UNIX実験

Mac OSX Programing

提出日：2005/07/01 消印有効

仲村そば

店員：
035737F：中村匡
035709A：宇座瞬
035740F：根保光秀
035743A：比嘉雅樹
035756B：宮國渡
035758J：村山正嗣

1　実験の目的

この実験では、MacOsX 上で動作するプログラムを作成し、GUI(Graphical User Interface) やオブジェクト指向のプログラミングについて学ぶ。 また、プログラムをチームで分担して開発することで”チーム作業で一つのプログラムを作り上げる”経験を積むことを目的とする。テーマが MacOsX プログラミングと広範囲におよぶので、ここでは特にネットワーク、グラフィックなどを中心に実験を進めていく。

2　実験の背景

GUI プログラミングに挑戦してみたかったので、新しいテーマの MaxOsX プログラミングに挑戦した。 MacOs でアプリケーションを開発するには、Unix/Linux の X window 、MacOs9 用の Classic、Classic をもとに開発された Carbon そして OsX 用の Cocoa。これらの形式を使って開発することができる。また使用できる言語は C,Perl,Java,Objective-C AppleScript etc... と非常に多い。本実験では、Cocoa-Java、Cocoa(Objective-C) を用いて実験を行うした。

3　Level1：スクリーンセーバー

3.1　作成目的

何気なく目にしているスクリーンセーバーを、自分で作成してみたいと感じ、Objective C を用いて作成した。

3.2　開発環境

- PC：ibook G4 1.07GHz
- OS：Mac OSX v10.4.1
- メモリ：768MB

3.3　スクリーンセーバーを作るには

今回、スクリーンセーバーを作成するにあたり、Xcode を使用した。
Xcode でスクリーンセーバーを作成するには Xcode を起動後に、 [ファイル] → [新規プロジェクト] → [Screen Saver] を選択し、任意に名前をつける。
するとそのディレクトリができる、その中の〜.m というファイルを加工する事でスクリーンセーバーを作成する事ができる。加工後にビルドボタンを押すとコンパイルされ、〜.server というファイルが作られるので、そのファイルを〜/Library/ScreenSavers に置けば良い。

3.4　スレッドの説明

〜.m の中には、あらかじめいくつかのメソッドが記述されている。
各メソッドの説明を下記に示す。基本的には、この中の- (void)animateOneFrame を加工する事で今回作成を行った。

表 1: 各メソッドの説明

メソッド名	説明
- (id)initWithFrame:(NSRect)frame isPreview:(BOOL)isPreview	インスタンスの初期化メソッド
- (void)startAnimation	スクリーンセーバーが開始するときに呼ばれるメソッド
- (void)stopAnimation	アニメーションが終了する時に呼ばれるメソッド
- (void)drawRect:(NSRect)rect	NSView の drawRect と同じく、実際の描画作業を行う
- (void)animateOneFrame	initWithFrame で指定された、[self setAnimation TimeInterval:〜] の間隔で呼び出されるメソッド
- (BOOL)hasConfigureSheet	システム環境設定→スクリーンエフェクトの、「設定」ボタンを Enable にするかしないかを決めるメソッド
- (NSWindow*)configureSheet	上記の設定ボタンを押すとこのメソッドが呼び出される

3.5 簡単なものを作ってみる

まず、四角形と円をランダムで表示させるものを作成した。
ソースファイルは下記参照。
`naha:/net/home/y03/j03040/public_html/nakamurasoba/43/sample.txt`

ソース中で、円と四角形の違いは、bezierPathWithOvalInRect と bezierPathWithRect の違い。
また、表示させる際は、fill だと塗りつぶして表示し、stroke だと線を引いて表示する。

3.5.1 実行結果

次に、作成したスクリーンセーバーの実行結果を下図に示す。

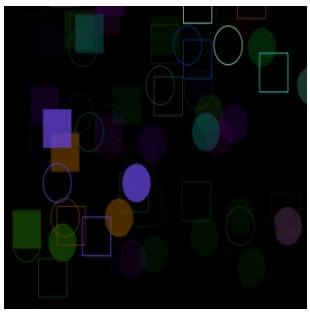


図 1: プログラムの実行結果

3.6 多角形を描く

次に、多角形を描くスクリーンセーバーを作成した。
ソースファイルは下記参照。
`naha:/net/home/y03/j03040/public_html/nakamurasoba/43/sample2.txt`

- (void)animateOneFrame に多く書きすぎると見にくくなるため、ランダムに色を設定する関数、ランダムなポイントを返す関数、多角形を描く関数をそれぞれ作った。描く際の `[ [self makePoly: [self rand_point] angle:0 r:0] stroke];` 、これは polynum と r に 0 を入れる事で、関数内で値をランダムで決めている。angle は図形を回す角度。(関数内で加算されていく) 多角形を描く makePoly 関数での num は角の数、r は辺の長さとなっている。

先程作った円と四角形を表示するスクリーンセーバーと、上記で作成した関数を合わせてできたス

クリーンセーバーのソースと実行結果を以下に示す。
naha:/net/home/y03/j03040/public_html/nakamurasoba/43/sample3.txt

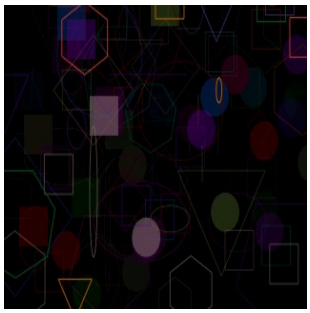


図 2: プログラムの実行結果

3.7 今後の課題

今回は ObjectC を用いて単純なスクリーンセーバーを作成したが、次の機会では OpenGL を使用して 本格的なデザインのスクリーンセーバーを作成してみたい。

参考文献

[1] はじめての Mac OSX プログラミング　”中村正弘”著

[2] http://homepage.mac.com/harawo/makeScreensaver.html
”はらを mac.com”

[3] http://www.big.or.jp/~crane/cocoa/ ”Cocoa はやっぱり！”

4 Level 2： Widget プログラミング

4.1 このレベルの目的

このレベルでは、MacOsX ”Tiger”の新機能である”DashBoard”上で動作するプログラム群”Widgets”のプログラミングを通して新しい tool や技術に挑戦する心構えやオブジェクト指向プログラミングについて学ぶ。

4.2 Level 0： Widget について調べる.

Widgets の構成は、メインとなる HTML とデザインを整えるための CSS、Widgets の動作を記述する Javascript、Widgets の情報を記述する XML だという事が分かる。この事から Widget は html と javascript の知識があれば簡単に開発できるという事が分かった。MacOSX には、javascript に widget 用のオブジェクトが用意されていて他のプログラムとの連携や OS のシステムコールを利用したり、ネットワーク通信が可能である。また、サブプログラムとして shell や perl,Python などを利用する事もできる。

4.3 Info.plist の記述

Info.plist は Widgets の情報をまとめたファイルで XML で記述されている。このファイルが無いと Widgets の実行ができないので記述しなければならない、必須項目を下記にまとめる。

表 2: Info.plist の記述

Key	説明
MainHTML	Widgets のメイン html
CFBundleIdentifier	DashBoard が Widgets を識別するための key
CFBundleName	Widgets の名前
CFBundleDisplayName	表示される名前
Width	Widget の幅
Height	Widget の高さ

また、ネットワークやシステムコールを使用したいときは下記の Key を追加する必要がある。

表 3: 特殊な Key

Key	説明
AllowNetworkAccess	ネットワークの使用を許可する
AllowSystem	システムコールの使用を許可する
AllowFileAccessOutsideOfWidget	外部プログラムとの連携を許可する
AllowFullAccess	上記すべてに許可を与える。

4.4 Level 1 簡単な Widget を作る。Hello Widget!

このレベルではサンプルコードを見ながら簡単な Widget を制作し、html、CSS や javascript の基本的な記述方法を学ぶ。サンプルコードを参考に widget を作成した。

