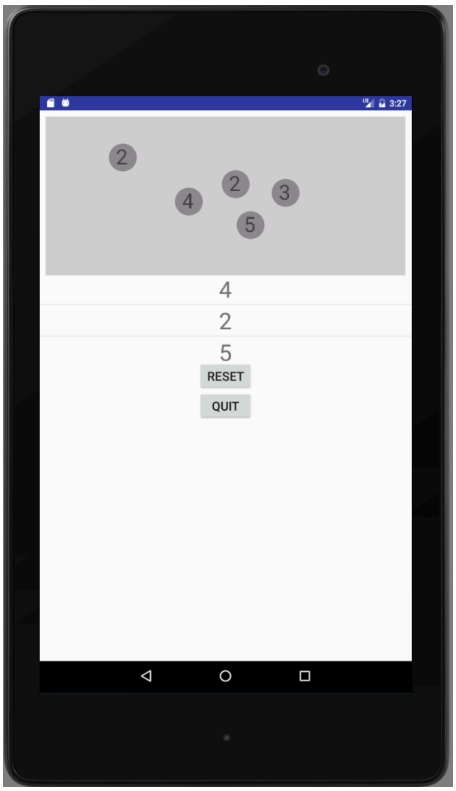
Un jeu à peine moins bête

Le jeu des jetons

Au début du jeu, on dispose d'une liste de jetons numérotés. Le joueur doit prendre des jetons et les poser sur la surface de jeu.

Une interaction simple

- toucher un *item* de la liste pour le prendre
- toucher un point de la surface pour le poser



Modèle du jeu

Une réprésentation des jetons

```
public class Token {
    int id, x, y;
    Token(int id, int x, int y) {
        this.id = id; this.x = x; this.y = y;
    }
    int getId() { return id; }
    int getX() { return x; }
    int getY() { return y; }
}
```

Légère modification du modèle

```
public class Model {
    ArrayList<Token> ts;
    Model() { ts = new ArrayList<Token>(); }
    void add(int id, int x, int y) { ts.add(new Token(id,x,y)); }
    ListIterator<Token> getAll() { return ts.listIterator(); }
}
```

Gestion de la partie

Confiée à l'application :

- liste de jetons à poser `newGame` et `getGame`
- prise d'un jeton (sélection) `setSelectedToken`
- poser un jeton (désélection) `unsetSelectedToken`
- connaître le jeton pris `getSelectedToken`

La piste de jeu

La *surface view* `GameBoard` (renommée `TokenBoard`) avec modification de l'affichage

```
public void drawToken(Token t, Canvas c, Paint p) {
        p.setColor(Color.GRAY);
        c.drawCircle(t.getX(),t.getY(),45,p);
        p.setColor(Color.DKGRAY);
        p.setTextSize(67);
        c.drawText(""+t.getId(), t.getX()-22, t.getY()+22,p);
    }
```

Dans `onDraw`

```
        ListIterator<Token> it = m.getAll();
        while(it.hasNext()) {
            Token t = it.next();
            p.setColor(Color.DKGRAY);
            drawToken(t,c, p);
        }
```

La liste des jetons à poser : visualisation

Utilisation d'une `ListView` : liste déroulante contenant des *vues* éventuellement encapsulée dans un layout

1. la *vue* des *item* de la liste : `token_item.xml`.
2. actualisation des *item* visibles : classe `TokenListAdapter`, hérite de `ArrayListAdapter`.
3. réaction à la sélection (touché) d'un *item* : classe `TokenListListener`, implémente `AdapterView.OnItemClickListener`.

Vue d'un *item*

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >
    <TextView
        android:id="@+id/textItem"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:textSize="37dp"
        android:text=""/>
</LinearLayout>
```

Notez : un `TextView` nommé (`@+id/textItem`)

Renseignement des *item*

```
public class TokenListAdapter extends ArrayAdapter<Integer> {
   LayoutInflater inflater;
   TokenListAdapter(Context c, List<Integer> ns) {
     super(c,R.layout.token_item,ns);
     inflater = LayoutInflater.from(c);
   }
   @Override
   public View getView(int i, View v, ViewGroup vg) {
     if (v == null) {
       v = inflater.inflate(R.layout.token_item,vg, false);
     }
     TextView txt = (TextView)v.findViewById(R.id.textItem);
     txt.setText(getItem(i).toString());
     return cv;
  }
}
```

Notez : `inflater.inflate` de XML à JAVA

Réaction

