resources.yaml

- Описание
- backdrop_bg
- backdrop_text
- bg
- cg
- catalogs
- characters
- chibis
- collections
- colors
- entryPoint
- help
- live2d_characters
- menu
- narrators
- notes
- particles
- positions
- scenarios

- sizes
- spritecolor
- sound
- transitions
- variables

Описание

Файл, содержащий информацию о всех ресурсах мода. Именно в нём прописываются все **не**встроенные в игру ресурсы, которые вы хотите использовать в игре.

Поддерживает следующие параметры:

- backdrop_bg
- backdrop_text
- bg
- cg
- catalogs
- characters
- chibis
- collections
- colors
- entryPoint
- help
- live2d_characters
- menu
- narrators
- notes
- particles
- positions
- scenarios

- sizes
- spritescolors
- sound
- transitions
- variables

Большая часть из этих параметров — списки, внутри которых прописываются те или иные ресурсы.

backdrop_bg

Тип: список ресурсных нодов

Типы принимаемых ассетов:

- `texture` — тип по умолчанию
- `sprite`
- `prefab`

Определяет фоны для команды backdrop.

Пример использования:

```
# resources.yaml
---
  backdrop_bg:
    eli_1: backdrops/eli_1.jpg
    himitsu_2: backdrops/himitsu_2.jpg
    kagome_3: ~@bundle[prefab]://backdrop_kagome_3.prefab #подгрузка префаба из каталога "bundle"
```

backdrop_text

Тип: список языковых нодов

Определяет локализируемый текст для команды backdrop.

Пример использования:

```
# resources.yaml
---
  backdrop_text:
    day1: День 1
    day2:
      ru: День 2
      en: Day 2
    day3: День 3
```

bg

Тип: список ресурсных нодов
Типы принимаемых ассетов:

- `texture` — тип по умолчанию
- `sprite`
- `prefab`

Определяет фоны для команды scene.

Ноды данного параметра могут принимать путь к одному или нескольким ассетам:

```
# resources.yaml
---
  bg:
    ext_road_day: bg/ext_road_day.jpg
```

В таком случае указанный ассет — `bg/ext_road_day.jpg` — будет использоваться вне зависимости от настроек анимации фонов в игре.

Чтобы задать разные ассеты для разных настроек анимации, следует указывать пути внутри параметров **`static`** и **`anim`**:

```
# resources.yaml
---
  bg:
    ext_road_day:
      static: bg/ext_road_day.jpg
      anim: ~@bundle[prefab]://bg/ext_road_day.prefab #префаб анимированного фона из каталога "bundle"
    int_bus_day:
      static: bg/int_bus.jpg
      anim: bg/int_bus_2.jpg #статичное изображение, однако оно будет использоваться, если в настройках игры включена анимация фонов
```

Если один из параметров не указан (`static` или `anim`), то в обоих режимах будет использоваться ассет указанного параметра.

cg

Тип: список ресурсных нодов
Типы принимаемых ассетов:

- `texture` — тип по умолчанию
- `sprite`
- `prefab`

Определяет цг для команды scene.

Синтаксис полностью идентичен bg, отличается только название нода:

```
# resources.yaml
---
  cg:
    d1_rena_sunset: cg/d1_rena_sunset.jpg
    d3_dv_guitar:
      static: cg/d3_dv_guitar.jpg
      anim: ~@bundle[prefab]://cg/d3_dv_guitar.prefab
```

catalogs

Тип: список ресурсных нодов

Определяет Unity-каталоги для загрузки ассетов.

В нодах указывается путь к `.json` файлу собранного каталога:

```
# resources.yaml
---
 catalogs:
   bundle: catalogs/bundle/catalog.json #определение каталога "bundle"
```

> Каталог должен лежать внутри папки мода и не может находится внутри другого каталога

Также можно указать каталоги для разных платформ:

```
# resources.yaml
---
 catalogs:
   bundle: #определение каталога "bundle" под разные платформы
     windows: catalogs/bundle/win/catalog.json
     linux: catalogs/bundle/linux/catalog.json
     macos: catalogs/bundle/mac/catalog.json
     android: catalogs/bundle/android/catalog.json
     ios: catalogs/bundle/ios/catalog.json
```

Список доступных плафторм:

- `windows`
- `linux`
- `macos`
- `android`

- `ios`

Если платформа не указана (как в первом примере), то будет выставлено значение по умолчанию — `windows`.

После определения каталога, вы можете использовать ассеты из него при помощи следующего синтаксиса:
`~@имяКаталога://адресАссета`

Тип ассета указывается в квадратных скобках после имени каталога:
`~@имяКаталога[типАссета]://адресАссета`

Если тип не указан, будет использован тип по умолчанию в зависимости от параметра.

Примеры использования путей с каталогом "bundle":
`~@bundle://bg/bus_stop # ассет с адресом bg/bus_stop из каталога`
`~@bundle[prefab]://bg/bus_stop # префаб с адресом bg/bus_stop из каталога`

characters

Тип: список

Определяет статичные спрайты персонажей для команды show.

