МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт кибернетики

Кафедра высшей математики

КУРСОВАЯ РАБОТА
по дисциплине
«Языки и методы программирования»

Тема курсовой работы
«Аквариум 2D»

Студент группы КМБО-01-18 *Терехов Т.А.*

Руководитель курсовой работы *Шерстнев Е.В.* к.ф.-м.н., доцент

Работа представлена к защите «1» 10 202 0 г. *(подпись студента)*

«Допущен к защите» «1» 10 202 0 г. *(подпись руководителя)*

МОСКВА — 2020

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт кибернетики

Кафедра высшей математики

Утверждаю

Заведующий кафедрой _____ *Ю.И.Худак*

«__1__» ____09____ 2020г.

ЗАДАНИЕ
на выполнение курсовой работы
по дисциплине «Языки и методы программирования»

Студент *Терехов Т.А.* Группа *КМБО-01-18*

1. **Тема: «Аквариум 2D»**

2. **Исходные данные:**
Реализовать имитацию рыбок в аквариуме. Пользователь задает размер аквариума, количество рыбок и количество видов. Рыбки плавают с произвольной (меняющейся скоростью), направление движения меняется либо при встрече с аквариумом, либо с другой рыбой, либо случайным образом.

3. **Перечень вопросов, подлежащих разработке, и обязательного графического материала:**
1)Реализовать несколько видов рыб, различающихся размером и средней скоростью.
2)Добавить Хищных рыб, при встрече с которыми более мелкие пожираются.

4. **Срок представления к защите курсовой работы: до** «__1__» ____12____ 2020 г.

Задание на курсовую работу выдал «__1__»__09__2020 г. _____ (Щуплов В.Р.)

Задание на курсовую работу получил «__1__»__09__2020 г. _____ (Терехов Т.А.)

Оглавление

Введение

Целью курсовой работы является получение навыков самостоятельной разработки программного продукта в соответствии с принципами объектно-ориентированного и структурного программирования, рассмотренными в процессе изучения дисциплины.

Для демонстрации навыков был реализован аквариум с различным количеством рыб , в том числе и хищных.

Задачи, которые следует решить в данной курсовой следующие:

1)Реализовать задание пользователем размера аквариума, количество рыбок и количество видов.

2)Реализовать изменение скорости и направления движения рыб либо при встрече с аквариумом, либо с другой рыбой, либо случайным образом.

3) Реализовать несколько видов рыб, различающихся размером и средней скоростью.

4) Добавить хищных рыб, при встрече с которыми более мелкие пожираются.

Для того чтобы продемонстрировать решение данных задач необходимо использовать графическую библиотеку . В своей курсовой работе я использовал графическую библиотеку TX Library. TX Library - компактная графическая библиотека для Win32 на C++. Это небольшая "песочница" для начинающих реализована с целью помочь им в изучении простейших принципов программирования. Документация на русском языке. Что и является ее основным плюсом.

Описание классов

Основным классом является класс fish. Данный класс описывает основные свойства рыбы и её поведения.

```
class fish {
public:
    int size;
    int x,y;
    int vx;
    int vy;
    int vcr;
    COLORREF color;
    bool life;
```

Были реализованы следующие свойства:

Size – задаёт размер рыбы

X,y – координаты рыбы в пространстве

Vx – Скорость рыбы по X(вдоль оси OX)

Vy - Скорость рыбы по Y(вдоль оси OY)

Vcr – Максимальная скорость

Color – Цвет рыбы

Life – состояния (жива – true , мертва – false)

Методы , описанные ниже, описывают все движения и взаимодействия рыб.

Ниже представлена **перегрузка метода** vzaimodeictvie

```
void vzaimodeictvie(int xmax,int ymax) //   описывает перемещение рыбки в следующий момент времени, не учитывая других рыб.
{
    if(!life)return;
    x+=vx;
    y+=vy;
    if(rand()%25==0) // шанс в 4 процента, что рыба поменяет скорость и траекторию
    {
        vx=rand()%vcr-vcr/2;
        vy=rand()%vcr-vcr/2;

    }
    if(x+2*size>xmax){ // удар об правую стенку
        x=xmax-2*size;
        vx=-vx;
    }

    if(y+2*size>ymax){// удар об низ
        y=ymax-2*size;
        vy=-vy;
```

```
    }

    if(x-2*size<0){ // удар об левый край
        x=2*size;
        vx=-vx;
    }

    if(y-2*size<0){  //удар об верхний край аквариума
        y=2*size;
        vy=-vy;
    }

}
```

