Простая компьютерная игра «Лабиринт» на языке программирования C++: этапы реализации

Алексей Куликов, 8 класс

Руководитель: Дединский И.Р., МФТИ

…Началось все с того, что когда в восьмом классе начались уроки программирования, наш учитель сразу запретил нам играть в компьютерные игры. Исключения, по его словам, могли составлять лишь игры, написанные самостоятельно. Тогда я решил написать собственную простую двумерную (2D) игру под Windows на основе графической библиотеки *TXLib*. Потому что это самое простое, что можно было сделать, начиная изучать программирование.

Основной алгоритм таких аркадных игр изображен на рис.1. Он достаточно прост: сначала в качестве фона рисуется текущий уровень игры, затем двигаются и рисуются враги, после этого персонаж управляется игроком с клавиатуры. Затем проверяется, проиграл ли игрок (в нашем случае, врезался ли персонаж в стену) или он выиграл (закончен ли уровень, побеждены ли враги и т.д.). В самом конце на экран выводится главный герой игры.

Этот алгоритм довольно легкий, но это была моя первая игра, и поэтому было решено вести работу поэтапно, т.е. сначала нарисовать статическую картинку, потом добавить движение персонажей, получив мультфильм, и затем, добавив реакцию на клавиши, превратить мультфильм в игру.

Работа была начата с создания статического изображения (рис. 2), которое рисовалось непосредственно с помощью функций библиотеки *TXLib*. Для рисования линий была использована функция *txLine (x1, y1, x2, y2),* которая рисует линию из точки с координатами *x1, y1* в точку с координатами *x2, y2.* Для изменения цвета была использована *функция txSetColor (цвет),* которая устанавливает цвет рисующихся объектов, а для вывода на экран круга – функция *txCircle (x, y, радиус),* которая рисует круг заданного радиуса в заданной точке. (Полное описание всех функций библиотеки есть в ее системе помощи, а сама библиотека бесплатна и доступна на сайте http://ded32.net.ru.)

Рис. 1. Основной алгоритм.

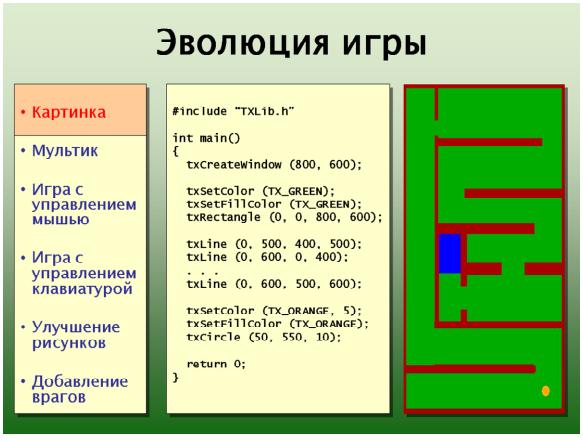

Рис. 2. Рисование изображения.

На следующем этапе было реализовано движение персонажей (рис. 3), т.е. их координаты начали плавно изменяться с течением времени. Это можно сделать только с помощью цикла, например, конструкции цикла *while:*

```
while (условие)
    {
    /*  действия, которые выполняются, если условие верно */
    }
```

На данном этапе основная функция main(), скорее всего, станет очень большой и трудно понимаемой, и тогда программу нужно будет разделить на функции.

После этого можно реализовать управление героем мышью или клавиатурой. Управление мышью: необходимо использовать функции *txMouseX()* и *txMouseY(),* которые возвращают значения координат мыши по осям *X* и *Y*, и возвращаемые значения использовать как координаты для какого-то объекта (рис. 4).

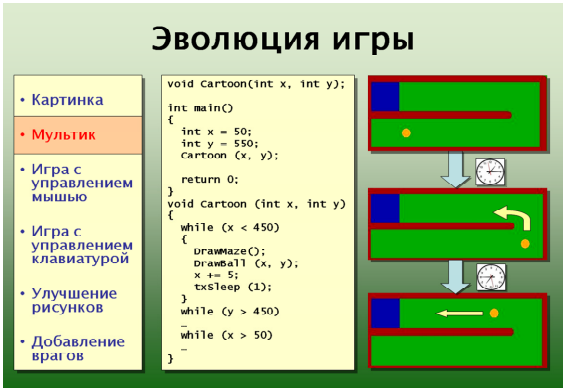

Рис. 3. Движение персонажей.

