An implementation of CSG modeling technique on Machining Simulation using C++ and Open GL

Duy Le[a], Su-Jin Kim[a], Jong-Min Lee[a], Anh-Thi Nguyen[b],Vy-Thoai Ha[b]

[a]School of Mechanical and Aerospace Engineering, Gyeongsang National University

900 Gajwa, Jinju, Gyeongnam, 660-701, South Korea

duyle2003@gmail.com, sujinkim@gnu.ac.kr, ljm30422@yahoo.co.kr

[b]Department of Aeronautical Engineering, University of Technologies

268 Ly Thuong Kiet st., dst. 10, Ho Chi Minh City

thinguyen@hcmut.edu.vn, havythoai@yahoo.com

Abstract

An application of CSG (Constructive Solid Geometry) modeling technique in Machining Simulation is introduced in this paper. The current CSG model is based on z-buffer CSG Rendering Algorithm. In order to build a CSG model, frame buffers of VGA (Video Graphic Accelerator) should be used in term of color buffer, depth buffer, and stencil buffer. In addition to using CSG model in machine simulation Stock and Cutter Swept Surface (CSS) should be solid. Method to create a solid Cuboid stock and Ball-end mill CSS are included in the present paper. Boolean operations are used to produce the after-cut part, especially the Difference operation between Stock and CSS as the cutter remove materials form stock. Finally, a small program called MaSim which simulates one simple cut using this method was created.

Keywords: CSG model, Frame buffer, Z-buffer, Machining Simulation, Boolean Operations

1. Introduction

Computer Numerically Control (CNC) machining is a widely used manufacturing process. The simulation of CNC machining is an important component of CAM, it can check errors and enhance the automation of machining process. Almost all the commercial CAD/CAM software, comprises machining simulation functions. Machining simulation falls in two aspects, physical process simulation and geometric process simulation. The former discusses metal cutting, MRR(metal removal rate) and cutting force, etc. The latter concerns the geometry modeling of machining process, for example, curve modeling for cutter edges, surface modeling for the rake face, and solid modeling for removal material. Geometry modeling is used widely in 3D machining simulation, and there are a number of algorithms in the previous papers. GWB(Geometry workbench)[1], Z-Map[2], Modified octree algorithm[3], SDE(Sweep Differential Equation)/SEDE (Sweep Envelope Differential Equation)[4], B-rep algorithm[5], Volumetric Model[6]. These algorithms are almost used for milling machining simulation, especially for cutter path simulation. The basic method of these algorithms is boolean operator of CSG. But CSG is a common method for geometry modeling, so we can build geometry modeling using CSG method to simulate the cutting process, and show the machined surface of mold.

Using the CSG modeling technique combined with the z-buffer CSG rendering algorithm[10], we present our method to simulate the cutting process in C++ and OpenGL.

2. Constructive Solid Geometry

Constructive Solid Geometry (CSG) is a technique for geometric modeling. CSG allows a modeler to create a complex object by using Boolean operations to combine object form simple shape as cuboids, cylinders, spheres, cones, prisms, pyramids. The Boolean operations in CSG are Union (∪), Intersection (∩), and Difference (-). Figure 1 clarifies a basic CSG Object tree.

CSG model has great advantages for some fields. One such field, which is the main motivation for our work, is Machining Simulation. Milling process is mentioned when material is removed from a cuboid

stock. A machine tool in this kind of process is present in Fig. 2.

For each milling operation, the volume swept by the cutter can be represented into triangles. This boundary surface is checked with volume of the stock in order to display the after-cut object.

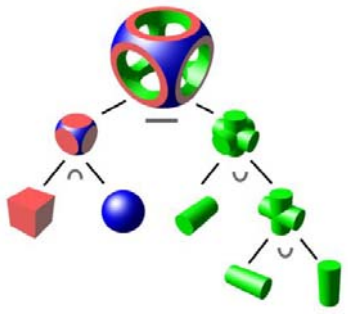

Fig. 1. A CSG tree

3. Frame buffer and difference operation

3.1 Frame buffer

There are four buffers in frame buffer but for the purpose of this paper we just present three buffers. Color buffer store color of each pixel of a frame. This buffer is what you see. Depth buffer store depth information for each pixel of a frame. It's also call Z-buffer. The last, stencil buffer used to create mask for restricting drawing area. This buffer store value associated to each pixel. Typically, we begin to draw a shape and tell to set 1 where pixels (of this shape) are drawn. Then, we make read only this buffer and tell we only want to draw where value is 1. Following objects will only be drawn in the corresponding region of '1'. The use of the stencil buffer is the main solved problem in the difference operation in the paper.

