Programmable 2D Arrangements for Element Texture Design

HUGO LOI
Inria-LJK (UGA, CNRS)
and
THOMAS HURTUT
École Polytechnique de Montréal
and
ROMAIN VERGNE and JOELLE THOLLOT
Inria-LJK (UGA, CNRS)

Target textures

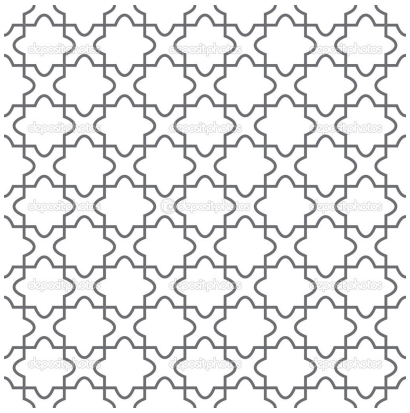


Fig. 1. “puzzle”

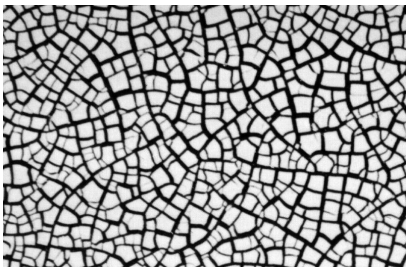


Fig. 2. “cracks”

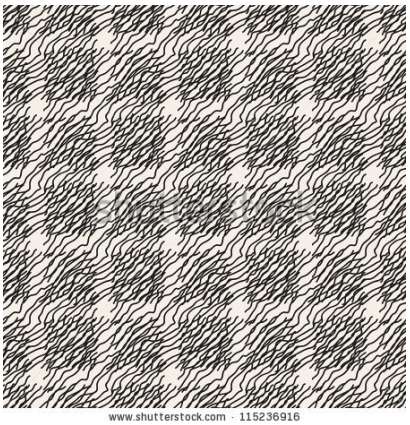
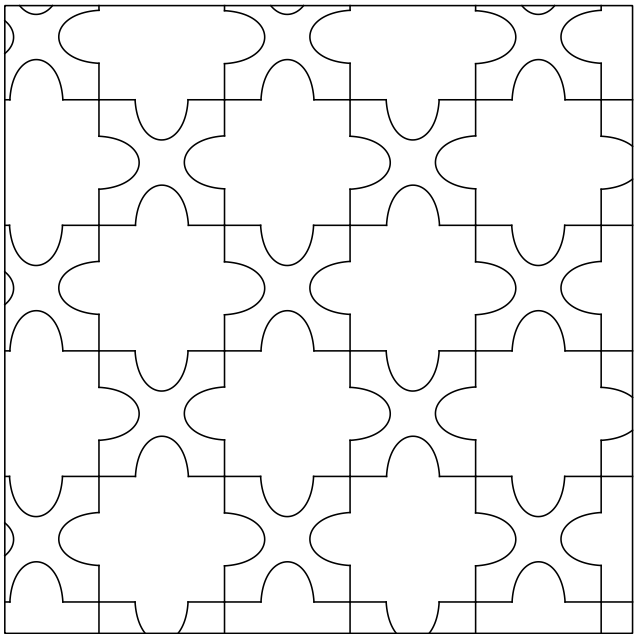


Fig. 3. “waves”

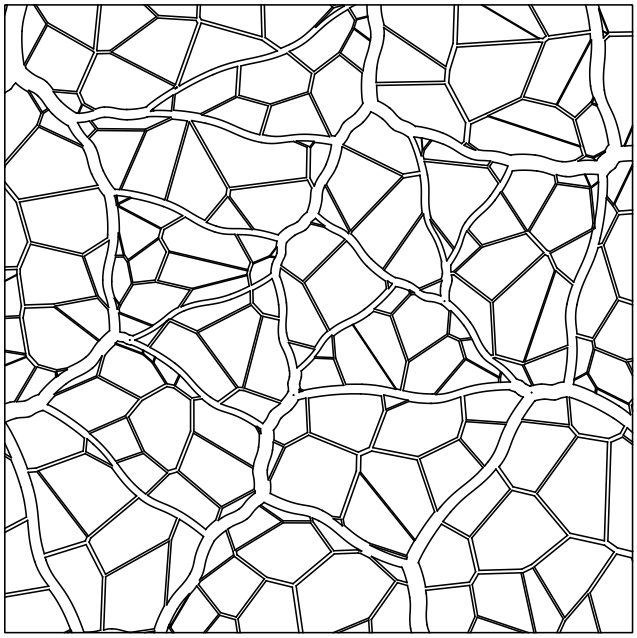
User 1 - puzzle. 21 lines, two operators (green), three executions. User satisfaction: 10/10

```
def test2():
    line = ImportSVG("data/estelle/bitonio.svg")

    def edge_to_bitonio(edge):
        dst_c = PointLabeled(line,"start")
        src_c = PointLabeled(line,"end")
        if (HasLabel(LeftFace(edge), "v1") and HasLabel(
            LeftFace(edge), "h1") ):
            src_c = PointLabeled(line,"start")
            dst_c = PointLabeled(line,"end")
        elif (HasLabel(LeftFace(edge),"v2") and HasLabel(
            LeftFace(edge),"h2")):
            src_c = PointLabeled(line,"start")
            dst_c = PointLabeled(line,"end")
        src_v = Location(SourceVertex(edge))
        dst_v = Location(TargetVertex(edge))
        return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    lines1 = StripesProperties(0,200)
    lines2 = StripesProperties(pi/2,200)
    SetFaceLabels(lines1,"h1","h2")
    SetFaceLabels(lines2,"v1","v2")
    grid_tex = GridPartition(lines1,lines2,KEEP_OUTSIDE)
    part = MapToEdges(edge_to_bitonio, grid_tex)

    ExportSVG(part, 1000)
```



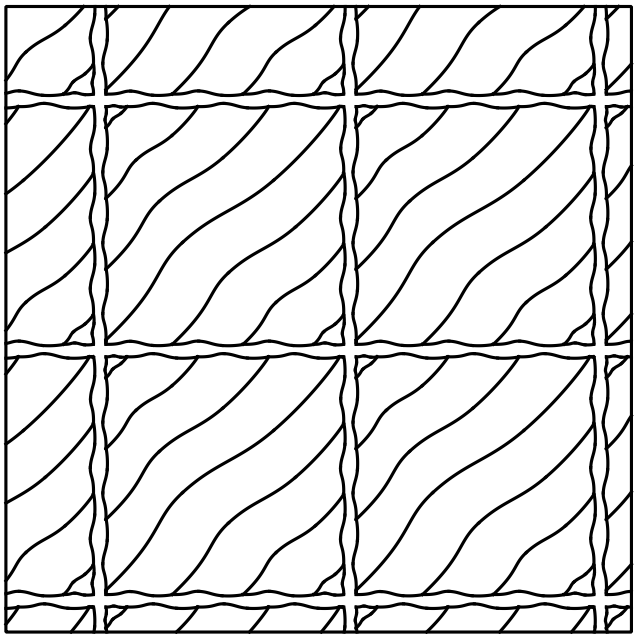
User 1 - cracks. 20 lines, six operators (green), three executions. User satisfaction: 10/10

```
def test3():
    props    = IrregularProperties(1/100000)
    init_tex = RandomPartition(props,KEEP_OUTSIDE)

    def hatch_map(face):
        props    = IrregularProperties(1/10000)
        return RandomPartition(props, CROP_ADD_BOUNDARY)(face)

    line     = ImportSVG("data/line6.svg")
    def line_to_curve(edge):
        src_c = PointLabeled(line,"start")
        dst_c = PointLabeled(line,"end")
        src_v = Location(SourceVertex(edge))
        dst_v = Location(TargetVertex(edge))
        return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    def scale_map(face):
        return Scale(Contour(face),0.95)

    texture = MapToEdges(line_to_curve,init_tex)
    texture = MapToFaces(scale_map, texture)
    texture = MapToFaces(hatch_map, texture)
    texture = MapToFaces(scale_map, texture)

    ExportSVG(texture, 1000)
```



