

ikace вчера в 18:47

Pyxel для любителей ретро игр

Python *

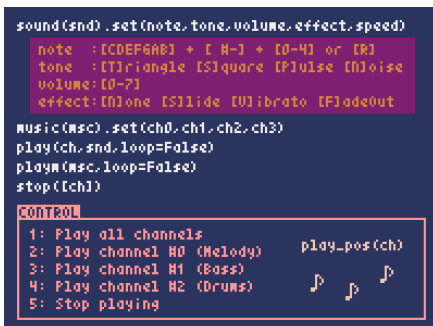
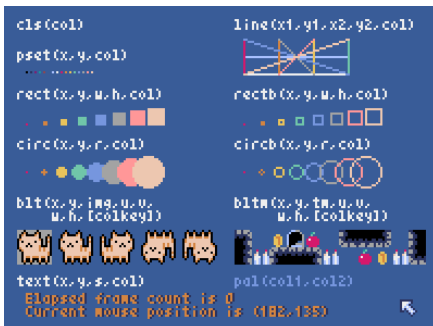
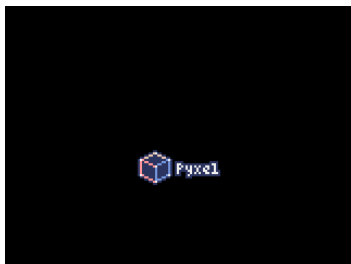
Из песочницы

Перевод

Автор оригинала: Takashi Kitao

Pyxel — это игровой движок для Python в стиле ретро.

Благодаря своей простоте, вдохновленной старыми игровыми консолями (например, палитра состоит всего из 16 цветов, и только 4 звука могут быть проиграны одновременно), вы можете легко создавать игры в стиле пиксель-арт.



ЧИТАЮТ СЕЙЧАС

Антибиотикорезистентность: ура, мы дождались! Читайте, что вышел анонс следующей пандемии

👁 40K 💬 320 +320

Эпоха красивого кода прошла. Пришло время быдлокода

👁 76K 💬 363 +363

Как я потеряла 1 миллион рублей на Wildberries из-за своей ошибки

👁 32K 💬 113 +113

Поиск Google умирает

👁 97K 💬 408 +408

Telegram позволяет узнавать координаты людей с точностью до метра

👁 36K 💬 74 +74

Open Source vs «коробки»: что думают хабровчане

Meranoct

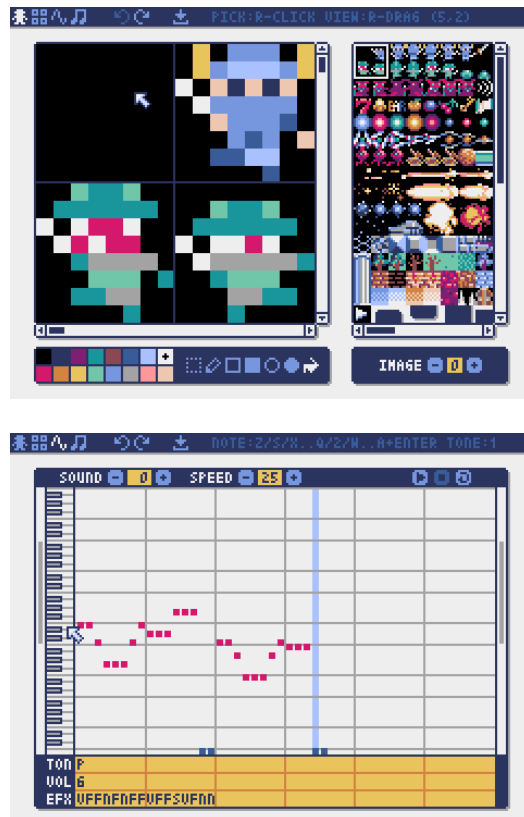
РАБОТА

Django разработчик
117 вакансий

Python разработчик
244 вакансии

Data Scientist
146 вакансий

Все вакансии



Спецификации и API Ruxel вдохновлены [PICO-8](#) и [TIC-80](#).

Ruxel — программа с открытым кодом и бесплатна для использования. За дело!

