Ring's Anatomy - Parametric Design of Wedding Rings

René Berndt, Volker Settgast, Eva Eggeling
Fraunhofer Austria Research GmbH
Visual Computing
Graz, Austria
Email: {rene.berndt, volker.settgast, eva.eggeling}
@fraunhofer.at

Christoph Schinko, Ulrich Krispel, Sven Havemann
Institute of Computer Graphics and Knowledge Visualization
Graz University of Technology
Graz, Austria
Email: {c.schinko, u.krispel, s.havemann} @cgv.tugraz.at

Dieter W. Fellner
Fraunhofer IGD and TU Darmstadt
Darmstadt, Germany
Email: d.fellner@igd.fraunhofer.de

***Abstract*—We present a use case that demonstrates the effectiveness of procedural shape modeling for mass customization of consumer products. We show a metadesign that is composed of a few well-defined procedural shape building blocks. It can generate a large variety of shapes and covers most of a design space defined by a collection of exemplars, in our case wedding rings. We describe the process of model abstraction for the shape space spanned by these shapes, arguing that the same is possible for other shape design spaces as well.**

***Keywords*-metadesign; procedural content; generative modeling; wedding rings; mass customization; GML.**

I. Introduction

The wedding day is one of the most important days in most peoples lifes. Many couples dignify this event by choosing a unique individual wedding ring design. This desire is met by a large number of wedding ring workshops. A professional jeweler guides through the process of designing, fabricating, and finishing a pair of individual unique wedding rings. Wedding rings are therefore an ideal case for industrial mass customization.

The great advantage of modern CNC machinery (*Computerized Numerical Control*) is that machine parameters can be varied with every single produced item. This is the basis for mass customization. A paradigm shift has to take place because industrial designers have to create no longer only a static product (a *design*), but a whole product family (a *metadesign*). By specifying different parameter sets, an infinite variety of unique design instances can be obtained from a single metadesign.

A great obstacle, however, is the cost of design verification. Every produced item must be checked for its functionality, durability, manufacturability, aesthetics, and production cost. The parameter space of a metadesign must therefore be defined very carefully, and all values must be suitably limited to valid ranges. Creating a metadesign can become very intricate and involved with complex products.

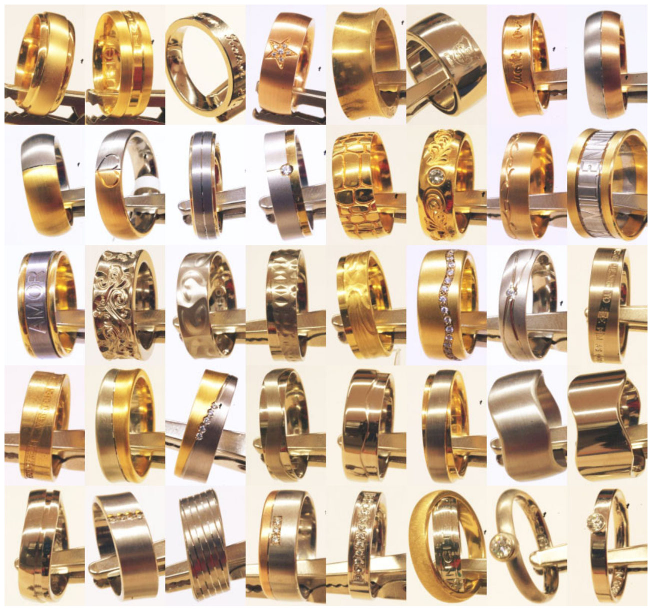


Figure 1. The challenge: Samples from the JohannKaiser wedding ring design space. This is the input for the creation of a *metadesign*, which is both an abstraction and a generalization of the given individual designs.

As good practice, it has proven very useful to proceed from a design population to a metadesign. In this case, the challenge is whether it is actually possible to find a suitably general metadesign. Fig. 1 shows the input of the creative process that is described in detail in Section IV, after the analysis of the design space described in Section III. Sections V and VI describe design refinements, and Section VII presents some results. But first, we take a look at some of the related work.

