DYNAMIC LANDSCAPE GENERATION USING PAGE MANAGEMENT

Maurice Danaher

School of Computer and Information Science,
Edith Cowan University, Mt Lawley, Western Australia 6050
Email: m.danaher@cowan.edu.au

ABSTRACT

Landscape visualisation is the process of recreating a natural environment and displaying it in an interactive graphical simulation. Current systems that use large datasets to represent the terrain have a number of drawbacks including large storage requirements, low level of detail and overcrowding in multiuser games. In most systems when the landscape is stored to disk the terrain area is quite small or conversely if the area is large the detail is quite low. Here an approach is described in which the terrain is procedurally generated as required. The terrain is produced in the form of blocks and displayed using an innovative page management technique. This approach allows for the generation of a detailed environment, participation by a very large number of players in multiplayer games and easy download of the environment generator via the WWW.

Keywords: Web-based systems, games programming

1. INTRODUCTION

Many games and real-time graphics simulations are set in outdoor landscapes. The process of continuously rendering the landscape to the screen in a real-time manner is known as terrain visualisation. The complexity and detail of the images that are displayed determine the realism of the scene.

The work presented in this paper addresses a number of drawbacks in Internet-based games and graphics simulation systems. The limitations listed below are due to the need to store large datasets representing the terrain to be visualised. These limitations are: (1) in multiplayer games the environment quickly becomes overcrowded when more than a dozen or so players join; (2) these environments demand a lot of storage space and they are usually distributed by CD ROM rather than via the WWW; (3) in order to display relatively large terrain areas the detail must be drastically reduced; (4) to maintain a reasonable frame rate the terrain area and the detail must be kept within certain limits; (5) user/player participation is reduced because the environment cannot be downloaded via the WWW.

Here we are proposing an approach in which the terrain is procedurally generated as required. This approach allows for the generation of a detailed environment, participation by a very large number of players in multiplayer Internet based games and easy download of the environment generator via the WWW.

2. TERRAIN VISUALISATION

Terrain visualisation typically renders a dataset that represents a terrain. These datasets consist of height values sampled at regular grid intervals. By constructing a lattice in three-dimensional space, and using the values in the datasets to displace the intersections of the lattice, we create a renderable mesh of triangles.

The main problem with the use of these simple meshes is the large number of triangles involved. Modern hardware is not capable of rendering in real time the amount of triangles necessary in a mesh to accurately depict a landscape over a reasonably large area.

Many present solutions to this problem involve rendering only objects that are close to the user's position and using a fog effect to hide the missing detail. Another common solution is to decrease the resolution at which the terrain is sampled and hence the resolution of the lattice used for the rendered triangle mesh. This results in fewer but larger triangles. The terrain in this case is visible for a great distance but severely lacking in detail.

Viewed at or near ground level most of the triangles in these meshes are distant from the user. After perspective viewing is applied to the image these distant triangles will only occupy a few pixels on the screen. Lindstrom et al [Linds96a, Linds98b] developed techniques for creating meshes that involved different triangle sizes. These meshes use smaller triangles near the user's viewpoint where detail is important, and larger triangles at areas distant from the user. By using different meshes at different viewpoints a user could explore a terrain richer in detail and larger in size. These meshes are referred to as continuous level of detail meshes, abbreviated to CLOD, or more commonly, LOD meshes.

This method was not without some problems though. The major problem occurs as the user approaches large triangles in the distance. These triangles are spilt into multiple triangles and the user can see the sudden increase in detail. That is, details in the terrain would suddenly appear when a user got close enough. This is known as *popping*. Rottger et al [Rottg98] devised a geomorphing algorithm that alleviated the effects of popping. This algorithm worked by detecting sharp changes in the terrain and using more detail to define these areas when viewed from a distance.

These techniques discussed so far bring us to the forefront of work in the field to date. The bulk of the research being currently conducted is focused on providing higher levels of detail. Though this has resulted in better quality images, there still remains the problem of using larger terrains. During the last five years there has been little progress in relation to increasing the size of terrains used in simulations. The increase in terrain sizes is due mainly to the increase in storage space provided by storage devices.

