Real-time Automatic 3D Scene Generation from Natural Language Voice and Text Descriptions

Lee M. Seversky
Lee.Seversky@binghamton.edu

Lijun Yin
Lijun@cs.binghamton.edu

Department of Computer Science
State University of New York at Binghamton
PO Box 6000, Binghamton, NY 13902

ABSTRACT

Automatic scene generation using voice and text offers a unique multimedia approach to classic storytelling and human computer interaction with 3D graphics. In this paper, we present a newly developed system that generates 3D scenes from voice and text natural language input. Our system is intended to benefit non-graphics domain users and applications by providing advanced scene production through an automatic system. Scene descriptions are constructed in real-time using a method for depicting spatial relationships between and among different objects. Only the polygon representations of the objects are required for object placement. In addition, our system is robust. The system supports different quality polygon models such as those widely available on the Internet.

Categories and Subject Descriptors

I.3.6 [**Computer Graphics**]: Methodology and Techniques—*Interaction Techniques*

General Terms

Algorithms, Human Factors

Keywords

Text-to-scene, Storytelling, 3D graphics, Natural language, Voice recognition, Real-time

1. INTRODUCTION

Manual scene composition in 3D is a time-consuming and intensive process. Interacting with 3D graphics-based media and applications requires users to adopt and learn specific graphics tools and interfaces. Such requirements limit the number of potential users and their interactions with a given media. Automatic scene generation reduces these requirements and makes 3D graphics more accessible to users in non-graphics domains. Using natural language descriptions in conjunction with automatic scene generation enables non-graphics areas to take advantage of the benefits of visualization in 3D while reducing the need for graphics specific knowledge. Additionally, real-time scene generation is ideal for collaborative and interactive applications. We present a new system for automatically generating 3D scenes from voice and text descriptions in real-time. Figure 1 shows the components of our real-time composition system.

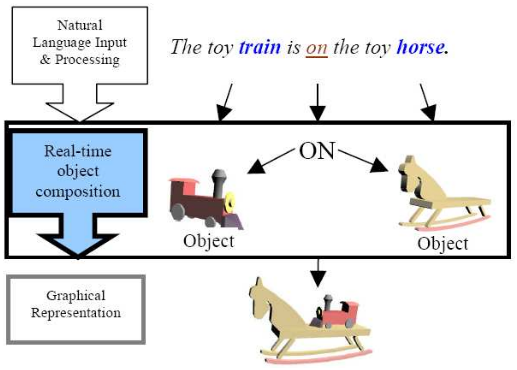
Natural Language Input & Processing
The toy train is on the toy horse.
Real-time object composition
ON
Object
Object
Graphical Representation

Figure 1: Real-time automatic scene composition framework.

This work has significant impacts in any domain that can benefit from real-time 3D graphics and visualization such as education by using 3D scenes to reinforce concepts and ideas during a lesson, discussion, or story. Communication during collaborative environments can be improved by providing a central 3D scene that all members can interact with and see. Other areas such as law enforcement and the movie industry can also benefit from this work.

The presented system constructs a bridge between natural language input and graphical representation. The graphic depiction of a spatial relationship is dependent on the objects and their geometric representation. We focus on a real-time system for correctly depicting objects based on a given relationship. Figure 2 shows a generated scene using our system. The manual definition of per object depiction information is not required with our placement method.

The paper is divided into sections as follows: Section 2 gives an overview of similar composition systems. Section 3 describes in detail the placement algorithm used for scene composition. Section 4 describes our real-time scene composition system. Section 5 discusses our experiments. Section 6 discusses future work and in section 7 concluding remarks are presented.

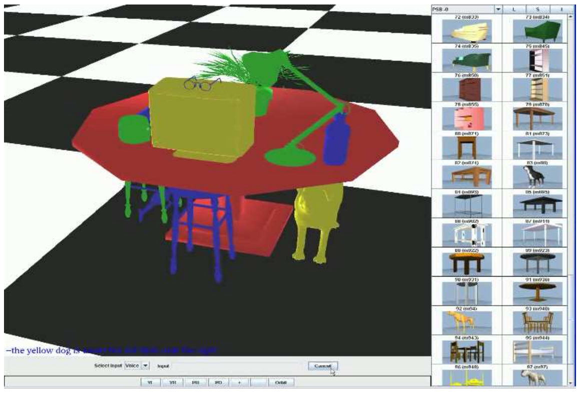
Figure 2: A scene composed of 12 objects generated from voice input in real-time. The following are excerpts from the input description: *A green plant is on the red table. The blue water bottle is next to the green lamp. The blue stool is under the red table towards the front.*

