1.Інтерфейс GDI Windows. Поняття контексту графічного пристроя. Огляд графічних примітивів GDI Windows. Приклади програмного коду.

GDI — це інтерфейс Microsoft Windows для представлення графічних об'єктів і передачі їх на пристрої відображення

Спочатку треба вказати, що для усіх функцій виводу графічних примітивів першим параметром вказується хендл (ідентифікатор) контексту графічного пристрою (HDC – Handle of Device Context). Значення hdc можна отримати для вікна так:  
*HDC hdc = BeginPaint(hWnd, &ps);*

*EndPaint(hWnd, &ps);*

парні функції BeginPaint – EndPaint можна використовувати лише при обробці повідомлення WM\_PAINT.

Для відображення у вікні також можна використати комплект функцій GetDC та ReleaseDC – так як записано нижче

*HDC hdc = GetDC(hWnd);*

*ReleaseDC(hWnd, hdc);*

Щодо графічних примітивів GDI Windows, GDI підтримує набір примітивів, серед яких є просто малювання окремих пікселей:

*SetPixel(hdc, x, y, clr);*

Також декілька графічних примітивів, які призначені для малювання ліній, ось декілька прикладів:

*MoveToEx(hdc, x1, y1, NULL);*

*LineTo(hdc, x2, y2);*

*Polyline(hdc, pt, 3);*

*PolyPolyline(hdc, pt, npt, 3);*

*Arc(hdc, x1, y1, x2, y2, xSt, ySt, xEn, yEn);*

*PolyBezier(hdc, pt, 7);*

Звісно можна змінити стиль лінії, наприклад змінити товщину чи малювати пунктирну лінію

Також є декілька графічних примітивів, які призначені для малювання фигур із заповненням:

*Ellipse(hdc, x1, y1, x2, y2); //Еліпс*

*Chord(hdc, x1, y1,x2, y2, xSt, ySt, xEn, yEn); // хорда еліпсу;*

*Pie(hdc, x1, y1, x2, y2, xSt, ySt, xEn, yEn); // сектор еліпсу;*

*Polygon(hdc, pt, 4); // полігон;*

*PolyPolygon(hdc, pt, npt, 3); //декілька полігонів та полігони з дірками;*

*Rectangle(hdc, x1, y1, x2, y2); // прямокутник;*

*RoundRect(hdc, x1, y1, x2, y2, cx, cy); //прямокутник із заокругленими кутами.*

2. Інтерфейс Canvas Android. Особливості налаштування відображення графіки у вікні застосунку. Огляд графічних примітивів Canvas Android. Приклади програмного коду.

Canvas — це API Android для роботи з 2D-графікою. Він забезпечує функції малювання ліній, кіл, тексту та взаємодіє з вікнами.

Canvas підтримує такі основні графічні примітиви як:

*drawLine(x1, y1, x2, y2, paint) — малювання лінії*

*drawCircle(cx, cy, radius, paint) — коло*

*drawRect(left, top, right, bottom, paint) — прямокутник*

*drawText(text, x, y, paint) — текст*

Але також є багато інших видів методів класу android.graphics.Canvas, виклик яких означатиме певну дію по відображенню чогось на растровій поверхні Canvas Bitmap.

Приклад коду Canvas:

*@Override*

*protected void onDraw(Canvas canvas) {*

*Paint paint = new Paint();*

*paint.setColor(Color.RED);*

*paint.setStrokeWidth(5);*

*canvas.drawLine(50, 50, 200, 200, paint);*

*canvas.drawCircle(150, 150, 50, paint);*

*canvas.drawRect(100, 100, 300, 300, paint);*

*}*

3. Інтерфейси OpenGL та OpenGL ES. Особливості налаштування відображення графіки OpenGL. Огляд графічних примітивів OpenGL. Приклади програмного коду на основі OpenGL.

OpenGL — це API для 3D-графіки, що забезпечує взаємодію з GPU для малювання складних сцен. OpenGL ES (Embedded Systems) — оптимізована версія OpenGL для мобільних пристроїв.

