AppKit

Bibliothèque de développement d'interfaces graphiques

Anass BOUASSABA

Généralités

- Bibliothèque de développement d'interfaces graphiques pour des applications de bureau
- Ecrite en Langage C++
- Design et mécanismes orientés objet
- Multiplateforme (Windows, Linux, Mac OS X)
- Actuellement pas de version pour Mobile
- Distribuée sous licence " Creative Commons Attribution-NoDerivs 3.0 "

Historique

- Crée par en 2008 sous forme d'une bibliothèque d'interfaces en Langage C sous le nom «MDUI» utilisent le modèle procédurale style Windows API

- Redéveloppé en décembre 2011 en utilisent un modèle orienté objet sous le nom «AppKit»

Introduction à la programmation AppKit

Comment développer une application AppKit ?

La classe akApplication

CRÉER UNE APPLICATION

La classe **akApplication**

- Créer une application et démarrer la boucle principale

```
#include <AppKit.h>

int main(int argc, char *argv[])
{
        akApplication *app = new akApplication("Mon application");
        return app->Run();
}
```

La classe akApplication

```
class  akApplication : public akObject
{
public:
        akApplication(const string &name);
        akApplication(akSize size, const string &name);
        int Run();
private:
        void DispatchEvent(akInputEvent *evt);
};
```

La classe akWindow

CRÉER UNE FENÊTRE

La classe **akWindow**

- Créer une fenêtre
- Modifier les styles
- Ajouter des contrôles

```
#include <AppKit.h>

int main(int argc, char *argv[])
{
        akApplication *app = new akApplication("Mon application");

        akWindow *fenetre = new akWindow(akRect(50,50,400,300),
                                         "Mon application",
                                         akWS_CLOSABLE|akWS_VISIBLE);

        return app->Run();
}
```

Les styles de fenêtre

- **akWS_VISIBLE** : permet à la fenêtre d'être visible dès sa création
- **akWS_CLOSABLE** : créer un bouton de fermeture
- **akWS_MAXIMIZABLE** : créer un bouton d'agrandissement de la fenêtre
- **akWS_MINIMIZABLE** : créer un bouton pour réduire la fenêtre

La classe akWindow

```
class akWindow : public akObject
{
public:
    akWindow(akRect contentRect, string title, long style);
    string GetTitle();
    akRect GetRect();
    akRect GetContentRect();
    void SetRect(akRect rect);
    void SetContentRect(akRect rect);
    bool IsVisible();
    void SetVisible(bool visible);
    void SetMaximizable(bool maximizable);
    void SetMinimizable(bool minimizable);
    void SetClosable(bool closable);
    bool IsMaximizable();
    bool IsMinimizable();
    bool IsClosable();
```

