

Министерство образования и науки Российской Федерации  
федеральное государственное автономное образовательное учреждение  
высшего образования  
**“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,  
МЕХАНИКИ И ОПТИКИ”**

Работа допущена к защите  
зав. кафедрой

\_\_\_\_\_ Яблочников Е.И., к.т.н., доцент

«\_\_\_\_\_» \_\_\_\_\_ 2015 г.

## **ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К**

**Тема: Реализация программного комплекса ... ЧПУ**

Направление: 230100.62 – Информатика и вычислительная техника

Выполнил студент гр. 4652

\_\_\_\_\_ Крылова Анастасия Андреевна

Научный руководитель,

\_\_\_\_\_ Афанасьев М.Я. к.т.н., доцент

# Оглавление

<b>Список используемых сокращений . . . . .</b>	4
<b>Введение . . . . .</b>	5
<b>Глава 1. Сравнительный анализ существующих систем и предлагаемого подхода . . . . .</b>	8
1.1. Современные системы автоматизированной подготовки УП для гравировальных станков с числовым программным управлением . . . . .	8
1.2. Предлагаемый подход . . . . .	12
1.3. Результаты сравнения . . . . .	14
<b>Глава 2. Модули реализуемого программного комплекса . . . . .</b>	16
2.1. Модуль 3D-моделирования рельефа . . . . .	16
2.1.1. Назначение модуля . . . . .	16
2.1.2. Принципы построения модели . . . . .	17
2.1.3. Выбор формата . . . . .	18
2.2. Модуль редактирования изображений . . . . .	20
2.2.1. Назначение модуля . . . . .	20
2.2.2. Алгоритм изменения яркости . . . . .	21
2.2.3. Алгоритм изменения контраста . . . . .	21
2.3. Модуль генерации управляющих программ . . . . .	23
2.3.1. Назначение модуля . . . . .	23
2.3.2. Реализация модуля . . . . .	24
<b>Глава 3. Демонстрация работы и перспективы развития . . . . .</b>	27
3.1. Демонстрация работы на базе FabLab . . . . .	27

3.2. Демонстрация работы с помощью модуля визуализации систем jViewer и CutViewer Mill . . . . .	29
<b>Заключение . . . . .</b>	<b>31</b>
<b>Список литературы . . . . .</b>	<b>32</b>
<b>Приложение А. Дерево проекта . . . . .</b>	<b>34</b>
<b>Приложение Б. Листинг модуля генерации G-кода . . . . .</b>	<b>35</b>
<b>Приложение В. Управляющая программа . . . . .</b>	<b>37</b>
<b>Приложение Г. Руководство оператора . . . . .</b>	<b>38</b>
Г.1. Назначение . . . . .	38
Г.2. Условия выполнения . . . . .	38
Г.3. Выполнение . . . . .	39
Г.4. Сообщения оператору . . . . .	40
<b>Приложение Д. Руководство программиста . . . . .</b>	<b>41</b>
Д.1. Обращение к программе . . . . .	41
Д.1.1. Функции координирующие работу модулей и клиентской части . . . . .	41
Д.1.2. Модуль генерации УП . . . . .	41
Д.1.3. Модуль генерации 3D-модели . . . . .	42
Д.2. Входные и выходные данные . . . . .	42
Д.3. Сообщения . . . . .	42
<b>Приложение Е. Руководство администратора . . . . .</b>	<b>43</b>
Е.1. Настройка программы . . . . .	43
Е.2. Запуск . . . . .	43
Е.3. Сообщения . . . . .	43

## **Список используемых сокращений**

ПО — Программное обеспечение.

ТПП — Технологическая подготовка производства.

УП — Управляющая программа.

ЧПУ — Числовое программное обеспечение.

ASCII — American standard code for information interchange.

CAM — Computer Aided Manufacturing.

DXF — Drawing eXchange Format.

HTML — HyperText Markup Language.

JS — JavaScript.

JSON — JavaScript Object Notation.

NURBS — Non-uniform rational Bezier spline.

RAM — Random Access Memory.

RML — Roland machine language.

STL — Stereolithography.

SVG — Scalable Vector Graphics.

VRML — Virtual Reality Modeling Language .

# Введение

Технологическая подготовка производства (ТПП) является одним из важнейших этапов жизненного цикла изделия. Она обеспечивает технологическую готовность производства и оказывает значительное влияние на себестоимость конечной продукции. ТПП включает в себя довольно разнообразный спектр задач, решаемых технологом, одной из них является подготовка управляющих программ (УП) для станков с ЧПУ. Стоит отметить, что на реальном производстве практически не бывает случаев, когда УП возможно написать вручную в силу их высокой сложности и объема. Процесс гравировки не является исключением – УП, необходимая для нанесения растрового изображения, состоит из множества однотипных G-кодов, чье количество может исчисляться тысячами.

На данный момент не так много систем автоматизации, специализирующихся на гравировке. Большинство таких программных пакетов поставляются к конкретным станкам, из-за чего эти системы являются неуниверсальными. Стоит заметить, что гравировка может выполняться на универсальном оборудовании, а к нему программное обеспечение, необходимое для нанесения изображения, может не прилагаться вовсе. Программные системы, которые не привязаны к конкретному оборудованию представляют собой массивные десктопные приложения, требующие установки и занимающие большое количество вычислительных ресурсов. Таким образом, очевидно, что вопросы реализации и построения систем автоматизированной подготовки управляющих программ для гравировки **исследованы в недостаточной степени.**

Все вышеперечисленное указывает на **актуальность** рассматриваемой темы и доказывает необходимость применения иного подхода, а именно – применение веб-технологий.

**Объектом исследования** является система автоматизированной подготовки управляющих программ для гравировальных станков с ЧПУ.

В качестве **предмета исследования** рассматриваются методики, алгоритмы и инструменты, применяемые для построения таких систем.

**Целью** данной работы является реализация программного комплекса автоматизированной подготовки управляющих программ для гравировальных станков с ЧПУ. Для достижения поставленной цели были выделены следующие **задачи**:

- Провести сравнительный анализ современных систем автоматизированной подготовки управляющих программ для гравировальных станков и предложенного подхода.
- Разработать алгоритм обработки растрового изображения, с целью получения 3D-модели поверхности.
- Сравнить характеристики форматов 3D-моделей, для поиска наилучшего при применении в веб-сфере.
- Реализовать модуль редактирования изображений.
- Реализовать модуль генерации УП.
- Опробовать разработку на конкретных примерах.

Для решения данных задач было решено применить следующие **инструменты и технологии**: веб-технологии [1], технология разработки веб-приложений с помощью микрофреймворка Flask [2, 3], основные положения функционирования клиент-серверной архитектуры [4], технология отображения 3D-графики в браузере с помощью библиотеки Three.js [5, 6], инструменты обработки и редактирования изображений [7, 8], правила построения управляющих программ для станков с ЧПУ [9].

**Научная новизна** работы состоит в предложенном подходе генерации управляющих программ для гравировальных станков с ЧПУ с использованием веб-технологий.

Также стоит отметить, что автоматизация получения УП имеет значительную **практическую ценность** не только для работ, связанных с декорированием, часто выполняющихся мелкими частными фирмами, но и для крупных предприятий. Примером могут служить гравировка дорожек на печатных платах, нанесение символов на клавиши управляющих стендов и лицевые поверхности приборов.

**Структура** выпускной квалификационной работы. Выпускная квалификационная работа состоит из введения, 3 глав, заключения, библиографии и 6 приложений.

**Во введении** рассматриваются актуальность темы работы, степень ее исследования, а также объект и предмет исследования. Далее сформулированы цель и задачи работы. Указаны инструменты и технологии применяемые для реализации.

**В первой главе** рассматриваются существующие программные продукты, проводится их сравнительный анализ.

**В второй главе** рассмотрены модули реализуемого программного комплекса. Перечень модулей включает в себя: модуль построения 3D-модели рельефа, модуль редактирования изображений, модуль генерации G-кодов.

**В третьей главе** демонстрируется работа разработанного программного комплекса автоматизированной подготовки УП для гравировальных станков с ЧПУ.

**В заключении** приводятся результаты и выводы.

# Глава 1

## Сравнительный анализ существующих систем и предлагаемого подхода

### 1.1. Современные системы автоматизированной подготовки УП для гравировальных станков с числовым программным управлением

В настоящее время существует большое количество САМ-систем, однако лишь малая часть из них пригодна для получения УП для гравировальных станков. Это обуславливается тем, что САМ-системы чаще всего работают с 3D-моделями или векторным представлением изображения, а большинство изображений являются растровыми. Таким образом, можно выделить три способа получения УП из растрового изображения:

- 1) предварительная векторизация изображения и сохранение его в формате DXF или аналогичных, затем передача полученного файла в САМ-систему;
- 2) использование специализированных систем для подготовки УП для гравировальных станков;
- 3) использование программного обеспечения, предлагаемого производителем станка.

