Федеральное агентство по науке и образованию Российской Федерации

Государственное образовательное учреждение высшего профессионального образования
Волгоградский Государственный Технический Университет
(ВолгГТУ)

Кафедра ПОАС

Логическая игра «Летние линии»

Выполнил:
студент группы ИВТ-264
Мельников М.П.
Проверил:
Андреев Н.В.

Волгоград 2008

Содержание

Постановка задачи

Спроектировать программу, представляющую собой логическую игру “Summer Lines”, являющуюся модификацией игры “Цветные линии”, разработанной российской компанией Gamos в 1992 году.

Условия игры “Цветные линии”: на экране показано квадратное поле 9×9 клеток, на которое в случайные клетки программа выставляет три шарика разных цветов.

- За один ход игрок может передвинуть один шарик, выделив его и указав его новое местоположение.
- Для успешности хода необходимо, чтобы между начальной и конечной клетками существовал путь из свободных клеток.
- Цель игры состоит в удалении максимального количества шариков, которые исчезают при выстраивании шариков одного цвета по пять и более в ряд (по горизонтали, вертикали или диагонали).
- Время, за которое делается ход, значения не имеет.
- Игра заканчивается, когда заканчиваются свободные клетки.

Условия игры “Summer Lines” отличаются следующим:

- Изначально размер поля 6×6 клеток.
- Шарики удаляются, если они выстроены в линию из 4 и более шариков.
- Когда на поле нет свободных клеток поле увеличивается по горизонтали и вертикали на 1 ряд пустых клеток.

Объектом является поле игры, состоящее из клеток, каждая из которых может быть свободна или занята шариком определенного цвета.

Клетка имеет следующие атрибуты:

- Координату по горизонтали (в начале игры целое число от 0 до 6, в конце до 8);
- Координату по вертикали (в начале игры целое число от 0 до 6, в конце до 8);
- Признак занятости (целое число 0 – клетка свободна, 1 – клетка занята красным шариком, 2 – желтым, 3 – зеленым, 4 – синим, 5 – фиолетовым).

Главная функция программы – проведение игры “Summer Lines”.

Основные функции программы:

- Отображение поля игры;
- Псевдослучайное добавление на поле новых шариков
- Удаление шариков по правилам игры;
- Изменение размеров поля;
- Перемещение шарика игроком;

Сервисные функции:

- Организация многооконного интерфейса;
- Вывод справки.

Общие требования к программе:

- Проектируемая программа должна быть многомодульной и иерархической;
- Интерфейс программы должен быть многооконным, а меню – многоуровневым (не менее двух уровней). К наиболее часто используемым командам должны быть назначены эргономичные кнопки на панели инструментов.

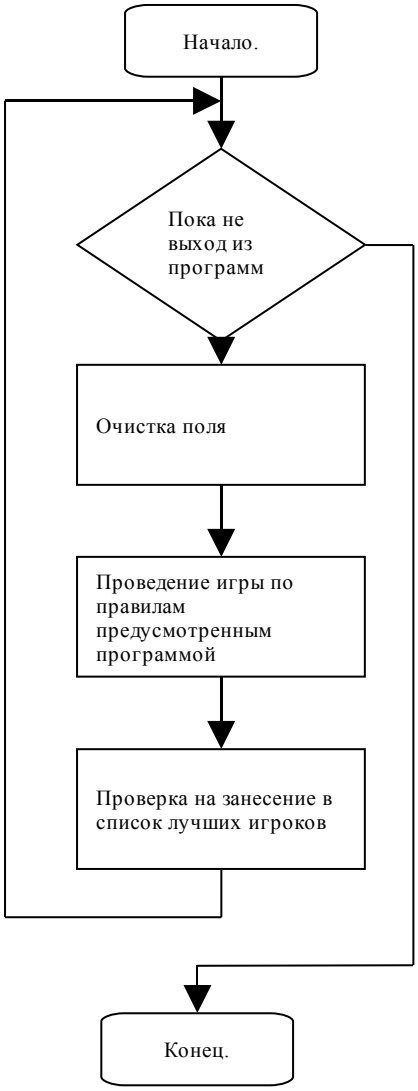
Выводимые данные:

- Поле игры с размещёнными на нём шариками;
- Сообщение, в котором указано, количество набранных очков.
- Список лучших игроков.

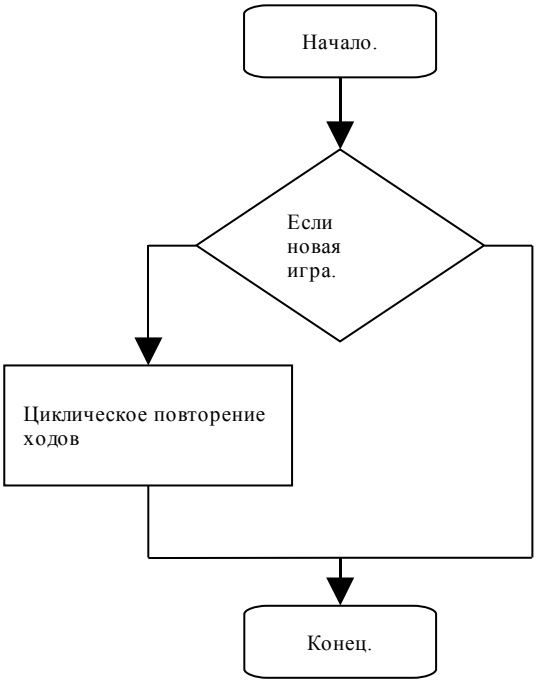
Вводимые данные:

- Координаты клеток, начального и конечного положения шариков;
- Сервисные команды пользователя.

Логическая структура программы
Проведение игры “Summer Lines”



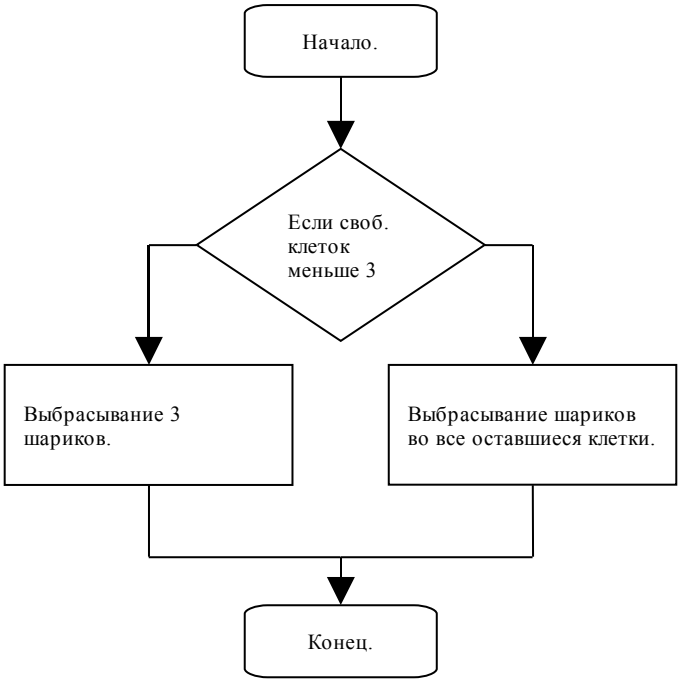
Проведение игры по правилам, предусмотренным программой.



