IT FOR SOCIETY, Vol. 03, No. 01
ISSN 2503-2224

Rubik's Snake Simulator For PC

Rikip Ginanjar*, Natasha Janice
Information System Study Program, President University
*Corresponding author: rikipginanjar@president.ac.id

Abstract - Rubik's Snake is a toy that can be played by people with almost all ages. It is usually formed of 24 edges of right isosceles triangular prisms that are connected with spring bolts so that each edges can be rotated in some way to form a shape. Rubik's Snake is a tool to test an idea of shape in space. This research will develop a 3D Rubik's Snake simulator for PC with "Snix" as the name of the application. The simulator is expected to imitate a real Rubik's Snake behavior that its edge can be rotated in some way to form shapes by implementing existing 3D computer graphics algorithms.

I. INTRODUCTION

One of many functionalities of computer is computer simulation. Computer simulation is a way to model real life system into graphical models by implementing mathematic equations and algorithms. Computer simulation can also be done for toys included Rubik's Snake [4].

Playing actual Rubik's Snake might be easier than playing the simulator. However, there are some advantages that can be achieved by simulator. One advantage of using the simulator instead of the actual Rubik's Snake is that shapes that have been made can be saved to different files that can be loaded without breaking other shapes.

The research was intended to solve how to choose and implement existing 3D computer graphics algorithms for there are many 3D computer graphic algorithms that have been developed. Several algorithms solved the same problem. One solution might be better than the others in terms of performance and graphics result, one solution might be better in performance but poor in terms of graphic visual result, and one solution might be better in graphic visual result but has poor performance. One has to choose which algorithm to be used based on needs.

The objective of this research is to create user interactive 3D Rubik's Snake simulator without any 3D engine. A simulator that lets user perform changes on the rubik as similar as changes done in actual Rubik's Snake by implementing algorithms such as Hierarchical Modeling, Scanline, Back Face Culling, and Phong Illumination Model with Flat Shading.

This research has several limitations. First limitation is the edges of rubik must be right isosceles triangular prisms, the number of edges is limited to twenty-four, and there is no collision detection between the edges.

II. METHODOLOGY

1.1 Hierarchical Modeling

Hierarchical modeling is a way to put a 3D objects in a hierarchical structure where one object might be a parent to another object. There will be local coordinate system or object coordinate system for each object. Transformation of an object in its local coordinate system will affect its children objects. Because Transformation done for a 3D primitive in is the result of matrix multiplication result of the parents (from the root until first parent) transformation matrices to find the position of each primitive in the world coordinate system and its own transformation matrix [6].

1.2 Back Face Culling

Back Face Culling is a way to eliminate surfaces of 3D object that does not visible to the viewer for example back surface or surface that perpendicular to the viewer. Determining whether or not the surface is visible can be done by finding dot product of viewing direction vector and surface normal. If the result is equal or greater than 0 surface is not visible and will be eliminated from rendering process [5][6].

1.3 Scanline Algorithm

Scanline algorithm is a visible surface detection algorithm, that will show front most visible surface and hide other surface or part or surface that is behind or hidden by other surface. Scanline algorithm will generate Sorted Edge Table from surfaces edges and process them. Scanline will determine the front most object at each line [6][11].

1.4 Phong Illumination Model

Phong Illumination model is a way to model to calculate color intensity on a point which components are ambient, diffuse, and specular reflection [8].

Ambient is not a light that comes from the light source and reaches the eye. For light hitting on a surface will either be transmitted, absorbed, and reflected, the object that is not directly exposed to the light source could still be visible by those reflected light [8].

Diffuse light is light that comes directly from the light source and hitting a surface, and the light hitting the surface will be reflected to directions based on surface texture and direction [8].

Specular reflection models direct light reflection from light source to the eye. Specular reflection gives the shiny effect to the surface, such as highlight that can be seen on polished metal that depends on view direction [8].

The equation of Phong Illumination Model is $I_{total} = ambient + diffusal + specular$, or exactly as follows:

$$I_{total} = k_a I_a + \sum_{i=1}^{m} (\mathrm{k_d I_L}(\mathbf{L} \cdot \mathbf{N}) + \mathrm{k_s I_L}(\mathbf{V} \cdot (2(\boldsymbol{L} \cdot \boldsymbol{N})\boldsymbol{N} - \boldsymbol{L}))^{\mathrm{n_s}})$$

where ka, kd, ks are coefficient for ambient, diffuse, and specular respectively, m is number of light, L is unit vector of light from a point on a surface, N is unit vector of normal vector of a surface, and V is unit vector of view vector [6].

