Procedural Generation of Favela Layouts on Arbitrary Terrains

Natan Luiz Paetzhold Berwaldt
Laboratório de Computação Aplicada
Universidade Federal de Santa Maria
Santa Maria, Brazil
nlberwaldt@inf.ufsm.br

Rafael Vales Bettker
Laboratório de Computação Aplicada
Universidade Federal de Santa Maria
Santa Maria, Brazil
rvales@inf.ufsm.br

Cesar Tadeu Pozzer
Laboratório de Computação Aplicada
Universidade Federal de Santa Maria
Santa Maria, Brazil
pozzer@inf.ufsm.br

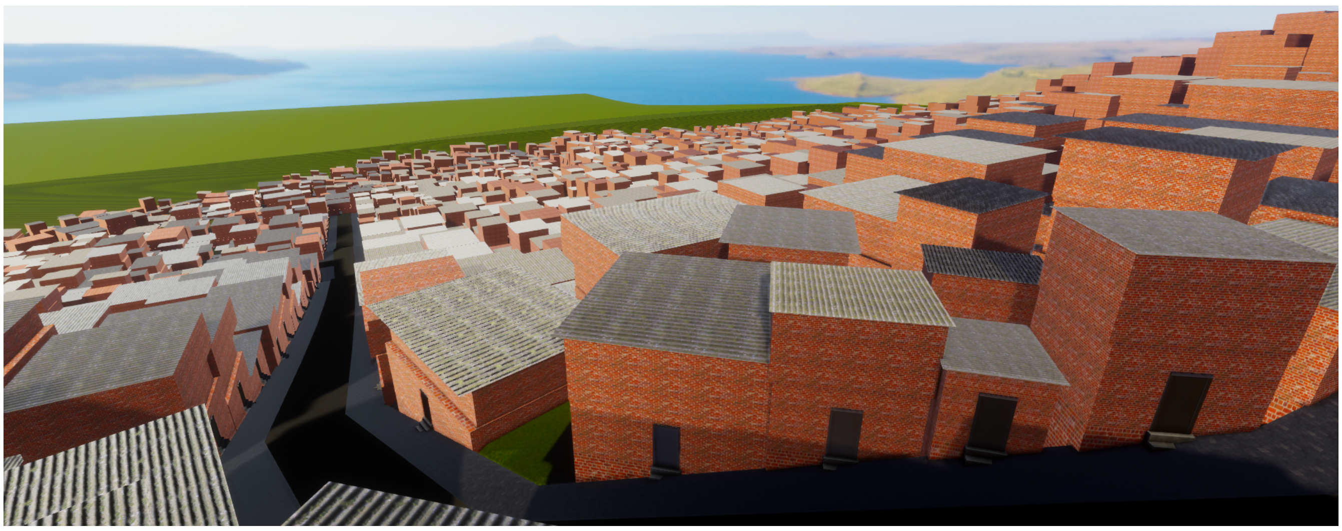
Fig. 1. A favela generated with our approach.

***Abstract*—The procedural generation of content for virtual worlds is a technique widely explored by the game industry since it allows a reduction in the development costs and time. Procedural city generation is a very common topic, given the growth in the size of scenarios in games and simulations. However, traditional city-generation techniques do not represent some particular variations found in urban spaces, such as informal urban settlements. This paper proposes a method for generating procedural favelas, containing its road system, division of building lots, and alleys. The road system uses the A* algorithm and is based on the heightmap, having the characteristics of favelas on uneven terrains. The method allows zigzag roads to adapt to the steep elevations, a higher number of dead ends, and large blocks, fitting several buildings, and an alley system. The generation of building lots and alley system is based on the quadtree algorithm. Each block is subdivided to fit its building lots. The buildings are positioned on their lot, leaving a free space for the alleys. The alley system connects all the buildings that are inside the city block to the road. Our purpose is to generate the layout of a favela on arbitrary terrain procedurally. The results can be used in games, animations, and different types of simulations involving this scenario, such as traffic analysis, natural disasters, and future urban growth.**

***Index Terms*—procedural city, virtual terrains, favela, informal settlements**

I. Introduction

Procedural city generation is a theme explored in various papers [1] and also with more specific features, such as road networks [2] and land use [3]. But these works tend to make generic cities, not representing particular varieties of urban formation.

The Brazilian favelas are one of the places that present a very peculiar way of land use, since these places are densely populated, and the terrain is improper for building, the constructions are made with an architecture that does not follow the building rules seen in other places, generating such different structures.

These places differ from other urban areas in many aspects. The roads are sparse and only covers a small part of the area. Unlike typical neighborhood where almost all houses have the entrance facing a road, in favelas, the dwellers generally need to cross through lots of narrow alleys in the way from the road to their houses.

This makes a new strategy necessary for the building distribution and alley generation not based exclusively on the roads, but still need to be influenced by them to generate the correct building alignment.

Another difference is the vertical expansion of the constructions. Due to the constant population growth combined with the difficulties to expand to other parts of the terrain because of the irregularities, the expansion is obligated to be vertical, filling all the available space without following standards or official building rules.

Most of the time, limited resources do not allow the construction of large buildings at once. So, the upper floors tend to be built at different times, making them have different styles, formats, and materials. In a procedural generation, this must be considered, which forces the generation of each one to be individual, not be possible to generate a whole regular set of identical floors like can be seen in standard buildings.

Since the buildings grow on top of improper and difficult-to-reach locations, such as hills, there is often no space to build many roads with various paths, or corners with several intersections, resulting in streets that are often dead ends, and blocks that tend to be larger than city standards, with several dozen buildings between roads.

Also, the roads adapt to the terrain on which the favela is located, seeking the smoothest path. In cases of a steep climb to reach over hills, an adaptation is necessary to respect the limit angle allowed for vehicles to travel.

