Sketch, Click, Plug and Play: Accelerated Design of Virtual Environments by Integrating Multimedia and Sketch Content into Game Engines

Burkhard Wünsche
Department of Computer Science
University of Auckland
Private Bag 92019, Auckland, New Zealand
burkhard@cs.auckland.ac.nz

Daniel Keymer
Department of Computer Science
University of Auckland
Private Bag 92019, Auckland, New Zealand
dkey012@aucklanduni.ac.nz

Robert Amor
Department of Computer Science
University of Auckland
Private Bag 92019, Auckland, New Zealand
trebor@cs.auckland.ac.nz

ABSTRACT

Virtual Environments are becoming increasingly popular outside the area of entertainment and are now used for a diverse range of applications such as education, training, simulation, urban design, architecture and archaeology. One of the main challenges of using virtual environments is the high cost of creating content. While there is a large variety of modelling tools available, they all require training, usually have a steep learning curve, and even simple structures can take hours or days to model depending on the required level of detail and the user's experience. In order to make virtual environments more accessible to a wider group of users the content creation step needs to be simplified.

In this paper we present a framework for effectively and efficiently creating virtual environments by integrating different media such as high resolution images and QuickTime VR movies into game engines. Terrains and simple 3D content are modelled using sketch and paint-like interfaces. The QuickTime VR movies are "fused" into the game engine environment providing a context-and-focus view of different aspects of the scene.

We demonstrate the usefulness of our approach by creating an archaeological virtual environment. The representation is much easier to create than a detailed fully 3D environment. However, it provides multiple advantages over traditional media for study and exploration, e.g., collaboration, overview and detail views, and improved perception of spatial and temporal relationships which is essential for understanding the usage of an archaeological site.

Preliminary user studies indicate that the tool considerable facilitates the creation of virtual environments. Compared to traditional 3D worlds our environments are simpler, but due to the integration of existing multimedia content they provide a similar amount of information for exploring and understanding the simulated scene. The QuickTime VR integration is virtually seamless and together with the context provided by the 3D environment allows similar types of navigation as for environments modelled entirely in 3D.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]: Artificial, augmented, and virtual realities; H.5.2 [**User Interfaces**]: Prototyping; I.3.5 [**Computational Geometry and Object Modeling Software**]: Modeling packages

Keywords

virtual environments, QuicktimeVR, sketch-based modelling, rapid prototyping, game engines

1. INTRODUCTION

Virtual environments (VE) were initially motivated by military applications with flight simulators being one of the most complex applications. Since then the use of virtual environments has grown exponentially and they have become an essential part in the entertainment domain (e.g., computer games). Over the past decade usage has expanded into virtually every application field including education and training, e-commerce, engineering, scientific and biomedical visualisation, social networking, urban design and landscape planning, visual impact studies, architecture and archaeology [7, 8, 13, 16, 1, 22]. The dramatic improvement in graphics hardware has made it possible to use virtual environments on consumer level PCs. However, the creation of virtual environments is still time consuming and usually performed by computer experts and/or graphic artists.

A large variety of tools exist to support programmers and artists. For general applications that includes modelling and animation software such Maya, 3D Max or Blender. New game worlds can be constructed using level editors and scripting support for modifying and extending existing games. These tools are also suitable for non-game applications, such as creating virtual medical training simulations. However, content creation is time consuming and involves a steep learning curve [24]. For several application domains special tools have been developed to simplify content creation, e.g., for urban design procedural modelling has been used successfully [37]. Content creation for virtual worlds is now a big business with 3D models and virtual characters available for purchase on the Internet and in virtual environments such online games and "Second Life" [22].

An important concept for simplifying content creation is *Selective Fidelity*: only components essential for the use and understanding of the virtual environment must be recreated in high fidelity whereas other components can be approximated or eliminated entirely. Selective fidelity has been shown to dramatically reduce costs while maintaining a simulator's effectiveness [16]. Similarly the amount of interactivity and narration can be adjusted according to the application. However, in order to maintain immersion and functionality of a virtual environment it is important to provide context and to enable the user to navigate reasonably freely within the environment. Anders presents VEs as a space for perception and cognition which spans physical presence, perceivable representation and cognitive presence [2].

In many applications the usage of the virtual environment involves mostly exploration and navigation tasks and limited interaction with 3D models. Typical examples are archaeology, visual impact studies, urban design and architecture. In such cases simplified non-physical representations of objects are possible. Furthermore for many of these applications there exist already a large amount of content in the form of two-dimensional media such as images, videos, GIS data and concept drawings. However, a simple collection of linked images and movies is not suitable for creating virtual environments because it limits navigation and perception as discussed in the above paragraphs. Usability can be increased by "fusing" images together as demonstrated in the Photosynth project [32, 33, 26]. The resulting representation allows navigation through the simulated world and provides context in the form of surrounding images, but does not create the feeling of being in a 3D environment and provides no opportunities for users to collaborate.

While multimedia content is information rich, it does not have 3D properties and is non-trivial to create in cases where no physical representation exists (e.g., a building which was destroyed or is yet to be build). In such cases an input tool is necessary for rapid prototyping of 3D structures. Sketch-based modelling has been shown to be a suitable interface because the underlying pen-and-paper metaphor is intuitive and effective.

In this paper we present a framework for effectively and efficiently creating virtual environments by integrating different media such as high resolution images and QuickTime VR movies into a game engine. The QuickTime VR movies are directly fused into the game engine environment providing a context-and-focus view of different aspects of a site. 3D content is created using various paint and sketch-like interfaces. The tool is suitable for applications simulating real sites where a large collection of media exists already or can be easily created by the user.