**Первый способ.** Первый этап данного способа – векторизация. Векторизация – это довольно трудоемкий и сложный процесс. Степень автоматизации этого процесса чрезвычайно низкая. Трассировка раstra может осуществляться с помощью специальных систем, например Scan2Cad,

Corel Draw, Inkscape или WinTOPО, однако стоит отметить, что их применение не всегда целесообразно. Качественная трассировка получается только в случае очень простых изображений. Если же изображение имеет много деталей, то эти детали теряются ([рис. 1.1, а](#)), либо происходит перегрузка изображения большим количеством пересекающихся контуров ([рис. 1.1, б](#)). Таким образом, в настоящее время многие специалисты предлагают услуги ручной векторизации, которая произовидится художником с помощью векторных редакторов. Стоит отметить, что стоимость таких услуг достаточно высокая.

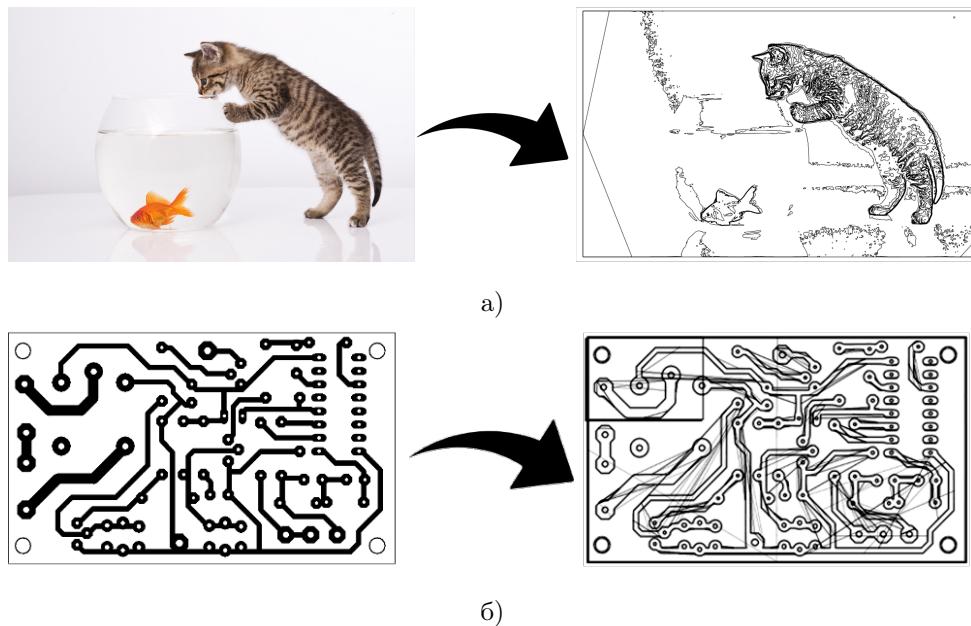


Рисунок 1.1 — Трассировка растральных изображений в программе Inkscape

Предварительная векторизация требуется для того, чтобы в дальнейшем можно было загрузить изображение в САМ-систему.<sup>1</sup> Таким образом, второй этап имеет высокую степень автоматизации. Например, система PyCAM [10] принимает на вход DXF<sup>2</sup> и SVG<sup>3</sup> файлы. Следовательно, получив векторное изображение, можно получить и УП.

<sup>1</sup>CAM – Computer Aided Manufacturing.

<sup>2</sup>DXF (Drawing eXchange Format) – открытый формат файлов для обмена векторной графической информацией между приложениями САПР.

<sup>3</sup>SVG (Scalable Vector Graphics) – это векторный формат, описывающий изображения как фигуры, контуры, текст и эффекты фильтра.

Однако такой двухэтапный подход имеет значительные недостатки:

- высокая стоимость и трудоемкость векторизации;
- потеря деталей при использовании систем с функцией трассировки;
- необходимость использования CAM-систем, которые поддерживают форматы DXF и SVG.

**Второй способ.** Специализированных программных продуктов для получения УП для гравировальных станков не так много. На данный момент самая используемая система – это ArtCAM. Для анализа ArtCAM по характеристикам качества, выбрана модель качества программного обеспечения, представленная в стандарте ISO/IEC 25010:2011 [11]. Она в большей степени, чем остальные аналоги, отражает качественные характеристики с точки зрения пользователя. В ней выделено шесть показателей:

- функциональность;
- надежность;
- удобство;
- эффективность;
- сопровождаемость;
- портируемость.

*Функциональность.* С точки зрения функциональности ArtCAM не уступает своим аналогам. Широкий набор функций позволяет легко подготовить УП для нанесения изображения. ArtCAM имеет встроенный графический редактор, который является довольно полезным инструментом, так как с его помощью можно отредактировать, например, яркость

и контрастность, что в большинстве случаев позволяет нейтрализовать дефекты, порождаемые растровым представлением. Однако поддержка других функций графического редактора скорее приводит к избыточности, потому что обычно для гравировки они не столь востребованы.

Также в ArtCAM присутствуют инструменты для сглаживания 3D-моделей рельефа, что позволяет увидеть прототип будущего изделия и отредактировать его.

*Надежность.* Надежность в данной работе не будет рассмотрена, так как достоверные данные возможно получить лишь при длительном использовании системы.

*Удобство.* ArtCAM построена в виде традиционного десктопного приложения. Таким образом, обычный кнопочный интерфейс с разнообразными панелями инструментов и полем для редактирования является достаточно привычным для пользователя. Тем не менее появляется и типичный недостаток такого подхода – из-за большого количества иконок и панелей не всегда интуитивно понятно, для чего они предназначены. Использование системы становится невозможным без предварительного обучения и чтения справочной документации.

*Эффективность.* Под эффективностью чаще всего подразумевается использование ресурсов. Минимальная версия ArtCAM занимает не слишком много места – около 700 Мб. Однако рекомендуемые характеристики для оборудования довольно высокие. Так, например, требуется процессор не ниже IntelCore i7 (или аналоги) и 16 Гб RAM. Следовательно, данная система будет работать эффективно только на довольно дорогом оборудовании.

*Сопровождаемость.* Как известно, сопровождаемость десктопных приложений – довольно трудоемкая задача, в том числе и для пользователей. В отличие от веб-приложений, где изменения на сервере сразу доступны пользователям, для десктопного приложения потребуется скачать пакет обновлений или, так называемый патч.

*Портируемость.* Основной характеристикой портируемости является мультиплатформенность. ArtCAM поддерживает только операционную систему Windows, а именно Windows версий 7 и 8.

Из вышесказанного следует, что у ArtCAM имеется следующий перечень недостатков:

- высокая стоимость;
- из-за изобилия функций и настроек освоение системы требует времени;
- требования к оборудованию достаточно высокие, что влияет на стоимость ТПП;
- отсутствует мультиплатформенность;

**Третий способ**, а именно – использование утилит предлагаемых производителем используемого станка, чаще всего имеет те же недостатки, что и второй способ. Однако такие утилиты обычно предоставляются бесплатно или частично бесплатно.

## 1.2. Предлагаемый подход

По результатам анализа существующих систем был сделан вывод о том, что для упразднения недостатков необходимо изменить подход к генерации УП. Предлагаемый подход состоит в применении веб-технологий, где система представляет собой веб-приложение.

Веб-приложение построено на клиент-серверной архитектуре ([рис. 1.2](#)), где в качестве клиентской части выступает веб-браузер, а в качестве серверной части – веб-сервер.

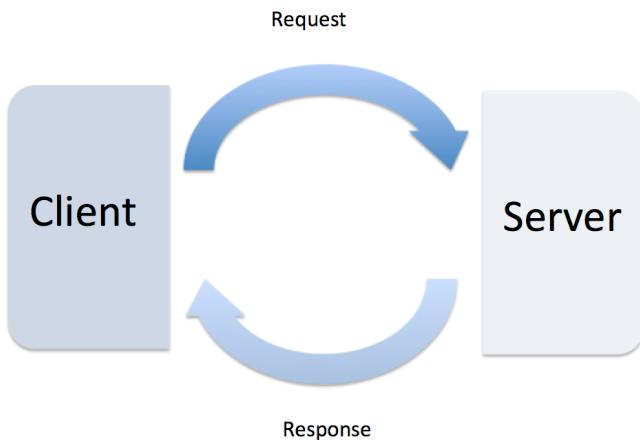


Рисунок 1.2 — Клиент-серверная архитектура

Идея клиент-серверной архитектуры такова: клиент и сервер взаимодействуют друг с другом через компьютерную сеть посредством сетевых протоколов (для веб-приложений это протокол HTTP); клиент в соответствии с действиями пользователя посыпает запросы (чаще всего это HEAD, GET и POST); сервер в соответствии с запросами либо запускает какой-либо процесс, либо предоставляет доступ к своим ресурсам в виде данных. На сервере хранятся статические файлы (js-файлы, файлы каскадных таблиц стилей, html-шаблоны, изображения и т. д.), файл для запуска сервера, файл с конфигурацией сервера, файл с обработчиками запросов и файлы с программами, представляющими функционал системы (программа генерации УП, программа генерации 3D-моделей и т. д.). Такое жесткое разграничение логики, данных и обработчиков позволяет легко отлаживать работу и вносить изменения.

Клиентская часть приложения – браузер, через который пользователь имеет доступ к ресурсам сервера. Также на клиентской части можно реализовать дополнительный функционал системы – редактор загружаемых изображений и просмотрщик 3D-моделей.

