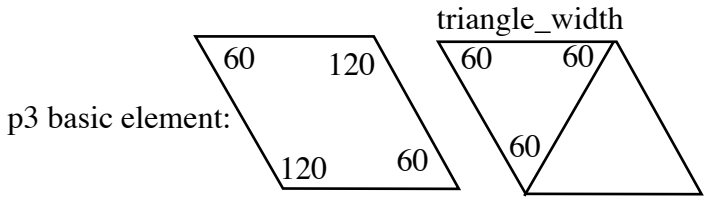
Draft V 0.8

Plane Groups with ImageMagick/PerlMagick: p3_tile

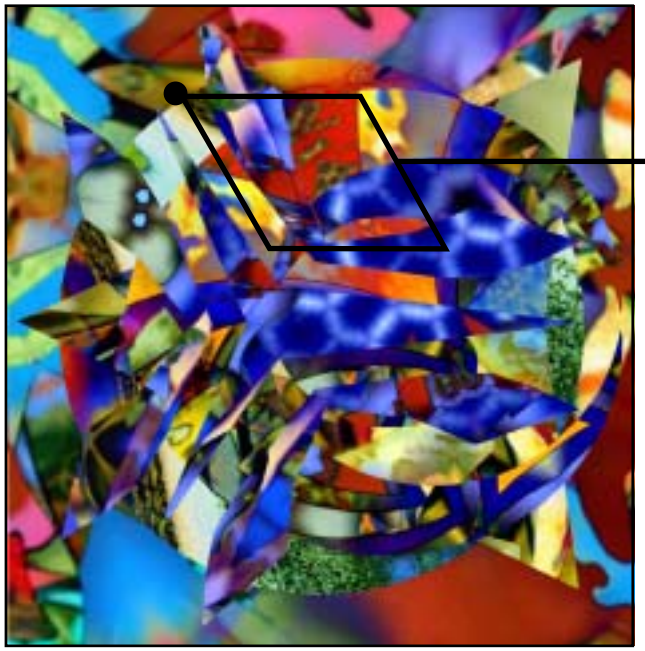
Günter Bachelier
www.aroshu.de AROSHU® Evolutionary Art © Günter Bachelier

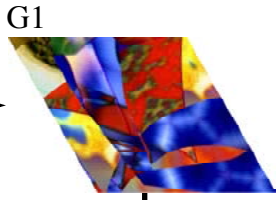


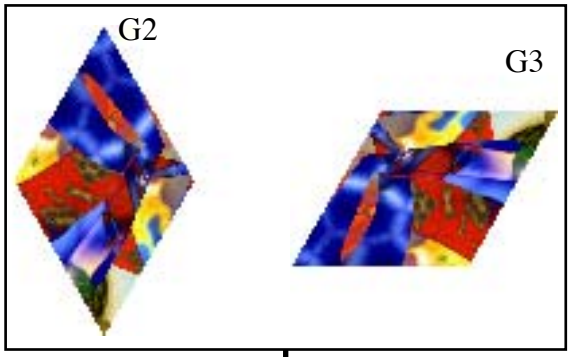
<u>given in the deterministic case</u>
a) image.jpg with (image_width, image_height)
image_width = image_height: 4000 [pixel]
b) format of the basic element: (triangle_width)
c) Top left point P1 in image.jpg for selection of image_p3

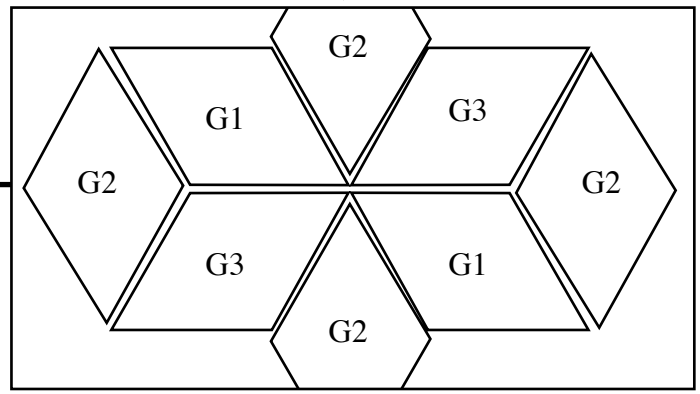
<u>given in the stochastic case</u>
a) image.jpg with (image_width, image_height)
b) random variable interval for the selection of triangle_width:
[triangle_min_Faktor, triangle_max_Faktor] = [0.4, 0.6]

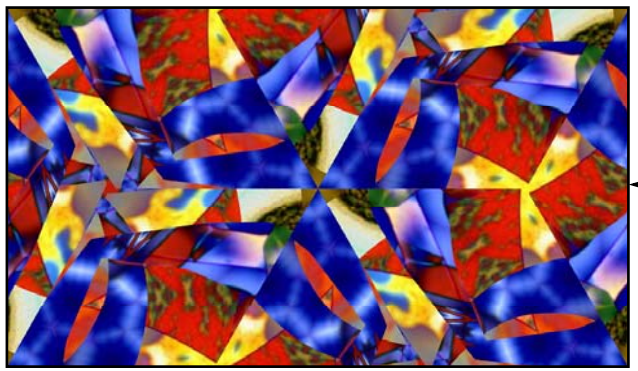
p4m-procedure:
Start
1) Generate G1, G2, G3
2) Generate G2_down, G2_up, G2_left, G2_right
3) Generate tile_part_up
4) Generate tile_part_down
5) Generate tile_center
6) Generate tile_right
7) Generate tile
End







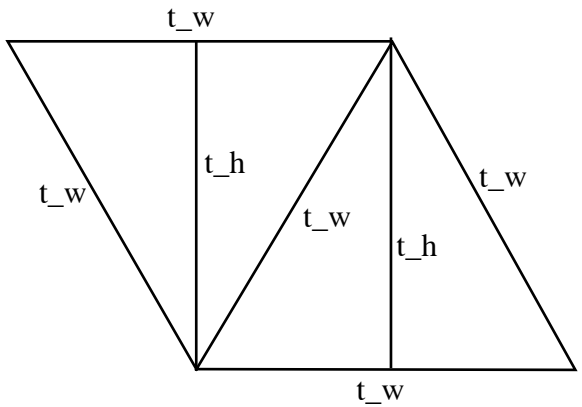


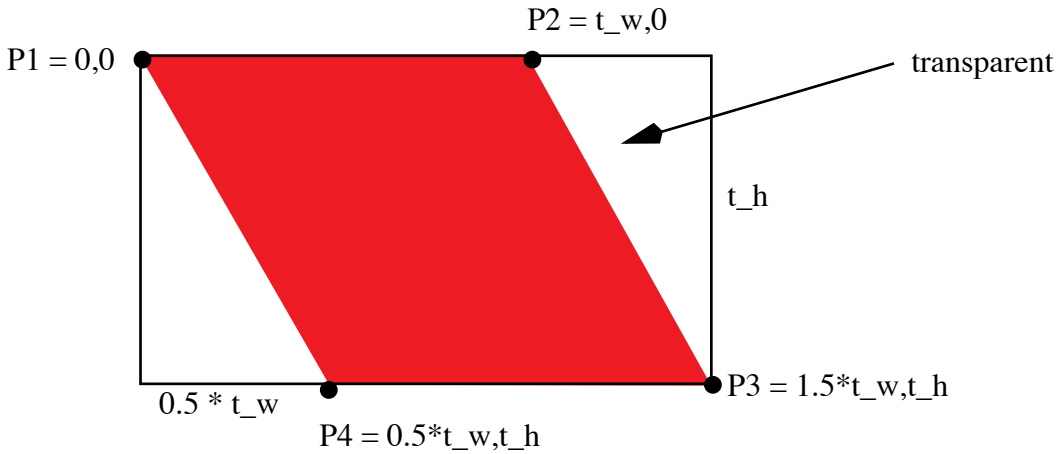


Only relevant for the stochastic case: Calculate scalare triangle_width, $triangle_height, x1, y1
$triangle_width_min_Faktor = 0.4;
$triangle_width_max_Faktor =0.6;
if ($image_width < $image_height) {$image_min = $image_width} else {$image_min = $image_height};
$triangle_width_min = $triangle_width_min_Faktor * $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(0.5 * √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;

