# DGD Gallery: Storage, Sharing, and Publication of Digital Research Data

**Michael Joswig, Milan Mehner, Stefan Sechelmann, Jan Techter and Alexander I. Bobenko**

**Abstract** We describe a project, called the DGD Gallery, whose goal is to store geometric data and to make it publicly available. The DGD Gallery offers an online web service for the storage, sharing, and publication of digital research data.

## 1 Introduction

Software produces data. Mathematical software produces scientific data, and this is often worth keeping. One reason for this can be the vast amount of CPU time spent on a specific experiment. Another reason can be that the output is obtained only via a complex interaction process between the software and its user. That latter situation is typical in mathematical visualization, where producing a satisfying or even beautiful picture of a geometric object is a form of art. The purpose of this text is to describe a new project, called the "Discretization in Geometry and Dynamics Gallery", or DGD GALLERY for short, whose goal is to store geometric data and to make it publicly available. The URL of the web-site is

http://gallery.discretization.de

M. Joswig (✉) · M. Mehner · S. Sechelmann · J. Techter · A.I. Bobenko
Inst. für Mathematik, Technische Universität Berlin,
Straße des 17. Juni 136, 10623 Berlin, Germany
e-mail: joswig@math.tu-berlin.de

M. Mehner
e-mail: mmehner@math.tu-berlin.de

S. Sechelmann
e-mail: sechel@math.tu-berlin.de

J. Techter
e-mail: techter@math.tu-berlin.de

A.I. Bobenko
e-mail: bobenko@math.tu-berlin.de



A.I. Bobenko (ed.), *Advances in Discrete Differential Geometry*,
DOI 10.1007/978-3-662-50447-5_14

Today it is safe to say that finally mathematical software has reached every branch of mathematics. While computers have played a role in mathematical applications for a long time, it took considerably longer for software to be appreciated fully in parts of mathematics traditionally considered as "pure". The array of tools at our fingertips now includes solvers for linear programs and partial differential equations, but also software for dealing with real algebraic sets [1] or delicate constructions in sheaf theory [2]. To get an idea of how rich the mathematical software landscape has become, see, e.g., [3]. The success of each single software system raises the question of how the respective data produced should be stored. With an increasing number of relevant mathematical results relying on non-trivial computations in an essential way (see, e.g., the `Flyspeck` project [4]) it becomes more and more crucial to publish such results in a way such that they can be scrutinized (and used) by the mathematical community.

The mathematical data we have in mind for the DGD GALLERY are the geometric objects that occur naturally on the border between differential geometry and geometric combinatorics. This includes various classes of surfaces (embedded or immersed) in 3-space, convex polytopes and polyhedral fans of various dimensions, circle patterns, and many more. Yet, we believe that several of our design decisions and architecture ingredients will be useful for other collections of mathematical data. Key features include the following:

- structured storage of research data,
- review process for increased reliability,
- migration process for sustainability,
- licensing scheme.

To further stress the relevance of our endeavor, it is worth noting that scientific funding agencies have begun to add requirements concerning the preservation of scientific data to their regulations. For instance, in a recent announcement [5] of Deutsche Forschungsgemeinschaft (DFG) says[1]:

> The documentation of research data according to standards depending on the subject and their long-term archival are relevant for controlling the quality of scientific work. Further, these data are the basic requirements for the subsequent use of research results.

The DGD GALLERY evolved as a project within the DFG Collaborative Research Center SFB/TRR 109 "Discretization in Geometry and Dynamics". Its usage is currently restricted to the members of the center. However, it is intended that future versions allow other researchers to contribute their work, too.

The paper is organized as follows. First we compare our design to existing collections of geometric data (Sect. 2). Then, in Sect. 3, we exhibit some examples already published on the gallery. This should give a good idea of what kind of collection we have in mind. At the same time this also shows some of the technical features and capabilities. The core is Sect. 4, where we elaborate on the architecture and the design decisions. The key concept is the *model*, which is our technical realization of a geometric object. Some aspects of the implementation are covered in Sect. 5. For

[1] Translated from German.

instance, we explain how we use the XML document database BASEX [6] and meet current standards of web technology.

## 2 Comparison with Previous Work

To store geometric data digitally and make it accessible through a web-site is clearly not a new idea. On the contrary, since the early days of the Internet people have set up numerous web-sites with all kinds of information on geometric objects, e.g.: The Geometry Center's "Geometry Reference Archive" [7], "The Scientific Graphics Project"of MSRI [8], or David Eppstein's "Geometry Junkyard" [9], to name a few prominent examples. Clearly, all of the above still contain lots of interesting information. However, there are some shortcomings. In the case of the archive of The Geometry Center we have a static collection of data that will not see any updates or additions. Yet there is the advantage that all data is available from one source, and so it cannot degenerate over time (except for eventually outdated file formats). Not so with the "Geometry Junkyard". This is a collection of links to other interesting resources on the web. Many of the links are dead already. This is mainly due to a discontinued provider service or simply a change of position of the person who provided the data. The "Scientific Graphics Project" is mainly a collection of surfaces and differential geometry related publications. The DGD GALLERY wants to cover geometric objects from a much wider collection.

A more recent project is the "GeometrieWerkstatt" [10] maintained by a group of geometers at Tübingen University. It contains visualizations of mostly smooth constant mean curvature surfaces. Surfaces are visualized using videos, images, and interactive 3D viewers. The main difference to the DGD GALLERY is that there is no geometric data that can be accessed via the web-site. On the other hand, how to provide the data for smooth surfaces is far from obvious and cannot be separated from the mathematical methods. For the DGD GALLERY we propose to include a discretization of a smooth surface in a reasonable resolution.

"IMAGINARY—open mathematics"is a platform [11] which has a strong educational focus. It features images and mathematical software for a broad audience such as exhibitions, high school education, and museums. As an essential feature, IMAGINARY is open for the public to contribute material by cross-linking to other web-sites. In this way it works like a collection of collections.

The focus of the SymbolicData project [12] is on developing concepts and tools for profiling, testing and benchmarking Computer Algebra Software. This includes storing scientific data from various sources, but visualization does not play a role.

