Project SMD

8. Let's read and print the GRP file. (One)

why do you ask
2013. 9. 25. 13:53

This document is a course-style log of developing a StarCraft viewer .
Since I am also in the position of learning, what is the point of calling this a 'course';
This is a project to make a map editor, realizing that making a map editor involves under standing the overall structure of the game.
I started with ..., but I don't think there's any need to even make a map editor, and I'll hav e to use the unprotect technology someday anyway,
I'm just trying to make an upgraded version of TriggerViewer2. Then the course will start again.

http://blog.naver.com/whyask37

---------- ----------

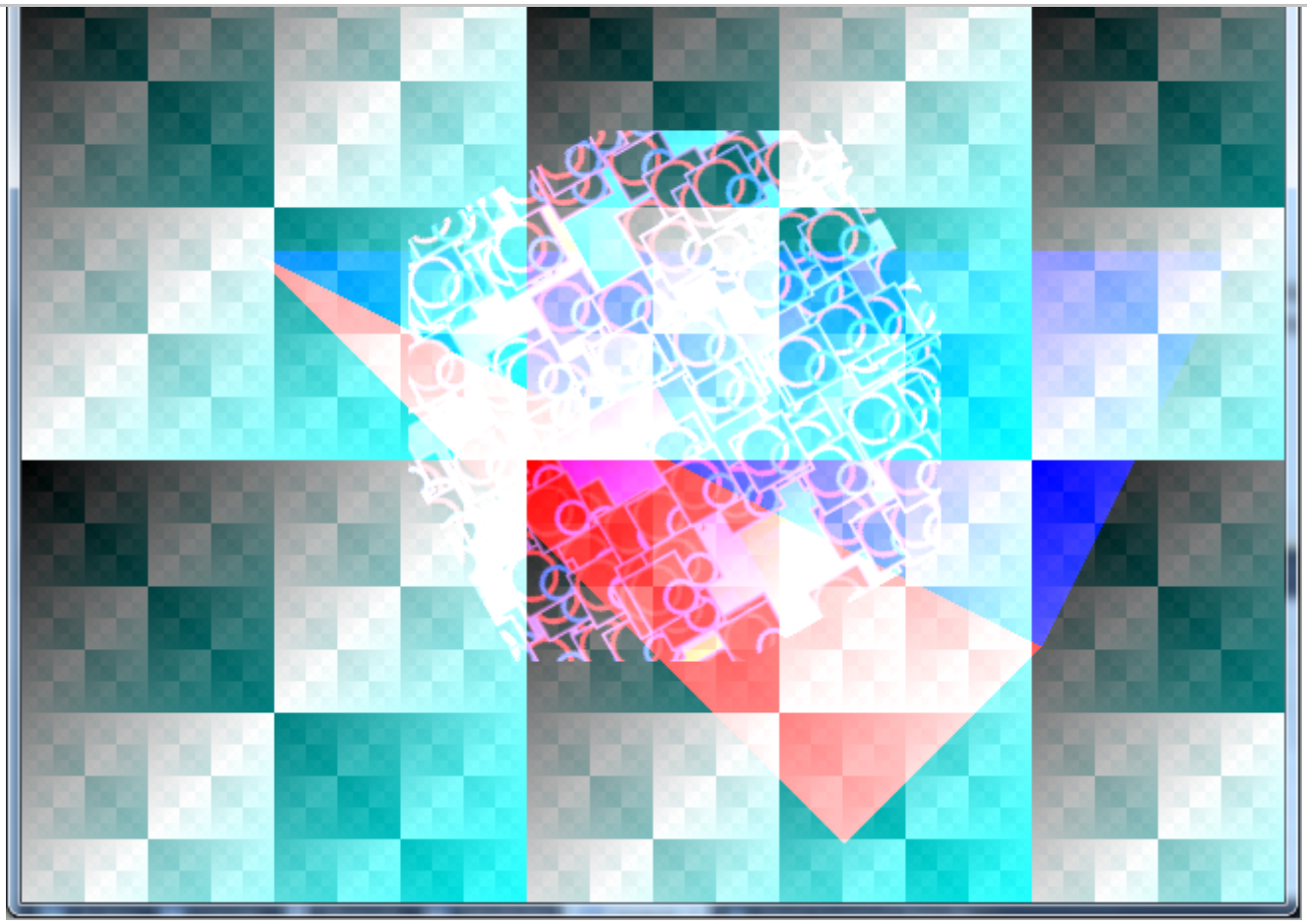
What is a GRP file?
Unit sprites that are printed except for terrain are all grp.
That's easy, right?

