11. game technology for serious applications

immersion does not require illusion but involvement

learning objectives

After reading this chapter you should have an idea how to approach the development of a moderately complex game, and you should also be able to discuss the notion of immersion and argue why using game technology is relevant for serious applications.

Game playing is fundamental to human life. Not only for entertainment, but also to acquire the necessary skills for survival. Game playing can take a variety of forms, but nowadays the dominant game paradigm is undoubtedly the interactive video game, to be played on a multimedia-enhanced PC or game console. Currently, games are being (re) discovered in the academic field, another serious areas of society, as excellent means for both the transfer of knowledge and, perhaps more importantly, for attitude change.

In this chapter we will look at the various issues in developing a game, and more specifically, in section 11.2 at requirements for a promotional game for our faculty and the issues that came up when giving a masterclass game development for high school students using this game. Finally, we will sketch the history of immersive systems, in particular panoramas, in section 11.3, and we will discuss how immersion is to be realized in a game context.

1

11.1 constructing a game

Since ancient times, games have been an essential part of culture. According to Huizinga, game playing is not merely meant for entertainment, but is (one way or another) fundamental for our survival:

> *... in the game we confront a function of the living creature which cannot be determined either biologically or logically ...*

As a terminological aside, in the english language there is a distinction between *play* and *game*. The difference seems to be primarily that a game is subject to additional rules, which determine the success in the game, whereas play seems to be less constrained and more phantasy-driven. In addition, games seem to have a competitive element, a challenge, where you may either win or lose.

Anyhow, as observed in Kress and van Leeuwen (1996):

visual culture

> *games are an increasingly important element in our visual culture.*

Nowadays, with rapid advances in technology, the game industry is said to be overtaking both the film industry as well as the software industry, in terms of total budget. This situation has a somewhat paradoxical effect on game development, however. On the one hand, it is more easy to develop a game, considering the great variety of tools and technologies that are available. On the other hand, to make a successful game has become more difficult, since the competition is much more harsh and does not allow for amateurism. Nevertheless, game development is worthwhile in itself, even if it does not lead to economic success. As a computer science or multimedia student, you should at least be familiar with the process of game development, and gain experience with (a selection of) available game technology.

game engine component(s) Game development covers a wide range of activities, and although preferably done in a team, requires a correspondingly wide set of skills, artistic as well as technical. When speaking about *game programming* it is important to make a distinction between:

- game play programming
- game engine programming

Game play programming, usually, consists of scripting the behavior of characters or elements of a game scene, using a scripting language such as, for example, Lua[1]. It may considered to be part of game design, and certainly requires an artistic approach.

In contrast, game engine programming is a much more technical activity and concerns the development of functional components of a *game engine*, which according to Sherrod (2006) encompass:

game engine component(s)

- rendering system – 2D/3D graphics
- input system – user interaction
- sound system – ambient and re-active
- physics system – for the blockbusters

[1] www.lua.org

- animation system – motion of objects and characters
- artificial intelligence system – for real challenge(s)

Although it is possible to build one's own game engine using OpenGL or DirectX, or the XNA[2] framework built on top of (managed) DirectX, in most cases it is more profitable to use an existing game engine or 3D environment framework, since it provides the developer with a load of already built-in functionality. In section 4.4, a brief comparative overview of game engine(s), that may be used to built virtual worlds, has been given. This overview did not include the flex 2 SDK, nor related flash engines, that may be used for the development of online (flash) games.

2

elements of game design Game development projects are similar to multimedia projects, or for that matter software development projects, in that they may be sub-divided in phases covering the definition of requirements, the actual development, as well as testing and delivery. However, instead of looking at these phases, or trying to explicate the differences between software development and game development projects, it seems to be more worthwhile to look at what characterizes game developement, and, as a consequence, what must be the focus of game design.

Eventhough the question *what is a good game?* is rather elusive, it is evident that a game must be fun to play! When creating a game, we may according to Schuytema (2007) capture fun in two areas:

fun

- in the general flow of the game experience and
- in the individual moments during a playing session.

which, together, determine the player's unique experience in playing the game.

As a first working definition, Schuytema (2007) states:

> *a game is a series of processes that takes a player to a result.*

which covers both the skill aspect of game playing, as well as the fact that a player generally must make a series of decisions. Refining this initial definition, Schuytema (2007) proposes to characterize a game as follows:

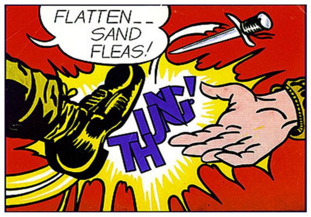
interactive electronic game

[2]msdn.microsoft.com/directx/XNA

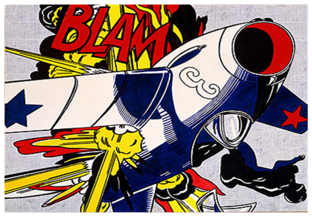
> A game is a play activity comprised of a series of actions and decisions, constrained by rules and the game world, moving towards an end condition. The rules and the game world are delivered by electronic media and controlled by a digital program.
>
> The rules and game world exist to create interesting situations to challenge and oppose the player. The player's actions, his decisions, excitement, and chances, really, his journey, all comprise the "soul of play".
>
> It is the richness of context, the challenge, excitement, and fun of a player's journey, and not simply the attainment of the end condition that determines the success of the game.

Although this definition could certainly be improved in terms of conciseness and clarity, it does cover the main characteristics of game playing, in particular *context* and *challenge(s)*.

varoom



thung



blam

3

With respect to challenge(s), we may observe that these may take the form of *occasional battle(s)*, as we have called them in the PANORAMA application, discussed in section 5.4. A battle may be characterized by the following conditions:

battle condition(s)

- confrontation on well-established area
- delimited in space/time
- audience/participants who judge victory/loss

It is in this context interesting to note that being mentioned in the *hall of fame*, that is a list of players with the highest scores, seems to be an intrinsic motivation for players to frequently return to the game, either to acquire or maintain a position on that list.

The actual development of a game is a complex process, and, in our experience, for each game unique, since there are many forces influencing a project. Nevertheless, whether you work on your own or in a team, inevitably the following aspects or roles have to be taken in consideration:

design team role(s)

- manager(s) – keep everything together
- producer(s) – maintaining focus
- programmer(s) – solve problem(s)

- tester(s) – control quality
- designer(s) – elaborate idea(s)

In our experience, for example in developing VU Life, that we will discuss in section 11.2, one person may take up several roles, but at some point it becomes clear what the strength(s), and for that matter weaknesses, of each individual participating in the project are.

history of game(s) There are many overviews of the hostory of (video) games, for example at wikipedia[3]. One overview that takes an interesting technological perspective is worth mentioning. Sanchez-Crespo Dalmau (2004) distinhuishes between eight phases, where each phase is characterized by one or more unique technical development(s):

history

1. phase i: before space war – hardwired
2. phase ii: spacewar on atari – console with game
3. phase iii: game console and PC – separate game development
4. phase iv: shakedown and consolidation – player code in data files
5. phase v: advent of the game engine – user level design
6. phase vi: the handheld revolution – the GameBoy
7. phase vii: the cellular phenomenon – larger installed user base
8. phase viii: multiplayer games – from MUD to Everquest

For those interested in more details, Sanchez-Crespo Dalmau (2004) provides as characteristic examples for each phase the following (additional) data: 1) tennis for two William Higinbotham Brookhaven National Labs New York, 1950' 2) Steve Russell, 1961, MIT, spacewar, two player game on Digital PDP-1 3) Atari VCS, Apple II, IBM PC (Dos) 4) Donkeykong, Pacman -> Nintendo 5) Doom -> Valve Halflife 6) Gameboy with well-established collection of game 7) NTT Docomo I-Mode, Samurai Romanesque 8) MUD (1979), MULE (1983), Ultima/Everquest 1600 hours/year.

