ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра ПМиК

Курсовая работа
по дисциплине «ООП»
на тему «Арканоид»

Выполнил:
ст. гр. ИВ-823
Шиндель Э. Д.

Проверила:
Ассистент
Кафедры ПМиК
Петухова Я. В.

Новосибирск, 2019

Оглавление

1. Введение и постановка задачи

Курсовая работа нацелена на закрепление и применение навыков, полученных в течение изучения курса ООП на примере выполнения работы по созданию игры «Арканоид»

Задача: реализовать игру «Арканоид», используя технологий объектно-ориентированного программирования на языке программирования C++.

2. Суть курсовой работы

Арканоид — видеоигра для игровых автоматов, разработанная компанией Taito в 1986 году. Игра основана на играх серии Breakout фирмы Atari. Именно её название стало нарицательным для класса подобных игр.

Игрок контролирует небольшую платформу-ракетку, которую можно передвигать горизонтально от одной стенки до другой, подставляя её под шарик, предотвращая его падение вниз. Удар шарика по кирпичу приводит к разрушению кирпича. Кирпичи бывают: красными (100 очков), зелёными (200), синими (300) и серыми (не разрушаются). После того как все кирпичи на данном уровне уничтожены, кроме серых, происходит переход на следующий уровень, с новым набором кирпичей, всего три.

Цель игры: пройти все три уровня с кирпичами, набрав максимальное количество очков.

3. Технологии ООП

1. **Наследование**
   Ниже представлена иерархия классов

Абстрактный класс blocks	Абстрактный класс ball	Абстрактный класс board
Поля класса: protected: int x, y; - Координаты блока int dx, dy; - Размеры блока по x и y int vis; - Виден ли блок int points; - очки за его разрушение	Поля класса: protected: float x, y; - Координаты мяча int dx, dy; - Изменение координат int const color = 14; - Цвет мяча жёлтый int const radius = 10; - Радиус мяча = 10	Поля класса: protected: int x, y; - Координаты платформы int dx; - Координата перемещения платформы int color; - Цвет
Методы класса: public: blocks(); - Конструктор класса int get_x(int n); - Возвращает x (начала или конца) int get_y(int n); - Возвращает y (начала или конца) int get_vis(); - Возвращает переменную vis (виден блок или нет) int point(); - Возвращает количество очков за разрушение void unvis(); - Функция делает блок невидимым	Методы класса: public: ball(); - Конструктор класса void rebound_y(); - Отскок мяча по y void rebound_x(); - Отскок мяча по x int get_y(int n); - Возвращает y (середины или середина +- радиус) int get_x(int n); - Возвращает x (середины или середина +- радиус) int get_d(int n); - Возвращает dx или dy virtual void move() – Движение и отрисовка мяча	Методы класса: public: board(); - Конструктор класса int get_y(); - Возвращает y virtual void move(); - Движение и отрисовка платформы virtual void death(); - Функция делает платформу невидимой

Класс red_block	Класс green_block
Поля класса: protected: int const color = 4; - Цвет блока красный	Поля класса: protected: int const color = 2; - Цвет блока зелёный
Методы класса: public: red_block(); - Конструктор	Методы класса: public: green_block(); - Конструктор

Класс blue_block	Класс grey_block
Поля класса: protected: int const color = 1; - Цвет блока синий	Поля класса: protected: int const color = 7; - Цвет блока серый
Методы класса: public: blue_block(); - Конструктор	Методы класса: public: grey_block(); - Конструктор

Класс normal_ball	Класс normal_board
Поля класса: protected: float const speed = 0.25; - Скорость движения мяча	Поля класса: protected: int const x1 = 100; - Размеры платформы по x int const y1 = 10; - Размеры платформы по y int const dx = 3; - Изменение координаты по x

Методы класса: public: normal_ball(); - Конструктор void move(); - Движение и отрисовка мяча void death(); - Делает мяч невидимым	Методы класса: public: normal_board(); - Конструктор void move(); - Перемещение и отрисовка платформы int get_x(int n); - Возвращает x (начала или конца) void death(); - Делает платформу невидимым

2. Инкапсуляция

В классах ball, blocks и board поля имеют модификатор доступа protected

3. Динамический полиморфизм

Указатель на базовый класс может ссылаться на любой объект, выведенный из этого базового класса.

Примером служит:

blocks *b[4][10] – Динамический массив для блоков

При вызове метода по указателю на базовый класс будет вызываться именно тот метод, который нам нужен.

point+= b[i][j]->point(); - Увеличение очков на очки разрушенного блока

4. Программная реализация

В main():

1. Создает окно

```
initwindow(1270, 660, "The_Game");
```

2. Запускает функцию меню и возвращает значение выбранного ответа

```
int choice = menu();
```

3. Запускает выбранное пользователем действие:
    a. Выход

    ```
    return 0;
    ```

    b. Запуск мануала

    ```
    manual();
    ```

    c. Запуск функции игры

    ```
    life = game(level);
    ```

В функции игры происходит:

1. Отрисовка игрового поля

```
cleardevice();
setfillstyle(1, 9);
bar(0, 0, 1270, 660);
setfillstyle(7, 15);
setbkcolor(4);
bar(74, 24, 726, 624);
setfillstyle(1, 0);
bar(100, 50, 702, 624);
setcolor(4);
line(100, 600, 702, 600);
settextstyle(0, 0, 1);
setcolor(15);
setbkcolor(0);
outtextxy(380, 602, "Death");
```

