I. Introduction to Interactive Graphical Programming

1. Background

Interactive graphical programming is characterized by:

- ◊ Processing both graphics and non-graphics data through graphical input/output devices and CAD/CAD software.
- ◊ Having a user-friendly interface to support user-computer dialogue.
- ◊ Customizing CAD/CAM systems to perform certain design tasks and functions that cannot be carried out directly by using the standard functions offered by the systems.
- ◊ Providing graphical interfaces to complex engineering analysis programs.

Some of the representative interactive graphical programming functions of include:

- ◊ Automatic generation of mechanical components, such as bolt and gear.
- ◊ Customization of CAD menus
- ◊ Creation of user-defined geometry, such as tent design.
- ◊ Parametric design, such as gear design (size, number of teeth, etc.)
- ◊ Automation of graphics related design and planning process, such as automated tool path generation.
- ◊ Engineering analysis and simulation, such as tolerance or fit analysis, and simulation of a mechanism.

Interactive graphical programming can be carried out at three different levels:

- ◊ Medium and High Level Programming
  Developing an interactive graphics program using the programming language provided by a CAD system (AutoLISP, ADS, ARX, Pro/E Develop, Pro/E Tool Kit, etc.) or the graphics routines provided by a Graphical Kernel package (GKS, PHIGS, etc.). These programming environments often support high level programming languages, such as C.
- ◊ On-screen Menu Programming
  Programming the on-screen or pull-down menu using supporting toolkits or by changing the system menu file.
- ◊ Database Level Programming
  The high-level programming that deals with graphical data at record and sub-record level (for new or specialized graphics systems).
- ◊ Device Level Programming
  Development of device drivers (assessment of the bits of the display buffer.)

2. Why we need to write programs in AutoCAD?

Interactive graphical programming has been a central part for the course of Computer Aided Design at many universities. In addition to a general introduction to commercial CAD systems and the background study on computer graphics, the knowledge on interactive graphical programming allows one to fully understand how the CAD/CAM systems function. This understanding is essential to the effective use of various CAD/CAM software tools and to keep up with the fast changing CAD systems.

To reduce the steep learning curve and to maximize the benefit of interactive graphical programming, the AutoCAD programming environment is chosen to carry out the teaching of interactive graphical programming in CAD. AutoCAD is a widely used drafting package. The majority of technical and management staff in industry has some previous exposures to AutoCAD. The programming and customization of standard AutoCAD package to suit the special needs of a company can significantly increase productivity and fully utilize the potential of this CAD tool. Recent advance of AutoCAD programming tools made it an ideal platform to the development of CAD/CAM applications. An AutoCAD based design and manufacturing application can share data with a large number of users, utilize the built-in graphics capability of a proven CAD system, and avoid a lot of efforts on interface to peripheral devices. In addition, the management of design databases and coordination of work done at different stages of product development can be more effectively carried out.

- AutoCAD is an open system, which means that users can adapt many different aspects of AutoCAD to their own needs to build a custom CAD system. AutoCAD provides a very good environment for interactive graphics programming. A user can avoid the time-consuming, low-level graphics programming tasks by
  - ◊ composing a new design and manufacturing application program using the built-in CAD functions
  - ◊ changing the menu interface
  - ◊ writing customized commands
- Unlike many other CAD programs, AutoCAD was intended for more than one specific engineering field. This leads to some awkward structures from mechanical engineering point of view, and the lack of some essential functions for mechanical design.

3. AutoCAD Programming Environment

Most modern CAD programs offer some capabilities to accommodate user-developed programs and extensions to their built-in functions. AutoCAD Release 12 & 13 support three different programming environments: **AutoLISP**, **AutoCAD Development System (ADS)**, and **AutoCAD Runtime Extension (ARX)**. In addition, a **Script File** can be used to "reprogram'' the default AutoCAD menu system by either adding addition items or changing the menu layout.

