MULTI AXIS NC CODE SIMULATION BASED ON THREE-DEXEL MODEL REPRESENTATION AND GPU

Received: 27 September 2015 / Accepted: 20 November 2015

Abstract: This paper presents a program system for multi axis NC code simulation based on three-dexel approach. Computing of tool subtraction from a workpiece is done in the GPU (Graphics Processing Unit) as computation power of modern GPU is 10 to 100 times higher than the CPU (Central Processing Unit). Initially, blank workpiece STL model is converted to three-dexel model by depth peeling algorithm. Then, structures for dexels in all three dimensions are transferred into the GPU. Tool movement is approximated by instances between two consecutive positions. For every tool instance, depth and normal textures are generated in all three directions and subtracted from the workpiece model.

Key words: three-dexel model, NC code simulation, GPU computation

1. INTRODUCTION

In the NC (Numerically Controlled) simulation process, workpiece is usually approximated with voxel or dexel elements. The voxel model represents an object with many small cubes, whilst dexel model represents an object with a grid of long columns compacted together. Dexel representation is widely used in NC simulation process, because it is less memory demanding than voxel representation [1].

In Fig. 1a, one circle is represented with dexels in one dexel direction (single dexel representation). The weak point of single dexel representation is that surfaces which are almost parallel with dexel direction can be missed in dexel generation. Much better surface representation for the same number of rays is shown in Fig 1b, where circle is intersected with rays in two orthogonal directions. For three dimensional case presented in Benouamer et al. [1], three dexel model is introduced, where model is approximated with dexels in three orthogonal directions.

Because of simplicity and fast Boolean operations, dexel representation is widely used in program systems where a real-time performance is important, such as, NC machining simulations [2,3,4,5] or virtual sculpting systems [6,7,8,9].

2. MODEL OF THE DEVELOPED PROGRAM SYSTEM

Program system is primarily developed with a goal to simulate free form modelling by hand movement in a virtual environment [9]. User moves tool in space and program system should subtract a tool from workpiece as soon as possible. Also, tool can be of a freeform shape. This program system had two limitations. Model and tool were approximated by a single dexel representation and it was a view dependent. This system is therefore extended with a three-dexel model representation, with subtraction operations now done with the GPU and it is not view dependent. It can also be used in the NC multi axes code simulation, as it is shown in this paper.

Model of the developed program system is shown in Fig. 2. Blank workpiece and tools models are imported in the system as STL files as shown in Fig. 2. Next step presents a generation of three-dexel textures in GPU (Graphics Processing Unit). After that, from generated textures, structures of every dexel are formed by CPU (Central Processing Unit). Matrices of tool instances are generated from the NC code. Then, formed dexel structures are transferred into GPU and
complete simulation of tool subtraction from workpiece is performed by the GPU. For every tool instance, model of the tool is transformed by the matrix, and three-dexel textures of the tool are generated. Then, tool instance textures are subtracted from the workpiece and the workpiece is modified. This step is repeated for every tool instance. When computation is done, three-dexel model of machined workpiece can be transferred from the GPU to the main memory, where it can be processed afterwards.

Fig. 2. Model of the developed program system

Detailed steps of the procedure explained above are presented in the following sections.

2.1 Blank workpiece STL to three-dexel model conversion

Blank workpiece model is imported in the program system in STL format (as triangle polygon collection where each triangle has a normal). For three-dexel model generation, ray intersection with model in three orthogonal directions should be calculated. This process can be very time consuming (blank workpiece may consist of millions of triangles). To speed up computation, depth peeling algorithm [10] is applied in the program system. Model is rendered in projection as many times as maximal complexity of point in this projection is. In Fig. 3, 2D case is presented.

As shown in Fig. 3, all visible faces are rendered first (Layer 1). After that, these faces are discarded and next visible faces are rendered. Process is finished when there are no faces left to render. Shape shown in Fig. 3 has a complexity of three in the view direction. Surfaces are rendered in images (textures) and every texture pixel has a depth value of surface point in the view direction.

For the model presented in Fig. 4a, rays in x and y direction intersect model in four points (two dexels) when rays in z direction intersect model in two points (one dexel). This means that in depth peeling of the model shown in Fig. 4 number of passes in x and y direction is four, and in z direction is two. Before depth peeling algorithm is applied, bounding view volume should be determined in respect that it covers complete model in all three orthogonal dimensions.

First step presents model bounding box determinations with minp, maxp and midp points (Fig. 4b). After that, maximal projection on one of orthogonal axis is determined (Fig. 4c). This projection is an edge of the cube (with center midp) from which view bounding box is calculated. Then, model is scaled and transformed in position in such way that midp is transformed in WCS origin and bounding square for texture generation is (-0.5,0.5,-0.5,0.5). Transformed and scaled model in front view is shown in Fig. 4d. For any depth peel layer model is rendered by the GPU and two images (textures) are generated. In the first texture (depth texture) every pixel contains depth value in the range 0.0-1.0. In the second texture (normal texture) every pixel has a RGB (Red Green Blue) colour, where red, green and blue components contain x, y and z normal information.

