Division of Digital Media, Ajou University
Interactive Entertainment Lab

nucl.ai

nucl.ai 2016

A 3-Step Generic Framework for Procedural Game Level Generation

Gerhard Grimm* and Gyuhwan Oh**
Interdisciplinary Program of Life Media, Ajou University, South Korea
gerhard.grimm.87@gmail.com*, drgoh@ajou.ac.kr**

Motivation

Procedural content generation (PCG) for game level generation facilitates endless replayability, exploration of new and personal worlds, adaption to the player's preferences and cuts development costs, however, suffers from a lack of control; solvability, controllable difficulty, a feeling of progression and flow as well as macro and micro structures are difficult to integrate [1]. We introduce a 3-step generic framework as helpful guideline for easier controllable PCG for game level generation and show its applicability on a 2D platform game.

Related Work

Whilst classic PCG largely fails to provide control over desired constraints, recent research [2, 3] shows success by separating the process. Shaker et al. [2] create levels for a 2D physics puzzle game by evolving timelines of player actions that are subsequently mapped onto playable design using a simulation step. Dormans [3] creates levels for action adventure games by first creating a mission graph using a graph grammar and subsequently mapping the mission graph onto playable content using a shape grammar. We think the general idea behind [2, 3] of mapping an abstract definition of what defines a level onto playable content is crucial to allow better controllable PCG in level generation. We build on this concept and introduce a more generic framework.

A 3-Step Generic Framework

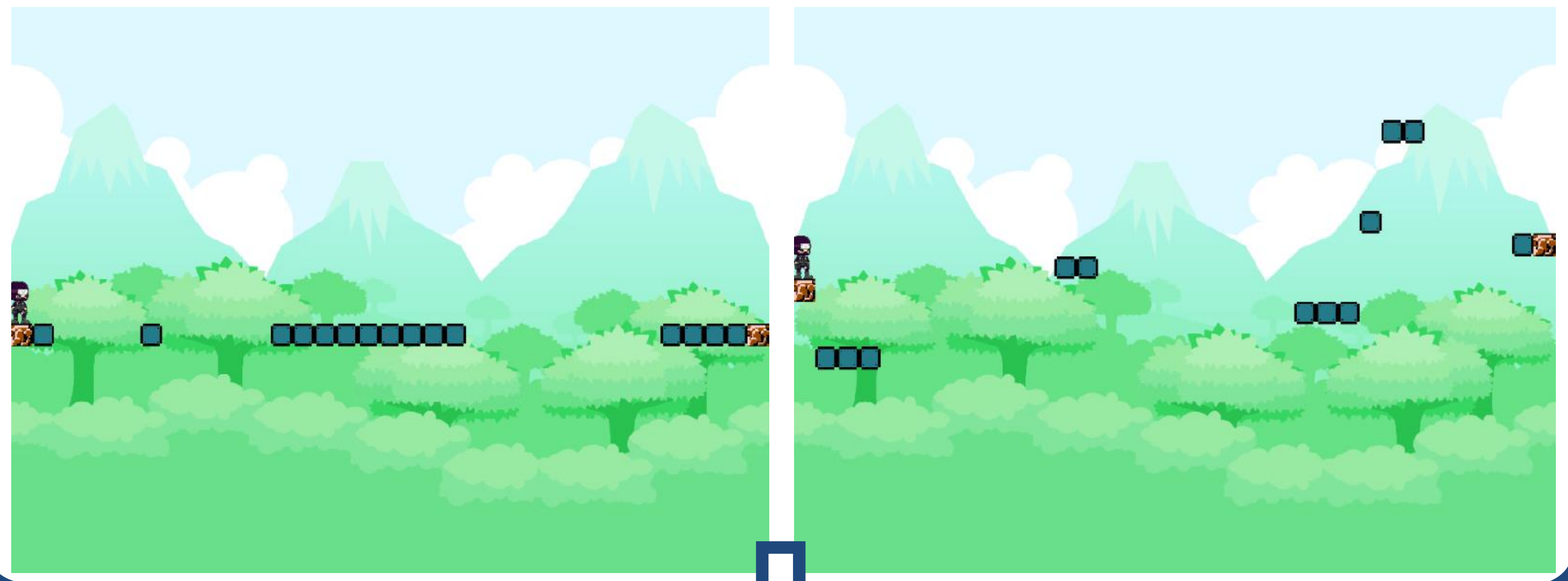
(1) Playthrough creation: create an abstract playthrough through the level using a simple but sufficient representation of what defines a level. The developer can freely choose how to represent and create such a playthrough. As guideline: instead of thinking how the level should look like, think of the player's experience when playing through the level and try to model this experience.

