**УДК 004.9**

# РЕАЛИЗАЦИЯ В ПРОГРАММНОМ КОМПЛЕКСЕ "ВИРТУАЛЬНЫЙ 3D-ПРИНТЕР" ВОЗМОЖНОСТИ ПРИМЕНЕНИЯ ВЕКТОРНЫХ ШРИФТОВ ДЛЯ АДДИТИВНОГО ПРОИЗВОДСТВА

М. А. Титов
(ФГУП "РФЯЦ-ВНИИЭФ", г. Саров Нижегородской области)

Представлена возможность применения векторных шрифтов в интегрирующей оболочке программного комплекса "Виртуальный 3D-принтер". Она обеспечивает нанесение текста, символов или логотипа предприятия на конструкции или детали, получаемые в процессе аддитивного производства методом послойного лазерного спекания металлической порошковой смеси. Рассмотрен SVG-формат компьютерных масштабируемых векторных шрифтов. Представлено его краткое описание на языке команд двумерной векторной графики Scalable Vector Graphics. Предложен подход получения координат точек, лежащих на границе обводки символа векторного шрифта. Описаны реализованные для этого синтаксический SVG-анализатор — парсер глифов символов шрифта — и библиотека растеризации квадратичных и кубических кривых Безье. Приведен пример использования разработанных программ для формирования трехмерной сцены из строки символов, выполненной векторным шрифтом. Представлены фотографии полученного в процессе аддитивного производства трехмерного текста.

*Ключевые слова:* SVG-формат векторных шрифтов, кривые Безье, парсер глифов символов, библиотека растеризации, аддитивное производство.

## Введение

В РФЯЦ-ВНИИЭФ с 2019 г. разрабатывается комплекс программного обеспечения "Виртуальный 3D-принтер" [1] для моделирования физических процессов, протекающих при селективном лазерном сплавлении, с целью прогнозирования структуры, свойств материалов, получения изделий с заданными свойствами, а также для проведения топологической оптимизации изделий. Разрабатываемый комплекс обеспечивает:

- создание и редактирование геометрий деталей;
- оптимизацию формы деталей на основе методов топологической оптимизации, в том числе с использованием решетчатых структур;
- подготовку деталей к печати;
- выбор оптимальных параметров работы оборудования;
- прогнозирование свойств деталей на базе многомасштабного математического моделирования на макро- и мезоуровнях.

Важной частью комплекса "Виртуальный 3D-принтер" является функциональный блок "Интегрирующая оболочка" (далее просто интегрирующая оболочка), объединяющий модули комплекса в единую программу с графическим интерфейсом и регламентирующий процесс подготовки изделий к печати посредством пользовательских диалогов. Статья посвящена развитию возможностей интегрирующей оболочки — реализации на языке программирования C++ и с помощью библиотек STL и QT возможности применения векторных шрифтов для обеспечения нанесения текста (например

марки металла изделия, поясняющих знаков и символов или логотипа предприятия) на выпекаемую конструкцию или деталь (рис. 1).

В своей ранней версии нтегрирующая оболочка уже обеспечивала поддержку собственного растрового шрифта (рис. 2). Этот шрифт ограничен буквенно-цифровым набором символов, геометрии частей его *кегельных площадок (глифов)* имеют равную ширину и эстетически несовершенны. Файл шрифта требует ручного вмешательства в случае необходимости его пополнения нестандартными символами. Перечисленные недостатки актуализировали задачу реализации возможности применения более совершенных *масштабируемых* векторных шрифтов.

Для обеспечения поддержки векторных шрифтов в первую очередь нужно сформировать формы фигур требуемых буквенно-цифровых символов — получить координаты точек, лежащих на контурах обводки глифов этих знаков. Один из подходов к решению поставленной задачи — получение нужных данных из компьютерных векторных шрифтов.

Компьютерный шрифт — это файл, содержащий в себе описание набора буквенных, цифровых, служебных и псевдографических символов, используемый для отображения этих символов программой или операционной системой. В векторных шрифтах символы представляют собой криволинейные контуры, описываемые математическими формулами. Каждый знак описан с помощью серии векторных команд, определяющих координаты опорных точек, которые соединены прямыми или кривыми и образуют контур знака без привязки к абсолютному размеру или разрешению. Такое описание обеспечивает увеличение масштаба изображения без потери качества. Обеспечение чтения векторного файла шрифта позволит получить серии команд, описывающих криволинейные контуры обводки символов, а реализация программ растеризации этих команд — требуемые координаты точек, лежащих на границах криволинейных контуров. В интегрирующей оболочке имеется возможность построения трехмерных моделей из плоских фигур, представленных координатами точек, лежащими на их контурах. Используем эту возможность для построения трехмерных моделей символов и печати сформированной сцены на принтере.


Рис. 1. Металлическая деталь с нанесенными символами


Рис. 2. Растровый шрифт интегрирующей оболочки

## SVG-формат векторных шрифтов

Существует несколько форматов шрифта, различающихся способом хранения и представления информации, — SVG, EMF, CDR, CGM, DXF, OPENVG, GXL и т. д. Для данной работы был выбран открытый стандартный формат SVG. Этот формат текстовый, а значит, наиболее понятный для изучения. Он реализует язык разметки Scalable Vector Graphics [2] описания двумерной векторной графики. Некоторые (free software) редакторы векторной графики, поддерживающие SVG-формат: Inkscapex [3], SK1 [4], Scribus [5].

