International Journal of Pure and Applied Mathematics
Volume 85 No. 4 2013, 781-811
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
doi: http://dx.doi.org/10.12732/ijpam.v85i4.14

ijpam.eu

COMPUTER GRAPHICS AND GEOMETRIC MODELLING – A HYBRID APPROACH

Alexander Penev

Faculty of Mathematics and Informatics
Plovdiv University "Paisii Hilendarski"
236, Bulgaria Blvd., Plovdiv 4003, BULGARIA

Abstract: The present paper describes an approach to geometric modelling, in which the representation of solids is hybrid and consists of an easily expandable collection of representations. We also consider an experimental prototype of a framework for the development of systems for geometric modelling based on open hybrid representation schemes. We present a method for creating software-hardware (hybrid) systems based on this approach.

AMS Subject Classification: 68U05, 97R60
Key Words: geometric modeling, solid modeling, hybrid representation, education

1. Introduction

The need for automation of the design work led to the emergence and development of a research area that aims to provide the necessary knowledge on hardware and mathematical provision in system analysis and engineering methodology for the specification, design, implementation, deployment and use of computer systems for design purposes. This led to the development of the so-called CAD [1] systems (and later to CAM [1], [6] and CAE [4] systems). The problems in the development were mainly due to the lack of complete information models of three-dimensional objects. Research investigations of these problems form and develop the scientific field geometric modelling – a central part of computer graphics [5].

Received: March 21, 2013

1.1. Computer Graphics and Geometric Modelling

One of the main tasks of Computer Graphics (CG) is the construction of models of scenes from the physical world and their visualization as images. This task is fundamental in the so-called geometric modelling. Geometric modelling includes theories, methods and systems aimed at creating complete information representations of three-dimensional real objects, which allow as to automatically calculate any well-defined geometric property of the objects they describe [5], [16].

1.2. Problems and Goals

In order to work with the models in the computer systems, they should be stored in the computer memory, which allows processing, converting, and ultimately displaying them. One problem is the choice of the representation scheme. There are many known and well studied in theory and practice representation schemes (B-Rep, CSG, F-Rep, etc.), as well as various visualization algorithms providing different speed and quality of their results.

The different purposes lead to the choice of different representation schemes (representations of solids) because each of them has its advantages and disadvantages – for example, one scheme may be faster when implementing some algorithms and slower in others. Experience shows that the advantages of different representation schemes are likely to complement each other in the so-called hybrid representation schemes. Today, almost all modern systems for graphic/geometric modelling use hybrid representation schemes, but in them the set of selected schemes is fixed, they are homogenized "by hand" and their processing algorithms are created in accordance with the specific combination of schemes, and usually one of them dominates the others.

The main objective/goal of this work is to explore the possibility and to propose a concept for building an open hybrid system for geometric modelling based on an open hybrid (non-homogenized) representation scheme, and to develop a prototype framework for creating such a system. Of course, such a framework should be open regarding the used representation schemes, because this would lead to greater applicability, flexibility and compatibility with the widespread schemes already in use, as well as to an easy adaptation and possible expansion by new representation schemes. Other areas of openness and hybridness besides the representations are: programming languages, operating systems, computer systems, software-hardware implementation of the processing algorithms or the use of modern graphics hardware, etc.

In order to achieve the main goal the following tasks were planned: Creating a concept for an open hybrid system for geometric modelling based on an open hybrid representation scheme; Selecting a framework architecture to support the development of such systems; Designing and implementing a prototype of such a framework (called OpenF) and applying it to practically create an example of an open hybrid system for geometric modelling based on an open hybrid (non-homogenized) representation scheme. This example will constitute a constructive proof of the goal in view.

1.3. Related Works

For the first time the concept of a hybrid representation scheme is found in [16]. Almost all modern systems use two, three or more homogenized representation schemes. They are hardly a subject to change and expansion with other representation schemes.

There are hundreds of systems based on the hybrid approach. We will examine some of them based on the following criteria: to maintain and work with more than one representation, to be modern representatives of widely-used (and different) sub-classes of geometric modelling systems or to realize important algorithms, to have mechanisms to expand the capabilities of the system, to use some form of specialized graphics hardware, etc.

One example of a similar existing system is HyperFun [3], but it has different goals, architecture, and is, to a great extent, (only) F-Rep centric. HyperFun has a scene description language (SDL), a modelling framework and visualization algorithms. It relies heavily on the relatively easy homogenization of some representation schemes and their inclusion as elements of the F-Rep.

Another system is POV-Ray [15] – a Ray tracer with its own scene description language (SDL). In this system different types of representation schemes can be used relatively freely. A set of primitives is used that is converted to the internal representation, which uses mathematical definitions of objects (spheres, planes, cylinders, etc.). The main emphasis is on the visualization of the scenes described by the algorithm Ray tracing.

Autodesk Maya [13] – a widely used commercial system for geometric modelling. It supports parametrized B-Rep/CSG models (through a constructive tree and a dependency graph) with possibilities to use other representations (hybrid, homogenized). Autodesk Maya has a powerful mechanism for extending the functionality. It visualizes through polygonal meshes and Ray tracing. It makes maximum use of modern graphics hardware.

N	Application/ Framework	Representation schemes								Hybridness	
		BRep	FRep	CSG	Voxel	Param	Sweep	Part	Other	Reps	Hardware
1	HyperFun***	−	+	+	+	∼	+	−	∼	+	∼
2	POV-Ray***	+	+	+	−	+	+	∼	+	+	−
3	Autodesk Maya**	+	−	+	−	+	+	+	∼	+	+
4	Blender**	+	−	∼	−	−	+	+	+	+	+
5	OpenSceneGraph*	+	−	+	∼	−	−	+	∼	+	∼
6	OpenF*	+	+	#	+	#	#	#	#	open	open

Table 1: Comparison of the systems in terms of the supported representations and hybridity

Blender [2] – a widely known open source noncommercial system for geometric modelling. It supports B-Rep/CSG models with options for the use of other representations (hybrid, homogenized). Blender has a powerful mechanism for extending functionality. It visualizes by means of polygonal meshes and Ray tracing. It uses modern graphics hardware.

OpenSceneGraph [22] – a framework for building systems based on a scene graph. It mainly supports B-Rep. It has a mechanism for extending functionality, and a mechanism for adding new types of nodes (NodeKits) in the constructive tree of the scene (using homogenization). OpenSceneGraph uses the modern graphics hardware (mainly OpenGL) for visualization.

OpenF – a prototype of an experimental framework for creating open hybrid systems for geometric modelling based on open hybrid representation schemes. It is one of the main tools for achieving the goals of this work (see section 3).

Analysis and comparison of the capabilities of the systems examined (compared with those of OpenF) are given in Table 1. Here we use the following symbols: "*" stands for the systems that are frameworks, "**" is used for applications, "***" denotes the systems having SDL and implementing mainly visualization of objects described with SDL, "+" / "-" means that the corresponding system supports/does not support the given representation, "˜" stands for partial support, "#" indicates that support can be readily added, but is not implemented in the prototype.

