Evaluation of Game Engines for Simulated Clinical Training

Stefan Marks[*]
dev.stefan.marks@gmx.net

John Windsor[†]
j.windsor@auckland.ac.nz

Burkhard Wünsche[‡]
burkhard@cs.auckland.ac.nz

The University of Auckland
Private Bag 92019, Auckland Mail Centre
Auckland 1142, New Zealand

ABSTRACT

The increasing complexity and costs of clinical training and the constant development of new procedures has made virtual reality based training an essential tool in medical education. Unfortunately, commercial training tools are very expensive and have a small support base. Game engines offer unique advantages for the creation of highly interactive and collaborative environments.

This paper examines the suitability of currently available game engines for developing applications for clinical education and training. We formally evaluate a list of available game engines for stability, availability, the possibility of custom content creation and the interaction of multiple users via a network. Based on these criteria, three of the highest ranked engines are used for further case studies.

We found that in general it is possible to easily create scenarios with custom medical models that can be cooperatively viewed and interacted with, though limitations in physical simulation capabilities make some engines less suitable for fully interactive applications. We show that overall game engines represent a good foundation for low cost clinical training applications and we discuss technologies which can be used to further extend their physical simulation capabilities.

Categories and Subject Descriptors

I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Gaming*; J.3 [**Life and Medical Sciences**]: Medical information systems; K.3.1 [**Computers and Education**]: Computer Uses in Education—*Collaborative learning*

[*]Department of Computer Science, Division for Biomedical Imaging and Visualization

[†]Faculty of Medical and Health Sciences, Department of Surgery, Advanced Clinical Skills Center

[‡]Department of Computer Science, Division for Biomedical Imaging and Visualization

NZCSRSC 2008, April 2008, Christchurch, New Zealand.

Keywords

clinical simulation, game engine, physically-based animation, collaboration

1. INTRODUCTION

The rising complexity and costs of clinical training and the development of new procedures has increased the importance of clinical simulators for education and training purposes. *Surgical simulators* represent a major part of the large variety of applications. Most commercially available simulators cover the area of endoscopic respectively laparoscopic procedures (e.g. Procedius MIST [26], LapSim [31], LAP Mentor [30], VEST System One [28], LapVR [22], EndoTower [38]). This kind of procedure requires well developed skills of the surgeon with respect to coordination of the camera and the surgical instruments that are not in direct view and are represented only on a 2D screen, sometimes with changed orientation due to camera rotation. The above mentioned systems are all able to train basic procedures in camera and instrument handling, before training the medical and surgical aspects.

Nevertheless, those technical skills are not the only necessity for a surgeon. The AGCME Outcome project [1] lists six general competencies which include other skills like patient care, medical knowledge, the ability to continuously learn and improve by practise, interpersonal and communication skills, professionalism, and the awareness of the health care system with its resources and demands as a whole.

Most of the mentioned simulator systems only train the technical and procedural skills of a surgical procedure, but lack the other aspects of the above list. Few physical simulators with mannequins (e.g. MedSim-Eagle [11]) enable small groups of residents to practise the cooperative aspects (i.e. interpersonal and communication skills) of medical or surgical procedures, but are very cost-intensive and thus likely to be unaffordable for most institutions.

One factor that is responsible for high costs of surgical simulators is the fact that certain parts of them are repeatedly reinvented. All simulators need at least graphical output capable of displaying 3-dimensional models with a high level of realism and user interfaces for operating and configuration of the simulator, an underlying physical simulation model, and event handling for input devices (see Figure 1). Some simulators are capable of adding the audible aspect of a surgical procedure and thus need a module for sound generation.

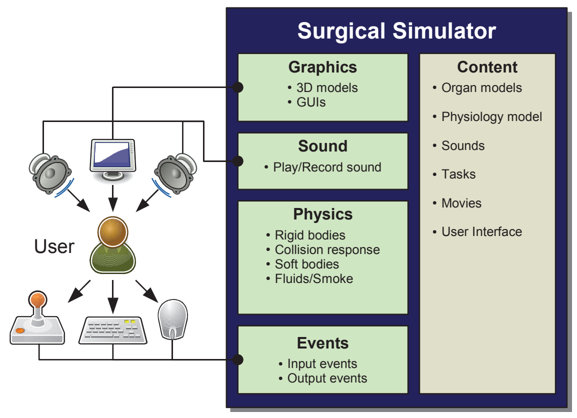


Figure 1: Functional blocks of a surgical simulator.

There have been attempts to create extensible frameworks for building surgical simulators upon (e.g. SPRING [27], GiPSi [10], SOFA [3]). They all incorporate the above mentioned modules and a variety of mathematical models for the physical simulation and interaction. But except for SPRING (ironically the oldest project in the list) they all lack the capability of networking with other simulators to build collaborative scenarios.

Teamwork is an overall important factor not only for surgeons but for all clinical personnel. Therefore, simulation systems that neglect the collaboration and communication of users can only deliver a part of the overall education.

