Министерство образования и науки Российской Федерации

федеральное государственное автономное образовательное учреждение
высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»

Институт дополнительного образования
Высшая инженерная школа

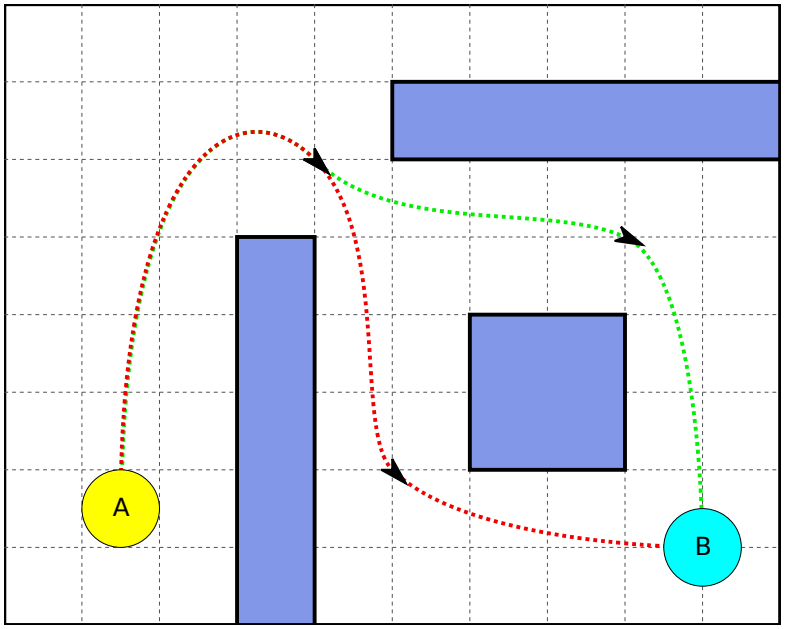
ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

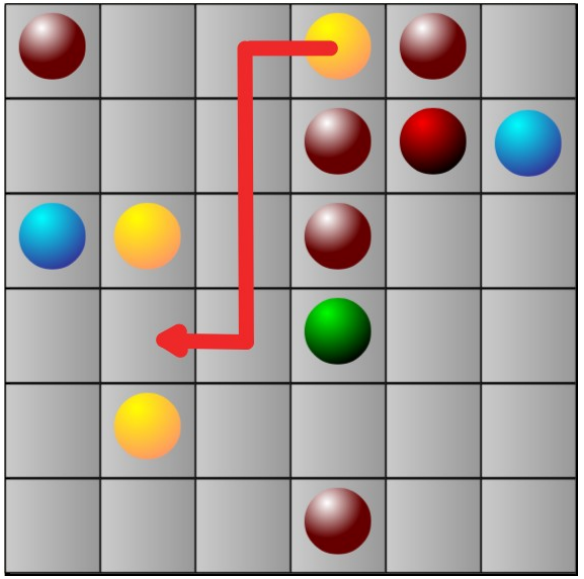
"Реализация алгоритмов поиска пути на примере игры "The Lines"

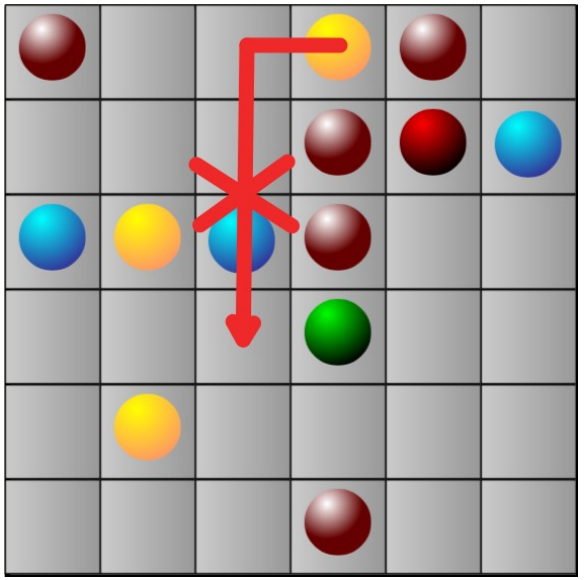
по программе профессиональной переподготовки:
«Разработчик прикладного программного обеспечения (Языки C и C++)»

Выполнил: Бойков Дмитрий Сергеевич
Руководитель: ст. преподаватель Полубенцева Марина Игоревна

Предметная область







7	6	5	6	7	8	9	10	11		19	20	21	22
6	5	4	5	6	7	8	9	10		18	19	20	21
5	4	3	4	5	6	7	8	9		17	18	19	20
4	3	2	3	4	5	6	7	8		16	17	18	19
3	2	1	2	3	4	5	6	7		15	16	17	18
2	1	0	1	2	3	4	5	6		14	15	16	17
3	2	1	2	3	4	5	6	7		13	14	15	16
4	3	2	3	4	5	6	7	8		12	13	14	15
5	4	3	4	5	6	7	8	9	10	11	12	13	14
6	5	4	5	6	7	8	9	10	11	12	13	14	15

Постановка цели и задач

Цель работы: реализовать алгоритмы поиска пути и сравнить их между собой по времени выполнения на примере игры "Lines".

