Last updated: Yeh-Liang Hsu (2010-09-12).

Note: This is the course material for "ME550 Geometric modeling and computer graphics," Yuan Ze University. Part of this material is adapted from *CAD/CAM Theory and Practice*, by Ibrahim Zeid, McGraw-Hill, 1991. This material is be used strictly for teaching and learning of this course.

CAD systems -- basic concepts

1. CAD systems

Before the 1980s, engineering design facilities often consisted of rows of drafting tables, with a designer hunched over a drawing on each table. Engineering students were required to take several years of engineering drafting courses in university, in which they spent endless hours practicing lettering and drawing perfect circles.

The introduction of computer revolutionized engineering graphics. **The first computer-based drafting tool was SKETCHPAD developed at MIT in 1963 by Ivan Sutherland** and was soon commercialized. By the 1980s **computer-aided drafting** became a standard tool in industry. **Autodesk launched AutoCAD in 1982**, which was the first commercially successful 2D drafting software. The purpose was to replace paper-pencil drawings with a system that is more automated, efficient and accurate. The principal output of a 2D CAD software are the drawings themselves, rather than a model from which drawings can be extracted. **Solid modeling** technology came about in the late 1980s, **the introduction of Pro/E in 1988 and SolidWorks in 1995**, and remains the state-of-the-art technology.

The use of computer-aided design (CAD) tools has had a great impact on engineering design. In engineering practice, CAD has been utilized in different ways by different people. Some utilize it to **produce drawings and document designs.** Others may employ it as **a visual tool by generating shaded images and animated displays.** A third group may perform **engineering analysis**, for example, stress analysis to verify the strength of the design, interface checking to detect collision between components in an assembly, and

kinematic analysis to check whether the design will provide the required motion. A fourth group may use it to perform **process planning** (to establish which processes and the proper parameters for the processes are to be used) **and generate NC part programs.**

It is obvious that **computer aided design starts with the description of the geometry of the machine or machine parts** in a complete and unambiguous manner by way of a set of computer-stored information that can be further utilized for analysis and design. **Geometric modeling deals with the mathematical representation of curves, surfaces, and solids necessary in the definition of complex physical or engineering objects.**

◇Problem 1.

What is the CAD software you choose for this course? Among the 4 aspects of computer-aided design discussed above, what does your CAD software do? Give a brief description. ◇

A user interface is defined as a collection of commands that users can use to interact with a particular CAD system. User-interface or man-machine dialogue represents the only means of communication between users and CAD software. The human-computer interface for most systems involves extensive graphics, the so-called **"graphical user interface (GUI)"**, regardless of the application. Typically, general systems now consist of windows, pull-down, pop-up, or hierarchical menus, icons, and pointing devices, such as a mouse or spaceball, for positioning the screen cursor.

◇Problem 2.

Introduce the basic user interface of your CAD software (multiple windows, menus, and icons). Describe the different ways of input (by typing in commands through keyboard, clicking on pull-down menus or icons using a mouse, drawing on a tablet etc.). ◇

2. Coordinate systems

Three types of coordinate systems are needed in order to input, store, and display model geometry and graphics. These are **the model coordinate system (MCS), the working coordinate system (WCS), and the screen coordinate system (SCS).**

2.1 Model coordinate system

The model coordinate system is defined as the reference space of the model with respect to which all the model geometrical data is stored. The MCS is the only coordinate system that the software recognizes when storing or retrieving geometrical information in or from a model database. Many existing software packages allow the user to input coordinate information in Cartesian (*x*, *y*, *z*), cylindrical (r,θ,z), and/or spherical (r,θ,ϕ) systems. However, this **input information is transformed to (x, y, z), coordinates relative to the MCS before being stored in the database.**

2.2 Working coordinate system

It is often convenient in the development of geometric models and the input of geometrical data to refer to an auxiliary coordinate system instead of the MCS. This is usually useful **when a desired plane (face) of construction is not easily defined as one of the MCS orthogonal planes**, as in the case of inclined faces of a model shown in Figure 1. **The user can define a Cartesian coordinate system whose *x*-*y* plane is coincident with the desired plane of construction.** That system is the working coordinate system (WCS). It is a convenient user-defined system that facilitates geometric construction. It can be established at any position and orientation in space that the user desires.

