VRML as means of expressive 4D illustration in CAM education

Nikolay Avgoustinov*

Institute of Production Engineering/CAM, University of Saarland, Postfach 151150, 66041 Saarbrücken, Germany

Abstract

Computer graphics (CGr) lies in the fundaments of computer aided engineering (CAE). As a result from the fast development in both CGr and CAE, more and more computer aided systems (CA-systems) have been ported on low-end and personal computers. Despite the continuous drop in the CA-system prices, this type of software is still very expensive compared to typical office applications. This paper discusses the possibility to use VRML for scientific, educational and even industrial purposes. As an illustration, it is shown how VRML can be used as inexpensive means for simulation of one of the most interesting but also most time and resource consuming areas of the computer aided manufacturing (CAM) — machining of complex parts. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Verification of NC-programs; Education; Simulation

1. Background and motivation

Numerically controlled (NC) machine-tools are expensive devices, which means that operators or students should be allowed to use them only after acquiring practical skills. But for acquiring practical skills and getting familiar with any machine one should work with it, so that is a vicious circle. For this reason, most manufacturers of NC machine-tools offer so-called off-line programming systems. These systems allow for acquiring some practical skills that concern mainly the programming and operation but neither machining nor shaping. On the other hand, the users of NC machine tools have always had the wish for extremely cheap, fast and safe means for proof-testing each new NC-program before putting it in real use. While searching the ideal solution, different means and methods have been tested. Among others were substitution of a pencil for the cutting tool and a piece of paper for the workpiece and the use of soft material (wood, plastic, etc.) instead of the original material, allowing to decrease the proof testing time and costs. Employing such extremely cheap, fast and safe means for educational purposes is even more important, since:

1. The process of learning implies many errors, but they should neither cause a danger for people or equipment, nor increase the costs.
2. Learning is optimal if the provided environment is maximally close to the expected real conditions.
3. Errors during manufacturing often lead to extremely expensive damages and are undesirable, but causing them during the education intentionally, for illustrating the respective consequences, is very educative if no additional expenses arise.

While searching for well-suited means, the author came to the idea of a VRML-based (cf. Ref. [3]), network-oriented modeling system. The desired capabilities of this system are:

4. To be able to illustrate the information flow within the CAD–CAM–NC process chain.
5. To be able to visualize certain parts of this information and to assist in their management (cf. [4,5]).
6. To be able to simulate the processing of a workpiece or even the whole work of a NC-machine tool on the basis of NC-code with arbitrary origin (cf. [2,6]),

7. To allow for playing and comparing the results of different “what if” scenarios.

The idea of graphical verification of NC-programs is not new, but the large number of NC dialects and the lack of means for realistic, efficient and affordable visualization of data and processes in 3D hindered its (full) implementation for a long time. With the invention of the Knowledge-based Adaptive Conversion Kit (KNACK, see [1]) the first obstacle (the large number of NC dialects) was overcome. The first version of VRML was already a good approximation of the sought-for affordable means for data-visualization, and the improvements in its second version (appeared in 1996) provided also means for visualization of processes (animation, events, dynamical change of objects). At that time all obstacles were (at least theoretically) overcome. So the way to simulations in 4D (3D + time), which is of crucial importance when simulating manufacturing processes, was free and in 1997 at the Institute of Production Engineering/CAM of the University of Saarland a project for verification of NC programs via their simulation in virtual reality (in short virtual shaping, see [2]) was started at the Institute of Production Engineering/CAM of the Saarland University.

2. Approach

2.1. Illustrating of (NC) machining

From educational point of view the following areas of interest can be distinguished in this process:

1. visualization of geometry and, mainly, the tool trajectories;
2. visualization of the involved objects (tools, tool-holders, workpieces, clamps);
3. simulation:
   - visualization of the machine-tool behavior and illustrating how different NC commands influence it;
   - visualization of the interactions between the involved objects as well as other components of the process;
   - visualization of machining and/or shaping processes (virtual shaping).

Now let us consider where VRML could be helpful.

2.1.1. Static visualization (paths)

The simplest application of VRML is to use it for static visualization of tool paths as illustrated in Fig. 1. Through the “walk” and “examine” functions of any VRML-browser the students can explore and investigate the tool-trajectories. Further, they can either try to guess what shall be produced by the respective CLDATA or NC-program, or can compare these trajectories with an already machined part and analyze the achieved results.

The main problem of the static visualization is that the plain conversion of a NC-program into static geometry produces too congested scenes. Therefore, these scenes do not help much to estimate the result of running the respective NC-program on a real machine-tool (cf. Fig. 1).

