CopyRect でアニメーション

ゲームでキャラクターをアニメーションさせたり多くのザコキャラを扱うとなると、大量にコンポーネントの Image を動かすやり方をとっていてはコンピュータの動作が重くなってしまいます。

そこで、Image の数を少なく、かつ多くの絵を扱う事ができる「CopyRect」という技を使います。

●CopyRect を使ってみる

＜一般式＞

Form1->Canvas->CopyRect(①表示先の Rect , ②表示する画像, ③表示する画像の Rect);

まずは Image1 を作っておいて、今まで通り画像を読み込みます。

Image1->Picture->LoadFromFile(“ファイル名.拡張子”);

サイズが 640×480 の画像を使うとして内容を説明します。

① 表示先の Rect

Form1 上のどの範囲に画像を貼り付けるかを指定します。

Form の左上に画像を等倍で貼り付ける場合は

Rect(0,0,640,480) にします。

② 表示する画像

今回は Image1 に取り込んだ画像を使うので、

Image1->Canvas にします。

③ 表示する画像の Rect

Image に取り込んだ画像の切り出したい範囲を指定します。

300×500 の画像をそのまま使いたい場合は

Rect(0,0,640,480) です。

```
//---------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    Image1->Picture->LoadFromFile("gazou.bmp");
    Image1->Visible = false;
    //↑取り込んだ画像のサイズは640×480

    Form1->Canvas->CopyRect(Rect(0,0,640,480),
    Image1->Canvas,Rect(0,0,640,480));
                        //↑ ↑TRect↑ ↑
}
//---------------------------------------------------------
```

↑これで 640×480 の画像が等倍で Form の左上に表示できます。

ちなみに、CopyRect は fastcall に書いても動作しないので、Timer や Button などにに書きましょう。

ところで、Rect(0,0,640,480)とありますが、これは「TRect 型」といわれる一つの固まりで、 この「Rect(0,0,640,480)」を変数のようにして使う事が可能です。

まずヘッダに「TRect 型変数」を宣言

```
TRect X;
```

「int X;」と宣言するのと同じ要領です。
この変数 X に値を代入しましょう。

```
X = Rect(0,0,640,480);
```

これで、X を書くだけで範囲を指定できるようになりました。

この要領で、ソースコードを書き換えてみましょう。

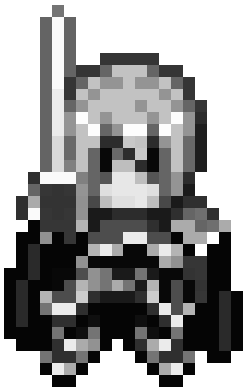
```
//---------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    Image1->Picture->LoadFromFile("gazou.bmp");
    Image1->Visible = false;
    //↑取り込んだ画像のサイズは640×480

    //「TRect X,Y;」とヘッダに宣言しておく
    X = Rect(0,0,640,480);
    Y = Rect(0,0,640,480);

    Form1->Canvas->CopyRect(X,Image1->Canvas,Y);
                          //↑すっきりして見やすくなる！
}
//---------------------------------------------------------
```

見た目もきれいで間違いも減ります。

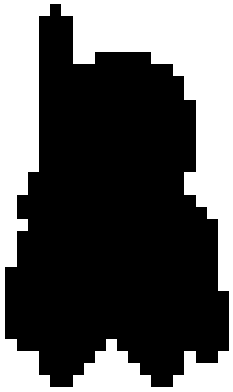
また、X や Y に Rect を代入し直せば描画する範囲をかえる事ができます。

●マスク

画像は四角形をしているので、キャラクターが描かれた画像を普通に貼り付けても四角い背景が残ってしまい、ゲームに使えません。そこで画像の背景を透過して必要な部分だけを表示させる「マスク」というものがあります。



↑元画像（背景は白）



↑マスク画像（背景は白、キャラは黒）

元の画像とマスク画像を次のように重ねて表示させて表示させます。

```
//---------------------------------------------------
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    //↓元画像とマスクをそれぞれ読み込む
    Image1->Picture->LoadFromFile("gazou.bmp");
    Image1->Visible = false;
    Image2->Picture->LoadFromFile("gazou-mask.bmp");
    Image2->Visible = false;

    X = Rect(0,0,640,480);
    Y = Rect(0,0,640,480);

    //↓元画像とマスクを合体させる
    Form1->Canvas->CopyMode = cmSrcErase;
    Form1->Canvas->CopyRect(X,Image2->Canvas,Y);

    Form1->Canvas->CopyMode = cmSrcInvert;
    Form1->Canvas->CopyRect(X,Image1->Canvas,Y);
}
//---------------------------------------------------
```

これで Form1 に背景が透けた画像が描画されます。
しかし、この方法だと Timer などで連続描画した際にちらつきが生じてしまいます。

●CopyMode

CopyMode というのは CopyRect を使った際「どのように描画するか」を設定するプロパティです。

cmSrcCopy：そのままコピーする
cmSrcErase,cmSrcInvert：これらを合わせて使い画像の背景を透明化する

●ちらつきを抑える

マスク処理を行った場合、１つの画像を表示する毎に２回描画しなければならないので、ゲームなどで使うと画像がちらついてしまいます。

① cmSrcErace でマスクを描画

↓

② cmSrcInvert で元画像を描画、重ねる

↓

③ 背景の透過された画像が完成

↓

① cmSrcErase でマスクを描画

以下繰り返し

人に見てもらいたいのは③の完成した画像だけですが、この方法だと全ての過程を見せる事になりちらついてしまいます。

③だけを見せるためには、まず Image3 を新たに作り、必要な物を全て Image3 の中で合体させてしまいましょう。そして Image3 上の「完成した画面」だけを Form にコピーすれば余計な工程を見せずに済みます。

