

PHÁT TRIỂN ỨNG DỤNG TRÊN THIẾT BỊ DI ĐỘNG

Hồ Văn Tú

Bộ môn Tin học ứng dụng
Khoa CNTT và truyền thông

hvtu@ctu.edu.vn

Chương 7

Đồ họa và Game trong Android

NỘI DUNG

- Đồ họa 2D cơ bản trong Android
 - Vẽ đối tượng
 - Thao tác trên đối tượng vẽ
- Game trong Android với SurfaceView và Thread
 - Các đối tượng và phương thức thông dụng
 - Tạo khung nền cơ bản cho Game
 - Tạo và sử dụng nhân vật Game
 - Một số vấn đề khác khi tạo Game trong Android

Vẽ đối tượng

- Sử dụng thư viện đồ họa trong gói android.graphics
- Các đối tượng cơ bản: Color, Paint, Bitmap, Canvas
- Thực hiện vẽ đối tượng trong phương thức
 - `onDraw(Canvas canvas)` của lớp `android.view.View`
 - `draw(Canvas canvas)` của lớp `android.view.SurfaceView`

Vẽ đối tượng - Color

- Thuộc lớp android.graphics.Color
- Hỗ trợ màu vẽ đối tượng
- Sử dụng màu cho trước hoặc tự chọn theo dạng ARGB
A: Alpha, độ trong suốt

Ví dụ:

```
int c1 = Color.Blue;  
  
int c2 = Color.argb(127, 0, 255, 0);
```

Vẽ đối tượng - Paint

- Thuộc lớp android.graphics.Paint
- Xác định màu sắc (color), kiểu (style) để vẽ đối tượng

Ví dụ:

```
Paint paint = new Paint();  
paint.setColor(Color.WHITE);  
canvas.drawPaint(paint); //Gán màu nền  
paint.setColor(Color.BLUE); //Gán màu vẽ  
paint.setTextSize(25); //Gán Font size  
paint.setStyle(Style.FILL); //Gán kiểu vẽ
```

Vẽ đối tượng - Bitmap

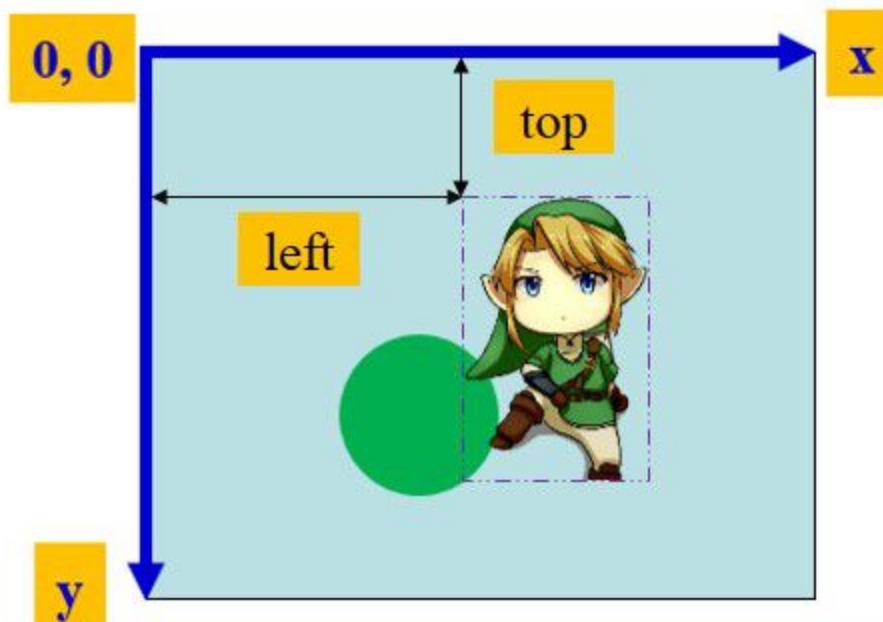
- Thuộc lớp android.graphics.Bitmap
- Hỗ trợ lưu trữ và xử lý đối tượng hình ảnh

Ví dụ:

```
Bitmap bitmap = BitmapFactory.decodeResource(  
    this.getResources(), R.mipmap.boboiboy);  
  
bitmap = Bitmap.createScaledBitmap(bitmap,  
    (int) (bitmap.getWidth() * 1.8),  
    (int) (bitmap.getHeight() * 1.8), true);
```

Vẽ đối tượng – Canvas (1)

- Thuộc lớp android.graphics.Canvas
- Đại diện cho bề mặt dùng để vẽ đối tượng
- Cung cấp các phương thức vẽ: văn bản, điểm, đường, hình chữ nhật, hình tròn, hình ảnh, ...



