

SRL Documentation

SRL Developers

August 16, 2010

Users.scar

LoadCheckListForm

procedure [LoadCheckListForm](#);

Description: Initializes the check list form needed for Users.scar

ShowCheckListFormModal

procedure [ShowCheckListFormModal](#);

Description: Shows the form.

ShowCheckListForm

procedure [ShowCheckListForm](#);

Description: Safely initializes and shows the form

SavePlayers

procedure [SavePlayers](#);

Description: Saves the players to a file / creates a file when needed.

arPinEditsPress

procedure [arPinEditsPress](#);

Description: Detects if user press non numerical keys to remove them

SwitchInTabArray

procedure [SwitchInTabArray](#);

Description: Switches values in a TTabSheet array

SwitchInEditArray

procedure [SwitchInEditArray](#);

Description: Switches values in a TEdit array

SwitchInComboArray

procedure [SwitchInComboArray](#);

Description: Switches values in a TComboBox array

OnSRLPFMouseDown

procedure [OnSRLPFMouseDown](#);

Description: Form action on Mouse click

OnSRLPFDragDrop

procedure [OnSRLPFDragDrop](#);

Description: Form action on drag drop

OnSRLPFDragOver

procedure [OnSRLPFDragOver](#);

Description: Form action on drag over

DeleteInEditArray

procedure [DeleteInEditArray](#);

Description: Deletes a value in a TEdit array

DeleteInEditArrayOfArray

procedure [DeleteInEditArrayOfArray](#);

Description: Deletes a value in a TEdit array

DeleteInEditArrayOfArrayOfArray

procedure [DeleteInEditArrayOfArrayOfArray](#);

Description: Deletes a value in a TEdit array

DeleteInComboArray

procedure [DeleteInComboArray](#);

Description: Deletes a value in a TComboBox array

DeleteInComboArrayOfArray

procedure [DeleteInComboArrayOfArray](#);

Description: Deletes a value in a TComboBox array

DeleteInComboArrayOfArrayOfArray

procedure [DeleteInComboArrayOfArrayOfArray](#);

Description: Deletes a value in a TComboBox array

DeleteInTabArray

procedure [DeleteInTabArray](#);

Description: Deletes a value in a TTabSheet array

DeleteInTabArrayOfArray

procedure [DeleteInTabArrayOfArray](#);

Description: Deletes a value in a TTabSheet array

DeleteInPageControlArray

procedure [DeleteInPageControlArray](#);

Description: Deletes a value in a TTabSheet array

SetLabelsParent

procedure [SetLabelsParent](#);

Description: Sets all labels to the desired parent

DeletePlayerInTab

procedure [DeletePlayerInTab](#);

Description: Deletes a player in a desired tab and goes to the next desired tab.

ClearPlayerInTab

procedure [ClearPlayerInTab](#);

Description: Clears player settings in tab

AddPlayerInTab

procedure [AddPlayerInTab](#);

Description: Deletes a player in a desired tab and goes to the next desired tab.

AddArmy

procedure [AddArmy](#);

Description: Adds an army to the form

DeleteArmy

procedure [DeleteArmy](#);

Description: Deletes an army in the form

RenameArmy

procedure [RenameArmy](#);

Description: Renames an army's name

ExportPlayerData

procedure [ExportPlayerData](#);

Description: Turns the info input into the forms into a copy and pasteable

ImportPlayerData

procedure [ImportPlayerData](#);

Description: Inputs information from the DeclarePlayers procedure into the form

OnSRLPFMainMenuClick

procedure [OnSRLPFMainMenuClick](#);

Description: Form actions on the MainMenu items

OnSRLPFClick

procedure [OnSRLPFClick](#);

Description: Form action on a click on a button

OnSRLPFShow

procedure [OnSRLPFShow](#);

Description: Form action on a Tab show

SRLPFMenuClick

procedure [SRLPFMenuClick](#);

Description: Form action on Menu click

SetSRLPlayerFormArrays

procedure [SetSRLPlayerFormArrays](#);

Description: Sets the arrays for the form

FillSRLPlayerFormTab

procedure [FillSRLPlayerFormTab](#);

Description: Fills the tabs with components

LoadSRLPlayerForm

procedure [LoadSRLPlayerForm](#);

Description: Loads the main SRL player form

ShowSRLPFFormModal

procedure [ShowSRLPFFormModal](#);

Description: Shows the form

SRLPlayerForm

procedure [SRLPlayerForm](#);

Description: Main function sets and loads values needed for player settings and the form.

Doorprofiles.scar

IsDropDot

function IsDropDot(Pt: TPoint): Boolean ;
Description: Returns true if a point is in a drop dot.

FilterDropDots

function FilterDropDots(P: TPointArray): TPointArray ;
Description: Returns a TPointArray filtered of drop dots.

GetDoorPoints

function GetDoorPoints;
Description: Returns all door points on the minimap in a T2DPointArray.

GetDoorColors

function GetDoorColors;
Description: Returns all door colors on the minimap in an array.

GetDoorColor

function GetDoorColor;
Description: Returns the first door color found from the center of the minimap.

RearrangeTPA

function RearrangeTPA(TPA: TPointArray; sortX up: Boolean): TPointArray ;
Description: Sorts a TPA according to params.

MiddleDoor

function MiddleDoor(var P: TPointArray; var s: extended): TPoint ;
Description: Returns the slope and midpoint of a door.

GetDoors

function `GetDoors`;

Description: Returns information about the doors on the minimap in an array of

GetDoorsIn

function `GetDoorsIn`(`x1` `y1` `x2` `y2`: `integer`): array ;

Description: Returns information about the doors on a portion of the minimap in

SortDoorsFrom

procedure `SortDoorsFrom`;

Description: Sorts an array of DoorProfile from a TPoint.

QuickChat.scar

QCOpen

function [QCOpen](#);

Description: Returns true/false depending on if QuickChat window is open.

OpenQC

procedure [OpenQC](#);

Description: Opens QuickChat window if it isn't open.

CloseQC

procedure [CloseQC](#);

Description: Closes QuickChat window if it is open.

QCOpenTab

procedure [QCOpenTab](#);

Description: Opens QuickChat clicks 'Menu' menu then uses 'SubMenus' chars to

QCSayLevel

procedure [QCSayLevel](#);

Description: Says 'Which' skill level using Quick Chat.

Smart.scar

NewAutoColor.scar

NAC_GetAutoColorConstants

procedure [NAC_GetAutoColorConstants](#);
Description:

NAC_LoadtColors

procedure [NAC_LoadtColors](#);
Description:

SetupNewAutoColor

procedure [SetupNewAutoColor](#);
Description: Call this once to set up the auto color.

NAC_SetGlobalColors

procedure [NAC_SetGlobalColors](#);
Description:

SetAutoColorReference

procedure [SetAutoColorReference](#);
Description:

GrandExchange.scar

InGE

function [InGE](#);

Description: Checks if we are in GE

GEIsMember

function [GEIsMember](#);

Description: Checks if it is a member account using GE

GESellMenu

function [GESellMenu](#);

Description: Checks if we are in the sell menu

GEBuyMenu

function [GEBuyMenu](#);

Description: Checks if we are in the buy menu

GEMenuExit

procedure [GEMenuExit](#);

Description: Exits the Buy or Sell menu

GESpotToTBox

function [GESpotToTBox](#)([spot](#): [integer](#)): TBox ;

Description: Returns the coords of a box in GE

GEIsSpotFree

function [GEIsSpotFree](#)([spot](#): [integer](#)): boolean ;

Description: Checks if a spot is available for buying/selling

GEFinishedAt

function `GEFinishedAt`(`spot`: `integer`): `boolean` ;

Description: Checks if a trade has been concluded at `spot`: `spot`

GERecoverAtEx

procedure `GERecoverAtEx`;

Description: Collects the items traded received at a certain `spot`

GERecoverAt

procedure `GERecoverAt`;

Description: Recovers items at `spot`: `spot` does not abort the trade

GERecoverFinishedAt

procedure `GERecoverFinishedAt`;

Description: Recovers items at `spot`: `spot` if the trade is concluded

GEAbortAll

procedure `GEAbortAll`;

Description: Aborts and recovers all current trades

GERecoverAll

procedure `GERecoverAll`;

Description: Recovers all current trades does not abort

GERecoverFinished

procedure `GERecoverFinished`;

Description: Recovers all finished trades

GEFindFreeSpot

function `GEFindFreeSpot`(`var spot`: `integer`): `boolean` ;

Description: Finds a free spot to trade in

GESellAt

function `GESellAt`(`spot`: `integer`): `boolean` ;

Description: Presses the sell button at `spot`: `spot`

GEBuyAt

function GEBuyAt(**spot**: integer): boolean ;
 Description: Presses the buy button at spot: spot

GESetPrice

function GESetPrice(**value**: string): boolean ;
 Description: Sets the price of an item.

GESetQuantity

procedure GESetQuantity;
 Description: Sets the price of an item.

GEConfirmOffer

function GEConfirmOffer;
 Description: Confirms an offer

GEBuy

function GEBuy(**What Price Quantity**: string): boolean ;
 Description: Buys an item from the GE

GESell

function GESell(**Price Quantity**: string; **InvSpot**: integer): boolean ;
 Description: Sells an item at GE

DraynorColorFinder.scar

S

procedure `s`;
No Description

FindDraynorRoadColor

function `FindDraynorRoadColor`(`var RoadColor` : `Integer`; `RX1` `RY1` `RX2` `RY2`: `Integer`): `Boolean` ;
Description: Finds Draynor Road Color

FindDraynorLadderColor

function `FindDraynorLadderColor`(`var LadderColor` : `Integer`): `Boolean` ;
Description: Finds Draynor Road Color

FindDraynorHouseColor

function `FindDraynorHouseColor`(`var HouseColor` : `Integer`): `Boolean` ;
Description: Finds Draynor House Color

IsRoad

function `IsRoad`(`RX1` `RY1` `RX2` `RY2`: `Integer`): `Boolean` ;
Description: Finds Road Bitmap inside given coordinates.

FindDraynorWaterColor

function `FindDraynorWaterColor`(`var WaterColor` : `Integer`): `Boolean` ;
Description: Find Draynor Water Color.

WaterFound

function `WaterFound`(`WX1` `WY1` `WX2` `WY2`: `Integer`): `Boolean` ;
Description: Finds Water Color at given coordinates.

FindDraynorRockColor

function FindDraynorRockColor(var RockColor : Integer): Boolean ;
 Description: Find Draynor Rock Color.

RockFound

function RockFound(var RockColor : Integer; WX1 WY1 WX2 WY2: Integer): Boolean ;
 Description: Finds Rock Color inside given coordinates.

FindDraynorDoorColor

function FindDraynorDoorColor(var DoorColor : Integer): Boolean ;
 Description: Find Draynor Door Color. (Red: 230-245)

FindDraynorColors

function FindDraynorColors(var RoadColor WaterColor BankColor RockColor DoorColor LadderColor HouseColor : Integer; DraynorMiscColor : Boolean): Boolean ;
 Description: Find All Draynor Colors.

FontUpdater.scar

srl_UpdateFonts

procedure [srl_UpdateFonts](#);

Description: Updates all outdated fonts (as said in the Fonts array) in SCAR.

Reports.scar

SRLProgressReport

function SRLProgressReport(ResultType: integer; ScriptName ScriptAuthor
ScriptVersion: String; VarNames: TStringArray; VarValues: TVariantArray
) : Variant ;

Description: Prints a Standardized Progress Report with the variables of your choice

SRLPlayerReport

function SRLPlayerReport(ResultType SkillLevel: Integer; LongReport: boolean;
DefaultSettings: TBooleanArray; VarNames: TStringArray; PlayerBooleans
PlayerIntegers PlayerStrings PlayerExtendeds: TIntegerArray) : Variant ;

Description: Prints a Standardized MultiPlayer Report with the variables of your choice

Debug.scar

TakeScreenshotEx

procedure [TakeScreenshotEx](#);

Description: Takes a screenshot of everything within X1 Y1 X2 Y2 bounds and

TakeScreenshot

procedure [TakeScreenshot](#);

Description: Takes a screenshot saving it to 'Path'. See TakeScreenShotEx.

DebugBitmap

procedure [DebugBitmap](#);

Description: Shows a bitmap in the debug image window.

