Version 0.10.0 2020-3-17

1. 每个游戏建一个子目录（比如: R6），存放脚本和数据。主脚本必须命名为main.lua。识别用到的数据在<游戏子目录>/data目录下面。
2. 启动辅助命令：love.exe <游戏子目录名>

如果有错误，检查Log目录下面 <游戏子目录名>.log文件。

1. R6/TestCase.lua 是基本类（SSize, SRect, GameScene等）测试和例子代码。
2. NotePad/TestInputText.lua 是按键测试和例子代码。
3. Driver/KeyboardTestUtility.exe是一个不错的工具，可以显示按键的VK和ScanCode。

输入驱动模式可以绕开一些反作弊软件的跨进程发消息保护。使用输入驱动模式才需要安装Driver/install-interception.exe, 安装或卸载后一定要重启机器。重启机器之前不要进行卸载或者安装反向操作（否则键盘会失效）。

1. 所有时间单位都是毫秒。

Window类

{

Window()

IsValid()

CloseWindow()

FromHandle(int)

GetHandle()

SRect GetWindowRect

SetWindowRect(SRect)

SRect GetClientRect()

IsVisible()

IsMinimized()

IsMaximized()

ShowWindow(show) -- same as Win32 ShowWindow

GetWindowText()

SetWindowText(string)

GetWindowClass()

GetWindowLong(index)

SetWindowLong(index, value)

FlashWindow()

SetForegroundWindow()

FindChildWindow(title, class)

WindowWrapArray s\_FindWindow(title, class) --静态函数

Window s\_GetForegroundWindow() --静态函数

}

WindowWrapArray类

{

WindowWrapArray()

Size()

Window At(int)

}

Wolves表

{

Version() --返回版本号string

bool Initialize(gameKey) -- 例如: R6

Finalize()

Update(dt)

GetRobot() -- Robot是单例，多线程之间共享

Sleep(int msTime) -- [线程安全]

int GetCurTime() -- [线程安全]

-- 1 Debug; 2 Info; 3 Warning; 4 Error; 5 Fatal

SetLogLevel(level) -- [线程安全]

LogDebug(string) -- [线程安全]

LogInfo(string) -- [线程安全]

LogWarn(string) -- [线程安全]

LogError(string) -- [线程安全]

LogFatal(string) -- [线程安全]

-- SharedManager 函数都是线程安全的

SharedManager表

{

sharedTable NewSharedTable()

bool ShareSharedTable(sharedTable, name)

sharedTable AcquireSharedTable(name)

DumpSharedTable(sharedTable, codeFormat)

bool CreateThread(name)

bool IsThreadValid(name)

bool PauseThread(name)

bool ResumeThread(name)

bool AbortThread(name)

bool SendMessageToThread(name, sharedTable)

sharedTable FetchCurThreadMessage()

bool SendMessageToMainThread(sharedTable)

}

}

-- Robot类(注意： 只有标记为[线程安全]的函数才能在所有线程里面使用)

{

-- 这些函数需要在同一个线程里面使用

--[

bool Initialize(g\_EnableMonitoringWindow, g\_InputUseDriver)

--如果Monitor不起作用，请用管理员权限启动辅助

bool InitializeEx(InputUseDriver, PositionMonitor, SizeMonitor, asChildWnd)

Finalize()

bool HookWindow(wnd, simple, bFlipY) -- 如果simple==false, Hook 3D rendering

StopHookWindow()

GameScene\* TakeSnapshot() -- 对目标游戏窗口截图

GameScene\* GetCurGameScene()

GameScene\* TakeSnapshotWindow(wnd)

--匹配的图片来源于上次TakeSnapshot()/TakeSnapshotWindow()

SRectVector IsSubSceneMatched(subSceneName)

SRectVector IsAnySubSceneMatched(StringVector subSceneNames)

SRectVector IsSubSceneMatchedInRect(subSceneName, rect)

bool IsPixelMatched(SSize posStart, int offSetX, int offSetY, int pixelCount, SRGB rgb, SRGB rgbT)

bool IsGrayRect(SRect)

--]

