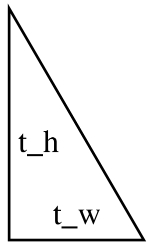
Draft V 0.9

Seamless Plane Groups with ImageMagick/PerlMagick: 1.4) p6m_tile

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

p6m basic element:



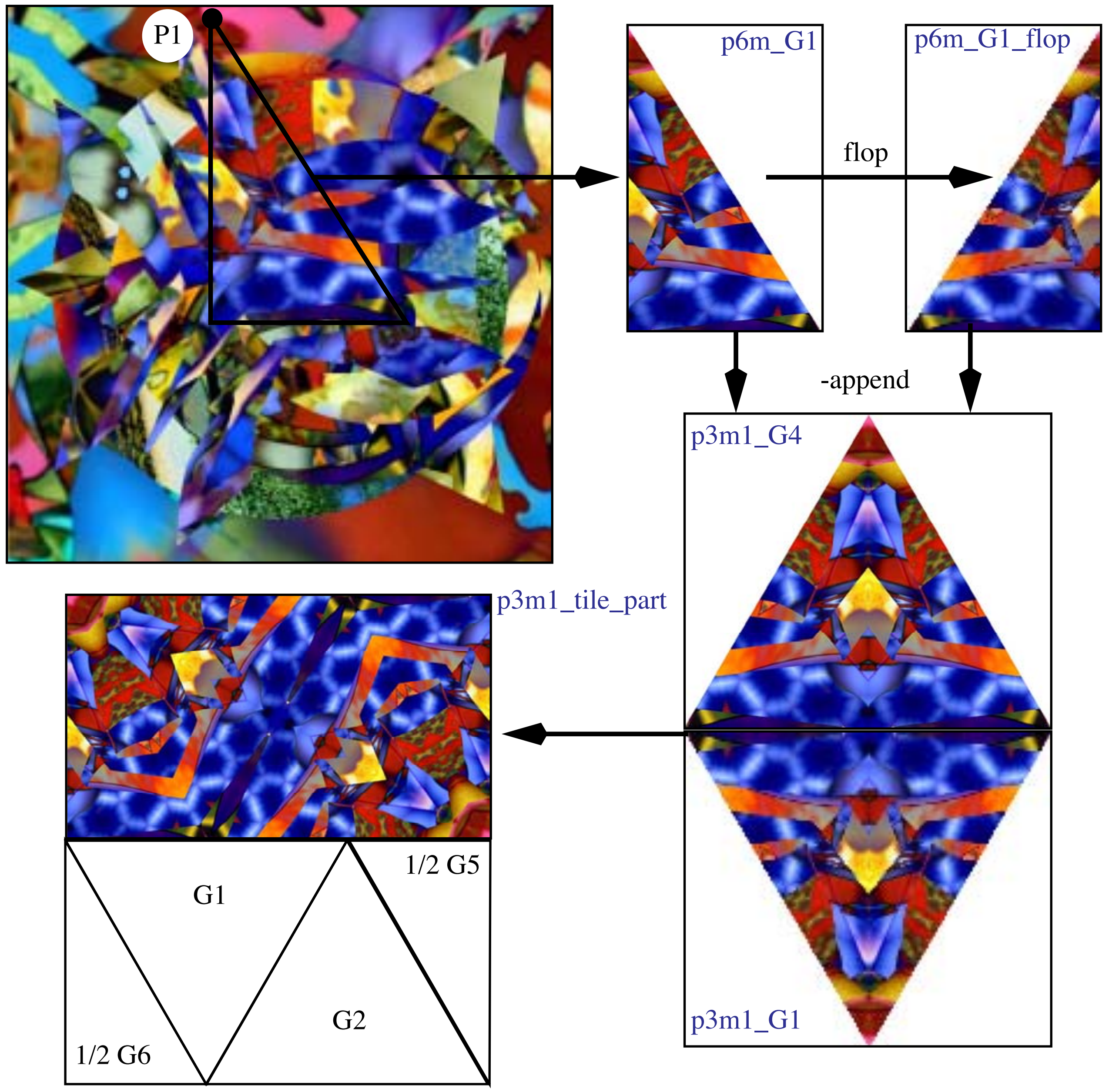
<u>given in the deterministic case</u>

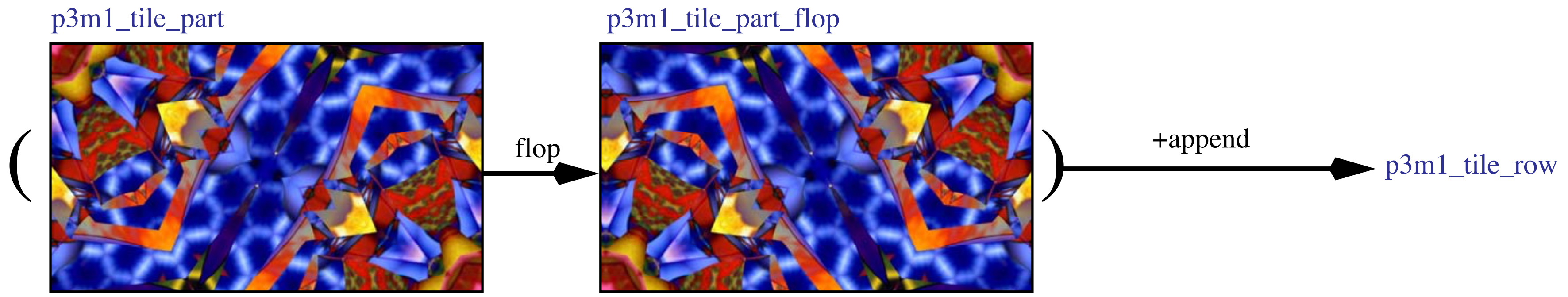
a) image.jpg with (image_width, image_height)
    example: image_width = image_height: 4000 [pixel]
b) format of the basic element: (triangle_width, triangle_height)
    example: (1272, 2204) [pixel]
