イメージマジック v6 の 例 -- アニメーションの基礎

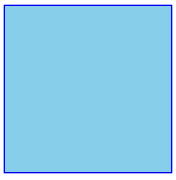
インデックス

これらの例は、複数の画像のレイヤーの前の例ページを続けますが、複数の画像を重ねて単一の画像を生成するのではなく、画像のアニメーションを生成するために各画像を短時間表示します。 次のセクションでは、アニメーションと特に GIF アニメーションの複雑さの基本的な理解を提供します。アニメーションの生成に使用される基本的な方法と、既存のアニメーションを調べ、アニメーションの動作を理解する方法を紹介します。 この方法は、後のアニメーション セクションの詳細を参照する前に、この資料を読むことをお勧めします。

GIF アニメーションとアニメーション のメタ データ

ImageMagick がイメージ リストの出力を処理する既定の方法は、マルチページ イメージを生成することです。ただし、GIF イメージ形式の場合は、この形式は "GIF アニメーション" という特殊な形式になります。

```
convert -delay 100  -size 100x100 xc:SkyBlue ¥
        -page +5+10  balloon.gif   -page +35+30 medical.gif  ¥
        -page +62+50 present.gif   -page +10+55 shading.gif  ¥
        -loop 0  animation.gif
```



シェルスクリプト"star_field"を使用する、より高度な「輝き」の例を次に示します。このスクリプトは、ランダムな星のフィールドを生成する私の実験から開発されました。

```
star_field 70x46  stars1.gif
star_field 70x46  stars2.gif
star_field 70x46  stars3.gif
convert rose:   -compose Screen ¥
        ¥( -clone 0 stars1.gif -composite ¥) ¥
        ¥( -clone 0 stars2.gif -composite ¥) ¥
        ¥( -clone 0 stars3.gif -composite ¥) ¥
        -delete 0 -set delay 25 -layers Optimize rose_sparkle.gif
rm stars[123].gif
```

基本的に3つのランダムな星のフィールドが生成され、適切なサイズで、その後、私たちの画像に重ね合わされたIM組み込み""、''アルファ組成を使用して、与えられた星のパターンで画像を明るくします。その後、IM 一般的な GIF アニメーション オプティマイザを使用して、全体が実行されます。 上記は、まだ紹介していないいくつかの高度なIM機能を使用しているので複雑に見えるかもしれませんが、結果は比較的シンプルですが、3フレームのアニメーションが最適化されています。 また、歪みアニメーションの単純なシェルスクリプトを使用して作成された、より複雑なアニメーションのいくつかを見ることもできます。 GIF アニメーションで使用するために特別に作成されたいくつかの追加の IM 設定があり、これらについて知ることは、GIF アニメーションの世界への最初のステップです。rose:Screen

-dispose {method}
: GIF アニメーションの以前の結果に対して、次の画像が実行する必要があります。有効なオプションは''、''、'、'、'です。(設定の説明については下記を参照)UndefinedNonePreviousBackground

-loop {number}
: GIF アニメーションが停止する前にイメージ シーケンスを循環する回数。出力の「イメージ書き込み」設定なので、コマンドラインの任意の場所で設定できますが、最後の設定だけが使用されます。 通常、この値はデフォルトでゼロ(無限ループ)に設定されますが、読み込まれたイメージの値が異なる場合、この設定はそのイメージ値に設定されます。そのため、すべての画像が読み込まれた後、GIFアニメーションを作成する際には常に""を設定することをお勧めします。 詳細については、以下の「ループの終わり」を参照してください。

  -loop

-delay {time}
: この設定を定義した後に読み込まれたイメージまたは作成されたイメージを描画した後に、一時停止する時間遅延(1/100 秒)を設定します。 '' スケーリングを指定することで、時間遅延に異なるスケールを指定できます (1 秒あたりのティック数を指定します)。例えば''は10、1秒ティック、''は10、2番目のティックの100分の1です。 基本的に '' は分数 '' 記号に相当します。たとえば、'' を指定すると、160フレーム/秒に適した遅延が設定されます。

  x10x110x100

  x/1x160

  
  GIF アニメーションの遅延は、正しく動作するために *100* 分の *1* 秒で指定する必要があります。x要素は、*MNG* や *AVI* のような、他のムービー形式などのムービーを生成するためにより多く使用されます。

-set dispose {method}
-set delay {time}
: 前のオプション設定では、新しく作成されたイメージ属性、または読み込まれたイメージにイメージ属性が設定されますが、そのオプションが指定された後は、""オプションはオペレータであり、現在のイメージ シーケンスに既に存在するすべてのイメージにイメージ属性を設定

できます。これにより、イメージの読み込みまたは変更後に、アニメーション全体または単一のフレームで設定を変更できます。-set

-page {w}x{h}+{x}+{y}

これにより、読み込まれる画像のオフセット位置を設定できます。これは設定オプションであるため、設定に従って指定した画像にのみジオメトリが適用されます。既にメモリに読み込まれたイメージには影響しません。

指定しない場合、または「」を使用してオフにすると、読み取られた画像のオフセットが保持されます。画像にオフセットがない場合は、'' または作業キャンバスまたは 'ページ' の左上隅に配置されます。+page+0+0

幅の 「'」高さを指定することで、より大きな作業キャンバスを定義するためにも使用できます。シーケンスの最初の画像の幅と高さのページ設定のみが、GIF アニメーション キャンバス全体のサイズを設定するために使用され、アニメーションが最終的に書き込まれると、その他のページ サイズ設定はすべて無視されます。GIF アニメーションが読み込まれると、アニメーション内のすべてのフレームに対して、キャンバスサイズが設定されます。 MNG アニメーションではフレームオフセットを保存できますが、キャンバスサイズは保存されません。最初の画像のサイズは、アニメーション全体のキャンバスサイズを定義します。x

GIF 画像形式では、キャンバス上の画像に負のオフセットを指定することはできません。負のオフセット *IM* を使用すると、その画像 *(*またはアニメーション フレーム*)* が *GIF* ファイルに書き込まれるときに、*IM* がゼロにリセットされます。 イメージ キャンバスよりも大きい正のオフセットは許容されますが、表示時にイメージがキャンバス描画領域に表示されないことがあります。*GIF* アニメーション表示プログラムがこれを処理する方法は未定義です。注意をお勧めします。

-repage {w}x{h}+{x}+{y}

これは、設定ではなくイメージ演算子であることを除けば、まったく同じです。つまり、既にメモリに読み込まれているイメージまたはアニメーション フレームの 'ページ ジオメトリ' を変更またはリセットできます。 より単純な""形式は、現在の画像シーケンスの各フレームの実際の画像に対して、すべての画像の「ページジオメトリ」をゼロオフセットにリセットし、実際のサイズにリセットするだけです。この操作は、アニメーションから個々のフレームを抽出する場合に重要です(後述の「Adjoin の例」を参照)。 しかし、"" は各画像に格納されている多くの位置情報を破棄するので、後で再利用するためにこの情報を別のファイルに抽出する必要があります。以下のアニメーションリスト情報を参照してください。-page

+repage

+repage

重点

処理を終えていない中間のアニメーションをGIFに直接保存しないでください。一連の処理手順でアニメーションを操作する場合は、IM 内部形式 MIFF を一時ファイル形式として使用できます。 繰り返します。。。

中間ファイル形式として GIF を使用しない、代わりに MIFF を使用する

GIF に保存するという大きな間違いを犯した場合、IM が自動カラー量子化を実行して存在する色の数を減らすように、結果として得られるアニメーションを悪化させただけです。それだけでなく、各フレームで完全に独立して他のフレームに対してそうし、それ以上の処理、特にGIFの最適化はそれほど難しくなりました。 これを解決することは複雑なマルチレベルの問題で、次のセクション「アニメーション最適化」で見られます。

フレーム廃棄方法

GIFアニメーションを作成する人が最初に問題を抱えているのは、「」設定です。これは複雑な設定であるため、驚くべきことではありません。さらに悪いことに、多くの Web ブラウザを含む多くのアニメーション プログラムは、必ずしも GIF 廃棄メタデータ設定を正しく処理するとは限りません。しかし、適切な廃棄を使用すると、アニメーションの動作と最適化の効果に大きな違いが生じます。 ImageMagickで最初に覚えておくべきことは、ほとんどすべての特別なアニメーションオプションは、画像の読み取りのための設定です。つまり、読み込まれた後、設定が与えられている画像に適用されます。""設定は、アニメーションが保存される直前に、アニメーションが完了した後に通常使用される唯一の設定です。 「」の基本的なタスクは、画像が「」期間に表示された後に、どのように削除するかを定義します。つまり、フレームの画像を読み取る前に、画像の"""と""の設定を与える必要があります。ただし、そのイメージが表示された後にアクションが適用されます。これは少し直感的ではありませんが、IMが画像に対して動作する方法では意味があります。 あなたがこれを覚えているなら、あなたは何の問題もないはずです。 IM の他のほとんどの設定と同様に、これらのオプションの 「プラス」 形式は、読み取り中の画像に適用される設定を停止します。つまり、設定を指定しない場合、フレームイメージはイメージと一緒に読み込まれた設定を引き続き使用します(存在する場合)。 これは、後で GIF アニメーションを読み取って処理を続ける場合に重要になります。または、ある GIF アニメーションを別の GIF アニメーションにマージする場合 (最も難しいアニメーション手法)。-dispose