-- 后台按键操作，x, y是窗口客户区坐标。Dx游戏要前台方式。

-- left: lmr = 0, middle: lmr = 1, right: lmr = 2

InputMouseClick(wnd, lmr, x, y) -- [线程安全]

InputMouseButtonDown(wnd, lmr, x, y) -- [线程安全]

InputMouseButtonRelease(wnd, lmr, x, y) -- [线程安全]

InputMouseHoldMove(wnd, lmr, x, y) -- [线程安全]

InputMouseMove(wnd, x, y) -- [线程安全]

InputKeyEvent(wnd, vk, isDown) -- [线程安全]

InputText(wnd, str, mode) -- [线程安全] str must in UTF-8; mode, 0: keep original; 1: to MBCS; 2: to Unicode

// 剪贴板相关

bool CopyTextToClipboard(str) -- [线程安全]

str = GetTextFromClipboard() -- [线程安全]

InputCut(wnd) -- [线程安全]

InputCopy(wnd) -- [线程安全]

InputPaste(wnd) -- [线程安全]

-- 前台按键操作，x, y是窗口客户区坐标。Wnd是顶层窗口，不能是子窗口。

InputForegroundMouseButtonEvent(wnd, useDriver, lmr, isDown, ensureForeground) -- [线程安全]

InputForegroundMouseMove(wnd, useDriver, x, y, ensureForeground) -- [线程安全]

InputForegroundMouseMoveRelative(wnd, useDriver, xDelta, yDelta, ensureForeground) -- [线程安全]

InputForegroundMouseScroll(wnd, useDriver, isUp, ensureForeground) -- [线程安全]

-- [线程安全] vk is in virtual-key code

InputForegroundKeyEvent(wnd, useDriver, vk, isExtend, isDown, ensureForeground)

InputForegroundProbeKeyBoardIndex(int) --刺探驱动模式下键盘索引，可以尝试设置1-9，然后发送按键，检查按键结果，如果监测到正确输入内容，说明刺探到索引；重置设置为0。如果刺探之前已有物理键盘按键，刺探不起作用。

}

SharedTable是可以跨线程共享的类lua表(支持各种简单数据和嵌套表，不支持对象/函数)，具体用法见下面例子。

TestSharedTable.lua

--其它类/表使用说明/例子

require "Hook"

print("\n-------------------------------------------------------------------------------")

print("Test SSize...")

local pt = SSize(1, 2)

print(pt)

pt.cx = 3

pt.cy = 4

print(pt)

print("\n-------------------------------------------------------------------------------")

print("Test SSizeFloat...")

local pt2 = SSizeFloat(1.5, 2.5)

print(pt2)

pt2.cx = 3.5

pt2.cy = 4.5

print(pt2)

print("\n-------------------------------------------------------------------------------")

print("Test SRect...")

local rect = SRect(1, 2, 3, 4)

print("rect ", rect)

print("rect:IsValid()", rect:IsValid())

print("rect:Contains(2, 3)", rect:Contains(2, 3))

print("rect:Width() " .. rect:Width())

print("rect:Height() " .. rect:Height())

print("rect:Center()", rect:Center())

local rect2 = SRect(5, 6, 7, 8)

print("rect2 ", rect2)

print("rect:Union(rect2)", rect:Union(rect2))

print("rect2:OffsetRect(2, 2)", rect2:OffsetRect(2, 2))

local rectOut = SRect(0, 0, 0, 0)

print("rect:Intersect(rect2, rect3)", rect:Intersect(rect2, rectOut))

print("rectOut ", rectOut)

print("rect:Less(rectOut)", rect:Less(rectOut))

print("\n-------------------------------------------------------------------------------")

print("Test SRectVector...")

local rcVector = SRectVector()

print(rcVector:Size())

rcVector:Push(SRect(1, 2, 3, 4))

print(rcVector:Size())

print(rcVector:At(1))

print("\n-------------------------------------------------------------------------------")

print("Test String...")

local str = String("This is a test string.汉字。")

print(str)

print(tostring(str))

print("\n-------------------------------------------------------------------------------")

print("Test StringVector...")

local strVect = StringVector()

print("strVect:Size(): ", strVect:Size())

strVect:Push("str1")

strVect:Push("str2")

strVect:Push("str3")

print("strVect:Size(): ", strVect:Size())

print(strVect:At(1))

print("\n-------------------------------------------------------------------------------")

print("Test RGB...")