c) Top left point P1 in image.jpg for selection of image_p6m
    example: P1 = (1544, 36)

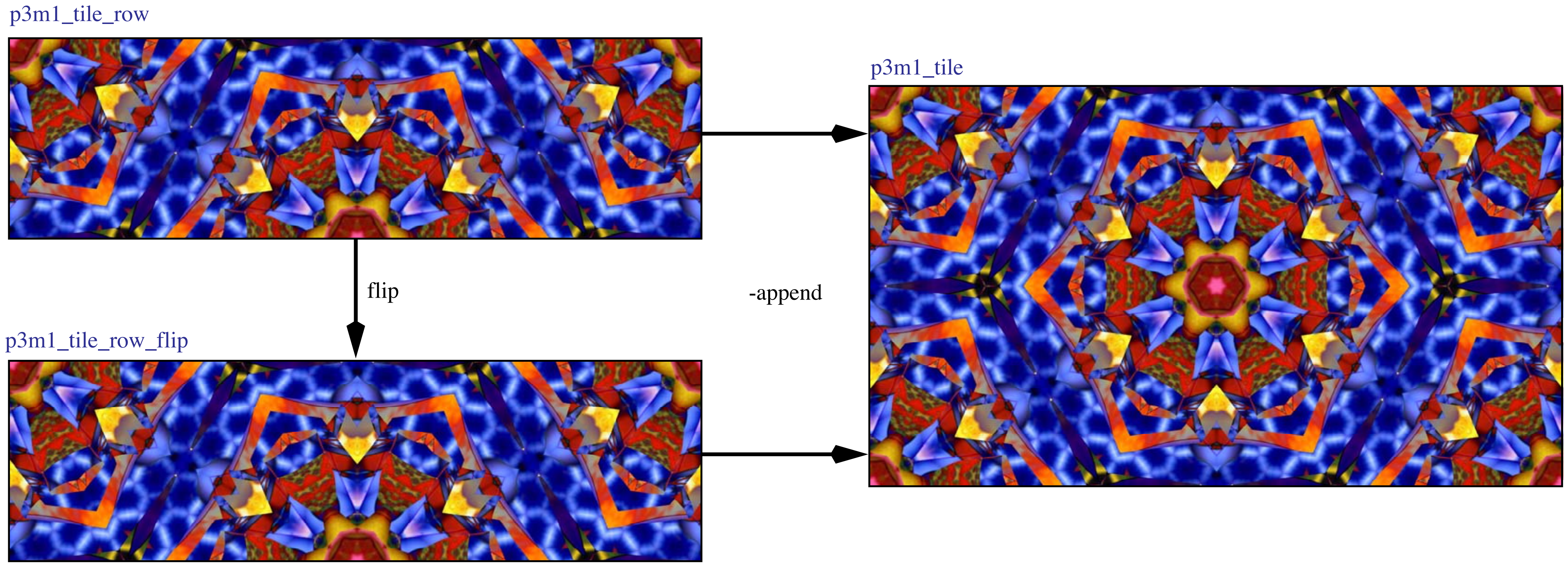
<u>given in the stochastic case</u>

a) image.jpg with (image_width, image_height)
b) random variable interval for the selection of triangle_width:
    [triangle_min_Faktor, triangle_max_Faktor] = [0.4, 0.6]

```
p6m-procedure (reduce p6m to p3m1 after second step)
Start
    1) Generate p6m_G1
    2) Generate p3m1_G4
    3) Generate p3m1_G1, G6, G5, G2
    4) Generate p3m1_tile_part
    5) Generate tile
End
```



