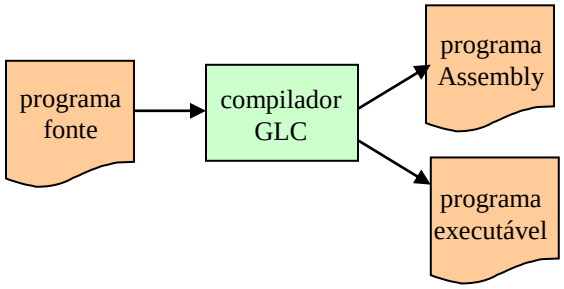
Pontifícia Universidade Católica de Minas Gerais
ICEI - Curso de Ciência da Computação
Disciplina: Compiladores
Prof. Alexei Machado

Trabalho Prático

A construção de um compilador para uma linguagem gráfica que executa scripts de animação

Objetivo

O objetivo do trabalho prático é o desenvolvimento de um compilador completo que traduza programas escritos na linguagem fonte "GL" para o ASSEMBLY e a linguagem de máquina de um processador virtual. No caso do programa conter erros, o compilador deve reportar o primeiro erro e terminar o processo de compilação. O programa executável do compilador deve se chamar "GLC" e receber como parâmetro (argumento) o nome completo do programa fonte a ser compilado (extensão .GL). No caso de sucesso, o compilador deverá produzir 2 arquivos, com mesmo nome do programa-fonte: o correspondente em linguagem ASSEMBLY (extensão .ASM) e o correspondente em linguagem de máquina (extensão .EXE).



Definição da Linguagem-Fonte GL

A linguagem "GL" é uma linguagem gráfica de alto nível que executa scripts de animação. "GL" possibilita a definição de modelos e transformações elementares de rotação, translação e escala. A máquina-alvo para a qual são gerados os programas-objetos possui um coprocessador que executa rotinas específicas para gerar a visualização dos modelos, portando seu conjunto de instruções contém instruções especiais, além do usualmente encontrado na maioria das linguagens de montagem para máquinas de propósito geral.

A linguagem oferece tratamento para 5 tipos de estruturas gráficas: ponto, face, objeto, cor e luz. Estas estruturas são declaradas como constantes no início do programa. Além disso, é possível utilizar variáveis e constantes do tipo inteiro e real de ponto fixo (máximo de 4 casas para a parte inteira e 4 casas para a parte fracionária, separadas por ponto). Números

reais podem começar ou terminar com ponto. Os identificadores são compostos de letras e dígitos, começando necessariamente por uma letra. Há distinção entre maiúsculas e minúsculas.

Os caracteres permitidos em um arquivo fonte são as letras, dígitos, espaço, ponto, vírgula, ponto-e-vírgula, dois-pontos, parênteses, colchetes, chaves, mais, menos, aspas, apóstrofo, barra, asterisco, cifrão, exclamação, interrogação, maior, menor e igual, além da quebra de linha (bytes 0Dh e 0Ah). Qualquer outro caractere é considerado inválido, não podendo aparecer nem em comentários.

Os comandos existentes em "GL" permitem escalar, rotacionar e transladar um objeto e paralisar a animação por um período de tempo. A linguagem permite, ainda, comandos de atribuição a variáveis, blocos (begin - end), estruturas de repetição (while), estruturas de teste (if - then - else), expressões aritméticas com inteiros e reais, expressões lógicas e expressões relacionais para condição de teste e repetição. Todo comando ou declaração é terminado ponto-e-vírgula. Comentários vêm entre /* e */.

A ordem de precedência nas expressões é:

a) parênteses;
b) negação lógica (not);
c) multiplicação aritmética (*), lógica (and) e divisão (/);
d) subtração (-), adição aritmética (+) e lógica (or);
e) comparação aritmética (==, !=, <, >, <=, >=).

São palavras reservadas da linguagem:

declare	point	face	object	color	light	var
const	integer	real	scale	pause	rottrans	
while	if	then	else	and	or	not
==	=	(	)	,	+	-
*	<	>	!=	>=	<=	/
begin	end	;				

A estrutura básica de um programa-fonte é da forma:

declare ***Declarações Bloco-de-Comandos***

A seguir, é feita a descrição informal da sintaxe das declarações e comandos da linguagem:

1. Declaração de pontos: é iniciada por *point*, à qual segue-se uma lista de uma ou mais declarações da forma *id = (cx, cy, cz);* onde *id* é o identificador do ponto e *cx, cy, cz* são constantes inteiras ou reais, precedidas ou não de sinal, que definem as coordenadas x,y,z do ponto no espaço.

2. Declaração de pontos de luz: é iniciada por *light*, à qual segue-se uma lista de uma ou mais declarações semelhantes às descritas na declaração de pontos.

3. Declaração de cores: é iniciada por *color*, à qual segue-se uma lista de uma ou mais declarações da forma *id = n-tons, r1,g1,b1, r2,g2,b2;* onde *id* é o identificador da cor, *n-tons* é o número de tons que a cor oferecerá para as rotinas de visualização, *r1,g1,b1* são as componentes RGB do tom inicial e *r2,g2,b2* são as componentes RGB do tom final da cor. Cada componente deve ser um inteiro de 0 a 63.

4. Declaração de faces: é iniciada por *face* , à qual segue-se uma lista de uma ou mais declarações da forma *id = id-cor,lista-de-pontos ;* onde *id* é o identificador da face, *id-cor* é o identificador da cor da face (a qual já deve ter sido declarada) e *lista-de-pontos* é uma série de 3 ou mais identificadores de pontos, separados por vírgulas.

