18th World IMACS / MODSIM Congress, Cairns, Australia 13-17 July 2009
http://mssanz.org.au/modsim09

Using game engine technology to create real-time interactive environments to assist in planning and visual assessment for infrastructure

Philip Greenwood[1], Jesse Sago[1], Sam Richmond[1], Vivi Chau[1]

[1]ASPECT Studios - Digital Studio, Melbourne, Australia
Email: philg@aspect.net.au

Abstract: In the field of landscape and infrastructure project planning, real-time 3D environments provide a useful complement to traditional methods of site assessment such as aerial photography, plans and maps. A virtual model gives the user an insight into the intricacies of the site and how site features relate to each other. In particular, use of a real-time 3D model of the landscape prior to an actual site visit enables a much more efficient assessment while in the field.

Recent years have seen the release of a range of game development platforms that greatly facilitate the development of real-time 3D models. These game development platforms or game engines offer great extensions over earlier technologies such as Virtual Reality Mark-up Language (VRML).

Game engines provide a suite of visual development tools in addition to reusable software components. These tools are an integrated development environment to enable simplified, rapid development of interactive models and environments. They primarily provide powerful render engines that allow the visualisation of complex, highly detailed landscapes in 3D in real-time. They also typically provide a scripting interface, physics engine, sound, animation, artificial intelligence, and networking, which enable the development of sophisticated interactivity. Some popular commercial game engines include the Torque engine, Quest3d, Unity3d and Virtools.

We present a new software application ('ASPECT Studios 3D Viewer') which was built using the Virtools game development platform for the purpose of visualising digital terrain models and typically also incorporating buildings and vegetation. The application is designed for visualising areas up to approximately to 100 km2. As well as a navigation system, the application also provides tools for the user to analyse the model in various ways as well as modify and maintain the model. Thus the 3D Viewer itself is best thought of as a set of interactivity tools independent of the model, or content.

The application was produced within an established landscape architecture studio in a response to the needs of industry. It particularly addresses the requirements of tenders offered by government departments. The 3D Viewer is used by landscape architects and urban planners to help resolve planning issues, in particular visual assessment issues. It has found widest applications for models of urban areas, although it has been used for rural areas.

Keywords: *Game Engines, Virtual Reality, Landscape Visualisation, Urban Planning*

1. INTRODUCTION

We aimed to develop a software application for use by landscape architects and urban planners to help resolve planning issues through interactive visualisation and analysis of digital terrain models with incorporation of buildings and vegetation.

In previous years simple real-time environments were produced by us and others primarily using VRML. While VRML advanced the field of visualization of 3D objects, it is only capable of quite static models and minimal interactivity and so its application for landscape planning and development is limited. In addition it is quite difficult to abstract interaction features into reusable components, making it onerous to construct full featured models after the assets had been created, even if the features are very similar between models. Therefore models developed in VRML are usually written as singular entities.

Another limitation of VRML is that the format does not support the real-time rendering of either shadows or reflections. Navigation through the models is typically limited by that of the VRML browser, and there is no provision for multi user environments.

Google Earth is another technology that offers special benefits and is well suited for viewing landscape models over a very wide area. Google earth's spherical coordinate system is of clear benefit for visualisation of large areas. However its main drawback is that the terrain models are not highly accurate. Also, while the aerial photography is of high quality in many areas, some areas do not have high resolution images. Game engines allow easier development of sophisticated and customised interactivity with the landscape, and enable the user to have full control over the terrain model.

To select the gaming engine for use, we assessed some popular commercial game engines including the Torque engine, Quest3d, Unity3d and Virtools. Our selection criteria were: affordability; the capacity to import information in real-time, built features such as shadowing, the right balance between flexibility and ease of use, and suitable for rapid development of custom features by a small development team. We excluded Unity3d because at the time it was limited to Macintosh computers; we chose Virtools over other options because we already had some access to it and it was comparable on other selection criteria.

The Virtools game development platform has a major strength in development environment. It has a large number of preassembled software components that allow the programmer to quickly and efficiently operate at a very high level. This enables model development to proceed much more rapidly than in traditional programming environments.

