Модули INSTEAD

23.01.13

Оглавление

Модуль click

Подключение

`require "click"`

Тип

расширение кода

Зависимости

INSTEAD не ниже 1.8.0

Описание

Модуль позволяет удобным способом отслеживать клики мышкой по картинке сцены. При этом, во время клика будет вызван обработчик `click` текущей сцены, или одноименный обработчик `game.click`. В параметрах передаются координаты клика `(x, y)` в системе координат оригинального (не масштабированного) изображения.
Координата `(0, 0)` соответствует верхнему левому углу.

Если необходимо получать события кликов в любой области игрового экрана (если щелчок пришелся на фон), необходимо установить переменную `click.bg`:

```
click.bg = true
```

При этом, в обработчик сначала придут координаты клика фона и координаты клика картинки (если клик пришелся на картинку);

```
game.click(s, x, y, px, py)
```

Внимание!!! В режиме прямого доступа (см. модуль sprites) координаты кликов всегда приходят относительно фона.

Если необходимо получать события не только нажатой клавиши, но и события при отпускании клавиши, используйте переменную `click.press`:

```
click.press = true;
```

Тогда, в обработчик придет булево значение `press`. При нажатии кнопки мыши `press` устанавливается в `true`, при отпускании в `false`.

```
game.click(s, press, x, y, px, py)
```

Если необходимо получать клики не только от первой кнопки мыши, используйте переменную `click.button`. При этом, в обработчик будет передан код кнопки мыши.

```
click.button = true
```

Примеры использования

```
-- $Name: Тест модуля click$
-- $Version: 0.1$
-- $Author: instead$

instead_version = "1.8.0"

require "click"

game.click = function(s, x, y)
    p ("Клик упал по координатам: ", x, ", ", y);
end

main = room {
    forcedsc = true,
    nam = 'Лес',
    pic = 'house.png',
    dsc = [[ Вы вышли на поляну где стоит домик.
            Вы видите, что дверь открыта.
            Нужно скорее попасть внутрь,
            где-то по лесу бродит медведь. ]],
    click = function(s, x, y)
        if x > 80 and x < 200 and y > 225 and y < 325 then
            walk('house');
        else
            return 'Это не дом и тем более не дверь.';
```

```
        end;
    end,
};

house = room {
    forcedsc = true,
    nam = 'Дом',
    pic = 'door.png',
    dsc = [[ Вы сидите у себя в уютном домике.
             Перед собой вы видите дверь
             на улицу. Медведь остался голодным.
             Тест успешно пройден.
             ]],
};

--...
```

- Скачать [76KB]

Модуль format

Подключение

`require "format"`

Тип

расширение кода

Зависимости

нет

Описание

Модуль `format` выполняет форматирование вывода. По умолчанию все настройки выключены.
Вы так же можете пользоваться модулями **para**, **dash**, **quotes** для включения отдельных настроек.

Примеры использования

```
-- $Name: Моя игра$
-- $Version: 0.1$
-- $Author: Я$

instead_version = "1.8.0"

require "format"

format.para = false -- отступы в начале абзаца;
format.dash = false -- замена двойного - на тире;
format.quotes = false -- замена " " на << >>;
format.filter = nil -- пользовательская функция замены;
...
```

ru:gamedev:modules:hideinv

Модуль hideinv

Подключение

`require "hideinv"`

Тип

расширение кода

Зависимости

INSTEAD не ниже 1.8.0

Описание

Модуль `hideinv` позволяет временно прятать объекты в инвентаре для выбранных комнат.

Если ваша игра использует **модуль xAct** и вы хотите использовать свойства модуля `hideinv` в комнатах типа `xroom`, включите модуль `hideinv` раньше модуля `xAct`.

Примеры использования

При определении комнаты, просто задайте атрибут **hideinv**, например:

```
-- $Name: Моя игра$
-- $Version: 0.1$
-- $Author: Я$

instead_version "1.8.0"

require "hideinv"

happyend = room {
    nam = 'Конец';
    hideinv = true;
    dsc = [[ Вы прошли игру! ]];
}
...
```

ru:gamedev:modules:kbd

Модуль kbd

Подключение

`require "kbd"`

Тип

игровой

стандартная библиотека

Зависимости

INSTEAD не ниже 1.8.0

Описание

Если вы хотите организовать ввод текста с клавиатуры, используйте **модуль keyboard**.

Модуль позволяет удобным способом обрабатывать события нажатия/отжатия клавиш клавиатуры.
Для перехвата событий используйте `hook_keys('<key_1>', '<key_2>', ..., '<key_n>')`, для отмены перехвата используйте `unhook_keys('<key_1>', '<key_2>', ..., '<key_n>')`, где клавиши `'<key_1>', '<key_2>', ..., '<key_n>'` – это список текстовых идентификаторов.

Событие придет в виде вызова метода `kbd` у текущей комнаты или, если такой метод не определен, у объекта `game`.

Ниже приводится список идентификаторов клавиш:

Идентификатор	Клавиша	Идентификатор	Клавиша
a	Английская «a»	[/]	/ на цифровой клавиатуре
b	Английская «b»	[*]	* на цифровой клавиатуре}
c	Английская «c»	[-]	- на цифровой клавиатуре}
...	...	[+]	+ на цифровой клавиатуре
z	Английская «z»	enter	Enter на цифровой клавиатуре
0	0	[0]	0 на цифровой клавиатуре
1	1	[1]	1 на цифровой клавиатуре
2	2	[2]	2 на цифровой клавиатуре
3	3	[3]	3 на цифровой клавиатуре
...	...	...	...
9	9	[9]	9 на цифровой клавиатуре
return	Enter	[.]	. на цифровой клавиатуре
escape	Esc	left ctrl	Левый Ctrl
backspace	Backspace	left shift	Левый Shift
tab	Tab	left alt	Левый Alt
space	Пробел	right ctrl	Правый Ctrl
-	-	right shift	Правый Shift
=	=	right alt	Правый Alt
[	Английская «[»	numlock	Num Lock
]	Английская «]»	caps lock	Caps Lock
\	\	scroll lock	Scroll Lock
;	Английская «;»	’	Английская «’»
‘	Английская «‘»	,	Английская «,»
.	Английская «.»	/	Английская «/»
f1	F1	print screen	Print Screen
f2	F2	pause	Pause
f3	F3	insert	Insert
...	...	home	Home
f12	F12	end	End
delete	Delete		
page up	Page Up	page down	Page Down
up	Up (стрелка курсора вверх)	down	Down (стрелка курсора вниз)
right	Right (правая стрелка курсора)	left	Left (левая стрелка курсора)

