Санкт-Петербургский государственный университет
Фундаментальная информатика и информационные технологии
Математическое и программное обеспечение вычислительных машин,
комплексов и компьютерных сетей

Усманов Шавкат Камилович

Построение двухмерных поверхностей на основе сплайновых аппроксимаций

Выпускная квалификационная работа

Научный руководитель:
д.ф. - м.н., профессор Бурова И. Г.

Рецензент:
Домнин Н. К.

Санкт-Петербург
2017

Saint-Petersburg State University
Fundamental Informatics and Information Technology
Mathematical and Software Support for Computers, Computer System and Networks

Usmanov Shavkat Kamilovich

Construction of two-dimensional surfaces based on spline approximations

Graduation project

Scientific supervisor:
professor I. G. Burova

Reviwer:
N. K. Domnin

Saint-Petersburg
2017

Оглавление

Цель работы

Целью данной магистерской диссертации является визуализация двухмерного изображения на языке Java, в качестве предмета исследования, сплайновые апроксимации.

Для визуализации изображения необходимо решить следующие задачи:

1) Изучить особенности визуализации линий и поверхностей с помощью сплайнов;

2) Разработать приложение для платформы android, позволяющее передвигать и вращать плоские треугольники на экране;

3) Разработать приложение для платформы android, по построению интерактивной триангуляции на плоскости, позволяющее пользователю фиксировать узлы триангуляции на плоскости и записывать координаты этих узлов в текстовый файл, с целью последующего использования для построения поверхностей.

Введение

Решение задачи построения поверхности в двумерном пространстве будем строить с помощью сплайн-функций. Сплайн-функция — это новая быстро развивающаяся область теории приближения функций и численного анализа. Получив распространение в 60-х годах, главным образом как средство интерполяции сложных кривых, сплайны в дальнейшем стали важным методом для решения разнообразных задач вычислительной математики и прикладной геометрии [4].

По сравнению с классическим аппаратом приближения многочленами сплайн-функции обладают по крайней мере двумя важными преимуществами. Во-первых, бесспорно,лучшими аппроксимативными свойствами и, во-вторых, удобством реализации построенных на их основе алгоритмов на ЭВМ [4].

Большинство численных методов решения задач математического анализа так или иначе связано с аппроксимацией функций. Это и собственно задачи приближения функций (интерполяция, сглаживание, наилучшие приближения) и задачи, в которых аппроксимация присутствует как промежуточный этап исследования (численное дифференцирование и интегрирование, численное решение дифференциальных и интегральных уравнений).

В алгебраическом смысле сплайны трактуются как некоторые гладкие кусочно-мпогочленные (включая обобщенные многочлены) функции с однородной структурой. Сюда относятся так называемые L — сплайны, составляемые из решений линейного однородного дифференциального уравнения $LS(x) = 0$. Случай кубических сплайнов соответствует $L = d^4/dx^4$[4].

В последнее время в вычислительной практике широкое распространение получили B-сплайны (от английского слова bell — колокол). Эти сплайны сосредоточены на конечном носителе. Они используются как для интерполяции функций, так и в качестве базисных функций при построении методов типа конечных элементов. Данная категория сплайнов является наиболее используемой, и функции B-сплайнов широко применяются в системах автоматизированного проектирования и многих пакетах графического программирования.

Подобно сплайнам Безье, B-сплайны генерируются путём аппроксимации набора контрольных точек. В то же время, B-сплайны обладают двумя

преимуществами по сравнению со сплайнами Безье: во-первых, степень полинома B-сплайна можно задать независимо от числа контрольных точек (с определенными ограничениями); во-вторых, B-сплайны допускают локальный контроль над формой кривой. Платой за это является большая сложность B-сплайнов по сравнению со сплайнами Безье [21]. Базисные сплайны заданной степени являются линейно независимыми функциями и образуют базис в функциональных пространствах, что можно использовать для представления с их помощью других функций этих же пространств. Любая, например, кусочно-постоянная функция на отрезке, составленном из равных интервалов, может быть единственным образом представлена как линейная комбинация B-сплайнов нулевой степени, любая кусочно-линейная функция — B-сплайнов первой степени и т. д. Базисные сплайны играют существенную роль при построении численных методов решения задач математической физики, например, метода конечных элементов в теории приближения функций, при решении задач компьютерной графики.

Решение задач аппроксимации и изучение аппроксимативных свойств сплайнов при этом сводятся к исследованию линейных алгебраических систем. Вопрос об экстремальных свойствах является здесь производным в том смысле, что отыскиваются постановки вариационных задач, решениями которых были бы сплайн-функции.

Обращаясь к сплайн-функциям многих переменных, приходится признать, что если мы хотим сохранить для них выше указанные свойства, установленные для одномерных сплайнов, то неизбежно должны ограничиться функциями с клеточной структурой. Под этим мы понимаем функции, область определения которых разделена на ячейки (в плоском случае прямоугольники, треугольники и т. п., в многомерном — параллелепипеды, пирамиды и т. п.). В каждой ячейке функция определена в некотором смысле однородным способом с условиями гладкости вдоль границ ячеек. При интерполировании функций многих переменных для сплайнов, в отличие от мпогочленов, не возникает особых трудностей с проблемой существования и единственности решения. Опыт применения сплайн-функций как аппарата приближения функций в численном анализе показывает, что во всех известных случаях удавалось добиться ощутимых результатов по сравнению с классическим аппаратом многочленов. В одних задачах переход к сплайнам приводит к повышению точности результатов, в других — к значительному сокращению вычислительных затрат, в третьих — достигаются оба эффекта одновременно. Наконец, с помощью сплайнов удалось решить и такие задачи, которые другим путем решить было бы невозможно. Среди них на первом месте стоит проблема представления и хранения геометрической информации в самых различных областях знания, будь то естественные на-

уки, техника, архитектура, картография. Традиционно в более или менее сложных ситуациях эта задача решается путем изображения объекта или процесса на плоскости в виде графиков, чертежей и т. п. Вследствие ограниченности масштабов изображений этот способ принципиально не может обеспечить требуемую точность во всех случаях. Применение для данных целей сплайнов (как одной, так и многих переменных) позволяет хранить геометрическую информацию в числовой форме и с любой точпостыо. При обработке информации на ЭВМ использование сплайнов позволяет на единой методологической основе разрабатывать математическое обеспечение средств машинной графики (графопостроители и дисплеи).

Особенно широкое применение получили сплайны в технике как аппарат для математического моделирования поверхностей деталей и агрегатов сложной формы, таких, как аэродинамические обводы летательных аппаратов, корпуса судов, лопасти гидротурбин, кузова легковых автомобилей и т. п. Такие математические модели стали необходимыми при создании систем автоматизации проектирования изделий на основе ЭВМ, технологической подготовки их производства, включая разработку программ для оборудования с цифровым программным управлением.

Постановка задачи

Известны различные способы построения триангуляции[36], в ряде случаев пользователю необходимо иметь возможность первоначальную триангуляцию строить интерактивно.

Задача состоит в разработке комплекса программ, позволяющих строить триангуляцию на плоскости[38], задавая последовательно узлы, и изменяя расположение последнего задаваемого узла. При этом массив узлов треугольников должен записываться во внешний файл.

Имея триангуляцию на плоскости, зная координаты узлов и значения функции в узлах, далее строим изображение поверхности.

Глава 1. Основные сведения о теории сплайнов. Описание минимальных локальных полинимиальных сплайнов

1.1 Построение кусочно-линейных сплайнов одной переменной

Пусть a, b — вещественные числа, $\{x_j\}$ — упорядоченная сетка узлов на промежутке $[a, b]$(рис.1).

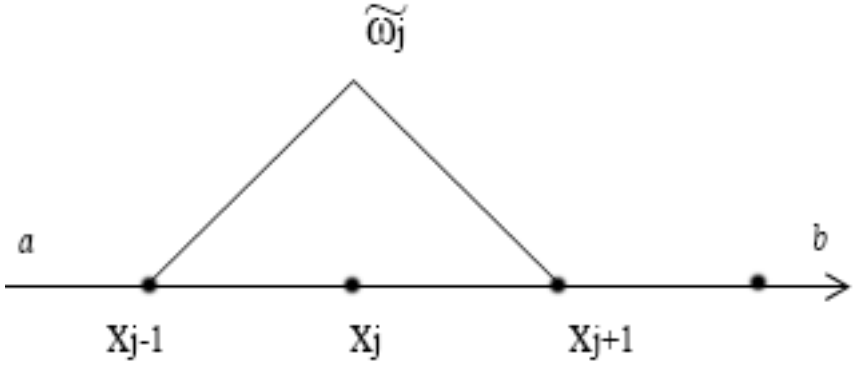

Рис.1

Пусть функция $U(x)$, такова что $U \in \mathrm{C}^2[a, b]$, обозначим $\mathrm{u}_k = U(x_k)$.

Аппроксимацию функции $U(x)$ строим отдельно на каждом промежутке $[x_j,x_{j+1}]$:

$$\widetilde{U}(x) = u(x_j)\varpi_j(x) + u(x_{j+1})\varpi_{j+1}(x), \tag{1}$$

где базисные функции — $\varpi_j(x)$, $\varpi_{j+1}(x)$, таковы что $supp\ \varpi_j = [x_{j-1}, x_{j+1}]$.

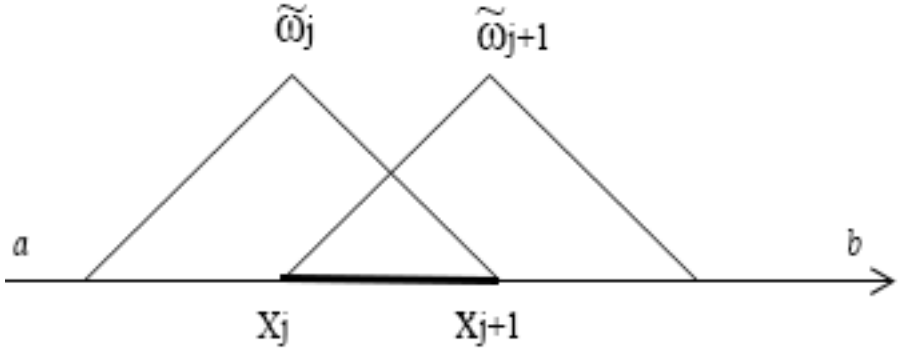

Рис.2

Известно, что кусочно-линейный сплайн, задается формулами (см.[1])

$$\varpi_j(x) = \begin{cases} \dfrac{(x - x_{j+1})}{(x_j - x_{j+1})}, & x \in [x_j, x_{j+1}], \\ \dfrac{(x - x_{j-1})}{(x_j - x_{j-1})}, & x \in [x_{j-1}, x_j], \\ 0, & x \notin [x_{j-1}, x_{j+1}]. \end{cases} \tag{2}$$

График функции $\varpi_j(x)$ изображен на рис.1. Для апроксимации (1) используются две базисные функции $\varpi_j(x)$ и $\varpi_{j+1}(x)$(рис.2).

Из (2) следует,что

$$\varpi_{j+1}(x) = \frac{(x - x_j)}{(x_{j+1} - x_j)}, x \in [x_j, x_{j+1}]. \tag{3}$$

1.2 Построение изображений поверхностей с помощью функций Куранта

Для построения изображения поверхности строим прямоугольную сетку узлов на плоскости. Пусть c, d — вещественные числа. Вдоль оси Y на промежутке $[c, d]$ строим упорядоченную сетку узлов $\{y_k\}$, и проводим прямые параллельные оси X. На координатной оси X отмечаем точки x_j и через эти точки проводим прямые параллельные оси Y(см. рис. 5), точки пересечения линий называем узлами сетки и обозначаем их через (x_j,y_k).

Считаем, что поверхность $U(x, y)$ задаётся значениями в узлах сетки. Отмечаем на плоскости точки пересечения (x_j, y_k), $(x_j, y_{k+1}), (x_{j+1}, y_k)$,

(x_{j+1}, y_{k+1}). Прямоугольник с этими вершинами называем элементарным и обозначаем его P_{jk} (рис.4). Приближение $\widetilde{U}(x, y)$ строим отдельно в каждом элемантарном прямоуголнике.

В каждом узле сетки (x_j, y_k) задано значение функции $U(x_j, y_k)$. Для построения поверхности используем мультипликативные базисные функции Куранта (рис.3) :

$$\Omega_{jk}(x, y) = \varpi_j(x)\varpi_k(y). \tag{4}$$

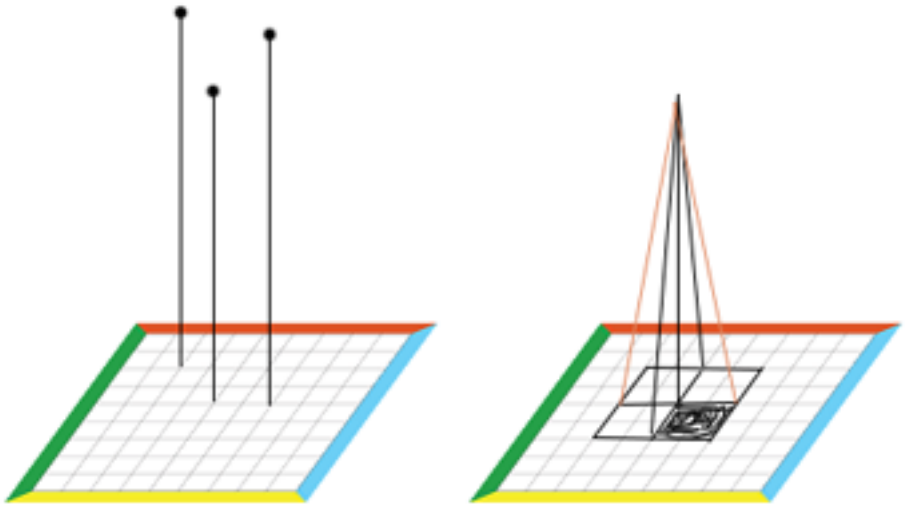

Рис.3

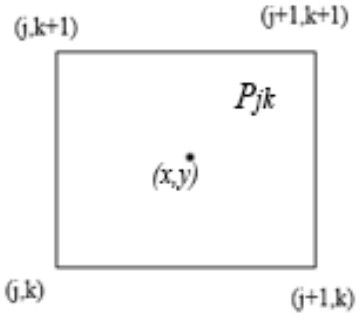

Рис.4

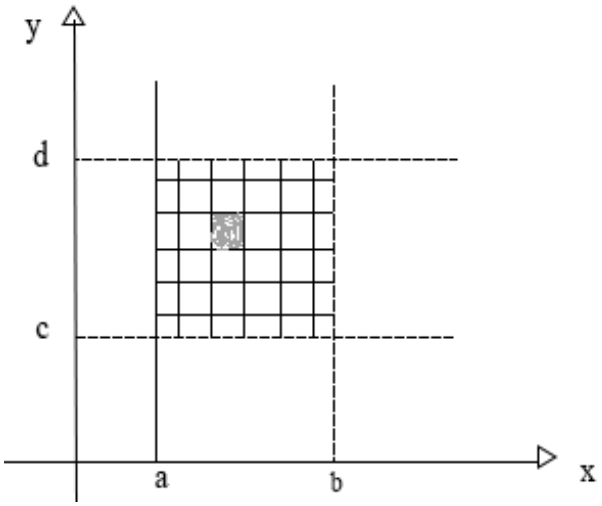

Рис.5

$$\widetilde{U}(x,y) = U(x_j, y_k)\varpi_j(x)\varpi_k(y) + U(x_{j+1}, y_k)\varpi_{j+1}(x)\varpi_k(y) +$$

$$+U(x_j, y_{k+1})\varpi_j(x)\varpi_{k+1}(y) + U(x_{j+1}, y_{k+1})\varpi_{j+1}(x)\varpi_{k+1}(y), (x,y) \in P_{jk}. \quad (5)$$

1.3 Построение кубических сплайнов

Пусть a, b — вещественные числа, $\{X_j\}$-упорядоченная сетка узлов на промежутке $[a,b]$(рис.6).