Vẽ đối tượng – Canvas (2)

- Phương thức thông dụng
 - drawColor(Color): gán màu vẽ
 - drawPaint(Paint): gán màu nền
 - drawText(String, x, y, Paint): vẽ nội dung văn bản tại vị trí cho trước
 - drawPoint(x, y, Paint): vẽ một điểm tại vị trí cho trước
 - drawLine(startX, startY, stopX, stopY, Paint): vẽ đường thẳng từ vị trí bắt đầu đến vị trí kết thúc

Vẽ đối tượng – Canvas (3)

- Phương thức thông dụng
 - drawCircle(centerX, centerY, radius, Paint): vẽ hình tròn với bán kính và tâm cho trước
 - drawRect(left, top, right, bottom, Paint): vẽ hình chữ nhật với vị trí hai điểm (left, top) và (right, bottom) cho trước
 - drawBitmap(Bitmap, left, top, Paint): vẽ Bitmap tại vị trí điểm bắt đầu (left, top) cho trước

Vẽ đối tượng – Canvas (4)

Ví dụ

@Override

```
public void onDraw(Canvas canvas) {  
    super.onDraw(canvas);  
    Paint paint = new Paint();  
    paint.setColor(Color.WHITE);  
    canvas.drawPaint(paint); //Gán màu nền  
    paint.setColor(Color.BLUE); //Gán màu vẽ  
    paint.setTextSize(70);  
    canvas.drawText("Graphics example", 10, 90, paint);  
    paint.setColor(Color.CYAN); //Gán màu vẽ  
    paint.setStyle(Paint.Style.FILL); //Gán kiểu vẽ  
    canvas.drawCircle(xPos, yPos, radius, paint);  
    canvas.drawBitmap(bitmap, bmLeft, bmTop, null);  
    Update(); // Cập nhật đối tượng  
}
```

Hoạt động của đối tượng (1)

- Làm thay đổi tọa độ, trạng thái, và vẽ lại đối tượng một cách liên tục
- Phương thức trong lớp View
 - getWidth(): nhận chiều rộng của View
 - getHeight(): nhận chiều cao của View
 - invalidate(): vẽ lại hình tại vị trí mới

Hoạt động của đối tượng (2)

- VD: cho đối tượng di chuyển: hình tròn bán kính radius, vị trí ban đầu (xPos, yPos), vận tốc ngang và dọc: dx và dy, nếu chạm vào biên thì hình quay lại.

```
private void Update() {  
    xPos = xPos + dx;  
    yPos = yPos + dy;  
    if ((xPos - radius <= 0) ||  
        (xPos >= getWidth() - radius))  
        dx = -dx;  
    if ((yPos - radius <= 0) ||  
        (yPos >= getHeight() - radius))  
        dy = -dy;  
    invalidate(); // Vẽ lại đối tượng
```

Xử lý sự kiện tương tác của người dùng (1)

- Tiếp nhận và xử lý sự kiện tương tác của người dùng
- Sử dụng phương thức **onTouchEvent**(MotionEvent event)

Phương thức event.**getAction()** nhận sự kiện tương tác

- MotionEvent.ACTION_DOWN: chạm xuống
- MotionEvent.ACTION_MOVE: chạm và di chuyển (vuốt)
- MotionEvent.ACTION_UP: nhấc lên
- ...

Xử lý sự kiện tương tác của người dùng (2)

- VD: nhận và xử lý sự kiện khi người dùng chạm xuống và khi người dùng chạm và di chuyển

```
@Override  
public boolean onTouchEvent(MotionEvent event) {  
    switch (event.getAction()) {  
        case MotionEvent.ACTION_DOWN:  
            xPos = (int) event.getX();  
            yPos = (int) event.getY();  
            break;  
        case MotionEvent.ACTION_MOVE:  
            bmLeft = (int) event.getX();  
            bmTop = (int) event.getY();  
            break;  
    }  
    return true;  
}
```

Tạo ứng dụng đồ họa 2D cơ bản (1)

- VD: Tạo ứng dụng thực hiện vẽ các đối tượng (văn bản, hình tròn, ảnh Bitmap), thêm hoạt động của đối tượng, nhận và xử lý sự kiện tương tác của người dùng



Tạo ứng dụng đồ họa 2D cơ bản (2)

```
public class MyView extends View {  
    private float xPos = 250;  
    private float yPos = 500;  
    private float radius = 100;  
    private Bitmap bitmap;  
    private float bmLeft = 500;  
    private float bmTop = 150;  
    private float dx = 6;  
    private float dy = 3;  
  
    public MyView(Context context) {  
        super(context);  
        bitmap = BitmapFactory.decodeResource(  
            this.getResources(), R.mipmap.boboiboy);  
        bitmap = Bitmap.createScaledBitmap(bitmap,  
            (int) (bitmap.getWidth() * 1.8),  
            (int) (bitmap.getHeight() * 1.8), true);  
    }  
}
```

Tạo ứng dụng đồ họa 2D cơ bản (3)

```
@Override
```

```
public void onDraw(Canvas canvas) {  
    super.onDraw(canvas);  
    Paint paint = new Paint();  
    paint.setColor(Color.WHITE);  
    canvas.drawPaint(paint); //Gán màu nền  
    paint.setColor(Color.BLUE); //Gán màu vẽ  
    paint.setTextSize(70);  
    canvas.drawText("Graphics example", 10, 90, paint);  
    paint.setColor(Color.CYAN); //Gán màu vẽ  
    paint.setStyle(Paint.Style.FILL); //Gán kiểu vẽ  
    canvas.drawCircle(xPos, yPos, radius, paint);  
    canvas.drawBitmap(bitmap, bmLeft, bmTop, null);  
    Update(); // Cập nhật đối tượng  
}
```

Tạo ứng dụng đồ họa 2D cơ bản (4)

```
private void Update() {  
    xPos = xPos + dx;  
    yPos = yPos + dy;  
    if ((xPos - radius <= 0) ||  
        (xPos >= getWidth() - radius))  
        dx = -dx;  
    if ((yPos - radius <= 0) ||  
        (yPos >= getHeight() - radius))  
        dy = -dy;  
    invalidate(); // Vẽ lại đối tượng  
}
```

