Федеральное агентство по образованию

ФГБОУ ВО Уфимский государственный авиационный технический

университет

Кафедра ВМиК

ЛАБОРАТОРНАЯ РАБОТА №5

По дисциплине: «Объектно-Ориентированное Программирование»

«Виртуальность»

Выполнил: студент группы ПРО-222

Сафронов И.А.

Проверил: Котельников В.А.

Уфа 2020

1. **Цель работы:** определение простейшего класса CCircle с координатами и постоянным радиусом, использование хранилища из Л.Р.3 для хранения объектов класса CCircle
2. **Задачи:** реализовать следующее поведение: при нажатии мышкой на форме создается новый объект CCircle с координатами нажатия и помещается в хранилище; при событии Paint должны отрисовываться на форме все объекты из хранилища

Реализовать следующее поведение с выбором объектов: вариант 1(Макеев Г.А.): при создании объекта он становится единственным выделенным при нажатии ЛКМ (левой клавиши мыши) на какой-либо объект CCircle на форме, он становится «выделенным» и отрисовывается отлично от других объектов при нажатии на кнопку Del, все выделенные объекты должны удаляться при выделении объекта с помощью ЛКМ и удерживаемой клавиши Ctrl, выделенными становятся несколько объектов

1. **Ход** **работы**

При запуске программы появляется стартовое окно, с полем для рисования и кнопками управления. (Рис. 1)

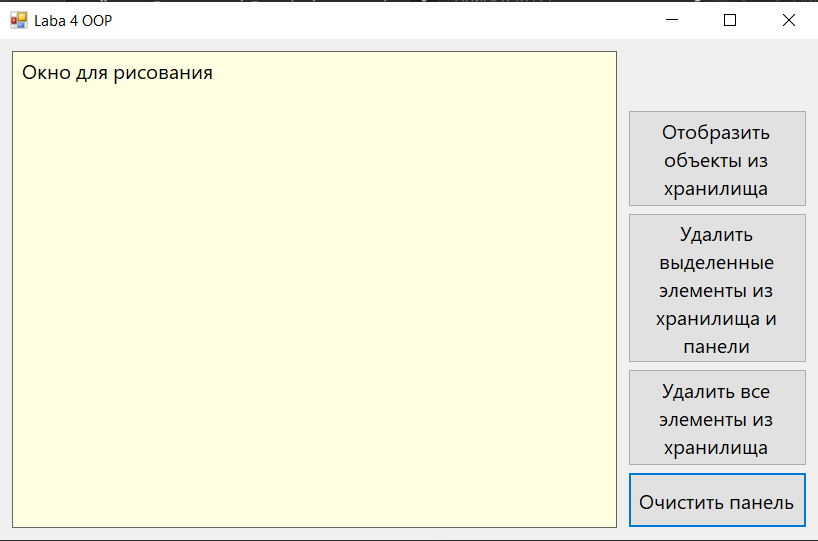


Рис. 1 Стартовое окно

При нажатии мышкой на окне для рисования создается новый объект Circle с координатами нажатия, помещается в хранилище и отображается на панели. (Рис. 2)

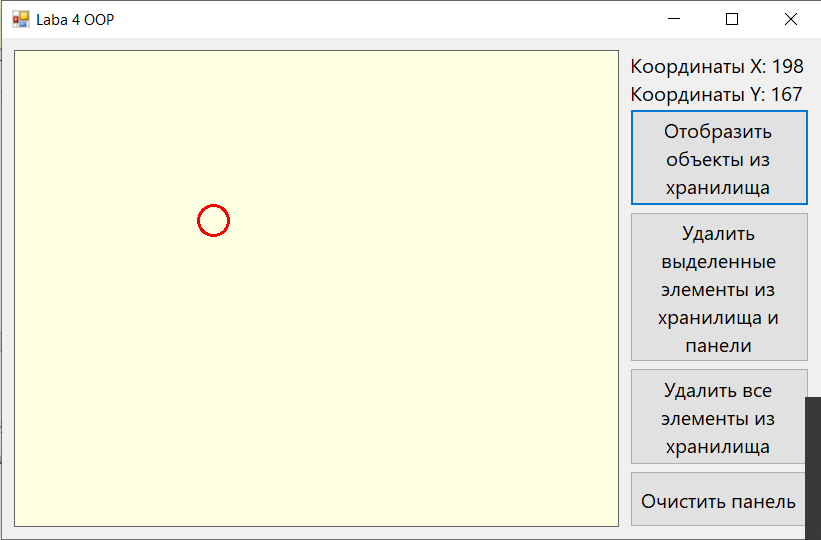


Рис. 2 Отображение круга на полотне

При создании нового объекта он становится единственным выделенным. (Рис. 3)