This paper summarises the results of a first year doctoral research programme in which the potential of gaming engines is evaluated particularly in relation to the above mentioned features and in their ability to support collaboration and communication between multiple users.

2. GAME ENGINES

The use of games or game engines for medical education has not been extensively explored with many aspects still to be investigated. One reason for this might be the lack of concordance between the seriousness of medicine and the playful, sometimes violent nature of computer games. Nevertheless, game engines offer a vast pool of useful concepts and resources in both technical and educational aspects.

Projects like the "Serious Game Initiative" focus on offering help to "organize and accelerate the adoption of computer games for a variety of challenges facing the world today." A subproject founded by this initiative is "Games for Health" [29], mainly focusing on games used in various health care sectors.

Previous authors have so far concentrated on applications where the game content was about learning facts, rather than tasks, procedures and teamwork. For example, Wünsche et al. [41] have examined how game engines can be used for visualising medical datasets, and Mackenzie et al. [24] utilise a game engine for anatomical education.

"Pulse!!" [33] is a recently developed, major project of the Texas A&M University utilising a game engine for teaching single users the procedures and systems of a health-care facility. Even more recently, the simulation "3DiTeams" [32] has extended this principle to include multiple users who collaborate in an emergency room setting over a network.

2.1 Game Engine Design

A game engine is a complex software system necessary for developing and playing games. Two different games with the same underlying engine differ by the *game content*, i.e. graphics, sounds, storyline. Game engines build a bridge between this content and the underlying hardware. With the help of an operating system abstraction layer, the same game content can be run on many platforms (e.g. Windows, Linux, XBox) without change.

Modern game engines consist of all or a subset of functional blocks depicted in Figure 2.

The *Graphics Engine* loads, displays, manipulates and manages all the data related to graphical content and visual effects. 3D models of objects, landscapes, buildings, objects, animals, and players can be loaded, textured, lit, and animated. Additional effects (e.g. blurring, lens distortion, depth of field) can be added to enhance the visual realism. Particle systems are utilised to simulate fire, smoke, bubbles, blood, etc.

All audible content like sound effects, ambient noise, and music is handled by the *Audio Engine*. In connection with modern soundcards it is possible to simulate acoustic obstruction by objects, environments other than air, reverb, Doppler effect and the spatial position of sound sources.

The total memory usage of game content is often higher than the memory provided by the gaming platform. Because not all of this data is needed simultaneously, the *Memory Management* is responsible for purging unused content from memory and in turn providing and managing requested memory for new content. Modern engines parallelise tasks like graphics, sound, physics, and AI. To balance the workload of all these tasks efficiently, especially on multiprocessor platforms, the *Process Management* is utilised.

Another essential part is the *Event handling*. Input devices, like joysticks, mice, keyboards, and gamepads generate events as well as network, timers, other components of the gaming hardware, game scripts and many other sources. These events are handled, filtered and distributed by one central event loop.

Some media like music or video does not need to be loaded into memory before being played back, but can instead be streamed to save precious memory resources. The *Streaming* mechanism is also capable of loading resources via the network from other servers.

Realistic behaviour of game objects has become more and more important in the recent years. The *Physics Engine* implements advanced mathematical models for calculating rigid body simulations of arbitrarily shaped and articulated objects (e.g. vehicles, machines). With the devel-

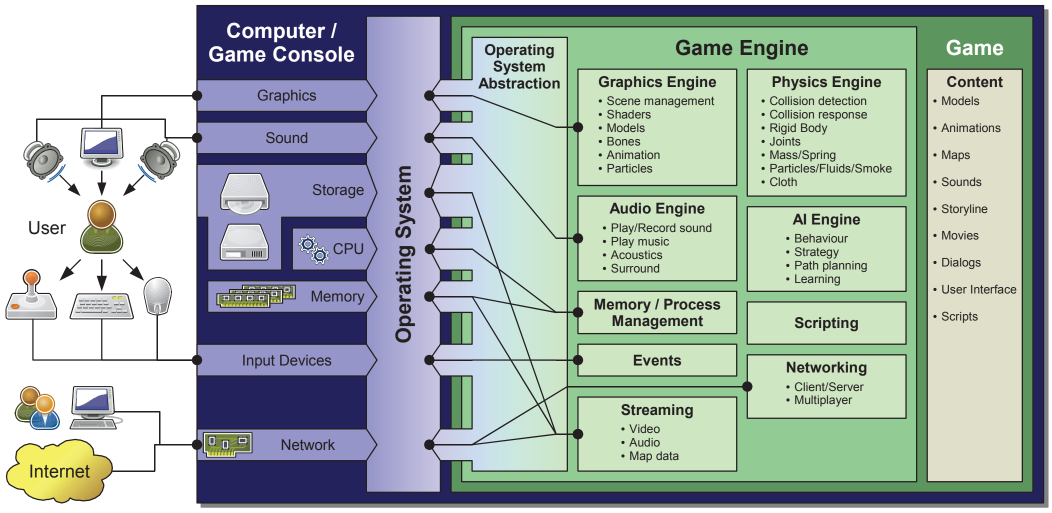


Figure 2: Functional Blocks of a Game Engine.

opment of highly sophisticated physics engines like Havok Physics^{TM}[20] or PhysX^{TM}[2], the simulation of soft bodies, cloth, fluids, and smoke has become possible, accelerated either by specialised hardware or the computational power of the graphics hardware [19].