Характеристики

- Запускается на Windows, Mac и Linux
- Код пишется на Python
- 16-цветная палитра
- 3 набора изображений 256x256 пикселей
- 8 тайлмапов 256x256 пикселей
- 4 канала с 64 определяемыми пользователем звуками
- 8 музыкальных композиций
- Ввод с клавиатуры, мышки или игрового контроллера
- Редактор изображений и звука

Цветовая Палитра

0	000000	1	82838F	2	87E2072	3	819859C
	0.0.0		43.51.95		126.32.114		25.149.156
4	8B84B52	5	8395C88	6	8A8C1FF	7	8EEEEE
	139.72.82		57.92.152		169.193.255		238.238.238
8	8D41B6C	9	8D38441	10	8E9C35B	11	870C6A9
	212.24.108		211.132.65		233.195.91		112.198.169
12	87696DE	13	8A3A3A3	14	8FF979B	15	8EDC780
	118.150.222		163.163.163		255.151.152		237.199.176



Как установить

Предоставляется два варианта Ruxel, в виде пакета и в виде автономной версии.

Установка сборки в виде пакета

Версия Pyxel в виде пакета представляет собой модуль расширения для Python.

Рекомендуется знакомым с управлением пакетами Python с помощью команды `pip` и разрабатывающим полноценные приложения на Python.

Windows

После установки [Python3](#) (версии 3.7 или выше) необходимо выполнить следующую команду:

```
pip install -U pyxel
```

Mac

После установки [Python3](#) (версии 3.7 или выше) необходимо выполнить следующую команду:

```
pip3 install -U pyxel
```

Linux

После установки пакета SDL2 (`libsdl2-dev` для Ubuntu), [Python3](#) (версии 3.7 или выше) и `python3-pip` выполните следующую команду:

```
sudo pip3 install -U pyxel
```

Если приведённый выше способ установки не работает, вы можете собрать пакет Pyxel самостоятельно, установив `cmake` и `rust` и затем выполнив следующую последовательность команд:

```
git clone https://github.com/kitao/pyxel.git
cd pyxel
make clean all
sudo pip3 install .
```

Установка автономной версии

Автономная версия Pyxel представляет собой самостоятельное приложение, не зависящее от Python.

Рекомендуется желающим сразу начать писать код, не отвлекаясь на установку и настройку Python, а также тем, кто непосредственно хочет запускать игры.

Windows

Необходимо скачать и запустить последнюю версию установщика для Windows (`pyxel-[version]-windows-setup.exe`) со [страницы загрузки](#).

Mac

После установки [Homebrew](#) необходимо выполнить следующую последовательность команд:

```
brew tap kitao/pyxel
brew install pyxel
```

Linux

После установки пакета SDL2 (`libsdl2-dev` для Ubuntu) и установки [Homebrew](#) необходимо выполнить следующую последовательность команд:

```
brew tap kitao/pyxel
brew install pyxel
```

Если приведённый выше способ установки не работает, вы можете попробовать собрать пакет

Pyxel самостоятельно.

Попробуйте примеры

После установки Pyxel, примеры Pyxel будут скопированы в открытую директорию по выполнении этой команды:

```
pyxel copy_examples
```

Список примеров, которые будут скопированы:

- [01_hello_pyxel.py](#) - Простейшее приложение
- [02_jump_game.py](#) - Игра прыжков с простейшими ресурсными файлами Pyxel
- [03_draw_api.py](#) - Демонстрация API для рисования
- [04_sound_api.py](#) - Демонстрация API для работы со звуком
- [05_color_palette.py](#) - Цветовая палитра
- [06_click_game.py](#) - Игра с кликами мышкой
- [07_snake.py](#) - Змейка с BGM
- [08_triangle_api.py](#) - Демонстрация API по рисованию треугольных полигонов
- [09_shooter.py](#) - Игра жанра «убей всех» с переходом между экранами
- [10_platformer.py](#) - Платформер с боковым скроллингом и картой
- [11_offscreen.py](#) - Внеэкранный рендеринг с помощью класса Image
- [30SecondsOfDaylight.pyxapp](#) - 1-я победная игра Pyxel Jam от [Adam](#)
- [megaball.pyxapp](#) - Аркадная игра с физикой мяча от [Adam](#)