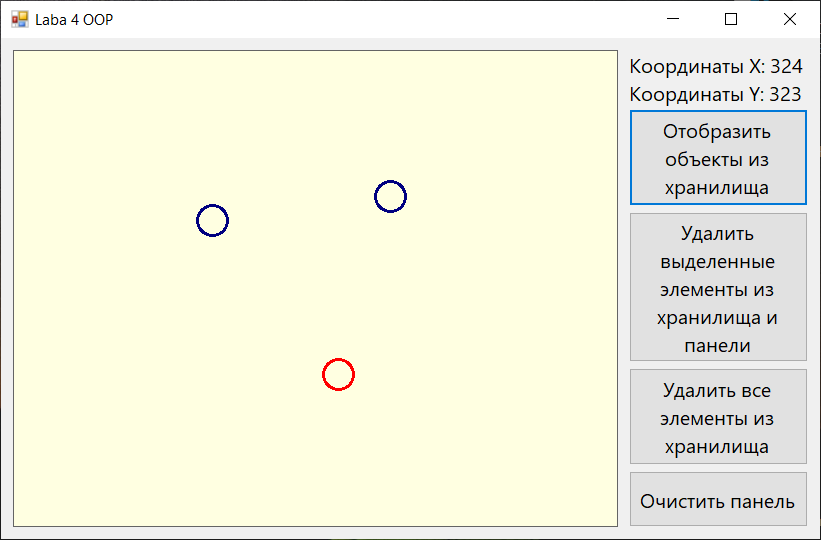


Рис. 3 Выделен только текущий круг

При нажатии ЛКМ (левой клавиши мыши) на какой-либо объект Circle на панели, он становится «выделенным» и отрисовывается отлично от других объектов. (Рис. 4)

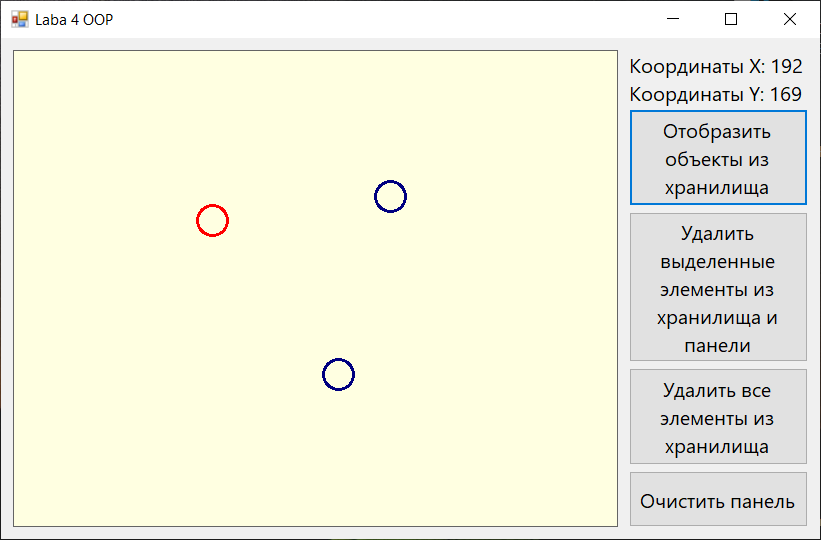


Рис. 4 Выделение уже созданного объекта Circle

При выделении объекта с помощью ЛКМ и удерживаемой клавиши Ctrl, выделенными становятся несколько объектов. (Рис. 5)

