Министерство образования и науки российской федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра вычислительной техники

Расчетно-графическая работа

по дисциплине «Программная инженерия»

на тему: «***Применение шаблонов проектирования***»

Факультет: АВТ

Группа: АВТ-309

Студент: Халфина Г.М.

Преподаватель: Романов Е.Л.

Вариант: 5

Новосибирск

2016

Оглавление

[Теоретические сведения 3](#_Toc469901349)

[Описание разработки 5](#_Toc469901350)

[Функционал приложения 5](#_Toc469901351)

[Архитектура приложения 5](#_Toc469901352)

[Описание методов 6](#_Toc469901353)

[Конструктор 6](#_Toc469901354)

[show() 6](#_Toc469901355)

[setXY() 6](#_Toc469901356)

[delete() и deleteR() 6](#_Toc469901357)

[moveAt(e) 6](#_Toc469901358)

[setFocus(flag) 7](#_Toc469901359)

[add() 7](#_Toc469901360)

[resize(obj) 7](#_Toc469901361)

[dataForImage() 7](#_Toc469901362)

[getColor() 7](#_Toc469901363)

[Заключение 7](#_Toc469901364)

[Список литературы 8](#_Toc469901365)

[Приложение (листинг программы) 8](#_Toc469901366)

[figure.js 8](#_Toc469901367)

[client.js 20](#_Toc469901368)

# Теоретические сведения

Паттерн «компоновщик» (composite) относится к классу структурных паттернов проектирования. Паттерн компонует объекты в древовидные структуры, позволяющие представить иерархии вида часть - целое. В такой иерархии объекты могут быть условно разделены на два больших класса: индивидуальных и составных. Индивидуальные объекты (примитивы) представляют собой листовые вершины древовидной иерархии, а составные (контейнеры), напротив, являются внутренними узлами и могут включать в себя другие объекты, как индивидуальные, так и составные. Программа должна по-разному обращаться с примитивами и контейнерами, хотя пользователь, как правило, работает с ними обоих типов единообразно. Необходимость различать эти типы усложняет программу. Компоновщик решает эту проблему, позволяя клиенту единообразно работать и с примитивами, и с контейнерами.

Хорошей демонстрацией паттерна является программа работы с графикой. Такая программа позволяет пользователю сгруппировать мелкие элементы рисунка (линии, геометрические фигуры, надписи и т.д.) в более крупные компоненты, их – в еще более крупные и т.д. Налицо иерархическая структура, описанная в предыдущем абзаце. Рассмотрим на ее примере работу паттерна компоновщик. Общая диаграмма классов для паттерна приведена на рисунке 1.

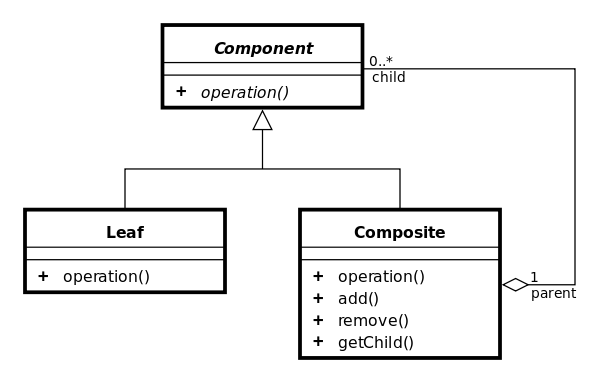


Рисунок 1 – Диаграмма классов паттерна компоновщик

Ключевым является абстрактный класс Component, представляющий одновременно и примитивы, и контейнеры. В нем объявлены операции, специфичные для всех объектов (в нашем примере такой операцией может быть операция отрисовки графического объекта ), а также общие для всех составных объектов, например операции для управления потомками. Функции этого класса:

* объявление интерфейса для копонуемых объектов;
* предоставление подходящей реализации операций по умолчанию, общей для всех классов;
* объявление интерфейса для доступа к потомкам;
* определение интерфейса для доступа к родителю компонента (необязательно).