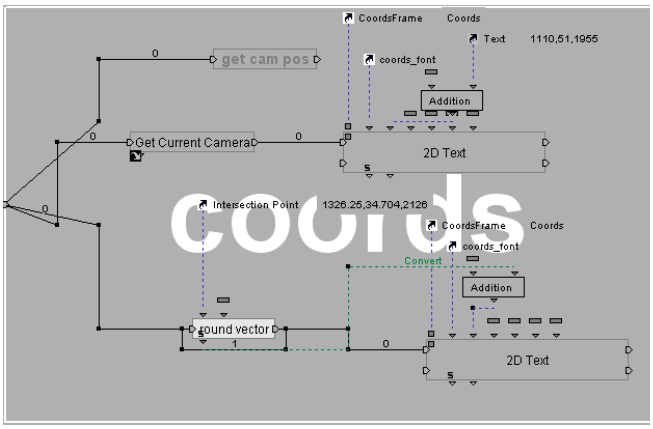

Figure 1. An illustration of Virtools visual scripting.

Virtools scripting can be performed with either a traditional type of object oriented language, or through a visual scripting interface (Figure 1). Typically the programmer uses a combination of these approaches. We have found the latter increases the speed and transparency of software development. New software components can also be added to either the scripting language or the visual scripting interface using the C++ programming language.

We combine Virtools with a Windows application to produce a highly interactive software application for the interactive visualisation and analysis of real-time environments of landscape.

The rationale behind this project was needs-driven in the context of the commercial setting of a landscape architecture and urban planning company with a dedicated 3D visualisation studio. Existing clients (mainly local government urban planners and GIS specialists) specifically requested a software package that allowed them to navigate through a city model freely and enable them to make better use of their data for the purpose of decision making and communication. Specific features requested included shadowing, distance measurements, importing of new scenarios (such as new buildings), visualisation of future building envelopes for planners, GIS data visualisation overlaid on their city models and viewshed analysis. To be

competitive, the software needed to be user-friendly, simple, affordable and built quickly. We responded to these requests by synthesising our existing services into a new software package.

Our rationale is somewhat different to that within a purely academic setting. We recognise that we are not the only group to successfully use game engines for visualisation of landscape. For example, (Stock et al 2008) have developed a suite of tools SEIVE viewer, which features GIS data interoperability, which has found applications particularly in rural landscape visualisation. ASPECT Studios 3D Viewer was developed in a commercial context and as such our project was developed in parallel with developments elsewhere in the field, but we do not claim that our development has necessarily built directly on published works.

2. METHODS

A model of a typical production pipeline for our real-time model development is shown in Figure 2. We produced a Windows software application written in the C# language to encapsulate the software component provided with the Virtools game engine for displaying the model. The Windows application also contained most of the user interface.

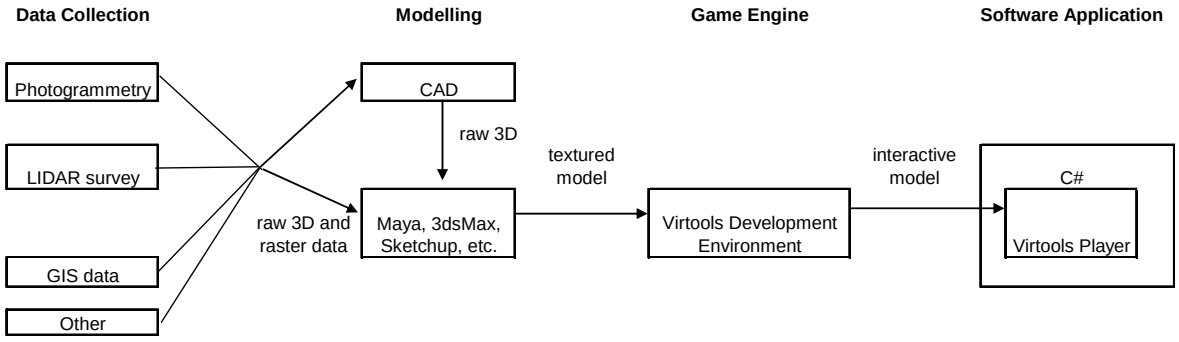

Figure 2. Simplified model of a typical production pipeline for our real-time model development.

The user interface elements such as menus, buttons, and dialogues, were implemented in a Windows application so that existing standard technologies could be exploited, rather than using the game engine itself to reinvent these things. However it was appropriate to implement some user interfaces within the game engine, such as the Navigation Control, Notification Area, and Coordinate Display (Figure 3).

