Команды сценария

- rollcredits
- sequence и trigger
- unlock_ending
- with

Введение

В этой главе будут описаны команды, которые можно использовать внутри игровых сценариев.

Синтаксис команд во многом схож с синтаксисом Ren'Py, однако важно помнить, что на визуальных все сходства и заканчиваются. Love, Money, Rock'n'Roll использует собственный движок, написанный на Unity, поэтому многих возможностей, которые доступны в Ren'Py, в игре попросту нет.

Большинство команд может принимать различные параметры (аргументы), которые делятся на три типа:

- Обычные
- Ключевые
- Флаги

Обычные — параметры, которые являются каким-либо значением.

Ключевые — параметры, которые начинаются с определенного ключевого слова.

Рассмотрим на примере команды, выводящей backdrop-фон:
`backdrop bg el duration 10`

Здесь `bg` и `duration` — названия ключевых параметров, а `el` и `10` — их значения.

Ключевые параметры могут располагаться в каком угодно порядке, однако они должны быть прописаны после обычных параметров.

Флаги — флаги представляют собой какое-либо ключевое слово, которое не принимает никаких параметров. Обычно включает или отключает какое-либо действие. Как и ключевые параметры, должны прописываться после обычных (за редкими исключениями).

Условные обозначения

Далее при описании синтаксиса команд будут применяться следующие обозначения:

- `(название)` — опциональный параметр;
- `[название]` — обязательный параметр;
- `{название}` — ключевой параметр;
- `|название|` — флаг;

Комментарии

Внутри сценария можно оставлять комментарии — они не будут никак интерпретироваться игрой.

Каждый комментарий должен начинаться с символа `#`, стоящего в начале строки:

```
# это комментарий
scene bg akihibara_night with dissolve
# выше команда для вывода фона, а эта строчка — тоже комментарий
```

Комментарии должны располагаться на отдельных строках.

Лейблы

Определение лейблов

Лейблы — это своеобразные маркеры внутри сценария, по которым может осуществляться переход.

Лейблы определяются при помощи команды `label`:

```
label [name]
```

Здесь параметр `name` — это имя лейбла, которое может содержать латинские буквы, цифры и нижнее подчёркивание.

Все команды, идущие после команды `label`, будут находится внутри данного лейбла — до следующего лейбла, если такой есть.

> Лейблы не могут определяться внутри других команд и условий.

Пример определения лейблов `day1_1` и `day1_2`:

```
label day1_1
    # остальные команды


label day1_2
    # остальные команды
```

У каждого сценария есть лейбл `init`, который автоматически ставится игрой в самом начале. Т.к., это имя зарезервировано игрой, его нельзя использовать в качестве значения для параметра `name`.

Также лейблы используются в сохранениях — каждый раз, когда игра создаёт сохранение, помимо состояния текущей строки, она также сохраняет состояние первой строки внутри активного лейбла — подобное сохранение используется, если при загрузке сохранения в сценарии были обнаружены изменения и загрузить игру с сохранённой точки невозможно.

Поэтому, при возможности старайтесь расставлять лейблы почаще, — как минимум, в критически важных частях сценария.

goto

Осуществляет переход на заданный лейбл.

Синтаксис

```
goto [name]
```

Параметры

- `name` — название лейбла.
  Обязательный параметр.

Пример использования

```
# переход на лейбл day1_1
goto day1_1
```

Текстовое окно

Вывод текста

Синтаксис

```
(narratorID) [text]
```

Параметры

- `narratorID` — ID говорящего, опциональный параметр.
- `text` — текст, обязательный параметр. Указывается двойных или одинарных кавычках.

Примеры использования

```
"Hello, World!
```

```
hi "Hello, World!"
```

```
hi 'Hello, World!'
```

Тэги разметки

Текст поддерживает тэги разметки Rich Text, а также некоторые специальные тэги, такие как `<note>`, `<w>` и `<seq>`.

<note>

Специальный тэг, который можно использовать при выводе текста для обозначения сноски.

Синтаксис

```
<note=[id]>[caption]</note>
```

Параметры

- `id` — ID сноски.
- `caption` — текст, к которому нужно пояснение.

Пример использования

```
# делает слово "рамен" кликабельным и показывает сноску note3 при клике
hi "То есть заварной <note=note3>рамен</note>, по-твоему, лучше?"
```

Демонстрация

<w>

Тэг, используемый для расстановки пауз в тексте. Доходя до позиции тэга в тексте, игра приостанавливает вывод и ожидает действия от игрока для продолжения (клика мышкой, нажатия пробела, тапа по экрану).

Синтаксис

```
<w>
```

Пример использования

```
"Школа…<w> А там Кетрин."
```

Демонстрация

<seq>

Запускает определённую заранее последовательность команд.

Синтаксис

```
<seq [sequenceID]>
```

Параметры

- `sequenceID` — ID сиквенса, обязательный параметр.

Пример использования

```
scene bg kitchen_mc_day with dissolve
show ka

# определение сиквенса со сменной эмоции спрайта персонажа
sequence ka_surprise
{
  show ka surprise
}

"..."
# <seq ka_surprise> запускает определённый выше сиквенс
"За пару секунд на лице Кагомэ<seq ka_surprise> сменилась целая гамма эмоций."
```

Демонстрация

clear

Очищает текстовое окно.

Синтаксис

```
clear
```

mode

Задаёт режим текстового окна.

Синтаксис

```
mode [name]
```

Параметры

- `name` — название мода.
  Обязательный параметр.
  Может иметь следующие значения:
  `adv`
  `nvl`

Пример использования

```
# режим adv
mode adv

"Так выглядит текстовое окно в режиме adv"

# очистка текстового окна
clear

# режим nvl
mode nvl

"Так выглядит текстовое окно в режиме nvl"
```

Демонстрация

time

Меняет цвет оформления текстового окна.

