

Работа с таймерами

Платформа Lamda-Mu 2 позволяет использовать системный таймер. Для запуска таймера используется функция **sys.StartTimer()** (подробнее см. статью «[Библиотека функций](#)»)

Пример:

запуск таймера

```
sys.StartTimer('refresh')
```

обработчик события остановки таймера

```
function H_refresh(ev)
-- some code
end
AddEventHandler('timer.refresh.finished.', 'H_refresh')
```

Остановка таймера (событие остановки не наступает)

```
sys.StopTimer('refresh')
```

Создадим проект, который будет работать с системным таймером. Исполняемые модули находятся в репозитории [Im2.engine.ce](https://github.com/lamda-mu/Im2.engine.ce). Файлы для проекта находятся в репозитории [Im2.examples](https://github.com/lamda-mu/Im2.examples). Необходимый минимум доступен по ссылкам:

- [win_timer_project_start.zip](#)
- [rpi_timer_project_start.zip](#)

Создание структуры директорий

Создайте папку `empty_project`. Это корень проекта. Скопируйте в корень исполняемый модуль (исполняемый файл `.exe` и динамические библиотеки `.dll` необходимые для старта платформы в случае Windows или скомпилированный бинарный файл в случаях систем на основе ядра Linux).

Создайте в корне папку `images`. Это каталог для хранения рендеров. Далее мы укажем к нему путь в главном конфигурационном файле. Скопируйте в этот каталог файлы `.png`.

Создайте в корне папку `configs`. Это каталог для хранения конфигурационных файлов (кроме главного конфигурационного файла, он должен быть в корне проекта).

Создайте в корне папку `fonts`. Это каталог для используемых шрифтов. Далее мы подключим шрифты в главном конфигурационном файле. Скопируйте в этот каталог файл `LiberationSerif-Regular.ttf`.

Создание файлов

Главный конфигурационный файл

Основная статья «[Главный конфигурационный файл](#)». Создайте в корне проекта файл config.xml. Платформа по умолчанию к нему обратиться. Никаких дополнительных действий не требуется. Файл должен содержать следующий код:

```
<!-- объявление xml -->
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!-- корневой элемент -->
<root>
  <!-- элемент общих настроек, задаются следующие: язык, путь к рендерам,
  путь к конфигурационным файлам, флаг логирования -->
  <System
    Language="ru"
    ImagesPath="images/"
    ConfigsPath="configs/"
    Log="true"/>
  <!-- элемент логической машины, задаются следующие свойства: уникальный
  идентификатор, файл описание панелей, файл конфигурации окон, первый
  файл с lua-кодом для запуска -->
  <VM
    ID="main"
    Visual="panels.xml"
    StartupLayout="viewports.xml"
    StartupLogic="start.lua">
  </VM>
  <!-- элемент шрифта, задаются следующие свойства: уникальный идентификатор,
  полный путь к файлу -->
  <Font ID="Default" FileName="fonts/LiberationSerif-Regular.ttf" />
</root>
```

Файл конфигурации окон

Основная статья «[Конфигурация окон](#)». В папку configs создайте viewports.xml. Название файла должно совпадать с названием, указанным в главном конфигурационном файле. Определите окно на первом мониторе (если у вас несколько мониторов) с разрешением 800x480. Файл должен содержать следующий код:

```
<!-- объявление xml -->
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!-- корневой элемент, задается следующее свойство: разрешение -->
<root MonitorWidth="800" MonitorHeight="480">
  <!-- элемент вьюпорт, задаются следующие свойства: положение на мониторе,
  уникальный идентификатор -->
  <ViewPort Monitor="0,0" ID="main_viewport" />
</root>
```

