УчИМИ Ученически институт по математика и информатика

СЕДЕМНАДЕСЕТА УЧЕНИЧЕСКА СЕКЦИЯ
УС'17

ТЕМА НА ПРОЕКТА

3D designer –

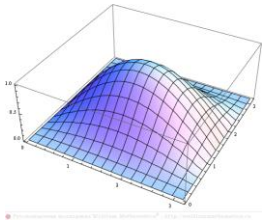
мобилно приложение за проектиране на ефекти

Автор:

Найден Костов Найденов
СУ „Васил Левски“, гр. Вълчи дол, обл. Варна, X клас

Научен ръководител (консултант):

НЯМА



3D графиките са изображения, с които може да се работи в 3 измерения, въпреки че се извеждат на двумерен екран. Разглеждащият може да се "движи" около изображението и да променя гледната си точка, както при реални пространствени обекти. Приложенията за тримерна графика могат автоматично да променят светлосенки в съответствие със зададени светлинни източници и материали.

Мобилното приложение **3D designer** е за всеки, които иска сам да проектира изображение с абстрактни художествени форми. Чрез взаимодействието между естетически и математически му познания на потребителя приложението може да послужи за насърчаване на творческа изява, за развитие на необхватни възможности за развитие на възприятието, въображението и асоциативното мислене на човека.

I. Описание на приложната област

Мобилните приложения са бъдещето и заемат все по-сериозно присъствие в ежедневието ни. Разчитаме на тях, за да ни е по-удобно когато търсим, споделяме и запазваме информация.Те са едно от най-великите изобретения на 21-ви век. През последните няколко години смартфоните станаха неразделна част от нашето ежедневие. Ние получаваме достъп до цялото знание, познато на човечеството и публикувано в Интернет от смартфона си, а не от лаптопа или настолния компютър. И това е така, защото телефона е само на една ръка разстояние и е винаги с нас. Ето защо телефонът и таблетът, днес, не са просто технологични играчки, а мултифункционални помощници в ежедневието ни. Затова за повишаване на функционалността им възниква необходимост от разработката все повече и все по-разнообразни мобилни приложения.

II. Описание на приложната програма

В началото при стартиране на приложение **3D designer** се зареждат всички текстури и 3D модели нужни за приложението. След това се рисуват двуизмерните бутони над екрана. Сцената се рендерира на Framebuffer Object. Ако е поставен нов обект той се рендерира на сцената с различни координати, ротация и цвят.

Структурата на приложението се състои от шест класа:

Object – клас за всички обекти, позицията им и текстурата. Разделя се на две части – за двуизмерно рисуване на бутони, панели и др. и рисуване на текст.

Text – клас за рисуване на текста на екрана

Renderer – клас за определяне на начина и позицията на обектите и моделите

Splash screen – клас за зареждане на ресурси

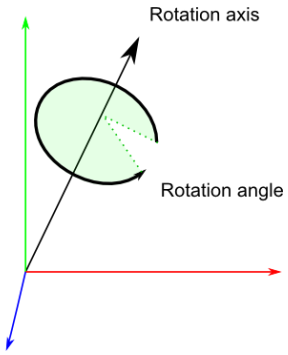
Model – клас за рисуване на триизмерни модели

Loader – клас за четене на XML файлове

За реализацията на tracball ротацията се използват кватерниони като линейна комбинация във формата q=a+bi+cj+dk. Изразът bi + cj + dk е векторната част на кватерниона, а "a" е скаларната част.

$i^2 = j^2 = k^2 = ijk = -1$, ij=k, jk=i, ki=j

Първо се вземат координатите на началната точка и се слагат върху сфера. След това се взема текущата точка и се изчислява ъгъла на преместване и посоката. Кватернионът се използва за изчисляване на ротацията.



Кватернионът има важното свойство да се превръща в матрица.

Използва се като ротационна матрица и се умножава с матрицата на камерата.

Реализацията на това преобразуване се осъществява с кода:

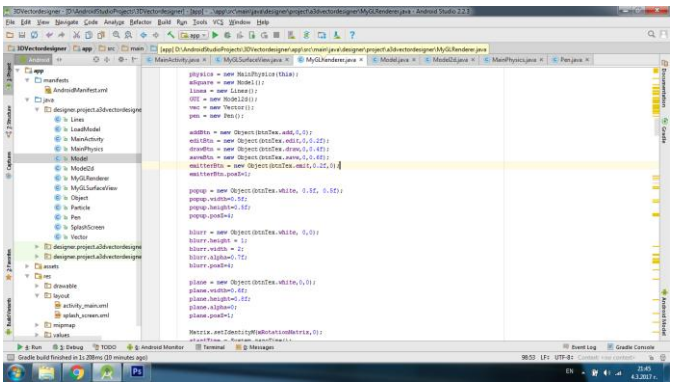
```
public float[] sphereCords(float tempX, float tempY){
    float d, a;
    float[] vert=new float[3];
    vert[0] = (2*tempX - wid) / wid;
    vert[1] = (hei - 2*tempY) / hei;
    d = (float)Math.sqrt(vert[0]*vert[0] + vert[1]*vert[1]);
    vert[2] = (float)Math.cos((Math.PI/2.0) * ((d < 1.0) ? d : 1.0));
    a = 1 / (float)Math.sqrt(vert[0]*vert[0] + vert[1]*vert[1] +
vert[2]*vert[2]);
    vert[0] *= a; vert[1] *= a; vert[2] *= a;

    return vert;
}

public float[] rotateQuat(){

    float[] v1 = sphereCords(realX, realY);
    float[] v0 = lastAngl;
    lastX = realX;
    lastY = realY;

    v1 = physics.normalizeVector3(v1);
    v0 = physics.normalizeVector3(v0);
    System.arraycopy(v1, 0, lastAngl, 0, 3);

    float cosTheta = physics.dotProduct(v0, v1);

    float[] rotAxis = physics.crossProduct(v0, v1);

    float s = (float)Math.sqrt( (1+cosTheta)*2 );
    float invs = 1 / s;
    float
angl=(float)Math.acos(cosTheta)/(physics.length(v0)*physics.length(v1))
;
    // Converts all degrees angles to radians.

    float[] quat = new float[4];

    quat[0] = s/2;
    quat[1] = rotAxis[0]*invs;
    quat[2] = rotAxis[1]*invs;
    quat[3] = rotAxis[2]*invs;
```

```
        physics.x=quat[1];
        physics.y=quat[2];
        physics.z=quat[3];


        float[] matrix = new float[16];
        matrix[0] = 1 - 2 * (quat[2] * quat[2] + quat[3] * quat[3]);
        matrix[1] = 2 * (quat[1] * quat[2] + quat[3] * quat[0]);
        matrix[2] = 2 * (quat[1] * quat[3] - quat[2] * quat[0]);
        matrix[3] = 0;

        // Second Column
        matrix[4] = 2 * (quat[1] * quat[2] - quat[3] * quat[0]);
        matrix[5] = 1 - 2 * (quat[1] * quat[1] + quat[3] * quat[3]);
        matrix[6] = 2 * (quat[3] * quat[2] + quat[1] * quat[0]);
        matrix[7] = 0;

        // Third Column
        matrix[8] = 2 * (quat[1] * quat[3] + quat[2] * quat[0]);
        matrix[9] = 2 * (quat[2] * quat[3] - quat[1] * quat[0]);
        matrix[10] = 1 - 2 * (quat[1] * quat[1] + quat[2] * quat[2]);
        matrix[11] = 0;

        // Fourth Column
        matrix[12] = 0;
        matrix[13] = 0;
        matrix[14] = 0;
        matrix[15] = 1;
        return matrix;
}
```

III. Избор на програмно-технически средства

Мобилното приложение **3D designer** е написано на **Java -** обектно-ориентиран език, при който основна концепция са класовете, които се дефинират чрез свойствата и поведението на обектите. Езикът Java се разпространява заедно със специална среда, върху която се изпълнява, наречена Java Runtime Environment (JRE). Тя включва т. нар. Java виртуална машина (JVM) и пакет стандартни библиотеки, предоставящи базова функционалност.

Интегрираната среда за разработка е **Android Studio 2.2.3**, който е софтуер, даващ възможност едно и също приложение да се разработва паралелно за няколко вида устройства: Android смартфони, таблети, базирана на Android Wear електроника, Android TV, Android Auto системи и Google Glass.



По време на инсталацията си, Android студиото автоматично слага и настроен и оптимизиран емулатор с готови presets за различни устройства. Емулаторът дава възможност бързо да се провери как точно ще изглежда и действа разработваното приложение на конкретно устройство при зададен размер и резолюция на екрана.

Android Studio 2.2.3 също така позволява удобно компилиране на APK файлове с различна функционалност от един проект, така че лесно могат да се създават и няколко версии на приложението с различни възможности. Освен това в IDE средата има и готови “чернови” за различни видове приложения и интеграция с GitHub.

