抽象的非構造情報の可視化 - プログラミング言語史の可視化 -

菊池智、濱本和彦

東海大学　情報理工学部　情報メディア学科

目的: 現代の社会では、非常に多くの種類の情報が氾濫していて、その中には複雑な情報から簡単なものまであるが、とりわけ膨大で複雑な情報は、その理解が非常に難しい。ここで挙げる複雑な情報とは、定量的な単位で表現が出来ない抽象的な情報や、同じディメンジョンで二次元以上の情報を持たない非構造情報などである。ここでは、その両方の条件を持つ情報を抽象的非構造情報と定義する。

本研究では、そのような複雑な情報を可視化して、没入型環境での提示を可能とするデータ表現方法を開発し、より直感的にわかりやすく情報を提示すること目的としている。今回は、その一つの例であるプログラミング言語の歴史を題材として、データ構造の決定と没入型環境での提示を試みた。

プログラミング言語史: 抽象的非構造情報の例として、プログラミング言語の歴史を挙げる。プログラミング言語は、今までに数多くの言語が発表され、それぞれ、他の言語からの影響を受け作成されたり、また現行の言語の後継として改良されたりしている。プログラミング言語の歴史を知ることにより、現在ある言語がどのような考えの基に作成されたのか、どのような流れから来ているかを知ることが出来る。これは、今後の言語の流れの方向性を知る参考にもなる。このため、プログラミング言語の可視化は、プログラミング言語の開発者や使用者にとって、有意義な事となる。

プログラミング言語史を計る単位として、唯一の具体的情報は発表の年代である。これは西暦などの数値により定量的に表現ができるが、それ以外のパラメータ、例えば、オブジェクト指向などの言語のタイプやネットワーク言語などの利用環境などに関する特長には、その単位を定量的に測る尺度は無いために抽象的な情報となる。また、年代という要素とそれ以外の違った要素から構成される「多次元プログラミング言語情報」は非構造でもあるので、これは多次元形状モデルとしての表現が困難である。

可視化方法: 三次元空間上に三軸のグラフを作成し、そこの適当な場所に各言語を配置する方法を取る。そして、それぞれ関係のある言語を線で結ぶという形で提示を行う。

具体的には、汎用型可視化ソフト AVS を使用し、提案する可視化方法にあわせたデータフォーマットを新たに作成し可視化を行う。その映像の提示は、没入型のスクリーンである HoloStage™ で行う。これにより、作成された空間に対して、入り込んでいるかのような状態を提示し、さらに空間の操作により、提示されたプログラミング言語史の空間を見渡し、歩き回る事ができる。

xyz の三軸にそれぞれ以下の要素を振り分けた。

X 軸：言語のパラメータ１

Y 軸：言語のパラメータ２

Z 軸：言語の発表年

まず、Z 軸については発表年を割り当てた。これにより、ある Z 軸の一点、XY 平面で空間を見ると、その年代に発表されたプログラミング言語が見られるということになる。

そして、X 軸 Y 軸の二つについては、言語の特徴を現すパラメータを用いる。この例を挙げると、「ある言語がオブジェクト指向言語か、手続き型言語か」また「ある言語がネットワーク型か、スタンドアロン型か」という分け方がある。これらを割り当てて、提示された映像全体を見渡すと、時代ごとの傾向や流れなどが見られるようになる。本来はＸＹ面状の位置は何らかのルールに従って決定されるべきであるが、今回は、見易さを優先して座標を決めた。

言語同士の結合線は、その言語の後継であり直接関係する場合や、ある言語の作成時に、別の言語の影響を特に受けた場合、言語同士を線で結び、その関係を表した。結合線の色については、線自体の長さの比例で決まり、短いと青系統の色に、長いと赤系統の色になる。

データフォーマット: AVS で読み込むデータファイルは、２種類ある。一つが、言語の座標を記述したファイルで、もう１つが、言語同士の関係を記述したファイルである。それぞれの具体的なデータを以下に示す。