Данный метод просчитывает движение рыбы и её взаимодействие со стенками аквариума, в случае касание стены произойдёт отталкивание от стены

```
void vzaimodeictvie(fish *A) // описывает взаимодействие данной рыбы с рыбой которая во входных парамантрах
{
    if(! life || ! (*A).life)return;
    int Ax=(*A).x;
    int Ay=(*A).y;
    int Asize=(*A).size;
    if((x-Ax)*(x-Ax)+(y-Ay)*(y-Ay)<2*(size+Asize)*(size+Asize)) // если суммарный размер рыб больше чем расстояние их центров, то они отталкиваются в противоположные друг от друга направления

    {
        vx = ((int)(1.0*vcr*(x-Ax)/sqrt(1.0*(x-Ax)*(x-Ax)+(y-Ay)*(y-Ay))));
        vy = ((int)(1.0*vcr*(y-Ay)/sqrt(1.0*(x-Ax)*(x-Ax)+(y-Ay)*(y-Ay))));
        (*A).vx=-vx;
        (*A).vy=-vy;
        x+=2*vx;
        y+=2*vy;
        (*A).x-=2*vx;
        (*A).y-=2*vy;

    }
}
```

Данный метод реализует взаимодействие травоядных рыб друг с другом . а именно при их столкновении они будут отталкиваться друг от друга.

```
class predatory_fish: public fish
{
public:
    predatory_fish(){}
    predatory_fish(int vx_vcr,int vx_size,int vx_x, int vx_y)
    {
        size=vx_size;
        color=RGB(255,0,0);
        vcr=vx_vcr;
        x=vx_x;
        y=vx_y;
        vx=rand()%vcr+1;
        vy=rand()%vcr+1;
        life=true;
    }
```

Класс predatory_fish является наследуемым по отношению к классу fish.

```
void eating(fish *A) //описывает взаимдействие хищной рыбы с травоядной
{
    int Ax=(*A).x;
    int Ay=(*A).y;
    int Asize=(*A).size;

    if(Asize*1.5 < size && (x-Ax)*(x-Ax)+(y-Ay)*(y-Ay)<2*(size+Asize)*(size+Asize))
        (*A).life=false;
}
```

В нём реализован новый метод eating ,который взаимодействие хищной рыбы с травоядной и в случае близости и значительного превосходства(более чем в 1.5 раза) размеров хищной рыбы по сравнению с травоядной, последняя съедается.

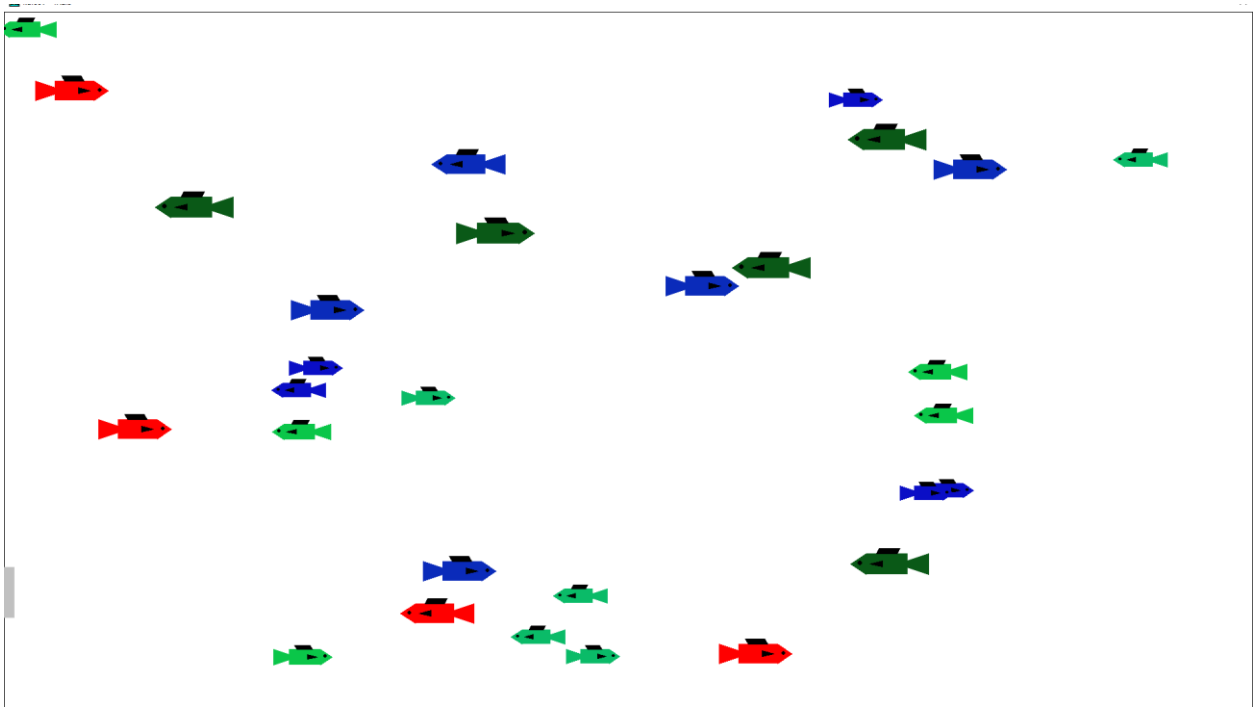
```
template <typename T>
void otricovka(T value)
```

