Преобразование графических форматов данных посредством применения методологии целостности-смысла на примере формата VRML

А.И. Разумовский,
с.н.с., к.т.н., alr@ipu.ru,
ИПУ РАН, г. Москва

Предложена методология чтения графических данных, которая ориентирует разработчика алгоритма на поиск и сублимацию отдельных локальных целостных сущностей, имеющих определенный смысл. Это позволяет избегать лавинообразного роста контролируемых параметров и тем самым ведет к легкости и эффективности разработки алгоритма, снижая время разработки и повышая качество целевого алгоритма.

The methodology of reading graphic data which focuses the developer of an algorithm on search and sublimation of the separate local complete entities making a certain sense is offered. It allows to avoid the avalanche growth of controlled parameters and by that leads to ease and efficiency of development of an algorithm, reducing time of development and increasing quality of a target algorithm.

Любая значимая информация, в частности данные графических форматов типа VRML, всегда содержит в себе смысл, то есть осмысленна. Соответственно, и подход, и исследование, и преобразования информации должны сочетаться с указанным фактом. В сегодняшнее время напротив, всякий алгоритм чтения данных сообразуется с информационными частностями – символ, строка, порядок строк, число, - а не с тем, что эти символы и строки представляют собой в целом, - то есть идут по пути лавинообразного нарастания сложности.

Разрешение сложности издавна соотносят с требованием разделения исходного материала на части, элементы [1-4], а также формализацией и автоматизацией на основе формальной логики [5-6]. Поиском подобного преодоления сложности занимались в 60-х и 70-х годах, в конце которых появился объектно-ориентированный язык программирования C++. Факт разрешения сложности при разбиении исходного качества кажется столь очевидным, сколь порой обманчивым в реализации. Например, разделение часто трактовалось как исчезновение, сокрытие элементов качества. Так идея Парнаса [1] заключалась в сокрытии информации о проектном решении, относящемся к точкам ветвления или возможных изменений, от других модулей.

Корень проблемы, однако, находится вовсе не в преодолении некой технологической сложности обработки имеющейся информации. Настоящим камнем преткновения является творчески действующий человеческий фактор, который иногда присутствует явно, а зачастую невидимо, в каждом информационно-технологическом процессе. В зависимости от личных способностей человека, преходящих обстоятельств, а также от индивидуальных особенностей решаемой задачи человеческому творчеству приходится оперировать различным материалом, как в количественном, так и в качественном отношении. А человеческое мышление весьма ограничено, из-за малого объема кратковременной памяти, – магическим числом Миллера, 7+-2 [7]. Кроме того, согласно Саймону [8], скорость обработки информации человеком также имеет значение – операция восприятия данных «стоит» примерно 5 секунд. Соответственно, объем и качество данных имеют основополагающее значение.

На примере чтения данных графического формата VRML, покажем, как и за счет чего упрощается процесс разработки алгоритма, если акцент делается на поиске и анализе элементов данных, заключающих в себе определенный смысл (рис. 1).

рис. 1 Информационные составляющие методологии смысла

Как и для всякого алгоритма следует определить главную структуру хранения данных.

```
class  readDataVRML{
  int iter_curr; // итератор
  int res_size;
 ////////////////////////////////
public:
    map< int,string>  mw;          // ассоциативный массив  слов (в том числе чисел), где ключ - координата слова
  map<int, VRMLgroup>  mGroups1;
  map<int, VRMLgroup>  mGroups2;
  map<int, VRMLgroup>  mGroups;  // группы всех типов вместе
  vector< int > vGroupClose1;  // вектор  индексов конца группы 1
  vector< int > vGroupClose2;  // вектор  индексов конца группы 2
  vector< int > vGroupClose ;  // вектор  индексов конца  групп всех типов вместе
 readDataVRML( ) {   iter_curr=0;}
  void add (string& s ,int cooWord, int o1, int cl1, int o2,int cl2){   mw[cooWord]=s; }
  void end (){iter_curr= mw.size()-1;}
   int size() {return  mw.size();}
   bool eof(){ return iter_curr>=  mw.size();}
  void unget(){iter_curr?iter_curr--:0; }
  void start (){iter_curr=0;}
};
```

рис. 2 Объявление главной структуры данных алгоритма

Главной структурой хранилища для алгоритма был определен класс readDataVRML (рис. 2).

