Créer un menu pour une activité

Créer un menu pour une activité

- Tous les modèles Android possèdent un **bouton Menu.** Grâce à ce bouton, il vous est possible de proposer à vos utilisateurs des **fonctionnalités supplémentaires n'apparaissant pas par défaut à l'écran** et d'ainsi mieux gérer la taille limitée de l'écran d'un appareil mobile.

- Chaque menu est propre à une activité, c'est pourquoi toutes les opérations que nous allons traiter dans cette partie se réfèrent à une activité. Pour proposer plusieurs menus vous aurez donc besoin de le faire dans chaque activité de votre application.

Création d'un menu

- Pour créer un menu il vous suffit de surcharger la méthode **onCreateOptionsMenu** de la classe Activity.

- Cette méthode est appelée la première fois que l'utilisateur appuie sur **le bouton menu** de son téléphone.

- Elle reçoit en paramètre **un objet de type Menu** dans lequel nous ajouterons nos éléments ultérieurement.

Création d'un menu

- Si l'utilisateur appuie une seconde fois sur le bouton Menu, cette méthode ne sera plus appelée tant que l'activité ne sera pas **détruite puis recréée.**

- Si vous avez besoin d'ajouter, de supprimer ou de modifier un élément du menu après coup, il vous faudra surcharger une autre méthode, la méthode **onPrepareOptionsMenu** que nous aborderons plus tard dans cette partie.

Création d'un menu

- Pour créer un menu, commencez par créer un fichier de définition d'interface nommé **main.xml**:

Code 5-18 : Définition de l'interface de l'exemple avec menu

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content">
   <TextView
      android:id="@+id/TextView01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Appuyez sur la touche menu">
   </TextView>
</LinearLayout>
```

Code 5-19 : Création d'un menu

```
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class Main extends Activity {

   // Pour faciliter la gestion des menus
   // nous créons des constantes
   private final static int MENU_PARAMETRE = 1;
   private final static int MENU_QUITTER = 2;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Nous créons un premier menu pour accéder aux paramètres.
      // Pour ce premier élément du menu, nous l'agrémentons
      // d'une image.
      menu.add(0, MENU_PARAMETRE, Menu.NONE, "Paramètres").setIcon(R.drawable.icon);
      // Nous créons un second élément.
      // Celui-ci ne comportera que du texte.
      menu.add(0, MENU_QUITTER, Menu.NONE, "Quitter");

      return true;
   }
```

Création d'un menu

menu.add(groupID, itemID, ordre, "nom du menu") ;

- Cette méthode permet de créer un nouveau menu. Les paramètres nécessaires sont les suivants :

- **groupID :** identifiant du groupe. Il est possible de regrouper les éléments,

- **itemID :** identifiant de ce menu. Il nous sera utilisé pour identifier ce menu parmi les autres. On doit donner un identifiant différent à chaque menu (1, 2, 3… par exemple).

- **ordre :** associer un ordre d'affichage au menu. On donnera ici la valeur 0.

- Nom du menu : chaîne qui représente le titre du menu.

Création d'un menu

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   switch (item.getItemId()) {
   case MENU_PARAMETRE:
      // Lorsque l'utilisateur cliquera sur le menu 'paramètres',
      // un message s'affichera.
      Toast.makeText(Main.this, "Ouverture des paramètres",
                     Toast.LENGTH_SHORT).show();
      // onOptionsItemSelected retourne un booléen,
      // nous lui envoyons la valeur "true" pour signaler
      // que tout s'est correctement déroulé.
      return true;

      case MENU_QUITTER:
         // Lorsque l'utilisateur cliquera sur le menu 'Quitter',
         // nous fermerons l'activité avec la méthode finish().
         finish();
         return true;
      default:
         return true;
      }
   }
}
```

Création d'un menu

- Dans le code précédent, nous avons surchargé deux méthodes, l'une responsable de la création du menu de l'activité, l'autre de la logique lors de la sélection par l'utilisateur d'un élément du menu.

