Semi-automatic Generation of Virtual Reality Environments for Electric Power Substations

Leandro Mattioli[1], Alexandre Cardoso[2], Edgard A. Lamounier Jr.[2], and Paulo do Prado[3]

[1] Federal Center of Technological Education of Minas Gerais (CEFET-MG), Araxá, Brazil
leandromattioli@araxa.cefetmg.br
[2] Federal University of Uberlândia (UFU), Uberlândia, Brazil
{alexandre,lamounier}@ufu.br
[3] Energy Company of Minas Gerais (CEMIG), Belo Horizonte, Brazil
prmprado@cemig.com.br

Abstract. The use of Virtual Reality environments in power substations offers a new paradigm for supervisory control. The existence of a tridimensional model, geometrically compatible with the real substation, minimizes the difference between the mental model constructed by field operators and the control room, improving communication. Additionally, such virtual environments may be used as simulators and training environments. Nevertheless, the development process related to these applications is quite complex, including activities as computer programming, 3D modeling and usability studies with significant managing tasks. This paper presents some new software layers to make the development of 3D virtual power substations easier. By providing tight integration between traditional CAD software and 3D engines, the tools presented here were successfully applied in the construction of several virtual electric power substations.

Keywords: Virtual-reality, supervisory systems, geometric modeling.

1 Introduction

Virtual Reality environments are valuable tools for system operations and training. Being a high representative human-machine interface, these systems reduce the difference between the operation model and the real environment, providing a singular experience for operators. Besides, VR environments allow remote training without significant loss of visual information. The use of such systems was already evaluated in several works [1-3]. However, as exposed in [4], the creation of these environments is commonly quite complex. Great effort has been made to improve the time required in the development of virtual environments, automating some of the tasks needed and inherent to the process. For instance, batch positioning 3D models based on fuzzy inputs is possible thanks to systems such as the one proposed in [5], which receives relative positioning information between objects (e.g.: NORTHEAST) and evaluates to relative coordinates (e.g.: $\Delta x = 20$; $\Delta y = 15$), in such fashion that the scene as a whole looks realistic. The WordsEye system, presented in [6], places

Á. Rocha et al. (eds.), *New Contributions in Information Systems and Technologies*,
Advances in Intelligent Systems and Computing 353, DOI: 10.1007/978-3-319-16486-1_83

models in the scenario by using semantic information extracted from a narrative. In [7], a scenario is described textually in terms of objects position constraints, which can be used to specify absolute coordinates, block collisions and group objects.

The exact equipment positioning is essential for the success of VR projects related to the training and operation of power substations. While photos, 3D scanners and floor plans, can help modeling, many researchers have worked in the generation of buildings virtual environments by means of processing architectural drawings, as exposed in [8-11]. In the particular case of power substations, some solutions were developed to extract topology information from one-line electrical diagrams and create representative environments [12]. However, these approaches ignore the equipment physical coordinates.

Also, new power substations projects, conceited with VR application in mind, may benefit from a tighter integration between CAD environments and 3D engines. For instance, devices inserted or moved inside the CAD application could result in the insertion or translation of 3D models in the virtual environment, automatically. Finally, a database application for managing such devices and their associated symbols and 3D models may be of great use when dealing with big projects.

This paper demonstrates some tools designed to extend AutoCAD© and Unity 3D© software packages. These tools were successfully applied in the creation of several virtual environments, related to Brazilian power substations, all managed by CEMIG electric company.

2 Geometric Modeling

This section presents some modeling conventions that are required in order to benefit from the automation tools. This task might be decomposed into two levels: individual equipment modeling and the scenario as a whole.

2.1 Equipment Modeling

When 3D models are not provided by the equipment vendor, they can be created using 3D scanners, photo sets, detailed views etc. The use of *ad hoc* devices may be expensive and result in meshes with too much polygons, affecting the performance of scene rendering.

