MOONGATE: MULTIPURPOSE RTS ENGINE FOR WIN-DOWS AND XBOX 360

Rudolf Kajan
Master Degree Programme, FIT BUT
E-mail: xkajan03@stud.fit.vutbr.cz

Supervised by: Adam Herout
E-mail: herout@fit.vutbr.cz

ABSTRACT

This paper deals with implementation of a real-time strategy engine. The implemented engine should support along the Windows platform also currently most powerful gaming console – Xbox360, thus should provide easy access to console's hardware and exploit it for creation of strategy games, but also for development and use of shaders and allow visualization of AI algorithms, which are very tightly bound to this game genre.

1. INTRODUCTION

Real-time strategy (RTS) is the genre of computer games that encompasses war games that happen in real-time. "Real-time" means that there is a continuous flow of time in the game world, so that "thinking on your feet" and responding quickly to arising situations is important. "In a typical RTS, the goal is for the player to collect resources, build an army, and control his units to attack the enemy."[1] But besides being one of the most popular game genres, papers such as [2], [3], or [4] suggest, that RTS games have more to offer than simple entertainment. This work presents MoonGate – multipurpose and multiplatform RTS game engine.

2. ANALYSIS

Rather than being just a single, static RTS game, MoonGate is a RTS *engine*. Users define the game in form of scripts, which describe all unit types, structures, and their interactions. MoonGate was designed and implemented with following goals in mind:

☑ **To be easily understandable, multiplatform, open source RTS starter kit for independent game developers** (indie developers), whether they are individuals or small teams. MoonGate is compatible with Xbox 360 and Windows platforms, so developers can target both of these platforms at the same time, without any porting, which is expensive and time consuming.

☑ **Serve as a test bed for shaders written in HLSL programming language**. MoonGate's rendering system exploits both material shaders (normal mapping, color replacement) and post screen shaders (bloom, glow). This way it is possible to adjust input values for shaders in real time, immediately see results and compare performance of shaders on different platforms.

☑ **Provide visualization for artificial intelligence algorithms**. RTS games can be viewed as simplified military simulations. Several players struggle over resources scattered over terrain by setting up an economy, building armies, and guiding them into battle in real-time. RTS games offer a large variety of fundamental AI research problems, unlike other game genres studied by the AI community, for example decision making under uncertainty, spatial and temporal reasoning, collaboration and other. [1]

3. MOONGATE AND THE XNA FRAMEWORK – IMPLEMENTATION

In order to achieve multiplatformity, MoonGate uses the XNA Framework. The XNA Framework is based on the native implementation of .NET Compact Framework for Xbox 360 development and .NET Framework on Windows. MoonGate game engine is built on top of XNA Framework, which acts as a lowest layer of game engine, providing low-level functionality and connection directly to hardware.

3.1. COMPONENTS

MoonGate is based on GameComponents – highly reusable pieces of code, which are able to communicate among themselves through well defined interfaces. What puts MoonGate ahead of general game engines is availability of components with game logic that provide necessary functionality directly out-of-the-box, thus enabling rapid development in an area user is really interested in – whether it is development of shaders, AI algorithms or game logic. Most interesting components available in MoonGate are:

Terrain – one of most complex components. It is based on height map, where every pixel represents height in one terrain vertex. To render only visible parts of terrain every frame, terrain is divided into smaller patches with bounding boxes (bboxes), which are stored within quad tree data structure. Every frame quad tree is recursively searched and only patches with bboxes colliding with camera frustum are rendered. This way only ~15% of high detailed terrain is drawn every frame. Terrain supports bump mapping and texture blending with up to 4 layers of textures through custom HLSL shader.

Hierarchy of objects – enables fast creation and immediate use of game objects. Game objects are either static objects that fill environment (trees, rocks,…), or controllable objects (units), in which case they support execution of orders (path finding and movement, following, attack). To achieve speed and memory efficiency, design patterns are heavily used (Factory, Prototype,…).