Примеры использования

```
-- $Name: Моя игра$
-- $Version: 0.1$
-- $Author: Я$

instead_version = "1.8.0"

require "kbd"

function init()
    hook_keys('a','b', 'c', 'd', 'e', 'f', 'g', 'h',
                'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
                'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z');
    hook_keys('space', 'backspace', 'return');
    hook_keys('1', '2', '3', '4', '5', '6', '7', '8', '9', '0');
end

main = room {
    nam = 'kbd';
    dsc = [[Нажимайте клавиши.]];
    kbd = function(s, down, key)
        p [[Событие от клавиши:]];
        p (key);
        if down then p [[нажата]]; else p [[отжата]]; end
        pn "";
    end
};
```

Модуль prefs

Подключение

`require "prefs"`

Тип

расширение кода

стандартная библиотека

Зависимости

INSTEAD не ниже 1.8.0

Описание

Этот модуль позволяет сохранять настройки игры.
Другими словами, сохраненная информация не зависит от состояния игры. Такой механизм можно использовать, например, для реализации системы достижений или счетчика количества прохождений игры.

`prefs` это объект, все переменные которого будут сохранены.

Сохранить настройки:

```
prefs:store()
```

Удалить все настройки:

```
prefs:purge()
```

Загрузка настроек выполняется автоматически при инициализации игры (перед вызовом функции **init()**), но вы можете инициировать загрузку и вручную:

```
prefs:load();
```

Примеры использования

```
-- $Name: Моя игра$
-- $Version: 0.1$
-- $Author: instead$

instead_version = "1.8.0"

require "click"
require "prefs"

prefs.counter = 0;
```

```
game.click = function(s)
    prefs.counter = prefs.counter + 1;
    prefs:store();
    p("На данный момент сделано ", prefs.counter ," кликов");
end;

game.pic = 'clickme.png';

main = room {
    forcedsc = true,
    nam = "Комната кликов",
    dsc = [[ Этот тест был написан специально
            для проверки работы модуля <<click>> ]];
};
--...
```

- Скачать [20KB]

Модуль timer

Подключение

require “timer”

Тип

расширение кода

Зависимости

нет

Описание

Модуль позволяет получать события от таймера удобным способом. Функцию обработчика таймера выполняет game.timer. Если game.timer возвращает пустое значение, сцена не перерисовывается. В противном случае, возвращаемое значение интерпретируется как действие.

Вы можете делать локальные для комнаты обработчики timer. Если в комнате объявлен обработчик timer, он вызовется вместо game.timer

Примеры использования

```
game.timer = function(s)
    set_sound('gfx/beep.ogg');
    p "Timer:"
    p (time())
end
function init()
    timer:set(1000)
end
```

```
myroom = room  {
    entered = function(s)
        timer:set(1000);
    end;
    timer = function(s)
        timer:stop();
        walk 'myroom2';
    end;
    nam = 'Проверка таймера';
    dsc = [[Ждите.]];
}
```

ru:gamedev:modules:xact

Модуль xact

Подключение

require “xact”

Тип

расширение кода

Зависимости

нет

Описание

Модуль позволяет делать ссылки на объекты из других объектов, реакций и **life** методов в форме: *{объект(параметры)/текст}*.

Где *'объект'* – это сам объект или атрибут **nam** объекта.

(Параметры) – необязательные параметры в виде: (текст, текст, ...)

'Текст' – то, как ссылка выглядит в игре для игрока.

Модуль содержит в себе функцию **xact**, которая создает объект – простейшую реакцию. Первый параметр функции – имя, второй – реакция, которая может быть строкой, функцией или **code**.

Модуль содержит в себе реализацию комнаты с расширенным описанием: **xroom**. Если в такой комнате задать атрибут **xdsc**, то он будет выведен в области описаний объектов.

Функция **xdsc** позволяет более гибко управлять выводом текста в области предметов. *См. примеры.*

Модуль **xact** содержит в себе специальный **xact** – **«xwalk»**, который позволяет делать переходы по ссылкам, как в книгах играх. *См. примеры.*

Примеры использования

```
main = room {
    nam = 'Начало';
    forcedsc = true;
    dsc = [[От автора. Эту игру я писал очень  {note1|долго}.]];
    obj = {
        xact('note1', [[Больше 10 лет.]]);
    }
}
```

```
main = room {
    nam = 'Комната';
    forcedsc = true;
    dsc = [[Я в комнате.]];
    xdsc = [[ Я вижу {apple|яблоко} и {knife|нож}. ]];
    other = [[ Еще здесь лежат {chain|цепь} и {tool|пила}.]];
    obj = {
        xdsc(), -- 'xdsc method by default'
        xdsc 'other',
```

```
        'apple', 'knife', 'chain', 'tool',
    }
}
```

```
main = xroom {
    nam = 'Комната';
    forcedsc = true;
    dsc = [[Я в комнате.]];
    xdsc = [[ Я вижу {apple|яблоко} и {knife|нож}. ]];
    obj = {
        'apple', 'knife', 'chain', 'tool',
    }
}
```

```
main = room {
    nam = 'Начало';
    forcedsc = true;
    dsc = [[ Начать {xwalk(startgame)|приключение}? ]];
}
startgame = room {
    dsc = [[ В одной далекой-далекой галактике... ]];
}
```

ru:gamedev:modules:sprites

Модуль sprites

Подключение

require «sprites»

Тип

игровой/расширение кода

Зависимости

theme

Описание

Начиная с версии 1.4.0 INSTEAD поддерживает расширенные возможности для работы с изображениями, позволяющие в том числе делать 2d игры.