Обводка глифа любого символа в этом формате представляется компактной SVG-строкой, описывающей путь от начальной точки до конечной через любые промежуточные координаты. Строка с данными задается атрибутом `d` тега `glyph` и содержит серию команд, закодированных набором букв

и чисел. Буква определяет тип команды, числа — ее параметры (чаще всего это координаты). Так, например, тег `glyph`, описывающий обводку глифа цифры "5" векторного шрифта DejaVuSansMono-BoldOblique.svg, выглядит следующим образом:

```
<glyph glyph-name="five" unicode="5"
d="M340 1493 h811 l-51 -260 h-572l-53 -277q33 13 69 19t77 6q197 0 313.5 -
116.5 t116.5 -313.5q0 -267 -178 -423.5t-486 -156.5q-95 0 -190.5 16t -190.5
48l51 266q70 -40 155.5 -61t180.5 -21q172 0 271.5 79t99.5 214 q0104 -71
164.5t-193 60.5q-78 0 -160.5 -20 t-161.5 -58z" />
```

Описание обводки глифа может состоять из отрезков прямых, дуг, а также квадратичных и кубических кривых Безье (рис. 3).

Кривые Безье [6] были представлены в 60-х годах XX века независимо друг от друга П. Безье из автомобилестроительной компании "Рено" и П. де Кастельжо из компании "Ситроен" и применялись для проектирования кузовов автомобилей. Кривые относятся к частному классу алгебраических кривых третьего и второго порядков и являются одним из важнейших инструментов современных систем автоматизированного проектирования и программ компьютерной графики.

Квадратичная кривая Безье задается тремя опорными точками $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$ и описывается формулой

$$\mathbf{B}(t) = (1-t)^2 \mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2, \quad t \in [0,1]. \tag{1}$$

Параметр $t$ определяет, на каком расстоянии от $\mathbf{P}_0$ до $\mathbf{P}_2$ находится точка $\mathbf{B}(t)$. Например, при $t = 0{,}25$ значение функции $\mathbf{B}(t)$ соответствует четверти расстояния между точками $\mathbf{P}_0$ и $\mathbf{P}_2$.

Кубическая кривая Безье задается четырьмя опорными точками $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P_3}$ и описывается формулой

$$\mathbf{B}(t) = (1-t)^3 \mathbf{P}_0 + 3t(1-t)^2 \mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3, \quad t \in [0,1]. \tag{2}$$

Опорные точки $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$ и $\mathbf{P}_3$, заданные в двух- или трехмерном пространстве, определяют форму кривой (см. рис. 3, *б*). Линия берет начало из точки $\mathbf{P}_0$, направляясь к $\mathbf{P}_1$, и заканчивается в точке $\mathbf{P}_3$, подходя к ней со стороны $\mathbf{P}_2$. То есть кривая не проходит через точки $\mathbf{P}_1$ и $\mathbf{P}_2$, они используются лишь для указания ее направления. Длина отрезка между $\mathbf{P}_0$ и $\mathbf{P}_1$ определяет, как скоро кривая повернет к $\mathbf{P}_3$. Точки $\mathbf{P}_0$ и $\mathbf{P}_3$ называются конечными, а $\mathbf{P}_1$ и $\mathbf{P}_2$ — управляющими.

В табл. 1 приведено описание основных команд и параметров атрибута `d` тега `glyph` на языке Scalable Vector Graphics, где `x_prev` и `y_prev` — координаты предыдущей конечной точки, а `xc_prev` и `yc_prev` — координаты предыдущей управляющей точки.

Все команды существуют в двух вариантах: команда, начинающаяся заглавной буквой, использует абсолютные координаты, а начинающаяся строчной буквой — относительные (например, команда `l 31 26` означает перемещение на 31 пиксель вверх и 26 пикселей влево от последней точки).

На рис. 4, *а* показан пример построения одноконтурного символа "C". Контур состоит как из отрезков, так и из кривых. В терминах языка разметки масштабируемой векторной графики для

