INTERACTIVE GRAPHICS SIMULATION SYSTEM(IGSS) FOR THE ASSEMBLY OF MECHANICAL PARTS

Hema A.Murthy and R.G.S.Asthana
Department of Electrical and Computer Engineering
McMaster University, Hamilton, Canada

ABSTRACT

In mechanical assembly process, the precise positioning of parts is of utmost importance. IGSS, Interactive Graphics Simulation System, provides the user with a set of tools for the creation, modification and display of mechanical assemblies. The system defines a phrase structured grammar for the creation and assembly of mechanical parts. The grammar defines a set of rules(productions) for the assembly of parts. Assemblies defined by the user which do not conform to the constraints of the language generated by this grammar are declared as illegal and hence not created.
KEYWORDS: parts, partinstances, assembly, assembly sentence, grammar, productions.

INTRODUCTION: The set of tools available in IGSS for the assembly of mechanical parts are: a graphics package, an object library, a grammar, and an editor. The graphics package is a general purpose three-dimensional graphics package(1) The object library provides a set of parts which can be used to create an assembly. In its present form the object library consists of four parts, namely, a nut, a bolt, a bracket and a plate. The user can also create his own parts which will be included in the library.

A part can be defined by (i) instances of parts stored in the object library, (ii) polygonal description(2) or (iii) its orthogonal views(plan, elevation, sideview)(3). The system is capable of creating a perspective view from a user defined location or a parallel view in a user defined viewing system. Correct positioning of parts is important in the creation of an assembly. IGSS uses (i) a set of rules(legal steps) which defines the position of the parts with respect to each other and (ii) the alignment axis associated with each part at the time of its creation to perform the assembly.

A phrase- structured grammar has been developed and implemented for describing the parts and to assemble them one by one into a final product. A set of legal steps(productions) for the assembly of a set of given parts is defined. These legal steps are defined in terms of an assembly instruction which forms part of the language generated by this grammar.

The editor basically decodes the commands given by the user. It processes the assembly instructions as per the requirements of the grammar. If the assembly is valid the instructions are translated into instructions in the graphics package.

IGSS also provides the facility (i) add(delete) parts to(from) the library, (ii) add(delete) partinstances to (from) an assembly, (iii) add(delete) steps(productions) to (from) an assembly, (iv) scale, rotate, shear assemblies. Figure 1 shows the block diagram of the system developed.

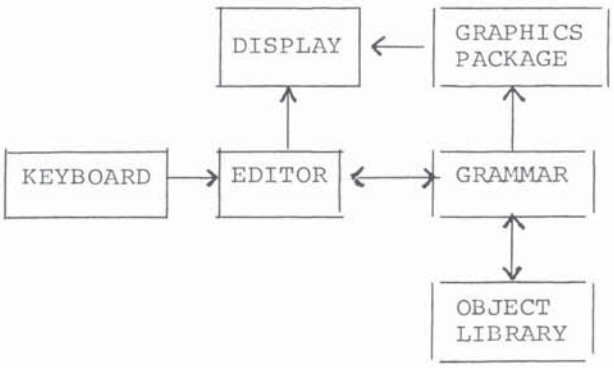


Fig.1: Blockdiagram of the system.

The following discussion defines the grammar and explains a few of the commands in the editor. The grammar used is a tree grammar defined by a four tuple (4):

$$G=\{V_N,\ V_T,\ P,\ S\}\ ,$$

where

V_N= {assembly , part , plan , elev , sview , polygons , arcs , vertices , radii , angles} ,

V_T= {nut, bolt, plate, bracket, V_{11},.... V_{MN}, R_1,...,R_M, A_1,...,A_M} ,

where

V_{11},...,V_{MN}: the vertices of all polygons (1,...,M), (the vertices are defined in a clockwise or anticlockwise direction),

R_1,...,R_M : the radii of all arcs (1,.M)

A_1,...,A_M : the angles of all arcs(1,.M)