II. Related Work

Various ways exist for realizing a metadesign for wedding rings: There is specialized jewelry and ring design software

```
(0,0,-2)  (1,1,0)  2 quad
/cyan setcurrentmaterial 5 poly2doubleface
(0,1,1)  extrude
(0,0,1)  (1,0,1)  normalize 0 project_ringplane
(2,0,0)  (0,1,-1)  2 quad
/yellow setcurrentmaterial 5 poly2doubleface
0 bridgerings
```

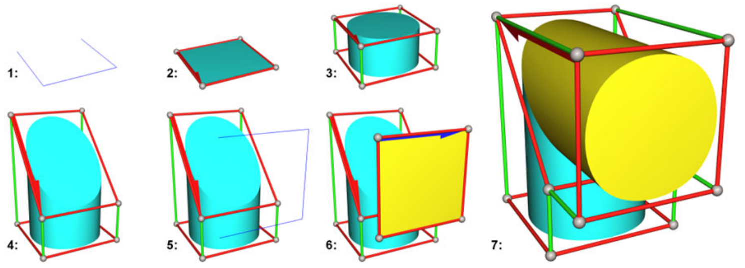


Figure 2. Procedural mesh modeling with GML. The quad operator (1) expects two points and a mode integer on the stack and produces an array of 3D points. It is converted to a double-sided mesh face (2) one side of which is extruded (3). Two faces, represented by half-edges, can be connected using bridgerings (7). Faces with smooth (green) edges are rendered as subdivision surfaces.

[1] [2], but also more flexible solutions like scriptable CAD tools or shape programming languages.

A. Jewelry Design

Traditional jewelry design is an art with a history of thousands of years. Only recently, the tools have changed. One way to realize a metadesign today is as an executable computer program. This may run very efficiently, but it lacks flexibility as the ring design is hard-coded.

The jewelry modeler from Singh et al. [3] is based on parametric voxels to allow designing carved bangles. Carved jewelry consists of copies of small parametric cavities with user-defined parameters. Various repetitive designs can be obtained this way. The sophisticated system from Wannarumon [4] uses iterated function systems (IFS) fractals. It incorporates case-based reasoning methods and fuzzy logic to increase the efficiency of mimicking existing art forms. The system provides the option to control the cost of jewelry produced using CNC or rapid prototyping machines.

JewelCAD [1] is a specialized 3D freeform modeler with modeling tools that allow much freedom in creating artistic and stylish designs. It facilitates the import of profile paths from 2D sketches for path extrusions defining patterns and lattices. However, it is not for procedural path generation.

iRing3D [2] is an iPhone app for end users with a simple but powerful interactive touch-interface for ring design. However, the basic structure, and thus, the metadesign, is pre-defined, as rings are edited on a per-segment basis.

B. Procedural Modeling

iRing3D is based on an underlying engine for parametric shape design, ParaCloud GEM [5]. A parametric engine greatly facilitates the realization of metadesigns, as it provides the functionality for geometry generation, typically accessible through a scripting language. In fact there are many options for creating shape procedurally. Examples include Lindenmayer or L-systems for describing plants [6], shape grammars for buildings [7], or by visual editing of geometry-generating dataflow graphs [8]. An exhaustive treatment is beyond the scope of this paper. Instead, we describe two representative approaches with a large user base, namely *Processing* and *Rhino Grasshopper*.

Processing started as a simplistic Java-based software sketchbook to teach students the fundamentals of computer programming [9]. With its comprehensive library of visual elements it quickly turned into a tool for visual artists. A great number of interactive art installations is realized today with Processing. But, the approach reaches its limits with mesh modeling, since scripting mesh operations can quickly become difficult to handle.

A complementary approach is taken by Grasshopper [10]. It is a plugin that adds parametric modeling capabilities to a well established interactive NURBS modeler, Rhinoceros. Rhino's mesh modeling operations can be wired together by interconnecting components on a 2D Grasshopper canvas, which requires no programming skills. This facilitates parametric mesh modeling up to a certain level of complexity. But even though defining a desired ring metadesign using Grasshopper is feasible, it is not possible to deploy it as a stand-alone software package because it requires Rhino. A simpler solution is needed.

C. GML, the Generative Modeling Language

Our choice for encoding the metadesign of wedding rings is GML, a simple stack-based shape programming language [11]. GML is syntactically similar to PostScript, but it provides mesh modeling rather than 2D typesetting operators. A GML program is basically a stream of tokens that are either data (which are put on a stack) or processing instructions (which are executed). The example in Fig. 2 shows how easily profiles (point arrays) can be turned into mesh faces that are then connected using 'bridges'. Mesh modeling operations in GML are based on halfedges; a halfedge unambiguously denotes one vertex, one face, and one edge of a mesh. The example from Fig. 2 can easily be extended to obtain a closed ring, provided all profiles contain the same number of points.