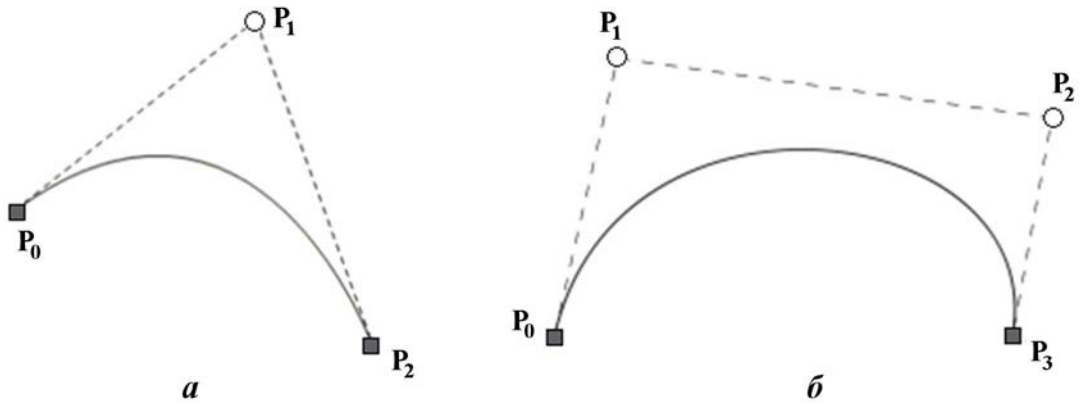


Рис. 3. Квадратичная (*а*) и кубическая (*б*) кривые Безье

*Таблица 1*

**Команды и параметры атрибута `d` тега `glyph`**

| Команда | Описание |
|---|---|
| `M x y` | Переместить перо в точку (`x,y`) |
| `m x y` | Переместить перо в точку (`x_prev`+`x`, `y_prev`+`y`) |
| `V y` | Провести вертикальную линию из точки (`x_prev`, `y_prev`) в (`x_prev, y`) |
| `v y` | Провести вертикальную линию из точки (`x_prev`, `y_prev`) в (`x_prev`, `y_prev`+`y`) |
| `H x` | Провести горизонтальную линию из (`x_prev`, `y_prev`) в (`x, y_prev`) |
| `h x` | Провести горизонтальную линию из (`x_prev`, `y_prev`) в (`x_prev`+`x`, `y_prev`) |
| `l x y` | Провести линию из (`x_prev`, `y_prev`) в (`x_prev`+`x`, `y_prev`+`y`) |
| `Q xc yc x y` | Построить квадратичную кривую Безье: начальная точка (`x_prev`, `y_prev`), конечная точка (`x,y`), управляющая точка (`xc, yc`) |
| `q xc yc x y` | Построить квадратичную кривую Безье: начальная точка (`x_prev`, `y_prev`), конечная точка (`x_prev`+`x`, `y_prev`+`y`), управляющая точка (`x_prev`+`xc`, `y_prev`+`yc`) |
| `T x y` | Построить симметричную кривую Безье: начальная точка (`x_prev`, `y_prev`), конечная точка (`x,y`), управляющая точка (`2*x_prev-xc_prev`, `2*y_prev-yc_prev`) |
| `t x y` | Построить симметричную кривую Безье: начальная точка (`x_prev`, `y_prev`), конечная точка (`x_prev`+`x`, `y_prev`+`y`), управляющая точка (`2*x_prev-xc_prev`, `2*y_prev-yc_prev`) |
| `Z` | Замкнуть контур |

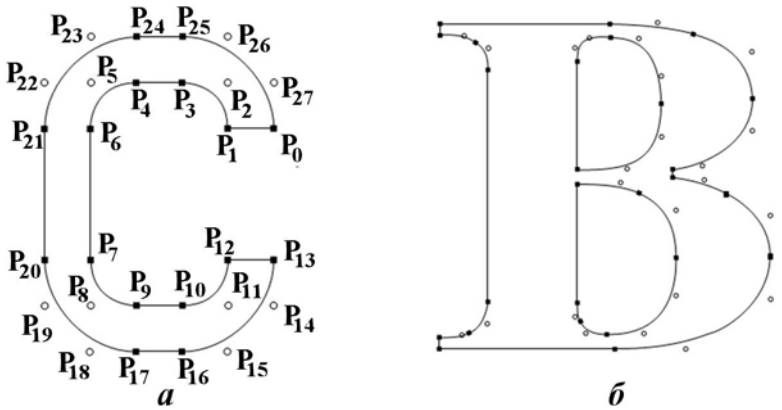


Рис. 4. Пример построения одноконтурного (*а*) и трехконтурного (*б*) векторных символов

построения этого символа использовались точки, лежащие на кривых контура (*on-curve*): $\mathbf{P}_1$, $\mathbf{P}_3$, $\mathbf{P}_4$, $\mathbf{P}_6$, $\mathbf{P}_7$, $\mathbf{P}_9$, $\mathbf{P}_{10}$, $\mathbf{P}_{12}$, $\mathbf{P}_{13}$, $\mathbf{P}_{16}$, $\mathbf{P}_{17}$, $\mathbf{P}_{20}$, $\mathbf{P}_{21}$, $\mathbf{P}_{24}$, $\mathbf{P}_{25}$, и управляющие точки, не лежащие на кривых контура (*off-curve*): $\mathbf{P}_2$, $\mathbf{P}_{26}$, $\mathbf{P}_{27}$, $\mathbf{P}_{11}$, $\mathbf{P}_{14}$, $\mathbf{P}_{15}$, $\mathbf{P}_8$, $\mathbf{P}_{18}$, $\mathbf{P}_{19}$, $\mathbf{P}_5$, $\mathbf{P}_{22}$, $\mathbf{P}_{23}$. 25-я точка соединяется с 0-й; 26-я и 27-я точки являются управляющими (off-curve) и служат для построения кривой $\mathbf{P}_{25}$—$\mathbf{P}_0$.

На рис. 4, *б* показан пример построения трехконтурного символа "B". В соответствии с SVG-форматом обход точек внешнего контура происходит по часовой стрелке, а двух внутренних — против часовой стрелки.

## Парсер глифов символов и библиотека растеризации

Итак, в файлах SVG-шрифтов строки данных с атрибутом `d` содержат геометрию обводки глифов символов. Глифы имеют векторный формат описания данных, а принтер работает с растровыми данными — координатами позиционирования лазерного луча. Следовательно, задача получения обводки глифа сводится к формированию набора SVG-команд и вычислению из набора отрезков и кривых Безье векторного формата координат точек, лежащих на контуре обводки, т. е. к растеризации данных — процессу, обратному векторизации. Для этого были реализованы:

– синтаксический анализатор, или *парсер*, тега `glyph` для получения серии SVG-команд векторного формата, описывающих обводку глифа;
– библиотека подпрограмм растеризации SVG-команд для нахождения множества точек кривой по координатам ее опорных точек.

Схема взаимодействия парсера с библиотекой растеризации следующая (рис. 5): парсер получает в качестве аргумента строку данных тэга `glyph` и разбивает ее на серию SVG-команд, из которых выделяет аргументы и активирует соответствующие им подпрограммы библиотеки растеризации, при необходимости вычисляя абсолютное смещение координат (если команды начинаются со строчной буквы). Подпрограммы библиотеки вычисляют координаты точек обводки глифа и сохраняют их в динамической структуре хранения данных.

Например, если в качестве аргумента парсера используется строка данных глифа цифры "2" векторного шрифта DejaVuSansMono-BoldOblique.svg (`M354 260h617l-51 -260h-949l49 252l277 248q51 46 145 128q402 354 402 507q0 72 -56 112.5t-155 40.5q-82 0 -186 -29.5t-222 -87.5l51 267q109 40 216 61t208 21q193 0 314 -95t121 -245q0 -249 -454 -638l-40 -34z`), то результат его работы можно представить в виде табл. 2.

