МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО»

ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

**ТЕМА КУРСОВОЙ РАБОТЫ**

Курсовая работа

по дисциплине «Программирование»

студента 1 курса группы ПИ-111

Шадловской Вероники Юрьевны

направления подготовки 09.03.04 «Программная инженерия»

|  |  |  |
| --- | --- | --- |
| Научный руководитель  ассистент кафедры компьютерной инженерии и моделирования | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | Фамилия И.О. |

Симферополь, 2018

**РЕФЕРАТ**

Целью курсовой работы является реализация игры «Линии». На поле 9х9 появляются по три шара случайного цвета на свободных клетках со случайными координатами. Нужно перемещать шары так, чтобы выстроить линию шаров одинакового цвета. Если выстраивается линия из пяти и более шаров одинакового цвета, то вся линия удаляется. Игра заканчивается, когда на поле не остаётся свободного места. Цель игры набрать как можно больше очков, удаляя линии.

**О**ГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 4](#_Toc475371222)

[ГЛАВА 1 АНАЛИЗ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ 5](#_Toc475371223)

[ГЛАВА 2 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ 6](#_Toc475371224)

[2.1 Постановка задачи 6](#_Toc475371225)

[2.2 Описание алгоритмов 6](#_Toc475371226)

[2.3 Описание структур данных 6](#_Toc475371227)

[2.4 Описание основных функций 6](#_Toc475371228)

[ГЛАВА 3 ТЕСТИРОВАНИЕ ПРОГРАММЫ 7](#_Toc475371229)

[ЗАКЛЮЧЕНИЕ 8](#_Toc475371230)

[ЛИТЕРАТУРА 9](#_Toc475371231)

ВВЕДЕНИЕ

В курсовой работе должна быть реализована игра «Линии». Для создания данной игры необходимо разработать графический интерфейс (поле 9х9 и табло, на котором выводится счёт) и решить следующие задачи:

1. Перемещение шара с помощью мыши.
2. Автоматическое выставление на три пустые клетки поля трёх шариков случайных цветов после того, как пользователь сделал ход, и если после этого хода не была удалена линия.
3. Поиск линий по диагонали, вертикали и горизонтали и их удаление. Минимальная длина линии-5 шаров.
4. Проверка пути шара к выбранному месту. Шар может перемещаться только при условии, что существует свободный путь от его исходного положения к выбранным координатам.
5. Протестировать программу.

ГЛАВА 1  
АНАЛИЗ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

Программа разрабатывалась в Visual Studio 2015 Community. это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio.  Среда позволяет разрабатывать программы с использованием следующих языков программирования: C++, C#, Visual Basic, F#, JavaScript, TypeScript, Python.

Для разработки графической составляющей использовалась свободная кроссплатформенная мультимедийная библиотека SFML(Simple and Fast Multimedia Library). SFML содержит ряд модулей для простого программирования игр и мультимедиа приложений:

System — управление временем и потоками, он является обязательным, так как все модули зависят от него.

Window — управление окнами и взаимодействием с пользователем.

Graphics — делает простым отображение графических примитивов и изображений, для своей работы требует модуль Window.

Audio — предоставляет интерфейс для управления звуком.

Network — для сетевых приложений.

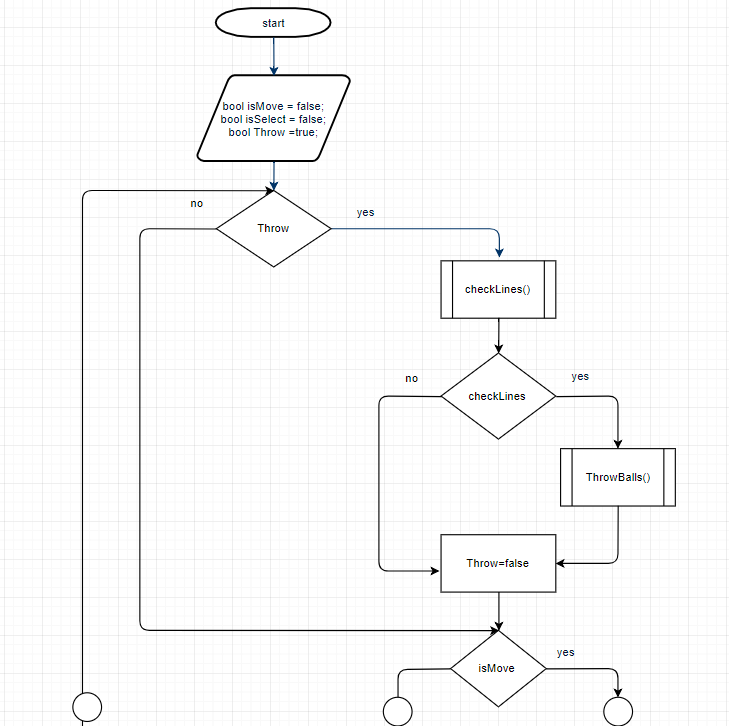
ГЛАВА 2  
ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

