Procedural Human Posing Using CGA Grammars

Stefan Fiedler
Student

Martin Ilčík
Supervisor

Institute of Computer Graphics and Algorithms
Vienna University of Technology
November 2, 2009

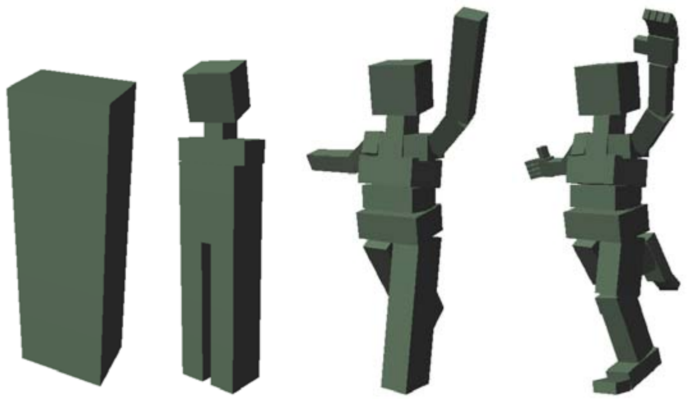
Figure 1: A humanoid figure (right) modeled from an initial shape (left) using a shape grammar. Two intermediate steps (center) show partially constructed figures. From left to right, production rules incrementally refine the figure and its pose.

Abstract

This report presents a grammar for the procedural modeling of humanoid characters and poses based on CGA shape, a shape grammar for computer generated architecture. Various models can be derived with the same set of parametrized rules for geometric operations, and include a skeletal system for posing. The main part of this report defines basic rules and their effect on shapes and skeletons and discusses the results of an actual implementation of the grammar with examples.

1. Introduction & Motivation

Human and humanoid characters play an important role in application areas of computer graphics like simulations, computer games and movies [Preda et al., 2006]. Common modeling techniques for human characters typically involve creating a geometric representation of the surface, a process that is mostly human-guided and, depending on the details of the model, is very time-consuming and requires great care [Ratner, 2003]. When models are animated, at least one other step is usually necessary, in which a skeleton consisting of bones and joints is created and connected to the model surface [Maestri, 1999]. While allowing best control of the final results, it can be difficult to create variations of existing models, up to the point where it is more efficient to create a new model instead of modifying a similar one.

Since humanoid models often share the same structure of the artistic anatomy, but vary in parameters like stature or body weight, being able to create a variety of similar models from a template by just specifying the desired parameter values would be very useful. In this report we present an approach to the problem that uses procedural modeling based on a type of shape grammar called CGA shape. Our main motivation was to investigate the possibilities and difficulties of the application of CGA shape to the field of human modeling.

CGA shape was introduced in [Müller et al., 2006] and developed specifically for the automatic design of computer-generated architecture. In the same paper Müller et al. have shown the grammar to be useful for efficiently generating realistic, large scale, highly detailed models of cities from a relatively small set of rules. The grammar is explained in some detail in the next chapter.

The scope of this work comprises of two main topics: the generation of simple, three dimensional humanoid models, and of a skeleton attached to the model geometry, suitable for the application of forward and inverse kinematics. Another goal was to use the skeleton to apply different poses to the model.
Several problems had to be solved for this project. First we had to decide on the definition of the grammar and its most important rules, together with their geometric interpretation. The grammar rules for the manipulation of geometry are similar to CGA shape rules, with differences explained in section 3.

To simplify posing humanoid models, a uniform mathematical description of the kinematic relations between parts of the body is necessary. For this a skeletal system is attached to the model. The kinematic skeleton is created automatically during the application of shape rules, and can be adapted through an extension of the grammar. Section 4 explains the mechanisms and rules related to the skeleton.

Finally, being able to actually perform geometric operations on arbitrary shapes requires the use of appropriate data structures and algorithms. For this purpose CGAL, the Computational Geometric Algorithms Library (www.cgal.org), provides most of the needed functionality, specifically triangulation of and boolean operations on two- and three-dimensional polytopes. In the following paragraphs each of the related works is described in more detail.