DebugTPA

function [DebugTPA](#)(Points: TPointArray; BmpName: string): Boolean ;

Description: Shows an image of the Debug screen plotting colors over the points

DebugATPA

function [DebugATPA](#)(aPoints: array of TPointArray; BmpName: string):

Boolean ;

Description: Shows an image of the client plotting each TPA of the 2D point

DebugATPABounds

procedure [DebugATPABounds](#);

Description: Shows an image of the client plotting each TPA's bounds as a box

DebugColorArray

procedure [DebugColorArray](#);

Description: Writes the Color Arrays information from GetColorArrayInfo into the

arrayLoader.scar

strToTPtArrayEx

function strToTPtArrayEx(**tPtStr** Separator: **string**): TPointArray ;
Description: Loads a String into a TPointArray. (Example String: '50 10 80 53 350 765 8560 1897 2348 346')

strToTPtArray

function strToTPtArray(**tPtStr**: **string**): TPointArray ;
Description: Refer to strToTPtArrayEx.

tPtArrayToStrEx

function tPtArrayToStrEx(**newTPoint**: TPointArray; Separator: **string**): **string** ;
Description: Loads a TPointArray into a String. (Example String: '50 10 80 53 350 765 8560 1897 2348 346')

tPtArrayToStr

function tPtArrayToStr(**newTPoint**: TPointArray): **string** ;
Description: Refer to tPtArrayToStrEx.

strToIntArrayEx

function strToIntArrayEx(**intArray** Separator: **string**): TIntegerArray ;
Description: Loads a String into an array of integer. (Example String: '567 389 3234 98342 7998767')

strToIntArray

function strToIntArray(**intArray**: **string**): TIntegerArray ;
Description: Refer to strToIntArrayEx.

intArrayToStrEx

function `intArrayToStrEx`(`intArray`: `TIntegerArray`; `Separator`: `string`): `string` ;
 Description: Loads an array of integer into a String. (Example String: '567 389 3234 98342 7998767')

intArrayToStr

function `intArrayToStr`(`intArray`: `TIntegerArray`): `string` ;
 Description: Refer to `intArrayToStrEx`.

strToFloatArrayEx

function `strToFloatArrayEx`(`floatArray` `Separator`: `string`): `TExtendedArray` ;
 Description: Loads a String into an array of float. (Example String: '567.356 389.6756 3234.9865 98342.2314 7998767.324')

strToFloatArray

function `strToFloatArray`(`floatArray`: `string`): `TExtendedArray` ;
 Description: Refer to `strToFloatArrayEx`.

floatArrayToStrEx

function `floatArrayToStrEx`(`floatArray` : `TExtendedArray`; `Separator`: `string`): `string` ;
 Description: Loads an array of float into a String. (Example String: '567.356 389.6756 3234.9865 98342.2314 7998767.324')

floatArrayToStr

function `floatArrayToStr`(`floatArray`: `TExtendedArray`): `string` ;
 Description: Refer to `floatArrayToStrEx`.

strToStrArrayEx

function `strToStrArrayEx`(`strArray` `Separator`: `string`): `TStringArray` ;
 Description: Loads a String into an array of String. (Example String: 'Hello how are you today my friend? :D')

strToStrArray

function `strToStrArray`(`strArray`: `string`): `TStringArray` ;
 Description: Refer to `strToStrArrayEx`.

strArrayToStrEx

function strArrayToStrEx(strArray: TStringArray; Separater: string): string ;
 Description: Loads an array of String into a String. (Example String: 'Hello how are you today my friend? :D')

strArrayToStr

function strArrayToStr(strArray : Array of String): String ;
 Description: Refer to strArrayToStrEx.

strToBoolArrayEx

function strToBoolArrayEx(boolArray Separater: string): TBooleanArray ;
 Description: Loads a String into an array of boolean. (Example String: 'true false false true true false true false')

strToBoolArray

function strToBoolArray(boolArray: string): TBooleanArray ;
 Description: Refer to strToBoolArrayEx.

boolArrayToStrEx

function boolArrayToStrEx(boolArray : TBooleanArray; Separater: string): string ;
 Description: Loads an array of boolean into a String. (Example String: 'true false false true true false true false')

boolArrayToStr

function boolArrayToStr(boolArray : TBooleanArray): string ;
 Description: Refer to boolArrayToStrEx.

WizzyPlugin.scar

tSwap

procedure `tSwap`;

Description: Swaps the two TPoints.

tpaSwap

procedure `tpaSwap`;

Description: Swaps the two TPointArrays.

SwapE

procedure `SwapE`;

Description: Swaps the two Extended values.

RAaSTPAEx

procedure `RAaSTPAEx`;

Description: Leaves one point per box with side lengths W and H to the TPA.

RAaSTPA

procedure `RAaSTPA`;

Description: Leaves one point per box with the side length Dist.

NearbyPointInArrayEx

function `NearbyPointInArrayEx`(`const` `P`: `TPoint`; `w` `h`: `Integer`; `a`: `TPointArray`): `Boolean` ;

Description: Returns true if the point P is near a point in the TPA a with the

NearbyPointInArray

function `NearbyPointInArray`(`const` `P`: `TPoint`; `Dist`: `Integer`; `a`: `TPointArray`): `Boolean` ;

Description: Returns true if the point P is near a point in the TPA a with the

ReArrangeandShortenArrayEx

function ReArrangeandShortenArrayEx(a: TPointArray; w h: Integer): TPointArray ;

Description: Results the TPointArray a with one point per box with side lengths

ReArrangeandShortenArray

function ReArrangeandShortenArray(a: TPointArray; Dist: Integer): TPointArray ;

Description: Results the TPointArray a with one point per box with side length

TPAtoATPAEx

function TPAtoATPAEx(TPA: TPointArray; w h: Integer): T2DPointArray ;

Description: Splits the TPA to boxes with sidelengths W and H and results

TPAtoATPA

function TPAtoATPA(TPA: TPointArray; Dist: Integer): T2DPointArray ;

Description: Splits the TPA to boxes with sidelength Dist and results

SortTPAFrom

procedure SortTPAFrom;

Description: Sorts the TPointArray a from the point From.

SortATPAFrom

procedure SortATPAFrom;

Description: Sorts the T2DPointArray a from the point From.

SortATPAFromFirstPoint

procedure SortATPAFromFirstPoint;

Description: Sorts the T2DPointArray a from the point From.

InvertTPA

procedure InvertTPA;

Description: Inverts / Reverts the TPointArray a.

InvertATPA

procedure `InvertATPA`;

Description: Inverts / Reverts the T2DPointArray a.

MiddleTPAEx

function `MiddleTPAEx`(`TPA`: `TPointArray`; `var x y`: `Integer`): `Boolean` ;

Description: Stores the coordinates of the middle of the TPointArray a to X & Y.

MiddleTPA

function `MiddleTPA`(`tpa`: `TPointArray`): `TPoint` ;

Description: Returns the middle of the TPointArray tpa.

SortATPASize

procedure `SortATPASize`;

Description: Sorts the T2DPointArray a from either largest or smallest by the

CombineTPA

function `CombineTPA`(`Ar1 Ar2`: `TPointArray`): `TPointArray` ;

Description: Combines the TPointArrays Ar1 and Ar2 and results the combination.

CombineIntArray

function `CombineIntArray`(`Ar1 Ar2`: `TIntegerArray`): `TIntegerArray` ;

Description: Combines the TIntegerArrays Ar1 and Ar2 and results the

InIntArrayEx

function `InIntArrayEx`(`a`: `TIntegerArray`; `var Where`: `Integer`; `const Number`: `Integer`): `Boolean` ;

Description: Returns true if the integer Number was found in the integer array

InIntArray

function `InIntArray`(`a`: `TIntegerArray`; `Number`: `Integer`): `Boolean` ;

Description: Returns true if the integer Number was found in the integer array a.

ClearSameIntegers

procedure [ClearSameIntegers](#);

Description: Clears the duplicates in the integer array a.

ClearSameIntegersAndTPA

procedure [ClearSameIntegersAndTPA](#);

Description: Clears the duplicates in the integer array a and the TPointArray p.

SplitTPAEx

function [SplitTPAEx](#)([arr](#): TPointArray; [w](#) [h](#): Integer): T2DPointArray ;

Description: Splits the points with max X and Y distances W and H to their

SplitTPA

function [SplitTPA](#)([arr](#): TPointArray; [Dist](#): Integer): T2DPointArray ;

Description: Splits the points with max distance Dist to their own TPointArrays.

FilterPointsPie

procedure [FilterPointsPie](#);

Description: Removes the points in the TPointArray Points that are not within

RemoveDistTPointArray

function [RemoveDistTPointArray](#)([x](#) [y](#) [dist](#): Integer; [ThePoints](#): TPointArray; [RemoveHigher](#): Boolean): TPointArray ;

Description: Removes the points that are inside or outside the distance Dist

GetTPABounds

function [GetTPABounds](#)([TPA](#): TPointArray): TBox ;

Description: Returns the boundaries of the TPA as a TBox.

FindTPAinTPA

function [FindTPAinTPA](#)([SearchTPA](#) [TotalTPA](#): TPointArray; [var](#) [Matches](#): TPointArray): Boolean ;

Description: Looks for the TPA SearchTPA in the TPA TotalTPA and returns

FindTextTPAinTPA

function FindTextTPAinTPA(Height: Integer; SearchTPA TotalTPA: TPointArray; var Matches: TPointArray): Boolean ;

Description: Read the description of FindTPAinTPA. Additional Height parameter.

FindGapsTPA

function FindGapsTPA(TPA: TPointArray; MinPixels: Integer): T2DPointArray ;

Description: Finds the possible gaps in the TPointArray TPA and results the

CreateTPAFromBMP

function CreateTPAFromBMP(BmpDC: HDC): TPointArray ;

Description: Creates a TPointArray of the bitmap dc BmpDC.

SortCircleWise

procedure SortCircleWise;

Description: Sorts all points in tpa by distance from degree (Deg) and distance from

LinearSort

procedure LinearSort;

Description: Sorts all points in tpa by distance from degree (Deg) and distance from

MergeATPA

function MergeATPA(ATPA: T2DPointArray): TPointArray ;

Description: Merges the TPointArrays of the T2DPointArray ATPA in to one TPA.

Path.scar

SetPathLength

procedure `SetPathLength`;
Description:

SetupPath

procedure `SetupPath`;
Description:

SetPath

function `SetPath`(`Number`: `Integer`): `Integer` ;
Description: Assembles DTM from Path. Inputs Path-Index outputs DTM.

WalkPath

function `WalkPath`(`ThePath`: `Integer`): `Boolean` ;
Description: Simple Roadclicker checks for RoadColor changes using FlagEx.

FindPath

function `FindPath`(`ThePath`: `Integer`; `Click`: `Boolean`): `Boolean` ;
Description: Returns True if Path was found.

PathRecorder

procedure `PathRecorder`;
Description: Simple PathRecorder.

FindXP.scar

ConvertXpToLvl

function `ConvertXpToLvl`(`Xp`: `Integer`): `Integer` ;
Description: Returns skill level based on xp level

ConvertLvlToXP

function `ConvertLvlToXP`(`Xp`: `Integer`): `Integer` ;
Description: Returns exp based on level

FindXP

function `FindXP`(`Skill sType`: `string`): `Extended` ;
Description: Returns any XP for any item in any skill.

trade.scar

SomeoneTrades

function [SomeoneTrades](#);

Description: Returns true if someone trades.

GetTradersName

function [GetTradersName](#);

Description: Returns the name of the trade as displayed in the chatbox.

TradeScreenName

function [TradeScreenName](#);

Description: Returns the name of trader from the trade screen.

AcceptTrade

function [AcceptTrade](#);

Description: Accepts the trade by clicking the chatbox.

Accept

procedure [Accept](#);

Description: Accepts the opened trade.

Decline

procedure [Decline](#);

Description: Declines the opened trade.

Waiting

function [Waiting](#);

Description: Returns true if you are waiting for the other player to accept the trade.

PlayerAccepted

function `PlayerAccepted`;

Description: Returns true if the other player accepted the trade.

CheckItem

function `CheckItem`(`them`: `Boolean`; `name`: `string`): `Integer` ;

Description: returns number item at second screen

InTradersArray

function `InTradersArray`(`TraderName`: `string`; `var Position`: `Integer`): `Boolean` ;

Description: Checks for the occurrence of Name in the Trader array and returns

UpdateTradersArray

procedure `UpdateTradersArray`;

Description: Updates the Trader array by setting TimeMark if player exists or

HandleTrade

function `HandleTrade`;

Description: Handles a trade.

Friends.scar

FriendName

function FriendName(PlayerNumber: Integer): string ;
Description: Gets FriendsName and saves it to a string.

FriendWorld

function FriendWorld(PlayerNumber: Integer; SameWorld: Boolean): String ;
Description: Returns current world of a friend.

GetFriends

procedure GetFriends;
Description: Returns first 13 friends World/Status/Name into the Friends Record.

FriendIsOnline

function FriendIsOnline(var iX iY: Integer; Friend: string): Boolean ;
Description: Returns coordinate of Online friend if Online.

MessageFriendEx

function MessageFriendEx(Friend TheMessage: string): Boolean ;
Description: If a certain friend is online sends them a message.

NumberOfOnlineFriends

function NumberOfOnlineFriends;
Description: Returns # of friends online from 0-11.

AddFriend

procedure AddFriend;
Description: Add's 'Friend' to your friends list.

DeleteFriend

procedure `DeleteFriend`;

Description: Delete's 'Friend' from your friends list.

AddIgnore

procedure `AddIgnore`;

Description: Add's 'Friend' to your ignore list.

DeleteIgnore

procedure `DeleteIgnore`;

Description: Deletes 'Friend' from Ignore list.

GetFriendTexts

function `GetFriendTexts`;

Description: Returns all the friend chats even the empty lines.

FromFriend

function `FromFriend`(`Friend`: `string`): `string` ;

Description: Returns the first occurrence of text sent by friend Friend.

FromFriendMulti

function `FromFriendMulti`(`Friend`: `string`): `TStringArray` ;

Description: Returns all the occurrences of text sent by friend Friend.

GetChatterOfLine

function `GetChatterOfLine`(`Line`: `string`): `string` ;

Description: Returns the friend who sent the text Line.

RespondToWith

function `RespondToWith`(`rTo` `rWith`: `string`): `Boolean` ;

Description: Returns true if could respond to message rTo with rWith.