Файл описания панелей

Основная статья «[Описание панелей](#)». Создайте в папке configs файл panels.xml. Название файла должно совпадать с названием, указанным в главном конфигурационном файле. Необходимо подключить библиотеку элементов, файл со строковыми данными, описать панель с фоном, двумя кнопками, логотипом и двумя элементами содержащими спрайштиты лампы и цифр таймера. Файл должен содержать следующий код:

```
<!-- объявление xml -->
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!-- корневой элемент -->
<root>
  <!-- элемент подключающий строковые данные -->
  <Text FileName="text.xml"/>
  <!-- элемент подключающий библиотеку элементов -->
  <Library FileName="templates.xml"/>
  <!-- элемент родитель панелей -->
  <GUI>
    <!-- элемент описывающий панель, задается следующее свойство: размеры вьюпорта,
    на который будет выводиться панель -->
    <Panel ID="main" VPWidth="800" VPHeight="480">
      <!-- элемент на панели, задается следующее свойство: уникальный идентификатор-->
      <Entry ID="bg" TypeID="bg"/>
      <!-- элемент на панели, задается следующее свойство: уникальный идентификатор,
      положение на панели-->
      <Entry ID="btExit" TypeID="button" Top="300" Left="300"/>
      <!-- элемент на панели, задается следующее свойство: уникальный идентификатор,
      положение на панели-->
      <Entry ID="btStartStop" TypeID="button" Top="300" Left="520">
        <!-- меняется свойство, описанное в библиотеке элементов (она уже загружена) -->
        <PropertySet Property="text.Text" ValueTID="txtStart"/>
      </Entry>
      <!-- элемент на панели, задаются следующие свойства: уникальный идентификатор,
      положение на панели -->
      <Entry ID="Logo" TypeID="logo" Top="60" Left="60"/>
      <!-- элемент на панели, задаются следующие свойства: уникальный идентификатор,
      положение на панели -->
      <Entry ID="Clock" TypeID="clock" Top="60" Left="220" />
      <!-- элемент на панели, задаются следующие свойства: уникальный идентификатор,
      положение на панели -->
      <Entry ID="Lamp" TypeID="lamp" Top="60" Left="555" />
    </Panel>
  </GUI>
</root>
```

Файл с библиотекой элементов

Основная статья «[Библиотека элементов](#)». Создайте в папке configs файл templates.xml. Название файла должно совпадать с названием, указанным в файле описания панелей. Необходимо описать элементы на панели. Файл должен содержать следующий код:

```
<!-- объявление xml -->
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
```

```

<!-- корневой элемент -->
<root>
  <!-- объявление библиотеки -->
  <Library>
    <!-- элемент библиотеки, задаются следующие свойства: уникальный идентификатор,
    размеры элемента, -->
    <Element ID="bg" Width="800" Height="480">
      <Comment>Background</Comment>
      <!-- подэлемент прямоугольник, задаются следующие свойства: уникальный
      идентификатор,
      размеры и положение, цвет, цвет заливки, толщина контура, радиус закругления углов
      -->
      <Rectangle ID="bg" Rect="50,50,700,380" Color="white" BColor="black"
      Thickness="1" CornerRadius="10"/>
    </Element>
    <!-- элемент библиотеки, задаются следующие свойства: уникльный идентификатор,
    размеры элемента -->
    <Element ID="button" Width="200" Height="80">
      <!-- подэлемент прямоугольник, задаются следующие свойства: уникальный
      идентификатор,
      размеры и положение, цвет, цвет заливки, толщина контура, радиус закругления углов
      -->
      <Rectangle ID="bg" Rect="full" Color="white" BColor="black"
      Thickness="2" CornerRadius="5"/>
      <!-- подэлемент текстовое поле, задаются следующие свойства: уникальный
      идентификатор,
      размер и положение, идентификатор строковых данных, шрифт, кегль, выравнивание -
      -->
      <TextBox ID="text" Rect="full" TextTID="txtExit" Font="Default"
      Size="30" Align="Center" VAlign="Center"/>
    </Element>
    <!-- элемент библиотеки, задаются следующие свойства: уникальный идентификатор,
    размеры элемента, масштаб -->
    <Element ID="logo" Width="369" Height="595" Scale="0.25">
      <!-- подэлемент изображение, задаются следующие свойства: размеры и положение,
      путь к рендеру -->
      <Image Rect="full" Source="logo.png"/>
    </Element>
    <!-- элемент библиотеки, задаются следующие свойства: уникальный идентификатор,
    размеры элемента, -->
    <Element ID="clock" Width="300" Height="130">
      <Comment>Цифровые часы</Comment>
      <!-- подэлемент изображение, задаются следующие свойства: уникальный
      идентификатор,
      размеры и положение, путь к рендеру -->
      <Image ID="face" Rect="full" Source="clock2Face.png"/>
      <!-- подэлемент изображение, задаются следующие свойства: уникальный
      идентификатор,
      размеры и положение, путь к рендеру, индекс спрайта, видимость -->
      <Image ID="digit1" Rect="20,33,38,64" Source="digits.png"
      SpriteIndex="0" Visible="true"/>