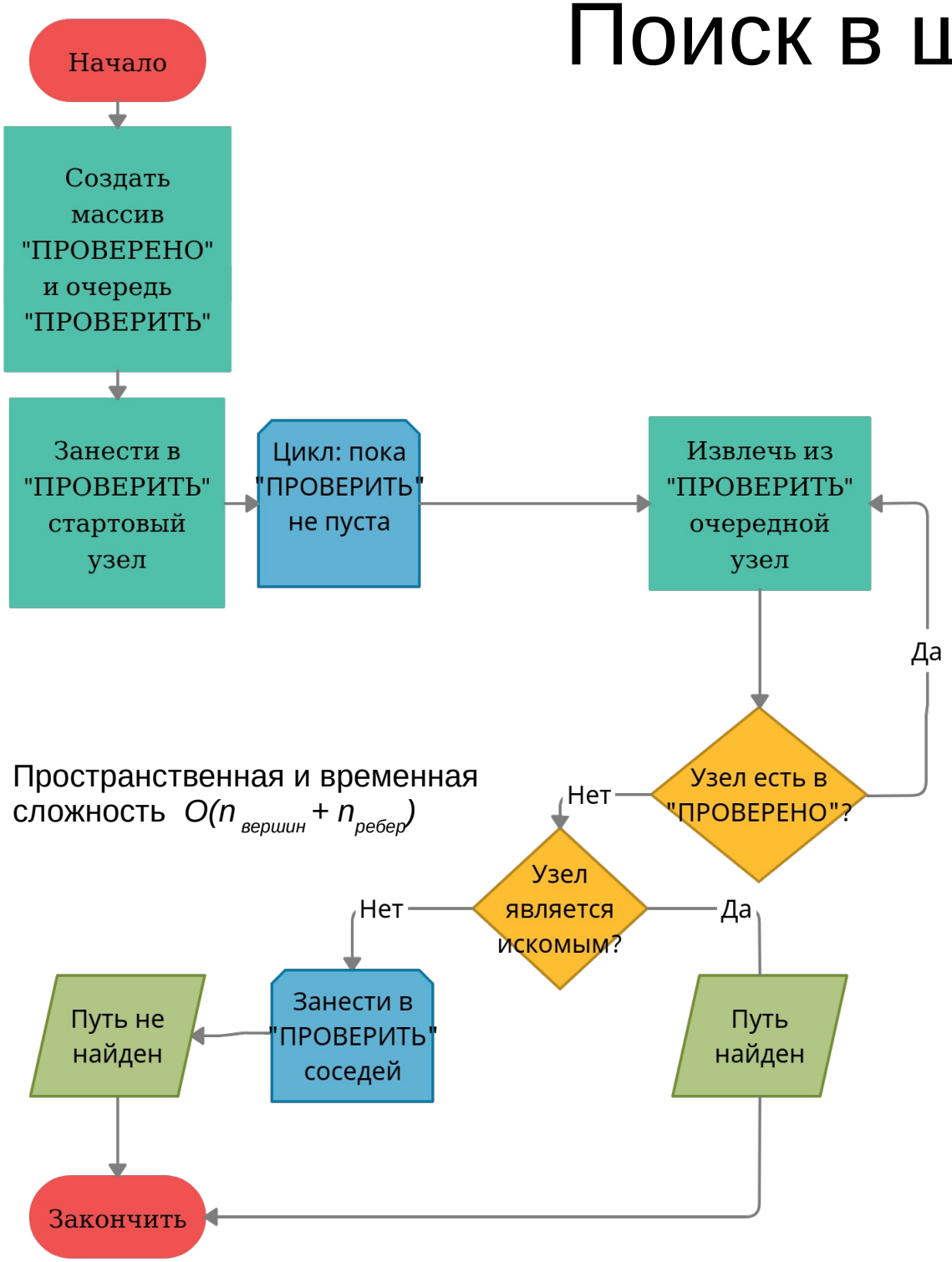
Задачи:

- Обзор предметной области.
- Формирование требований к программе.
- Разработать архитектуру приложения в парадигме MV.
- Реализовать алгоритм поиска в ширину.
- Реализовать жадный алгоритм по первому соответствию.
- Реализовать алгоритм A*.
- Сравнить время выполнения алгоритмов.

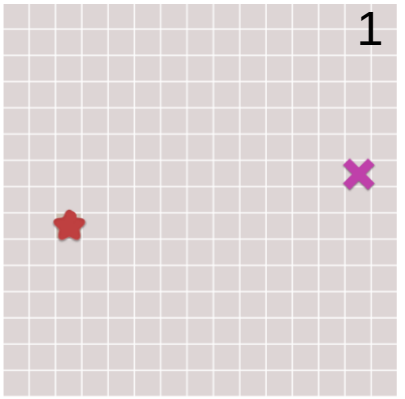
Обзор алгоритмов

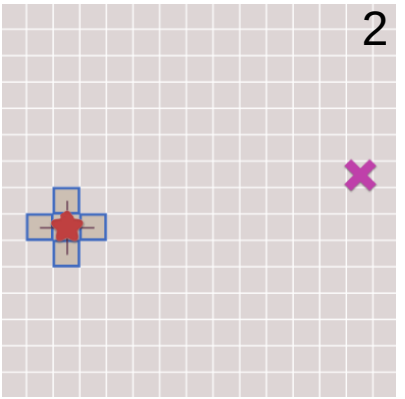
Название	Описание	Примечание
Поиск в ширину	Сначала определяет все соседние узлы, затем все узлы в двух шагах, затем в трех, и так далее, пока цель не достигнута.	Имеет наибольшую временную сложность
Алгоритм Дейкстры	Принимает во внимание стоимость или длину пути и обновляет узлы, если к ним найден лучший путь.	Игнорирует направление к цели, используется для поиска пути во взвешенных графах.
Поиск в глубину	Вместо посещения вначале всех соседей, а потом их наследников, он сначала посещает всех наследников, а только затем переключается на соседей.	Может путаться вокруг себя и тратить время на бессмысленные пути.
«Лучший - первый»	Принимает во внимание знания о пространстве поиска для направления своих усилий. Узлы в исследуемом списке оцениваются по приблизительному оставшемуся расстоянию до цели.	Не учитывает вес пути, путь вокруг препятствия не прямой, а, скорее всего, изгибается вокруг препятствия
A*	Сочетает в себе учет длины предыдущего пути и эвристическую оценку пути до конечной точки	Гарантированно находит кратчайший путь, но потребляет много ресурсов памяти.

