AIAA 99-0829

Interactive Parametric Geometry Design

J.C.Trapp, H. Sobieczky
DLR German Aerospace Center

37th AIAA Aerospace Sciences Meeting and Exhibit
January 11- 14, 1999 / Reno, NV

INTERACTIVE PARAMETRIC GEOMETRY DESIGN

Jens Trapp, Helmut Sobieczky
DLR, German Aerospace Center
Bunsenstr. 10
D-37073 Göttingen, Germany

Abstract

Mathematically accurate shape generation for aerospace applications needs to keep up with progress in CFD and production tools. One reason is the present lack of good and powerful tools for the geometric definition of airframes. Standard CAD tools may be used, but they do not support the design of aircraft in a problem-oriented way. Based on well established and tested parametric algorithms a new design tool will be presented which offers the possibility to define configurations in an object-oriented manner. Interactive control makes the tool easy to use. It may be used as a preprocessor for any kind of CAD program to build windtunnel models or to define the net topology needed for numerical simulation. Furthermore it will fit perfectly in an automatic optimization environment due to the parametric definitions of the geometry. This paper covers a description of the underlying geometric alorithms, the software design and the usage of the new tool.

Introduction

In the recent past improved computer power and operational 3D flow solvers made the calculation of complex aircraft configurations faster and more affordable. In the near future those calculations will be performed on desktop computers instead of supercomputers. This will change the way in which aircraft will be designed. Today, complex configurations of complete aircraft are calculated only for a few shapes with different parameters. In the future many different shapes can be investigated and complete configurations will be the subject of multidisciplinary design optimization (MDO) procedures. This will enlarge the aerodynamic knowledge base and will lead to better configurations. Current computer-aided design software (CAD) is not designed for such purpose: These programs usually do not define the shape of an aircraft in a way suitable for aerodynamic design and automatic optimization.

A different approach for generating aircraft geometries was presented by H. Sobieczky (see [8], [9], [10] and [11]). The surface of an aircraft will be calculated from characteristic two-dimensional curves, such as the crownlines of the fuselage, the twist axis or leading and trailing edges of the wing planform. The parameters for the characteristic curves have a direct geometric meaning, expressing a specific property of the aircraft. For different wing or fuselage types different algorithms are provided for the geometry definition. This keeps the total number of parameters small. The different algorithms for the surface definition enhance the knowledge base for aerodynamic design.

The geometry generator has already proven to be a good concept for modelling several configurations in the whole range of flight Mach numbers (for an example see figure 1). But until now the software had no visual design support. The geometric parameters were calculated from three-view drawings and entered in an input file. This led to a long learning time for new users. Lately the generator was redesigned to allow for interactive parameter definition using an object-oriented approach[1].

Figure 1: Generic HSCT configuration shape defined with the geometry generator [10]

In the new tool geometry definitions are handled as objects (so called components). Each part of an airplane and each function needed to define those parts will become a single object. The airplane configuration is designed by

1. Asides casual CAD programs there are few interactive design tools specialized for aircraft configurations. Some are presented in [1], [4] and [7]. For the described methods a simple interactive curve editor was illustrated in [6]. None of these systems is designed using an object-oriented paradigm.

building a tree of objects, where dependent objects are directly connected. This enlarges flexibility, since the structure of the tree is not restricted to only a few types and new objects can easily be added to the system.

The shape of each component is defined by a small number of parameters. Those parameters needed as input for the different components can be changed successively and interactively starting from a generic type. Direct geometric control allows for fast decisions depending on whether a parameter change has the desired effect.

The program was written in the Java programming language ([3]) which enables object oriented features, dynamic linking, and the use of internet and browser technologies. It combines the object-oriented paradigm with geometric algorithms and interactive control.

Due to a general approach high flexibility has been reached for the new tool. This resulted in many extensions to the geometry generator which is no longer restricted to airframe design but may be used in other fields, where few parameters define a geometric shape. As a consequence of the extensions, the parameter files are not backward compatible to the original ones, but still this tool may be used as a graphic editor generating the parameters to enter into the original input files for the original programs.

Characteristic Curves

The geometry representation differs from usual CAD-programs. Surfaces are calculated from characteristic curves. Those curves can prescribe the crownlines, contour lines, distributions or something similar. The user enters those curves as two dimensional functions. The functions can be controlled by a small number of parameters, where the parameters define geometrical properties, such as the slope or the curvature at certain points.

The characteristic curves are composed piecewise of simple functions, which are called base functions. There is a set of about fifteen base functions already available, but new functions may be added whenever a new type would describe a behaviour more precisely. The set of functions includes polynomials, trigonometric functions and conics (see figure 2). All base functions define graphs on a unit square. Each base function is determined by at most four parameters. For most functions two of the parameters are given from the slopes at the interval start and end.