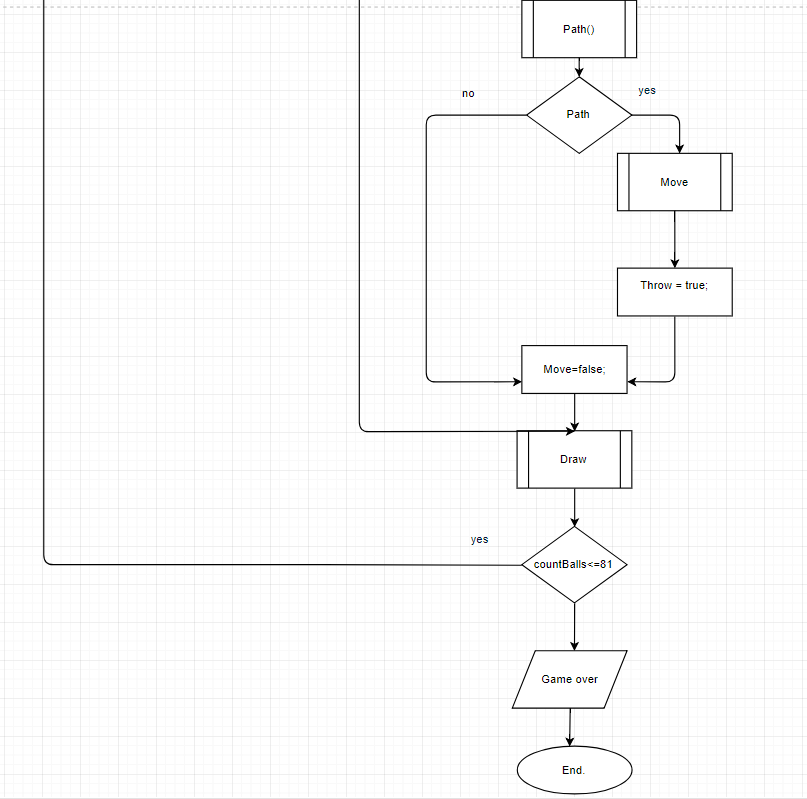
## 2.1 Постановка задачи

Целью курсовой работы является создание программы, реализующую игру «Линии». На поле 9x9 позиций строятся линии из одноцветных шаров. После каждого хода игрока происходит проверка, есть ли на поле линии из пяти и более шаров одинакового цвета. Если такие линии найдены, то шары этой линии удаляются, и новые шары на поле не появляются. Если линии не были найдены, тона поле случайным образом появляются три разноцветных шара. Игрок перемещает шар по полю с помощью мыши. Игра заканчивается, если на поле не остается свободных позиций для вывода шаров. На верхнем табло выводится счёт.

## 2.2 Описание алгоритмов

Для решения поставленной задачи воспользуемся алгоритмом, представленным на рисунке.





Изначально булевым переменным isMove(значение переменной зависит от возможности переместить шарик) и isSelect(значение переменной зависит от того выбран ли шарик или нет) присваивается значение false, а булевой переменной Throw (переменная принимает значение в зависимости от того должны ли появится новые шарики на поле) присваивается значение true. Пока количество шариков на поле меньше 81, выполняется основной цикл:

Если переменная Throw верна, то вызывается функция checkLines, если во время её работы не было найдено и удалено линий, то на поле генерируются три случайных шара. Затем переменно Throw присваивается значение false.