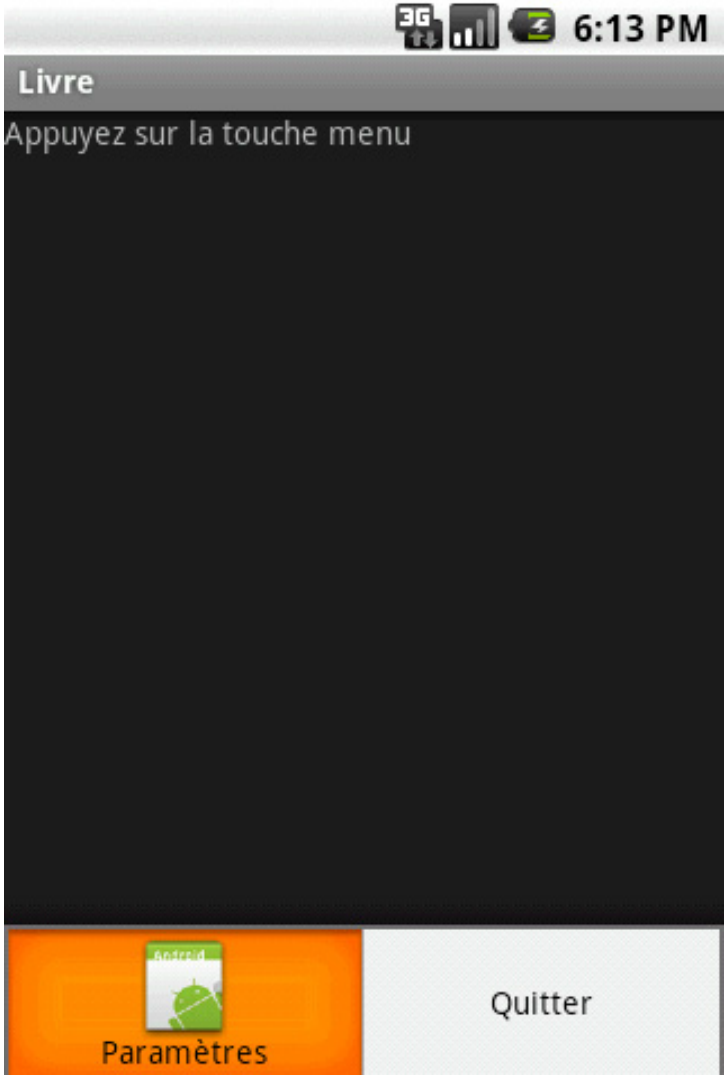
- La méthode **onCreateOptionsMenu** sera appelée uniquement la première fois lorsque l'utilisateur appuiera sur le bouton **Menu.**

- Dans cette méthode, nous créons deux éléments de menu **Paramètres** et **Quitter.**

- La méthode **onOptionsItemSelected** est appelée lorsque l'utilisateur clique sur l'un des éléments du menu.

- Dans notre cas, si l'utilisateur clique sur l'élément **Paramètres** l'application affiche **un message**, alors qu'un clic sur **Quitter** ferme l'activité grâce à la méthode **finish.**

Création d'un menu



Mettre à jour dynamiquement un menu

- Il vous sera peut-être nécessaire de changer l'intitulé, de supprimer ou de rajouter un élément du menu en cours de fonctionnement.

- La méthode **onCreateOptionsMenu** n'étant appelée qu'une fois, la première fois où l'utilisateur clique sur le bouton Menu, vous ne pourrez pas mettre à jour votre menu dans cette méthode.

- Pour modifier le menu après coup, par exemple dans le cas où votre menu propose de s'authentifier et qu'une fois authentifié, vous souhaitez proposer à l'utilisateur de se déconnecter, vous devrez surcharger la méthode **onPrepareOptionsMenu.**

Mettre à jour dynamiquement un menu

- Un objet de type **Menu** est envoyé à cette méthode qui est appelée à chaque fois que l'utilisateur cliquera sur le bouton **Menu**.

