Procedural Playable Cave Systems based on Voronoi Diagram and Delaunay Triangulation

Aitor Santamaría-Ibirika, Xabier Cantero, Sergio Huerta, Igor Santos, Pablo G. Bringas
S³lab
University of Deusto
Bilbao, Spain
{*a.santamaria, xabier.cantero, shuerta, isantos, pablo.garcia.bringas*}*@deusto.es*

***Abstract*—The volumetric approach for terrain representation is a technique used by several video-games and other graphic applications to manage both surface and geological data from the virtual world. To enhance the exploration experience and due to the amount of data required by this approach, volumetric terrains are usually generated with procedural methods. Nevertheless, one of the main issues of those methods is the generation of cave systems with playable features and a natural appearance.**

In this paper we propose a new method to generate playable cave systems for 2D and 3D volumetric terrains, based on Voronoi diagrams and Delaunay triangulations. Our approach is completely customizable by the designer by a set of parameters directly related to the cave itself avoiding technical concepts. Additionally, the method runs in a completely independent way, with no interactive steps.

***Keywords*-procedural terrain; procedural caves; volumetric terrain;**

I. Introduction

The terrain representation issue has been handled using different approaches, such as height maps (which can represent only the surface data of the terrain) or volumetric terrains (which contains both surface and geological data). Usually, video-games which use volumetric terrains base their mechanics and playability on the exploration of the virtual world. So, the exploration of the underground structures of those terrains is a crucial feature for them. Additionally, procedural methods are widely used by those video-games to generate infinite and unexpected terrains, improving the exploration experience.

One of the main issues in this situation is the procedural generation of underground cave systems with a natural appearance, but keeping in mind the playable features to create interesting structures to be explored by the players.

In this paper, we present a novel approach to procedurally generate cave systems for two-dimensional and three dimensional virtual worlds. Our method is based on Voronoi diagrams and Delaunay triangulations, and it enables the designer to establish the type of the cavities a cave system should contain to be playable (e.g. treasure areas, encounters with enemies, merchants or exits) and their characteristics (e.g. the depth or the number of connections with other stances). Additionally, the input data is completely related to the cave itself, avoiding technical concepts related to the mathematical method used.

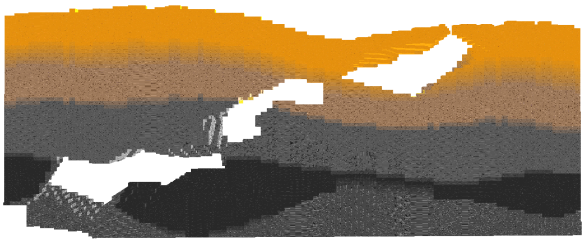


Figure 1. Slice of terrain with a cave generated with our approach.

The remainder of this paper is organized as follows: First, section II presents a review of the existing literature about this topic. Next, section III makes an overview of our method, and section IV presents the mathematical methods used. Section V describes the algorithm. Then, the results obtained by our method are displayed in section VI, discussing them in section VII. Finally, the paper ends with a conclusion and a proposal for future work in section VIII.

II. Related Work

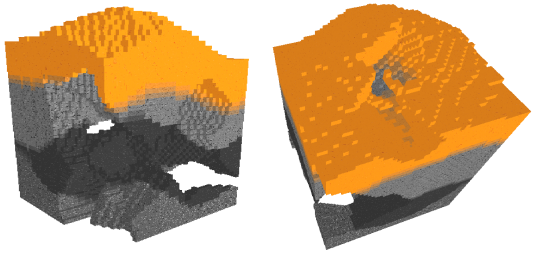
Procedural generation is a popular topic, existing several approaches for procedural terrains, cities or buildings [1]. On the contrary, procedural cave generation is an area not extensively researched.

Boggus and Crawfis proposed a method which simulates the action of the water and the erosion of the rock [2], [3]. This method limited the caves to only two surfaces (the floor and the roof). They overcame this limitation in their following research [4], obtaining more realistic results.

Peytavie *et al.* [5] presented a complete framework for generating volumetric terrains based on voxels. Their system simulates Thermal and fluvial erosion, generating caves and complex rock formations.