Generated depth textures in all three orthogonal directions are shown in Fig. 5. Also, in Fig. 5a, generated normal texture for first layer in z direction is shown.

Fig. 5. Generated depth textures a) in z direction with one normal texture, b) in y and c) in x direction
2.2 Three-dexel structure definition

Based on the generated textures in the previous step, structures for dexels in all three dimensions should be generated and transferred into GPU. Parameters for dixel definition are presented in Fig. 6. Every texture has a resolution (texture resolution). For dixel presented in Fig. 6a, parameters depth and length define coordinates of first and last point of dixel in the view direction, when positionx and positiony define dixel position in a plane. Also, from normal textures every dixel point normals are calculated normal1 and normal2. Two last parameters in dixel structure, cut1 and cut2, give information if blank workpiece dixels are cut in the simulation process.

Fig. 6. Dixel structure parameters

When dixel structures for all three directions are generated, they are transferred into GPU as three buffers.

Dixels in all three dimensions generated from textures from Fig. 5 are presented in Fig. 7.

Fig. 7. Generated dixels in y(a), x(b), z(c) directions and a complete three dixel model (d)

2.3 Tool definition in program system

Tool position in the developed program system is presented with instances between two consecutive tool positions. Another approach is to generate tool swept volume and subtract it from a workpiece. Drawback of the second approach is that tool swept volume generation and its subtraction is more time consuming than subtraction of tool instances. Also, for free shape tool, swept volume is very difficult and sometimes impossible to generate. An example of generated tool instances for different interpolation steps is shown in Fig. 8.

Fig. 8. Tool instances with different interpolation steps

Interpolation of tool positions is done by quaternions and after that position and orientation of every tool instance is presented by 4x4 transformation matrix.

2.4 Subtracting a tool from a workpiece

Tool instance is transformed and rendered in all three directions as a workpiece. As a result, depth and normal textures of tool instance are generated. Subtracting is performed in GPU geometry shader stage and six possible cases are shown in Fig 9.

Fig. 9. Possible cases for subtracting tool from workpiece dixels

When tool dixel is above (1) or bellow (2), intersection does not exists. If tool is cutting workpiece from top (3), top workpiece dixel is set with bottom tool dixel. When tool is cutting workpiece from above (4), bottom worpiece dixel is set with top tool dixel. If a tool dixel divides workpiece dixel (5), then two workpiece dixels are generated. And the last case (6) is when tool dixel completely removes workpiece dixel, then workpiece dixel is deleted from dixel buffer.

3. A SIMULATION EXAMPLE

Simulation example of the developed program system is NC code for five axis milling, which consists of 20976 lines of code. Tools are flat end mill with 10mm diameter and ball end mill with diameter of 5mm. Blank workpiece is box with dimensions 140x80x70mm and it is defined with approximately 7x10^5 dixels.

Simulation is performed on a computer with the following configuration:
- CPU Intel i5 760, 2.8 GHz
- GPU NVidia GTX 660 (960 shader units)
- Memory 4 GB
In Fig. 10 screenshots from this simulation are shown. Simulation is done with 160 fps (frames per second), which means that operation of tool instance subtraction from workpiece takes 6.25ms.

Fig. 10. Program system screenshots of simulated NC code operations

4. CONCLUSION

In this paper, a program system for multi axis NC code simulation based on three-dexel approach is presented. Tool can be a free form shape, which enables program system to be used for free form modelling with very few modifications. Computing of tool subtraction from a workpiece is done by the GPU, which causes very short computation time. Expectation of this system is that with the latest GPU generation subtraction time may be close to 1ms. This will be convenient for free form sculpting and force feedback generation in real time.

5. REFERENCES


In this paper some results of the project: CONTEMPORARY APPROACHES TO THE DEVELOPMENT OF SPECIAL SOLUTIONS RELATED TO BEARING SUPPORTS IN MECHANICAL ENGINEERING AND MEDICAL PROSTHETICS – TR 35025, carried out by the Faculty of Technical Sciences, University of Novi Sad, Serbia, are presented. The project is supported by the Ministry of science and technological development of the Republic of Serbia.

Authors: Assoc. Prof. dr Zoran Milojević, Assoc. Prof. dr Slobodan Tabaković, Assist. MSc. Mirjana Bojanić, Full Prof. dr Milan Zeljković, University of Novi Sad, Faculty of Technical Sciences, Institute for Production Engineering, Trg Dositeja Obradovica 6, 21000 Novi Sad, Serbia, Phone.: +381 21 450-366, Fax: +381 21 454-495. 
E-mail: zormil@uns.ac.rs; tabak@uns.ac.rs; bojanicm@uns.ac.rs; milanz@uns.ac.rs