Министерство просвещения Российской Федерации

Муниципальное автономное общеобразовательное учреждение

«Лицей №14 имени Заслуженного учителя Российской Федерации А.М. Кузьмина»

Индивидуальный проект

**ГЕНЕРАЦИЯ ПОЛЕЙ ДЛЯ СОРЕВНОВАНИЯ РОБОТОВ**

**«ТРАЕКТОРИЯ-ПАЗЛ» И ИХ МОДИФИКАЦИИ**

Автор: Бобылёва Елена,

ученица 9К класса

Научный руководитель:

Вязовов Сергей Михайлович,

зав. кафедрой информатики

Тамбов, 2019

Оглавление

[Введение 3](#_Toc7026609)

[Актуальность 3](#_Toc7026610)

[Цель 4](#_Toc7026611)

[Задачи 4](#_Toc7026612)

[Ход работы 5](#_Toc7026613)

[Описание соревнования «Траектория-пазл» 5](#_Toc7026614)

[Среда разработки приложения 7](#_Toc7026615)

[Архитектура приложения 7](#_Toc7026616)

[Процесс генерации поля 11](#_Toc7026617)

[Используемые модули 14](#_Toc7026618)

[История версий приложения 14](#_Toc7026619)

[Заключение 16](#_Toc7026620)

[Список используемых источников 17](#_Toc7026621)

[Приложения 18](#_Toc7026622)

[Приложение 1. Код программы 18](#_Toc7026623)

[Приложение 2. Варианты различных полей 27](#_Toc7026624)

# Введение

## Актуальность

Робототехника – одно из ведущих направлений развития научно-технического творчества в Тамбовской области. Ежегодно, начиная с 2012 года на базе МАОУ «Лицей №14 имени Заслуженного учителя Российской Федерации А.М. Кузьмина» проводятся фестивали и соревнования регионального и межрегионального уровней.

Для каждого соревнования Главный судья готовит жеребьевочные материалы. Одним из таких соревнований является Траектория-пазл и всевозможные его модификации. В этом соревновании поле составляется из специальных пластинок, размера 30х30 см каждая пластинка. На пластинке нанесена линия определённой формы. Таким образом, используя эти пластины, можно составить поле произвольного размера и характера линии.

В классическом виде соревнований этой категории поле представляет собой целостную линию, то есть состоит из одной компоненты связности в терминологии теории графов.

Для генерации поля для этого специфического задания не существует приложений, так как задача имеет конкретную узкую направленность. Поэтому на данный момент судьей используется графический редактор, в котором это поле создаётся вручную перетаскиванием пластин одной за другой в нужное место.

Генерация поля вручную достаточно трудоёмкая процедура, которая отнимает много времени: нужно в правильном порядке расположить элементы поля, используя для этого какой-либо редактор. Для одного соревнования необходимо сгенерировать не менее 10 различных полей. Проблема усугубляется тем, что Главный судья один, а соревнований, для которых надо подготовить подобный материал - много.

В разы сократить время генерации полей позволит наша программа. В ней генерация одного поля занимает меньше 1 секунды. Если учитывать выбор интересных, с точки зрения судьи, полей, то сгенерировать все 10 полей можно меньше, чем за 5 минут.

Таким образом, целью проекта является создание приложения для генерации случайных полей.

Это приложение можно будет использовать не только судьям соревнований, но и тренерам команд при подготовке их к состязаниям.

## Цель

Создание приложения для генерации полей соревнования роботов категории «Траектория-пазл» с помощью языка программирования Python.

## Задачи

1. Изучить материалы Всероссийской робототехнической олимпиады и её региональных этапов, связанных с темой проекта.
2. Изучить средства разработки приложений.
3. Проработать функционал и построить на его базе архитектуру приложения.
4. Реализовать приложение.
5. Протестировать приложение и исправить возникшие ошибки.
6. Доработать функционал приложения с учётом пожеланий и предложений.

# Ход работы

## Описание соревнования «Траектория-пазл»

Соревнование «Траектория-пазл» впервые появилось на Всероссийской олимпиаде роботов в 2015 году в том виде, в котором оно было использовано для проведения Региональных соревнований по робототехнике в Тамбовской области. С тех пор ежегодно на всероссийском уровне проводится данное соревнование в различных своих модификациях, а в Тамбовской области это соревнование в том или ином виде присутствует два раза в год на соревнованиях и олимпиадах по робототехнике.

Классический вариант этого соревнования предполагает наличия замкнутой линии с одной компонентой связности.

Робот с размерами, как правило 25х25х25см на старте располагается в одной заранее известной или неизвестной ячейке. Его задача пройти по маршруту и собрать груз. На региональном этапе поле и место расположение груза заранее известно.

Для решения этой задачи участники команды должны написать для работа эффективную программу, которая позволит обойти поле и собрать груз. В этом случае эффективной программой будет алгоритм Дейкстры для поиска кратчайшего пути из клетки старта к месту расположения первого груза, затем из этого места в место расположения второго груза и так далее.

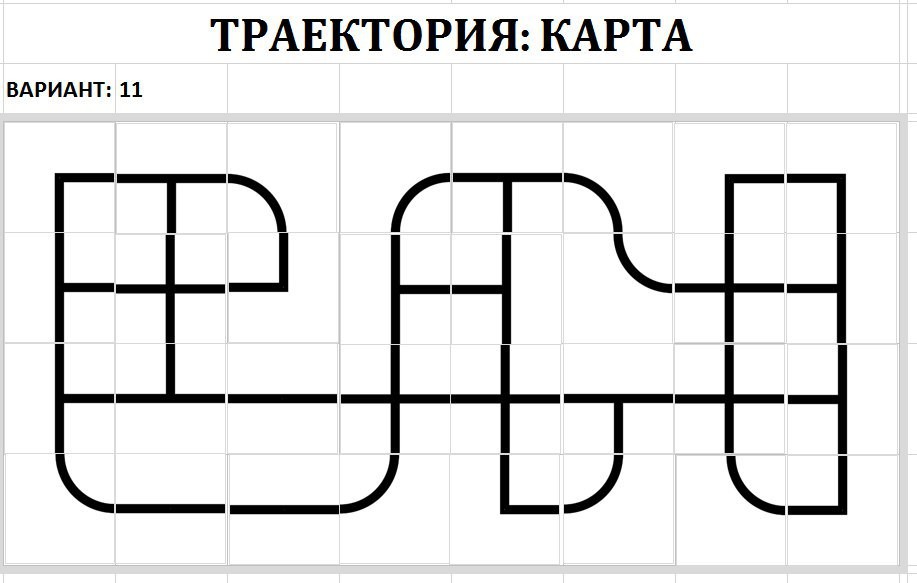


Рисунок 1. Вариант Траектории для соревнования

Как правило, для соревнований используется поле размера 8 клеток по горизонтали на 4 клетки по вертикали.

