

Portfolio of Personal Projects

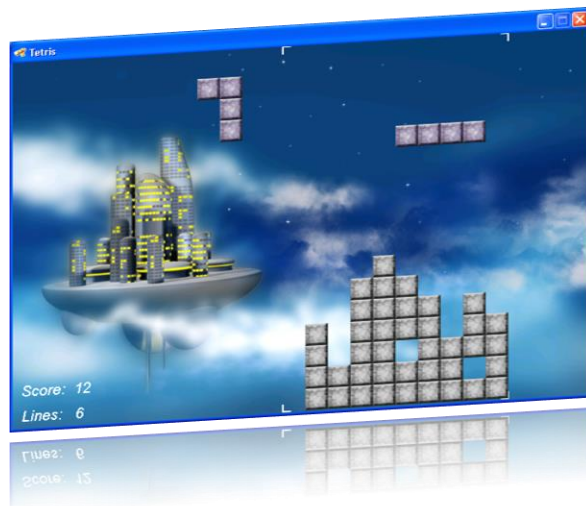
Table of Contents

First Major Step in Computer Programming	1
Bachelor's Thesis	2
Master's Thesis.....	6
Viewer of 3D Models	7
Low-Level 3D Renderer	8

First Major Step in Computer Programming

Before I started the bachelor's degree study in Computer Science, I had been developing my skills at programming by writing simple applications. For instance, at the high school, I created two video games using Borland C++ Builder IDE.

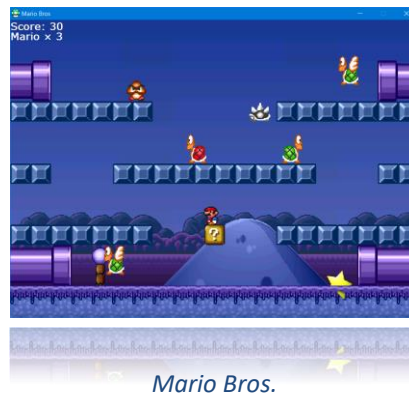
The first of them was a puzzle game where a user has to arrange the falling figures in an appropriate way to score points. The concept of this program was similar to the quite popular product called *Tetris* which had been released by Nintendo many years before for their video game consoles.



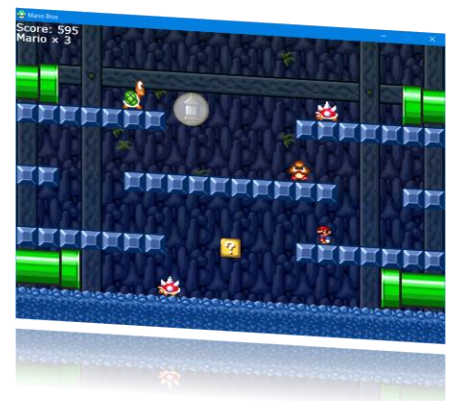
Tetris



The second engineering attempt was a 2D platform game created as a clone of well-known *Mario Bros.* but additionally based on some original ideas. This time, a player had to defeat many clever and dangerous enemies in order to advance the game. Hidden bonuses and traps additionally twisted the plot.



Mario Bros.



The two mentioned applications are available to download and test on your explicit request, but you must not use them for commercial purposes. Textures and soundtracks were used in these projects only with the aim of practicing programming. I created these games using the following technologies: C++ and Borland C++ Builder 6 Personal IDE.

