**РЕЗУЛЬТАТИВНОСТЬ РАЗЛИЧНЫХ РАСКЛАДОК ПРИ НАБОРЕ ТЕКСТОВ БОЛЬШИХ РАЗМЕРОВ**

Данный документ предназначен для описания работы проделанной в рамках исследования результативности различных раскладок. Код проекта и последняя версия статьи расположены в репозитории:

https://github.com/dguard/shpya-python-1-keyboard-layout-py

Для изучения эффективности различных раскладок был составлен следующий план исследования:

1. Найти основные используемые раскладки для русского и английского языков
2. Выделить критерии, по которым раскладка считается эффективной
3. Спроектировать программу анализа эффективности раскладок
4. Написать программу на языке JavaScript в режиме работы "Single Page Application" с целью наглядного показа работы программы
5. Протестировать программу на реальном тексте с использованием нескольких раскладок
6. В случае необходимости повторить проектирование, написание и тестирование программы
7. Перенести программу с языка JavaScript на языке Python в режиме работы "консоль"
8. Сделать выводы об эффективности раскладок
9. Сделать выводы о потраченном времени

**1. Тестируемые раскладки:**

* Qwerty
* Colemak
* Дворак
* Йцукен

**2. Критерии эффективности раскладки**

В качестве критериев эффективности были выделены следующие

* пальцы редко убегают из среднего ряда;
* работают в основном указательные и безымянные;
* наибольшая активность сосредоточена в центре клавиатуры;
* руки эффективно чередуются;
* правая рука задействована чуть больше, чем левая;

**3. Проектирование программы**

Были сформулированы следующие функции программы:

1. Анализ эффективности клавиатуры
2. Представление в реальном времени набираемого текста на виртуальной клавиатуре
3. Представление каждого из критериев оценивания на виртуальной клавиатуре
4. Возможность добавления новых критериев оценивания
5. Настройка выбранных фильтров
6. Сохранение состояния приложения
7. Поддержка мобильных устройств

Руководствуясь правилами эффективной разработки, более 5 часов было потрачено на проектирование программы: отношения между классами, используемые структуры данных, реализуемые функции, интерфейс. Было исписано и разрисовано около двух тетрадей по 12 листов. Результатом стала первая версия на языке JavaScript.

**4. Написание программы**

Первая версия программы была написана на языке JavaScript без использования сторонних библиотек: как js, так css-фреймворков.

Однако с ростом функциональности было решено использовать дополнительные библиотеки: такие как RequireJs, с целью создания namespace'ов, которые в языке JavaScript отсутствуют в привычном понимании.

Усложнение логики и желание "не писать свой велосипед" привело к использованию популярного фреймворка Bootstrap3. Теперь страница поддерживается на мобильных на минимальном разрешении 500px. Однако некоторое время было потрачено на перенос существующих стилей и обновление некоторых стилей библиотеки.

**5. Тестирование**

Программа была протестирована с использованием текста "Госпожа Бовари", большинство из желаемых функций были реализованы. Была идея следовать TDD, однако она была отклонена, т.к. полагалось, что это потребует больше времени для реализации. Однако позже автор жалел, что не последовал ей, т.к. провел много времени в отладке при расширении интерфейса и рефакторинге.

**6. Повторное проектирование и реализация**

Периодически возникали трудности с добавлением функциональности и оптимизации существующей. Поэтому программа и используемые структуры данных переосмысливались и безжалостно переписывались.

**7. Перенос программы на язык Python**

Программа была переписана с нуля на язык Python, используя другие структуры и связи между классами, а не как это было в языке JavaScript - в виду специфики языка JavaScript и переосмыслений структуры данных в результате работы над программой.

**8. Выводы об эффективности раскладок**

С целью анализа эффективности, программа была запущена на языке Python. Для английского языка использовались раскладки Qwerty, Colemak, Dworak и текст "Госпожа Бовари". Для тестирования русской раскладки использовался текст "Преступление и наказание".

