**Федеральное агентство образования РФ**

**Ивановский государственный энергетический университет**

**Отчёт**

**Дисциплина:**

**Тема: Игра «Змейка»**

**Выполнил студент 3 курса 43 группы**

**Сиваков Глеб**

**Проверила:**

**Иваново 2022**

**Постановка задачи**

Цель игры «Змейка» заключается в том, чтобы игрок смог собрать как можно больше фруктов, тем самым вырастив самую длинную змейку. При прикосновении к фрукту, змейка растут на 1 блок, а фрукт возрождается в другом месте. Чем больше фруктов змейка съела, тем она длиннее. Игра начинается снова, если змейка вышла за пределы поля или коснулась себя. После смерти змейка становится снова маленькой, размером в 1 блок.

**Объектно-ориентированный анализ предметной области**

Для того, чтобы реализовать игру, мне придётся реализовать некоторые классы.

Необходимо начать с класса поля: во-первых, задаётся матрица, в которой каждый элемент отвечает за то, что сейчас находится на данном элементе. Размер окна задаётся размером этой матрицы. Во-вторых, поле будет отвечать за пересечение объектов на нём, создавать фрукт и задавать параметры рисования.

Далее можно поговорить о змейке. С помощью функции, которой будет выполняться каждый кадр, змейка может передвигаться, а также спрашивать у поля с чем она пересекается в данный момент. В зависимости от пересечений определяется поведение игры. Также в этом классе определено последнее направление для блока и сами блоки.

Помимо всего я реализовал класс для отрисовки. В этом классе сохраняется дескриптор окна. В окне отрисовщик может показывать квадрат(змейка), круг(фрукт) и закрашивать поле.

Реализован класс игры. Он принимает последний ввод с клавиатуры, вызывает функцию рисования, а также создаёт нужные объекты.

**Реализация объектов**

**SnakeProj**

Окно, в котором происходит игра

**Field**

**public:**

enum { WIDTH = 500, HEIGHT = 500 }; - размеры поля и размеры экрана соответственно

enum { BLOCK\_WIDTH = 1, BLOCK\_HEIGHT = 1 }; - размеры ограничивающего блока

enum Type { EMPTY, SNAKE\_BLOCK, FRUIT }; - состояния элемента поля

Field(); - конструктор класса, заполняет поле пустым типов, а также вызывает функцию newFruit(), тем самым создаёт новый фрукт на поле

void setBlock(Type type, int x, int y); - устанавливает текущее состояние Type элемента x y поля

Type block(int x, int y) const; - проверяет пересечения передаваемого значения с краем поля и фруктом. Для того, чтобы определить пересечение с фруктом она просматривает поле относительно передаваемых x и y в небольшом радиусе. Сделано это с целью соотнесения размеров пересечения фрукта с размерами змейки.

void draw(Painter&) const; - обходит все элементы поля, в зависимости от элемента находящимся на текущей итерации происходит его отрисовка. То есть вызываются функции отрисовщика.

void newFruit(); - создаёт новый фрукт

**private:**

Type m\_[HEIGHT][WIDTH]; - матрица типов поля

int xfruit, yfruit; - положения фрукта на поле

**Snake**

**public:**

enum Direction { LEFT, UP, RIGHT, DOWN }; - возможные направления змейки

Snake(); - конструктор змейки. При создании объекта задаёт ему рандомное направление и устанавливает скорость в зависимости от размеров окна.

bool tick(Field&); - функция, которая выполняется каждый кадр. Вычитает координаты в зависимости от его направления, а также проверяет все варианты его столкновения (с собой, границами, фруктом)

void keyEvent(Direction); - устанавливает направление змейки.

