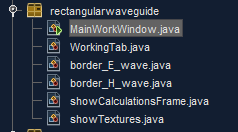
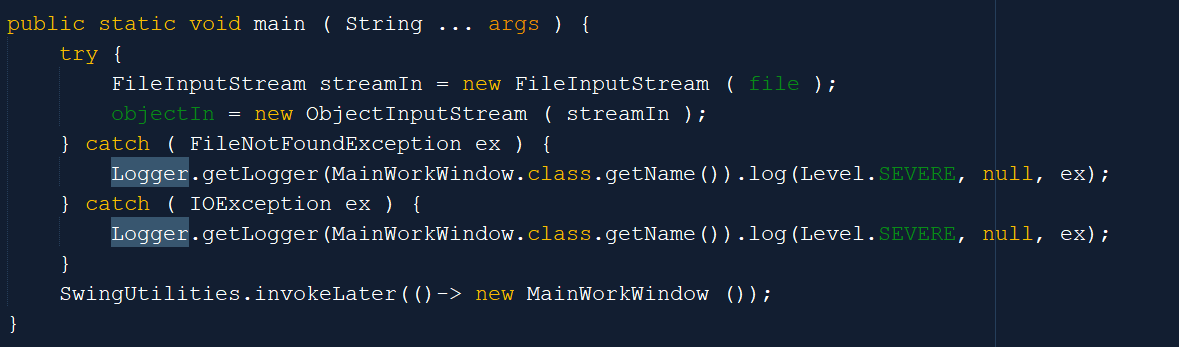
Код программы располагается в 6 папках.

## Каталог rectangularwaveguide

В папке rectangularwaveguide содержаться классы основного графического интерфейса:



Класс MainWorkWindow является главным классом программы. Именно в нём располагается метод main. 

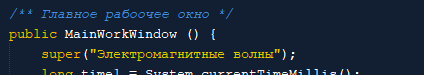
В начале создаётся объект потокового вода объектов. Это необходимо для сериализации объекта класса ElectromagneticWave. Далее в потоке диспетчеризации событий запускается основное рабочее окно:

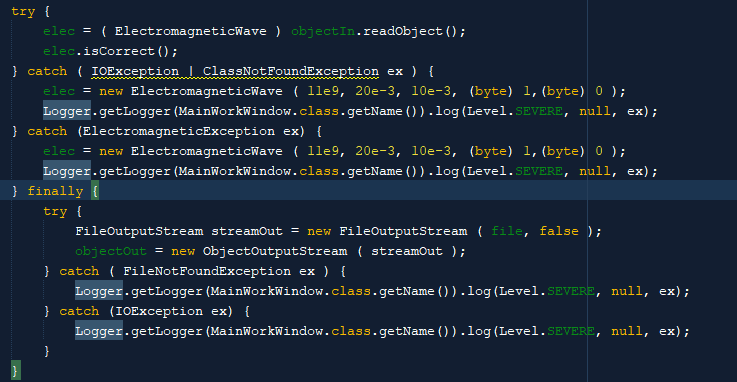


При создании объекта рабочего окна десиреализуется объект ElectromagneticWave, который хранит последний созданный объект электромагнитной волны. Если параметры объекты не каретные, то будет создан стандартный объект со следующими параметрами:

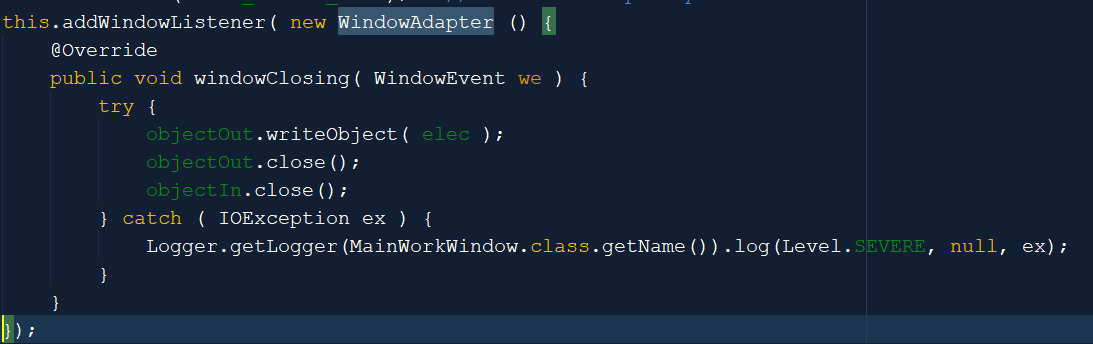


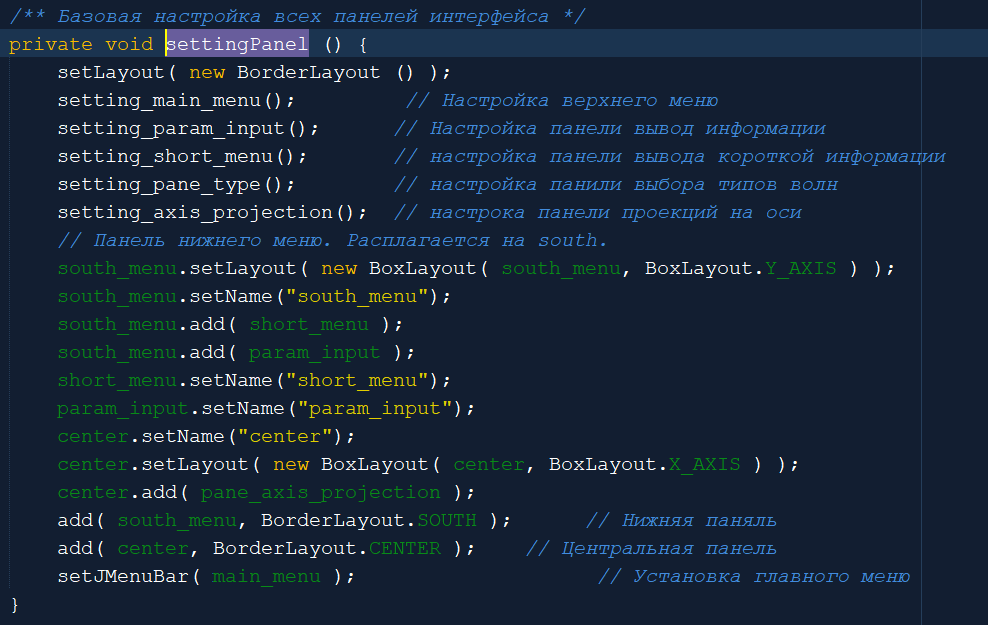
Блок кода располагается в конструкторе:



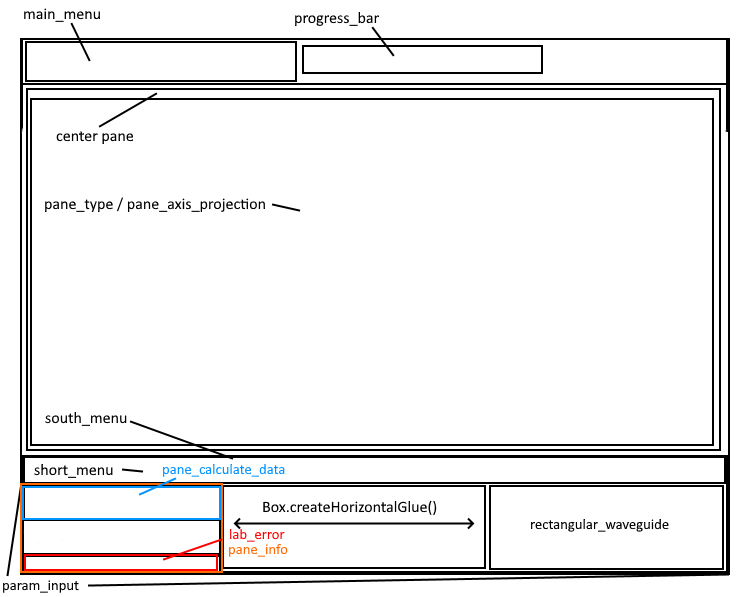


К окну был добавлен слушатель события окна для того, чтобы сериализовать объект ElectromagneticWave при закрытии кона приложения.



Метод settingPanel выполняет настройку панелей графического интерфейса. 

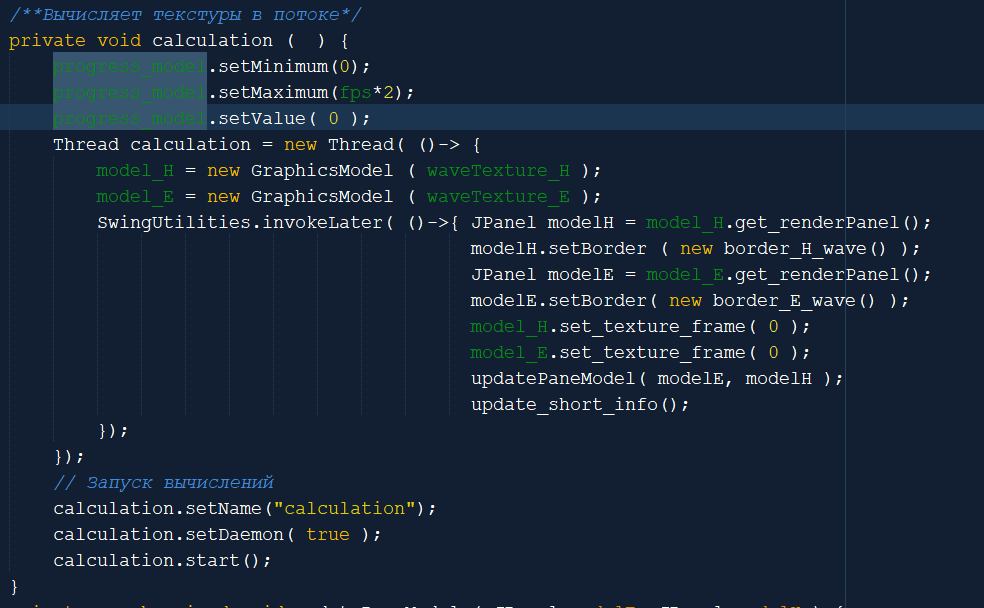
Структура рабочего окна:



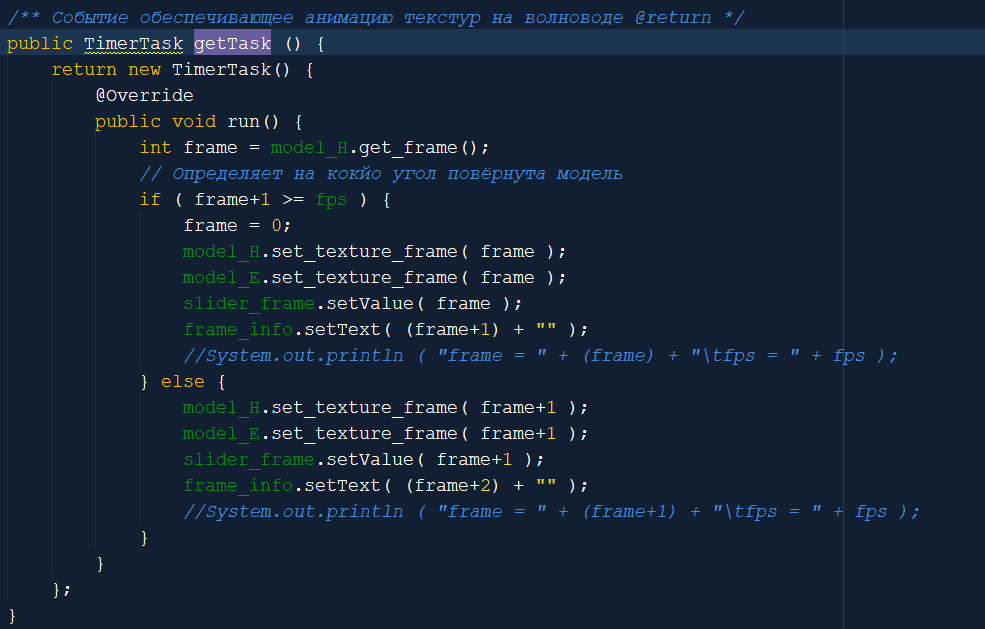
При нажатии на кнопку вычислить сначала производится проверка на наличие пустых полей ввода данных отделается вызовом метода isCorect:



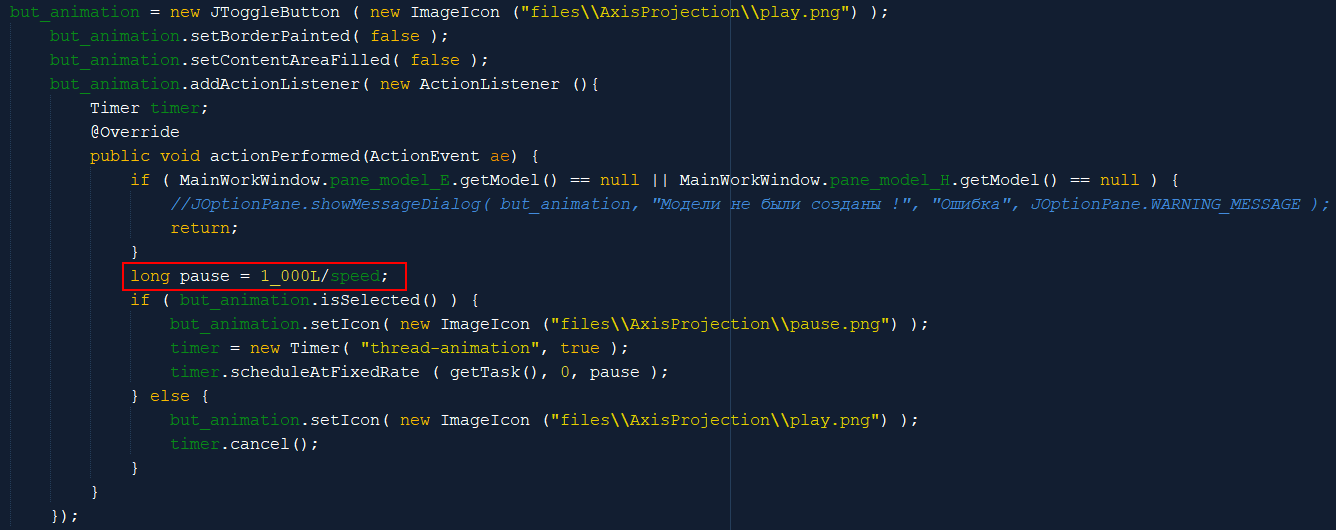
При вычислении текстур для моделей волновода текстуры вычисляются в отдельном потоке:



Анимация смены текстур осуществляется при помощи класса TimerTask. Объект данного класса создаётся в методе getTask:



Через определённый промежуток времени определяемый следующим образом:



Делать паузу меньше, чем ½ секунды не стоит, так как модель не успевает отрендериться за такое быстрое время. Решением может служить использование вычислительных возможностей графического процессора, но это сделать не совсем просто и требует отдельного изучения. В программе реализован только программный рендер.

# Класс GraphicsModel

Рассмотрим объект модели волновода. Данный класс содержит всю необходимую информацию о модели волновода, а именно: координаты вершин полигонов волновода, текстуры полигонов, матрицу трансформации и всё что связанно с аффинным преобразование координата.

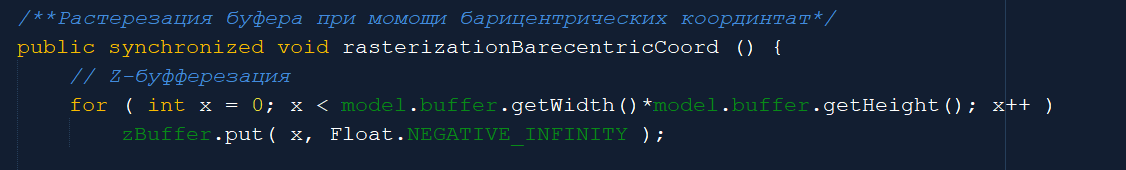
Размеры окна рендера модели:



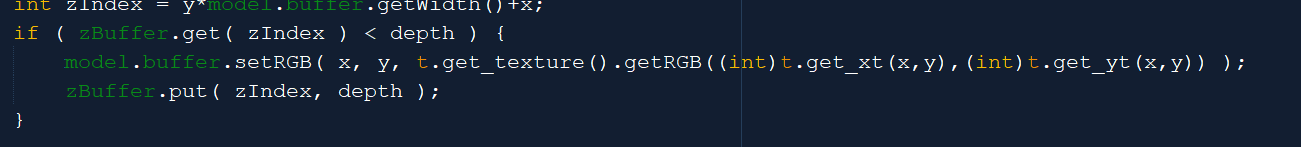
Данное поле представляет собой z-буфер. В алгоритме растеризации используется стандартная z-буферизация для правильного отображения полигонов.