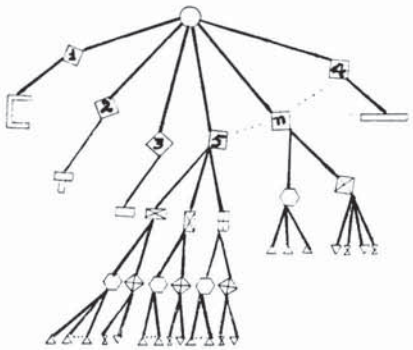
The productions P are as follows:

assembly --> <parts>
part --> <plan>,<elev>,<sview>
plan --> <polygon>,<arc>
elev --> <polygon>,<arc>
sview --> <polygon>,<arc>
part --> <polygon>,<arc>
part --> plate
part --> nut
part --> bolt
part --> bracket

polygon M -- V_{M1} ,..., V_{MN}

arc M -- R_M , A_M

Figure 2 shows the derivation tree for the above defined set of productions. Each part is associated with a name when it is created. The starting symbol S is the root of the tree in Figure 2.



(a)

(b)

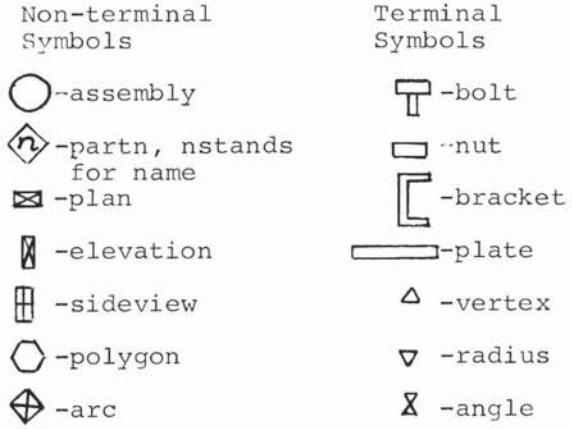


Fig.2: (a) The derivation tree for an assembly
(b) Conventions.

Syntax of an assembly: The basic vocabulary defining a complete assembly consists of four different letters of the alphabet and a few special characters as shown in TableI.

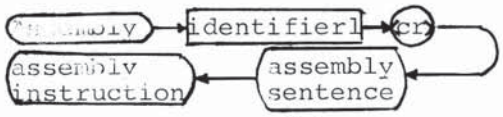
Table I

Sr. no.	Special characters	letters of the alphabet
1	.	A(above)
2	,	B(below)
3	:	L(left)
4	-	R(right)
	<cr> (carriage return)	

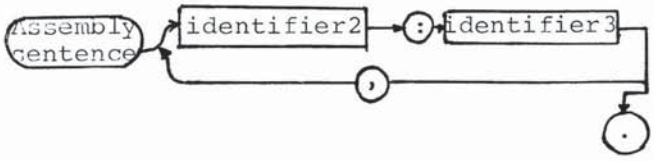
An assembly is defined by an assembly sentence and an assembly instruction as shown in the syntax diagram of Figure 3. Starting at the diagram named assembly, a path through the diagram defines a syntactically correct assembly. Figure 3 is analogous to the syntax diagram used to define Pascal statements in Ref.6. The assembly sentence defines the number of instances of a part that will be required in an assembly. For example 1:2, 2:3.This implies that two instances of part 1 will be required and three instances of part 2 will be required in the assembly. The assembly instruction defines the position of a partinstance in an assembly with respect to its closest neighbours. Letters of the alphabet in Table 1 are used to define the positions. For example 1a-2aRB is a syntactically correct step

in an assembly instruction. Here 'R' implies that 2a will be rotated about its centre by 180°. The 'B' implies that 2a is below partinstance 1a. The partinstances are aligned along the vertical line of the axis of partinstance 1a. The position of the partinstance to the right of the '-' is defined with respect to the partinstance on the left of the '-'.