2. RELATED WORK

In our work we generate scenes using a system that automatically determines the correct graphical depiction of an object given a spatial relation, which in conjunction with our real-time constraint and voice and text natural language descriptions, distinguishes our system from other works. The WordsEye text-to-scene conversion system [3] converts text descriptions into 3D scenes. WordsEye extracts 3D information from text and generates low level depictor rules that represent the text graphically. The objects in WordsEye database are manually annotated with information describing the object geometries. Other systems [1] [4] use natural language and 3D graphics to depict scenes specific for a certain domain. Limiting scene generation to a specific domain reduces the objects and types of relationships to a small known set enabling assumptions to be made about the output scenes. A constraint based placement system [6], defines a number of intuitive placement constraints used to place many objects in a scene simultaneously. This approach places objects relative to each other and generates realistic scenes. Unlike our work, scene construction is not in real-time and is not interactive. Additionally, placement is limited to positions that are *on* other objects. Finally, the Put system [2] builds scenes with spatial relationships by assuming simplified object geometry to determine a correct placement and graphical depiction.

3. ALGORITHM

This section describes the placement method for composing objects together using spatial relationships. A spatial relationship represents a spatial term that can be applied to two or more 3D objects (e.g, mouse *under* horse). The following core relationships are supported: *on*, *under*, *above*, *below*, *behind*, *in front of*, and *in*. These relationships provide a high level of descriptive information about object placement allowing realistic scenes to be composed. Additionally, the modifiers: *to the left of*, *to the right of*, *next to*, *towards*, and *center* can be used independently or in conjunction with the core relationship types to provide additional control over placement.

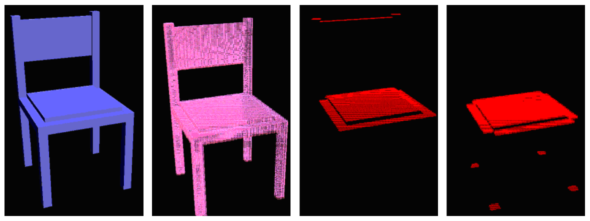
Figure 3: From left to right: a) Original polygon mesh representation. b) Voxel representation. c) Surfaces belonging to the *top* region. d) Surfaces belonging to the *bottom* region.

There are five major stages that make up the placement algorithm. Stage one is a polygon to voxel conversion process which converts the input polygon objects to their voxelized representations for processing. This stage is executed once during a preprocessing pass for each 3D model in the input collection. The second stage uses the voxel representation to extract the object's surfaces and partition them into spatially coherent regions. The third stage selects and validates a location from the regions that satisfies the spatial relationship for the object. The final stage adds the object to the scene.

The input model set consists of polygon models aligned along the positive Z-axis. Given a relationship type γ and two objects, O1 and O2 where O1 is the object to be placed and O2 is the reference object, the placement of the two objects represented by a standard mesh is calculated using the following steps.

3.1 Object mesh voxelization

The first stage converts the polygon mesh representations of objects O1 and O2 into a voxelized representation, V1 and V2. Figure 3b shows an example of the voxel representation. The resulting voxel data consists of numerous cubes that represent the shape of the original model. The voxelization serves two main purposes. First, the conversion process accounts for interior cavities, double walls, and errors caused by cracks or holes that may be present in the original polygon model [5]. The voxelization of the objects provides a consistent normalized representation of all objects to our system independent of the quality of the object's polygon mesh representation. The Binvox 3D Mesh Voxelizer[1] creates the voxel representations with a resolution of a 128x128x128 voxels. A dynamic method for determining resolution based on the object's complexity can also be used.

3.2 Surface extraction & spatial partitioning

Different regions on and around an object correspond to specific spatial relationships. The region boundaries are calculated by segmenting the object's geometry with respect to the surface of the object. Using the voxel representation the surface voxels are extracted. A voxel is on the surface when at least one of its sides is on the exterior of the object. Surface voxels are grouped to form regions based on the side of the object that the surface voxel is located. The surface voxels within these regions are grouped such that the voxels contained in a region all belong to either the top, bottom, front, back, left or right sides of the object.

[1]http://www.cs.johncabot.edu/~min/binvox/

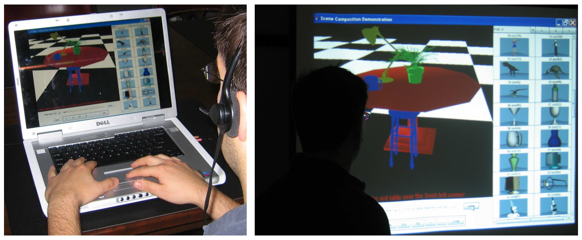

Figure 4: Use of our scene generation system in single user and collaborative environments.