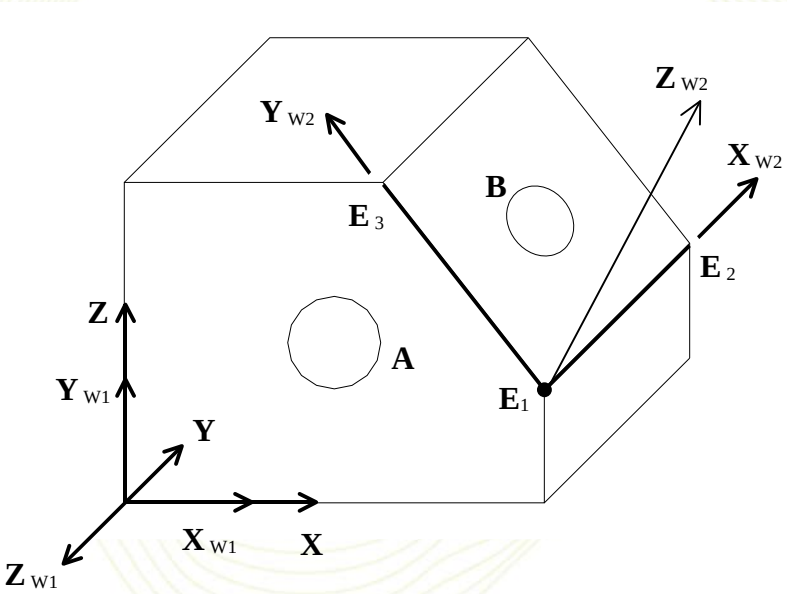


Figure 1. WCSs required to construct through holes A and B

While the user can input data in reference to the WCS, the software performs the necessary transformations to the MCS before storing the data, using the following equation:

$$\mathbf{P} = [T]\mathbf{P}_w \tag{1}$$

where $\mathbf{P}$ is the position vector of a point relative to the MCS and $\mathbf{P}_w$ is the vector of a point relative to the active WCS. Each vector is given by

$$\mathbf{P} = [x, y, z, 1]^T \tag{2}$$

The matrix $[T]$ is the homogeneous transformation matrix. It is a 4×4 matrix and is given by

$$[\mathbf{T}] = \left[\begin{array}{ccc:c} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc:c} & {}^{M}_{W}[\mathbf{R}] & & {}^{M}\mathbf{P}_{W.org} \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{3}$$

where ${}^{M}_{W}[\mathbf{R}]$ is the rotation matrix that defines the orientation of the WCS relative to the MCS and ${}^{M}\mathbf{P}_{W.org}$ is the position vector that describes the origin of the WCS relative to the MCS. These transformation will be discussed in details in later sections.

◇Problem 3.

Create a simple geometric model similar to that in Figure 1 in your CAD software. Demonstrate how you can create several coordinate systems and switch from one coordinate system to another in your CAD software. How do you get the coordinate of the center of through hole **B** in MCS in your CAD software? ◇

2.3 Screen coordinate system

The SCS is defined as two-dimensional, device-dependent coordinate system, whose origin is usually located at the lower left corner of the graphics display, as shown in Figure 2. The physical dimensions of a device screen (aspect ratio) and the type of device (vector or raster) determine the range and the measurement unit of the SCS. For raster graphics displays, the pixel grid serves as the SCS. A 1024×768 display has an SCS with a range of (0, 0) to (1024, 768).

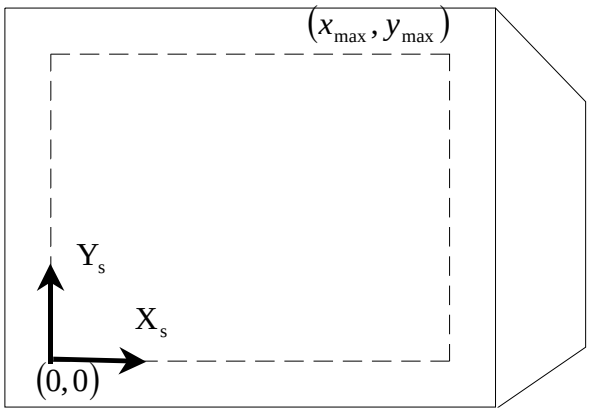


Figure 2. Typical SCS

3. Transformation

Geometric transformations play a central role in model construction and viewing. They are used in modeling to express locations of objects relative to others. In generating a view of an object, they are used to achieve the effect of different viewing positions and directions. Typical CAD construction commands to **translate, rotate, zooms, and mirror** entities are all based on geometric transformations.

