Jibestream`s Android SDK

Quick Overview

Introduction

The Jibestream Android SDK is engineered utilizing the canvas technology in order to achieve the best performance and usability on the constrained hardware environment that the platform run onto.
Thus a prerequisite for efficiently utilizing the SDK must have some understanding of the Canvas.

SDK inter communication – Eventbus

The chosen method for communication among the different components is the android's internal dispatch mechanism LocalBroadcastManager.
The developer can obtain a reference to the instance with the following code:

```
LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(context);
```

http://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.htm (LocalBroadcastManager)

SDK Overview

To successfully display a map the following outlines the steps involved.

1. Obtain the data for a map.
2. Instantiate a map.
3. Create an instance of the map`s viewport and connect it to map instance.

Getting venue's data

In order to populate a map we must first get its data from the backend, we can achieve this by invoking the static method VenueDataService.getData() and by listening on the platform's LocalBroadcastManager event bus for the VenueDataService.DONE message.
On a successfully response the SDK will store the data and provide the filename within the bundle of the Intent.
Example snippet provided bellow.

```
  BasicAuthentication basicAuthentication = new
BasicAuthentication("userValue","passcodeValue","apiValue","languageCodeValue")
;
  LocalBroadcastManager.getInstance(this).registerReceiver(broadcastReceiver,
new IntentFilter(VenueDataService.DONE));
  LocalBroadcastManager.getInstance(this).registerReceiver(broadcastReceiver,
new IntentFilter(VenueDataService.ERROR));
  VenueDataService.getData(context, url, projectId, basicAuthentication);

  private BroadcastReceiver broadcastReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {

LocalBroadcastManager.getInstance(context).unregisterReceiver(broadcastReceiver
);
      String action = intent.getAction();
      switch (action) {
        case VenueDataService.DONE:
          // Get the filename with the data
          String filename =
intent.getStringExtra(VenueDataService.KEY_FILENAME);
          M m = binder.getM();

LocalBroadcastManager.getInstance(context).registerReceiver(broadcastReceiverFr
ameworkInit, new IntentFilterEngine(m, IntentFilterEngine.COMPLETE));
          m.start(filename, R.raw.config);
          //
          break;
        case VenueDataService.ERROR:
          // handle error
          break;
      }
    }
  };
```

The 3 VenueDataService.getData() parameters are:

- *String*: The server endpoint where you will get all your data from.
- *int*: The location/project ID of the map/building to display.
- *BasicAuthentication*: BasicAuthentication instance if applicable, can be null if there is no authentication in place.

Map Model – M.java class

The model for the venue map is encapsulated with the M.java class. It holds all convenient states, map information and the view representations of the parsed information.

Instantiating a map

Keeping a reference within your application is mandatory and depending on your needs you might want to create a service or a static variable for easy referencing and persistenting the state outside an activity's lifecycle.

```
  m = new M(context);
```

Setting up the M.java

The class has the following lifecycle:

1. **Start**
2. **Complete**
3. **Resume**
4. **Pause**
5. **Stop**
6. **Destroy**

M`s Start state

Dispatching: IntentFilterEngine.START
Dispatched when the user calls the following function.

```
m.start(String fileName, int config);
```

Which bootstraps the map, parsing and populating according to the venues data set found in the provided filename contents.
If there is a config resource present that any relevant settings to the platform will be taken in consideration.
Relative settings are limited to the visibility and the idle visual representational state of the view types.

```

LocalBroadcastManager.getInstance(context).registerReceiver(broadcastReceiverFr
ameworkInit, new IntentFilterEngine(m, IntentFilterEngine.COMPLETE));
    m.start(filename, R.raw.config);
```