The project that is most similar in spirit to our DGD GALLERY is "Electronic Geometry Models" [13, 14], which is a refereed online journal for digital geometry models on the web. It features XML file formats, visualization separated from descriptions, and a reviewing process. All of these are also implemented in the DGD GALLERY.

However, our technical realization substantially differs from "Electronic Geometry Models". The DGD GALLERY employs modern web technology for the user interface and a standard data base implementation for storing. One advantage of this

is the possibility to work in teams. Each team member can contribute to a model if he/she is a registered user with the suitable permissions. Permissions can be granted by owners of content; for details see Fig. 7 below. Moreover, the entire work flow from the submission, through reviewing and revising, to the final publication has one consistent setup through a common front end. Most importantly, the overall design is highly modularized. For instance, the DGD GALLERY features a variety of media renderers with different visualization strengths to accommodate for heterogeneous hard- and software environments at the users' end. This is also relevant for being able to preserve the data over a long period.

Another difference to "Electronic Geometry Models" is that the DGD GALLERY aims at a broader outreach and therefore seeks to include more models of purely educational value. This results in a different set of criteria for accepting a model for publication. Moreover, the DGD GALLERY allows for changes to a model after publication.

## 3 Examples

In this section we present some selected models from the early contributions to the DGD GALLERY. They are intended as guidelines and inspiration for future models to be submitted.

### *3.1 Discrete S–Conical Catenoid and Helicoid*

Authors: Alexander Bobenko, Tim Hoffmann, Benno König, and Stefan Sechelmann (Fig. 1)

http://gallery.discretization.de/models/sc-catenoid

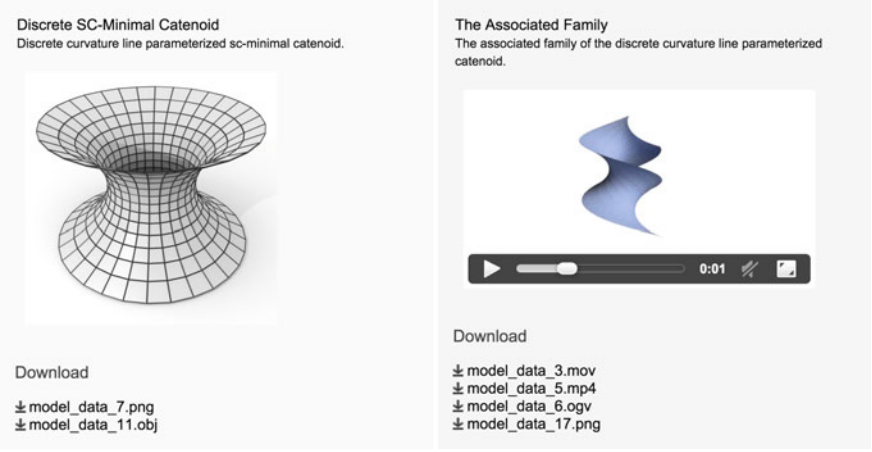


**Fig. 1** Screenshot of two media objects contained in the "Discrete S-Conical Catenoid and Helicoid" model as presented by a modern web browser. *Left* Discrete s-conical catenoid. *Right* Associated family animation between s-conical catenoid and its conjugate, the discrete helicoid

This model shows discrete s-conical versions of the catenoid and the helicoid, which are classical minimal surfaces [15]. The smooth versions are among the first classical minimal surfaces ever investigated. Their s-conical counterparts are quadrilateral polyhedral surfaces with the property that at each vertex the adjacent faces are tangent to a cone of revolution. The theory of these discrete minimal surfaces is closely related to the theory of orthogonal circle patterns and Koebe polyhedra; see Sect. 3.3 below. Its features and constructions are similar to the theory of s-isothermic surfaces. A minimal surface is (Christoffel) dual to its Gauss map. This property is preserved in the discrete setup, and so discrete minimal surfaces are constructed from Koebe polyhedra. The associate family of minimal surfaces is contained in the discrete theory as well.

The model features images of the catenoid and helicoid using representations with discrete curvature line parameterizations as well as discrete asymptotic line parameterizations in the associate family. It contains a video with an animation of the associate family animating the angle parameter. Geometric data is given as OBJ files and corresponding preview images.

## 3.2 $z^a$ Circle Pattern

Authors: Jan Techter and Jürgen Richter-Gebert (Fig. 2)

http://gallery.discretization.de/models/zalpha_circle_pattern

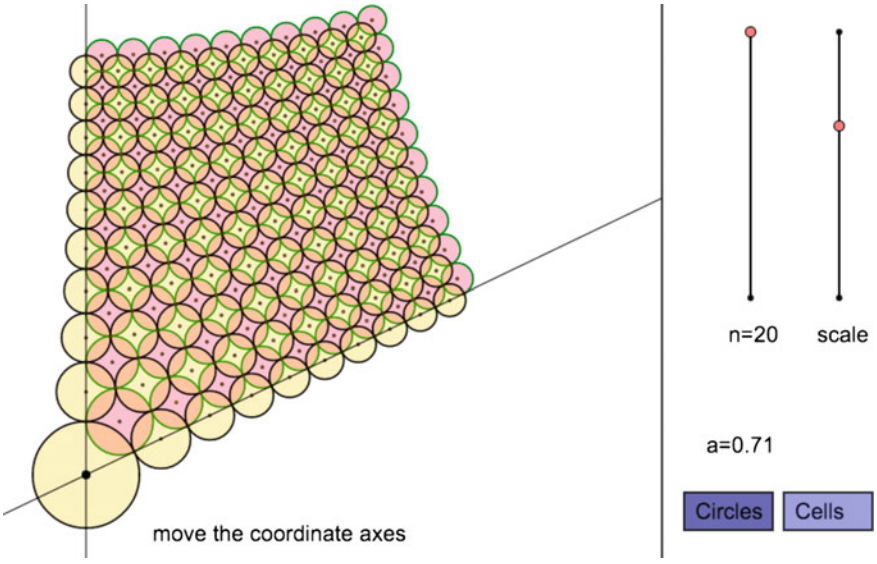


**Fig. 2** Screenshot of the interactive element of the "$z^a$ *circle pattern*" model. A user can adjust the number of *circles* in a row as well as the overall scale of the drawing. The angle $\alpha$ is entered by moving the axes with the mouse

The representation of discrete holomorphic functions by circle patterns with square-grid combinatorics was first studied by Schramm [16].

