I am not as good in documentation, here is a try.

libsota was made to help other to create their own mods for Shroud. Also libsota contains some workarounds to take away some headache from working with the Shroud Api.
libsota is object oriented and provides events and some additional data produced from data that is coming from Shroud Api.
It does data polling and generate additional events from them and caches these data, so that calls to the client are reduced.
All the mods that are using libsota sharing the data already retrieved.
It is also carefully with callbacks coming from the Shroud Api that may produce lag on the client or may causes trouble with user input or system messages.
Some counter measures are taken to not block/hindering the player from playing the game because of LUA.

Labels are auto hidden during loading a scene, but can also be set to be shown on load screen

libsota is still in development and when finished it is made from 3 files:
libsota.lua - the core library interacting directly with the Shroud Api
libsota.util.lua - helper functions for the global namespace; procedure style.
libsota.ui.lua - UI controls like windows, labels, window manager, images, buttons, ...

Even when libsota is object oriented and likes closures it can also be used with a procedural coding style.

I would be happy when the one or other small or big mod appears here in the forum, based on libsota.
Some here have good ideas for small goodies. I do not have so many ideas which mods may needed / wanted.

Here are the objects:
(when libsota and libsota.util are installed all objects can be outputted to the console with \info player, \info client, \info scene, \info ui)

libsota.0.4.5:
[CODE]

client = {
	timeStarted = timestamp: time when the game was started

timeToLoad = int: duration in seconds needed to load into the first scene
timeInGame = int: duration in seconds already playing the game
timeDelta = float: ShroudTimeDelta
fps = int: fps
accuracy = float: time drift between one second
screen = {
	width = int: width of screen in pixel
	height = int: height of screen in pixel
	isFullScreen = bool: true if played in full screen mode
},
isHitching = bool: true when the client hitches
isLoading = bool: true when the client is loading a scene
api = {
	luaVersion = string: contains the version number of LUA
	luaPath = string: contains the path of the lua directory (/lua path)
	list = table: contains all available Shroud Api functions of the current client
	isImplemented = function(<shroudfunctionname>) - returns true when this function is available, false otherwise
},
mouse = {
	button = table: containing button states
	x = int: x position of mouse on screen
	y = int: y position of mouse on screen
},
window = {
	paperdoll = {
		open = bool: true when paperdoll/charactersheet is open
		left = int: left pos of this window
		top = int: top pos of this window
		width = int: with of this window
		height = int: height of this window
	}
}

```
}
scene = {
	name = string: contains the display name of the current scene
	maxPlayer = int: contains the number of player allowed in this scene
	isPvp = bool: true if PvP is allowed in this scene
	isPot = bool: true if the current scene is a POT
	timeInScene = float: duration in seconds already in this scene
	timeToLoad = float: duration in seconds needed to load into this scene
	timeStarted = timestamp: time when this scene was loaded / entered
}
player = {
	caption = string: contains the caption of the player like it is shown on screen
	name = string: contains the name of the player (without any flags)
	flag = string: contains the flags the player currently has
	isPvp = bool: true when player is flagged for PvP
	isAfk = bool: true when player is AFK
	isGod = bool: true when the player is in god mode
	isMoving = bool: true when the player is moving
	isStill = bool: true when the player has stillness bonus
	lastMoved = timestamp: time when player last moved (or time since the player is standing)
	location = {
		x = float: x position of player in scene
		y = float: y position of player in scene (is height)
		z = float: z position of player in scene
		scene = points to the scene object
	},
	health = {
		current = float: current health of the player
		max = float: current max health of the player
		percentage = float: current percentage of max health (may used for progress bars)
	},
	focus = {
```

```
		current = float: current focus of the player
		max = float: current max focus of the player
		percentage = float: current percentage of max focus (may used for progress bars)
	},
	xp = {
		producer = number: current pooled producer xp
		adventurer = number: current pooled adventurer xp
	},
	inventory = array: a array that contains a table with this fields: {
		name = string: caption of the item
		durability = float: durability
		primaryDurability = float:
		maxDurability = float:
		weight = float:
		quantity = int:
		value = int:
	}
	stat = function(index) - returns a table containing: number, name, value and description of stat. Index is the name or number of the stat
}

ui = {
	version = string: contains the version of the library

	timer = {
		add = function(timeout, once, callback, ...) - add a callback that is called after/every (the) time slice. returns the index
		get = function(index) - returns a timer instance
		remove = function(index) - removes a timer instance
		enabled = function(index, enabled) - enable or disable a timer. returns the enabled state. if enabled is nil, only return
		pause = function(index) - pauses a timer (enabled = false)
		resume = function(index) - resumes a timer (enabled = true)
		toggle = function(index) - toggles the enabled state of a timer
	},
```

setTimeout = function(timeout, callback) - utility function to call a function after the timeout
setInterval = function(interval, callback) - utility function to call a function periodical (interval)