表 1：言語の記述

```
X Y Z data text
0 2 1954 10 Fortran
0 1 1959 10 COBOL
0 -1 1958 10 IAL
0 -1 1958 10 Algol_58
```

表 2：関係の記述

```
dx dy dz X Y Z p
0 0 2 0 2 1954 10
0 0 1 0 2 1956 10
0 -3 2 0 2 1956 10
```

表１は、言語一つ一つの座標と名前を記述したデータで、一行目が、データの種類を記述したラベルになる。二行目から一行がひとつの言語のデータとなり、左から、X 座標、Y 座標、Z 座標、パラメータ、言語名となっている。四つ目の要素については、今回は使用をしていないが、例えば、言語の色や大きさを変えるなど、言語ごとにパラメータを持つ場合に使用できるに用意をした。

表２は、言語間の関係を記述したデータで、これも一行目がデータの種類を記述したラベルである。二行目からは、一行がひとつの関係線のデータとなり、左から、X 座標増加分、Y 座標増加分、Z 座標増加分、X 座標、Y 座標、Z 座標、パラメータとなる。このデータにも関係線ごとにパラメータを持つことが出来るようにしてあり、色や太さなどに使うことも出来るが今回は使用していない。

これらはテキストエディターで編集できるため、すべて手で記述も出来るが、今回は表計算ソフトを使用して言語の座標を記述し、その関係は言語の座標と関係から、関係線の座標を計算するプログラムを組み作成した。

ＡＶＳモジュール構造: ファイルの読み込みは、二つとも、 Rd Column File で読み込み、さらにどちらも table to scatter field で、座標値の数値を、テーブルデータから離散型フィールドデータに変換する。ここで、言語を記述したデータは、同時に extract column で

言語名の列のデータを抜き出し、 text glyph に抜き出した言語名とその座標値に変換したフィールドデータを使用して配置した。関係を記述したデータは、変換したフィールドデータを combine vect でベクトルコンポーネントのデータに変換して glyph で配置をしている。以下に今回使用したモジュール構造を図１に示す。

