StreamLevels: Using Visualization to Generate Platform Levels

LUCAS N. FERREIRA, Augmented Design Lab, University of California Santa Cruz

Level design is one of the hardest and most time-consuming tasks during the development of a game. It is hard because the designer has to create a gaming experience with a balanced difficulty. Therefore, it requires to find a level such that all the audience of the game will have an engaging experience. Typically designers work using level editors in which they can only place game objects in a virtual space and configure parameters. In such editors, it is hard to measure quality and feasibility during the edition time, so the designer must use design-and-test iterations in order to achieve its goal. Motivated by this problem, recently some authors have proposed AI-assisted level editors, in which the designer work together with an AI agent that evaluate feasibility, generate new sections, suggest changes, etc. This paper presents a new hybrid level editor for 2D platform games which suggests entire levels based on a streamline representing a player trace. A player trace can be defined by either drawing a new streamline or uploading real data. In the second case, the designer can use the editor to visualize a real player trace and edit it as desired.

Additional Key Words and Phrases: AI-assisted design, procedural content generation, levels, visualization, streamline

ACM Reference Format:
DOI: 0000001.0000001

1. INTRODUCTION

Successful games have the ability of drawing players in a very immersive experience. To create such experience for a given game mechanics, a designer must create a set of levels with difficulties proportional to the experience of the considered audience. Since the traditional level editors do not represent the playspace of the game, the designer is only able to create experiences by placing objects in the virtual world. Therefore, the design process typically consists of iterations in which the designer tune/tweak the position of the objects and test it with several players until finding the best balance. This process works well, but it turns game development very time-consuming and expensive.

In order to speed up and cheapen the development process, it is possible to create AI-assisted level editors, in which the designer work together with an AI agent. This agent can perform several tasks while the designer is working: feasibility evaluation, difficulty estimation, level generation, etc. Moreover, AI-assisted level editors can be used to allow the designer working in the playspace, i.e. in the possible sequences of play [Zook et al. 2014]. In such tools, it is possible to visualize the level playspace, allowing the designer to immediately evaluate high-level properties, such as difficulty, risk, and importance to the larger space. Working in the playspace also facilitate sophisticated
edits to the level, which would be automatically translated into the necessary changes to the underlying level.

It is possible to create an editor which works in playspace by using procedural content generation (PCG) to generate levels. In this approach, the designer visually defines the overall characteristics of the desired experience and the generator is responsible for defining the specific properties, such as position of objects for instance. Several excellent level generators have been proposed in the PCG field, however most of them use numerical parameters, which do not directly represent high-level characteristics of the playspace. Therefore, this work proposes a new AI-assisted level editor that describes a level as a streamline, which is then used as input to a level generator. The focus of this paper is on 2D platform games, however it seems possible to generalize this tool for other game mechanics.

The developed tool is a web application that allows the player to define a streamline by either drawing it or uploading player trace data. The line is then discretized in "rooms", where each one contains a set of line points. A new level generator is also proposed, and it uses the points to generate challenges in each room. This system was qualitatively analyzed in order to evaluate the general experience of designing levels in the playspace.

The remainder of this paper is organized as follows: Section 2 contains a brief review of PCG and AI-assisted design; Section 3 describes the design environment of the proposed tool; Section 4 presents a constructive level generator that uses data from StreamLevels and Section 5 presents results for preliminary experiments. Finally, Section 6 concludes this work.

2. RELATED WORK

This work proposes an AI-assisted design tool to allow level edition in playspace. The proposed approach consists in using streamlines as input for a 2D platform level generator. Therefore, this section presents a brief review of AI-assisted design and PCG method for level generation in platform games.

2.1. Procedural Content Generation

PCG has been deeply studied in the context of 2D platform games and most of the works are dedicated to level generation. Different approaches were explored and we can highlight three major classes: constructive, search-based an experience-driven. The first one is a general class that involves methods that produces only one output per run. They can be based in very different algorithm, however they are all relatively fast and hence able of creating levels at runtime. Other point in common is that, in general, they do not provide a lot of control over the output and its properties. An example of constructive method is presented in [Dahlskog and Togelius 2012]. It uses a pattern-based level generator that combines hand crafted building-blocks with small variations depending on assigned parameters on difficulty and reward settings.

Search-based is perhaps the approach investigated the most in the last decade. In this case, PCG is treated as an optimization problem, so the methods work by finding the maximum value of a fitness function that represents the entertainment of a given content [Togelius et al. 2011b]. For instance, [Togelius et al. 2011a], [Shaker et al. 2010] and [Ferreira et al. 2014] proposed different evolutionary algorithms for the generation of platforms games.

The experience-driven approach uses models of player experience built with information collected from the interaction between the player and the game. This model is then used to generate personal content, adapting the game for a specific player. For instance, in [Pedersen et al. 2009], a neural network is used to model playing behavior characteristics considering features such as completion time, amount of collected...
items, number of deaths, etc. This model is then used to define placement and sizes of gaps in the level and the existence of direction changes.

2.2. AI-assisted Design

AI-assisted Design consists in using AI agents to help designers in a task that involves crafting a virtual artifact. Recently, this approach has been applied in the PCG field and several novel tools have been developed for production of different types of content. Typically, these tools speed up the game production process by a fast generation of a huge variation of content. For example, SpeedTree is a commercial software for procedural generation of entire forests. This system visually expose the parameters for the user, which can tweak them in order to achieve a desired goal [Inc. 2002]. Another example is [Lipp et al. 2008], a visual editor that uses a grammar approach for procedural modeling of 3D buildings.