2. Related work

CGA shape, as already mentioned, is a type of shape grammar developed for the procedural modeling of computer generated architecture [Mueller et al., 2006]. It is partially based on an earlier grammar introduced by [Wonka et al., 2003]. Some of its main features are the ability to create large scale, highly detailed mass models, it employs conditional, context-sensitive, stochastic evaluation of rules to retain realistic layouts of architectural elements while allowing for a wide variety buildings generated from a set of rules, the basic rules are general enough to support the development of rule sets for different architectural styles, and the model generation does not require user input to select rules. Key elements of CGA shape are the notion of shape, the definition and geometric interpretation of basic rules, and the control of their evaluation.

Each shape consists of a symbol as well as geometric and numeric attributes. Symbols relate to the semantics of shapes and can identify them, especially for the purpose of selecting applicable rules. Geometric attributes define the visible form of a shape, and most importantly include an oriented bounding box called scope, given as a spatial position, three orthogonal vectors forming a coordinate system, and a size vector. Shapes can be three- or lesser-dimensional. Numeric attributes allow to parametrize rules and to further control the derivation process.

One part of the basic rules modifies the scope by translation, rotation or scaling, respectively, which also affects the geometry contained within the scope. The split rule creates two or more shapes of the same dimensionality by splitting the scope along one or more of its axes. Split sizes and symbols of resulting shapes are specified as parameters of the rule. The repeat split rule works similarily but creates as many shapes of the same kind as will fit into the original shape. Since both split rules should work well on a range of differently sized scopes, some of the split sizes have to be scaled, but some elements are more suitable to scaling than others. To accomodate for this fact, split sizes can be absolute or relative to the size of the original scope. Finally, the component split rule decomposes a shape into its lower-dimensional features, for example to create a shape for each face of a three-dimensional shape. To go back to higher dimensions, shapes can be extruded, expressed in the grammar as scaling along a scope axis.

CGA shape is a sequential grammar, which means that one rule is applied at a time. On the one hand, the order of application is determined by the priority of rules, so that rules applied earlier coarsly structure a model, and later rules gradually add more details. Each rule can also have preconditions based on numeric attributes of shapes to decide on its suitability. If several rules are applicable, one is selected based on a probability value for each rule.

CGAL, the Computational Geometric Algorithms Library (www.cgal.org), offers a rich collection of efficient and reliable geometric algorithms, of which the ones pertaining to boolean operations on three-dimensional Nef polyhedra are especially important to this project. A Nef polyhedron as defined by Swiss mathematician Walter Nef is a point set generated from a finite number of open halfspaces by set complement and set intersection operations [Nef, 1978]. By definition, Nef polyhedra are closed under boolean operations, like intersection and difference, and topological operations, like interior and closure. Therefore it is possible to represent non-manifold situations, open and closed sets, and mixed-dimensional features. This is a major advantage over other representations, which may not be able to accurately represent all possible results of boolean operations on polyhedra. The data structure used to represent Nef polyhedra in CGAL and the implementation of boolean and other operations is described in detail in [Hachenberger, 2006].

In the field of robotics the problem of forward and inverse kinematics has been extensively studied and numerous methods for calculating solutions have been developed, which are also used for posing and animating humanoid models [Smidt, 1998]. A common method for character animation uses a skeletal system to describe poses and movements [Maestri, 1999]. Generally speaking, a skeleton consists of rigid sections connected by rotational joints. Each section has a joint that connects it to a parent section, and the joints describe the rotation and translation of a section relative to its parent. Together they form a hierarchy where the position of each section depends on the pose of all those which precede it in the skeleton hierarchy. Different poses can be applied to a model by rotating the joints, and it is possible to limit the rotations of joints to an arbitrary range relative to a rest position.

3. Modifications to CGA

Compared to CGA shape, the grammar used in this project is different in several aspects. Some differences are due to the limited scope of the project, others are intended to adapt the grammar to the modeling of humanoid characters.