Внутри нода прописываются ноды с параметрами статичных (неанимированных) спрайтов персонажей:

```
# resources.yaml
---
  characters:
    dv:
      # ... параметры
    un:
      # ... параметры
    sl:
      # ... параметры
    us:
      # ... параметры
```

Каждый спрайтовый нод может содержать следующие параметры:

- `poses`
- `positions`
- `sizes`
- `shortnames`
- `default`

poses

Тип: список

Определяет позы спрайта.
Под позами понимаются различные вариации спрайта:



3 различных позы на примере спрайта из Бесконечного Лета

Каждая поза может содержать несколько параметров:

- `parts` — обязательный параметр
- `canvasSize`
- `positions`
- `sizes`
- `spritecolors`

```
# resources.yaml
---
  characters:
    mt:
      poses:
        front_1:
          parts:
           # ... параметры
          canvasSize: 900 1080
          positions:
           # ... параметры
          sizes:
           # ... параметры
          spritecolors:
           # ... параметры
```

parts

Тип: список

Определяет части, из которых собирается спрайт.

Части могут быть двух типов:

- постоянные
- непостоянные

Тип ассетов у частей — `texture`.

Постоянные части

Постоянные части видны всегда и никогда не меняются (например, тело персонажа).

Определяются следующим образом:

```
# resources.yaml
---
  characters:
    mt:
      poses:
        front_1:
          parts:
            body: sprites/mt/mt_1_body.png # тело персонажа, которое должно отображаться всегда


          # ... остальные параметры
```

На примере выше мы определили часть `body` — тело спрайта, на которое будут накладываться остальные элементы.

Непостоянные части

Непостоянные — соответственно, те части, которые могут меняться. Это эмоции, одежда, различные аксессуары и прочее. Такие части объединяются в группы, которые позволяют менять активную (отображаемую) часть из сценария.

Непостоянные части определяются так же, как и постоянные, только внутри отдельного нода, начинающегося со знака `~` — такой нод называется группой:

```
# resources.yaml
---
  characters:
    mt:
      poses:
        front_1:
          parts:
            body: sprites/mt/mt_1_body.png # это постоянная часть, которую мы определили ранее
            ~uniform: # группа uniform
              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
```

У частей спрайта в группе также может указываться параметр `position`, синтаксис аналогичен.

На примере выше мы создали группу `uniform` с различными вариациями одежды, которые можно использовать в игре. Однако мы не указали, какая часть должна отображаться по умолчанию — в случае, если мы явно не обозначили это в сценарии.

Чтобы сделать какую-либо часть в группе частью, отображаемой по умолчанию, можно воспользоваться параметром `default`:

```
# resources.yaml
---
  characters:
    mt:
      poses:
        front_1:
          parts:
            body: sprites/mt/mt_1_body.png # это постоянная часть, которую мы определили ранее
            ~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
```

Как видно из примера, ID части, выводимой по умолчанию, прописывается в ноде `default`, а сами части — в `parts`.

Если нод `default` не указан, то по умолчанию ни одна часть не будет выводиться.

Позиционирование частей

Каждой части спрайта можно задать положение относительно канваса:

```
# resources.yaml
---
  characters:
    mt:
      poses:
        front_1:
          parts:
            body:
              asset: sprites/mt/mt_1_body.png
              position: 0 100

          # ... остальные параметры
```

На примере выше изображение будет смещено на 100 пикселей по вертикали **относительно левого верхнего угла** канваса спрайта. Ассет же прописывается в ноде `asset`.

> Использование параметра `position` у части спрайта возможно только если указан параметр `canvasSize` у позы.

Порядок наложения частей

Части спрайта (будь то постоянная часть или группа) накладываются друг на друга согласно порядку, в котором они определены:

```
# resources.yaml
---
  characters:
    mt:
      poses:
        front_1:
          parts:
            body: sprites/mt/mt_1_body.png # сначала будет выведено тело
```

```
      ~uniform: # затем форма
        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
      ~face: # затем лицо
        normal: sprites/mt/mt_1_normal.png # нейтральное лицо
        smile: sprites/mt/mt_1_smile.png # улыбка
        sad: sprites/mt/mt_1_sad.png # грустное лицо

    # ... остальные параметры
```

canvasSize

Тип: текстовый нод

Определяет размер канваса спрайта. Этот параметр нужен для использования параметра `position` у частей спрайта, т.к. относительно значения этого параметра производится позиционирование изображений.

Значение должно состоять из двух целых чисел, разделённых пробелом, и представлять собой корректное разрешение:

```
# resources.yaml
---
  characters:
    mt:
      poses:
        front_1:
          canvasSize: 900 1080 # 900x1080

          # ... остальные параметры
```

positions

Тип: список

Определяет именованные положения спрайта относительно игровой сцены. Не стоит путать с параметром `positions` у частей спрайта.

```
# resources.yaml
---
  characters:
    mt:
      poses:
        front_1:
          positions:
            center: 0 0 # центр
            left: -0.5 0 # лево
            right: 0.5 0 # право

          # ... остальные параметры
```

Данный параметр, указанный внутри позы, позволяет задавать положения, действительные только для этой позы. Это может быть полезно, если у разных поз спрайта разные размеры изображений, и они должны использовать разные координаты.