From the analysis presented in Table 1 we can conclude that hybridity in modern systems for geometric modelling is not a novelty, but most of them are based on the B-Rep (and CSG), the other representations being complementary. In the cases when more of the known representation schemes are used, usually the main goal is just to visualize (HyperFun, POV-Ray). Typically, having mechanisms for extending the systems (plugins, etc.) for seamless support of more representation schemes, the approach of homogenization of the new schemes in some basic ones (e.g. in B-Rep or F-Rep) is applied. It is still not common to use the specialized graphical hardware in a hybrid way, i.e. one rarely sees smooth interchangeable use of graphics and other hardware.

2. Geometric Modelling

One of the most frequently used representation schemes in geometric modelling is Boundary Representation (B-Rep). For this scheme a lot of algorithms are developed that mainly use its special features and advantages. There are many programme systems and libraries that are based on this method for describing geometric information (CAD/CAM, research systems, games, etc.). Hardware was developed in order to speed up visualization.

Research on a new representation scheme, called Function Representation (F-Rep) [14], started to develop intensively during the last 15 years. It is based on the description of the objects included in the scene with the help of explicit functions, *R*-functions, etc. Theoretically it is not a new idea and could be found in V. Rvachev's work in the 1960s [18], [19]. This scheme allows easy integration with the elements of other well-known representation schemes such as CSG, B-Rep, Sweeping, etc. The most important reason for limiting F-Rep's usage until now is the necessity of performing a large number of calculations that are required to obtain accurate algorithm results. An example of this are the visualization algorithms. Nevertheless, the advantages of the scheme outnumber its disadvantages, and with the fast growth in computing power the last mentioned drawback becomes less important.

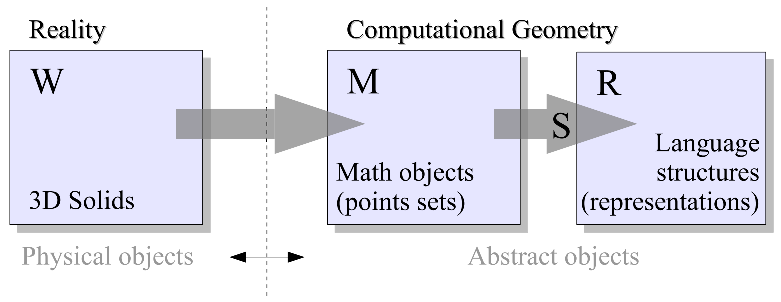

Figure 1: Two-stage approach to geometric modelling

Working with real bodies (three-dimensional solids) in geometric modelling is usually performed in a two-stage process (see Figure 1). First, the bodies are modelled in a mathematical space by mathematical objects (sets of points). Second, these mathematical objects are described by the information models (language structures or representations). The first stage is called *a modelling method* and the second stage is called *a representation method* or *a representation scheme* [16], [5].

2.1. Geometric Information

From the viewpoint of geometric modelling we are interested only in the general properties of the set of (three-dimensional) solids. This set is called geometric information, i.e. this term refers to a representation scheme used in geometric modelling [5].

Definition 1. *Geometric information* is called a set of attribute classes, which is perceived by one as a (rigid) solid. We denote it by the ordered triplet: $G = (\{s\}, \{m\}, \{p\})$, where $\{s\}$ is the set of spatial forms, $\{m\}$ is the set of metric characteristics, and $\{p\}$ is the set of parameters giving the location and orientation.

Geometric information is formed from the sets of attribute classes $\{s\}$, $\{m\}$ and $\{p\}$. They can describe any (rigid) solid. But what exactly will be included in these sets and the relationships among them depend on the particular representation scheme. To use the appropriate attributes, geometric information must be somehow modelled in the computer system, and the ways to do that are the so-called representations, i.e. one must have an information model.

Each geometric information G induces in the mathematical space (for example in E^3) an abstract object (set of points) or a family of abstract objects defined by the numerical characteristics of $\{m\}$ and $\{p\}$, as well as the form $\{s\}$. Obviously, two different pieces of geometric information G_1 and G_2 (or as we will understand from now on – their concrete representations) may describe the same sets of points. Hence the problem of comparing G_1 and G_2 arises. Two of the main tasks of geometric modelling are finding and generating equivalent and identical pieces of geometric information. [5]

For computer graphics and its applications the concept of geometric information is essential. It is the geometric information that is stored as data in the computer by using representation schemes (one or more). Since computers process data, the main subject to processing are the concrete representations of any specific geometric information. There does not exist a universal representation and this is due to the various modelling purposes. The characteristics to be described (i.e. the ones that are important to achieve the objectives), the way they are processed, the expected results after processing and many others are influenced by the choice of a representation. Some of the conditions that frequently contradict each other are: performance of the algorithms (visualization, etc.); memory size; flexibility and ease of editing scenes described by the appropriate representation, etc.

2.2. Representation Schemes. Representations

Consider the second stage of the two-stage process in geometric modelling in Figure 1.

Definition 2. *A representation* of an abstract object is called the syntactically correct symbolic structure, built by the characters of some alphabet according to some rules.

This looks like a language generated by some grammar, and representations are strings. But the representations are not limited to strings [16].

The set of syntactically correct representations is called a representation space, and it is indicated by R on the scheme.

A question arises about the meaning of the representations, i.e. their semantics. In geometric modelling, this means whether a real three-dimensional object (solid) corresponds to a particular representation. If we consider the sets in the mathematical space, which correspond to the three-dimensional solids (e.g. the R-sets), then the issue of semantics can be resolved by introducing rules that link any R-set with a corresponding representation in the space of representations R. This collection of rules is called *a representation method* or *a representation scheme.*

Definition 3. A *representation scheme* is called every mapping $S : M \to R$, where M is a mathematical space, and R is the space of representations.

The domain of S will be denoted by $D \subseteq M$, and the range of values by $V \subseteq R$. Obviously there exist many representation schemes with different properties. In practice, different representation schemes are used for different purposes. Some of the most common features of the representation schemes are: power, reality of the representations; unambiguity; conciseness; uniqueness; ease of creation; effectiveness of the applications, and others.

The following important question arises in case we have more than one representation scheme. Can we convert (and how) the representations from one of them into representations from the other, and vice versa? Under what conditions?

2.3. Boundary Representation Schemes

One of the most common representation schemes in geometric modelling and computer graphics is the boundary representation scheme (B-Rep). It is simple and clear, and the main algorithms for creating and visualizing are well known. Another major advantage of B-Rep is the existence of specialized graphics hardware (graphics accelerators) that implements a quick preview of the B-Rep based models (scenes).

2.3.1. Basic Concepts of B-Rep

The main idea of B-Rep [5], [16], [7] is to describe only the boundary of the solids as this is usually sufficient to perform the visualization. The boundary of the solid is represented as a finite number of bounded subsets called faces, patches or shells. An analogous representation is valid for the faces, and so on. Usually, the faces are triangles (rectangles, polygons or other NURBS surfaces), which are defined by their sides (edges), which in turn are defined by their end points (vertices) plus combinatorial structures describing the affiliation of the tops of the edges and the edges of the faces.

2.3.2. Visualization

B-Rep allows for a quick (and of relatively good quality/realistic) visualization of the two types of algorithms – of the object space and the observer. The algorithm Z-buffer [7] is often used for fast visualization of the scene, and when it is applied the invisible parts of the solids are removed. The existence of a hardware implementation of this algorithm in most of the modern computer systems (in the graphics accelerators) makes it a very good candidate not only for a quick visualization but also for one of good quality.