Если был выбран шар и пустая клетка для его перемещения, вызывается функция pathSearch, в зависимости от её значение шар перемещается или остаётся на месте. Переменной isMove присваивается значение false.

Происходит отрисовка. Если на поле меньше 81, то есть есть пустые клетки, то цикл продолжается.

## 2.3 Описание структур данных

## Чтобы задавать положение шаров на поле и их цвет используется массив структур. Структура — тип данных: запись, состоящая из нескольких переменных (или констант) разного типа.

struct Ball

{ bool isFree = true;

Color color;

Sprite ballSprite;

} ball[9][9];

## Структура Ball объединяет в себе три переменные. Булева переменная isFree принимает значения в зависимости от того, свободна ли клетка поля с координатами структуры ball[9][9]. Переменная color типа Сolor содержит цвет шара. Переменная ballSprite хранит спрайт шара.

## 2.4 Описание основных функций

1. Функция кидает в случайном порядке 3 шара на поле.

void ThrowBalls(Texture &ballTexture)

{

Color color[7] = { sf::Color::Red,sf::Color::Blue,sf::Color::Yellow, sf::Color::Cyan, sf::Color::White, sf::Color::Green, sf::Color::Magenta };

int count = 0;

srand(time(0));

do

{

if (countBalls == 81) return;

int x = (rand() % 9);

int y = (rand()% 9);

if (ball[x][y].isFree)

{

ball[x][y].ballSprite.setTexture(ballTexture);

ball[x][y].color = color[rand() %7];

ball[x][y].ballSprite.setColor(ball[x][y].color);

ball[x][y].ballSprite.setPosition(x\*80, y\*80+80);

ball[x][y].isFree =false;

count++;

countBalls++;

}

} while (count != 3);

}

С помощью генератора случайных чисел выбираются координаты клетки, в которую будет помещён шар. Если клетка с данными координатами не занята, то с помощью генератора случайных чисел и массива типа Color выбирается цвет. Действия повторяются, пока на поле не будут помещены три новых шара.

2. Функция проверки свободного пути.

bool pathSearch(int startX, int startY, int endX, int endY)

{

int map[9][9];

for (int x = 0;x < 9;x++)

{

for (int y = 0;y < 9;y++)

{

if (ball[x][y].isFree) map[x][y] = -2;

else map[x][y] = -1;

}

}

map[startX][startY] = 0;

map[endX][endY] = -2;

int dx[4] = { 1, 0, -1, 0 };

int dy[4] = { 0, 1, 0, -1 };

int step = 0;

int x, y, k;

bool stop;

bool added;

do {

stop = true;

for (y = 0; y < 9; ++y)

for (x = 0; x < 9; ++x)

if (map[x][y] == step)

{

for (k = 0; k < 4; ++k)

{

int iy = y + dy[k],

ix = x + dx[k];

if (iy >= 0 && iy < 9 && ix >= 0 && ix < 9 &&

map[ix][iy] == -2)

{

stop = false;

map[ix][iy] = step + 1;

}

}

}

step++;

} while (!stop&& map[endX][endY] == -2);

if (map[endX][endY]==-2) return false;

else return true;

}

Для проверки свободен ли путь шара, до выбранной клетки реализован алгоритм волновой трассировки (алгоритм Ли).

Строится образ множества ячеек обрабатываемого поля, каждой ячейке приписываются атрибуты проходимости/непроходимости, запоминаются стартовая и финишная ячейки.

Далее, от стартовой ячейки порождается шаг в соседнюю ячейку, при этом проверяется, проходима ли она, и не принадлежит ли ранее меченной в пути ячейке. Соседними ячейками считаются только 4 ячейки по вертикали и горизонтали.

При выполнении условий проходимости и непринадлежности её к ранее помеченным в пути ячейкам, в атрибут ячейки записывается число, равное количеству шагов от стартовой ячейки, от стартовой ячейки на первом шаге это будет 1. Каждая ячейка, меченая числом шагов от стартовой ячейки становится стартовой и из неё порождаются очередные шаги в соседние ячейки. Если по завершению цикла в клетке с конечными координатами остаётся записано исходное число, то путь не найден, в остальных случаях шарик может быть перемещён.