3.1 Transformations of geometric models

The simplest motion is the **rigid-body motion in which the relative distances between object particles remain constant**; that is, **the object does not deform during the motion**. Geometric transformations that describe this motion are often referred to as rigid-body transformations and typically include translation, reflection, rotation, and any combination of them.

Displaying and/or transforming a given entity require the transformation of its key points first. Transformation of a point represents the core problem in geometric trans-formation because it is the basic element of object representation. For example, a line is represented by its two endpoints, and a general curve, surface, or solid is represented by a collection of points. The problem of transforming a point can be stated as follows. Given a point **P** that belongs to a geometric model that undergoes a rigid-body motion, find the corresponding point **P*** in the new position such that

$$\mathbf{P}^* = f(\mathbf{P}, \text{transformation parameters}) \qquad (4)$$

That is, the new position vector **P*** should be expressed in terms of the old position vector **P** and the motion parameters.

Geometric transformation should be unique. A given set of transformation parameters must yield one and only one new point for each old point. Another characteristic is **the concatenation, or combination, of transformations**. Intuitively, two transformations can be concatenated to yield a single transformation, which should have the same effect as the sequential application of the original two.

It is desirable to express it in terms of matrix notation as

$$\mathbf{P}^* = [\mathbf{T}]\mathbf{P} \tag{5}$$

where $[\mathbf{T}]$ is the transformation matrix.

Applying Equation (5) repeatedly to key points in a geometric model database or a particular entity enables the transformation of the model or the entity. For example, to transform a straight line, its two endpoints are transformed and then connected to produce the transformed line. Similarly, to transform a curve, points on the curve are generated utilizing its parametric equation, transformed, and then connected to give the transformed curve.

A coordinate transformation of the form

$$\begin{aligned} x' &= a_{xx}x + a_{xy}y + b_x \\ y' &= a_{yx}x + a_{yy}y + b_y \end{aligned} \tag{6}$$

is called a **two-dimensional affine transformation. Each of the transformed coordinates x' and y' is a linear function of the original coordinates x and y**, and parameters a_{ij} and b_k are constants determined by the transformation type. Affine transformations have the general properties that parallel lines are transformed into parallel lines, and finite points map to finite points.

Translation, rotation, scaling, reflection, and shear are examples of two-dimensional affine transformations. Any general two-dimensional affine transformation can always be expressed as a composition of these five transformations. **Affine transformation involving only rotation, translation, and reflection preserves angles and lengths, as well as parallel lines.**

3.2 Translation

When every entity of a geometric model remains parallel to its initial position, the rigid-body transformation of the model is defined as translation. Translating a model implies that every point on it moves an equal given distance in a given direction.

$$\mathbf{P}^* = \mathbf{P} + \mathbf{d} \tag{7}$$

$$\begin{aligned} x^* &= x + x_d \\ y^* &= y + y_d \\ z^* &= z + z_d \end{aligned} \tag{8}$$

3.3 Scaling

Scaling is used to change, increase or decrease, the size of an entity or a model. Pointwise scaling can be performed if the matrix [*T*] in Equation (5) is diagonal, that is,

$$\mathbf{P}^* = [S]\mathbf{P} \tag{9}$$

where [*S*] is a diagonal matrix. In three dimensions, it is given by

$$[S] = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \tag{10}$$

The elements s_x, s_y, and s_z of the scaling matrix [*S*] are the scaling factors in the *X*, *Y*, and *Z* directions respectively. Scaling factors are always positive (negative factors produce reflection). **If the scaling factors are smaller than 1, the geometric model or entity to which scaling is applied is compressed; if the factors are greater than 1, the model is stretched.** If the scale factors are equal, that is, $s_x=s_y=s_z=s$, the model changes in size only and not in shape; this is the case of **uniform scaling.**

Differential scaling occurs when $s_x \neq s_y \neq s_z$; that is, different scaling factors are applied in different directions. Differential scaling changes both the size and the shape of a geometric model or curve.