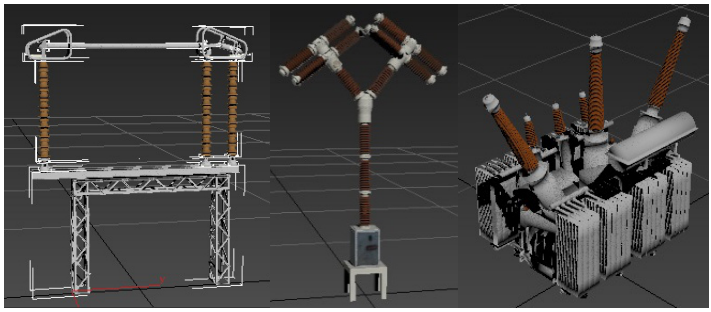

Fig. 1. Virtual models for a disconnector, a circuit breaker and a transformer

The creation of high optimized models is beyond the scope of this work. However, it is required to establish some conventions, notably the compatibility of reference (pivot) points between the 2D symbol and the 3D model, as shown in Figure 1.

2.2 Scenario Modeling

Ad hoc devices can equally accomplish creating the 3D scenario model and photo sets. However, it is possible to use drawings, especially floor plans and operation diagrams. Once all equipment models are already created, these plans can be used as inputs for an automatic generation process. Old floor plans, composed essentially by primitive entities like lines and arcs, have poor level of abstraction to be used in such process. A symbol recognizer can enhance these drawings, replacing a given set of primitives to a composed entity. Once processed, 2D symbols positions found in the drawing can be mapped into 3D models positions on the scene. The process is shown in Figure 2.

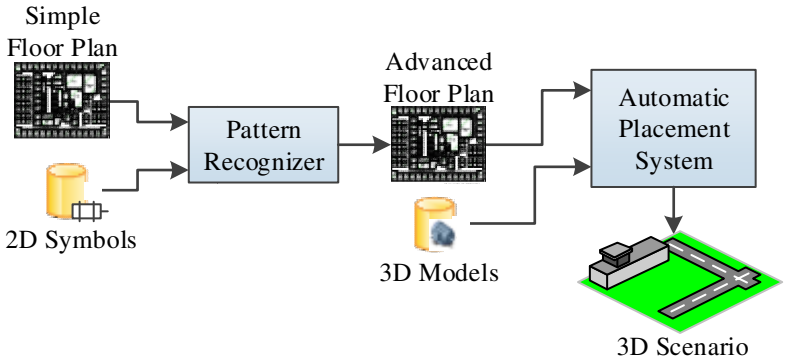

Fig. 2. 3D scenario creation process

3 Pattern Recognizer

This section describes an algorithm to increase the level of abstraction of drawings composed by primitive entities like circles and lines with many occurrences of 2D symbols. A sample drawing is shown in Figure 3, which is actually a fragment of an electric power substation floor plan.

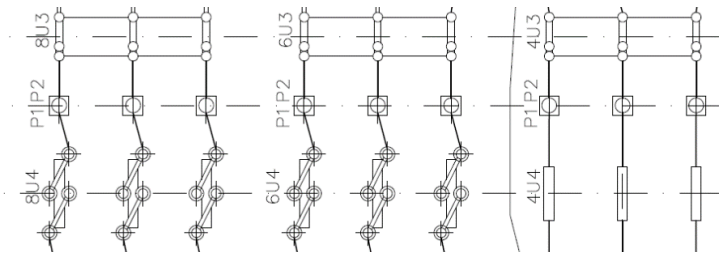

Fig. 3. Fragment of a power electric substation floor plan

Initially, a block must be created for each unique symbol, resulting in a set of 2D symbols as shown in Figure 4.

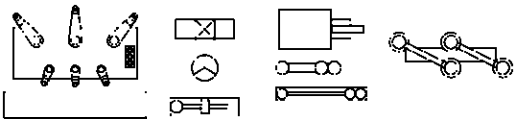

Fig. 4. Set of equipment symbols

A block (composite object) is a list of sub-entities with relative coordinates to a reference point. Figure 5 shows the internal structure of a high voltage disconnector switch block. In this case, P is the reference point, meaning that all sub entities coordinates are relative to P.