Both the geometry and the skeleton are generated from a hierarchy of parametrized rules to support the creation of a variety of models by changing the parameters. Using a hierarchy instead of a set of rules allows to specify their order of application and thus iteratively refine the model to add more details. Unlike in CGA shape, where rules have priorities, in this project production rules are attached to other rules and apply only to their resulting shapes.

Operations on the geometry currently support polygons with holes (for two-dimensional shapes) and three-dimensional, two-manifold polyhedra. This is a small limitation as non-manifold situations usually do not occur in humanoid models. The repeat rule is not implemented because it is of lesser importance for our application than it is for modeling of architecture. The split rule can now split the scope along an arbitrary direction, not only along an axis. To simplify kinematics, each scope is now embedded in a kinematic section, which means its position and orientation are relative to the coordinate system of the section, not the world coordinate system. Figure 2 shows a shape in relation to sections.

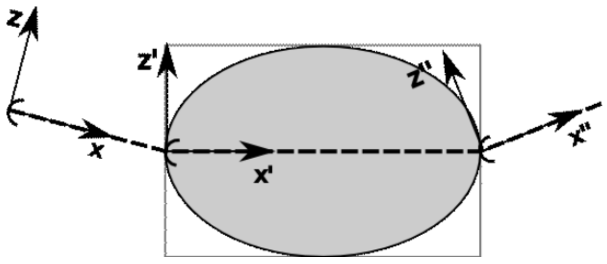

Figure 2: Shape, scope and skeleton. An elliptic shape is located inside a bounding box representing its scope. The scope is positioned inside a kinematic section with local coordinates (x', z'). The section has a parent (x, z) and one child section (x'', z''). Joints located at the origin of each local coordinate system connect each section with its parent. The figure shows joints as semi-circles. Dashed lines represent the bones of the skeleton, rigid connections between adjacent section.

For component splits it is sometimes necessary to assign specific symbols to different components, but in most cases it is not possible to guarantee a certain indexing order of components or even their numbers. Therefore the component split rule takes as an additional parameter a function which assigns symbols to component shapes based on their (geometric) attributes, for example the normal vectors of faces. A new rule moves a pivot point, which is by default located at the scope position, relative to the scope to allow for easy scaling of shapes around an arbitrary point in space, which would otherwise require an additional translation dependent on the scope size and point.

4. Skeleton grammar

The humanoid characters can be posed by using a skeletal system which describes possible movements and actual positions of parts of the body. It is created automatically along with the shapes of the model, as the production rules for shapes also change the structure of the skeleton. For this, the rules create joints and sections if necessary and connect them to existing sections. An additional set of rules allows to modify each joint and its current pose. The shapes of a model are themselves attached to the sections and move with the skeleton.

Whenever a production rule is applied to a shape, the skeleton structure of the resulting shapes must be defined (Figure 3). Most importantly, the rules define for each resulting shape the section to which it is attached, as well as its parent, rotation and translation. For rules that produce only one shape, the result replaces the original shape in the skeleton and therefore its section is the same as the original.

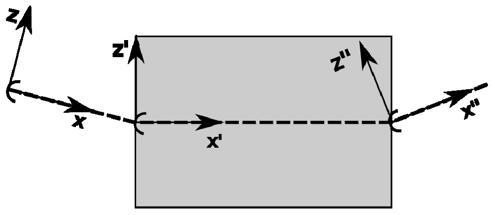

Figure 3: Shape operations can also affect the local skeleton, in this case the section (x', z') of a rectangular shape and its parent and child. When a section for a new shape is added, its joint must be placed and connected to the rest of the model.