In the future, we will use the gui library to write simple examples.
I made it as a study in the past, and it's pretty useful.

It doesn't output a gui, it just supports pixel rendering.

You can roughly do something like this

whyask37's blog



That was the moment when I found software rendering incredibly useful;

I think I need more explanation about the grp format.
Source: http://www.modcrafters.com/wiki/index.php?title=GRP_files
출처 : http://www.staredit.net/starcraft/GRP_Image_Format

grp 파일은 여러개의 프레임을 가진 이미지 파일입니다.
제일 간단한 예제로 grp 파일을 분석해서 안에 있는 프레임 모두를 추출하는 프로그램을 만들어보겠습니다.

ShadowFlare님의 grpapi는 쓰지 않도록 하겠습니다. 직접 다 만들꺼예요.

일단 GRP 포맷에 대한 이해부터 시작합시다. GRP 포맷에 대한 이해를 위해서 우리의 **귀요귀요미한** 저글링을 수동으로 렌더링해봅시다.
뭔가를 이해하는 가장 좋은 방법은 역시 수동으로 일일이 다 해보는거죠!

2. GRP frame header
3. GRP line offset table
4. GRP line data

하나하나 분석해나가죠.

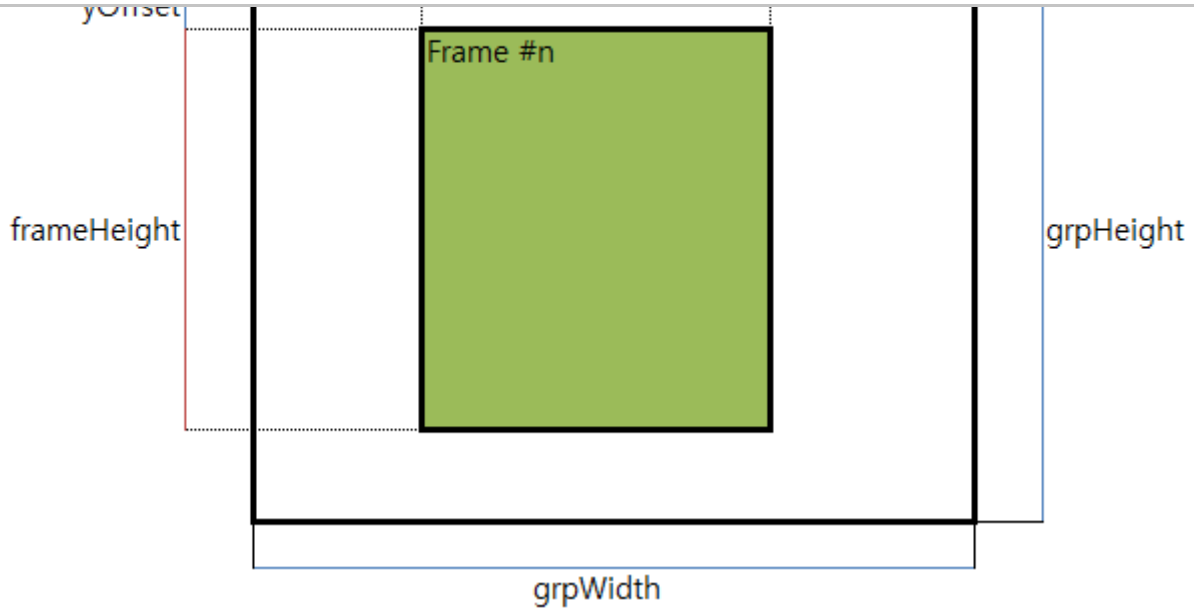
GRP 헤더와 GRP 프레임 헤더는 다음과 같은 포맷으로 되어있습니다.
(네이버의 표 기능은 정말... 뭐같아요)

```
struct GRPHeader { uint16 frameCount; uint16 grpWidth; uint16 grpHeight; }; struct GRPFrameHeader { uint8 frameXOffset; uint8 frameYOffset; uint8 frameWidth; uint8 frameHeight; uint32 lineTableOffset; };
```

GRPHeader::
frameCount : 총 프레임 갯수
grpWidth: grp 파일이 차지하는 가로 길이
grpHeight : grp 파일이 차지하는 세로 길이