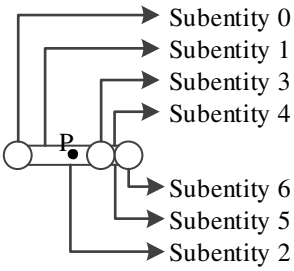

Fig. 5. Power disconnector block internal structure

The proposed algorithm iterates through all the entities in the drawing and checks for pattern matches. The procedure consists of: (i) looking for valid candidates for the first sub-entity, defining the candidate's reference point and (ii) iterating through the next siblings, looking for geometric matches considering the reference points defined in the previous step. These steps are clarified in the algorithm below, which illustrates the recognition procedure for a single block in the form of pseudo-code. Also, only circles and lines will be considered for the sake of simplicity.

```
Pre-defined Variables
  docEntities: list of all drawing entities
  blkEntities: list of all block entities
Procedure
  matches ← []
  sub1 ← blkEntities[0]
  for ent in docEntities:
    if validCandidate(ent, sub1) then
      append ent to matches
      if sub1 is circle then
        baseX ←  ent.center.X - sub1.center.X
```

```
      baseY ← ent.center.Y - sub1.center.Y
    else
      baseX ← ent.P0.X - sub.P0.X
      baseY ← ent.P0.Y - sub.P0.Y
    endif
    for sub in blkEntities[1..last]:
      for otherEnt in docEntities:
        if otherEnt = ent then
          continue
        endif
        if geomMatch(otherEnd, sub, baseX, baseY) then
          append otherEnt to matches
          break
        endif
      endfor
    endfor
    if matches.length = blkEntities.length then
      insert block at (baseX, baseY)
      remove all elements in matches from docEntities
    endif
    clear matches vector
  endif
endfor
```

If the first block sub-entity is a circle (as in the high voltage disconnector example), then all circles in the floor plan with compatible radius are considered valid match candidates for the sub-entity. If, in the other hand, such sub-entity happens to be a line, then all lines in the drawing with similar design attributes, such as length and direction, will be seen as valid candidates. In order to handle imperfections from the digitization process accordingly, and considering that all block dimensions are in the same order of magnitude, a simple absolute error threshold might be used:

$$\text{isNear}(a, b) \rightarrow |a - b| < \varepsilon$$

The value of ε depends on the drawing's scale. The first sub-entity matching also defines the absolute coordinates for the reference point. Then, subsequent entities are processed so geometric compatibility can be verified. In the case of circles, for instance, this means having similar center and radius. Matching lines must have similar length and direction. Each block definition goes through a series of rotation adjustments in steps of 90°, repeating the procedure for each new rotation until the zero-degree orientation is finally reached.

Once all symbols have been detected, the floor plan is ready to be used as input for the automatic placement system discussed in the next section.

4 Automatic Placement System

This section gives a brief overview on the working of the scripts responsible for automatically placing the objects in the 3D scene. A simplified view of the process is shown in Figure 6.

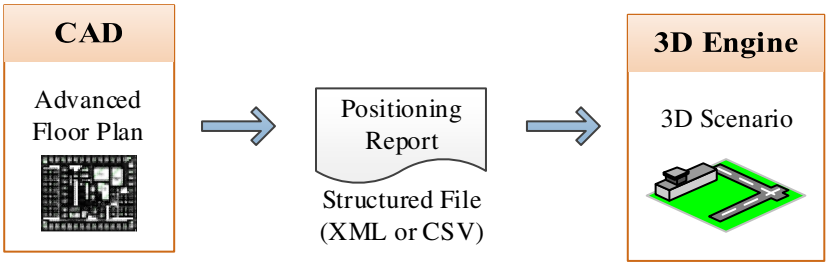

Fig. 6. Placement system based on a structured file

The floor plan blocks' transform coordinates are stored in a structured file by a script executed at the CAD application. One possible structure to this file is a simple CSV with the format:

BlockName;x;y;rot

In this case, x and y are the coordinates for the translation component and rot is the rotation angle in degrees. Although these are the only essential fields for operation, interoperability with future applications may justify the following format:

BlockName;tx;ty;tz;rx;ry;rz;sx;sy;sz

This format comprises all the elements of the transformation matrix. Translation is given by fields *tx*, *ty* and *tz*. Rotation is given by fields *rx*, *ry* and *rz*. Finally, scale is given by fields *sx*, *sy* and *sz*.