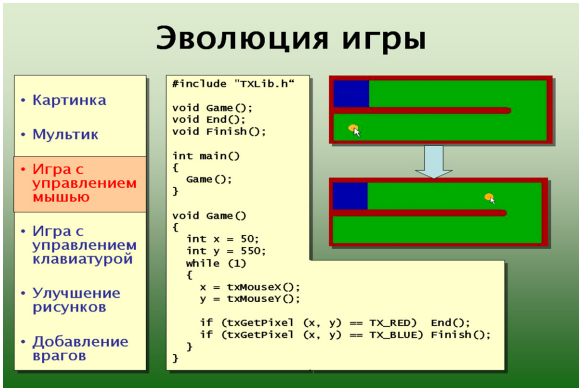

Рис. 4. Управление мышью.

Поскольку появилась возможность управлять действиями персонажа, то написанная программа – уже игра, и значит, в ней должна быть возможность выиграть или проиграть. Легко добавить эту возможность, использовав реакцию на цвет (функцию *txGetPixel (x, y),* возвращающую цвет нужного пикселя), т.е. если, например, цвет под персонажем совпадает с цветом стены, то игрок проиграл.

Для управления с клавиатуры можно использовать функцию *GetAsyncKeyState (код клавиши),* которая проверяет, нажата ли указанная клавиша в момент вызова этой функции.

В игре должна быть приятная графика, а то играть будет совсем не интересно. Рисовать эту «приятную графику» геометрическими фигурами по координатам очень сложно. Поэтому изображения можно заранее нарисовать в любом графическом редакторе, сохранить в формате *.bmp,* после этого загрузить их в программу функцией *txLoadImage (имя файла изображения)* и вывести на экран с помощью функции *txBitBlt*. В конце загруженное изображение надо обязательно удалить, чтобы оно не занимало место в оперативной памяти компьютера и не отнимало ресурсы у Windows (рис. 6).

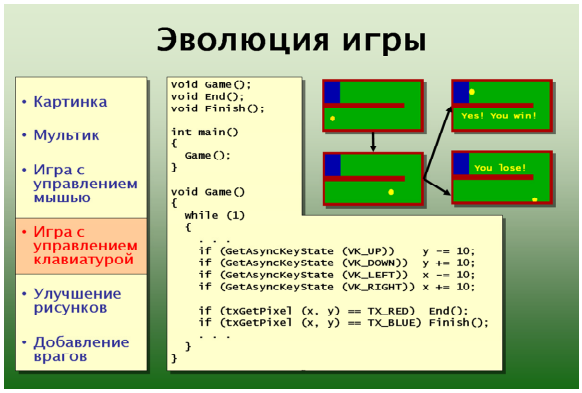

Рис. 5. Управление клавиатурой.

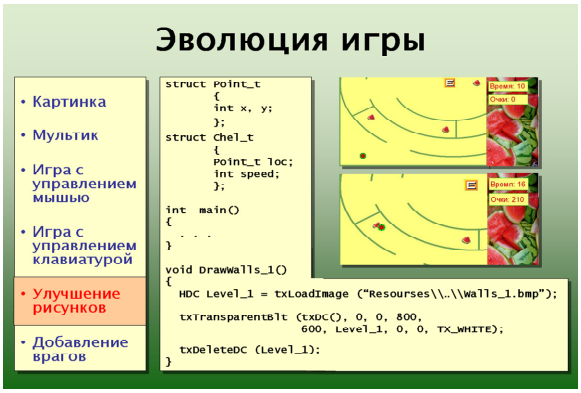

Рис. 6. Графика.

Теперь можно добавить простейшего врага, который будет патрулировать ту или иную территорию, путем введения в программу ещё двух переменных, хранящих значения его координат, и их изменения по какому-то закону (рис 7).

Можно также добавить анимацию объектов путем добавления счетчика, изменения его на единицу каждый ход цикла и вывода определенного изображения в зависимости от показаний счетчика (например, в зависимости от остатка от деления на два, рис. 8). Само собой, количество картинок, которые нужно будет заготовить для отрисовки анимированных персонажей, увеличится.

Рис. 7. Добавление врагов.

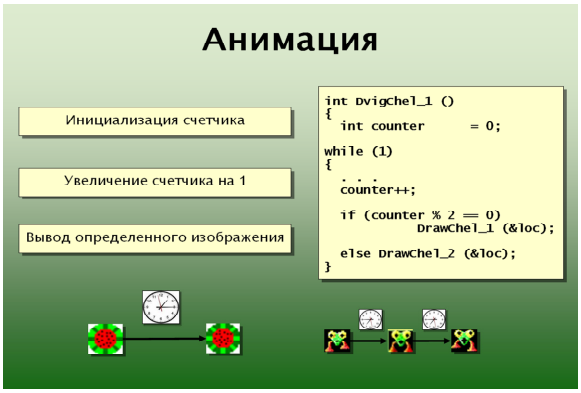

Рис. 8. Реализация простой анимации.

После этого можно, наконец, начать играть на уроках совершенно спокойно, правда, только в эту игру – развивая ее и тестируя. Но, если она надоела, всегда можно начать писать новую!