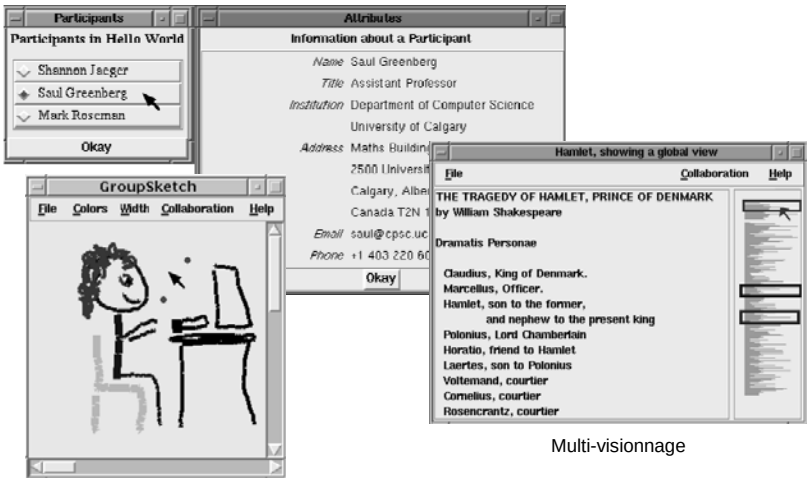
- Mises à jour régulières
- Connexion directe entre les utilisateur
  - Le registrar ajoute l'utilisateur à la session
  - Tout le reste est géré au niveau des clients
- Multicast grâce à RPC !
  - Mais possibilité d'unicast (envoie d'info à l'utilisateur n)
- Plusieurs utilisateurs peuvent créer du contenu en même temps
  - Au programmeur de créer des mécanismes de résolution de conflits
- Définir la granularité
  - Par défaut, au niveau objet. Les changements sont transmis en direct !
  - Sélection possible à partir d'un menu : changement à la volée autorisé

Widgets

- Liés à l'awareness
  - Le client influe sur son propre widget
  - Il est au courant des actions des autres
- Quelques exemples
  - Télé-Pointeur
    - Voir les curseurs des autres en temps-réel
    - Différencier son curseur
  - Barre de défilement multi-utilisateurs
    - Lecture ou édition simultanée d'un document

Quelques Widgets

Informations sur les participants



Télépointeurs

Multi-visionnage

Principes coopératifs utilisés

- Wysiwis
  - Strict ou relâché
- Awareness
  - Qui fait quoi ? Rôle des Widgets !
- Granularité
  - Choisir le niveau de partage
- Gestion des participants
  - Qui parle ? Tout le monde en même temps !
  - Modèle Optimiste
- GroupKit et le cercle 3C
  - Coopération
  - Communication
  - Coordination

Partie 3

- En pratique

19/29

En pratique

- Téléchargement de Tcl/Tk et GroupKit
  - Active TCL pour Windows
  - GroupKit (groupkit.org ou sourceforge)
- Installation rapide, moins de 5 minutes
- Démarrage du registrar
- Connexion des clients au registrar
  - Créer ou rejoindre une conférence
  - Fichier de préférences `.grouptkitrc`

20/29

Créer une application GroupKit

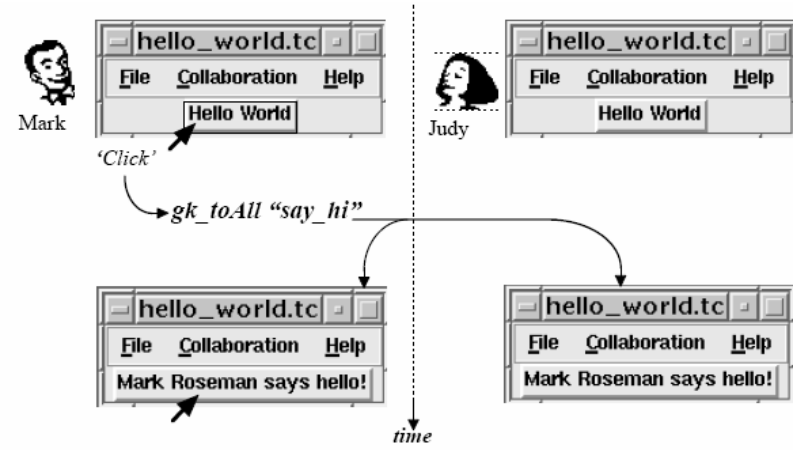
- 1ère ligne du programme
  - `gk_initConf $argv`
- Ajout du menu GroupKit
  - `gk_defaultMenu .menubar`
  - `pack .menubar -side top -fill x`
- Utilisation des primitives
  - `gk_toAll, gk_toOthers, gk_toUsernum`
- Enregistrement de *MonProgramme* sur le Registrar
  - `userprefs prog.MonProgramme.cmd “exec gkwish -f /home/moi/monprog.tcl”`