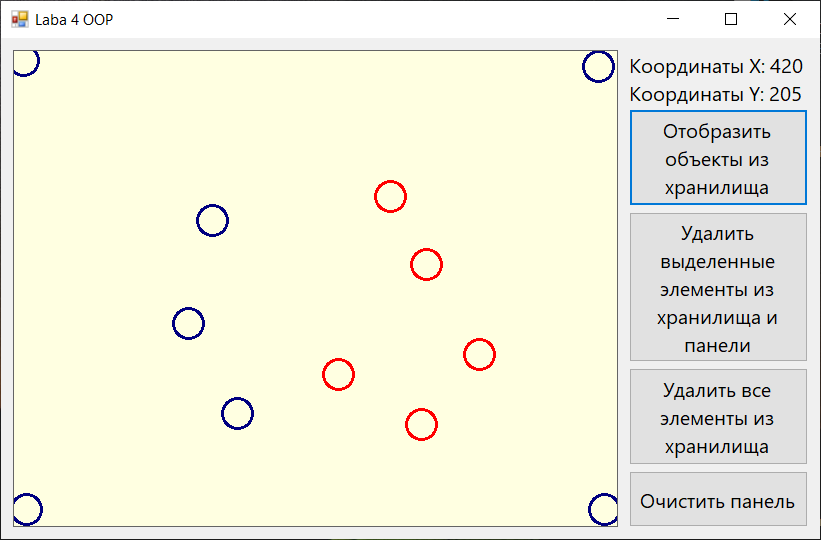


Рис. 5 Выделено несколько объектов

При нажатии на кнопку “Удалить выделенные элементы из хранилища и панели”, все выделенные объекты удаляются из хранилища и перестают отображаться на панели. (Рис. 6)

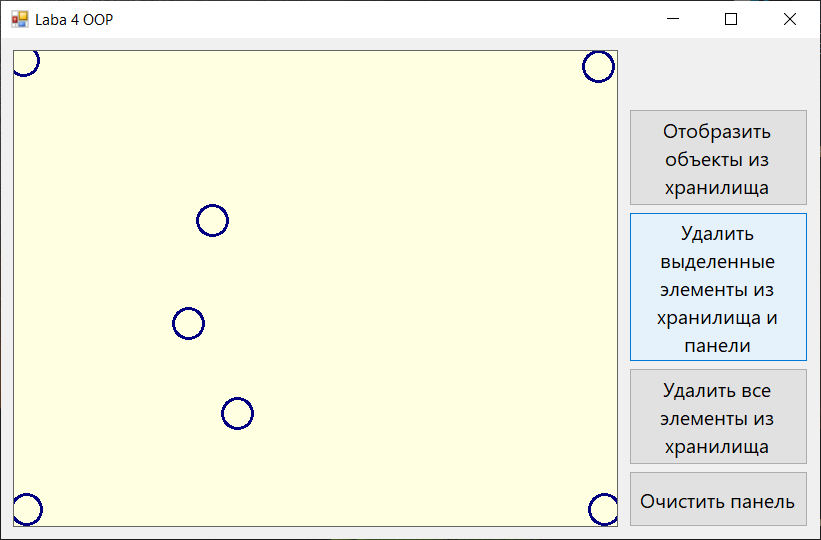


Рис. 6 Демонстрация работы кнопки “Удалить выделенные элементы из хранилища и панели”

При нажатии на кнопку “Очистить панель”, происходит перерисовка панели. (Рис. 7)

