|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА  Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра математического обеспечения и стандартизации ИТ**

**ОТЧЁТ ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ №13**

**по дисциплине**

**«Разработка мобильных приложений»**

|  |  |  |  |
| --- | --- | --- | --- |
| Отчет представлен к  рассмотрению:  Студенты группы ИНБО-04-20 | « » февраля 2022 г. | (подпись) | Ло В.Х. |
|  |  |  |  |
| Преподаватель | « » 2022 г. | (подпись) | Фандеев И.И. |

Москва, 2022г.

**СОДЕРЖАНИЕ**

[ОТЧЕТ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКОЙ РАБОТЫ. 3](#_Toc104154487)

[1. Создание среды OpenGL ES 3](#_Toc104154488)

[1.1 Реализовать объявление использования OpenGL ES в манифесте 3](#_Toc104154489)

[1.2 Реализовать создание явления для графики OpenGL ES 3](#_Toc104154490)

[2.1 Реализовать описание треугольника 5](#_Toc104154491)

[2.2 Реализовать описание квадрата 6](#_Toc104154492)

[3. Рисование фигур 7](#_Toc104154493)

[3.1 Реализовать инициализацию фигур 7](#_Toc104154494)

[3.2 Реализовать рисование фигуры 10](#_Toc104154495)

[4. Применение проекций и камер 10](#_Toc104154496)

[4.1 Реализовать описание проекции 11](#_Toc104154497)

[4.2 Реализовать описание камеры 12](#_Toc104154498)

[4.3 Реализовать применение проекции и камеры 12](#_Toc104154499)

[5. Добавление движения 12](#_Toc104154502)

[5.1 Реализовать вращение фигур 12](#_Toc104154503)

[5.2 Обеспечить разрешение постоянной отрисовки 13](#_Toc104154504)

[6. Обработка сенсорных событий 13](#_Toc104154507)

[6.1 Обеспечить отслеживание сенсорных событий 13](#_Toc104154508)

[6.2 Обеспечить выставление угла поворота 14](#_Toc104154509)

[6.3 Реализовать применение вращения 14](#_Toc104154512)

[ВЫВОД 16](#_Toc104154513)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc104154514)

# ОТЧЕТ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКОЙ РАБОТЫ.

**1.** **Создание среды OpenGL ES**

Для того, чтобы рисовать графику с помощью OpenGL в Android приложениях, необходимо создать для этого контейнер просмотра.

Один из прямолинейных способов, это использовать объекты GLSurfaceView и GLSurfaceView.Renderer. GLSurfaceView это контейнер для просмотра графики, нарисованной с помощью OpenGL, а GLSurfaceView.Renderer управляет тем, что будет нарисовано в этом контейнере.

**1.1 Реализовать объявление использования OpenGL ES в манифесте**

Для использования OpenGL ES 2.0 в приложении, вы должны добавить следующие строки в файл манифеста:



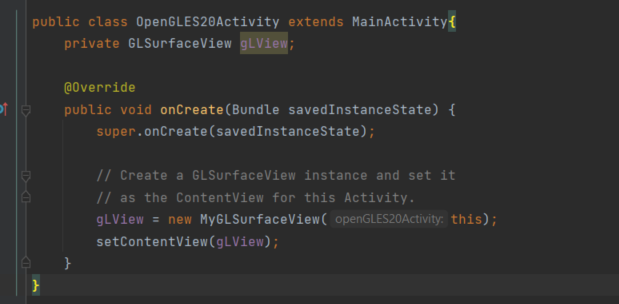
**Рисунок 1 – OpenGL ES 2.0 в приложении**

**1.2 Реализовать создание явления для графики OpenGL ES**

Приложения, использующие OpenGL ES содержат явления, подобно любым другим приложениям с пользовательским интерфейсом.

Главное отличие от других приложений в том, какие компоненты будут находиться в разметке явления.

В то время как в обычных приложениях используются кнопки, поля ввода и списки, в приложениях с OpenGL ES вы можете добавить также компонент GLSurfaceView.