Bones Animation - provides out-of-the-box support for animation of bones inside game objects. Implemented bones animation system is based on controllers – small pieces of code that manipulate with bones in real time. This system is compatible with most widely used 3D modeling and animation packages like 3D Studio Max, Maya, XSI or Blender.

Material and rendering system – allows to render objects with powerful HLSL shaders, thus producing visual quality comparable to modern commercial RTS games.

Map editor – powerful, yet very user-friendly editor allow fast creation and modification of environment. Editor supports terrain painting with 4 textures, variable brush size and placement of objects on the surface of terrain.

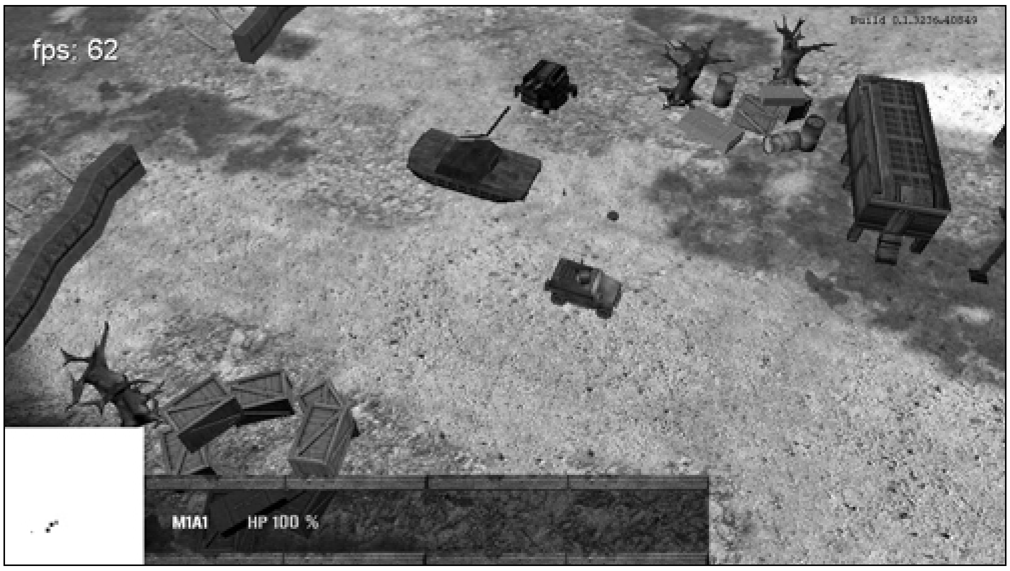

Figure 1: Screenshot from MoonGate.

4. CONCLUSION

In this paper MoonGate – multiplatform, open source, RTS game engine was presented as a tool of choice not only for indie game developers. Its flexibility allows beside creation of a game in one of the most popular game genres also exploitation of Xbox360's powerful hardware for shader development and visualization of AI algorithms, which are very tightly bound to this game genre.

Some parts of MoonGate are still under development – because, as all game developers would certainly agree, there is no such thing as a 100% complete game or game engine. There is always a way to make current game (engine) a little bit faster, to allocate lower amount of system resources or to make it more visually attractive. In near future improvements in material system and in map editor will be implemented.

The amount of attention that MoonGate attracted after revealing on main XNA community forums "creators.xna.com" and "ziggyware.com" clearly shows, that individuals and even small independent teams are interested in MoonGate, and that it is definitely a step in right direction.

REFERENCES

[1] Rabin, S.: Introduction to Game Development, Charles River Media 2005, ISBN 1-58450-377-7

[2] Buro, M. Real-Time Strategy Games: A New AI Research Challenge. In *IJCAI*. [s.l.] : [s.n.], 2003. p. 1534-1535.

[3] Sharma, M., et al. Transfer Learning in Real-Time Strategy Games Using Hybrid CBR/RL. In *IJCAI*. [s.l.] : [s.n.], 2007. p. 168.

[4] Orkin, J.: Applying Goal-Oriented Action Planning to Games. In *AI Game Programming Wisdom II*. Charles River Media, 2003.