Visualising an Amusement Park – A Case Study

Martin Naef, Douglas Pritchard, Chris McMahon, Paul Anderson

Digital Design Studio, The Glasgow School of Art

ABSTRACT

This paper presents a real-time visualisation of an early architectural design for an amusement park. This commercial project, aimed at generating political support and investor interest for the proposed development, was implemented to strict budgets and deadlines and provides the case study to analyse the limitations of current content creation tools and formulate requirements for further research and development.

KEYWORDS: Real-time architectural visualisation.

INDEX TERMS: I.3.8 [Computer Graphics] Applications

1 INTRODUCTION

High quality off-line renderings have long been the standard and only way to visualise architectural projects. Commercial modelling and rendering software today empowers even smaller design companies to produce photorealistic images and stunning videos to present their designs in the best light. However, such glossy pictures are often received with scepticism and the desire to see different, potentially less flattering aspects or viewpoints. With turnaround times of between hours and days for high quality renderings and videos, it is not possible to react spontaneously to audience requests and present new image material during presentations. Real-time visualisation tools provide the means to fill in this gap and to impress the audience with dynamic and spontaneous presentations.

The Digital Design Studio of the Glasgow School of Art has created real-time presentations of a wide range of architectural design and urban planning projects for commercial partners. This paper presents the lessons learned from visualising an early design of a large amusement park with the aim of generating political support and investor interest. The particular case study represents well the requirements for this type of application and the limitations of current, off-the-shelf content creation and display tools.

The paper focuses on the challenges and trade-offs required to meet a range of requirements while operating within strict time and budget limitations. The case study is not representative of the current state of the art in research; instead it aims to provide an understanding of the requirements and scene complexity of commercial applications as opposed to isolated aspects typically addressed by research projects.

2 VISUALISATION REQUIREMENTS

The architectural design of the amusement park was in its early stages at the beginning of the project. The visualisation team received a series of hand drawings from the architect representing the top-down view of the site including its immediate surroundings and a rough CAD model including building outlines. The development covers an area of roughly 1 km square, which is typical for many similar projects, although the team has previously handled real-time urban visualisations for larger regions of approximately 5 by 3 km. The amusement park is a parkland setting including green zones, transportation, pedestrian areas, large artificial ponds and a range of buildings representing different geographic regions and industries.

The modelling team was tasked to visually enhance a basic block model provided by the architect to move away from the abstract look towards a "realistic" experience. This required a degree of guess-work and artistic freedom as the details of the architecture had not been specified at that point. This was a rather unusual project brief, as it required the modellers to make decisions usually left to the architect about the look of buildings. At the same time, it allowed the modellers to take a "game development" approach where design decisions were influenced by the polygon budget. Normally, the modellers are presented with fully specified CAD models that may need optimisation for real-time display. A rendering of the resulting model is shown in Figure 1.

Figure 1. Offline-rendering of the augmented model.

2.1 Presenting the Model

Presentations using a real-time environment typically consist of a rehearsed and closely scripted introduction, followed by an interactive Q&A session where the visualisation is used to answer audience questions.

For the introduction section, a set of viewpoints are defined in advance. With a key press, the application smoothly flies to the next view to present individual aspects of the design. Animation paths can be defined as well, enabling automatic fly-through. While the audience's experience is similar to showing pre-rendered animations, the real-time system enables the presenter to change and optimise viewpoints until the last minute – an all too common request frowned upon by traditional off-line visualisation providers where turn-around times of several hours or days per animation sequence are common.

During the interactive session, the presenter typically refers back to preset viewpoints and then navigates to the desired view using a SpacePilot 3D joystick. Flying or hovering using this six degree of freedom input device requires some initial training for the operator, but is intuitive enough for a non-expert user to quickly learn how to use it efficiently and smoothly.

