The Intergeo File Format in Progress

Miguel Abánades[1], Francisco Botana[2], Jesús Escribano[3], Maxim Hendriks[4], Ulrich Kortenkamp[5], Yves Kreis[6], Paul Libbrecht[7], Daniel Marques[8], Christian Mercat[9]

[1] CES Felipe II - UCM, Aranjuez, Spain
[2] Universidad de Vigo, Spain
[3] Universidad Complutense de Madrid, Spain
[4] Eindhoven University of Technology, The Netherlands
[5] University of Education Karlsruhe, Germany
[6] University of Luxembourg, Luxembourg
[7] DFKI GmbH, Saarbrücken, Germany
[8] Maths for More (WIRIS), Barcelona, Spain
[9] I3M, Université Montpellier 2, France

Abstract. In this paper we describe the ongoing effort to specify a common file format for Interactive or Dynamic Geometry Systems (DGS). Our approach is based on the OpenMath standard, and uses its flexible extension mechanisms like Content Dictionaries.
We discuss the various design decisions, the Content Dictionaries that have been defined, as well as open questions to be resolved.

1 Introduction: The Intergeo Project

Interactive geometry is one of the most well known family of computer-based tools to support teaching of mathematics by means of personal explorations.

Intergeo (`http://inter2geo.eu`) is an eContent*plus* European project dedicated to the sharing of interactive geometry constructions across boundaries. It enables teachers and pupils all over Europe to share resources and experiences as tools for teaching, learning, and research.

Educational contents that were hard to access shall be made available, tagged with relevant topics and competency based metadata and categorised according to curricula, they are searchable and easily (re-)usable by everyone. It is our goal to offer them in a common interoperable format that this article describes.

For more information about the project, we refer to its website and the documentation available there, as well as [1,2].

A wide variety of DGSs exists. Before the Intergeo project, each system used incompatible proprietary file formats to store its data. Thus, most of the DGS makers have joined to provide a common file format that will be adopted either in the core of the systems or just as a way to interchange content.

The Intergeo file format is a file format designed to describe any construction created with a Dynamic Geometry System (DGS). Dynamic Geometry Systems, also called Interactive Geometry Systems, are programs that are used to experiment with geometric objects. A construction, a drawing consisting of geometric elements, is displayed to the user. But it is not just a still picture. The construction is interactive, it reacts to user's input, who can move some of the elements with the mouse pointer. The whole construction is then recomputed according to the defining geometric relationships. For example, the circumcircle of a triangle could follow the three vertices of the triangle wherever they are dragged.

The Intergeo file format is based on three design decisions: the packaging as an archive with particular files inside it, the separation of the elements part describing the (static) initial geometry from the constraints part where the geometric relationships are expressed using OpenMath, and the use of OpenMath Content Dictionaries (CDs) to describe the elements and the constraints. These CDs are provisionally called the i2geo CDs.

This paper describes the ongoing specification effort of the aforementioned Content Dictionaries and the progress on implementation.

2 Review of pre-existing CDs

The primary and ambitious goal of OpenMath (`http://www.openmath.org`) is to develop a standard for representing mathematical objects with their semantics. The fact that its original designers were mainly developers of computer algebra systems lead to little attention being paid to geometry. The geometry-related Content Dictionaries one can find at the OpenMath website at the time of writing are bundled in a group called `plangeo`. There are six of them, `plangeo1,...,plangeo6`. The symbols defined in these CDs deal with planar Euclidean geometry and with generating polynomial systems from geometric configurations. These `plangeo` CDs were designed at Eindhoven University Of Technology as part of a project for automatically proving theorems sketched in the DGS Cinderella. Although Cinderella has an efficient randomized prover, its proofs cannot be verified. The goal of the project was to let Cinderella communicate with GAP by means of OpenMath, in order to use bracket algebra to obtain sound proofs of geometric theorems [3].

Another use of the `plangeo` CDs has been reported in [4]: constructions in Cabri, The Geometer's Sketchpad and Cinderella dealing with geometric loci, proving and discovering, are rewritten in OpenMath and exported to Mathematica and CoCoA for algebraic manipulation. As far as we know, no other use of the `plangeo` CDs has been published.