For more realistic (and usually slower) visualization the algorithm Ray tracing [7], [5] is usually applied, as in order to implement the algorithm it is sufficient to create an additional algorithm for finding the intersection of a ray with the solids, i.e. their faces. The large number of faces that describe the solids is problematic for the performance of this algorithm, so methods (acceleration structures) for quick elimination of most faces must be used: bounding spheres, bounding boxes (AABB), kD-trees, BHV-trees and more.

2.4. Functional Representation Schemes

Functional representation schemes (F-Rep) are schemes that are used for describing geometric objects (solids). The F-Rep [14] represents a geometric object by a real continuous function f defined in an Euclidean space.

2.4.1. Basic Concepts of F-Rep

A real continuous function f, which describes a solid, is defined by:

$$f : E^n \to R.$$

This function may define geometric information $G = (\{f\}, \emptyset, \emptyset)$, and it induces a point set $S_G = \{X \in E^n | f(X) \geq 0\}$.

Another way to interpret the value of the function f is as the signed distance from a point X to the surface of the solid S_G. These functions are called signed distance functions. Modelling by using these functions is more restrictive, but it also has some advantages (mainly for developing fast visualization algorithms).

The geometric operations in F-Rep are defined analytically. For example, the set-theoretic operations are implemented by using the so-called R-functions [18], [19], [21], (see (1)). Other known operations are [14]: blending, offsetting, Cartesian product, metamorphosis, bijective and linear mapping, projection, etc.

A fundamental advantage of F-Rep [11] is its openness (extensibility) in view of the possibility for adding new primitives, operations, and relations. Among its big advantages is also the easy implementation of nonlinear transformations and other complicated operations. For example, the operation metamorphosis (which means morphing one solid into another one) is hard to be implemented in B-Rep. However, in F-Rep it has a simple solution $f(t) \equiv t \cdot f_1 + (1-t) \cdot f_2$, $t \in [0, 1]$, where f_1 and f_2 are functional representations of two solids, t is a parameter determining the intermediate state of the resulting solid (the morphing phase). The use of the R-functions makes F-Rep more powerful. R-functions [18], [19] are real functions of real variables which inherit some properties of the logical functions (binary or ternary logic). For example, the conjunction $X \wedge Y$ is called the logical friend of the R-function

$$f_1 \wedge_a f_2 \equiv \frac{1}{1+a} \cdot \left(f_1 + f_2 - \sqrt{f_1^2 + f_2^2 - 2a \cdot f_1 \cdot f_2} \right). \qquad (1)$$

If $a = 1$, from (1) we find $f_1 \wedge_1 f_2 \equiv \min(f_1, f_2)$, and if $a = 0$, we have $f_1 \wedge_0 f_2 \equiv f_1 + f_2 - \sqrt{f_1^2 + f_2^2}$. Analogously, similar functions exist for other set-theoretical operations.

The contribution of R-functions to computer graphics, and F-Rep in particular, is the possibility for composing practical arbitrary solids (functions) based on other simpler and already constructed functions or primitives as spheres, cylinders, cones, etc. In general, F-Rep provides an easy opportunity to incorporate elements from other representations (not only ones from CSG by R-functions) in itself.

2.4.2. Visualization of F-Rep

Visualization algorithms are divided into two classes: polygonization [14], [9] based (for example marching cubes, marching triangles, adaptive polygonization, particle systems polygonization) and Ray tracing [14] based. The first class converts F-Rep into B-Rep, which is visualized by B-Rep approaches (Z-Buffer, for example). Polygonization is used not only for visualization but also for other modelling goals. The second class makes visualization directly from observer's viewpoint, without intermediate conversion. A basic operation in these algorithms is finding the nearest point in some direction (the intersection of a ray with a model). There exists a fast finding method, when functions are signed distance (normalized).

2.5. Other Representations Schemes

Other popular representation schemes are [7], [16], [5]: parametrized primitive instancing, spatial occupancy enumeration, cell decomposition, constructive solid geometry (CSG), sweeping, parametric and feature based modelling, etc.

The parametrized primitive instancing scheme [7], [5] is based on the notion of families of objects, each member of a family distinguishable from the other by a few parameters. Each object family is called a generic primitive, and individual objects within a family are called primitive instances. For example, a family of spheres is a generic primitive, and a single sphere specified by a particular set of parameters is a primitive instance. It is a simple and not particularly flexible scheme, but in combination with other schemes, it is suitable to describe the basic primitives included in the scenes. When adding a mechanism for constructing more complex objects based on these primitives, we remove one of the main weaknesses of this scheme. One such possibility is the combination of the scheme with the CSG scheme in a hybrid scheme.

The spatial occupancy enumeration scheme (Voxel) [7], [5] represents the solids as a set of spatial cells. The cells, also called voxels, are usually cubes of a fixed size and are arranged in a fixed spatial grid. Each cell may be represented by the coordinates of the cell's centroid. They can represent approximations of objects in the scene and can be used to improve the performance of algorithms. Very often they are used in combination with CSG.

The cell decomposition scheme [7], [5] describes a solid as a decomposition of several cells that collected together form the whole solid. The former scheme is a special case of this one, using a special type of cells (cubes lying in a regular grid).

The constructive solid geometry (CSG) schemes [7], [5] represent rigid solids as Boolean constructions or combinations of primitives via the regularized set operations. CSG and boundary representations are currently the most popular representation schemes for solids. They are commonly used as a hybrid scheme. The relatively simple data structure and the elegant recursive algorithms have further contributed to the popularity of CSG.

The sweeping schemes [7], [5] represent solids as a parametric union of primitives moving through space. They are applied to CAD systems, since they can easily describe rotary (or similar) solids. Sweeping schemes easily describe the removal of material by cutting tools.

The parametric and feature based modelling schemes [17], [7] are defined to be parametric shapes associated with attributes such as intrinsic geometric parameters (length, width, depth etc.), position and orientation, material properties, and references to other features. Thus, features have a semantically higher level than primitive closed regular sets. Features are generally expected to form a basis for linking CAD with manufacturing.

The particle systems schemes [7] – in the modern geometric modelling systems there is an increasing necessity of modelling natural phenomena such as rain, snow, fire, running water, clouds, wind, tornadoes, fluids, etc. The idea of the scheme Particle systems is to present these phenomena as a collection of many (thousands, and sometimes millions), small particles, which by some rules are added (usually at random), changed (moved, changed colour, etc.) and removed (typically based on the time elapsed after the creation of the particle) from the model. In this way models of highly complex special effects can be created.

From the discussion in this section, it is clear that there exist many representation schemes, each of them with different areas of application, power, usability, visualizing and processing algorithms and so on. Many good combinations of these (and only these) schemes are well known. To study all their possible combinations and to create hybrid homogeneous schemes is not an easy task. Homogenization has some advantages, but usually it is a manual and laborious process. This leads to the conclusion that this study of open hybrid non-homogenized representation schemes is relevant.