表 4: html と CSS と javascript

```
<!–簡単な Widget プログラム—>
<?xml version=”1.0” encoding=”EUC-JP”?>
<!DOCTYPE html PUBLIC ”-//W3C//DTD XHTML 1.1//EN”
”http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd”>
<html xml:lang=”ja” xmlns=”http://www.w3.org/1999/xhtml”>
<!—CSS ファイルを読み込む—->
<style type=”text/css”>
@import ”Hello.css”;
</style>
¡!–javascript を読み込む—¿
<script type=’text/javascript’ src=’Hello.js’ />

</head>
<!– html が読み込まれたときに実行 –>
<body onload=”setup();”>
<!– 背景画像–>
<img src=”Default.png”>
<span class=”HelloText”>Hello Widgets!!</span>
</body>
</html>
```

```
CSS ファイル

#body{
margin:0;
}

.HelloText{
font: 23px ”Helvetica Neue”;
font-weight: bold;
color: #ffffff;
text-shadow: black 0px -1px 0px;
position: absolute;
left: 31px;
top: 100px;
}

javascript

function setup(){
return 0;
}
```

上表のように widget は普通の web コンテンツを制作するのと同じように開発する事ができる事が w かあった。また、その実行結果を下記に示す。

図 3: Hello！ Widget!実行結果

4.5 Level 2 : Widgets の”Flip”をプログラムする。

Flip とは、実行結果などを表示する画面と設定画面を切り替えるときの widget 独特の動きである。アニメーション自体は widget.prepareForTransition() として用意されているが実装するサンプルコードが無かったので 参考文献やサンプルコードを参考に Filp する widget を作成した。

表 5: Flip html

```
<html xml:lang=”ja”>

<style type=”text/css”> @import ”Flip.css”; </style>
<script type=’text/javascript’ src=’flip.js’ />
</head>
<body>
<!—-”fornt”を定義。マウスイベントも宣言する。—->
<div id=”front” onmousemove=”mousemove(event)”
onmouseout=”mouseexit(event)”>>
<img class=”bgimage”src=”Default.png” />
<span class=”FrontText”> 表 </span>
```

```
<!– 切り替えボタン–>
<div class=”flip” id=”fliprollie”></div>
<div class=”flip” id=”flip” onclick=”showPrefs()” on-
mouseover=”enterflip(event)”
onmouseout=”exitflip(event)”></div>
</div>

<!– ”back”を定義–>
<div id=”back”>
<img class=”bgimage” src=”Default_reverse.png” />
<span class=”BackText”> 裏 </span>
<div class=”doneButton”
onclick=”hidePrefs()”>Done</div>
</div>
</body>
</html>
```

表 6: Flip javascript のポイント

```
//フリップするプログラム
//表から裏へフリップする関数
function showPrefs() {
//flip アニメーションを流す
if (window.widget) {
widget.prepareForTransition("ToBack");
}
//表を隠して裏を表示する。
front.style.display="none";
back.style.display="block";

if (window.widget) {
setTimeout ('widget.performTransition();', 0);
}
document.getElementById('fliprollie').style.display =
'none';
}
//裏から表へフリップする関数
function hidePrefs() {
//showPrefs の逆の処理
}
.....
//widget 内にマウスが入ったら切り替えボタンを表示する関数
function mousemove (event) {
if (!flipShown) {
if (animation.timer != null) {
clearInterval (animation.timer);
animation.timer = null;
}
flipShown = true;
}
}
.....
```

```
//widget 内からマウスが出た時の処理 (切り替えボタンを消す)
function mouseexit (event) {
if (flipShown) {
if (animation.timer != null) {
clearInterval (animation.timer);
animation.timer = null;
.....
}
}
}
function animate() {
var T;
var ease;
var time = (new Date).getTime();

T = limit_3(time-animation.starttime, 0, anima-
tion.duration);

if (T ¿= animation.duration) {
clearInterval (animation.timer);
animation.timer = null;
animation.now = animation.to;
} else {
ease = 0.5 - (0.5 * Math.cos(Math.PI * T / anima-
tion.duration));
animation.now = computeNextFloat (animation.from,
animation.to, ease);
}

animation.firstElement.style.opacity = animation.now;
}
.....
function enterflip(event) {
document.getElementById('fliprollie').style.display =
'block';
}

function exitflip(event) {
document.getElementById('fliprollie').style.display =
'none';
}
```

プログラムの説明

Flip を実現するために showPrefs 関数と hidePrefs 関数を定義する。まず、widget.performTransition() を使って Flip アニメーションを表示した後"front" を非表示、"back" を表示にするだけである。hide-Prefs 関数はその逆で back.style.display="none"; front.style.display="block"; とするだけである。

また、mousemove 関数や、mouseexit 関数を定義して、Widget のウインドウにマウスが入ってきたら切り替えボタンを表示している。実行結果を下記に示す。

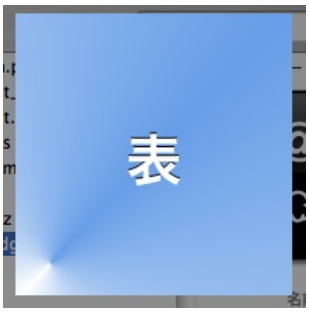


図 4: Flip 実行結果 1 "font"

図 5: Flip 実行結果 2 "back"

4.6 Level 3 : システムコールを使った Widgets

widget はプログラム中にシステムコールを呼び出して利用する事ができる。ここでは/usr/bin/host と/usr/bin/hostname を使って入力された HOST の IP アドレスを表示するプログラムを作成した。

```
//HOST → IP、IP → HOST を表示するプログラム

//最初読み込まれたとき実行する関数
function initrc()
{
// /usr/bin/hostname で自分の HOST 名を取得
var buf=widget.system("/usr/bin/hostname",
null).outputString;
var tmp=buf.split("¥n");
//入力欄に HOST 名を入力して Output 関数を呼び出す。
document.getElementById("inputText").value = tmp[0];
Output();
}

//ShowIP ボタンが押されたときの処理
function nslookup()
{
document.getElementById("doButton").src =
"Images/Button_down.png";
Output();
}

function buttonUpOut()
{
document.getElementById("doButton").src = "Im-
ages/Button.png";
}
```

```
//結果を表示する関数
function Output(){
var buf= new String();

//入力されたデータを /usr/bin/host に渡して結果を得る。
buf=widget.system("/usr/bin/host "+ doc-
ument.getElementById("inputText").value,
null).outputString;
//split を使って入力されたデータが IP なのか HOST 名なのかを
判断する。
var tmp=buf.split("has ");
var num=tmp.length;
var tmp2=buf.split("pointer ");
var num2=tmp2.length;

//結果を表示
if(num==2){
document.getElementById("outputText").value = tmp[1];
}else if(num2==2){
document.getElementById("outputText").value = tmp2[1];
}else{
document.getElementById("outputText").value = buf;
}
}
```

プログラムの説明

まず、widget が読み込まれたときは、自分の HOST 名と IP アドレスを表示させる。この処理は initrc() 関数で行っている。widget.system() を使い/usr/bin/hostname を呼び出して、結果を入力欄に出力して IP アドレスを表示する関数 Output() を呼び出す。Output() 関数は/usr/bin/host を widget.system で呼び出して引数として入力欄のデータを与えている。IP アドレスの時は"hostname has IP address"と返ってくるので split(has) で IP だけを取り出している。host 名の時は (入力が IP) "IP address pointer hostname"と返ってくる。もし split されていないのであれば入力したデータは無効であるという事で/usr/bin/host のエラーメッセージをそのまま表示させる。