****

**Рисунок 2 – создание явления для графики OpenGL ES**

**1.3 Реализовать создание объекта GLSurfaceView**

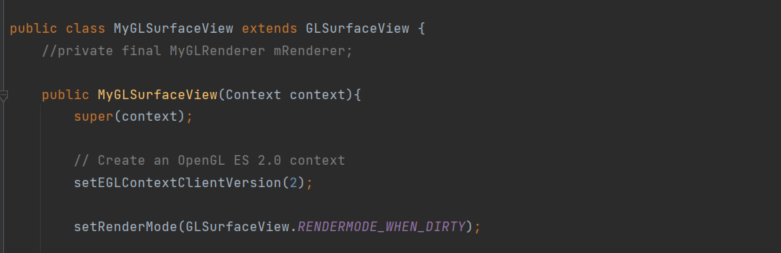
GLSurfaceView это специальный контейнер, в котором можно рисовать графику с использование OpenGL ES.

Но его недостаточно. В действительности рисованием объектов управляет объект GLSurfaceView.Renderer.

Фактически, для этого объекта так мало кода, что вы возможно захотите пропустить его описание и просто использовать немодифицированный объект GLSurfaceView, но не делайте этого.

Если вы используете OpenGL ES 2.0, вы должны добавить другой вызов конструктора GLSurfaceView, показывающий, что вы хотите использовать API версии 2.0:

При использовании дополнительной опции GLSurfaceView.RENDERMODE\_WHEN\_DIRTY для GLSurfaceView, прорисовка в контейнере происходит только при изменении данных:

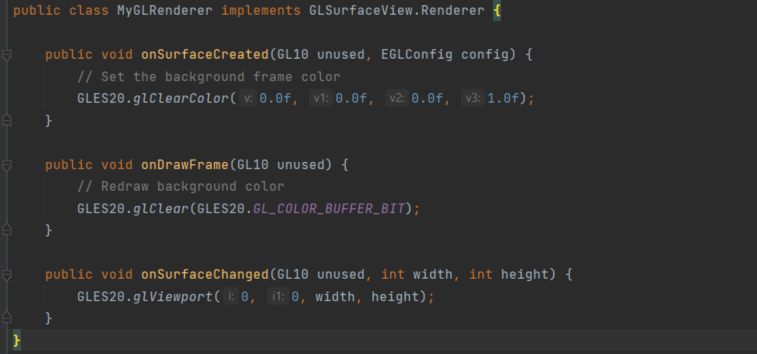


**Рисунок 3 – создание объекта GLSurfaceView**

**1.4 Реализовать создание класса**

Класс управляет рисованием в контейнере GLSurfaceView, с которым он ассоциирован. Есть три метода, которые вызывает Android, чтобы понять что и как нарисовать в контейнере GLSurfaceView:

* onSurfaceCreated() – вызывает однократно, чтобы задать среду выполнения OpenGL ES.
* onDrawFrame() – вызывается при каждой перерисовке контейнера.
* onSurfaceChanged() – вызывается при изменении геометрии контейнера, например при изменении ориентации экрана



**Рисунок 4 – создание класса**

#### **2 Описание фигур**

Описание фигур, который будут нарисованы в контейнере OpenGL ES, это первый шаг к созданию вашего высококлассного шедевра. Рисование в OpenGL ES может быть немного сложным, если не знать базовых вещей об описании фигур.

В уроке рассматривается система относительных координат OpenGL ES, описание базовых фигур, лицевая сторона фигур, а также определение треугольника и квадрата.

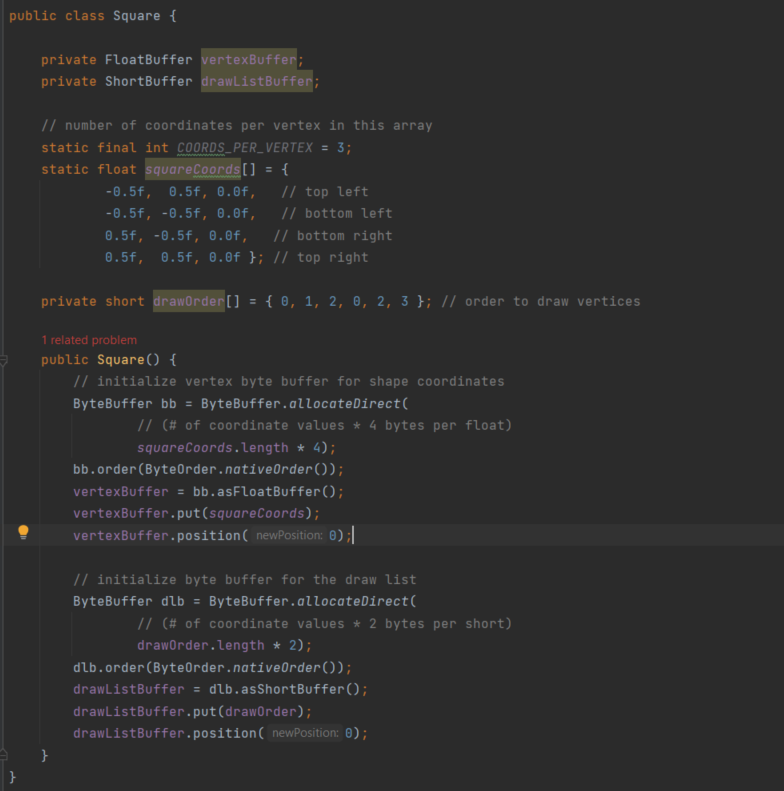
2.1 **Реализовать описание треугольника**