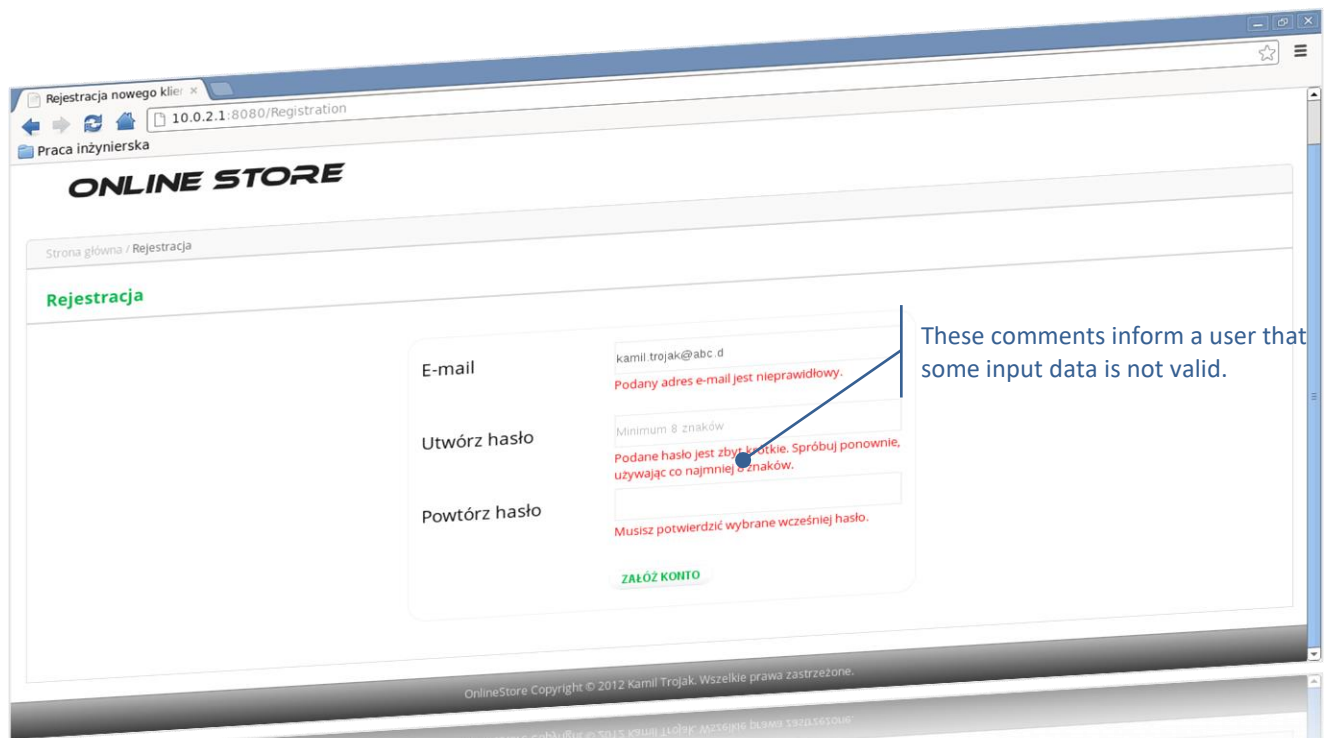
The thesis, which I prepared at the end of the bachelor's degree study, also contained a practical part. I had to program an online shop using the most advanced open source technologies. The range of duties included:

- designing and programming a web application,
- preparing a modern user interface,
- creating data structures and configuring a database,
- setting up Linux servers.

Using Java Enterprise Edition and MySQL relational database management system allowed me to create a sophisticated application which possesses a set of features that can be the fundamental part of every commercial online shop. Currently, my program supports only payment on receipt of the goods. The project does not contain a graphical user interface written in English, so the following pictures were labeled with appropriate descriptions.



The main website of the online shop for users who are not logged on.



These comments inform a user that some input data is not valid.

The process of registering a new client. The web form is validated by the appropriate server-side code.