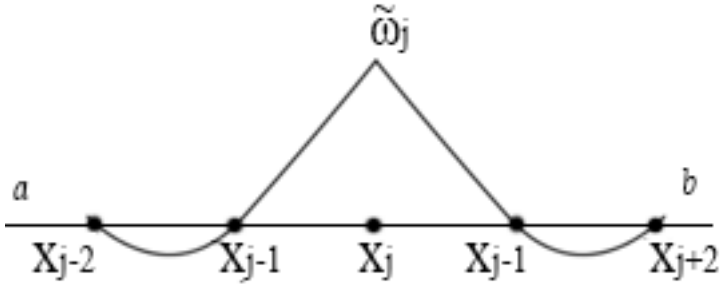

Рис.6

Пусть функция $U(x)$, такова что $U \in C^3[a,b]$.

Аппроксимацию функции $U(x)$ строим отдельно на каждом промежутке $[x_j, x_{j+1}]$:

$$\widetilde{U}(x) = u_{j-2}\varpi_{j-2}(x) + u_{j-1}\varpi_{j-1}(x) + u_j\varpi_j(x) + u_{j+1}\varpi_{j+1}(x) + u_{j+2}\varpi_{j+2}(x), \quad (6)$$

где

$$\varpi_j(x) = \begin{cases} \dfrac{((x-x_{j+1})(x-x_{j+2})(x-x_{j+3}))}{((x_j-x_{j+1})(x_j-x_{j+2}))}, & x \in [x_{j+1}, x_{j+2}], \\ \dfrac{((x-x_{j-1})(x-x_{j+1})(x-x_{j+2}))}{((x_j-x_{j-1})(x_j-x_{j+1})(x_j-x_{j+2}))}, & x \in [x_j, x_{j+1}], \\ \dfrac{((x-x_{j-2})(x-x_{j-1})(x-x_{j+1}))}{((x_j-x_{j-2})(x_j-x_{j-1})(x_j-x_{j+1}))}, & x \in [x_{j-1}, x_j], \\ \dfrac{((x-x_{j-3})(x-x_{j-2})(x-x_{j-1}))}{((x_j-x_{j-3})(x_j-x_{j-2})(x_j-x_{j-1}))}, & x \in [x_{j-2}, x_{j-1}]. \end{cases} \quad (7)$$

График функции $\varpi_j(x)$ изображен на рис.6. Для апроксимации (6) используются четыре бизисные функции(см. рис.7).

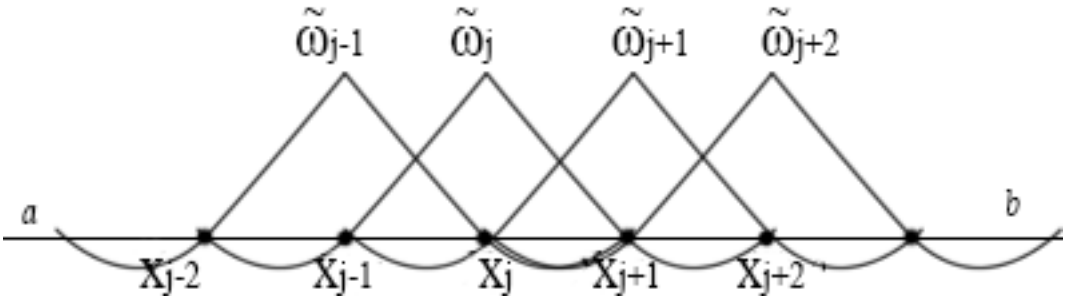

Рис.7

Из (7) следует, что

$$\varpi_j(x) = \frac{((x-x_{j-1})(x-xj+1)(x-x_{j+2}))}{((x_j-x_{j-1})(x_j-x_{j+1})(x_j-x_{j+2}))}, \quad x \in [x_j, x_{j+1}],$$
$$\varpi_{j-1}(x) = \frac{((x-x_j)(x-xj+1)(x-x_{j+2}))}{((x_j-x_{j-1})(x_{j-1}-x_{j+1})(x_{j-1}-x_{j+2}))}, \quad x \in [x_j, x_{j+1}], \qquad (8)$$
$$\varpi_{j+1}(x) = \frac{((x-x_{j-1})(x-x_j)(x-x_{j+2}))}{((x_{j+1}-x_{j-1})(x_{j+1}-x_j)(x_{j+1}-x_{j+2}))}, \quad x \in [x_j, x_{j+1}],$$
$$\varpi_{j+2}(x) = \frac{((x-x_{j-1})(x-x_j)(x-x_{j+1}))}{((x_{j+2}-x_{j-1})(x_{j+2}-x_j)(x_{j+2}-x_{j+1}))}, \quad x \in [x_j, x_{j+1}].$$

1.4 Построение изображений поверхностей с помощью тензорного произвдения

Для построения изображения поверхности строим прямоугольную сетку узлов на плоскости. Пусть c, d — вещественные числа. Вдоль оси Y на промежутке $[c, d]$ строим упорядоченную сетку узлов $\{y_k\}$, и проводим прямые параллельные оси X. На координатной оси X отмечаем точки x_j и через эти точки проводим прямые параллельные оси Y(см. рис. 5), точки пересечения линий называем узлами сетки и обозначаем их через (x_j,y_k).

Считаем, что поверхность $U(x, y)$ задаётся значениями в узлах сетки. Отмечаем на плоскости точки пересечения (x_j, y_k), $(x_j, y_{k+1}), (x_{j+1}, y_k)$, (x_{j+1}, y_{k+1}). Прямоугольник с этими вершинами называем элементарным и обозначаем его P_{jk} (рис.4). Приближение $\widetilde{U}(x, y)$ строим отдельно в каждом элемантарном прямоуголнике.

В каждом узле сетки (x_j, y_k) задано значение функции $U(x_j, y_k)$. Для построения поверхности используем мультипликативные базисные функции:

$$\Omega_{jk}(x, y) = \varpi_j(x)\varpi_k(y). \qquad (9)$$

Обозначим $U_{me} = U(x_m, y_e)$, тогда

$$\widetilde{U}(x,y) = \sum_{m=j-1,j,j+1,j+2} \quad \sum_{l=k-1,k,k+1,k+2} U_{me}\varpi_m(x)\varpi_l(y), \quad (x,y) \in P_{jk}. \quad (10)$$

1.5 Построение изображений в трехмерном пространстве

Пусть даны четыре точки V_1, V_2, V_3, V_4, расположенные в трехмерном пространстве. Точка V_i задается координатами (x_i, y_i, z_i)(рис.8).

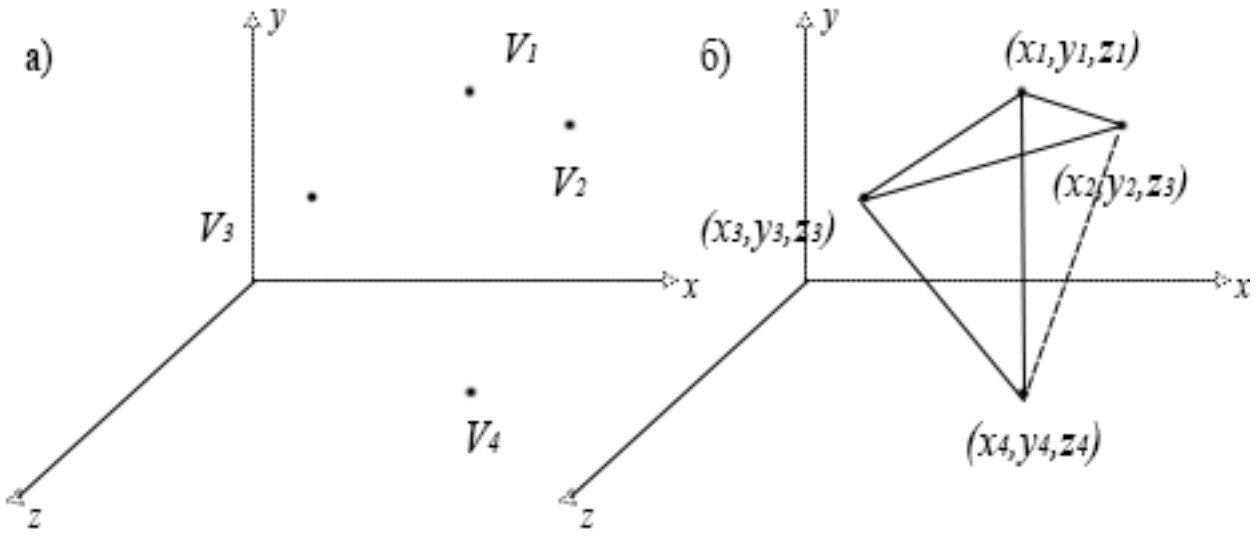

Рис.8

Пусть функции $x(t)$,$y(t)$,$z(t)$, таковы что $x, y, z \in \mathrm{C}^2(R^1)$.

Аппроксимацию функций $x(t)$,$y(t)$,$z(t)$ строим отдельно на каждом промежутке $[t_j, t_{j+1}]$.

$$\widetilde{x}(t) = x(t_j)\varpi_j(t) + x(t_{j+1})\varpi_{j+1}(t), \quad (11)$$

$$\widetilde{y}(t) = y(t_j)\varpi_j(t) + y(t_{j+1})\varpi_{j+1}(t),$$

$$\widetilde{z}(t) = z(t_j)\varpi_j(t) + z(t_{j+1})\varpi_{j+1}(t).$$

где базисные функции — $\varpi_j(x), \varpi_{j+1}(x)$, таковы что $supp\ \ \varpi_j = [x_{j-1}, x_{j+1}]$. Исходя из (см.11)

$$\varpi_j(t) = \frac{(t - t_j)}{(t_j - t_{j+1})}, t \in [t_j, t_{j+1}]. \tag{12}$$

Для апроксимации (11) используются две базисные функции $\varpi_j(t)$ и $\varpi_{j+1}(t)$.

Из (12) следует,что

$$\varpi_{j+1}(t) = \frac{(t - t_j)}{(t_{j+1} - t_j)}, t \in [t_j, t_{j+1}]. \tag{13}$$

На рис.8 б изображена пирамида построенная по формулам (11).

Глава 2. Обзор существующих решений

В данное время задача отображения и построение графического контента на мобильных устройствах является очень актуальной. С учетом развития информационных технологий, развивается и усовершенствуется подача графического контента для всех владельцев электронных устройств. Для того чтобы ускорить процесс передачи определнной информации в данное время наблюдается тенденция оптимизации всей информации под мобильные устройства. Так как с помощью мобильных устройств и приложений можно легко и быстро получать необходимую информацию, в отличии от персональных компьютеров. Ниже представлены мобильные приложения, которые предоставляют возможность построения линий и поверхностей.

2.1 Приложение Linear Fit

Мобильное приложение соданное разработчиком О. Г. Кальдероном[39], позволяет выполнять линейную аппроксимацию данных, предоставляемых в (x, y) координатах. Данные вводить в соответствующие области. Цифры должны быть написаны через запятую, без пробелов. Приложение вычисляет линию $y = m * x + b$ методом наименьших квадратов и отображает значение наклона "m" и коэффициента "b". Также в данном приложении предоставляется коэффициент корреляции. Из этого выходит что, в данном приложении реализуется лишь частный случай построение двухмерных поврехностей, к тому же построение графиков производится с помощью уже готовых наборов инструментов и библиотек реализуемые в Google Chart Tools.

2.2 Приложение Spline Interpolation

Данное приложение было разработанно в Университете прикладных наук Jade в Германии[40]. Приложение может проводить вычисления для построения линий и поверхностей. Результат работы програмы можно увидеть в декартовой системе координат, а также имеется возможность по-

лучить рассчитанные коэффициенты. Кроме того, присутствует возможность перемещать точки в системе координат и наблюдать, как меняется функция. Однако, в данном приложении имеются строгие ограничения по построениея функций и отсутствует вывод необходимых данных для дальнешего использования.

2.3 Curve Fit - Tools

Инструментарий данного приложения поможет найти наиболее подходящий вариант для построения кривой с помощью аппроксимации наименьших квадратов. Вы можете найти полиномиальную, экспоненциальную или линейную функцию для любой кривой. Также есть возможность сохранить данные для дальнейшей обработки[41]. В данном приложении отсутвует построение определенных геометрических фигур, так как оно лишь демонтсрирует граффик кривой на плоскоскости. Также осутствует динамическое изменение кривых и добавление нескольких объектов одновременно.

2.4 Приложение Math Professional

Графический калькулятор[42] позволяет вычислять основные функции научного калькулятора, имеется возомжность строить двухмерные и трехмерные функции, обеспечивает преобразование чисел в основных системах счисления, вычисляет производные. Реализует логарифмические и тригонометрические функции. Отсутствует добавление добавлнеие нескольких функций одновременно.Основной недостаток это наличие только платной версии.

2.5 Приложение Grapher Pro

Эффективное мобильное приложение для построения уравнений[43], способное создавать большинство функций (включая комплекснозначные), выполнять решение уравнений и вычислять различные выражения. Также имеется широкий набор определенных функций, в том числе гиперболические и тригонометрические функции, дифференциирование. В данном приложении как примеры, изображаются кривые Декарта, полярные, параметрические, неявные, трехмерные и другие кривые. Отсутсвует построение триангулирующей поверхности и самой триангуляции. Использовать данное приложение можно лишь при платной подписки.

2.6 Выводы

Рассмотренные выше программные продукты реализуют некоторые возможности для построения двухмерных поверхностей социальных взаимодействий. Часть из них специализируются на узком сегменте задач или функций по построению графических объектов или графиков. Другие программные продукты предоставляют обширный функционал, но в то же время не реализуют такие задачи как построение триангуляции или сохранение полученных значение, к тому же предоставляют платное использование. Для построение большинства функций наиболее подходящими решениями в данное время являются Math Professional[42] и Grapher Pro[43]. Но не стоит опираться только на эти продукты, так как обработка входящих данных в динамическом режиме производится очень медленно, либо отсутствует. Во-первых, отсутвует возможность построения триангулирующей поверхности или изменение её свойств. Во-вторых, в данных программных продуктах большинство функций строятся с помощью уже готовых библиотек и примитивов, которые создаются уже поставщиками программного обеспечения, следовательно возникает небольшая ограниченность в масштабируемости продукта.

Решение, предложенное в данной работе, это унифицирование и создание общего подхода по построению двухмерных поверхностей. Возможность задавать на вход различные точки, а на выходе получать различные фигуры как геометрически правильные так и нет. Также есть возможность динамического изменения поверхнойстей, путем добавления новых данных в алгоритм.

Глава 3. Программная реализация

3.1 Архитектура мобильного приложения

В ходе проведения анализа программных продуктов по построению двухмерных поверхностей был сделан выбор в пользу создания мобильного приложения на операционной системе android, доступного всем современным мобильным устройствам данной ОС. Данное решение позволит большинству пользователей без труда пользоваться данным приложением на своих устройствах и упростит поиск потенциальных решений для конкретных задач. Блок-схема рабооты приложения представлена на рисунке 9.

