

Модули INSTEAD

04.06.17

Оглавление

Модуль click	5
Описание	5
Примеры использования	6
Модуль format	7
Описание	7
Примеры использования	8
Модуль hideinv	10
Описание	10
Примеры использования	10
Модуль kbd	14
Описание	14
Примеры использования	16
Модуль prefs	17
Описание	17
Примеры использования	18
Модуль timer	18
Описание	19
Примеры использования	19

Модуль хаст	20
Описание	20
Примеры использования	20
Модуль sprites	22
Описание	22
sprite.load(file_name)	22
sprite.box(w,h,[color[,alpha]])	22
sprite.blank(w,h)	22
sprite.free(spr)	22
sprite.screen()	22
sprite.font_scaled_size(size)	22
sprite.font(font_path, size)	23
sprite.free_font(font)	23
sprite.font_height(font)	23
sprite.alpha(spr, alpha)	23
sprite.dup(spr)	23
sprite.scale(spr, xs, ys, [smooth])	23
sprite.rotate(spr, angle, [smooth])	23
sprite.text(font, text, col, [style])	23
sprite.size(spr)	23
sprite.text_size(font, text)	23
sprite.draw(src_spr, fx, fy, fw, fh, dst_spr, x, y, [alpha])	24
sprite.draw(src_spr, dst_spr, x, y, [alpha])	24
sprite.copy(src_spr, fx, fy, fw, fh, dst_spr, x, y, [alpha])	24
sprite.compose(src_spr, fx, fy, fw, fh, dst_spr, x, y, [alpha])	24
sprite.copy(src_spr, dst_spr, x, y, [alpha])	24
sprite.fill(spr, x, y, [w, h, [col]])	24
sprite.pixel(spr, x, y, col, [alpha])	24
sprite.pixel(spr, x, y)	24
sprite.colorkey(spr, color)	24
Примеры использования	25
Общие рекомендации	25

Модуль sound	29
Описание	29
Примеры использования	30
Модуль pouse	30
Описание	30
Примеры использования	30
Модуль counters	32
Описание	32
Примеры использования	33
Модуль wroom	35
Описание	35
Примеры использования	36
Модуль nolife	37
Описание	37
Примеры использования	37
Модуль proxumeni	38
Описание	38
Примеры использования	40
Модуль dash	43
Описание	44
Примеры использования	44
Модуль hotkeys	44
Описание	44
Примеры использования	44
Модуль para	44
Описание	45
Примеры использования	45

Модуль quotes	46
Описание	46
Примеры использования	47
Модуль theme	47
Описание	47
Примеры использования	50
Модуль snapshots	50
Описание	50
Примеры использования	51
Модуль dbg	52
Описание	52
Примеры использования	52
Модуль trigger	52
Описание	52
Примеры использования	53
Модуль keyboard	54
Описание	54
Примеры использования	54
Модуль cutscene	54
Описание	55
Примеры использования	56
Модуль fonts	57
Описание	57
Примеры использования	58

Модуль click

Подключение	require "click"
Тип	расширение кода
Зависимости	нет

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"

-- вызываем модуль
require "click"

-- определяем функцию game.click
game.click = function(s, x, y)
    -- вывод сообщения при клике с указанием координат
    p ("Клик упал по координатам: ", x, ", ", y);
end

main = room {    nam = 'лес',
    -- жесткая фиксация статической части описания комнаты
    forcedsc = true,
    -- вставка изображения
    pic = 'house.png',
    -- описание комнаты
    dsc = [[ Вы вышли на поляну где стоит домик.
              Вы видите, что дверь открыта.
              Нужно скорее попасть внутрь,
              где-то по лесу бродит медведь. ]],
    -- вызов модуля с определенными параметрами, определяем в виде функции
    click = function(s, x, y)
```

```

    -- проверяем условие попадания курсором в область двери домика
    if x > 80 and x < 200 and y > 225 and y < 325 then
        -- переходим в следующую комнату
        walk('house');
    else
        -- если условие не выполняется сообщаем об этом
        return 'Это не дом и тем более не дверь.';
    end;
end,
};

house = room { nam = 'Дом',
    -- так же фиксируем описание сцены, чтобы при клике описание не исчезало
    forcedsc = true,
    -- вставка изображения
    pic = 'door.png',
    -- описание комнаты
    dsc = [[ Вы сидите у себя в уютном домике.
        Перед собой вы видите дверь
        на улицу. Медведь остался голодным.
        Тест успешно пройден. ]],
};

```

Модуль format

Подключение	require "format"
Тип	расширение кода
Зависимости	нет

Описание

Модуль format выполняет форматирование вывода. По умолчанию все настройки выключены.

Основные параметры модуля:

```

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

```

Вы так же можете пользоваться модулями **para**, **dash**, **quotes** для включения отдельных настроек.

Примеры использования

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

instead_version "1.8.2";

game.codepage = "UTF-8";

-- подключаем модуль
require "format"

main = room { nam = "Главная комната",
  -- описание содержащее текст который мы сейчас будем форматировать
  dsc = txtc('Вас приветствует тест модуля '..txtb('format')).."! ^^"..
    [[ Вы находитесь в главной тестовой комнате.^
      Давайте посмотрим на что способен этот прекрасный модуль.^
      При нажатии ссылок ниже, весь текст в комнате -- будет
      менять свой вид. Чтобы все выглядело более наглядно,
      добавим пару абзацев всем излюбленного текста для дизайнеров --
      "lorem ipsum".^^
      "Neque porro quisquam est qui dolorem ipsum quia dolor sit amet,
      consectetur, adipisci velit..."^
      "Нет никого, кто любил бы боль саму по себе, кто искал бы
      её и кто хотел бы иметь её просто потому, что это боль.."^^
      Lorem ipsum dolor sit amet, consectetur adipiscing elit.
      Nulla vitae erat sapien, vitae pellentesque est.
      Nulla facilisi. Nullam posuere posuere facilisis.
      Morbi suscipit lacus et. ]],
  -- для работы с облегченными объектами определенные ниже -- vobj,
  -- в виде которых исполнены ссылки-модификаторы используем функцию act
  act = function(s, w)
    -- с помощью условий определяем какую из ссылок использовали
    if w == 'para' and not format.para then
      -- выводим сообщение о том что произошло
      p "Был переключен режим отображения отступов в начале абзаца";
      -- включаем соответствующий модификатор форматирования
```



```

        format.para = true;
    elseif w == 'dash' and not format.dash then
        -- выводим сообщение
        p "Был переключен режим отображения двойного тире"
        -- включаем модификатор
        format.dash = true;
    elseif w == 'quot' and not format.quotes then
        p "Был переключен режим отображения кавычек"
        format.quotes = true;
    elseif w == 'filt' and format.filter == nil then
        p "Был добавлен пользовательский фильтр, перевода строки"
        -- добавляем фильтр для оформления, допустим, пустую строку
        -- оформим все в виде функции
        format.filter = function(s)
            -- эскейп-последовательностью '\n' добавляем в начало
            -- каждого описания dsc перевод на новую строку,
            -- выделим для наглядности строку двумя знаками плюс ++
            return '++\n'..s
        end;
        -- ошибка какого либо из условий сбрасывает все параметры
    else
        -- вывод сообщения
        p "Произошел сброс всех режимов";
        -- сброс
        format.para = false;
        format.dash = false;
        format.quotes = false;
        format.filter = nil;
    end;
    -- следующей строкой обновляем статичное описание комнаты
    return stead.need_scene();
end,
-- раздел объектов комнаты
obj = {
    -- облегченные объекты, которые служат ссылками-модификаторам
    vobj('para', "{Параграф}^"),
    vobj('dash', "{Тире}^"),
    vobj('quot', "{Кавычки}^"),
    vobj('filt', "{Фильтр}^"),
},
};

```

Модуль hideinv

Подключение	require "hideinv"
Тип	расширение кода
Зависимости	нет

Описание

Модуль `hideinv` позволяет временно прятать объекты в инвентаре для выбранных комнат, при переходе в другую комнату не содержащую кода: `hideinv = true`; инвентарь вновь активируется.

Важно!!!

Если ваша игра использует **модуль хаст** и вы хотите использовать свойства данного модуля в комнатах типа `хroom`, укажите в коде подключение модуля `hideinv` раньше модуля `хаст`.

