POSSIBLE IMPROVEMENTS
OF MODERN DYNAMIC GEOMETRY SOFTWARE

Đorđe Herceg [1], PhD, full professor, herceg@dmi.uns.ac.rs
Davorka Radaković[1], MSc., teaching associate, davorkar@dmi.uns.ac.rs
Mirjana Ivanović[1], PhD, full professor, mira@dmi.uns.ac.rs
Dejana Herceg[2], PhD, assistant professor, vuletic@uns.ac.rs

[1]University of Novi Sad, Faculty of Science and Mathematics, Trg D. Obradovića 4, 21000 Novi Sad, Serbia
[2]University of Novi Sad, Faculty of Technical Sciences, Trg D. Obradovića 6, 21000 Novi Sad, Serbia

Abstract

Contemporary education is starting to supersede the traditional one (teacher-to-student lessons) with technology-rich learning using various educational tools and a selection of materials that are effective, efficient and appealing to students. Dynamic Geometry Software (DGS) today is widely used in teaching and learning mathematical topics. Such kind of educational software can evolve in several ways, by either adding new features on the surface or by evolving the evaluation engine at its core. The implementation of a DGS needs to be straightforward and modular.

To achieve the evolution of a DGS core we have developed a programming framework for the Dynamic Geometry Software, SLGeometry, with a genericized functional language and the corresponding expression evaluation engine. Engine acts as a framework into which specific semantics is embedded in the form of code, annotated with metadata. An ordinary expression tree evaluator is transformed into an object-oriented one by this framework. Whilst other DGS are based on purely functional expression evaluators, our solution has the advantages of being more general, maintainable, understandable, easy to implement, and providing a natural way of specifying object properties in the user interface, minimizing typing and syntax errors. The modular approach enables independent development of subject-specific components, which are easily added to the evaluation engine in the form of plug-ins. The object-oriented nature of the framework enables development of self-contained units, such as objects and visual elements which encapsulate domain-specific semantic and present it to the user as virtual placeholders for real-life objects and notions.

In this paper we present several possible improvements of Dynamic Geometry Software, particularly having in mind the platform that we have implemented. Additionally we discuss benefits of these features and their influence on the users/students. The approach is tested on SLGeometry — our DGS platform, developed in C# on the .NET Framework.

Keywords: *Dynamic Geometry Software, Teaching, Component development.*

1. INTRODUCTION

Computers have been used for geometric visualization since the early days of establishing computer graphics as one of important areas in ICT. A distinct class of educational graphical software has appeared, i.e. the Dynamic Geometry Software (DGS). In DGS, a geometric drawing is represented by a set of expressions which represent geometric objects on a graphic display. The expressions, assigned to named variables, can be parametrized and referenced by each other. This way, relationships between objects in a drawing are established. When the user moves an object, all dependent objects have to be recalculated and moved. There are numerous DGS frameworks but among the most popular today are GeoGebra [1], Cabri [2], Cinderella [3] and The Geometer's Sketchpad [4].

The authors have been using GeoGebra and Mathematica [5] in everyday practice for the teaching of geometry, numerical analysis and geography for more than 15 years. From our experience, as well as from the students' feedback, we have identified several aspects of modern DGS which could be improved, and tested them on SLGeometry [6–9], an experimental platform for dynamic geometry, which we developed to test various concepts and approaches. This paper presents our ideas and observations for possible improvements of our platform.

The paper is organized as follows: Section 2 provides an overview of related work. In Section 3 we introduce extensibility in DGS. Section 4 introduces the full object model into the DGS programming language. The use of the map components in SLGeometry is demonstrated in Section 5. Section 6 presents the just in time (JIT) partial compilation of expressions in SLGeometry, whilst next section presents connecting Arduino-based microcomputers to SLGeometry. Section 7 concludes the paper.

2. RELATED WORK

Since the late 1960s some mathematics educators positively assesed the role of mathematical software tools, models and modeling in the teaching and learning of mathematics [10]. In that time the availability of computers has changed the nature of process of problem solving in mathematics. Also it started to have significant influence on changes in mathematics curriculum at all levels of education. Furthermore, many conferences held panels giving recommendations to explore and summarize current thinking about the role of the computer for the curricula in the four fields of science: mathematics, physics, statistics and chemistry [11].

The application of computers and different kinds of educational software is a topic of constant interest for scientists and different educational stakeholders who significantly influence creation and dissemination of school curricula, especially in science and mathematics. Many talks and papers are presented at conferences, providing overview of the current state of the art in technology enhanced learning, including both practice-oriented experiences and research-based evidence.

In early 1990s Blum and Niss [12] reviewed applied problem solving, modeling and applications in mathematics education, extended with use of computers, and its relations to other subjects. In contemporary mathematical modeling, the process of translating between the real world situations and mathematics in both directions is one of the essential topics in mathematics education [13].

Kauffmann [14] observes that using computer-aided design software in high school and university education contributes to better and faster comprehension of geometric problems than using traditional teaching methods. Drijvers et al. [15] emphasize that new demands occur in

educational systems in order to prepare students for future professions where technology offers enormous opportunities for teaching and learning. Exploiting these opportunities requires rethinking educational paradigms and strategies.

STEM (science, technology, engineering, and mathematics) education is also concentrated on developing tools and processes for teaching, which integrate concepts that are usually taught as separate subjects in different classes and emphasizes the application of knowledge to real-life situations, enabling teachers to teach mathematics and science much better and more effectively [16, 17]. Providing early exposure to STEM content can ensure that students will continue their interest in STEM subjects through middle and high school up to university level [18]. Nowdays seems that a new goal of education is to move from STEM to STE-A-M, i.e. to include arts, design and creation, as one of the successful models of coexistence between art and science education [19].