Эти примеры могут быть запущены следующей командой:

```
cd pyxel_examples
pyxel run 01_hello_pyxel.py
pyxel play 30SecondsOfDaylight.pyxapp
```

Как использовать Pyxel

Создание Pyxel-приложения

После импортирования модуля Pyxel в ваш код на Python, сначала укажите размер окна с помощью команды `init`, затем запустите Pyxel-приложение с помощью функции `run`.

```
import pyxel

pyxel.init(160, 120)

def update():
    if pyxel.btnp(pyxel.KEY_Q):
        pyxel.quit()

def draw():
    pyxel.cls(0)
    pyxel.rect(10, 10, 20, 20, 11)

pyxel.run(update, draw)
```

Аргументы функции `run` -- это функции `update` для обновления внутренней игровой логики каждый кадр и функции `draw` для отображения объектов на экране по мере необходимости.

В самом приложении рекомендуется свернуть код Pyxel в один класс (смотрите пример).

```
import pyxel

class App:
    def __init__(self):
        pyxel.init(160, 120)
        self.x = 0
        pyxel.run(self.update, self.draw)
```

```

def update(self):
    self.x = (self.x + 1) % pyxel.width

def draw(self):
    pyxel.cls(0)
    pyxel.rect(self.x, 0, 8, 8, 9)

App()

```

Можно также писать простые программки, используя функции `show` и `flip` для отображения простейшей графики и анимаций.

Функция `show` выводит изображение на экран и ждет нажатия клавиши `ESC`.

```

import pyxel

pyxel.init(120, 120)
pyxel.cls(1)
pyxel.circb(60, 60, 40, 7)
pyxel.show()

```

Функция `flip` обновляет изображение на экране единожды.

```

import pyxel

pyxel.init(120, 80)

while True:
    pyxel.cls(3)
    pyxel.rectb(pyxel.frame_count % 160 - 40, 20, 40, 40, 7)
    pyxel.flip()

```

Запуск Pyxel-приложения

Созданный сценарий на Python может быть запущен путём выполнения следующей команды:

```

pyxel run ИМЯ_PYTHON_ФАЙЛА

```

При использовании Pyxel в виде пакета сценарий может быть выполнен как обычный код на Python:

```

cd pyxel_examples
python3 ИМЯ_PYTHON_ФАЙЛА

```

(Под Windows, набирайте `python` вместо `python3`)

Особые клавиши

Следующие особые клавиши можно применять во время выполнения Pyxel-приложения:

- `Esc` Выйти из приложения
- `Alt (Option) +1` Выполнить снимок экрана и сохранить его на рабочий стол
- `Alt (Option) +2` Начать захват экрана игры
- `Alt (Option) +3` Сохранить видео, полученное захватом экрана на рабочий стол (до 10 секунд)
- `Alt (Option) +0` Включить/выключить мониторинг производительности (fps, время на update, время на draw)
- `Alt (Option) +Enter` Войти/выйти из полноэкранного режима

Как создать ресурсный файл

Встроенный Pyxel Editor может создавать изображения и звуки, используемые в Pyxel-приложении.

Он запускается следующей командой:

```
pyxel edit [имя_ресурсного_файла]
```

Если указанный ресурсный файл (.pyxres) существует, то он будет загружен. В противном случае будет создан файл с указанным именем. Если имя файла пропущено, то используется стандартное имя `my_resource.pyxres`.

После запуска Pyxel Editor, можно переключаться между различными файлами способом drag-and-drop. Если данное действие произвести, зажав клавишу `Ctrl (Cmd)`, то будет загружен только редактируемый в этот момент тип ресурса (изображения/карта тайлов/звук/музыка). Это позволяет комбинировать несколько ресурсных файлов в один.

Созданный ресурсный файл может быть загружен в программу с помощью функции `load`.

Редактор Pyxel Editor оснащён следующими режимами редактирования.

Редактор изображений:

Режим редактирования наборов изображений.



