A framework to research procedural content generation in First Person Shooters

Marco Ballabio

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Developing a video game is a really complex process, which involves a wide range of professional figures. Since video games are interactive products, one of the most important roles is the game designer, who defines the game design and the gameplay.

Chapter 2

State of the art

In this chapter we analyze the current state of *level design* and of its common practices, both in academic and in professional environments, with attention to the genre of *First Person Shooters* (or *FPS*).

We then talk about *Procedural Content Generation* (or *PCG*), focusing on how it allows to enrich and ease the design process.

Finally, we give an overview of the *First Person Shooter* genre, analyzing its features, history and evolution, devoting special attention to the games that lead to greater innovation in the field and to the ones that are used to perform academic research in this field.

2.1 Level Design Theory

Level design is a game development discipline focused on the creation of video game levels.

Today, the *level designer* is a well-defined and fundamental figure in the development of a game, but it was not always so. In the early days of the video game industry, it was a widespread practice to assign the development of levels to members of the team with other roles, usually programmers. Apart from the limited number of team members and budget, this was because there were no tools such *level editors*[1], that allowed the *level designer* to work on a level without being involved with code.

The *level designer* has a really important role in the development of a good game, since he is responsible for the creation of the world and for how the player interacts with it. The level designer takes an idea, which is the

[1] A level editor is a software used to design levels, maps and virtual worlds for a video game. An individual involved with the creation of game levels is a level designer.

game design, and makes it tangible. Despite the importance of this role, after all this years, it has not been established a common ground or a set of standards yet, instead, *level design* is often considered as a form of art, based on heuristics, observation, previous solutions and personal sensibility.

In addition to game play, the game designer must consider the visual appearance of the level and the technological limitations of the *game engine*[2], combining all this elements in a harmonic way.

One of the core components of level design is the **"level flow"**. For single player games it translates into the series of actions and movements that the player needs to perform to complete the level. A good practice for *level design* is to guide the player in a transparent way, by directing his attention towards the path he needs to follow. This can be achieved in different ways. Power ups and items can be used as breadcrumbs to suggest the right direction in a one way fashion, since they disappear once picked up. Lighting, illumination and distinctly colored objects are another common approach to this problem. A brilliant example of this is *Mirror's Edge*[3], which uses a really clear color code, with red interactive objects in an otherwise white world, to guide the player through its fast-paced levels. There are also even more inventive solutions, like the dynamic flock of birds in *Half Life 2*[4], used to catch the player attention or to warn him of incoming dangers[1]. Finally, sounds and particular architectures are other elements that can be used to guide the player. In the academic environment, a lot of researchers have analyzed the effectiveness of this kind of solutions: Alotto[2] considers how architecture influences the decisions of the player, whereas Hoeg[3] also takes into account the effect of sounds, objects and illumination, with the last being the focus of Brownmiller's[4] work.

In multiplayer games the *level flow* is defined by how the players interact with each other and with the environment. Because of this, the control of the *level designer* is less direct and is exercised almost exclusively by modeling the map. Considering *FPS*, the *level flow* changes depending on how much an area is attractive for a player. The more an area is easy to navigate or offers tactical advantage, such as cover, resources or high ground, the more players will be comfortable moving in it. This doesn't mean that all areas need to be designed like this, since zones with a "bad" flow but an attractive reward, such as a powerful weapon, force the player to evaluate risks and benefits, making the game play more engaging. The conformation

[2]A game engine is a software framework designed for the creation and development of video games.

[3]Digital Illusions CE, 2008.

[4]Valve, 2004.

of the map and the positioning of interesting resources are used to obtain what Güttler et al.[5] define as **"points of collisions"**, i.e. zones of the map were the majority of the fights are bound to happen. Moving back to academic research, Güttler et al. have also noticed how aesthetic design loses importance in a multiplayer context. Other researches are instead focused on finding **patterns** in the design of multiplayer maps: Larsen[6] analyzes three really different multiplayer games, *Unreal Tournament 2004*[5], *Day of Defeat: Source*[6] and *Battlefield 1942*[7], identifying shared patterns and measuring their effect on gameplay, suggesting some guidelines on how to use them, whereas Hullet and Whitehead identify some patterns for singleplayer games[7], many of whom are compatible with a multiplayer setting, with Hullett also proving cause-effect relationships for some of this patterns by confronting hypnotized results with the ones observed on a sample of real players[8]. Despite these experimental results contributing to a formalization of *level design*, we are still far from a structured scientific approach to the subject.