От класса Component наследуются примитивы (в случае графического редактора сюда можно отнести классы Line, Rectangle, Text и т.п.; на диаграмме они все представлены классом Leaf) и составные объекты (класс Composite на диаграмме). Функции класса Leaf:

* представление листовых узлов в дереве композиции;
* определения поведения примитивов;

Функции класса Composite:

* определение поведения компонентов, имеющих потомков ;
* хранение компонентов - потомков ;
* реализация относящихся к управлению потомками операций класса Component.

Клиент взаимодействует с объектами, используя интерфейс класса Component. Если получателем запроса от клиента является объект Leaf, он и обрабатывает запрос. Если же получателем запроса является составной объект, то он, как правило, перенаправляет запрос своим потомкам, возможно, выполняя некоторые дополнительные операции (например, при отрисовке составного объекта в графическом редакторе вызываются методы отрисовки для всех потомков этого объекта).

Результаты применения паттерна:

* компоновщик определяет иерархии классов, состоящие из примитивных и составных объектов. Из примитивных объектов можно составлять более сложные, которые, в свою очередь, участвуют в еще более сложны композициях и т.д. Любой клиент, ожидающий примитивного объекта, может работать и с составным;
* компоновщик упрощает архитектуру клиента. Клиенты могут единообразно работать, как с примитивами, так и с составными объектами. Обычно клиенту неизвестно, работает ли он с индивидуальным или с составным объектом. Это упрощает код клиента, поскольку нет необходимости писать функции, ветвящиеся в зависимости от типа объекта, с которым они работают;
* компоновщик облегчает добавление новых видов компонентов. Новые подклассы классов Composite и Leaf будут автоматически работать с уже существующими структурами и клиентским кодом.
* компоновщик способствует созданию общего дизайна.

# Описание разработки

## Функционал приложения

В рамках работы реализовано web-приложение графический редактор, с функционалом:

* Создание объектов прямоугольник, треугольник, круг
* Группировка/разгруппировка объектов
* Изменение размера объектов
* Изменение порядка отображения (на передний/задний план)
* Перетаскивание объектов
* Выбор объектов по клику или нескольких объектов, попадающих в прямоугольник
* Сохранение изображения в формате jpg.

Приложение реализовано на языке JavaScript. Это [прототипно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%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_%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) [сценарный](https://ru.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%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%D1%8F) со слабой типизацией. Для поддержки функционала создания изображения в формате jpg используется серверный язык программирования PHP.

## Архитектура приложения

Приложение реализует архитектуру «толстого клиента», так как весь функционал, кроме создания jpg-файла, работает в браузере. Модель с данными – иерархическая структура, реализующая шаблон «компоновщик». JS-код обрабатывает пользовательские события DOM-дерева документа, обращается к модели. Для создания jpg-файла JS-код делает GET-запрос на сервер, к PHP-скрипту. В качестве параметра запроса выступает JSON-объект, содержащий информацию, необходимую для отрисовки (координаты, размеры, цвет и тип фигур). PHP-скрипт создает jpg-файл с нарисованными на нём фигурами и возвращает браузеру. Браузер сразу сохраняет файл в директорию (действие по-умолчанию).

Помимо шаблона «компоновщик» при разработке использовался шаблон «фабрика». Для создания объектов фигур JS-коду доступна коллекция из объектов-фабрик. Конструкторы объектов инкапсулированы в анонимной функции и не доступны из JS-кода.

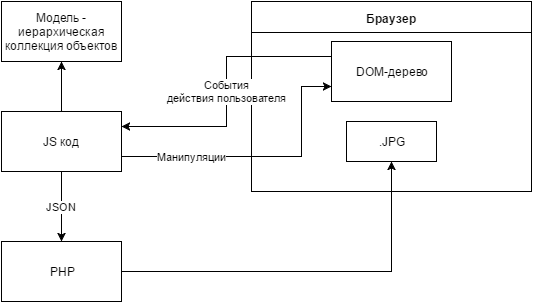


Рисунок 2 – Структура приложения

C:\Users\Галина\Downloads\Untitled Diagram.png

Рисунок 3 - Диаграмма объектов

## Описание методов

Методы общие для всех фигур, поэтому хранятся в объекте Figure. Если для какой-то фигуры нужен специфичный метод, он переопределяется. «Листовыми» объектами в программе являются объекты прямоугольник, круг, треугольник. Объект типа контейнер используется для группировки объектов.