(2) Playthrough simulation: simulate the abstract playthrough using the actual game's rules/physics/controls. The goal is to create an exact playthrough, a detailed, per frame representation of the game's state or all necessary data to exactly recreate this state. This exact playthrough serves as guaranteed solution through the level and can be used as an ingame replay to show the player how the level can be solved.

(3) Content creation: create the actual level content. As long as the exact playthrough stays intact, any content can be added to the game world (e.g., surrounding structures, enemies, obstacles, decorations).

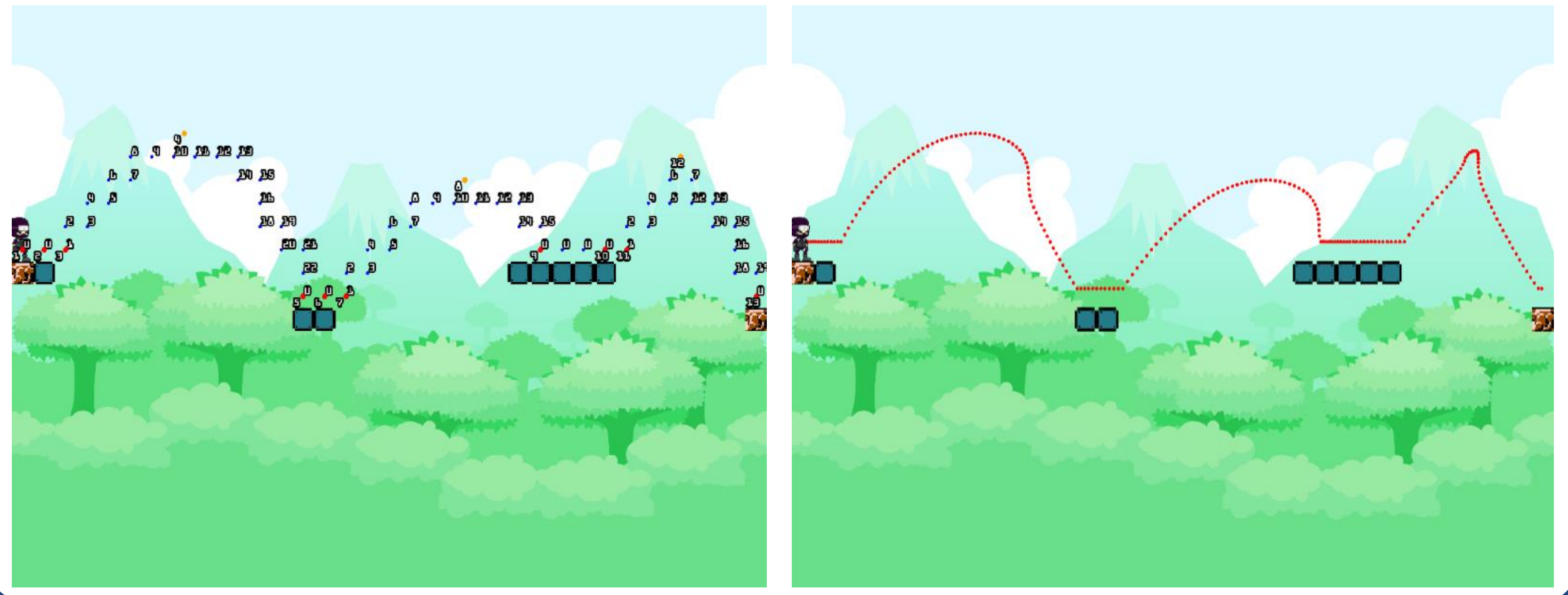
(1) Playthrough Creation

First, an abstract playthrough is created (a sufficient representation of what describes the player's primary experience when playing the level). For simplicity, we start with a simple set of blocks that define a most basic path through the level (using simple heuristics). Another possibility would be the use of a generative grammar to create a timeline of player actions.