2.6. Hybrid Representation Schemes

In practice, representation schemes are rarely used in their pure form. The combination of two or more schemes for representing objects is commonly used to benefit from the advantages of both, or to achieve greater power and ease of representation. For example, CSG schemes are often combined with B-Rep, i.e. a scheme in which the description of the scene is CSG, and the final (primitive) objects are described by B-Rep. There are other common fixed combinations of schemes F-Rep/CSG [14], [3] F-Rep/B-Rep [11], and many others.

Such representation schemes are called hybrid [16] or non-homogeneous or a multi-representation.

Definition 4. *A hybrid representation scheme* is a solid representation scheme which describes the solids by combining two or more representation schemes.

Algorithms for calculating the properties of representations with such schemes are not easily implemented, because they must comply with the specifics of the scheme.

One approach is the so-called homogenization of the scheme [16], i.e. the scheme turns into a more complex scheme in which those characteristics of the original schemes are included that we want to be present in the homogenized scheme. By this approach we can get the most effective schemes, but they can no longer be regarded as any of the original schemes nor can they be easily extended with other systems and their capabilities.

Another possible approach is to do conversions between the representations, and then easily apply the algorithms, which are available for the resulting representation. The strength of the hybrid schemes would be increased if we could combine the advantages of the schemes, since some algorithms are more effective in some representations, other algorithms – in other representation, and so on. The representation conversion is a very important aspect not only in hybrid representations. Sometimes one has to deal with very difficult and time consuming algorithms. This requires that special attention be paid to those representations in a hybrid system. If possible, at the expense of more memory (excess information), more time efficient systems can be developed.

Definition 5. *An open hybrid representation scheme* is a hybrid representation scheme which describes the solids by combining a non-fixed finite set of representation schemes.

Typically, these representation schemes do not require homogenization, which is their strong point, but sometimes they need more than one conversion (during system operation) to some of the available representation schemes. This drawback can be eliminated almost completely with an appropriate caching mechanism at the expense of using more system memory.

2.6.1. The Search for Conversion between Representations

The conversion of representations is usually performed when there is an explicit indication by the user or there is a need to implement an algorithm that can only work with certain representations. This is often the case in open hybrid representation schemes because at any time a new scheme can be dynamically added to the set of representation schemes, and this distinguishes them from the classical fixed hybrid representation schemes, where the set and the conversion between representations are known in advance.

An example of an object with a current representation *Format 1* and a desired representation *Format 5* is given in Figure 2. Arrows indicate algorithms for direct conversion between representations available in the system. We see that in this example there are four ways to reach the desired format from the current one. These are: F1 → F2 → F4 → F5 (two paths, because there are two direct converters between *Format 2* and *Format 4*), F1 → F3 → F5 → F1, and F3 → F4 → F5. Also, if there exists more than one path/way of conversion, it is reasonable to find the fastest path or the one of the best quality (as well as a path satisfying a more complex requirement).

This is the well known problem of finding the best path in an oriented multigraph. It can be successfully solved by the Dijkstra's algorithm for finding the best path in a graph.

For the purpose of finding optimal paths for conversion, any algorithm for direct conversion has to have as an additional description a set of attributes with weights, evaluating its characteristics such as performance, quality of approximation, etc. Based on these attributes, one can establish criteria for evaluation and identification of the most appropriate path – for example, the sum of the attributes for performance of the converters in the whole path from the initial to the final format.

The results of the paths should be cached for later use without having to search. The key to identifying the converters must include the initial and final formats, quality criteria, etc.

When selecting data structures and algorithms in the implementation one

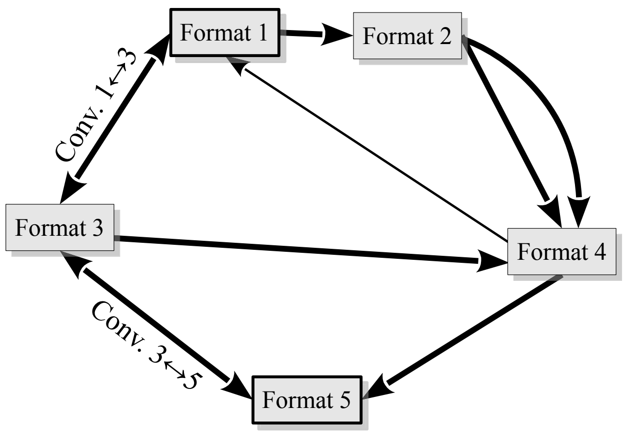

Figure 2: Finding paths of conversion between representations

must take into account the following characteristics: the number of representations is relatively small; there may be more than one algorithm for direct conversion between two representations; in a hybrid system the search of an optimal path for conversion may turn out to be a frequently used algorithm; the implementation of the conversion algorithms themselves may be much more complex and/or consuming more resources and/or running on a much larger amount of data than in the search for the optimal path of conversion. Furthermore, it is likely that the shortest path (i.e. the path with the least number of intermediate formats) is the fastest and the one of the best quality, therefore the use of the wave algorithm, split into two parts (and the use of the background task for the second part), is a good idea. The search for an optimal path continues until the first path (usually the shortest one) is found. After that, it is returned as a result and at the same time a background task is started to find the other paths. Apart from that, one can use another background task, using Dijkstra's algorithm or a similar algorithm to find all the best paths in advance and store them in the cache.

2.7. A Hybrid System – User Application, OpenGL, OpenCL, . . . , Hardware

The current state of development of computer graphics and its applications is unthinkable without the use of modern specialized graphics processors. They provide an opportunity most of the necessary calculations and processes to run faster and more parallel.

Lately, the emergence of General Purpose GPU (GPGPU) created new opportunities for more numerous and more flexible applications. The previously difficult RealTime Ray tracing [20] and the realistic kinematics simulations, fluid dynamics, global illumination in real time, etc., are becoming more real and even an every day experience.

The technological innovations opened the door for these hardware implementations in practice. Moreover, since long time ago no one can imagine the B-Rep applications and the related algorithms for visualization, simulation, and processing, in a purely software context without a graphics accelerator or GPU. With the arrival and development of GPGPU, after a while the situation will certainly be similar for a larger number of representation schemes, algorithms to work with them, and visualizations based on Ray tracing. This requires that any modern system for geometric modelling be seen as a set of software and hardware components with the increasing importance of the latter.

With the advent of OpenGL, the rapid development of applications based on B-Rep began. The corresponding graphics hardware evolved and became public. In recent years, the emergence of OpenCL [10], [12], which is a similar standard, led to the introduction of a large number more diverse parts of representations and algorithms related to them, and some new algorithms as well. Here the hybrid representation schemes will play a more important role. The development of hybrid systems using hybrid representation schemes becomes present. Until now almost all hybrid systems and almost all systems with hybrid representation schemes were somewhat fixed. The purpose of our investigations is to help change this, so that the systems, and in particular the schemes, become open and hybrid.

2.7.1. Open Hybrid Scene Graph Model

A geometric modelling system can run with more than one representation scheme without using a hybrid representation. It can support different representations and at some time work with one of them or have a single conversion of all representations to a basic representation. Such a system would have greater applicability and would provide its users with greater convenience, but the real capabilities of the hybrid schemes would offer even more than this. Therefore, systems that enable simultaneous work with multiple representations in a scene graph are called hybrid (in terms of the representation), and the systems that non-simultaneously support more than one representation – multi-representation systems.