Johnson *et al.* [6] proposed a new approach based on cellular automatons which generates infinite cave levels in real time. The main issue of this method is that it can not generate three dimensional caves.

Cui *et al.* [7] proposed a method to generate caves using three dimensional noise, and they improved it generating stalactites and the stalagmites [8]. Those structures have been also researched by Tortelli and Walter [9].



(a) (b)

Figure 2. A terrain plot with a cave.

III. Our approach

Against this background, our method presents the following features:

Playability based on points of interest:
Resulting cave systems are constructed using a set of points of interest (POI). A POI is a point of the cave with a special playable interest, such as a treasure chamber or an encounter with a monster. Those points are defined by the designer, establishing features such as their depth in the terrain or the relationship with other POIs.

Gallery based cave systems:
Our method generates the cave with underground galleries, which are constructed using the relations between POI types (defined by the designer). This feature enables the process to generate cave systems with a playable criteria (e.g. after an encounter against enemies, the player will found a treasure).

Usability:
The input parameters are transparent to the technical methods used in the process. That is, all the parameters are related to the cave system itself.

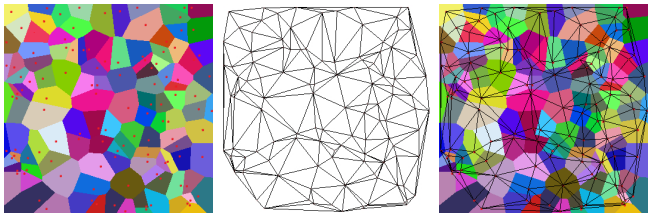
Autonomous execution:
The execution of the generator is completely autonomous. It does not require interactive steps, so, it can be executed in the systems of the players, generating different cave systems in each execution.

Based on our reality:
The results of our proposed approach are not simulations of the real world. In spite of that, the cave systems present a natural appearance and its realism is similar to the realism of the caves used in other interactive applications.

We can observe some caves generated with our approach in figures 1, 2 and 8.

IV. Materials and Methods

The generation method is based on two mathematical constructions: Voronoi diagrams [10] and Delaunay triangulations [11].



(a) (b) (c)

Figure 3. Voronoi diagram (a), Delaunay triangulation (b) and their relation (c). They are constructed using 120 random seeds in a two dimensional space.

A. Voronoi Diagrams

A Voronoi diagram [10] is a way of dividing space into regions (called Voronoi cells) basing the division on a set of points (called seeds). For each seed there will be a corresponding region consisting of all points closer to that seed than to any other. Figure 3(a) shows a two dimensional Voronoi diagram. They have been widely used in scientific and technical areas for several purposes [12], including procedural terrain generation [13].

B. Delaunay Triangulations

A Delaunay triangulation [11] is a mathematical technique to triangulate the space using a set of points. Formally, it is a triangulation such that the circumcircle of any triangle has no point inside (circumsphere of any tetrahedron for 3D). Higher dimension triangulations are also available. Figure 3(b) shows a two dimensional Delaunay triangulation. This technique has been extensively used for mesh generation [14].

The Delaunay triangulation corresponds to the dual graph of the Voronoi diagram. That is, the centers of the circumcircles of the triangles in the Delaunay triangulation are the vertex of the Voronoi cell of the Voronoi diagram. Figure 3(c) shows this relation overlapping a Voronoi diagram and the Delaunay triangulation of their seeds. The consequence of this characteristic is that the points connected in the triangulation have their Voronoi cells adjoining [1].

V. Cave Generation Process

A. Input Data

Before the generation of the terrain, the designer must establish a set of input parameters. The input data is divided in three main sets of parameter: the descriptors of the POIs, the descriptors of the POIs relationships and the global parameters.

First, the descriptors of the POI types are provided:

- **ID:** Unique ID of the POI type.
- **Name:** The name of the POI type.

[1]In the rest of the paper we call neighbor cells to each pair of cells of the Voronoi diagram with their seeds interconnected in the triangulation.

