# Ropossum: An Authoring Tool for Designing, Optimizing and Solving Cut the Rope Levels

**Mohammad Shaker**[1], **Noor Shaker**[2] **and Julian Togelius**[2]
[1]Faculty of Information Technology Engineering, Damascus, Syria
[2]Center of Computer Game Research, IT University of Copenhagen, Copenhagen, Denmark
{mohammadshakergtr}@gmail.com, {nosh, juto}@itu.dk

**Abstract**

We present a demonstration of Ropossum, an authoring tool for the generation and testing of levels of the physics-based game, Cut the Rope. Ropossum integrates many features: (1) automatic design of complete solvable content, (2) incorporation of designer's input through the creation of complete or partial designs, (3) automatic check for playability and (4) optimization of a given design based on playability. The system includes a physics engine to simulate the game and an evolutionary framework to evolve content as well as an AI reasoning agent to check for playability. The system is optimised to allow on-line feedback and realtime interaction.

## 1 Introduction

Physics-based puzzle games are a flourishing genre of games that is receiving increasing attention in the game industry. Typical examples are the very popular games *Angry Birds*, *Tower of Goo*, *Crayon Physics* and *Cut the Rope* which sells millions of copies. Very few attempts can be found, however, on studying these games in academia.

Physics-based puzzle games provide an interesting testbed both for content generation and for investigating the applicability of various AI methods. The generation and testing of playable content for this genre is not an easy task since this presents several distinctive challenges. The physics constraints applied and generated by the different components of the game necessitate considering factors when evaluating the content other than the ones usually considered for other genres – it is far from obvious what makes a good level for such a game. Testing for playability is another issue that differentiates this genre since this can be best done based on a physics simulator.

Few examples are reported in the literature on the design of authoring tools (Smith, Whitehead, and Mateas 2011; Liapis, Yannakakis, and Togelius 2013). In this paper, we present *Ropossum*, an mix-initiative design tool that allows designers to actively interact with an automatic generation and testing system for a physics-based puzzle game. The designer can edit procedurally generated levels, play them or ask an AI agent to solve them. The tool also assists game designers by suggesting modifications so that the final design is guaranteed to be playable. The designer can further define a set of constraints and ask the system to generate levels that satisfy them.



## 2 Testbed Game: Cut The Rope: Play Forever

The testbed game chosen for our experiments is a clone of *Cut The Rope*. There is no open source code available for the game so we had to implement our own clone called *Cut The Rope: Play Forever* that features most of the fundamental characteristics of the original games. The gameplay in CTR revolves around feeding a candy to *Om Nom*. Solving the level puzzle depends to a great extent on timing: specific actions should be taken in certain game states.

## 3 System Architecture

The system we build constitutes of two main modules: an evolutionary framework for procedural content generation (Shaker et al. 2013) and a physics-based playability module to solve the game (Shaker, Shaker, and Togelius 2013). The second module is used both for evolving playable content and for play testing levels designed by humans. We tried to optimize the parameters of the evolutionary system and the AI agent so that the system can respond to the user's inputs within a reasonable amount of time.

### Evolving Game Levels

Grammatical Evolution (GE) is used to evolve the content. The level structure is defined in a Design Grammar (DG) employed by GE. The DG specifies the structure of the levels by defining the positions and properties of the different components of the game and it permits an easy to read and manipulate format (Shaker, Shaker, and Togelius 2013).

### AI Reasoning Agent

First-order logic is used to encode the game state and the physics relationships and properties of the objects (Shaker et al. 2013). The game state is represented as facts specifying the components of the level and their properties. The relationships between the components are represented as rules used to infer the next action to be performed. For example, cutting the rope is performed if doing this action results in an interaction between the candy and at least one other component.

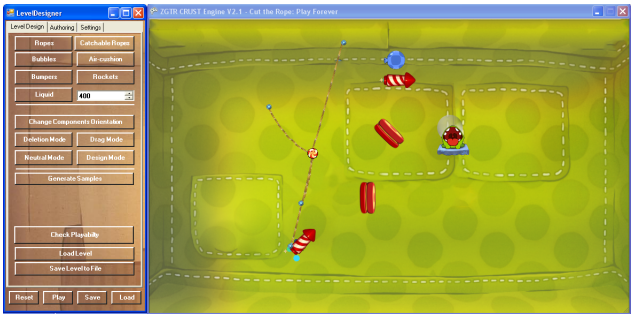


Figure 1: A screenshot for the first set of functions in the authoring tool.