M`s Complete state

Dispatching: IntentFilterEngine.COMPLETE
Dictates the completeness of the parsing, population phases found according to the VenueData returned filename contents.
Addition of custom elements and custom styles overrides should happen here.

```
LocalBroadcastManager.getInstance(context).registerReceiver(broadcastReceiverFr
ameworkComplete, new IntentFilterEngine(m, IntentFilterEngine.COMPLETE));
    m.start(filename, R.raw.config);

    private BroadcastReceiver broadcastReceiverFrameworkComplete = new
BroadcastReceiver() {
      @Override
      public void onReceive(Context context, Intent intent) {

LocalBroadcastManager.getInstance(context).unregisterReceiver(broadcastReceiver
FrameworkInit);

          // Custom idle styles for mover representations
          RenderStyle stairsStyle = new RenderStyle(Paint.Style.FILL);
          stairsStyle.paintFill.setColor(ColorsMaterialDesign.RED9);
          m.cssStyles.setStairs(stairsStyle, true);

          RenderStyle escalatorsStyle = new RenderStyle(Paint.Style.FILL);
          escalatorsStyle.paintFill.setColor(ColorsMaterialDesign.RED7);
          m.cssStyles.setEscalators(escalatorsStyle, true);

          RenderStyle elevatorsStyle = new RenderStyle(Paint.Style.FILL);
          elevatorsStyle.paintFill.setColor(ColorsMaterialDesign.RED5);
          m.cssStyles.setElevators(elevatorsStyle, true);

          // Custom elements addition to the map
          ElementMap custom = new CustomElement();
          m.addToMap(new MiniMap());
      }
    };
```

M`s Resume state

Dispatching: IntentFilterEngine.RESUME
resumes M instance, causes updates to happen in respond to the device's display events.

M's Pause state

Dispatching: IntentFilterEngine.PAUSE
pauses M instance, causes to stop listening to the device's display events.

M's Stop state

Dispatching: IntentFilterEngine.STOP
reset M for new venue data

M's Destroy state

Dispatching: IntentFilterEngine.DESTROY
destroys M

Available Types

The android SDK has the following types that can be extended and customized.

- Amenity
- Unit
- UnitLabel
- Obstacle
- Escalator
- Elevators
- Stair
- ParkingLot
- StreetMajor
- StreetMinor
- StreetSmallAlley
- MallBoundary
- Restroom
- Corridor
- Background
- YouAreHere
- Route
- Pin
- MoverHead
- MoverTail
- WayfindKiosk
- Kiosk

Custom Types Assignment.

After instantiation the developer can assign the logic for custom behavior by extending the relevant base class and setting the class field M.classLib to point to the custom classes.

```
m = new M(context);
m.classLib.unitLabelClass = UnitLabelsSameSize.class;
m.classLib.amenityClass = AmenityCustomScale.class;
```

Custom Type implementation

Custom behaviour can be achieved by extending a base class of the desired type and overriding the callback functions found on all map elements.
One restriction is that constructors must provide empty signatures.

```
//Initialization happens here, runs once on every element the first time the
engine starts
void onCreate(Context context, M m, long timeElapsed, long timeTotal, Camera
camera);

//Updates happen here, runs every frame.
void onUpdate(M m, long timeElapsed, long timeTotal, int fps, Camera camera);

//Physics updates or an other post updates happen here, called every frame
void onPostUpdate(M m, long timeElapsed, long timeTotal, int fps, Camera
camera);

// On collision detection enter runs on custom frame rate defined in the model:
M.java
void onCollision(M m, ArrayList<Element> collidesWith, long timeElapsed, long
timeTotal, int fps, Camera camera);

// Post callback runs after onCollision fn runs on custom frame rate defined in
the model: M.java
void onPostCollision(M m, long timeElapsed, long timeTotal, int fps, Camera
camera);

// Users wont need to override unless the nee to define custom transformation
Matrix onPreRender(M m, long timeElapsed, long timeTotal, int fps, Camera
camera);

// Canvas draw calls placed here, the engine will call this function once a
frame for every element
void onRender(Canvas canvas, Paint touchPaint);

// callback when destroy has been called on M instance.
void onDestroy(Context context);
```