Actually, such static visualization is still far from simulation of the work of a NC-machine or simulation of the machining process (virtual shaping). For simulating the latter we need (apart from the mentioned trajectory of the tool) VRML models of the workpiece and the cutting tool. For simulating the work of a NC-machine, we need (in addition to the already mentioned models of tool and workpiece) a VRML model of the whole machine tool with its carriages, saddles, revolvers, shields, magazine, etc. Preparing all mentioned models is possible, but if the estimated development effort for both simulations is compared to the expected profit from the use of the respective simulation, virtual shaping is the clear winner. The simulation of the whole machine-tool is very interesting from an educational viewpoint, but since its implementation would improve the verification of NC-programs only inappreciably, the initial research was restricted to applications of virtual shaping.

2.1.2. Tool modeling

Cutting tools are manufactured in a great diversity. For choosing the tool that is best-suited for a given processing, the designer or the NC operator has to consult manufacturers’ catalogs. Unfortunately, such catalogs are very voluminous and most of them are (still!) simply “paper” versions. Compared to any tool from a “paper” catalog, the VRML model laying
behind the cutting tool illustrated in Fig. 2 has the following advantages:
1. It can mimic almost any used tool-geometry through the seven parameters used for generalized description of cylindrical tools in CLDATA (DIN55215) and other similar languages.
2. It can be viewed from different viewpoints, perspectives and zoom factors.
3. It can simulate the rotation and the movements of a cutter in a real machine tool and even the changes in the respective speeds.
4. Even sound (proportional to the turning speed and to the depth of cut) can be produced.
5. It can be easily combined with other objects in scenes for illustrating the machining.
A similar generalized model can be developed for cutting tools for turning.

2.1.3. Dynamic visualization (path-walk)

The next step was to make the model of any tool walk along the defined in a NC-program path using also other parameters as feed rate, turning speed, etc. If additional objects from the real environment are also modeled and included in the scene, a path-walk would allow a visual control for collisions of the tool with them. Although VRML itself provides for collision-checks only between avatar and other objects and not between two non-avatar objects, such additional functionality was achieved through additional scripts. No illustration is attached to this topic since the paper cannot represent the dynamic of the respective model and a static snapshot would be similar to “superposing” of Figs. 1 and 2 (with the cutter-tool respectively positioned).

2.1.4. Workpiece modeling

Since the blanks (or workpieces) have a relatively simple geometry before the (NC) machining, it is not difficult to represent them in VRML. The most
often used shapes are cylinder, prism (box) and cone. Modeling them just statically, though, was considered to have only a modest added value for the educational process. Therefore, the accent of the work was shifted to providing these geometric primitives with additional functionality: parameterized geometry, positioning (translation and rotation) capabilities, detection of collisions.

2.1.5. Virtual shaping

The virtual shaping is entirely based on collision detection. The simplest possible action after a collision of the tool is detected would be to indicate it. This can be achieved by a (crash) sound, (red) color or by simply “stopping the dynamic” (freezing) of the visualization. Much more useful (and more difficult to implement) is to give feedback for locating the line causing collision in the source NC-program. In our case, eventOuts were coded in the PROTOs of clamps and workpieces, so that it is possible to get the position of the point \((x, y, z)\) and the time of collision. Depending on what type of object the tool collided with, the further processing is performed in two possible directions. Collisions between tool and a non-workpiece object are processed as “undesired”. The position of the collision’s point is typically routed back to the tool’s PROTO for determining the number of the point in the current Coordinate node, and this number is used in turn for locating the source line causing collision. Collisions between tool and workpiece are considered “desirable” or intended and should be processed from the scripts of the workpiece’s PROTO.

The challenge of a VRML-based simulation was to test whether the use of events and inter-object communication (e.g., eventIn and exposedFields in geometry nodes like spine in Extrusion, heights in ElevationGrid, etc.) would allow to simulate the dynamic interaction of the tool with the workpiece and the changes in the form of the material resulting from this interaction. As already mentioned, since both the material and the tool used are virtual, the process was named virtual shaping. Of course, it is important to allow not only slowed-down, but also real-time and speeded-up simulations so that each viewer could concentrate on those aspects of the process which are most important for him.

2.2. Implementation

So on the one hand we have the modern and attractive capabilities of VRML for visualization and simulation. On the other hand there are large amounts of (already existing) geometrical and technological information contained in NC-programs (e.g., cutting speed, timing, use of several tools with synchronous control of two or more of them, etc.). Obviously, VRML and the NC-programs could complement each other and compensate each other’s disadvantages. Thus, powerful CAD-systems could act indirectly (through the generated NC-programs) as VRML authoring tools and the VRML browsers would act as a means for NC-programs’ simulation and visualization. Consequently, similar complement would not only allow to proof-test NC-programs, but also would present an inexpensive means for investigation, illustration and even advertisement of technological processes.