HasResponded

function `HasResponded`(`Friend`: `string`): `Boolean` ;

Description: Returns true if friend Friend has responded to you.

SmartColors.scar

SC_Debug

procedure [SC_Debug](#);

Description: Will Debug SmartColors if DEBUG_SMART_COLORS = True

MakeObjEx

procedure [MakeObjEx](#);

Description: Will save the information about an object to a global array.

MakeObj

procedure [MakeObj](#);

Description: Will save the information about an object to a global array.

GetObject

function [GetObject](#)(Which: [string](#); var ObjID: [integer](#)): TMSObj ;

Description: Retrieves object 'Which' from the global array

SetupSmartColorsDetailsEx

procedure [SetupSmartColorsDetailsEx](#);

Description: Sets up several variables used in SmartColors.

SetupSmartColorsDetails

procedure [SetupSmartColorsDetails](#);

Description: Sets up several variables used in SmartColors.

CheckNewColor

function [CheckNewColor](#)(Color: [integer](#); Rock: [TMSObj](#)): boolean ;

Description: Returns true if Color is in tolerance range of the color

AnalyseColor

function AnalyseColor(Color: integer; Rock: TMSObj): integer ;

Description: Scans MS for Color returning the number of times it was found

OrganizeColors

function OrganizeColors(Colors Lengths: TIntegerArray): TIntegerArray ;

Description: Organizes Colors rarest first returns organized TIntegerArray

GatherColors

function GatherColors(var Rock: TMSObj): boolean ;

Description: Gathers Colors from a box with double side length of

CheckAndClick

function CheckAndClick(Pos: TPoint Rock: TMSObj): boolean ;

Description: Checks Pos for matches with Rock.UpTexts. Return true if

FirstFinder

function FirstFinder(var Rock: TMSObj): boolean ;

Description: Finds object through Object record and sets up the SCArrary

SmartColorsEx

function SmartColorsEx(Index: integer): boolean ;

Description: Finds object by index. Need to call MakeObj first.

SmartColors

function SmartColors(FindWhat: string): boolean ;

Description: Finds object 'FindWhat'. Need to call MakeObj first.

BuySell.scar

ShopItemCoords

function ShopItemCoords(I: Integer): TPoint ;
Description: Results the coords for shop item. (Left to right).

ExistsShopItem

function ExistsShopItem(i: Integer): Boolean ;
Description: Checks if the shop item exists. (Left to right).

BuyCoords

function BuyCoords(x y Amount: Integer): Boolean ;
Description: Buys the specified amount of the Item at x y

BuyAmount

function BuyAmount(i Amount: Integer): Boolean ;
Description: Buys the specified amount of the Item at assigned row and column

BuyColor

function BuyColor(UpText: string; Color Amount: Integer): Boolean ;
Description: Buys all items with that color

SellCoords

function SellCoords(x y Amount: Integer): boolean ;
Description: Sells Item at x y

Sell

function Sell(i Amount: Integer): Boolean ;
Description: Sells Items at specified Inv Slot

FindItemsArray

function FindItemsArray(color tol: Integer; Text: string; num: Integer): TPointArray ;

Description: Returns Array of all Coords of that Item

FindItems

function FindItems(color tol: Integer; text: string; num: Integer): TPointArray ;

Description: Finds Items with color and up text.

SellItems

procedure SellItems;

Description: Sells items with that color UpText and Tolerance by

DungeonWalk.scar

DetectDungeonPoints

procedure DetectDungeonPoints;
Description:

GetBestDegree

function GetBestDegree(Degree: Integer): Integer ;
Description:

DungeonWalkEx

function DungeonWalkEx(Dir Radius MaxOff: Integer; Click: Boolean): Boolean
;
Description:

DungeonWalk

function DungeonWalk(Dir mRadius MaxOff: Integer): Boolean ;
Description:

DungeonDirExists

function DungeonDirExists(Dir Radius MaxOff: Integer): Boolean ;
Description:

Stats.scar

Players.scar

LoggedIn

function `LoggedIn`;

Description: Returns True if Logged In.

NumberOfPlayers

procedure `NumberOfPlayers`;

Description: Makes the Players array have Number indexes also sets strings and

PlayersActive

function `PlayersActive`;

Description: Numbers of players active in the Players array.

AllPlayersInactive

function `AllPlayersInactive`;

Description: Returns true if all players are inactive in the Players array.

PlayerWorked

function `PlayerWorked`(`player` : `integer`): `integer` ;

Description: Returns the time a player has worked..

Object.scar

FindAllObjsRecord

function FindAllObjsRecord(**var** Objects: TPointArray; AutoInfo: TAutoColorInfo; x1 y1 x2 y2: Integer; QuickSorting RGBXYZCheck: Boolean): Boolean ;

Description: Finds an Object using a TAutoColorInfo record returns the center of the found objects.

FindObjRecordEx

function FindObjRecordEx(**var** fx fy: Integer; AutoInfo: TAutoColorInfo; QuickSorting RGBXYZCheck: Boolean): Boolean ;

Description: Finds an Object using a TAutoColorInfo record with extra options.

FindObjRecord

function FindObjRecord(**var** fx fy: Integer; AutoInfo: TAutoColorInfo): Boolean ;

Description: Finds an Object using a TAutoColorInfo record.

FindObjEx

function FindObjEx(**var** cx cy: Integer; Text: TStringArray; Color: TIntegerArray; Tol Step xs ys xe ye: Integer): Boolean ;

Description: Searches for an object using an array of colors and texts within the area

FindObj

function FindObj(**var** cx cy: Integer; Text: string; Color Tol: Integer): Boolean ;

Description: Finds Object

FindObjCustom

function FindObjCustom(**var** cx cy: Integer; Text: TStringArray; Color: TIntegerArray; Tol: Integer): Boolean ;

Description: Finds Object with custom amount of colors and custom amount of Uptext

FindObjectDeformed

function FindObjectDeformed(**var** ObjX ObjY: Integer; UpText1 UpText2: String; BMP a b c d: Integer): Boolean ;

Description: Finds Object in Box a b c d specified by BitMap and Uptext1 or Uptext2

FindObjTPA

function FindObjTPA(**var** x y: integer; color tol cts ObjWidth ObjHeight minCount : Integer; UpText: tstringarray): Boolean ;

Description: Finds object using WizzyPlugin functions.

FindObjTPAMulti

function FindObjTPAMulti(**var** X Y: integer; Color: TIntegerArray; Tol CTS ObjWidth ObjHeight minCount : Integer; UpText: TStringArray): Boolean ;

Description: This is a modification of FindObjTPA which uses a TIntegerArray for the Colors.

FindObjThroughMM

function FindObjThroughMM(**var** x y: Integer; Kind: string; Colors Tol: TIntegerArray; UpText: TStringArray; Width Height: Integer): boolean ;

Description: Finds a object with the given parameters.

Flag.scar

FlagPresent

function FlagPresent;

Description: T/F depending on flag exist.

FlagDistance

function FlagDistance;

Description: Returns distance in pixels your char dot is from flag on mini-map

WalkingToFlag

function WalkingToFlag(WaitTime: Integer): Boolean ;

Description: Checks if your character is moving; stores current FlagDistance

FlagEx

function FlagEx(Dist: Integer; RoadColorCheck: Boolean): Boolean ;

Description: Waits until Flag is within "Distance" distance.

MouseFlagEx

function MouseFlagEx(X Y rX rY Xmod Ymod FlagDist: Integer; CheckForFlag CheckForRoadColor: Boolean): Boolean ;

Description: MouseFlag MFF & MFNF combined into one procedure.

MFF

function MFF(X Y Xmod Ymod: Integer): Boolean ;

Description: If the flag is not present it will click at (ax ay) increasing by

MFNF

function MFNF(ax ay xmod ymod: Integer): Boolean ;

Description: Like MFF except it will click even if the flag is present when the

FFlag

function FFlag(Dist: Integer): Boolean ;

Description: Waits until Flag is within "Distance" distance.

Flag

function Flag;

Description: Waits while flag exists.

MouseFlag

procedure MouseFlag;

Description: Mouse and Flag combined into one easy procedure.

AutoColor.scar

SetupAutoColors

procedure SetupAutoColors;

Description: Need to initialize the global vars Which store information

LoadColorArray

function LoadColorArray(Which: Integer): TIntegerArray ;

Description: Returns a color array depending on the parameter Which.

LoadAutoColor

function LoadAutoColor(which: Integer; var theAC: TAutoColorInfo): Boolean ;

Description: Finds the color on the minimap depending on the parameter which.

AutoColorEx

function AutoColorEx(i: Integer): TAutoColorInfo ;

Description: Check the isACLoaded boolean for the asked AC if true we know the

AutoColor

function AutoColor(which: Integer): Integer ;

Description: Returns the correct color based on the autocolor constant

GlobalStats.scar

SetLastCommitVars

procedure [SetLastCommitVars](#);

Description: Stores the last committed stats

SendSRLReport

procedure [SendSRLReport](#);

Description: Sends Current Stats to web server

SetUpSRLReport

procedure [SetUpSRLReport](#);

Description:

Bank.scar

BankScreen

function `BankScreen`;

Description: Finds Bankscreen. Returns true if Found.

DepositScreen

function `DepositScreen`;

Description: Returns true if deposit screen is Found.

PinScreen

function `PinScreen`;

Description: Finds Pin Screen. Returns true if Found.

InPin

function `InPin`(`Pin`: `String`): Boolean ;

Description: Enters bank pin. Will try 3 times returns true if bank is opened.

MSTPointToBankPoint

function `MSTPointToBankPoint`(`P`: `TPoint`): `TPoint` ;

Description: Converts a point on the Mainscreen to the point in the Bank.

BankPointToBankIndex

function `BankPointToBankIndex`(`P`: `TPoint`): Integer ;

Description: Converts a Bank Point (Row Col) to Bank Index (spot in bank).

BankIndexToBankPoint

function `BankIndexToBankPoint`(`ind`: Integer): `TPoint` ;

Description: Converts the Bank Index (spot in bank) to a Bank Point (Row Col).

BankIndexToMSPoint

function BankIndexToMSPoint(Index: Integer): TPoint ;

Description: Converts the Bank Index (spot in bank) to a Mainscreen TPoint of

BankIndexToMSBox

function BankIndexToMSBox(ind: Integer): TBox ;

Description: Converts the Bank Index (spot in bank) to a Mainscreen TBox of

DepositItemBox

function DepositItemBox(Item: Integer): TBox ;

Description: Returns a TBox surround Item (1 to 28) in the Deposit Box screen.

FixBank

procedure FixBank;

Description: Scrolls the bank screen up.

FixBankTab

function FixBankTab;

Description: Fixes the BankTab to 'View All'

CurrentBankTab

function CurrentBankTab;

Description: Returns Current Bank Tab selected

ExistsBankTab

function ExistsBankTab(T: Integer): Boolean ;

Description: Returns true if Bank Tab Exists

BankTab

function BankTab(T: Integer): Boolean ;

Description: If Bank Tab is there will switch to it.

SearchBank

procedure SearchBank;

Description: Searches for Item in your bank.

DepositAll

function `DepositAll`;

Description: Deposits all items in your inventory

Deposit

procedure `Deposit`;

Description: Deposits from Slot to ToSlot.

WithdrawEx

function `WithdrawEx`(Col Row Amount: Integer; Uptexts: TStringArray):

Boolean ;

Description: Withdraws Amount at Column/Row.

Withdraw

function `Withdraw`(Col Row Amount: Integer): Boolean ;

Description: Withdraws Amount at Column/Row.

WithdrawItemEx

function `WithdrawItemEx`(Ident: integer; IdentType: string; var Index: integer; Amount: integer; UpText: String; Tol: TIntegerArray): boolean ;

Description: Withdraws an item from the bank by using "Ident" with tol[0]

WithdrawItem

function `WithdrawItem`(Ident: integer; IdentType: string; Amount: integer; UpText: TStringArray; Tol: TIntegerArray): boolean ;

Description: Withdraws an item using WithdrawItemEx but removes the Index check.

Notes

procedure `Notes`;

Description: Toggles note withdrawal on or off.

CloseBank

function `CloseBank`;

Description: Closes the bank window - Tries twice before exiting

OpenBankQuiet

function OpenBankQuiet(WhichBank: string): Boolean ;
 Description: Opens any given bank in the free world using

OpenBankGlass

function OpenBankGlass(WhichBank: String; ChangeCompass ChangeAngle: Boolean): Boolean ;
 Description: Opens the bank.

OpenBankFast

function OpenBankFast(Location: String): Boolean ;
 Description: Opens the bank.

OpenBankNPC

function OpenBankNPC;
 Description: Opens the bank by using Banker.

OpenBank

function OpenBank(WhichBank: String ChangeCompass ChangeAngle: Boolean): Boolean ;
 Description: Opens the bank if possible.

FindBank

function FindBank(TheBank: String): Boolean ;
 Description: Finds and opens any given bank in the free world.

OpenDepositBox

function OpenDepositBox;
 Description: Opens the deposit box.