Изображение (png/gif/jpeg) может быть загружено в выбранный набор путем перетаскивания png файла на экран редактора изображений.

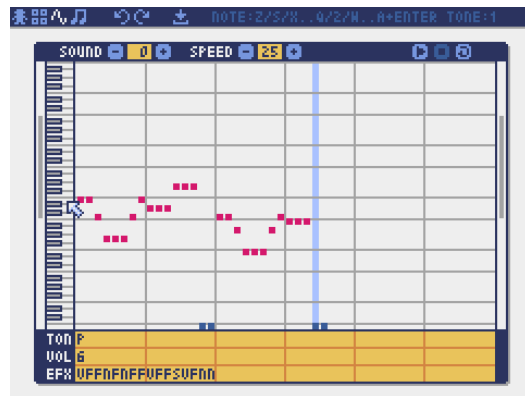
Редактор тайлмапов:

Режим редактирования тайлмапов, в котором изображения расположены в плиточном порядке.



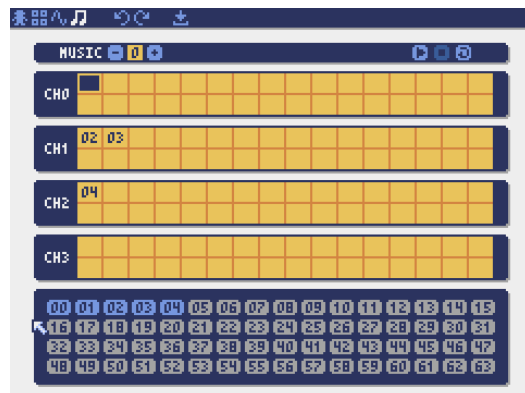
Редактор звука:

Режим для редактирования звуковых файлов.



Редактор музыки:

Режим для редактирования музыки, в которой звуки расставлены в порядке проигрывания.



Другие методы создания ресурсов

Изображения и карты тайлов Ruxel могут также быть созданы следующим образом:

- Создайте изображение из списка строк с помощью функций `Image.set` или `Tilemap.set`.
- Загрузите png файла, выполненный в палитре Ruxel, с помощью функции `Image.load`.

Звуки Ruxel могут также быть созданы следующим образом:

- Создайте звук из строк с помощью функций `Sound.set` или `Music.set`.

Обратитесь к руководству по API (ниже) для получения более подробной информации об использовании этих функций.

Как распространять приложение

Ruxel предлагает формат распространения приложений (файл Ruxel-приложения), работающий на всех поддерживаемых платформах.

Создать файл Ruxel-приложения (.рухарр) можно с помощью следующей команды:

```
ruxel package корневой_каталог_приложения имя_файл_запускающего_скрипта
```

Если приложение должно включать в себя дополнительные ресурсы или модули, поместите их в каталог приложения.

Созданный файл приложения может быть запущен следующей командой:

```
ruxel play ФАЙЛ_РУХЕЛ_ПРИЛОЖЕНИЯ
```

Руководство по API

Система

- `width`, `height`
Ширина и высота окна
- `frame_count`
Количество отрисованных кадров
- `init(width, height, [title], [fps], [quit_key], [capture_scale], [capture_sec])`
Инициализирует Pyxel-приложение с указанными размерами экрана (`width`, `height`). Дополнительно могут быть заданы: заголовок окна с помощью параметра `title`, количество кадров в секунду с помощью параметра `fps`, клавиша для выхода из приложения — `quit_key`, коэффициент масштабирования при захвате экрана — `capture_scale` и максимальное время записи при захвате экрана с помощью `capture_sec`.
Пример: `pyxel.init(160, 120, title="My Pyxel App", fps=60, quit_key=pyxel.KEY_NONE, capture_scale=3, capture_sec=0)`
- `run(update, draw)`
Запустить Pyxel-приложение, использующее функцию `update` для обновления внутренней логики и `draw` для рисования.
- `show()`
Отрисовать кадр и ждать выхода из приложения по нажатию клавиши `Esc` (не для использования в настоящих приложениях).
- `flip()`
Принудительно отрисовать кадр (не для использования в настоящих приложениях).
- `quit()`
Завершить работу Pyxel-приложения.