GRPFrameHeader::
frameXOffset : 프레임의 X 오프셋
frameYOffset : 프레임의 Y 오프셋
frameWidth : 프레임의 가로 너비
frameHeight : 프레임의 세로 높이
lineTableOffset : 이미지 데이터 위치

whyask37's blog

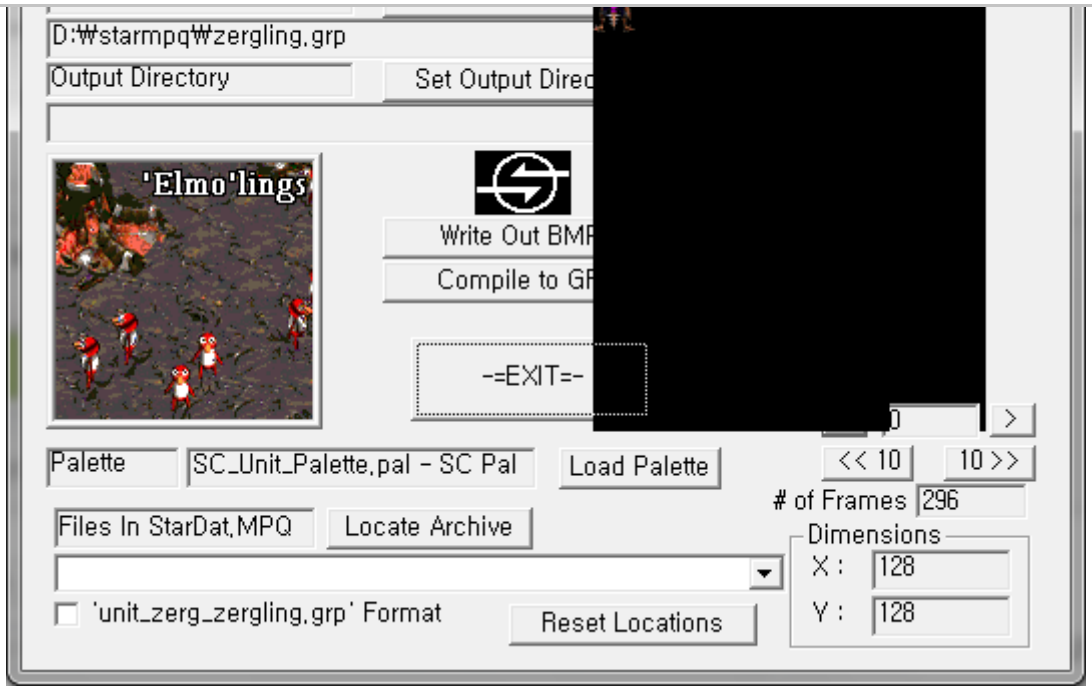


//엑셀은 위대한 발명품입니다. 엑셀만세. 위에 그림도 엑셀로 그린거예요.

대충 저런 식으로 출력해주면 됩니다.

우리가 출력해줄 귀요귀요미한 저글링을 미리 보도록 합시다.
GRP를 에디트해주는 좋은 툴로 RetroGRP가 있습니다.
(는 그냥 아무거나 갖다쓰는거예요. 툴 비교는 안해봤어요)
Download from http://broodwarai.com/wiki/index.php?title=Tools

그럼 이걸로 Stardat.mpq\unit\zerg\zergling.grp를 열어보겠습니다. 팔레스트는 RetroGRP 폴더 안에 있는 SC_Unit_Palette.pal 를 한번 써보도록 합시다.