CAutoRespond.scar

SetupSRLAutoResponder

procedure [SetupSRLAutoResponder](#);

Description:

Respond

function [Respond](#);

Description:

Inventory.scar

GetInven

function GetInven(Inven: Integer): TInvenItem ;

Description: Returns The Width Height Count ItemBox and CenterPoint of the

GetInventoryArray

function GetInventoryArray;

Description: Returns The Width Height Count ItemBox and CenterPoint

InvBox

function InvBox(i : Integer): TBox ;

Description: Returns TBox of surrounding Inventory SlotBox

ItemCoords

function ItemCoords(i: Integer): TPoint ;

Description: Returns X and Y of Specified Inventory Spot

CoordsToItem

function CoordsToItem(X Y: Integer): Integer ;

Description: Returns which Inventory slot the X and Y coords are in. If the

GetInvItemBounds

function GetInvItemBounds(InvSpot: Integer; T: TBox): Boolean ;

Description: If item exists returns TBox of the item else returns the normal InvBox.

ExistsItem

function `ExistsItem`(`i`: `Integer`): Boolean ;

Description: Checks if item in inventory at specified position exists

ExistsItemDTM

function `ExistsItemDTM`(`dtm`: `Integer`; `var x y`: `Integer`): Boolean ;

Description: Checks if the item with dtm is in the inventory. Returns the item's

InvMouse

function `InvMouse`(`InvSlot` `Action`: `Byte`): boolean ;

Description: A combined MouseItem & MMouseItem will hopefully replace both

MMouseItem

procedure `MMouseItem`;

Description: Moves mouse to specified inventory spot.

MouseItem

procedure `MouseItem`;

Description: Moves Mouse to inv spot then clicks left or right.

DragItem

procedure `DragItem`;

Description: Drags Item from first position Inv1 to Inv2 (Inventory Slots)

InvCount

function `InvCount`;

Description: Returns amount of items in your inventory

InvEmpty

function `InvEmpty`;

Description: Returns True if inventory is empty

InvFull

function `InvFull`;

Description: Returns True if inventory is full

CountItems

function `CountItems`(`ItemType`: `string`; `Identifier`: `Integer`; `tol`: `TIntegerArray`): `Integer` ;

Description: Counts Items in the inventory.

ClickAllItemsExcept

function `ClickAllItemsExcept`(`ItemType`: `String`; `Identifier`: `Integer`; `option`: `string`; `SlotIgnores`: `TIntegerArray`; `waitnum`: `Integer`; `tol`: `TIntegerArray`): `Integer` ;

Description: Performs "option" popup menu action on all items with:

ClickAllItems

function `ClickAllItems`(`ItemType`: `String`; `Identifier`: `Integer`; `option`: `string`; `waitnum`: `Integer`; `tol`: `TIntegerArray`): `Integer` ;

Description: See ClickAllItemsExcept this doesn't ignore any slots.

DropItem

procedure `DropItem`;

Description: Drops item at given position (1-28)

DropArray

procedure `DropArray`;

Description: Drops item positioned equivalent to numbers in "InvSlots" array.

DropPattern

procedure `DropPattern`;

Description: Drops all items in inventory according to pattern which.

DropAll

procedure `DropAll`;

Description: Drops all items in inventory.

DropAllExcept

procedure `DropAllExcept`;

Description: Drops everything in inventory. Ignores slots specified by

DropItemsByIdentifier

procedure [DropItemsByIdentifier](#);

Description: See description of ClickAllItemsExcept.

ArrangeInv

procedure [ArrangeInv](#);

Description: Arrange's inventory in an orderly fashion.

CountItemsBlackLine

function [CountItemsBlackLine](#)([Count](#) [Tol](#): [Integer](#)): Integer ;

Description: Returns the number of items in inventory with Count amount of

ClickAllItemsBlackLine

function [ClickAllItemsBlackLine](#)([Count](#) [Tol](#): [Integer](#); [Option](#): [String](#)): Integer ;

Description: Finds all items in inventory with blackline points amount of Count

ItemActivated

function [ItemActivated](#)([Slot](#): [Integer](#)): Boolean ;

Description: Checks if item in specified inventory slot has a white outline.

SelectedItem

function [SelectedItem](#);

Description: Returns the selected item in inventory (if any) else returns -1.

Globals.scar

LoadSRLBitMaps

procedure [LoadSRLBitMaps](#);
Description: Loads SRL Bitmaps into memory.

FreeSRLBitMaps

procedure [FreeSRLBitMaps](#);
Description: Frees SRL Bitmaps from memory.

LoadSRLFonts

procedure [LoadSRLFonts](#);
Description: Loads SRL Fonts into memory.

FreeSRLFonts

procedure [FreeSRLFonts](#);
Description: Frees SRL Fonts from memory.

srl_Warn

procedure [srl_Warn](#);
Description: WriteLn's a warning message.

SetupBoxes

procedure [SetupBoxes](#);
Description: Sets up Box variables

Symbol.scar

LoadSymbolBitmapColor

function LoadSymbolBitmapColor(Name: String): Integer ;
Description: Loads Bitmap and Color of a Specific Symbol.

GetSymbolColorIn

function GetSymbolColorIn(var rx ry: Integer; Name: string; xs ys xe ye: Integer): Integer ;
Description: Returns the Color of the Chosen Symbol Searching in a selected area

GetSymbolColor

function GetSymbolColor(var rx ry: Integer; Name: string): Integer ;
Description: Returns the Color of the Chosen Symbol searching the whole Minimap

FindSymbol

function FindSymbol(var rx ry: Integer; Name: String): Boolean ;
Description: Finds a minimap Symbol and returns a Boolean if True.

FindSymbolIn

function FindSymbolIn(var rx ry: Integer; Name: string; xs ys xe ye: Integer): Boolean ;
Description: Finds a minimap Smbol and returns a Boolean if True.

IsPointInTPA

function IsPointInTPA(Point: TPoint; TPA: TPointArray): Boolean ;
No Description

FindSymbolsIn

function FindSymbolsIn(**var** AnsTPA: TPointArray; SymbolName: string; x1 y1 x2 y2: integer): Boolean ;

Description: Finds a symbol in multiple places within the search coords. Results

FindSymbols

function FindSymbols(**var** AnsTPA: TPointArray; SymbolName: string): Boolean ;

Description: Finds a symbol in multiple places searching the whole minimap.

FindSymbolsMultiIn

function FindSymbolsMultiIn(**var** aTPA: TPointArray; Symbols: TStringArray; x1 y1 x2 y2: integer): Boolean ;

Description: Finds multiple symbols the specified area. Results true if at least

FindSymbolsMulti

function FindSymbolsMulti(**var** aTPA: TPointArray; Symbols: TStringArray): Boolean ;

Description: Finds multiple symbols in multiple places. Results true if at least

Text.scar

TypeByte

procedure `TypeByte`;

Description: Types one char.

TypeSendEx

procedure `TypeSendEx`;

Description: Sends human like text.

TypeSend

procedure `TypeSend`;

Description: Sends human like text.

AddMistakes

function `AddMistakes`(`Orig`: `string`; `Chance`: `Integer`): `string` ;

Description: Adds human mistakes to Orig such as mistypes missing letters

GetUpText

function `GetUpText`;

Description: Returns text in upperleft corner.

IsUpText

function `IsUpText`(`UpText`: `String`): `Boolean` ;

Description: Returns up text that might be to the right

IsUpTextMultiCustom

function `IsUpTextMultiCustom`(`Text`: `TStringArray`): `Boolean` ;

Description: Checks for the occurance of any of the strings in the array.

GetLevelAtPos

function `GetLevelAtPos`(`x y`: `Integer`): `integer` ;

Description: Returns the level of the character at (x y) uses uptext. Returns

FindText

function `FindText`(`var cx cy`: `Integer`; `txt`: `String`; `font xs ys xe ye`: `Integer`): `Boolean` ;

Description: Searches for text in specified box with specified font.

ClickText

function `ClickText`(`Text`: `string`; `Chars x1 y1 x2 y2`: `Integer`; `Left`: `Boolean`): `Boolean` ;

Description: Clicks text if found in box.

LoadTextTPA

function `LoadTextTPA`(`Text` : `String`; `Chars` : `Integer`; `var height` : `integer`): `TPointArray` ;

Description: Loads a TPA of the text.. It also returns the real height

ChooseOptionMulti

function `ChooseOptionMulti`(`Text`: `TStringArray`): `Boolean` ;

Description: Finds Popup menu then clicks on it.

ChooseOptionEx

function `ChooseOptionEx`(`Txt TextType`: `string`): `Boolean` ;

Description: Finds an option in a popup menu then clicks on it. Searches for

ChooseOption

function `ChooseOption`(`txt`: `String`): `Boolean` ;

Description: Finds an option in a popup menu then clicks on it.

OptionsExist

function `OptionsExist`(`txt`: `TStringArray`; `Move`: `boolean`): `boolean` ;

Description: Returns true if an Option in txt is found in the RS Option Menu

CloseWindow

function `CloseWindow`;

Description: Closes any open windows.

InStrArrEx

function `InStrArrEx`(`S`: `Sstring`; `StrArr`: `TStringArray`; `var Where`: `Integer`):
Boolean ;

Description: Returns true if the string S was found in the StrArr and if so

StrInArr

function `StrInArr`(`str`: `string`; `arrS`: `TStringArray`): Boolean ;

Description: Returns True if str is found in the Array. Case sensitive. Uses Pos

ArrInStr

function `ArrInStr`(`arrS`: `TStringArray`; `str`: `string`): Boolean ;

Description: Returns True if the Array is found in the str. Case sensitive. Uses Pos

SetDebugColors

procedure `SetDebugColors`;

Description: Sets the next WriteLn's colours to Text and Back respectively.

WriteArrayCols

procedure `WriteArrayCols`;

Description: Writes a TStringArray with Text and Back colours. If not using

Simba.simba

Amount.scar

TradeScreen

function TradeScreen;

Description: Returns 1 or 2 for the respective tradescreens 0 if neither is open.

ShopScreen

function ShopScreen;

Description: Returns true if a shop interface is opened.

FindItemEx

function FindItemEx(var x y: Integer; IdentType: string; Ident: Integer; Area: TBox; Tol: TIntegerArray): Boolean ;

Description: Finds an item in Area specified by TBox in Area. Valid IdentTypes are:

FindItem

function FindItem(var x y: Integer; IdentType: string; Ident x1 y1 x2 y2: Integer; Tol: TIntegerArray): Boolean ;

Description: Finds an item in Area specified by x1 y1 x2 y2 in the Area.

GetAmount

function GetAmount(ItemX ItemY: Integer): Integer ;

Description: Returns the amount of an item at coordinates (ItemX ItemY).

AreaInfo

procedure AreaInfo;

Description: Returns info on certain interfaces.

CheckArea

function CheckArea(area: String): Boolean ;
 Description: Checks if the given interface is opened.

ItemCoordinates

function ItemCoordinates(Area ItemType: string; Item Tol: TIntegerArray): TPointArray ;
 Description: Returns a TPA with the positions of all occurrences of Item.

CountItemsIn

function CountItemsIn(Area ItemType: string; Item: Integer; Tol: TIntegerArray): Integer ;
 Description: Counts the number of items found within the Area. (Does not count stacks)

CountItemsArea

function CountItemsArea(area: String): Integer ;
 Description: Counts items in a specified area. (Doesn't count stacks)

ItemAmount

function ItemAmount(Area ItemType: string; Item: Integer; Tol: TIntegerArray): Integer ;
 Description: Counts the number of items found with a color in a certain area. (Counts stacks)

FindCoins

function FindCoins(var X Y: Integer; Area: string): Boolean ;
 Description: Finds coins in Area and stores the coords in x and y.

CoinAmount

function CoinAmount(Area: String): Integer ;
 Description: Returns the amount of coins in a specified interface.

LoadCoinBmp

function LoadCoinBmp;
 Description: Returns a bitmap to use for coins.

RuneAmount

function RuneAmount(area runetype: String): Integer ;

Description: Returns the amount of a certain rune in the specified area.

ShopSwitchTab

procedure ShopSwitchTab;

Description: Switches the shop tab to the one you want only moves mouse if

Login.scar

RSReady

function [RSReady](#);

Description: Returns true if we are ready to auto (on loginscreen or logged in).

Login_SetAudio

procedure [Login_SetAudio](#);

Description: Sets audio options through the login screen.

Login_SetGraphics

procedure [Login_SetGraphics](#);

Description: Sets graphic options in the login screen.

SetAudioOff

procedure [SetAudioOff](#);

Description: Sets the Audio to off (1 for each bar). Works logged in and out.

SetAutoingDefaults

procedure [SetAutoingDefaults](#);

Description: Sets the graphic options for best SRL compatibility. Works both logged in and out.

LobbyScreen

function [LobbyScreen](#);

Description: Returns true if we are at the Lobby Screen

ExitToLobby

function [ExitToLobby](#);

Description: Exits to the Lobby (for breaks?)

LeaveLobby

function `LeaveLobby`;

Description: Logs you out from lobby screen.

Logout

function `Logout`;

Description: Logs you all the way out from ingame.

LoginPlayerToLob

function `LoginPlayerToLob`;

Description: Like LoginPlayer but only to the lobbyscreen.

LoginPlayer

procedure `LoginPlayer`;

Description: Logs in the Player[CurrentPlayer]. Detects most Client Login Errors

NextPlayerIndex

function `NextPlayerIndex`;

Description: Returns the player number of the next active player.

RandomPlayerIndex

function `RandomPlayerIndex`;

Description: Returns the Player number of a random active player.

SwitchToPlayer

function `SwitchToPlayer`(PlayerNo: Integer; Active: Boolean): Boolean ;

Description: Switches to a specific player setting the current player's

NextPlayer

procedure `NextPlayer`;

Description: Logs in the next player that isn't inactive.

