Triplex Language Guide

Version 3.0
October 18, 2021

1. Introduction

What is Triplex?

Triplex is the term used for representing variables with 3 components (x,y,z). This makes equations very simple and easy to read. It works well with 3D models that also have 3 dimensions.

What would an equation in Triplex look like?

Writing an equation in triplex would be familiar to anyone that has seen fractal equations in 2D as they look very similar. For example a 2D Mandelbrot looks like:

```
f = f^8 + cp;
```

Writing an equation for a 3D Mandelbulb in triplex looks like:

```
f = f^8 + tp;
```

The built in variable at the end is different and stands for Complex Point in 2D and Triplex Point in 3D. Like all Mandelbrot equations, that is the point where the fractal equation is calculated.

Limitations

One downside of using triplex algebra is that it does not have a complete set of defined math like some other number systems. Currently, the only algebraic commands that are fully defined are add, subtract, and power. Multiply and divide are fully defined for scalars only. Scalars are numbers like 3 or more precisely (3,0,0). This may change in the future as more definitions are added to Fracton.

2. Predefined Variables

Some variables are pre-defined and are used as an interface between the language and the application. The following variables are pre-defined: f, pi, tp, p1-p6, maxit, n, test.

f:

f is the main triplex variable and is usually iterated. For a Julia type expression, f is initialized to the triplex value tp. For a Mandelbrot type expression, f is initialized to zero. The magnitude of f is tested to see if it is larger than bailout. If so, the iteration is stopped and f is declared to have escaped. The number of iterations required before f escapes is used to determine the outer surface of the 3D fractal. The value of f after the last iteration is returned to Fracton but is not used at this time.

Summary:

```
Julia: f0 = tp; f = f^8 + c; // Common Julia Fractal of f to the 8th power
Mandelbrot: f0 = 0; f = f^8 + tp; // Mandelbulb Fractal of f to the 8th power
```

Stop iterating f if mag(f) >= bailout

pi:

pi is the triplex scalar constant (3.14…,0,0).

e:

e is the triplex scalar constant (2.718…,0,0).

tp:

tp is the triplex point in 3D fractal space where the formula will be evaluated. The formula should make no assumption about the limits of tp. The value may be in any direction and may not be visible on any view.

p1-p6:

p1-p6 are triplex values supplied to the formula from the settings view. The values are related to tp and can be used to offset, rotate, and scale tp without any other arithmetic. p1-p6 are also used for animation.

p1-p6 = scale(rotate(tp - position))

maxit:

maxit is the maximum number of iterations from the settings view.

n:

n is the iteration number of the loop. n is zero based so the first time through the loop n = 0.

test:

test is the escape test result for the fractal. If test is false, the fractal is declared escaped and the loop stops. If test is true the loop continues as long as the maximum number of iterations has not been exceeded. Use the magsqr function (magnitude squared function) for faster execution than the mag function. The value of test is returned to Fracton after each iteration.

Example: test = magsqr(f) < 4;

3. User Defined Variables

To add a user defined variable, simply use it in an equation. You should set its value before you read its value.

4. Initialization of Variables

Statements before the @beginloop; compiler directive are only executed on the first iteration of the loop.

Example:
```
f =0; // Initializes f to 0
@beginloop; // Start of the loop
f = f^8 + tp; // Execute until test is false or maxIter is reached
test = magsqr( f ) <= 4; // Stop if |f| squared is greater than 4
```

5. Constants

Constants are triplex double precision. Exponents are IEEE compliant. To enter a triplex constant use the triplex constant function or its shortcut (1,2,3) as shown in the example below.

Examples:

```
a = -3.14e-10; b = 4E10; c = 3; // c is (3,0,0)
b = (2,4,6);
c =triplex(2,4,6);
```

6. Command Summary

Arithmetic: + - * / ^ %

Assignment: =

Comment: // /* */

Compare: < > <= >= == != && ||

Conditional Expression: ?:

Non-Trigonometric Functions: abs ceil floor mag magsqr pow sqrt

Trigonometric Functions: acos acosh asin asinh atan atan2 atanh cos cosh exp log log10 sin sinh tan tanh

Removed: flip imag real conj cosxx

Triplex Functions: triplex xcomponent ycomponent zcomponent

7. Arithmetic

In all of the descriptions below a and b are triplex numbers. Each has a x, y, and z component. So, the variable a could be represented as (x,y,z).

+

Adds two triplex numbers a and b and returns a triplex number.

```
a + b
c.x = a.x + b.x
c.y = a.y + b.y
c.z = a.z + b.z
```

Example: c = a + b;

-

Subtracts two triplex numbers a and b and returns a triplex number.

```
a - b
c.x = a.x - b.x
c.y = a.y - b.y
c.z = a.z - b.z
```

Example: c = a - b;

Unary minus negates a variable or parenthesized expression.

