Федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук
Основная образовательная программа
Прикладная математика и информатика

КУРСОВАЯ РАБОТА

Программный проект на тему
Графические JavaScript библиотеки

Выполнил студент группы 185, 3 курса:
Войтецкий Артем Александрович

Руководитель КР:
Доцент, Никитин Алексей Антонович

Москва 2021

Оглавление

Введение

В современном мире люди привыкли получать больше всего информации с помощью визуальных образов. Поэтому использование визуальных материалов может существенно упростить понимание материала школьниками и студентами. Сейчас для этого преподаватели чаще используют доску в классе, на которой каждый раз от руки рисуют схематичные примеры, но даже это происходит не всегда. Именно поэтому внедрение в образовательный процесс современных информационных технологий для создания визуальных материалов очень важная и актуальная задача. В этом может помочь разрабатываемый сервис «VisualMath.ru».

Постановка задачи

Хотелось бы, чтобы каждый желающий мог открыть учебные материалы на своем компьютере, ноутбуке, смартфоне или планшете. И, несмотря на то, что мощности современных устройств без проблем хватает на работу в браузере, это не значит, что мы не должны стремиться к оптимизации работы наших программ. Замедление работы всей системы влечет за собой низкую продуктивность обучения, ведь пользователь отвлекается от главного. Сокращение времени автономной работы устройств вредит учащимся, которые могли бы использовать сервис прямо на занятиях для лучшего понимания материала.

К сожалению, проблемы не обошли и VisualMath. Множество существующих материалов страдали от утечек памяти (наглядно это будет продемонстрировано позднее), что не может не вредить всему проекту по вышеописанным причинам.

Моей задачей стал поиск и исправление этой проблемы.

Начало работы

Материалы проекта были написаны на языке программирования JavaScript с использованием библиотеки Grafar. Она разрабатывалась в сотрудничестве с сервисом VisualMath, поэтому эта библиотека является важной частью всего проекта и соответствует его нуждам. Работа осуществляется с помощью высокоуровневого интерфейса, что повышает удобство разработки, а подробная документация позволяет с легкостью разобраться в основах. Также неоспоримым плюсом является прямой контакт с разработчиком, что в некоторых ситуация может помочь с решением возникающих проблем.

Именно потому что Grafar является такой важной и неотъемлемой частью VisualMath, я решил начать свою работу со знакомства с этой библиотекой. Для этого было реализован тестовый проект, в котором реализована визуализация морфинга куба в сферу.

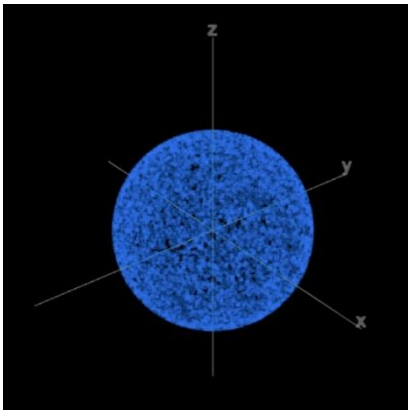
Реализация

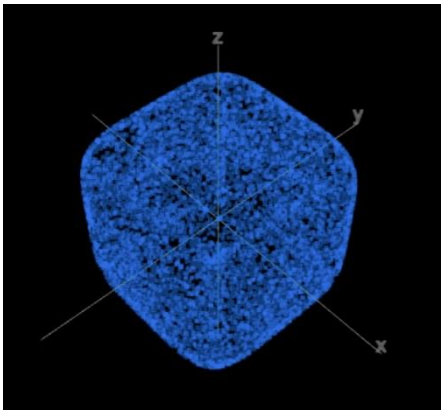
Для реализации я использовал формулу, где R – это радиус сферы и половина стороны куба:

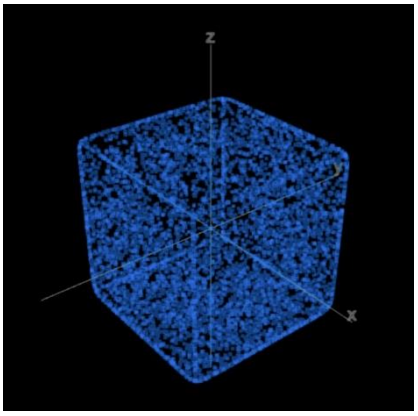
$$x^n + y^n + z^n - R = 0$$

$$2 \leq n < \infty$$

При n = 2 в результате получается сфера с радиусом R, а с увеличением n результат приближается к кубу. Таким образом реализуется плавное преобразование от одной фигуры к другой. Для этого в библиотеке Grafar я использовал метод grafar.vsolve(), в который передается формула, сколько решений необходимо найти и размерность объекта, а на выходе получаем набор точек, удовлетворяющий условиям. Получается, что фигура отрисовывается множеством точек.







