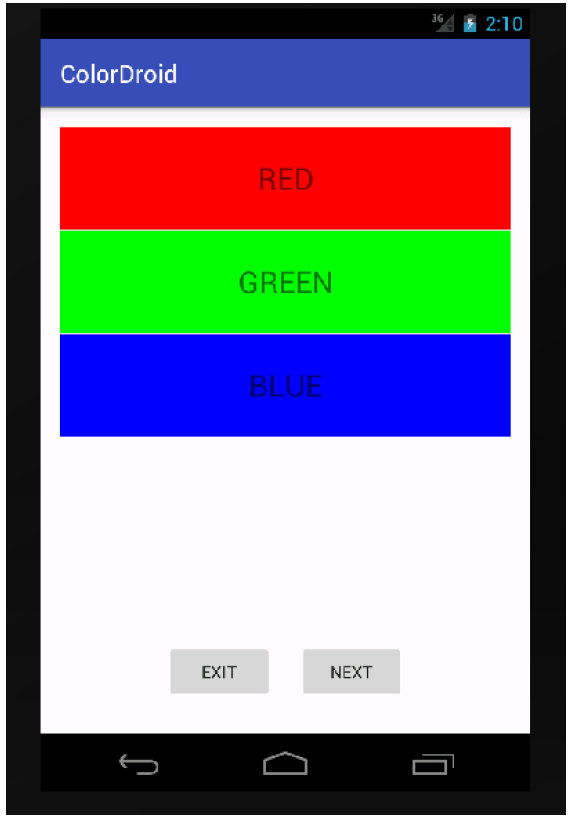
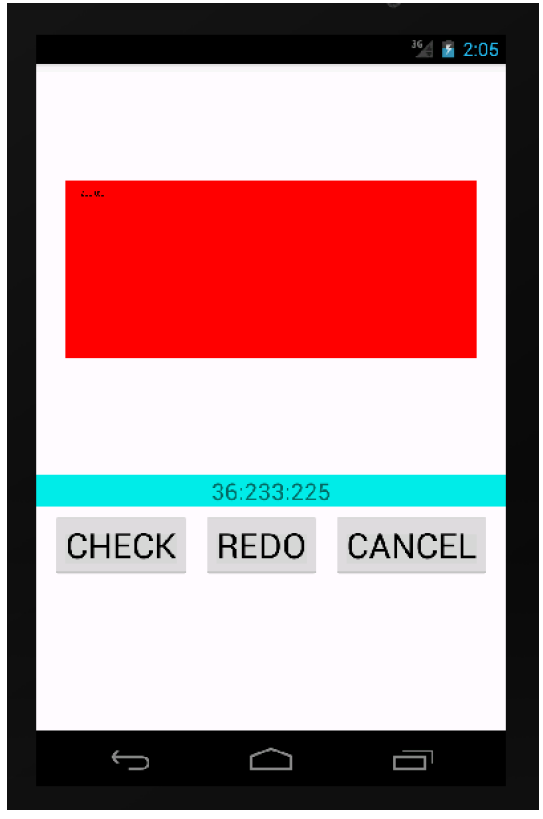
Un nouveau jeu

Trouver, à partir d'une couleur de départ, une couleur en modifiant les composantes rouge, vert et bleu de la couleur de départ. On peut choisir la couleur de départ.



`MainActivity`



`GameActivity`

GameModel

Modèle crée à la création de l'application.

```
public class GameModel {

    int getColor();
    int getStartColor();
    int getEndColor();
    void setStartColor(int c);
    void setEndColor(int c);
    void resetColor();
    void changeColor(int c, boolean sign);
    boolean endOfGame();
}
```

Constructeur : `GameModel()`

Classes utilitaires

Une couleur

```
public class ColorData {
    ColorData(String name, int color)
    String getName()
    int getColor()
    boolean getDone()
    void setDone(boolean b)
}
```

Un jeu : plusieurs couleurs de départ, une couleur à atteindre

```
public class ColorPuzzle {
    ArrayList<ColorData> getStartColors()
    ColorData getStartColor(int i)
    int getEndColor()
    public void addStartColor(ColorData colorItem)
    public void setEndColor(int endColor)
}
```

L'application

```
public class TheApplication extends Application {
    @Override
    public void onCreate()

    GameModel getGame()
    void setGame(ColorPuzzle puzzle, int i)

    int getCurrentPuzzleIndex()
    ColorPuzzle getCurrentPuzzle()
    void setCurrentPuzzle(int i)
    void nextPuzzle()
```