This model shows the Schramm type circle pattern corresponding to the holomorphic map $z \mapsto z^a$ for $0 < a < 2$ in the first quadrant of the complex plane. Taking the centers and intersections of the circles as complex fields on the first quadrant of $\mathbb{Z}^2$, the discrete map was introduced in [17] as a special isomonodromic solution of the cross-ratio equation (cross-ratio equal to $-1$ on each elementary quadrilateral). The numerics of these discrete maps is studied in [18].

The model features an interactive Cinderella [19] application where the user can adjust the exponent $a$ and the number of circles, see also Sect. 5.3.

### *3.3 Koebe Polyhedra*

Author: Stefan Sechelmann (Fig. 3)

http://gallery.discretization.de/models/koebe_polyhedra

A *Koebe polyhedron* is a 3-dimensional convex polytope whose edges are tangent to the unit sphere. Koebe polyhedra have a strong connection to the theory of circle patterns, see [20]. The theory of discrete minimal surfaces of s-isothermic and s-conical type is based on Koebe polyhedra. Each combinatorial type of 3-polytope admits a representation as Koebe polyhedron, which is unique up to Möbius transformation.

The first step for the construction of a Koebe polyhedron is to create an orthogonal circle pattern corresponding to the desired polytopal cell decomposition of the sphere. This is generally done by finding critical points of a functional expressed in the the variables $\rho_i = \log \tan \frac{r_i}{2}$ given by the spherical radii $r_i$. Once the radii are known

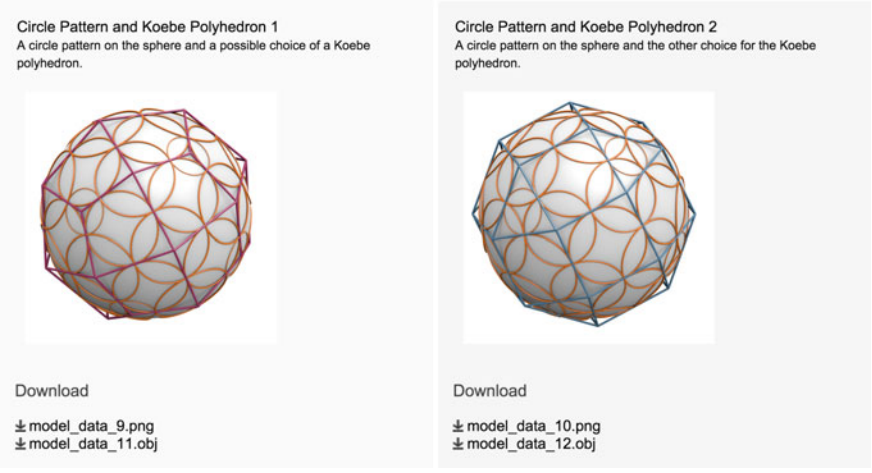


**Fig. 3** Screenshot of two media objects of the "Koebe Polyhedra" model. The two images show the two corresponding Koebe polyhedra for a given *circle pattern*

the circles can be layed out. The still remaining freedom of applying a Möbius transformation can be fixed (up to a simple rotation) by requiring the center of mass to be at the sphere center. The vertices of the circumscribed Koebe polyhedron that corresponds to the circle pattern can now easily be found by inverting the euclidean centers of the circles in $\mathbb{S}^2$ (the cone tips are the points polar to the planes containing the circles). Here we have the freedom to choose one of the two orthogonal families of circles to become vertices of the Koebe polyhedron, and the other family to become faces.

The online model features a selection of Koebe polyhedra. Each one with an OBJ geometry file and a PNG image file.

## *3.4 Lawson's Surface Uniformization*

Authors: Stefan Sechelmann, Alexander Bobenko, and Boris Springborn (Fig. 4)

http://gallery.discretization.de/models/lawsons_surface_uniformization

Fuchsian uniformizations of the Riemann surface of Lawson's genus 2 minimal surface in $\mathbb{S}^3$ [21] are presented in this model. The results were created in [22] using the discrete uniformization theory. Three different conformally equivalent representations of the surface and of the corresponding hyperbolic tilings are presented.

Lawson's minimal surface in $\mathbb{S}^3$ is conformally equivalent to the hyperelliptic curve $\mu^2 = \lambda^6 - 1$. The branch points $\lambda_1, \ldots, \lambda_6$ are the 6th roots of unity.

An embedding of Lawson's surface in $\mathbb{R}^3$, see Fig. 4, is obtained via stereographic projection from $\mathbb{S}^3$ [23]. For this surface the hyperelliptic involution of the Riemann

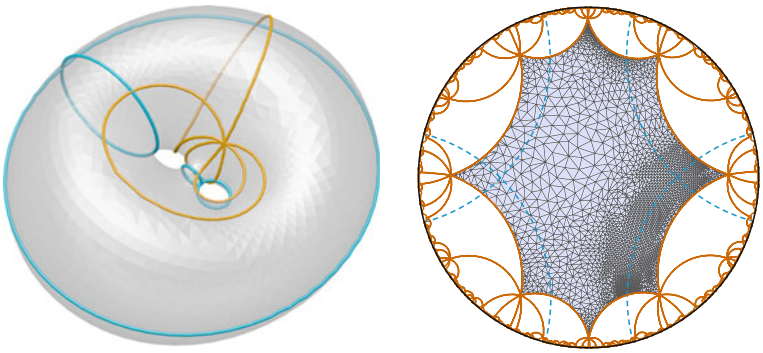

**Fig. 4** *Left* The Lawson surface in $\mathbb{R}^3$, the boundary curves of the fundamental domain of the uniformizing group in the *right picture* are shown in *red*. *Blue curves* correspond to simple closed geodesics corresponding to the axes of generators of the group. *Right* The uniformization of the Lawson surface in the Poincaré model of hyperbolic space with a canonical fundamental domain (*red*) and axes of the hyperbolic generators of the uniformizing group

surface is realized as a rotation by 180°. The axis meets the surface in six points, which are the branch points of the hyperelliptic curve.