Реализация адаптеров, изображенных на схеме взаимодействия парсера с библиотекой растеризации (см. рис. 5), обусловлена форматом языка разметки масштабируемой векторной графики Scalable Vector Graphics: в его командах построения линий/кривых начальная точка не задается явно, а определяется из конечных координат предыдущей команды.

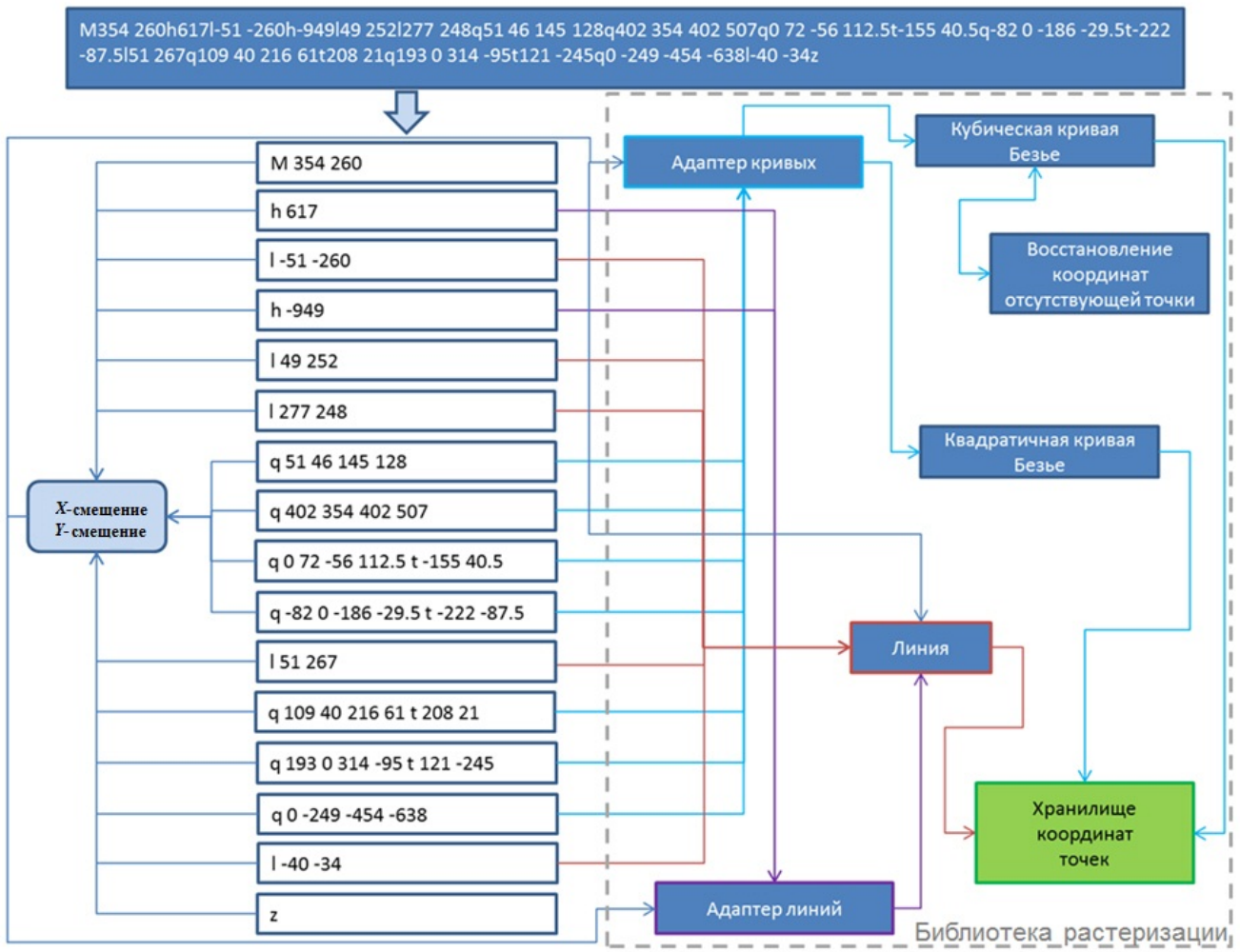


Рис. 5. Схема автоматического взаимодействия парсера с библиотекой растеризации

*Таблица 2*

**Результат парсинга данных тэга `glyph` цифры "2" векторного шрифта**

| SVG-команда тэга `glyph` | Вызываемая подпрограмма библиотеки растеризации и значения ее аргументов | Координаты абсолютного смещения |
|---|---|---|
| `M 354 260` | wraperM(354,260) | 354,260 |
| `h 617` | wraperL(P, 617, 0) | 971,260 |
| `l -51 -260` | wraperL(P, -51, -260) | 920,0 |
| `h -949` | wraperL(P, -949, 0) | −29,0 |
| `l 49 252` | wraperL(P, 49, 252) | 20,252 |
| `l 277 248` | wraperL(P, 277, 248) | 297,500 |
| `q 51 46 145 128` | wraperBezierPlusMultiT(P,51,46,145,128, NULL) | 225,1169 |
| `q 402 354 402 507` | wraperBezierPlusMultiT(P,402,354,402,507, NULL) | 442,628 |
| `q 0 72 -56 112.5t-155 40.5` | wraperBezierPlusMultiT(P, 0,72,-56,112.5, Points) | 844,1135 |
| `q -82 0 -186 -29.5t-222 -87.5` | wraperBezierPlusMultiT(P, 82,0,-186,-29.5, Points) | 633,1287 |
| `l 51 267` | wraperL(P, 51, 267) | 276,1436 |
| `q 109 40 216 61t208 21` | wraperBezierPlusMultiT(P, 109,40,216,61, Points) | 700,1518 |
| `q 193 0 314 -95t121 -245` | wraperBezierPlusMultiT(P, 193,0,314,-95, Points) | 1 135,1178 |
| `q 0 -249 -454 -638` | wraperBezierPlusMultiT(P, 0,-249,-454,-638, NULL) | 618,540 |
| `l -40 -34` | wraperL(P, -40, -34) | 641,506 |
| z | wraperM(354,260) | 354,260 |