-loop

-dispose-delay-dispose-delay

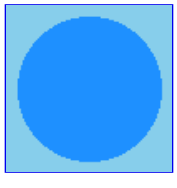
破棄なし- 各フレームが順番に重ね合

GIF アニメーションのデフォルトの "" 設定は '' で、ほとんどのアニメーションプログラムは''廃棄設定と同じように扱います。基本的に、これはコンピュータに、この特定のフレームでオーバーレイされているものは何でも残すように指示します。より正確には、「何もしない」。 ただし、キャンバス全体は、ループと繰り返しの前に、アニメーションシーケンスの最後に常にクリアされることに注意してください。 たとえば、標準の「なし破棄」アニメーションです。-disposeUndefinedNone

```
convert -delay 100 -dispose None ¥
            -page 100x100+5+10  balloon.gif  ¥
            -page +35+30 medical.gif  ¥
            -page +62+50 present.gif  ¥
            -page +10+55 shading.gif  ¥
        -loop 0  anim_none.gif
```

この処理手法は、ソリッドやパターンの背景に描画されるアニメーションなど、透明な形式を含まないアニメーションに最適です。

```
convert -dispose none  -delay 100 ¥
              -size 100x100 xc:SkyBlue +antialias ¥
              -fill DodgerBlue -draw 'circle 50,50 15,25' ¥
              -page +5+10  balloon.gif  ¥
              -page +35+30 medical.gif  ¥
              -page +62+50 present.gif  ¥
              -page +10+55 shading.gif  ¥
        -loop 0  canvas_none.gif
```



この手法では、アニメーションに表示される色しか追加できません。アニメーションの一部を再び透明にすることはできません。(下の「アニメーションをオーバーレイする」を参照してくだ

さい)。 透明度を処理するには、他の種類の破棄方法のいずれかを使用する必要があります。

前に破棄- バックグラウンド キャンバスを保持

'' 廃棄方法は比較的単純です。現在の画像が完成したら、画像がオーバーレイされる前の状態にキャンバスを戻します。前のフレームイメージも '' 破棄メソッドを使用した場合、結果はそのフレームの前と同じになります。等。。等。。等。。。 たとえば、このアニメーションでは、後のフレームの各フレームは、そのフレームに関連付けられたイメージをオーバーレイする前に、'破棄設定を持つイメージの最初のフレームに戻ります。 その結果、各フレームイメージが、そのイメージの持続時間だけ重ね合った背景キャンバスになります。PreviousPrevious

None

```
convert -dispose none  -delay 0 ¥
              -size 100x100 xc:SkyBlue +antialias ¥
              -fill DodgerBlue -draw 'circle 50,50 15,25' ¥
        -dispose previous -delay 100 ¥
              -page +5+10  balloon.gif  ¥
              -page +35+30 medical.gif  ¥
              -page +62+50 present.gif  ¥
              -page +10+55 shading.gif  ¥
        -loop 0  canvas_prev.gif
```

最初の画像に使用される "" メソッド '' に注意してください。これは重要です、そうでなければ'前の'フレームは最初のフレームの前に存在していた元の空のキャンバスに戻ります。 また、上記のアニメーションでは''の「」を使用していることにも注意してください。これは、この「背景キャンバス」に最初のフレームをオーバーレイする前に待つなと言います。それがなければ、その上に何もないキャンバス画像だけを示す短い遅延が表示されます。 もちろん、私はまだ後の画像のために長い ""を設定する必要があります、または彼らは目のウインクで表示され、消え、ついでに視聴者のCPUサイクルの多くを使い切ります。 '' 破棄方法の使用は、若干のちらつきや、特に低速なマシンでは、一部の Web ブラウザーで一時停止を起こす可能性があります。それは最近めったに見られないが、ちらつき自体はまだ存在し、私はバグであると考えるもの。詳細については、以下の「ゼロ遅延フレーム」を参照してください。 以前のスタイルのアニメーションを使用するアニメーションはほとんどありませんが、その理由はコンピュータが最適化するのが非常に困難であるためです。問題は、コンピュータが背景画像になるために選択すべきフレームだけですか?私たち人間が使用するのに最適な画像を見つけるのは簡単ですが、コンピュータが決めるのは難しいです。アニメーションで使用する最適な背景画像は、現在の例のように表示される意味でなくても、最適化されていないバージョンのアニメーションには存在しない可能性があります。-disposeNone

-delay0

-delay

Previous

背景を破棄- 背景にクリア

最初の 2 つの "" メソッドは比較的単純ですが、'' はおそらく理解するのが最も難しいです。 特定のフレームの遅延時間が終了すると、そのフレームによってオーバーレイされていた領域がクリアされます。キャンバス全体ではなく、オーバーレイされた領域だけです。それが完了すると、結果のキャンバスは、そのフレームイメージによってオーバーレイされるアニメーションの次のフレームに渡されるものです。 ここでは、各フレームを次のフレームに置き換えるだけです。-disposeBackground

```
convert -delay 100 -dispose Background ¥
            -page 100x100+5+10  balloon.gif  ¥
            -page +35+30 medical.gif  ¥
            -page +62+50 present.gif  ¥
            -page +10+55 shading.gif  ¥
        -loop 0  anim_bgnd.gif
```

そのため、何が起こっているのかを正確に確認でき、アニメーションに最初のキャンバスイメージを追加できるため、アニメーション表示からそのフレームがどのように実際に「破棄」されるかを確認できます。Background

```
convert -delay 100 -dispose none ¥
              -size 100x100 xc:SkyBlue +antialias ¥
              -fill DodgerBlue -draw 'circle 50,50 15,25' ¥
        -dispose background ¥
              -page +5+10  balloon.gif  ¥
              -page +35+30 medical.gif  ¥
              -page +62+50 present.gif  ¥
              -page +10+55 shading.gif  ¥
        -loop 0  canvas_bgnd.gif
```

オーバーレイされた各フレームが破棄される場合は、次のイメージがオーバーレイされる前に、そのフレーム領域が透明にクリアされます。これは、アニメーションのフレーム履歴に関係なく、GIF アニメーションが任意のピクセルをクリアできる唯一の方法であるため、この GIF 破棄メソッドの重要性です。 ピクセルをクリアする唯一の方法は、''を使用して、それらのピクセルがクリアされたフレームに戻る方法です。しかし、それはアニメーションシーケンスの歴史を知ることに依存しており、コンピュータの最適化がはるかに困難になります。

Previous

透明な色にオーバーレイ領域をクリアするのではなく、この処分は*GIF*アニメーションに格納された「背景色」のメタデータ設定にそれをクリアする必要があるという考えがあります。実際には、古い "" ブラウザ *(バージョン 2 と 3)* は、まさにそれをしました。しかし、その後も '' 破棄メソッドを正しく実装できませんでした。 一方、最初のキャンバスも「背景」の色の形式から設定する必要があり、それも行われません。しかし、すべての現代の*Web*ブラウザは、これが今受け入れられているように、透明性に重ねられていた領域だけをクリアし、*IM*は現在何に従っています。NetscapePrevious

IM バージョン *6.2.6-1* 以前は、*IM* "" と "" 操作は、すべての主要な *Web* ブラウザーに従って、ピクセルを透明にするために '' 破棄を使用するアニメーションを処理しませんでした。例と詳細については、アニメーションバグを参照してください。 しかし、ピクセルクリアが適用されていないか、意図されていない場合、これらの関数は正常に動作しました。これは「」で修正され、*GIF*アニメーションフレーム最適化機能として"""の使用を置き換える"メソッドが作成されました。-coalesce-deconstructBackground

-coalesce-layers OptimizeFrame-deconstruct

アニメーションの研究

GIF アニメーションの基本、その種類、最適化、および処理の手法を続ける前に、既存のアニメーションを研究するためのいくつかのテクニックが必要です。

識別- アニメーションに関する情報

アニメーションは、各フレームに詰め込まれた多くの情報で構成されています。この情報の一部は、デフォルトの IM "" コマンドを使用して確認できます。identify

```
identify canvas_prev.gif
```

```
canvas_prev.gif[0] GIF 100x100 100x100+0+0 8-bit PseudoClass 2c 1.42kb
canvas_prev.gif[1] GIF 32x32 100x100+5+10 8-bit PseudoClass 16c 1.42kb
canvas_prev.gif[2] GIF 32x32 100x100+35+30 8-bit PseudoClass 8c 1.42kb
canvas_prev.gif[3] GIF 32x32 100x100+62+50 8-bit PseudoClass 8c 1.42kb
canvas_prev.gif[4] GIF 32x32 100x100+10+55 8-bit PseudoClass 4c 1.42kb
```

上記のような出力が表示されない場合は、*IM*は少し古いです、そして、あなたは本当に最新のバージョンに、インストールされたバージョンの*ImageMagick*をアップグレードする必要があります。あなたがしない場合は、*IM*の*GIF*アニメーションの処理と制御の新しい進歩の多くを見逃すでしょう。

2 番目以降のフレームに保存された実際のイメージは 32 x 32 ピクセルですが、すべてのフレームは 100 x 100 ピクセルの 「仮想キャンバス」に置かれ、その大きなキャンバスに「仮想オフ