The third realization of the Riemann surface is made of squares identified along suitable edges. The fundamental domain is identified with the two others.

The model features the data of the discrete uniformizations in XML format. It contains the combinatorial data, the coordinates of the points, and the uniformizing groups data. PDF vector graphics and PNG images provide 2D renderings of objects in 3D space.

## *3.5 Tropical Grassmannian TropGr(2,6)*

Authors: Michael Joswig and Benjamin Schröter (Fig. 5)

http://gallery.discretization.de/models/tropical_grassmannian_gr26

Tropical geometry studies piecewise linear images of classical algebraic varieties. Many interesting properties remain visible in the tropicalization. Additionally, this method reveals relations between geometry and optimization. One outcome are combinatorial algorithms for dealing with classical objects.

The *tropical Grassmannian* $\mathrm{TropGr}(d, n)$ is the tropicalization of the classical Grassmannian $\mathrm{Gr}(d, n)$, defined over some field. It parameterizes the tropical $d$-planes in the tropical $(n-1)$-torus; see [24, §4.3]. For $d = 2$ the tropical Grassmannian coincides with the corresponding *Dressian*, which arises as the subfan of the secondary fan of the hypersimplex $\Delta(d, n)$ corresponding to those regular decompositions whose cells are matroid polytopes [25].

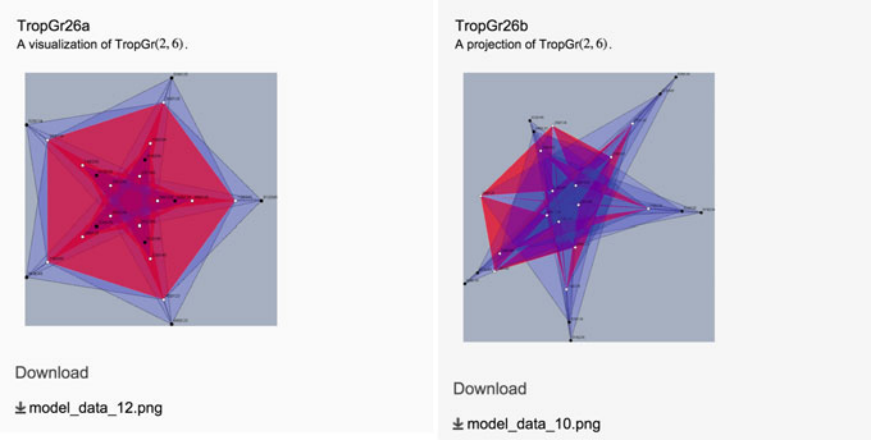


**Fig. 5** Screenshots of two images contained in the "Tropical Grassmannian TropGr(2,6)" model

How to properly visualize TropGr(2, 6) is far from obvious, since (modulo its lineality space and intersected with the corresponding unit sphere) this is a 2-dimensional spherical simplicial complex naturally embedded in the 8-sphere. It has 25 vertices, 105 edges and 105 triangles. The approach here employs a fixed copy of smaller tropical Grassmannian TropGr(2, 5), obtained by deletion, as a frame of reference and uses projections. The *deletion* of a matroid as a smaller matroid which is induced on fewer elements, and this notion carries over matroid decompositions.

The media objects associated with this model are a `polymake` [26] description and pictures of various projections, in PNG format.

## 4 Architecture

In this section we describe the structure of a model and the organization of data within the DGD GALLERY. It is also explained how users create, edit, and interact with models using model permissions. Finally, we give details on the submission system and the review process.

### *4.1 What Is a Model?*

The architecture of the DGD GALLERY is built around the definition of the `Model`, see Fig. 6. The `teletype` font is used to indicate that a word is the name of an abstract data type, one of its attributes, or an admissible value. From a high level perspective a *model* is a collection of files together with a description. The description contains fields for the title, authors, a description text, keywords, literature references, and the creation date. The data files associated with a model are bundled into media objects. A *media object* is a set of files together with a title and a description text. These files may be images, videos or data for specific software systems. While some file formats are more common (and more reasonable) than others, conceptually we allow for any file format to become part of a media object. In this way our design is very flexible and thus could be applied in other contexts.

The data type `Model` has a `key`, a `version` number, a `status` field, and an `edited-by` username. The model key is a unique identifier that is used, e.g., to assemble the permanent link of the model on the web. The version number is assigned automatically for keeping track of a model's history. Throughout the following the word "model" refers both to a specific version and to the entire history of a model. The standard representative of a model is given by its latest version. The `edited-by` field contains the username of the author of a particular model version. Hence, in a database a model can be uniquely identified by its key, a model version is identified by its key and version number.