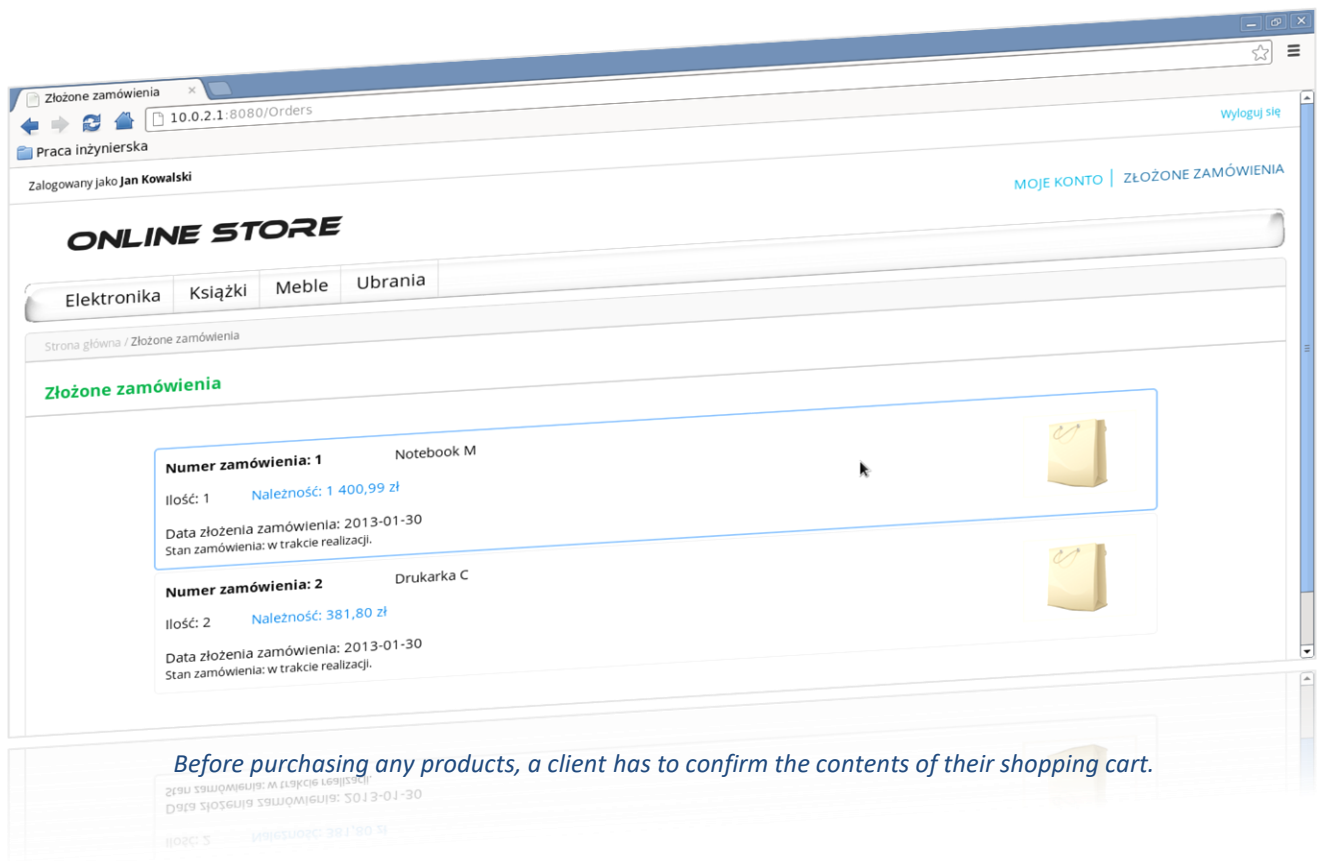


The red horizontal block reminds users to fill in their details (name, address etc.).

