# Project Documentation

http://ocerus.sourceforge.net

February 10, 2011

# Contents

# Chapter 1

# Introduction

The Ocerus is a game engine suited for middle-sized 2.5D projects. 2.5D means the physics and most of the graphics are in two dimensions, but some 3D objects are supported as well. The main focus lies on the editing tools integrated to the engine. So every change is immediately visible in the game context and can be tested in a real time. The engine takes care of rendering, physics, customization of a game object behavior by scripts, resource management and input devices. The game objects are easily expandable using new components or scripts.

# Chapter 2

# Similar Projects

A tool for game development is nothing new. Every development studio has its own tools for creating certain aspects of the game, but only few of those are generic enough to make them usable for other people as well. An example of these might be Unreal Technology, which provides very complex and sophisticated tool for AAA developers. Unfortunately it is also very heavyweight and expensive, so it's not so useful for small development studios. On the other side of the spectrum, there are simple tools likeGame Maker, which are very easy to use but lack a lot of features and performance and generally are considered to be targeted at hobbyists instead of professional developers.

Our project attempts to fit into the middle of the spectrum. It is not too simple but at the same it is not too complex either. We do this mostly by putting our hands away from the 3D and focusing on 2D (or 2.5D) instead. In this field, the biggest competitor is probably Unity, which grew into a very powerful tool during the last year. It surpasses ours in many ways but for a price, of course: there is no source code available, and it's targeted to 3D games which makes its use for 2D somewhat complicated.

# Chapter 3

# Project Timeline

There is a project development timeline in the following list:

- 01/2010 - reviewing existing code, use cases, unit test framework, library structure, log system, utilities
- 02/2010 - script system, graphic system, entity system, 2D graphic components
- 03/2010 - entity messages, camera and script component, profiler, memory management
- 04/2010 - gui system, entity picking, dynamic properties
- 05/2010 - rendering, gui console, resource system, input system, layer manager
- 06/2010 - physic and model components, string manager, project management, editor
- 07/2010 - documentation, poligon collider component
- 08/2010 - loading screen, resource refreshing, hierarchy window
- 09/2010 - configuration, sprite animation, testing, sample game
- 10/2010 - GUI layout component, sample game GUI, shortcuts, documentation, game deploy
- 11/2010 - testing, reviewing documentation
- 12/2010 - bug fixing, reviewing documentation
- 1/2011 - reviewing documentation

# Chapter 4

# Conclusion

This project had two main objectives: to make game objects generic by constructing them from components and to make it possible to edit the game while it's running. We find these two things crucial during the development of a game in a small team. That said we believe we've successfuly put these features into the final application.

However, everything wasn't working as smooth as we intended. The main source of problems was our decision to use CEGUI for the GUI of the editor. CEGUI is designed to work inside the game itself and it turned out the editor has very different needs. This led to a lot of troubles; be it a poor performance or missing features that we had to implement by ourselves. It caused the delay of the first version that was able to display something. Until then we had to work on the engine itself without being able to debug features visually.

There are also things we were not able to implement due to the time limit set on the project. This includes sound and music support and certain performance optimizations. Also parts of the engine would need porting to make them run on game consoles. Due to the need of being an officially registered developer at the consoles' manufacturers we opted not to focus on this. And as it was mentioned before the editor's GUI is currently far from perfect due to the bad choice of the GUI system. So it would make sense to rewrite it to use another library; Qt being an example.