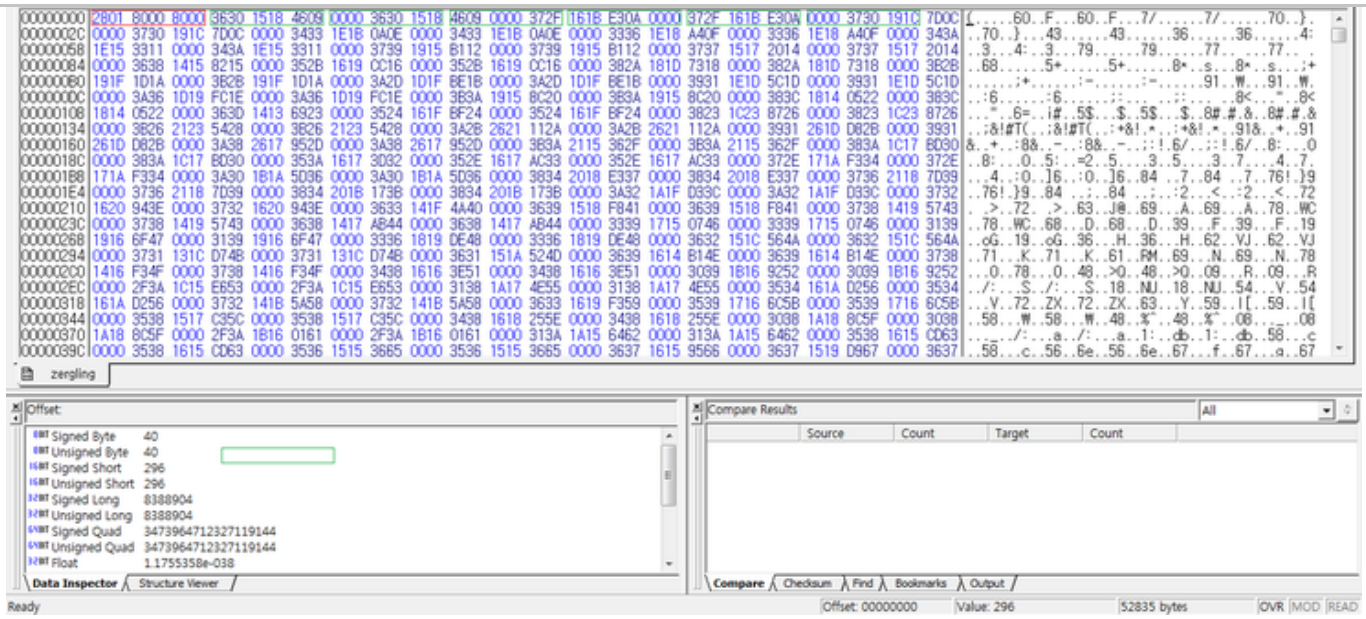


귀엽네요. 대충 이렇게 렌더링되면 됩니다.

왜 제가 수동으로 하자고 했는지는 모르겠지만 일단 수동으로 가봅시다.
저 미친듯이 많은 프레임을 모두 렌더링한다는건 좀 어렵고,
0번 프레임의 10번째 줄만 수동으로 렌더링해봅시다.

우리의 귀요미 zergling.grp를 헥스 에디터로 열어줍시다. 원래 HxD를 썼는데 너무 불편해서 그냥 Hex Workshop으로 가겠습니다.

whyask37's blog



화질이 깨졌군요. 뭐 괜찮습니다. 이제 이거를 해독해보도록 하죠.

```
struct GRPHeader { uint16 frameCount; uint16 grpWidth; uint16 grpHeight; }; struct GRPFrameHeader { uint8 frameXOffset; uint8 frameYOffset; uint8 frameWidth; uint8 frameHeight; uint32 lineTableOffset; };
```

일단 파일 맨 처음에는 GRPHeader가 옵니다. GRPHeader에 따라 읽으면
frameCount = 0x0128 = 296 //리틀엔디안, 기억나시죠?
grpWidth = 0080 = 128
grpHeigh = 0080 = 128

좋아요. 이제 그 다음에는 GRPFrameHeader가 frameCount개만큼 옵니다. 대충 이런거죠.
앞에 20개정도만 헥스에디터에서 앞에 6바이트 지우고 8바이트 단위로 그대로 읽은겁니다. uint32는 리틀엔디안이고요 당연히

2	37	55	2F	47	16	22	1B	27	00000AE3	2787
3	37	55	2F	47	16	22	1B	27	00000AE3	2787
4	37	55	30	48	19	25	1C	28	00000C7D	3197
5	37	55	30	48	19	25	1C	28	00000C7D	3197
6	34	52	33	51	1E	30	1B	27	00000E0A	3594
7	34	52	33	51	1E	30	1B	27	00000E0A	3594
8	33	51	36	54	1E	30	18	24	00000FA4	4004
9	33	51	36	54	1E	30	18	24	00000FA4	4004
10	34	52	3A	58	1E	30	15	21	00001133	4403
11	34	52	3A	58	1E	30	15	21	00001133	4403
12	37	55	39	57	19	25	15	21	000012B1	4785
13	37	55	39	57	19	25	15	21	000012B1	4785
14	37	55	37	55	15	21	17	23	00001420	5152
15	37	55	37	55	15	21	17	23	00001420	5152
16	36	54	38	56	14	20	15	21	00001582	5506
17	35	53	2B	43	16	22	19	25	000016CC	5836
18	35	53	2B	43	16	22	19	25	000016CC	5836
19	38	56	2A	42	18	24	1D	29	00001873	6259

```
00000020 3730 191C 7D0C 0000 70..}...
00000028 3730 191C 7D0C 0000 70..}...
00000030 3433 1E1B 0A0E 0000 43......
00000038 3433 1E1B 0A0E 0000 43......
00000040 3336 1E18 A40F 0000 36......
00000048 3336 1E18 A40F 0000 36......
00000050 343A 1E15 3311 0000 4:..3...
00000058 343A 1E15 3311 0000 4:..3...
00000060 3739 1915 B112 0000 79......
00000068 3739 1915 B112 0000 79......
00000070 3737 1517 2014 0000 77.. ...
00000078 3737 1517 2014 0000 77.. ...
00000080 3638 1415 8215 0000 68......
00000088 352B 1619 CC16 0000 5+......
00000090 352B 1619 CC16 0000 5+......
00000098 382A 181D 7318 0000 8*..s...
000000A0 382A 181D 7318 0000 8*..s...
```