3 The Intergeo file format: design decisions

The file format of Intergeo is based on three major design decisions, which we explain below: the choice of zip-packaging, the choice of a constructions-based approach as opposed to a constraints-based approach, and the choice of OpenMath as semantic infrastructure. This paper provides a summary of the details described in [5].

3.1 Packaging

The Intergeo files, just as many other formats that have appeared recently, are ZIP archives containing several files. The most important file is the central file `intergeo.xml`; optional files can be compressed containing media and style elements which should be detached from the construction.

The `intergeo.xml` file encodes the initial positions, in XML, and the constraints, in OpenMath being made of references to the Content Dictionaries. See [5] for a detailed specification of the archive. On the right you see an example archive listing.

```
construction/
construction/intergeo.xml
construction/preview.svg
construction/preview.png
metadata/
metadata/i2g-lom.xml
resources/
resources/photo-jump.jpeg
```

An example content of `intergeo.xml` is given in figure 1 where one can see the usage of construction and constraint elements which we explain below.

3.2 Constructions-Based Description

The objective of the file format specification is providing a semantics of interactive geometry which should be understandable by all DGS implementors as well as further systems such as proof assistants. The mathematical semantics is, thus, important. In this section, we describe the chosen conceptual approach while the next describes the current concrete specification of OpenMath symbols that has been achieved.

Dynamic Geometry Systems deal with sets of geometrical objects that have certain relations. We call such a set of objects with given relations a *configuration.* All objects are part of some underlying space, for example the euclidean plane. In principle, if nothing else is said about them, objects can move around freely in this space. Relations then specify *constraints* on the movement of these objects.

Example 1. Two points P and Q, together with a line l; there is the following constraint:

line l is incident to both points P and Q.

Example 2. A circle Γ, a point P, a line l and the following constraints:

P is on l
l is tangent to Γ
the distance of P to the center of Γ is 10.

We can make the simple observation that the constraints do not determine the positions of the objects uniquely. This causes multiple problems that lie at the heart of dynamic geometry.

```
<construction>
  <elements>
    <point id="P">
      <homogeneous_coordinates>
        <double>2</double>
        <double>5</double>
        <double>1</double>
      </homogeneous_coordinates>
    </point>
    <line id="l">
      <homogeneous_coordinates>
        <double>7</double>
        <double>3</double>
        <double>-29</double>
      </homogeneous_coordinates>
    </line>
    <point id="Q">
      <homogeneous_coordinates>
        <double>5</double>
        <double>-2</double>
        <double>1</double>
      </homogeneous_coordinates>
    </point>
  </elements>
  <constraints>
    <line_through_two_points>
      <line out="true">l</line>
      <point>P</point>
      <point>Q</point>
    </line_through_two_points>
  </constraints>
</construction>
```

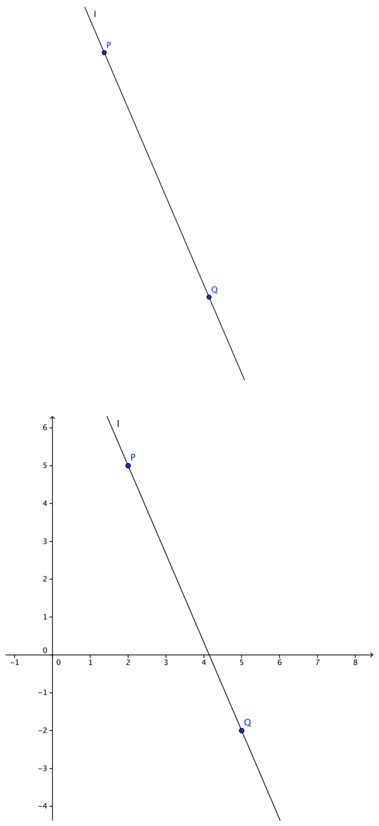

Figure 1: The content of an `intergeo.xml` for a simple construction of two points and one line through them. On the right, two possible (default) graphical representations of it: with or without axes; an important distinction between several interactive geometry systems which decide to present the geometry within a coordinate system or not per default

There is the problem of how to create an instance of the configuration. We say this has to do with the *static* aspect of the configuration. In Example 1, the points P and Q could still lie anywhere on the line l as it stands. For any instance, we must specify where l, P and Q should be. But once we have specified one, the other two are not completely free anymore. This particular example is not hard. And Example 2, although more difficult, is still doable. But in general, it is very difficult to give any particular solution for a set of constraints. There is not even a quick method to decide whether there are any instances: a set of constraints could be too restrictive and leave none.