Modifier une application existante

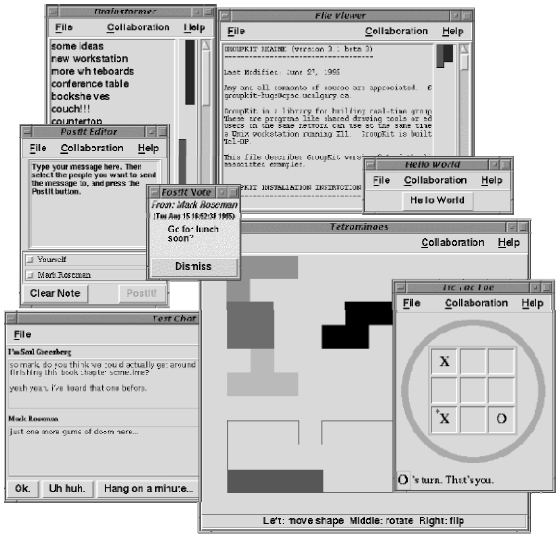
- Hello World
  - Ce qui est en gras rend l’application coopérative

```
1 gk_initConf $argv
2 gk_defaultMenu .menubar
3 pack .menubar -side top -fill x
4 set greetings “[users local.username] says hello!”
5 button .hello -text “Hello World” \
      -command "gk_toAll say_hi \”$greetings\”""
6 pack .hello -side top
7 proc say_hi {new_label} {
8      .hello configure -text $new_label
9      after 2000 {.hello configure -text “Hello World”}
10 }
```

Hello World multi-utilisateurs



D'autres exemples



Encore plus loin avec GroupKit

- Différents gestionnaires de session
  - Représentation du monde réel (salle de réunion, etc…)
  - Gestion des droits d'accès à une application
    - Mais l'utilisateur a tous les droits dans l'application
- Il est aussi possible de
  - Créer de nouveaux widgets
  - Programmer ses propres événements
  - Relier différentes applications entre elles

25/29

Contraintes

- Applications peu attractives
  - Qui programme en Tcl/Tk ?
- Utilisation restreinte
  - Pas d'authentification, pas de gestion des droits : groupe de confiance.
  - Tout le monde doit utiliser GroupKit
- Pas un media-space
  - Impossible de créer des liens entre différentes applications
- Pas de gestion des conflits
  - A l'utilisateur de faire « comme dans la vie réelle »
- Impossible d'utiliser un document créé dans GroupKit hors-ligne
  - Utilisation synchrone (temps réel) uniquement
  - Impossible de sauvegarder le fichier chez soi

26/29

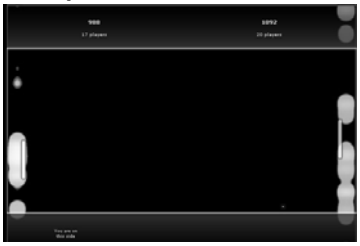
Partie 4

- Pour conclure

27/29

Le futur ?

- Applications coopératives liées au web
  - Utilisation de framework (JAVA, .NET, AJAX)
  - Coopératisation des applications existantes (MS Office avec Sharepoint par exemple)
- De nouveaux domaines coopératif
  - Jeux vidéo
    - Pong coopératif



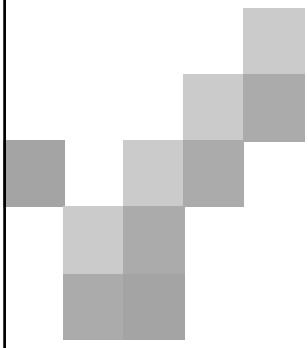
28/29

Conclusion

- GroupKit pose les bases de la couche coopérative
- Puissant et rapide à mettre en œuvre
  - Peu de lignes de code
  - Modification d'un programme mono-utilisateur grâce aux primitives
- Idées innovantes (widgets)
- Très utile pour faire du travail coopératif rapidement

Mais

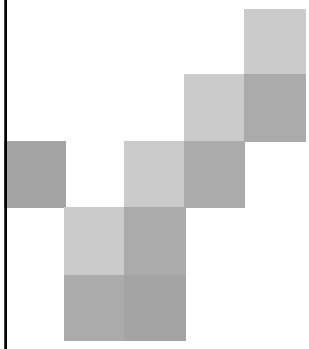
- Trop de contraintes pour une utilisation grand public



Merci de votre attention

GroupKit
Groupware toolKit

Cédric LÉVY-BENCHETON



Discussion

GroupKit
Groupware toolKit

Cédric LÉVY-BENCHETON