Wonder Swan で 4096〜ハイカラー表示を

みかり*

2003 年 10 月 27 日

概要

本稿では Wonder Swan でわんだらぁ氏の天然色ツール [わんだらぁ亭] を超えた発色を行うための考察を行う。

なお、本稿はバンダイ、Qute よりの正式な情報を利用して書かれたものでないので所々に誤りや不正確な個所があると思われる。(得に DMA 関連に間違いがある可能性が高い) もし誤りに気づいた場合は筆者 (mailto:mikarim@m18.alpha-net.ne.jp) までご連絡願う。

本稿中のサンプルコードは Light Macro Assembler*1[Light Macro Assembler(LASM)] でアセンブル可能なもので、掲載においてシンボルを書き替えているわけでない。*2上に、LASM の拡張機能 (ラベル生成時オフセット加算機能) を利用しているため、他のアセンブラを利用している方は注意されたい。

目次

---

* ☆はハンドルに含まない。

*1 LASM と略される

*2 すなわち、LASM においては日本語をシンボルに利用可能である

1 表示ハードウェアの構成

略

2 原理

ワンダースワンには水平帰線を検知するタイプの割り込みがあるためそれを利用してパレット 01-0F,11-1F,21-2F,31-3F,41-4F …… [*3]の縞模様で描かれた BG 画面を表示し、パレット割り当てをライン毎に書き替えることで、多色表示（4096 色同時発色）を目指す。

なお、割り込みの発生は画面表示を OFF にし[*4]ボーダーカラーのパレットを LCD 描画ライン表示 I/O のポーリング、もしくは、HSync 割り込み中に切り替えたところ、その位置で色が切り替わっていたことより、

1 ライン前の、LCD の左 1/3 辺りを描画しているタイミングで発生するものと推察される（スワンクリスタル調べ)。

3 実験

まず、画像データーを本体内蔵の、IRAM[*5]に設置 し、1 ラインにつき、最大でいくらのパレットを書き替えることが可能であるかを調査することとした。

3.1 CPU のみで処理

割り込み中に CPU でパレットの描きこみを行ったところ、32 パレット/HSync を超えると暴走[*6]した。V30MZ のデーターシート [V30MZ] によると、MOV reg*mem 命令の実行速度は、1 クロックであるということなので（rep movs は 7＋5 クロック ×cx） 1 ライン 256 クロックであることより、100 パレット程度は書き替えが可能と考えていたのだが実際にはその 1/3 程度の性能しか発揮できなかった。理由は不明である。

次に、割り込み処理ルーチンを IRAM においてみたのだが性能に向上は見られなかった。これはコードを IRAM に置いた場合と、外部 FLASH ROM に置いた場合の命令フェッチに速度差が無いことを意味していると思われる（実際にはあるのかもしれないが、3.072MHz の CPU 動作クロックでは顕在化しないということであろう）

*3 即ち透明でないパレット色
*4 ボーダーカラーのみの表示
*5 本体内蔵 SRAM の事 internal RAM、Color スワンでは、64KB ある
*6 割り込み処理中に再度割り込みが発生し、本体の処理が実行できなくなった

3.2 DMA 転送

cygne[cygne, 用語集も参照] の作成過程での調査結果をまとめたドキュメント [cygne, WStech] が公開されているのを以前入手していたので参照し、DMA 転送を試みた。すると、Hsync ポーリング にて 84 × 2 バイトの IRAM to IRAM 転送が実現できた。85 × 2 バイト以上にした場合同期が取れなかった[*7]。ただし、この事は 84 ドット/ラインの表示が出来ることは意味しない。何故なら各パレットの Index0 はカラーでは透明色となっているからである。[*8] よって、最大で 79 色の表示可能パレットを書き替えることが出来るということになる。

次に CPU 転送と、DMA 転送の組み合わせを試みた。即ち、パレット書き換えルーチンの先頭で DMA 転送を設定しそれと同時に CPU 転送もりようすることで、79+21=100 ドット/ラインの表示に挑戦した。

ところが割り込み、もしくは、ポーリングルーチン内で CPU によるパレット書き込みを行うと画像が乱れてしまった。

どうやら、CPU でのアクセスが DMA の実行遅延につながっているようだ。検証していないが、DMA で書き込みを行っているブロックと同じブロックへの書き込みが発生することで DMA の処理に遅れが発生するのだろう。[*9]