Sometimes the teachers are willing to learn new tools for teaching geometry, but they are not also ready to implement these tools in their everyday teaching practice in schools. This appears in particularly in developing countries, mostly in the rural areas where different necessary resources (including computers) for integration of technology in school education are very limited [20–23].

Similar problem appears in Serbia as well, where adequate teaching materials and applications in Serbian language that teachers can use and display to the students are lacking. Further, in last decade, teachers are obliged to have adequate skills and knowledge to use computers, and there are curricula instructions compatibility improvement with modern trends also. However, the optimistic fact is that the number of teachers who successfully implement information and communication technologies (ICT) in their classroom environment is steadily increasing [24–26].

The importance of use of technology in mathematical education and research is accepted predominantly at a rhetorical level. Since the computers are everywhere, school mathematics has not changed to reflect this situation, i.e. the school curriculum still needs to be innovated and improved. The use of well known DGS in faculty courses makes the students to play a much more active role than in the traditional learning [27]. Later they will probably continue to use such technological advancement in primary and secondary schools and attract and motivate pupils for different mathematical subjects.

According to Sarama and Clements geometry is the most relevant mathematical subject which lies at the heart of physics, chemistry, biology, geology and geography, art, civil engineering and architecture [28]. Using Logo environment they showed that computer manipulations can help students build on their physical experiences, tying them tightly to symbolic representations. Also computer manipulations can encourage students to make their knowledge explicit, which helps them build integrated-concrete knowledge [29].

Research and practice in education have to enhance each other through the development of a new set of tools for understanding and supporting powerful mathematics classroom instruction as well as instruction across a wide range of disciplines [30].

The use of mathematical software, starting with Computer Algebra Systems (CAS) on HP calculators, (some of first Hewlett-Packard scientific calculators), and continuing with, e.g. Mathematica and Maple, has made significant influence in teaching of mathematics. Some of researchers use CAS and DGS for the mathematical modeling of a real-life problems as a problem-solving activity that suits the purposes of mathematical learning. This contributes to the understanding of known mathematical concepts to the learning of new mathematical concepts and establishing interdisciplinary relationships [31, 32].

Along with the development of automatic theorem provers occurred a need for developing accompanying graphic interfaces which allow interactive manipulation by mouse dragging and manipulation of mathematical expressions in symbolic form. Commercial products such as Cabri [2, 33], Cinderella [3] and Geometer's Sketchpad [4, 34], cover high and middle school mathematics and physics education through algebra, analysis, geometry, trigonometry, mechanics and optics, as well as many other subjects. Books, tutorials and forums contain many teaching materials with presented and solved problems.

Recently, besides the well-known commercial DGS new free and open sources dynamic geometry systems, some of them having functionality of computer algebra systems, have also emerged. For example GeoGebra [1, 35] is a free DGS, created by Markus Hohenwarter in 2001, maintained by an international team of developers, which contribute to a more comprehensive set of features integrating geometry, algebra, calculus, statistics, graphing, spreadsheets and 3D augmented reality. A large self-sustained community of users helps students and teachers with numerous examples of teaching materials and offers help through online forums. It had become the leading provider of DGS worldwide as software for teaching and learning: available in many languages, organizing International GeoGebra institutes (IGI) which focus on training and maintain on-line support system for teachers, develop and share workshop resources and classroom materials, conduct and implement research projects in teaching and learning mathematics and other STEM subjects.

The aforementioned DGS use the functional approach for specifying dynamic drawings. Our extensible framework leverages the common Component Object Programming practices. After identifying the core structures, mechanisms and behaviors in a typical DGS, we devised the abstract model of an expression evaluator, and a set of code annotations (metadata) to provide concrete functionality. This OOP approach is very good accepted by students and pupils in our experiments [8, 9].

Also, we want to introduce development by parts in DGS by allowing component development, independent of the development of DGS itself. It is well known principle introduced with enhancement of software and its development in teams in different development cycles. The components are included in DGS in run-time and have the same treatment as first-class citizens in DGS. The natural extension of this approach is to provide visual components that, beside the program's aspects (properties and behavior), also have a graphical representation.

In the following sections we present these approaches.

3. EXTENSIBILITY IN DYNAMIC GEOMETRY SOFTWARE

SLGeometry framework for dynamic geometry we have been developing for several years consists of following components:

— Specification for Types, Functions and Visual elements,
— CAS Engine,
— Extensibility infrastructure,
— JIT compilation subsystem,
— Expression parser,
— Interactive components.