Щоб використовувати OpenGL ES в Android, можна скористатися класом GLSurfaceView. Потрібно створити власний клас MyGLSurfaceView, який буде успадковувати GLSurfaceView, і задати його як контейнер для рендерингу в MainActivity

Також необхідно реалізувати інтерфейс GLSurfaceView.Renderer з методами:

onSurfaceCreated() – викликається один раз при створенні поверхні, тут можна задати початкові налаштування, наприклад, колір фону.

onSurfaceChanged(width, height) – викликається при зміні розмірів вікна (наприклад, при повороті екрану).

onDrawFrame() – відповідає за рендеринг кадрів.

Режими рендерингу: GLSurfaceView.RENDERMODE\_WHEN\_DIRTY – кадр оновлюється тільки при виклику requestRender() або GLSurfaceView.RENDERMODE\_CONTINUOUSLY – кадри рендеряться постійно (підходить для анімації).

Базовими графічними примітивами в OpenGL ES є: точка, відрізок прямої лінії, трикутник.

Малювання, графічний вивід таких примітивів виконується викликом метода glDrawArrays( )

*public void onDrawFrame(GL10 unused) {*

*GLES32.glDrawArrays(mode, startindx, count);*

*}*

Можна вказати різні параметри для цього методу, наприклад: GL\_POINTS, GL\_LINES, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_TRIANGLES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN

Додатковий приклад коду OpenGL ES:  
*GLES20.glClear(GLES20.GL\_COLOR\_BUFFER\_BIT);*

*GLES20.glUseProgram(shaderProgram);*

*GLES20.glDrawArrays(GLES20.GL\_TRIANGLES, 0, 3);*

Цей код створює трикутник за допомогою OpenGL ES

4. Поняття шейдерів OpenGL. Основні типи шейдерів. Приклади вихідних текстів шейдерів.

Шейдер (англ. shader) – це програма, яка завантажується у пам'ять відеокарти і виконується графічним процесором (GPU) безпосередньо в ході рендерингу. Програмний код шейдерів для OpenGL ES має вигляд невеличкої програми з С-подібним синтаксисом.

Основними типами шейдерів OpenGL є

Вершинний шейдер (Vertex Shader), який обробляє кожну вершину об'єкта тавизначає позицію у просторі, застосовує трансформації (переміщення, масштабування, обертання).

Фрагментний шейдер (Fragment Shader), який обробляє кожен фрагмент (піксель) перед відображенням та визначає колір, текстуру та ефекти освітлення.

Приклади вихідних текстів шейдерів:

Вершинний шейдер (Vertex Shader):

*#version 300 es*

*layout(location = 0) in vec4 aPosition; // Вхідні координати вершини*

*uniform mat4 uMVPMatrix; // Матриця перетворення*

*void main() {*

*gl\_Position = uMVPMatrix \* aPosition; // Трансформація вершини*

*}*

Фрагментний шейдер (Fragment Shader):

*#version 300 es*

*precision mediump float;*

*out vec4 fragColor; // Вихідний колір*

*void main() {*

*fragColor = vec4(1.0, 0.0, 0.0, 1.0); // Червоний колір*

*}*

Цей код задає простий рендеринг, де всі пікселі мають червоний колір.

5. Приклади програмування поверхонь простих об’єктів таких як: трикутник, чотирикутник, піраміда, куб.

У OpenGL ES для відображення 3D-об’єктів використовуються вершини та індекси. Основні примітиви — трикутники (GL\_TRIANGLES), лінії (GL\_LINES), точки (GL\_POINTS).

Трикутник

*float[] vertices = {*

*0.0f, 0.5f, 0.0f,*

*-0.5f, -0.5f, 0.0f,*

*0.5f, -0.5f, 0.0f*

*};*

*GLES20.glDrawArrays(GLES20.GL\_TRIANGLES, 0, 3);*

Чотирикутник

Чотирикутники в OpenGL ES малюються як два трикутники:

*float[] vertices = {*

*-0.5f, 0.5f, 0.0f,*

*-0.5f, -0.5f, 0.0f,*

*0.5f, -0.5f, 0.0f,*

*0.5f, 0.5f, 0.0f*

*};*

*short[] indices = { 0, 1, 2, 0, 2, 3 };*

*GLES20.glDrawElements(GLES20.GL\_TRIANGLES, 6, GLES20.GL\_UNSIGNED\_SHORT, indexBuffer);*

Піраміда (з основою з 4-х трикутників)