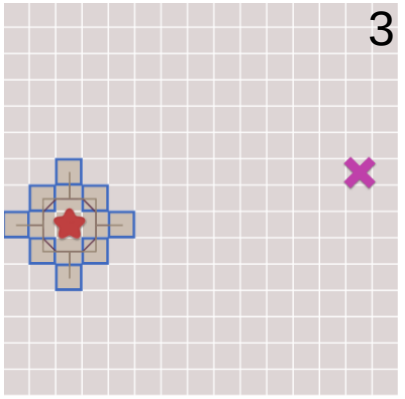
Поиск в ширину

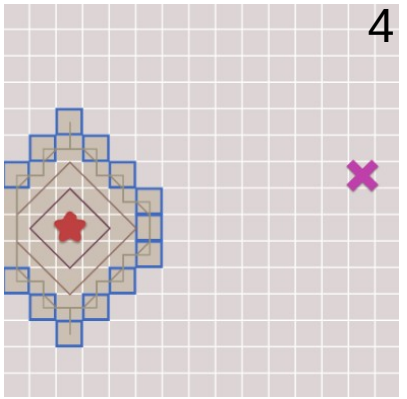


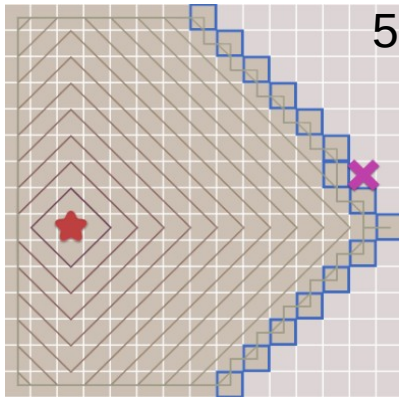
Пространственная и временная сложность $O(n_{вершин} + n_{ребер})$

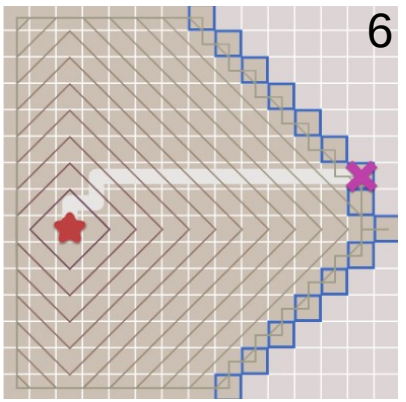










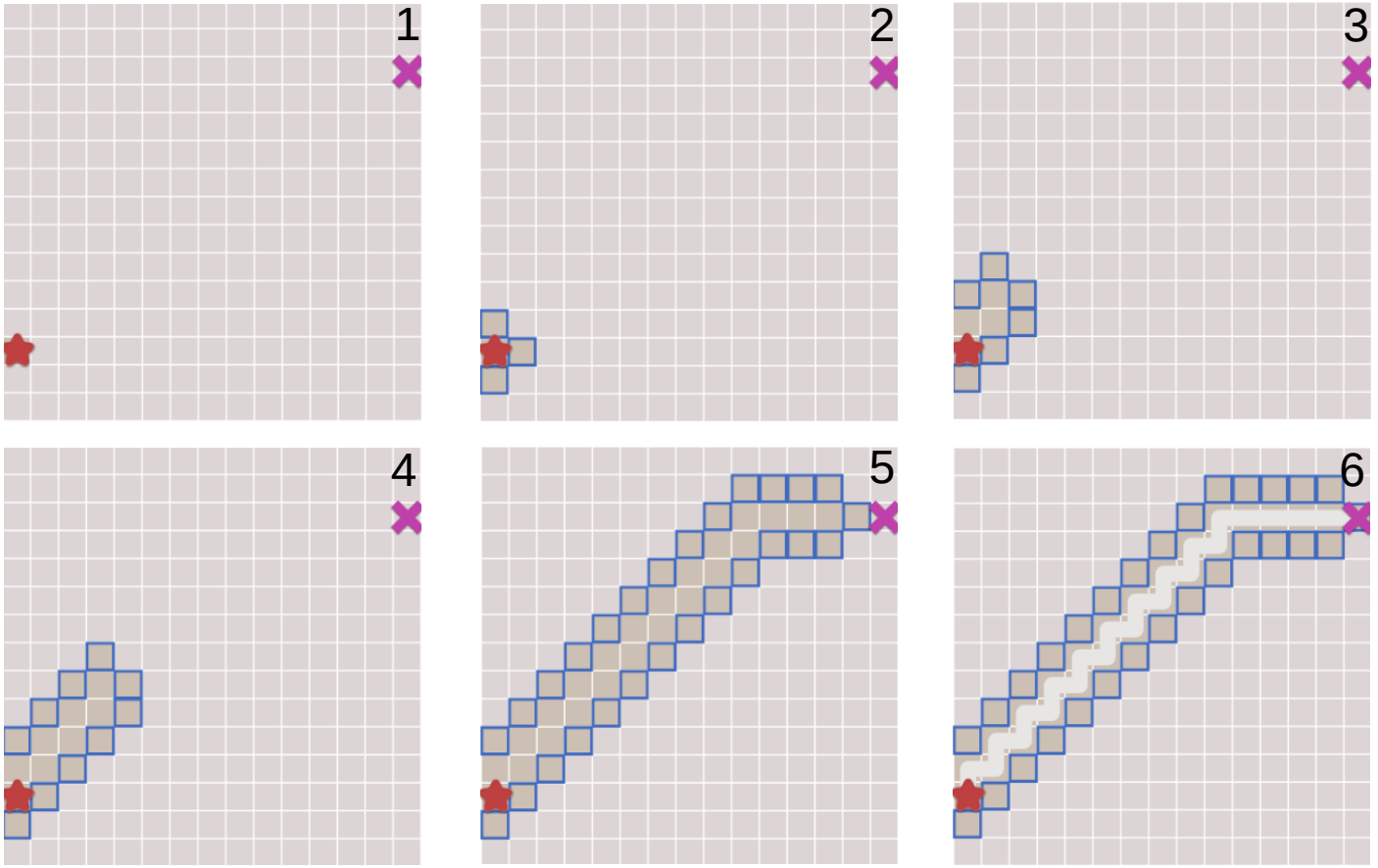


«Первый - лучший»

- Вводим эвристическую функция, сообщающая насколько мы близки к цели.