OpenGL ES позволяет описывать фигуры, используя координатную систему в трех плоскостях. Прежде чем нарисовать треугольник, необходимо описать его координаты. Типичным способ сделать это в OpenGL является описание массива вершин – чисел с плавающей точкой, задающих точки на плоскости. Для максимальной эффективности используйте объект типа ByteBuffer, который передается в графический контейнер OpenGL для обработки.



**Рисунок 5 – описание треугольника**

2.2 **Реализовать описание квадрата**

****

**Рисунок 6 – Метод onPostExecute() в классе BitmapWorkerTask**

3. **Рисование фигур**

Рисование фигур с помощью OpenGL требует немного больше кода, чем вы могли себе вообразить, поскольку OpenGL предоставляет широкие возможности управления при рисовании внутри контейнера.

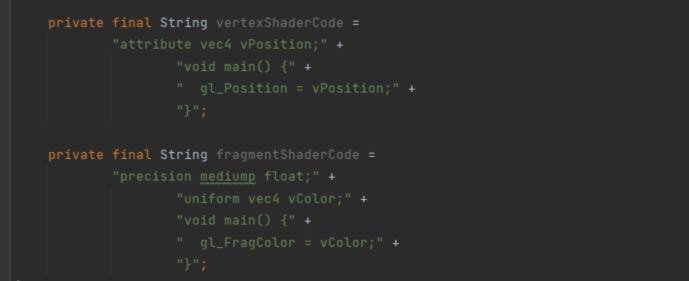
Здесь мы покажем как нарисовать фигуры, описанные в предыдущем материале, используя API OpenGL ES 2.0.

3.1 **Реализовать инициализацию фигур**

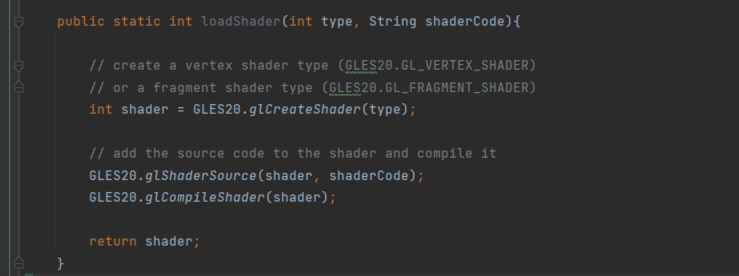
Рисование описанных фигур с помощью OpenGL требует значительного количества кода, поскольку нужно предоставить много информации о деталях визуализации.

Конкретно, должны описать следующее:

* Шейдер вершин – графический код OpenGL для визуализации вершин фигуры.
* Шейдер фрагмента – код OpenGL для визуализации лицевой стороны фигуры с цветами и текстурами.
* Программа – объект OpenGL, который содержит шейдеры, которые вы хотите использовать для рисования одной или нескольких фигур.



