МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(московский ордена Ленина, ордена Октябрьской Революции

авиационный институт имени Серго Орджоникидзе)

(национальный исследовательский университет)

МАИ

Кафедра 304 «Вычислительные машины, системы и сети»

Отчёт по Лабораторной работе №3

По учебной дисциплине «Объектно-ориентированное программирование»

Вариант №11

Выполнил:

студент группы М3О-225Бк-21 Вельковский З.И.\_\_\_\_\_\_\_\_\_

Принял:

Доцент кафедры 304, к.т.н. Новиков П.В.

Старший преподаватель кафедры 304, Ивашенцев И.В.

Москва 2023

# Постановка задачи

**Тема работы**

Динамический полиморфизм

**Цель работы**

Изучение взаимодействия фигур между собой, просчёт коллизии и организация взаимодействия на примере языков программирования C++ с использованием редактора кода Visual Studio.

**Индивидуальное задание на выполнение**

Реализовать взаимодействие объектов через динамический полиморфизм с использованием матрицы перехода.

# Проектирование классов

### Класс Location

Поля (защищённые):

x, y – координаты 1 и 2

Методы (открытые):

Конструктор: входные параметры: значения координат; в конструкторе происходит их первоначальная установка

Методы-акцессоры:

setXY – функция установки защищённых полей координат, присваивает координатам объекта переданные значения

getX, getY – функции получения значений координат

### Класс Point : публичное наследование от класса Location

Методы (открытые):

Конструктор: входные параметры: значения координат; в конструкторе происходит их первоначальная установка

move – функция перемещения; сменяет координаты на указанное значение

show – функция отрисовки; отрисовывает точку

### Класс Barrier : публичное наследование от класса Point

Методы (открытые):

Конструктор: входные параметры: значения координат; в конструкторе происходит их первоначальная установка

Поля (открытые):

right, bottom – координаты нижнего левого угла каждого препятствия

### Класс Aster : публичное наследование от класса Point

Поля (защищённые):

width – ширина линии, которой рисуется фигура

Методы (открытые):

Конструктор: входные параметры: значения координат; в конструкторе происходит их первоначальная установка

onKeyDown - функция обработки нажатия клавиш; сменяет координаты, при нажатии кнопок (W, A, S, D)

show - функция отрисовки; отрисовывает всю фигуру

hit – функция присчитывания столкновения с объектом, по переданным значениям верхней левой и нижней правой точки объекта

### Класс Rocket : публичное наследование от класса Barrier

Поля (защищённые):

width – ширина линии, которой рисуется фигура

Методы (открытые):

Конструктор: входные параметры: значения координат; в конструкторе происходит их первоначальная установка

show - функция отрисовки; отрисовывает фигуру

### Класс Aster\_2 : публичное наследование от класса Barrier

Поля (защищённые):

width – ширина линии, которой рисуется фигура

Методы (открытые):

Конструктор: входные параметры: значения координат; в конструкторе происходит их первоначальная установка

show - функция отрисовки; отрисовывает фигуру

### Класс aster\_dmg\_1 : публичное наследование от класса Face

Методы (открытые):

Конструктор: входные параметры: значения координат; в конструкторе

show - функция отрисовки; отрисовывает фигуру

### Класс aster\_dmg\_2 : публичное наследование от класса Face

Методы (открытые):

Конструктор: входные параметры: значения координат; в конструкторе

show - функция отрисовки; отрисовывает фигуру

### Класс aster\_dmg\_3 : публичное наследование от класса Face

Методы (открытые):

Конструктор: входные параметры: значения координат; в конструкторе

show - функция отрисовки; отрисовывает фигуру

# Текст программы

### oop\_3.cpp

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* КАФЕДРА № 304 2 КУРС \*

\*------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : LR OOP3 \*

\* File Name : LR OOP3.cpp \*

\* Language : C++ \*

\* Programmers(s) : Вельковский З.И. \*

\* Created : 12/10/23 \*

\* Last Revision : 12/10/23 \*

\* Comments(s) : Динамический полиморфизм \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdlib.h>

#include <string.h>

#include <tchar.h>

#include <windows.h>

#include "Figures.hpp"

static TCHAR szWindowClass[] = \_T("DesktopApp");

static TCHAR szTitle[] = \_T("Labwork 2");

HINSTANCE hInst;

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

int WINAPI WinMain(\_In\_ HINSTANCE hInstance, \_In\_opt\_ HINSTANCE hPrevInstance, \_In\_ LPSTR lpCmdLine, \_In\_ int nCmdShow) {