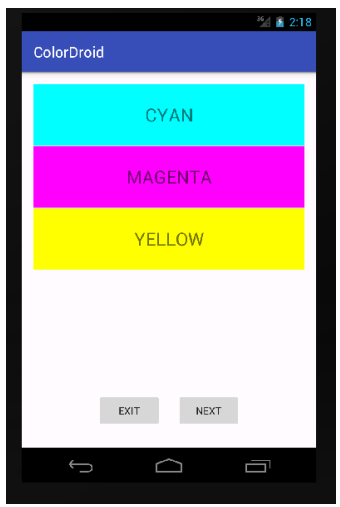
L'application (suite)

```
void inputSomeColorPuzzles() {

  colorPuzzles = new ArrayList<ColorPuzzle>();
  Random ran = new Random();

  ColorPuzzle colorPuzzle = new ColorPuzzle();
  colorPuzzle.setEndColor(Color.rgb(ran.nextInt(255), ran.nextInt(255),
                                    ran.nextInt(255)));
  colorPuzzle.addStartColor(new ColorData("RED", Color.RED));
  colorPuzzle.addStartColor(new ColorData("GREEN", Color.GREEN));
  colorPuzzle.addStartColor(new ColorData("BLUE", Color.BLUE));
  colorPuzzles.add(colorPuzzle);

  colorPuzzle = new ColorPuzzle();
  colorPuzzle.setEndColor(Color.rgb(ran.nextInt(255), ran.nextInt(255),
                                    ran.nextInt(255)));
  colorPuzzle.addStartColor(new ColorData("CYAN", Color.CYAN));
  colorPuzzle.addStartColor(new ColorData("MAGENTA", Color.MAGENTA));
  colorPuzzle.addStartColor(new ColorData("YELLOW", Color.YELLOW));
  colorPuzzles.add(colorPuzzle);

  }

}
```

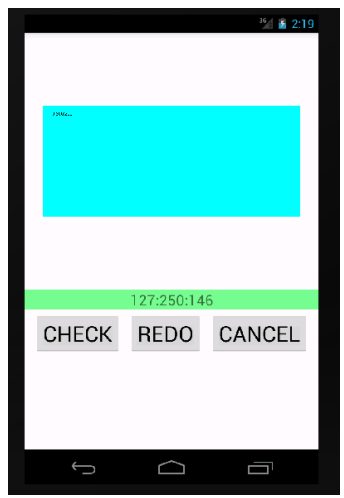
MainActiviy

Composants de l'interface (la *vue*)

- une *liste* pour choisir la couleur de départ (composant `ListView`) ;
- un bouton pour terminer (`EXIT`) ;
- un bouton pour accéder à une autre liste de couleurs de départ (`NEXT`)



`MainActivity`



`GameActivity`

activity_main.xml

```
<RelativeLayout [..] >
    <TextView [..]
        android:text="Color Droid" />
    <ListView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/colorList" />
    <LinearLayout [..] >
        <Button [..]
            android:id="@+id/buttonExit" />
        <Button [..]
            android:text="NEXT"
            android:onClick="onClickNext"
            android:id="@+id/buttonNext" />
    </LinearLayout>
</RelativeLayout>
```

GameActivity

Composants de l'interface et comportement

- la surface de jeu : `SurfaceView` réactive ;
- la couleur à atteindre : `TextView`
- trois boutons :

`CHECK`	si la couleur de la surface est suffisament proche de celle à atteindre, retour à `MainActivity` avec message de succès ;
`REDO`	réinitialisation de la couleur de la surface ;
`CANCEL`	retour à `MainActivity` avec message d'abandon.

Retour à `MainActivity` avec succés $\Rightarrow$ affichage `COLOR:DONE`

activity_game.xml

```
<LinearLayout [..] >

<android.appdroid.GameView [..] />

<TextView
    android:id="@+id/colorToFind"
    [..] />

<LinearLayout [..] >
    <Button [..]
        android:onClick="onClickCheck" />
    <Button [..]
        android:onClick="onClickRedo" />
    <Button [..]
        android:onClick="onClickCancel" />
</LinearLayout>
</LinearLayout>
```

`MainActivity` : la liste des couleurs de départ

Gestion des `ListView` :

1. définir le contenu de la liste (déroulante) et sa visibilité
2. définir la réaction à la sélection d'un *item*

Le contenu :

- Une liste de `TextView` avec un texte et une couleur de fond donnés dans une `List<ColorData>` ;
- Une méthode pour actualiser les `items` visibles : `ColorListAdapter` ;