In this work we propose a procedural method for road generation, building lots distribution, basic structure creation, and alleys connecting them. It is focused on the favela layout generation, not focusing on building modeling or realism of facades.

Games have a special need for these kinds of procedural generation since their worlds are getting larger and more complex. But it does not mean that all games need scenarios with realistic details; racing games generally only need to focus on the scenery around the road where the competitors are going to cross, and the racing game Descenders has a scenario representing a shanty town, by focusing only on the overall structures, with almost no detail.

The Microsoft Flight Simulator focuses on the simulation of flying planes, which makes it unnecessary for most parts of the map to have good quality in areas not visible from the sky. This game uses real world data to generate its structures to replicate the planet, but for not well-known places, such as favelas, it could be used a procedural approach, requiring much less effort from the development team.

This solution is also applicable for animations and different kinds of simulations on this type of settlements, like traffic analysis, natural disasters, and future urban growth. Fig. 1 shows a favela generated with our method.

The paper is organized as follows: Section II presents related works involving favelas and procedural generation of cities. An overview of the proposed method is presented in Section III. Section IV details the road network, and in Section V the buildings. Finally, Section VI and VII present and discuss the results.

II. Related Work

There are several procedural modeling techniques for generating cities [1], including roads, building lots, exteriors, and furnished interiors. The methods use different aspects to create virtual scenarios close to reality, such as the population, environment, vegetation, architecture, elevation, geology, and culture [4].

A method for generating cities from a single image is presented in CityCraft [5]. Online street images are used to train neural networks, capable of guessing the town from the input photograph. Then, the city's height map is obtained, and, together with the city properties, the generation is made.

In [6], Mizdal et al. describe an approach of procedural generation of villages based on the characteristics of the land to construct the road system and distribute the buildings. From an analysis of the terrain, more planar areas are defined where the village structures will be fitted. Then, using the A* algorithm, paths are generated connecting these villages.

Nishida et al. [7] present an interactive tool to create realistic roads. It has a sketching step to delimit the area and choose an example model, and then a graph is built from the characteristics of the example, forming the road network. This approach is based on examples, and unless there is a large database, the algorithm may not be able to behave correctly defining solutions for climbing steep slopes, since it was originally designed without this objective. Besides, for a mountain, the average height of the area is used to obtain a corresponding pattern. The result in these areas is the same pattern that is repeated frequently and is not likely to be used in favelas, since in them, the pattern differs in nearby locations, based directly on the variation of the local terrain.

To divide building lots inside city blocks, Vanegas et al. [8] propose a method that with a road graph, two styles of subdivisions can be made: the first uses polygon skeletons, and ensures that all front sides are on the street, while the rear-sides are adjacent to other lots; the other divide the lots in quadrilateral formats with or without access to the street.

Vanegas' strategy is very customizable and can be adapted to different kinds of urban layout, but dedicates much effort in the alignment of the blocks to ensure similarity with well-designed cities. Therefore we present a similar, but simpler solution considering that we do not focus on these detailed constraints like homogeneous area distribution, aspect-ratio, and easy access to the roads.

Several works focus on the generation of procedural cities [1] [2] [3] [4] [5], but they represent more generic urban areas, without the characteristics of the informal settlements, and [6] focus on the specific culture of villages.

In the scope of the favelas, Gadiraju et al. [9] propose a method for detecting informal settlements based on satellite images. For this, per-pixel based classification methods were used, which, combined with other features, identify the favelas' location and limits. The detection of informal settlements does not focus on procedural generation but the analysis of these locations.

Some works are based on informal settlements in South Africa. Glass et al. [10] extract, from aerial images, features from two urban complexes and use them as a basis, together with techniques such as Voronoi diagrams, subdivision, and L-system to generate procedural road patterns.

This technique does not use terrain height, only satellite images. The use of the Voronoi diagram restricts the blocks to patterns without straight roads, and without the possibility of creating dead ends outside the borders. This makes its use more specific for informal settlements like those in South Africa.

Rautenbach et al. [11] present a process to model 3D building for favelas. Starting from 2D building footprints data, the facades are constructed, adding details and styles, and producing the 3D model.

Also, Bade [12] presents a procedural environment tool with Houdini Engine for shanty towns. The method distributes materials on the facades according to several parameters. There is the random creation mode and the personalized one – which gives more control for the creation of the expected scenario –. This is not the subject addressed in our work, but it can be used as a complement to obtain a final model of a favela.

Finally, Buehler et al. [13] reproduce a favela within ArcGIS CityEngine, software for modeling and generating urban environments. The entire generation is done with the mechanisms available in the software, being limited to it. Besides, its facade modeling is analyzed and compared to other architectures by Beneš [14].

III. Overview

In this section, we present an overview of the proposed method. The process can be divided into three main steps (Fig. 2).

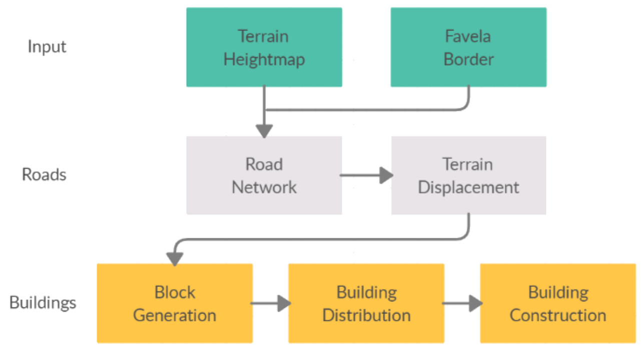

Fig. 2. Workflow of our approach.

