カット

Creator
タナカミノル（pickles）

αビデオを使ったFlashムービー

Flash 8で扱えるようになったαビデオを使って、
動画と背景をリアルタイムに合成する。

Title
カット

Sample URL
http://www.pickles.tv/dotfla/02/

Archive
cut.zip

File
10

スクリプト
ActionScript 1.0

対応プレーヤー
Flash Player 8以上

制作アプリケーション
Flash 8

発案～デザイン
グラフィックでおおまかな路線を決定する

仕事で追いつめられて、死にたいと思って作ってみました…という冗談（？）はさておき、本当の発想のきっかけは、昔リストカットしていたという友人です。興味本位でいろいろ聞いたところ、「そんなに死にたいわけじゃないけど、なんかやってしまう」とのこと。それなら、自分を傷つけなくても代用品があればよいのではないかと漠然と考えたのが始まりです。

リサーチした結果、

- やっちゃいけないことをする自分
- 命を削ってる感
- 血が出るのがきれい

という、このあたりの要素を満たせたらスッキリするのではないかと思い、上記を満たす演出を考えました。ただ、どうフィニッシュさせるか悩みました。当初はホラームービーや不謹慎ゲームにまとめようと思っていたのですが、最終的には「勝手に公共広告機構」という感じでまとめてみました（ホラーチックになっているのは、当初案のなごりです）。

グラフィック作り

まずは、グラフィックを作ります。“自分が切る”イメージと“命を削ってる”感じを出したいので、切った手首からはリアルな血ではなくハートが出ている形にしました。

このときは背景が白ですね。きれいなイメージということで白にまとめるつもりでしたが、これは、途中で仮り組みしてみたときに怖さが出なかったので黒に変更しています。それと画面の形も縦長に変更しました。ハートの飛沫を見せるためです。

できあがったグラフィックを見て、具体的な表現の検討に入ります。ハートが飛び散るイメージは、パーティクルをスクリプトで作成して再現することにします。手の部分は、最初は静止画で制作するつもりでしたが、リアル感を出すために動画にすることにしました。

最初のグラフィックイメージ

素材の作成
動画の撮影→αビデオ化

撮影

動画の撮影を行います。必要なのは両手の部分です。重なる部分があるのと後で背景が変えられるので、Flash 8のαビデオを使うことにしました。αビデオの作成には、撮影後のキーイング（対象となるオブジェクトだけを抜き取る）作業が必要となります。

キーイングについては、セトウさんのページ（167 ページ）でも触れていますが、撮影が難しいです。キーイングは背景色を利用して対象を抜きますが、影が付いてしまうときれいに抜けません。そのため、仕事では通常、リフレクメディアクロマキーという製品を使用しています。

URL仓 **CreativeSuite**
http://www.creativesuite.com/

これは、カメラにブルーまたはグリーンライトを取り付け、その照明を背景の特殊な幕に反射させれば影が発生せず、背景がキーイング色になるというものです。レンタルで１日５万円程度かかかります（それでも、まともなスタジオ撮影に比べたら安価です）。ですが、「そんなに金ないよ！」という方のために、今回は安価な方法を紹介します。

それは、「ニューカラーボード」という発泡スチロールのボードを４枚使用して撮影する方法です。ニューカラーボードは画材屋さんに置いてあるもので、2000 円弱で購入できます（緑もあります）。なぜこのボードを使用したかというと、布や紙と違いヨレができないので影が発生しないのと、カラー塗料が吹き付けであるため、光の反射が均一になり背景色も均一に保たれるからです。これで背景の影、色むらも解消できます。

照明は、家庭用のスポットライトとスタンドライトを使用しています。キーイングでの照明のコツは、背景と対象を同一の照明で当てないことです。対象の照明を先に作り、背景に影が発生してしまったら、それを背景用の照明で消すといった方法です。

リフレクメディアクロマキーでの撮影

今回の撮影セット

○ 編集

撮影が終わったら、使用する部分を編集します。必要なのは「左手のループ画像」「左手の跳ね上がっている状態から戻る部分」「右手のカミソリを下げる」の３つです。

●左手のループ画像

ループ画像は再生と逆再生を組み合わせて作りますが、単純につないでしまうと違和感が生じます。違和感を減らすコツは２点あります。

- 再生と逆再生の速度を変える
- 逆再生が始まる場所の見極め

速度が違うと別の動画だと錯覚しやすくなるので、ループの違和感を減らせます。また、逆再生が始まる場所をうまく見極めます。よほどパントマイムがうまい人でない限り、人の動作はつねに揺れています。その揺れの頂点を見極めて、そこを逆再生につなげます。それにより、違和感を減らすことができます。