Second, there is the dynamic behaviour of a configuration, caused by the freedom still left by the constraints. In Example 1, what should the user be able to move? May the line be picked up and translated or rotated in its entirety, the points being translated and rotated with it? Can the user only move one of the two points, the line being adjusted accordingly? Constraints of a strictly classical geometrical nature, such as the ones stated above, do not say anything about this behaviour. For the approach of a DGS, this is not enough.

A natural way to shed light on both these problems is a more precise specification of how the objects depend on each other. We could stipulate which objects are *free*, meaning that they can be varied over the whole range of possibilities in the underlying space (think of the plane) by the user. We would then proceed saying which objects depend only on the free objects, which ones depend only on these new objects and the free objects, etcetera. Such a specification is called a *construction.* It allows a DGS to rapidly create instances or decide that there are none. It also enables a DGS to give more consistent dynamic behaviour: objects are only movable insofar as they still have some degrees of freedom left, supposing the objects they depend on are kept fixed. The behaviour for all different cases (e.g. a line through a fixed point) can be decided in advance. Other objects dependent on the object being varied have to change as well, and this still leads to decision problems, but they are less severe. We could give a construction for Example 1 as follows:

Example 3. Two points P and Q, together with a line l, and the following construction:

$$\begin{gathered}\text{free\_point}(P)\\ \text{line\_through\_point}(l,P)\\ \text{point\_on\_line}(Q,l)\end{gathered}$$

The line l would then depend on where P is placed. That point could be varied freely. The line could then be rotated around P (and Q would most logically rotate with it), and while P and l are kept fixed, Q could still slide over the line. Note that such information could not be gleaned from the configuration.

It thus seems like a configuration might be too general to be practical, and we might be better off with a construction. We therefore decided to go with constructions. This decision implies less interoperability with constraint-based systems, since some of their resources will not be encodable into the format. But it ensures that construction-based DGSs will be able to interpret the resources, which they might not if we used configurations. Indeed, although some systems like Geometer's Sketchpad [6] and Geometry Expressions [7] take a constraint-based approach, most systems use constructions.

Another effect of the decision is the potential explosion of keywords. We have to distinguish between "line_through_point" and "point_on_line". This is in sharp contrast to configurations, where one relation "incident" would suffice. In general, if there are n different types of objects, the construction approach now forces n^2 different types of incidence on us. This means a more bloated specification of the file format. On the other hand, it is easier for software developers to parse constructions, so it saves trouble there.

3.3 Design Decisions: OpenMath

The advantage of using OpenMath [8] as opposed to a self-chosen XML-format lies in the fact that the use of a Content Dictionary makes for a flexible, open, and reusable standard whose mathematical rules can be described. First of all, the use of OpenMath enables INTERGEO to use other Content Dictionaries already in existence, so it saves development time. Second, other kinds of software that want to use the format in the future can combine it with other Content Dictionaries to enrich its expressive power.

The choice of OpenMath also lies in the documentation of the extensibility: although the INTERGEO consortium groups a wide majority of the implementors of interactive geometry software and the agreement they have reached, the file formats of each of these softwares will evolve. Thanks to the formal-mathematical-properties, new symbols can be used by one software with a fair chance that they will be also usable, mathematically correctly, by other softwares [9].

The OpenMath XML syntax specified in [8] is not used within the file-format because of the numerous restrictions on the constraints and elements parts: they make it easy to use a more expressive XML syntax which uses the element-name instead of the symbol name. A look to the sample of figure 1 will allow the reader to easily translate. The format is used, however, within the content-dictionaries to express both the examples and the formal properties.

4 Content Dictionaries: achieved set of symbols

As the title of this paper indicates, the Intergeo File Format is still work in progress. The Intergeo team is regularly discussing issues and trying to reach consensus on a substantial part of them. At the end of July 2008, a first version had been constructed that pertained to lines and points. Although outdated now, it can still be found at `http://svn.activemath.org/intergeo/Deliverables/WP3/D3.3/`. At the end of July 2009, a second version was constructed, which looks more or less like described in this section. The content-dictionaries are edited after more informal writing happening in the project's wiki and a discussion has taken place. They can be browsed from `http://svn.activemath.org/intergeo/Drafts/Format/cd/`.

