**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ**

**ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное учреждение высшего образования

**«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ)**

**Институт информационных технологий, математики и механики**

**ОТЧЕТ по учебной практике**

на тему: **«ООП. Визуальное программирование»**

**Выполнил:**

студент группы 381806-2

Лукьянченко Иван Алексеевич

**Научный руководитель:**

Барышева Ирина Викторовна

Нижний Новгород 2019 г.

**Содержание:**

*1.Введение……………………………………………………………… 3 стр.*

*2.Постановка задачи…………………………………………………4 стр.*

*3.Структура проекта ………………………………………………5 стр.*

*4.Инструкция пользователю ………………………………………8 стр.*

*5.Контрольный пример ……………………………………….........9 стр.*

*6.Список литературы … …………………………………………10 стр.*

*8.Приложение……………………………………………………… 11 стр.*

**1.Введение**

**Визуальное программирование** — способ создания программы для ЭВМ путём манипулирования графическими объектами вместо написания её текста. Визуальное программирование часто представляют, как следующий этап развития текстовых языков программирования. Наглядным примером может служить утилита Визуальный Pascal или Microsoft Visual Studio, где редактируются графические объекты и одновременно отображается соответствующий текст программы. В последнее время визуальному программированию стали уделять больше внимания — в связи с развитием мобильных сенсорных устройств (КПК, планшеты). Визуальное программирование в основном используется для создания программ с графическим интерфейсом для операционных систем с графическим интерфейсом пользователя. Среда визуального программирования позволяет написать Веб-приложение для браузеров. Среда визуального программирования позволяет создать консольное приложение (программа без графического интерфейса и без вывода сообщений в консоль) для программирования микроконтроллеров, программируемых микросхем.

**Объектно-ориентированное программирование (ООП)** — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования. Идеологически ООП — подход к программированию как к моделированию информационных объектов, решающий на новом уровне основную задачу структурного программирования: структурирование информации с точки зрения управляемости, что существенно улучшает управляемость самим процессом моделирования, что, в свою очередь, особенно важно при реализации крупных проектов.

Управляемость для иерархических систем предполагает минимизацию избыточности данных (аналогичную нормализации) и их целостность, поэтому созданное удобно управляемым — будет и удобно пониматься. Таким образом, через тактическую задачу управляемости решается стратегическая задача — транслировать понимание задачи программистом в наиболее удобную для дальнейшего использования форму

**2.Постановка задачи**

Была поставлена задача создать движущийся объект в Window form с некоторым элементами ввода данных и управлением действий:



**3.Структура проекта**

Для Выполнения поставленной задачи, был создан проект, написанный на языке «С++» в среде «Visual Studio 2017». Программа рисует на форме «паровоз» и приводит его в движение. Проект имеет следующую структуру:

* Заголовочный файл библиотеки данных- Pr.h

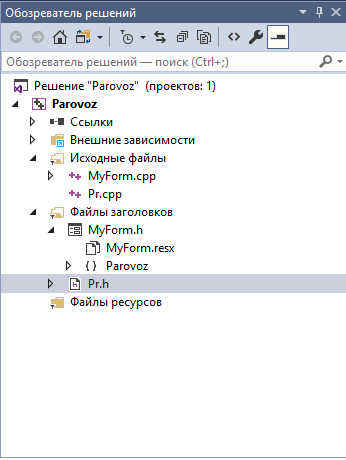
В данной библиотеке содержатся текста всех классов(прямоугольник, линия, окно, колесо, дверь, труба, сцепка, движок, вагон, паровоз, поезд).