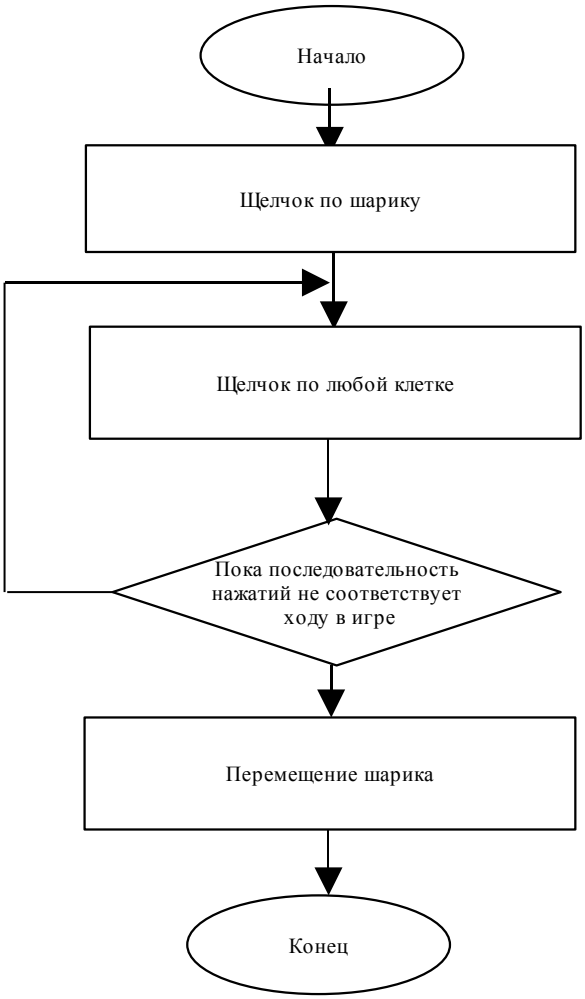
Циклическое повторение ходов.

Выбрасывание и удаление шариков в соответствии с ходами пользователя.

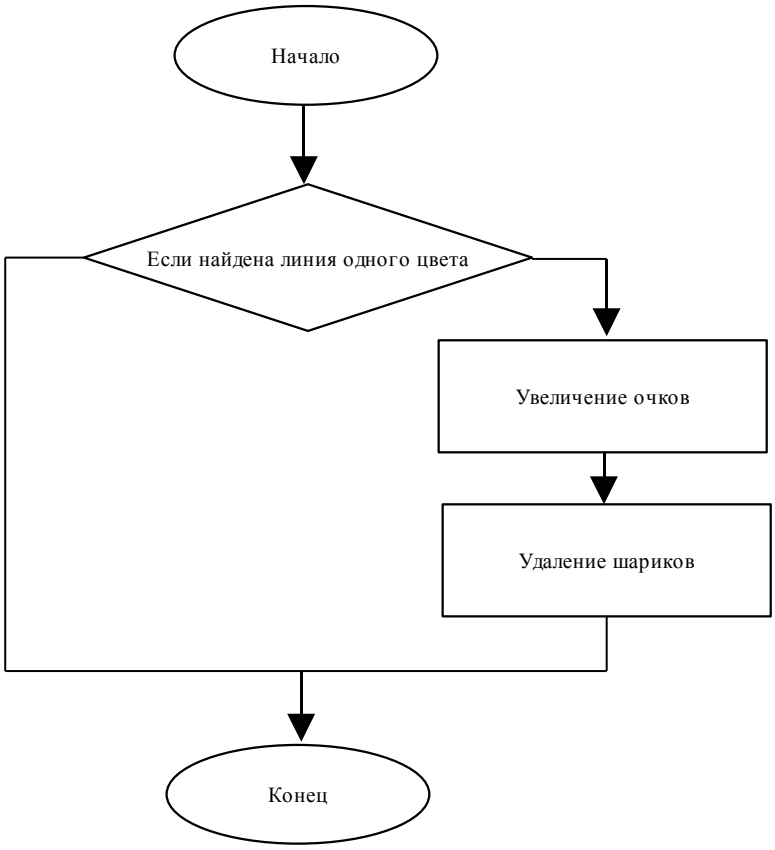
Выбрасывание шариков.



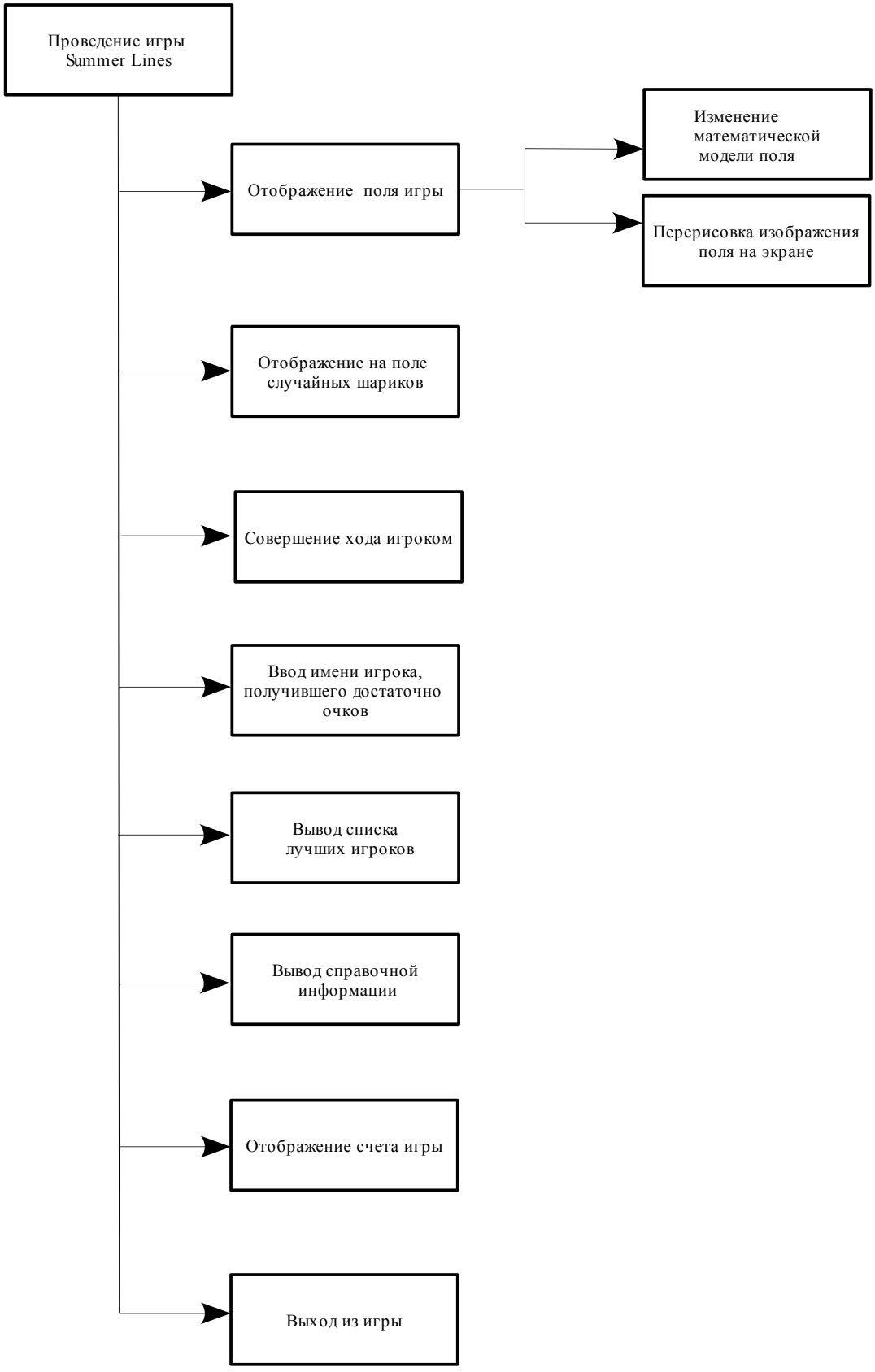
Ход игрока по перемещению шарика.



Удаление шариков, которые составляют прямую из одного цвета.



