Proceedings of the 2009 IEEE
International Conference on Mechatronics and Automation
August 9 - 12, Changchun, China

Realistic Vehicle Driving Simulator with Dynamic Terrain Deformation

Tao Ni, Dingxuan Zhao and Hongyan Zhang
College of Mechanical Science and Engineering
Jilin University
Changchun, China
ture_nitao@163.com

***Abstract* - Driving simulators are booming in recent years for vehicle system development, human factor study, and training of driving skills. In this paper, a new type of driving simulator is introduced with four Liquid Crystal Displays (LCDs) placed at the corresponding widows position in the driving cab, instead of using commonly front or rear multiple projector for the projection solution. In such a system, the driver sits in the driving cab and drives the vehicle facing the computer generated graphic images on the LCDs just as if looking through the real window. To enhance the visual effect of the simulation, dynamic terrain base on Real-time Optimally Adapting Mesh (ROAM) algorithm is involved to produce the realistic terrain deformation during the vehicle tire-soil interaction. Considering the terrain mesh in tire-soil interacting fields need a more detail level than that of un-touched region and the precise location where deformation takes place in not known until run time, a dynamic resolution for touched area is used to extend the hierarchy of terrain mesh as the terrain deformation takes place. The software of the simulator is designed using open source game engine—Delta3D, which support the most advanced feature of the latest graphics hardware and shader technology, and lack of proprietary vendor lock-in and licensing fees at the same time.**

Index Terms* - *ROAM, terrain deformation, LCD window, Delta3D, perspective correction.

I. INTRODUCTION

The use of simulators for training is an old and well accepted method used in situations where training in real environments is difficult, dangerous or expensive[1]. Moreover, the simulator is indispensable to the test for such driver's critical situations as in a collision, a crash and a sliding. In particular, computer based flight simulator had been used since 1960's, and the driving simulators are booming in recent years, from which a driver can obtain realistic driving feelings through movements, visual displays, and sound effects in virtual driving situations by implementing real-time simulation over the vehicle. It seeks to accurately simulate vehicle movement and display an image to the viewer that is faithful to what would be seen in real-life. Many automotive makers and research institutions have developed and applied their own simulators for purpose of new product design and evaluation, drive safety researching as well as vehicle behavior investigation, such as the IOWA Driving Simulator (IDS)[2] located in The University of Iowa's Center for Computer Aided Design (CCAD) and the National Advanced Driving Simulator (NADS)[3] developed by the National Highway Traffic Safety Administration in United States, etc.

However, despite the booming development of vehicle driving simulator, the relationship of the vehicle and the terrain is ignored by most modern driving simulations at the expense of visual correctness, unfortunately. Although a great deal of research effort has been devoted to the realistic and efficient rendering of terrain data, the existing research mainly focuses on displaying static terrain. Lack of a deformation strategy reflective of the underlying simulation model will hinders the realism of visual system, which leads to limited usability and applicability. As the terrain visualization plays an important role in the visual immersion of driving simulation, not only static terrain models but also dynamic deformable terrain models are involved in the terrain database for producing the terrain surface deformation and tire-soil dynamic interaction. In this paper, Real-time Optimally Adapting Mesh (ROAM)[4] method is applied for generating and updating the deformable terrain.

II. STRUCTURE OF THE DRIVING SIMULATOR

Currently, most of the advanced driving simulators around the world have the following common ground: first of all, the vehicle type is mini car or truck in general; secondly, use a six degree of steward platform to provide the motion presence for driver; thirdly, more than three image generators are used for high-resolution graphic projection; and finally, high performance parallel-processed computers or a group of server computers are involved in the system for real-time graphics, 3D sounds and dynamics.