2. Создание и отрисовка блоков по уровням

```
blocks *b[4][10];
int y = 50;
for (int i = 0; i < 4; i++) {
        int x = 100;
        for (int j = 0; j < 10; j++) {
                if (level == 1) {
                        if (i == 0) b[i][j] = new blue_block(x, y);
                        if (i == 1) b[i][j] = new green_block(x, y);
                        if (i > 1) b[i][j] = new red_block(x, y);
                        level_point = 7000;
                }
                if (level == 2) {
                        if (i == 0) b[i][j] = new blue_block(x, y);
                        if ((i == 1) && ((j != 4) || (j != 5))) b[i][j] = new green_block(x, y);
                        if ((i == 1) && ((j == 4) || (j == 5))) b[i][j] = new grey_block(x, y);
```

```
                        if ((i == 2) && ((j != 4) || (j != 5))) b[i][j] = new red_block(x, y);
                        if ((i == 2) && ((j == 4) || (j == 5))) b[i][j] = new grey_block(x, y);
                        if (i == 3) b[i][j] = new red_block(x, y);
                        level_point = 13400;
                }
                if (level == 3) {
                        if ((i == 0) && (j % 2 != 0)) b[i][j] = new blue_block(x, y);
                        if ((i == 0) && (j % 2 == 0)) b[i][j] = new grey_block(x, y);
                        if ((i == 1) && (j % 2 != 0)) b[i][j] = new green_block(x, y);
                        if ((i == 1) && (j % 2 == 0)) b[i][j] = new grey_block(x, y);
                        if ((i > 1) && (j % 2 != 0)) b[i][j] = new red_block(x, y);
                        if ((i > 1) && (j % 2 == 0)) b[i][j] = new red_block(x, y);
                        level_point = 17900;
                }
                x += 60;
        }
        y += 51;
}
```

3. Создание и отрисовка мяча и платформы, а также запуск функции start(life), которая выводит на экран жизни и заканчивает работу по нажатию ‘enter’

```
normal_ball game_ball(400, 540);
normal_board game_board(350, 550);
game_board.move(0);
game_ball.move(0);
start(life);
```

4. Движение мяча и уничтожение блоков

```
death = 1;
while(death) {
        game_board.move(0);
        game_ball.move(1);
        for (int i = 0; i < 4; i++) {
                for (int j = 0; j < 10; j++) {
                        int k = 0;
                        if ((game_ball.get_y(1) == b[i][j]->get_y(1)) || (game_ball.get_y(2) == b[i][j]->get_y(0))) {
                                if ((game_ball.get_x(0) > b[i][j]->get_x(0)) && (game_ball.get_x(0) < b[i][j]->get_x(1))) {
                                        if (b[i][j]->get_vis() != 0) {
                                                if (k == 0) {
                                                        game_ball.rebound_y();
                                                        k = 1;
                                                }
                                                b[i][j]->unvis();
                                                point+= b[i][j]->point();
                                                print_points(point);
                                                break;
                                        }
                                }
                                if ((game_ball.get_x(2) > b[i][j]->get_x(0)) && (game_ball.get_x(2) < b[i][j]->get_x(1))) {
                                        if (b[i][j]->get_vis() != 0) {
                                                if (k == 0) {
                                                        game_ball.rebound_y();
```

```
                                        k = 1;
                                }
                                b[i][j]->unvis();
                                point+= b[i][j]->point();
                                print_points(point);
                                break;
                        }
                }
                if ((game_ball.get_x(1) > b[i][j]->get_x(0)) && (game_ball.get_x(1) < b[i][j]->get_x(1))) {
                        if (b[i][j]->get_vis() != 0) {
                                if (k == 0) {
                                        game_ball.rebound_y();
                                        k = 1;
                                }
                                b[i][j]->unvis();
                                point+= b[i][j]->point();
                                print_points(point);
                                break;
                        }
                }
        }

        if ((game_ball.get_x(2) == b[i][j]->get_x(0)) || (game_ball.get_x(1) == b[i][j]->get_x(1))) {
                if ((game_ball.get_y(0) > b[i][j]->get_y(0)) && (game_ball.get_y(0) < b[i][j]->get_y(1))) {
                        if (b[i][j]->get_vis() != 0) {
                                if (k == 0) {
                                        game_ball.rebound_x();
                                        k = 1;
                                }
                                b[i][j]->unvis();
                                point+= b[i][j]->point();
                                print_points(point);
                                break;
                        }
                }
                if ((game_ball.get_y(1) > b[i][j]->get_y(0)) && (game_ball.get_y(1) < b[i][j]->get_y(1))) {
                        if (b[i][j]->get_vis() != 0) {
                                if (k == 0) {
                                        game_ball.rebound_x();
                                        k = 1;
                                }
                                b[i][j]->unvis();
                                point+= b[i][j]->point();
                                print_points(point);
                                break;
                        }
                }
                if ((game_ball.get_y(2) > b[i][j]->get_y(0)) && (game_ball.get_y(2) < b[i][j]->get_y(1))) {
                        if (b[i][j]->get_vis() != 0) {
                                if (k == 0) {
                                        game_ball.rebound_x();
                                        k = 1;
                                }
```

```
                                        b[i][j]->unvis();
                                        point+= b[i][j]->point();
                                        print_points(point);
                                        break;
                                }
                        }
                }

                if ((game_ball.get_y(1) == b[i][j]->get_y(1)) && ((game_ball.get_x(2) == b[i][j]->get_x(0)) ||
(game_ball.get_x(1) == b[i][j]->get_x(1)))) {
                        if (b[i][j]->get_vis() != 0) {
                                if (k == 0) {
                                        if (game_ball.get_d(1) > 0) game_ball.rebound_x();
                                        else game_ball.rebound_y();
                                        k = 1;
                                }
                                b[i][j]->unvis();
                                point+= b[i][j]->point();
                                print_points(point);
                        }
                }
                if ((game_ball.get_y(2) == b[i][j]->get_y(1)) && ((game_ball.get_x(2) == b[i][j]->get_x(0)) ||
(game_ball.get_x(1) == b[i][j]->get_x(1)))) {
                        if (b[i][j]->get_vis() != 0) {
                                if (k == 0) {
                                        if (game_ball.get_d(1) > 0)     game_ball.rebound_y();
                                        else game_ball.rebound_x();
                                        k = 1;
                                }
                                b[i][j]->unvis();
                                point+= b[i][j]->point();
                                print_points(point);
                        }
                }
                if (point == level_point) return life;
        }
    }
    if (game_ball.get_y(2) == game_board.get_y()) {
        if ((game_ball.get_x(1) <= game_board.get_x(1)) && (game_ball.get_x(2) >= game_board.get_x(0))) {
                game_ball.rebound_y();
        }
    }
    if (game_ball.get_y(0) >= 590) {
        death = 0;
        game_board.death();
        game_ball.death();
    }
    if (kbhit()) {
        ch = getch();
        switch(ch) {
                case 'a':
                        game_board.move(-1);
                        break;
```

```
                        case 'd':
                                game_board.move(1);
                                break;
                        case 'u':
                                return life;
                }
        }
    }
    life--;
}
}
```