[3]en.wikipedia.org/wiki/History_of_video_games

4

Technical progress does not stop here. Two developments worth mentioning are the rise of Second Life as a 3D immersive interface to the web, and the recent availability of a new 3D engine for flex 2, PaperVision3D[4], both of which may influence the choice of (online) game platform(s) for the near future. Another development that we may expect for the near future is according to Sanchez-Crespo Dalmau (2004): *a truly cinematic gaming experience.*

Screens from Samurai Romanesque.

5

example(s) – *samurai romanesque*

Samurai Romanesque, available on Japan's NIT DoCoMo packet-switched i-mode network, is an example of a mobile game with a large following. This massive multi-player game is developed by the japanese game developer Dwango. It runs on the Java 2 platform Micro Edition (J2ME). Players take, as we read in Krikke (2003) a virtual journey through 15-th century Japan, engage other players in real-time battles, visit historical towns and villages, practice the art of Zen, engage in romances and even can have children. This massive multiplayer role-playing game can accomodate up to half a million players, and is accounted to be a huge success in Japan. *Samurai Romanesque* is an example of a mobile game incorporating features such as position awareness, player history, chatting, and effective graphics. In Krikke (2003), it is further explained how the technology with which the game is implemented positions itself in the *battle for mobile cyberspace.*

research direction(s)– *serious games*

Serious games are here to stay. Currently there is for example already a great offer of business management games. When googling on *game*, *business*, and *management*, we find, among many other offerings games to train leadership[5] (which provides urgent problem situations in a variety of areas, including military and health care applications), and entrepreneurship[6] (which provides a eight round cycle of sessions instruction how to start a business, get clients, etc., with extensive feedback in the form of reports and comments after each round). A general observation we may make here is, however, that the games we have seen

[4]papervision3d.org
[5]www.experiencepoint.com
[6]www.marketplace-simulation.com

so far primarily focus on functionality and offer at best an efficient interface, which we do not find very appealing from a more artistic perspective.

There are many (more) resources on serious games[7]. To indicate what it is all about, we present a quote from virtual heroes[8]:

> *Serious games and simulations are poised for a second revolution. Today's children, our workforce and scientists are increasingly playing, learning, and inventing in visually intensive "virtual" environments. In our increasingly experiential economy, immersive educational and training solutions are needed to advance the workforce of tomorrow. Game-based learning and technologies meet this challenge.*

However, regardless of the fuss being made, apart from the euphorics their is little attention to determine in a more scientific way what the actual value of the game is in the learning process, and what elements or aspects of the game contribute to the learning experience. To provide such a foundation, we will propose a game reference model in section 12.1, that we have applied to formulate criteria for effective service management games in Eliens & Chang (2007).

There is a wide choice of technology available for the realization of serious games. For example, in the *climate game* project, we did explore various technologies, including interactive video with flash, as well as the use of the HalfLife2 game engine, with which we gained experience in developing a promotional game for our faculty, Eliens and Bhikharie (2006). With regard to the use of 3D we may remark that since ancient times a walk in space has served as a mnemonic device, and as such spatial memory may aid in retention and understanding, which might also provide a decisive argument for the use of 3D in aa serious game, such as a service management game!

As explained in section 3.4, we found great inspiration for Clima Futura, our climate game, in *Peacemaker*[9], that provided us with an example of how to translate a serious issue into a turn-based game

From an interview with the makers:

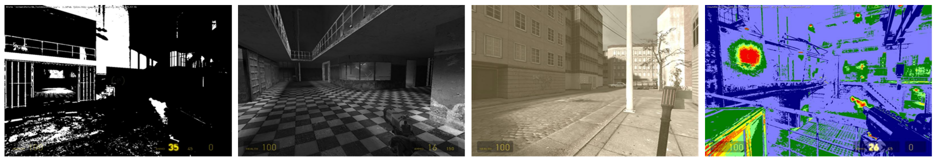
peace maker(s)[10]

> Q: With the lion's share of strategy games on the market being devoted to ending a conflict through violence, why was it important to you to emphasize the need for a peaceful solution?
>
> A: When we started to work on the project and looked around at other video games, we encountered the notion that war is much more challenging and conflict is essential to engage players. Many people we talked to perceived peacemaking as mere negotiations, where a group of diplomats sit at a table for lengthy discussions and sign agreements. We tried to shed light on what we see as the other side of peacemaking how challenging it is for a leader to gain trust and understanding in the face of constant violence. How difficult it is to execute concessions, while your own population is under

[7]www.cs.vu.nl/∼eliens/media/resource-serious.html
[8]www.virtualheroes.com
[9]www.peacemakergame.com
[10]seriousgamessource.com/features/feature_071806_peacemaker.php

> stress or feeling despair. In a sense, peacemaking can be more complicated, sophisticated and rewarding than war making, and it is a message that we would like to convey to young adults, the future generation of leaders.

In summary, Peacemaker covers both political and social issues, with appealing visuals, not sacrificing the seriousness of the topic. By presenting real-time events using video and (short) text, awareness is created by allowing a choice between the points of view of the various parties involved. Such awareness may lead to political action and will no doubt influence choices, also when it comes to issues of climate change. Essentially, serious games aim at attitude change, the rest follows automatically ...



black/white greyscale sepia thermal

6

11.2 game @ VU

In june 2005 we started with the development of a game, nicknamed VU-Life 2, using the Half-Life 2 SDK. We acquired a Cybercafe license for Half-Life 2, with 15 seats, because we would like to gain experience with using a state-of-the-art game engine, and we were impressed by the graphic capabilities of the Half-Life 2 Source game engine. After some first explorations, we set ourselves the goal:

- to develop a game that could be used for promoting our institute, and
- to prepare a masterclass game development for high-school students.

Our first ideas concerning a game included a game in which the subject chases a target, a game where the subject has to escape, and an adventure game. In the end we decided for a less ambitious target, namely to develop a game which gives the subject information about our institute, by exploring a realistic game environment, representing part of our faculty. As an incentive, a simple puzzle was included which gives the subject information on how to obtain a 'hidden treasure', to be found in a specific location in the game environment.

7

With only about eight months time, we decided to do a feasibility study first, to gain experience with the Half-Life 2 SDK technology, and to determine whether our requirements for the game and the masterclass could be met. For the VU-Life 2 game, we can summarize our requirements as follows:

- the game must provide information about the faculty of sciences of the VU,
- the game environment must be realistic and sufficiently complex, and
- the interaction must be of a non-aggressive, non-violent, nature.

