Castle Game Engine - Game Engine Using X3D as a Scene Graph

Michalis Kamburelis*
Michalis Kamburelis Software

Figure 1: *Screens from games done using Castle Game Engine*

Abstract

Castle Game Engine (`http://castle-engine.sourceforge.net/`) is a modern, open-source game engine closely connected with the X3D standard. It uses X3D as a scene graph, and also as it's main 3D and 2D interchange format. In this poster we would like to highlight some engine architectural advantages.

Keywords: game, game engine, x3d, vrml, scene graph

1 X3D Nodes as a Scene Graph

Loading any 3D or 2D asset to the engine results in a graph of X3D nodes.

Developer can freely animate and operate on the loaded scene graph, making the content fully dynamic. It can be processed using X3D events or by directly modifying X3D nodes and fields.

X3D worlds can be composed and processed. Many 3D scenes can be combined, and can be *animated or build or modified*. Their behaviour can be coded as scripts (inside X3D) or using powerful object-oriented API in *modern Object Pascal*.

Moreover, *many other data formats can be loaded and are converted seamlessly to the X3D scene graph*. For example, *Spine 2D animations* or *Collada 3D assets*. Everything cooperates in the engine, as it's all a graph of X3D nodes after loading.

2 Accessible for Everyone

The engine is accessible to all content creators, regardless of the tool they prefer — since virtually every 3D tool allows to export it's data to X3D or VRML.

For X3D authors, it's trivial to just use X3D to build a 3D or 2D world.

*michalis.kambi@gmail.com

3 Procedural Generation

Developer can build an X3D graph by code, from scratch or by compositing modeled 3D parts. This allows to generate 3D worlds using a variety of interesting techniques, e.g. generate terrains using a smooth noise, or generate cities using grammar-based procedural generation algorithms.

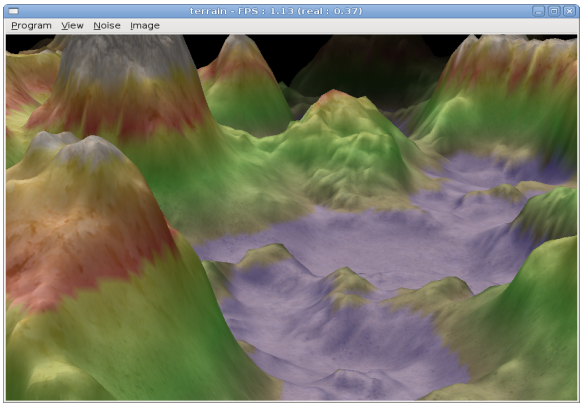

Figure 2: *Generated valley in the mountains.*

4 Streaming

As we can load and save any X3D graph (to XML or classic encoding), the developer has tools to save the 3D world easily.

5 Portable

The engine is truly portable, supporting various standalone (Windows, Mac OS X, Linux) and mobile (Android, iOS) platforms.

6 Modern Graphic Effects

The engine provides a plethora of modern graphic effects and techniques, like:

- *Bump mapping*, in various variants (like steep parallax mapping with self-shadowing).

- *Shadows*, by shadow maps or shadow volumes.
- *Custom shaders*, by enhancing or overriding engine rendering.
- *Mirrors*, by cubemaps or flat texture reflections.
- *Rendered textures*, like from a security camera.
- *Custom viewports*.
- *Screen-space effects*, programmable by a shading language.
- *3D textures*, for example for fancy dense fog effects.

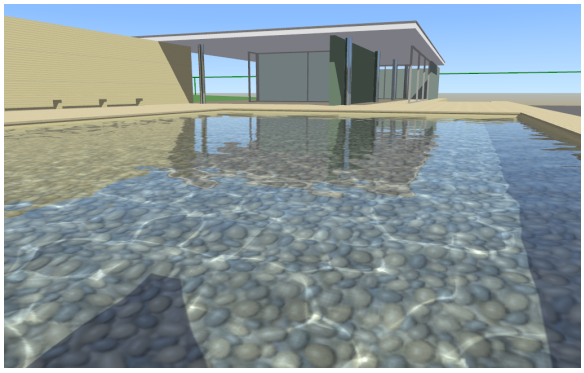

Figure 3: *Caustics in a water pool, rendered in real-time, using a couple of shader and texture tricks.*

Everything is expressed using X3D nodes, with some nodes and fields added as extensions to the X3D standard. The X3D extensions used in the engine are documented on `http://castle-engine.sourceforge.net/x3d_extensions.php`. Compositing shaders extensions are described in full in our paper [Kamburelis 2011]. Shadow mapping extensions are presented in [Kamburelis 2010].

7 3D and 2D

Just like X3D, the engine is suitable for 2D games as well as 3D.

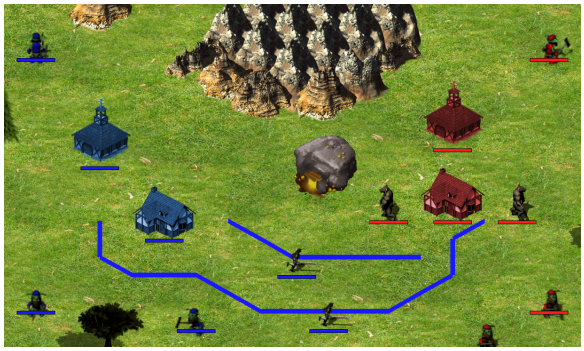

Figure 4: *Real-time strategy game with isometric view.*

8 High-level 3D World Management

- Comfortable *scene manager* for 2D or 3D game worlds.
- Movement planning by *waypoints and sectors in 3D*.
- Ready to use classes implementing *artificial intelligence* (AI). Implementing custom AI, or extending the existing one, is very easy. Any 3D object can move in the 3D world, knowing it's surroundings.
- *3D sound*, integrated with *X3D Sound component*.

9 Animations

- Animations can be interactive, using standard X3D interpolator mechanisms.
- Humanoid animations following *H-Anim* standards are also supported (both skeletal and skinned animations are supported).
- Animations can also be *baked* to have some animation frames precalculated in memory, making rendering them as fast as rendering static models (at the cost of additional memory usage).

Figure 5: *Animated knight walks and attacks.*

10 Developing in Object Pascal

The developers can use a full-featured object-oriented API to load and manage the game. The engine is written using modern *Object Pascal* language.

- Modern hybrid programming language, with everything you expect — units, classes and interfaces system, generics, rich runtime library, tools etc.
- Compiled to native, optimized code.
- Type-safe.

11 Complete with Tools

One of the most important engine tools is *view3dscene*, a full-featured browser for VRML/X3D, and a viewer for many other 3D and 2D formats supported by the engine. It's great to test your assets, to see how they are rendered by the engine. See `http://castle-engine.sourceforge.net/view3dscene.php`

The engine includes a *build tool* that handles the whole process of compilation and packaging the game to a standalone or mobile target (for example, to `apk` on Android). See `https://sourceforge.net/p/castle-engine/wiki/Build%20tool/`.

12 Well Documented

As expected from any good game engine, we include all the necessary documentation. *Tutorial*, *guide to creating game data*, *engine reference* and more documentation are available at the engine website on `http://castle-engine.sourceforge.net/`.

References

KAMBURELIS, M. 2010. Shadow maps and projective texturing in X3D. In *Proceedings of the 15th International Conference on Web 3D Technology*, ACM, New York, NY, USA, Web3D '10, 17–26.

KAMBURELIS, M. 2011. Compositing Shaders in X3D. In *Theory and Practice of Computer Graphics*, The Eurographics Association, I. Grimstead and H. Carr, Eds.