Example of custom unit labels.

```
// extend the base class of UnitLabels
public class UnitLabelsSameSize extends
com.jibestream.jibestreamandroidlibrary.elements.UnitLabel {
private static final int STEP = 255/20;
private final Paint paint;
private Rect bounds = new Rect();
private float textOffsetY;
private float textWidth;

// make sure the constructor takes no parameters.
public UnitLabelsSameSize() {
  setConstantScale(true);
  paint = new Paint();
  paint.setTextSize(30);
  paint.setTextAlign(Paint.Align.CENTER);
  paint.setColor(ColorsMaterialDesign.GREY9);
  paint.setAntiAlias(true);
```

```java
  calc();
}

// react to updated text for the label
@Override
public void setText(String s) {
  super.setText(s);
  calc();
}

private void calc() {
  final String text = getText();
  if (text == null) return;
  paint.getTextBounds(text, 0, text.length(), bounds);
  textWidth = bounds.width();
  textOffsetY = bounds.height() * 0.5f;
}

private boolean testDesiredVisibility(float cameraZoom) {
  float calculatedWidth = width * cameraZoom;
  if (calculatedWidth < textWidth) {
    return true;
  } else {
    return false;
  }
}

@Override
public void onCreate(Context context, M m, long timeElapsed, long timeTotal,
Camera camera) {
  super.onCreate(context, m, timeElapsed, timeTotal, camera);
  if (testDesiredVisibility(camera.zoom)) {
    setVisible(false);
    paint.setAlpha(0);
  } else {
    setVisible(true);
    paint.setAlpha(255);
  }
}

// Update the label every frame
@Override
public void onUpdate(M m, long timeElapsed, long timeTotal, int fps, Camera
camera) {
  super.onUpdate(m, timeElapsed, timeTotal, fps, camera);
  if (textString == null || textString.isEmpty()) {
    setVisible(false);
  }
  super.onPreRender(m, timeElapsed, timeTotal, fps, camera);
  if (testDesiredVisibility(camera.zoom)) {
    addAlpha(-STEP*3);
  } else {
    addAlpha(STEP);
  }
}

private void addAlpha(int v) {
  v = paint.getAlpha() + v;
  if (v < 0) {
    paint.setAlpha(0);
  } else if (v > 255) {
    paint.setAlpha(255);
  } else {
    paint.setAlpha(v);
  }
  if (paint.getAlpha() > 0) {
    setVisible(true);
  } else {
    setVisible(false);
  }
}

// override the SDK's rendering implementation and define the look of the label
@Override
public void onRender(Canvas canvas, Paint touchPaint) {
  canvas.save();
  canvas.concat(transformation);
  canvas.translate(offsetX, offsetY);
  canvas.translate(0, textOffsetY);
  canvas.drawText(textString, 0, 0, paint);
  canvas.restore();
}
}
```

Custom Element implementation

The user can add custom elements to the map by creating a class that extends ElementMap and define the behaviour as demonstrated above.
Example of a custom element

```
public class Popup extends ElementMap {

private String titleString = "Title";
private String subTitleString = "SubTitle";
private String bodyString = "Lorem ipsum dolor sit amet, consectetur.";
private final TextStaticLayout titleStaticLayout = new TextStaticLayout();
private final TextStaticLayout subTitleStaticLayout = new TextStaticLayout();
private final TextStaticLayout bodyStaticLayout = new TextStaticLayout();

private int width = 400;
private JPath jPath;
private Path path;
private MultiShape multiShape;
```

```
public Popup() {
  this(null, null, null);
}

public Popup(String titleString, String subTitleString, String bodyString) {
  super();
  if (titleString != null) this.titleString = titleString;
  if (subTitleString != null) this.subTitleString = subTitleString;
  if (bodyString != null) this.bodyString = bodyString;

  jPath = new JPath();
  path = new Path();
  jPath.setPath(path);
  multiShape = new MultiShape();
  multiShape.iShapes.add(jPath);
  multiShape.iShapes.add(titleStaticLayout);
  multiShape.iShapes.add(subTitleStaticLayout);
  multiShape.iShapes.add(bodyStaticLayout);
  setShape(multiShape);
  calc();
}

private void calc() {
  titleStaticLayout.setTextString(titleString);
  final int padding = 20;
  titleStaticLayout.setWidth(width - padding * 2);
  titleStaticLayout.setOffsetX(padding);

  subTitleStaticLayout.setTextString(subTitleString);
  subTitleStaticLayout.setWidth(width - padding * 2);
  subTitleStaticLayout.setOffsetX(padding);

  bodyStaticLayout.setTextString(bodyString);
  bodyStaticLayout.setWidth(width - padding * 2);
  bodyStaticLayout.setOffsetX(padding);


  float accumY = padding;
  titleStaticLayout.setOffsetY(accumY);
  accumY += titleStaticLayout.getHeight();
  accumY += 5;
  subTitleStaticLayout.setOffsetY(accumY);
  accumY += subTitleStaticLayout.getHeight();
  accumY += 5;
  bodyStaticLayout.setOffsetY(accumY);
  accumY += bodyStaticLayout.getHeight();
  accumY += padding;

  path.reset();
  path.moveTo(0, 0);
  path.lineTo(width, 0);
  path.lineTo(width, accumY);
  float halfW = width * 0.5f;
  path.lineTo(halfW + 50, accumY);
  path.lineTo(200, accumY + 50);
  path.lineTo(halfW - 50, accumY);
  path.lineTo(0, accumY);
  path.close();

  multiShape.setOffsetX(-halfW);
  multiShape.setOffsetY(-accumY - 50f);
}


public String getTitleString() {
  return titleString;
}

public void setTitleString(String titleString) {
  this.titleString = titleString;
  calc();
}

public String getSubTitleString() {
  return subTitleString;
}

public void setSubTitleString(String subTitleString) {
  this.subTitleString = subTitleString;
  calc();
}

public String getBodyString() {
  return bodyString;
}

public void setBodyString(String bodyString) {
  this.bodyString = bodyString;
  calc();
}

public TextStaticLayout getBodyStaticLayout() {
  return bodyStaticLayout;
}


public TextStaticLayout getTitleStaticLayout() {
  return titleStaticLayout;
}


public TextStaticLayout getSubTitleStaticLayout() {
  return subTitleStaticLayout;
```

```
}

public int getWidth() {
  return width;
}

public void setWidth(int width) {
  this.width = width;
  calc();
}
```