* Заголовочный файл библиотеки данных- MyForm.h

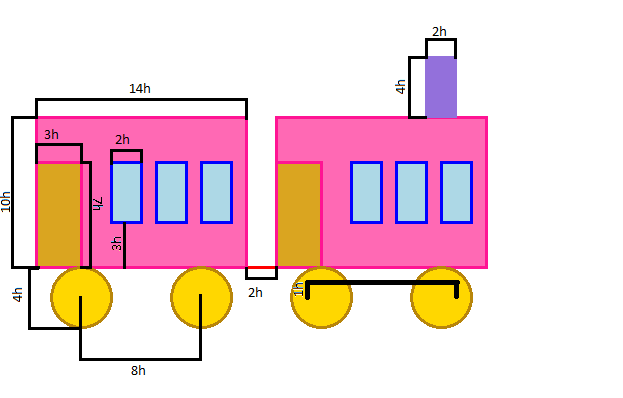
В данной библиотеке содержатся устройство самой формы и функции, выполняемые при нажатии клавиш.

* Исходный файл- MyForm.cpp

Здесь создается форма.

****

Так же был создан макет с размерами паровоза:



Каждая геометрическая фигура задается несколькими параметрами:

* Прямоугольник:

(x;y)-координаты точки верхнего левого угла;

а-высота прямоугольника;

b-ширина прямоугольника;

Цвет формы и контура;

* Эллипс:

(x;y)-координаты точки верхнего левого угла;

а-высота прямоугольника, описывающего эллипс;

b-ширина прямоугольника, описывающего эллипс;

Цвет формы и контура;

* Линия:

(x1;y1)-координаты точки начала линии;

(x2;y2)-координаты точки конца линии;

Цвет;

Толщина;

Схема наследования классов(элементов) поезда от геометрических фигур:

**Прямоугольник**

**Линия**

Сцепка

Движок

Труба

Колесо

Корпус

Окно

Дверь

**Паровоз**

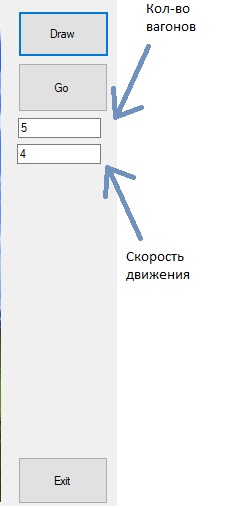
**Вагон**

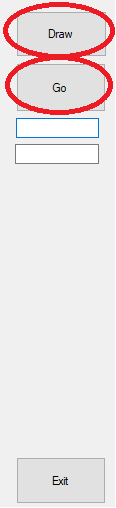
**n 1**

**Поезд**

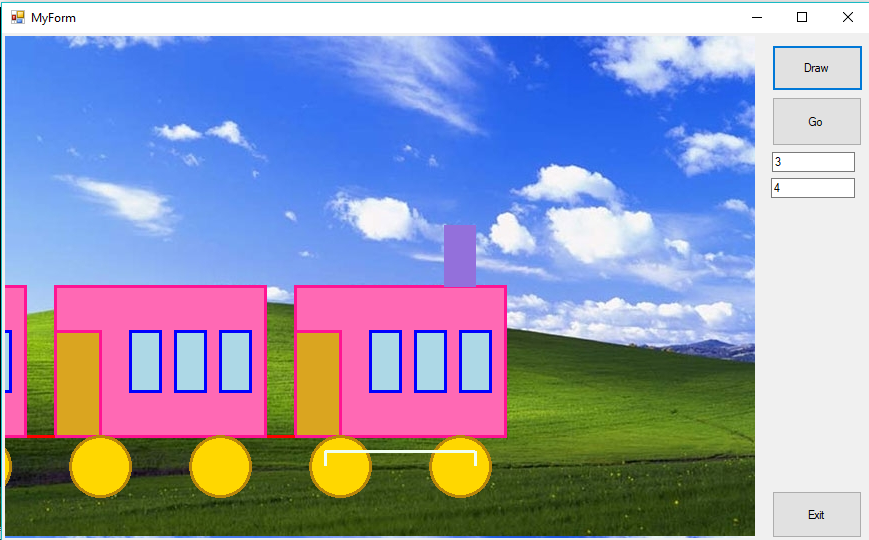
**4.Инструкция пользователю**

Программа имеет интуитивный графический интерфейс. Сначала предлагается ввести количество вагонов и скорость движения в поля ввода:



После чего можем с помощью 2 кнопок нарисовать поезд «Draw» и привести его в движение «Go».Для выхода и закрытия окна необходимо нажать на кнопку «Exit»:

**5.Контрольный пример**



**Вывод:** Поставленная нами задача в начале проекта была выполнена. Программа рисует и приводит в движение объект. **6.Список литературы**

1. Пахомов Б. И. «C/C++ и MS Visual C++ 2008 для начинающих» / БХВ-Петербург, 2009 – 621c.
2. Лафоре Р. «Объектно-ориентированное программирование в С++» / Пер. с англ. - [Кузнецов А](https://www.labirint.ru/authors/48126/), 2018.- 928 c.
3. Википедия – свободная энциклопедия [Электронный ресурс], - https://ru.wikipedia.org/- статья в интернете.
4. Вставская, Е. Программирование [Электронный ресурс], - https://prog-cpp.ru/- статья в интернете.
5. PureCodeCpp. Основы программирования на С/С++ для начинающих. Алгоритмы в С/С++ [Электронный ресурс],- http://purecodecpp.com/algoritmy-v-c/ - статья в интернете

**7.Приложение**

\*а-массив неотсортированных данных; n-длина массива

**Класс Прямоугольник:**

class Pr

{

protected:

int x, y, a, b;

public:

Pr(int \_x = 0, int \_y = 0,int \_a = 0, int \_b = 0) {

x = \_x;

y = \_y;

a = \_a;

b = \_b;

}

~Pr() {};

virtual void show(Graphics^g)=0;

void move(int dx){

x += dx;

}

**Класс дверь:**

class my\_door :public Pr {

public:

my\_door(int \_x = 0, int \_y = 0, int \_h = 0):Pr(\_x,\_y,3\*\_h,7\*\_h)

{};

void show(Graphics^g) {

SolidBrush^brush = gcnew SolidBrush(Color::Goldenrod);

g->FillRectangle(brush, x, y, a, b);

Pen^pen = gcnew Pen(Color::DeepPink, 3);

g->DrawRectangle(pen, x, y, a, b);

}

};

**Класс окно:**

class my\_window :public Pr {

public:

my\_window(int \_x = 0, int \_y = 0, int \_h = 0) :Pr(\_x, \_y, 2 \* \_h, 4 \* \_h) {};

void show(Graphics^g) {

SolidBrush^brush = gcnew SolidBrush(Color::LightBlue);

g->FillRectangle(brush, x, y, a, b);

Pen^pen = gcnew Pen(Color::Blue,3);

g->DrawRectangle(pen, x, y, a, b);

}

};

**Класс Корпус:**

class my\_body :public Pr {

public:

my\_body(int \_x = 0, int \_y = 0, int \_h = 0) :Pr(\_x, \_y, 14 \* \_h, 10 \* \_h) {};

void show(Graphics^g) {

SolidBrush^brush = gcnew SolidBrush(Color::HotPink);

g->FillRectangle(brush, x, y, a, b);

Pen^pen = gcnew Pen(Color::DeepPink, 3);

g->DrawRectangle(pen, x, y, a, b);

}

};

**Класс колесо:**

class my\_circle :public Pr {

public:

my\_circle(int \_x = 0, int \_y = 0, int \_h = 0) :Pr(\_x, \_y, 4 \* \_h, 4 \* \_h) {};

void show(Graphics^g) {

SolidBrush^brush = gcnew SolidBrush(Color::Gold);

g->FillEllipse(brush, x, y, a, b);

Pen^pen = gcnew Pen(Color::DarkGoldenrod, 3);

g->DrawEllipse(pen, x, y, a, b);

}

};