Section 2 discusses technology choices. Section 3 presents the design of our virtual environment rendering framework. Section 4 discusses how 3D content can be created using sketch-based interfaces and section 5 provides implementation details. Section 6 demonstrates the use of our framework for the design of an archaeological virtual environment and section 7 summarises our results. We conclude the paper with section 8 and provide suggestions for future research.

2. TECHNOLOGY SELECTION

The goal of our research is to provide a tool for efficiently creating and exploring complex virtual environments. We analysed a wide variety of 3D visualisation technologies [18] and found that game engines fulfilled our requirements best since they are cheap, are optimised for consumer level hardware, are frequently updated to use the latest graphics technologies, provide intuitive interaction tools, and have a large user base resulting in well tested and stable code. We eventually settled on Esperient Creator [11], as it is highly flexible, has excellent tools, strong plug-in support, multiplayer support (for collaborative visualizations), has a free viewer, is targeted at the development of non-game virtual environments such as those we wish to support, and it is used in related projects by our collaborators and us.

Most game engines, including Esperient Creator, support the integration of images and videos into virtual environments via static or animated billboards and textures. However, images and videos do not provide contextual information and are hence of limited use for exploring and understanding a virtual environment. In order to create a three-dimensional view of complex objects and scenes without requiring modelling them we decided to integrate QuickTime VR movies.

QuickTime VR panoramas can be easily created from traditional camera images using image stitching, or by using specialised panoramic cameras. A wide variety of authoring tools are available whose usage does not require any special expertise in modelling or photography [5].

3. DESIGN

3.1 Virtual Environment Rendering Framework

The virtual environment rendering framework combines an Esperient scene and a C++ plug-in, which allows the use of the QuickTime VR API and external scene definition files (figure 1). The data model for the virtual environment is kept within the plug-in including some data representing the program state. This makes the plug-in code less engine-specific and permits the choice of more efficient algorithms for manipulating data. It also makes it much simpler for external systems, such as databases, to manipulate or access the scene data, and to support remote data sources. Finally, having program state data available within the plug-in itself makes determination of this state for the purposes of program control much simpler and more efficient. The state of the Esperient scene itself must "shadow" our own data representation so that the renderings are consistent with the virtual environment view within our framework.

The virtual environment scene file for the rendering framework contains material such as terrain and buildings, templates for panoramas, user interface elements, and various animations (which are generally activated from the C++ plug-in). Certain objects, such as panorama templates, link to external scene files within our framework and are run from there. Interactions within the virtual environment must be consistent throughout the framework as explained in subsection 5.1.

The C++ plug-in has been structured into a model layer and an integration layer. The *model layer* contains the data forming the abstract representation and state of the virtual environment (see subsection 3.2). It handles the loading of the site from XML files, and the majority of the interaction with the QuickTime API. The model layer contains no code specific to Esperient Creator; it is intended to be agnostic of the game engine. This improves portability as the model layer can be used as-is with any engine, and maintained separately.

The *integration layer* handles the interaction between the model layer and Esperient Creator and improves portability to other game engines. This requires it to serve a role in both directions. Firstly, when the model layer changes, it must translate those changes into the Esperient scene. Secondly, when an event occurs within the Esperient engine that may change the state of the abstract virtual environment, the integration layer must process the event and notify the model layer of the change. An example is given in figure 2. Any engine-dependent aspects of a feature of the virtual environment system are defined in the integration layer, e.g., procedures used to add objects to a scene, or to determine the position and orientation of the user's viewpoint within the scene.

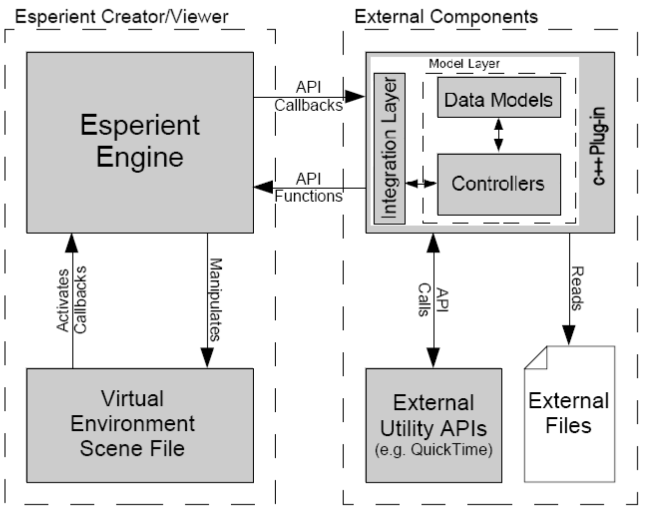


Figure 1: The structure of the virtual environment system. The only new software required is our C++ plug-in integrating the different components.

3.2 Virtual Environment Data Model

In order to enable dynamic scene generation, efficient data access and manipulation, and interaction with other systems or APIs we developed a custom data model. The data is imported from an XML scene descriptor file which can be modified to reflect different types of virtual environments. Our prototype has been developed for visualizing archaeological environments and contains descriptors for terrains, large scale wall structures, buildings, seas and rivers, panoramas, game objects and era objects (representing time frames over for the existence of other objects). Scene components can be created by importing existing models or creating new models such as terrains and city walls using efficient and easy-to-use sketch and paint-like interfaces. Details are given in [18].

3.3 QuickTime VR Integration

In order to make QuickTime VR movies a natural part of a virtual environment the following requirements must be fulfilled: the implementation must make use of the resolution and detail present in the original media; it has to allow immersive navigation between panoramas using a first-person view; both panoramas and object movies ought to be simple to find and use; and the behaviour of panoramas should mirror the behaviour of the virtual environment as a whole as closely as possible.

