| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

Real Estate Visitor Portal using Interactive Walkthrough on Mobile

A.P.Khan, Poonam S. Patel, Lakita K. Patil, Devyani V. Patil , Prerna L. Chaudhari

Department of Computer Engineering, D.N.Patel College of Engineering, Shahada, Maharashtra, India

ABSTRACT: Real Estate visitor portal using interactive walkthrough is a relatively large scale three dimensional scene. It shows view of the house interior through desktop system. Design two scenes one is Interior walkthrough and another is exterior walkthrough, in taking into account the difficulty of large-scale scene modeling is relatively high, each object in project environment is modeled respectively, and then each model is in lined to the interface of the main background with sky and ground, which form whole house interior scene is developed in Unity 3d 2018.3.31fl and scripting tool Visual Studio community 2017. In the process of developing a virtual house interior, the most important task is modeling, design environment by using models and function provide through script on models. After designed scene adding VR effect by using Google cardboard for Android platform.

KEYWORDS: Mobile devices, X3D, Virtual Reality, Gaming, 3D.

I. INTRODUCTION

Virtual Reality (VR) technology is an advanced human computer interface .VR (Virtual Reality) technology is a very active research field of IT industry in recent years. It is a collection of a series of high and new technology, including computer graphics, image processing, pattern recognition, intelligent interface technology, artificial intelligence, multi-sensor technology and highly parallel real-time computer technology. Conception mainly emphasizes the imaginable space of the virtual reality technology. It can widen the human knowledge range, not only reproducing the virtual environment, but freely conceiving the environment which did not exist or is impossible. Virtual Campus is a relatively large-scale three-dimensional scene. It shows view of the college through the desktop system. Taking into account the difficulty of large-scale scene modelling is relatively high, each object of the campus is modelled respectively, and then each model is in lined to the interface of the main background with sky and ground, which form the whole campus scene. In the process of developing a virtual campus, the most important task is modelling, including various types of buildings and landscape [1].

Digital design of interiors of house models is essential to avoid unnecessary construction errors. Virtual Reality (VR) software breaks traditional design environment by providing interactive imagery creation [2].

Gaming software, being a type of VR software, act as a new approach to interior design practices. This provides new ideas and perspectives in the application of VR software for interior design. There has not been much in depth research in game design elements for interior design. Using game software technology in the 3D design field has overall shorter virtual construction process (Indraspratha& Shinozaki, n.d.).

The difference between using average 3D design software and a gaming software as a 3D design software is the ability for interactive virtual environment. Users are allowed to experience a dynamic and interactive virtual interior environment by using 3D gaming software as the design software. Users can then explore the interior design by walking through the room layout. Creating a good quality virtual interior enables users' satisfaction. An interactive 3D content attract more users for house investments [3]. Using gaming development software acts as a dynamic design process and solution due to the freedom and limitless options that such software can offer to the field of interior design. Also, it provides new opportunities for comparisons with other design solutions.

The overall goal of the Walkthrough Project is to create interactive computer graphics systems that enable a viewer to experience an architectural model by simulating a walkthrough of the model. Through the years the Walkthrough project has developed many different systems; with each it has been our goal to:

a) Drive existing dynamic graphics engines to the utmost,
b) Push forward the development of methods for tracking position and orientation,
c) Have users evaluate our systems frequently so that we can identify aspects of a system which most impair the illusion of real presence, and
d) Learn more about the behaviour of people in simulations. Our long-term goal is to develop a personal, portable visualization system that will allow users to walk through and interact with models of meaningful complexity while receiving realistic visual, proprioceptive, and auditory feedback at interactive rates (>25 updates per second).

This paper describes the approach we followed for interactively visualizing large 3D buildings on mobile devices. We exploit portal culling as well as view frustum culling (polygons that are out of an approximation of the user's field of view doing not need to be processed). The culling algorithm we implemented is conservative: it does not cull polygons that are visible, but it can sometimes process polygons that are not visible. To load and render X3D files, our system uses the MobiX3D player [1]. MobiX3D was intended for playing generic X3D content on Pocket PCs, and the interactive rendering of a whole model of a large building could be impossible because of memory limitations and computational constraints of the target devices.

II. EXISTING SYSTEMS

Unity's current built-in input management system was designed before we supported the many platforms and devices that we do today. Over the years, we realized that it wasn't very easy to use and, occasionally, it even struggled with simple situations – like plugging in a controller after the executable was launched. That's why we have been working on something new; a complete rewrite. (P.S. There is currently no timeline for the removal of the current Input Manager [5].