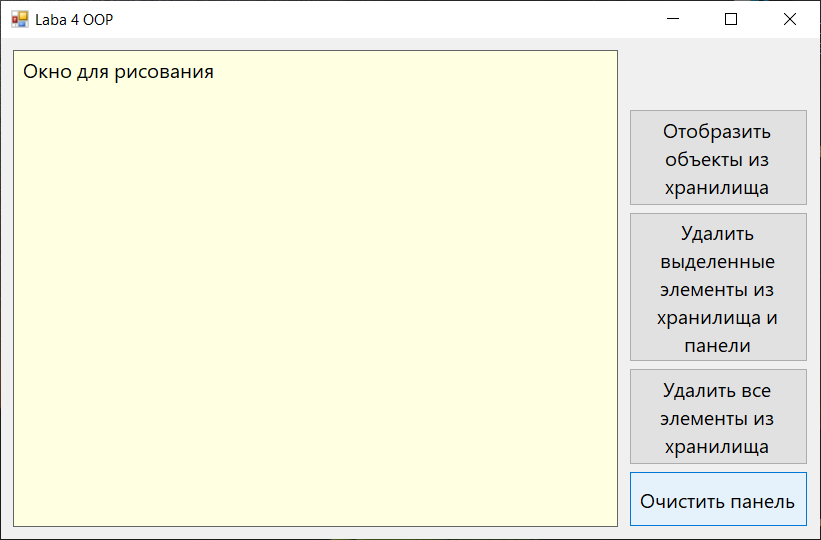


Рис. 7 Демонстрация работы кнопки “Очистить панель”

При нажатии на кнопку “Отобразить объекты из хранилища” на панели будут отображены все объекты, которые хранятся в хранилище. (Рис. 8)

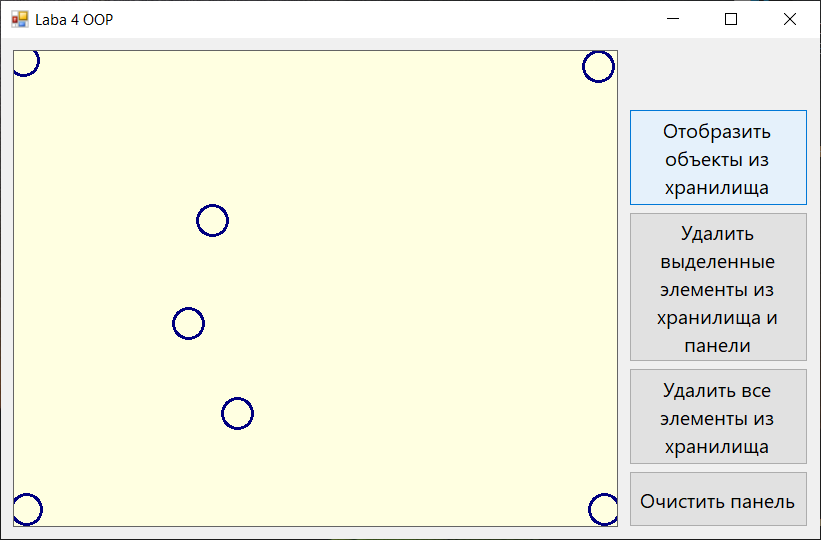


Рис. 8 Демонстрация работы кнопки “Отобразить объекты из хранилища”

При нажатии на кнопку “Удалить все элементы из хранилища” произойдёт очистка хранилища, но панель не будет очищена. Только после нажатии на кнопку “Очистить панель” произойдёт очистка панели. (Рис. 9-10)

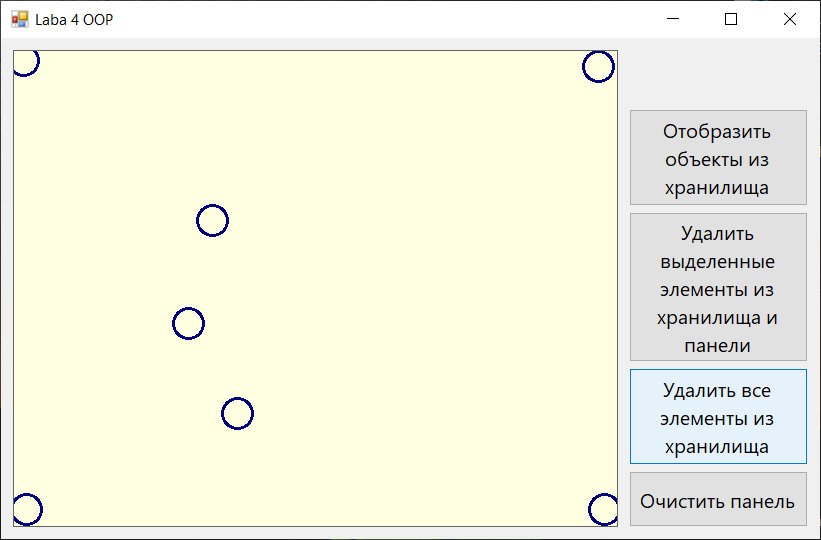


Рис. 9 Демонстрация работы кнопки “Удалить все элементы из хранилища”