A. Physical Configuration

The physical configuration of our special vehicle driving simulator consists of a driving cab mounted on a hexapod steward motion platform. The four windows of the driving cab are replaced by four LCDs(Liquid Crystal Displays) of the approximately same size at the corresponding positions. The computer generated visual image is divided into four channels and displays respectively on the front left, front right, side left, side right LCD windows of the driving cab. Currently, a common primary graphic card can at most drive two displays simultaneously, and therefore two graphic cards are plugged in the computer for the driving simulation system. The multiple surrounding displays could provide a 180 degree forward field-of-view for the driver with incredible high immersion.

At the same time, since sound rendering plays an important role in the immersion feeling of a driving simulator, directional audio cueing is provided by multiple speakers placed in and around the vehicle cab. Spatialized sound generation associating with the vehicle power train, tires, wind, passing

978-1-4244-2693-5/09/$25.00 ©2009 IEEE

traffic, and other environmental sources transmitted to the vehicle cabin can significantly enhance the overall realism of a driving simulator.

Several other mechanisms, including the throttle and brake pedal, gun pedal, clutches, steering and manual transmissions, etc., are important to a successful driving simulation. Force feedback produced by a counter torque motor is applied to the steering wheel and brake system to provide proper tactile and proprioceptive cues to the driver. Applied forces are derived from the vehicle performance model to properly account for tire and steering system dynamics. All vehicle instruments are fully operational, including speedometer, tachometer, turn signals, warning lights, and other in-vehicle devices required for specific experimental objectives.

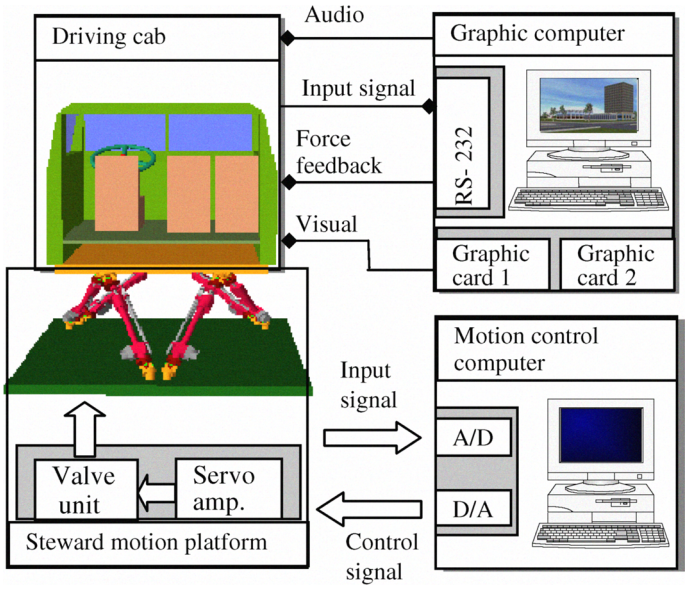

Fig. 1 System physical configuration

The Stewart platform developed in this research has six degrees of freedom and is operated by the six hydraulic cylinders whose maximum strokes are 500 mm. It induces motion cues to the driver with maximum frequencies up to 8 Hz and maximum accelerations exceeding 1.0 g.

Thus, in such a system, the user sits in the driving cab and is surrounded by stereoscopic computer images rendered on four individual displays in the front, left and right side of him/her. The driver input signals such as steering, braking ,clutching and accelerating etc., are sensed and transmitted to the graphic computer through the RS-232C port. Simulation of vehicle dynamics is then performed in real-time to predict the resultant vehicle motion and realistic graphic images as well as noise of driving environment corresponding to driving conditions are generated and updated every frame to provide the driver with continuous visual and aural cues. At the same time, the actual vehicle motion characteristics of ongoing lateral and longitudinal acceleration is calculated based on "washout" algorithms[5] and transmitted to the motion control computer to provide the driver with realistic proprioceptive motion cues.