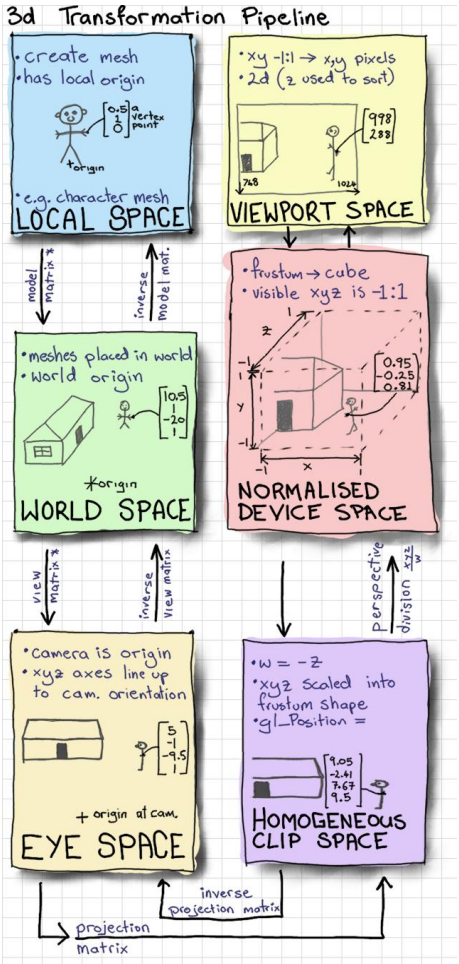
Мобилното приложение **3D designer** използва библиотеката **OpenGL** ES, версия 2.0.

OpenGL ES (Open Graphics Library) е мощен приложно-програмен интерфейс (API) за реализиране на лесно преносими графични приложения. **OpenGL** е един от най-популярните програмни интерфейси за реализиране на 2D и 3D графика. За това допринасят широката му достъпност, съвместимостта му с различни операционни системи и с различни компютърни платформи. Тази библиотека е подходяща за приложения изискващи високо качество на изображението, комбинирано с добра производителност, за да бъде възможно генерирането му в реално време.

Приложенията, създадени с **OpenGL**, могат да се изпълняват на компютърни платформи с разнообразни функции и възможности. В резултат на това разработчиците на софтуер могат да бъдат сигурни, че техният продукт ще бъде широко достъпен. **OpenGL** е добре структуриран, има интуитивен дизайн и логични команди. Това обикновено води до по-малък размер на кода на програмите, които използват този стандарт. Освен това библиотеката напълно капсулира информацията за графичния хардуер, така че приложението не трябва да се притеснява за специфичните му особености.

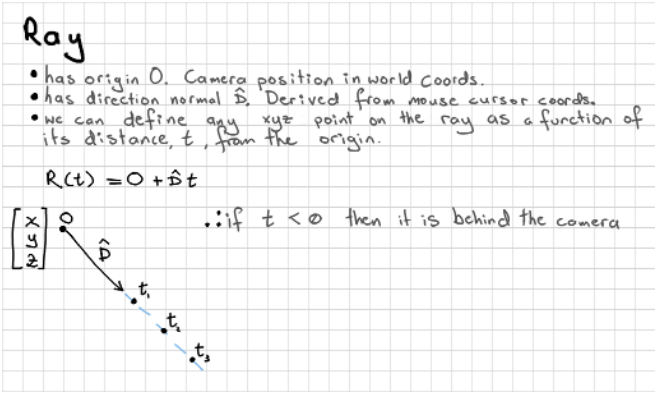
IV. Описание на реализацията

1. Архитектура на системата



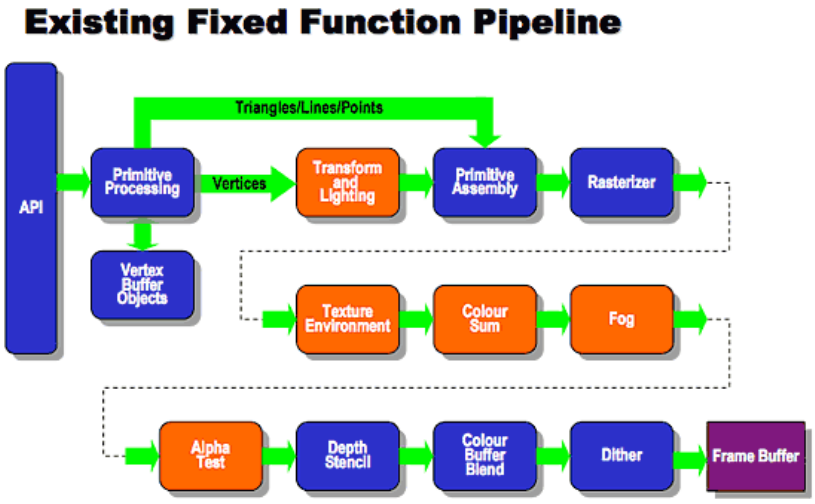
Мобилното приложение **3D designer** има Graphics Engine направен с помощта на OpenGL и Physics Engine, който проверява кой предмет е докоснат. В началото на приложението се зареждат всички нужни обекти и текстури в GPU, за да могат да се използват по-късно. Обектите се зареждат с помощта на мой XML Parser, който чете Collada файлове. След като се заредят следващото им рисуване ще бъде много по-бързо. Използват се Low Poly модели за минимизиране на паметта и рисуваните триъгълници на сцената. По този начин успявам да задържа стабилен fps от 60 frames. Зареждането на модели и чертаенето на стени от менюто от лявата страна на екрана. В приложението се използва техниката Mouse Picking с Ray tracing за определяне на мястото на докосване от двуизмерния екран в 3D пространството. Прави се лъч, който започва от центъра на камерата и завършва в точка най-далечната в зададеното пространство. След това трябва да се провери къде се засичат лъча и равнината, където се прави сградата. OpenGL представя 3D света в 2D екран с помощта на две матрици, които са като камера. За да се направи лъч от 2D пространството към