1.1) Generate mask p3_mask.png

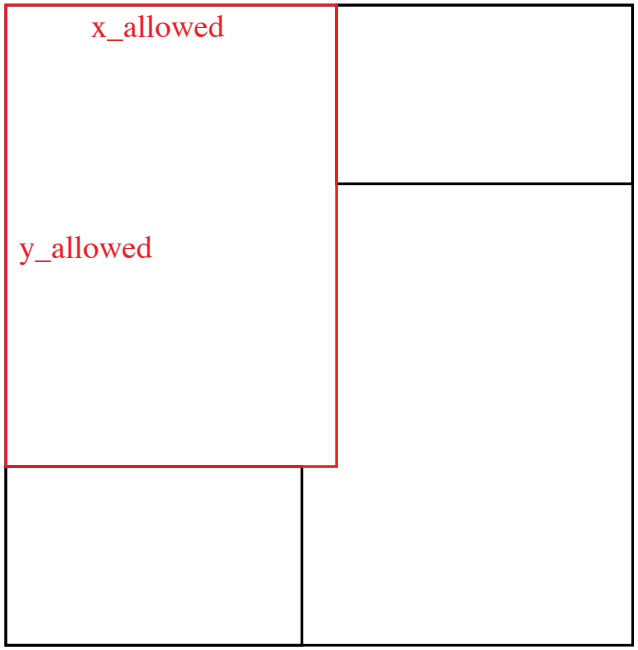
```
convert -size {1.5 * $triangle_width}x{$triangle_height} xc:none -fill red
-draw "polyline 0,0 $triangle_width,0 1.5*$triangle_width,$triangle_height 0.5*$triangle_width,$triangle_height" p3_mask.png
```

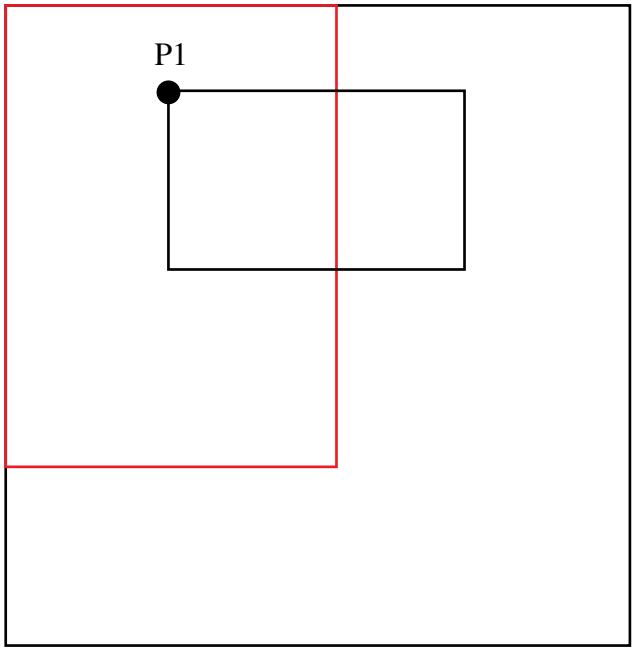


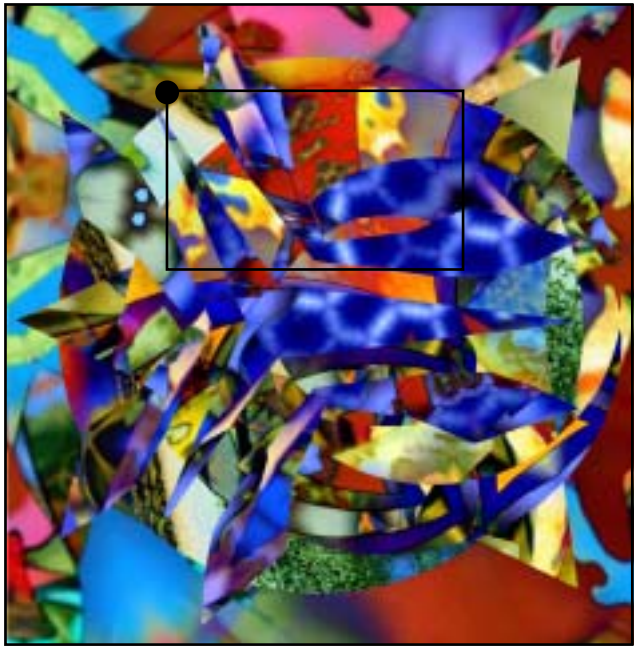


1.2) Copy from image.jpg a (random) rectancle

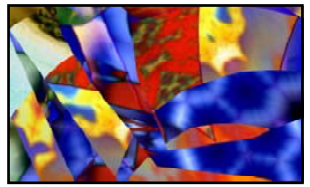
convert image.jpg -crop {1.5*$triangle_width}x{$triangle_height}+$x1+$y1 +repage image_p3.png