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(wcex.hInstance, IDI\_APPLICATION);

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = NULL;

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, IDI\_APPLICATION);

if (!RegisterClassEx(&wcex)) {

MessageBox(NULL, \_T("Call to RegisterClassEx failed!"), szTitle, NULL);

return 1;

}

hInst = hInstance;

HWND hWnd = CreateWindowEx(WS\_EX\_OVERLAPPEDWINDOW, szWindowClass, szTitle,

WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT,

1024, 512, NULL, NULL, hInstance, NULL);

if (!hWnd) {

MessageBox(NULL, \_T("Call to CreateWindow failed!"), szTitle, NULL);

return 1;

}

ShowWindow(hWnd, nCmdShow);

MSG msg;

while (GetMessage(&msg, NULL, 0, 0)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return (int)msg.wParam;

}

//Инициализаци объектов

Rocket rocket(800, 50);

Aster\_2 aster\_2(0, 50);

int startX = 400, startY = 0;

Aster Aster1(startX, startY);

aster\_dmg\_1 Aster2(startX, startY);

aster\_dmg\_2 Aster3(startX, startY);

aster\_dmg\_3 Aster4(startX, startY);

//Массив переходов

int AsterNow = 0;

Aster\* AsterArr[5] = { &Aster1, &Aster2, &Aster3, &Aster4 };

Aster\* AsterPtr = AsterArr[AsterNow];

//Матрица переходов

int transitionMatrix[2][4] = {

{ 3, 1, 1, 2},

{ 2, 0, 3, 0}

};

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {

PAINTSTRUCT ps;

HDC hdc;

TCHAR greeting[] = \_T("Hello, Windows desktop!");

switch (message) {

case WM\_KEYDOWN:

{

AsterPtr->onKeyDown(wParam);

if (AsterPtr->hit(rocket.getX(), rocket.getY(), rocket.getX() + rocket.right, rocket.getY() + rocket.bottom) == 1) {

AsterNow = transitionMatrix[0][AsterNow];

AsterPtr = AsterArr[AsterNow];

AsterPtr->setXY(startX, startY);

}

if (AsterPtr->hit(aster\_2.getX(), aster\_2.getY(), aster\_2.getX() + aster\_2.right, aster\_2.getY() + aster\_2.bottom) == 1) {

AsterNow = transitionMatrix[1][AsterNow];

AsterPtr = AsterArr[AsterNow];

AsterPtr->setXY(startX, startY);

}

InvalidateRect(hWnd, NULL, TRUE);

break;

}

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

AsterPtr->show(hdc);

rocket.show(hdc);

aster\_2.show(hdc);

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

break;

}

return 0;

}

### Figures.hpp

#ifndef LR\_OOP2\_FIGURES

#define LR\_OOP2\_FIGURES

#include <windows.h>

class Location;

class Point;

class Barrier;

class Aster;

class Rocket;

class Aster\_2;

class aster\_dmg\_1;

class aster\_dmg\_2;

class aster\_dmg\_3;

class Location

{

protected:

int x;

int y;

public:

Location(int x, int y);

void setXY(int x, int y);

int getX();

int getY();

};

class Point : public Location

{

public:

Point(int x, int y);

void move(int offsetX, int offsetY);

virtual void show(HDC hdc);

};

class Barrier : public Point

{

public:

Barrier(int x, int y);

int right = 100;

int bottom = 100;

};

class Aster : public Point

{

protected:

int width = 3;

public:

Aster(int x, int y);

void onKeyDown(WPARAM wParam);

void show(HDC hdc);

bool hit(int left, int top, int right, int bottom);

};

class Rocket : public Barrier

{

protected:

int width = 2;

public:

Rocket(int x, int y);

void show(HDC hdc);

};

class Aster\_2 : public Barrier

{

protected:

int width = 5;

public:

Aster\_2(int x, int y);

void show(HDC hdc);

};

class aster\_dmg\_1 : public Aster

{

public:

aster\_dmg\_1(int x, int y);

void show(HDC hdc);

};

class aster\_dmg\_2 : public Aster

{

public:

aster\_dmg\_2(int x, int y);

void show(HDC hdc);

};

class aster\_dmg\_3 : public Aster

{

public:

aster\_dmg\_3(int x, int y);

void show(HDC hdc);

};

#endif

### Figures.cpp

#include "Figures.hpp"

Location::Location(int x, int y)