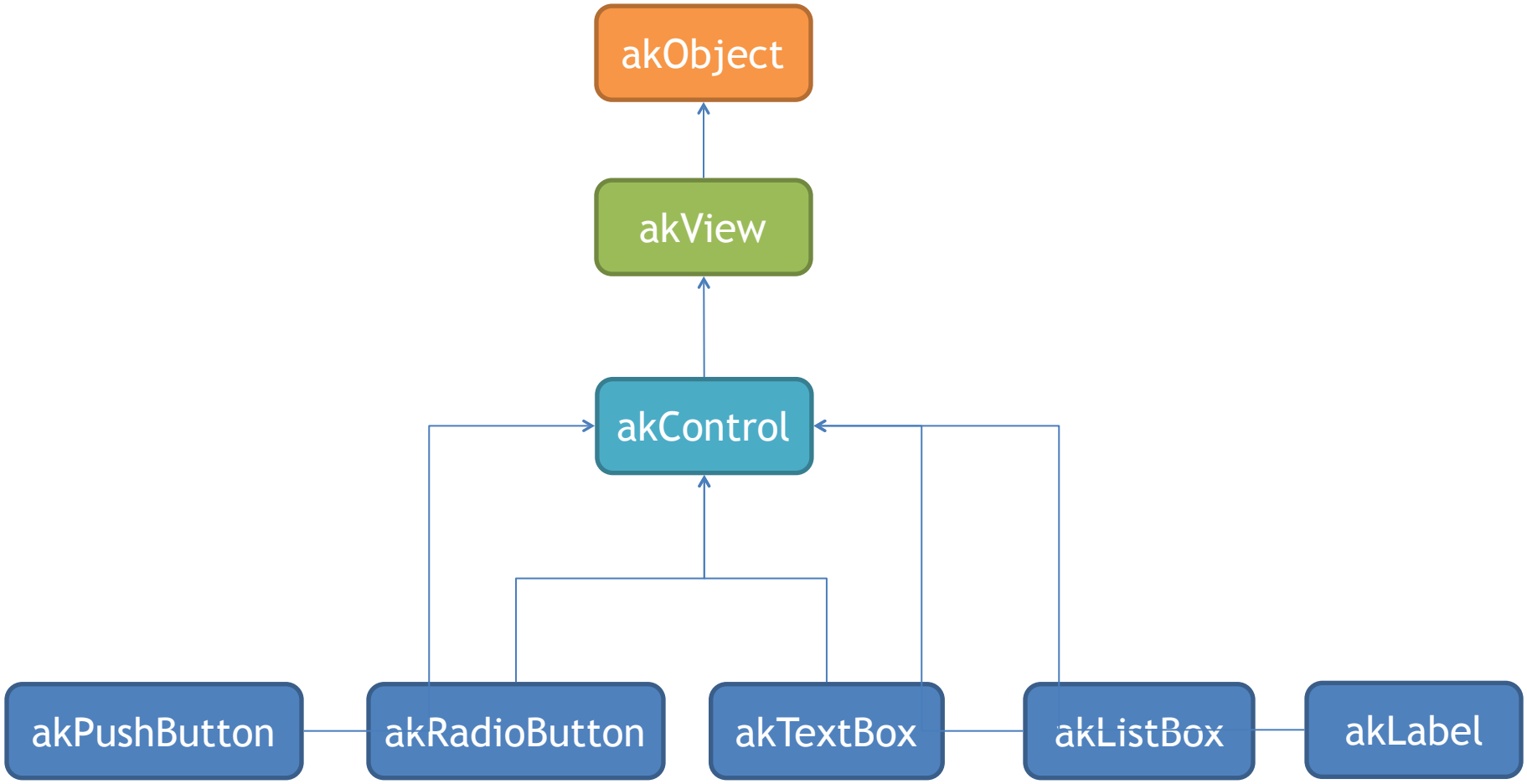
La classe akWindow

```
void SetBackgroundColor(akColor color);
akColor GetBackgroundColor();
void Repaint();
void AddView(akView *view);
void RemoveView(akView *view);
akView* GetFirstResponder();
void SetFirstResponder(akView *view);
void DispatchInputEvent(akInputEvent *evt);
void SetCanReceiveMouseMoveEvents(bool receive);
bool CanReceiveMouseMotionEvents();
void Close();
```

La class akControl

LES CONTRÔLES

Hiérarchie


akObject
akView
akControl
akPushButton
akRadioButton
akTextBox
akListBox
akLabel

Les contrôles

- Un contrôle est une objet de type **akControl**
- La class **akControl** hérite de la classe **akView**
- Donc chaque contrôle est une vue (akView)
- Les vues peuvent être ajouté à une fenêtre (akWindow) pour être visible à l'utilisateur
- Tous les objets de **AppKit** héritent de la classe **akObject**

La classe **akObject**

```
class akObject
{
public:
        akObject();
        void SetClassName(string className);
        string GetClassName();
};
```