The Input System is built from the ground up with ease of use, consistency across platforms, and flexibility in mind. Today, we'd like to invite you to try it out and give us feedback ahead of its planned release alongside Unity 2020.1. The minimum requirement for using the system is and will remain Unity 2019.1.Methodology

III. PROPOSED SYSTEM

This study uses observation and analysis of the elements in the software, Unity3D. The researches investigate the potential advantages of Unity3D software to create the virtual interior walkthrough. The usage of 3D software acts as a medium for visual representation and for visual-based analysis in the early stages of interior design process. The virtual environment should allow user to interact in exploration method by using navigation and portraying the 3D interior using various type of user interface.

There is high level framework with already available infrastructural work. This study can focus on modeling instead of building foundation for the 3D model. In other words, users can produce assets using existing tools, instead of focusing on creating an object from scratch.

A.CONCEPTUAL FRAMEWORK

Based on research objectives and background of study, a research model is derived is shown below:

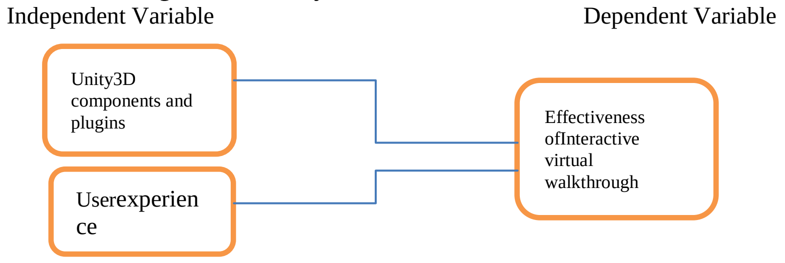

Independent Variable
Dependent Variable
Unity3D components and plugins
Userexperien ce
Effectiveness ofInteractive virtual walkthrough

Fig 1.1 Research model to create an interactive virtual walkthrough

The dependent variable is an interactive virtual model of an interior design. How the model turns out depends on Unity3D software and assets imported either from other software or websites. Other assets include the assets not found in Unity3D software. This can include floor, specific type of lighting, and so on.

According to the system flow, interior design of a room is needed at first to model the virtual environment. In this study, a meeting house showroom is chosen. The intended idea for the layouts of the showroom is sketched before the creation of the 3D model. After that, assets for the showroom, which include textures, images, 3d models, that are found as suitable for the showroom is imported. If 3d models needed cannot be found, it is to be create with the software itself [5][6].

3D modeling is executed following from the proposed bedroom interior design. Complexity and level of detail affects this process. Different scenes in various views require different proper lighting. Integration of proper lighting creates more acceptable graphics display.

B.SYSTEM ARCHITECTURE

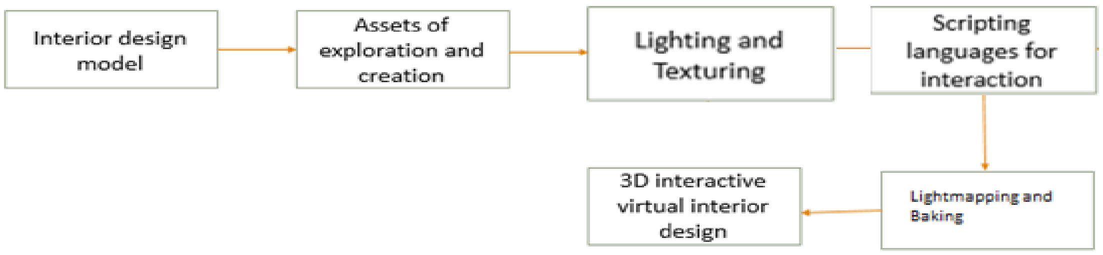

Interior design model
Assets of exploration and creation
Lighting and Texturing
Scripting languages for interaction
Lightmapping and Baking
3D interactive virtual interior design

Fig 2 System Architecture

C.SYSTEM ARCHITECTURE

Directional Light
A directional light typically simulates sunlight and a single light can illuminate the whole of a scene.This means that the shadow map will often cover a large portion of the scene at once and this makes the shadows susceptible to a problem called perspective aliasing. Perspective aliasing means that shadow map pixels seen close to the camera.