We conceived of three possible solutions. The first is simply to launch a separate QuickTime player to handle the display of the panoramas or object movies. This method, while simple and easy to implement, addresses some of the requirements quite poorly. For instance, it cannot be considered particularly seamless if the user must manually close QuickTime and switch back to the virtual environment after viewing media. Furthermore, if the user starts a panorama, turns the view by 180 degrees, and then returns to the virtual environment, the user will be looking in a completely different direction. This can cause disorientation and impedes the intention of the system to provide the user with a context when travelling between successive panoramas on the site as it introduces a disconnection between the view in the panoramas and the view in the virtual environment.

The second solution is to extract the image data from the QuickTime VR panoramas for use within the Esperient Creator engine.

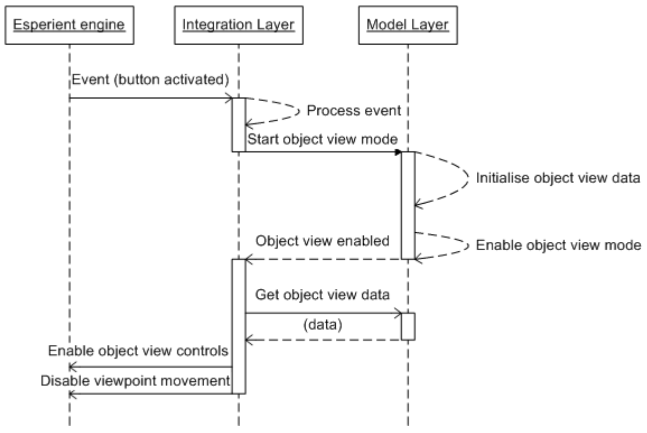


Figure 2: A sequence diagram illustrating how the model layer is updated to reflect events within the Esperient Creator scene. When an event occurs within the Esperient engine, e.g., viewing an object represented by a QuickTime Object movie, then the integration layer must determine what kind of event took place and how it will affect the model layer. In this case an object view is started, so the integration layer informs the model layer of this event. The model layer proceeds to initialise state data for the object view and passes control back to the integration layer, which makes the appropriate changes to the Esperient scene.

Panoramas within QuickTime VR are in fact implemented by mapping images onto the inside of primitive 3D objects; one image onto a cylinder for a single-row (horizontal rotation only) panorama, or six images onto a cube for a cubic (horizontal and vertical rotation) panorama [3]. It is possible to extract this data and reproduce the process using the game engine's rendering component. However, the solution is unnatural since it requires artificial scene components which are invisible to other users.

We hence decided to display a built-in QuickTime VR "window" within the virtual environment itself. Although this requires complicated interactions between the Esperient and QuickTime APIs, it allows both the seamless integration with the virtual environment and the use of in-built QuickTime features. Like the second solution, it mimics the behaviour of the virtual environment as a whole, ensures effective navigation and context between panoramas, and

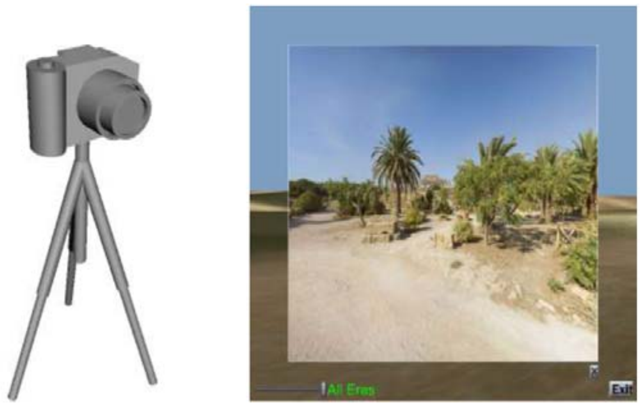


Figure 3: The panorama marker object (left) and the QuickTime VR panorama screen (right).

displays QuickTime media in full detail. However, unlike the second solution it also retains the capacity to make use of new features in future versions of QuickTime.

Several design choices were considered for activating QuickTime VR panoramas while in the virtual environment. An automatic activation is possible, but would be distracting when exploring the scene. We decided to use a tripod-mounted camera as a marker to indicate the presence of a panorama (figure 3, left). When the user clicks this object, a window displaying the panorama is shown (figure 3, right). Clicking and dragging on the screen will rotate the panorama similarly to the interaction with a standalone QuickTime player. Closing the window will return the user to the virtual environment retaining the view orientation from the panorama view.

Several other features were added to improve the system. The QuickTime VR panorama does not fill the entire screen and the virtual environment is shown in the region surrounding the panorama window providing a visual context. The idea was motivated by the "focus + context" technique in Information Visualization which has been shown to significantly improve perception and understanding of visualized data [10]. To ensure the two views match in origin and direction, the view direction within the virtual environment is aligned with the view direction in the panorama. Additionally, when the user first clicks on the camera object in the virtual environment, the viewpoint is gradually changed toward the camera perspective of the QuickTime VR movie before the panorama view is displayed. This prevents "jumps", which are visually disturbing and reduce comprehension of the surrounding environment [31].

4. SKETCH TOOLS FOR 3D CONTENT CREATION

The previous chapter described the integration of multi-media content into the virtual environment. The resulting content is information rich but has no inherent 3D properties and can only be obtained if an actual physical model is present. In order to enrich the virtual environment we have devised a number of sketch-based modelling algorithms. Consistency with the previously described design approach is achieved by requiring that the algorithms are intuitive, do not require artistic or 3D modelling skills, and that they enable rapid prototyping (less than 1 minute for an object). The following subsections describe current prototypes.