[m.naef | d.pritchard | p.anderson]@gsa.ac.uk

2.2 The Impact of Viewpoints

The technical and modelling requirements of a real-time visualisation are highly dependent on the expected viewpoints. A pedestrian's point of view provides a good impression of how a development will look for the general public. The pedestrian's point of view imposes heavy requirements on the modelling: Objects nearby must be displayed with a lot of detail to render a realistic impression, resulting in very polygon- or texture-intensive scene data that easily exceeds rendering as well as memory capacity of the image generator hardware if a larger area has to be covered. At the same time, though, specialised visibility culling algorithms [1] tend to be very effective at reducing graphics hardware bandwidth as buildings nearby cover large portions of the scene.

A bird's-eye viewpoint, on the other hand, is preferred for judging how a building fits into an existing context or, in the case of the amusement park, how a larger development is structured. This perspective is less demanding on the modelling detail simply due to the larger distance, but it also renders most visibility culling techniques ineffective.

Many common applications such as flight- or driving simulators as well as games use predominantly one type of viewpoint, enabling the application designer to optimise the model for the particular application. The visualisation of urban or architectural developments, on the other hand, typically requires switching between both types of viewpoint. This was clearly the case with the amusement park model where a strong focus was put on top views to display the spatial concept while still being able to convey a visitor's impression. Other visualisation projects, such as the visualisation of six design alternatives for an urban regeneration project within the City of Glasgow, imposed very similar requirements.

Figure 2. Top view of the amusement park with 1 million triangles.

2.3 Level of Detail

Modelling the environment has to strike a balance between enabling high-quality street level views while keeping the polygon count low enough to enable smooth top views. Ideally, one would start with a highly detailed model from which automatic levels of detail are automatically generated. In practice, however, automatic polygon reduction often results in rather awkward looking buildings. Manually creating enough LOD models to enable smooth switching is rarely realistic within the given project budgets.

In the case of the amusement park visualisation, there was no level of detail switching implemented for any of the architectural elements. Instead, the original block model was refined within a given polygon budget. For other projects based on a city environment, a low-resolution "blue block" model was available for general overviews (approximately 350,000 triangles), and textured high-resolution models of selected individual buildings could be switched on manually where required.

Given the huge difference between the low-resolution base model and the high quality textured version, manual selection of the LOD was considered preferable. Instead of surprising the audience with sudden and potentially erratic changes, changing to a high-resolution view was made part of the presentation narrative.

Nonetheless, it is our hope that future modelling tools will introduce automatic level of detail generation suitable for architectural and urban visualisation. In particular, new modelling paradigms such as grammar-based hierarchies [2] have the potential of offering progressive refinement without introducing disturbing artefacts.

3 Content Creation Tool Chain

The visualisation project uses a range of commercially available software tools to implement the full tool chain from modelling to the real-time presentation environment. Custom software development is limited to the last element, the real-time viewing environment, which includes automatic generation or instantiation of scene elements and animation and all interaction.

All architectural models are built using an industry standard animation/modelling package, 3D Studio Max. The modelling tools are mature and well understood.

As none of the real-time rendering systems read 3D Studio data files natively, all models must be converted into a portable data format, in this case OpenFlight (FLT). Data conversion using a tool such as Nugraf's Polytrans is relatively straightforward as long as the initial modelling takes the limitations of the target format and converter into account, namely the restriction to the basic Phong shading model with a single texture layer and polygonal mesh modelling. Nonetheless, conversion remains quirky in practice, requiring a significant amount of "cleaning up" and effectively rendering the conversion process a one-way street. Despite several attempts from both industry and academia, there are no truly portable data formats and tools available that allow going back and forth between real-time viewers and modelling applications.

Multigen-Paradigm's Creator application takes a central role in the content creation process. Creator is used to clean up the OpenFlight data files after export and define the final object hierarchy including level of detail switches and transformation/ animation nodes. It is also used to tweak material parameters, textures and assign CG shaders to parts of the geometry.

Additional tools used for content creation include Bionatic's RealNat vegetation modelling system and Right Hemisphere's Deep Exploration software used to simplify polygonal models.