2.2 Procedural Content Generation

Procedural Content Generation refers to a family of algorithms used to create data and content in an automatic fashion. In game development it is commonly used to generate weapons, objects, maps and levels, but it is also employed for producing textures, models, animations, music and dialogues.

The first popular game to use this technique was *Rogue*[8], an ASCII dungeon exploration game released in 1980, where the rooms, hallways, monsters, and treasures the player was going to find were generated in a pseudo-random fashion at each playthrough. Besides providing a huge replay value to a game, *PCG* allowed to overcome the strict memory limitations of the early computers. Many games used pseudo-random generators with predefined *seed values* in order to create very large game worlds that appeared to be premade. For instance, the space exploration and trading game *Elite*[9] contained only eight galaxies, each one with 256 solar systems, of the possible 282 trillion the code was able to generate, since the publisher was afraid that such an high number could cause disbelief in the players. Another

[5]Epic Games, 2004.
[6]Valve, 2005.
[7]DICE, 2002.
[8]Michael Toy, Glenn Wichman, 1980.
[9]David Braben, Ian Bell, 1984.

example is the open world action role-playing game *The Elder Scrolls II: Daggerfall*[10], which game world has the same size as Great Britain.

As computer hardware advanced and CDs become more and more capacious, *Procedural Generation* of game worlds was generally put aside, since it could not compete with the level of detail that hand-crafted worlds were able to achieve.

However, in the last years, with the players' expectations and the production value of video games constantly increasing, *Procedural Generation* made a comeback as a way to automate the development process and reduce costs. Many *middleware* tools, as *SpeedTree*[11] and *World Machine*[12], are used to produce content, like terrain and natural or artificial environments.

Many modern *AAA*[13] games use *Procedural Generation*: in *Borderlands*[14] a procedural algorithm is responsible for the generation of guns and other pieces of equipment, with over a million unique combinations; in *Left 4 Dead*[15] an artificial intelligence is used to constantly make the players feel under threat, by dynamically changing the music, spawning waves of enemies and changing the accessible paths of the level; in *Spore*[16] *procedural animation* is employed to determine how the creatures created by the player move.

Nowadays, *PCG* is widely used by *independent* developers, that, lacking the high budgets of *AAA* games, try to obtain engaging and unusual gameplay using unconventional means. The most famous example is *Minecraft*[17], a sandbox survival game which worlds, composed exclusively by cubes, are generated automatically. Currently, the most extreme form of *Procedural Generation* is the one found in *No Man's Sky*[18], a space exploration game, where space stations, star-ships, planets, trees, resources, buildings, animals, weapons and even missions are generated procedurally. Following in the footstep of their forefather, many *roguelike* games still use *PCG*, like *The Binding of Isaac*[19].

[10] Bethesda Softworks, 1996.
[11] IDV, Inc.
[12] World Machine Software, LLC.
[13] Video games produced and distributed by a major publisher, typically having high development and marketing budgets.
[14] Gearbox Software, 2009.
[15] Valve, 2008.
[16] Maxis, 2008.
[17] Mojang, 2011.
[18] Hello Games, 2016.
[19] Edmund McMillen, 2011.

2.3 Procedural Content Generation for First Person Shooter maps

2.4 History of Level Design in FPS

2.5 Summary

Bibliography

[1] M. Gallant, "Guiding the player's eye," May 2009.

[2] B. Alotto, "How level designers affect player pathing decisions: Player manipulation through level design," 2007.

[3] T. Hoeg, "The invisible hand: Using level design elements to manipulate player choice," 2008.

[4] J. Brownmiller, "In-game lighting and its effect on player behavior and decision-making," 2012.

[5] C. Güttler and T. D. Johansson, "Spatial principles of level-design in multi-player first-person shooters," in *Proceedings of the 2Nd Workshop on Network and System Support for Games*, NetGames '03, (New York, NY, USA), pp. 158–170, ACM, 2003.

[6] S. Larsen, "Level design patterns," 2006.

[7] K. Hullett and J. Whitehead, "Design patterns in fps levels," in *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, (New York, NY, USA), pp. 78–85, ACM, 2010.

[8] K. M. Hullett, "The science of level design: Design patterns and analysis of player behavior in first-person shooter levels," 2012.