3.2 Difference operation

Draw a back face of CSS into depth buffer when enabling the depth test and disabling the color buffer update (step1). Use stencil plane to find part of CSS in Stock and draw it into depth buffer (step2). Use stencil plane to find part of Stock in CSS and draw it into depth buffer (step3). Disable stencil test for reset and enable depth test for 3D final view (step4). Finally view the toolpath. Table 1. shows the code of this operation.

4. Machining simulation

4.1 Stock and cutter

Basic cuboid stock is created through Open GL environment by using *glBegin (GL_QUADS)*.

A solid cutter model is created by three parts. The first - the bottom of the cutter - which is spherical shape, is drawn by dividing a sphere (30 slices, 20 stacks) to take a half (using *glClipPlane*).

```
GLdouble eqn[4] = {0.0, 0.0, -1.0, 0.0};
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
glClipPlane (GL_CLIP_PLANE0, eqn);
glEnable (GL_CLIP_PLANE0);
gltDrawSphere(5.0f, 30, 20);
glDisable (GL_CLIP_PLANE0);
```

The others are body and top of the cutter. Both are cylindrical shape which can be built by two single cylinder with the same 30 slices and 1 stack. After that they need to be rearranged in the same coordinate to form a complete ball-end mill.

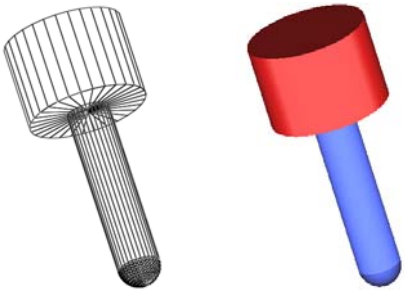

Fig. 2. Ball-end mill

4.2 Cutting swept surface

CSS is a surface that is created when cutter move over and cut the stock. An approximate surface with CSS is created by set of triangles. During cutting process the cutter bottom surface always contact with the stock. This contact area creates grazing curve (or Silhouette curve in [5]). For the simplest case grazing curve is chosen a half-circle.

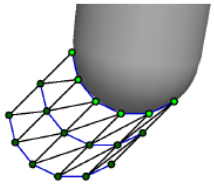

Fig. 3. CSS created by nodes on grazing curve

Next is the method to create a CSS of a Ball-end mill using global coordinates (stock's coordinates).

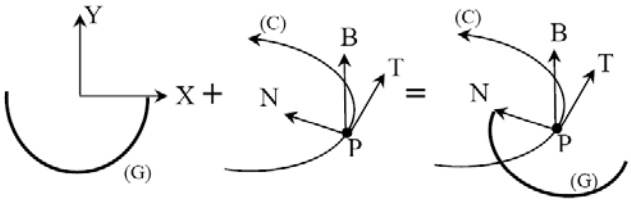

Fig. 4. Position and direction of grazing curve

(C) is toolpath. T, N is tangent and normal vector of (C). B is T cross N. So T, N, B is local coordinates of every single point on (C). Then (C) is divided into

points and grazing curves are created in each point. Finally all curves are connected together form the CSS. But in order to using the CSG model it should be solid. So it's made solid by changing grazing curve to a full circle.

