International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 08 | Aug 2020 www.irjet.net p-ISSN: 2395-0072

Using 3D Models from Blender for use in OpenGL Virtual Reality Applications

Aditya[1], Anitha M[2]

[1]*Student, Dept. of Computer Science, Dayananda Sagar College of Engineering, Bangalore, India*
[2]*Professor, Dept. of Computer Science, Dayananda Sagar College of Engineering, Bangalore, India*

Abstract - *OpenGL is used in various graphical software implementations that vary from games to medical imaging and virtual reality applications. The use of 3D modelling software such as Blender has increased over time for games, simulations and animations. Such software allows graphical models to be developed with ease and to be visualized immediately rather than at run time. But they lack in terms of platform compatibility and have significant overheads. In this paper we suggest a hybrid approach using Blender for modelling and OpenGL to perform the rendering, animation and transformations of the models is outlined and contrasted against a pure OpenGL workflow. Better workflow efficiency, platform compatibility and low overhead aspects are reaped from this hybrid approach when compared to pure OpenGL.*

Key Words: Graphical Modelling, 3D Animation, Blender, OpenGL, Assimp, Virtual Reality

1. INTRODUCTION

Virtual reality is an artificial three-dimensional environment created by a computer and presented to a person in an interactive way [1]. It refers to the computer simulation displaying an environment through which one can walk and interact with objects and simulated computer-generated people. With the increasing availability of cost accessible hardware components, virtual reality applications are finding its way into various applications such as gaming, learning and social skills training, military training, architectural design, simulations of surgical procedures, rehabilitation of psychological disorders such as anxiety, schizophrenia, depression, and eating disorders. [2,3,4,5,6,7].

The artificial environment is created by 3D modelling which creates a realistic world so that the users could immerse into this environment by interacting with these objects using the hardware devices such as computer mouse, joystick, force sensor, cyberglove having the sense of the real world. The efficiency of VR applications solely depends on the creation of 3d objects. In the period between 1996 and 2005 key changes emerged on the platforms such as Superscape World Builder and OpenGL that supported easier development and distribution of desktop VR applications [8]. OpenGL provides the better workflow efficiency, platform compatibility and low overhead that is needed for the VR applications. Hence, in this paper we suggest the usage of improvised technique of creating 3D objects with the combination of OpenGL and Blender software.

2. MODELLING FOR GRAPHICS APPLICATIONS

OpenGL is a cross-platform, cross-language software interface to graphics hardware for rendering 2D and 3D vector graphics [9]. The interface consists of several hundred procedures and functions that allow a programmer to specify the objects and operations involved in producing high-quality vector graphical images. Blender is a free and open source 3D modelling, simulation and animation suite [10,11]. It supports the entirety of the 3D pipeline including rigging, rendering, compositing and motion tracking, video editing and game creation. A Python API is available for use by advanced users.

Modelling for graphics applications can be performed using one of three techniques:

1. OpenGL only: OpenGL allows models to be created by specifying triangles or quads as arrays of vertices, edges & normals, with additional arrays for color and texture data. Another technique to create models in OpenGL is enabled with the use of toolkits such as freeglut which provide routines to create basic structures such as cubes, cones, cylinders and teapots. This approach is powerful but basic and requires knowledge of the coordinates, normals and colors beforehand, leaving little room for experimentation and creativity. Complex models such as a human face are challenging to work with in pure OpenGL.
2. Blender only: Blender supports game and animation development through its GUI interface. This seems to be the best option, but misses out on the widespread adoption and testing that OpenGL has received. OpenGL has a bare metal approach to graphical programming which may be useful in some cases. Complex models are easily tweaked and flexibility exists in the design of models. Models can be viewed as they are being created. Blender also provides basic structures such as cubes, spheres etc. [26]
3. Creating the models in Blender and importing the data into OpenGL applications: In this approach modelling software such as Blender is used to create the models, and a translation program or library such as Assimp is used to convert the models to corresponding OpenGL code. The data format generated by the modelling software is usually

defined by a set of polygons with position and normal data. This allows for easy modification of models if needed, and incorporates the cross - platform, cross - language abilities of OpenGL.