> Указанные внутри нода персонажа позиции будут переопределять одноимённые позиции выше по иерархии.

Также можно указать значение только для одной оси, прописав `.` в качестве значения для второй:

```
# resources.yaml
---
  characters:
    mt:
      poses:
        front_1:
          positions:
            center: -0.5 . # изменена только X-координата
            left: . -0.5 # изменена только Y-координата
```

```
        # ... остальные параметры
```

Значение для недостающей координаты будет браться из позиций с таким же ID выше по иерархии, если же таких нет — будет выставлен 0.

Подробнее о положениях можно почитать здесь.

sizes

Тип: список

Определяет именованные размеры спрайта относительно игровой сцены. Синтаксис схож с `positions`:

```
# resources.yaml
---
  characters:
    mt:
      poses:
        front_1:
          sizes:
            normal: 1 1
            close: 2 .

          # ... остальные параметры
```

Внутри каждого нода можно также прописать позиции — они будут использоваться, когда у спрайта будет указанный размер:

```
# resources.yaml
---
  characters:
    mt:
      poses:
```

```
front_1:
  positions:
    center: 0 0 # это значение будет использоваться при любых других размерах
  sizes:
    normal: 1 1
    close:
      value: 2 .
      positions: # позиции при размере close
        center: 0 0.25 # это значение будет использоваться при размере close

  # ... остальные параметры
```

Размер в таком случае указывается внутри нода `value`*.*

На примере выше, когда у спрайта в сценарии будут прописаны `size close` и `at center`, в качестве координат для `center` будет использоваться значение `0 0.25`, а не `0 0`.

Аналогично `positions`, размеры, указанные внутри нода персонажа, будут переопределять одноимённые размеры выше по иерархии.

Подробнее о размерах можно почитать здесь.

spritecolor

Тип: список

Цвета для наложения на спрайт. Используются для более естественного отображения спрайтов на различных фонах (день, ночь и т.п.).

Значением может быть как цвет, заданный в `colors`, так и HEX-код или RGB(A)-код.

```
# resources.yaml
---
  characters:
    mt:
      poses:
        front_1:
          spritecolor:
            tokyo_street_sunset: #FFF1ED

          # ... остальные параметры
```

Подробнее о цветах спрайтов можно почитать здесь.

shortnames

Тип: список

Определяет короткие имена для поз и различных вариаций частей групп, которые можно использовать в сценарии.

Короткие имена прописываются так же, как прописываются позы и группы внутри сценария:

```
# resources.yaml
---
  characters:
    mt:
      shortnames:
        front_sd: front smile dress # или front face:smile uniform:dress
```

После определения коротого имени, его можно использовать внутри сценария:

```
show mt front_sd with dissolve
```

default

Тип: текстовый нод

Определяет позу и части групп, которые будут выводиться по умолчанию (если конкретные значения не указаны в команде сценария).

В качестве значения можно прописать короткое имя или позу и параметры групп:

```
# resources.yaml
---
  characters:
    mt:
      default: front smile dress
      # ... остальные параметры
```

Теперь, когда в сценарии будет прописано `show mt`, по умолчанию будет спавнится вариация `front smile dress`.

positions

Тип: список

Определяет позиции, которые могут использовать все позы спрайта. Любые позиции внутри конкретной позы будут переопределять данные позиции, если у них совпадают ID.

Синтаксис идентичен синтаксису positions внутри позы:

```
# resources.yaml
---
```

```
  characters:
    mt:
      positions:
        center: 0 0
        left: -0.5 0
      # ... остальные параметры
```

sizes

Тип: список

Определяет размеры, которые могут использовать все позы спрайта. Любые размеры внутри конкретной позы будут переопределять данные размеры, если у них совпадают ID.

```
# resources.yaml
---
  characters:
    mt:
      sizes:
        normal: 1 1
        close: 2 .
        far:
          value: 0.7 0.7
          positions:
            center: 0 -0.25
      # ... остальные параметры
```

spritecolor

Тип: список

Определяет оверлей-цвета спрайтов, которые могут использовать все позы спрайта. Любые цвета внутри конкретной позы будут

переопределять данные цвета, если у них совпадают ID.

```
# resources.yaml
---
  characters:
    mt:
      spritecolor:
        tokyo_street_sunset: #FFF1ED
      # ... остальные параметры
```

chibis

Тип: список

Тип принимаемых ассетов:

- `texture` — тип по умолчанию
- `sprite`

Добавляет изображение для команды `chibi`.

Синтаксис:

```
# resources.yaml
---
chibis:
  chibi1: images/chibi1.png
  chibi2: images/chibi2.png
```

collections

Тип: список

В данном ноде определяются элементы галереи — фоны, цг, музыка, концовки и сноски (доступны из меню "Помощь").

Все элементы, за исключением концовок, становятся доступны в галерее после того, как игрок увидит (услышит) их в игре. Концовки же открываются специальной командой в сценарии.