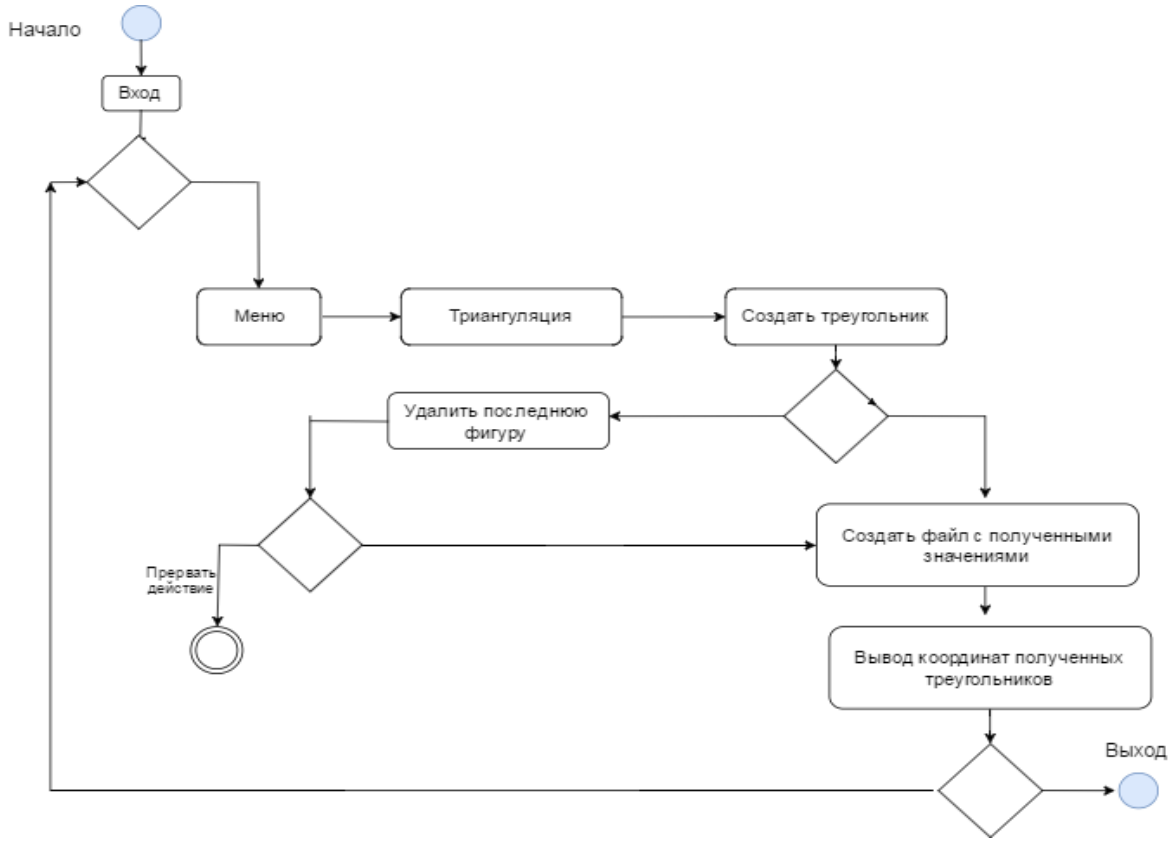

Рис.9

Так как было выбрано решение в пользу мобильного приложения, главным языком программирования стал Java, который является наиболее рас-

пространенным стандартом для создания и распространения мобильных приложений, игр т.д. Недостатком использования языка заключается в том, что яхык не полностью объектно-ориентирован. Проявляется это в отсутствии некоторых свойств, таких как индивидуальные переменные, множественное наследование и другие свойства. В свою очередь язык популярен в окружении проектов с открытым кодом и технически превышает многочисленные сторонние аналоги. Диаграмма классов представлена на рисунке 10.

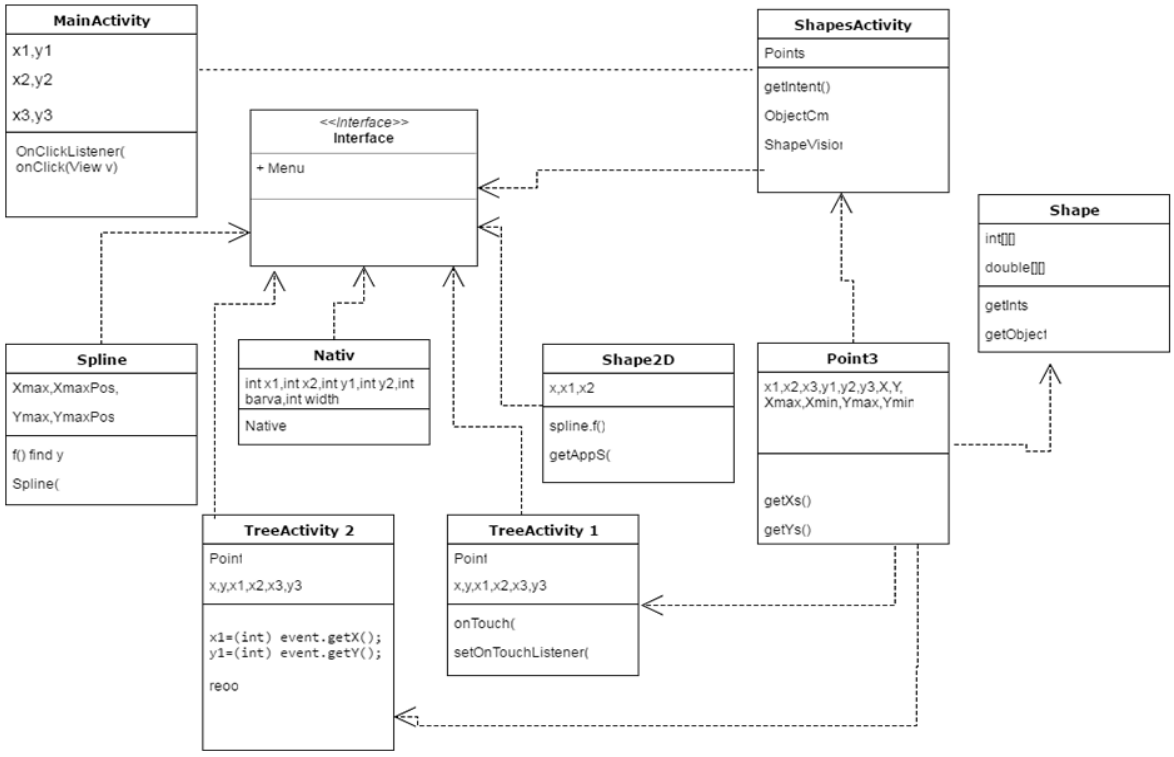

Рис.10

3.2 Программные средства, используемые для создания двухмерных поверхностей

В ходе разработки мобильного приложения использовались следующие технологии:

— язык Java;
— интегрированная среда разработки Android Studio;
— прикладной программный интерфейс Canvas;
— эмулятор Android Genymotion.

3.2.1 Язык Java

Технологические процессы языка Java имеют многочисленные отличительные черты, которые выделяют данный язык от прочих языков программирования.

Переносимость — создание программного представления на одной платформе и его последующий пуск в каждой иной платформе, в месте где устанавливается JVM.

Безопасность — Java гарантирует защиту, сдерживая влияние прогаммы исполняющей средой java, запрещая её доступ к другой части ОС компьютера. Надежность — в Java нет структур, приводящие к потенциальным ошибкам.К примеру, арифметика указателей, потеря точности при преобразованиях типов и т.д. Автоматическое обнуление оперативной памяти, так как Java выделяет ресурсы для уборки неприменяемых объектов на удаление. В корректно созданном проекте на java все ошибки должны и могут подвергаться обработке программой [7].

С помощью использования объектно-нацеленного подхода предоставляет использование готовых объектов в собственных програмках. Java предполагает собой базу почти абсолютно всех видов сетевых систем и считается общим эталоном для внедрения и распространения интегрированных и мобильных приложений, игр, и коллективного программного обеспечения.

3.2.2 Интегрированная среда разработки Android Studio

Android Studio — интегрированная и бесплатная среда разработки на базе IntelliJ IDEA [15], содержит одну из версий Android SDK и дополнительные средства графических интерфейсов, классов и шаблонов которые упрощающают разработку новых приложений, в добавок содержит такие средства, которые выполняют упаковку приложений и их работу[10].

Android-приложения состоят из компонентов, которые система может инициализировать и запустить при необходимости.

В целом в приложениях на Android емеется четыре типа компонентов:
— деятельность (Activity);
— служба (Service);
— приемник широковещательных намерений (Broadcast Receiver);
— контент-провайдер (Content Provider).

Деятельность — представляет собой видимую часть приложения, т.е ви-

зуальный пользовательский интерфейс. Все деятельности реализуются как подкласс базового класса Activity. Пример представлен в Листинге 1.

```
public class MainActivity extends Activity{
...
}
```

Листинг 1

Служба — элемент, не имеющий интерфейса пользователя и выполняется в низкоприоритетном порядке на протяжение неопределенного периода [16]. Например, элемент способен дать возможность пользователю и пользоваться другим приложением, доступна загрузка данных из сети, при этом не блокируется доступ с деятельностью.

Приемник вещательных целей — элемент, целью которого является получение событий внешних компонентов и их откликов. Способен привести в действие деятельность ответом на информацию, которую он приобретает, либо продемонстрировать уведомление, для пользователя [16]. К примеру, оповестить о невысоком заряде баттареи.

Контент-провайдер — распоряжается конкретным комплексом данных, применяемых приложением, с доступом для мобильных приложений. Контент-провайдеры используют систему разрешений для безопасного доступа т.е пользователь может создавать контент-провайдер и предоставить доступ к своим данным из других систем, затем использовать другие контент-провайдеры для обращений к их базам данных [16] .

3.2.3 Прикладной программный интерфейс Canvas

Главная концепция применения методов в Canvas заключается в том, то что они считаются семантически независимыми элементами. Подразумевается, то что данный класс считается белым холстом, внутри которого можно "изобразить"какой-либо компонент.В Canvas существуют способы и методы для изображения графического конента в виде примитивов. Для начала необходимо сформировать класс paint, предоставляющий обозначать как будут отображаться стиль, формат шрифта, цвет и т.д.

Рентабельность использования Canvas, будет лучше подходить когда требуется постоянная собственная прорисовка. Canvas содержит собственные методы и способы рисования, и при необходимости применяться.

К примеру, такие методы как:

— onDraw() - метод является Canvas объектом, используется чтобы нарисовать себя. Canvas Класс определяет методы для рисования текста, линий, растровых изображений, а также многих других графических примитивов. Этот метод позволяет создать свой собственный пользовательский интерфейс (UI).

— drawBitmap() - рисует растровое изображение на холсте. Можно изменять внешний вид целевой картинки, указывая итоговый размер или используя матрицу для преобразования;

— drawPoint() - рисует точку в заданном месте;

— drawLine(s)() - рисует линию (или последовательность линий) между двумя точками.

3.3 Расчётная и программная часть

Прикладная программа разрабатывалась как мобильное приложение ориентированное на ОС Android версии 4.4.4 KitKat.

Разработку приложение можно разделить на основные состовляющие:

— создание класса MainActivity;

— создание класса ShapesActivity;

— создание класса Point3;

— создание интерфейса Shape;

— создание класса Shape2D;

— создание класса Spline;

— создание класса Nativ;

— создание класса TreeActivity1;

— создание класса TreeActivity2;

3.3.1 Класс MainActivity

Класс MainActivity — это класс-навигатор активностей с функцией перехода по кнопке. Функции содержат в себе координаты будущих фигур, которые отправляются по ключевому слову(тег) в другую активность для рисования линий и построения треугольников. Фрагмент кода представлен в Листинге 2.

```
findViewById(R.id.shape2d).setOnClickListener
(new OnClickListener() {
@Override
public void onClick(View v) {
i = new Intent(getApplicationContext(), Shape2d.class);
```

```
int[] value = new int[]{0, 0, 100, 100, 120, 10,
200, 200, 250, 0, 300, 100, 350, 100};
int[] PointsO = new int[]{200, 300, 200, 200};
i.putExtra("PointsOpor", PointsO);
i.putExtra("KFWH", 1);
i.putExtra("offsetY", 300);
i.putExtra("offsetX", 200);
i.putExtra("Points", value);
i.putExtra("w", v.getRootView().getRootView().getWidth());
i.putExtra("Heig", findViewById(R.id.jostic1).getY());
startActivity(i);}});
```

Листинг 2

В данном листинге мы обращаемся к встроенному методу findViewById который находит view(т.е вид), который был идентифицирован с помощью атрибута ID из XML. Далее переопределяем метод интерфейса OnClickListener(). Этот метод обрабатывает нажатия.В нем добавляется массив int[] value типа integer в котором задаются точки будущих фигур, по парам x и y массив опорных точек int[] PointsO. Задается коэффициент KFWH, который увеличиват или уменьшает значения точек. Далее в задается ширина родителя кнопки, т.е ширина экрана. Также добавляется высота джойстика для управления фигурами и далее стартуется активность.

3.3.2 Класс ShapesActivity

Класс ShapesActivity — этот класс осуществляет принятие массива точек различных фигур и затем визуализирует эти фигуры как двумерные объекты. Листинг 3.

```
class ShapesActivity extends Activity
//принимаем массив
int[]as = getIntent().getIntArrayExtra("Points");
//обрабатываем
int[][]   ObjectCm=new int[as.length/2][2];
for(int i=0;i< ObjectCm.length;i++){
ObjectCm[i][1]= as[i*2];
ObjectCm[i][0]= as[i*2+1];}
 for(int i=0;i< ObjectCm.length/3;i++){
 Xmax-ObjectCmSpline.Xmin),
```

```
(ObjectCmSpline.Ymax-ObjectCmSpline.Ymin));
Point p1=new Point(ObjectCm[i*3][1],
ObjectCm[i*3][0]);
Point p2=new Point(ObjectCm[i*3+1][1],
ObjectCm[i*3+1][0]);
Point p3=new Point(ObjectCm[i*3+2][1],
ObjectCm[i*3+2][0]);
int w=Math.max(Math.abs(p2.x-p1.x),
Math.max(Math.abs(p2.x-p3.x),
Math.abs(p3.x-p1.x)));
int h=Math.max(Math.abs(p2.y-p1.y),
Math.max(Math.abs(p2.y-p3.y),
Math.abs(p3.y-p1.y)));
final ShapeVision sv=new ShapeVision
(getApplicationContext(),
p1,p2,p3,w,h,colors[i],i);
dragAndMove(sv);
root_view.addView(sv,w,h);
Button child=new Button(getApplicationContext());
child.setTextColor(0xff000000);
child.setBackgroundColor(colors[i]);
child.setText(String.valueOf(i+1));
TableRow.LayoutParams params=new TableRow.LayoutParams(-2,-2);
params.setMargins(10, 0, 10, 20);
child.setLayoutParams(params);
cv.addView(child);
child.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
sv.setRotation(sv.getRotation() + 90);
}});}
```

Листинг 3

В данном листинге используется класс ShapeVision. Он отвечает за конкрентую фигуру, то есть ее отображение в виде View и используется как массив таких фигур. В зависимости от количества треуголников используются экхемляры этого класса.

3.3.3 Класс Point3

Класс Point3 — этот класс отвечает за три точки образующий треугольник, максимумы к минимуму, начальное значение и текущее смещение. Листинг 4.

```
public class Point3 {
public int x1,x2,x3,y1,y2,y3,X,Y,
Xmax,Xmin,Ymax,Ymin,XNominal,YNominal,RAD;
double xnk1,xnk2,xnk3,ynk1,ynk2,ynk3;
public int[] getXs(){
int[] out=new int[3];
Xmax=Math.max(x1+X, Math.max(x3+X, x2+X));
Xmin=Math.min(x1+X, Math.min(x3+X, x2+X));
out[0]=x1+X;
out[1]=x2+X;
out[2]=x3+X;
return out;}
public int[] getYs(){
int[] out=new int[3];
Ymax=Math.max(y1+Y, Math.max(y3+Y, y2+Y));
Ymin=Math.min(y1+Y, Math.min(y3+Y, y2+Y));
out[0]=y1+Y;
out[1]=y2+Y;
out[2]=y3+Y;
return out;}
```

Листинг 4

В этом классе импользуется мотод getXs() который возвращает массив из x-координат, метод getYs() возвращает массив из y-координат по которым образуется трекголник.