Модуль sprites предоставляет api, содержащие следующие функции:

sprite.load(file_name)

Загрузка спрайта из файла изображения. При этом функция вернет дескриптор загруженного спрайта (далее spr).

sprite.box(w,h,[color[,alpha]])

Создание спрайта, закрашенного заданным цветом.

sprite.blank(w,h)

Создание прозрачного спрайта.

sprite.free(spr)

Освобождение спрайта.

sprite.screen()

Возвращает спрайт - игровой экран. Используется только в режиме прямого доступа.

sprite.font_scaled_size(size)

Возвращает размер шрифта с учетом масштабирования шрифтов.

sprite.font(font_path, size)

Загружает шрифт, возвращает дескриптор загруженного шрифта (далее font).

sprite.free_font(font)

Выгружает шрифт.

sprite.font_height(font)

Возвращает высоту шрифта в пикселях.

sprite.alpha(spr, alpha)

Создает новый спрайт с заданной прозрачностью alpha (255 - не прозрачно).

sprite.dup(spr)

Создает копию спрайта.

sprite.scale(spr, xs, ys, [smooth])

Масштабирование спрайта, для отражений используйте масштаб -1.0. (медленно! не для реального времени).

sprite.rotate(spr, angle, [smooth])

Поворот спрайта на заданный угол в градусах (медленно! не для реального времени).

sprite.text(font, text, col, [style])

Создание текстового спрайта, col - здесь и далее - цвет в текстовом формате (в формате '#rrggbb' или 'текстовое название цвета').

sprite.size(spr)

Возвращает ширину и высоту спрайта в пикселях.

sprite.text_size(font, text)

Вычисляет размер, который будет занимать текстовый спрайт, без создания спрайта.

sprite.draw(src_spr, fx, fy, fw, fh, dst_spr, x, y, [alpha])

Рисование области src спрайта в область dst спрайта (задание alpha сильно замедляет выполнение функции).

sprite.draw(src_spr, dst_spr, x, y, [alpha])

Рисование спрайта, укороченный вариант; (задание alpha сильно замедляет выполнение функции).

sprite.copy(src_spr, fx, fy, fw, fh, dst_spr, x, y, [alpha])

Копирование содержимого спрайта (рисование - замещение)

sprite.compose(src_spr, fx, fy, fw, fh, dst_spr, x, y, [alpha])

Копирование содержимого спрайта (рисование - с учетом прозрачности обоих спрайтов).

sprite.copy(src_spr, dst_spr, x, y, [alpha])

Копирование содержимого спрайта (рисование - замещение), укороченный вариант.

sprite.fill(spr, x, y, [w, h, [col]])

Заполнение спрайта цветом.

sprite.pixel(spr, x, y, col, [alpha])

Заполнение пикселя спрайта.

sprite.pixel(spr, x, y)

Взятие пикселя спрайта (возвращает цвет в текстовой форме).

Примеры использования

Внимание!!! Состояние спрайтов не попадает в файл сохранения игры, поэтому задача восстановления игровой ситуации на основе сохраняемых переменных лежит на авторе игры.

Общие рекомендации

В функции **init** можно загружать и создавать те спрайты, которые будут необходимы во время цикла всей игры, например:

```
function init()
    bg = sprite.load 'background.png'
    font = sprite.font ('sans.ttf', 32);
end
```

В функции **start** вы можете восстанавливать игровую ситуацию на основе сохраненных переменных. **start** выполняется после загрузки игры или после первого запуска игры, например:

```
function start()
    if here() == main then
        main.pic = sprite.text(font, 'BIG ADVENTURE', 'black');
    end
end
```

Если вы создаете временные спрайты, освобождайте их, когда они больше не нужны, например:

```
function show_score()
    local t = sprite.text(font,'Score:'..tostring(score), 'white');
    sprite.draw(t, sprite.screen(), 0, 0);
    sprite.free(t);
end
```

Спрайты могут быть встроены в игру как и любая другая графика – с помощью **img/imgl/imgr** или присвоены переменной **pic** сцены, но в последнем случае, любое изменение содержимое спрайта **pic** (например, в обработчике таймера) будет отражено в реальном времени в игре. Эту особенность можно использовать для анимационных квестов или заставок.

Например:

```
instead_version '1.8.1'
require 'sprites'
require 'timer'
main = room {
```

```
    nam = 'demo';
    pic = sprite.load 'box:320x200,black';
}

function init()
    timer:set(30);
end

game.timer = function()
    sprite.pixel(main.pic, rnd(320), rnd(200), 'white');
end
```

Если обработчик не возвращает ничего, то игровая сцена не изменяется, за исключением модификаций pic сцены, если это спрайт.

Игра может задействовать режим прямого доступа и рисовать непосредственно в экранную область INSTEAD. Переключение в режим осуществляется с помощью параметра темы:

```
scr.gfx.mode = direct
```

Вы можете задать этот параметр в **theme.ini** игры или менять его динамически, с помощью модуля **theme**.

В режиме прямого доступа, все отрисовки в специальный спрайт **sprite.screen()** отображаются в реальном времени.

Таким образом, если вы пишите 2d-игру на INSTEAD, типовой алгоритм ее работы выглядит следующим образом.

1. *init()* – загрузка спрайтов
2. *start()* – задание начальных значений или восстановление;
3. *game.timer()* – отрисовка кадра игры (модуль timer);
4. *game.click()* – получение событий мыши (модуль click);
5. *game.kbd()* – получение событий клавиатуры (модуль kbd);

INSTEAD всегда скрывает факт масштабирования от игры, поэтому, обычно игра работает независимо от выбранного разрешения. Все размеры и координаты выглядят так, как будто масштабирования нет. В отдельных случаях, в результате погрешностей округления это может стать проблемой (например подгонка тайлов пиксель в пиксель). В этом случае автор может запретить масштабирование:

```
scr.gfx.scalable = 0
```

В INSTEAD существует возможность отслеживать интервалы времени в миллисекундах. Для этого используйте функцию **stead.ticks()**.

Опрос или установка координат курсора мыши: **stead.mouse_pos([x, y])**.

Пример работы со спрайтами:

```
instead_version "1.8.1"
require "timer"
require "sprites"
spr = sprite
function init()
    fnt = spr.font(theme.get 'win.fnt.name', 32);
    ball = spr.text(fnt, "INSTEAD 1.4.0", 'white', 1);
    ballw,ballh = spr.size(ball);
    bg = spr.load 'box:640x480,black';
    line = spr.load 'box:320x8,lightblue';
end

function start()
    timer:set(10)
    G = 9.81
    by = -ballh
    bv = 0
    bx = 320
    t1 = stead.ticks()
end

function phys()
    local t = timer:get() / 1000;
    bv = bv + G * t;
    by = by + bv * t;
    if by > 400 then
        bv = - bv
    end
end

game.timer = function(s)
    local i
    for i = 1, 10 do
```

```
        phys()
    end
    if get_ticks() - t1 >= 20 then
        spr.copy(bg, spr.screen(), 0, 0);
        spr.draw(ball, spr.screen(), (640 - ballw) / 2, by - ballh/2);
        spr.draw(line, spr.screen(), 320/2, 400 + ballh / 2);
        t1 = get_ticks()
    end
end
```

Файл **theme.ini**

```
scr.w = 640
scr.h = 480
scr.gfx.mode = direct
```

Еще один вариант, пропускающий кадры при необходимости:

```
game.timer = function(s)
    local i
    for i = 1, 10 do
        phys()
    end
    if get_ticks() - t1 >= 15 then
        t1 = get_ticks()
        return
    end
    t1 = get_ticks()
    spr.copy(bg, spr.screen(), 0, 0);
    spr.draw(ball, spr.screen(), (640 - ballw) / 2, by - ballh/2);
    spr.draw(line, spr.screen(), 320/2, 400 + ballh / 2);
end
```

ru:gamedev:modules:sound

Модуль sound

Подключение

require «sound»

Тип

игровой/расширение кода

Зависимости

нет

Описание

Данный модуль существует в INSTEAD начиная с версии 1.4.0 и предоставляет расширенные возможности по работе со звуком. Эти возможности главным образом востребованы при разработке двухмерных игр.

