Annals of DAAAM for 2009 & Proceedings of the 20th International DAAAM Symposium, Volume 20, No. 1, ISSN 1726-9679
ISBN 978-3-901509-70-4, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2009
Make Harmony Between Technology and Nature, and Your Mind will Fly Free as a Bird
DAAAM World Symposium

USING IRRLICHT ENGINE TO CREATE A REAL-TIME APPLICATION

POKORNY, P[avel]

Abstract: *Today, a computer programmer is a very lucrative business. Its work requires expert knowledge and high efforts. Therefore is programmers work exacting and it is too well paid. Simultaneously, it is on the look for a way to simplify and accelerate programmers work. Libraries and software engines offer the collection of the most used universal algorithms and functions in many applications. This paper describes, how to simple create a real-time application, based on the one engine - Irrlicht.*
Key words: *irrlicht, engine, graphics, application, visualization*

1. INTRODUCTION

In computer science, a software engine is relevant to the core of a computer program. Software engines control the functionality of the program, and are distinct from peripheral aspects of the program. For the layman audience is the term engine frequently presented like a piece of software compared to libraries, platforms or SDK.

The term software engine is frequently used in the many multimedia applications and computer games, where is the engine the base. There are many game engines that are designed to work on video game consoles and desktop operating systems such as Microsoft Windows, Linux, and Mac OS X. The core functionality typically provided by a game engine includes a rendering engine for 2D or 3D graphics, a physics engine or collision detection (and collision response), sound, scripting, animation, artificial intelligence, networking, streaming, memory management, threading, and a scene graph. The process of game development is frequently economized by in large part reusing the same game engine to create different games and multimedia applications.

Some game engines only provide real-time 3D rendering capabilities instead of the wide range of functionality required by games. These engines rely upon the game developer to implement the rest of this functionality or assemble it from other game middleware components. These types of engines are generally referred to as a "graphics engine," "rendering engine," or "3D engine" instead of the more encompassing term "game engine." However, this terminology is inconsistently used as many full-featured 3D game engines are referred to simply as "3D engines." A few examples of graphics engines are: Irrlicht, RealmForge, Truevision3D, OGRE, Crystal Space or Genesis3D Engine. Modern game or graphics engines generally provide a scene graph, which is an object-oriented representation of the 3D game world which often simplifies game design and can be used for more efficient rendering of vast virtual worlds.

Most often, 3D engines or the rendering systems in game engines are built upon a graphics API such as Direct3D or OpenGL which provides a software abstraction of the GPU or video card. Low-level libraries such as DirectX, SDL, and OpenAL are also commonly used in games and applications as they provide hardware-independent access to other computer hardware such as input devices (mouse, keyboard, and joystick), network cards, and sound cards. Before hardware-accelerated 3D graphics, software renderers had been used. Software rendering is still used in some modeling tools or for still-rendered images when visual accuracy is valued over real-time performance (frames-per-second) or when the computer hardware does not meet requirements such as shader support or, in the case of Windows Vista, support for Direct3D 10.

2. IRRLICHT ENGINE

Irrlicht engine is an open source high performance real-time 3D engine written and usable in C++ and also available for .NET languages. It is completely cross-platform and has all of the state-of-the-art features which can be found in the commercial 3d engines. (***, 2003)

The great benefit of this engine is the support 3D rendering via OpenGL, DirectX 8 and 9, OpenGL and internal software rasterizers. External renderers and windowing systems plugged in through simple interfaces, giving rise to community-made SDL, iPhone and SymbianOS devices. The engine comes with a library of standard material renderers, allowing fallback materials where user hardware is unable to handle advanced techniques. New materials can be added to the engine at run-time, allowing users to write their own as required. In addition to legacy fixed-function pipeline materials, programmable Pixel and Vertex Shaders, ARB Fragment and Vertex Programs, HLSL and GLSL materials are supported.

Irrlicht supports a lot of file formats. It will load and display 3ds Max files, Quake 2 MD2 Models, Maya .obj objects, Quake 3 .bsp maps, Milkshape3D objects, and DirectX .x files. Lights, cameras and 3D objects are managed as a tree of 'Scene Nodes', arbitrary groupable entities which are responsible for their own behaviour. Nodes can be managed by generic animators, by each other, or manually by the user. A large number of built-in node types exist and can be used together to make complex indoor and outdoor scenes, new nodes are trivial to make and can be added at runtime, many extra ones are provided by the community. (Wikipedia contributors, 2009)

Next, Irrlicht supports a skinnable 2D GUI. It supports many controls and the ability for users to plug in their own (or community made) custom widgets at runtime. Irrlicht's internal event system provides mouse, keyboard, joystick and GUI events without having to rely on additional libraries.

Irrlicht's provides support for simple collision detection including mouse picking, but users are advised that this is not intended as a replacement for a full featured Physics engine.

3. THE DESIGN OF APPLICATION

3.1 The real-time application architecture

The application is based on C++ language and object oriented programming. This conception is composed from a collection of cooperating objects. Each object is capable of receiving messages, processing data, and sending messages to other objects and can be viewed as an independent 'machine' with a distinct role or responsibility. (Liberty, 2004)

When the application start, at first it has to be created object Scene, which manages all object in the application. Then for the scene, objects Octree (a space representation in the scene) and Physics (a physics behavior – see below) are created there.

Until later, there are created all other objects, which we can find in the scene – dynamic objects, mans and vehicles. All of these objects are saved in XML file with the own features. Some of these objects can be differentiated on the basis of physics behavior. The simplified block scheme of the connections between the objects is shown on the figure 1.