5. Declaração de objetos: é iniciada por *object* , à qual segue-se uma lista de uma ou mais declarações da forma *id = lista-de-faces;* onde *id* é o identificador do objeto e *lista-de-faces* é uma série de 1 ou mais identificadores de faces, separados por vírgulas.

6. Declarações de variáveis: é iniciada por *var*, à qual segue-se uma lista de uma ou mais declarações da forma *tipo lista-de-ids;* onde *tipo* pode ser *integer* ou *real* e *lista-de-ids* é uma série de 1 ou mais identificadores, separados por vírgulas.

7. Declarações de constantes: é iniciada por *const*, à qual segue-se uma lista de uma ou mais declarações da forma *id = num;* onde *id* é um identificador e *num* uma constante numérica, precedida ou não de sinal.

8. Blocos de comandos: São da forma *begin lista-de-comandos end*

9. Comandos de atribuição: São da forma *id = expressão;* .

10. Comando de repetição: é da forma *while expressão-lógica comando* . *Comando* pode ser único ou um bloco.

11. Comando de teste: é da forma *if expressão-lógica then comando1 else comando2*. *Comando1 e comando2* podem ser únicos ou substituídos por bloco. A parte do *else* é opcional.

12. Comando de escala: é da forma *scale id,ce;* onde *id* é um identificador de objeto e *ce* uma expressão real ou inteira.

13. Comando de Pausa: é da forma *pause ct;* onde *ct* é uma expressão real ou inteira denotando o número de segundos em que a animação ficará suspensa.

14. Comando de Luz: é da forma *light id;* onde *id* é um identificador de luz. Este comando acende a luz referenciada pelo identificador *id*. Apenas uma luz pode estar acesa em um determinado instante da animação.

15. Comando de Rotação - Translação: é da forma
rottrans id,Dalpha,Dbeta,Dgama,Dx,Dy,Dz; onde *id* é um identificador de objeto, *Dalpha,Dbeta,Dgama* são expressões representando os intervalos de rotação em torno dos eixos y,z,x e *Dx,Dy,Dz* são expressões representando os intervalos de translação ao longo dos eixos x,y,z. Apenas um objeto pode ser visualizado em um determinado instante da animação.

Considerações Gerais

O trabalho será desenvolvido durante o semestre e dividir-se-á em 4 partes:

I. Implementação do Analisador Léxico
II. Implementação do Analisador Sintático
III. Implementação das rotinas semânticas
IV. Implementação do Gerador de Código

1. O trabalho deverá ser feito em grupos de dois ou três alunos, sem qualquer participação de outros grupos e/ou ajuda de terceiros. Cada aluno deve participar ativamente em todas as etapas do trabalho. Os componentes dos grupos devem ser informados na aula de 01/03/2013 e não poderão ser alterados durante o semestre. Os alunos que não tiverem feito grupos até esta data serão agrupados pelo professor de maneira arbitrária, em grupos de 2 ou 3 alunos.

2. A codificação do trabalho deve ser feita em linguagem C, C++ ou Java, em uma plataforma de desenvolvimento homologada e instalada nos laboratórios do Instituto de Informática. Não poderão ser utilizados bibliotecas gráficas ou qualquer recurso que não esteja instalado oficialmente no laboratório.

3. O trabalho será avaliado em 2 etapas:

   a) as práticas TP1 e TP2 (10 pontos), em uma única versão final, deverão ser postadas no SGA até às 12:00 horas do dia 16/04/2013, juntamente com a documentação, e apresentadas conforme o cronograma. O atraso na entrega implicará em perda de 3 pontos por dia.
   b) as práticas TP3 e TP4 (15 pontos), em uma única versão final, deverão ser postadas no SGA até às 12:00 horas do dia 04/06/2013, juntamente com a documentação, e apresentadas conforme o cronograma. **Para esta etapa não se admite atraso, ou seja, não serão avaliados trabalhos entregues após 04/06.**

4. Os trabalhos devem ser postados na forma de um arquivo compactado com software disponível no laboratório, com **tamanho máximo de 500KB**, e seu nome deve ser o número de matrícula de um dos componentes (Ex:346542.zip). **Os arquivos fontes devem estar no diretório raiz** e devem conter o nome de todos os componentes do grupo no início do código.

5. **Trabalhos iguais, na sua totalidade ou em partes, copiados, "encomendados" ou outras barbaridades do gênero, serão severamente penalizados. É responsabilidade do aluno manter o sigilo sobre seu trabalho, evitando que outros alunos tenham acesso a ele. No caso de cópia, ambos os trabalhos serão penalizados, independentemente de quem lesou ou foi lesado no processo. Partes de trabalhos apresentados em outros semestres só poderão ser aproveitados se os componentes forem exatamente os mesmos em ambos os semestres.**

6. Será pedida ao Colegiado uma advertência formal no caso de cópia.

7. Durante a apresentação poderão ser feitas perguntas relativas ao trabalho, as quais serão consideradas para fim de avaliação. Todos os componentes devem comparecer e serem capazes de responder a **quaisquer perguntas e/ou alterar o código de qualquer parte do trabalho**. A avaliação será individual.

8. A avaliação será baseada nos seguintes critérios:

    - Correção e robustez dos programas
    - Conformidade às especificações
    - Clareza de codificação (comentários, endentação, escolha de nomes para identificadores)
    - Organização dos arquivos do projeto
    - Parametrização
    - Apresentação individual
    - Documentação, descrita em cada uma das etapas