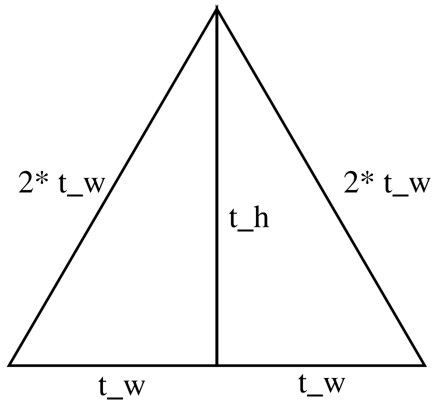

p3m1_tile_part
p3m1_tile_part_flop
(
flop
)
+append
p3m1_tile_row


p3m1_tile_row
p3m1_tile
flip
-append
p3m1_tile_row_flip

Only relevant for the stochastic case

Calculate scalare triangle_width, triangle_height, $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 = $triangle_width_min + (int(rand($triangle_width_max - $triangle_width_min)) + 1);
$triangle_height = int(√3 * $triangle_width) + 1;
$x_allowed = $image_width - $triangle_width;
$y_allowed = $image_hight - $triangle_width;
$x1 = int(rand($x_allowed)) + 1;
$y1 = int(rand($y_allowed)) + 1;
```



$4 * t\_w^2 = t\_w^2 + t\_h^2$

$\Leftrightarrow t\_h^2 = 3 * t\_w^2$

$\Leftrightarrow t\_h = \sqrt{3} * t\_w$

****

Calculate scalare triangle_height, triangle_width, $x1, $y1

```
$triangle_height_min_Faktor = 0.4;
$triangle_height_max_Faktor =0.6;
if ($image_width < $image_height) {$image_min = $image_width} else {$image_min = $image_height};
$triangle_height_min = $triangle_height_min_Faktor * $image_min;
$triangle_height_max = $triangle_height_max_Faktor * $image_min;
$triangle_height = $triangle_height_min + (int(rand($triangle_height_max - $triangle_height_min)) + 1);
$triangle_width = int(1/√3 * $triangle_height) + 1;
$x_allowed = $image_width - $triangle_width;
$y_allowed = $image_hight - $triangle_width;
$x1 = int(rand($x_allowed)) + 1;
$y1 = int(rand($y_allowed)) + 1;
```

$4 * t\_w^2 = t\_w^2 + t\_h^2$

$\Leftrightarrow 3 * t\_w^2 = t\_h^2$

$\Leftrightarrow t\_w = 1/\sqrt{3} * t\_h$

Generate mask p6m_mask.png

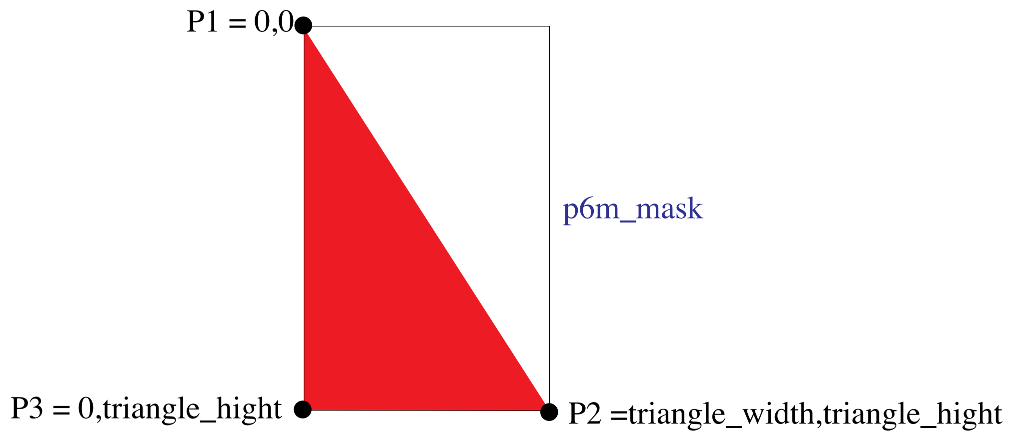
Generate canvas with width = triangle_width and height = triangle_height and
draw a triangle P1P2P3 = (x1,y1 x2,y2 x3,y3) = (0,0 triangle_width,triangle_height 0,triangle_height):

```
convert -size {$triangle_width}x{$triangle_height} xc:none -fill red -draw "polyline 0,0 $triangle_width,$triangle_width 0,$triangle_height" p6m_mask.png
```

For the example holds:

$triangle_height = int(√3 * 1272) + 1 = 2004