The hybrid scene model can be defined using the design pattern Compo-

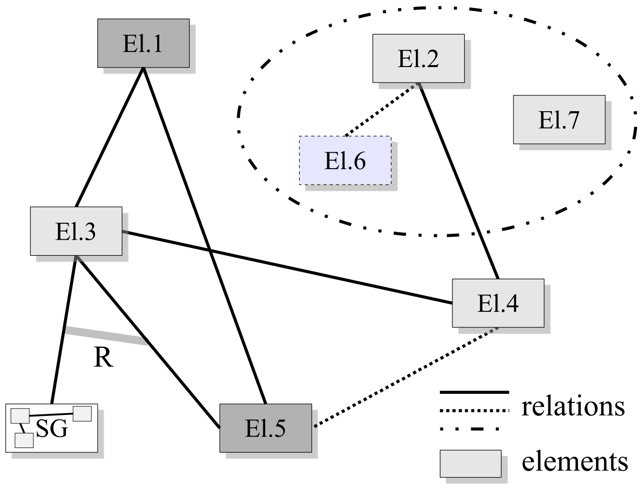

Figure 3: A hybrid scene graph

sition [8] as follows. We use a basic class/interface for a graphic/geometric element (object). Its successors will be all kinds of graphic/geometric elements and relations (i.e. the relation is also an element or the graphic/geometric element is a relation of its points), and one of the main successors is a scene graph, i.e. a special type of a composite element containing and managing (effectively) sub-elements of the base class (or implementing the base interface for a graphic/geometric object). This means that the scene graph is a relation and it gives a hierarchical structure to the scene, and in addition it may contain as a sub-element other scene graphs, i.e. sub-scenes. In other words, the scene graph is a set of objects (with different representations) and relations (over one or more elements, including relations over relations, etc.). This allows for easy adding of new elements/representations/ relations, an arbitrarily complex structure of the graph, as well as a uniform way to work over elements and relations. The type of representation is determined by whether the class of the element implements a certain interface. Defining an interface for any representation gives the opportunity to include the element in more than one representation at the same time, and to hide the internal data structure that describes it.

Figure 3 shows an example of a scene graph. The elements are represented by rectangles, and the relations – by lines connecting or encircling the elements, which are in a relation.

The elements El.1 and El.5 are in one representation, for example F-Rep;

El.2, El.3, El.4 and El.7 are in another, for example B-Rep; El.6 is in a third type of representation. The element SG is a scene sub-graph, i.e. a sub-scene. This division allows for geometric modelling systems to load into the memory only a part of the scene based on some criteria (such as visibility, i.e. only those parts of the graph that are visible to the viewer at the time, or only those parts of the layers that divide the scene, or only the first level of the scene without the sub-scenes; in the last case the sub-scenes are loaded when needed or on command from the user, etc.). One can introduce different types of elements (not just with different representation) that behave differently, for example proxy elements that load the sub-scenes when a password is set by the user, etc. Some elements may be a "cache" of other elements and be used if necessary; for example, if an F-Rep element has been polygonized, its corresponding element in B-Rep can be added to the scene (without visualization) and subsequently be used without polygonization. Other types of elements are the so-called elements decorators and elements adapters that can modify a certain element or bring another element into the system – one that exists but is not intended for use in this system.

In the scene (the scene graph) there might be different relations, and their type and number are not limited. For example, in Figure 3, El.1 and El.3 are in one type of relation, El.3 and El.5 are in a second type of relation, and so on, El.4 and El.5 are in another (again binary) type of relation. El.2, El.6 and El.7 are in a relation, which is not binary (i.e. they are grouped). There might be unary relations. Examples for types of relations are: group (number of elements considered by the system as one, or connected and inseparable from one another), cache (one element is a cache of another), version (one element is a newer version of another, i.e. the relation "memorizes" the change of the element in the scene), etc. In Figure 3, R denotes a relation between two other relations. If the application requires it, relations between elements and other relations may be established (this is not surprising, since the model distinguishes between a graphic object and a relation only formally, i.e. they are all elements of the model). Of course, their meaning and usage depend on the semantics of the model and the specific application.

2.7.2. Interaction with an Open Hybrid Scene Graph Model

How to modify the model significantly depends on the representation scheme chosen for its description. For example, in B-Rep we can "catch" the vertices of the solid and move or remove them, etc. In CSG we can change the constructive operations applied over the primitives, add or remove primitives, etc. In F-Rep we have functions that we can modify by superposition and composition, we can deform the solids (by applying specific functions over the existing ones), we can add or remove solids, etc. Even in the same representation scheme, depending on the specific application, the user can work with different terms – in a CAD system the solids are details, in a system for creating games the solids are players.

As we can see, the terms and methods that the users apply may vary, but most of them ultimately come down to the same. Whatever the representation scheme is, there always exist the data operations that are performed by users in the same way. This is not surprising because all the representation schemes describe the same thing – geometric information. As we have shown, the geometric information has mainly three types of parameters – shape, metrics, and location/orientation in the space. Hence, whatever the representation is, the user has to be able to change these characteristics of the model.

The interaction in the hybrid models cannot be defined as in the case of a model (a scene) with a strictly defined homogeneous composition and structure. To reach the general idea of the interaction valid for the hybrid models as well, we will consider in detail some special cases (B-Rep and F-Rep), and based on them we will summarize our results, so that the summary is valid for both homogeneous and open hybrid models.

If we only have a boundary representation (B-Rep), then the composition of the scene consists of vertices associated with edges, defining faces, of which the solids and the whole scene are built (e.g. vertices, edges, triangles, etc.), and the structure of the scene is determined by a combinatorial structure, describing the correlation between these elements (there may be grouping of the solids, etc.). In this case the interaction occurs as the user selects the elements (groups, solids, faces, edges or vertices) and modifies their characteristics (location, colour, etc.). The user can create and remove parts of the model (the new vertices, edges, faces, solids, etc.). Besides these simple operations over the model, one can apply many more complicated operations that work over the model (the whole model or only a part of it) – operations that are specific to the area of application, or are borrowed from other representation schemes (union/intersection of solids, finding rotary surfaces, etc.). After the change of the scene (or a part of it) a new visualization is triggered.

If the model is F-Rep, then in the general case it is a function (which may not be defined analytically). Here we shall consider the case when a collection of functions is given, and the scene is a set of all of them. This is not very different from having only one function (because we can always combine them into one using R-functions), but is closer to the logic of the user experience. In this case, the scene consists of the functions (describing the solids or parts of the scene), and the structure of the scene is determined again by the functions and their composition. It is possible that the functions are composed of sub-functions (a composition and a superposition of functions) and/or are associated with R-functions, giving the set-theoretical operations over them, thus introducing an additional structure in the model. In this case the interaction occurs as the user works with the model (the functions) as a whole, or modifies them by making superpositions of functions, existing in the model, and other new functions. The functions that exist in the model may become parameters of the newly added (for example, if we intersect two solids, this means to add a new R-function that has two parameters and they are the solids/functions being intersected) or the modification may be a replacement of a parameter or a sub-function of some of the existing in the model. Characteristics such as shape, location, colour, etc. can be varied for each point of the solid or the solid as a whole. The user can create or remove parts of the model (new functions). Since there are no specific points (as the vertices in B-Rep) and parts (as the edges and faces in B-Rep), then all parts in a solid are "equal" in terms of the operations applied on them, i.e. the user can select any point on the solid and deform it by using the point with some nonlinear transformation. For convenience, a grid can be displayed on the model (UV-lines, working curves and planes, etc.) for easier and more intuitive work with the solid. After the change of the scene (or a part of it) we proceed analogously to B-Rep.

