Abstract
In this paper we want to point out that Visual Programming ought to mean "Programming by gestures and image manipulation" where the computer manufactures the executable code, rather than forcing the user to learn just another programming language where, for example, the Pascal notation is replaced by flow diagram symbols. Instead we define, as an example of our ideas, a simple visual notation for querying databases via images, where the manipulation of the images and the gestures performed by the user generate SQL-code that can be executed by any Database Manager based on SQL.

Introduction
The programming of computers is an area where development has been very rapid. Programming started as a work for specialists. From switches on a front panel via low level languages, as assemblers, and primitive high level languages it has reached a state where "naïve" users quite simply can produce fairly complex programs.

During the last decade, however, programming has again become a specialists work because of the graphical workstations. It is no longer enough to be a programmer. You must have detailed knowledge of the workstation and its graphical language. Modern programs often require multiple windows, scrollbars and a variety of complicated functions. Good knowledge about algorithms is also required, as interactive programs - with multiple windows and complex user interfaces - are big and slow, so speed is important and thus the ability to choose good and efficient algorithms.

To allow specialists, using computers as tools, to continue as specialists in their own field, without forcing them to become specialists also in the field of modern programming, it is necessary to develop new languages that allow a naïve user to program the graphical workstation without detailed knowledge about the complex environment. Such a development would simplify programming in the same way as the development from assemblers to modern high level languages once did.

Visual Programming
One discipline, partly with aims as above, is visual programming. It is a collection of disciplines, dealing with "programming by the aid of pictures", "teaching the computer to repeat a sequence of operations" and, of course, theoretical aspects of this new research area. Visual programming can be used to produce executable code from the manipulation of pictures and gestures performed by the user by direct translation to the internal "language" of some computer or computer program.

This requires clear syntax and semantics, which in turn can be difficult if the application field of the language is not restrained to a fairly small portion of the otherwise vast field of computer programming. We have chosen to illustrate the main direction of our thoughts by a visual language for querying databases. We translate the manipulation of the images on the screen and the gestures performed by the user to the query language of the database system. A small set of operations has been chosen. The set of operations allow users to do "everything that can be done", but it is not meant to be a versatile tool for manipulating databases. The meaning is to illustrate what we mean by visual programming.

The contents of the database is presented to the user as a set of tables on the screen. We make a difference between what we call "abstract tables" and "real tables". An abstract table allows the user to perform general operations as queries, while the real tables allow the user to browse, update, insert or delete information. Our aim is to point at the possibility to increase the usability of modern computers for "naïve" users by the definition of similar visual means of expression and also the possibility to let such "naïve" users take advantage of the resources of a modern computer without having to become a programming specialist each time he encounters a new problem or starts using a new computer system.
There are three equivalent mathematical theories behind relational databases: Relational algebra, tuple calculus and domain calculus. The operations in our program are based on relational algebra, which is closely related to set algebra and thus, fairly easy to visualize. Actions taken by the user are translated to SQL, a query language based on tuple calculus, where all queries to the database system are on the form:

```
SELECT list-of-columns
FROM list-of-tables
WHERE logical-expression
```

The logical expression can contain so-called sub-selects, each of which is a query of the above form, so fairly complicated queries can be made and even though SQL is not fit to formulate every kind of query it has become a "de facto" standard.

By manipulating the tables on the screen the following operations may be performed:

1. **Projection**, is similar to choosing one or more columns from a table and make a new table consisting of the chosen columns by dragging the selection out from the table. The operation can be made in an ordinary Macintosh manner. When the program recognizes the operation, the cursor changes to the mathematical symbol for the operation:
b. Selection, is similar to choosing rows in a table that meet some logical requirement. The operation can be performed by dragging a "rubber" rectangle around any part of the table excluding the title and the column titles. It can be performed as in projection, and projection can be performed with dragging a rubber rectangle touching the column titles.

Generated expression:
SELECT Name, Dept FROM Employee

h. Union, is similar to adding the rows of one table to those of another table. It requires both tables to have columns with same names and data types. This is performed by putting one table "above" another tables and, twice, "bump" them together.