Функциональная структура



Спецификация функций

№	Название	Действие	Объект	Ограничение	Рисунок
	Проведение игры	Проведение	Игра	При выборе пункта меню «Игра =>Новая игра» или нажатия кнопки «Новая игра» производится запуск новой игры	
1	Отображение поля игры	Отображение	Поле игры	Сразу после начала новой игры поле игры очищается. При совершении хода игроком изменение отображаемой новой игровой ситуации	Рис. 1
1.1	Изменение математической модели поля	Отображение	Математическая модель поля	В соответствии с игровой ситуацией выставление чисел, соответствующих шарикам на поле	
1.2	Перерисовка изображения поля на экране	Отображение	Графическая модель поля	Вывод на экран цветных шариков и пустых клеток, соответствующих математической модели игрового поля	Рис.7
2	Отображение на поле случайных шариков	Отображение	Математическая модель поля Графическая модель поля	Генерация случайного положения и случайного цвета	Рис.4
3	Совершение хода игроком	Совершение	Математическая модель поля Графическая модель поля	Игрок выбирает клетку на поле, в случае, если она соответствует ходу в игре, совершение хода.	Рис.2

4	Ввод имени игрока, получившего достаточно очков	Ввод	Диалог ввода	По завершению игры, вывод стандартного диалога ввода строки на экран для ввода имени игрока. Выход из сообщения осуществляется его закрытием или нажатием клавиши «ENTER» или кнопки «ОК»	Рис.5
5	Вывод списка лучших игроков	Вывод	Сообщение	По завершению игры вывод на экран стандартного сообщения со списком лучших игроков, хранящемся в отдельно файле. Выход из сообщения осуществляется его закрытием или нажатием клавиши «ENTER» или кнопки «ОК»	Рис.6
6	Вывод справочной информации	Вывод	Сообщение	При выборе пункта меню «Помощь => Справка» или «Помощь=> О программе» Вывод на экран информации с правилами игры или контактными сведениями автора. Выход из сообщения осуществляется его закрытием или нажатием клавиши «ENTER» или	Рис.8 Рис.9

				кнопки «ОК»	
7	Отображение счета игры	Отображение	Игра	При изменении счета игры его измененное значение отображается на экране. В начале новой игры счет обнуляется	Рис.3
8	Выход их игры	Выход		Действие происходит по выбору пункта меню “Игра→Выход” или закрытия окна.	

Иерархия главного меню

- Игра
  - Новая игра
  - Лучшие игроки...
  - Выход
- Справка
  - О программе…
  - Помощь...

Макеты экранных форм

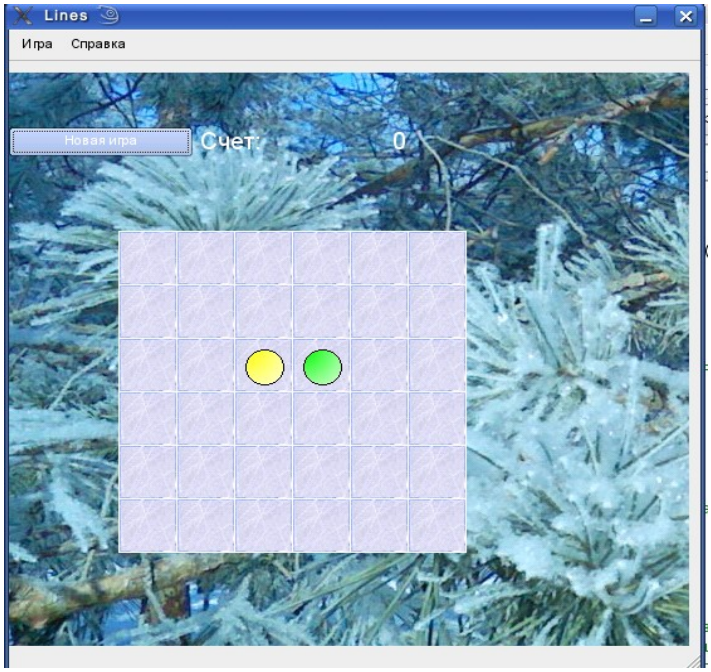


Рис.1. Главное окно программы.

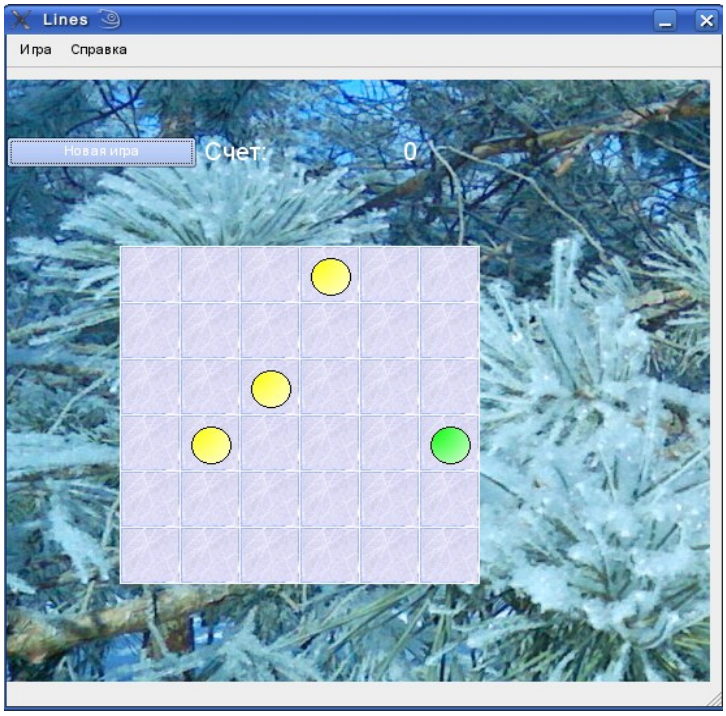


Рис.2. Совершение хода.

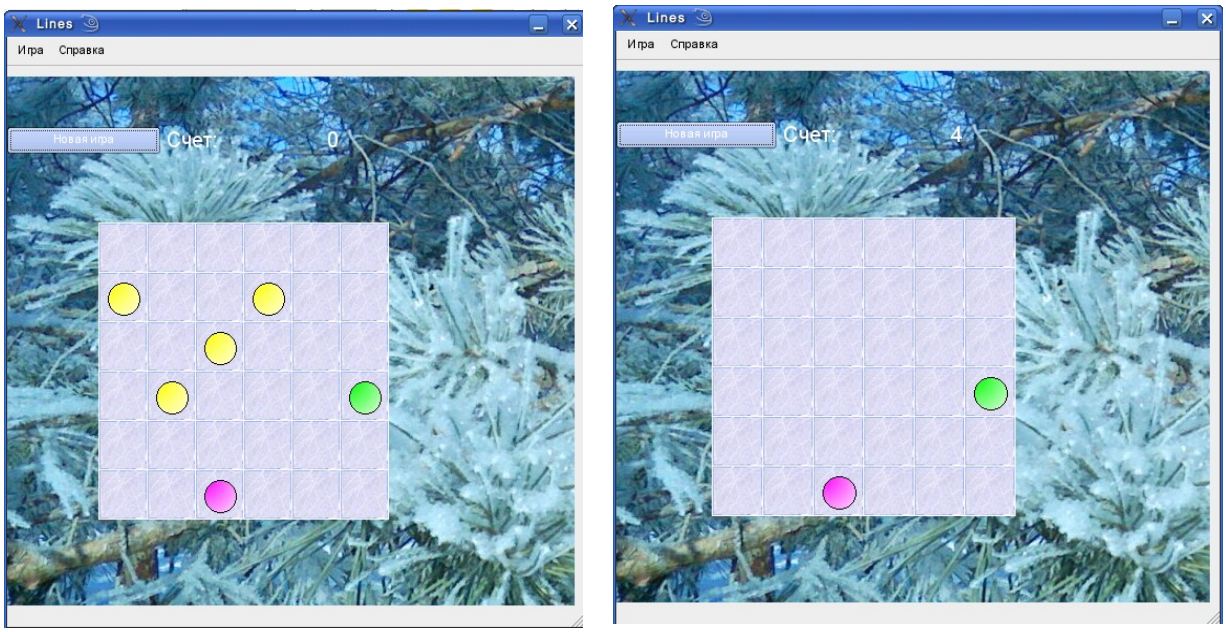


Рис.3. Изменение счета игры.