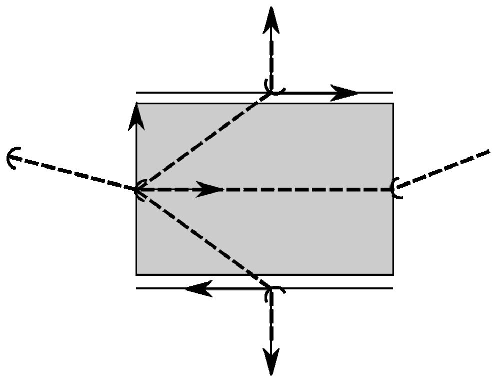

Figure 4: A component split rule applied to Figure 3 can for example create new shapes for the top and bottom sides. Each component lies in its own section which is connected to the original section. The rule places new joints at the center points of sides. The z-axis of new sections point along the normal vector of side faces. This results in a rotated coordinate system for the component at the bottom.

In a component split, the original shape is copied to the result and it becomes the parent of all the components (Figure 4). The joint of each component is placed at its center, and its section is rotated according to the orientation of the component. For the clip rule, the new shapes also become children of the original shape, but their sections are not rotated.

The split rule can create two different skeletal structures. The first type of split connects pairs of adjacent shapes with joints and thus creates a chain of shapes which replaces the original shape in the skeleton. The first shape in the result is placed in the same section as the original, and the last shape becomes the new parent of the original children (Figure 5). This can be seen as splitting a section into several ones, with joints between them. This split rule places new joints on the split planes and on a line that is parallel to the split direction and lies on the joint position of the original shape. The split direction influences the order in which the split shapes are connected and should usually point away from the parent of the original shape to its children. Connecting parent and children to other than the first and last section is also possible (Figure 6).

The second type of split represents a fork in the skeleton and can be used to model legs or fingers (Figure 7). In this split each new shape is connected to the parent of the original shape. The joints are placed at the midpoint between split planes, on a line that is parallel to the split direction and goes through the original joint position.

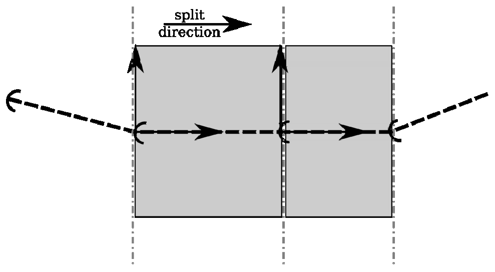

Figure 5: An example split rule applied to Figure 3 creates two new shapes and connects them in an ordered sequence. A direction vector defines the split planes, the order of resulting shapes and the placement of new joints. In this split the first shape (on the left) has the same parent as the original, and the child gets connected to the second shape.

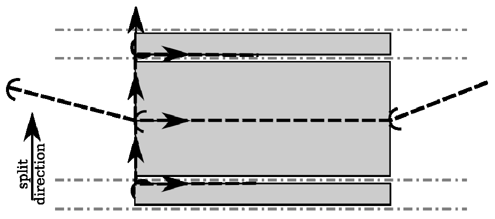

Figure 6: Here the shape of Figure 3 is split in three along horizontal planes. As shown here any of the new sections can replace the original section in the skeleton hierarchy. All other shapes resulting from the split become its direct or indirect children.

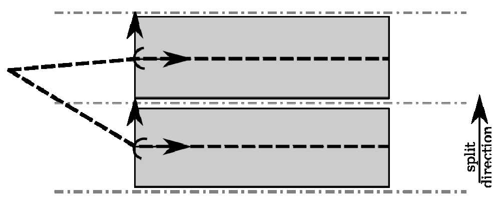

Figure 7: Another type of split connects each new shape to the original parent section instead of just one. This allows to create an arbitrary number of children at each level of the skeleton hierarchy.

Since the default placement of joints may not be the best for all cases, a production rule exists which places a joint at an arbitrary position relative to the connected shape. Other rules allow to change possible movements by setting the rest position and rotation limits of a joint. Finally the current rotation of a joint is changed by its own rule, which is used to apply actual poses to the model.