- Modifiez le code 5-19 pour rajouter la redéfinition de la méthode **onPrepareOptionsMenu :**

Code 5-20 : Modification dynamique du menu

```
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    // Nous modifions l'intitulé de notre premier menu
    // Pour l'exemple nous lui donnerons un intitulé
    // différent à chaque fois que l'utilisateur cliquera
    // sur le bouton menu à l'aide de l'heure du système
    menu.getItem(0).setTitle(SystemClock.elapsedRealtime()+"");

    return super.onPrepareOptionsMenu(menu);
}
```

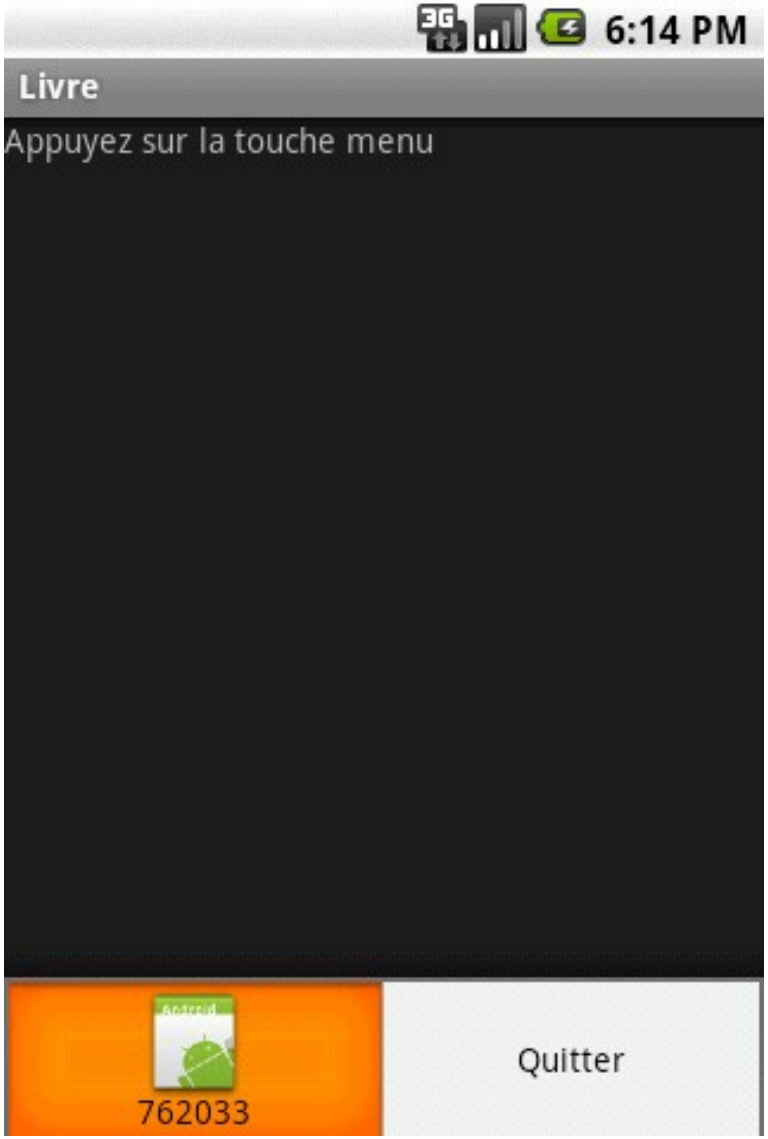
Mettre à jour dynamiquement un menu

- Dans cet exemple, lorsque l'utilisateur appuie sur le bouton **Menu**, l'intitulé du premier menu affiche une valeur différente à chaque fois.