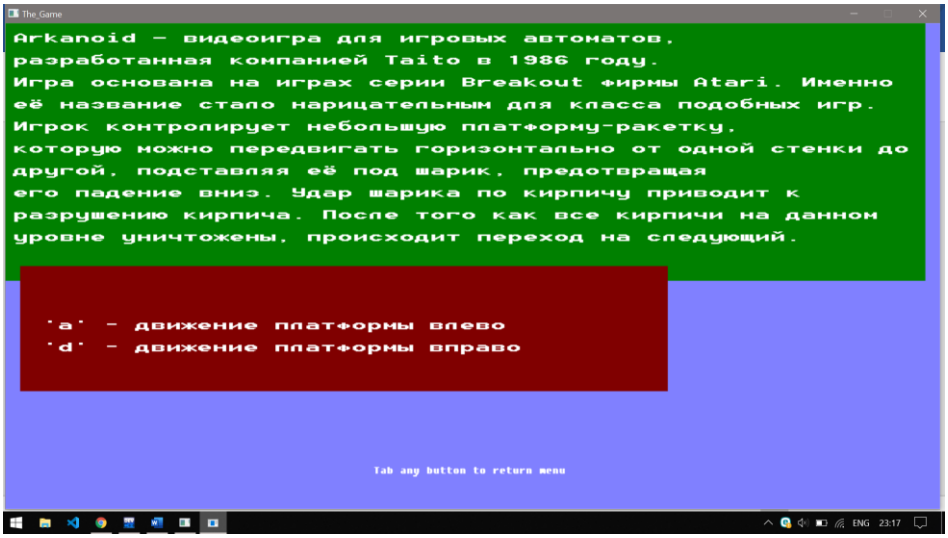
5. Из игры возвращается количество жизней, если они равны 0, то запускается функция game_over(), иначе level_up(&level) для увеличения уровня, пока он не равен 4, после чего запускается функция you_win(), и заново

```
do {
        life = game(level);
        if (life > 0) level_up(&level);
} while ((life != 0) && (level < 4));
if (life == 0) game_over();
else you_win();
break;
```