Example: b = -a; b = -(a * 2);

*

Multiplies two triplex numbers a and b and returns a triplex number. A scalar is a number whose y and z components are both 0. When neither a or b is a scalar the result may not be consistent with the result obtained from the power operation.

```
a * b
if (a is a scalar)
{
        c.x = a.x * b.x
        c.y = a.x * b.y
        c.z = a.x * b.z
}
else if (b is a scalar)
{
        c.x = a.x * b.x
        c.y = a.y * b.x
        c.z = a.z * b.x
}
else // neither a or b is a scalar
{
        r1 = sqrt(a.x * a.x + a.y * a.y + a.z * a.z)
        r2 = sqrt(b.x * b.x + b.y * b.y + b.z * b.z)
        phi1 = atan2(a.y,a.x)
        phi2 = atan2(b.y,b.x)
        if(r1 > 0.0) theta1 = acos(a.z/r1) else theta1 = 0.0
        if(r2 > 0.0) theta2 = acos(b.z/r2) else theta2 = 0.0
        c.x = r1 * r2 * sin(theta1 + theta2) * cos(phi1 + phi2)
        c.y = r1 * r2 * sin(theta1 + theta2) * sin(phi1 + phi2)
        c.z = r1 * r2 * cos(theta1 + theta2)
}
```

Example: c = a * b;

/

Divides two triplex numbers a and b and returns a triplex number. If neither number is a scalar the result may not yield the same result as the power operation.

Starting with version 3.0.3, dividing by a scalar now produces results consistent with division in other number systems.

Before version 3.0.3:

1/2 = -0.5 yes there is a minus sign there.

Starting with version 3.0.3:

1/2 = 0.5 like you would expect.

If you have a fractal made with the earlier behavior use the following commands to get the original result.

For example if you had the equation:

```
f = f^2 - f^4/6 + tp;
```

Replace the line above with the following commands:

```
temp = f^4/6;
temp = (-xcomponent(temp),xcomponent(temp),xcomponent(temp));
f = f^2 - temp + tp;
```

```
a / b
if (b is a scalar)
{
        if( b ≠ 0)
        {
                c.x = a.x / b.x
                c.y = a.y / b.x
                c.z = a.z / b.x
        }
        else c = 0
}
else if (a is a scalar)
{
        bin = b^-1
        c.x = a.x * bin.x
        c.y = a.x * bin.y
        c.z = a.x * bin.z
}
else // neither a or b is a scalar
{
        bin = b^-1
        r1 = sqrt(a.x * a.x + a.y * a.y + a.z * a.z)
        r2 = sqrt(bin.x * bin.x + bin.y * bin.y + bin.z * bin.z)
        phi1 = atan2(a.y,a.x)
        phi2 = atan2(bin.y,bin.x)
        if(r1 > 0.0) theta1 = acos(a.z/r1) else theta1 = 0.0
        if(r2 > 0.0) theta2 = acos(bin.z/r2) else theta2 = 0.0
        c.x = r1 * r2 * sin(theta1 + theta2) * cos(phi1 + phi2)
        c.y = r1 * r2 * sin(theta1 + theta2) * sin(phi1 + phi2)
        c.z = r1 * r2 * cos(theta1 + theta2)
}
```

Example: c = a / b;

^

Raises the triplex number a to the power of the x component of b and returns a triplex number. The operator ^ is right associative so a ^ b ^ c = a ^ (b ^ c). The c language uses pow(a , b) instead of the ^ operator.

```
a ^ b
r = sqrt(mx * mx + my * my + mz * mz)
phi = atan2(my,mx)
if (r > 0) theta = acos(mz / r)
else theta = 0
rpower = pow(r,fp.power)
phipower = phi * fp.power
thetapower = theta * fp.power
c.x = rpower * sin(thetapower) * cos(phipower)
c.y = rpower * sin(thetapower) * sin(phipower)
c.z = rpower * cos(thetapower)
```

Example: c = a ^ b;

%

Performs the scalar modulus operation on the x component of a triplex number. For the triplex modulus see the mag function. Returns 0 for b <= 0.

```
a % b
if (b > 0) c.x = a.x % b.x
else c.x = 0
c.y = 0
c.z = 0
```

Example c = a % b;

8. Assignment

=

The equal copies the value of a triplex expression to a triplex variable. Every expression must start with a variable followed by = .

a = b

Example: a = b;

9. Comment

//

Any time // is encountered the rest of the line is ignored by the compiler.

Examples:

```
// This line is a comment
a = b + c; // Everything to the right of // is a comment
```

/*

The start of a block comment.

*/

The end of a block comment.

Example:

```
/* All of these three lines
a = b * c;
are comments */
```

10. Compare

<

a < b returns 1 if a.x is less than b.x. Otherwise, it returns a 0.

Example c = a < b;

>

a > b returns 1 if a.x is greater than than b.x. Otherwise, it returns a 0.

Example c = a > b;

<=

a <= b returns 1 if a.x is less than or equal to b.x. Otherwise, it returns a 0.

Example c = a <= b;

>=

a >= b returns 1 if a.x is greater than or equal to b.x. Otherwise, it returns a 0.

Example c = a >= b;

==

a == b returns 1 if a.x is equal to b.x. Otherwise, it returns a 0.

Example c = a == b;

!=

a != b returns 1 if a.x is not equal to b.x. Otherwise, it returns a 0.

Example c = a != b;

&&

a && b returns 1 if both a.x and b.x are greater than 0. Otherwise, it returns a 0.