Основной функцией алгоритма является процедура Split, которая содержит последовательный набор операций, направленный на получение конкретного результата. Каждая из операций имеет определенный и отчётливый смысл (рис. 3).

```
void Split(){
  readDataVRML  rd;
    SplitVrml(  src,  rd );
    FormGroup(rd.mGroups, rd.vGroupClose );
   FindGroupContext( rd.mGroups, rd.mGroups);
    GroupMeaning( rd, rd.mGroups );
      vector< vector< string> >    vvDest;
      GetDataEx  ("Normal",  "vector",  rd.mGroups,  vvDest  );
}
```

рис. 3 Программный код основной функции алгоритма чтения данных VRML

Первой операцией – SplitVrml - является сканирование входящей информации для определения координат элементов, разделяемых пробелами, запятыми и некоторыми другими символами. Эта операция наполняет соответствующие вектора объекта rd.

Следующей операцией осуществляется поиск и формирование составных элементов, групп, которые внутри могут содержать атомарные элементы и такие же группы: FormGroup. Основное предназначение этой операции найти границы содержания смысловых групп (рис. 4).

```
{
    vector< int >  vg=vg0;
    map<int,VRMLgroup>  mg =mg0;
 for(int i=0;;i++){
   map<int, VRMLgroup>::iterator itm = mg.end() ; --itm;
                    vector<  int  >::iterator  itv=upper_bound(vg.begin(),  vg.end(),  (*itm).second.oc  );
   mg0[(*itm).first].cc=*(itv);
   mg.erase(itm); vg.erase(itv);
   if(mg.empty()) break;
 }
```

рис. 4 Алгоритм операции поиска границ групп

Границами групп данных формата VRML являются открывающиеся и закрывающие скобки двух сортов: ‘{’ ,‘[,’ ,‘}’ и ‘]’. Фрагмент VRML- данных изображен на рисунке 5.

```
normal DEF FaceN Normal {
vector  [
0 0 -1,
0 0 1,
0 1 0,
-0.866 0.5 0,
-0.866 -0.5 0,
0 -1 0,
0.866 -0.5 0,
0.866 0.5 0,
]}
normalPerVertex TRUE
coordIndex [
4, 3, 5, -1,0, 5, 3, -1,3, 2, 0, -1,
1, 0, 2, -1,
]
normalIndex [
0, 0, 0, -1,0, 0, 0, -1,0, 0, 0, -1,
0, 0, 0, -1,
]
}
```

рис. 5 Фрагмент файла данных формата VRML

Для определения взаимоотношения групп между собой используется операция FindGroupContext (рис.6), где исследуются отношения групп между собой, то есть снаружи или внутри находятся группы друг относительно друга.

```
int  FindGroupContext
    ( map<int, VRMLgroup>& mWhere,
      map<int, VRMLgroup>& mWhat )
```

рис. 6 Сигнатура функции определения взаимоположения групп

Операцией, завершающей построение целостного смысла исследуемых VRML- данных, осуществляется поиск внутреннего содержания и контекста групп - GroupMeaning.

После того, как создан целостный смысл данных, можно использовать накопленную информацию для нахождения необходимых для обработки и помещения в соответствующие хранилища элементов. Для этого применяется функции GetData и GetDataEx (рис.7). Каждая из указанных функций извлекает из смысла данных содержание найденной по заданному контексту группы. Поскольку группы могут иметь разные сорта границ, то исследуются сначала группы с границами сорта ‘{’ и ‘}’, а затем – группы с границами ,‘[‘ и ‘]’.

```
vector< string> vs;
  rr= GetData ("coordIndex", rd.mGroups1, vs);
  if(!vs.size()) rr= GetData ("coordIndex", rd.mGroups2, vs);
```