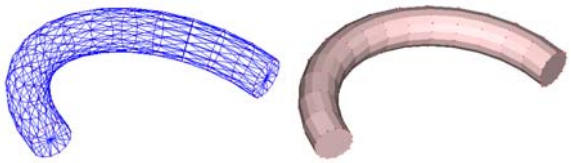

Fig. 5. Solid Swept Surface

4.3 Simulation result

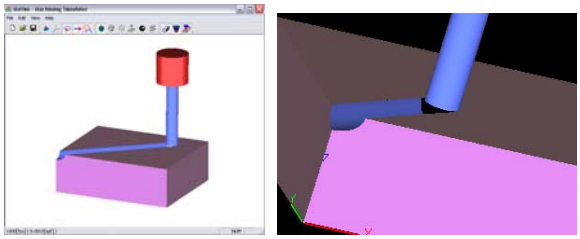

Fig. 6. Masim program

Table 1. Code applied for difference operation

```
glEnable(GL_DEPTH_TEST);
glColorMask(GL_FALSE, GL_FALSE, GL_FALSE,
GL_FALSE);                                          } Step1
glCullFace(GL_FRONT);
RenderSweptSurface();

glDepthMask(GL_FALSE);
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_ALWAYS, 0, 0);
glStencilOp(GL_KEEP, GL_KEEP, GL_INCR);
glCullFace(GL_BACK);
RenderStock();
glStencilOp(GL_KEEP, GL_KEEP, GL_DECR);
glCullFace(GL_FRONT);
RenderStock();

glDepthMask(GL_TRUE);
glColorMask(GL_TRUE, GL_TRUE, GL_TRUE,
GL_TRUE);
glStencilFunc(GL_NOTEQUAL, 0, 1);                   } Step2
glDisable(GL_DEPTH_TEST);
glCullFace(GL_FRONT);
RenderSweptSurface();

glColorMask(GL_FALSE, GL_FALSE, GL_FALSE,
GL_FALSE);
glEnable(GL_DEPTH_TEST);
glDisable(GL_STENCIL_TEST);
glDepthFunc(GL_ALWAYS);
RenderStock();
glDepthFunc(GL_LESS);

glCullFace(GL_BACK);
RenderStock();
```

```
glDepthMask(GL_FALSE);
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_ALWAYS, 0, 0);
glStencilOp(GL_KEEP, GL_KEEP, GL_INCR);
glCullFace(GL_BACK);
RenderSweptSurface();

glStencilOp(GL_KEEP, GL_KEEP, GL_DECR);
glCullFace(GL_FRONT);
RenderSweptSurface();                               } Step3

glDepthMask(GL_TRUE);
glColorMask(GL_TRUE, GL_TRUE, GL_TRUE,
GL_TRUE);

glStencilFunc(GL_EQUAL, 0, 1);
glDisable(GL_DEPTH_TEST);
glCullFace(GL_BACK);
RenderStock();

glDisable(GL_STENCIL_TEST);
glEnable(GL_DEPTH_TEST);                            } Step4
```

5. Conclusion

Through the work of present paper we implement the CSG modeling technique via OpenGL and get the acceptable result. Compare to another method such as z-map it runs with a higher speed but costs less memory space. It's worth to predict and test manufacturing process by using CSG model.

References

[1] Hazim A. EI-Mounayri. Generic Solid Modeling Based Machining Process Simulation [Dissertation].Hamilton: McMaster University, 1997.

[2] Seung Ryol MAENG Nakhoon BAEK. A Fast NC Simulation Method for Circularly Moving Tools in the Z-Map Environment. Proceedings of the Geometric Modeling and Processing 2004 (GMP'04).

[3] Trung Thanh Pham,Yong Hyun Kim, Sung Lim Ko. Development of a Software for Effective Cutting Simulation using Advanced Octree Algorithm. Fifth International Conference on Computational Science and Applications.2007, pp.324-332

[4] Liping Wang. Modeling of 3D Swept Volumes Using SDE/SEDE Methods And its Application To Five-axis NC Machining [Dissertation]. Newark: New Jersey Institute of technology,1997.

[5] R.V. Fleisig, A.D. Spence. Techniques for accelerating B-rep based parallel machining simulation. Computer-Aided Design 37 (2005) 1229–1240.

[6] J. Zhang, S.K. Ong, A.Y.C. Nee. A Volumetric Model-Based CNC Simulation and Monitoring System in Augmented Environments. Proceedings of the 2006 International Conference on Cyberworlds (CW'06).

[7] Richard S. Wright, Benjamin LipChak. Nicolas Heamel, OpenGL Super bible or "The Blue Book", Third and Fourth Edition, SAMS and Addition – Wesley Publisher.
[8] Jackie Neider, Tom Davis, Mason Woo, OpenGL Programming Guide or "The Red Book", Addition – Wesley Publisher.
[9] Tom McReynolds, David Blythe, Advanced Graphics Programming Using OpenGL, Morgan Kaufmann Publisher.
[10] J. Goldfeather, J. Hultquist, H. Fuchs, "Fast Constructive Solid Geometry in the Pixel-Powers Graphics System", Computer Graphics (Proc. Siggraph), Vol. 20, No. 4, Aug. 1986, pp.107-116.
[11] Chung, Park, Choi, 1998, Modeling the surface swept by a generalized cutter for NC verification.