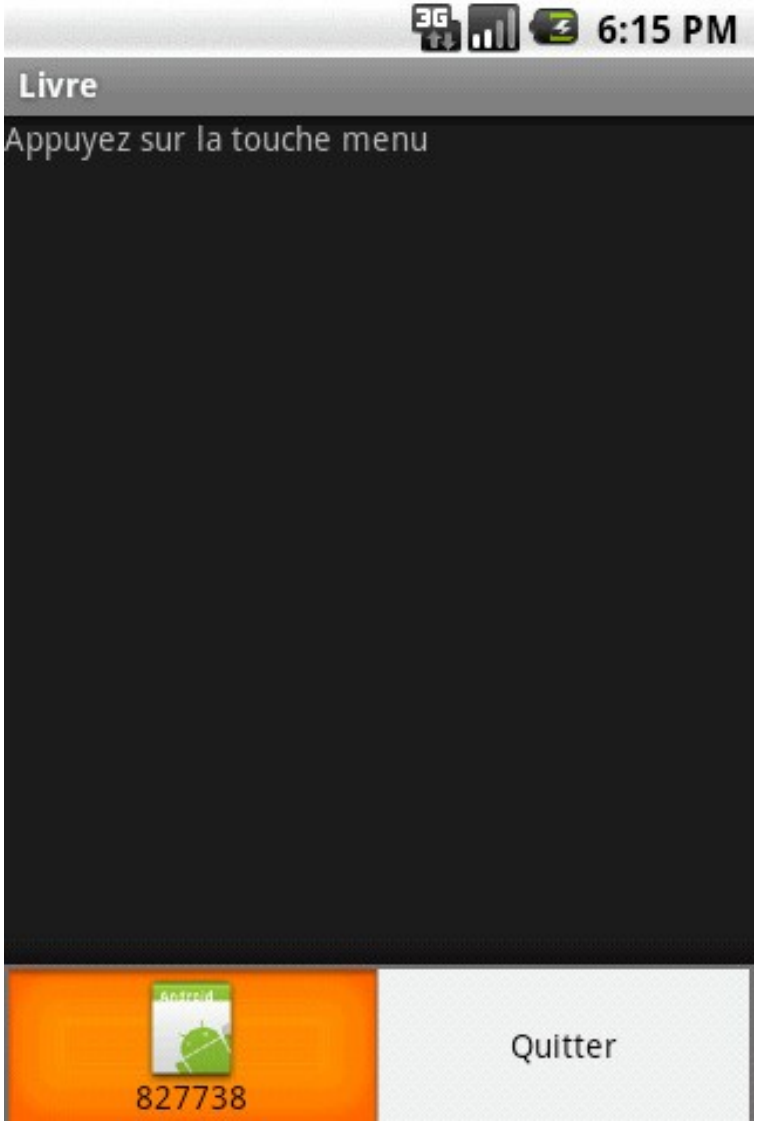
- Vous pourrez placer à cet endroit le code nécessaire pour adapter l'intitulé que vous souhaitez afficher à l'utilisateur s'il est nécessaire de changer le menu.

- Si l'utilisateur appuie à nouveau, l'intitulé changera de concert. S'il rappuie encore, l'intitulé changera de même.

Mettre à jour dynamiquement un menu



Mettre à jour dynamiquement un menu



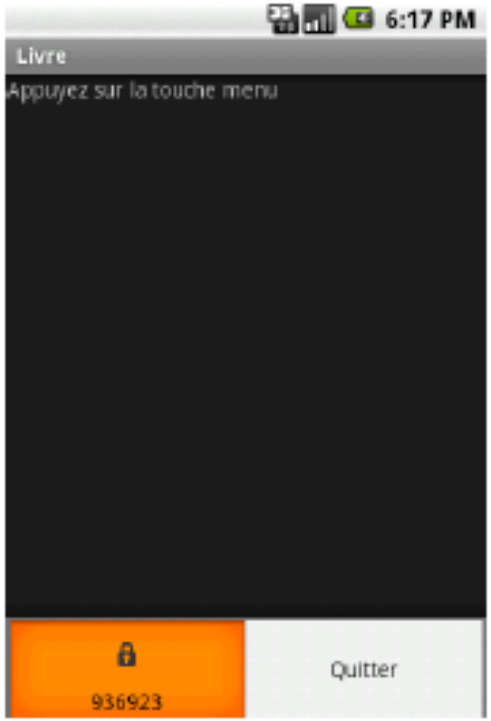
Mettre à jour dynamiquement un menu

- De la même façon, vous pouvez changer l'image du menu comme ceci :

Code 5-21 : Changer l'image d'un élément de menu

```
menu.getItem(0).setIcon(android.R.drawable.icon_secure);
```

Figure 5–15
L'icône d'un élément du menu changée dynamiquement



Créer des sous-menus

- Créer des sous-menus peut se révéler utile pour proposer plus d'options sans encombrer l'interface utilisateur.