La réaction

- Une méthode pour lancer l'activité du jeu sélectionné : `ColorListListener`

Une ressource XML

La vue d'un *item* de la liste

```
<TextView
  xmlns:android="http://schemas.android.com/apk/res/andro
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:padding="25dp"
  android:background="#fff"
  android:gravity="center"
  android:text="empty"
  android:textSize="25dp"
  android:id="@+id/color_item"
  />
```

ColorListAdapter

```
public class ColorListAdapter extends ArrayAdapter<ColorData> {

  ColorListAdapter(Context context, List<ColorData> itemDatas) {
      super(context, 0, itemDatas); // 0 ?
  }
  @Override
  public View getView(int position, View convertView,
                              ViewGroup parent) {
      if (convertView == null) {
        convertView =
          LayoutInflater.from(getContext())
          .inflate(R.layout.color_item,parent, false);
      }
      setData((TextView)convertView, getItem(position));
      return convertView;
  }
  [...]
```

Expansion d'une description XML : LayoutInflater

ColorListAdapter (suite)

```
    [...]
    void setData(TextView itemView, ColorData itemData) {
        if (itemData.getDone())
            itemView.setText(itemData.getName()+" : DONE");
        else
            itemView.setText(itemData.getName());
        itemView.setBackgroundColor(itemData.getColor());

    }
}
```

Modification dynamique du contenu d'une vue : `setText`, `setBackGroundColor`

ColorListListener

```
public class ColorListListener
             implements AdapterView.OnItemClickListener {
  TheApplication app;
  MainActivity act;
  ColorListListener(TheApplication app, MainActivity act) {
    super(); this.app = app; this.act = act;
  }
  @Override
  public void onItemClick(AdapterView<?> parent,
                          View v, int position, long id) {
    app.setGame(app.getCurrentPuzzle(), position);
    Intent gameIntent =
      new Intent(app.getApplicationContext(),
                 GameActivity.class);
    act.startActivityForResult(gameIntent,
                               MainActivity.GAME_ACTIVITY);
  }
}
```

On attend un résultat : `startActivityForResult`

MainActivity (1/3)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  app = (TheApplication)this.getApplication();
  app.inputSomeColorPuzzles();
  app.setCurrentPuzzle(0);

  List<ColorData> itemDatas =
     app.getCurrentPuzzle().getStartColors();
  colorListView = (ListView)findViewById(R.id.colorList);
  colorListView.setAdapter(new ColorListAdapter(this, itemDatas));

  ColorListListener listener = new ColorListListener(app,this);
  colorListView.setOnItemClickListener(listener);
}
```

MainActivity (2/3)

```
public void onClickNext(View view) {
  app.nextPuzzle();
  colorListView = (ListView) findViewById(R.id.colorList);
  List<ColorData> itemDatas =
     app.getCurrentPuzzle().getStartColors();
  colorListView.setAdapter(new ColorListAdapter(this, itemDatas)
}
```

MainActivity (3/3)

```
@Override
protected void onActivityResult(int requestCode,
                                int resultCode, Intent data) {
  if (resultCode == RESULT_OK) {
    ArrayList<ColorData> startColors =
       app.getCurrentPuzzle().getStartColors();
    int resColor = data.getIntExtra("color", 0);
    boolean found = false;
    for (int i=0; !found && (i < startColors.size()); i++)
       if (startColors.get(i).getColor() == resColor) {
         startColors.get(i).setDone(true);
         found = true;
       }
    colorListView
       .setAdapter(new ColorListAdapter(this, startColors));

  }
}
```

GameActivity (1/3)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_game);
  app = (TheApplication)this.getApplicationContext();

  TextView endColorView =
     (TextView)findViewById(R.id.colorToFind);
  int endColor = app.getGame().getEndColor();
  endColorView.setBackgroundColor(endColor);
  endColorView.setText(app.stringRGB(endColor));
}
```

GameActivity (2/3)

```
public void onClickCheck(View view) {
  if (app.getGame().endOfGame()) {
    Intent res = new Intent();
    res.putExtra("color", app.getGame().getStartColor());
    setResult(RESULT_OK, res);
    finish();
  }
}
```

GameActivity (3/3)

```
public void onClickRedo(View view) {
  app = (TheApplication)this.getApplication();
  app.getGame().resetColor();
}

public void onClickCancel(View view) {
  setResult(RESULT_CANCELED);
  finish();
}
```