Chuyển động và tập hợp đối tượng

Đối với việc chuyển động 1 đối tượng, thread chạy liên tục, các hàm vẽ cũng được chạy liên tục do đó khi mỗi lần vẽ ta chỉ cần cập nhật lại tọa độ x,y của đối tượng để tạo thành chuyển động.

Khi có nhiều đối tượng của cùng một class (vd: Các viên đạn được bắn ra) ta nên đưa nó vào 1 tập hợp (vd: dùng ArrayList) để dễ quản lý, và khi một đối tượng không còn dùng nữa (vd:viên đạn đã bay ra khỏi màn hình) ta sẽ xóa nó khỏi tập hợp. Cơ chế dọn rác của java sẽ dọn nó.

Thực hiện:

Phát triển tiếp từ bài 3:

Ta sẽ tạo ra các viên đạn được bắn ra liên tục từ đối tượng element.

Phần 1: Tạo class bullet (viên đạn).

1. Chép hình bên dưới (lua.png) vào thư mục res/drawable.

2. Tạo một class mới tên "Bullet" có mã lệnh như sau:

```
public class Bullet {
    int x;
    int y;
    Bitmap bitmap;
    int tocdo=20;

    public Bullet(Resources res,int x,int y)
    {
            this.x=x;
            this.y=y;
            bitmap=BitmapFactory.decodeResource(res,R.drawable.lua);
    }
    public Bullet(Resources res,int x,int y, int hinh)
    {
            this.x=x;
            this.y=y;
            bitmap=BitmapFactory.decodeResource(res, hinh);
    }
    public void doDraw(Canvas canvas)
    {
            canvas.drawBitmap(bitmap, x,y, null);
            x+=tocdo;

    }
    public void setXY(int x,int y)
    {
            this.x=x;
            this.y=y;
    }
    public void setTocDo(int x)
    {
            this.tocdo=x;
```

```
    }
}
```

Class này có 2 biến x,y dùng để điều khiển tọa độ x,y cho đối tượng. Biến bitmap dùng để lưu hình. Đặc biệt là biến tốc độ, do việc vẽ được thực hiện liên tục (ta đã giảm bớt nó trong file MainThread.java với lệnh sleep(50) trong hàm vẽ) khi ta vẽ viên đạn ra ta cần cập nhật lại tọa độ cho viên đạn, nếu tăng 1 sẽ rất chậm do đó ta tạo ra biến tốc độ để mỗi lần vẽ đối tượng được cập nhật theo tốc độ để viên đạn bay nhanh hơn.

Ta xây 2 hàm tạo, hàm sẽ nhận vào 2 biến int để cập nhật vị trí ban đầu cho viên đạn, khi tạo đối tượng viên đạn, ta sẽ truyền vô vị trí x,y chính là theo đối tượng element (vd: hình phi thuyền). Thì viên đạn sẽ được xuất phát từ đối tượng element (vd: xuất hiện ngay mũi phi thuyền).

Hàm doDraw sẽ đảm nhận việc vẽ viên đạn, xuất phát là theo tọa độ x,y. Sau đó mỗi lần vẽ lại thì trục x sẽ được cộng thêm biến tốc độ để viên đạn bay đi. Ta không cập nhật biến y vì viên đạn bay ngang trên 1 đường thẳng.

Ta xây thêm hàm setXY để khi cần cập nhật lại tọa độ, ta cũng xây thêm hàm setTocDo để sau này phát triển tiếp việc cập nhật lại tốc độ cho viên đạn.

Phần 2:kiểm tra xem viên đạn chạy được chưa

1. Quay về file “GamePanel.java” khai báo 1 biến toàn cục để tạo ra 1 đối tượng viên đạn.

```
    Bullet motviendan;
```

2. Trong hàm tạo khởi tạo biến. Cứ lấy tọa độ vẽ là 0,0. Ta sẽ sửa nó sau.

```
            motviendan=
                        new Bullet(getResources(),0, 0,R.drawable.lua);
```

3. Trong hàm onDraw ta thêm lệnh để vẽ đối tượng một viên đạn.

```
            if(myelement!=null)
            {
                  myelement.doDraw(canvas);//ve may bay
                  motviendan.doDraw(canvas);
            }
```

4. Chạy thử chương trình để thấy viên đạn được bay như thế nào.

Phần 3: vẽ nhiều viên đạn bay tại vị trí của element (phithuyen)

Do có nhiều viên đạn được bắn ra nên ta phải tạo ra một tập hợp các viên đạn. Ở đây ta sẽ dùng ArrayList.