대충 이런 식으로 데이터를 얻었는데 뭔가 신기한게 있습니다.

(0-1), (2-3), ... , (15-16), 16 단위로 데이터가 똑같습니다. RetroGRP로 보면 각각 저글링이 16방향을 북쪽부터 동쪽을 거쳐서 남쪽으로 바라보는것까지의 그래픽이 있고요.

왼쪽 바라보는것은 이러한 GRP 데이터를 좌우대칭시켜서 그리는거고
저글링이 아니라 좀 고급스러운 유닛, 배틀크루저같은것은 저 17개의 값이 모두 다릅니다. (32방위를 바라볼 수 있다는 뜻이죠)
저글링은 그냥 용량 아까워서 그냥 16방위만 넣은듯하네요...

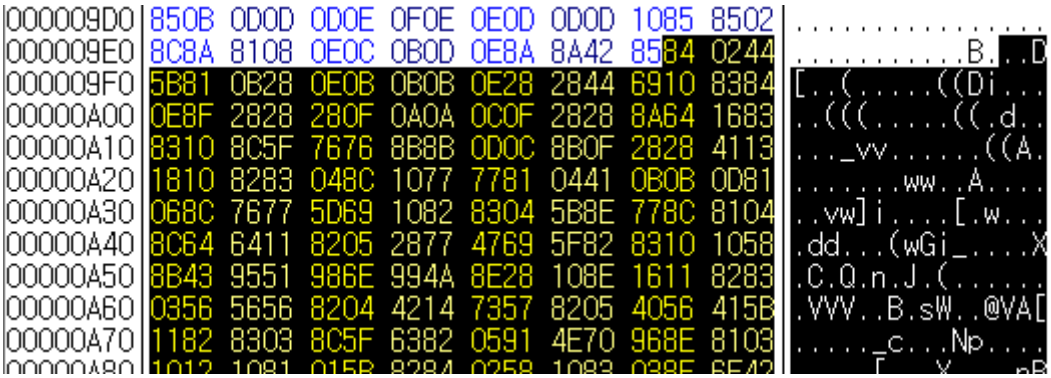
frame #1이 21 * 24 사이즈라는군요.
좋아요. 한번 이거를 디코딩해봅시다.
전체 디코딩은 좀 에바고 10번줄만 디코딩하기로 했었죠? 전체 디코딩은 프로그래밍으로 하도록 합시다 귀찮아요;

ltbOff (lineTableOffset) = 2374로 적혀있군요. 원래 GRP 파일에서 Ctrl+G를 잘 써서 파일 시작으로부터 0946(2374)으로 이동해보겠습니다.

하여튼 총 가로줄이 24개니까 (21*24) 24개의 uint16이 각각 line data까지의 relative offset이 된다고 합니다.
그러니까 2374 ~ 2422의 48byte가 line data에 대한 offset이 되죠.

```
4132 96CA 0000 3000 3800 4100 4A00 5200
5E00 6E00 7C00 8A00 9800 A700 B900 CA00
DD00 F200 0601 1901 2C01 4101 4F01 5D01
```

GRP 포맷에 따르면 실제 라인 데이터는 2374 (lineOffsetTable) + 167 (relative offset) = 2541 번째 부터 있다고 하는군요. 좋아요. 가보죠.



대충 데이터가 84 02 44 5B 81 0B 28 0E 0B 0B 0B 0E 28 ... 이런 식으로 가는군요. 이 데이터는 RLE 압축되어있습니다.
이제 이 RLE 압축을 풀면 팔레트 인덱스를 얻을 수 있죠.

압축을 푸는 방법은

1. byte >= 0x80 : (byte - 0x80)만큼 0을 출력
2. 0x80 > byte >= 0x40 : (byte - 0x40)만큼 다음 바이트를 반복해서 출력
3. 0x40 > byte : 다음 byte만큼의 byte를 그대로 출력

입니다. 그러니까 위에 데이터는
(**8**4) (02 44 5B) (**8**1) (0B 28 0E 0B 0B 0B 0E 28 28 44 69 10) (**8**3) ...