From these functions the designer chooses the one that is best suited to describe a part of the characteristic curve. Therefore the graph will be mapped from the unit square onto an interval of the characteristic curve as shown in figure 3. The parameters needed to control the base functions will be entered according to the coordinates of the characteristic curve. An interval of the characteristic curve is determined by the interval start and end points (supports

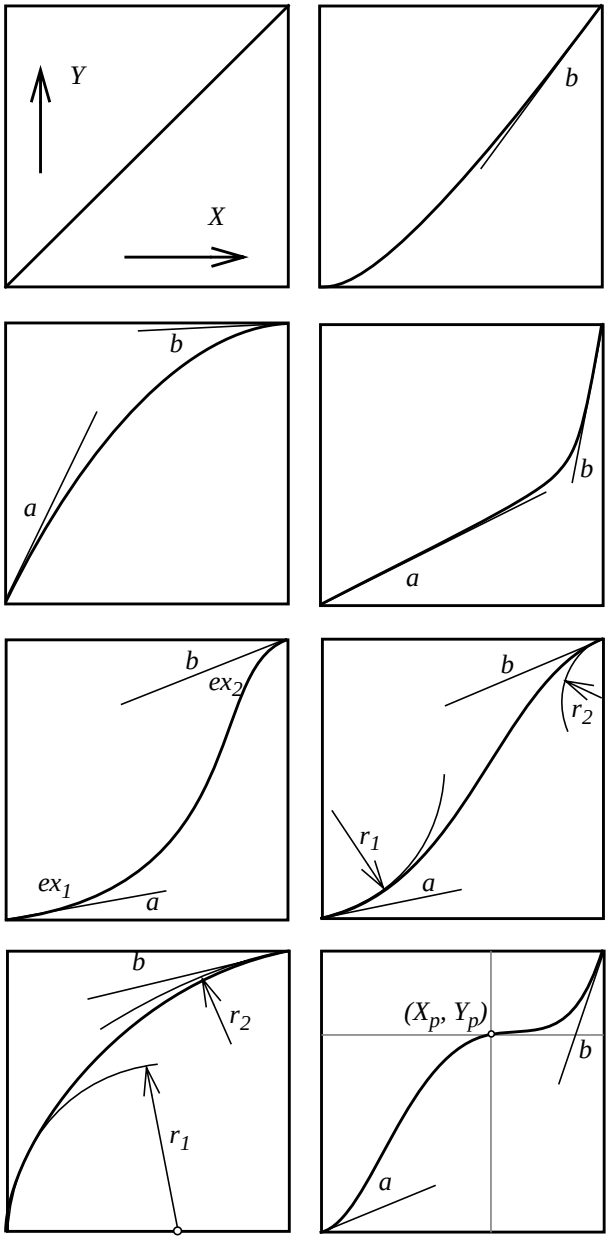

Figure 2: Selection of eight base functions Y(X) within the unit square using up to four parameters.

(x_i,y_i) and (x_{i+1},y_{i+1})), an index for the chosen base function and the four parameters for the base function control. Continuity in slope (and curvature) may or may not be needed at certain support points: To guarantee continuity in the derivatives, the slopes (and curvatures) at the boundaries of an interval can be set equal. By restricting to a subset of functions, curvature continuity can be enforced in a similar way. The interactive design tool en-

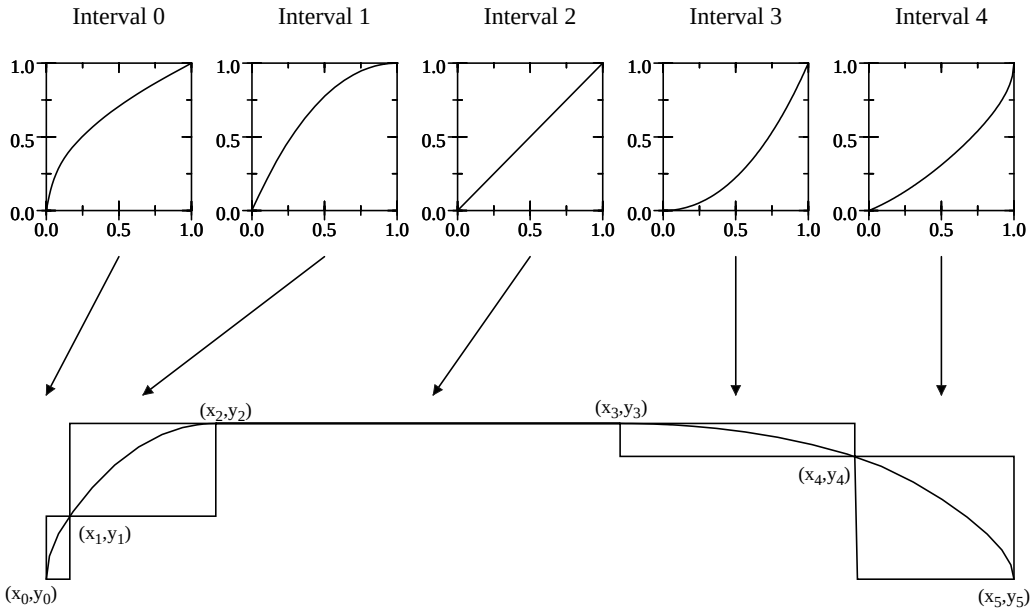

Figure 3:
Base functions mapped on the intervals of a characteristic curve. Due to the mapping the unit squares of the base functions will be stretched to fit in the boxes given by the intervals start and end points. For smooth curves the slopes (and curvatures) at the interval borders have to correspond.

ables to firmly define these conditions. Changing one interval will automatically update the neighbouring intervals. This keeps the total number of parameters that have to be entered small.

There is no limit for the number of base functions used to model a characteristic curve. Simple characteristics may be defined from a single base function while complex properties could be composed from an arbitrary number of single functions.

Surface representation

Depending on the configuration different algorithms for the surface definition, and therefore a varying number of characteristic functions, are used for generating the various parts. A simple shape of such a kind are surfaces of revolution, that can be used to form simple fuselages. For such axisymmetric fuselages the radius will be defined along the axis. For defining a fuselage with elliptical body sections one would add a second function defining the second axis of an ellipse. In this way configurations of increasing complexity may be built. Figure 4 shows a fuselage definition where additional parameters allow even for higher flexibility.