1. Bỏ các lệnh tại bước 2 bao gồm: Bỏ biến toàn cục motviendan. Bỏ lệnh khởi tạo biến motviendan trong hàm tạo. Bỏ lệnh vẽ motviendan.
2. Ta khai báo 1 biến tòa cục tên bullets kiểu tập hợp và từng phần tử là từng đối tượng thuộc lớp Bullet.

```
    ArrayList<Bullet> bullets=new ArrayList<Bullet>();
```

3. Ta xây dựng 1 hàm mới tên doDrawBullet để vẽ các viên đạn.

```
    //ve tap hop cac vien dan
    public void doDrawBullet(Canvas canvas)
    {
          Bullet motviendan=
                      new Bullet(getResources(), myelement.mX,
myelement.mY,R.drawable.lua);

          bullets.add(motviendan);
          for(int i=0;i<bullets.size();i++)
                bullets.get(i).doDraw(canvas);

    }
```

Hàm này sẽ khởi tạo một viên đạn có tọa độ (myelement.mX, myelement.mY) chính là tọa độ của element (phi thuyền) vì vị trí đầu tiên của viên đạn là bắn ra từ phi thuyền.

Sau đó ta thêm nó vào ArrayList. Và cuối cùng ta sẽ dùng for để duyệt tất cả các phần tử trong tập hợp và vẽ lại từng bullet một. Do hàm này cũng sẽ đưa vào vẽ liên tục nên từng bullet cũng liên tục được tạo ra và tất cả các bullet cũ và bullet mới tạo đều được vẽ lại và cập nhật vị trí.

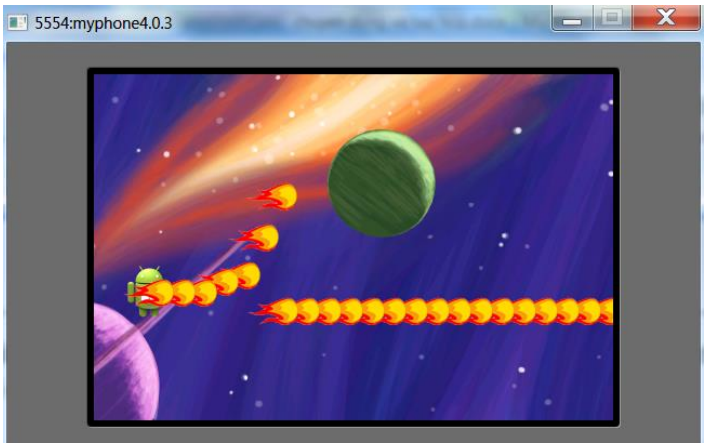
4. Trong hàm onDraw, sau khi vẽ element ta thực hiện gọi hàm doDrawBullet để vẽ tập hợp cac Bullet.

```
          if(myelement!=null)
          {
                myelement.doDraw(canvas);//ve may bay
                this.doDrawBullet(canvas); //ve tap hop vien dan
          }
```

5. Chạy chương trình và thấy rằng bullet được bắn ra đã đúng vị trí cũng như có tốc độ bay nhưng bullet bắn ra lại liên tục do thread chạy liên tục, việc vẽ cũng liên tục nên đối tượng bullet cũng được tạo ra liên tục.

Nếu ở đây ta cho người sử dụng điều khiển việc bắn thì rất dễ chỉ cần bắt sự kiện touch của người sử dụng để tạo ra 1 bullet mới. Nhưng ở đây ta cho chạy tự động nên phát sinh vấn đề.

Bây giờ ta phải cho bullet bắn có độ trễ giữa 2 bullet. Có thể dùng thread mới nhưng rất mệt. Ở đây ta chỉ cần cho 1 biến đếm để việc vẽ lại thực hiện vài lần thì bullet mới khởi tạo 1 lần. Và ta xem nó đơn giản là thời gian nạp đạn giữa 2 lần bắn.



Phần 4: Xây dựng thời gian nạp đạn giữa 2 lần bắn.