| **Элемент поля** | **Описание** |
| --- | --- |
| https://psv4.userapi.com/c848324/u9032150/docs/d16/b4129448a727/6.png?extra=tQTk-9hYfYm4giaBLtYSdbUKef7Z38INTZVu5dMU2t-FUdxhy5F-oDyxSPkRAjDKUbh_diu55JpKdHfcuXWAYZVHD_pqF9a2G1ddj_-xMTtNOtbZR0DHCn028cKj4LjN-54pPV-kcNFaBJgbRYww | Пустая ячейка.  Имеет одно состояние. |
| https://psv4.userapi.com/c848220/u9032150/docs/d17/5721f3688872/5.png?extra=ac9kqksoKEAFmKSw7P8dbRYRy1nGFqZ31UJIUZ_uYmKCseJIRzHQWQGUmT_upCNsjkSEO8Qa49PPWTodH9zlSldKBNIvosrV3SKdoIxMEjBSGIVmlYzO2NN844K2Yxk8rhSe7mihdjZlFS9OH1Cv | Т-образная линия.  Может находиться в 4 различных состояниях: ориентирована вверх, влево, вниз, вправо.  На рисунке приведена ориентация: вправо. |
| https://psv4.userapi.com/c848028/u9032150/docs/d3/5de9fd3422df/2.png?extra=xPVsVYJu6mg8LIek3xQGOypDe7MFLVDMmOni3tHe_FFR4paVskPwLkS1jQ6GfUbDrqqIFO90T3GFhmBM6Mz_26hNo4uohUyzjuIqMViU3hrStbgO-je7MZu-KSfAp5Acu6DUrKtGAIrLVPRsGyOy | Х-образная линия.  Может находиться в одном состоянии. |
| https://psv4.userapi.com/c848028/u9032150/docs/d14/5aa761984ea6/4.png?extra=OMM-Nh2gq1y5uuTIUb7yfvZMxbQKv9u0FK2vzUdRImsK39GR1SV1h6BRPT8wWcyylpSVjDfEH-qiC8AQ2I8Wp1ukN1yk8ESaRBy0omdBJXYgsF1dJdN9zUBFlArVM3zl_TqRqKTX8ijfFtEgviwc | Г-образная линия (поворот).  Может находиться в четырёх состояниях: вправо-вниз, вниз-влево, влево-вверх, вверх-вправо.  На рисунке приведено состояние: вправо-вниз. |
| https://psv4.userapi.com/c848028/u9032150/docs/d15/9d2433a67478/3.png?extra=x39gcPGkWvj7_JLR6wnXSQdit-PQfTPeT6fAlqi_TFuNVq_7mtEd6r68qSdeg0QddmlSI6QjwgWTKPOX6wgckrueuWKSzqeSasXrMQBF3UeZNNXvSxmZ0FR1Yk81dJcn5KfLWSxuHS9QjJcPaMas | Скруглённый поворот.  Может находиться в четырёх состояниях: вправо-вниз, вниз-влево, влево-вверх, вверх-вправо.  На рисунке приведено состояние: влево-вверх. |
| https://psv4.userapi.com/c848028/u9032150/docs/d12/be091de0f1b5/1.png?extra=kcsJLGAQKJT96GluwQlZBofkfPVUW4YOlQuWbLdamY3qi71-vnUcC2LrVShi18VpCw5ZQ493Rdb7Ndr1TAsqO2b5grJUDAivE6qxo_ACNeCGvcJZUgstLHu972xRn9UUNiYs1FA82txPO5v--eDv | Прямая линия.  Может находиться в двух состояниях: влево-вправо, вверх-вниз.  На рисунке приведено состояние: вверх-вниз. |

## Среда разработки приложения

В качестве языка для разработки приложения был выбран простой в использовании и освоении современный язык Python, версия 3. Для разработки приложений с визуальным интерфейсом мы будем использовать библиотеку Qt5, а также дополнительные модули, о которых мы расскажем ниже. Они нам понадобятся для разработки функционала приложения.

## Архитектура приложения

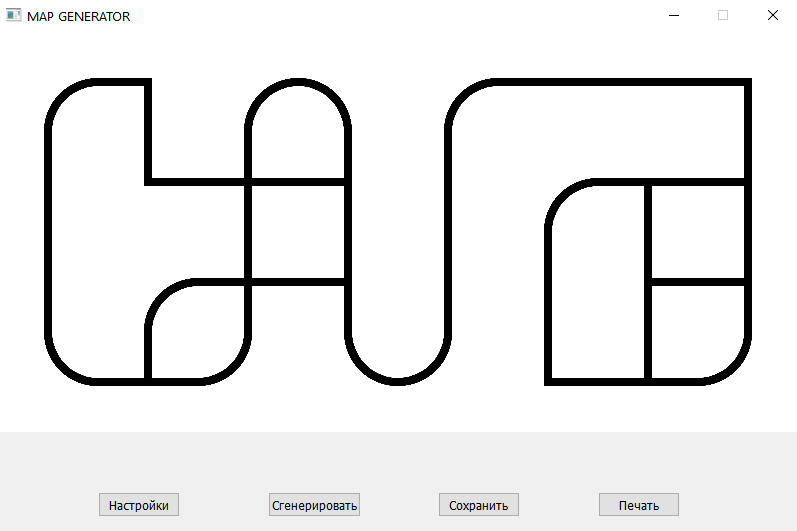


Рисунок 2. Окно программы Map Generator

Приложение представляет собой рабочую область с картой (1) и блок функциональных кнопок: «Сгенерировать» (2), «Сохранить» (3), «Печать» (4), «Настроить» (5).

Кнопка ***«Сгенерировать»*** позволяет получить новый вариант случайного поля с Траекторией.

Кнопка ***«Сохранить»*** позволяет сохранить текущий вариант поля в папку «SAVE» в корневой папке проекта (по умолчанию), либо в выбранную папку пользователем через меню настроек.