Нод может содержать следующие параметры, каждый из которых является списком:

- `bg`
- `cg`
- `endings`
- `notes`
- `tracks`

Все элементы в игре выводятся в том порядке, в котором они указаны (сверху вниз).

bg / cg

Тип: список

Список фонов/цг для галереи. ID каждого нода в данном списке должен соответствовать существующему фону/цг — встроенному в игру или же определённого внутри `resources.yaml`:

```
# resources.yaml
---
  bg:
    bus_stop: bg/bus_stop.jpg # определение фона bus_stop
  cg:
    d1_rena_sunset: cg/d1_rena_sunset.jpg # определение цг d1_rena_sunset

  collections:
    bg:
      bus_stop: # добавление фона bus_stop в галерею
        thumb: gallery/thumbs/bg/bus_stop_284x160.jpg # превью
    cg:
      d1_rena_sunset: # добавление цг d1_rena_sunset в галерею
        thumb: gallery/thumbs/cg/d1_rena_sunset_284x160.jpg

    # остальные параметры
```

Каждый нод может содержать следующие параметры:

- `name` — название
- `thumb` — превью
- `variations` — вариации

name

Тип: языковой нод

Название фона/цг. Выводится в меню при просмотре, если у фона/цг есть вариации.

Если не указан, будет использован ID фона/цг.

thumb

Тип: ассет
Типы принимаемых ассетов:

- `texture` — тип по умолчанию
- `sprite`

Файл превью для галереи.
Предпочтительное разрешение: **284x160.**

variatons

Тип: список

Вариации фона/цг. Содержит ноды с вариациями, каждый из которых может содержать параметр `name` — название, которое будет отображаться в меню при просмотре:

```
# resources.yaml
---
 bg:
  bus_stop: bg/bus_stop.jpg # определение фона bus_stop
  bus_stop_day: bg/bus_stop_day.jpg # определение фона bus_stop_day
 cg:
  d1_rena_sunset: cg/d1_rena_sunset.jpg # определение цг d1_rena_sunset
  d1_rena_day: cg/d1_rena_day.jpg # определение цг d1_rena_day

 collections:
  bg:
   bus_stop: # добавление фона bus_stop в галерею
    name: "Автобусная остановка"
    thumb: gallery/thumbs/bg/bus_stop_284x160.jpg # превью
    variations: # вариации
     bus_stop_day:
      name: "Автобусная остановка, день"
  cg:
   d1_rena_sunset: # добавление цг d1_rena_sunset в галерею
    name: "Рена"
    thumb: gallery/thumbs/cg/d1_rena_sunset_284x160.jpg,
    variations: # вариации
```

```
      d1_rena_day:
        name: "Рена, день"


  # ... остальные параметры
```

> Вариации не могут иметь вариаций.

> Вариациям можно не задавать параметр `thumb`, т.к. превью для них не используются.

endings

Тип: список

Список концовок, отображаемых в меню "Помощь".

Каждый нод внутри списка — языковой нод.

Поддерживают тэги разметки Rich Text.

```
# resources.yaml
---
  collections:
    endings:
      himitsu_good_end: Химицу, хорошая концовка
      himitsu_bad_end: Химицу, плохая концовка
      kagome_good_end: # концовка с переводами на два языка
        ru: Кагомэ, хорошая концовка
        en: Kagome, good ending


      # остальные параметры
```

Разблокировка той или иной концовки происходит при помощи команды внутри игрового сценария.

notes

Тип: список

Список сносок, отображаемых в меню "Помощь".
Внутри списка прописываются ID сносок:

```
# resources.yaml
---
notes: # определение сносок
  note1: "Сноска #1"
  note2: "Сноска #2"
  note3: "Сноска #3"

collections:
  notes: # сноски, которые должны отображаться в меню "Помощь"
    - note1
    - note2
    - note3
```

tracks

Тип: список

Список треков, отображаемых в галерее.

ID каждого нода внутри списка должен соответствовать ID встроенного в игру музыкального трека или трека, прописанного в `resources.yaml`.

Название трека указывается внутри нода:

```
# resources.yaml
---
  sound:
    echo_avenue_old: sound/music/echo_avenue_old.mp3 # определение трека


  collections:
    tracks:
      echo_avenue_old: Echo Avenue [Old Version] # добавление трека в галерею
```

Если название не указано, будет выведен ID трека.

Использование встроенных ресурсов

Чтобы вывести в галерее какой-либо элемент, встроенный в игру, достаточно просто прописать его ID и ID необходимых вариаций. Название и превью нужно указывать только в том случае, если вы хотите их изменить.

```
# resources.yaml
---
  collections:
    # для всех этих элементов будут использованы значения по умолчанию, встроенные в игру
    bg:
      tokyo_street_day:
        variations:
          tokyo_street_night:
          tokyo_street_sunset:
    tracks:
      a_painful_story:
      ddd_take_two:
    endings:
      himitsu_bad: # Химицу, плохая
      himitsu_good: # Химицу, хорошая
```

> Обратите внимание, что после ID нода нужно ставить двоеточие, но не нужно указывать параметры.

colors

Тип: список

Определяет цвета, которые можно использовать внутри игрового сценария и в параметрах `resources.yaml`, принимающих цвет в качестве значения.

