daScript Standard Library

Release 0.1 alpha

Anton Yudintsev

Nov 16, 2019

CONTENTS

CHAPTER

ONE

INTRODUCTION

The daScript standard libraries consist in a set of modules implemented in C++. While are not essential for the language, they provide a set of useful services that are commonly used by a wide range of applications (file I/O, regular expressions, etc.), plus they offer a foundation for developing additional libraries.

All libraries are implemented through the daScript API and C++ runtime library. The modules are organized in the following way:

- fio : input and output
- math : basic mathematical routines
- random : LCG random mathematical routines
- string : string library

CHAPTER

TWO

THE MATH LIBRARY

Floating point math in general is not bit-precise. Compiler can optimize permutations, replace divisions with multiplications, and some of functions are not bit-exact. If you need precise math use double precision type. All functions and symbols are in "math" module, use require to get access to it.

```
require math
```

2.1 Common Symbols

2.1.1 all numerics (uint*, int*, float*, double)

min (*x*, *y*)
: returns the minimum of *x* and *y*

max (*x*, *y*)
: returns the minimum of *x* and *y*

2.1.2 float* and double

abs (*x*)
: returns the absolute value of *x* as same time

acos (*x*)
: returns the arccosine of *x*

asin (*x*)
: returns the arcsine of *x*

atan (*x*)
: returns the arctangent of *x*

atan2 (*x*, *y*)
: returns the arctangent of *x/y*

cos (*x*)
: returns the cosine of *x*

sin (*x*)
: returns the sine of *x*

tan (*x*)
: returns the tangent of *x*

exp (*x*)
returns the exponential value of the float parameter *x*

log (*x*)
returns the natural logarithm of *x*

exp2 (*x*)
returns the 2^x value of the float parameter *x*

log2 (*x*)
returns the logarithm base-2 of *x*

pow (*x*, *y*)
returns *x* raised to the power of *y*

sqrt (*x*)
returns the square root of *x*

rcp (*x*)
returns the 1/x of *x*

PI
The numeric constant pi (3.141592) is the ratio of the circumference of a circle to its diameter

ceil (*x*)
returns a float value representing the smallest integer (type is still float) that is greater than or equal to *x*

floor (*x*)
returns a float value representing the largest integer that is less than or equal to *x*

abs (*x*)
returns a positive value representing with same module as *x*

saturate (*x*)
returns a clamped to [0..1] inclusive range *x*

2.1.3 float* only

atan2_est (*x*, *y*)
returns the faster approximation of arctangent of *x/y* - float only

rcp_est (*x*)
returns the fast approximation 1/x of *x* - float only

ceili (*x*)
returns a value representing the smallest integer (integer type!) that is greater than or equal to *x*

floori (*x*)
returns a integer value representing the largest integer that is less than or equal to *x*

roundi (*x*)
returns a integer value representing the integer that is closest to *x*

trunci (*x*)
returns a integer value representing the float without fraction part of *x*

2.2 Noise functions

uint32_hash (*x:uint*)
returns hashed value of *x*

uint_noise_1D (*position: int; seed:uint*)
returns noise value of *position* in *seed* sequence

uint_noise_2D (*x, y: int; seed:uint*)
returns noise value of *x,y* position in *seed* sequence

uint_noise_3D (*x, y, z: int; seed:uint*)
returns noise value of *x,y,z* position in *seed* sequence

2.3 Vector functions

2.3.1 float2, float3, float4:

length (*x*)
returns a non-negative value representing magnitude of *x*

length_sq (*x*)
returns a non-negative value representing squared magnitude of *x*

inv_length (*x*)
returns a non-negative value representing 1/magnitude of *x*

inv_length_sq (*x*)
returns a non-negative value representing 1/squared magnitude of *x*

distance (*x, y*)
returns a non-negative value representing distance between *x* and *y*

distance_sq (*x*)
returns a non-negative value representing squared distance between *x* and *y*

inv_distance (*x, y*)
returns a non-negative value representing 1/distance between *x* and *y*

inv_distance_sq (*x*)
returns a non-negative value representing 1/squared distance between *x* and *y*