After the map is complete the user can add custom elements

```
  Popup popUp = new Popup();
  //
  popUp.setSelectable(true);
  popUp.setVisible(false);
  popUp.setHeadsUp(true);
  popUp.setConstantScale(true);
  //
  m.addToMap(popUp);


// Listen to events broadcasted by the framework
    LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(context);
    localBroadcastManager.registerReceiver(broadcastRieverTouchLong, new
IntentFilterTouch(m, IntentFilterTouch.TYPE_LONG_PRESS));
    localBroadcastManager.registerReceiver(broadcastReceiverTouchSinglePopUp,
new IntentFilterTouch(m, IntentFilterTouch.TYPE_SINGLE));

  broadcastRieverTouchLong = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
      IntentTouch intentTouch = (IntentTouch) intent;
      final Element element = m.getElementWithID(intentTouch.id);
      Unit unit = null;
      if (element instanceof Unit) {
        unit = (Unit) element;
      }
      if (unit != null) {
        popUp.setVisible(false);
        final Destination[] destinations = unit.getDestinations();
        if (destinations != null && destinations.length > 0) {
          final Destination destination = destinations[0];
          if (destination == null) return;
          popUp.setLevel(element.getLevel());
          popUp.setTitleString(destination.name);
          //
          if (destination.category != null && destination.category.length > 3)
{
            String asString = destination.category[0] + ", " +
destination.category[1] + ", " + destination.category[2];
            int max = Math.min(50, asString.length() - 1);
            String s = asString.substring(0, max) + "...";
            popUp.setSubTitleString(s);
          } else {
            popUp.setSubTitleString("");
          }
          //
          if (destination.description != null &&
destination.description.length() > 0) {
            int maxBodyString = Math.min(300, destination.description.length()
- 1);
            String bodyString = destination.description.substring(0,
maxBodyString) + "...";
            popUp.setBodyString(bodyString);
          }
          final Waypoint[] waypointsOfDestination =
Building.getWaypointsOfDestination(m, unit.getDestinations()[0]);
          popUp.getTransform().setTranslationX(waypointsOfDestination[0].x);
          popUp.getTransform().setTranslationY(waypointsOfDestination[0].y);
//          unit.addChild(popUp);
          popUp.setVisible(true);
        }
      }
    }
  };


  broadcastReceiverTouchSinglePopUp = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
      IntentTouch intentTouch = (IntentTouch) intent;
      if (intentTouch.id == popUp.id) {
        popUp.setVisible(false);
      }
    }
  };
```

The EngineView.java class

You must instantiate the frameworks EngineView class and connect it to the M class

```
    engineView = (EngineView) findViewById(R.id.engineview);
    m.setEngineView(engineView);
    m.onResume();
```

Defining the view in xml layout would look something to this.

```
<com.jibestream.jibestreamandroidlibrary.main.EngineView
                android:id="@+id/engineview"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                />
```

Engine's events dispatches.