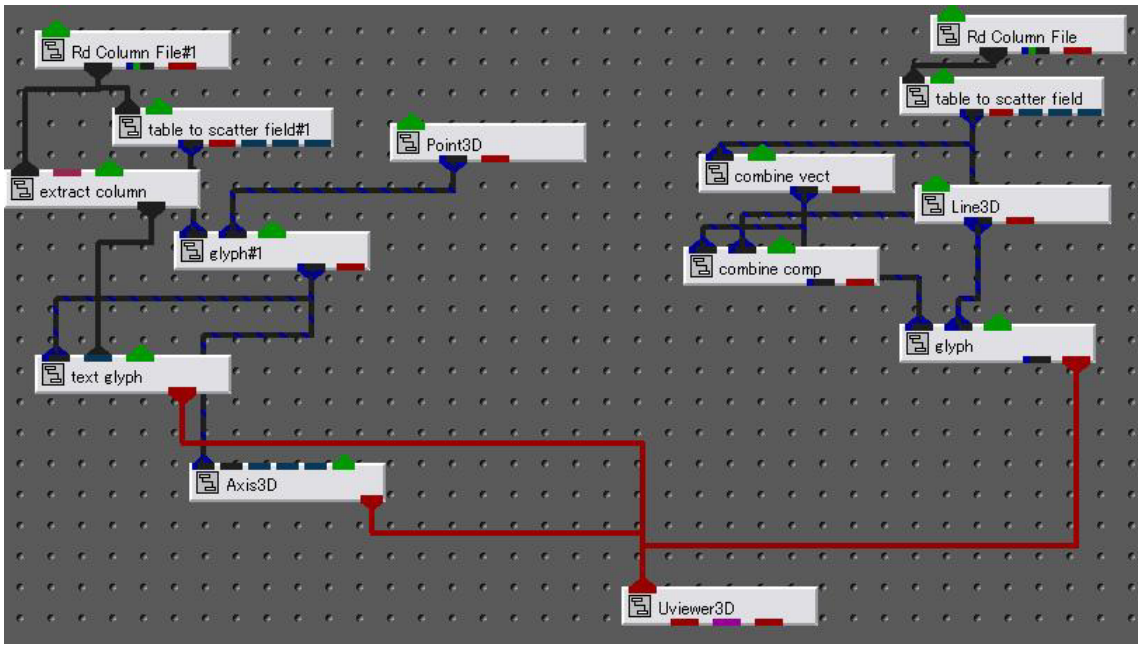

図 1：モジュール構造

評価: 主要言語のみで構成された三次元空間を図２に示す。

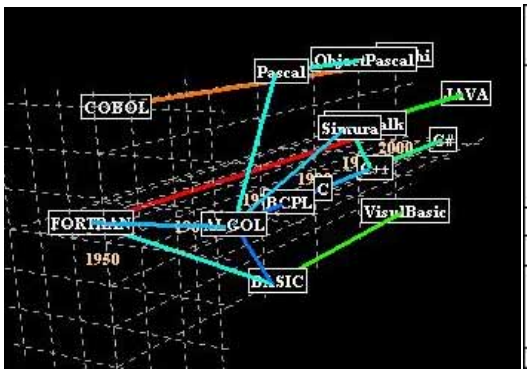

図 2：可視化された主要言語

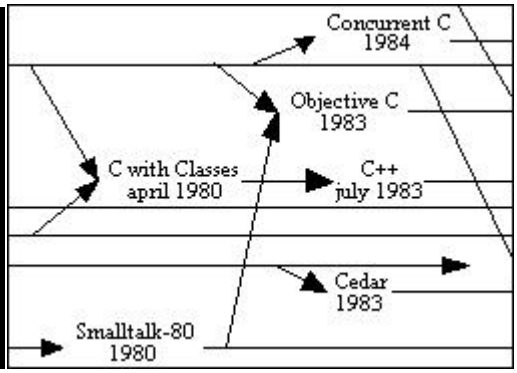

図 3：二次元平面状表現

Ｚ 軸の正の方向へ進んで観察をしたとき、図２にあるような従来の二次元平面での情報表現に比べ、各年代でどの言語が発表されどのように影響されているかがわかり易くなった。そして、時代とともにどのように言語が変遷してきたかががわかりやすくなり、言語の関係についてもその変遷が分かりやすくなった。さらに、三次元物体として扱うことができるため、回転をして、斜めから見ることもできるため、全体の把握が容易になった。しかし、残りの二軸については、見易さを優先して入力した数値であるので、数値自体に意味があるものではなく、流れがどういう言語からどういう言語へ移り変わっているかという情報を提示することができていなかった。そして、言語の数も資料では、300 以上あり、これを一度に提示すると言語の集中した年代付近では、関係線が見難くなってしまった点

もあったので、言語の集中した年代での提示方法が課題である。

提示方法に関して、関係を記述したデータの場合、座標の変化など一部の小さな変化でも、座標を計算するプログラムを使用するため、そのつど手間が掛かり、また，データが二つのファイルにより管理されるため，管理が困難である。そのためデータ管理を簡略化する方法の検討が今後の課題である。

さらに、没入型環境での提示では、５０を超える言語を表示した場合、広い視野角での提示により、さらに全体の把握が容易になるが、入力に対する反応が遅くなり、動作が重くなるという課題がある。

結論: 今回の可視化により、抽象的非構造情報である、プログラミング言語史の三次元空間への可視化が、各時代における情報提示、時代間での言語の変遷、言語の関係についての情報提示に関して有効である事がわかった。

ただ、X-Y 平面上のマッピングをどのように定量化するか，表示データの作成の手間をどれだけ簡略化するか、言語の集中した年代での提示方法の検討、没入型環境での動作の重さ、が課題である。将来的には，AVS の新たなデータフォーマットの一つとして提案したい。

参考文献:

[1] Computer Languages History：http://www.levenez.com/lang/