Игра «Змейка». Отображение движения змейки в графическом интерфейсе.

Темы: обработка нажатия клавиш, использование функций библиотеки для графического вывода.

В проекте будет использоваться библиотека, которая содержит функции, управляющие графическим выводом на экран.

Для того, чтобы воспользоваться этими функциями, необходимо:

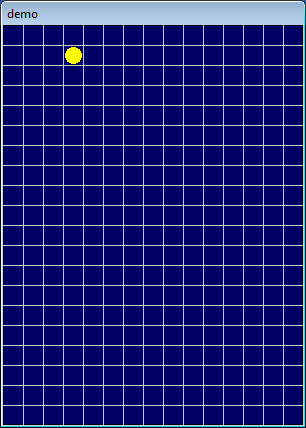
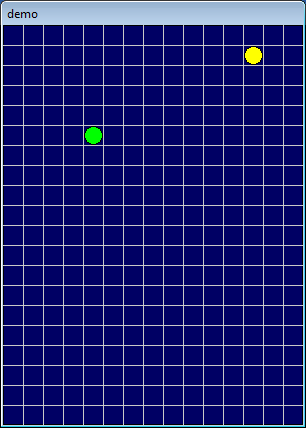
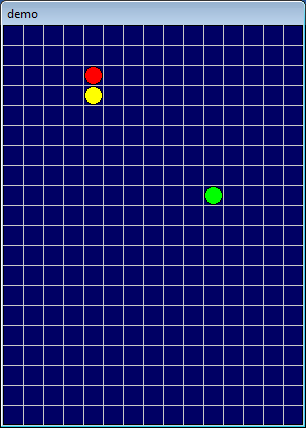
1. В рабочий каталог проекта копировать файлы:

* GraphicsLib.h - содержит объявление функций
* GraphicsLib.lib - статическая библиотека, содержащая функции, управляющие графикой
* demo.exe - оконное приложение, отображением которого будем управлять с помощью функций библиотеки GraphicsLib.lib

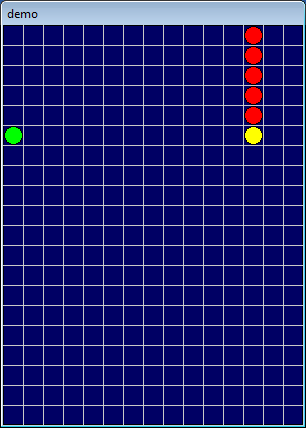
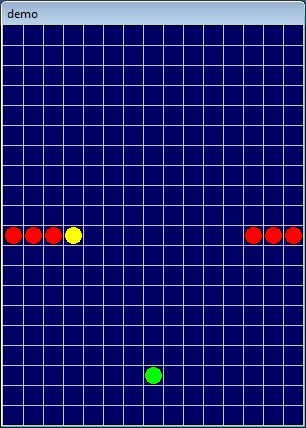
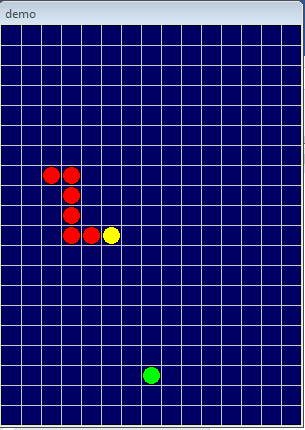
2) В проект добавить заголовочный файл GraphicsLib.h и файл библиотеки GraphicsLib.lib

3) [ можно определить задержку отображения для выполнения анимации змейки с помощью функции **pause**()]

Пример использования функций библиотеки приведен в приложении.

  Chart

Description automatically generated

**Задание 1**

1. Добавьте в проект файлы, которые были созданы в предыдущем задании:

* snake.cpp и snake.h, которые будут содержать информацию, относящуюся к понятию "Змейка"
* game.cpp и game.h, которые будут содержать информацию, относящуюся к понятию "Игра".
* Заголовочный файл const.h, который будет содержать константные данные
* Файл main.cpp, который будет содержать все управляющие инструкции
* field.cpp и field.h, которые будут содержать информацию, относящуюся к понятию "Поле игры".
* food.cpp и food.h, которые будут содержать информацию, относящуюся к понятию "Еда".

## Данные модуля const

Важно! Потребуется изменить значения констант в этой версии проекта, т.к. вместо символов будут отображаться шарики разного цвета, т.е. в качестве значений для констант надо задать «цвет», в качестве которого выбирается элемент перечисления Colors. В качестве типа констант можно задать Colors или int.

|  |  |  |
| --- | --- | --- |
| Описание | Идентификатор | Тип |
| Максимальный размер массива змейки | L |  |
| Символ для хвоста змейки | tail\_symbol | Colors или int |
| Символ для головы змейки | head\_symbol | Colors или int |
| Символ еды | food\_symbol | Colors или int |
| Символ поля | field\_symbol | Colors или int |
| Символ границы | border\_symbol | Colors или int |
| Перечисление «направление движения» | DIRECTION |  |
| Константы для обработки клавиатурного ввода | KEYS |  |

**Задание 2**

Объявите структуры, описывающие основные сущности программы.

