

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 ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

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

Санкт-Петербург – 2015

Оглавление

Введение	4
Глава 1. Сравнительный анализ современных систем с предлага- емым подходом	7
1.1. Современные системы автоматизированной подготовки УП для гравировальных станков с ЧПУ	7
1.2. Предлагаемый подход	11
1.3. Результаты сравнения	13
Глава 2. Модули реализуемого программного комплекса	14
2.1. Модуль 3D-моделирования рельефа	14
2.1.1. Назначение модуля	14
2.1.2. Принципы построения модели	15
2.1.3. Выбор формата	15
2.2. Модуль редактирования изображений	17
2.2.1. Назначение	17
2.2.2. Алгоритм изменения яркости	18
2.2.3. Алгоритм изменения контраста	18
2.3. Модуль генерации управляющих программ	20
2.3.1. Назначение модуля	20
2.3.2. Реализация модуля	21
Глава 3. Демонстрация работы и перспективы развития	24
3.1. Демонстрация работы на базе FabLab	24
3.2. Демонстрация работы с помощью модуля визуализации системы РуСАМ	26
3.3. Перспективы развития разработки	26

Заключение	27
Список литературы	28
Приложение А. Дерево проекта	29
Приложение Б. Листинг модуля генерации G-кода	30
Приложение В. Управляющая программа	32

Введение

Технологическая подготовка производства 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, включает в себя множество 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 такие системы являются неуниверсальными. Стоит заметить, что гравировка может выполняться на универсальном оборудовании, а к нему необходимое для нанесения изображения программное обеспечение 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 УП 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. А в качестве **предмета исследования** рассматриваются методики, инструменты и алгоритмы,

применяемые для построения таких систем.

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

- Провести сравнительных анализ современных систем 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, основы построения веб-приложений с помощью микрофреймворка Flask[1, 2], основы функционирования клиент-серверной архитектуры, основы отображения 3D-графики в браузере с помощью библиотеки Three.js[3], методы обработки и редактирования изображений[4].

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 работы представляет собой совокупность следующих пунктов: введение, 3 главы, заключение, библиография.

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 анализ их недостатков. Предлагается способ их упразднения.

Во 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 рассматриваются перспективы применения данной разработки.

В заключении приводится перечень достигнутых результатов.

Глава 1

Сравнительный анализ современных систем с предлагаемым подходом

1.1. Современные системы автоматизированной подготовки УП для гравировальных станков с ЧПУ

В настоящее время существует большое количество САМ-систем, однако 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, в настоящее время можно выделить три способа получения УП из растрового изображения:

- Предварительная векторизация изображения и сохранение его в формате DXF или аналогичных. Затем, подача полученного файла в САМ-систему.
- Использование специализированных систем для подготовки УП для гравировальных станков.
- 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 WinTOPO, 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), либо происходит перегрузка изображения большим количеством пересекающихся контуров (рис. 1.2). Таким образом, в настоящее время многие специалисты предлагают услуги ручной векторизации, которая производится художником вручную с помощью векторных редакторов. Стоит отметить, что стоимость таких услуг достаточно высокая.

