|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Студент Климов Илья Сергеевич \_

*фамилия, имя, отчество*

Группа ИУ7-52Б \_

Тип практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Название предприятия\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Климов И.С. \_

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2021 г.*

**Оглавление**

[**Введение** 3](#_Toc81489427)

[**1.** **Аналитическая часть** 6](#_Toc81489428)

[**1.1.** **Существующие программные обеспечения** 6](#_Toc81489429)

[**1.1.1.** **Приложение «Кубик Рубика»** 6](#_Toc81489430)

[**1.1.2.** **Приложение «Free Super Cube»** 7](#_Toc81489431)

[**1.1.3.** **Приложение «Cube Genius»** 8](#_Toc81489432)

[**1.2.** **Алгоритмы удаления невидимых поверхностей.** 9](#_Toc81489433)

[**1.2.1.** **Алгоритм Робертса** 9](#_Toc81489434)

[**1.2.2.** **Алгоритм художника** 10](#_Toc81489435)

[**1.2.3.** **Алгоритм Z-буфера** 11](#_Toc81489436)

[**1.3.** **Методы закрашивания** 12](#_Toc81489437)

[**2.** **Конструкторская часть.** 13](#_Toc81489438)

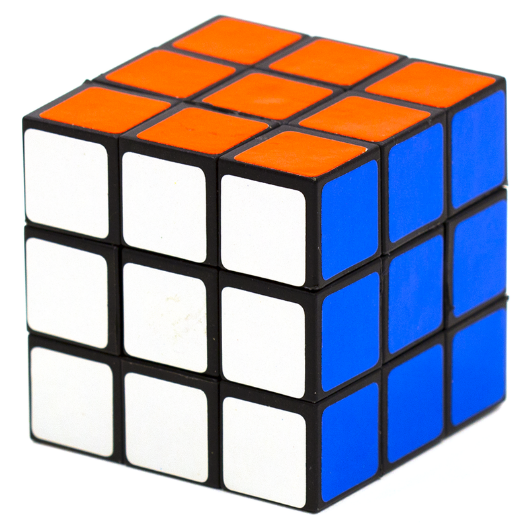
[**3.** **Технологическая часть.** 14](#_Toc81489439)

[**Заключение.** 15](#_Toc81489440)

[**Список литературы** 16](#_Toc81489441)

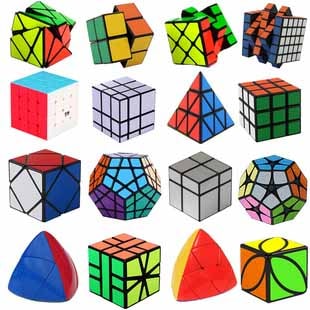
# **Введение**

В 1974 году венгерский скульптор Эрнё Рубик изобрел головоломку, которая впоследствии стала одной из самых популярных в мире, и, наверное, нет такого человека, который бы не слышал про нее. Головоломка носит название в честь её изобретателя – кубик Рубика и представляет собой пластмассовый куб с подвижными деталями.



*Рис.1 – Кубик Рубика в собранном состоянии.*

Изначально размерность составляла только 3x3x3, то есть в таком случае кубик состоит из крестовины с 6-ю центральными элементами, 8 углов и 12 реберных элементов. Затем начали появляться различные модификации, включающие в себя не только другую размерность, но и другую форму с отличающимся внутренним строением. Одними из самых популярных являются «пирамидка Мефферта» и «мегаминкс» (головоломка в форме додекаэдра). Они имеют схожий принцип с кубиком, но разное количество граней и деталей.



*Рис.2 – Разновидности головоломок.*

Задача решения кубика состоит в его сборке в изначальное положение путем поворотов граней. Сложность обуславливается тем, что число всех достижимых состояний составляет приблизительно 43 квинтиллиона (43 \* 1018). Занимательно, что из любого положения можно привести к единственному собранному всего за 20 ходов. На сегодняшний день проводится множество соревнований по различным дисциплинам. Классический кубик Рубика 3 на 3 был собран буквально за 3 секунды. [1]

Однако не все хотят тратиться на покупку головоломки, и гораздо удобней воспользоваться виртуальным аналогом. Благо, что с развитием технологий начали появляться приложения с визуализацией этой головоломки и возможностью ее сборки.

