

Torque Game Builder – TorqueScript Reference

(Adapted from Edward Maurina's book, "The Game Programmer's Guide to Torque" [GPGT]. Grab the entire book to take your Torque knowledge to the next level.)

Operators

Operator	Name	Example	Explanation
----------	------	---------	-------------

Variable Operators			
\$	global	\$a	\$a is a global variable.
%	local	%b	%b is a local variable.

Arithmetic Operators			
*	multiplication	\$a * \$b	Multiply \$a and \$b.
/	division	\$a / \$b	Divide \$a by \$b.
%	modulo	\$a % \$b	Remainder of \$a divided by \$b.
+	addition	\$a + \$b	Add \$a and \$b.
-	subtraction	\$a - \$b	Subtract \$b from \$a.
++	auto-increment (post-fix only)	\$a++	Increment \$a after use. Note: ++\$a is illegal.
--	auto-decrement (post-fix only)	\$b--	Decrement \$b after use. Note: --\$b is illegal.

Relations and Logical Operators			
<	Less than	\$a < \$b	1 if \$a is less than \$b (0 otherwise.)
>	More than	\$a > \$b	1 if \$a is greater than \$b (0 otherwise.)
<=	Less than or Equal to	\$a <= \$b	1 if \$a is less than or equal to \$b (0 otherwise.)
>=	More than or Equal to	\$a >= \$b	1 if \$a is greater than or equal to \$b (0 otherwise.)
==	Equal to	\$a == \$b	1 if \$a is equal to \$b (0 otherwise.)
!=	Not equal to	\$a != \$b	1 if \$a is not equal to \$b (0 otherwise.)
!	Logical NOT	!\$a	1 if \$a is 0 (0 otherwise.)
&&	Logical AND	\$a && \$b	1 if \$a and \$b are both non-zero (0 otherwise.)
	Logical OR	\$a \$b	1 if either \$a or \$b is non-zero (0 otherwise.)

Torque Game Builder – TorqueScript Reference

Bitwise Operators			
~	Bitwise complement	~\$a	flip bits 1 to 0 and 0 to 1. (i.e. ~10b == 01b)
&	Bitwise AND	\$a & \$b	composite of elements where bits in same position are 1. (i.e. 1b & 1b == 1b)
	Bitwise OR	\$a \$b	composite of elements where bits 1 in either of the two elements. (i.e. 100b & 001b == 101b)
^	Bitwise XOR	\$a ^ \$b	composite of elements where bits in same position are opposite. (i.e. 100b & 101b == 001b)
<<	Left Shift	\$a << 3	element shifted left by 3 and padded with zeros. (i.e. 11b << 3d == 11000b)
>>	Right Shift	\$a >> 3	element shifted right by 3 and padded with zeros. (i.e. 11010b >> 3d == 00011b)

Assignment Operators			
=	Assignment	\$a = \$b;	Assign value of \$b to \$a.
op=	Compound Assignment	\$a op= \$b;	Equivalent to \$a = \$a op \$b. op can be any of: * / % + - & ^ << >>

String Operators / Constants			
@	String catenation	\$c @ \$d	Concatenates strings \$c and \$d into a single string. Numeric literals/variables convert to strings.
NL	New Line	\$c NL \$d	Same as catenation example with new-line between \$c and \$d.
TAB	Tab	\$c TAB \$d	Same as catenation example with tab between \$c and \$d.
SPC	Space	\$c SPC \$d	Same as catenation example with space between \$c and \$d.
\$=	String equal to	\$c \$= \$d	1 if \$c equal to \$d .
!\$=	String not equal to	\$c !\$= \$d	1 if \$c not equal to \$d.