Синтаксис

```
time [mode]
```

Параметры

- `mode` — название оформления.
  Возможные значение:
  `day` — дневное оформление;
  `sunset` — вечернее оформление;
  `night` — ночное оформление;

Пример использования

```
scene bg kitchen_mc_day with dissolve

# определение лейбла time_test
label time_test

time day
"Это дневное оформление окна."
time sunset
"Это вечернее оформление окна."
time night
"Это ночное оформление окна."

# скрытие окна
window hide

# nvl-окно
mode nvl
```

```
# переход в начало лейбла time_test
goto time_test
```

Демонстрация

window

Показывает или скрывает текстовое окно.

Синтаксис

```
window [state]
```

Параметры

- `state` — `show` или `hide`. Первое — показывает окно, второе — скрывает.

Пример использования

```
window hide
```

```
window show
```

> Текстовое окно автоматически скрывается перед выборами, а также показывается при выводе текста, если было скрыто.

Переменные

Синтаксис

Обращение к переменным происходит при помощи символа `$`, который пишется перед названием переменной.

Допустим, мы хотим изменить кол-во lovepoint'ов Химицу, которые хранятся в глобальной переменной `lp_hi`:

```
# прибавляет единицу к значению глобальной переменной lp_hi
$lp_hi += 1
```

Или проверить кол-во очков:

```
if ($lp_hi >= 10)
{
  hi "Условие выполнено!"
}
```

Также мы можем просто написать переменную отдельной строкой, в таком случае игра выведет её значение в текстовом окне:

```
$lp_hi
```

Переменные работают при помощи интерпретатора JavaScript, поэтому с ними можно проводить любые манипуляции, поддерживаемые JS (за исключением использования функций и присваиваний внутри текста).

Допустим, можно вывести значение переменной внутри текста при помощи интерполяции строки:

```
$lp_hi += 10
`У вас ${$lp_hi} очков!`
```

> Весь текст внутри файлов сценариев, за исключением текста внутри кавычек, приводится к нижнему регистру. Внутри интерполируемых строк регистр учитывается — используйте только нижний регистр, когда обращаетесь к переменным.

Виды переменных

Переменные в игре делятся на 3 типа:

- Глобальные переменные
- Persistent переменные (постоянные)
- Runtime переменные (временные)

Глобальные переменные

Глобальные переменные — переменные, определённые заранее в игре или в файле ресурсов мода. Эти переменные доступы в любой точке сценария и их значения сохраняются при переходе на следующий сценарий и в файлах сохранений.

Persistent переменные

Также, как и глобальные, эти переменные доступны в любой точке игрового сценария.

Отличие в том, что они не определяются нигде заранее и их значения доступны между разными модами.

Чтобы воспользоваться persistent-переменной, достаточно просто обратиться к ней, написав `$persistent.` перед её именем:

```
# задаёт значение 10 persistent-переменной some_value
$persistent.some_value = 10

if ($persistent.some_value == 10)
{
    "Условие выполнено"
}
```

Даже если переменная с таким названием ранее нигде не использовалась, она будет определена автоматически.

Runtime переменные

В отличии от глобальных и persistent переменных, данный тип переменных доступен только в пределах одного сценария, и их значения не сохраняются при переходе на следующий сценарий. Это своего рода вспомогательные переменные, которые могут использоваться для каких-то одноразовых проверок или действий.

Они определяются при помощи команды `var`:

```
var some_variable
```

При определении также можно указать значение переменной:

```
var some_variable = true
```

После определения, она будет доступна также, как и любая другая переменная:

```
var some_variable = true

# можно также записать как if ($some_variable)
if ($some_variable == true)
{
  "Условие выполнено!"
}
```

Условия

Для различных проверок в игре используются условия.

Поддерживаются следующие команды:

- `if`
- `else if`
- `else`

Примеры:

```
# lp_hi - глобальная переменная, определённая в игре. В ней хранятся лавпоинты (очки) Химицу.

if $lp_hi >= 10
{
  "У вас достаточно лавпоинтов Химицу!"
}
else if $lp_hi >= 5
{
  "У вас не хватает 5 лавпоинтов Химицу!"
}
else
{
  "У вас слишком мало лавпоинтов Химицу."
}
```

datetime

Это вспомогательная переменная, которая позволяет получить информацию о текущей дате на устройстве игрока.

> Для вызова этой переменной **не** нужно ставить символ `$`.

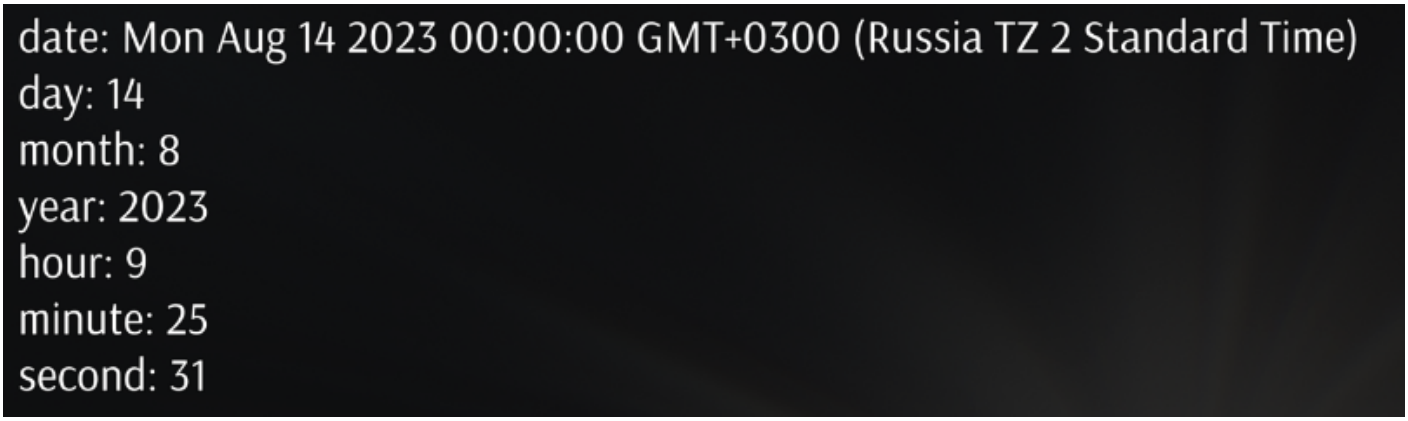
У неё есть следующие свойства:

- `date`
- `day`
- `month`
- `year`
- `hour`
- `minute`
- `second`

Доступ к любому из них можно получить при помощи `.` после названия переменной:

```
datetime.date
datetime.day
datetime.month
datetime.year
datetime.hour
datetime.minute
datetime.second
```

Пример вывода:



Фоны и персонажи

scene

Меняет фон, а также автоматически убирает все спрайты персонажей на экране.

Синтаксис

```
scene [type] [name] {with} {spritecolor} |skip|
```

Параметры

- `type` — тип фона.
  Возможные значения:
  `bg` — фон;
  `cg` — цг;
  `color` — цвет;

- `name` — название фона/цг (или цвета, если `type` = `color`; также может быть значением цвета).

- `with` — название перехода.

- `spritecolor` — оверлей-цвет для спрайтов. Может быть конкретным значением цвета или названием заранее определённого.

  Значение по умолчанию — название фона.

  Может также иметь два дополнительных значения:
  `none` — отключает оверлей-цвета у спрайтов;
  `default` — выставляет значение цветов спрайтов по умолчанию;

- `skip` — флаг. Если установлен, то игра не будет ожидать окончания выполнения команды.

Примеры использования

```
# выводит фон akihibara_night
scene bg akihibara_night

"..."

# выводит цг sakura с переходом dissolve
scene cg sakura with dissolve

"..."

# выводит синий фон с переходом fade
scene color blue with fade

"..."

# выводит фон school_corrdior с переходом slideout right и окрашивает спрайты персонажей в зелёный цвет
scene bg school_corridor spritecolor green with slideout right

"..."

# вывод спрайтов
```

```
show hi at center with dissolve
show ca at right with dissolve


"..."
```

Демонстрация

show

Показывает спрайт персонажа.

Синтаксис

Описание синтаксиса данной команды будет разделёно на две части.

В первой будет описан синтаксис основных параметров, во второй — синтаксис дополнительных параметров.
Оба синтаксиса одинаково применимы к обоим видов спрайтов (статичным и Live2D) с небольшими отличиями, о которых будет написано далее.

Основные параметры

```
show [charID или alias] ([poseInfo]) (animations)
```

Параметры

- `charID/alias` — ID или псевдоним персонажа. При первичном спавне модели всегда указывается ID персонажа. Если во время спавна вы присвоили персонажу псевдоним при помощи параметра `as`, то дальнейшие команды для данного персонажа должны использовать этот самый псевдоним (`alias`).
  Обязательный параметр.
- `poseInfo` — поза и скин персонажа, разделённый пробелом. Данный параметр является обязательным, если у спрайта не прописаны поза и скин по умолчанию.

  Также в качестве значения этого параметра может быть

использовано краткое название сочетания позы и скина (`shortname`), если оно определено.
- `animations` — список анимаций.
  Подробнее о том, как указываются анимации, будет написано в отдельной главе ниже.

Дополнительные параметры

```
{at} {size} ({layer} или {behind} или {ahead}) {spritecolor} {with} {as} (|skip| или |await|)
```

Параметры

- `at` — положение спрайта на экране. Значением может быть название положения или координаты, разделённые пробелом.
  При первичном спавне, если значение не указано, спрайт будет выведен в положении `center`.
- `size` — размер спрайта на экране. Значением может быть название размера или числовые значения, разделённые пробелом.
  При первичном спавне, если значение не указано, спрайту будет присвоен размер `normal`.
- `layer/behind/ahead` — определяет приоритетность отображения спрайта относительно других спрайтов на экране.

  `layer` может принимать до двух числовых значений, разделённых пробелом — номер слоя и позиция в слое. Чем выше значение — тем приоритетнее спрайт, т.е. он будет отображаться поверх других спрайтов, у которых номер слоя или позиция в нём меньше.

  Оба значения должны быть больше или равны нулю.
  Значения по умолчанию для обоих параметров при первичном спавне — `0`.

  `behind` и `ahead` спавнят спрайт за или перед каким-то

определённым спрайтом соответственно. Данные параметры принимают только имя персонажа, относительно которого нужно расположить спрайт на слое (они не меняют положение спрайта, только приоритетность отображения на экране).
- `spritecolor` — оверлей-цвет для спрайта. Если указан, то будет переопределять глобальные настройки оверлей-цветов, заданные командой `scene`. Аналогично параметру команды `scene`, значение может быть как названием цвета, так и конкретным цветом;
  `none` — отключает цвет у спрайта и игнорирует любые глобальные изменения цветов спрайтов;
  `default` — выставляет спрайту текущий активный глобальный цвет;
- `with` — название (и опционально параметры) перехода появления спрайта.
- `as` — выводит спрайт под псевдонимом, что позволяет вывести на экран несколько спрайтов одного и того же персонажа.

  Также копирует параметры отображения спрайта на экране, если в `charID`
  используется выведенный на экран персонаж (работает только для Live2D спрайтов).