Tạo ứng dụng đồ họa 2D cơ bản (5)

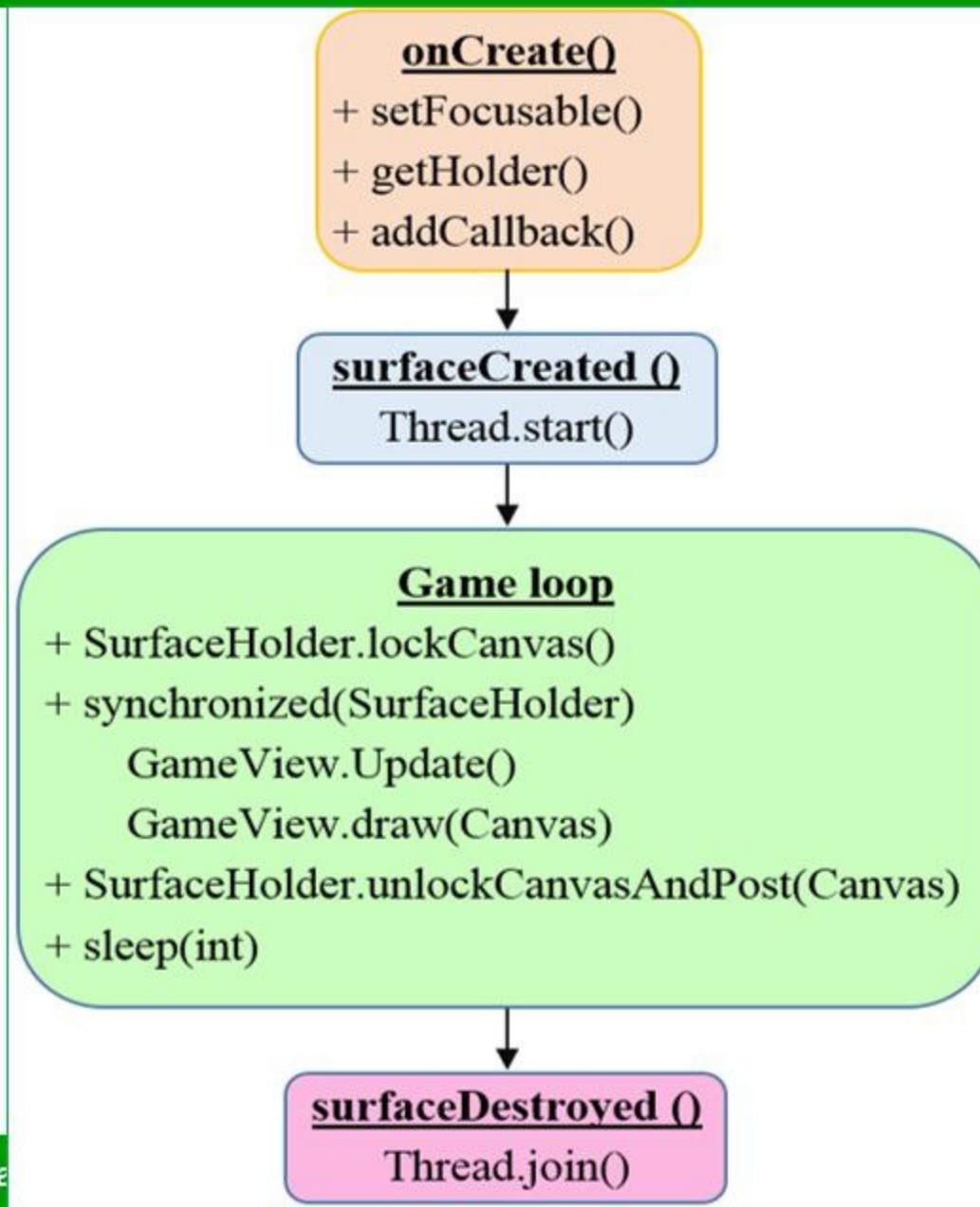
@Override

```
public boolean onTouchEvent(MotionEvent event) {  
    switch (event.getAction()) {  
        case MotionEvent.ACTION_DOWN:  
            xPos = (int) event.getX();  
            yPos = (int) event.getY();  
            break;  
        case MotionEvent.ACTION_MOVE:  
            bmLeft = (int) event.getX();  
            bmTop = (int) event.getY();  
            break;  
    }  
    return true;  
}
```

Tạo ứng dụng đồ họa 2D cơ bản (6)