This position report serves as direct input to the script in the engine side. The 2D symbols must be coherent with the 3D models, respecting the following conditions: (i) the former's reference point must correspond to the latter's pivot point; (ii) when viewed from a plane parallel to z = K (or y = K in Y-Up systems), the latter's 0° rotation must correspond to the former's 0° rotation and (iii) the block name should match to the 3D model name.

The 3D engine must support executing scripts from the editor (not at runtime) and dynamically instantiate models and apply transformation matrices on them. If these features are available, then importing the objects is only a matter of iterating the Position Report file records and instantiating the entities accordingly. A sample 3D scenario created by using this system is presented in Figure 7.

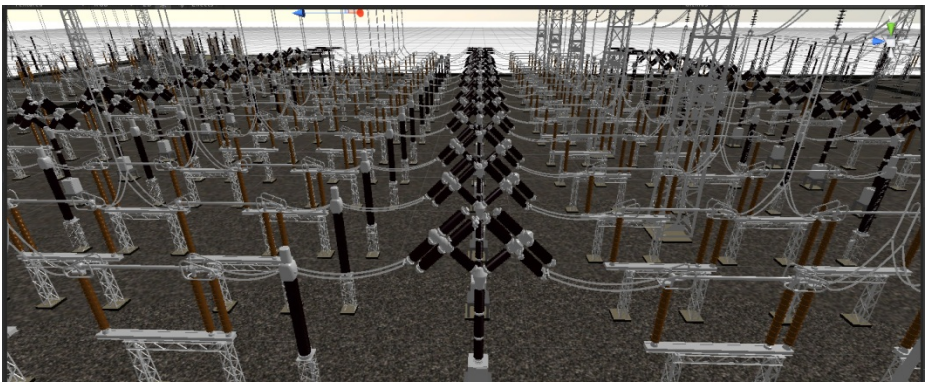

Fig. 7. Virtual environment created with automation tools

5 CAD and 3D Engine Bridge

Both AutoCAD and Unity 3D software packages support scripting with an extensible Application Programming Interface (API). This section describes a method of triggering actions in one application when some particular events occur in the other. For instance, when a given symbol is moved in AutoCAD, its corresponding GameObject in Unity 3D may change its position accordingly. Figure 8 shows the components of this approach.

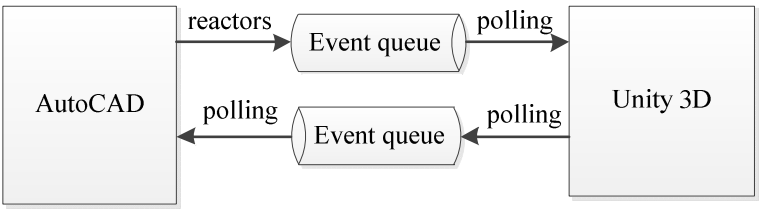

Fig. 8. AutoCAD and Unity 3D bridge

The event queue is a regular temporary file. Whenever a symbol is added, changed or removed inside AutoCAD environment, a line is appended to the bridge file, describing the change. Detecting these changes inside AutoCAD is quite easy, since its programming environment already includes the concept of reactors. Unity 3D supports editor scripts, which can accomplish similar functionality. Events are consumed so that the described changes are committed. With this tool, one can construct the scene using both applications, having the sensation of one being just an extension of the other. The architecture and equipment positioning can be done inside the CAD environment, whereas illumination and other environment characteristics can be defined inside the 3D engine editor.