Artificial intelligence, provided by the *AI Engine*, is needed for controlling Non-Player Characters (NPC), the computer controlled antagonists in games. NPCs have to make decisions about how to follow, avoid or attack the player, and how to react to aggressive or defensive actions in a realistic and effective manner. AI Engines incorporate Computer Science topics like neural networks, state machines, A* search, and much more.

The flexibility of game engines is their greatest strength in creating manifold content. This is achieved by *Scripting* languages that allow an immediate access to the functions of the game engine. By scripts, the game content gets its typical "fingerprint", how a game is to be played and controlled, how the story develops, and how interactive and immersive the game environment appears.

The final important functional block of a game engine is *Networking*. By sending the state, movement and other information of each player and NPC over the network, other connected human players can collaborate in a game, because they all see the identical state of the game world at the same moment. The networking functions cope with network problems like packet loss or different runtimes of data packets from clients to servers and vice versa.

2.2 Technical Advantages

Since the game market is incredible competitive and incorporates both large established and innovative new vendors, game engines are constantly updated and utilise the latest graphics hardware and graphics algorithms. In addition, since most users are unable to constantly update their machines, game engines are designed for handling the same game content on hardware with different speed, memory size and features.

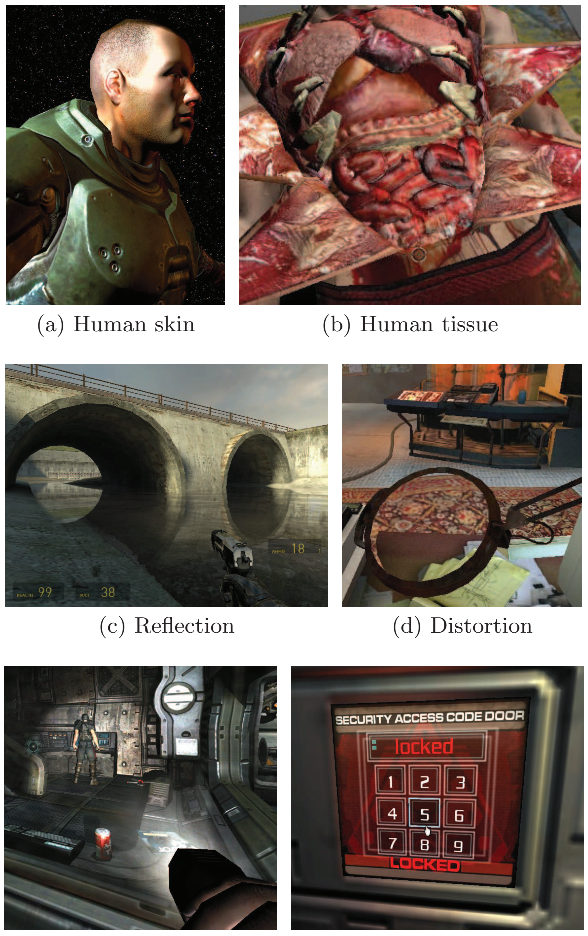
Playing computer games is no longer an action for individuals but has evolved into multiplayer gaming, bringing together several thousand players at the same time. Consequently, many game engines incorporate properly constructed and tested network support that serves well in connecting multiple users for cooperatively accomplishing tasks, even worldwide, unrestricted by location and physical boundaries. Built-in support for recording and playing sound over the network enables the players to communicate in a natural way to coordinate their actions. Textual input of messages serves as an alternative way.

Due to the fact that games are marketed internationally, game engines are able to deal with different languages. User interfaces, sound support and input devices can be customised accordingly.

After games have been introduced to the market, they are constantly and intensively used by customers which results in massive feedback about errors and flaws. After some months and sometimes only weeks, patches are available to fix these issues. During this time, a large number of developers will have built up, who have gathered experience in modifying the game content. They form an international support community, often willing to help others when problems arise in building custom content for a game engine.

2.3 Important Features

One major important aspect for medical simulation is visual realism. With game engines supporting the most modern graphics hardware, this issue can easily be addressed. To enhance the realism in their hystheroscopy simulator, Bachofen et al. introduces bump mapping, spotlights, shadows, lens distortion, depth of field, bubbles, and floating tissue [7]. This list is only a subset of effects used in modern games, as the screenshots in Figure 3 illustrate.



(a) Human skin (b) Human tissue

(c) Reflection (d) Distortion

(e) Spotlights (f) User interfaces

Figure 3: Graphical capabilities of modern game engines. The screenshots are taken from the games Doom 3, Half-Life 2 and Quake 4.

