StreamLevels: Automatic Balancement of Platform Levels

Lucas N. Ferreira
University of California, Santa Cruz
Augmented Design Laboratory
lnferrei@ucsc.edu

Abstract

Balancing difficulty is a very complex task inside the process of level design. The goal of this task is to find a level such that all the audience of the game will have an engaging experience. Typically it involves iterations in which the designer tune/tweak the level and test it with several players until finding the best balance. This paper presents a visual tool that helps the designer in this task. The focus of this work is on 2D classic platform games with mechanics similar to the Super Mario Bros. The tool has three main modules: visualization, suggestion and generation. The first one maps the data to visual attributes such as streamlines in order to show the designer the players experiences. The second one suggests adaptations to the level based on the data collected from the players. The last one allows the designer to create new levels by drawing streamlines interactively. To evaluate the tool, it will be collected data from different players in an open-source version of the Super Mario Bros game.

Introduction

Successful games have the ability of drawing players in a very immersive experience. When players have a high level of engagement with a game, they are in a state called Cognitive Flow (Baron, 2012). While in this state, players experience extreme focus on a task, a sense of active control, merging of action and awareness, loss of self-awareness, distortion of the experience of time and the task being the only necessary justification for continuing it. Flow state is related to the players skills and the difficulty of the level. As illustrate in Figure 1, these two attributes interact to result in different cognitive and emotional states. If players’s skill is too low and the level is too hard, it becomes anxious. Alternatively, if the level is too easy and the player’s skill is too high, player become bored. However, when skill and difficulty are roughly proportional, it enters the Flow state.

The main goal of a game designer is to create experiences that keep as many players as possible into the Flow state. Therefore, it must find a balance between player’s skills and the difficulty of the levels. However, each player has a different skill prior to the beginning of the game and they develop their skills in a different pass as well. A typical approach to balance the game for the majority of its audience consists of perform play testing sessions with players who have different profiles (Zook, Fruchter, & Riedl, 2014). The data collected in this process are then used to tune the difficulty of the game. These tasks can be performed several times among the development of the game, until finding the “perfect” balance.

This paper is motivated by the big effort that is needed by the designers for iterating with players to balance the game difficulty. The focus is on classic 2D platform games, i.e. linear games which the player has an initial location \((x_i, y_i)\) and must reach a final location \((x_j, y_j)\) of a level by jumping over platforms and killing enemies in its way (It is assumed here that \(x_j > x_i\)). Considering this context, three methods will be developed and compiled in a tool that will help them in this task.

The first method consists in visualizing data extracted from play testing sessions in order to show the pattern of the players experiences. It helps the designer to understand its floss in order to improve the level. The second one is a method that gives improvement suggestions based on the collected data. This can help the designer to start the tun-
ing task or even fully automatize the task, depending on the method’s performance. The last method allows the designer to interactively define the behavior of a fictitious player which will be used to generate a brand new level. This method also can generate a new level for each of the behavior collected during the play testing.

**Related Work**

Automatic balancing is an important topic inside the field of Artificial Intelligence in Games and it has been considered in two different scenarios: offline and online (Zook et al., 2014). Offline automatic balancing is the use of tools for helping designers to explore the design parameter space, improving quality or accelerating the design process. To evaluate design during the game development process, this kind of tool has used simulations and formal checking.

The simulation approach is typically used for evaluating game feasibility in an intractable large space of possible parameters. For instance, (Shaker, Shaker, & Togelius, 2013) combine a rule-based reasoning approach with simulation to generate content for a physics-based game. (Bauer, A., & Popovic, 2013) and (Cook, Colton, & Gow, 2012) use sampling methods to evaluate platform game level playability.

The model-checking approach defines game mechanics in a logical language to provide guarantees on generated designs having formally defined properties typcially at the cost of being limited to more coarse design parameter decisions. For example, (Smith, Butler, & Popovic, 2013) use logic programming and constraint solving to generate sets of levels meeting given design constraints. (Jaffe et al., 2012) use game-theoretic analysis to understand the effects of game design parameters on competitive game balance.

On the other hand, online automatic balancing consists in adapting the design parameters in real-time based on specific hand-crafted rules or on data collected until the current state of the game. The first one allows designers to describe precisely how a method adjusts a game considering the design goals. However, this might not be an easy task since this description must cover all the scenarios in which the game design can change. For instance, (Hunicke & Chapman, 2004) defined a rule considering average/variance of player damage and inventory status to adjust the amount of enemies and powerups. (El-Nasr, 2007) and (Thue, Bulitko, Spetch, & Wasylishen, 2007) defined models of the player based on attributes such as skills and personality. Thus, they are able to select personalized content for players using hand-crafted rules.

Since describing rules can be a hard effort, some researchers have developed methods that use data to propose design improvements. For instance, (Shaker, Yannakakis, & Togelius, 2013) and (Liapis, Martinez, Togelius, & Yannakakis, 2013) use machine learning to model player preferences and optimize the output of these models to select potential game parameters. (Harrison & Roberts, 2013) optimize player retention and (Zook & Riedl, 2012) optimize game difficulty with similar techniques.

---

1If it possible for the player to reach the final state of the game with the current design parameters.

**Proposed Research Directions**

The proposed tool for helping the designers tune the levels after a play testing iterations will be developed in the context of 2D platform games and it will be composed of three main modules: data visualization, improvement suggestion and interactive generation. The visualization module will assume that each example of the dataset is a grid of size $w \times h$, where $w \geq l$ are the width and height of the level, respectively. Each element of the grid is a set of vectors (a player can visit a given element more than one times) representing the velocity of the player at that point every time it passed through that element. Therefore, data will be visualized as streamlines showing the whole path that the player used until dying or reaching the end of the level. The magnitude of the velocity vector will be mapped to colors along the streamlines in order to show how fast the player was in a given region of the level. To show if the player killed an enemy or got a powerup, the streamline will be marked respectively with o or + in the position where the event happened.

The second step of the research will consist in investigating how to create an algorithm that receives all the streamlines as input and tries to find a new line that represents the pattern of the player’s behavior. This might be achievable performing a regression on the data. The last step will be to design an algorithm capable of generating a level from a streamline. One possible approach consists in evaluating the streamline from the end to the beginning, placing platforms, enemies and powerups based on the relative position of the line and its direction variations.

**References**