B. *Software Architecture*

Originally, our special vehicle driving simulator is developed using a commercial Virtual Reality(VR) engine OpenGVS. However, OpenGVS has a node lock license and cost very expensive which limit the popularization of our driving simulator. Now the software of our simulation program is totally redesigned based on an open source game engine Delta3D. The latest source code of this actively maintained engine can be downloaded from a CVS (Concurrent Versioning System) repository. The primary goal of Delta3D is to provide a single, flexible program interface with the basic elements needed by all visualization applications. The most recent upgrades, available in versions 2.2 and later, include adding capability for After Action Review, integration with SCORM-compliant learning management systems (LMS's), and distributed interactive simulation (DIS) networking. Additionally, more applications, created by both government users and civilian companies, continue to be built using Delta3D and its expanding capabilities.

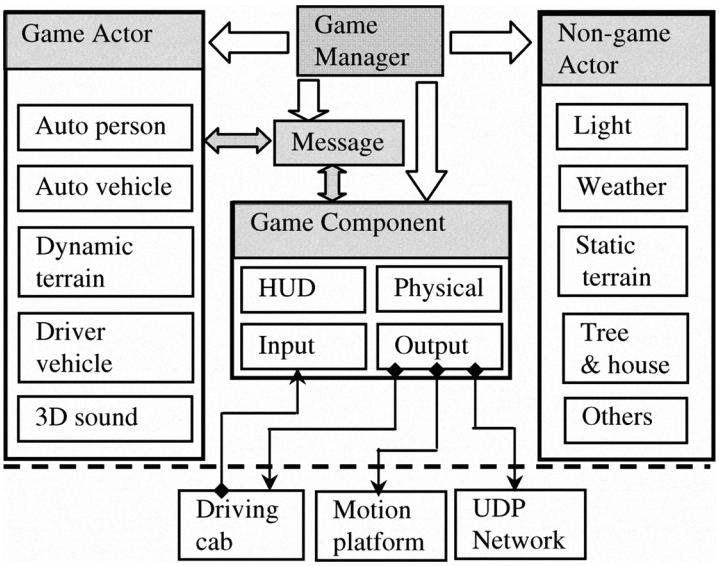

Fig. 2 System software architecture

The Core of our current Delta3D based simulation software is the Game Manager (GM), and it owns all the Actors, Components and Messages. All the active objects such as driving vehicle, autonomous vehicles, moving person, dynamic terrain and the inactive objects including the static terrain, lights, sounds, etc., are induced into the simulation software as Actors, the difference between them is that the former are designed as Game Actors while the later are designed as Non-game Actors. And Components are responsible for receiving all the Messages from the GM and deciding whether processing these Messages or just sending them to a specific Actor. Here, Messages are user defined data used for the communication between Actors and Components. The overall architecture of the driving simulation software is described in Fig. 2. The HUD Component provide the graphic user interface (GUI) to the driver, and the Physical Component is responsible for perform collision detection among all the Actors in the

virtual world. The Input Component just get the manipulation signals through computer's serial port from the driving cab and send these signals to the Vehicle Dynamic Module (VDM) of driving vehicle Actor. It is the VDM that perform the vehicle dynamic calculation and determines the final position as well as velocity of our special type vehicle in the virtual scene. Output Component mainly performs the following functions: the first is to drive the various instruments in the driving cab; the second is to send posture information of the driving vehicle to the motion control computer through RS-232; and the third is to publish the position and rotation of driving vehicle in 3D world to other driving simulators on net.

Considering the data exchange in our driving simulator application is a kind of data broadcasting , User Datagram Protocol (UDP) was adopted in our programming for simplicity and efficient. For a realistic driving simulation programming, the largest portion of CPU time is usually occupied by the vehicle kinetics calculation and graphics processing. To make sure the CPU allocates time slices to process network communication just in time, the tasks of sending and receiving should be launched in two separate threads with relative high priority.

III. GENERATION OF DEFORMABLE TERRAIN

For a realistic driving simulator, it should not only contain the static models, but also dynamic models such as a deformable terrain. Static terrain uses a fixed, rigid terrain mesh to model the landscape which dominate in the past and is far from a perfect solution, whereas dynamic terrain supports runtime surface deformation of the terrain model which could increase the reliability of virtual driving environment greatly.