**Цель работы** – создание программного продукта, визуализирующего головоломки «кубик Рубика», «пирамидка Мефферта» и «мигаминкс» и позволяющего собирать их. Для достижения цели были выделены следующие **задачи**:

1. Определить требования для приложения основе анализа аналогичных;
2. Провести анализ алгоритмов удаления невидимых поверхностей и выбрать подходящий;
3. Провести анализ методов закрашивания и выбрать наиболее подходящий;

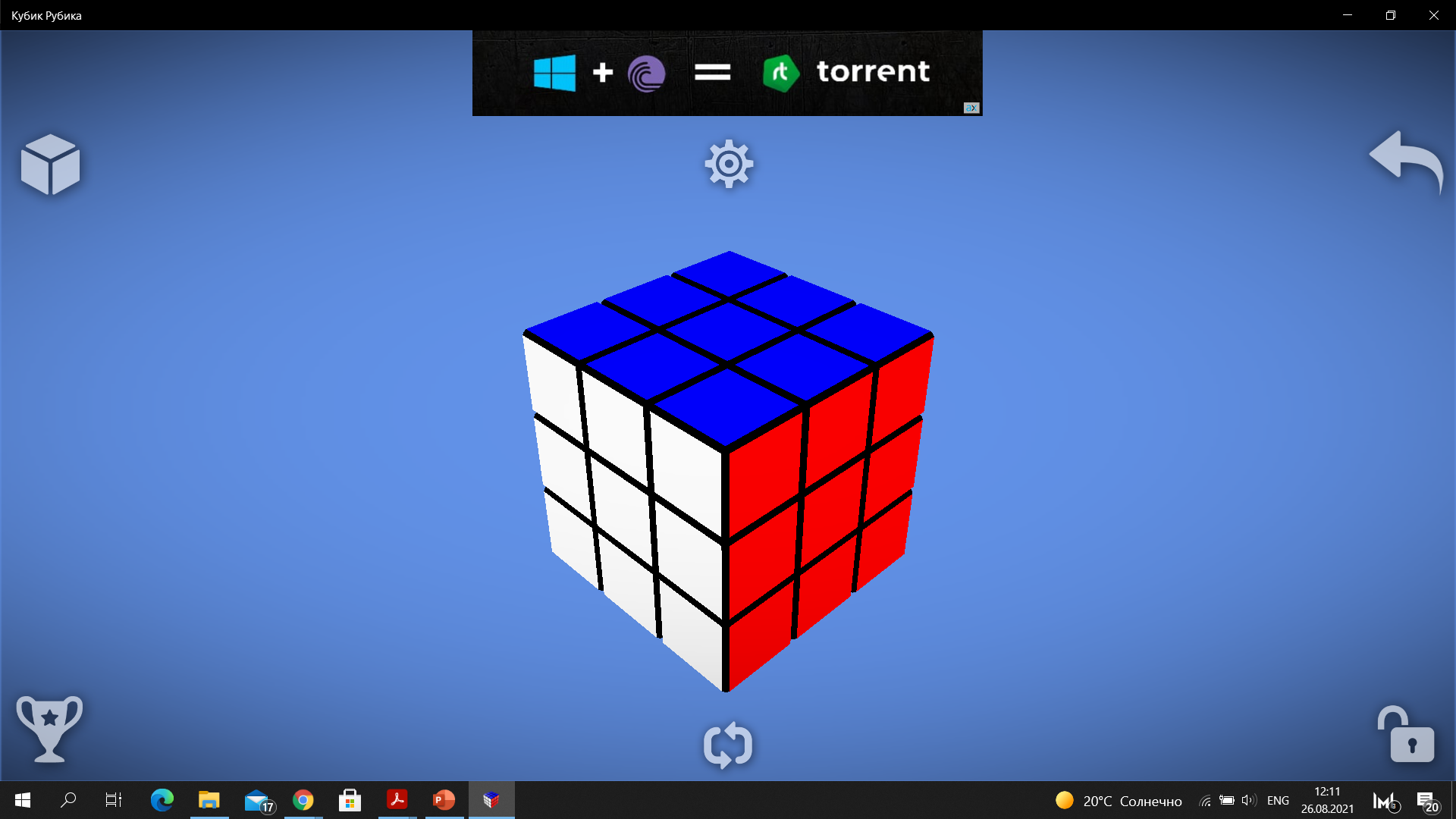
# **Аналитическая часть**

В данном разделе рассматриваются существующие программы по заданной теме, анализируются алгоритмы удаления невидимых поверхностей, закрашивания. Исходя из анализа выбирается наиболее подходящий вариант.

## **Существующие программные обеспечения**

В качестве аналогов для проекта были взяты приложения из Microsoft Store[[1]](#footnote-1). По соответствующему запросу было найдено три продукта, каждый из которых имеет как плюсы, так и минусы.