## Поддержка векторных шрифтов в модуле задания 2D-контуров

Для того чтобы отобразить сформированный контур символа и хранить координаты его точек, парсер глифов и библиотека растеризации были интегрированы в разработанный автором модуль задания 2D-контуров интегрирующей оболочки. Этот модуль обеспечивает импорт и экспорт DXF-файлов САПР-систем [7], а также позволяет создавать и редактировать наборы замкнутых контуров, описывающие комплекты геометрий плоскостных срезов трехмерной детали в едином (уникальном) формате хранения данных со счетным модулем программного комплекса "Виртуальный 3D-принтер".

На рис. 6 представлен контур глифа цифры "2", полученный с использованием библиотеки растеризации и отображенный графическим интерфейсом модуля задания 2D-контуров. Из рисунка видно, что сформированный контур растеризован — он состоит только из отрезков.

Схема работы программ поддержки векторных шрифтов в модуле задания 2D-контуров, представленная на рис. 7, отображает два пути получения контуров обводки глифов:

- вывод текста, заданного стандартными буквенно-цифровыми символами (от "0" до "9", от "А" до "Я", от "а" до "я", от "A" до "Z", от "a" до "z", ",", ".", "+", "∗", "−"), данные тэгов глифов которых интегрированы в код программы;
- вывод любого нестандартного символа по его юникоду.

На рис. 8 показан интерфейс модуля задания 2D-контуров для ввода текста и получения координат контуров обводки глифов символов. Программа запрашивает

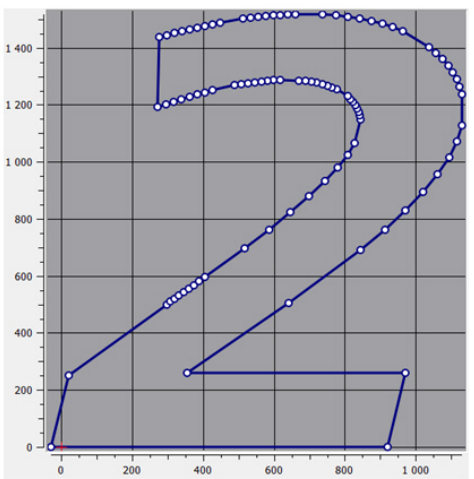


Рис. 6. Построенный контур цифры "2"

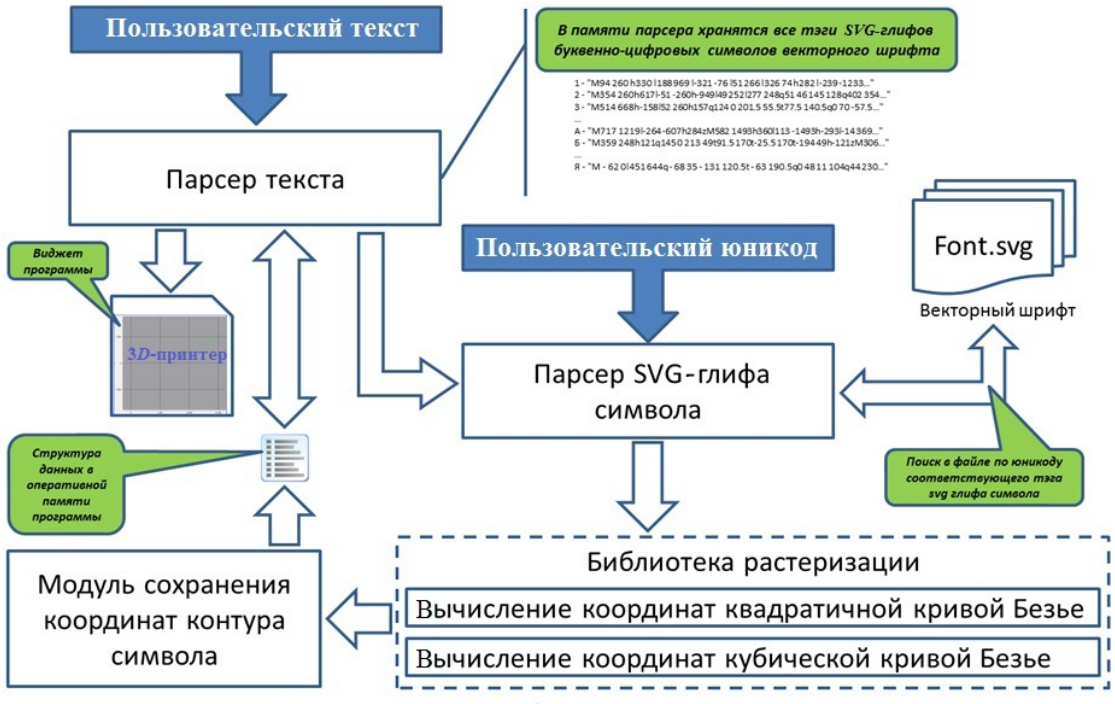


Рис. 7. Схема работы программ поддержки векторных шрифтов в модуле задания 2D-контуров