Table I
EXAMPLE OF A SET OF DESCRIPTORS OF THE POI TYPES

ID	Name	Location	Depth min/opt/max	Parameters	Cavity size min/opt/max	Branches min/opt/max
0	Entrance	surface		start(1)	0.5 / 1 / 1	1 / 1 / 1
1	Encounter	inside	0 / - / 1	none	1 / 1.5 / 2	1 / 1 / 3
2	Treasure	inside	0 / 0.5 / 1	none	0.5 / 0.5 / 2	0 / 1 / 3
3	Route	inside	0 / - / 1	union(0.5)	1 / 1 / 1	0 / 1 / 3
4	Exit	surface		none	0.5 / 1 / 1	0 / 0 / 0

An optimum parameter with a - value indicates that the elements follows a white noise distribution among the minimum and the maximum value. Surface POIs do not include information of their depth.

Table II
EXAMPLE OF POIS RELATIONSHIPS (ENCOUNTER(2) WITH OTHERS)

POIs Prev./Next	Crossable	Affinity	Galleries min/opt/max	Sinuosity min/opt/max	Distance min/opt/max
1 / 0	-	0	-	-	-
1 / 1	true	0.01	1 / 1 / 3	0.1 / 0.5 / 1	0 / 0.5 / 1
1 / 2	true	1	1 / 1 / 1	0 / 0.5 / 0.5	0 / 0.1 / 0.2
1 / 3	true	0.5	1 / 1 / 3	0 / 0.5 / 1	0 / 0.5 / 1
1 / 4	true	0.1	1 / 1 / 3	0 / 0.5 / 1	0 / 0.5 / 1

Incompatible relations (affinity = 0) ignore crossable, galleries, sinuosity and distance parameters.

- **Location:** The desirable position of the POI. It can be *inside* the terrain or at its *surface*. Figure 4(b) shows this classification of the cells of a Voronoi diagram.
- **Depth:** The depth at which the POI is going to be constructed. It is formed by three values (minimum, maximum and optimum depth) relative to the terrain depth. Only used if the location of the POI is defined as *inside*.
- **Special parameters:** List of parameters of the POI type which establish some special features for the point. Those parameters have associated a sub-parameter.

  start(N) Indicates the type for the first POIs in the system (e.g. the entrance of the cave). The sub-parameter N is the probability of a cave system to start with this POI.

  union(N) This parameter indicates that the POI is able to be reached from more than one POI (e.g. a crossroad). The sub-parameter N is the probability of the POI to be reached from more than one POI.
- **Cavity size:** The size of the underground cavity in which the POI is going to be constructed. It is formed by three relative values (minimum, maximum and optimum depth).
- **Branches:** The number of new POIs which are going to be constructed connected to the POIs of this type, generating forks in the cave. It is formed by three values (minimum, maximum and optimum depth).

The table I presents an example of a set of descriptors of the POIs used by our approach.

Second, the descriptors for the POIs relationships are provided. Those parameters must be established for every ordered pair of POIs previously defined.

- **Previous POI:** The ID of the first POI of the relation.
- **Next POI:** The ID of the last POI of the relation.
- **Crossable:** Boolean value which indicates if the galleries between the two POIs of the relation are crossable by other galleries. This should be false when the relation between the POIs should not be interfered by other elements of the cave (e. g. a big treasure before the boss monster of the dungeon).
- **Affinity:** The affinity of the relation. It is provided as a normalized value. While value 1 indicates the maximum affinity, value 0 indicates incompatibility between the two POI types.
- **Number of galleries:** The number of galleries which connect the two POIs. It is formed by three values (minimum, maximum and optimum number of galleries).
- **Sinuosity:** The relative sinuosity of the galleries connecting the two POIs. It is formed by three relative values (minimum, maximum and optimum sinuosity).
- **Distance:** The relative distance from the first POI to the second POI of the relation. It is formed by three relative values (minimum, maximum and optimum distance).

The table II presents an example of a set of parameters which defines the relationships of the POIs defined in the table I. Due to the extension of this data, the table shows only the relationship of the *Encounter* POI type.

Finally, global parameters are provided. Those parameters are:

- **Terrain depth:** The depth of the terrain. Usually, this parameter can be obtained from the volumetric terrain.
- **Global sinuosity:** An absolute value which establishes the sinuosity of the galleries of the cave system. This parameter modifies the relative sinuosity values of the relationships of the POIs.
- **Global distance between POIs:** An absolute value

which establishes the distance between POIs of the cave system. This parameter modifies the relative distance values of the relationships of the POIs.