Значением цвета может быть HEX-код или RGB(A)-код:

```
# resources.yaml
---
  colors:
    red_brown: "#a55e55" # HEX-код указывается в кавычках, т.к. иначе символ "#" будет расценён как начало комментария
    aquamarine: (127, 255, 212) # rgb-код
    pink_half_opacity: (255, 20, 147, 127) # rgba-код
```

Поддерживаемые форматы HEX-кодов:

- #RGB
- #RRGGBB
- #RGBA
- #RRGGBBAA

entryPoint

Тип: текстовый нод

Входная точка сценария.

В этом ноде прописывается ID сценария, который должен запускаться при нажатии "Новая игра". Значением должен быть ID существующего сценария (встроенного в игру или определённого в модификации).

Значение по умолчанию: `day1`

Пример:

```
# resources.yaml
---
  scenarios:
    scenario1: scenarios/scenario1.txt # определение сценария scenario1


  entryPoint: scenario1 # scenario1 будет запускаться при начале новой игры
```

help

Тип: список

В этом ноде прописывается текст "Контактов" и "Титров" для меню "Помощь".
Может содержать следующие параметры:

- `credits` — титры
- `contacts` — контакты

Оба параметра являются языковыми нодами и поддерживают тэги разметки Rich Text.

```
# resources.yaml
---
  help:
    contacts: "Текст для обратной связи"
    credits: "Текст титров"
```

live2d_characters

Тип: список

В этом ноде определяются Live2D-спрайты персонажей.

Так же, как и в случае со статичными спрайтами, каждый спрайтовый нод может содержать следующие параметры:

- `poses`
- `positions`
- `sizes`
- `spritecolor`
- `shortnames`
- `default`

Все параметры, кроме `poses`, идентичны параметрам статичных спрайтов, поэтому здесь будут рассмотрены только отличия.

Кроме того, Live2D спрайты также могут содержать параметр `layers` с информацией об анимационных слоях — синтаксис этого параметра также будет рассмотрен.

layers

Тип: список

Внутри данного параметра прописываются названия анимационных слоёв и анимаций, которые они содержат.

Слои делятся на два типа:

- Основные
- Вспомогательные

Основные слои — это те слои, состояние которых игра запоминает. Эмоции, положения рук, ног, головы и прочее. То, что должно быть записано в файле сохранения, если спрайт на экране.

Вспомогательные — слои для одноразовых анимаций, которые проигрываются и затем возвращают спрайт в исходное состояние. Это могут быть различные кивки, подпрыжки, махания руками и прочее. Информация о состоянии таких слоёв не записывается в файлах сохранений.

Определение слоёв и анимаций происходит в редакторе Unity, этот процесс рассмотрен в отдельной статье. В этой же статье будет рассмотрен процесс определения слоёв и анимаций в файле ресурсов мода, чтобы их можно было использовать внутри сценария.

Рассмотрим процесс определения нескольких слоёв:

```
# resources.yaml
---
live2d_characters
  saya:
    layers:
      emotions:
        - normal
        - smile
        - sad
      hands:
        - hands1
        - hands2
        - hands3
      ~shake:
        - head
```

Здесь определяются 3 слоя — `emotions`, `hands` и `shake`. Первые два — основные слои, для эмоций и положения рук соответственно, последний — вспомогательный. Внутри нодов слоёв прописаны названия анимаций, которые этот слой содержит.

> Для определения вспомогательного слоя нужно перед его именем поставить символ `~`.

poses

Тип: список

Так же, как и у статичных спрайтов, внутри данного нода прописываются различные позы и их параметры.

Каждая поза может содержать следующие параметры, синтаксис которых идентичен аналогичным параметрам поз статичных спрайтов:

- `positions`
- `sizes`
- `spritecolor`
- `default`

Также им доступны два новых параметра:

- `skins`
- `layers`

`layers` внутри позы определяет слои и анимации, действительные только для этой позы (или перезаписывает их, если слой с таким же названием прописан для всего спрайта в целом).

`skins` же — аналог parts статичных спрайтов, в нём прописываются пути Live2D-префабов спрайта.

skins

Тип: список
Тип принимаемых ассетов: `prefab`

Синтаксис достаточно прост, внутри нода прописываются пути к префабам Live2D моделей:

```
# resources.yaml
---
live2d_characters:
  saya:
    poses:
      front:
        skins:
          school_unfiform: ~@[bundle]://characters/saya/front/school_uniform/model.prefab # путь к префабу внутри каталога
```

Краткая форма записи

Ноды `poses` и `skins` необязательно указывать явно, если вы не намереваетесь использовать какие-либо другие параметры.

Допустим, тот же пример с определением позы `front` и скина `school_uniform` можно записать так:

```
# resources.yaml
---
live2d_characters:
  saya:
      front:
          school_unfiform: ~@[bundle]://characters/saya/front/school_uniform/model.prefab # путь к префабу внутри каталога
```

Если бы мы захотели указать какие-то позиции, размеры, слои анимации и прочее — необходимо было бы явно указывать названия нодов.