Была реализована шаблон функция отрисовки рыб. Она принимает на вход объекты двух различных классов, тем самым происходит отрисовка травоядных, так и хищных рыб.

Тестирование

```
Vvedite x:1920

Vvedite y:1080

Vvedite kol-vo vidov:5

Vcego 0 ribok, Vvedite kol-vo ribok 1 ogo vida :5

Vcego 5 ribok, Vvedite kol-vo ribok 2 ogo vida :5

Vcego 10 ribok, Vvedite kol-vo ribok 3 ogo vida :5

Vcego 15 ribok, Vvedite kol-vo ribok 4 ogo vida :5

Vcego 20 ribok, Vvedite kol-vo ribok 5 ogo vida :5

Vvedite kol-vo xicshix ribok:2
```

Ввод данных производится через консоль, а именно размер аквариума (X и У) ,количество видов травоядных рыб и хищных.



После чего создаётся окно, где происходит отрисовка всего аквариума.

Для того чтобы завершить программу необходимо нажать клавишу ESC

Заключение

В результате выполнения курсовой работы были получены навыки разработки программ на языке программирования C++. Были реализованы перегрузка метода**,** шаблон функция и наследование классов, а также реализована несложная работа с графикой.

Список литературы

1) Страуструп, Б. Язык программирования C++: Специальное издание/ Б. Страуструп; Пер. с англ. Н.Н. Мартынов. - М.:БИНОМ, 2017. - 1136 с.

2) Ашарина, И.В. Основы программирования на языках С и С++: Курс лекций для высших учебных заведений/ И.В. Ашарина - М.: Гор. Линия-Телеком, 2018. - 208 с.

3) Гергель,В.П. Современные языки и технологии паралелльного программирования: Учебник / В.П. Гергель. — М.: МГУ, 2016. — 408 с.

