Procedural terrain generator for platform games using Markov chain

Gleidson Mendes Costa
Vision and Image Processing Laboratory - VIPLab
Universidade Federal do Maranhão - UFMA
São Luis-MA, Brasil
gleidsonmcosta@gmail.com

Tiago Bonini Borchartt
Vision and Image Processing Laboratory - VIPLab
Universidade Federal do Maranhão - UFMA
São Luis-MA, Brasil
tiago.bonini@ufma.br

***Abstract*— The game design is a complex and huge task and game companies spend much time to create a beautiful and exquisite design for a video game. The terrain is the most extensive part of a game. Some games need to present different terrains on each level so that the game does not become repetitive to the players. Procedural Terrain Generation is a technique that can streamline much of this work using algorithms to create new terrain randomly. This work presents a methodology for procedural terrain generation for 2D platform games. The approach developed was based on the use of Markov models, specifically the hidden Markov chains, to produce new terrain probabilistically. The results show that the technique presented here is able to generate a 2D platform game terrain.**

Keywords- 2D platform game; procedural terrain generation; Markov models; hidden Markov chains

I. INTRODUCTION

The game design is a complex and huge task, it requires putting in a lot of manpower and material resources and spend some time in order to create a beautiful and exquisite design of good games.

One of the most essential aspects of most video games is the terrain that the player will traverse. Procedural Terrain Generation (PTG) can streamline much of this work using pseudo-random algorithms. The term PTG refers to the creation of content by an automated system, rather than being produced manually [1].

There are many of effective techniques currently that can be applied to procedural game content generation. However, the main algorithms are based on noise function. The Diamond-Square Algorithm proposed by [2] have fractal nature based on recursive subdivision. In [3], several noise functions are compared to generating terrain texture: Value Noise, Perlin Noise, Simplex Noise and Whorley Noise. The authors conclude that each technique present a weak point, e.g., Simplex Noise is very difficult to implement in a game context, Value Noise are very simple, but need a secondary interpolation function based on fractional Brownian motion to generate a good result.

2D platform games are on the rise due to recent big hits in mobile operating systems, e.g., Angry Birds and Super Mario Run, each with more than 100 bi downloads [4][5]. A platform game is a video game genre of action game where the player controls a character or avatar to jump between suspended platforms and avoid obstacles.

In [6], the authors presented a hierarchical approach to generating 2D platform video game maps using multi-dimensional Markov chains for learning, sampling and clustering for finding high-level structures. Their approach need a training data instead of domain knowledge to generate a new map.

The objective of this work is present a methodology for procedural terrain generation for 2D platform games. The approach developed was based on the use of Markov models, specifically the hidden Markov chains, to produce new terrain probabilistically.

The game designer can define the terrain behavior of the game by reporting a transition matrix, where each state is a terrain element.

Finally, a filtering method was applied to verify the consistency and viability of the terrain generated. Therefore, such tool may aid the game designer during the game development and improve the overall quality of the game.

In the next section, we present the fundamental concepts to understand the developed methodology. In Section III the PTG methodology developed are presented. Section IV discusses the obtained results. The Section V shows the conclusions and future works.

II. BACKGROUND

A. 2D Platform Game

Platform games originated in the early 1980s, with 3D successors popularized in the mid-1990s. The term describes games where jumping on platforms is an integral part of the gameplay and came into use after the genre had been established [7].

One of the most famous platform game is "*Super Mario World*" (SMW). This game was launched in 1990, and now is a classic and successful game. SMW is still used as a template for many games titles of the same genre. It is a platform-based game, where the main character must follow a path picking items and destroying or deviating enemies.

The return of 2D games would be related to aspects of nostalgia or aesthetics. The 2D style is not necessarily a technical limitation, since just as hardware has evolved and transformed the 3D gaming experience, these techniques may well be applied in two-dimensions [8].

According to [9], mobile games emerge as the favorite of the public, with 43.6% choosing smartphones as best to play. In them, the games need to be very "light", which does not mean that they must be simple, but that they cannot use high realism to be accepted in the market. This combination of factors enabled the return of 2D games, in particular platform games.

B. Procedural Terrain Generation (PTG)

Player demand for more content is increasing as games grow in complexity and scope. Traditionally developers have hand-crafted content for games, requiring a substantial investment of time and resources.

Procedural content generation methods are used to create content algorithmically instead of manually [10]. PTG are specific techniques for the generation of terrain. Virtual terrains have an important role in computer graphics, and their applications range from landscape design and flight simulators to movies and computer games. A terrain is the dominant visual element of the scene, or it plays a central part in the application.

