УДК 004.942

Создание игры змейка на C++

Семченко Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В работе описаны возможности кода на C++ и разработка культовой игры змейка, известная еще со старых кнопочных телефонов. Разработка происходит в среде Visual Studio.
Ключевые слова: смартфон, C++, змейка, игра.

Creating a snake game in C ++

Semchenko Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The paper describes the features of the C ++ code and the development of the cult game Snake, known from the old push-button phones. Development takes place in the Visual Studio environment.
Keywords: Smartphone, C ++, Snake, Game.

У всех взрослых и детей были кнопочные мобильные телефоны, где была установлена игра змейка. Большинство людей знают и играли в эту игру. Сама игра представляет движущуюся змейку, которой нужно собирать различные фрукты, для того, чтобы вырасти и набрать максимальное количество очков. В самом начале игры, змея выглядит маленьким головастиком. Но с каждым поеденным фруктом она становится больше, пока не заполнит всю область карты. [1]

Исследовательской задачей является описание проекта разработки консольного приложения.

Программа написана в Visual Studio на языке C++, язык программирования Visual C ++ и инструменты разработки помогут

разработать собственные универсальные приложения Windows, собственные приложения для настольных компьютеров и серверов, кроссплатформенные библиотеки, которые работают на Android и iOS, а также Windows, и управляемые приложения, которые работают на .NET Framework.[2]

Первым делом необходимо написать функции ввода, логики, рисования поля и т.д.

```
#include <iostream>

void Setup() {

}

void Draw() {

}

void Input() {

}

void Main () {

}
```

Теперь можно вводить переменные которые будут использоваться.

```
using namespace std;

bool gameOver;
const int width = 20;
const int height = 20;
int x, y, fruitX, fruitY, score;
int tailX[100], tailY[100];
int nTail;
enum eDirection { STOP = 0, LEFT, RIGHT, UP, DOWN };
eDirection dir;
```

Переменная gameOver используется при столкновении со стенкой, либо со своим хвостом. Далее обозначаем ширину и высоту поля, а так же добавляем переменные расположения змейки, ее хвоста, фруктов и очков. Далее заполняем функцию Setup.

```
void Setup() {
	gameOver = false;
	dir = STOP;
	x = width / 2 - 1;
	y = height / 2 - 1;
	fruitX = rand() % width;
	fruitY = rand() % height;
	score = 0;
}
```

Изначально переменная gameOver должна иметь значение ложь, чтобы в самом начале игрок не проигрывал. Также расположим змейку ровно по центру, фрукт в случайном месте, а количество очком выставим 0.

Следующим шагом заполним функцию Draw.

```
void Draw() {
	system("cls");
	for (int i = 0; i < width + 1; i++)
		cout << "#";
	cout << endl;

	for (int i = 0; i < height; i++) {
		for (int j = 0; j < width; j++) {
			if (j == 0 || j == width - 1)
				cout << "#";
			if (i == y && j == x)
				cout << "0";
			else if (i == fruitY && j == fruitX)
				cout << "F";
			else {
				bool print = false;
				for (int k = 0; k < nTail; k++) {
					if (tailX[k] == j && tailY[k] == i)
						print = true;
						cout << ".";
				}
				if (!print)
				cout << " ";
			}
		}
		cout << endl;
	}

	for (int i = 0; i < width + 1; i++)
```

```
            cout << "#";
        cout << endl;
        cout << "Score: " << score << endl;
}
```

Здесь используется множество циклов, для прорисовки змейки, фруктов, хвоста змейки. А так же в самом начале при запуске этой функции она очищается и рисует все заново, чтоб все не хранилось в памяти.

Дальше на очереди функция ввода.

```
void Input() {
        if (_kbhit()) {
            switch (_getch())
            {
            case 'a':
                dir = LEFT;
                break;
            case 'w':
                dir = UP;
                break;
            case 'd':
                dir = RIGHT;
                break;
            case 's':
                dir = DOWN;
                break;
            case 'x':
                gameOver = true;
                break;
            }
        }
}
```

Обозначаем, какие клавиши будут использоваться для передвижения змейки, и кнопка досрочного завершения игры.