User 1 - waves. 21 lines, five operators (green), three executions. User satisfaction: 8/10

```
def test4():
    lines1   = StripesProperties(0,200)
    lines2   = StripesProperties(pi/2.0,200)
    grid_tex = GridPartition(lines1,lines2,KEEP_OUTSIDE)

    line     = ImportSVG("data/line6.svg")
    def line_to_curve(edge):
        src_c = PointLabeled(line,"start")
        dst_c = PointLabeled(line,"end")
        src_v = Location(SourceVertex(edge))
        dst_v = Location(TargetVertex(edge))
        return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    def scale_map(face):
        return Scale(Contour(face),0.95)

    def hatch_map(face):
        props   = StripesProperties(pi/4.0,40)
        zorglub = StripesPartition(props)
        bloub = MapToEdges(line_to_curve, zorglub)
        return bloub(face)

    texture = MapToFaces(scale_map, grid_tex)
    texture = MapToFaces(hatch_map, texture)

    ExportSVG(texture, 500)
```

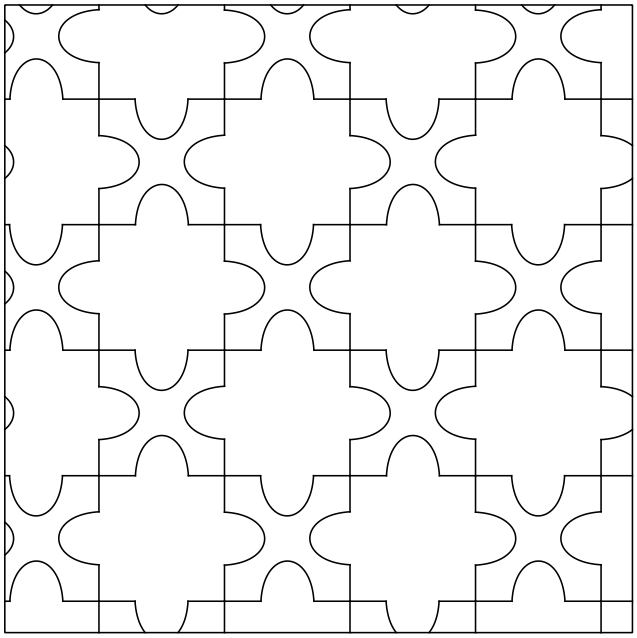


Interview of User 1

How easy was it to decide what you would do in order to reach the target designs? Extremely easy. No difficulty. The decomposition process looks very natural because it corresponds to the way our brain decomposes images into separate structures.
How easy was it to realize your plans by scripting in our tool? Not so hard. With a little bit of practice it becomes easy, and this learning appears to take little time. I would say labels took me the most time to handle in practice.
How often did you loose the understanding of what your script was doing? Never. The scripts are short and the syntax is clear. No problem.
How did you feel about the general principle of designing textures with our partitions+mappers+combinations? It works quite well. It fits our natural way to split structures into simpler parts. I liked this set of operators.
What are your thoughts about what you liked or disliked while experiencing our tool? The system itself is not far from being extremely comfortable. I would say the synthesis speed is the thing to be improved, because some trial-and-errors are slown down when the texture becomes complex.

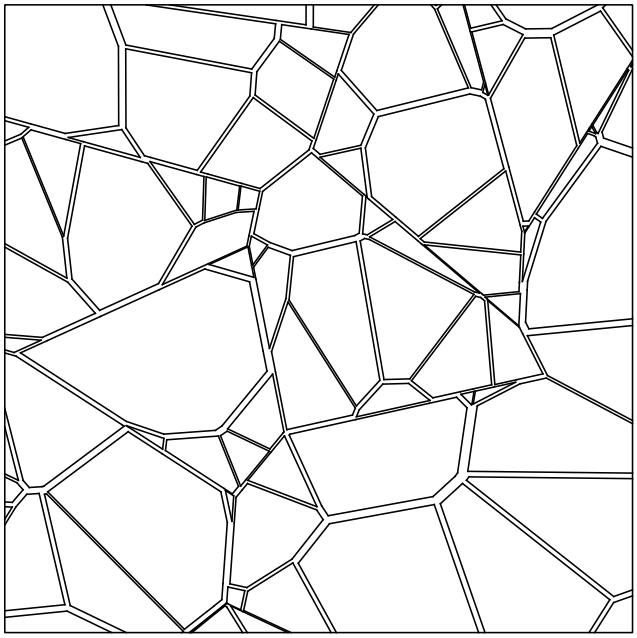
User 2 - puzzle. 26 lines, two operators (green), three executions. User satisfaction: 10/10

```
def test3():
    lines1 = StripesProperties(0,200)
    lines2 = StripesProperties(pi/2,200)
    SetFaceLabels(lines1,"h1","h2")
    SetFaceLabels(lines2,"v1","v2")
    grid_tex = GridPartition(lines1,lines2,KEEP_OUTSIDE)

    line = ImportSVG("data/estelle/bitonio.svg")

    def line_to_curve(edge):
        if IsBoundary(edge):
            return Nothing()

        face = LeftFace(edge)

        if((HasLabel(face,"h1") and HasLabel(face,"v1")) or
            (HasLabel(face,"h2") and HasLabel(face,"v2"))):
            src_c = PointLabeled(line,"start")
            dst_c = PointLabeled(line,"end")
            src_v = Location(SourceVertex(edge))
            dst_v = Location(TargetVertex(edge))
            return MatchPoints(line,src_c,dst_c,src_v,dst_v)
        else:
            src_c = PointLabeled(line,"end")
            dst_c = PointLabeled(line,"start")
            src_v = Location(SourceVertex(edge))
            dst_v = Location(TargetVertex(edge))
            return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    texture = MapToEdges(line_to_curve, grid_tex)
    ExportSVG(texture, 1000)
```



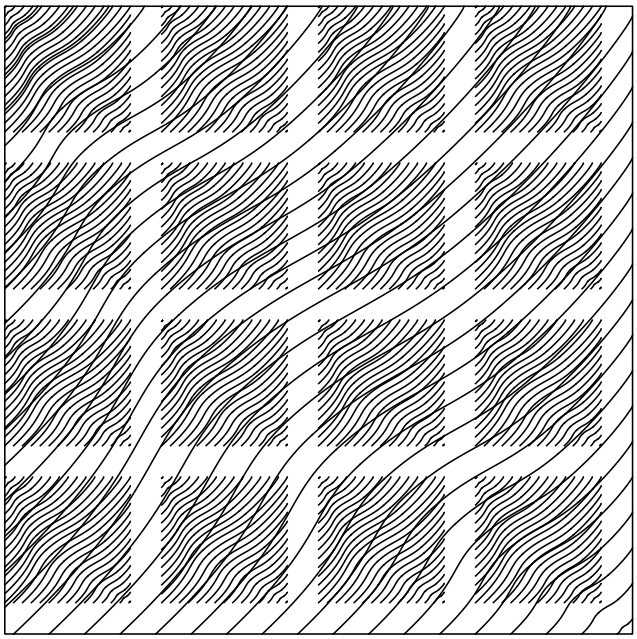
User 2 - cracks. 12 lines, four operators (green), two executions. User satisfaction: 6/10

```
def test4():
    props   = IrregularProperties(10/4000000)
    part    = RandomPartition(props,KEEP_OUTSIDE)

    def subdivideFace(face):
        angle = Random(face,0,2*pi,1)
        props2  = IrregularProperties(100/4000000)
        return RandomPartition(props2,CROP_ADD_BOUNDARY)(face)

    def scale_map(face):
        return Scale(Contour(face),0.95)

    texture = MapToFaces(subdivideFace, part)
    texture2 = MapToFaces(scale_map, texture)
    ExportSVG (texture2, 1000)
```



User 2 - waves. 29 lines, seven operators (green), three executions. User satisfaction: 7/10

```
def test5():
    lines1 = StripesProperties(0,50,200)
    lines2 = StripesProperties(pi/2,50,200)
    SetFaceLabels(lines1,"h1","h2")
    SetFaceLabels(lines2,"v1","v2")
    grid_tex = GridPartition(lines1,lines2,KEEP_OUTSIDE)

    def vagues(face):
        if((HasLabel(face,"h1") and HasLabel(face,"v1"))):

            angle = pi/4
            lines = StripesProperties(angle,10)
            return StripesPartition(lines)(face)
        else:
            return Nothing()

    def line_to_curve(edge):
        if IsBoundary(edge):
            return Nothing()

        line    = ImportSVG("data/line6.svg")
        src_c = PointLabeled(line,"start")
        dst_c = PointLabeled(line,"end")
        src_v = Location(SourceVertex(edge))
        dst_v = Location(TargetVertex(edge))
        return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    texture = MapToFaces(vagues, grid_tex)
    T1 = MapToEdges(line_to_curve, texture)

    props   = StripesProperties(pi/4,40)
    stripes = StripesPartition(props)
    T2 = MapToEdges(line_to_curve, stripes)
    textureFinale = Union(T1,T2)

    ExportSVG (textureFinale, 1000)
```