First, the terrain heightmap and the favela boundaries are loaded. The boundaries of the favela correspond to a polygon – a list of 3D points – that will delimit the area where the favela will be built. Additionally, some parameters can be changed, such as the minimum size of blocks and buildings or the road's maximum slope.

The second step is the construction of the roads. The A* algorithm is used with a custom cost function, influenced by the height of the terrain (Section IV-A). The roads are generated using the method presented in Section IV-B, which is based on the typical characteristics of the favelas. During generation, each new road is added to a graph structure called a road graph.

After, the roads and sidewalks meshes are created. The terrain's height is compared to the height of the road to creating the displacement map, which is used to adjust the meshes to the terrain. Finally, the terrain heightmap is displaced. Details are described in Section IV-C.

The next step is the generation of buildings. The building blocks are explained in Section V-A. From the borders of the road mesh, polygons are generated. The polygons are used to limit and orient the building distribution, considering the road meshes previously created, and the area's limits passed as input.

Next, inside the polygon of each building block, the building lots will be distributed and aligned using the techniques described in Section V-B.

The last step generates a 3D representation of the building inside each building lot, translating it to the correct height over the terrain and creating the structure of each building floor.

IV. Road Network

The proposed road network generation method is based on the differential characteristics of the layout of a real favela, when compared to city standards previously described. The road network has a branching pattern in which new roads are created as the favela expands. The generation is according to the terrain, seeking to cover less steep areas to facilitate the transit of people and vehicles.

One approach that can be used to separate passable areas is navmeshes. From a maximum street slope limit, it is possible to define the areas that are likely to receive streets, then use pathfinding algorithms, such as A*, to find paths connecting two areas.

This technique is not effective for cases where the difference in height between two nearby points is large – but within the maximum slope limit –, resulting in uneven streets instead of crossing for a nearby flat area. So, to find for the most possible flat paths, Section IV-A presents a custom cost function for the A* algorithm on the grid generated from the map vertices.

To generate street patterns similar to the real favelas in arbitrary terrains, some specific characteristics are needed, such as frequent dead end streets, and access points in elevated areas. A street may not be able to climb a mountain, so adaptations must be made to provide access to the buildings at the top of a mountain. The approach used is shown in Section IV-B. Finally, Section IV-C shows the generation of street and sidewalk meshes.

A. *A* Algorithm*

The custom cost function is strongly influenced by the elevation of the terrain. For the calculation, the first step is to distribute n points in a straight line from the source to the target. Each point will have its partial cost and weight –

the closer to the source point, the higher the weight – for calculating the final cost. The value of n can be adapted according to the terrain's size and irregularity, using a larger number of points for more irregular terrain.

Each point has its partial cost function C_p (1) which is the sum of the distance between p_i (position of the i-th point from the source) and p_t (position of the target) on the x and z – axes parallel to the ground plane –, and the square of the distance of the height y, to have an exponential disadvantage for slope paths.

$$C_p(i) = |p_i.x - p_t.x| + |p_i.y - p_t.y|^2 + |p_i.z - p_t.z| \quad (1)$$

The final cost formula C_t (2) for getting from one point to another is given by the sum of each point's partial cost multiplied by its weight.

$$C_t = \sum_{i=1}^{n} \frac{C_p(i)}{i} \quad (2)$$

Thus, the algorithm advances from source to target, being more influenced by the nearest slopes, seeking to circumvent them before being influenced by the slopes from afar. When the target is reached, the points that form the path are transformed into edges to be added to the road graph, presented in the next section.

B. Road Graph

The representation of the roads is made with a planar graph, where the vertices correspond to corners or curves, and the edges to each path of the road. The first step is to create an initial road from the point of the border with the lowest y value – representing an entrance to the favela – to the opposite edge, crossing the entire area. From that road, branches will be made to create a road network.

With the start and end points of the road in hand, the A* algorithm searches for the most appropriate path connecting them. The path is returned as a set of 3D points, which are added as vertices in the road graph. Edges are also added, connecting each vertex of the path with its successor.

After adding the first road, each vertex of the graph is visited and has a given possibility of branching. The chance for the first road vertices should be higher than the others since it is the main road that crosses the entire favela. The percentage can vary depending on the desired complexity of the road network, a higher probability represents a favela with a higher density of streets. In general, the appropriate values are around 0.3 and 0.15 for, respectively, the main road and the others.

If a vertex is branched, it is checked which side the branch will be made, with equal chances for both. During the branching of a vertex, it is not possible to branch to both sides. Thus, a four-way intersection only occurs when a road is built crossing an existing one.

When a branch occurs, a line perpendicular to the vertex's road segment is calculated, extending to the border of the favela. A random point from the middle onwards of that line is selected to be the end of the new road, and then sent to the A* algorithm to generate its path. As new roads are created, their vertices are added to the queue to possibly branch. The process ends after all vertices are verified. Fig. 3 shows the steps of the generation.

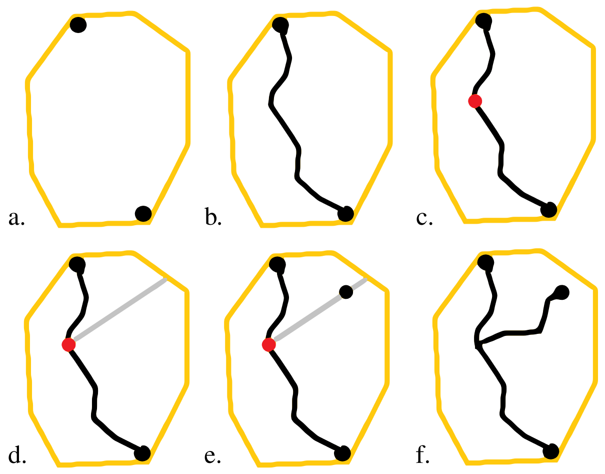

Fig. 3. (a) Points for the first road are defined from the border of the favela. (b) The path is created. (c) A vertex is chosen to branch. (d) A perpendicular line (grey) is created from the vertex. (e) The end point of the new road is defined. (f) The path is created.