Dynamic terrain modeling is one of the most popular research topics in 3D computer graphics. Currently, the static models are constructed using MultiGen Paradigm Creator, and the dynamic terrain is created by method of approximately specifying the feature and the dimensions of terrain using drawing utilities. In our application, the heights of dynamic terrain vertices are defined in an elevation map, the elevation map is generally encoded in colors and stored as a grayscale image, in which low intensities represent low-lying land and bright intensities represent high land. Different material features of the terrain such as the plain, the road and the mountain, etc., are specified in a mask file where different range of RGB color represents different region of the terrain. In order for this 3D polygon mesh constructed from the elevation map to "look" like a real terrain and to enhance its realistic appearance, a texture composition technique is used in a terrain-shading model and detail textures are applied for creating the illusion of highly detailed surfaces without consuming exorbitant amounts of memory.

Real-time Optimally Adapting Mesh method, originally devised at the Los Alamos National Laboratory, is selected as an extendable efficient tool for the internal data representation and dynamic updates of the terrain model. It allows controlling the details of the mesh ant to maximize the quality and minimize the number of triangle primitives used in the process[6] by split and merge operation using a Binary Triangle Tree. A split operation replaces a triangle with its two children in the Binary Triangle Tree, while a merge operation replaces two siblings in the Binary Triangle Tree with their parent. This process is shown in Fig. 3. Here, a Binary Triangle Tree is a tree data structure in which each node is a right triangle (an isosceles triangle with one right angle) and either zero or two children. Children are formed by splitting a node along the midline of the triangle (the line between the apex and the center of the base).

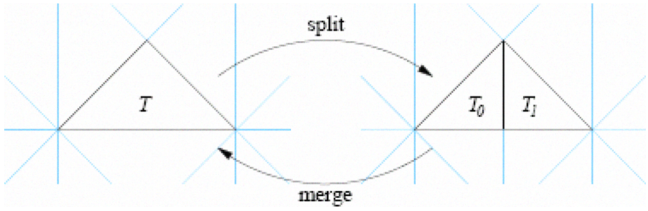

Fig. 3 Split and merge operation

During the generating of dynamic terrain, the ROAM algorithm is a recursive process that starts with a single root node and determines the node at the desired level of detail is or if it should be split into two children, in which case each child is then processed recursively. Determining the desired level of detail for each node is based on two factors: the roughness of terrain (called the variance) and the distance from the current node to the camera[7]. Unless the height field is dynamically altered, the variance value for each node will not change, and the actual decision on whether or not split the node and how deeply it should be split is performed by a *split metric* calculation. This project implements the *split metric* as follows[7]:

$$Split\ metric = variance * width * 2 / distance$$

Here, *variance* is the metric error for the given node, *width* is the width of the whole terrain. And the *distance* is the distance between the center of the base of node and the current camera position. The error metric for a node in a Binary Triangle Tree is relatively simple to compute and takes place entirely on a triangle's hypotenuse. It is calculated as the difference in the interpolated height at that point and the actual height from the height field.

The node being processed is split if the following condition holds:

$$split\ metric > frame\ variance$$

Here, the *frame variance* is a global parameters specifies an upper limit on the desired variance for any node in the current frame.

From above, the general ROAM terrain generating algorithm can be described as follows:

1) Get the elevation of terrain from the specified height map file, and set up the feature of terrain of different regions such as mountain, plain, river and road based on the color information from the specified coverage map file.

2) Build a Binary Triangle Tree data structure to represent the geometric properties of the rendered terrain.

3) Create two queues, the split queue and the merge queue, to keep the priorities for each individual triangle in the mesh.

4) Traverse every node of the terrain and implant the split metric calculation to decide whether the current node needed to be split or not.

5) Implement the split and merge operations for generating the dynamic terrain mesh with different level of detail.