The EngineView implements the simple gestures from the platform and for convenience propagates the events by dispatching them via the LocalBroadcastManager.
If the user wishes for some reason to react to one of the following events.

*EngineView.SURFACE_CREATED
*EngineView.SURFACE_DESTROYED
*EngineView.SURFACE_CHANGED
*EngineView.ON_TOUCH_EVENT
*EngineView.ON_ROTATION
*EngineView.ON_SCALE
*EngineView.ON_DOWN
*EngineView.ON_SINGLE_TAP_UP
*EngineView.ON_SINGLE_TAP_CONFIRMED
*EngineView.ON_DOUBLE_TAP
*EngineView.ON_SCROLL
*EngineView.ON_LONGPRESS
*EngineView.ON_FLING

```

LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(context);
localBroadcastManager.registerReceiver(broadcastRecieverEngineView, new
IntentFilter(EngineView.SURFACE_CREATED));
localBroadcastManager.registerReceiver(broadcastRecieverEngineView, new
IntentFilter(EngineView.SURFACE_DESTROYED));

BroadcastReceiver broadcastRecieverEngineView = new BroadcastReceiver() {
  @Override
  public void onReceive(Context context, Intent intent) {
    String action = intent.getAction();
    if (engineView.id != intent.getIntExtra("id", -1)) return;
    switch (action) {
      case EngineView.SURFACE_CREATED:

        break;
      case EngineView.SURFACE_DESTROYED:

        break;
    }
  }
};
```

Model and Building

All engine's states are stored under the class M.Java from here you can retrieve state`s values either from public fields
or from getters and setters if a state dispatches an Event.
M.java maintains all data and Building.java is a static class with helper methods attached to it.

Showing the wayfind route.

The developer can show the route by setting the M.setFromWaypoint() and the M.setToWaypoint() and can reset by nullifying the setters.

```
Waypoint waypointFrom = Building.getWaypointsOfDestination(m, aDestinationFrom)
[0];
m.setFromWaypoint(aWaypointFrom);

Waypoint waypointTo = Building.getWaypointsOfDestination(m, aDestinationTo)[0];
m.setToWaypoint(waypointTo);
```

Manipulating the view

Each instance of M.java class comes with a camera that the developer can manipulate.
The frame of reference is within map coordinates and the center of the camera is the origin point of the transformations applied to it.

```
      m.camera.setRoll(aFloat);
      //
      m.camera.setTranslation(x,y);
```

Frame a region

In allot of cases the developer would like to place the camera in a fashion that frames a certain region.

```
      m.camera.zoomTo(aRectF, padding);
```

Listening to an element touch event.

The developer can set if a certain type or an individual element is selectable.
By default the Unit Type is selectable and in the SDK will broadcast an event whenever a user interacted with the visual represntation.
An example follows how to react to double tapping on a Unit and framing that unit within the viewport.

```
LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(context);
localBroadcastManager.registerReceiver(broadcastReceiverTouchDouble, new
IntentFilterTouch(m, IntentFilterTouch.TYPE_DOUBLE));


broadcastReceiverTouchDouble = new BroadcastReceiver() {
  @Override
  public void onReceive(Context context, Intent intent) {

    IntentTouch intentTouch = (IntentTouch) intent;
    final Element element = m.getElementWithID(intentTouch.id);

    if (element instanceof Unit) {
      Unit unit = (Unit) element;
      m.camera.zoomTo(unit.getBBox(), 10);
    }
  }
};
```

Sample app.

Bellow there implementations how to use the engine.