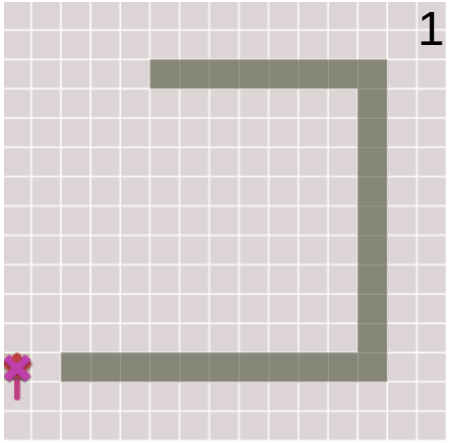
  Например, для сетки это может быть $H(a,b) = |a.x — b.x| + |a.y — b.y|$;

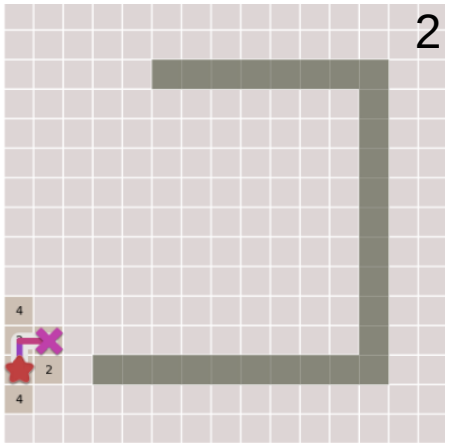
- Вместо FIFO-очереди используем очередь с приоритетами.
- Точка ближайшая к конечной исследуется первой.

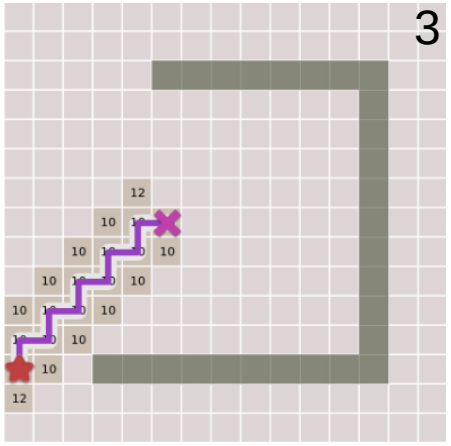


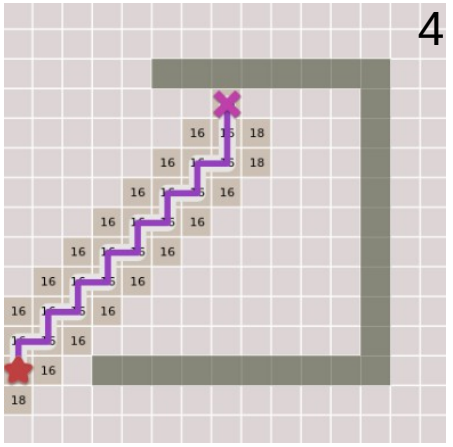
A*

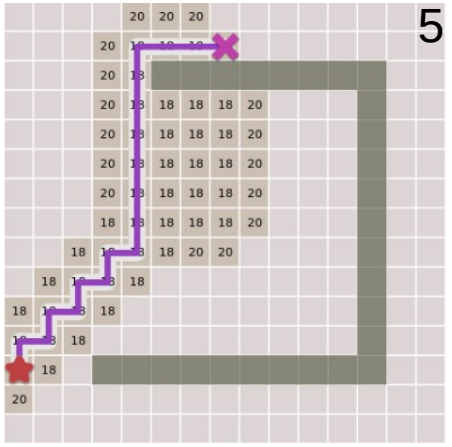
- Для каждого исследуемого узла считаем стоимость перемещения до него из начальной позиции: *C(a)*;
- Может получиться так, что один узел посещается несколько раз с разной стоимостью, поэтому мы добавляем его в очередь «ПРОВЕРИТЬ», если новый путь к этому узлу лучше, чем наилучший предыдущий путь, даже если этот узел уже исследовался.
- Для порядка очереди с приоритетами используем сумму стоимости перемещения до исследуемого узла и эвристическую функцию *F(a) = C(a) + H(a)*;