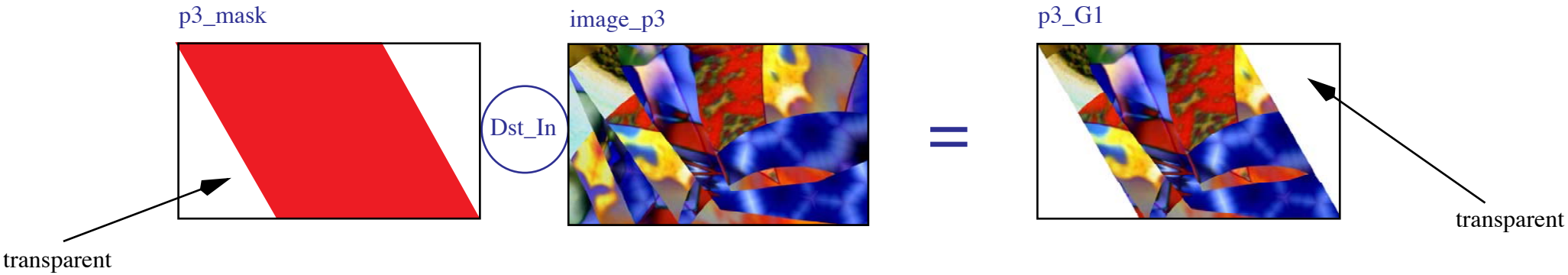


image_p3



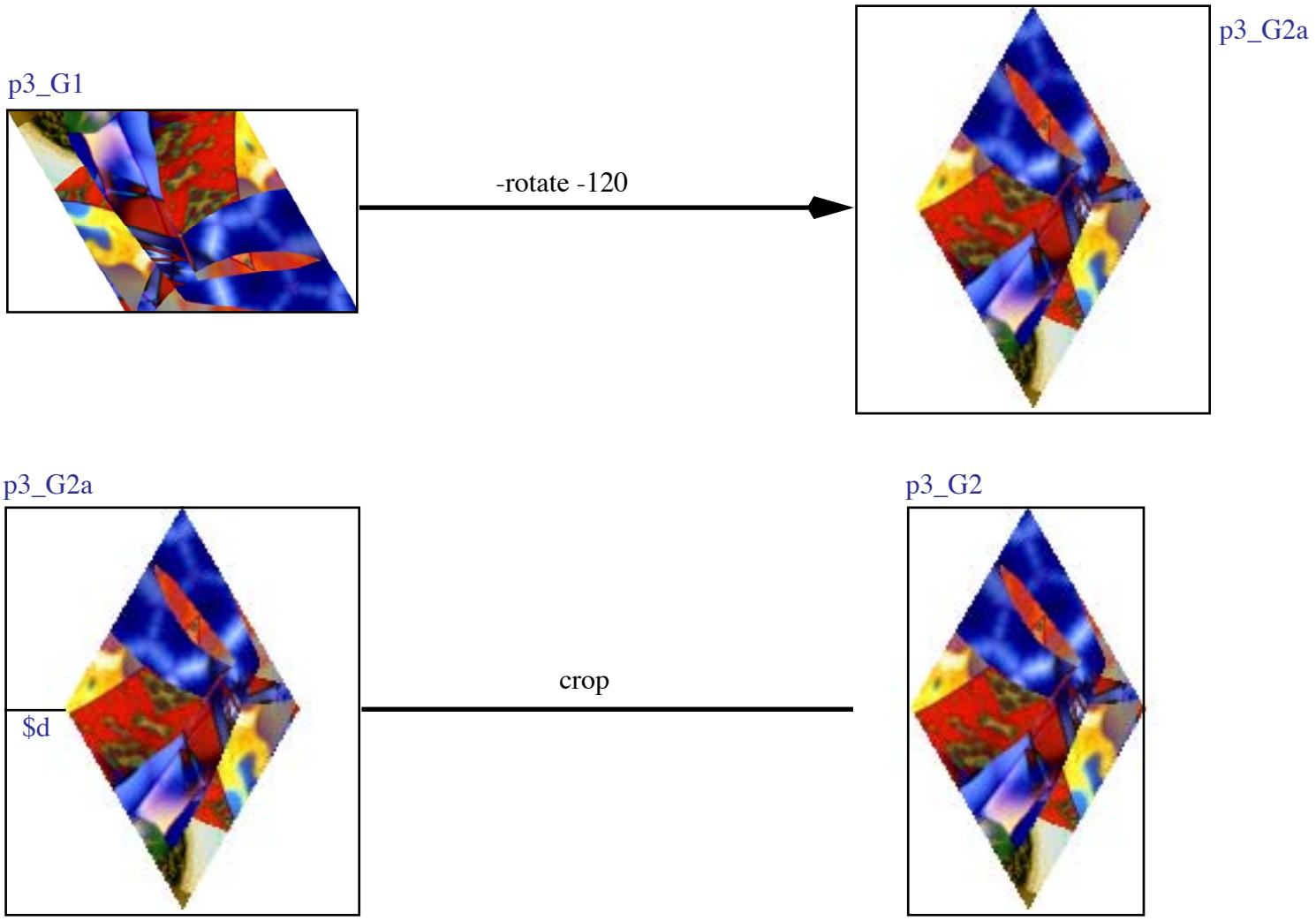
1.3) Generate p3 basic element p3_G1

composite p3_mask.png image_p3.png -matte -compose Dst_In p3_G1.png



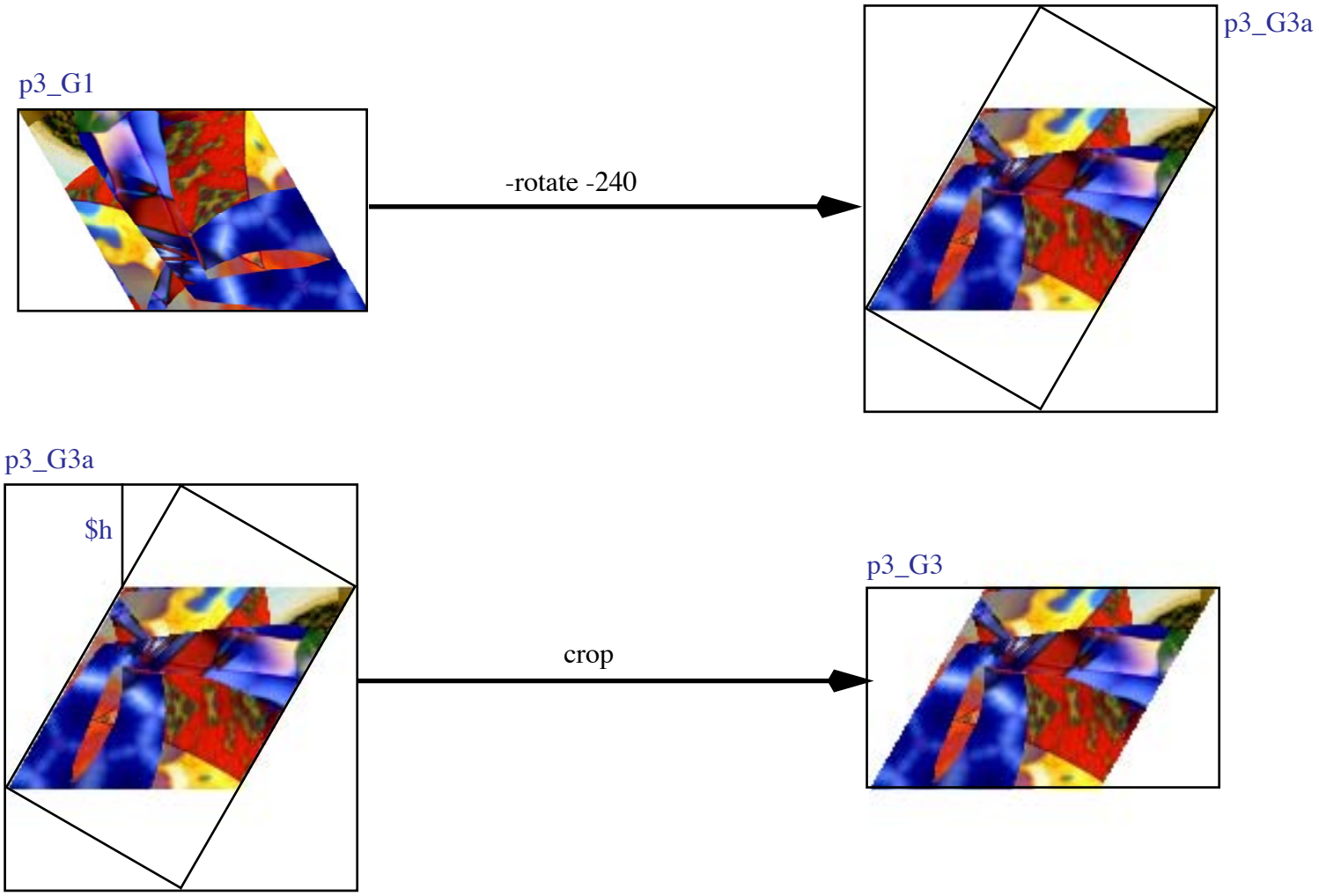
1.4) Generate with G1 the basic element G2

```
convert p3_G1.png -rotate -120 p3_G2a.png
$p3_G2a_width = $p3_G2a->get('columns');
$p3_G2a_height = $p3_G2a->get('rows');
$d = 0.5 * ($p3_G2a_width - $triangle_width);
convert p3_G2a.png -crop {$triangle_width}x{$p3_G2a_height}+$d+0 +repage p3_G2.png
```



1.5) Generate with G1 the basic element G3

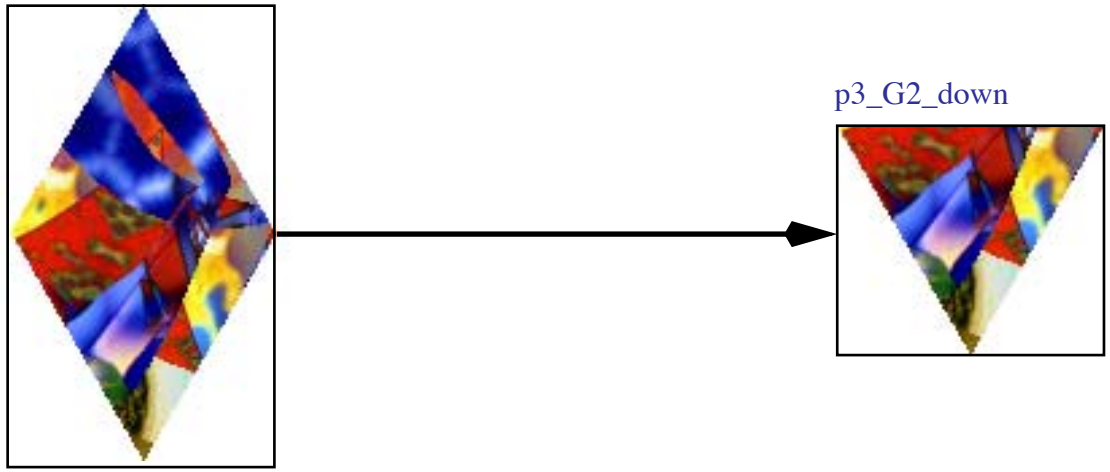
```
convert p3_G1.png -rotate -240 p3_G3a.png
$p3_G3a_height = $p3_G3a->get('rows');
$h = 0.5 * ($p3_G3a_height - $triangle_height);
convert p3_G3a.png -crop {1.5*$triangle_width}x{$triangle_height}+0+$h +repage p3_G3.png
```



2.1) Generate p3_G2_down

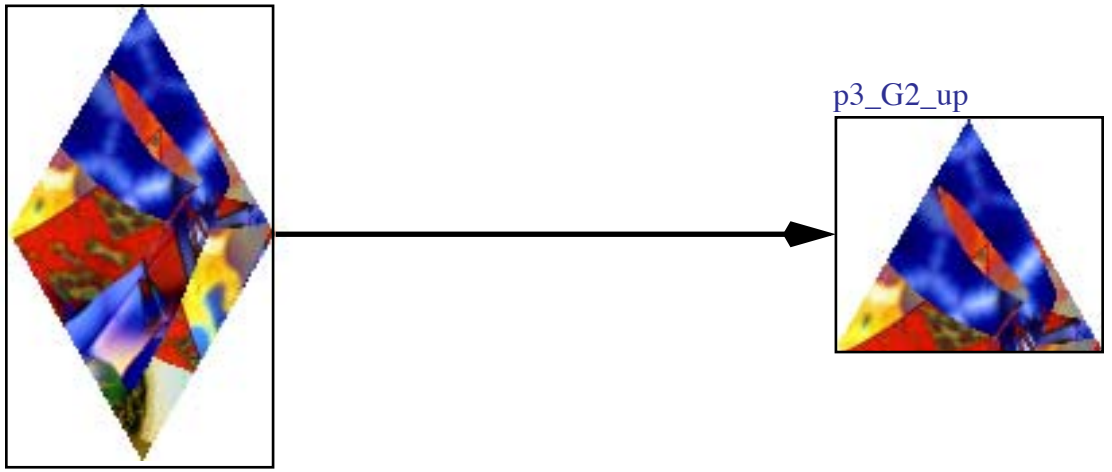
```
$p3_G2_height = $p3_G2->get('rows');
convert p3_G2.png  -crop {$triangle_width}x{0.5*$p3_G2_height}+0+{0.5*$p3_G2_height} +repage p3_G2_down.png
```

