game @ VU – developing a masterclass for high-school students using the Half-life 2 SDK

A. Eliëns, S.V. Bhikharie
Intelligent Multimedia Group
Vrije Universiteit, Amsterdam, The Netherlands
eliens@cs.vu.nl svbikh@few.vu.nl

No Institute Given

Abstract. In this paper, we will describe our experiences with developing a masterclass game development for 14-16 year old high-school students at the Vrije Universiteit, Amsterdam. For the masterclass, we developed a game using the Half-life 2 SDK, called *VU-life 2*, for which we created a realistic level covering part of the faculties premisses, as well as a simple assignment (of a non-violent nature) that the high school students had to complete before developing their own (variation on a) game level. Our experiences indicate that the moderately complex task of developing a game level using the Half-life 2 SDK is feasible, provided that the instructions and assignments are sufficiently well-focused.
keywords: game development, Half-life 2 SDK, education.

1 Introduction

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. See section 2 for more information on this.

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 geme 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.

In this paper we will report our experiences in developing the VU-Life 2 game and the associated masterclass[1]. The structure of this paper is as follows. In section 2 we will give an impression of the VU-Life 2 game by presenting a typical usage scenario. In the sections that follow, we will discuss the technical issues encountered in developing the VU-Life 2 game, and the assignments for the masterclass. In section 4, we will moreover describe the documentation we developed for the masterclass. In section 5 we will discuss the lessons we learned and in particular our experiences in presenting the masterclass to high-school students. And finally, we will draw our conclusions by giving a summary of our efforts and indicating our plans for the future.

2 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.

When starting VU-Life 2, the player is positioned somewhere in the game environment, such as a lecture room, fig. 1(a). In the front left corner of the lecture room, middle right of fig. 1(a), 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

[1] www.cs.vu.nl/∼eliens/game

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.

To visit all the information spots, the player has to explore the game environment, including another lecture room, fig. 1(b), the student administration office, figs. 1(c) and 2, and the student dining room, fig. 3. While exploring the game environment, the player may read information about the curriculum, meet other students, fig. 2, and encounter potentially dangerous individuals, fig. 3(b).

As illustrated in figs. 1-3, 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 in fig. 1(b).

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.

3 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 are, as partially illustrated in figs. 2-4, restaurant (fig. 4), lecture room S111 (fig. 2(a)), lecture room KC159 (fig. 2(b)), student office (figs. 2(c) and 3), multimedia room S353 (not shown).

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.

4 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 in section 1, 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.

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 in fig. 5(a).

To allow the students to easily apply their textures, a texture conversion tool, fig. 4(a), 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, fig. 4(b), and apply the texture using the Hammer editor, figs. 4(c) and 5.

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!

5 Lessons learned

In the second session, the high school students started working with great fervour, see fig. 7.

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

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 participant in their creativity, we awarded the best result, according to our judgment, 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.

6 Conclusions

In this paper, we reported our experiences in developing a moderately complex game environment and associated masterclass for highschool students, illustrating the effort needed to develop such an application in an educational setting, indicating technical constraints as well as the documentation requirements that must be met. Somewhat surprisingly, our target audience preferred a step-by-step approach, using the paper manual, over the use of the online material and help on a by-need basis. Finding a suitable range range of assignments, sufficiently variable in difficulty, however, will remain a challenge for future efforts.

Acknowledgements

We gratefully acknowledge the contribution of the following people to the development of the VU-life 2 game and the masterclass *game development*:

Anthony Agustin (developer), Kin Hung Cheng (developer), Niels Rietkerk (documentation writer), Steve Stomp (character modeller), and Mikhail Zouskov (technical support)

References

Juul J. (2005), *Half-real – Video Games between Real Rules and Fictional Worlds*, MIT Press