- Pour ajouter un sous-menu, vous devrez ajouter un menu de type **SubMenu** et des éléments le composant.

- Le paramétrage est le même que pour un menu, à l'exception de l'image : vous ne pourrez pas ajouter d'image sur les éléments de vos sous-menus.

- Néanmoins, il sera possible **d'ajouter une image dans l'entête du sous-menu.**

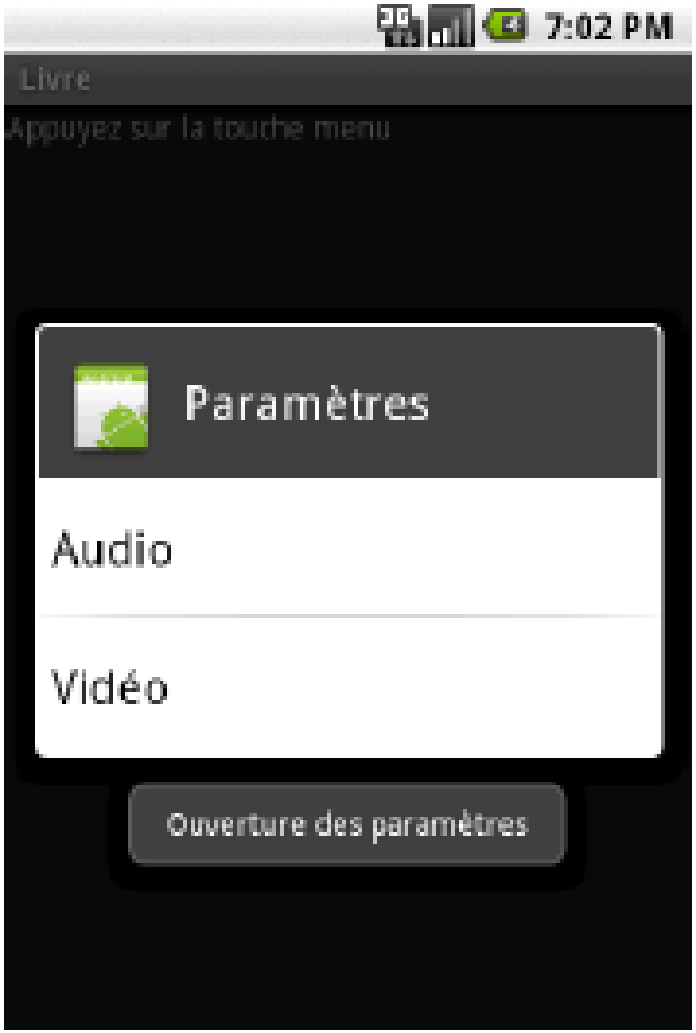
- Remplacez la méthode **onCreateOptionsMenu** du code 5-19 par celle-ci :

Créer des sous-menus

Code 5-22 : Création de sous-menus

```
private final static int SOUSMENU_VIDEO= 1000;
private final static int SOUSMENU_AUDIO= 1001;

...

@Override
public boolean onCreateOptionsMenu(Menu menu) {
   // Nous créons un premier menu sous forme de sous-menu
   // Les sous menu associés à ce menu seront ajoutés à ce sous-menu
   SubMenu sousMenu = menu.addSubMenu(0, MENU_PARAMETRE, Menu.NONE,
                                      "Paramètres").setIcon(R.drawable.icon);
   // Nous ajoutons notre premier sous-menu
   sousMenu.add(0, SOUSMENU_AUDIO, Menu.NONE, "Audio").setIcon(R.drawable.icon);
   // Nous ajoutons notre deuxième sous-menu
   sousMenu.add(0, SOUSMENU_VIDEO, Menu.NONE, "Vidéo");

   // Il est possible de placer une
   // image dans l'entête du sous-menu
   // Il suffit de le paramétrer avec la méthode
   // setHeaderIcon de notre objet SubMenu
   sousMenu.setHeaderIcon(R.drawable.icon);

   // Nous créons notre deuxième menu
   menu.add(0, MENU_QUITTER, Menu.NONE, "Quitter");

   return true;
}
```

Créer des sous-menus



Créer des sous-menus

- Lorsque l'utilisateur clique sur le bouton Menu, le menu qui s'affiche est identique à l'exemple précédent.