## Данные программы

Важно! Так как в графическом интерфейсе будут отображаться шарики разного цвета, то основой игрового поля будет двумерный динамический массив типа int.

|  |  |
| --- | --- |
| Описание | Объявление |
| Структура Snake:   * Текущий размер змейки * Массив координат змейки по горизонтали (X) * Массив координат змейки по вертикали (X) * Направление движения змейки |  |
| Структура Food:   * Координата еды по горизонтали (X) * Координата еды по вертикали (Y) * Признак, что еда установлена |  |
| Структура Field:   * Массив игрового поля (динамический) * Длина игрового поля по горизонтали * Длина игрового поля по вертикали |  |
| Структура Game:   * Таймаут задержки между шагами игры * Признак продолжения игры * Экземпляр структуры Snake * Экземпляр структуры Food * Экземпляр структуры Field |  |

## Данные (переменные) модуля main

|  |  |  |
| --- | --- | --- |
| Описание | Идентификатор | Тип |
| Экземпляр структуры Game |  | Game |
| Текущий размер змейки |  |  |
| Таймаут задержки между шагами игры |  |  |
| Направление движения змейки |  |  |
| Длина игрового поля по горизонтали |  |  |
| Длина игрового поля по вертикали |  |  |
|  |  |  |

### Алгоритм функции init\_game (один из возможных)

1. инициализация игры (вызов функции **init\_game ()**)
2. инициализация игрового поля (вызов функции **init\_field ()**)
3. инициализация змейки (вызов функции **init\_snake ()**)
4. [ инициализация еды (вызов функции **init\_food ()**) ]
5. “Установка” змейки в поле игры (вызов функции **set\_snake ()**)

### Алгоритм функции exec\_game (один из возможных)

В цикле, пока идет игра выполняется алгоритм одного шага

1. Установка еды на поле (вызов функции **set\_food ()**)
2. Очистка старых позиций змейки в поле (вызов функции **clear\_snake ()**)
3. Проверка не съела ли еду змейка (вызов функции **check\_eating ()**)
4. Передвижение змейки (вызов функции **move\_snake ()**)
5. Проверка не уперлась ли змейка в граничный элемент (вызов функции **check\_snake ()**)
6. Установка новых позиций змейки в поле игры (вызов функции **set\_snake ()**)
7. Вывод поля игры на экран (вызов функции **print\_field ()**)
8. Проверка завершения игры (вызов функции **check\_game ()**)
9. Временная задержка (**Sleep** ())
10. Обработка пользовательского ввода

### Алгоритм функции main (один из возможных)

1. Выполняются инициализирующие действия для организации графического вывода:

* Запуск процесса **demo.exe** (вызов функции из графической библиотеки **StartDemo ()**)
* Задаются размерности динамического двухмерного массива - поля игры (вызов функции из графической библиотеки **SetDimensions ()**)
* Определяется динамический двумерный массив

1. создается экземпляр структуры (объект) **Game**
2. производится инициализация игры (вызов **init\_game ()**)
3. производится запуск игры (вызов **exec\_game ()**).
4. После завершения игры выполняется завершение процесса (вызов функции из графической библиотеки **FinishDemo ()**).

### Алгоритм функции clear\_field [модуль game]

* Поле игры очищается (заполняется шариками цвета EMPTY)

### Алгоритм функции print\_field [модуль field]

Этот алгоритм полностью заменяется.

Вместо печати двумерного массива поля требуется вывести массив в графическом интерфейсе с помощью функции из графической библиотеки **drawBalls ()**.

Приложение

Пример использования функций из графической библиотеки

#include "GraphicsLib.h"

int main()

{

//Запуск процесса demo.exe

StartDemo();

std::cout << "Press any key\n";

std::cin.get();

//1) Задайте размерности n и m динамического двумерного массива

// (или предложите ввести их пользователю)

int n = ..., m = ...;

SetDimensions(n, m);

// Определение динамического двумерного массива (1-й способ)

//2) Определите динамический двумерный массив

// (используйте 1-й способ, когда под двумерный массив выделяется монолитный фрагмент памяти,

// так как обмен данными с рисующим приложением предполагает непрерывность передаваемых данных)

int \*p = ...;

//3) для работы с двумерным массивом удобнее использовать вспомогательный массив

int \*\*pp = ...;

//

//4) Cформирмируйте начальные значения двумерного массива

//...

//5) Отрисовка шариков

DrawBalls(p, n, m);

//6) можно определить задержку отображения

pause(1000);

//7) Здесь приведен пример отображения красного шарика в пределенных координатах массив

//pp[4][4] = RED;

//DrawBalls(p, n, m);

//8) ) Завершение процесса работы

std::cout << "FinishDemo (нажмите Enter)\n";

std::cin.get();

FinishDemo();

}