`sound.load(filename)` – возвращает дескриптор звука (далее snd);

`sound.free(snd)` – освобождает звук (внимание! данная функция не останавливает проигрывание звука!);

`sound.play(snd, [channel], [loop])` – запуск звука на проигрывание, канал от 0 до 7, loop - количество проигрываний, 0 - вечно. Имейте в виду, что канал 0 практически всегда занят звуком клика.

`sound.stop([channel])` – остановить проигрывание выбранного канала или всех каналов;

`sound.playing([channel])` – узнать проигрывается ли звук на любом канале или на выбранном канале; если выбран конкретный канал, функция вернет хандл проигрываемого в данный момент звука или nil. Внимание! Звук клика не учитывается и обычно занимает 0 канал.

`sound.pan(chan, l, r)` – задание паннинга. Канал, громкость левого[0-255], громкость правого[0-255] каналов. Необходимо вызывать перед проигрыванием звука, чтобы имело эффект;

`sound.vol(vol)` – задание громкости звука (и музыки и эффектов) от 0 до 127.

Примеры использования

```
instead_version "1.4.0"
require "sound"

init = function()
   hello = sound.load "hello.ogg"
end

start = function()
```

```
    sound.play(hello)
end
```

ru:gamedev:modules:nouse

Модуль nouse

Подключение

require «nouse»

Тип

расширение кода

Зависимости

INSTEAD 1.7.0

Описание

Модуль позволяет более удобным способом прописывать реакции на действия, не предусмотренные игрой.

У каждого объекта могут быть атрибуты (методы) nouse или noused. Если в результате действия игрока одним предметом (a) на другой (b), реакция игры не предусмотрена (пустой вывод), то будет вызван метод nouse у объекта a. Если вывод a.nouse пустой, будет вызван b.noused. Если вывод b.noused пустой, будет вызван метод game.nouse.

Примеры использования

```
instead_version "1.7.0"
require "nouse"

game.nouse = 'Бесполезно';

worm = obj {
    nam = 'червячок';
    inv = 'Маленький.';
    use = function(s, w)
        if w == apple then
```

```
            p 'Он уже сыт.'
            return
        end
    end;
    noused = 'Не буду его трогать.'
}

apple = obj {
    nam = 'яблоко';
    dsc = 'На столе лежит {яблоко}.';
    tak = 'Я взял яблоко.';
    inv = function(s)
        if not taken 'worm' then
            p [[В яблоке червяк!]]
            take 'worm'
        else
            p 'Оставлю про запас.';
        end
    end;
    use = function(s, w)
        if w == table then
            drop(s, table)
            p 'Я вернул яблоко на стол.';
        end
    end;
    nouse = 'Яблоко тут не поможет.';
}

table = obj {
    nam = 'стол';
    dsc = 'В центре комнаты стоит {стол}.';
    act = 'Стол как стол.';
    obj = { 'apple' };
}

tree = obj {
    nam = 'пальма';
    dsc = 'У стены стоит {пальма}.';
    act = 'Декоративная...';
    noused = 'Это не поможет пальме.';
}

main = room {
    nam = 'комната';
    obj = { 'table', 'tree' };
```

```
}
```

ru:gamedev:modules:counters

Модуль counters

Подключение

`require "counters"`

Тип

расширение кода

Зависимости

INSTEAD не ниже 1.7.0

Описание

Модуль ведет статистику по действиям игрока: `use, inv, walk и act`.
Счетчики увеличиваются перед выполнением действия, таким образом, учитываются даже не успешные переходы игрока с помощью `walk`.

Действие **tak** рассматривается как **act**.

Функции для чтения счетчиков:

`inv_count()`	проверка состояния счетчика инвентаря
`act_count()`	проверка состояния счетчика действий
`use_count()`	проверка состояния счетчика взаимодействий
`walk_count()`	проверка состояния счетчика переходов

Каждая функция, вызванная без параметров, возвращает общее число событий данного типа.

Если первый параметр число, то устанавливается общий счетчик событий данного типа.

```
if act_count() == 1 then
    ...
end;
```

Если первый параметр функции это объект, возвращается число событий для данного объекта.

```
if act_count(s) < 2 then
      p ("Общее количество событий для объекта s ", act_count(s));
end;
```

Если первый параметр функции это объект, а второй – число, то устанавливается число событий для данного объекта.

```
if act_count(s) == 1 then
     act_count(s, 2);
     p ("Установили для объекта s счетчик событий равный 2.");
end;
```

Примеры использования

```
--$Name: Тест модуля counters$
--$Author: instead$
--$Version: 0.1$

instead_version "1.8.2";

require "counters"

apple = obj {
    nam = 'яблоко',
    dsc = 'На столе лежит {яблоко}.',
    tak = function(s)
        pn ('Текущий счетчик TAK для яблока -- ', act_count(s));
        p 'Я взял яблоко.';
    end,
    inv = function(s)
        pn ('Текущий счетчик INV для яблока -- ', act_count(s));
        p 'Красное!'
    end,
    use = function(s, w)
```

```
        pn ('Текущий счетчик USE для яблока -- ', use_count(s));
        if use_count(s) < 3 then
            if w == table then
                p 'Я положил яблоко на стол.';
                drop(s, w);
            else
                pn 'Ничего не понимаю.';
            end;
        else
            pn ('Счетчик USE > 3. Уже ничего не получится.');
        end;
    end,
}:disable();

table = obj {
    nam = 'стол',
    dsc = 'В центре комнаты стоит {стол}.',
    act = function(s){{:ru:gamedev:modules:instead-counters-0.1.zip}}
        pn ('Текущий счетчик ACT для стола -- ', act_count(s));
        if act_count(s) == 1 then
            p 'На столе яблоко!';
            apple:enable();
        else
            p 'Стол как стол.';
        end;
    end,
    obj = { 'apple' },
};


main = room {
    nam = 'комната',
    obj = { 'table' },
};
```

- Скачать [0.9 KB]

Модуль wroom

Подключение

```
require "wroom"
```

Тип

расширение кода

стандартная библиотека

Зависимости

INSTEAD не ниже 1.7.0

Описание

Модуль `wroom` является более продвинутым вариантом упрощенной сцены переходов `vroom` и позволяет:

- задавать два отображаемых имени перехода (для ситуации, когда переход никогда не был использован, и для ситуации, когда переход был выполнен хотя бы один раз);
- задавать отображаемые имена в виде функций;
- задавать точку назначения в виде функции, возвращающей объект назначения;

Кроме того, `wroom` полностью совместим по синтаксису с `vroom(имя перехода, сцена назначения)`.