Ресурсы

- `load(filename, [image], [tilemap], [sound], [music])` Загрузить ресурсный файл (.рухрес). Если `False` указано для типа ресурса (`image/tilemap/sound/music`), соответствующий ресурс не будет загружен.

Ввод

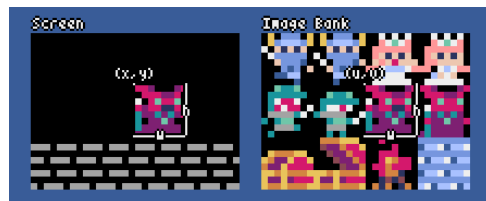
- `mouse_x`, `mouse_y`
Получить положение курсора мышки
- `mouse_wheel`
Получить значение колесика мышки
- `btn(клавиша)`
Получить `True`, если `клавиша` нажата, в противном случае получить `False`. ([Список определений клавиш](#))
- `btnp(клавиша, [hold], [period])`
Получить `True`, если `клавиша` нажата в данный кадр, в противном случае получить `False`. В случае, если указаны параметры `hold` и `period`, `True` будет возвращено каждые `period` кадров, когда `key` уже зажата более `hold` кадров
- `btnr(клавиша)`
Получить `True`, если `клавиша` была отпущена в данный кадр, в противном случае получить `False`
- `mouse(видна)`
Установить видимость курсора: если `visible` равно `True`, сделать видным, если `False`, то невидимым. Даже если курсор не отображается, его позицию всё равно можно получить соответствующими функциями.

Графика

- `colors`
Список цветов палитры. Цвет кодируется 24-битным целым числом. Используйте `colors.from_list` и `colors.to_list` для установки и получения списка Python.
Пример: `org_colors = pyxel.colors.to_list(); pyxel.colors[15] = 0x112233; pyxel.colors.from_list(org_colors)`
- `image(img, [system])`
Оперировать набором изображений `img` (0-2) (смотрите класс `Image`).
Пример: `pyxel.image(0).load(0, 0, "title.png")`
- `tilemap(tm)`
Оперировать таймпапом `tm` (0-7) (смотрите класс `Tilemap`)

Сторону экрана задается `clip(x, y, w, h)`; диаметр круга — `circ(x, y, r, col)`.

- `clip(x, y, w, h)`
Установить площадь рисования экрана с `(x, y)` до ширины `w` и высоты `h`. Сбросить площадь рисования до полного экрана можно с помощью `clip()`.
- `camera(x, y)`
Изменить координаты левого верхнего угла экрана на `(x, y)`. Координаты левого верхнего угла экрана могут быть сброшены в `(0, 0)` вызовом `camera()`.
- `pal(col1, col2)`
Поменять цвет `col1` с цветом `col2` во время рисования. Восстановить изначальную палитру можно с помощью `pal()`.
- `cls(col)`
Заполнить (очистить) экран цветом `col`.
- `pget(x, y)`
Получить цвет пикселя по координатам `(x, y)`.
- `pset(x, y, col)`
Нарисовать пиксель цвета `col` по координатам `(x, y)`.
- `line(x1, y1, x2, y2, col)`
Нарисовать отрезок цвета `col` из `(x1, y1)` в `(x2, y2)`.
- `rect(x, y, w, h, col)`
Нарисовать прямоугольник ширины, высоты `w` и цвета `h` по координатам `(x, y)`.
- `rectb(x, y, w, h, col)`
Нарисовать контур прямоугольника ширины, высоты `w` и цвета `h` по координатам `(x, y)`.
- `circ(x, y, r, col)`
Нарисовать круг радиуса `r` и цвета `col` центром в `(x, y)`.
- `circb(x, y, r, col)`
Нарисовать окружность радиуса `r` и цвета `col` центром в `(x, y)`.
- `elli(x, y, w, h, col)`
Нарисуйте эллипс шириной `w`, высотой `h` и цветом `col` из `(x, y)`.
- `ellib(x, y, w, h, col)`
Нарисуйте контур эллипса шириной `w`, высотой `h` и цветом `col` из `(x, y)`.
- `tri(x1, y1, x2, y2, x3, y3, col)`
Нарисовать треугольник с вершинами в координатах `(x1, y1)`, `(x2, y2)`, `(x3, y3)` и цвета `col`.
- `trib(x1, y1, x2, y2, x3, y3, col)`
Нарисовать контур треугольника с вершинами в координатах `(x1, y1)`, `(x2, y2)`, `(x3, y3)` и цвета `col`.
- `fill(x, y, col)`
Нарисуйте эллипс шириной `w`, высотой `h` и цветом `col` из `(x, y)`.
- `blt(x, y, img, u, v, w, h, [colkey])`
Скопировать область размеров `(w, h)`, по координатам `(u, v)` набора изображений `img` (0-2) по координатам `(x, y)` на экране. Если для `w` и/или `h` установлено отрицательное значение, изображение будет развернуто горизонтально и/или вертикально. Если указан параметр `colkey`, соответствующий цвет будет считаться цветом фона (прозрачным цветом).