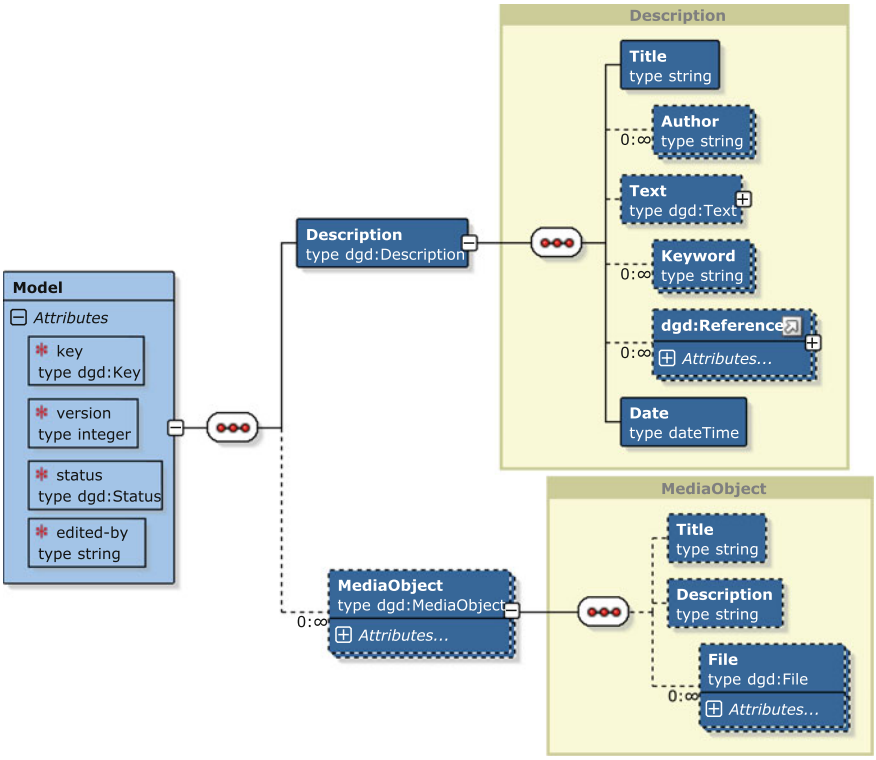


**Fig. 6** The structure of a model. A model is a collection of media objects together with a description

The status of a model can take the values `edit`, `pending`, `rejected`, or `approved`. See Sect. 4.5 for a detailed description of the model status and the submission process for models in the gallery.

The model description is a collection of the following information that is provided by the editor of a model. While this somehow resembles the structure of a traditional research paper, there are some notable differences.

- The `Title` of the model.
- A sorted list of `Authors`. Since this is a frequent source of misunderstanding, it is worth explaining. The authors of the model are those who create the content that is presented online. Like for a research paper all the scientific work that leads to the model must be properly acknowledged in the references. However, clearly, the set of authors cannot comprise all the authors who contributed something to the entire history of a mathematical idea. For instance, suppose that Alice first describes a new type of surface in a traditional research paper, and Bob afterwards produces a model from Alices description (without Alice's help). Then Bob is the only author, who must cite Alice's paper.

- The description `Text`, which can contain any valid LaTeX source code that can be compiled using the MathJax library, see [27]. References to the literature or to media objects can be cited via the `\cite{.}` command. Previews of media objects can be included with the `\media{.}` command.
- A set of keywords can be assigned to the model. The keywords are used on the web-site to, e.g., improve search features.
- A set of references each of which consitsts of a reference key that is to be used in `\cite` commands and a set of key value pairs. The user interface of the gallery maps BibTeX entries to model references. Conversely a model reference is rendered and referenced using common BibTeX styles.
- The date field of the description contains the creation date of the particular version of the model.

A model contains a number of media objects for visualization and use in other software systems. This concept will be explained below.

## *4.2 Media Objects and Data Files*

A media object is a collection of files that describe the same set of data associated with the model. For instance, several media objects might correspond to various views of the same model; e.g., see the tropical Grassmanian in Example in Sect. 3.5. A different use case for several media objects for the same model is displayed for the discrete catenoid and helicoid model in Example in Sect. 3.1. One media object shows a catenoid, whereas the other media object contains a dynamic rendering of the transformation from the catenoid to the helicoid.

The various file formats for one media object are meant to display one view of the model on several backends. For instance, the discrete catenoid media object comes with a PNG file to be displayed in a standard web browser and with an OBJ file which allows 3-dimensional interactive visualization with a suitable viewer software. The data files comprising the media objects are stored in the file system separately from the model database. The media objects of a model contain links to those data files, see Sect. 4.1.

In principle, we do not restrict the file formats for data files of any media objects. This makes the DGD Gallery very flexible, but this also creates potential trouble with file formats that are uncommon. We support the direct visualization of a few well chosen standard file formats. So far these include the following: PNG and JPG (for raster image data), SVG and PDF (for vector graphics), OBJ (for 3-dimensional geometric data), MOV, MP4, and OGV (for video content), POLY (for `polymake` data), and others. Interactive content is not excluded, see Sect. 5.3, but the danger of a particularly low stability over time should be well considered. We rely on the review process for a sound selection.

### 4.3 Versioning

The DGD GALLERY tracks the history of each model via the `version` attribute, see Fig. 6. Editing a model amounts to adding a new version with modified content. If a model is deleted, all versions of the model and the data files linked are deleted from the database. A data file is kept in the file system as long as there exists a link to it from some version of a model.

The version system is particularly useful for models with several authors who can collaborate through our front end.

It is worth noting that we also allow published models to be edited further and resubmitted. Upon acceptance this new version will appear as the current version of the model on the web page. The previously published versions remain visible and can be compared. This way authors can keep their models up to date; see also Sect. 5.2 below which describes our migration process.

### 4.4 Users

The users of the DGD GALLERY are represented by their usernames, i.e., their `login` names on the web-site. The access is password restricted, see also Sect. 5.1. In addition to the username and password we store the name and email address of the person that is associated with the user.

A user has a global user-role that can take the values `admin`, `reviewer`, or `author`. In addition to the global user-role we store model-roles for each model associated with a user. A user can be the `owner` or an `editor` of a model. The read and write access to models is restricted such that it is based on a combination of the global user-role, the model-role and the state of the model, Fig. 7. This implementation allows reviewers to act as model authors but prevents them from approving their own models.

A notable design decision is that a reviewer can modify a submitted model to correct obvious typos and other minor changes before approval. Each owner of a model can invite other users to become either owners or editors of that model.

### 4.5 Submission Process

The DGD GALLERY uses a submission system to publish models on the web-site. The idea is that a board of reviewers approves, sends back for revision, or rejects a submitted model. The review process should concern the quality of the content and address technical issues with the digital data. The review board has to work out and agree on some quality criteria for a model.