DMA 転送サイズを減らしてみたところ、問題無く実行できたのだが、84 パレットより転送量が少なくなってしまったため本末転倒といえる。これより、WonderSwanCristal においては 84 パレット (実効 79 パレット)/ライン を超えるパレット書き換えは不可能という結論が導き出された。

これ以上のパレットを書き替えることに成功された場合はぜひ、実行手段を添えて公表してほしい[*10]。

3.3 補足：DMA 転送について

DMA 転送は、I/O ポート 40h-48h を制御することで利用可能である。

```
outb    macro   byteAddress,data
    mov al,data
    out byteAddress,data
    endm

outw    macro   byteAddress,data
    mov ax,data
    out byteAddress,data
    endm
```

とのマクロ定義がされていた場合の利用例は次のとおりである。

[*7] CPU 転送と違い割り込み処理事態は終了しているのでボタンでシェルに戻ることは可能である。

[*8] 下層背景 (BG1) は透明色を持たないという仕様であったら良かったのだが（天然色ツールでも、256 色利用可能となる)。現在のボーダーカラーと同じ事は、16 の倍数のパレットに同じ値を設定することで可能となるため上位互換である。しかし残念ながらそのような仕様ではない。

[*9] この辺りかなり怪しい、つまり不正確である恐れがあるため、できるならば追試を願う。

[*10] ただしクロックアップを施すなどの一般に販売されている WonderSwan そのままでは不可能である手段は除く

```
    outw    40h,8000h   ; 転送元アドレス（IRAM:8000）
    outb    42h,0       ; ／

    outw    44h,FE00h   ; 転送先アドレス（IRAM:FE00）
    outb    43h,0       ; ／

    outw    46h,512     ; 転送バイト数

    outb    48h,80h     ; 転送開始

Lp: in  al,80h      ; 転送終了待ち
    test    al,80h      ; Read して bit7 == 0 なら
    jnz lp      ; 処理完了
```

42h、43h にはそれぞれバンクアドレスを指定するらしい。バンク切り替え I/O に指定する値と同じものを指定するとよいのではないか[*11]。SRAM や、Witch カートリッジ の Flash を含んだ転送の方法について調査されたかたが居られたら連絡を乞う。（恐らく、ファイルシステムのコードを解析わかるのではないかと思うが、やるわけにも行かず……。ブート部分を調べると面白いことが判るのではないかと思っているのだが、一文の存在が残念である[*12]。）

この時の転送力であるが、16384 バイトの転送を実行し、その間にラインカウンタが何ライン進むか、即ち Hsync 何回で転送が完了するかを調べたところ、16K の転送で、64 ラインカウンタが進んでいた（データー転送量は先のパレット転送量より、適切と思われる値を選んだ）よって、16384 / 64 = 256Bytes/Line であり、1Bytes/Clock となる。

なおコードを Flash に置いても、IRAM に置いても、動作に影響はなかったため読み出しアクセスは DMA と衝突しないと予想される。

4 実践

以上の実験結果より、WonderSwanCrystal 上では、32*32 チップの BG 表示領域の上に最大で 79*144 の大きさの天然色表示ウィンドウを設けることが可能であることがわかった。ただし、通常の BG と併用する場合は 1 ライン先行処理がほぼ不可能であるため表示位置によってはちらつきが出る[*13]。

1 ライン先行処理に関しては後段を参照のこと。

[*11] バンクが 64K 区切りであるので、物理アドレスの、16-23 bit なのだと思うのだが詳しくは未調査

[*12] BANDAI との契約上か

[*13] 4 パレット以上を利用する特異点にスプライトを配置することで可能となる筈。

```
DMA 転送力ベンチ proc    far
    ; IRAM でプログラムを実行中の、IRAM IRAM 転送速度を調べる。
    開始待ち:
        in  al,DisplayLine_b        ; < in al,02h
        cmp al,20
        jne 開始待ち

        outw    40h,0x8000          ; src ofs
        outw    42h,0000h           ; ah:al = dst:src bank
        outw    44h,0xB000          ; dst ofs
        outw    46h,16384           ; length
        outb    48h,80h             ; start

    完了待ち:
        in  al,48h
        test    al,80h
        jne 完了待ち

        in  al,DisplayLine_b
        mov ah,0
    ret
DMA 転送力ベンチ endp
```