6 Equipment Gallery

The modeling process can benefit from similarities among devices. Once many virtual worlds are designed, considerable time is spent to retrieve model information. A small information system was developed to solve this particular problem. Instead of limiting organization to files and directories, a web database application provides the team all relevant information in an easy and practical manner. The models are associated with tags, allowing searches based on voltage levels, substations names, number of connection terminals etc. Additionally, a revision control system is used, giving fine-grained control over changes in models.

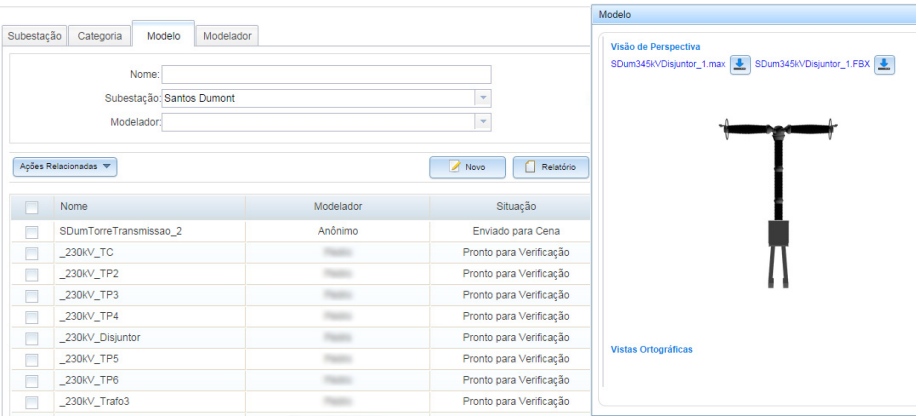

Fig. 9. Equipment Gallery information system screenshots

7 Results

The entire process was applied in the construction of five scenarios. The team was composed by two programmers, four 3D modelers and one database administrator.

The first scenario had been built, initially, using neither the automation tools nor the information system. Once all the device models were made and validated, the task of placing equipment onto the scene and naming objects took approximately 20 man-hours. This first version of the scenario contained non-optimized models with an exaggerated number of polygons, significantly affecting performance both in runtime and while editing the scene.

In order to evaluate the potential of the automation tools developed meanwhile, the same scenario was built using the process based on the floor plan. Firstly, all models were remade using techniques that reduced the number of polygons. The floor plan provided by the energy company was an old format drawing composed by lines and circles. The document was analyzed so that a set of 2D symbols were defined as composite entities. The reference point of each block was compatible with the 3D model pivot point. Then, all primitive entities were checked against the established patterns, resulting in a structured drawing. The entire task of processing the floor plan

took about 6 man-hours. The step of generating the positioning report and feeding it to the scenario editor took less than 3 minutes. Finally, the naming activity was done in 2 man-hours. The environment was presented to the enterprise real system operators, which evaluated the system and reported that the generated model has good quality and accuracy. Currently, the methodology for using the system for training and simulation is being prepared. Equipment data and status are acquired by a Web Service, which may send real or virtual data, the latter being convenient for simulation purposes. Two finished VR scenarios are shown in Figure 10.