4.1 Terrain Modelling

Terrain modelling is achieved by enabling the user to draw a set of contour lines. Contour gradients are indicated by arrows. As long as all contour lines are covered by at least one arrow the relative heights of regions between the contour lines can be determined and used to label the contour lines using a partial order [23]. The labelled contour lines are then converted into a 3D terrain [38]. Since no absolute height values are given we currently use a fixed scale factor in order to achieve a visually pleasing ratio between terrain height and size. An example is shown in figure 4. The current implementation does not yet check terrain gradients for consistency and is best suited for modelling the overall layout of a terrain. We are currently experimenting with adding terrain details automatically by using a multi-fractal model or by sketching terrain cross-sections [12].

4.2 Modelling of Vegetation

An important component of natural environments is vegetation. We have devised a simple method for modelling trees and bushes based on work by Wither et al. [36]. In contrast to the original

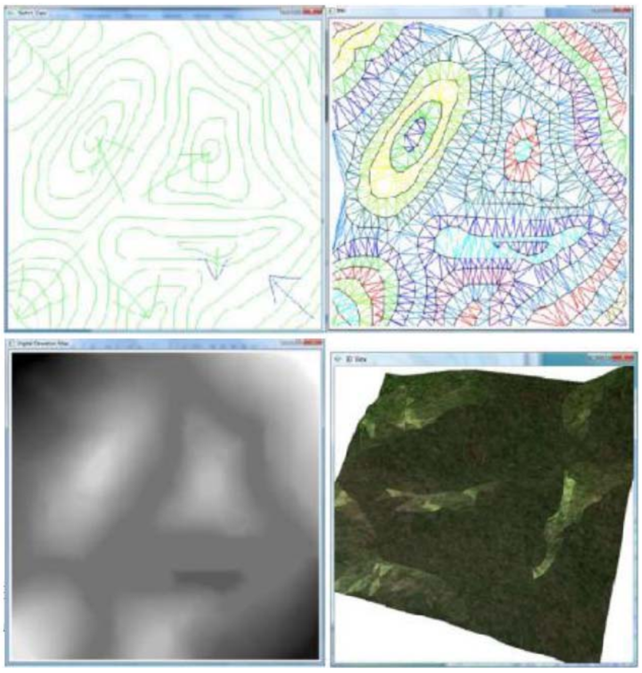


Figure 4: Sketch-based terrain modelling: The user sketches contour lines and arrows indicating the terrain gradient direction (top-left). The contour lines are sampled and triangulated. Relative height information is derived from the terrain gradients (top-right). A digital elevation map (DEM) is created by interpolating for each sample point the vertex heights of the triangle covering it (bottom-left). The resulting DEM can be rendered with any game engine or directly with OpenGL (bottom-right).

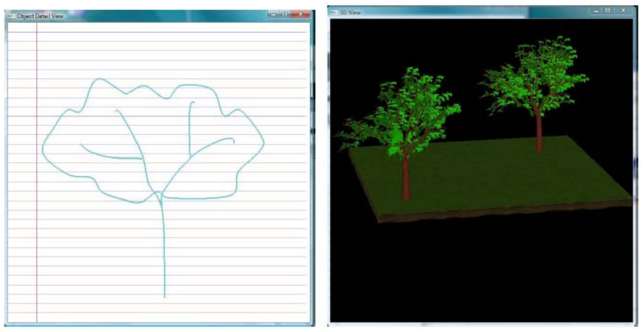


Figure 5: Sketch-based tree modelling: The user draws the tree skeleton (stem and main branches) and the silhouette of the leaf region (left). The algorithm automatically creates smaller branches to fill the silhouette region and adds leaves to them (right).

paper in our case the modelling process needs to be completed in seconds rather than minutes. Furthermore a rough model is sufficient as long as it is correctly perceived as tree, bush etc. We achieve this goal by letting the user draw a tree skeleton using a few strokes and the silhouette of the tree. The skeleton is then expanded to a 3D object with a branch distribution such that its 2D projection resembles the sketched skeleton. Smaller branches for filling the silhouette region are created automatically by mirroring the sketched branch structure. Finally leaves are distributed randomly along those branches. An example is shown in figure 5.

Currently the leaf shape and colour are fixed as indicated in figure 5. We previously implemented techniques for modelling leaves with just a few sketches [25], but in this application this would result in sketch input on two very different scales. Furthermore, designing leaves for different tree shapes would require colour or texture selection. Since in our application speed and intuitiveness is more important than realism we decided to use default values for leaf shape and colour. A possible improvement would be to create a data based with different tree and bush shapes and to decide the leaf shape, distribution and colour automatically by finding the closest matching shape.

4.3 Creatures and “Blobby Objects”

Most creatures have a rounded shape with limbs which are roughly rotation symmetric. Such objects can be modelled by drawing their contour, defining a skeleton, sampling the contour and rotating the sample points around the skeleton. The most famous example of this class of algorithms is Igarishi et al.’s “Teddy” application [15]. Various modifications have been suggested, e.g., for smoothing the resulting 3D surface [14], smoothing the underlying skeleton [21], or for modifying the shape by sketching contours of local features [41].

We have implemented a simplified version of the “Teddy” algorithm which automatically computes movable joints such that body parts can be animated [39]. Examples are shown in figure 6. We are now working on evolutionary algorithms for automatically animating such modelled creatures and objects.