В результате работы программы были получены следующие результаты:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий оценивания | Qwerty | Colemak | Dworak | Йцукен\* |
| 1. Общее время набора в секундах (часах) | 33389  (556.48) | 32218  (536.96) | 32601  (543.35) | 59208  (986.8) |
| 2. Средняя эффективность клавиш\*\* | 66.44 | 68.91 | 68.09 | 68.37 |
| 3. Скорость набора (символов за минуту) | 19.80 | 20.52 | 20.28 | 20.32 |
| 3. Процент использования большого пальца | 16.36 | 16.36 | 16.36 | 16.59 |
| 4. Процент использования указательного пальца | 35.42 | 34.25 | 26.45 | 54.60 |
| 5. Процент использования безымянного пальца | 17.52 | 14.05 | 16.95 | 6.35 |
| 6. Процент использования правой руки | 53.32 | 62.47 | 64.63 | 58.89 |
| 7. Процент чередования рук | 57.89 | 56.19 | 57.04 | 52.85 |
| 8. Процент использования центральной части клавиатуры | 55.10 | 53.87 | 48.61 | 65.43 |
| 9. Процент нахождения рук в центральном ряду | 30.94 | 61.94 | 58.66 | 32.71 |

\* Русская раскладка может сравниваться по всем критериям, кроме времени набора текста, т.к. текст для тестирования использовался другой

\*\* Эффективность - произведение рейтинга клавиши на количество использований клавиши

Наиболее эффективной оказалась раскладка Colemak - она показала результат 20.52 символов за минуту. Наименее эффективной - Qwerty (19.80 символов в минуту). У Colemak наиболее удобное расположение клавиш - 61.94% текста были написаны, находясь в центральном ряду клавиатуры.

Необходимо помнить, что скорость набора текста у разных людей может значительно отличаться. Рассматривалась базовая скорость набора одного символа - 200 мсек. В случае, если рейтинг клавиши был меньше, чем 1 (100%), скорость клавиши изменялась обратно пропорционально рейтингу (например, 50% рейтинг при скорости 200 мсек = 300 мсек).

С точки зрения расположения клавиш наиболее эффективной раскладкой является Colemak. На втором месте - Dworak. На третьем - Qwerty. Великолепная возможность оценить эффективность раскладок по другим критериям предоставляется читателю.

**9. Сделать выводы о потраченном времени**

|  |  |
| --- | --- |
| Действие | Затраты по времени |
| Осмысление задачи, сырое обдумывание алгоритма и используемых функций | 30 минут |
| **JavaScript** | |
| Проектирование приложения на JavaScript: установление отношения между объектами, дальнейший анализ функций приложения, поиск лучших способов хранения данных | 5 часов |
| Написание виртуальной клавиатуры Html+Css+Js и окна вывода текста | 2 часа |
| Добавление чтения файла (ошибки с Cross Domain Resource Sharing и политики безопасности) | 2 часа |
| Написание модели фильтра оценивания и его коллекций | 10 часов |
| Написание раскладок и их взаимодействия с приложением | 5 часов |
| Связь логики приложения с представлением | 5 часов |
| Добавление возможности сохранения состояния приложения | 1 час |
| **Python** | |
| Проектирование приложения на языке Python | 1 час |
| Написание модели раскладки и коллекций | 1 час |
| Написание модели фильтров и их коллекций | 3 часа |
| Устранение ошибок в работе фильтров оценивания | 1 час |
| Рефакторинг | 1 час |
| Тестирование на реальных данных | 30 минут |
| Написание отчета | 1 час |

Общее время: 39 часов

Большую часть времени - около 80% занял debug, поиск ошибок тогда, когда ты уверен, что их нет. Например, некоторые странности поведения в JavaScript, такие как политика безопасности и CORS (получение файла через ajax из внешнего источника). Невнимательность при работе с языком программирования.

Вероятно, это время можно было бы провести с большей пользой, если писать приложения используя Test Driven Development. Эта гипотеза будет проверена в следующем домашнем задании - планируется его написание в стиле TDD. Тем не менее был получен ценный опыт, пусть и не всегда положительный.