- **Terrain scale:** The scale of the terrain. It is determined by a value representing the length of each voxel side in *points*. A *point* is an abstract unit of length (the designer decide if they represent meters, inches or other type of unit).
- **Underground cavity global size:** A parameter which establishes the size of the underground cavities. It determines the average number of cubical *points* (a unit established by the scale parameter) per underground cavity.

B. Algorithm overview

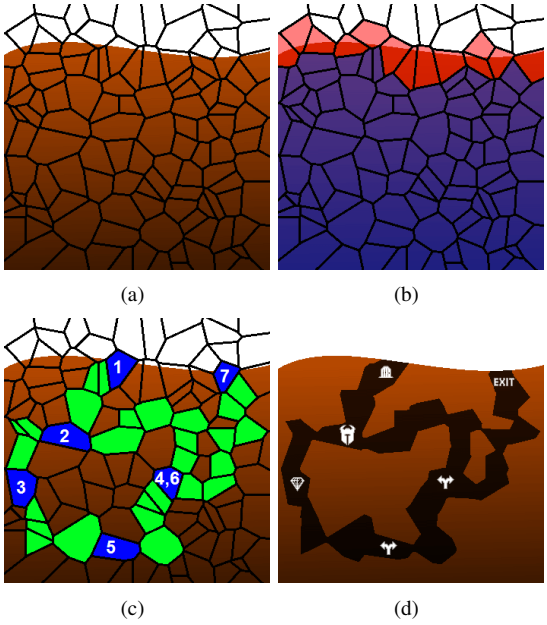


Figure 4. Cave generation process. This cave has been generated with the input data provided on the table I. (a) shows a Voronoi diagram generated over a two dimensional terrain. (b) shows the classification of the cells: *surface* (in red) and *inside* (in blue). (c) shows the cells selected by the process to generate the cave. It is formed by 7 POIs (the numbers in the POIs indicate the generation order), being the 4th and the 6th in the same cell due to the *union* parameter. (d) shows the final cave hollowed on the terrain with the POI types assigned.

The algorithm to generate the caves is formed by the following steps:

Step 1: Voronoi diagram & Delaunay triangulation:
The first step in creating the cave system is the generation of the Voronoi diagram and the Delaunay triangulation.

This process starts with the location of the seeds for the Voronoi diagram. These seeds are random points following a white noise distribution (so, they are uniformly distributed over the terrain). The number of seeds is calculated using the equation 1. N is the number of voxels of the terrain plot in which the cave is going to be generated, C is the underground cavity global size parameter and S is the terrain scale.

$$Seeds = \frac{N}{C} S^3 \tag{1}$$

Next, the algorithm use these points to construct the Voronoi diagram and the Delaunay triangulation. Figure 4(a) shows a two dimensional terrain with a Voronoi diagram generated over it.

Step 2: First point generation:
The next step is the generation of the first POI for the cave.

First, the POI type for the first point of the cave is chosen. This selection is randomly performed using the probability given in the *start* parameter of each POI type with this parameter defined.

Second, the algorithm selects the cell of the Voronoi diagram with the best characteristics to locate a POI of the previously selected type. This process is explained in more detail in section V-D.

Finally, the point is stored in a list of unfinished points. This list contains all the points from which the algorithm is going to construct the rest of the cave.

Step 3: Cave generation:
This step of the process is the generation of the cave itself. It is iteratively performed, taking the first point of the unfinished points list, removing it from the list and constructing the next POIs and the galleries between them.

Once the first point is picked from the unfinished points list, the number of branches is obtained. This value is randomly selected using the values given in the input data for this type of POI. The number of branches determines the number of new POIs to be constructed from this point. If this value is 0, no branch is constructed from it, and the next point of the unfinished points list is taken, repeating this step.

Next, if the branch number is higher than 0, the method constructs a new POI per branch.

For each branch, it selects the POI type for the next POI in the cave. This process is explained in depth in section V-C. Then, the new point is located in the terrain (explained in the section V-D) and the galleries between them are constructed. The number of galleries between the two points is randomly obtained using the values given by the number of galleries parameter. defined in the relation between the two types of POI. The generation process of each gallery is explained in the section V-E.

Finally, when all branches are constructed, the new points are added to the unfinished points list.