1.5 Flat Shading

Flat Shading is shading technique that calculate color intensity on a single point on a surface and color the surface with the color. This technique is fast in computation; however, it lessens the smoothness of the object and not good for object with many curves [5].

III. Experimental Results

Table 1 and Table 2 are important testing scenarios that have to fulfilled and work as expected on the program

Table 1. Playing Rubik Testing Scenario

No	Scenario	Expected Result
1	User runs the application	Application is opened, all controls and the rubik is displayed
2	User click numeric up button for changing edges height	If height is less than 4.0 then height will increase by 0.1, rubik will be resized, and the changes will be drawn to screen
3	User clicks numeric down button for changing edges height	If height is greater than 0.1 then height will decrease by 0.1, rubik will be resized, and the changes will be drawn to screen.
4	User scrolls changing degree between height and hypotenuse line track bar to the right	Rubik will be resized with the degree increased, changes of rubik will be displayed to screen until it reaches the maximum value.
5	User scrolls changing degree between height and hypotenuse line track bar to the left	Rubik will be resized with the degree decreased, changes of rubik will be displayed to screen until it reaches the minimum value.
6	User clicks on a drawn rubik's edge bitmap screen	The edge will be selected
7	User clicks bitmap screen not on a drawn rubik's edge	If an edge is selected, the edge will be deselected
8	User clicks and drag mouse up on bitmap screen	If an edge is selected, the edge will be rotated by a negative degree else the view will be rotated on selected axis by a negative degree, result will be displayed to screen
9	User clicks and drag mouse down on bitmap screen	If an edge is selected, the edge will be rotated by a positive degree else the view will be rotated on selected axis by a positive degree, result will be displayed to screen
10	User clicks and drag horizontally to the left on bitmap screen	If an edge is selected, the edge will be rotated by a negative degree else the view will be rotated on selected axis by a negative degree, result will be displayed to screen
11	User clicks and drag mouse horizontally to the right on bitmap screen	If an edge is selected, the edge will be rotated by a positive degree else the view will be rotated on selected axis by a positive degree, result will be displayed to screen
12	User clicks on change color button	Color dialog form will be displayed.
13	User selects a color and clicks 'ok' on color dialog form	Color dialog form will be closed, selected edge color and change color button back color will be changed to selected color.
14	User clicks 'cancel' button on color dialog form	Color dialog will be closed, no action will be performed
15	User clicks button 'reset view'	View direction will be reset and the change will be displayed on screen
16	User clicks radio button 'forward'	Selected movement direction is forward
17	User clicks radio button 'backward'	Selected movement direction is backward
18	User clicks radio button 'upward'	Selected movement direction is upward
19	User clicks radio button 'downward'	Selected movement direction is downward
20	User clicks radio button 'leftward	Selected movement direction is leftward
21	User clicks radio button 'rightward	Selected movement direction is rightward
22	User clicks button 'move'	Rubik will move on expected direction
23	User performs another action other that	The action will be processed and displayed and rubik is still moving

	clicking button 'stop' and 'reset' while rubik is still moving	to its selected direction
24	User clicks button 'stop' while rubik is still moving	Rubik movement will be stopped.
25	User clicks button 'reset'	A confirmation dialog will be displayed
26	User clicks button 'Yes' on reset confirmation dialog	Confirmation dialog will be closed and application will be reset
27	User clicks button 'No' on reset confirmation dialog	Confirmation dialog will be closed, no action will be performed

Table 2. Animation Testing Scenario

No	Scenario	Expected Result
1	User selects "1. Rectangle" on animation shape combo box	Animation to form a rectangle will rum
2	User selects "2. Ball" on animation shape combo box	Animation to form a ball will rum
3	User selects "3. Bow" on animation shape combo box	Animation to form a bow will rum
4	User perform another actions while animation is still running	The action will be processed and animation will still running
5	All rubik's edges rotation degrees is equal to selected animation rotation degrees	The animation will stop

IV. DISCUSSION

Rotating or translating an edge of real Rubik's Snake will affect edges on its right or its left. That's why hierarchical model should be implemented in modeling the rubik. An edge will be a parent of an edge on its right.

