

АХМЕТГАРЕЕВ ТИМУР РАМИЛЕВИЧ
ДУБОВ МИХАИЛ СЕРГЕЕВИЧ

271ПИ

КОМПОНЕНТНАЯ МОДЕЛЬ С
ДЕКЛАРАТИВНЫМ
ОПИСАНИЕМ СОСТАВНЫХ
ТИПОВ

- ПАРСЕРЫ И ГЕНЕРАТОРЫ КОДА
- ВИЗУАЛИЗАЦИЯ АРХИТЕКТУРЫ КОМПОНЕНТ

СРЕДСТВА ОПИСАНИЯ МОДЕЛЕЙ

- Используются подмножества декларативных языков **VRML** и **X3D**.

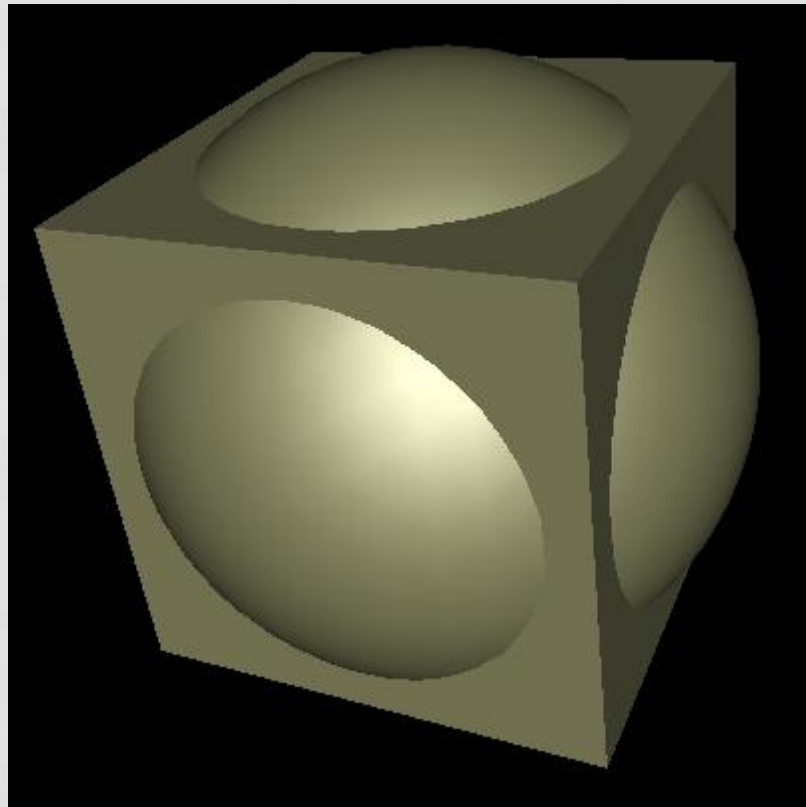
➤ Являются стандартами **ISO**

```
#VRML 97
```

```
Group {  
  children [  
    Box {}  
    Sphere {  
      radius 1.3  
    }  
  ]  
}
```

```
<X3D>  
  <Group>  
    <fieldValue name='children'>  
      <Box />  
      <Sphere radius='1.3' />  
    </fieldValue>  
  </Group>  
</X3D>
```

VRML/X3D КАК ЯЗЫК ОПИСАНИЯ 3D



VRML/ХЗD КАК ЯЗЫК ОПИСАНИЯ ДРУГИХ МОДЕЛЕЙ

```
DEF person1 Person {  
  name "Alice Naur"  
  age 30  
}  
  
Person {  
  name "Clara Backus"  
  age 9  
  
  mother USE person1  
  
  father Person {  
    name "Bob Backus"  
    age 35  
  }  
}
```

РАЗБОР ИСХОДНЫХ ТЕКСТОВ

1. Лексический анализ

- Разбиение входного потока символов на лексемы.