Примеры использования

```
-- $Name: Моя игра$
-- $Version: 0.1$
-- $Author: Я$

instead_version = "1.8.0"

require "wroom"

main = room {
    nam = 'Комната 1';
        dsc = function(s)
        if from() == r2 then
             p [[ Вы пришли из комнаты "В неведомое". ]];
        else
             p [[ Вы находитесь в главной комнате. ]];
        end,
    way = { wroom('В неведомое', 'В комнату r2', 'r2') };
```

```
};

r2 = room {
    nam = 'Комната 2';
        dsc = [[ Вы зашли из "главной" комнаты. ]];
    way = { wroom('Назад', main) };
};
```

Модуль nolife

Подключение

require “nolife”

Тип

расширение кода

Зависимости

INSTEAD 1.7.0

Описание

Модуль Nolife позволяет временно выключать life события (и триггеры) для выбранных комнат.

Примеры использования

При определении комнаты, просто задайте аттрибут nolife, например:

```
require "hideinv"
require "nolife"

happyend = room {
    nam = 'Конец';
    hideinv = true;
    nolife = true;
    dsc = [[ Вы прошли игру! ]];
}
```

ru:gamedev:modules:proxymenu

Модуль proxymenu

Подключение

require «proxymenu»

Тип

игровой/расширение кода

Зависимости

INSTEAD 1.7.0

Описание

proxymenu позволяет делать игры в стиле адвенчур для ZX-Spectrum. Примерами таких игр на INSTEAD являются: Зеркало, Kayleth, Резервная копия.

При использовании proxymenu, в отличие от классических игр INSTEAD, предполагается, что существуют различные варианты действий. Например: осмотреть, взять, бросить, говорить, отдать и др. При этом все взаимодействие с объектами происходит через область инвентаря.

Для создания элемента меню нужно воспользоваться одной из функций:

- obj_menu - для создания действия, в котором участвует только один объект;
- use_menu - для создания действия, в котором задействованы два объекта;
- act_menu - для создания действия без объекта;

И добавить полученный элемент меню в игрока.

Рассмотрим все три функции.

Действие без объекта: act_menu(имя, название обработчика)

Например:

```
game.rest = function(s)
    p [[Я отдохнул.]]
end
rest = act_menu("ОТДОХНУТЬ", "rest")
place(rest, me())
```

ВНИМАНИЕ: Модуль proxymenu переопределяет функцию получения инвентаря inv(), поэтому для помещения предметов (а не пунктов меню!) следует использовать inv():add(), inv():del() и прочее, или put/remove/прочее(что, inv()), вместо put/remove/прочее(что, me()). Таким образом, для работы с объектами-меню используйте me(), а для игрового инвентаря: inv().

Как видим, при клике на пункт ОТДОХНУТЬ, вызовется обработчик game.rest, так как мы явно задали «rest» вторым параметром act_menu.

Действие, в котором участвует один объект: obj_menu(имя, название обработчика, объекты сцены?, объекты инвентаря?, переходы?)

Например:

```
take_menu = act_menu("ВЗЯТЬ", "take", true);
place(take_menu, me())
```

Теперь, при щелчке на «ВЗЯТЬ», в выпадающем списке будут представлены объекты сцены, так как мы явно указали это через true (два последних параметра пусты – что в данном случае синоним задания false).

При щелчке на объекте в списке «ВЗЯТЬ», будет вызвана следуюая цепочка обработчиков:

- game.before_take – если определена. При возврате false – цепочка вызовов прерывается;
- объект:take – если определена. При возврате false – цепочка вызовов прерывается;
- game.after_take – если определена. При возврате false – цепочка вызовов прерывается;
- game.take – если определена, и все предыдущие обработчики вернули пустоту;

В качестве примера рассмотрим:

```
game.after_take = function(s, w)
    take(w)
end

apple = obj {
    nam = 'яблоко';
    dsc = [[На полу лежит яблоко]]; -- {} здесь не нужны, dsc опционален.
    take = "Я взял яблоко.";
}
```

Теперь, яблоко можно взять.

Действие, в котором участвует два объекта:

use_menu(имя, название обработчика, название обратного обработчика, название обработчика сам-на-себя, брать со сцены?, брать из инвентаря?, первый объект должен быть в инвентаре?)

Например:

```
use_menu = use_menu('ИСПОЛЬЗОВАТЬ', 'useon', 'used', 'useit', true, true);
place(use_menu, me())
```

Теперь, при щелчке на «ИСПОЛЬЗОВАТЬ», в выпадающем списке будут представлены объекты сцены и инвентаря, так как мы явно указали это через true, (последний параметр пуст – что в данном случае синоним задания false).

При щелчке на объекте в списке «ВЗЯТЬ», курсор перейдет в режим использования, в котором будет ожидаться второй клик. После клика на второй объект, вызовется следующая цепочка обработчиков (obj, obj2 – объекты первого и второго клика):

- game.before_useon – если определена. При возврате false – цепочка вызовов прерывается;
- obj:useit() – если obj == obj2
- obj:useon(obj2) – если obj не равен obj2 и если определена. При возврате false – цепочка вызовов прерывается;
- obj2:used(obj) – если obj не равен obj2 и если прошлый обработчик пуст;
- game.after_useon – если определена. При возврате false – цепочка вызовов прерывается;
- game.useon – если определена, и все предыдущие обработчики вернули пустоту;

Следет отметить, что название обратного обработчика, и название обработчика сам-на-себя – не обязательные параметры.