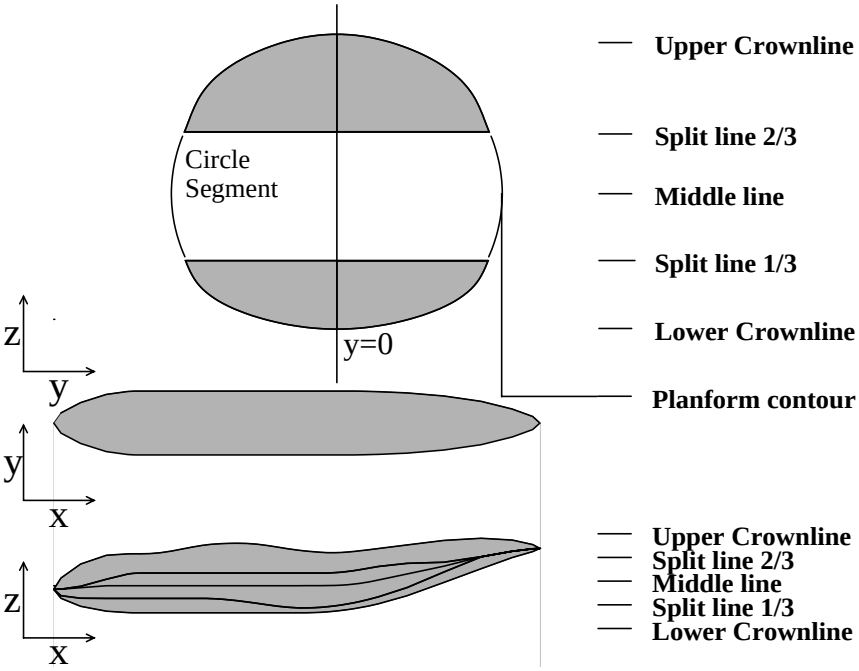

Figure 4:
Three cutting planes for a generic high wing transport aircraft. For the generation of a fuselage the parameters for cross sections will be specified. In this example the cross sections are divided into three sections. The splitting lines for those sections are given as parameters. Additional parameters specify the slope at the junctions. The cutting planes through the xy- and the xz-plane show the change of the sectional parameter along the body axis. The configuration can be evaluated at any number of crossections.

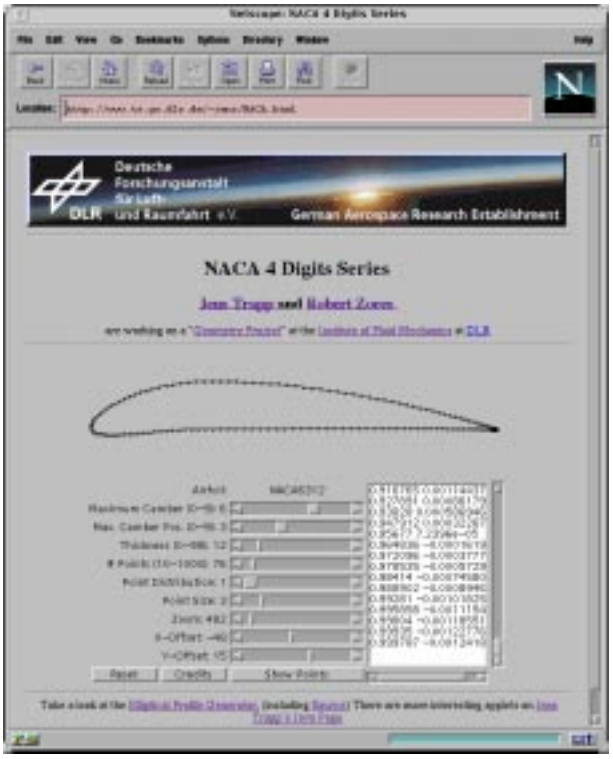

Figure 5: The NACA 4 Digit Series is an illustrative example for parametric definition of airfoil shapes. It is defined by three parameters defining the thickness, the maximum camber and the position of the maximum camber. The thickness is given in hundredths of the chord length while the camber parameters are defined in tenths of chord length (see [5]). For the geometry generator, the definition is extended to allow for non-integer values of those parameters.

In general, surfaces of fuselage type are defined from cross sections, where characteristic curves define the change of the cross section parameters along the body axis. For surfaces of wing type the characteristic curves are defined along the twist axis and they describe the change of the wing section parameters. In this way they define pitch, taper and twist of the wing. The sections themselves are defined either by analytical methods (e.g. NACA 4 Digit Series, see figure 5) or by a list of two dimensional point data prescribed by choice of special airfoils.

All surface types are parametric functions, mapping a two-dimensional area into three-dimensional space ([2]). For further data processing the partial derivatives for the unknowns will be calculated. Therefore the derivatives for the section definitions are calculated first, giving one of the directions. For airfoil sections given from a set of points the derivatives are calculated numerically. The second direction will be calculated from the change in the section settings and from the mapping of sections onto the surface. From the partial derivatives the surface normals may be calculated. For some algorithms also the second order partial derivatives need to be calculated allowing for an analytic analysis of the curvature of the surface.