GML programs are typically developed in a very interactive fashion using the GMLStudio IDE available from the GML website [12]. The GML engine is available as ActiveX control, which makes it particularly easy to embed it into other applications.

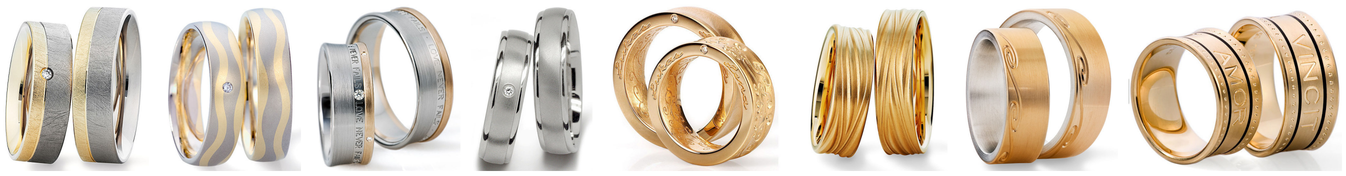


Figure 3. Rings with different materials, surface characteristics, ornamental patterns and artistic engravings.

III. Ring Features

This section presents background knowledge about the domain, i.e., the characteristic elements of a wedding ring.

A. Profile

The profile is the most characteristic feature defining the shape of the ring. The classic profiles are:

- **Flat section profile** is the traditional wedding ring profile, which is flat on the inside and on the outside.
- **D shaped profile** rings have a flat profile on the inside and a heavily bent (half-circle) profile on the outside.
- **Halo profile** rings have a perfectly round cross section.
- **Oxfort court** is an oval profile with flat rounded internal and external facets.

B. Material

Another characteristic property of a wedding ring is its material. Various noble metals, e.g., rose gold, white gold, silver, platinum, but also stainless steel, are typically used used for a wedding ring. Depending on the design, a ring may also combine different materials; but even for a single material the appearance can be varied. The surface can range from a polished, highly specular surface, over glossy, to abraded (”brushed”) with a matt finished appearance. Figure 3 shows samples of rings with different material and surface characteristics. Combining metals with different colors can yield a quite unique ring with subtle visual contrasts, which has a striking look without the need for additional engravings or jewelry.

C. Patterns and Engravings

Modern wedding rings may contain both engravings and patterns. Engravings include artistic strokes, but also simple text, typically the names of the couple on the inside. With patterns the ring surface gets a somewhat rough appearance, as in Fig. 3 (ring pairs 1,5,6).

D. Gems

Typical gemstones used in wedding rings are diamonds, sapphires, rubies, emeralds, amethysts and aquamarines. Gems not only add value, but the gem distribution pattern often also has a symbolic meaning (“starry sky”, etc.).

IV. Parameterization

Fig. 1 shows the collection of shape instances spanning the design space of JohannKaiser wedding rings that is to be covered by a metadesign. Careful analysis reveals that the base shape of most rings can be defined using the following parameters: (1) a profile polygon, (2) the angular step size defined by the number of supporting profiles to be placed around the ring’s center, (3) the radius, and (4) a vertex transformation function. The idea is to decompose the design variations into a set of transformation functions. Each function selectively transforms certain vertices in a certain way. By calling a sequence of different transformations, the effects can be combined. This way, e.g., horizontal and vertical deformations can be separated; countless combinations are possible by simple function concatenation.

Figure 5a shows an example of a profile polygon, copies of which are placed radially around the origin (Fig. 5b). These polygons are first converted to double-sided faces. Then, corresponding back- and front-sides are connected using the `bridgerings` operator; it expects two halfedges of corresponding vertices of two faces that have the same number of vertices (Fig. 5d). The profile polygon, the rotation angle, and a set of custom parameters are the input to this transformation function.

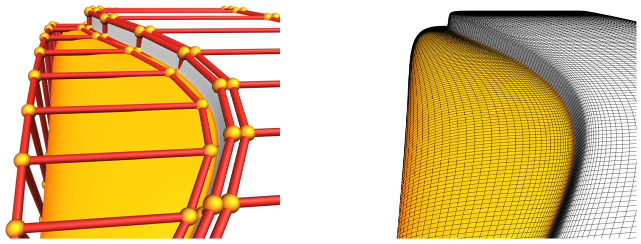


Figure 4. Semi-sharp creases can be obtained by close parallel edge pairs. Note that the transform function faithfully maintains this parallelity, which would not be possible, e.g., by simple scaling.

