Оглавление

[Инициализация 3](#_Toc68034189)

[H 3](#_Toc68034190)

[Сайт: 3](#_Toc68034191)

[Main.py: 3](#_Toc68034192)

[Что в итоге видит пользователь: 3](#_Toc68034193)

[E 3](#_Toc68034194)

[Сайт: 3](#_Toc68034195)

[Main.py: 3](#_Toc68034196)

[Что в итоге видит пользователь: 4](#_Toc68034197)

[L3 4](#_Toc68034198)

[Сайт: 4](#_Toc68034199)

[Main.py: 4](#_Toc68034200)

[Что в итоге видит пользователь: 4](#_Toc68034201)

[L 4](#_Toc68034202)

[Сайт: 4](#_Toc68034203)

[Main.py: 4](#_Toc68034204)

[Что в итоге видит пользователь: 4](#_Toc68034205)

[O2 5](#_Toc68034206)

[Сайт: 5](#_Toc68034207)

[Main.py: 5](#_Toc68034208)

[Что в итоге видит пользователь: 5](#_Toc68034209)

[SPACE 5](#_Toc68034210)

[Сайт: 5](#_Toc68034211)

[Main.py: 5](#_Toc68034212)

[Что в итоге видит пользователь: 5](#_Toc68034213)

[W 5](#_Toc68034214)

[Сайт: 5](#_Toc68034215)

[Main.py: 5](#_Toc68034216)

[Что в итоге видит пользователь: 5](#_Toc68034217)

[O 6](#_Toc68034218)

[Сайт: 6](#_Toc68034219)

[Main.py: 6](#_Toc68034220)

[Что в итоге видит пользователь: 6](#_Toc68034221)

[R 6](#_Toc68034222)

[Сайт: 6](#_Toc68034223)

[Main.py: 6](#_Toc68034224)

[Что в итоге видит пользователь: 6](#_Toc68034225)

[L2 7](#_Toc68034226)

[Сайт: 7](#_Toc68034227)

[Main.py: 7](#_Toc68034228)

[Что в итоге видит пользователь: 7](#_Toc68034229)

[D 7](#_Toc68034230)

[Сайт: 7](#_Toc68034231)

[Main.py: 7](#_Toc68034232)

[Что в итоге ~~видит~~ слышит пользователь: 7](#_Toc68034233)

[exc 7](#_Toc68034234)

[Сайт: 7](#_Toc68034235)

[Main.py: 8](#_Toc68034236)

[Что в итоге видит пользователь: 8](#_Toc68034237)

# Инициализация

Сначала запускается main.py, который открывает локальный сервер в папке с сайтами, считывает настройки из файла config/settings.txt, преобразовывая большую их часть в понятные Питону вызовы функций, а также извлекая из них предпочитаемую пользователем конфигурацию фоновой музыки.

Далее программа начинает обход сайтов в указанном в настройках порядке, выполняя на каждом сайте соответствующую (опять же, указанную в настройках) функцию. Предпочитаемый лично мной порядок выглядит так:

# H

## Сайт:

Main.js заполняет страницу кнопками (9 в высоту, 5 в ширину). Кнопки настраиваются так, что при нажатии становятся красными и издают звук.

## Main.py:

Программа находит все кнопки и выбирает из них те, которые находятся либо в левом столбце, либо в правом, либо в полоске, соединяющей эти столбцы. Затем она нажимает на эти кнопки, делая после каждого нажатия паузу, время которой задано аргументом.

## Что в итоге видит пользователь:

Пользователь видит, как автоматизированный браузер нажимает на кнопки, которые, становясь красными, образуют заглавную ‘H’.

# E

## Сайт:

Main.js заполняет страницу клетками (10 в высоту и в ширину). В зависимости от объектов, находящихся в клетке, та меняет цвет. Пустая клетка – цвет пшена, клетка со змеёй – зелёный, клетка же с яблоком выглядит не полностью заполненной, т.к. кроме красного в центре, там присутствуют границы, совпадающие по цвету с пустой клеткой. Изначальные координаты змеи и яблок предопределены. Управление змеёй происходит при помощи WASD. Другой конец змеи не двигается, пока присутствует «заказанная длина»: она изначально равна 2, за каждое съеденное яблоко она увеличивается на 3. Это имитирует рост змеи после поедания яблока. Змея не может съесть себя. При движении, при поедании и при попытке движения в недопустимое место воспроизводятся звуки.