Такой подход имеет множество плюсов:

- мультиплатформенность;
- мобильность;
- экономия вычислительных ресурсов и, соответственно, затрат на оборудование;
- отсутствие необходимости установки;
- быстрое обновление системы.

После загрузки изображения на сервер, оно подвергается обработке. В большинстве систем производится векторизация, недостатки которой отмечались ранее. Данный способ подразумевает, что программы, хранящиеся на сервере, обрабатывают параметры пикселей, в результате чего строится 3D-модель и генерируется управляющая программа.

Такой подход позволяет сохранить все детали изображения, а дефекты, вызываемые растровым представлением, устранить посредством настройки яркости и контрастности.

### 1.3. Результаты сравнения

Для подведения итога приводится результирующая таблица сравнения современных систем и предлагаемого подхода.

Из [таблицы 1.1](#) видно, что предлагаемый подход лучше по всем приведенным выше параметрам.

Таблица 1.1 – Результатирующая таблица сравнения современных систем и предлагаемого подхода

Параметры	Двухэтапный способ с векторизацией изображения	ПО для станка ArtCAM (3D-Engrave Software Roland)	Предлагаемый подход
Степень автоматизации	низкая	высокая	высокая
Стоимость	низкая / высокая*	высокая	отсутствует
Мультиплатформенность	присутствует	отсутствует	отсутствует
Сопровождение	отсутствует	присутствует	присутствует
Функциональность	широкая	широкая	узкая

\*Зависит от сложности изображения

## Глава 2

# Модули реализуемого программного комплекса

## 2.1. Модуль 3D-моделирования рельефа

### 2.1.1. Назначение модуля

Построение 3D-модели рельефа важно по двум причинам:

- 3D-модель рельефа позволяет визуализировать будущий результат.
- 3D-модель является входными данными для множества САМ-систем.

Данный модуль реализует обе эти функции (рис. 2.1).

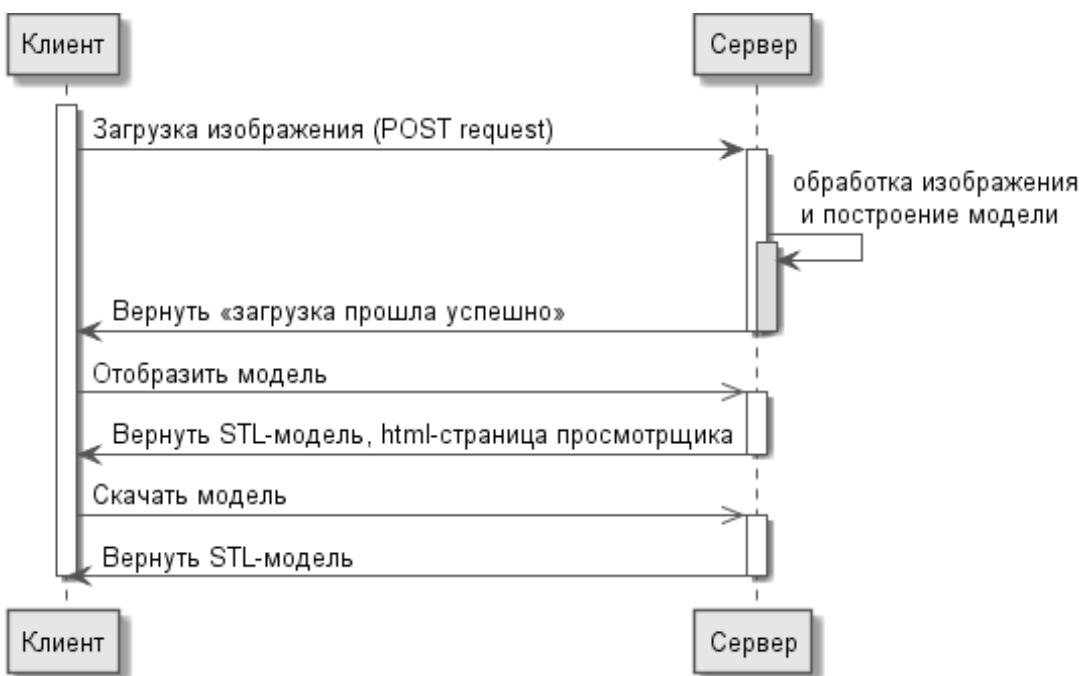


Рисунок 2.1 — Функционирование модуля 3D-моделирования

После загрузки изображения и отправки его на сервер происходит обработка изображения, в результате которой 3D-модель записывается в файл. Затем 3D-модель может быть отображена в браузере с помощью

библиотеки Three.js [6]. Также приложение предоставляет возможность скачивания модели. Такая функциональность может оказаться полезной, например, когда пользователь хочет получить УП не в G-кодах, а в другой нотации (например, RML-код для оборудования фирмы Roland).

### 2.1.2. Принципы построения модели

После загрузки на сервер изображение подвергается обработке. В гравировке вся цветовая информация преобразуется в рельеф с впадинами разной глубины. Для того, чтобы корректно перенести цветное изображение на изделие, его преобразуют в градации серого. Это производится с помощью библиотеки Python Image Library [12], которая позволяет оперировать пикселями изображения.

Цветное растровое изображение состоит из пикселей, каждый из которых характеризуется тремя величинами, определяющими содержание красного, синего и зеленого цветов. Известно, что оттенки серого получаются в результате одинакового содержания этих цветов в пикселе. Таким образом, преобразование в градации серого осуществляется по формуле 2.1:

$$H = \frac{(R + G + B)}{3}, \forall p : \exists p = (R, G, B) \quad (2.1)$$

$$R = \frac{H \times NORM}{256} \quad (2.2)$$

Результирующая величина  $H$  будет лежать в промежутке от 0 до 255, поэтому необходимо провести нормирование по формуле 2.2, в результате которого величина переносится в промежуток от 0 до 3. Затем с шагами  $\Delta x$  и  $\Delta y$  генерируются вершины треугольных граней будущей модели рельефа. В результате работы модуля данные о модели записываются в файл в соответствии с форматом.

### 2.1.3. Выбор формата

Существует множество форматов для хранения 3D-моделей, однако далеко не все из них пригодны для хранения рельефов и отображения их в веб-приложении.

Можно определить два основных критерия, по которым стоит отбирать формат для разрабатываемого приложения:

- совместимость с библиотекой Three.js;
- размер файла модели.

Three.js поддерживает довольно широкий перечень традиционных форматов: STL ASCII,<sup>1</sup> Binary STL,<sup>2</sup> VRML<sup>3</sup> и т. д. Также Three.js предлагает возможность хранения 3D-моделей в JSON<sup>4</sup> формате. Изначально, был выбран формат JSON ([рис. 2.2](#)), так как организация данных в нем наиболее проста и интуитивно понятна, а сам формат изначально пред назначен для представления данных в веб-приложениях. Тем не менее после реализации стало очевидно, что для высокополигональных моделей рельефов такой формат не подходит. Основной проблемой оказался размер файла, который был слишком большим. А это в свою очередь сильно замедляло работу приложения.

Самым оптимальным форматом оказался Binary STL ([рис. 2.3](#)), так как бинарный формат позволял максимально уменьшить размер 3D-модели рельефа.

---

<sup>1</sup>STL ASCII (от англ. stereolithography) – формат файла, используемый для хранения трехмерных моделей объектов, где список треугольных граней и нормалей хранится в обычном текстовом файле.

<sup>2</sup>Binary STL – формат файла, используемый для хранения трехмерных моделей объектов, где список треугольных граней и нормалей хранится в бинарном виде.

<sup>3</sup>VRML (англ. Virtual Reality Modeling Language) – стандартизованный формат файлов для демонстрации трёхмерной интерактивной векторной графики.

<sup>4</sup>JSON (от англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript.



Рисунок 2.2 — Представление модели в формате JSON

00000000	42 69 6e 61 72 79 20 53	54 4c 20 57 72 69 74 65	Binary STL Write
00000010	72 00 00 00 00 00 00	00 00 00 00 00 00 00 00	r.....
00000020	00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
*			
00000050	88 93 00 00 7b 14 2e bd	00 00 00 3e 7b 14 2e bd	....{.....>{...}
00000060	00 00 00 00 00 50 40	00 00 00 00 00 00 00 00	.....P@.....
00000070	48 e1 5a 40 00 00 00 3f	00 00 80 3e 48 e1 5a 40	H.Z@...?...>H.Z@
00000080	00 00 80 3e 00 00 00 00	00 00 00 00 00 00 3e 7b 14	...>.....>{.
00000090	ae bd 00 00 00 00 00 00	50 40 00 00 00 00 00 00	.....P@.....
000000a0	80 3e 48 e1 5a 40 00 00	80 3e 00 00 00 3f 00 00	.>H.Z@...?... .
000000b0	50 40 00 00 00 00 00 00	7b 14 2e 3d 00 00 00 3e	P@.....{.=...>
000000c0	7b 14 2e bd 00 00 00 3f	00 00 50 40 00 00 00 00	{.....?..P@....
000000d0	00 00 80 3e 48 e1 5a 40	00 00 80 3e 00 00 00 3f	...>H.Z@...?...?
000000e0	48 e1 5a 40 00 00 00 3f	00 00 00 00 00 00 00 00	H.Z@...?.....
000000f0	00 3e 00 00 00 00 00 00	00 00 48 e1 5a 40 00 00	.>.....H.Z@... .
00000100	00 3f 00 00 00 3f 48 e1	5a 40 00 00 00 3f 00 00	.?...?H.Z@...?... .
00000110	80 3e 48 e1 5a 40 00 00	80 3e 00 00 0a d7 23 bb	.>H.Z@...?....#. .