- Néanmoins en cliquant sur le menu **Paramètres** un sous-menu s'affiche sous forme de liste.

Créer un menu contextuel

Créer un menu contextuel

- La plate-forme Android autorise également la création des menus contextuels, autrement dit de menus dont le contenu change en fonction du contexte.

- Sous Android, l'appel d'un menu contextuel se fait lorsque l'utilisateur effectue un clic de quelques secondes sur un élément d'interface graphique.

- Les menus contextuels fonctionnent d'ailleurs de façon similaire aux sous-menus.

Créer un menu contextuel

- La création d'un menu contextuel se fait en deux étapes :
  - tout d'abord la création du menu proprement dit, puis
  - son enregistrement auprès de la vue, avec la méthode **registerForContextMenu** ; en fournissant en paramètre la **vue concernée.**

Créer un menu contextuel

- Concrètement, redéfinissez la méthode **onCreateContextMenu** et ajoutez-y les menus à afficher en fonction de la vue sélectionnée par l'utilisateur.

- Pour sélectionner les éléments qui s'y trouveront, redéfinissez la méthode **onContextItemSelected.**

Créer un menu contextuel

- Créez un nouveau fichier de définition d'une interface comme ceci :

Code 5-23 : Définition de l'interface de l'exemple avec menu contextuel

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical"
   android:gravity="center_vertical|center_horizontal">
   <TextView
      android:id="@+id/monTexteContextMenu"
      android:layout_height="wrap_content"
      android:text="Cliquez ici 3 sec"
      android:textSize="30dip"
      android:layout_width="fill_parent"
      android:gravity="center_horizontal">
   </TextView>
</LinearLayout>
```

Créer un menu contextuel

Code 5-24 : Création d'un menu contextuel

```
...
import android.view.ContextMenu;
import android.view.MenuItem;
import android.view.ContextMenu.ContextMenuInfo;
...

   // Pour faciliter notre gestion des menus
   // nous créons des variables de type int
   // avec des noms explicites qui nous aideront
   // à ne pas nous tromper de menu
   private final static int MENU_CONTEXT_1= 1;
   private final static int MENU_CONTEXT_2= 2;


   @Override
   public void onCreate(Bundle savedInstanceState) {
      ...

      // Nous enregistrons notre TextView pour qu'il réagisse
      // au menu contextuel
      registerForContextMenu((TextView)findViewById(R.id.monTexteContextMenu));
   }
```

Créer un menu contextuel

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
      ContextMenuInfo menuInfo) {
   // Nous vérifions l'identifiant de la vue.
   // Si celle-ci correspond à une vue pour laquelle nous souhaitons
   // réagir au clic long, alors nous créons un menu.
   // Si vous avez plusieurs vues à traiter dans cette méthode,
   // il vous suffit d'ajouter le code nécessaire pour créer les
   // différents menus.
   switch (v.getId()) {
   case R.id.monTexteContextMenu:
      {
            menu.setHeaderTitle("Menu contextuel");
            menu.setHeaderIcon(R.drawable.icon);
            menu.add(0, MENU_CONTEXT_1, 0, "Mon menu contextuel 1");
            menu.add(0, MENU_CONTEXT_2, 0, "Mon menu contextuel 2");
            break;
         }
   }
   super.onCreateContextMenu(menu, v, menuInfo);
}
```

Créer un menu contextuel

```
    @Override
    public boolean onContextItemSelected(MenuItem item) {
       // Grâce à l'identifiant de l'élément
       // sélectionné dans notre menu nous
       // effectuons une action adaptée à ce choix
       switch (item.getItemId()) {
       case MENU_CONTEXT_1:
          {
                Toast.makeText(Main.this, "Menu contextuel 1 cliqué !",
                               Toast.LENGTH_SHORT).show();
                break;
          }
       case MENU_CONTEXT_2:
          {
                Toast.makeText(Main.this, "Menu contextuel 2 cliqué !",
                               Toast.LENGTH_SHORT).show();
                break;
          }
       }
       return super.onContextItemSelected(item);
    }
}
```

Créer un menu contextuel

- Si l'utilisateur clique pendant trois secondes sur la vue **TextView** centrale, le menu contextuel s'affiche.