{

this->x = x;

this->y = y;

}

void Location::setXY(int x, int y)

{

this->x = x;

this->y = y;

}

int Location::getX() { return x; }

int Location::getY() { return y; }

Point::Point(int x, int y) : Location(x, y) {};

void Point::show(HDC hdc)

{

SetPixel(hdc, x, y, RGB(0, 0, 0));

}

Barrier::Barrier(int x, int y) : Point(x, y) {};

void Point::move(int offsetX, int offsetY)

{

this->x += offsetX;

this->y += offsetY;

}

Aster::Aster(int x, int y) : Point(x, y) {};

void Aster::onKeyDown(WPARAM wParam) {

int step = 50;

switch (wParam) {

case 'W':

move(0, -step);

break;

case 'S':

move(0, step);

break;

case 'A':

move(-step, 0);

break;

case 'D':

move(step, 0);

break;

}

}

void Aster::show(HDC hdc)

{

HPEN pen1 = CreatePen(PS\_SOLID, width, RGB(0, 0, 0));

SelectObject(hdc, pen1);

Ellipse(hdc, 100 + x + width, 100 + y + width, 200 + x + width, 200 + y + width);

Ellipse(hdc, 130 + x + width, 140 + y + width, 142 + x + width, 152 + y + width);

SelectObject(hdc, GetStockObject(WHITE\_PEN));

Ellipse(hdc, 120 + x + width, 115 + y + width, 125 + x + width, 120 + y + width);

Ellipse(hdc, 135 + x + width, 165 + y + width, 140 + x + width, 170 + y + width);

SelectObject(hdc, pen1);

DeleteObject(pen1);

}

bool Aster::hit(int left, int top, int right, int bottom)

{

return (!( (x > right) || (y > bottom) || ((x + 200 + 2 \*width) < left) || ((y + 200 + 2 \*width) < top) ));

}

Rocket::Rocket(int x, int y) : Barrier(x, y) {};

void Rocket::show(HDC hdc)

{

HPEN pen = CreatePen(PS\_SOLID, 2, RGB(0, 0, 0));

SelectObject(hdc, pen);

// Рисуем нос ракеты

POINT nose[3];

nose[0] = { 30 + x, 0 + y };

nose[1] = { 50 + x, -30 + y };

nose[2] = { 70 + x, 0 + y };

Polygon(hdc, nose, 3);

// Рисуем корпус ракеты

Rectangle(hdc, 30 + x, 0 + y, 70 + x, 100 + y);

// Рисуем крылья

POINT wing1[3];

wing1[0] = { 30 + x, 50 + y };

wing1[1] = { 10 + x, 70 + y };

wing1[2] = { 30 + x, 90 + y };

Polygon(hdc, wing1, 3);

POINT wing2[3];

wing2[0] = { 70 + x, 50 + y };

wing2[1] = { 90 + x, 70 + y };

wing2[2] = { 70 + x, 90 + y };

Polygon(hdc, wing2, 3);

SelectObject(hdc, GetStockObject(BLACK\_PEN));

DeleteObject(pen);

}

Aster\_2::Aster\_2(int x, int y) : Barrier(x, y) {};

void Aster\_2::show(HDC hdc)

{

HPEN pen1 = CreatePen(PS\_SOLID, width, RGB(0, 0, 0));

SelectObject(hdc, pen1);

// Рисуем астероид

Ellipse(hdc, 40 + x + width, 40 + y + width, 140 + x + width, 140 + y + width);

Ellipse(hdc, 60 + x + width, 60 + y + width, 80 + x + width, 80 + y + width);

SelectObject(hdc, GetStockObject(BLACK\_PEN));

DeleteObject(pen1);

}

aster\_dmg\_1::aster\_dmg\_1(int x, int y) : Aster(x, y) {};

void aster\_dmg\_1::show(HDC hdc)

{

HPEN pen1 = CreatePen(PS\_SOLID, width, RGB(0, 0, 0));

SelectObject(hdc, pen1);

Ellipse(hdc, 100 + x + width, 100 + y + width, 200 + x + width, 200 + y + width);

Ellipse(hdc, 120 + x + width, 120 + y + width, 130 + x + width, 130 + y + width);

Ellipse(hdc, 130 + x + width, 140 + y + width, 142 + x + width, 152 + y + width);

Ellipse(hdc, 140 + x + width, 175 + y + width, 160 + x + width, 190 + y + width);