- `skip/await` — по умолчанию, при первичном спавне спрайта на экране, игра будет ожидать завершения команды (т.е. пока спрайт полностью появится), все дальнейшие команды для спрайта (пока он на экране) игра ожидать не будет.

  При необходимости подобное поведение можно изменить при помощи этих двух флагов.

  `skip` пропускает ожидание выполнения, а `await` — наоборот — ждёт, пока команда будет выполнена.

Примеры использования

```
# выводит фон akihibara_night
scene bg akihibara_night

# спавн спрайта Химицу под псевдонимом test_sprite с переходом dissolve
show hi with dissolve as test_sprite

"..."

# перемещает спрайт в левую часть экрана и меняет ему размер
show test_sprite at left size close with dissolve

"..."
```

Демонстрация

```
# выводит фон akihibara_night
scene bg akihibara_night

# спавн спрайта Химицу
show hi at left size close with dissolve

"..."

# выводит спрайт Кетрин поверх спрайта Химицу
show ca at left size close ahead hi with dissolve

"..."

# выводит копию спрайта Кетрин под псевдонимом ca2 справа
show ca at right as ca2 with dissolve

"..."
```

```
# меняет спрайту ca2 скин
show ca2 ca_front dom with dissolve

# перемещает Химицу в центр экрана, а также меняет ей позу и скин
show hi himitsu_l plate at center with dissolve

"..."
```

Демонстрация

Анимации

У всех Live2D спрайтов, встроенных в игру, существуют следующие слои анимации:

- emotions (эмоции)
- head (голова)
- hands (руки)
- legs (ноги)
- additive (вспомогательные анимации)

> Указанный выше порядок действителен только для встроенных в игру спрайтов. Для Live2D спрайтов, определённых модом, порядок и названия слоёв определяются в `resources.yaml`

Все слои, кроме `additive`, принимают название одной конкретной анимации. В `additive` могут прописываться несколько анимаций одновременно, т.к. этот параметр объединяет все вспомогательные слои.

> Данное правило распространяется только на встроенные в игру спрайты.

Синтаксис

Анимации для Live2D-спрайтов могут прописываться двумя способами — кратким и полным.

Краткий

Краткий синтаксис выглядит так:

```
(emotions) (head) (hands) (legs) (additive)
```

Названия анимаций для конкретных слоёв прописываются друг за другом, при этом необязательно указывать анимации для всех слоёв — допустим, можно указать только название эмоции (emotions) и положение рук (hands), пропустив положение головы (head). При использовании данного синтаксиса необходимо строго соблюдать порядок слоёв.

Пример:

```
# выводит спрайт Химицу с эмоцией angry, положением головы head2left и положеним рук hands3
show hi angry head2left hands3
```

Полный

В полном же синтаксисе указываются названия слоя и анимации, разделённые двоеточием:

```
layer: animation layer: animation layer: animation
```

При этом порядок, в котором они указываются, неважен.

Пример:

```
# выводит спрайт Химицу с эмоцией angry, положением головы head2left и положеним рук hands3
# порядок, в котором прописываются анимации, неважен
show hi emotions: angry head: head2left hands: hands3
```

> При использовании полного синтаксиса, для вспомогательных анимаций также необходимо указывать названия слоёв.

Статичные спрайты

Статичные спрайты — это спрайты, которые собираются из изображений. У них нет анимации. Такие спрайты можно определить внутри файла `resources.yaml`, и они могут содержать различные части, которые могут меняться.

Для отображения или скрытия тех или иных частей спрайта используется синтаксис, аналогичный синтаксису анимаций, только вместо названия анимаций (и слоев) используются названия частей и их групп.

Например, мы определили в моде спрайт `mt`, в котором определена группа `uniform` — различные вариации одежды, и группа `face` — различные вариации эмоций:

```
# resources.yaml
---
  characters:
    mt:
      poses:
        front_1:
          parts:
            body: sprites/mt/mt_1_body.png # это постоянная часть, которую мы определили ранее
            ~face:
              default: normal # по умолчанию будет отображаться эмоция normal
              parts:
                normal: sprites/mt/mt_1_normal.png # нейтральное выражение лица
```

```
                smile: sprites/mt/mt_1_smile.png # улыбка
                sad: sprites/mt/mt_1_sad.png # грусть
            ~uniform: # группа uniform
              default: pioneer # по умолчанию при вызове спрайта будет выводиться пионерская
форма
              parts:
                pioneer: sprites/mt/mt_1_pioneer.png # пионерская форма
                dress: sprites/mt/mt_1_dress.png # платье
                swimsuit: sprites/mt/mt_1_swim.png # купальник

      # остальные параметры
```

Чтобы поменять эмоцию и одежду, мы можем воспользоваться кратким синтаксисом:

```
# выводит спрайт mt с улыбкой и в платье
show mt smile dress
```

Как и в случае с анимациями, при использовании краткого синтаксиса важен порядок групп. В данном случае порядок определяется внутри файла ресурсов.

Также мы можем воспользоваться полным синтаксисом:

```
# выводит спрайт mt с улыбкой и в платье
# порядок групп неважен
show mt face: smile uniform: dress
```

Кроме того, можно убрать какую-то часть при необходимости. Для этого достаточно прописать точку вместо названия части спрайта:

```
# выводит спрайт mt с улыбкой и без одежды, краткий синтаксис
show mt smile .

# делает то же самое, полный синтаксис
show mt face: smile uniform: .
```

hide

Убирает спрайт персонажа со сцены.

Синтаксис

```
hide [charName] {with}
```

Параметры

- `charName` — имя персонажа.
  Обязательный параметр

- `with` — название и параметры перехода.

Примеры использования

```
# убирает спрайт Химицу без перехода
hide hi
```

Демонстрация

```
# убирает спрайт Химицу с переходом dissolve
hide hi with dissolve
```

Демонстрация

Переходы

Синтаксис

Переходы прописываются у определённых команд после ключевого параметра `with`.