セット」が付きます。 メタデータのさまざまなビットを表示するには、特殊化されたパーセントエスケープ形式のいくつかを使用して、IM に出力を取得する必要があります。

```
identify -format "%f canvas=%Wx%H size=%wx%h offset=%X%Y %D %Tcs¥n" ¥
         canvas_prev.gif
```

```
canvas_prev.gif canvas=100x100 size=100x100 offset=+0+0 None 0cs
canvas_prev.gif canvas=100x100 size=32x32 offset=+5+10 Previous 100cs
canvas_prev.gif canvas=100x100 size=32x32 offset=+35+30 Previous 100cs
canvas_prev.gif canvas=100x100 size=32x32 offset=+62+50 Previous 100cs
canvas_prev.gif canvas=100x100 size=32x32 offset=+10+55 Previous 100cs
```

これは、キャンバスのサイズ、画像サイズ、オフセットだけでなく、個々のフレームに使用される処分と遅延時間も明確に示しています。最初のフレームが、後の '前の ' 破棄方法を適切に使用するために必要な、異なる破棄と時間の遅延を持っている方法に注意してください。

隣接- アニメーションをフレームに分割する

ここまで見たように、ImageMagick は、ファイル形式で許可されている場合、デフォルトで複数の画像を 1 つのファイルに保存しようとします。ただし、「マルチイメージ リストIM の作成」で説明したように、"" 設定を使用して、各イメージを個別のイメージとしてディスクに保存するように指示します。 たとえば、ここでは、GIF アニメーションの 1 つを読み取り、アニメーション シーケンス内の個々のフレーム イメージを出力します。+adjoin

```
convert canvas_prev.gif -scene 1 +adjoin  frame_%03d.gif
```

上記の実際の画像を調べる場合、ほとんどの Web ブラウザは 100x100 の領域を大きく表示していますが、各サブフレームが表示されます。実際のところ、実際の画像のほとんどは、上記の「識別」コマンドに示されているように、実際には32x32ピクセルに過ぎません。 それは、領域のほとんどは、画像「ページジオメトリ」または「仮想キャンバス」として知られている、何も描かされていないキャンバスです。アニメーションの最初の画像は、大きな 'canvas' と他のフレームごとに、この大きなキャンバス上の「オフセット」位置を定義します。 この追加情報は、"""設定で保存されたフレームに保存されます。そのように、あなたはGIFアニメーションを再構築することができます。ページ情報は、各フレーム イメージに保存されるだけでなく、遅延、ループ、および GIF 破棄の設定も保持されます。

+adjoin

つまり、アニメーションを再構築するには、すべての画像を読み取るだけで済みます。

```
convert frame_???.gif  anim_rebuilt.gif
```

ただし、このページのジオメトリ情報を保持したくない場合があります。たとえば、他のプロジェクトに個別のフレームを使用する場合などです。ページサイズとオフセットをリセットするには、「」オプションを使用して、実際の画像だけを残して「仮想キャンバス」情報を削除します。
通常、アニメーション サブイメージを抽出する場合は、通常、画像の遅延をリセットし、設定を破棄して、編集や表示に影響を与えないようにします。 たとえば、ここでは不要な仮想キャンバスを削除し、タイミングの遅延と廃棄を相殺してリセットします。+repage

```
convert canvas_prev.gif  +repage  -set delay 0   -set dispose None ¥
        +adjoin  repage_%03d.gif
```

もちろん、そのメタデータをジャンクした場合は、そのデータを記録して編集する何らかの方法が必要です。サブイメージの両方を抽出し、アニメーションのメタデータを保存するスクリプトについては、アニメーションの再ビルドに使用できる形式でアニメーションリスト情報(下)を参照してください。

合体- 完全にフレームを記入

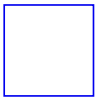
ただし、通常は、サブフレームの形でアニメーションを表示することは、一般的なアニメーションではあまり役に立ちません。 一つには、高度に最適化されたアニメーションは、それらが一緒に収まる方法を視覚的に示すことなく、非常に小さな部品の多くで構成することができます。また、圧縮最適化のために追加された他の多くのノイズを持ち、アニメーションのファイル全体のサイズを小さくすることもできます。 たとえば、アニメーションの個々のサブフレームを見るだけで、このアニメーションが実際に何をしたのかを理解することは非常に困難です。

```
convert script_k.gif  +repage  +adjoin  script_k_%02d.gif
```

""操作は、基本的に、前のフレームが正しく破棄され、次のサブフレームがオーバーレイされた後に、アニメーションの外観に正確にイメージを変換します。 これは、各フレームが前の「破棄された」フレームにオーバーレイされた変更のみを表すアニメーション シーケンスの代わりにです。このオペレータは、アニメーション シーケンスではなく、真のフィルム ストリップのような、各ポイントでアニメーションの完全なビューを作成します。このようなシーケンスは、合体アニメーションと呼ばれ、研究、編集、修正、再最適化が非常に簡単です。 例えば、上記と同じ「混乱」のアニメーションシーケンスのモンタージュを生成しますが、今回はシーケンスを「」行うので、実際に何が起こっているのかを見ることができます。-coalesce

-coalesce

```
montage script_k.gif -coalesce ¥
        -tile x1 -frame 4 -geometry '+2+2' ¥
        -background none -bordercolor none coalesce_k_montage.gif
```



あなたが見ることができるように、結果はアニメーションのフィルムストリップのようなもので、前の作品がどのように組み合わさって手描きの文字「K」を形成するかをはっきりと見ることができます。 IM バージョン 6.2.6 の時点で、""" コマンドは "" の使用を理解し、アニメーション フレームのイメージのような「フィルム ストリップ」を作成することができ、上記のとおりにしました。このバージョンには、合体の修正も含まれており、GIFアニメーションの作品は少なくともこのバージョン(または最新バージョン)でなければなりません。

montage-coalesce

> アニメーションを調べるためのより優れたモンタージュテクニックは、次の例のセクションで説明します。

合体したアニメーションでは、合体した画像シーケンスの""設定は、実際には無関係です。しかし、ユーザーの心の部分のために""オペレータは、合体された画像シーケンスが(上記に示すよう

に)正しくアニメーションを続けるように、各フレームの「」設定を必要に応じて''または''のいずれかに設定します。-dispose-coalesce-disposeNoneBackground

'' の破棄を含むフレームとは、次のフレームが正しく表示するために、少なくとも *1* つ以上のピクセルをクリアする必要があるということを意味します。 「」が「破棄」を加えたアニメーションとは、アニメーションを単純なオーバーレイアニメーション(下記参照)として保存できないということを意味します。 技術的には、結合されたイメージ シーケンスのすべての破棄設定を '' または '' に設定して、クリアされたフレーム アニメーションを生成できます(下記参照)。すべてのアニメーションがその形式でうまく最適化されるわけではありませんが。Background

-coalesceBackground

BackgroundPrevious

また、"" 演算子のアニメーション以外の用途もあります。これらの用途の「合体」および「進行性平坦化」の例を参照してください。-coalesce

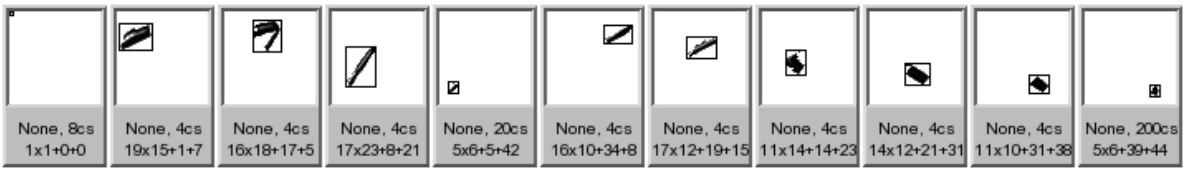
アニメーションフレームモンタージュ-「gif_anim_montage」スクリプト

"" 演算子を使用すると、アニメーションから実際のイメージを抽出でき、"" はアニメーションの結果のフレームを見ることができますが、どちらの方法でもアニメーションに関する多くの情報が省けます。 アニメーションイメージを慎重に操作することで、実際のフレームだけでなく、大きなキャンバス上でのフレームの配置も表示するようにフレームを表示できます。ここでは、アニメーションを表示する方法の1つです。+adjoin-coalesce

```
convert -dispose Background   script_k.gif  -matte ¥
        -compose Copy -bordercolor black -border 1x1 -compose Over ¥
        -coalesce  -bordercolor none   -frame 4x4+2+2 ¥
        -bordercolor none -border 2x2 +append  script_k_parts.gif
```

ここでは、アニメーションの動作を明確に確認できます。各サブフレーム イメージは、前のすべてのオーバーレイに追加されるように配置されます。その結果、ゆっくりと成長している画像が表示されます。各フレームは、それが配置されている「仮想キャンバス」よりもはるかに小さいです。 私はGIFアニメーションの開発とデバッグの間にこの表示技術をたくさん使用し、シェルスクリプトに変換しました"gif anim montage"、 そして、アニメーションの各フレームの上の詳細の一部も一覧表示するために拡張しました。

```
gif_anim_montage   script_k.gif   script_k_frames.gif
```