Примечание

```
#include <iostream>
#include "TXLib.h"
#include "stdio.h"
using namespace std;


class fish {
public:
    int size;
    int x,y;
    int vx;
    int vy;
    int vcr;
    COLORREF color;
    bool life;


    fish(){}
    fish(COLORREF vx_color,int vx_vcr,int vx_size,int vx_x, int vx_y)
    {
        size=vx_size;
        color=vx_color;
        vcr=vx_vcr;
        x=vx_x;
        y=vx_y;
        vx=rand()%vcr-vcr/2;// вычитаем vcr/2 для того чтобы был диапазон
траектория(т.е влево если -vcr/2, вправо если + значение)
        vy=rand()%vcr-vcr/2;
        life=true;
    }


    void vzaimodeictvie(int xmax,int ymax) //   описывает перемещение рыбки в
следующий момент времени, не учитывая других рыб.
    {
        if(!life)return;
        x+=vx;
        y+=vy;
        if(rand()%25==0) // шанс в 4 процента, что рыба поменяет скорость и
траекторию
        {
            vx=rand()%vcr-vcr/2;;
            vy=rand()%vcr-vcr/2;
        }
        if(x+2*size>xmax){ // удар об правую стенку
            x=xmax-2*size;
            vx=-vx;
        }

        if(y+2*size>ymax){// удар об низ
            y=ymax-2*size;
            vy=-vy;
        }

        if(x-2*size<0){ // удар об левый край
            x=2*size;
            vx=-vx;
        }
```

```
        if(y-2*size<0){  //удар об верхний край аквариума
            y=2*size;
            vy=-vy;
        }

    }
    void vzaimodeictvie(fish *A) // описывает взаимодействие данной рыбы с рыбой
которая во входных параментрах
    {
        if(! life || ! (*A).life)return;
        int Ax=(*A).x;
        int Ay=(*A).y;
        int Asize=(*A).size;
        if((x-Ax)*(x-Ax)+(y-Ay)*(y-Ay)<2*(size+Asize)*(size+Asize)) // если суммарный
размер рыб больше чем расстояние их центров, то они отталкиваются в противоположные
друг от друга направления
        {
            vx = ((int)(1.0*vcr*(x-Ax)/sqrt(1.0*(x-Ax)*(x-Ax)+(y-Ay)*(y-Ay))));
            vy = ((int)(1.0*vcr*(y-Ay)/sqrt(1.0*(x-Ax)*(x-Ax)+(y-Ay)*(y-Ay))));
            (*A).vx=-vx;
            (*A).vy=-vy;
            x+=2*vx;
            y+=2*vy;
            (*A).x-=2*vx;
            (*A).y-=2*vy;

        }
    }



};

class predatory_fish: public fish
{
public:
    predatory_fish(){}
    predatory_fish(int vx_vcr,int vx_size,int vx_x, int vx_y)
    {
        size=vx_size;
        color=RGB(255,0,0);
        vcr=vx_vcr;
        x=vx_x;
        y=vx_y;
        vx=rand()%vcr+1;
        vy=rand()%vcr+1;
        life=true;
    }
    void eating(fish *A) //описывает взаимдействие хищной рыбы с травоядной
    {
        int Ax=(*A).x;
        int Ay=(*A).y;
        int Asize=(*A).size;

        if(Asize*1.5 < size && (x-Ax)*(x-Ax)+(y-Ay)*(y-
Ay)<2*(size+Asize)*(size+Asize))
            (*A).life=false;
    }


};
```

```
template <typename T>
void otricovka(T value)
{
    if(!value.life)return;
    txSetFillColor (value.color);// цвет фона
    txSetColor (value.color);// цвет контура
    txRectangle ((int)(value.x-2*value.size), (int)(value.y-value.size),
(int)(value.x+2*value.size), (int)(value.y+value.size));
    if(value.vx>0)
    {
        POINT star[3] = {{value.x+2*value.size,value.y+value.size},
{value.x+2*value.size,value.y-value.size}, {value.x+3.5*value.size,value.y}};//лицо
        txPolygon (star, 3);
        POINT star2[3] = {{value.x-4*value.size,value.y+value.size}, {value.x-
4*value.size,value.y-value.size}, {value.x-value.size,value.y}};// плавник задний
        txPolygon (star2, 3);
        txSetFillColor (RGB(0,0,0));
        txSetColor (RGB(0,0,0));
        POINT star3[3] = {{value.x+0.4*value.size,value.y+0.3*value.size},
{value.x+0.4*value.size,value.y-0.3*value.size}, {value.x+1.6*value.size,value.y}};//
плавник центральный
        txPolygon (star3, 3);

        POINT star4[4] = {{value.x-1*value.size,value.y-value.size},