図 1: DMA の単位時間辺りの転送量を調査するためのコード

5 応用

5.1 1ライン先行処理

全画面の表示を行うためにはパレットを全て消費してしまうが、実験より、84 パレットの書き換えが最大と判明したため 6 つのパレットブロックを 2 組持ち、それぞれを交互に書き換える事で、1 ラインだけパレット書き換えを先行処理することが出来る。

この処理のメリットは、処理タイミングに余裕が出来るという点である。

先行処理を行わない場合は、水平同期割り込みのタイミングの関係で同じ処理を行っていても画面左と画面右で 1 ライン上下にずれたり、表示位置によってちらつきが広範囲で発生したりする。しかし、1 ライン分の余裕をもって処理することによって、表示中のラインに対しては変更が加わらないこととなるので、1 ラインのずれや、パレット書き換えに表示が追いついたことによるちらつきは完全に抑制されることとなる。

以下に書くことは、10/21 現在において実験が完了していない。Hsync 時のスプライト座標の書き換えが画面に反映されるか、HSync 時のスクロールレジスタの書き換えのタイミングが随時であるか、それともライ

ン描画開始時にラッチされるのかは未調査であるため、現時点では机上の空論に過ぎない。

もし、スプライト座標が全て、VBlank 時にラッチされる場合や、スクロールレジスタへの書き込みが随時反映されてしまう場合は、本セクションで記述したことは実現できないことになる。

5.1.1 スプライトを使う場合

1ライン先行処理を行うには、パレットブロック A で描かれた縞模様と、パレットブロック B で描かれた縞模様が1ライン毎にあれば良い。その際、パレットブロックが4つ必要となる特異点が発生するが、Swan の画面は 2BG+Sprite である為に通常では対応できない。

ここで通常の BG 表示も行いたい場合（天然色ウィンドウ外に文字や枠などを表示する場合）は対応方法は1つしかない。即ち Sprite を利用して特異点を補う方法である。

特異点は最大で5箇所に発生し、1箇所に付き Sprite を2つ消費するため、10 枚のスプライトを用意して4ラインの処理毎に表示座標をずらすことで対応が可能であると思われる。（ただし、表示ブロックの下端ではウィンドウを超えてスプライトを移動させないよう注意が必要である）

5.1.2 垂直スクロールさせる場合

通常の BG 表示をまったく行わない場合は別の対応方法がある。BG を2段利用しそれぞれに別のパレットを割り当てた縞模様を用意する。

即ち、ライン 0-7 を パレット 00-5F で構成された縞模様ライン 8-15 を パレット 60-7F で構成された縞模様とし、初期の垂直スクロールラインを 7 に設定する。そして、2ライン処理したところで、スクロールラインを 9 に更新する。これによって、1つの 8x8 ブロック中に複数のパレットをラスタ単位に混在させることが可能となる。

あとはスプライトを利用した場合と同じようにパレットを書き替えることで対応が可能となる。

5.2 表示領域を大きくする

通常は最大でも、横幅 79 ドットの領域しか取ることは出来ないが、縦縞を構成するパターンを2ドット幅で描画することによってドット数は変化しないが最大で、158 ピクセルまで表示領域を広げることが出来る。当然横幅3ピクセルで縞模様を描けば全画面をサポートできるわけだが恐らく画質には相当のダメージがあることと予想される。[*14]

横幅を2倍に拡大したサンプルをご覧いただければ判るように、このくらいの表示サイズと画質であればタイトル画面に用いるのには最適であろう。また、サイズを2倍にしたことにより、1ライン先行処理を行わずとも適切な速度で書き換えが発生することとなるので[*15][*16][*17] 、表示可能領域がほぼ全域となる利点がある。

[*14] 実記では確認していないがペイントソフトで加工したところかなりギザギザしていた

[*15] 先行処理を行わない場合、1ラインの描画と、パレット書き換えの速度がちょうどつりあうのが理想 (Hsync の発生は、画面右端でなく、左 1/3 程度を描画中となるため、書き換えが遅い場合ちらつく)

[*16] 1dot/palette の場合、ドットを1つ書きかえるのに、2クロックかかるため、パレット書き換えを表示がすぐに追い越してしまう。しかし、2dot/palette とすると、1パレット転送の時間がほぼ、液晶1ドット表示の時間に等しいためパレット書き換えの追い越しは発生しないことになる