В качестве значения может быть указано название перехода и его параметры, или сокращенное имя из заранее определённого списка (также существует встроенный в игру список).

Параметры перехода указываются после его названия в том порядке, в котором они прописаны в документации. Если у каких-то параметров есть значение по умолчанию, их можно не указывать — но только в том случае, если параметры, стоящие после, тоже не планируется использовать.

В противном же случае можно явно указать название параметров и их значения, при этом порядок неважен, и часть параметров (если у них есть значения по умолчанию) можно пропустить.

Это делается при помощи символа `:` (двоеточие), которое разделяет название параметра и его значение:

```
with [название перехода] параметр: значение параметр2: значение
```

Рассмотрим на примере перехода `crossfade` спрайтов: у данного перехода есть два параметра — `fadeOut` и `fadeIn`, которые отвечают за продолжительность исчезновения активного спрайта и появление нового соответственно. У обоих параметров есть значения по умолчанию, поэтому указывать их необязательно.

Попробуем указать значение параметра `fadeIn`, не указывая при этом `fadeOut`:

```
# выводим спрайт Химицу
show hi with dissolve
"..."
# применяем переход crossfade, указав нужный параметр
show hi at right with crossfade fadein: 3
```

Таким образом активный спрайт Химицу исчезнет через 0.5 секунд — именно такое значение по умолчанию у `fadeOut` — а новый спрайт появится справа через 3 секунды.

Демонстрация

> При использовании такого синтаксиса должны быть указаны явно названия всех параметров, значения которых планируется изменить.

Если бы мы использовали короткий синтаксис, пришлось бы дополнительно указывать значение для `fadeOut`:

```
# выводим спрайт Химицу
show hi with dissolve
"..."
# применяем переход crossfade; делает то же самое, что и в предыдущем примере
show hi at right with crossfade 0.5 3
```

Фоны (background)

Список переходов, которые можно использовать с командой `scene`.

Dissolve

```
dissolve [duration] (useLerp)
```

- `duration` — продолжительность перехода в секундах.
- `useLerp` — булевое значение. Если true, будет использоваться линейная интерполяция. По умолчанию стоит `false`.

Демонстрация

1

```
scene bg akihibara_day
"..."
window hide
scene bg akihibara_night with dissolve 2
pause hard 2
"..."
```

2

```
scene bg akihibara_day
"..."
window hide
scene bg akihibara_night with dissolve 2 true
pause hard 2
```

```
"..."
```

Fade

```
fade [fadeOut] [hold] [fadeIn] (color)
```

- `fadeOut` — продолжительность затемнения в секундах.
- `hold` — продолжительность паузы после затемнения в секундах.
- `fadeIn` — продолжительность исчезновения затемнения в секундах.
- `color` — цвет затемнения, может быть ID цвета или его значение. По умолчанию чёрный.

Демонстрация

1

```
scene bg akihibara_day
"..."
window hide
scene bg akihibara_night with fade 1 2 1 yellow
pause hard 2
"..."
```

Pattern

```
pattern [pattern] (duration)
```

- `pattern` — название изображения паттерна.
- `duration` — продолжительность перехода в секундах. По умолчанию 1 секунда.

Демонстрация

1

```
scene bg akihibara_day
"..."
window hide
scene bg akihibara_night with pattern bottomleft 2
pause hard 2
"..."
```

SlideIn

```
slidein [direction] (duration)
```

- `direction` — направление, в котором фон должен "заезжать" в кадр.
  Возможные значения:
  - `up` — верх
  - `down` — низ
  - `left` — лево
  - `right` — право
- `duration` — продолжительность перехода в секундах. По умолчанию 1 секунда.

Демонстрация

1

```
scene bg akihibara_day
"..."
window hide
scene bg akihibara_night with slidein left 2
pause hard 2
```

```
"..."
```

SlideOut

```
slideout [direction] (duration)
```

- `direction` — направление, в котором активный фон должен "выезжать" из кадра.
  Возможные значения:
  - `up` — верх
  - `down` — низ
  - `left` — лево
  - `right` — право
- `duration` — продолжительность перехода в секундах. По умолчанию 1 секунда.

Демонстрация

1

```
scene bg akihibara_day
"..."
window hide
scene bg akihibara_night with slideout left 2
pause hard 2
"..."
```

Спрайты (character)

Список переходов, которые можно использовать с командой `show`.

Crossfade

Одновременно меняет прозрачность у активного спрайта и его замены.

```
crossfade (fadeOut) (fadeIn)
```

- `fadeOut` — продолжительность исчезновения активного спрайта.
  По умолчанию 0.5 секунд.
- `fadeIn` — продолжительность появления нового спрайта.
  По умолчанию 0.5 секунд.

Демонстрация

```
scene bg akihibara_day
"..."
show hi
"..."
window hide
show hi oldforma with crossfade 1 1
pause hard 2
"..."
```

TrueDissolve

Выводит новый спрайт позади активного, затем уменьшает прозрачность активного.

```
truedissolve [duration]
```

- `duration` — продолжительность перехода.

Демонстрация

```
scene bg akihibara_day
"..."
show hi
"..."
window hide
show hi plate with truedissolve 1
pause hard 2
"..."
```

Dissolve

Вариативный переход, объединяющий в себе `Crossfade` и `TrueDissolve` .

В случае, если у спрайтов совпадают значения `CharID` и `PoseID`, будет использован переход `TrueDissolve`, в противном случае — `Crossfade`.

```
dissolve [duration]
```

- `duration` — продолжительность перехода

Fade

Убирает активный спрайт, затем после определённой паузы выводит новый (если используется в команде hide, то просто убирает).

```
fade [fadeIn] [hold] [fadeOut]
```

- `fadeIn` — продолжительность исчезновения активного спрайта в секундах.
- `hold` — продолжительность паузы.
- `fadeOut` — продолжительность появления нового спрайта в секундах.