При этом программа сохраняет каждый файл под номером текущего сохраняемого варианта. Так, если имя файла с последним сохранённым вариантом поля будет «Line4.jpg», то имя следующего файла будет «Line5.jpg». Это сделано для того, чтобы в будущем можно было повторно использовать сгенерированные поля.

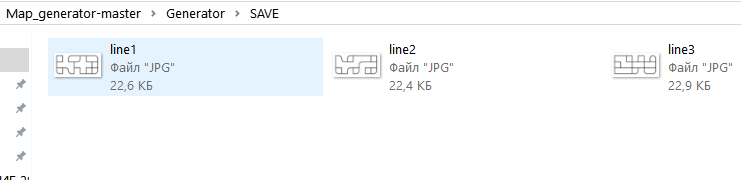


Рисунок 3. Содержимое папки SAVE

Кнопка ***«Печать»*** позволяет вывести сгенерированную случайную карту на печать. При нажатии на неё открывается диалоговое окно с выбором и настройками устройства печати. Это штатное окно операционной системы.

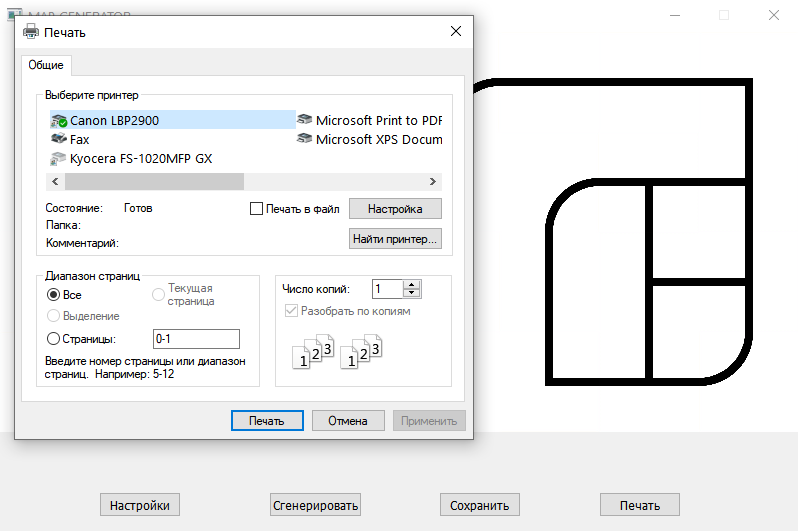


Рисунок 4. Диалоговое окно вывода на печать

При выводе на печать на листе бумаги по мимо самого поля отображается название соревнования и номер варианта поля. По умолчанию, если иного не задано в настройках, выводится название «Траектория-пазл».

При этом в программе настроена автоматическая ориентация листа в «Альбомный» вариант.

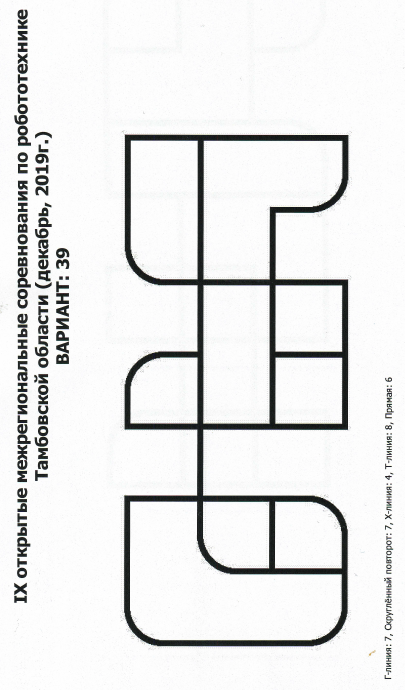


Рисунок 5. Результат вывода на печать

Дополнительно при выводе на печать отображается Легенда, в которой указано количество элементов поля каждого вида.

По кнопке ***«Настройки»*** вызывается окно настроек приложения (рисунок 6).

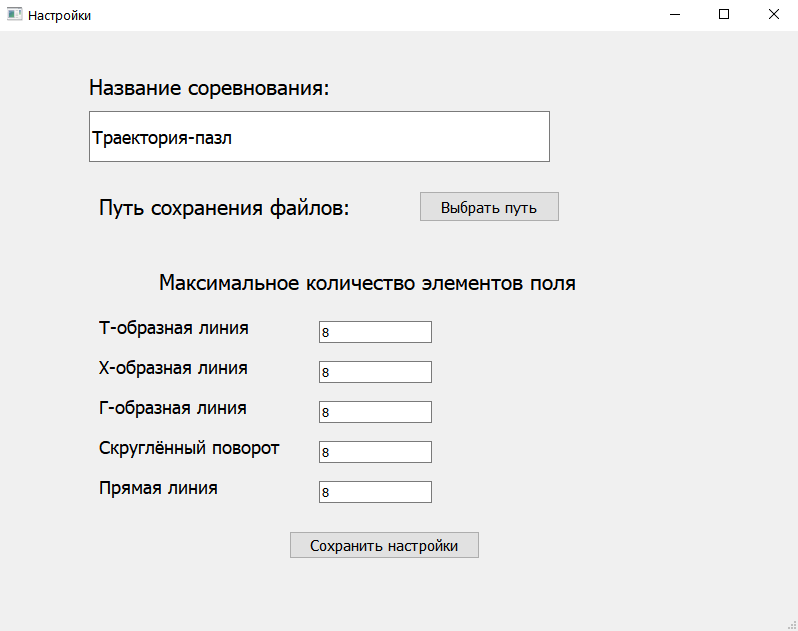


Рисунок 6. Окно настроек приложения

Здесь мы можем задать название соревнования, например «Траектория-пазл», выбрать путь для сохранения изображений сгенерированных полей, а также максимальное количество элементов для генерации поля.

