Glossary

Although the terminology associated with object-oriented programming and C++ has become reasonably standardized, some object-oriented terms may be slightly different depending on the implementation. Therefore, brief definitions of some of the terminology used in this paper are given below. For more information on these terms see the references in the accompanying article.

Abstract Class. Abstract classes represent the interface to more than one implementation of a common, usually complicated concept. Because an abstract class is a base class to more than one derived class, it must contain at least one pure virtual function. Objects of this type can only be created through derivation in which the pure virtual function implementation is filled in by the derived classes.

The following is an example of an abstract base class:

```
class polygon {
public:
     // constructor, destructor and other member functions
     // could go here...
     virtual void rotate (int i)  =  O; //a pure virtual function
     // other functions go here...
};
```

Other classes, such as square, triangle, and trapezoid, can be derived from polygon, and the rotate function can be filled in and defined in any of these derived classes.

Base Class. To reuse the member functions and member data structures of an existing class, C++ provides a technique called class derivation in which a new class can derive the functions and data representation from an old class. The old class is referred to as a base class since it is a foundation (or base) for other classes, and the new class is called a derived class. Equivalent terminology refers to the base class as the superclass and the derived class as the subclass.

Catch Block. One (or more) catch statements follow a try block and provide exception-handling code to be executed when one or more exceptions are thrown. Caught exceptions can be rethrown via another throw statement within the catch block.

Class. A class is a user-defined type that specifies the type and structure of the information needed to create an object (or instance) of the class.

Concrete Data Class. Concrete data classes are the representation of new user-defined data types. These user-defined data types supplement the C++ built-in data types such as integers and characters to provide new atomic building blocks for a C++ program. All the operations (i.e., member functions) essential for the support of a user-defined data type are provided in the concrete class definition. For example, types such as complex, date, and character strings could all be concrete data types which (by definition) could be used as building blocks to create objects in the user's application.

The following code shows portions of a concrete class called date, which is responsible for constructing the basic data structure for the object date.

```
typedef boolean int;
#define TRUE 1
#define FALSE O

class date {
public:
     date (int month, int day, int year); //Constructor
     ~date(l;                             //Destructor
     boolean set date(int month, int day, int year);
     // Additional member functions could go here. . .

private
     int year;
     int numerical_date;
     // Additional data members could go here...
};
```

Constructors. A constructor creates an object, performing initialization on both stack-based and free-storage allocated objects. Constructors can be overloaded, but they cannot be virtual or static. C++ constructors cannot specify a return type, not even void.

Derived Class. A class that is derived from one (or more) base classes.

Destructors. A destructor effectively turns an object back into raw memory. A destructor takes no arguments, and no return type can be specified (not even void). However, destructors can be virtual.

Exception Handling. Exception handling in C++ provides language support for synchronous event handling. The C++ exception handling mechanism is supported by the throw statement, try blocks, and catch blocks.

Member Functions. Member functions are associated with a specific object of a class. That is, they operate on the data members of an object. Member functions are always declared within a class declaration. Member functions are sometimes referred to as methods.

Object. Objects are created from a particular class definition and many objects can be associated with a particular class. The objects associated with a class are sometimes called instances of the class. Each object is an independent object with its own data and state. However, an object has the same data structure (but each object has its own copy of the data) and shares the same member functions as all other objects of the same class and exhibits similar behavior. For example, all the objects of a class that draws circles will draw circles when requested to do so, but because of differences in the data in each object's data structures, the circles may be drawn in different sizes, colors, and locations depending on the state of the data members for that particular object.

Throw Statement. A throw statement is part of the C++ exception handling mechanism. A throw statement transfers control from the point of the program anomaly to an exception handler. The exception handler catches the exception. A throw statement takes place from within a try block, or from a function in the try block.

Try Block. A try block defines a section of code in which an exception may be thrown. A try block is always followed by one or more catch statements. Exceptions may also be thrown by functions called within the try block.

Virtual Functions. A virtual function enables the programmer to declare member functions in a base class that can be redefined by each derived class. Virtual functions provide dynamic (i.e., run-time) binding depending on the type of object.