SLGeometry consists of a parser, an expression evaluator (Engine) and a graphical surface (GeoCanvas) (Fig. 1). A set of expressions, stored in named variables, which represent the elements of a dynamic drawing are maintained the Engine. GeoCanvas displays geometric shapes and UI controls, and responds to user interaction.
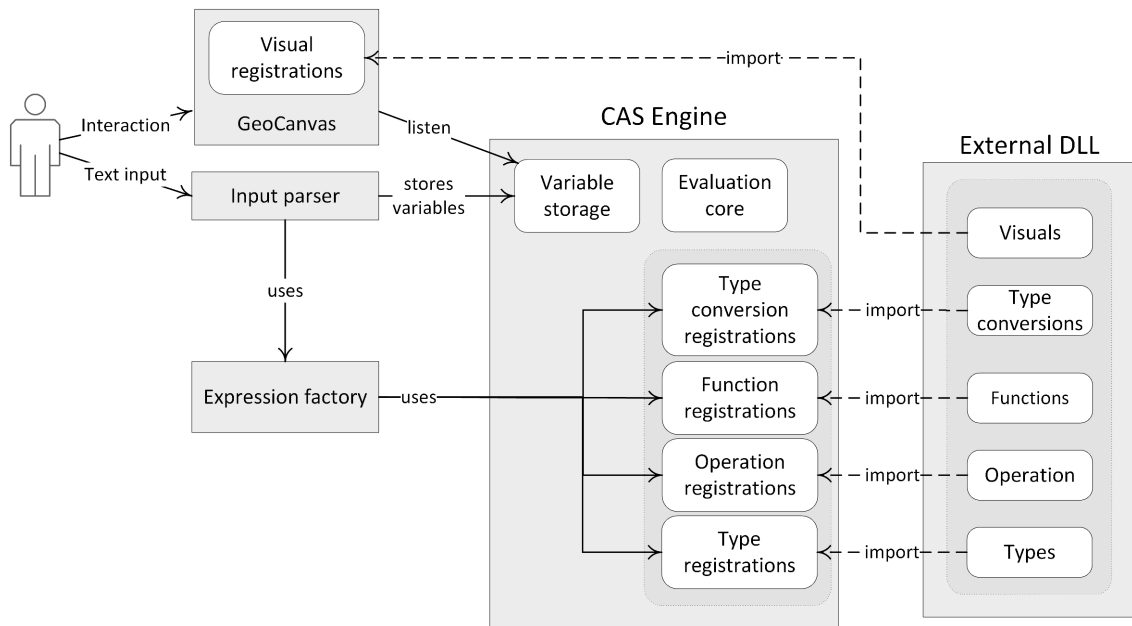
Figure 1. SLGeometry system architecture overview

New types, functions and visual objects can be imported from DLL files into the Engine at runtime. The DLL files can be implemented by independent developers in C# outside of the regular implementation cycle of SLGeometry, by inheriting from a set of standard C# classes and interfaces and following guidelines and recommendations.

Dynamic drawings are created by constructing expressions in a functional domain-specific language (abbreviated: FLG) and assigning them to variables in the Engine. In that regard, the set of variables can be considered equivalent to the drawing it represents. The set of types (T), type conversions (C), operations (O), functions (F) and visuals (V) in the FLG is denoted with $\tau = \{T, C, O, F, V\}$.

The Engine evaluates the expressions stored in variables. If the result of an evaluation corresponds to a visual type registered with the GeoCanvas, the visual object, is created and shown on the GeoCanvas. In this way, a one-to-one mapping is established between a variable and a visual object. Expressions can reference other variables and in this way, dependencies are established between variables. A change in one variable triggers recursive recalculation of all dependent variables. The user can manipulate one visual object, and all dependent variables — and their corresponding visual object — will be recalculated and their visuals updated on the GeoCanvas.

Usually for any software product is characteristic that it is getting mature during their use and new features are adding to it to increase quality and functionality. A DGS can also be extended by adding new features. For example, extension can be achieved by adding new data types (object types) and appropriate functions dealing with these new types, to its programming language.

As a particular example, a Triangle object type can be added in SLGeometry, which represents a triangle defined by three points. A corresponding Triangle function can also be added, which takes one of several sets of arguments and produces a Triangle object as a result. For example, the following function can be considered (Fig. 2):

Triangle (point1, point2, point3) defines a triangle via three corners,

Triangle (point1, side1, side2, side3) defines a triangle via three sides,

Triangle (point1, angle1, side1, side2) defines a triangle via two sides and the angle between them.
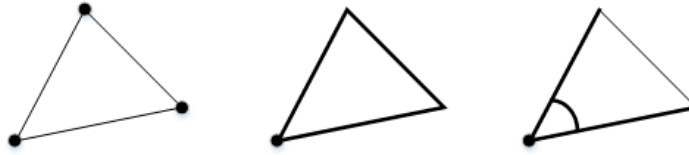

Figure 2. Three ways to define a triangle by the Triangle function

Software development is a complex process which includes teams of developers working semi-independently to implement different parts (components) of the final product. A software component should be a self-contained and self-described unit which is integrated into the main software either at the development stage, or at runtime. In the latter case the components are usually called plug-ins. We developed a metadata specification which is used to describe the plug-ins and enable seamless integration of new content in SLGeometry. Thus, new functions and data types are easily imported into SLGeometry by way of plug-ins. Our solution enables independent development of new functionalities outside of the main development cycle, and easy customization of SLGeometry for various subjects.

4. OBJECT DATA TYPES AND NOTATION

Geometric shapes are regarded by humans as objects with specific properties. We infer information about shapes by simply looking at them, or by performing some calculations. For example, the triangles from Fig. 2 have three corners each; regardless of the way they were constructed. The corners of the first triangle are specified in its definition, while the corners of the other two triangles are calculated from available data: lengths of sides and the angle between sides.

Existing DGS have designated functions for obtaining data about properties of geometric shapes. For example, in GeoGebra, one can use the Midpoint function (called "command" in GeoGebra) to obtain the midpoint of a segment, and of several other object types. The function-centric approach requires the user to apply the function on the object in order to extract the property value, for example, $Midpoint(Segment((1,1),(1,5)))$ or $Midpoint(s)$ if $s = Segment((1,1),(1,5))$.