The acoustic environment of a medical procedure is also an important part of a simulation [39] and can also be easily and accurately simulated with the features available in game engines. It may provide audible feedback of instruments used during medical procedures as well as reactions of the simulated patient, like pain or relief.

The physical simulation of objects in games is a relatively young area and thus does not yet cover the mathematically demanding aspects of soft tissue simulation (e.g. [14, 12, 23]) or cutting (e.g. [16]). This drawback can be compensated for either by playback of animations, skeletal animation, simple mass-spring systems or, if possible, extension of the physics engine. The manufacturers of physics engines are currently working on introducing new features like fluids and soft body simulation, so the features missing right now may be available in the near future.

Networking (multiplayer) capabilities of games offer a great chance of building collaborative training scenarios. Players can see the position and state of their team members, and can communicate with each other by microphones and headphones or textual messages.

All of these features are useful for clinical simulations. Advanced graphics makes the simulation more realistic and limited hardware requirements allow users in development countries and smaller clinics to employ the software. The network support and GUI customisation can bring together multiple users for training of cooperative tasks, unrestricted by classroom walls and country boundaries and, when designed carefully, even independent of language barriers.

3. METHODOLOGY

3.1 Engine Selection

We started our selection of suitable game engines with an evaluation of an internet game engine database [15]. At the time of this evaluation (July 2007), this database contained 278 engines. We disregarded engines still in an early development state or those that were not developed or maintained any more. Engines without sound or other essential components were also removed from the list. Of the remaining engines, we selected those with in-built means of creating new game environments (maps). This last criterion is an important aspect for reducing the complexity of the editing process as there is no need for purchasing, installing and setting up external editors and necessary conversion tools, assumed the latter exist at all.

After the reduction of the original list by this selection process, we chose three inexpensive and in our opinion popular game engines for further evaluation.

- Unreal Engine 2 [17]
- id Tech 4 [40]
- Source Engine [35]

3.2 Evaluation

All engines were tested for their suitability for collaborative simulated surgical training applications by examining the following aspects:

Editing: Is everything that is necessary for creating and manipulating custom content included in the software? How is the editing process for a map started? Are the construction principles that are used during the process of building a map intuitive?

Content: How easy is the process of including game content as well as external models into the map? Which restrictions have to be considered when importing custom models?

Gameplay: How well can two or more users interact within the map and with the custom model? Are there any restrictions in the physical interaction?

Editing of models was performed with the 3D editor Blender [8]. This software is free, in contrast to commercial and expensive 3D editors such as 3D Studio Max [4] or Maya [5], and has import and export filters for all important 3D formats that were necessary for including custom models into the maps created for each game engine.

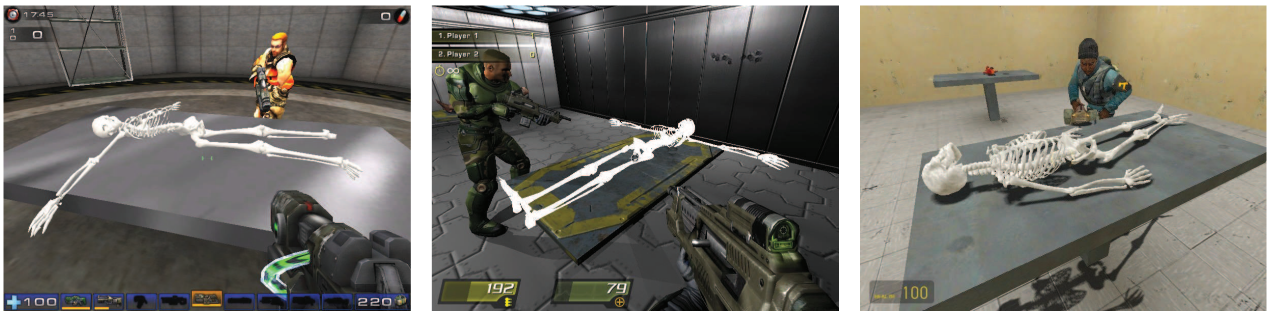

(a) Unreal Engine 2 (b) id Tech 4 (c) Source Engine

Figure 4: Screenshots of our simulation scenarios implemented with different game engines.

4. RESULTS

4.1 Unreal Engine 2

The following results are based on the game "Unreal Tournament 2004."

Editing: The Unreal Engine 2 map editor "UnrealEd 3" is started as a standalone program. It incorporates model viewer, texture browser, script editor and other components necessary for editing a map (see Figure 6(a)). In contrast to the editors of the other two discussed engines, the editing process is of subtractive nature. Volumes where players are supposed to move in have to be "carved out" from the originally solid game world.

An editing concept common to all editors of the three evaluated game engines is the "brush." It is used for selecting, for example, the areas that will be subtracted from the game world. But it can also be used for adding walls, spheres, stairs or other simple geometries.

