Правительство Санкт-Петербурга

Комитет по образованию

Государственное бюджетное нетиповое образовательное учреждение

«Президентский физико-математический лицей №239»

ВСЕРОССИЙСКАЯ ОЛИМПИАДА ШКОЛЬНИКОВ

Технология (Труд). Профиль «Робототехника»

**ПРОЕКТ**

**«РОБОТ-ХУДОЖНИК С ИСКУССТВЕННЫМ ИНТЕЛЛЕКТОМ»**

Выполнил:

ученик 10 класса

ГБНОУ Президентский ФМЛ №239

Дроканов Олег

Руководитель:

Иванов Василий Леонидович,

педагог дополнительного образования

ГБНОУ «Президентский ФМЛ №239»

Санкт-Петербург

2024

**СОДЕРЖАНИЕ**

[**1** **ПОИСКОВО-ИССЛЕДОВАТЕЛЬСКИЙ ЭТАП** 3](#_Toc189561827)

[**1.1** **Актуальность проекта** 3](#_Toc189561828)

[**1.2** **Цель и задачи** 3](#_Toc189561829)

[**1.3** **Практическая значимость** 4](#_Toc189561830)

[**1.4** **Предпроектное исследование** 4](#_Toc189561831)

[**1.5** **Концепция** 8](#_Toc189561832)

[**1.6** **Техническое задание** 9](#_Toc189561833)

[**2** **КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКИЙ ЭТАП** 12](#_Toc189561834)

[**2.1** **Подбор электроники** 12](#_Toc189561835)

[**2.2** **Проектирование и изготовление устройства** 16](#_Toc189561836)

[**2.3** **Программное обеспечение** 19](#_Toc189561837)

[**2.4** **Этапы работы над проектом** 26](#_Toc189561838)

[**3** **ЗАКЛЮЧИТЕЛЬНЫЙ ЭТАП** 31](#_Toc189561839)

[**3.1** **Результат** 31](#_Toc189561840)

[**3.2** **Актуальность услуги** 31](#_Toc189561841)

[**3.3** **Способы применения проекта** 32](#_Toc189561842)

[**4** **СПИСОК ИСТОЧНИКОВ** 33](#_Toc189561843)

[**ПРИЛОЖЕНИЕ А. Код выделения траектории** 34](#_Toc189561844)

[**Приложение Б. Анкета для пользователей робота-художника** 37](#_Toc189561845)

# **ПОИСКОВО-ИССЛЕДОВАТЕЛЬСКИЙ ЭТАП**

# **Актуальность проекта**

Согласно Указу Президента РФ от 10 октября 2019 г. N 490  
"О развитии искусственного интеллекта в Российской Федерации" [1] Российская Федерация обладает существенным потенциалом для того, чтобы стать одним из международных лидеров в развитии и использовании технологий искусственного интеллекта. Внедрение технологий искусственного интеллекта актуально в России для сохранения технологического суверенитета.

На данный момент проводится активное внедрение генеративного ИИ в робототехнику. Например, исследование разработчиков из Google [2], в котором они дали возможность большой языковой модели генерировать код для робота-манипулятора.

Я предлагаю способ применения генеративного ИИ для робототехнического устройства.

В современном мире ассортимент открыток в магазинах ограничивается фантазией дизайнеров и количеством наборов. Я решил, что сделаю робота, который сможет поддержать полет мысли любого человека и создать ему уникальный сувенир по мановению пальца.

В результате применения технологий искусственного интеллекта появился концепт робота-художника.

# **Цель и задачи**

Целью моего проекта стало создать робота-художника с искусственным интеллектом. Для выполнения поставленной цели потребовалось выполнить следующие задачи:

1. провести предпроектное исследование,
2. на основе исследования сформулировать идею проекта,
3. спроектировать устройство и составить список материалов и компонентов,
4. изготовить робота,
5. разработать программное обеспечение с использованием ИИ,
6. протестировать прототип в общественном месте.

# **Практическая значимость**

Робот имеет практическую значимость во многих сферах, например, развлечение посетителей в торговых центрах, в парках аттракционов, на ярмарках. Также можно его использовать в туристических целях: создавать уникальные памятные открытки, связанные с определенным местом, городом или страной. Такого робота можно установить и в учебных центрах, музеях, на выставках и конференциях, тогда он будет наглядным примером для популяризации робототехники среди детей и взрослых.

# **Предпроектное исследование**

Перед тем, как сформулировать окончательный вариант идеи проекта, я изучил аналогичные устройства по двум направлениям: роботов-художников и различные цифровые киоски и автоматы, представленные на рынке.

Первыми я рассмотрел различных роботов-художников.

1. Роботы-художники, основанные на манипуляторах.
   1. Робот-художник FRIDA [3].

|  |  |
| --- | --- |
|  |  |
| Рисунок 1 - Робот FRIDA | Рисунок 2 - Работа робота FRIDA |

Робот представляет собой манипулятор с кистью на конце (рисунок 1). Он умеет воспроизводить картины по шаблону, анализировать эскиз и запрос пользователя. Совмещая полученные данные, робот генерирует изображение из отдельных мазков и воспроизводит его (рисунок 2).

* 1. Робот-художник TEKO [4].

Манипулятор компании TEKO (рисунок 3) анализирует полученное изображения с помощью искусственного интеллекта и воспроизводит его мазками кисти (рисунок 4).

|  |  |
| --- | --- |
|  |  |
| Рисунок 3 - Робот-художник TEKO | Рисунок 4 - Картины робота TEKO |

1. Вторая разновидность роботов-художников – это координатные устройства.
   1. Bantam Tools NextDraw - перьевой плоттер для письма и рисования [5].

Этот плоттер разработан для письма и рисования на различных поверхностях. Умеет писать заданный текст и рисовать заданное изображение.

|  |  |
| --- | --- |
|  |  |
| Рисунок 5 - Bantam Tools NextDraw | Рисунок 6 - Чертеж робота |

* 1. Робот-художник разработанный в МГТУ им. Н. Э. Баумана [6].