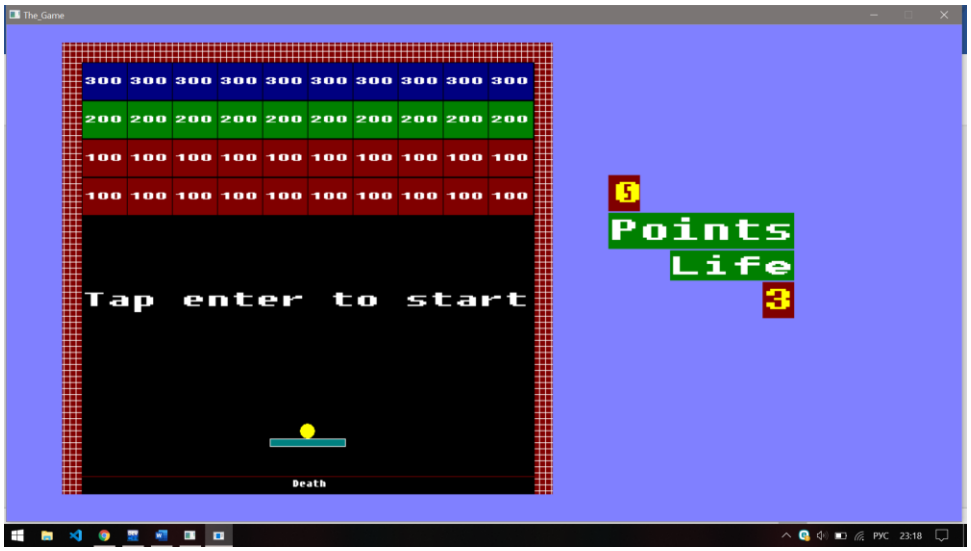
5. Результаты работы

1. Запуск программы

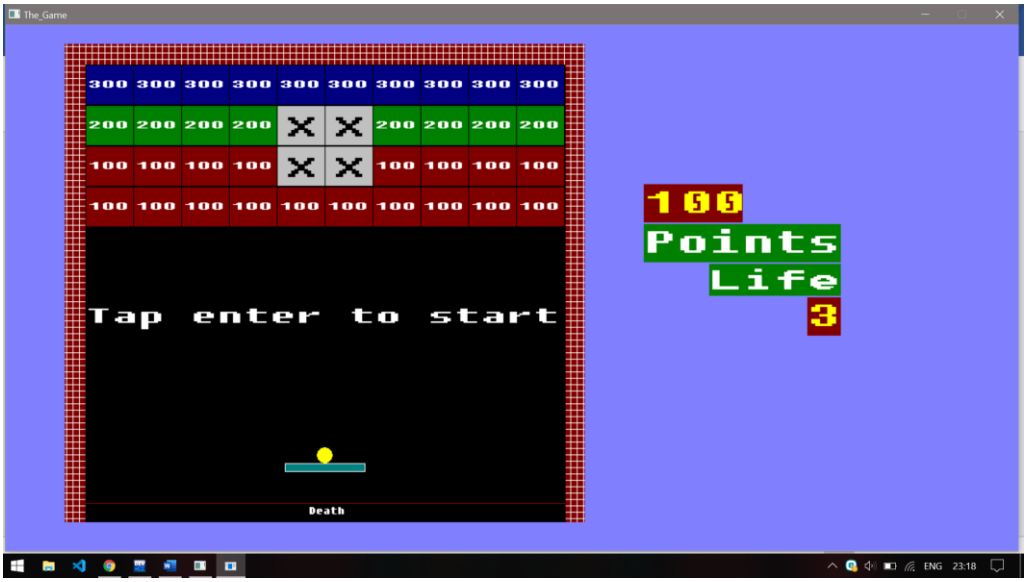
2. Мануал



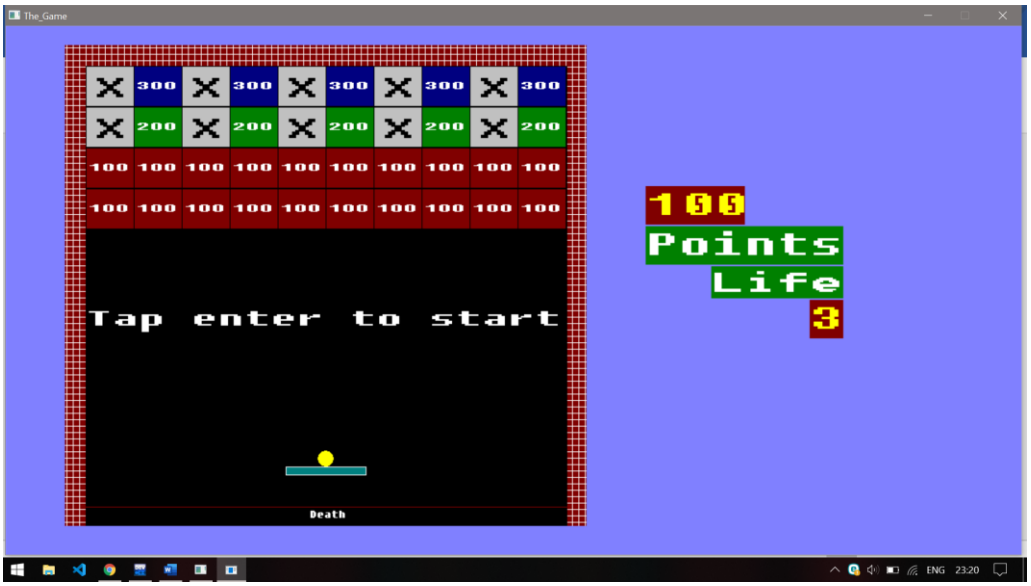
3. Игра 1 уровень



4. Игра 2 уровень



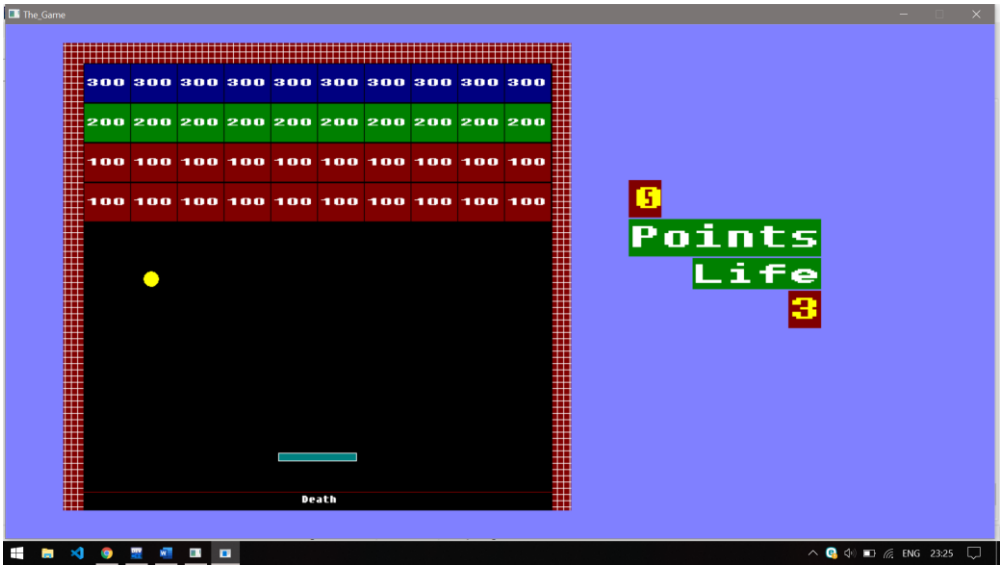
5. Игра 3 уровень

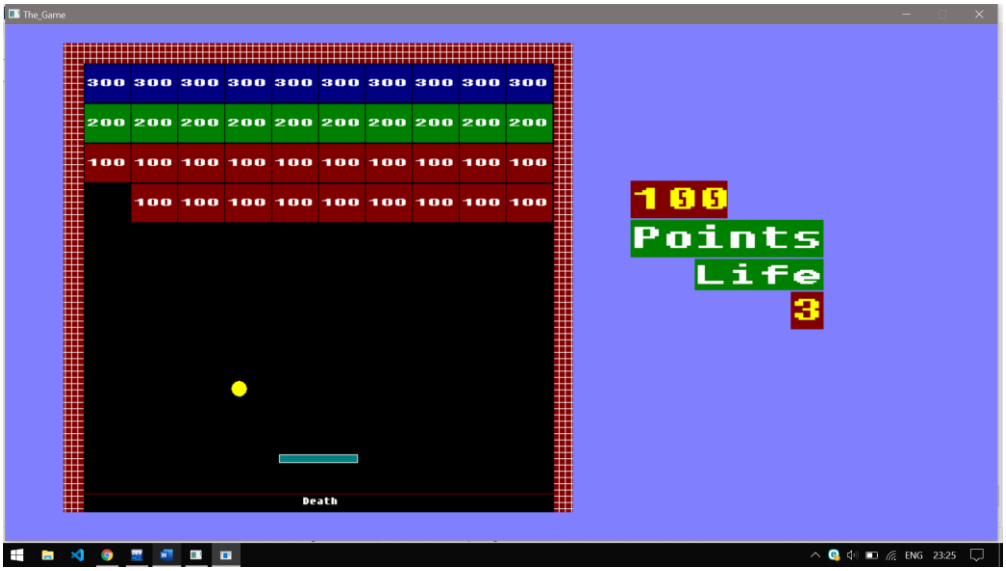


6. Победа

7. Поражение

8. Отскок мяча и исчезновение блока





6. Приложение. Текст программы

```
#include <iostream>
#include <conio.h>
#include <cstdlib>
#include <graphics.h>
#include <ctime>

int point = 0;

class blocks {
	protected:
		int x, y, dx, dy, vis, points;
	public:
		blocks() {
			dx = 60;
			dy = 50;
			vis = 1;
		}
		int get_y(int n) {
			return (n == 0 ? y : y + dy);
		}
		int get_x(int n) {
			return (n == 0 ? x : x + dx);
		}
		int get_vis() {
			return vis;
		}
		int point() {
			return points;
		}
		void unvis() {
			if (vis == 2) return;
			vis = 0;
			setfillstyle(1, 0);
			setcolor(0);
			bar(x, y, x + dx, y + dy);
			rectangle(x, y, x + dx, y + dy);
		}
};

class red_block: public blocks {
	protected:
		int const color = 4;
	public:
		red_block(int x1, int y1) {
			points = 100;
			x = x1;
			y = y1;
			setfillstyle(1, color);
			setcolor(0);
			bar(x, y, x + dx, y + dy);
			rectangle(x, y, x + dx, y + dy);
			settextstyle(0, 0, 2);
			setbkcolor(color);
			setcolor(15);
			outtextxy(x + 3, y + 17, "1OO");
		}
};

class green_block: public blocks {
	protected:
		int const color = 2;
	public:
		green_block(int x1, int y1) {
			points = 200;
			x = x1;
			y = y1;
			setfillstyle(1, color);
			setcolor(0);
			bar(x, y, x + dx, y + dy);
			rectangle(x, y, x + dx, y + dy);
			settextstyle(0, 0, 2);
			setbkcolor(color);
			setcolor(15);
			outtextxy(x + 3, y + 17, "2OO");
		}
};

class blue_block: public blocks {
	protected:
		int const color = 1;
	public:
		blue_block(int x1, int y1) {
			points = 300;
			x = x1;
			y = y1;
			setfillstyle(1, color);
```

```
                setcolor(0);
                bar(x, y, x + dx, y + dy);
                rectangle(x, y, x + dx, y + dy);
                settextstyle(0, 0, 2);
                setbkcolor(color);
                setcolor(15);
                outtextxy(x + 3, y + 17, "3OO");
        }
};

class grey_block: public blocks {
        protected:
                int const color = 7;
        public:
                grey_block(int x1, int y1) {
                        points = 0;
                        vis = 2;
                        x = x1;
                        y = y1;
                        setfillstyle(1, color);
                        setcolor(0);
                        bar(x, y, x + dx, y + dy);
                        rectangle(x, y, x + dx, y + dy);
                        settextstyle(0, 0, 5);
                        setbkcolor(color);
                        setcolor(0);
                        outtextxy(x + 10, y + 7, "X");
                }
};

class ball {
        protected:
                float x, y;
                int dx, dy;
                int const color = 14;
                int const radius = 10;
        public:
                ball() {
                        dy = -1;
                        do {
                                dx = rand() % 3 - 1;
                        } while (dx == 0);
                }
                void rebound_y() {
                        dy = -dy;
                }
                void rebound_x() {
                        dx = -dx;
                }
                int get_y(int n) {
                        if (n == 0) return y;
                        return (n == 1 ? y - radius : y + radius);
                }
                int get_x(int n) {
                        if (n == 0) return x;
                        return (n == 1 ? x - radius : x + radius);
                }
                int get_d(int n) {
                        return (n == 0 ? dx : dy);
                }
                virtual void move() {};
};

class normal_ball: public ball {
        protected:
                float const speed = 0.25;
        public:
                normal_ball(float x1, float y1) {
                        x = x1;
                        y = y1;
                }
                void move(int n) {
                        if (n == 0) {
                                setcolor(color);
                                setfillstyle(1, color);
                                fillellipse(x, y, radius, radius);
                                ellipse(x, y, 0, 360, radius, radius);
                        } else {
                                setcolor(color);
                                setfillstyle(1, color);
                                fillellipse(x, y, radius, radius);
                                ellipse(x, y, 0, 360, radius, radius);
                                setcolor(0);
                                setfillstyle(1, 0);
                                fillellipse(x, y, radius, radius);