} UINT8[80]  
Header

} UINT32  
Number of triangles  
foreach triangle  
REAL32[3] – Normal  
REAL32[3] – Vertex 1  
REAL32[3] – Vertex 2  
REAL32[3] – Vertex 3  
UINT16 – Attribute byte  
end

Рисунок 2.3 — Представление модели в формате Binary STL

Сравнение JSON и Binary STL представлено на [рис. 2.4](#), диаграмма демонстрирует, насколько сокращается время загрузки модели при использовании Binary STL. Диаграмма построена на основе данных, полученных при следующих характеристиках системы:

- виртуальная машина Cloud9;
- 512 Mb RAM;
- 1 GB hard drive;
- канал 10 Мбит/с;
- изображение 180×256.

В результате применения формата STL Binary удается добиться сокращения размера файла более чем вполовину, а время загрузки модели сокращается в 6 раз.

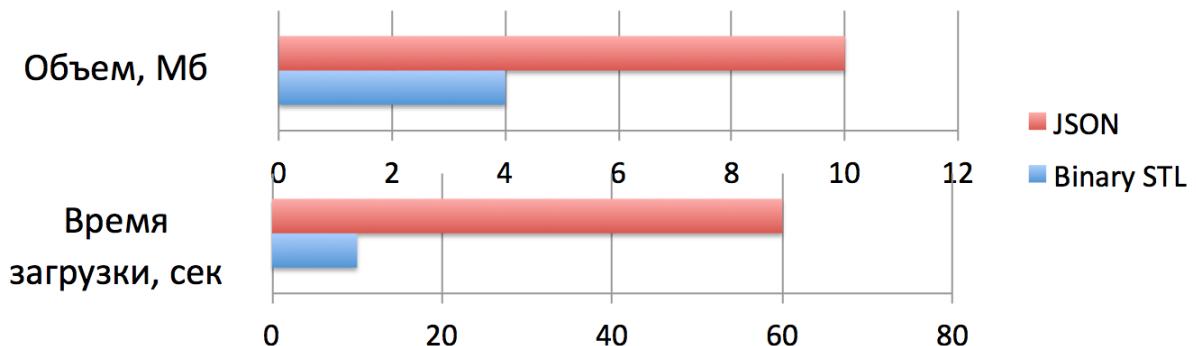


Рисунок 2.4 — Сравнение JSON и Binary STL

## 2.2. Модуль редактирования изображений

### 2.2.1. Назначение модуля

Очевидно, что в некоторых случаях результат, визуализированный с помощью 3D-модели, может по определенным причинам не удовлетворить пользователя. И для того, чтобы получить подходящий результат, необходимо отредактировать входные данные, в качестве которых в разрабатываемом приложении выступает изображение.

Чаще всего причинами неудовлетворительного результата служат:

- низкое качество изображения;
- слишком светлое изображение;
- излишние детали изображения.

Все эти проблемы можно решить минимумом инструментов, а именно – настройками яркости и контрастности. Наиболее распространенная проблема для растровых изображений – это низкое качество. Из-за «пиксельности» изображения по контурам могут возникнуть деформации. Однако настройки контрастности и яркости позволяют избежать нежелательных деформаций ([рис. 2.5](#)).

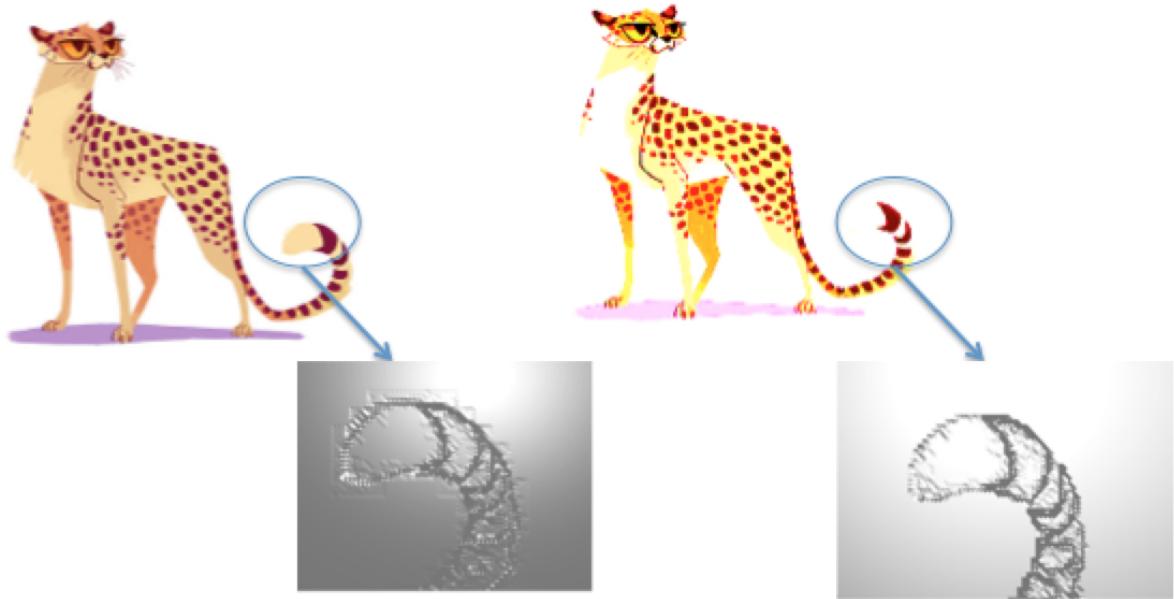


Рисунок 2.5 — Упразднение деформаций в моделях на основе растровых изображений

### 2.2.2. Алгоритм изменения яркости

Для регулировки яркости к каждому каналу каждого пикселя прибавляется определенное значение. Если оно положительное яркость увеличивается, если отрицательное яркость уменьшается ([рис. 2.6](#)).

### 2.2.3. Алгоритм изменения контраста

Для регулировки контраста вначале вычисляется средняя яркость изображения. Для этого вначале вычисляется яркость каждого пикселя. Затем для каждого пикселя находится отклонение от средней яркости. После чего это отклонение умножается на коэффициент контраста. Полученная величина прибавляется к значению каждого канала ([рис. 2.7](#)).