3.3.4 Класс Shape

Класс Shape — этот интерфейс служит для получения и установления параметров фигур. Листинг 5.

```
public interface Shape {
Object[][] getObject();
void setDoubles(double[][] in);
```

```
void setInts(int[][] in);
int[][] getInts();
double[][] getDoubles();
void setObject(Object[][] object);}
```

Листинг 5

3.3.5 Класс Shape2D

Класс Shap2D — этот класс представляет собой активность полумесяца. Он строится с помощью сплайнов в виде дуги. Они имеют координаты x и y, есть функция в классе spline() которая вычисляется от одной точки до второй на основании 3 точек- вершин. По формулам, когда $x < x1 < x2$, находим все y при росте x. Листинг 6.

```
// заполнение точками сплайна
double[][] getAppS(double [][]xF){
int i=0;
double[] xww=new double[3];
for(int j=0;j<3;j++)xww[j]=xF[j][0];
double[] yw=new double[ 3];
for(int j=0;j<3;j++)yw[j]=xF[j][1];
Spline.SplineTuple[] spl = Spline.BuildSpline1(xww,yw);
//это  изначальная величина предварительного массива(400 точек)
int count= Math.min(w,H)/2;//(int) Math.max(
Math.abs(xF[i][1]-xF[i+1][1] ),
Math.abs(xF[i][1]-xF[i+1][1]) );
double kfy=2*Math.abs(xF[i+1][0]-xF[i][0])/(double)count;
double[][] out=new double[count][2];
for(int j=0;j<count/2+1;j++){
//это найдена определенная точка - y по x= i
double ff = Spline.f(xF[i][0]+j*kfy,spl);
//Spline.f -  статическая вспомогательная функция
out[j][1]=xF[i][0]+j*kfy;
out[j][0]=ff; }
i++;
kfy=2*(xF[i+1][0]-xF[i][0])/(double)count;
//берется 2-половина массива и заполняется она
for(int j=count/2;j<count;j++){
```

```
double ff = Spline.f(xF[i][0]+(count-2-j)*kfy,spl);
out[j][1]=xF[i][0] + (count-2-j)*kfy;
out[j][0]=ff;
}return out;}
```

Листинг 6

В данном листинге представлнео заполнение сплайна точками, вспомогательная функция spline.f() используется для поиска y-ков из массивов.

3.3.6 Класс Nativ

Класс Nativ — этот класс переводит все данные о фигурах, в нативную форму через java native interface, которая считает методы быстрее. Используется для получения быстрого визуального отображения на экране смартфона. Листинг 7.

```
class Nativ {
native static int[] TrivialAlg
(int[] out,int x1,int x2,int y1,int y2,int barva,int width);
native static void Sizego
(int[] in,int rad,int h,int starty ,int color);
native static int[] DrawLineApprox
(int[] out, int x1,int x2,int y1,int y2, int barva ,
int w);
static {
System.loadLibrary("b431");}}
```

Листинг 7

В данном листинге представлены методы:TrivialAlg рисует линию по координатам с зданной шириной; Sizego зарисовывает фигуру заданным цветом; DrawLineApprox выполняет аппроксимацию по заданным координатам.

3.3.7 Класс Spline

Класс Spline — этот класс осуществляет нахождение и вычисление сплайна. Главным образом хранит координаты прямых в виде 2-го массива. Листинг 8.

```
public class Spline  implements Shape{
int Xmax,XmaxPos,Xmin=Integer.
MAX_VALUE,XminPos,Ymax,YmaxPos,
Ymin=Integer.MAX_VALUE,YminPos;
public Spline(int[][] in) {
super();
setInts(in);}
public Spline(double[][] in) {
super();
setDoubles(in);}
public Spline() {
super(); }
public Spline(int[]in) {
super();
setDoubles(createSpline(in));
}} }}
public static double f(double x,SplineTuple[] splines ) {
SplineTuple s;
int n= splines.length;
if (x <= splines[0].x)
s = splines[1];
else if (x >= splines[n - 1].x)
s = splines[n - 1];
else {
int i = 0, j = n - 1;
while (i + 1 < j) {
int k = i + (j - i) / 2;
if (x <= splines[k].x)
j = k;
else
i = k;}
s = splines[j];}
 double dx = (x - s.x);
return s.a + (s.b + (s.c / 2.0 +
s.d * dx / 6.0) * dx) * dx;}
```

Листинг 8

В этом листинге описанна раннее используемая функция $f()$, в которой осуществляется поиск y-ков.

3.3.8 Класс TreeActivity 1

Класс TreeActivity 1 и TreeActivity 2 являются активностями(видимостями) для наглядной демонстрации работы триангуляции. Также они содержат функцию вывода координат полученных в ходе работы точек в текстовый файл. Для отображения используется класс Shape extends View.Листинг 9.

```
class Shape extends View{
//нарастающие точки,при касании экрана
//определяющего x  и y
ArrayList<Point> al;
ArrayList<Point3> add;//прирастающие треугольники
Bitmap bm;//это отражение рисунка который получается через canvas
int[] pixels;
// закрашивание невидимых точек;
public Shape(Context context) {
super(context);
setBackgroundColor(0xffffffff);
pixels=new int[w*h];
bm = Bitmap.createBitmap(w,  h, Bitmap.Config.ARGB_8888);
al=new ArrayList<Point>();
setOnTouchListener(new OnTouchListener()
```

Листинг 9

При нажатии на экран метод onTouch создается новая точка координаты которой записываются в массив. Листинг 10.

```
public boolean onTouch(View v, MotionEvent event) {
int x1;int y1;
switch(event.getAction()){
case MotionEvent.ACTION_DOWN:break;
case MotionEvent.ACTION_UP:
x1=(int) event.getX();
y1=(int) event.getY();
al.add(new Point( x1,y1));
remath();   } break;
default :return false;}
return true;  }
```

Листинг 10

Если количество точек равно 2 , то формируется и отображается прямая. Листинг 11.

```
void remath() {
if(al.size()==2){
Spline spline =new Spline
(Spline.DrawLineApprox
(al.get(0).x, al.get(1).x, al.get(0).y, al.get(1).y));
spline.getPaint( pixels, w, 0xff000000);
bm.setPixels(pixels, 0,w, 0, 0,w, h);
postInvalidate(); }
```

Листинг 11

Если количество точек равно 3, то формируется и отображается один треугольник. Каждая дополнительная точка будет формировать следующий треугольник, 2 вершины которого будут составлять ближайшее найденное ребро из чила уже построенных треугольников. За это отвечает функция perfect. Листинг 12.

```
if(al.size()==3){
add=new ArrayList<>();
Point3 ss = new Point3();
ss.setPointsNomin(al.get(0).x, al.get(1).x,
al.get(2).x,al.get(0).y, al.get(1).y, al.get(2).y);
add.add(ss);
endPaint(); }
```

Листинг 12

Класс TreeActivity 2 отличается от класса TreeActivity 1 только ограничениями в построении треугольников, вне определенного периметра и пересечением внутреннего четырехугольника.

3.4 Полученные результаты

Вся разработка мобильного приложения проводилась на машине с процессором Intel-Core-$i7$, графической видеокартой NVIDIA GeForce-740M и на операционной системе Windows 7. Разработка осуществлялась в IDE Android Studio с эмулятором Android Genymotion и образом смартфона Samsung Galaxy-$S4$, версии 4.4.4 Kit-Kat.

Для демонстрации работы самого мобильного приложения на эмуляторе, необходимо скомпелировать проект. Для этого необходимо в IDE Android Studio нажать на вкладку "Run app"в верхней панели управления. Результат работы приложения продемонстрирован на рис.11.

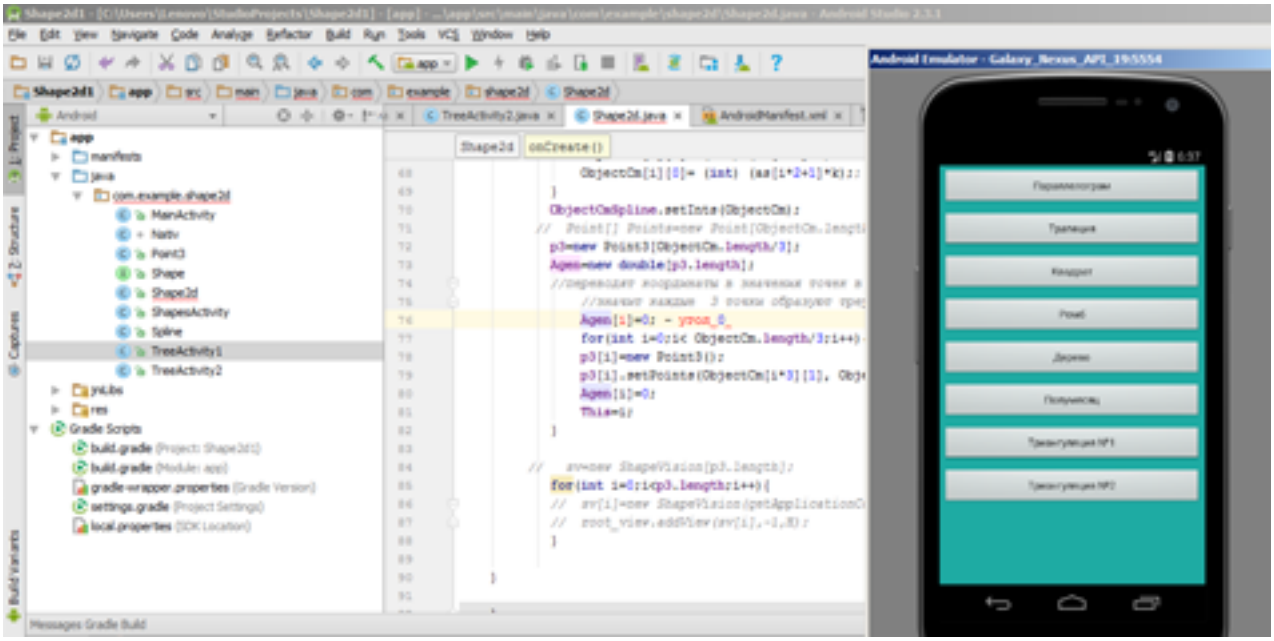

Рис.11

Для того чтобы приложение загрузить на смартфон, необходимо для начала создать файл с разрешением .apk. Для этого в среде разработки нажимаем на вкладку "Build"и выбираем команду Build APK (Рис. 12).

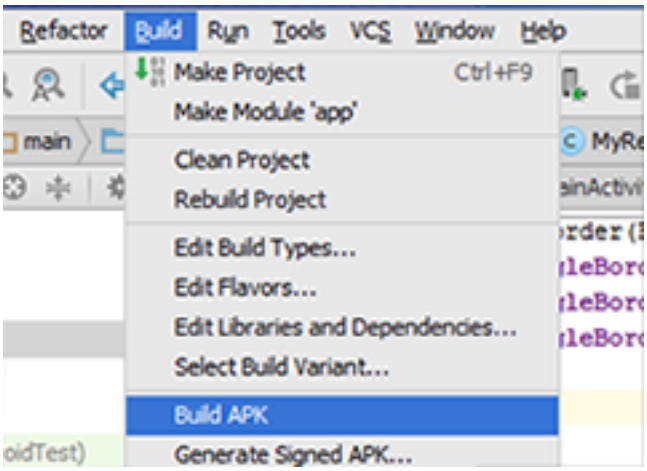

Рис.12

Приложение скомпанованно и готово к использованию. В право верхнем углу появиться вкладка "APK generated successfully". При нажатии на неё, откроется папка местонахождения apk файла (рис. 13).

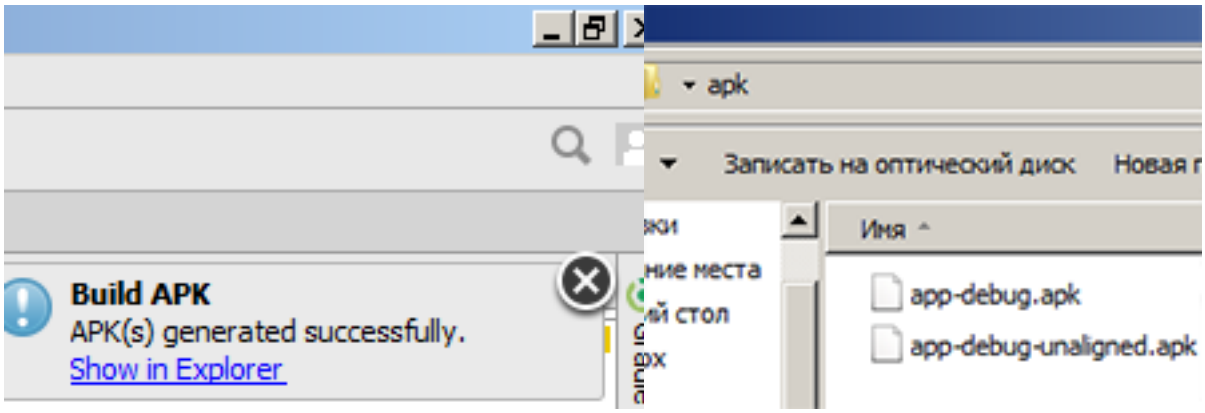

Рис.13

Для использования приложения необходимо загрузить файл app-debug.apk на смартфон.

3.5 Руководство пользователя

Пользователь может войти в мобильное приложение, нажав на кнопку войти, ввод пароля не потребуется. Далее откроется меню приложения, в котором будут представлены восемь разделов (Рис. 14). Каждый раздел имеет своё практическое применение. После нажатия на конкретный раздел, пользователь окажется в окне раздела, который будет иметь своё практическое пременение (Рис. 15).

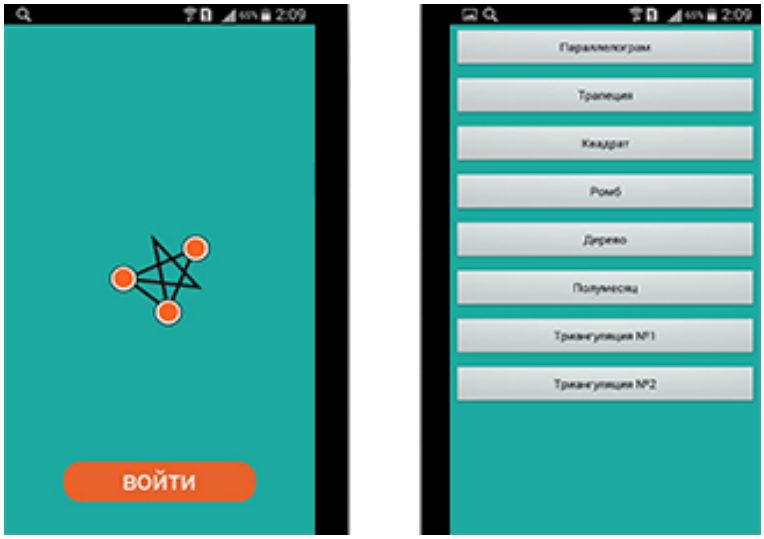

Рис.14

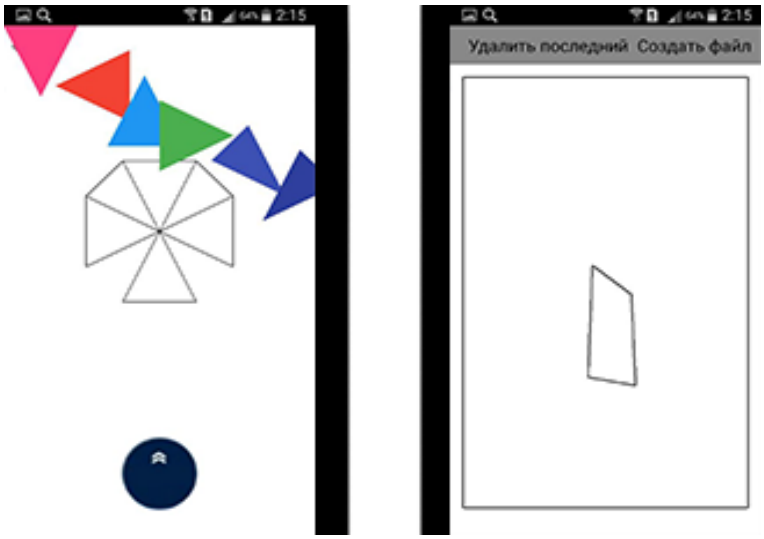

Рис.15

К примеру, если войти в раздел Дерево, нам будет предложено собрать пазл в виде дерева. На экране будут хаотически расположены треугольники, которые были построены с помощью сплайновой аппроксимации и использованием кусочно-линейных сплайнов (Рис. 16).