                                ellipse(x, y, 0, 360, radius, radius);
                                if (y <= 60) dy = -dy;
                                if ((x <= 110) || (x >= 690)) dx = -dx;
                                x += dx * speed;
                                y += dy * speed;
```

```
                                        setcolor(color);
                                        setfillstyle(1, color);
                                        fillellipse(x, y, radius, radius);
                                        ellipse(x, y, 0, 360, radius, radius);
                                }
                        }
                        void death() {
                                setfillstyle(1, 0);
                                fillellipse(x, y, radius, radius);
                        }
};

class board {
        protected:
                int x, y, color, dx;
        public:
                board() {
                        color = 3;
                }
                int get_y() {
                        return y;
                }
                virtual void move() {}
                virtual void death() {}
};

class normal_board: public board {
        protected:
                int const x1 = 100;
                int const y1 = 10;
                int const dx = 3;

        public:
                normal_board(int x2, int y2) {
                        x = x2;
                        y = y2;
                }
                void move(int n) {
                        if (n == 0) {
                                setcolor(15);
                                setfillstyle(1, color);
                                bar(x, y, x + x1, y + y1);
                                rectangle(x, y, x + x1, y + y1);
                        } else {
                                setcolor(0);
                                setfillstyle(1, 0);
                                bar(x, y, x + x1, y + y1);
                                rectangle(x, y, x + x1, y + y1);
                                if ((x - dx <= 100) && (n == -1)) return;
                                if ((x + x1 + dx >= 700) && (n == 1)) return;
                                (n == -1 ? x -= dx : x += dx);
                                setcolor(15);
                                setfillstyle(1, color);
                                bar(x, y, x + x1, y + y1);
                                rectangle(x, y, x + x1, y + y1);
                        }
                }
                int get_x(int n) {
                        return (n == 0 ? x : x + x1);
                }
                void death() {
                        setcolor(0);
                        setfillstyle(1, 0);
                        bar(x, y, x + x1, y + y1);
                        rectangle(x, y, x + x1, y + y1);
                }
};

int menu() {
        int choice = 1;
        int x1, x2, y1, y2;
        char ch;

        x1 = 400;
        x2 = 800;
        y1 = 280;
        y2 = 340;

        cleardevice();
        setfillstyle(1, 9);
        bar(0, 0, 1270, 660);
        settextstyle(0, 0, 10);
        setcolor(14);
        setbkcolor(9);
        outtextxy(400, 100, "Arkanoid");
        setcolor(14);
        settextstyle(0, 0, 4);
        outtextxy(530, 300, "Play");
        outtextxy(500, 350, "Manual");
        outtextxy(530, 400, "Quit");
```

```
		setcolor(GREEN);
		setfillstyle(1, GREEN);
		fillellipse(635, 700, 700, 100);
		setcolor(15);
		rectangle(x1, y1, x2, y2);
		while((ch = getch()) != 13) {
			switch(ch) {
				case 's':
					if (choice == 3) break;
					choice++;
					setcolor(9);
					rectangle(x1, y1, x2, y2);
					setcolor(15);
					y1 += 53;
					y2 += 53;
					rectangle(x1, y1, x2, y2);
					break;
				case 'w':
					if (choice == 1) break;
					choice--;
					setcolor(9);
					rectangle(x1, y1, x2, y2);
					setcolor(15);
					y1 -= 53;
					y2 -= 53;
					rectangle(x1, y1, x2, y2);
					break;
			}
		}
		return choice;
}

void print_points(int point) {
		char points[2];

		itoa(point, points, 10);
		setcolor(15);
		setbkcolor(2);
		settextstyle(0, 0, 6);
		outtextxy(800, 250, "Points");
		settextstyle(0, 0, 6);
		setcolor(14);
		setbkcolor(4);
		outtextxy(800, 200, points);
}

void print_life(int life) {
		char lifes[2];

		setbkcolor(2);
		itoa(life, lifes, 10);
		setcolor(15);
		settextstyle(0, 0, 5);
		outtextxy(882, 300, "Life");
		settextstyle(0, 0, 6);
		setcolor(14);
		setbkcolor(4);
		outtextxy(1005, 342, lifes);
}

void start(int life) {
		char ch;
		print_life(life);
		do {
			settextstyle(0, 0, 4);
			setcolor(15);
			setbkcolor(0);
			outtextxy(100, 350, "Tap enter to start");
		} while ((ch = getch()) != 13);
		setcolor(0);
		outtextxy(100, 350, "Tap enter to start");
}

int game(int level) {
		char ch;
		int life = 3, death, level_point = 0;

		cleardevice();
		setfillstyle(1, 9);
		bar(0, 0, 1270, 660);
		setfillstyle(7, 15);
		setbkcolor(4);
		bar(74, 24, 726, 624);