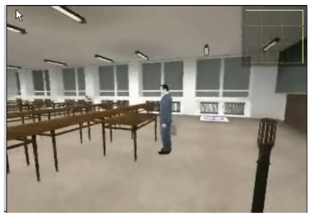
The last requirement has to do with the fact that the VU is by its origin a Christian university, so that anny agressive or violent interaction could hardly be considered to be an appropriate theme for a promotional game. For the masterclass, we stated the following requirements:

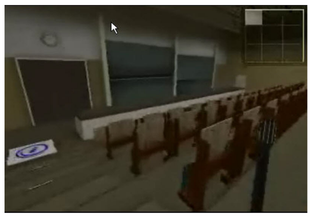
- it must be suitable for beginners, in particular high school students,
- it must explain basic texture manipulation, and
- offer templates for modifying a game level, and finally
- there must be a simple (easy to understand) manual.

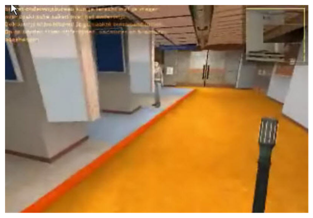
The format for a masterclass for high-school students at our institute is three times two hours of instruction. The goal is to attract (more) students for the exact sciences. However, if the masterclass would be too complex, we would run the risk to chase potential students away, which would be highly counter-productive.

VU-Life 2 – the game

To give an impression of the game and how we used the Source game engine and the associated Half-Life 2 SDK, let's start with a typical game scenario, illustrated with a walkthrough.

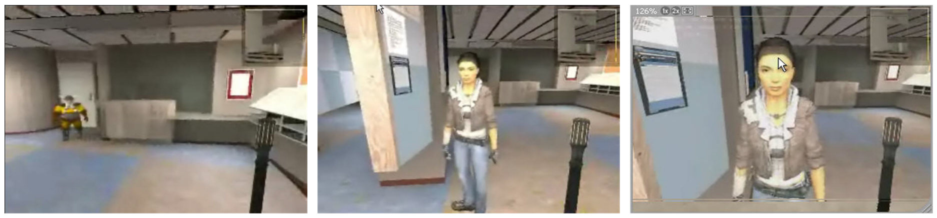

(a) lecture room


(b) lecture room


(c) student office

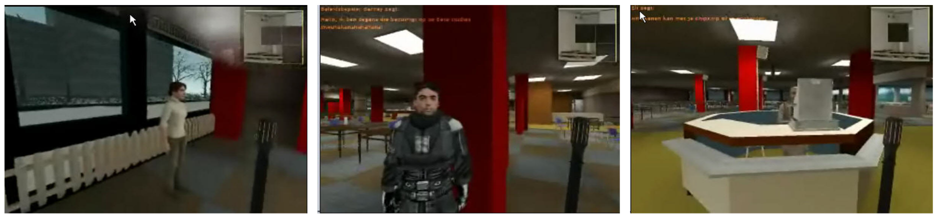
8

When starting VU-Life 2, the player is positioned somewhere in the game environment, such as a lecture room.. In the front left corner of the lecture room, middle right of left screenshot in the figure above, there is a place marked as an information spot. The information spot corresponds with one of the nine in the top right of the screen. The player is expected to detect this correspondence by exploring the game environment. The nine squares together form a puzzle, indicating, when all squares are filled, where the hidden treasure can be found. In other words, when the player visits all the nine information spots contained in the game environment, the player has solved the puzzle and may proceed to obtain the hidden treasure.


(a) student office (b) student office (c) student office

9

To visit all the information spots, the player has to explore the game environment, including another lecture room, the student administration office, and the student dining room. While exploring the game environment, the player may read information about the curriculum, meet other students, and encounter potentially dangerous individuals.


(a) restaurant (b) restaurant (c) restaurant

10

As illustrated in the figure above, the puzzle squares will gradually become filled, and when complete, the combined puzzle squares will indicate the location of the hidden treasure, which is the 7th row of chairs of the lecture room depicted previously.

Despite the fact that we intended to create a non-violent game, we must admit that the hidden treasure actually consists of obtaining the power to use weapons. Form our observations, and this was exactly what motivated us to include this feature, the use of weapons proved to be a most enjoyable aspect for the high school students playing the VU-Life 2 game, in particular when allowed to play in multi-user mode.

using the Half-Life 2 SDK – technical issues

The VU-Life 2 team had no prior experience with the Half-Life 2 Source SDK. Therefore we started by exploring three aspects of the Source SDK: level design with the Hammer editor, making game modifications, and importing (custom) models into Half-Life 2. During the exploration of these aspects we came across various technical issues, which we will discuss below.

level design First, we made various smaller levels. Each level was compiled and tested seperately so that it worked fine as a standalone level. The idea was to combine them, that is to create one large world containing the smaller levels. However, the initial coupling caused several compiling errors. After analyzing the errors, some important restrictons for building (large) levels became clear.

In the second part of the level compilation process called VVIS, a visibility tree of the level is made. This tree is used to tell the renderer what to draw from a given (player) viewpoint in the level. The amount of used brushes (the default shapes for creating a level) determine the size of the visibility tree. The bigger the tree, the longer VVIS will take to build the visibility tree at compile time and the more work the renderer has to determine what to draw at runtime. Therefore, the standard brushes should only be used for basic level structure. All other brushes that do not contribute to defining the basic level structure should be tied to so-called *func_detail* entities. This makes VVIS ignore them so that they do not contribute to the visibility tree, thus saving compiling and rendering time.

In addition, there is a (hardcoded) maximum to the number of vertices/faces you can use for a level. Each brush-based entity contributes to the number of vertices used. It is possible, however, to reduce the number of vertices used by converting brush-based objects to entities. This is done outside of the Hammer level editor with the use of 3D modelling software and the appropriate conversion tools.

With the above mentioned restrictions in mind we were able to create a relatively large level that more or less realistically represents the faculty of exact sciences of the VU campus. The key locations, as partially illustrated in the figures above, are the restaurant, lecture room S111, lecture room KC159, student office(s), and the multimedia room S353.

To give an impression of the overall size of the *VU.vmf* game level, as map information we obtained 6464 solids, 41725 faces, 849 point entities, 1363 solid entities, and 129 unique textures, requiring in total a texture memory of 67918851 bytes (66.33 MB).

game modifications Since a multi-user environment was required. we chose to modify the Half-Life 2 Deathmatch source code, The biggest challenge for modifying the code was finding out how to implement the features for VU-Life 2. To this end, relevant code fragments were carefully studied in order to find out how the code is structured and works. Furthermore, by experimenting, it was possible to get the features working. Below is a list of features for the VU-Life 2 Mod.

- *player properties* – players start out immortal, meaning that they cannot "die" while exploring the world. Furthermore, continuous sprinting is enabled, which allows the player to walk around faster.