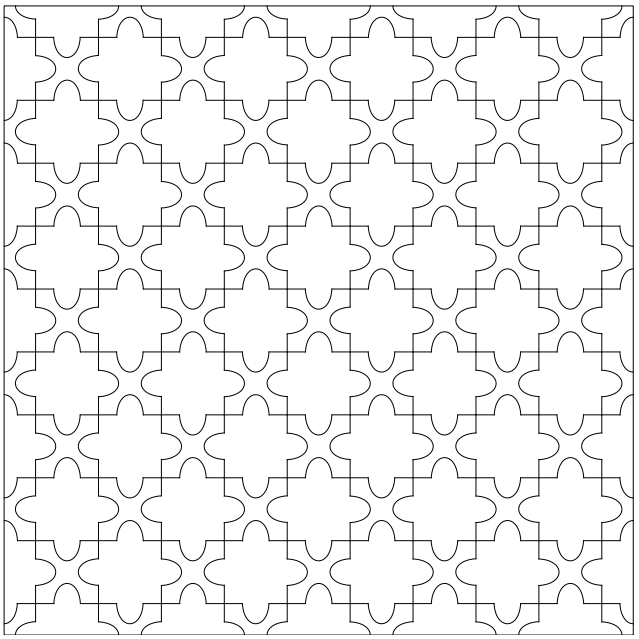


Interview of User 2

How easy was it to decide what you would do in order to reach the target designs? It was quite simple. The nodal formulation is common to other usual modeling tools. Only the labeling design was a bit harsh.
How easy was it to realize your plans by scripting in our tool? It was simple, and it becomes quicker and quicker when you start re-using bricks you designed before.
How often did you loose the understanding of what your script was doing? Never.
How did you feel about the general principle of designing textures with our partitions+mappers+combinations? I thought it was correct and logical. I liked the way of thinking textures as simple bases which are refined and then combined. It reminded me Maya's system for designing procedural textures.
What are your thoughts about what you liked or disliked while experiencing our tool? I liked the operator syntax embedded in Python, which is quite practical as a scripting language. I think the main limitation is the computational cost of the textures. I would like them to update in real-time when I achieve one piece of code. I liked the way modeling is thought in the system. You have an idea, you can easily make it real. The needed reasoning looks obvious. I also liked the quality of random textures that can be done, such as in the cracks example. Such textures are really hard to make with current commercial software.

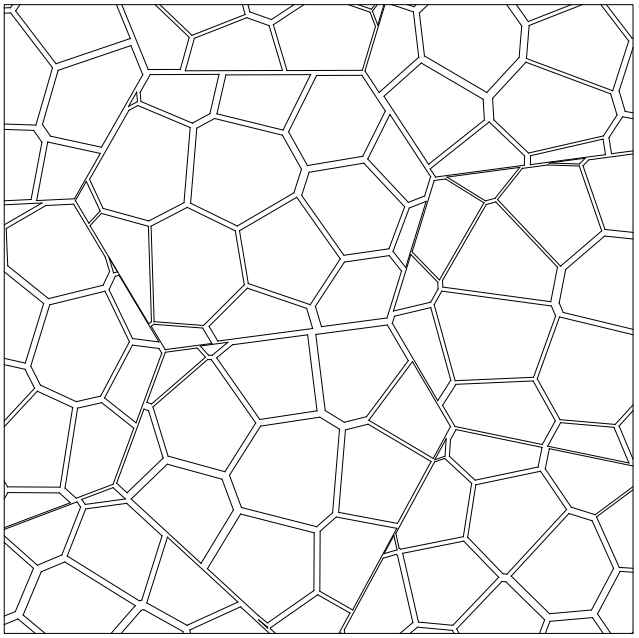
User 3 - puzzle. 20 lines, two operators (green), three executions. User satisfaction: 8/10

```
def test3():
    line = ImportSVG("data/estelle/bitonio.svg")

    def edge_to_bla(edge):
        if (HasLabel(LeftFace(edge), "v1") and HasLabel(
             LeftFace(edge), "h1") ) or (HasLabel(LeftFace(edge
             ), "v2") and HasLabel(LeftFace(edge), "h2")):
            src_c = PointLabeled(line,"start")
            dst_c = PointLabeled(line,"end")
        else :
            dst_c = PointLabeled(line,"start")
            src_c = PointLabeled(line,"end")

        src_v = Location(SourceVertex(edge))
        dst_v = Location(TargetVertex(edge))
        return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    lines1 = StripesProperties(0,200)
    lines2 = StripesProperties(pi/2,200)
    SetFaceLabels(lines1,"h1","h2")
    SetFaceLabels(lines2,"v1","v2")
    grid_tex = GridPartition(lines1,lines2,KEEP_OUTSIDE)
    part = MapToEdges(edge_to_bla, grid_tex)

    ExportSVG(part, 2000)
```



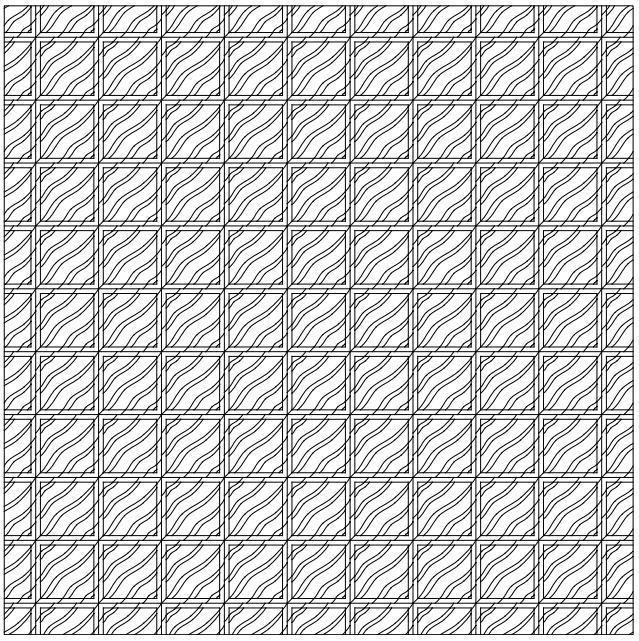
User 3 - cracks. 12 lines, four operators (green), two executions. User satisfaction: 5/10

```
def test4():
    def shrink_face(face):
        return Scale(Contour(face), Random(face, 0.90, 0.97, 1))

    def uniform_to_uniform(face):
        props2  = IrregularProperties(50/4000000)
        part2   = UniformPartition(props2,CROP_ADD_BOUNDARY)
        part2 = MapToFaces(shrink_face, part2)
        return part2(face)

    props  = IrregularProperties(5/4000000)
    part   = UniformPartition(props,KEEP_OUTSIDE)
    part = MapToFaces(uniform_to_uniform, part)

    ExportSVG(part, 2000)
```



User 3 - waves. 30 lines, nine operators (green), three executions. User satisfaction: 10/10

```
def test5():
    line    = ImportSVG("data/line6.svg")
    def line_to_curve(edge):
        if IsBoundary(edge):
            return ToCurve(edge)

        src_c = PointLabeled(line,"start")
        dst_c = PointLabeled(line,"end")
        src_v = Location(SourceVertex(edge))
        dst_v = Location(TargetVertex(edge))
        return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    def shrink_faces(face):
        return Scale(Contour(face), 0.85)

    def wavify_face(face):
        props   = StripesProperties(pi/4,50)
        stripes = StripesPartition(props)
        stripes = MapToEdges(line_to_curve, stripes)
        return stripes(face)

    props   = StripesProperties(pi/4,50)
    stripes = StripesPartition(props)
    lines1   = StripesProperties(0,200)
    lines2   = StripesProperties(pi/2.0,200)
    stripes = GridPartition(lines1,lines2,KEEP_OUTSIDE)
    stripes = MapToFaces(wavify_face, stripes)

    lines1   = StripesProperties(0,200)
    lines2   = StripesProperties(pi/2.0,200)
    grid_tex = GridPartition(lines1,lines2,KEEP_OUTSIDE)

    grid_tex = MapToFaces(shrink_faces, grid_tex)
    grid_tex = MapToFaces(wavify_face, grid_tex)

    final = Union(grid_tex, stripes)

    ExportSVG(final, 2000)
```

