

# SSK

(for Corona SDK)

# Users' Guide

Written by Ed Maurina (RoamingGamer)  
Roaming Gamer, LLC.

Version 1.1  
Last modified: May 11, 2015

**Table of Contents**

About SSK.....3

Basic Usage.....4

Tip: Localize For Speedup and Simplicity.....5

    Long Hand SSK.....5

    Localized SSK.....5

Credits Note.....6

SSK License.....7

## About SSK

SSK is short for (Super Starter Kit). It has its roots in a similarly named software kit I wrote and maintained for the Torque Game Engines between 2001 and 2010.

In 2012 I started using the Corona SDK and decided to transfer my (Torque) SSK learnings to (Corona) SSK.

SSK is a collection of LUA libraries and modules, including:

- **Core** – The Core SSK libraries and modules (freely available to all)
  - **Globals** – A small set of global variables and functions used to simplify and speed up Corona coding.
  - **Extensions** – Adds new and improved features to existing LUA and Corona SDK libraries/modules (display, io, math, native, string, table, transition).
  - **Display** – A powerful replacement for the `display.new*()` functions. It utilizes a concise syntax that enables one line creation of complex display objects w/ or w/o physics bodies.
  - **Collision Calculator** – Never do collision body math again. Simply use names and simple rules.
  - **2D Math** – A complete 2D math library.
  - **Easy Interfaces** – A collection of easy push/toggle/radio/slider button builders and callback generators, including persistence features.
  - **More...** - Much more.
- **Free** – A collection of non-core features which I've decided to release for free to the community. These are generally answers to interesting Forums questions.
- **Paid** – An ever growing of low-priced add-ons solving specific app and game design tasks, and designed to save you hours of frustrating development work.

## Basic Usage

To use SSK, do the following:

1. Download the latest version of SSK: <https://github.com/roaminggamer/SSKCorona>
2. Create a new app or project folder including this minimum set of files:
  - main.lua – Main entry point into your app/game.
  - config.lua – Used to set up scaling rules and design target resolution.
    - <http://docs.coronalabs.com/daily/guide/basics/configSettings/index.html>
  - build.settings – Used to set up device specific rules and to include plugins, etc.
    - <http://docs.coronalabs.com/guide/distribution/buildSettings/index.html>
3. Copy ssk into your project and place it in your project folder
4. Add this line near the top of your main.lua file

```
require "ssk.loadSSK"
```

Done!

## Tip: Localize For Speedup and Simplicity

As you start to use SSK, you will find that you can get better performance and reduce the effort it takes to type/code by localizing features when you use them.

Please compare these two versions of code to see what I mean.

*(Note: The examples below are very simple and are not meant to demonstrate the power of SSK. I am merely demonstrating localization.)*

### Long Hand SSK

```
ssk.display.newCircle( group, 100, 100, { fill = {1,0,0} } )  
ssk.display.newRect( group, 200, 200, { fill = {1,0,0} } )  
ssk.display.newImageRect( group, 300, 300, "images/smiley.png" )
```

### Localized SSK

At the top of a file using SSK features do this:

```
local newCircle      = ssk.display.newCircle  
local newRect        = ssk.display.newRect  
local newImageRect   = ssk.display.newImageRect
```

Now, later in the same file, you can do this:

```
newCircle( group, 100, 100, { fill = {1,0,0} } )  
newRect( group, 200, 200, { fill = {1,0,0} } )  
newImageRect( group, 300, 300, "images/smiley.png" )
```

The code above will be both faster to execute, and easier to type.

## Credits Note

For the most part, SSK is my own creation, however I am not above 'stealing with pride'. That is, if I see a beautiful solution implemented by someone else and freely available I will occasionally incorporate said solution into SSK with or without modification. In each case, I give credit in the source file(s) and insert any required license statements.

## SSK License

I want you to use SSK to make your games and apps more quickly and better.

However I do ask that you respect the work that went into making this and not try to sell it as a standalone product or as part of a collection or book.

In a nutshell:

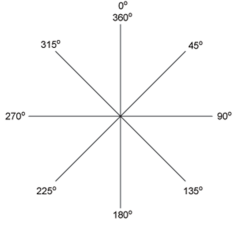
- SSK is free to use.
- SSK is free to edit.
- SSK is free to use in a free or commercial game.
- SSK is free to use in a free or commercial non-game app.
- SSK is free to use without crediting the author (credits are still appreciated).
- SSK is free to use without crediting the project (credits are still appreciated).
- SSK is NOT free to sell for anything.
- SSK is NOT free to credit yourself with.