3. Поиск и удаление линий

В представленных ниже функциях массив ball[9][9] просматривается по горизонтали, вертикали и по диагонали слева направо и справа налево. Если была найдена и удалена линия, то функции возвращают отрицательное значение.

Поиск и удаление линий по горизонтали.

bool checkLines1()

{

bool DeletedLines = true;

for (int y = 0;y < 9;y++)

{

Color color = Color::Black;

int count = 0;

int left = 0;

for (int x = 0;x < 9;x++)

{

if ((color == ball[x][y].color) && (!ball[x][y].isFree))

count++;

if ((color != ball[x][y].color) && (!ball[x][y].isFree) || (ball[x][y].isFree))

{

if (count >= 5)

{

for (int i = left;i < x;i++)ball[i][y].isFree = true;

DeletedLines = false;

score += count;

countBalls -= count;

}

if (ball[x][y].isFree)

{

count = 0;

color = Color::Black;

}

else

{

left = x;

count = 1;

color = ball[x][y].color;

}

}

}

if (count >= 5)

{

for (int i = left;i <= 8;i++)ball[i][y].isFree = true;

DeletedLines = false;

score += count;

countBalls -= count;

}

}

return DeletedLines;

}

Поиск и удаление линий по вертикали

bool checkLines2()

{

bool DeletedLines = true;

for (int x = 0;x < 9;x++)

{

Color color = Color::Black;

int count = 0;

int top = 0;

for (int y = 0;y < 9;y++)

{

if ((color == ball[x][y].color) && (!ball[x][y].isFree))

count++;

if ((color != ball[x][y].color) && !ball[x][y].isFree || (ball[x][y].isFree))

{

if (count >= 5)

{

for (int i = top;i < y;i++)ball[x][i].isFree = true;

DeletedLines = false;

score += count;

countBalls -= count;

}

if (ball[x][y].isFree)

{

count = 0;

color = Color::Black;

}

else

{

top = y;

count = 1;

color = ball[x][y].color;

}

}

}

if (count >= 5)

{

for (int i = top;i <= 8;i++) ball[x][i].isFree = true;

DeletedLines = false;

score += count;

countBalls -= count;

}

}

return DeletedLines;

}

Поиск и удаление линий по диагонали слева направо над главной диагональю

bool checkLines3()

{

bool DeletedLines = true;

for (int x = 0;x <= 5;x++)

{

int i = x;

int k = 0;

int count = 0;

Color color = Color::Black;

int topI = x;

int topK = 0;

do {

if (!ball[i][k].isFree && (ball[i][k].color == color))

count++;

if ((!ball[i][k].isFree && (ball[i][k].color != color)) || (ball[i][k].isFree))

{

if (count >= 5)

{

for (;((topI < 9) && (topK < 9));((topI++) && (topK++))) ball[topI][topK].isFree = true;

DeletedLines = false;

score += count;

countBalls -= count;

}

if (ball[i][k].isFree)

{

count = 0;

color = Color::Black;

}

else

{

color = ball[i][k].color;

count = 1;

topI = i;

topK = k;

}

}

i++;

k++;

} while ((i < 9) && (k < 9));

if (count >= 5)

{

for (;((topI < 9) && (topK < 9));((topI++) && (topK++))) ball[topI][topK].isFree = true;

DeletedLines = false;

score += count;

countBalls -= count;

}

}

return DeletedLines;

}

Поиск и удаление линий по диагонали слева направо под главной диагональю

bool checkLines4()

{

bool DeletedLines = true;

for (int y = 0;y <= 5;y++)

{

int i = 0;

int k = y;

int count = 0;

Color color = Color::Black;

int topI = 0;

int topK = y;

do {

if (!ball[i][k].isFree && (ball[i][k].color == color))

count++;

if ((!ball[i][k].isFree && (ball[i][k].color != color)) || (ball[i][k].isFree))

{

if (count >= 5)