Это многофункциональный робот-художник, генерирующий свои изображения с помощью нейросети.

|  |  |
| --- | --- |
|  |  |
| Рисунок 7 - Робот-художник МГТУ им. Н. Э. Баумана | Рисунок 8 - Рисунок робота-художника |

На основании рассмотренных аналогов я решил, что в роботе будет воспроизведено координатное устройство, потому что:

1. манипулятор сложно изготовить в условиях школьной лаборатории,
2. координатное устройство легче изготовить и программировать, его формат больше подходит к идее цифрового киоска и основной задумке.

Затем я начал изучать разные кинематики 3D-принтеров для того, чтобы выбрать из них самую подходящую для нашей задачи. В основном я рассматривал те, в которых печатающая головка двигалаcь по двум осям X и Y, так как для моей цели не нужно перемещение печатающей головки по оси Z. Были рассмотрены кинематики H-bot, Core-XY и Ultimaker.

1. Кинематика H-bot.

Печатающая головка перемещается по осям за счет одного ремня и параллельной работы двух моторов (рисунки 9 и 10).

|  |  |
| --- | --- |
|  |  |
| Рисунок 9 - H-bot | Рисунок 10 - Пример кинематики H-bot |

1. Кинематика Core-XY

Она очень похожа на H-bot, но имеет более сложное устройство. Она использует два ремня со сложной системой роликов (рисунки 11 и 12). Она имеет большую точность и надежность, но сложна в реализации.

|  |  |
| --- | --- |
|  |  |
| Рисунок 11 - Core-XY | Рисунок 12 - Пример кинематики Core-  XY |

1. Кинематика Ultimaker

В ней печатающая головка перемещается по направляющим валам, от качества и прямоты которых напрямую зависит качество ее передвижения (рисунки 13 и 14). Она также имеет большую точность, но сложна в реализации и производстве.

|  |  |
| --- | --- |
|  | Picture background |
| Рисунок 13 - Ultimaker | Рисунок 14 - Пример кинематики Ultimaker |

Среди всех кинематик я выбрал H-bot, потому что она понятна, проста в реализации, имеет достаточную точность перемещений и отлично подходит для нашей задачи.

# **Концепция**

Идея такова: человек рисует в воздухе рукой упрощённый эскиз и говорит, что он нарисовал, робот распознает движения и слова, затем нейронная сеть генерирует изображение на основе полученного наброска и темы рисунка, а плоттер рисует его на листе бумаги. В результате пользователь получает уникальную открытку на память. Такой робот позволяет раскрыть человеку свои эмоции через изобразительное искусство, даже если он не умеет рисовать.

Проект является роботом в соответствии с ГОСТ 60-0-0-4 2023, потому что:

* 1. в проекте имеется программируемый механизм перемещения каретки с маркером, поэтому проект является программируемым исполнительным механизмом;
  2. программное обеспечение позволяет роботу выполнять автономно все действия, заложенные в функционал: распознавание жестов и голоса человека, взаимодействие с человеком при помощи интерфейса, генерация изображения, определение положения бумаги в рабочей области плоттера, выделение команд для плоттера, опускание маркера на нужный уровень, рисование сгенерированного изображения посредством механизмов плоттера, отслеживание вмешательства в процесс рисования со стороны человека;
  3. робот способен позиционировать маркер и манипулировать им.

# **Техническое задание**

На основе проведённого предпроектного исследования были сформированы требования к роботу.

1. Комплектность

Устройство помещено в чемодан.

В основном отсеке чемодана должны находиться:

* 1. Плоттер для рисования открыток
  2. Системный блок робота

В крышке чемодана должны находиться:

* 1. 2 камеры
  2. Дисплей

1. Массогабаритные параметры

Робот должен умещаться в чемодан, внутреннее пространство которого имеет размеры 649х501х133,5.

В рабочей области плоттера должен умещаться лист А5 (для открытки) с возможностью поворота или сдвига.

1. Режимы работы робота

У робота может быть только один ч режим работы, обеспечивающий его основной функционал - автономный.

1. Требования к функционалу робота

Робот должен быть способным выполнять следующие задачи:

* 1. распознавать жесты пользователя и рисовать на экране линию на основе траектории руки пользователя,
  2. взаимодействовать с пользователем при помощи интерфейса на экране,
  3. распознавать голос пользователя,
  4. генерировать 3 варианта изображения по наброску пользователя и его голосовой подсказке,
  5. на основе сгенерированного изображения рисовать стилизованную одноцветную открытку по контурам при помощи плоттера.

1. Требования к материалам и их обработке

Корпус системного блока должен быть выполнен из фанеры.

Каркас плоттера должен быть из конструкционного алюминиевого профиля.

Каретка может быть собрана как из фанерных деталей, так и из деталей, напечатанных на 3D-принтере.

Все фанерные детали должны быть вырезаны на лазерном ЧПУ-станке.

Экран и камера в крышке чемодана должны помещаться в детали из пеноплэкса, отделанной самоклеящимся карпетом.

1. Требования к электронике робота

Контроллером плоттера должна выступать плата Arduino с расширением для управления шаговыми моторами.

Также необходимы 2 шаговых мотора для перемещения каретки плоттера.

Для распознавания голоса необходим микрофон.

Для определения уровня маркера необходим оптический концевой переключатель.

Для опускания и поднимания маркера необходим сервомотор.

Для запуска генеративной нейросети вычислительный блок должен содержать следующие компоненты:

* 1. внешняя видеокарта с видеопамятью не менее 12 Гб,
  2. ноутбук с процессором с тактовой частотой не менее 2.4 ГГц, оперативной памятью не менее 16 гб и внутренним хранилищем объемом не менее 128 Гб,
  3. плата подключения внешней видеокарты.

Также к вычислительному блоку необходимо подключить 2 камеры разрешением 1920х1080 и дисплей диагональю 15’6 дюймов.

1. Требования к источникам энергии и их подключению

В основном робот должен работать от сети, но также должен иметь возможность питаться от литий-полимерной батареи 6S 16000 мАч.

1. Требования к ПО робота

Программное обеспечение проекта должно состоять из следующих модулей:

* 1. программа, реализующая интерфейс. В нём должна быть область рисования, кнопки для управления рисованием и окно для предпросмотра видео с камеры,
  2. нейронная сеть для распознавания жестов и код для её использования,
  3. программа для взаимодействия с генеративной нейросетью,
  4. библиотека для перевода растрового изображения в команды GCODE при помощи выделения контуров,
  5. программа для детекции листка бумаги в рабочей области,
  6. прошивка для управления плоттером,
  7. программа для взаимодействия с платой управления плоттером, должна включать в себя программное обеспечение безопасности пользователя (детекция руки в рабочей области и остановка работы в случае обнаружения).