		setfillstyle(1, 0);
		bar(100, 50, 702, 624);
		setcolor(4);
		line(100, 600, 702, 600);
		settextstyle(0, 0, 1);
		setcolor(15);
		setbkcolor(0);
```

```
	outtextxy(380, 602, "Death");

	blocks *b[4][10];
	int y = 50;
	for (int i = 0; i < 4; i++) {
		int x = 100;
		for (int j = 0; j < 10; j++) {
			if (level == 1) {
				if (i == 0) b[i][j] = new blue_block(x, y);
				if (i == 1) b[i][j] = new green_block(x, y);
				if (i > 1) b[i][j] = new red_block(x, y);
				level_point = 7000;

			}
			if (level == 2) {
				if (i == 0) b[i][j] = new blue_block(x, y);
				if ((i == 1) && ((j != 4) || (j != 5))) b[i][j] = new green_block(x, y);
				if ((i == 1) && ((j == 4) || (j == 5))) b[i][j] = new grey_block(x, y);
				if ((i == 2) && ((j != 4) || (j != 5))) b[i][j] = new red_block(x, y);
				if ((i == 2) && ((j == 4) || (j == 5))) b[i][j] = new grey_block(x, y);
				if (i == 3) b[i][j] = new red_block(x, y);
				level_point = 13400;
			}
			if (level == 3) {
				if ((i == 0) && (j % 2 != 0)) b[i][j] = new blue_block(x, y);
				if ((i == 0) && (j % 2 == 0)) b[i][j] = new grey_block(x, y);
				if ((i == 1) && (j % 2 != 0)) b[i][j] = new green_block(x, y);
				if ((i == 1) && (j % 2 == 0)) b[i][j] = new grey_block(x, y);
				if ((i > 1) && (j % 2 != 0)) b[i][j] = new red_block(x, y);
				if ((i > 1) && (j % 2 == 0)) b[i][j] = new red_block(x, y);
				level_point = 17900;
			}
			x += 60;
		}
		y += 51;
	}
	print_points(point);
	while (1) {
		normal_ball game_ball(400, 540);
		normal_board game_board(350, 550);
		game_board.move(0);
		game_ball.move(0);
		if (life == 0) return life;
		start(life);
		death = 1;
		while(death) {
			game_board.move(0);
			game_ball.move(1);
			for (int i = 0; i < 4; i++) {
				for (int j = 0; j < 10; j++) {
					int k = 0;
					if ((game_ball.get_y(1) == b[i][j]->get_y(1)) || (game_ball.get_y(2) == b[i][j]->get_y(0))) {
						if ((game_ball.get_x(0) > b[i][j]->get_x(0)) && (game_ball.get_x(0) < b[i][j]->get_x(1)))
{
							if (b[i][j]->get_vis() != 0) {
								if (k == 0) {
									game_ball.rebound_y();
									k = 1;
								}
								b[i][j]->unvis();
								point+= b[i][j]->point();
								print_points(point);
								break;
							}
						}
						if ((game_ball.get_x(2) > b[i][j]->get_x(0)) && (game_ball.get_x(2) < b[i][j]->get_x(1)))
{
							if (b[i][j]->get_vis() != 0) {
								if (k == 0) {
									game_ball.rebound_y();
									k = 1;
								}
								b[i][j]->unvis();
								point+= b[i][j]->point();
								print_points(point);
								break;
							}
						}
						if ((game_ball.get_x(1) > b[i][j]->get_x(0)) && (game_ball.get_x(1) < b[i][j]->get_x(1)))
{
							if (b[i][j]->get_vis() != 0) {
								if (k == 0) {
									game_ball.rebound_y();
									k = 1;
								}
								b[i][j]->unvis();
								point+= b[i][j]->point();
								print_points(point);
								break;
							}
```

```
                                        }
                                }

                                if ((game_ball.get_x(2) == b[i][j]->get_x(0)) || (game_ball.get_x(1) == b[i][j]->get_x(1))) {
                                        if ((game_ball.get_y(0) > b[i][j]->get_y(0)) && (game_ball.get_y(0) < b[i][j]->get_y(1)))
{
                                                if (b[i][j]->get_vis() != 0) {
                                                        if (k == 0) {
                                                                game_ball.rebound_x();
                                                                k = 1;
                                                        }
                                                        b[i][j]->unvis();
                                                        point+= b[i][j]->point();
                                                        print_points(point);
                                                        break;
                                                }
                                        }
                                        if ((game_ball.get_y(1) > b[i][j]->get_y(0)) && (game_ball.get_y(1) < b[i][j]->get_y(1)))
{
                                                if (b[i][j]->get_vis() != 0) {
                                                        if (k == 0) {
                                                                game_ball.rebound_x();
                                                                k = 1;
                                                        }
                                                        b[i][j]->unvis();
                                                        point+= b[i][j]->point();
                                                        print_points(point);
                                                        break;
                                                }
                                        }
                                        if ((game_ball.get_y(2) > b[i][j]->get_y(0)) && (game_ball.get_y(2) < b[i][j]->get_y(1)))