- *puzzle HUD* – when the player starts out, the puzzle HUD is the only HUD element displayed.
- *puzzle setter* – allows puzzle parts to be displayed on the puzzle HUD.
- *weapon enabler* – allows weapons to be enabled/disabled for the player. Enabling the weapons also enables damage, and swithes from the puzzle HUD to the default Half-Life 2 HUD, which displays weapon and damage information along with a crosshair.

importing models Getting a model into the Half-Life 2 environment requires two steps:

- the model must be exported to the custom Valve format smd
- the model must be compiled from *smd* to *mdl* format

The first step required finding the correct plugin that allowed a conversion to the *smd* format. The second step required using Valve tool *studiomdl* and defining a *qc* file, which is used to specify properties for the compiled model. The default Valve tool *studiomdl.exe* proved to be difficult to work with, because it requires a lot of parameters have to be set. By using the StudioMDL 2.0 GUI, compiling the *smd* file was very easy. It sets the appropriate parameters, allowing the user to focus on the compiling of the model.

the masterclass – instruction and assignments

The masterclass consisted of three sessions, two hours each. In the first session, the (high school) students were given an overview and general instructions on how to accomplish the assignments, and were then set to play the VU-Life 2 game.

The assignments, as already indicated previously, were:

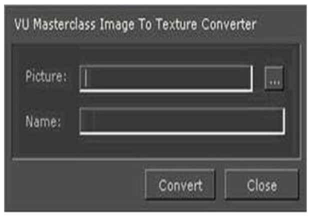
1. to modify an existing game level by applying different textures,
2. to create objects within an existing game level, and
3. (for advanced students only) to create a new level.

More complex assignments, such as creading a Mod, were considered to be outside of the scope of this masterclass.

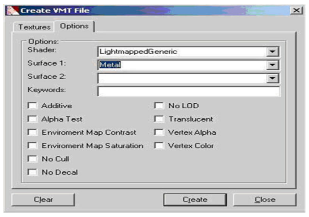
The overview and instructions given in the first session included:

- an overview of the history of games,
- a general introduction on modelling characters and objects,
- the use of the Hammer editor, and finally,
- an explanation of the assignments.

The history of games encompassed historic landmarks such as Pong, Tetris and The Sims, as well as a brief discussion of current games like Worlds of Warcraft, and Half-Life 2.

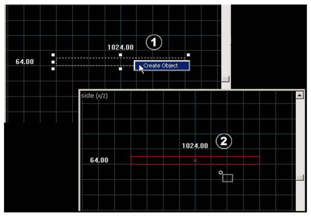


(a) converter



Options:
Shader: LightmappedGeneric
Surface 1: Metal
Surface 2:
Keywords:
☐ Additive ☐ No LOD
☐ Alpha Test ☐ Translucent
☐ Enviroment Map Contrast ☐ Vertex Alpha
☐ Enviroment Map Saturation ☐ Vertex Color
☐ No Cull
☐ No Decal

(b) VMT tool



(c) camera

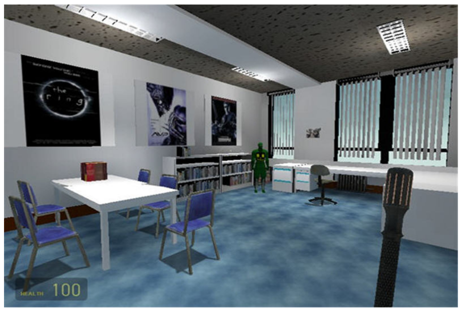
11

In the introduction on modelling an overview was given of the major tools, like Maya and 3DSMax, as well as a brief explanation of notions such as vectors, polygons, textures, lights, and skeleton-based animation.

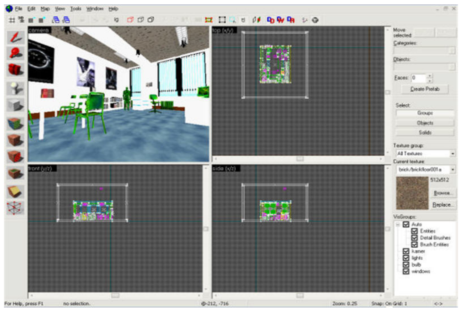
Both the explanation of the use of the Hammer and the assigments were explicitly meant as a preparation for session two, in which the students started working on their assignments.

In addition to the oral overview and instructions, the students were given a manual, that was made available in paper as well as online, to prepare themselves for the assignments. The homework for the second session was to make pictures suitable for the application as textures in the *masterclass room*, which is depicted below.

To allow the students to easily apply their textures, a texture conversion tool, was offered, that converts and image file into a texture for a particular location in the game level based on keywords, e.g. *mc_floor* for the texture on the floor of the *multimedia room*. Alternatively the students could use the VMT-Edit tool, and apply the texture using the Hammer editor.



(a) masterclass room



(b) room in Hammer editor

12

The introduction on how to use the Hammer editor covered the basic tools, including the

- *block tool* – for creating simple object,
- *selection tool* – to select objects for texturing,
- *entity tool* – to select dynamic or interactive objects, and the

- *texture tool* – to apply textures to an object;

as well as how to compile a level into a map ready for play, including an explanation of the BSP (world), VIS (visibility), and RAD (radiosity) components.

The students were explicitly told that the assignments did not involve any programming, creating game AI, or modelling. (To learn these aspects of game development, they were simply adviced to sign up for our curriculum.) Instead, we told them, use your phantasy and be creative!

lessons learned

In the second session, the high school students started working with great fervour.

Somewhat surprisingly, all students worked directly from the (paper) manual, rather than consulting the online documentation, or the help function with the tool.

masterclass at work

13

In retrospect, what appeared to be the main difficulty in developing the masterclass was to create challenging assignments for every skill level. In our case, the basic skill level (modifying textures of a template level) allowed the high school students to start immediately. By having optional advanced assignments like creating your own objects, you can keep all students interested, since there are assignments to match the various skill levels.

competition To stimulate the participants in their creativity, we awarded the best result, according to our judgement, with a VU-Life 2 T-shirt and a CD with Half-Life 2. The results varied from a music chamber, a space environment, a *Matrix* inspired room, and a messy study room. We awarded the *Matrix* room with the first prize, since it looked, although not very original, the most coherent.

example(s) – *dead media*

In Zielinski (2006), the *dead media* project is described, in a way that deserves to be presented without any transliteration. The following quotes characterize both the *dead media* project, as well as its context and implications for our (notions) of culture:

civilisation

> Media are special cases within the history of civilisation. They have contributed there share to the gigantic rubbish heaps that cover the face of our planet or to the mobile junk that zips through outer space.

dead media project

> Together with like-minded people, in 1995, Bruce Sterling started a mailinglist (at that time still an attractive option) to collect *obsolete software*. This list was soon expanded to collect *dead ideas*, or *dead artifacts*, and systems from the *history of technical media*: inventions that appeared suddenly and disappeared just as quickly, which dead-ended and were never developed further; models that never left the drawing board; or actual products that were bought and used and subsequently vanished into thin air.

machines can die

> Sterling's project confronted burgeoning phantasies about the immortality of machines with the simple facticity of a continuously growing list of things that have become defunct.

technology and death

> Once again, romantic notions of technology and of death were closely intertwined in the *Dead Media* Project.