Comparing the interaction approach with B-Rep and F-Rep and distancing ourselves from the specifics of each of them, we obtain the following scheme that can be applied when interacting with a hybrid model:

- Each representation has a set of operations that can be executed over its elements;
- If any of the algorithms cannot be applied to a given representation, we may try to convert it to a representation on which the algorithm can be applied, and then transfer the result back (if possible);
- If an operation is applied over multiple solids, then the above point is applied for each of the solids;
- The system must use any available converters between representations to expand the possible operations over the different representations, without them having to be implemented specifically for each of the representations;
- In case the model is changed it must be visualized.

3. OpenF – An Open Hybrid Geometric Modelling Framework

OpenF is a prototype of an open hybrid geometric modelling framework, designed to study such systems. It is an open source system, composed of an object-oriented framework for hybrid geometric modelling, documentation, and numerous sample programmes demonstrating how to use some of its applications. One of the main objectives of developing OpenF is to help research and training in certain aspects of CG.

3.1. Principles

The system has the following characteristics:

- Openness – the possibility for expanding the system in one or more directions. This is achieved through (see Figure 4 and Figure 5):
  - Inclusion of new representations;
  - Addition of converters between representations, called converters;
  - Addition of logical input, output and input-output system elements, called Sources, Targets, Storages;
  - Inclusion of application elements in the system;
  - Inclusion of compound elements (sub-systems) as elements of the system;
  - Inclusion of communication elements in the system – new types of inter-element communications.
- Hybridness – the availability and the possibility for simultaneous work with more than one inner representation; the use of compatible software and hardware resources in order to fully exploit the modern graphics (and other) hardware;
- Flexibility – the easy adaptation to the different applications; provided by a powerful configuration sub-system according to the applications;
- Distributiveness – the simultaneous work of the system parts on different computer systems; the possibility for realizing elements as software, hardware or a mixture of both;
- Multi-user orientation – the possibility for parallel (or/and simultaneous) work of many system users.

All these features influence the system architecture.

Further, the system architecture allows the system developing to be:

- In stages – to construct the system in separate consecutive stages so that after each of them the system is efficient at the level of the particular stage; there are three stages in the system development – system stage, applied stage and user stage;
- Done by many developers (teams) – to achieve this characteristic we should have loose coupling among the sub-systems, clearly described ideology for development, documentation, open source construction of the system, etc.;
- Independent of the programming language or the operating system – the system architecture and its implementation must not be based on means and technologies related to a particular programming language or an operating system.

The solutions to these problems lie in the use of modern means and programming techniques (methods).

Finally, the desired system should be suitable for research, applied, and educational purposes.

The above requirements influence both the architecture and the stages of the system development.

3.2. Architecture

System elements could be considered on three abstract levels: logical, conceptual, and physical. On the logical level the elements are elements of a system – atomic elements, sub-systems and relations. On the conceptual level we have a core, extension modules, plugins, set of services, inter-element communication and relations, etc. On the physical level we have classes, methods, interfaces, dynamic libraries, etc.

The general scheme (Figure 4a) of the system architecture consists of three layers: a core, expansion modules (plugins), and applications.

Geometric information, as a special type of information, is subject to the same types of information processing as any other information. Therefore, all major processes in the system for geometric modelling can be classified as processes for collecting, processing, storing and distributing geometric information or data describing this information.

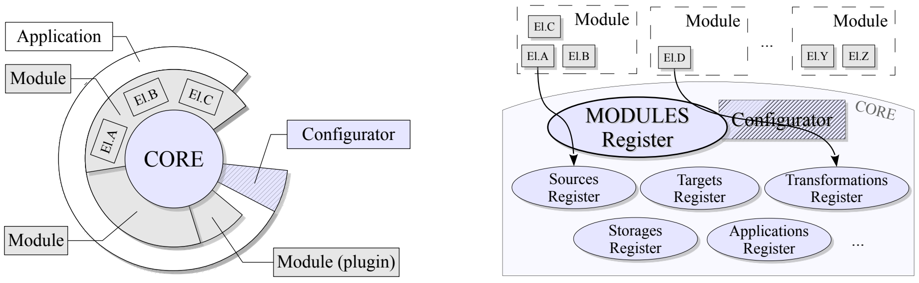

Figure 4: (a) Architecture of an OpenF. (b) Registration of the elements in the system

Main kinds (types/classes) of elements of the system (Figure 4b) in accordance with their functions in it are: Sources, Targets, Transformation-elements, System-core, Registration-elements, and Configuration-elements, etc. Sub-system configuration is a very important element of any modern software system. In the OpenF architecture a special attention is paid to this sub-system.

In Figure 5, we show the possible cases for the distribution of the elements of the system. They can be included in one computer system or distributed in a network. Each host in the network has its own core and necessary items, and the communication between the systems is realized by a core sub-system. The elements can be both software and hardware implementations (communication between the software and hardware part of each element is performed either by a message exchange or by specialized API, such as OpenGL, OpenCL, etc.).

The goals of the system and its architecture determine three stages of development:

(I) System stage – creating a core (registration-elements, configuration-elements, communication, transformation, statistics, visualization, interaction, storage, supporting sub-systems, etc.);

(II) Applied stage – developing basic elements (converters, communications, etc.);

(III) User stage – developing and using the system.

With respect to realization, each of the last two stages includes two aspects depending on whether new capabilities (system realization) are realized or already existing ones are adjusted and used (applied realization).

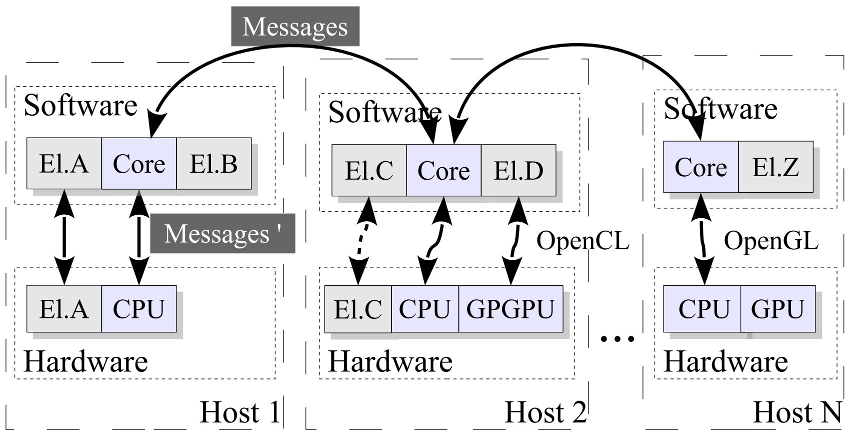

Figure 5: Distributed architecture of a multiuser system for geometric modelling

3.3. Implementation

The realization of the prototype system (OpenF) is done in C# and uses some basic classes of the .NET framework. The choice of this language is due to several reasons. One of them is the fact that the .NET platform is portable among the most widespread operating systems. Another advantage of .NET, which is essential for our requirements to the system, is that it does not limit the programming languages used for writing applications and for OpenF. However, the main advantage is that the language C# provides great tools for simple and clear description of the algorithms using a well-developed object model. Of course, the choice of the .NET platform has not only advantages. The main disadvantage is that at least for now the JIT compiler of .NET is not sufficiently oriented towards algorithms with large-scale computations. The performance of the systems for geometric modelling is essential for them. Overcoming this deficiency can be done in two ways – using newer versions of the JIT compiler (which are still not available, or just recently came out), or using additional resources for preliminary (automatic) processing of the programmes and libraries of the OpenF framework, in order to achieve better performance without changing the source code. The latter approach was chosen for the framework OpenF. The idea is to use the compiled code of the system as a model to automatically generate a code from it that is completely optimized (OpenCL, C, etc.). Of course, this converter is not universal, but corresponds to the specifics of the application domain and the limitations of the particular implementation of

OpenF.