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        getWindow().setFlags(  
            WindowManager.LayoutParams.FLAG_FULLSCREEN,  
            WindowManager.LayoutParams.FLAG_FULLSCREEN);  
        requestWindowFeature(Window.FEATURE_NO_TITLE);  
        setContentView(new MyView(this));  
    }  
}
```

Game trong Android với SurfaceView và Thread (1)



Game trong Android với SurfaceView và Thread (1)

- Đối tượng thông dụng
 - **SurfaceView**: thực hiện các thao tác đồ họa
 - **Thread**: điều khiển việc cập nhật lại giao diện, được xem là vòng lặp Game (Game loop)
 - **SurfaceHolder**: giao diện hỗ trợ quản lý bề mặt vẽ
 - **SurfaceHolder.Callback**: giao diện hỗ trợ nhận và cập nhật sự thay đổi bề mặt vẽ

SurfaceView (1)

- Thuộc lớp android.view.SurfaceView, kế thừa từ lớp View
- Là View được thiết kế lại hỗ trợ thể hiện hình ảnh cần cập nhật với tốc độ cao
- Thích hợp thể hiện dữ liệu video, camera, hoạt hình, Game
- Luôn chiếm một vùng màn hình (không thể bị che hoặc có bóng mờ)
- Đối tượng quản lý bề mặt hiển thị: SurfaceHolder
- Thực hiện vẽ đối tượng hợp lệ: giữa surfaceCreated và surfaceDestroyed

SurfaceView (2)

- Sự kiện và phương thức thông dụng
 - setFocusable(boolean): cho phép nhận Focus
 - getHolder() trả về SurfaceHolder quản lý bề mặt vẽ
 - Ghi đè phương thức draw(Canvas) của lớp SurfaceView để thực hiện các thao tác đồ họa
 - Thêm phương thức cập nhật vị trí và trạng thái của đối tượng vẽ
 - Tiếp nhận và xử lý sự kiện tương tác của người dùng: ghi đè sự kiện onTouchEvent(MotionEvent) của lớp View

Luồng_Thread (1)

- Tiến trình (Process): là chương trình trong thời gian thực hiện
- Luồng (Thread): là tiến trình nhỏ thực hiện công việc
- Đa luồng (MultiThreading): có hai hoặc nhiều phần chạy đồng thời và mỗi phần có thể xử lý tác vụ khác nhau tại cùng một thời điểm
- Lập trình luồng: lập trình hỗ trợ chương trình thực thi đa luồng

Luồng Thread (2)

- Phương thức thông dụng
 - start(): thực thi luồng
 - join(): yêu cầu chờ để kết thúc luồng
 - sleep(int): tạm dừng luồng
- Thực hiện
 - Tạo lớp mới GameThread kế thừa lớp Thread
 - Khai báo biến logic running
 - Ghi đè phương thức run
 - Thêm phương thức setRunning

Luồng Thread (3)

```
public class GameThread extends Thread {  
    private boolean running;  
    @Override  
    public void run() {  
        while (running)  
            try {  
                sleep(1000);  
            } catch (InterruptedException e) {  
            }  
    }  
  
    public void setRunning(boolean running) {  
        this.running = running;  
    }  
}
```

Luồng Thread (4)

- Sử dụng
 - Thực hiện

```
gameThread = new GameThread();
```

```
gameThread.setRunning(true);
```

```
gameThread.start();
```

- Kết thúc

```
gameThread.setRunning(false);
```

SurfaceHolder

- Thuộc lớp android.view.SurfaceHolder
- Là giao diện hỗ trợ quản lý bề mặt vẽ
- Các phương thức thông dụng
 - addCallback(SurfaceHolder.Callback): thêm phương thức gọi lại hỗ trợ quản lý bề mặt vẽ
 - lockCanvas(): nhận Canvas để thực hiện cập nhật (từng Pixels) trên bề mặt vẽ
 - unlockCanvasAndPost (Canvas): kết thúc cập nhật bề mặt vẽ

SurfaceHolder.Callback

- Thuộc lớp android.view.SurfaceHolder.Callback
- Là giao diện hỗ trợ nhận và cập nhật sự thay đổi bề mặt vẽ
- Thực hiện: cho lớp thực thi giao diện này và triển khai ba phương thức surfaceCreated(), surfaceChanged(), và surfaceDestroyed().
- Thời gian luồng vẽ tác động vào bề mặt vẽ hợp lệ: ở giữa surfaceCreated() và surfaceDestroyed().

Tạo khung nền cơ bản cho Game (1)

- **Bước 1:** tạo lớp GameView kế thừa SurfaceView
 - Ghi đè phương thức draw(Canvas) của lớp SurfaceView để thực hiện các thao tác đồ họa
 - Thêm phương thức cập nhật vị trí và trạng thái của đối tượng vẽ
 - Ghi đè phương thức sự kiện onTouchEvent(MotionEvent) của lớp View để tiếp nhận và xử lý sự kiện tương tác của người dùng

Tạo khung nền cơ bản cho Game (2)

- **Bước 2:** tạo lớp GameThread kế thừa Thread điều khiển việc cập nhật lại giao diện
 - Phương thức khởi tạo, gán/ nhận thuộc tính
 - Ghi đè phương thức run() thực hiện
 - Nhận Canvas và yêu cầu cho phép cập nhật bề mặt vẽ (lockCanvas)
 - Đồng bộ hóa SurfaceHolder (synchronized (SurfaceHolder)): cập nhật trạng thái, vị trí, và vẽ lại các đối tượng hình

Tạo khung nền cơ bản cho Game (3)

- **Bước 3:** khai báo lớp GameView thực thi giao diện SurfaceHolder.Callback. Triển khai ba phương thức
 - surfaceCreated(): khởi tạo đối tượng và thực thi lớp kế thừa Thread
 - surfaceChanged()
 - surfaceDestroyed(): yêu cầu chờ để kết thúc luồng
- **Bước 4:** tạo Activity MyGame không giao diện
 - MyGame hiển thị lớp GameView ở chế độ toàn màn hình, không có tiêu đề
 - Gán MyGame là Activity chính của ứng dụng

Khung nền cơ bản cho Game (1)

```
public class GameView extends SurfaceView
    implements SurfaceHolder.Callback{

    // Khai báo các biến hỗ trợ
    private GameThread myThread;

    public GameView(Context context) {
        super(context);
        setFocusable(true);
        getHolder().addCallback(this);
    }
}
```

Khung nền cơ bản cho Game (2)