The final version will be available at the end of June 2010. This will also mean that at that time, all software partners will have a working API that supports this format. For more about APIs, see section 8.

The second version of the file format mainly concerns itself with points, lines, (directed) line segments and rays, polygons, and conics.

First, in the *elements part* of an i2g file, all geometric objects in the construction are declared. This implies that the valid OpenMath symbols appearing there, the vocabulary of the `intergeo_elements` Content Dictionary, is also the collection of mathematical *types* that we work with. This setup therefore makes explicit what kind of objects we consider to be first-class citizens in a geometric construction. For now, the types we have are:

Point	Line	Linear_equation
Direction	Ray	Line_segment
Directed_line_segment	Polygon	Conic
Ellipse	Circle	Parabola
Hyperbola	Locus	

We note that some types are subtypes of other types, e.g. circles are ellipses, which are themselves conics, just like parabolas and hyperbolas. Subtyping gives rise to some specific problems which we are tackling at the moment. For example, we are pondering including artificial types like `Linear_object` for generalizing `Line`, `Ray`, `Line_segment` and `Directed_line_segment`, as well as `Object` to encompass all types.

Once all objects in a construction have been declared in the *elements part* of the i2g file, in the *configuration part* the geometric relations between the objects can be described. The description is done by predicates. Formally, this means that a geometric relation is cast in the form of a functional type, in the sense of type theory, with input the types of the relevant objects and output type `Bool`. For example, we might say that a point is constrained to lie on a previously defined line by writing

$$\texttt{point\_on\_line}(p, l)$$

with type

$$\texttt{Point} \times \texttt{Line} \to \texttt{Bool}.$$

Some predicates do not refer to geometric relations but to dynamic behavior. For example, an object can be specified to be freely movable, independent of anything else. Since the behaviour of a free object may depend on its type, we chose to explicitly include this type, whence we have `free_point`, `free_line`, etc.

We currently have the following list of constraints:

free_point	free_line
point_on_line	line_through_point
line_through_two_points	line_perpendicular_to_line
line_perpendicular_to_line_through_point	line_parallel_to_line
line_parallel_to_line_through_point	point_intersection_of_lines
line_angular_bisector	line_segment_by_points
carrying_line_of_line_segment	endpoint_of_line_segment
point_on_line_segment	directed_line_segment_by_points
starting_point_of_directed_line_segment	end_point_of_directed_line_segment
line_segment_of_directed_line_segment	ray_from_point_to_point
ray_from_point_in_direction	starting_point_of_ray
carrying_line_of_ray	direction_of_ray
point_on_ray	point_on_conic

5 Basic Requirements: A Wish List

From our experience with the OpenMath geometric symbols (`plangeo` CDs), and mainly from the discussion with the software partners of the Intergeo project, we have described a basic comprehensive list of geometric elements and functions that should be supported by the common file format. Of course, it is not a complete and exhaustive list of all the elements from all DGSs involved but rather the set including the most common elements and functions. The purpose of this wish-list is hence to serve as a beacon to lead the way towards a common ground. It consists of a limited number of basic elements, but more elements can (and must) be added in the future.

The list has two parts: Constructions and Functions. In the *Constructions* part, with 53 elements, we list the basic elements of a geometric construction, from the simplest ones (a free point) to the more complicated ones (for example, a geometric locus). We have identified different categories of elements. For example, in the category *Points* we can find *Free point*, *Point on line*, ...

In the *Functions* part, with 10 elements, we consider calculations that can be performed on a geometric construction, like computing the area of a triangle or the length of a segment; or transformations on a construction, like a rotation.

The following is a schematic version of the list in which the number of construction subtypes has been added for each type.