local rgb0 = SRGB(255, 0, 128)

print(rgb0, rgb0.r, rgb0.g, rgb0.b)

print("\n-------------------------------------------------------------------------------")

print("Test GameScene...")

local gs = GameScene("E:/1.png", false);

print("gs:SaveToFile")

gs:SaveToFile("E:/1\_bak.png")

print("gs:IsValid: ", gs:IsValid())

print("gs:GetSize: ", gs:GetSize())

local gs2 = gs:Clone()

print("gs2 = gs:Clone()")

print("gs2:GetSize: ", gs2:GetSize())

local rcSub = SRect(100, 100, 300, 300)

print("gs3 = gs2:SubScene(rcSub)")

local gs3 = gs2:SubScene(rcSub)

print("gs3:GetSize: ", gs3:GetSize())

print("gs4 = gs:SubChannel(0)")

local gs4 = gs:SubChannel(0)

gs4:SaveToFile("E:/1\_sub.png")

local rgb = gs:GetPixel(SSize(350, 250))

print("GetPixel: ", rgb, rgb.r, rgb.g, rgb.b)

gs2:Rectangle(SRect(100, 100, 300, 300), 255, 0, 255, 1, 8)

gs2:Line(SSize(100, 100), SSize(300, 300), 255, 0, 255, 2, 4)

gs2:Circle(SSize(100, 100), 100, 0, 255, 0, 1, 4)

gs2:PutText("This a string. Chinese characters are not supported.", SSize(100, 100), 255, 0, 0, 1, 1)

gs2:PutText("This an another string.", SSize(100, 120), 0, 255, 0, 1, 2)

gs2:ShowDebugWindow("gs2 Drawings")

GameScene.s\_WaitKey(2000)

gs2:DestroyDebugWindow("gs2 Drawings")

-- CV\_BGR2GRAY =6,

-- CV\_RGB2GRAY =7,

gs2:TransformColor(6) --CV\_BGR2GRAY

--gs2:ShowDebugWindow("gs2 Drawings")

--GameScene.s\_WaitKey(2000)

-- > 1增强, < 1减小

gs2:ContrastAdjust(1.5)

--gs2:ShowDebugWindow("gs2 Drawings")

--GameScene.s\_WaitKey(2000)

-- xScale

-- yScale

-- INTER\_NEAREST = 0, INTER\_LINEAR, INTER\_CUBIC, INTER\_AREA, INTER\_LANCZOS4

gs2:Resize(2.0, 2.0, 1)

print("gs2:GetSize: ", gs2:GetSize())

-- ksize must be positive and odd

gs2:Blur(3, 3)

--gs2:ShowDebugWindow("gs2 Drawings")

--GameScene.s\_WaitKey(2000)

-- CV\_THRESH\_BINARY =0, /\* value = value > threshold ? max\_value : 0 \*/

-- CV\_THRESH\_BINARY\_INV =1, /\* value = value > threshold ? 0 : max\_value \*/

-- CV\_THRESH\_TRUNC =2, /\* value = value > threshold ? threshold : value \*/

-- CV\_THRESH\_TOZERO =3, /\* value = value > threshold ? value : 0 \*/

-- CV\_THRESH\_TOZERO\_INV =4, /\* value = value > threshold ? 0 : value \*/

-- CV\_THRESH\_MASK =7,

-- CV\_THRESH\_OTSU =8 /\* use Otsu algorithm to choose the optimal threshold value; combine the flag with one of the above CV\_THRESH\_\* valus \*/

gs2:Threshold(220, 255, 3) --CV\_THRESH\_TOZERO

--gs2:ShowDebugWindow("gs2 Drawings")

--GameScene.s\_WaitKey(2000)

-- MORPH\_RECT=0, MORPH\_CROSS=1, MORPH\_ELLIPSE=2

-- is\_erode = true

-- size must be odd

gs2:ErodeDilate(1, true, 3)

--gs2:ShowDebugWindow("gs2 Drawings")

--GameScene.s\_WaitKey(2000)

gs2:SaveToFile("E:/1\_draw.png")