```
@Override  
public void surfaceCreated(SurfaceHolder surfaceHolder) {  
    // Khởi tạo giá trị (nếu có)  
    myThread = new MyThread(this, surfaceHolder);  
    myThread.setTime(500);  
    myThread.start();  
}  
  
@Override  
public void surfaceChanged(SurfaceHolder surfaceHolder,  
                           int i, int i1, int i2) {  
}
```

Khung nền cơ bản cho Game (3)

```
@Override  
public void surfaceChanged(SurfaceHolder surfaceHolder,  
                           int i, int i1, int i2) {  
  
}  
  
@Override  
public void surfaceDestroyed(SurfaceHolder surfaceHolder) {  
    boolean retry = true;  
    while (retry) {  
        try {  
            myThread.join();  
        } catch (InterruptedException e) {  
        }  
        retry = false;  
    }  
}
```

Khung nền cơ bản cho Game (4)

```
@Override  
public void draw(Canvas canvas) {  
    super.draw(canvas);  
    // Thực hiện vẽ các đối tượng  
}  
  
public void update() {  
    // Cập nhật trạng thái, vị trí đối tượng  
}  
  
@Override  
public boolean onTouchEvent(MotionEvent event) {  
    // Tiếp nhận và xử lý sự kiện tương tác của người dùng  
}  
}
```

Khung nền cơ bản cho Game (5)

```
public class GameThread extends Thread {  
    private int time;  
    private GameView mySurface;  
    private SurfaceHolder myHolder;  
  
    public GameThread(GameView mySurface, SurfaceHolder myHolder) {  
        this.mySurface = mySurface;  
        this.myHolder = myHolder;  
    }  
  
    public int getTime() {  
        return time;  
    }  
  
    public void setTime(int time) {  
        this.time = time;  
    }  
}
```

Khung nền cơ bản cho Game (6)

```
@Override  
public void run() {  
    while (time > 0) {  
        Canvas canvas = null;  
        try {  
            // Nhận Canvas và cho phép cập nhật bề mặt vẽ  
            canvas = myHolder.lockCanvas();  
            // Đóng bộ hóa.  
            synchronized(myHolder) {  
                mySurface.Update();  
                mySurface.draw(canvas);  
            }  
        } catch (Exception e) {
```

Khung nền cơ bản cho Game (7)

```
        } finally {
            if (canvas != null) {
// Kết thúc cập nhật bề mặt vẽ
                myHolder.unlockCanvasAndPost(canvas);
            }
        }
    try {
// Ngừng chương trình một chút
        sleep(10);
        time--;
    } catch (InterruptedException e) {
        }
    }
}
```

Khung nền cơ bản cho Game (8)

```
public class MyGame extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        getWindow().setFlags(  
            WindowManager.LayoutParams.FLAG_FULLSCREEN,  
            WindowManager.LayoutParams.FLAG_FULLSCREEN);  
        requestWindowFeature(Window.FEATURE_NO_TITLE);  
        setContentView(new GameView(this));  
    }  
}
```

Ví dụ Game với SurfaceView và Thread (1)

```
public class GameView extends SurfaceView
    implements SurfaceHolder.Callback{

    private float xPos = 250;
    private float yPos = 500;
    private float radius = 100;
    private Bitmap bitmap;
    private float bmLeft = 500;
    private float bmTop = 150;
    private float dx = 6;
    private float dy = 3;
    private GameThread myThread;

    public GameView(Context context) {
        super(context);
        setFocusable(true);
        getHolder().addCallback(this);
    }
}
```

Ví dụ Game với SurfaceView và Thread (2)

```
@Override  
public void surfaceCreated(SurfaceHolder surfaceHolder) {  
    bitmap = BitmapFactory.decodeResource(  
        this.getResources(),  
        R.mipmap.boboiboy);  
    bitmap = Bitmap.createScaledBitmap(bitmap,  
        (int) (bitmap.getWidth()*1.8),  
        (int) (bitmap.getHeight()*1.8), true);  
    myThread = new GameThread(this, surfaceHolder);  
    myThread.setTime(500);  
    myThread.start();  
}  
  
@Override  
public void surfaceChanged(SurfaceHolder surfaceHolder,  
    int i, int i1, int i2) {  
}  
}
```

Ví dụ Game với SurfaceView và Thread (3)

@Override

```
public void surfaceDestroyed(SurfaceHolder surfaceHolder) {  
    boolean retry = true;  
    while (retry) {  
        try {  
            myThread.join();  
        } catch (InterruptedException e) {  
        }  
        retry = false;  
    }  
}
```

Ví dụ Game với SurfaceView và Thread (4)