The implementation of the prototype with OpenF is accomplished by applying many good practices and design patterns [8] used in modern object-oriented programming.

3.3.1. Modelling

The modelling in OpenF is based on the concept (given in sections 2.6 and 2.7) for an open hybrid representation scheme, realized as an open hybrid scene graph. Unlike other similar systems/frameworks, in OpenF there is no problem-oriented language for description of the scenes. Defining another specialized language would be pointless and would not help to realize the objectives of the system. Furthermore, programming languages (like C#, etc.) are a sufficiently powerful and convenient tool for describing not only the models, so duplication of functionalities in another problem-oriented language is not justified. OpenF is a framework for creating open hybrid systems by programmers, not a file format or a scene description language. If it is necessary to save the model in a file, one can use serialization (binary, XML-based, etc.) or generate a programme (in IL), which creates the model in the memory.

3.3.2. Extensibility

To be easily expandable a system must have a clear and powerful mechanism for adding functionality. This can be done in different ways, but one of the most popular nowadays is the plugin module mechanism. It is selected as the main mechanism in the framework OpenF. Each plugin module can extend the system with any number of classes defining new functionality through one or more services (interfaces that the objects implement). Loose coupling among the plugin modules and the algorithms that use the services is implemented by a sub-system of the core of OpenF. Logically it is composed of multiple registers that store the objects providing certain services.

In practice, the core of OpenF must contain only this sub-system, and all the other functionality (even the one of the core) is optional and is implemented as plugin modules. There are two types of modules – core modules and system modules.

3.3.3. Visualization

Another major sub-system (a set of services) of OpenF is the sub-system for visualization of the hybrid model.

As we usually have three-dimensional representations of solids, in these cases we need to have some kind of projection of the model on a plane. In geometry there are many projection methods which maintain different characteristics of the projected object, the most important of them – the illusion of three-dimensionality. Important features of the visualization algorithms are their speed and quality (i.e. how close the image is to the original/model). These two features are contradictory, i.e. different algorithms tend to compromise one of them for the other. Of course, all algorithms have their place and use, because the purposes of visualization can be different and therefore one algorithm may or may not be suitable to achieve them.

Generally, the visualization algorithms can be divided into two types determining the design of the sub-system for visualization in OpenF:

- Visualizing from the spatial scene (Figure 6a), i.e. working in the object space – For these algorithms every object (or the part of it that is in the clipping volume) is projected onto an image plane, and then displayed in the viewport (if we want a raster image, the image is rasterized). The performance of these algorithms depends on the number of objects in the scene (or the square of their number in some algorithms). The quality may be very good, as images can be scaled arbitrarily, but it depends on the representation of the objects. The Z-buffer algorithm is a typical representative of this class.

- Visualizing from the observer (Figure 6b), i.e. working in the screen

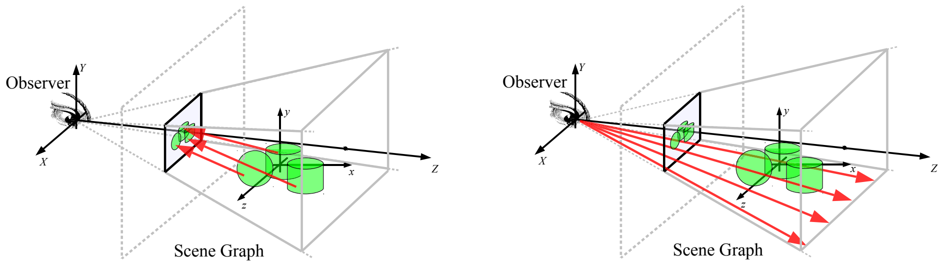

Figure 6: (a) Visualization from the scene. (b) Visualization from the observer

area – With these algorithms one usually gets a raster image, and it is determined for each pixel which object is projected into it. The viewport is divided into cells and each of them is assessed. The performance of these algorithms depends on the number of pixels in the viewport (and to some extent on the number of objects). The higher the number of the pixels of the viewport, the better the quality, i.e. we can get the quality we need. These algorithms very little depend on the representation of the objects. The Ray tracing algorithm is a typical representative of this class.

Of course, there may be a mixed approach, in which the algorithm works bidirectionally.

3.4. Documenting

In [5] the methodology for geometric modelling is described. It is based on the logical separation of the system functionality for processing geometric information. Each functional group is called a virtual machine or a processor. Each virtual machine handles a specific part of the graphic information as it processes it and interacts with other virtual machines. Their purpose is to encapsulate relatively independent and similar operations. They communicate with each other via preliminary defined interface and eventually pass processed data. This hierarchy of virtual machines consists of: Display processor, Geometric processor, Structure processor, Semantic processor, Dialog processor. This approach was used to develop a prototype of a specific open hybrid system for geometric modelling. This system is a part of the samples of OpenF, and a part of its documentation.

Besides this application, as part of the documentation, numerous small sample programmes were implemented, demonstrating various aspects of the framework functionality.

3.5. Applications

As already mentioned, OpenF may be used for research, educational and applied purposes.

The use of OpenF for research purposes has already been discussed. At this stage, it mainly includes the study of open hybrid models, the interaction with them, the visualization, and a less general study of the properties and applicability of F-Rep.

Here we will focus on educational purposes and applications.

OpenF is highly applicable in various fields (thanks to properties such as openness and flexibility, its architecture and method of construction). Education is only one of them. It is not recommended that the content of the basic course in computer graphics be expanded and complicated, so OpenF currently finds and will continue to find (more) applications in the accompanying forms of education.

Here are some possible applications in education: seminar classes using the libraries of OpenF as a base for developing projects in CG (elective courses); extracurricular activities with outstanding students interested in CG; individual investigations of students studying the samples, documentation, and the object-oriented design of OpenF; graduation theses – development of systems, applications, etc.; an illustration of (electronic) tests and materials for self-learning; an illustration of the teaching material using visualization of three-dimensional (and two-dimensional) models in different areas (mathematics, informatics, chemistry, physics).