Because Rubik's Snake consists of 24 triangular prisms it will be heavier for computation if all surfaces are included, with back face culling a hidden surface will be eliminated before it is processed by Scanline.

Because a Rubik Snake's include more than one 3D objects, there might be a condition where objects behind other objects and objects might intersect each other. With scanline algorithm it can be determined which part of the object should be drawn.

Using Phong Illumination Model with Flat Shading for the simulator is advantageous because the simulator is user interactive fast computing algorithm is needed. The shading itself is needed so that an edge can be differentiated from each other if they have same color.

V. CONCLUSION

There are several conclusions that can be taken from this development of this research. The first one is that 3D Rubik's Snake simulator can be done without using any 3D engine. The second conclusion is that 3D rubik's snake simulator that allows user to perform changes on the simulator as similar as changes that can be done on actual Rubik's Snake can be done by implementing 3D computer graphics algorithms which are, Scanline algorithm for visible surface detection,
Back Face Culling to eliminate back or side surface of 3D object, Hierarchical modeling for modeling the rubik, Phong Illumination Model can be implemented with Flat Shading method for lighting effect.

References

[1] P. Shirley, M. Ashikhmin, M. Gleicher, S. R. Marschner, E. Reinhard, K. Sung, W. B. Thompson and P. Willemsen, Fundamentals of Computer Graphics, Massachusetts: A K Peters, 2005.

[2] H. Ruiter, "Warp3D Nova: 3D Lighting - Part 1," Kea Sigma Delta, 18 November 2016. [Online]. Available: https://keasigmadelta.com/blog/warp3d-nova-3d-lighting/. [Accessed 9 October 2017].

[3] E. Lengyel, Mathematics for 3D Game Programming and Computer Graphics, Third Edition, Boston: Course Technology, 2012.

[4] P. Jain and M. Sampaolo, "Computer Simulation," Encyclopædia Britannica, 27 April 2017. [Online]. Available: https://www.britannica.com/technology/computer-simulation. [Accessed 9 October 2017].

[5] J. F. Hughes, A. Van Dam, M. McGuire, D. F. Sklar, J. D. Foley, S. K. Fiener and K. Akeley, Computer Graphics: Principles and Practice, Third Edition, Addison-Wesley, 2014.

[6] D. Hearn and M. P. Baker, Computer Graphics, C Version, Second Edition, Prentice Hall, 1996.

[7] Y. Dzhurov, I. Krasteva and S. Ilieva, "Personal Extreme Programming–An Agile Process for Autonomous Developers," Proceedings of International Conference on Software, Services & Semantic Technologies, pp. 252-259, 2009.

[8] F. Dunn and I. Parberry, 3D Math Primer for Graphics and Game Development, Plano: Wordware Publishing, Inc, 2002.

[9] R. Clifford, Pinterest, [Online]. Available: https://id.pinterest.com/pin/535928424383602831/. [Accessed 9 October 2017].

[10] "Triangle Mesh," Wikipedia, 19 February 2017. [Online]. Available: https://en.wikipedia.org/wiki/Triangle_mesh. [Accessed 9 October 2017].

[11] "Scanline Rendering," Wikipedia, 28 June 2017. [Online]. Available: https://en.wikipedia.org/wiki/Scanline_rendering. [Accessed 9 October 2017].

[12] "Rubik's Snake," Wikipedia, 28 June 2017. [Online]. Available: https://en.wikipedia.org/wiki/Rubik%27s_Snake. [Accessed 9 October 2017].

[13] "Normal Vector Calculation Function," [Online]. Available: https://densorobotics.com/content/user_manuals/19/005531.html. [Accessed 9 October 2017].

[14] "Normal (Geometry)," Wikipedia, 12 January 2017. [Online]. Available: https://en.wikipedia.org/wiki/Normal_(geometry). [Accessed 9 October 2017].
[15] "Cartesian Coordinate System," Wikipedia, 1 October 2017. [Online]. Available: https://en.wikipedia.org/wiki/Cartesian_coordinate_system. [Accessed 9 October 2017].
[16] "(Poculeka) Three-JS-Snake," Github, 17 July 2016. [Online]. Available: https://github.com/poculeka/three-js-snake. [Accessed 9 October 2017].