TKOM – Dokumentacja wstępna

Język opisu scen graficznych

Mateusz Plesiński

**Opis**

Celem projektu jest stworzenie prostego języka programowania umożliwiającego przedstawianie scen graficznych. Język zapewnia procedury rysujące figury geometryczne, wyrażenia warunkowe, pętle (typu for, for\_each) , wykonywanie funkcji co pewien określony czas itp.

**Funkcjonalność:**

* wbudowane procedury rysujące figury - prymitywy (box, sphere)
* wbudowane procedury umożliwiające manipulacje figurami (move, scale, chngcol)
* możliwość definiowania i używania własnych funkcji
* operator okresowy („ ~ ”) zapewniający możliwość powtarzania funkcji co pewien zadeklarowany czas
* podstawowe mechanizmy programowania tj. :
  + instrukcja warunkowa
  + pętle for i for\_each
  + operatory matematyczne
  + operatory logiczne
  + operatory przyrównania
  + nawiasowanie

**Wymagania funkcjonalne:**

* Kontrola poprawności podawanych danych oraz zgłaszanie wykrytych błędów
* Poprawne odczytywanie, parsowanie i analiza skryptów z plików tekstowych
* Generacja scen 3D ze skryptów

**Wymagania niefunkcjonalne:**

* Komunikaty o błędach podczas procesu analizy pliku powinny być zrozumiałe dla użytkownika i jasno wskazywać popełnione błędy

**Specyfikacja techniczna**

Projekt będzie wykonany w języku C++ (standard C++11) kompilacja za pomocą kompilatora Microsoft Visual Studio. Przewiduję wykorzystanie zewnętrznych bibliotek do testów jednostkowych *Boost,* konfiguracji procesu budowania zostanie wykorzystany program *Scons.*

**Przykłady**

1. func moja\_funkcja(figura) {
2. figura move (10, 0, 0);
3. }
4. func scene (){
5. szescian draw box pos(10, 10, 10) col(255, 0, 0) dim(10, 10, 10);
6. moja\_funkcja(szescian) ~(10);
7. i = 0;
8. szesc\_kolek[10] draw box pos(50, 50, 25) col(0, 0, 0) dim(10, 10, 10);
9. for\_each szesc\_kolek : iterator {
10. iterator move (0, 0, 20 + i);
11. i = i + 20;
12. }
13. j;
14. for (j = 0; j < 10; j = j + 1) {
15. if (j < 5) {
16. szesc\_kolek[j]move (0, 20, 0);
17. } else {
18. szesc\_kolek[j]move (0, -20, 0);
19. }
20. }
21. }
22. func skaluj(figura) {
23. figura scale (10, 10, 10);
24. }
25. func scene {
26. kula draw sphere pos(50, 50, 0) col(255, 0, 0) dim(100);
27. skaluj(kula) ~(10);
28. prostokat draw box pos(10, 10, 0) col(125, 50, 0) dim(10, 10, 0);
29. for (j = 0; j < 100; j = j + 1) {
30. if (j < 50) {
31. prostokat scale (0, 0.1, 0);
32. } else {
33. prostokąt scale (0, -0.1, 0);
34. }
35. }
36. }

**Gramatyka**

**Program** = { FuctionDeffinition } .

**FunctionDeffinition** = ‘func’ , Identifier , ‘(’ , [ Parameters ] , ‘)’ , Block .

**Block** = “{” , { Instruction | Block } , “}” .

**Instruction** = { ConditionalStatment , “;” | LoopStatment , “;”| InitializationStatment , “;” | AssignStatment , “;” | FunctionCall , “;” | GraphicFunction , “;”| ReturnStatment , “;” } .

**GraphicFunction** = Identifier , [Index] , Operation , [FigureType] , (FigureAttributes | NewVector).

**Operation** = “draw” | “move” | “chngcol” | “scale” .

**NewVector** = “(“ , Assignable , “,” , Assignable , “,’ , Assignable , “)” .

**FigureType** = “box” | “sphere” .

**FigureAttributes** = Position , Color , Dimensions .

**Position** = “pos” , “(“ , Assignable , “,” , Assignable , “,’ , Assignable , “)” .

**Color** = “col” , “(“ , Assignable , “,” , Assignable , “,’ , Assignable , “)” .

**Dimensions** = “dim” , “(“ , Assignable , “,” , Assignable , “,’ , Assignable , “)” .

**ReturnStatment** = “return” , Assignable

**ConditionalStatment** = “if ” , “(” , Condition , “)” , Block, [ “else” , Block] .

**LoopStatment** = ForLoop | ForEachLoop .

**ForLoop** = “for” , “(” , (AssignStatment ) , “;” ,

Condition , “;” , AssignStatment, “)” Block .

**ForEachLoop** = “for\_each” , Identifier , “:” , Identifier , Block .

**InitializationStatment** = Identifier , [Index] , [ “=” Assignable ] .

**AssignStatment** = Identifier , [Index] , “=” , Assignable .

**FunctionCall** = SimpleFunctionCall | PeriodicFUnctionCall .

**SimpleFunctionCall** = Identifier “(” Parameters “)” .

**PeriodicFunctionCall** = SimpleFunctionCall [ PeriodicOperator “(“ Assignable “)” ] .

**Condition** = AndCndition , { OrOperator , AndContition } .

**AndCond** = EqualityCOndition , {AndOperator , EqualityCondition}

**EqualityCondition** = RelationalCondition , [ EqualOperator , RelationalCondition ]

**RelationalCondition** = SimpleCondition , [ RelativeOperator ,SimpleCondition ]

**SimpleCondition** = [NegOperator] , ( Assignable )

**Parameters** = { Identifier } .

**Assignable** = Expression | FunctionCall .

**Expression** = MultipExpresion { AddtivOperator , MultipExpresion} .

**MultipExpresion** = SimpleExpresion . { MultipOpperator , SimpleExpresion} .

**SimpleExpresion** = Variable | Number | ParentExpresion .

**ParentExpresion** = “(” , Expresion , “)” .

**Variable** = Identifier , [ Index ] .

**Index** = “[“ , Assignable , “]” .

**NegOperator** = “!” .

**OrOperator** = “or” .

**AndOperator** = “and” .

**EqualOperator** = “==” | “!=” .

**RelativeOperator** = “<” | “<=” | “>” | “>=” .

**LogicalOperator** = “or” | “and” .

**PeriodicOperator** = “~” .

**AddtivOperator** = “+” | “-“

**MultipOpperator** = “\*” | “/” .

**Number** = [ “-“ ] , Digit , { Digit } [ “.” { Digit } ] .

**Digit** = “0” ..”9” .

**Identifier** = Letter {Digit | Letter} .

**Letter** = “a”..”z” | “A”..”Z” | ”\_” | ”-”.

**Lista zdefiniowanych tokenów**

‘.’ , ‘,’ , ‘func’ , ‘(‘ , ‘(‘ , ‘{’ , ‘}’ , ‘[‘ , ‘]’ , ‘;’ , ‘:’ , ‘||’ , ‘&&’ , ‘!’ , ‘=’ , ‘==’, ‘!=’ , ‘~’ , ‘<’ , ‘>’ , ‘>=’ , ‘<=’ , ‘+’ , ‘-‘ , ‘/’ , ‘\*’ , ‘if’ , ‘else’ , ‘return’ , ‘for’ , ‘for\_each’ , ‘draw’ , ‘scale’ , ‘move’, ‘chngcol’, ‘box’, ‘sphere’, ‘pos’, ‘dim’, ‘col’