When adding a new vertex, it is checked if there is already a vertex next to the location that can be reused. In this way, the vertex to be added is compared with all those already existing. If there is one at a distance less than α (value defined as a parameter), it is used as the vertex to connect the new edges; otherwise, the new vertex is added to the graph. This reduces the number of vertices in the graph and to avoid very short edges, which must be α as the minimum size.

When adding a new edge, Breadth-First Search is used to check whether adding it will result in a cycle with a circumference less than a given value of β, received as a parameter, preventing very small blocks from being created. If this occurs, the edges belonging to the cycle are checked, and the one with the largest size is removed. Additionally, if an edge crosses with an existing one, a vertex is created at the crossing point, ensuring the planar graph.

With each new road created, before being added to the graph, the slope of each edge is checked to ensure that the road is within a given maximum allowed slope. If the angle of the edge is greater than a θ angle, given as a parameter, it is removed along with any subsequent ones that also inflate the limit, until an edge below the maximum angle is found, or until the end of the road. The start and end points of the removed sequence are used to build a new path.

To generate the new path respecting the angle limit, several segments of varying sizes between α and 2α are created, rotated from $-75°$ to $75°$ in front of the starting point. Those segments with an angle higher than the limit, or that increase the distance to the end, are ignored. Among the others, the segment with the smallest angle of inclination must be chosen to be created, is the basis for creating the next, and so on.

If all segments are ignored at some stage, the generation is interrupted, and the final road becomes the path built until the interruption. If there is more road beside the segment removed, it is also removed to avoid roads without connections. When any segment reaches a distance less than α to the end point, it is connected, and the process is completed. Examples of road patterns obtained are seen in Fig. 4.

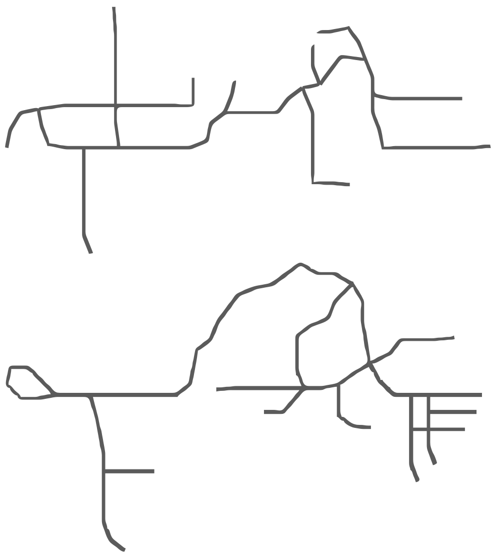
Fig. 4. Examples of road patterns with our approach. The first half (left to right) of each pattern represents a more planar area, while the other half represents a steep area.

C. Mesh

After the road graph is generated, it is necessary to create the road and sidewalk meshes and adapt the terrain to match them. The meshes are created from the graph, where each vertex is used as a base to create the triangles of the road mesh, and then the sidewalk mesh is built around it.

Since the roads are flat, displacement mapping must be made to adapt the terrain to the road. For that, all the triangles of the meshes are traversed and checked which vertices of the terrain are within those triangles. For optimization purposes, only the terrain vertices between the minimum and maximum points of each triangle's edges are traversed.

The verification is made from barycentric coordinates, which indicate whether each vertex of the terrain is within the mesh triangle. If so, the height difference between the y value of the triangle and the heightmap for the vertex position is calculated. If the vertex is not under the mesh, the displacement value is zero. The resulting values form the displacement map that is used to model the terrain when adding the meshes.

V. Building

A. Blocks

Each region (area bounded by edges) of the road graph represents a city block that must be divided into building lots. The buildings, in the real favelas, follow the orientation of the others that are nearby. This means that the structures in an inner layer of the block will face a nearby road. For our approach, each block is divided into several pieces, each influenced by the orientation of a nearby road.

For this stage, a new graph structure – called block graph – is created. Vertices have two additional attributes; one indicates whether it is near the road (inner) or away from the road (outer), and another to show to which vertex of the road graph it is related. Edges have an attribute to indicate whether it is connecting inner-inner, inner-outer, or outer-outer vertices.

Initially, each vertex of the road graph has its edges numbered according to its angle (Fig. 5a). Then, each pair of neighboring edges is used to create a vector with their average angle. There will be a vector for each edge, as shown in Fig. 5b.

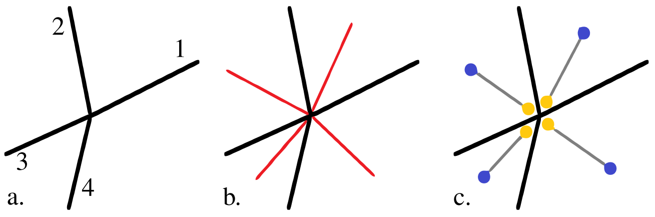

Fig. 5. (a) The road graph edges are numbered according to their angle. (b) Middle vectors (red) are created with the average angle of each corner. (c) The inner (yellow) and outer (blue) vertices are positioned, and an inner-outer edge (grey) connecting them.

The middle vector starts from the connection of two streets and divides the block into two parts. A building positioned to the left of the vector is closer to the left street, and vice versa. Thus, it is possible to divide the block into several parts, each oriented by a nearby street.

A polygon represents each part of the block. Every building within a polygon receives the same orientation, facing the nearest street. For this, the block graph is constructed, and then each region becomes a polygon.