Допустим, мы определяем слои анимаций и несколько позиций для всего спрайта персонажа в целом:

```
# resources.yaml
---
live2d_characters:
  saya:
    layers:
      emotions:
        - normal
        - smile
        - sad
      hands:
        - hands1
        - hands2
        - hands3
    positions:
      center: 0 -0.25
    poses:
      front:
          school_unfiform: ~@[bundle]://characters/saya/front/school_uniform/model.prefab # путь к префабу
внутри каталога
```

Т.к. нод спрайта содержит не только позы, но и другие параметры, нам нужно явно указать название нода с позами. Внутри же нода позы `front` никаких других параметров, кроме скинов, нет — поэтому названием нода `skins` можно пренебречь. Если бы нужно было указать какие-то дополнительные параметры для позы `front`, скины бы прописывались внутри нода `skins`.

menu

Тип: список

Определяет элементы кастомизации главного меню.
Доступные параметры:

- `bg`
- `logos`
- `tracks`

Каждому элементу внутри параметров присваивается ID — целое число, которое определяет порядок выбора элементов (чем меньше число — тем выше приоритет):

```
# resources.yaml
---
  menu:
    bg:
      0:
        # параметры
      1:
        # параметры
    logos:
      0:
        # параметры
      1:
        # параметры
    tracks:
      0:
        # параметры
      1:
        # параметры
```

Также у каждого элемента может быть задан параметр `condition`, в котором прописывается условие, при котором данный элемент может быть выбран для отображения.

bg

Тип: список

Список фонов для главного меню.

Фон может задаваться одним из двух параметров:

- `asset`
- `name`

Первый используется для указания пути конкретного ассета, второй — для указания ID определённого в `resources.yaml` или встроенного в игру фона.

```
# resources.yaml
---
bg:
  bus_stop: bg/bus_stop.jpg # определение фона bus_stop


menu:
  bg:
    0:
      asset: bg/menu_bg.jpg # подгрузка фона из папки bg
    1:
      name: bus_stop # подгрузка фона bus_stop
```

Параметр `asset` принимает следующие типы:

- `texture` — тип по умолчанию
- `sprite`
- `prefab`

Также можно в ноде просто указать путь до ассета без указания других параметров, в таком случае это будет равносильно указанию параметра `asset`:

```
# resources.yaml
---
bg:
  bus_stop: bg/bus_stop.jpg # определение фона bus_stop


menu:
  bg:
    0: bg/menu_bg.jpg # подгрузка фона из папки bg
```

Эффекты постобработки

Помимо описанных выше параметров, нод также может принимать ещё несколько параметров:

- `particles`
- `twirl`
- `noise`
- `blur`

Все эти параметры — булевые переменные, т.е. их значением может быть либо `true`, либо `false`. Они отвечают за включение/отключение соответствующих эффектов пост-обработки фона в главном меню:

```
# resources.yaml
---
  menu:
    bg:
      0:
        name: house_mc_day
        twirl: true
        particles: true
        blur: true
        noise: true
```

> По умолчанию значения всех параметров — **true**.

Также можно задать overlay-цвет для фона при помощи параметра `color`:

```
# resources.yaml
---
  menu:
    bg:
      0: # пример дневной вариации фона из главного меню игры
        name: house_mc_day
        color: "#AC8E0E32"
```

logos

Тип: список

Список логотипов для главного меню.

Изображение логотипа задаётся при помощи параметра `asset`:

```
# resources.yaml
---
menu:
  logos:
    0:
      asset: images/menu_logo.png
```

Также значением `asset` может быть `default` — в таком случае игра будет использовать встроенный логотип LMR:

```
# resources.yaml
---
menu:
  logos:
    0:
```

```
    asset: default # будет использован логотип Love, Money, Rock'n'Roll
```

> Параметр `asset` данного нода — своего рода исключение, которое помимо пути ассета принимает и значение `default`.

Ещё одной особенностью параметра `asset` является возможность указать разные ассеты под разные языки, аналогично синтаксису языковых нодов, только в качестве значения нужно писать путь к ассету:

```
# resources.yaml
---
menu:
  logos:
    0:
      asset:
        ru: images/menu_logo_ru.png
        en: images/menu_logo_en.png
```

Аналогично `bg`, можно указать только путь к ассету, игнорируя остальные параметры:

```
# resources.yaml
---
menu:
  logos:
    0: images/menu_logo.png
    1: default
```

Положение и размер логотипа

Положение и размер логотипа настраиваются двумя параметрами:

- `position`
- `size`

Оба параметра принимают как и ID положения/размера, указанного в `resources.yaml`, так и числовые значения:

```
# resources.yaml
---
sizes:
  logo_normal: 0.8 0.8
positions:
  logo_top: 0 0.25

menu:
  logos:
    0:
      asset: images/menu_logo.png
      position: 0 0.25
      size: 0.8 0.8
    1:
      asset: images/menu_logo.png
      position: logo_top
      size: logo_normal
```

tracks

Тип: список

Фоновые треки для главного меню.