**private:**

typedef std::list<std::pair<int, int> > Blocks; - блоки змейки

Blocks blocks\_;

Direction direction\_; - направление змейки

Direction lastMove\_; - последнее движение змейки

int Speed; - скорость змейки

**Painter**

**public:**

Painter(HDC hdcWindow); - конструктор, в котором сохраняется дескриптор окна

void bar(int x1, int y1, int x2, int y2); - отрисовка квадрата

void circle(int x, int y, int radius); - отрисовка круга

void background(int x1, int y1, int x2, int y2); - отрисовка заднего фона

**private:**

HDC hdc; - сохранённый дескриптор

**Game**

**public:**

void tick(); - функция, которая проверяет наличие объектов на экране

void draw(Painter&) const; - вызывает функцию отрисовки у поля, выполняется каждый кадр

void keyEvent(Snake::Direction); - принимает нажатие клавиш, вызывает функцию изменения направления у змейки

**private:**

Field field\_; - экземпляр объекта поля

Snake snake\_; - экземпляр объекта змейки

**Инструкции по использованию**

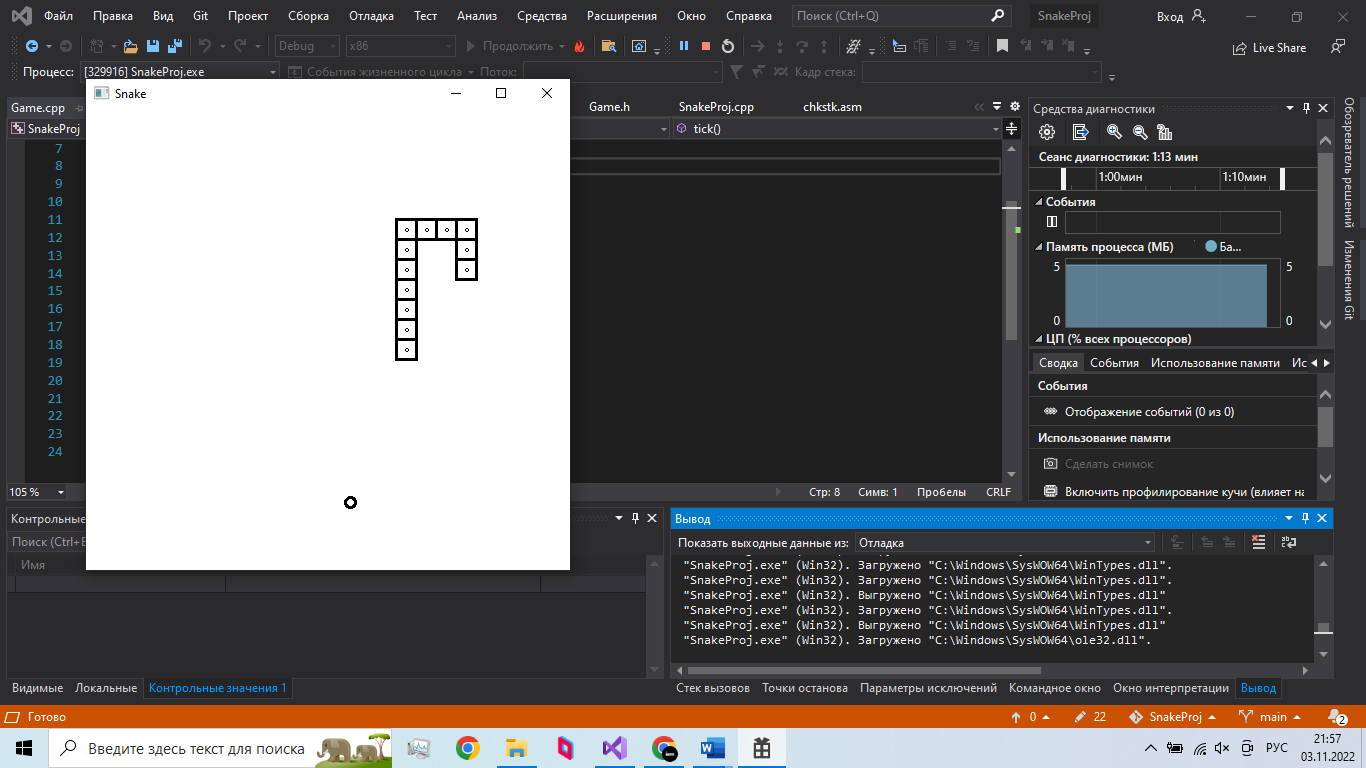


Рис 1. Демонстрация приложения

Для того, чтобы управлять змейкой необходимо нажимать стрелочки на клавиатуре. Стрелка вверх заставляет змею передвигаться вверх, влево – заставляет передвигаться влево, вправо и вниз соответственно. Если змея достигнет фрукта, то к ней добавится один блок, змея станет длиннее.