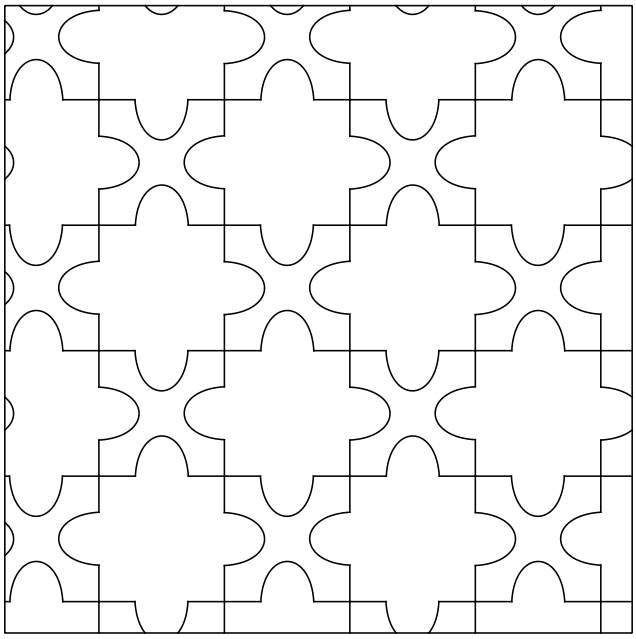


Interview of User 3

How easy was it to decide what you would do in order to reach the target designs? Quite easy. You always succeed in extracting some structure, some hierarchy in the target images. I think it would remain easy as long as there is some clear arrangement to see.
How easy was it to realize your plans by scripting in our tool? It was simple and enjoyable. I liked to have base bricks provided, that helped a lot constructing my own examples.
How often did you loose the understanding of what your script was doing? Never. I had some difficulties with the first combination involving a partition inside a mapper. After that I had no trouble.
How did you feel about the general principle of designing textures with our partitions+mappers+combinations? It works very well. I find the idea sound.
What are your thoughts about what you liked or disliked while experiencing our tool? I liked a lot that some ready-to-use bricks are provided on top of the set of operators. It helps a lot finding your way during the first moments. I found the computation a bit slow, but it is not blocking the creative process. Let say, it is not slow enough to be frustrating. One other cool point is that since you are scripting, you can use whatever text editor or IDE you want. It looks neglictible, but actually the usual sources of frustration with user interfaces are the basic shortcuts and the undo/redo feature. All of these are no problem here.

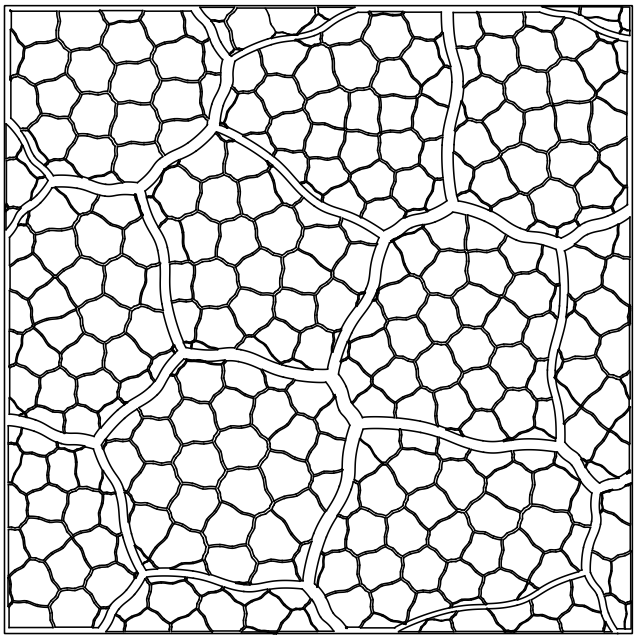
User 4 - puzzle (10min only). 20 lines, two operators (green), five executions. User satisfaction: 10/10

```
def test3():
    size = 2000
    lines1 = StripesProperties(0,200)
    lines2 = StripesProperties(pi/2,200)
    SetFaceLabels(lines1,"h1","h2")
    SetFaceLabels(lines2,"v1","v2")
    grid_tex = GridPartition(lines1,lines2,KEEP_OUTSIDE)

    bitonio = ImportSVG("data/estelle/bitonio.svg")

    def line_to_curve(edge):

        src_c = PointLabeled(bitonio,"start")
        dst_c = PointLabeled(bitonio,"end")
        src_v = Location(SourceVertex(edge))
        dst_v = Location(TargetVertex(edge))

        face = LeftFace(edge)

        if((HasLabel(face,"h1") and HasLabel(face,"v1")) or
            (HasLabel(face,"h2") and HasLabel(face,"v2"))):
            return MatchPoints(bitonio,src_c,dst_c,src_v,dst_v)
        else:
            return MatchPoints(bitonio,dst_c,src_c,src_v,dst_v)

    T1 = MapToEdges(line_to_curve, grid_tex)
    ExportSVG(T1, 1000)
```



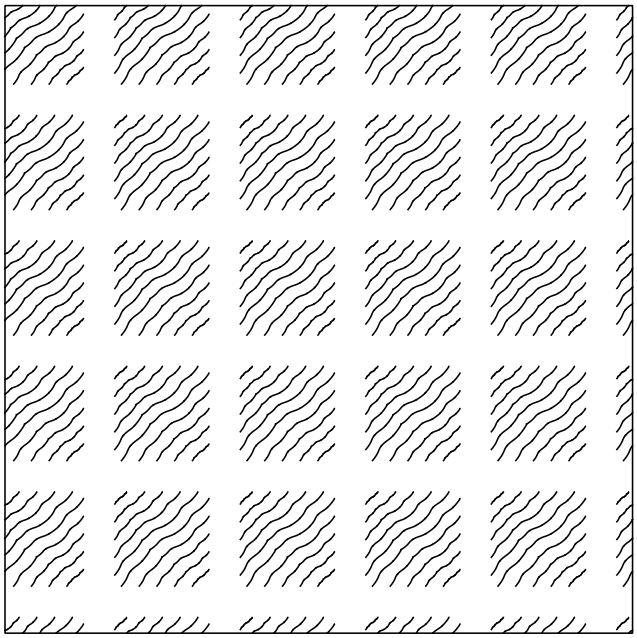
User 4 - cracks. 25 lines, seven operators (green), three executions. User satisfaction: 8/10

```
def test4():
    size = 2000
    props1  = IrregularProperties(30/(size*size))
    tex1    = UniformPartition(props1,CROP_ADD_BOUNDARY)

    def scale_map(face):
        return Scale(Contour(face),0.95)

    def part_map(face):
        s = 2000
        props  = IrregularProperties(1000/(s*s))
        return UniformPartition(props, CROP_ADD_BOUNDARY)(face)

    line    = ImportSVG("data/line6.svg")

    def line_to_curve(edge):
        if IsBoundary(edge):
            return ToCurve(edge)

        src_c = PointLabeled(line,"start")
        dst_c = PointLabeled(line,"end")
        src_v = Location(SourceVertex(edge))
        dst_v = Location(TargetVertex(edge))
        return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    T1 = MapToEdges(line_to_curve, tex1)
    T2 = MapToFaces(scale_map, T1)
    T3 = MapToFaces(part_map, T2)
    T4 = MapToEdges(line_to_curve, T3)
    T5 = MapToFaces(scale_map, T4)
    ExportSVG(T5, 1000)
```