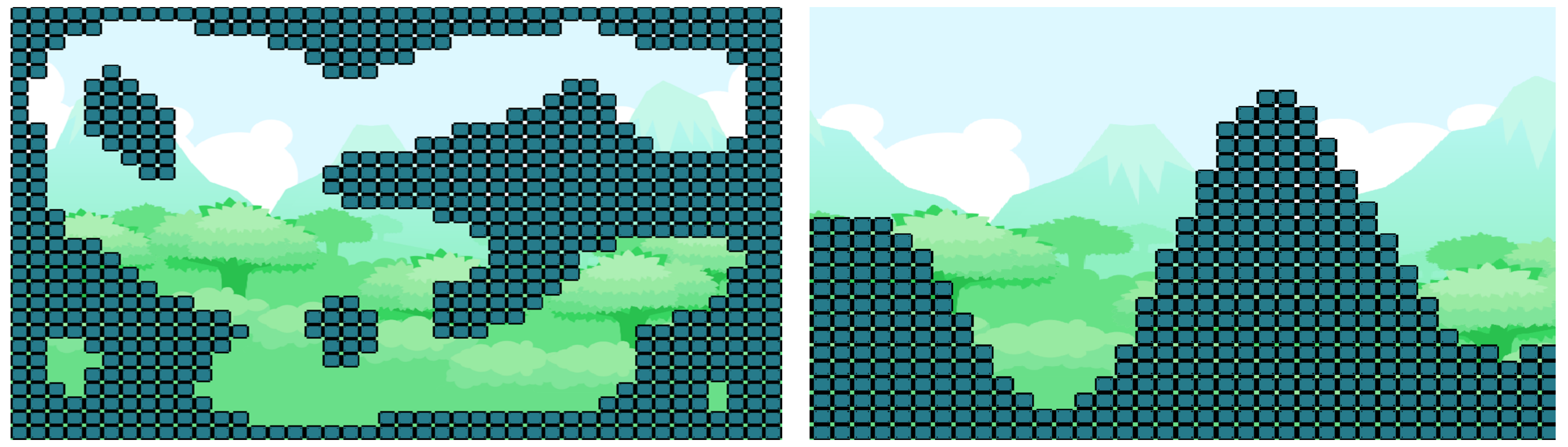
(2) Playthrough Simulation

Next, we simulate the created abstract playthrough using the game's actual physics and controls. Therefor, we implemented A* search that finds an approximate path on grid level (left). Subsequently, we use a rule-based agent that tries to follow the A* path as good as possible, using the game's controls. The resulting exact path as well as the player's timestamps are the desired exact playthrough; it serves as guaranteed solution through the level and can also be used as an ingame replay to show the solution.