Архитектура приложения

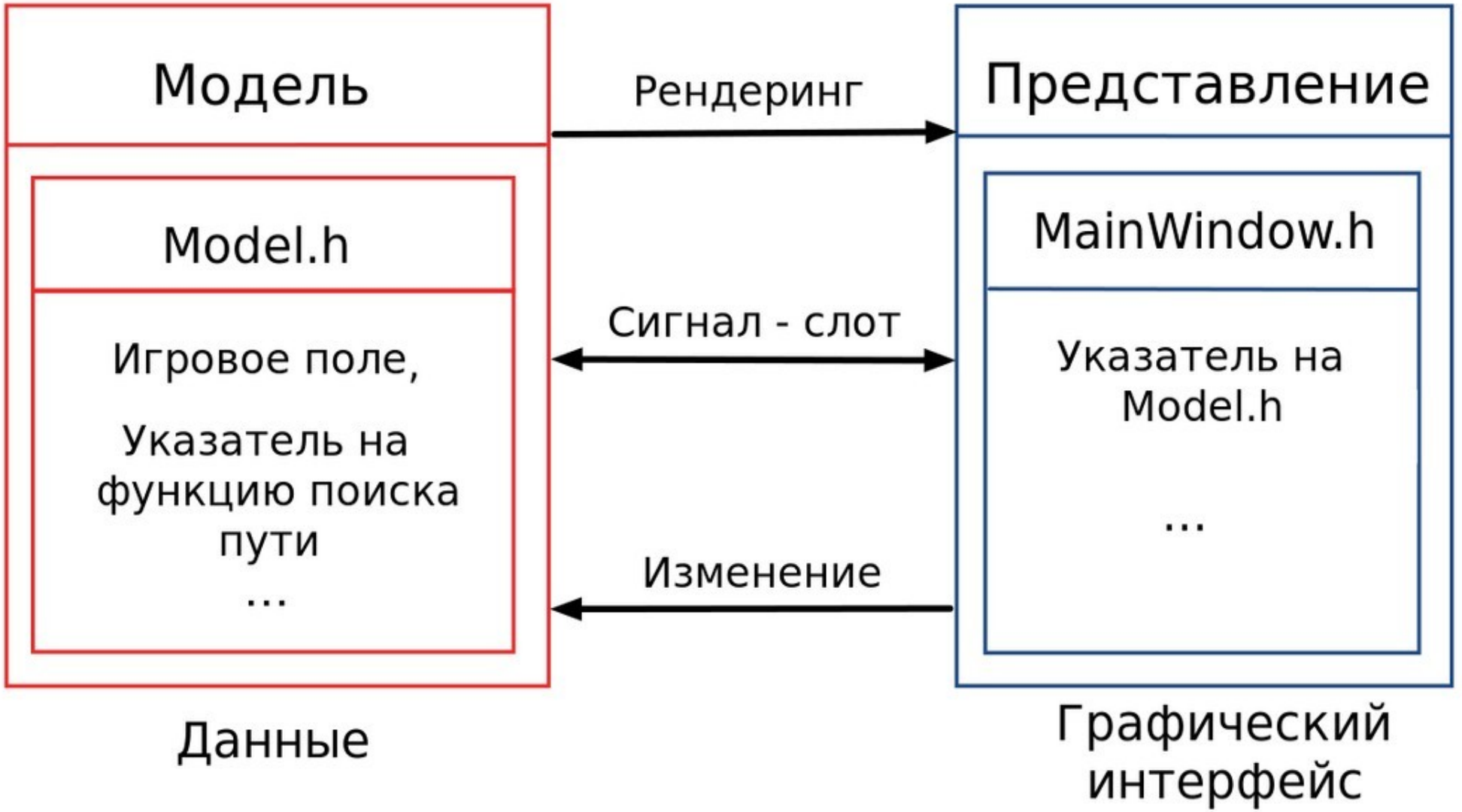
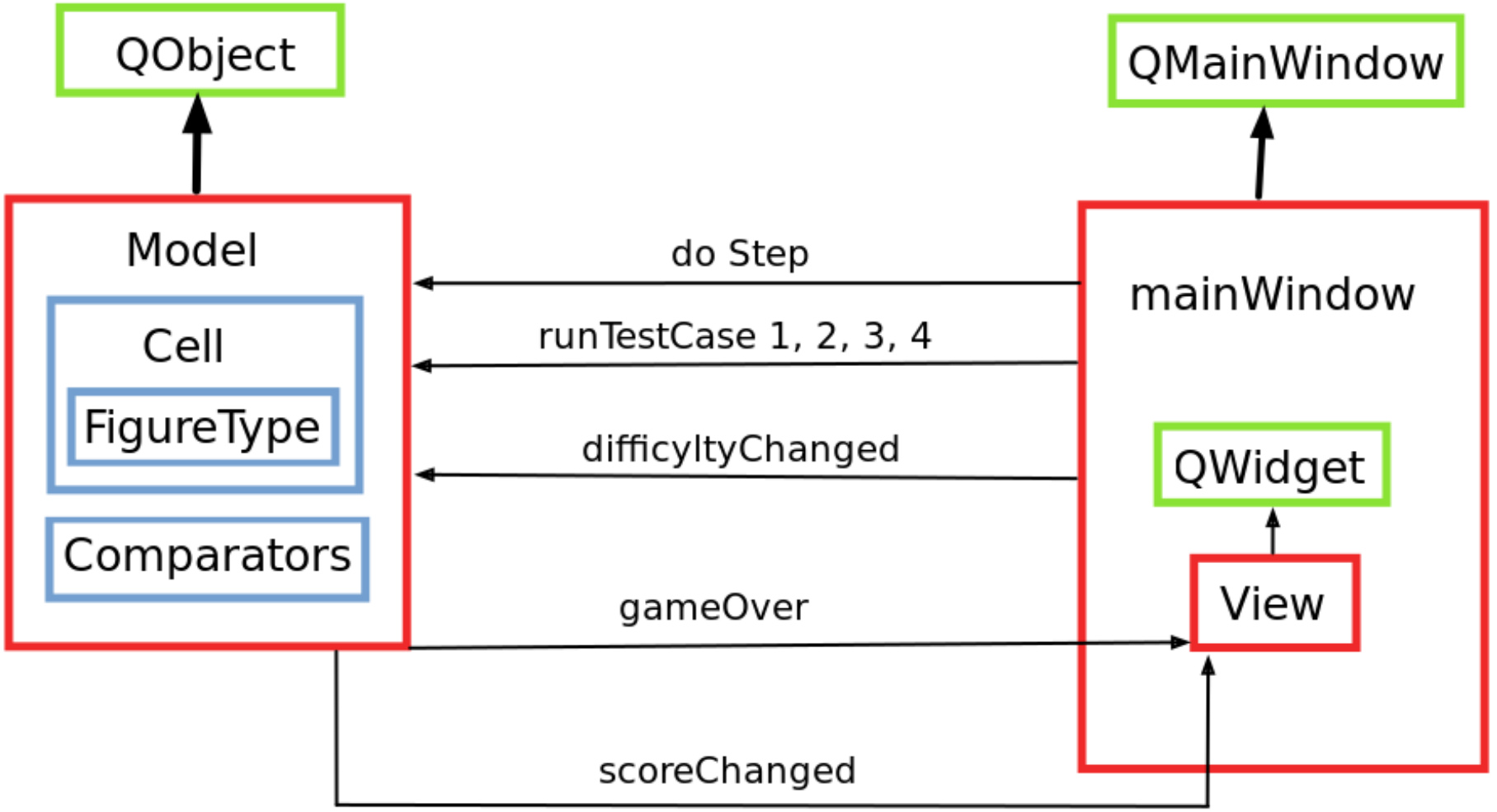


Диаграмма классов



Класс CELL

Член класса	Описание
FigureType _type;	Тип фигуры.
Cell* _parent;	Указатель на клетку после анализа которой алгоритм поиска пути начал анализ текущей клетки.
bool _visited;	Флаг, означающий, что данная клетка уже была проанализирована алгоритмом поиска пути. Используется для предотвращения зацикливания работы алгоритмов поиска пути.
int _heuristic;	Значение эвристической функции для данной клетки.
int _moveCost;	Стоимость передвижения в рассматриваемую клетку из точки старта.
int _y;	Номер строки клетки в сетке.
int _x;	Номер столбца клетки в сетке.