User 4 - waves. 28 lines, six operators (green), four executions. User satisfaction: 6/10

```
def test5():
    size = 2000
    lines1 = StripesProperties(0,200)
    lines2 = StripesProperties(pi/2,200)
    SetFaceLabels(lines1,"h1","h2")
    SetFaceLabels(lines2,"v1","v2")
    grid_tex = GridPartition(lines1,lines2,KEEP_OUTSIDE)

    def scale_map(face):
        return Scale(Contour(face),0.75)

    def hatch_map(face):
        angle = pi/4.
        size = Random(face,0,5)
        lines = StripesProperties(angle,20)
        return StripesPartition(lines)(face)

    line    = ImportSVG("data/line6.svg")

    def line_to_curve(edge):
        if IsBoundary(edge):
            return Nothing()

        src_c = PointLabeled(line,"start")
        dst_c = PointLabeled(line,"end")
        src_v = Location(SourceVertex(edge))
        dst_v = Location(TargetVertex(edge))
        return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    T1 = MapToFaces(scale_map, grid_tex)
    T2 = MapToFaces(hatch_map, T1)
    T3 = MapToEdges(line_to_curve, T2)
    T4 = MapToEdges(line_to_curve, T3)
    ExportSVG(T4, 1000)
```

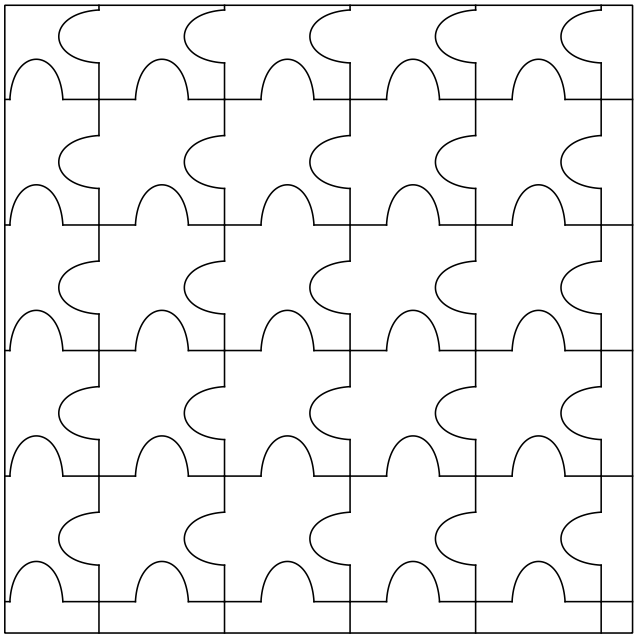


Interview of User 4

How easy was it to decide what you would do in order to reach the target designs? Quite easy. You only have to look at how to decompose the image, and then how to do each part with mappers. These are a lot like shaders, which is very practical.
How easy was it to realize your plans by scripting in our tool? It is quite easy to put the ideas into practice - I already coded with Python. I had a few difficulties only with the parameters of the Random operator and the density in RandomPartition and UniformPartition. Besides, it is possible to design the textures iteratively, which is super practical when you are editing code. Also the mapper/mapping operator design is very compact, which made my life a lot simpler.
How often did you loose the understanding of what your script was doing? Never.
How did you feel about the general principle of designing textures with our partitions+mappers+combinations? The concept is really sound. It allows to create complex results very easily, step by step. In particular I found that the way of editing topology in mappers is very powerful.
What are your thoughts about what you liked or disliked while experiencing our tool? Among all things I appreciated, I think the mappers were my favourite part. Maybe the thing I disliked was the computation time which was a bit heavy.

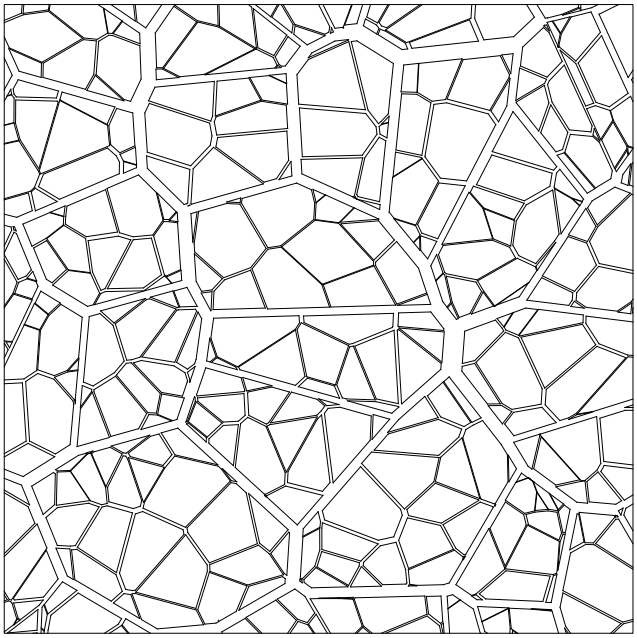
User 5 - puzzle. 23 lines, two operators (green), four executions. User satisfaction: 6/10

```
def test3():
  lines1 = StripesProperties(0,200)
  lines2 = StripesProperties(pi/2,200)
  SetEdgeLabels(lines1,"h1","h2")
  SetEdgeLabels(lines2,"v1","v2")
  grid_tex = GridPartition(lines1,lines2,KEEP_OUTSIDE)
  line = ImportSVG("data/estelle/bitonio.svg")

  # element to edge
  def line_to_curve(edge):
    if IsBoundary(edge):
      return Nothing()

    src_c = PointLabeled(line,"start")
    dst_c = PointLabeled(line,"end")
    src_v = Location(SourceVertex(edge))
    dst_v = Location(TargetVertex(edge))

    if (HasLabel(IncidentEdges(TargetVertex(edge)), "h2")) or (
        HasLabel(IncidentEdges(TargetVertex(edge)), "v2")) and
         (HasLabel(edge, "v2") or HasLabel(edge, "h1")):
      return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    elif (HasLabel(IncidentEdges(TargetVertex(edge)), "h1")) or
         (HasLabel(IncidentEdges(TargetVertex(edge)), "v1"))
        and (HasLabel(edge, "v1") or HasLabel(edge, "h2")):
      return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    else:
      return MatchPoints(line, src_c, dst_c, dst_v, src_v)

  T = MapToEdges(line_to_curve, grid_tex)
  ExportSVG(T, 1000)
```



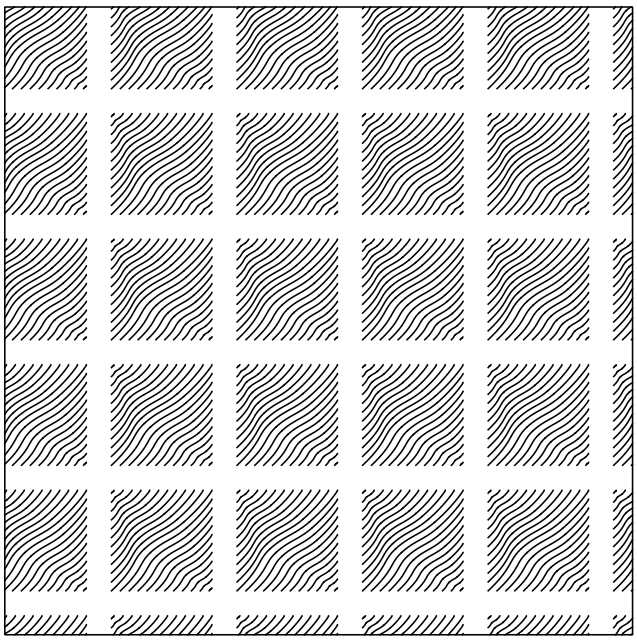
User 5 - cracks. 14 lines, five operators (green), two executions. User satisfaction: 8/10

```
def test4():
  size = 2000
  props    = IrregularProperties(30/(size*size))
  init_tex = RandomPartition(props,KEEP_OUTSIDE)

  def face_to_stripes(face):
    props    = IrregularProperties(150/(size*size))
    init_tex = RandomPartition(props,CROP_ADD_BOUNDARY)
    return init_tex(face)

  def scale_map(face):
    return Scale(Contour(face),Random(face, 0.9, 0.99,0))

  T = MapToFaces(scale_map,init_tex)
  T2 = MapToFaces(face_to_stripes,T)
  T3 = MapToFaces(scale_map, T2)

  Add(T3(StartDomain(2000)))
```