– Constructions
  - Points (17), Vector (1), Segment (1), Lines (15), Rays (3), Polygons (2), Circles (6), Conics (6), Locus (2)

– Functions
  - Angle, Area, Length, Distance, Reflection wrt a point, Reflection wrt a line, Reflection wrt a circle, Rotation by two lines, Rotation by a point and an angle, Traslation

6 Implementation progress, library, and test-suite

A wide variety of DGSs exists nowadays. Before this project, each system used incompatible proprietary file formats to store its data. The Intergeo file format aims to be the convergence of the common features of the current DGSs together with the vision of future developments and the opinion of external experts. Its final version, based on modern technologies and planned to be extensible to capture the flavour of different DGSs, could serve as a standard in the DGS industry.

The specification of the first version of the Intergeo file format has been released as [5] after intensive collaboration between DGS software developers and experts. It specifies only a restricted subset of possible geometric elements, which however lead to an agreement on the structure and basic composition of the format.

As soon as Version 1 of the file format got more concrete, some of the software developers started to investigate its practical usage by integrating it (partially) into their software (see `http://i2geo.net/xwiki/bin/view/About/I2GformatImplementations`). It was possible to move simple content between several of the packages in the project. The gained experiences influenced the further steps: the development of the next version of the file format is ongoing, will be released before summer and can be followed on `http://svn.activemath.org/intergeo/Drafts/Format/cd/`. The development of a common API for the file format (ClientAPI) has started and will soon be available to all developers to simplify their implementation work.

The need of a test suite emerged to check the compatibility level of the different DGSs and to be able to release compatibility certificates. The test suite is composed of several test cases for the different elements and/or constraints. Its result is one of the following four levels:

- cannot read the element/constraint technically (which should not happen)
- can read the element/constraint technically -without throwing exceptions etc.- so it can at least be ignored (for unsupported elements/constraints)
- can read the element/constraint and represent it internally, though maybe with wrong or missing semantics (partial support of the element/constraint)
- can read the element/constraint and represent it internally with correct semantics (full support of the element/constraint)

Furthermore another API (ServerAPI) has been defined to allow the Intergeo platform to display DGS resources. It allows retrieving important information like a preview image, the HTML fragment and the MIME type. It will also enable converting the construction to the Intergeo file format. Thus all committed constructions will be available in the common file format as soon as the corresponding DGSs will implement both APIs.

7 Mathematical Issues and Problems

While developing the file format we identified several issues and mathematical problems that have to be addressed. We will discuss some of them briefly here; for further discussions we refer to the mailing list archives of Intergeo Work Package 3, and we also invite others to join the discussion there.[10]

This section repeats parts of the Deliverable 3.3, and highlights the important aspects for the OpenMath community. Despite the fact that we are listing a lot of yet to be solved issues, we are confident that the first version of the Intergeo file format is capable of handling any design decisions that result from the following discussion. For a description of some of the underlying mathematical problems, we refer to [10].

7.1 More Elements and Polymorphism

We already specified a wish list for further elements and constraints in Section 5, that contains the most important geometric extensions. Still, there are many other objects that have to be expressible by the file format, more general ones like functions or numbers, as well as stylistic elements like text objects or images. A task for the next version of the Intergeo file format is to collect and specify all elements that are currently in geometry software.

Certain elements are very similar to others, e.g. segments and rays can be used instead of lines in many cases. Other examples are arcs of circles in comparison to plain circles. The next version of the Intergeo file format has to be able to handle such polymorphism, as well as symbols that have a variable number of arguments. An example for the latter is the definition of polygons by vertices.

Some objects can also be replaced by others in certain special cases, for example, a circle might degenerate to a line, or a conic might degenerate to two lines. Although currently no DGS uses these degenerations, this could be desirable for the future. A DGS might construct a parallel line to a degenerate circle through three collinear points — with our current specification and typing mechanism it is not possible to capture this in the file format. However, we request advise from DGS developers and users on this issue.

7.2 Ambiguity Resolving

Ambiguity Resolving is crucial in finding the correct positions of elements in stored construction after loading, also known as the persistent naming problem [11] from parametric CAD. Assume that an intersection point of two circles is used in a construction. If the two circles are moved into a tangent position, and then the construction is stored, then both intersections have the same coordinates and thus cannot be distinguished. If the circles are moved into a position where both intersection points can be distinguished, then it is essential to pick the correct intersection point.