Point lights:
A point light is located at a point in space and sends light out in all directions equally. The direction of light hitting a surface is the line from the point of contact back to the center of the light object. The intensity diminishes with distance from the light, reaching zero at a specified range. Light intensity is inversely proportional to the square of the distance from the source. This is known as 'inverse square law' and is similar to how light behaves in the real world. Point lights are useful for simulating lamps and other local sources of light in a scene. You can also use them to make a spark or explosion illuminate its surroundings in a convincing way.

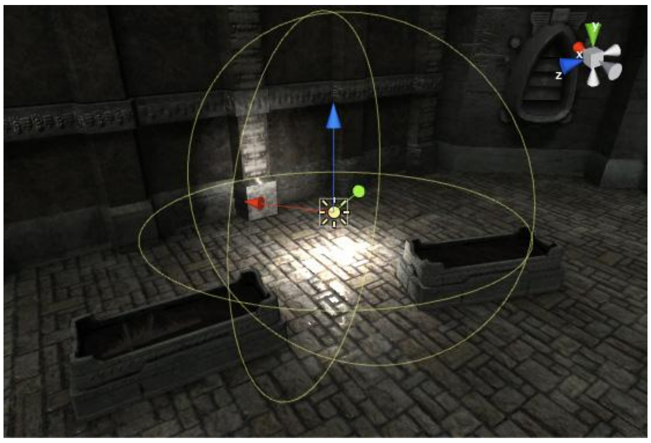


Fig 3 Effect of a Point Light in the scene

Spot lights:
Like a point light, a spot light has a specified location and range over which the light falls off. However, the spot light is constrained to an angle, resulting in a cone-shaped region of illumination. The center of the cone points in the forward (Z) direction of the light object. Light also diminishes at the edges of the spot light's cone. Widening the angle increases the width of the cone and with it increases the size of this fade, known as the 'penumbra'[7].
Spot lights are generally used for artificial light sources such as flashlights, car headlights and searchlights. With the direction controlled from a script or animation, a moving spot light will illuminate just a small area of the scene and create dramatic lighting effects.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

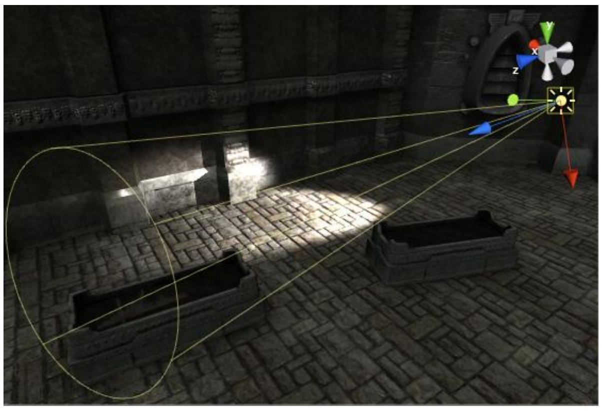


Fig 4 Effect of a Spot Light in the scene

Directional lights:

Directional lights are very useful for creating effects such as sunlight in your scenes. Behaving in many ways like the sun, directional lights can be thought of as distant light sources which exist infinitely far away. A directional light does not have any identifiable source position and so the light object can be placed anywhere in the scene. All objects in the scene are illuminated as if the light is always from the same direction. The distance of the light from the target object is not defined and so the light does not diminish.

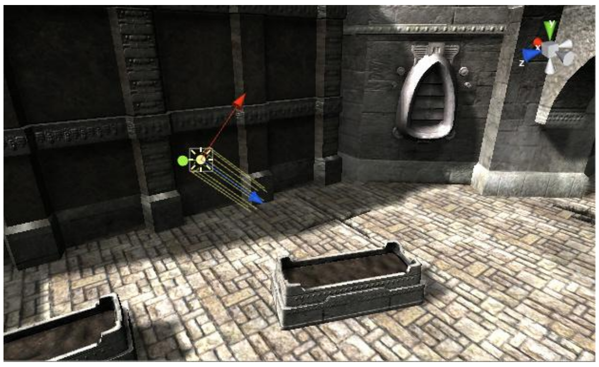


Figure 5 Effect of a Directional Light in the scene

Directional lights represent large, distant sources that come from a position outside the range of the game world. In a realistic scene, they can be used to simulate the sun or moon. In an abstract game world, they can be a useful way to add convincing shading to objects without exactly specifying where the light is coming from.