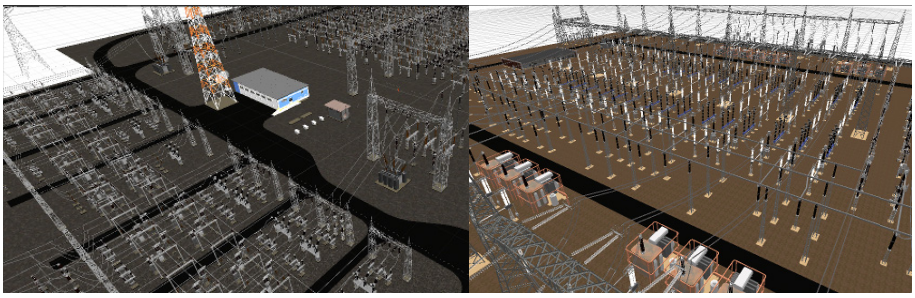

Fig. 10. Electric power substations virtual environments

8 Conclusion and Future Work

The approach proposed in this paper has been successfully applied in the generation of several power substation virtual reality environments. It has proved itself to be time-saving and cost-effective time. Some fine-tuning was needed for the purpose of correcting small inconsistencies introduced by old documents and the non-standardized symbology they contained. Future work includes supporting and evaluating other CAD and 3D engines software packages. The other scenarios constructed with these tools will be evaluated using the same methodology we've undertaken. Also, the pattern detection algorithm could be improved so that: (i) rotation is not restricted to four basic values and (ii) undesirable ambiguities, like a 2D symbol associated with many 3D models, can be solved with the aid of more complex methods. Finally, the concepts here described may reach other scopes, not limited to power substations.

Acknowledgements. The authors thank all the support provided by the project CEMIG GT411 – Virtual Environments for Development Representative of Substations and Power Plants of CEMIG – P&D ANEEL, CEFET-MG (Federal Center of Technological Education of Minas Gerais), and sectors CAPES (Coordination for the Improvement of Higher Education Personnel), CNPq (National Development Council Science and Technology) and FAPEMIG (Support Foundation Research the state of Minas Gerais), for the financial support which enabled this work.

References

1. Okapuu-Von Veh, A., et al.: Design and operation of a virtual reality operator-training system. IEEE Transactions on Power Systems, 1585–1591 (1996)
2. Fanqi, M., Yunqi, K.: An improved virtual reality engine for substation simulation. In: 2nd International Conference on Future Computer and Communication (ICFCC), vol. 1, pp. 846–849. IEEE Press, Wuhan (2010)
3. Guangwei, Y., Zhitao, G.: Scene Graph Organization and Rendering in 3D Substation Simulation System. In: Asia-Pacific Power and Energy Engineering Conference 2009 (APPEEC 2009), pp. 1–4. IEEE Press, Wuhan (2009)
4. Simoes, F., et al.: Challenges in 3D Reconstruction from Images for Difficult Large-Scale Objects: A Study on the Modeling of Electrical Substations. In: 14th Symposium on Virtual and Augmented Reality (SVR), pp. 74–83. IEEE Press, Rio de Janeiro (2012)
5. Lu, J., et al.: A new framework for automatic 3D scene construction from text description. In: IEEE International Conference on Progress in Informatics and Computing (PIC), pp. 964–968. IEEE Press, Shanghai (2010)
6. Coyne, B., Sproat, R.: WordsEye: An Automatic Text-to-Scene Conversion System. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2001), pp. 487–496. ACM, New York (2001)
7. Xu, K., Stewart, J., Fiume, E.: Constraint-Based Automatic Placement for Scene Composition. In: Graphics Interface 2002, pp. 25–34 (2002)
8. Lewis, R., Séquin, C.: Generation of 3D building models from 2D architectural plans. Computer-Aided Design 30(10), 765–779 (1998)
9. Dosch, P., et al.: A complete system for the analysis of architectural drawings. International Journal on Document Analysis and Recognition 3(2), 102–116 (2000)
10. Horna, S., et al.: Building 3D Indoor Scenes Topology from 2D Architectural Plans. In: Proceedings of 2nd International Conference on Computer Graphics Theory and Applications (GRAPP), Barcelona, pp. 37–44 (2007)
11. Yang, R., Lu, T., Cai, S.: 3D building reconstruction based on interpretation of architectural drawings. In: International Conference on Information and Automation (ICIA 2008), pp. 1474–1479. IEEE Press, Changsha (2008)
12. Arroyo, E., Arcos, J.L.L.: SRV: a virtual reality application to electrical substations operation training. In: IEEE International Conference on Multimedia Computing and Systems, vol. 1, pp. 835–839. IEEE Press, Florence (1999)