**Рисунок 7 – описани**е **основных шейдеров**

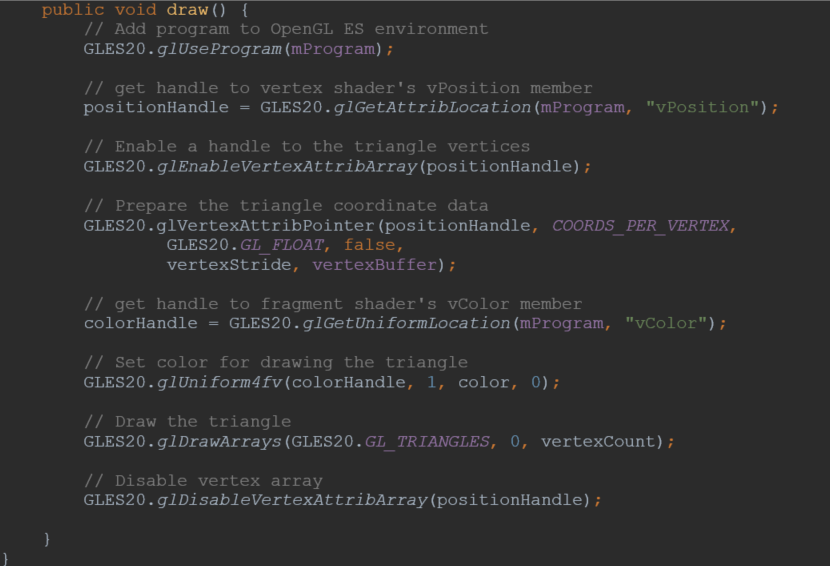


**Рисунок 8 – метод в классе renderer**



**Рисунок 9 –**  структура одноразового кода

Когда у вас есть весь этот код, для рисования этого объекта просто требуется вызов метода draw() из метода onDrawFrame() вашего рендерера:



**Рисунок 10 – код устанавливает позицию и цвет шейдеров вершин и фрагментов для фигуры**

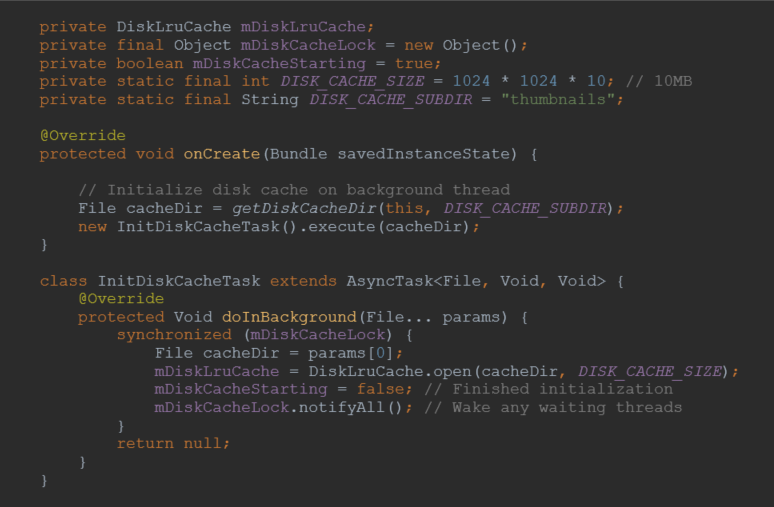
3.2 **Реализовать рисование фигуры**

Кэш в памяти полезен для мгновенного доступа к недавно просмотренным изображениям, однако вы не можете полагаться на то, что изображения могут быть доступны в кэше.

Компоненты вроде GridView с большим набором данных обычно забивают кэш-память.

Выполнение ваше приложения может быть прервано, например входящим звонком, и пока оно остановлено, кэш-память может быть уничтожена.

Дисковый кэш может использоваться в данном случае для хранения обработанных изображений и помочь уменьшить время загрузки изображений, которые уже недоступны в кэш-памяти.



**Рисунок 11 – Дополнение к существующей кэш-памяти**

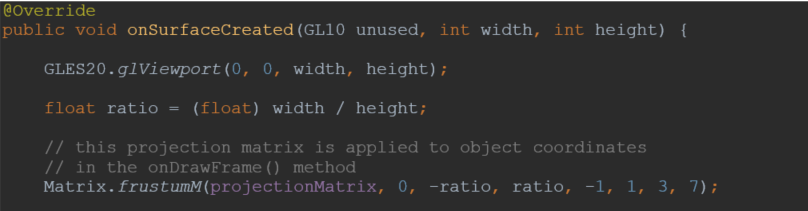
**4. Применение проекций и камер**

Эта симуляция физического зрения строится на математических методах преобразования координат объекта:

* Проекция – это преобразование регулирует координаты объекта, основываясь на ширине и высоте контейнера GLSurfaceView, в котором они нарисованы. Без этой трансформации объекты сжимаются и выглядят непропорциональными в окне просмотра. Преобразование проекции применятеся только при установке и изменении пропорций контейнера в методе onSurfaceChanged(). За подробной информацией о проекции и подсчете координат, обращайтесь в раздел Подсчет координат для графических объектов.
* Камеры – это преобразование регулирует координаты, основываясь на позиции виртуальной камеры. Важно помнить, что OpenGL не описывает фактический объект камеры, но предоставляет методы для симуляции камеры и преобразования объектов на экране. Преобразование с помощью камеры может быть рассчитано однократно при добавлении объекта GLSurfaceView, или многократно, динамично меняя картинку в ответ на действия пользователя.