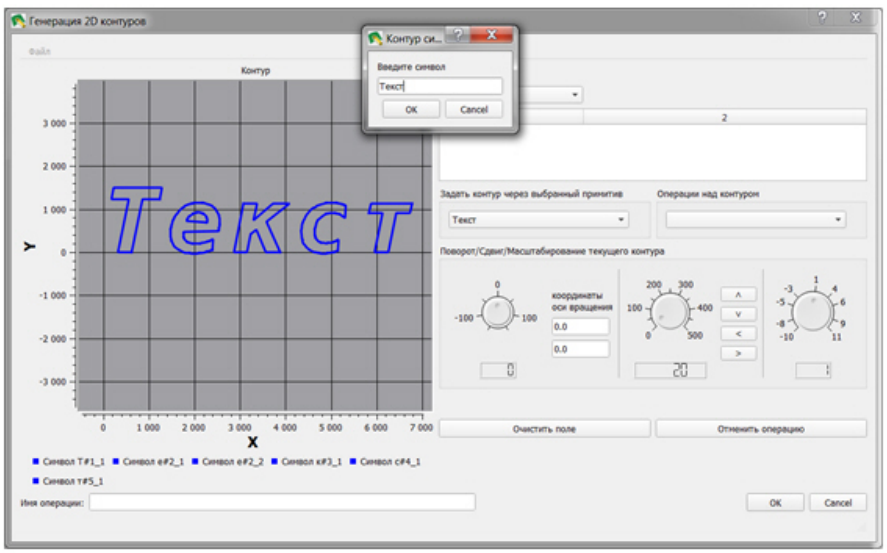


Рис. 8. Интерфейс ввода текста для получения контуров обводки глифов символов

физические размеры символов и автоматически вычисляет межсимвольный интервал введенного текста, т. е. смещение координат точек контуров глифов, следующих по отношению к первому контуру. С помощью инструментов этого модуля полученные контуры можно вращать, перемещать, масштабировать, редактировать координаты его точек (рис. 9) и т. д.

Для получения координат контура любого символа из произвольного SVG-файла векторного шрифта достаточно знать его юникод (Unicode). Так, указанием с помощью интерфейса модуля пути к файлу шрифта fontawesome-webfont.svg, а также юникода *xf1cc* его символа (рис. 10) был получен контур символа бесконечности (рис. 11).

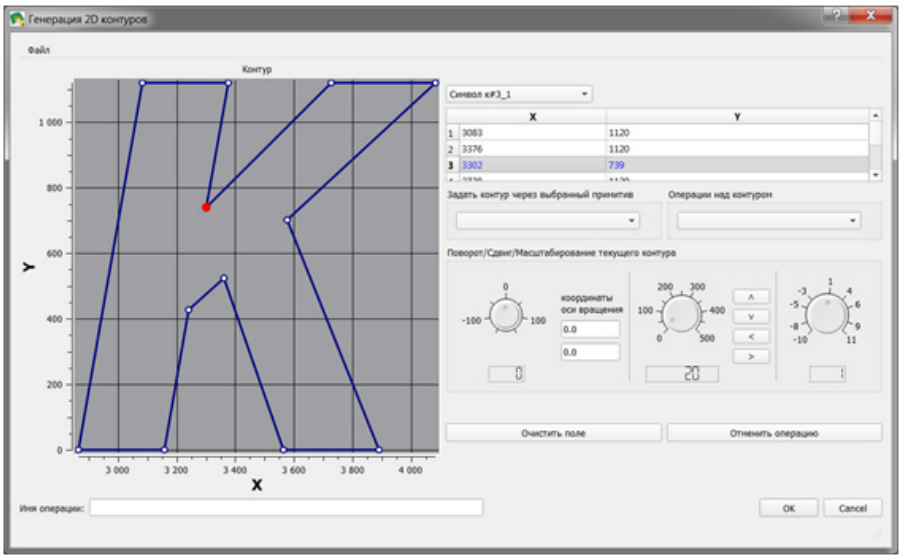


Рис. 9. Выделение точки контура символа инструментами модуля

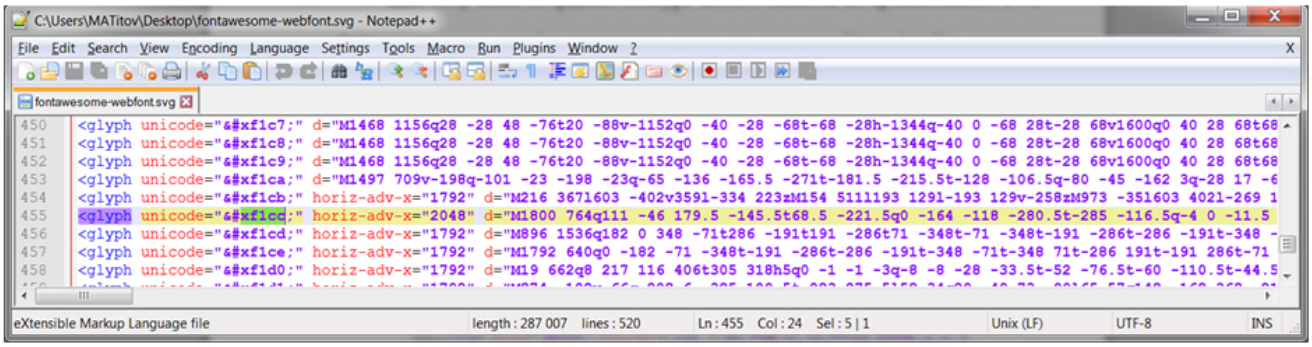


Рис. 10. Фрагмент SVG-файла шрифта fontawesome-webfont.svg с выделенной строкой с юникодом *xf1cc*