Initially, the middle vectors are used to create the first vertices and edges of the block graph. An inner and an outer vertex are positioned over each middle vector. The inner vertex is added at a distance equal to the width of the road and sidewalk meshes, to start positioning the buildings after the sidewalk. The outer vertex is positioned at a distance α equal to the minimum size of a road segment, which prevents vertices from being placed outside its block. An edge connects the created vertices. Fig. 5c shows the vertices and the edge connecting each pair.

The next step creates edges to connect vertices, inner with inner and outer with outer. For each vertex of the block graph, edges are generated connecting them with all vertices of the same type (inner or outer) of the neighboring road vertex to which is related. Those edges that collide with any road are removed, while the others are added to the block graph, as shown in Fig. 6.

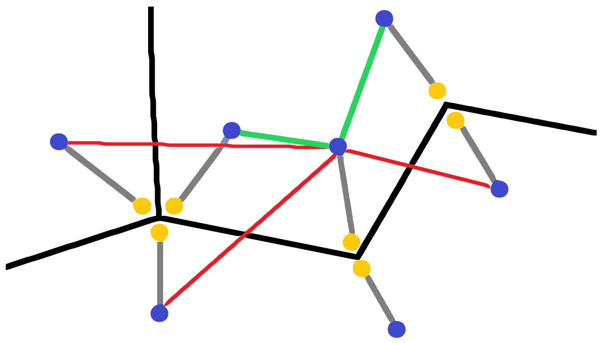
Fig. 6. The black line representing the road, with the inners (yellow) and outers (blue) vertices, positioned previously. The example shows the process of an outer vertex connecting with the outers of neighboring road segments. The red lines are unsuccessful connections, as they cannot collide with the road. The green lines are successful connections, which are added to the block graph.

Vertices of the road graph that have only one edge have a special treatment. Two middle vectors are created, displaced 135 and -135 degrees, from the edge (Fig. 7a). Then, the vertices on each middle vector are created, and the edge connecting them, as shown in Fig. 7b. These will be connected with the neighboring vertices, in the same way as previously presented, to create a polygon for the buildings at the end of the street, ensuring that they face the road.

Also, the end of the street can cause an unwanted connection of outer-outer edges – represented by the red line in Fig. 7c –, since the condition of their creation is based on the collision or not with a street. To avoid this, the road edge is extended with a value equal to half the distance of the outer vertex. This extension only prevents the crossing of outer-outer edges, therefore inner-inner are not affected.

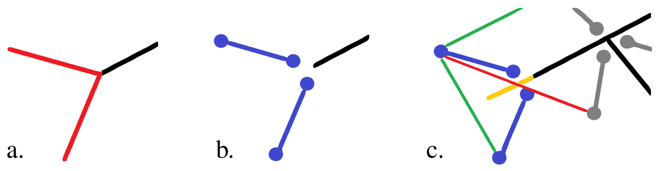

Fig. 7. (a) Middle vectors are created. (b) The vertices are created, and the edge connecting them. (c) From the upper outer vertex, edges with neighboring outer vertex are created. The red edge is not expected. A street segment (yellow) is added to ensure collision so that it is not considered.

If a new edge intersects with an existing one, a vertex of each must be moved to the point of intersection. If it is an inner-inner edge, the chosen vertex is the closest to the point of intersection. If it is an inner-outer edge, the inner vertex is selected. Outer-outer edges must not intersect.

The result of the block graph is shown in Fig. 8. The constructions between the contour of the graph and the favela border are randomly oriented.

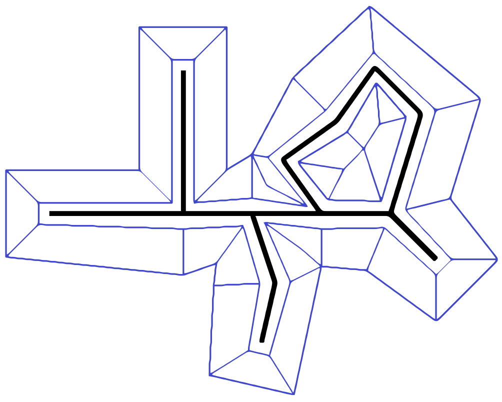
Fig. 8. Example of a block graph (blue) based on a simple road (black). Each region of the graph is transformed into a polygon, where the constructions will be built. The orientation of the buildings is the same as the edge of the polygon parallel to the street. The area outside the block graph is not influenced by any street, so assumes random orientation.

The final step generates the polygons that will be used for the distribution of building lots. Each inner-inner edge is used for creating a four-point polygon. Each of the two inner vertices of this edge is connected to exactly one outer vertex, so the two inners and two outers are used to form the polygon. The orientation of the buildings within that polygon is the same as the inner-inner edge.

If a block is large, the central area will be far from all roads, so its polygon will not be obtained with the method described above. To get around this, a search is done for cycles in the block graph. Cycles that have only outer-outer edges form an n-point polygon with a random orientation. Afterward, all the polygons obtained are sent to distribute the building lots.

B. *Distribution*

The algorithm for generating the building lots is a 2D process, ignoring the terrain and building height. For this, some parameters have to be previously set:

- Minimum and average building size;
- Minimum alley width;
- Maximum building floor height.

The first step generates a bounding rectangle from the polygon received from the division of the blocks. The first two points of the polygon determine its orientation. For most polygons, it is the line that matches with the road. Those who do not collide with the road – are within a block – are independent and therefore have a random orientation. See Fig. 9a.

The bounding rectangle is subdivided using an irregular Point Quadtree algorithm [15]. A point is randomly selected inside the area, with a distance from the borders of, at least, the average size of a building. This point then turns to be the vertex on which the rectangle will divide itself into four new nodes of the quadtree. The process is recursively applied until the point cannot be generated anymore without disrespecting the minimum distance to fit a building and alley.