1. Требования к отделке и сборочно-монтажным работам

Для соединения каркаса плоттера необходимо использовать т-образные гайки.

Для установки камер и дисплея требуется создать посадочное место в детали из пеноплекса при помощи резки горячей проволокой.

1. Требования к функциональным характеристикам робота

Результатом работы робота должна быть открытка на листе формата А5.

Одна открытка должна рисоваться не дольше 20 минут.

# **КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКИЙ ЭТАП**

# **Подбор электроники**

Компьютерные комплектующие были выбраны по следующим причинам.

Генеративная нейросеть для большей скорости обработки требовала мощный графический процессор с как минимум 8 гб видеопамяти. Было два варианта решить эту проблему: либо встроить в робота полноценный компьютер и к нему подсоединить видеокарту достаточной мощности, либо использовать высокопроизводительный модуль NVIDIA Jetson. Первый вариант достаточно простой в реализации, так как все используемое в проекте ПО рассчитано для запуска на обычном компьютере, однако необходимые комплектующие (материнская плата, блок питания и т.п.) занимают очень много места. А второй отличается компактностью, но имеет минус в виде своей специфичной архитектуры процессора и видеомодуля, из-за чего некоторые библиотеки могут на нем не работать так же, как и на обычном ПК.

После проведённых с Jetson Nano тестов выяснилось, что 4 гигабайт видеопамяти, которые он имеет, недостаточно для запуска Stable Diffusion. Другие же модели Jetson, на которых возможно было запустить нейросеть, имели очень большую цену, поэтому было решено вернуться к идее встроить компьютер с видеокартой в устройство (рисунки 23 и 24).

|  |  |  |
| --- | --- | --- |
| Компоненты и оборудование | Название, количество, технические характеристики | боснование выбора |
| Контроллеры | Arduino Mega,  входное напряжение 12 В | Прошивка для управления шаговыми моторами разработана для Arduino в том числе, также планируется подключение большего количества датчиков |
| Электродвигатели | Шаговые моторы Nema 17 42SHDC, 2 шт, крутящий момент 4.2 Н\*кг, номинальный ток 1.7А | Шаговые моторы являются традиционным решением для 3D-принтеров, так как в их конструкцию заложено перемещение на маленькие шаги, благодаря чему обеспечивается высокая точность передвижения, необходимая для координатных устройств. Также обеспечивается достаточный крутящий момент (эти моторы используются в 3D-принтерах для перемещения экструдера, который тяжелее, чем каретка в проекте) |
| Платы управления двигателями | Protoneer CNC Shield V3, максимальное количество подконтрольных моторов – 2;  драйвера DRV8825, диапазон рабочих напряжений 8.2 – 45В | Подходит для выбранного контроллера и шаговых моторов, настройки для него воспроизведены в прошивке |
| Сервоприводы | PDI-HV2006MG, диапазон рабочих напряжений 6 – 8.4В, усиление на валу 4.8 кг/см | Реализует достаточный |
| Датчики | Оптический концевой выключатель | Обеспечивает достаточную точность позиционирование, взаимодействие реализовано в прошивке. |
| Видеокамера | Logitech C920, 2 шт., разрешение 1920х1080, интерфейс подключения USB | Высокое разрешение увеличивает точность распознавания, есть возможность прямого подключения к компьютеру |
| Прочие электронные компоненты | Микрофон |  |
| Аккумулятор | Литий-полимерная батарея |  |

Для проекта была разработана структурная схема Э1.

15

Изображение выглядит как диаграмма, текст, План, Технический чертеж

Контент, сгенерированный ИИ, может содержать ошибки.

# **Проектирование и изготовление устройства**

Все этапы моделирования проводились в САПР Компас-3D. Он был выбран, так как является отечественным ПО и предоставляет возможность создания чертежей в соответствии с ГОСТ. Подбор размеров всех деталей и комплектующих был произведен методами 3D-моделирования.

Опираясь на результаты предпроектного исследования для плоттера, была использована модель 3D-принтера с кинематикой H-bot из открытого доступа [7]. Из этой 3D-модели была удалена ось Z.

Рама плоттера была спроектирована на основе алюминиевого конструкционного профиля 20х20 и 20х40 и скрепляющих его деталей. На двух его углах установлены металлические пластины, повышающие жесткость конструкции (рисунок 15). На других 2 углах закреплены детали из фанеры толщиной 6мм, на которых крепятся шаговые моторы Nema17-42SHDC (рисунок 16). Они смогут обеспечивать высокую точность передвижения печатающей головки, а также достаточный крутящий момент.

Для обеспечения подвижности портала были добавлены две фанерные деталей с роликами, толщиной 6мм.

|  |  |
| --- | --- |
|  |  |
| Рисунок 15 - Плоттер. Общий вид | Рисунок 16 - Плоттер. Вид  сверху |

Далее нужно было создать модель печатающей головки. Она состоит из двух 3мм-пластин из фанеры, скрепленных собой роликами и специальной деталью, и крепится на конструкционный профиль 20х20. На одной из пластин находятся две направляющие MGN7C с ещё одной пластиной. В ней есть вырез для подшипника, который соединен с качалкой сервомотора. За счет этого она сможет двигаться вверх и вниз под действием мотора. Общий вид и вид сверху спроектированной каретки представлены на рисунках 17 и 18. На этой же пластине был спроектирован держатель маркера, который будет напечатан на 3D-принтере.

|  |  |
| --- | --- |
|  |  |
| Рисунок 17 - Каретка. Общий вид | Рисунок 18 - Каретка. Вид сверху |

Все модели деталей робота были объединены в одну сборку (рисунки 19 и 20).

|  |  |
| --- | --- |
|  |  |
| Рисунок 19 - Сборка. Общий вид | Рисунок 20 - Сборка. Вид сверху |

После того, как полная сборка была готова, были заказаны конструкционные профили, крепеж, алюминиевые уголки, подшипники и прочие детали, а также были вырезаны из фанеры на лазерном ЧПУ-станке и напечатаны на 3D-принтере остальные детали.