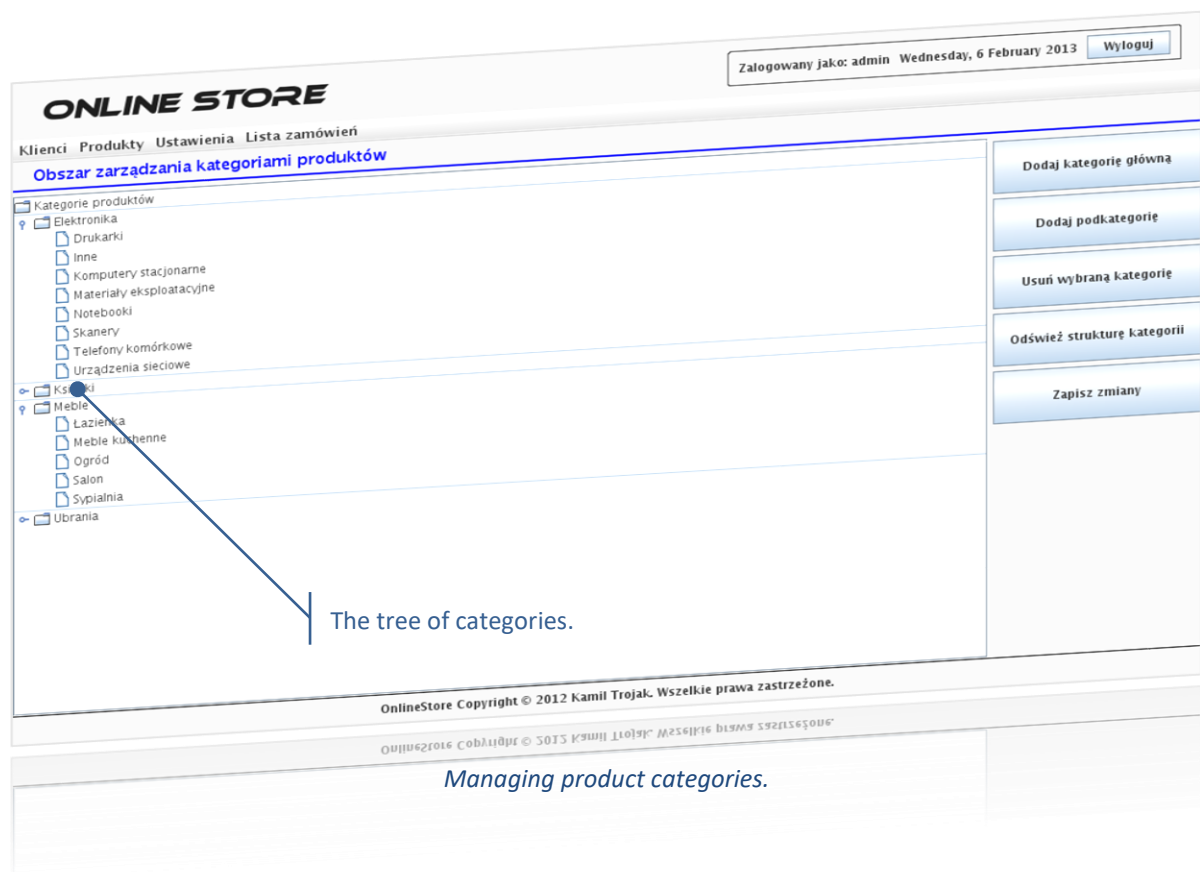
These buttons are used for managing the client account settings and orders.

The list of the recently viewed products by the current client.

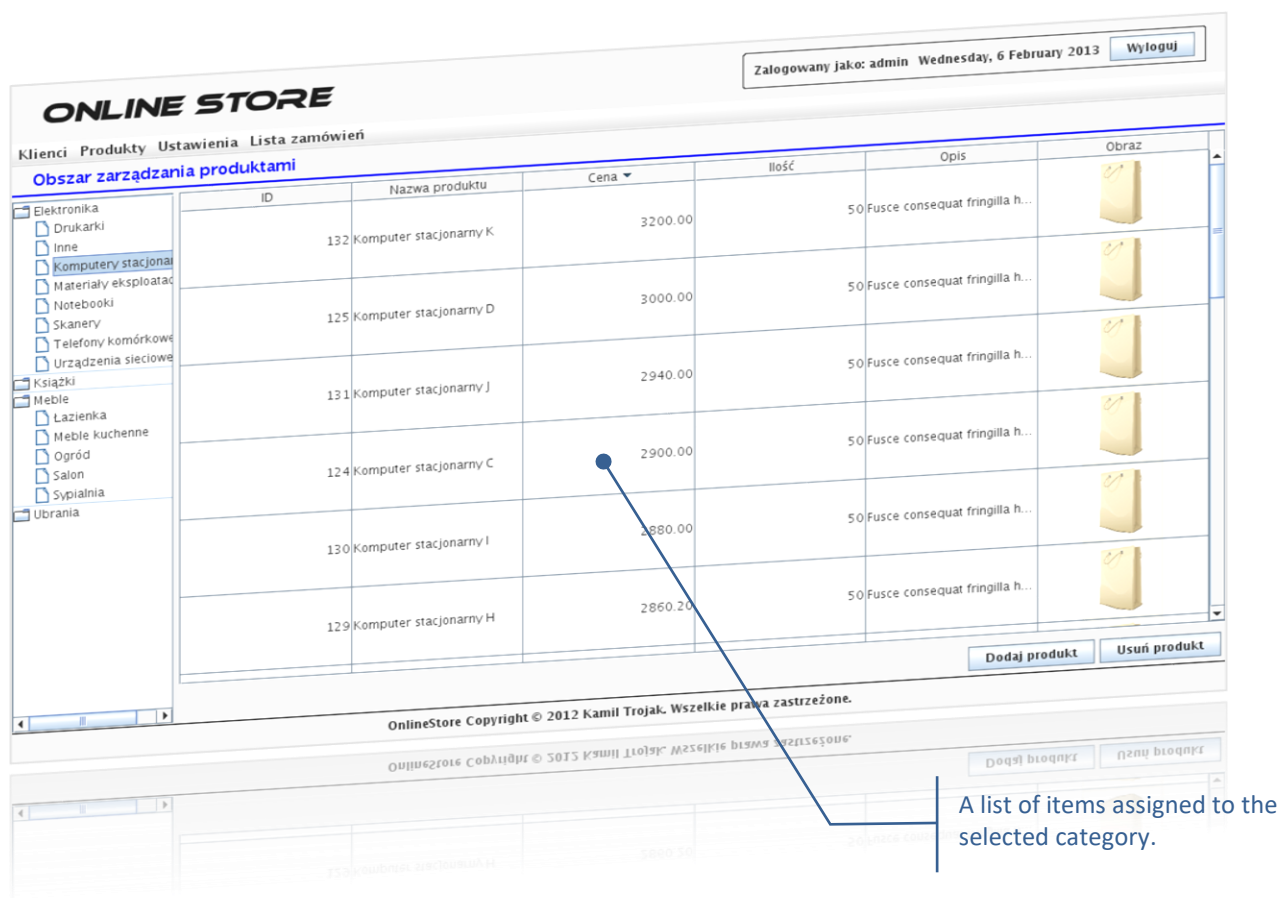
The logged on clients get a website with additional features like the ability to check their shopping cart or buy something.