IV. DYNAMIC INTERACTION BETWEEN VEHICLE AND TERRAIN

The real-time simulation and visualization of vehicle-terrain interaction, including tire-soil dynamics simulation, will bring significant visual effect to the simulation system, and also is a challenging computational problem. Driving simulators that do not account for tire-soil interaction are incomplete because they use a model that produces crude, overly-simplified visual interpretations of physical reality[8].

In a dynamic terrain application, surface geometry, color and material properties could change over time. Tread marks, footprints, and other residual information can remain in soft terrain surfaces after the causal source has left the affected area. Specialized methods are required to achieve convincing visual displays without hindering runtime performance. As far as our driving simulation is concerned, terrain deformation is often sparse and the precise location and degree of the deformation is not known until run time which is determined by the tires position of driving vehicle. A pre-constructed "prepare for the worst" hierarchy that represents everywhere of the terrain with high resolution, can be unnecessarily space inefficient and memory wasting[9]. Therefore, in order to successfully use a dynamic terrain solution in large-scale terrain visualizations, the multi-resolution strategy and dynamic terrain solution must be able to co-exist.

Dynamic Extension of Resolution (DEXTER) is used to add resolution to a mesh in a manner that seamlessly integrates with the level of detail method. Increasing the resolution of the mesh is necessary in cases where the mesh density is too sparse for effectively reflecting terrain deformations. DEXTER allows geometry to be added to an in-memory mesh representation of a height field at runtime by forgoing the resolution and density specifications of the original mesh.

As mentioned above, the ROAM algorithm is built upon a Binary Triangle Tree structure that supplies triangle information for the split and merge operations[6]. The split produces a new set of children nodes in the tree, which uses more polygons to represent the same area. By recursively splitting the polygon(s) at identified leaf nodes, dynamic terrain systems tend to add geometry of greater resolution. A stop condition, such as a maximum depth, is employed to avoid the production of overly complex geometry. As a result, the new, high-detail geometry can be manipulated to create high fidelity terrain deformations. At the same time, the Dynamic Extension to ROAM uses the update process in order to maintain other vertex values, like texture coordinates and material properties.

Considering the main purpose of a driving simulator is to improve the road sense of a driver and avoiding of a high computational cost, we are not aimed to build a physically accurate model of soil mechanics to achieve realistic terrain deformation by computing the soil slippage and soil mass displacement[10] or by representing the terrain surface as an elastic sheet made up of particle-mass[11]. In this paper, appearance-based solution is applied create a visually plausible rendition of terrain dynamics without the conditional constraint of using a physically accurate model. The solution executes quickly because the computational complexity associated with a physics-based animation is bypassed. It uses a four step execution cycle to create a visually-convincing depiction of terrain surface interactivity. The four steps are:

1) Get all the tires position of driver vehicle and obtain the corresponding terrain feature information at tire-terrain contact points. One important thing to note is that, in order to realize a realistic rendering effect just as in the real world, we only perform the visual dynamic interaction effect on region of muddy soil, not on region of road. Therefore, if the contacted terrain feature is road material, the following process will not go on.

2) Implement splitting operation at the tire-soil contacting region and their neighboring regions, so as to increase the resolution of the required deformable terrain area.

3) Change the elevation as well as the texture of the tire-soil contacting areas to simulate the effect of vehicle trail.

4) Implement the merge operation at the previous split terrain regions when the distance between the region and camera is exceed a certain defined value.

Each frame, the four steps are executed and the final surface configuration is rendered to the screen. Continual, incremental updating causes the surface changes to accumulate over time, which gives a tractable history of events and makes the animation more realistic. The appearance-based ROAM terrain deformation solution carries out real-time performance with convincing visual phenomena(shown in Fig.4), and most suitable for a small, local updates on the terrain, such as tire impressions.

