


Android

Interagindo com o Google Maps



Tópicos Abordados




- Google Maps Android API v2
- Configurando a aplicação
- Mapas
- Marcadores
- Formas geométricas

Google Maps API v2



- A Google Maps API permite integrar aplicativos do Android com o Google Maps
 - Posicionamento
 - Zoom
 - Desenhos
 - etc.
- O dispositivo precisa ter o Google Play Services e o Google Maps instalado para que a visualização dos mapas funcione

Criando um Mapa


- Um mapa deve ser exibido a partir de um fragment

```

<fragment
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.MapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

```

API Level >= 11

```

<fragment
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

```

API Level < 11

Referenciando um Mapa


- Um mapa é normalmente referenciado a partir de uma activity ou fragment
- Um objeto **GoogleMap** é usado para interagir com o mapa

```

protected void onCreate(Bundle savedInstanceState) {
    ...
    FragmentManager fm = getSupportFragmentManager();
    ((MapFragment) fm.findFragmentById(R.id.map)).getMapAsync(this);
}

```


Implementa OnMapReadyCallback

```

public void onMapReady(GoogleMap googleMap) {
    ...
}

```

Para versões anteriores à API Level 11, usar
getSupportFragmentManager()

Exibição do Mapa




Tipos de Mapas



- 6 tipos de mapas podem ser exibidos
 - Representados por constantes na classe *GoogleMap*

Tipo	O que mostra	Constante
Normal	Ruas e estradas	MAP_TYPE_NORMAL
Hybrid	Ruas, estradas e imagens de satélite	MAP_TYPE_HYBRID
Satellite	Imagens de satélite	MAP_TYPE_SATELLITE
Terrain	Topografia do terreno	MAP_TYPE_TERRAIN
None	Sem informações	MAP_TYPE_NONE

Tipos de Mapas



- O método **setMapType()** é utilizado para definir o tipo de mapa a ser mostrado

```
map.setMapType(GoogleMap.MAP_TYPE_NORMAL);
```



Objeto *GoogleMap*

Trabalhando com a Câmera



- Para entender como funciona a visualização de um mapa, imagine que você está vendo o mapa através de uma câmera
- Os parâmetros desta câmera definem como o mapa será visto
 - Target (localização)
 - Zoom
 - Bearing (orientação)
 - Tilt (ângulo de visualização)

Target



- A localização da câmera é dada em termos de latitude e longitude
- Esta informação é encapsulada em um objeto da classe **LatLng**

```
LatLng loc = new LatLng(40.700126, -74.013926);
```

- Um objeto **CameraUpdate** atualiza a posição da câmera

```
CameraUpdate update = CameraUpdateFactory.newLatLng(loc);
```

```
map.moveCamera(update);  
map.animateCamera(update);
```

Zoom



- O zoom permite aproximar ou afastar a câmera do mapa
- O zoom é representado por um número inteiro
 - Quanto maior o número, mais perto do mapa
- Um **CameraUpdate** é usado para alterar o zoom

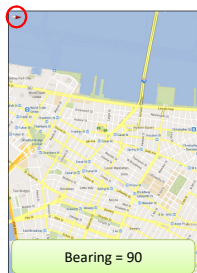
```
CameraUpdate update = CameraUpdateFactory.zoomIn();  
CameraUpdate update = CameraUpdateFactory.zoomOut();  
CameraUpdate update = CameraUpdateFactory.zoomTo(10);  
CameraUpdate update = CameraUpdateFactory.zoomBy(2);
```

```
map.moveCamera(update);  
map.animateCamera(update);
```

Bearing



- A orientação é dada em graus a partir do norte, com os valores aumentando no sentido horário



Bearing



- A alteração da orientação é feita através do uso de um objeto **CameraUpdate**

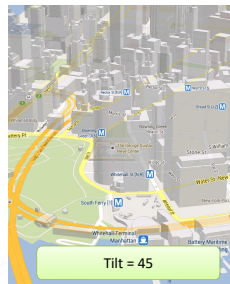
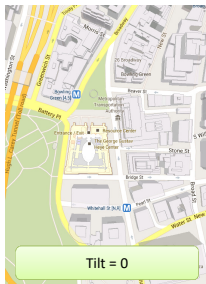
```
CameraPosition.Builder builder = CameraPosition.builder();  
CameraPosition pos = builder.bearing(90).build();  
CameraUpdate update = CameraUpdateFactory.newCameraPosition(pos);
```

```
map.moveCamera(update);  
map.animateCamera(update);
```

Tilt



- O ângulo de visualização é aproximação ou afastamento da câmera da superfície da Terra



Tilt



- A alteração do ângulo de visualização é feita através do uso de um objeto **CameraUpdate**

```
CameraPosition.Builder builder = CameraPosition.builder();  
CameraPosition pos = builder.tilt(45).build();  
CameraUpdate update = CameraUpdateFactory.newCameraPosition(pos);
```

```
map.moveCamera(update);  
map.animateCamera(update);
```