The custom parameters depend on the particular transformation used; the sine transformation in Fig. 6 for instance requires three parameters: frequency, amplitude, and the indices of the points of the profile polygon to which the transformation is to be applied. Depending on the placement angle it can modify the input polygon before it is placed in the scene. The result of the code in Fig. 6 is shown in Fig. 5f. In this case, the transformation is applied selectively only to the points 0-7 of the profile (see Fig. 4a). The transformation

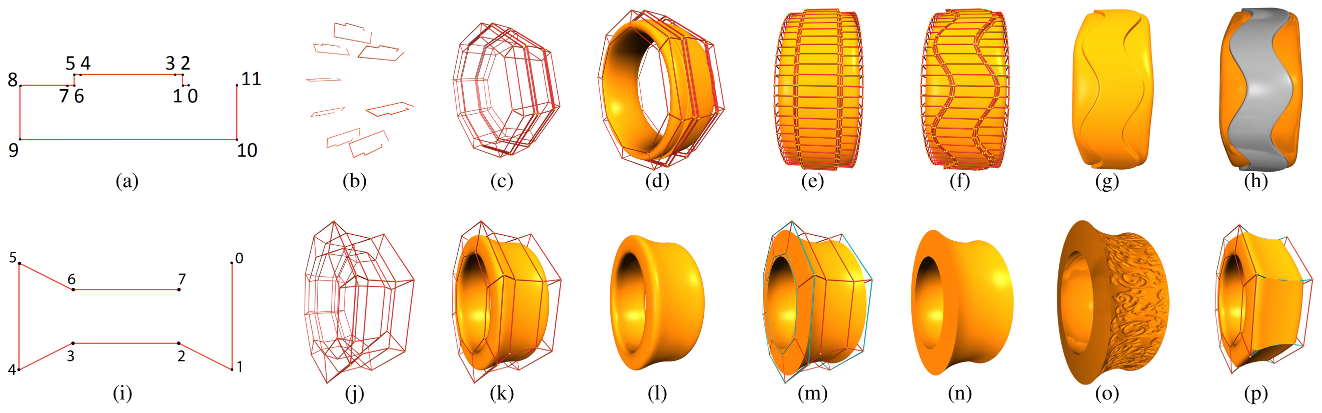


Figure 5. Parametric wedding ring construction. *Top row:* n copies of the the profile polygon (a) are radially placed around the center (b). Consecutive profiles are connected using the *bridgerings* operator to form a control mesh (c) that has a simple toroidal structure. It defines a subdivision surface forming the actual ring shape (d). Switching the profile transformation function from identity (e) to a selective sine transformation (f) creates a decorative wave on the surface (g). Using a post-processing step the wave faces can be differently colored. *Bottom row:* (j) The same basic toroidal construction applied to a another profile polygon. The resulting ring shape (l) contains only smooth edges. Using a post-processing step the connecting edges of the polygon points 0,1,4 and 5 are marked *sharp* (m), resulting in a sharp crease (n). The result can be decorated with a pattern (o). Tagging edges as sharp has great expressiveness in design (p).

function may arbitrarily displace the profile points; but the number of points may not change, otherwise bridgerings will fail. As optional input, an array of the first half-edge of each profile allows additional shape modifications.

The parametric construction described above creates a smooth ring shape with just one material. Assigning different materials to specific parts of the ring is done as a post processing step. Fig. 7 (left) shows the code for changing the material only of the wave. The parameters for RING.Tools.colorize are an array of half-edges (line 1), the indices of the affected faces (in this example those between the 3rd and the 4th profile point), and the material name. Figures 5m,n show the result of this code snippet.

Another post-processing step is adding sharp creases, which is simply done by tagging a selected set of mesh edges as *sharp*. The respective function call (Fig. 7 (right)) has the same structure as the material change.