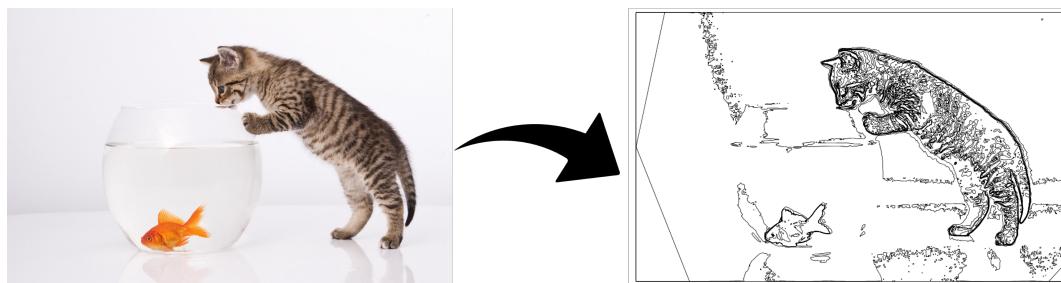


Рис. 1.1. Трассировка растрового изображения в программе Inkscape

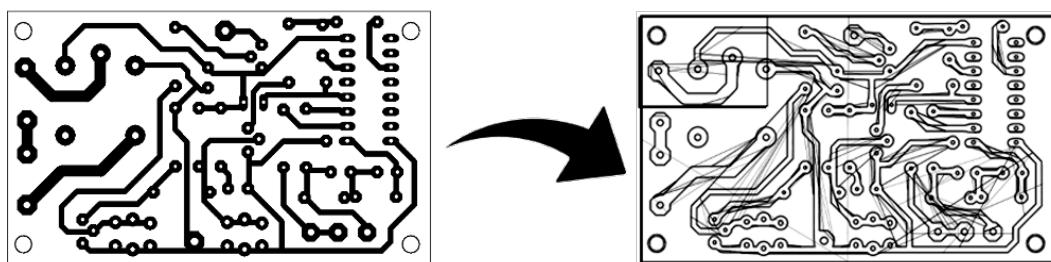


Рис. 1.2. Трассировка растрового изображения разводки платы в программе Inkscape

Предварительная векторизация требуется для того, чтобы в дальнейшем можно было загрузить изображение в CAM-систему. Таким образом, второй этап имеет высокую степень автоматизации. Например, система РуСАМ принимает на вход DXF и SVG файлы. Следовательно, получив векторное изображение, можно получить и УП.

Однако такой двухэтапный подход имеет значительные недостатки:

- высокая стоимость и трудоемкость векторизации;
- потеря некоторых деталей в случае использования систем с функцией трассировки;
- использование только таких 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 система ArtCAM. Для анализа ArtCAM по основным характеристикам качества, выбирается модель качества программного обеспечения, представленная в наборе стандартов ISO/IEC 25010:2011, поскольку именно 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, так как они для гравировки не являются востребованными. Также в ArtCAM присутствуют инструменты сглаживания непосредственно на 3D-модели рельефа, что позволяет сразу 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. Однако 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 потребляемость 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. Таким образом, 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 отличии 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 платформу Windows, а именно Windows версий 7 и 8.

Из выше сказанного следует, что ArtCAM имеет следующие недостатки:

- 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 чаще всего предоставляются бесплатно или частично бесплатно.

1.2. Предлагаемый подход

После краткого 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. 1.3](#)), где в качестве клиента выступает веб-браузер, а в качестве сервера – веб-сервер.

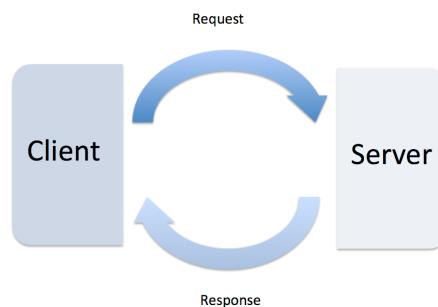


Рис. 1.3. Клиент-серверная архитектура

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([прил. А](#)).

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. В данном подходе программы, хранящиеся на сервере, проделывают манипуляции с пикселями, 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.

Таблица 1.1

Результирующая таблица сравнения современных систем и предлагаемого подхода

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

Из данной таблицы видно, что предлагаемый подход лучше по всем приведенным выше параметрам.

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). После загрузки изображения в специальное поле в приложении 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. 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).

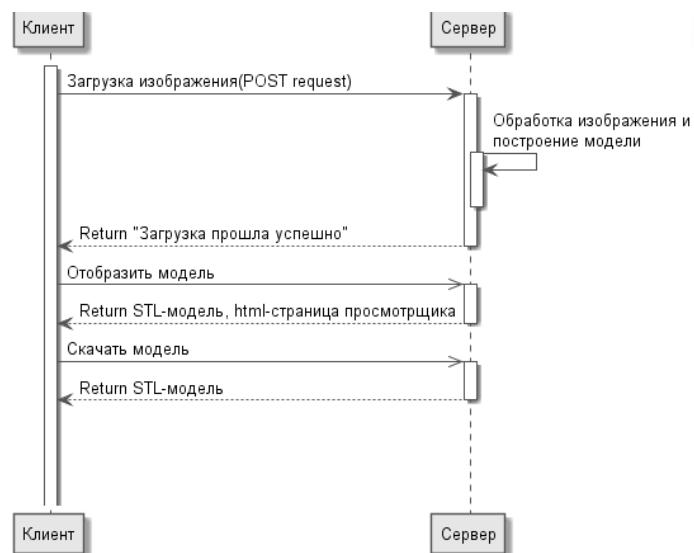


Рис. 2.1. Функционирование модуля 3D-моделирования.

