Описание

Программа может работать в следующих режимах:

- запись
- декомпозиция
- воспроизведение
- создание сценариев

Файл с настройками лежит в файле ./looper.config

Краткая справка по CLI

```
# Общий формат
python looper.py [режим] [действие] [параметры]

# Основные режимы:
python looper.py --record|-r <action_name>                # Запись д
python looper.py --play|-p <action_name> [параметры]      # Воспроиз
python looper.py --decompose|-d <action_name>             # Декомпоз
python looper.py --scenario|-sc <action_name> [параметры] # Создан

# Помощь
python looper.py --help|-h                                # Показать
python looper.py --version|-v                             # Показать

# Примеры использования:
python looper.py -r open_notepad
python looper.py -d open_notepad
python looper.py -p open_notepad -f custom_actions.json
python looper.py -sc open_notepad -o my_scenario --delay 1.5
```

Запись

Пользователь вызывает программу в режиме записи и указывает в качестве параметра - имя действия. Например действие "открыть notepad":

```
python looper.py --record open_notepad
# или короткая форма:
python looper.py -r open_notepad
```

Программа записыват все действия пользователя, которые он делает с мышью и клавиатурой до нажатия Esc.

- движения мышью: поднятие (up), опускание (down), левая/правая.
- ввод с клавиатуры: какая клавиша была нажата и какая при этом была раскладка. Записываются только печатные символы, space, enter.

Действия записываются в лог-файл: ACTION_FOLDER/open_notepad/log.json.

ACTION_FOLDER берётся из конфигурационного файла looper.config

В момент совершения действия:

- если действие было активным (нажатие мышью или ввод с клавиатуры enter или space), то делается скрин экрана, который сохраняется в отдельный файл с уникальным именем и в лог действия пишется имя файла.
- сохранятеся время прошедшее с момента запуска программы к моменту совершения действия и пишется в лог
- раскладка клавиатуры

Например:

open_notepad_log.json


```
[
    {
        "source": "mouse",
        "button": "left",
        "dir": "down",
        "x": 775,
        "y": 1058,
        "timestamp": 2.430940866470337,
        "screen": "1.png"
    },
    {
        "source": "mouse",
        "button": "left",
        "dir": "up",
        "x": 775,
        "y": 1058,
        "timestamp": 2.5047407150268555
    },
    {
        "source": "keyboard",
        "key": "n",
        "timestamp": 5.583201885223389,
        "layout": "0409"
    },
    {
        "source": "keyboard",
        "key": "o",
        "timestamp": 5.711046934127808,
        "layout": "0409"
    },
    {
        "source": "mouse",
        "button": "left",
        "dir": "down",
        "x": 989,
        "y": 13,
        "timestamp": 80.71313333511353,
        "screen": "2.png"
    },
    {
        "source": "mouse",
        "button": "left",
        "dir": "up",
        "x": 989,
        "y": 13,
        "timestamp": 80.82457709312439
    }
```

```
]
```

Воспроизведение

Воспроизводит действия пользователя, точнее базовые действия полученные на этапе декомпозиции.

Вызывается командой

```
python looper.py --play open_notepad
# или короткая форма:
python looper.py -p open_notepad
```

Дополнительные параметры:

- `--actions-file <filename>` или `-f <filename>` - имя файла с базовыми действиями. По умолчанию используется `actions_base.json`

Пример:

```
python looper.py -p open_notepad -f custom_actions.json
```

Декомпозиция

Вызывается командой:

```
python looper.py --decompose open_notepad
# или короткая форма:
python looper.py -d open_notepad
```

Программа раскладывает все действия пользователя из файла с логами ACTION_FOLDER/open_notepad/log.json на базовые действия. Список базовых действий:

```
[
    {
        name : "click left",
        begin :
        {
            "source": "mouse",
            "button": "left",
            "dir"   : "down",
        },
        end :
        {
            "source": "mouse",
            "button": "left",
            "dir"   : "up",
        },
        max_delay : 0.3
    },
    {
        name : "click right",
        begin :
        {
            "source": "mouse",
            "button": "right",
            "dir"   : "down",
        },
        end :
        {
            "source": "mouse",
            "button": "right",
            "dir"   : "up",
        },
        max_delay : 0.3
    },
    {
        name : "typing",
        // объединяет все действия от певрого ввода символа, до пер
        text : "строка ввода пользователя"
    },
    {
        // нажатие enter
        name : 'enter'
    },
    {
        // нажатие space, не внутри 'typing' (!)
        name : 'space'
    },
    {
        name: "wait",
```

```
            // событие после наступления которого происходит следующее
            event:
                {
                    name: "timer"
                    // время сколько нужно спать.
                    time: T
                }
        }
    ]
```

Разложение на базовые действия сохраняется в файл ACTION_FOLDER/open_notepad/actions_base.json.