## Процесс генерации поля

| **Действие** | **Описание** |
| --- | --- |
| 1 этап  Первыми заполняются угловые клетки | В качестве угловых клеток могут быть выбраны только пластинки с Г-образной линией или скруглённым поворотом. |
| 2 этап  Заполняются клетки первого и последнего столбца, первой и последней строки | Пластинки здесь выбираются так, чтобы линия была замкнутой по одному из контуров (левому или правому, верхнему или нижнему). |
| 3 этап  Заполняются все оставшиеся ряды и строки, кроме последнего | Пластинки подбираются аналогично предыдущему этапу, образуя замкнутую линию по одному из контуров. |
| 4 этап  Заполнение последней строки | По аналогичному принципу здесь выбираем все пластинки, за исключением последней. При этом нам необходимо учитывать состояние трёх соседних клеток так, чтобы линия траектории была замкнутой. |
| 5 этап  Заполнение последней клетки | Последняя клетка имеет в качестве соседей 4 клетки, поэтому для неё пластина с линией выбирается отдельно. |
| 6 этап  Проверка целостности поля | Как было уже сказано выше, поле должно содержать замкнутую линию с одной компонентой связности.  После генерации поля для проверки количества компонент связности мы запускаем dfs из клеки (0,0). Если по окончании работы dfs все клетки окажутся посещенными, то наша линия состоит из одной компоненты связности.  Для работы dfs при генерации поля создаётся двумерный список, который хранит структуру поля. |
| 7 этап  Учёт конфигурационных настроек количества элементов поля | После того, как карта сгенерирована, прежде, чем выводить её в рабочее поле, происходит проверка: не превышает ли количество элементов, заданное в параметрах настроек, с тем количеством, которое получилось при генерации.  Если в результате проверки обнаружено превышение заданного числа количества элементов поля, то алгоритм генрации запускается с самого начала.  Если за разумное время – не более 5 секунд – не было сгенерировано поле для заданного числа элементов, то выводится сообщение об этом. За это время алгоритм успевает перебрать примерно 15 тысяч различных вариантов полей.  Мы полагаем, что если за это время не найдено ни одного варианта, удовлетворяющего заданным ограничениям на число элементов поля, то необходимо сменить настройки, так как, вероятней всего, выставленные настройки не отвечают полям для реальных соревнований. |

Разбиение процедуры генерации на такое количество этапов обусловлено архитектурной сложностью приложения. С этой точки зрения, на каждом этапе определяется своя процедура выбора нужной пластины для поддержания свойства замкнутости линии. При этом на каждом этапе учитывается разное количество соседних клеток.

Такой вариант генерации поля никак не учитывает количество компонент связности, поэтому поля могут получиться как односвязными, так и многосвязными. Именно поэтому необходимо запускать алгоритм перегенерации, пока не получится поле с одной компонентой связности.

Стоит отметить, что вероятность выпадения поля с двумя компонентами связности не превышает 20%.

## Используемые модули

При реализации проекта были использованы следующие модуль:

* виджеты из QtGui и QtWidgets модуля PyQt5 для графического представления программы;
* возможности модуля random для генерации разных карт;
* модуль PIL для создания и сохранения скриншота экрана;
* модуль os для поиска сохранённой карты с наибольшим номером в папке SAVE;
* возможности виджета QPrinter для печати карты.

## История версий приложения

**Версия 1.0**

Версия 1.0 содержала следующие функциональные особенности:

1. Генерация поля размером 8х8. Генерация поля состояла из 5 этапов (см. Процесс генерации поля), то есть не содержала проверки на односвязность линии. Из-за этого иногда (с небольшой вероятностью) могли появится двусвязные поля.
2. Функция «Сохранить» позволяла сохранить файл в туже директорию, что и файлы программы. При этом оно затирало предыдущую версию. Это было неправильным техническим решением и было исправлено в следующей версии приложения.
3. Функция «Печать» позволяет вывести на печать сгенерированную версию. Данная функция не претерпела серьёзных изменений в следующих версиях приложения.

**Версия 1.1**

В версии 1.1 было доработано следующее:

1. В файлах проекта был наведён порядок. Файлы исходных картинок для генерации поля были помещены в папку «RES», а для сохранения сгенерированных полей была добавлена папка «SAVE». Само же приложение состоит из одного исполняемого файла.
2. Функция «Сохранить» теперь сохраняет каждое новое поле за новым номером в папку «SAVE». Теперь сгенерированные поля можно использовать повторно для тренировок команд.
3. Функция «Печать» теперь не только выводит поле на печать, но и сохраняет его в туже папку «SAVE» за новым номером. Также при выводе на печать указывает название соревнования «Траектория-пазл» и номер варианта поля.

**Версия 1.2**

В версии 1.2 было доработано следующее:

1. При генерации поля добавился 6 этап (см. Процесс генерации поля) – проверка линии на количество компонент связности. Теперь невозможно сгенерировать поле, состоящее из более чем одной компоненты связности.
2. Добавлена кнопка с настройками, вызывающая диалоговое окно. В настройках мы можем задать название соревнования, путь для сохранения генерируемых полей, количество элементов поля.

**Версия 1.3**

В версии 1.3 было доработано следующее:

1. Добавлена легенда к карте, в которой перечисляется количество различных видов пазлов для конфигурирования поля.
2. Исправлены мелких баги и недочёты.

# Заключение

Поставленная нами цель создать приложение, готовое к использованию для при проведении соревнований по робототехнике в Тамбовской области была полностью достигнута.

В ходе работы над проектом было создано приложение, которое позволяет генерировать неограниченное количество полей для соревнования роботов категории «Траектория-пазл» за короткий промежуток время.

Сгенерированное поле удовлетворяет двум требованиям: поле имеет единственную компоненту связности, то есть начав движение в любой клетке поля рано или поздно можно в неё вернуться, посетив все до одной клетки, содержащие линию; учитывается количество пазлов каждого типа, из которого собирается поле.

Сгенерированные поля можно сохранить в виде изображений в папку, заданную в настройках, а также вывести на печать. Для вывода на печать можно задать название соревнований. При этом на отпечатанном банке отображается название соревнований, номер варианта поля, само поле и легенда с указанием количества пазлов каждого вида, необходимых для сборки поля.

# Список используемых источников

1. Дуонсон М. Программируем на Python. – СПб.: Питер, 2016. – 416 с.: ил.
2. Qt Documentation [Электронный ресурс]. URL: <https://doc.qt.io/QT-5/qfiledialog.html>
3. Диалоговые окна в PyQt5 [Электронный ресурс].

URL: <https://pythonworld.ru/gui/pyqt5-dialogs.html>

1. PyQt5 file dialog [Электронный ресурс].

URL: <https://pythonspot.com/pyqt5-file-dialog/>

1. PyQt5.Qt.QDialog [Электронный ресурс].

URL: <https://programtalk.com/python-examples/PyQt5.Qt.QDialog/>

1. Мэтиз Эрик. Изучаем Python. Программирование игр, визуализация данных, веб-приложения. — СПб.: Питер, 2017. — 496 с.: ил.
2. Прохоренок Н.А., Python 3 и PyQT5. Разработка приложений / Н.А. Прохоренок, В.А. Дронов. – СПб.: БХВ-Петербург, 2016. – 832 с.: ил.
3. Правила соревнования «Траектория: противостояние» (РРО-2015) [Электронный ресурc].