Я реализовал два варианта визуализации. Вариант, в котором пользователь сам выбирает стадию морфинга через специальный инструмент, а также вариант, в котором постепенный морфинг зациклен и происходит автоматически.

Но, несмотря на то, что получившиеся картинки достаточно информативны, их можно вращать в разных плоскостях, приближать и отдалять, есть и некоторые недочеты. Такой способ реализации требует больше вычислительных ресурсов, ведь сначала нужно найти подходящие точки, а потом нарисовать каждую в своем месте. Также это не самый красивый вариант изображения фигур.

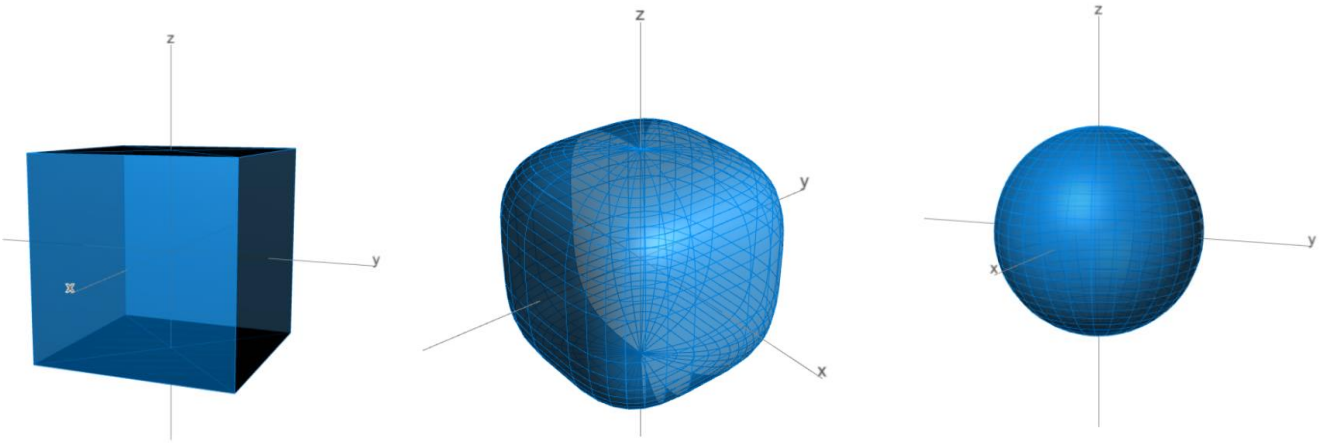
Альтернативный вариант

Для лучшего результата можно использовать суперформулу:

$$x = sign(\cos(\alpha)) * |\cos(\alpha)^n| * sign(\cos(\beta)) * |\cos(\beta)^n|$$
$$y = sign(\cos(\alpha)) * |\cos(\alpha)^n| * sign(\sin(\beta)) * |\sin(\beta)^n|$$
$$z = sign(\sin(\alpha)) * |\sin(\alpha)^n|$$

$$-\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2}$$
$$-\pi \leq \beta \leq \pi$$

Также в библиотеке Grafar есть методы grafar.pin() и grafar.map(), комбинацией которых можно отобразить представленную выше суперформулу. В таком случае уже не отрисовывается множество конкретных точек, а строятся поверхности по результатам формулы, что существенно сокращает требуемые вычислительные ресурсы.