SelectObject(hdc, GetStockObject(WHITE\_PEN));

Ellipse(hdc, 120 + x + width, 115 + y + width, 125 + x + width, 120 + y + width);

Ellipse(hdc, 135 + x + width, 165 + y + width, 140 + x + width, 170 + y + width);

SelectObject(hdc, pen1);

DeleteObject(pen1);

}

aster\_dmg\_2::aster\_dmg\_2(int x, int y) : Aster(x, y) {};

void aster\_dmg\_2::show(HDC hdc) {

HPEN pen1 = CreatePen(PS\_SOLID, width, RGB(0, 0, 0));

HPEN pen2 = CreatePen(PS\_SOLID, width, RGB(255, 255, 255));

SelectObject(hdc, pen1);

// Рисуем астероид

Ellipse(hdc, 100 + x + width, 100 + y + width, 200 + x + width, 200 + y + width);

Ellipse(hdc, 120 + x + width, 120 + y + width, 130 + x + width, 130 + y + width);

Ellipse(hdc, 130 + x + width, 140 + y + width, 142 + x + width, 152 + y + width);

Ellipse(hdc, 140 + x + width, 175 + y + width, 160 + x + width, 190 + y + width);

SelectObject(hdc, pen2);

Pie(hdc, 90 + x + width, 170 + y + width, 158 + x + width, 190 + y + width, 100 + x + width, 100 + y + width, 200 + x + width, 200 + y + width);

SelectObject(hdc, pen1);

Ellipse(hdc, 90 + x + width, 170 + y + width, 130 + x + width, 190 + y + width);

SelectObject(hdc, pen2);

Pie(hdc, 80 + x + width, 155 + y + width, 125 + x + width, 195 + y + width, 90 + x + width, 170 + y + width, 130 + x + width, 190 + y + width);

SelectObject(hdc, pen1);

DeleteObject(pen1);

}

aster\_dmg\_3::aster\_dmg\_3(int x, int y) : Aster(x, y) {};

void aster\_dmg\_3::show(HDC hdc) {

HPEN pen1 = CreatePen(PS\_SOLID, width, RGB(0, 0, 0));

HPEN pen2 = CreatePen(PS\_SOLID, width, RGB(255, 255, 255));

SelectObject(hdc, pen1);

Ellipse(hdc, 100 + x + width, 100 + y + width, 200 + x + width, 200 + y + width);

Ellipse(hdc, 120 + x + width, 120 + y + width, 130 + x + width, 130 + y + width);

Ellipse(hdc, 130 + x + width, 140 + y + width, 142 + x + width, 152 + y + width);

Ellipse(hdc, 140 + x + width, 175 + y + width, 160 + x + width, 190 + y + width);

SelectObject(hdc, pen2);

Pie(hdc, 90 + x + width, 170 + y + width, 158 + x + width, 190 + y + width, 100 + x + width, 100 + y + width, 200 + x + width, 200 + y + width);

Pie(hdc, 180 + x + width, 100 + y + width, 220 + x + width, 160 + y + width, 100 + x + width, 100 + y + width, 200 + x + width, 200 + y + width);

SelectObject(hdc, pen1);

Ellipse(hdc, 90 + x + width, 170 + y + width, 130 + x + width, 190 + y + width);

Ellipse(hdc, 170 + x + width, 122 + y + width, 200 + x + width, 170 + y + width);

SelectObject(hdc, pen2);

Pie(hdc, 80 + x + width, 155 + y + width, 125 + x + width, 195 + y + width, 90 + x + width, 170 + y + width, 130 + x + width, 190 + y + width);

SelectObject(hdc, GetStockObject(BLACK\_PEN));