La class akView

```
class akView : public akObject
{
public:
        akView();
        akView(akRect rect, akView *parent = NULL);
        void SetRect(akRect rect);
        akRect GetRect();
        void AddChild(akView *view);
        akView* RemoveChild(akView *view);
        void Repaint();
        akWindow* GetWindow();
        void SetWindow(akWindow *wnd);
        void AddPainter(akPainter *painter);
        void RemovePainter(akPainter *painter);
        void AddKeyEventReceiver(akKeyEventReceiver *receiver);
        void RemoveKeyEventReceiver(akKeyEventReceiver *receiver);
        void AddMouseEventReceiver(akMouseEventReceiver *receiver);
        void RemoveMouseEventReceiver(akMouseEventReceiver *receiver);
        void AddViewNotificationReceiver(akViewNotificationReceiver *receiver);
        void RemoveViewNotificationReceiver(akViewNotificationReceiver *rcv);
};
```

La classe **akControl**

```
class akControl : public akView
{
public:
        akControl(akRect rect, akView *parent = NULL);
        void AddActionReceiver(akActionReceiver *receiver);
        void RemoveActionReceiver(akActionReceiver *receiver);
};
```

La class akPushButton

CRÉER UN BOUTON

La class akPushButton

```
#include <AppKit.h>

int main(int argc, char *argv[])
{
        akApplication *app = new akApplication("Mon application");

        akWindow *fenetre = new akWindow(akRect(50,50,400,300),
                                         "Mon application",
                                         akWS_CLOSABLE|akWS_VISIBLE);

        akPushButton *bouton = new akPushButton(akRect(10,10,80,35),"Bouton 1");
        fenetre->AddView(bouton);


        return app->Run();
}
```

La classe akPushButton

```
class akPushButton : public akControl
{
public:
        akPushButton(akRect rect, string text,
                          akView *parent = NULL);
        string GetText();
        void SetText(string text);
}
```

La class akRadioButton

CRÉER UN BOUTON RADIO

La class akRadioButton

```
#include <AppKit.h>

int main(int argc, char *argv[])
{
        akApplication *app = new akApplication("Mon application");

        akWindow *fenetre = new akWindow(akRect(50,50,400,300),
                                         "Mon application",
                                         akWS_CLOSABLE|akWS_VISIBLE);


        akRadioButton *boutonRadio = new akRadioButton(akRect(10,10,80,35),
                                                       "Bouton Radio 1");
        fenetre->AddView(boutonRadio);


        return app->Run();
}
```

La class akRadioButton

```
class akRadioButton : public akControl
{
public:
     akRadioButton(akRect rect, string text, akView *parent = NULL);
     void SetText(string text);
     string GetText();
     void SetSelected(bool selected);
     bool IsSelected();
     void SetGroup(akRadioButtonGroup *group);

     };
```

La class akTextBox

CRÉER UNE ZONE TO TEXTE

La classe akTextBox

```
#include <AppKit.h>

int main(int argc, char *argv[])
{
        akApplication *app = new akApplication("Mon application");

        akWindow *fenetre = new akWindow(akRect(50,50,400,300),
                                         "Mon application",
                                         akWS_CLOSABLE|akWS_VISIBLE);


        akTextBox *zoneDeTexte = new akTextBox(akRect(10,10,80,35),"Texte");
        fenetre->AddView(zoneDeTexte);

        string zoneDeTexte->GetText();

        string autreTexte = "Un autre text";
        zoneDeTexte->SetText(autreTexte);


        return app->Run();
}
```

La classe akTextBox

```
class akTextBox : public akControl
{
public:
        akTextBox(akRect rect, string text, akView *parent = NULL);
        string GetText();
        void SetText(string text);
};
```