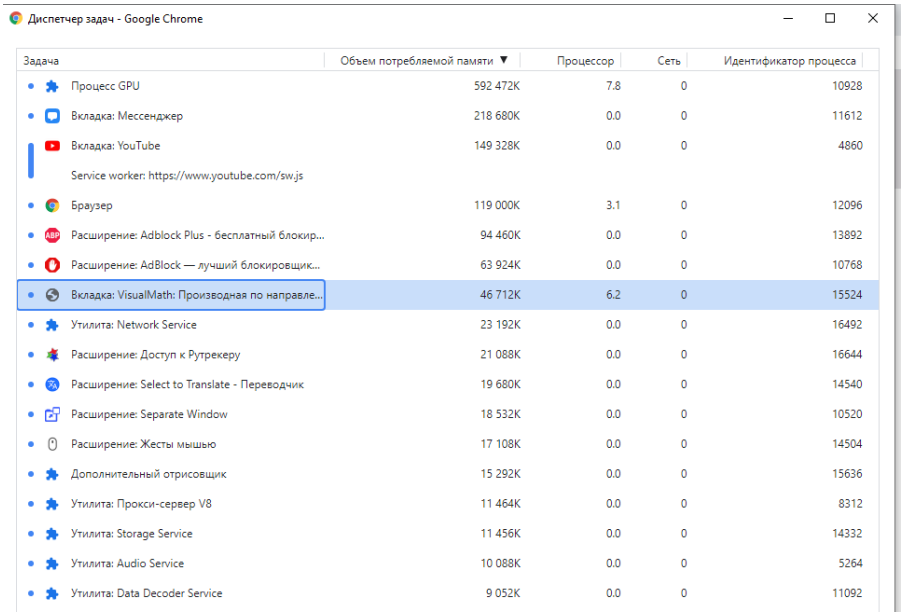
Как видно из скриншотов, результат получился визуально приятнее. Также использование этих методов раскрывает одно важное достоинство библиотеки. Она построена с использованием принципа реактивной анимации. Этот принцип позволяет явно выразить зависимости между элементами, а также реагировать на событие, вычисляя новые значения всех зависимых элементов. Таким образом достигается простота в построении различных анимаций, ведь нет нужды вручную просчитывать все шаги.

На основе именно второго метода в дальнейшем строились более сложные визуализации.

Исследование и решение проблемы

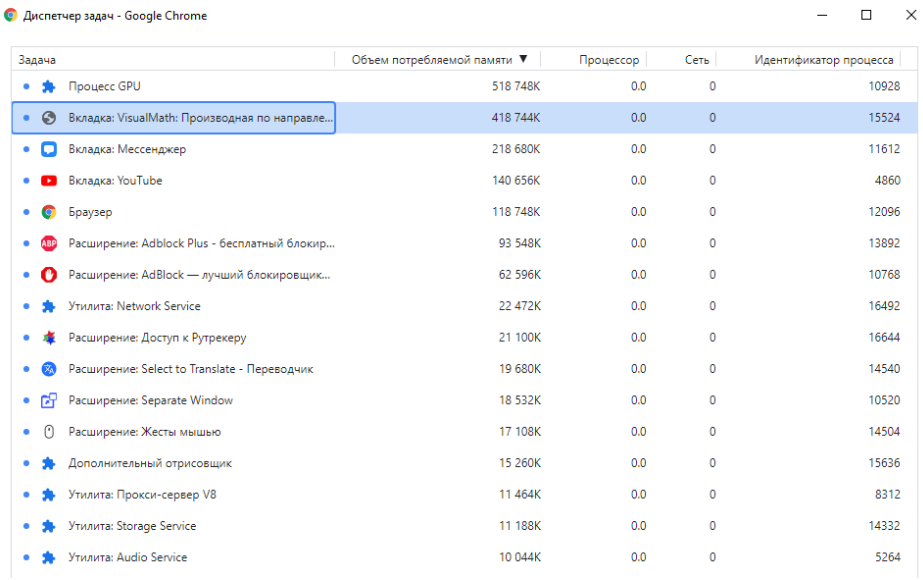
Далее я приступил к исследованию проблемы, которую предстояло решить.

На скриншотах представляю вам потребление памяти для двух популярных интернет-страниц: vk.com и youtube.com, а также потребление памяти одного из проблемных проектов сразу после его запуска. В начале все достаточно хорошо, но утечка проявляется во время работы.



Задача	Объем потребляемой памяти ▼	Процессор	Сеть	Идентификатор процесса
Процесс GPU	592 472K	7.8	0	10928
Вкладка: Мессенджер	218 680K	0.0	0	11612
Вкладка: YouTube	149 328K	0.0	0	4860
Service worker: https://www.youtube.com/sw.js				
Браузер	119 000K	3.1	0	12096
Расширение: Adblock Plus - бесплатный блокир...	94 460K	0.0	0	13892
Расширение: AdBlock — лучший блокировщик...	63 924K	0.0	0	10768
Вкладка: VisualMath: Производная по направле...	46 712K	6.2	0	15524
Утилита: Network Service	23 192K	0.0	0	16492
Расширение: Доступ к Рутрекеру	21 088K	0.0	0	16644
Расширение: Select to Translate - Переводчик	19 680K	0.0	0	14540
Расширение: Separate Window	18 532K	0.0	0	10520
Расширение: Жесты мышью	17 108K	0.0	0	14504
Дополнительный отрисовщик	15 292K	0.0	0	15636
Утилита: Прокси-сервер V8	11 464K	0.0	0	8312
Утилита: Storage Service	11 456K	0.0	0	14332
Утилита: Audio Service	10 088K	0.0	0	5264
Утилита: Data Decoder Service	9 052K	0.0	0	11092