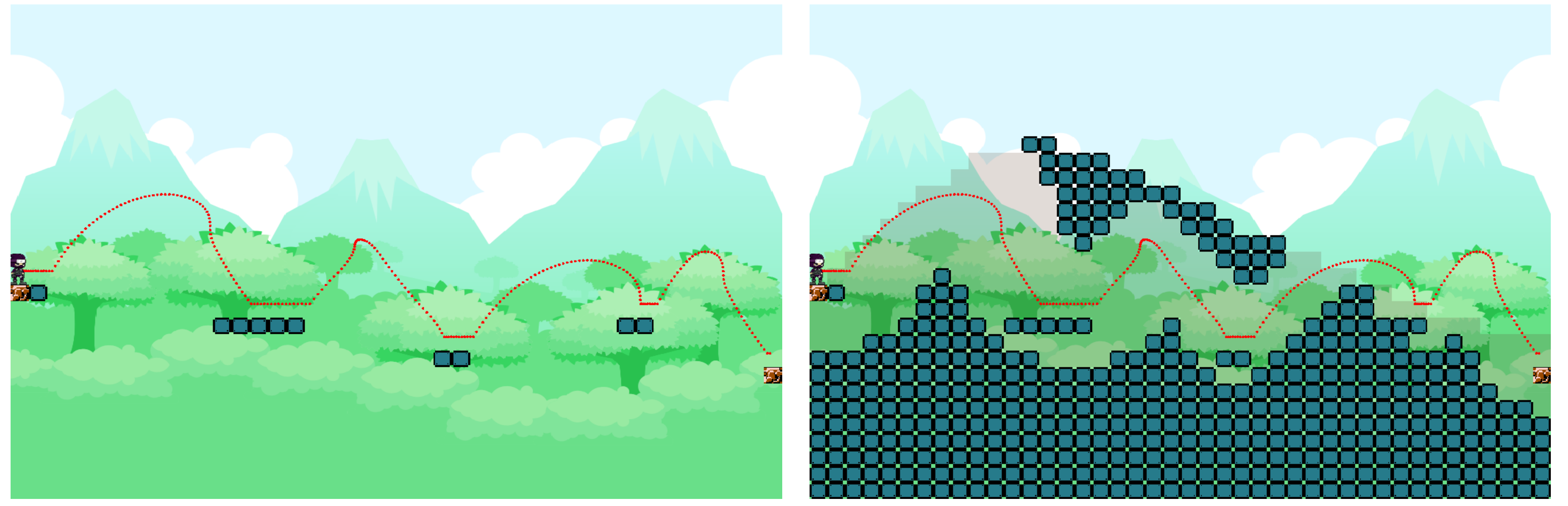


(3) Content Creation

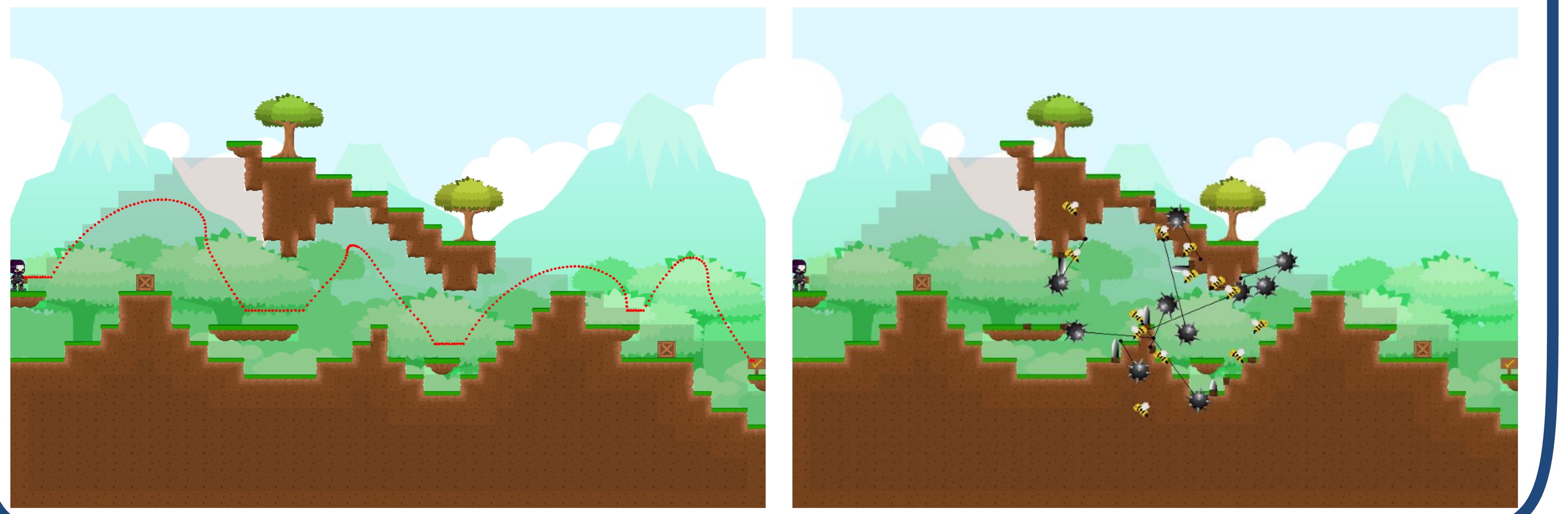
(1) We create a level's macro structure using Cellular Automata (left) or Perlin Noise (right).



(2) Next, we overlay the created exact playthrough (left) with a macro structure (here: Perlin Noise), just making sure not to intersect with the exact playthrough's path (right).



(3) Finally, we replace assets and add decorations (left). Further, we add obstacles (right), also making sure to not destroy the exact playthrough (by checking for possible collisions with the player at the corresponding timestamps when the obstacles cross the path). Thus, solvability is always guaranteed.



Conclusion

Our implementation shows that the introduced framework is feasible. We guarantee solvability, allow difficulty control (abstract playthrough adjustment, AI simulation adaption, more obstacles), model a sense of progression/flow (clear path through a level) and embed levels in an overall macro structure. As long as the exact playthrough stays intact, further (micro) structures are easily added.

References

[1] Julian Togelius, Alex J Champandard, Pier Luca Lanzi, Michael Mateas, Ana Paiva, Mike Preuss, and Kenneth O Stanley. Procedural content generation: Goals, challenges and actionable steps. Dagstuhl Follow-Ups, 6, 2013.
[2] Mohammad Shaker, Noor Shaker, Julian Togelius, and Mohamed Abou-Zleikha. A progressive approach to content generation. EvoApplications, Springer, 2015.
[3] Joris Dormans. Adventures in level design: generating missions and spaces for action adventure games. Proceedings of the 2010 workshop on PCG in games, ACM, 2010.