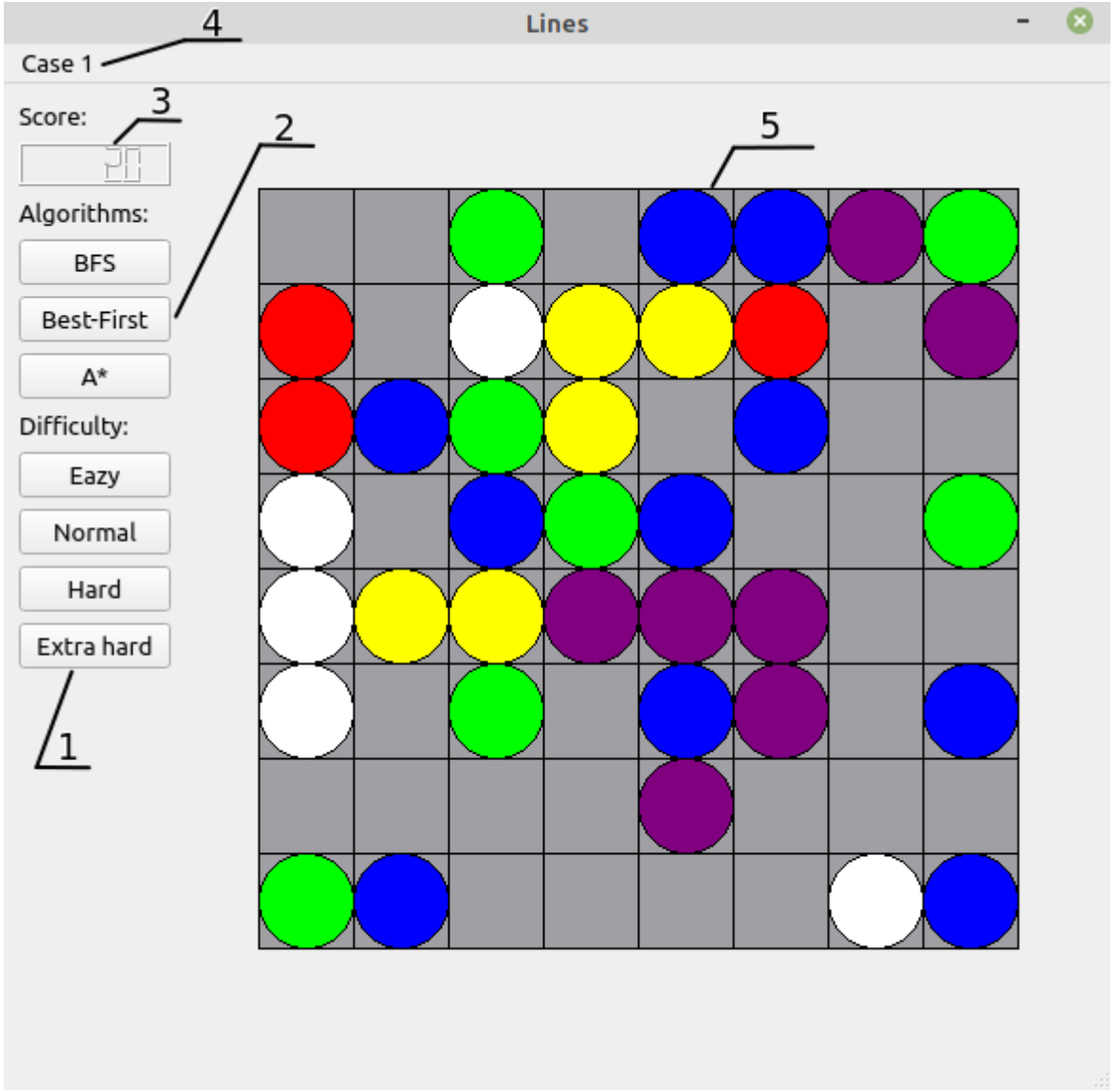
Класс MODEL

Член класса	Описание
Q_OBJECT	Макрос, необходим для работы механизма сигнал-слот Qt.
int _col;	Количество столбцов игрового поля
int _row;	Количество строк игрового поля
int _equalCount;	Количество шаров, необходимых для удаления линии
int _score;	Набранные очки
int _fromCol;	Номер столбца начальной точки пути
int _fromRow;	Номер строки начальной точки пути
int _toCol;	Номер столбца конечной точки пути
int _toRow;	Номер строки конечной точки пути
bool _testMode;	Флаг, для переключения режима работы в тестовый
bool (Model::*_algorithm)();	Указатель на функцию — алгоритма поиска пути, для обеспечения возможности переключения алгоритмов из интерфейса пользователя программы (без пересборки программы)
DifficultyType _difficulty;	Текущая сложность игры
std::mt19937 _mt_rand;	Генератор псевдослучайных чисел «Вихрь — Мерсенна» из стандартной библиотеки.
Gridtype _grid;	Двойной массив клеток (вектор векторов клеток(Cell), игровое поле)

Класс VIEW

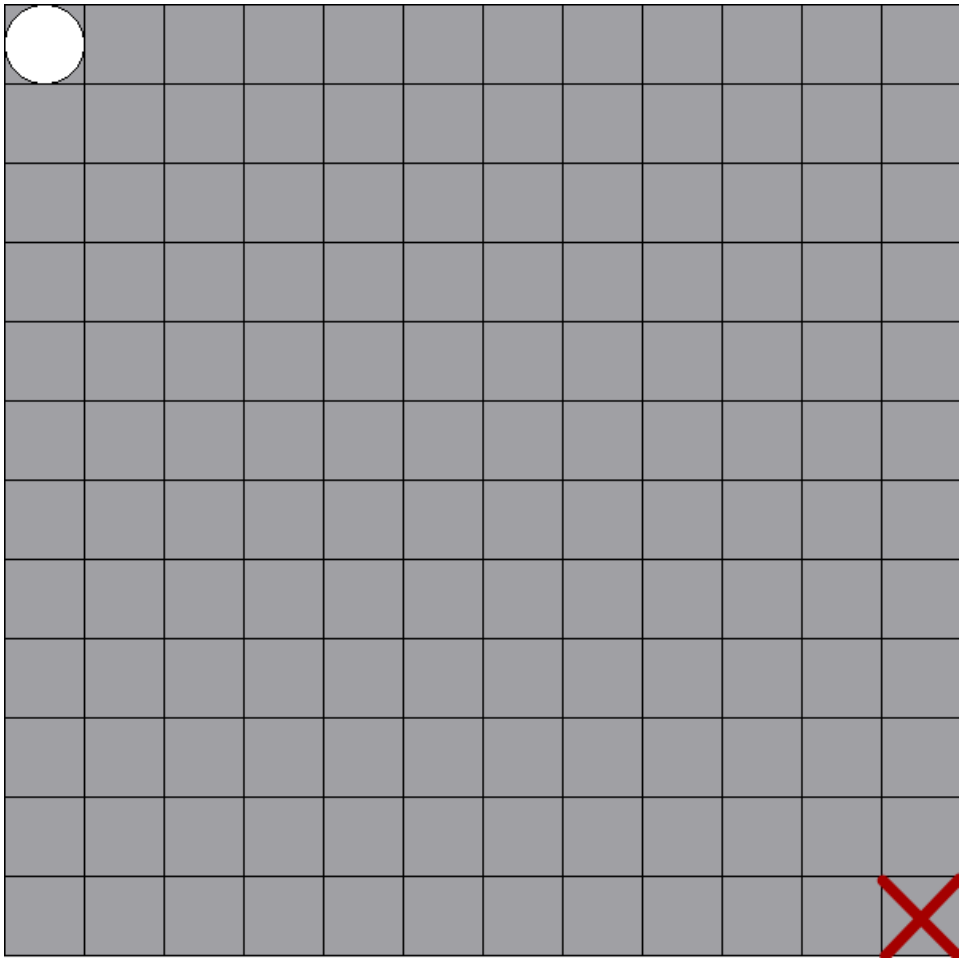
Метод	Описание
void paintEvent(QPaintEvent *event) override;	Отрисовывает игровое поле
void mousePressEvent(QMouseEvent *event) override;	Вызывается при нажатии ЛКМ по игровому полю. Устанавливает начальную точку перемещения (_fromRow, _fromCol класса Model).
void mouseReleaseEvent(QMouseEvent *event) override;	Вызывается при отжатии ЛКМ по игровому полю. Устанавливает конечную точку перемещения (_toRow, _toCol класса Model).