Mais sobre o CameraUpdate



- Um objeto **CameraUpdate** pode ser usado para definir todos os parâmetros da câmera

```
CameraPosition.Builder builder = CameraPosition.builder()
    .target(loc)
    .zoom(5)
    .bearing(90)
    .tilt(30);

CameraUpdate update =
    CameraUpdateFactory.newCameraPosition(builder.build());
map.moveCamera(update);
```

Animação de Câmera



- A forma mais fácil de animar a câmera para uma nova posição é o método **animateCamera()**

```
map.animateCamera(update);
```

- Com este método, é possível também controlar a animação

```
map.animateCamera(update, 5000, callback);
```

Duração da
animação (em ms)

Objeto de callback

Animação de Câmera



- O objeto de callback deve ser de uma classe que implementa a **CancelableCallback**

```
public class MyCallback implements CancelableCallback {
    public void onCancel() {
    }

    public void onFinish() {
    }
}
```

- Métodos chamados pelo Android
 - **onCancel()**: animação foi cancelada
 - **onFinish()**: animação terminou

Controles no Mapa e Gestos



- O método **getUISettings()** permite habilitar ou desabilitar controles do mapa

```
UISettings settings = map.getUISettings();
settings.setZoomControlsEnabled(true);
settings.setCompassEnabled(true);
```

- Com esta mesma abordagem é possível também habilitar ou desabilitar gestos

```
settings.setZoomGesturesEnabled(true);
settings.setRotateGesturesEnabled(true);
settings.setScrollGesturesEnabled(true);
settings.setTiltGesturesEnabled(true);
settings.setAllGesturesEnabled(true);
```

Eventos em Mapas



- Um objeto **OnMapClickListener** pode ser chamado quando ocorre um toque no mapa

```
public class MainActivity extends FragmentActivity
    implements OnMapClickListener {

    public void onMapClick(LatLng pos) {
        //...
    }
}
```

Implementa
OnMapClickListener

Chamado quando há
um toque no mapa

- Registro do listener

```
map.setOnMapClickListener(listener);
```

Eventos em Mapas



- Um objeto **OnMapLongClickListener** pode ser chamado quando ocorre um toque longo no mapa

```
public class MainActivity extends FragmentActivity
    implements OnMapLongClickListener {

    public void onMapLongClick(LatLng pos) {
        //...
    }
}
```

Implementa
OnMapLongClickListener

Chamado quando há um
toque longo no mapa

- Registro do listener

```
map.setOnMapLongClickListener(listener);
```

Eventos de Câmera



- Um objeto **OnCameraMoveListener** pode ser chamado quando um parâmetro da câmera muda

```
public class MainActivity extends FragmentActivity
    implements OnCameraMoveListener {

    public void onCameraMove() {
        CameraPosition c = map.getCameraPosition();
        LatLng target = c.target;
        float zoom = c.zoom;
        float bearing = c.bearing;
        float tilt = c.tilt;
    }
}
```

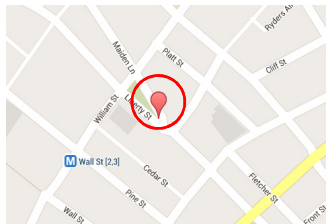
- Registro do listener

```
map.setOnCameraMoveListener(listener);
```

Markers



- Marcadores permitem destacar alguma posição no mapa
- São customizáveis
- São interativos



Criando Markers



- Primeiro é preciso criar um objeto **MarkerOptions** e depois chamar **addMarker()**

```
MarkerOptions options = new MarkerOptions();
options.position(new LatLng(40.711, -74.002));
Marker marker = map.addMarker(options);
```

A posição do marcador é obrigatória

- Outras customizações

```
MarkerOptions options = new MarkerOptions()
    .position(new LatLng(40.711, -74.002))
    .title("Meu marcador")
    .snippet("Este é o marcador que eu criei")
    .icon(BitmapDescriptorFactory.fromResource(
        android.R.drawable.star_big_on));
map.addMarker(options);
```

Criando Markers

Softblue

Markers de Outras Cores

Softblue

- É possível usar o ícone padrão do marcador, mas variar a sua cor

```

MarkerOptions options1 = new MarkerOptions()
    .position(new LatLng(40.711, -74.002))
    .icon(BitmapDescriptorFactory.defaultMarker(
        BitmapDescriptorFactory.HUE_GREEN));
map.addMarker(options1);

MarkerOptions options2 = new MarkerOptions()
    .position(new LatLng(40.713, -74.00
    .icon(BitmapDescriptorFactory.defaultMarker(
        BitmapDescriptorFactory.HUE_BLUE));
map.addMarker(options2);