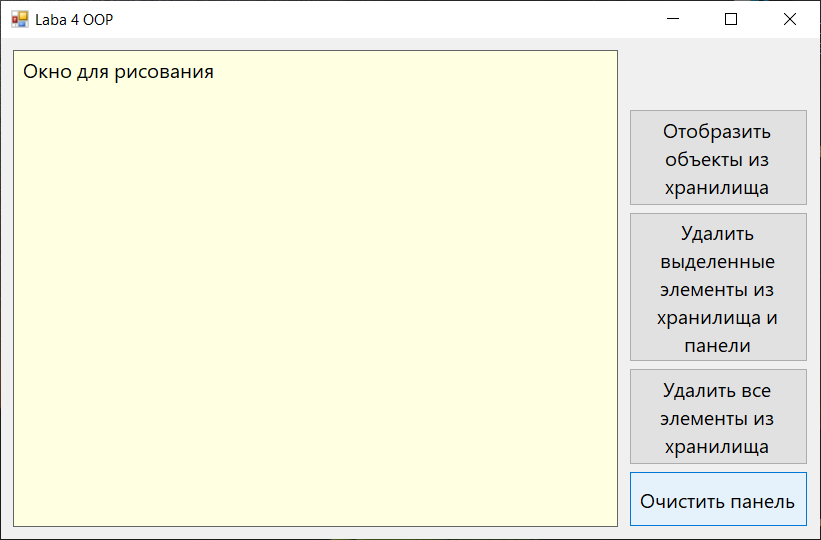


Рис. 10 Демонстрация работы кнопки “Очистить панель”

1. **Вывод:**

В рамках лабораторной работы я разобрался:

* Использовать хранилище для класса Circle
* Отображение на панели различных геометрических фигур
* Выделение нескольких элементов на панели
* Выполнять различные операции с выделенными элементами

Приложение 1. Листинг программы:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace \_4\_laba\_OOP

{

public partial class Main : Form

{

public Main()

{

InitializeComponent();

}

class Circle

{

public int x, y; // Координаты круга

public int rad = 15; // Радиус круга

public Color color = Color.Navy; // Выделен ли элемент

public bool is\_drawed = true; // Нарисован ли круг на полотне

public Circle()

{

x = 0;

y = 0;

}

public Circle(int x, int y)

{

this.x = x-rad;

this.y = y-rad;

}

~Circle() { }

}

private void paint\_box\_MouseMove(object sender, MouseEventArgs e)

{

label\_x.Text = "Координаты X: " + e.X.ToString();

label\_y.Text = "Координаты Y: " + e.Y.ToString();

}

private void paint\_circle(Color name, ref Storage stg, int index)

{ // Рисует круг на панели

Pen pen = new Pen(name, 3);

// Объявляем объект - карандаш, которым будем рисовать контур

if (!storag.check\_empty(index))

{

if (storag.objects[index].is\_drawed == true)

{

paint\_box.CreateGraphics().DrawEllipse(

pen,

stg.objects[index].x,

stg.objects[index].y,

stg.objects[index].rad \* 2,

stg.objects[index].rad \* 2);

stg.objects[index].color = name;

}

}

}

private void remove\_selection\_circle(ref Storage stg)

{ // Снимает выделение у всех элементов хранилища

for (int i = 0; i < k; ++i)

{

if (!storag.check\_empty(i))

{ // Вызываем функцию отрисовки круга

paint\_circle(Color.Navy, ref storag, i);

}

}

}

private void remove\_selected\_circle(ref Storage stg)

{ // Удаляет выделенные элементы из хранилища

for (int i = 0; i < k; ++i)

{

if (!storag.check\_empty(i))

{

if(storag.objects[i].color == Color.Red)

{

storag.delete\_object(i);

}

}

}

}

private void Main\_MouseMove(object sender, MouseEventArgs e)

{

label\_x.Text = "";

label\_y.Text = "";