Графический пользовательский интерфейс

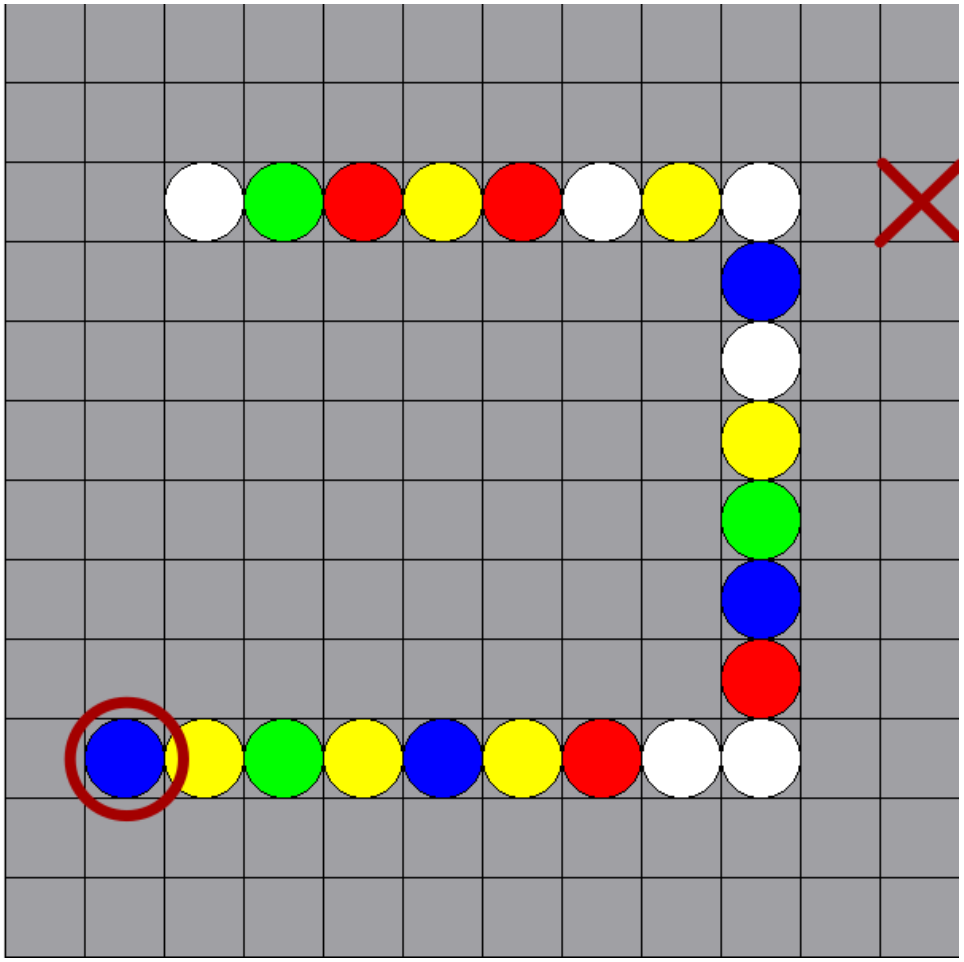


Представлен классом MainWindow.
1 — Кнопки переключения уровня сложности, 2 — кнопки переключения алгоритма поиска пути,
3 — виджет, для отображения набранных очков, 4 — кнопка перехода в тестовый режим, 5 — игровое поле (предоставлено классом View).