Area lights:

An Area light is defined by a rectangle in space. Light is emitted in all directions uniformly across their surface area, but only from one side of the rectangle. There is no manual control for the range of an Area Light, however intensity will diminish at inverse square of the distance as it travels away from the source. Since the lighting calculation is quite processor-intensive, area lights are not available at runtime and can only be baked into light maps. Since an area light illuminates an object from several different directions at once, the shading tends to be more soft and subtle than the other light types. You might use it to create a realistic street light or a bank of lights close to the player. A small area light can simulate smaller sources of light (such as interior house lighting) but with a more realistic effect than a point light.

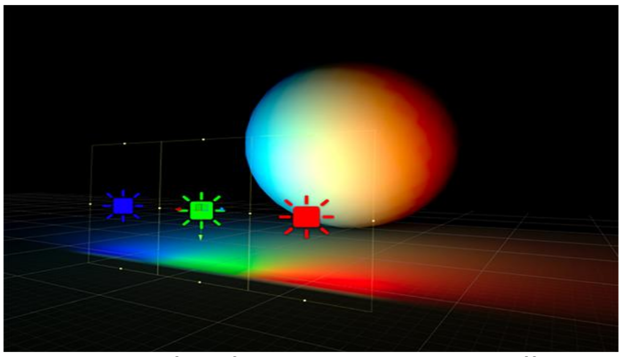
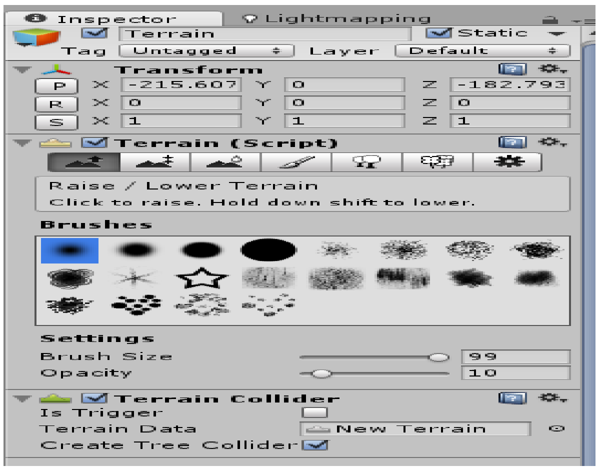


Fig 6 Light is emitted across the surface of an Area Light producing a diffuse light with soft shadowing.

Exterior Design Model:
Environment:
Typically users initialize their environment when they log in by setting environment information for every application they will reference during the session. The Environment Modules package is a tool that simplify shell initialization and lets users easily modify their environment during the session with module files.

Each module file contains the information needed to configure the shell for an application. Once the Modules package is initialized, the environment can be modified on a per-module basis using the module command which interprets module files. Typically module files instruct the module command to alter or set shell environment variables such as PATH, MANPATH, etc. modulefiles may be shared by many users on a system and users may have their own collection to supplement or replace the shared module files.

Modules can be loaded and unloaded dynamically and atomically, in an clean fashion. All popular shells are supported, including bash, ksh, zsh, sh, csh, tcsh, fish, as well as some scripting languages such as perl, ruby, tcl, python, cmake and R.Modules are useful in managing different versions of applications. Modules can also be bundled into met modules that will load an entire suite of different applications [8][9].
Terrain: To create a terrain, Game Object is selected, and it is proceed to create other and skybox and terrain is chosen sequentially. Skybox display the extend of what the virtual world looks like in the entire scene. Terrain asset is created and sculpted. The height is modified by raising it in the inspector to create an uneven surface. The heights are then smoothed out to make it more natural and less jagged using the smooth Height Terrain tool. When the surface have been done, suitable textures are painted onto the terrain using the Paint Texture tool in the inspector


```
Inspector    Lightmapping
✓ Terrain                          ✓ Static
Tag  Untagged        Layer  Default
Transform
P  X -215.607   Y 0   Z -182.793
R  X 0          Y 0   Z 0
S  X 1          Y 1   Z 1
✓ Terrain (Script)
Raise / Lower Terrain
Click to raise. Hold down shift to lower.
Brushes
Settings
Brush Size    99
Opacity       10
✓ Terrain Collider
Is Trigger
Terrain Data          New Terrain
Create Tree Collider ✓
```
Figure 7 Inspector for terrain