}

int p = 0; // Нажат ли был ранее Ctrl

static int k = 5; // Кол-во ячеек в хранилище

Storage storag = new Storage(k); // Создаем объект хранилища

static int index = 0; // Кол-во нарисованных кругов

int indexin = 0; // Индекс, в какое место был помещён круг

private void paint\_box\_MouseClick(object sender, MouseEventArgs e)

{

Circle krug = new Circle(e.X, e.Y);

if (index == k)

storag.doubleSize(ref k);

// Проверка на наличие круга на данных координатах

int c = check\_circle(ref storag, k, krug.x, krug.y);

if (c != -1)

{ // Если на этом месте уже нарисован круг

if (Control.ModifierKeys == Keys.Control)

{ // Если нажат ctrl, то выделяем несколько объектов

if (p == 0)

{

paint\_circle(Color.Navy, ref storag, indexin);

p = 1;

}

// Вызываем функцию отрисовки круга

paint\_circle(Color.Red, ref storag, c);

}

else

{ // Иначе выделяем только один объект

// Снимаем выделение у всех объектов хранилища

remove\_selection\_circle(ref storag);

// Вызываем функцию отрисовки круга

paint\_circle(Color.Red, ref storag, c);

}

return;

}

// Добавляем круг в хранилище

storag.add\_object(index, ref krug, k, ref indexin);

// Снимаем выделение у всех объектов хранилища

remove\_selection\_circle(ref storag);

// Вызываем функцию отрисовки круга

paint\_circle(Color.Red, ref storag, indexin);

label\_paintbox.Visible = false;

++index;

p = 0;

}

private int check\_circle(ref Storage stg, int size, int x, int y)

{ // Проверяет есть ли уже круг с такими же координатами в хранилище

if (stg.occupied(size) != 0)

{

for (int i = 0; i < size; ++i)

{

if (!stg.check\_empty(i))

{

int x1 = stg.objects[i].x - 15;

int x2 = stg.objects[i].x + 15;

int y1 = stg.objects[i].y - 15;

int y2 = stg.objects[i].y + 15;

// Если круг есть, возвращет индекс круга в хранилище

if ((x1 <= x && x <= x2) && (y1 <= y && y <= y2))

return i;

}

}

}

return -1;

}

private void button\_clear\_paintbox\_Click(object sender, EventArgs e)

{ // Очищает панель от кругов

paint\_box.Refresh(); // Перерисовывем панель paint\_box

label\_paintbox.Visible = true;

for(int i = 0; i < k; ++i)

{

if (!storag.check\_empty(i))

{ // Меняем is\_drawed на false

storag.objects[i].is\_drawed = false;

}

}

}

class Storage

{

public Circle[] objects;

public Storage(int count)

{ // Конструктор по умолчанию

// Выделяем count мест в хранилище

objects = new Circle[count];

for (int i = 0; i < count; ++i)

objects[i] = null;

}

public void initialisat(int count)

{ // Выделяем count мест в хранилище

objects = new Circle[count];

for (int i = 0; i < count; ++i)

objects[i] = null;

}

public void add\_object(int ind, ref Circle object1, int count, ref int indexin)

{ // Добавляет ячейку в хранилище

// Если ячейка занята ищем свободное место

while (objects[ind] != null)

{

ind = (ind + 1) % count;

}

objects[ind] = object1;

indexin = ind;

}

public void delete\_object(int ind)

{ // Удаляет объект из хранилища

objects[ind] = null;

index--;

}

public bool check\_empty(int index)

{ // Проверяет занято ли место хранилище

if (objects[index] == null)

return true;

else return false;

}

public int occupied(int size)

{ // Определяет кол-во занятых мест в хранилище

int count\_occupied = 0;

for (int i = 0; i < size; ++i)

if (!check\_empty(i))

++count\_occupied;

return count\_occupied;

}

public void doubleSize(ref int size)

{ // Функция для увеличения кол-ва элементов в хранилище в 2 раза

Storage storage1 = new Storage(size \* 2);

for (int i = 0; i < size; ++i)

storage1.objects[i] = objects[i];

for (int i = size; i < (size \* 2) - 1; ++i)

{

storage1.objects[i] = null;

}

size = size \* 2;

initialisat(size);

for (int i = 0; i < size; ++i)

objects[i] = storage1.objects[i];