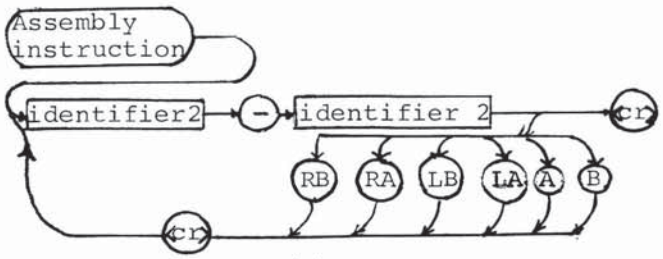
Semantics of an assembly: (1) No more than three partinstances can exist at an axis of a partinstance.
(2) A partinstance cannot be defined both above and below another partinstance.
(3) A partinstance cannot be defined both to the left and right of another partinstance.
(4) A bolt cannot be defined between a plate and bracket.
(5) A nut cannot be defined in between a plate and a bracket.
(6) A nut cannot have a partinstance exactly below it.
(7) A bolt cannot have a partinstance exactly above it.



(a)



(b)

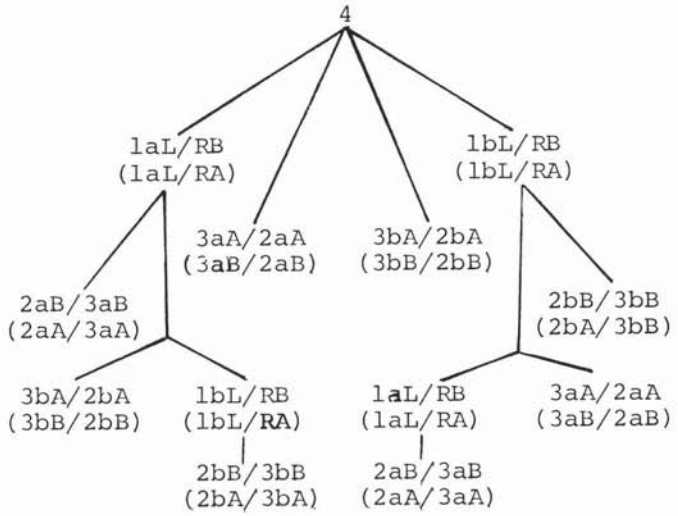


(c)

identifier 1 - assembly name, must be alpha-numeric, can be a maximum of four characters.
identifier 2 - partinstance- or partnames, must be alphanumeric, maximum four characters long, should not contain any of the symbols used in the vocabulary.
identifier 3 - must be an integer. This defines the number of instances of a part in an assembly.

Figure 3: (a) Syntax of an assembly, (b) syntax of an assembly sentence, (c) syntax of an assembly instruction.

The following example illustrates the salient features of the editor.
Command : CREATE <cr>
part/assembly : assembly <cr>
partnumber, number of parts :1:2,2:2,3:2, 4:1 cr
In the above command '1:2' implies that two numbers of part 1 will be used in the assembly. Hence, they are renamed as '1a' and '1b'. The relational graph shown in Figure 4 describes the legal productions for the parts chosen.



Convention :
A ⇒ above the part of the previous node.
B ⇒ below the part of the previous node.
L/R ⇒ left or right of the part of the previous node.
a,b ⇒ suffixes to differentiate parts of same type.
1,2,3,4 ⇒ partnames as defined in Figure 2.

Fig.4 : Relational graph for assemblies using two brackets, two bolts, two nuts and one plate.
The following set of productions(legal steps) create an assembly.

Productions : Step 1 : 4-1aRB<cr>
Step 2 : 4-2aA<cr>
Step 3 : 4-1bLB<cr>
Step 4 : 4-2bA<cr>
Step 5 : 1a-3aB<cr>
Step 6 : 1b-3bB<cr>
Step 7 : <cr>

Note : <cr> without an arguement will terminate the set of productions.
Step 1 : implies that the plate goes above the bracket 1a and the longside of the bracket is to the right of the first hole in the plate.
Step 2 : implies that the bolt goes above the plate.
Step 3 : implies that the plate is also above the bracket 1b and the longside of the bracket is to the left of the second hole in the plate.
Step 4 : implies that the bolt 2b is above the second hole in the plate.
Step 5 : implies that the nut 2a is below the bracket 1a.
Step 6 : implies that the nut 2b is below the bracket 1b.