Uniform scaling is available on CAD/CAM systems in the form of a “zoom” command. The command requires users to input the scale factor *s* and digitize the entity or the view to be zoomed.

3.4 Reflection

Reflection (or mirror) transformation is useful in constructing symmetric models. If, for example, a model is symmetric with respect to a plane, then only half of its geometry is created which can be copied by reflection to generate the full model.

Reflection through the x=0, y=0, or z=0 plane can be achieved by negating the x, y, or z coordinate respectively. Thus, the reflection transformation can be expressed by the following equation:

$$\mathbf{P}^* = [M]\mathbf{P} \tag{11}$$

where [M] (mirror matrix) is a diagonal matrix with elements of ± 1, that is,

$$[M] = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & 0 \\ 0 & 0 & m_{33} \end{bmatrix} = \begin{bmatrix} \pm 1 & 0 & 0 \\ 0 & \pm 1 & 0 \\ 0 & 0 & \pm 1 \end{bmatrix} \tag{12}$$

For reflection through the x=0 plane, $m_{11} = -1$ and $m_{22} = m_{33} = 1$. Similarly, setting $m_{11} = m_{33} = 1$ and $m_{22} = -1$, or $m_{11} = m_{22} = 1$ and $m_{33} = -1$, produces reflection through the $y = 0$ or $z = 0$ plane respectively. Reflection through the x-axis requires $m_{11} = 1$ and $m_{22} = m_{33} = -1$, through the y-axis requires $m_{11} = m_{33} = -1$ and $m_{22} = 1$, and through the z-axis requires $m_1 = m_{22} = -1$ and $m_{33} = 1$. Selecting all the diagonal elements to be negative, that is, $m_{11} = m_{22} = m_{33} = -1$, produces reflection through the origin.

3.5 Rotation

Rotation is an important form of geometrical transformation. It enables users to view geometric models from different angles and also helps many geometric operations.

Rotation has a unique characteristic that is not shared by translation, scaling, or reflection－that is, **noncommutativeness**.

Consider the rotation of point **P** a positive angle θ about the z-axis.

$$\mathbf{P}^* = [R_Z]\mathbf{P} \tag{13}$$

$$[R_Z]=\begin{bmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix} \tag{14}$$

In general, **P***=[*R*]**P**, where

$$[R_X]=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta & -\sin\theta\\ 0 & \sin\theta & \cos\theta\end{bmatrix} \tag{15}$$

$$[R_Y]=\begin{bmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{bmatrix} \tag{16}$$

Compare the transformation Equations (7) to (16) with Equation (6), we can easily show that these are three-dimensional affine transformations.

◇Problem 4.

Create a rectangle in *x*-*y* plane; specify the coordinates of the 4 corner points. Assume some parameters and do the 4 transformation using your CAD software. Generate figures to illustrate these transformations. What are the names of the transformations in your CAD software?

Write a program in Matlab and use Equation (7) and (16) to do the same transformation to the 4 corner points. Redraw and connect these 4 corner points after transformation. Do the figures appear the same as generated by your CAD software? Show your Matlab program too. ◇

3.6 Shear

A transformation that distorts the shape of an object such that the transformed shape appears as if the object were composed of internal layers that had been cause to slide over each other is called a shear. Two common shearing transformations are those that shift coordinate *x* values and those that shift *y* vales.

An *x*-direction shear relative to the *x*-axis is produced with the transformation matrix

$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{17}$$

Similarly, a *y*-direction shear relative to the *y*-axis is produced with the transformation matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{18}$$

◇Problem 5.

Create a rectangle in *x*-*y* plane; specify the coordinates of the 4 corner points. Write a program in Matlab and use Equation (17) and (18) to transform the 4 corner points. Redraw and connect these 4 corner points after transformation. Does it appear to be a "shear"? Show your Matlab program too. ◇

4. Data structure

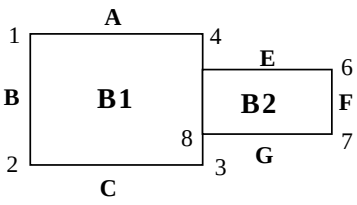
Formally **a data structure is defined as a set of data items or elements that are related to each other by a set of relations.** Applying these relations to the elements of the set results in a meaningful object. From a CAD point of view, a data structure is a scheme, logic, or a sequence of steps developed to achieve a certain graphics, non-graphics, and/or a programming goal.