p3_G2



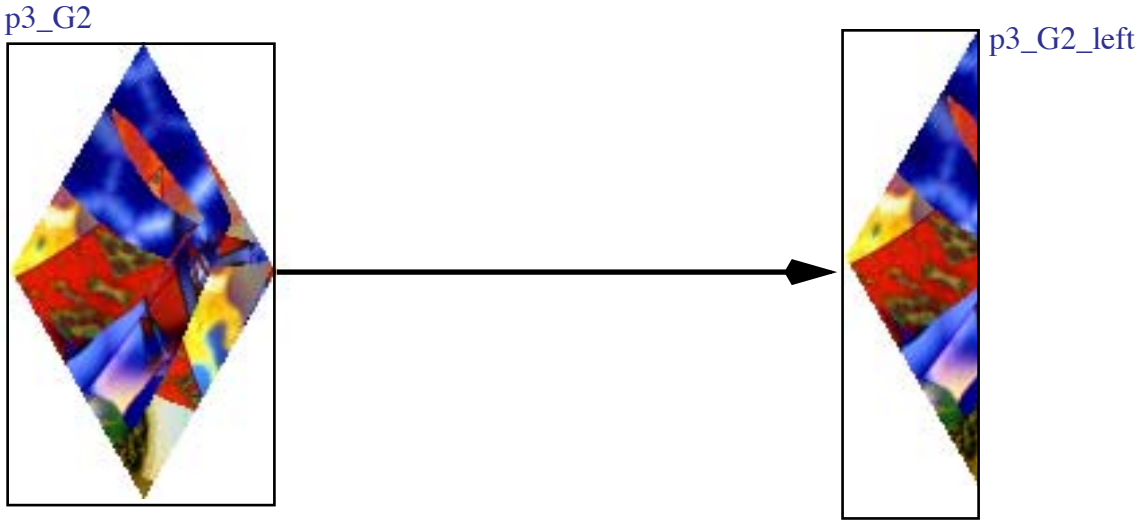
2.2) Generate p3_G2_up

```
convert p3_G2.png  -crop {$triangle_width}x{0.5*$p3_G2_height}+{0.5*$p3_G2_height}+0 +repage p3_G2_up.png
```



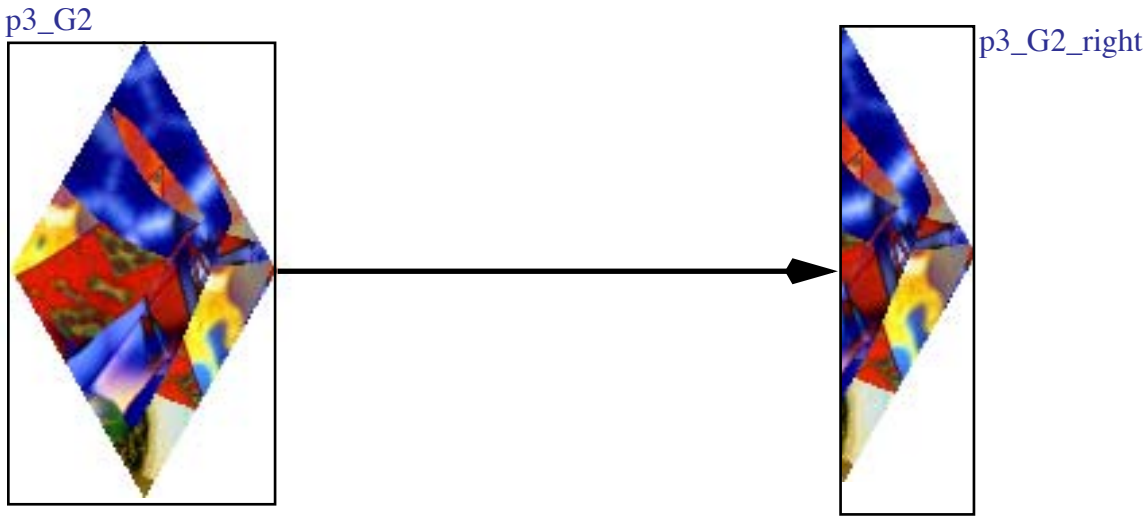
2.3) Generate p3_G2_left

```
convert p3_G2.png  -crop {0.5*$triangle_width}x{$p3_G2_height}+0+0 +repage p3_G2_left.png
```



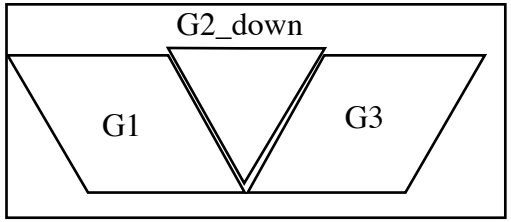
2.4) Generate p3_G2_right

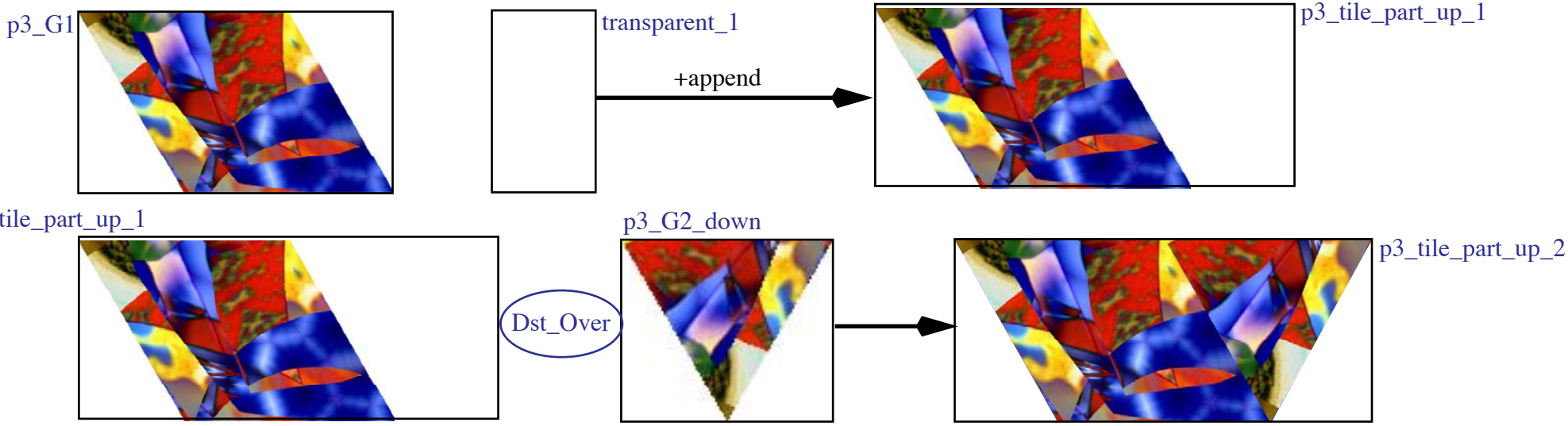
```
convert p3_G2.png  -crop {0.5*$triangle_width}x{$p3_G2_height}+{0.5*$triangle_width}+0 +repage p3_G2_right.png
```