Modelling is used by several industries with many more such as medical simulations, and virtual surgery as described in [12,13]. Dental processes can be better understood by reconstructing a person's jaw in software[14,15]. Engineering models are now being modelled in 3D graphical environments that enable printing of prototypes and automated CAD generation. Industrial workflow simulations are better illustrated using animations rather than by a verbal description [16]. Human bio-mechanical simulations help understand the internal working of skeletal muscles better as described in [17].

3. MODELLING WITH PURE OPENGL AND FREEGLUT

OpenGL provides functions to generate primitives such as points, lines and polygons with additional functions for view manipulation, texture projection, object transformations, rendering and so on. This along with the cross platform compatibility and open source licensing makes it the de facto choice for development of graphics applications.

OpenGL being an API, does not provide advanced functions for complicated 3D modeling. So creating complicated 3D models has to be performed through the combination of the basic geometry such as points, lines and polygons (Fig 1). OpenGL provides a function (glDrawArrays)[9,18,19] to render a model using arrays of vertex coordinates, edges & edge order, normals, and vertex color.

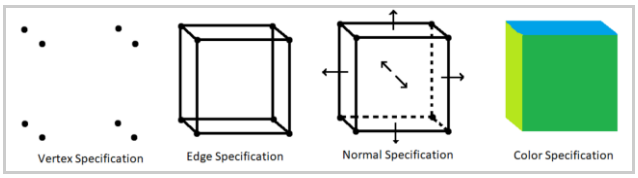

Fig -1: Modelling of a 3D object in OpenGL using arrays [20]

A) Basic Geometry Modelling: Simple models can be plotted by hand and the coordinates provided as input to the OpenGL pipeline. This works well for models that are simple and provides great control over the individual vertices and normals. While rotation, scaling or translation can be handled using functions in OpenGL, transformations that affect sections of a model (facial expressions) are not simple. As the polygon count in a model increases, simple modelling is not feasible.

B) Modelling using OpenGL and freeglut: Utility toolkits like freeglut (A replacement for the original GLUT toolkit) provide abstractions to create simple 3D structures such as boxes, cylinders, cones, spheres, discs, teapots and more[20]. While these toolkits simplify basic 3D objects that could be the building blocks of a model, they restrict the ability to modify individual faces, normals, vertices or colors of faces. Models of realistic looking objects, curved objects such as pillows are not fit for generation through this method, and hence developers rely on glDrawArrays instead.

Using the freeglut toolkit, modelling of a basic snowman structure took about 2 minutes with boilerplate code and 6 lines of modelling code as outlined below. Boilerplate code was required to visually verify the appearance of the model and the positioning of structures [21]. The boilerplate code contained the initializations for OpenGL and freeglut as well as a mouse input handler to allow for a 3D walk around of the model for verification of the model's accuracy to the design.

```
glScalef(1.0,4.0,1.0);
glColor3i(255,255,255);
glutSolidSphere(1.0);
glTranslatef(0.0,5.0,0.0);
glScalef(1.0,0.25,1.0);
glutSolidSphere(0.5);
```

4. MODELLING WITH BLENDER

Blender is an open source modelling and animation suite, which makes it a good candidate for most graphics application development endeavors. Contrary to modelling with OpenGL, it provides real time visualization of the created model and transformations applied to it with an interactive GUI. This makes designing easier and training new designers simpler.

Similar to OpenGL toolkits, Blender provides some basic structures to construct models, similar to OpenGL that can be manipulated to obtain any shape desired [25]. For example, a thin cylinder can be manipulated into a plate. The model/scene generated is saved in BLEND format by default, but can be exported in many other formats including OBJ, DAE, ABC, FBX, X3D. Additionally, Blender allows for easier generation of curved models and high polygon count models through the use of GUI based click and drag tools rather than requiring understanding of various mathematical formulae. Specifying and changing colors, textures are convenient and simple to perform.

Using Blender alone, modelling a simple white snowman structure took about a minute and was achieved using the UV sphere mesh, scale and move options. Then a material with color set to white was applied to both meshes at the same time. The resulting model could be visually verified without requiring any boiler plate code.