This approach, although simple at first glance, has a serious disadvantage in a plugin-enabled DGS. When a new object type is imported into DGS, it should be accompanied by a set of functions related to property evaluation and extraction, which quickly leads to a cluttered function set, with many similarly named functions, which have a very specific and narrow application. Let us consider the Triangle object. It has dozens of properties, a subset of which is shown in Table 1. By the function-centric principle, each of these properties should be supported by a separate function.

The answer, to the issue of the function set overcrowding, is to introduce the full object model into the DGS programming language. In SLGeometry, geometric shapes are objects with their properties. Properties are accessed by using the dot notation, in the same manner as in

Table 1. A subset of properties of the Triangle object type

Property	Type
A, B, C	Point
Area, Perimeter	Number
SideA, SideB, SideC	Segment
AngleA, AngleB, AngleC	Angle
MedianA, MedianB, MedianC	Segment
Centroid, Orthocenter	Point
AltitudeA, AltitudeB, AltitudeC	Segment
Incircle, Circumcircle	Circle

object oriented languages, e.g. Java and C#. Thus, the example with the midpoint can be rewritten in a simpler and more natural way, as $Segment((1,1),(1,5)).Midpoint$ or $s.Midpoint$ if $s = Segment((1,1),(1,5))$.

Dot notation brings the "object-dot-property" naming scheme into DGS. Considering that properties of geometric shapes may be shapes themselves, this scheme becomes recursive. For example, the length of a side of a triangle is accessed as $Triangle(a,b,c).SideA.Length$ or $t.SideA.Length$ if $t = Triangle(a,b,c)$ and $a = (1,1), b = (1,5), c = (2,4)$.

As the properties of an object are kept within the object itself, it is possible to implement a graphical user interface which enables the user to discover the objects' properties interactively, e.g. by right-clicking or touching it. By the same recursion as above, the user can 'drill down' the object hierarchy and discover objects within objects.

Midpoint(\<Segment\>)

   Returns the midpoint of the segment.

   Example: Let `s = Segment((1, 1), (1, 5))` . `Midpoint(s)` yields *(1, 3)*.

Midpoint(\<Conic\>)

   Returns the center of the conic.

   Example: `Midpoin(x^2 + y^2 = 4)` yields *(0, 0)*.

Midpoint(\<Interval\>)

   Returns the midpoint of the interval (as number).

   Example: `Midpoint(2 < x < 4)` yields *3*.

Midpoint(\<Point\>, \<Point\>)

   Returns the midpoint of two points.
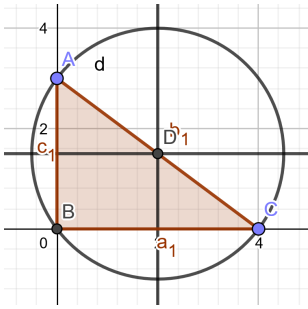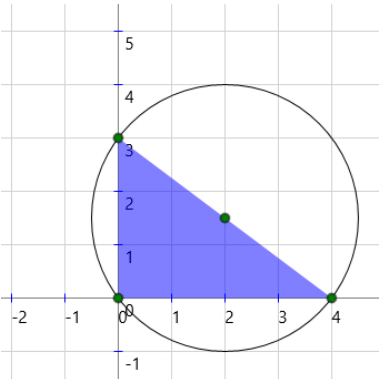
   Example: `Midpoint((1, 1), (5, 1))` yields *(3, 1)*.

Midpoint(\<Quadric\>)

   Returns the midpoint of the given quadric (e.g. sphere, cone, etc.)

   Example: `Midpoint(x^2 + y^2 + z^2 = 1)` yields *(0, 0, 0)*.

Figure 3. The Midpoint function in GeoGebra

Table 2. Circumcircle in GeoGebra and SLGeometry

GUI	Expression input	Dynamic drawing
GeoGebra	$A = (0,3)$ $B = (0,0)$ $C = (4,0)$ $t1 = Polygon(A,B,C)$ $a = Segment(B,C,t1)$ $b = Segment(C,A,t1)$ $c = Segment(A,B,t1)$ $f = PerpendicularBisector(a)$ $g = PerpendicularBisector(c)$ $D = Intersect(f,g)$ $d = Circle(D,C)$	
SLGeometry	$A = (0,3)$ $B = (0,0)$ $C = (4,0)$ $T = Triangle(A,B,C)$ $K = T.Circumcircle$	

Let us observe a triangle defined by 3 points, A, B and C and compare the way its circumcircle is drawn in GeoGebra and SLGeometry (Table 2). In GeoGebra, triangle is defined by way of the Polygon function, and each individual segment is automatically assigned a variable. A center of the circumcircle is defined as the intersection of the sides' bisectors, which are obtained with the PerpendicularBisector function. The intersection is calculated with the Intersect function. Finally, the Circle function is used to draw the circumcircle.

In contrast, the Triangle shape in SLGeometry already has the Circumcircle property. The triangle T is defined by 3 points A, B and C using the Triangle function. Now it suffices to address the T.Circumcircle property and the circle appears in the drawing. This is obviously an improvement, as Circumcircle needs not be implemented as a separate, stand-alone function in SLGeometry. It is implemented under the Triangle object. The resulting syntax in SLGeometry is thus simplified compared to the classical functional approach in other DGS.