- `bltm(x, y, tm, u, v, w, h, [colkey])` Нарисовать из тайлмэпа `tm` (0-7) по координатам `(x, y)` тайл размером `(w, h)`, находящийся по координатам `(u, v)`. Если переданы отрицательные значения `w` и/или `h`, то изображение будет отражено по горизонтали и/или вертикали. Если указан параметр `colkey`, соответствующий цвет будет считаться цветом фона (прозрачным цветом). Размер тайла равен 8x8 точек и хранится в карте тайлов в виде кортежа `(tile_x, tile_y)`.



- `text(x, y, s, col)` Нарисовать строку текста `s` цвета `col` по координате `(x, y)`

Аудио

- `sound(snd)`
Оперировать звуком `snd` (0-63).
Пример: `pyxel.sound(0).speed = 60`
- `music(msc)`
Оперировать музыкой `msc` (0-7) (смотрите класс Music)
- `play_pos(ch)`
Получить позицию канала `ch` (0-3) в виде кортежа (номер звука, номер ноты).
Возвращает `None` если проигрывание выключено.
- `play(ch, snd, [tick], [loop])`
Проиграть звук `snd` (0-63) на канале `ch` (0-3). Если `snd` — список, он будет проигран по порядку. Позиция начала воспроизведения может быть указана с помощью `tick` (1 tick = 1/120 секунды). Если в качестве значения `loop` передано `True`, проигрывание будет зациклено.
- `playm(msc, [tick], [loop])`
Проиграть трек `msc` (0-7). Позиция начала воспроизведения может быть указана с помощью `tick` (1 tick = 1/120 секунды). Если в качестве значения `loop` передано `True`, проигрывание будет зациклено.
- `stop([ch])`
Остановить воспроизведение на канале `ch` (0-3). `stop()` останавливает воспроизведение на всех каналах.

Математика

- `ceil(x)`
Возвращает наименьшее целое число, большее или равное `x`.
- `floor(x)`
Возвращает наибольшее целое число, меньшее или равное `x`.
- `sgn(x)`
Возвращает 1, если `x` положительно, 0, если оно равно нулю, и -1, если оно отрицательно.
- `sqrt(x)`
Возвращает квадратный корень из `x`.
- `sin(deg)`
Возвращает синус градуса `deg`.
- `cos(deg)`
Возвращает косинус градуса `deg`.
- `atan2(y, x)`
Возвращает арктангенс угла `y / x` в градусах.
- `rseed(seed: int)`
Устанавливает заправку генератора случайных чисел.
- `rndi(a, b)`
Возвращает случайное целое число, большее или равное `a` и меньшее или равное `b`.
- `rndf(a, b)`
Возвращает случайную десятичную дробь, большую или равную `a` и меньшую или равную `b`.
- `nseed(seed)`
Устанавливает семя шума Перлина.
- `noise(x, [y], [z])`
Возвращает значение шума Перлина для указанных координат.