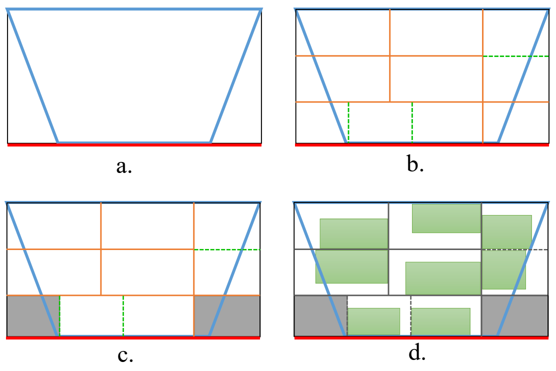

Fig. 9. (a) The polygon in blue and the bounding rectangle in black where the red line defines its orientation. (b) Quadtree subdivision (in orange) and node refinement (dashed green line). (c) The gray areas have the center point outside the polygon and are not used. (d) 2D building area constructed inside the lot limits.

In sequence, the nodes undergo a refinement process. If a node's size in an axis is much greater than the other – e.g., the x being 2.5 times larger than the z size – the area is subdivided into smaller equal areas respecting the minimum building size (Fig. 9b).

Each center point of these nodes is added to a hash table [16] that covers the whole map and undergoes a test to check if its center lies inside the polygon. If so, all its nodes are too. If the cell is partially inside, each node needs to have its position checked. Those that have their area inside the polygon are collected and used as building delimiter to the construction. See Fig. 9c.

To construct the base of the building, a rectangle is generated inside the area, with random dimensions limited by minimum building size and maximum percentage occupation. The percentage is obtained subtracting the area size by the minimum alley width. This building base is aligned to one of the area's corner to add more randomness to the distribution. The result is seen in Fig. 9d.

The unused space serves as an alley to access the buildings and needs to be large enough for people to transit, as set in the alley width parameter. Since the building is always aligned to a corner, and its size never reaches the whole space, the other three corners of the rectangle are vacant, which increases the possibility of an alley to connect with the adjacent ones without the need to adjust it manually.

Finally, the central points of the areas with buildings are updated in the hash table for the construction center. Then, it is compared with all buildings present in the same or adjacent cells of the hash. Constructions that collide with the ones created by another polygon are discarded.

C. Construction

Once generated the structure's base in 2D, the 3D buildings need to be constructed. A box is created based on the rectangle dimensions and a height set as a parameter for the first floor. This box is translated to its correct world position sampled from the terrain.

In the favelas, the buildings can have multiple floors, based on the number of family members, financial conditions, and space available. Since it is common to build a new story after a long time the house already exists, it is usual that each floor has different dimensions, materials, and styles.

Some favelas have buildings with a high number of floors, while in others, the buildings do not have more than 2 stories. In this way, parameters are used to define the maximum number of floors, and their distribution across buildings.

Each building height is randomly selected between one and the maximum number of floors, whose probabilities are set as parameters, as this work does not consider the characteristics of the dwellers. As the new floor is generated, it has a probability of having dimensions slightly different from the lower floor, but still respecting the max build area for that construction.

Since we do not make the displacement of the terrain to align with buildings, the first floor can be partially underground or floating over the terrain, so a foundation is created under the structure to keep the whole building over the map surface.

In the end, an entrance is made for each building, represented by a model of a door. This entrance is preferably facing the road if the construction is adjacent to the sidewalk, or an alley in other cases. An example of the resulting building can be seen in Fig. 10. The visual is not meant to be realistic or detailed, but to highlight the building distribution efficiently.

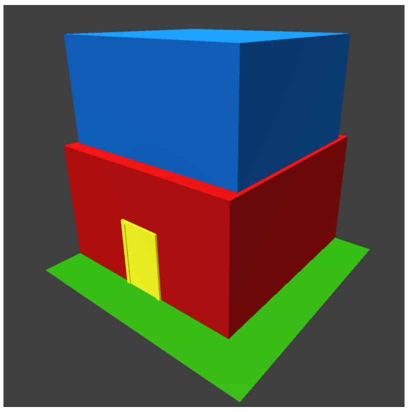

Fig. 10. Example of generated building. In green unoccupied space of the individual building area (also alley space), red the building first floor box, blue the second floor, and yellow the building entrance.

VI. Results

The presented method was implemented in C# language using Unity Engine, however, it is generic and can be implemented on any development platform. To evaluate the performance, the time for each stage was computed, which can be seen in Tables I and II. The experiments were performed on an Intel Core i5-7200U 2.50 GHz processor with 8 GB DDR4 RAM.

For the tests, 10 runs were made, and then an average of the times was calculated. The buildings were created with a

TABLE I
GENERATION TIME ON SMOOTH TERRAIN

Stage	Area size		
	25,000 m^2	50,000 m^2	100,000 m^2
Road Network	4.21s	5.54s	7.60s
Buildings	1.95s	4.88s	16.09s
Total	6.16s	10.42s	23.69s

TABLE II
GENERATION TIME ON STEEP TERRAIN

Stage	Area size		
	10,000 m^2	50,000 m^2	100,000 m^2
Road Network	5.10s	6.42s	9.24s
Buildings	2.18s	5.96s	21.20s
Total	7.28s	12.38s	30.44s

minimum size of 2.5 meters in all 3 axis and alleys with a minimum width of 0.5 meters. A scale of 1 pixel per square meter was used.

For smaller favelas, the road network time is longer than the building generation, while it is practically the same for a 50,000 m^2 favela. In larger favelas, the time to generate buildings grows faster than to generate roads.