4.1 **Реализовать описание проекции**

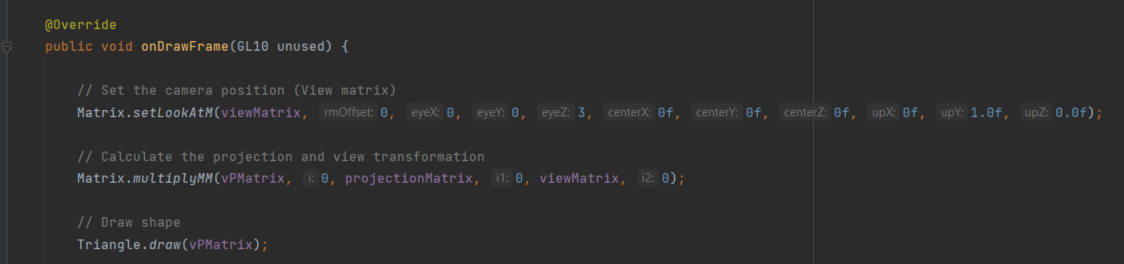
Данные для проекции рассчитываются в методе onSurfaceChanged() класса GLSurfaceView.Renderer. В следующем примере мы берем высоту и ширину контейнера GLSurfaceView и используем их для заполнения матрицы преобразования проекции – объекта типа Matrix в методе Matrix.frustumM():



**Рисунок 12 – описание проекции**

4.2 **Реализовать описание камеры**

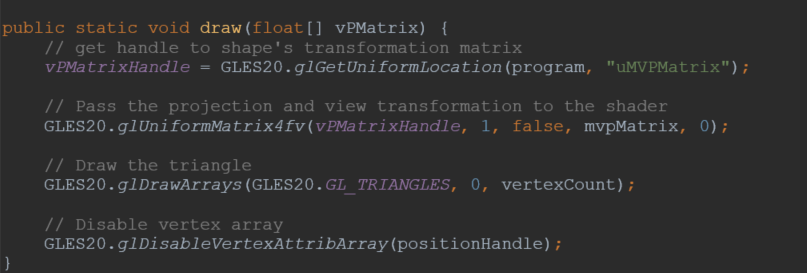
Камера рассчитывается с помощью метода Matrix.setLookAtM(), а затем сочетается с расчетами проекции из предыдущего раздела.



**Рисунок 14 – описание камеры**

**4.3 Реализовать применение проекции и камеры**

Чтобы использовать комбинированные преобразования проекции и камеры из предыдущего раздела урока, измените метод draw() вашего графического объекта для применения их к фигуре:

****

**Рисунок 15 – применение проекции и камеры**

**5. Добавление движения**

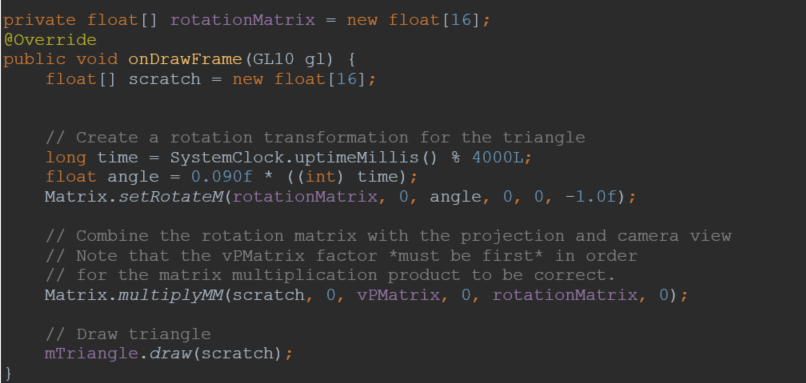
Рисование объектов на экране, это основная функция OpenGL, но тем же успехом вы могли бы использовать и другие графические классы Android, вроде Canvas или Drawable.