MService class

wraps the M.java in a service with overrides for custom styles and adding custom elements to the map

```
package com.jibestream.androidsdk;

import android.app.Service;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.Paint;
import android.os.Binder;
import android.os.IBinder;
import android.support.v4.content.LocalBroadcastManager;
import android.text.TextPaint;
import android.util.Log;

import com.jibestream.androidsdk.customElements.*;
import com.jibestream.jibestreamandroidlibrary.elements.*;
import
com.jibestream.jibestreamandroidlibrary.intentFilters.IntentFilterEngine;
import com.jibestream.jibestreamandroidlibrary.intentFilters.IntentFilterTouch;
import com.jibestream.jibestreamandroidlibrary.intents.IntentTouch;
import com.jibestream.jibestreamandroidlibrary.main.Building;
import com.jibestream.jibestreamandroidlibrary.main.M;
import
com.jibestream.jibestreamandroidlibrary.mapBuilderV3.dataObjects.Destination;
import
com.jibestream.jibestreamandroidlibrary.mapBuilderV3.dataObjects.Waypoint;
import com.jibestream.jibestreamandroidlibrary.styles.CssStyles;
import com.jibestream.jibestreamandroidlibrary.styles.RenderStyle;
import com.jibestream.jibestreamandroidlibrary.styles.RenderStyleIcon;
import com.jibestream.jibestreamandroidlibrary.utils.ColorsMaterialDesign;

import java.util.Random;
import java.util.Timer;
import java.util.TimerTask;

/**
 * Model class that holds all shared data and
 * application state.
 * Created by emmanuel on 15-05-29.
 */
public class MService extends Service {
private static final String TAG = "MService";
private final IBinder mBinder = new MBinder();
private M m;
public final Popup popUp = new Popup();

public BroadcastReceiver broadcastRieverTouchLong;
public BroadcastReceiver broadcastReceiverTouchSinglePopUp;
public BroadcastReceiver broadcastReceiverTouchDouble;
public BroadcastReceiver broadcastReceiverTouchSingle;

//
@Override
public IBinder onBind(Intent arg0) {
  return mBinder;
}

public class MBinder extends Binder {
  public M getM() {
    return MService.this.m;
  }
}

@Override
public void onCreate() {
  super.onCreate();
  Log.d(TAG, ">>> MService created");
  //
  m = new M(getApplicationContext());
  //Set the custom classes to be used instead of the defaults
  m.classLib.unitClass = com.jibestream.androidsdk.customElements.Unit.class;
  m.classLib.unitLabelClass = UnitLabelsSameSize.class;
  m.classLib.amenityClass = AmenityCol.class;
  m.classLib.pinClass = Pin.class;
  m.classLib.youAreHereClass = YouAreHere.class;
  m.classLib.routeClass = Route.class;
  m.classLib.wayfindKioskClass = WayfindKiosk.class;
  m.classLib.moverHeadClass = MoverHead.class;
  m.classLib.moverTailClass = MoverTail.class;
  //
  initBroadcastListeners();
  //
  LocalBroadcastManager.getInstance(this).registerReceiver(new
BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
      customStyles();
      customElements();
      //
      LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(context);
    }
  }, new IntentFilterEngine(m, IntentFilterEngine.START));
  //
  LocalBroadcastManager.getInstance(this).registerReceiver(new
BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
      LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(context);
      localBroadcastManager.registerReceiver(broadcastReceiverTouchSingle, new
IntentFilterTouch(m, IntentFilterTouch.TYPE_SINGLE));
      localBroadcastManager.registerReceiver(broadcastReceiverTouchDouble, new
IntentFilterTouch(m, IntentFilterTouch.TYPE_DOUBLE));
      localBroadcastManager.registerReceiver(broadcastRieverTouchLong, new
IntentFilterTouch(m, IntentFilterTouch.TYPE_LONG_PRESS));
      localBroadcastManager.registerReceiver(broadcastReceiverTouchSinglePopUp,
new IntentFilterTouch(m, IntentFilterTouch.TYPE_SINGLE));
    }
  }, new IntentFilterEngine(m, IntentFilterEngine.RESUME));


  LocalBroadcastManager.getInstance(this).registerReceiver(new
BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
      LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(context);
      localBroadcastManager.unregisterReceiver(broadcastReceiverTouchSingle);
      localBroadcastManager.unregisterReceiver(broadcastReceiverTouchDouble);