```
@Override  
public void draw(Canvas canvas) {  
    super.draw(canvas);  
    Paint paint = new Paint();  
    paint.setColor(Color.WHITE);  
    canvas.drawPaint(paint);  
    paint.setColor(Color.BLUE);  
    paint.setTextSize(70);  
    canvas.drawText("Graphics example", 10, 90, paint);  
    int current = myThread.getTime();  
    canvas.drawText("Time to play: "  
        + current/10, 10, 200, paint);  
    paint.setStyle(Paint.Style.FILL);  
    paint.setColor(Color.CYAN);  
    canvas.drawCircle(xPos, yPos, radius, paint);  
    canvas.drawBitmap(bitmap, bmLeft, bmTop, null);  
}
```

Ví dụ Game với SurfaceView và Thread (5)

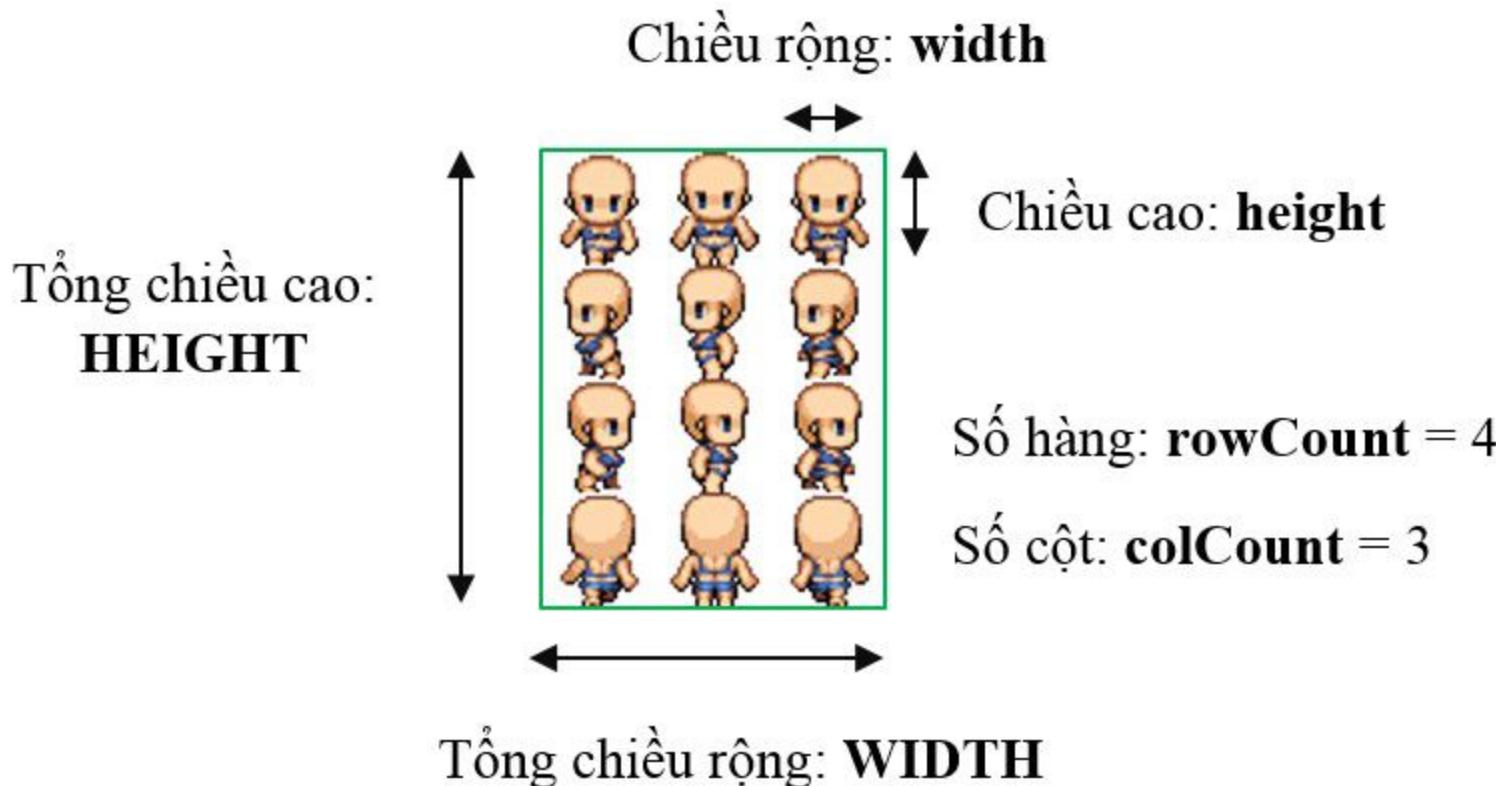
```
public void Update() {  
    xPos = xPos + dx;  
    yPos = yPos + dy;  
    if ((xPos - radius <= 0) ||  
        (xPos >= getWidth() - radius))  
        dx = -dx;  
    if ((yPos - radius <= 0) ||  
        (yPos >= getHeight() - radius))  
        dy = -dy;  
}
```

Ví dụ Game với SurfaceView và Thread (6)