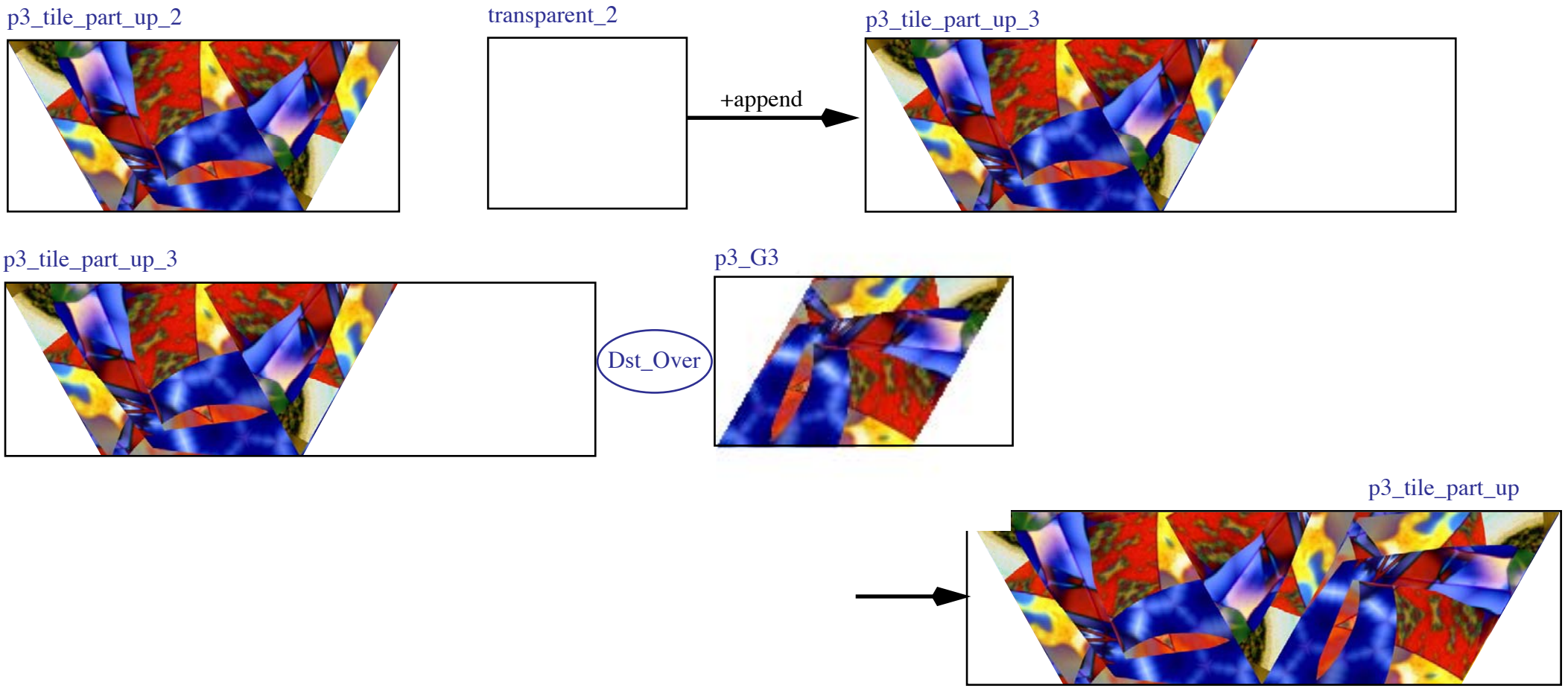
3) Generate p3_tile_part_up

```
convert -size {0.5 * $triangle_width}x{$triangle_height} xc:none transparent_1.png
convert p3_G1.png transparent_1.png +append -background none +repage p3_tile_part_up_1.png
composite p3_G2_down.png p3_tile_part_up_1.png -gravity East -compose Dst_Over p3_tile_part_up_2.png
```





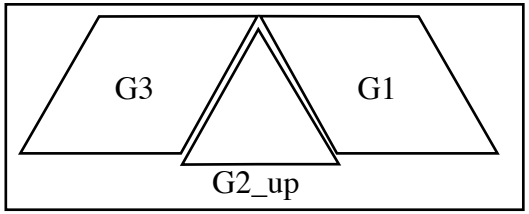
```
convert -size {$triangle_width}x{$triangle_height} xc:none transparent_2.png
convert p3_tile_part_up_2.png transparent_2.png +append -background none +repage p3_tile_part_up_3.png
composite p3_G3.png p3_tile_part_up.png -gravity East -compose Dst_Over p3_tile_part_up.png
```



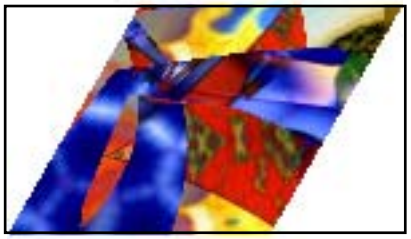
4) Generate p3_tile_part_down

convert p3_G3.png transparent_1.png +append -background none +repage p3_tile_part_down_1.png

composite p3_G2_up.png p3_tile_part_down_1.png -gravity East -compose Dst_Over p3_tile_part_up_2.png



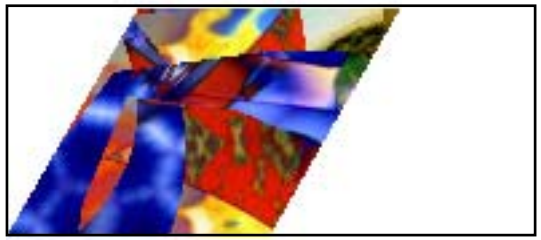
p3_G3



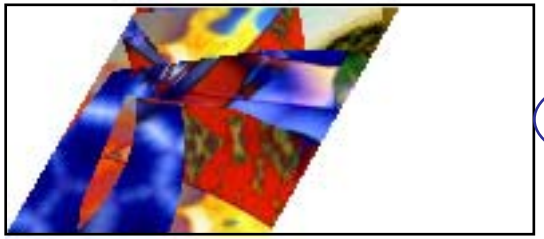
transparent_1

+append

p3_tile_part_down_1

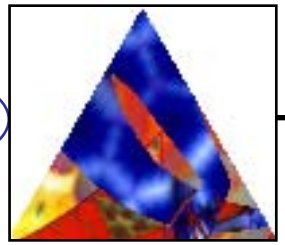


p3_tile_part_down_1

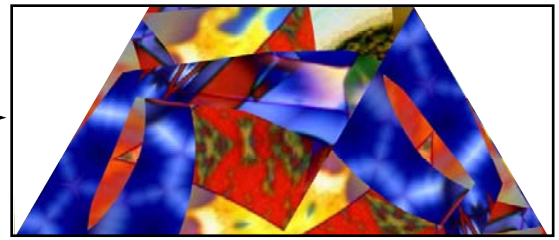


Dst_Over

p3_G2_up

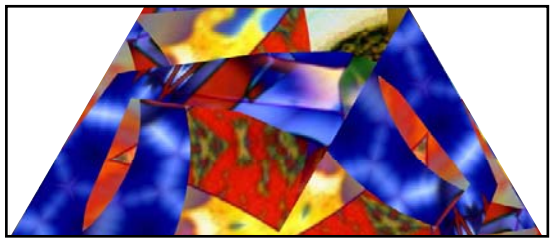


p3_tile_part_down_1

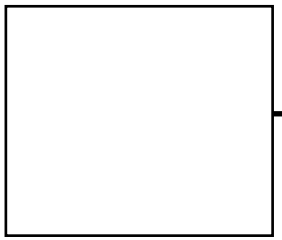


convert p3_tile_part_down_2.png transparent_2.png +append -background none +repage p3_tile_part_down_3.png
composite p3_G1.png p3_tile_part_up.png -gravity East -compose Dst_Over p3_tile_part_up.png

p3_tile_part_down_2

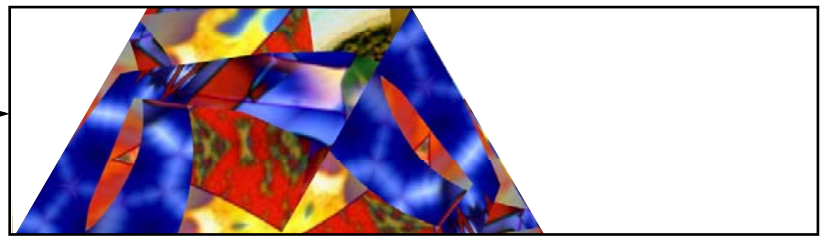


transparent_2

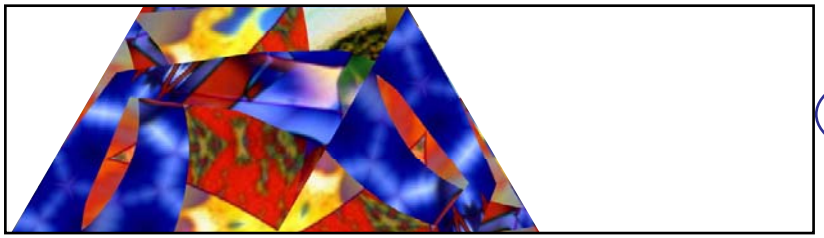


+append

p3_tile_part_down_3

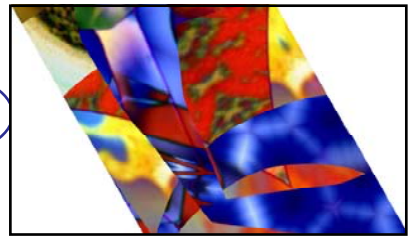


p3_tile_part_down_3

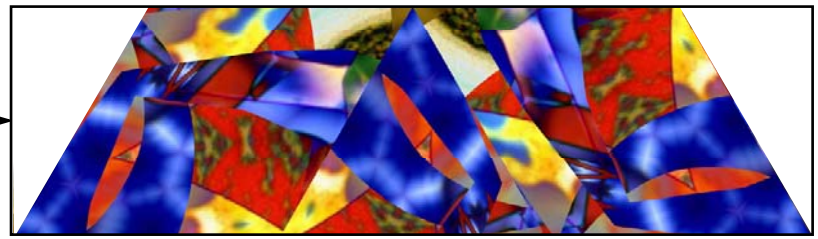


Dst_Over

p3_G1



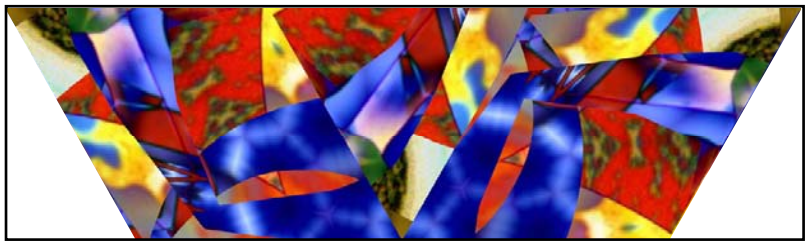
p3_tile_part_down

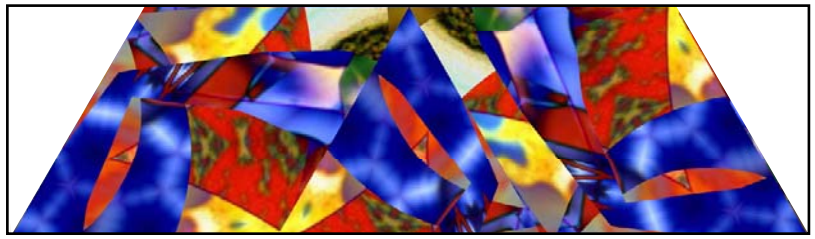


5) Generate p3_tile_part_center

```
convert p3_tile_part_up.png p3_tile_part_down.png -append -background none +repage p3_tile_part_center.png
```