      localBroadcastManager.unregisterReceiver(broadcastRieverTouchLong);

localBroadcastManager.unregisterReceiver(broadcastReceiverTouchSinglePopUp);
```

```
    }
  }, new IntentFilterEngine(m, IntentFilterEngine.PAUSE));

}

@Override
public int onStartCommand(Intent intent, int flags, int startId) {
  Log.d(TAG,"onStartCommand");
  return super.onStartCommand(intent, flags, startId);
}

@Override
public void onTaskRemoved(Intent rootIntent) {
  super.onTaskRemoved(rootIntent);
  m.onDestroy();
  stopSelf();
}

private void initBroadcastListeners() {
  // Tap event updates selection snd Destination FRom
  broadcastReceiverTouchDouble = new BroadcastReceiver() {
    Element prev = null;
    @Override
    public void onReceive(Context context, Intent intent) {
      IntentTouch intentTouch = (IntentTouch) intent;
      final Element element = m.getElementWithID(intentTouch.id);
      prev = element;
      if (element instanceof Unit) {
        Unit unit = (Unit) element;
        if (unit.getWaypoints() == null) return;
        Waypoint waypoint = unit.getWaypoints()[0];
        Building.setFrom(m, waypoint);
      } else if (element instanceof Amenity) {
        Amenity amenity = (Amenity) element;
        Building.setFrom(m, amenity.waypoint);

      }
    }
  };

  // Tap event updates selection snd Destination TO
  broadcastReceiverTouchSingle = new BroadcastReceiver() {
    Element prev = null;

    @Override
    public void onReceive(Context context, Intent intent) {
      IntentTouch intentTouch = (IntentTouch) intent;
      final Element element = m.getElementWithID(intentTouch.id);
      if (element instanceof Unit) {
        Unit unit = (Unit) element;
        if (unit.getDestinationIDs() == null || unit.getDestinationIDs().length
== 0) return;
```

```
        Waypoint waypoint = unit.getWaypoints()[0];
        Building.setTo(m, waypoint);
        if (prev != null) prev.setSelectState(false);
        element.setSelectState(true);
        prev = element;
      } else if (element instanceof Amenity) {
        Amenity amenity = (Amenity) element;
        Building.setTo(m, amenity.waypoint);
        if (prev != null) prev.setSelectState(false);
        element.setSelectState(true);
        prev = element;
      }
    }
  };

  //

  broadcastRieverTouchLong = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
      IntentTouch intentTouch = (IntentTouch) intent;
      final Element element = m.getElementWithID(intentTouch.id);
      Unit unit = null;
      if (element instanceof Unit) {
        unit = (Unit) element;
      }
      if (unit != null) {
        popUp.setVisible(false);
        final Destination[] destinations = unit.getDestinations();
        if (destinations != null && destinations.length > 0) {
          final Destination destination = destinations[0];
          if (destination == null) return;
          popUp.setLevel(element.getLevel());
          popUp.setTitleString(destination.name);
          //
          if (destination.category != null && destination.category.length > 3)
{
            String asString = destination.category[0] + ", " +
destination.category[1] + ", " + destination.category[2];
            int max = Math.min(50, asString.length() - 1);
            String s = asString.substring(0, max) + "...";
            popUp.setSubTitleString(s);
          } else {
            popUp.setSubTitleString("");
          }
          //
          if (destination.description != null &&
destination.description.length() > 0) {
            int maxBodyString = Math.min(300, destination.description.length()
- 1);
            String bodyString = destination.description.substring(0,
maxBodyString) + "...";
```

```
            popUp.setBodyString(bodyString);
          }
          final Waypoint[] waypointsOfDestination =
Building.getWaypointsOfDestination(m, unit.getDestinations()[0]);
          popUp.getTransform().setTranslationX(waypointsOfDestination[0].x);
          popUp.getTransform().setTranslationY(waypointsOfDestination[0].y);
//          unit.addChild(popUp);
          popUp.setVisible(true);
        }
      }
    }
  };


  broadcastReceiverTouchSinglePopUp = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
      IntentTouch intentTouch = (IntentTouch) intent;