### Конструктор

В конструкторе объявляются и инициализируются переменные, создаётся узел DOM-дерева htmlElm, который будет отображаться на странице. Для htmlElm задаётся CSS-класс, внешний вид (размеры, цвет), генерируется уникальный id.

### show()

Помещает htmlElm в DOM-дерево потомком по отношению к parent. Для контейнера рекурсивно вызывается для каждого дочернего объекта.

### setXY()

При изменении координат объекта нужно вызвать этот метод, он отображает координаты.

### delete() и deleteR()

Оба этих метода удаляют объект, но deleteR() удаляет в том числе всех потомков для контейнера. При вызове delete на контейнере, возвращается массив его дочерних объектов.

### moveAt(e)

Отвечает за движение объекта, меняет координаты. E – объект события.

### setFocus(flag)

Устанавливает фокус объекта (если он выделен flag = true) и всех дочерних объектов.

### add()

Метод заглушка для «листовых» объектов, добавляет в контейнер новый объект.

### resize(obj)

Отвечает за изменения размера объектов (кроме контейнера). В качестве параметра подаётся объект obj с полями x1,y1,x2,y2, обозначающими верхний левый и правый нижний углы объекта. По ним определяются размеры и координаты объекта.

### dataForImage()

Возвращает объект с данными для рисования объекта на картинке (координаты, размер, цвет, тип объекта). Приложение формирует

### getColor()

Возвращает цвет объекта.

# Заключение

В ходе работы был изучен и реализован паттерн компоновщик. Он удобен в использовании в случаях работы с древовидными структурами, чтобы обеспечивать доступ и к примитивам, и к составным объектам.

# Список литературы

1. Флэнаган Д. JavaScript. Подробное руководство, 6-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2016. – 1080с., ил.
2. Гамма Э., Хелм Р., Джонсон Р. , Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб: Питер , 2001 . — 368 с.: ил.

# Приложение (листинг программы)

## figure.js

var randomMethods = {

getRandom: function(){

return parseInt(Math.random()\*1000)+(+new Date());

},

getRandomColor: function(){

return "#" + this.getRandomArbitary(50,255).toString(16)+ this.getRandomArbitary(50,255).toString(16)+ this.getRandomArbitary(50,255).toString(16);

},

getRandomArbitary: function(min, max)

{

return Math.floor(Math.random() \* (max - min + 1)) + min;

}

};

var FigureCreators = (function(){

function setShell(element){

var shell = $(document.createElement("div"));

shell.append(element);

var l = $(document.createElement("div"));

l.addClass("l");

shell.append(l);

var r = $(document.createElement("div"));

r.addClass("r");

shell.append(r);

var t = $(document.createElement("div"));

t.addClass("t");

shell.append(t);

var d = $(document.createElement("div"));

d.addClass("d");

shell.append(d);

var tl = $(document.createElement("div"));

tl.addClass("tl");

shell.append(tl);

var tr = $(document.createElement("div"));

tr.addClass("tr");

shell.append(tr);

var dl = $(document.createElement("div"));

dl.addClass("dl");

shell.append(dl);

var dr = $(document.createElement("div"));

dr.addClass("dr");

shell.append(dr);

return shell;

}

function Figure() {

this.show = function(context){

if(context !== undefined){

this.parent = context;

}

else{

if(this.parent === null)

this.parent = $("#objects-area");

}

this.htmlElm.detach();

this.parent.append(this.htmlElm);

for(figure in this.collection){

this.collection[figure].show(this.htmlElm);

}

}

this.setXY = function(){

this.htmlElm.css("left", this.x + 'px');

this.htmlElm.css("top", this.y + 'px');

}

this.delete= function(){

var returnObj = {};

this.htmlElm.detach();

for(i in this.collection){

var figure = this.collection[i];

this.collection[i].x += this.x;

this.collection[i].y += this.y;

this.collection[i].setXY();

returnObj[this.collection[i].id] = this.collection[i];

this.collection[i].show(this.parent);

}

return returnObj;

}

this.deleteR= function(){

this.htmlElm.detach();

for(i in this.collection){

this.collection[i].deleteR();

}

}

this.moveAt = function(e) {

this.x = e.pageX - this.shiftX;

this.y = e.pageY - this.shiftY;

this.setXY();