Пример ACTION_FOLDER/open_notepad/actions_base.json :

```
[
    {
        "id"  : 1,
        "name": "click left",
        "type": "mouse_click",
        "button": "left",
        "x": 775,
        "y": 1058,
        "consumed_indices": [
            0,
            1
        ]
    },
    {
        "id"  : 2,
        "name": "wait",
        "event":{
                "name": "timer",
                "time": 5.583201885223389
            }
    },
    {
        "id"  : 3,
        "name": "typing",
        "type": "keyboard_typing",
        "text": "notepad",
        "consumed_indices": [
            2,
            3,
            4,
            5,
            6,
            7,
            8
        ]
    },
    {
        "id"  : 4,
        "name": "wait",
        "event":{
                "name": "timer",
                "time": 3.537025451660156
            }
    },
    {
        "id"  : 5,
        "name": "click left",
        "type": "mouse_click",
```

```
            "button": "left",
            "x": 775,
            "y": 538,
            "consumed_indices": [
                9,
                10
            ]
        }
    ]
```

Можно, чтобы LLM давала название/характеристику каждому действию по картинке.

Создание сценариев

Вызывается командой:

```
python looper.py --scenario open_notepad --output scenario_name
# или короткая форма:
python looper.py -sc open_notepad -o scenario_name
```

Результат пишется в файл `ACTION_FOLDER/open_notepad/scenario_name.json`.

Дополнительные параметры для создания сценариев:

Программа создаёт последовательность (базовых) действий, которые будут в дальнейшем исполняться. Входными данными для неё является последовательность базовых действий полученных на этапе декомпозиции. Данный скрипт меняет базовые действия по определённому правилу.

Опишем, что можно сделать с базовыми действиями.

Фиксированная задержка.

Поставить фиксированную задержку после клика мышкой, enter, space. Для этого нужно написать:

```
python looper.py -sc open_notepad -o scenario_name --delay 2.5
```

где `2.5` - количество секунд для задержки. В созданном сценарии по этой команде все задержки перед клика мышкой, enter, space составляют 2.5 секунды.

Изменить текст ввода.

Базовое действие typing имеет параметр "text". На вход программе можно дать файл typing_parameters.csv с указанием всех значений параметров в ходе воспроизведения:

```
python looper.py -sc open_notepad -o scenario_name --typing-params
```

Формат файла typing_parameters.csv. Например, в списке actions_base.json всего имеются два действия типа typing : {"login", "pass"}. Тогда файл typing_parameters.csv имеет следующий вид:

id	login	pass
1	alex	123
2	sasha	456

Программа для каждой строки файла typing_parameters.csv воспроизводит последовательность базовых действий с соответствующим набором параметров. Будем называть одно такое выполнение (для одной строки) сценарием. Между сценариями программа бездействует S секунд, где S указывается во время запуска:

```
python looper.py -sc open_notepad -o scenario_name --sleep S
```

По умолчанию S=3. Если typing_parameters.csv не задан, то берутся исходные параметры из записи actions_base.json.

Параметр клика мышки.

Базовое действие "wait" имеет параметр event, то есть событие наступление которого следуют ждать прежде чем выполнять следующее действие. По умолчанию все event имеют тип timer и просто ждут фиксированное количество секунд.

Рассмотрим другой тип события "picOnScreen" - ждёт пока на экране появится определённая картинка. Это событие может быть только перед действием "клик мышки". Опишем какую картинку ждёт событие. В момент, когда наступает следующее событие, клик мышки, мы сохраняем скрин экрана и вырезаем из него референсный прямоугольник 10 на 10 с центром в точке клика. Событие для действия считается произошедшим, если на экране появился данный прямоугольник и, тогда в качестве координаты клика используется центра найденного прямоугольника. Параметр референсного прямоугольнка можно изменять. Файл с настройками событий указывается так:

```
python looper.py -sc open_notepad -o scenario_name --click-params c
```

Формат click_parameters.json

```
[
    {
        "id": 1,
        "fn": "pic1.png"
    },
    {
        "id": 4,
        "fn": "pic2.png"
    }
]
```

В поле fn записывается имя файла с референсным прямоугольником