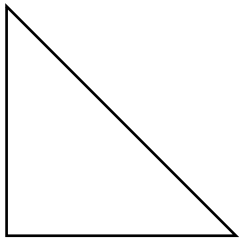
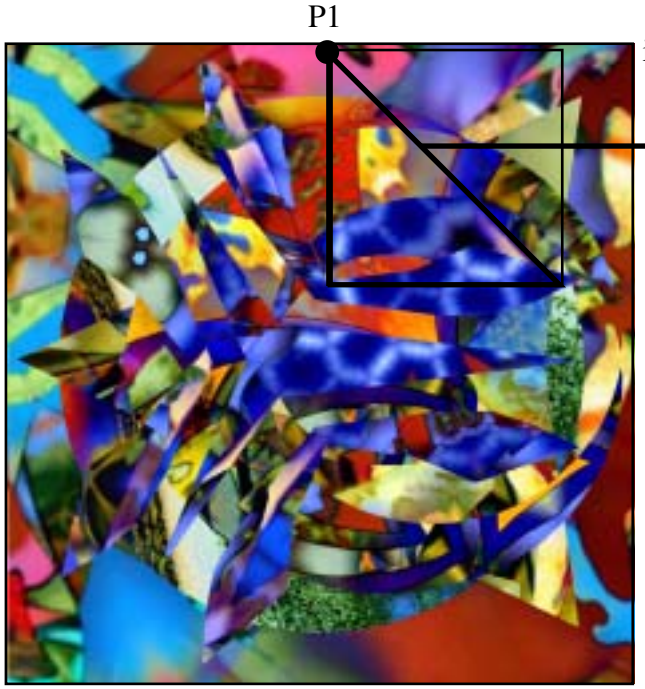
Draft V 0.9

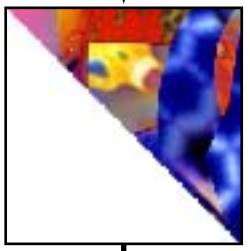
Seamless Plane Groups with ImageMagick/PerlMagick: 1.2) p4m_tile

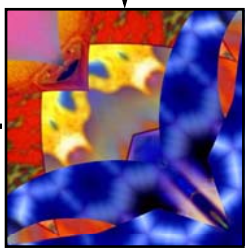
Günter Bachelier
www.aroshu.de AROSHU® Evolutionary Art © Günter Bachelier

p4m basic element:









<u>given in the deterministic case</u>
a) image.jpg with (image_width, image_height)
    given the example image: M04-05-08b-1-026Z.jpg with
    image_width = image_height: 4000 [pixel]
b) format of the basic element: (triangle_width, triangle_width)
    given the example mask: p4m_mask.png with
    triangle_width: 1500 [pixel]
c) Top left point P1 in image.jpg for selection of image_p4m
    given the example: P1 = (2016,14)

<u>given in the stochastic case</u>
a) image.jpg with (image_width, image_height)
b) random variable interval for the selection of triangle_width:
    [triangle_min_Faktor, triangle_max_Faktor] = [0.3, 0.7]

p4m-procedure:
Start
    1) Generate G1
    2) Generate G2
    3) Generate tile_part
    4) Generate tile
End

Only relevant for the stochastic case:
1.1) Calculate scalare triangle_width, x1, y1

```
$triangle_width_min_Faktor = 0.3;
$triangle_width_max_Faktor =0.7;
if ($image_width < $image_height) {$image_min = $image_width} else {$image_min = $image_height};
$triangle_width_min = $triangle_width_min_Faktor * $image_min;
$triangle_width_max = $triangle_width_max_Faktor * $image_min;
$triangle_width = $triangle_width_min + (int(srand($triangle_width_max - $triangle_width_min)) + 1);
$x_allowed = $image_width - $triangle_width;
$y_allowed = $image_hight - $triangle_width;
$x1 = int(srand($x_allowed)) + 1;
$y1 = int(srand($y_allowed)) + 1;
```