}

this.setFocus = function(flag){

if(flag){

this.htmlElm.addClass("focus-figure");

for(figure in this.collection){

this.collection[figure].setFocus(flag);

}

}

else{

this.htmlElm.removeClass("focus-figure");

for(figure in this.collection){

this.collection[figure].setFocus(flag);

}

}

}

this.add = function(figure){};

this.startMove = function(){

for(figure in this.collection){

this.collection[figure].shiftX = 0;

this.collection[figure].shiftY = 0;

this.collection[figure].startMove();

}

}

this.resize = function(obj){

if(obj.x2 !== undefined && obj.x2 > this.x){

this.width = obj.x2 - this.x;

this.htmlElm.css("width",this.width);

}

if(obj.x1 !== undefined && obj.x1 < this.x + this.width){

this.width += this.x - obj.x1;

this.x = obj.x1;

this.htmlElm.css("left",this.x);

this.htmlElm.css("width",this.width);

}

if(obj.y2 !== undefined && obj.y2 > this.y){

this.height = obj.y2 - this.y;

this.htmlElm.css("height",this.height);

}

if(obj.y1 !== undefined && obj.y1 < this.y + this.height){

this.height += this.y - obj.y1;

this.y = obj.y1;

this.htmlElm.css("height",this.height);

this.htmlElm.css("top",this.y);

}

}

this.dataForImage = function(parentX,parentY){

var dataArray = new Array();

var data = {};

data["x"] = this.x + parentX;

data["y"] = this.y + parentY;

data["type"] = this.htmlElm[0].className.split(" ")[0];

data["width"] = this.width;

data["height"] = this.height;

data["color"] = this.getColor();

dataArray.push(data);

return dataArray;

}

this.getColor = function(){

var mc = this.htmlElm.children(".fig").css("background-color").match(/\d+/g);

return {

r: mc[0],

g: mc[1],

b: mc[2]

};

}

}

function Rectangle(){

this.x = 0;

this.y = 0;

this.height = 60;

this.width = 100;

this.shiftX = 0;

this.shiftY = 0;

this.parent = null;

this.id = randomMethods.getRandom();

this.htmlElm = $(document.createElement("div"));

this.htmlElm.addClass("fig");

this.htmlElm.css("background-color",randomMethods.getRandomColor());

this.htmlElm = setShell(this.htmlElm);

this.htmlElm.addClass("rectangle");

this.htmlElm.css("height", this.height);

this.htmlElm.css("width", this.width);

this.htmlElm.attr("data-id",this.id);

this.collection = {};

}

Rectangle.prototype = new Figure();

Rectangle.prototype.constructor = Rectangle;

function Circle(){

this.x = 0;

this.y = 0;

this.height = 60;

this.width = 60;

this.shiftX = 0;

this.shiftY = 0;

this.parent = null;

this.id = randomMethods.getRandom();

this.htmlElm = $(document.createElement("div"));

this.htmlElm.addClass("fig");

this.htmlElm.css("background-color",randomMethods.getRandomColor());

this.htmlElm = setShell(this.htmlElm);

this.htmlElm.addClass("circle");

this.htmlElm.css("height", this.height);

this.htmlElm.css("width", this.width);

this.htmlElm.attr("data-id",this.id);

this.collection = {};

}

Circle.prototype = new Figure();

Circle.prototype.constructor = Circle;

function Triangle() {

this.x = 0;

this.y = 0;

this.shiftX = 0;

this.shiftY = 0;

this.height = 60;

this.width = 60;

this.parent = null;

this.id = randomMethods.getRandom();

this.htmlElm = $(document.createElement("div"));

this.htmlElm.addClass("fig");

this.htmlElm.css("border-bottom-color",randomMethods.getRandomColor());

this.htmlElm.css("border-bottom-width",this.height);

this.htmlElm.css("border-left-width",this.width/2);

this.htmlElm.css("border-right-width",this.width/2);

this.htmlElm = setShell(this.htmlElm);

this.htmlElm.addClass("triangle");

this.htmlElm.css("height", this.height);

this.htmlElm.css("width", this.width);

this.htmlElm.attr("data-id",this.id);

this.collection = {};

this.resize = function(obj){

if(obj.x2 !== undefined && obj.x2 > this.x){

this.width = obj.x2 - this.x;

this.htmlElm.css("width",this.width);

var fig = this.htmlElm.children(".fig");

fig.css("border-right-width",this.width/2);

fig.css("border-left-width",this.width/2);