Deliberating upon how to combine them together the author came to the scheme shown in Fig. 3.

Following this scheme, the converters in Fig. 3 prepare the most voluminous part of the information: NC-geometry. This information is converted into static VRML-geometry (e.g., Coordinate nodes are created from the input geometry and visualized as IndexedPointSets or IndexedLineSets). This static geometry, loaded into the appropriate browser, can be used for viewing the trajectory of the respective tools from different perspectives and comparing its topology with the expected one. Models of clamps, tools or other equipment, designed in any CAD system, can also be converted into VRML (static) geometry. The parameterization of the models, achieved by means of VRML-PROTOs, increase the reusability of the models and thus reduce the need for authoring work. However, the most important and interesting aspect of the approach — to liven up the scene and allow dynamic visualization (path walk) or virtual shaping — is achieved through incorporating additional functionality in the models by means of scripts.

The two most important for the virtual shaping models are those of the workpiece and of the tool. Before discussing these models (respectively — the PROTOs for them), let us recall that the workpieces are generally classified (according to their form) into two
main types — rotational and prismatic workpieces. Both types have their own peculiarities, therefore, their models should be based on different VRML nodes. More information about possible modeling of these two types is given in [2], together with some examples.

Now let us consider the simplest scenario: a workpiece and a tool moving along the coordinates saved in some Coordinate node. In order to simulate the shaping process, all collisions between the tool and the workpiece have to be detected and used to change dynamically the shape of the latter. Since any cutting-tool is much harder than any workpiece, the volume of the former should be “subtracted” from that of the latter. The problem here is twofold. On the one hand, cutting processes imply “sustained” collision (i.e. up to 99% of the tool-movement-time). On the other hand, for providing a reasonable quality the above-mentioned subtraction has to be performed (together with the respective visualization) many times per second — between 16 and 24. Very few of the modern CAD-systems (if at all) are capable of providing similar performance, let alone the average computer on the Internet. Moreover, the calculation power requirements grow exponentially with the increase of the complexity. The conventional algorithms for Boolean operations are obviously not applicable in this case, and some substitution has to be looked for. In this sense, yet the choice of workpiece’s and tool’s representation is crucial for the success of the whole simulation.

The requirements to the tool are simpler, so let us start with them. The geometry of a simplified mill has only two parameters — radius and height (cf. Table 1). The mill-object is controlled through \texttt{set\_stop} and through its technological parameters (shown in Table 1 as numbered comments and received indirectly through \texttt{NCPath} instead). The \texttt{toolIndex} is used for choosing the outlook of the tool from a predefined set, \texttt{home} sets up its start position, and
the two `eventOuts` are used for interaction with the rest of the scene. Additionally, in the script serving this prototype different query-functions are implemented. They allow other objects interacting with the tool object to extract tool (and tool-instance) related information during run-time.

Until there are coordinates left in the `Coordinate` node of `NCPath`, the mill-script reads a point and with the help of a `PositionInterpolator` makes the mill object move towards the point read. During this movement, `pos_changed` events are generated and sent to the processed workpieces.

The next step was to decide how to model the workpiece for allowing dynamic changes in its geometry. Let us start with the prototype and its parameterization given in Table 2.

The geometry is defined with only two parameters — width and depth (or sizes along the X- and Z-axes). There is a parameter allowing to position the workpiece in the space (`offset`) and another one for pre-selecting the needed accuracy. After each `eventIn toolPos` a collision-check is made. The parameters of the tool geometry necessary for this check are given in Table 2 as numbered comments, but are actually read (via Java1 API) from the `Tools` parameter, which is an array of `Mill` nodes. The script

---

1 Java (TM) is a copyright of Sun Microsystems.
of this PROTO has to change the node geometry according to the results of the collision-check and in case of cutting to signal it out for allowing additional processing.

Now let us consider how the geometry has to be changed. As already mentioned, the well-known algorithms for Boolean operations, used in the CAD-systems, seem to be too slow for achieving the aimed performance. Moreover, these algorithms are based on primitive objects and data structures which are very different from those, available in VRML. Therefore, well-suited VRML primitives had to be chosen and respectively extended with scripts (Java, ECMA or others). Nodes in question were all those with geometry (related) elements in exposedFields or eventIns like Extrusion (set_crossSection, set_spine, set_orientation and set_scale), Elevation Grid (set_height, color, normal), as well as the group nodes with their addChildren and removeChildren events. After performing some tests with different combinations of these nodes (including a parameterized lattice of parameterized objects), the ElevationGrid was chosen for the following reasons:

1. It is a native VRML-node, which simplifies the implementation and the portability of the resulting models.
2. It gave the best ratio for visualization performance and accuracy.
3. It is well suited for representing objects with complex geometry and even for free-form surfaces (cf. Fig. 5).
4. It is well suited for some optimization of the necessary calculation.
5. Together with the PROTO and LOD mechanisms and with Java programs it offers good parameterization and scalability.