Créer un menu contextuel

Créer un menu contextuel

Pour insérer une image dans le gabarit de mise en page de l'interface, ajoutez les lignes suivantes dans le fichier XML après la balise <TextView> :

Code 5-25 : Ajout d'une image dans le menu contextuel de l'exemple

```
...
</TextView>
<ImageView
   android:id="@+id/ImageView01"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:src="@drawable/icon">
</ImageView>
...
```

Créer un menu contextuel

- À ce stade, si l'utilisateur clique quelques secondes sur l'image, il ne se produira rien car la vue de l'image n'a pas enregistré de menu contextuel.

- Vous devez ajouter la ligne suivante dans la méthode **onCreate** de votre fichier main.java :

Code 5-26 : Enregistrer le contrôle de l'image pour réagir au menu contextuel

```
registerForContextMenu((TextView)findViewById(R.id.monTexteContextMenu));
...
registerForContextMenu((ImageView)findViewById(R.id.monImageContext));
```

Créer un menu contextuel

- L'image est désormais enregistrée et un clic dessus déclenchera l'apparition du menu contextuel. Il reste à adapter les méthodes **onCreateContextMenu** et **onContextItemSelected** pour que les menus contextuels destinés à cette image s'affichent correctement :

Code 5-27 : Modification du code 5-24 pour gérer le menu contextuel de l'image

```
...

private final static int MENU_CONTEXT_IMAGE_1= 3;
private final static int MENU_CONTEXT_IMAGE_2= 4;

@Override
public void onCreateContextMenu(ContextMenu menu, View v,
      ContextMenuInfo menuInfo) {
   // Nous vérifions l'identifiant de la vue envoyée.
   // Si celle-ci correspond à celle que nous souhaitons,
   // nous créons nos menus.
   // Si vous avez plusieurs vues à faire réagir, il vous suffira
   // d'ajouter les conditions et le code nécessaire
   // pour afficher ces même menus ou d'autre menus.
   switch (v.getId()) {
   case R.id.monTexteContextMenu:
      {
         menu.setHeaderTitle("Menu contextuel");
         menu.setHeaderIcon(R.drawable.icon);
         menu.add(0, MENU_CONTEXT_1, 0, "Mon menu contextuel 1");
         menu.add(0, MENU_CONTEXT_2, 0, "Mon menu contextuel 2");
         break;
      }
   case R.id.monImageContext:
      {
         menu.setHeaderTitle("Menu contextuel Image");
         menu.setHeaderIcon(R.drawable.icon);
         menu.add(0, MENU_CONTEXT_IMAGE_1, 0, "Mon menu contextuel IMAGE 1");
         menu.add(0, MENU_CONTEXT_IMAGE_2, 0, "Mon menu contextuel IMAGE 2");
         break;
      }
   }
   super.onCreateContextMenu(menu, v, menuInfo);
}

@Override
public boolean onContextItemSelected(MenuItem item) {
   // Grâce à l'identifiant de l'élément
   // sélectionné dans notre menu nous
   // effectuons une action adaptée à ce choix
   switch (item.getItemId()) {
   case MENU_CONTEXT_1:
      {
```

```
            Toast.makeText(Main.this, "Menu contextuel 1 cliqué !",
                           Toast.LENGTH_SHORT).show();
            break;
        }
    case MENU_CONTEXT_2:
        {
            Toast.makeText(Main.this, "Menu contextuel 2 cliqué !",
                           Toast.LENGTH_SHORT).show();
            break;
        }
    case MENU_CONTEXT_IMAGE_1:
        {
            Toast.makeText(Main.this, "Menu contextuel IMAGE 1 cliqué !",
                           Toast.LENGTH_SHORT).show();
            break;
        }
    case MENU_CONTEXT_IMAGE_2:
        {
            Toast.makeText(Main.this, "Menu contextuel IMAGE 2 cliqué !",
                          Toast.LENGTH_SHORT).show();
            break;
        }
    }
    return super.onContextItemSelected(item);
}

...
```

Créer un menu contextuel

- Désormais, en cliquant quelques secondes sur l'image, un menu contextuel dédié s'affichera et les clics de l'utilisateur sur les éléments de ce menu seront gérés.