User 5 - waves. 24 lines, six operators (green), three executions. User satisfaction: 9/10

```
def test5():
  lines1 = StripesProperties(0,200)
  lines2 = StripesProperties(pi/2,200)
  grid_tex = GridPartition(lines1,lines2,KEEP_OUTSIDE)

  props   = StripesProperties(pi/4,10)
  stripes = StripesPartition(props)
  line    = ImportSVG("data/line6.svg")

  def scale_map(face):
    return Scale(Contour(face),0.8)

  def face_to_stripes(face):
    lines = StripesProperties(pi/4,10)
    tex_stripes = MapToEdges(line_to_curve, StripesPartition(
        lines))
    return tex_stripes(face)

  def line_to_curve(edge):
    if IsBoundary(edge):
      return Nothing()

    src_c = PointLabeled(line,"start")
    dst_c = PointLabeled(line,"end")
    src_v = Location(SourceVertex(edge))
    dst_v = Location(TargetVertex(edge))
    return MatchPoints(line,src_c,dst_c,src_v,dst_v)

  T2 = MapToFaces(scale_map, grid_tex)
  T3 = MapToFaces(face_to_stripes, T2)

  ExportSVG(T3, 1000)
```

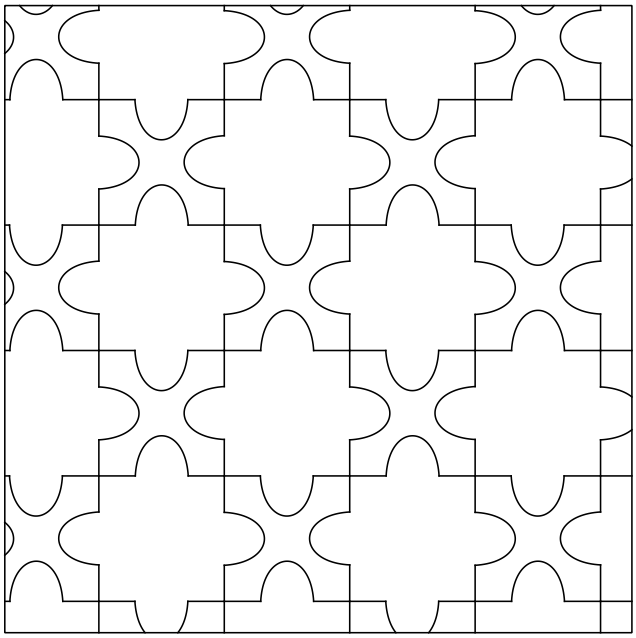


Interview of User 5

How easy was it to decide what you would do in order to reach the target designs? Easy, natural. As long as there is a structure to observe, it takes just an instant. It became a little harder for me for random patterns (cracks).
How easy was it to realize your plans by scripting in our tool? Surprisingly easy. In particular, I got exactly what I expected for the waves although I coded everything in a single strike.
How often did you loose the understanding of what your script was doing? Once with labels in the puzzle. Then never again.
How did you feel about the general principle of designing textures with our partitions+mappers+combinations? At least it was appropriate for the given examples, which is already something given the broad variety of targets. The principle is easy to understand and easy to visualize. It is easy to imagine the resulting texture of a given composition of operators.
What are your thoughts about what you liked or disliked while experiencing our tool? It was a nice experience. I liked that the operators fit well in a node-based strategy, which could be used for making an even more convenient interface.

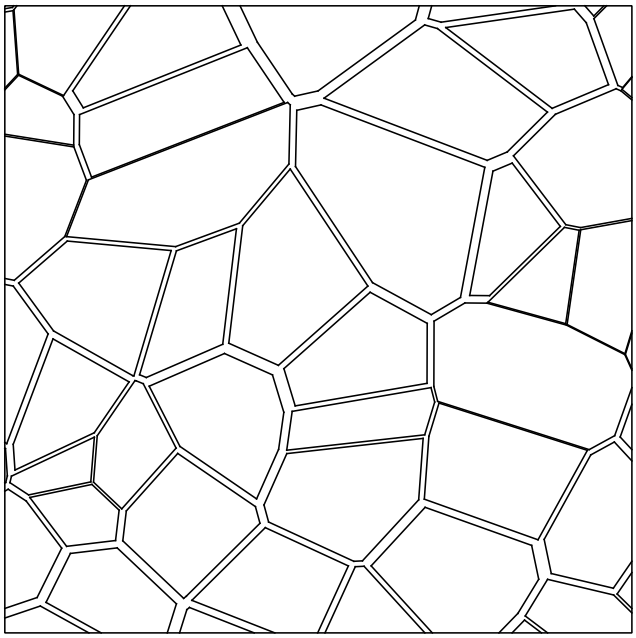
User 6 - puzzle. 20 lines, two operators (green), three executions. User satisfaction: 10/10

```
def test4():

  lines1 = StripesProperties(0,200)
  lines2 = StripesProperties(pi/2,200)
  SetEdgeLabels(lines1,"h1","h2")
  SetEdgeLabels(lines2,"v1","v2")
  grid_tex = GridPartition(lines1,lines2,KEEP_OUTSIDE)

  bitonio = ImportSVG("data/estelle/bitonio.svg")

  def line_to_curve(edge):
    if IsBoundary(edge):
      return Nothing()

    src_c = PointLabeled(bitonio,"start")
    dst_c = PointLabeled(bitonio,"end")
    src_v = Location(SourceVertex(edge))
    dst_v = Location(TargetVertex(edge))
    if((HasLabel(edge,"v1") and HasLabel(IncidentEdges(
        TargetVertex(edge)), "h1")) or (HasLabel(edge,"h1")
        and HasLabel(IncidentEdges(SourceVertex(edge)), "v1"))
         or (HasLabel(edge,"v2") and HasLabel(IncidentEdges(
        TargetVertex(edge)), "h2")) or (HasLabel(edge,"h2")
        and HasLabel(IncidentEdges(SourceVertex(edge)), "v2"))
        ):
      dst_v = Location(SourceVertex(edge))
      src_v = Location(TargetVertex(edge))
    return MatchPoints(bitonio,src_c,dst_c,src_v,dst_v)

  T = MapToEdges(line_to_curve, grid_tex)

  ExportSVG(T, 1000)
```



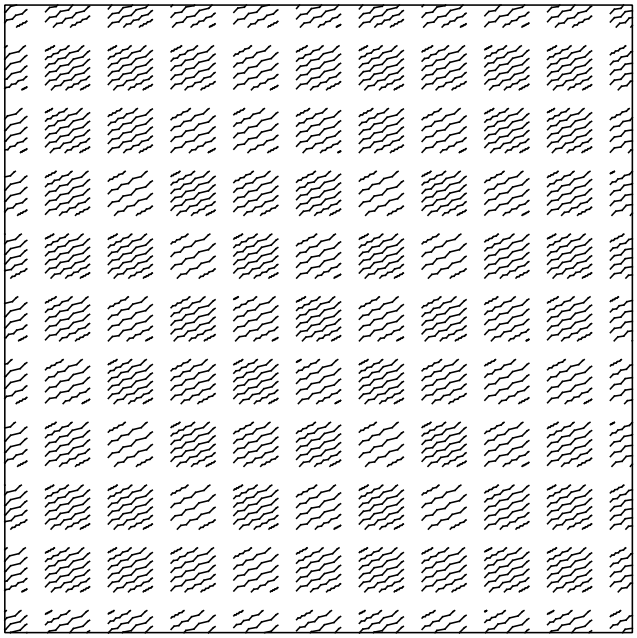
User 6 - cracks. Nine lines, two operators (green), two executions. User satisfaction: 8/10

```
def test5():
  size = 1000
  jensaisrien = 0.999999
  props    = IrregularProperties(30/(size*size))
  init_tex = RandomPartition(props,KEEP_OUTSIDE)

  def scale_map(face):
    return Scale(Contour(face),Random(face, 0.9, jensaisrien,
        1))

  T = MapToFaces(scale_map, init_tex)
  ExportSVG(T, 1000)
```



User 6 - waves. 25 lines, five operators (green), four executions. User satisfaction: 7/10

```
def test6():
  theta = 0
  width = 100
  lines1   = StripesProperties(theta,width)
  lines2   = StripesProperties(theta+pi/2.0,width)
  grid_tex = GridPartition(lines1,lines2,KEEP_OUTSIDE)

  def scale_map(face):
    return Scale(Contour(face),0.7)

  T = MapToFaces(scale_map,grid_tex)

  def face_to_stripes(face):
    width2 = BBoxWidth(face)/Random(face,4,6,0)
    lines = StripesProperties(pi/6.0,width2)
    return StripesPartition(lines)(face)
  T2 = MapToFaces(face_to_stripes, T)

  line    = ImportSVG("data/line4.svg")

  def line_to_curve(edge):
    if IsBoundary(edge):
      return Nothing()

    src_c = PointLabeled(line,"start")
    dst_c = PointLabeled(line,"end")
    src_v = Location(SourceVertex(edge))
    dst_v = Location(TargetVertex(edge))
    return MatchPoints(line,src_c,dst_c,src_v,dst_v)

  T3 = MapToEdges(line_to_curve, T2)

  ExportSVG(T3, 1000)
```