3.1 Real-time Environment

Once the models and environment items are converted into a format suitable for real-time display (e.g. OpenFlight) they must be loaded and assembled into a scene graph within a real-time display environment. There are a range of toolkits available, both commercial (e.g. VEGA Prime, Quest 3D, EON, etc.) and open-source (OpenSceneGraph, OpenSG, VRJuggler, etc.). While the freely available scene graph and VR application toolkits offer excellent rendering performance and a rich feature set, they lack the rapid development environments of their commercial counterparts. The time saved by defining complex object hierarchies and parameters through an intuitive GUI before any code is written can quickly justify the seemingly high licence cost.

The visualisation of the amusement park is based on the Multigen-Paradigm VEGA Prime software development environment. VEGA Prime has a strong following within the simulator market segment thanks to its capability of handling large environments and terrain. It is also well suited for large area urban visualisation projects. Application development is based on

standard C++ code and libraries, with an additional tool (Lynx) to define and test the object hierarchy and special effects.

3.2 Issues

The content creation tool chain has one major weakness that should be addressed: It is essentially a one-way street. Each stage in the modelling process introduces certain node types and optimisations that are not transferable back to the original modelling environment. If errors in the model are detected late, or if changes in the original model are requested, a large portion of the model optimisation and tweaking stage has to be repeated manually. Any late change request therefore has significant cost and time implications, and the model must be carefully reviewed after every conversion step. Breaking the model into more manageable sections reduces that cost, but the inevitable errors that occur still result in relatively high conversion times that are easily underestimated during initial project planning.

4 Populating the Environment

The basic environment model including the ground plane, all buildings and static cars consists of approximately 200k triangles. This is easily handled by modern graphics hardware. While such a model is effective at presenting the overall structure, it is insufficient to convey the excitement that should go with a theme park. A range of animated objects and special effects were therefore added. Most of these objects are not part of the original 3DSMax model, instead they are either added programmatically (props, vegetation, crowds and shadows) or within the real-time modelling tool, Creator (shaders, texture layers).

4.1 Props

The basic architectural design was populated with a large amount of small props to add life and excitement. In particular, animated items attract a lot of attention. A significant number of static props were added during the modelling stage. This includes parked cars, benches, tents, lamps, etc. Some elements of the amusement park, particularly the rides and carousels, were included within the static model and then animated manually. A lot of life is added through cars that drive around the outer ring road and a number of rocking sailboats within a pond.

The impact of props on the total geometry count is easily underestimated. Traffic is very light in this simulation, with only 30 cars on the road. However, each car model (labelled as "real time models" by the library vendor) includes between 20,000 and 30,000 triangles, adding up to a million triangles for the cars alone in a top view. Fortunately, automatic LOD generation using Deep Exploration worked well for the cars, bringing the total polygon count down to a reasonable level. Nonetheless, a busy city scene could easily include several hundred cars.

4.2 Vegetation

Realistic trees and bushes are a key element for any parkland scene, and also often feature prominently in urban visualisations. Fortunately, there are a range of commercial tools and libraries available to grow virtual plants suitable to real-time environments.

The amusement park model is populated with 490 instances of 13 different tree models generated with Bionatic's RealNat Premium software. Each model includes five levels of detail, from a high-polygon model (up to 1,500 triangles) down to a basic textured billboard cross. Thanks to the efficient LOD generation, the vegetation does not overly stretch the polygon count; however each tree model comes with between 4 to 8 MB of texture data, which effectively limits the number of vegetation models to be used simultaneously. To avoid an overly uniform look of the trees, all instances are stretched and rotated randomly; a casual observer barely notices the very small number of different vegetation models. Hedges are modelled as simple polygonal boxes with an RGBA texture.

Texture filtering becomes an issue when rendering vegetation. As the scene graph does not provide depth-sorting at primitive level, semi-transparent vegetation results in ugly halo effects. Short of implementing custom primitive sorting, the only available solution is to use nearest neighbour texture interpolation therefore avoiding semi-transparent alpha values. For static viewpoints, the resulting aliasing is not disturbing as plants are inherently noisy, but it becomes obvious when moving. Fortunately, enabling the multi-sampling feature of the graphics hardware reduces aliasing to an acceptable level.