Other applications are possible, but their practical implementation depends on the students' interest in the subject CG, and the new trends in it as a whole. During the academic year 2006/2007 a survey explored the students' attitude towards the course CG and their receptiveness to applying innovations. An object of the survey were the third year students who major in informatics, and the fourth year students who major in mathematics and informatics at the Faculty of Mathematics and Informatics (FMI) at Plovdiv University "Paisii Hilendarski". The survey results were compared with some of the data related to the system for quality assessment, introduced a few years ago at FMI. Our survey showed that the students are interested in CG. We also reached other important conclusions about the teaching and learning process.

A practical aspect of using OpenF is that the framework can be used to create a variety of applications. The first ones are already in the process of design and development. We mean not purely commercial products, but rather experimental prototypes, aimed at creating hybrid systems for geometric modelling that subsequently, except for research purposes, could develop into systems of general application. At the moment there are two applications based on the OpenF prototype with graduate students involved in their development: OpenStudio (a demonstration hybrid system for geometric modelling based on open hybrid representation scheme) and F-Rep Designer (a system for geometric modelling, based on F-Rep, that uses for visualization an OpenCL based Ray tracer, as well as an optionally analogous to it C# based Ray tracer).

4. Results

The related theoretical background was studied and documented. A concept for open hybrid system for geometric modelling, based on an open hybrid representation scheme, was created. Architecture, object model and a prototype of the framework OpenF, which implemented the concept in practice, were established. The framework was documented with samples and descriptions of its modules, classes and methods. A system of test cases was created and the framework was tested with it.

An open hybrid model for the description of geometric information was designed and implemented.

A flexible mechanism for visual manipulation of such models was proposed and implemented in practice, applying practices for hybrid usage of the modern (graphics) hardware.

By using the framework OpenF, a system was created called OpenStudio – a sample demo open hybrid system for geometric modelling based on an open hybrid (non-homogenized) representation scheme.

5. Conclusion

The achievement of the main objective/goal of this work, namely to consider the implementation of an open hybrid system for geometric modelling, based on an open hybrid (non-homogenized) representation scheme, was demonstrated by building and testing a prototype of the framework OpenF. By using this prototype, a simple demonstration experimental system was built. This is a constructive proof of the goal and shows in practice how such systems can be created.

The system architecture of OpenF provides the desired qualities – openness, hybridness, flexibility, and distributiveness. It can be run by many users and developers.

The object model and the implementation follow the best practices for creating object-oriented systems.

The documentation and the samples are easily accessible and understandable. The samples are subject to development and expansion in order to construct an even better system of basic templates, ready for use in new applications.

A system of test cases, which ensures maintaining the efficiency of OpenF while being modified and developed, is created. Of course, this system of test cases should be used and developed together with the system.

The programme implementation of the framework OpenF in this work mainly includes the system stage, the applied stage (system realization), parts of the applied stage (applied realization), and small parts of the user stage.

Prospects:

The open architecture and the ideology of the framework OpenF enable its large improvements in different directions. Future research and development should be directed towards:

- Improving F-Rep and developing other less common representations;
- Studying the interaction between the user and the hybrid model as a whole;
- Improving visualization of hybrid models using specialized hardware (supporting OpenCL, Ray tracing based, etc.);
- Creating a library of problem-oriented sub-systems (component approach), ready for inclusion as blocks in the systems.

In these directions the framework OpenF with its characteristics, such as openness and hybridness, will help to further the research and applications of the open hybrid representation schemes in CG.

Acknowledgments

The work is partly funded by Fund Research of Plovdiv University under contract NI13 FMI-002.

References

[1] D. Bedworth, M. Henderson, P. Wolfe, *Computer-Integrated Design and Manufacturing*, McGraw-Hill Inc., New York (1991).

[2] *Blender*, May 2013, http://www.blender.org

[3] R. Cartwright, V. Adzhiev, A. Pasko, Y. Goto, T. Kunii, Web-based shape modeling with HyperFun, *IEEE Computer Graphics and Applications*, **25**, No. 2 (2005), 60-69, **doi:** 10.1109/MCG.2005.49

[4] K.H. Chang, *Product Performance Evaluation Using CAD/CAE: The Computer Aided Engineering Design Series*, Academic Press, Oxford (2013).

[5] D. Dimov, *Computer Graphics*, Plovdiv University Press, Plovdiv (1999). (in Bulgarian)

[6] C. Elanchezhan, S.T. Selwyn, S. Sundar, *Computer Aided Manufacturing*, Laxmi Publications, New Delhi (2008).

[7] J. Foley, et al., *Computer Graphics: Principles and Practice*, Addison-Wesley Professional, Boston (1995).

[8] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Pearson Education, Boston (2004).

[9] M. Kazakov, A. Pasko, V. Adzhiev, *Fast Isosurface Polygonization with Embedded Sharp Features Extraction*, Technical Report HCIS-2003-04, Hosei University, Tokyo (2003), http://cis.k.hosei.ac.jp/TR/HCIS-2003-04.pdf

[10] Khronos OpenCL Working Group, *The OpenCL Specification Version 1.2*, Khronos Group, USA (2012), http://www.khronos.org/registry/cl/specs/opencl-1.2.pdf

[11] D. Kravtsov, O. Fryazinov, V. Adzhiev, A. Pasko, P. Comninos, Real-time controlled metamorphosis of animated meshes using polygonal-functional hybrids, In: *ACM SIGGRAPH ASIA 2010 Sketches (SA '10)*, ACM, New York (2010), Article 36, **doi:** 10.1145/1899950.1899986

[12] J. Lee, et al., An OpenCL framework for heterogeneous multicores with local memory, In: *Proceedings of the 19th international conference on Parallel architectures and compilation techniques (PACT '10)*, ACM, New York (2010), 193-204, **doi:** 10.1145/1854273.1854301

[13] *Maya API Guide*, May 2013, http://docs.autodesk.com/MAYAUL/2013/ENU/Maya-API-Documentation/index.html

[14] A. Pasko, V. Adzhiev, A. Sourin, V. Savchenko, Function representation in geometric modeling: concepts, implementation and applications, *The Visual Computer*, **11**, Issue 8 (1995), 429-446, **doi:** 10.1007/BF02464333

[15] *POV-Ray*, May 2013, http://www.povray.org

[16] A. Requicha, Representations for rigid solids: theory, methods, and systems, *ACM Computing Surveys*, **12**, No. 4 (1980), 437-464, **doi:** 10.1145/356827.356833

[17] D. Roller, An approach to computer-aided parametric design, *Computer-Aided Design*, **23**, Issue 5 (1991), 385-391, **doi:** 10.1016/0010-4485(91)90033-S

[18] V.L. Rvachev, *Geometrical Application of Logic Algebra*, Technika, Kiev (1967). (in Russian)

[19] V.L. Rvachev, *R-Function Theory and its Applications*, Naukova Dumka, Kiev (1982). (in Russian)

[20] J. Schmittler, S. Woop, D. Wagner, P. Slusallek, Real time ray tracing of dynamic scenes on an FPGA chip, In: *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware (HWWS '04)*, ACM, New York (2004), 95-106, **doi:** 10.1145/1058129.1058143

[21] V. Shapiro, Theory of R-functions and applications: a primer, *Computer Science Technical Reports*, TR91-1219, Cornell University, New York (1991), **doi:** 1813/7059

[22] R. Wang, X. Qian, *OpenSceneGraph 3.0: Beginner's Guide*, Packt publishing, Birmingham (2010).