이런식으로 되겠죠. 이걸 디코딩하면
(00 00 00 00) (44 5B) (00) (28 0E 0B 0B 0B 0E 28 28 44 69 10) (00 00 00)
이렇게 정확하게 21byte가 읽혔죠. 이걸로 디코딩 완료.

음 이렇게 21byte를 써서 10번째 줄을 디코딩했는데 뭔가 허전한것같아요.
빨리 전체 이미지를 보고싶네요.

보도록 합시다.

위에서 자세한 내용은 다 설명했으니 바로 코드로 들어가겠습니다. 주석은 전 강의에서 말했듯이 영어를 주로 쓸거고요.
일단 gui library에 대해서는 자세한 설명은 하지 않겠습니다. 중요하지 않아요.
간단히 gui library에 대해서는

1. 임의의 픽셀에 setpixel/getpixel이 됨. (당소 배열 연산)

형이다지)

gui library는 public domain입니다. zlib 넣으려했는데 라이센스가 코드 길이와 반비례하게 길어서 그냥 안넣었어요.

그럼 이제 이걸 가지고 재미있는 grp library를 만들어봅시다.
앞에서 했던 일들을 포인터 연산들만 가지고 잘 다루면 됩니다.

Chunk는 이런거예요. 미리 알아두세요. MapCanvas에서의 Chunk랑 조금 달라요. MapCanvas도 이 형태의 Chunk로 바꿀거고요.

```
typedef struct {
    char type[4];
    uint32 len;
    uint8 data[1];
}Chunk;
```

그러면 코딩을 해서... 이렇게 됐습니다!

grp.h

Colored By **Color Scripter™**

```
1  /**
2  * GRP reader. Made by whyask37(@naver.com)
3  * Coded at : 2013-09-26 (Thr)
4  *
5  * This library reads Starcraft GRP(Graphics) file.
6  */
7
8  #ifndef GRP_HEADER
9  #define GRP_HEADER
10
11 #include "gui/gui.h"
12 #include "typedef.h"
13 #include "chunk.h"
14 #include <vector>
15 #include <map>
16
17 // GRP format reference :
18 //   http://www.staredit.net/starcraft/GRP_Image_Format
19 //   http://www.modcrafters.com/wiki/index.php?title=GRP_files
20
```

```
26          RGBAbyte color[256];
27      };
28
29      struct GRPHeader {
30          uint16  frameCount;
31          uint16  grpWidth;
32          uint16  grpHeight;
33      };
34
35      struct GRPFrameHeader {
36          uint8   frameXOffset;
37          uint8   frameYOffset;
38          uint8   frameWidth;
39          uint8   frameHeight;
40          uint32  lineTableOffset;
41      };
42
43      typedef uint16* lineTable;
44      typedef uint8* lineData;
45  }
46  #include <packoff.h>
47
48  // GRP class
49
50  class GRP {
51  public:
52      GRP();
53      ~GRP();
54
55      int load(const Chunk* orig);
56      void clear();
57
58      int getFrameCount() const;
59      void setPalette(const grp::GRPPalette *palette);
60      int getImage(Image& img, int frame) const;
61
62  private:
63      const grp::GRPHeader* getGRPHeader() const;
64      const grp::GRPFrameHeader* getFrameHeader(int frame) const;
65      const grp::lineTable getLineTable(int frame) const;
66      const grp::lineData getLineData(int frame, int y) const;
67
68  private:
69      Chunk* _data;
70      const grp::GRPPalette* _palette; //shared.
71  };
72  }
```