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[5], 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 следующей схеме:

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

Результирующая величина H будет лежать в промежутке от 0 до 255, поэтому необходимо провести нормирование, в результате которого величина переносится в промежуток от 0 до 3. Затем, с шагом Δx и Δy 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:

- поддерживаемость библиотекой Three.js;
- размер файла с моделью.

Three.js поддерживает довольно широкий перечень традиционных форматов: STL ASCII, Binary STL, VRML и т.д. Также Three.js предлагает возможность хранения 3D-моделей в JSON формате. Изначально, был выбран формат JSON (рис. 2.2), так как организация данных в нем наиболее проста и интуитивно понятна, а сам формат изначально предназначен для представления данных в веб-приложениях. Однако 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.

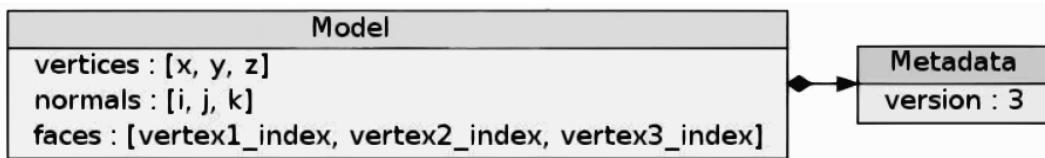


Рис. 2.2. Представление модели в формате JSON.

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

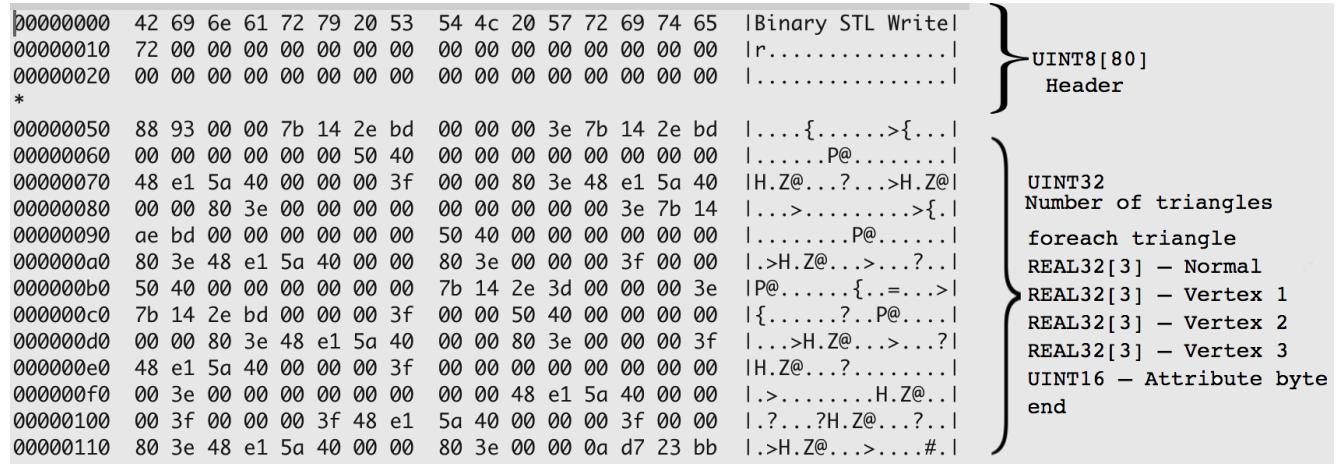


Рис. 2.3. Представление модели в формате Binary STL.

Для сравнения форматов приведена диаграмма (рис. 2.4), которая 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 180x256.