Класс Image

- `width , height`
Ширина и высота изображения
- `data`
Данные изображения (матрица 256x256)
- `get(x, y)`
Получить данные изображения в точке (`x` , `y`)
- `set(x, y, data)`
Установить данные изображения в точке (`x` , `y`) списком строк.
Пример: `pyxel.image(0).set(10, 10, ["0123", "4567", "89ab", "cdef"])`
- `load(x, y, filename)`
Загрузить файл изображения (png/gif/jpeg) в координаты (`x` , `y`).

Класс Tilemap

- `width , height`
Ширина и высота тайлмапа
- `refimg`
Банк изображений (0-2), на который ссылается карта тайлов
- `set(x, y, data)`
Установить данные карты тайлов в точке (`x` , `y`) списком строк.
Пример: `pyxel.tilemap(0).set(0, 0, ["000102", "202122", "a0a1a2", "b0b1b2"])`
- `pget(x, y)`
Получить тайл в координатах (`x` , `y`). Возвращаемое значение представляет собой кортеж (`tile_x`, `tile_y`).
- `pset(x, y, tile)`
Задать тайл в координатах (`x` , `y`). Тайл передаётся в виде кортежа (`tile_x`, `tile_y`).

Класс Sound

- `notes`
Список нот (0-127). Чем больше значение, тем выше нота. Значение 33 соответствует ноте «ля» второй октавы 'A2' (440Hz). Пауза задаётся значением -1.
- `tones`
Список тонов (0:Треугольник / 1:Квадрат / 2:Пульс / 3:Шум)
- `volumes`
Список громкости(0-7)
- `effects`
Список эффектов (0:Нет / 1:Слайд / 2:Вибрато / 3:Затухание)
- `speed`
Длительность воспроизведения. 1 — самая быстрая, чем выше значение, тем ниже скорость воспроизведения. При значении, равном 120 длительность воспроизведения одной ноты составляет 1 секунду.
- `set(notes, tones, volumes, effects, speed)`
Установить ноты, тоны, громкость и эффекты с помощью строк. Если длины строк для тона, громкости и эффектов короче строки для нот, они зацикливаются.
- `set_notes(notes)`
Установить ноты с помощью строки, составленной по форме 'CDEFGAB'+#+'+0123' или 'R'. Регистр и пробелы игнорируются.
Пример: `pyxel.sound(0).set_note("G2B-2D3R RF3F3F3")`
- `set_tones(tones)`
Установить тоны строкой, составленной из 'TSPN'. Регистр и пробелы игнорируются.
Пример: `pyxel.sound(0).set_tone("TTSS PPN")`
- `set_volumes(volumes)`
Установить громкость с помощью строки, составленной из '01234567'. Регистр и пробелы игнорируются.
Пример: `pyxel.sound(0).set_volume("7777 7531")`
- `set_effects(effects)`
Установить эффекты с помощью строки, составленной из 'NSVF'. Регистр и пробелы игнорируются.
Пример: `pyxel.sound(0).set_effect("NFNF NVVS")`

Класс Music

- `sequences`

Двумерный список описаний звуков (0-63) по каналам

- `set(seq0, seq1, seq2, seq3)`

Установить список звуков (0-63) для всех каналов. Пустой список означает, что канал не используется для проигрывания.

Пример: `pyxel.music(0).set([0, 1], [2, 3], [4], [])`

Расширенный APIs

Pyxel имеет «расширенные API», не упомянутые в этом документе, так как они «могут смутить пользователя» или «требуют специальных знаний для использования».

Если вы уверены в своих силах, используйте [это](#) в качестве подсказки!

Как сделать вклад в развитие проекта?

Сообщение о проблемах

Используйте [трекер проблем](#) для отправки отчётов о проблемах или предложений по улучшению/добавлению новых возможностей. Перед созданием новой задачи, убедитесь что схожие открытые задачи отсутствуют.

Ручное тестирование

Ручное тестирование кода и написание отчетов о проблемах, предложений по улучшению в [трекере проблем](#) приветствуется!

Опубликование запроса на слияние

Патчи/фиксы принимаются в форме запросов на слияние (pull-запрос, PR). Убедитесь, что проблема, к которой относится запрос на слияние изменений, открыта в трекере проблем.

Опубликованный pull-запрос считается опубликованным под лицензией [MIT License](#).