Geometrically complex objects like shelves or engines are selected from a list, added into the map, and can then be moved, rotated, and changed in their behaviour or attributes. The same principle applies for physical objects like rigid bodies or joints.

Content: We constructed a room with a metal shelf (game content) and a custom skeleton model on a table (custom content). The skeleton[1] was split into parts (torso, skull, legs, and arms), which were then inserted as physical objects and connected by ball joints (see Figure 5). The file format for inserting custom models can be one of `.LWO` (Lightwave Object File), or `.ASE` (ASCII Scene Exporter). We used the latter due to having an `.ASE` export filter in Blender,

Gameplay: We started the map in multiplayer mode and interacted with the static and dynamic objects.

Non-physical actions and states are well synchronised between the server and the client. Player positions, orientations and states and also optical effects like decals (e.g. for scorchmarks) appear equally on both sides.

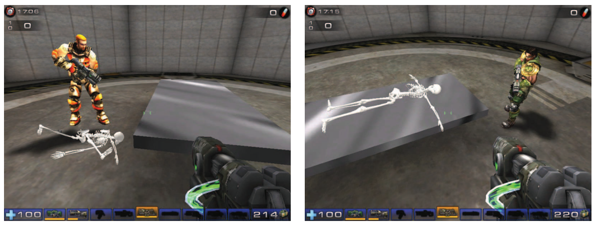


Figure 5: Asynchronous state of the skeleton on the server (left) and the client (right).

The articulated skeleton can be moved by applying forces. This works well in single player mode and on the server side in multiplayer mode. However, the multiplayer client shows unexpected behaviour. When force is applied, the graphical representation of the skeleton stays in place, whereas its physical representation moves (see Figure 5).

This asynchronism of the physics engine is not considered an error, as at 2003, the time of the release of the game, the physical simulation of game objects was not yet an important aspect of gameplay. Nevertheless, users wanted to create multiplayer maps with synchronised physical objects and thus developed a modification of the physics engine [42]. Due to the age of the Unreal Engine 2, this project has undergone no further improvement since 2005 and is now no longer available on servers.

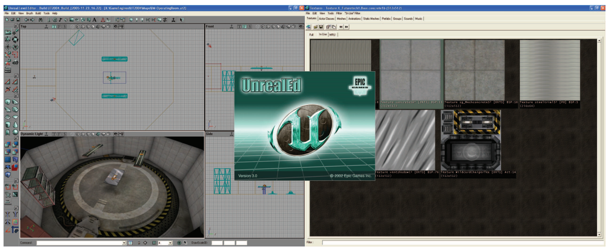
4.2 id Tech 4

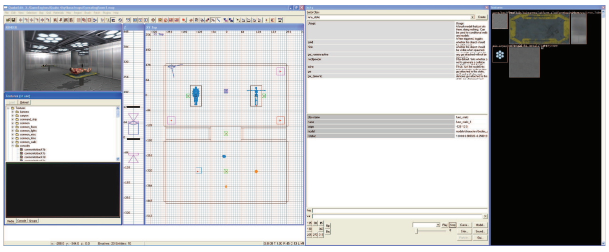
The following results are based on the game "Quake 4." This game uses a more recent version of the id Tech 4 engine than the game "Doom 3".

Editing: The id Tech 4 engine incorporates a set of editors necessary for building maps and inserting custom content (see Figure 6(b)). All of them can be started separately to edit, for example, maps, articulated figures, effects, materials, and scripts.

The map editor "Radiant" has a simple user interface, including a world view and a texture and model browser. Like the editors of the other two discussed engines, it also uses the brush concept for adding simple geometries and a selection

[1]Skeleton model source: `http://artist-3d.com/free_3d_models/dnm/model_disp.php?uid=637`


(a) "UnrealEd 3" (Unreal Engine 2)


(b) "Radiant" (id Tech 4)

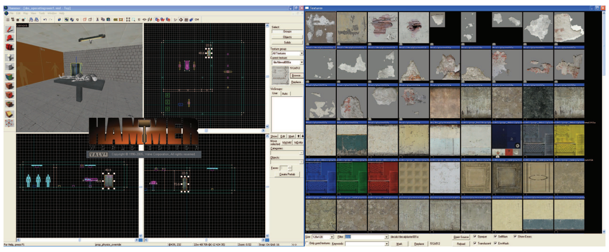

(c) "Hammer" (Source Engine)

Figure 6: Screenshots of the map editors of the three evaluated game engines.

list for more complex objects. In contrast to the other two editors which use four windows for the top, front, side, and 3D view of the scene, this editor is restricted to a single window with the top view in conjunction with a simplified tall window for adjusting the height of placed objects.

Content: With the map editor we created a simple room with two tables, on which we placed a game content model of a dissected body and a static custom content skeleton model, imported from an `.ASE` file (see Figure 4(b)). We placed additional objects (fire extinguisher, book, gas bottle) in the scene to evaluate collaborative interactions with physical objects.

