FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Diplomarbeitspräsentation

Irrational Image Generator

Master- /Diplomstudium:
Medieninformatik

Simon Parzer

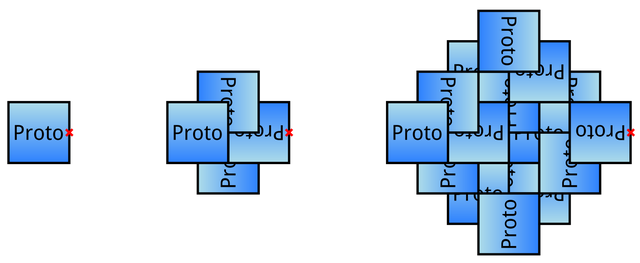
Technische Universität Wien
Institut für Computergraphik und Algoithmen
Arbeitsbereich: Computergraphik
Betreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Dipl.-Ing. Christoph Traxler
Kurt Hofstetter

1. Problem Statement

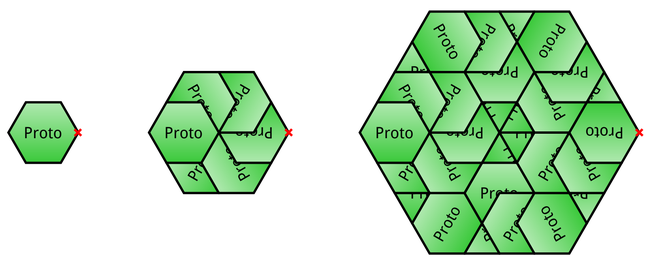
Inductive Rotation (IR), invented by artist Hofstetter Kurt, is an approach for creating seemingly nonperiodic patterns on the 2D plane by recursive translations and rotations of a single prototile. The point of the thesis is to create a tool that automates the generation process to a degree where the user only has to insert a prototile and can instantly see the patterns resulting from the IR approach.

IR has never been addressed in scientific publications before, so a formal proof of the nonperiodicity of the generated patterns is still missing. Future work in the field of geometry will be needed to formally define the properties of the generated patterns.
Currently, there are three variants of the IR method:

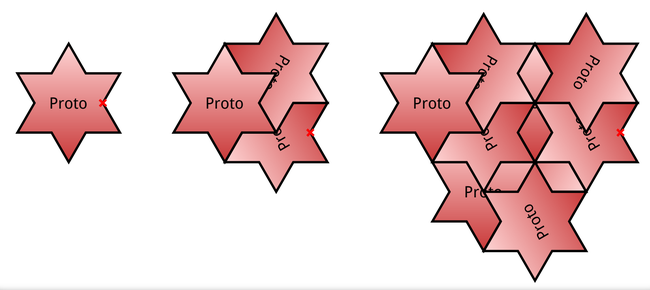
3-way Inductive Rotation
90°



5-way Inductive Rotation
60°



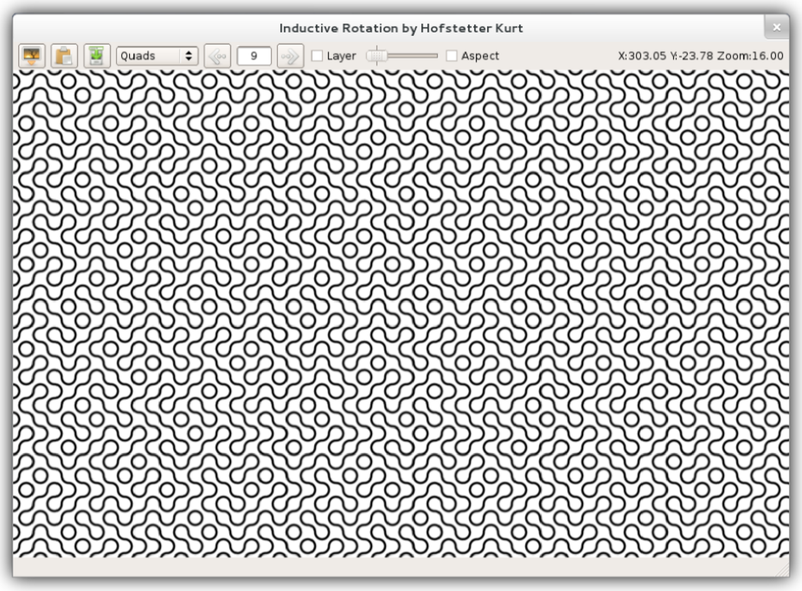
2-way Inductive Rotation
120°



2. Methodology/Development Process

An iterative development process with a user-centered approach was chosen. At the end of each iteration, a prototype showing new functionality was evaluated together with Hofstetter Kurt, and results were used to improve upon subsequent iterations.

All programs are implemented in C++ and use OpenGL for rendering. The final implementation uses the QT4 library to display GUI elements and increase portability.



3. Implementation

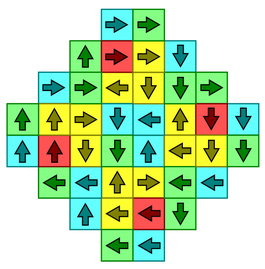
The iterative development process has led to two different implementations that are both included in the final program.

The **grid-based generator** splits the prototile into sub-tiles that can be aligned on a regular grid. The type and rotation of each sub-tile is encoded as an integer value. The regular grid is represented by a two-dimensional array of integers internally.

The **sprite-based generator** treats the prototile as a textured rectangle. Each rectangle is defined by the x,y coordinates of its center point and its rotation. The data representation is a list of such rectangles sorted by their z values.

Each implementation generates a list of textured polygons that are then passed to OpenGL for rendering.

prototile



pattern for iteration 2

(0,-3)

-1	-1	-1	6	4	-1	-1	-1
-1	-1	0	7	5	10	-1	-1
(0,0) -1	6	4	-1	9	8	4	-1
0	1	5	10	14	1	11	10
2	3	9	8	2	13	9	8
-1	12	14	-1	5	12	14	-1
-1	-1	2	13	15	8	-1	-1
-1	-1	-1	12	14	-1	-1	-1

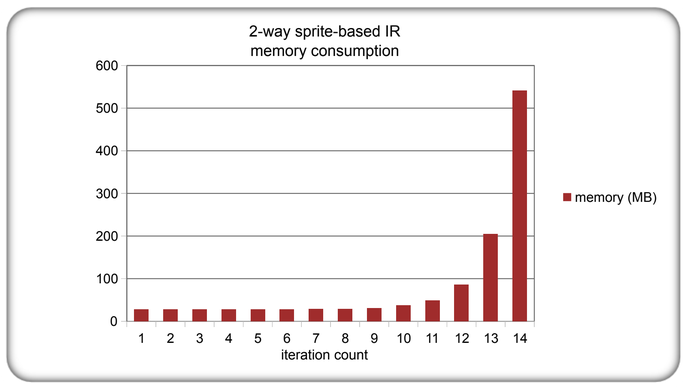
grid-based data

```
( 1.0, 1.0)   0
( 2.0, 0.0)  90
( 3.0, 1.0) 180
( 2.0, 2.0) 270
( 4.0,-2.0)  90
( 5.0,-1.0) 180
( 4.0, 0.0) 270
( 3.0,-1.0)   0
( 7.0, 1.0) 180
( 6.0, 2.0) 270
( 5.0, 1.0)   0
( 6.0, 0.0)  90
( 4.0, 4.0) 270
( 3.0, 3.0)   0
( 4.0, 2.0)  90
( 5.0, 3.0) 180
```

sprite-based data

4. Results

The maximum iteration count is limited by memory and run-time constraints.
As it turned out, both run-time and memory usage grow exponentially with the number of iterations.
Nonetheless, the program is able to generate patterns large enough for the use cases it was designed for. Since the generation algorithms work on geometry instead of bitmap data, it is possible to switch between different prototile graphics without having to re-calculate the IR pattern.



contact: simon.parzer@gmail.com