5. INTERACTIVE COMPONENTS

DGS are designed to operate primarily with numbers, geometric objects and mathematical functions. Today, however, many teachers employ DGS to construct interactive examples for subjects other than mathematics, such as physics, chemistry, biology, geography, engineering and art. Creating dynamic drawings for such subjects can be a tedious undertaking. For example, an interactive clock with moving hands can consist of as many as 30 separate geometric objects. The many objects that make up the clock are difficult to distinguish from the other objects, drawing the pupils' attention away from the main focus of the material. Making a simple copy of the clock generates another 30 separate objects and adding to the clutter. Furthermore, it is difficult to reuse the clock in another drawing. The same limitation also appears when creating teaching materials for other subjects. For example, teachers often use DGS to show geographic maps, consisting of tens and hundreds of points.

Our proposed solution is to introduce interactive components in SLGeometry. Each component, no matter how complex, can be included into dynamic drawings as single object. Each component must be developed in C# by a developer. All data needed for drawing countries are added to Country function in advance.

For example, the geographic map component (Country function) displaying Italy is shown in Table 3. The component takes the country name as a parameter and generates a list of points comprising its outline.

Table 3. Italy shown by the Country component
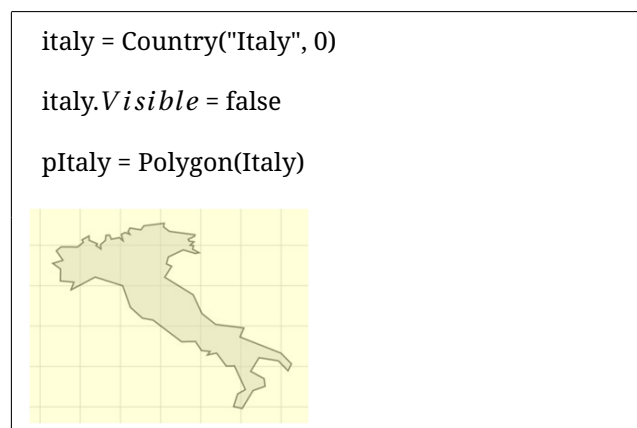
Components in SLGeometry are either functions or visual objects, which are added to the existing functions and visual objects. When considering the use of components, one should have in mind several important issues.

Without components

— One complex object is constructed from many geometric shapes,
— Many variables required to hold these shapes,
— No logical connection between the shapes,
— Construction can be completed in DGS alone,
— Duplicating the complex object requires duplication of all its constituent parts.

With components

— One complex object is represented by one component,

— One variable is required to hold the component,
— One-to-one correspondence between the component and its real life counterpart,
— Component is developed by an experienced developer, outside the DGS,
— Duplicating the complex object creates only one additional. variable

As one can see, the existence of components in DGSs have many positive aspects. However, there is also a drawback — each component must be developed in C# by a developer, outside of SLGeometry. This means that the majority of ordinary users do not possess the necessary skills for component development. On the other hand, given the proper instructions and adequate guidelines, a reasonably skilled developer could make a number of components and make them available for download, for example via a dedicated Web site or from within the DGS .

6. JUST IN TIME COMPILATION

From our teaching experience and our colleagues, we learned that the dynamic drawings, created by teachers and students, are becoming more complex. This phenomenon stems from DGS being applied to an ever growing range of teaching subjects, such as geography, physics, electrical engineering, architecture and arts. Compared to geometric drawings, the drawings for other subjects contain a substantially greater number of expressions and corresponding visual components and real-time recalculations of dynamic drawings is becoming CPU intensive. Therefore it is logical to try to speed up calculation of expressions in DGS.

We are currently working in order to solve these problems. Our main intention is to work on just in time (JIT) partial compilation of expressions in SLGeometry. We have developed the necessary infrastructure, which enables compilation to be implemented on any individual function. The main idea behind partial compilation is to use the metadata and dynamic typing to identify parts of expressions trees which are compilable, and to transform those parts to compiled code. Authors of different functions can decide whether they want to support compilation, by implementing a Boolean flag. Therefore it is possible that some parts of expression trees are not compilable . Other factors can also affect compilability. Therefore an expression tree may be only partially compilable (Fig. 4).
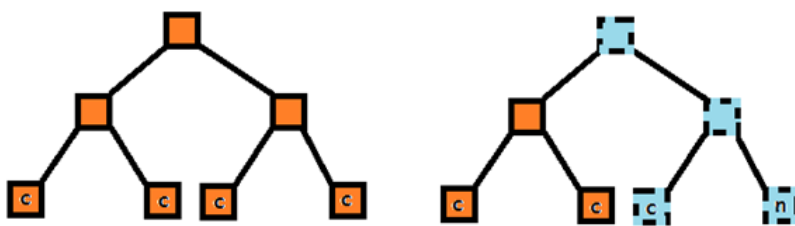

Figure 4. A completely compilable tree vs. a partially compilable tree

There are several possible compilation approaches that have been considered — MSIL code, Expression trees and delegates. It is up to the authors of functions to decide which approach they want to use, based on possible performance gains and the overall difficulty of implementation. In our preliminary testing, we have found that delegates may provide a good balance in that regard. Each function passes the "initialization" phase during which it can decide, based on the types of supplied arguments, which delegate to use for evaluation, assuming that the types of the

arguments will not change between evaluations. This way, under repeated evaluation, the function can omit parameter type checking and branching, and proceed to evaluate the optimized delegate, thus saving a considerable amount of time.

We have performed several tests on compiled code and found out that significant reduction in calculation time is possible.

7. ARDUINO INTERFACE

We are considering a possibility for connecting Arduino to SLGeometry. Arduino [36] is a family of microcontroller-based development boards and accompanying open source software which are affordable, simple to program and easy to connect to a wide variety of electronic components and peripherals such as sensors, radios, motors etc. Although intended for education and hobby projects, Arduino is also used in commercial products. Arduino software consists of a base platform with integrated development environment and an open-access library which contains hardware drivers and software components. Arduino is programmed in C++.