Colored By Color Scripter™

```
#include "grp.h"

namespace scdata {
    GRP::GRP() : _data(nullptr), _palette(nullptr) {}
    GRP::~GRP() { clear(); }

    int GRP::load(const Chunk* chk) {
        Chunk* _data2;
        CopyChk(_data2, chk);
        free(_data);
        _data = _data2;
        return 0;
    }


    void GRP::clear() {
        free(_data);
    }


    int GRP::getFrameCount() const {
        return getGRPHeader()->frameCount;
    }


    void GRP::setPalette(const grp::GRPPalette* pal) {
        _palette = pal;
    }


    int GRP::getImage(Image &img, int frame) const {
        if(frame < 0 || frame > getFrameCount()) throw std::bad_exception("bad frame no");

        //prepair rendering buffer
        const grp::GRPHeader* grpheader = getGRPHeader();
        const grp::GRPFrameHeader* header = getFrameHeader(frame);
        img.LoadBlank(grpheader->grpWidth, grpheader->grpHeight);

        //draw every pixel
        for(int i = 0 ; i < header->frameHeight ; i++) {
            const grp::lineData data = getLineData(frame, i);
            for(int j = 0 ; j < header->frameWidth ; j++) {
                img.SetPixel(j + header->frameXOffset, i + header->frameYOffset, _palette->color[data[j]]);
            }
        }

        return 0;
```

```
52  const grp::GRPHeader* GRP::getGRPHeader() const {
53      return (const grp::GRPHeader*)_data->data;
54  }
55
56
57  const grp::GRPFrameHeader* GRP::getFrameHeader(int frame) const {
58      return (const grp::GRPFrameHeader*)(_data->data + sizeof(grp::GRPHeader) + sizeof(grp::GRPFrameHeader
59  }
60
61
62  const grp::lineTable GRP::getLineTable(int frame) const {
63      const grp::GRPFrameHeader* header = getFrameHeader(frame);
64      return (grp::lineTable)(_data->data + header->lineTableOffset);
65  }
66
67
68  const grp::lineData GRP::getLineData(int frame, int y) const {
69      // Patch 1 : Originally this code reused rendered data using std::map.
70      //  Now this function decodes rle in every call. This should be faster.
71      //  RLE decoding is faster than std::map::find & std::map::insert;
72      //
73
74      uint8* paldata;
75      const grp::GRPFrameHeader* header = getFrameHeader(frame);
76
77      //decode GRP RLE compression
78      // Prepare buffer
79      paldata = (uint8*)malloc(header->frameWidth);
80      memset(paldata, 0, header->frameWidth);
81
82      // RLE decode
83      const uint32 lineDataOffset = getLineTable(frame)[y];
84      const uint8* readp = _data->data + header->lineTableOffset + lineDataOffset;
85      uint8* outp = paldata;
86      int left = header->frameWidth;
87
88      while(left > 0) {
89          if(*readp >= 0x80) { //0x80 <= compSect
90              int rep = *readp - 0x80;          //number of repetition of transparent color
91              while(rep > 0) {
92                  *outp = 0;
93                  rep--; left--; outp++;
94                  if(left == 0) break;
95              }
96              readp++;
97          }
98
```

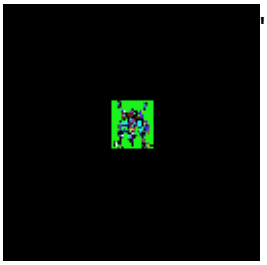
```
104                 *outp = col;
105                 rep--; left--; outp++;
106                 if(left == 0) break;
107             }
108         }
109
110         else { //type 3 : compSect < 0x40
111             int n = *readp;                 //number of bytes to copy
112             readp++;
113             while(n > 0) {
114                 *outp = *readp;
115                 n--; left--; outp++; readp++;
116                 if(left == 0) break;
117             }
118         }
119     }
120
121     //decode complete.
122     return paldata;
123   }
124 }
```

음 그래도 직관적인 편이죠?

좋아요. 이렇게 grp라이브러리를 이용해서 대충 저글링을 렌더링하고싶어요.

팔레스 인덱스를 어떻게 하면 RGB로 바꿀 수 있을까요?

여기에 대해서는 나중에 생각해보도록 합시다. 사실 저글링에서 나온 것들은 꼭 팔레트 인덱스도 아닙니다.
이거를 하기 위해서는 다음 강의에서 pcx를 파싱하는 라이브러리를 만들고 저글링에게 색깔을 입혀주도록 하죠.
지금은 귀요귀요미한 저글링이 주겄습니다 ㅠㅠ



... 뭐 RetroGRP로 본것과 대략 형태가 비슷하다고정도만 하죠.
각 팔레트 인덱스에 랜덤으로 컬러를 입혀본거예요 그냥.