再生と逆再生を組み合わせる

●左手の跳ね上がっている状態から戻る部分

跳ね上がる瞬間は勢いも付けたいので、動画はなしにして戻ってきた部分をループ動画の最初の部分につなげています。つなぎの部分はスムーズにつなげるためにディゾルブ（クロスフェード）を使用しています。

●右手のカミソリを下げる

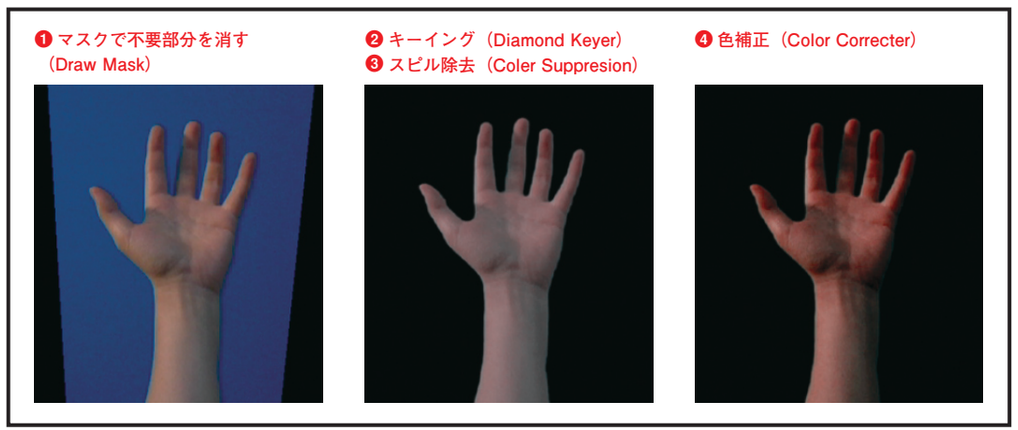
これは、撮影時にカミソリが水平に止められていなかったので（要は NG）、上げる動画を逆再生して下げる動画にしました。

○ キーイング

編集した動画をキーイングします。キーイング可能な合成ソフトでメジャーなのは「Adobe After Effects」（アフターエフェクト）ですが、きれいにキーイングするにはテクニックを要します。そこで、専門のソフトにしては安価なうえ、キーイングが優秀な Autodesk の「Combustion 4」（コンバッション 4）を使用しました（値段は 15 万円弱です）。

どんなソフトを使用してもキーイングの手順としては以下のとおりになります。（ ）内は Combustion 4 の場合のメニューです。

キーイングの手順



撮影部分で影が出ると問題と書きましたが、Combustion 4 の場合は Diamond Keyer が優秀なので、多少の影が出てもきれいに抜いてくれます。ローカライズされておらず、英語のため多少使いにくいのですが、そのデメリットを考えても、使用する価値はあると思います。

「ソフトにそこまでお金かけられないよ！」という方は、やはり After Effects の使用をお勧めします。また、環境が Mac でしたら「Motion2」が非常に安価です。時間はかかりますが、使用に耐えられる動画は作れます。

URL台 **Adobe After Effects**
http://www.adobe.co.jp/products/aftereffects/

URL台 **Autodesk**
http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=5562397

URL台 **Motion2**
http://www.apple.com/jp/finalcutstudio/motion/

○FLV の作成

使用する動画ができたら、書き出して FLV（Flash Video）にします。動画にはさまざまなコーデックがあり、α情報を保持できるのは「非圧縮」か「アニメーションコーデック」になります。非圧縮だと重すぎるので、通常はアニメーションコーデックを使用します。注意点は圧縮プログラムの深さを「約 1670 万色以上」にすることです。この「以上」という部分がα情報になります。

書き出したものを「Flash 8 Video Encoder」を使用してまとめて変換します。「設定」→「詳細設定」を選んでチャンネルの「エンコード」にチェックを入れて書き出せば、αチャンネル付きの FLV が生成されます。その際、フレームレートは Flash に合わせておきましょう。間違えると動画が同期しません。これを Flash に取り込みます。