[10] See `http://lists.inter2geo.eu/mailman/listinfo/wp3`

Most DGSs solve this problem by having an implicit order of multiple outputs. This order is dependent on the implementation details of the algorithms and cannot be part of a specification. Also, a point might switch branches later due to homotopy-conserving implementations. A detailed review of these problems can be found in [12]. This means that this approach cannot be used for a cross-software file format.

As soon as circle (or conic) intersections are introduced, we will have to find a solution to this problem.

7.3 Functions and Scripting

Almost all DGSs support some form of plotting graphs, defining element dependencies, or changing the style of elements dynamically by using functions that are defined symbolically. All of these use a different language to specify the functions, though many aspects are shared. The conformance to standards varies wildly between "OpenMath compliance" and "unspecified."

Right now it seems impossible to homogenize the various dialects. Actually, the translation from one language to the other can be done easily by humans if an automatic conversion fails, so we decided that for now all functions should be specified in the private sections of the file format. Each DGS may try to interpret the other function specification, of course, and store its own interpretation as well.

For this, we need a notion of "alternatives", which will be specified in an upcoming version of the Intergeo file format.

Another difficulty in dealing with functions is that some DGSs extend the notion of function to a general-purpose functional programming language. This proves that it is impossible to find equivalent functions algorithmically. Nevertheless, in many cases the translation is straightforward, and so it might be sufficient to use a heuristic approach.

One solution would be to use the API approach as described in Section 8. Actually, a DGS A could use the "function dialect" of another DGS B by asking the other system B to interpret the native parts that A cannot understand via the API. While this sounds very entangled, it may be the general solution to the specification problem we face here.

7.4 Mathematical Typesetting

Usually, mathematical typesetting is done with TeX [13], and a browser-compliant way is to use MathML [14]. DGS software uses both approaches, while the TeX implementation used is usually only a subset of the full TeX system as created by Knuth[11].

We could not agree on a definitive way to typeset formulae. Probably it would be a good idea to support MathML, but most of the DGS developers do not want to adopt it, as it seems. So this is currently unspecified and mathematical typesetting has to be specified in the private part of a construction.

[11] Some use the hoteqn library, others use custom implementations

7.5 Macro Constructions

So far there is no notion of *macro constructions* in the file format specification. We expect to treat them basically as subconstructions, and it is probably sufficient to add an additional `inmacro` attribute to the constraints and elements. However, we have postponed this matter until we have a more substantial set of examples.

7.6 Number Representation

Currently, the specification of coordinates uses the IEEE standard for doubles for historical reasons. While this is probably sufficient for most purposes, it lacks the ability to describe *real* coordinates, for example the irrational numbers π or $\sqrt{5}$. As there are constructions even in elementary geometry that require such numbers, it is desirable to be able to express them.

The OpenMath standard provides a means to specify all the real numbers likely to appear in normal applications of Interactive Geometry, but some implementation difficulties have been pointed out. For the time being, we will restrict the number representation to the IEEE standard, in particular due to the reason that no DGS so far uses another specification. This should not be the cause of severe problems, because the coordinates of dependent elements can be recalculated up to arbitrary precision by the DGS itself. If there is a need for other number representations, we will extend the mechanism. This will not need a major redesign, as all occurrences of numbers will just be replaced with real number values.

8 File Format and API Interaction

The obvious way to use the file format is to exchange files; save them in one DGS and load them in the other. While this is usually done manually, the file format is even more useful when it comes to automated exchange between software. Therefore, we are currently defining an application programming interface (API) that will make use of the file format at several places [15].

Basic interaction facilities of the API are loading and saving of files, both in native and Intergeo file format, which allows for creating server-side software that delivers files in the users' preferred format. Also, copy and paste operations will be described in terms of the Intergeo file format, as any copied part of a construction is in itself a construction.

We also allow for communication from and to non-DGS software, e.g. a CAS. Getting and setting values like objects' coordinates using a CAS opens a wide variety of applications.[12] In particular, most DGSs currently work with double

[12] As a reviewer points out it is a problem to send values like π to a CAS if there is no internal representation for it in the DGS. In practise it usually does not pose a problem, as only coordinates of free elements have to be sent, and there is no need to set a free element to exact real coordinates.

precision internally, which is not sufficient for research-grade use. As a primary concern of interactivity is real-time operation, it is not surprising that most software only uses a precision that can be handled in hardware. Using the API it is possible to externalize this calculations in situations where we can sacrifice time for exactness.