### **Приложение «Кубик Рубика»**



*Рис.3 – Экран приложения «Кубик Рубика»*

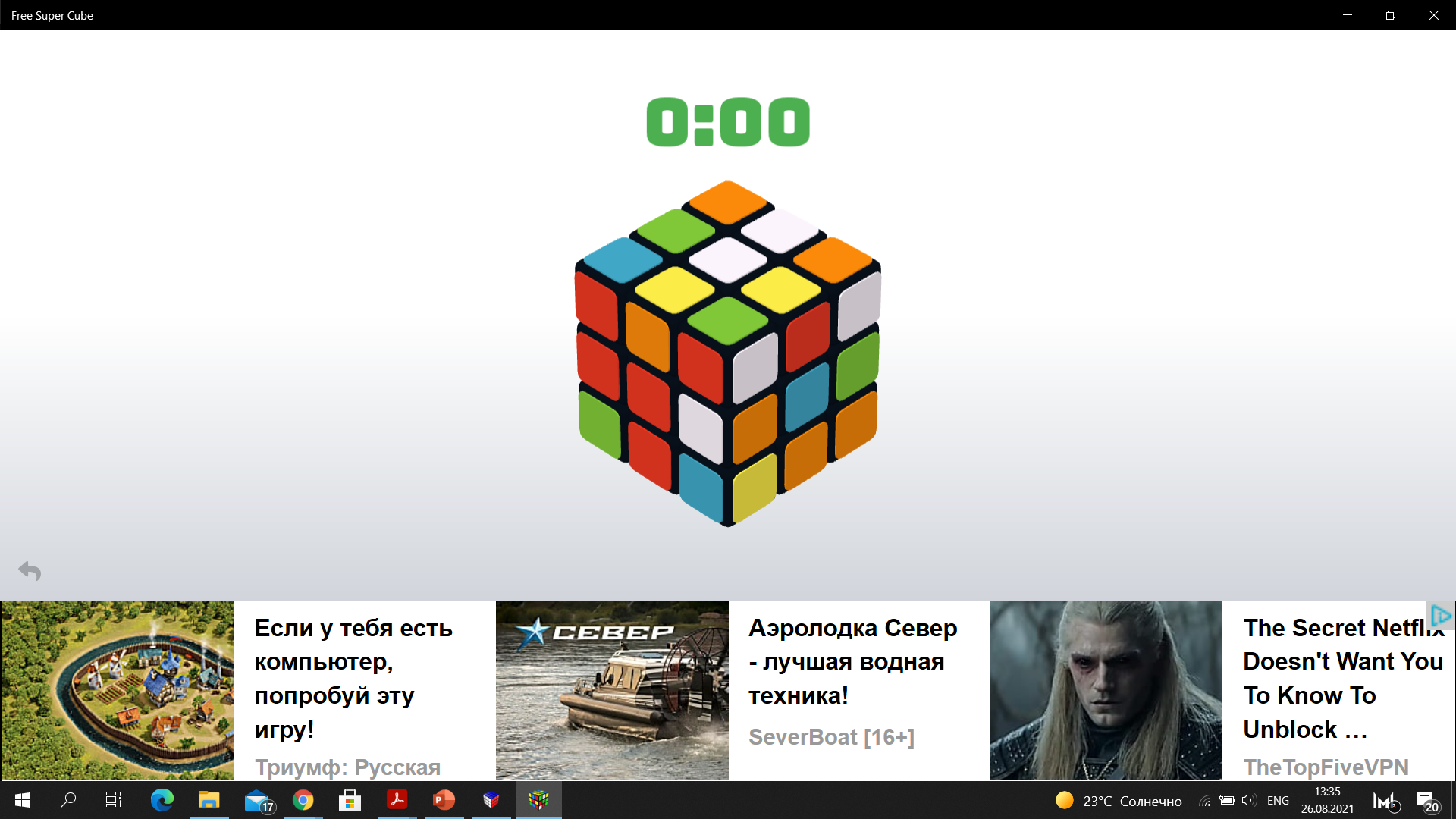
**Достоинства:**

* Возможность выбора головоломки (кубик Рубика, пирамидка Мефферта, мегаминкс) и её размерности;
* Возможность масштабирования;
* Возможность вращения головоломки в разных плоскостях;
* Возможность вернуть предыдущее состояние.

**Недостатки:**

* Грани поворачиваются быстро, без контроля;
* На гранях головоломки виден «лестничный эффект»;
* Отсутствие возможности кастомизации;
* Отсутствие кнопок для управления головоломкой;
* Наличие рекламы.