An Arduino board has several input and output pins. Some pins are digital, while other are analog. Digital pins only support logical values of true and false, represented by discrete voltage levels of 5V and 0V. Analog pins can measure voltages in the continuous range from 0V to 5V. Peripherals communicate with Arduino via interconnected pins, which can be read from and written to in a C++ program.

There exist many educational examples, from digital clocks and thermometers to robots and Internet of Things (IoT) applications. Currently we are working on an object model of the hardware which would represent Arduino as a visual component in SLGeometry, with inputs and outputs that can be bound to arbitrary mathematical expressions. Our aim of connecting Arduino and SLGeometry is twofold:

1. to obtain real-world data from Arduino and have it processed in real-time in SLGeometry, and
2. to enable SLGeometry to control various devices on the Arduino.

The virtual Arduino component for SLGeometry will contain an array of input and output pins, each mapped to a corresponding pin on Arduino. Changes on input pins will propagate from Arduino hardware to SLGeometry, while changes in the virtual Arduino component will propagate from SLGeometry to the hardware. Additional electronic components, connected to Arduino, will also be mapped to corresponding sets of pins in the software component. A two-way, real-time transfer of pin states will be established between the hardware and the software, thus enabling the software to act upon multiple inputs generated from the hardware, and send its outputs in the opposite directions.

A simple example demonstrates the benefits of this approach in education. Let us consider a monitored capacitor charging circuit (Fig. 5). As the capacitor charges, the voltage across its terminals rises, quickly in the beginning and more slowly as it approaches full capacity. One method to determine the stopping criterion for charging is to observe the slope of the voltage curve. Charging stops when the slope becomes almost horizontal, i.e. falls below a certain threshold.

To determine the slope of the experimental curve, one can employ numerical differentiation. On the other hand, school pupils may use geometric methods: for example, construct a line through the last two measured points and compare its slope to the threshold. In this example, measured data is obtained from the Arduino, and automatically transferred to the virtual Arduino component in SLGeometry. From there, a simple expression extracts the last two points
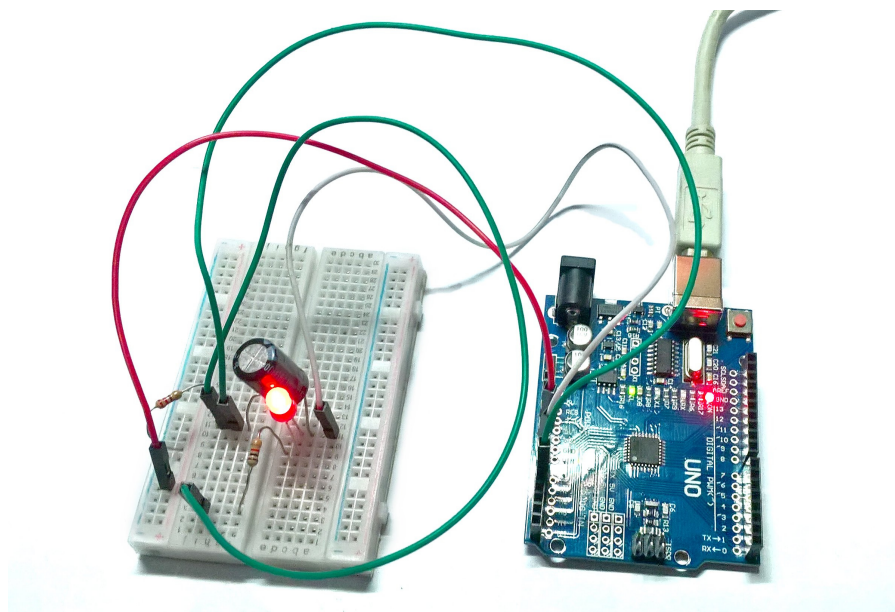
Figure 5. Capacitor charging experiment with Arduino

and constructs a straight line through them. A Line object is created as a result, which has the Slope property. Thus, an interactive example is created, which does not require any advanced knowledge of numerical mathematics or geometry, yet illustrates the solution in an obvious way.

We hope that results of such our efforts will find its positive place and be motivational for pupils and teachers in primary and secondary education, especially applied to subjects such as physics and electronics.

8. CONCLUSIONS

As teachers and student become more proficient in the use of DGS, they are trying to apply them in various subjects, not just mathematics and geometry. Although their interactive nature makes them attractive to use in teaching and teaching materials creation, some drawbacks stem from their essentially geometric nature. We have tried to develop new features and have them built into SLGeometry in order to test new approaches to using DGS in teaching.

Extensibility is a key feature of any DGS that aims to cover the ever growing spectrum of applications in education. Our solution to extensibility provides the users with new functionality, suitable for their needs, and developers with a simple framework and guidelines for development of new data types and functions. Full object model, named properties and dot notation are a commonplace in modern programming languages. Bringing those features into DGS helps reduce the clutter in the function pool, which occurs as a consequence of constantly increasing number of functions and shapes that modern DGS support. It also helps bring the objects and their properties together, and also enables interactive discovery of properties of geometric shapes. Just in time compilation and Arduino interface are two features that are currently under development. Our aim is to explore the ways to make a DGS more CPU efficient. Also, we are looking for the best way to collect, process and explore real-time data from microcontrollers in a DGS.