実行結果

図 6: ShowIP 実行結果 1 hostname → ip

図 7: ShowIP 実行結果 2 ip → hostname

4.7 Level 4 : ネットワークを使った Widgets

widget はネットワーク通信を使って情報を取得したり、任意の URL にアクセスしたりできる。ここでは、入力した文字列をブラウザに渡して入力データを検索するというプログラムを作成した。

表 7: wSerch javascript

```
//検索するプログラム

//変数を定義
var q = "";
var url = "http://www.google.co.jp/search?q=";

function srch(q)
{
//スペースを+に置き換えて url とあわせる。
var srch = url + encodeURI(q.replace(/[\= s]/gi, '+')) ;

 ..........
if (widget)
//本当なら widget が用意している widget.openURL() 関数を使用するべきなのだが、
//日本語に対応していないのか日本語を入力すると実行されない。
//そこで、新しく関数を定義してそれを呼び出す。
//widget.openURL(srch);
better_openURL(srch);
}
}
```

```
//http://einekatze.jp/mt/を参考に日本語が通るように変更
function better_openURL(fullurl)
{
var    output=widget.system("/usr/bin/open    "+"fullurl",
null).outputString;
}

function setup() {
createGenericButton(document.getElementById('done'),
'done', hidePrefs);
}

//検索エンジンの切り替え
function changefr(elm){
switch( parseInt(elm.options[elm.selectedIndex].value) )
{
case 1:
url = "http://www.google.co.jp/search?q=";
break;
case 2:
url="http://images.google.co.jp/images?q=";
break;
case 3:
url="http://news.google.co.jp/news?&q=";
break;
case 4:
url="http://www.infoseek.co.jp/ITitles?&qt=";
break;
}
}
```

プログラムの説明

検索窓に入力した文字列を encodeURI() を使って URL 形式に直す、その時の入力が 2 つ以上あるときはスペースを+に置き換えて変数 url と結合。つぎに widget.openURL() を使って URL にアクセスするのだが日本語は扱えないようなので better_openURL() 関数を新たに作り代わりにする。better_openURL はシステムコールの/usr/bin/open を使って URL にアクセスする関数である。

実行結果

図 8: wSerch 実行結果

4.8 参考文献

- http://numata.aquasky.jp/
- http://www.piyosystems.com/kuki/B2060435708/index.html
- http://www.macdevcenter.com/pub/a/mac/2005/05/06/dashboard.html
- ibook の/Developer/Examples/Dashboard/Sample Code/以下

5 Cocoa-java プログラミング

5.1 このレベルの目的

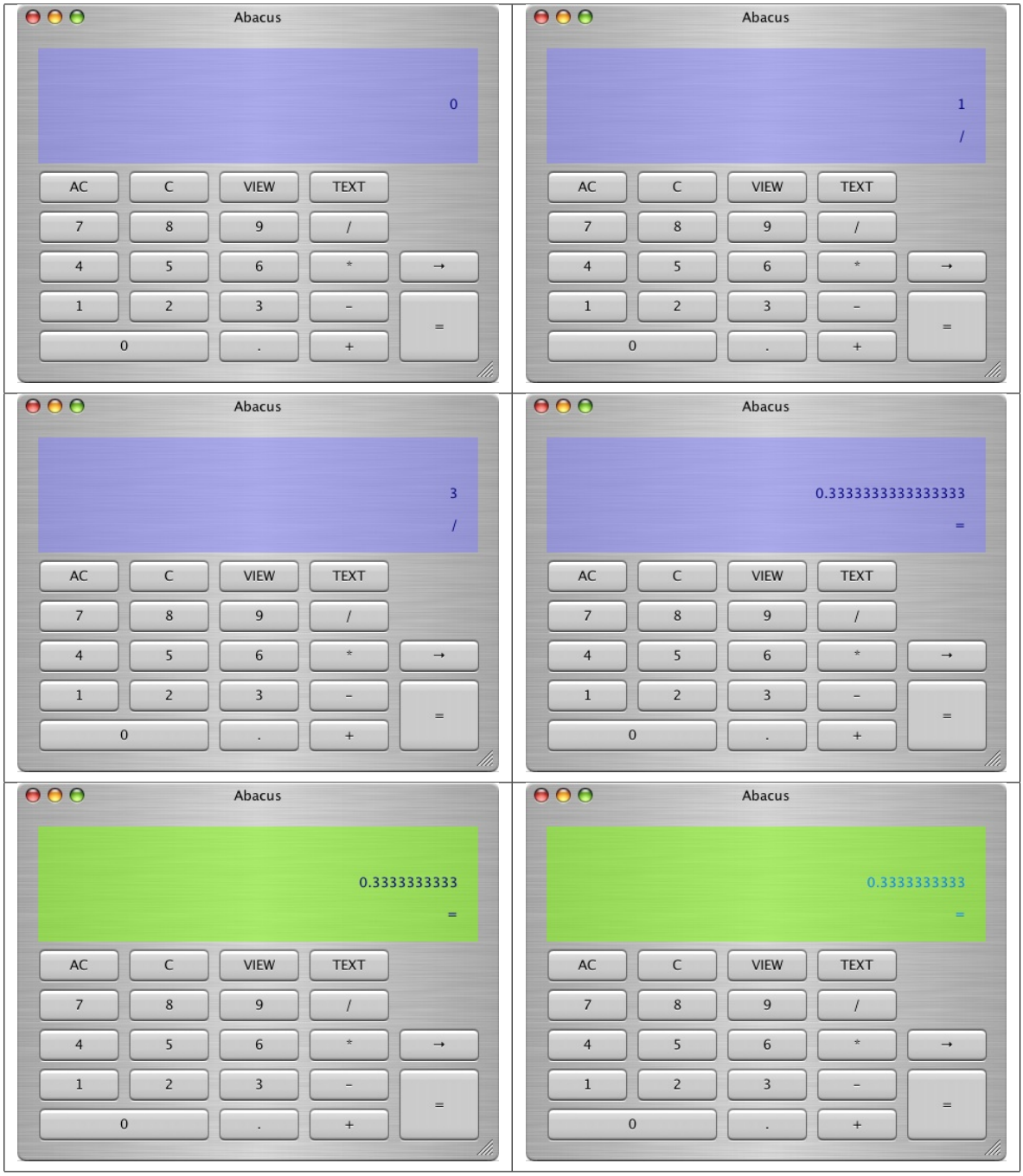
あまり流行していない Cocoa-java プログラミングの基礎を学ぶ

5.2 電卓の概要

メインプログラム (Abacus)

- Outlet
  - main_tf(NSTextFIeld) 主に数値を表示
  - sub_tf(NSTextFIeld) 演算記号などを表示
  - main_view(MyView(implements NSView)) 背景色を表示
- Action
  - key_arithmetric(NSButton sender) 四則演算と→、=のボタンと接続
  - key_num(NSButton sender) 数字のボタンと接続
  - key_mode(NSButton sender)VIEW,TEXT ボタンと接続
  - key_clear(NSButton sender)AC,C と接続
- MyView
  - public void setColor(float red,float green,float blue); これにより、BG の色を設定している。

5.3 実行結果



5.4 参考文献

Marc Prud'hommeaux　http://storrow.org/

6 Level4：電卓

6.1 作成目的

object_C を用いてとりあえずプログラムを動かしてみたかったため、お絵描きソフトを作成することにした。

6.2 アプリケーションの概要

作成、実行環境

- Mac OS X 10.4 Tiger
- Xcode 2.0
- 言語:Object-C

6.3 プログラムの説明

ウインドウやボタン等を Interface Builder で作成した。
PaintView.m の説明

```
- (void)mouseDown:(NSEvent *)theEvent
   {
    NSPoint mouseLocation = [self convertPoint:[theEvent locationInWindow] fromView:nil];

    myPath=[[NSBezierPath alloc] init];
    [myPath setLineWidth:[widthSlider floatValue]];
    [myPath setLineCapStyle:NSRoundLineCapStyle];
    [myPath setLineJoinStyle:NSRoundLineJoinStyle];

    [myPath moveToPoint:mouseLocation];
}
```