＜　　こ　　こ　　が　　大　　事　　！　　＞

```
//---------------------------------------------------
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    //↓元画像とマスクをそれぞれ読み込む
    Image1->Picture->LoadFromFile("gazou.bmp");
    Image1->Visible = false;
    Image2->Picture->LoadFromFile("gazou-mask.bmp");
    Image2->Visible = false;
    Image3->Visible = false;

    X = Rect(0,0,640,480);
    Y = Rect(0,0,640,480);

    //↓Image3で画像を合体させる
    Image3->Canvas->CopyMode = cmSrcErase;
    Image3->Canvas->CopyRect(X,Image2->Canvas,Y);

    Image3->Canvas->CopyMode = cmSrcInvert;
    Image3->Canvas->CopyRect(X,Image1->Canvas,Y);

    //Image3の画像をFormに表示させる
    Form1->Canvas->CopyMode = cmSrcCopy;
    Form1->Canvas->CopyRect(X,Image3->Canvas,Y);
}
//---------------------------------------------------
```

画像の読み込みは毎回する必要がないので、コンストラクタに書いてもよい

なお、背景画像を Image3 に予め読み込ませた上で元画像とマスクを合体させれば、背景の上にキャラクター表示させる事ができます。

●Rect の中身

「描画と Rect」の項目でも書かれているが、

```
Rect(0,0,640,480);
```

と書いてきた Rect の中身は、

```
Rect(Left,Top,Right,Bottm);
```

ている。

また、変数に代入する事でこれらの数値を変える事ができ、画像を動かす事ができます。

☆アニメーション

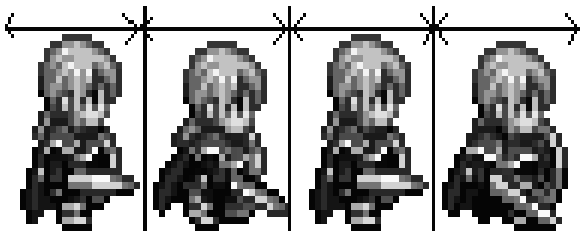
CopyRect による描画ができたら、CopyRect を使ってアニメーションさせてみましょう。

まずは画像を用意します。

・元画像（例）

・マスク

１つの画像に人物の挙動全てをまとめておけば必要な Image が少なくて済みます。



画像を作る際、独立している絵ごとの幅を同じサイズにしておきましょう。
こうすることでソースコードを簡略化する事ができます。
ちなみにこの画像の１つのコマのサイズは、横 100 縦 120 です。

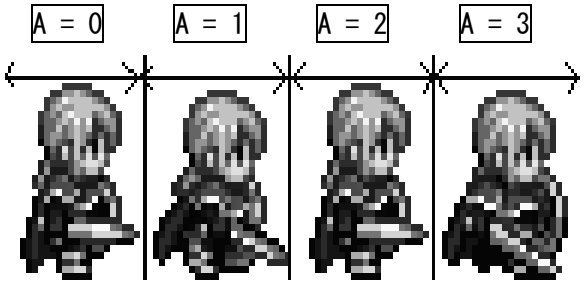
●変数を使って Rect の座標をコントロール

```
TRect X,Y;
int A;
```

ヘッダにこれを書いた上で

```
X = Rect(0,0,100,120);
Y = Rect(100*A,0,100*A+100,120);
        //(Left,Top,Right,Bottom)
```

と Rect 変数に代入します。
タイマーをつくり、変数 A が「0→1→2→3→0→1･･･」となるように書きます。



変数 A に応じた座標が切り抜かれるように書きましょう。

```
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    //背景を読み込んで、Image3を初期化している
    Image3->Picture->LoadFromFile("haikei.bmp");

    //↓元画像とマスクをそれぞれ読み込む
    Image1->Picture->LoadFromFile("chara1.bmp");
    Image1->Visible = false;
    Image2->Picture->LoadFromFile("chara1-mask.bmp");
    Image2->Visible = false;
    Image3->Visible = false;

    //変数Aに対応した画像をFormの左上に表示したい場合
    X = Rect(0,0,100,120);
    Y = Rect(100*A,0,100*A+100,120);

    //↓Image3で画像を合体させる
    Image3->Canvas->CopyMode = cmSrcErase;
    Image3->Canvas->CopyRect(X,Image2->Canvas,Y);

    Image3->Canvas->CopyMode = cmSrcInvert;
    Image3->Canvas->CopyRect(X,Image1->Canvas,Y);

    //Image3の画像をFormに表示させる
    Form1->Canvas->CopyMode = cmSrcCopy;
    Form1->Canvas->CopyRect(X,Image3->Canvas,X);

    //Aは「0,1,2,3,0,1,2,3･･･」と変化する
    A += 1;
    if(A > 3){
        A = 0;
    }
}
```

これで、アニメーションしてくれます。Image3 を毎回初期化させないと、キャラクターが足踏みするごとに残像が残る事になるので、背景を読み込んだり Image3 を塗りつぶしたりする必要があります。

＜塗りつぶしの例＞

```
Image3->Canvas->Brush->Color = clWhite;
Image3->Canvas->FillRect(Rect(0,0,ClientWidth,ClientHeight));
```