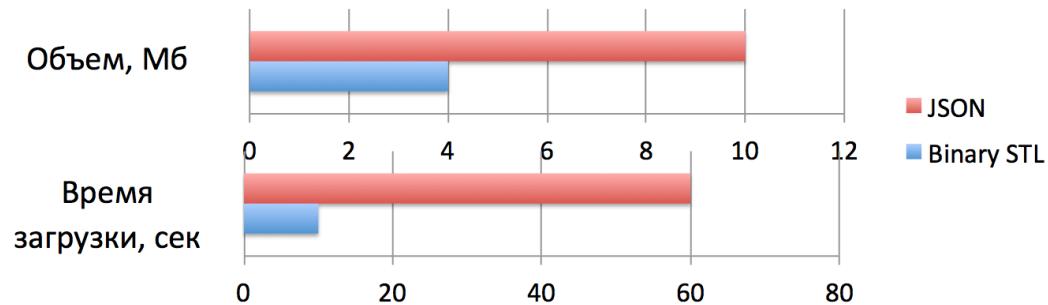


Рис. 2.4. Сравнение JSON и Binary STL.

2.2. Модуль редактирования изображений

2.2.1. Назначение

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](#)).

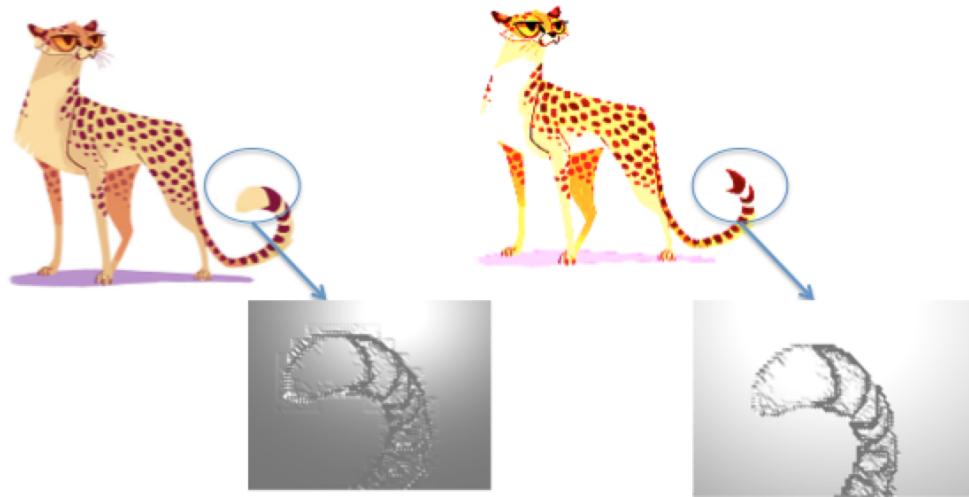


Рис. 2.5. Упразднение деформаций в моделях на основе растровых изображений.

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](#)).

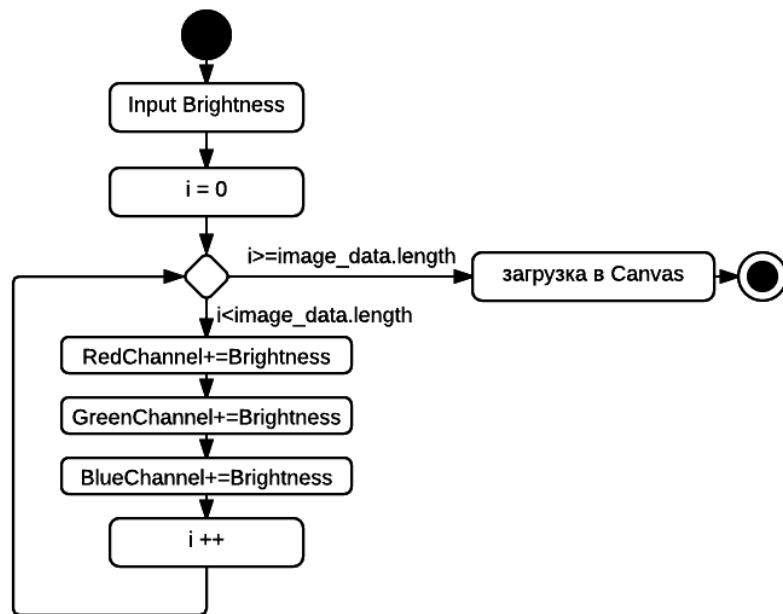


Рис. 2.6. Алгоритм изменения яркости.