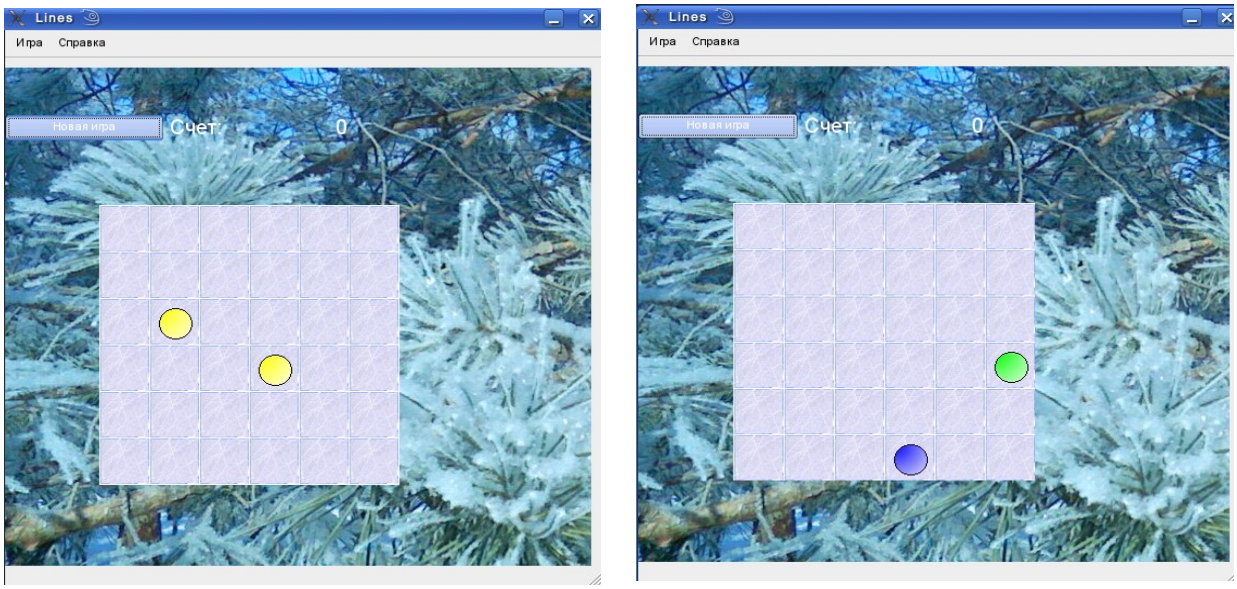


Рис.4. Отображение на поле случайных шариков.

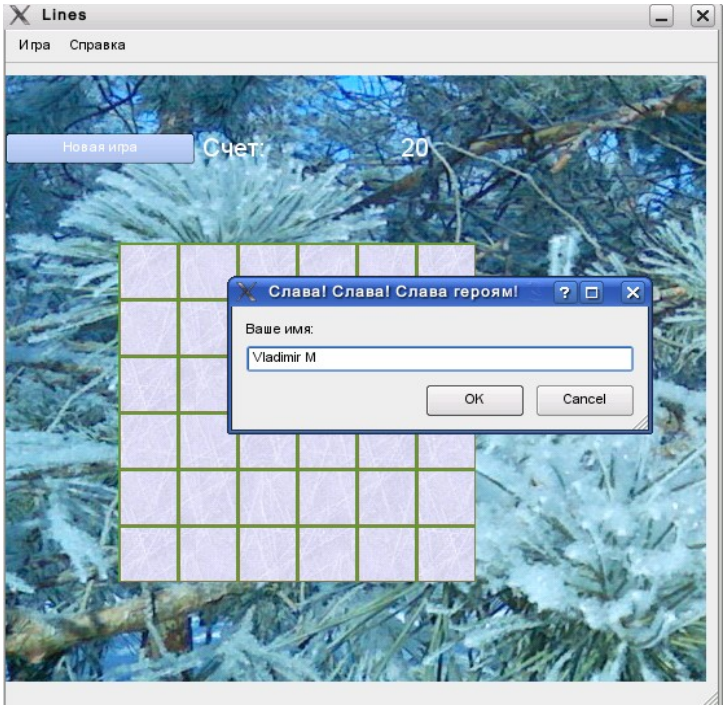


Рис.5. Ввод имени игрока.