This process ends when the unfinished points list is empty.

Figure 4(c) shows an example of the result of this step.

Step 4: Hollow the cave:
The final step of the method is the physical generation of the cave in the terrain. The difficulty of this step depends on the terrain representation system. In this research we use

the volumetric terrain representation method proposed in our previous research [15], [16].

We can observe an example of this process in the figure 2. Figure 2(a) shows a general view of the terrain plot and the cave and figure 2(b) shows its entrance on the terrain surface.

Figure 4(d) shows an example cave generated on a two dimensional terrain.

C. POI type selection

The goal of this process is the selection of the next type of POI to construct in the cave. This selection depends on the last type of POI generated.

This process normalizes the affinities of the last POI type in the cave with the rest of POI types. Then, the next type is randomly selected using these normalized affinities. If the affinity between the last type of POI with all the POI types is 0, the method converts this point of the cave in a dead end.

D. POI location

This process is responsible for locating the next POI type in one cell of the Voronoi diagram. Additionally, if the point to locate is not the first POI of the cave, it needs the data of the last generated POI.

The steps to perform this process are different if the type of the POI to locate has the *union* parameter defined.

On the one hand, if the *union* parameter is not defined, the process has the following steps:

1. **Candidate selection:** First, the algorithm selects the cells of the Voronoi diagram at the location defined in the type of the POI to locate (*inside* the terrain or in its *surface*. Figure 4(b) shows a terrain with the *inside* cells in blue and the *surface* cells in red). Additionally, already assigned cells, uncrossable cells (galleries with the uncrossable flag active in the input data) and their walls (their neighbors in the diagram) are removed from this list of candidates.
2. **Calculation of the desirable features:** Next, the algorithm calculates the desirable features of the underground cavity to locate the POI. Those features are the depth of the cavity, the size of the cavity and the distance from the previous POI (only if the point is not the first POI of the cave).
   The values for those desirable features are calculated multiplying the relative by the absolute values. On the one hand, the process takes the relative values from the descriptors of the POI types (the depth and the size of the POI) and the descriptors of their relations (the distance between the two types of POI). On the other hand, the absolute values of those parameters are taken from the global parameters (terrain depth, underground cavity global size and global distance between POIs). All the desirable features are formed by three values: minimum, optimum and maximum.
3. **Candidate removal:** Then, the cells of the Voronoi diagram out of the ranges defined by the desirable features are removed from the candidate list. These ranges are delimited by the minimum and the maximum values of the desirable features. If this step removes all the candidates, the method considers that this branch can not be constructed, discarding it from the previous POI in the cave.
4. **Best candidate selection:** Finally, the method selects the best candidate of the list. This process compares the features of each candidate against the optimum value of desirable features. If there is not an optimum value defined, a random candidate is obtained. The selected candidate is the cell where the next POI is going to be located.

On the other hand, if the *union* parameter is defined, the process is different. The union parameter indicates that the POI is able to be reached from more than one POI. To get this type of construction, those POIs are located in the same cell of other POIs of the same type, obtaining several entrances to the POI from different POIs. This process has the following steps:

1. **Union probability:** First, a random value is obtained. If this value is higher than the probability of the *union* parameter, the process stops and locates the POI with the steps for the POIs without the *union* parameter defined. If the value is the same or lower than the probability, the generation process continues.
2. **Candidate selection:** Second, the method selects all the cells of the Voronoi diagram with the same type of POI as the type of the POI to locate, creating a candidate cell collection. If there is no cell with this type of POI assigned, the process stops and locates the POI with the steps for the POIs without the *union* parameter defined.
3. **Calculation of the desirable features:** Third, the desirable features are calculated. In this case, only the distance from the last point of the cave is required.
   This value is calculated multiplying the relative value given in the POI descriptions (the distance of the relation between the two POI types) by the absolute value provided in the global parameters (the global distance between POIs).
4. **Candidate removal and best candidate selection:** The last two steps of the process are the same as in the POIs with no union parameter defined locating procedure.

E. Gallery generation between two POIs

This process generates an underground gallery between two POIs already located in the terrain.