```
@Override  
public boolean onTouchEvent(MotionEvent event) {  
    switch (event.getAction()) {  
        case MotionEvent.ACTION_DOWN:  
            xPos = (int) event.getX();  
            yPos = (int) event.getY();  
            break;  
        case MotionEvent.ACTION_MOVE:  
            bmLeft = (int) event.getX();  
            bmTop = (int) event.getY();  
            break;  
    }  
    return true;  
}  
}
```

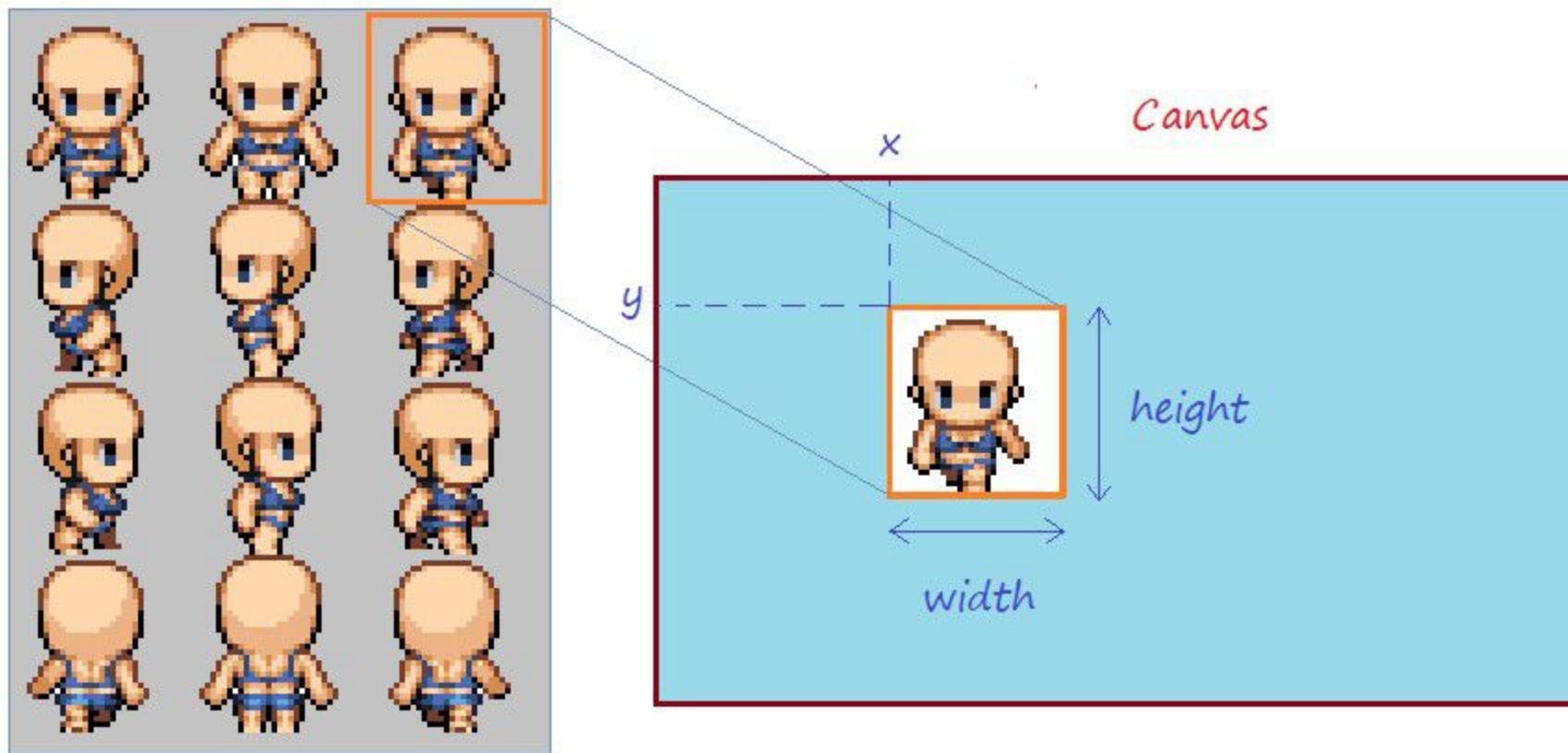
Tạo và sử dụng nhân vật Game (1)

- Để thuận tiện phát triển Game, tạo đối tượng nhân vật thay cho hình ảnh



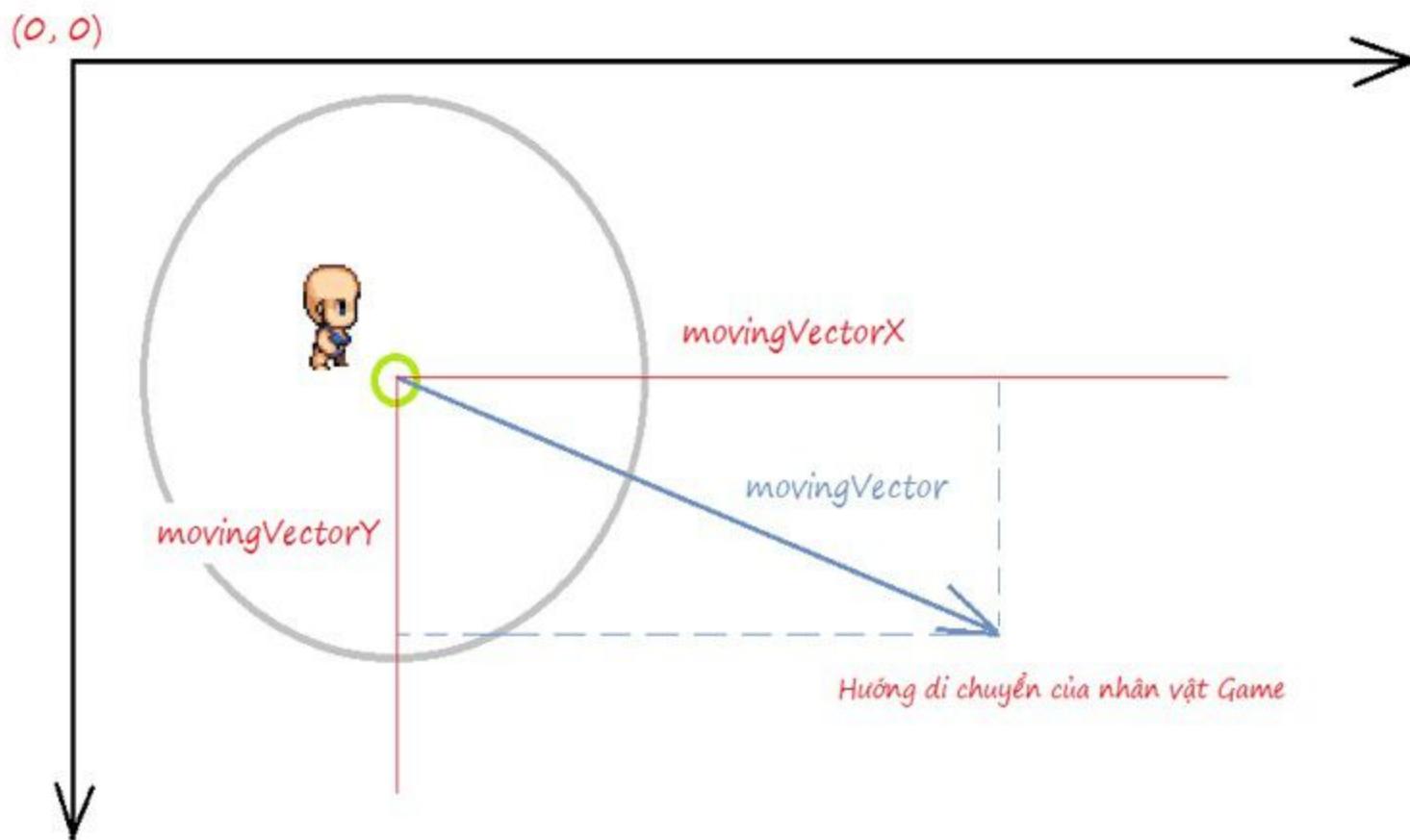
Tạo và sử dụng nhân vật Game (2)

- Sử dụng vòng lặp để liên tục vẽ lại trên Canvas tạo sự chuyên động của nhân vật



Tạo và sử dụng nhân vật Game (3)

- Xác định hướng di chuyển của nhân vật Game



Một số vấn đề khác khi tạo Game trong Android (1)

- Tạo lớp lưu các thông số chung
 - Các biến level, score, screenWidth, screenHeight, và screenScale lưu cấp độ, điểm, chiều rộng, chiều cao, và tỷ lệ theo kích thước màn hình
 - Phương thức GetScreenSize tính chiều rộng, chiều cao, và tỷ lệ theo kích thước màn hình (dựa trên màn hình chuẩn có chiều rộng 960) để có kích thước đối tượng vẽ phù hợp theo từng màn hình

Một số vấn đề khác khi tạo Game trong Android (2)