```
convert -size 1272x2004 xc:none -fill red -draw “polyline 0,0 1272,2004 0,2004” p6m_mask.png
```



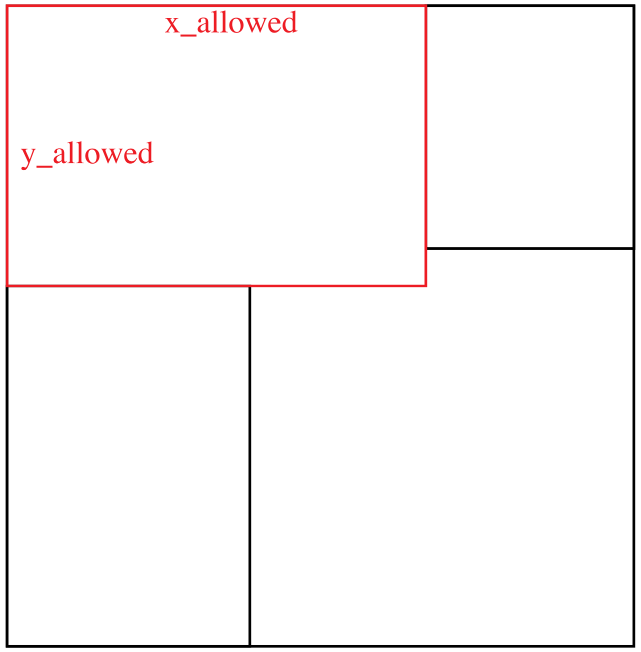
http://www.vi-anec.de/Trance-Art/IM-examples/IM-plane_group_p6m/p6m_mask.png

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_p6m.png
```

For the example holds:

```
convert image.jpg -crop 1272x2204+1544+36 +repage image_p6m.png
```



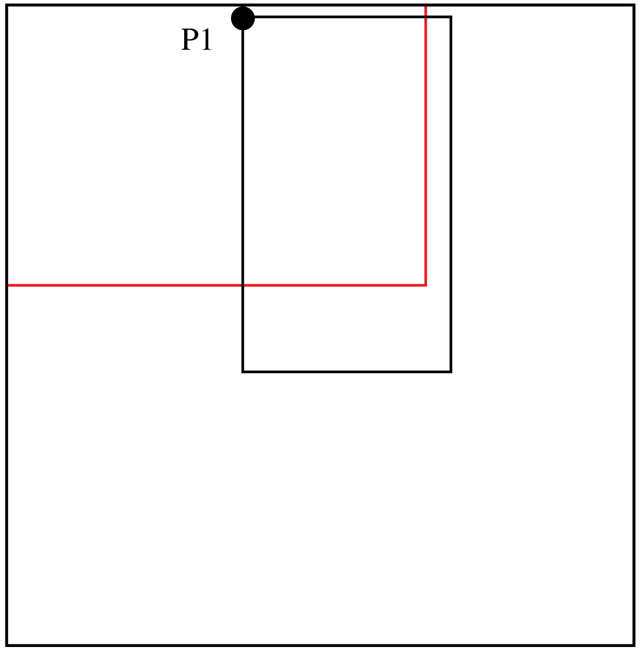
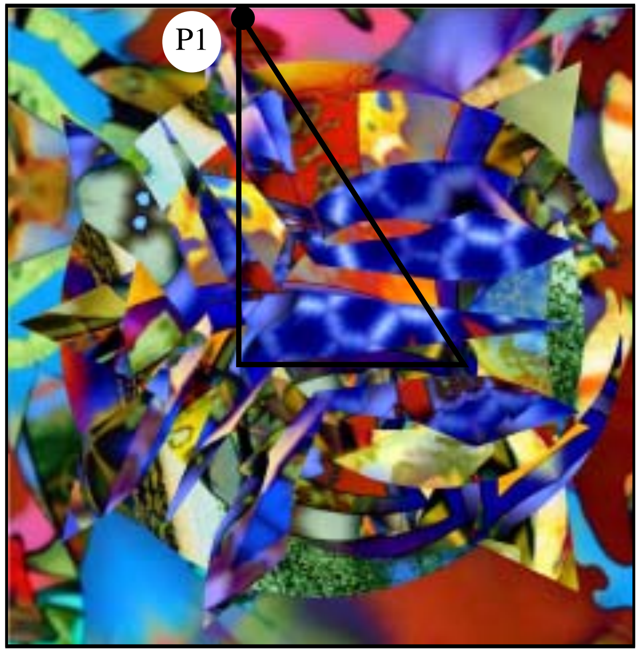
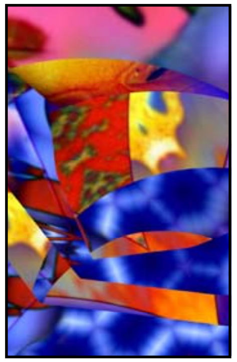