The accelerated growth in the time of construction generation happens because as the construction area increases, the number of constructions also grows. The buildings generated are about 1,400, 2,500, and 4,200 for 10,000, 50,000 and 100,000 square meters, respectively. Besides, the building generation time is greatly affected by the number of road segments, which means more building blocks, causing the whole building generation process and collision detection to be executed more times.

In turn, the road network has the growth in time caused by the longer roads in the larger favelas, making it take longer to find the way. One option to decrease this time is to make the A* algorithm skip very close pixels since, in most cases, there is no relevant difference in height in neighboring pixels.

When comparing the generation time between favelas on smooth and steep terrains, steeps tends to take slightly longer. The time of the road network is affected because it searches for paths with the lowest number of elevations, thus having to pass through more points until reaching the destination. This search results in more irregular paths and with more curves, increasing the number of building lot polygons generated and, consequently, the generation time of the constructions.

To evaluate the visual aspect of the generation, favelas were created on different types of terrain. Fig. 11 shows a favela generated in a smooth terrain, with straight roads and more planar alleys. Fig. 1 and Fig. 12, on the other hand, show a generation on steep terrain, with zigzag streets to circumvent the elevation of the terrain, and buildings with foundations to adapt to the inclination.

Fig. 13 shows a view of the street with the buildings and the entrance to an alley, and a view from within.

The resulting scenarios still need lots of features to be a realistic representation of the favelas, but what was proposed here

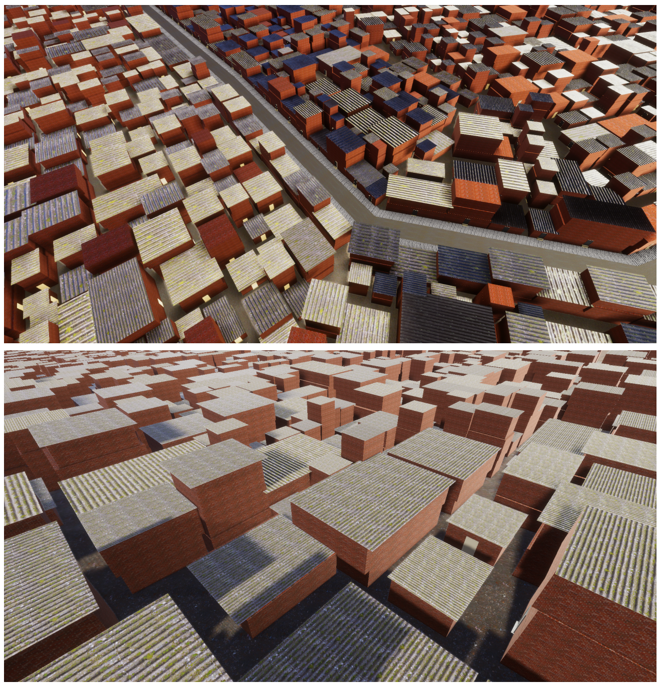
Fig. 11. Favela on smooth terrain.

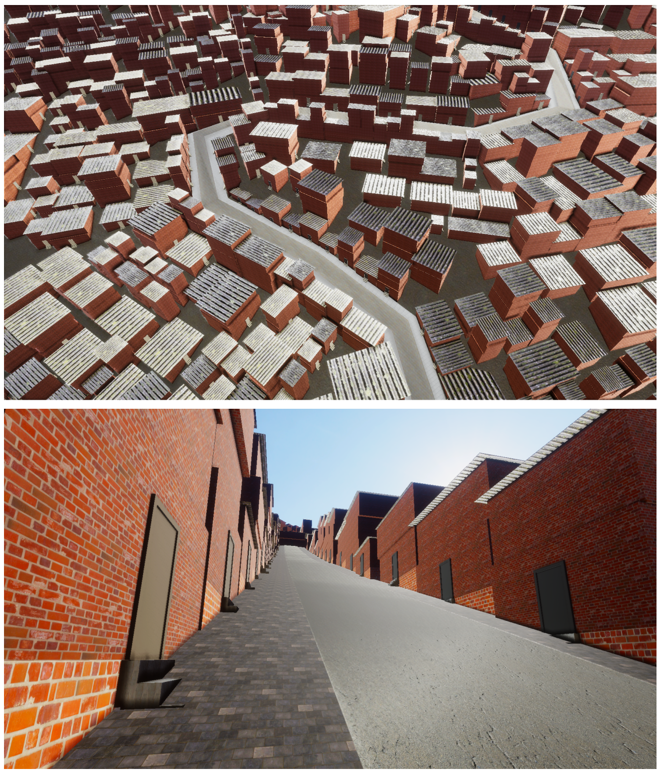
Fig. 12. Favela on steep terrain.

is not a faithful representation, which would need profound studies about sociology, geography, and architecture. Instead, we focus on a simple building and road distribution method

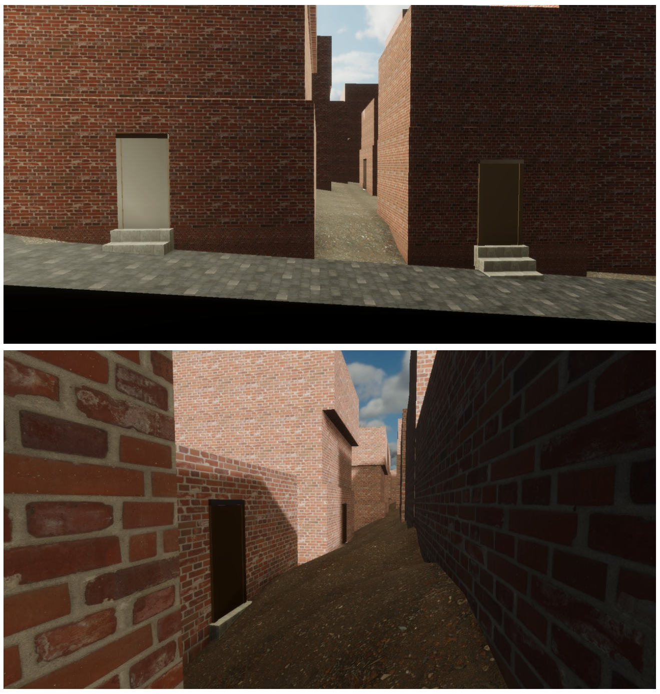

Fig. 13. The entrance and inside of an alley.

to be used in applications that do not demand this level of realism.