```

```

    <!-- подэлемент изображение, задаются следующие свойства: уникальный
идентификатор,
    размеры и положение, путь к рендеру, индекс спрайта, видимость -->
    <Image ID="digit2" Rect="80,33,38,64" Source="digits.png"
SpriteIndex="0" Visible="true"/>
    <!-- подэлемент изображение, задаются следующие свойства: уникальный
идентификатор,
    размеры и положение, путь к рендеру, индекс спрайта, видимость -->
    <Image ID="digit3" Rect="170,33,38,64" Source="digits.png"
SpriteIndex="0" Visible="true"/>
    <!-- подэлемент изображение, задаются следующие свойства: уникальный
идентификатор,
    размеры и положение, путь к рендеру, индекс спрайта, видимость -->
    <Image ID="digit4" Rect="230,33,38,64" Source="digits.png"
SpriteIndex="0" Visible="true"/>
    <!-- задается свойство lua-кодом -->
    <Property PropertyID="value" Default="0000">
        <LUA><![CDATA[
-- этот код управляет цифрами на таймере
    local digits = value
    for i = 1, 4 do
        local ch = digits:byte(i)
        if ch >= 48 and ch <= 57 then
            Set("digit"..i.."SpriteIndex", (ch - 48))
        end
    end
    ]]></LUA>
    </Property>
</Element>
    <!-- элемент библиотеки, задаются следующие свойства: уникальный идентификатор,
размеры элемента, -->
    <Element ID="lamp" Width="130" Height="130">
        <!-- подэлемент изображение, задаются следующие свойства: уникальный
идентификатор,
    размеры и положение, путь к рендеру, индекс спрайта -->
        <Image ID="off" Rect="full" Source="lamp.png" SpriteIndex="0"/>
        <!-- подэлемент изображение, задаются следующие свойства: уникальный
идентификатор,
    размеры и положение, путь к рендеру, индекс спрайта, видимость -->
        <Image ID="on" Rect="full" Source="lamp.png" SpriteIndex="1"
Visible="false"/>
        <!-- задается свойство lua-кодом -->
        <Property PropertyID="state" Default="off">
            <LUA><![CDATA[
                if value == "default" then
                    Set("state.=off")
-- состояние горящей лампы
                elseif value == "on" then
                    Set("on.Visible=ani(0:true)")
                    Set("on.Opacity=ani(0:0;200:1.0)")
                    Set("off.Visible=ani(0:true;200:false)")

```

```

        Set("off.Opacity=ani(0:1.0;200:0)")
-- состояние выключенной лампы
    elseif value == "off" then
        Set("on.Visible=ani(0:true;200:false)")
        Set("on.Opacity=ani(0:1.0;200:0)")
        Set("off.Visible=ani(0:true)")
        Set("off.Opacity=ani(0:0;200:1.0)")
-- состояние мигающей лампы
    elseif value:sub(1, 5) == "blink" then
        local t1, t2 = value:match("blink(%d+),(%d+)")
        if t1 == nil then t1 = 1000 end
        if t2 == nil then t2 = 1000 end
        Set("on.Visible=ani(0:true)")
        Set("off.Visible=ani(0:true)")
        local dt1 = 200 + tonumber(t1)
        local dt2 = dt1 + 200
        local dt3 = dt2 + tonumber(t2)
        local s_on =
"ani(0:0.0;200:1.0; "..dt1..":1.0; "..dt2..":0.0; "..dt3..":0.0;loop:true)"
        local s_off =
"ani(0:1.0;200:0.0; "..dt1..":0.0; "..dt2..":1.0; "..dt3..":1.0;loop:true)"
        Set("on.Opacity="..s_on)
        Set("off.Opacity="..s_off)
    end
]]></LUA>
</Property>
</Element>
</Library>
</root>