Casually, a database is synonymous with the terms "files" and "collection of files". Formally, **a database is defined as an organized collection of graphics and nongraphics data stored on secondary storage in the computer.** A brief description of the popular database models is provided below.

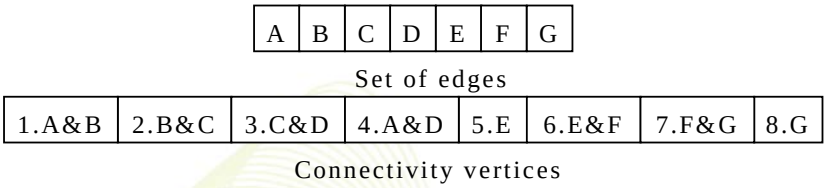
4.1 Relational database

In relational database, data is stored in tables, called relations, which are related to each other. The relations are stored in files that can be accessed sequentially or in a random access mode. Sequential access files are widely used. As an example, the relations needed to describe the object in Figure 4 are shown in Figure 5. The object is represented by the three relations **POINT, LINE/CURVE**, and **SURFACE**. A particular data structure shown in Figure 4 determines which relations are to be entered by the user and which are to be

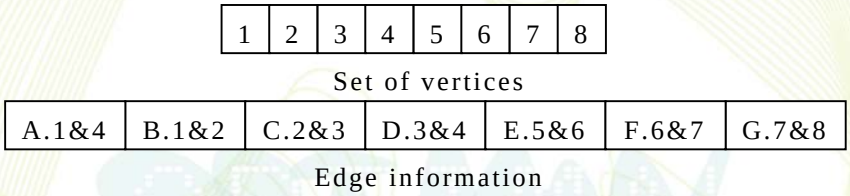
calculated automatically. One of the disadvantages of the relational database is that it requires substantial sorting, which might result in slowing the system response to user commands.



(a) Object



(b) Data structure based on edges



(c) Data structure based on vertices

(d) Data structure based on blocks

Figure 4. Various data structures of an object

Point	x	y
1	x_1	y_1
2	x_2	y_2
3	x_3	y_3
4	x_4	y_4
5	x_5	y_5
6	x_6	y_6
7	x_7	y_7
8	x_8	y_8

Line	Start Point	End point
A	1	4
B	1	2
C	2	3
D	3	4
E	5	6
F	6	7
G	7	8

Surface	Line/ Curve	Type
1	A	Line
	B	Line
	C	Line
	D	Line
2	E	Line
	F	Line
	G	Line
	D	Line

Figure 5. Sample relational database of object shown in Figure 4

4.2 Hierarchical database

In this model, data is represented by a tree structure. The top of the tree is usually known as the “root” and the superiority, or hierarchy, of the tree levels relative to each other descends from the root down. Figure 6 shows a hierarchical database of the object shown in Figure 4. Four levels are required to represent the object completely. One of the drawbacks of the hierarchical approach is the asymmetry of the tree structure, which forces database programmers to devote time and effort to solving problems, introduced by the hierarchical approach, which are not intrinsic to the object modeling itself.