## Main.py:

Программа считывает карту, путь к которой задан аргументом. На карте должны быть символы, начиная с ‘1’. Найдя единицу, программа будет искать позицию двойки и т.д. (цифры [1,2,3….9] -> большие английские буквы [A,B,C…] -> маленькие английские буквы [a,b,c…]), если какой-то символ не найден то программа считает, что путь окончен. Программа нажимает на клавиши WASD в зависимости от позиции следующего элемента относительно предыдущего (считается, что следующий элемент обязательно должен граничить с предыдущим). Таким образом, увидев карту [[1, 2],[4,3]], программа сделает следующие движения: вправо (1->2), вниз (2->3), влево (3->4). Между движениями происходит пауза, время которой задано аргументом.

## Что в итоге видит пользователь:

Т.к. в изначальной конфигурации дана карта формы ‘E’, пользователь видит, как автоматизированный браузер оптимально съедает все яблоки и ведёт змейку так, чтобы она образовала заглавную букву ‘E’.

# L3

## Сайт:

Страница заполняется canvas’ом, на который можно кликать. Происходит обработка клика, на его месте рисуется круг с чёрной обводкой шириной 5 и заливкой случайного цвета. Радиус круга случайный, зависящий от разрешения экрана (от 2% до 20% минимальной стороны экрана). При клике играется звук.

## Main.py:

Программа совершает клики в зоне, напоминающей ‘L’. Каждый третий клик приходится на правую часть этой буквы, а остальные – на левую. Совершается какое-то количество кликов (задано аргументом) с каким-то промежутком (задано аргументом).

## Что в итоге видит пользователь:

Пользователь видит, как постепенно из случайных кругов образуется фигура, напоминающая заглавную ‘L’.

# L

## Сайт:

Сайт состоит из двух элементов, левый имитирует левую часть ‘L’ (‘I’), а правый – правую (‘\_‘). В левом элементе помещается редактируемый текст, в который можно писать вертикально слева направо. В правом элементе тоже помещается редактируемый текст, однако в него можно писать только горизонтально слева направо. Main.js постоянно мониторит количество строк в этих текстовых объектах, и если строк слишком много (если следующая строка заставит текст быть слишком большим для отцовского элемента), то редактирование текста запрещается.

## Main.py:

Программа переходит на сайт, в котором перечислены все слова, начинающиеся на ‘L’. Из него она извлекает какое-то (задано аргументом) количество слов, после чего закрывает вкладку и возвращается на сайт буквы. Эти слова (сразу несколько, тоже задано аргументом) она вписывает в текстовые поля сайта, пока редактирование в этих полях разрешено, с определённым промежутком (задано аргументом).

## Что в итоге видит пользователь:

Пользователь сначала любуется верхней частью сайта, в котором перечислены слова на ‘L’, а потом видит, как эти самые слова вводятся в текстовые поля, образуя из словарного запаса вполне реальную заглавную ‘L’.

# O2

## Сайт:

Сайт представляет из себя (сверху) текст с некоторыми редактируемыми элементами и кнопку, а также (снизу) canvas. Текст хранит информацию о квадратной функции, где коэффициенты настраиваемы. Кнопка заставляет canvas изобразить график описанной функции. Координаты по горизонтали (x) идут от -100 к 100, а по вертикали (y) от 100 (верх сайта) до 0. При нажатии на кнопку происходит обработка коэффициентов, затем происходит итерация по ширине страницы, на каждый пиксель в ширину рисуется квадрат (5\*5) в точке, согласно описанной квадратной функции. Кроме того, при нажатии на кнопку играется звук.

## Main.py:

Программа (не забывая перед этим при помощи backspace удалять уже введённые символы) вводит в редактируемые текстовые поля данные, заданные аргументом. После каждого ввода она нажимает на кнопку.

## Что в итоге видит пользователь:

Пользователь видит две параболы, которые напоминают (особенно если смотреть на пересечение их внутренних частей) заглавную ‘O’.

# SPACE

## Сайт:

Загружается .gif перекати-поле, который центрируется.

## Main.py:

Программа не делает совершенно ничего.

## Что в итоге видит пользователь:

Пользователь видит мучительно долго катящееся в одной точке перекати-поле, что должно, по идее, передать пробел как отсутствие чего-то, конкретно здесь – отсутствие действия.

# W

## Сайт:

При помощи three.js создаётся 3d сцена, рендерер и камера. Загружается шрифт и создаётся большая буква ‘W’, которая располагается перед камерой. При нажатии на ‘A’ или ‘D’ происходит установка положительной или отрицательной скорости вращения (по часовой или против часовой). Когда происходит событие анимации, эта скорость анализируется и происходит само вращение. Когда буква совершила пол-оборота (90 градусов) вправо или влево, дальнейшее вращение блокируется, а специальный индикатор, созданный до этого, меняет название своего класса, давая понять об истощении функционала сайта.