В начале рендера буфер сбрасывается:



Далее буфер заполняется по мере отрисовки полигонов.

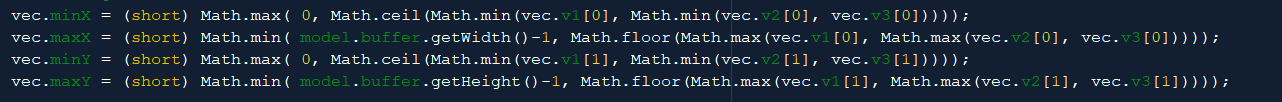


Сам алгоритм растеризации основан на алгоритме закрашивания полигоном, содержащего 3 точки (по сколько это самый быстрый и простой вариант) на основе барицентрических координата.

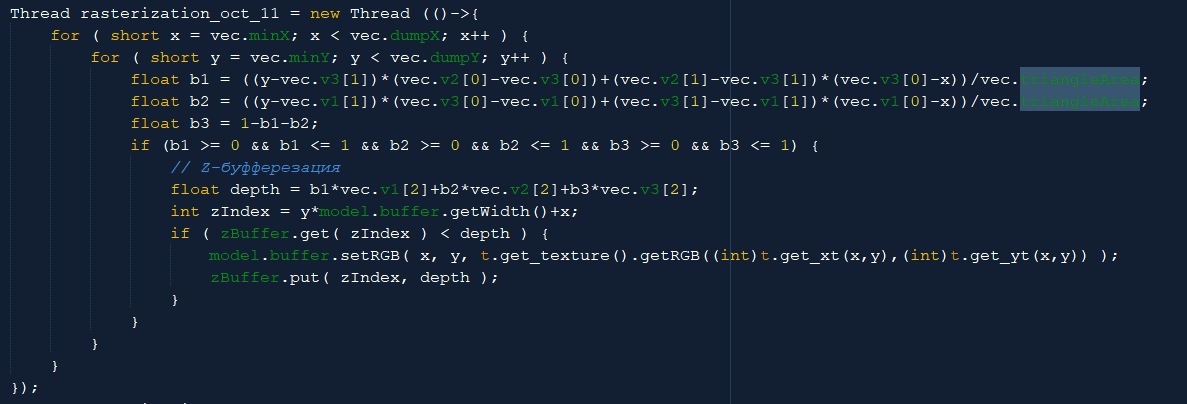
При вычислении барицентрических координата нужна площадь треугольника (полигона) она вычисляется в самом начале алгоритма растеризации.



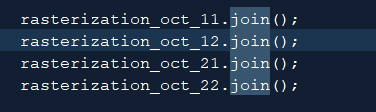
Затем выполняется поиск максимальных и минимальных координат полигона:



Теперь можно пройтись циклом от минимальный значений координат до максимальных. В программе для ускорения процесса растеризации цикл разбит на 4 части и выполняется параллельно в 4х отдельных потоках. Замеры времени выполнения показали, что это даёт довольно существенный прирост в производительности.



Остальной 3 потока подобны только диапазоны значений координат соответственно разные. Выход из метода растеризации полигона происходит только после завышения этих 4-х потоков.

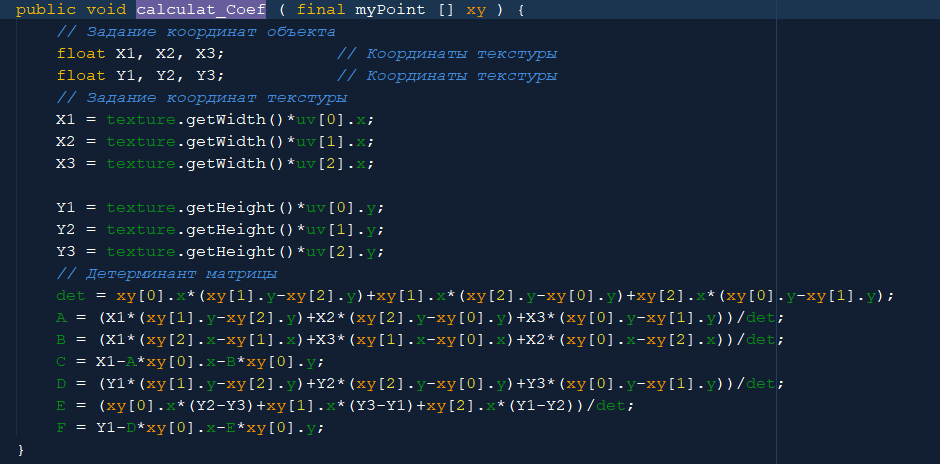


Есть ещё один нюанс. На каждый полигон в соответствии с текстурной картой накладывается текстура. Это осуществляется по средством решения системы уравнений. Так как используется простое аффинное преобразовании координат.