image.jpg

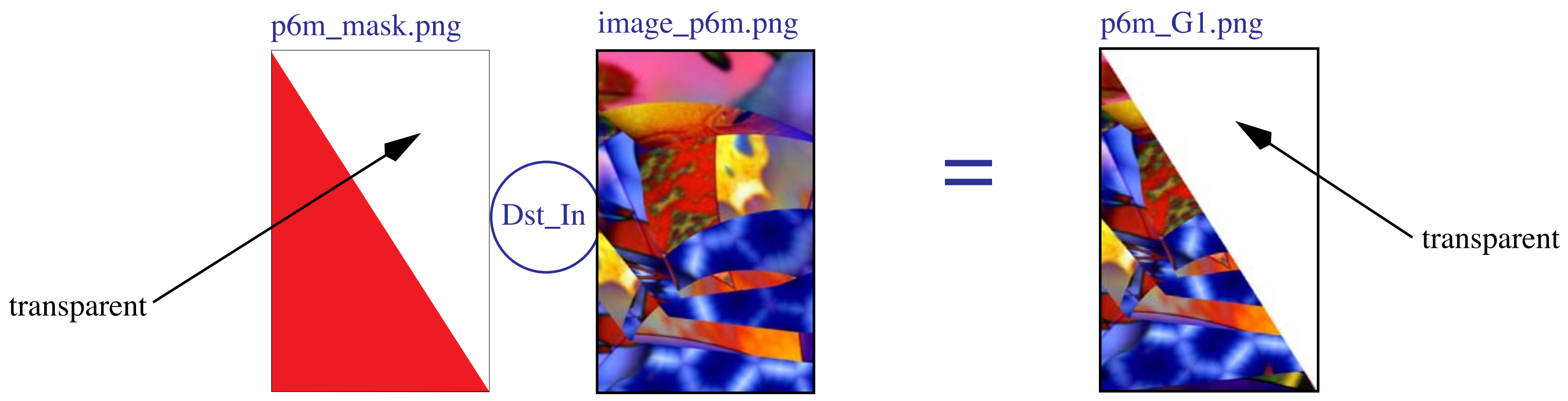


image_p6m.png



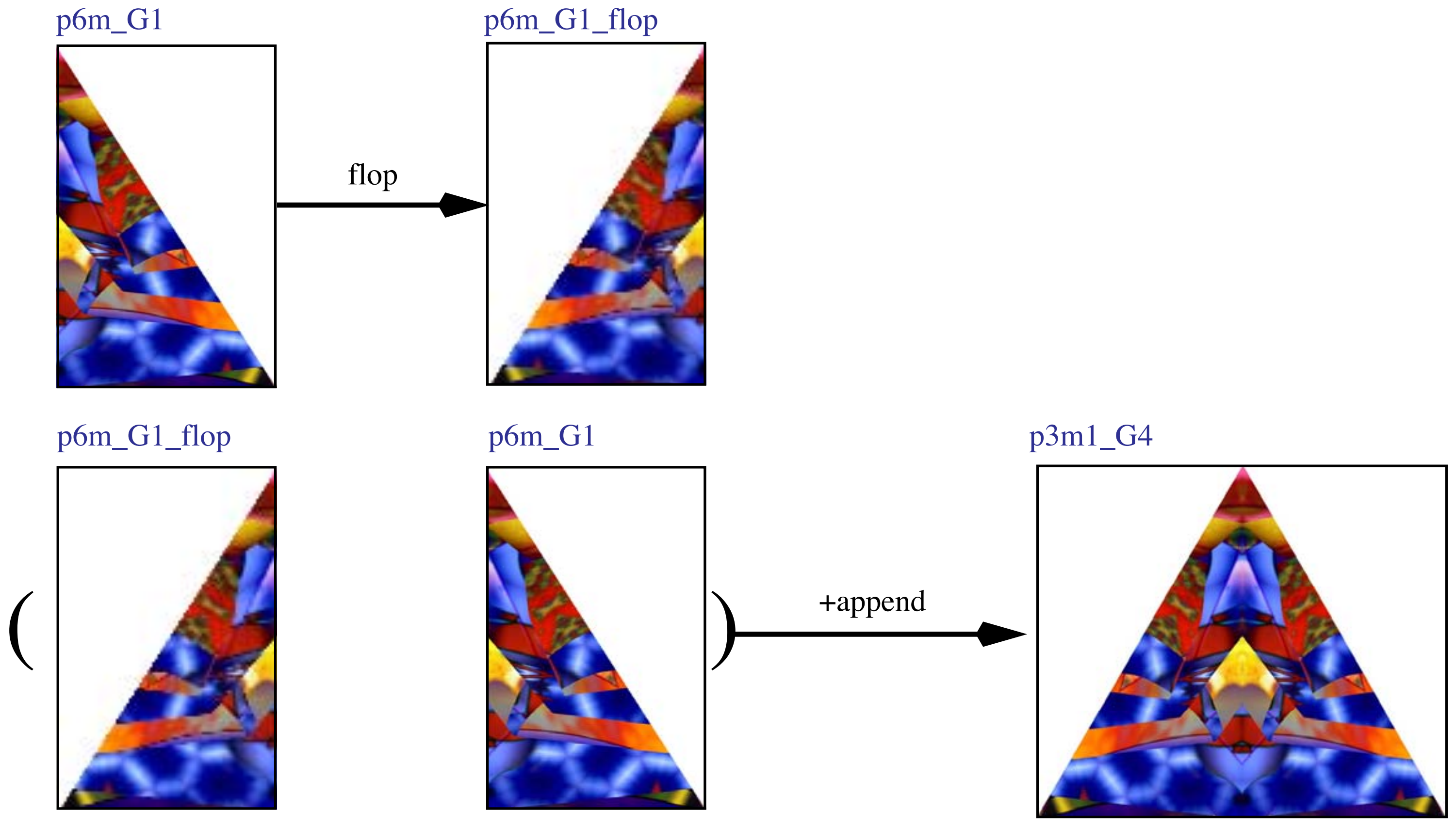
Generate p6m basic element p6m_G1

```
composite p6m_mask.png image_p6m.png -matte -compose Dst_In p6m_G1.png
```