Аналогично нодам `bg`, трек указывается либо через `asset`, либо через `name`. Также можно указать только путь к ассету:

```
# resources.yaml
---
sizes:
  logo_normal: 0.8 0.8
positions:
  logo_top: 0 0.25
```

```
menu:
  tracks:
    0:
      asset: sound/menu_track.mp3
    1:
      name: summertime # трек, встроенный в игру
    2: sound/menu_track.mp3 # аналогично ноду с "0"
```

Параметр `asset` принимает только один вид ассетов — `sound`.

Условия вывода

Каждый нод фона/лого/трека может иметь параметр `condition`, который отвечает за условие выбора того или иного элемента.

Условие может содержать глобальные и persistent переменные, а также значения datetime:

```
# resources.yaml
---
menu:
  bg: # пример вывода стандартных фонов главного меню в зависимости от времени суток
    0:
      name: house_mc_day
      color: "#AC8E0E32"
      condition: "datetime.hour >= 7 && datetime.hour < 16"
    1:
      name: house_mc_sunset
      color: "#87173E2D"
      condition: "datetime.hour >= 16 && datetime.hour < 22"
    2:
      name: house_mc_night
      color: "#3A00BE3C"
      condition: "datetime.hour >= 22 || datetime.hour < 7"
```

> Значения глобальных переменных будут браться из continue-сохранения (которое загружается при нажатии "Продолжить" в главном меню).
>
> Если такого нет, будут использованы значения по умолчанию.

Пустые значения

Если вы хотите не выводить вообще ничего, достаточно в качестве значения для параметра `asset` указать `null`:

```
# resources.yaml
---
menu:
  bg:
    0:
      asset: null # также можно указать как 0: null
  logos:
    0:
      asset: null
  tracks:
    0:
      asset: null
```

narrators

Тип: список

Определяет имена персонажей, которые можно использовать в игровом сценарии.

Параметры:

- `name` — имя персонажа, языковой нод. Обязательный параметр.
- `advColor` — цвет для режима adv, по умолчанию #4E322BFF.
- `nvlColor` — цвет для режима nvl, по умолчанию #8D5E55FF.
- `simpleAdv` — булевый параметр.

  По умолчанию в режиме `adv` цвет имени не изменяется — он всегда белый, изменяется только цвет полупрозрачной обводки вокруг имени — это сочетается не со всеми цветами, их нужно тщательно подбирать.

  Параметр `simpleAdv` позволяет отключить обводку и использовать обычный текст, цвет которого задаётся параметром `advColor`.

  Если `true`, то будет выводиться обычный цветной текст, без обводки.
  Если `false`, то будет выводиться белый текст с цветной обводкой.

  Значение по умолчанию: `false`.

Пример:

```
# resources.yaml
---
narrators:
  hi:
    name: "Химицу"
    advColor: "#91483c"
    nvlColor: "#EC7070"
    simpleAdv: true
```

notes

Тип: список

В этом ноде определяются внутриигровые сноски.

Каждая сноска является языковым нодом и поддерживает тэги разметки Rich Text.

Пример:

```
# resources.yaml
---
notes:
  note1: "Сноска #1"
  note2:
    ru: "Сноска #2"
    en: "Note #2"
```

particles

Тип: список
Тип ассетов: `prefab`

Определяет внутриигровые партикли (различные частицы, типа лепестков сакуры).

Партикли можно загружать только из Unity-каталогов — ассет должен быть префабом с компонентом `Particle System`.

```
# resources.yaml
---
catalogs:
  bundle: catalogs/bundle/catalog.json # определение каталога "bundle"

particles:
  snow: ~@bundle://particles/snow.prefab # определение партиклей "snow" из префаба внутри каталога "bundle"
```

positions

Тип: список

В этом ноде определяются различные положения на экране, которые могут использоваться внутри игры — чаще всего они используются для расположения спрайтов персонажей.

Координаты задаются при помощи двух осей: X и Y — горизонтальной и вертикальной соответственно.

Видимое пространство находится в диапазоне от `-1` до `1`.
Абсолютный центр экрана — `0 0`.

```
# resources.yaml
---
# примеры некоторых стандартных позиций спрайтов персонажей в игре
positions:
  center: 0 -0.59
  left: -0.67 -0.59
  right: 0.67 -0.59
```

scenarios

Тип: список
Тип принимаемых ассетов: `text`

В этом списке определяются игровые сценарии.
Каждый нод может принимать путь к ассету сценария, а также к различным локализациям сценария на разные языки.

```
# resources.yaml
---
scenarios:
  day1: scenarios/day1.txt # определение сценария "day1"
  day2:
    asset: scenarios/day2.txt # определение сценария "day2"
    en: scenarios/day2_en.csv # английская локализация для "day2"
```

Как видно из примера, в случаях, когда к сценарию нужно прописать локализацию, путь к исходному сценарию указывается в параметре `asset`, а переводы — внутри параметров с соответствующим языковыми кодами.

> Параметр `common` внутри данного нода неприменим.

sizes

Тип: список

В этом ноде определяются различные размеры, которые можно использовать внутри сценария и в других параметрах, принимающих размеры.