First, the method collects the impassable cells of the Voronoi diagram. Those cells are:

- The set of cells of the Voronoi diagram with a POI assigned.
- If the gallery which is going to be generated is defined as uncrossable in the input parameters, the set of cells of the Voronoi diagram of the already generated galleries.
- If the gallery which is going to be generated is not defined as uncrossable in the input parameters, only the set of cells of the Voronoi diagram of the galleries created with a relation defined as uncrossable.
- All the neighbor cells in the Voronoi diagram of the cells selected in the previous three sets (that is, all the walls of the previously selected impassable cells).

Next, the sinuosity of the gallery is calculated. To this extent, the relative sinuosity given in the description of the relation between the POI types is multiplied by the global sinuosity provided in the global parameters.

Finally, the method generates the underground gallery with a pathfinding algorithm over the graph obtained by the Delaunay triangulation. We use the A* search algorithm, which is a method extensively used in computer games [17]. Additionally, the sinuosity of the path is generated adding an error to the movement cost of each node of the A* algorithm. This error is randomly obtained between two values: $-sinuosity/2$ and $sinuosity/2$.

The process ends when the algorithm reaches the final node or when it has searched all the nodes and it can not reach the final objective. In the first case, the gallery is used in the cave, but in the second case the branch is discarded, dismissing the second POI of the relation and the gallery.

F. Generating several caves in the same terrain plot

When several cave systems are going to be constructed in the same terrain plot, the execution of the steps is different.

The first step of the process must be performed only once, regardless of the number of caves which are going to be constructed in the terrain. On the other hand, the second, the third and the four steps must be performed one time per cave.

Table III
MEASURED CONFIGURATIONS

Config.	POI types	Branches of each POI	Galleries between each pair of POI
A(simple)	4	1	1
A(complex)	4	1-2	1-2
B(simple)	8	1	1
B(complex)	8	1-2	1-2

VI. RESULTS

We measure the time of the cave generation using four different configurations. They have only one start POI type and one end POI type (that is, with 0 branches), all their POIs have the same affinity between them and all the galleries are crossable with a sinuosity and length of $(0.5 - 1.5)$. Table III shows the differences between the four configurations. The global cavity size is 100, the scale is 0.5, the global sinuosity is 20 and the global distance between POI is 20. They are constructed in a terrain formed by 40x50x40 voxels (the terrain size only affects the time taken by the algorithm to construct the Voronoi diagram).

Table IV
AVERAGE GENERATED ELEMENTS

Config.	Number of POIs	Number of gallery cells	Number of cells
A(simple)	3,53	10,67	100
A(complex)	5,56	20,45	100
B(simple)	4,53	15,16	100
B(complex)	7,41	29	100

We have tested our method on an Intel(R) Core(TM) i7 950 CPU, running at 3.07GHz.

We generate 200 caves for each configuration taking the average values for each measured aspect. Table IV shows the average number of POIs, the average number of galleries between POIs and the average number of seeds obtained per configuration.

Table V shows the time results of the process. We can observe our method needs more than 10 seconds to generate the cave regardless of the configuration. Figure 5 shows these results visually.

Table V
EXECUTION TIME

Config.	Average time (ms)	Standard deviation (ms)	Absolute deviation (ms)
A(simple)	10,742.72	209.20	165.87
A(complex)	11,213.37	960.32	491.05
B(simple)	10,803.00	241.58	190.16
B(complex)	11,584.60	1118.79	686.36

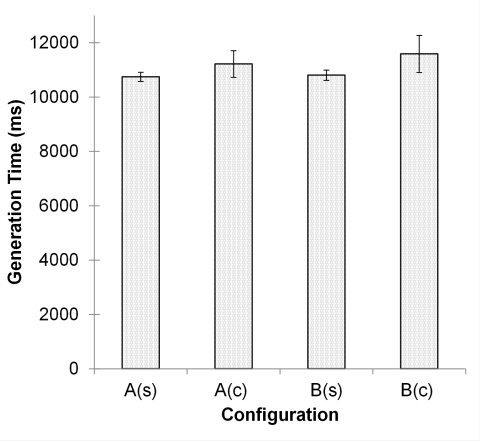


Figure 5. Execution time

Additionally, we measure the time needed by our method to perform different steps. First, we measure the time required by the Voronoi diagram and Delaunay triangulation generation step (see table VI). Second, we measure the cave generation process (see table VII). We can observe those results graphically in the figures 6 and 7.