{value.x+1*value.size,value.y-value.size}, {value.x+0.7*value.size,value.y-
1.5*value.size}, {value.x-1.3*value.size,value.y-1.5*value.size}}; // вверхний
параллелограм
        txPolygon (star4, 4);
        txCircle (value.x+2.6*value.size, value.y-0.1*value.size, 0.2*value.size); //
глаз
    }
    else
    {
        POINT star[3] = {{value.x-2*value.size,value.y+value.size}, {value.x-
2*value.size,value.y-value.size}, {value.x-3.5*value.size,value.y}};
        txPolygon (star, 3);
        POINT star2[3] = {{value.x+4*value.size,value.y+value.size},
{value.x+4*value.size,value.y-value.size}, {value.x+value.size,value.y}};
        txPolygon (star2, 3);
        txSetFillColor (RGB(0,0,0));
        txSetColor (RGB(0,0,0));
        POINT star3[3] = {{value.x-0.4*value.size,value.y+0.3*value.size}, {value.x-
0.4*value.size,value.y-0.3*value.size}, {value.x-1.6*value.size,value.y}};
        txPolygon (star3, 3);

        POINT star4[4] = {{value.x-value.size,value.y-value.size},
{value.x+value.size,value.y-value.size}, {value.x+1.3*value.size,value.y-
1.5*value.size}, {value.x-0.7*value.size,value.y-1.5*value.size}};

        txPolygon (star4, 4);
        txCircle (value.x-2.6*value.size, value.y-0.1*value.size, 0.2*value.size);
    }


}



int   main(){
```

```
    int i,j;
    int x,y,n=0,nx,kolvo_vid;// n - кол-во травоядных , nx - кол-во хищных
    printf("VVedite x: ");
     scanf("%d",&x);
    printf("\nVVedite y: ");
    scanf("%d",&y);
    printf("\nVVedite kol-vo vidov: ");
    scanf("%d",&kolvo_vid);


    int vidi[10000]; //сколько рыб каждого вида, 2 4 1 2 5

    for(int i=0;i<kolvo_vid;i++)
    {
        printf("\nVcego %d ribok, VVedite kol-vo ribok %d ogo vida : ",n,i+1);
        scanf("%d",&vidi[i]);
        n+=vidi[i];
    }

    printf("\nVvedite kol-vo xicshix ribok: ");
    scanf("%d",&nx);

    fish TR[10000]; // массив травоядых рыб
    n = 0;
    for(int i=0;i<kolvo_vid;i++)
    {
        int size = rand()%15 + 2;
        int vcr = rand()%10 + 2;
        COLORREF color = RGB(10,10+rand()%240,10+rand()%240);
        for(int j=0;j<vidi[i];j++)
        {
            TR[n] =  fish(color,vcr,size,rand()%(x-20)+10,rand()%(y-20)+10);
            n++;
        }

    }
    predatory_fish xishn[10000]; // массив хищных рыб
    for(int j=0;j<nx;j++)
    {
        xishn[j] =  predatory_fish(12,15,rand()%(x-20)+10,rand()%(y-20)+10);
    }


///До сюда инициализация была, теперь сам цикл работы



    system("cls");



    txCreateWindow(x,y);
    txSetFillColor (RGB(255,255,255));
    txRectangle (0, 0, x, y);

    while(!txGetAsyncKeyState (VK_ESCAPE))
    {
        txSetFillColor (RGB(255,255,255));
        txRectangle (0, 0, x, y);
```

```
        //взаимодейтвие жищных рыб с обычными, они сьедят всех маленьких рыб
        for(int i=0;i<nx;i++)
            for(int j=0;j<n;j++)
                xishn[i].eating(&TR[j]);



        //взаимодейтвие травоядных рыб друг с другом

        for(int i=0;i<n;i++)
            for(int j=i;j<n;j++)
            {
                if(i==j)continue;
                TR[i].vzaimodeictvie(&TR[j]);
            }

          //взаимодейтвие жищных рыб с обычными, они оттолкнться от травоядных
которые примерно тех же размеров
        for(int i=0;i<nx;i++)
            for(int j=0;j<n;j++)
                xishn[i].vzaimodeictvie(&TR[j]);


        //взаимодейтсвие рыб со стенами
        for(int i=0;i<n;i++)
            TR[i].vzaimodeictvie(x,y);
        for(int i=0;i<nx;i++)
            xishn[i].vzaimodeictvie(x,y);



        //отрисовка рыб

        for(int i=0;i<n;i++)
            otricovka(TR[i]);

        for(int i=0;i<nx;i++)
            otricovka(xishn[i]);
        txSleep(50);
    }

}
```

Реализован 2D инвентарь с использованием
библиотеки TXLib. Созданы необходимые
классы для раб. Есть небольшие
замечания по организации кода
и работой с памятью.
Оценка хорошо.