OpenGL ES предоставляет дополнительные возможности для движения и трансформации объектов в трех измерениях, а также другие уникальные инструменты для создания неповторимого пользовательского интерфейса.

5.1 Реализовать вращение фигур

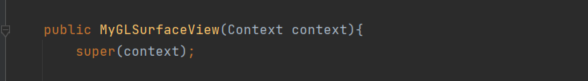
Вращать нарисованные фигур в OpenGL ES 2.0 относительно просто.

Создать еще одну матрицу (матрицу вращения) и комбинируете ее с матрицами проекции и камеры:



**Рисунок 16 –** вращение фигур

5.2 Обеспечить разрешение постоянной отрисовки

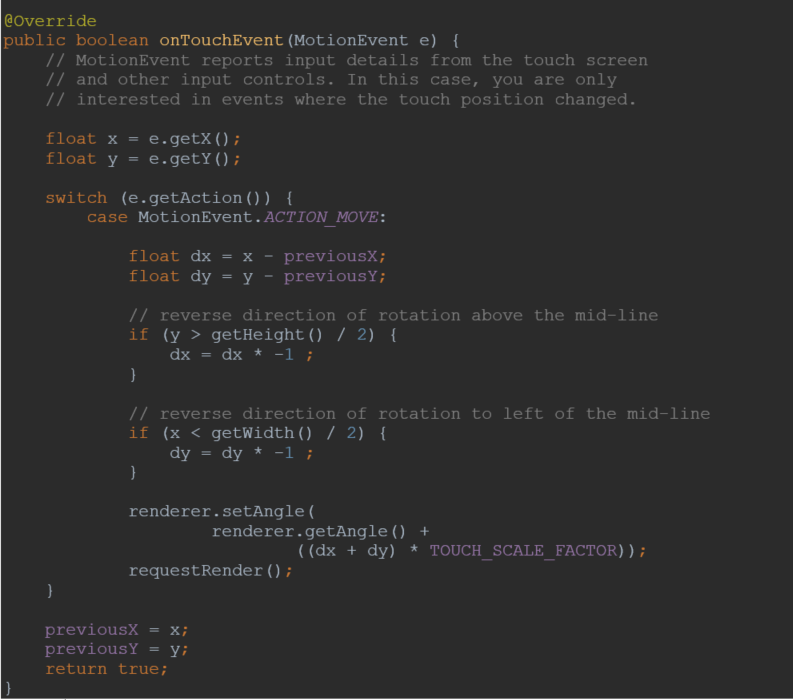


**Рисунок 17 –** разрешение постоянной отрисовки

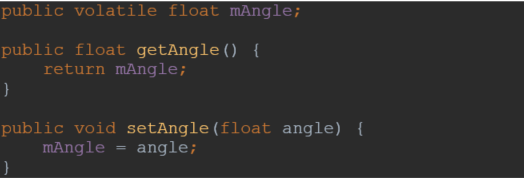
**6. Обработка сенсорных событий**

**6.1 Обеспечить отслеживание сенсорных событий**

Чтобы сделать OpenGL приложение отзывчивым на события, необходимо добавить метод onTouchEvent() в ваш класс-наследник GLSurfaceView.

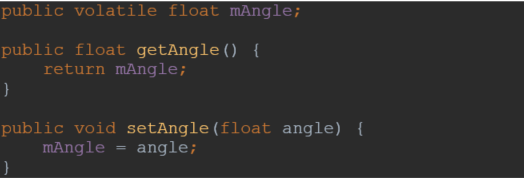


**Рисунок 18 – метод onTouchEvent()**



**Рисунок 19 – метода setRenderMode()**

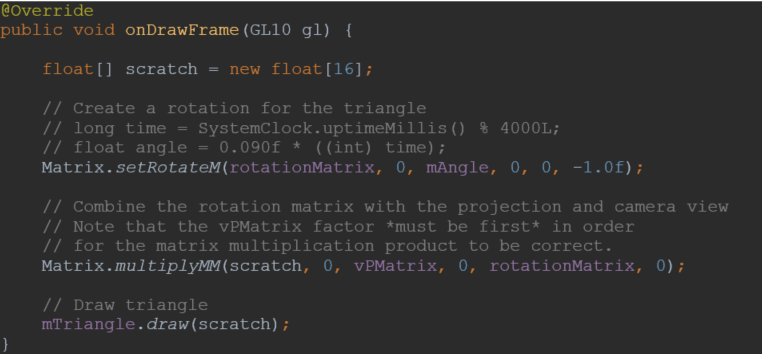
**6.2 Обеспечить выставление угла поворота**