マウスのボタンが押されたときに呼ばれ、NSPoint mouseLocation =... で座標を自分自身のビューの座標に修正して取得する。NSBezierPath のインスタンスを作成し、パスの属性 (線の幅、両端の形態、接続形態) を設定するパスの開始点を設定する

```
   //マウスがドラッグされているときに連続的に呼ばれるメソッド
- (void)mouseDragged:(NSEvent *)theEvent
{
    NSPoint mouseLocation = [self convertPoint:[theEvent locationInWindow] fromView:nil];

    [bufferImage lockFocus];

    [myPath lineToPoint:mouseLocation];
    [[colorWell color] set];
    [myPath stroke];

    [bufferImage unlockFocus];
    [self display];
}
```

マウスがドラッグされているときに呼ばれ、イベントが発生した座標を取得し、bufferImage にフォーカスを設定する。パスに直線を引く座標を追加し、描画色をカラーウェルから取得して設定して bufferImage に描画する。bufferImage のフォーカスを解除し、ビューにコピーして表示する。

```
 //マウスボタンが押された後、押されていない状態に戻るときに呼ばれるメソッド
- (void)mouseUp:(NSEvent *)theEvent
{
    [myPath release];
}
```

NSBezierPath クラスのインスタンス myPath を解放する

```
 //ファイルにセーブする
- (void)didEndSaveSheet:(NSSavePanel *)savePanel
returnCode:(int)returnCode contextInfo:(void *)contextInfo
{
    if (returnCode == NSOKButton) {
        NSString *filename=[savePanel filename];
        NSData *tiffData=[bufferImage TIFFRepresentation];
        [tiffData writeToFile:filename atomically:YES];
    }
}
```

表示させたシートを使い、TIFFRepresentation メソッドで TIFF 形式のデータを作成する。このデータを指定されたファイル名で保存する。

```
//ファイルを開く
- (void)didEndOpenSheet:(NSOpenPanel *)openPanel
returnCode:(int)returnCode contextInfo:(void *)contextInfo
{
    if (returnCode == NSOKButton) {
        NSString *filename=[openPanel filename];
        [bufferImage release];
        bufferImage=[[NSImage alloc] initWithContentsOfFile:filename];
        [bufferImage setSize:[self frame].size];
        [bufferImage setScalesWhenResized:YES];
    }
}
```

これまで使用していた bufferImage を解放し、新しい bufferImage を作成し failmame で示される TIFF ファイルを読み込んで初期化する。イメージのサイズをビューと同じ大きさにあわせ、オリジナルの TIFF 画像の内部データのサイズも設定する。

6.4 実行結果

6.5 今後の課題

今回は簡易お絵描きソフトだったので色と、太さを選ぶボタンのみを作ったがもっとボタンを増やし、多角形や円を書いたりするようにしたい。

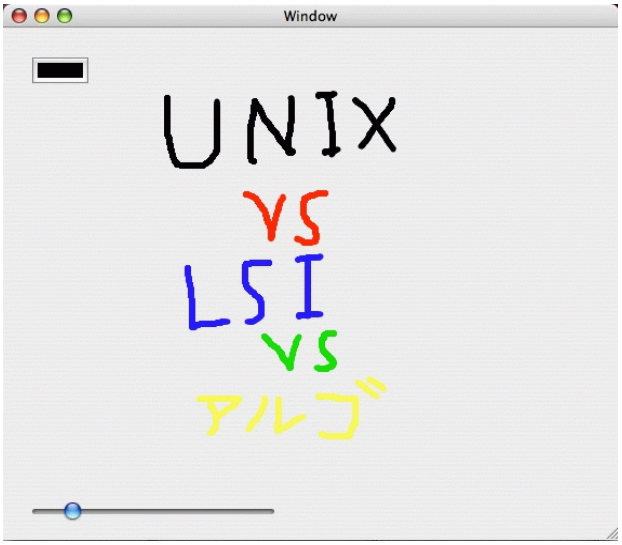


図 9: SamplePaint の実行結果

参考文献

[1] はじめての Mac OSX プログラミング ”中村正弘”著

[2] Cocoa はやっぱり! http://www.big.or.jp/~crane/cocoa/

7 Level 5 オセロット (オセロ)

7.1 目的

オセロ開発を通し cocoa-java になれる。

7.2 アプリケーションの概要

作成・実行環境

- Mac Os X 10.4 Tiger
- Xcode 2.0
- 言語：java

7.3 プログラム説明

ソース

- http://www.ie.u-ryukyu.ac.jp/ j03040/nakamurasoba/37/Gamectrl.java
- http://www.ie.u-ryukyu.ac.jp/ j03040/nakamurasoba/37/Kernel.java
- http://www.ie.u-ryukyu.ac.jp/ j03040/nakamurasoba/37/MyView.java

7.3.1 Gamectrl.java について

このプログラムはウィンドウに表示される pass ボタンの設計をしている。

7.3.2 Kernel.java について

```
public void pass(){
        turn = (turn+1)%2;
        System.out.println("pass");
}
```

ここでは、pass をするときの設定をしてる。

```
public int[][] clicked(int x2,int y2){
        x=x2; y=y2;
        int count = 0;
        if( desk[x2][y2] == -1 ) {
 ...
```

ここではクリックされた時の設定をしている。

$if(desk[x2][y2] == -1)$ では石が置かれていなかったらを分岐させている。次に出てくる $desk[x2][y2] = turn$ での turn は石の色を指定するためのもの。

```
    for(int i = 0;i < MATRIX;i++){
            System.out.println();
            for(int n=0;n < MATRIX;n++) {
       …
```

ここでは配列の中の値 (石の有無や色) を出力する。

```
              count += top(x2,y2);
              count += right(x2,y2);
              count += left(x2, y2);
               …
```

ここでは石を置くかどうかの判断と置いたときに裏返すことを指示している。コメントアウトされている数字は、石を置いたときにその方向に裏返させれるかを調べるもの。どの方向にも裏返らせることができない場合 ($count == 0$) は石を置けないようになっている。

7.3.3 MyView.java について

```
public class MyView extends NSView {
        NSPoint p1;
        NSPoint list[] = new NSPoint[64];
         …
        }
```

ここでは MyView を使うための準備をする。

```
 public void drawRect(NSRect rect) {
        NSRect r2 ;
        NSBezierPath p2;
         …
        }
```

ここでは、drawRect メソッドを呼び出して渡す準備をする。

```
  /*デフォルト描画部分*/
NSColor.blackColor().set();
for(int d_x = 0;d_x < matrix;d_x++){
        for(int d_y=0;d_y < matrix;d_y++)
NSBezierPath.bezierPathWithRect(newNSRect(d_x*size+w_point_x,
d_y*size+w_point_x,size,size)).stroke();
                          }
```

ここでは初期状態のマスを書いています。¡br¿ NSColor.blackColor().set() では、NSColor というオブジェクトから黒を呼び出すメソッドで、その色を使うことを表している。
for 文で枠を決め、 .stroke() で中身を塗りつぶさないように線を引く。

```
public void mouseDown(NSEvent event) {
        float aho,baka;
        int xza,yza;
        p1 = event.locationInWindow();
        p1 = convertPointFromView(p1,null);
 ...
}
```

ここでは、マウスにあわせての反応を表している。
まず、 public void mouseDown がマウスが押し下げられた時に呼び出されるもので、(NSEvent event) がこのときに発生するイベントをまとめたクラスである。

次に locationInWindow というメソッドを呼び出し、マウスポインタの位置を調べる。そして、位置を示す NSPoint インスタンスを返してくる。ただし、これで得られる値はコントロールの値ではなく、コントロールが置かれているウィンドウ内の位置であるので、そのままでは使いにくいため、設定をする。