|  |  |
| --- | --- |
|  |  |
| Рисунок 21 - Плоттер | Рисунок 22 - Компьютер |

В соответствии с техническим заданием для антуража и удобства перемещения робота-художника он был помещён в старый чемодан. На его верхней крышке был установлен монитор (диагональ 15.6'') с 2 укрепленными на нем камерами Logitech HD 1080p. Внутри чемодана был закреплён корпус компьютера с процессором Intel Core i5-8400, 16 GB оперативной памяти и NVIDIA GeForce RTX 3060 с 12 GB видеопамяти.

|  |  |
| --- | --- |
|  |  |
| Рисунок 23 – Компьютер в сборе | Рисунок 24 – Материнская плата включая видеокарту с установленным процессором |

Провода от сервомотора были проведены через шлейф внутри кабель-канала. Вместе с шаговыми моторами, они подключены к плате Arduino Mega с расширением Protoneer CNC Shield V3. В него была загружена прошивка Marlin для 3D-принтеров. Она проста в конфигурации и использовании, поэтому выбрана для проекта. Команды в формате G-code отправляются с компьютера на плату по последовательному интерфейсу.

# **Программное обеспечение**

Структура ПО проекта:

1. программа, реализующая интерфейс,
2. нейронная сеть для распознавания жестов и код для её использования,
3. программа для взаимодействия с генеративной нейросетью,
4. библиотека для перевода растрового изображения в команды GCODE при помощи выделения контуров,
5. программа для детекции листка бумаги в рабочей области,
6. прошивка для управления плоттером Marlin,
7. программа для взаимодействия с платой управления плоттером, включает в себя программное обеспечение безопасности пользователя (детекция руки в рабочей области и остановка работы в случае обнаружения).

Программное обеспечение проекта сохранено в облачном репозитории GitHub [8].

Изображение выглядит как Шрифт, Графика, дизайн, графический дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 25 - QR-код на репозиторий.

Весь код проекта был написан на языке Python, за исключением программы по переводу изображения в траекторию плоттера. Для части с искусственным интеллектом было необходимо простое взаимодействие, а для части с траекторией – высокая скорость обработки информации, поэтому языки программирования были выбраны именно так.

Выделение ключевых точек рук реализовано при помощи библиотеки mediapipe. В ней уже был реализован и оптимизирован данный алгоритм. Далее собственная нейросеть распознает жест, который показывает рука, на основе набора из 21 координат этих точек.

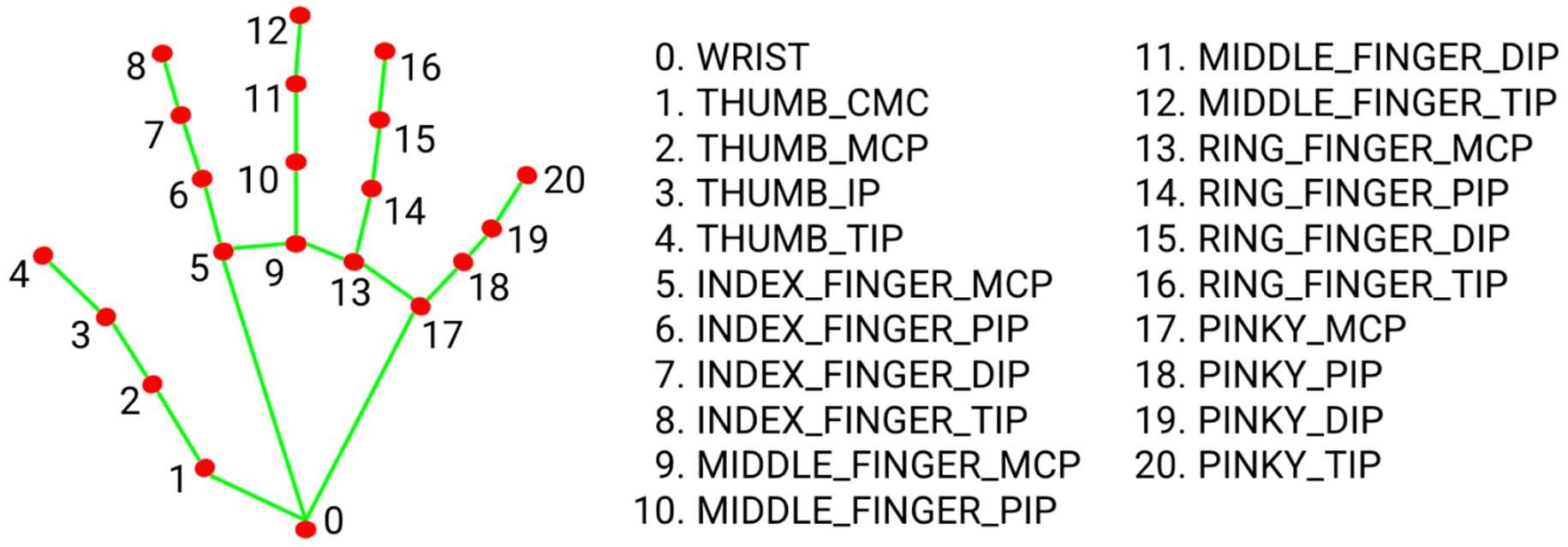


Рисунок 26 - Ключевые точки руки.

В создании классификатора (рисунок 26) использовалась библиотека Keras, один из самых популярных фреймворков для обучения нейронных сетей.