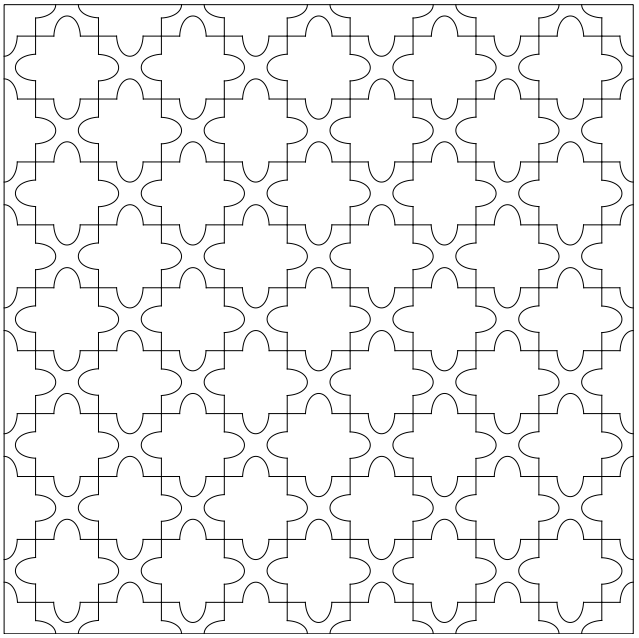


Interview of User 6

How easy was it to decide what you would do in order to reach the target designs? Easy. As a structure-finding task, it was easy.
How easy was it to realize your plans by scripting in our tool? It was very easy as well even though I never coded using Python before.
How often did you loose the understanding of what your script was doing? Never.
How did you feel about the general principle of designing textures with our partitions+mappers+combinations? It looked natural and sound.
What are your thoughts about what you liked or disliked while experiencing our tool? I liked that the model is intuitive.

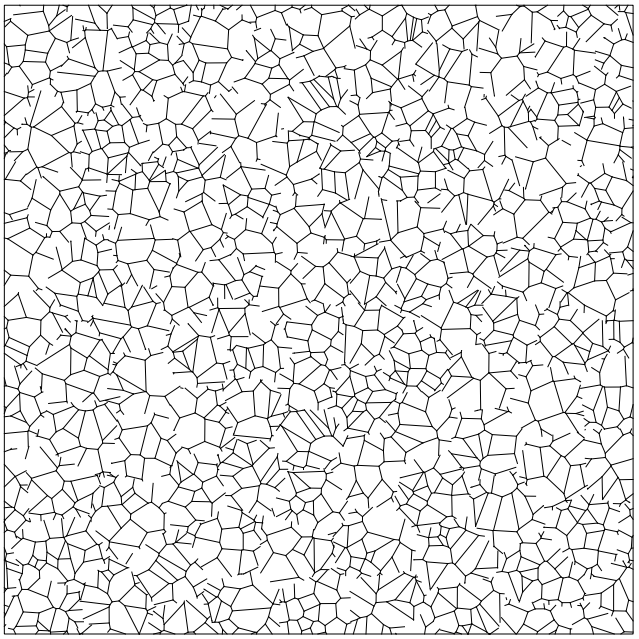
User 7 - puzzle. 25 lines, two operators (green), two executions. User satisfaction: 10/10

```
def test3():
    theta = 0
    width = 200
    lines1    = StripesProperties(theta,width)
    lines2    = StripesProperties(theta+pi/2.0,width)
    SetFaceLabels(lines1,"h1","h2")
    SetFaceLabels(lines2,"v1","v2")
    grid_tex = GridPartition(lines1,lines2,KEEP_OUTSIDE)

    line = ImportSVG("data/estelle/bitonio.svg")

    def face_to_stripes2(edge):
        face_left = LeftFace(edge)

        sens = True

        if((HasLabel(face_left,"h1") and HasLabel(face_left,"v1
            ")) or
           (HasLabel(face_left,"h2") and HasLabel(face_left,"v2
               "))):
            sens = False

        src_c = PointLabeled(line,"start")
        dst_c = PointLabeled(line,"end")
        src_v = Location(SourceVertex(edge))
        dst_v = Location(TargetVertex(edge))
        if sens:
            return MatchPoints(line,src_c,dst_c,src_v,dst_v)
        else:
            return MatchPoints(line,src_c,dst_c,dst_v,src_v)

    tex2 = MapToEdges(face_to_stripes2,grid_tex)
    ExportSVG(tex2,2000)
```



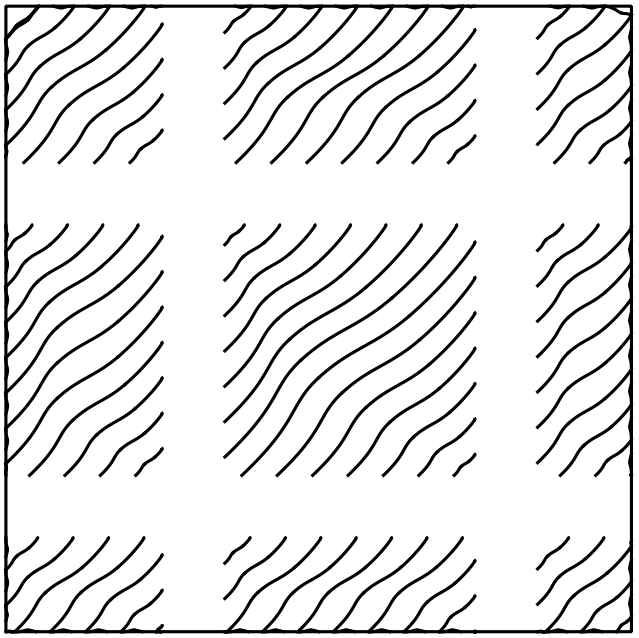
User 7 - cracks. Ten lines, three operators (green), two executions. User satisfaction: 7/10

```
def test4():
    size = 2000
    props1  = IrregularProperties(100/(size*size))
    tex1    = UniformPartition(props1,KEEP_OUTSIDE)

    def hatch_map(face):
        angle = Random(face,0,2*pi,1)
        props2  = IrregularProperties(1000/(size*size))
        return RandomPartition(props2,CROP)(face)

    tex3 = MapToFaces(hatch_map,tex1)
    ExportSVG(tex3,2000)
```



User 7 - waves. 25 lines, five operators (green), four executions. User satisfaction: 6/10

```
def test5():
    line    = ImportSVG("data/line6.svg")
    theta = 0
    width = 200

    lines1   = StripesProperties(theta,width, width/4)
    lines2   = StripesProperties(theta+pi/2.0,width, width/4)
    SetFaceLabels(lines1,"h1","h2")
    SetFaceLabels(lines2,"v1","v2")
    grid_tex1 = GridPartition(lines1,lines2,KEEP_OUTSIDE)

    def clean(face):
        if((HasLabel(face,"h1") or HasLabel(face,"v1"))):
            return Nothing()
        return Contour(face)

    grid_tex1_clean = MapToFaces(clean,grid_tex1)

    props   = StripesProperties(pi/4,width/10)
    stripes = StripesPartition(props)

    grid_tex3 = Inside(stripes,grid_tex1_clean,CROP)

    def line_to_curve(edge):

        src_c = PointLabeled(line,"start")
        dst_c = PointLabeled(line,"end")
        src_v = Location(SourceVertex(edge))
        dst_v = Location(TargetVertex(edge))
        return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    tex3 = MapToEdges(line_to_curve,grid_tex3)
    ExportSVG(tex3,500)
```