### **Приложение «Free Super Cube»**



*Рис.4 – Экран приложения «Free Super Cube»*

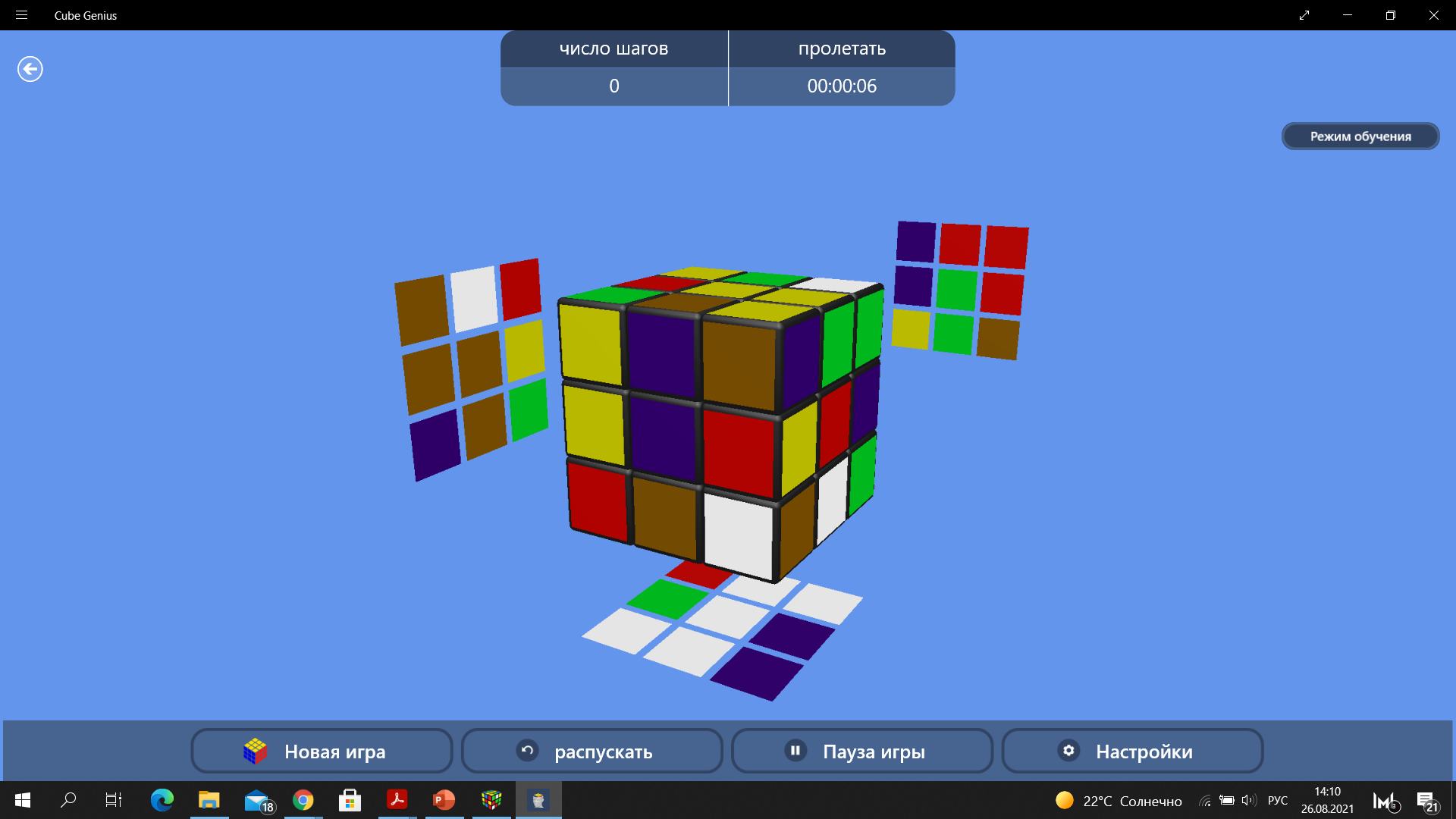
**Достоинства:**

* Плавный поворот граней мышкой, возможность контроля;
* Головоломка хорошо прорисована, гладкие линии;
* Наличие настроек головоломки;
* Возможность выбора размерности головоломки.

**Недостатки:**

* Поворот вокруг осей четко фиксирован;
* Отсутствие выбора головоломок;
* Отсутствие кнопок для управления головоломкой.

### **Приложение «Cube Genius»**



*Рис.5 – Экран приложения «Cube Genius»*

**Достоинства:**

* Видны цвета нелицевых граней;
* Отсутствие рекламы.

**Недостатки:**

* Грани поворачиваются быстро, без контроля;
* На гранях головоломки виден «лестничный эффект»;
* Отсутствие возможности кастомизации;
* Поворот вокруг осей четко фиксирован;
* Отсутствие выбора головоломок;
* Отсутствие кнопок для управления головоломкой.

**Вывод:** анализ аналогичных приложений, выявление их плюсов и минусов показали, какие требования следует предъявить для будущего приложения:

* Плавный поворот граней мышкой, возможность контроля;
* Головоломка хорошо прорисована, гладкие линии;
* Возможность выбора головоломки (кубик, пирамидка, мегаминкс) и её размерности;
* Возможность вращения головоломки в разных плоскостях;
* Возможность масштабирования;
* Наличие настроек головоломки;
* Возможность поворота граней с помощью кнопок и клавиш.