На втором скриншоте представлено потребление памяти для тех же двух ресурсов и нашего подопытного проекта, но уже спустя непродолжительное время работы в нем.



Задача	Объем потребляемой памяти ▼	Процессор	Сеть	Идентификатор процесса
Процесс GPU	518 748K	0.0	0	10928
Вкладка: VisualMath: Производная по направле...	418 744K	0.0	0	15524
Вкладка: Мессенджер	218 680K	0.0	0	11612
Вкладка: YouTube	140 656K	0.0	0	4860
Браузер	118 748K	0.0	0	12096
Расширение: Adblock Plus - бесплатный блокир...	93 548K	0.0	0	13892
Расширение: AdBlock — лучший блокировщик...	62 596K	0.0	0	10768
Утилита: Network Service	22 472K	0.0	0	16492
Расширение: Доступ к Рутрекеру	21 100K	0.0	0	16644
Расширение: Select to Translate - Переводчик	19 680K	0.0	0	14540
Расширение: Separate Window	18 532K	0.0	0	10520
Расширение: Жесты мышью	17 108K	0.0	0	14504
Дополнительный отрисовщик	15 260K	0.0	0	15636
Утилита: Прокси-сервер V8	11 464K	0.0	0	8312
Утилита: Storage Service	11 188K	0.0	0	14332
Утилита: Audio Service	10 044K	0.0	0	5264

Проблема утечки памяти становится не просто очевидна, а пугает своим масштабом, ведь это далеко не предел. Разумеется, аналогичное наблюдалось и в других проектах.

После подтверждения проблемы, начались поиски ее причины. Утечка памяти сохранялась во всех проектах в той или иной степени, поэтому поиски я начал с общих элементов - библиотек. Такой подход привел меня к успеху, источник был найден, обнаружился он в библиотеке Grafar. Утечка происходила из-за того, что старое состояние визуализации удалялось не до конца, т.е. потребление памяти накапливалось со временем. Поговорив с разработчиком (наглядная демонстрация преимущества тесного сотрудничества), он подтвердил мои выводы.

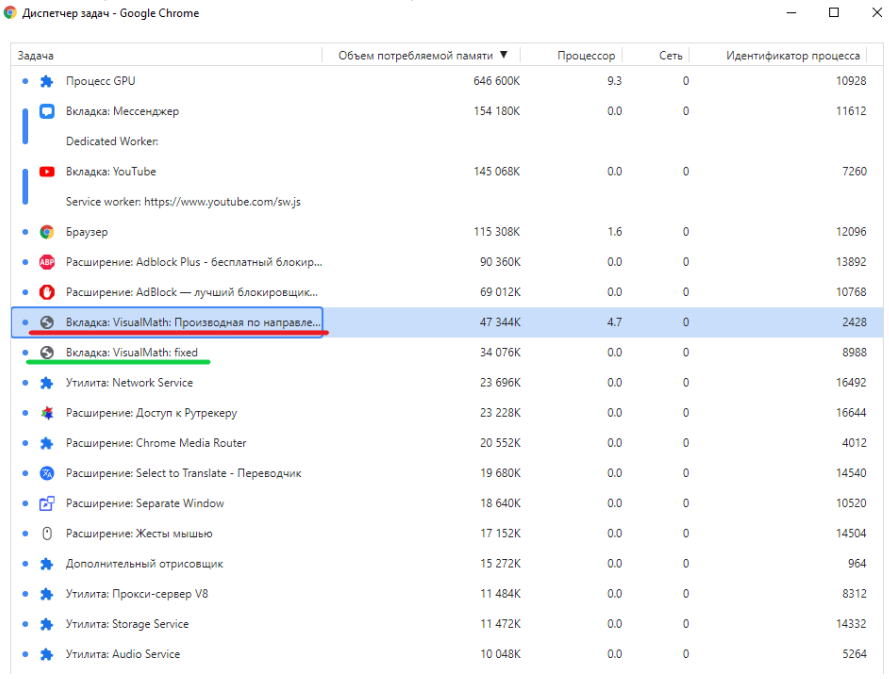
Решение нашлось в обновлении версии Grafar на более свежую. Предоставленные проекты были написаны достаточно давно, чтобы библиотека успела сильно обновиться и избавиться от этой проблемы. Но возникла иная – отсутствие обратной совместимости. Некоторые методы изменились, а некоторые и вовсе перестали существовать. Также дополнительную сложность вызвало отсутствие полной документации. На момент написания отчета появился удобный сайт, в котором многое описано, но изначально учиться можно было лишь на небольшом количестве предоставленных примеров, а о работе некоторых методов из предыдущей версии приходилось догадываться или проверять опытным путем.