Table VI
DIAGRAMS CONSTRUCTION TIME

Config.	Average time (ms)	Standard deviation (ms)	Absolute deviation (ms)
A(simple)	10,690.72	184.93	146.45
A(complex)	10,791.29	212.42	162.60
B(simple)	10,700.74	208.97	161.85
B(complex)	10,707.71	223.25	173.44

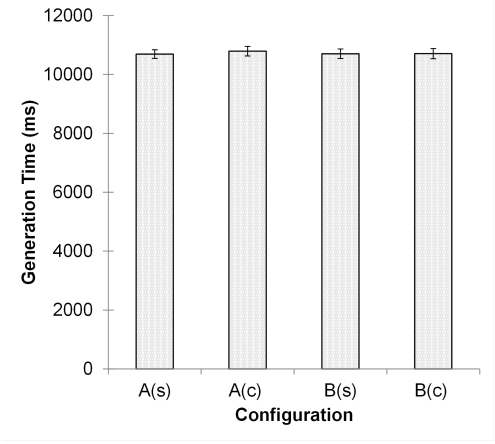


Figure 6. Diagram construction time

Table VII
CAVE GENERATION TIME

Config.	Average time (ms)	Standard deviation (ms)	Absolute deviation (ms)
A(simple)	52	85.54	69.67
A(complex)	422.09	961.51	470.21
B(simple)	102.26	120.52	106.72
B(complex)	876.89	1118.95	682.45

VII. DISCUSSION

There are several interesting aspects to discuss about the proposed method.

First, we can observe in the section VI that the main factor which affects the execution time is the growing capability of the cave. This aspect is determined by the number of branches of each POI type, the affinity of each POI type with the POI types with few branches, and the number of galleries between the POI types.

Second, table VII shows that the random nature of the proposed method causes a high variability in the time needed to perform the cave generation process. Additionally, this

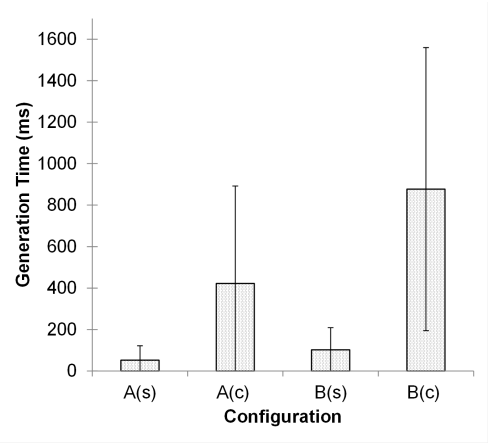


Figure 7. Cave generation time

table shows that if the input data allows more branches per POI, both time and variability increase significantly. This fact has three main consequences:

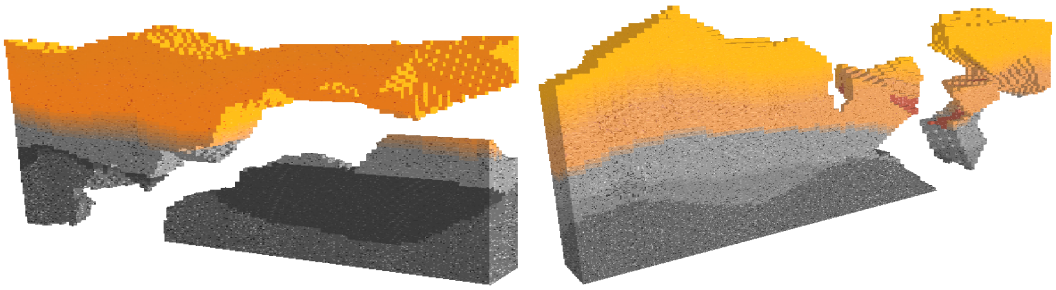
- It is very difficult to predict the time needed by the generator to create the cave.
- It is very difficult to predict the size of the resulting cave.
- The method is highly innovative, generating unexpected elements and structures.

Although the third consequence is a good aspect for procedural generators [15], the first and the second could be terrible in some situations. It is possible to palliate those effects with a carefully design of the input data, paying special attention to the number of branches of each POI and their affinity with other POIs.