5. HYBRID APPROACH WITH BLENDER AND OPENGL

Translation of models created in Blender can be performed in a few ways, each with its own characteristics. A suitable approach has to be chosen based on the requirement of the application. Translator programs that work on a single file format can be simpler to implement if the additional features

of import libraries are not used in the application, while import libraries like Assimp can convert multiple file formats comprehensively to a single data structure format to be translated.

A) Writing a translator program: Blender allows output files to be in several formats. Since OBJ files have a simple file structure, it is a good candidate to use for writing a translator program. Each line in the file represents an entity. Vertices are represented by a 'v' at the start and faces are represented by a 'f' [18,22]. A triangle in OBJ file representation consists of three vertices and a face as enlisted below.

```
v 0.0 0.0 0.0
v 0.0 1.0 0.0
v 1.0 0.0 0.0
f 1 2 3
```

A simple translator program would be (pseudocode):

```
for line in file:
 lineElements = line.split(" ")
 if lineElements[0]=='v':
  queue.push(lineElements[1:])
 elif lineElements[0]=='f':
  glBegin(GL_POLYGON)
  while !queue.isEmpty():
   glVertex3f(queue.pop())
  glEnd()
```

B) Using a import/translation library such as Assimp (Open Asset Import Library):

Writing a translator program directly requires detailed knowledge about the data structure and internal format used by a particular file. This also restricts further development of the graphics application to use that file format. Additional limits are imposed by the efficiency, capability and portability of the translation program on different platforms. Libraries such as Assimp allows a programmer to load model files of various formats dynamically with the added benefit of being well tested and robust. Assimp accepts most of the file formats exported by Blender including BLEND format. It then processes the file into a data structure; a simplified representation of which is Fig 2. The structure of each component of the model is a node in Assimp's tree-like data structure and hence can be recursively traversed. Each node consists of an index reference to mMeshes. Each mesh structure contains vertex, normal, texture, face, and material information either as a reference, or as arrays. A scene is a structure generated by Assimp when a file is imported using the aiImportFile function. aiImportFile preprocesses the model/scene by triangulating faces, correcting for winding order, removing the redundant and hidden polygons and so on [23].

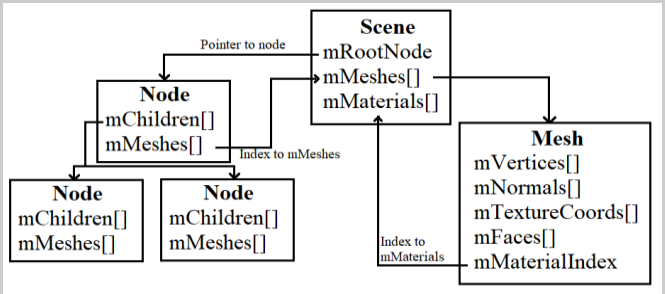

Fig -2: Data structure generated by Assimp after file processing [24]

The following pseudocode is a simple implementation of using Assimp to process Blender generated model files and render the corresponding models in OpenGL.

```
def recursive_generate(Assimp Scene sc, Assimp Node node):
 for each mesh in node:
   Load material from mMaterials and corresponding mesh
   from mMeshes[]
   Generate OpenGL object using GL_POINTS, GL_LINES,
   GL_POLYGON or GL_TRIANGLES.
   Generate substructures of current node recursively by
   calling Recursive_generate on mChildren[]
   Create an OpenGL list with the tree of objects and return
def display():
 For all the lists, apply transformations and call
 glCallList(list) to render onto the screen
```

The tree when recursively traversed and processed results in a list of OpenGL instructions which can be stored in a list for reduced processing needed later. The list can be called using glCallList whenever needed to create the desired model. This makes it easier to decouple model design and software logic development for applications such as game development and AR/VR application development.