Generate mask p4m_mask.png

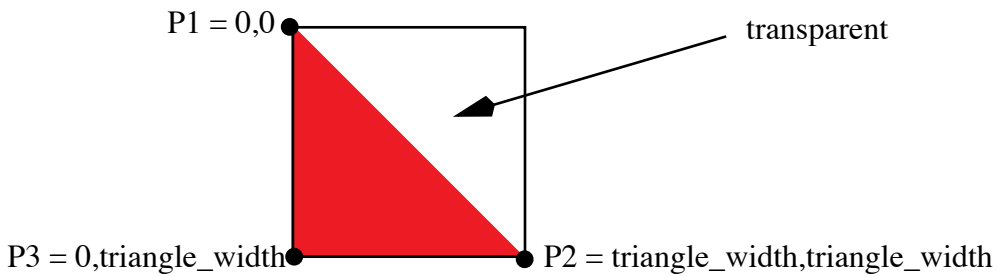
Generate canvas with width = triangle_width and height = triangle_width and draw a triangle P1P2P3 = (1,1 triangle_width,triangle_width 1,triangle_width):

```
convert -size {$triangle_width}x{$triangle_width} xc:none -fill red -draw "polyline 0,0 $triangle_width,$triangle_width 0,$triangle_width" p4m_mask.png
```

For the example holds:

```
convert -size 1500x1500 xc:none -fill red -draw “polyline 0,0 1500,1500 0,1500”  p4m_mask.png
```

http://www.vi-anec.de/Trance-Art/IM-examples/IM-plane_group_p4m/p4m_mask.png



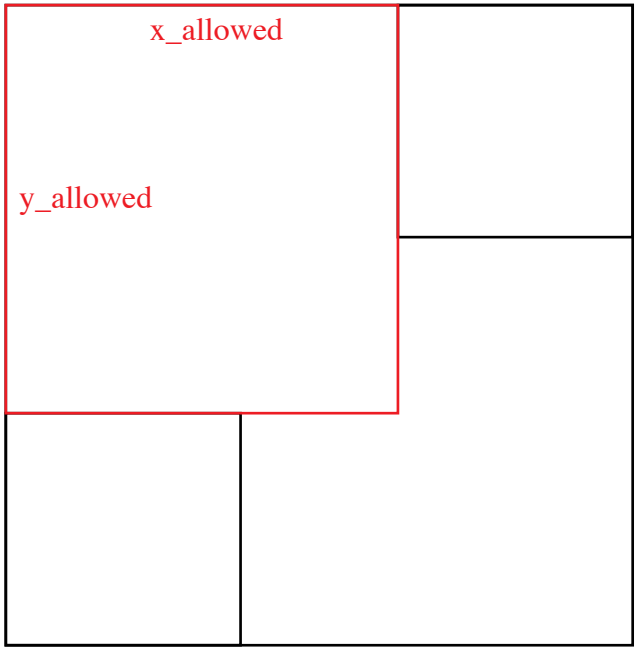
Copy from image.jpg a (random) rectancle with width = triangle_width and height = triangle_width

```
convert image.jpg -crop {$triangle_width}x{$triangle_width}+$x1+$y1 +repage image_p4m.png
```

For the example holds:

```
convert image.jpg -crop 1500x1500+2016+14 +repage image_p4m.png
```

http://www.vi-anec.de/Trance-Art/IM-examples/IM-plane_group_p4m/image.jpg
http://www.vi-anec.de/Trance-Art/IM-examples/IM-plane_group_p4m/image_p4m.png



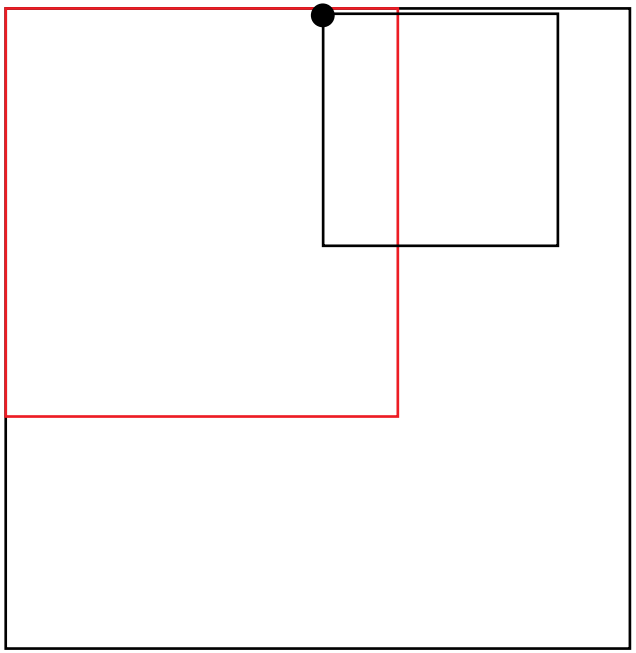
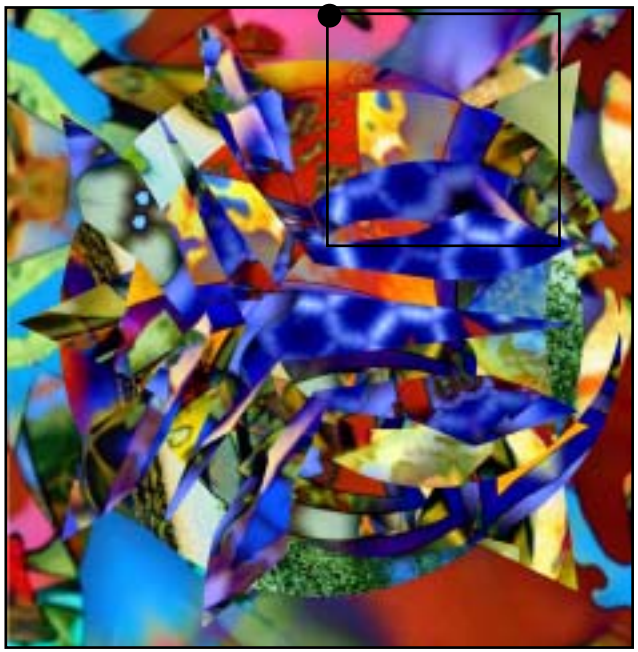


