Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
Кафедра ПМиК

КУРСОВАЯ РАБОТА

по дисциплине «Объектно-ориентированное программирование»

Тема: «Реализация игры «крестики-нолики» в графическом режиме»

Выполнил:
ст. гр. ИВ-921
Ярошев Роман Александрович
Проверил:
доцент кафедры ПМиК
Ситняковская Е. И.

Новосибирск, 2020

Содержание

Постановка задачи

Реализовать игру «крестики-нолики» в графическом режиме.

Технологии ООП

- Инкапсуляция (все поля данных не доступны из внешних функций)
- Наследование (минимум 3 класса, один из которых - абстрактный)
- Полиморфизм
- Конструкторы
- Перегрузка конструкторов
- Списки инициализации
- виртуальные функции
- параметры по умолчанию

Структура классов

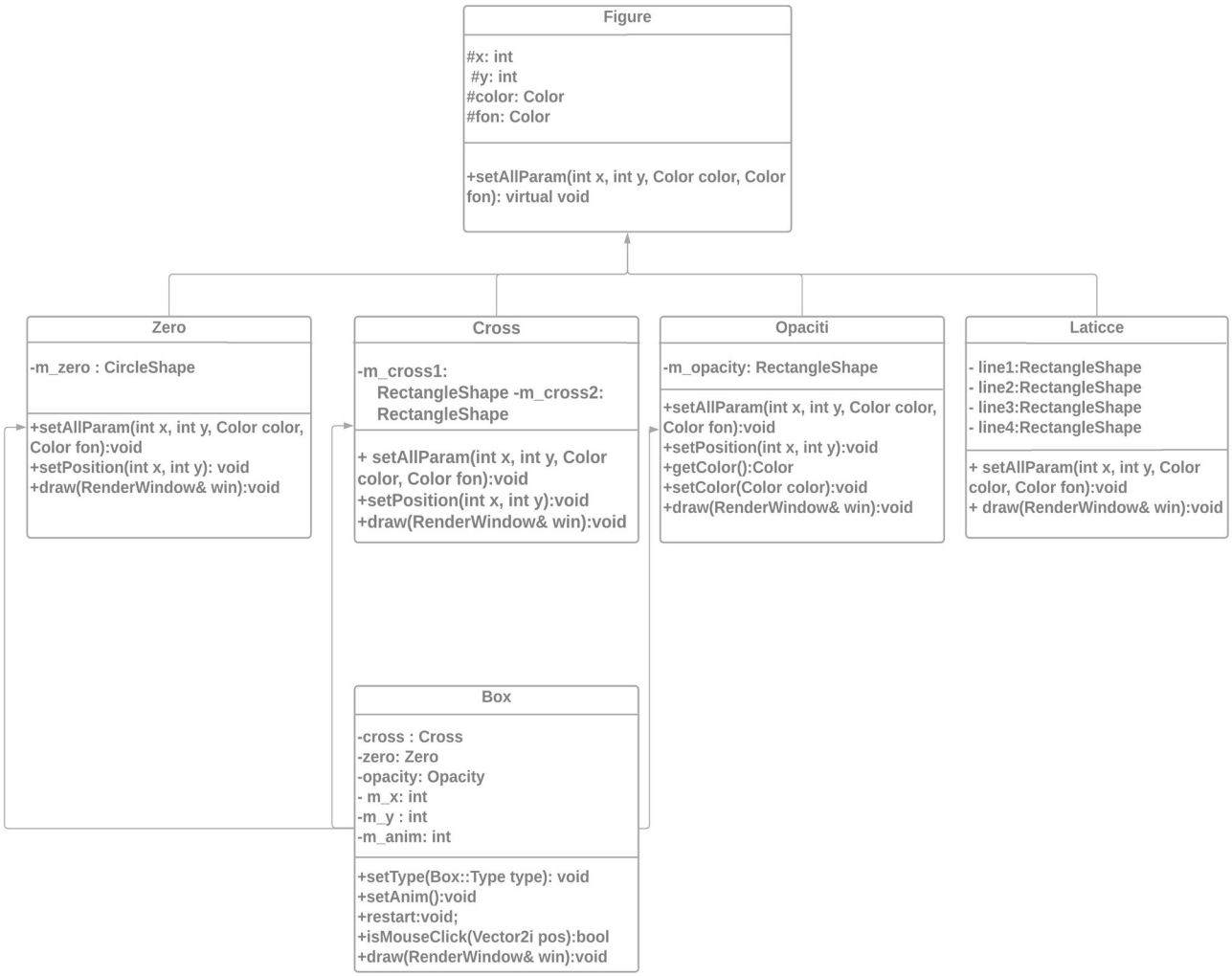

Рис 1. Схема UML

Класс Box связан отношением ассоциации с классами Zero, Cross, Opacciti т.к объект типа Box использует объекты типов Zero, Cross, Opacciti.

Класс Figure связан отношением наследование с классами Zero, Cross, Opacciti, Laticee. Figure – является базовым классом, а Zero, Cross, Opacciti, Laticee наследники от Figure.

Программная реализация

Классы

Box – класс, отвечающий за создание типа ячейки

Box::Box() – конструктор для первой инициализации со списком инициализации

Box::Box(int x, int y, Box::Type type) : m_x{ x }, m_y{ y }, m_type{ type }, m_anim{ 0 } - конструктор, нужный для использования в main.cpp для инициализации в массив ячеек со списком инициализации

Основные методы

setType(Box::Type type) – присваивает тип ячейки

Box::Type getType() – возвращает тип ячейки