“”

References

1. GeoGebra. [Online], Available: https://www.geogebra.org/
2. Cabri. [Online], Available: http://cabri.com/en/
3. The Interactive Geometry Software Cinderella. [Online], Available: https://cinderella.de/tikiindex.php
4. The Geometer's Sketchpad. [Online], Available: http://www.keycurriculum.com/sketchpad.1.html
5. Wolfram Mathematica [Online], Available: https://www.wolfram.com/mathematica/
6. D. Radaković and D. Herceg, "Towards a Completely Extensible Dynamic Geometry Software with Metadata," *Computer Languages, Systems & Structures*, vol. 52, pp. 1–20, 2018; doi: 10.1016/j.cl.2017.11.001
7. Đ. Herceg and D. Radaković, "The Extensibility of an Interpreted Language Using Plugin Libraries," In *AIP Proc. Numerical Analysis and Applied Mathematics ICNAAM 2011*, vol. 1389, pp. 837–840, 2011; doi: 10.1063/1.3636863
8. Đ. Herceg and D. Radaković, "A Platform for Development of Mathematical games on Silverlight," *Acta Didactica Na pocensia*, vol. 6, no. 1, pp. 77–90, 2013. [Online], Available: https://files.eric.ed.gov/fulltext/EJ1053670.pdf
9. Đ. Herceg, V. Herceg-Mandić, and D. Radaković, "The Teaching of Geography Using Dynamic Geometry Software," In *Local Proceedings of the Fifth Balkan Conference in Informatics, BCI 2012,* Novi Sad, pp. 11–15, 2012.
10. M. Niss, "Models and Modelling in Mathematics Education," *EMS Newsletter December 2012*, pp. 49–52, 2012. [Online], Available: http://www.euro-math-soc.eu/ems_education/Solid_Findings_Modelling.pdf
11. M. M. Underkoffler, "Computer assisted instruction in college general education mathematics," *PhD thesis*, Digital Repository @ Iowa State University, 1969; doi: 10.31274/rtd-180813-1201
12. W. Blum and M. Niss, "Applied mathematical problem solving, modeling, applications, and links to other subjects: state, trends and issues in mathematics instruction," *Educ. Stud. Math.*, Kluwer Academic Publishers, Dordrecht 22, pp. 37–68, 1991.
13. W. Blum and R. Borromeo-Ferri, "Mathematical Modelling: Can It Be Taught And Learnt?" *J. Math. Model. Appl.*, vol. 1, no. 1, pp. 45–58, 2009.
14. H. Kaufmann and D. Schmalstieg, "Mathematics and geometry education with collaborative augmented reality," *ACM SIGGRAPH 2002 Conference Abstracts and Applications*, San Antonio, Texas: ACM, pp. 37–41, 2002; doi: 10.1145/1242073.1242086
15. P. Drijvers, L. Ball, B. Barzel, M. K. Heid, Y. Cao, and M. Maschietto, "Uses of Technology in Lower Secondary Mathematics Education," *Uses of Technology in Lower Secondary Mathematics Education*, Springer International Publishing, 2016; doi: 10.1007/978-3-319-33666-4
16. H. Burkhardt, *Towards Research-based Education*, Shell Centre for Mathematical Education Publications Ltd., pp. 1–25, 2018. [Online], Available: https://www.mathshell.com/papers/pdf/hb_2018_research_based_education.pdf
17. H. B. Gonzalez and J. J. Kuenzi, *Science, Technology, Engineering, and Mathematics (STEM) Education: A Primer*, 2012. [Online], Available: https://fas.org/sgp/crs/misc/R42642.pdf
18. N. DeJarnette, "America's children: providing early exposure to stem (science, technology, engineering and math) Initiatives," *Educ.*, vol. 133, no. 1, pp. 77–84, 2012.
19. Z. Lavicza, K. Fenyvesi, D. Lieban, H. Park, M. Hohenwarter, J. Mantecon, and T. Prodromou, "Mathematics Learning Through Arts, Technology and Robotics: Multi-and Transdiscpilinary Steam Approaches," In *8th ICMI-East Asia Regional Conference on Mathematics Education*, Taipei, Taiwan, 2018, pp. 110–122.
20. B. R. Mainali and M. B. Key, "Using dynamic geometry software GeoGebra in developing countries: A case study of impressions of mathematics teachers in Nepal," *Int. J. Math. Teaching Learn.*, pp. 1–16, 2012. [Online], Available: http://www.cimt.org.uk/journal/mainali.pdf
21. K. K. Bhagat and C. Chang, "Incorporating GeoGebra into Geometry Learning-A lesson from India," *Eurasia J. Math., Sci. Technol. Educ.*, vol. 11, no. 1, pp. 77–86, 2015; doi: 10.12973/eurasia.2015.1307a
22. M. Khalil, R. A. Farooq, E. Cakiroglu, U. Khalil, and D. M. Khan, "Mathematical Achievement in Analytic Geometry of Grade-12 Students through GeoGebra Activities," *J. Eurasia Math. Sci. Technol. Educ.*, vol. 14, no. 4, 1453–63, 2018; doi: 10.29333/ejmste/83681
23. O. B. Han, N. D. B. Abd Halim, R. S. B. Shariffuddin, and Z. B. Abdullah, "Computer Based Courseware in Learning Mathematics: Potentials and Constrains," *Procedia Soc. and Behav. Sci.*, vol. 103, pp. 238–244, 2013; doi: 10.1016/j.sbspro.2013.10.331