When we tried to articulate the skeleton by connecting the limbs and skull to the torso, we discovered that the physical support is limited to simple rigid bodies. This limitation was unexpected, due to the fact that we also worked with the id Tech 4 engine based game "Doom 3." In this game, a moveable crane with heavy, swinging load appears at least in one map. Its movements can be controlled by the user and the animation of the load is handled by the physics engine. Further investigation revealed that the physics engine used in the game "Doom 3" is part of the game content, but not of the basic id Tech 4 engine [21].

Gameplay: The map was loaded in multiplayer mode and entered by two users. Player positions, orientations and states as well as optical effects are synchronised well between server and client.

Physical objects can be manipulated by both, although the refresh rate of the position and orientation of physical objects on the client is slow and results in a jerking movement. This problem could not be solved by manipulations of the server settings. Additionally, some physical items also showed the asynchronous behaviour of their graphical and physical representations as for the Unreal Engine 2. It is yet unknown for which kind of objects this applies and if there are possible countermeasures.

4.3 Source Engine

The SDK of the Source Engine includes editors and helper programs and thus enables the construction of new maps and even modification of the source code of the engine. Permission to download it is obtained by purchasing a game of the "Half-Life 2" series.

Editing: Maps are created and modified with the map editor "Hammer" (see Figure 6(c)). The producer of the engine, Valve, also refers to the free XSI ModTool from Softimage [6], which, in conjunction with a special plugin that is available on the website of Valve, can be used for creating static and animated models.

The editor can be switched into different modes, like constructing solid objects, placing complex objects, moving objects, and texturing them. Complex objects (entities) are not only geometrically complex models, but also physical objects, physical constraints, light sources, etc.

Content: We created a test room with some game content objects (e.g. locker, lamps) and a table with the custom content skeleton. In contrast to the Unreal Engine 2 and the id Tech 4 engine, a model imported into the Source Engine may only consist of a maximum of 32768 vertices. This also limits the number of triangles to a maximum of about 11000. These limits are coded into the engine and may not be changed. For performance reasons, Valve suggests to reduce the complexity of models to below 10000 triangles [36]. We discovered that this limitation can be overcome by splitting the model into parts that are assembled into one object when converting the model data into the game engine's internal format.

The conversion programs for models and textures are purely command line based. To convert a single model into the engine format, one has to drag the compilation file onto the converter in the explorer view, or start the process by entering a command line instruction. These command line based programs could be utilised easily to automate a complex process of creating maps and models from medical datasets.

Gameplay: Compared to the other two engines, the Source Engine performed best. The position, orientation and state of the users character as well as the physical simulation synchronised well and fluently on server and client (see Figures 7 and 4(c)).

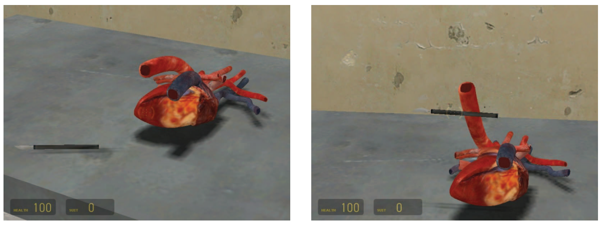


Figure 7: Screenshots of an interactive simulation scenario implemented with the Half-Life 2 engine. The user can interactively bend the vessels of the heart model with a tool.

5. CONCLUSION

Modern game engines contain many features that would be necessary for building a clinical training application. Graphics, audio and network capabilities are highly developed and allow the creator of applications to focus on content rather than details of the implementation. The underlying hardware is optimally used. Multiuser interaction is possible by multiplayer scenarios and allows the training of teamwork and cooperation.

In contrast, highly mathematical physics models for simulation of soft tissue are (not yet) possible with game engines. Basic soft tissue interaction can therefore either be simulated by the use of simple mass-spring models [25] or by the extension of the physics of a game engine to mathematically more sophisticated models, if the engine allows for these changes (e.g. Source Engine).

Predefined file formats can pose difficulties when converting medical images and models. These formats may be limited in their number of vertices or faces and thus would need preprocessing to reduce the amount of information without loss of optical detail. Another possibility for overcoming these limits is to split complex objects into parts that are kept together (e.g. by constraints [41]). On the positive side, the necessary file formats for the examined three engines are well documented (`.ASE`: [34], `.SMD`: [37]).

An interesting aspect of the Source Engine is the fact that not only the material and model compilation files are text based, but also the file format for maps. In conjunction with the command line based tools, this could lead to the development of a fully automated tool that reads patient related medical data and constructs a map including the custom patient models for interaction and training.

6. DISCUSSION AND FUTURE WORK

Game engines allow for a relatively quick creation of interactive scenarios for simulated clinical training. Nevertheless, there are still restrictions in the physical modelling of soft tissue, which prevent their use for training surgeons in the area of technical skills, e.g. laparoscopic procedures. The true strength of game engines lies in their networking capabilities. The ability to interactively bring together several users in a simulation scenario is ideal for the training and assessment of *teamwork*, which is an aspect of healthcare that has often been overlooked, but is now beginning to receive due attention [9].