bool Box::isMouseClick(Vector2i pos) – проверяет нажатие на данную ячейку

Figure – базовый класс, для классов Cross, Zero, Opacity, Laticce

Figure() : x{ 0 }, y{ 0 } { }, Figure(int x, int y) : x{ x }, y{ y } { }

- конструкторы со списком инициализации

Основные функции

virtual void setAllParam(int x, int y, Color color, Color fon) = 0; виртуальный и абстрактный метод заполнения параметров, для наследников

Cross – класс, создающий фигуру крестик

Zero - класс, создающий фигуру нолик

Opacity - класс, создающий фигуру для мерцания

Laticce - класс, создающий поле-решётку

Menu – класс, для создания меню, где происходит выбор фигуры

GameState – класс перечисления для создания состояния игры

main.cpp

Основные переменные:

GameState game_state – объект класса перечисления, отвечающий за то, чтобы выполнялся нужный фрагмент кода

Texture texture – объект класса, нужный для картинки фона

Sound sound - объект класса, нужный для звука нажатие мышки

Music sound_back – объект класса, нужный для музыки на фон

const int SET_BOX[8][3]{

{0, 1, 2},

{3, 4, 5},

{6, 7, 8},

{0, 3, 6},

{1, 4, 7},

{2, 5, 8},

{0, 4, 8},

{2, 4, 6}

}; - массив, хранящий в себе номера ячеек выигрышных случаев.

// ===

Laticce lattice - объект класса решетка, отвечающий за то, чтобы реализовать поле.

Box::Type user_box, Box::Type player_box – объекты, отвечающие за хранение типа фигуры в данной ячейки

int event_time – время для мерцания

vector<Box> vector_box – массив, хранящий в себе 9 полей решетки

Menu menu - объект, отвечающий за реализацию меню (выбор фигуры, которой пользователь хочет пользоваться)