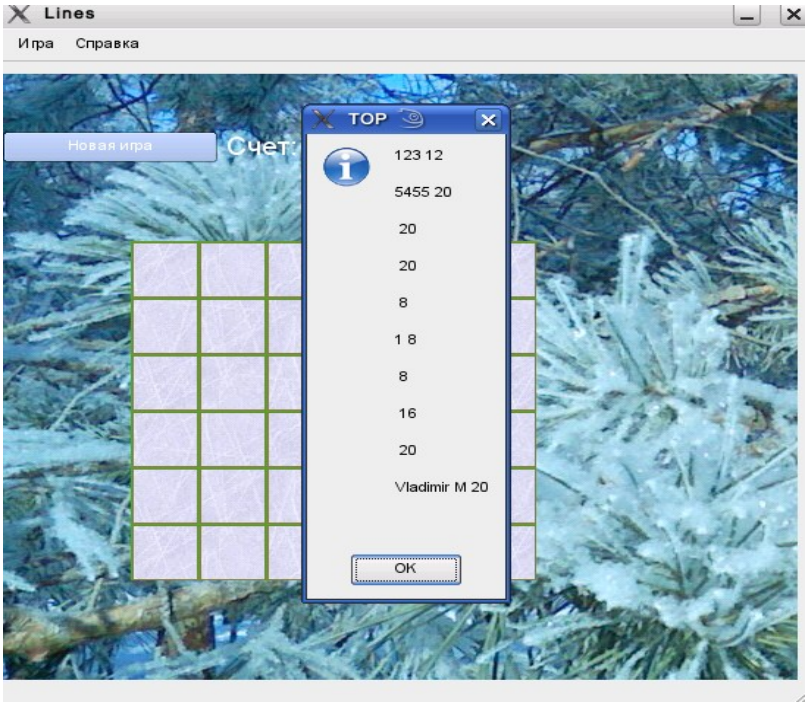


Рис.6. Отображение списка лучших игроков

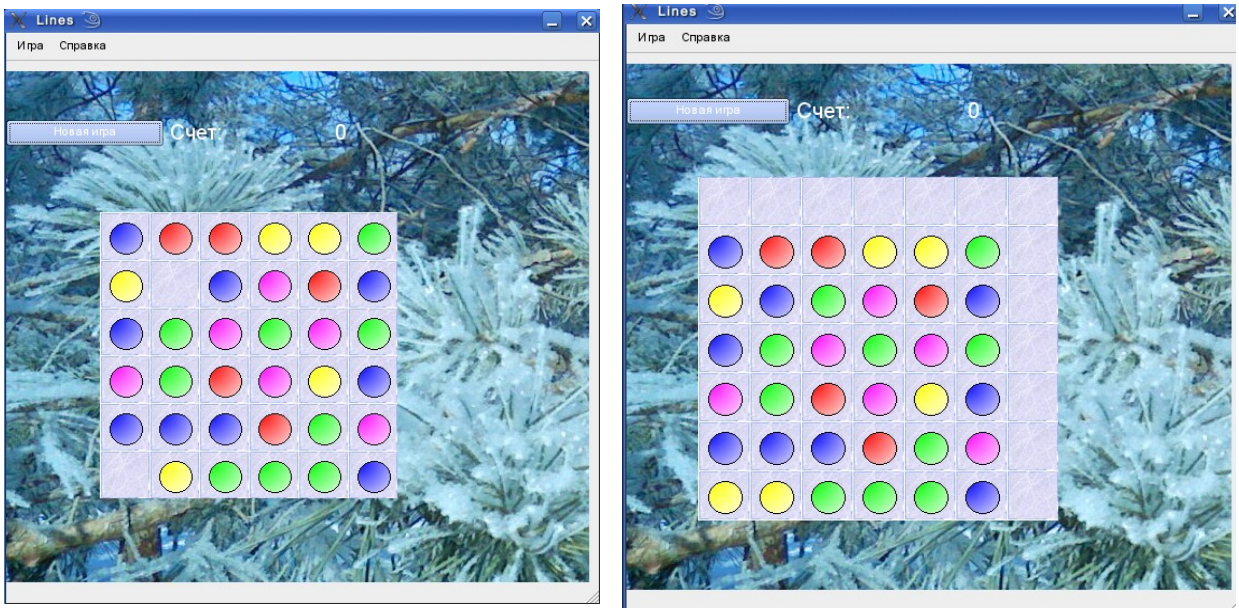


Рис.7. Изменение размеров поля.

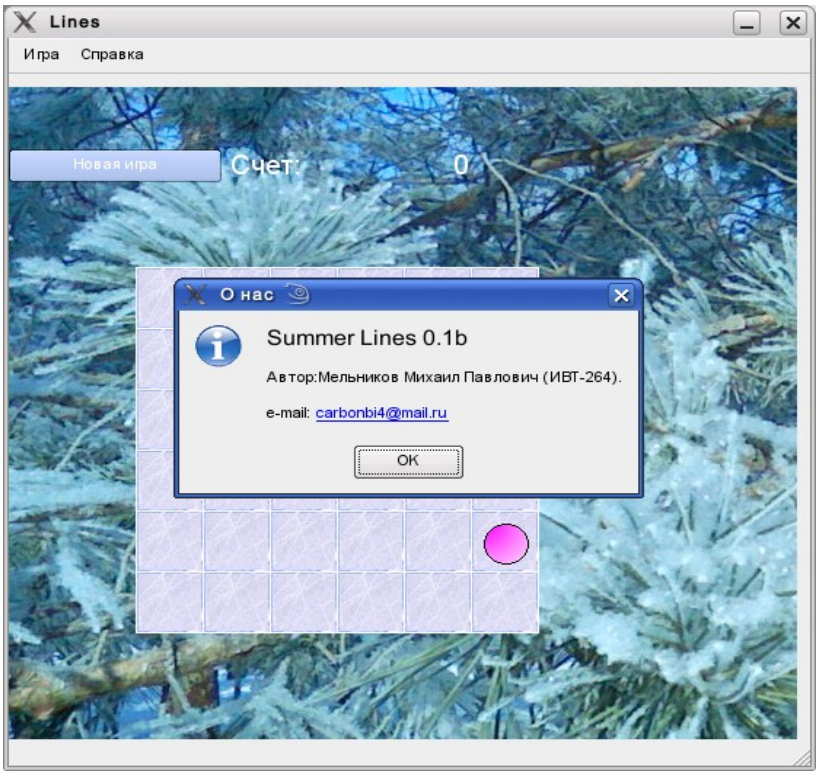


Рис.8. Отображение информации о программе.

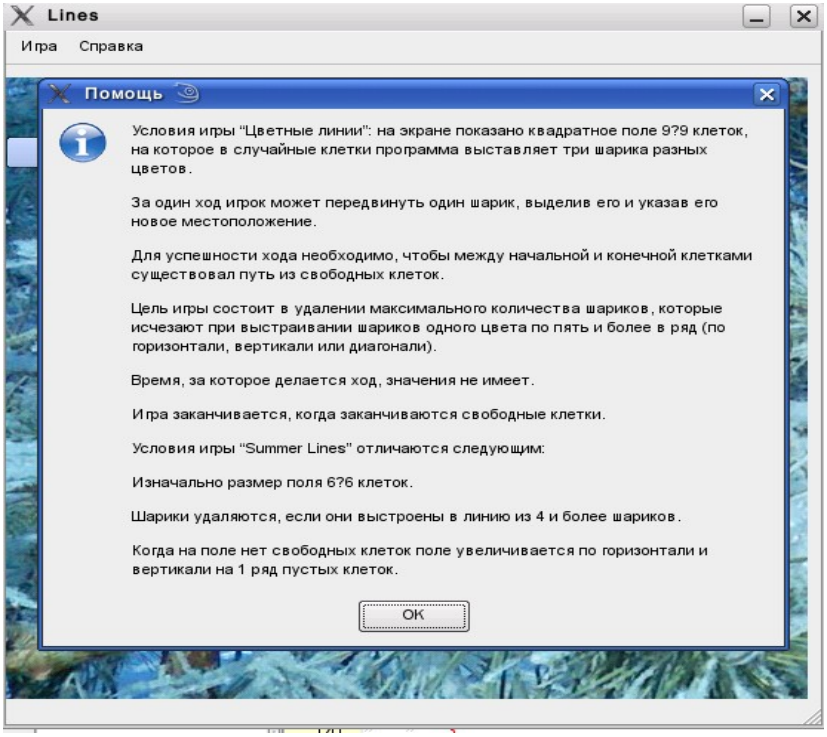


Рис.9. Отображение правил игры.

Структура данных

Класс, представляющий собой окно игры.

`class Lines : public QMainWindow`

Поле	Тип	Описание
`board`	указатель на объект класса «Доска для линий» (**`*linesBoard`**)	Переменная хранит указатель на объект класса, которые отвечает за графическое отображение игрового поля.

Класс, представляющий собой графическое отображение игрового поля.

`class linesBoard : public Qlabel`

Поле	Тип	Описание
`backgroundPixmap`	Указатель на объект класса «Карта изображения» (**`*QPixmap`**)	Хранит указатель на переменную, в которой хранится текущее изображение фона поля и шариков.
`mathLinesBoard`	Объект класса «Математическая модель поля» (**`MathBoard`**)	Хранит текущее положение шариков на поле, состояние и правила игры.
`path`	Очередь строк (**`QQueue<QString>`**)	Хранит путь, который проходит шарик при текущем поле в формате, установленном внутри класса.
`queueForPauseSet`	Очередь строк, (**`QQueue<QString>`**)	Хранит список координат и цветов шариков, которые требуется установить, по завершению перемещения шарика.
`queueForPauseDel`	Очередь строк, (**`QQueue<QString>`**)	Хранит список координат и цветов шариков, которые требуется удалить, по завершению перемещения шарика.
`boardIsCreated`	Булевая перменная (**`bool`**)	Логическая переменная, хранящая состояние выделения памяти для изображения.
`timer`	Указатель на объект класса «Таймер» (**`QTimer`**)	Хранит указатель на объект, вызывающий через равные промежутки времени функцию перерисовки шарика при анимации движения.
`randomTimer`	Указатель на объект класса «Таймер» (**`QTimer`**)	Хранит указатель на объект, вызывающий через равные промежутки времени функцию для проверки можно ли в данный момент времени рисовать шарик на поле.

Поле	Тип	Описание
randomDelTimer	Указатель на объект класса «Таймер» (**QTimer**)	Хранит указатель на объект, вызывающий через равные промежутки времени функцию для проверки можно ли в данный момент времени удалять шарик с поля.