4.3 Crowds

Any urban scene looks artificial and lifeless without animated humans, regardless of modelling quality and special effects. Modelling and animating crowds has been an active area of research for several years with impressive results. Unfortunately, crowd simulation is not yet available as a standard feature within the tools used here. Due to time und budget constraints, only a simplistic crowd animation model could be developed: A set of paths (polyline with a width attribute) is defined along which humans move. At the start, humans are scattered along the path with a random side offset and walking speed. At each frame, the position is moved along the path with the given speed. Random speed changes are introduced to break up visible patterns. No further strategies (e.g. collision detection, crowd behaviour, etc.) are implemented. Each human is represented as a billboard rotated around the Z-axis to face the camera. A range of textures is used to generate diversity.

Despite the simplistic nature of the animation system, the results are actually quite impressive especially when viewed from a higher perspective. Having thousands of people moving within the park adds a very lifelike quality to an otherwise empty environment. Even from a pedestrian's viewpoint, where the artificial nature of the crowd becomes obvious, they add a feeling for the scale of the environment.

Figure 3. A busy bazaar area with hundreds of people.

Crowd simulation comes at a cost. Populating the whole 1 km square area with a reasonable amount of people requires the simulation of approximately 4,000 humans. Within the application, updating their position and rotating the billboards is the single most expensive operation outside the rendering process and takes 3.5 ms on a Xeon 3.6 GHz processor. While some of that overhead (rotation and interpolation) could be moved into a vertex shader, it is nonetheless an indication of how many individual objects can be animated before the frame rate suffers.

A full agent-based simulation for each individual human within this large area and population density is probably not within reach today (e.g. [3] cites 7.5ms per frame to simulate 500 agents without animation), but it is also not required for this type of application. Instead, a level of detail approach could be taken

where only humans nearby are simulated at high fidelity while the rest follows simplistic paths as described above. A focus must be set on making such technology easy and fast to use within a commercial system, as it is unlikely that a client would budget for a significant amount of scripting effort.

4.4 Shadows

Shadows are arguably the single most important visual effect for architectural visualisation. They greatly support the understanding of the spatial relationship of objects. Computer graphics literature lists a large range of shadow rendering algorithms, yet commercial real-time rendering systems often only support rudimentary shadows.

The size of the environment causes some difficulties for shadow simulation. Algorithms based on the projection of the geometry onto the ground plane, as provided by the toolkit, place a significant burden per frame onto the rendering pipeline. They also fail to simulate shadows cast onto buildings or props nearby. Texture-based algorithms such as depth maps are not supported by the scene graph used here. In any case, they would require very large texture maps to provide a sufficient resolution, taking a significant toll on texture memory and rendering performance. Calculating shadows offline and storing them as textures ("baking") results in very large texture sizes as well and precludes time of day simulation that was considered important for the visualisation of the amusement park.

Acknowledging the importance of shadows while not having the required tools available led to the implementation of a low-fidelity solution. Shadows are created by rendering a single dark quad primitive with a semi-transparent alpha texture, creating a smooth impression of a shadow underneath each object. The corners of the quad represent the convex hull of a box around the shadow caster projected onto the ground plane. Different shadow textures are used to at least provide a hint of the original shape of the associated object. Despite the gross simplification, these fake shadows provide enough cues to avoid the effect of hovering objects with very little computational overhead. The shadows are particularly successful at providing more definition to the billboard humans that would otherwise almost disappear from a bird's perspective and to avoid the impression of trees hovering above the grass. The simulation includes 605 shadows for general objects and 4,000 shadows attached to humans.