6. COMPARISON BETWEEN THE APPROACHES

OpenGL with toolkits like freeglut allow for a wider range of models to be created by using predefined 3D objects. This is however not always useful, and in some cases may increase the number of faces to be processed to render a model. For example, a lamp would be at the least made up of a disc, a cylinder, and a frustum of a cone. Parts of each of those objects would be clipped, and processing power used to render additional regions of the lamp would be wasted. Objects like human models would be complicated to generate without any additional software, and with additional software to generate vertex and face data, any changes made to the model's design would require rework to update the stored vertex and face data to match the color & texture data. Blender and other similar modelling suites overcome this inconvenience and allow for the design of models and program logic to be decoupled. Vertex, color, texture and normal data are stored with the models, and do not require manual updating in OpenGL when a part of the model is

changed. Import libraries convert several file formats to a single data structure that can be used with a custom translation program to render the models in OpenGL. Since Blender has an interactive GUI, creation and design of complex models such as humans is easier with continuous feedback on the changes made to refine the model. This reduces time spent on waiting for the test driver program to compile for verifying that a refinement has the desired effect. Performance does have an impact, but this can be mitigated by importing the model before the program execution starts, importing on a separate thread, or embedding the imported data directly in a production build. Moreover, importing and processing of models is a onetime task, and is not repeated on each frame as with clipping or culling of faces on inefficient model designs.

Fig -3: Model translated and rendered in OpenGL

Fig -4: Model rendered in Blender

Fig. 3 is a screen capture of the OpenGL render of a model created in Blender as saved as a BLEND file. The file was converted into OpenGL code using Assimp and a translator program similar to the one described above. Fig 4 is the same model rendered using Blender's internal render engine.

The render generated through importing the model into OpenGL took an additional second at the beginning of the application to load, process and translate the BLEND file into OpenGL code. This is a single occurrence task and can be hidden behind a loading screen or performed ahead of time and cached. The render generated through Blender's internal renderer has a similar appearance to the one from OpenGL translation. The observed difference is the variance in specular reflection in Fig. 3. This can be attributed to the translation program not processing the stored specular values in the BLEND file. Another difference is that the Blender render processes ambient occlusion while the OpenGL render was not programmed to do so. No other significant differences could be found.

7. CONCLUSION

For the development of graphics applications with OpenGL, the graphics modelling tools available are limited and lack user friendly techniques. A hybrid approach to application development with OpenGL wherein a professional 3D modelling software such as Blender is better suited to modern workflows. The models can be imported into the application using an import library like Assimp. The imported data is translated into OpenGL commands using a translation program. Once translated, the OpenGL commands can be run multiple times without requiring any loading or processing of the model files. This technique allows for faster development of models, and makes it easier to change the models as required. The result concludes that models can be easily and accurately imported into OpenGL when precautions are taken to correct for any parameters that differ between the modelling software and OpenGL.

ACKNOWLEDGEMENT

This work was undertaken in the Department of Computer Science, Dayananda Sagar College of Engineering, Bengaluru, India under the guidance of Prof. Anitha M of Department of Computer Science, Dayananda Sagar College of Engineering.

REFERENCES

[1] Kamińska, D.; Sapiński, T.; Wiak, S.; Tikk, T.; Haamer, R.; Avots, E.; Helmi, A.; Ozcinar, C.; Anbarjafari, G. Virtual Reality and Its Applications in Education: Survey. Information 2019, 10, 318.

[2] Schmidt, M., Beck, D., Glaser, N., and Schmidt, C. (2017). "A prototype immersive, multi-user 3D virtual learning environment for individuals with autism to learn social and life skills: a virtuoso DBR update," in International Conference on Immersive Learning, Cham: Springer, 185–188. doi: 10.1007/978-3-319-60633-0_15

[3] Gallagher, A. G., Ritter, E. M., Champion, H., Higgins, G., Fried, M. P., Moses, G., et al. (2005). Virtual reality simulation for the operating room: proficiency-based training as a paradigm shift in surgical skills training. Ann. Surg. 241:364. doi: 10.1097/01.sla.0000151982.85062.80

[4] Song, H., Chen, F., Peng, Q., Zhang, J., and Gu, P. (2017). Improvement of user experience using virtual reality in open-architecture product design. Proc. Inst. Mech. Eng. B J. Eng. Manufacturing 232.