How romantic we are, may be felt most intensely when our favorite machine breaks down, or threathens to be destroyed. This occurred to me recently, tripping over an electricity wire, connected to a thinkpad notebook. A small domestic drama occurred, and not even the promise of an improved (tablet) version of the same notebook could drive away the atmosphere of sadness and loss. Fortunately, after some fiddling with a screw driver, the damage seemed to have been reduced to an occasional blue screen. Yet, the incident illustrates how deeply involved we are with our machines, even if by profession we should know better.

research direction(s) – *service oriented computing*

Currently, one of the leading open source game engines is Delta3D[11], which consists of a collection of (open source) software components, well-maintained by a team at the Naval Postgraduate School in Monterey (CA). This section, which was previously called *open source game development*, will look at the way existing game engines may be enhanced by using services, for example web services that may be invoked by a REST-protocol (using urls), SOAP (Sinple Object Access protocol) or XML-RPC.

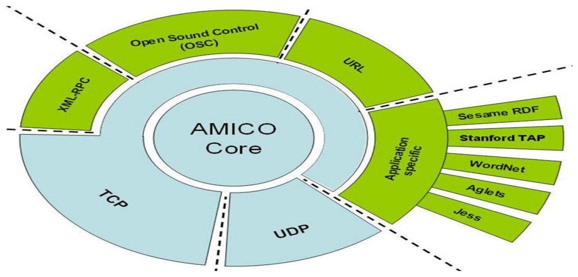
In Obrenovic & Eliens (2007), we present the following scenarion, or problem statement:

scenario(s)

> Michelle is a writer, and she writes a book about Dutch paintings. To collect necessary information, she analyses different sources about Dutch culture, many of them in Dutch language. However, being a beginner in Dutch, she often has to translate phrases from and to English. She also needs to find additional information about particular concepts and facts. Instead of using several tools, such as online dictionaries, definition books and search engines, she decides to compose a simple service that aggregates all the services she needs.

[11] www.delta.org

It should be not to difficult to think of similar scenarios in the context of games, for example language or culture training games. A generic solution to this type of scenario(s) is provided by AMICO[12] (Adaptable Multi-Interface COmmunicator), which is, according to its developer Zeljko Obrenovic, a generic platform, used to support rapid prototyping with heterogeneous software services, Obrenovic & Gasevic (2007). AMICO is based on the publish-subscribe design pattern, which is well suited for integration of loosely-coupled components, and often used in context-aware and collaborative computing. A publisher updates a shared data repository without being concerned with whether any subscribers are listening for updates. In the loosely coupled model, components can run on different machines in a distributed environment. An architectural overview of AMICO is given in the figure belew.



14

Our example scenario illustrates the integrated usage of various software components and services. Service-oriented computing (SoC) has a relatively long history, and may be regarded to have its roots in object-oriented software development, to the extent that services can be considered as components which offer their functionality at a sufficiently high level of abstraction. [Eliens2000]. Services provide higher-level abstractions facilitating implementation and configuration of software applications in a manner that improves productivity and application quality. Most of the existing work in SoC is concentrated on Web services, which is often seen as a main way to bring business to the Web [Blevec07]. Service-oriented computing is, however, not limited to Web services, but embodies key principles such as loose coupling, implementation neutrality, flexible configurability, persistence, granularity, and teams. Services provide higher-level abstractions for organizing applications for large scale, open environments. Adding service abstraction on top of heterogeneous open-source components, for example, enables their easier integration with other systems [Obrenovic07b]. In the category of service-oriented computing solutions, we distinguish between 4+1 dimensions:

dimension(s)

- scope – individual vs. corporate
- platform – local vs. web-based
- functionality – data aggregation vs. interaction
- developers – end-user vs. professionals

[12]amico.sourceforge.net

Additionally we address the issue of

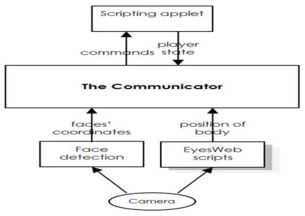
- licensing – OS vs. proprietary

Dependent on how a particular application is positioned in the category of service-oriented computing solutions, different issues play a role. For example, for corporate applications, security is of vital importance. In our approach, which is more targeted to the individual end-users, security plays as consequently less important role. However, other issues, such as time-management, are crucial. We introduce the licensing dimension to emphasis that only a very limited number of the solutions available for the construction of (primarily web-based) services and mashups is available as an open-source projects [Mashups]. In a general fashion, service-oriented computing requires three mechanisms, or layers of functionality;

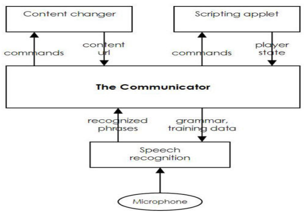
service-oriented computing

- service brokering – to connect services,
- service adapters – to transform data, and
- integration mechanisms – to deploy services.

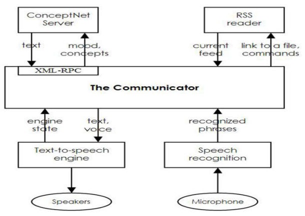
Traditionally, due to their complexity, these layers of functionality necessitated the help of professional developers with expertise in CORBA, COM or more recently SOAP, to construct suitable end-user applications. However, with the recent surge of web-based services, exemplified in the increasingly growing number of mashups, service-oriented computing seems to have come closer to end-users. That is end-users with non-trivial programming or scripting skills. For example, some of the existing end-user environments support extension of functionality by providing an access to Web services. Excel's Web Services [ExcelWS], enable end-users to create a code wrapper for Web services, and use functions from this wrapper within the spreadsheets. Limited forms of combinations of EUD and SoC can be found in Web based spreadsheets, such as Google spreadsheets [GoogleSS]. These environments enable receiving updates from remote sources for small number of predefined data types, such as stock prices and currency exchange rates. As a relatively new phenomenon, mashups introduce end-user service integration to create composed website or application that combines content from more than one source into an integrated experience, usually with a end-user interface. Content used in mashups is typically sourced from a third party via a public interface. Other methods of sourcing content for mashups include Web feeds (e.g. RSS or Atom), web services and Screen scraping. Many people are experimenting with mashups from Google, eBay, Amazon, Flickr, Yahoo [YahooPipes]. Although the dividing line between end-users and professional developers seems to become blurred, for example where end-users learn how to use scripting languages, we wish to emphasize that our notion of end-users is quite strict. An end-user should not be assumed to be able to do any scripting beyond simple formulas and an occasional conditional expression using the IF-THEN construct.



camera input



voice input



news reader

15

In comparison with the existing solutions, AMICO has a wider scope than most web-based mashups or EUD integration platforms, since it supports not only data-aggregation but, as illustrated in the figure above, provides support also for additional services, including interaction facilities involving speech recognition and TTS output. Yet, it is primarily aimed at individual users, supporting the integration of services for which both a service connection as well as data transformation functionality is available.

End-user development, as we will argue in section 11.4 becomes even more urgent,'where much of the content and functionality is actually provided by (a community of) users. How to provide a generic solution for user-added game content is however still an open problem, for which the service-oriented computing paradigm, however, seems to provide promising solutions.