```
public class Publics {  
    public static int level;  
    public static int score;  
    public static int screenWidth;  
    public static int screenHeigh;  
    public static float screenScale;  
  
    public static void GetScreenSize(Context context) {  
        Display display = ((Activity) context).  
            getWindowManager().getDefaultDisplay();  
        DisplayMetrics metrics = new DisplayMetrics();  
        display.getMetrics(metrics);  
        screenWidth = metrics.widthPixels;  
        screenHeigh = metrics.heightPixels;  
        screenScale = (float) screenWidth / 960;  
        if (screenWidth > screenHeigh)  
            screenScale = (float) screenHeigh / 960;  
    }  
}
```

Một số vấn đề khác khi tạo Game trong Android (3)

- Sử dụng
 - Trong sự kiện onCreate() của Activity MyGame: gọi phương thức GetScreenSize(Context)
 - Khai báo Bitmap (các đối tượng vẽ khác thực hiện tương tự)

```
bitmap = Bitmap.createScaledBitmap(bitmap,  
        (int) (bitmap.getWidth() * Publics.screenScale) ,  
        (int) (bitmap.getHeight() * Publics.screenScale) ,  
        true);
```

Một số vấn đề khác khi tạo Game trong Android (4)

- Trong lớp GameView
 - Ghi đè phương thức onConfigurationChanged nhận biết sự thay đổi cấu hình (hướng màn hình, kích thước màn hình, sử dụng bàn phím)
 - Ghi đè phương thức onWindowFocusChanged nhận biết sự thay đổi cửa sổ ứng dụng (khi người sử dụng nhấn nút Recent Apps hoặc nhấn nút Home)

Một số vấn đề khác khi tạo Game trong Android (5)

- Trong lớp GameView

```
@Override  
public void onWindowFocusChanged(boolean hasWindowFocus) {  
    super.onWindowFocusChanged(hasWindowFocus);  
}  
  
@Override  
protected void onConfigurationChanged(  
    Configuration newConfig) {  
    super.onConfigurationChanged(newConfig);  
}
```

Một số vấn đề khác khi tạo Game trong Android (6)

- Tạo Activity và Service hỗ trợ
 - Activity chọn độ khó/ cấp độ, bật tắt âm thanh/ nhạc nền,
 - Activity lưu điểm cao,
 - Service âm thanh/ nhạc nền
 - Activity giới thiệu, hướng dẫn chơi.

Một số vấn đề khác khi tạo Game trong Android (7)

- Trong AndroidManifest
 - Cho phép sử dụng hay giới hạn hướng màn hình (đứng, ngang)
 - Khai báo lắng nghe sự thay đổi cấu hình để tránh khởi động lại Game

```
<activity  
    android:name=".MyGame"  
    android:screenOrientation="portrait"  
    android:configChanges="orientation|  
        screenSize|keyboard|keyboardHidden"  
    android:label="@string/my_game" >  
</activity>
```