Примеры использования

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

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

instead_version "1.8.0"

-- подключаем модуль
require "hideinv"

-- функцией инициализации добавляем в инвентарь несколько объектов
function init ()
    inv():add('wrench');
    inv():add('cube');
end;

main = room {    nam = "Песочница",
    -- закрепляем статичную часть описания комнаты
    forcedsc = true;
    -- выводим описание по проверке условия посещения диалога
    dsc = function(s)
```

```

    p [[ Вы находитесь в песочнице. ]];
    if visited(robot_talk) == nil then
        p [[ ^^Для просмотра возможностей модуля
            работы с инвентарем проведем небольшой тест.
            ^После осмотра объектов, поговорите с роботом,
            инвентарь на время диалога должен исчезнуть. ]];
    else
        p [[ ^^Вы поговорили с роботом.
            Попробуйте теперь взаимодействовать с ним объектами,
            ведь инвентарь на месте! ]];
    end;
end,
-- раздел объектов в комнате
obj = {
    'robot',
},
-- переходы из комнаты
way = {
    'robot_talk',
},
};

wrench = obj { nam = "Гаечный ключ",
    -- вывод при осмотре объектов в инвентаре
    inv = [[ Регулируемый гаечный ключ. ]];
    -- функция для взаимодействия объекта с другими объектами
    use = function(s, w)
        -- проверка условий для выбора вида взаимодействия
        if w == robot and have(cube) then
            if visited(robot_talk) == nil then
                p [[ Так не пойдет сначала, надо поговорить с роботом. ]];
                return false;
            else
                p [[ Взяв покрепче гаечный ключ, с неистовой скоростью и
                    остервенением вы набросились и разобрали робота, ломая запчасти
                    и детали, разбросав его всего на болтики и гаечки. ]];
                -- при переходе в следующую комнату инвентарь должен исчезнуть
                walk(badend);
            end;
        elseif not have(cube) then
            p [[ Провозившись несколько часов, аккуратно разбирая деталь, за
                деталью, вы наконец разобрали робота. Теперь можно заняться
                техобслуживанием старого друга. ]];
        end;
    end;
end,

```

```

        -- при переходе в следующую комнату инвентарь должен исчезнуть
        walk(happyend);
    else
        p [[ Ничего не получится. ]];
        return false;
    end;
end,
];

cube = obj {    nam = "Голокуб",
    inv = [[ Современный голокуб с данными. ]];
    use = function(s, w)
        if w == robot then
            if visited(robot_talk) == nil then
                p [[ Робот изучающе смотрит на вас и по взгляду его сенсорных
                    окуляров вы понимаете, что он хочет что-то сказать, но ждет от
                    вас команды. ]];
                return false;
            else
                p [[ Вы протянули роботу голокуб, взяв его манипулятором, он
                    погрузил его в слот, на своем корпусе и произнес: "Начинаю
                    трансляцию".^ Перед вам спроецировалось изображение
                    документации, трехмерные модели, описания и различные советы как
                    правильно разобрать робота. Изучив ее теперь вы понимаете, что и
                    как нужно разбирать в корпусе, чтобы ничего не сломать. ]];
                inv():del(cube);
            end;
        else
            p [[ Ничего не получится. ]];
            return false;
        end;
    end,
];

robot = obj {    nam = "Робот",
    -- описание объекта
    dsc = [[ Недалеко от вас стоит {робот}. ]],
    -- простое взаимодействие с объектом
    act = [[ Обыкновенный робот-говорун, весь из хромированного металла. ]],
];

robot_talk = dlg {    nam = "Разговор с роботом",
    -- с помощью модуля сделаем инвентарь невидимым

```

```

hideinv = true;
-- описание сцены
dsc = [[ Подойдя к роботу поближе, вы произносите: "Инициализация!". Робот,
        казалось ожил и из его громкоговорителя механическим
        голосом донеслось: "Приветствую, тебя, хозяин!" ]];
-- раздел диалога
phr = {
    { always = true, "Гибернация!",
      [[ - Гибернация, - прозносите вы, и робот тут же засыпает,
          только диодные огоньки в его оптосенсорах
          продолжают наблюдать за вами. ]],
      code [[ back() ]]]];
    {"Кто ты такой?",
      [[ - Кто ты такой? - спрашиваете, вы у робота.^
          - Я робот-говорун, произведен в 2012 году, -- отвечает робот.]],
      [[ ron(3) ]]]];
    {3, false, "Что ты умеешь делать?",
      [[ - Что ты умеешь делать?!^
          - Я умею много говорить и давать вредные советы, а так же,
            у меня есть устройство для воспроизведения информации. ]],
      [[ ron(4) ]]]];
    {4, false, "Почему я не вижу свой инвентарь?",
      [[ - Почему пропал мой инвентарь? - удивляетесь вы.^
          - Очевидно, что вы находитесь в диалоге со мной,
            зачем вам инвентарь? - рассудительно отвечает робот.]],
      [[ ron(5) ]]]];
    {5, false, "Ты можешь включить мой инвентарь?",
      [[ - Ты можешь активировать мой инвентарь?
          - задаете вы очередной вопрос роботу.^
          - Это конечно не входит в мою компетенцию, проверять чужие
            карманы, но что не сделаешь для лучшего друга?!
          - робот прекратил разговор и ваш инвентарь снова на месте. ]],
      [[ back() ]]]];
  },
};