The content of the models can be generated in a variety of ways. The 3D assets of the models can be extracted from LIDAR surveys, photogrammetry, or manually constructed in a 3D modelling package from other data (such as GIS data). In every project the type and amount of data available is different and a full discussion of the model content generation is beyond the scope of this paper. Once the content of the model is generated, the content is divided into the categories of terrain, buildings and vegetation.

The terrain model holds a special place over other 3D objects in the model. In effect the terrain becomes the fundamental part of the space itself. Even though the terrain is embedded in three dimensional space, we need only two parameters to indentify a point in the terrain because the terrain is topologically equivalent to a plane. This is important because it provides a way to convert two dimensional maps to three dimensional ones, by assuming that position of an object in the map is on the ground, which is true in almost every practical situation (thanks to gravity). Converting two dimensional maps to three dimensions in this way is particularly important for GIS visualisation. For this reason the terrain is fixed for each model and cannot be altered by the user.

3. FEATURES OF THE 3D VIEWER

The user interface is shown in Figure 3, and described in detail below.

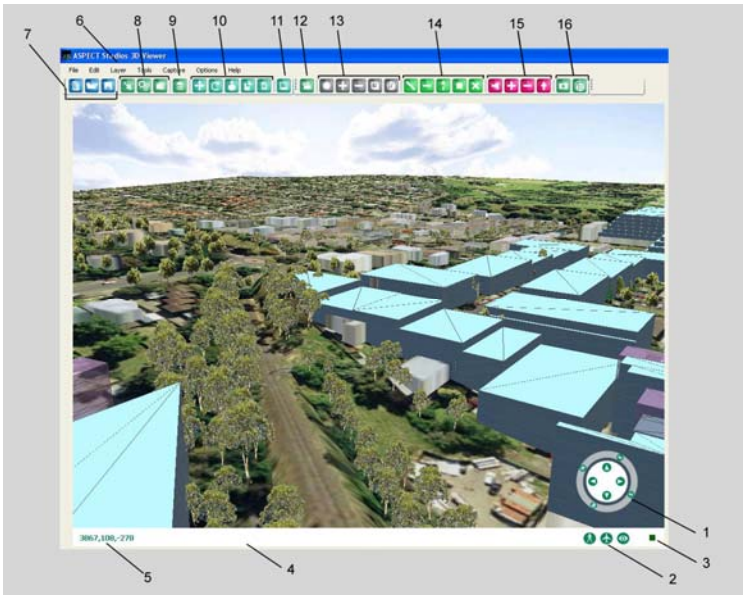

Figure 3. Legend of the 3D Viewer Screen.

1. Navigation Control – Move around in the model i.e., left / right / up / down
2. Walk, Fly and Plan views – Change the viewing mode from walk to fly to plan view
3. Extra Controls – Switch to separate Navigation controls
4. Notification Area – Messages and information regarding current tool in use
5. Coordinates – The current position and height in projected GIS coordinates
6. Menu – Features can be operated from the menu as well as toolbars
7. Standard Tools – New file, Open file and Save
8. Import Tools – Import .3DS 3D objects and vegetation data
9. Layer Tool – Organise imported objects onto layers that can be shown, hidden and saved
10. Edit Object Tools - Move/scale/rotate/copy/delete imported 3D objects
11. Facade Edit tool – Edit specified building's textured facades
12. Saved Views – Save camera positions and create fly-throughs
13. Shadow Tool – Analyse shadows in the model
14. Measure Tool – Make measurements in the model
15. Viewshed Tool - Analyse line of sight in the model
16. Capture Tool – Save screenshots and videos

3.1. Navigation Control

The most important interactivity feature in any real-time 3D environment is the navigation feature. Whilst at first it may seem like a trivial problem, the tool must allow the user to travel through three dimensional space at different scales and speeds, simulate different modes of interaction, and incorporate it into a unified user interface that is easy and intuitive to use and extremely robust. More effort went into development of this tool than any other. Indeed the users overall experience of the environment is determined by how the navigation control works.

There are three modes of interaction; flying, walking and plan view. Each view is a perspective projection with a default focal length of 35mm (user is also able to choose focal length). The canonical view mode is the flying view and the walking and plan views are referenced from the flying view. The walking view is the position on the surface of the terrain directly below the flying view, and the plan view is a fixed orientation from a chosen height above the flying view.