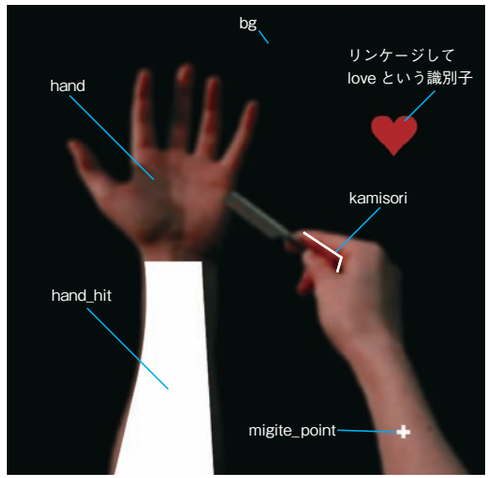
スクリプト
パーティクルの生成

label の main 前後はおまけみたいなものなので、main 部分のみ説明します。各 MC は以下の形でインスタンス名（シンボル名も同一）を付けています。

登場する MC の役割

インスタンス名	説明
hand	「左手のループ動画」と「戻る部分の動画」
hand_hit	この部分にカミソリの原点が当たると切れる
kamisori	「カミソリを下げる動画」にカミソリが光るアニメーションを付ける
migite_point	kamisori が追いかけるようにするための MC
love	attachMovie() で生成される元

登場する MC



スクリプトの構成としては簡単で、クリック中に kamisori の原点が hand_hit に当たれば、当たっている最中の「位置」「距離」「時間」を取得して、その「時間」（単位はミリ秒）分の数だけハートを大量に生成するようにしています。

ここでは、パーティクル部分のみ掲載しておきます。

「パーティクル」というのは、3D アニメーションで用いられているモデリング技法で、自然界の火、煙、雲などポリゴンでは表現しにくいものを粒子の集合として表現する方法です。

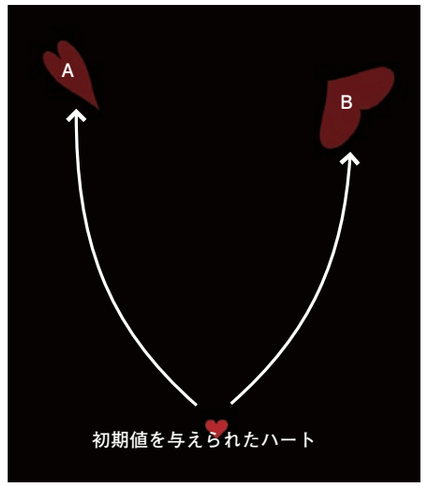
今回のハートが飛び散るという表現もパーティクルを利用し、ハートの 1 つ 1 つが初期値をもとに勝手に動くようにしています。

ハートの動きの初期値は、以下の 11 種類を設定しています。

- X 発生位置
- Y 発生位置
- 角度
- 大きさ
- X 方向と X 速度
- Y 速度

- 回転方向と回転速度
- X 減速
- Y 減速
- X 拡大率
- Y 拡大率

初期値を与えられたハートは A、B のような動きをする



ランダムに値を設定しているので、上図の A や B のようにいろいろな動きをするようになっています。あとは、このハートを大量に生成するのみです。

スクリプトは以下になります。次ページの B の部分の値を変更すると、動きがさまざまに変化しますので、試してみてください。

SOURCE カット後のパーティクル

```
function cut_main(xpos, kyori, ypos, ed) {//ed は時間
    var i = 0;// ハートの ID 番号
    var edcount = 0;// 発生したハートの数
    var maxh;// 一度に発生するハートのマックス数
    this.onEnterFrame = function() {
        maxh = (ed-edcount)/10;// 残りハートの数からマックスを割り出す
        if (maxh>=20) {
            maxh = 20;//20 以上なら 20 にする
        } else if (maxh<2) {
            maxh = 2;//2 未満なら 2 にする
        }
        if (edcount<ed) {// 発生数が満たされるまで以下の処理
            var m = (Math.floor(Math.random()*maxh));// 一度に発生するハートの数
            edcount += m;// 発生したハートの数に加算
            for (f=1; f<=m; f++) {
```