## Main.py:

Программа находит индикатор, и пока тот не изменит свой класс, она симулирует удержание ‘A’ или ‘D’ в зависимости от того, что передано в аргументе (‘left’ или ‘right’).

## Что в итоге видит пользователь:

Пользователь видит, как в его строну постепенно поворачивается заглавная буква ‘W’.

# O

## Сайт:

Страница заполняется canvas’ом, на который можно кликать. В зависимости от чётного клика либо запоминается точка клика, либо строится линия, соединяющая точку предыдущего клика с позицией этого. При каждом втором клике играется звук.

## Main.py:

Программа сначала совершает клик на позицию, отклоняющуюся вправо от центра на величину радиуса (задано аргументом). Затем из этой точки она начинает вращаться по окружности, перемещаясь на случайное количество градусов (мин. и макс. перемещения заданы в аргументах). После каждого перемещения, программа совершает два клика на позицию, отклоняющуюся от центра на умноженные на величину радиуса косинус текущего угла (по горизонтали) и синус текущего угла (по вертикали). Величина радиуса после каждого перемещения (кроме последнего) терпит незначительные изменения (это тоже конфигурируемо аргументом).

## Что в итоге видит пользователь:

Пользователь видит, как на его глазах рисуется достаточно неровная заглавная ‘O’.

# R

## Сайт:

Сайт заполняется клетками (10 в ширину и в длину), каждая по умолчанию является пустой клеткой (цвет пшена). В Main.js указаны все возможные фигуры и порядок их появления на координатах (2,0). Если фигуры ещё нет, либо если прошлая столкнулась с чем-то, появляется новая, которой можно управлять при помощи ASD. После движении фигуры проверяются клетки под ней, чтобы понять, приземлилась ли она. Проверка клеток происходит при помощи системы «шаблонов»: иногда нас могут интересовать только клетки, которые войдут в фигуру, если мы её переместим, поэтому кроме новых координат мы отправляет в функции проверки «шаблон» фигуры. Чтобы предотвратить столкновение с другой фигурой при движении в сторону, всегда происходит проверка, пересечётся ли фигура с другой если мы поместим её на новые координаты, если же это действительно произойдёт: ход блокируется. При движении и приземлении играется звук. После того, как последняя фигура приземлилась (или же если новая фигура просто не может появиться), индикатор, заранее созданный, меняет название своего класса, давая понять об истощении функционала сайта.

## Main.py:

Программа считывает карту, путь к которой передан аргументом. На этой карте в каждой строке слева написана координата по горизонтали, а справа – разделённые запятыми числа, являющиеся номерами фигур, которые программа попытается расположить на соответствующих координатах (к примеру, ‘1:2,3’ значит, что программа попытается расположить фигуру#2 и фигуру#3 на столбце#1) (отсчёт с нуля).

## Что в итоге видит пользователь:

Пользователь видит, как автоматизированный браузер, следуя изначальным конфигурациям, располагает тетрисные фигуры таким образом, чтобы они составили заглавную ‘R’.

# L2

## Сайт:

Страница заполняется canvas’ом, на который можно кликать. В зависимости от чётности клика (если, конечно, клик происходит внутри ‘L’) либо запоминается точка клика, либо создаётся объект луча, базирующийся в точке первого клика и направленный в сторону второго клика. Каждый tick (0.035 секунды) происходит обработка всех лучей, происходит их перемещение. Если они сталкиваются с границами воображаемой L (левая палка: ширина 20% экрана, высота – 100% экрана; правая палка: ширина 20% экрана, высота: 20% экрана), то по знаку меняется либо X, либо Y вектора их направления (в зависимости от того, с какой из границ они столкнулись). При столкновении играется звук.

## Main.py:

Программа анализирует полученные в качестве аргумента инструкции. На каждую инструкцию она совершает два клика: первый в позицию, координаты которой переданы в инструкции (координаты идут от 0 до 100), а второй отклоняется от этой позиции на умноженные на величину радиуса косинус угла (по горизонтали) и синус угла (по вертикали).

## Что в итоге видит пользователь:

Пользователь видит, как лучи проходят определённый путь, постоянно меняя своё направление из-за столкновений с невидимыми препятствиями, подозрительно напоминающими границы заглавной ‘L’.

# D

## Сайт:

На странице присутствуют элементы, изображающие ноты на клавиатуре фортепиано. Main.js анализирует элементы и даёт им уникальные id в зависимости от октавы и от их обозначения. Кроме того, main.js имеет таблицу, которая может конвертировать id элементов в частоты, которые изображают реальную ноту конкретного элемента. При нажатии на эти элементы играется как раз эта частота, ноты достаточно долгое время звучат.