さまざまなフレームで使用されるタイミングのバリエーションに注意して、ペンがページから持ち上げられ、再配置されているかのように一時停止します。タイミングが変化するアニメーションは、最も興味深いものの一部ですが、後の IM の例ページで見るように、処理が難しくなります。 ""スクリプトはまた、合体アニメーションの半透明のコピーを下敷きにする特別なオプション'' も。これにより、新しいサブフレームが表示されるアニメーションをどのように変更するかを確認できます。

gif anim montage-u

```
gif_anim_montage  -u  script_k.gif  script_k_frames.png
```

もちろん、これは半透明のピクセルを持っているので、「PNG」画像形式が必要でした、 またはスクリプトが提供する多くの「背景」オプションのいずれかを使用して、アニメーションの夏のイメージにGIFまたはJPEGフォーマットを使用することもできます。 その他のオプションでは、使用する行または列の数を定義したり、透明でないさまざまな背景を設定したり、デフォルトの黒ではなく赤いボックスを使用したりできます。 このスクリプトは、IM の例の次の数ページで多くの使用されます。提案やコメントは歓迎します。

アニメーションリスト情報- アニメーションの構築に使用されるオプション

私が述べたように、「」と「」を使用すると、「」は、すべての抽出とGIFアニメーションを見て便利な方法です。しかし、それらはすべて、その過程で元のアニメーションに関する情報を破棄します。 この追加情報は、IM ""コマンドと"""オプションを使用して、フレーミング、遅延時間、フレーム廃棄などに関する追加情報を参照できます。しかし、私は、おそらく他のほとんどのユーザーは、このコマンドからの出力を見つけます, 圧倒的で、実際には直接使用できません. これは私が書いた別の特別なシェルスクリプトが入ってくるところです。"gif2anim"スクリプトは、アニメーションの個々のフレームを分離しますが、これらのイメージからアニメーションを再構築するために必要な IM "オプションを正確に把握します。 "" はアニメーション逆アセンブラと考えることができ、IM オプションの観点からアニメーションの概要を生成できます。 たとえば、作成に使用された元の ""設定と使用された個々の画像を復元するために使用してきたアニメーションの例をデコードできます。+adjoin-coalesce+repage

identify-verbose

convert

gif2anim

convert

```
gif2anim canvas_prev.gif
```

```
#
# Animation Sequence File "canvas_prev.gif"
# using a BASENAME of     "canvas_prev"
# Extracted on    2007-05-12 17:36:43
#
-loop 0
-page 100x100
-dispose None
                BASENAME_001.gif  # 100x100+0+0
-delay 100
-dispose Previous
-page +5+10     BASENAME_002.gif  # 32x32+5+10
-page +35+30    BASENAME_003.gif  # 32x32+35+30
-page +62+50    BASENAME_004.gif  # 32x32+62+50
-page +10+55    BASENAME_005.gif  # 32x32+10+55
```

デフォルトでは、"" スクリプトは、個々のイメージと "" オプション ファイルに同じベース ファイル名を使用します。このように、上記のコマンドで生成されたアニメーションシーケンスファイルは、個々のフレーム画像を「」から「」にして「」という名前が付けられます。 結果を詳しく調べると、この GIF アニメーションを最初に作成したときに使用した元のオプションを実際に再作成できたことがわかります (「前のアニメーションを破棄する」を参照)。また、実際にアニメーションを生成することは重要ではありませんが、オーバーレイされたフレームのサイズとタイミングもコメントとしてリストされ、学習しやすくなります。 結果をファイルに保存するのではなく、アニメーション シーケンスオプションを画面にリスト表示するには、"" フラグを使

用します。これは、アニメーション シーケンス ファイルを保存するのではなく、アニメーション シーケンス ファイルを出力するだけ、またはアニメーションの個々のフレーム イメージを出力するだけです。gif2anim.animcanvas_prev.animcanvas_prev_001.gifcanvas_prev_005.gif

-l

```
gif2anim -l canvas_prev.gif
```

"" ファイルと個々のフレーミング イメージを指定すると、アニメーションを再構築するために補完的なスクリプト"anim2gif"を使用できます。.anim

```
anim2gif canvas_prev.anim
```

デフォルトでは"" は""""接尾辞を付けて GIF アニメーションを再作成します。生成された "" アニメーションが生成され、元のアニメーションとまったく同じように見え、動作することがわかります。 このスクリプトは、単に"アニメーションシーケンスファイル"で使用される特別な文字列"を置き換え、すべてのコメントを取り除き、その後、""コマンドに残された変換オプションを渡すだけです。つまり、上記のファイルをコメント付きの 'convert' スクリプトのタイプとして扱います。 特別な文字列が使用された理由は、この場合、"" ファイル自体の名前とは異なるベース ファイル名を指定できるためです。このようにして、オリジナルの修正版など、まったく異なるフレームイメージのセットを使用して、古いアニメーションとは異なるアニメーションを再作成できます。これは非常に便利な機能で、より複雑なアニメーション処理に使用されます。(例については、アニメーションを並べて追加するを参照してください)。 """と同様に、"" スクリプトには、アニメーションの処理と変更に役立つ便利なオプションが多数あります。これらのオプションの一部は後で使用されます。例については、「アニメーションの追加」を参照してください。 また、""" ファイルはプレーン テキストなので、アニメーションのデコードされたイメージを使用して、タイミング、位置、アニメーションの繰り返しセクション、アニメーションへの新しいフレームやイメージの追加など、GIF のメタデータを調整できます。これは、私がIMの例に関わるずっと前に、私が最初にスクリプトを書いた理由の後でした。 ここでは、アニメーション シーケンスを調べて、何が起こっているのか、フレーム間で適用されているタイミングを確認するのに""が最も役立ちます。anim2gif_anim.gifcanvas_prev_anim.gif

BASENAMEconvert

.anim

gif2animanim2gif

.anim

gif2anim

イメージを破棄する - GIF 破棄形式のフレーム

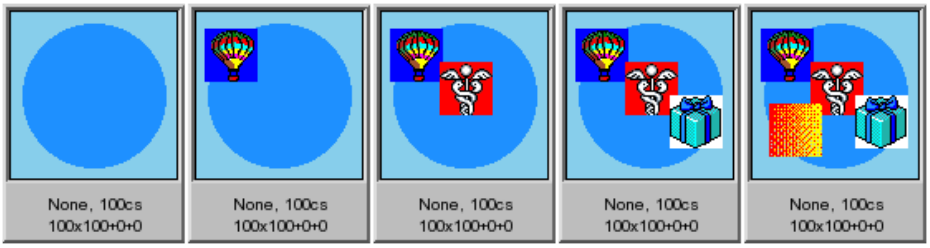
この特別な "" メソッド '' は、時間の遅延が終了した後にフレームの外観を示し、GIF 破棄メソッドが適用されていますが、次のフレーム イメージがオーバーレイされる前に行われます。 つまり、これは、GIF "" メソッドの設定が実際にフレームに何を行うかを正確に示し、アニメーションの問題を正確に把握できます。 たとえば、個々のフレーム破棄メソッドが適用された後の、3 つのDispose メソッドの例アニメーションのそれぞれについて説明します。これらのアニメーションは、それぞれ「キャンバス画像」で構成され、「キャンバス画像」に「」設定が設定され、その後に4つの小さな画像がオーバーレイされ、その後、さまざまなGIF破棄メソッドによって破棄されたことを覚えておいてください。 '' はアニメーションを破棄します。-layersDispose

-dispose

None-dispose

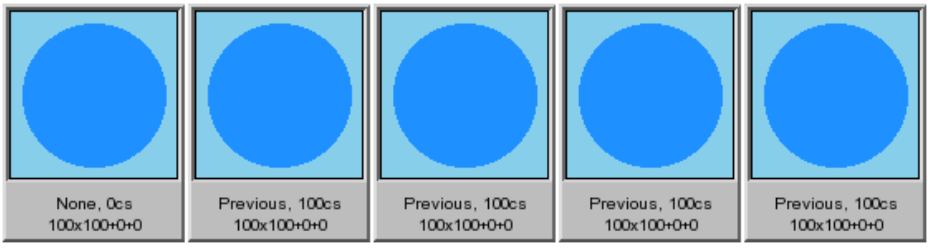
None

```
convert  canvas_none.gif -layers Dispose canvas_none_dispose.gif
gif_anim_montage canvas_none_dispose.gif canvas_none_dispose_frames.gif
```



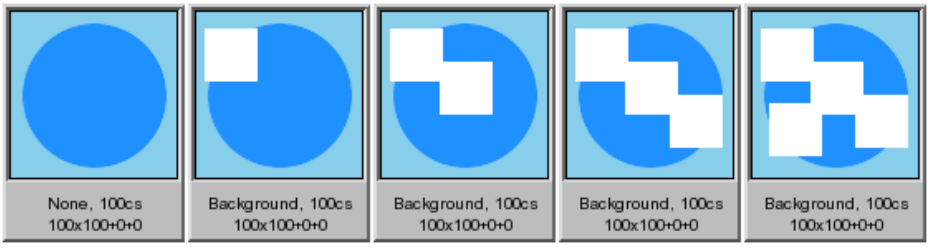
'Previous' 破棄アニメーション.

```
convert  canvas_prev.gif -layers Dispose canvas_prev_dispose.gif
gif_anim_montage canvas_prev_dispose.gif canvas_prev_dispose_frames.gif
```