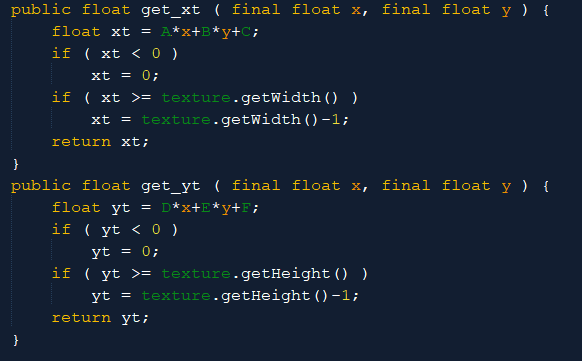
Вычисление коэффициентов уравнения:



Сам метод решения системы уравнений:



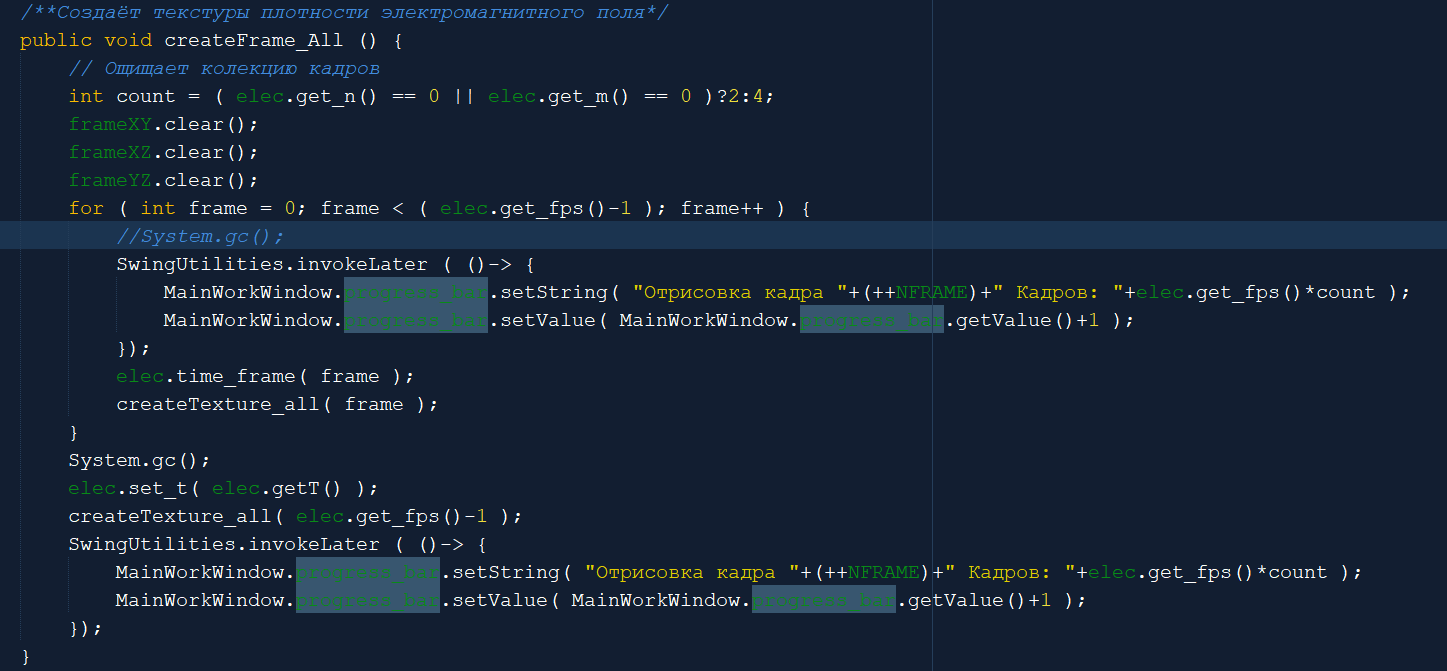
Это позволяет сопоставить координату полигона с соответствующей координатой на изображении текстуры. Затем координата текстуры получается методами get\_xt и get\_yt.



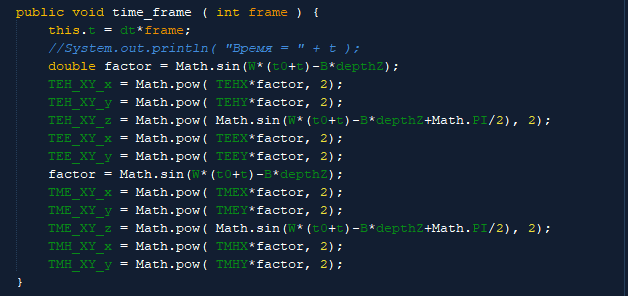
# Класс WaveLevelGraph

Данный класс вычисляет текстуры для моделей волновода. В программе рассчитываются текстуры для H волны и, если может существовать E волна, то и для неё тоже. Для уменьшения итерация циклов используется линейная интерполяция. Независимо от выбранной точности графика линий уровня (поле countX) цикл будет производить столько итерация сколько пикселей в самой текстуре (поля heightXY, widthXY, heightXZ, widthXZ, heightYZ, widthYZ).