## **Алгоритмы удаления невидимых поверхностей.**

Задача удаления невидимых поверхностей является одной из самых важных и сложных в компьютерной графике. Выделяют три класса данных алгоритмов:

1. Работающие в объектном пространстве – работают с физической системой координат, в которой описаны объекты. Получаются весьма точные результаты, объем вычислений растет, как квадрат числа объектов;
2. Работающие в пространстве изображения – работают с системой координат экрана, на котором визуализируются объекты. Точность вычислений ограничена разрешением экрана, объем вычислений растет, как произведение количества объектов на число пикселей;
3. Формирующие списки приоритетов – работают попеременно с физической и системой координат экрана. [2]

Так как взаимодействие между пользователем и программой происходит в режиме реального времени, то одной из главных характеристик будет время, затрачиваемое на алгоритм.

### **Алгоритм Робертса**

Алгоритм Робертса является первым известным решением задачи удаления невидимых линий. Он позволяет определить, какие рёбра или часть рёбер видимы, а какие заслонены другими гранями. [3] Относится к алгоритмам, работающим в объектном пространстве.

Алгоритм Робертса проходит в два этапа:

1. Определение нелицевых для каждого объекта;
2. Определение и удаление невидимых ребер. Каждое из ребер последовательно сравнивается со всеми остальными телами. В классической версии количество вычислений растет теоретически как квадрат числа объектов. [4]

**Достоинства:**

* Относительная простота.