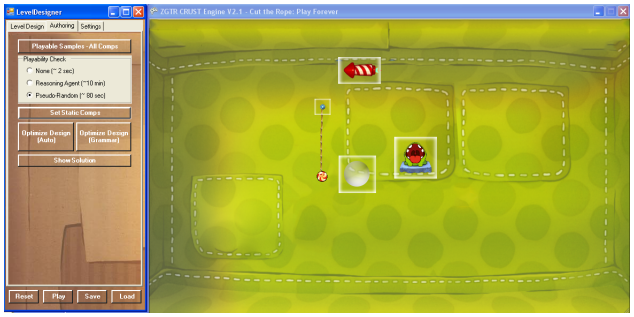
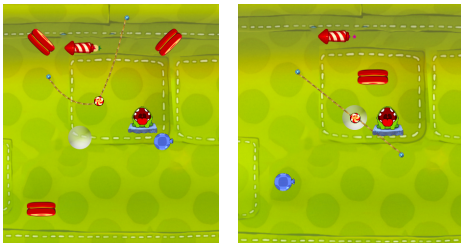


Figure 2: A screenshot from the second set of functions. The components highlighted are the ones fixed by the designers and therefore will not be changed during evolution.



(a) (b)

Figure 3: Two example playable levels evolved satisfying the set of constraints by the designers presented in Figure 2.

### Evolving Playable Levels

The two methods for evolving game design and assessing whether the design is playable are combined together in a framework to evolve playable content. An initial level design, according to the design grammar, is generated and encoded as facts that can be used by the AI agent. Given the game state, the agent infers the next best action to perform. This action is then sent to the physics simulator that performs the action and updates the game state accordingly. The new game state is sent to the agent to infer the next action. If the sequence of actions does not lead to winning the level, the system backtracks. A state tree is generated that represents the actions and states explored. For each action performed, a node in the tree is generated and the tree is explored in a depth-first approach.

### Evolving Actions

In the framework described, a precise playability check is performed for each level design evolved. Although this guarantees that the final design will be playable, the process is time consuming. Therefore, the software provides an additional feature that allows faster, but not as precise, playability testing. In this method, a set of actions according the level design is generated. A given design is evaluated a number of times and in each time we start by a random action selected from this set. The physics simulation then proceeds by randomly selecting an action from the remaining subset according to the new game state. This continues until either the level is won, lost or a predefined timer expires (Shaker et al. 2013).

## 4 Demonstration

Figure 1 presents the user interface of the authoring tool designed. The functions presented can be used to design a new level by choosing from the set of different components and changing their properties. The designer can then play the level or ask the system to do an automatic check for playability. The designer can then chose to further modify or save the level. Figure 2 presents the more advance features. The designer choses whether to evolve designs that are not necessarily playable, designs that are guaranteed to be playable (using the AI agent) or do a faster check for playability using evolved actions. The designer can query the system about the actions performed to solve the level which are then demonstrated via a play-through. The interface also allows modifying the design grammar permitting the evolving of different patterns. This can be done by directly editing the grammar or through the visual interface by specifying a set of constraints that are later embedded in the grammar. Figure 2 present an example of the latter case where the highlighted components are the ones fixed by the designers. Examples levels generated to complete these designs so that it becomes playable are presented in Figure 3.

## References


Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2013. Sentient sketchbook: Computer-aided game level authoring. In *Proceedings of ACM Conference on Foundations of Digital Games*.

Shaker, M.; Sarhan, M.; Al Naameh, O.; Shaker, N.; and Togelius, J. 2013. Automatic generation and analysis of physics-based puzzle games. In *IEEE Conference on Computational Intelligence and Games*.

Shaker, M.; Shaker, N.; and Togelius, J. 2013. Evolving playable content for cut the rope through a simulation-based approach. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Smith, G.; Whitehead, J.; and Mateas, M. 2011. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):201–215.