```

Файл со строковыми данными

Основная статья «[Строковые данные](#)». Создайте в папке configs файл text.xml. Название файла должно совпадать с названием, указанным в файле описания панелей. Необходимо добавить все строковые поля. Файл должен содержать следующий код:

```

<!-- объявление xml -->
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!-- корневой элемент -->
<root>
  <!-- строковые данные, задается следующее свойство: уникальный идентификатор -->
  <Text ID="txtEmptyProject">
    <!-- языковая версия, задается следующие свойства: язык, текст -->
    <Entry Lang="ru" Value="Пустой проект" />
    <Entry Lang="en" Value="Empty project" />
  </Text>
  <!-- строковые данные, задается следующее свойство: уникальный идентификатор -->
  <Text ID="txtExit">
    <!-- языковая версия, задается следующие свойства: язык, текст -->
    <Entry Lang="ru" Value="Выход" />
    <Entry Lang="en" Value="Exit" />
  </Text>
</root>

```

```

</Text>
<!-- строковые данные, задается следующее свойство: уникальный идентификатор -->
<Text ID="txtStart">
  <!-- языковая версия, задается следующие свойства: язык, текст -->
  <Entry Lang="ru" Value="Старт" />
  <Entry Lang="en" Value="Start" />
</Text>
<!-- строковые данные, задается следующее свойство: уникальный идентификатор -->
<Text ID="txtStop">
  <!-- языковая версия, задается следующие свойства: язык, текст -->
  <Entry Lang="ru" Value="Стоп" />
  <Entry Lang="en" Value="Stop" />
</Text>
</root>

```

Lua-код

Основная статья «[Lua-код](#)». Создайте в папке configs файл start.lua. Название файла должно совпадать с названием, указанным в главном конфигурационном файле. Необходимо привязать панель к выводу, описать обработчик кнопок и таймеров и привязать их к событиям. Файл должен содержать следующий код:

```

-- start.lua
Panel('main_viewport', 'main')
Include("timer.lua")

function H_Exit(_ev)
  sys.Exit()
end
AddEventHandler('main.btExit.Mouse.LeftPress', 'H_Exit')

-- timer.lua
-- в Clock хранится текущее время в виде секунд - seconds и минут - minutes, а
-- также интервал таймера в миллисекундах - interval и статус таймера - on =
-- false|true
Clock = {seconds = 0; -- секунды
         minutes = 0; -- минуты
         interval = 1000; -- интервал обновления
         on = false; -- статус таймера
        }

-- При нажатии на кнопку Старт - включаются таймеры tick и lamp и текст кнопки
-- становится Стоп
-- При нажатии на кнопку Стоп - таймер tick выключается и текст кнопки становится Старт
function H_startstoptimer(ev)
  if Clock.on == false then
    Clock.on = true
    SetText('main.btStartStop.text.Text', 'txtStop')
    -- Включаем таймер на Clock.interval с циклическим срабатыванием (once = false)
    sys.StartTimer("tick", Clock.interval, false)
  end
end

```

```
-- Включаем лампу и одноразовый таймер на 5с
Set('main.Lamp.state.', 'on')
sys.StartTimer("lamp", 5000)
else
    Clock.on = false
    SetText('main.btStartStop.text.Text', 'txtStart')
-- Останавливаем циклический таймер
    sys.StopTimer("tick")
end
end
AddEventHandler('main.btStartStop.Mouse.LeftPress', 'H_startstoptimer')

-- По таймеру tick обновляются часы
function H_refresh(ev)
    Clock.seconds = Clock.seconds + Clock.interval/1000
    if Clock.seconds > 60 then
        Clock.minutes = Clock.minutes + 1
        Clock.seconds = Clock.seconds - 60
    end
    if Clock.minutes > 60 then
        Clock.minutes = Clock.minutes - 60
    end
    -- Устанавливаем значение цифровых часов
    Set("main.Clock.value.", string.format("%02d%02d", Clock.minutes,
math.floor(Clock.seconds)))
end
AddEventHandler('timer.tick.finished.', 'H_refresh')

-- По таймеру lamp выключается лампа
function H_lamp(ev)
    Set('main.Lamp.state.', 'off')
end
AddEventHandler('timer.lamp.finished.', 'H_lamp')
```

Запуск проекта

Прямые ссылки на готовый проект:

- [win_timer_project.zip](#)
- [rpi_timer_project.zip](#)

В следующих статьях описано как запускать проекты:

- [Установка и запуск платформы на Windows](#)
- [Установка и запуск платформы на Raspbian](#)

From:

<https://wiki.lambda-mu.com/> - **Lambda-Mu Wiki**

Permanent link:

<https://wiki.lambda-mu.com/lm2/ce/timer?rev=1596650525>

Last update: **2020/11/30 14:12**