Để tạo độ trễ giữa 2 lần bắn thì ta sẽ tạo ra 1 biến. sau đó mỗi lần vẽ lại ta tăng biến này lên. Trong ham vẽ các bullet ta sẽ xét biến này khi nó bằng 10 (tức là đã vẽ lại 10 lần) thì ta mới tạo đối tượng bullet mới và cập nhật biến về 0.

1. Khai báo 1 biến toàn cục

```
    int thoigiannapdan=0; //bang 10 moi ban tiep duoc, tao do tre khi ban
```

2. Trong hàm onDraw mỗi lần vẽ ta sẽ cập nhật biến này lại

```
……
            background.doDrawRunning(canvas);
            thoigiannapdan++;
            if(myelement!=null)
            {
                  myelement.doDraw(canvas);//ve may bay
                  this.doDrawBullet(canvas); //ve tap hop vien dan
            }

……
```

3. Trong hàm doDrawBullet ta dùng drawText để vẽ ra biến thoigiannapdan để dễ kiểm tra kết quả. Tiếp theo ta xét biến thoigiannapdan. Nếu nó >=10 thì ta gán nó về 0 và tạo đối tượng mới bullet mới. Vậy ta sẽ chạy hàm vẽ 10 lần thì mới tạo bullet 1 lần, đây chính là thời gian trể giữa 2 lần tạo bullet.

```
    //ve tap hop cac vien dan
    public void doDrawBullet(Canvas canvas)
    {

            Paint p=new Paint();
            p.setColor(Color.WHITE);
            p.setTextSize(20);
            canvas.drawText("napdan:"+thoigiannapdan, 20, 20,p);

            if(thoigiannapdan>=10)
            {
                  thoigiannapdan=0;
                  Bullet motviendan=
                              new Bullet(getResources(), myelement.mX,
myelement.mY,R.drawable.Lua);

                  bullets.add(motviendan);
```

```
        }
        for(int i=0;i<bullets.size();i++)
                bullets.get(i).doDraw(canvas);
    }
```

4. Chạy và thấy bullet đã chạy đúng ý đồ.

Phần 5: Tối ưu việc tạo bullet và vẽ thời gian nạp đạn.

1. Trong hàm doDraw trong file GamePanel ta thêm 1 lệnh dưới cùng trong hàm

```
        Log.d("viendan","so vien: "+bullets.size());
```

2. Chạy và theo dõi Logcat ta thấy số viên đạn sẽ được tăng lên liên tục do mỗi viên đạn được bắn ra thì đối tượng bullet được tạo ra và đưa vào ArrayList. Điều này sẽ không tốt do ArrayList sẽ càng lúc càng lớn ra và chứa một khối lượng rất lớn các bullet.

3. Ta sẽ khắc phục bằng cách sau khi vẽ xong các bullet ta sẽ duyệt và xem nếu bullet nào vượt khỏi chiều ngang của thiết bị thì sẽ xóa bullet đó ra khỏi ArrayList, sau đó cơ chế dọn rác tự động sẽ dọn dẹp nó. Sau vòng for để vẽ các bullet, ta thêm 1 vòng for để xét và xóa các bullet.

```
        ……….
        for(int i=0;i<bullets.size();i++)
                bullets.get(i).doDraw(canvas);
        for(int i=0;i<bullets.size();i++)
                if(bullets.get(i).x>canvas.getWidth())
                        bullets.remove(i);
        ……….
```

4. Chạy và xem lại logcat ta thấy số lượng bullet trong ArrayList sẽ tăng và giảm liên tục phụ thuộc vào số viên đạn còn thấy được trên màn hình.
5. Để trang trí thêm ngay tại vị trí vẽ Text thoigiannapdan ta sửa lại.

```
            //left, top, right, bottom, paint
            canvas.drawRect(10,10, thoigiannapdan*10, 20, p);
```

Ở đây thay vì chạy số từ 0->10 ta sẽ thay bằng việc vẽ 1 hình chữ nhật. Thoigiannap sẽ thay đổi liên tục nên ta sẽ có hình chữ nhật chạy dài ra và ngắn lại liên tục (thay đổi chiều rộng của hình chữ nhật theo biến thoigiannapdan), ở đây do biến thoigiannapdan chỉ tối đa là 10 nên vẽ ra hơi nhỏ ta nhân biến cho 10 để hình chữ nhật dài hơn.