Currently terrain size is dependent on the storage limitations of the computer on which the simulation runs. Often the dataset for a detailed landscape of fair size may run into hundreds of megabytes. For extremely large datasets used in detailed simulations of entire planets the storage space is measured in gigabytes. Typically solutions to storage problems involve extrapolation or prediction of extra detail not stored in the dataset. Some novel approaches attempt compression of the dataset [Savch00]. These approaches have not proved to be very effective however.

3. PAGE MANAGEMENT APPROACH

Our solution is based on the assumption that the terrain to be visualised is a terrain of fantasy, one that does not exist in real life. The solution involves the creation of a viewing system that procedurally generates all the graphics that are to be displayed in real time. The terrain is created only around the user's position.

To maintain a terrain around a user's position we create the terrain in small blocks. These blocks join together to form the terrain the user sees. We will refer to these blocks as terrain pages. We then use a page management approach to display these blocks as required. The advantage of this paging approach is that as the user travels across the landscape new pages can be created and added. This is considerably faster than reproducing the entire scene around the users point of view.

Our work involves developing and implementing methods for (1) creating the terrain blocks and (2) performing the page management. The terrain blocks are created from height field data and level of detail (LOD) algorithms. The page management system is based on a spherical page wrapping approach.

Height field data is used to specify the height of the landscape at regular intervals. This height field data is commonly derived from satellite photographs. The photograph is grey scaled according to height, and the grey scale picture is stored as an array. The distance value is determined by dividing the distance covered by the photograph by the number of pixels used in the image. The landscape is generated as a mesh of triangular polygons based on this height field data.

The goal of a LOD algorithm is to simplify the landscape mesh in appropriate places so as to reduce the number of triangles used while maintaining the quality of the scene as much as possible. The major task is to select which areas of a terrain are going to be optimised and how much optimisation is going to be applied to those parts of the terrain.

Our LOD algorithm is based on an adaptive quad-tree refinement algorithm [Wrigh00]. This recursive algorithm utilises a data structure that stores a square that is optionally made up of four other squares which in turn are optionally made up of four other squares.

Figure 1 shows two images that were generated with this algorithm. The left image demonstrates detail reduction and the right image is a level-of-detail terrain visible in real-time.

We have named our approach to page management *the spherical offset method*. This novel approach is very efficient and produces very good results. We use the term *map* when referring to the entire landscape and *submap* when referring to that part of the map that represents the user's visual vicinity. The LOD algorithm operates directly on the submap and the terrain is produced as a collection of tessellating pages. As the user moves a distance equivalent to one page the submap is adjusted around the user's position. New pages are created as the user comes within viewing distance of them, and, pages are removed as the user moves away from them.

This method is fundamentally different to existing page management techniques. In existing methods a user would remain centred in the submap with the pages changing. Here the user moves across the submap. For example, a user travelling in a straight line will move through different pages in the submap. When a user encounters the edge of the submap they will reappear on the other side of the submap. An example of a user moving to the right is shown in Figure 2. As the user moves from square 5 to square 6 new pages are loaded into 1, 4 and 7. The user can always see one page ahead. As the user moves off square 6 he/she moves onto square 4 which is displayed in front, i.e. the view has wrapped around to the other side of the submap. It should be noted that the user does not experience any discontinuity in viewing as he/she can at all times see one page ahead.

This method is far more efficient than existing techniques as far fewer pages are processed when the submap is updated. In figure 2 when a user moves one page to the right three new pages overwrite three old pages. In traditional page management techniques the same process involves creating three new pages and moving six existing pages.

The terrain pages for any particular part of the landscape are created in an identical manner every time that area is revisited. This is achieved by using the same seed values when generating the mesh for a particular page. It is the efficient page management and the easy integration with the other components of a terrain simulation that make this approach to terrain visualisation highly desirable.

4. CONCLUSION

Here we have presented a method for producing a graphical simulation in which the virtual environment is continuously produced without the need for a large terrain dataset. At all times the environment is procedurally generated in correlation with other users.

This technology allows for graphically intensive Internet-based games and virtual environments to be downloaded easily. The approximate download size is 1-2 Mbytes as opposed to current systems which can be in the order of many hundreds of Mbytes. This easy distribution of the virtual environment encourages user participation.