handler = {
add = function(name, callback) - adds a named callback handler / event you can use to provide events to other or your scripts. returns index
remove = function(index) - removes a callback handler
invoke = function(name, ...) - invokes a named callback handler
},
onInit = function(callback) - this event is invoked when the Shroud Api and the lib is fully initialized
onStart = function(callback) - this event is invoked when all mods are initialized that are using this library
onUpdate = function(callback) - this event is invoked on every frame. Please use timer instead.
onConsoleInput = function(callback) - this event is invoked for each line reaching the chat window, except it is a command
onConsoleCommand = function(callback) - this event is invoked when a command is entered in the chat window. Commands starting with \, _ or !
onSceneChanged = function(callback) - this event is invoked when the new scene is finished with loading
onPlayerChanged = function(callback) - this event is invoked when something has changed with the player (name, flag, moving, standing, stillness, damage, inventory)
onPlayerMoveStart = function(callback) - this event is invoked when the player starts moving
onPlayerMoveStop = function(callback) - this event is invoked when the player stops moving
onPlayerIsStill = function(callback) - this event is invoked when the player is still (has stillness bonus)
onPlayerDamage = function(callback) - this event is invoked when the players focus or health is about to change
onPlayerInventory = function(callback) - this event is invoked when the inventory of the player changed
onClientWindow = function(callback) - this event is invoked when a client window opens or closes (paperdoll)
onClientIsHitching = function(callback) - this event is invoked when the client starts hitching
onClientIsLoading = function(callback) - this event is invoked when the client is loading a scene
onMouseMove = function(callback) - this event is invoked when the mouse is moved
onMouseButton = function(callback) - this events is invoked when a mouse button is pressed

label = {
add = function(left, top, width, height, caption) - adds a GUI label. returns the index
get = function(index) - returns the GUI label instance

remove = function(index) - removes a GUI label
rect = function(index, rect) - applies a rect structure for the label (from libsota.util)
caption = function(index, caption) - sets or gets the caption of a GUI label. caption = nil to get only
visible = function(index, visible) - sets or gets the visible state of a GUI label
toggle = function(index) - toggles the visible state of a GUI label
moveTo = function(index, x, y) - moves a GUI label to the given position
moveBy = function(index, x, y) - moves a GUI label by the given values (delta)
resizeTo = function(index, w, h) - resizes a GUI label to the given size
resizeBy = function(index, x, y) - resizes a GUI label by the given values (delta)
shownInScene = bool: true when the label should be shown in scene
shownInLoadScreen = bool: true when the label should be shown on load screen
zIndex = int: the z-index of the GUI label
},

texture = {
add = function(left, top, filename, clamped, scaleMode, width, height) - add a GUI texture instance. returns index. loads the texture if necessary
clone = function(index) - clones the texture instance and returns the index of the new instance. the new instance has visible set to false
get = function(index) - returns the GUI texture instance
remove = function(index) - removes a GUI texture instance. (purging textures are currently not supported)
clamp = function(index, clamped) - sets or gets if texture is clamped (bool)
scaleMode = function(index, scaleMode) - set or gets the scaleMode (StretchToFill, ScaleAndCrop, ScaleToFit)
rect = function(index, rect) - applies a rect structure to the texture (from libsota.util)
visible = function(index, visible) - sets or gets the visible state of the texture
toggle = function(index) - toggles the visible state of the texture
moveTo = function(index, x, y) - moves a texture the given position
moveBy = function(index, x, y) - moves a texture by the given values (delta)
resizeTo = function(index, w, h) - resizes a texture to the given size
resizeBy = function(index, x, y) - resizes a texture by the given values (delta)
shownInScene = bool: true when the texture should be shown in scene
shownInLoadScreen = bool: true when the texture should be shown on load screen
zIndex = int: the z-index of the GUI texture

```
		textureID = int: shroud id of the texture
		textureWidth = int: width of the texture
		textureHeight = int: height of the texture
	},


	shortcut = {
		add = function(action, ...) - adds a shortcut that calls the given function. returns the index.
		remove = function(index) - removes a shortcut
		invoke = function(index) - calls the function belonging to that index
	},

	command = {
		add = function(command, callback) - adds a command callback. returns the index
		remove = function(command) - removes a command callback
		invoke = function(command, ...) - invokes the callback belonging to the command
		restrict = function(command, channel, sender, receiver) - restrict command invokation on channel, sender or receiver
	},

	verbosity = int: verbosity level
	consoleLog = function(message, verbosity) - send a multiline message to the console depending on verbosity level
}

Remarks:
callbacks:
onInit(function() end)
onStart(function() end)
onUpdate(function() end)
onConsoleInput(function(channel, sender, receiver, messages) end)
onConsoleCommand(function(source, command, tail) end) - source is a table containing channel, sender, receiver
onPlayerChanged(function(what, health, focus) end) - what contains what has changed. health and focus are only set when player takes damage
onPlayerMoveStart(function() end)
```

onPlayerMoveStop(function() end)
onPlayerIsStill(function() end)
onPlayerDamage(function(health, focus) end) - health and focus are the new values, the old ones are in player.health and player.focus. Both are tables
onPlayerInventory(function(changed) end) - changed contains a table what has changed
onClientWindow(function(which, window) end) - which contains the name, window { open = bool, left, top, width, height }
onClientIsHitching(function() end)
onClientIsLoading(function() end)
onMouseMove(function(button, x, y) end)
onMouseButton(function(state, button, x, y) end)