La classe akListBox

CRÉER UNE LISTE

La class akListBox

```
#include <AppKit.h>

int main(int argc, char *argv[])
{
        akApplication *app = new akApplication("Mon application");

        akWindow *fenetre = new akWindow(akRect(50,50,400,300),
                                         "Mon application",
                                         akWS_CLOSABLE|akWS_VISIBLE);


        akListBox *listbox = new akListBox(akRect(10,10,150,200));
        fenetre->AddView(listbox);

        listbox->AddItem("Windows");
        listbox->AddItem("Mac");
        listbox->AddItem("Linux");

        string listbox->GetItemText(1);
        listbox->SetItemText(1,"Android");
        int index =  listbox->GetSelectionIndex();
        int nombreElements = listbox->GetItemCount();

        return app->Run();
}
```

La class akListBox

```
class akListBox : public akControl
{
public:
     akListBox(akRect rect, akView *parent = NULL);
     void AddItem(string text);
     void RemoveItem(int index);
     string GetItemText(int index);
     void SetItemText(int index, string text);
     int GetSelectionIndex();
     int GetItemCount();

     };
```

La class akLabel

CRÉER UN LABEL

La classe akLabel

```
#include <AppKit.h>

int main(int argc, char *argv[])
{
        akApplication *app = new akApplication("Mon application");

        akWindow *fenetre = new akWindow(akRect(50,50,400,300),
                                         "Mon application",
                                         akWS_CLOSABLE|akWS_VISIBLE);


        akLabel *label = new akLabel(akRect(20,20,120,25), "Texte du label");
        fenetre->AddView(label);

        return app->Run();
}
```

La classe akLabel

```
class akLabel : public akControl
{
public:
        akLabel(akRect rect, string text, akView *parent = NULL);
        void SetText(string text);
        string GetText();
};
```

La class akMessageBox

AFFICHER UN MESSAGE

La classe akMessageBox

```
#include <AppKit.h>

int main(int argc, char *argv[])
{
        akApplication *app = new akApplication("Mon application");

        akWindow *fenetre = new akWindow(akRect(50,50,400,300),
                                         "Mon application",
                                         akWS_CLOSABLE|akWS_VISIBLE);


        akMessageBox *msg = new akMessageBox("Un titre",
                                             "Un message",
                                             NULL);


        return app->Run();
}
```

Les Receveurs

GESTION DES ÉVÈNEMENTS

Receveurs

- Un receveur est une interface C++ à implémenter
- Ensuite la class implémenté pourra être ajouter à un contrôle dont on veux gérer les évènements
- Il y a trois types de receveurs : Clavier, Souris, Actions utilisateur (ex: clic sur un bouton)

L’interface akKeyEventReceiver

ÉVÈNEMENTS CLAVIER

L'Interface akKeyEventReceiver

```
class akKeyEventReceiver
{
public:
    virtual void KeyPress(akView* sender, akKeyEvent *event) = 0;
    virtual void KeyRelease(akView* sender, akKeyEvent *event) = 0;
};
```

L'Interface akKeyEventReceiver

```
#include <AppKit.h>


class MonReceveurClavier : public akKeyEventReceiver
{
        virtual void KeyPress(akView* sender, akKeyEvent *event) {
        }

        virtual void KeyRelease(akView* sender, akKeyEvent *event) {
        }
}

int main(int argc, char *argv[])
{
        akApplication *app = new akApplication("Mon application");

        akWindow *fenetre = new akWindow(akRect(50,50,400,300),
                                                "Mon application",
                                                akWS_CLOSABLE|akWS_VISIBLE);


        akView *view1 = new akView(akRect(10,10,150,200));
        fenetre->AddView(view1);
        view1->AddKeyEventReceiver(new MonReceveurClavier());

        return app->Run();
}
```