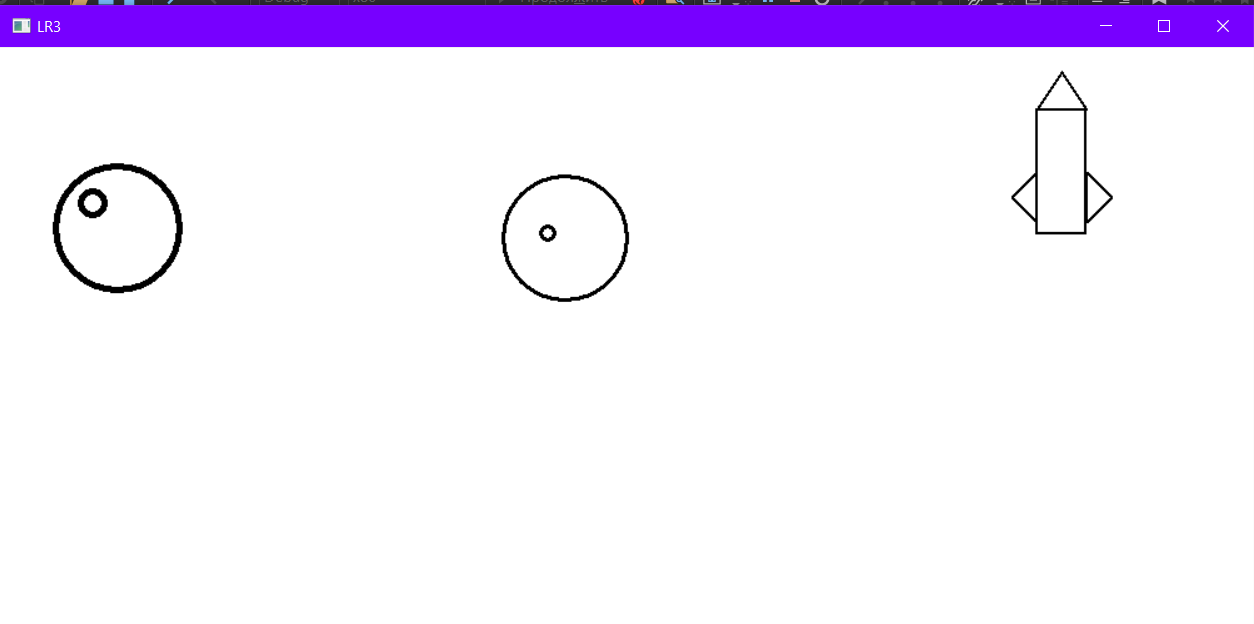
DeleteObject(pen1);

DeleteObject(pen2);

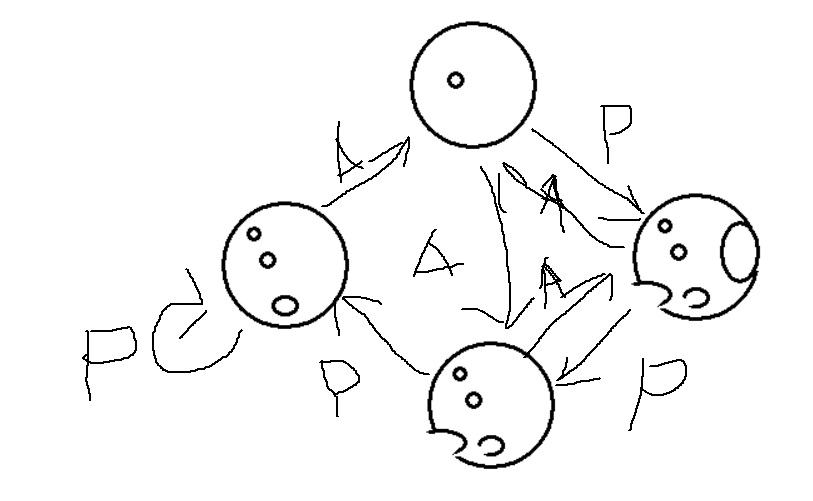
}

# Результат работы программы

При запуске программы появляется астероид, которым можно управлять клавишами W, A, S, D. Так же на экране присутствуют объекты «ракета» и «астероид 2» по бокам от основной фигуры.



При взаимодействии с объектами астероид меняется следующим алгоритмом («р» – ракета, «а» - астероид).



# Вывод

В результате выполнения лабораторной работы были изучены взаимодействия фигур между собой, просчёт коллизии и организовано взаимодействие на примере языков программирования C++ с использованием редактора кода Visual Studio.

Список источников:

1. Лекционные материалы по дисциплине «Объектно-ориентированное программирование» (2023г. Чечиков Юрий Борисович, МАИ)
2. Гайд по фигурам (КАК рисовать в Win32 API):  
   <http://radiofront.narod.ru/htm/prog/htm/winda/api/paint.html>
3. Объектно-ориентированное программирование МАИ Новиков П.В.: <https://docs.yandex.ru/docs/view?url=ya-mail%3A%2F%2F182677259885232628%2F1.2&name=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%9C%D0%90%D0%98%20%D0%9D%D0%BE%D0%B2%D0%B8%D0%BA%D0%BE%D0%B2%20%D0%9F.%D0%92..pdf&uid=276902236>