In each view user clicks and drags in the navigation icon to move (Figure 4). Movement speed is proportional to the cube of the distance dragged vertically. This allows fine control at low speed yet gives the user the

ability to move around quickly if desired. Similarly, rotation about the vertical axis is performed by clicking and dragging horizontally and again the speed of rotation is proportion to the cube of distance dragged. Furthermore there is alternate movement style that where movement and orientation can be controlled separately (Figure 5).

Figure 4. The standard movement control.

Figure 5. Separated movement controls.

3.2. Saved views

The user can save a list of camera locations and create a list of animated camera paths or `fly-throughs' with the same tool. Animated camera paths are created by selecting a series of `keyframes', that is, camera positions and orientations. The animation proceeds by moving the camera to each keyframe in sequence, and then interpolating positions and orientations between the keyframes as it goes. A saved camera location is just a degenerate animation, that is, an animation with just a single keyframe.

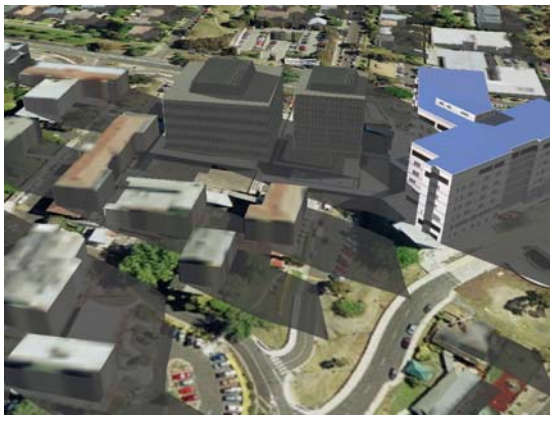

Figure 6. Real-time shadows.

3.3. Shadows

Shadows are an important consideration for visual impact assessment. We used an established algorithm to compute the solar vector for any point on earth (Blanco-Muriel, 2001). The algorithm takes time, date, latitude and longitude to compute accurate azimuth and elevation angles. Latitude and longitude are fixed to the centre of the location of the model and then the user interface allows specific times and date to be entered and controls for animating the sun either through an entire specified day, or an entire specified year (Figure 6).

3.4. Measurement

Measurement of distance, slope and area are important tools for analysis. Measurement of distance and slope is performed by selecting a pair of points in the model with the mouse and calculating the distance and gradient in 3 dimensional space. Measurement can also be constrained to the vertical axis, for measurements of height, and constrained to the horizontal plane, for measurements in plan view. Areas can also be measured by drawing a simple polygon (through selecting vertices with the mouse). All areas are constrained to the horizontal plane.

3.5. Viewshed

A 'viewshed' is the set of regions that are visible to the human eye from a particular vantage point. Viewsheds are particularly important for visual impact assessment in urban planning. The viewshed tool allows generation of the 3 dimensional viewshed from any point in the model, and also allows the vantage

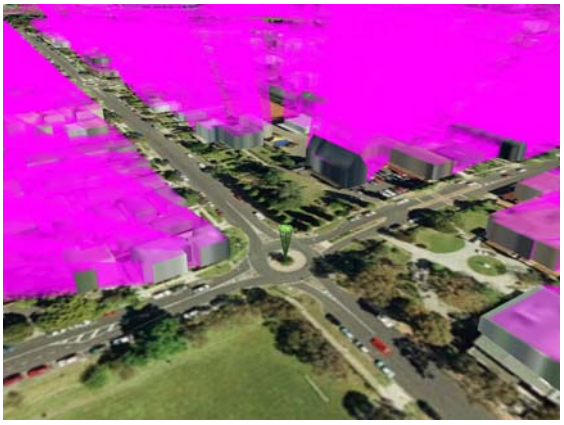

Figure 7. The viewshed analysis tool.

point to be dynamically moved through the model using the mouse. The generated viewshed is displayed by obscuring the parts of the model that can not be seen from the vantage point with a pink colour (Figure 7).

Importantly, since a viewshed is in reality is a 3 dimensional volume, projections to a 2 dimensional plane may result in some loss of information. For example, if the viewshed was performed next to an overpass, it would appear from a plan view projection that all views were obscured in the direction of the overpass, however only a small volume would be obscured and in reality there would still be views below and above the overpass.

3.6. Import

Models need to reflect changes over time and be able to visualise possible future outcomes. Therefore it is necessary to allow the users to import new objects. Objects can be imported in 3 ways.