Демонстрация

```
scene bg akihibara_day
"..."
show hi
"..."
window hide
show hi plate with fade 0.5 1 0.5
pause hard 2
"..."
```

Экран (screen)

Список переходов экрана, которые можно использовать с командой `with`.

Fade

Затемняет экран, затем после определённой паузы убирает затемнение.

```
fade (fadeOut) (hold) (fadeIn) (color)
```

- `fadeOut` — продолжительность затемнения в секундах.
  По умолчанию 0.5 секунд.
- `hold` — продолжительность паузы после затемнения в секундах.
  По умолчанию 1 секунда.
- `fadeIn` — продолжительность исчезновения затемнения в секундах.
  По умолчанию 0.5 секунд.
- `color` — цвет затемнения, может быть ID цвета или его значение.
  По умолчанию чёрный.

Демонстрация

```
scene bg akihibara_day
"..."
with fade 1 1 1 red
"..."
```

Hpunch / Vpunch

Горизонтальная / вертикальная встряска экрана.

```
hpunch (duration) (strength) (vibrato) (randomness) (fadeOut)
```

- `duration` — продолжительность тряски в секундах.
  По умолчанию 1.
- `strength` — сила тряски.
  По умолчанию 3.
- `vibrato` — сила вибрации.
  По умолчанию 10.
- `randomness` — значение от 0 до 180, определяет, насколько направление тряски будет случайным.
  По умолчанию 0.
- `fadeOut` — булевое значение. Если true, то тряска закончится плавно.
  По умолчанию true.

hpunch

```
scene bg akihibara_day
"..."
with hpunch
"..."
```

vpunch

```
scene bg akihibara_day
"..."
with vpunch
```

```
"..."
```

Звук

play

Воспроизводит различные звуки.

Синтаксис

```
play [channelName] [trackName] {fadeIn} {fadeOut} {skip} {pause} {forceloop} | crossfade
```

Параметры

- `channelName` — название звукового канала, на котором должен воспроизводиться трек (звук). Подробнее о звуковых каналах написано ниже.
  Обязательный параметр.
- `trackName` — название трека, который нужно проиграть.
  Обязательный параметр.
- `fadeIn` — время подъема громкости в секундах.
  Значение по умолчанию: `0`.
- `fadeOut` — время затухания текущего активного на канале трека в секундах.
  Значение по умолчанию: `0`.
- `skip` — время в секундах, которое будет пропущено у активного трека (например, чтобы воспроизвести трек не с самого начала, а с середины).
  Значение по умолчанию: `0`.
- `pause` — задержка в секундах перед началом воспроизведения трека.
  Значение по умолчанию: `0`.
- `forceloop` — булевое значение (`true` или `false`). Позволяет переопределить настройки зацикливания на звуковом канале. Если указано `true` — после окончания проигрывания трек будет воспроизводиться заново, если `false` — то не будет.

- `crossfade` — флаг. Если указан, то при смене трека на канале (включая повторное воспроизведение) будет использоваться перекрестное затухание, т.е. плавный переход между двумя треками. Данная функция использует значения параметров `fadeIn` и `fadeOut` для создания перехода между треками.

Звуковые каналы

Всего существует 4 звуковых канала:

- `music` — музыка
- `ambience` — фоновые звуки, звуки окружения
- `sfx` — различные звуковые эффекты
- `loop` — дополнительный канал для всего остального

У следующих каналов по умолчанию включено зацикливание треков:

- `music`
- `ambience`
- `loop`

У канала `ambience` все треки по умолчанию зацикливаются с флагом `crossfade`.

Производные каналы

Если вам нужно больше звуковых каналов (например, чтобы воспроизвести одновременно несколько звуков), вы можете создать производный канал из любого из 4-х основных каналов.

Для этого достаточно просто использовать любое название, которое начинается с имени нужного вам канала:

```
# воспроизводит трек summertime на канале music_2.
# music_2 - копия канала music
play music_2 summertime
```

```
# воспроизводит звук door_bell_bar на канале sfx_2
# sfx_2 - копия канала sfx
play sfx_2 door_bell_bar
```

stop

Останавливает воспроизведение звука.

Синтаксис

```
stop [channelName] {fadeout} |all|
```

Параметры

- `channelName` — название канала.
  Обязательный параметр.
- `fadeOut` — время затухания в секундах.
  Значение по умолчанию: `0`.
- `all` — флаг. Если указан, то будут остановлены все звуки на производных каналах.

> При указании этого флага, значением `channelName` должно быть имя одного из 4-х основных каналов.

Пример использования

```
# останавливает воспроизведение на канале music
stop music fadeout 1

# останавливает воспроизведение на канале ambience и на всех производных от него каналах
stop ambience fadeout 1 all
```

volume

Устанавливает значение громкости звука на канале.

Синтаксис

```
volume [channelName] [volume] {fade}
```

Параметры

- `channelName` — название канала.
- `volume` — громкость в диапазоне от `0` до `1`.
- `fade` — продолжительность плавного изменения громкости в секундах.
  Значение по умолчанию: `0`.

Пример использования

```
# плавно в течение 1 секунды уменьшает громкость канала music на половину
volume music 0.5 fade 1
```

Партикли

add particles

Добавляет на игровую сцену партикли.

Синтаксис

```
add particles [name] (fadein)
```

Параметры

- `name` — название партиклей.
  Обязательный параметр.

- `fadein` — продолжительность времени появления партиклей в секундах.
  Значение по умолчанию: `0`.