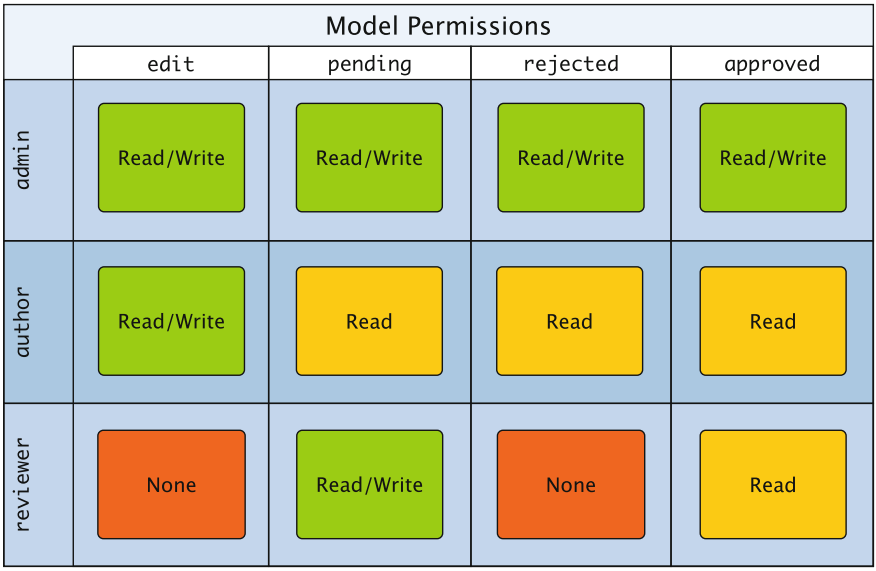


**Fig. 7** Read and write permissions during the life-cycle of a model. A user with global `admin` privileges can read and write on the model at any state (*first row*). The `author` of the model can edit his model if it is in `edit` state (*second row*). A `reviewer` can edit a model if it has been submitted (`pending` state, *third row*)

During its life-cycle a model has assigned a status value. A newly created model starts its life in `edit` state. It can be previewed and edited by the `owners` of the model, typically the creator of the model, and any additional user with the `editor` model-role, see Sect. 4.4.

A model can be submitted by a user with the `owner` model-role. The status of the model changes to `pending`. A model with `pending` status is read-only for the `owner` and all `editors`.

Reviewers can preview and edit `pending` models. A reviewer edits a model to resolve small issues such as typos. If the quality of a model is sufficiently high then a reviewer can accept a model. The status of the model is changed to `approved`. If the content has flaws or technical issues that can be resolved by the creator of the model, the reviewer sends the model back to `edit` state. Any action by a reviewer is accompanied by a review text, which is presented to the authors of the model.

If a model is sent back for revision, the authors can edit the model according to the review text and resubmit. If the model is `rejected` it can neither be edited nor resubmitted. A model will be rejected if it contains major flaws or its content is not appropriate for publication in the DGD GALLERY.

Approved models become publicly available on the DGD GALLERY web-site (see Sect. 4.6). To further improve public models, e.g. by correcting errors or replacing outdated file formats, a new version of an approved model can be created, which is back in `edit` state. To publish the new version it has to be submitted und undergo the revision process again.

The model submission system dispatches messages to the users of the DGD GALLERY on every model status change. Reviewers are notified about submitted models. Model owners/editors are notified upon acceptance, rejection, or call for revision.

In principle any reviewer can accept, send back, or reject a model. We rely on a reasonable communication between the reviewers to organize the review process. Accepting a model is based on formal correctness, technical soundness, mathematical content and visualization quality.

### *4.6 Publication and Licensing*

Content that has been approved by the board of reviewers is published on the DGD GALLERY web page. The presentation of the content on this page is equivalent to the preview during `edit` state of the model. The `key` defines the permanent absolute URL of a model:

https://gallery.discretization.de/model/

The content of the DGD GALLERY is published under the Creative Commons Attribution-ShareAlike 4.0 International license, short CC BY-SA 4.0, see [28]. This means in particular that we allow for our data to be used commercially, enabling newspapers or commercial web blogs to include content from the gallery without further complications. Appropriate credit must be given if any content is reproduced or used, and this includes a link to the DGD GALLERY.

## 5 Implementation

In this section we elaborate on the technical decisions that we made in order to implement the DGD GALLERY. It should give an impression of the system architecture, libraries, frameworks, and languages in use and their respective purposes.

We imposed some a priori constraints on the implementation mainly to ensure reusability and persistence of the data over time.

- **Human readable data format** (with enough structure to allow for easy validation and transformation): We chose XML for storage on the server and as the web server API data format. It fits the tree-like structure of our data and can be transformed to anything else, e.g. using XSLT. This allows for easy migrations which can range from changing the structure of the models to getting rid of XML itself (replacing it with some more sophisticated data format in the future). To ensure that all stored data, and in particular data entered by users of the system, agrees with our specifications we use the XML Schema concept [30]. This allows to validate all data on insert and during migration, see Sect. 5.2.