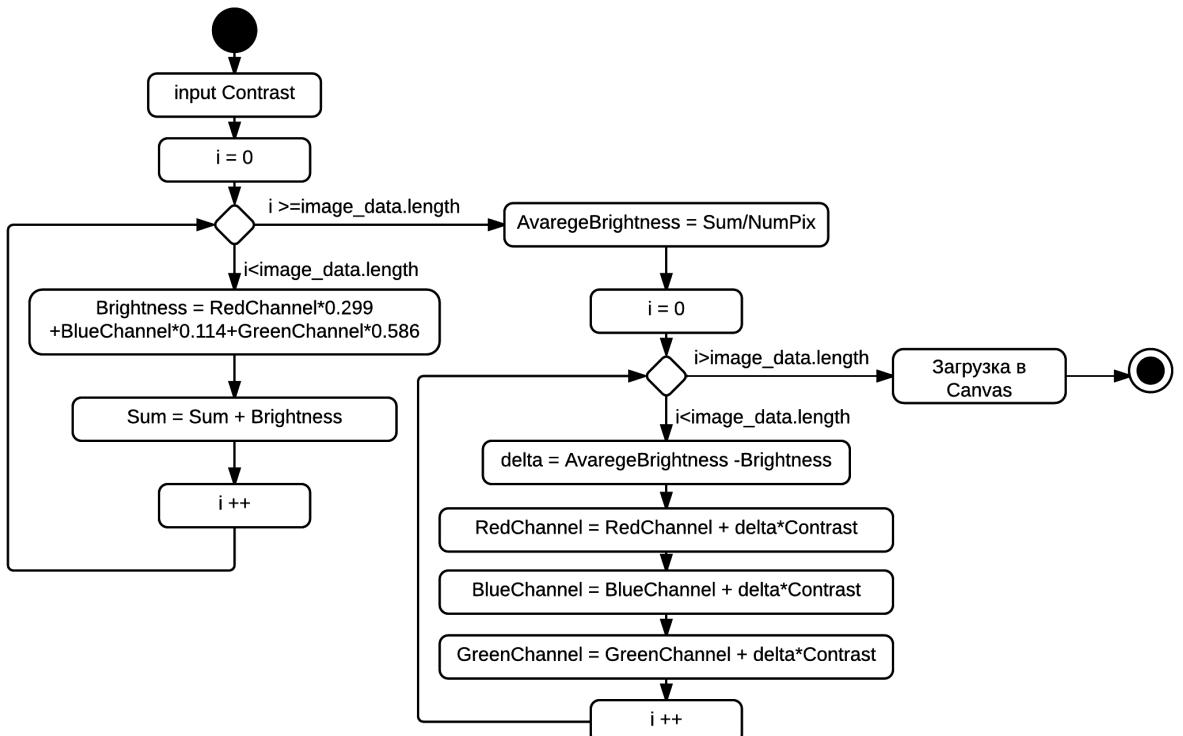


Рис. 2.7. Алгоритм изменения контраста.

2.3. Модуль генерации управляющих программ

2.3.1. Назначение модуля

Модуль генерации управляющих программ 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[6]. Большинство программного обеспечения, управляющего оборудованием, использует в качестве управляющих программ последовательности G-кодов. Стоит отметить, что сам G-код, определяемый стандартами ISO 6983-1:1982 и ГОСТ 20999-83, 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 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 (рис. 2.8).

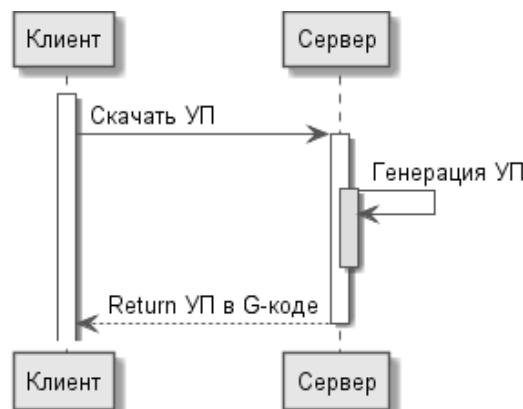


Рис. 2.8. Функционирование модуля генерации управляющих программ.