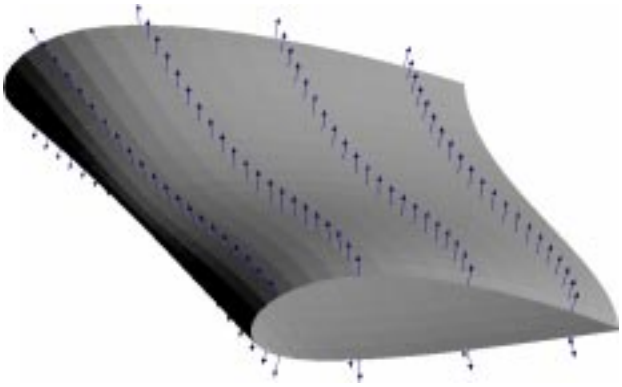

Figure 6: Analytical normals on wing calculated from the partial derivatives.

The partial derivatives for example are needed, whenever search algorithms for certain points on the surface are used. This, for example, is used for surface-surface-intersection algorithms where the intersection curve will be determined by a Newton-algorithm.

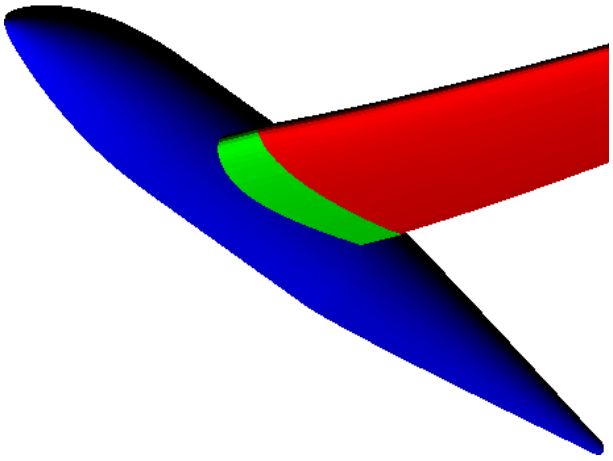

Figure 7: Junction region showing tangential continuation of the wing. The junction is generated as a seperate part.

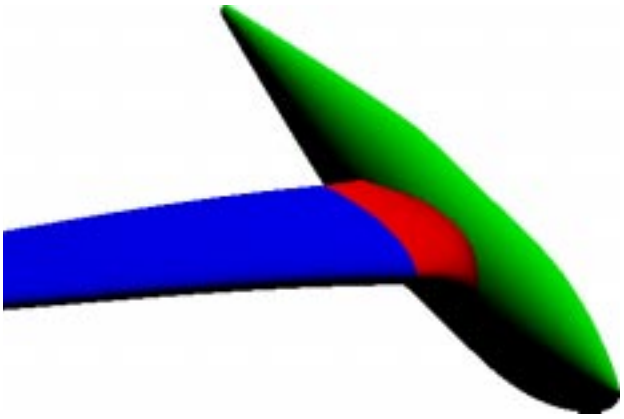

Figure 8: Smooth body-wing junction calculated with ar offset from the intersection curve. The offset is given as function depending on the wing chord. This allows fo the definition of a wide fillet at the wing leading edge and at the same time a vanishing fillet at the trailing edge.

The generator includes several algorithms for the connection of the wing and the body part based on similar techniques. In order to perform this, body and wing are defined leaving a gap inbetween, which is filled by the junction part. Figure 7 shows a junction as the tangential continuition of the wing for a simple wing-body configuration. The connecting part is defined as the linear combination of the inner wing section and a curve that is defined by the intersections of the spanwise wing tangents on the body. Figure 8 is an example for a smooth junction. The parameters for the calculation of the body-wing junction define the connecting curve on the body as the distance from the intersection curve of the tangential continuition of the wing with the body.

Object-Oriented Geometrical Design

The algorithms presented for the geometric definition are suited well for the design of aircraft. Only relatively few parameters will control the shape of the airframe. The meaning of the parameters are adapted to the geometry that will be calculated. This enables an intuitive usage of the program. To understand and use the tool a designer does not have to be a CAD-expert.

The described algorithms have already been used in a family of geometry generators written in the FORTRAN programming language. Due to the lack of flexibility in this language several versions of the program have been written for new applications using the same underlying routines. Separate programs are difficult to maintain and algorithms developed in the different programs have not been combined. Also, our newly developed algorithms for new options of this geometry tool would not fit into the fixed structure of the original FORTRAN program. For those reasons the generator was redesigned using object-oriented concepts. Besides solving the problems mentioned above, the redesign should include some further improvements, such as an interactive user interface for parameter input and control. The Java language ([3]) supports all these desired features and was used to implement the new tool.

For the redesign a global approach was chosen which does not restrict the tool's usage to aircraft design. It may be used as the user interface for any kind of graphical object. A graphical object (also called component) in this sense has a limited number of parameters from which point data will be calculated. A simple example for such a component is an airfoil of the NACA 4-Digit-Series, which is generated from four digits, defining the camber and thickness (see figure 5).

The types of parameters are not restricted to integer or float numbers. They can also be given in form of other graphic components or even lists of components. Therefore, objects can be nested or combined to form complex geometric configurations. A whole configuration can be presented as a tree of objects where the resulting point data will be passed from the leafs to the root object (see figure 9).

The design tool always has a root object, which holds all the active components, takes their point information and will render them on the canvas. The canvas size and the view are defined as parameters of the root object and may be changed in the same way as other parameters may. There are many different types of components (so called classes) that are already available for the tool, ranging from simple ellipse or text components up to complex three dimensional surface definitions used for arbitrary airframe design.