The constrained size of current environments limits the number of participants in multiuser games. Simulations using our methodology can be very large in size and allow thousands of players to participate without overcrowding.

5. REFERENCES

[Linds96a] Lindstrom et al., Real Time Continuous Level of Detail Rendering of Height Fields, *Proceedings of SIGGRAPH*, pp109-118, 1996.

[Linde98b] Lindstrom, P and Turk, G., Fast and Memory Efficient Polygonal Simplification, *IEEE Visualization*, pp 279-286, 1998.

[Rottg98] Rottger et al., Real-Time Generation of Continuous Levels of Detail for Height Fields, *Proceedings of SIGGRAPH*, 1998.

[Savch00] Savchenko, S. *3D Graphics Programming Games and Beyond. Indianapolis*: Sams Publishing, 2000.

[Wrigh00] Wright T., Continuous LOD Terrain Meshing Using Quadtrees. *Proceedings of SIGGRAPH*, 2000.

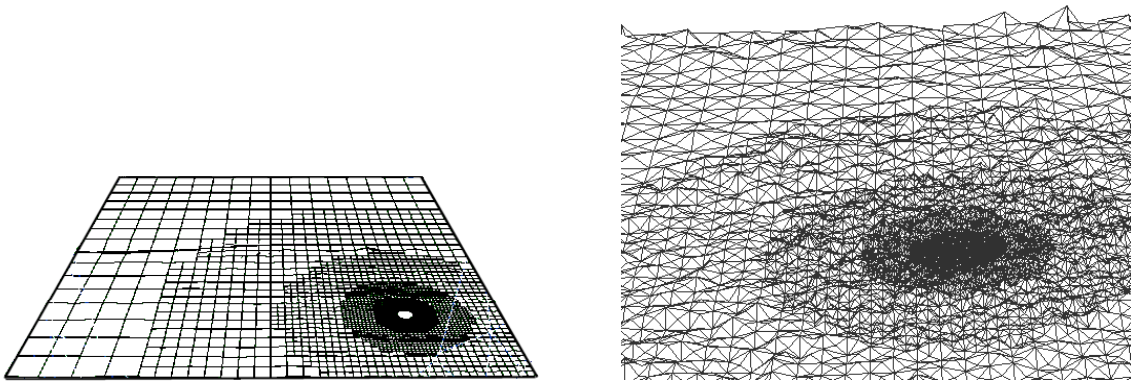

Figure 1: Meshes generated by our CLOD algorithm

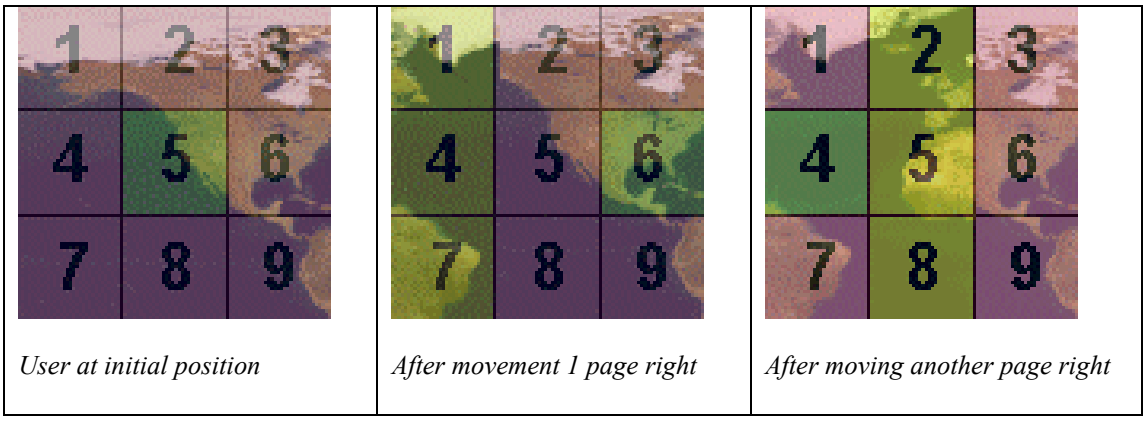

Figure 2: Spherical offset page management