2.3.2. Реализация модуля

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

- карта высот, которая генерируется 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 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 из себя неизменяемый блок. Она включает в себя символ начала программы, имя программы, команды смены системы координат и единиц измерений, команды холостого хода, команду начала вращения шпинделя, команды установки скорости вращения шпинделя и скорости рабочей подачи. Блок представляет собой список строк и ставится всегда в начало УП.

Листинг 2.1. Преамбула

```
#пreamble
self.start_block = [
    '%', '001', 'G90', 'G21', 'G00X0Y0',
    'G00Z' + str(0),
    'G00Z' + str(2),
    'G91', 'F300.0 S6000M03'
]
```

Блоки тела УП. Выделяется 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. Так выглядят блоки для четных и нечетных строк:

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. Вид концевика представлен ниже:

Листинг 2.3. Блоки тела управляющей программы

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

Во время 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 этого процесса 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 файл с расширением .txt ([прил. В](#)), который становится доступным для скачивания.

Глава 3

Демонстрация работы и перспективы развития

3.1. Демонстрация работы на базе FabLab

В лаборатории FabLab на станке Roland MDX-40A (рис. 3.1) была испытана работоспособность модуля 3D-моделирования рельефа. После загрузки изобра-

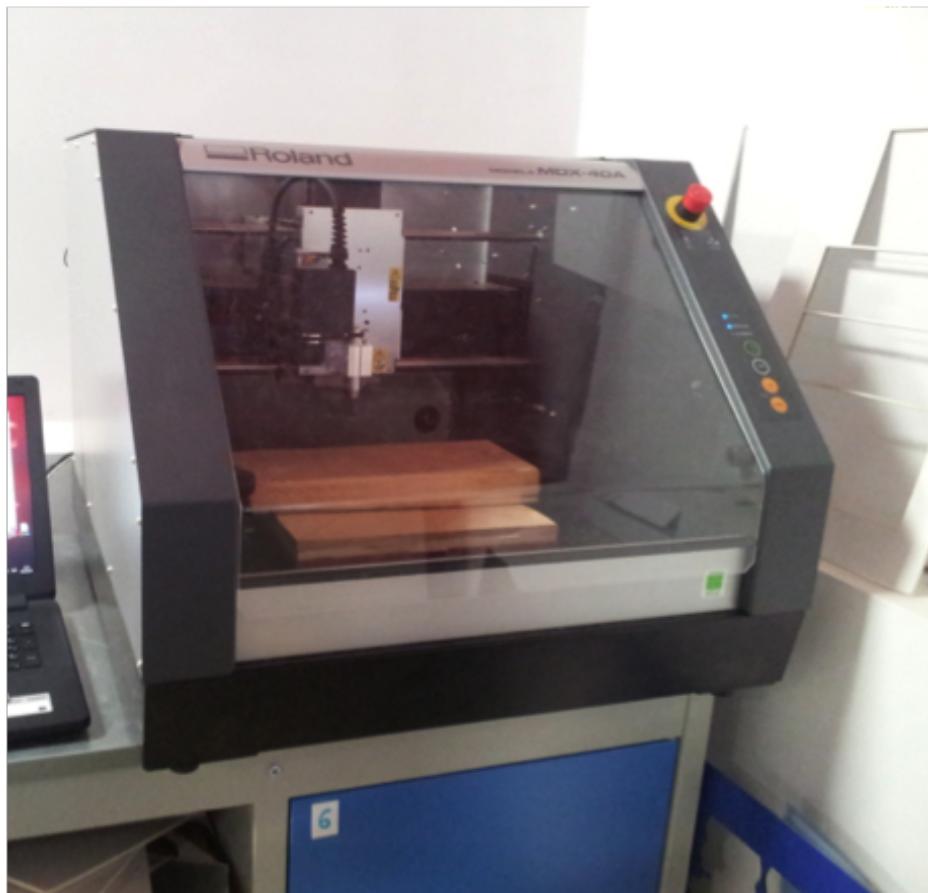


Рис. 3.1. Гравировально-фрезерный станок Roland MDX-40A

жения 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). SPR Player – это CAM-система, которая позволяет преобразовывать модели форматов IGES, STL, 3DM в управляющую программу. При этом управляющая программа состоит не из G-кодов, а из команд в нотации