## Main.py:

Сначала программа меняет громкость фоновой музыки согласно переданному аргументу (она вернёт громкость после выполнения функции). Затем она находит все клавиши и, в соответствии с переданными инструкциями, нажимает на них, различая их по уникальным id, или делает паузу.

## Что в итоге ~~видит~~ слышит пользователь:

Пользователь в данном случае не видит, а слышит аккорд D минор (Dm).

# SPACE2

## Сайт:

На странице присутствуют элементы, изображающие ноты на клавиатуре фортепиано. Main.js анализирует элементы и даёт им уникальные id в зависимости от октавы и от их обозначения. Кроме того, main.js имеет таблицу, которая может конвертировать id элементов в частоты, которые изображают реальную ноту конкретного элемента. При нажатии на эти элементы играется как раз эта частота, ноты достаточно долгое время звучат.

## Main.py:

Сначала программа меняет громкость фоновой музыки согласно переданному аргументу (она вернёт громкость после выполнения функции). Затем она находит все клавиши и, в соответствии с переданными инструкциями, нажимает на них, различая их по уникальным id, или делает паузу.

## Что в итоге ~~видит~~ слышит пользователь:

Пользователь в данном случае не видит, а слышит аккорд D минор (Dm).

# n2

## Сайт:

На странице присутствуют элементы, изображающие ноты на клавиатуре фортепиано. Main.js анализирует элементы и даёт им уникальные id в зависимости от октавы и от их обозначения. Кроме того, main.js имеет таблицу, которая может конвертировать id элементов в частоты, которые изображают реальную ноту конкретного элемента. При нажатии на эти элементы играется как раз эта частота, ноты достаточно долгое время звучат.

## Main.py:

Сначала программа меняет громкость фоновой музыки согласно переданному аргументу (она вернёт громкость после выполнения функции). Затем она находит все клавиши и, в соответствии с переданными инструкциями, нажимает на них, различая их по уникальным id, или делает паузу.

## Что в итоге ~~видит~~ слышит пользователь:

Пользователь в данном случае не видит, а слышит аккорд D минор (Dm).

# n0

## Сайт:

На странице присутствуют элементы, изображающие ноты на клавиатуре фортепиано. Main.js анализирует элементы и даёт им уникальные id в зависимости от октавы и от их обозначения. Кроме того, main.js имеет таблицу, которая может конвертировать id элементов в частоты, которые изображают реальную ноту конкретного элемента. При нажатии на эти элементы играется как раз эта частота, ноты достаточно долгое время звучат.

## Main.py:

Сначала программа меняет громкость фоновой музыки согласно переданному аргументу (она вернёт громкость после выполнения функции). Затем она находит все клавиши и, в соответствии с переданными инструкциями, нажимает на них, различая их по уникальным id, или делает паузу.

## Что в итоге ~~видит~~ слышит пользователь:

Пользователь в данном случае не видит, а слышит аккорд D минор (Dm).

# n2(повтор)

# n1

## Сайт:

На странице присутствуют элементы, изображающие ноты на клавиатуре фортепиано. Main.js анализирует элементы и даёт им уникальные id в зависимости от октавы и от их обозначения. Кроме того, main.js имеет таблицу, которая может конвертировать id элементов в частоты, которые изображают реальную ноту конкретного элемента. При нажатии на эти элементы играется как раз эта частота, ноты достаточно долгое время звучат.

## Main.py:

Сначала программа меняет громкость фоновой музыки согласно переданному аргументу (она вернёт громкость после выполнения функции). Затем она находит все клавиши и, в соответствии с переданными инструкциями, нажимает на них, различая их по уникальным id, или делает паузу.

## Что в итоге ~~видит~~ слышит пользователь:

Пользователь в данном случае не видит, а слышит аккорд D минор (Dm).

# exc

## Сайт:

Страница заполняется пустыми клетками (6 в ширину и высоту) – цвет пшена.

## Main.py:

Сначала программа находит все клетки, выделяя из них только те, которые соответствуют фигуре восклицательного знака. Затем она находит путь к папке этой буквы и перебрасывает туда скриншоты, сделанные в течение сессии. После этого она добавляет в интересующие клетки элементы <img>, ссылающиеся на скриншоты, которые уже лежат в папке буквы.

## Что в итоге видит пользователь:

Пользователь видит, как скриншоты, сделанные в ходе этой сессии, образуют собой восклицательный знак.