A

次のページに続く

```
                i++;//ID を加算して
                var newname = "love_"+i;// インスタンス名を設定
                var syoki_x = xpos+(Math.floor(Math.random()*kyori));//X 発生位置
                var syoki_y = ypos+(Math.floor(Math.random()*5));//Y 発生位置
                var syoki_r = Math.floor(Math.random()*20);// 角度
                var syoki_xyscale = 30+(Math.floor(Math.random()*40));// 大きさ
                var syoki_xsokudo = (Math.floor(Math.random()*20))-10;//X 方向と X 速度
                var syoki_ysokudo = 20+(Math.floor(Math.random()*20));//Y 速度
                // 回転方向と回転速度
                if (0 == Math.floor(Math.random()*2)) {
                     var syoki_rsokudo = (5+(Math.floor(Math.random()*5)))/10;
                } else {
                     var syoki_rsokudo = ((5+(Math.floor(Math.random()*5)))*-1)/10;
                }
                var xgensoku = (85+(Math.floor(Math.random()*10)))/100;//X 減速
                var ygensoku = (87+(Math.floor(Math.random()*9)))/100;//Y 減速
                var xscal = 6+(Math.floor(Math.random()*20));//X 拡大率
                var yscal = 6+(Math.floor(Math.random()*20));//Y 拡大率
                // 上記設定を生成する MC に入れ込む
                var syoki = {_x:syoki_x, _y:syoki_y, _rotation:syoki_r, _xscale:syoki_
                            xyscale, _yscale:syoki_xyscale, syoki_xsokudo:syoki_xsokudo,
                            syoki_ysokudo:syoki_ysokudo, syoki_rsokudo:syoki_rsokudo,
                            xgensoku:xgensoku, ygensoku:ygensoku, xscal:xscal,
                            yscal:yscal};
                // ハートの MC 生成
                this.attachMovie("love", newname, i, syoki);
                // ハートの動作
                this[newname].onEnterFrame = function() {
                     if (this.syoki_xsokudo != 0) {
                          this.syoki_xsokudo *= this.xgensoku;
                     } else {
                          this.syoki_xsokudo = 0;
                     }
                     if (this.syoki_ysokudo>=5) {
                          this.syoki_ysokudo *= this.ygensoku;
                     } else {
                          this.syoki_ysokudo = 5;
                     }
                     this._x += this.syoki_xsokudo;
                     this._y -= this.syoki_ysokudo;
                     this._rotation += this.syoki_rsokudo;
                     this._xscale += this.xscal;
                     this._yscale += this.yscal;
                     this._alpha -= 3;
                     // 透明度が 0 になったら消滅する
                     if (this._alpha<=0) {
                          this.removeMovieClip();
                     }
                };
           }
      } else {
           cut_pk = 0;// カット可能状態にする
           day += Math.round(ed*1.17);// 寿命が減る日数に加算
           hand.gotoAndPlay("end");// 手が戻る動画へ
```

次のページに続く

```
                this.gotoAndPlay("main_end");// 終了
                delete onEnterFrame;// この処理停止
           }
      };
}
```

工夫としては、徐々に発生数が減るようになっているところ（A の部分）にあります。この部分では、残りのハートの数から、1 回の発生数を決めています。この部分がないとハートの飛沫がいきなり止まる表現になってしまいますので、表現が稚拙になってしまいます。

ハートの初期値が非常に大きかったり、徐々に発生数を少なくなるようにしたりと、どちらもあまり意味のないようなことに思えますが、小さな積み重ねが全体としてクオリティアップにつながるので、このような部分はなるべく手を抜かないように心がけています（ホントは手を抜きたいのだけど…）。

TIPS 変数名やインスタンス名の名前付け

変数名やインスタンス名の名前付けには、半角英数文字、先頭は必ずアルファベット（大文字、小文字どちらでも可）、あるいは「_」（アンダーバー）、また予約語は使えないなどの基本ルールがありますが、慣習的なルールとしてあげられるのが、先頭はアルファベット小文字、単語同士をつなぐ部分には大文字を使用するというもの。たとえば「minoruGoodJob」のように付けます（Flash のスクリプトでもこんな感じですね）。こんな感じにするメリットというのは、「コイツすげーデキそう！」と見た人に思わせることにあります。だって「minoru_ii_shigoto」って書くとバカそうでしょ？（ヲイ！）ホントは複数人で作業するときのルールとしてです。「名前の付け方もルールで定義しておかないと、みんながわからなくなっちゃうもんね」と誰かが言い出したんでしょう。

でも僕は、「単語区切りの大文字」はすごく間違えやすいのでやらないようにしています。一時期チャレンジしたこともあるのですが、ひどく間違えて時間的損失が大きかったので、使うのやめました。そもそも一人で作っているので意味がなかったです…。で、何がいいたいかというと、一人で制作している人は自分がわかればよいと割り切って名前を付けるようにしましょうということ。結果、動いたら問題ないです。