そこで、convertPointFramView を使い、ウィンドウ内の相対位置をコントロール内の相対位置にする。ここで NSPoint を変換した値 (NSPoint) を返すことになり、第一引数が元の NSPoint、第二引数が元の基準となっているコントロールのインスタンスとなる。普段、ウィンドウにこのコントロールが収められている場合、値は null にしておく。

```
public void plotadd(NSPoint p2){
      list[i++]=p2;
      display();
}
public void plot(){
      display();
}
```

ここの記述で上記に示したポインタに出力する。

7.4 実行結果

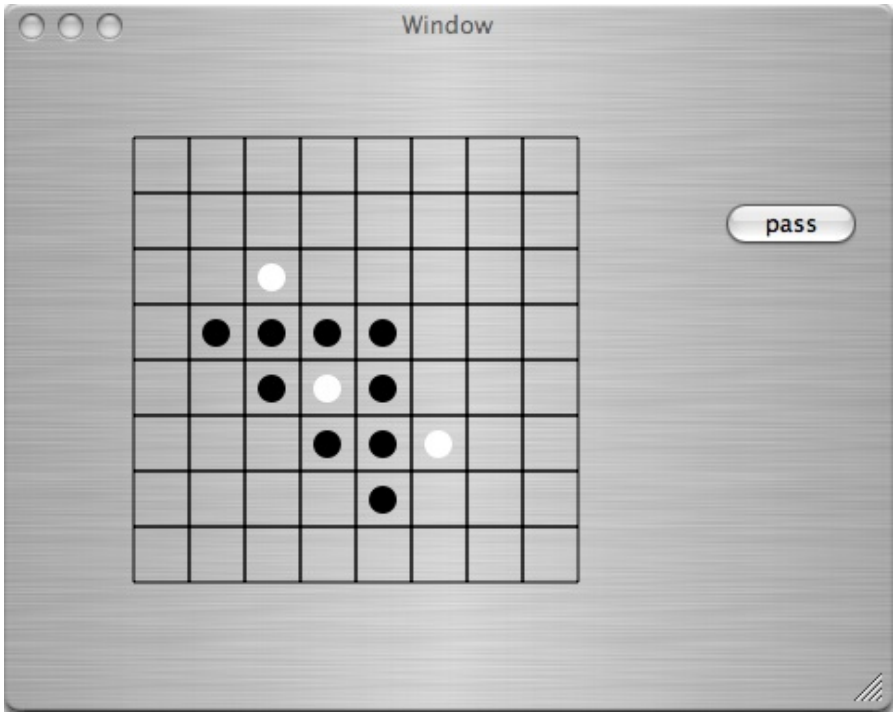


図 10: 実行結果

7.5 今後の課題

cocoa には今回使用した言語の cocoa-java 以外にも objective-c、Apple-script 等があるのでそれぞれ開発できるくらいのレベルになりたいと思う。

また、今回は普通のオセロだが、設定次第では色々なルールを作ったり、板の大きさを変えたりできるので面白いルールを考え改良したいと思う。

参考文献

[1] Cocoa リファレンス
http://www.oomori.com/cocoafw/index.html

[2] Cocoa-Java 入門教室
http://www.h5.dion.ne.jp/

8 Level6：RSS リーダー

8.1 RSS リーダーとは

Web サイトを巡回して RSS 形式の更新情報を受信し、リンク一覧の形で表示するアプリケーションソフトである。指定したサイトの RSS 情報を一定時間ごとに自動的にダウンロードし、更新があると記事へのリンクを表示してユーザに知らせるのが一般的である。

今回作成する RSS リーダーは、あらかじめ RSS を追加して一覧化し、Parse ボタンを押すと選択した RSS 情報をダウンロードし、画面に表示するものとする。

8.2 アプリケーションの概要

(http://members.at.infoseek.co.jp/Route147/RSSReader.tar.gz) をベースとして、以下の機能を拡張した。

- RSS の追加による RSS の一覧化とその削除
- 「link」項目をクリックするとデフォルトブラウザで開く

Xcode2.0 による設計で、言語は Object-C を使う。

8.3 プログラムの説明

今回作成するアプリケーションの主なルーチンを以下に示す。

Controller	メインウィンドウ、RSS の読み込み、起動及び終了の処理
AddUrl	サブウィンドウ、RSS の追加及び削除処理
NSTextViewWithLinks	リンク上でマウスカーソルを変化させる。

各ルーチンでの処理は以下の節で順を追って説明する。

8.3.1 ウィンドウ

アプリケーションのメインウィンドウを図 14 に示す。

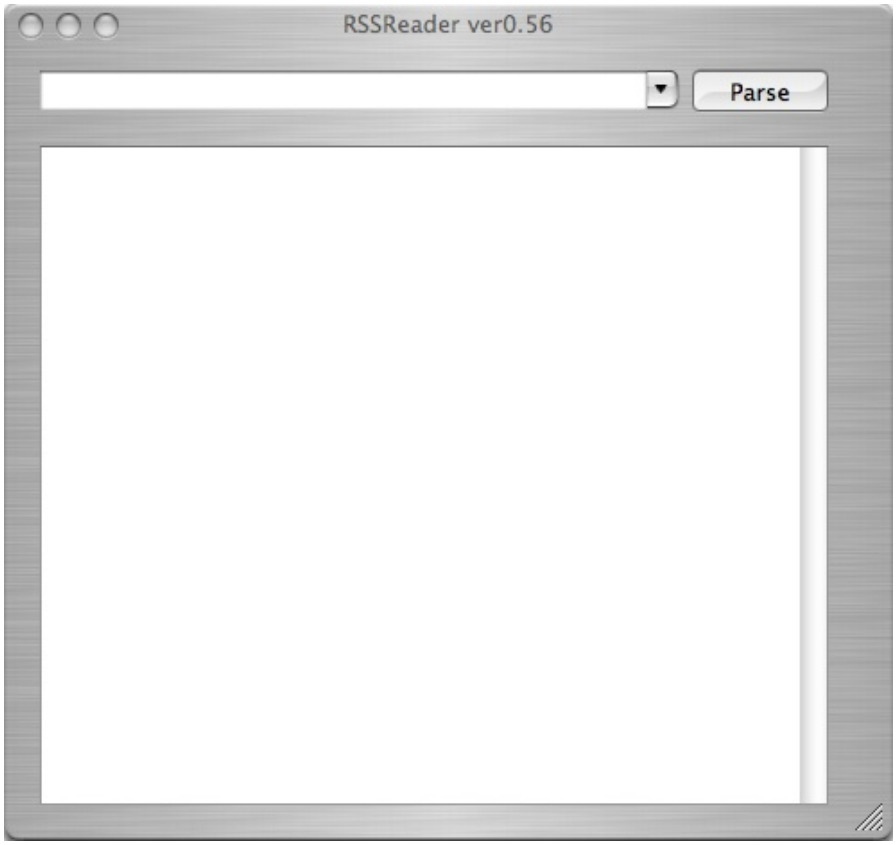


図 11: メインウィンドウ

Parse ボタンを押すと $-(IBAction)getContents:(id)sender$ メソッドが呼び出され、TextField 中の RSS 情報のダウンロードを開始する。

```
- (IBAction)getContents:(id)sender
{
  // 指定した URL
  // selUrl は ComboBox のインスタンス
  NSURL *url = [NSURL URLWithString: [selUrl stringValue]];

  ...
```