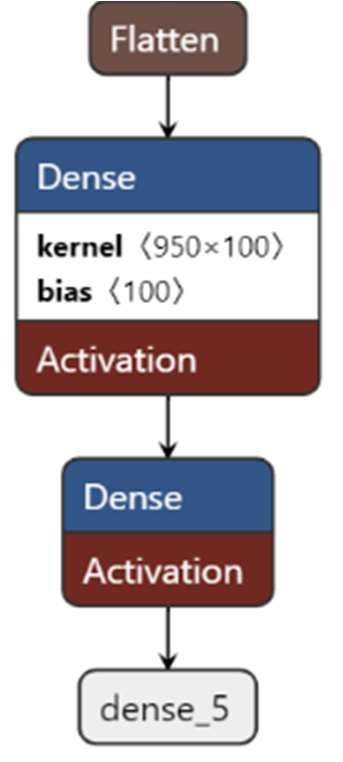
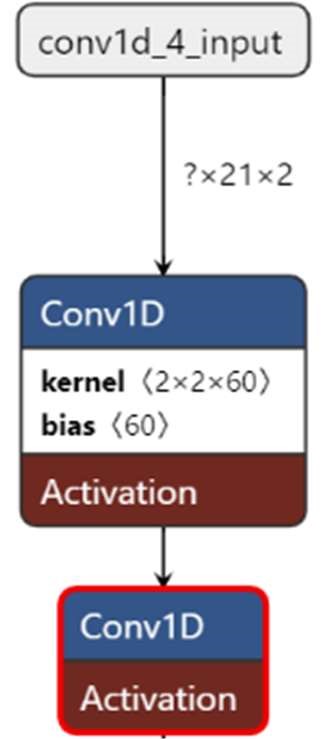


Рисунок 27 - Архитектура классификатора

Датасет для неё был собран следующим образом: сначала было отснято на камеру несколько видео с показанными жестами, затем из каждого кадра были выделены ключевые точки руки и записаны в файл.

Модель была обучена на 20 эпохах.

Для сокращения времени распознавания этой моделью она была оптимизирована при помощи TensorRT SDK.

Модель может распознавать 3 жеста: поднятый большой палец, раскрытую ладонь и поднятый указательный палец.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Рисунок 28 – Большой палец | Рисунок 29 – Раскрытая ладонь | Рисунок 30 – Указательный палец |

На последовательности этих жестов реализован конечный автомат основной программы. Его последовательная схема такова:

1. Ожидание показанного большого пальца.
2. Если жест – поднятый указательный палец, то курсор перемещается в координаты подушечки этого пальца.
3. Если большой и указательный палец соединены вместе (это определяется по отношению расстояния между ними к расстоянию от запястья до кончика указательного пальца), то имитируется «перетаскивание» мышки в координаты указательного пальца.
4. Если показан второй раз большой палец – рисование прекращается, ожидается голосовой ввод подсказки к генерации.
5. Когда голосовой ввод окончен, начинается генерация изображения.
6. Рисуем полученное изображение через плоттер и возвращаемся к пункту 1.

Голосовой ввод использует библиотеки google-trans и SpeechRecognition. Сначала считывается речь на русском языке, затем переводится на английский и отправляется в качестве подсказки в Stable Diffusion.

Управление генеративной нейросетью реализовано при помощи библиотеки Diffusers (рисунок 31). Используются веса моделей Stable Diffusion 1.5 и ControlNet-Scribble, находящиеся в открытом доступе.

# Импорт необходимых библиотек

import torch

import tomesd

from DeepCache import DeepCacheSDHelper

from diffusers import (

StableDiffusionControlNetPipeline,

UniPCMultistepScheduler, ControlNetModel

)

import matplotlib.pyplot as plt

from imageio.v3 import imwrite, imread

from skimage.util import invert

import numpy as np

# Загрузка предобученной модели ControlNet для работы с контурными изображениями

controlnet = ControlNetModel.from\_pretrained(

"lllyasviel/sd-controlnet-scribble",

torch\_dtype=torch.float32

)

# Создание пайплайна Stable Diffusion с ControlNet

pipe = StableDiffusionControlNetPipeline.from\_pretrained(

"runwayml/stable-diffusion-v1-5",

controlnet=controlnet,

safety\_checker=None,

use\_safetensors=True,

torch\_dtype=torch.float32

).to('cuda')

# Настройка планировщика для пайплайна

pipe.scheduler = UniPCMultistepScheduler.from\_config(pipe.scheduler.config)

# Инициализация DeepCache для оптимизации работы пайплайна

helper = DeepCacheSDHelper(pipe=pipe)

helper.set\_params(

cache\_interval=5, # Интервал кэширования

cache\_branch\_id=0, # ID ветки кэширования

)

helper.enable() # Включение DeepCache

# Применение патча для оптимизации памяти с использованием tomesd

tomesd.apply\_patch(pipe, ratio=0.5)

# Включение эффективного использования памяти с помощью xformers

pipe.enable\_xformers\_memory\_efficient\_attention()

# Преобразование формата памяти UNet и VAE для повышения производительности

pipe.unet.to(memory\_format=torch.channels\_last)

pipe.vae.to(memory\_format=torch.channels\_last)

# Определение текстового запроса и негативного запроса

prompt = "ship, high contrast grayscale drawing, only contours, wide lines, white background"

negative\_prompt = "many lines, bad anatomy, worst quality, bad quality"

# Загрузка и предобработка изображения с контурами

img = imread('images/scribble.png')

img = invert(img)[..., :3] # Инвертирование цветов изображения

img[img != 255] = 0 # Установка всех пикселей, кроме белых, в 0

img[img == 255] = 1 # Установка белых пикселей в 1

# Генерация изображения с использованием пайплайна

with torch.inference\_mode():

images = pipe(

prompt,

[img],

num\_inference\_steps=50,

height=512, width=512,

negative\_prompt=negative\_prompt,

output\_type='np'

).images[0] \* 255

# Сохранение сгенерированного изображения

imwrite('images/gen.png', images.astype('uint8'))

Рисунок 31 - Управление нейросетью

При помощи библиотеки CUDA все модели запускаются на графическом процессоре, что ускоряет их работу.

Также была создана библиотека для определения траектории движения каретки. Принцип его работы следующий. Сначала изображение преобразуется в оттенки серого. Затем из картинки выделяются контуры фильтром canny из библиотеки scikit-image. После этого к нему применяется бинарная диляция, то есть утолщение всех линий. Это нужно, чтобы алгоритм выделял и закрашивал больше линий. Далее действует алгоритм, написанный на языке С++:

1. Выбирается случайная точка, вокруг которой в радиусе d/2 (d – величина, характеризующая диаметр кончика маркера) нет ни одной закрашенной или не входящей в рисунок точки.
2. Вычисляются все точки, которые принадлежат пути из предыдущего пикселя в текущий (рисунок 32) (если это первая итерация цикла, то закрашиваются точки вокруг найденного пикселя), а затем для каждой из них все пиксели, принадлежащие окружности с центром в этой точке и диаметром d, закрашиваются.
3. Координаты точки добавляются в ответ.
4. Далее ищется точка, находящаяся на расстоянии d + 1 и удовлетворяющая условию из пункта 1.
5. Если такой не найдено, то выбирается случайная точка изображения, удовлетворяющая этому условию.
6. Если всё ещё не существует подходящей точки, то условие смягчается: достаточно, чтобы все пиксели в радиусе d/2 не выходили за рамки изначальной области, которую нужно закрасить.
7. Когда получены координаты точки, переходим к пункту 2.
8. Цикл повторяется, пока все пиксели не закрашены.