image.jpg

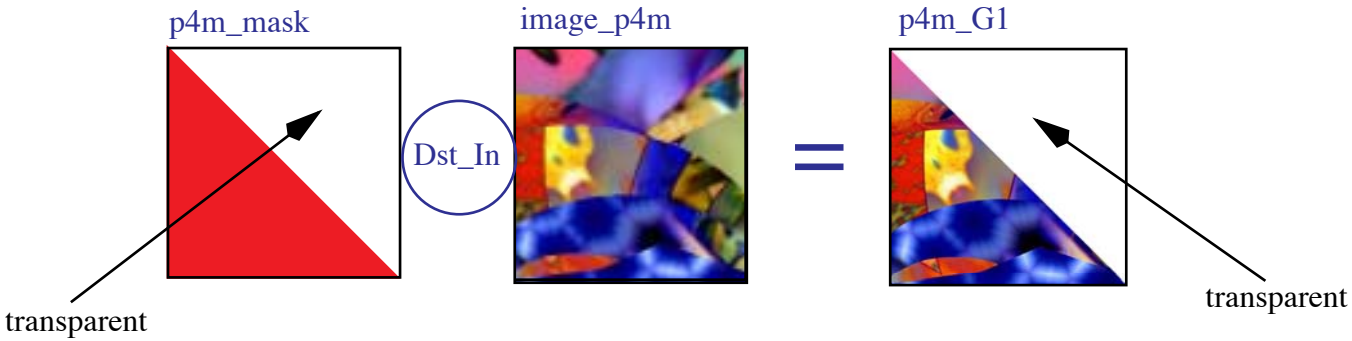


image_p4m.png

1.4) Generate p4m basic element p4m_G1

composite p4m_mask.png image_p4m.png -matte -compose Dst_In p4m_G1.png

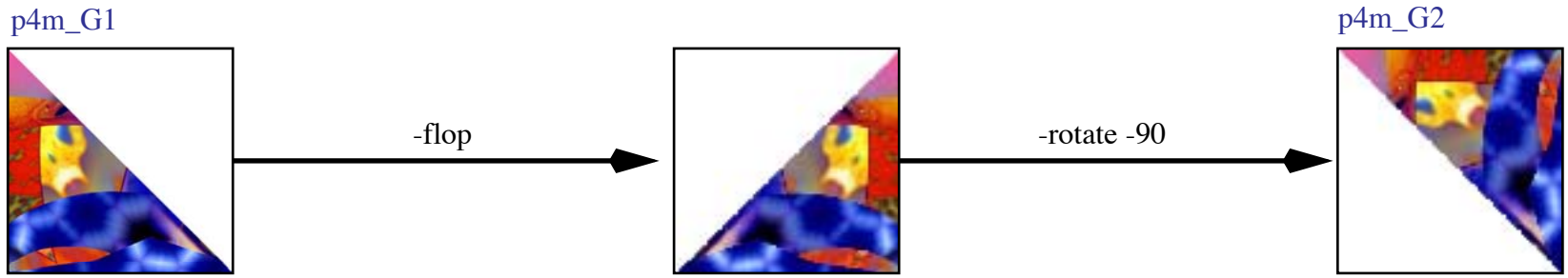
http://www.vi-anec.de/Trance-Art/IM-examples/IM-plane_group_p4m/p4m_G1.png