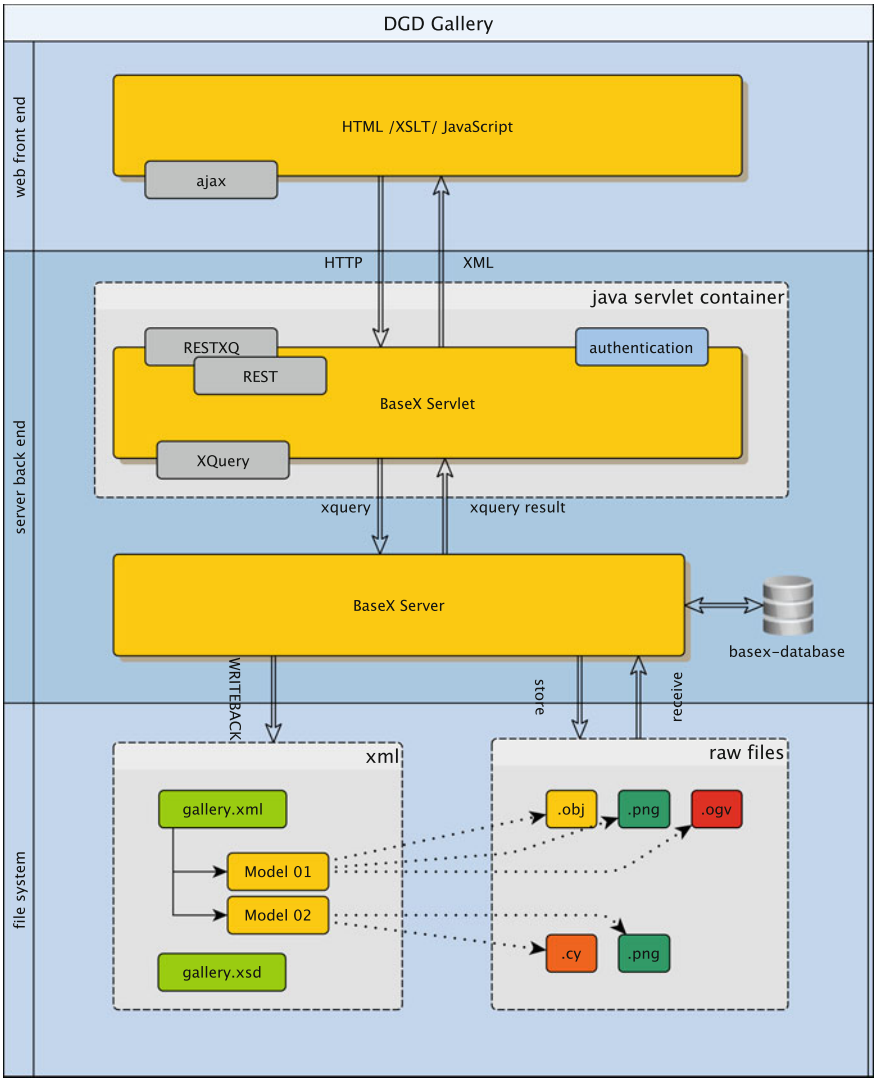


**Fig. 8** System architecture of the DGD GALLERY. On the file system level we store XML model data and data files. A BASEX server manages the read/write access to the XML documents and data files. It maintains an instance of a document database to optimize access to the XML data. At the same time a BaseX servlet provides a REST API [29] to connect the HTML/Javascript web front end of the gallery. It runs inside an Apache Tomcat servlet container executed within an Apache HTTP web server. The front end uses AJAX techniques and XSLT to create an interactive application using the API provided by the application server

- **Database framework agnostic storage** (while still using a database): The XML and any binary data are stored and handled using the XML document database BASEX [6], see Sect. 5.1. This gives us low access times (for data cached in main memory) and a transparent mechanism for permanent storage on the server's file system. The binary data files of the model's media objects are stored next to the XML data and linked appropriately, see the file system section in Fig. 8.
- **Separation of data and presentation**: We separate our application into a back end (on the server for database management only) and a front end (creating a HTML representation on the client machine). The BASEX database already provides the means for a complete implementation of the back end via XQUERY. This includes the specification of a REST API [29], which is a standard way to define the communication interface between server and client in the internet. The API returns XML or binary data in response to specified HTTP requests from the front end. XML is already close to HTML, while still not carrying explicit information on the visualization. This allows for the easy generation of multiple presentations from just one XML. In Sect. 5.3 we elaborate on the front end, which is based an XSLT, JavaScript and AJAX [31].

## 5.1 *XML Based Backend and the XML Document Database* BASEX

We use the established XML document database BASEX for storing our data. This automatically provides us with permissions, versioning, and life-cycle management for the models. BASEX runs on any Java application server. We use Apache Tomcat 7, see [32].

BASEX allows for the implementation of (web) applications using the XML query language XQUERY, see [33]. It combines the database access and application server logic implementation into one language. Additionally BASEX can be used to implement a REST API via RESTXQ, which is a set of XQUERY annotations for handling HTTP requests and generating HTTP responses [34], see Listing 1.

```
1  (:~
2  : REST API function to create a new model.
3  :
4  : @param $title mapped to POST parameter title, the title of the
       new model
5  : @param $user optional user name if no session can be inferred by
        the server
6  : @param $pass optional password
7  :)
8  declare %rest:POST
9  %rest:path("/createmodel")
10 %rest:form-param("title", "{$title}")
11 %rest:form-param("user", "{$user}", "dummy_id")
12 %rest:form-param("pass", "{$pass}", "")
13 %output:method("text")
14 %updating function api:createModel(
15     $title as xs:string,
16     $user as xs:ID,
```

```
17      $pass as xs:string
18  ) {
19      let $user := user:checkUser($user, $pass)
20      return model:createModel($user, $title)
21  };
```

**Listing 1** XQuery with RESTXQ annotations. The function api:createModel defines the web API function to create a model for a specified user and title string. The RESTXQ annotations, lines 8–13, define the REST interface of the server. User permissions are checked, line 19, and the corresponding database function to create a new model is called, line 20. User credentials (`$user`, `$pass`) are optional parameters and are transmitted using HTTPS API calls. Once logged in we use session cookies to authenticate users.

Generally, all API calls to our back end have to be authenticated. Either a username and password pair, or a session-cookie has to be provided along each request. The front end implementation uses session-cookies, which are obtained by an authenticated call to the login API function. A user's password is stored in the form of a salted bcrypt hash to provide protection against password recovery through an attacker in case of a server breach, see [35].

## *5.2 A Fail-Safe Release and Migration Process*

While a project like the DGD GALLERY evolves the precise technical requirements for the database are likely to change. This means that old versions will have to be migrated into new ones. We implemented a release process for new versions of the web application and its data using Apache ant [36]. We use XSLT 2.0 and XML Schema to define and validate database migrations [37].

In principle this allows for more general migrations than just XML to XML conversions between different schema versions of the database. We can envision scenarios in the future where XML may turn into a legacy format and will be replaced by a more general versatile format. With XSLT we can also convert our XML data into arbitrary text based formats allowing for a final conversion into file formats entirely different from XML.

## *5.3 A JavaScript Web Front End*

The standard way to enter a new model into the DGD GALLERY is through our web front end. This part of the application is completely separated from the back end, relying only on the REST API to BASEX for communication.

We use the AJAX scheme of web application development. The application, with its HTML, XSLT, and JavaScript components, is initially loaded from the server. The access to the database is organized as HTTP connections via JavaScript. Once XML model data has arrived from the server we process it with XSL Transformations [37]

to provide dynamic HTML and JavaScript for each client. We use the SaxonCE XSLT JavaScript framework to execute XSLT 2.0 in the browser, see [38].

Media renderers are provided for several common media formats, see Sect. 4.2. In the case of images and videos we are relying on the standards built into HTML5. We have support for the web capabilities of Cinderella to allow for interactive content [19, 39]. For the web browser in particular we use CindyJS [40], an open source JavaScript variant of Cinderella that aims to be compatible with Cinderella.