Our future work will emphasise the development of training frameworks, scenarios and assessment methods that are appropriate for the evaulations and training of teams in clinical settings. The existent physical simulation capabilities of game engines could be utilised to enhance the realism of these training scenarios, e.g moving instrument trays, cables connected to the patient. Tools for automatically converting real patient data to game engine specific files (e.g. based on the Source Engine command line tools) could be used to create a variety of training scenarios which are based on real-life cases.

We will also contine to monitor the latest development of new game engines, e.g. CryEngine 2 [13] and Unreal Engine 3 [18]. Among their features are enhanced graphics and physical modelling techniques. With these it is possible to blend animations while maintaining a set of constraints. Geometric models can be deformed arbitrarily by displacement textures. Physically correct simulation of smoke and liquids can also be used. These features will allow even more realistic simulations and will be subject to further analysis.

7. REFERENCES

[1] ACGME. ACGME Outcome Project – General Competencies [online]. Sept. 1999 [cited 17.08.2007]. Available from: `http://www.acgme.org/outcome/comp/compMin.asp`.

[2] Ageia Technologies. Ageia [online]. 2007 [cited 21.06.2007]. Available from: `http://www.ageia.com/`.

[3] J. Allard, S. Cotin, F. Faure, P.-J. Bensoussan, F. Poyer, C. Duriez, H. Delingette, and L. Grisoni. SOFA – an Open Source Framework for Medical Simulation. *Medicine Meets Virtual Reality (MMVR 15)*, 125:13–18, February 2007.

[4] Autodesk. Autodesk 3ds Max [online]. 2007 [cited 16.08.2007]. Available from: `http://www.autodesk.com/3dsmax`.

[5] Autodesk. Autodesk Maya [online]. 2007 [cited 16.08.2007]. Available from: `http://www.autodesk.com/maya`.

[6] Avid Technology. Welcome to Softimage – 3D Software Solutions for Games, Films, and Television Artists [online]. 2007 [cited 17.08.2007]. Available from: `http://www.softimage.com/`.

[7] D. Bachofen, J. Zátonyi, M. Harders, G. Székely, P. Früh, and M. Thaler. Enhancing the Visual Realism of Hysteroscopy Simulation. *Studies in Health Technology and Informatics*, 119:31–36, Jan. 2005.

[8] Blender Foundation. Blender [online]. 2007 [cited 17.08.2007]. Available from: `http://www.blender.org/`.

[9] C. S. Borrill, J. Carletta, A. J. Carter, J. F. Dawson, S. Garrod, A. Rees, A. Richards, D. Shapiro, and M. A. West. The Effectiveness of Health Care Teams in the National Health Service. Technical report, Department of Health, 2001. Available from: `http://homepages.inf.ed.ac.uk/jeanc/DOH-final-report.pdf` [cited 18.02.2008].

[10] M. C. Çavuşoğlu, T. G. Göktekin, and F. Tendick. GiPSi: A Framework for Open Source/Open Architecture Software Development for Organ Level Surgical Simulation. *IEEE Transactions on*

Information Technology in Biomedicine, 10(2):312–322, Apr. 2006.

[11] CISL. The MedSim-Eagle Patient Simulator [online]. 2007 [cited 17.08.2007]. Available from: `http://med.stanford.edu/VAsimulator/medsim.html`.

[12] S. Cotin, H. Delingette, and N. Ayache. Real-Time Elastic Deformations of Soft Tissues for Surgery Simulation. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):62–73, Mar. 1999.

[13] Crytek. CryEngine 2 Specifications [online]. 2002 [cited 17.08.2007]. Available from: `http://www.crytek.com/technology/index.php?sx=eng2`.

[14] H. Delingette. Toward Realistic Soft-Tissue Modeling in Medical Simulation. *Proceedings of the IEEE*, 86(3):512–523, Mar. 1998.

[15] DevMaster.net. 3D Game Engines Database [online]. 2007 [cited 17.08.2007]. Available from: `http://www.devmaster.net/engines/`.

[16] J. Dworzak and L. Gu. Combining progressive and non-progressive cutting for soft tissue surgery simulations. *International Journal of Computer Assisted Radiology and Surgery*, 2(Suppl 1):S163–S165, June 2007.

[17] Epic Games. Unreal Engine 2 [online]. 2004 [cited 17.08.2007]. Available from: `http://www.unrealtechnology.com/html/technology/ue2.shtml`.

[18] Epic Games. Unreal Engine 3 [online]. 2006 [cited 17.08.2007]. Available from: `http://www.unrealtechnology.com/html/technology/ue30.shtml`.

[19] D. Geer. Vendors Upgrade Their Physics Processing to Improve Gaming. *Computer*, 39(8):22–24, Aug. 2006.