```

Eventos em Markers: Click

Softblue

- Um objeto **OnMarkerClickListener** pode ser chamado quando o marcador é clicado

```

public class MainActivity extends FragmentActivity
    implements OnMarkerClickListener {

    public boolean onMarkerClick(Marker marker) {
        //...
    }
}

```

Implementa OnMarkerClickListener

Chamado quando há um clique no marcador

- Registro do listener

```
map.setOnMarkerClickListener(listener);
```

Eventos em Markers: Click



- **onMarkerClick()** retorna um booleano

```
public boolean onMarkerClick(Marker marker) {}
```

- O clique em um marcador tem um comportamento padrão
 - A câmera é posicionada no local do marcador
 - O título do marcador é exibido
- O retorno do método indica se o comportamento padrão será executado (false) ou não (true)

Eventos em Markers: Drag



- Um objeto **OnMarkerDragListener** pode ser chamado quando o marcador é arrastado

```
public class MainActivity extends FragmentActivity  
implements OnMarkerDragListener {  
  
    public void onMarkerDragStart(Marker marker) {  
    }  
    public void onMarkerDrag(Marker marker) {  
    }  
    public void onMarkerDragEnd(Marker marker) {  
    }  
}
```

- Registro do listener

```
map.setOnMarkerDragListener(listener);
```

Eventos em Markers: Drag



- Para que os eventos de drag funcionem, o marcador deve ser "arrastável"

```
options.draggable(true);
```

- Por padrão, o marcador é fixo


Minha Localização

Softblue

- O mapa pode exibir a sua localização

```
// Habilita a checagem de localização
map.setMyLocationEnabled(true);

// Mostra o botão de localização
map.getUiSettings().setMyLocationButtonEnabled(true);
```




Formas Geométricas em Mapas


Softblue

- A API do Google Maps permite que sejam desenhadas formas geométricas sobre os mapas
- Podem ser de três tipos


Polyline



Polygon



Circle

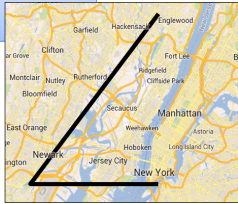


Polyline

Softblue

- A criação de polylines é feita usando a classe **PolylineOptions** e o método **addPolyline()**

```
PolylineOptions options = new PolylineOptions()
    .add(new LatLng(40.7, -74.0))
    .add(new LatLng(40.7, -74.2))
    .add(new LatLng(40.9, -74.0));
map.addPolyline(options);
```

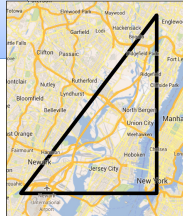


Polygon



- A criação de polígonos é feita usando a classe **PolygonOptions** e o método **addPolygon()**

```
PolygonOptions options = new PolygonOptions()
    .add(new LatLng(40.7, -74.0))
    .add(new LatLng(40.7, -74.2))
    .add(new LatLng(40.9, -74.0));
map.addPolygon(options);
```

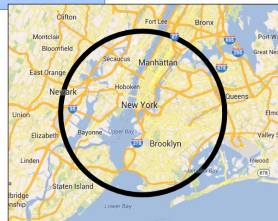


Circle



- A criação de círculos é feita usando a classe **CircleOptions** e o método **addCircle()**

```
CircleOptions options = new CircleOptions()
    .center(new LatLng(40.7, -74.0))
    .radius(15000);
map.addCircle(options);
```



Propriedades de Desenho



- **Stroke color**
 - Define a cor da linha (preto é a cor padrão)
 - Métodos
 - *PolylineOptions.color()*
 - *PolygonOptions.strokeColor()*
 - *CircleOptions.strokeColor()*
 - Ex: *strokeColor(Color.BLUE)*

Propriedades de Desenho



• Fill color

- Define a cor de preenchimento (a cor padrão é transparente)
- Métodos
 - *PolygonOptions.fillColor()*
 - *CircleOptions.fillColor()*
- Ex: *fillColor(Color.GREEN)*

Propriedades de Desenho



• Stroke width

- Define a espessura da linha (a espessura padrão é 10 pixels)
- Métodos
 - *PolylineOptions.width()*
 - *PolygonOptions.strokeWidth()*
 - *CircleOptions.strokeWidth()*
- Ex: *strokeWidth(5)*

Propriedades de Desenho



• Z Index

- Define a ordem com relação a outros desenhos no mapa
- Elementos com Z index maior são desenhados sobre elementos com Z index menor
- Métodos
 - *PolylineOptions.zIndex()*
 - *PolygonOptions.zIndex()*
 - *CircleOptions.zIndex()*
- Ex: *zIndex(1)*

- **Visibility**

- Define se a figura é visível ou não (o padrão é ser visível)

- Métodos

- *PolylineOptions.visible()*
 - *PolygonOptions.visible()*
 - *CircleOptions.visible()*

- Ex: *visible(true)*