The admin panel of my online shop was created as a Java applet in order to achieve higher security level and better performance. The following picture shows the part of my application that is used for creating a tree of product categories.



Among many important features, the admin panel supplies also a user-friendly tool to manage the set of products in our online shop. The picture below this paragraph shows a part of the applet that is used for creating new products and assigning them to the appropriate categories.



Every part of this project was created completely by me using only free and open source software. The data, which was used for filling in the structures of the database, is absolutely random and isn't connected with any real products.

Technologies which were used for creating this project:

- Java Standard Edition 6, Java Enterprise Edition 6, JavaServer Pages 2.1.
- Java Database Connectivity (JDBC) API.
- Java Applets, Swing GUI Components.
- HTML 5, CSS 3, JavaScript, jQuery, Google Web Fonts.
- Oracle Linux 6, OpenSUSE 12.1, GlassFish Server Open Source Edition 3.1, MySQL Community Edition 5.5.
- NetBeans IDE and free of charge graphics tools (GIMP, Inkscape).

Master's Thesis

My master's thesis was also strictly based on a practical programming work. A significant part of it was building a simulator of particles moving in a 3D virtual space. It was designed as a generic purpose framework which would allow to connect my solution with other applications. Despite the limited time and technical requirements, I managed to write a library that was able to construct a 3D environment and render flying particles. Additionally, the simulator was capable of detecting collisions of small pieces of virtual matter with other moving objects and simulating the bounce effect as a result.



Detecting collisions and changing the velocity vectors of particles is performed entirely by GPU. The customized shaders use the feature of OpenGL 3.3 called *Transform Feedback* to reduce the share of CPU in preparing the animation. It allowed me to dramatically increase the performance of the entire framework in comparison with other CPU based solutions.