'Background' 破棄アニメーション.

```
convert  canvas_bgnd.gif -layers Dispose canvas_bgnd_dispose.gif
gif_anim_montage canvas_bgnd_dispose.gif canvas_bgnd_dispose_frames.gif
```



上記を調べれば、3つのGIF破棄メソッドのそれぞれが、そのフレームのアニメーションをオーバーレイした画像をクリアする方法を正確に確認できます。これらの 3 つのアニメーションの最初のフレームは、常に '' の破棄に設定されるので、変更されないままであることに注意してください。後のフレームにおける破棄メソッドの効果が刺激性である。None

> "-layers Dispose"操作は、破棄フレームの "結合された" シーケンスのみを生成します。廃棄設定自体はリセットされないため、結果が正しくアニメーション化されない場合があります。上記のアニメーションを正しく行うために、すべての破棄方法を '前の ' または '背景' に設定するか、またはアニメーションを最適化してから保存します。

> *GIF* 破棄メソッドの後の最終フレームの外観は、通常、アニメーションが繰り返される前にキャンバス全体が完全にクリア *(ループ)* するので、*GIF* アニメーションに影響を与えない。'ループ' しないが、アニメーション シーケンスの最後に停止する場合は、最終的なフレームの破棄は適用されません。 つまり、上に示した最後のフレーム*(*廃棄後*)*の外観、または最後のフレームの実際の破棄設定は、*GIF*アニメーションに影響を与えます。*IM* は通常、フレーム最適化アニメーション中に適切な破棄方法を実行しようとする場合に、この値を前のフレームと同じに設定します。

分解- フレームの差異のレポート領域

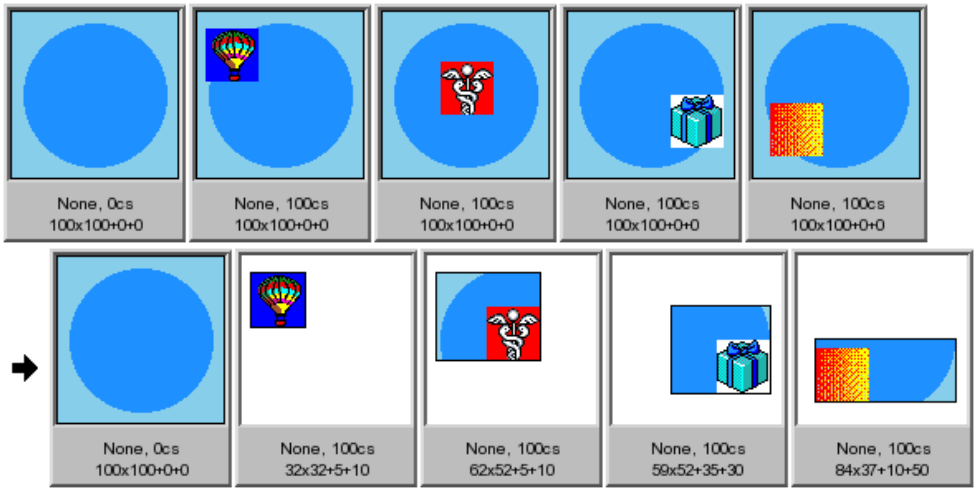
ImageMagick でアニメーションを最適化する従来の方法は、結果をダウンロードしてアニメーション化する際に、その "" 形式 " を "" より小さく、より高速にすることです。これはもはやお勧めしません。代わりに、一般的な GIF オプティマイザーを使用する必要があります。 このオペレータは、一連の画像 (実際に表示されるときにアニメーション フレーム) を取り、2 番目以降の画像を前の画像と比較します。そのイメージを、変更されたピクセルの最小の長方形領域に置き換えます。ピクセルの変更は、カラー変更(オーバーレイ)かクリア(消去)かに関係なくカウントされます。 これは非常に簡単で、一般的なオーバーレイ アニメーションでは、そのアニメーションに最適なフレーム最適化が生成されます。ただし、オーバーレイ アニメーションでは''

破棄メソッドのみを使用します。 たとえば、上で生成した前のアニメーションを合体させ、オーバーレイ アニメーションを形成し、""演算子を使って実行します。-deconstruct-coalesce

None

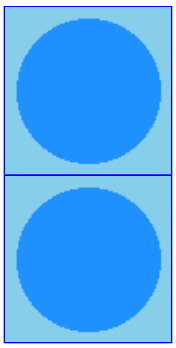
-deconstruct

```
convert  canvas_prev.gif   -coalesce     coalesce.gif
convert  coalesce.gif      -deconstruct  deconstruct.gif
gif_anim_montage  coalesce.gif     coalesce_frames.gif
gif_anim_montage  deconstruct.gif  deconstruct_frames.gif
```



最後の破棄フレーム (この場合は最初の背景キャンバス) まで、各フレームをクリアしたことを覚えていれば'前の破棄アニメーション' 。ご覧のとおり、「」は、1つの結合されたフレームから次のフレームに変更された領域を返しました。オーバーレイアニメーションが最適化され、特別な破棄設定は必要ありません。 これは、私が始めたオリジナルの手で生成されたアニメーションほど最適なものではありませんが、それ自体が便利です。-deconstruct

残念ながら""はGIFアニメーション""設定を全く理解していません。したがって、上で作成した (左図を参照) の背景の破棄されたアニメーションなど、あるフレームから次のフレームにピクセルをクリアするアニメーションでこれを試してみると、失敗します。-deconstruct-dispose



ここでは、ちょうど示したアニメーションを取り、"と""サイクルを使用して実行します。-coalesce-deconstruct

```
convert canvas_bgnd.gif  -coalesce  -deconstruct  deconstruct_erase.gif
```

「」を見ることができるように、ゆっくりとアニメーションを破壊します。基本的には""は、単に画像レイヤー間の違いを見つけるために設計されています。アニメーションを正しく最適化するように設計されたことはありませんし、以前にオーバーレイされたピクセルをクリア(消去または透明にする)するためにさまざまな廃棄手法を使用する必要があるアニメーションでは失敗します。-deconstruct-deconstruct

フレーム比較- フレームのより詳細な比較

IM v6.2.6-2では、いくつかの追加のGIFフレーム比較メソッドが追加されました。 これらは、アニメーションを適切に最適化するために内部的に必要とされていましたが、コマンド ラインや他の API インターフェイスで使用できるほど役に立つと考えられていました。

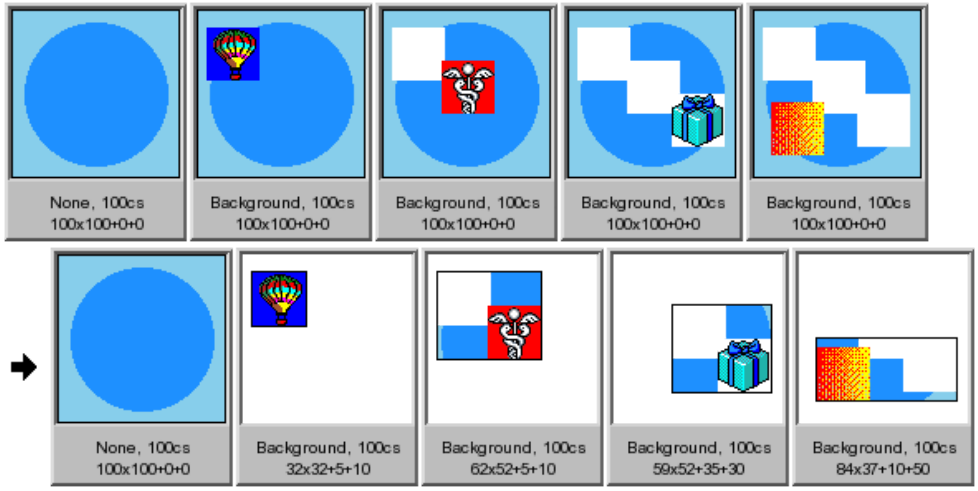
Compare_Any

"" メソッド '' は、実際には "" とまったく同じです。実際には、"" 演算子は '' メソッドの機能別名にすぎません。 ここでも、'バックグラウンドで破棄された' アニメーションの '分解' または '' の実際のイメージ結果を見ることができます。 -layersCompareAny-deconstruct-deconstructCompareAny

CompareAny

```
convert canvas_bgnd.gif  -coalesce  canvas_bgnd_coal.gif
gif_anim_montage canvas_bgnd_coal.gif canvas_bgnd_coal_frames.gif

convert canvas_bgnd_coal.gif  -layers CompareAny   compare_any.gif
gif_anim_montage compare_any.gif compare_any_frames.gif
```

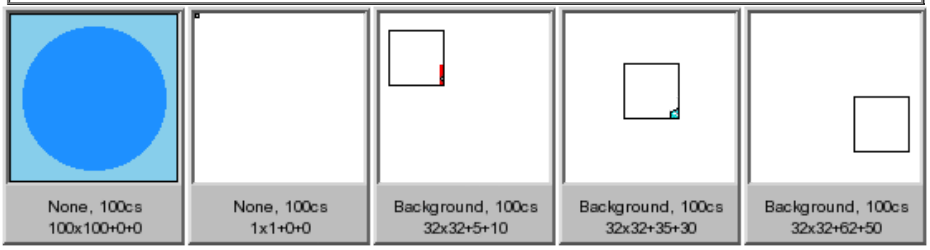


2 番目以降のイメージを見ることができるように、変更されたすべてのピクセルを含む最小の四角形領域(新しいピクセルカラーのオーバーレイ、または古いピクセルの透明化のクリア)です。

