Introduction to the Apple IIGS Toolbox

KansasFest 2009

Getting Started

Getting Started

- A quick look at the very basics of Toolbox programming.

Getting Started

- A quick look at the very basics of Toolbox programming.
- We'll be using ORCA/C for the examples.

About the Toolbox

About the Toolbox

- Divided into multiple tool sets

About the Toolbox

- Divided into multiple tool sets
- Each tool set offers routines for a specific functional area

About the Toolbox

- Divided into multiple tool sets
- Each tool set offers routines for a specific functional area
- Except the Miscellaneous Tool Set

Tenets of the Toolbox

Tenets of the Toolbox

- If you’re not sure it’s already in memory, load it.

Tenets of the Toolbox

- If you’re not sure it’s already in memory, load it.
- When you need it, start it up.

Tenets of the Toolbox

- If you’re not sure it’s already in memory, load it.
- When you need it, start it up.
- When you’re done with it, shut it down.

Tenets of the Toolbox

- If you’re not sure it’s already in memory, load it.
- When you need it, start it up.
- When you’re done with it, shut it down.
- If you loaded it, unload it before your program ends.

The Big Three

- Tool Locator
- Memory Manager
- Miscellaneous Tool Set

The Desktop Tools

- QuickDraw II
- QuickDraw II Auxiliary
- Event Manager
- Desk Manager
- Window Manager
- Control Manager
- Menu Manager
- LineEdit Tool Set
- Font Manager
- Scrap Manager
- Standard File Operations
- Dialog Manager
- List Manager
- Print Manager
- TextEdit Tool Set
- Resource Manager

Universal Calls

Universal Calls

- Every tool set has these:

Universal Calls

- Every tool set has these:
    - BootInit

Universal Calls

- Every tool set has these:
  - BootInit
  - Reset

Universal Calls

- Every tool set has these:
  - BootInit
  - Reset
  - StartUp

Universal Calls

- Every tool set has these:
  - BootInit
  - Reset
  - StartUp
  - ShutDown

Universal Calls

- Every tool set has these:
  - BootInit
  - Reset
  - StartUp
  - ShutDown
  - Version

Universal Calls

- Every tool set has these:
  - BootInit
  - Reset
  - StartUp
  - ShutDown
  - Version
  - Status

Loading Tools (1.0)

Loading Tools (1.0)

```
ToolTable toolList = {
    14,
    { 4, 0x0200 },
    { 5, 0x0100 },
    { 14, 0x0200 },
    { 15, 0x0200 },
    { 16, 0x0200 },
    { 18, 0x0100 },
    { 19, 0x0100 },
    { 20, 0x0100 },
    { 21, 0x0100 },
    { 22, 0x0100 },
    { 23, 0x0100 },
    { 27, 0x0100 },
    { 28, 0x0100 }
};

Word userID;
Handle directPageHndl;
void *dpPtr;
```

```
TLStartUp();
userID = MMStartUp();
MTStartUp();
LoadTools(toolList);

if (toolerror()) {
    exit(1);
}


directPageHndl = NewHandle(
                   userID,
                   attrBank +
                   atttrPage +
                   attrFixed +
                   attrLocked,
                   0);
if (toolerror()) {
    exit(1);
}
```

Loading Tools (1.0)

```
ToolTable toolList = {
    14,
    { 4, 0x0200 },
    { 5, 0x0100 },
    { 14, 0x0200 },
    { 15, 0x0200 },
    { 16, 0x0200 },
    { 18, 0x0100 },
    { 19, 0x0100 },
    { 20, 0x0100 },
    { 21, 0x0100 },
    { 22, 0x0100 },
    { 23, 0x0100 },
    { 27, 0x0100 },
    { 28, 0x0100 }
};

Word userID;
Handle directPageHndl;
void *dpPtr;
```

```
TLStartUp();
userID = MMStartUp();
MTStartUp();
LoadTools(toolList);

if (toolerror()) {
    exit(1);
}


directPageHndl = NewHandle(
                    userID,
                    attrBank +
                    atttrPage +
                    attrFixed +
                    attrLocked,
                    0);
if (toolerror()) {
    exit(1);
}
```