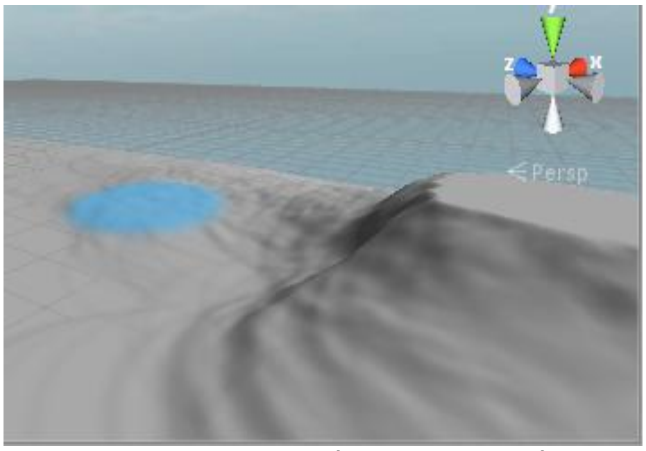


Figure 8 brushing and smoothing for an uneven height for the terrain

The exterior of the house is created using cubes. It is resized and adjusted to make the whole house. For windows, suitable texture is applied onto the material used. Different textures of materials are applied onto the terrain using brush option.

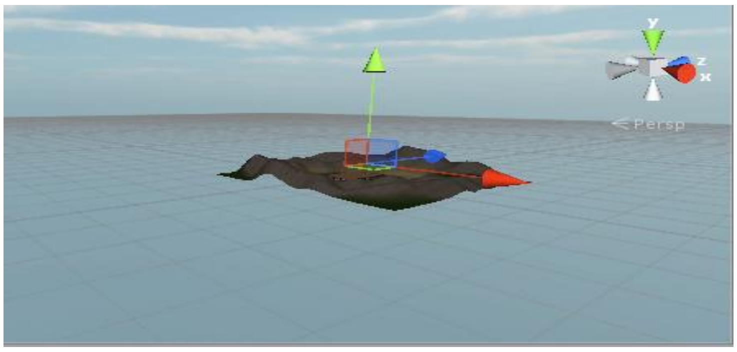


Figure 9 Terrain with skyview, readjusting the position

First Person Controller:

First person controller is used, where it gives user a feeling that they are navigating the showroom themselves. This function allows user to navigate as they desire in the virtual showroom. The height of the First Person Controller function is adjusted to make it to have1 a normal human’s height.

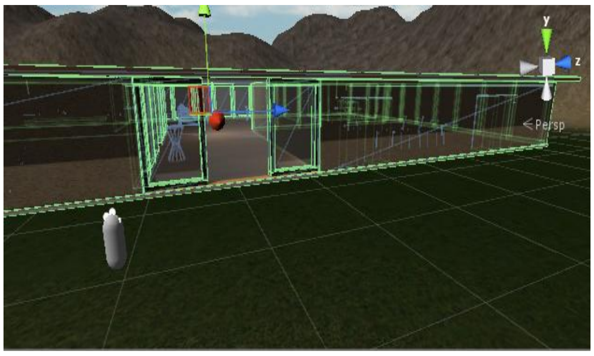


Figure 10 3D house model based on cubes, and first person controller.

GVR controller main:

Main entry point for the Daydream controller API.Touse this API, add this script to a game object in your scene, or use the GvrControllerMain prefab. This is a singleton object. There can only be one object with this script in your scene. To access a controller's state, get a device from GvrControllerInput.GetDevice then query it for state. For example, to the dominant controller's current orientation, use GvrControllerInput.GetDevice;(GvrControllerHand.Dominant).

Lighting and Texture:

Texture:

Furniture for the house is imported. Suitable house furniture models are imported to the scene and scaled. Suitable materials are then applied to the furniture. Some models does not require application of materials as it is imported along with it. The furniture models can be in various file types, such as .fbx, .sbx and .obh formats. Images with higher resolutions are selected to be apply as the materials of the house, as those that has at least 1200 x 800 pixels. Higher resolution will result in a more realistic virtual house. Box collider is added to prevent users from walking through the objects (i.e., walls, furniture) in the scene.