2) Generate p3m1_G4

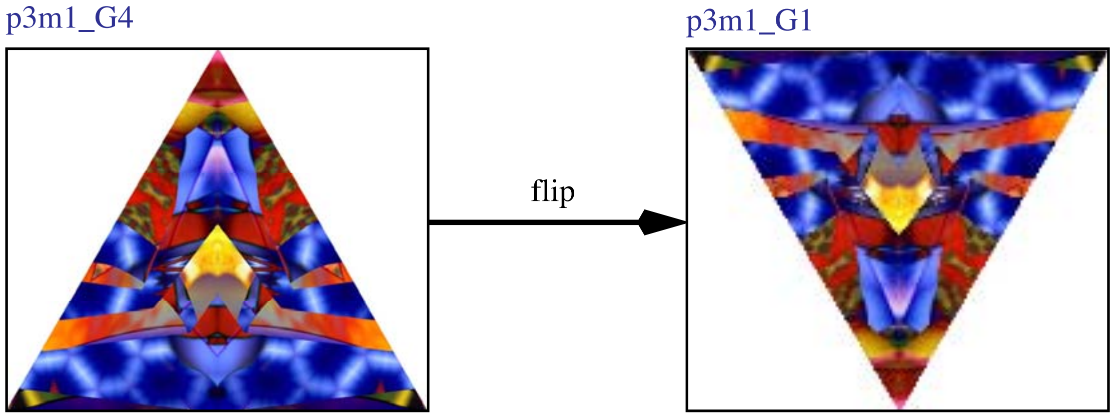
```
convert p6m_G1.png -flop p6m_G1_flop.png
convert p6m_G1_flop.png p6m_G1.png +append p3m1_G4.png
```