3D трябва да се обърне процеса, по който OpenGL представя света. Първо се “нормализират” координатите, защото в OpenGL те са от 0 до 1. След това се умножава по първата обърната матрица (projection matrix) и след това по втората (View matrix). Вече докосването е в World space. Лъчът се умножава по една променлива t, докато удари нещо.



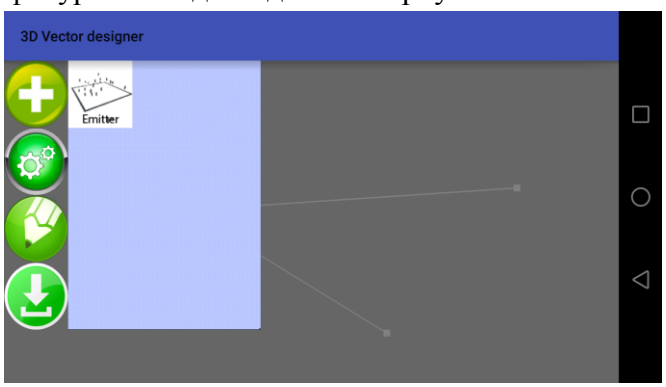
Всичко това се рендерира на Framebuffer Object, на който се рисува сцената. Framebuffer съдържа:

- byte array изображение или текстура, която после става screenshot,
- Renderbuffer – там се добавя дълбочината на обекта.



2. Интерфейс

Интерфейсът на програмата е лесен за използване. В лявата част на екрана е разположено меню. То се състои от четири бутона. Първият бутон е за добавяне на нови обекти. Когато се натисне се отваря панел с други бутони – emitter за генериране на частици. Вторият бутон е settings, при който могат да се променят следните настройки: размер на частичките, брой на частици в секунда, липса или наличие на текстура. Третият бутон е за изчертаване на линия с отсечки. След това от бутон settings може да се избере каква геометрична фигура може да се добави върху вече начертаната линия, за да се получи триизмерен обект. Четвъртият бутон е за съхранение на полученото изображение във формат png или jpg.



Обектът emitter излъчва частички. Всяка една от тях се движи по случаен начин. Има времетраене преди да изчезне и може да се свързва с всички други частички, за да образува линии. Всяка една частичка може да има текстура с различна прозрачност. Могат да се рендерират до 40 000 линии на 30 fps.

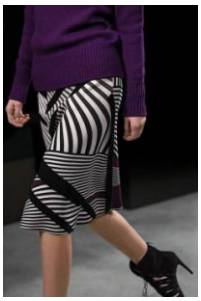
V. Приложения

Генерираните изображения могат да намерят широко приложение като лични проекти за:

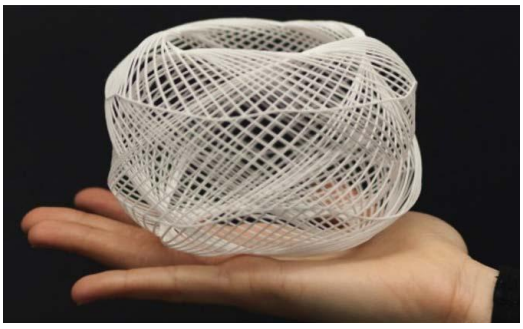
- Фототапети с абстрактни изображения
- Модерни арт-пана

- Дизайн на покривки, завеси и възглавници

- Украса на кутии и саксии
- Дизайн на облекла



- Сувенири за 3D принтиране



- Проекти за 3D принтирани кейсове за телефон

Използвана литература:

1. Anton's OpenGL 4 Notes
2. „Въведение в програмирането с Java” Лъчезар Цеков
3. Уикипедия
4. Математически основи за изчислителни алгоритми. Ускоряване на изчисленията в CAD/CAM системи и програми за анимация и визуализация чрез използване на кватерниони, инж. Валентин Лишков