# PaTux: An Authoring Tool for Level Design through Pattern Customisation Using Non-Negative Matrix Factorization

**Mohamed Abou-Zleikha**[1] **and Noor Shaker**[2]
[1]Department of Electronic Systems, Aalborg University, Aalborg, Denmark
[2]Center for Computer Games and Interaction Design at the IT University of Copenhagen, Copenhagen, Denmark
nosh@itu.dk, moa@es.aau.dk

**Abstract**

We present a demonstration of PaTux, an authoring tool for designing levels in *SuperTux* game through combining patterns. PaTux allows game designers to specify the design of their levels using patterns extracted from training level samples. The Non-negative Matrix Factorisation (NMF) method is utilised to approximate pattern and weight matrices from the training data. The patterns are visualised for designers to choose from and the changes made on the level structure are visualised in realtime. The designer can also specify the weight of each pattern permitting exploration of a wider variety. The data used to train the model can also be specified by the designer resulting in learning a new set of patterns. The system also suggests variations for a given design. When the designer is satisfied with the design, the system allows loading the resultant level in the game to be played.

## 1 Introduction

Level design is an essential part of the game design process. Designers usually put much of their time and effort in creating an interesting, playable level. This process is typically iterative between generating and testing until the designer is satisfied with the result. The last few years have witnessed the flourish of independent video games. This has increased the demand for low-budget tools that are affordable and easy to use. Partly as a response to this need, there has recently been increasing interest in academia given to the creation of authoring and mixed-initiative tools. Some notable works include *SkectchaWorld* a tool for modelling 3D world (Smelik et al. 2010), *Tanagra* a mixed-initiative design tool for 2D platformer (Smith, Whitehead, and Mateas 2010), *Sentient Sketchbook* a tool that aids the design of game maps (Liapis, Yannakakis, and Togelius 2013), and *Ropossum* a tool that permits automatic generation and testing of physics-based puzzle games (Shaker, Shaker, and Togelius 2013).

In this paper, we introduce *PaTux*, an authoring tool for level generation through combining patterns. The patterns used are extracted using the NMF method that is applied on a large dataset of training level samples. The tool permits selecting patterns for different game components, customising their weights to provide variations, realtime visualisation of edits and it permits playing the created design. The designer can also retrain the model on his own dataset which results in the extraction of a new set of patterns that represents the style of the generator used to create the truing set.



## 2 Testbed Game: SuperTux

SuperTux (SuperTux Development 2003) is a free open source 2D platform game inspired by Super Mario Bros. We use the source code available for the Platformer AI Competition (Shaker, Togelius, and Yannakakis 2013) to generate the training data and to implement the tool.

## 3 The NMF Model

Non-Negative Matrix Factorization (NMF) is a technique for representing non-negative data as a linear parts-based combinations (Lee and Seung 1999). The non-negativity property makes NMF results easy to inspect since the representation is purely additive.

Given a non-negative data matrix $\mathbf{V}$, NMF finds an approximate factorization $\mathbf{V} \approx \mathbf{WH}$ where $\mathbf{W}$ and $\mathbf{H}$ are non-negative factors. The objective function of NMF is to minimize the Euclidean distance between each column of the matrix V and its approximation $\mathbf{V}' = \mathbf{WH}$. The columns of the matrix $\mathbf{W}$ can be seen as parts (i.e. building blocks) of the data while the coefficient vectors describe how strongly each part is present.

Utilising NMF for level generation corresponds to the extraction of the patterns matrix $W$ from a set of training levels $V$. In order to construct the $V$ matrix, the structure of the levels is converted into one dimensional vectors of numbers corresponding to platform structure $V_p$, hills $V_h$, gaps $V_g$, items $V_t$ and enemies $V_e$. The final model consists of five independent NMF models constructed for each level component. Fig. 1 illustrates the framework followed. More details can be found in (Shaker and Abou-Zleikha 2014).

## 4 The NMF Generator

Once the model is built and the $Ws$ and $Hs$ matrices are estimated, the model can be used to generate a new level. A coefficient vector $h'$ representing the weights applied on the estimated pattern matrices $Ws$ is set as input to the model (see Fig. 1). Reconstructing a single levels corresponds to estimating each of its basic components independently. Consequently, the results are $< \mathbf{V'_p}, \mathbf{V'_h}, \mathbf{V'_g}, \mathbf{V'_t}, \mathbf{V'_e} >$ that