A ternary region tree decomposes each computed region into a set of nine spatially partitioned subsets. Each subset can be further decomposed until each region is one voxel in size. The region tree enables specific areas of an object to be retrieved at different resolutions quickly. The partitioned subsets of the tree map to spatial modifiers such that the nine subsets correspond to the spatial region functions: center, left, right, front, back, upper right, upper left, lower right and lower left.

Regions are computed for objects O1 and O2 from their voxel representations. The regions corresponding to γ are then used to determine placement of O1 relative to O2.

3.3 Placement determination

Given relationship γ, complementary regions R1, R2 are retrieved from O1 and O2. Two regions are complementary if they contain surfaces that are of opposing sides of an object. Consider the case $\gamma = on$, then region R1 is the bottom region of O1 and R2 is the top region of O2.

A given region of object O2 can contain many possible locations for O1 placement. A valid placement for O1 is a location relative to O2 that satisfies two conditions. Firstly, the maximum number of voxel collisions of any adjoining surface must be less than or equal to a threshold (*i.e.*, 0). Secondly, any portions of O2 that reside in the bounding volume of O1 or any other object already in the scene must not collide. There can be multiple valid placements. A single placement can be automatically chosen or the user can select a placement.

Voxel collisions are detected by overlaying O1 positioned at the candidate location with O2. The voxel representation of O2 is merged with O1's representation. Two voxel representations are merged by comparing the voxels from both representations that occupy the same location. There is a collision if both representations have a voxel defined for the same location. The second condition requires that each object already in the scene have its voxel representation kept in the system. Upon checking a new object's placement location the bounding volume of the object is compared with those objects already in the scene. If there are any overlaps, collisions between these voxel representations are checked.

Upon satisfying both conditions, the position of V2 relative to V1 is converted from a voxel to a world coordinate space. The composed objects O1, O2 (e.g. polygon models) are then updated in the scene. Figure 6 shows objects positioned using this algorithm.

4. SYSTEM FRAMEWORK

Our current framework for a real-time automatic scene composition system incorporates a voice and text interface

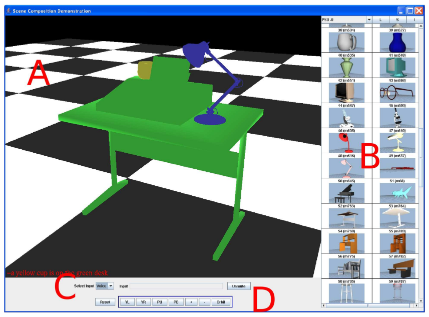

Figure 5: Scene generation interface. a) main viewing panel b) object browser c) input selection d) camera controls.

for composing scenes using natural language spatial relationships. For this implementation we concentrate on descriptions based on spatial relationships and extract only the *object names*, *relationship types*, and a set of *optional attributes* (i.e. color). The primary focus of this initial implementation is to show the contribution of our placement algorithm within an automatic scene placement system and to illustrate its ability to be used in a real-time and interactive environment. Figure 4 illustrates usage examples.

The system contains a database of polygon mesh models representing various types of objects. We chose to compose scenes consisting of objects from the Princeton Shape Benchmark model database [2]. This database contains models collected from the World Wide Web and represents a wide spectrum of models varying in quality. This database tests the robustness of our placement algorithm in handling a wide range of arbitrary models that are freely available.

4.1 Interface

The user interface is divided into three sections. The interface is show in Figure 5. The main viewing window depicts the current scene and is updated in real-time as pieces of a scene are described. On the right, an object viewer displays the available objects for composition. Currently, objects are selected for placement by their 2D thumbnail and their corresponding name. The bottom panel is used to control a single camera in the scene.

Two different input modes exist for scene composition. During voice operation the system listens for descriptions and automatically detects depictable input phrases. A depictable phrase describes at least one spatial relationship between two objects. After depicting at least one phrase the system automatically begins processing and updates the scene when a valid placement is computed. In text mode a field for a textual description is provided.

4.2 Scene Description Processing

A scene description consists of natural language text or voice describing objects in a scene. The input description undergoes natural language part of speech tagging. The tagger builds a tree representing the different language components from the description. The structure is traversed to locate the components that make up a spatial relationship. From these components a scene instance is constructed. A scene instance represents the depictable information extracted

[2]http://shape.cs.princeton.edu/benchmark/

from the phrase. Many scene instances can exist for a given scene description.