Each geometric algorithm will become a new class. For the definition of the geometry classes inheritance is used. A hierarchy of classes is generated from generalization and specification. This allows to derive new shapes from given classes. Using this mechanism only a small amount of code has to be added when implementing the new algorithm. When developing airframes, we mainly deal with parametric surfaces. This generalized definition is implemented in a class. It supports the common services for all surface algorithms used. The wing, body and junction algorithms are derived from this class. These classes are further specialized offering different ways in which the surfaces are defined.

Due to classification similar parts can be handled in the same way and parts that have a common generalized class can be exchanged against each other.

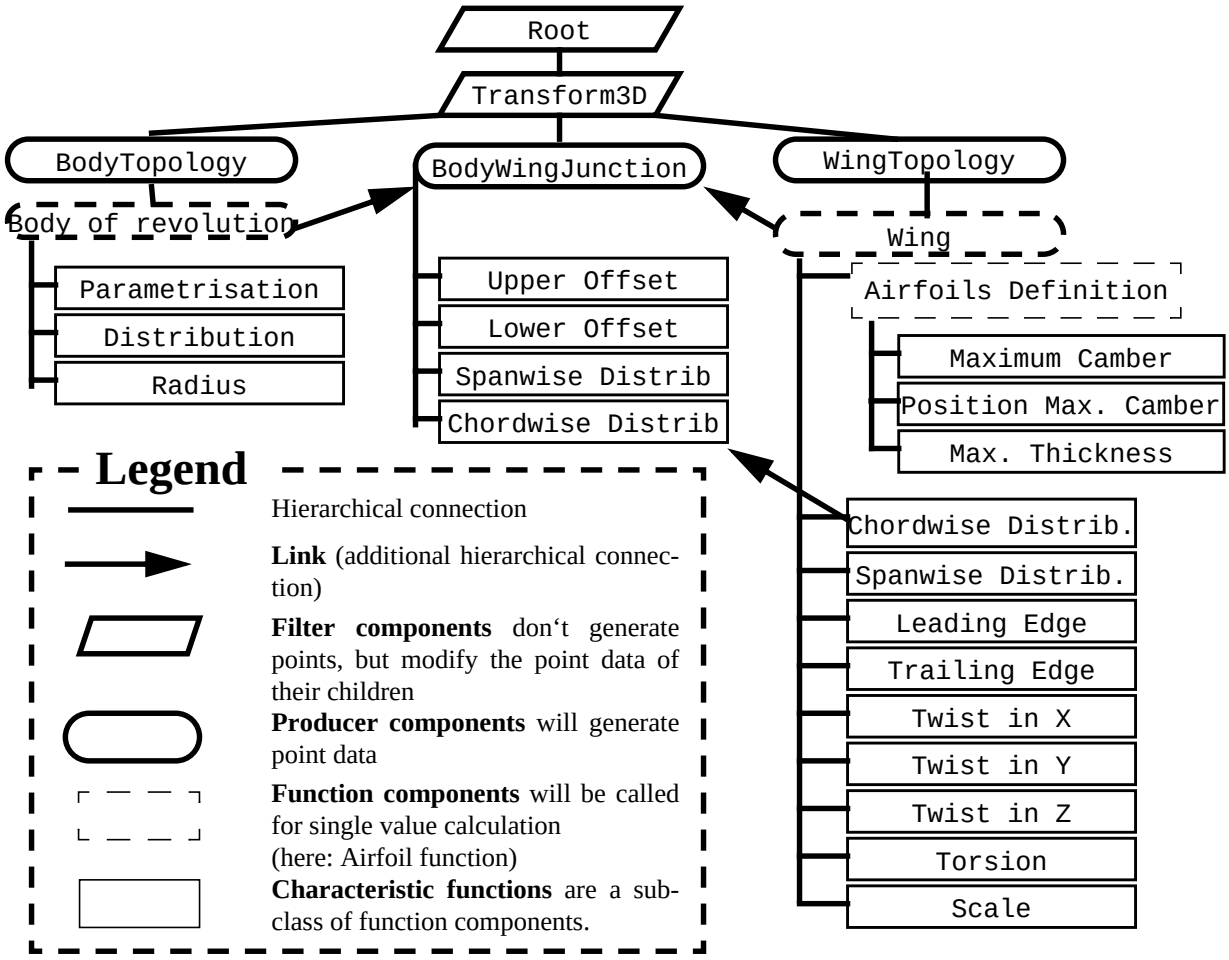

Figure 9: Object hierarchy for the design tool for a simple body wing configuration with a smooth juncture between fuselage and wing as shown in figure . The root object will display the point data, that will be evaluated from the producing objects. The functions used for calculating the geometry can readily be exchanged by new objects. The airplane parts contain the characteristic curves as additional parameters. The base functions that act as parameters to the characteristic functions are not shown in this graph.

Parameters

All parameters needed as input for the algorithm of the component are inserted in a list. They are attached to their names by which they can be easily accessed. The names appear in the user interface and may contain any character including whitespaces[2].

Flexibility was one of the major goals. Therefore the amount of work needed to extend the program for new shapes was minimized. To link a new algorithm to the system only a few rules have to be kept in mind. The parameters used as input for the algorithms need to be selected from a set of predefined types. All parameters must be added to a list. From that list all methods needed for the program control and the in- and output are obtained automatically. An interactive graphical user interface (GUI) is generated on the fly. When parameters are changed in the user interface, the geometry will be updated immediately.

This automatic behaviour shortens the time needed for the creation of new configurations and will make the generator easy to use.