Мне пришлось полностью переписать данные проекты, сохраняя задачу каждого. Также удалось придать коду более понятный вид и избавиться от некоторых ненужных действий. Но не везде удалось полностью повторить исходное изображение, так как часть необходимого функционала отсутствует в современной версии библиотеки. Например, управление толщиной линии или размером точки. Но даже в таких случаях удалось найти альтернативное решение, которое сохраняет для пользователя простоту визуального восприятия.

Результаты и выводы

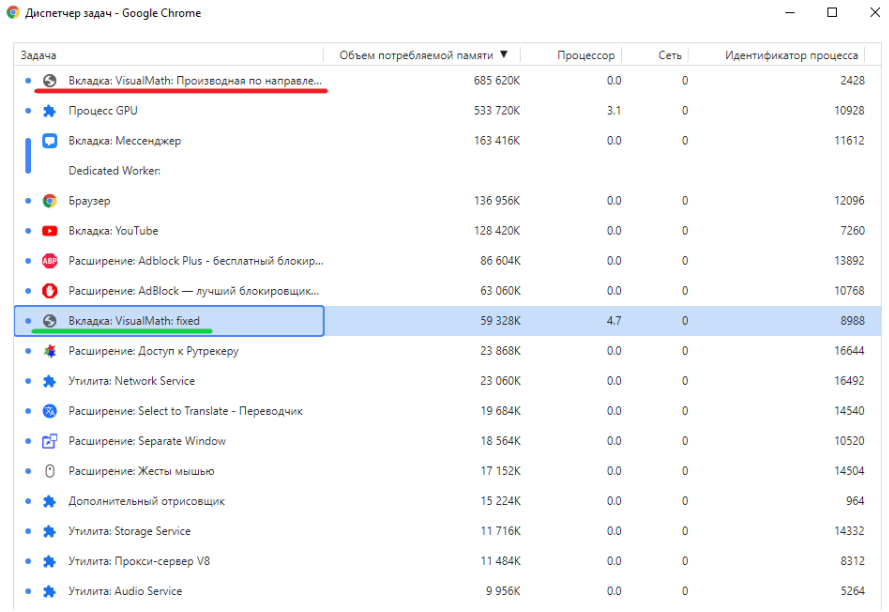
Начну подведение итогов со сравнения изначального проекта и исправленного.

Как и в прошлый раз, начну с только что открытого проекта. Можно заметить экономию по памяти уже сейчас, но не будем останавливаться.