dot (*x, y*)
returns scalar representating dot product between *x* and *y*

normalize (*x*)
returns normalized *x*, or nan if length(x) is 0

safe_normalize (*x*)
returns normalized *x*, or 0 if length(x) is 0

2.3.2 float3 only:

cross (*x, y*)
returns vector representating cross product between *x* and *y*

reflect (*v*, *n*)
returns vector representating reflection of *x* from *n* same as

```
def reflect(v, n: float3)
    return v - 2. * dot(v, n) * n
```

refract (*v*, *n*)
returns vector representating reflection of *x* from *n* same as

```
def refract(v, n: float3; nint: float; outRefracted: float3&)
    let dt = dot(v, n)
    let discr = 1. - nint * nint * (1. - dt * dt)
    if discr > 0.
        outRefracted = nint * (v - n * dt) - n * sqrt(discr)
        return true
    return false
```

2.4 lerp/madd/clamp

lerp (*a*, *b*, *t*)
returns vector or scalar representating *a* + (*b* - *a*) * *t*

madd (*a*, *b*, *c*)
returns vector or scalar representating *a* * *b* + *c*

clamp (*t*, *a*, *b*)
returns vector or scalar representating min(max(*t*, *a*), *b*)

CHAPTER

THREE

THE SYSTEM LIBRARY

The system library exposes operating system facilities like environment variables, date time manipulation etc.

3.1 Global Symbols

clock - builtin type. You can use clock-clock to get difference of time in seconds (float). clock type can be printed and string built.

getClock()
: returns current clock (current time)

CHAPTER

FOUR

THE STRING LIBRARY

the string lib implements string formatting and regular expression matching routines.

4.1 Global Symbols

ends_with (*str*, *cmp*)
returns *true* if the end of the string *str* matches a the string *cmp* otherwise returns *false*

starts_with (*str*, *cmp*)
returns *true* if the beginning of the string *str* matches the string *cmp*; otherwise returns *false*

strip_left (*str*)
Strips white-space-only characters that might appear at the beginning of the given string and returns the new stripped string.

strip_right (*str*)
Strips white-space-only characters that might appear at the end of the given string and returns the new stripped string.

strip (*str*)
Strips white-space-only characters that might appear at the beginning or end of the given string and returns the new stripped string.

find (*str*, *substr*[, *at*])
Return index where substr czn be found within str (starting from optional at), or -1 if not found

length (*str*)
Return length of string

reverse (*str*)
Return reversed string

to_lower (*str*)
Return all lower case string

to_upper (*str*)
Return all upper case string

slice (*str*, *start*[, *end*])
Return all part of the strings starting at start and ending by end. Start can be negative (-1 means "1 from the end").

4.2 String builder

CHAPTER

FIVE

THE BLOB LIBRARY

The blob library implements binary data manipulations routines. The library is based on *blob objects* that represent a buffer of arbitrary binary data.

5.1 daScript API

5.1.1 Global symbols

binary_save (*f*, *block*<*blob: string*>)
: saves f to returned blob (provided to a block)

binary_load (*var obj; blob: string*)
: loads obj from blob

5.1.2 The blob

The blob object is a buffer of arbitrary binary data, which still is typed as string.

CHAPTER

SIX

THE INPUT/OUTPUT LIBRARY

the i/o library implements basic input/output routines.

todo: to be written

CHAPTER

SEVEN

THE LCG RANDOM LIBRARY

the random library implements basic random routines.

7.1 Unit functions

int4 random_seed(*seed:int32*)
constructs seed vector out of single integer seed

int random_int(*seed:int4*)
random integer 0..32767

int4 random_int4(*seed:int4*)
random int4, each component is 0..32767

float random_float(*seed:int4*)
random float 0..1

float4 random_float4(*seed:int4*)
random float4, each component is 0..1

7.2 Distribution functions

float3 random_unit_vector(*seed:int4*)
random float3 unit vector (length=1)

float3 random_in_unit_sphere(*seed:int4*)
random float3 unit vector (length=1) which happens to be inside a sphere R=1

float3 random_in_unit_disk(*seed:int4*)
random float3 unit vector (length=1) which happens to be inside a disk R=1, Z=0