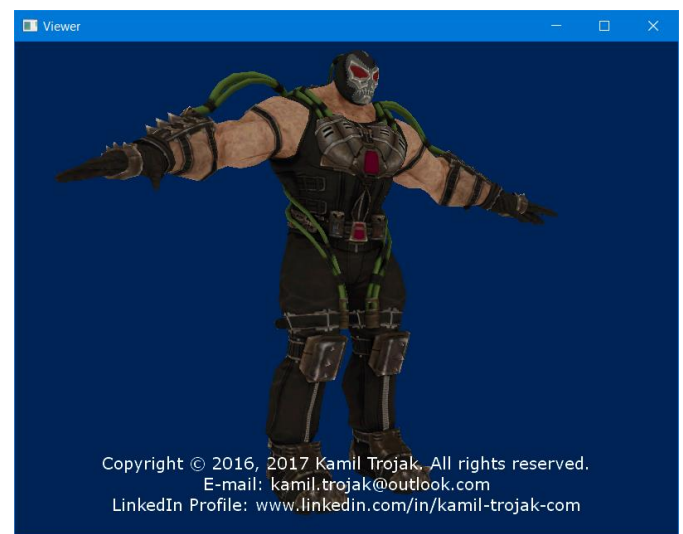
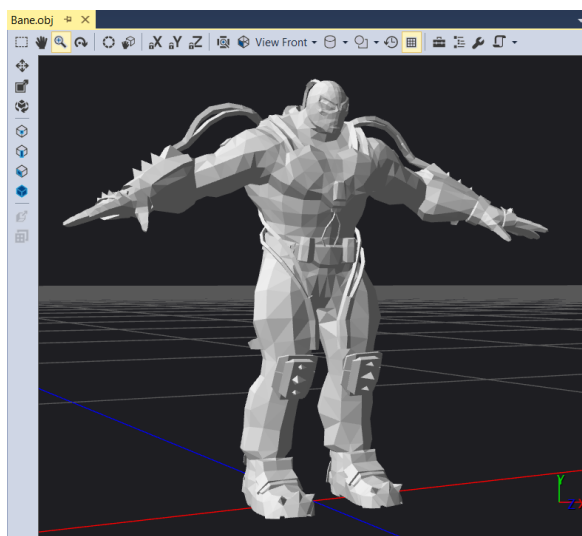
Technologies which were used for building this project:

- C++ and the Standard Library.
- OpenGL 3.3 Core Profile, OpenGL Shading Language 3.30 (GLSL).
- Qt Framework 5.3.2.
- Fedora 20 (The framework was designed only for Linux systems).

Viewer of 3D Models

After the master's degree study, I decided to sink my teeth into programming parsers of text files. For this reason, I created a handy application for viewing 3D models written in the commonly used OBJ format. It allowed me to boost my technical skills in terms of designing analyzers of text files.

Parsers and lexers, which I created on my own, can understand the syntax of Wavefront Object (OBJ) and Wavefront Material Template Library (MTL) file formats. They are also connected with a powerful system of exceptions which is responsible for signaling every deviation from the standard. If a process of parsing all necessary text files is finished successfully, an appropriate 3D model is presented to a user.

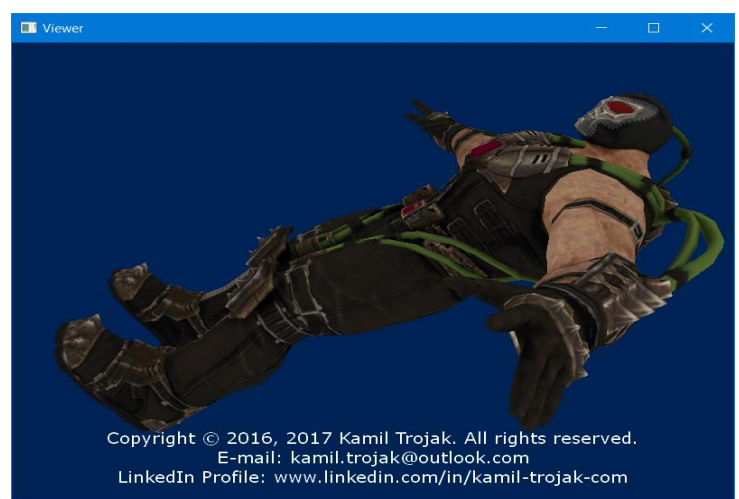


These two pictures show the same object loaded by Visual Studio 2015 (on the left, without a texture) and my application (on the right). The 3D mesh looks identical which means that my analyzer of the OBJ format works correctly.

Currently, the project supports the major part of the OBJ standard. Loading other models is possible by changing some parameters in the application. The 3D object and texture were used only for educational purposes and they are not an integral part of the project. My application is available to download and test only on request.