Класс, представляющий собой математическое отображение игрового поля.
class MathBoard : public QObject

Поле	Тип	Описание
boardArr	Двумерный массив, целых чисел (**int[END_SIZE][END_SIZE]**)	Хранит массив с числами, соответствующим шарикам и пустым полям.
copyArr	Двумерный массив, целых чисел (**int[END_SIZE][END_SIZE]**)	Хранит массив с числами, соответствующим шарикам и пустым полям и отметки для прохождения пути шариком.
prevX	Целое число (**int**)	Горизонтальная координата предыдущей выделенной клетки
prevY	Целое число (**int**)	Вертикальная координата предыдущей выделенной клетки
curSize	Целое число (**int**)	Хранит текущую размерность игрового поля.
freeCells	Целое число (**int**)	Хранит текущее количество свободных клеток
score	Целое число (**int**)	Хранит текущий счет игры.
freeCellsList	Связаный список пар целых чисел (**QList<QPair<int,int>>**)	Хранит список пустых клеток.

Константы.

Имя	Значение.
BEGIN_SIZE	6
END_SIZE	9
MOVE	1
NO_MOVE	0
DELETE_LENGTH	4
NEW_GAME_CODE	-100

Тесты

Тест №1. Передвижение шарика.

Входные данные

Переменная	Тип	Описание	Значение
mathLinesBoard.prevX	Целое число	Горизонтальная координата предыдущей выбранной клетки	0
mathLinesBoard.prevY	Целое число	Вертикальная координата предыдущей выбранной клетки	3
mathLinesBoard.boardArr	Двумерный массив целых чисел	Массив поля игры.	0,0,0,0,0,0 0,0,0,0,0,0 1,0,1,0,0,0 0,0,0,0,0,0 0,0,0,0,0,0 0,0,0,0,0,0 (индексы нарастают снизу вверх, слева направо)
mathLinesBoard.copyArr	Двумерный массив целых чисел	Копия массива поля игры	0,0,0,0,0 0,0,0,0,0 0,0,0,0,0 0,0,0,0,0 0,0,0,0,0
xCell	Целое число	Горизонтальная координата текущей выбранной клетки	4
yCell	Целое число	Вертикальная координата текущей выбранной клетки	5
curSize	Целое число	Текущая размерность поля	6

Выходные данные:

Переменная	Тип	Описание	Значение
mathLinesBoard.prevX	Целое число	Горизонтальная координата предыдущей выбранной клетки	-1
mathLinesBoard.prevY	Целое число	Вертикальная координата предыдущей выбранной клетки	-1
mathLinesBoard.boardArr	Двумерный массив целых чисел	Массив поля игры.	0,0,0,0,1,0 0,0,0,0,0,0 0,0,1,0,0,0 0,0,0,0,0,0 0,0,0,0,0,0 0,0,0,0,0,0 (индексы нарастают снизу вверх, слева направо)
mathLinesBoard.copyArr	Двумерный массив целых чисел	Копия массива поля игры	0,0,0,0,108,107 0,101,102,103,104,105,106 1,100,1,0,0,0 0,0,0,0,0,0 0,0,0,0,0,0 0,0,0,0,0,0

Тест №2. Удаление шариков.

Входные данные:

mathLinesBoard.boardArr	Двумерный массив целых чисел	Массив поля игры.	0,0,0,0,0,0 0,0,0,0,0,0 1,1,1,1,1,0 0,0,0,0,0,0 0,5,0,0,0,0 0,0,0,3,0,0 (индексы нарастают снизу вверх, слева направо)

Выходные данные:

mathLinesBoard.boardArr	Двумерный массив целых чисел	Массив поля игры.	0,0,0,0,0,0 0,0,0,0,0,0 0,0,0,0,0,0 0,0,0,0,0,0 0,5,0,0,0,0 0,0,0,3,0,0 (индексы нарастают снизу вверх, слева направо)

Функции

Класс, представляющий собой окно игры.

class Lines : public QMainWindow

Прототип	Описание
Lines(QWidget *parent = 0, Qt::WFlags flags = 0);	Конструктор по-умолчанию.
~Lines();	Деструктор.
void editScore(int);	Функция перерисовки счета игры.
void editResizeBoardWidget(int);	Функция изменения размера поля игры
void checkTopScore(int);	Функция проверки необходимости внесения имени игрока в список лучших игроков.
void showTop();	Функция отображения списка лучших игроков.
void newGameStart();	Функция подготовки окна к началу новой игры.
void aboutMe();	Функция отображения информации об авторе.
void helpGame();	Функция отображения справки.

Класс, представляющий собой графическое отображение игрового поля.

class linesBoard : public Qlabel

Прототип	Описание
linesBoard(QWidget *parent);	Конструктор по-умолчанию.
~linesBoard();	Деструктор.
void newGameClicked();	Функция перерисовки поля для новой игры
void scoreChanged (int newScore);	Сигнализирует о изменении счета.
void newBoardSize(int newSizePixels);	Сигнализирует об изменении размера доски.
void endGame(int);	Сигнализирует о наступлении завершения игры.

void doStep();	Функция прорисовки хода.
void setBallAfterPause();	Функция установки шарика после паузы.
void delBallAfterPause();	Функция удаления шарика после паузы.
void setBall(QColor ballColor, int xCell, int yCell);	Функция установки шарика на графическое поле
void deleteBall(int xCell, int yCell);	Функция удаления шарика с графического поля
void resize (int newSize);	Функция изменения размера графического поля
void continueGame(int xCell, int yCell);	Функция, получения команд от класса математической модели поля и их выполнения.
void virtual mousePressEvent (QMouseEvent * mouseCondition);	Функция обработки щелчка по игровому полю.

Класс, представляющий собой математическое отображение игрового поля.
`class MathBoard : public QObject`

Прототип	Описание
MathBoard(QObject *parent);	Конструктор.
MathBoard();	Конструктор по-умолчанию.
~MathBoard();	Деструктор.
int getScore();	Функция, возвращающая счет игры.
QString run(int xCell, int yCell);	Функция, выбирающая действия для полученных координат.
int move(int xCell, int yCell, int *xDelCell, int *yDelCell);	Функция перемещения шарика.
bool moveEnabled (int xCell, int yCell, int curX, int curY, int waveNumber);	Функция проверки возможности хода.
void setMathBall(int color, int xCell, int yCell);	Функция установки шарика.
bool deleteNecessary();	Функция проверки необходимости удаления.

int getCurSize ();	Функция, возвращающая текущий размер поля.
bool isCellForDelete(int i, int j);	Функция проверки необходимости удаления шарика.
void addScore(int addPoints);	Функция увеличения счета.
bool deleteLineSearch(int iFirstCell,int jFirstCell);	Функция удаления линий из шариков с поля.
void prepareLineDelete(int iFirstCell, int jFirstCell, int iLast, int jLast, int direction);	Функция подготовки поля к удалению линий.
void prepareCellDelete(int i, int j);	Функция подготовки шарика к удалению линий.
QStack<int> findPath(int beginX, int beginY, int endX, int endY);	Функция поиска пути в массиве поля.
bool isNextCoordsNearer(int curX, int curY, int nextX, int nextY);	Функция проверки ближе ли следующие координаты к цели.
bool setRandomBall(int *xRandom, int *yRandom);	Функция установки случайного шарика на поле.
QString color(int xCell, int yCell)	Функция, возвращающая цвет шарика по координате.
void commandDeleteGraphicBall (int xCellDelete, int yCellDelete, QString &commandString);	Команда удаления графического отображения шарика.
void commandSetGraphicBall (int xCellAdd, int yCellAdd, QString &commandString);	Команда установки графического отображения шарика.
void commandSetGraphicScore(QString &commandString);	Команда установки графического отображения счета.
void commandGraphicMoving (QStack<int> path, QString curColor, QString &commandString);	Команда графического перемещения шарика
void commandGraphicResize(QString &commandString);	Команда графического изменения размера поля.
void commandGameGraphicEnd(QString &commandString);	Команда графического отображения процесса завершения игры.
void commandGameGraphicBegin(QString &commandString);	Команда графического отображения начала новой игры.