(i) A general object can be imported in Autodesk 3DS format. The object is positioned manually using the mouse.

(ii) There is a library of standard objects that can be imported and placed in a similar way. The Viewer can be configured so that the library can contain whatever objects the user desires. For example, tree of a specific species or street furniture.

(iii) Objects can be imported by copying from an internal library, and positioning and scaling the copied object using an ASCII text file that contains a dataset of position and scale vectors. This third option is particularly well suited for large numbers of similar objects such as trees and vegetation, although it could be used to arrange any type of object. It is important to note that when using the ASCII text file import feature the user need only specify two coordinates for position. The third coordinate is determined by the height of the terrain at that point. Thus the ASCII text file import feature serves as a way for the user to convert two dimensional maps to three dimensional ones. This kind of dataset can be maintained in a GIS and so this feature allows visualisation of some GIS data.

3.7. Texture editing

The model can be updated by allowing the user to change the textures, that is, raster images, on 3D objects. This is particularly important in urban settings which have a large amount of textures, particularly photographs of buildings on the 3D models representing them. On invoking the tool the user can select an image by clicking on it, and then automatically open the image editing package associated with that file type. The user can perform manipulations such as changes to the scale and orientation in the editing package, and then save the file. The texture is updated in the model on the fly when the update button is pressed.

3.8. Transformation

Objects that have been imported can be transformed in simple ways. Any imported object can be moved, rotated in the vertical axis, or scaled. Transformation is performed by selecting the object with the mouse and then clicking and dragging.

3.9. Layers

One of the most important requirements of a visualisation system for visual impact is to compare scenarios. For example, the ability compare the relative impact of two proposed buildings is paramount for urban planners. Another important requirement is to show and hide detail or simply show and hide information such as vegetation. We developed a tool that allows the user to create, and show and hide layers. A layer in this context is a set of 3D objects. The user creates a layer by opening the layer dialogue and entering a name for the layer, and then selecting an object and adding it to the layer. A layer is also automatically created when objects are imported by an ASCII dataset.

3.10. Saving and concept of a file

The user has the option of saving files and reloading them at a later date. A file in this context is the set of all current layers including the objects that reside within those layers. Therefore a "blank" file in this context is the static parts of the model. This is the terrain and possibly existing static built form. Files are saved in the Virtools NMO format.

3.11. Video and screen capture

Video and screen capture allow the user to take full advantage of the real-time render engine. With these capabilities the model becomes an infinite source of images and video. Both still images can be captured as BMP files and videos can be recorded as AVI files on the fly.

4. DISCUSSION AND CONCLUSION

As demonstrated by our increasing client base, our tool is affordable and works well to serve the needs of large urban councils. We were motivated to use available software to produce effective and affordable software with minimal staff time requirements and in as short a time as possible. A strength of our approach is that we utilised existing software in a novel and user-driven way.

Future development will focus in several areas. Our first priority is making all objects, except the terrain, editable. The goal is to make an application that allows the user to fully maintain and edit the content. The fact that existing buildings cannot currently be removed easily by the user is perhaps that biggest limitation. Next, further work needs to be done in interoperability with GIS databases. Most users of this technology already have large amounts of information in a GIS that they would like to be able to import and overlay with the existing model. A further area that many have expressed interest, particularly for urban environments, is animated traffic and crowds. Many sophisticated software packages exist for modelling traffic and crowds (Caliper, 2007; Quadstone, 2008), and these usually contain their own real-time 3D visualisation tool, but are usually only very limited in their functionality and quality. An import feature for traffic and crowds would greatly enhance the capabilities for both pieces of software.

ACKNOWLEDGMENTS

The author would like to acknowledge Banyule City Council for their extensive feedback and assisting in formulating the requirements and specifications of the software.

REFERENCES

Blanco-Muriel., M., Alarcon-Padilla., D.C., Lopez-Moratalla., T., Lara-Coira., M. (2001), Computing the solar vector. *Solar Energy*, 70(5), 431-441.

Caliper (2007), TransModeler. Newton MA, USA

Dassault Systemes (2006), Virtools 4. Paris, France

Stock, C., Bishop, I.D., O'Connor, A.N., Chen, T., Pettit, C.J. and Aurambout, J-P. (2008), SIEVE: Collaborative decision-making in an immersive online environment. *Cartography and Geographic Information Science*, 35, 133-144.

Quadstone (2008), Paramics. Edinburgh, Scotland.