아어는 코드는 이렇습니다.

main.cpp

Colored By **Color Scripter™**

```
1  #include "gui/gui.h"
2  #include "grp.h"
3  #include "chunk.h"
4
5  Chunk* getChunk(const char* fname) {
6      Chunk* chk;
7      FILE *fp;
8      fp = fopen(fname, "rb");
9      if(fp == NULL) return NULL;
10     else {
11         int fsize;
12         fseek(fp, 0, SEEK_END);
13         fsize = ftell(fp);
14         rewind(fp);
15         chk = GetBlankChunk("    ", fsize);
16         fread(chk->data, 1, fsize, fp);
17         fclose(fp);
18         return chk;
19     }
20 }
21
22 int CALLBACK WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {
23     Image window;
24     scdata::GRP grp;
25     Chunk* chk;
26     scdata::grp::GRPPalette pal;
27
28     //load test data
29     chk = getChunk("zergling.grp");
30     grp.load(chk);
31
32     //generate palette.
33     // using real in-game palette requires more work. Let's do that later;
34
35     uint32 randtable[256] = { //generated with RAND() of MS office excel 2010...
36         2843569540,  1219377508,  790069832,  1204880951,  4271431743,  299555302,  3405645734,  376
37         1028735742,  3740597423,  2923759501,  449651840,  157744625,  3951809007,  3808710785,  369
38         2074779243,  2912315326,  1108437379,  3828093129,  3090680543,  1754834629,  2723103424,  3
39         2976685988,  1303325962,  1767428637,  2070059986,  2449426930,  2971311316,  3455687905,  2
40         3092682093,  202685793,  2510546026,  649464849,  2025967377,  2572540109,  3456165043,  131
41         4025212033,  880719151,  2827919318,  2737045553,  1003102213,  2312253497,  2893141436,  24
42         2001861746,  3527484138,  1291786044,  2492870868,  2567760185,  2931473610,  2885064034,  2
43         3148814297,  514785075,  2233969984,  3840397041,  3303458271,  3734739260,  2455259217,  56
44         917292792,  3180460358,  4181949868,  2379016515,  1670474837,  746318529,  44095572,  55663
```

```
    3108482278,  592379132,  2669557436,  1246370512,  2088334682,  2746554225,  710420589,  206
    2876639670,  3745857969,  279935625,  3979732294,  2038362993,  1515140697,  1285097566,  13
};

grp.setPalette(&pal);

//extract
Image img;
int framen = grp.getFrameCount();
for(int i = 0 ; i < framen ; i++) {
    grp.getImage(img, i);
    img.ApplyFilter([&](RGBAbyte col, int, int) -> RGBAbyte {
        col.a = 255;
        return col;
    });

    char fname[512];
    sprintf(fname, "out\\img%d.png", i);

    img.SavePNG(fname);
}

MessageBox(NULL, "Done", "Done", MB_OK);

return 0;
}
```

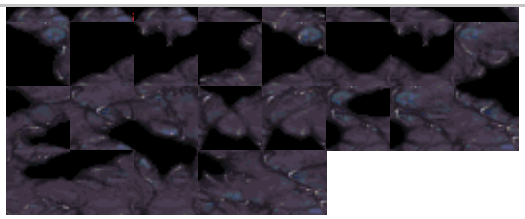
네 대충 이렇게 만들면 될것같아요. 코드는 찬찬히 읽어보시고

이렇게 해가지고 하면 첨부파일처럼 재미있는 프로그램이 만들어져요. 참 재미있어요.

gui.cpp는 유니코드 지원 안하니까 언어셋을 Not Set으로 해주세요.

근데 뭐가 슬픈것같아요. 뭔가 저글링처럼 깨진것만 출력되면 뭔가 슬플것같으니까
.wpe를 사용해서 지형 파일마다 있는 grp를 한번 출력해보도록 합시다.

Twilight.wpe랑 Twilight.grp를 BrooDat.mpq\tileset에서 추출합니다.
zergling.grp를 twilight.grp로 바꾸고 wpe에서 대충 팔레트를 읽은 뒤에 출력하면 아래와 같이 멋진 그림이 나옵니다.
와! 크립같은게 나옵니다. 뭔가 신기하네요.
저걸로 크립을 렌더링하는건가?



하여튼 좋아요.

네 오늘은 여기서 마치도록 할께요. 충분히 한것같아요.

첨부파일

out.zip
grptest.zip

0

8. Let's read and print the GRP file. (One)

2013. 9. 25.

6

7. Let's print the terrain again.

2013. 9. 19.

6

6. Foundation work

2013. 9. 12.

0

이 블로그 인기글

MPQ 가지고놀기 (1) - 간단한 MPQ 파일 분석

2013. 10. 19.

11

5. SFmpq (ShadowFlare's MPQ Library) 와 예제

2013. 9. 11.

1

[뻘강의] 13. 트리거 프로그래밍 - TRIG-MRGN 루프

2014. 2. 24.

0

4. scenario.chk

6

whyask37's blog

[뻘강의] 2. 데스 사이의 대입, 더하기

2014. 1. 19.

1

back to top

View in PC version

6