2. Синтаксический анализ (парсинг)

- Сопоставление потока лексем с формальной грамматикой языка;
- Построение дерева разбора, в данном случае – графа сцены.

3. Семантический анализ

- Определение соответствия типов.

ЛЕКСИЧЕСКИЙ АНАЛИЗ

- ***StreamTokenizer*** – реализованный в библиотеках Java лексический анализатор.
- **Настраивается** путем задания:
 - Терминальных символов (зарезервированных символов и ключевых слов языка);
 - Символов, определяющих однострочные комментарии;
 - Символов-кавычек;
 - Символов, играющих роль пробелов;
 - ...

ЛЕКСИЧЕСКИЙ АНАЛИЗ

```
StreamTokenizer tokenizer = new StreamTokenizer(...);
```

```
tokenizer.wordChars('a', 'z'); // Id's  
tokenizer.wordChars('A', 'Z'); // Id's  
tokenizer.wordChars('0', '9'); // Id's  
tokenizer.wordChars('_', '_'); // Id's
```

```
tokenizer.quoteChar('');
```


```
tokenizer.whitespaceChars(' ', ' ');  
tokenizer.whitespaceChars('\n', '\n');  
tokenizer.whitespaceChars('\t', '\t');  
tokenizer.whitespaceChars('\r', '\r');
```

```
tokenizer.commentChar('');
```

```
tokenizer.ordinaryChar('{'); // Terminal  
tokenizer.ordinaryChar('}'); // Terminal  
tokenizer.ordinaryChar('['); // Terminal  
tokenizer.ordinaryChar(')'); // Terminal
```

ЛЕКСИЧЕСКИЙ АНАЛИЗ

```
Group {  
  children [  
    Box {}  
  
    Sphere {  
      radius 1.3  
    } # Sphere  
  ] # Children  
} # Group
```



**<ID, "Group"> <'{'> <ID, "children"> <'['>
<ID, "Box"> <'{'> <'}'> <ID, "Sphere"> <'{'>
<ID, "radius"> <Number, 1.3> <'}'> <']'> <'}'>**

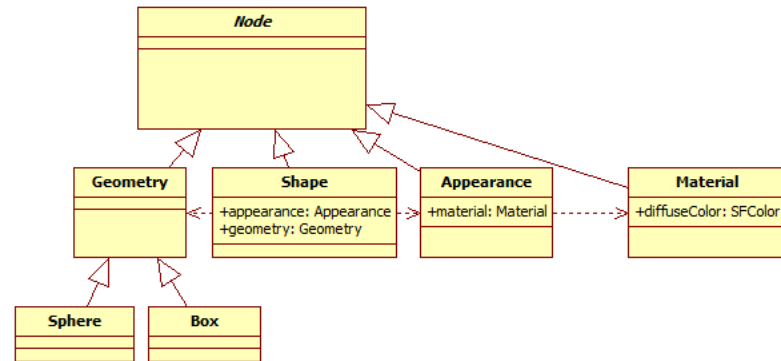
СИНТАКСИЧЕСКИЙ АНАЛИЗ

- Парсеры VRML и X3D реализуют **два разных алгоритма.**
 - **Метод рекурсивного спуска** для VRML
 - Нисходящий метод.
 - **SAX-парсер** для X3D
 - Последовательный;
 - Событийный.