[/CODE]

libsota.util:
[CODE]

-- timer utility functions

function setTimeout(timeout, callback) - calls the callback after the given timeout. returns the index
function setInterval(interval, callback) - calls the callback each interval. returns the index
function getTimer(index) - returns the timer instance
function cancelTimer(index) - cancel the given timer
function pauseTimer(index) - pause the given timer
function resumeTimer(index) - resume the given timer

-- label utility functions

function createLabel(left, top, width, height, caption) - creates a label. returns the index
function createLabelWithShadow(left, top, width, height, caption) - creates a label with shadow. returns the index. the utility functions have to be used
function getLabel(index) - returns the GUI label instance

function getLabelCaption(index) - retruns the caption of the GUI label
function setLabelCaption(index, caption) - sets the caption of the GUI label
function removeLabel(index) - removes GUI label
function showLabel(index) - shows the GUI label
function hideLabel(index) - hide the GUI label
function toggleLabel(index) - toggle the visible state of the GUI label
function isLabelVisible(index) - returns if the GUI label is visible
function setLabelVisible(index, visible) - sets the visible state of the GUI label
function moveLabelBy(index, x, y) - moves the GUI label by x,y (delta)
function moveLabelTo(index, x, y) - moves the GUI label to x,y (absolute)
function resizeLabelBy(index, w, h) - resizes the GUI label by w,h (delta)
function resizeLabelTo(index, w, h) - resizes the GUI label to w,g (absolute)
function moveLabelOffsetCenter(index, x, y) - moves the GUI label by x, y relative to the center of the screen and auto size it

--- other

ui.onShortcutPressed("key" [,key][,key]..., callback) - adds a shortcut and invokes the callback when the shortcut is pressed
ui.onShortcut("key" [,key][,key]..., callback) - adds a shortcut and invoke the callback when ever something is changed: down, up, held, pressed
ui.onCommand(command, callback) - adds a command and invoke the callback when the comand is entered. The first argument is the source after this all args entered following

--- structurs / obejects

rect = {
	new = function(left, top, width, height) - returns a rect object. can also be used r = rect(..)
	fromString = function(string) - returns a rect taken vom string. width and height are filled in. rect.fromString(string)
	moveTo = function(rect, x, y) - moves a rect to x,y (absolute). rect.moveTo(rect, x, y) or r:moveTo(x, y)
	moveBy = function(rect, x, y) - moves a rect by x,y (delta). rect.moveBy(rect, x, y) or r:moveBy(x, y)
	resizeTo = function(rect, w, h) - resizes a rect to w,h (absolute). rect.resizeTo(rect, w, h) or r:resizeTo(w, h)
	resizeBy = function(rect, w, h) - resizes a rect by w,h (delta). rect.resizeBy(rect, w, h) or r:resizeBy(w, h)
}

string.style = function(string, style) - applies a style to a string. a style is a table that may contain { size = nn, color = color, bold = true/false, italic = true/false }
Eg: string:style({ color = blue, italic = true })
string.rect(string) is a alias for rect.fromString - string:rect() returns a rect taken from the string

[/CODE]

Quick and dirty window from libsota.util
[CODE]
w = window(left, top, width, height, title) - create a window object
window.moveAble - allows window to be moved by the "window manger" or not
window.resizeAble - allows window to be resized by the "window manager" or not
window:moveBy, window.moveTo - moves window (by = delta, to = absolute)
window:resizeBy, window.resizeTo - resizes window (by = delta, to = absolute)
window:visible(bool) - set window visible or not
window:zIndex(zIndex) - set the z-index of the window
window:createLabel(name, x, y, caption) - creates a label as child for that window.
window:createTextbox(name, x, y, w, h) - creates a label as child for that window
window:setCaption(name, caption) - set te caption of label that is child of that window
window:setText(name, caption) - set the text of label that is a child of that window
window:getLabel(name) - returns the index of the ui.label used

wdm.add(window) - adds a window object to the window manager

the window manager allows:
switching through the windows using LeftControl + Tab.
Focused window can be moved using LeftControl and the ArrowKeys.
LeftControl + F4 hides or unhides the focused window.
Focused window can be resized using LeftAlt and the ArrowKeys
Windows are brought to front when they get the focus.
[/CODE]

Commands added by libsota.util
\lua lua - shows lua version and path
\lua api - shows list of available functions and globals from Shroud api
\info xp -- shows players pooled xp
\info stat <number or name> -- shows stat info and value
\info client -- shows client info and variable names (client object from libsota)
\info player -- shows player info and variable names (player object from libsota)
\info scene -- shows scene info and variable names (scene object from libsota)
\info ui -- shows the functions you can use (api) from libsota (ui object from libsota)
\info lib -- shows other info from libsota. incl. list of registered / added commands