*float[] vertices = {*

*0.0f, 0.5f, 0.0f,*

*-0.5f, -0.5f, 0.5f,*

*0.5f, -0.5f, 0.5f,*

*0.5f, -0.5f, -0.5f,*

*-0.5f, -0.5f, -0.5f*

*};*

*short[] indices = { 0, 1, 2, 0, 2, 3, 0, 3, 4, 0, 4, 1 };*

*GLES20.glDrawElements(GLES20.GL\_TRIANGLES, 12, GLES20.GL\_UNSIGNED\_SHORT, indexBuffer);*

Куб, що складається з 12 трикутників (2 на кожну грань):

*float[] vertices = {*

*-0.5f, 0.5f, 0.5f,*

*-0.5f, -0.5f, 0.5f,*

*0.5f, -0.5f, 0.5f,*

*0.5f, 0.5f, 0.5f,*

*-0.5f, 0.5f, -0.5f,*

*-0.5f, -0.5f, -0.5f,*

*0.5f, -0.5f, -0.5f,*

*0.5f, 0.5f, -0.5f*

*};*

*short[] indices = { 0,1,2, 0,2,3, 4,5,6, 4,6,7, 0,3,7, 0,7,4, 1,2,6, 1,6,5, 3,2,6, 3,6,7, 0,1,5, 0,5,4 };*

*GLES20.glDrawElements(GLES20.GL\_TRIANGLES, 36, GLES20.GL\_UNSIGNED\_SHORT, indexBuffer);*

6. Основні способи вказування кольорів об'єктів у середовищі OpenGL ES. Приклади програмного коду.

Глобальний колір (glUniform4f), де встановлюється єдиний колір для всього об’єкта

*int colorHandle = GLES20.glGetUniformLocation(shaderProgram, "uColor");*

*GLES20.glUniform4f(colorHandle, 1.0f, 0.0f, 0.0f, 1.0f); // Червоний*

Кольори вершин (атрибути aColor), що дозволяє задати різні кольори для кожної вершини

*attribute vec4 aColor;*

*varying vec4 vColor;*

*void main() {*

*vColor = aColor;*

*gl\_Position = uMVPMatrix \* aPosition;*

*}*

Текстури, коли колір береться з текстури:

*uniform sampler2D uTexture;*

*varying vec2 vTexCoord;*

*void main() {*

*gl\_FragColor = texture2D(uTexture, vTexCoord);*

*}*

7. Координатний метод. Поняття системи координат. Перетворення систем координат. Лінійні перетворення. Опис перетворень матрицями. Приклади програмного коду OpenGL.

Є декілька систем координат

Локальна (модельна) система координат — прив’язана до об'єкта.

Світова система координат — координати сцени.

Камера (видова) система координат — відображає сцену з точки зору камери.

Ортографічна та перспективна проекції — визначають відображення сцени на екран.

У OpenGL використовуються 4×4 матриці для перетворень, ось декілька прикладів лінійних перетворень:

Трансляція (переміщення об'єкта)

*Matrix.setIdentityM(modelMatrix, 0);*

*Matrix.translateM(modelMatrix, 0, 0.5f, 0.0f, 0.0f);*

Масштабування

*Matrix.scaleM(modelMatrix, 0, 2.0f, 2.0f, 2.0f);*

Обертання

*Matrix.rotateM(modelMatrix, 0, 45, 0, 1, 0);*

Перспективна проекція

Matrix.perspectiveM(projectionMatrix, 0, 45, aspectRatio, 0.1f, 100f);

Приклад програмного коду OpenGL з перетвореннями буде виглядати приблизно так:  
*float[] modelMatrix = new float[16];*

*float[] viewMatrix = new float[16];*

*float[] projectionMatrix = new float[16];*

*float[] MVPMatrix = new float[16];*

*Matrix.setLookAtM(viewMatrix, 0, 0, 0, 3, 0, 0, 0, 0, 1, 0);*

*Matrix.multiplyMM(MVPMatrix, 0, projectionMatrix, 0, viewMatrix, 0);*

*Matrix.multiplyMM(MVPMatrix, 0, MVPMatrix, 0, modelMatrix, 0);*

*int mvpHandle = GLES20.glGetUniformLocation(shaderProgram, "uMVPMatrix");*

*GLES20.glUniformMatrix4fv(mvpHandle, 1, false, MVPMatrix, 0);*