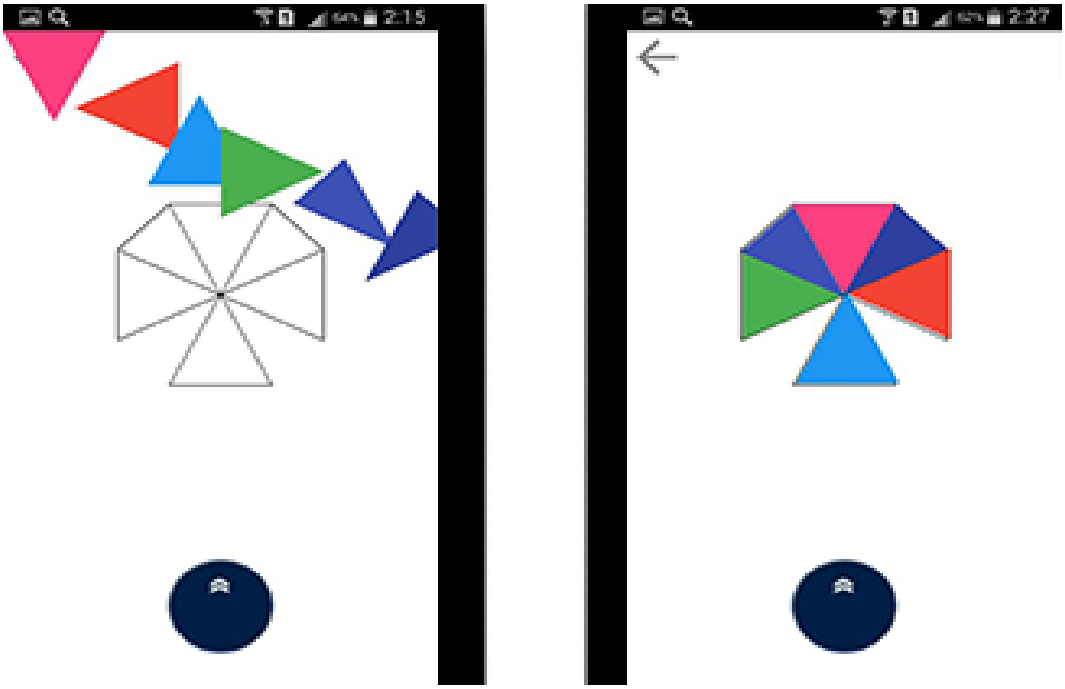

Рис.16

В разделе триангуляциция, нам предоставляется пространство внутри которого ограничивающая выпуклая фигура. На данном экране также существует кнопка удалить последнее действие и сохранить файл. Для построения триангуляции пользователю необходимо отмечать на экране координаты, программа будет сама вычислять ближайшую точку для соединения с последней проставленной точкой. Если пользователь в дальнейшем будет использовать координаты установленных точек, то в мобильном приложении предосталена функция сохранения координат в любом желаемом формате, в данном примере реализуется сохранение в txt файл. Файл point.txt создается в разделе storage в корневой папке проекта shape2d (Рис. 17).

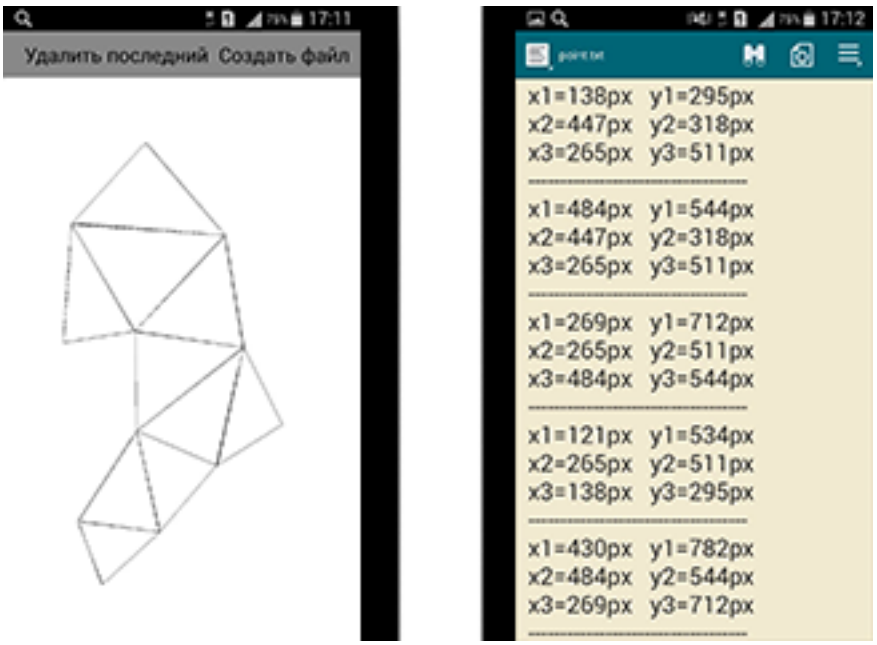

Рис.17

Имея триангуляцию на плоскости и занчения функции в узлах сетки, можно построить изображение поверхности[38]. В дополенение, в разделе триангуляции реализовано более сложное экспериментальное проектирование с добавлением значения функции $Z[i]$ (Рис 18.).

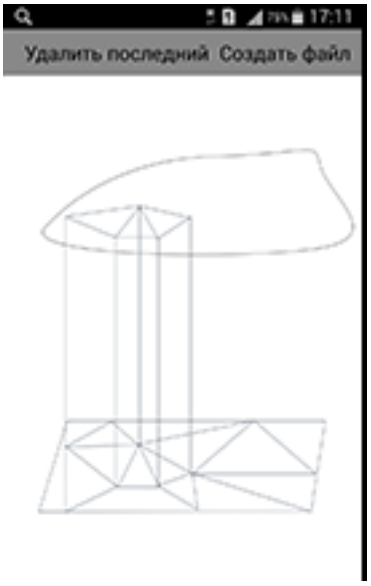

Рис.18

Используя триангуляцию на плоскости и задавая значения функции в узлах сетки, получаем триангуляцию поверхности (Рис. 19).

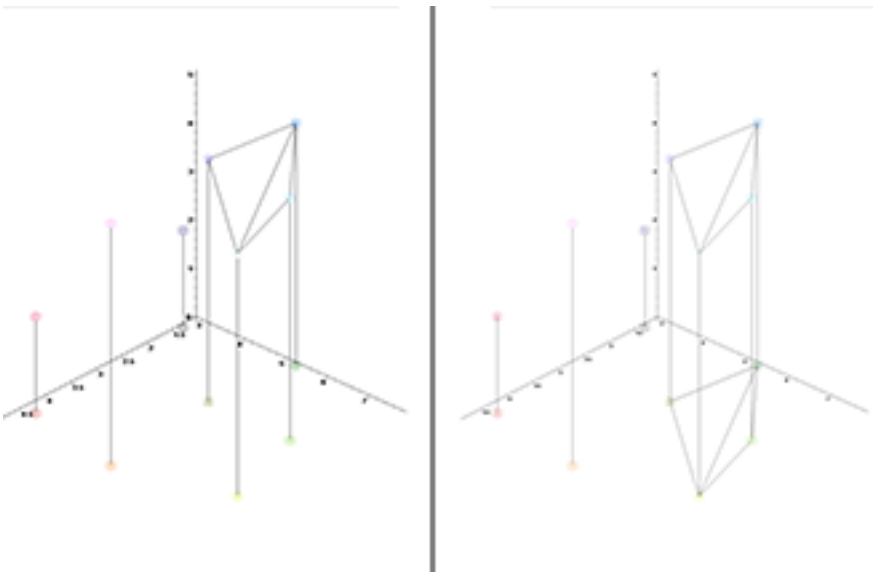

Рис.19

Заключение

В ходе выполнения данной магистерской диссертации были выполнены следующие задачи:

- Исследованны возможности по применению локальных сплайнов для разработки мобильного приложения на платформе Android;
- Разработано интерактивное мобильное приложение для платформы android, позволяющее пользователю интерактивно строить триангуляцию на плоскости, задавая узлы триангуляции последовательно касаясь экрана и имея возможность изменить расположение очередного вводимого узла;
- Разработано интерактивное мобильное приложение (для android) для перемещения плоских треугольников по экрану;
- Проведен анализ работоспособности приложения.

В дальнейшем планируется оптимизировать алгоритм построения триангуляции, а также провести экперементы с использованием друхих методов и способов разработки кросплатформенного приложения.

Список использованной литературы

[1] Бурова И.Г., Демьянович Ю.К. «Минимальные сплайны и их приложения». ISBN 978-5-288-04978-1, 2010, с. 364.

[2] Бурова И.Г. «Аппроксимация вещественными и комплексными минимальными сплайнами». ISBN 978-5-288-05466-2, 2013, с. 144.

[3] Демьянович Ю.К. «Локальная аппроксимация на многообразии и минимальные сплайны». ISBN 5-288-00481-1, 1994, с. 356.

[4] Завьялов Ю.С., Квасов Б.И., Мирошничен «Методы сплайн-функций». ISBN 978-5-86134-180-6, 1980, с. 352.

[5] Стечкин С.Б., Субботин Ю.Н. «Методы сплайн-функций». 1976, с. 248.

[6] Львовский С.М. «Набор и верстка в системе LaTeX ». ISBN 5-94057-091-7, 2006, с. 448.

[7] Герберт Шилдт «Java: The Complete Reference, Ninth Edition ». ISBN 978-5-8459-1918-2, 2015, с. 1376.

[8] Дейтел П., Дейтел Х. «Android for Programmers: An App-Driven Approach ». ISBN 978-5-496-01517-2, 2015, с. 384.

[9] Вязовик Н.А «Программирование на Java. Курс лекций ». ISBN 5-9556-0006-X, 2016, с. 592.

[10] Дон Гриффитс, Дэвид Гриффит «Head First. Программирование для Android ». ISBN 978-5-496-02171-5, 2016, с. 704.

[11] Timothy M. Wright «Fundamental 2D Game Programming with Java ». ISBN 978-1305076532, 2014, с. 656.

[12] Брюс Эккель «Thinking in Java (4th Edition) ». ISBN 978-5-496-01127-3, 2015, с. 1168.

[13] Баяковский Ю.М., Игнатенко А.В., Фролов А.И. «Графическая библиотека OpenGL. Учебно-методическое пособие. ». ISBN 5-89407-153-4, 2003, с. 132.

[14] Jim X.Chen, Edward J.Wegman «Foundations of 3D Graphics Programming: Using JOGL and Java3D ». ISBN 1-84628-185-7, 2006, с. 305.

[15] Давыдов С.В., Ефимов А.А. «IntelliJ IDEA. Профессиональное программирование на Java ». ISBN 5-94157-607-2, 2005, с. 800.

[16] Голощапов А. Л. «Google Android: программирование для мобильных устройств ». ISBN 978-5-9775-0562-8 , 2011, с. 448.

[17] Хашими С., Коматинени С., Маклин Д «Разработка приложений для Android ». ISBN 978-5-459-00530-1 , 2011, с. 736.

[18] Dave Shreiner, Graham Sellers «OpenGL Programming Guide ». ISBN 978-0-321-77303-6 , 2013, с. 984.

[19] Belén Cruz Zapata «Android Studio Application Development ». ISBN 978-1-78328-527-3 , 2013, с. 110.

[20] Васильев А. Н. «Java. Объектно-ориентированное программирование: Учебное пособие. ». ISBN 978-5-49807-948-6 , 2011, с. 400.

[21] Майер. Р «Android 2. Программирование для планшетных компьютеров и смартфонов ». ISBN 978-5-699-50323-0 , 2011, с. 671.

[22] Медникс З., Дорнин Л. «Программирование под Android. 2-е изд ». ISBN 978-5-496-00526-5 , 2013, с. 671.

[23] Фелкер Д. «Android: разработка приложений для чайников ». ISBN 978-5-8459-1748-5 , 2012, с. 336.

[24] Коматинени С. «Android 4 для профессионалов. Создание приложений для планшетных компь- ютеров и смартфонов. ». ISBN 978-5-8459-1801-7 , 2012, с. 880.

[25] Ласло М. «Вычислительная геометрия и компьютерная графика на C++ ». ISBN 5-7989-0008-8 , 1997, с. 304.

[26] Никулин Е.А. «Компьютерная геометрия и алгоритмы машинной графики ». ISBN 5-94157-264-6 , 2003, с. 560.

[27] Рождерс Д. «Матемаматические основы машинной графики ». ISBN 5-03-002143-4 , 2001, с. 604.

[28] Альберг Дж., Нильсон Э. «Теория сплайнов и её приложения. ». 1972, с. 319.

[29] Захаров А.А. «Математические модели геометрических объектов», Курс-лекций МГТУ им. Н. Э. Баумана «

[30] Федорова О. П. «Метод построения сплайна, сохраняющего интеграл функции двух переменных по области ее задания , Научный альманах ». 2016. – №. 1-3. – С. 31-35.

[31] Просочкин А. С. «Исследование спектра двумерных фундаментальных сплайнов , Филиал «Восход» МАИ ». 2016. – №. 1-3. – С. 31-35.

[32] Шкловец А. В., Аксак Н. Г. «Проецирование большого количества многомерных данных на двумерную кусочно-гладкую самоорганизующуюся карту Кохонена ». 2012. – №. 3 (1). – С. 81-84.

[33] Выпускная работа. Кемеровский государственный университет. «Программная реализация алгоритма Bubble Mesh для численного моделирования методом конечных элементов задач движения жидкости. ». 2015.

[34] Коннор Дж. «Программная реализация алгоритма Bubble Mesh для численного моделирования методом конечных элементов задач движения жидкости ». 1979, с. 264.

[35] Делоне Б.Н. «О пустоте сферы ». 1934, с. 793–800.

[36] Скворцов А.В. «Триангуляция Делоне и её применение. ». 2002, с. 128.

[37] Препарата Ф., Шеймос М. «Вычислительная геометрия: Введение ». 1989, с. 478.

[38] Голованов Н.Н. «Геометрическое моделирование : учебник для учреждений высш. проф. образования ». 2011, с. 272.

[39] https://play.google.com/store/apps/details?id=appinventor.ai_oscar_gomezcalderon.LinearFit_ShaDB

[40] https://www.jade-hs.de/

[41] https://play.google.com/store/apps/details?id=com.bragitoff.curvefit_leastsquares