**Класс Вагон:**

class vagon {

private:

my\_body body;

my\_door door;

my\_window \*wind;

my\_circle rcircle, lcircle;

public:

vagon(int \_x = 0, int \_y = 0, int \_h = 1) {

body = my\_body(\_x, \_y, \_h);

door = my\_door(\_x, \_y+3\*\_h, \_h);

lcircle = my\_circle(\_x + \_h, \_y + 10 \* \_h, \_h);

rcircle = my\_circle(\_x + 9 \* \_h, \_y + 10 \* \_h, \_h);

wind = new my\_window[3];

for (int i = 0; i < 3; i++) {

wind[i] = my\_window(\_x + 5 \* \_h + i \* 3 \* \_h, \_y + 3 \* \_h, \_h);

}}

vagon(const vagon& tmp) {

body = tmp.body;

door = tmp.door;

rcircle = tmp.rcircle;

lcircle = tmp.lcircle;

wind = new my\_window[3];

for (int i = 0; i < 3; i++) {

wind[i] = tmp.wind[i];

}

}

~vagon() {

delete[]wind;

}

vagon&operator=(const vagon&tmp) {

body = tmp.body;

door = tmp.door;

rcircle = tmp.rcircle;

lcircle = tmp.lcircle;

wind = new my\_window[3];

for (int i = 0; i < 3; i++) {

wind[i] = tmp.wind[i];

}

return \*this;

}

void show(Graphics^g) {

body.show(g);

door.show(g);

rcircle.show(g);

lcircle.show(g);

for (int i = 0; i < 3; i++) {

wind[i].show(g);

}

}

void move(int dx) {

body.move(dx);

door.move(dx);

rcircle.move(dx);

lcircle.move(dx);

for (int i = 0; i < 3; i++) {

wind[i].move(dx);

}

}

};

**Класс Линия:**

class lin {

protected:

int xl, yl, xr, yr;

public:

lin(int \_xl = 0, int \_yl = 0, int \_xr = 0, int \_yr = 0) {

xl = \_xl;

yl = \_yl;

xr = \_xr;

yr = \_yr;

}

~lin() {};

virtual void move(int dx) {

xl += dx;

xr += dx;

}

virtual void show(Graphics ^g) {

Pen^pen = gcnew Pen(Color::Honeydew);

pen->Width = 3;

g->DrawLine(pen, xl, yl, xr, yr);

}

};

**Класс Движок:**

class engine {

private:

int xlc, xrc; int yc;

int xlb, ylb;

int xre, yre; int yd;

int h;

double a; //=0

int yv;

lin leftline;

lin middleline;

lin rightline;

public:

engine() {};

engine(int x, int y, int \_h) {

h = \_h;

yc = y + 2 \* h;

yd = yc - h;

xlc = x + 2 \* h;

xrc = x + 12 \* h;

xlb = x + 2 \* h;

xre = x + 12 \* h;

leftline = lin(xlc, yc, xlb, yd);

middleline = lin(xlb, yd, xre, yd);

rightline = lin(xre, yd, xrc, yc);

}

engine(const engine &tmp) {

leftline = tmp.leftline;

middleline = tmp.middleline;

rightline = tmp.rightline;

}

void move(int dx) {

a += 5 \* 3.14159265 / 180;

xlc += dx;

xrc += dx;

xlb = xlc + dx + h \* cos(a);

yd = yc + h \* sin(a);

xre = xrc + dx + h \* cos(a);

yre = ylb;

leftline = lin(xlc, yc, xlb, yd);

middleline = lin(xlb, yd, xre, yd);

rightline = lin(xre, yd, xrc, yc);

}

void show(Graphics ^g) {

leftline.show(g);

middleline.show(g);

rightline.show(g);

}

};

**Класс Сцепка:**

class coup :public lin

{public:

coup(int \_x = 0, int \_y = 0, int \_h = 0) :lin(\_x, \_y, \_x+2\*\_h, \_y) {};//vozmojno ne rabotaet