Torque Game Builder – TorqueScript Reference

Miscellaneous Operators			
<code>? :</code>	Conditional	<code>x ? y : z</code>	Substitute y if x equal to 1, else substitute z.
<code>[]</code>	Array element	<code>\$a[5]</code>	Sixth element or array \$a
<code>.</code>	Field/Method selection	<code>%obj.field</code> <code>%obj.method()</code>	Select a console method or field
<code>()</code>	Grouping	--	--
<code>{ }</code>	Blocking	--	--
<code>,</code>	Listing	--	--
<code>::</code>	Namespace	<code>Item::onCollision()</code>	This definition of the onCollision() function is in the Item namespace.
<code>" "</code>	String constant (normal)	<code>"Hello world"</code>	This is a normal string.
<code>' '</code>	String constant (tagged)	<code>'Torque Rocks'</code>	This is a tagged string. The value of a tagged string is sent only once to a client from the server (across the network). Subsequently, only the 'tag' is sent. All clients will store the string at and index identified by the 'tag' and can look up the value instead of requiring it be sent repeatedly.
<code>//</code>	Single line comment	<code>// This is a comment</code>	Used to comment out a single line of TS.
<code>/* */</code>	Multi-line comment	<code>/*This is a multi-line comment*/</code>	Used to comment out multiple consecutive lines. /* opens the comment, and */ closes it.

Torque Game Builder – TorqueScript Reference

Literals

Numbers	123	integer
	1.23	floating-point
	1.00E-003	scientific notation
	0xabc	hexadecimal

String	"abcd"	string
	'abcd'	tagged string

String Operators	@	concatenation
	TAB	tab concatenation
	SPC	space concatenation
	NL	newline concatenation

Torque Game Builder – TorqueScript Reference

Escape Sequences	\n	newline
	\r	carriage return
	\t	tab
	\c0 .. \c9	colorize subsequent console output
	\cr	reset to default color
	\cp	push color to color stack
	\co	pop color from color stack
	\xhh	wo-digit hex value ASCII code
	\\	Backslash

booleans	TRUE	1
	FALSE	0

arrays	\$MyArray[n]	single-dimension
	\$MyMultiArray[n,m]	multi-dimension
	\$MyMultiArrayn_m	multi-dimension

vectors	"1.0 2.0 1.0 2.0"	4-element vector
----------------	-------------------	------------------

Torque Game Builder – TorqueScript Reference

Keywords

Note: Although keywords are not reserved, it is considered bad practice to use variables that have the same spelling as a keyword.

- **break** - Use *break* to exit the innermost *for* or *while* loop. *break* can also be used to exit a *switch* or *switch\$* statement, though it is not necessary to use *break* statements to separate *case* statements as in C/C++.
- **case** - Used to label cases in a *switch* or *switch\$* statement.
- **continue** - The *continue* keyword causes the script to skip the remainder of the innermost loop in which it appears and continue execution with the next iteration of the loop.
- **default** - This labels the default case in a *switch* or *switch\$* statement. The default case is the case that is executed if no other cases match the *switch/switch\$* value.
- **else** - The *else* keyword is used with the *if* keyword to control the flow of a script. The block of code after an *else* statement is executed if the *if* condition evaluates to *false*.
- **false** - The *false* keyword is used for boolean comparison and evaluates to 0.
- **for** - The *for* keyword starts a loop that will execute a section of code multiple times until an exit condition is met.
- **function** - The *function* keyword is used to define a new console function or method.
- **if** - The *if* keyword is used with or without the *else* keyword to control the flow of a script. The block of code after an *if* statement is executed if the *if* condition evaluates to *true*.
- **new** - The *new* keyword is used to create an instance of a class.
- **package** - The *package* keyword tells the console that the subsequent block of code is to be declared but not loaded. Packages provide dynamic function-polymorphism in TorqueScript.
- **parent** - The *Parent* keyword is used with the namespace operator (::) to reference the previous definition of a function that has been over-ridden either through inheritance or packaging.
- **return** – The *return* keyword is used to return a value from a function.
- **switch** - The *switch* keyword begins a switch statement that is used to control the flow of script execution. Execution will be directed to the block of code after the case statement that matches the switch value.
- **switch\$** - The *switch\$* keyword is identical to *switch* with the exception that it is used exclusively for strings.
- **true** - The *true* keyword is used for boolean comparison and evaluates to 1.
- **while** - The *while* keyword starts a loop that will execute a section of code multiple times until an exit condition is met.