{
                                                if (b[i][j]->get_vis() != 0) {
                                                        if (k == 0) {
                                                                game_ball.rebound_x();
                                                                k = 1;
                                                        }
                                                        b[i][j]->unvis();
                                                        point+= b[i][j]->point();
                                                        print_points(point);
                                                        break;
                                                }
                                        }
                                }

                                if ((game_ball.get_y(1) == b[i][j]->get_y(1)) && ((game_ball.get_x(2) == b[i][j]->get_x(0)) ||
(game_ball.get_x(1) == b[i][j]->get_x(1)))) {
                                        if (b[i][j]->get_vis() != 0) {
                                                if (k == 0) {
                                                                if (game_ball.get_d(1) > 0) game_ball.rebound_x();
                                                                else game_ball.rebound_y();

                                                                k = 1;
                                                }
                                                b[i][j]->unvis();
                                                point+= b[i][j]->point();
                                                print_points(point);
                                        }
                                }
                                if ((game_ball.get_y(2) == b[i][j]->get_y(1)) && ((game_ball.get_x(2) == b[i][j]->get_x(0)) ||
(game_ball.get_x(1) == b[i][j]->get_x(1)))) {
                                        if (b[i][j]->get_vis() != 0) {
                                                if (k == 0) {
                                                                if (game_ball.get_d(1) > 0)
        game_ball.rebound_y();
                                                                else game_ball.rebound_x();
                                                                k = 1;
                                                }
                                                b[i][j]->unvis();
                                                point+= b[i][j]->point();
                                                print_points(point);
                                        }
                                }
                                if (point == level_point) return life;
                        }
                }
                if (game_ball.get_y(2) == game_board.get_y()) {
                        if ((game_ball.get_x(1) <= game_board.get_x(1)) && (game_ball.get_x(2) >= game_board.get_x(0))) {
                                game_ball.rebound_y();
                        }
                }
                if (game_ball.get_y(0) >= 590) {
                        death = 0;
                        game_board.death();
                        game_ball.death();
                }
                if (kbhit()) {
                        ch = getch();
                        switch(ch) {
                                case 'a':
```

```
                                                                game_board.move(-1);
                                                                break;
                                                        case 'd':
                                                                game_board.move(1);
                                                                break;
                                                        case 'u':
                                                                return life;

                                                }
                                        }
                                }
                                life--;
                        }
        }

        void game_over() {
                        char ch;
                        cleardevice();
                        setcolor(0);
                        setfillstyle(1, 0);
                        bar(0, 0, 1270, 660);
                        setbkcolor(0);
                        settextstyle(0, 0, 10);
                        setcolor(15);
                        outtextxy(400, 400, "Game over");
                        settextstyle(0, 0, 1);
                        outtextxy(540, 575, "Tab enter to continue");
                        setcolor(15);
                        setfillstyle(1, 15);
                        bar(550, 75, 700, 105);
                        bar(510, 105, 740, 135);
                        bar(470, 135, 780, 165);
                        bar(470, 165, 510, 195);