~coup() {};

void show(Graphics^g) {

Pen^pen = gcnew Pen(Color::Red,3);

g->DrawLine(pen, xl, yl, xr, yr);

}

};

**Класс Труба:**

class Truba :public Pr {

public:

Truba(int t\_x = 0, int t\_y = 0, int h = 0) :Pr(t\_x, t\_y, 2 \* h, 4 \* h) { ; };

void show(Graphics^ g) {

SolidBrush^ myBrush = gcnew SolidBrush(Color::MediumPurple);

g->FillRectangle(myBrush, x, y, a, b);

Pen^ myPen = gcnew Pen(Color::MediumPurple, 2.0f);

g->DrawRectangle(myPen, x, y, a, b);

}

};

**Класс Паровоз:**

class par {

private:

vagon vag;

Truba tr;

engine eng;

public:

par(int x = 0, int y = 0, int h = 1) {

vag = vagon(x, y, h);

tr= Truba(x+10\*h,y-4\*h,h);

eng = engine(x , y + 10 \* h, h);

}

par(const par& tmp) {

vag = tmp.vag;

tr=tmp.tr;

eng = tmp.eng;

}

~par() {};

par&operator=(const par&tmp) {

vag = tmp.vag;

tr=tmp.tr;

eng = tmp.eng;

return\*this;

}

void move(int dx){

vag.move(dx);

tr.move(dx);

eng.move(dx);

}

void show(Graphics^g) {

vag.show(g);

tr.show(g);

eng.show(g);

}

};

**Класс Поезд:**

class train {

private:

par p;

int n;

vagon \*vag;

coup \*c;

public:

train(int \_x = 0, int \_y = 0, int \_h = 0, int \_n = 0) {

n = \_n;

p = par(\_x - 14 \* \_h, \_y - 10 \* \_h, \_h);

c = new coup[n];

vag = new vagon[n];

for (int i = 0; i < n; i++) {

c[i]=coup(\_x-16\*\_h\*(i+1),\_y,\_h);

vag[i] = vagon(\_x - (14 \* \_h + 16 \* \_h\*(i + 1)), \_y - 10 \* \_h, \_h);

}

}

train(const train&tmp) {

p = tmp.p;

n = tmp.n;

c = new coup[n];

vag = new vagon[n];

for (int i = 0; i < n; i++) {

c[i] = tmp.c[i];

vag[i] = tmp.vag[i];

}

}

~train() {

delete[]c;

delete[]vag;

}

train&operator=(const train tmp) {

if (n != tmp.n) {

if (n != 0) {

delete[]c;

delete[]vag;

}

n = tmp.n;

vag = new vagon[n];

c = new coup[n];

}

for (int i = 0; i < n; i++) {

c[i] = tmp.c[i];

vag[i] = tmp.vag[i];

}

return \*this;

}

void show(Graphics^g) {

p.show(g);

for (int i = 0; i < n; i++) {

c[i].show(g);

vag[i].show(g);

}

}

void move(int dx) {

p.move(dx);

for (int i = 0; i < n; i++) {

c[i].move(dx);

vag[i].move(dx);

}

}

};

**Кнопка «Draw»:**

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

Graphics^g = pictureBox1->CreateGraphics();

int n;

n = System::Int32::Parse(this->textBox1->Text);

train t(500, 400, 15, n);

t.show(g);

}

**Кнопка «Go»:**

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

Graphics^g = pictureBox1->CreateGraphics();

int n,dx;

n = System::Int32::Parse(this->textBox1->Text);

dx = System::Int32::Parse(this->textBox2->Text);

train t(250, 400, 15, n);

for (int i = 0; i < 10000; i++) {

t.move(dx);

pictureBox1->Refresh();

t.show(g);

}

}

**Кнопка «Exit»:**

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

this->Close();

}