FUNDAÇÃO GETULIO VARGAS
ESCOLA DE MATEMÁTICA APLICADA

CRISTHIAN GRUNDMANN

GEODESIC TRACING: VISUALIZAÇÃO DE CURVAS E SUPERFÍCIES ATRAVÉS DE GEODÉSICAS

Rio de Janeiro
2022

CRISTHIAN GRUNDMANN

GEODESIC TRACING: VISUALIZAÇÃO DE CURVAS E SUPERFÍCIES ATRAVÉS DE GEODÉSICAS

Trabalho de conclusão de curso apresentada para a Escola de Matemática Aplicada (FGV/EMAp) como requisito para o grau de bacharel em Matemática Aplicada.

Área de estudo: curvas e superfícies.

Orientador: Asla Medeiros e Sá

Rio de Janeiro

2022

Lista de códigos

1 Programas

O usuário se comunica com a interface através de um texto, chamado de programa, que contém as objetos de interesse. Por exemplo:

Código 1.1 – Exemplo de objetos

```
1 #circle and tangents
2 param r : [/2, 1];
3 param o : [0, 2pi];
4 curve c(t) = r(cost, sint, 0), t : [0, 2pi];
5 grid k : [0, 2pi, 8];
6 define k2 = k + o;
7 point p = ck2;
8 vector v = c'k2 @ p;
```

O programa começa com uma linha de comentário, estilo Python. Os objetos `r` e `o` são parâmetros nos intervalos indicados e `c` é um círculo parametrizado por `t` no intervalo indicado($/2 = 1/2$). Os parâmetros podem ser alterados por controles deslizantes na interface. O objeto `k` é uma grade de 8 pontos igualmente espaçados no intervalo indicado. A definição `k2` serve apenas de conveniência, usada no ponto `p` e no vetor `v`. Note que `p = ck2` ($p = c(k2)$) usa uma notação sem parênteses para a aplicação de funções. Como o ponto `p` depende de `k`, um ponto é desenhado para cada valor de `k` da grade. O mesmo ocorre para o vetor `v`. As grades são tratadas como constantes, e servem para desenhar múltiplas instâncias dos objetos desenháveis.

Os objetos só podem se referir aos objetos declarados anteriormente. Aplicações de funções devem ter a quantidade certa de argumentos. Operações com tuplas devem ter quantidades consistentes de elementos. Componentes devem ter índices corretos. Intervalos não podem depender de parâmetros ou grades.

O programa deve seguir uma gramática formal que determina as estruturas sintáticas permitidas. A gramática especifica: o formato da declaração de cada tipo de objeto; e as operações matemáticas e suas ordens de precedência. A gramática é definida pelo código 1.2.

Código 1.2 – Gramática livre de contexto

```
1  PROG    = DECL PROG | ;
2
3  DECL    = "param"     id ":" INTS ";" ;
4  DECL    = "grid"      id ":" GRIDS ";" ;
5  DECL    = "define"    id "=" EXPR ";" ;
6  DECL    = "curve"     FDECL "," TINTS ";" ;
7  DECL    = "surface"   FDECL "," TINTS ";" ;
8  DECL    = "function"  FDECL ";" ;
9  DECL    = "point"     id "=" EXPR ";" ;
10 DECL    = "vector"    id "=" EXPR "@" EXPR ";" ;
11
12 FDECL   = id "(" IDS ")" "=" EXPR ;
13 IDS     = IDS "," id | id ;
14 INT     = [ EXPR "," EXPR ] ;
15 GRID    = [ EXPR "," EXPR "," EXPR ] ;
16 TINT    = id ":" INT ;
17 INTS    = INTS "," INT | INT ;
18 TINTS   = TINTS "," TINT | TINT ;
19 GRIDS   = GRIDS "," GRID | GRID ;
20
21 EXPR    = ADD;
22 ADD     = ADD "+" JUX | ADD "-" JUX | JUX ;
23 JUX     = JUX MULT2 | MULT ;
24 MULT    = MULT "*" UNARY | MULT "/" UNARY | UNARY ;
25 MULT2   = MULT2 "*" UNARY | MULT2 "/" UNARY | APP ;
26 UNARY   = "+" UNARY ;
27 UNARY   = "-" UNARY ;
28 UNARY   = "*" UNARY ;
29 UNARY   = "/" UNARY ;
30 UNARY   = APP ;
31 APP     = FUNC UNARY | POW ;
32 FUNC    = FUNC2 "^" UNARY | FUNC2 ;
33 FUNC2   = FUNC2 "_" var | FUNC2 "'" | func ;
34
35 POW     = COMP "^" UNARY | COMP ;
36 COMP    = COMP "_" num | FACT ;
37 FACT    = const | num | var | "(" TUPLE ")" | "[" TUPLE "]" | "{" TUPLE
       "}" ;
38 TUPLE   = ADD "," TUPLE | ADD ;
```

Os termos à esquerda de uma igualdade são os não-terminais, em maiúsculo, e `PROG` é o não-terminal inicial. Os termos em minúsculo são terminais, e representam um token. Os termos entre aspas representam literalmente o texto entre aspas. Por exemplo, `func` representa o nome de uma função. Termos O lado direito de uma igualdade especifica as possíveis formas sentenciais do não-terminal à esquerda, separadas por `|`. Uma forma

sentencial pode conter símbolos terminais e não-terminais. Uma sentença, ou programa, é gerado a partir de `PROG`. Iterativamente, se substitui um não-terminal por uma de suas formas sentenciais. Note que a primeira substituição é a de `PROG`. Observe na linha 10 a especificação de um vetor. A palavra chave `vector` é necessária, seu nome é representado por `id` (identificador), e os termos `EXPR` representam expressões matemáticas.

O não-terminal `EXPR` está definido na linha 21, e representa uma expressão matemática.

Implementação de sintaxe que considera a estetica natural da escrita matemática que precisa ser traduzida para sintaxe de linguagem computacional. Isso é feito pelo parser. A justificativa para essa abordagem é motivada pela experiencia de especificar desenhos de objetos gráficos em bibliotecas de uso corrente, tais como sagemath, manin etc. Isso justifica a especificação de uma gramática livre de contexto. Explicitada a seguir. Trabalhos futuros: grade variável Sobre o texto que descreve a gramática Fazer referencia à calculadora C++ Fazer referência ao site que valida a não ambiguidade da gramática Por ter influenciado na sintaxe proposta (como?) Tentar lembrar um exemplo de ambiguidade que foi resolvido para a versão atual da gramática. Descrever o LL1 como certificado de não ambiguidade.