Примеры использования

```
# добавляет партикли petalsmain - лепестки сакуры
add particles petalsmain
```

Демонстрация

```
# лепестки сакуры с %fadein% = 5
add particles petalsmain 5
```

Демонстрация

remove particles

Убирает со сцены партикли.

Синтаксис

```
remove particles [name] (fadeout)
```

Параметры

- `name` — название партиклей.
  Обязательный параметр.

- `fadeout` — продолжительность времени исчезновения партиклей в секундах.
  Значение по умолчанию: `0`.

Пример использования

```
# убирает партикли petalsmain
remove particles petalsmain fadeout 1
```

Остальные команды

ach

Разблокирует определенное достижение в Steam.

Синтаксис

```
ach [id]
```

Параметры

- `id` — ID достижения.
  Обязательный параметр.

Пример использования

```
# разблокирует достижение "First Step"
ach firststep
```

backdrop

Выводит текст поверх определённого набора фонов с затемнением. Используется в основном при переходах между главами сценария.

Синтаксис

```
backdrop |xml| (text) {duration} {bg}
```

Параметры

- `xml` — флаг. Помечает, что значение параметра `text` — это ID заранее заданного текста для команды backdrop.
  Опциональный параметр.

- `text` — текст сноски. Если перед текстом указан флаг `xml`, то будет восприниматься как ID заранее определённого текста, иначе будет выведен заданный текст.
  Опциональный параметр.

- `duration` — продолжительность вывода в секундах.
  Опциональный параметр.
  Значение по умолчанию: `5`.

- `bg` — ключевой параметр, после него указывается ID backdrop фона.
  Опциональный параметр.
  Значение по умолчанию: `default`.

Примеры использования

```
# выводит стандартный backdrop-фон на 5 секунд без текста
backdrop
```

Демонстрация

```
# выводит стандартный backdrop-фон и текст с ID back1
backdrop xml back1
```

Демонстрация

```
# выводит backdrop-фон el с текстом "День первый" на 10 секунд
backdrop "День первый" bg el duration 10
```

Демонстрация

chibi

Выводит иконки-подсказки с характерным звуком в левом нижем углу экрана.

Синтаксис

```
chibi |mute| [ids]
```

Параметры

- `mute` — флаг. Если указан, то при выполнении команды звук проигрываться не будет.

- `ids` — ID иконок, которые нужно вывести. Разделяются пробелом. Ограничений на кол-во нет, однако стоит учитывать, что размер видимой области экрана ограничен.
  Обязательный параметр.

Примеры использования

```
chibi ca_angry hi_happy
```

Демонстрация

```
# без звука
chibi mute ca_happy hi_angry
```

Демонстрация

creditsmode

Включает режим титров, в котором у игрока отбирается возможность использовать пропуск строк. Также игрок не сможет открыть внутриигровое меню — только вернуться в главное.

Синтаксис

```
creditsmode (state)
```

Параметры

- `state` — включает или выключает режим титров, булевое значение. true или false соответственно.
  Значение по умолчанию: `true`

load

Загружает сценарий.

Синтаксис

```
load [name] (label)
```

Параметры

- `name` — название сценария.
  Обязательный параметр.
- `label` — название лейбла, с которого нужно начать проигрывать сценарий.
  Необязательный параметр.

Пример использования

```
# загружает сценарий day2
load day2
```

```
# загружает сценарий day2 с лейбла day2_4
load day2 day2_4
```

menu

Выводит меню с выборами.

Синтаксис

```
menu
{
   [choices]
}
```

Параметры

- `choices` — список выборов.
  Обязательный параметр, должен содержать хотя бы один выбор.

Примеры использования

```
menu
{
    "Выбор 1"
    {
      # команды внутри этого блока будут выполняться, если игрок выберет первый вариант ответа
      "Вы выбрали 1 вариант!"
    }

    "Выбор 2"
    {
      # команды внутри этого блока будут выполняться, если игрок выберет второй вариант ответа
      "Вы выбрали 2 вариант!"
    }
}
```

Демонстрация

Условия

Также у выборов могут быть условия. Условия указываются при помощи ключевого слова if, которое пишется после названия выбора. В случае, если условие не выполнено, выбор выводиться не будет:

```
var lovePoints = 0

menu
{
    # данный вариант выбора будет показан, если значение lovePoints будет больше 0
    "Выбор 1" if $lovePoints > 0
    {
      # команды внутри этого блока будут выполняться, если игрок выберет первый вариант ответа
      "Вы выбрали 1 вариант!"
    }

    "Выбор 2"
    {
      # команды внутри этого блока будут выполняться, если игрок выберет второй вариант ответа
      "Вы выбрали 2 вариант!"
    }
}
```

Демонстрация

Аналогично можно указывать условия для команды `menu` в целом:

```
var lovePoints = 0
```

```
# меню с выборами будет показано, если значение lovePoints будет больше 0
menu if $lovePoints > 0
{
    "Выбор 1"
    {
      # команды внутри этого блока будут выполняться, если игрок выберет первый вариант ответа
      "Вы выбрали 1 вариант!"
    }

    "Выбор 2"
    {
      # команды внутри этого блока будут выполняться, если игрок выберет второй вариант ответа
      "Вы выбрали 2 вариант!"
    }
}
```

overlay_color

Выводит поверх всего игрового экрана определённый цвет.

Синтаксис

```
overlay_color [color] {opacity} {fade} |skip|
```

Параметры

- `color` — цвет.
  Значение `clear` убирает оверлей-цвет.
  Обязательный параметр.

- `opacity` — прозрачность в диапазоне от `0` до `1`. Если указан, то значение прозрачности цвета будет изменено в соответствии с параметром.

- `fade` — время перехода в секундах. По умолчанию `0`.

- `skip` — флаг. Если указан, то игра не будет ждать окончания выполнения команды.