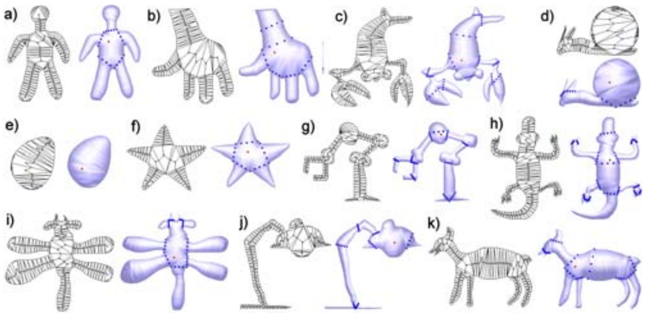


Figure 6: Examples of triangulated sketches (left) and the resulting 3D models with automatically determined joints (right): (a) doll, (b) hand, (c) lobster, (d) snail, (e) egg/stone, (f) sea star (star fish), (g) robot, (h) crocodile, (i) dragonfly, (j) desk lamp, (k) deer. Each image shows the sketched contour and resulting skeleton (left), and the Gouraud shaded 3D model (right). The joints separating movable components are indicated by thick blue dotted lines. The root of the hierarchical skeleton for skeletal animation is indicated by a red dot.

4.4 Buildings and Man-made Objects

Many computer designed items are characterized by a blocky shape, flat or arced surfaces, sharp or evenly rounded edges and corners, many parallel and orthogonal edges and faces, and symmetrical features. These features can be captured using silhouettes which can then be interpreted using application specific constraints, e.g., surfaces of CAD objects frequently form 90 degree angles. Examples include “SKETCH” [40], where complex objects are constructed using a combination of sketched 3D primitives. A subsequent system, “3D Sketch”, creates an edge graph from a user sketch and matches it with existing topologies. After identifying planar faces and determining a view point the edge graph can be projected into a 3D model [28]. Recently active contour models were proposed to allow both curve creation and modification from totally arbitrary view points [17]. The depth coordinates are computed by minimizing the spatial deviation from the original target curve. Impressive models of buildings have been obtained with the “Sketching reality” application [9].

While the review above shows that sketched-based tools do exist for modelling buildings and CAD objects, we would like a simpler and more intuitive interface which allows creation of rough models in less then 30 seconds. At this point of time we are still evaluating options and haven’t found a satisfactory solution.

5. IMPLEMENTATION

QuickTime VR support has been integrated into the system using the QuickTime SDK published by Apple Inc. [4]. The QuickTime VR panorama is drawn to the screen during the SceneLoop callback. The integration layer forwards this event to the QTVR Controller class, which sends new pan and tilt values to the QuickTime movie controller, and requests that the QuickTime VR movie be redrawn. The integration layer is then supplied with the memory address of the panorama output buffer in the form of a Windows bitmap handle. The bitmap data is retrieved from this handle and copied into an Esperient texture buffer. This copying process was used because it was not possible to allow QuickTime to render directly into an Esperient texture buffer. QuickTime supports creating “GWorlds” by wrapping them around existing Windows device-independent bitmaps, but there is no practical method to create these from preexisting memory segments. Furthermore, if QuickTime were permitted to write directly to the texture buffer it may be more difficult to guarantee it would not render at inappropriate times (because Esperient Creator is multi-threaded, it is recommended to lock the texture buffer before writing data to it so other threads are prevented from using it). The QuickTime buffer is copied to the texture buffer of the panorama window in the Esperient scene by directly copying raw bitmap data from the memory space of one buffer to the other. This method only works correctly when both the source and destination bitmaps have the same format. The Esperient Creator API fortunately provides functions to obtain the image format of a texture; this information is then supplied to QuickTime as parameters for creating the GWorld object, which ensures the buffer formats match. The process is illustrated in figure 7.

5.1 Updating View Direction in Panoramas

One of the most important aspects for QuickTime VR integration is that view direction changes within the movie and virtual environment have to correspond. The values are usually set automatically by QuickTime when the user clicks and drags on the panorama image. In our case this is not possible because of the rendering method explained in the previous subsection.

Instead, we are using the Esperient engine to determine the view direction. Because Esperient already supports rotating the view by clicking and dragging, we have been able to accomplish this simply by switching to an alternate viewpoint control system based on the “Shooter” mode, but with movement disabled (allowing viewpoint rotation only). The view direction is then passed to the QuickTime movie controller. The Esperient engine stores this information in a view matrix; this must first be converted into the pan and tilt angles that QuickTime uses. This process, which occurs during the SceneLoop callback, is as follows:

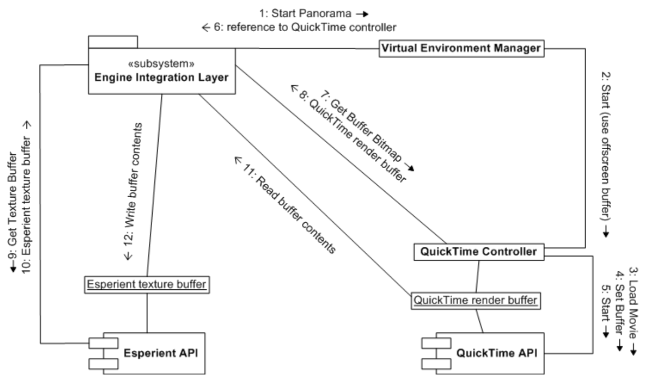


Figure 7: The QuickTime VR rendering process used by the virtual environment rendering framework. 1-2: The integration layer sends a request to display a panorama; an off-screen buffer is chosen to be used. 3-5: The QuickTime controller initialises and starts the movie using the QuickTime API. 6-8: The integration layer requests and is given the buffer QuickTime is using for rendering. 9-10: The integration layer requests the texture buffer of the target rendering object from the Esperient engine. 11-12: The integration layer copies the contents of the QuickTime rendering buffer into the Esperient texture buffer.