11.3 immersion is not illusion

We live in a media-rich culture, full of images and audio-visual stimuli. Somehow, we never seem to get enough of that, and after a day of work, sitting in front of a computer screen, we still explore youtube.com for more exciting clips. The need for such audio-visual stimuli is not surprising, when you consider the high visual complexity of our daily (urban) life. The ability to process (visual) data quickly, seems to be a vital condition for our survival in the information society, where reality becomes an ephemeral phenomenon, as it is increasingly represented by images.

Images have a special status in our (western) culture. As Roland Barthes said, cited from Kress and van Leeuwen (1996):

analogon of reality

> Certainly, the image is not the reality but at least it is its perfect *analogon* and it is exactly this analogical perfection which, to our common sense, photography. This can be seen as the special status of the photographic image, it is a message without a code.

In the light of our observation(s) on *re-mediation*, in section 2.3, it is no surprise that early computer graphics took photo-realism as a benchmark. In an extremely well readable discussion (with examples) on *video game aesthetics*[13], David Hayward observes that *depending on your point of view, photo-realism is*

[13]modetwo.net/users/nachimir/vga

either a scourge or a grail. The main tendency seems to be towards ultra-realistic cinematographic visuals. However, fortunately, there are many examples where an aesthetically more viable approach is taken, either by introducing cartoon-style elements, or by abstractions suprpassing photorealism.

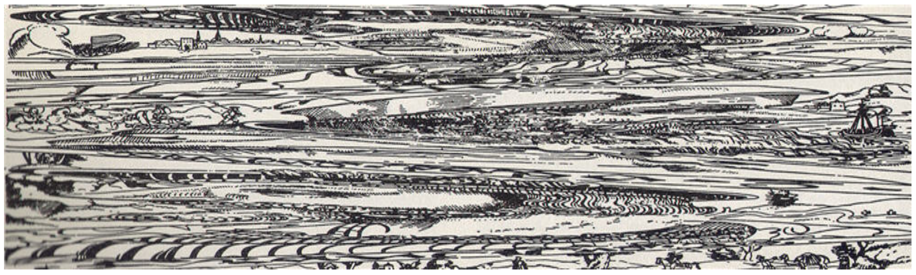
As member of the *creative industry*, that is multimedia or game developer, you will contribute to our visual culture, and hence it is worthwhile to reflect on what underlies the construction of images and (visual) narrative(s): such as *perspective*, *modality* and *composition*.

Perspective is fundamental in our understanding of images. Interestingly, it is a relatively new discovery, Alberti, dominating our visual culture from the days of the *enlightment* until now, despite artistic revolts of the 20th century. As we observed in section 9.4, the notion of perspective describes both

perspective(s)

- the organisation of the image, as well as
- the (optimal) point of view of the viewer.

The intricate relation between viewer and image, dependent on perspective, implies that when looked at from the 'wrong point of view', there will be a distortion of the image. The 'normal' perspective, as we know it, is the 'central' perspective. Variants of perspective, such as the anamorphism depicted below, force the viewer into an abnormal point of view, Anamorphisms. Better viewed online, watch it from the right, with you head close to the paper/screen.



16

In a multi-dimesnional space often a change of perspective, that is another point of view, brings to light many new aspects, not previously seen, or, in some cases, the correction of a reducing or distorting projection. Instruction in drawing perspective, by constructing a horizon with so-called *dis-appearance* points, used to be a standard element of art-school curricula, until recently.

Another interesting aspectof images is *modality.* The notion of modality may be implicitly characterized by using a quote from Kress and van Leeuwen (1996), where they discuss the *realism* of documentary film:

realism

> ... documentary modality of black and white realism ...

In a similar way, cartoons may be said to represent a particular *modality,* although not so much as a kind of realism, but rather as commenting on a situation, in a comic, or perhaps sarcastic or ironic way.

17

More in general, (visual) modality may considered to be part of a visual grammar, that, one way or another, defines the meaning of the image(s) of our culture: According to Kress and van Leeuwen (1996):

visual grammar

> grammar goes beyond formal rules of correctness. It is a means of representing *patterns of experience*. It enables human beings to build a mental picture of reality, to make sense of their experience of what goes on around them and inside them.

In section 9.3, we have discussed features of PANORAMA that were inspired by compositional elements of the visual grammar proposed in Kress and van Leeuwen (1996). There we observed that, however helpful these notions were, some cultural relativism is necessary, because the meaning associated with visual elements and their composition does not adhere to universal truth(s), but is rather the result of social processes involved in the construction of meaning.

Interestingly, panorama displays, from which we took inspiration for our system supporting social awareness (PANORAMA), were wide-spread popular artform of the 19th century. In a discussion of the origins of virtual reality, Grau (2003) observes

virtual reality

> the idea of *virtual reality* only appears to be without a history: in fact, it rests firmly on historic art traditions, which belongs to a discontinuous movement of seeking illusionary image spaces.

Linear perspective, as well as (perspective-related) atmospheric painting techniques (think of fog in CG terms), contributed significantly to the success of the immersive image spaces of the panaorama displays of the 19th century. Often the panorama displays were not realistic in a social or political sense, though. For example, the Mesdag Panorama depicted a rustic image of the Netherlands, that in that time already had been destroyed by industrialization and urbanisation.

More in general, we may ask: were these panaroma displays art? Grau (2003) unambiguously states:

immersion

> the concept of immersion when implemented as an artwork surrenders most of the essential properties of an artwork.

Although a discussion of what constitutes the essential properties of artworks is far beyond the scope of this manuscript, it is interesting to look at what properties an artwork may have, which according to Grau (2003) encompass:

properties of artwork(s)

- *form* – aesthetic whole
- *structure* – organisation of elements
- *function* – context of display
- *processuality* – (implicit) narrative structure
- *statement* – existential/political implication(s)

Both multimedia applications and games, one way or another, share these functions with art. Where art is probably most disctinct is in the existential implications, that is the way it makes a statement about our condition of being. Nevertheless, in a somewhat different way, als games, whether serious or not, do have implications with respect to our existence, and may be taken as a comment on our existince or an appeal to change our attitude towards it. From a more educational perspective, we may observe, following Grau (2003), that

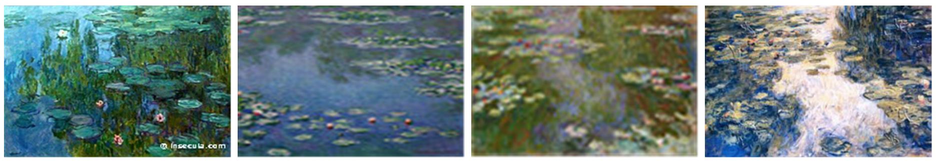
collective memory

> it is an apparent feature of the concept of immersion that it engages with the spatial and pictorial concentration of the awareness of one's own people, the formation of collective identity through powerful images that occupy the function of memory.