Technologies which I used:

- C++ and the Standard Library.
- Qt Framework 5.
- OpenGL 3.3 Core Profile, GLSL.



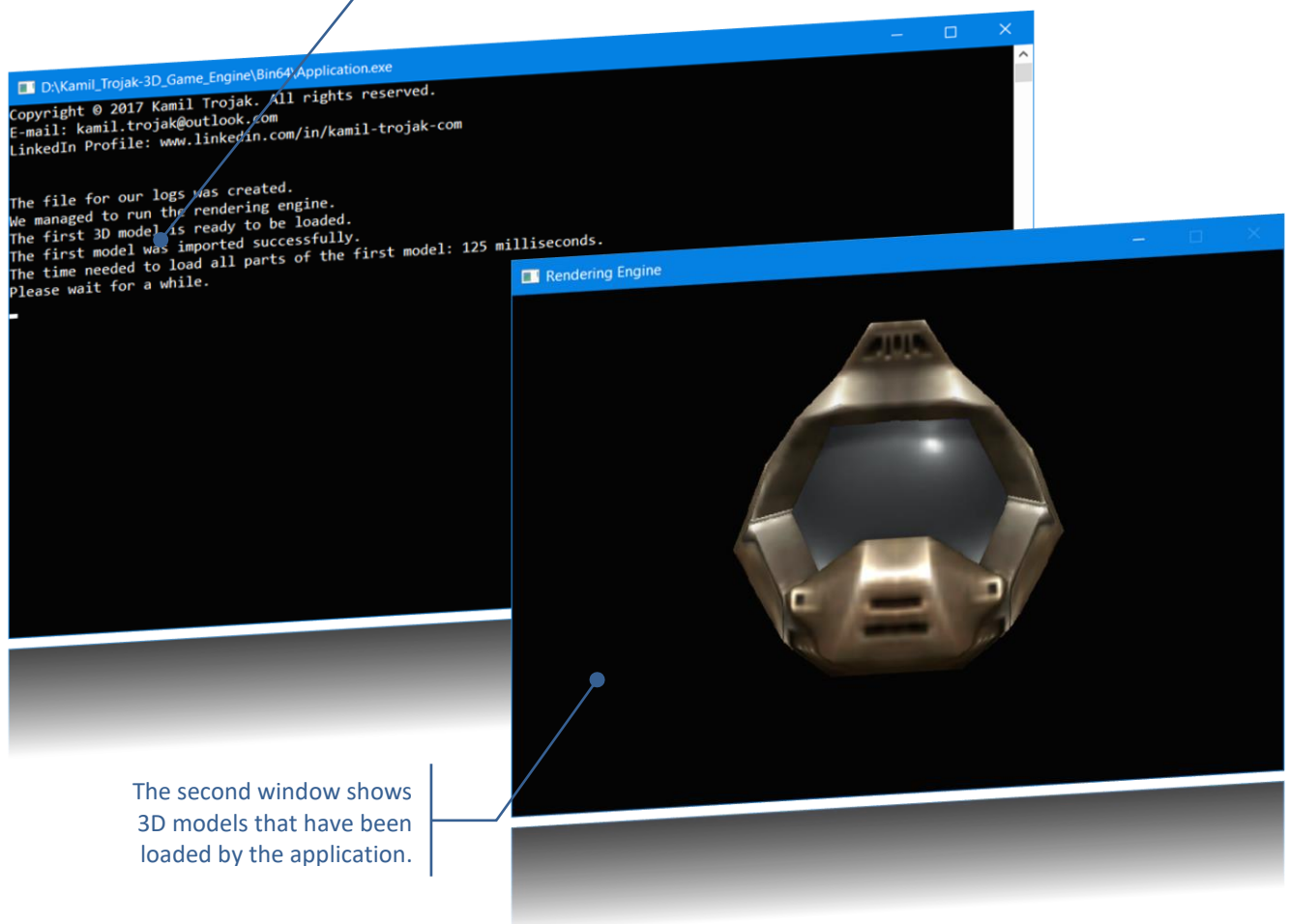
The application supports rotating and moving the camera.