Алгоритм работы программы

1. Заполнение массива решетки `vector_box` пустыми значениями(default) с помощью параметра по умолчанию и обозначение сразу всех возможными значений(zero, cross,opaciti) .
2. Заход в цикл while, который длится пока окно не закрыто, пока пользователь не закрыл его, нажав на крестик.
3. С помощью оператора switch код условно делится на несколько частей, а именно на меню, на саму игру, а и на конец игры – мигание полей.
4. С помощью объекта класса Menu вызывается его метод `setParam`, который переводит нас на фрагмент кода с реализацией самой игры, запоминает выбор фигуры(x или 0), которой хочет играть пользователь
5. Переход на реализацию самой игры. Первый условный оператор отвечает за ход пользователя. Если там свободное место(а это проверяется с помощью метода getType класса Box, который возвращает тип поля(x, 0 или пустое значение), и если не было выхода за поля и было нажато на данное поле, то ставится в массив тот тип, который был выбран пользователем
6. Далее идет проверка на ничью или на победу одного из игроков, если что-то одно выполнилось идет конечный фрагмент кода с анимацией(подсвечиванием) и снова переход на фрагмент кода меню
7. Следующий if ищет количество свободных ячеек, и заполняет массив arr значениями индекса свободной ячейки
8. Далее реализация ии. Он ходит в зависимости от count - количеству свободных ячеек. Первый всегда ходит пользователь. Если свободно 8 ячеек, он походит рандомно в одну из ячеек придавая тип полю, с помощью метода setType. Если свободно 6 ячеек, он с помощью двумерного массива SET_BOX, хранящего в себе победные заполненные поля, сначала просмотрит нужно ли защититься(т.е остался ли пользователю один ход до победы) , если нужно - защититься, а если нет с помощью того же массива просчитает победные ситуации(относительно его ячейки, где у него уже есть своя фигура, просмотрит свободны ли еще 2 ячейки для выигрыша) и поставит в новое выбранное поле свою фигуру. Если осталось 4 свободные ячейки также проверит нужно ли защититься, если нет также будет искать победные ситуации только уже с учетом того, что на победной линии у него 2 заполненных поля, а свободная ячейка 1, если таких нет, то будет искать где у него одно заполненное поле в победной линии и 2 свободные ячейки. Если 2 свободных поля, он сначала

проверит нужно ли защищаться, если нет, проверит нужно ли поставить последний свой знак для выигрыша, и если не то не другое(т.е точно ничья в люобом случае) поставит в случайное свободное место

Результаты работы

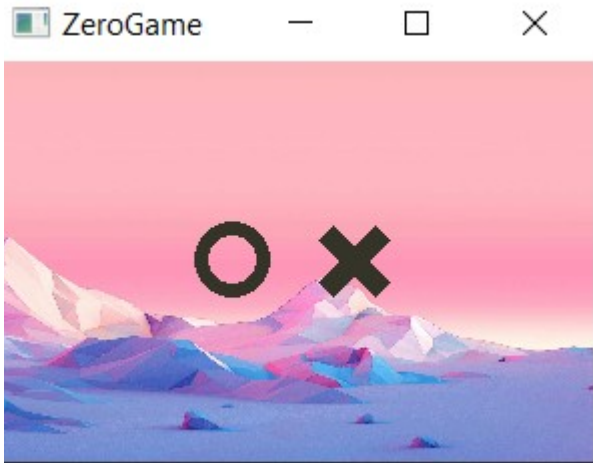

Рис.2 Меню

Рис.3 Игра

Рис.4 Победа компьютера

Рис.5 Ничья

Вывод

В результате выполнения данного курсового проекта была разработана игра «Крестики нолики». С помощью библиотеки sfml игра была реализована в графическом режиме. Также были реализованы логика игры, ход пользователя, искусственный интеллект, который является противником пользователя, а также результат игры относительно пользователя(победа, проигрыш, ничья).. Вся программа реализовывалась на принципах ООП. Использовался принцип наследования: создавался базовый класс, который хранил в себе переменные и методы, необходимые в классах потомках для их дальнейшей реализации. Также использовались: полиморфизм с помощью создания виртуального метода, для того чтобы в классах потомках реализация такого метода отличалась, инкапсуляция - у всех полей классов спецификаторы доступа были private, перегрузка конструкторов - путем создание нескольких конструкторов для переопределения переменных в нужный момент, списки инициализации в конструкторе – для того, чтобы сделать конструктор наследуемым, параметры по умолчанию - чтобы вызывать свой конструктор не трогая 3 параметр, который постоянен.