Compare_Clear

"" メソッド '' は、1 つのフレームから次のフレームにクリアする必要があるすべてのピクセルを含む最小の矩形領域を表示します。-layersCompareClear

```
convert canvas_bgnd_coal.gif -quiet -layers CompareClear compare_clear.gif
gif_anim_montage compare_clear.gif compare_clear_frames.gif
```

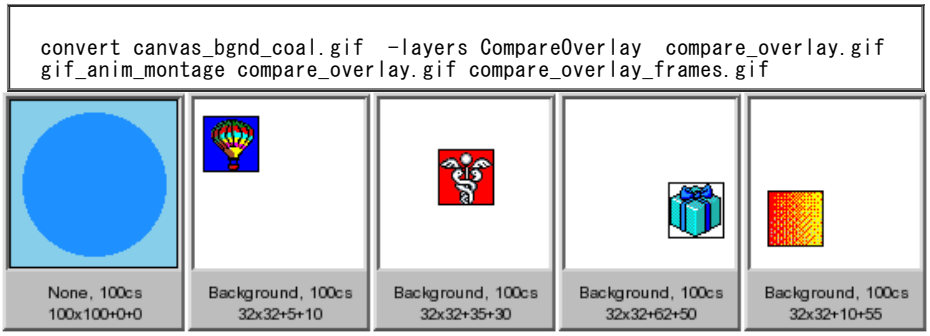


1 番目と 2 番目のフレームの間にピクセルがクリアされなかったため、特別な[見逃したイメージ]が生成されていることに注意してください。"" 設定は、このイメージに関する警告を出さないことを IM に指示するために使用されました。 それ以降のフレームがすべて「見逃した」画像になった場合、GIF アニメーションはピクセルをクリアすることはなく、アニメーションをオーバーレイ アニメーションとして分類できます。-quiet

Compare_Overlay

最後の "" 比較メソッド '' は、前のフレーム以降にオーバーレイされた (色で追加または変更されたが、クリアされていない) ピクセルの領域を返します。-layersCompareOverlay

```
convert canvas_bgnd_coal.gif  -layers CompareOverlay  compare_overlay.gif
gif_anim_montage compare_overlay.gif compare_overlay_frames.gif
```



これは、特殊な IM 固有の '変更マスク" アルファ合成メソッドに似ています。ただし、これは、変更された四角形の領域ではなく、イメージを変更するピクセルだけを返します。「透明度の最適化」も参照してください。

"-layers"比較メソッドも、"-deconstruct"演算子も、使用するイメージ *GIF* 破棄メソッドを参照または変更しません。結果は単なる画像のリストであり、アニメーション自体として使用されるものではありません。

演算子は、合体したイメージ シーケンスを使用するように設計されていますが、エラーを生成することなく、非合体シーケンスのイメージ レイヤーを受け入れます。 この場合、各フレームは、フレームが比較される前に、"アルファコンポジション法を使用して、前のオーバーレイされたフレームにオーバーレイされます。このアルファ合成方法により、レイヤー内の透過表示もターゲットイメージに追加されます。これを行わなければ、上記は、合体した画像シーケンスで透明度にクリアされるピクセルを見つけることはできないでしょう。 これは、"" 演算子が *GIF* アニメーションを表示するために必要な *'dispose/overlay'* サイクルを処理するために使用する、通常の " コンポジションメソッドとは異なっています。

Copy

Over-coalesce

アニメーションの種類

見つけたほとんどの GIF アニメーションは、いくつかの基本的な種類のアニメーションに分類されます。これらのタイプを把握することで、あるフレームから別のフレームにアニメーションがどのように表示されているかを理解でき、アニメーションの処理方法や変更方法にショートカットを使用できます。

合体アニメーション

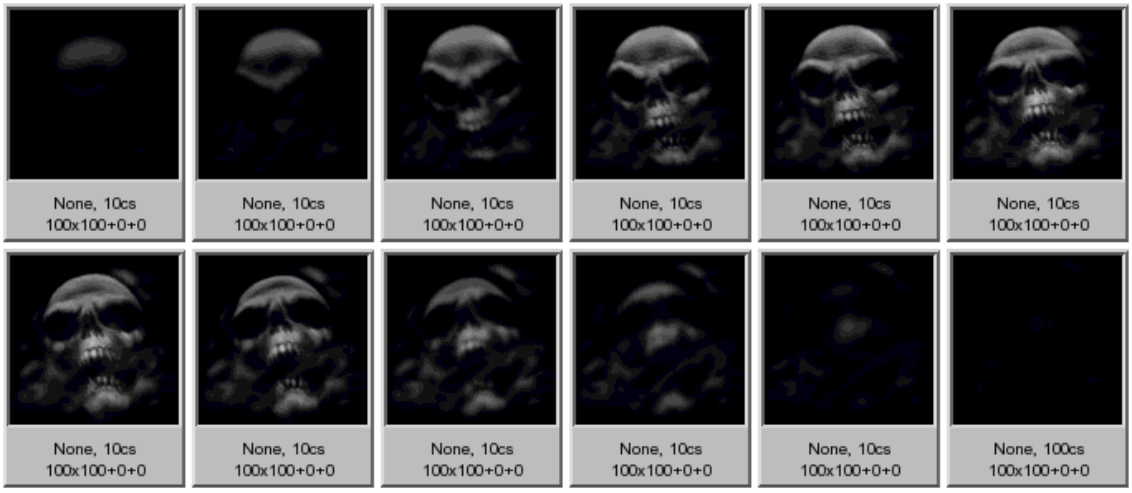
'**合体アニメーション**' は、基本的には、各 '破棄/オーバーレイ' サイクルの後にユーザーに表示されるアニメーションの外観を示すイメージ シーケンスです。画像は、アニメーションの実際の「フィルムストリップ」を見ている場合に、基本的に見るのと同じものです。これは、アニメーションの単純化された完全に最適化されていない形式です。 この命名規則は、GIF 廃棄アニメーションを最適化されていない「合体アニメーション」に変換するために使用される IM "" 演算子の名前から取得されます。 ほとんどのビデオフォーマット(MPEG、AVIなど)は、実際には、その性質上、実際には「合体アニメーション」です。ただし、これらのピクセルは透明なピクセルを持たない傾向があり、一般的にアニメーションのフレーム間の時間遅延は一定です。ただし、合体化された GIF アニメーションは、透明なピクセルを持ち、即時の 0 の遅延から非常に高速、または非常に非常に低速に遅延する時間が大きく変化します。 合体アニメーションの GIF 破棄設定には意味がありませんが、"" 演算子は適切に破棄を設定し、結果のイメージ シーケンスが有効な GIF アニメーションとして動作するようにします。 常に 1 つのフレームから次のフレームにすべてのピクセルを置き換えるビデオ形式は、一般的に '' または '' GIF の破棄設定を使用できます。 ここでは、アニメーションの性質に合体したアニメーションの例に加えて、アニメーションの個々のフレームの""」表示を示します。

-coalesce

-coalesce

NoneUndefined

gif_anim_montage



透明度を含まない、または使用しないアニメーション、キャンバス全体をアニメーション化するアニメーションのほとんどは、通常、結合されたアニメーションとして保存および配布されます。

オーバーレイ アニメーション

オーバーレイ**アニメーション**は、アニメーションの各フレームが現在表示されているアニメーションに新しいピクセルのみをオーバーレイするアニメーションです。つまり、アニメーションのポイントで、ピクセルを透明にする必要はありません。個々のフレームには、背景として、または最適化の一環として、透明度を含めることができますが、透明に戻ってピクセルをクリアすることはありません。 透明度をまったく使用しない場合は、アニメーションを単純なオーバーレイアニメーションに変換できることが保証されます。 これは、フレーム最適化アニメーションの最も単純なタイプであり、クライアントによる特別な処理を必要としないものです。ユーザーに表示される各フレームは、前のすべてのフレームの「フラット化」画像として単純に表示できます。 '' GIF 破棄メソッドのみを使用するアニメーションは、"オーバーレイ アニメーション" です。最後の例は、完全に結合されたアニメーションだけでなく、'オーバーレイ アニメーション' でもありますが、完全に結合されたアニメーションがすべて 「オーバーレイ アニメーション」というわけではありません。 アニメーションがオーバーレイ アニメーションになるかどうかをテストするには、結合されたアニメーションで '' レイヤ メソッドを使用し、2 番目以降のイメージがすべて 「見逃したイメージ」であるかどうかを確認します。これは、あるフレームから次のフレームに対してピクセルをクリアまたは消去する必要はありません。 実際、アニメーションが変更なしでオーバーレイ アニメーションになる場合、IM "" オペレータはすべてのフレームに対して '' 破棄メソッドのみを使用します。この場合、"" は少なくとも一部のフレームに対して '' 破棄を使用しているでしょう。これにより、「オーバーレイのみ」機能の別のテストが行われます。 オーバーレイ アニメーションでは透過表示を使用できますが、透明な背景には可動部分はありません。たとえば、手書き文字 'K' のアニメーションはオーバーレイ アニメーションで、各パーツは透明な背景上の既存のパーツを追加または変更するだけです。結果のイメージに新しい透明度が追加されることはありません(最初のフレームの一部として除く)。