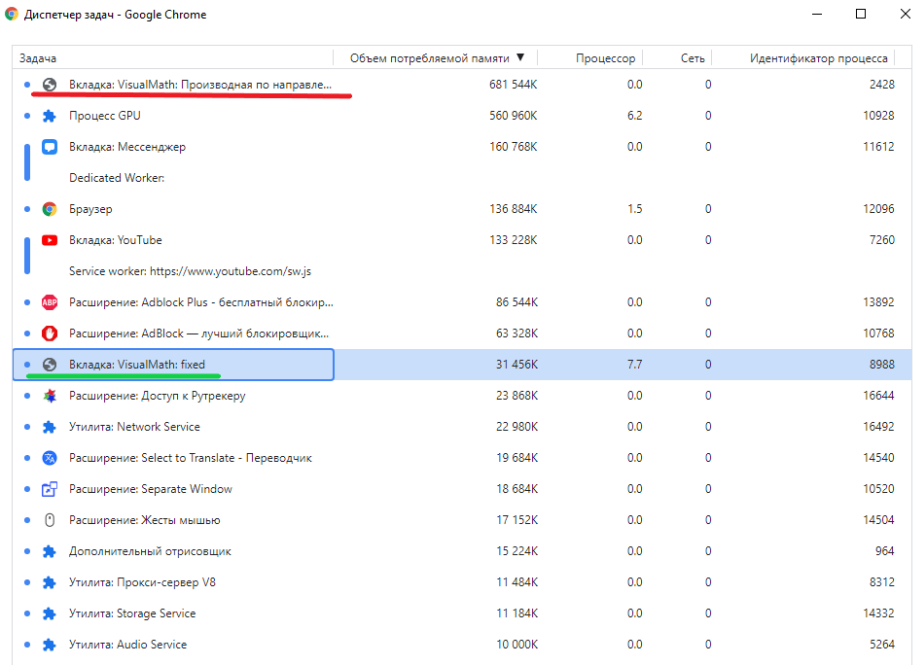
Задача	Объем потребляемой памяти ▼	Процессор	Сеть	Идентификатор процесса
Процесс GPU	646 600K	9.3	0	10928
Вкладка: Мессенджер	154 180K	0.0	0	11612
Dedicated Worker:				
Вкладка: YouTube	145 068K	0.0	0	7260
Service worker: https://www.youtube.com/sw.js				
Браузер	115 308K	1.6	0	12096
Расширение: Adblock Plus - бесплатный блокир...	90 360K	0.0	0	13892
Расширение: AdBlock — лучший блокировщик...	69 012K	0.0	0	10768
Вкладка: VisualMath: Производная по направле...	47 344K	4.7	0	2428
Вкладка: VisualMath: fixed	34 076K	0.0	0	8988
Утилита: Network Service	23 696K	0.0	0	16492
Расширение: Доступ к Рутрекеру	23 228K	0.0	0	16644
Расширение: Chrome Media Router	20 552K	0.0	0	4012
Расширение: Select to Translate - Переводчик	19 680K	0.0	0	14540
Расширение: Separate Window	18 640K	0.0	0	10520
Расширение: Жесты мышью	17 152K	0.0	0	14504
Дополнительный отрисовщик	15 272K	0.0	0	964
Утилита: Прокси-сервер V8	11 484K	0.0	0	8312
Утилита: Storage Service	11 472K	0.0	0	14332
Утилита: Audio Service	10 048K	0.0	0	5264

На следующем скриншоте представлено потребление памяти после 2 минут активного взаимодействия с каждым из проектов.



Задача	Объем потребляемой памяти ▼	Процессор	Сеть	Идентификатор процесса
Вкладка: VisualMath: Производная по направле...	685 620K	0.0	0	2428
Процесс GPU	533 720K	3.1	0	10928
Вкладка: Мессенджер	163 416K	0.0	0	11612
Dedicated Worker:				
Браузер	136 956K	0.0	0	12096
Вкладка: YouTube	128 420K	0.0	0	7260
Расширение: Adblock Plus - бесплатный блокир...	86 604K	0.0	0	13892
Расширение: AdBlock — лучший блокировщик...	63 060K	0.0	0	10768
Вкладка: VisualMath: fixed	59 328K	4.7	0	8988
Расширение: Доступ к Рутрекеру	23 868K	0.0	0	16644
Утилита: Network Service	23 060K	0.0	0	16492
Расширение: Select to Translate - Переводчик	19 684K	0.0	0	14540
Расширение: Separate Window	18 564K	0.0	0	10520
Расширение: Жесты мышью	17 152K	0.0	0	14504
Дополнительный отрисовщик	15 224K	0.0	0	964
Утилита: Storage Service	11 716K	0.0	0	14332
Утилита: Прокси-сервер V8	11 484K	0.0	0	8312
Утилита: Audio Service	9 956K	0.0	0	5264

Здесь может показаться, что проблема решена не до конца, ведь потребление памяти увеличилось все равно, пусть и не так значительно, но стоит немного подождать и память вернется обратно к изначальному потреблению.



Таким образом, можно сделать вывод, что проблема с утечкой памяти полностью решена. Также, согласно субъективным ощущениям, визуализации стали более отзывчивы на действия пользователя.

Ссылки

Ссылка на проект: https://github.com/MrARVO/JavaProj
Grafar: https://github.com/thoughtspile/Grafar
Документация Grafar: https://thoughtspile.github.io/grafar/?new#/README
Superformula wiki: https://en.wikipedia.org/wiki/Superformula
Superformula: http://paulbourke.net/geometry/supershape/