[*17] 各種タイマーの速度と、CPU 速度や描画速度の間には奇妙な符合がある。ということは、スワン内部では、1つのクロックを分周あるいは逓倍してつかっているのだろう（分解してパターンを追うなどの手法で検証された方はおられるだろうか？）

5.3 低解像度表示

1ライン先行処理を応用して、低解像度で全画面の天然色表示を行うことが出来そうである。

5.3.1 112 x 114

仮想 VRAM の大きさは、80*114*2(23040 Bytes) [*18]

この場合、左端の 144*114 もしくは、152*114 を天然色用として利用し、右端には通常のタイルを置くことになるだろう。転送速度が、84 パレット/ラインであることより、これ以上の領域を利用することは不可能である。

ノベルタイプのゲームには一番ふさわしいのではないかと思われる。[*19]

5.3.2 112 x 72

仮想 VRAM の大きさは、128*72*2(18432 Bytes)

まず、2x2 ドットにすることで、先行処理を行うならば、2 ラインに付き、119 色[*20]を書きかえることが十分に可能である。(先行処理を行わない場合は書き換えが表示に追い越されるため実現不可能)

よって、112*72 ドットでの全画面天然色表示が可能となると予想される。ただし、DMA 転送を使う関係上、仮想 Vram は、15 ドットおきに未使用領域が存在するという奇妙な構成となってしまう。なお、BG は天然色表示に総動員となるが、スプライトは通常どおり利用できるので、文字などを画面に重ねたいといった場合には、仮想 VRAM に直接書きこむか、高解像度が必要な場合はスプライトを並べて表現することとなる。

全画面がほしい場合はこのモードを用いることになるだろう。

特異点をスプライトのみで解決できると良いが処理負荷が高いかもしれない。

5.3.3 74 x 48

仮想 VRAM の大きさは、80*50*2(8000 Bytes) [*21]

ここより、IRAM でダブルバッファが利用可能となる。

基本ドットを 3x3 にしただけで原理は同じである。15 パレットづつ、3 ラインに分けて転送、即ち、15*3=45 ドット = 90 バイト を DMA で転送し、残りのパレットを、10,10,9 に分けて、CPU 転送することで、リニアな仮想 VRAM を得ることが可能であるが、描画時に 15 ライン毎の空きを考慮し表示は全て DMA で行うのと、表示時に CPU 転送を利用して変換するのとではどちらが負荷が高いだろうか?

拡大回転縮小を駆使したアクションゲームに向いているだろう。

特異点が少なくなっているため、全てスプライトで解決し、BG2 をまるまる開放することも、このサイズからは可能になるだろう。

[*18] 格納可能なサイズを満たすきりの良い大きさを選んでいる

[*19] スワンのボタンが完全に左右対称だとしたら、左端に画像を表示右端に左 90 度に回転した文字を表示することで定番の画面配置が出来たのだが。せめて、AB でなく、ABCD ボタンだったら……

[*20] 16(色)*7(パレットブロック)+7(112 を 15 で割ったあまり) = 119 最後のパレットブロックの Index 0 の分は、最初のパレットブロックの Index 0 で相殺

[*21] Y 方向は、NEC 98x1 ローレゾリューション画面にあわせて、2 ライン余分に取っている

5.3.4　56 x 36

基本ドットサイズ、4 x 4、仮想 VRAM の大きさは、64*40*2(5120 Bytes)

DMA 転送のみ、もしくは、CPU 転送を用いて実用負荷で、リニアな仮想 VRAM を持つことが出来るようになる（と思われる）のがこのサイズからである。パレットを 全て使い切って 8 ライン先行処理を行う方法と、スクロールやスプライトとの併用で 4 ライン先行処理を行う方法がある。（8 ライン先行処理のほうが処理負荷は軽いがスプライトに使えるパレットがほとんど残らなくなることを考えると、4 ライン先行処理を採用することが懸命であろう）

5.3.5　28 x 18

基本ドットサイズ、8 x 8、仮想 VRAM の大きさは、32*20*2(1280 Bytes)

もはやほとんど冗談としか思えないが、504 ドットしかないため、計算量がかなり少なくてすむ。よって、動きで度肝を抜くような作品[*22]が作れるならばこれもありだろう。このサイズであれば特異点は発生しないため、BG2 が丸々利用可能なのも利点である。