Example c = a > 1 && a < 0;

||

a || b returns 1 if either a.x or b.x are greater than 0. Otherwise, it returns a 0.

Example c = a > 1 || a < 0;

11. Conditional Expression:

?:

The conditional expression a ? b : c returns b if a.x > 0 otherwise it returns c. This differs from the C language in that all of the expressions are evaluated whether they are used or not.

Example: f = f >= 0 ? (f - 1) : (f + 1);

12. Non-Trigonometric Functions

abs

Returns the absolute value of the individual components of a triplex number. See mag() for the triplex modulus, triplex absolute value, or triplex magnitude. For the modulus of the x component only see % in the arithmetic section.

```
abs( a )
b.x = abs( a.x )
b.y = abs( a.y )
b.z = abs( a.z )
```

Example: b = abs(a);

ceil

Returns the ceiling of the individual components of a triple number.

```
ceil(a)
b.x = ceil( b.x )
b.y = ceil( b.y )
b.z = ceil( b.z )
```

Example: b = ceil(a);

floor

Returns the floor of the individual components of a triplex number.

```
floor(a)
b.x = floor( a.x )
b.y = floor( a.y )
b.z = floor( a.z )
```

Example: b = floor(a);

mag

Returns the scalar magnitude of a triplex number. This operation is sometimes referred to as the triplex modulus or triplex absolute value. In math formulas it is often written as |a|.

```
mag(a)
b.x = sqrt(a.x * a.x + a.y * a.y + a.z * a.z)
b.y = 0
b.z = 0
```

Example: b = mag(a);

magsqr

Returns the scalar magnitude of a triplex number. This operation is sometimes referred to as the triplex modulus squared or triplex absolute value squared. In math formulas it is often written as |a|^2. This function executes slightly faster than the mag function since it has no square root.

```
magsqr(a)
b.x = a.x * a.x + a.y * a.y + a.z * a.z
b.y = 0
b.z = 0
```

Example: b = magsqr(a);

pow

Raises the triplex number a to the power of the x component of triplex number b and returns a triplex number. See the ^ operation above for a code description.

```
pow(a , b)
// See ^
```

Example: c = pow(a , b);

round

Returns the round of the individual components of a triplex number.

```
round(a)
b.x = round( a.x )
b.y = round( a.y )
b.z = round( a.z )
```

Example: b = round(a);

trunc

Returns the trunc of the individual components of a triplex number.

```
trunc(a)
b.x = trunc( a.x )
b.y = trunc( a.y )
b.z = trunc( a.z )
```

Example: b = trunc(a);

sqrt

Returns the square root of a triplex number. This is implemented by raising a triplex number to the exponent of 0.5 using the power operation.

```
sqrt(a)
b = a^0.5 // See ^
```

Example b = sqrt(a);

13. Trigonometric Functions

Trigonometric functions work with scalar triplex numbers only. All trigonometric functions return 0 when used with non scalar numbers. The trigonometric functions are: acos acosh asin asinh atan atan2 atanh cos cosh exp log log10 sin sinh tan tanh.

14. Triplex Functions

triplex

Returns a triplex number from the x component of three triplex numbers.

```
triplex(a,b,c)
d.x = a.x
d.y = b.x
d.z = c.x
```

```
Example
a = (1,2,3);
b = (4,5,6);
c = (7,8,9);
d = triplex(a,b,c); // d is (1,4,7)
```

xcomponent

Returns the x component of a triplex number as a scalar triplex number.

```
xcomponent(a)
b.x = a.x
b.y = 0
b.z = 0
```

```
Example
a = (1,2,3)
b = xcomponent( a ); // b is (1,0,0)
```

ycomponent

Returns the y component of a triplex number as a scalar triplex number.

```
ycomponent(a)
b.x = a.y
b.y = 0
b.z = 0
```

```
Example
a = (1,2,3)
b = ycomponent( a ); // b is (2,0,0)
```

zcomponent

Returns the z component of a triplex number as a scalar triplex number.

zcomponent(a)
b.x = a.z
b.y = 0
b.z = 0

Example
a = (1,2,3)
b = zcomponent(a); // b is (3,0,0)

15. Precedence:

( )

Changes the precedence of an expresssion.

Example: a = 2 * (1 + 3); // a is (8,0,0)

16. Control Flow

if, if else, and else are used to allow logical control of program execution.

if

The if operator must be followed by an expression and one or more statements enclosed by brackets. If the expression is not equal to zero the expression is true and the statements are executed. Unlike C, the brackets are required around statements.

if(expression) {one or more statements}

else if

The else if operator is executed only if the previous if or else if operators were false. The else if operator must be followed by an expression and one or more statements enclosed by brackets. If the expression is not equal to zero the expression is true and the statements are executed.

else if(expression) {one or more statements}

else

The else operator is executed only if the previous if or else if operators were false. The else operator must be followed by one or more statements enclosed by brackets.

else {one or more statements}

Example

```
if( a < 5) {
a = 0;
b = b + 1;
}
else if( a > 10){
a = 0;
}
else {
a = 10;
b = 0;
}
```