**Acknowledgments** This research was supported by DFG SFB/TRR 109 "Discretization in Geometry and Dynamics".



## References

1. Bates, D.J., Hauenstein, J.D., Sommese, A.J., Wampler, C.W.: Numerically solving polynomial systems with Bertini, Software, Environments, and Tools, vol. 25. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (2013)
2. Barakat, M., Lange-Hegermann, M.: Gabriel morphisms and the computability of Serre quotients with applications to coherent sheaves (2014). Preprint arXiv:1409.2028
3. Hong, H., Yap, C. (eds.): Mathematical software–ICMS 2014. Lecture Notes in Computer Science, vol. 8592. Springer, Heidelberg (2014)
4. Hales, T., Adams, M., Bauer, G., Dang, D.T., Harrison, J., Hoang, T.L., Kaliszyk, C., Magron, V., McLaughlin, S., Nguyen, T.T., Nguyen, T.Q., Nipkow, T., Obua, S., Pleso, J., Rute, J., Solovyev, A., Ta, A.H.T., Tran, T.N., Trieu, D.T., Urban, J., Vu, K.K., Zumkeller, R.: A formal proof of the Kepler conjecture (2015). Preprint arXiv:1501.02155
5. DFG verabschiedet Leitlinien zum Umgang mit Forschungsdaten. http://www.dfg.de/foerderung/info_wissenschaft/2015/info_wissenschaft_15_66/
6. BaseX Team: BaseX. The XML database (2014). http://basex.org
7. The Geometry Center—Geometry Reference Archive (2000). http://www.geom.uiuc.edu/docs/reference/
8. Hoffman, D., Hoffman, J., Weber, M.: The Scientific Graphics Project (1998–2004). http://www.msri.org/publications/sgp/SGP/
9. Eppstein, D.: The Geometry Junkyard. https://www.ics.uci.edu/~eppstein/junkyard/
10. Bohle, C., Loose, F., Schmitt, N., Heller, S.: GeometrieWerkstatt. https://www.math.uni-tuebingen.de/ab/GeometrieWerkstatt/
11. Imaginary | open mathematics. http://imaginary.org/
12. The SymbolicData project. http://wiki.symbolicdata.org/
13. EG-Models. http://www.eg-models.de

14. Joswig, M., Polthier, K.: EG-Models—a New Journal for Digital Geometry Models. In: Borwein J., Morales M., Polthier K., Rodrigues J. (eds.) Multimedia Tools for Communicating Mathematics, pp. 165–190. Springer (2002)
15. Bobenko, A.I., Hoffmann, T., König, B., Sechelmann, S.: S-conical minimal surfaces. Towards a unifying theory of discrete minimal surfaces (2015)
16. Schramm, O.: Circle patterns with the combinatorics of the square grid. Duke Math. J. **86**, 347–389 (1997)
17. Bobenko, A.I.: Discrete conformal maps and surfaces. In: Clarkson P., Nijhof F. (eds.) Symmetries and Integrability of Difference Equations, Proceedings of the SIDE II Conference, Canterbury, July 1–5, 1996, pp. 97–108. Cambridge University Press (1999)
18. Bornemann, F., Its, A., Olver, P., Wechslberger, G.: Numerical methods for the discrete map $z^a$ (in this volume)
19. Cinderella (2013). http://www.cinderella.de
20. Bobenko, A.I., Springborn, B.A.: Variational principles for circle patterns and Koebe's theorem. Trans. Amer. Math. Soc **356**, 659–689 (2004)
21. Lawson, H.B.J.: Complete minimal surfaces in $\mathbb{S}^3$. Annals of Mathematics **92**(3), 335–374 (1970)
22. Bobenko, A.I., Sechelmann, S., Springborn, B.: Discrete conformal maps: Boundary value problems, circle domains, Fuchsian and Schottky uniformization (In this volume)
23. Oberknapp, B., Polthier, K.: An algorithm for discrete constant mean curvature surfaces. In: Hege, H.C., Polthier, K. (eds.) Visualization and Mathematics, pp. 141–161. Springer, Berlin Heidelberg (1997)
24. Maclagan, D., Sturmfels, B.: Introduction to Tropical Geometry, Graduate Studies in Mathematics, vol. 161. American Mathematical Society, Providence, RI (2015)
25. Herrmann, S., Jensen, A., Joswig, M., Sturmfels, B.: How to draw tropical planes. Electronic J. Combin. **16**(2), R6 (2009–2010). http://www.combinatorics.org/ojs/index.php/eljc/article/view/v16i2r6
26. Gawrilow, E., Joswig, M.: `polymake`: a framework for analyzing convex polytopes. In: Polytopes—combinatorics and computation (Oberwolfach, 1997), *DMV Sem.*, vol. 29, pp. 43–73. Birkhäuser, Basel (2000)
27. The MathJax Consortium: Mathjax. https://www.mathjax.org
28. creativecommons.org: CC BY-SA 4.0. http://creativecommons.org/licenses/by-sa/4.0/
29. REST—Representational State Transfer. https://en.wikipedia.org/wiki/Representational_state_transfer
30. W3C: XML Schema (2004). http://www.w3.org/TR/xmlschema-0/
31. Ajax programming. http://wikipedia.org/wiki/Ajax_(programming)
32. Apache: Apache Tomcat (2015). https://tomcat.apache.org
33. W3C: Xquery 1.0: An xml query language (2010). http://www.w3.org/TR/xquery/
34. BaseX Team: RESTXQ documentation web page. http://docs.basex.org/wiki/RESTXQ
35. Provos, N., Mazières, D.: A future-adaptive password scheme. Proceedings of the Annual Conference on USENIX Annual Technical Conference. ATEC '99, pp. 32–32. USENIX Association, Berkeley, CA, USA (1999)
36. Apache: Apache ant (2012). http://ant.apache.org/
37. W3C: XSL Transformations (XSLT) version 2.0 (2007). http://www.w3.org/TR/xslt20/
38. Saxonica: Saxon-CE (Client Edition) (2015). http://www.saxonica.com/ce/index.xml
39. Richter-Gebert, J., Kortenkamp, U.: The Cinderella.2 Manual: Working with The Interactive Geometry Software. Springer (2012)
40. CindyJS—A JavaScript framework for interactive (mathematical) content. https://github.com/CindyJS