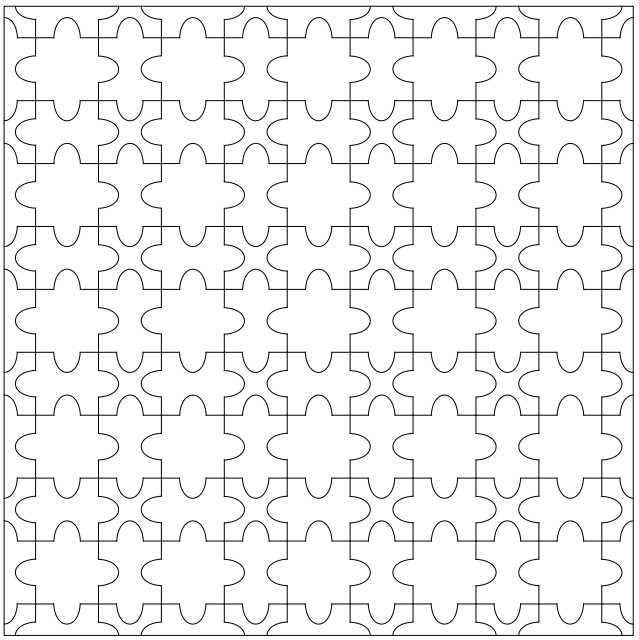


Interview of User 7

How easy was it to decide what you would do in order to reach the target designs? It was quite easy to set up a plan to reach the target designs. Sometimes you even see several ways to achieve the goal. I liked this because I find it always more safe to think before doing rather than dive into a dead end.
How easy was it to realize your plans by scripting in our tool? Quite easy. Maybe the only difficulty I had was the label filtering.
How often did you loose the understanding of what your script was doing? Never.
How did you feel about the general principle of designing textures with our partitions+mappers+combinations? It looks very natural. Actually I cannot think of any other way to model textures now that I learnt this model. I especially liked the intuitive way of stacking texture levels so as to obtain "multi-level" patterns.
What are your thoughts about what you liked or disliked while experiencing our tool? The tool was enjoyable. In particular there are few lines of code to manage even for complex textures. The computation time is a bit long, but nothing that cannot be optimized I believe. I think this approach has some big potential to be combined with a node-based GUI.

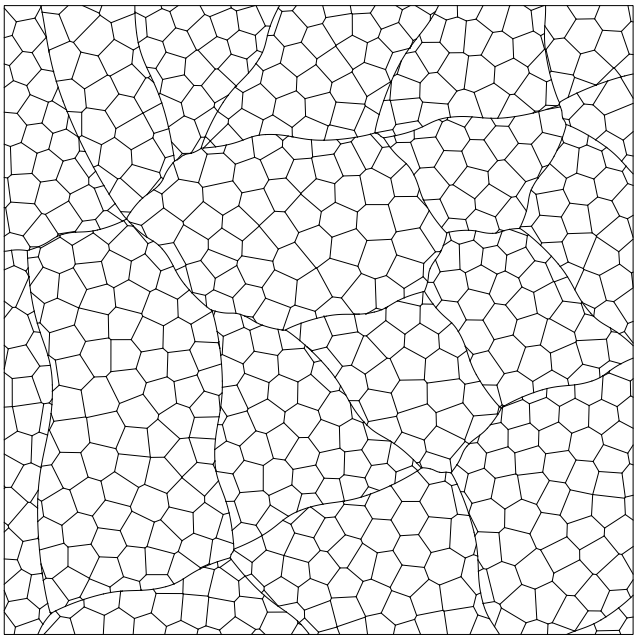
User 8 - puzzle. 21 lines, two operators (green), two executions. User satisfaction: 9/10

```
def test3():
    line = ImportSVG("data/estelle/bitonio.svg")
    lines1 = StripesProperties(0,200)
    lines2 = StripesProperties(pi/2,200)
    SetEdgeLabels(lines1,"h1","h2")
    SetEdgeLabels(lines2,"v1","v2")
    grid_tex = GridPartition(lines1,lines2,KEEP_OUTSIDE)

    def line_to_curve(edge):
        if IsBoundary(edge):
            return Nothing()
        src_c = PointLabeled(line,"start")
        dst_c = PointLabeled(line,"end")
        src_v = Location(SourceVertex(edge))
        dst_v = Location(TargetVertex(edge))
        if(HasLabel(edge,"h2")):
            return MatchPoints(line,src_c,dst_c,dst_v,src_v)
        if(HasLabel(edge,"v2")):
            return MatchPoints(line,src_c,dst_c,dst_v,src_v)
        return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    texE = MapToEdges(line_to_curve, grid_tex)
    ExportSVG(texE, 2000)
```



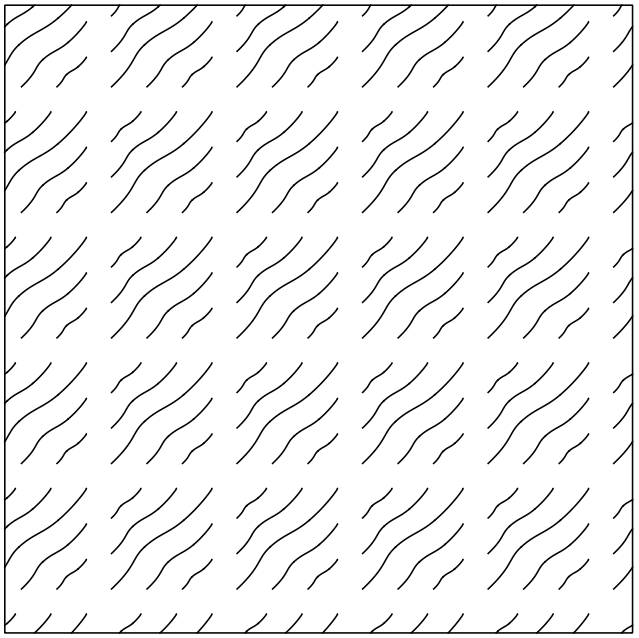
User 8 - cracks. 21 lines, four operators (green), three executions. User satisfaction: 10/10

```
def test4():
    size = 2000
    props2  = IrregularProperties(10/(size*size))
    tex2 = RandomPartition(props2,KEEP_OUTSIDE)
    line = ImportSVG("data/line6.svg")

    def line_to_curve(edge):
        if IsBoundary(edge):
            return Nothing()
        src_c = PointLabeled(line,"start")
        dst_c = PointLabeled(line,"end")
        src_v = Location(SourceVertex(edge))
        dst_v = Location(TargetVertex(edge))
        return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    texE = MapToEdges(line_to_curve, tex2)

    def fill_map(face):
        density=500/(size*size)
        irr_props = IrregularProperties(density)
        unif_part = UniformPartition(irr_props,
            CROP_ADD_BOUNDARY)
        return unif_part(face)

    texF = MapToFaces(fill_map, texE)

    ExportSVG(texF, 2000)
```



User 8 - waves. 26 lines, five operators (green), five executions. User satisfaction: 9/10

```
def test5():
    theta = 0.0
    width = 200

    lines1   = StripesProperties(theta,width)
    lines2   = StripesProperties(theta+pi/2.0,width)

    grid_tex0 = GridPartition(lines1,lines2,KEEP_OUTSIDE)

    def scale_map(face):
        return Scale(Contour(face),0.8)

    grid_tex = MapToFaces(scale_map, grid_tex0)

    line = ImportSVG("data/line6.svg")
    def line_to_curve(edge):
        if IsBoundary(edge):
            return Nothing()

        src_c = PointLabeled(line,"start")
        dst_c = PointLabeled(line,"end")
        src_v = Location(SourceVertex(edge))
        dst_v = Location(TargetVertex(edge))
        return MatchPoints(line,src_c,dst_c,src_v,dst_v)

    def hatch_map(face):
        angle = pi/4
        lines = StripesProperties(angle,40)
        part = StripesPartition(lines)
        myTex = MapToEdges(line_to_curve, part)
        return myTex(face)

    hatch_tex = MapToFaces(hatch_map, grid_tex)

    ExportSVG(hatch_tex, 1000)
```



Interview of User 8

How easy was it to decide what you would do in order to reach the target designs? Very easy. I guess it is partly because I already know some node-based systems such as Blender's Cycles rendering engine.
How easy was it to realize your plans by scripting in our tool? Relatively easy. Trial-and-error is a good option which is allowed by the tool and that made my life easier for the most complex cases. It would be even easier if the computation time was a bit shorter.
How often did you loose the understanding of what your script was doing? Never. It takes a bit of time to learn each feature of the model, but once you have done it one time, it is very easy.
How did you feel about the general principle of designing textures with our partitions+mappers+combinations? You have to handle the learning curve, but then you have a clear and strong base for referring yourself to.
What are your thoughts about what you liked or disliked while experiencing our tool? I am very happy of the textures I was able to create during this first trial of the tool. I did not expect to reach such complex results, that was a good surprise.