Through the list of parameters the infrastructural tasks, needed to load, store or update the parameters, are solved without any further work inside the classes that define geometric algorithms. This makes the system easy to extend to future applications. New classes can be derived from existing classes by adding more parameters or by redefining the algorithm. For adding new object types to the tool, one only has to specify the list of parameters, their types and meaning and finally to implement the calculation algorithm of the routine.

2. Names must not contain the string ':='.

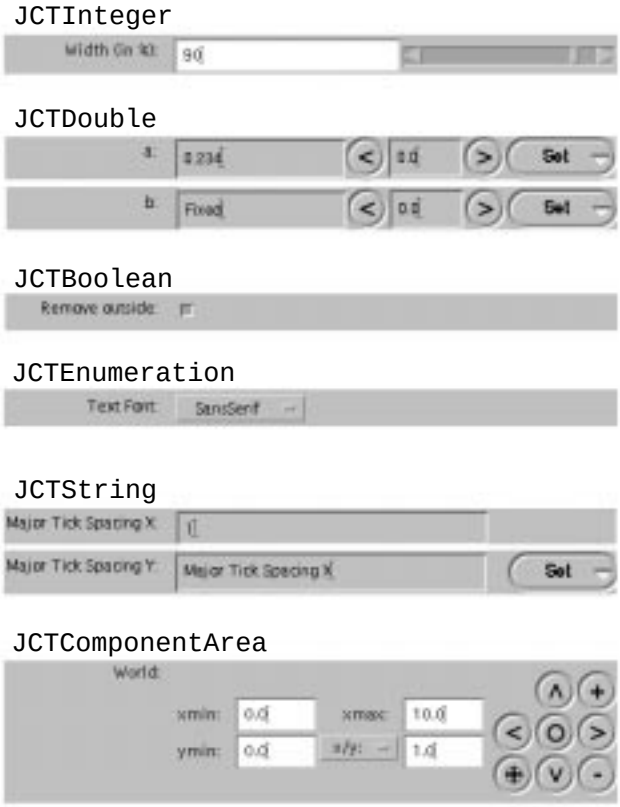

Figure 10: Widgets for the primitive parameter types. For unique handling of the parameter types, all primitives are encapsulated in wrapper classes that inherit common services from the root parameter class ***JCTParameter****. The parameter name appears in front of the widget. Integer values may be changed by using a slider. Constant expressions can be used with most types. The constant names will be given as alternative using the set-menu button (e.g.* ***JCTDouble*** *or* ***JCTString****).* ***JCTComponentArea*** *defines a more complex widget. It is used for selection of world coordinates of the canvas.*

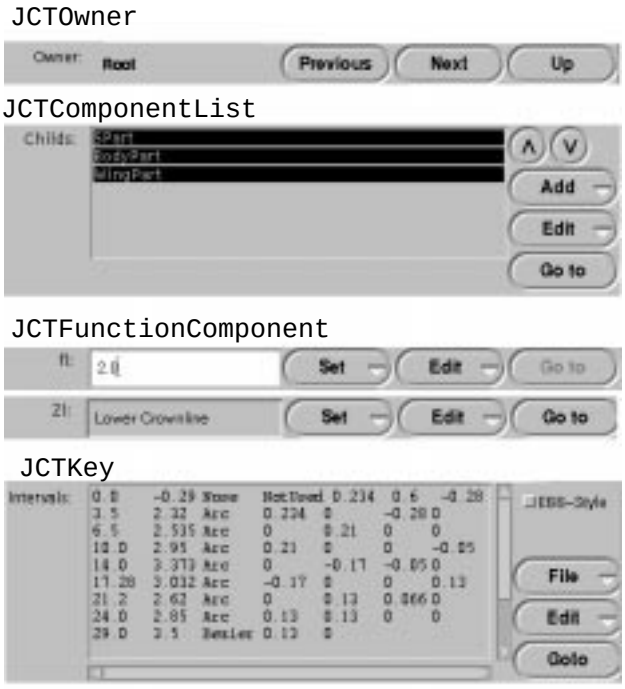

Figure 11: Widgets representing parameters that contain components. ***JCTOwner*** *links to the owner in the hierarchy of components. The add- and set-buttons are used for including objects from files, while the edit button offers cut'n'paste-functionality. When pressing the goto-button the user interfaces switches to the card of the selected component. For function values, either a constant value or a function component may be selected. The* ***JCTKey*** *parameter is used to represent the characteristic curves as a list of functions.*

User Interface

The user interface for the generator will be generated automatically on the fly from the list of parameters. Therefore, for every type of parameter a window component is definied (see figures 10 and 11). The widget and the parameter communicate through set and get methods defined inside the widget class. Changes inside the user interface will invoke the get-routine of the widget which calls set-routines on the parameter. Changes in the parameter will invoke the update of all depending components.

Each parameter has additional attributes that control their behaviour. A parameter can be defined as constant and will then stay unchangable by the user. Parameters may be invisible and will then not be shown in the GUI. If the parameter is insignificant, a change of the parameter will not cause any update of the point data. Transient parameters will not be stored in a file and for global parameters the same value is used for all components of the same type.

All widgets needed to represent the list for a single object will be shown in a single card. Only one card is visible at a time. Figure 12 shows a card for a base function. Because the geometric objects contain each other, they are included as parameters in the list. Aside to all child objects that are needed the owner object is stored as parameter as well. The widgets for those parameters allow changing to a card of any directly connected geometric component.