happyend = room {   nam = 'Конец';
  -- прячем инвентарь
  hideinv = true;
  dsc = [[ Поздравляем, Вы успешно прошли тест! ]];
};

badend = room {     nam = "Конец",

```

```

-- прячем инвентарь
hideinv = true;
dsc = [[ Вы стоите в песочнице, держа в руках гаечный ключ, над обломками
        бесславно почившего робота. Собрать его теперь уже не удастся.^
        Тест пройден.]];
};

```

Модуль kbd

Подключение

```
require "kbd"
```

Тип

игровой

стандартная библиотека

Зависимости

нет

Описание

Внимание!!!

Если вы хотите организовать ввод текста с клавиатуры, используйте [модуль keyboard](#).

Модуль позволяет удобным способом обрабатывать события срабатывания 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}

Идентификатор	Клавиша	Идентификатор	Клавиша
...	...	[+]	+ 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

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

instead_version "1.8.0"

-- подключаем модуль
require "kbd"

-- определяем функцию, которая при инициализации игры назначит список клавиш,
-- которые будет отслеживать модуль
function init()
    -- собственно список всех отслеживаемых клавиш
    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';
    -- описание сцены
    dsc = [[Нажимайте клавиши.]];
    -- функция обработки срабатывания клавиш
    kbd = function(s, down, key)
        -- выводим сообщение
        p [[Событие от клавиши:]];
        -- выводим литеру\название той клавиши, которая была использована
        p (key);
        -- проверяем условие нажатия клавиши
        if down then
            -- выводим сообщение
            p [[нажата]];
        else
            -- выводим сообщение при невыполнении условия
            p [[отжата]];
        end
        -- выводим пустую строку
        pn "";
    end
end
};
```


Модуль prefs

Подключение

require "prefs"

Тип

расширение кода

стандартная библиотека

Зависимости

нет

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.

По своей сути 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

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

instead_version "1.8.0"

-- подключаем модуль click
require "click"
-- подключаем модуль prefs
require "prefs"

-- устанавливаем счетчик в нулевое значение
prefs.counter = 0;

-- определяем функцию отслеживания количества "кликов"
game.click = function(s)
    -- увеличиваем счетчик
    prefs.counter = prefs.counter + 1;
    -- сохраняем счетчик
    prefs:store();
    -- выводим сообщение
    p("На данный момент сделано ", prefs.counter , " кликов");
end;

-- добавляем изображение, по которому можно производить клики
game.pic = 'clickme.png';

main = room {    nam = "Комната кликов",
    -- делаем фиксацию статичной части описания
    forcedsc = true,
    -- добавляем описание для сцены
    dsc = [[ Этот тест был написан специально
               для проверки работы модуля <<prefs>>.
               ]];
};
```

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 обновляется. В противном случае, возвращаемое значение интерпретируется как реакция.

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

Напомним, что таймер программируется с помощью объекта `timer`:

- `timer:set(мс)` – задать интервал таймера в миллисекундах;
- `timer:stop()` – остановить таймер;

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;
end;
```

```

nam = 'Проверка таймера';
dsc = [[Ждите.]];
}

```

Модуль хаст

Подключение	require "хаст"
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... ]];
}

```

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-прямоугольника, закрашенного заданным цветом (строка типа: «green» или «#cc4242»).

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 прямоугольника *fw-на-fh* из спрайта *src_spr* в спрайт *dst_spr* по координатам *[x,y]* (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 четыре компонента цвета).

sprite.colorkey(spr, color)

Начиная с версии 2.1.0

Задаёт в спрайте цвет, который выступает в роли прозрачного фона. При этом, при последующем выполнении операции `sprite.copy`, из рассматриваемого спрайта будут скопированы только те пиксели, цвет которых не совпадает с цветом прозрачного фона.

Для отмены цвета прозрачности, используйте вызов функции без задания параметра color:
`sprite.colorkey(spr)`

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 пишете 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
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
```

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. Начиная с версии 2.1.0, вторым параметром можно задавать время затухания звука в мс. при его приглушении;

`sound.playing([channel])` – узнать проигрывается ли звук на любом канале или на выбранном канале; если выбран конкретный канал, функция вернет хангл 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
```

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' };
}