Прочая информация

- [Q&A](#)
- [User Examples](#)
- [Discord Server \(English\)](#)
- [Discord Server \(Japanese - 日本語版\)](#)

Лицензия

Pyxel распространяется по лицензией [MIT License](#). Он может быть использован в проприетарном программном обеспечении при условии того, что все копии этого программного обеспечения или значительные его части содержат копию MIT License terms and the copyright notice.

Набор Спонсоров

Pyxel ищет спонсоров на GitHub Sponsors. Рассмотрите возможность спонсирования Pyxel для продолжения обслуживания и добавления функций. Спонсоры могут проконсультироваться о Pyxel в качестве преимущества. Подробнее см. [Здесь](#).

Теги: [pyxel](#), [python](#), [игры](#)

Хабы: [Python](#)

Редакторский дайджест

Присылаем лучшие статьи раз в месяц



5

7.6

Карма Рейтинг

@lkace

Пользователь

ПОХОЖИЕ ПУБЛИКАЦИИ

вчера в 10:53

Шрифты в играх: (почти) идеальные засечки, кернинги и иероглифы

+7 1.1K 20 0

вчера в 08:48

Моя первая игра — Picross.io

+1 1.6K 8 6 +6

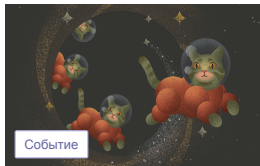
14 января в 15:10

Как решить популярную в 2022 головоломку Wordle на Python

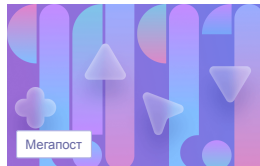
+9 11K 41 6 +6

МИНУТОЧКУ ВНИМАНИЯ

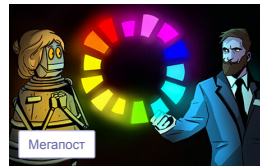
[Разместить](#)



Событие
**Конкурс технических статей
Технотекст 2021**



Мегапост
**Зарплаты айтишников во втором
полугодии 2021**



Мегапост
**Почему программисты пишут
статьи**

ВОПРОСЫ И ОТВЕТЫ

Как одновременно читать и записывать файл в Python?

Python · Простой · 0 ответов

Как определить язык пользователя в Python?

Python · Простой · 2 ответа

Как написать бота для телеграм с командами через /?

Python · Простой · 2 ответа

Как можно убрать лишние строки в TXT?

Python · Простой · 1 ответ

Python Telethon, как вызвать get_participants в приватном канале?

Python · Средний · 1 ответ

[Больше вопросов на Хабр Q&A](#)

ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

вчера в 13:01

Антибиотикорезистентность: ура, мы дождались! Читайте, что вышел анонс следующей пандемии

+210 40K 96 319 +319

вчера в 11:12

Как я потеряла 1 миллион рублей на Wildberries из-за своей ошибки

+117 32K 46 113 +113

вчера в 20:32

Как я винду по-реверсерски чинил

+74 5.8K 40 36 +36

вчера в 13:00


Миллионы рублей за 1/100 секунды

Материалы рубрики «IT-тусовки»

 +61  11K  88  40 +40

вчера в 21:53

18 февраля — День памяти Криса Касперски. Пять лет как нет Николая Лихачева

 +48  7.4K  26  15 +15

А как прошел ваш 2021? Исследуем рынок труда в IT, строим тренды и ищем аномалии

Промо 

Ваш аккаунт

[Войти](#)

[Регистрация](#)

Разделы

[Публикации](#)

[Новости](#)

[Хабы](#)

[Компании](#)

[Авторы](#)

[Песочница](#)

Информация

[Устройство сайта](#)

[Для авторов](#)

[Для компаний](#)

[Документы](#)

[Соглашение](#)

[Конфиденциальность](#)

Услуги

[Реклама](#)

[Тарифы](#)

[Контент](#)

[Семинары](#)

[Мегaproекты](#)

© 2006–2022 «Habr»

[Вернуться на старую версию](#)

[Техническая поддержка](#)

[О сайте](#)

[Настройка языка](#)