2.3. Intermediate results

Figs. 4 and 5 illustrate different phases of the virtual shaping of a (half) mould for a simple crankshaft: static visualization of cutter-trajectories for roughing and finishing (Fig. 4) and the results of the processing itself (Fig. 5).

2.4. Problems

The main problems encountered in this project were the immense volumes of information and their representation for allowing efficient processing and visualization. Although both of the used browsers (Cosmo-Player\(^2\) 2.0 and WorldView\(^3\) 2.1) have implemented most of the nodes with boundaries much higher than the compliance with the standard requires, some models were so large that they caused overflow of internal variables.

Attempting to prepare the sample model, given in Fig. 6, as (one) IndexedLineSet caused its field MFInt32 coordIndex to overflow. Changing the model to use a PointSet instead still worked and reduced the size of the model from 18 to 11 MB. The virtual shaping of the same model, i.e. applying this trajectory together with a given cutter on a PrismWorkPiece (with accuracy about 1 mm), caused no problems except the time for the virtual processing. Saving of the processed PrismWorkPiece without the trajectory information produced a model with under 300 kB uncompressed size (cf. Fig. 7).

---

\(^2\) Cosmo (TM) Player is a copyright of PLATINUM technology IP.
\(^3\) WorldView (TM) is a copyright of PLATINUM technology IP.
2.5. Current and future work

Currently under development are:
- Form-based generation of VRML on the fly for allowing additional parameterization and more comfort through the HTML-based interface.
- Heuristic algorithm allowing reuse of already calculated normals.
- Enhanced possibilities for interaction with the virtual shaping via external authoring interface (EAI).
- Implementation of more levels of details for facilitating the browser and allowing efficient visualization.
- Heuristic method for dynamical changing of the accuracy.

As a main challenge for the future it is considered to exploit the portability and network-suitability of VRML and its script-languages ECMA-Script [7] and Java for offering platform-independent and distributed (or place-independent) simulation and verification of NC-programs over local or wide-area networks. Desired application schemes are with client-side processing, server-side processing (the processor of the client machine is used only for visualization) and mixed processing of the information.

3. Conclusion

Although during the last three months we succeeded to reduce the calculation time (on the same hardware) about 1800 times, the means of optimization and improvements are far not exhausted.

The developed methods, techniques, algorithms and libraries can be used not only in (distant) education, but also as a means for proof-test (cf. [6]), for research, consulting services (e.g., in advisory boards or information centers) and as means of advertisement. The approach is extremely flexible and hardware-independent. Unique features are the possibility to use geometry from different sources and formats and to compose parameterized scenarios. Playing these scenarios with different parameters allows finding answer of questions like “what would happen if the workpiece is rotated/shifted”, “how would a change in the tool-radius influence the surface quality” and others.

Acknowledgements

I am very indebted to Professor Helmut Bley for providing me with excellent research conditions and for giving me the possibility to carry out the presented work, as well as for his helpful comments and suggestions.

I am very grateful to Pavel Avgoustinov for his contribution in implementing some of the ideas, for writ-
ing and correcting code, performing intensive tests and for proof reading this paper.

References


Nikolay Avgoustinov (born 27 March 1960) did his schooling in Sofia, Bulgaria. He was in military service during 1978–1980 when he obtained a diploma in maintenance of communications and measuring devices. He studied at the Technical University, Sofia and became a certified engineer in 1985. His postgraduation was from the Department of Production Engineering/CAM, University of Saarland, in 1996 and defended his doctoral thesis in 1997. He served as CAX-support engineer (interfaces between computer and NC-machines) at CAD-division of the High Voltage Devices Research Institute, Sofia, in 1985–1986. He was a (research) fellow of the Institute of Programming and Use of Computer Systems, Technical University, Sofia between 1986 and 1991. He had received EC-grant (TEMPUS programme) for the individual project “Computer Aided Technologies in Educating CAD/CAM Know-How Engineers in the Field of Mechanical Engineering”, accomplished at the Institute of Production Engineering/CAM, University of Saarland during 1991–1992. Since 1992 he is (research) fellow of the Institute of Production Engineering/CAM at the University of Saarland.