L’interface akMouseEventReceiver

ÉVÈNEMENTS SOURIS

L'Interface akMouseEventReceiver

```
class akMouseEventReceiver
{
public:
        virtual void MousePress(akView* sender, akMouseEvent *event) = 0;
        virtual void MouseRelease(akView* sender, akMouseEvent *event) = 0;
        virtual void MouseMove(akView* sender, akMouseEvent *event) = 0;
        virtual void MouseDrag(akView* sender, akMouseEvent *event) = 0;
        virtual void MouseWheelUp(akView *sender, akMouseEvent *event) = 0;
        virtual void MouseWheelDown(akView *sender, akMouseEvent *event) = 0;
};
```

L'Interface akMouseEventReceiver

```
#include <AppKit.h>


class MonReceveurSouris : public akMouseEventReceiver
{
        virtual void MousePress(akView* sender, akMouseEvent *event) {}
        virtual void MouseRelease(akView* sender, akMouseEvent *event) {}
        virtual void MouseMove(akView* sender, akMouseEvent *event) {}
        virtual void MouseDrag(akView* sender, akMouseEvent *event) {}
        virtual void MouseWheelUp(akView *sender, akMouseEvent *event) {}
        virtual void MouseWheelDown(akView *sender, akMouseEvent *event) {}
}

int main(int argc, char *argv[])
{
        akApplication *app = new akApplication("Mon application");

        akWindow *fenetre = new akWindow(akRect(50,50,400,300),
                                         "Mon application",
                                         akWS_CLOSABLE|akWS_VISIBLE);


        akView *view1 = new akView(akRect(10,10,150,200));
        fenetre->AddView(view1);
        view1->AddMouseEventReceiver(new MonReceveurSouris());


        return app->Run();
}
```

L’Interface akActionReceiver

LES ÉVÈNEMENTS UTILISATEUR OU ACTIONS

l'Interface **akActionReceiver**

```
class akActionReceiver
{
public:
        virtual void ActionPerformed(akControl *sender) = 0;
};
```

L'Interface akActionReceiver

```
#include <AppKit.h>


class MonReceveurClickBouton : public akActionReceiver
{
        virtual void ActionPerformed(akControl *sender) {
                new akMessageBox("Mon application", "Mon application", NULL);
        }
}

int main(int argc, char *argv[])
{
        akApplication *app = new akApplication("Mon application");

        akWindow *fenetre = new akWindow(akRect(50,50,400,300),
                                                "Mon application",
                                                akWS_CLOSABLE|akWS_VISIBLE);

        akPushButton *bouton = new akPushButton(akRect(10,10,80,35),"Bouton 1");
        fenetre->AddView(bouton);
        bouton ->AddActionReceiver(new MonReceveurClickBouton());

        return app->Run();
}
```

L’interface akPainter

FAIRE DU DESSIN

L'Interface **akPainter**

```
class akPainter
{
public:
        virtual void Paint(akView *view, SDL_Surface *destination) = 0;
};
```

L'Interface **akPainter**

```
#include <AppKit.h>


class MonPaintre : public akPainter
{
        virtual void Paint(akView *view, SDL_Surface *destination)
{
                // Dessiner un rectangle
                SDL_FillRect(destination, NULL, SDL_MapRGB(destination->format,0.255,128));
        }
};

int main(int argc, char *argv[])
{
        akApplication *app = new akApplication("Mon application");

        akWindow *fenetre = new akWindow(akRect(50,50,400,300),
                                                "Mon application",
                                                akWS_CLOSABLE|akWS_VISIBLE);


        akView *view1 = new akView(akRect(10,10,150,200));
        fenetre->AddView(view1);
        view1->AddPainter(new MonPaintre());


        return app->Run();
}
```

Exemple d’utilisation: zone de texte, déplacement du focus

LA NOTION DE «FIRST RESPONDER»

La notion de «**First Responder**»

- Le «First Responder» et le premier View/ Contrôle qui reçoit l'évènement clavier
- Un contôle qui a le focus est un «First Reponder»
- On peut mettre le focus sur une zone de texte on appelant la méthode `SetFirstResponder(akView *view)` de la class akView

Créer une application simple avec AppKit

DEMO