Many DGSs already use built-in facilities for algebraic (or better: analytic, as the focus is on doing calculations) manipulation. This is a major obstacle for compatibility, as all the dialects are different. Using the API it could be possible to use the DGS part of one software package, while using the CAS part of the other. Again, the core of the interaction is the clear specification of Intergeo elements.

Finally, we want to mention the pedagogical benefits of a standardized API. The integration of interactive exercises that can be checked by other software (like theorem proving systems) could be a huge step for technology-based teaching and learning of mathematics.

9 Outlook

This paper has presented the current progress on the development of the Intergeo file format towards interoperable interactive geometry: the choices made and the implementations achieved. We believe that the format will support sharing helped by the Intergeo platform which aims at breaking the other barrier of sharing interactive geometry: the barriers between educational regions.

As a more current preoccupation, the team of authors is involved into the maturation process with a strong hope to have a satisfactory format to indeed break many software barriers that split interactive geometry communities in too small chunks. We expect that it will renew a competition among dynamic geometry systems' shining by their exclusive features but providing the basic exchange format for the core of interactive geometry.

The common file format has attracted several interested parties thus far, all makers and users of softwares at the edge of interactive geometry, in such domains as algebraic geometry.

References

1. Kortenkamp, U., Blessing, A.M., Dohrmann, C., Kreis, Y., Libbrecht, P., Mercat, C.: Interoperable interactive geometry for europe – first technological and educational results and future challenges of the intergeo project. In: Proceedings of CERME 6, Lyon. (2009)
2. Kortenkamp, U., Dohrmann, C., Kreis, Y., Dording, C., Libbrecht, P., Mercat, C.: Using the intergeo platform for teaching and research. In Bardini, C., Fortin, C., Oldknow, A., Vagost, D., eds.: Proceedings of the 9th International Conference on Technology in Mathematics Teaching, Metz, ICTMT-9 (2009)
3. Roozemond, D.: Automated proofs using bracket algebra with cinderella and openmath. In: RWCA 2004: Proceedings of the 9th Rhine Workshop on Computer Algebra. (2004)

4. Escribano, J., Botana, F., Abánades, M.: Adding remote computational capabilities to dynamic geometry systems. Mathematics and Computers in Simulation (To appear)
5. Grothmann, R., Hendriks, M., Kortenkamp, U., Kreis, Y., Laborde, J.M., Libbrecht, P., Marquès, D.: D3.3: Common file format v1. Technical report, Intergeo Project (2008) See `http://i2geo.net/files/D3.3-Common-File-Format-v1.pdf`.
6. KCP Technologies Inc.: Geometer's sketchpad v4 (2008)
7. Saltire Software: Geometry expressions v1.1 (2008)
8. Stephen Buswell, Olga Caprotti, D.C.M.D.M.G., Kohlhase, M.: The OpenMath Standard, version 2.0. Technical report, The OpenMath Society (2004) Available at `http://www.openmath.org/`.
9. Davenport, J.H., Libbrecht, P.: The freedom to extend openmath and its utility. Journal of Computer Science and Mathematics **59** (2008) 1–19 NL=yes;RB=Jan.
10. Kortenkamp, U.: Foundations of Dynamic Geometry. Dissertation, ETH Zürich, Institut für Theoretische Informatik, Zürich (1999)
11. Marcheix, D., Pierra, G.: A survey of the persistent naming problem. In: SMA '02: Proceedings of the seventh ACM symposium on Solid modeling and applications, New York, NY, USA, ACM (2002) 13–22
12. Denner-Broser, B.: Tracing-Problems in Dynamic Geometry. PhD thesis, Freie Universität Berlin, Berlin (2008)
13. Knuth, D.E.: The TEXbook. Addison-Wesley, Reading, Massachusetts (1984)
14. Carlisle, D., Ion, P., Miner, R.: Mathematical markup language (mathML) version 3.0. World Wide Web Consortium, Working Draft WD-MathML3-20080409 (2008)
15. Kortenkamp, U., Kreis, Y.: D3.5: I2g api specification. Technical report, Intergeo Project (2009) To appear. See `http://i2geo.net/files/D3.5-API-specification.pdf`.