For each scene instance, the geometric and voxel representations of the referenced objects are retrieved. This information and the relationship type in used in the placement algorithm described in section 2. The system updates and draws the scene after placement is computed.

5. EXPERIMENTS

Our system is a cross platform application designed to run on common hardware. The scene generation system is implemented in JAVA and uses JView a 3D graphics API developed by the Air Force Research Laboratory for rendering the 3D scenes. The experiments were performed on a Pentium 4 1.8 GHZ machine using both LINUX and Windows operating systems. We selected an inexpensive off the shelf microphone for our voice driven scene generation tests.

The speed for which an object can be placed in a scene depends on the reference object and the number of objects already placed in the desired placement area. On average successful placements of objects similiar to those in Figure 2 require 650ms.

The placement algorithm requires that any new objects to be placed do not collide with the reference object. Performance is impacted when no valid placement within a region exists or there exists many possible placements but only a few are valid placements. When there is no valid location the algorithm checks N voxels representing the entire surface of a given region. The number of checks can be reduced by defining a threshold for the maximum number of locations to validate or by uniformly sampling the entire surface at a given level in the region tree.

6. FUTURE WORK

Scenes generated from our system are promising. However we see our system as an initial framework which focuses on the depiction of natural language descriptions in real-time. In our current work we have assumed that the descriptions contain spatial relationships. The linguistic challenges of more general descriptions have not been addressed.

Currently our system constructs scenes according to a specified description. This allows objects to be placed and repositioned effectively. In addition to spatial relationship based placement we will support action based positioning in the future, for example *rotate*, *turn around*, *zoom in/out*. The *move* action is already supported in our system.

We have observed that it is desirable for such a system to also provide feedback in terms of composition suggestions or hints. For example when an object is an incorrect size for the desired relationship the system can suggest or automatically resize the object. Additionally, manipulating object orientation will increase the flexibility and realism of the generated scenes. Physics is also not addressed, however, we believe physics can be applied on top of our positioning method.

Finally, the voxelization technique is effective for normalizing model mesh representations, however there are alternative methods for detecting object collisions when positioning objects. These methods should be evaluated as alternatives to voxelization.

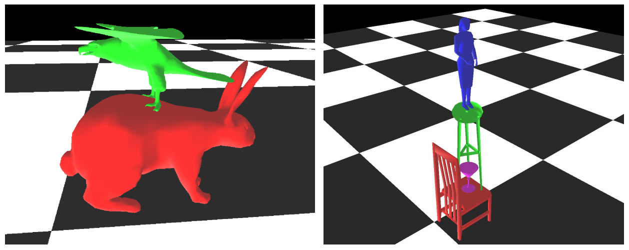

Figure 6: Placement of multiple objects from the following input descriptions a) ***The green bird is on the red rabbit.*** **b)** ***The green stool is on the red chair. A purple wine glass is under the green stool. The blue lady is on the green stool.***

7. CONCLUSION

Our scene generation system incorporates natural language voice and text input to construct 3D scenes in real-time based on spatial relationships. The system is capable of composing objects without a priori information about the objects and is robust to varying quality polygon models. The use of voice and text enables scenes to be described without requiring knowledge about 3D graphics or specific tools. This type of system forges the way for systems in which interaction with and construction of 3D graphical media can be as natural as communicating via text or voice. We hope this work serves as a foundation to much more encompassing and automated systems.

8. ACKNOWLEDGMENTS

We thank Jason Moore at the AFRL for his suggestions and our lengthy creative discussions and Richard Hodges for setting up and recording the live video demonstrations of our scene composition framework.

9. REFERENCES

[1] O. Akerberg, H. Svensson, B. Schulz, and P. Nugues. Carsim: an automatic 3d text-to-scene conversion system applied to road accident reports. In *EACL*, 2003.

[2] S. Clay. and J. Wilhelms. Put: language-based interactive manipulation of objects. In *IEEE Computer Graphics and Applications*, pages 31–39, 1996.

[3] B. Coyne and R. Sproat. Wordseye: An automatic text-to-scene conversion system. In *SIGGRAPH*, pages 487–496, 2001.

[4] F. Durupinar. Intelligent indexing, querying and reconstruction of crime scene photographs. In *TAINN*, 2004.

[5] F. Nooruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. In *IEEE Transactions on Visualization and Computer Graphics,*, 2002.

[6] K. Xu, J. Stewart, and E. Fiume. Constraint-based automatic placement for scene composition. In *Graphics Interface*, pages 25–34, 2002.