Procedural techniques are a popular choice in game development because of the simple implementation and wide range of terrains they provide when a few parameters are changed [11].

The desire for automatic terrain generation stems from the goal of provide a new content to player without a large investment of developer resources.

In [12], the authors cite the properties of a PTG technique:

- Novelty: contains an element of randomness and unpredictability;
- Structure: is not merely random noise, but contains larger structures;
- Interest: has a combination of randomness and structure that players find engaging;
- Speed: can be quickly generated;
- Controllability: can be generated according to a set of natural designer-centric parameters.

In 2D platform games, PTG techniques can be used to generate the path that the character will have to pass to complete the level. When the player learns the static path of a level, the game tends to get less interesting to him. The PTG makes the game dynamic and challenging at every new level.

C. Markov chains

A stochastic process is a family of random variables representing the evolution of a value system over time. In cases of discrete time, as opposed to continuous time, the stochastic process is a sequence of random variables, such as a Markov chain [13].

Define X(t) a random variable evolving as a function of time. Let X(0) = 1, X(1) = 6, X(2) = 2, X(3) = 5. This evolution process is called Markovian if its evolution does not depend on its past, but only on its current state.

A theoretical model called "Markov Model" or "Markov chain" can model a Markov process.

Markovian chains modeled a set of states. These states are co-related in a matrix, where each combination is a probabilistic chance of a state go to another. The set of probabilistic values of the output and arrival at a new state become a state transition matrix. There are two types of Markovian models: observable and hidden.

1) Observable Markov chain

The evolution of the Markov process can be presented by a state transition graph that shows the structure of the process according to the following rules [14]:

- States are represented by nodes that belong to the alphabet '*S*' of states, $S = \{s_1, s_2, \ldots, s_n\}$;
- The transitions (possibilities of passing from one state to the other state) are represented by (directed) edges, they are weighted by their probabilities. The probabilities are grouped in a transition matrix '*A*':
$$A = \{a_{i,j} = P(s_i|s_j)\}, where\ \sum_{j=1}^{N} a_{i,j} = 1 \quad (1)$$
- The starting probabilities are the probabilities of starting in one state or another. They are grouped into a startup vector '$\prod$':
$$\prod = \{\pi_i = P(s_i)\}, where\ \sum_{i=1}^{N} \pi_i = 1 \quad (2)$$

A model λ is observable because states are directly observable, and a transition matrix *A* and an initialization vector $\prod$ characterize the model, where:

$$\lambda = \{\prod, A\} \quad (3)$$

2) Hidden Markov chain

In a Hidden Markov Model (HMM) the states $S = \{s_1, s_2, \ldots, s_n\}$, are not observable. However, they emit observable signals $O = \{o_1, o_2, \ldots, o_t\}$ that are weighted by their probabilities.

An HMM is characterized by Eq. (5), a transition matrix *A*, an initialization vector $\prod$ and an emission matrix *B*, defined by:

$$B = \{b_i(o_k) = P(o_k|s_i)\}, where\ \sum_{j=1}^{T} b_i(o_j) = 1 \quad (4)$$

where, b_i is probability of the state s_i to emit the signal o_k.

$$\lambda = \{\prod, A, B\} \quad (5)$$

III. METHODOLOGY

The methodology is based on the follow steps:

1. Define the alphabet states $S = \{s_0, s_1, \ldots, s_n\}$;
2. Define startup probabilities vector: $\prod = \{\pi_i = P(s_i)\}$;
3. Define the trasition matrix probabilities: $A = \{a_{i,j} = P(s_i|s_j)\}$;
4. Sorts the initial state conform $\prod$;
5. Sorts the next state conform *A*;
6. Validate generated state, if not valide, repeat step 5;
7. Repeat steps 5 and 6 until reach the end of the level.

The first 3 (three) steps are dependent from the application, where the game designer defines the components that compose the terrain.

The game designer can change the elements of the alphabet in the first step. The elements entered by the designer are used to generate terrain.

The probability that each terrain of the alphabet has to appear as the first element of the terrain is defined in the second step. The game designer is free to choose the probability values of vector $\prod$ and matrix A.

The step 4 sets the initial state. The steps 5 and 7 build the terrain of the game, following the probability of the Markovian process observable, explained in Section II.

The step 6 is necessary to avoid sequences that escape the expected behavior of the terrain, e.g., distances that the character cannot jump.

After create the platforms to the current level at the game, another Markov process are executed to populate the scenery with decorative elements. This process is based on HMM. An emission matrix is defined based on the alphabet of states. Each component generated by the previous steps receive a decorative element predetermined by the game designer and sorted conform HMM.

