Literate Programming

...a quick introduction
Documentation
is GOOD!
er..., Why ?
It Help Us Understand!
Architecture
```c
#include <ncurses.h>

int m[256][256], a, b;
char l="\176qxl", q = 'q', k = 'k', w, u, n, Q;

W
int u, int v;

WINDOW *w; char *l="qxl", "q", "x", "l", "u", "n", Q;
]
int u, int v;

;

if (v != m[u][v] & 64) m[v][w] = 64; else m[v][w] = if (m[u][v] & 64) break;

m[w][v] = if (m[u][v] & 64) break;

main(V, char *C[1])

FILE *f = fopen(V == 1 ? "arachnid.c" : C[1], "r");

int x, y, c, v = 0;

initscr();

curs_set(0); noecho();

keypad(stdscr, TRUE);

w = newpad(300, 300);

for (x = 255; x >= 0; x--)

for (y = 255; y >= 0; y--)

m[x][y] = 16;

a = b = c = 0;

x = y = 1;

do {v++; mvwaddch(w, y, x, m[x][y] & 32 ? m[x][y] & 16 ? acs_map[l[m[x][y] & 15]] : 46 : 32);
c == 0 || c == 119 || ! (m[x][y - 1] & 16) ? y-- : 0;

c = 97 && ! (m[x - 1][y] & 16) ? x-- : 0;

c == 100 && ! (m[x + 1][y] & 16) ? x++ : 0;

if (c == 3 - 1 + 1) {endwin(); return(0);} x = a ? <57 >S 5 ? a - S - 5 : (a == 0);

x -= a;

S-5 a <255 - S - 5 a = 27 a + S

-5: (a = 256 - S) : 0;
y-b <57 b <L-5 b = L-5 ; (b = 0) : 0;
y-b <L-5 b <L-5 * 2 ?

b +=

L-5 : (b = 256 - L)

0; printf(Q, v);}
```

Code
Problem!
Divide between code and documentation
Synchronisation is hard
Solution
Mix Code and Documentation
example
JavaDoc
Doxygen
/** Processes a channel of an audio buffer. Size frames are processed. */
@todo description no pos sequential processing
*
* @param data audio buffer to be modified
* @param channels number of audio channels in the passed buffer
* @param channel index of the audio channel to process
* @param size number of frames to be processed
*/
virtual void process (SampleType* data, unsigned channels,
                     unsigned channel, unsigned size) = 0;

/*******************************************************************/
/** Returns the sample rate. This is used to convert time-based */
/** parameters to sample positions. */
/** */
unsigned sampleRate() const
{
    return rate;
}
Use Metadata (written in comments) to generate documentation
view::MMPanel Class Reference

Inheritance diagram for view::MMPanel:

- view::MMPanel
- view::FilterPanel
- view::LibraryPanel
- view::PlaylistPanel

List of all members.

Public Member Functions

- function get title ()
- function set title (str:String)
- function redraw ()

Static Public Attributes

- static var LINKAGE:String = "com.pluswoe.presentatfx.view.MMPanel"

Detailed Description

A UI Panel with a Macromedia Flash IDE style title bar. Uses the GhostWire Panel component (include intrinsics for AS2).

Author:
Richard Leggett

Member Function Documentation

function view::MMPanel::redraw ( ) [inline]
Redraw the MMPanel

function set view::MMPanel::title ( str:String ) [inline]
Setter for title TextField
Good for API, Not so good for programs
Literate Programming
Literate programming is the art of preparing programs for human readers.
Don Knuth
• 1979, DOC
• 1979, DOC
• 1981, WEB & TeX
• 1979, DOC
• 1981, WEB & TeX

Not exactly new...
Approach
JavaDoc/Doxygen approach

Master Document

CODE + META

DOC

PROGRAM
Literate Programming approach

Master Document (weave) 

DOC + CODE (tangle) 

CODE
Same ?
No !
JavaDoc

CODE + DOC
Literate Programming

DOC + CODE
example
This is my toy program. We start with a main function:

```c
int main(int argc, char** argv) {
    int result = 0;
    result = 2 + 2;
    printf("result: %d\n", result);
}
```

The simple work done is very simple:

```c
result = 2 + 2;
```
This is my toy program. We start with a main function:

```c
int main(int argc, char** argv) {
    // Declarations
    // Work
    // Print
}
```
@ \subsection{Printing}
The [[print]] simply prints the result of the [[work]]:

```c
<<print>>=
printf("result: %d\n", result);
```

@ Let’s not forget to add the [[result]] variable to our declarations:

```c
<<declarations>>=
int result = 0;
```

@ And include stdio:

```c
<<includes>>=
#include <stdio.h>
```

@ \subsection{Work}
The [[work]] done is very simple:

```c
<<work>>=
result = 2 + 2;
```
```c
#include <stdio.h>

int main(int argc, char** argv) {
    int result = 0;
    result = 2 + 2;
    printf("result: %d\n", result);
}
```

notangle test.nw > test.c
1 Introduction

This is my toy program. We start with a main function:

(*\equiv\)
\(\langle\text{includes}\rangle\equiv\)
\begin{verbatim}
int main(int argc, char** argv) {
    \langle\text{declarations}\rangle
    \langle\text{work}\rangle
    \langle\text{print}\rangle
}
\end{verbatim}

1.1 Printing

The print simply prints the result of the work:

\(\langle\text{print}\rangle\equiv\)
\begin{verbatim}
    printf("result: %d
", result);
\end{verbatim}

Let’s not forget to add the result variable to our declarations:

\(\langle\text{declarations}\rangle\equiv\)
\begin{verbatim}
    int result = 0;
\end{verbatim}

And include stdio:

\(\langle\text{includes}\rangle\equiv\)
\begin{verbatim}
    #include <stdio.h>
\end{verbatim}

1.2 Work

The work done is very simple:

\(\langle\text{work}\rangle\equiv\)
\begin{verbatim}
    result = 2 + 2;
\end{verbatim}
Characteristics

- non-linear
- explain your program top-down or bottom-up
- introduce code step by step
- write for humans, not computers
Is it good?

- not very well adapted to OOP
- no good GUI tools
- still, very interesting concept...
- AOP, TDD, etc
Leo

http://webpages.charter.net/edreamleo/front.html
Links

- http://www.literateprogramming.com/
- NoWeb: http://www.eecs.harvard.edu/~nr/noweb/
- better example: http://moonflare.com/code/select/index.php
- Leo: http://webpages.charter.net/edreamleo/front.html
Images sources:

allbasset.com

innoxious http://flickr.com/photos/46922409@N00/308920352


Aleksi Aaltonen http://flickr.com/photos/aleksiaaltonen/1361437606/

http://www.jugglerdave.com/JugglerDaveRings.jpg