RandomNextPlayer

procedure `RandomNextPlayer`;

Description: Logs in a random player that isn't inactive.

CheckUserNicks

procedure [CheckUserNicks](#);

Description: Checks if all nicks are set correct.

Timing.scar

MsToTime

function `MsToTime`(`MS StrType`: `Integer`): string ;

Description: Takes MS in milliseconds and outputs a string with hours mins and

TimeRunning

function `TimeRunning`;

Description: Returns Time since the script was started (GetTimeRunning).

MarkTime

procedure `MarkTime`;

Description: Sets TimeMarker to current system time

TimeFromMark

function `TimeFromMark`(`TimeMarker`: `Integer`): Integer ;

Description: returns Milliseconds since MarkTime was set

TheTime

function `TheTime`;

Description: Returns current time as a string

TheDate

function `TheDate`(`DateFormat` : `Integer`): String ;

Description: TheDate will return the current date. DateFormats can be...

WaitOptionMulti

function `WaitOptionMulti`(`S`: `TStringArray`; `Time`: `Integer`): Boolean ;

Description: Waits for a TStringArray of options and selects one of them

WaitOptionEx

function WaitOptionEx(S TextType: String; Time: Integer): Boolean ;
 Description: Waits for an Option and selects it. Searches for TextType.

WaitOption

function WaitOption(S: String; Time: Integer): Boolean ;
 Description: Waits for an Option and selects it

WaitUpTextMulti

function WaitUpTextMulti(S: TStringArray; Time: integer): Boolean ;
 Description: Waits for a TStringArray of UpText returns true if found

WaitUptext

function WaitUptext(S: String; Time: Integer): Boolean ;
 Description: Waits for an UpText returns true if found

WaitInvMaxCount

function WaitInvMaxCount(Count MaxTime: integer): boolean ;
 Description: Waits for a maximum inv count. Returns true if InvCount $\geq$ Count

WaitInvMinCount

function WaitInvMinCount(Count MaxTime: integer): boolean ;
 Description: Waits for a minimum inv count. Returns true if InvCount $\leq$ Count

WaitColor

function WaitColor(x y Color Tol MaxTime: integer): Boolean ;
 Description: Waits for a color at (x y) with tolerance Tol returns true if found

WaitFindColor

function WaitFindColor(var x y: integer; Color Tol x1 y1 x2 y2 MaxTime: integer): Boolean ;
 Description: Waits for a color at (x y) with tolerance Tol returns true if found

WaitFindColors

function WaitFindColors(var x y: integer; Color Tol x1 y1 x2 y2 MaxTime: integer): Boolean ;

Description: Waits for a color at (x y) with tolerance Tol returns true if found

WaitColorCount

function WaitColorCount(Color x1 y1 x2 y2 Tol MinCount MaxCount MaxTime: integer): boolean ;

Description: Waits Color count in box (x1 y1 x2 y2) with Tol

WaitColorGone

function WaitColorGone(Color x y Tol MaxTime: integer): Boolean ;

Description: Waits until a colour is gone at (x y) with tolerance and a timeout.

WaitColorGoneIn

function WaitColorGoneIn(Colour x1 y1 x2 y2 Tol MaxTime: integer): Boolean ;

Description: Waits until a colour is gone in an area with tolerance and a timeout.

function (Func: Function: Boolean; WaitPerLoop MaxTime: Integer): Boolean ;

Description: Waits for function Func to be true. WaitPerLoop is how often you

Defines.scar

Animation.scar

PixelShiftMulti

function PixelShiftMulti(B: TBoxArray; T: Integer): TIntegerArray ;
Description: Calculates how many pixels change in boxes B during time T.

PixelShift

function PixelShift(B: TBox; Time: Integer): Integer ;
Description: Calculates how many pixels change in box B during time T.

AnimatingMulti

function AnimatingMulti(B: TBoxArray; Time MinCount: Integer): TBooleanArray ;
Description: Finds constant animation in boxes B during time T. If PixelShift

Animating

function Animating(B: TBox; Time MinCount: Integer): Boolean ;
Description: Finds constant animation in box B during time T. If PixelShift

IsMoving

function IsMoving;
Description: Results true if character is moving.

MapWalk.scar

MakeCompass

function `MakeCompass`(`Direction`: `Variant`): Boolean ;
Description: Aligns RS to specified direction

SetAngle

procedure `SetAngle`;
Description: Sets the mainscreen at highest
lowest angle (Depends on the boolean)

ClickNorth

procedure `ClickNorth`;
Description: Clicks the compass to make it perfect North. Will set angle at

GetWallAngle

function `GetWallAngle`(`Ps`: `TPointArray`): Extended ;
Description: Get's the Minimap angle using Walls. Returns Angle in Radians.

RunAway

procedure `RunAway`;
Description: Runs away in minimap related direction based on north.

StoreToRoadColorArray

procedure `StoreToRoadColorArray`;
Description: Stores RoadColor to an array. Debugging and logging purposes.

GetOldRoadColors

function `GetOldRoadColors`;
Description: Checks if RoadColor matches with any color stored in RoadColors

GetNewRoadColor

function `GetNewRoadColor(xs ys xe ye tol: Integer): Boolean` ;
 Description: Searches for new RoadColor in box xs ys xe ye by comparing

RoadColorChecker

function `RoadColorChecker;`
 Description: Checks for presence of RoadColor. If none is found it will try

LinearWalkEx

function `LinearWalkEx(var TPA: TPointArray; cx cy TheColor tol Direction Radius: Integer): Boolean` ;
 Description: Finds TheColor from Radial (scanning outwards) for Radius Distance.

RadialWalkEx

function `RadialWalkEx(var TPA: TPointArray; cx cy TheColor tol StartRadial EndRadial Radius: Integer): Boolean` ;
 Description: Finds TheColor from StartRadial to EndRadial for Radius Distance.

RadialWalk

function `RadialWalk(TheColor StartRadial EndRadial Radius Xmod Ymod: Integer): Boolean` ;
 Description: Walks TheColor from StartRadial to EndRadial for Radius Distance

LinearWalk

function `LinearWalk(TheColor Direction Radius Xmod Ymod: Integer): Boolean` ;
 Description: Walks TheColor from Direction MIDDLE OUT!by performing windscreenlike scanning movements for Radius Distance

LinearWalkTolerance

function `LinearWalkTolerance(TheColor Direction Radius Xmod Ymod Tol: Integer): Boolean` ;
 Description: Same as LinearWalk but with a tolerance.

RadialRoadWalk

function `RadialRoadWalk`(`TheColor` `StartRadial` `EndRadial` `Radius` `Xmod` `Ymod`: `Integer`): `Boolean` ;
 Description: Walks `TheColor` from `StartRadial` to `EndRadial` for `Radius` Distance

RadialWalkTolerance

function `RadialWalkTolerance`(`Color` `sRadial` `eRadial` `r` `modX` `modY` `Tol`: `Integer`): `Boolean` ;
 Description: Same as `RadialWalk` but with a tolerance.

LinearRoadWalk

function `LinearRoadWalk`(`TheColor`: `Integer`; `Direction`: `Integer`; `Radius`: `Integer`; `Xmod` `Ymod`: `Integer`): `Boolean` ;
 Description: Walks `TheColor` from `Direction` MIDDLE OUT!by performing windscreenlike scanning movements for `Radius` Distance

WaterWalk

function `WaterWalk`(`StartRadial` `EndRadial` `Radius` `FFlagBreakOut` `Xoff` `Yoff` `Xmod` `Ymod`: `Integer`): `Boolean` ;
 Description: Walks alongside Water.

GetMiniMapDotsIn

function `GetMiniMapDotsIn`(`WhatDot`: `String`; `x1` `y1` `x2` `y2`: `Integer`): `TPointArray` ;
 Description: Results the dots specified by `WhatDot` in `x1` `y1` `x2` `y2`.

GetMiniMapDots

function `GetMiniMapDots`(`WhatDot`: `String`): `TPointArray` ;
 Description: Results the dots specified by `WhatDot` on the Minimap. See `GetMiniMapDots`

CountDots

function `CountDots`(`WhatDot`: `String`): `integer` ;
 Description: Counts the occurance of `WhatDot` on the MiniMap. See `GetMiniMapDots`

MouseMM

procedure [MouseMM](#);

Description: Clicks (x y) on the MM taking Map Angle into Account.

Mouse.scar

WindMouse

procedure [WindMouse](#);
Description:

MMouse

procedure [MMouse](#);
Description: Moves the mouse.

MouseAction

function [MouseAction](#)([left](#): [boolean](#)): integer ;
Description: Converts the (SCAR) boolean for mouse button the the Simba integer for the button

ClickMouse2

procedure [ClickMouse2](#);
Description: Clicks the mouse in a human way on the current mouse spot.

Mouse

procedure [Mouse](#);
Description: Moves then clicks mouse.

SleepAndMoveMouse

procedure [SleepAndMoveMouse](#);
Description: Waits for specified time and moves mouse around like bored human would.

IdleTime

procedure IdleTime;

Description: Randomly moves the mouse (Rand and Gravity) for Time milliseconds

SpiralMouseMulti

function SpiralMouseMulti(var fx fy: Integer; x1 y1 x2 y2: Integer; UpText: TStringArray; PpC aInc: Integer): Boolean ;

Description: Spirals the mouse in a circle from the center of the defined box.

SpiralMouse

function SpiralMouse(var fx fy: Integer; x1 y1 x2 y2: Integer; UpText: String; PpC aInc: Integer): Boolean ;

Description: Spirals the mouse in a circle from the center of the defined box.

DidClick

function DidClick(Red: Boolean; Time: integer): boolean ;

Description: Checks for the X after a click Times Out after Time.

DidRedClick

function DidRedClick;

Description: Checks for the Red X after a Click Times out after 500 ms

DidYellowClick

function DidYellowClick;

Description: Checks for the Yellow X after a Click Times out after 500 ms

MouseOval

procedure MouseOval;

Description: Clicks a random point within an oval shape. The size of the oval is

MouseCircle

procedure MouseCircle;

Description: Same as MouseOval but clicks within a circle instead.

MouseBoxEx

procedure [MouseBoxEx](#);

Description: Does mouse stuff (ClickType) and uses a point that is within Dist from

DragMouse

procedure [DragMouse](#);

Description: Leftclicks StartX StartY and drags mouse to EndX EndY.

WorldSwitcher.scar

WorldScreen

function [WorldScreen](#);

Description: Returns true if the World screen is open.

OpenWorldScreen

function [OpenWorldScreen](#);

Description: Opens the world screen. Returns true if world screen is opened.

UpdateWorlds

procedure [UpdateWorlds](#);

Description: Parses the worlds page into Worlds.INI

LoadWorldArrays

procedure [LoadWorldArrays](#);

Description: Loads the world array into WorldArray from Worlds.ini

FindWorld

function [FindWorld](#)([W](#): [integer](#)): Boolean ;

Description: Scrolls to area and returns true if world was found

GetWorldInfo

function [GetWorldInfo](#)([World](#): [Integer](#); [var WorldRec](#): [TWorld](#)): Boolean ;

Description: Records World info into TWorld record also Results true if world

SelectWorld

function [SelectWorld](#)([W](#): [Integer](#)): Boolean ;

Description: Switches to the specified world. Returns true if successful.

RandomWorld

function RandomWorld(Members PvP: Boolean): Integer ;

Description: Returns a random world by paramaters.

ChangeWorld

function ChangeWorld(W: Integer): Boolean ;

Description: Changes to world W. Returns True if successful.

OrderWorlds

procedure OrderWorlds;

Description: Orders worlds by 'by' in order 'Descending'

ChooseWorldBy

function ChooseWorldBy(SettingsNames: TStringArray; SettingsDescending: array of boolean; Members PVP: boolean): boolean ;

Description: Chooses a world by SettingsNames ordered by SettingsDescending

Chat.scar

SetChat

function SetChat(**state**: string; **chat**: Integer): Boolean ;
Description: Sets Chat to desired state

SetAllChats

procedure SetAllChats;
Description: Sets all the chats to a particular state in one simple procedure.

ChatsOff

procedure ChatsOff;
Description: Sets all Chats to off.

PleaseWait

function PleaseWait;
Description: Returns true if 'Please wait...' is in chat

ChatWait

procedure ChatWait;
Description: Waits while there's Please Wait in the chatbox.

ClickContinue

function ClickContinue(**Click Wait**: Boolean): Boolean ;
Description: Returns true if the blue 'Click To Continue' text in the chat box

ClickToContinue

function ClickToContinue;
Description: Click the blue 'Click To Continue' text in the chat box.

TextCoords

function `TextCoords`(`textn`: `Integer`): `TPoint` ;

Description: Returns x y for text. (1 to 8)

GetChatBoxText

function `GetChatBoxText`(`Line` `TextCol`: `Integer`): `string` ;

Description: Gets text on the line Line with colour TextCol. (Line 1 is the top

FindChatBoxText

function `FindChatBoxText`(`Text`: `string`; `Line`: `Integer`; `TextCol`: `Integer`):
`Boolean` ;

Description: Searches for Text on Line with colour TextCol. True if found.