VII. Conclusion and Future Work

From an analysis of aerial images and videos recorded inside real favelas, a method was proposed with a focus on generating the layout of a favela procedurally on arbitrary terrain. A road network has been created, taking into account the characteristics that differentiate it from a standard city road network, such as huge blocks, adaptations to the terrain with the A* algorithm, and zigzag method to allow roads in steep areas.

The building blocks generated around the roads serve well as position limits and orientation for the displacement of building slots. The point quadtree algorithm proved to be a simple and efficient way to distribute the buildings without requiring lots of exception treatments like buildings generated over the streets or smaller than the minimum size.

The method is suitable to be used in games, animations, and different types of simulations, such as traffic analysis, urban growth, and natural disasters, such as landslides since it has mostly buildings in inappropriate locations.

Finally, the method is not concerned with making models with detailed textures and realistic scenarios. Our focus is to deliver the layout of a favela based on a heightmap, making the distribution of streets and buildings according to real favelas. The generated buildings floors have a simplified form of boxes that can be used as simple structures or as bounding boxes for more detailed construction shapes.

The next step of the work is to procedurally generate models of houses and stores, with particular textures from favelas, such as exposed brick and concrete walls, roofs, windows, and balconies. The interiors for buildings can also be explored to increase the immersion in the scenario.

Beyond that, the generation of exterior objects must be exploited to obtain a favela rich in detail. Generation of vegetation, streetlights with their wiring, irregularities in streets and sidewalks, and other properties of cities, must be considered.

The parallelism of the GPUs can be used to increase the algorithm's performance, making real-time generation possible. With this, the presented techniques can be used together with tools to provide the users with the possibility of making adjustments to the generated features, adapting the scene to the desired results.

References

[1] E. Cogo, I. Prazina, K. Hodžić, H. Haseljic, and S. Rizvić, "Survey of integrability of procedural modeling techniques for generating a complete city," in *2019 XXVII International Conference on Information, Communication and Automation Technologies (ICAT)*, 2019.

[2] J. Beneš, A. Wilkie, and J. Krivanek, "Procedural modelling of urban road networks," *Computer Graphics Forum*, vol. 33, no. 6, pp. 132–142, 2014.

[3] T. Lechner, B. Watson, P. Ren, U. Wilensky, S. Tisue, and M. Felsen, "Procedural modeling of land use in cities," pp. 1–11, 2006.

[4] D. Carli, F. Bevilacqua, C. T. Pozzer, and M. d'Ornellas, "A survey of procedural content generation techniques suitable to game development," in *Proceedings of SBGames 2011*, pp. 26–35, 2011.

[5] S. Kim, D. Kim, and S. Choi, "Citycraft: 3d virtual city creation from a single image," *The Visual Computer*, 2019.

[6] T. B. Mizdal and C. T. Pozzer, "Procedural content generation of villages and road system on arbitrary terrains," in *Proceedings of SBGames 2018*, pp. 556–562, 2018.

[7] G. Nishida, I. Garcia-Dorado, and D. Aliaga, "Example-driven procedural urban roads," *Computer Graphics Forum*, vol. 35, no. 6, pp. 1–13, 2015.

[8] C. Vanegas, T. Kelly, B. Weber, J. Halatsch, D. Aliaga, and P. Müller, "Procedural generation of parcels in urban modeling," *Computer Graphics Forum*, vol. 31, no. 2, pp. 681–690, 2012.

[9] K. Gadiraju, R. Vatsavai, N. Kaza, E. Wibbels, and A. Krishna, "Machine learning approaches for slum detection using very high resolution satellite images," in *2018 IEEE International Conference on data Mining Workshops (ICDMW)*, pp. 1397–1404, 2018.

[10] K. Glass, C. Morkel, and S. Bangay, "Duplicating road patterns in south african informal settlements using procedural techniques," in *Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa - Afrigaph '06*, vol. 2006, pp. 161–169, 2006.

[11] V. Rautenbach, Y. Bevis, S. Coetzee, and C. Combrinck, "Evaluating procedural modelling for 3d models of informal settlements in urban design activities," *South African Journal of Science*, vol. 111, no. 11/12, 2015.

[12] M. A. Bade, "Procedural environment for unity with houdini." https://80.lv/articles/procedural-environment-for-unity-with-houdini/, 2018. Accessed on: Aug. 12, 2020. [Online].

[13] M. Buehler, "Making of favela." https://www.ronenbekerman.com/making-favela/, 2014. Accessed on: Aug. 12, 2020. [Online].

[14] J. Beneš, T. Kelly, F. Děchtěrenko, J. Krivanek, and P. Müller, "On realism of architectural procedural models," *Computer Graphics Forum*, vol. 36, pp. 225–234, 2017.

[15] H. Samet, "The quadtree and related hierarchical data structures," *ACM Computing Surveys*, vol. 16, no. 2, p. 187–260, 1984.

[16] C. T. Pozzer, C. A. de Lara Pahins, and I. Heldal, "A hash table construction algorithm for spatial hashing based on linear memory," in *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology*, p. 35, 2014.