Loading Tools (2.0)

- Create a start/stop record
- Start the Tool Locator, Memory Manager, and Tool Locator
- Call StartUpTools

Loading Tools (2.0)

```
resource rToolStartup(resToolList) {
    $C080,
    {
       4, $0307,
       5, $0304,
       6, $0301,
       7, $0300,
       11, $0300,
       14, $0303,
       15, $0303,
       16, $0303,
       18, $0304,
       20, $0303,
       21, $0304,
       22, $0301,
       23, $0303,
       27, $0303,
       28, $0303,
       34, $0103
    };
```

Loading Tools (2.0)

```
Word userID;

unsigned long startStopRefRet;

TLStartUp();
userID = MMStartUp();
MTStartUp();

startStopRecRet = StartUpTools(userID, refIsResource, resToolList);
if (toolerror()) {
      exit(1);
}
```

Loading Tools by Cheating

```
startdesk(640);
```

Shutting Down Tools

```
ShutDownTools(refIsHandle, startStopRefRet);
MTShutDown();
MMShutDown(userID);
TLShutDown();

/* or */

enddesk();
```

Creating a Menu Bar

- Design the resources
- Write the code

Designing a Menu Bar

- Easy (but tedious) using resources
- Don’t forget to include the standard items

The rMenuBar

```
resource rMenuBar(resMenuBar) {
    {
       resAppleMenu,
       resFileMenu,
       resEditMenu
    };
};
```

The rMenu

```
resource rMenu(resFileMenu) {
    resFileMenu,
    ItemRefShift*refIsResource + MenuTitleRefShift * refIsResource
          + fAllowCache,
    resFileMenu,
    {
       resItemFileOpen,
       resItemFileClose,
       resItemFileQuit
    };
};

resource rPString(resFileMenu) {
    “  File  “
};
```

The rMenuItem

```
resource rMenuItem(resItemFileClose) {
     resItemFileClose,
     “W”
     “w”,
     0x0,
     ItemTitleRefShift*refIsResource + fDivider,
     resItemFileClose
};

resource rPString(resItemFileClose) {
     “Close“
};
```

The Event Loop

The Event Loop

1. Ask the Toolbox, “Did anything happen?”

The Event Loop

1. Ask the Toolbox, “Did anything happen?”
2. If so, do something about it.

The Event Loop

1. Ask the Toolbox, “Did anything happen?”
2. If so, do something about it.
3. Is it time to quit?

The Event Loop

1. Ask the Toolbox, “Did anything happen?”
2. If so, do something about it.
3. Is it time to quit?
4. If not, go to step 1.

The Event Loop

```
int quitFlag;
EventRecord lastEvent;

void EventLoop(void) {
    Word event;

    quitFlag = 0;
    lastEvent.wmTaskMask = 0x1FFBFFL;

    do {
        event = TaskMaster(everyEvent, &lastEvent);

        switch(event) {
            case wInSpecial:
            case wInMenuBar:
                HandleMenu((Word) lastEvent.wmTaskData);
                break;
        }
    } while (!quitFlag);
}
```

Handling Menus

```
void HandleMenu(Word itemID) {
    switch(itemID) {
        case resItemFileQuit:
            quitFlag = 1;
            break;
        case resItemFileClose:
             if (FrontWindow() == theWindow) {
                DoClose();
             }
             break;
        case resItemAppleAbout:
            DoAbout();
            break;
    }

    HiliteMenu(FALSE, (int) (lastEvent.wmTaskData >> 16));
}
```

Displaying an About Box

```
void DoAbout(void) {
    AlertWindow(awResource + awButtonLayout, NULL, resAlertAbout);
}
```

Closing the Window

```
static void DoClose(void) {
   /* Check here to see if closing is okay */

   /* now close the window */

   CloseWindow(theWindow);
   quitFlag = 1;
}
```