Command : DISPLAY <cr>
This command displays the figure as shown in Figure 5(a). Different views of the object can be obtained using the VIEW command. The following command repertoire will create the assembly shown in figure 5(b).

Command : VIEW <cr>
Perspective/parallel : perspective cr
eyepos : X Y Z cr
Partinstances can be added or deleted to an assembly using the following commands:
Command : ADDPART <cr>
partnumber :
Command : DELPART <cr>
partnumber :
Figure 5(c) shows the deletion of a partinstance. Productions can be deleted or added using the following commands :
Command : ADDPROD <cr>
step N :
Command : DELPROD <cr>
step N :
Commands are also available to scale, rotate, shear assemblies. See Figs.5(d), 5(e), 5(f).
The quad tree visibility algorithm is used to remove hidden lines(5).

IGSS can be used to create new 3-dimensional parts or assemblies and view them from any desirable angle to achieve a final product. With an enhanced library and a knowledge of all possible legal assemblies the system can be utilized for practical applications.

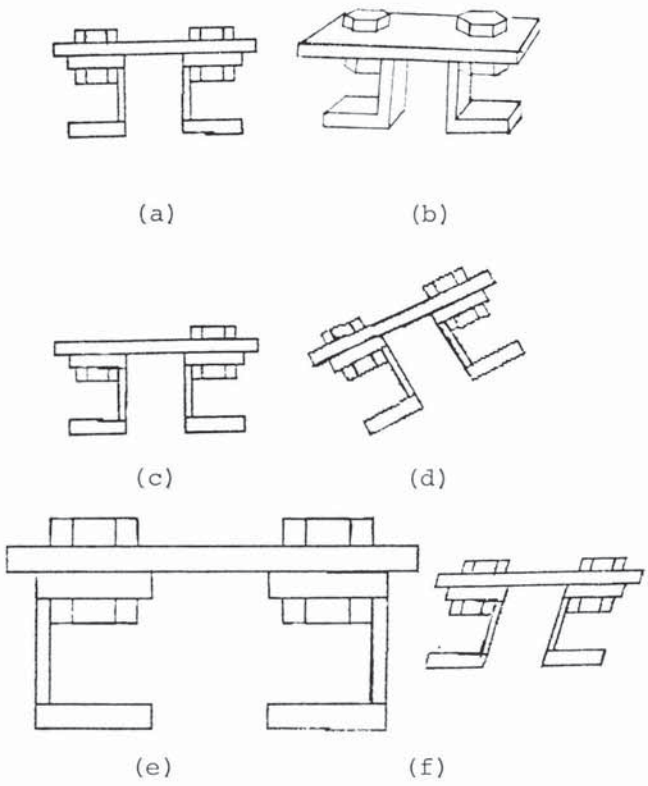


Fig.5 : (a) Orthogonal view of picture created in the example, (b) perspective view of the same, (c) deletion of a partinstance, (d) rotation of an assembly (e) scaling of an assembly and (f) shearing of an assembly.

REFERENCES:
(1) R.R.Joshi and Hema A. "GRASP- A 3D graphics system for Pascal users", (to appear in Computers and Graphics).
(2) K.Sugihara, "Picture language for skeletal polyhedra", Computer Graphics and Image Processing, 8, 1978.
(3) R.M.Haralick and Queeney D. "Understanding engineering drawings", Computer Graphics and Image Processing, 20, 1982.
(4) K.S.Fu, Syntactic pattern Recognition and Applications, Prentice Hall, Inc, New Jersey, 1982.
(5) T.Pavlidis, Algorithms for graphics and image processing, Computer Science Press, 1982.

(6) K.Jenson and Wirth N. Pascal User Manual and Report, Springer Verlag, New York, Heidelberg, Berlin, 1978.