}

if(obj.x1 !== undefined && obj.x1 < this.x + this.width){

this.width += this.x - obj.x1;

this.x = obj.x1;

this.htmlElm.css("left",this.x);

this.htmlElm.css("width",this.width);

var fig = this.htmlElm.children(".fig");

fig.css("border-right-width",this.width/2);

fig.css("border-left-width",this.width/2);

}

if(obj.y2 !== undefined && obj.y2 > this.y){

this.height = obj.y2 - this.y;

this.htmlElm.css("height",this.height);

var fig = this.htmlElm.children(".fig");

fig.css("border-bottom-width",this.height);

}

if(obj.y1 !== undefined && obj.y1 < this.y + this.height){

this.height += this.y - obj.y1;

this.y = obj.y1;

this.htmlElm.css("height",this.height);

this.htmlElm.css("top",this.y);

var fig = this.htmlElm.children(".fig");

fig.css("border-bottom-width",this.height);

}

}

this.getColor = function(){

var mc = this.htmlElm.children(".fig").css("border-bottom-color").match(/\d+/g);

return {

r: mc[0],

g: mc[1],

b: mc[2]

}

}

}

Triangle.prototype = new Figure();

Triangle.prototype.constructor = Triangle;

function Composite(x1,y1,x2,y2){

this.x = x1;

this.y = y1;

this.height = y2 - y1;

this.width = x2 - x1;

this.shiftX = 0;

this.shiftY = 0;

this.parent = null;

this.id = randomMethods.getRandom();

this.htmlElm = $(document.createElement("div"));

this.htmlElm.addClass("composite");

this.htmlElm.attr("data-id",this.id);

this.htmlElm.css("left",this.x+"px");

this.htmlElm.css("top",this.y+"px");

this.htmlElm.css("width",this.width+"px");

this.htmlElm.css("height",this.height+"px");

this.collection = {};

this.add = function(figure){

figure.x -= this.x;

figure.y -= this.y;

figure.setXY();

this.collection[figure.id] = figure;

};

this.resize = function(){};

this.dataForImage = function(parentX, parentY){

var dataArray = new Array();

for(i in this.collection){

dataArray = dataArray.concat(this.collection[i].dataForImage(this.x+parentX, this.y+parentY));

}

return dataArray;

}

}

Composite.prototype = new Figure();

Composite.prototype.constructor = Composite;

function FigureCreator() {this.factoryMethod=null;}

function RectangleCreator() {

this.factoryMethod=function(){return new Rectangle();};

}

RectangleCreator.prototype = new FigureCreator();

RectangleCreator.prototype.constructor = RectangleCreator;

function CircleCreator() {

this.factoryMethod=function(){return new Circle();};

}

CircleCreator.prototype = new FigureCreator();

CircleCreator.prototype.constructor = CircleCreator;

function TriangleCreator() {

this.factoryMethod=function(){return new Triangle();};

}

TriangleCreator.prototype = new FigureCreator();

TriangleCreator.prototype.constructor = TriangleCreator;

function CompositeCreator(x1,y1,x2,y2) {

this.factoryMethod=function(x1,y1,x2,y2){return new Composite(x1,y1,x2,y2);};

}

CompositeCreator.prototype = new FigureCreator();

CompositeCreator.prototype.constructor = CompositeCreator;

return {

rectangle: new RectangleCreator(),

triangle: new TriangleCreator(),

composite: new CompositeCreator(),

circle: new CircleCreator()

};

})()

## client.js

var figuresList = {};

(function($,undefined){

var mouseDown = false;

var shiftX = 0;

var shiftY = 0;

var startY = 0, startX = 0;

var focusObj = null;

function coords(e){

var posx = 0;

var posy = 0;

if (e.pageX || e.pageY) {

posx = e.pageX;

posy = e.pageY;