URL: <http://robolymp.ru/rules-and-regulations/basic-rules/>

1. Правила соревнования «Траектория-карта» (РРО-2016) [Электронный ресурс].

URL: <http://robolymp.ru/season-2016/rules-and-regulations/traektoriya-karta/>

1. Материалы V открытого межрегионального фестиваля робототехники Тамбовской области [Электронный ресурс].

URL: <http://robo.tofmal.ru/?folder=13>

1. Материалы VI открытого межрегионального фестиваля робототехники Тамбовской области [Электронный ресурс].

URL: <http://robo.tofmal.ru/?folder=14>

1. Материалы Регионального этапа РРО-2017 в Тамбовской области [Электронный ресурс]. URL: <http://robo.tofmal.ru/?folder=15>
2. Материалы Регионального этапа РРО-2018 в Тамбовской области [Электронный ресурс]. URL: <http://robo.tofmal.ru/?folder=17>

# Приложения

## Приложение 1. Код программы

**from** PyQt5 **import** QtGui, QtWidgets  
**from** PyQt5 **import** uic  
**from** PyQt5.QtWidgets **import** QInputDialog  
**from** random **import** choice  
**from** time **import** sleep  
**from** pprint **import** pprint  
**from** PyQt5.Qt **import** \*  
**import** os  
**from** PIL **import** ImageGrab  
**import** sys  
  
n, m = 4, 8  
flag\_print = **False**contest\_name = **'ТРАЕКТОРИЯ-ПАЗЛ'**path = os.getcwd()  
path\_set = path  
path\_default = **False**d = {**"Г"**: 0, **"С"**: 0, **"Х"**: 0, **"Т"**: 0, **'l'**: 0}  
**if** path[-4:] != **'SAVE'**:  
 path += **'\\SAVE'***# Проход по графу из верхнего правого угла с целью посещения всех вершин из данной компоненты связанности***def** dfs(x, i, j, used):  
 used[i][j] = **True  
 if** i < n - 1 **and not** used[i + 1][j] **and** x[i + 1][j] **in** [**'tb'**, **'trbl'**, **'tr'**, **'tl'**,  
 **'ctr'**, **'ctl'**, **'trb'**, **'tbl'**, **'trl'**]:  
 dfs(x, i + 1, j, used)  
 **if** j < m - 1 **and not** used[i][j + 1] **and** x[i][j + 1] **in** [**'rl'**, **'trbl'**, **'tl'**, **'bl'**,  
 **'ctl'**, **'cbl'**, **'rbl'**, **'tbl'**, **'trl'**]:  
 dfs(x, i, j + 1, used)  
  
 **if** i >= 1 **and not** used[i - 1][j] **and** x[i - 1][j] **in** [**'tb'**, **'trbl'**, **'bl'**, **'rb'**,  
 **'cbl'**, **'crb'**, **'trb'**, **'rbl'**, **'tbl'**]:  
 dfs(x, i - 1, j, used)  
 **if** j >= 1 **and not** used[i][j - 1] **and** x[i][j - 1] **in** [**'rl'**, **'trbl'**, **'tr'**, **'rb'**,  
 **'ctr'**, **'crb'**, **'trb'**, **'rbl'**, **'trl'**]:  
 dfs(x, i, j - 1, used)  
  
  
*# Поиск в графе непосещённых пустых клеток***def** find\_clear\_area(x, used):  
 **for** i **in** range(n):  
 **for** j **in** range(m):  
 **if** x[i][j] == **''**:  
 used[i][j] = **True***# Проверка графа на связанность***def** check\_connectivity(used):  
 count = 0  
 **for** i **in** range(n):  
 **for** j **in** range(m):  
 **if** used[i][j]:  
 count += 1  
 **return** count == n \* m  
  
  
**def** count\_elements(x):  
 **global** d  
 d = {**"Г"**: 0, **"С"**: 0, **"Х"**: 0, **"Т"**: 0, **'l'**: 0}  
 **for** i **in** range(n):  
 **for** j **in** range(m):  
 **if** x[i][j] == **'trbl'**:  
 d[**'Х'**] += 1  
 **elif** x[i][j] **in** [**'rl'**, **'tb'**]:  
 d[**'l'**] += 1  
 **elif** x[i][j] == **''**:  
 **continue  
 elif** x[i][j] **in** [**'trb'**, **'rbl'**, **'trl'**, **'tbl'**]:  
 d[**'Т'**] += 1  
 **else**:  
 **if 'c' in** x[i][j]:  
 d[**'С'**] += 1  
 **else**:  
 d[**'Г'**] += 1  
 print(d)  
  
  
**def** check\_elements(parent):  
 **try**:  
 **if** d[**'Г'**] <= int(parent.new\_window.lineEdit\_3.text()) **and** \  
 d[**'С'**] <= int(parent.new\_window.lineEdit\_4.text()) **and** \  
 d[**'Х'**] <= int(parent.new\_window.lineEdit\_2.text()) **and** \  
 d[**'Т'**] <= int(parent.new\_window.lineEdit.text()) **and** \  
 d[**'l'**] <= int(parent.new\_window.lineEdit\_5.text()):  
 **return True  
 else**:  
 **return False  
 except** BaseException:  
 warning = QDialog()  
 warning.setLayout(QVBoxLayout())  
 warning.layout().addWidget(QLabel(**'<h1 style="color:red">'  
 'Количество элементов поля должно'  
 ' задаваться числом!</h1>'**))  
 warning.setWindowTitle(**'WARNING!'**)  
 warning.exec\_()  
 **return None  
  
  
def** make\_map(n=4, m=8, parent=**None**):  
 **def** gen(n=4, m=8):  
 a = [[**''**] \* m **for** \_ **in** range(n)]  
 *# Заполнение угловых клеток* a[0][0] = choice([**'rb'**, **'crb'**])  
 a[0][m - 1] = choice([**'bl'**, **'cbl'**])  
 a[n - 1][0] = choice([**'tr'**, **'ctr'**])  
 a[n - 1][m - 1] = choice([**'tl'**, **'ctl'**])  
 *# pprint(a)  
 # print()* **for** i **in** range(1, m - 2):  
 *# Заполнение верхней строки* **if** a[0][i - 1] **in** [**'rb'**, **'rbl'**, **'crb'**, **'rl'**]:  
 a[0][i] = choice([**'rbl'**, **'bl'**, **'rl'**, **'cbl'**])  
 **else**:  
 a[0][i] = choice([**'rb'**, **'crb'**])  
 *# Заполнение нижней строки* **if** a[n - 1][i - 1] **in** [**'tr'**, **'ctr'**, **'trl'**, **'rl'**]:  
 a[n - 1][i] = choice([**'tl'**, **'ctl'**, **'trl'**, **'rl'**])  
 **else**:  
 a[n - 1][i] = choice([**'tr'**, **'ctr'**])  
  