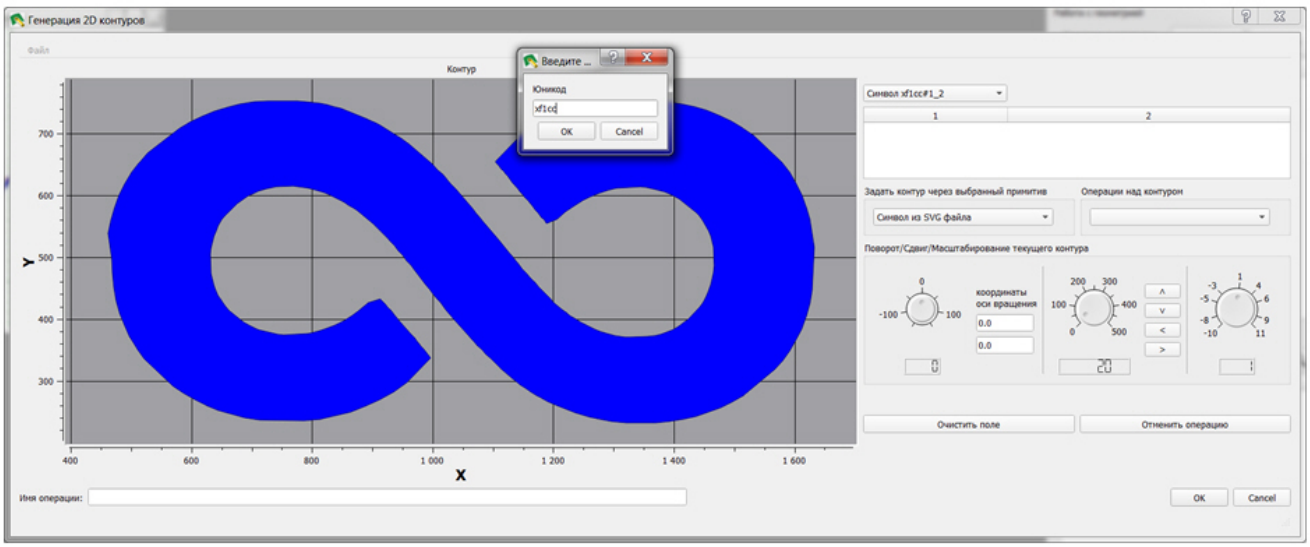


Рис. 11. Сформированный по юникоду *xf1cc* глиф символа бесконечности

### Печать текста векторным шрифтом

Проиллюстрируем все вышесказанное практическим примером работы пользователя в интегрирующей оболочке. С помощью инструментов модуля задания 2D-контуров сформируем контуры обводки глифов символов, составляющих фразу "*virtuAl 3d printer*" (рис. 12). Затем способом экструзии (путем "выдавливания" двумерного компонента как поперечного сечения объекта вдоль оси $OZ$), реализованным в модуле создания твердотельной геометрии из 2D-контуров, преобразуем двумерные контуры символов в трехмерные объекты (рис. 13). Далее посредством диалога *Генерация управляющего (G) кода* сгенерируем управляющий файл.

В программном комплексе "Виртуальный 3D-принтер" на данный момент реализована возможность генерации файла управляющего кода для принтеров производства ОАО НПО "ЦНИИТМАШ" линейки "MeltMaster" [8]. Управляющие файлы для аддитивной машины этой линейки имеют текстовый формат и состоят из команд собственного языка управления: выбор режимов работы лазеров, перемещение луча лазера и платформы, насыпание слоя металлической порошковой смеси.

Используя сгенерированный управляющий файл, напечатаем на принтере "MeltMaster 250" сформированную трехмерную сцену. Результат печати представлен фотографиями на рис. 14, 15.