IV. RESULTS

The methodology was implemented in Unity game engine [15]. An experiment was conducted with 12 different possible states. The states can be seen in Figure 1.

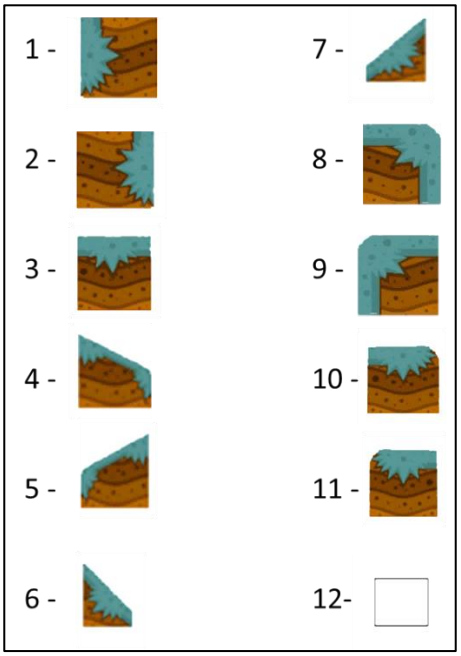

Figure 1. Alphabet defined as sample.

The follow initialization vector ∏ was used in the experiments with the values in Table I:

TABLE I. INITIALIZATION VECTOR ∏.

1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	1	0	0	0

The initialization vector probabilities make the level always start with the state 9.

The transition matrix A was defined by Table II.

Figures 3, 4 and 5 show the result of applying the technique to generate platforms for a 2D game, based on Markov chain. The game was developed in Unity game engine.

The scenes shown in Figures 3 and 4 were produced without the step of inserting decorative elements. The scene presented in Figure 5 represents the application of the complete method developed.

The decorative elements selected to experiments are showed in Figure 2.

The emission matrix B was defined with the values in Table III.

TABLE II. TRANSITION MATRIX *A*.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	0.4	0	0	0	0.1	0	0.1	0	0.4	0	0	0
2	0.2	0.2	0	0	0	0	0	0	0	0	0	0.6
3	0	0	0.2	0.1	0.1	0.1	0.1	0.2	0	0.2	0	0
4	0	0	0.2	0.2	0	0	0	0.2	0	0.2	0	0.2
5	0	0	0.2	0.2	0	0	0	0.2	0	0.2	0	0.2
6	0	0	0.2	0	0	0.4	0.2	0	0	0	0	0.2
7	0	0	0.2	0	0	0	0.4	0.2	0	0.2	0	0
8	0	0.5	0	0	0	0	0	0	0	0	0	0.5
9	0	0	0.4	0.1	0.1	0	0	0.2	0.2	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0.4	0.1	0.1	0	0	0.2	0.2	0	0	0
12	0	0	0	0	0	0	0	0	0.5	0	0.5	0

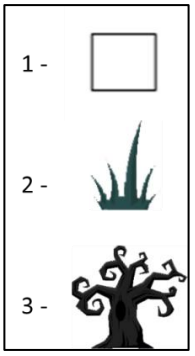

Figure 2. Decorative elements.

TABLE III. EMISSION MATRIX B.

	1	**2**	**3**
1	1	0	0
2	1	0	0
3	0	0.5	0.5
4	0.2	0.4	0.4
5	0.2	0.4	0.4
6	0.2	0.4	0.4
7	0.2	0.4	0.4
8	0.2	0.4	0.4
9	0.2	0.4	0.4
10	0.2	0.4	0.4
11	0.2	0.4	0.4
12	1	0	0

Figure 3. Sample of generated level by Markov chain.

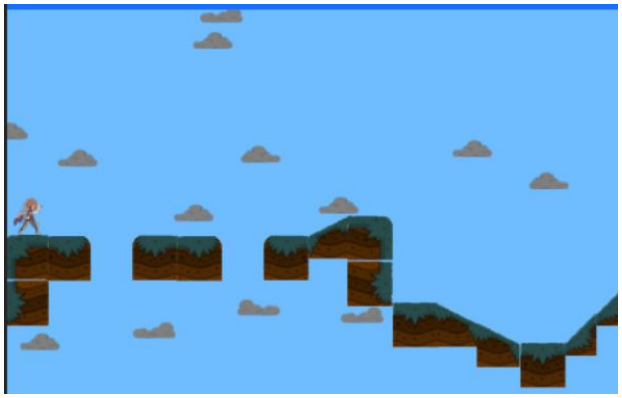

Figure 4. Sample of generated level by Markov chain.