**Недостатки:**

* Объем вычислений растет, как квадрат числа объектов;
* Ориентирован на работу с выпуклыми многогранниками;
* Невозможность работы с прозрачными объектами.

### **Алгоритм художника**

Алгоритм художника позволяет определить, какие грани видимы, а какие заслонены, выводит каждую из них целиком по мере приближения к наблюдателю, то есть подобно тому, как художник наносит изображением на холст слой за слоем. Изначально все объекты необходимо отсортировать от заднего плана к переднему. [5] Данный алгоритм относится к алгоритмам, работающим в пространстве изображения.

**Достоинства:**

* При тривиальной сортировке обладает высоким быстродействием;
* Простота.

**Недостатки:**

* Сортировки по оси Z может быть нетривиальной задачей;
* При некотором расположении граней не может дать верный результат;
* Происходит отрисовка всех граней объектов, из-за чего время работы может сильно увеличиться.

### **Алгоритм Z-буфера**

Алгоритм Z-буфера позволяет определить, какие пиксели граней видны, а какие заслонены другими. Является одним из простейших алгоритмов удаления невидимых поверхностей, работает в пространстве изображения. [3] Можно сказать, что это обобщение идеи о буфере кадра. Он используется для запоминания цвета каждого пикселя. Z-буфер же является двухмерным массивом, размеры которого равны размерам окна, и в каждой ячейке хранится значение глубины пикселя. В процессе работы z-координата каждого пикселя сравнивается с соответствующим значением в Z-буфере. Если она ближе, то значения в двух буферах обновляются. [6] Производительность можно оценить, как O(CN), где N – количество граней, C – количество пикселей [3]

**Достоинства:**

* Простота;
* Делает тривиальной визуализацию пересекающихся поверхностей, сцены могут быть любой сложности;
* Отсутствие предварительной сортировки по глубине;
* Высокая производительность.

**Недостатки:**

* Требуется большой объем памяти;
* Высокая стоимость устранения лестничного эффекта, невозможность реализовать эффекты прозрачности и просвечивания.

**Вывод:** на основе анализа алгоритмов удаления невидимых поверхностей можно осуществить выбор наиболее подходящего. Одним из основных факторов является время выполнения работы, при этом желательно, чтобы оно не зависело от количества объектов (в данном случае – деталей). Поэтому алгоритм Робертса можно отбросить, так как время возрастает, как квадрат числа объектов. Однако можно заметить, что первый этап может подойти для отрисовки головоломки и её вращении, при этом это не будет затратным по времени. Этого будет достаточно, так как она в таком случае головоломка представляет собой единый предмет без разделения на элементы. При повороте грани такой подход не сможет дать верный результат, так как одни элементы будут перекрывать другие. Для отрисовки поворачиваемой грани предлагается использовать либо алгоритм Z-буфера, либо алгоритм художника. Так как для использования первого необходимо создавать буфер кадра и Z-буфер для всего экрана, то при небольших размерах головоломки это будет неэффективно (также учитывая тот факт, что алгоритм будет применяться для одной грани). Сортировку же по глубине возможно сделать тривиальной. Стоит обратить внимание, что невозможна ситуация, когда грани перекрывают друг друга одновременно. То есть минусы алгоритма художника ликвидируются[[2]](#footnote-2), и можно остановить выбор на нем. Таким образом, был составлен алгоритм из двух этапов:

1. Определение нелицевых граней всего объекта;
2. В случае поворота грани применение для неё алгоритма художника.

## **Методы закрашивания**

# **Конструкторская часть.**

# **Технологическая часть.**

# **Заключение.**

# **Список литературы**

1. [Электронный ресурс]: Википедия
2. С сайта того
3. ПДФка
4. Алгоритм Робертса с того сайта
5. <https://studfile.net/preview/954959/page:8/>
6. <http://compgraph.tpu.ru/zbuffer.htm>
7. Роджерс

1. Все программные продукты, которые будут упоминаться далее, рассмотрены на операционной системе Windows 10 [↑](#footnote-ref-1)
2. Отрисовку всех объектов здесь не следует считать минусом, так как объектов будет не так много [↑](#footnote-ref-2)