[20] Havok. Havok [online]. 2007 [cited 17.08.2007]. Available from: `http://www.havok.com/`.

[21] id Software. id.sdk [The Code] [online]. 2007 [cited 17.08.2007]. Available from: `http://www.iddevnet.com/doom3/code.php`.

[22] Immersion Corporation. Immersion Medical [online]. 2007 [cited 17.08.2007]. Available from: `http://www.immersion.com/medical/`.

[23] Y.-J. Lim and S. De. Real time simulation of nonlinear tissue response in virtual surgery using the point collocation-based method of finite spheres. *Computer Methods in Applied Mechanics and Engineering*, 196(31-32):3011–3024, June 2007.

[24] J. Mackenzie, G. Baily, M. Nitsche, and J. Rashbass. Gaming Technologies for Anatomy Education. Online, May 2003. Available from: `http://www.virtools.com/news/pdf/2004/CARET.pdf` [cited 17.08.2007].

[25] S. Marks, J. Windsor, and B. Wünsche. Collaborative Soft Object Manipulation for Game Engine-Based Virtual Reality Surgery Simulators. In M. J. Cree, editor, *Proceedings of Image and Vision Computing New Zealand*, page 205–210, Hamilton, New Zealand, Dec. 2007.

[26] Mentice. Mentice [online]. 2007 [cited 17.08.2007]. Available from: `http://www.mentice.com/`.

[27] K. Montgomery, C. Bruyns, J. Brown, S. Sorkin, F. Mazzella, G. Thonier, A. Tellier, B. Lerman, and A. Menon. Spring: A General Framework for Collaborative, Real-time Surgical Simulation. *Studies in Health Technology and Informatics*, 85:296–303, 2002.

[28] select IT VEST Systems AG. Select-IT VEST Systems AG – medical science at your fingertips [online]. 2007 [cited 17.08.2007]. Available from: `http://www.select-it.de/`.

[29] Serious Games Initiative. Games For Health [online]. 2007 [cited 17.08.2007]. Available from: `http://www.gamesforhealth.org`.

[30] Simbionix. Simbionix, medical training simulators and clinical devices for MIS (minimally invasive surgery) [online]. 2007 [cited 17.08.2007]. Available from: `http://www.simbionix.com/`.

[31] Surgical Science. Surgical Science - Safer surgeons faster [online]. 2007 [cited 17.08.2007]. Available from: `http://www.surgical-science.com/`.

[32] J. Taekman, N. Segall, E. Hobbs, and M. Wright. 3DiTeams – Healthcare Team Training in a Virtual Environment. *Anesthesiology*, 107(A2145), Oct. 2007.

[33] Texas A&M University Corpus Christi. Pulse!! – The Virtual Clinical Learning Lab [online]. 2007 [cited 05.10.2007]. Available from: `http://www.sp.tamucc.edu/pulse/home.asp`.

[34] UnrealWiki. UnrealWiki: ASE File Format [online]. 2007 [cited 17.08.2007]. Available from: `http://www.unrealwiki.com/wiki/ASE_File_Format`.

[35] Valve Corporation. Valve Source Engine Features [online]. 2004 [cited 17.08.2007]. Available from: `http://www.valvesoftware.com/sourcelicense/enginefeatures.htm`.

[36] Valve Developer Community. Compiling Models [online]. 2007 [cited 17.08.2007]. Available from: `http://developer.valvesoftware.com/wiki/Compiling_Models_Basics`.

[37] Valve Developer Community. SMD file format [online]. 2007 [cited 17.08.2007]. Available from: `http://developer.valvesoftware.com/wiki/SMD_file_format`.

[38] Verefi Technologies. Verefi Technologies, Inc. [online]. 2007 [cited 17.08.2007]. Available from: `http://www.verefi.com/`.

[39] J. D. Westwood, R. S. Haluck, H. M. Hoffman, G. T. Mogel, R. Phillips, R. A. Robb, and K. G. Vosburgh. Highly-Realistic, Immersive Training Environment for Hysteroscopy. *Studies in Health Technology and Informatics*, 119:176–181, Jan. 2005.

[40] Wikipedia. id Tech 4 — Wikipedia, The Free Encyclopedia [online]. 2007 [cited 17.08.2007]. Available from: `http://en.wikipedia.org/wiki/Doom_3_engine`.

[41] B. C. Wünsche, B. Kot, A. Gits, R. Amor, J. Hosking, and J. Grundy. A Framework for Game Engine Based Visualisations. In *Proceedings of Image and Vision Computing New Zealand 2005*, Nov. 2005. Available from: `http://www.cs.auckland.ac.nz/~burkhard/Publications/IVCNZ05_WuenscheKotEtAl.pdf`.

[42] J. Zepp. GoodKarma Physics Mod Beta 4 [online]. 2005 [cited 16.08.2007]. Available from: `http://www.ataricommunity.com/forums/showthread.php?t=440477`.