5.4　ウィンドウ内スクロール

天然色ウィンドウの内部は仮想 VRAM の開始アドレスを移動させることでスクロールが可能。仮想 VRAM がリニアな場合は X,Y それぞれ 1 ドット単位でのスクロールが可能だが、BG 画面と違い左右はつながっていない（上下のみ連結可能)。

仮想 VRAM がパレット形式の場合は、横１５ドット単位でのみスクロール可能となる。この場合でも、左右は連結できない。（ただし、極低解像度画面でならば左右の連結処理を行うことも可能かもしれない）

5.5　ハイカラー表示

仮想 Vram を２面用意し、隣り合う輝度のドットを垂直帰線で、切り替えて表示することで 赤、青、緑、それぞれ 31 段階の表現を行うことが可能となるとおもわれる。よって、31*31*31 = 29791 色 のほぼハイカラー の表示が可能となる。[*23]

ただし、フリップ用の画像情報を保持するために仮想 VRAM が２面以上必要となるため、112 x 72 以上の解像度の場合、仮想 VRAM を、SRAM [*24]に確保できるようにする必要がある。DMA 機能の更なる調査が必要であろう。

6　IL 化

IL 化にあたっては幾つか解決せねばならない問題がある。IRAM の確保問題とスタートアップルーチン、BIOS 管理割り込み処理との共存である。

なお、この IL でサスペンドレジュームに対応の予定はないのであらかじめご了承頂きたい。

[*22] MZ 版スペースハリアー？

[*23] 32768 色も、65536 色もいっしょくたにしてハイカラーと呼んでいるのだから３万色弱でもハイカラーと称しても大丈夫ではなかろうか

[*24] プロセス２用の SRAM に置きたい

6.1 IRAM 確保

中解像度の場合、仮想 VRAM の大きさは、128*72*2(18432 Bytes) となる。また、コード作業用ワークも IRAM 上に必要となる。

そのため、IRAM を確保する必要があるのだが、それは、連続かつ OS が利用しない領域である必要がある。カートリッジが基本的に初代とカラーの両対応ということ、そのような大きな領域は、BIOS ファンクションでは確保できない（Cマガジン記事による情報）ということ、ファンクションで確保できる領域が２K 程度という記述より、カラーで拡張された 0C000h-0FE00h の約 16KB の領域が未使用であると推察される。また、1024 キャラを利用しないことで 約 32KB の連続領域を 利用できることとなる。

さらに、各種スタートアップルーチンで BG/SP の配置テーブルに指定されている領域も使用されないと考えられるため、それらの領域も利用可能であると思われる。

よって、IL 化にあたっては当該領域にワークを確保することにする。

ただし、BIOS の IRAM 確保ファンクションの挙動に関しては一切実験を行っていないため、それが本当に可能かについては保証の限りではない。（SOUND IL の波形定義テーブルとぶつかっている可能性もある）

IRAM 確保ファンクションについて確保サイズ、確保オフセット、総確保量等についての、試験の実施が望まれる。[*25]

6.2 スタートアップルーチン

スタートアップルーチンは、DISPLAY_MODE_JAPANESE2 相当のものを利用することを想定している。他のモデルでは動作を保証しない。（動作未確認であるだけで、実際には問題ないと思われる）

6.3 BIOS の割り込み処理

本処理では、WonderSwan の能力を極限まで使う必要がある為、OS の管理する割り込み処理は一切利用しない。ゆえに、sys_wait() 等は当然のように利用できなくなっているので、割り込みや、タイマー、キー操作チェックをユーザープログラムで利用する場合は IL の提供するサービスを利用する、もしくは直接ハードを叩くことが必要となるため、プログラムの難度が上昇することが考えられる。

これは互換機能を IL もしくはスターティックライブラリとして提供することで解決可能だと思われるが、開発負荷が高くなることが予想される。ライブラリは共同開発が出来れば理想だろう。

6.4 音楽表現

SYSTEM の提供する割り込みが全て利用できなくなるということは、BIOS の一部が利用できなくなるだけでなく、当然従来の IL も利用できなくなるものが出るということである。その代表格が SoundIL であろう。殆どのゲームが音楽表現をこれに頼っているため、現状では音楽表現が不可能となるため、何らかの代替策が必要であろう。割り込のフック部分をユーザーが記述できる、互換 API を持った IL の開発が望まれる。（WW2002 の Hamming Cat なら利用可能であろうが未確認）