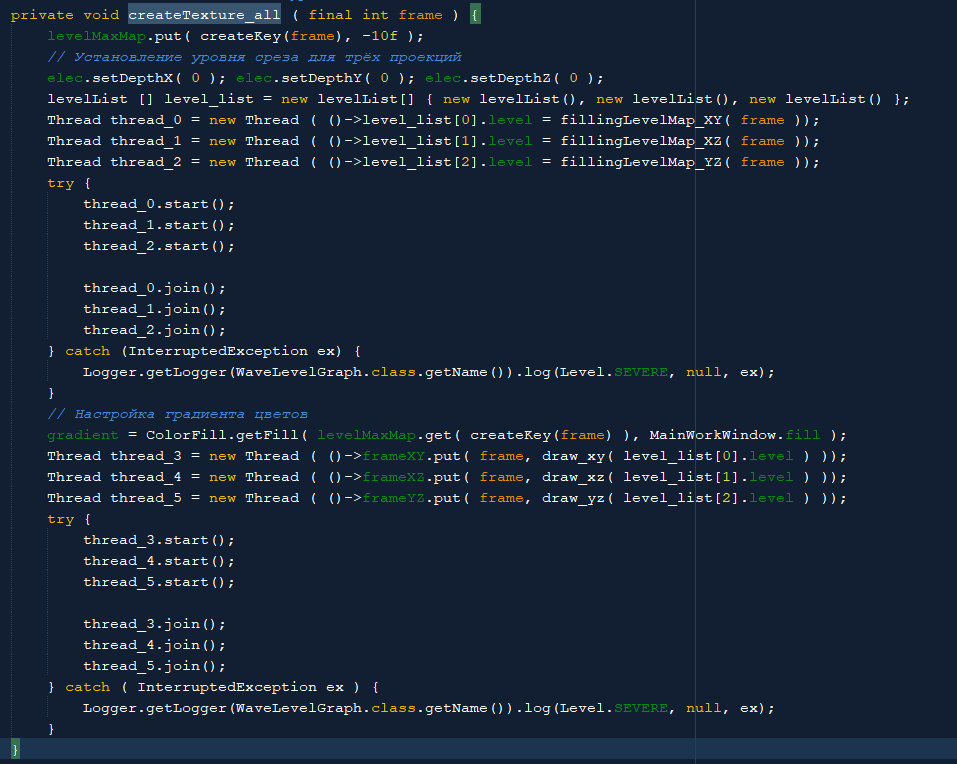
Для создания текстур для всех измерений волновода вызывается метод createFrame\_All:



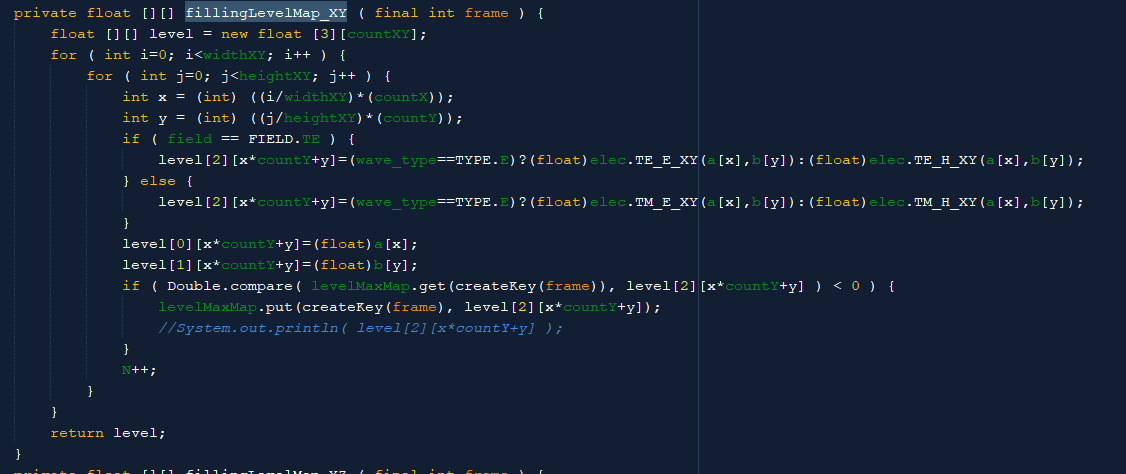
При каждой итерации происходит перережет приращений времени для графиков линий уровня. Для этого в классе электромагнитной волны ElectromagneticWave существует специальный метод time\_frame он устанавливает приращение в зависимости от значения текущего кадра.



Вычисления в данном методе так же распараллелены. Каждая итерация цикла вызывает метод createTexture\_all:



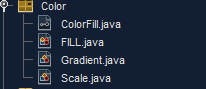
В сначала 3 параллельных потока высчитывается карта линий уровня для текущего значения прерщения времени.



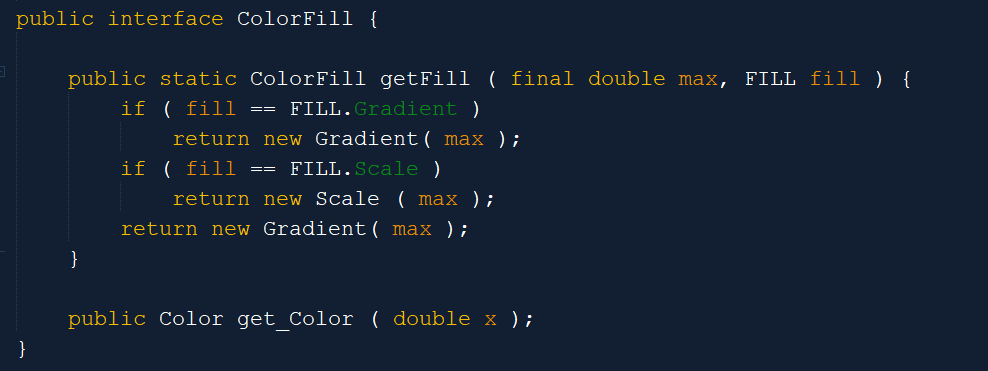
Максимальное значения уровня необходимо для закрашивания в соответствии со значением уровня. В данном классе существует потокобезопасная коллекция.



Она заполняется в том случае, если значение в данной коллекции меньше, чем, то значения, которое имеет текущий уровень. Таким образом для каждого кадра будет наедено максимальное значение линии уровня к концу заполнения карты линии уровня. После того как все карты линии уровня были заполнены необходимо настроить класс закраски текстуры.