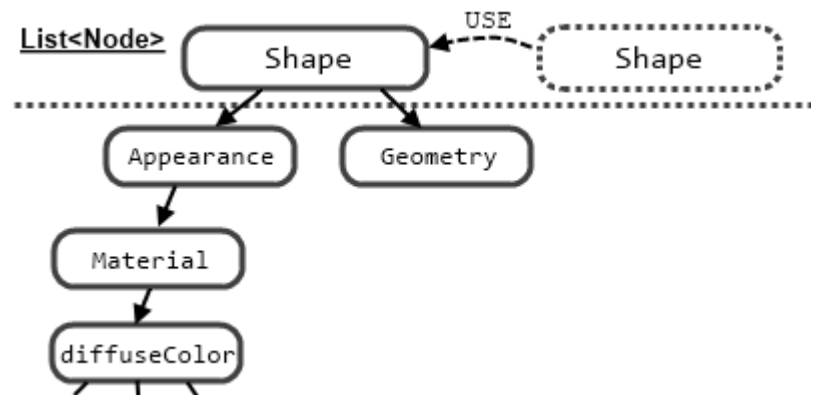
СИНТАКСИЧЕСКИЙ АНАЛИЗ

```
DEF shape1 Shape {  
  appearance Appearance {  
    material Material {  
      diffuseColor 0.2 0.5 1  
    }  
  }  
  geometry Sphere {}  
}
```

USE shape1



PARSER



СИНТАКСИЧЕСКИЙ АНАЛИЗ VRML

- **Метод рекурсивного спуска** (*recursive-descent parsing*)
 - Каждое из правил (продукций) **формальной грамматики** языка реализуется в парсере с помощью соответствующего метода.

...

nodeStatement ::=

DEF *nodeName* *id* *node* |

USE *nodeName* *id* |
node

...

...

private boolean **parseNodeStatement**() {

return

(tryMatch("DEF") && matchId() && parseNode()) ||

(tryMatch("USE") && matchId() && instantiateNodeById()) ||

(parseNode());

}

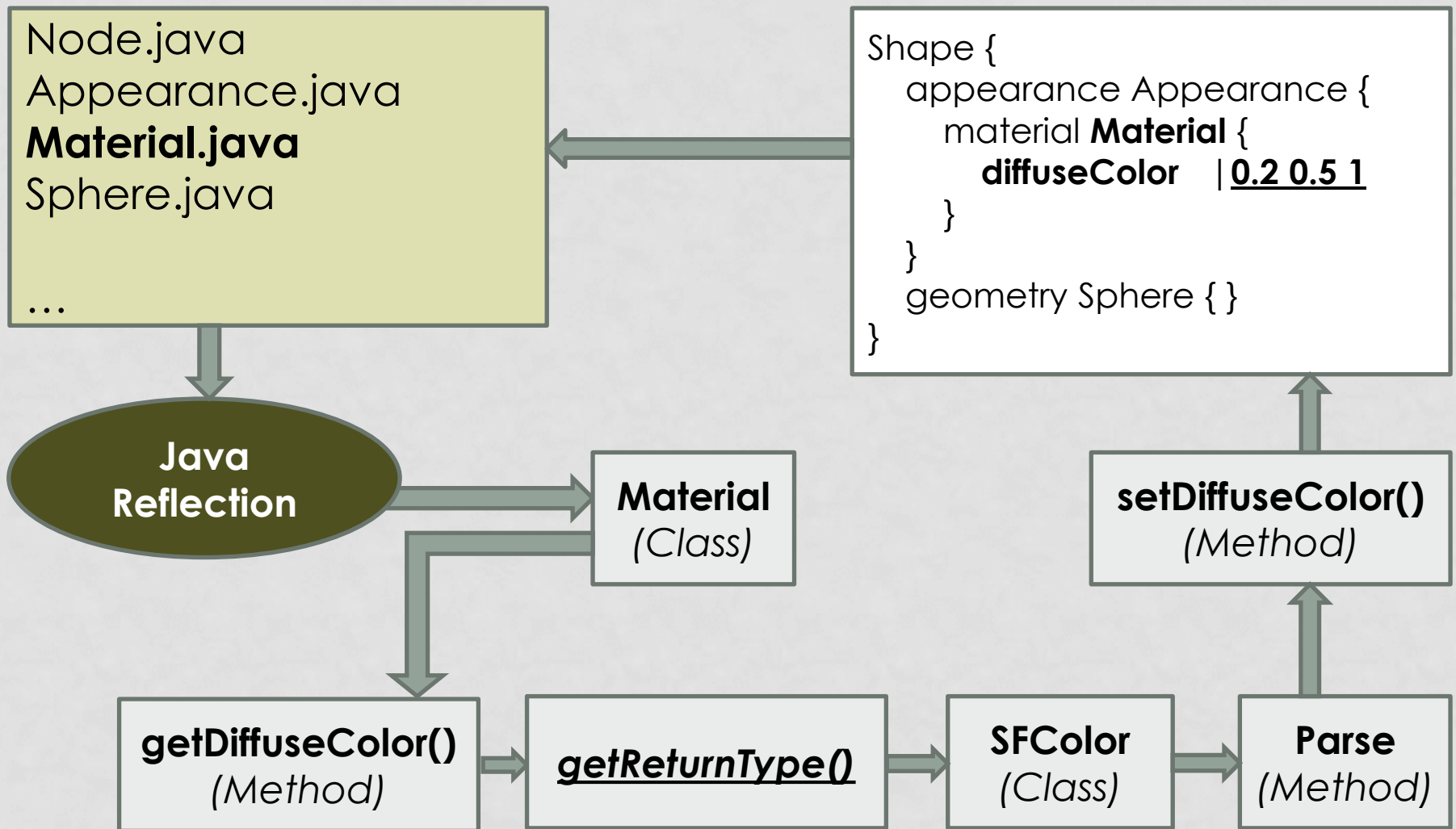
...

