PLT (Summer 2014): Proposal

Firefly3D

An Educational Programming Language for Creating 3D Graphics

Roy Aslan, Prerna Chikersal, Alex Shnayder – {ra2752, pc2667, ajs2119}@columbia.edu

1. Description and Motivation

Logo[1] is a family of programming languages originally conceived in 1967 to be a tool for interactive learning. One of the most popular features found in Logo environments is that of the "Turtle", representing a hypothetical robotic creature whose two-dimensional movements could be controlled by a user supplying a series of commands and values. As the turtle would move, its path could be traced with a colored line, thus allowing the user to build shapes and images, and employ what is known as "turtle graphics" or "turtle geometry"[1].

Our language, Firefly3D, will be a modification of the more traditional, 2D Logo dialects. It will give our turtle, or more specifically our firefly, the ability to move in three dimensions. Therefore, Firefly3D will allow users to build 3D graphics, ranging from basic to complex, via turtle geometry in a simple, intuitive manner. Ultimately, the goal of this language is to help students learn about 3D geometry.

2. Coordinate System

We will be using the Right-handed coordinate system, as explained In [2].

3. Comments

```
/*...This is a
        multi-line
                comment...*/
```

4. Primitive Types

int -- Declares an integer type value, which is usually used for specifying number of units the firefly should fly or the degrees by which it should turn.
float -- Declares a floating point value, which is usually used for specifying number of units the firefly should fly or the degrees by which it should turn.
bool -- Declares a true/false value.
string -- Declares a string, which is usually used for printing text on the screen/console.

5. Structured Types

vec3 -- Declares a 3-elements vector, which is usually used to specify pen colors.
planeEq -- Declares an equation of the form ax+by+cz+d=0, which defines a plane.

6. Keywords

if	elif	else	switch	-- Used for conditional statements
while				-- Used for control loops
repeat				-- Used for control loops, where the number of iterations is known.
break				-- Used to break or exit from a control loop
true		false		-- Boolean values
define				-- Used to define functions

7. Standard Library

penUp()	-- Turns off the "pen" mode. Firefly can move without leaving a line
penDown()	-- Turns on the "pen" mode if it was already off
isPenDown()	-- Returns true if the "pen" mode is on
rotateHori(int or float)	-- Rotates the firefly in the xz plane
rotateVerti(int or float)	-- Rotates the firefly in the xy plane
forward(int or float)	-- Moves the firefly forward by the specified distance
setPenColor(vec3)	-- Sets a 3-element vector as the pen color
vec3 getPenColor()	-- Returns a 3-element vector describing the current pen color
getX()	-- Returns the 1st element of a vec3 type 3-element vector
getY()	-- Returns the 2nd element of a vec3 type 3-element vector
getZ()	-- Returns the 3rd element of a vec3 type 3-element vector
getCurrPos()	-- Returns the current coordinates of the firefly as a vec3 type 3-element vector
getCurrHoriR()	-- Returns the rotation of the firefly on the xz plane
getCurrVertiR()	-- Returns the rotation of the firefly on the xy plane
clearScr()	-- Clears all lines from the screen
resetPos()	-- Resets the position of the firefly to (0,0,0)
setViewRotation(true or false)	-- Turns continuous rotation of the model on or off
print(int or float or tuple or string)	-- Prints arguments to the console. Usually used for debugging.

8. Math Library

rand()	-- Returns a random number
sin(angle)	-- Returns the sine of an angle in degrees
cos(angle)	-- Returns the cosine of an angle in degrees

9. Operators

+ addition
- subtraction
* multiplication
/ division
^ power

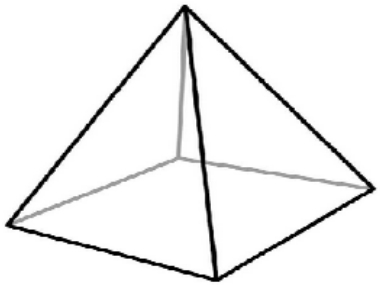
== exactly equal to
!= not equal to
>, <, >=, <= comparison operators
&& and operator
|| or operator
= assignment
>> Eg: scalar [x1,y1,z1] >> planeEq returns true if point (x1,y1,z1) is above planeEq
<< Eg: scalar [x1,y1,z1] << planeEq returns true if point (x1,y1,z1) is below planeEq

10. Other reserved symbols

; end of statement
[ ] encloses a scalar
. dot access Eg: scalar.getX, etc
{} defines scope
"..." encompasses strings

11. Sample program

Goal: Display a simple 4 x 4 square pyramid.



Sample Code:

```
/* First create the 4 x 4 square base of the triangle. */

define makeSquare (int x)
{
   /* Creates a square of side length x */
   int y = 4;
   while (y > 0)
   {
      forward(x);
      rotateHori (90);
   }
}
makeSquare (4);
```

```
/* Now create a leg of the pyramid */
rotateHori (135);
rotateVert (45);

forward (4);

/* Continue the next leg back to the base of the pyramid */
rotateVert (-90);

forward (4);

/* Move the firefly to an adjacent corner of the base in
preparation for the last two pyramid legs */
rotateHori (135);
rotateVert (45);
penUp ();
forward (4);
penDown ();

/* Create the last two legs */
rotateHori (135);
rotateVert (45);

forward (4);
rotateVert (-90);
forward (4);
```

12. References

[1] Logo Foundation - http://el.media.mit.edu/logo-foundation/logo/index.html
[2] Right-handed coordinate system - http://mathworld.wolfram.com/Right-HandedCoordinateSystem.html