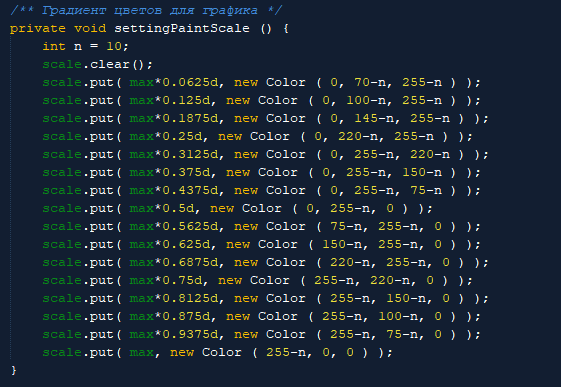


Классы выполняющие роль закрашивающих должны реализовать интерфейс ColorFill.

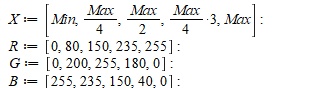


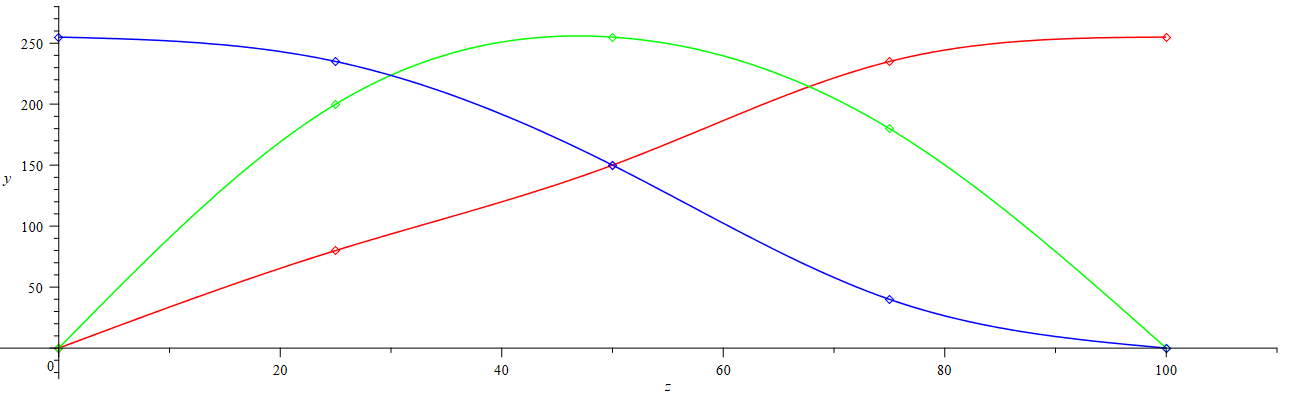
Метод get\_Color на вход получает уровень карты а возвращает соответствующий цвет.

В программе имеется два класса соответственно два способа закрашивания. Первый обычное закрашивание. Оно ступенчатое и осуществляется путём сравнения текущего значения с заданными значениями в алгоритме.

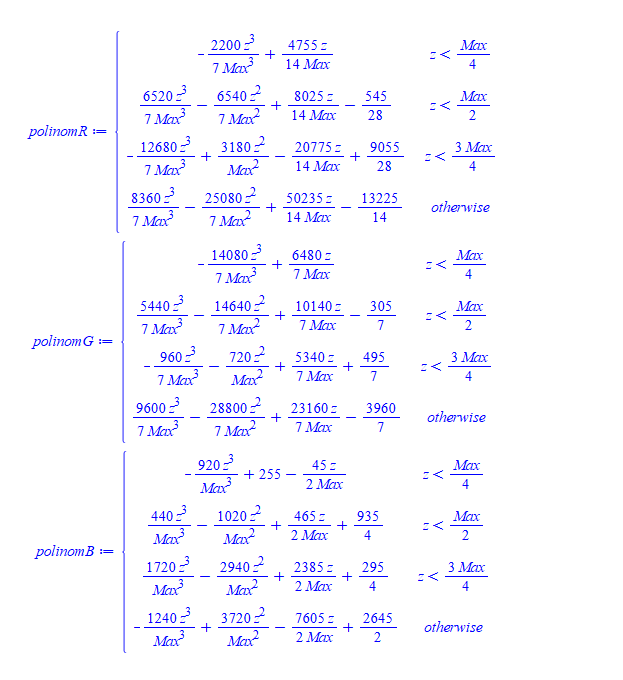


Для более плавного закрашивания необходим градиент. Для получения функции градиента была выполнена кубическая интерполяция каждого цветового канала по следующим точкам:

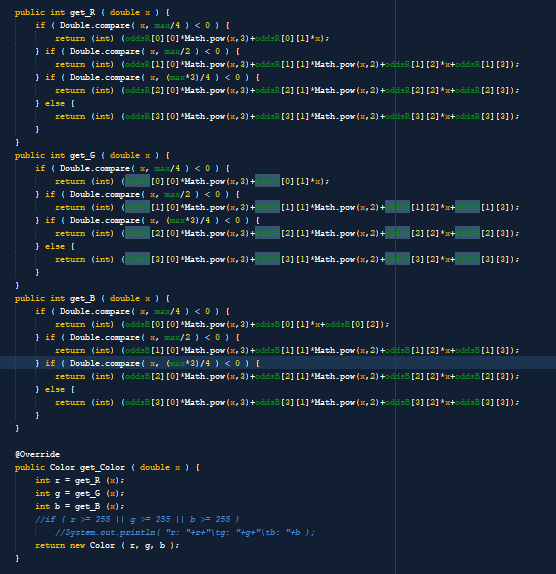




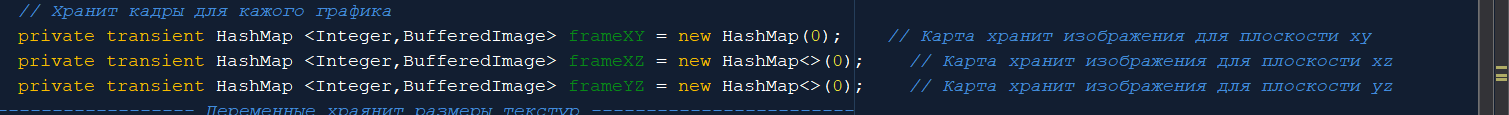
Расчёты выполнялись в программе Maple. В итоге была получена следующие функции:



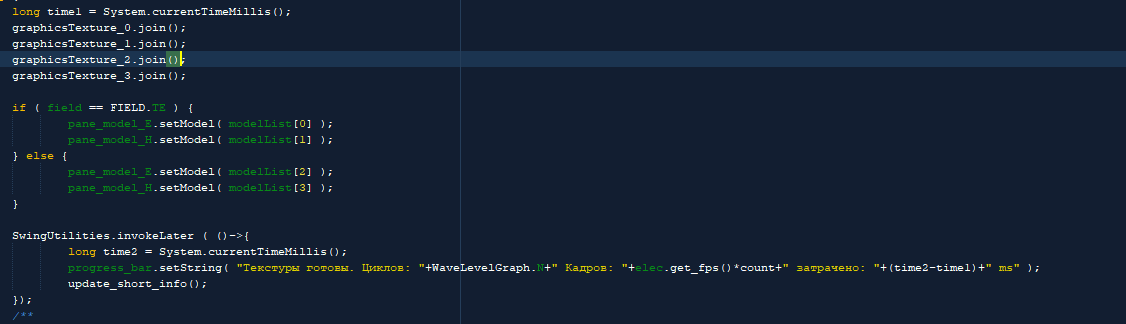
Алгоритм получения цвета:



В главном рабочем окне вызов расчёта всех текстур модели так же распараллелен. Сначала запускается новый поток, задача которого рассчитать все новые текстуры. Этот поток порождает ещё 4 потока для каждого типа волны E и H. Если хотя бы одна мода равна 0 то будет вызвано только два потока. После того как текстуры будут расчитаны, то следующие поля:

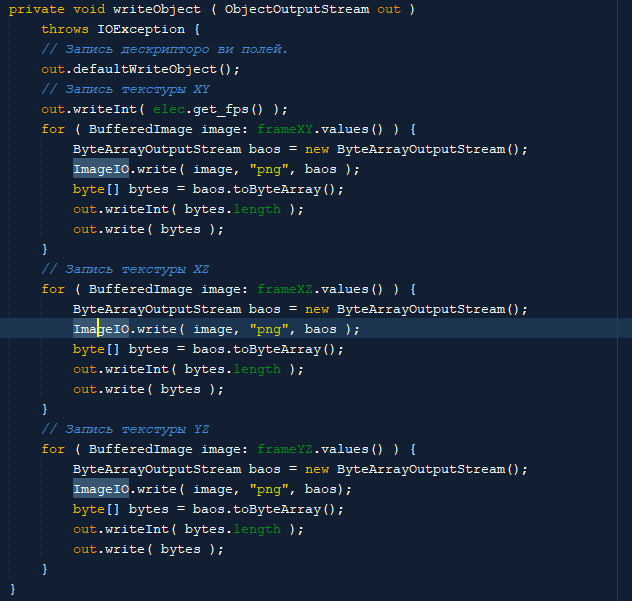


Заполнены значит можно отобразить модель волновода с наложенной текстурой.



Иногда пророс бар обновляется значительнее быстрее чем панель отображения моделей волновода. Вероятно, это связано с потоком событий в потоке диспетчеризации событий.

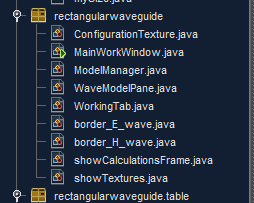
Так же имеется возможность сохранить текущую отображённую модель в файл и затем загрузить его. Это делается по средствам сериализации и десериализации. Стоит отдельно рассмотреть сериализацию класса WaveLevelGraph поскольку там имеется коллекция, содержащая BufferedImage и данный класс, не является сериализуемым по умолчанию. Решение данной проломы на самому деле довольно тривиальное. Нужно вмешаться в процесс сериализации и записать изображения по своими. Описывать подробней не буду так как это уже касается механизма сериализации а не описания самой программы:



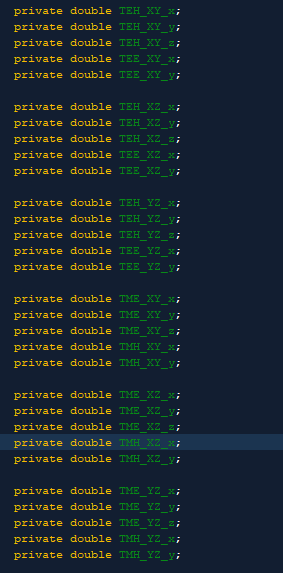


# Описание остальных классов

Классы в данном пакете относится к отображению графического интерфейса для различных задач. Особого смысла описывать их работу нет.



Класс ElectromagneticWave является абстрактной моделью электромагнитной волны в прямоугольном волноводе. При расчётах используются стандартные формулы их описывать нет смысла. Имеются следующие поля:



Задача этих полей заключается только в том, чтобы уменьшить число умножения на каждой итерации расчётов. Тоесть это просо множители, которые можно рассчитать заранее и которые на зависят от координат или от времени.