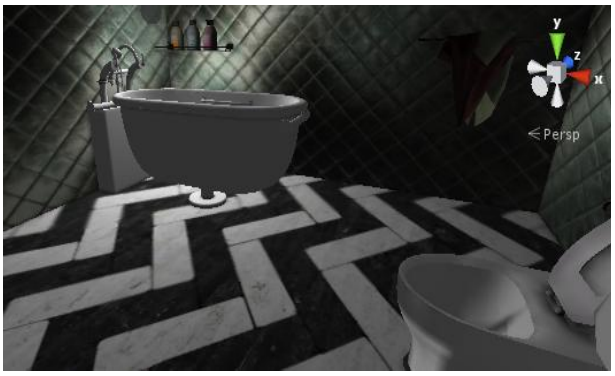


Fig 11 Importing assets such as 3d models and implementing materials

Light mapping:
After every object in the scene is arrange according to desired, the scene is light mapped and baked. Light mapping baking is necessary to reduce render process in real time. Real time is graphics that update quickly so that there is no obvious delay experienced by users [10].

Once light mapping is done, the positions of the 3d models cannot be change. Other than that, the textures and materials needs to remain the same. This is because light mapping has fixed the lighting for the scene involved. The lighting will not change even when objects are being moved. As can be seen in the figure below, the scene displayed before the light mapping process is less realistic. The floors have lighting that is evenly distributed. After light mapping process has be done, it can be seen where the light source coming from even by looking at the floor of the showroom. In figure 17, it can be seen that the light source is coming from near the table and vase. This is less visible from the figure 16, where light mapping has not been done yet.

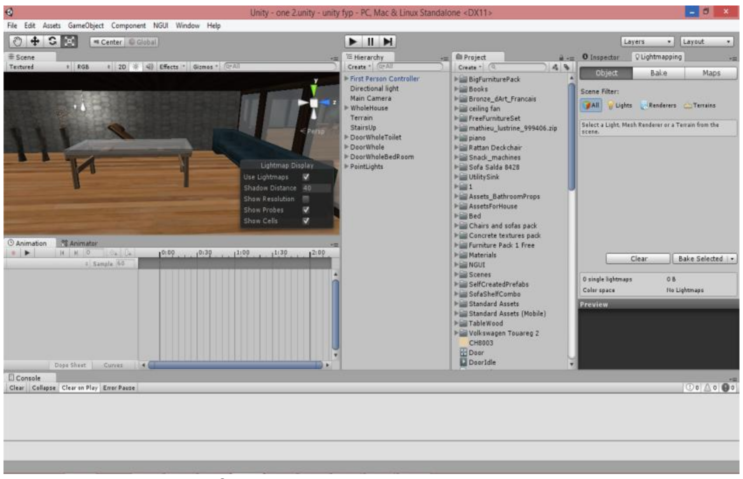


Fig 12 before light mapping is applied on the scene

IV. METHODOLOGY & MODEL

A). PORTAL CULLING ALGORITHM

As in [Luebke and Georges 1995], our algorithm starts from the viewer's cell, considers all of its portals and checks whether they are visible or not: if a portal is visible, the cell connected to the viewer's cell by the portal will be visible as well. The same portal check method can be recursively applied to the adjacent cells connected by a visible portal. More formally, we can define the visible screen area (VSA) of a cell as the part of the viewport where the rendered cell can be seen, i.e. the part of the viewport where no occludes have been rendered. We can notice that the VSA of a cell is usually reduced by passing through a portal. Thus, the portal check method described above can be refined using VSAs instead of the whole viewport of the portals.

We represent a VSA using its axis-aligned bounding rectangle (Figure 2), or in an analogous way, indicating it with ranges on the two viewport axes. This simple representation has both advantages and disadvantages: it is conservative (by definition of the bounding rectangle) and can be handled in constant time (since we are using a fixed-size and axis-aligned representation instead of an arbitrary polygon) but there are situations where the bounding rectangle can be too loose thus leading to the computation of a VSA much bigger than the actual portal area. We chose this solution for its constant performance qualities. The next four subsections illustrate the pseudocode of our algorithm. The first subsection deals with the culling algorithm that exploits cell-to-cell visibility, and is very similar to Luebke and Georgas's algorithm, the other three discuss three optimizations of the algorithm: the first two have been introduced by us and respectively exploit distance from the viewer and cell-to-geometry visibility, the third is an improvement of the solution proposed in [Luebke and Georges 1995] for avoiding multiple renderings of the same cell [2][15].