// Возвращает путь между двумя точками (start\_x, start\_y) и (end\_x, end\_y) в виде последовательности точек

PointVector get\_path(ld start\_x, ld start\_y, ld end\_x, ld end\_y) {

PointVector path\_points; // Вектор для хранения точек пути

ld delta\_x, delta\_y;

// Вычисляем вектор направления от начальной точки к конечной

delta\_x = end\_x - start\_x;

delta\_y = end\_y - start\_y;

// Вычисляем длину вектора направления

ld length = sqrt(delta\_x \* delta\_x + delta\_y \* delta\_y);

// Если начальная и конечная точки совпадают, возвращаем одну точку

if (length == 0) {

path\_points.push\_back(make\_pair(start\_x, start\_y));

return path\_points;

}

// Нормализуем вектор направления

delta\_x /= length;

delta\_y /= length;

// Инициализация текущих координат и счетчика шагов

ld current\_x = start\_x, current\_y = start\_y;

ld step = 0;

// Флаг для управления циклом

bool is\_path\_incomplete = true;

// Генерация точек пути

while (is\_path\_incomplete) {

// Добавляем текущую точку в результат

path\_points.push\_back(make\_pair(current\_x, current\_y));

// Обновляем текущие координаты с учетом шага

current\_x = start\_x + step \* delta\_x;

current\_y = start\_y + step \* delta\_y;

step++;

// Проверяем, достигли ли конечной точки с учетом погрешности

is\_path\_incomplete = !((abs(current\_x - end\_x) < 1e-6) &&

(abs(current\_y - end\_y) < 1e-6));

// Дополнительные проверки для предотвращения выхода за пределы конечной точки

if (end\_x - current\_x != 0) {

is\_path\_incomplete = is\_path\_incomplete &&

((current\_x - start\_x) / (end\_x - current\_x) >= 0);

}

if (end\_y - current\_y != 0) {

is\_path\_incomplete = is\_path\_incomplete &&

((current\_y - start\_y) / (end\_y - current\_y) >= 0);

}

}

// Добавляем конечную точку в результат

path\_points.push\_back(make\_pair(end\_x, end\_y));

return path\_points;

}

Рисунок 32 – Поиск пути между 2 точками

Так как точек обрабатывается очень много, даже C++, являющийся довольно быстрым языком программирования, справлялся недостаточно быстро. Поэтому некоторые циклы и функции были переписаны с использованием stdpar. Это функция стандарта C++17, которая позволяет запускать вычисления в параллельном режиме. А так как графический процессор в основном специализируется как раз на параллельной обработке, при помощи Nvidia HPC SDK, а в частности, компилятора nvc++, код стал запускаться на GPU. Благодаря этому производительность увеличивается в десятки раз по сравнению с чистым Python или C++.

Для реализации взаимодействия кода на двух языках программирования была использована библиотека ctypes. Эта библиотека – единственная в своём роде.

Интерфейс взаимодействия с пользователем написан с использованием библиотеки tkinter, потому что она представляет базовые объекты для его создания. В нём есть:

* + окно предпросмотра изображения с камеры,
  + кнопка очистки поля рисования,
  + индикатор того, рисует пользователь или нет,
  + строка прогресса генерации изображения,
  + поле для распознанного из голоса текста,
  + кнопка изменения толщины и всплывающее окно с вариантами толщины линии,
  + кнопка для завершения генерации,
  + поле для рисования.

На рисунке 33 изображен общий вид интерфейса программы.



Рисунок 33 - Интерфейс

# **Этапы работы над проектом**

Проект был начат в сентябре 2023 года. Можно разделить прогресс по проекту на 2 составляющие – конструкторская и программная.

|  |  |  |
| --- | --- | --- |
| Даты | Конструкция и электроника | Программа |
| Сентябрь 2023 | Поиск аналогов в Интернете | Тесты программы по распознаванию рук |
| Октябрь 2023 | Поиск и обработка основной 3D-модели | Обучение нейросети и встраивание её в программу |
| Ноябрь 2023 | Разработка модели каретки | Доработка рисования в воздухе |
| Декабрь 2023 | Разработка модели каретки | Настройка Nvidia Jetson, начало работы с выделением траектории |
| Январь 2024 | Настройка прошивки, сборка каретки | Настройка Nvidia Jetson, выделение траектории, создание интерфейса |
| Февраль 2024 | Подбор микроконтроллера, тесты, доработка 3D-моделей | Перевод программы по выделению траектории на Cython, тесты Stable Diffusion |
| Март 2024 | Сборка всего устройства, тесты перемещения | Пробы новых методов обработки изображения, объединение всех программ |
| Апрель -май 2024 | Доработка плоттера и тестирование его работы | Создание программы по переводу траектории в команды для плоттера, тесты рисования |
| Июнь 2024 | Улучшение рисования  Пробы разных вариантов маркера | Доработка программы по выделению траектории  Переписывание траектории на C++, добавление параллелизации, улучшение интерфейса, оптимизация Stable Diffusion |
| Сентябрь - октябрь 2024 |  | Добавление распознавания границ листка в рабочей области |
| Январь 2025 | Переработка корпус и замена электроники | Добавление автоматического опускания маркера, оптимизация программы через мультипроцессинг |

* 1. **Тестирование и модификации проекта**

Для апробации проекта, было решено установить робота в общественном месте. Каждому желающему предлагалось повзаимодействовать с роботом: нарисовать эскиз и задать тему рисунка. После этого робот генерировал изображение и рисовал его на глазах у зрителей.