- **AutoLISP**

  AutoLISP was the first programming language supported by AutoCAD since 1995. As one of the many variations of the LISP (LISt Processing) programming language, AutoLISP is a symbolic manipulation-based, interpreted language that provides a simple mechanism for adding commands to AutoCAD. AutoLISP permits a user to program user input, loops, decisions and more. The LISP syntax of AutoLISP allows convenient manipulation of AutoCAD commands. Although there are some variations depending on the platform, AutoLISP is logically a separate process that communicates with AutoCAD through inter-process communication (IPC).

  The major drawback of AutoLISP is its slow execution speed due to the need of on-line interpretation. This nature precludes its use in serious interactive graphical programming.

- **ADS Programming Interface**

  Similar to AutoLISP, ADS is a programming interface to AutoCAD, introduced with AutoCAD Release 11. ADS applications are written in C and are compiled. This has considerably improved the execution speed of the application programs.

  In AutoCAD Release 11, ADS only supported graphical programming with 2D geometry. Graphical programming with 3D geometric entities and operations were carried out using **Advanced Programming Interface (API)**, also a C library and AutoSolid.

  With the introduction of AutoCAD Release 12, AutoSolid was seamlessly merged into AutoCAD, All API functions are added to the ADS C library.

  Although written in C, ADS applications appear identical to AutoLISP applications to AutoCAD. An ADS application is written as a set of external functions that are loaded by and called from the AutoLISP interpreter. ADS applications communicate with AutoLISP by IPC.

- **The ARX Programming Environment**

  The ARX programming environment was introduced in AutoCAD Release 13. This programming environment differs from the ADS and AutoLISP in a number of ways. An ARX application is a *dynamic link library (DLL)* that shares AutoCAD's address space and makes direct function calls to AutoCAD. Designed with extensibility in mind, the ARX libraries include macros to facilitate defining new classes and offer the ability to add functionality to existing classes in the library at run time. The ARX libraries can be used in conjunction with the AutoCAD Development System (ADS) and the AutoLISP application programming interfaces.

  The ARX programming environment provides an object-oriented C++ application-programming interface that enables developers to use, customize, and extend AutoCAD. The ARX libraries comprise a versatile set of tools for application developers to take advantage of AutoCAD's open architecture, providing direct access to AutoCAD database structures, the graphics system, and native command definition. In addition, these libraries are designed to work in conjunction with the AutoLISP and

AutoCAD Development System (ADS) application programming interfaces so that developers can choose the programming tools best suited to their needs and experience.

In addition to speed enhancements, you can add new classes to the ARX program environment and export them for use by other programs. ARX entities you create are virtually indistinguishable from built-in AutoCAD entities. You can also extend ARX protocol by adding functions at run time to AutoCAD classes.

Part of the ARX environment is a complete library of the ADS functions. This library, often referred to as ADS-Rx, is functionally identical to the standard C ADS library; however, it is actually implemented as a part of AutoCAD. Consequently, it shares AutoCAD's address space along with the other ARX libraries. It uses the ADS library for the following:

Entity selection

Selection set manipulation

Programmable dialog boxes

AutoCAD utility requests, such as *ads_trans()*, *ads_command()*, and *ads_cmd()*, and

Data acquisition

4. Other Useful Information

In addition to this introduction, other useful information was gathered and provided to feasilate a better understanding of interactive graphics programming. These include:

1. A Brief Overview of AutoCAD Development System (ADS)
2. A Tutorial on ADS Programming
3. Additional Reading Materials from Autodesk®
   - Fundamentals of ADS I
   - Fundamentals of ADS II
   - Fundamentals of ADS Environment: Windows™/Visual C++™
4. AutoCAD Script Files
5. AutoCAD Runtime Extension (ARX) Programming Environment
6. Procedures for Building An ARX Application Using MSVC++ 4.0
7. Additional Reading Materials from Autodesk®
   - AutoCAD R13 AutoCAD Runtime Extension (ARX) Overview
   - Building and Debugging ARX Programs using Microsoft Visual C++ 4.0 Developer Studio