2) Generate with G1 the basic element G2

convert p4m_G1.png -flop -rotate -90 p4m_G2.png

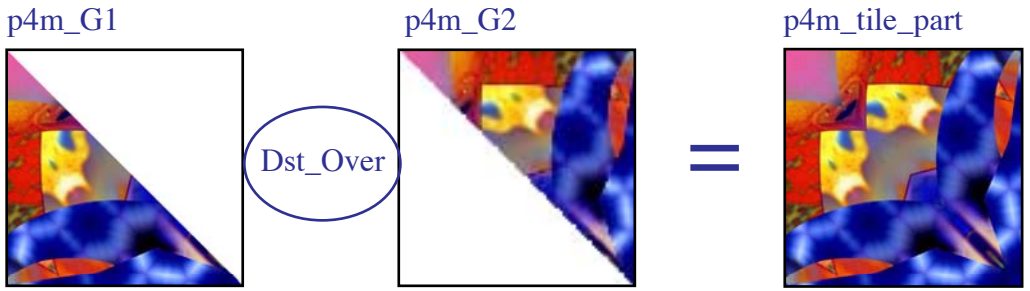
http://www.vi-anec.de/Trance-Art/IM-examples/IM-plane_group_p4m/p4m_G2.png



3) Generate tile_part by composing p4m_G1 and p4m_G2 with Dst_Over:

composite p4m_G1.png p4m_G2.png -compose Dst_Over p4m_tile_part.png

http://www.vi-anec.de/Trance-Art/IM-examples/IM-plane_group_p4m/p4m_tile_part.png



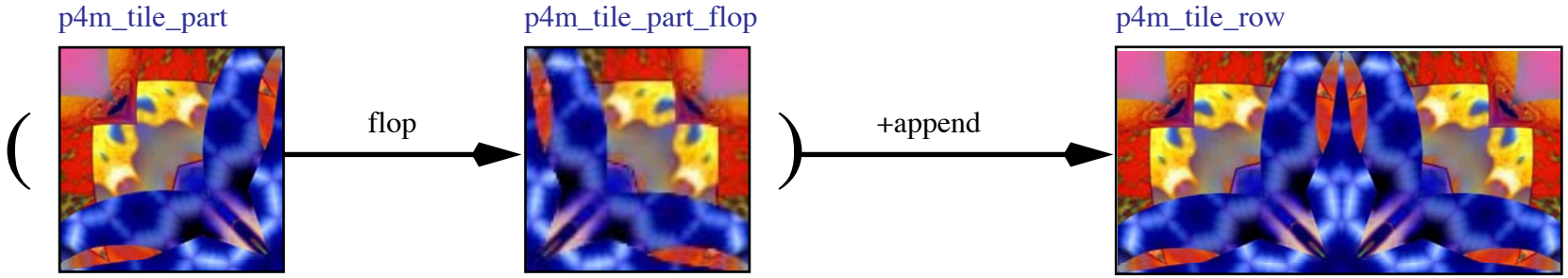
4) Generate p4m_tile

4.1) Generate p4m_tile_part_flop.png: convert p4m_tile_part.png -flop p4m_tile_part_flop.png

4.2) Generate tile_row: convert p4m_tile_part.png p4m_tile_part_flop.png +append p4m_tile_row.png

4.3) Generate tile_row_flip: convert p4m_tile_row.png -flip p4m_tile_row_flip.png

4.4) Generate p4m_tile: convert p4m_tile_row.png p4m_tile_row_flip.png -append p4m_tile.png