IsChatBoxTextBetween

function `IsChatBoxTextBetween`(`Text`: `string`; `TextCol` `StartLine` `EndLine`:
`Integer`): `Boolean` ;

Description: Returns true if Text with colour TextCol is found anywhere in the

IsChatBoxTextAnyLine

function `IsChatBoxTextAnyLine`(`Text`: `string`; `TextCol`: `Integer`): `Boolean` ;

Description: Returns true if Text with colour TextCol is found anywhere in the

ChatBoxTextArray

function `ChatBoxTextArray`(`TextCol`: `Integer`): `TStringArray` ;

Description: Gets and return a TStringArray with the contents of each line with

GetBlackChatMessage

function `GetBlackChatMessage`;

Description: Gets text with color 0 in last chat line.

FindBlackChatMessage

function `FindBlackChatMessage`(`ChatMsg`: `String`): `Boolean` ;

Description: Returns True if text with color 0 in last chat line is found.

LastChatter

function `LastChatter`(`var name`: `String`): `Boolean` ;

Description:Returns name and true if a person was talking or false and the text

GetLastChatText

function GetLastChatText(**var** chat: String): Boolean ;

Description: Returns true if blue chat text was found

InChat

function InChat(Text: String): Boolean ;

Description: Grabs Last Chat Line and performs checks on occurrence of text

FixChat

procedure FixChat;

Description: Scrolls ChatText scrollbar down.

SaveToChatLog

function SaveToChatLog;

Description: Saves chat to log file

ColorEffectAdd

function ColorEffectAdd(Color Effect: Boolean): string ;

Description: Returns a RS chat effect based on 'Color' and 'Effect' booleans.

Bitmaps.scar

srl_LoadBitmaps

procedure [srl_LoadBitmaps](#);

Description: Preloads bitmaps identified by ID in the srl bitmaparray.

srl_LoadBitmapRange

procedure [srl_LoadBitmapRange](#);

Description: Preloads bitmaps identified by ID in range StartID to EndID

srl_FreeBitmapEx

procedure [srl_FreeBitmapEx](#);

Description: Frees SRL bitmap which

srl_FreeBitmapArray

procedure [srl_FreeBitmapArray](#);

Description: Frees an array of assigned SRL bitmaps determined by bmpArr.

srl_FreeBitmaps

procedure [srl_FreeBitmaps](#);

Description: Frees all SRL bitmaps

srl_FreeBitmapRange

procedure [srl_FreeBitmapRange](#);

Description: Frees all SRL bitmaps with IDs ranging from StartID to EndID.

Overwrite.scar

MouseBox

procedure `MouseBox`;

Description: Moves mouse into a random position in the box. Clicks if told to.

RBoolEx

function `RBoolEx`(`chance`: `integer`): `boolean` ;

Description: Returns a `boolean` with a chance of 1 / `chance`

RBool

function `RBool`;

Description: Returns a `boolean` with a 50

RC.scar

RC_SetStandards

procedure [RC_SetStandards](#);

Description: Sets your account settings so that you can use the

RCReadCommand

function [RCReadCommand](#);

Description: Returns string of red typed message (color 128)

RC_FixFL

procedure [RC_FixFL](#);

Description: Scrolls to the top of the Friend list.

RC

function [RC](#);

Description: High Level Remote Control Command. Handles Remote Control.

SetupRC

procedure [SetupRC](#);

Description: Sets up everything needed to use RC.scar.

SRLLog.scar

InitializeSRLLogFile

function [InitializeSRLLogFile](#);

Description: Sets the SRLSavePath variable by looping through a few folders.

AddToSRLLog

procedure [AddToSRLLog](#);

Description: Adds a line to the SRL Log file.

FreeSRLLog

procedure [FreeSRLLog](#);

Description: Frees the opened SRL logfile.

Certer.scar

InCerterRandom

function [InCerterRandom](#);

Description: Checks if we're in the Certer random.

CerterScreen

function [CerterScreen](#);

Description: Returns True if the Certer screen is open.

LoadCerterItems

procedure [LoadCerterItems](#);

Description: Loads Certer items information.

GetItemInfo

procedure [GetItemInfo](#);

Description: Gets the item's Count Width Height Area and Percentage Cover.

GetItem

function [GetItem](#);

Description: Gets the item and returns it name.

ExitPortal

function [ExitPortal](#);

Description: Finds and clicks on the exit portal.

LeaveCerter

function [LeaveCerter](#);

Description: Leaves the certer area returns True if successful.

FindCarter

function FindCarter;

Description: Finds Carter-guy and talks to him.

SolveCarter

function SolveCarter;

Description: Solves the Carter Random.

PrisonPete.scar

InPrisonPete

function [InPrisonPete](#);

Description: Checks if we're in the Prison Pete Random.

SolvePrisonPete

function [SolvePrisonPete](#);

Description: Solves the prison pete random!

Molly.scar

FindMollyDoor

function FindMollyDoor(DP : TPoint; SortBy : Integer): Boolean ;
Description: Opens the door in molly.

Molly_FindMS

procedure Molly_FindMS;
Description: Finds the position of molly on the main screen

IsMollyOnDot

function IsMollyOnDot(CP : TPoint): Boolean ;
Description: Results true if molly is standing on dot.

Molly_Operate

procedure Molly_Operate;
Description: Clicks a direction based on the input

Molly_MoveCrossTo

procedure Molly_MoveCrossTo;
Description: Moves the crosshair to molly

SolveMolly

function SolveMolly;
Description: Solves the Molly Random

Abyss.scar

InAbyss

function [InAbyss](#);

Description: Searches the Minimap for a black count. If in range we are in the

Abyss_WalkToCenter

procedure [Abyss_WalkToCenter](#);

Description: Walks to center and aligns everything high angle and compass
45deg

Abyss_ClickLever

procedure [Abyss_ClickLever](#);

Description: Clicks lever if it didn't find a unique it guesses.

SolveAbyss

function [SolveAbyss](#);

Description: Solves the Abyss Random.

BeeKeeper.scar

BeeScreen

function [BeeScreen](#);

Description: Returns True if the Bee Hive Construction screen is open.

SolveBeeKeeper

function [SolveBeeKeeper](#);

Description: Solves the Bee Keeper Random.

CapnArnav.scar

arnav_ClickArrow

procedure [arnav_ClickArrow](#);

Description: Clicks the arrow at Col Count number of times.

SolveArnav

function [SolveArnav](#);

Description: Solves the Cap'n Arnav random.

Leo.scar

Leo_LoadDTMs

procedure [Leo_LoadDTMs](#);

Description: Loads Leo Solver DTMs and Bitmaps etc.

InLeoGraveyard

function [InLeoGraveyard](#);

Description: Checks if we're in the Leo random.

Leo_WaitForScreen

procedure [Leo_WaitForScreen](#);

Description: Waits for a gravestone/coffin screen.

Leo_ToCenter

procedure [Leo_ToCenter](#);

Description: Walks to the center of the graveyard

Leo_GrabCoffins

procedure [Leo_GrabCoffins](#);

Description: Grabs all coffins from the screen.

SolveLeo

function [SolveLeo](#);

Description: Solves the Leo random.

Mime.scar

Mime_ClickOption

procedure [Mime_ClickOption](#);

Description: Uses the Action to and clicks on it.

SolveMime

function [SolveMime](#);

Description: Solves the Mime Random.

Sandwich.scar

SolveSandwichLady

function [SolveSandwichLady](#);

Description: Solves the Sandwich Lady random.

Maze.scar

InMaze

function [InMaze](#);

Description: Checks if we're in the Maze Random.

MazRotPoint

procedure [MazRotPoint](#);

Description: Get's map angle using walls and fixes the given point

FindMazeDoor

function [FindMazeDoor](#)(**var** x y: [Integer](#); **MaxDistFromCenter**: [Integer](#)): Boolean
;

Description: Finds the Maze door on the main screen.

TouchShrine

function [TouchShrine](#);

Description: Touches the shrine at the end of the maze.

GetStartPosition

function [GetStartPosition](#);

Description: Returns where we started in the maze.

LoadMazePaths

procedure [LoadMazePaths](#);

Description: Loads the paths we can walk.

WalkMazePath

procedure [WalkMazePath](#);

Description: Follows the paths from the maze.

142

SolveMaze

function [SolveMaze](#);

Description: Main maze solving routine.

Mordaut.scar

InMordaut

function [InMordaut](#);

Description: Finds if your in Mr.Mordaut random

ExitMordaut

function [ExitMordaut](#);

Description: Exits the classroom!

M_BigPuzzle

procedure [M_BigPuzzle](#);

Description: Solves the large puzzle choose three common themed.

SolveMordaut

procedure [SolveMordaut](#);

Description: Solves the Mordaut random.

Pillory.scar

SolvePilloryPuzzle

function [SolvePilloryPuzzle](#);

Description: Solves the pillory random.

Quiz.scar

InQuizRandom

function `InQuizRandom`;

Description: Returns true if the quiz random is found.

QuizItemBox

function `QuizItemBox`(`Index`: `Integer`): TBox ;

Description: Returns a TBox of the Quiz Item Slow defined by Index.

LoadQuizArray

procedure `LoadQuizArray`;

Description: Loads all the data.

GetQuizObjectInfo

function `GetQuizObjectInfo`(`Index`: `Integer`): QuizObj ;

Description: Get's the data of a QuizObj once.

GetQuizObjectInfoAverage

function `GetQuizObjectInfoAverage`(`Index` `Times`: `Integer`): QuizObj ;

Description: Get's the average data of a QuizObj

QuizObjStats

procedure `QuizObjStats`;

Description: WriteLn's a QuizObj's data.

GetQuizItem

function `GetQuizItem`(`Index`: `Integer`): QuizObj ;

Description: Identifies a Quiz Item.

SolveQuizQuestion

procedure [SolveQuizQuestion](#);

Description: Solves one question of the quiz.

SolveQuiz

function [SolveQuiz](#);

Description: Solves the quiz random and returns true if solved.

FindQuiz

function [FindQuiz](#);

Description: Solves the quiz random and returns true if solved.

Demon.scar

NoSignObstruction

function NoSignObstruction;

Description: Checks if the signs are fully visible.

DemonCenter

function DemonCenter;

Description: Walks to the center of the random.

IdentifySigns

procedure IdentifySigns;

Description: Gets the actions on the signs.

PerformTask

function PerformTask(TaskIndex: Integer): Boolean ;

Description: Performs the correct workout.

GetSigns

function GetSigns(var JogIndex StarIndex SitIndex PushIndex: Integer): Boolean
;

Description: Handles the getting of the signs.

SolveDemon

function SolveDemon;

Description: Main solving procedure.

RandomTool.scar

r_ItemClicked

procedure [r_ItemClicked](#);

Description: Performs the desired action

r_InitForm

procedure [r_InitForm](#);

Description:

r_SafeInitForm

procedure [r_SafeInitForm](#);

Description:

r_ShowFormModal

procedure [r_ShowFormModal](#);

Description:

r_SafeShowFormModal

procedure [r_SafeShowFormModal](#);

Description:

r_ResetAll

procedure [r_ResetAll](#);

Description: Resets all vars.

r_RunAlert

procedure [r_RunAlert](#);

Description: Runs the form.

SetupRandomTool

procedure [SetupRandomTool](#);

Description: Sets up RandomTool so it can be used.

AntiRandoms.scar

SRLRandomsReport

procedure [SRLRandomsReport](#);

Description: Displays Randoms Report.

FindBoxOption

procedure [FindBoxOption](#);

Description: Finds the text of options based on the text loaded from the

UseRewardBox

procedure [UseRewardBox](#);

Description: Finds the reward box and opens it getting the reward.

UseRewardBox

procedure [UseRewardBox](#);

Description: Finds the reward box and opens it getting the reward.

FindMod

function [FindMod](#);

Description: Results True if a mod is found then it waits 10-20 mins and logs back in

FindLamp

function [FindLamp](#)(**skill**: [String](#)): Boolean ;

Description: handles Genie. Results True if lamp found and handled.

FindDead

function [FindDead](#);

Description: Checks for text "Oh dear you are dead".

LoadTeleRandoms

procedure [LoadTeleRandoms](#);

Description: Loads the teleport random information for each random.

SolveTeleportRandoms

function [SolveTeleportRandoms](#);

Description: Handles the solving of Teleport randoms. Returns True if one is

FindNonInventoryRandoms

function [FindNonInventoryRandoms](#);

Description: Searches for the common NON-Inventory randoms: Won't switch to GameTab(4)!

FindNormalRandoms

function [FindNormalRandoms](#);

Description: Calls the 'normal' random checks.

Frog.scar

InSecondCave

function [InSecondCave](#);

Description: Returns true if the second larger frog cave is found.

SolveFrogCave

function [SolveFrogCave](#);

Description: Solves the larger frog cave.

SolveFrog

function [SolveFrog](#);

Description: Solves the Frog Random also solves the larger frog cave.

EvilBob.scar

InEvilBobsIsland

function [InEvilBobsIsland](#);
Description: Checks if we're in Evil Bob's Island. }

EB_MiddleIsland

procedure [EB_MiddleIsland](#);
Description: Walks to the middle of the island }

EB_FindPortal

procedure [EB_FindPortal](#);
Description: Finds the portal needed to exit }

SolveEvilBob

function [SolveEvilBob](#);
Description: Solves the EvilBob event }

Forester.scar

ForesterCenter

function `ForesterCenter`(`var x y: Integer`): Boolean ;
Description: Returns position of the center.