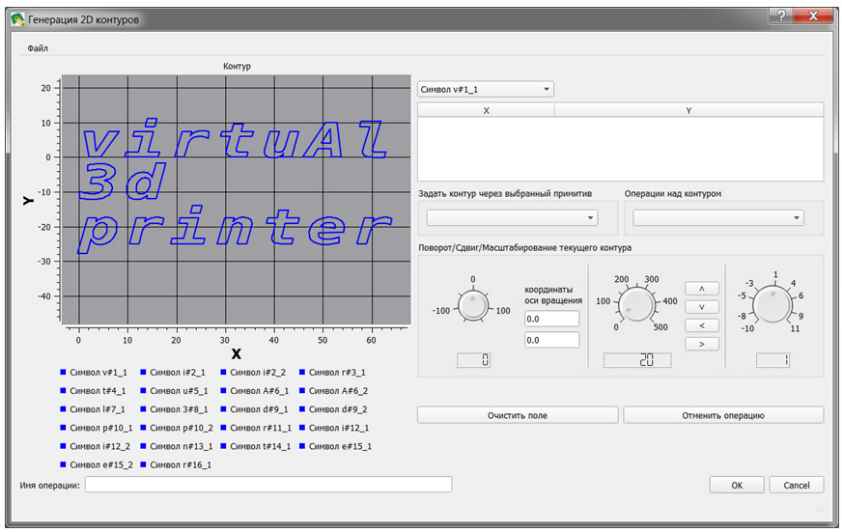


Рис. 12. Сформированные контуры обводки глифов символов

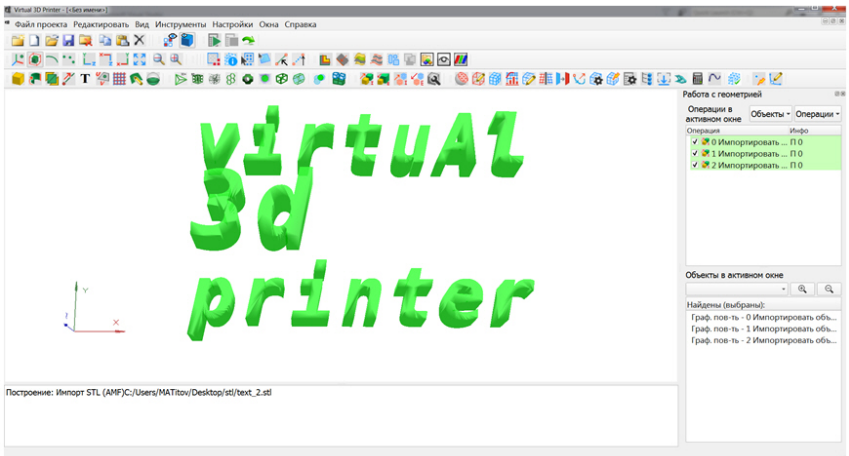


Рис. 13. Трехмерная сцена, подготовленная к печати способом экструзии

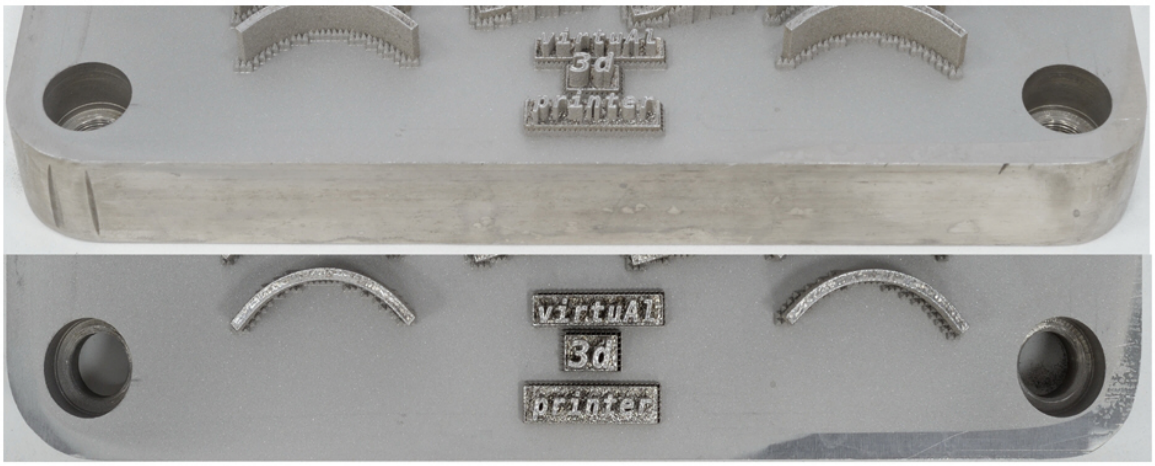


Рис. 14. Фрагменты фотографий платформы размером 250 × 250 мм с напечатанной сценой (материал – сталь 03Х17Н14М2), виды сбоку и сверху


Рис. 15. Фотография выполненной векторным шрифтом надписи, полученной селективным лазерным спеканием металлической порошковой смеси

## Заключение

В результате проделанной работы в функциональном блоке "Интегрирующая оболочка" программного комплекса "Виртуальный 3D-принтер" реализована возможность применения масштабируемых векторных шрифтов для нанесения текста на конструкцию или деталь, получаемую в процессе послойного лазерного спекания металлической порошковой смеси. Предложенный подход не ограничивается стандартным буквенно-цифровым набором символов, он позволяет получать кон-

туры любого символа любого файла шрифта SVG-формата, обеспечивая масштабирование символов без потери качества.

Реализованная возможность способствовала приобретению нового отличительного свойства выпекаемыми изделиями. В дальнейших планах развития интегрирующей оболочки — обеспечение нанесения текста на криволинейные поверхности изделий.

## Список литературы


1. *Дьянов Д. Ю., Медведкина М. В., Быков А. Н., Попов В. В.* Методы топологической оптимизации в программном комплексе 3D Printer // Математическое моделирование. 2019. Т. 19, № 7. С. 75—90.
   *Dyanov D. Yu., Medvedkina M. V., Bykov A. N., Popov V. V.* Metody topologicheskoy optimizatsii v programmnom komplekse 3D Printer // Matematicheskoe modelirovanie. 2019. T. 19, № 7. S. 75—90.
2. *Фрэйн Б.* Использование SVG для достижения независимости от разрешения // HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств: Пер. с англ. 2 изд. С.-Пб.: Питер, 2017. С. 175—204.
   *Freyn B.* Ispolzovanie SVG dlya dostizheniya nezavisimosti ot razresheniya // HTML5 i CSS3. Razrabotka saytov dlya lyubykh brauzerov i ustroystv: Per. s angl. 2 izd. S.-Pb.: Piter, 2017. S. 175—204.
3. *Bah T.* Inkscape: Guide to a Vector Drawing Program. Prentice Hall, 2011.
4. Редактор sK1 для работы с векторной графикой. https://sk1project.net.
   Redaktor sK1 dlya raboty s vektornoy grafikoy. https://sk1project.net.
5. Программа Scribus для визуальной верстки документов. https://www.scribus.net.
   Programma Scribus dlya vizualnoy vyerstki dokumentov. https://www.scribus.net.
6. *Роджерс Д., Адамс Дж.* Математические основы машинной графики: Пер. с англ. М.: Мир, 2001.
   *Rodzhers D., Adams Dzh.* Matematicheskie osnovy mashinnoy grafiki: Per. s angl. M.: Mir, 2001.
7. *Малюх В. Н.* Введение в современные САПР: Курс лекций. М.: ДМК Пресс, 2010.
   *Malyukh V. N.* Vvedenie v sovremennye SAPR: Kurs lektsiy. M.: DMK Press, 2010.
8. Сайт ОАО НПО "ЦНИИТМАШ". https://www.cniitmash.ru.
   OAO NPO "TsNIITMASH". https://www.cniitmash.ru.


Статья поступила в редакцию 01.03.22.