[*25] サポートに問い合わせたところ、IRAM の後半領域をユーザープログラムで利用することは構わないが、その領域が OS/BIOS の管轄外であるかは保証しないとのこと

7 用語集

天然色 4096 色同時発色のこと。古の名機 FM77AV が 4096 色同時発色可能であり、総天然ショック とのコピーで宣伝していた。それが わんだらぁ氏のツールの名称に採用されたことにより、現代に復活した古語。[*26]

ハイカラー 32768 色 (32*32*32) もしくは、65536 色 (32*64*32) 表示のこと。32*32*32*2(X68000) は含まれないらしい。

厳密に言うと 本稿で実現できるのは、31*31*31 の 29791 色であるので、ハイカラーではないが、天然色の上（ハイ）と言うことで意味は通る（ということにしておいてほしい)。

cygne オープンソースの WonderSwan エミュレーター。[*27]

Witch のライブラリ、その他の逆アセンブル等による解析は仕様承諾に触れるが、このようなエミュレーターを解析する分には仕様承諾には触れないため、ソースが公開されているエミュレーターを探していたときに発見した。本実験では、ここで同時に公開されていた、WStech21.txt というドキュメントが大いに役に立った。[*28]

狭義のリバースエンジニアリングが禁じられているといっても、解析者とその利用者が別であって、解析者が直接の解析情報を利用者に伝えず、利用者は解析者が再構成した情報を元にコードを作成していることを証明することによって間接的に、利用者のコードが解析元のコードのデッドコピーでないと証明できるのであれば法的には問題ないだろうが、倫理的問題はあるだろうし、自身が解析者になってしまうと、解析結果を利用した応用を発表できなくなるのが悩ましいところである。

リバースエンジニアリング 広義には製品から仕様を起こすこと。通常は仕様から製品を作成するわけでその逆であるので、リバースである。

狭義にはソフトウェア製品を逆アセンブルや、バストレース等の手法で解析すること。

ソフトウェア製品、あるいはフリーソフトであっても、

「本ソフトウェアの逆アセンブル、逆コンパイルなどによる解析を禁じます」

という文言があるが、筆者はこれはまったくのナンセンスであると信じている[*29]。[*30]

なお、ここに書いたことは 一般論 であって、特定の個人、商品、団体を指したものでは無い。

参考文献

参考文献でなく、参考サイトですが。

[わんだらぁ亭]

http://asagao.sakura.ne.jp/ wanderer/index.html

天然色ツールに関しては、こちら

http://asagao.sakura.ne.jp/ wanderer/wwitch/color/index.html

[*26] 77AV は、4096 ショック、で総天然ショックは、77AVEX だったと思うのは筆者の記憶違いか。詳しい方、訂正求む

[*27] これらを利用して、Witch のデバッグ環境が作れるととても便利そうだが……

[*28] WSTECH で検索したところ、WStech23.txt というもの、も存在するようだが、内容は未確認

[*29] 明確に書面を交わして行った契約であるならばまた別の話である。

[*30] 信条とそれを実際に行うかは別

[Light Macro Assembler(LASM)]

多摩ソフトウエア有限会社 [http://www.tamasoft.co.jp] の現役商品で、16800 円。シンボルに日本語文字が利用でき、数々の拡張機能が組み込まれているのでとても便利。紹介ページは、こちら

http://www.tamasoft.co.jp/lasm/index.html

似たような特徴をもつ C コンパイラはこちら

http://www.tamasoft.co.jp/lc/index.html

[V30MZ]

データーシートはこちらにある模様。

http://www.pocket-viewer.ru/developer/files/16bitmpcv30mz.pdf

V30 の 3 倍くらいは速そうに思われるがベンチを取ってはいないため、正確なところは不明。

[cygne]

http://cygne.emuunlim.com/

技術文書はこちら

http://cygne.emuunlim.com/files/wstech21.txt

[Humming Cat]

土居弘幸氏製作の使用割り込みをユーザーが選択可能なサウンドドライバ

WWGP 2002 出展作品

http://www.ne.jp/asahi/myu/doi-hiro/wwitch/