V. Materials and Engravings

In our metadesign, we have first created a control mesh for the base shape of the ring (see Section IV). The inspection of the input shapes from Fig. 1 has led us to separating the *base shape* from the engravings (the so-called *meso-structure*) and the material appearance (*micro-structure*). For applying the meso-structure we use the technique of *displacement mapping*. The idea is to create on the fly a micro-tesselation, but only in regions of the surface with displacement (see Fig. 9). This exploits the vertex shader capabilities of modern graphics hardware, i.e., the micro-tesselation is created entirely on the GPU. This technique is very flexible, and useful not only for engravings. It can also be used to model certain irregular shape features that are too cumbersome to model on the control mesh level.

```
1  usereg !params !angle !profile
2
3  :params /points      [ ]  RING.tools.get-param !points
4  :params /frequency  12    RING.tools.get-param !frequency
5  :params /amplitude  0.5   RING.tools.get-param !amplitude
6
7  :points
8  {  !i
9     : profile  :i  get !p
10    :p
11       :p getX :angle :frequency mul sin
12       :amplitude mul add
13       putX !p
14    : profile  :i  :p put
15 } forall
```

Figure 6. A specialized sine transformation function for profiles. Input parameters are marked blue. parameter contains additional custom information: the indices of the polygon points to which the transformation is applied, and the frequency and amplitude of the sine function.

```
1  :edges
2  [ 3 ]
3  / silver
4  RING.Tools.colorize
```

```
1  :edges
2  [ 0 1  4 5 ]
3  { faceCCW }
4  RING.Tools.sharpen
```

Figure 7. Postprocessing code. **Left:** Assigning the material /silver to a set of selected faces. **Right:** Marking the specified half-edges as sharp.

A. Displacements

The basic idea of displacement mapping is to take sample points from a surface, and displace in the direction of the surface normal in a distance corresponding to the grey value in a supplied height map. The normal vector of the displaced vertices is calculated by combining the surface normal with the supplied normal map.

Displacement mapping techniques can generally be classified in per-vertex and per-pixel techniques. The latter only affects surface normals and, thus, the lighting calculation;

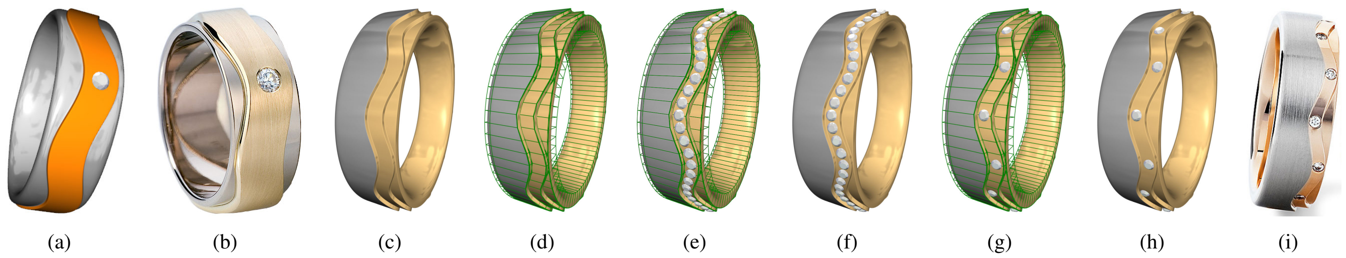


Figure 8. Placing diamonds: (a) A single diamond is placed on the ring. (b) The ”real” specimen. (c) Diamond placement can also exploit the structure of the parametric construction, e.g., by following the control mesh wave. (d,e) One diamond is placed at the midpoint of each half-edge of the wave. (g,h) Placement of diamonds at the points of maximum curvature of the sine wave. (i) The corresponding ”real” specimen.

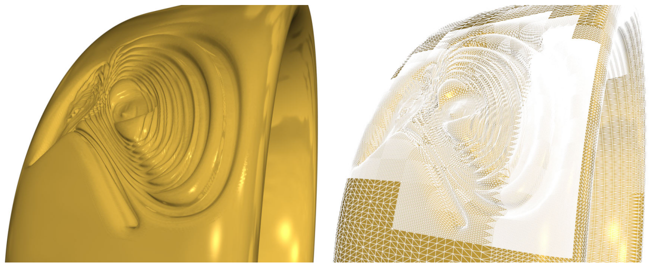


Figure 9. **Left:** The CGV logo from Fig. 10 (middle) is applied as displacement map. The grey value determines the distance in normal direction of the original surface. For light calculations, the normal map from Fig. 10 (right) is used. **Right:** The micro-tesselation is generated on the GPU, with one vertex for each pixel of the displacement map.

it virtually displaces points corresponding to texel centers. Consequently, per-pixel displacement is carried out by the texture unit of the fragment shader. Per-vertex displacement changes the positions of the of the original mesh vertices, and is therefore implemented in the vertex or geometry shader. So per-vertex displacement mapping creates actual geometry, whereas per-pixel displacement only creates the visual effect of a displacement. An overview of current displacement mapping techniques can be found in [13].