Теперь остается самое сложное, это написание функции логики, здесь будут определены возможности змейки, и ее логическое поведение.

```
void Logic() {
        int prevX = tailX[0];
        int prevY = tailY[0];
        int prev2X, prev2Y;
        tailX[0] = x;
        tailY[0] = y;
```

```
for (int i = 1; i < nTail; i++) {
        prev2X = tailX[i];
        prev2Y = tailY[i];
        tailX[i] = prevX;
        tailY[i] = prevY;
        prevX = prev2X;
        prevY = prev2Y;
}
switch (dir)
{
case LEFT:
        x--;
        break;
case RIGHT:
        x++;
        break;
case UP:
        y--;
        break;
case DOWN:
        y++;
        break;
default:
        break;
}

//if (x > width || x < 0 || y > height || y < 0)
        //gameOver = true;
if (x >= width)
        x = 0;
else if (x < 0)
        x = width;

if (y >= height)
        y = 0;
else if (x < 0)
        y = height;

for (int i = 0; i < nTail; i++) {
        if (tailX[i] == x && tailY[i] == y)
                gameOver = true;
}

if (x == fruitX && y == fruitY) {
        score += 10;;
```

```
            fruitX = rand() % width;
            fruitY = rand() % height;
            nTail++;

        }
}
```

Задаем ей направления движения по нажатию на кнопку, возможность проходить через стены и если такое возможно, чтоб она выходила с другого конца поля. Можно так же задать возможность ограничить непроходимыми стенами, для этого удаляем цикл под строчкой

```
//if (x > width || x < 0 || y > height || y < 0)
            //gameOver = true;
```

И открываем ее из комментариев.

Ну и самое последнее, что необходимо сделать, так это заполнить функцию Main.

```
int main() {
    Setup();
    while (!gameOver) {
        Draw();
        Input();
        Logic();
    }
    return 0;
}
```

Делаем повторение каждой функции, чтобы игра выглядела похожа на оригинал из мобильных телефонов.

Теперь при запуске этого кода будет открываться консоль (рис.1) и там будет отображено поле со змейкой, так же можно поменять вид фруктов, заменив их на другие символы и проделать ту же самую работу со змейкой.

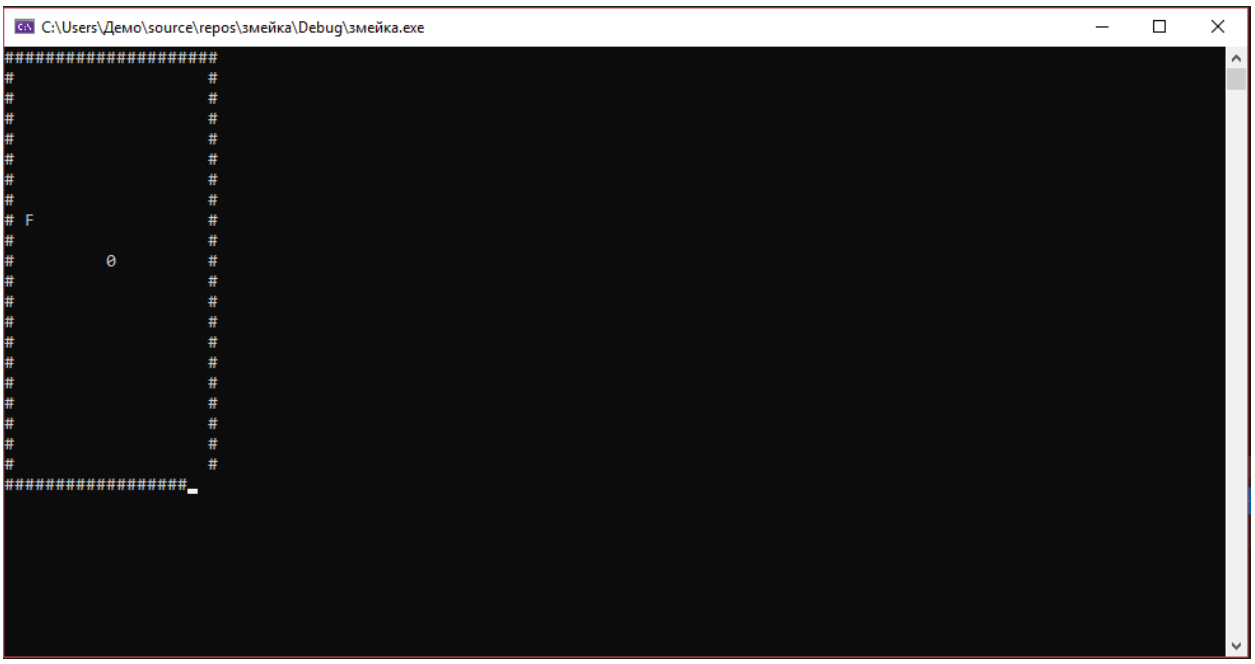

Рисунок 1 – Консоль с игрой

Таким образом, реализация игры Змейка на языке C++ возможна и реализуется с помощью несложного кода. В результате исследования была продемонстрирована пошаговая реализация проекта разработки консольного приложения. Практическим результатом исследования является рабочее консольное приложение.

Библиографический список

1. Игра змейка // Wikipedia URL: https://ru.wikipedia.org/wiki/Snake_(%D0%B8%D0%B3%D1%80%D0%B0) (дата обращения: 15.05.2019).
2. Visual Studio URL: https://visualstudio.microsoft.com/ (дата обращения: 15.05.2019).