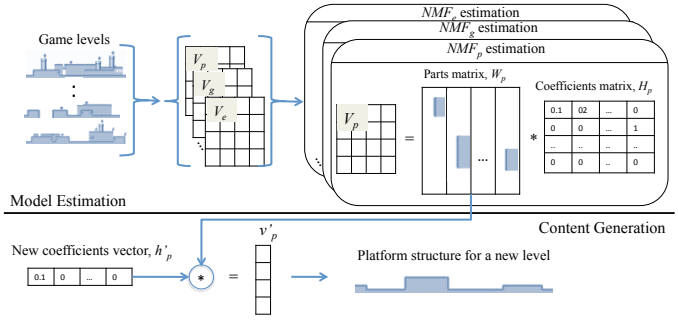


Figure 1: The framework implemented to estimate the model and to use it to generate new content. The upper part shows the steps followed to construct the five NMFs. Levels are first converted into sequences of numbers and arranged in the $V$ matrices that are used as input to estimate the $W$ and $H$ matrices for each level component. In the content generation phase, and to create a new level, a coefficient vector is used as input. The result is a vector representing the content of the new level.

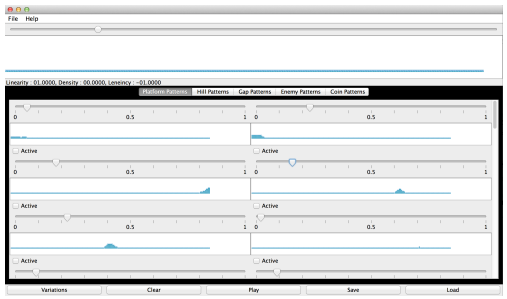

Figure 2: A snapshot from the interface provided with the tool. The upper part presents the initial level design which is modified in realtime to provide visual feedback about designer's choices. Patterns of different components are presented in the lower frames. The designer can activate/deactivate each pattern and change its weight and edits will be directly reflected in the upper frame.

represents a newly generated level. These vectors are then parsed and composed to create a level in its 2D map format that can be loaded and played.

## 5 Demonstration

Fig. 2 presents the user interface of the authoring tool. Five tabs are used to show patterns of the game components. Designers can activate a number of patterns, change their weights and reflections of the edits are directly visualised. Designers can also navigate between the taps to enrich their design by adding items from different components. Fig. 3 presents a snapshot for the same level presented in Fig. 2 after performing a number of edits and adding different artifacts including hills, enemies and coins.

After finalising a design, the tool provides the option of visualising similar variations such as the ones presented in Fig. 4. These variations are generated by assigning low weights for a randomly chosen set of unselected patterns. At any point during the level design process, the designer can play the level to get more thorough insights on the experience it elicits.

Figure 3: The final design of a level after performing several edits and adding patterns from different components.

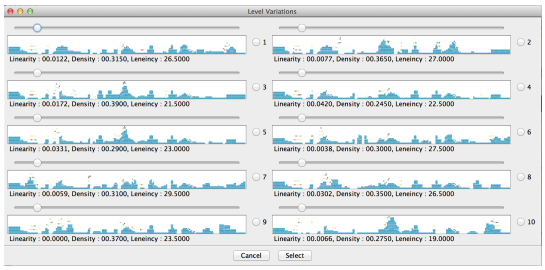

Figure 4: Ten variations for the level presented in Fig. 3.

The tool also allows retraining the models on another set of levels leading to the extraction of new patterns. The training set can be chosen by the designer and can be another set of automatically generated levels, or a number of handcrafted levels. In both cases, representative patterns of the style of the training generator will be extracted.

## 6 Acknowledgement

The research is supported in part by the Danish Research Agency, Ministry of Science, Technology and Innovation; project "PlayGALe" (1337-00172) and supported in part by the Danish Council for Strategic Research of the Danish Agency for Science Technology and Innovation under the CoSound project, case number 11-115328. This publication only reflects the authors views.

## References

Lee, D. D., and Seung, H. S. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–791.

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

Shaker, N., and Abou-Zleikha, M. 2014. Alone we can do so little, together we can do so much: A combinatorial approach for generating game content. In *AIIDE*.

Shaker, M.; Shaker, N.; and Togelius, J. 2013. Ropossum: An authoring tool for designing, optimizing and solving cut the rope levels. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE). AAAI Press*.

Shaker, N.; Togelius, J.; and Yannakakis, G. 2013. Platformer AI Competition.

Smelik, R.; Tutenel, T.; de Kraker, K.; and Bidarra, R. 2010. Integrating procedural generation and manual editing of virtual worlds. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, 2. ACM.

Smith, G.; Whitehead, J.; and Mateas, M. 2010. Tanagra: A mixed-initiative level design tool. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, 209–216. ACM.

SuperTux Development. 2003. SuperTux.