Пользователи рисовали сложные эскизы, состоящие из нескольких траекторий. Чаще всего рисовали цветы, но встречались и другие идеи, например: метро, попугай на ветке, кот, слон и другие.

|  |  |
| --- | --- |
| Изображение нейросети, сгенерированные по запросам пользователей | Рисунок плоттера по изображению |
| Изображение выглядит как текст, рисунок, иллюстрация, зарисовка  Контент, сгенерированный ИИ, может содержать ошибки. |  |
|  |  |
|  |  |

Для удобства пользования графическим интерфейсом, на основном экране компьютера была выведена инструкция (рисунок 34), поясняющая значение каждого жеста.

Изображение выглядит как текст, визитная карточка, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 34 - Инструкция пользователя

Тем не менее помощь была нужна каждому пользователю, чтобы объяснять непонятные моменты и отвечать на вопросы.

При тестировании проекта в общественном месте (рисунок 35) не было получено ни одного отрицательного отзыва от пользователей. На вопросы анкетирования об уровне удовлетворенности использованием киоска и желания воспользоваться создания открыток еще раз 100% посетителей ответили утвердительно. Анкета для пользователей помещена в приложении.

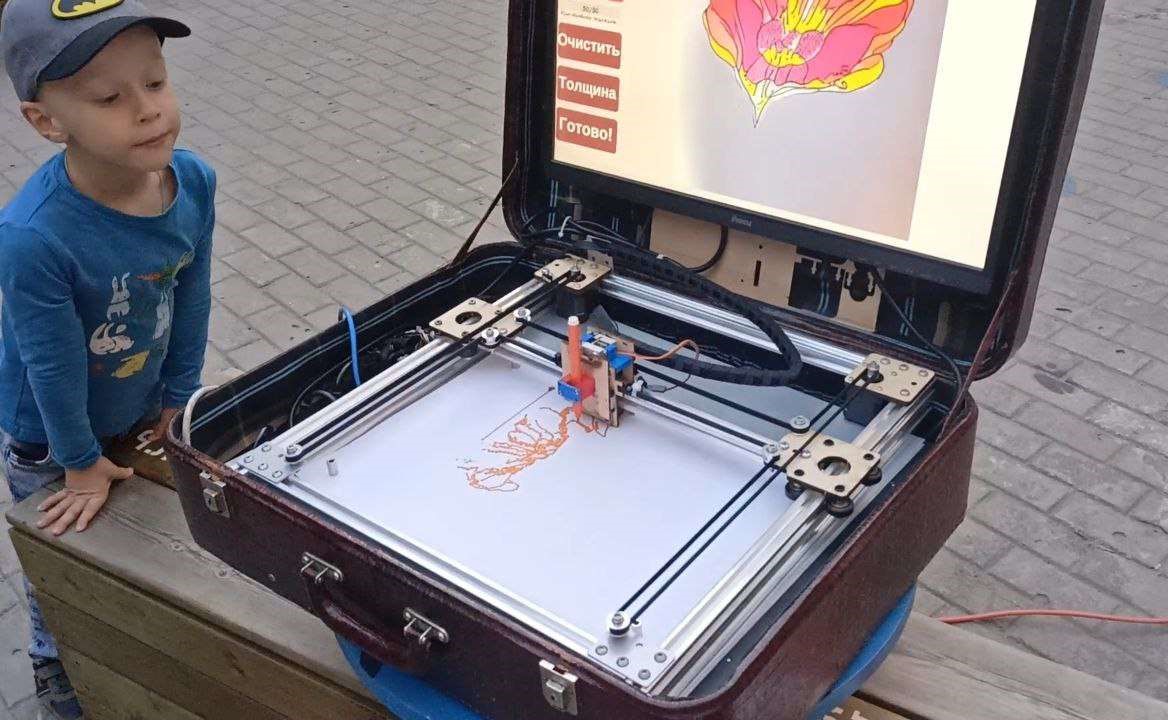


Рисунок 35 - Популяризация робототехники среди детей и взрослых

По результатам тестирования проекта в общественном месте были сделаны выводы о необходимости следующих шагов и модификации проекта:

1. спроектировать и установить в робота систему замены маркеров,
2. разработать ПО для разложения изображения на цвета и последующей обработки полученных цветовых пятен.

# **ЗАКЛЮЧИТЕЛЬНЫЙ ЭТАП**

# **Результат**

В результате получился робот-художник с искусственным интеллектом, который способен:

1. распознавать жесты пользователя и рисовать на экране линию на основе траектории руки пользователя,
2. взаимодействовать с пользователем при помощи интерфейса на экране,
3. распознавать голос пользователя,
4. генерировать 3 варианта изображения по наброску пользователя и его голосовой подсказке,
5. на основе сгенерированного изображения рисовать стилизованную одноцветную открытку по контурам при помощи плоттера.

Рисунок 36 – Робот-художник

# **Актуальность услуги**

Апробация проекта в общественном месте показала, что робот-художник точно не оставит равнодушным туристов и посетителей различных заведений. Людей привлекает возможность создать персональный сувенир памятную открытку и поработать в паре с искусственным интеллектом. Бумажная открытка – классический презент родственникам и детям, который легко отправить по почте или положить к книгам и документам. При этом, процесс рисования изображения роботом дарит огромное количество ярких эмоций.

# **Способы применения проекта**

Несмотря на перспективы коммерческого применения данного проекта в сфере развлечений, я считаю, что необходимо перенести приоритеты на благотворительное применение робота-художника для популяризации образовательной робототехники. Вместе с этим планируется использовать проект для сбора средств, арт-терапии и социальной инклюзии.

Существуют следующие шаги для использования проекта в социально значимых акциях в будущем учебном году:

* открытки с благодарностями или создание открыток для благотворительных акций. Размещение киоска в торговых центрах, на ярмарках и в общественных местах, где посетители смогут нарисовать свой эскиз, который будет превращен в открытку и передан волонтерам или благотворительным организациям.
* установка киоска в медицинских учреждениях, реабилитационных центрах и хосписах. Пациенты могут выразить свои эмоции через создание открыток, что может стать частью их психологической реабилитации.
* размещение киоска в детских домах, школах-интернатах и домах престарелых. Создание открыток может быть использовано как средство для самовыражения и улучшения эмоционального состояния.
* организация мастер-классов и творческих занятий с использованием киоска в школах или учебных центрах. Детям и взрослым можно объяснить, как работает киоск и предложить создать свои открытки.
* проведение конкурсов на лучшие открытки среди лицеистов и учащихся Центра робототехники. Лучшие работы можно использовать для украшения общественных пространств, или могут быть подарены людям, находящимся в трудной жизненной ситуации.

