CindyJS

Mathematical visualization on modern devices

Martin von Gagern[1], Ulrich Kortenkamp[1], Jürgen Richter-Gebert[2] and Michael Strobel[2⋆]

[1] University of Potsdam, Germany
{gagern,ukortenk}@uni-potsdam.de,
http://www.math.uni-potsdam.de/professuren/didaktik-der-mathematik
[2] Technical University of Munich, Germany
{richter,strobel}@ma.tum.de,
http://www-m10.ma.tum.de

The final publication is available at link.springer.com
doi:10.1007/978-3-319-42432-3_39

Abstract. The *CindyJS* Project brings interactive mathematical visualization to a broad variety of devices. Using projective geometry, homotopy methods and well tuned algorithms the CindyJS project is one of the first real time capable software projects in this field that at the same time approaches high-level mathematical descriptions and performance.

Keywords: dynamic geometry, interactive visualization, projective geometry, homotopy methods, web technology

1 Introduction

Visualization and real-time interactive simulation play an important role both in mathematical research and in mathematical communication. The *CindyJS* Project aims at the development of a software platform and its mathematical foundation that allows a versatile and fast prototyping of mathematical experiments and visualizations which can be used for research and demonstration. The project attacks both the mathematical and the software related aspects of such a platform. In particular, the system should be usable as a flexible authoring system for providing mathematical content that can run in contemporary web browsers, taking advantage of modern hardware and software technologies.

To understand the relevance and challenges of the creation of such a visualization system in the current decade one has to take recent developments in the landscape of browser based interaction possibilities into account. The recent past showed a dramatic change of possible environments for such a general math

⋆ The authors 1, 3 and 4 were supported by the DFG Collaborative Research Center TRR 109, "Discretization in Geometry and Dynamics". The authors 1 and 2 were supported by the project "M C Squared" which has received funding from the EU 7th Framework Programme (FP7/2007-2013) under grant agreement no. 610467.

visualization system. One of the major achievements of the *Cinderella* platform as developed by Kortenkamp and Richter-Gebert was the ability to provide interactive math-related content via webpages [**?**]. With this philosophy, in recent years extensive collections of interactive applets have been created and used as tools for teaching and research [**?**,**?**] as well as for communication to the general public. However, the underlying technology of *Cinderella* on the web was Java in browsers, which became increasingly difficult to use due to security concerns. Recently, Oracle even announced the discontinuation of the Java Plugin.

As a consequence to the gradual demise of Java Applet usability, a new project called *CindyJS* was started. Its aim is allowing the creation of such mathematical visualizations and embedding these into web pages using modern plugin-less web technology like JavaScript, HTML5 and WebGL. The implementation of such a system poses several challenges, and while some of them have been resolved already, others still remain open and challenging. By now, the present prototype is already being used in production for the next generation of interactive content.

CindyJS aims to be a viewer for interactive mathematical content (generated by *Cinderella* or by explicit coding) in modern web browsers. In particular, mobile devices like cell phones and tablet computers were included as target platforms. The *CindyJS* project started in late 2013 and has already achieved about 80% compatibility to *Cinderella*: an implementation of an interpreter for the scripting language, the implementation of the physics simulation engine, and the geometry kernel are already usable for every day work. All of these have been implemented with extensive optimizations for fast real-time interaction. Special care was paid to software-ergonomic aspects in relation to usability on touch devices. In particular, the experiences collected in the project *iOrnament*, as described in [**?**], influenced the design of *CindyJS*. Like *Cinderella*, *CindyJS* is intended to be an open-ended project.

2 Project Guidelines

Our system has intentionally been written and designed from scratch for several reasons. Based on the experience of prior projects, like the Java version of *Cinderella*, the major development guidelines are:

- Performance: A detailed performance analysis of implementation patterns was preceding the core development to avoid performance bottlenecks. Cross compiling techniques, as used in other visualization oriented projects, lead to poor execution speed which makes these less suitable for advanced examples.
- Extendability: It is very easy to add additional functionality to the system, while still maintaining global consistency (also under mathematical aspects).
- Interoperability: The system has thin and lightweight interfaces to communicate with other software components in the browser. By this it is relatively easy to include plugins and third party frameworks.