Notifier l'utilisateur par le mécanisme des « toasts »

toasts

- Notifier un utilisateur ne signifie pas le déranger alors qu'il utilise une autre application que la vôtre ou qu'il rédige un SMS. À cette fin,

- la plate-forme Android propose le concept de toast, ou message non modal s'affichant **quelques secondes** tout au plus.

- De cette façon, ni l'utilisateur ni l'application **active** à ce moment ne sont **interrompus.**

toasts

- L'utilisation d'un toast est rendue aisée par les méthodes statiques de **la classe Toast** et notamment **makeText.**

- Il vous suffit de passer le **Context** de l'application, **le texte** et **la durée d'affichage** pour créer une instance de Toast que vous pourrez afficher avec la méthode **show** autant de fois que nécessaire.

toasts

Code 11-17 : Création d'un toast

```
// La chaîne représentant le message
String message = "Vous prendrez bien un toast ou deux ?";
// La durée d'affichage (LENGTH_SHORT ou LENGTH_LONG)
int duree= Toast.LENGTH_LONG;
```

```
// Création du Toast (le contexte est celui de l'activité ou du service)
Toast toast = Toast.makeText(this, message, duree);
// Affichage du Toast
toast.show();
```

La durée du toast ne peut être librement fixée : elle peut soit prendre la valeur LENGTH_SHORT (2 secondes) ou LENGTH_LONG (5 secondes).

L'extrait de code précédent affiche le résultat suivant :

toasts

Figure 11–2
Affichage d'un toast s'exécutant depuis un service : l'utilisateur est notifié sans entraver son utilisation.

Positionner un toast

- Par défaut, Android affiche le message en **bas de l'écran** de l'utilisateur.
- Ce comportement peut être redéfini pour en changer la position.
- Vous pouvez spécifier la disposition d'un toast en spécifiant son ancrage sur l'écran ainsi qu'un décalage sur l'axe horizontal et vertical.

Positionner un toast

Code 11-18 : Positionner un toast

```
// Création du Toast
Toast toast = Toast.makeText(this, "Vous prendrez bien un toast ou deux ;",
Toast.LENGTH_LONG);
// Spécifie la dispostion du Toast sur l'écran
toast.setGravity(Gravity.TOP, 0, 40);
// Affichage du Toast
toast.show();
```

- Le code précédent affiche le message en haut de l'écran avec un décalage vertical.

Figure 11–3
Modifiez la gravité du toast pour le positionner à un endroit approprié.

- La position du toast est bien sûr fonction de l'importance du message.
- Affiché plus haut sur l'écran, le message aura plus de chance d'attirer l'attention de l'utilisateur que s'il se trouve tout en bas.

Personnaliser l'apparence d'un toast

- Par défaut, Android utilise **une fenêtre grisée** pour afficher le texte du toast à l'utilisateur.
- Mais ce comportement n'est pas toujours le plus adapté pour communiquer des informations complexes à l'utilisateur.
- Dans certains cas, un affichage **graphique** sera plus approprié qu'un affichage **textuel.**

Personnaliser l'apparence d'un toast

- Sachez qu'Android est capable d'utiliser une vue personnalisée conçue pour l'occasion, en lieu et place du « **carré grisâtre aux bords arrondis** » par défaut.

- Pour spécifier cette vue à afficher, utilisez la méthode **setView** de **l'objet Toast.**

- Même si vous ne souhaitez pas afficher autre chose que du texte, le simple fait de personnaliser le design de la vue vous permettra de **différencier les messages émis par votre application des autres,** et pour l'utilisateur de **distinguer clairement l'origine des messages.**