p3_tile_part_up

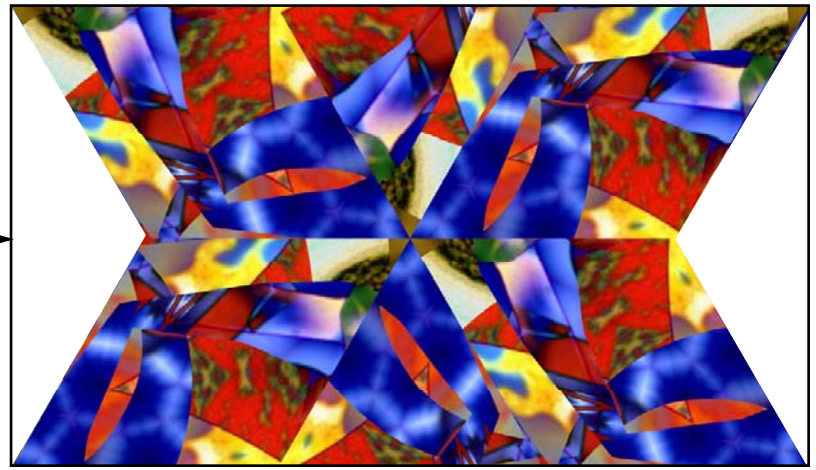




p3_tile_part_down

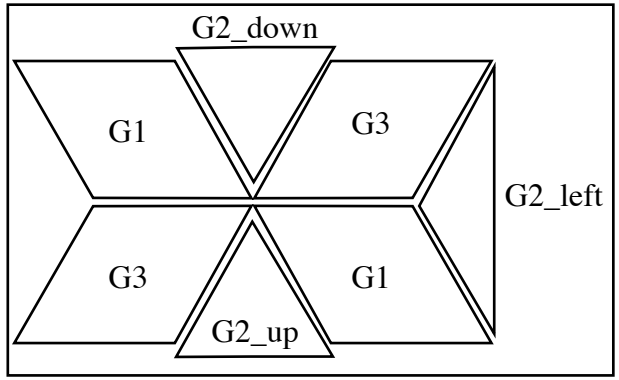
-append

p3_tile_part_center

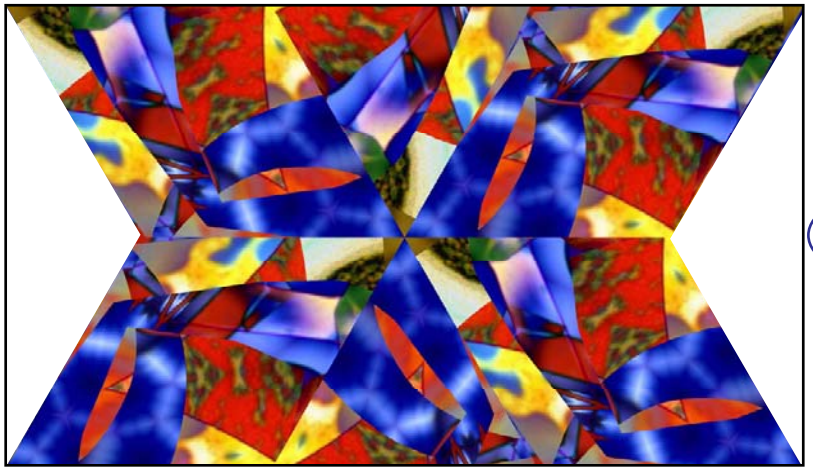


6) Generate tile_left

composite p3_G2_left.png p3_tile_part_center.png -gravity East -compose Dst_Over p3_tile_part_right.png



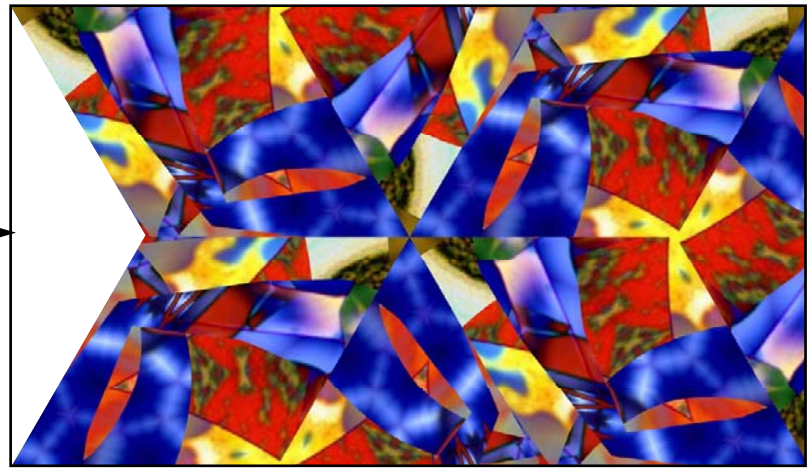
p3_tile_part_center



p3_G2_left

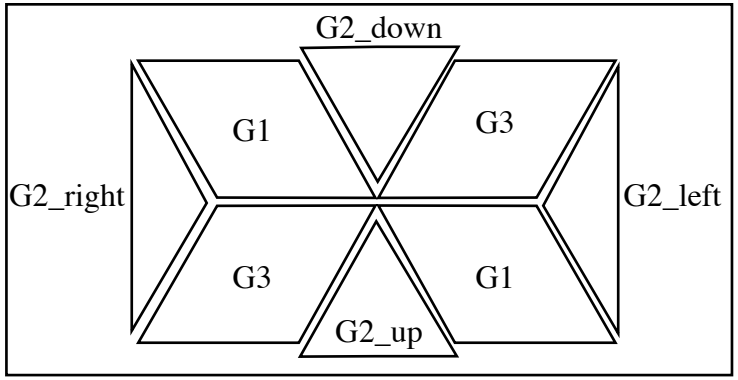
Dst_Over

p3_tile_part_right



7) Generate p3_tile

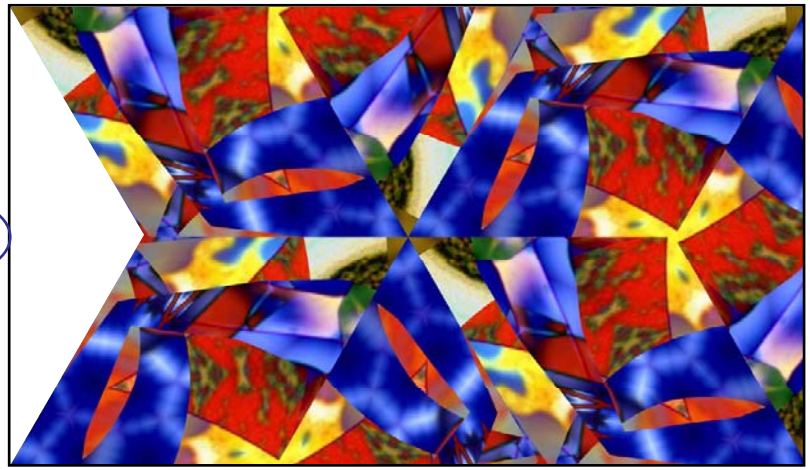
composite p3_G2_right.png p3_tile_part_right.png -gravity West -compose Dst_Over p3_tile.png