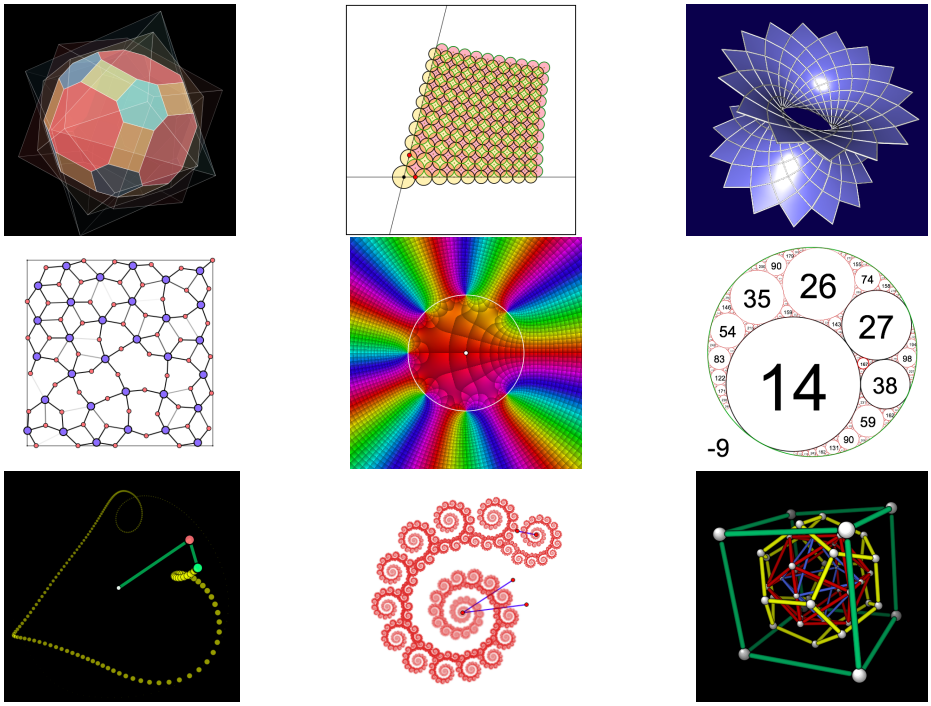

Fig. 1. Snapshots of interactive animations with *CindyJS*: Different applications of the *CindyJS* viewer, demonstrating its versatility. All these images, ordered left to right, are snapshots of dynamic interactive visualizations. (a) Intersection of polytopes (Deutsches Museum), (b) z^α grid (DGD Gallery), (c) Discrete CMC Surface (DGD Gallery), (d) Crystallization processes (MiMa, Oberwolfach), (e) Complex function plot, (f) Apollonean gasket (Mathe-Vital), (g) Double pendulum (Deutsches Museum), (h) Fractal (with WebGL support), (i) Nested polyhedra (ix-quadrat). More examples can be found on [**?**].

– Mathematical expressiveness: Many commonly used mathematical functionality is already included in the system as primitive operations. The scripting language allows for a high level implementation of mathematical concepts.

– Mathematical consistency: Implementing a visualization system for mathematical contexts is often preceded by a (rather subtle) modeling step in which a mathematical realm is mapped to algorithmic representations. One of the biggest achievements of the development of *Cinderella* and *CindyJS* was the capability to resolve ambiguities in geometric constructions by expanding the domain of $\mathbb{RP}^2$ using complex detours.

3 Architecture

Over the last two years a first prototypical implementation of the system has been created that contained many of the necessary key components and adheres to the above design principles. It inherits some features of the overall architecture of *Cinderella*, however it is designed to be even more modular and open for interoperation with other programs. Among the main components implemented so far there are a version of the scripting language *CindyScript*, a geometry kernel based on projective geometry, a first implementation of tracing strategies for homotopy continuation of geometric configurations, a basic physics engine, a 3D viewer and many other components. For an overview see Figure **??**.

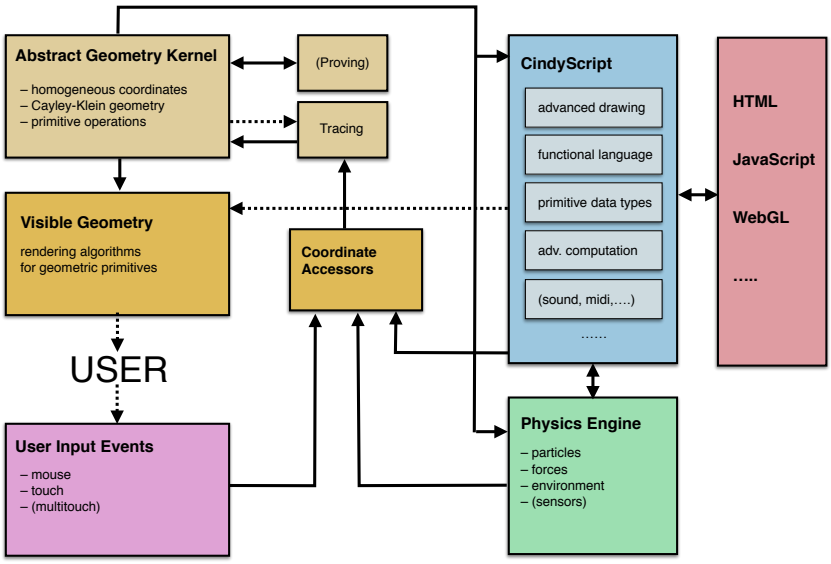

Fig. 2. Current top-level architecture of *CindyJS*

4 Geometric Primitive Operations and Tracing

All geometric operations in *CindyJS* are strictly based on principles of projective geometry and Caley-Klein geometries. This ensures a consistent treatment that has to take care only of a minimum number of special cases. It also allows for an easy generalization to other geometries like for instance hyperbolic geometry. Most of the concepts are explained in [**?**], for examples see Figure **??**. *Cinderella* was the first software to resolve the problem of continuity in presence of ambiguous operations in geometric constructions like the intersection of a circle