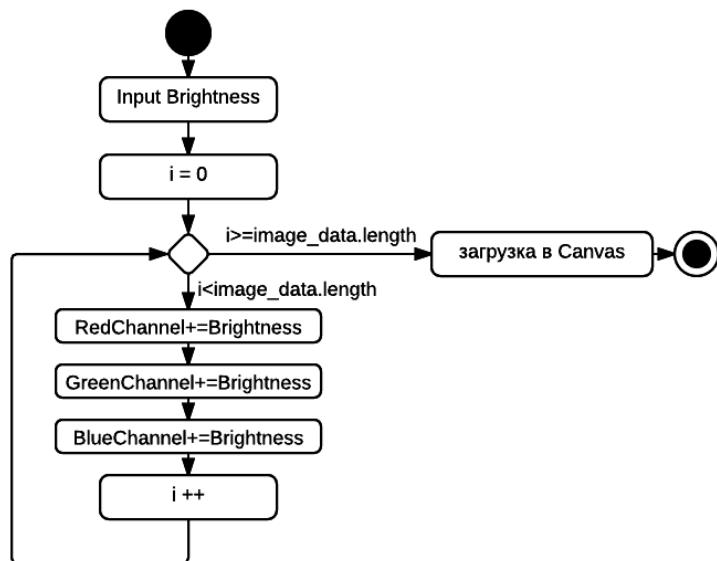


Рисунок 2.6 — Алгоритм изменения яркости

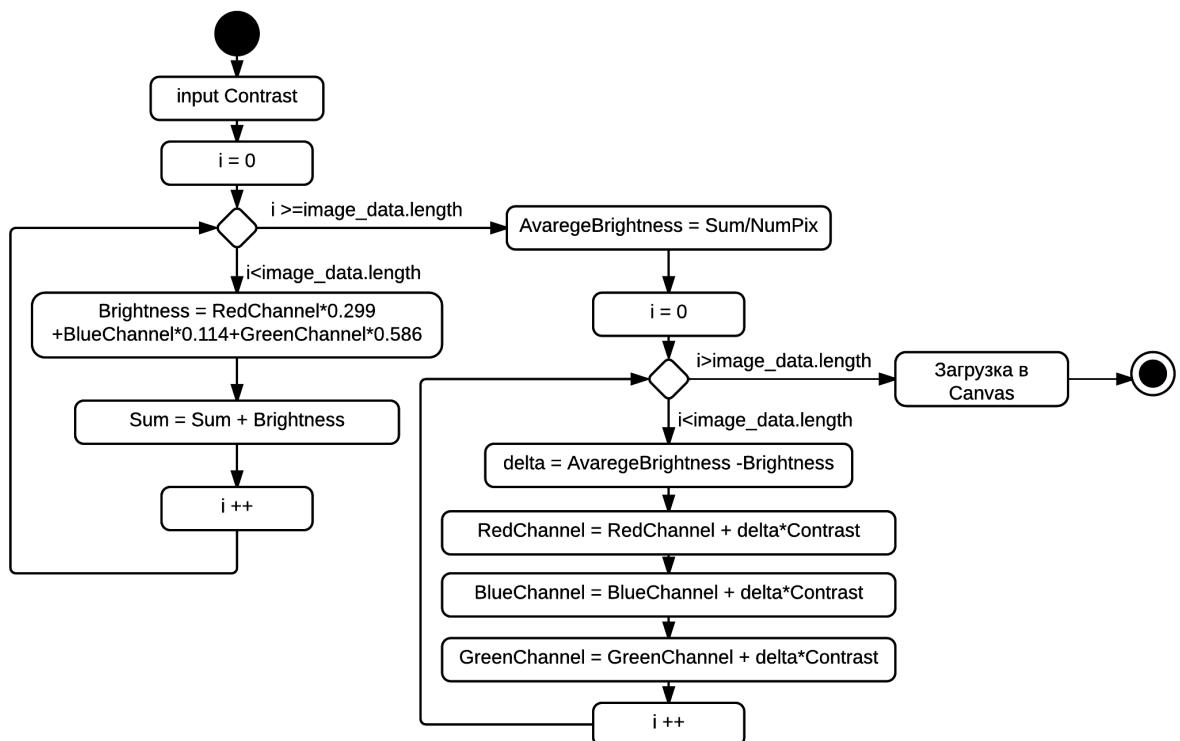


Рисунок 2.7 — Алгоритм изменения контраста

## **2.3. Модуль генерации управляющих программ**

### **2.3.1. Назначение модуля**

Модуль генерации УП предназначен для получения последовательности G-кодов. G-код включает в себя команды управления перемещением инструмента и рабочих органов оборудования, команды определяющие режимы обработки, технологические команды, команды выбора системы координат и единиц измерений [13]. Большинство программного обеспечения, управляющего оборудованием, использует в качестве управляющих программ последовательности G-кодов. Стоит отметить, что сам G-код, определяемый стандартами ISO 6983-1:2009 [14] и ГОСТ 20999-83 [15], может расширяться в зависимости от назначения и производителей оборудования. Фактически, G-код является некоторой основой для реального языка программирования устройств с ЧПУ. Однако использование базовых команд обеспечивает высокую вероятность корректной работы на большинстве видов оборудования.

С точки зрения реализации, растровое изображение проще всего наносить точечно. Тем более, что при таком подходе понадобятся только стандартные G-коды. Таким образом, на основе данных, полученных на этапе обработки изображения и построения модели, генерируется управляющая программа, которая в дальнейшем становится доступной для скачивания с сервера.

### 2.3.2. Реализация модуля

Генератор управляющих программ представляет собой модуль, который хранится на сервере (прил. Б). Внутри данного модуля реализован класс Generator, инициализирующийся следующими значениями:

- карта высот, генерируемая на сервере при обработке изображения;
- высота изображения в пикселях;
- ширина изображения в пикселях;
- величина смещения по осям;
- величина нормирования;
- хэш по имени изображения (рассчитывается по алгоритму Роберта Седжвика [16] при загрузке на сервер).

Также при инициализации данного класса создаются основные блоки, из которых при генерации будет складываться УП:

- преамбула;
- блоки тела УП;
- концевик.

**Преамбула.** Преамбула представляет собой неизменяемый блок. Она состоит из символа начала программы, имени программы, команд смены системы координат и единиц измерений, команды холостого хода, команды начала вращения шпинделя, команд установки скорости вращения шпинделя и скорости рабочей подачи. Блок преамбулы, представленный в листинге 2.1, является списком строк и ставится всегда в начало управляющей программы.

---

Листинг 2.1 — Преамбула

---

```
#пreamble
self.start_block = [
    '%', '001', 'G90', 'G21', 'G00X0Y0',
    'G00Z' + str(0),
    'G00Z' + str(2),
    'G91', 'M03', 'F300.0S6000'
]
```

---

**Блоки тела УП.** Можно выделить три блока тела управляющей программы: блок четных строк, блок нечетных строк и блок перемещений по оси Y. Блоки чередуются и образуют тело управляющей программы, в котором команды отвечают непосредственно за нанесение изображения. Все эти блоки состоят из команд линейной интерполяции с разными координатами. Блоки представляют собой списки списков, где внутренний список включает в себя два строковых литерала: пустой (куда производится подстановка координаты) и литерал с обозначением команды и оси.

Реализация блоков представлена в [листе 2.2](#):

---

Листинг 2.2 — Блоки тела управляющей программы

---

```
#блок для нечетных строк
self.body_block_unevenstr = [
    ['G01X -', ''],
    ['G01Z -', ''],
    ['G01Z', '']
]
#блок для четных строк
self.body_block_evenstr = [
    ['G01X', ''],
    ['G01Z -', ''],
    ['G01Z', '']
]
```

---

Блок перемещения по оси Y состоит из одной команды G01 с параметром равным величине смещения.

**Концевик.** Концевик является неизменяемым блоком, состоящим из трех команд и символа завершения программы. Команды входящие

в блок: команда холостого хода в начало координат, остановка вращения шпинделя, команда конца программы. Концевик представлен в [листинге 2.3](#):

Листинг 2.3 — Блоки тела управляющей программы

---

```
#концевик
self.end_block = [ 'G00X0Y0', 'M05', 'M02', '%' ]
```

---

В процессе инициализации входных параметров и блоков УП производится вычисление глубины врезания для всех точек изображения. Каждый элемент карты высот вычитается из величины нормирования, благодаря чему карта высот преобразуется в карту глубин. Затем производится генерация тела программы из основных блоков посредством их чередования и подстановки параметров из карты глубин. Стоит отметить, что во время генерации происходит оптимизация. Она заключается в том, что части изображения с белым фоном не обрабатываются, и инструмент смещается до ближайшего не белого участка. Это позволяет заметно уменьшить количество G-кодов в УП и ускорить процесс обработки.

Результат работы модуля записывается в обычновенный текстовый файл ([прил. В](#)), который становится доступным для скачивания.

Стоит отметить, что кроме вышеуказанного подхода, использующего линейную интерполяцию, существует подход использующий NURBS.<sup>5</sup> С помощью NURBS можно описать поверхности любой сложной формы. Однако очевидно, что для того, чтобы перевести растровое изображение в NURBS-геометрию потребуется трудоемкий процесс поиска контрольных узлов, касательных, весов и т. д. К тому же на данный момент синтаксис NURBS [13] в нотации G-кодов недостаточно стандартизирован, а многими станками не поддерживается вовсе.

---

<sup>5</sup>NURBS (англ. Non-uniform rational Bezier spline) – математическая форма, применяемая в компьютерной графике для генерации и представления кривых и поверхностей.

## Глава 3

# Демонстрация работы и перспективы развития

## 3.1. Демонстрация работы на базе FabLab

В FabLab Университета ИТМО на станке Roland MDX-40A ([рис. 3.1](#)) была испытана работоспособность модуля 3D-моделирования рельефа.

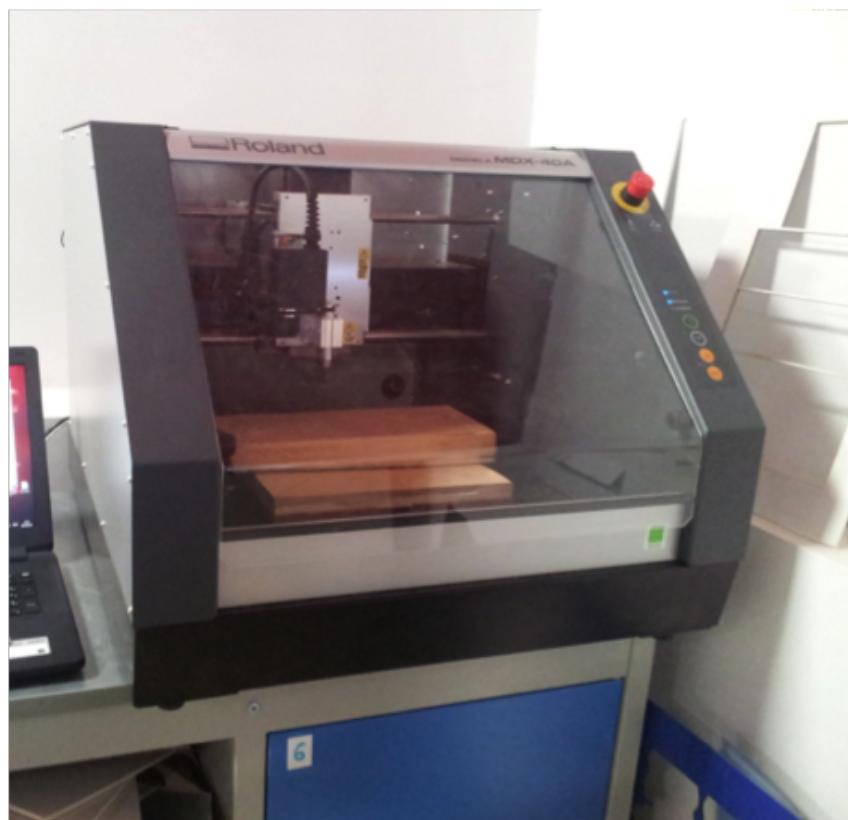


Рисунок 3.1 — Гравировально-фрезерный станок Roland MDX-40A

После загрузки изображения и построения 3D-модели в формате STL ([рис. 3.2](#)), модель скачивается на персональный компьютер, подключенный к станку.