V. RESULTS

The model used for the tests was a reproduction of the main building of our University, made of 28608 triangles. We started from a 3D model that was fully textured, subdivided into 39 cells and with 56 portals. The cells were too big and dynamic loading and flushing to and from memory were too slow, mainly because of file I/O operations. The lack of a texture caching system (which we are planning to investigate as a future work, together with the possibility of loading each texture on a separate thread) also led to loading time inefficiencies, because textures had to be loaded from storage as a cell became visible. With low-resolution textures, the mean loading time for a cell is about 18 seconds, while the mean loading time of a cell without textures is about 6 seconds (2 seconds for file I/O and 4 seconds for parsing). We therefore decided to substitute textures with colored materials; the model was also subdivided into smaller cells, leading to the final 3D model used for testing and consisting of 93 cells and 114 portals. The mean loading time for the smaller cells is about 2.5 seconds (1 second for file I/O and 1.5 seconds for parsing). The number of triangles per cell varies from a minimum of 102 up to a maximum of 452, and cells have between 2 and 4 portals. The subdivision in cells and portals of a part of the 3D model.

The results are encouraging and the large 3D model can be explored with satisfactory frame rates. The biggest problem encountered was the speed of loading and flushing cell data (both textures and geometry) between the storage and main memory. Moreover, it is not possible to keep more than 2000 triangles in memory. Anyway, these are not limitations of the proposed method, but optimization issues of X3D loading and resource handling code. We plan to solve these issues through the:

Optimization of X3D parsing code to obtain shorter loading times for the cells;
Design of a texture and geometry caching method to reduce the stream of data from storage to main memory;
Design of a predictive scheme to load cells in advance instead of loading them when they become visible.

VI. CONCLUSION AND FUTURE SCOPE

This Paper designing and implementing virtual environments to support design and evaluation of interior spaces. Desktop virtual reality technology has been used as a medium to generate realistic visualizations of the environment in real-time and to let users formulate and refine their design concepts directly in 3D. The proposed methodology emphasizes in the usability of the environment especially for inexperienced users and aims to increase user performance in the process of modeling an interior environment, selecting furniture and decoration elements, and trying out various arrangements. The paper also presented a prototype implementation of the proposed methodology. The results of an initial user evaluation indicated that the environment can be easily used by non-expert users, although there were significant differences in performance. Furthermore, we are planning to enhance the environment with new features that would further support the design process, such as multi-user capabilities for synchronous collaboration between clients and designers, and new interactive components, such as sketch on surfaces or posting of annotations to visualize design requirements and abstract concepts.

REFERENCES

[1] Alessadrao Mulloni, Deniele Nadalutti, Luca Chittaro, " Interactive Walkthrough of Large 3D Moldes of Building on Mobile Device", HCI Lab, 2005.

[2] Aila, T., Miettinen, V. 2004. dPVS: An occlusion culling system for massive dynamic environments. IEEE Computer Graphics and Applications 24, 2, 86–97.

[3] Airey, J. 1990. Increasing update rates in the building walkthrough system with automatic model-space subdivision and potentially visible set calculations. PhD thesis, University of North Carolina at Chapel Hill.

[4] Assarsson, U., M¨oller, T. 2000. Optimized view frustum culling algorithms for bounding boxes. Journal of Graphics Tools 5, 1, 9–22.

[5] Clark, J. H. 1976. Hierarchical geometric models for visible surface algorithms. Communications of the ACM 19, 10, 547–554.

[6] Cohen-or, D., Chrysanthou, Y. L., Silva, C. T., and Durand, D. 2003. A survey of visibility for walkthrough applications. IEEE Transactions on Visualization and Computer Graphics 09, 3, 412–431.

[7] Funkhouser, T. A., S´equin, C. H., Teller, S. J. 1992. Management of large amounts of data in interactive building walkthroughs. In SI3D '92: Proceedings of the Symposium on Interactive 3D Graphics, ACM Press, New York, NY, USA, 11– 20.

[8] Haumont, D., Debeir, O., Sillion, F. 2003. Volumetric cell-and-portal generation. Computer Graphics Forum 22, 3, 303–312.

[9] Hci Lab – University Of Udine. 2006. MobiX3D website. http://hcilab.uniud.it/MobiX3D.

[10] Kumar, S., Manocha, D., Garrett, W., and Lin, M. 1996. Hierarchical back-face computation. In Proceedings of the Eurographics Workshop on Rendering Techniques, Springer-Verlag, Berlin, Germany, 235–244.

[11] Lerner, A., Chrysanthou, Y., Cohen-or, D. 2003. Breaking the walls: scene partitioning and portal creation. In PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications, IEEE Computer Society, Los Alamitos, CA, USA, 303–312.

[12] Lerner, A., Chrysanthou, Y., Cohen-or, D. 2006. Efficient cells-and-portals partitioning: Research articles. Computer Animation and Virtual Worlds 17, 1, 21–40.