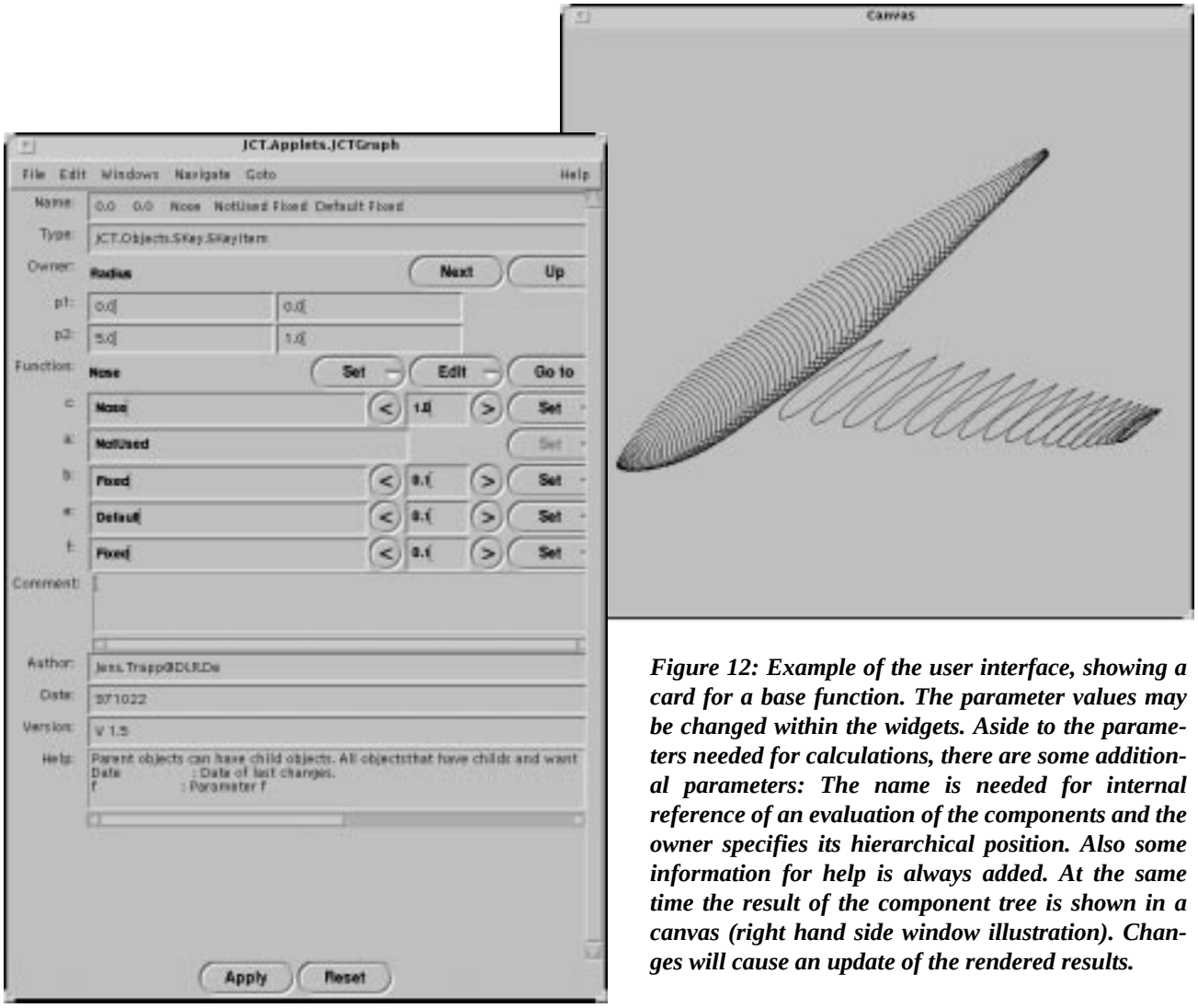

Figure 12: Example of the user interface, showing a card for a base function. The parameter values may be changed within the widgets. Aside to the parameters needed for calculations, there are some additional parameters: The name is needed for internal reference of an evaluation of the components and the owner specifies its hierarchical position. Also some information for help is always added. At the same time the result of the component tree is shown in a canvas (right hand side window illustration). Changes will cause an update of the rendered results.

All component parameters have predefined values. New shapes can easily be created by derivation from the standard settings. Also it is possible in the new design tool to set the parameters to named constants or to define geometric properties rather than entering their numerical values. This will reduce the number of free parameters and will provide better control. For backward compatibility all curves may be printed in the old-style list input format to be interpreted by the original geometry generator. Figure 12 shows an illustration of the present user interface.

When designing an airplane, the body, wing and junction are specified as components. Those components define the topology of each part and will contain the characteristic curves as additional parameters. Those curves need base functions as parameters. The base functions will have their own parameters again. The whole configuration will be defined in that way by a tree of components as shown in figure 9.

Some classes may have special geometric properties which are requested for the combination of components. For example, components of a function type always must specify a method on how to calculate a single function value. For the surface components special functions must be provided to evaluate single surface points or to calculate the partial equations. Those special properties fit in the object-oriented concept through the inheritance mechanism. All classes that are derived from a main function class or from a class for parametric surfaces will inherit those properties and must implement the appropriate methods. Components may accept only certain types and will then rely on those features.

Because the design of the tool is using a general approach, the geometric algorithms described in the first part of this paper may easily be intermixed with totally different geometrical components. For example these may be used to visually compare a characteristic curve with a set of points

originating from any other source. The key functions may be substituted by a different kind of function, whenever the exchanged function would reduce the number of parameters to enter. For this purpose a parser for functions is included.