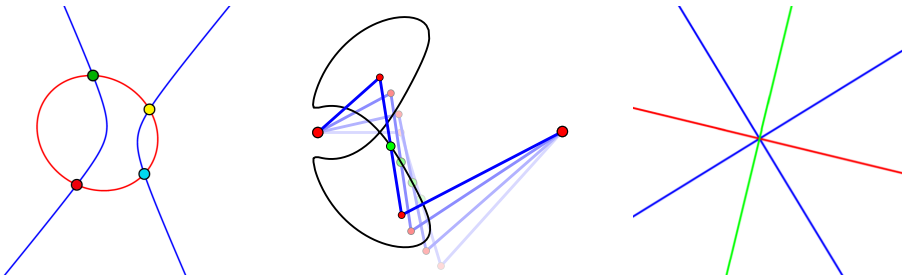

Fig. 3. Geometric operations: intersection of conics, traced locus and angle bisector

and a line. Singularities which naturally arise are resolved via complex detours. We won't go into details here, but a short example may illustrate the core idea. Assume we are moving a free element of the construction in such a way that two points of intersection merge into a single one before becoming distinct again (and perhaps complex-valued). At the point of the singularity, where the two points became one, we can no longer tell them apart. But by choosing a complex parametrization of the movement, we can move from the situation before the singularity to that after the singularity in such a way that the points remain distinguishable at all times. An instance of locus generation using this tracing technique can be found in Figure **??**, and we refer to [**?**] for further details.

5 *CindyScript* – Programming on a Napkin

Have you ever explained something to another mathematician, while sitting in a loud pub and having nothing but a pen and a napkin? *CindyScript* is the programming equivalent of the napkin. In other words, in *CindyScript* concepts should be expressible in sketchy, sometimes non-formal, nevertheless complete and most of all understandable way. It should be easy to write down simple concepts. Close to what you have in mind, also with respect to notation. It should be flexible and forgiving with respect to small glitches. Code should be easily explainable to non-experts. The size of a napkin is limited. *CindyScript* has (originally) never been intended to create huge software systems. It is not really a programming language for programmers, instead it is more a programming language for people who have little to no programming experience, but perhaps have some mathematical background. At the same time it is not an educational tool for "learning how to program" (at least it was not intended to be). It is a language in which small and easy things should be expressible in short an easy terms. In a sense, it is a "special purpose language" that on purpose avoids concepts that are complicated to explain to a non-expert. The core of the sunflower example in Figure **??** is essentially the following:

```
repeat(500,i,
    w=i*pi/180*(137.508+B.x*0.5);
    p=A+(cos(w),sin(w))*0.3*sqrt(i);
    draw(p,size->sqrt(i)*.4,color->hue(i/34));
);
```

These lines should show exemplary the easiness and the power of expression that comes with *CindyScript*. In particular *CindyScript* has dynamic typing and internal objects (like numbers, vectors, matrices, etc.) that are close to the mathematical realm. For further details we refer to the *Cinderella 2* handbook [**?**].

6 Plugins

CindyJS was designed with extendability and interoperability in mind. This enables us to attach plugins easily which extend the features of *CindyJS* without bloating the core functionality. This keeps the essence of our project swift and lightweight. Details of some plugins will be discussed in separate articles [**?**,**?**], we just highlight some of them here.
Cindy3D is our OpenGL binding to *CindyScript*. Employing WebGL, Cindy3D is used to display 3D content using the GPU.
CindyGL gives the user access to the GPU through WebGL in *CindyScript*. While Cindy3D serves for rather classic displaying purposes, CindyGL provides access to the GPU fragment shader in *CindyScript*, which can be used for colorplots and advanced custom image manipulation. It aims to overcome technical obstacles by integrating a pipeline which translates easy-to-write *CindyScript* code into highly parallelized shader code driving GPU calculations.
A version of *TEX* was also integrated in our system. After some performance tests we decided to drop MathJax and switchted to KaTeX [**?**]. With some modifications to KaTeX, the most common tasks using TEX are computed in real time.
We provide some selected examples in Figure **??**.