На данном компьютере установлено программное обеспечение фирмы Roland – SPR Player (Subtractive Rapid Prototype) [17]. SPR Player – это САМ-система, которая позволяет преобразовывать модели форматов



Рисунок 3.2 — Сгенерированная модель

IGES<sup>1</sup>, STL, 3DM<sup>2</sup> в управляющую программу. При этом управляющая программа состоит не из G-кодов, а из команд в нотации RML (Roland Machine Language). Пример команд в нотации RML приведен в листинге 3.1:

Листинг 3.1 — Управляющая программа в нотации RML

---

```
^DF
! NR
! 1;
V 30;
Z 0,0,50;
Z 33458,270,50;
V 7;
Z 33458,270,-100;
V 30;
Z 0,270,-100;
V 7;
Z 0,270,-200;
V 30;
Z 33458,270,-200;
^DF;
```

---

<sup>1</sup>IGES (Initial Graphics Exchange Specification) – двумерный/трехмерный векторный формат графики

<sup>2</sup>3DM – используется для передачи геометрии NURBS.

После получения управляющей программы, была проведена черновая обработка, в результате чего было получено следующее изделие (рис. 3.3):



Рисунок 3.3 — Результат обработки

### 3.2. Демонстрация работы с помощью модуля визуализации систем jViewer и CutViewer Mill

Для того, чтобы продемонстрировать корректность работы модуля генерации управляющих программ, необходимо визуализировать процесс обработки по получаемому G-коду.

Изначально визуализация была проведена в программе jViewer [18]. jViewer – это бесплатное приложение, позволяющее представить результат работы управляющей программы в виде каркасной модели. Тем не менее каркасная модель не так наглядна, как твердотельная модель. Результат визуализации представлен на рис. 3.4:

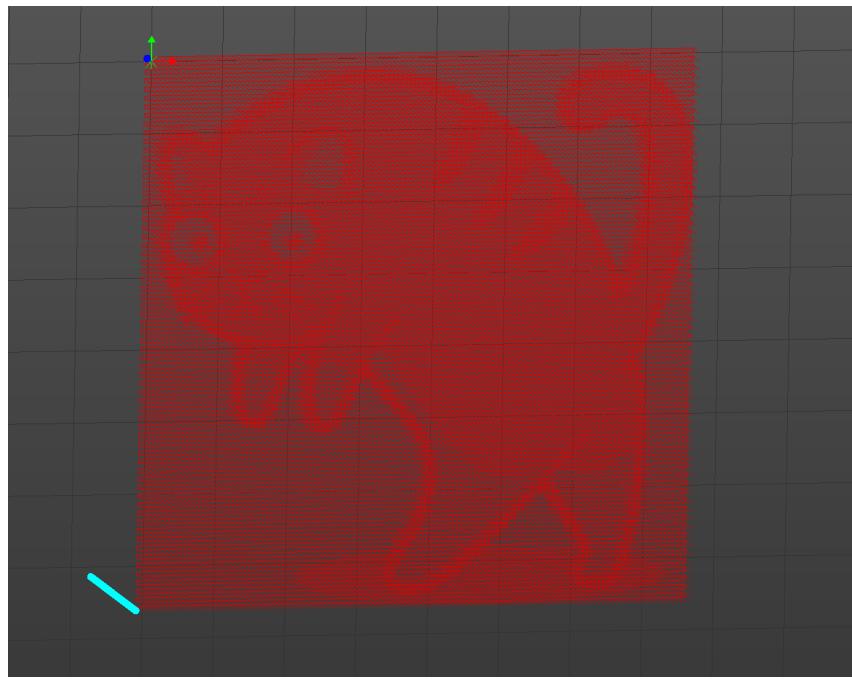


Рисунок 3.4 — Результат визуализации в jViewer

Чтобы визуализировать не только результат, но и процесс обработки, может быть использована система CutViewer Mill [19]. CutViewer Mill – платное приложение, но разработчики предоставляют бесплатную версию на 30 дней. CutViewer Mill позволяет назначить инструмент, а также форму заготовки. В результате визуализации была получена твердотельная модель, которая представлена на [рис. 3.5](#):

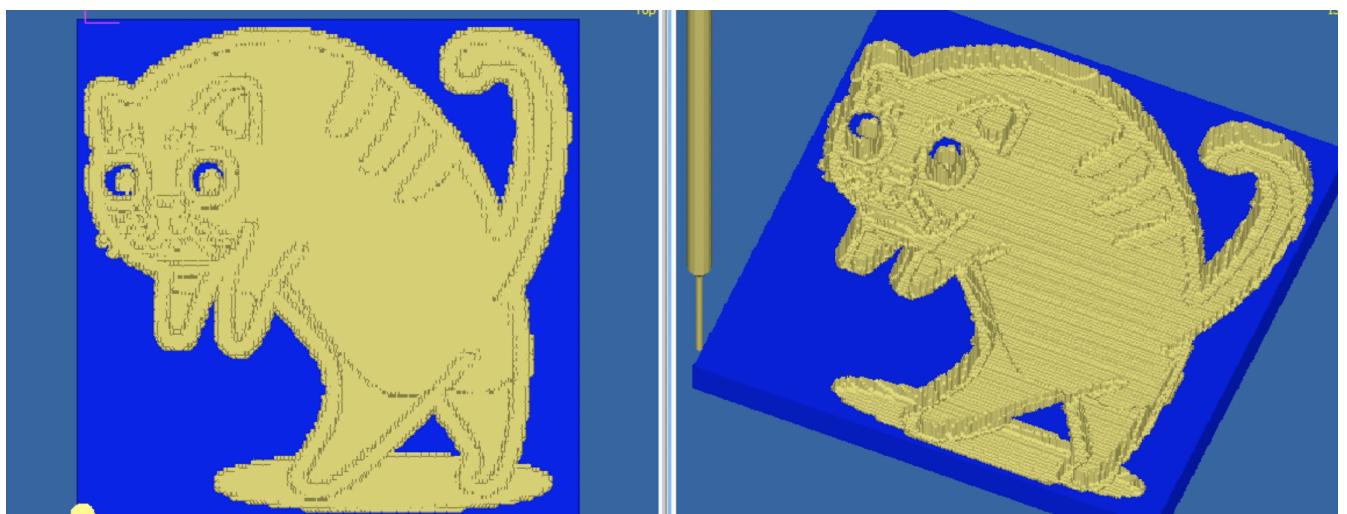


Рисунок 3.5 — Результат визуализации в CutViewer Mill

# Заключение

В выпускной квалификационной работе был реализован программный комплекс автоматизированной подготовки управляющих программ для гравировальных станков с ЧПУ. При разработке программного комплекса был применен новый подход к генерации управляющих программ. В ходе выполнения работы были достигнуты следующие результаты:

- Проведен сравнительный анализ современных систем автоматизированной подготовки управляющих программ для гравировальных станков и предложенного подхода.
- Разработан алгоритм обработки растрового изображения, с целью получения 3D-модели поверхности.
- Найден наилучший формат 3D-моделей для применения в веб-сфере.
- Реализован модуль редактирования изображений.
- Реализован модуль генерации УП.
- Разработка опробована на конкретных примерах.

Предлагаемый подход обеспечил возможность генерировать управляющие программы на основе растровых изображений, имея только выход в интернет. А программный комплекс, разработанный для поставленной цели, оказался лучше аналогичных программных продуктов по большинству показателей качества.

## Список литературы

1. Пьюривал С. Основы разработки веб-приложений. Питер, 2015 г. 272 с.
2. The Flask Mega-Tutorial [Электронный ресурс]. URL: <http://habrahabr.ru/post/193242/> (дата обращения: 20.02.2015).
3. Гринберг М. Разработка веб-приложений с использованием Flask на языке Python. ДМК Пресс, 2014. 272 с.
4. Designing an MVC Model for Rapid Web Application Development // Procedia Engineering. 2014. Vol. 69, no. 0. P. 1172 – 1179. 24th International Symposium on Intelligent Manufacturing and Automation, 2013.
5. Evans A., Romeo M., Bahrehamd A. et al. 3D graphics on the web: A survey // Computers & Graphics. 2014. Vol. 41, no. 0. P. 43 – 61.
6. Three.js docs [Электронный ресурс]. URL: <http://threejs.org/> (дата обращения: 20.02.2015).
7. Fulton S. HTML5 Canvas, 2nd Edition. O'Reilly Media, 2012. 750 p.
8. Флэнаган Д. JavaScript. Подробное руководство. Символ-Плюс, 2008. 992 с.
9. Аверченков В. И. Автоматизация подготовки управляющих программ для станков с ЧПУ. 2011. 19 с.
10. PyCAM documentation [Электронный ресурс]. URL: <http://sourceforge.net/projects/pycam/> (дата обращения: 03.06.2015).
11. ISO/IEC 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. British Standards Institution, 2011. 44 p.
12. Sathaye N. Python Multimedia. Packt Publishing, 2010. 292 p.
13. Suh S.-H., Kang S.-K., Chung D.-H., Stroud I. Theory and Design of CNC Systems. Springer-Verlag, 2008. 455 p.