***(Note: The cheat sheets have been inserted into this document, starting on the next page, but if you want them as separate images, please look in the directory where you found this document.)***

Legend	Global Functions
bp – Table of body parameters g – display group myCC – SSK collision calculator instance obj – display object pts – SSK points list value – value (type varies) vp – Table of visual parameters	<b>Various Helpers</b> value fnn( ... ) – First not nil. bool isDisplayObject( obj ) [legacy] float round( val [ , n ] ) nextFrame( func [ , delay ] )  <b>Improved Runtime:* Shorthand</b> listen( eventName, listener ) ignore( eventName, listener ) autoIgnore( eventName, obj ) post( eventName [ , params [ , debuglvl ] ] )  <b>Listener Management</b> removeListeners( [ obj ] )  <b>Colors</b> t = rgba2( { r, g, b, a } ) - Graphics 1.0 to 2.0 t = hexcolor( code ) - "RRGGBBAA", "0x ...", "# ..." t = randomColor()  <b>display.setDefault() Stacking</b> pushDisplayDefault( defaultName, newValue ) popDisplayDefault( defaultName )
Globals	
<b>Meaurement and Spacing</b> w, h, fullw, fullh, centerX, centerY, unusedWidth, unusedHeight, deviceWidth, deviceHeight, left, top, right, bottom, orientation, isLandscape, isPortrait  <b>Environment</b> onSimulator, oniOS, onAndroid, onOSX, onWin  <b>Device</b> oniPad, oniPhone4, oniPhone5, oniPhone5s, oniPhone6, oniPhone6Plus, onAndroidTablet, onTablet  <b>Easy Colors (predefined color codes)</b> _TRANSPARENT_, _WHITE_, _BLACK_, _GREY_, _DARKGREY_, _DARKGREY_, _DARKERGREY_, _LIGHTGREY_, _RED_, _GREEN_, _BLUE_, _CYAN_, _YELLOW_, _ORANGE_, _ORANGE_, _BRIGHTORANGE_, _PURPLE_, _PINK_, _T_, _W_, _K_, _R_, _G_, _B_, _Y_, _O_, _P_, _C_	
Collision Calculator (ssk.ccmgr.*)	
myCC = newCalculator() <div>             myCC:addName( colliderName )              myCC:addNames( ... )              myCC:collidesWith( colliderName, ... )              myCC:dump()              myCC:getCategoryBits( colliderName )              myCC:getCollisionFilter( colliderName )              myCC:getMaskBits( colliderName )           </div>	
(Extended Display Objects) ssk.display.*	
<b>Quick Layers</b> layers = quickLayers( g, ... ) <div>             layers:destroy()              layers:purge( layerName )           </div>	
<b>Object Builders</b> obj newRect( g, x, y [ , vp [ , bp ] ] ) obj newCircle( g, x, y [ , vp [ , bp ] ] ) obj newImage( g, x, y [ , path, vp [ , bp ] ] ) obj newImageRect( g, x, y [ , path, vp [ , bp ] ] ) obj newSprite( g, x, y, path, seqdat [ , vp [ , bp ] ] )	
<b>Default Physics Parameters</b> value = getDPP( name ) setDPP( name, value ) listDPP()	
	<b>Object Builder Visual Parameters (vp)</b>  <ul style="list-style-type: none"> <li>All legal display.* fields (properties)               <ul style="list-style-type: none"> <li><a href="http://docs.coronalabs.com/daily/api/library/display/index.html#properties">http://docs.coronalabs.com/daily/api/library/display/index.html#properties</a></li> </ul> </li> <li>fill - Color table OR fill (gradient, composite, image )</li> <li>stroke - Color table OR fill (gradient, composite, image )</li> <li>'listener' – Assign a function to any of these listener names and it will be automatically listened for: accelerometer, audio, axis, collision, colorSample, completion, enterFrame, fbconnect, finalize, gameNetwork, gyroscope, heading, inputDeviceStatus, key, licensing, location, mapAddress, mapLocation, mapMarker, mapTap, memoryWarning, mouse, networkRequest, networkStatus, notification, orientation, particleCollision, popup, postCollision, preCollision, productList, resize, scene, sprite, storeTransaction, system, tap, timer, touch, unhandledError, urlRequest, userInput.</li> </ul>
	<b>Object Builder Body Parameters (bp)</b>  <ul style="list-style-type: none"> <li>bodyType, angularDamping, isBodyActive, isBullet, gravityScale, isFixedRotation, isSensor, isSleepingAllowed, linearDamping, density, radius, angularVelocity</li> <li>calculator – Reference to collision calculator.</li> <li>colliderName – Collider name for this object (as specified in 'collision calculator')</li> </ul>
	<b>Lines (See code for visual params)</b> obj newAngleLine( g, x1, y1, angle, len, vp ) obj newLine( g, x1, y1, x2, y2, vp ) obj newPointsLine( g, pts, vp )