1. The view matrix (used to transform the scene to create a unique view) is retrieved from the Esperient engine.
2. The rotation aspects of the view matrix are isolated (translation and scaling information are removed).
3. The inverse of the matrix is calculated, yielding the rotation of the view (rather than the scene).
4. The rotation matrix is converted into a quaternion describing a direction and orientation. Quaternions are numerically more stable when accumulating multiple rotations.
5. The quaternion is converted into yaw (pan), pitch (tilt) and roll values. The roll value is discarded (it is constant in the "Shooter" navigation mode); the other two values are passed to QuickTime after shifting the tilt angle by $\pi/2$ radians.

6. CASE STUDY - DEVELOPMENT OF AN ARCHAEOLOGICAL VIRTUAL ENVIRONMENT

We used the above presented technologies to create a 3D virtual environment of an archaeological site. Archaeological virtual environments have been used since the early 90s and can be divided into two classes: research aids and historical reconstructions.

Interactive three-dimensional visualisations aid research by allowing an accurate 3D representation of the location where artefacts were found. Examples include ARCHAVE [35], which allows the analysis of stratigraphy and the finds within an archaeological site, and GeoSCAPE [20], which facilitates the recording of the position of the features of an archaeological site including artefacts and structures, and their visualisation.

Virtual environments depicting historical reconstructions have often been aimed toward the general public, but can also be used by archaeologists, e.g., to formulate and evaluate hypotheses about how a site was used. They also allow a broader variety of ways to present information more richly than earlier technologies: any number of viewpoints is possible in a virtual environment, features of the site such as buildings and artefacts may be "hyperlinked" to relevant information such as bibliographic resources, and querying can be used to locate and concentrate on important details [30, 6]. One of the strongest advantages of these kinds of virtual environments is their usefulness for the rapid dissemination of research via the Internet and the possibility to collaborate inside the VE [6, 29]. "Virtual museums" or tours aimed at the general public are also becoming more common and are steadily increasing in complexity and animation capabilities [1, 19].

Archaeological sites can contain tens of thousands of objects distributed over a large site. In many cases media, such as images, of these artefacts exist already. 3D reconstruction, however, is time consuming, expensive and requires special equipment.

We created a virtual environment of a central portion of the ancient city of Selinus, a Greek colony in Sicily, as a case study. The University of Auckland's School of Architecture and Planning, in conjunction with its counterpart in the Università degli Studi di Palermo (University of Palermo), runs a joint program every few years in which students from Auckland and Palermo visit archaeological sites including Selinus, and work together on architectural projects [27].

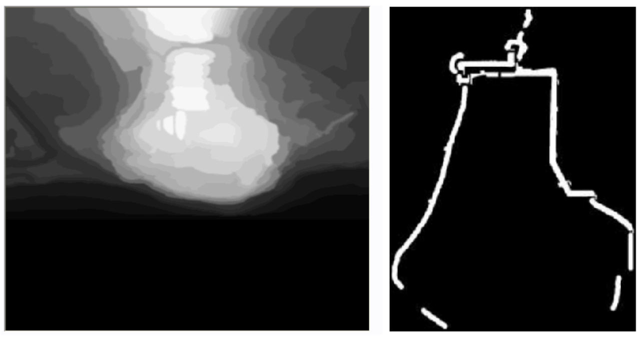
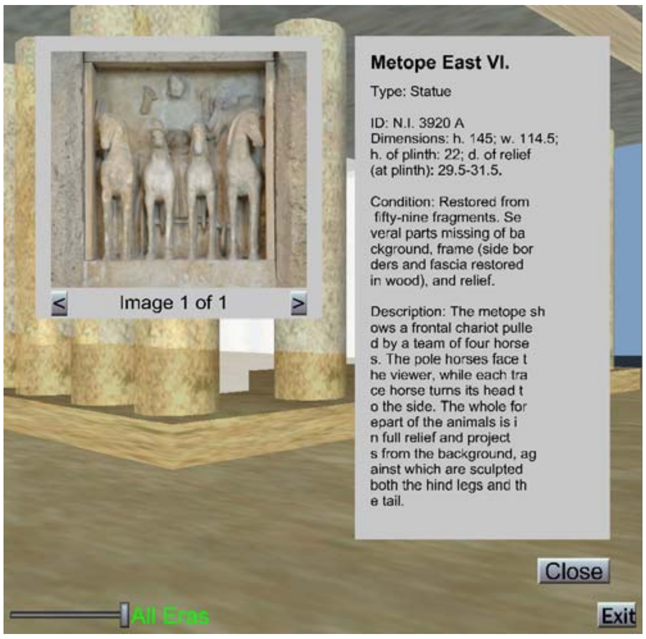


Figure 8: Left: Terrain created by sketching contours and filling regions to create a height map. Right: City walls modelled by drawing them using a paint-like interface.

We developed sketch and paint interfaces to create coarse virtual representation of the terrain and basic structures such as rivers, seas, and city walls (see figure 8). Rough representations of buildings were created using traditional modelling tools such as "Blender", but we are working on replacing them with sketch interfaces. The tools above create a very rough representation which provides a suitable context for exploring a site. In order to correctly perceive and analyse structures and artefacts a high resolution representation is necessary. We achieve this by using existing images and QuickTime VR movies and insert them into the appropriate locations of the virtual environment using a drag-and-drop tool. An example of a QuickTime VR movie is given in figure 3 (right) and a detail view of a structure is presented in figure 9.

7. RESULTS

Integrating QuickTime VR movies into virtual environments provides rich content with suitable context. Identifying the precise camera position for a QuickTime VR view is difficult without additional data such as its GPS position. However, even without seamless integration the media blend well and allow a focus-and-context style exploration of the domain. The game environment provides an overall context for QuickTime VR movies and enables users to get an overview of the entire scene and the QuickTime VR movie's


Metope East VI.

Type: Statue

ID: N.I. 3920 A
Dimensions: h. 145; w. 114.5;
h. of plinth: 22; d. of relief
(at plinth): 29.5-31.5.