14. ISO 6983-1:2009 Automation systems and integration. Numerical control of machines. Program format and definitions of address words. Part 1: Data format for positioning, line motion and contouring control systems. International Organization for Standardization, 2009. 34 p.
15. ГОСТ 20999-83 Устройства числового программного управления для металлообрабатывающего оборудования. Кодирование информации управляющих программ. Госстандарт России, 1983. 28 с.
16. Azevedo P. Algorithms in C: Robert Sedgewick, Addison-Wesley Publish Company, 1990. pp. 670, hardcover, BJ19.95. ISBN: 0 201 51425 7 // Advances in Engineering Software. 1992. Vol. 15, no. 1. P. 73 – 74.
17. SPR Player Software Guide[Электронный ресурс]. URL: <http://www.rolanddga.com> (дата обращения: 03.06.2015).
18. jViewer Sofrware [Электронный ресурс]. URL: <http://www.thingiverse.com/thing:72324> (дата обращения: 03.06.2015).
19. CutViewer Mill Sofrware [Электронный ресурс]. URL: <http://www.cutviewer.com> (дата обращения: 03.06.2015).

# Приложение А

## Дерево проекта



## Приложение Б

### Листинг модуля генерации G-кода

```
1 # -*- coding: utf-8 -*-
2 #python2.7
3 import copy
4 import readimage
5
6 class Generator(object):
7
8     def __init__(self, offset,
9                  norm, name, name_image):
10        self.name = name
11        self.norm = norm
12        heightmap, self.h, self.w = \
13            readimage.read_image(name_image, norm)
14        self.depth = self._depth(heightmap, self.h, self.w)
15        self.offset = offset
16        self.offset_counter = 0.0
17        self.start_block = [
18            '%', '001', 'G90', 'G21', 'G00X0Y0',
19            'G00Z' + str(0),
20            'G00Z' + str(2),
21            'G91', 'M03', 'F300.0S6000',
22        ]
23        #блок для нечетных строк
24        self.body_block_unevenstr = [[ 'G01X- ', ''],
25                                     [ 'G01Z- ', ''],
26                                     [ 'G01Z', '' ]]
27        #блок для четных строк
28        self.body_block_evenstr = [[ 'G01X', ''],
29                                     [ 'G01Z- ', ''],
30                                     [ 'G01Z', '' ]]
31        self.end_block = [ 'G00X0Y0', 'M05', 'M02', '%' ]
32
33    def generate(self):
34        """Generate G-codes list"""
35        res = []
36        for i in xrange(self.h):
37            for j in xrange(self.w):
38                if (i%2 == 0) and \
39                    (round(float(self.depth[i][j]), 2)<=0.03):
40                    self.offset_counter += self.offset
41                    if j == self.w-1:
42                        res+=['G01X' + str(self.offset_counter)]
43                        continue
44                    if (i%2 == 1) and \
45                        (round(float(self.depth[i][self.w-j-1]), 2)<=0.03):
46                        self.offset_counter += self.offset
```

```

47             if j == self.w-1:
48                 res+=['G01X-' + str(self.offset_counter)]
49                 continue
50
51             if i%2 == 0:
52                 res += self._substitution(
53                     self.body_block_evenstr,
54                     i, j)
55             if i%2 == 1:
56                 res += self._substitution(
57                     self.body_block_unevenstr,
58                     i, self.w-j-1)
59             self.offset_counter = self.offset
60             self.offset_counter = self.offset
61             res += ['G01Y-' + str(self.offset)]
62         for item in self.end_block: res.append(item)
63         for item in self.start_block[::-1]: res.insert(0,item)
64     return res
65
66 #подставляем глубину в списки вида [ 'G01Z' , ' ' ]
67 def _substitution(self,block,i,j):
68     block = copy.deepcopy(block)
69     for item in block:
70         if (isinstance(item,list) and
71             item[0].startswith('G01Z')):
72             item[1] = str(float(self.depth[i][j]) + 2)
73         else:
74             item[1] = str(self.offset_counter)
75             #item[1] = str(self.offset)
76     return block
77
78 def write(self,gcode_list):
79     """Writes G-codes in file g_codes.txt"""
80     with open(self.name, 'w') as f:
81         for item in gcode_list:
82             if len(item) == 1:
83                 f.write(item + '\n')
84             else:
85                 f.write(''.join(item) + '\n')
86
87 def _depth(self,heightmap,h,w):
88     temp_depth = []
89     for i in xrange(h):
90         temp_depth.append( [ str(self.norm - \
91             float(heightmap[i*w + j])) for j in xrange(w) ] )
92     print(len(temp_depth))
93     return temp_depth

```

---

## Приложение В

### Управляющая программа

---

```
1 %
2 001
3 G90
4 G21
5 G00X0Y0
6 G00Z0
7 G00Z2
8 G91
9 F300.0 S6000 M03
10 G01Y0.3
11 G01Y0.3
12 G01Y0.3
13 ...
14 G01X48.3
15 G01Z-2.05
16 G01Z2.05
17 G01X2.1
18 G01Z-2.05
19 G01Z2.05
20 G01X5.1
21 G01Z-2.05
22 G01Z2.05
23 G01X0.3
24 G01Z-2.06
25 G01Z2.06
26 G01X0.9
27 G01Z-2.05
28 G01Z2.05
29 ...
30 G01Y0.3
31 G01Y0.3
32 G01Y0.3
33 G01Y0.3
34 G01Y0.3
35 G00X0Y0
36 M05
37 M02
38 %
```

---

## Приложение Г

# Руководство оператора

### Г.1. Назначение

Программный комплекс автоматизированной подготовки управляющих программ для гравировальных станков с ЧПУ предназначен для:

- генерации управляющих программ на основе растровых изображений;
- генерации 3D-моделей рельефов на основе растровых изображений;
- просмотра генерируемых 3D-моделей рельефов;
- редактирования изображений.

### Г.2. Условия выполнения

Для работы программного комплекса требуется выполнение следующих условий:

- наличие оборудования с подключением к сети;
- наличие веб-браузера с поддержкой технологий WebGL и Canvas.

### Г.3. Выполнение

Работа с программным комплексом начинается с открытия главной страницы ([рис. Г.1](#)). Для выбора и загрузки изображения требуется нажать на кнопки 1 и 2 соответственно, при этом надпись «файл не выбран» поменяется на название загруженного файла. Далее изображение можно отредактировать, нажатие на кнопку 3 перенаправляет пользователя на страницу редактирования ([рис. Г.3](#)). Для того, чтобы скачать 3D-модель рельефа и УП, нужно нажать на кнопки 5 и 6 соответственно. Кнопка 4 перенаправляет нас на просмотрщик 3D-моделей ([рис. Г.2](#)).

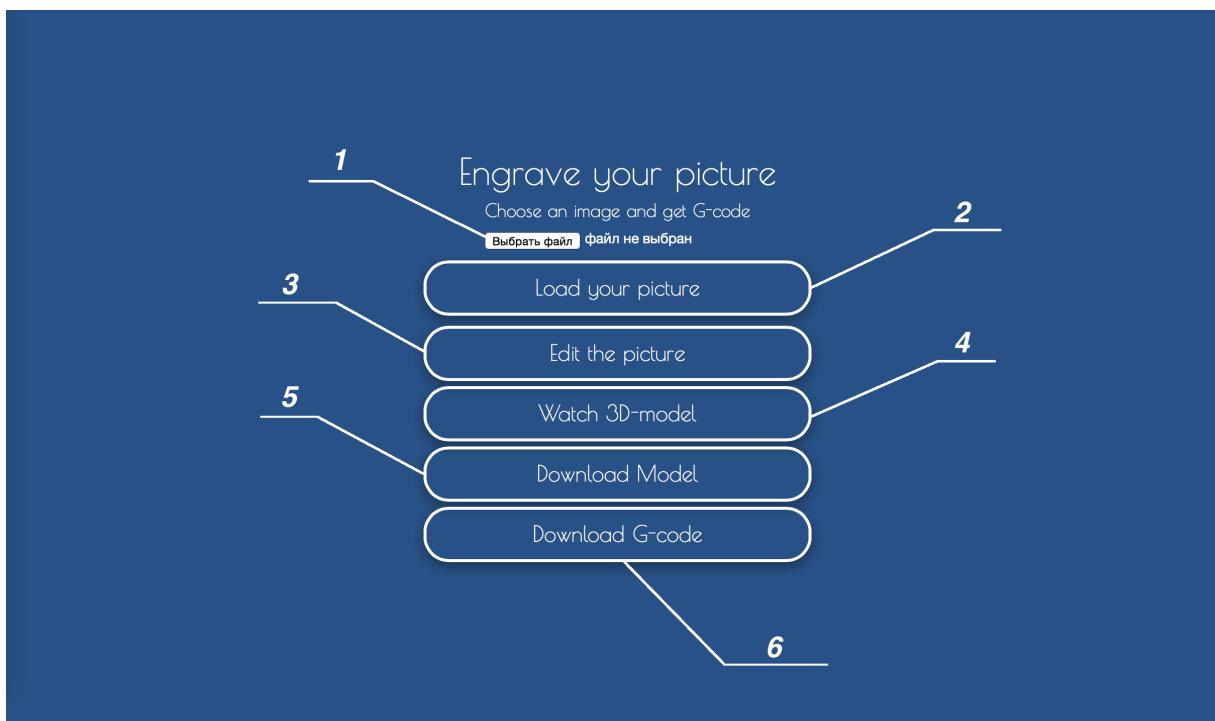


Рисунок Г.1 — Главная страница приложения

Страница редактора изображений ([рис. Г.3](#)) в левом верхнем углу имеет три ползунка, которые позволяют редактировать размер, яркость и контраст, и кнопку сохранения изменений.