}

else if (e.clientX || e.clientY) {

posx = e.clientX + document.body.scrollLeft

+ document.documentElement.scrollLeft;

posy = e.clientY + document.body.scrollTop

+ document.documentElement.scrollTop;

}

return new Array(posx,posy);

}

function doSelection(x1,y1,x2,y2){

focusObj = FigureCreators["composite"].factoryMethod(x1,y1,x2,y2);

for(figure in figuresList){

if(figuresList[figure].x > x1 && figuresList[figure].x < x2 && figuresList[figure].y > y1 && figuresList[figure].y < y2){

focusObj.add(figuresList[figure]);

}

}

focusObj.setFocus(true);

focusObj.show();

}

$("#add-rectangle").click(function(e){

e.preventDefault();

var rect = FigureCreators["rectangle"].factoryMethod();

figuresList[rect.id] = rect;

rect.show();

});

$("#add-circle").click(function(e){

e.preventDefault();

var circle = FigureCreators["circle"].factoryMethod();

figuresList[circle.id] = circle;

circle.show();

});

$("#add-triangle").click(function(e){

e.preventDefault();

var tri = FigureCreators["triangle"].factoryMethod();

figuresList[tri.id] = tri;

tri.show();

});

$("#delete").click(function(e){

e.preventDefault();

if(focusObj !== null){

if(figuresList[focusObj.id] !== undefined){

focusObj.deleteR();

delete figuresList[focusObj.id];

}

else{

var m = focusObj.delete();

for(i in m){

if(figuresList[m[i].id] !== undefined){

delete figuresList[m[i].id];

m[i].deleteR();

}

}

}

focusObj = null;

}

});

$("#clear").click(function(e){

e.preventDefault();

for(i in figuresList){

figuresList[i].deleteR();

delete figuresList[i];

}

if(focusObj !== null){

focusObj.deleteR();

focusObj = null;

}

});

$("#group").click(function(e){

e.preventDefault();

if(focusObj !== null){

figuresList[focusObj.id] = focusObj;

for(i in focusObj.collection){

if(figuresList[focusObj.collection[i].id] !== undefined){

delete figuresList[focusObj.collection[i].id];

}

}

focusObj = null;

}

});

$("#ungroup").click(function(e){

e.preventDefault();

if(focusObj !== null){

var content = focusObj.delete();

for(i in content){

figuresList[content[i].id] = content[i];

content[i].setFocus(false);

content[i].show();

}

focusObj = null;

}

});

$("#down").click(function(e){

e.preventDefault();

if(focusObj !== null){

var z = parseInt(focusObj.htmlElm.css("z-index"));

if(z === 0) return;

focusObj.htmlElm.css("z-index",--z);

}

});

$("#up").click(function(e){

e.preventDefault();

if(focusObj !== null){

var z = parseInt(focusObj.htmlElm.css("z-index"));

focusObj.htmlElm.css("z-index",++z);

}

});

$("#getImg").click(function(e){

e.preventDefault();

var ajaxObj = new Array();

for(i in figuresList){

ajaxObj = ajaxObj.concat(figuresList[i].dataForImage(0,0));

}

ajaxObj = JSON.stringify(ajaxObj);

console.log(ajaxObj);

document.location = 'getimage.php?values=' + ajaxObj;

});

$("#objects-area").on("mousedown",".rectangle,.triangle,.circle,.composite",function(e){

e.stopPropagation();

if(focusObj !== null){

focusObj.setFocus(false);

if(focusObj.id != e.currentTarget.dataset.id && figuresList[focusObj.id] === undefined){

focusObj.delete();

focusObj = null;

}

}

if(figuresList[e.currentTarget.dataset.id] !== undefined && focusObj === null)

focusObj = figuresList[e.currentTarget.dataset.id];

else if(focusObj === null)

return;

focusObj.shiftX = e.pageX - e.currentTarget.getBoundingClientRect().left;

focusObj.shiftY = e.pageY - e.currentTarget.getBoundingClientRect().top;

focusObj.setFocus(true);

focusObj.startMove();

document.onmousemove = function(e) {

focusObj.moveAt(e);

};

focusObj.htmlElm.ondragstart = function(){

return false;

}

$("#objects-area").on("mouseup","div:not(#select-frame)",function(e){

document.onmousemove = null;