The Main Program

```
int main(void) {
     StartTools();
     EventLoop();
     StopTools();
     return 0;
}
```

Let’s Try It Out

Adding a Window

Adding a Window

- Create your window resource

Adding a Window

- Create your window resource
- Use the Window Manager to open it

Adding a Window

- Create your window resource
- Use the Window Manager to open it
- Add code to draw its contents

Setting Up

```
#define win_width 400
#define win_height 120

#define win_left (((640-win_width)/2)+13)
#define win_top (((200-win_height)/2)+13)
#define win_right (win_left+win_width)
#define win_bottom (win_top+win_height)
#define win_rect {win_top, win_left, win_bottom, win_right}
```

The Window Resource

```
resource rWindParam1(resWindowDemo) {
   fCtlTie+fVis+fQContent+fMove+fClose+fTitle,
   resWindowDemo,
   0x0,
   {0, 0, 0, 0},
   0x07FF0001,
   {0, 0},
   {0, 0},
   {0, 0},
   {0, 0},
   {0, 0},
   0x0,
   0,
   win_rect,
   infront,
   resWindowDemo,
   0xA00 + ResourceToResource
};

resource rPString(resWindowDemo) {
```

Using the Window

```
static void CreateWindow(void) {
   theWindow = NewWindow2(NULL, 0, DrawTheWindow, NULL,
refIsResource,
                   resWindowDemo, 0x800E);
   ShowWindow(theWindow);
}

#pragma databank 1
static void DrawTheWindow(void) {
   DrawControls(GetPort());
}
#pragma databank 0
```

The New main()

```
int main(void) {
   StartTools();
   CreateWindow();
   EventLoop();
   StopTools();
   return 0;
}
```

Let’s Try It Out

Creating a Button

```
resource rControlTemplate (resButtonDoSomething) {
   resButtonDoSomething,
        {win_height - win_vmargin - btn_height,
            win_width - win_hmargin - btn_width,
            win_height - win_vmargin, win_width - win_hmargin},
   SimpleButtonControl {
       {   /* optional Fields */
           DefaultButton,
           FctlWantsEvents + FctlProcNotPtr + RefIsResource,
           ctlVisible,
           resButtonDoSomething,
           0x0,
           {   /* array: 1 elements */
               /* [1] */
               "\n",
               "\n",
               0x0,
               0x0
           }
       }
```

Adding a Control to a Window

```
resource rControlList(resWindowDemo) {
   {
       resButtonDoSomething
   };
};
```

Update the Event Loop

```
void EventLoop(void) {
   Word event;           // The event code returned by TaskMaster

   quitFlag = 0;         // We don't want to quit yet
   lastEvent.wmTaskMask = 0x1FFBFFL;

   do {
       event = TaskMaster(everyEvent, &lastEvent);
       switch(event) {  /* handle the events we need to */
                case wInSpecial:
            case wInMenuBar:
                HandleMenu((Word) (lastEvent.wmTaskData >> 16),
                           (Word) lastEvent.wmTaskData);
                break;
           case wInControl:
                HandleControl(lastEvent.wmTaskData4);
                break;
            default:
                break;
       }
```

Update the Event Loop

```
void EventLoop(void) {
   Word event;          // The event code returned by TaskMaster

   quitFlag = 0;        // We don't want to quit yet
   lastEvent.wmTaskMask = 0x1FFBFFL;

   do {
       event = TaskMaster(everyEvent, &lastEvent);
       switch(event) {  /* handle the events we need to */
               case wInSpecial:
           case wInMenuBar:
               HandleMenu((Word) (lastEvent.wmTaskData >> 16),
                          (Word) lastEvent.wmTaskData);
               break;
           case wInControl:
               HandleControl(lastEvent.wmTaskData4);
               break;
           default:
               break;
       }
```

Handling Control Clicks

```
void HandleControl(LongWord id) {
   switch(id) {
       case resButtonDoSomething:
          DoSomething();
          break;
   }
}
```

Let’s Try It Out

Questions?