サブウィンドウを図 15 に示す。

図 12: RSS 追加・削除ウィンドウ

追加ボタンを押すと、$-(IBAction)addURL:(id)sender$ が呼び出され、TextField 中の URL をメインウィンドウの ComboBox に追加する。また、削除ボタンを押すと $-(IBAction)rmURL:(id)sender;$ が呼ばれ、メインウィンドウの ComboBox の要素をすべて削除する。

```
- (IBAction)addURL:(id)sender
{
  NSString    *url  = [addUrl stringValue];
  [selUrl setUsesDataSource:NO];
  [selUrl addItemWithObjectValue: url];
}

- (IBAction)rmURL:(id)sender;
{
  [selUrl setUsesDataSource:NO];
  [selUrl removeAllItems];
}
```

8.3.2 RSS 情報の読み込み、解析

通常、RSS リーダーでは XML 文書である RSS を利用する際に、XML パーサというソフトウェアを使用する。XML はテキストファイルの形で存在するため、そのままアプリケーションで利用するのは困難である。しかし、一定の形式が定められているため、汎用的な解釈プログラムである XML パーサによって変換したデータを使うほうが効率が良い。

今回は Object-C に用意されているパーサとして、NSXMLParser を使用する。しかし、NSXMLParser はツリー構造にパースしてくれるわけではなく、XML 文書の最初から読み下っていき、「どんなタグが始まった」「どんな値が入っていた」という処理を逐一拾っていくという方式をとらなければならない。そこで、NSXMLParser のデリゲートを Controller のインスタンスにセットする。

```
// 指定した URL からデータを読み込んで、
// XML パーサオブジェクトを返す。
NSXMLParser *parser = [ [ NSXMLParser alloc ]
                        initWithContentsOfURL:url];
// デリゲートをセット
[parser setDelegate:self];
```

url はメインウィンドウで TextField に入力されていた URL である。デリゲートをセットすることで、パース中に以下のメソッドが間接的に呼び出されるため、RSS の解析がしやすくなる。

```
// ドキュメントのパースの開始
- (void)parserDidStartDocument:(NSXMLParser *)parser;

// ドキュメントのパースの完了
- (void)parserDidEndDocument:(NSXMLParser *)parser;

// タグの開始
- (void)parser:(NSXMLParser *)parser
  didStartElement:(NSString *)elementName
  namespaceURI:(NSString *)namespaceURI
  qualifiedName:(NSString *)qName
  attributes:(NSDictionary *)attributeDict
```

```
// タグの中身
- (void)parser:(NSXMLParser *)parser
  foundCharacters:(NSString *)string;

// タグの終了
- (void)parser:(NSXMLParser *)parser
  didEndElement:(NSString *)elementName
  namespaceURI:(NSString *)namespaceURI
  qualifiedName:(NSString *)qName;
```

各メソッドでタグを認識したり、その時の値を読み込むなどの処理を行っている。

8.3.3 RSS 情報の出力

RSS のパースに成功すると、メインウィンドウの TextView への出力処理を開始する。

今回は「title」「description」「link」項目を出力するが、「link」に関しては『クリックするとブラウザで開く』という動作を行うようにする。

```
// NSDictionary:キーと値のペアを格納する連想配列クラス
// contents:RSS 情報から取り出した「link」の値

NSDictionary *attrs =
[NSDictionary dictionaryWithObject:contents
forKey:NSLinkAttributeName];

// temp:NSAttributedString クラス
// 属性（色やフォントなど）の情報をもつ文字列クラス

temp =
[[NSAttributedString alloc]
initWithString:[NSString stringWithFormat:@"%@\n",contents]
attributes:attrs];
```

まず「contents はリンクである」という情報を attrs に格納し、それからリンクの属性を contents に付加して temp に返す。こうすることにより、クリックするとデフォルトブラウザが起動する処理が行われるようになる。

また、リンク上でマウスカーソルが変化するような処理も行っている。これは (http://developer.apple.com/samplecode/ から NSTextViewWithLinks.{m,h}、fingerCursor.tiff をダウンロードし、プロジェクトに追加する事により実現した。

8.3.4 設定の保存・読み込み

アプリケーションの起動時と終了時に、RSS の追加及び削除に関する設定を保存する処理を行う。

```
// アプリケーション終了処理
// 環境設定を保存
- (void)windowWillClose:(NSNotification *)aNotification
{
  NSUserDefaults *defaults =
                  [NSUserDefaults standardUserDefaults];
```

```
  NSArray *temp = [[NSArray alloc]
                   initWithArray:[selUrl objectValues] ];
  [selUrl setUsesDataSource:NO];
  [defaults setObject:temp forKey:@"listUrl"];
  [defaults synchronize];
}
```

まず NSUserDefaults という、初期設定を操作するクラスのインスタンス化を行う。そして「listUrl」というキーで RSS の情報が格納されている配列を defaults にセットする。その後の synchronize にて初期設定ファイルに保存される。ここで、初期設定ファイルはプロジェクト中の Info.plist で指定した名前で”˜/Library/Preferences/*.plist”に保存される。

```
// アプリケーションの起動処理
// 環境設定の読み込み
- (void) applicationDidFinishLaunching:
         (NSNotification *)aNotification
{
  [spin setDisplayedWhenStopped:NO];
  NSUserDefaults *defaults =
                [ NSUserDefaults standardUserDefaults ];
  id sel = [defaults objectForKey:@"listUrl"];

  // selUrl:メインウィンドウの ComboBox のインスタンス
  [selUrl setUsesDataSource:NO];
  [selUrl addItemsWithObjectValues:sel];
}
```

初期設定から、「listUrl」というキーに格納されている値を sel に返し、メインウィンドウの ComboBox に追加するという流れである。

8.4 実行画面

実際にアプリケーションを実行した様子を図 13 に示す

8.5 今後の課題

RSS には異なるバージョンがいくつか存在し、それはバージョンアップではなく派生したものという少し複雑な状況が発生している。また、それに対抗しようと新たなフォーマット（Atom や OPML など）も作成されてきており、今回のような単純に入力された URL から情報をとるという方式では、いくつかの RSS では情報が読み切れずエラーを起こしたり結果が出力されなかったりする。

今回は簡易 RSS リーダということで異なるバージョンへの対処はしきれなかったが、さらに上のアプリケーションを作成するのであればこういった互換性に対しての処理が必要になってくるであろう。

参考文献

[1] Cocoa リファレンス
http://www.oomori.com/cocoafw/index.html

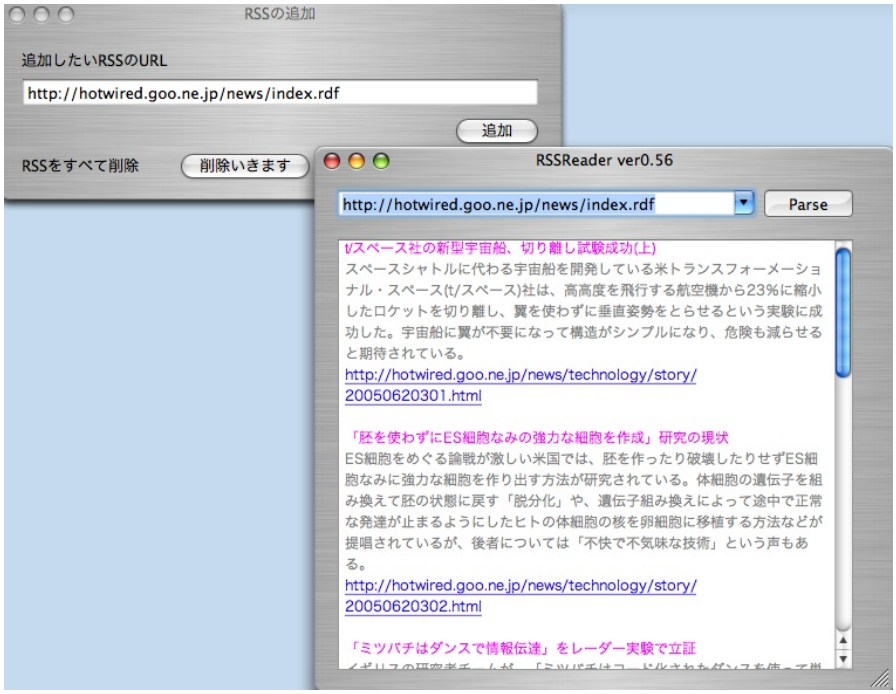


図 13: 実行画面

[2] Cocoa に関するヌマタメモ
http://numata.aquasky.jp/numamemo/viewer.cgi?group=cocoa

[3] Cocoa Lab.
http://hogehoge.plala.jp/pub/lab/cocolog/hogelog.php?