Third, table V shows that the generator needs near to 11 seconds to obtain the result. Usually, this time is too high to generate the results in real time. In spite of that, we can observe that the diagrams generation is the main time consuming step, needing only less than 1 second for the generation of the cave itself (table VII). As we see in the section V-F, the method can be executed performing the Voronoi diagram and the Delaunay triangulation generation step only the first time, generating more than one cave only with the cave generation steps. So, the diagram generation could be performed in a step previous to the execution (for example, using a loading screen) and the cave can be generated in real time.

VIII. CONCLUSION

In this paper we propose a novel method to procedurally generate cave systems basing the process in a playability defined by the game designers. Additionally, we measure the time required by the process to perform the generation, analyzing the results and discussing their advantages and disadvantages.



(a) (b)

Figure 8. Slices of terrains with caves generated with our approach.

As future lines of work, we propose the acceleration of the diagrams generation step, adapting the complete process to the real time generation paradigm. Additionally, we propose the increase of the realism of the results, adding elements such as stalactites and stalagmites.

ACKNOWLEDGMENT

The authors would like to thank Mikel Salazar and Iván García for their support over the research process.

REFERENCES

[1] R. M. Smelik, K. J. De Kraker, T. Tutenel, R. Bidarra, and S. A. Groenewegen, "A survey of procedural methods for terrain modelling," in *Proceedings of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)*. Citeseer, 2009.

[2] M. Boggus and R. Crawfis, "Procedural creation of 3d solution cave models," in *Proceedings of the 20th IASTED International Conference on Modelling and Simulation*, 2009, pp. 180–186.

[3] ——, "Explicit generation of 3d models of solution caves for virtual environments," in *Proceedings of the 2009 International Conference on Computer Graphics and Virtual Reality*, 2009, pp. 85–90.

[4] ——, "Prismfields: a framework for interactive modeling of three dimensional caves," in *Advances in Visual Computing*. Springer, 2010, pp. 213–221.

[5] A. Peytavie, E. Galin, J. Grosjean, and S. Merillou, "Arches: a framework for modeling complex terrains," in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 457–467.

[6] L. Johnson, G. N. Yannakakis, and J. Togelius, "Cellular automata for real-time generation of infinite cave levels," in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. ACM, 2010, p. 10.

[7] J. Cui, Y.-W. Chow, and M. Zhang, "A voxel-based octree construction approach for procedural cave generation," *International Journal of Computer Science and Network Security*, vol. 11, no. 6, pp. 160–168, 2011.

[8] ——, "Procedural generation of 3d cave models with stalactites and stalagmites," *IJCSNS*, vol. 11, no. 8, p. 94, 2011.

[9] D. M. Tortelli and M. Walter, "Modeling and rendering the growth of speleothems in realtime," in *Proc. International Conference on Computer Graphics Theory and Applications*, 2009, pp. 27–35.

[10] G. Voronoï, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques." *Journal für die Reine und Angewandte Mathematik*, vol. 134, pp. 198–287, 1908.

[11] B. Delaunay, "Sur la sphere vide," *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, vol. 7, no. 793-800, pp. 1–2, 1934.

[12] F. Aurenhammer, "Voronoi diagrams—a survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 345–405, 1991.

[13] J. Olsen, "Realtime procedural terrain generation," University of Southern Denmark, Tech. Rep., 2004.

[14] M. Bern and D. Eppstein, "Mesh generation and optimal triangulation," *Computing in Euclidean geometry*, vol. 1, pp. 23–90, 1992.

[15] A. Santamaría-Ibirika, X. Cantero, M. Salazar, J. Devesa, I. Santos, S. Huerta, and P. G. Bringas, "Procedural approach to volumetric terrain generation," *The Visual Computer*, pp. 1–11, 2013.

[16] A. Santamaría-Ibirika, X. Cantero, M. Salazar, J. Devesa, and P. G. Bringas, "Volumetric virtual worlds with layered terrain generation," in *2013 International Conference on Cyberworlds (CW)*. IEEE, 2013, pp. 20–27.

[17] A. Nareyek, "Ai in computer games," *ACM Queue*, vol. 1, no. 10, p. 58, 2004.