3) Generate p3m1_G1, G2, G6, G5

3.1) Generate p3m1_G1

```
convert p3m1_G4.png -flip p3m1_G1.png
```



```
# 3.2) Generation of p3m1_G2
        convert p3m1_G1.png -flop -rotate -60 p3m1_G6b.png
        convert p3m1_G2b.png -gravity NorthEast -crop {$triangle_width}x{$triangle_height}+0+0 -background none +repage p3m1_G2.png
# 3.3) Generation of p3m1_G6
        convert p3m1_G1.png -flop -rotate -60 p3m1_G6b.png
        convert p3m1_G6b.png -crop {$triangle_width}x{$triangle_height}+0+0 -background none +repage p3m1_G6.png
# 3.4) Generation of p3m1_G5
        convert p3m1_G6.png -flip  p3m1_G5.png

# 4) Generate p3m1_tile_part
# 4.1) crop 1/2 G6: p3m1_G6h
        convert p3m1_G6.png -gravity East -crop {int(1/2 * $triangle_width)+1}x{$triangle_height}+0+0 -background none +repage p3m1_G6h.png
# 4.2) insert p3m1_G6h in G1 at the left side: p3m1_tile_part-1
        composite p3m1_G6h.png p3m1_G1.png -gravity West -compose Dst_Over -background none +repage p3m1_tile_part-1.png
# 4.3) extend p3m1_tile_part on the right with transparency: p3m1_tile_part-2
        convert -size {int(1/2 * $triangle_width) + 1}x{$triangle_height} xc:none transparent.png
        convert p3m1_tile_part-1.png transparent.png +append -background none +repage p3m1_tile_part-2.png
# 4.4) insert G2 at the right side: p3m1_tile_part-3
        composite p3m1_G2.png p3m1_tile_part-2.png -gravity East -compose Dst_Over -background none +repage p3m1_tile_part-3.png
# 4.5) crop1/2 G5: p3m1_G5h
        convert p3m1_G5.png -gravity West -crop {int(1/2 * $triangle_width)+1}x{$triangle_height}+0+0 -background none +repage p3m1_G5h.png
# 4.6) insert p3m1_G5h right in p3m1_tile_part-3: p3m1_tile_part
        composite p3m1_G5h.png p3m1_tile_part-3.png -gravity East -compose Dst_Over -background none +repage p3m1_tile_part.png

# 5) Generate p3m1_tile
# 5.1) Generate p3m1_tile_part_flip.png
        convert p3m1_tile_part.png -flip -background none +repage p3m1_tile_part_flip.png
# 5.2) Generate tile_row
        convert p3m1_tile_part.png p3m1_tile_part_flip.png +append -background none +repage p3m1_tile_row.png
# 5.3) Generate tile_row_flop
        convert p3m1_tile_row.png -flip -background none +repage p3m1_tile_row_flip.png
# 5.4) Generate p3m1_tile
        convert p3m1_tile_row.png p3m1_tile_row_flip.png -append -background none +repage p6m_tile.png
```

```
use Image::Magick;


# 0)Input
# 0.1) counting variable, folders, file
$p = '1'; #counting variable
$sourcefolder_path = "C:/ART/IM_PlaneCovering/1_Sourceimages/";
$sourcefile_name = 'image.jpg';
$p6m_resultfolder_path = "C:/ART/IM_PlaneCovering/2_Tiles/2_04_p6m-Tiles/";


# 0.2) definition of lokal parameters
$triangle_height_min_Faktor = 0.4;
$triangle_height_max_Faktor = 0.6;


# 1) definition of PerlMagick objects
$image_p6m = new Image::Magick;
$p6m_mask = new Image::Magick;
$transparent_extention = new Image::Magick;


# 2) inilisation of random number generator
srand;


# 3) open the source image
$sourcefile_fullname = $sourcefolder_path . $sourcefile_name;
$image_p6m->Read("$sourcefile_fullname");
```

```
# 4) local parameter calculations: $triangle_height, $triangle_width, $x1, $y1
$image_width = $image_p6m->Get('columns');
$image_height = $image_p6m->Get('rows');
if ($image_width < $image_height) {$image_min = $image_width} else {$image_min = $image_height};

$triangle_height_min = $triangle_height_min_Faktor * $image_min;
$triangle_height_max = $triangle_height_max_Faktor * $image_min;
$triangle_height = $triangle_height_min + (int(rand($triangle_height_max - $triangle_height_min)) + 1);
$triangle_width = int(1/sqrt(3) * $triangle_height) + 1;

$x_allowed = $image_width - $triangle_width;
$y_allowed = $image_height - $triangle_height;
$x1 = int(rand($x_allowed))+1;
$y1 = int(rand($y_allowed))+1;



# 5) Generate mask p6m_mask
#convert -size {$triangle_width}x{$triangle_height} xc:none -fill red -draw "polyline 0,0 $triangle_width,$triangle_width 0,$triangle_height"
p6m_mask.png
$p6m_mask->Set(size=>"$triangle_width x $triangle_height");
$p6m_mask->Read('xc:none');
$p6m_mask->Draw(primitive=>'polyline', points=>"0,0 $triangle_width,$triangle_height 0,$triangle_height", fill=>'red');



# 6) Generate image_p6m
#convert image.jpg -crop {$triangle_width}x{$triangle_height}+{$x1}+{$y1} -background none +repage image_p3m1.png
$image_p6m->Crop(geometry=>"$triangle_width x $triangle_height+$x1+$y1");
```

```
# 7) Generate p6m basic element p6m_G1
#composite p6m_mask.png image_p6m.png -matte -compose Dst_In p6m_G1.png
$p6m_G1 = $p6m_mask->Clone();
$p6m_G1->Composite(image=>$image_p6m, compose=>in, color=>'transparent', matte=>'true');


# 8) Generate p3m1_G4
# 8.1) Generate $p6m_G1_flop
#convert p6m_G1.png -flop p6m_G1_flop.png
$p6m_G1_flop = $p6m_G1->Clone();