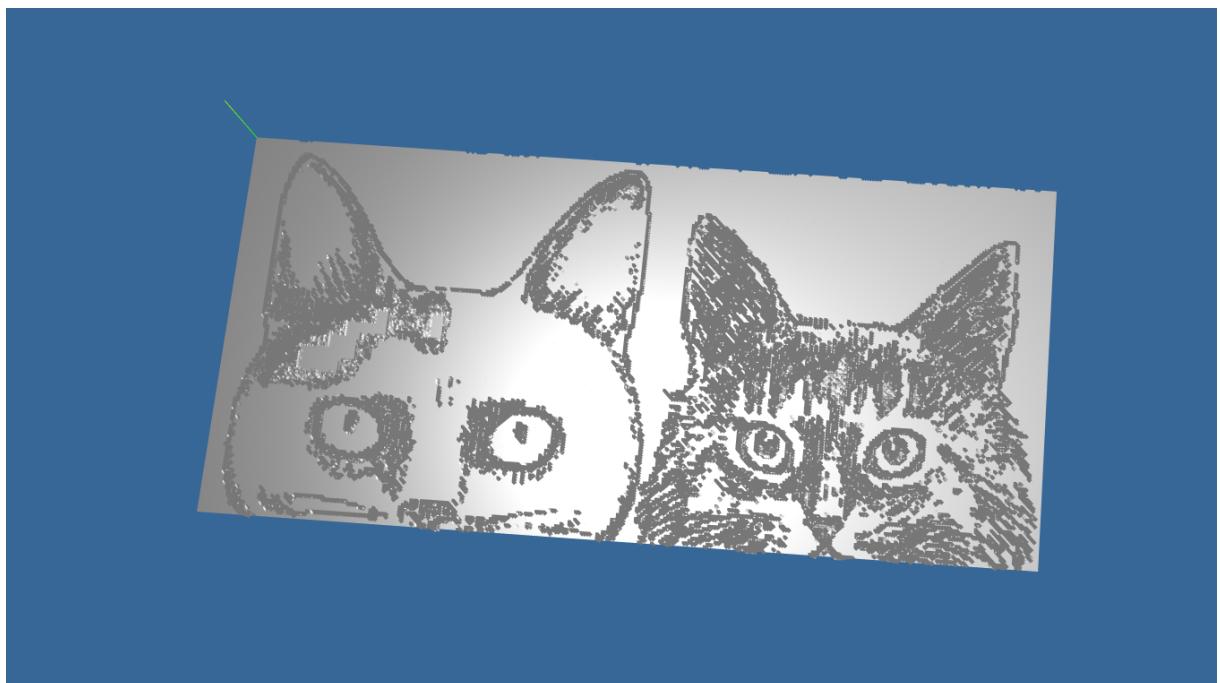


Рисунок Г.2 — Страница просмотра модели

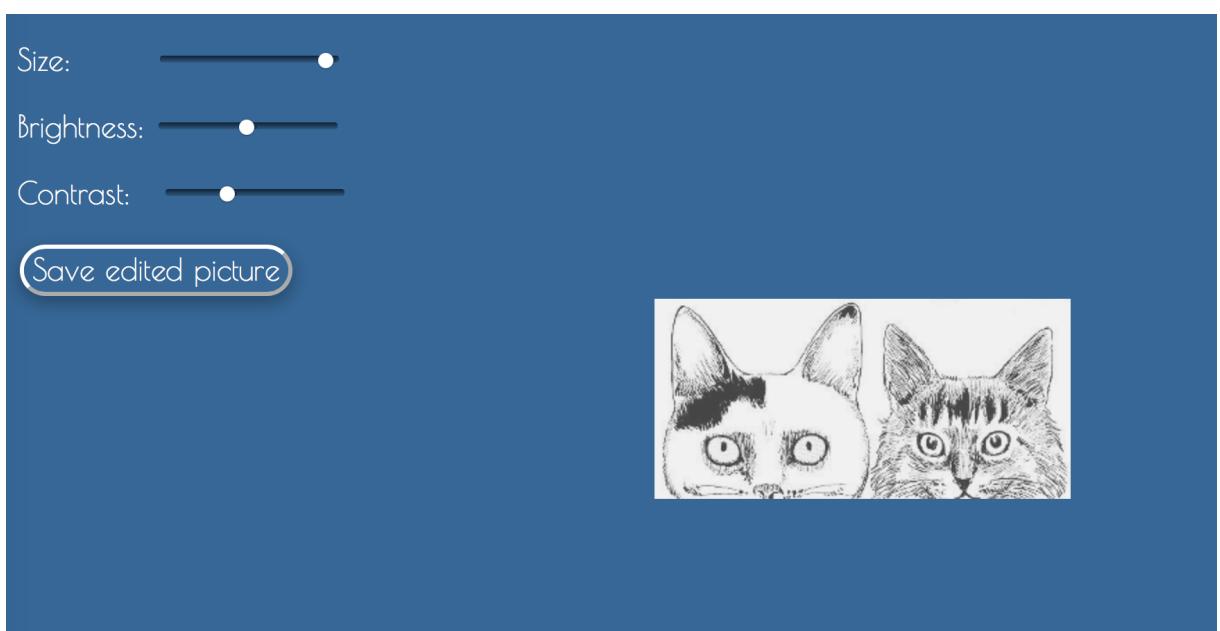


Рисунок Г.3 — Страница редактора изображений

#### Г.4. Сообщения оператору

При успешной загрузке изображения, оператору выводится сообщение «Success!».

## Приложение Д

# Руководство программиста

## Д.1. Обращение к программе

### Д.1.1. Функции координирующие работу модулей и клиентской части

Основные модули хранятся в директории с названием python\_scripts.

Работа приложения координируется следующими функциями:

index() — Загрузка главной страницы.

show\_editor() — Загрузка страницы редактора изображений.

upload\_edited\_pic() — Сохранение отредактированного изображения.

model() — Загрузка страницы отображения 3D-модели.

upload\_file() — Загрузка изображения и построение модели.

download\_file() — Отправка УП на скачивание.

download\_model() — Отправка 3D-модели на скачивание.

### Д.1.2. Модуль генерации УП

Исходный текст модуля генерации УП представлен в [прил. Б](#). Модуль состоит из единственного класса Generator, который имеет следующие методы:

\_\_init\_\_() — Конструктор класса.

generate() — Генератор УП.

\_substitution() — Подстановка параметров в стандартные блоки УП.

write() — Запись УП в текстовый файл.

\_depth — Вычисление глубины врезания.

### **Д.1.3. Модуль генерации 3D-модели**

Модуль состоит из функции `make_stl_model()`, которая осуществляет генерацию модели и ее запись в файл, и класса `Triangle`. Класс `Triangle` служит для представления треугольных граней в бинарном виде.

Подробное описание не приводится, так как программный код является самодокументированным и с большим количеством комментариев.

## **Д.2. Входные и выходные данные**

Входными данными являются растровые изображения. Выходными данными — STL-модели и текстовые файлы. В модуле редактирования изображения выходными данными служат изображения в кодировке Base64.

## **Д.3. Сообщения**

Сообщения для программиста – стандартные сообщения фреймворка Flask и языка Python.

## Приложение Е

# Руководство администратора

### E.1. Настройка программы

Для настройки и запуска разработанного программного комплекса необходимо отредактировать файл run.py. Его расположение в дереве проекта представлено в [прил. А](#).

В файле run.py задается имя хоста и прослушиваемого порта, а также указывается режим функционирования сервера. Для этого нужно отредактировать параметры host, port и debug соответственно.

---

Листинг Е.1 — Файл конфигураций run.py

---

```
1 #python2.7
2 from app import app
3 app.run(host = '0.0.0.0', port = 8080, debug = True)
```

---

### E.2. Запуск

Для запуска работы сервера программного комплекса необходимо запустить файл run.py.

---

Листинг Е.2 — Запуск сервера в терминале

---

```
>>python run.py
```

---

### E.3. Сообщения

Сообщения для администратора – стандартные сообщения фреймворка Flask, отображаемые в терминале.