```
public class TokenListListener
       implements AdapterView.OnItemClickListener {
    TheApplication app;
    TokenListListener(TheApplication app) {
       super();
       this.app = app;
    }
    @Override
    public void onItemClick(AdapterView<?> av,
                            View v, int i, long id) {

        app.setSelectedToken(app.getGame().get(i));
    }
}
```

Ajout de la liste à l'activité

Dans `activity_play.xml`, entre `GameBoard` et `Button`

```
<ListView
    android:layout_width="match_parent"
    android:layout_height="128dp"
    android:layout_gravity="center"
    android:id="@+id/token_list_view"
    />
```

Dans la classe `PlayActivity` (constructeur)

```
tokenListView = (ListView)findViewById(R.id.token_list_view);
ArrayList<Integer> ns = app.getGame();
TokenListAdapter adapter = new TokenListAdapter(this, ns);
tokenListView.setAdapter(adapter);
TokenListListener listener = new TokenListListener(app);
tokenListView.setOnItemClickListener(listener);
```

Une autre interaction

Le glisser-déposer

Activé sur «click long»

1. Répondre à l'événement «click long» dans `TokenListListener`
   - interface `AdapterView.OnItemLongClickListener`
   - définir méthode `OnItemLongClick`
2. Répondre à l'événement «déposer» dans `TokenBoard`
   - interface `View.OnDragListener`
   - ajouter à `onCreate` : `setOnDragListener`
   - définir méthode `onDrag`
3. Mettre la liste de jetons à l'écoute des «clicks longs»
   - dans `PlayActivity` ajouter à `onCreate` `setItemLongClickListener`

Répondre au «click long»

Déclencher le «glisser» (`startDrag`)

```
public class TokenListListener
       implements AdapterView.OnItemClickListener,
                  AdapterView.OnItemLongClickListener {
    [..]
    public boolean onItemLongClick(AdapterView<?> av,
                                   View v, int i, long id) {
        app.setSelectedToken(app.getGame().get(i));
        ClipData data = ClipData.newPlainText("", "");
        View.DragShadowBuilder shadowBuilder =
             new View.DragShadowBuilder(v);
        v.startDrag(data, shadowBuilder, null, 0);
        return true;
    }
```

Effectuer le «déposer»

```
public class TokenBoard extends SurfaceView
        implements SurfaceHolder.Callback, View.OnDragListener {
      [...]
    public TokenBoard(Context c, AttributeSet as) {
        [...]
        this.setOnDragListener(this);
    }
    @Override
    public boolean onDrag(View v, DragEvent event) {
        (A SUIVRE...)
        }
    }
}
```

Effectuer le «déposer» (SUITE)

```
@Override
public boolean onDrag(View v, DragEvent event) {
    int x = (int) event.getX();
    int y = (int) event.getY();
    int action = event.getAction();
    switch (action) {
        case DragEvent.ACTION_DROP:
            app.getModel().add(app.getSelectedToken(),x,y);
            reDraw();
            return true;
        case DragEvent.ACTION_DRAG_ENDED:
            app.unsetSelectedToken();
            return true;
        default:
            return true;
    }
}
```

Mettre à l'écoute des «clicks longs»

Dans la classe `PlayActivity`

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    [..]
    TokenListListener listener = new TokenListListener(app);
    [..]
    tokenListView.setOnItemLongClickListener(listener);
}
```

Une liste d'images

1. modifier `token_item.xml`
2. engendrer l'image d'un jeton : méthode `drawableToken`
3. assigner l'image aux *items* de la liste : méthode `getView` de `TokenListAdapater`

On pourrait aussi prendre des images préfabriquées stockées dans `res/drawable`.

token_item.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >
    <ImageView
        android:id="@+id/imageItem"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"/>
</LinearLayout>
```

Engendrer une image

Dans TokenListAdapter (par exemple)

```
Drawable drawableToken(int id) {
   Bitmap b = Bitmap.createBitmap(90,90,Bitmap.Config.RGB_565);
   Canvas cv = new Canvas(b);
   cv.drawColor(Color.WHITE);
   Paint pt = new Paint();
   [ dessin: cf. drawToken ]
   return new BitmapDrawable(this.getContext().getResources(),b)
}
```

L'assigner aux éléméments de la liste

```
public View getView( ... ) {
 if (convertView == null) {
    convertView = inflater.inflate(R.layout.token_item,parent, fa
 }
 ImageView img = (ImageView)convertView.findViewById(R.id.imageIt
 img.setImageDrawable(drawableToken(getItem(position)));
 return convertView;
}
```