5. Conlusions/Summary

This report has shown how the CGA shape grammar can be adapted to the modeling of humanoid characters, and how the grammar can be extended to create and modify a skeleton for the purpose of posing characters. Figure 8 shows several models created from a set of parametrized rules. So far the models described with the grammar are rather simplistic in shape, and producing more detailed models would require a large amount of splits. CGA shape uses pre-made instances of architectural elements which replace existing shapes and add details where necessary. In this case the pre-made instances could be taken from a geometry library of parts of the body.

The component split works well for simple shapes, but for more complex shapes it might not be as useful to treat each face as a single shape. Instead it might be better to create components out of larger areas of connected faces.

The choice of coordinate systems for new shapes is difficult because it has an influence on the rotational direction of joints and the direction of extrusion and other geometric operations. As of now, it requires special care when a rule for one side of the model, for example the left arm, is applied to the other side because the axes of the shapes point in different directions. This could be solved with mirrored coordinate systems so that for example one axis of each arm points towards the front of the model, and another points upwards. The mirrored system would however be left-handed, and would have to be handled properly by the production rules. The skeleton that is created can be successfully used to pose the model but suffers from the same problem with coordinate systems.

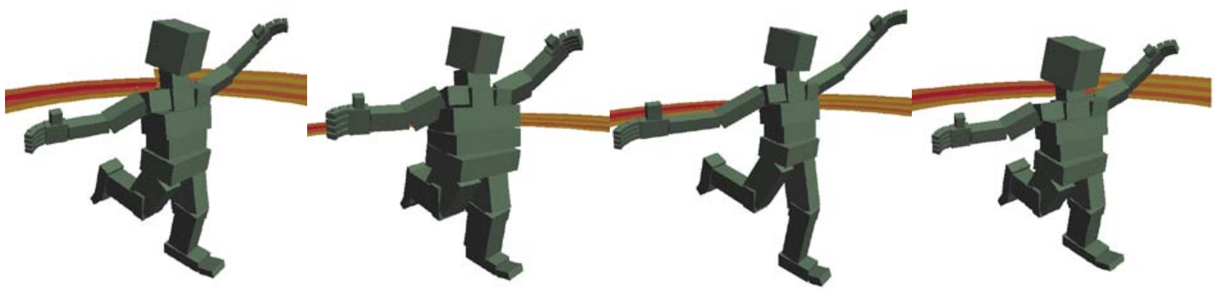

Figure 8: Four models created from the same set of rules with different parameters. The images from left to right show an average figure, a heavy figure, a figure with long limbs, and a figure of smaller stature. The skeleton created along with the shapes allows to apply the same pose to each model.

6. Literature

HACHENBERGER, P. 2006. *Boolean Operations on 3D Selective Nef Complexes: Data Structure, Algorithms, Optimized Implementation, Experiments, and Applications.* Doctoral thesis, Naturwissenschaftlich-Technische Fakultäten, Universität des Saarlandes.

MAESTRI, G. 1999. *Digital Character Animation, Volume 1 – Essential Techniques*. Newriders.

MÜLLER, P., WONKA, P., HAEGLER, S., ULMER, A., AND GOOL, L. V. 2006. *Procedural Modeling of Buildings.*

NEF, W. 1978. *Beiträge zur Theorie der Polyeder*. Herbert Lang, Bern.

PREDA, M., SALOMIE, I. A., PRETEUX, F., AND LAFRUIT, G. 2005. Virtual Character Definition and Animation within the MPEG-4 Standard. *3D Modeling and Animation: Synthesis and Analysis Techniques for the Human Body*. IRM Press, Sarris N., Strintzis, M. G., Ed.

RATNER, P. 2003. *3D Human Modeling and Animation, 2nd Edition*, John Wiley & Sons, Inc.

SMIDT, W. 1998. *Verallgemeinerte inverse Kinematik für Anwendungen in der Robotersimulation und der virtuellen Realität.* Diploma thesis, Institut für Roboterforschung, Universität Dortmund.

WONKA, P., WIMMER, M., SILLION, F., AND RIBARSKY, W. 2003. Instant architecture. *ACM Transactions on Graphics 22*, 3, 669–677.