[5] Alexander, T., Westhoven, M., and Conradi, J. (2017). "Virtual environments for competency-oriented education and training," in Advances in Human Factors, Business Management, Training and Education, (Berlin:

Springer International Publishing), 23–29. doi: 10.1007/978-3-319-42070-7_3

[6] Neri, S. G., Cardoso, J. R., Cruz, L., Lima, R. M., de Oliveira, R. J., Iversen, M. D., et al. (2017). Do virtual reality games improve mobility skills and balance measurements in community-dwelling older adults? Systematic review and meta-analysis. Clin. Rehabil. 31, 1292–1304. doi: 10.1177/0269215517694677

[7] Englund, C., Olofsson, A. D., and Price, L. (2017). Teaching with technology in higher education: understanding conceptual change and development in practice. High. Educ. Res. Dev. 36, 73–87. doi: 10.1080/07294360.2016.1171300

[8] Keshner EA, Weiss PT, Geifman D, Raban D. Tracking the evolution of virtual reality applications to rehabilitation as a field of study. Journal of Neuroengineering and Rehabilitation. 2019 Jun;16(1):76. DOI: 10.1186/s12984-019-0552-6.

[9] Khronos Group, OpenGL 4.0 Specification (2010), https://www.khronos.org/registry/OpenGL/specs/gl/glspec40.core.pdf

[10] Blender Foundation, Blender.org, https://www.blender.org/about/

[11] Ben Crowder, "Blender" Linux Journal, Volume 1999 Issue 60es, 1999

[12] Zhou, Z.-H & Wen, X.-J. (2018). 3D Reconstruction of medical image based on opengl. Journal of Computers (Taiwan). 29. 249-260. 10.3966/199115992018042902024.

[13] Timothy Ellis, “Animating to improve learning: a model for studying multimedia effectiveness,”31st ASEE/IEEE Frontiers in Education Conference, 10 - 13, 2001 Reno, NV, 2001.

[14] Dovramadjiev, Tihomir. (2016). Advanced creating of 3D dental models in Blender software. Scientific-technical union of mechanical engineering Bulgaria. IV. 32-33.

[15] Cankova, Kremena & Dovramadjiev, Tihomir & Jecheva, Ginka. (2017). Computer parametric designing in Blender software for creating 3D paper models. ANNUAL JOURNAL OF TECHNICAL UNIVERSITY OF VARNA. Vol.1 Issue 1. 1. 77 - 84. 10.29114/ajtuv.vol1.iss1.44.

[16] Ma, Yan & Dong, Tianping & Lan, Xiaohong & Liu, Lunpeng & He, Guotian. (2012). Research of Industrial Robot Simulation based on OpenGL. International Journal of Advancements in Computing Technology. 4. 248-255. 10.4156/ijact.vol4.issue19.30.

[17] Zeng, Yanhong, Jianlong Fu and Hongyang Chao. “3D Human Body Reshaping with Anthropometric Modeling.” ICIMCS (2017).

[18] Paul Bourke, Object Files (.obj), http://paulbourke.net/dataformats/obj/

[19] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. 1999. OpenGL Programming Guide: The Official Guide to Learning Opengl, Version 1.2 (3rd ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[20] Angel, Edward; Shreiner, Dave. (2009). Interactive Computer Graphics., 6th Edition Book

[21] Angel, E., & Shreiner, D. (2011). Introduction to modern openGL programming. SIGGRAPH '11.

[22] Wikipedia contributors. (2020, August 10). Wavefront .obj file. In Wikipedia, The Free Encyclopedia. Retrieved, August 20, 2020, from https://en.wikipedia.org/w/index.php?title=Wavefront_.obj_file&oldid=972078922

[23] Assimp process.h documentation, http://assimp.sourceforge.net/lib_html/postprocess_8h.html

[24] Joey de Vries, Assimp, https://learnopengl.com/Model-Loading/Assimp

[25] Takala, Tuukka & Mäkäräinen, Meeri & Hamalainen, Perttu. (2013). Immersive 3D modeling with Blender and off-the-shelf hardware. IEEE Symposium on 3D User Interface 2013, 3DUI 2013 - Proceedings. 191-192. 10.1109/3DUI.2013.6550243.

[26] T. Dai, Z. Wang and S. Xu, "Research of Creating and Fetching 3D Models of Virtual Reality Based on OpenGL," 2006 International Conference on Mechatronics and Automation, Luoyang, Henan, 2006, pp. 1991-1995.doi: 10.1109/ICMA.2006.257560