Generated expression:
SELECT Name, Dept FROM Employee WHERE Dept = 'Sports'

We will consider different "languages" for the user's definition of logical expressions. So far we have only implemented text input of the formulae but intend to develop interactive ("visual") means of expression, similar to the way operations are made.

If we enter "Dept = Sports" the generated expression becomes:
SELECT * FROM Employee WHERE Dept = 'Sports'
Generated expression:

```
SELECT * FROM FishSupply
UNION
SELECT * FROM Meatsupply
```

d._Difference_, is similar to taking the rows from one table if they do not exist in the other table – that is taking the union between two tables and then delete any row that appears in the second. In the program the user starts as in the union operation, but instead of releasing the mouse button at the second “bump”, he separates the tables again. Again, it is required that both tables have columns with same names and data types.

release mousebutton
e. **Intersection**, is similar to taking all rows that are identical in two tables. Also in this operation it is required that both tables have columns with same names and data types. It is performed by taking one table and put it on top of another tables and then, twice, "bump" the top table against the inner border of the bottom table.

Generated expression:

```sql
SELECT * FROM FishSupply A
WHERE NOT EXISTS
  (SELECT *
   FROM MeatSupply
   WHERE A.Dept=Dept
   AND A.Item=Item
   AND A.Volume=Volume)
```

f. **Natural join**, is similar to taking all columns from two tables and merge them into one table containing those rows where values from the original tables, in columns with the same name, are equal, and then omitting the duplicate columns. This operation may be made even if there are no columns with the same name in the two tables but then, as there are no duplicate columns, omissions are not made and all rows remain in the resulting table. The operation is, in this latter case, equal to cartesian product. It is performed by, twice, "bumping" the tables sideways against each others.

Generated expression:

```sql
SELECT * FROM FishSupply A
WHERE EXISTS
  (SELECT *
   FROM MeatSupply
   WHERE A.Dept=Dept
   AND A.Item=Item
   AND A.Volume=Volume)
```
Conclusion

Our example shows that it is possible, at least in a limited area, to define visual means to produce executable code directly from the manipulation of images and gestures performed by the user.

We know that our example suffer from the fact that we, originally, are mathematicians, but we wanted to try something that we understand before undertaking a larger project where the aim is to define visual means of programming for people that have little or no knowledge of computers.

It is difficult to define general purpose visual languages, but it should be possible, as in our example, to find such languages in a large number of fairly limited application areas.

We have also shown that visual means of expression can be a powerful aid for "non-programmer" users as well as for the professional.

Future work

We intend to find means of visual expression for data definition.

After that we will aim at finding visual expressions for "naive" users. In that follow-up project, people from many parts of the scientific community as well as "naive" users will be involved.

Acknowledgement

This work has been performed in the interdisciplinary environment of IPIlab, the Interaction and Presentation Laboratory, where researchers from Computing Science, Psychology and Linguistics work together on human-computer interaction problems. We thank all of them for stimulating discussions which have had strong influence on our work.

The program

A test version of our program is at hand. It implements the "abstract table" part of the project and we hope to have a full version of the program at the beginning of September.

It is written in Object Pascal, utilizes the class library MacApp, and runs om any Macintosh. However, the Macintosh II is needed for speed.

Generated expression:

```
SELECT A.Name, A.Sal, A.Mgr, A.Dept, B.Floor
FROM Employee A, Location B
WHERE A.Dept=B.Dept
```

Constraints to aid the user

We found it difficult to manipulate the images on the screen. Therefore, a number of constraints have gradually been added, so that precision is not needed when performing the operations.

Some examples: The tables are objects and act almost as real objects. When they collide on the screen the "moving" table stops at the edge of the other table. It is possible to move over another table by going far beyond its edge. Also, when "dragging" a table or a column it is sufficient to point anywhere within its title, push the mouse button and drag.

The program

A test version of our program is at hand. It implements the "abstract table" part of the project and we hope to have a full version of the program at the beginning of september.

It is written in Object Pascal, utilizes the class library MacApp, and runs om any Macintosh. However, the Macintosh II is needed for speed.