24. J. Jezdimirović, "Computer Based Support for Mathematics Education in Serbia," *Int. J. Tech. Inclusive Educ. (IJTIE)*, vol. 3, no. 1, pp. 277–285, 2014; doi: 10.20533/ijtie.2047.0533.2014.0036
25. E. Ljajko and V. Ibro, "The Development of Development of ideas in a GeoGebra — aided mathematics instruction," *Mevlana Int. J. Educ. (MIJE)*, vol. 3, no. 3, pp. 1–7, 2013; doi: 10.13054/mije.si.2013.01
26. Lj. Diković, "Applications GeoGebra into teaching some topics of mathematics at the college level," *Comp. Sci. Inf. Sys.*, vol. 6, no. 2, pp. 191–203, 2009; doi: 10.2298/csis0902191D
27. H. Burkhardt, "Curriculum Design and Systemic Change," *Math. curriculum sch. educ.*, pp. 13–34, 2013; doi: 10.1007/978-94-007-7560-2_2
28. J. Sarama and D. H. Clements, *Studies in Mathematical Thinking and Learning Series*, Taylor and Francis, 2009. [Online], Available: https://books.google.rs/books?id=Z5KOAgAAQBAJ
29. J. Sarama and D. H. Clements, ""Concrete" Computer Manipulatives in Mathematics Education," *Child Development Perspectives* vol. 3, no. 3, pp. 145–150, 2012; doi: 10.1111/j.1750-8606.2009.00095.x
30. A. H. Schoenfeld, "What Makes for Powerful Classrooms, and How Can We Support Teachers in Creating Them? A Story of Research and Practice, Productively Intertwined," *Educ. Researcher*, vol. 43, no. 8, pp. 404–412, 2014; doi: 10.3102/0013189X14554450
31. R. M. Zbiek and A. Conner, "Beyond Motivation: Exploring Mathematical Modeling as A Context for Deepening Students' Understandings of Curricular Mathematics," *Educ. Stud. Math.*, vol. 63, no. 1, pp. 89–112, 2006; doi: 10.1007/s10649-005-9002-4
32. M. Aktümen, T. Horzum, and T. Ceylan, "Modeling and Visualization Process of the Curve of Pen Point by GeoGebra," *European Journal of Contemporary Educatio*, vol. 4, no. 2, pp. 88–99, 2013; doi: 10.13187/ejced.2013.4.88
33. C. Laborde and B. Capponi, "Cabri-géomètre constituant d'un milieu pour l'apprentissage de la notion de Figure géométrique. Recherches en Didactique des Mathématiques," *Recherches en didactique des mathématiques*, vol. 14, no. 1.2, pp. 165–210, 1994.
34. N. Jackiw and W. Finzer, *The Geometer's Sketchpad: Programming by Geometry*, A. Cypher et al, Eds., MIT Press, Cambridge, MA, 1993, pp. 293–308.
35. GeoGebra Materials. [Online], Available: https://www.geogebra.org/materials/
36. Arduino. [Online], Available: https://www.arduino.cc/

Возможные усовершенствования современного программного обеспечения динамической геометрии

Херцег Д.[1], профессор, herceg@dmi.uns.ac.rs
Радакович Д.[1], магистр, ассистент, davorkar@dmi.uns.ac.rs
Иванович М.[1], профессор, mira@dmi.uns.ac.rs
Херцег Д.[2], доцент, vuletic@uns.ac.rs

[1]Университет Нови-Сад, Факультет естественных наук и математики, Нови-Сад, Сербия
[2]Университет Нови-Сад, Факультет технических наук, Нови-Сад, Сербия

Аннотация

Современное образование начинает вытеснять традиционное (уроки от учителя к ученику) высокотехнологичным обучением с использованием различных обра-

зовательных инструментов и подбором материалов, которые являются эффективными, действенными и привлекательными для учащихся. Программное обеспечение динамической геометрии (DGS) сегодня широко используется в преподавании и изучении математики. Такое образовательное программное обеспечение может развиваться несколькими способами, либо надстраивая новые функции, либо добавляя новые функции на поверхности, либо развивая механизм оценки в его ядре. Реализация DGS должна быть простой и модульной.

Чтобы добиться развития ядра DGS мы разработали среду программирования для программного обеспечения динамической геометрии SLGeometry с обобщенным функциональным языком и соответствующим механизмом оценки выражений. Механизм действует как фреймворк, в который встроена конкретная семантика в виде кода, аннотированного метаданными. Этот фреймворк преобразует обычный вычислитель дерева выражений в объектно-ориентированный. В то время как другие DG основаны на чисто функциональных оценщиках выражений, наше решение обладает преимуществами более общего, поддерживаемого, понятного, простого в реализации и обеспечивающего естественный способ задания свойств объекта в пользовательском интерфейсе, минимизируя типизацию и синтаксические ошибки. Модульный подход позволяет самостоятельно разрабатывать предметно-ориентированные компоненты, которые легко добавляются в механизм оценки в виде плагинов. Объектно-ориентированный характер фреймворка позволяет разрабатывать автономные единицы, такие как объекты и визуальные элементы, которые инкапсулируют предметную семантику и представляют ее пользователю в виде виртуальных заполнителей для реальных объектов и понятий.

В этой статье мы представляем несколько возможных улучшений программного обеспечения динамической геометрии, в первую очередь на платформе, которую мы внедрили. Кроме того, мы обсуждаем преимущества этих функций и их влияние на пользователей/студентов. Подход тестируется на SLGeometry — нашей платформе DGS, разработанной в C# на платформе .NET Framework.

Ключевые слова: *программное обеспечение для динамической геометрии, обучение, разработка компонентов.*