СИНТАКСИЧЕСКИЙ АНАЛИЗ VRML

- Грамматика VRML является **неоднозначной** (*ambiguous*)
 - Корректное считывание значений полей узлов требует семантического знания об их типах

```
Shape {  
  appearance Appearance {  
    material Material {  
      diffuseColor 0.2 0.5 1  
    }  
  }  
  geometry Sphere { }  
}
```

СИНТАКСИЧЕСКИЙ АНАЛИЗ VRML



СИНТАКСИЧЕСКИЙ АНАЛИЗ VRML

■ Ошибки:

1. Лексические

```
Shape {  
  goemetry Sphere {}           # geometry  
}
```

2. Синтаксические

```
Shape __  
  geometry Sphere {}           # {  
}
```

3. Семантические

```
Shape  
  geometry Appearance {}       # ???  
}
```

4. Логические

```
DEF geom01 Box {}  
Shape  
  geometry DEF geom01 Sphere {} # defined  
}
```

СИНТАКСИЧЕСКИЙ АНАЛИЗ VRML

- Распознавание ошибок – в «режиме паники» (*panic-mode recovery*)
 - Используются «синхронизирующие символы» - например, пара скобок { }
- Предложение исправлений лексических ошибок

```
Shape {  
  appearance Appearance {  
    material Mate rial {  
      diffuseColor 0.2 0.5 1  
    }  
  }  
  geomeetry Sphere { }    #geometry – возможная замена  
}
```

СИНТАКСИЧЕСКИЙ АНАЛИЗ X3D

- **X3D – XML-подобный** язык, может быть разобран методом **SAX** (*Simple API for XML*).
- Весь анализ основан на обработке 4-х событий:
 1. `openingTag(name)`
 2. `closingTag(name)`
 3. `attribute(value, name)`
 4. `textNode(text)`

СИНТАКСИЧЕСКИЙ АНАЛИЗ X3D



```
<SomeTag attribute1='aaa', attribute2='2.3'> Hello, world </SomeTag>
```

attribute("attribute1", "aaa")

textNode("Hello, world")

attribute("attribute2", "2.3")

openingTag("SomeTag")

closingTag("SomeTag")

СЕМАНТИЧЕСКИЙ АНАЛИЗ

- Синтаксически код верен, семантически – нет:

```
Shape {  
  appearance Appearance {  
    material Box { }  
  }  
}
```

- Такие ситуации, опять же, можно выявить с помощью **рефлексии**.