In other words, the media have a strong impact on what we consider to be part of our domain of living. However, evidently in our current visual culture, and as anyone famaliar with for example the VJ-culture can testify to, representation of meaning is not the primary target. As eloquently expressed in Grau (2003):

ecstatic transport

> using contemporary image techniques, immersive art very often visualizes elements that can best be described as Dionysian: ecstatic transport and exhilaration.



18

From a more epistemological perspective, reflecting on how we gain knowledge of the world around us, we may ask (ourselves) what role the media play in defining our reality. And as a consequence of the observations above, what is the meaning of this *ecstatic transport*, induced by the variety of audio-visual effects enabled by our new technologies?

With respect to the notion of *realism* Kress and van Leeuwen (1996) observe:

realism

> a *realism* is produced by a particular group as an effect of the complex of practices which define and constitute that group.

Basically that means that *realism* is a neytral term, that is anything may be considered to be *realistic*, as long as it is supported by a group or community. This as a remakr, should make us careful in denouncing particular styles as *naive*, as for example has been done with african art. Extrapolating this relativistic notion of *realism* to the commonly used notion of *naturalism*, Kress and van Leeuwen (1996) state:

naturalism

> each realism has its *naturalism*, that is a realism that is a definition of what counts as real, a set of criteria for the real, and it will find its expression in the *right*, the best, the most *natural* form of representing that kind of reality, be it a photograph or a diagram.

Indeed, for a scientist a diagram may be more natural than a colorful depiction of atmospheric pollution!

19

Let's return to the observation with which we started this section:

dominant paradigm(s)

> the dominant standard by which we judge visual realism (and hence *visual modality*) remains for the moment, naturalism as conventionally understood, *photorealism*.

To make things slightly more complex, we may add that the *visual modality* is the dominant modality of our culture, which means that we pay more attention to what we see than what we hear. According to feminist criticism, this may be de-constructed as expressing male dominance, which puts higher value on the ability of sight, than of touch and the perception of sound. Whether this is tur, and if so why, I leave to your own research. Apparently, this might have a cause in our need(s) for survival, where hunting was (once) more important than anything else. Think about how our need to be partially aware of everything, as discussed in section 9.4, might be related to that.

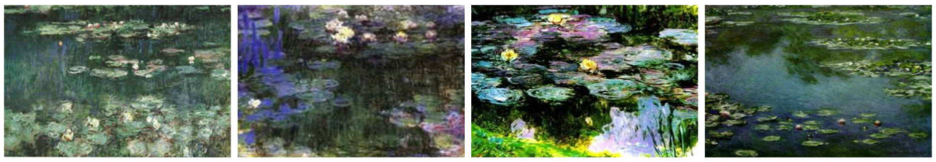
To conclude this section, let's briefly consider what is involved in creating an immersive application such as PANORAMA, or slightly more general, what is involved in what Sidney Fels in a UIU04 keynote called *designing intimate experience(s)*. A key observation here is that *people form relationships with objects external to their own self.*

relation(s)

- *response* – object disembodied from self
- *control* – self embodies object
- *reflection* – self disembodied from object

- *belonging* – object embodies self

These relationships, or notions of belonging, are essential in developing an *aesthetics of interaction*, a topic that will approach in the next chapter.



20

example(s) – *Monet's Nympheas*

One of the recommendations I make to everyone who is planning to go to Paris is to visit Monet's Waterlilies[14] which are exhibited in a small gallery near the Louvre. Monet, at that time at the at of his career, living in Giverney, was well aware of the paradoxical nature of his Herculean effort. As stated in Grau (2003):

mass medium

> thus, one year after Monet's death and fifty years after his *Impression soleil levant*, a late example of modern art reached the changed artistic landscape of the 1920's, transported in a derivative of *the* mass medium for images in the 19th century.

Being one of the innovators of painting in the early twentieth century, Monet nevertheless endeavored in painting panorama-wide paintings, with an aestehics of illusionistic immersiveness, however with a painterly touch so refined that this series of works easily trancends the ordinary illusionism of its predecessors. Monet himself might have seen it as an exercise to improve his painterly craft to the limits of perfection.

research directions– *information art*

Given the rise of the *creative industry* it is not suprising that initiatives have been taken to provide academic curricula in *creative technology*. The Institute of Creative Technology (LA), which developed the Mission Rehearsal Exercise discussed in chapter 9, may be taken as an example of an institute combining both technological and creative expertise in its staff.

The *new media* or *creative technology* play an important role in our society, in that they contribute in envisaging our dreams and (through advertisement) selling our reality. Wilson (2002), which provides an in any sense exhaustive investigation in the relation between technology and art, observes:

new media

[14] www.kahlil.org/monet10.html

> a consequence of the constitutive function of artistic-illusionary utopias for the inception of new media of illusion is that the media are both a part of the history of culture and of technology.

Although the research agenda's of scientists may easily conflict with the artistic agenda's of artists, Wilson (2002), describes numerous projects where projects where artists and scientists cooperate, with mutual benefit. Wilson (2002) is together with Zielinski (2006), Grau (2003) and many others, part of series of books from MIT Press, dealing with the cross section between technology, art and culture We repeat the quote, already given in section 1.1, that accompanies this series of books:

cultural convergence

> *the cultural convergence of art, science, and technology provides ample opportunity for artists to challenge the very notion of how art is produced and to call into question its subject matter and its meaning in society.*

Examples of projects with both relevance science and art, may for example be found in the area of *tele-presence*, to thematize, following Grau (2003):

tele-presence

- notions of artificial life
- fusion with (infinite) virtual image worlds
- transformation of self into digital data

Such themes, apart from all technical problems involved, deeply affect human aspirations, as expressed in our myths and movies:

human aspiration(s)

> *telepresence also combines the contents of three archetypal areas of human aspirations: automation, virtual illusion and metaphysical views of the self.*

In particular, such notions may be used to analyse, or de-construct, our behavior(s) on the web and our adoption of, for example, Second Life:

cybergnosis

> what is being preached is the phantasm of union in a global net community, cybergnosis, salvation through technology, disembodied as a post-biological scattering of data that lives forever.

This, eveidently may lead to criticism(s), which after all is a function of art, to make us aware of our limits, and the intrinsic qualities of our existence:

zealot(s)

> what we observe are hyperzealots of a new technoreligion running wild, zapping, excerpting and floating in cyberspace.

However, although we may encounter similar criticisms in popular culture, that is cartoons and games, it is interesting to reflect on the difference between art and the of new media applications that are somehow related to art. Again following Grau (2003):

aesthetics

> since the eighteenth century, aesthetic theories have regarded *distance* as a constitutive element of reflection, self-discovery and the experience of art and nature.

Aesthetic distance, however, is a notion that is also subject to criticism from new developments within art itself, for example *performance art*, which aspire a more direct existantial impact, as for example the works of Marina Abramovic, discussed in chapter ,a href=1.html>1.