PickUpDroppedItem

function `PickUpDroppedItem`;
Description: Picks up a dropped item.

InForesterRandom

function `InForesterRandom`;
Description: Checks if we're in the Freaky Forester Random.

TalkToFreakyForester

function `TalkToFreakyForester`;
Description: Talks to the forester.

KillPheasant

function `KillPheasant`(`Tails: Integer`): Boolean ;
Description: Attempts to kill a pheasant with given tail size.

WarningScreen

function `WarningScreen`;
Description: Results true if there is a warning screen about dropped items.

BringRawPheasant

procedure `BringRawPheasant`;
Description: Brings the raw pheasant to the forester.

LeaveForester

function [LeaveForester](#);

Description: Leaves the random through the portal.

SolveForester

function [SolveForester](#);

Description: Main solving procedure.

Pinball.scar

ExitPinball

function [ExitPinball](#);

Description: Exits the pinball random.

SolvePinball

function [SolvePinball](#);

Description: Attempts to solve the Pinball random.

Common.scar

TakeScreen

procedure [TakeScreen](#);

Description: Takes a screenshot if the global Screenshots is true

InBlack

function [InBlack](#);

Description: Checks for the existence of 4 blackdots in MM indicating a random event has occurred.

PercentBlackMM

function [PercentBlackMM](#);

Description: The percentage of the minimap which is black (colour 0 with tolerance 1).

InConvo

function [InConvo](#);

Description: Returns True if you're in a conversation with an NPC (your name

AreTalking

function [AreTalking](#);

Description: Checks if we're talking.

GetMusic

function [GetMusic](#);

Description: Get the music in the area according to gametab 13.

IntArrayMatch

function IntArrayMatch(Ar1 Ar2: TIntegerArray): Boolean ;

Description: Returns True if Ar1 is the same as Ar2.

AnalyzeObjects

function AnalyzeObjects(Colours Tolerances: T2DIntegerArray; Boxes: TBoxArray; Time: Integer): TRandomObjectArray ;

Description: Analyzes measures and colours of rotating 3D objects in random events.

WaitFindBitmapInBoxArray

function WaitFindBitmapInBoxArray(BMP: Integer; B: TBoxArray; Tolerance Time: Integer): Integer ;

Description: Tries to find bitmap in box array B in Time milliseconds. Returns

CompareInfo

function CompareInfo(Info: TRandomObjectArray; Optimal: TRandomObject; cWidth cHeight cCount cMaxLum cMinLum: Boolean): Integer ;

Description: Compare during-runtime analyzed info to carefully considered optimal

FilterObject

procedure FilterObject;

Description: Self explanatory

DebugTRandomObject

procedure DebugTRandomObject;

Description: Used to debug info gathered from AnalyzeObjects()

AntiBan.scar

RandomRClick

procedure [RandomRClick](#);

Description: Performs random right click to stay active.

HoverSkill

procedure [HoverSkill](#);

Description: Hovers mouse over selected skill to "check exp"

PickUpMouse

procedure [PickUpMouse](#);

Description: Acts just like you picked your mouse off the pad...

RandomMovement

procedure [RandomMovement](#);

Description: Presses random arrow keys and moves the mouse. Able to press 2 keys simultaneously.

BoredHuman

procedure [BoredHuman](#);

Description: Performs 5-15 times:

ExamineInv

procedure [ExamineInv](#);

Description: Chooses a random item in the inventory to examine.

RandomAngle

procedure [RandomAngle](#);

Description: Randomly changes the angle of the compass. Finishes depending

166

on

LevelUp

function [LevelUp](#);

Description: Checks if you've leveled up - click Continue and returns True.

Color.scar

FindDeformed

function FindDeformed(var Xoff Yoff: Integer; BMP a b c d: Integer): Boolean ;
Description: Finds Bitmap in Box a b c d. Returns coords in ObjX and ObjY.

FindColorsPie

function FindColorsPie(var Points: TPointArray; Color Tol: Integer; StartD EndD MinR MaxR: Extended; x1 y1 x2 y2 Mx My: Integer): Boolean ;
Description: Finds all Colors with tolerance in the Pie defined by

DTMRotated

function DTMRotated(DTM: Integer; var x y: Integer; x1 y1 x2 y2: Integer): Boolean ;
Description: First looks for a DTM without rotation then increases the amount

GetColorsBox

function GetColorsBox(x1 y1 x2 y2: Integer; ClearSameColors: Boolean): TIntegerArray ;
Description: Returns an array of all colors within box ((x1 y1) (x2 y2)).

PercentColorMMEx

function PercentColorMMEx(Color Tol StartRadial EndRadial StartRadius EndRadius: Integer): integer ;
Description: Returns the percentage of colour on the minimap section defined by

PercentColorMM

function PercentColorMM(Color Tol: Integer): Integer ;
Description: Returns the percent of a colour on the whole minimap.

FindMMColorsTolerance

function FindMMColorsTolerance(**var** Points: TPointArray; Color Tol: Integer): Boolean ;

Description: Finds all colors within "Tol" tolerance in the minimap circle and

GetColorArrayInfo

function GetColorArrayInfo(C: TIntegerArray; **var** BestColors Tolerances: TIntegerArray; Speed2Mods: TExtendedArray; **var** RGB: array of TIntegerArray; **var** HSL XYZ: array of TExtendedArray): array ;

Description: Takes an array of colours and outputs a multitude of information:

GetBestColor

function GetBestColor(Colors: TIntegerArray; CTS: Integer; **var** Tol: Integer; **var** Speed2Modifiers: TExtendedArray): Integer ;

Description: Gets the best colour to find as well as the tolerance and modifiers

CreateAutoColorInfo

function CreateAutoColorInfo(Colors: TIntegerArray): TAutoColorInfo ;

Description: Creates a TAutoColorInfo from the inputted array of colors.

Math.scar

LoadCoSineArrays

procedure `LoadCoSineArrays`;

Description: Loads arrays for use with Radial- and LinearWalk.

CreateTPAFromText

function `CreateTPAFromText`(`Txt` : `string`; `Chars` : `string`): `TPointArray` ;

Description: Returns the `TPointArray` of the inputted Text. Needs Wizzyplugin

CreateTPAFromText

function `CreateTPAFromText`(`Txt` : `string`; `Chars` : `Integer`): `TPointArray` ;

Description: Returns the `TPointArray` of the inputted Text. Needs Wizzyplugin

GetSplinePt

function `GetSplinePt`(`Points`: `TPointArray`; `Theta`: `Extended`): `TPoint` ;

Description: Returns the point on a spline defined by control points `Points` at `Theta`

MakeSplinePath

function `MakeSplinePath`(`Points`: `TPointArray`; `ThetaInc`: `Extended`): `TPointArray` ;

Description: Returns a spline defined by control points `Points` incrementing theta by `ThetaInc`

MidPoints

function `MidPoints`(`Path`: `TPointArray`; `MaxDist`: `Integer`): `TPointArray` ;

Description: Adds midpoints to `Path` so no distance on it is greater than `MaxDist`

InAbstractBox

function InAbstractBox(x1 y1 x2 y2 x3 y3 x4 y4: Integer; x y: Integer): Boolean ;

Description: Returns true if point x y is in an abstract box defined by x1 y1 x2 y2 x3 y3 x4 y4

inAngle

function inAngle(Origin: TPoint; Angle1 Angle2 Radius1 Radius2: Extended; X Y: Integer): Boolean ;

Description: Returns True if X and Y fall within Radius1 to Radius2 and Angle1 to Angle2

Sine

function Sine(degrees: Integer): Extended ;

Description:

Cose

function Cose(degrees: Integer): Extended ;

Description:

MMToMS

function MMToMS(MM: TPoint): TPoint ;

Description: Turns a Minimap point into a close MS point.

DeleteValueInStrArray

procedure DeleteValueInStrArray;

Description: Deletes value with ValuePosition in "Arr" string array.

DeleteValueInIntArray

procedure DeleteValueInIntArray;

Description: Deletes value with ValuePosition in "Arr" Integer array.

DeleteValueInFloatArray

procedure DeleteValueInFloatArray;

Description: Deletes value with ValuePosition in "Arr" Float array.

DeleteValueInBoolArray

procedure DeleteValueInBoolArray;

Description: Deletes value with ValuePosition in "Arr" Boolean array.

BoxClosestPoint

function BoxClosestPoint(p : TPoint; b : TBox): TPoint ;

Description: There are 'nine' places where the point can be (relative to a box)..

RandomPointBoxEx

function RandomPointBoxEx(p : TPoint; b : TBox; extraDist : integer):
TPoint ;

Description: Returns a point in the box which is 'aimed' near the TPoint given..

MiddleBox

function MiddleBox(b : TBox): TPoint ;

Description: Returns the middle of the box.

GameTab.scar

GetCurrentTab

function `GetCurrentTab`;

Description: Returns current tab.

TabExists

function `TabExists`(`Tab`: `Integer`): Boolean ;

Description: Returns True if the Tab exists.

GameTab

function `GameTab`(`Tab`: `Integer`): Boolean ;

Description: Switches to a specific game tab.

SetFightMode

function `SetFightMode`(`oFightMode`: `Integer`): Boolean ;

Description: Sets fight mode. Returns false if failed to set desired mode.

SetCombatType

function `SetCombatType`(`Kind` : `string`): Boolean ;

Description: Sets Combat mode inputted into Kind treat like UpText.

GetCombatLevel

function `GetCombatLevel`;

Description: Returns the players combat level.

Retaliate

procedure `Retaliate`;

Description: Set's Auto Retaliate ON of OFF.

SkillCoords

function SkillCoords(Row Column: Integer): TPoint ;

Description: Returns Coords of Skill's Row and Column (Used for GetSkill functions)

SkillToCoords

function SkillToCoords(Skill: Variant): TPoint ;

Description: Turns skill string into TPoint.

GetSkillInfo

function GetSkillInfo(Skill: Variant; Amount : Boolean): Integer ;

Description: Gets the amount / level of a skill.

GetSkillLevel

function GetSkillLevel(skill: Variant): Integer ;

Description: Gets the max level for a particular skill.

IsResting

function IsResting;

Description: Returns True if you are currently resting.

GetMMLevels

function GetMMLevels(LevelType : String; var ColorSign : String): integer ;

Description: Returns the level shown next to the minimap.

GetXP

function GetXP(Skill: Variant): Integer ;

Description: Returns current xp for a skill. Returns -1 if failed.

XpTillNextLevel

function XpTillNextLevel(Skill: Variant): Integer ;

Description: Returns current xp until you level up in a skill.

HpPercent

function HpPercent;

Description: Returns Hp left as a percentage.

GetAllLevels

procedure [GetAllLevels](#);

Description: Sets all 25 skilllevels to Players.Level[1..25]

EquipmentCoords

function [EquipmentCoords](#)([EquipSlot](#): [Integer](#)): TPoint ;

Description: Returns X and Y of the coordinates of the specified equipment item

GetEquippedItemBounds

function [GetEquippedItemBounds](#)([Which](#): [String](#)): TBox ;

Description: Returns TBox of equipped item in which.

MouseEquippedItem

procedure [MouseEquippedItem](#);

Description: Mouses Equipped item like MouseItem.

WearingItem

function [WearingItem](#)([i](#): [Integer](#)): Boolean ;

Description: Results True if an item is equipped at equipment slot defined by I.

TakeOff

procedure [TakeOff](#);

Description: Unequips Item specified in i.

TakeOffAllExcept

procedure [TakeOffAllExcept](#);

Description: Takes off equipped items defined by StripSlots.

TakeOffAllItems

procedure [TakeOffAllItems](#);

Description: Takes off all equipped items.

CountEquippedItems

function [CountEquippedItems](#);

Description: Counts the amount of equipped items. Results "-1" if failed.

CurrentWorld

function `CurrentWorld`;

Description: Returns the current world you are on.

ObjectivePercent

function `ObjectivePercent`;

Description: Returns (Roughly) Objective Completion in Percent.

ObjectiveCoords

function `ObjectiveCoords`(`ObjSlot`: `Integer`): `TPoint` ;

Description: Returns Coords of Random Objective Slot (`ObjSlot` : 1..6)

HasObjective

function `HasObjective`;

Description: Returns True if an Objective is set.

ClearObjective

function `ClearObjective`;

Description: Returns True if an Objective is Cleared or no Objective Exists.

SetObjective

function `SetObjective`(`Skill`: `String`; `Level`: `Integer`): `Boolean` ;

Description: Sets Objective according to Skill and Level.

SetRandomObjective

function `SetRandomObjective`(`ObjectiveSlot`: `Integer`; `KeepOld`: `Boolean`): `Boolean` ;

Description: Sets a Random Objective Slot (`ObjSlot` : 1..6). If an objective is

SetRun

function `SetRun`(`Run`: `Boolean`): `Boolean` ;

Description: Sets Run on or off.

RunEnergy

function `RunEnergy`(`Min`: `Integer`): `Boolean` ;

Description: Toggles run if energy is greater or equal to `MinEnergy`. Results

SetRest

function [SetRest](#);

Description: Turns Rest on. Result True if successful.

RestUntil

function [RestUntil](#)([Energy](#): [Integer](#)): Boolean ;

Description: Rests until your Run energy is above Energy. Results True if

RestUntilHP

function [RestUntilHP](#)([HPPerc](#): [Integer](#)): Boolean ;

Description: Rests until your Run energy is above the given HP percentage.