```

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";

-- подключаем модуль
require "counters"

-- добавляем объект яблоко
apple = obj {   nam = 'яблоко',
    -- описание выводимое в сцене
    dsc = 'На столе лежит {яблоко}.',
    -- определяем функцию для того чтобы можно было взять яблоко
    tak = function(s)
        -- выводим счетчик
        pn ('Текущий счетчик ТАК для яблока -- ', act_count(s));
        -- выводим надпись, о том что ГГ взял яблоко
        p 'Я взял яблоко.';
    end;
}
```

```

end,
-- определяем функцию для осмотра яблока в инвентаре
inv = function(s)
    -- выводим счетчик инвентаря для яблока
    pn ('Текущий счетчик INV для яблока -- ', act_count(s));
    -- выводим реакцию на осмотр в инвентаре
    p 'Красное!'
end,
-- определяем функцию для взаимодействия
use = function(s, w)
    -- выводим счетчик использования
    pn ('Текущий счетчик USE для яблока -- ', use_count(s));
    -- определим условие по которому яблоко будет взаимодействовать с
    -- другими объектами, сначала проверяем счетчик
    if use_count(s) < 3 then
        -- проверяем объект
        if w == table then
            -- выводим сообщение
            p 'Я положил яблоко на стол.';
            -- ложим яблоко на стол
            drop(s, w);
        else
            -- выводим сообщение при невыполнении условий
            pn 'Ничего не понимаю.';
            return false;
        end;
    else
        -- выводим сообщение, что счетчик больше чем проверяется в условии
        pn ('Счетчик USE > 3. Уже ничего не получится.');
```

```

    if act_count(s) == 1 then
        -- выводим сообщение
        p 'На столе яблоко!';
        -- включаем объект яблоко
        apple:enable();
    else
        -- выводим сообщение если условие не выполняется
        p 'Стол как стол.';
    end;
end,
-- раздел объектов, которые находятся в объекте стол
obj = { 'apple' },
};

main = room {    nam = 'комната',
    -- раздел объектов в комнате
    obj = { 'table' },
};

```

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

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

instead_version "1.8.0"

-- подключаем модуль
require "wroom"

main = room {
  nam = 'Комната 1';
  -- описание сцены, оформляем как функцию
  dsc = function(s)
    -- проверяем условие из какой комнаты пришел игрок
    if from() == r2 then
      -- выводим сообщение
      p [[ Вы пришли из комнаты "В неведомое". ]];
    else
      -- выводим сообщение при не выполнении условия
      p [[ Вы находитесь в главной комнате. ]];
    end;
  end,
  -- раздел переходов из комнаты
  way = {
    -- используем модуль, для обозначения комнаты <r2>
    wroom('В неведомое', 'В комнату r2', 'r2')
  };
};

r2 = room {
  nam = 'Комната 2';
  -- описание сцены
  dsc = [[ Вы зашли из главной комнаты "Комната 1". ]];
  -- раздел переходов
```

```

    way = {
      -- можно использовать формат записи такой же как и wroom
      wroom('Назад', main)
    };
  };

global { somewhere = r2 };

r3 = room {
  nam = 'Комната 3';
  way = {
    wroom('Идти куда-то', function() return somewhere end)
    -- цель перехода возвращает функция
  };
};

```

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 атрибут nolife, например:

```

require "hideinv"
require "nolife"

happyend = room {
  nam = 'Конец';
  hideinv = true;
}

```

```
nolife = true;
dsc = [[ Вы прошли игру! ]];
}
```

Модуль proхumenu

Подключение	require «proхumenu»
Тип	игровой/расширение кода
Зависимости	INSTEAD 1.7.0

Описание

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

При использовании proхumenu, 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 `proхumenu` 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 следующая 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 «ИСПОЛЬЗОВАТЬ», курсор перейдет в режим использования, в котором будет ожидать 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 },
}

```

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!!!]];
}
```

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.

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

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

instead_version "1.8.0"

-- подключаем модуль
require "para"

-- переопределяем параметр по умолчанию для оформления параграфа
format.para = "      "

main = room {  nam = "Lorem Ipsum",
  -- добавляем текст, который будет подвергнут форматированию
  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

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

instead_version "1.8.0"

-- подключаем модуль
require "quotes"

main = room {    nam = "Lorem Ipsum",
    -- описание сцены с различными видами кавычек
    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.