Алгоритмы

Алгоритм функции “Получение команд от класса математической модели поля и их выполнения”(continueGame) класса графического отображение игрового поля (linesBoard).

Список переменных членов класса , используемых в алгоритме:

- timer;
- queueForPauseSet;
- queueForPauseDel;
- randomDelTimer;
- path;

Список локальных переменных, используемых в алгоритме:

Переменная	Тип	Описание
commandString	Строка (**QString**)	Строка с командами.
commandList	Связанный список строк **(QStringList)**	Разбитая на отдельные слова последовательность команд.
i	Итератор (**QStringList::const_iterator i**)	Итератор для перебора членов списка **commandList.**
colorName	Строка (**QString**)	Имя цвета.
cellColor	Класс цвета (**QColor**)	Цвет шарика.
newSize	Целое число (**int**)	Новый размер.
newScore	Целое число (**int**)	Новый счет.
finalScore	Целое число (**int**)	Финальный счет.
xCell	Целое число (**int**)	Горизонтальная координата клетки.
yCell	Целое число (**int**)	Вертикальная координата клетки.

1. Получение команд от класса математической модели поля и их выполнения

1.1. Вызов функции совершения хода класса математической модели (**MathBoard::run**) с координатами **xCell** и **yCell** и сохранение полученного значения в переменной **commandString.**

1.2. Разбиение строки **commandString** на отдельные слова и сохранение в виде списка **commandList**.

1.3. Пока итератор (**i**) не достиг конца списка **commandList**.

1.3.1. Если считанная строка = «SET»

1.3.1.1. Считывание из списка команд координат и цвета.

1.3.1.2. Если таймер происходящего перемещения (**timer**) не активен установка шарика.

1.3.1.3. Если таймер происходящего перемещения активен, помещение в

очередь **queueForPauseSet** полученных координат и цвета и запуск таймера **randomTimer**

1.3.2. Если считанная строка = «DEL»

1.3.2.1. Считывание из списка команд координат.

1.3.2.2.Если таймер происходящего перемещения (**timer**) не активен удаление шарика.

1.3.2.3.Если таймер происходящего перемещения (**timer**) активен помещение координат в очередь **queueForPauseDel** и запуск таймера **randomDelTimer**.

1.3.3. Если считанная строка = «NEW_SIZE» выполнение изменения размера поля.

1.3.4. Если считанная строка = «SCORE_CHANGE» выполнение перерисовки счета.

1.3.5. Если считанная строка = «MOVE» сохранение в очереди **path** последовательности ходов.

1.3.6. Если считанная строка = «START_MOVING» запуск таймера **timer**.

1.3.7. Если считанная строка = «END_GAME» выполнение завершения игры.

1.3.8. Увеличение итератора **i**.

Алгоритм функции “Функции, выбирающей действия для полученных координат.”(run) класса математического отображение игрового поля (MathBoard).

Список переменных членов класса , используемых в алгоритме:

- curSize;
- freeCells

Список локальных переменных, используемых в алгоритме:

Переменная	Тип	Описание
commandLine	Строка (**QString**)	Строка, в которую записывается последовательность команд.
xDelCell	Указатель на целое число (**int***)	Координата удаляемой клетки.
yDelCell	Указатель на целое число (**int***)	Координата удаляемой клетки.
path	Стек целых чисел (**QStack<int>**)	Последовательность координат для перемещения.
curColor	Строка (**QString**)	Текущий цвет.
deleted	Целое число (**int**)	Количество удаляемых шариков.
xNew	Указатель на целое число (**int***)	Координата устанавливаемого шарика.
yNew	Указатель на целое число (**int***)	Координата устанавливаемого шарика.
xCell	Целое число (**int**)	Полученная координата выбранной клетки.

Переменная	Тип	Описание
yCell	Целое число (**int**)	Полученная координата выбранной клетки.

1. Выбор действия для полученных координат
    1.1. Если полученная координата (**xCell**) = коду начала новой игры.
        1.1.1. Вызов функции, записывающей команду начала игры в **commandLine** для графического отображения.
    1.2. Если полученная координата (**xCell**) не равна коду начала новой игры.
        1.2.1. Если совершается пермещение шарика
            1.2.1.1. Вызов функции, записывающей команду перемещения в **commandLine** для графического отображения.
            1.2.1.2. Если удаление необходимо выполнение удаления и запись команды для графического отображения.
            1.2.1.3 Если удаление не нужно, выбрасывание случайных шариков и запись команды для графического отображения.
            1.2.1.4. Если свободные клетки (**freeCells**) закончились и поле не достигло максимального размера, расширение поля и запись команды для графического отображения.
            1.2.1.5. Если свободные клетки (**freeCells**) закончились и поле достигло максимального размера, завершение игры и запись команды для графического отображения.
    1.3. Вернуть строку с командами **commandLine.**

Алгоритм функции “Перемещения шарика”(move) класса математического отображение игрового поля (MathBoard).

Список переменных членов класса , используемых в алгоритме:

- boardArr;
- copyArr;

Список локальных переменных, используемых в алгоритме:

Переменная	Тип	Описание
xCell	Целое число (**int**)	Текущая горизонтальная координата.
yCell	Целое число (**int**)	Текущая вертикальная координата.
xDelCell	Целое число (**int**)	Предыдущая горизонтальная координата.
yDelCell	Целое число (**int**)	Предыдущая вертикальная координата.

1. Перемещения шарика
    1.1. Копирование массива **boardArr** в **copyArr**.
    1.2. Если пермещение возможно.
        1.2.1.Установка шарика на новой позиции.
        1.2.2. Удаление шарика со старой позиции.
        1.2.3. Возвращение кода успешного совершения хода.

1.3. Если пермещение невозможно.

1.3.1. Копирование текущих координат (**`xCell, yCell`**) на местро предыдущей (**`prevX, prevY`**).

1.3.2. Возвращение кода неудачного пермещения.

Алгоритм функции “Проверка возможности пермещения”(moveEnabled) класса математического отображение игрового поля (MathBoard).

Список переменных членов класса , используемых в алгоритме:

- boardArr;
- copyArr;

Список локальных переменных, используемых в алгоритме:

Переменная	Тип	Описание
`waveNumber`	Целое число (**`int`**)	Номер рекурсивного запуска.
`xCell`	Целое число (**`int`**)	Текущая горизонтальная координата.
`yCell`	Целое число (**`int`**)	Текущая вертикальная координата.
`previousX`	Целое число (**`int`**)	Предыдущая горизонтальная координата.
`previousY`	Целое число (**`int`**)	Предыдущая вертикальная координата.

1. Проверка возможности пермещения

1.1. Если координат текущей клетки равны предыдущим и рекурсивный запуск функции не первый (**`waveNumber!=0`**), вернуть "Истина".

1.2. Если предыдущего шарика выбрано не было, вернуть "Ложь".

1.3. Если координат текущей клетки равны предыдущим и рекурсивный запуск функции первый (**`waveNumber == 0`**), вернуть "Ложь".

1.4. Если какие-то из пар координат выходят за пределы поля, вернуть ложь.

1.5. Если все координаты не выходят за пределы поля, клетка занята и это не первый рекурсивный запуск функции, вернуть "Ложь"

1.6. Увеличение счетсика рекурсивного запуска функции.

1.7. Выставление пометки в предыдущей клетке (**`copyArr[previousX] [previousY] = 100+waveNumber`**).

1.8. Если рекурсивнй запуск функции для соседних с текцщей клеток возвращает "Истина", вернуть "Истина".