9 LevelX：ネットゲーム

9.1 内容および目的

Level X では、実験目標を満たすような課題を考え、ネットワークゲームを開発する事になった。チーム作業を行う上で、グループをゲーム班、ネットワーク班、インターフェース班に分けて開発を行った。 ゲーム班はトランプゲームのプログラミング、ネットワーク班はネットワーク通信、インターフェースは GUI の開発+合体を行う。 この Level を通して、cocoa プログラミング、ネットワーク、GUI プログラミング、そして、チーム作業について学ぶ事を目的とする。

班分けは以下のようにして行った。

- ゲーム班
  035737F：中村匡
  035743A：比嘉雅樹
  035756B：宮國渡
- ネットワーク班
  035709A：宇座瞬
  035758J：村山正嗣
- インターフェース班
  035740F：根保光秀

9.2 ブラックジャック

カジノの定番とも言えるカードゲーム「ブラックジャック」。今回は親 (ディーラー) と子の一対一のゲームを設計する。

9.3 アプリケーションの概要

作成環境

- Mac OS X 10.4
- Xcode 2.0
- Cocoa Java

プログラムソースは Web レポートからリンクを辿るか、以下からダウンロードしてください。

naha:/home/y03/j03040/public_html/nakamurasoba/xXx/56/*.java

9.4 プログラムの説明

9.4.1 トランプクラス (Card.java)

カードの山を二次元配列で表し、カードをリセットしたり、カードを取り出したりする。以下に二次元配列でのカードの表現を示す。

card_stat[2][2]=1 のとき、ダイヤの 3 は使っている (山にない)。
card_stat[1][8]=0 のとき、スペードの 9 は使ってない (山にある)。

主なメソッドを以下に示す。

```
// 引数で、カードの山に設定するジョーカーの数を設定する。
public Card(int baban);

// カードの状態をリセットする (全て未使用にする)。
public void reset();

// 乱数によりカードを一枚選び、カードを表す int 型の値を返す。
// このとき、そのカードの状態を「使用済み」にする。
int get_card();
```

9.4.2 プレイヤークラス (Player.java)

プレイヤーの行動や得点を出力する。このクラスのサブクラスとして、親クラスと子クラスがある。

```
// 所持カードを表す Vector 型変数 cards を初期化
// 同時にカードの情報を表す Card クラスのインスタンス st を設定。
Player(Card st);

//「カードの山からトランプを一枚抜く」を行う。
void pullCard();

//「所持カードをすべて山に戻す (所持カードの初期化)」を行う。
```

```
void backCard();

/* 所持カードの点数の合計を、
 * ブラックジャックのルールに沿って求める
 * (11 以上のカードは全て 10 とみなす。
 * 1 は 11 にもなるが、自動で最適値をとる。)
 */
int totalScore();
```

9.4.3 親クラス (Parent.java)

基本的にここは自動で行う。

```
// 親に、子とメインプログラムの情報 (インスタンス) を教える。
void setClass(Child c, Blackjack bj);

// 親に、所持カードを表示するメイン画面の場所を教える。
void setView(GameView view);

/* 親の最初の行動。開始時に配られたカードのうち、
 * 一枚を裏、一枚を表にして表示する。
 * その後、子に操作を譲り、以後子がバースト (またはオリ) するまで
 * 親は何にもしない。
 */
void turn1();

/* 子の処理が全て終了したら呼ばれる。
 * 自分の所持カードの点数の合計が
 * 17 以上になるまではカードを引き続ける。
 * それを終了すると自分の所持カードを全て表で
 * メイン画面に出力してゲームの終了処理に入る。
 */
void turn2();
```

9.4.4 子クラス (Child.java)

実際にユーザが操作するプレイヤーである。

```
/* Blackjack から送られてきたボタンの情報をセットする。
 * これにより、そのボタンを押すとこのクラス内の
 * 指定したメソッド (stand , hit) を呼び出せる。
 * 通常、この処理は Interface Builder で行える。
 */
void setButton(NSbutton s, NSbutton h);

// 子の最初の処理。"hit"及び"stand"ボタンを使用可能にする。
// 今後、子の処理は設定したボタンを押すことによって行われる。
void turn();

// "hit"ボタンが押されたら呼ばれる。カードの山から一枚とる。
```

```
// この時点で所持カードの点数が 21 を越えたら、即時終了となる。
void hit(Object sender);

// "stand"ボタンが押されたら呼ばれる。カードを引かず勝負に出る。
void stand(Object sender);
```

9.4.5 メイン画面クラス (GameView.java)

Interface Builder でのメイン画面を図 14 に示す。

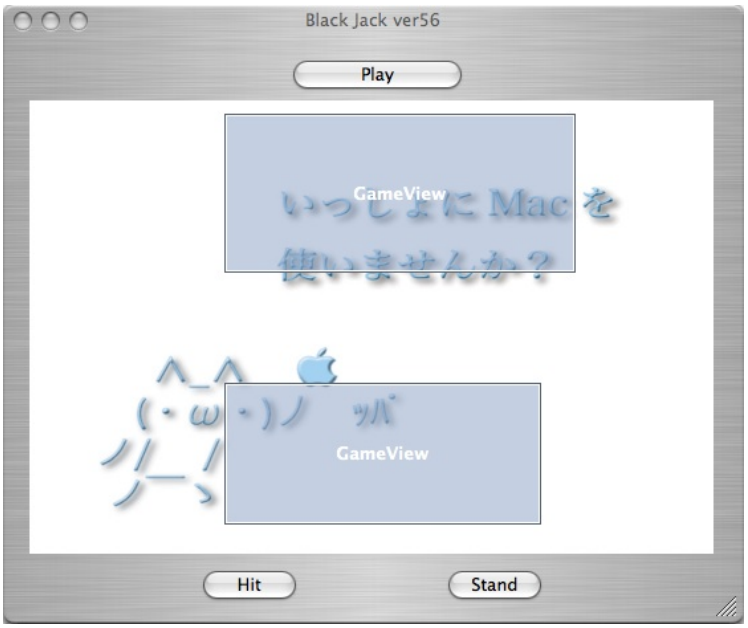


図 14: 親の領域 (上)、子の領域 (下)

上の GameView が親、下が子の領域である。

```
// 二次元配列 NSImage にカードの画像の情報を格納する。
public GameView(NSRect rect)

/* 画像を描写する。ここで、m は描写のモードを表している。
 * [m=0] 所持カードを全て表にして表示
 * [m=1] 所持カードのうち、一枚目を裏、
 *       二枚目を表にして出力する (ディーラー専用)
 */
void plot(int m);

// 親、もしくは子のインスタンスをセットする。
void setPlayer(Player c);

// 描写が必要になったとき、
// もしくは display(); を呼んだときに処理を開始する。
public void drawRect(NSRect rect);
```

9.4.6　メインプログラム

アプリケーション起動時に、Parent、Child、Card クラスのインスタンスを生成。ゲーム処理を開始する

```
// ボタンの設定や各クラスへのインスタンスの受け渡しなどを行う。
void gameInit();

// "start"ボタンが押されたら開始する。
// 親と子にそれぞれ二枚ずつカードを配り、親のターンに入る。
public void start(Object sender);

// 親と子の処理が全て終了したら呼ばれる。
// 所持カードの点数の合計から、勝敗を決める。
void gameResult();
```