These tools speed up the content generation process, however the designer still needs to put this content in the game to test the resultant experience. A novel approach has recently been used, it consists in the development of tools that work directly in the playspace of the game, allowing the designer to specify high-level attributes of the experience it is trying to create. For example, Biped [Smith et al. 2009] allows designers to rapidly specify prototypes of their games and view play traces from both human and computer players. The systems strength is in allowing a designer to view situations in which their game design “fails” by asking the system to show a playthrough that produces a certain undesirable result. [Smith et al. 2010] presents a system called Tanagra which generates 2D platform levels considering user-specified playspace constraints. Tanagra uses a rhythm-based approach, dividing a level into beats and letting the designer pin level properties for each beat, such as the position of the platform or the presence of an enemy. The system then automatically shapes the rest of the level to remain traversable from left to right.

3. DESIGN ENVIRONMENT

StreamLevels is a web-based system developed in Node.js that allows the user to describe level structures using streamlines. There are two ways to describe a streamline: drawing or uploading real player trace data. Figure 1 shows the main screen of the system. The tile-based canvas is the area where the user will draw and visualize the streamlines. It is possible to draw by either dragging the mouse or clicking down several times. More than one lines can be drawn, which helps to design complex levels.
The tile size is customizable, so the designer can control the size of the desired levels. As the designer draws in the canvas, the tiles hit by the line are highlighted in blue, giving a feedback to the user.

Once the player presses the button “Generate”, the system iterates among all the selected tiles, adding them to a list. Each element $i$ of this list stores actually a set of normalized points (the points inside the tile $i$). This list is then transformed into a JSON object, which is directly download. StreamLevels does not formally generate the levels, it is only used for defining the overall shape of the level. The idea is to have a general level editor, so one can uses it for different game mechanics. For example, with this system, it is very easy to define the shape of a track for a racing game (note the close path checkbox in Figure 1).

4. LEVEL GENERATION

In order to demonstrate how the proposed system can be connected to a level generator, it was proposed a simple constructive algorithm for generation 2D platform games. Since it is a constructive algorithm, it generates only one solution during a run. This solution is created by iterating over the list created by the StreamLevels system. For each element of the list, it is calculated the average point $p = \{x, y\}$, as showed by the red dot in Figure 2.

![Fig. 2. The five main steps for generating the platform challenges of each room.](image)

After calculating the average point, it is calculated its symmetric point $p' = \{1-x, y\}$. With these two points, a flat platform is created adding tiles from position $p$ to $p'$. During this process, there is a small probability of adding a gap in the generated platform. With the main platform created for the room $i$, the next step consists in checking if the player is able to jump over it from the ground. This is done by evaluating the height of the generated platform with relation to the ground of the room. If this height is greater than the jump height of the player, than we must add a new platform to keep the first one reachable. If that is the case, a new platform with random width is added with height $y/2$. 

ACM Computers in Entertainment, Vol. 9, No. 4, Article 1, Publication date: December 2015.
5. EXPERIMENTS AND RESULTS

StreamLevels was evaluated using the level generator described in the last section. The proposed experiments consist in three different design tasks:

— Create a level with height > 3 rooms.
— Create a level with multiple paths.
— Create a level with width > 10 rooms.

These experiments are intended to analyze the quality of the generated levels and the average time the system takes to output them. Each task were performed 10 times and the best result (judged by an expert human designer) was selected as the result. Figure 3 shows the best one for each task.

![Figure 3](image_url)

Fig. 3. The two levels in the top are respectively the best result of the first and the second experiments. The one in the bottom is the result of the third experiment.

It is important to highlight that all the showed levels have a feasible path from the first to the last room. The second level has three paths and all of them are feasible from the beginning to the end. However, the feasibility does not apply if the player wants to go back from the end to the start room.

In general, the quality of the levels were good. They have a mix of platforming and running gameplay, which can be noticed by the distribution of platforms versus the space with flat ground. The average times of the tasks were all less than 1 second, which means the designer can generate several variations of levels for a given design task without problem.

6. CONCLUSION

This work presented StreamLevels, an AI-assisted design tool that works in the game playspace. It generates levels from streamlines that represent player traces. A streamline can be described by either a drawing or a real player trace data. Levels are defined by discretizing the line in rooms. Each room contains the points of the line that hits it. The system has a web interface with a customizable canvas, which gives visual feedback for the player every time a new room is added to the level.

StreamLevels output a general object containing the set of rooms and points defined by the streamlines in the canvas. This output can be used as input for level generators of different game mechanics. To show the potential of this tool, it was developed a constructive level generator for 2D platform games. This generator place, at least, a
platform at the average position of each room. It also ensures that the added platforms are reachable, by adding support tiles in the bottom them if necessary.

The system was evaluated by three experiments. Each one is a different design task. These tasks were performed 10 times each and the best result of each one was analyzed. Results showed the levels were feasible from the start to the end, but the not always in the opposite direction. The average time to perform each task also showed that this is super fast system to prototype level.

The preliminary results pointed out that StreamLevels is a promising tool. Therefore, the future works of this project include adding colors to segments of the stream-line, in order to represent different difficulty. Another important future work is allow the designer to edit a streamline that represents a real player trace, in order to perform local changes to the original level.

REFERENCES