1.9. Если рекурсивнй запуск функции для соседних с текцщей клеток возвращает "Ложь", вернуть "Ложь".

Алгоритм функции “Проверка поля на удаление линий”(deleteNecessary) класса математического отображение игрового поля (MathBoard).

Список переменных членов класса , используемых в алгоритме:

- curSize;
- boardArr;

Список локальных переменных, используемых в алгоритме:

Переменная	Тип	Описание
deleteDone	Булевая (**bool**)	Необходимо ли удаление.

1. Проверка поля на удаление линий
    1.1. Для каждой клетки
        1.1.1. Если клетка не пустая.
            1.1.1.1.Если клетка входит в какую-то удаляемую линию deleteDone = "Истина"
    1.2. Вернуть deleteDone.

Алгоритм функции “Поиска линий для удаления”(deleteLineSearch) класса математического отображение игрового поля (MathBoard).

Список переменных членов класса , используемых в алгоритме:

- boardArr

Список локальных переменных, используемых в алгоритме:

Переменная	Тип	Описание
firstCell	Целое число (**int**)	Цвет первой ячейки.
iFirstCell	Целое число (**int**)	Горизонтальная координата первой ячейки.
jFirstCell	Целое число (**int**)	Вертикальная координата первой ячейки.
lineFinished	Булевая (**bool**)	Завершена ли линия.
deleteMustBe	Булевая (**bool**)	Необходимо ли удаление.
lineLength	Целое число (**int**)	Длина удаляемой линии.
derection	Целое число от 1 до 4 (**int**)	Направление.
i	Целое число (**int**)	Текущая горизонтальная координата.
j	Целое число (**int**)	Текущая вертикальная координата.

1. Поиск линий для удаления
    1.1. Для каждого направления
        1.1.1.пока линия не завершена.
            1.1.1.1. Длина линии = 0.

1.1.1.2. Линия закончена = "Ложь".

1.1.1.3. Начать поиск с полученных координат **`iFirstCell, jFirstCell`**.

1.1.1.4 Если текущие индексы вышли за пределы доски линия завершена.

1.1.1.5 Если текущие индексы не вышли за пределы доски

1.1.1.5.1. Если цвет совпадает с цветом полученных координат, изменение индексов в зависимости от текущего направления и увеличение счетчика длины линии.

1.1.1.5.2. Если цвет не совпадает с цветом полученных координат, линия завершена.

1.1.1.6. Если длина линии больше либо равна минимально удаляемое, подготовить линию к удалению, удаление должно быть = "Истина".

1.2. Вернуть должно ли быть удаление (**`deleteMustBe`**).

Возможные сообщения, которые класс linesBoard получает от класса mathBoard.

Сообщение	Описание
SET COLOR X_CELL Y_CELL	Установить шарик цвета **COLOR** в клетку с координатами **X_CELL, Y_CELL**.
DEL X_CELL Y_CELL	Удалить шарик с координатами **X_CELL, Y_CELL**.
NEW_SIZE SIZE	Установить новый размер доски равным **SIZE**.
SCORE_CHANGE NEW_SCORE	Установить счет игры равным **NEW_SCORE.**
MOVE COLOR PREV_X PREV_Y X_CELL Y_CELL	Переместить шарик цвета **COLOR** из клетки с координатами **PREV_X, PREV_Y** в клетку с координатами **X_CELL, Y_CELL.**
START_MOVING	Начать перемещение шарика.
END_GAME FINAL_SCORE	Завершить игру со счетом **FINAL_SCORE.**

Графические элементы

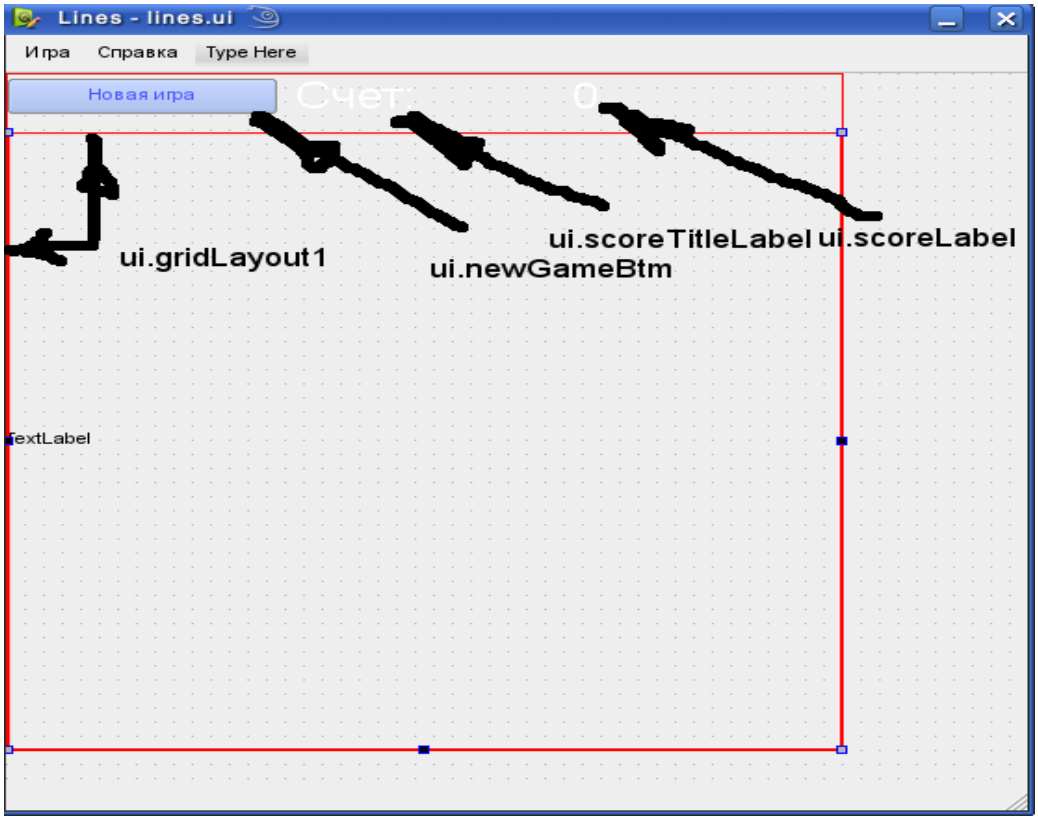


Рис.10.Графические.элементы.

Элемент	Описание	Свойства
ui.textLabel	Фоновая картинка.	
ui.gridLayout1	Пространство, на котором размещается игровое поле.	
ui.newGameBtm	Кнопка запуска новой игры.	text = «Новая игра»
ui.scoreTitleLabel	Выводит надпись «Счёт».	text = «Счёт»
ui.scoreLabel	Выводит текущий счет.	

Слоты и сигналы

Соединенны в конструкторе объекта класса Lines.

Объект отправитель	Сигнал	Объект получатель	Слот
board	**scoreChanged(int)**	**this**	**editScore(int)**
board	**newBoardSize(int)**	**this**	**editResizeBoardWidget(int)**
board	**endGame(int)**	**this**	**checkTopScore(int)**
ui.actionNewGame	**triggered()**	**this**	**newGameStart()**
ui.actionExit	**triggered()**	**this**	**close()**
ui.newGameBtm	**clicked()**	**this**	**newGameStart()**
ui.actionTop	**triggered()**	**this**	**showTop()**
ui.actionAbout	**triggered()**	**this**	**aboutMe()**
ui.actionHelp	**triggered()**	**this**	**helpGame()**

Соединенны в конструкторе объекта класса linesBoard.

Объект отправитель	Сигнал	Объект получатель	Слот
timer	**timeout()**	**this**	**doStep()**
randomTimer	**timeout()**	**this**	**setBallAfterPause()**
randomDelTimer	**timeout()**	**this**	**delBallAfterPause()**