Condition: Restored from
fifty-nine fragments. Se
veral parts missing of ba
ckground, frame (side bor
ders and fascia restored
in wood), and relief.

Description: The metope sh
ows a frontal chariot pulle
d by a team of four horse
s. The pole horses face t
he viewer, while each tra
ce horse turns its head t
o the side. The whole for
epart of the animals is i
n full relief and project
s from the background, ag
ainst which are sculpted
both the hind legs and th
e tail.

Figure 9: A detail view of a structure/artefact.

camera position within it. Vice versa the QuickTime VR movie is content rich and provides a more realistic representation than the modelled environment and hence provides a better context for understanding the terrain, e.g., how the architecture blends into the natural environment.

Integration of QuickTime VR movies into a game engine is challenging since interactions with both media must be synchronised. We hence used only the Esperient Creator API, i.e., not the core engine components. The resulting framework is quite flexible and can be ported to other game engines.

Integration of images and videos into virtual environments via billboards and textures has always been an integral tool for content creation and is supported by most game engines. By developing suitable interfaces these images can be made an integral part of the scene. In an archaeological context images are well suited to provide detail information about artefacts.

We performed a basic usability study [18] and found that navigation is easy, although the existing Selinus virtual environment is relatively simple. More challenging scenarios are required to test how well the system performs in complex environments. Interaction with QuickTime VR panoramas works well. They are easy to see and activate, and view rotation is synchronised accurately. Rotating the view with the mouse results in sluggish handling, which needs to be improved. Interaction with artefacts is relatively easy, but they can be difficult to see at a great distance due to their size. The inclusion of a top-down map should alleviate this problem.

Rough 3D content can be created using sketch and paint-like modelling tools. We have presented techniques for creating terrains, vegetation and simple creatures and rounded objects. In all cases the objects can be created in less than a minute and often just a few seconds are necessary (e.g., simple trees and creatures). The resulting models are suitable for prototyping and for providing context to the 3D environment. Direct animation of the sketched creatures has been shown to be unintuitive [39], and we are hence working on automatic algorithms where the user only has to indicate the general behaviour using sketch input, e.g., a character's motion path [34]. In order to create a crowd of people or a forest of trees in less than a minute we are currently developing tools to replicate models over a sketched region. The user has to draw one example model and sketch a point distribution and the algorithm will automatically fill the entire region with copies of the model according to the sketched distribution.

8. CONCLUSION AND FUTURE WORK

Virtual environments can be created efficiently by designing a rough 3D model using sketch and paint-like interfaces, and integrating images and QuickTime VR movies to provide detail views of relevant information. The QuickTime VR movies are "fused" into the game engine environment providing a context-and-focus view of different aspects of a site. The framework is most suitable for creating virtual representations of a real site which are used for exploration and where limited interaction with scene objects is required. Examples are urban design, architecture, archaeology and prototyping.

The development of a unified interface for the sketch and multimedia input is challenging. In our archaeological example application we used two specialised paint-like interfaces for creating the terrain and city walls. In order to employ the tool for arbitrary applications we need a unified interface which allows us to define different semantics for stroke input depending on the context, stroke position, and user selections. For example, a long curvy sketch could be a road, river, tree stem, or snake depending on the context (previously and subsequently drawn sketches, relative size, position and orientation). We are currently developing a prototype using a paper metaphor where one piece of paper shows the overall terrain and objects are indicated by symbols or drawn directly at the desired position. If a symbol indicates multimedia content then the user selects the corresponding file and associated information. For sketch content a new piece of paper is "opened" and the sketch input is interpreted according to the type of object indicated by the symbol (terrain, vegetation, character, building).

We are currently developing more advanced sketch tools for 3D content creation. Our goal is it to create a rough 3D environment with terrain, buildings, roads, rivers, lakes and vegetation in a couple of minutes. In addition to QuickTimeVR movies we are investigating the use of photo clusters as done in the "Photosynth" technology to incorporate realistic content [26]. We would also like to achieve an "incremental design" where the user can replace a simplified representation (sketched object, photo) with a more realistic representation (high-detail 3D model) if desired. This would be especially useful for rapid prototyping, e.g., for game development or movie production.

9. REFERENCES

[1] 3DRewind. 3D Rewind Rome, 2009. `http://www.3drewind.com`.

[2] P. Anders, editor. *Envisioning Cyberspace: Designing 3D Electronic spaces*. McGraw-Hill, New York, 1999.

[3] Apple Inc. QuickTime VR: QTVR Panoramas, 2007. `http://developer.apple.com/legacy/mac/library/documentation/QuickTime/InsideQT_QTVR/0Preface/QTVR-preface.html`.

[4] Apple Inc. QuickTime SDK, 2009. `http://developer.apple.com/quicktime/download`.

[5] Apple Inc. QuickTime VR Tools, 2009. `http://www.apple.com/quicktime/resources/tools/qtvr.html`.

[6] J. Barcelò, M. Forte, and D. Sanders, editors. *Virtual Reality in Archaeology (BAR International Series 843)*. Oxford: Archaeopress, 2000.

[7] C. Basdogan, M. Sedef, M. Harders, and S. Wesarg. VR-based simulators for training in minimally invasive surgery. *IEEE Computer Graphics and Applications*, 27(2):54–66, 2007.

[8] I.-H. Chen, B. MacDonald, and B. Wünsche. Mixed reality simulation for mobile robots. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA '09)*, pages 232–237. IEEE, May 2009.

[9] X. Chen, S. B. Kang, Y.-Q. Xu, J. Dorsey, and H.-Y. Shum. Sketching reality: Realistic interpretation of architectural designs. *ACM Trans. Graph.*, 27(2):1–15, 2008.