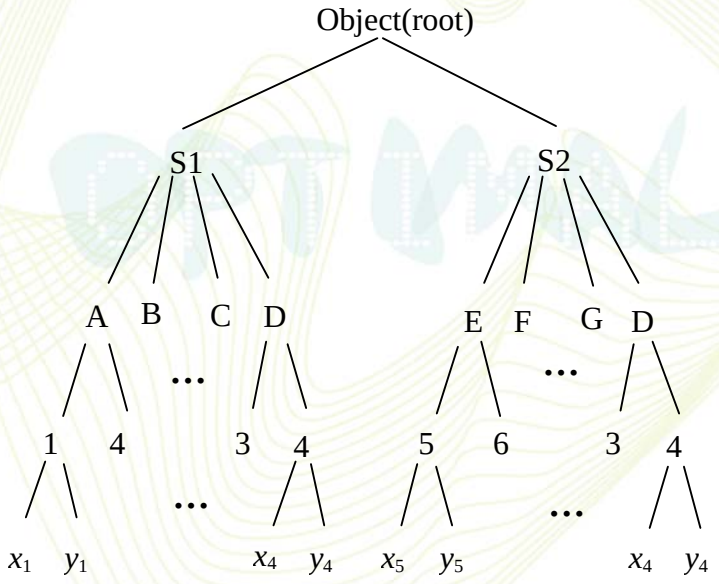


Figure 6. Sample hierarchical database of object shown in Figure 4

4.3 Network database

The network approach permits modeling of many-to-many correspondence more directly than the hierarchical approaches. Figure 7 shows a network database of the object shown in Figure 4. The prime disadvantage of the network approach is its undue complexity both in the database structure itself and in the associated programming of it.

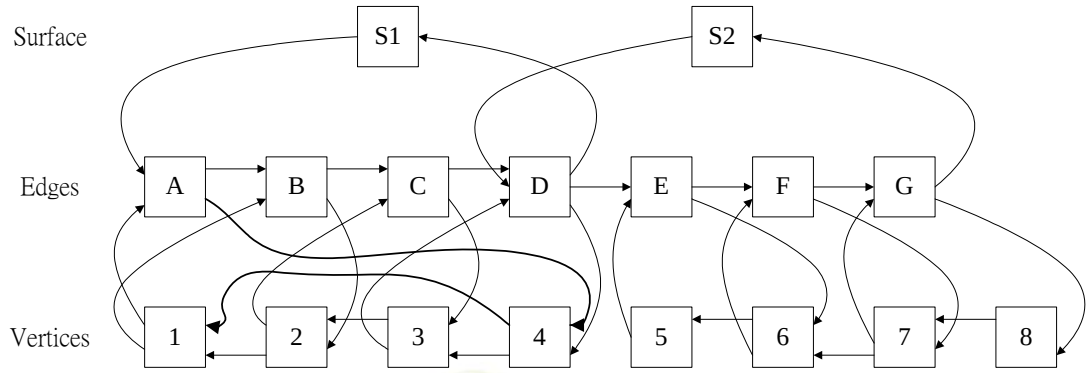


Figure 7. Sample network database of object shown in Figure 4

4.4 Object-oriented database

Unlike conventional database processing, **CAD/CAM applications require object-oriented accessing and manipulation; that is, units of retrieval and storage are design objects and not individual records in files.** These design objects also form the basis for **ensuring database integrity upon the insertion, deletion, or modification of component objects.** The object-oriented model should be able to capture all the relevant semantics of objects. This, in turn, results in a “rich”, well-integrated, and complete database readily accessible for applications. Object-oriented database models include the entity relationship model, complex object representation, molecular object representation, and abstract data model. The abstract data model is close to solid modeling databases. It employs abstract objects as primitives in the design of the database. Figure 8 shows an example of this database. Primitives are constructed from input data and form the lowest field or record of storage in the database.

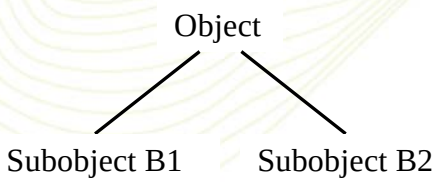


Figure 8. Sample object-oriented database of object shown in Figure 4

4.5 File transfer between CAD software

Sometimes designers need to import or export drawings created using one CAD software to another. Usually each CAD software has its own format, but translators are often available within most CAD software to translate the CAD file into standard formats. **DFX and DWG formats are used for 2D CAD drawings**. **Initial Graphics Exchange Specification (IGES) and Standard Exchange of Product (STEP) are two commonly used format for 3D solid modeling**.

However, **the "design intent" build into the model is often lost in the standard format**, which makes it difficult to edit the solid model after it is exported to another CAD software.

◇Problem 6.

Create a simple 3D object in your CAD software and export it as an IGES file. Import the file to another CAD software. Describe the process and show the model before and after file transfer. Is there anything missing in your object after this file transfer? ◇