[42] https://play.google.com/store/apps/details?id=it.smh17.math.professional

[43] https://play.google.com/store/apps/details?id=be.grapher.pro

Приложение. Тексты программ

Класс MainActivity

```
package com.example.shape2d;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;

public class MainActivity extends Activity {
Intent i;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.start);
 findViewById(R.id.start).setOnClickListener

 (new OnClickListener() {
@Override
public void onClick(View v) {
// TODO Auto-generated method stub
onstart();
}});}
void   onstart(){
setContentView(R.layout.activity_main);
findViewById(R.id.point_to_point).
setOnClickListener

(new OnClickListener() {
@Override
public void onClick(View v) {
// TODO Auto-generated method stub
i=new Intent(getApplicationContext(), TreeActivity1.class);
i.putExtra("w", v.getRootView().getRootView().getWidth());
i.putExtra("Heig",findViewById(R.id.jostic1).getY()+
```

```
+findViewById(R.id.jostic1).getHeight());
startActivity(i);
}
});
findViewById(R.id.point_to_point22).setOnClickListener
(new OnClickListener() {
@Override
public void onClick(View v) {
// TODO Auto-generated method stub
i=new Intent(getApplicationContext(), TreeActivity2.class);
i.putExtra("w", v.getRootView().getRootView().getWidth());
i.putExtra("Heig",findViewById(R.id.jostic1).getY()+

+findViewById(R.id.jostic1).getHeight());
startActivity(i);
}});
findViewById(R.id.shape2d).setOnClickListener


(new OnClickListener() {
@Override
public void onClick(View v) {
// TODO Auto-generated method stub
 i=new Intent(getApplicationContext(), Shape2d.class);
int[] value=new int[]{0,0, 100,100,  120, 10,
200, 200 , 250,0, 300,100 , 350,100  };
int[] PointsO=new int[]{   200,300 , 200,200  };
i.putExtra("PointsOpor", PointsO);
i.putExtra("KFWH", 1);
i.putExtra("offsetY", 300);
i.putExtra("offsetX", 200);
i.putExtra("Points", value);
i.putExtra("w", v.getRootView().getRootView().getWidth());
i.putExtra("Heig",findViewById(R.id.jostic1).getY());
startActivity(i);
}});
\thispagestyle{plain}
 findViewById(R.id.rhombus).setOnClickListener
 \thispagestyle{plain}
 (new OnClickListener() {
```

```
@Override

\thispagestyle{plain}
public void onClick(View v) {
// TODO Auto-generated method stub
i=new Intent(getApplicationContext(), ShapesActivity.class);
int[] value=new int[]{ 0, 100, 200,100, 100, 0,
0, 100 , 100,200 ,  200,100   };
i.putExtra("KFWH", 1.7);
i.putExtra("Points", value);
i.putExtra("w", v.getRootView().getWidth());
i.putExtra("Heig", findViewById(R.id.jostic1).getY());
startActivity(i);
}});
findViewById(R.id.trapezium).setOnClickListener

(new OnClickListener() {
@Override
public void onClick(View v) {
// TODO Auto-generated method stub
i=new Intent(getApplicationContext(), ShapesActivity.class);
int[] value=new int[]{100, 0, 100, 100, 0,100,
     100, 0 , 200, 0 ,  100, 100 ,
     200,0, 200,100, 100,100,
     200,0, 300,100, 200,100};
i.putExtra("KFWH", 1.7);
i.putExtra("Points", value);
i.putExtra("w", v.getRootView().getWidth());
i.putExtra("Heig", findViewById(R.id.jostic1).getY());
startActivity(i);
}
});
findViewById(R.id.square).setOnClickListener
\thispagestyle{plain}
(new OnClickListener() {
@Override

public void onClick(View v) {
// TODO Auto-generated method stub
i=new Intent(getApplicationContext(), ShapesActivity.class);
```

```
int[] value=new int[]{0,0, 0,200, 100,100,
     200, 200 , 200, 0 ,  100, 100,
     100,100, 0,0, 200,0,
     200,200, 0,200, 100,100};
i.putExtra("KFWH", 1.7);
i.putExtra("Points", value);
i.putExtra("w", v.getRootView().getWidth());
i.putExtra("Heig", findViewById(R.id.jostic1).getY());
startActivity(i);
}});
findViewById(R.id.parallelogram).setOnClickListener
\thispagestyle{plain}
(new OnClickListener() {
@Override
public void onClick(View v) {
i=new Intent(getApplicationContext(), ShapesActivity.class);
int[] value=new int[]{   100, 100, 0, 100, 100,0,
     100,100 ,  200, 0 , 100, 0,
      200, 0, 200,100,100,100 ,
     200,100, 300, 0, 200, 0};
i.putExtra("KFWH", 1.7);
i.putExtra("Points", value);
i.putExtra("w", v.getRootView().getWidth());
i.putExtra("Heig", findViewById(R.id.jostic1).getY());
startActivity(i);
}
});
findViewById(R.id.r).setOnClickListener

(new OnClickListener() {
@Override
public void onClick(View v) {
i=new Intent(getApplicationContext(), ShapesActivity.class);
int[] value=new int[]{   50, 0, 150, 0, 100,100
     ,200, 50 ,  200,150,  100, 100
     , 150, 200, 50,200,100,100
     , 0,150, 0, 50, 100, 100,
      0,50 ,50, 0 ,100,100,
      150,0, 200, 50 ,100, 100};
i.putExtra("KFWH", 1.7);
```

```
i.putExtra("Points", value);
i.putExtra("w", v.getRootView().getWidth());
i.putExtra("Heig", findViewById(R.id.jostic1).getY());
startActivity(i);}
});}}
```

Класс ShapesActivity

```
package com.example.shape2d;

import com.example.shape2d.R;

import android.app.Activity;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Point;
import android.os.Bundle;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnTouchListener;
import android.widget.Button;
import android.widget.FrameLayout;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.TableRow;

public class ShapesActivity extends Activity {
int[] colors = new int[]{0xffFF4081, 0xffF44336,
0xff2196F3, 0xff4CAF50, 0xff3F51B5, 0xff303F9F};
Spline ObjectCmSpline;
RelativeLayout root_view;
double[] Agen;
int H;
int This;
double k;
Point3[] p3;
ShapeVision[] sv;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
int[] as = getIntent().getIntArrayExtra("Points");
int w = getIntent().getIntExtra("w", 0);
H = (int) getIntent().getFloatExtra("Heig", 0);
k = getIntent().getDoubleExtra("KFWH", 1);
setContentView(R.layout.activity_rhombus);
root_view = (RelativeLayout) findViewById(R.id.root_view);
root_view.addView(new backGrShape(this), -1, H);

if (as != null) {
ObjectCmSpline = new Spline();
int[][] ObjectCm = new int[as.length / 2][2];
for (int i = 0; i < ObjectCm.length; i++) {
ObjectCm[i][1] = (int) (as[i * 2] * k);
ObjectCm[i][0] = (int) (as[i * 2 + 1] * k);
;
}
ObjectCmSpline.setInts(ObjectCm);
p3 = new Point3[ObjectCm.length / 3];
Agen = new double[p3.length];
\thispagestyle{plain}
for (int i = 0; i < ObjectCm.length / 3; i++) {
p3[i] = new Point3();
p3[i].setPoints(ObjectCm[i * 3][1], ObjectCm[i * 3 + 1][1],

ObjectCm[i * 3 + 2][1], ObjectCm[i * 3][0], ObjectCm[i * 3 +

1][0], ObjectCm[i * 3 + 2][0]);
Agen[i] = 0;
This = i;
}

sv = new ShapeVision[p3.length];
for (int i = 0; i < p3.length; i++) {
sv[i] = new ShapeVision(getApplicationContext(), w, colors[i], i);
root_view.addView(sv[i], -1, H);
}
```

```
}
findViewById(R.id.jostic).setOnTouchListener(new OnTouchListener() {
@Override
public boolean onTouch(View v, MotionEvent event) {
// TODO Auto-generated method stub
if (event.getAction() == MotionEvent.ACTION_MOVE) {

Agen[This] += (event.getX() - (v.getWidth() / 2)) / 3.14;
v.setRotation((float) Agen[This]);
sv[This].setPivotX(p3[This].XNominal + p3[This].X);
sv[This].setPivotY(p3[This].YNominal + p3[This].Y);
sv[This].setRotation((float) Agen[This]);
}
return true;
}
});
findViewById(R.id.back).setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
// TODO Auto-generated method stub
finish();
}
});
}
class backGrShape extends View {
int top, left, position;
private Paint paint;
public backGrShape(Context context) {
super(context);
paint = new Paint();
paint.setStrokeWidth(2);
paint.setColor(0xff000000);
init();
}

private void init() {
// TODO Auto-generated method stub
setOnTouchListener(new OnTouchListener() {
@Override
```

```
public boolean onTouch(View v, MotionEvent event) {
// TODO Auto-generated method stub
switch (event.getAction()) {
case MotionEvent.ACTION_DOWN:
for (int i = 0; i < p3.length; i++)
if (event.getY() < p3[i].Ymax && event.getY() > p3[i].Ymin)
if (event.getX() < p3[i].Xmax && event.getX() > p3[i].Xmin) {
This = i;
break;
}

break;
case MotionEvent.ACTION_MOVE:
if (event.getY() < p3[This].Ymax && event.getY() > p3[This].Ymin)
if (event.getX() < p3[This].Xmax && event.getX() > p3[This].Xmin) {
sv[This].repaint((int) (event.getX()), (int) (event.getY()));
}
break;
default:
return false;
}
if (event.getY() > 40 && event.getX() > 40)
return true;
else return false;
}});}
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
// TODO Auto-generated method stub
super.onMeasure(widthMeasureSpec, heightMeasureSpec);
top = (getMeasuredHeight() -
(ObjectCmSpline.Ymax - ObjectCmSpline.Ymin)) / 2;

left = (getMeasuredWidth() - (ObjectCmSpline.Xmax -

ObjectCmSpline.Xmin)) / 2;
position = getRootView().getHeight() - top;
}
@Override
protected void onDraw(Canvas canvas) {
// TODO Auto-generated method stub
```

```
super.onDraw(canvas);
if (ObjectCmSpline.getInts() != null)
for (int i = 1; i < ObjectCmSpline.getInts().length; i++) {
canvas.drawLine((float) ObjectCmSpline.getInts()[i][1] + left,
(float) ObjectCmSpline.getInts()[i][0] + top,
(float) ObjectCmSpline.getInts()[i - 1][1] + left,
(float) ObjectCmSpline.getInts()[i - 1][0] + top, paint);
}
}
}

//отвечает за конкрентую фигуру то-есть ее отображение в виде
//View  и используется как массив таких фигур
//в эависимости сколько треугольников нужно

//столько будет и этих классов
class ShapeVision extends View {
Bitmap bm;
int[] pixels;

@Override
protected void onDraw(Canvas canvas) {
// TODO Auto-generated method stub
super.onDraw(canvas);
canvas.drawBitmap(bm, 0, 0, null);
}

public ShapeVision(Context context, Point x,

Point y, Point z, int w, int h, int color) {
super(context);//this.count=count;
pixels = new int[w * h];
int sroff = offsetX(z, x, y);
x.x -= sroff;
y.x -= sroff;
z.x -= sroff;
int sdoff = offsetY(z, x, y);
x.y -= sdoff;
y.y -= sdoff;
z.y -= sdoff;
```

```
x.x = limitMax(x.x, w);
y.x = limitMax(y.x, w);
z.x = limitMax(z.x, w);
x.y = limitMax(x.y, h);
y.y = limitMax(y.y, h);
z.y = limitMax(z.y, h);
Spline spline1 = new Spline

(DrawLineApprox(x.x, y.x, x.y, y.y));
Spline spline2 = new Spline

(DrawLineApprox(x.x, z.x, x.y, z.y));
Spline spline3 = new Spline

(DrawLineApprox(z.x, y.x, z.y, y.y));

spline1.addSpline(spline2);
spline1.addSpline(spline3)
bm = Bitmap.createBitmap(w, H, Bitmap.Config.ARGB_8888);
}

int w;
int color, i;

public ShapeVision(Context context, int ww, int color, int i) {
super(context);
w = ww;
this.i = i;
this.color = color;
bm = Bitmap.createBitmap(w, H, Bitmap.Config.ARGB_8888);
Point3 p = p3[i];
int df = w / 6;
p.X = df * i;
p.Y = (df / 2) * i;
int[] pX = p.getXs();
int[] pY = p.getYs();
pixels = new int[w * H];
Spline spline1 = new Spline

(DrawLineApprox(pX[0], pX[2], pY[0], pY[2]));
```

```
spline1.addSpline(new Spline

(DrawLineApprox(pX[0], pX[1], pY[0], pY[1])));
spline1.addSpline(new Spline

(DrawLineApprox(pX[2], pX[1], pY[2], pY[1])));
spline1.getPaint(pixels, w, color);
Nativ.Sizego(pixels, w, H, 0, color);
bm.setPixels(pixels, 0, w, 0, 0, w, H);
}
void repaint(int x, int y) {
Point3 p = p3[This];
p.Y = y - p.YNominal;
p.X = x - p.XNominal;
repaint();
}
void repaint() {
pixels = new int[w * H];
int[] pX = p3[i].getXs();
int[] pY = p3[i].getYs();
draws(pixels, w, color, pX[0], pY[0], pX[1], pY[1], pX[2], pY[2]);
Nativ.Sizego(pixels, w, H, 0, color);
bm.setPixels(pixels, 0, w, 0, 0, w, H);
postInvalidate();
}

int offsetX(Point pointsX, Point pointsY, Point pointsZ) {
return Math.min(pointsY.x, Math.min(pointsX.x, pointsZ.x));
}

int offsetY(Point pointsX, Point pointsY, Point pointsZ) {
return Math.min(pointsY.y, Math.min(pointsX.y, pointsZ.y));
}

int limitMax(int x, int limit) {
if (x >= limit) x = limit - 1;
if (x <= 0) x = 1;
return x;
}
```

```
void draws(int[] out, int w, int color, int pointsX,
int pointsY, int pointsX2, int pointsY2, int pointsX3,
int pointsY3) {
out = Nativ.TrivialAlg(out, pointsX,
pointsX3, pointsY, pointsY3, color, w);
out = Nativ.TrivialAlg(out, pointsX,
pointsX2, pointsY, pointsY2, color, w);
out = Nativ.TrivialAlg(out, pointsX3,
pointsX2, pointsY3, pointsY2, color, w);
}

double[][] DrawLineApprox(double x, double x1,
double y, double y1) {
double count = Math.max(Math.abs(x1 - x), Math.abs(y1 - y));
double[][] out = new double[(int) count][2];
double kfY = (y1 - y) / count;
double kfX = (x1 - x) / count;
out[0][0] = y;
out[1][0] = y + kfY * 1;
for (int i = 0; i < count; i++) {
out[i][1] = x + kfX * i;
double[] dd = getApprox(out, i);
if (!Double.isNaN(dd[0]))
out[i][0] = (dd[0] * out[i][1] + dd[1]);
/*  можно записать  иначе
*   else out[i][0]=out[i-1][0];
* но лучше добавить тогда побольше начальнах точек
*
* */
else out[i][0] = y + kfY * i;
}
return out;
}
}
double[] getApprox(double[][] m, int n) {
double sumx = 0;
double sumy = 0;
double sumx2 = 0;
double sumxy = 0;
for (int i = 0; i < n; i++) {
```

```
sumx += m[i][1];
sumy += m[i][0];
sumx2 += m[i][1] * m[i][1];
sumxy += m[i][0] * m[i][1];
}

double a = (n * sumxy - sumx * sumy) / (n * sumx2 - sumx * sumx);
double b = (sumy - a * sumx) / n;
return new double[]{a, b};
}
public static double distance(double x1, double y1,
  double x2, double y2) {
x1 -= x2;
y1 -= y2;
return Math.sqrt(x1 * x1 + y1 * y1);
}
```

Point 3

```
package com.example.shape2d;

//Этот класс отвечает за три точки образующий треугольник
//максимумы минимуму начальное значение и текущее смещение

public class Point3 {
public int x1,x2,x3,y1,y2,y3,X,Y,Xmax,Xmin,
Ymax,Ymin,XNominal,YNominal,RAD;
double xnk1,xnk2,xnk3,ynk1,ynk2,ynk3;

public int[] line1,line2,line3;
public int[] getXs(){
int[] out=new int[3];
Xmax=Math.max(x1+X, Math.max(x3+X, x2+X));
Xmin=Math.min(x1+X, Math.min(x3+X, x2+X));
out[0]=x1+X;
out[1]=x2+X;
out[2]=x3+X;

return out;
}
public int[] getYs(){
```

```
int[] out=new int[3];
Ymax=Math.max(y1+Y, Math.max(y3+Y, y2+Y));
Ymin=Math.min(y1+Y, Math.min(y3+Y, y2+Y));
out[0]=y1+Y;
out[1]=y2+Y;
out[2]=y3+Y;
return out;
}
public void setPoints(int xp0,int xp1,int xp2,
int yp0,int yp1,int yp2){
Ymax=Math.max(yp0, Math.max(yp2, yp1));
Ymin=Math.min(yp0, Math.min(yp2, yp1));
Xmax=Math.max(xp0, Math.max(xp2, xp1));
Xmin=Math.min(xp0, Math.min(xp2, xp1));
x1=xp0-Xmin;x2=xp1-Xmin;x3=xp2-Xmin;
y1=yp0-Ymin;y2=yp1-Ymin;y3=yp2-Ymin;
Xmax-=Xmin;Xmin=1; Ymax-=Ymin;Ymin=1;
XNominal=(Xmax)/2;
YNominal=(Ymax)/2;
RAD= (int) Math.sqrt(XNominal*XNominal+YNominal*YNominal);
ynk1=3.14/(1.57+ (YNominal-y1)/(double) YNominal);
ynk2=3.14/(1.57+ (YNominal-y2)/(double) YNominal);
ynk3=3.14/(1.57+ (YNominal-y3)/(double) YNominal);
xnk2=XNominal-x2;ynk2=YNominal-y2;
xnk3=x3;ynk3=y3;
}

double findang(int in){
double out=0;
for(out=0;out<7;out+=0.01)
if((int)  (YNominal-Math.sin(out)*YNominal)==in)break;
return out;
}
public void angrad(double angradq){
double dis = Math.toRadians(angradq);
}
double dk(double a,double c){
double out=Math.cos((c)/a);
return out;
}
```

```
public void setPointsNomin(int xp0,int xp1,int xp2,
int yp0,int yp1,int yp2){

line1=new int[6];
line1[0]=xp0;line1[1]=yp0;line1[2]=
(xp0+xp1)/2;line1[3]=(yp0+yp1)/2;line1[4]=xp1;line1[5]=yp1;
line2=new int[6];
line2[0]=xp1;line2[1]=yp1;line2[2]=
(xp2+xp1)/2;line2[3]=(yp2+yp1)/2;line2[4]=xp2;line2[5]=yp2;
line3=new int[6];
line3[0]=xp2;line3[1]=yp2;line3[2]=
(xp0+xp2)/2;line3[3]=(yp0+yp2)/2;line3[4]=xp0;line3[5]=yp0;
x1=xp0;
x2=xp1;
x3=xp2;
y1=yp0;
y2=yp1;
y3=yp2;
}
public static int distance(double x1, double y1,
   double x2, double y2)

{
x1 -= x2;
y1 -= y2;
return (int) (x1 * x1 + y1 * y1);
}
}
```