Low-Level 3D Renderer

The last private project, which I have created, is a lightweight computer program that renders 3D models written in the MD3 format on a virtual scene. Such a way of storing data was introduced by *id Tech 3*, a game engine developed by *id Software*. It is also commonly used by many other open source applications, such as *OpenArena* and *Tremulous*.

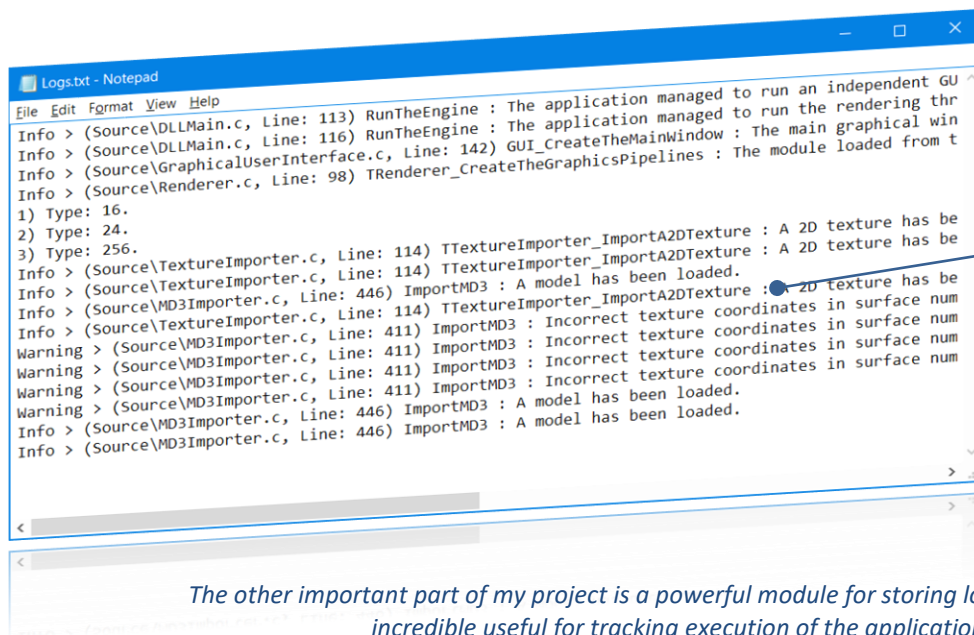
The crucial part of my application is the rendering engine which was written in C programming language using Microsoft DirectX 11 API. That module was additionally separated from the rest of the project as a dynamic-link library (DLL) in order to share the compiled code between other pieces of software. The entire program was designed as the basis for a small and efficient game engine and it could be a subject of further development.

The program creates two graphics windows. The text console is used for notifying users of events that have arisen and displaying some information about the performance.



The application is capable of displaying animations, arranging 3D objects on the scene and configuring the perspective view. What's more, elements of the virtual scene are loaded in the same time by a special thread pool. This approach has a beneficial effect on performance because it allows us to compute 3D images much faster than a single thread solution. The following pictures show several of the available animations of two 3D models.





A file with logs opened by notepad.exe.

The other important part of my project is a powerful module for storing logs which could be incredible useful for tracking execution of the application.

In order to speed up rendering of 3D images, I have decided to write my own library for matrix calculations in C language with the use of advanced *Intel Streaming SIMD Extensions* (SSE). It allowed me to create a much faster and lightweight implementation of such a module than any other free and open source solution available on the Internet. What's more, SSE instructions are also used in processing MD3 models which is a much more efficient approach than the one shipped by the original *id Tech 3* game engine.

The source code of this program was written entirely by me and it does not contain any pieces under the GNU General Public License. If you wanted to test the application on your own, you would receive the executable version on request. Keep in mind that you must not use it for commercial purposes. Disassembling and modification is also disallowed. The 3D models and textures were used only with the aim of practicing GPU programming.

The main technologies that were used in this project:

- C/C++ programming languages.
- Windows API (thread pools, critical section objects, condition variables, slim reader/writer locks, graphical user interface, etc.).
- COM interfaces.
- Microsoft DirectX 11.0 API.
- High Level Shading Language (HLSL).
- Windows Resource Files.
- Windows Imaging Component (WIC).
- Streaming SIMD Extensions (SSE).
- 32-bit and 64-bit Windows platform.
- MSBuild.