Имейте в виду, что изменение параметров темы возможно только для тех игр, которые содержат собственную тему или темы. При этом, сохранение изменений темы в файле сохранения поддерживается только в том случае, если используется единственный файл theme.ini в каталоге с игрой. В случае множественных игровых тем, автор игры должен сам восстановить параметры темы в функции start(), как это делается при работе с модулем спрайтов.

Для изменения параметров действующей темы, 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();  
-- ...
```

Существует функция, для того, чтобы узнать текущую выбранную тему.

```
theme.name()
```

Функция возвращает строку – имя каталога темы. Если игра использует собственный файл `theme.ini`, функция вернет точку. Это удобно, для определения того, включен ли механизм собственных тем игр:

```
if theme.name() ~= '.' then  
    error "Please, enable own theme mode in menu!"  
end
```

Если в игре используется механизм множественных тем, то имя темы начинается с точки, например:

```

if theme.name() ~= '.default' then
    -- наша встроенная тема default
elseif theme.name() ~= 'default' then
    -- стандартная тема default в INSTEAD
end

```

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'

```

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 снимка используйте функцию: `make_snapshot()`. В качестве параметра может быть задан номер слота.

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

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

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

Важно! При работе со снапшотами, файлы игры дозагружаются, предварительно убив все объекты — но не хаки типа:

```
old_vobj = vobj
```

В таком случае после восстановления снапшота программа провалится в бесконечный цикл. Чтобы избежать:

```
if not old_vobj then
  old_vobj = vobj
  ...
end
```

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() ]];
  -- ...
}
```

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».

Модуль trigger

Подключение	require «trigger»
Тип	расширение кода
Зависимости	INSTEAD 1.7.0

Описание

Модуль можно взять здесь:

<http://github.com/instead-hub/instead/raw/master/doc/examples/stead2/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.

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://github.com/instead-hub/instead/tree/master/doc/examples/stead2/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
end

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

Модуль cutscene

Подключение

```
require "cutscene"
```

Тип

расширение кода, для создания эффектов

Зависимости

INSTEAD не ниже 1.8.0

[Модуль хаст](#)

[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 находиться 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"
...
```

Модуль имеет две опции, которые могут оказаться полезны при написании длинных катсцен:

```
-- включить автоматическую прокрутку текста к новому сообщению катсцены,
-- если оно вышло за пределы окна
stead.cut_scroll = true
-- добавить разделитель, отделяющий последнее сообщение от предыдущего текста катсцены
stead.cut_delim = '^ *** ^'
```

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 {
    enter = function(s)
        pn (txtc(s1:txt "The End!"))
    end;
}
```



```
end;  
nam = 'end';  
way = {'r2'};  
}
```

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

Чтобы модуль можно было подключить строкой `require «fonts»`, поместите файл `fonts.lua` в каталог с файлами игры (тот, где расположен `main.lua`). Если же вы располагаете модуль 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"  
...
```

Найти модуль можно в папке `instead/doc/examples/`, либо скачать последнюю версию по ссылке выше. Зависимости подключать необязательно: если модули *sprites* или *theme* не

используются в вашей игре, то они подключатся автоматически.

Все внешние файлы шрифтов, используемые в игре, рекомендуется расположить в подкаталоге \$ваша_игра/fonts. Помните, что INSTEAD может быть запущен на разных операционных системах, а там такие шрифты могут отсутствовать в стандартной поставке. Если вы следуете данному совету, то при подключении шрифта следует указывать эту папку в пути: *fonts/имя_шрифта*

Примеры использования

```
instead_version "1.8.0"
require "fonts"

function init()
    s1 = font('fonts/georgia.ttf', 30); -- шрифт, который будет
    -- масштабироваться в соответствии с настройкой
    s2 = font('fonts/georgia.ttf', 12, false); -- шрифт, который
    -- не будет масштабироваться
end

main = room {
    nam = "Демонстрация шрифтов.";
    dsc = function(s)
        pn "Пример использования модуля:"
        pn (s1:txt "Привет, мир!");
    -- стили:
    -- 1: жирный. 2: курсив, 4: подчеркивание, 8: зачеркнуто
    -- комбинации:
    -- жирный курсив: 3 (1 + 2)
    -- подчеркнутый курсив: 6 (2 + 4)
    -- и т.д.
    -- при подчеркивании используйте неразрывные
    -- пробелы, иначе подчеркнутыми будут только слова

        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' };  
}
```

Важно!

Имейте ввиду, что метод *txt* возвращает объект *img*, а не настоящий текст - потому вы все равно можете использовать его в качестве ссылки, но подсвечиваться при наведении она не будет.