Класс Shape

```
package com.example.shape2d;
public interface Shape {
Object[][] getObject();
void setDoubles(double[][] in);
void setInts(int[][] in);
int[][] getInts();
double[][] getDoubles();
void setObject(Object[][] object);
}
```

Класс Shape

```
package com.example.shape2d;
public interface Shape {
Object[][] getObject();
void setDoubles(double[][] in);
void setInts(int[][] in);
int[][] getInts();
double[][] getDoubles();
void setObject(Object[][] object);
}
```

Класс Shape2D

```
package com.example.shape2d;


import com.example.shape2d.ShapesActivity.ShapeVision;
import com.example.shape2d.ShapesActivity.backGrShape;

import android.app.Activity;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Bitmap.Config;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.os.Bundle;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.ImageView;
import android.widget.RelativeLayout;

public class Shape2d extends Activity {
int[] colors = new int[]{0xffFF4081, 0xffF44336,
 0xff2196F3, 0xff4CAF50, 0xff3F51B5, 0xff303F9F};
Spline ObjectCmSpline, PointsOpor, Lines1, Lines2;
RelativeLayout root_view;
int offsetY, offsetX;
double[] Agen;
```

```
int H, w;
int This;
double k;
Point3[] p3;
ShapeVision[] sv;
@Override
protected void onCreate(Bundle savedInstanceState) {
// TODO Auto-generated method stub
super.onCreate(savedInstanceState);
int[] as = getIntent().getIntArrayExtra("Points");
offsetY = getIntent().getIntExtra("offsetY", 0);
offsetX = getIntent().getIntExtra("offsetX", 0);
w = getIntent().getIntExtra("w", 0);
int[] PointsO = getIntent().getIntArrayExtra("PointsOpor");
for (int i = 0; i < PointsO.length / 2; i++) {
PointsO[i * 2] += offsetX;
PointsO[i * 2 + 1] += offsetY;
}
if (PointsO != null) PointsOpor = new Spline
(new int[]{10, 100, 30, 30});
for (int i = 0; i < as.length / 2; i++) {
as[i * 2] += offsetX;
as[i * 2 + 1] += offsetY;
}
int[] newas = new int[]{30, 30, 130, 400};
PointsOpor = new Spline(newas);
PointsOpor.addSpline(new Spline(new int[]{130, 400, 430, 20}));

H = (int) getIntent().getFloatExtra("Heig", 0);
double k = getIntent().getDoubleExtra("KFWH", 1);
setContentView(R.layout.activity_rhombus);
findViewById(R.id.jostic).setVisibility(View.GONE);
root_view = (RelativeLayout) findViewById(R.id.root_view);
root_view.addView(new backGrShape(this), -1, H);
if (as != null) {
ObjectCmSpline = new Spline();
int[][] ObjectCm = new int[as.length / 2][2];
for (int i = 0; i < ObjectCm.length; i++) {
ObjectCm[i][1] = (int) (as[i * 2] * k);
ObjectCm[i][0] = (int) (as[i * 2 + 1] * k);
```

```
;
}
ObjectCmSpline.setInts(ObjectCm);
p3 = new Point3[ObjectCm.length / 3];

Agen = new double[p3.length];
//переводят координаты в значения точек в классе точек 3
//значит каждые  3 точки образуют треугольник
Agen[i] = 0;// - угол 0
for (int i = 0; i < ObjectCm.length / 3; i++) {
p3[i] = new Point3();
p3[i].setPoints(ObjectCm[i * 3][1], ObjectCm[i * 3 + 1][1],
 ObjectCm[i * 3 + 2][1], ObjectCm[i * 3][0],
 ObjectCm[i * 3 + 1][0], ObjectCm[i * 3 + 2][0]);
Agen[i] = 0;
This = i;

}
for (int i = 0; i < p3.length; i++) {}}}
class backGrShape extends ImageView {
int top, left, position, x1, y1;
private Paint paint;
public backGrShape(Context context) {
super(context);
paint = new Paint();
paint.setStrokeWidth(2);
paint.setColor(colors[1]);
init();
}
Bitmap bitmap;
private void init() {
// TODO Auto-generated method stub
bitmap = Bitmap.createBitmap(w, H, Bitmap.Config.ARGB_4444);
int[] in = new int[w * H];
x1 = 100;
y1 = 100;
double[][] ast = new double[][]{{0, 150}, {x1, y1}, {w, 175}};
setHroxim(in, ast);
double[][] ast2 = new double[][]{{0, 150}, {300, 300}, {w, 175}};

setHroxim(in, ast2);
```

```
setImageBitmap(bitmap);
setOnTouchListener(new OnTouchListener() {
@Override
public boolean onTouch(View v, MotionEvent event) {
// TODO Auto-generated method stub
switch (event.getAction()) {
case MotionEvent.ACTION_DOWN:
break;
case MotionEvent.ACTION_MOVE:
x1 = (int) event.getX();
y1 = (int) event.getY();
repaints();
break;
default:
return false;
}
return true;}});}
private void repaints() {
int[] in = new int[w * H];
double[][] ast = new double[][]{{0, 150}, {x1, y1}, {w, 175}};
// double[][] out = getAppS(  ast );
setHroxim(in, ast);
double[][] ast2 = new double[][]{{0, 150}, {300, 300}, {w, 175}};
setHroxim(in, ast2);
postInvalidate();
}
private void setHroxim(int[] in, double[][] ast2) {
double[][] out2 = getAppS(ast2);
Spline Lines = new Spline(out2);
Lines.getPaint(in, w, colors[4]);
Sizego(in, w, H, 0, colors[4]);
bitmap.setPixels(in, 0, w, 0, 0, w, H);
}

@Override
protected void onMeasure(int widthMeasureSpec,
int heightMeasureSpec) {
// TODO Auto-generated method stub
super.onMeasure(widthMeasureSpec, heightMeasureSpec);
top = (getMeasuredHeight() -
```

```
(ObjectCmSpline.Ymax - ObjectCmSpline.Ymin)) / 2;
left = (getMeasuredWidth() -
(ObjectCmSpline.Xmax - ObjectCmSpline.Xmin)) / 2;
position = getRootView().getHeight() - top;

}
@Override
protected void onDraw(Canvas canvas) {
// TODO Auto-generated method stub
super.onDraw(canvas);
canvas.drawCircle(x1, y1, 5, paint);
}

}
double[][] getAppS(double[][] xF) {
int i = 0;
double[] xww = new double[3];
for (int j = 0; j < 3; j++) xww[j] = xF[j][0];
double[] yw = new double[3];
for (int j = 0; j < 3; j++) yw[j] = xF[j][1];
Spline.SplineTuple[] spl = Spline.BuildSpline(xww, yw);
int count = Math.min(w, H) / 2;
double kfy = 2 * Math.abs(xF[i + 1][0] - xF[i][0]) / (double) count;
double[][] out = new double[count][2];
for (int j = 0; j < count / 2 + 1; j++) {
double ff = Spline.f(xF[i][0] + j * kfy, spl);
out[j][1] = xF[i][0] + j * kfy;
out[j][0] = ff;
}
i++;
kfy = 2 * (xF[i + 1][0] - xF[i][0]) / (double) count;

for (int j = count / 2; j < count; j++) {
double ff = Spline.f(xF[i][0] + (count - 2 - j) * kfy, spl);
out[j][1] = xF[i][0] + (count - 2 - j) * kfy;
out[j][0] = ff;
}
return out;
}
double[][] changeArray(double[][] in) {
```

```
double[][] out = new double[in.length][2];
for (int i = 0; i < in.length; i++) {
out[i][0] = in[i][1];
out[i][1] = in[i][0];
}

return out;
}
void Sizego(int[] out, int rad, int h, int starty, int color) {
int j;

for (int i = starty; i < rad; i++) {
int e1 = h - 1, e2 = -1;
for (j = 0; j < h; j++) {
if (out[i + j * rad] == color) {
e1 = j;
break;
}
}
for (j = h - 1; j > starty; j--) {
if (out[i + j * rad] == color) {
e2 = j;
break;
}
}
for (j = e1; j < e2; j++) out[i + j * rad] = color;}}
class paste {
int[] in;
int len = 0;
int w;
int xmax = 0, xmin = H, ymax = 0, ymin = w;

public paste(int[] ins, int ww) {
super();
in = ins;
len = in.length;
w = ww;
// TODO Auto-generated constructor stub
}

synchronized void matrix2(int isc, int color1, int color2)
```

```
{
xmax = Math.max(xmax, isc % w);
ymax = Math.max(ymax, isc / w);
xmin = Math.min(xmin, isc % w);
ymin = Math.min(ymin, isc / w);
if (isc - w + 1 < len && isc - w + 1 > 0
&& isc % w > 0 && isc % w < w)
if (in[isc - w + 1] != color1 && in[isc - w + 1] != color2) {
in[isc - w + 1] = color2;
try {
matrix2(isc - w + 1, color1, color2);
} catch (StackOverflowError e) {
}
}
if (isc + 1 < len && isc + 1 > 0 && isc % w > 0 && isc % w < w)
if (in[isc + 1] != color1 && in[isc + 1] != color2) {
in[isc + 1] = color2;
try {
matrix2(isc + 1, color1, color2);
} catch (StackOverflowError e) {
}
}
if (isc + w + 1 < len && isc + w + 1 > 0
&& isc % w > 0 && isc % w < w)
if (in[isc + w + 1] != color1 && in[isc + w + 1] != color2)
{
in[isc + w + 1] = color2;
try {
matrix2(isc + w + 1, color1, color2);
} catch (StackOverflowError e) {
}

}
if (isc - w < len && isc - w > 0 && isc % w > 0
&& isc % w < w)
if (in[isc - w] != color1 && in[isc - w] != color2) {
in[isc - w] = color2;
try {
matrix2(isc - w, color1, color2);
} catch (StackOverflowError e) {
```

```
}

}
if (isc - w - 1 < len && isc - w - 1 > 0
&& isc % w > 0 && isc % w < w)
if (in[isc - w - 1] != color1
&& in[isc - w - 1] != color2) {
in[isc - w - 1] = color2;
try {
matrix2(isc - w - 1, color1, color2);
} catch (StackOverflowError e) {
}

}
if (isc - 1 < len && isc - 1 > 0 && isc % w > 0
&& isc % w < w)
if (in[isc - 1] != color1 && in[isc - 1] != color2) {
in[isc - 1] = color2;
try {
matrix2(isc - 1, color1, color2);
} catch (StackOverflowError e) {
}
}
if (isc + w < len && isc + w > 0 && isc % w > 0
&& isc % w < w)
if (in[isc + w] != color1 && in[isc + w] != color2) {
in[isc + w] = color2;
try {
matrix2(isc + w, color1, color2);
} catch (StackOverflowError e) {
}

}
if (isc + w - 1 < len && isc + w - 1 > 0
&& isc % w > 0 && isc % w < w)
if (in[isc + w - 1] != color1 && in[isc + w - 1] != color2) {
in[isc + w - 1] = color2;
try {
matrix2(isc + w - 1, color1, color2);
} catch (StackOverflowError e) {}}}}}
```

Клас Spline

```
package com.example.shape2d;

import android.util.Log;

/*
* Главным образом хранит координаты прямых в виде 2-го массива
* Универсализм задания.. можно задавать как int так и double
* в принципе излишество
* */
public class Spline  implements Shape{
int Xmax,XmaxPos,Xmin=Integer.MAX_VALUE,XminPos,
Ymax,YmaxPos,Ymin=Integer.MAX_VALUE,YminPos;
public Spline(int[][] in) {
super();
setInts(in);
}
public Spline(double[][] in) {
super();
setDoubles(in);
}
public Spline() {
super();
}

public Spline(int[]in) {
super();
setDoubles(createSpline(in));
}
Object[][] arrayAplexsation;
@Override
public void setInts(int[][] in) {
// TODO Auto-generated method stub
arrayAplexsation=new Object[in.length][];
int c=in.length;
for(int i=0;i<c;i++){
int b=in[i].length;
arrayAplexsation[i]=new Object[b];
```

```
for(int j=0;j<b;j++){
int as = in[i][1];
if(as>Xmax){
Xmax=as;XmaxPos =i;
}

if(Xmin>as){
Xmin=as;XminPos =i;
}
int sa = in[i][0];
if(sa>Ymax){
Ymax=sa;YmaxPos =i;
}
if(Ymin>sa){
Ymin=sa;YminPos =i;
}
arrayAplexsation[i][j]=in[i][j];
}}}
@Override
public void setDoubles(double[][] in) {
// TODO Auto-generated method stub
arrayAplexsation=new Object[in.length][];
int c=in.length;
for(int i=0;i<c;i++){
int b=in[i].length;
arrayAplexsation[i]=new Object[b];
for(int j=0;j<b;j++){
arrayAplexsation[i][j]=in[i][j];
double as = in[i][1];
if(as>Xmax){
Xmax=(int) as;XmaxPos =i;
}
if(Xmin>as){
Xmin=(int) as;XminPos =i;
}
double sa = in[i][0];
if(sa>Ymax){
Ymax=(int) sa;YmaxPos =i;
}
if(Ymin>sa){
```

```
Ymin=(int) sa;YminPos =i;
}}}}
@Override

public Object[][] getObject() {
// TODO Auto-generated method stub
return arrayAplexsation;
}
@Override
public int[][] getInts() {
int[][] out=new int[arrayAplexsation.length][];
int c=arrayAplexsation.length;
for(int i=0;i<c;i++){
int b=arrayAplexsation[i].length;
out[i]=new int[b];
for(int j=0;j<b;j++)
try{
out[i][j]=  (Integer) arrayAplexsation[i][j];
}catch(java.lang.ClassCastException t){

double  oi=(Double) arrayAplexsation[i][j];
out[i][j]=(int) oi;
}

}
return out;
}
@Override
public double[][] getDoubles() {
double[][] out=new double[arrayAplexsation.length][];
int c=arrayAplexsation.length;
for(int i=0;i<c;i++){
int b=arrayAplexsation[i].length;
out[i]=new double[b];
for(int j=0;j<b;j++)
out[i][j]= (Double) arrayAplexsation[i][j];
}
return out;
}
@Override
```

```
public void setObject(Object[][] object) {
// TODO Auto-generated method stub
arrayAplexsation=object;
}

Object[][] Points(Spline a){
Object[][] out=new Object[2][4];
int x1=0,x2=0,y1=0,y2=0;
Spline amin,amax;
if(arrayAplexsation.length > a.arrayAplexsation.length){
amax=this;
amin=a;
}
else {
amax=a;
amin=this;
}
int tx=0,ty,sx,sy;
out1: for(int i=0;i<amin.arrayAplexsation.length;i++)
for(int j=0;j<amax.arrayAplexsation.length;j++)

if(amin.arrayAplexsation[i][0]== amax.arrayAplexsation[j][0])
{
for(int jy=j;jy<amax.arrayAplexsation.length;jy++)
if(amin.arrayAplexsation[i][1] == amax.arrayAplexsation[jy][1]
&& amin.arrayAplexsation[i][0] == amax.arrayAplexsation[jy][0])
{

tx=i;
y1=(Integer) amax.arrayAplexsation[jy][0];
x1=(Integer) amax.arrayAplexsation[jy][1];

break out1;
}  }
int[] sxw = sortmin(amin, amax, tx ,1);
x2=sxw[1];
y2=sxw[0];
out[0][0]=y1;
out[1][0]=y2;
out[0][1]=x1;
```

```
out[1][1]=x2;
return out;
}
int[] sortmin(Spline amin,Spline amax,int NULL,int xy){
int[] out=new int[2];
out2: for(int i=amin.arrayAplexsation.length-1;i>0;i--)
for(int j=amax.arrayAplexsation.length-1;j>0;j--)
if(amin.arrayAplexsation[i][0]== amax.arrayAplexsation[j][0])
{
for(int jy=j;jy>0;jy--)
if(amin.arrayAplexsation[i][1]== amax.arrayAplexsation[jy][1]
&& amin.arrayAplexsation[i][0]== amax.arrayAplexsation[jy][0] )
{
out[0]=(Integer) amax.arrayAplexsation[jy][0];
out[1]=(Integer) amax.arrayAplexsation[jy][1];
break out2;
}    }
return out;
}
void addSpline(Spline a){
Object[][] outV = new Object[arrayAplexsation.
length+a.arrayAplexsation.length]
[Math.max(arrayAplexsation[0].length,
a.arrayAplexsation[0].length)];
int count=0;
for(int i=0;i<arrayAplexsation.length;i++){
for(int j=0;j<arrayAplexsation[i].length;j++)
outV[i][j]=arrayAplexsation[i][j];
count++;
}
for(int i=0;i<a.arrayAplexsation.length;i++){
for(int j=0;j<a.arrayAplexsation[i].length;j++)
outV[count][j]=a.arrayAplexsation[i][j];
count++;
}

Ymax=Math.max(Ymax, a.Ymax);
Xmax=Math.max(Xmax, a.Xmax);
Ymin=Math.min(Ymin, a.Ymin);
Xmin=Math.min(Xmin, a.Xmin);
```

```
arrayAplexsation=outV;
}
void getPaint(int[] in,int width,int color){
int[][] ob = addob ();     // getInts();
// addob ();
int count = ob.length;
for(int i=0;i<count;i++){
int a=ob[i][1];
if(a> width-1)a=width-1;
if(a<0)a=0;
int x=ob[i][0];
if(x<0)x=0;
if(x*width+a<in.length)
in[x*width+a]=color;
}
}
int insort(Object[] in,Object pr){
int out=-1;
for(int i=in.length-1;i>0;i--)
if(in[i]==pr){
out=i;
break;
}

return out;
}
int[][] addob (){
int[][] ob = getInts();
int count = ob.length;
int c=0;

for(int i=1;i<count;i++)

c+= getAddcount(ob[i][1],ob[i-1][1],ob[i][0],ob[i-1][0]);
int[][] iit = getAddInts(ob[count-1][1],ob[count/2-1][1],
ob[count-1][0],ob[count/2-1][0]);
int[][] out =new int[count+c+iit.length][2];
c=0;
for(int i=1;i<count;i++){
int[][] iin = getAddInts(ob[i][1],ob[i-1][1],ob[i][0],ob[i-1][0]);
```

```
if(count/2!=i)
for(int j=0;j<iin.length;j++){
out[c][1]=iin[j][1];
out[c][0]=iin[j][0];
c++;}

}
for(int j=0;j<iit.length;j++){
out[c][1]=iit[j][1];
out[c][0]=iit[j][0];
c++;}
return out;}
int getAddcount(int x,int x1,int y,int y1){
int out = Math.max(Math.abs(x1-x),Math.abs (y1-y));
return out;
}
int[][] getAddInts(int x,int x1,int y,int y1){
int count=getAddcount(x,x1,y,y1);
int[][] out=new int[count][2];
if(count>0){
double kfY = (y1-y)/(double)count;
double kfX = (x1-x)/(double)count;
for(int i=0;i<count;i++){
out[i][1]=(int) (x+kfX*i);
out[i][0]=(int) (y+kfY*i);
}
}

return out;
}
public double[][] createSpline(int[] in){
return PointsLine((double)in[0] ,(double)in[2] ,
(double)in[1] ,(double)in[3]);
}
double[][] PointsLine (double x,double x1,double y,double y1){
double count = Math.max(Math.abs(x1-x),Math.abs (y1-y));
double[][] out=new double[(int) count][2];
double kfY = (y1-y)/count;
double kfX = (x1-x)/count;

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