Fig. 4 Scene of dynamic terrain deformation

V. CONFIGURATION OF MULTISCREEN DISPLAYS

The solution of a wide screen with several front or rear projectors is widely accepted by most of the driving simulators. However, in such a system, the distance between the driving cab and the surrounding screen can not be neglected which will hinder the driver from getting an accurate distance perception while driving the vehicle. The disadvantage becomes more obvious as the driver drives much more closer to an object. In order to overcome the above shortcomings, multiple LCD window solution is applied in our current simulator system. That is, LCD monitor is used to replace the window of the driving cab and the driver drives the vehicle looking at the surrounding LCD monitor just as if looking through the window of real vehicle. The four-screen design shown in Fig.5 is the same with that in the driving cab. The two front screens are placed parallel in front of the driver, and the two side screens are positioned perpendicular to the front screens on the right side and left side respectively with one edge overlap with that of front screen. The virtual driving environments are generated and displayed on each of the four screens with specified view rotations (0 degree for the front displays, -90 and 90 degrees for the side left and side right display respectively) and perspective corrections. Considering the potential low-cost solution, the two displays in the front are driven by graphic card 1 while the displays on the side are

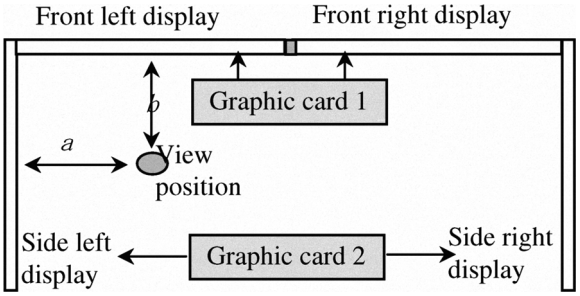

Fig. 5 Arrangement of LCD monitor

driven by graphic card 2, so as to ensure the system will still work even if there is only one graphic card in the computer.

Currently, most commercial VR software such as OpenGVS and Vega Prime have provide easy to use functions for the 3D graphic image joining in Stereo displays and other multiple screen implementations. However, in a Delta3d based simulation program, the projection matrix for different projection window has to be recalculated with off-axis correction so as to obtain a continuous and integrated multi-screen visual effect.

In a window-based projection solution, how the 3D virtual environments are viewed and finally shown in the monitor is determined by the position of the projection plane and the position of the view point. In our system, the projection plane locations correspond to the locations of the LCD window in driving cab and the position of view point correspond to that of driver's eyes. Thus, the general projective projection matrix P can be described as follows:

$$P = \begin{bmatrix} \frac{-2Z_{near}}{W} & 0 & shearX & 0 \\ 0 & \frac{-2Z_{near}}{H} & shearY & 0 \\ 0 & 0 & \frac{Z_{near}+Z_{far}}{Z_{near}-Z_{far}} & -\frac{2Z_{near}Z_{far}}{Z_{near}-Z_{far}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Here, Parameters Z_{near} and Z_{far} specify the distances to the near and far depth clipping planes for the view frustum. Both distances must be positive. Parameters W and H represent the width and height of the LCD display respectively. Parameter *shearX* and *shearY* are normalized distance offset between the position view point and the center of the projection plane. Consider the driver's eyes positions are vertically at the middle of the display in normal conditions of our driving simulator, the parameter of *shearY* for all the projection matrix of the four display is set to zero, and *shearX* for the front left display represented as $shear_{FL}$, for the front right display represented as $shear_{FR}$, for the side left display represented as $shear_{SL}$, for the side right display represented as $shear_{SR}$ can be calculated as follows:

$$Shear_{FL} = 1 - \frac{2a}{W}, Shear_{FR} = 3 - \frac{2a}{W}$$

$$Shear_{SL} = shear_{SR} = -(1 - \frac{2b}{W})$$

Here, a and b is the distance from the driver's eye point to the front and side LCD window respectively.

(a) Side displays

(b) Front displays

Fig. 6 Scene of multiple displays integrating effect

The screen joining effects are shown in Fig.6. The computer graphics displayed in different LCD window are integrated each other to provide the driver a whole and immersive virtual driving environment with wide field of view.