Рис. 3.2. Сгенерированная модель

RML (Roland machine Language). Пример команд в нотации RML приведен на листинге:

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;
V 30;
Z 33458,270,50;
V 30;
Z 0,0,50;
^DF;
```

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



Рис. 3.3. Результат обработки

3.2. Демонстрация работы с помощью модуля визуализации системы PyCAM

3.3. Перспективы развития разработки

Заключение

Список литературы

1. The Flask Mega-Tutorial [Электронный ресурс]. URL: <http://habrahabr.ru/post/193242/> (дата обращения: 20.02.2015).
2. Гринберг М. Разработка веб-приложений с использованием Flask на языке Python. ДМК Пресс, 2014. 272 с.
3. Three.js docs [Электронный ресурс]. URL: <http://threejs.org/> (дата обращения: 20.02.2015).
4. Fulton S. HTML5 Canvas, 2nd Edition. O'Reilly Media, 2012. 750 p.
5. Sathaye N. Python Multimedia. Packt Publishing, 2010. 292 p.
6. Suh S.-H., Kang S.-K., Chung D.-H., Stroud I. Theory and Design of CNC Systems. Springer-Verlag, 2008. 455 p.

Приложение А

Дерево проекта



Приложение Б

Листинг модуля генерации G-кода

```

# -*- coding: utf-8 -*-
import copy

class Generator(object):

    def __init__(self, heightmap, h, w,
                 offset, norm, name):
        self.name = name
        self.norm = norm
        self.h = h
        self.w = w
        self.depth = self._depth(heightmap, h, w)
        self.offset = offset
        self.offset_counter = 0.0
        self.start_block = [
            '%', '001', 'G90', 'G21', 'G00X0Y0',
            'G00Z' + str(0),
            'G00Z' + str(2),
            'G91', 'F300.0S6000M03',
        ]
        #блок для нечетных строк
        self.body_block_unevenstr = [['G01X-', ''],

        #блок для четных строк
        self.body_block_evenstr = [['G01X', ''],

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

    def generate(self):
        """Generate G-codes list"""
        res = []
        for i in xrange(self.h):
            for j in xrange(self.w):
                if (round(float(self.depth[i][j]), 1) == 0.0):
                    self.offset_counter += self.offset
                    continue
                if i%2 == 0:
                    res += self._substitution(
                        self.body_block_evenstr,
                        i, j)
                else:
                    res += self._substitution(
                        self.body_block_unevenstr,
                        i, j)

```

```

        self.offset_counter = self.offset
    self.offset_counter = self.offset
    res += [ 'G01Y' + str(self.offset) ]
for item in self.end_block: res.append(item)
for item in self.start_block[::-1]: res.insert(0, item)
return res

#подставляем глубину в списки вида [ 'G01Z', '' ]
def _substitution(self, block, i, j):
    block = copy.deepcopy(block)
    for item in block:
        if (isinstance(item, list) and
            item[0].startswith('G01Z')):
            item[1] = str(float(self.depth[i][j]) + 2)
        else:
            item[1] = str(self.offset_counter)
    return block

def write(self, gcode_list):
    """Writes G-codes in file g_codes.txt"""
    with open(self.name, 'w') as f:
        for item in gcode_list:
            if len(item) == 1:
                f.write(item + '\n')
            else:
                f.write(''.join(item) + '\n')

def _depth(self, heightmap, h, w):
    with open('temp.txt', 'w') as f:
        f = heightmap
    depth = []
    for i in xrange(h):
        depth.append( [ str(self.norm - float(heightmap[i*w + j])) for j in xrange(w) ] )
    return depth

```

Приложение В

Управляющая программа

```
1 %
2 001
3 G90
4 G21
5 G00X0Y0
6 G00Z0
7 G00Z2
8 G91
9 F300.0S6000M03
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 %
```