Листинг кода

main.cpp

```
#include <iostream>
#include <SFML/Graphics.hpp>
#include <SFML/Audio.hpp>
#include <vector>
#include "Box.h"
#include "Laticce.h"
#include "Menu.h"
#include "GameState.h"
using namespace sf;
using namespace std;

int main()
{

                case GameState::MENU:
                {
                        menu.setParam(game_state, user_box, win);
                }
                break;
                case GameState::RUN:
                {
                        if (player_box == user_box && Mouse::isButtonPressed(Mouse::Left)) {
                                Vector2i pos = Mouse::getPosition(win);
                                for (int i = 0, n = vector_box.size(); i < n; i++) {
                                        if (vector_box[i].getType() == Box::DEFAULT && vector_box[i].isMouseClick(pos)) {
                                                vector_box[i].setType(user_box);
                                                player_box = (player_box == Box::ZERO) ? Box::CROSS : Box::ZERO;
                                        }
                                }
                        }

                        // Full box
                        bool full_box = true;
                        for (int i = 0, z = 1, c = 1; i < 8; i++) {
                                for (int j = 0; j < 3; j++) {
                                        if (vector_box[SET_BOX[i][j]].getType() != Box::CROSS) c = 0;
                                        if (vector_box[SET_BOX[i][j]].getType() != Box::ZERO) z = 0;
                                        if (vector_box[SET_BOX[i][j]].getType() == Box::DEFAULT) full_box = false;
                                }
                                if (z == 1 || c == 1) {
                                        for (int j : SET_BOX[i])
                                                vector_box[j].setAnim();
                                        game_state = GameState::ANIM;
                                        event_time = 120;
                                        full_box = false;
                                        break;
                                }
                                z = c = 1;
                        }

                        if (full_box) {
                                for (Box& box : vector_box) {
                                        box.setAnim();
                                }
                                game_state = GameState::ANIM;
                                event_time = 120;
                        }

                        // PC
                        if (player_box != user_box) {
                                int arr[9];
                                int count = 0;
                                for (int i = 0, n = vector_box.size(); i < n; i++) {
                                        if (vector_box[i].getType() == Box::DEFAULT) {
                                                arr[count++] = i;
                                        }
                                }
```

```
                                        // logic pc game
                                        switch (count) {
                                        case 2:
                                        {
                                                int flag = 0;
                                                for (int i = 0; i < 8; i++) {
                                                        int user_value = 0, pc_value = 0, default_value = -1; // пк не между
противника

                                                        for (int j = 0; j < 3; j++) {
                                                                if (vector_box[SET_BOX[i][j]].getType() == user_box)
{ user_value++; }

                                                                if (vector_box[SET_BOX[i][j]].getType() == player_box)
{ pc_value++; }

                                                                if (vector_box[SET_BOX[i][j]].getType() == Box::DEFAULT)
{ default_value = SET_BOX[i][j]; }
                                                        }
                                                        printf("default = %d\nuser_value = %d\npc_value = %d\n\n", default_value,
user_value, pc_value);

                                                        if (user_value == 0 && pc_value == 2 && default_value != -1) {
                                                                vector_box[default_value].setType(player_box);
                                                                flag = 1;
                                                                break;
                                                        }
                                                }
                                                if (flag == 0) {
                                                        for (int i = 0; i < 8; i++) {
                                                                int user_value = 0, pc_value = 0, default_value = -1; // пк не между
противника

                                                                for (int j = 0; j < 3; j++) {
                                                                        if (vector_box[SET_BOX[i][j]].getType() == user_box)
{ user_value++; }

                                                                        if (vector_box[SET_BOX[i][j]].getType() ==
player_box) { pc_value++; }

                                                                        if (vector_box[SET_BOX[i][j]].getType() ==
Box::DEFAULT) { default_value = SET_BOX[i][j]; }
                                                                }
                                                                printf("default = %d\nuser_value = %d\npc_value = %d\n\n",
default_value, user_value, pc_value);

                                                                if (user_value == 2 && pc_value == 0 && default_value != -1) {
                                                                        vector_box[default_value].setType(player_box);
                                                                        flag = 2;
                                                                        break;
                                                                }
                                                        }
                                                }
                                                if (flag == 0) {
                                                        vector_box[arr[static_cast<int>(rand() % count)]].setType(player_box);
                                                }
                                        }
                                        break;

                                        case 4:
                                        {
                                                int flag = 0;
                                                for (int i = 0; i < 8; i++) {
                                                        int user_value = 0, pc_value = 0, default_value = -1; // пк не между
противника

                                                        for (int j = 0; j < 3; j++) {
                                                                if (vector_box[SET_BOX[i][j]].getType() == user_box)
{ user_value++; }

                                                                if (vector_box[SET_BOX[i][j]].getType() == player_box)
{ pc_value++; }

                                                                if (vector_box[SET_BOX[i][j]].getType() == Box::DEFAULT)
{ default_value = SET_BOX[i][j]; }
                                                        }
                                                        printf("default = %d\nuser_value = %d\npc_value = %d\n\n", default_value,
user_value, pc_value);

                                                        if (user_value == 0 && pc_value == 2 && default_value != -1) {
                                                                vector_box[default_value].setType(player_box);
                                                                flag = 1;