                        bar(575, 165, 675, 195);
                        bar(740, 165, 780, 195);
                        bar(470, 195, 510, 225);
                        bar(600, 195, 650, 225);
                        bar(740, 195, 780, 225);
                        bar(470, 225, 780, 255);
                        bar(510, 255, 600, 285);
                        bar(650, 255, 740, 285);
                        bar(550, 285, 700, 315);
                        bar(550, 315, 575, 345);
                        bar(600, 315, 650, 345);
                        bar(675, 315, 700, 345);
                        while ((ch = getch()) != 13);
        }

        void you_win() {
                        char ch;
                        cleardevice();
                        setcolor(7);
                        setfillstyle(1, 9);
                        bar(0, 0, 1270, 660);
                        setbkcolor(9);
                        settextstyle(0, 0, 10);
                        setcolor(14);
                        outtextxy(425, 300, "You win!");
                        char points[6];
                        itoa(point, points, 10);
                        settextstyle(0, 0, 1);
                        outtextxy(540, 575, "Tab enter to continue");
                        setcolor(0);
                        setfillstyle(1, 0);
                        bar(550, 40, 700, 50);
                        bar(550, 40, 560, 180);
                        bar(690, 40, 700, 180);
                        bar(550, 180, 580, 190);
                        bar(670, 180, 700, 190);
                        bar(570, 190, 600, 200);
                        bar(650, 190, 680, 200);
                        bar(590, 200, 620, 210);
                        bar(630, 200, 660, 210);
                        bar(610, 210, 620, 270);
                        bar(630, 210, 640, 270);
                        bar(600, 270, 620, 280);
                        bar(630, 270, 650, 280);
                        bar(590, 280, 600, 300);
                        bar(650, 280, 660, 300);
                        bar(590, 290, 660, 300);
                        bar(510, 70, 550, 80);
                        bar(700, 70, 740, 80);
                        bar(510, 80, 520, 120);
                        bar(730, 80, 740, 120);
                        bar(510, 120, 535, 130);
                        bar(715, 120, 740, 130);
                        bar(525, 130, 550, 140);
                        bar(700, 130, 725, 140);
                        setfillstyle(1, 14);
                        bar(560, 50, 690, 180);
```

```
		bar(580, 180, 670, 190);
		bar(590, 190, 660, 200);
		bar(620, 200, 630, 290);
		bar(600, 280, 650, 290);
		setfillstyle(1, 15);
		bar(570, 70, 580, 140);
		setcolor(14);
		setbkcolor(9);
		settextstyle(0, 0, 4);
		outtextxy(410, 400, "Points:");
		outtextxy(670, 400, points);
		while ((ch = getch()) != 13);
}

void level_up(int *level) {
		*level += 1;
}
void manual() {
		char ch;
		cleardevice();
		setcolor(7);
		setfillstyle(1, 9);
		bar(0, 0, 1270, 660);
		settextstyle(0, 0, 3);
		setcolor(15);
		char text1[] = "Arkanoid — видеоигра для игровых автоматов, ";
		char text2[] = "разработанная компанией Taito в 1986 году.";
		char text3[] = "Игра основана на играх серии Breakout фирмы Atari. Именно";
		char text4[] = "её название стало нарицательным для класса подобных игр.";
		char text5[] = "Игрок контролирует небольшую платформу-ракетку,";
		char text6[] = "которую можно передвигать горизонтально от одной стенки до";
		char text7[] = "другой, подставляя её под шарик, предотвращая";
		char text8[] = "его падение вниз. Удар шарика по кирпичу приводит к";
		char text9[] = "разрушению кирпича. После того как все кирпичи на данном";
		char text10[] = "уровне уничтожены, происходит переход на следующий.";

		setfillstyle(1, 2);
		bar(0, 0, 1250, 350);
		setbkcolor(2);
		outtextxy(10, 10, text1);
		outtextxy(10, 40, text2);
		outtextxy(10, 70, text3);
		outtextxy(10, 100, text4);
		outtextxy(10, 130, text5);
		outtextxy(10, 160, text6);
		outtextxy(10, 190, text7);
		outtextxy(10, 220, text8);
		outtextxy(10, 250, text9);
		outtextxy(10, 280, text10);
		setfillstyle(1, 4);
		bar(20, 330, 900, 500);
		setbkcolor(4);
		outtextxy(50, 400, "'a' - движение платформы влево");
		outtextxy(50, 430, "'d' - движение платформы вправо");
		setbkcolor(9);
		settextstyle(0, 0, 1);
		outtextxy(500, 600, "Tab any button to return menu");
		getch();
}
int main()
{
		srand(time(NULL));
		initwindow(1270, 660, "The_Game");
		int level, life;

		while (1) {
			int choice = menu();
			level = 1;
			point = 0;

			switch(choice){
				case 3:
					return 0;
				case 2:
					manual();
					break;
				case 1:
					do {
						life = game(level);
						if (life > 0) level_up(&level);
					} while ((life != 0) && (level < 4));
					if (life == 0) game_over();
					else you_win();
					break;
			}
		}
}
```