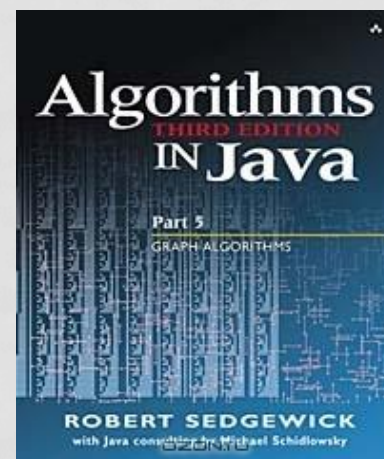
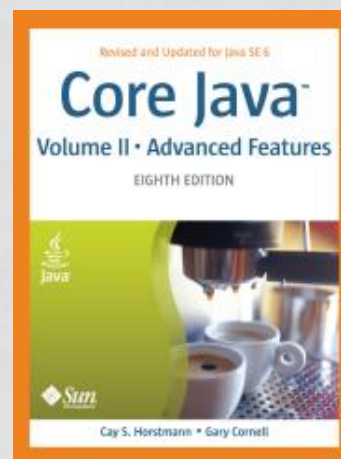
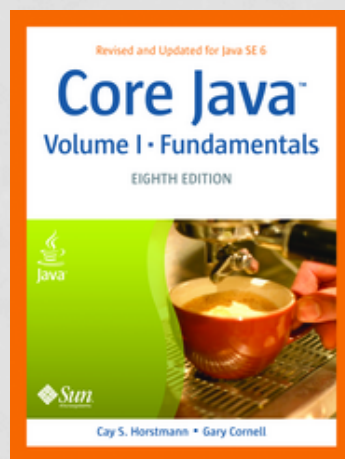
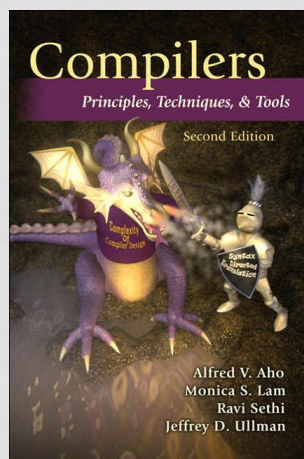
ГЕНЕРАТОРЫ КОДА

- Реализуются рекурсивной процедурой **траверсирования** DAG-графа сцены
- Связка **парсеры + кодогенераторы** позволяет решать ряд задач:
 - Построение редакторов компонентных моделей с функцией загрузки/сохранения;
 - Выполнение конвертации VRML \Rightarrow X3D и X3D \Rightarrow VRML

РЕЗУЛЬТАТЫ

- С использованием **разных алгоритмов** построены парсеры для двух разных декларативных языков;
- Эти парсеры находятся в числе наиболее развитых с точки зрения **диагностики ошибок**;
- Реализована возможность **конвертирования** из классического **VRML**-подобного стиля описания моделей в более современный **XML**-подобный стиль;
- Показана возможность использования VRML **не только** для визуализации 3D-графики.

ЛИТЕРАТУРА



- A. V. Aho, M. S. Lam, R. Sethi and J. D. Ullman, **Compilers**: principles, techniques, and tools, 2nd ed. MA: Prentice Hall, 2006.
- C. S. Horstmann and G. Cornell, **Core Java**, 8th ed., vol. 1: Fundamentals. MA: Prentice Hall, 2007.
- C. S. Horstmann and G. Cornell, **Core Java**, 8th ed., vol. 2: Advanced features. MA: Prentice Hall, 2008.
- R. Sedgewick, **Algorithms in Java**, 4th ed., CA: Addison-Wesley Educational Publishers Inc., 2010.
- ISO/IEC 14772-1:1997 and ISO/IEC 14772-2:2004 — **Virtual Reality Modeling Language** (VRML). ISO/IEC 19775 – **X3D**.