Updated: 10 MAY 2015



Extensions	RGMath2D (ssk.math2d.*)
<p><b>io.*</b>  bool io.exists( fileName [ , base ] )  string io.readFile( fileName [ , base ] )  io.writeFile( dataToWrite, fileName [ , base ] )  io.appendFile( dataToWrite, fileName [ , base ] )  table io.readFileTable( fileName [ , base ] )</p> <p><b>string (all return string; unless otherwise specified)</b>  string.comma_value( val [ , n ] )  string.endswith( s, send )  string.first_upper( str )  string:lpad( len, char )  string:merge( t )  string:rpadd( len, char )  string.shorten( text, maxLen, appendMe )  string:spaces2underbars( )  table string:split( tok )  string.startswith( s, piece )  string.trim( s )  string:underbars2spaces( )  (Words are separated by a single space (' '))  string:getWord( index )  integer string:getWordCount( )  string:getWords( index, endIndex )  string:setWord( index , replace )  (Fields are separated by a single TAB ('\t'))  string:getField( index )  integer string:getFieldCount( )  string:getFields( index, endIndex )  string:setField( index , replace )  (Records are separated by a single NEWLINE ('\n')).  string:getRecord( index )  integer string:getRecordCount( )  string:getRecords( index, endIndex )  string:setRecord( index , replace )</p> <p><b>table (all return table unless otherwise specified)</b>  integer table.count( src )  integer table.count_r( src )  none table.dump(theTable, padding, marker )  none table.dumpu( theTable [ , padding ] )  table.combineUnique( ... )  table.combineUnique_i( ... )  table.deepCopy( src [ , dst ] )  table.deepStripCopy( src [ , dst ] )  value table.getRandom( )  table.load( fileName [ , base ] )  value table.permute_iter( a )  none table.print_r( theTable )  table.removeByRef( t, obj )  table.repairIndices( theTable )  none table.save( theTable, fileName [ , base ] )  table.shallowCopy( src [ , dst ] )  table.shuffle( t [ , iter ] )</p> <p><b>transition</b>  transition.fromtocolor( obj, fromC, toC, time, delay, ease )</p>	<p><b>Flexible (Various Input &amp; Output Styles)</b>  add( objA, objB [ , altRet ] )  add( x1, y1, x2, y2, [ , altRet ] )  sub( objA, objB [ , altRet ] )  sub( x1, y1, x2, y2, [ , altRet ] )  dot( objA, objB )  dot( x1, y1, x2, y2 )  length( objA )  length( x1, y1 )  length2( objA )  length2( x1, y1 )  scale( objA, scale [ , altRet ] )  scale( x1, y1, scale, [ , altRet ] )  normalize( objA [ , altRet ] )  normalize( x1, y1 [ , altRet ] )  normals( objA [ , altRet ] )  normals( x1, y1 [ , altRet ] )  vector2Angle( objA )  vector2Angle( x1, y1 )  angle2Vector( angle [ , altRet ] )  cartesian2Screen( objA [ , altRet ] )  cartesian2Screen( x1, y1 [ , altRet ] )  screen2Cartesian( objA [ , altRet ] ) or ( x1, y1 [ , altRet ] )  screen2Cartesian( x1, y1 [ , altRet ] )</p> <p><b>Fast (Fixed Input/Output Signature For Speedup)</b>  x, y addFast( x1, y1, x2, y2 )  x, y subFast( x1, y1, x2, y2 )  float dotFast( x1, y1, x2, y2 )  float lengthFast( x, y )  float length2Fast( x, y )  x, y scaleFast( x, y, scale )  x, y normalizeFast( x, y )  x1, y1, x2, y2 normalsFast( x, y )  float vector2AngleFast( x, y )  x, y, angle2VectorFast( angle )</p>  <p><b>RGPoints (ssk.points.*)</b></p> <p>pointsInstance = new( ... )</p> <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"> pointsInstance.add( x1, y1, ... )  pointsInstance.insert( index, x1, y1, ... )  pointsInstance.get( index )  pointsInstance.remove( index )  pointsInstance.push( x1, y1, ... )  pointsInstance.peek()  pointsInstance.pop()  pointsInstance.push_head( x1, y1, ... )  pointsInstance.peek_head()  pointsInstance.pop_head() </div> <p><b>math</b>  math.calculateWrapPoint( objectToWrap, wrapRectangle )  math.getUID( uidLen )  math.haversine_dist( lat1, lng1, lat2, lng2 [ , R ] )  math.pointInRect( pointX, pointY, left, top, width, height )</p>

Updated: 10 MAY 2015

RGEasyBench (ssk.easyBench.*)
measureTime( func, iter ) measureABTime( func1, func2, iter ) getMemCount( collect )
RGAndroid (ssk.android.*)
captureBackButton( noCB, yesCB ) captureVolumeButtons( block, volUp, volDown ) easyAndroidUIVisibility( profile )
RGMisc (ssk.misc.*) [may move later]
blockTouchesForDuration( duration, subtle ) createEasyMeter( x, y, width, fontSize ) easyAlert( title, msg, buttons ) buttons = { { "button 1", opt_func1 }, { "button 2", opt_func2 }, ... }  easyBlur( group, time, color ) easyRemoteImage( curlmg, fileName, imageURL, base ) easyShake( obj, amplitude, time ) easyUnderline( obj, color, strokeWidth, extraW, yOffset ) fitText( obj, origText, maxWidth ) getImageSize( path, basePath ) isConnectedToWWW( [ url ] ) isValidEmail( val, debugEn ) noErrorAlerts() normRot( obj ) normRot2( rotation ) obottom( obj ) ohcenter( obj ) oleft( obj ) oright( obj ) otop( obj ) ovcenter( obj ) protoDim( group, obj, fontSize, color ) quickLine( group, x, y, len, color, strokeWidth, yOffset ) rotateAbout( obj, x, y, params ) secondsToTimer( seconds, version )

Updated: 10 MAY 2015