[13] Lipman, R. R. 2004. Mobile 3d visualization for steel structures. Automation in Construction 13, 119–125

[14] Luebke, D., Georges, C. 1995. Portals and mirrors: simple, fast evaluation of potentially visible sets. In SI3D '95: Proceedings of the Symposium on Interactive 3D graphics, ACM Press, New York, NY, USA, 105–106.

[15] Marvie, J.-E., Bouatouch, K. 2004. A VRML97-X3D extension for massive scenery management in virtual worlds. In Web3D '04: Proceedings of the 9th International Conference on 3D Web Technology, ACM Press, New York, NY, USA, 145– 153.

[16] Nadalutti, D., Chittaro, L., Buttussi, F. 2006. Rendering of X3D content on mobile devices with OpenGL ES. In Web3D '06: Proceedings of the 11th International Conference on 3D Web Technology, ACM Press, New York, NY, USA, 19– 26.

[17] Nurminen, A. 2006. m-LOMA - A mobile 3D city map. In Web3D '06: Proceedings of the 11th International Conference on 3D Web Technology, ACMPress, New York, NY, USA, 7–18.

[18] Parallelgraphics. 2004. Pocket Cortona. http://.parallelgraphics.com/products/cortonace/.

[19] Pulse Interactive. 2005. Quake Mobile. http://.pulsemobilegames.com/QuakeMobile.html.

[20] Shreiner, D., Woo, M., Neider, J., Davis, T. 2005. OpenGL programming guide: the official guide to learning OpenGL, Version 2 (5th Edition). Addison-Wesley Professional.

[21] Slater, M., Chrysanthou, Y. 1997. View volume culling using a probabilistic caching scheme. In VRST '97: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, ACM Press, New York, NY, USA, 71–77.

[22] Teller, S. J., S´Equin, C. H. 1991. Visibility preprocessing for interactive walkthroughs. In Siggraph '91: Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press, New York, NY, USA.

[23] Naidu Srinivas Kiran Babu, E. Madhusudhana Reddy, "Estimation of Calcaneal Human Foot Gait Estimation from Foot Parameters Measured By a Foot Feature Measurement System Using Machine Learning Algorithms," International Journal of Scientific Research in Computer Science and Engineering, Vol.7, Issue.6, pp.57-65, 2019.

[24] Neha N. Badwaik, Achal J. Wakade, Sudha Shrikanth, "Underground Cable Fault Detection System by Using IoT," International Journal of Scientific Research in Network Security and Communication, Vol.8, Issue.1, pp.25-28, 2020

BIOGRAPHY

Prof. A.P. Khan is anAssistant professor in Computer Engineering Department, D.N.Patel College of Engineering, Shahada, Dist.- Nandurbar(MS). He received Master of Technology (M.Tech) degree in 2012 from RGTU-Bhopal (MP), India. His research interests are Data Mining, Big Data Science, Computer Security, Software Engineering, etc.

Miss. Poonam S. Patel Research Student Computer Engineering Department, D.N.Patel College of Engineering, Shahada, Dist.- Nandurbar(MS). She is pursuing Bachelor of Engineering (BE) degree in 2020 from KBC NMU Jalgaon, MS, India. Her research interests are Data Mining, Computer Networks (wireless Networks), web 2.0 etc.

Miss Lakita K. Patil Research Student Computer Engineering Department, D.N.Patel College of Engineering, Shahada, Dist.- Nandurbar(MS). She is pursuing Bachelor of Engineering (BE) degree in 2020 from KBC NMU Jalgaon, MS, India. Her research interests are Data Mining, Computer Networks (wireless Networks), web 2.0 etc.

Miss. Devyani V. Patil Research Student Computer Engineering Department, D.N.Patel College of Engineering, Shahada, Dist.- Nandurbar(MS). She is pursuing Bachelor of Engineering (BE) degree in 2020 from KBC NMU Jalgaon, MS, India. Her research interests are Data Mining, Computer Networks (wireless Networks), web 2.0 etc.

Miss. Prerna L. Chaudhari Research Student Computer Engineering Department, D.N.Patel College of Engineering, Shahada, Dist.- Nandurbar(MS). She is pursuing Bachelor of Engineering (BE) degree in 2020 from KBC NMU Jalgaon, MS, India. Her research interests are Data Mining, Computer Networks (wireless Networks), web 2.0 etc.