Примеры использования

```
instead_version "1.6.3"
require "proxymenu"
require "hideinv"
```

```
game.forcedsc = true

minv = obj_menu('С СОБОЙ', 'exam', false, true);
mlook = obj_menu('ОСМОТРЕТЬ', 'exam', true);
mtake = obj_menu('ВЗЯТЬ', 'take', true);
mdrop = obj_menu('БРОСИТЬ', 'drop', false, true);
meat = obj_menu('ЕСТЬ', 'eat', true, true);
mpush = obj_menu('ТОЛКАТЬ', 'push', true);
muse = use_menu('ИСПОЛЬЗОВАТЬ', 'useon', 'used', 'useit', true, true);
mgive = use_menu('ОТДАТЬ', 'give', 'accept', false, true, true, true);
mwalk = obj_menu('ИДТИ', 'walk', false, false, true);

game.useit = 'Не помогло.'
game.use = 'Не сработает.'
game.give = 'Отдать? Ни за что!'
game.eat = 'Не буду это есть.'
game.drop = 'Еще пригодится.'
game.exam = 'Ничего необычного.'
game.take = 'Стоит ли это брать?'
game.push = 'Ничего не произошло.'

game.after_take = function(s, w)
    take(w)
end

game.after_drop = function(s, w)
    drop(w)
end

put(minv, me())
put(mlook, me())
put(mtake, me())
put(mdrop, me())
put(meat, me())
put(mpush, me())
put(muse, me())
put(mgive, me())
-- put(mwalk, me())

status = stat {
    _Turns = 0,
    life = function(s)
        s._Turns = s._Turns + 1;
    end;
    nam = function(s)
```

```
        return 'Статус игрока: '..s._Turns..'^';
    end
};
lifeon 'status'

put(status, me());

knife = obj {
    nam = 'ножик',
    dsc = 'На полу валяется ножик.',
    exam = 'Бесполезный перочинный ножик.',
}

main = room {
    nam = 'intro',
    hideinv = "true",
    dsc = 'Введение',
    exit = function(s)
        inv():add('knife');
    end,
    obj = { vway('next','{Дальше}.', 'r1') }
}

cube = obj {
    nam = 'куб',
    dsc = 'В центре комнаты находится куб.',
    take = 'Вы взяли куб',
    exam = 'Мультифункциональный куб -- написано на кубе.',
    drop = 'Вы положили куб.',
    useit = 'Как можно использовать куб?',
    talk = 'Вы поговорили с кубом.',
    eat = function(s)
        return 'Вы не можете разгрызть куб.', false;
    end,
    open = 'Вы открыли куб.',
    close = 'Вы закрыли куб.',
    push = 'Вы толкаете куб.',
    give = function(s, w)
        return 'Вы пытаетесь отдать куб объекту: '..deref(w)..'.', false
    end,
    useon = function(s, w)
        return 'Вы пытаетесь юзать куб на объект: '..deref(w)..'. Получилось!'
    end,
    used = 'Куб поюзан.',
};
```

```
sphere = obj {
    nam = 'сфера',
    dsc = 'В центре комнаты находится сфера.',
    take = 'Вы взяли сферу',
    exam = 'Мультифункциональная сфера -- написано на сфере.',
    drop = 'Вы положили сферу.',
    useit = 'Как можно использовать сферу?',
    talk = 'Вы поговорили с сферой.',
    eat = function(s)
        return 'Вы не можете разгрызть сферу.', false;
    end,
    open = 'Вы открыли сферу.',
    close = 'Вы закрыли сферу.',
    push = 'Вы толкаете сферу.',
    give = function(s, w)
        return 'Вы пытаетесь отдать сферу объекту: '..nameof(w)..'.', false
    end,
    useon = function(s, w)
        return 'Вы пытаетесь юзать сферу на объект: '..nameof(w)..'. Получилось!'
    end,
    used = 'Сфера поюзана.',
};

r1 = room {
    nam = 'комната',
    dsc = 'Вы в комнате',
    obj = { cube, sphere },
}
```

ru:gamedev:modules:dash

Модуль dash

Подключение

require “dash”

Тип

игровой

Зависимости

format

Описание

Заменяет последовательность символов – на символ –. Замена происходит *только* при выводе содержимого сцены.

Примеры использования

```
require "dash"
main = room {
    nam = 'Введение';
    dsc = [[ -- Ну, начнем!!!]];
}
```

ru:gamedev:modules:hotkeys

Модуль hotkeys

Подключение

require “hotkeys”

Тип

игровой

Зависимости

Kbd

Описание

Модуль Hotkeys используется для быстрого выбора фраз в диалогах с помощью цифровых клавиш 1-9 на клавиатуре.

Примеры использования

Пример использования достаточно очевиден: в диалогах при нажатии цифровых клавиш от одного до девяти выбирается реплика, соответствующая нажатой клавише. Реплику больше 9й выбрать горячей клавишей нельзя.

ru:gamedev:modules:para

Модуль para

Подключение

`require "para"`

Тип

игровой, оформление

Зависимости

Модуль format

Описание

Ставит отступ в начале каждого параграфа в соответствии с русской типографской традицией. Дополнение отступом производится *только* при выводе содержимого сцены.

Вы можете менять количество пробелов в отступе с помощью задания format.para_space:

```
format.para_space = "        ";
```

Примеры использования

```
-- $Name: Моя игра$
-- $Version: 0.1$
-- $Author: Я$

instead_version = "1.8.0"

require "para"

format.para_space = "        ";

main = room {
       nam = "Lorem Ipsum",
       dsc = [[
       Lorem ipsum dolor sit amet, consectetur adipiscing elit.
       Morbi id suscipit nisl. Praesent tincidunt ultricies volutpat.
       Praesent congue est eu ligula tincidunt posuere quis a tortor.
       Mauris cursus dolor vitae augue accumsan sit amet ultrices mauris venenatis.
```

```
        Nullam eu augue ipsum. Mauris convallis commodo pretium. Ut ultrices tempor dui et aliquam.^

        Nullam vitae adipiscing dui. Donec quam dolor, pellentesque ut placerat eu,
        scelerisque vel nisi. In posuere, nibh nec viverra mattis, nisl dui pellentesque
        ligula, pharetra convallis dolor leo ac mi. Etiam sagittis sem quis
        risus tristique ornare. Cras fermentum odio non est hendrerit sit amet
        ultricies enim dapibus. Class aptent taciti sociosqu ad litora torquent per
        conubia nostra, per inceptos himenaeos. Nullam hendrerit tempus lacus,
        at dignissim dolor laoreet ac. Mauris fringilla rhoncus massa id pulvinar.
        Aliquam erat volutpat.^

        Etiam porta enim id enim gravida fermentum. Donec sollicitudin ligula
        ut lacus sodales id venenatis purus semper. Nunc gravida venenatis massa,
        ac interdum nunc aliquam eget. Ut faucibus, ipsum eu euismod hendrerit,
        libero diam aliquet metus, ac suscipit urna nibh ac justo. Donec mollis
        orci quis sapien scelerisque ornare. Nullam ac velit vel lectus aliquet
        semper quis laoreet lectus. Suspendisse non ante arcu. In nulla urna,
        faucibus eu dapibus lacinia, aliquet ac eros. Integer adipiscing euismod imperdiet.
        Nulla pulvinar pharetra nulla, sit amet mollis magna porttitor sit amet.
        Ut a arcu vitae est consequat vehicula in in neque. Pellentesque sem ligula,
        faucibus eget porttitor vitae, bibendum sit amet metus. Integer condimentum
        molestie magna, ac mollis felis cursus nec. Morbi in sem nec nisl fringilla
        tempor. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
        Duis pellentesque purus ac ante eleifend aliquet.]];
};
```