[10] A. Cockburn, A. Karlson, and B. B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys*, 41(1):1–31, 2008.

[11] Esperient Corporation. Esperient homepage, 2009. `http://www.esperient.com`.

[12] J. Gain, P. Marais, and W. Strasser. Terrain sketching. In *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 31–38. ACM, 2009. `http://people.cs.uct.ac.za/~jgain/publications/terrsketch.pdf`.

[13] D. Gardner. The power of Second Life. E-Commerce Times, Dec. 2006. `http://www.ecommercetimes.com/story/54826.html`.

[14] T. Igarashi and J. F. Hughes. Smooth meshes for sketch-based freeform modeling. In *I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 139–142. ACM, 2003.

[15] T. Igarashia, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3d freeform design. In *Proceedings of SIGGRAPH '99*, pages 409–416. ACM Press, 1999.

[16] J. Isdale, C. Fencott, M. Heim, and L. Daly. Content design for virtual environments. In K. M. Stanney, editor, *Handbook of Virtual Environments - Design, Implementation and Applications*. Lawrence Erlbaum Associates, 2002.

[17] L. B. Kara and K. Shimada. Sketch-based 3d-shape creation for industrial styling design. *IEEE Computer Graphics and Applications*, 27(1):60–71, Jan. 2007.

[18] D. Keymer. User interfaces for the effective exploration and presentation of virtual archaeological sites. Master's thesis, Department of Computer Science, University of Auckland, Auckland, New Zealand, Mar. 2009.

[19] Y.-S. Kim, T. Kesavadas, and S. M. Paley. The virtual site museum: a multi-purpose, authoritative, and functional virtual heritage resource. *Presence: Teleoperators and Virtual Environments*, 15(3):245–261, 2006.

[20] J. Lee, H. Ishii, B. Duun, V. Su, and S. Ren. Geoscape: designing a reconstructive tool for field archaeological excavation. In *CHI '01 extended abstracts on Human factors in computing systems*, pages 35–36. ACM, 2001.

[21] F. Levet and X. Granier. Improved skeleton extraction and surface generation for sketch-based modeling. In *GI '07: Proceedings of Graphics Interface 2007*, pages 27–33. ACM, 2007.

[22] Linden Research Inc. Second Life Homepage, 2009. `http://secondlife.com/`.

[23] B. Liu, G. Sun, M. Zhang, and P. S. Jassal. Sketch-based terrain modelling. COMPSCI 715 Project Report, Dept. of Computer Science, University of Auckland, New Zealand, Oct. 2009.

[24] S. Marks, J. Windsor, and B. Wünsche. Evaluation of game engines for simulated surgical training. In *GRAPHITE '07: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 273–280. ACM, 2007.

[25] G. McCord, B. C. Wünsche, B. Plimmer, G. Gilbert, and C. Hirsch. A pen and paper metaphor for orchid modeling. In *Proceedings of the 3rd International Conference on Computer Graphics Theory and Applications (GRAPP 2008)*, pages 119–124, Jan. 2008.

[26] Microsoft Life Labs. Photosynth Homepage, 2009. `http://livelabs.com/photosynth`.

[27] M. Milojevic. Exceptional access: Re-presenting ancient selinus virtually. In *Proceedings of Interface: Virtual Environments in Art, Design and Education*, Sept. 2007.

[28] J. Mitani, H. Suzuki, and F. Kimura. *3D sketch: sketch-based model reconstruction and rendering*, pages 85–98. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[29] C. Renfrew and P. G. Bahn, editors. *Archaeology : theories, methods and practice*. Thames and Hudson, London, 4th edition edition, 2004.

[30] C. Renfrew, M. Forte, and A. Siliotti, editors. *Virtual archaeology: great discoveries brought to life through virtual reality*. Thames and Hudson, London, 1996.

[31] T. Ropinski, F. Steinicke, and K. Hinrichs. A constrained road-based VR navigation technique for travelling in 3d city models. In *ICAT '05: Proceedings of the 2005 international conference on Augmented tele-existence*, pages 228–235. ACM, 2005.

[32] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski. Finding paths through the world's photos. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–11. ACM, 2008.

[33] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 835–846. ACM, 2006.

[34] M. Thorne, D. Burke, and M. van de Panne. Motion doodles: an interface for sketching character motion. *ACM Trans. Graph.*, 23(3):424–431, 2004.

[35] E. L. Vote, D. A. Feliz, D. Laidlaw, and M. S. Joukowsky. Archave: A virtual environment for archaeological research. In *Proc. of the 28th Annual International Conference of Computer Applications in Archaeology*, Apr. 2000.

[36] J. Wither, F. Boudon, M.-P. Cani, and C. Godin. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. *Computer Graphic Forum. Special issue: Eurographics 2009*, 28(2):541–550, 2009.

[37] P. Wonka, P. Müller, B. Watson, and A. Fuller. Urban design and procedural modeling. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, pages 229–229. ACM, 2007.

[38] X. Xie and B. C. Wünsche. Efficient contour line labelling for terrain modelling. In *Proceedings of the 33rd Australasian Computer Science Conference (ACSC 2010)*, Jan. 2010.

[39] R. Yang and B. C. Wünsche. Lifesketch - a framework for sketch-based modelling and animation of 3d objects. In *Proceedings of the Australasian User Interface Conference (AUIC 2010)*, Jan. 2010.

[40] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. SKETCH: An interface for sketching 3d scenes. In *Proceedings of SIGGRAPH '96*, pages 163–170. ACM Press, Aug. 1996.

[41] J. Zimmermann, A. Nealen, and M. Alexa. Sketching contours. *Computers & Graphics*, 32(5):486–499, 2008.