}

~Storage() { }

};

private void button\_show\_Click(object sender, EventArgs e)

{ // Отобразить все круги из хранилища

paint\_box.Refresh();

if (storag.occupied(k) != 0)

{

label\_paintbox.Visible = false;

for (int i = 0; i < k; ++i)

{

if (!storag.check\_empty(i))

{ // Меняем is\_drawed на true

storag.objects[i].is\_drawed = true;

}

paint\_circle(Color.Navy, ref storag, i);

}

}

}

private void button\_deletestorage\_Click(object sender, EventArgs e)

{ // Удалить все круги из хранилища

for (int i = 0; i < k; ++i)

{

storag.objects[i] = null;

}

index = 0;

}

private void button\_del\_\_item\_storage\_Click(object sender, EventArgs e)

{ // Обработчик на удаление выделенных элементов из хранилища

remove\_selected\_circle(ref storag);

paint\_box.Refresh();

if (storag.occupied(k) != 0)

{

for (int i = 0; i < k; ++i)

{

paint\_circle(Color.Navy, ref storag, i);

}

}

}

}

}

Приложение 2. Скриншот коммитов

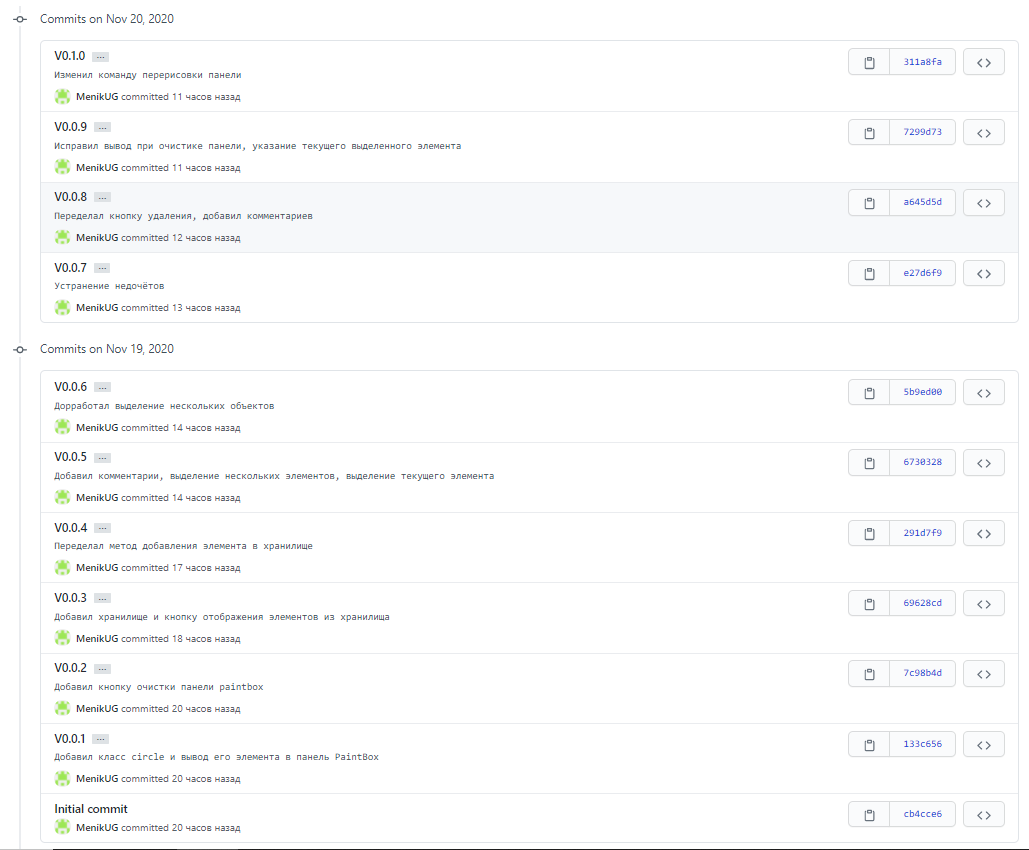


Рис. 11 Коммиты на GitHub

Приложение 3. Ссылка на GitHub

<https://github.com/MenikUG/4_laba_OOP>