 *# Заполнение предпоследней клетки в верхней строке* **if** a[0][m - 3] **in** [**'rb'**, **'rbl'**, **'crb'**, **'rl'**]:  
 a[0][m - 2] = choice([**'rbl'**, **'rl'**])  
 **else**:  
 a[0][m - 2] = choice([**'rb'**, **'crb'**])  
 *# Заполнение предпоследней клетки в нижней строке* **if** a[n - 1][m - 3] **in** [**'tr'**, **'ctr'**, **'trl'**, **'rl'**]:  
 a[n - 1][m - 2] = choice([**'trl'**, **'rl'**])  
 **else**:  
 a[n - 1][m - 2] = choice([**'tr'**, **'ctr'**])  
 *# pprint(a)  
 # print()* **for** i **in** range(1, n - 1):  
 *# Заполнение левого столбца* a[i][0] = choice([**'trb'**, **'tb'**])  
 *# Заполнение правого столбца* a[i][m - 1] = choice([**'tbl'**, **'tb'**])  
 *# pprint(a)  
 # print()  
  
 # Заполнение центральной части карты* **for** i **in** range(1, n - 2):  
 **for** j **in** range(1, m - 2):  
 **if** a[i][j - 1] **in** [**'ctl'**, **'cbl'**, **'bl'**, **'tl'**, **'tb'**, **'tbl'**]:  
 **if** a[i - 1][j] **in** [**'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[i][j] = choice([**'rb'**, **'crb'**])  
 **else**:  
 a[i][j] = choice([**'tr'**, **'ctr'**, **'trb'**, **'tb'**])  
 **else**:  
 **if** a[i - 1][j] **in** [**'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[i][j] = choice([**'bl'**, **'cbl'**, **'rbl'**, **'rl'**])  
 **else**:  
 a[i][j] = choice([**'tl'**, **'ctl'**, **'tbl'**, **'trl'**, **'trbl'**])  
 *# pprint(a)  
 # print()  
  
 # Заполнение предспоследней строки карты* **for** i **in** range(1, m - 2):  
 *# Если нижняя закрыта* **if** a[n - 1][i] == **'rl' or** a[n - 1][i] == **''**:  
 *# Если левая закрыта* **if** a[n - 2][i - 1] **in** [**''**, **'ctl'**, **'cbl'**, **'bl'**, **'tl'**, **'tb'**, **'tbl'**]:  
 *# Если верхняя закрыта* **if** a[n - 3][i] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[n - 2][i] = **''** *# Если верхняя открыта* **else**:  
 a[n - 2][i] = choice([**'tr'**, **'ctr'**])  
 *# Если левая открыта* **else**:  
 *# Если верхняя закрыта* **if** a[n - 3][i] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[n - 2][i] = **'rl'** *# Если верхняя открыта* **else**:  
 a[n - 2][i] = choice([**'tl'**, **'ctl'**, **'trl'**])  
 *# Если нижняя открыта* **else**:  
 *# Если левая закрыта* **if** a[n - 2][i - 1] **in** [**''**, **'ctl'**, **'cbl'**, **'bl'**, **'tl'**, **'tb'**, **'tbl'**]:  
 *# Если верхняя закрыта* **if** a[n - 3][i] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[n - 2][i] = choice([**'rb'**, **'crb'**])  
 *# Если верхняя открыта* **else**:  
 a[n - 2][i] = choice([**'tb'**, **'trb'**])  
 *# Если левая открыта* **else**:  
 *# Если верхняя закрыта* **if** a[n - 3][i] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[n - 2][i] = choice([**'bl'**, **'cbl'**, **'rbl'**])  
 *# Если верхняя открыта* **else**:  
 a[n - 2][i] = choice([**'tbl'**, **'trbl'**])  
 *# pprint(a)  
 # print()  
  
 # Заполнение предспоследнего столбца карты* **for** i **in** range(1, n - 2):  
 *# print(i, m - 2)  
 # Если правая закрыта* **if** a[i][m - 1] == **'tb' or** a[i][m - 1] == **''**:  
 *# Если левая закрыта* **if** a[i][m - 3] **in** [**''**, **'ctl'**, **'cbl'**, **'bl'**, **'tl'**, **'tb'**, **'tbl'**]:  
 *# Если верхняя закрыта* **if** a[i - 1][m - 2] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[i][m - 2] = **''** *# Если верхняя открыта* **else**:  
 a[i][m - 2] = **'tb'** *# Если левая открыта* **else**:  
 *# Если верхняя закрыта* **if** a[i - 1][m - 2] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[i][m - 2] = choice([**'bl'**, **'cbl'**])  
 *# Если верхняя открыта* **else**:  
 a[i][m - 2] = choice([**'tl'**, **'ctl'**, **'tbl'**])  
 *# Если правая открыта* **else**:  
 *# Если левая закрыта* **if** a[i][m - 3] **in** [**''**, **'ctl'**, **'cbl'**, **'bl'**, **'tl'**, **'tb'**, **'tbl'**]:  
 *# Если верхняя закрыта* **if** a[i - 1][m - 2] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[i][m - 2] = choice([**'rb'**, **'crb'**])  
 *# Если верхняя открыта* **else**:  
 a[i][m - 2] = choice([**'tr'**, **'ctr'**, **'trb'**])  
 *# Если левая открыта* **else**:  
 *# Если верхняя закрыта* **if** a[i - 1][m - 2] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[i][m - 2] = choice([**'rl'**, **'rbl'**])  
 *# Если верхняя открыта* **else**:  
 a[i][m - 2] = choice([**'trbl'**, **'trl'**])  
 *# pprint(a)  
 # print()  
  