Примеры использования

```
"..."
# выводит чёрный оверлей-цвет с непрозрачностью = 50% за 1 секунду
overlay_color black opacity 0.5 fade 1
"..."
```

Демонстрация

```
# убирает overlay-цвет за 1 секунду
overlay_color clear fade 1
```

Демонстрация

pause

Пауза.

Синтаксис

```
pause |hard| [duration]
```

Параметры

- `hard` — флаг. Если указан, то паузу нельзя будет пропустить.
- `duration` — продолжительность паузы в секундах. Обязательный параметр.

Пример использования

```
# обычная пауза на 5 секунд, которую можно пропустить кликом
pause 5

# hard-пауза на 5 секунд, которую нельзя пропустить
pause hard 5
```

posteffect

Включает эффект пост-обработки.

Синтаксис

```
posteffect [effectName]
```

Параметры

- `posteffect` — название эффекта.

  Возможные значения:
  `none` — отключает эффект
  `sepia` — эффект сепии
  `chromaticaberration` — хроматическая аберрация

  Обязательный параметр.

Пример использования

```
scene bg akihibara_day with dissolve
show hi
"..."

# включает эффект сепии
posteffect sepia

show hi hands2
hi "Это эффект сепии."

# включает хроматическую аберрацию
posteffect chromaticaberration

show hi laugh hands4
```

```
hi "Это хроматическая аберрация."

# отключает эффекты
posteffect none

show hi normal hands1
hi "А сейчас все эффекты выключены."
```

Демонстрация

rollcredits

Показывает титры.

Текст титров берётся из меню "Помощь".

Синтаксис

```
rollcredits (duration)
```

Параметры

- `duration` — продолжительность показа титров в секундах. Значение по умолчанию: `60`.

Пример использования

```
scene bg tokio_street_day with dissolve
rollcredits
```

Вложенные команды

Т.к. сценарий ожидает конца выполнения команды, все команды, прописанные после титров, будут выполнены только тогда, когда титры закончатся.

Чтобы выполнить какие-то команды во время показа титров, их нужно прописывать внутри команды `rollcredits`:

```
rollcredits
{
    scene bg akihibara_night with dissolve
    play music burita
```

```
    pause hard 10


    scene bg bar_inside with dissolve
}
```

sequence и trigger

sequence

Определяет последовательность команд, которую можно выполнить внутри сценария при помощи команды `trigger` или тэга `<seq>`.

Синтаксис

```
sequence [name]
{
   [commands]
}
```

Параметры

- `name` — название сиквенса.
- `commands` — команды.

Пример использования

```
# вывод фона
scene bg school_corridor with dissolve

# определяет сиквенс hi_sad, который меняет эмоцию у спрайта Химицу
sequence hi_sad
{
   show hi sad
}

# вывод спрайта Химицу
show hi with dissolve
```

```
hi "Нико-кун! Рада, что ты пришёл в школу."

# Сиквенс запускается в тэге <seq>
"Привычно встрепенулась она, но тут же <seq hi_sad> помрачнела."
```

trigger

Делает то же самое, что и тэг `<seq>`, — запускает сиквенс — только отдельной командой.

Синтаксис

```
trigger [name]
```

Параметры

- `name` — название сиквенса.

Пример использования

```
trigger hi_sad
```

unlock_ending

Разблокирует определённую концовку.

Синтаксис

```
unlock_ending [name]
```

Параметры

- `name` — название концовки

Пример использования

```
# разблокирует плохую концовку Элли
unlock_ending ellie_bad
```

with

Команда, у которой есть 3 назначения:

- Комбинировать переход фона со спрайтами
- Комбинировать переходы спрайтов
- Осуществлять переходы экрана

Синтаксис

```
with [transition] |strict|
```

- `transition` — название перехода и его параметры. Тип перехода зависит от того, для чего используется команда. Подробнее об этом далее.
- `strict` — флаг. Если указан, команда будет осуществлять только переходы экрана.

Переход фона со спрайтами

Спавнит спрайты персонажей на фоне до того, как он появится на экране. Таким образом, фон будет выведен уже со спрайтами.

Для того, чтобы использовать команду для осуществления таких переходов, нужно прописать под командой вывода фона команды для вывода (или скрытия) спрайтов, затем под ними прописать команду `with` с необходимым переходом **фона**:

```
scene bg akihibara_night
show hi
```

```
show ca at right
with dissolve
# dissolve - переход фона
```

Таким образом, фон появится сразу со спрайтами:

Демонстрация

Если у любой команды, стоящей выше `with`, будет указан переход — команда не сработает — будет предпринята попытка скомбинировать команды спрайтов, стоящих выше (до первого найденного перехода), или попытка осуществить переход экрана. Если ничего из этого не вышло, появится ошибка.

Переход нескольких спрайтов

Спавнит (или наоборот убирает) несколько спрайтов с одним переходом.

Возьмём пример выше и пропишем у команды `scene` собственный переход:

```
# у команды scene теперь собственный переход
scene bg akihibara_night with pattern bottomleft
show hi
show ca at right
with dissolve
# теперь dissolve - переход спрайтов
```

Теперь одновременно при помощи `dissolve` будут появляться только спрайты, т.к. у команды scene определён собственный переход:

Демонстрация

Переходы экрана

Также команда `with` может использоваться для осуществления переходов экрана. Такой тип переходов будет автоматически подразумеваться, если выше нет команды для вывода фона и/или команд для вывода/скрытия спрайтов, или же у ближайшей команды указан собственный переход.

Если по какой-то причине вам нужно будет осуществить переход экрана, когда выше находятся другие команды без собственных переходов (scene, show, hide) — вы можете воспользоваться флагом `strict`. При указании этого флага, команда будет интерпретироваться только как команда для перехода экрана, игнорируя все вышестоящие команды.