{

do {

ball[topI][topK].isFree = true;

DeletedLines = false;

topI++;

topK++;

} while ((topI < i) && (topK < k));

score += count;

countBalls -= count;

}

if (ball[i][k].isFree)

{

count = 0;

color = Color::Black;

}

else

{

color = ball[i][k].color;

count = 1;

topI = i;

topK = k;

}

}

i++;

k++;

} while ((i < 9) && (k < 9));

if (count >= 5)

{

do {

ball[topI][topK].isFree = true;

DeletedLines = false;

topI++;

topK++;

} while ((topI < i) && (topK < k));

score += count;

countBalls -= count;

}

}

return DeletedLines;

}

Поиск и удаление линий по диагонали справа налево над побочной диагональю

bool checkLines5()

{

bool DeletedLines = true;

for (int x = 4;x <9;x++)

{

int i = x;

int k = 0;

int count = 0;

Color color = Color::Black;

int topI = x;

int topK = 0;

do {

if ((!ball[i][k].isFree) && (ball[i][k].color == color))

count++;

if ((!ball[i][k].isFree && (ball[i][k].color != color)) || (ball[i][k].isFree))

{

if (count >= 5)

{

do {

ball[topI][topK].isFree = true;

topI--;

topK++;

} while (topI >= i && topK < k);

DeletedLines = false;

score += count;

countBalls -= count;

}

if (ball[i][k].isFree)

{

count = 0;

color = Color::Black;

}

else

{

color = ball[i][k].color;

count = 1;

topI = i;

topK = k;

}

}

i--;

k++;

} while ((i >=0) && (k < 9));

if (count >= 5)

{

do {

ball[topI][topK].isFree = true;

topI--;

topK++;

} while (topI >= 0 && topK < 9);

DeletedLines = false;

score += count;

countBalls -= count;

}

}

return DeletedLines;

}

Поиск и удаление линий по диагонали справа налево под побочной диагональю

bool checkLines6()

{

bool DeletedLines = true;

for (int y = 1;y <5;y++)

{

int i = 8;

int k = y;

int count = 0;

Color color = Color::Black;

int topI = 8;

int topK = y;

do {

if ((!ball[i][k].isFree) && (ball[i][k].color == color))

count++;

if ((!ball[i][k].isFree && (ball[i][k].color != color)) || (ball[i][k].isFree))

{

if (count >= 5)

{

do {

ball[topI][topK].isFree = true;

topI--;

topK++;

} while (topI >= i && topK < k);

DeletedLines = false;

score += count;

countBalls -= count;

}

if (ball[i][k].isFree)

{

count = 0;

color = Color::Black;

}

else

{

color = ball[i][k].color;

count = 1;

topI = i;

topK = k;

}

}

i--;

k++;

} while ((i >= 0) && (k < 9));

if (count >= 5)

{

do {

ball[topI][topK].isFree = true;

topI--;

topK++;

} while (topI >= 0 && topK < 9);

DeletedLines = false;

score += count;

countBalls -= count;

}

}

return DeletedLines;

}

bool checkLines()

{

if (checkLines1() && checkLines2() && checkLines3() && checkLines4()&&checkLines5()&&checkLines6()) return true;

else return false;

}

ГЛАВА 3  
ТЕСТИРОВАНИЕ ПРОГРАММЫ

При запуске …

ЗАКЛЮЧЕНИЕ

В результате …

ЛИТЕРАТУРА

1. ГОСТ 19.002-80 Схемы алгоритмов и программ. Правила выполнения [Текст] – Введ. с 01.07. 1981 г. М.: Изд-во стандартов, 1981. – 9 с.
2. ГОСТ 19.003-80 Схемы алгоритмов и программ. Обозначение условные графические [Текст] – Введ. с 01.07. 1981 г. М.: Изд-во стандартов, 1981. – 9 с.
3. Оформление выпускной квалификационной работы на соискание квалификационного уровня «Магистр» («Бакалавр»): методические рекомендации. / сост. Бержанский В.Н., Дзедолик И.В., Полулях С.Н. – Симферополь: КФУ им. В.И.Вернадского, 2016. – 31 с.