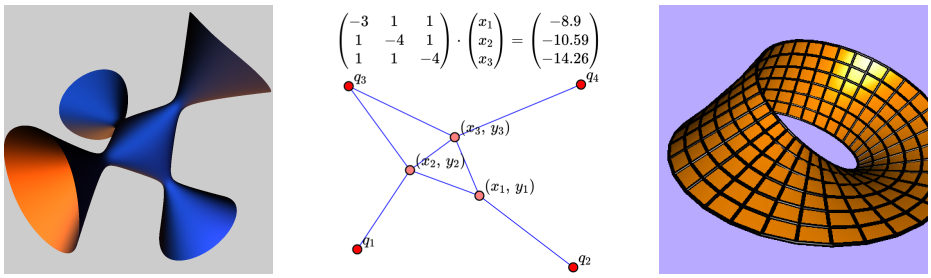

Fig. 4. *CindyJS* plugins: CindyGL raycaster, embedded TEX, Cindy3D Möbius strip

7 Technical Aspects

Modern browsers are a blessing and a curse at the same time. They are very well attuned for displaying content and since HTML5 broadened their capabilities introducing features like touch input or drag and drop. Also the execution speed of JavaScript increased dramatically over the years which is crucial for numerics. We implemented a feature-rich linear algebra package from scratch, which was pioneering work in complex arithmetics (for browsers). When possible, we used typed arrays in order to improve performance. Another crucial point was the embedding of *CindyScript* into HTML and the browser environment. We employ `<script>` tags which are directly written into the HTML code and will be interpreted by our libraries. Figure **??** shows a full example of the integration.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sunflower</title>
        <script type="text/javascript" src="Cindy.js"></script>
    </head>
<body>
<script id='csmove' type='text/x-cindyscript'>
        repeat(500,i,
                w=i*pi/180*(137.508+B.x*0.5);
                p=A+(cos(w),sin(w))*0.3*sqrt(i);
                draw(p,size->sqrt(i)*.4,color->hue(i/34));
        );
</script>

<script type="text/javascript">
    var gslp=[ {name:"A", kind:"P", type:"Free", pos:[0,0]},
               {name:"B", kind:"P", type:"Free", pos:[0,9]}];

    createCindy({canvasname:"CSCanvas",
                 movescript:"csmove",
                 geometry:gslp});
</script>
<canvas id="CSCanvas" width=500 height=500></canvas>
</body>
</html>
```

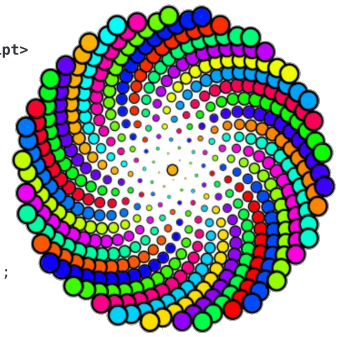

Fig. 5. Integration of *CindyJS* in HTML5

8 *CindyJS* in the wild, a selection

CindyJS is already suitable for every day work and is used in a large variety of places. We mention just a few of them:

Mathe-Vital The teaching platform Mathe-Vital focuses on interactive mathematical content at university level. Currently it consists of roughly 500 applets for scientific education. Within the next few years the platform will be entirely migrated to the *CindyJS* viewer.

Bruchrechnen! Funded by the Heinz Nixdorf Stiftung and in collaboration with the TUM School of Education, we are currently developing an interactive schoolbook about fractions. The schoolbook contains many interactive exercises almost all of which are realized within the *CindyJS* framework.

Teach@TUM The BMBF funded "Qualitätsoffensive Lehrerbildung" supports a project for the creation of structures and material to increase the quality of teacher education. In this context the *CindyJS* viewer is a major component for interactive teaching materials and micro laboratories.

ix-quadrat All interactive exhibits in our campus maths exhibition ix-quadrat are nowadays realized based with *CindyJS*. They are of particularly high complexity and form benchmark cases for much of the functionality of *CindyJS*.

M C Squared This EU-funded project for the creation of creative interactive content allows the use of *Cinderella* as a widget factory. It automatically converts the resulting widgets to *CindyJS* if the user is using the HTML5 player, e.g. on a tablet.

Mathe-Werkstatt The project Mathe-Werkstatt (Leuders, Prediger et al.) provides interactive material for all grades of mathematical school education. Over the last few months the team of authors there started to also support a *Cinderella/CindyJS* based environment. *CindyJS* formed an essential component in the design of this course.

DGD Gallery *CindyJS* is also used as a visualization component in the context of the SFB TR109 gallery. Besides 2D interaction *CindyJS* is also used as a 3D viewing engine [**?**].

9 Conclusion

Overall the *CindyJS* project aims to be one of the first medium to large scale mathematical visualization projects based on an HTML5 framework. Our project is licensed under the Apache 2 license and can be obtained from `https://github.com/cindyjs`.