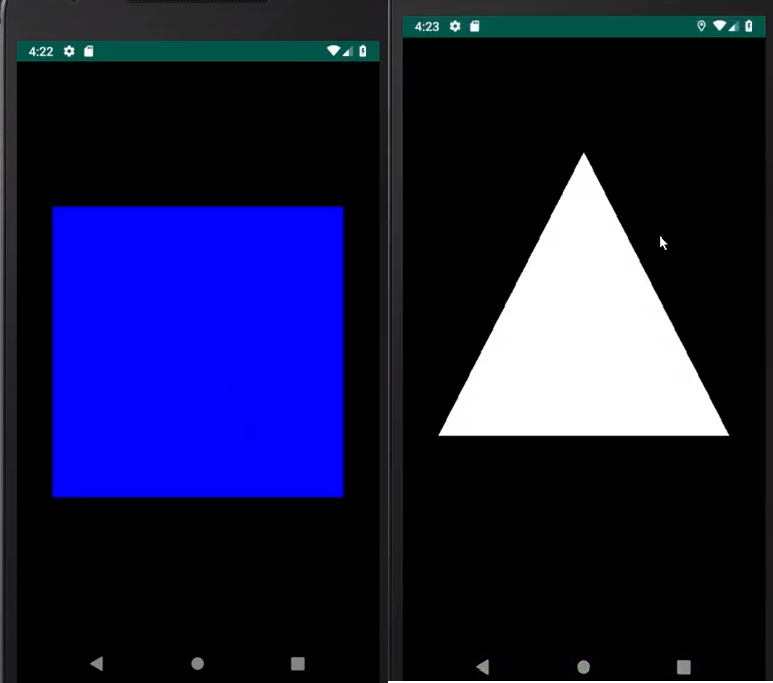
**Рисунок 20 – выставление угла поворота**

**6.3 Реализовать применение вращения**

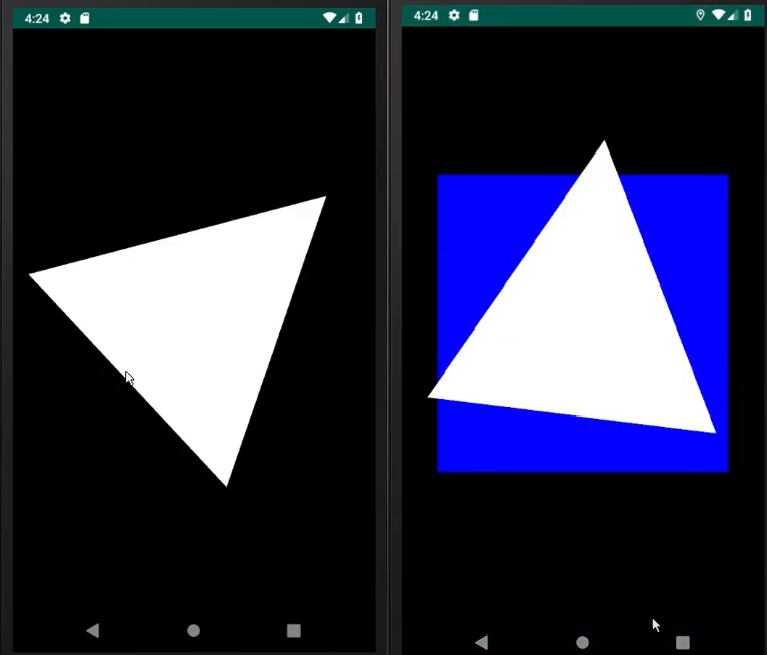
Чтобы применить вращение, сгенерированное сенсорным событием, закомментируйте код, который генерирует угол и добавьте переменную mAngle, которая хранит угол, заданный сенсорным событием:



**Рисунок 21 – применение вращения**



**Рисунок 22 – Результат 1**



**Рисунок 23 – Результат 2**

# ВЫВОД

В ходе выполнения практической работы были изучены:

* Методические указания Графика и анимация

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Лекции по дисциплине «Разработка мобильных приложений» / И. В. Синицын, МИРЭА — Российский технологический университет, 2022.