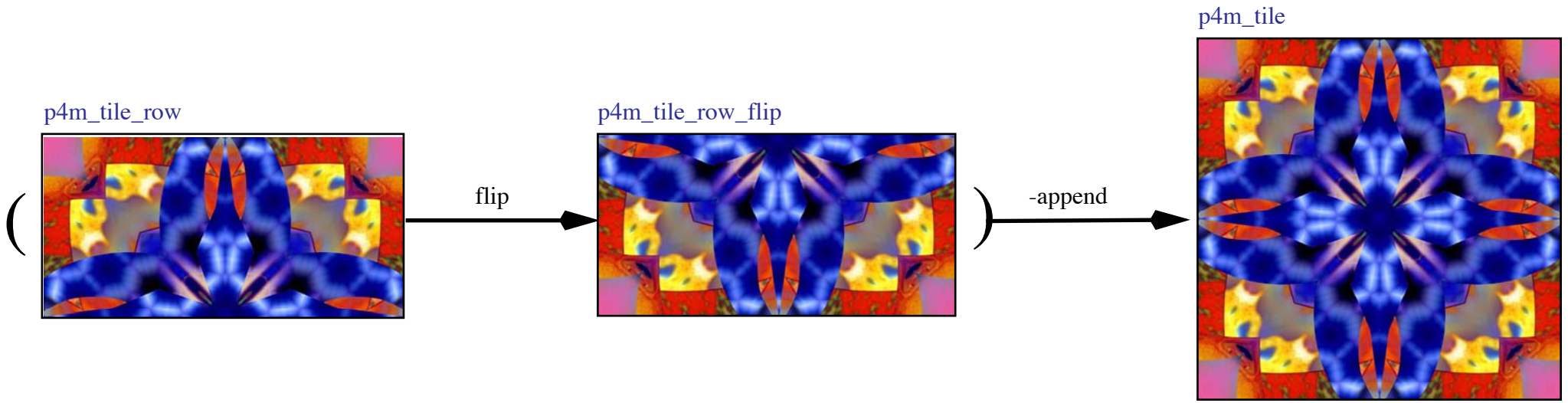
http://www.vi-anec.de/Trance-Art/IM-examples/IM-plane_group_p4m/p4m_tile_part_flop.png

http://www.vi-anec.de/Trance-Art/IM-examples/IM-plane_group_p4m/p4m_tile_row.png

4.3) Generate tile_row_flip: convert p4m_tile_row.png -flip p4m_tile_row_flip.png
4.4) Generate p4m_tile: convert p4m_tile_row.png p4m_tile_row_flip.png -append p4m_tile.png

Summarize tile generation from p4m_tile_part:

convert p4m_tile_part.png (+clone -flop) + append (+clone -flip) -append p4m_tile.png



http://www.vi-anec.de/Trance-Art/IM-examples/IM-plane_group_p4m/p4m_tile_row_flip.png
http://www.vi-anec.de/Trance-Art/IM-examples/IM-plane_group_p4m/p4m_tile.png

IM command sequence for p3m1 in general

1) convert -size {$triangle_width}x{$triangle_width} xc:none -fill red -draw "polyline 0,0 $triangle_width,$triangle_width 0,$triangle_width" p4m_mask.png
2) convert image.jpg -crop {$triangle_width}x{$triangle_width}+$x1+$y1 +repage image_p4m.png
3) composite p4m_mask.png image_p4m.png -matte -compose Dst_In p4m_G1.png
4) convert p4m_G1.png -flop -rotate -90 p4m_G2.png
5) composite p4m_G1.png p4m_G2.png -compose Dst_Over p4m_tile_part.png
6) convert p4m_tile_part.png -flop p4m_tile_part_flop.png
7) convert p4m_tile_part.png p4m_tile_part_flop.png +append p4m_tile_row.png
8) convert p4m_tile_row.png -flip p4m_tile_row_flip.png
9) convert p4m_tile_row.png p4m_tile_row_flip.png -append p4m_tile.png