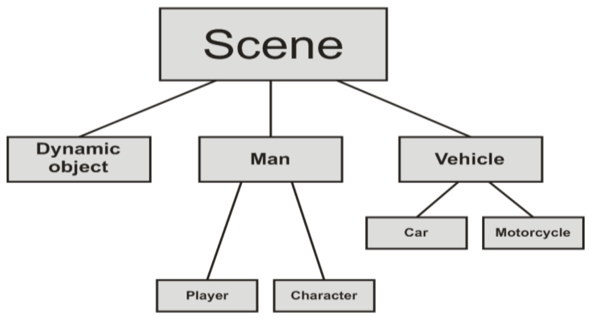

Fig. 1. The block scheme of connections between the objects in the application

3.2 Making 3D objects and textures

Each 3D application needs 3D models (meshes). Mesh model is composed by the vertices, edges and faces and the quantity of those entities specifies the quality of the each mesh model. Therefore, we must make the models with the suitable quality. The low quality models make our application with the low quality environment appearance. The high quality models give the more real environment appearance, but they need many memory and the system requirements.

Some of the models need animation keys. These keys specify their behavior in application. An example of the model with animation key is a character, which walks on the street.

All models and animation keys were created by Blender. Blender is the free open source 3D content creation suite, available for all major operating systems under the GNU General Public License. Its features (modeling, animation, rigging, rendering, shading, imaging and compositing) classify it to the best 3D graphics programs. (Pokorny, 2006)

All created models need textures. Some of them are painted, and the others are the modified photos. All of them are created by Gimp. Gimp is the acronym for GNU Image manipulation Program and it is an cross-platform application for making or modification of 2D raster images. Its possibilities and capabilities classify it to the best programs in this category. (Vybiral, 2008)

3.3 The physics engine

To realize a physics behavior in our application, we use Newton Game Dynamics physics engine, an integrated solution for the real time simulation of physics environments. This engine disposes of some benefits. It is small, fast, stable and easy to use. Its application programming interface (API) provides scene management, collision detection and naturally dynamic behavior. (***, 2000)

Newton Game Dynamics implements a deterministic solver, which is not based on traditional LCP or iterative methods, but possesses the stability and speed of both respectively. This feature makes our product a tool not only for the games, but also for any real-time physics simulation.

4. THE APPLICATION – USER VIEW

First, when is the application running, we can see a start screen with the menu. This menu contains three offers - the start application, settings and the end of the application. When we select Setting menu, we can set some possibilities - Graphics (resolution of application and color depth), Controls (change control keys for this application) and Car (change model of car, which we want to drive ant its color). When we start this application, the scene is loaded and the application is running.

In our application, we control own character in a city. We can walk with it on the street and meet other moving characters controlled by the computer, static objects (buildings, trees, lamps, etc.). Next, we can race with some cars. When we go near a car, we can get in, and then we can drive it in the all of the area. Any time, we can get out. One image of this application is shown on the figure 2.

We noted before, in this application we use the physics engine to simulate naturally behavior. There are simple collision detection (cannot go trough the solid objects), the gravity (falling down from on high) and more difficult collision detection (when the car crash into the small objects, like wall of the barrels, this wall is crashed).

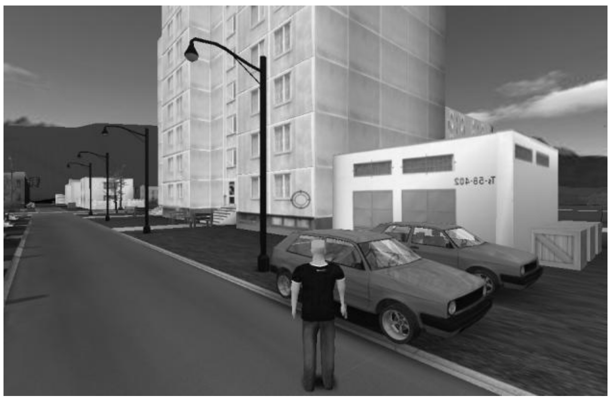

Fig. 2. The screenshot of the real-time application

5. ACKNOWLEDGMENTS

This work was supported by the Ministry of Education of the Czech Republic in the range of research projects No. MSM 7088352102.

6. CONCLUSION

In this paper there, the real-time application based on Irrlicht engine is described. Irrlicht engine is an open source real-time cross-platform engine, with the state-of-the-art features which can be found in the commercial 3d engines.

This application was developed on our faculty and this result evokes our next plans. First, we want to improve this application (better optimized graphics algorithms and thereby make better and superior 3D models and their textures) and we want to try some alternative algorithms. Second, it gives our experience, how to create other graphics real-time application, which can be used in some visualization applications.

7. REFERENCES

Liberty, J. & Jones B. L. (2004). *Sams Teach Yourself C++ in 21 Days,* Sams, ISBN 067232112, New York

Pokorny, P. (2006). *Blender–teach yourself 3D graphics* (in czech), BEN–technicka literatura, ISBN 80-7300-203-5, Prague

Vybiral, J. (2008). *Gimp – practical user handbook* (in czech), Computer Press, ISBN 80-251-1945-7, Prague

*** (2000) http://newtondynamics.com–Newton Game Dynamics-physics engine, *Accesed on:2009-07-08*

*** (2003) http://irrlicht.sourceforge.net–Irrlicht Engine–A free open source 3d engine, *Accesed on:2009-07-08*

***Wikipedia contributors. (2009) http://en.wikipedia.org/wiki/Irrlicht_Engine–Irrlicht Engine, *Accesed on:2009-07-08*