今回のルールは以下のようになる。

- 21 以下で、大きい数字のプレイヤーの勝ち
- 引き分けだと親の勝ち
- 子がバーストすると親の勝ち。両方バーストでも親の勝ち
- スプリットやダブルダウンなどのルールはない。単純に数字の大小できめる

9.5　実行結果

実行結果を図 15 に示す。

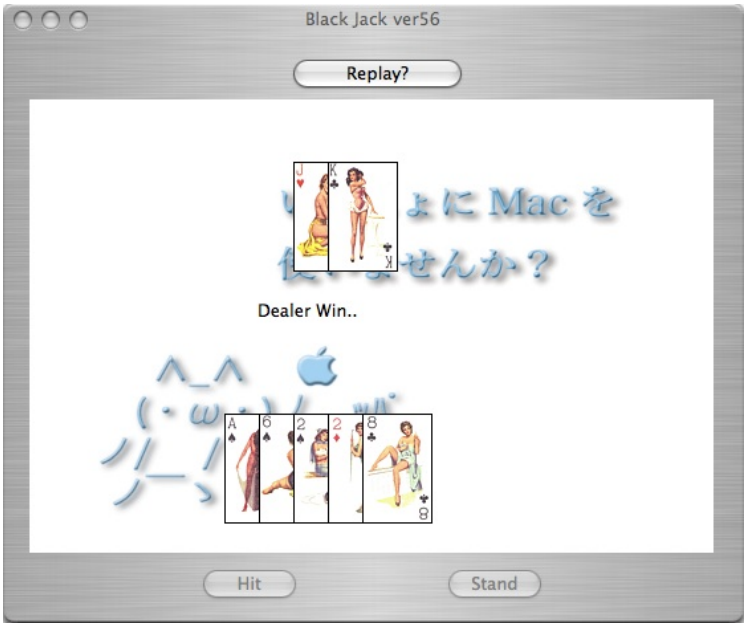


図 15: 実行結果

9.6 ネットワーク班

9.6.1 Network_IO の説明

複数のクライアントとの通信をひとまとめにし、できるだけ簡単に行う事を目的としたプログラム。

- コンストラクタ Network_IO(int port_no,int connection_max)
  コンストラクタで、ポート no と接続できるクライアントの数を設定
- public void start()
  スレッドのスタート同時にサーバとサーバの受信用スレッド、クライアントの送 信用スレッドを作成する。
- public void run()
  主に、ip テーブルの更新を行っている。接続が切断されたソケットは、削除する。接続を試みたクライアントのソケットを、ip テーブルに入れる。
- public void connect(String ip_addr) public void connect(String ip_addr,int port_no) 接続を試みる。接続用のスレッドのキューに追加する。
- public Vector get_message() ここでは、BufferReader の中身を読み出して、Vector 型に詰めて返す。フォーマットは、[ip_addres:message] といった形をとる。なので、正確な受信メッセージわからない。
- public void close()
  接続を閉じる

表 8: Network_IO

```
 ..................
public void send_message(String ip_addr,String s){
int i=0;
try{
for(i=0;i¡CONNECTION_MAX;i++){
if((IP_LIST[i]!=null)&&(IP_LIST[i].equals(ip_addr))){
socket_out[i].write(s);
socket_out[i].flush();
return ;
}
} .........
public Network_IO(int port_no,int connection_max){
PORT_NO=port_no;
CONNECTION_MAX=connection_max;
connect=new Socket[CONNECTION_MAX];
socket_in = new BufferedReader[CONNECTION_MAX];
socket_out=new OutputStreamWriter[CONNECTION_MAX];
IP_LIST=new String[CONNECTION_MAX];
for(int i=0;i¡CONNECTION_MAX;i++){
IP_LIST[i]="";
}
}
 ...........
```

```
 ..............
public Vector get_message(){
Vector v=new Vector();
String s;
try{
for(int i=0;i¡CONNECTION_MAX;i++){
if((socket_in[i]!=null)&&(socket_in[i].ready())){
s="";
while(socket_in[i].ready()){
s=socket_in[i].readLine();
v.add(s);
}
}
} ..............
```

9.7 インターフェース班のレポート

9.7.1 ゲームと通信の合体

ゲーム班の作成した。ブラックジャックとインディアンポーカーを一つにまとめて、ネットワーク班の作成した通信機能を付加する。

9.7.2 Game_Ctl の説明

- Game_Ctl ボタンを押した時の処理とネットワーク通信を行うクラス
- applicationDidFinishLaunching(NSNotification aNotification)
  Cocoa プログラムが実行された時に実行されるメソッド。bt_bj.setEnabled(true);　ボタンの表示、Network_IO の設定を行う。
- init_net()
  入力された IP アドレスにコネクトして、スレッドを走らせる。
- スレッドはメッセージの送受信を行う 。rev = this_network.get_message(); メッセージの受信を行う。this_network.send_message(ip_another,(String)send.get(i)); メッセージを送信する。
- pochi() ボタンが押された時実行するメソッド。ip_another=ip_area.stringValue(); 入力された IP アドレスを代入する。

9.7.3 Iporker の説明

- 基本的な構成はブラックジャックと同様。ルールが異なる。
- button_event(Object sender) ボタンが押されたとき実行されるメソッド。
- gameResult() カードの値を比較し、大きい方の勝ち

以上でコンピュータと対戦するゲームは完成した。

9.7.4 実行結果

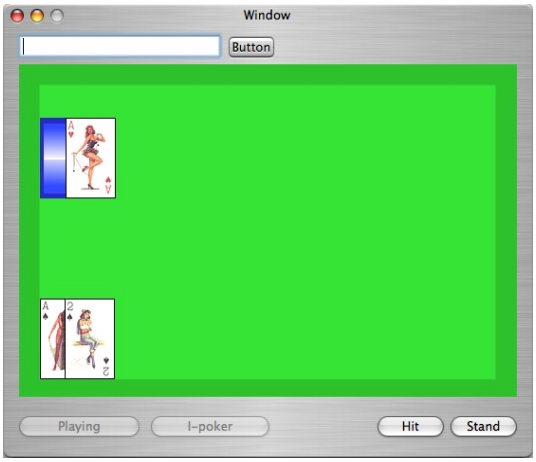


図 16: 実行結果

9.7.5 ネットワーク通信

メッセージの送受信までは実装できたが、情報が欲しいときにデータと取得するという処理ができなかった。相手が入力しなければメッセージはいつまでも送られない、反対にすでにデータ変数に蓄えられていた場合も送られてきたデータがどの処理のデータか判断できず、画面に表示できなかったため、ネットワークを通して対戦するゲームを実現する事ができなかった。

表 9: ネットワーク関連の処理

```
 ・・・・・・・・・・・・・・・・
/*ボタンが押された時入力された ipaddress に接続する処理*/ public void pochi(Object sender){
//System.out.println(”Pochi”);
String tmp;
ip_another=ip_area.stringValue();
init_net();
//send.add(tmp);
}
public void init_net(){
this_network.connect(ip_another);
//net_mode=true;

Vector v=this_network.get_listv();
if(v.size()==0){
chMod();
}
}
 ・・・・・・・
/*コネクトが失敗したとき呼び出される
net_mode と local_mode を切り替える
/ public void chMod(){
if(net_mode){
net_mode=false;
}else {
net_mode=true;
}
}  ・・・・・・・・・・・
```

9.8 今後の課題

チーム作業の一番大事な”まとめる”作業ができなかったのがとても残念である。ゲームの実装とメッセンジャーの実装はできたのだが、それをつなげる事ができなかったので 今後は動作するネットゲームを作成するのが課題である。