                                                                break;
                                                        }
                                                }
                                                if (flag == 0) {
                                                        for (int i = 0; i < 8; i++) {
                                                                int user_value = 0, pc_value = 0, default_value = -1; // пк не между
противника

                                                                for (int j = 0; j < 3; j++) {
```

```
                                                                        if (vector_box[SET_BOX[i][j]].getType() == user_box)
{ user_value++; }
                                                                        if      (vector_box[SET_BOX[i][j]].getType()      ==
player_box) { pc_value++; }
                                                                        if      (vector_box[SET_BOX[i][j]].getType()      ==
Box::DEFAULT) { default_value = SET_BOX[i][j]; }
                                                                }
                                                                if (user_value == 2 && pc_value == 0 && default_value != -1) {
                                                                        vector_box[default_value].setType(player_box);
                                                                        flag = 2;
                                                                        break;
                                                                }
                                                        }
                                                }
                                                if (flag == 0) {
                                                        for (int i = 0; i < 8; i++) {
                                                                int user_value = 0, pc_value = 0, default_value = -1; // пк не между
противника
                                                                for (int j = 0; j < 3; j++) {
                                                                        if (vector_box[SET_BOX[i][j]].getType() == user_box)
{ user_value++; }
                                                                        if      (vector_box[SET_BOX[i][j]].getType()      ==
player_box) { pc_value++; }
                                                                        if      (vector_box[SET_BOX[i][j]].getType()      ==
Box::DEFAULT) { default_value = SET_BOX[i][j]; }
                                                                }
                                                                if (user_value == 0 && pc_value == 1 && default_value != -1) {
                                                                        vector_box[default_value].setType(player_box);
                                                                        break;
                                                                }
                                                        }
                                                }

                                        }
                                        break;

                                        case 6:
                                        {
                                                int flag = 0;
                                                for (int i = 0; i < 8; i++) {
                                                        int user_value = 0, pc_value = 0, default_value = -1; // пк не между
противника
                                                        for (int j = 0; j < 3; j++) {
                                                                if      (vector_box[SET_BOX[i][j]].getType()      ==      user_box)
{ user_value++; }
                                                                if      (vector_box[SET_BOX[i][j]].getType()      ==      player_box)
{ pc_value++; }
                                                                if      (vector_box[SET_BOX[i][j]].getType()      ==      Box::DEFAULT)
{ default_value = SET_BOX[i][j]; }
                                                        }
                                                        if (user_value == 2 && pc_value == 0 && default_value != -1) {
                                                                vector_box[default_value].setType(player_box);
                                                                flag = 1;
                                                                break;
                                                        }
                                                        if (user_value == 2 && pc_value == 1) {
                                                                flag = 2;
                                                                break;
                                                        }
                                                }
                                                if (flag == 2 || flag == 0) {
                                                        for (int i = 0; i < 8; i++) {
                                                                int user_value = 0, pc_value = 0, default_value = -1; // пк не между
противника
                                                                for (int j = 0; j < 3; j++) {
                                                                        if (vector_box[SET_BOX[i][j]].getType() == user_box)
{ user_value++; }
                                                                        if      (vector_box[SET_BOX[i][j]].getType()      ==
player_box) { pc_value++; }
                                                                        if      (vector_box[SET_BOX[i][j]].getType()      ==
Box::DEFAULT) { default_value = SET_BOX[i][j]; }
                                                                }
                                                                printf("default = %d\nuser_value = %d\npc_value = %d\n\n",
default_value, user_value, pc_value);
                                                                if (user_value == 0 && pc_value == 1 && default_value != -1) {
                                                                        vector_box[default_value].setType(player_box);
                                                                        break;
                                                                }
```

```
                                                }
                                        }
                                }
                                break;

                                case 8:
                                {
                                        vector_box[arr[static_cast<int>(rand() % count)]].setType(player_box);
                                }
                                break;
                                default:
                                        break;
                                }
                                player_box = user_box;
                        }
                }
                break;
                case GameState::ANIM:
                {
                        if (event_time == 0) {
                                for (Box& box : vector_box) {
                                        box.restart();
                                }
                                game_state = GameState::RUN;
                        }
                        else {
                                event_time--;
                        }
                }
                break;
                }

                win.clear(Color(206, 230, 242));
                win.draw(sprite);
                if (game_state == GameState::MENU) {
                        menu.draw(win);
                }
                else {
                        laticce.draw(win);
                        for (Box& box : vector_box) {
                                box.draw(win);
                        }
                }

                win.display();
        }

        return 0;
}
```