$p6m_G1_flop->Flop();


# 8.2) Generate p3m1_G4
#convert p6m_G1_flop.png p6m_G1.png +append p3m1_G4.png
$q = ();
$q = $p6m_G1_flop->Clone();
push(@$q, $p6m_G1);
$p3m1_G4 = $q->Append(stack=>'false');



# 9) Generate G1, G2, G6, G5
# 9.1) Generate  p3m1_G1
#convert p3m1_G4.png -flip p3m1_G1.png
$p3m1_G1 = $p3m1_G4->Clone();
$p3m1_G1->Flip();
```

```
# 9.2) Generation of p3m1_G2
#convert p3m1_G1.png -flop -rotate -60 p3m1_G6b.png
#convert p3m1_G2b.png -gravity NorthEast -crop {$triangle_width}x{$triangle_height}+0+0 -background none +repage p3m1_G2.png
$p3m1_G2 = $p3m1_G1->Clone();
$p3m1_G2->Flop();
$p3m1_G2->Rotate(degrees=>60, color=>'transparent');

$p3m1_G2_width_rotated = $p3m1_G2->Get('columns');
$triangle_newwidth = 2 * $triangle_width;
$p3m1_G2_cropwidth = $p3m1_G2_width_rotated - $triangle_newwidth;
$p3m1_G2->Crop(geometry=>"$triangle_newwidth x $triangle_height + $p3m1_G2_cropwidth+0", background=>'transparent');


# 9.3) Generation of p3m1_G6
#convert p3m1_G1.png -flop -rotate -60 p3m1_G6b.png
#convert p3m1_G6b.png -crop {$triangle_newwidth}x{$triangle_height}+0+0 -background none +repage p3m1_G6.png
$p3m1_G6 = $p3m1_G1->Clone();
$p3m1_G6->Flop();
$p3m1_G6->Rotate(degrees=>-60, color=>'transparent');
$p3m1_G6->Crop(geometry=>"$triangle_newwidth x $triangle_height+0+0", background=>'transparent');


# 9.4) Generation of p3m1_G5
#convert p3m1_G6.png -flip  p3m1_G5.png
$p3m1_G5 = $p3m1_G6->Clone();
$p3m1_G5->Flip();
```

```
# 10) Generate p3m1_tile_part
# 10.1) crop 1/2 G6 left
#convert p3m1_G6.png -gravity East -crop {$triangle_width}x{$triangle_height}+{$triangle_width}+0 -background none +repage
p3m1_G6h.png
$p3m1_G6h = $p3m1_G6->Clone();
$p3m1_G6h->Crop(geometry=>"$triangle_width x $triangle_height + $triangle_width+0", background=>'transparent');


# 10.2) insert p3m1_G6h in G1 at the left side
#composite p3m1_G6h.png p3m1_G1.png -gravity West -compose Dst_Over -background none +repage p3m1_tile_part-1.png
$p3m1_tile_part = $p3m1_G1->Clone();
$p3m1_tile_part->Composite(image=>$p3m1_G6h, gravity=>West, compose=>over, color=>'transparent', matte=>'true');


# 10.3) extend p3m1_tile_part on the right with transparency
#convert -size {$triangle_width}x{$triangle_height} xc:none transparent_extention.png
#convert p3m1_tile_part-1.png transparent_extention.png +append -background none +repage p3m1_tile_part-2.png
$transparent_extention->Set(size=>"$triangle_width x $triangle_height");
$transparent_extention->Read('xc:none');
$q = ();
$q = $p3m1_tile_part->Clone();
push(@$q, $transparent_extention);
$p3m1_tile_part = $q->Append(stack=>'false');


# 10.4) insert G2 at the right side
#composite p3m1_G2.png p3m1_tile_part-2.png -gravity East -compose Dst_Over -background none +repage p3m1_tile_part-3.png
$p3m1_tile_part->Composite(image=>$p3m1_G2, gravity=>East, compose=>over, color=>'transparent', matte=>'true');
```

```
# 10.5) crop1/2 G5
# convert p3m1_G5.png -gravity West -crop {int($triangle_width)+1}x{$triangle_height}+0+0 -background none +repage p3m1_G5h.png
$p3m1_G5h = $p3m1_G5->Clone();
$p3m1_G5h->Crop(geometry=>"$triangle_width x $triangle_height+0+0",background=>'transparent');


# 10.6) insert p3m1_G5h right in p3m1_tile_part
#composite p3m1_G5h.png p3m1_tile_part-3.png -gravity East -compose Dst_Over -background none +repage p3m1_tile_part.png
$p3m1_tile_part->Composite(image=>$p3m1_G5h, gravity=>East, compose=>over, color=>'transparent', matte=>'true');


# 11) Generate p3m1_tile
# 11.1) Generate p3m1_tile_part_flip
#convert p3m1_tile_part.png -flip -background none +repage p3m1_tile_part_flip.png
$p3m1_tile_part_flip = $p3m1_tile_part->Clone();
$p3m1_tile_part_flip->Flip();


# 11.2) Generate tile_row
#convert p3m1_tile_part.png p3m1_tile_part_flip.png +append -background none +repage p3m1_tile_row.png
$q = ();
$q = $p3m1_tile_part->Clone();
push(@$q, $p3m1_tile_part_flip);
$p3m1_tile_row = $q->Append(stack=>'false');


# 11.3) Generate tile_row_flip
#convert p3m1_tile_row.png -flip -background none +repage p3m1_tile_row_flip.png
$p3m1_tile_row_flip = $p3m1_tile_row->Clone();
$p3m1_tile_row_flip->Flip();
```

```
# 11.4) Generate p3m1_tile
#convert p3m1_tile_row.png p3m1_tile_row_flip.png -append -background none +repage p3m1_tile.png
@$q = ();
$q = $p3m1_tile_row->Clone();
push(@$q, $p3m1_tile_row_flip);
$p3m1_tile = $q->Append(stack=>'true');


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

if ($p <= 9) {$resultimage_name = "p6m_tile_" . $sourcefile_name . "-0" . $p . ".jpg"}
else {$resultimage_name = "p6m_tile_" . $sourcefile_name . "-"  . $p . ".jpg"};

$resultimagepath_name = $p6m_resultfolder_path . $resultimage_name;


# 13) save result image
$p6m_tile = $p3m1_tile->Clone();
$p6m_tile->Write(filename=>"$resultimagepath_name", compression=>'JPEG', quality=>'95');
```