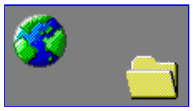
None

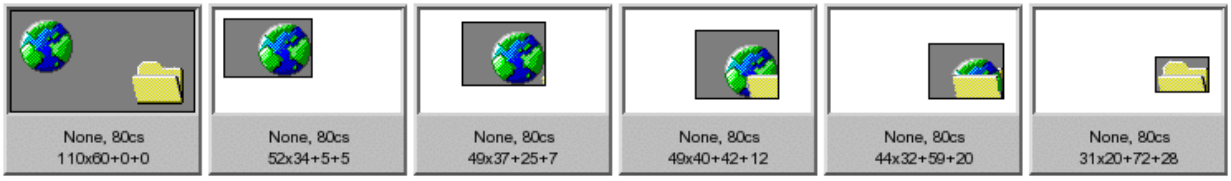
CompareClear

coalesceNonecoalesceBackground

つまり、移動するオブジェクトがオーバーレイアニメーションで処理できないということではなく、透明でない背景が必要なだけなので、透明な部分の古い位置を透明にすることなく「消去」

することもできます。例えば、この「フォルダに世界をダウンロードする」アニメーションのフレームを見てください。



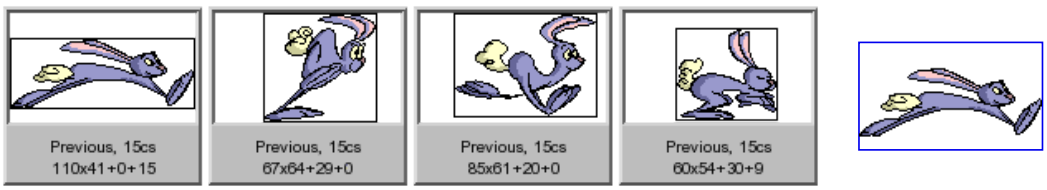


もちろん、元の背景をオーバーレイして古い部分を「消去」する必要があるため、オーバーレイされたサブイメージは一般的に大きく、したがって一般的にGIFアニメーションファイルのサイズが大きくなります。 残念ながら、IMの最適化フレームオペレータを使用して、小さなGIFファイルサイズを見つけようとして、「合体したオーバーレイアニメーション」を「オーバーレイアニメーション」ではないものに変えることができます。 ただし、[フレームを最適化]を使用する代わりにアニメーションでDeconstructを使用すると、アニメーションが単純な 「オーバーレイアニメーション」のままで、アニメーションが実際に 「オーバーレイ アニメーション」である場合にのみ、アニメーションを使用できます。アニメーションが「オーバーレイ アニメーション」でない場合、分解操作が正しくない可能性があります(「上記の分解」を参照)。 人間のスキルを使用すると、たとえば[フレームの更新を分割]を使用し、アニメーションの「オーバーレイのみ」要件を破棄せずに何らかの形式の圧縮最適化を適用するなど、オーバーレイ アニメーションをより適切に最適化できます。 通常、「オーバーレイアニメーション」は透明度をまったく表示しません(最適化の一部として使用できますが、表示されません)。透明度が表示されない場合、アニメーションは「オーバーレイ アニメーション」であることが保証されます。 なぜ「オーバーレイアニメーション」が重要なのですか?なぜなら、このタイプのアニメーションに限定されたソフトウェアがそこにあります。透明度を処理せずにオーバーレイだけが実行される、または GIF の破棄方法を処理するために前のフレームを保存するので、処理するのも非常に簡単です。このようなソフトウェアはまれですが、存在します。

クリアされたフレームアニメーション

アニメーションで '前へ' または '背景' GIF の破棄のみを使用する場合は、非常に特殊な種類のアニメーションを取得します。 'Background'の処理だけを使用する場合は、アニメーションのすべてのフレームが表示され、次のフレームが表示される前にクリアされます。また'' のみを使用すると、アニメーションは、次のフレームが表示される前に、常に最初にクリアされたキャンバスに戻され、同じ効果が得られます。これら 2 つの廃棄設定を組み合わせて使用する場合も同様です。 つまり、これらの破棄方法のみを使用するアニメーションには、表示される内容の完全なコピーであるフレームが含まれます。つまり、フレームには、その時点でユーザーに表示されるすべてのものが含まれます。 それは、アニメーションが'完全に合体したアニメーション'であるというわけではありません。サブフレームはアニメーションの仮想キャンバス領域よりはるかに小さいかもしれませんが、そのフレームの外側はすべて透明(または背景)とみなされ、重要なものは含みません。 たとえば、この実行中のバニーアニメーションを見てみましょう.

Previous



各サブフレームは、表示される完全なイメージであることに注意してください。これ以上、劣らず。また、実際にはアニメーションの仮想キャンバス全体を使用する必要があるフレームがない点にも注意してください。最後に、すべてのフレーム廃棄がどのように '' に設定されているかに注意してください。ただし、すべての廃棄設定は、最終的な結果を変更せずに'破棄または 2 つの組み合わせに設定されている可能性があります。より良い名前を求めるには、そのようなアニ

メーションを「**クリアフレームアニメーション**」と呼びますが、私は「前またはバックグラウンドディスポーダアニメーション」と呼ばれることも見てきました。 アニメーションがこのタイプでないのは、アニメーション内の少なくとも 1 つの非空白フレームが '' または '' (同じもの) の破棄メソッドを使用した場合だけです。 このアニメーションは、オーバーレイ アニメーションとは異なり、アニメーション シーケンス内の任意の量の透明化を処理できるため、非常に特別です。しかし、それはまた、本当に迅速に任意の静的な背景画像にオーバーレイすることができます。 これを行うには、「クリアフレームアニメーション」の定義を少し締める必要があります。具体的には、すべての処分が '' に設定されていることを確認する必要があります (これは既に例の場合です)。その場合は、画像を事前にペンプするだけで(遅延ゼロ)、背景を下敷きにできます。PreviousBackground

NoneUndefined

Previous

例えば、いくつかの草の上に私たちのバニーを置くことができます.

```
convert bunny_grass.gif bunny_anim.gif -loop 0  bunny_on_grass.gif
```

ご覧のとおり、多くのアプリケーションでは、これらのタイプの GIF アニメーションを使用して、シンボルやその他のインジケーター (ファイル ロック、スマイリー、星など) を大きなオブジェクトに追加します。 このような GIF アニメーションのアニメーション化も簡単で、アプリケーションは領域を簡単な一定の背景イメージにクリアし、シーケンス内の次のフレームをオーバーレイできます。静的な不変の背景表示以外の、処分を計算したり、「前の」表示を追跡したりする必要はありません。 これは、''破棄がクリアフレームアニメーションの好ましい処分である理由でもあります。 GIFアニメーションの特殊なサブセットにすぎないオーバーレイ アニメーションとは異なり、すべてのアニメーションはクリア フレーム アニメーションとして保存できます。アニメーションを合体し、必要に応じて周囲の透明エッジをトリミングしてフレームを最適化し、廃棄をリセットします。

Previous

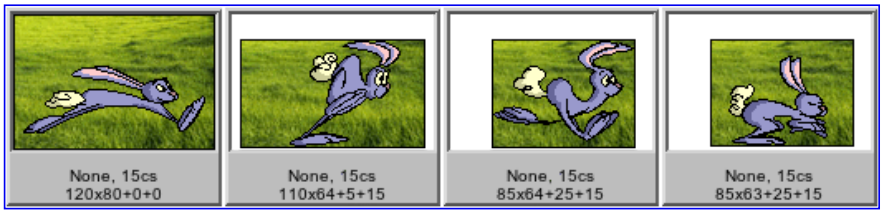
```
convert any_animation.gif -coalesce -trim ¥
        -set dispose previous    cleared_frame_animation.gif
```

アニメーションをその背景に再配置することもできます。

```
convert bunny_grass.gif ¥( bunny_anim.gif -repage 0x0+5+15¥! ¥) ¥
        -loop 0  bunny_on_grass2.gif
```

このようなクリアフレームアニメーションは、通常、透明な背景上で、小さな、絶えず変化または移動するオブジェクトで構成されています。これらは、Web ページで直接使用したり、アニメーションシンボルとして使用したり、他のアニメーションとマージして、より複雑なアニメーションを生成したりできます。 夏期には、このタイプのアニメーションは、より大きな複雑なアニメーションを作成する場合に使用するために、アニメーション化されたパーツのライブラリで使用するのに適したスタイルです。
しかし、GIF アニメーションにこのような背景を追加する場合は問題があります。前の例を見ると、動きの速いアニメーションの一時停止が大きく、邪魔になることがあります。これは、アニメーションがループし、背景画像が再ロードされたときに、一瞬消えたバニーです。(ゼロ遅延フレームの注を参照)この方法を使用するアプリケーションでは問題ではありませんが、表示されたオブジェクトにアニメーションシンボルを追加する場合は、その 「中間」フレームが表示されません。 たとえば、GIF アニメーションに透明でない背景を追加する場合は、一般的に、単純なクリアフレーム アニメーションをオーバーレイ アニメーションに変換することをお勧めします。これは、最初のキャンバスを与えるのではなく、アニメーションのすべてのフレームにその背景を追加することです。 これを行うには、上記のアニメーションを結合し、ゼロ遅延背景フレームを削除するか、元のアニメーションを静的背景に合成します。 例えば。。。

```
convert bunny_grass.gif ¥( bunny_anim.gif -repage 0x0+5+15¥! ¥) ¥
        -coalesce -delete 0 -deconstruct -loop 0  bunny_bgnd.gif
gif_anim_montage  bunny_bgnd.gif  bunny_bgnd_frames.gif
```