 # Если нижняя закрыта* **if** a[n - 1][m - 2] **in** [**'rl'**, **''**]:  
 *# Если правая закрыта* **if** a[n - 2][m - 1] **in** [**'tb'**, **''**]:  
 *# Если левая закрыта* **if** a[n - 2][m - 3] **in** [**''**, **'ctl'**, **'cbl'**, **'bl'**, **'tl'**, **'tb'**, **'tbl'**]:  
 *# Если верхняя закрыта* **if** a[n - 3][m - 2] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[n - 2][m - 2] = **''** *# Если верхняя открыта* **else**:  
 **return False** *# Если левая открыта* **else**:  
 *# Если верхняя закрыта* **if** a[n - 3][m - 2] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 **return False** *# Если верхняя открыта* **else**:  
 a[n - 2][m - 2] = choice([**'tl'**, **'ctl'**])  
 *# Если правая открыта* **else**:  
 *# Если левая закрыта* **if** a[n - 2][m - 3] **in** [**''**, **'ctl'**, **'cbl'**, **'bl'**, **'tl'**, **'tb'**, **'tbl'**]:  
 *# Если верхняя закрыта* **if** a[n - 3][m - 2] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 **return False** *# Если верхняя открыта* **else**:  
 a[n - 2][m - 2] = choice([**'tr'**, **'ctr'**])  
 *# Если левая открыта* **else**:  
 *# Если верхняя закрыта* **if** a[n - 3][m - 2] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[n - 2][m - 2] = **'rl'** *# Если верхняя открыта* **else**:  
 a[n - 2][m - 2] = **'trl'** *# Если нижняя открыта* **else**:  
 *# Если правая закрыта* **if** a[n - 2][m - 1] **in** [**'tb'**, **''**]:  
 *# Если левая закрыта* **if** a[n - 2][m - 3] **in** [**''**, **'ctl'**, **'cbl'**, **'bl'**, **'tl'**, **'tb'**, **'tbl'**]:  
 *# Если верхняя закрыта* **if** a[n - 3][m - 2] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 **return False** *# Если верхняя открыта* **else**:  
 a[n - 2][m - 2] = **'tb'** *# Если левая открыта* **else**:  
 *# Если верхняя закрыта* **if** a[n - 3][m - 2] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[n - 2][m - 2] = choice([**'bl'**, **'cbl'**])  
 *# Если верхняя открыта* **else**:  
 a[n - 2][m - 2] = **'tbl'** *# Если правая открыта* **else**:  
 *# Если левая закрыта* **if** a[n - 2][m - 3] **in** [**''**, **'ctl'**, **'cbl'**, **'bl'**, **'tl'**, **'tb'**, **'tbl'**]:  
 *# Если верхняя закрыта* **if** a[n - 3][m - 2] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[n - 2][m - 2] = choice([**'rb'**, **'crb'**])  
 *# Если верхняя открыта* **else**:  
 a[n - 2][m - 2] = **'trb'** *# Если левая открыта* **else**:  
 *# Если верхняя закрыта* **if** a[n - 3][m - 2] **in** [**''**, **'trl'**, **'rl'**, **'tr'**, **'ctr'**, **'tl'**, **'ctl'**]:  
 a[n - 2][m - 2] = **'rbl'** *# Если верхняя открыта* **else**:  
 a[n - 2][m - 2] = **'trbl'** *# pprint(a)  
 # print()* **return** a  
  
 *# Генерация поля происходит до тех пор, пока не будет соответстовать требуемым параметрам* count\_varieties = 0  
 **while True**:  
 count\_varieties += 1  
 **if** count\_varieties == 15000:  
 print(**'STOP'**)  
 **return None** x = gen(n, m)  
 used = [[**False**] \* m **for** \_ **in** range(n)]  
 **if** x:  
 dfs(x, 0, 0, used)  
 find\_clear\_area(x, used)  
 count\_elements(x)  
 res\_check = check\_elements(parent)  
 **if** check\_connectivity(used) **and** res\_check:  
 **break  
 elif** res\_check **is None**:  
 **return None  
 return** x  
  
  
**def** set\_settings(parent):  
 parent.new\_window = uic.loadUi(**"settings.ui"**)  
 parent.new\_window.setWindowTitle(**"Настройки"**)  
 *# Обработка нажатия на кнопки* parent.new\_window.save\_settings.clicked.connect(**lambda**: set\_contest\_name(parent))  
 parent.new\_window.path\_button.clicked.connect(**lambda**: set\_path(parent))  
  
  
**def** open\_settings(parent):  
 parent.new\_window.show()  
  
  
*# Настройка названия соревнования***def** set\_contest\_name(parent):  
 **global** contest\_name  
 contest\_name = parent.new\_window.contest\_name.text()  
 success = QDialog()  
 success.setLayout(QVBoxLayout())  
 success.layout().addWidget(QLabel(**'<h1 style="color:green">'  
 'Настройки успешно сохранены!</h1>'**))  
 success.setWindowTitle(**'SUCCESS!'**)  
 success.exec\_()  
  
  
*# Настройка пути сохранения файлов***def** set\_path(parent):  
 folder\_name = QFileDialog.getExistingDirectory(parent, **'Open folder'**)  
 **global** path  
 **global** path\_default  
 **if** folder\_name:  
 path = folder\_name  
 path\_default = **True  
  
  
class** Generator(object):  
 **def** \_\_init\_\_(self, form):  
 self.form = form  
 set\_settings(form)  
 form.setObjectName(**"Map\_gen"**)  
 form.setFixedSize(m \* 100, n \* 100 + 100)  
  
 self.save = QtWidgets.QPushButton(form)  
 self.save.move(440, n \* 100 + 60)  
 self.save.setText(**'Сохранить'**)  
 self.save.clicked.connect(self.save\_img)  
  
 self.print = QtWidgets.QPushButton(form)  
 self.print.move(600, n \* 100 + 60)  
 self.print.setText(**'Печать'**)  
 self.print.clicked.connect(self.print\_img)  
  
 self.gen = QtWidgets.QPushButton(form)  
 self.gen.move(270, n \* 100 + 60)  
 self.gen.setText(**'Сгенерировать'**)  
 self.gen.clicked.connect(self.regen)  
  
 self.settings = QtWidgets.QPushButton(form)  
 self.settings.move(100, n \* 100 + 60)  
 self.settings.setText(**'Настройки'**)  
 self.settings.clicked.connect(**lambda**: open\_settings(form))  
  