Модуль quotes

Подключение

`require "quotes"`

Тип

игровой, оформление

Зависимости

Модуль format

Описание

Заменяет все двойные кавычки на типографские («ёлочки»).

Также заменяет „ (две запятые) и ” (два апострофа) на кавычки-«лапки» („”).

Замена происходит *только* при выводе содержимого сцены.

Рекомендуется к применению для соответствия русской типографской традиции. Напомним, что обычно используются «ёлочки», но для употребления кавычек в кавычках и для передачи прямой речи следует использовать «лапки».

`"Текст в елочках"`	Результат: «текст в елочках»
`,,Текст в лапках''`	Результат: „текст в лапках“
`_"Текст в елочках _"вложенный"_`	Результат: «текст в елочках «вложенный»

Примеры использования

```
-- $Name: Моя игра$
-- $Version: 0.1$
-- $Author: Я$

instead_version = "1.8.0"

require "quotes"

main = room {
    nam = "Lorem Ipsum",
    dsc = [[
            "Lorem ipsum dolor" sit ,,amet'', ''consectetur'' adipiscing elit. _"Duis cursus"_.
            ]];
};
...
```

Модуль theme

Подключение

require «theme»

Тип

игровой

Зависимости

нет

Описание

Начиная с версии 1.3.0, модуль **theme** позволяет модифицировать параметры темы на лету. Для этого, используются следующие функции:

```
-- настройка окна вывода
theme.win.geom(x, y, w, h)
theme.win.color(fg, link, alink)
theme.win.font(name, size, height)
theme.win.gfx.up(pic, x, y)
theme.win.gfx.down(pic, x, y)

-- настройка инвентаря
theme.inv.geom(x, y, w, h)
theme.inv.color(fg, link, alink)
theme.inv.font(name, size, height)
theme.inv.gfx.up(pic, x, y)
theme.inv.gfx.down(pic, x, y)
theme.inv.mode(mode)

-- настройка меню
theme.menu.bw(w)
theme.menu.color(fg, link, alink)
theme.menu.font(name, size, height)
theme.menu.gfx.button(pic, x, y)

-- настройка графики
theme.gfx.cursor(norm, use, x, y)
theme.gfx.mode(mode)
theme.gfx.pad(pad)
theme.gfx.bg(bg)

-- настройка звука
theme.snd.click(name);
```

Если необходимо изменить только часть параметров, в качестве неизменяемых параметров можно указывать значение nil. Например:

```
theme.win.font(nil, 64);
```

Существует возможность чтения текущих параметров тем:

```
theme.get 'имя переменной темы';
```

Возвращаемое значение всегда в текстовой форме.

А также, устанавливать их:

```
theme.set ('имя переменной темы', значение);
```

Начиная с версии INSTEAD 1.4.0 вы можете сбросить значение параметра темы на то, которое было установлено во встроенной теме игры:

```
theme.reset 'имя переменной';
theme.win.reset();
-- ...
```

Примеры использования

```
theme.gfx.bg "dramatic_bg.png";
theme.win.geom (0,0, theme.get 'scr.w', theme.get 'scr.h');
theme.inv.mode 'disabled'
```

ru:gamedev:modules:snapshots

Модуль snapshots

Подключение

require “snapshots”

Тип

игровой / расширение кода

Зависимости

нет

Описание

Модуль snapshots предоставляет возможность восстанавливать предварительно сохраненные состояния игры. В качестве примера, можно привести ситуацию, когда игрок выполняет в игре действие, ведущее к проигрышу. Модуль позволяет автору игры написать код так, что игрок вернется к предварительно сохраненному состоянию игры.

Для создания снапшота используйте функцию: make_snapshot(). В качестве параметра может быть задан номер слота.

Внимание!!! Снапшот будет создан *после* завершения текущего такта игры, так как только в этом случае гарантирована непротиворечивость сохраненного состояния игры.

Загрузка снапшота осуществляется restore_snapshot(). В качестве параметра может быть задан номер слота.

Удаление снапшота делается с помощью delete_snapshot(). Следует удалять ненужные снапшоты, так как они занимают лишнее место в файлах сохранения.

Примеры использования

```
instead_version "1.4.0"
require "xact"
require "snapshots"

theend = xroom {
    nam = 'Конец';
    xdsc = [[Вы проиграли!!! Но может было все {заново|по другому}?]];
    obj = {
        xact('заново', code [[ restore_snapshot() ]]);
        }
-- ...
}

house = room {
    nam = 'У здания';
    entered = code [[ make_snapshot() ]];
-- ...
}
```

ru:gamedev:modules:dbg

Модуль dbg

Подключение

require “dbg”

Тип

игровой

Зависимости

input

Описание

Включает отладчик. Отладчик позволяет:

- переходить в разные локации;
- брать и выбрасывать предметы;
- выполнять lua код;
- делать дамп состояния объектов;

Примеры использования

После включения модуля в вашу игру, кликните на объект debug в инвентаре, или нажмите клавишу «F7».

ru:gamedev:modules:trigger

Модуль trigger

Подключение

require «trigger»

Тип

расширение кода

Зависимости

INSTEAD 1.7.0

Описание

Модуль можно взять здесь:

http://instead.googlecode.com/svn/trunk/doc/examples/trigger.lua

Триггеры позволяют выполнить некое событие по условию. При этом, событие срабатывает один раз. Триггеры реализованы как надстройка над life, и поэтому обрабатываются после каждого действия игрока.