これですべてのフレームが均等に表示されるようになったので、背景がリセットされると一時停止は見えなくなります。ただし、アニメーションは、アニメーション オブジェクトの動きのために背景が再描画されたオーバーレイ アニメーションになり、ファイル サイズが少し大きくなる可能性があります。 上記の結果の最適化の継続については、透明度の最適化を参照してください。

> *IM*フォーラムメンバー*del supremo*ピートは、*MagickWand*等しいスクリプト、クリアフレームを背景の例に投稿しました。 この例については、「*MagickWand* でクリアフレーム *GIF* アニメーションを作成する」で詳しく説明します。

混合処分アニメーション- マルチバックグラウンドアニメーション

1 つの GIF アニメーションでさまざまな廃棄方法を混在させる妨げとなるものはありません。クリアフレームアニメーションに背景を追加する場合は、正確にそうします。 混合処理方法を使用すると、私たち人間にとってそれほど単純ではありませんが、そうすることで、非常に複雑なアニメーションディスプレイを生成することができます。一般に、これらは特定のアニメーションの自動フレーム最適化の一部として作成されます。 特定の目的のために以前のアニメーションの種類として直接使用することはできません。.実際には、いくつかのGIF処理プログラムは、単に「混合処分アニメーション」を適切に処理していない場合は驚かないでください。これには、使用可能な GIF オプティマイザーの一部が含まれます。 「混合廃棄アニメーション」の典型的な例は、アニメーションの背景に半永久的だが一時的な静的な変化を起こす小さな移動オブジェクトです。レバーに当たるボールは一例でしょう。 同様に、大きな透明ディスプレイ上で 2 つの非常に小さな移動オブジェクトを含むアニメーションは、各オブジェクトが別々のアニメーション フレームを使用して移動されるように、廃棄手法を組み合わせて最適化することしかできません。 このようなアニメーションの*IM*の例を投稿したい場合は、あなたの名前を取得することができます, そして、ホームページのリンクここに!

```
Simple Example wanted

FUTURE,
A more complex animation for study.
 * Start with a semi-transparent canvas
 * run a little 'previous' disposal
 * leave one frame as a new 'canvas'
 * more previous animations
 * erase that 'addition' using a 'background' disposal set as new canvas
 * continue back to start point.
This animation should thoroughly test out not only IM disposal methods
but also various browser disposal methods.
```

ループの終わり- アニメーションが停止したとき

アニメーションのアニメーションを行っている間、クライアントマシンは継続的に動作する必要がありますので、アニメーションを永遠にループさせないことをお勧めします。そのため、アニメーションが実際にループする回数を考えるとよいでしょう。 基本的には。。

アニメーションにループ制限を追加する(実用的な場合)

これは、Web ページの上部にロゴとして使用される大きなアニメーションの場合に特に当てはまりません。アニメーションの全体的な実行時間に応じて、大きなロゴでは通常 10 ~ 30 ループで十分です。そして、ユーザーに親切にするように、このようなすべてのアニメーションにループ制限を追加する必要があります。 "" 保存設定が 0 の場合は、永遠にループすることを意味します。これは通常、小さなアニメーションで使用されるものであり、問題ありません。ただし、一部のブラウザーでは、アニメーションが内部ループの制限 (多くの場合 256 ループ) に達したときに停止を強制します。 IMの例では、アニメーションがページ上のどこにでも表示されるため、一般的に'0'""保存設定、「永遠にループ」を意味します。つまり、ユーザーがページを読み込んだ時点から、特定の GIF アニメーションを最終的に見て読み取るまでの時間が生じる可能性があります。私が使用する 'ループ数が少ない場合、アニメーションはもはや、その効果を失って、それが実証すべきことを行っていないでしょう。 最上位のフラッシュ ページの大きなアニメーションでは、通常の GIF アニメーションの既定値である '1' を使用できます。つまり、アニメーションを 1 回だけ実行してから停止します。 これは私たちにすべての重要な質問をもたらします.**どこで停止しますか?** 古い "" ブラウザのような一部のブラウザでは、アニメーションの最初のフレームが再表示されます。しかし、ほとんどの現代のブラウザは、フレームが無駄な破棄設定を無視して、最後のフレームで停止するだけです。 しかし、特定のアプリケーションが行うことは、真の標準がないため、アプリケーション次第です。実際には'loop'のメタデータの使用でさえ、それ自体は非標準であり、古い"」ブラウザが実装したもの、そしてそれ以降のすべてのブラウザがコピーしたものだけです。 このため、アニメーションを停止させたいフレームがある場合は、会社名またはロゴが付いたフレームを表示し、そのフレームをアニメーションの最初と最後のフレームの両方にすることをお勧めします。 アニメーションの全体的なループ時間の長さに影響しないようにします。 だから要約するだけです.

-loop

-loop

-loop

Netscape

Netscape

ループが制限されている場合は、最初と最後のフレームの両方として「停止」フレームを追加します。

ゼロ遅延中間フレーム

クリアフレームアニメーションに関しては、すでに「ゼロ遅延」のフレームの使用を見てきました。私はまた、前とバックグラウンドの処分を説明するためにそれらを使用しました. これらの特別なフレームは、実際にはもっと一般的であり、人々はおそらく考えるでしょう。これらは、アニメーションに背景の 「canvas」を追加するメソッドを表すだけでなく、(上記で使用した場合など)。しかし、フレームの倍取りや分割フレームの更新など、より複雑なフレーム最適化テクニックの一部には必須です。 もう 1 つ (非常に古い PNG 以前の形式) の使用法は、256 色の制限を超える静的 GIF イメージを作成できるようにすることでした。各フレームは 256 色を提供し、次のフレームは 256 色の次のセットを提供し、最後に遅延はゼロでループしません。議論のTLULのおかげで、これを指摘するための定量化されていないGIFを作成します。 これらの '遅延ゼロの中間フレーム' は、ユーザーに表示されることを意図していません。GIF 画像では、本来は、それ以外の場合は、GIF イメージを使用せずに取得するよりも適切に最適化されていない特殊効果を作成するために使用されます。 要約すると..

ゼロ遅延フレームは中間フレームであり
、ユーザーに表示されることを意図していません。

ImageMagick は、アニメーションでこのようなフレームを自動 'OptimizePlus' の一部として作成するだけでなく、 'RemoveZero' レイヤーメソッドを使用してそれらを削除する方法も提供し

ます。 彼らはしばしばアニメーションのあなたの処理を複雑にするので、それらに注意してください。 さて、彼らは重要ですので、それの何ですか?多くのアプリケーションは、それらを好きではないか、誤って処理するためです。何らかの理由で、意図的にアニメーションに追加する場合でも、"ゼロ遅延フレーム"は悪いことだと考えています。 ここでは、私が知っているか、「間違ったことをする」と言われたアプリケーションの要約です。

ギンプ 「ゼロ遅延フレーム」を保存しない、彼らは常にゼロの時間遅延を持つ任意のフレームに最小の時間遅延を追加します。:-(

ファイアフォックス そのようなフレームに対して、わずかにゼロ以外の一時停止を与えます。これは、時間の遅延がまったくないアニメーションが、すべてのコンピュータの CPU サイクルを使い切るためであると考えられます。しかし、アニメーションに全体的な非ゼロ表示時間がある場合、"" は依然としてその制限を緩和しません。`firefox`

インターネットエクスプローラ 6 センチ秒の最小時間遅延を持ち、これより小さい遅延を無視します。 Internet Explorer バージョン 8 も、最初のフレームで設定されたアニメーション境界を超えるイメージ フレームが拡張された場合に失敗 (すぐにループを再開します)。これは私が大きなバグとして分類します。

一方、ImageMagick 'RemoveZero' レイヤーメソッドは正しいことを行い、すべての画像が「ゼロの時間遅延」を持っている場合はフレームを削除しません。実際には、このレイヤーメソッドは、時間の遅延がまったくないアニメーションを見ると警告を表示します。 これは私たちを別の経験則に導きます。

「遅延」のない GIF (ループ) アニメーションを保存しない

そうすることは非常に悪い習慣であり、上記の「バギー」アプリケーションのほとんどが、彼らが何をすべきかよりもむしろ、彼らが何をすべきかを行う理由です。あなたがそれらを見たら所有者に文句を言う。 また、アプリケーション開発者に対しては、ゼロ遅延フレームを正しく処理するように苦情を言う。フレームをまったく表示しない場合でも、次に表示するフレームの準備として使用するだけです。彼らは結局ゼロ時間のために画面上にあります!

Created: 24 July 2004 (sub-division of "animation")
Updated: 27 February 2013
Author: Anthony Thyssen, <Anthony.Thyssen@gmail.com>
Examples Generated with: IM v6.6.3-8 2010-08-18
URL: `https://legacy.imagemagick.org/Usage/anim_basics/`