 **def** save\_img(self):  
 *# Делаем скриншот окна с картой* p = window.geometry()  
 bbox\_section = (p.x(), p.y(), p.x() + 800, p.y() + 400)  
 screen = ImageGrab.grab(bbox\_section)  
 x = 1  
 os.chdir(path)  
 *# Ищем файл с наибольшим номером* **for** current\_dir, dirs, files **in** os.walk(path):  
 **for** j **in** files:  
 **if** os.access(j, os.F\_OK):  
 **try**:  
 **if** j[:4] == **'line' and** j[-4:] == **'.jpg'**:  
 x = max(int(j[4:-4]) + 1, x)  
 **except** BaseException:  
 **continue** *# Сохраняем поле и возвращаемся в исходную папку* filename = **'line{}.jpg'**.format(x)  
 screen.save(filename)  
 os.chdir(path\_set)  
  
 **def** print\_img(self):  
 *# Сохраняем изображение* **global** flag\_print, d  
 p = window.geometry()  
 bbox\_section = (p.x(), p.y(), p.x() + 800, p.y() + 400)  
 screen = ImageGrab.grab(bbox\_section)  
 x = 1  
 os.chdir(path)  
 **for** current\_dir, dirs, files **in** os.walk(path):  
 **for** j **in** files:  
 **if** os.access(j, os.F\_OK):  
 **try**:  
 **if** j[:4] == **'line' and** j[-4:] == **'.jpg'**:  
 x = max(int(j[4:-4]) + 1, x)  
 **except** BaseException:  
 **continue  
 if not** flag\_print:  
 flag\_print = **True** filename = **'line{}.jpg'**.format(x)  
 screen.save(filename)  
 **else**:  
 x = x - 1  
 *# Печатаем изображение, написав название соревнования, номер варианта и установив альбомную ориентацию* printer = QPrinter()  
 te = QTextEdit()  
 html = **'<h1 align="center">{}<br>'** \  
 **'ВАРИАНТ: {}</h1><br><img src="line{}.jpg"><br>Г-линия: {},'** \  
 **' Скруглённый повторот: {}, X-линия: {}, Т-линия: {}, Прямая: {}'**. \  
 format(contest\_name, x, x, d[**'Г'**], d[**'С'**], d[**'Х'**], d[**'Т'**], d[**'l'**])  
 te.setHtml(html)  
 print\_dialog = QPrintDialog(printer)  
 **if** print\_dialog.exec() == QDialog.Accepted:  
 printer.setOrientation(QPrinter.Landscape)  
 te.print(printer)  
 os.chdir(path\_set)  
  
 **def** regen(self):  
 **global** path\_set  
 os.chdir(path\_set)  
 n = 4  
 m = 8  
 form = self.form  
 form.resize(m \* 100, n \* 100 + 100)  
 a = make\_map(n, m, form)  
 **if** a **is None**:  
 warning = QDialog()  
 warning.setLayout(QVBoxLayout())  
 warning.layout().addWidget(QLabel(**'<h1 style="color:red">'  
 'Не удалось сгенерировать'  
 ' поле с данными параметрами!</h1>'**))  
 warning.setWindowTitle(**'WARNING!'**)  
 warning.exec\_()  
 **return  
 global** flag\_print  
 flag\_print = **False** *# Вывод итоговой карты* resources\_path = os.getcwd()  
 **if** resources\_path[-3:] != **'RES'**:  
 resources\_path = os.getcwd() + **'\\RES'** os.chdir(resources\_path)  
 **for** i **in** range(n):  
 **for** j **in** range(m):  
 angle = 0  
 pic = QtGui.QPixmap()  
 **if** a[i][j] == **''**:  
 pic.load(**"''.jpg"**)  
 **elif** a[i][j] == **'bl'**:  
 pic.load(**'bl.jpg'**)  
 angle = 90  
 **elif** a[i][j] == **'cbl'**:  
 angle = -90  
 pic.load(**'cbl.jpg'**)  
 **elif** a[i][j] == **'crb'**:  
 angle = 180  
 pic.load(**'crb.jpg'**)  
 **elif** a[i][j] == **'ctl'**:  
 pic.load(**'ctl.jpg'**)  
 **elif** a[i][j] == **'ctr'**:  
 pic.load(**'ctr.jpg'**)  
 angle = 90  
 **elif** a[i][j] == **'rb'**:  
 pic.load(**'rb.jpg'**)  
 **elif** a[i][j] == **'rbl'**:  
 pic.load(**'rbl.jpg'**)  
 angle = 90  
 **elif** a[i][j] == **'rl'**:  
 pic.load(**'rl.jpg'**)  
 angle = 90  
 **elif** a[i][j] == **'tb'**:  
 pic.load(**'tb.jpg'**)  
 **elif** a[i][j] == **'tbl'**:  
 pic.load(**'tbl.jpg'**)  
 angle = 180  
 **elif** a[i][j] == **'tl'**:  
 pic.load(**'tl.jpg'**)  
 angle = 180  
 **elif** a[i][j] == **'tr'**:  
 pic.load(**'tr.jpg'**)  
 angle = -90  
 **elif** a[i][j] == **'trb'**:  
 pic.load(**'trb.jpg'**)  
 **elif** a[i][j] == **'rb'**:  
 pic.load(**'rb.jpg'**)  
 **elif** a[i][j] == **'trbl'**:  
 pic.load(**'trbl.jpg'**)  
 **elif** a[i][j] == **'trl'**:  
 pic.load(**'trl.jpg'**)  
 angle = -90  
 pic = pic.scaledToHeight(100)  
 t = QtGui.QTransform().rotate(angle)  
  
 self.lbl2 = QtWidgets.QLabel(form)  
 self.lbl2.move(100 \* j, 100 \* i)  
 self.lbl2.setPixmap(pic.transformed(t))  
 self.lbl2.show()  
 os.chdir(resources\_path[:-4])  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 app = QtWidgets.QApplication(sys.argv)  
 window = QtWidgets.QWidget()  
 ui = Generator(window)  
 window.setWindowTitle(**'MAP GENERATOR | v.1.3'**)  
 window.show()  
 sys.exit(app.exec\_())

## Приложение 2. Варианты различных полей

При помощи нашей программы были сгенерированы следующие карточки полей с односвязным вариантом линии.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

А также были сгенерированы карточки с многосвязным вариантом поля.

|  |  |
| --- | --- |
|  |  |
|  |  |