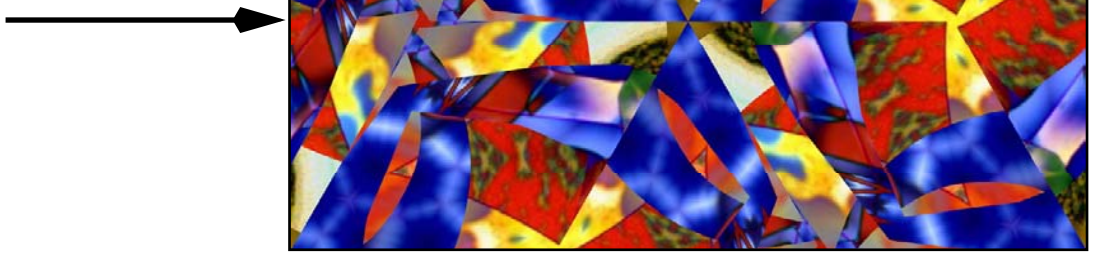
p3_G2_right

Dst_Over

p3_tile_part_right



p3_tile



IM command sequence for p3 in general

1.1) Generate mask p3_mask.png

```
convert -size {1.5 * $triangle_width}x{$triangle_height} xc:none -fill red
-draw "polyline 0,0 $triangle_width,0 1.5*$triangle_width,$triangle_height 0.5*$triangle_width,$triangle_height" p3_mask.png
```

1.2) Copy from image.jpg a (random) rectancle

```
convert image.jpg -crop {1.5*$triangle_width}x{$triangle_height}+$x1+$y1 +repage image_p3.png
```

1.3) Generate p3 basic element p3_G1

```
composite p3_mask.png image_p3.png -matte -compose Dst_In p3_G1.png
```

1.4) Generate with G1 the basic element G2

```
convert p3_G1.png -rotate -120 p3_G2a.png
$p3_G2a_width = $p3_G2a->get('columns');
$p3_G2a_height = $p3_G2a->get('rows');
$d = 0.5 * ($p3_G2a_width - $triangle_width);
convert p3_G2a.png -crop {$triangle_width}x{$p3_G2a_height}+$d+0 +repage p3_G2.png
```

1.5) Generate with G1 the basic element G3

```
convert p3_G1.png -rotate -240 p3_G3a.png
$p3_G3a_height = $p3_G3a->get('rows');
$h = 0.5 * ($p3_G3a_height - $triangle_height);
convert p3_G3a.png -crop {1.5*$triangle_width}x{$triangle_height}+0+$h +repage p3_G3.png
```

2.1) Generate p3_G2_down

```
$p3_G2_height = $p3_G2->get('rows');
convert p3_G2.png  -crop {$triangle_width}x{0.5*$p3_G2_height}+0+{0.5*$p3_G2_height} +repage p3_G2_down.png
```

2.2) Generate p3_G2_up

```
convert p3_G2.png  -crop {$triangle_width}x{0.5*$p3_G2_height}+{0.5*$p3_G2_height}+0 +repage p3_G2_up.png
```

2.3) Generate p3_G2_left

```
convert p3_G2.png  -crop {0.5*$triangle_width}x{$p3_G2_height}+0+0 +repage p3_G2_left.png
```

2.4) Generate p3_G2_right

```
convert p3_G2.png  -crop {0.5*$triangle_width}x{$p3_G2_height}+{0.5*$triangle_width}+0 +repage p3_G2_right.png
```

3) Generate p3_tile_part_up

```
convert -size {0.5 * $triangle_width}x{$triangle_height} xc:none transparent_1.png
convert p3_G1.png transparent_1.png +append -background none +repage p3_tile_part_up_1.png
composite p3_G2_down.png p3_tile_part_up_1.png -gravity East -compose Dst_Over p3_tile_part_up_2.png
convert -size {$triangle_width}x{$triangle_height} xc:none transparent_2.png
convert p3_tile_part_up_2.png transparent_2.png +append -background none +repage p3_tile_part_up_3.png
composite p3_G3.png p3_tile_part_up.png -gravity East -compose Dst_Over p3_tile_part_up.png
```

4) Generate p3_tile_part_down

```
convert p3_G3.png transparent_1.png +append -background none +repage p3_tile_part_down_1.png
composite p3_G2_up.png p3_tile_part_down_1.png -gravity East -compose Dst_Over p3_tile_part_up_2.png
convert p3_tile_part_down_2.png transparent_2.png +append -background none +repage p3_tile_part_down_3.png
composite p3_G1.png p3_tile_part_up.png -gravity East -compose Dst_Over p3_tile_part_up.png
```

5) Generate p3_tile_part_center

```
convert p3_tile_part_up.png p3_tile_part_down.png -append -background none +repage p3_tile_part_center.png
```

6) Generate tile_left

```
composite p3_G2_left.png p3_tile_part_center.png -gravity East -compose Dst_Over p3_tile_part_right.png
```