# **СПИСОК ИСТОЧНИКОВ**

1. Указ Президента РФ от 10 октября 2019 г. N 490  
   "О развитии искусственного интеллекта в Российской Федерации" – URL: <https://base.garant.ru/72838946/#block_1000>
2. Исследование Google. – URL: <https://code-as-policies.github.io> (дата обращения 04.02.2024)
3. Робот FRIDA – URL: <https://pschaldenbrand.github.io/frida/> (дата обращения 04.02.2024)
4. Робот-художник компании TEKO – URL: <https://b2b.tsml.ru/solutions/dopolnitelnoe-obrazovanie/robot-khudozhnik/> (дата обращения 04.02.2024)
5. Перьевой плоттер Bantam Tools NextDraw – URL: <https://store.bantamtools.com/products/bantam-tools-nextdraw-8511> (дата обращения 04.02.2024)
6. Робот-художник МГТУ им. Н. Э. Баумана – URL: <https://iz.ru/1610467/ivanchernousov/sovremennoe-iiskusstvo-v-rf-sozdali-mnogofunktcionalnogorobota-khudozhnika> (дата обращения 04.02.2024)
7. Робот-шахматист на кинематике H-Bot – URL: <https://grabcad.com/library/self-playing-chess-chessboard-using-h-bot-hbotxy-cartesian-system-1/details?folder_id=11517579> (дата обращения 15.06.2024)
8. Репозиторий GitHub проекта – URL: <https://github.com/olegg366/Artist> (дата обращения 04.02.2024)

# **ПРИЛОЖЕНИЕ А. Код выделения траектории**

void generate\_trajectory(IntMatrix2D &binary\_image, int distance,

int start\_x, int start\_y, PointVector &result) {

// Векторы для хранения смещений и максимальных смещений

PointVector deltas2, max\_deltas, deltas;

// Генерация смещений на основе заданного расстояния

generate\_deltas(deltas2, max\_deltas, deltas, distance);

// Фильтр для работы с бинарным изображением

IntMatrix2D filter = binary\_image;

// Переменные для хранения текущих и новых координат

int x, y, new\_x, new\_y, prev\_x, prev\_y, iteration, new\_new\_x, new\_new\_y;

// Флаги для отслеживания изменений и необходимости заполнения

bool change, fill\_flag = true;

// Получение случайной начальной точки на бинарном изображении

pair<int, int> random\_point\_result = get\_random\_point(binary\_image, max\_deltas);

x = random\_point\_result.first;

y = random\_point\_result.second;

// Добавление начальной точки в результат с учетом смещения

result.push\_back(make\_pair(x + start\_x, y + start\_y));

// Добавление маркера для разделения частей траектории

result.push\_back(make\_pair(-1e9, -1e9));

// Инициализация предыдущих координат и счетчика итераций

prev\_x = 0;

prev\_y = 0;

iteration = 0;

// Вектор для хранения пути между точками

PointVector path;

// Размеры бинарного изображения

pair<int, int> shape = {binary\_image.size(), binary\_image[0].size()};

// Основной цикл, пока в бинарном изображении есть ненулевые элементы

int zero = 0;

while (has\_elements\_not\_equal(binary\_image, zero)) {

change = false;

fill\_flag = true;

// Перебор всех возможных смещений для поиска следующей точки

for (pair<int, int> delta : deltas2) {

new\_x = x + delta.first;

new\_y = y + delta.second;

// Проверка, находится ли новая точка в пределах изображения

if (is\_point\_in\_image(new\_x, new\_y, shape)) {

// Если точка активна и удовлетворяет условиям, обновляем текущие координаты

if (binary\_image[new\_x][new\_y] &&

check\_nearby\_points(new\_x, new\_y, deltas, binary\_image)) {

x = new\_x;

y = new\_y;

// Добавляем новую точку в результат

result.push\_back(make\_pair(x + start\_x, y + start\_y));

change = true;

break;

}

}

}

// Если не удалось найти следующую точку, выбираем случайную точку

if (!change) {

random\_point\_result = get\_random\_point(binary\_image, deltas);

x = random\_point\_result.first;

y = random\_point\_result.second;

// Если случайная точка не найдена, используем фильтр

if (x == -1) {

random\_point\_result = get\_random\_point\_with\_filter(

binary\_image, deltas, filter);

x = random\_point\_result.first;

y = random\_point\_result.second;

}

// Если точка активна, добавляем маркеры и новую точку в результат

if (binary\_image[x][y]) {

result.push\_back(make\_pair(1e9, 1e9));

result.push\_back(make\_pair(x + start\_x, y + start\_y));

result.push\_back(make\_pair(-1e9, -1e9));

result.push\_back(make\_pair(x + start\_x, y + start\_y));

}

fill\_flag = false;

}

// Если нужно заполнить путь между предыдущей и текущей точкой

if (fill\_flag && iteration) {

path = get\_path(prev\_x, prev\_y, x, y);

// Перебор всех точек пути и обнуление соседних точек

for (pair<int, int> point : path) {

new\_x = point.first;

new\_y = point.second;

for (pair<int, int> delta : deltas) {

new\_new\_x = new\_x + delta.first;

new\_new\_y = new\_y + delta.second;

// Проверка, находится ли точка в пределах изображения

if (is\_point\_in\_image(new\_new\_x, new\_new\_y, shape)) {

binary\_image[new\_new\_x][new\_new\_y] = 0;

}

}

}

} else {

// Обнуление соседних точек вокруг текущей точки

for (pair<int, int> delta : deltas) {

new\_x = x + delta.first;

new\_y = y + delta.second;

if (is\_point\_in\_image(new\_x, new\_y, shape)) {

binary\_image[new\_x][new\_y] = 0;

}

}

}

// Обновление предыдущих координат и счетчика итераций

prev\_x = x;

prev\_y = y;

iteration++;

}

}

# **ПРИЛОЖЕНИЕ Б. Анкета для пользователей робота-художника**