We want to use our metadesign not only for visualization, but also for rapid-prototyping purposes. Using the per-vertex approach is therefore mandatory, as it allows exporting the displaced geometry to a 3D Printer. The disadvantage of per-vertex displacement is that it requires a rather dense tesselation for highly detailed displacement maps. In our 3D-engine we overcome this issue by using a view-dependent tesselation scheme. Depending on the distance of the camera from the geometry, as well as on rendering speed, several levels of subdivision are available on the CPU as well as on the GPU to always display the appropriate level of geometric detail. Also efficient encoding schemes for displacement as well as normal maps, like presented in [14], can be used to reduce CPU to GPU data transfer and thus improve the performance. The ring in Fig. 9 illustrates the adaptive tesselation technique with the high resolution-tesselation in surface regions where a displacement is applied, and the significantly coarser tesselation used elsewhere.

A per-vertex displacement is encoded in two bitmaps,

- a *displacement map* holding the height values of the engraving as greyscale values, and
- a *normal map* providing one surface normal vector for each displaced vertex.

Figure 10 shows a displacement map (middle) as well as a normal map (right) created from the logo of CGV (left, Institute of ComputerGraphics and Knowledge Visualization).

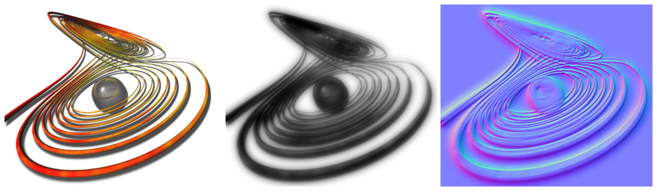


Figure 10. **Left:** The CGV logo. **Middle**: A displacement map with a slight blur effect. (Right:) The figure to the right shows the corresponding normal map. Displacement and normal information is encoded as pixels in the bitmaps.

These two bitmaps together with a starting half-edge, and with parameters to specify the scale and spatial extent, are sufficient to define the displacement. This is illustrated by the concise code fragment in Fig. 11.

B. Materials

The appearance of the microstructure of the material is applied in the final rendering stage. It uses the standard Blinn-Phong shading model in combination with an approximation of the Fresnel reflection term that can model anisotropic highlights. To make the surface look metallic, the specular highlight is modified by the color of the material, instead of

```
usereg
”dsp.png”   !dsp
”nrml.png”  !nrml
0.1         !scale
:dsp :nrml :scale :edge 3 3 dsp-displacetexture
```

Figure 11. Only a few lines of GML code are needed for applying displacements. The file names of displacement and normal map together with a scale factor, an edge, and two parameters defining the spatial extent are enough.

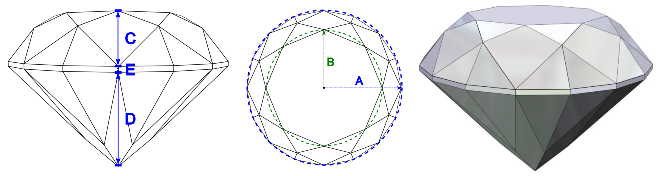


Figure 12. Side (left) and top (right) view of a brilliant. The five parameters that are used to define the shape of a brilliant in our system are *total diameter* (A), *table diameter* (B), *crown height* (C), *pavillion height* (D) and *girdle height* (E).

the color of the light source. We use a static cube map to reflect the environment.

The ring's surface can be shiny or matte, but accurate simulation of a matte surface would be very time consuming. To simulate matte surfaces, we use a blurry cube map. The reflected color is mixed with the result of the Blinn-Phong [15] shader using a Fresnel term F_λ [16]. The Fresnel reflection term is calculated as $F_\lambda = R_v \cdot N$ using the view ray R_v and the surface normal N. It exhibits more intensive reflections for flat viewing angles, and shows more of the material color for othogonal viewing angles.

VI. Adorners - Gems

In this section, we describe the parametrization of adorning elements like gems.

Our gem model is of the round brilliant cut type, due to its dominant position in the market [17]. Similar to Hemphill et. al, our mathematical model consists of a convex polyhedron (CP) representing the surface of the brilliant. A CP is a convex set bounded by planar regions, in other words a volume with planar facets that is approximately corresponding to a sphere (in the sense that it contains no dents). In our system a CP is defined by the planes of its facet. Each plane is defined using three 3D points; the orientation of these points specifies the orientation of the plane, thus permitting to distinguish between points *below* and *above* the plane. The interior of a CP is defined as the set of points that are simultaneously *below* all of the facet planes.

