4.4 development(s) – gaming is a waste of time

The title of this section is borrowed from a lecture given for the VU computer science student association (STORM[1]), indeed, entitled *gaming is a waste of time*. This admittedly provocative title was on the one hand meant to emphasize the notion *waste of time*, since according to some of my collegue staff members my involvement in game development and multimedia technology was a mere waste of time, from some (from my point of view) obscure academic perspective. On the other hand, it (that is the title) also raised a more serious issue. Not being a game player myself, I do (in some sense) consider game playing a waste of time. Not that I deny the learning or entertainment aspects of games. On the contrary! Yes, as a passing of time, I prefer to keep myself busy with the construction of games, that is the creative and technological aspects of game development. And, likewise, I advise my students to do so.

When I was asked, in an alumni interview with the magazine of CWI[2], whether I believed in Second Life, my answr was simply: *I believe in nothing*! I take Second Life as an object of study, not in the last place because it has recently become so surprisingly popular. Yet, to be fair, Second Life has, after closer inspection, also technological merits of its own right.

In VUSL, we wrote: from a technical perspective, Second Life offers an advanced game engine that visitors and builders use (implicitly) in their activities. For essential components of game engine(s), we refer to section 11.1. In the following table, we give a brief comparative technical overview of, respectively, the Blaxxun Community Server (BlC), AlphaWorld (AW), the open source Delta3D engine (D3D), the Half Life 2 Source SDK (HL2), and Second Life (SL).

	BlC	AW	D3D	HL2	SL
in-game building	-	+	+/-	-	++
avatar manipulation	+	++	+/-	+	++
artifical intelligence	+	-	+/-	+	-
server-side scripts	+	-	+/-	+	++
client-side scripts	++	-	+/-	+	-
extensibility	+	-	++	+	+/-
open source	-	-	++	-	+/-
open standards	-	-	+/-	-	+/-
interaction	+/-	+/-	++	++	+/-
graphics quality	+/-	+/-	++	++	+
built-in physics	-	-	+	++	+
object collision	-	-	++	++	+
content tool support	+/-	-	++	+	-

Obviously, open source engines allow for optimal extensibility, and in this respect the open source version of the SL client may offer many opportunities. Strong points of SL appear to be *in-game building*, *avatar manipulation*, and in compari-

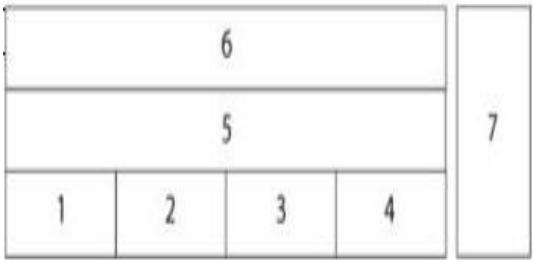
[1] www.storm.vu.nl
[2] www.cwi.nl

son with BlC and AW *built-in physics* and *object collision detection*. Weak points appear to be *content development tool support*, and especially in comparison with D3D and HL2 *interaction*. For most types of action-game like interaction SL is simply too slow. This even holds for script-driven animations, as we will discuss in the next section. In comparison with a game as for example Age of Empires III[3], which offers in-game building and collaboration, Second Life distinguishes itself by providing a 3D immersive physics-driven environment, like the 'real' game engines.

Although we do not intend to realize Clima Futura in Second Life, we actually use *flash* to reach an audience as wide as possible, as a pilot parts of the game could fruitfully be realized in the VU virtual campus in Second Life, in particular the search for knowlegde, that is looking for an expert in a particular area of (climate-related) research. A similar quest was implemented in our Half Life 2 based game VULife, VULife, where the player had to visit nine information spots, which resulted in displaying in a HUD nine square matrix the location of a hidden treasure, which was then actually the power to use arms. Technical issues in realizing Clima Futura in Second Life are support for ranking, as well as meta-information with respect to locations where relevant information can be found, which may be realized with the techniques indicated in section 2.4.

In the beginning, we wrote in Climate, we envisioned the realization of our climate game as a first-person perspective role-playing game in a 3D immersive environment as for example supported by the Half Life 2 SDK, with which we gained experience in creating a *search the hidden treasure*[4] game in a detailed 3D virtual replica of our faculty. However, we soon realized that the use of such a development platform, would require far too much work, given the complexity of our design. So, instead of totally giving up on immersion, we decided to use *flash* video[5], indeed as a poor-man's substitute for real 3D immersion, which, using *flash*[6] interactive animations, has as an additional benefit that it can be used to play games online, in a web browser. Together with the Flex 2 SDK[7], which recently became open source, *flash* offers a rich internet application (RIA) toolkit, that is sufficiently versatile for creating (online) games, that require, in relation to console games or highly realistic narrative games like Half Life, a comparatively moderate development effort. To allow for component-wise development, we choose for a modular architecture, with four basic modules and three (variants) of integration modules, as indicated below.

[3] www.ageofempires3.com
[4] www.cs.vu.nl/∼eliens/game
[5] www.adobe.com/products/flash/video
[6] www.adobe.com/devnet/flash
[7] www.adobe.com/products/flex/sdk



Clima Futura architecture

module(s)

1. climate model(s) - action script module(s)
2. game play interaction - event-handler per game event
3. video content module - video fragment(s) and interaction overlays
4. minigame(s) - flash module(s) with actionscript interface
5. Clima Futura - integration of modules 1-4, plus server-side ranking
6. adapted versions – educational, commercial
7. multi-user version –with server-side support

In addition, we would like to develop a facility that allows players not only submit their own video material, but also to build or modify their own minigames, which might then be included in the collection of mini-games provided by Clima Futura. This, however, requires apart from a participatory (web 2.0) web-site, an appropriate game-description format, which we will discuss in section 11.4.

collada – gluing it all together

The wide variety of formats and tools for content production has been a stumbling block for many projects. How to find a unified format for digital content creation (DCC), so that content may be easily reused across projects and tools? A promising attempt in this direction is the *collada* initiative, Collada. The standard proposed in Collada is meant to serve as an intermediate or interchange format for interactive (multimedia) applications, such as games, which can be characterized as:

interactive application(s)

- realtime interaction – providing information
- content navigation – providing view(s)

Interactive multimedia applications have as a common property that, in contrast for example to computer graphics (CG) movies, everything must be available, that is computed, in real time. The intermediate format (*collada*), presented in Collada, is an XML-encoding of the various elements that may result from the content pipeline, that is the workflow of (digital) content creation, of a (multimedia) project, including:

collada[8]

[8] www.collada.org

- document(s) – *schema, asset, library, technique, ...*
- geometry – *array, accessor, meshes, vertices, polygons, ...*
- scene(s) – *material, light, optics, camera, imager, ...*
- effect(s) – *shader, profiles, techniques, pass, ...*
- animation(s) – *sampler, channel, controller, skin, morphing, ...*
- physics – *collision, equation, rigid body, constraints, force, ...*

The list above gives an indication of what (description) primitives the *collada* standard offers, to facilitate the exchange of (content) information and to promote re-use across tools and platforms.