What is meant with *creative technology* or *new media*, and what will constitute the tools of the *new culture* is not entirely clear. For example, Wilson (2002) observes that

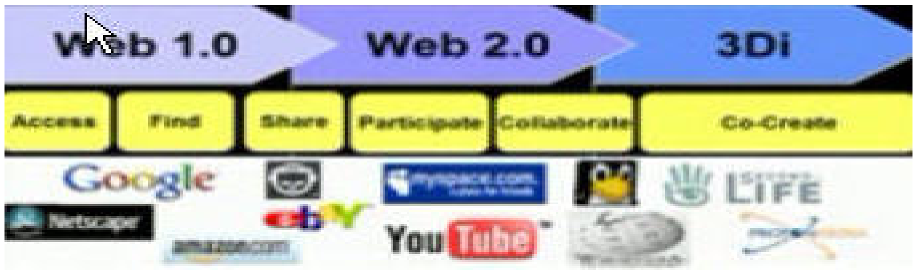
tool(s)

> aesthetic distance is no longer tenable when artist are engaging the same systems used in general communications and research

Does that mean that we must adopt *open source* to make an artistic statement, or shy away from the powerful visual effects enabled by for example shader technologies. Of course not. But it does indicate the need to critically reflect on the need and functions of these tools, and not adopt a technology, style or for that matter *realism* simply because everybody does so. Art, through its history, teaches us how to fight against both visions of dominance, and dominant vision(s)!

11.4 development(s) – game patterns

On youtube one can find many clips that somehow discuss the future of the web. One example of this is depicted below, a lecture that explains the differences between the various phases of the web, and the transition to a (3D) web in which the users *co-create*.



21

In summary, the following phases are mentioned:

webvolution(s)[15]

- access – *make available through url*
- find – *using search engine*
- share – *by storing online*
- participate – *by contributing content*
- collaborate – *to create meaningful sites*

[15]www.youtube.com/watch?v=-cZTdFTZV5Q

- co-create – *to build a (new) world*

These visions are (partly) realized in a multitude of applications. More interestingly, however, is the question how can deploy these mechanisms in a game context. In other words, can we think of a way to make use of *co-creation* facilities, to develop our Clima Futura game?

In Eliens et al. (2007b) we wrote: having decided on the general structure and elements of the Clima Futura game, a turn-based game loop, a climate-model driven simulation, exploratory video, and mini-games, the problem is how to connect these elements in a meaningfull way, and design a coherent collection of game events. This problem is further aggravated by the need to find a way to design in a collaborative fashion, necessitated by the sheer amount of disciplines and people involved.

To enable collaborative design we developed a game event description format, which standardizes the way game events are to be described, and for which we also developed an online form, structured as outlined below:

game event description format

- name of event – give a meaningful name
- event-id – for administrators only
- type – (generic/specific) game/model/video
- cause – game play/simulation/exploration
- feedback/information – give a logical description
- player actions – indicate all (logical) player options
- description of visuals – for feedback, information and player options
- additional information – give a url with reference(s) to information and visuals
- relates to event(s) – give id's or descriptions of related events

The *game description format* should not be confused with *game design patterns*, as introduced in Björk & Holopainen (2005), that we will discuss in section 12.1.

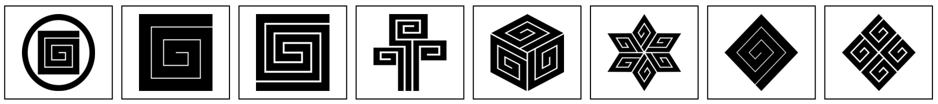
Before enforcing the game event description format, our ideas about the design of Clima Futura were gathered in a collection of narratives and brief descriptions, in what we called the *Clima Futura Design Bible*. Using the standardized game event description format, we hope to arrive at a more uniform way of describing the narratives, the perspectives from which these narratives can be experienced, the challenges or problems a player must solve, the resources available to the player, such as capital, knowledge and political power, the rewards, possibly using bonus credits for succesfully playing a mini-game, as well as the visuals, which will where possible be derived from the collection of videos we have available.

In addition to the game event description format, we also provided a minigame description form, containing a field to indicate the event that gave rise to the minigame, a field for a description of the minigame in words, as well as a field for the visual depiction of the minigame. Together, the game event and minigame description formats, provide a means to develop an online hyperlinked design document, that may serve as a reference for further design, development and coding. For the elaboration of the design, we are developing storyboards, which characterize in a visual way the major (dramatic) elements of narratives, structured using a subdivision in:

scenario(s)

- context – general setting, situation
- problem – event(s) to occur, problem to solve
- S-R situation(s) – stimulus/response (one or more)
- climax – action must be taken
- resolution – find solution or result

Although the actual workflow that we will deploy during development is at the moment of writing not clear, we will strive for developing templates that allow for a quick realization of the designs captured by the game event and minigame description format(s), along with the storyboards for visual design.



22

questions

game technology for serious applications

1. (*) What are the elementary steps in game development? Discuss the role technology plays in determining the game development project trajectory.

concepts

2. What phases can you distinguish in the actual development of a game?
3. Give arguments pro and con the use of a game engine.
4. Discuss the notion of immersion, and explain why immersion does not necessarily imply illusion.

technology

5. Characterize the built-in functionality that comes with a game engine.
6. Give a characterization of the tools that come with the Source Half Life 2 SDK.
7. Give a brief description of the history of immersive environments and application.
8. Discuss, on a suitable level of abstraction, the immersive features of games.

projects & further reading As a project, develop a non-violent game using the Source SDK. For example, you may develop an application that gives a community of users access their personal collections of photographs.

One interesting feature to explore is the use of narratives, that is a kind of guided tour that gives a user an overview of the collection of photographs by means of a story, taking (in other words) the user by the hand in navigating the gane space.

For further reading I suggest, apart from the manuals and learning materials that come with the Source SDK, books on game development such as Luna (2003), Gee (2003) and Klabbers (2006).

the artwork

1. *digital beauties* – taken from Wiedermann (2002).
2. Masereel, social realist works
3. Roy Lichtenstein, 1962
4. Masereel, social realist works
5. images from *Samurai Romanesque*, see section 1.3
6. HalfLife 2 shader programming
7. VU-Life 2 – opening screen
8. VU-Life 2 – screenshots
9. VU-Life 2 – screenshots
10. VU-Life 2 – screenshots
11. VU-Life 2 – tools
12. VU-Life 2 – tools
13. VU-Life 2 – masterclass
14. diagram AMICO core
15. diagram AMICO applications
16. Roy Lichtenstein, 1962, Stillives
17. Monet, Nympheas
18. Monet, Nympheas
19. Monet, Nympheas
20. Web 3Di – diagram for Webvolution[16]
21. signs – abstract, van Rooijen (2003), p. 146, 147.

The visual theme of this chapter is *realism*, on the one hand expressed by a choice from the work of the belgium artist Masereel, as well as in the screenshots of VU Life, which present a more or less realistic, that is recognizable rendering of our faculty. In the mean time, though, the restaurant has been rebuilt, making our virtual restaurant outdated, within a year after creating it. Also the cartooneske style of Roy Lichtenstein may considered to be strongly *realistic*, although in a slightly different sense. Finally, the *nympheas* of Monet represent a realistic epos of his famous garden, and in some sense an almost tragic attempt to express this in an artwork, which only adds to the *realism* of art, that is its vital role in our lives.

[16]www.youtube.com/watch?v=-cZTdFTZV5Q