In comparison to Hemphill et al. [17], our parametrization is slightly simplified (without culet facet); we use only five parameters to define the shape, as shown in Figure 12. As it is common that the gems used on one ring are all the same or very similar, the system differentiates between the instantiation of a diamond, given a parameter set, and the placements of instances of this diamond. GML provides the functionality to evaluate the surface of a CP given its facet planes, which is used to create the instance geometry of a procedural gem.

Procedural gem instances can then be placed by specifying the apex point, an upvector, and a scale factor. The gems are placed on the ring surface by performing a ray-mesh intersection to determine a ring surface point. Fig. 8 shows

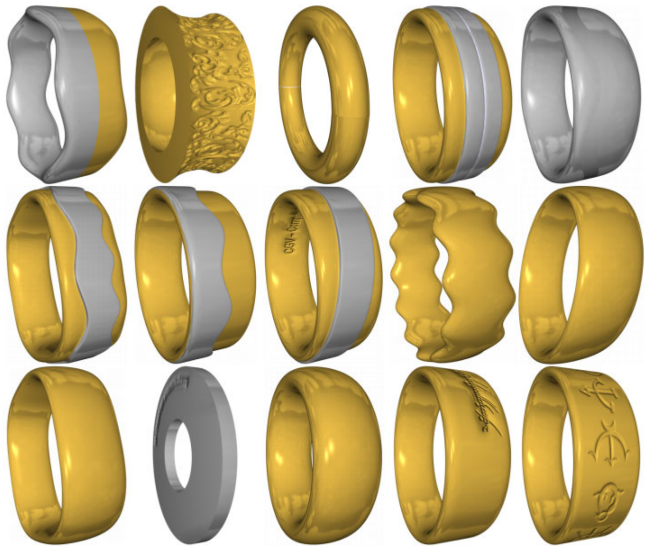


Figure 13. Output of the parameterization process: A metadesign that can generate a continuous variety of ring designs, from which only a few instances are shown

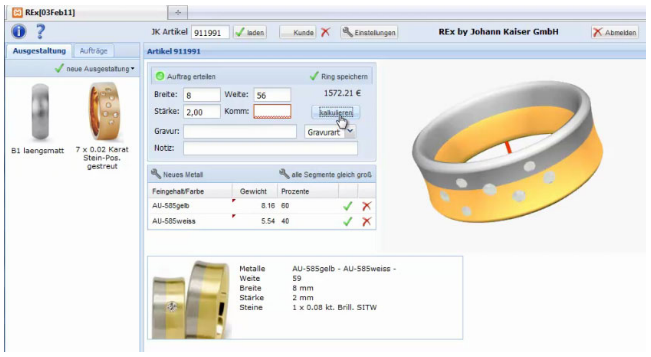
the placement of a single diamond as well as the parametric placement following a geometric feature.

VII. Conclusion and Future Work

We have presented a case example for the efficiency of procedural shape modeling in for the mass customization of wedding rings. Fig. 13 shows a few results of our parametric toolkit. The current implementation is integrated into the REx system (see Fig. 14), which is based on the GML browser plugin [18].

We consider the presented exercise to be a blueprint that can be efficiently carried out also for other tasks of creating metadesigns for mass-customizable shape domains. The production of a metadedesign that is able to reproduce almost all of the input shapes took only about six weeks. The resulting GML code is only less than 20 KB large, and the whole GML engine is contained in an ActiveX control of less than 6 MB size. So we think it is fair to say that we have produced a simple solution for a complex problem. The host application, the REx wedding ring designer, is already commercially available. The system supports immediate cost calculation within each design step (e.g., depending on the chosen material or the number of gems). The added value of our metadesign is that a variety of new wedding ring designs is available, which are part of the shape space spanned by the exemplars, but were so far not part of the (already extensive) product catalogue of the company.

Adorning elements like gems are a nice addition when designing rings. Sometimes, gems are not directly placed on the ring's surface but in a socket. Modeling sockets via the control mesh would be too cumbersome and may have unwanted side effects when using displacements, for



JK Artikel: 911991

Artikel 911991

Breite: 8 Weite: 56 1572.21 €

Stärke: 2,00 Komm:

Gravur:

Notiz:

Feingehalt/Farbe	Gewicht	Prozente
AU-585gelb	8.16	60
AU-585weiss	5.54	40

Metalle	AU-585gelb - AU-585weiss -
Weite	59
Breite	8 mm
Stärke	2 mm
Steine	1 x 0.08 kt. Brill. SITW

Figure 14. The resulting REx application from JohannKaiser, a wedding ring configurator to be used by jewelers, is already commercially available.

example. Help in the form of Constructive Solid Geometry (CSG) would allow to create complex surfaces using Boolean operations. The next important aspect for future work is to further enhance the visual quality. Currently, our renderer can handle shiny, polished or matte surfaces. A very common surface finish of rings is brushing. For case like this, the render should be able to handle various different surface characteristics. This requires a coherent, more global (u, v)-parametrization of the surface (procedural texture coordinates).