Box.h

```
#pragma once
#include <SFML/Graphics.hpp>
#include "Cross.h"
#include "Zero.h"
#include "Opacity.h"
using namespace sf;

class Box
{
public:
        enum Type
        {
                DEFAULT, CROSS, ZERO
        };

private:
        Type m_type;

        Cross cross;
        Zero zero;
        Opacity opacity;

        int m_x, m_y, m_anim;

public:
```

```
        Box();
        Box(int x, int y, Box::Type type = Box::DEFAULT);
        Box::Type getType() const;
        void setType(Box::Type type);
        void setAnim();
        void restart();

        bool isMouseClick(Vector2i pos);

        void draw(RenderWindow& win);
        ~Box();
};
```

Cross.h

```
#pragma once
#include "Figure.h"
#include <SFML/Graphics.hpp>
using namespace sf;

class Cross :
        public Figure
{
private:
        RectangleShape m_cross1;
        RectangleShape m_cross2;
public:
        Cross();
        void setAllParam(int x, int y, Color color, Color fon);
        void setPosition(int x, int y);
        void draw(RenderWindow& win);
};
```

Figure.h

```
#pragma once
#include <SFML/Graphics.hpp>
using namespace sf;
class Figure
{
protected:
        int x, y;
        Color color, fon;
public:
        Figure() : x{ 0 }, y{ 0 } { }
        Figure(int x, int y) : x{ x }, y{ y } { }
        Figure(int x, int y, Color color, Color fon) : x{ x }, y{ y }, color{ color }, fon{ fon } { }
        Figure(Color color, Color fon) :x{ 0 }, y{ 0 }, color{ color }, fon{ fon } { }
        virtual void setAllParam(int x, int y, Color color, Color fon) = 0;
        ~Figure();
};
```

Opacity.h

```
#pragma once
#include "Figure.h"
#include <SFML/Graphics.hpp>
using namespace sf;

class Opacity :
        public Figure
{
private:
```

```
        RectangleShape m_opacity;
public:
        Opacity();
        void setAllParam(int x, int y, Color color, Color fon);
        void setPosition(int x, int y);
        Color getColor();
        void setColor(Color color);
        void draw(RenderWindow& win);
        ~Opacity();
};
```

Zero.h

```
#pragma once
#include "Figure.h"
#include <SFML/Graphics.hpp>
using namespace sf;

class Zero :
        public Figure
{
private:
        CircleShape m_zero;
public:
        Zero();
        void setAllParam(int x, int y, Color color, Color fon);
        void setPosition(int x, int y);
        void draw(RenderWindow& win);
        ~Zero();
};
```