рис. 7 Вызов функции GetData

Функция GetDataEx отличается от GetData возможностью задать сложный контекст и вернуть множество содержаний сразу из нескольких групп (рис. 8).

```
vector< vector< string> >    vvDest;
   rr= GetDataEx ("Normal", "vector", rd.mGroups, vvDest );
```

рис. 8 Применение функции GetDataEx

Результаты, которые возвращает функция GetDataEx удобно наблюдать при помощи отладчика в IDE MS Visual Studio, и убедиться в том, что возвращаемые функцией данные адекватны исходным данным файла VRML (рис 9).

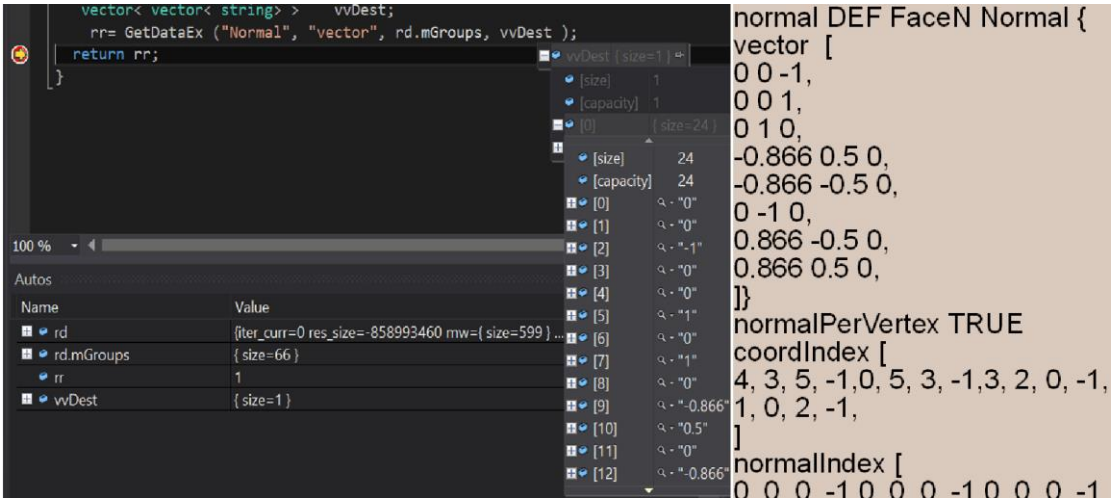

рис. 9 Проверка соответствия результатов алгоритма исходным данным

Предложенная методология чтения графических данных ориентирует разработчика алгоритма на поиск и сублимацию отдельных локальных целостных сущностей, имеющих определенный смысл. Это позволяет избегать лавинообразного роста контролируемых параметров и тем самым ведет к легкости и эффективности разработки алгоритма, снижая время разработки и повышая качество целевого алгоритма.

Литература

1. Parnas D.L., On the Criteria To Be Used in Decomposing Systems into Modules, Commun. Ass. Comput. Mach., Vol. 15, No. 12, December 1972 pp. 1053 - 1058.
2. Артамонов Е.И., Хачумов В.М. Синтез структур специализированных средств машинной графики. М., Институт проблем управления.-1991.
3. Дейкстра Э., Дисциплина программирования., М., Мир, 1978, 275 с.
4. Грис Д., Наука программирования. - М.: Мир, 1984. - 289 с.
5. Чери С, Готлоб Г., Танка Л. Логическое программирование и базы данных: Пер. с англ. - М.: Мир, 1992. - 352 с, ил.
6. Малпас Дж. Реляционный язык Пролог и его применение. /Пер. с англ.; Под ред. В.Н. Соболева. — М.: Наука, 1990.
7. Miller, G. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. The Psychological Review vol.63(2), p.86., March.1956. (Миллер Дж. Магическое число семь, плюс или минус два. - В кн.: Инженерная психология. М. 1964)
8. Simon, H. 1982. The Sciences of the Artificial. Cambridge, MA: The MIT Press.