4.5 Shaders and Special Effects

Shaders and special effects were used sparingly. The water surface of the pond includes animated bump maps read at two different scales for the simulation of waves, a sky texture for reflections and a granite slab texture whose coordinates are perturbed to simulate refraction. Grass areas include a large base colour texture with two layers of noise added depending on the distance, avoiding visible tiling artefacts. A vertex shader is used to animate the flags. Particle systems are used to simulate fountains. The effects were chosen with a view to generating maximum visual impact while keeping development time low and staying true to the overall visual style.

Additional vertex shaders were considered to move crowd animation onto the GPU. However, there would be no positive impact on the total frame rate as animation is currently calculated outside the rendering thread. The frame rate is limited by the rendering thread (typically between 30 and 50ms), all other calculations including culling, shadow calculations, animations etc. fit within a 10 to 15 ms time slot in the application thread. In a properly multi-threaded environment, there is nothing to be gained by moving animation code onto the GPU unless such a step would also reduce memory bandwidth to the graphics hardware, or animation would be too complex to fit within the time slot on its dedicated CPU core.

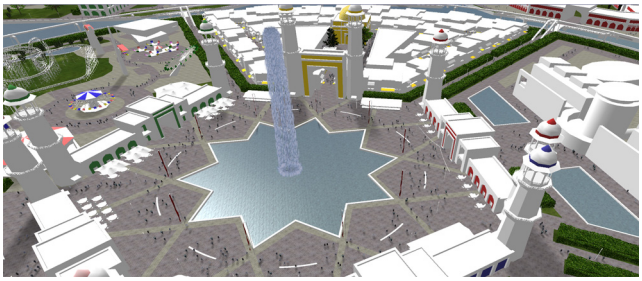

Figure 4. Special effects in the central area: Particle fountain, shadows, water surface and animated flags.

4.6 Scene Complexity

In total, the amusement park scene complexity adds up to about 1.8 million triangles, including 200k for the base environment, 900k for the cars, 7.5k for the boats, 16k for human billboards, and 600k for trees. LOD switching reduces that number to roughly one million triangles per frame when rendering a top view, resulting in approximately 20 frames per second on a Geforce 8800GTX graphics board at 1080p HD resolution. In this case study, the props dominate the scene complexity, but they are an essential part of the presentation narrative. On the other hand, building detail typically dominates scene complexity in visualisation projects within an urban context, where the assessment of the visual impact within a given environment is more important than conveying the excitement of an amusement park.

5 Conclusions

Developing the visualisation of the amusement park highlighted several shortcomings of the content creation pipeline, namely the lack of portable data formats to enable a two-way workflow. Given the importance of introducing life to the environment, crowd simulation should be considered as a standard feature for real-time visualisation tools in the future, whereas vegetation is reasonably well covered today. Similarly, implementing efficient shadow algorithms should be a priority.

Despite the ever increasing power of GPUs, scene complexity for urban and architectural models of this size will easily exceed the capacity of current and the next few hardware generations unless clever data reduction techniques are applied. Given the importance of top-views where visibility culling becomes less efficient, we would particularly hope for efficient level of detail generation tools suitable for architectural models.

Despite such deficiencies, real-time visualisation of urban models for city planning and architectural project presentation or critique is a commercial reality today that has been received enthusiastically by a range of clients.

References

[1] Peter Wonka, Michael Wimmer, Dieter Schmalstieg. Visibility Preprocessing with Occluder Fusion for Urban Walkthroughs *Proceedings of the Eurographics Workshop on Rendering 2000*. pp 71-82. June 2000.

[2] Pascal Mueller, Peter Wonka, Simon Haegler, Andreas Ulmer and Luc Van Gool. Procedural Modeling of Buildings. *Proceedings of ACM SIGGRAPH 2006 / ACM Transactions on Graphics.* pp 614-623, Vol. 25, No. 3, ACM Press, August 2006.

[3] Mankyu Sung, Michael Gleicher and Stephen Chenney. Scalable Behaviors for Crowd Simulation, *Computer Graphics Forum 23, 3(2004)* (Eurographics '04).