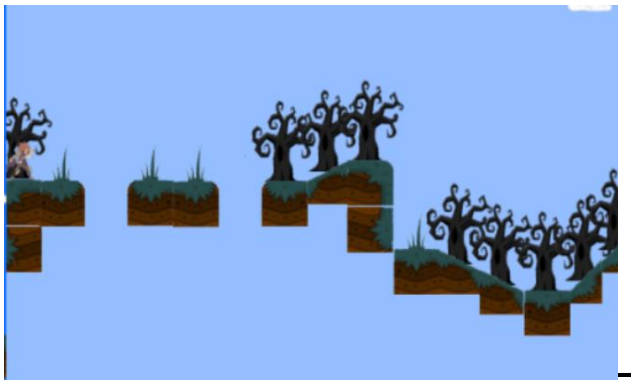

Figure 5. Complete level generated by Markov chain.

V. CONCLUSION

In this paper, we apply a stochastic model for the creation of terrain for 2D platform games. The PTG method is based on the probability of a new state is created from the current state, where each state represents a different segment of terrain. The land tends to connect probabilistically.

However, only an application of this probabilistic method can render the result unfeasible in a game, as it can create new states that hurt consistency of the terrain. To avoid inconsistent terrain, a rule filter was applied.

The hidden Markov model was applied as a second step to populate the terrain created with decorative game entities such as trees, shrubs, rocks, etc.

The methodology shows good results, as can be seen in Figures 3, 4 and 5. The method applied for filtering the new states produced to maintain the consistency are useful and the use of Markov chains alone was insufficient to obtain a productive result.

Thus, advances must be made to improve the use of hidden Markov chains, because despite obtaining a useful result, some errors occur, e.g., the instance of elements in places that should not exist, as in the case of the instance of a decorative element on top of an empty terrain segment.

REFERENCES

[1] T. J. Rose and A. G. Bakaoukas, “Algorithms and Approaches for Procedural Terrain Generation - A Brief Review of Current Techniques,” 8th International Conference on Games and Virtual Worlds for Serious Applications, Barcelona, 2016.

[2] G. S. P. Miller, “The definition and rendering of terrain maps,” ACM SIGGRAPH Comput. Graph., vol. 20, no. 4, pp. 39-48, 1986.

[3] T. Archer, “Procedurally Generating Terrain,” 44th Midwest Instruction and Computing Symposium, Duluth, 2011.

[4] Angry Birds Classic – Apps no Google Play, Available at https://play.google.com/store/apps/details?id=com.rovio.angrybirds, Accessed in 03 ago 2018.

[5] Super Mario Run – Apps on Google Play, Available at https://play.google.com/store/apps/details?id=com.nintendo.zara, Accessed in 03 ago 2018.

[6] S. Snodgrass and S. Ontañón, “A Hierarchical MdMC Approach to 2D Video Game Map Generation,” Proceedings of The Eleventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2015.

[7] S. Rabin, “Introduction to Game Development,” Vol. 1, Editora Cengage Learning, 2012.

[8] A. Santee, “GS2D e C++: 2D da nova geração em alto nível,” In Ponto V: Programação de Jogos Profissionais, Available at: http://www.pontov.com.br/site/cpp/64-gs2/225-gs2d-e-c-2d-da-nova-geracao-em-alto-nivel, Accessed in 03 ago 2018.

[9] W. Wakka, “Brasileiro joga mais no smartphone e ainda está na geração passada de consoles,” Available at: https://canaltech.com.br/games/brasileiro-joga-mais-no-smartphone-e-ainda-esta-na-geracao-passada-de-consoles-113590/, Accessed in 03 ago 2018.

[10] J. Togelius, G. N. Yannakakis, K. O. Stanley and C. Browne, “Search-based procedural content generation: A taxonomy and survey,” IEEE Transactions on Computational Intelligence and AI in Games, Vol. 3, No. 3, pp. 172–186, 2011.

[11] J.-D. Genevaux, É. Galin, É, Guérin, A. Peytavie and B. Benes, “Terrain Generation Using Procedural Models Based on Hydrology,” ACM Transactions on Graphics, Vol. 32, No. 4, 2013.

[12] J. Doran and I. Parberry, “Controlled Procedural Terrain Generation Using Software Agents,” IEEE Transactions on Computational Intelligence and AI in Games, Vol. 2, No. 2, pp. 111-119, 2010.

[13] R. G. Gallanger, “Stochastic Processes: Theory for Applications.” 1st Edition, Hardcover, 2014.

[14] C. Deng and P. Zheng. “A new hidden markov model with application to classification,” In Intelligent Control and Automation, 2006.

[15] Unity, Available at: https://unity3d.com/pt, Accessed in 03 ago 2018.