7) Generate p3_tile

```
composite p3_G2_right.png p3_tile_part_right.png -gravity West -compose Dst_Over p3_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';
$p3m1_resultfolder_path = "C:/ART/IM_PlaneCovering/2_Tiles/2_05_p3-Tiles/";



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



# 1) definition of PerlMagick objects
$image_p3 = new Image::Magick;
$p3_mask = new Image::Magick;
$transparent_extention_1 = new Image::Magick;
$transparent_extention_2 = new Image::Magick;



# 2) inilisation of random number generator
srand;
```

```
# 3) open the source image
my $sourcefile_fullname = $sourcefolder_path . $sourcefile_name;
$image_p3->Read("$sourcefile_fullname");



# 4) local parameter calculations: $triangle_width, $triangle_height, $x1, $y1
my $image_width = $image_p3->Get('columns');
my $image_height = $image_p3->Get('rows');
if ($image_width < $image_height) {my $image_min = $image_width} else {$image_min = $image_height};
my $triangle_width_min = $triangle_width_min_Faktor * $image_min;
my $triangle_width_max = $triangle_width_max_Faktor * $image_min;
my $triangle_width = $triangle_width_min + (int(rand($triangle_width_max - $triangle_width_min)) + 1);
my $triangle_height = int(1/2 * sqrt(3) * $triangle_width)+1;
my $x_allowed = $image_width - $triangle_width;
my $y_allowed = $image_height - $triangle_height;
my $x1 = int(rand($x_allowed))+1;
my $y1 = int(rand($y_allowed))+1;



# 5) Generate mask p3_mask
#convert -size {1.5 * $triangle_width}x{$triangle_height} xc:none -fill red -draw "polyline 0,0 $triangle_width,0
1.5*$triangle_width,$triangle_height 0.5*$triangle_width,$triangle_height" p3_mask.png
my $triangle_halfwidth = int(1/2 * $triangle_width)+1;
my $triangle_onehalfwidth = int(3/2 * $triangle_width)+1;
$p3_mask->Set(size=>"$triangle_onehalfwidth x $triangle_height");
$p3_mask->Read('xc:none');
$p3_mask->Draw(primitive=>'polyline', points=>"0,0 $triangle_width,0 $triangle_onehalfwidth,$triangle_height
$triangle_halfwidth,$triangle_height", fill=>'red');
$mask_width = $p3_mask->get('columns');
```

```
# 6) Generate image_p3
#convert image.jpg -crop {1.5*$triangle_width}x{$triangle_height}+$x1+$y1 +repage image_p3.png
#$image_p3->Crop(geometry=>"$triangle_onehalfwidth x $triangle_height+$x1+$y1"); # $triangle_onehalfwidth don't work see below

$image_p3->Crop(geometry=>"$mask_width x $triangle_height+$x1+$y1");

# odd behavior when width of image_p3 is set to $triangle_onehalfwidth, this differs from $mask_width, see print experiments
# no problem with image height and mask height
# this differentation causes unacceptable overlappings between image and mask
# solution is to set the width of image_p3 explicit to $mask_width
#my $mask_width = $p3_mask->get('columns');
#$image_p3_width = $image_p3->get('columns');
#my $mask_hight = $p3_mask->get('rows');
#my $image_p3_hight = $image_p3->get('rows');
#print "$mask_width $image_p3_width";



# 7) Generate G1, G2, G3
# 7.1) Generate  p3_G1
#composite p3_mask.png image_p3.png -matte -compose Dst_In -background none +repage p3_G1.png #gravity=>Center
$p3_G1 = $p3_mask->Clone();
$p3_G1->Composite(image=>$image_p3, compose=>in,  color=>'transparent', matte=>'true');

#$p3_G1->Write(filename=>'p3_G1.jpg', compression=>'JPEG', quality=>'95');



# 7.2) Generate  p3_G2
#convert p3_G1.png -rotate -120 p3_G2a.png
#$p3_G2a_width = $p3_G2a->get('columns');
#$p3_G2a_height = $p3_G2a->get('rows');
```

```
#$d = 0.5 * ($p3_G2a_width - $triangle_width);
#convert p3_G2a.png -crop {$triangle_width}x{$p3_G2a_height}+$d+0 +repage p3_G2.png

$p3_G2 = $p3_G1->Clone();
$p3_G2->Rotate(degrees=>-120, color=>'transparent');
my $p3_G2a_width = $p3_G2->get('columns');
my $p3_G2a_height = $p3_G2->get('rows');
my $d = int(0.5 * ($p3_G2a_width - $triangle_width))+1;
$p3_G2->Crop(geometry=>"$triangle_width x $p3_G2a_height+$d+0", background=>'transparent');

#$p3_G2->Write(filename=>'p3_G2.jpg', compression=>'JPEG', quality=>'95');



# 7.3) Generate  p3_G3
#convert p3_G1.png -rotate -240 p3_G3a.png
#$p3_G3a_height = $p3_G3a->get('rows');
#$h = 0.5 * ($p3_G3a_height - $triangle_height);
#convert p3_G3a.png -crop {1.5*$triangle_width}x{$triangle_height}+0+$h +repage p3_G3.png

$p3_G3 = $p3_G1->Clone();
$p3_G3->Rotate(degrees=>-240, color=>'transparent');
my $p3_G3a_width = $p3_G3->get('columns');
my $p3_G3a_height = $p3_G3->get('rows');
my $h = int(1/2 * ($p3_G3a_height - $triangle_height))+1;
$p3_G3->Crop(geometry=>"$triangle_onehalfwidth x $triangle_height+0+$h", background=>'transparent');

#$p3_G3->Write(filename=>'p3_G3.jpg', compression=>'JPEG', quality=>'95');
```

```
# 8) Generate p3_G2_down, p3_G2_up, p3_G2_left, p3_G2_right
#8.1) Generate p3_G2_down
#$p3_G2_height = $p3_G2->get('rows');
#convert p3_G2.png  -crop {$triangle_width}x{0.5*$p3_G2_height}+0+{0.5*$p3_G2_height} +repage p3_G2_down.png

my $p3_G2_height = $p3_G2->get('rows');
my $p3_G2_halfheight = int(1/2 * $p3_G2_height)+1;
$p3_G2_down = $p3_G2->Clone();
$p3_G2_down->Crop(geometry=>"$triangle_onehalfwidth x $p3_G2_halfheight+ 0+ $p3_G2_halfheight", background=>'transparent');

#$p3_G2_down->Write(filename=>'p3_G2_down.jpg', compression=>'JPEG', quality=>'95');



#8.2) Generate p3_G2_up
#convert p3_G2.png  -crop {$triangle_width}x{0.5*$p3_G2_height}+{0.5*$p3_G2_height}+0 +repage p3_G2_up.png
$p3_G2_up = $p3_G2->Clone();
$p3_G2_up->Crop(geometry=>"$triangle_onehalfwidth x $p3_G2_halfheight+0+0", background=>'transparent');

#$p3_G2_up->Write(filename=>'p3_G2_up.jpg', compression=>'JPEG', quality=>'95');



#8.3) Generate p3_G2_left
#convert p3_G2.png  -crop {0.5*$triangle_width}x{$p3_G2_height}+0+0 +repage p3_G2_left.png
$p3_G2_left = $p3_G2->Clone();
my $triangle_threefourwidth = int(1/2 * $triangle_onehalfwidth)+1;
$p3_G2_left->Crop(geometry=>"$triangle_threefourwidth x $p3_G2_height+0+0", background=>'transparent');

#$p3_G2_left->Write(filename=>'p3_G2_left.jpg', compression=>'JPEG', quality=>'95');
```

```
#8.4) Generate p3_G2_right
#convert p3_G2.png  -crop {0.5*$triangle_width}x{$p3_G2_height}+{0.5*$triangle_width}+0 +repage p3_G2_right.png
$p3_G2_right = $p3_G2->Clone();
$triangle_threefourwidth_plus1 = $triangle_threefourwidth + 1; # manual correction because of edge artifacts
$triangle_threefourwidth_minus1 = $triangle_threefourwidth - 1; # manual correction because of edge artifacts
$p3_G2_right->Crop(geometry=>"$triangle_threefourwidth_plus1 x $p3_G2_height+ $triangle_threefourwidth_minus1 +0", back-
ground=>'transparent');

#$p3_G2_right->Write(filename=>'p3_G2_right.jpg', compression=>'JPEG', quality=>'95');



#9) Generate p3_tile_part_up
#convert -size {0.5 * $triangle_width}x{$triangle_height} xc:none transparent_1.png
#convert p3_G1.png transparent_1.png +append -background none +repage p3_tile_part_up_1.png
#composite p3_G2_down.png p3_tile_part_up_1.png -gravity East -compose Dst_Over p3_tile_part_up_2.png

$transparent_extention_1->Set(size=>"$triangle_halfwidth x $triangle_height");
$transparent_extention_1->Read('xc:none');
my $q = $p3_G1->Clone();
push(@$q, $transparent_extention_1);
$p3_tile_part_up = $q->Append(stack=>'false');
$p3_tile_part_up->Composite(image=>$p3_G2_down, gravity=>East, compose=>over, color=>'transparent', matte=>'true');


#convert -size {$triangle_width}x{$triangle_height} xc:none transparent_2.png
#convert p3_tile_part_up_2.png transparent_2.png +append -background none +repage p3_tile_part_up_3.png
#composite p3_G3.png p3_tile_part_up.png -gravity East -compose Dst_Over p3_tile_part_up.png

$transparent_extention_2->Set(size=>"$triangle_width x $triangle_height");
$transparent_extention_2->Read('xc:none');
```

```
@$q = ();
$q = $p3_tile_part_up->Clone();
push(@$q, $transparent_extention_2);
$p3_tile_part_up = $q->Append(stack=>'false');
$p3_tile_part_up->Composite(image=>$p3_G3, gravity=>East, compose=>over, color=>'transparent', matte=>'true');

#$p3_tile_part_up->Write(filename=>'p3_tile_part_up.jpg', compression=>'JPEG', quality=>'95');



#10) Generate p3_tile_part_down
#convert p3_G3.png transparent_1.png +append -background none +repage p3_tile_part_down_1.png
#composite p3_G2_up.png p3_tile_part_down_1.png -gravity East -compose Dst_Over p3_tile_part_up_2.png

@$q = ();
$q = $p3_G3->Clone();
push(@$q, $transparent_extention_1);
$p3_tile_part_down = $q->Append(stack=>'false');
$p3_tile_part_down->Composite(image=>$p3_G2_up, gravity=>East, compose=>over, color=>'transparent', matte=>'true');


#convert p3_tile_part_down_2.png transparent_2.png +append -background none +repage p3_tile_part_down_3.png
#composite p3_G1.png p3_tile_part_up.png -gravity East -compose Dst_Over p3_tile_part_up.png
@$q = ();
$q = $p3_tile_part_down->Clone();
push(@$q, $transparent_extention_2);
$p3_tile_part_down = $q->Append(stack=>'false');
$p3_tile_part_down->Composite(image=>$p3_G1, gravity=>East, compose=>over, color=>'transparent', matte=>'true');

#$p3_tile_part_down->Write(filename=>'p3_tile_part_down.jpg', compression=>'JPEG', quality=>'95');
```

```
#11) Generate p3_tile_part_center
#convert p3_tile_part_up.png p3_tile_part_down.png -append -background none +repage p3_tile_part_center.png

@$q = ();
$q = $p3_tile_part_up->Clone();
push(@$q, $p3_tile_part_down);
$p3_tile_part_center = $q->Append(stack=>'true');

#$p3_tile_part_center->Write(filename=>'p3_tile_part_center.jpg', compression=>'JPEG', quality=>'95');



#12) Generate tile_left
#composite p3_G2_left.png p3_tile_part_center.png -gravity East -compose Dst_Over p3_tile_part_right.png

$p3_tile_left = $p3_tile_part_center->Clone();
$p3_tile_left->Composite(image=>$p3_G2_left, gravity=>East, compose=>over, color=>'transparent', matte=>'true');

#$p3_tile_left->Write(filename=>'p3_tile_part_left.jpg', compression=>'JPEG', quality=>'95');



#13) Generate p3_tile
#composite p3_G2_right.png p3_tile_part_right.png -gravity West -compose Dst_Over p3_tile.png

$p3_tile = $p3_tile_left->Clone();
$p3_tile->Composite(image=>$p3_G2_right, gravity=>West, compose=>over, color=>'transparent', matte=>'true');

#$p3_tile->Write(filename=>'p3_tile.jpg', compression=>'JPEG', quality=>'95');
```

```
# 14) make comment with parameter values
my $comment_string = $sourcefile_name . "*" . $triangle_width . "*" . $x1 . "*" . $y1 . "*";
my $p3_comment = $p3_tile->Comment($comment_string);



# 15) generate name of result image
if ($p <= 9) {$resultimage_name = "p3_tile_" . $sourcefile_name . "-0" . $p . ".jpg"};
if ($p > 9 and $p <= 99)  {$resultimage_name = "p3_tile_" . $sourcefile_name . "-"  . $p . ".jpg"};

$resultimagepath_name = $p3_resultfolder_path . $resultimage_name;



# 16) save result image
$p3_tile->Write(filename=>"$resultimagepath_name", compression=>'JPEG', quality=>'95');
```