Для создания триггера используйте:

```
<идентификатор> = trigger(<действие> [,условие])
```

Где ‘действие’ - строка вывода или функция, а ‘условие’ – функция или строка, которая будет вычисляться как условие. Если условие не задано, триггер сработает сразу.

Для активации триггера используйте:

```
идентификатор:on([приоритет]) -- более высоким
-- значениям приоритета соответствуют меньшие
-- числовые значения (1 -- самый высокий)
```

Для деактивации:

```
идентификатор:off()
```

Чтобы узнать состояние триггера, используйте:

```
идентификатор:state() -- в случае если триггер активен, будет возвращено true
```

Примеры использования

```
life_checker = trigger(code [[ walk 'badend' ]],
        [[ pl._life <= 0 ]]):on(1)
```

```
hello_dlg = trigger([[- Привет! Как дела? - спросил меня Макс.]],
             [[ visited(max_dialog) ]])
hello_dlg:on()
```

Так как триггер удаляется из списка life объектов сразу после срабатывания, безопасно писать конструкции вида:

```
d = dlg {
    nam = "Разговор с Александром";
    entered = function(s)
        trigger "Привет! Хорошо, что зашел!":on()
    end;
--    ...
```

Так как такой безымянный триггер сработает в этом же игровом такте, и будет тут же удален. Этот прием удобно использовать в диалогах.

ru:gamedev:modules:keyboard

Модуль keyboard

Подключение

require «keyboard»

Тип

игровой/расширение кода

Зависимости

INSTEAD 1.7.0

Описание

Модуль для создания полей ввода. Ввод может осуществляться как с клавиатуры, так и с помощью ссылок. Модуль находится в каталоге doc/examples в исходном коде INSTEAD: http://instead.googlecode.com/svn/trunk/doc/examples/keyboard/

Примеры использования

```
instead_version "1.7.0"
require "keyboard"
require "xact"
```

```
input.verbose = true
main = room {
    nam = '?';
    dsc = function(s)
        if read.text ~= '' then
            p "Привет, "
            p (read.text,"!")
        else
            p [[Как вас {xwalk(read)|зовут}?]];
        end
    end
}

read = keyboard {
    nam = 'Имя:';
    msg = "Поле ввода:";
}
```

ru:gamedev:modules:cutscene

Модуль cutscene

Подключение

`require "cutscene"`

Тип

расширение кода, для создания эффектов

Зависимости

INSTEAD не ниже 1.8.0

Модуль xact

Модуль timer

На данный момент *(2012/12/29 00:21):* модуль все еще находится в тестовом режиме и не включен в базу, для использования в разработке модуль можно загрузить отсюда.

Описание

Для вызова в коде игры через `require «cutscene»` файл `cutscene.lua`, находящийся в каталоге `instead/doc/examples/` должен находится в каталоге с файлами игры (там где расположен `main.lua`). Зависимости подключать необязательно, в том случае, если данные

модули не используются в вашей игре, они подключаются автоматически.
В случае когда модуль `cutscene` расположен в каком-либо подкаталоге, то необходимо указать этот каталог при вызове, например:

```
-- $Name: Моя игра$
-- $Version: 0.1$
-- $Author: Я$

instead_version = "1.8.0"

require "./lib/cutscene"
...
```

Поддерживаются следующие теги:

`{pause}`	задержка (время по умолчанию)
`{pause <n>}`	где n - указывается число (время в миллисекундах)
`{cls}`	очистить вывод
`{cut}`	ждать нажатия от пользователя (выводит указатель >>>)
`{cut <what>}`	вывести надпись на ссылке-указателе
`{walk <what>}`	переход в указанную комнату
`{code <what>}`	выполнить определенный код
`{pic <what>}`	вывести в графическую область картинку, путь к файлу указывается без спецсимволов
`{fading}`	вывод с эффектом перехода (выводится то, что попало в вывод раньше)
`{fading <n>}`	где n - число шагов перехода от 0 - 255

Примеры использования

```
instead_version "1.8.2"
require "cutscene"
require "fonts"


function init()
```

```
    s1 = font('georgia.ttf', 30);
    s2 = font('georgia.ttf', 15);
end

main = cutscene {
    nam = true;
    dsc = function(s)
        pn (txtc(s1:txt "INSTEAD"))
        pn "{fading}"
        pn (txtc(s2:txt "http://instead.syscall.ru"))
        pn "{code print 'a'}"
        pn "{fading}"
        pn ("{cut}{walk r2}")
    end;
}

r2 = cutscene {
    entered = function(s)
        pn (txtc(s1:txt "The End!"))
    end;
    nam = 'end';
    way =  {'r2'};
}
```

ru:gamedev:modules:fonts

Модуль fonts

Подключение

`require «fonts»`

Тип

расширение кода, оформление

Зависимости

INSTEAD не ниже 1.8.0

Модуль sprites

Модуль theme

На данный момент *(2012/12/29 00:17):* модуль все еще находится в тестовом режиме и не включен в базу, для использования в разработке модуль можно загрузить отсюда.

Описание

Для вызова в коде игры через `require «fonts»` файл `fonts.lua`, находящийся в каталоге `instead/doc/examples/` должен находится в каталоге с файлами игры (где расположен `main.lua`). Зависимости подключать необязательно, в том случае, если данные модули не используются в вашей игре, они подключаются автоматически.
Все внешние файлы шрифтов, используемые в игре, всегда более правильно расположить в подкаталоге `$ваша игра/fonts`, для верного отображения обработчиком INSTEAD на других операционных системах, где не установлены или отсутствуют такие шрифты и вызывать в коде именно оттуда.
В случае когда модуль `fonts` расположен в каком-либо подкаталоге, то необходимо указать этот каталог при вызове, например:

```
-- $Name: Моя игра$
-- $Version: 0.1$
-- $Author: Я$

instead_version = "1.8.0"

require "./lib/fonts"
...
```

Примеры использования

```
instead_version "1.8.0"
require "fonts"

function init()
    s1 = font('georgia.ttf', 30);
    s2 = font('georgia.ttf', 12);
end

main = room {
    nam = "Демонстрация шрифтов.";
    dsc = function(s)
        pn "Пример использования модуля:"
        pn (s1:txt "Привет, мир!");
        pn (s1:txt ("Привет, мир!", 'gray', 2));
        pn (s2:txt "Мелкий шрифт...");
        pn "Обычный шрифт"
```

```
    end;
    way = { 'm2' };
}

m2 = room {
    nam = 'Переход';
    dsc = function()
        pn (s1:txt "И снова шрифты!");
    end;
    way = { 'main' };
}
```