IM command sequence for p3m1 for the example

1) convert -size 1500x1500 xc:none -fill red -draw “polyline 0,0 1500,1500 0,1500” p4m_mask.png
2) convert image.jpg -crop 1500x1500+2016+14 +repage image_p4m.png
3) composite p4m_mask.png image_p4m.png -matte -compose Dst_In p4m_G1.png
4) convert p4m_G1.png -flop -rotate -90 p4m_G2.png
5) composite p4m_G1.png p4m_G2.png -compose Dst_Over p4m_tile_part.png
6) convert p4m_tile_part.png -flop p4m_tile_part_flop.png
7) convert p4m_tile_part.png p4m_tile_part_flop.png +append p4m_tile_row.png
8) convert p4m_tile_row.png -flip p4m_tile_row_flip.png
9) convert p4m_tile_row.png p4m_tile_row_flip.png -append p4m_tile.png

```
use Image::Magick;


# 0)Input
# 0.1) counting variable, folders, file
$p = '4'; #counting variable
$sourcefolder_path = "C:/ART/IM_PlaneCovering/1_Sourceimages/";
$sourcefile_name = 'image.jpg';
$p4m_resultfolder_path = "C:/ART/IM_PlaneCovering/2_Tiles/2_02_p4m-Tiles/";


# 0.2) definition of lokal parameters
$triangle_width_min_Faktor = 0.4;
$triangle_width_max_Faktor = 0.6;


# 1) definition of PerlMagick objects
$image_p4m = new Image::Magick;
$p4m_mask = new Image::Magick;


# 2) inilisation of random number generator
srand;


#3) open source image
$sourcefile_fullname = $sourcefolder_path . $sourcefile_name;
$image_p4m->Read("$sourcefile_fullname");
```

```
#4) local parameter calculations: $triangle_width = $triangle_height, $x1, $y1
$image_width = $image_p4m->Get('columns');
$image_height = $image_p4m->Get('rows');
if ($image_width < $image_height) {$image_min = $image_width} else {$image_min = $image_height};
$triangle_width_min = $triangle_width_min_Faktor * $image_min;
$triangle_width_max = $triangle_width_max_Faktor * $image_min;
$triangle_width = $triangle_width_min + (int(rand($triangle_width_max - $triangle_width_min)) + 1);
$x_allowed = $image_width - $triangle_width;
$y_allowed = $image_height - $triangle_width;
$x1 = int(rand($x_allowed))+1;
$y1 = int(rand($y_allowed))+1;


# 5)generate p4m_mask
#convert -size {$triangle_width}x{$triangle_width} xc:none -fill red -draw "polyline 0,0 {$triangle_width},{$triangle_width}
0,{$triangle_width}" p4m_mask.png
$p4m_mask->Set(size=>"$triangle_width x $triangle_width");
$p4m_mask->Read('xc:none');
$p4m_mask->Draw(primitive=>'polyline', points => "0,0 $triangle_width,$triangle_width 0,$triangle_width", fill=>'red');


# 6) generate image_p4m by cropping the source image
$image_p4m->Crop(geometry => "$triangle_width x $triangle_width+$x1+$y1");


# 7) generate p4m_G1
#composite p4m_mask.png image_p4m.png -matte -compose Dst_In p4m_G1.png
$p4m_G1 = $p4m_mask->Clone();
$p4m_G1->Composite(image=>$image_p4m, compose=>in);
```

```
# 8) generate p4m_G2
#convert p4m_G1.png -flop -rotate -90 p4m_G2.png
$p4m_G2 = $p4m_G1->Clone();
$p4m_G2->Flop();
$p4m_G2->Rotate(-90);


# 9) generate p4m_tile_part
#composite p4m_G1.png p4m_G2.png -compose Dst_Over p4m_tile_part.png
$p4m_tile_part = $p4m_G2->Clone();
$p4m_tile_part->Composite(image=>$p4m_G1, compose=>over);


# 10) generate p4m_tile_part_flop
#convert p4m_tile_part.png -flop p4m_tile_part_flop.png
$p4m_tile_part_flop = $p4m_tile_part->Clone();
$p4m_tile_part_flop->Flop();


# 11) generate p4m_tile_row
#convert p4m_tile_part.png p4m_tile_part_flop.png +append p4m_tile_row.png
$q = $p4m_tile_part->Clone();
push(@$q, $p4m_tile_part_flop);
$p4m_tile_row = $q->Append(stack=>'false');


# 12) generate p4m_tile_row_flip
#convert p4m_tile_row.png -flip p4m_tile_row_flip.png
$p4m_tile_row_flip = $p4m_tile_row->Clone();
$p4m_tile_row_flip->Flip();
```

```
# 13) generate p4m_tile_row_flip
#convert p4m_tile_row.png p4m_tile_row_flip.png -append p4m_tile.png
@$q = ();
$q = $p4m_tile_row->Clone();
push(@$q, $p4m_tile_row_flip);
$p4m_tile = $q->Append(stack=>'true');


# 14) generate name of result image
chop $sourcefile_name;
chop $sourcefile_name;
chop $sourcefile_name;
chop $sourcefile_name;

if ($p <= 9) {$resultimage_name = "p4m_tile_" . $sourcefile_name . "-0" . $p . ".jpg"}
else {$resultimage_name = "p4m_tile_" . $sourcefile_name . "-"  . $p . ".jpg"};

$resultimagepath_name = $p4m_resultfolder_path . $resultimage_name;


# 15) save result image
$p4m_tile->Write(filename=>"$resultimagepath_name", compression=>'JPEG', quality=>'95');
```