Results True if

FixGraphics

procedure [FixGraphics](#);

Description: Fixes the Graphic Options if LoggedIn

SetGraphics

procedure [SetGraphics](#);

Description: Sets graphic options through the Graphics Screen in GameTab 11.

SetAudio

procedure [SetAudio](#);

Description: Sets audio options through the Audio screen in GameTab 11

SetBar

procedure [SetBar](#);

Description: Sets each bar to the specific point. 1-4 for Brightness 1-5 for

DoEmote

procedure [DoEmote](#);

Description: Clicks on an emote specified by EmoteNumber (1 to 37)

GetLobbyTab

function [GetLobbyTab](#);

Description: Gets current lobby tab.

LobbyTab

function LobbyTab(Tab: Integer): Boolean ;

Description: Goes to the lobby tab specified by the integer 'Tab'.

Ranging.scar

ArrowAmount

function [ArrowAmount](#);

Description: Returns the number of equipped arrows.

GetArrowColor

function [GetArrowColor](#);

Description: Returns the number of equipped arrows.

FindInvArrows

function [FindInvArrows](#)(**var** [arrowx](#) [arrowy](#): [Integer](#)): Boolean ;

Description: Returns true if arrows are found in the inventory and the coordinates.

InvArrows

function [InvArrows](#);

Description: Returns the number of arrows in the inventory.

EquipArrows

function [EquipArrows](#);

Description: Equips arrows if they are found in the inventory.

SetupRanging

procedure [SetupRanging](#);

Description: Declares Global Bitmaps.

Slayer.scar

SetupSlayer

procedure [SetupSlayer](#);

Description: Declares Global Bitmaps.

Prayer.scar

GetPrayerCoords

function `GetPrayerCoords`(`Name`: `string`; `var Box`: `TBox`; `var Color`: `Integer`):
Boolean ;
Description: Returns a box and color from a certain prayerbutton.

PrayerExists

function `PrayerExists`(`Name`: `string`): Boolean ;
Description: Returns true if the players' level is high enough to use the prayer.

PrayerActivated

function `PrayerActivated`(`Name`: `string`): Boolean ;
Description: Returns true if the prayer is activated.

ClickPrayer

function `ClickPrayer`(`Name`: `string`; `Activate`: `Boolean`): Boolean ;
Description: Turns a certain prayer on or off. Returns true if done successfully.

AllPrayersOff

procedure `AllPrayersOff`;
Description: Will turn off any prayers that may be on

InQuickPrayerMenu

function `InQuickPrayerMenu`;
Description: Returns true if in menu to choose Quick Prayers

OpenQuickPrayerMenu

function `OpenQuickPrayerMenu`;
Description: Opens The Quick Prayer Menu

ExitQuickPrayerMenu

function [ExitQuickPrayerMenu](#);

Description: Closes The Quick Prayer Menu

QuickPrayerActive

function [QuickPrayerActive](#);

Description: Checks if Quick Prayer on / off

SwitchQuickPrayer

procedure [SwitchQuickPrayer](#);

Description: Turns Quick Prayer on / off

ChooseQuickPrayer

procedure [ChooseQuickPrayer](#);

Description: Turns a specific Prayer on/off

SetQuickPrayers

procedure [SetQuickPrayers](#);

Description: Enables quick prayers by number.

TurnOffAllQuickPrayers

procedure [TurnOffAllQuickPrayers](#);

Description: Turns off all quick prayer.

PrayerPoints

function [PrayerPoints](#);

Description:

SetupPrayer

procedure [SetupPrayer](#);

Description: Declares Global Bitmaps.

RuneCrafting.scar

EnterRCAltar

function EnterRCAltar;

Description: Enters a RuneCrafting altar.

ClickPouch

procedure ClickPouch;

Description: Click Pouch

FindTalismanInInventory

function FindTalismanInInventory(TalismanName: string; tol: Integer; Var Dx Dy: Integer): Boolean ;

Description: Returns true if the specified talisman is in inventory

ClickTalisman

function ClickTalisman(Talisman: string): Boolean ;

Description: Clicks the talisman Talisman in your inventory.

SetUpRuneCrafting

procedure SetUpRuneCrafting;

Description: SetUp RuneCrafting

Construction.scar

SetupConstruction

procedure [SetupConstruction](#);

Description: Declares Global Bitmaps.

Farming.scar

SetupFarming

procedure [SetupFarming](#);

Description: Declares Global Bitmaps.

Smithing.scar

FindFurnace

function FindFurnace(**var** X Y: Integer): Boolean ;
Description: Finds a furnace and stores the coordinates into X and Y.

SmeltingScreen

function SmeltingScreen;
Description: Returns true if the bar selection in chatbox is open.

OpenFurnace

function OpenFurnace;
Description: Returns true if managed to open the bar selection in chatbox.

BarToPoint

function BarToPoint(**Bar**: String): TPoint ;
Description: Returns the coordinates of the bar.

Smelt

procedure Smelt;
Description: Smelts the Bar bar for the amount Amount -

SmithingScreen

function SmithingScreen;
Description: Returns true if the smithing interface is open.

FixSmith

procedure FixSmith;
Description: Scrolls the smithing screen.

SmithItem

function [SmithItem](#)([Item](#): [String](#); [Amount](#): [string](#)): Boolean ;
Description: Returns true if could smith the item Item.

GetSmeltingXP

function [GetSmeltingXP](#)([bar](#): [string](#)): extended ;
Description: Result the xp gained from Smelting bar

GetSmithingXP

function [GetSmithingXP](#)([bar](#): [string](#)): extended ;
Description: Result the xp gained from Smithing bar

SetupSmithing

procedure [SetupSmithing](#);
Description: Declares Global Bitmaps.

Fishing.scar

FindFishingEquipmentColor

function FindFishingEquipmentColor(Equipment: String): Integer ;
Description: Finds Net Lobster Cage Fishing Rod Fly Fishing Rod or Harpoon color.

FindWhirlPoolXY

function FindWhirlPoolXY(X Y: Integer): Boolean ;
Description: Finds Whirlpools at the fishing spot located at X Y.

FindWhirlPoolMS

function FindWhirlPoolMS;
Description: Returns true if whirlpool found in the spots around your player in

HandleWhirlPool

procedure HandleWhirlPool;
Description: Finds and Handles Whirlpools by Standing still and waiting

FindFishSpotsTPA

function FindFishSpotsTPA(var VTPA: TPointArray): Boolean ;
Description: Finds all possible Fish Spots and stores them in the given TPA

FindFishSpot

function FindFishSpot(var xx yy: integer; which: string): Boolean ;
Description: Finds a fishing spot according to your needs.

SetupSRLFishing

procedure SetupSRLFishing;
Description: Loads Fishing Bitmaps into memory.

Crafting.scar

OpenOven

function [OpenOven](#);
Description: Opens oven.

SetupCrafting

procedure [SetupCrafting](#);
Description: Declares Global Bitmaps.

Magic.scar

SetupSpells

procedure [SetupSpells](#);

Description: Sets the SpellModes var to match the runescape spells.

DebugSpellVars

procedure [DebugSpellVars](#);

Description: Prints out the SpellModes var.

GetSpellInfo

function [GetSpellInfo](#)([Spell](#): [string](#)): TSpell ;

Description: Gets a spells info from the SpellModes var and results

SpellIsSort

function [SpellIsSort](#)([Spell](#): [string](#)): Integer ;

Description: Results what kind of sort the spell is..

SpellsAvailable

function [SpellsAvailable](#);

Description: Results which sort of spells are available.

ClickSpellMode

procedure [ClickSpellMode](#);

Description: Clicks the given parameter.

SetSpellMode

procedure [SetSpellMode](#);

Description: Sets the spellmodes on and off for each mode.

BookSortedBy

function `BookSortedBy`;

Description: Results how the book is sorted.

SortBook

procedure `SortBook`;

Description: Sorts the book from given parameter.

DefCastIs

function `DefCastIs`;

Description: Results if defensive casting is enabled.

ToggleDefCast

procedure `ToggleDefCast`;

Description: Turns defensive casting on/off.

ScrollSpellBook

procedure `ScrollSpellBook`;

Description: Scrolls the spellbook up or down.

SpellAvailable

function `SpellAvailable`(`var` `Coords`: `TBox`; `Spell`: `string`): `Boolean` ;

Description: Results true if the given spell is available and can be

Cast

function `Cast`(`Spell`: `string`; `AutoCast`: `Boolean`): `Boolean` ;

Description: Results true if the spell given is clicked.

SetupMagic

procedure `SetupMagic`;

Description: Declares Global Bitmaps.

Firemaking.scar

LoadFireMakingBitmaps

procedure [LoadFireMakingBitmaps](#);

Description: Loads bitmaps into memory or frees it from memory.

CantLightCheck

function [CantLightCheck](#);

Description: Results True if able to light logs

LightWait

function [LightWait](#);

Description: Waits till Logs are Lighted

WeHaveLogs

function [WeHaveLogs](#)(**var** [WhereX](#) [WhereY](#): [Integer](#)): Boolean ;

Description: Checks if there is Logs in your Inventory Returns X Y Cords of Log

ThereIsFire

function [ThereIsFire](#)(**var** [WhereX](#) [WhereY](#): [Integer](#)): Boolean ;

Description: Checks if there is lit fire in MainScreen. Returns X Y Cords.

SetupFiremaking

procedure [SetupFiremaking](#);

Description: Declares Global Bitmaps.

WoodCutting.scar

FindAxe

function FindAxe;

Description: Finds if there is an axe present. Axe can be wielded or unwielded.

FindBirdsNest

function FindBirdsNest;

Description: If a birds nest is found it will pick it up

FindEnt

function FindEnt(fx fy: Integer; AvoidWhenFound: Boolean): Boolean ;

Description: Uses UpText to check if an ent is present at fx fy.

SetupWoodcutting

procedure SetupWoodcutting;

Description: Declares Global Bitmaps.

Thieving.scar

SetupThieving

procedure [SetupThieving](#);

Description: Declares Global Bitmaps.

Mining.scar

FindPickHeadColor

function FindPickHeadColor;

Description: Finds PickHead colors. Returns hc1 2 3 4 & 5 for PickHead

FindPick

function FindPick;

Description: Finds and Attaches Pickhead.

CheckOre

function CheckOre(OreX OreY RefColor Tol: Integer): Integer ;

Description: Checks the certain ore. Returns how many colors match.

OreColorCount

function OreColorCount(OreX OreY RefColor Tol MinCount: LongInt): Boolean

;

Description: Checks the certain ore uses CheckOre.

FindAllRocksRecord

function FindAllRocksRecord(var RockObjs: TPointArray; RockIndex: Integer; QuickSorting RGBXYZCheck: Boolean): Boolean ;

Description: Finds the rocks using SRL's Rock Records and returns the found objects in a TPA

FindObjRockEx

function FindObjRockEx(var cx cy: Integer; RockIndex: Integer; QuickSorting RGBXYZCheck: Boolean): Boolean ;

Description: Finds the rocks using SRL's Rock Records

FindObjRock

function FindObjRock(**var** cx cy: Integer; RockIndex: Integer): Boolean ;

Description: Finds the rocks using SRL's Rock Records

LoadRockRecords

procedure LoadRockRecords;

Description: Loads the records for FindObjRock

LoadMiningBitMaps

procedure LoadMiningBitMaps;

Description: Declares Mining Bitmap.

SetupMining

procedure SetupMining;

Description: Loads Mining bitmap into memory.

Hunting.scar

SetupHunting

procedure [SetupHunting](#);

Description: Declares Global Bitmaps.

Agility.scar

GetCourseXP

function [GetCourseXP](#)(**Course**: [String](#)): Extended ;
Description: Returns the experience for a certain agility course.

SetupAgility

procedure [SetupAgility](#);
Description: Declares Global Bitmaps

Fletching.scar

SetupFletching

procedure [SetupFletching](#);

Description: Declares Global Bitmaps.

Herblore.scar

SetupHerblore

procedure [SetupHerblore](#);

Description: Declares Global Bitmaps.

Fighting.scar

Damage

function `Damage`;

Description: Returns the amount of damage.

InFight

function `InFight`;

Description: Returns as True if player is in a fight.

IsFightAt

function `IsFightAt`(`x` `y`: `Integer`): Boolean ;

Description: results true if it finds a fight near x y.

FightNPC

function `FightNPC`(`Colors`: `TIntegerArray`; `Tol` `TimeOut`: `Integer`): Boolean ;

Description: Fights NPC. Input an array of colors. (May be the same colors)

FindAllHPBars

function `FindAllHPBars`(`Sx` `Sy` `x1` `y1` `x2` `y2`: `Integer`): `TBoxArray` ;

Description: Finds all the HP bars (the TPAs of them) in the box formed by

GetHits

function `GetHits`(`Sx` `Sy` `x1` `y1` `x2` `y2`: `Integer`): `TIntegerArray` ;

Description: Finds all HP percents of the HP bars in `x1 y1 x2 y2`.

FindMainScreenHits

function `FindMainScreenHits`(`Bar`: `Integer`): `Integer` ;

Description: Returns the rounded HP percent of the hp bar `Bar` counting out

216

from

Cooking.scar

FindRange

function FindRange(**var** X Y: Integer): Boolean ;

Description: Will find a range inside the variables given it wont

SetupCooking

procedure SetupCooking;

Description: Declares Global Bitmaps.