$("#objects-area").off("mouseup","div");

});

});

$("#objects-area").on("mousedown",function(e){

if(focusObj !== null){

focusObj.setFocus(false);

if(figuresList[focusObj.id] === undefined){

focusObj.delete();

}

focusObj = null;

}

mouseDown = true;

var mousexy = coords(e);

startX = mousexy[0];

startY = mousexy[1];

document.onmousemove = function(e){

var mousexy = coords(e);

var x1 = startX;

var y1 = startY;

var x2 = mousexy[0];

var y2 = mousexy[1];

if (x1 == x2 || y1 == y2) return;

if (x1 > x2){

x1 = x1+x2;

x2 = x1-x2;

x1 = x1-x2;

}

if (y1 > y2){

y1 = y1+y2;

y2 = y1-y2;

y1 = y1-y2;

}

var sframe = $("#select-frame");

sframe.css("top",y1+"px");

sframe.css("left",x1+"px");

sframe.css("width",x2-x1+"px");

sframe.css("height", y2-y1+"px");

sframe.css("visibility", mouseDown?'visible':'hidden');

};

document.onmouseup = function(e){

if (!e) var e = window.event;

mouseDown = false;

var mousexy = coords(e);

var x1 = startX;

var y1 = startY;

var x2 = mousexy[0];

var y2 = mousexy[1];

if (x1 > x2){

x1 = x1+x2;

x2 = x1-x2;

x1 = x1-x2;

}

if (y1 > y2){

y1 = y1+y2;

y2 = y1-y2;

y1 = y1-y2;

}

if(x1 != x2 || y1 != y2)

doSelection(x1,y1,x2,y2);

document.getElementById('select-frame').style.visibility = mouseDown?'visible':'hidden';

document.onmousemove = null;

document.onmouseup = null;

}

return false;

});

$("#objects-area").on("mousedown",".r",function(e){

e.preventDefault();

e.stopPropagation();

document.onmousemove = function(e){

focusObj.resize({x2:e.pageX});

}

$("#objects-area").on("mouseup",".r",function(e){

document.onmousemove = null;

});

});

$("#objects-area").on("mousedown",".l",function(e){

e.preventDefault();

e.stopPropagation();

document.onmousemove = function(e){

focusObj.resize({x1:e.pageX});

}

$("#objects-area").on("mouseup",".l",function(e){

document.onmousemove = null;

});

});

$("#objects-area").on("mousedown",".t",function(e){

e.preventDefault();

e.stopPropagation();

document.onmousemove = function(e){

focusObj.resize({y1:e.pageY});

}

$("#objects-area").on("mouseup",".t",function(e){

document.onmousemove = null;

});

});

$("#objects-area").on("mousedown",".d",function(e){

e.preventDefault();

e.stopPropagation();

document.onmousemove = function(e){

focusObj.resize({y2:e.pageY});

}

$("#objects-area").on("mouseup",".d",function(e){

document.onmousemove = null;

});

});

$("#objects-area").on("mousedown",".tr",function(e){

e.preventDefault();

e.stopPropagation();

document.onmousemove = function(e){

focusObj.resize({x2:e.pageX,y1:e.pageY});

}

$("#objects-area").on("mouseup",".tr",function(e){

document.onmousemove = null;

});

});

$("#objects-area").on("mousedown",".tl",function(e){

e.preventDefault();

e.stopPropagation();

document.onmousemove = function(e){

focusObj.resize({x1:e.pageX,y1:e.pageY});

}

$("#objects-area").on("mouseup",".tl",function(e){

document.onmousemove = null;

});

});

$("#objects-area").on("mousedown",".dr",function(e){

e.preventDefault();

e.stopPropagation();

document.onmousemove = function(e){

focusObj.resize({x2:e.pageX,y2:e.pageY});

}

$("#objects-area").on("mouseup",".dr",function(e){

document.onmousemove = null;

});

});

$("#objects-area").on("mousedown",".dl",function(e){

e.preventDefault();

e.stopPropagation();

document.onmousemove = function(e){

focusObj.resize({x1:e.pageX,y2:e.pageY});

}

$("#objects-area").on("mouseup",".dl",function(e){

document.onmousemove = null;

});

});

})(jQuery);