В основном размеры используются для масштабирования спрайтов персонажей.

Размеры задаются двумя параметрами: шириной и высотой, разделёнными пробелом:

```
# resources.yaml
---
sizes:
  normal: 1 1 # определение размера "normal"
```

Также каждый размер может содержать внутри себя список положений, которые будут использоваться спрайтом персонажа в игре, если у него будут совпадать название размера и название позиции:

```
# resources.yaml
---
positions:
  center: 0 -0.59 # определение позиции "center"

sizes:
  close:
    value: 1.95 1.95 # размер указывается в параметре "value"
    positions:
      center: 0 -1.57 # переопределение положения "center" для размера "close"
```

```
# также вместо X-координаты можно прописать точку:
# center: . -1.57
# в таком случае значение X будет взято из исходного положения (0 в данном случае)
```

Значение размера указывается внутри параметра `value`*, а позиции — внутри списка* `positions`*.*

На примере выше мы определили размер `close`, при котором спрайт персонажа становится почти в 2 раза больше. Т.к. размеры спрайта изменились, значение для положения `center` необходимо немного подкорректировать, чтобы спрайт корректно отображался в игре. Для этого мы переопределили значение `center` при размере `close`, немного сместив его вниз.

В итоге спрайт в игре при размере `close` и положении `center` будет выглядеть так:



А не так:


1-104

spritecolor

Тип: список

В этом ноде определяются overlay-цвета для спрайтов персонажей. Эти цвета нужны для более корректного отображения спрайтов на различных фонах — дневных, ночных, закатных и т.д.

Цвета определяются следующим образом:

```
# resources.yaml
---
spritecolor:
  tokyo_street_night: "#E9F6FF"
  tokyo_street_sunset: "#FFF1ED"
```

Значением цвета может быть также и ID цвета, определённого в ноде colors.

Название цвета может быть любым (при необходимости вы сможете выбрать нужный цвет в сценарии вручную), однако удобнее всего использовать в качестве названий ID конкретных фонов, т.к. игра автоматически меняет цвета спрайтам на цвет, соответствующий названию активного фона, когда происходит смена фонов.



Спрайт на ночном фоне без overlay-цвета.



Тот же спрайт, но уже с цветом.

sound

Тип: список
Тип принимаемых ассетов: `sound`

В этом списке определяются звуковые файлы, которые можно использовать в сценариях.

Поддерживаются все форматы, распознаваемые Unity:

- mp3
- ogg (Vorbis)
- wav
- aiff / aif
- mod
- it
- s3m
- xm

> Чтобы звуковой файл распознался, он должен иметь корректное расширение.

Звуки определяются следующим образом:

```
# resources.yaml
---
sound:
  echo_avenue_old: sound/echo_avenue_old.wav
  into_the_uknown: sound/into_the_uknown.mp3
```

transitions

Тип: список

Список различных пресетов для переходов.
Переходы бывают нескольких видов:

- `background` — переходы фонов
- `character` — переходы спрайтов персонажей
- `screen` — переходы всего экрана
- `pattern` — подвид переходов фонов

В `backgroud`, `character` и `screen` прописываются пресеты переходов, аналогично тому, как они прописываются в файле сценария:

```
# resources.yaml
---
transitions:
  background:
    dissolve2sec: dissolve 2 # dissolve длительностью в 2 секунды
  character:
    crossfade2sec: crossfade 2 2 # crossfade с fadeOut = 2 секундам и fadeIn = 2 секундам
    fade212: fade 2 1 2 # fade с fadeIn = 1, hold = 1 и fadeOut = 2
  screen:
    fadeRed: fade color:#FF0000 # fade с красным цветом
    hpunch2: hpunch 2 3 10 0 true # горизонтальный punch с duration = 2, strength = 3, vibrato = 10,
randomness = 0 и fadeOut = true
```

В `pattern` же прописываются ассеты изображений для переходов фонов типа `pattern`:

```
# resources.yaml
---
transitions:
```

```
  pattern:
    chainsaw_pattern: images/patterns/chainsaw_pattern.jpg # определение изображения паттерна chainsaw_pattern
  background:
    chainsaw_pattern: pattern chainsaw_pattern 1.5 # определение перехода chainsaw_pattern длительностью в 1.5 секунды с использованием определенного выше изображения паттерна
```

///

Со списком доступных переходов и их параметрами можно ознакомиться здесь.

variables

Тип: список

В этом ноде определяются глобальные переменные мода, т.е., переменные, доступные во всех сценариях мода, и значения которых сохраняются при переходах между сценариями.

Переменные определяются следующим образом:

```
# resources.yaml
---
variables:
  himitsu_points: 0 # определение переменной himitsu_points со значением "0" по умолчанию
  day1_choice_made: false # переменная day1_choice_made со значением "false" по умолчанию
```

> Значения глобальных переменных могут содержать числа, текст, булевые значения (true/false), а также различные алгебраические операции.

> Значения определяемых глобальных переменных не могут содержать в себе другие переменные.