```
out[i][1]=x+kfX*i;
out[i][0]=y+kfY*i;
}
return out;
}
int[] sort(int[] in, int rash,int I){
int[] out=new int[in.length];
int count=0;
for(int i=0;i<in.length;i+=4)if(in[i+rash]==I){
out[count]=in[i];out[count+1]=in[i+1];
out[count+2]=in[i+2];out[count+3]=in[i+3];
count+=4;
}
if(count>0){
int[] outs=new int[count];
for(int i=0;i<outs.length;i++)outs[i]=out[i];
return outs;  }
else return null;
}
public static class SplineTuple {

public double x;
public double a;
public double c;
public double d;
public double b;
public double y;
public double h;
}
public static  SplineTuple[] BuildSpline(double[] x, double[] y) {
int n=x.length;
Spline.SplineTuple[] splines = new SplineTuple[n];
for(int i=0;i<n;i++)
splines[i] = new SplineTuple();

for (int i = 0; i < n; ++i) {
splines[i].x = x[i];
splines[i].a = y[i];
}
splines[0].c = splines[n - 1].c = 0.0;
```

```
double[] alpha = new double[n - 1];
double[] beta = new double[n - 1];
alpha[0] = beta[0] = 0.0;
for (int i = 1; i < n - 1; ++i) {
double h_i = x[i] - x[i - 1], h_i1 = x[i + 1] - x[i];
double A = h_i;
double C = 2.0 * (h_i + h_i1);
double B = h_i1;
double F = 6.0 * ((y[i + 1] - y[i]) / h_i1 -
(y[i] - y[i - 1]) / h_i);
double z = (A * alpha[i - 1] + C);
alpha[i] = -B / z;
beta[i] = (F - A * beta[i - 1]) / z;
}
for (int i = n - 2; i > 0; --i)
splines[i].c = alpha[i] * splines[i + 1].c + beta[i];
beta = null;
alpha = null;
for (int i = n - 1; i > 0; --i) {
double h_i = x[i] - x[i - 1];
splines[i].d = (splines[i].c - splines[i - 1].c) / h_i;
splines[i].b = h_i * (2.0 * splines[i].c +
splines[i-1].c) / 6.0 + (y[i] - y[i - 1]) / h_i;
}
return splines;
}

public static double f(double x,SplineTuple[] splines ) {
SplineTuple s;
int n= splines.length;
if (x <= splines[0].x)
s = splines[1];
else if (x >= splines[n - 1].x)
s = splines[n - 1];
else
{
int i = 0, j = n - 1;
while (i + 1 < j) {
int k = i + (j - i) / 2;
if (x <= splines[k].x)
```

```
j = k;
else
i = k;
}
s = splines[j];
}

double dx = (x - s.x);
return s.a + (s.b + (s.c / 2.0 + s.d * dx / 6.0) * dx) * dx;
}
static double[][] DrawLineApprox(double x,double x1,
double y,double y1){
double count = Math.max(Math.abs(x1-x),Math.abs (y1-y));
double[][] out=new double[(int) count][2];
double kfY = (y1-y)/count;
double kfX = (x1-x)/count;
out[0][0]=y;
out[1][0]=y+kfY*1;
for(int i=0;i<count;i++){
out[i][1]=x+kfX*i;
 double[] dd = getApprox(out,i);
if(!Double.isNaN(dd[0]) )
out[i][0]= (dd[0]*out[i][1]+dd[1]);
/*  можно записать  иначе
*   else out[i][0]=out[i-1][0];
* но лучше добавить тогда побольше начальнах точек
*
* */
else out[i][0]=y+kfY*i;
}

return out;
}
static double[] getApprox(double[][] m,  int n) {
double sumx = 0;
double sumy = 0;
double sumx2 = 0;
double sumxy = 0;
for (int i = 0; i<n; i++) {
sumx += m[i][1];
```

```
sumy += m[i][0];
sumx2 += m[i][1] * m[i][1];
sumxy += m[i][0] * m[i][1];
}
double a = (n*sumxy - sumx*sumy) / (n*sumx2 - sumx*sumx);
double  b = (sumy - a*sumx) / n;
return new double[]{a,b};
}
}
```

Класс Nativ

```
package com.example.shape2d;

class Nativ {
native static int[] TrivialAlg
//рисует линию как drawLine - cavnas
(int[] out,int x1,int x2,int y1,int y2,int barva,int width);
//зарисовывает фигуру
native static void Sizego
(int[] in,int rad,int h,int starty ,int color);
native static int[] raundImageCant
(int[] out,int w,int h,int color);
native static void matrx2
(int[] packName,int isc,int w,int len,int color1,int color2);
//аналог функции аппроксимации
native static int[] DrawLineApprox
(int[] out, int x1,int x2,int y1,int y2, int barva ,int w);
//x1 y1 -первая точка x2 y2 -вторая  barva цвет width ШИРИНА
native static int[]  DrawLinePlus
(int[] out, int len,int barva ,int w);
static {
System.loadLibrary("b431");
}
}
```

Класс TreeActivity

```
package com.example.shape2d;
```

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.ArrayList;
import android.app.Activity;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Point;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
public class TreeActivity1 extends Activity{
Shape v;
private int w,h;
@Override
protected void onCreate(Bundle savedInstanceState) {
// TODO Auto-generated method stub
super.onCreate(savedInstanceState);
w=getIntent().getIntExtra("w", 0);
h=(int) getIntent().getFloatExtra("Heig", 0);
setContentView(R.layout.treeac);
v=new Shape(getApplicationContext());
((ViewGroup) findViewById(R.id.g)).addView(v);
//очистка, удаление последнего действия
findViewById(R.id.kill).setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
// TODO Auto-generated method stub
kill(v);
}

});
```

```
//созданиие файла с координатами
findViewById(R.id.create).setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
// TODO Auto-generated method stub
create();
}
});
}
class Shape extends View{
//нарастающие точки,при касании экрана определяющего x  и y
ArrayList<Point> al;
ArrayList<Point3> add;//прирастающие треугольники
Bitmap bm;//это отражение рисунка который получается через canvas
int[] pixels;
// закрашивание невидимых точек;
public Shape(Context context) {
super(context);
setBackgroundColor(0xffffffff);
pixels=new int[w*h];
bm = Bitmap.createBitmap(w,  h, Bitmap.Config.ARGB_8888);
al=new ArrayList<Point>();
setOnTouchListener(new OnTouchListener() {
@Override
public boolean onTouch(View v, MotionEvent event) {
int x1;
int y1;

// TODO Auto-generated method stub
switch(event.getAction()){
case MotionEvent.ACTION_DOWN:
break;
case MotionEvent.ACTION_UP:
 if(event.getY()>30){
x1=(int) event.getX();
y1=(int) event.getY();
al.add(new Point( x1,y1));//добавляются к уже имеющимся
remath();   }
break;
```

```
default :
return false;
}
return true;}});
}
void remath() {
// TODO Auto-generated method stub
if(al.size()==2){
Spline spline =new Spline
(Spline.DrawLineApprox(al.get(0).x, al.get(1).x, al.get(0).y,
al.get(1).y));//рисуем простую линию
spline.getPaint( pixels, w, 0xff000000);
bm.setPixels(pixels, 0,w, 0, 0,w, h);
postInvalidate();
}
if(al.size()==3){
//начинаем добавлять треугольники и сразу их рисуем
add=new ArrayList<>();
Point3 ss = new Point3();
ss.setPointsNomin(al.get(0).x, al.get(1).x,
al.get(2).x,al.get(0).y, al.get(1).y, al.get(2).y);
add.add(ss);
endPaint();
}
if(al.size()>3){
Point3 aee=new Point3();
int xp1=0,xp2=0,yp1=0,yp2=0;
//просто какое-то большое число - ведь мы ищем наименьшее значение
int s=1000003;
for(int j=0;j<add.size();j++)
{

//переделанная функция делоне,там поиск по окружности - это долго
//вместо этого берутся середины ребер и сравниваются
//при помощи Point3.distance
int[] ffg = perfect(add.get(j),al.get(al.size()-1).
//функция определения ближайшего ребра из массива треугольников
x,al.get(al.size()-1).y  );
//  ffg[2] параметр возвращаемый perfect
//неизвестно сколько сейчас треугольников поэтому сравниваем каждый
```

```
if(ffg[2]<s){
s=ffg[2];
xp1=ffg[3];xp2=ffg[4];
yp1=ffg[5];yp2=ffg[6];// x1 x2  y1 y2  текущий минимальный
//ближайшее ребро - когда цикл завершится
}}
//создаем по итогам новый тругольник
aee.setPointsNomin(al.get(al.size()-1).x, xp1, xp2,
al.get(al.size()-1).y, yp1, yp2);
add.add(aee);//добавляем к массиву
endPaint();//конечная зарисовка
}}
//определяем ближайшее ребро
int[] perfect(Point3 mass, int x,int y){
int si1 = Point3.distance(mass.line1[2], mass.line1[3], x, y);
int si2= Point3.distance(mass.line2[2], mass.line2[3], x, y);
int si3= Point3.distance(mass.line3[2], mass.line3[3], x, y);
//минимальное значение ,это массив который вернется он передает
//только расстояние и координаты ребра
int [] m=new int[9];
//минимум si3, si1 сравнивается с si2
int abc = Math.min(si2, Math.min(si3, si1));
m[2]=abc;
//если   то это ребро найдено заполняем массив
//координатами,если оно то наше ребро 1
if(abc==si1){

m[1]=0;
m[3]=mass.line1[0];m[4]=mass.line1[4];
m[5]=mass.line1[1];m[6]=mass.line1[5];
}
//если   то это ребро найдено заполняем массив координатами
if(abc==si2){ m[1]=1;
m[3]=mass.line2[0];m[4]=mass.line2[4];
m[5]=mass.line2[1];m[6]=mass.line2[5];
}
///если   то это ребро найдено заполняем массив координатами
if(abc==si3){ m[1]=2;
m[3]=mass.line3[0];m[4]=mass.line3[4];
m[5]=mass.line3[1];m[6]=mass.line3[5];
```

```
}
return m;
}
void endPaint(){
pixels=new int[w*h];//новый массив для экрана
//Spline[] spline это массив сплайнов готовая функция для
//рисования еще с первых треугольков

//количество треугольников равно количеству рисунков
Spline[] spline =new Spline[add.size()];
for(int i=0;i<spline.length;i++){
spline[i]= new Spline(Spline.DrawLineApprox
(add.get(i).x1, add.get(i).x2, add.get(i).y1, add.get(i).y2));
spline[i].addSpline(new Spline(Spline.DrawLineApprox(add.get
(i).x3, add.get(i).x1, add.get(i).y3, add.get(i).y1)) );

spline[i].getPaint( pixels, w, 0xff000000);//линнии будут черными
}
//загружаем пиксели в битовую маску
bm.setPixels(pixels, 0,w, 0, 0,w, h);
postInvalidate();//обновляем экран
}
@Override
protected void onDraw(Canvas c) {
// TODO Auto-generated method stub
super.onDraw(c);

if(bm!=null)
c.drawBitmap(bm,0,0, null);

}
}
public void kill(View vr){
// Log.e("", String.valueOf(v.add.size() ));
if(v.add!=null)
if(v.add.size()>1){
v.add.remove(v.add.size()-1);
v.al.remove(v.al.size()-1);
v.endPaint();  }
}
```

```
public void create(){
String Pach=Environment.getExternalStorageDirectory()+"/shape2d/";
File dir=new File(Pach);dir.mkdirs();
File f=new File(dir,"point.txt");
try {
RandomAccessFile dr = new RandomAccessFile(f, "rw");
if(v.add!=null)
for(int i=0;i<v.add.size();i++){
dr.writeBytes(String.valueOf("x1="+v.add.get(i).x1+"px
y1="+v.add.get(i).y1 + "px\n" ));
dr.writeBytes(String.valueOf("x2="+v.add.get(i).x2+"px
y2="+v.add.get(i).y2 + "px\n" ));
dr.writeBytes(String.valueOf("x3="+v.add.get(i).x3+"px
y3="+v.add.get(i).y3 + "px\n" ));
dr.writeBytes(
String.valueOf("---------------------------------\n" ));
}

} catch (FileNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}}}
```