The presented approach is considered to be used for visualization, information purposes only. Ensuring manufacturability, functionality or durability of the created designs is part of future work. However, we think that a lot of potential problems can be avoided by carefully defining parametric constraints.

For dissemination purposes amongst end-users, a web-based user interface offering clear, but certainly limited means of configuration would be sensible. There is no need to create a browser plugin. Ideally, modern web-technology can be used to create a meaningful user interface and a server-side rendering approach can deliver photo-realistic renderings.

Acknowledgment

We would like to thank JohannKaiser for supplying sample wedding rings as well as for the permission to use their ring photographs in this paper. In addition, the authors gratefully acknowledge the generous support from the Austrian Research Promotion Agency (FFG) for the research project METADESIGNER, grant number 820925/18236.

References

[1] E. Li, “Jewellery CAD/CAM Limited,” online: http://www.jcadcam.com, retrieved: 06, 2012.

[2] E. Nir, “iRing3d,” online: http://www.iring3d.com, retrieved: 06, 2012.

[3] H. Singh, P. Tandon, and V. Gulati, “Article: A jewelry modeler for carved bangles,” *International Journal of Computer Applications*, vol. 5, no. 2, pp. 25–27, August 2010, published By Foundation of Computer Science.

[4] S. W., “An aesthetics driven approach to jewelry design,” *Computer-Aided Design and Applications*, vol. 7, no. 4, pp. 489–503, 2010.

[5] E. Nir, “Paracloud gem,” online: http://www.paracloud.com, retrieved: 06, 2012.

[6] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*, P. Prusinkiewicz and A. Lindenmayer, Eds. Springer-Verlag, 1990.

[7] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. V. Gool, “Procedural modeling of buildings,” in *ACM SIGGRAPH*, vol. 25, 2006, pp. 614 – 623.

[8] B. Ganster and R. Klein, “An Integrated Framework for Procedural Modeling,” *Proceedings of Spring Conference on Computer Graphics 2007 (SCCG 2007)*, vol. 23, pp. 150–157, 2007.

[9] C. Reas and B. Fry, *Processing: A Programming Handbook for Visual Designers and Artists*. MIT Press, 2007.

[10] S. Davidson, “Grasshopper - generative modeling for Rhino,” online: http://www.grasshopper3d.com, retrieved: 06, 2012.

[11] S. Havemann, “Generative Mesh Modeling,” Ph.D. dissertation, Braunschweig Technical University, Germany, November 2005.

[12] S. Havemann, R. Berndt, and D. W. Fellner, “Website of the *generative modeling language* gml,” online: http://www.generative-modeling.org, retrieved: 06, 2012.

[13] L. Szirmay-Kalos and T. Umenhoffer, “Displacement mapping on the GPU - State of the Art,” *Computer Graphics Forum*, vol. 27, no. 1, 2008.

[14] C. Schinko, T. Ullrich, and D. W. Fellner, “Simple and Efficient Normal Encoding with Error Bounds,” *Proceedings of Theory and Practice of Computer Graphics*, vol. 29, pp. 63–66, 2011.

[15] J. F. Blinn, “Models of light reflection for computer synthesized pictures,” *SIGGRAPH Comput. Graph.*, vol. 11, no. 2, pp. 192–198, Jul. 1977.

[16] M. Pharr and R. Fernando, *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems)*. Addison-Wesley Professional, 2005.

[17] S. T. Hemphill, I. M. Reinitz, M. L. Johnson, and J. E. Shigley, “Modeling the appearance of the round brilliant cut diamond: An analysis of brilliance,” *Gems & Gemology*, vol. 34, pp. 158–183, 1998.

[18] R. Berndt, D. W. Fellner, and S. Havemann, “Generative 3d models: A key to more information within less bandwidth at higher quality,” in *Proc. Web3D 2005 Intl. Symp.* ACM Siggraph, 2005, pp. 111–122.