VI. CONCLUSIONS

In this paper, we have presented the development of a driving simulator, which has been used in several driving schools for military special vehicles. In such a system, a rebuilt driving cab with four LCD windows provides the basic hardware of driving environment, the 6 DOF steward motion

platform is selectable for a low-cost solution. The software of the simulator is designed on basis of Delta3D open source engine. It provides programmer with easy to use API and at the same time achieves real-time performance by using new features of the latest graphics hardware and shader technology.

To enhance the visual effect of the simulator' graphic system, not only static terrain, but also dynamic terrain are involved. The key techniques such as dynamic resolution for deformable terrain based on ROAM and realistic graphic rendering for tire-soil dynamic interaction , not only provide satisfying solutions for driving simulation, but also fit for some other 3D virtual reality fields, including simulation of construction or destruction operation, a virtual battlefield and navigational simulation.

REFERENCES

[1] M. King, B. Zhu, and S. Tang, "Optimal path planning," *Mobile Robots*, vol. 8, no. 2, pp. 520-531, March 2001.

[2] H. Simpson, *Dumb Robots*, 3rd ed., Springfield: UOS Press, 2004, pp.6-9.

[3] M. King and B. Zhu, "Gaming strategies," in Path Planning to the West, vol. II, S. Tang and M. King, Eds. Xian: Jiaoda Press, 1998, pp. 158-176.

[4] B. Simpson, et al, "Title of paper goes here if known," unpublished.

[5] J.-G. Lu, "Title of paper with only the first word capitalized," *J. Name Stand. Abbrev.*, in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," *IEEE Translated J. Magn. Japan*, vol. 2, pp. 740-741, August 1987 [*Digest 9th Annual Conf. Magnetics Japan*, p. 301, 1982].

[7] M. Young, *The Technical Writer's Handbook*, Mill Valley, CA: University SCIENCE, 1989.

[8] Kwon Son, San-Hwa Goo, and Kyung-Hyun Choi ect., "A Driving Simulator of Construction Vehicles," *Internation Journal of the Korean Society of Precision Engineering*. vol. 2, no. 4, pp.12-22, 2001.

[9] J. Clerk Maxwell, "A Treatise on Electricity and Magnetism", 3rd ed., vol. 2. Oxford: Clarendon, pp.68–73,1892.

[10] Haug. E.J.. "Feasibility Study and Conceptual Design of a National Advanced Driving Simulator," DOT HS 807 596, NHTSA, 1990.

[11] Mark Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Miller, Charles Aldrich, and Mark B. Mineev-Weinstein. "ROAMing Terrain: Real-Time Optimally Adapting Meshes", *IEEE Visualization 97*, pp. 81-88, 1997.

[12] R. Nicole, "Title of paper with only first word capitalized," *J. Name Stand. Abbrev., in press*.

[13] Michael Hesse, and Marina L. "An Efficient Algorithm for Real-Time 3D Terrain Walkthrough", *International Journal of CAD/CAM.* vol. 3, No. 2, pp. 111-117, 2003.

[14] Derek Bradley, "Evaluation of Real-Time Continuous Terrain Level of Detail Algorithm," Carleton University, 2003, URL: http://www.derekbradley.ca/Papers/carleton03_TerrainLOD.pdf

[15] Anthony S. Aquilio. "A Framework for Dynamic Terrain with Application in Off-road Ground Vehicles Simulation," *Doctor Dissertation, Georgia State University,* 2006.

[16] Yefei He, James Cremer, and Yiannis Papelis, "Real-time Extendible Resolution Display of On-line Dynamic Terrain," *Proceedings of Graphic Interface, Calgary, Alberta,* pp. 27-29, 2002.

[17] X. Li and J. M. Moshell, "Modeling soil: realtime dynamic models for soil slippage and manipulation," *Proceedings of the 20th annual conference on Computer graphics and interactive techniques: ACM Press,* 1993.

[18] B. Chanclou, A. Luciani, and A. Habibi, "Physical Models of Loose Soils Dynamically Marked by a Moving Object.," *Computer Animation*, pp. 27-35, 1996.