Conclusion

The new design tool combines the features of a modern, platform independent, interactive geometry editor with flexible shape algorithms. Geometries may be calculated explicitly, non-iteratively and with analytical accuracy by defining a problem-specific number of parameters. Different algorithms are provided to offer different levels for complexity and flexibility of surface definitions.

A generalized approach makes the program useful for many purposes and new shape algorithms easily may be added. The direct graphic control helps to define new shapes, and due to a comfortable user interface the program is easy to learn and easy to use.

The parametric algorithms used for the geometric definition are very well suited for automatic optimization. The tool offers the opportunity to change the shape of a complete aircraft by changing a single parameter. With this ability the design tool makes geometry definition available for future design processing using multidisplinary optimization.

Future Goals

Since Java is a very young language there are still many things that are not state of the art. Especially the speed of Java programs will be enhanced with future versions of the language. This will decrease processing time rapidly and will enable more complex configurations being edited interactively with the design tool. When starting the project of interactive geometry design, there was no (hardware accelerated) graphics library for Java available. One major topic for the near future must be the interaction of the program with the new Java 3D API. The high quality graphics will support the design process, because the effect of bad parameter settings will become visible immediately.

The geometry generator is used as a predesign tool for other programs such as grid-generators for flow solvers or CAD-programs. A simple IGES-interface is used at present to convert the data for further work. The interface uses NURBS-Surfaces for an interpolation of the surface. No error estimation is performed for the time being, but for better quality the algorithm needs to adapt the number and location of surface points used for the interpolation.

The program is designed to be extendible. New geometry shapes will be added as soon as they will become focus of our research. Also there are some geometry definitions in the old FORTRAN sources that have not yet been ported to Java. New configurations such as the large scale aircraft shown in figure 13 are future goals and will then be available for further aerodynamical and aeroelastic investigations.

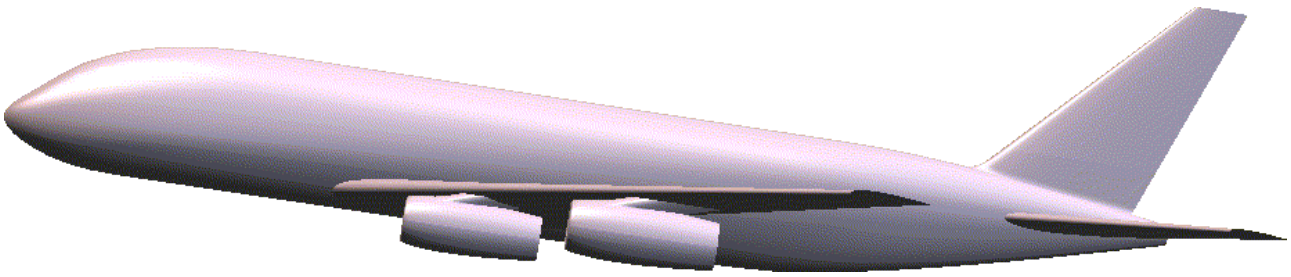

Figure 13: Design of a large scaled commercial aircraft

References

[1] Capron, W.K., Smit, K.: *Advanced Aerodynamic Applications of an Interactive Geometry and Visualization System*. AIAA 91-0800, USA (1991).

[2] Farin, G.: *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, London, 2. Edition (1990).

[3] Flanagan, D.: *Java in a Nutshell*. O'Reilly Associates, USA, 2. Edition (1997).

[4] LaBozetta, W. F., Cole, P. E., Born, K. E. : *Interactive Graphics for Geometry Generation - A Program with a Contemporary Design*. AIAA-84-2389, USA (1984).

[5] Ladson, C. L., Brooks, Jr., W.: *Development of a Computer Program to Obtain Ordinates for NACA 4-Digit, 4-Digit Modified 5-Digit, and 16-Series Airfoils*, NASA TM X-3284 (1975)

[6] Pagendarm, H.-G., Laurien, E., Sobieczky, H.: *Interactive Geometry Definition and Grid Generation for Applied. Aerodynamics.* AIAA 88-2415CP, USA (1988).

[7] Snepp, D. K., Pomeroy, R. C.: *A Geometry System for Aerodynamic Design*. AIAA 87-2902, USA (1987).

[8] Sobieczky, H.: *Geometry Generation for Transonic Design*. in: Habashi, W. G., (Ed.): *Recent Advances in Numerical Methods in Fluids*, Volume 4, pp.163-182. Pineridge Press: Swansea (1985)

[9] Sobieczky, H.: *Aircraft Surface Generation.* in: N. Weatherill et al (Eds.): *Results of EC Brite/ Euram Project 'Euromesh' 1990-92, Notes on Numerical Fluid Mechanics*, Vol. 44, pp.71-76, Vieweg: Braunschweig (1993)

[10] Sobieczky, H.: *Geometry Generator for CFD and Applied Aerodynamics*. In: Sobieczky, H. (Ed.), *New Design Concepts for High Speed Air Transport*. *CISM Courses and Lectures* Vol. 366, pp. 137-158, Springer: Wien, NewYork (1997)

[11] Trapp, J. C., Zores, R., Gerhold, T., Sobieczky, H.: *Geometrische Werkzeuge zur Auslegung Adaptiver Aerodynamischer Komponenten*. Tagungsband der DGLR Fachausschußsitzung T2.2 Experimentelle Aerodynamik, Deutsche Gesellschaft für Luft- und Raumfahrt (1996)