Проверка работоспособности алгоритмов и оценка времени выполнения

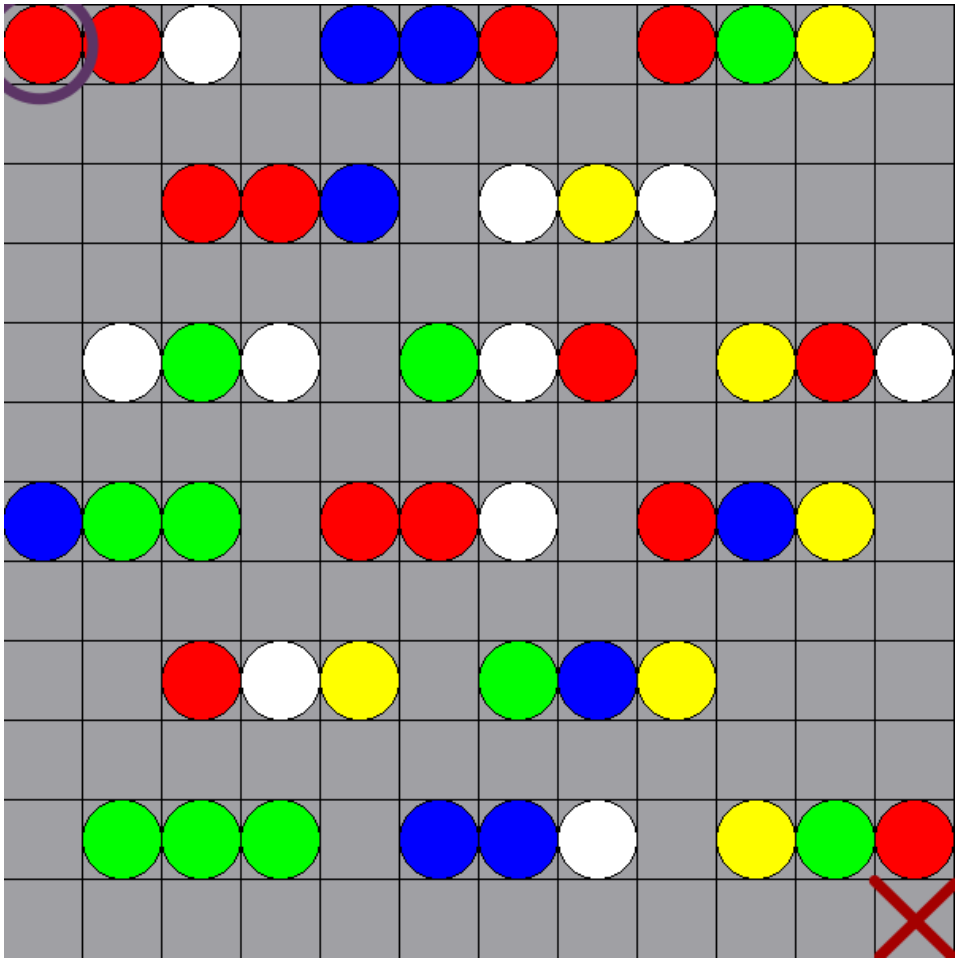


Карта 1

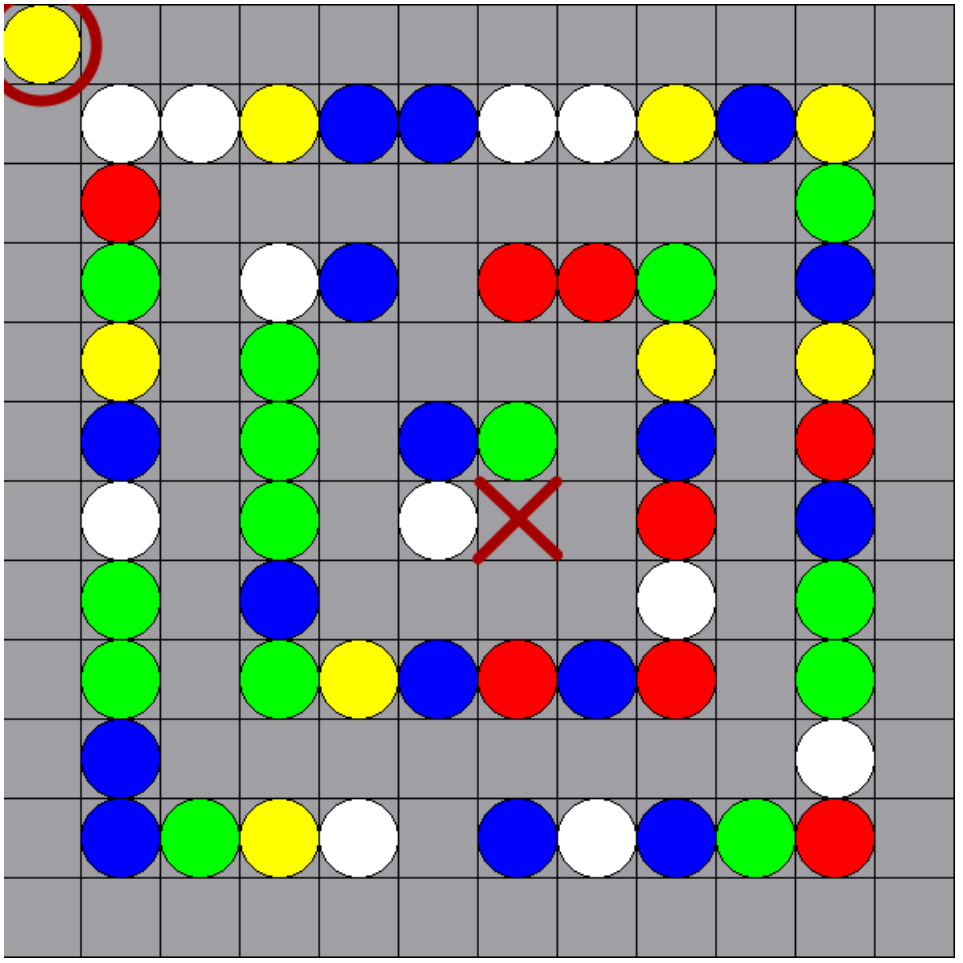


Карта 2

Проверка работоспособности алгоритмов и оценка времени выполнения



Карта 3



Карта 4

Результаты времени выполнения алгоритмов

Алгоритм «Первый-лучший» показал наименьшее время выполнения в рассматриваемых ситуациях.

	BFS	«Первый - лучший»	A*
Карта 1:	30286	8666	39029
Карта 2:	23359	8565	27302
Карта 3:	21013	9541	21156
Карта 4:	15241	6685	16248

Время выполнения алгоритмов, длительность в нс.
(Intel Core2 Quad Q6600 2.40GHz, Linux Mint 20.1 x86_64.)

Заключение

В результате проделанной работы были реализованы алгоритмы поиска пути на примере игры "Lines".

Решены поставленные задачи:

- Выполнен обзор предметной области и наиболее популярных алгоритмов поиска пути.

- Разработана программа соответствующая поставленным требованиям.

- Разработана архитектура приложения в парадигме MV.

- Реализован алгоритм поиска в ширину.

- Реализован жадный алгоритм по первому соответствию.

- Реализован алгоритм A*.

- Алгоритм «Первый — лучший» показал наименьшее время выполнения в рассматриваемых ситуациях.