      if (intentTouch.id == popUp.id) {
        popUp.setVisible(false);
      }
    }
  };

}

private void customStyles() {
  /////////////////////   Defining new styles  for elements the engine   will
instantiate
  RenderStyle unitStyle = new RenderStyle(Paint.Style.FILL_AND_STROKE);
  unitStyle.paintFill.setColor(ColorsMaterialDesign.GREY7);
  unitStyle.paintStroke.setColor(ColorsMaterialDesign.GREY9);
  unitStyle.paintStroke.setStrokeWidth(0.25f);
  unitStyle.paintStroke.setAntiAlias(true);
  unitStyle.paintStroke.setStyle(Paint.Style.STROKE);
  m.cssStyles.setUnits(unitStyle, true);
  m.cssStyles.setKiosks(unitStyle, true);

  RenderStyle obstaclesStyle = new RenderStyle(Paint.Style.FILL_AND_STROKE);
  obstaclesStyle.paintFill.setColor(ColorsMaterialDesign.GREY9);
  obstaclesStyle.paintStroke.setColor(ColorsMaterialDesign.GREY7);
  obstaclesStyle.paintStroke.setStrokeWidth(1);
  m.cssStyles.setObstacles(obstaclesStyle, true);

  RenderStyle parkingStyle = new RenderStyle(Paint.Style.FILL);
  parkingStyle.paintFill.setColor(ColorsMaterialDesign.GREY8);
  parkingStyle.paintFill.setAntiAlias(true);
  m.cssStyles.setParkingLots(parkingStyle, true);
  m.cssStyles.setParkingLotsInterior(parkingStyle, true);


  RenderStyleIcon renderStyleIconIdle = new RenderStyleIcon();
```

```
renderStyleIconIdle.renderStyleBG.paintFill.setColor(ColorsMaterialDesign.GREY3
);
  renderStyleIconIdle.renderStyleBG.paintFill.setAntiAlias(true);

}

private void customElements() {

/////////////////////////////////////////////////////////////////
////////////         Custom  objects addition         ///////////
////////////         to the display list              ///////////
/////////////////////////////////////////////////////////////////
  m.addToMap(new Dots());
  m.addToMap(new Compass());
  m.addToMap(new MiniMap());

  final SpinnerTest spinner = new SpinnerTest();
  RenderStyle spinnerRenderStyle = new RenderStyle(Paint.Style.FILL);
  spinnerRenderStyle.paintFill.setColor(ColorsMaterialDesign.YELLOW5);
  spinner.setStyleIdle(spinnerRenderStyle);
  // To appear only on the first map of the m.maps array
  spinner.setLevel(0);
  // Add to the map
  m.addToMap(spinner);
  spinner.getTransform().setTranslationX(2750);
  spinner.getTransform().setTranslationY(2750);
//
/////////////////////////////////////////////////////////////////
////////////       Hooking into Engine`s              ///////////
////////////       dispatch listening                 ///////////
/////////////////////////////////////////////////////////////////


  popUp.setSelectable(true);
  m.addToMap(popUp);
  popUp.setVisible(false);
  popUp.setHeadsUp(true);
  popUp.setConstantScale(true);


  final RenderStyle renderStyle = new RenderStyle(Paint.Style.FILL);
  renderStyle.setFillColor(ColorsMaterialDesign.GREY9);
  renderStyle.setFillAlpha((int) (255 * 0.7f));
  popUp.setStyleIdle(renderStyle);


  TextPaint titleTextPaint = new TextPaint();
  titleTextPaint.setTextSize(34);
  titleTextPaint.setColor(ColorsMaterialDesign.AMBER9);
  popUp.getTitleStaticLayout().setTextPaint(titleTextPaint);
```

~~~~
  TextPaint subTextPaint = new TextPaint();
  subTextPaint.setTextSize(26);
  subTextPaint.setColor(ColorsMaterialDesign.AMBER8);
  popUp.getSubTitleStaticLayout().setTextPaint(subTextPaint);

  TextPaint texPaint = new TextPaint();
  texPaint.setTextSize(25);
  texPaint.setColor(ColorsMaterialDesign.GREY4);
  popUp.getBodyStaticLayout().setTextPaint(texPaint);
}
}


### LiveMapWithStaticMapActivity class
Activity with map

```Java

package com.jibestream.androidsdk;

import android.animation.ObjectAnimator;
import android.animation.TypeEvaluator;
import android.animation.ValueAnimator;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.ServiceConnection;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.RectF;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.Looper;
import android.os.Message;
import android.support.v4.content.LocalBroadcastManager;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.SearchView;
import android.text.TextPaint;
import android.util.Log;
import android.view.Display;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
import android.view.animation.AccelerateDecelerateInterpolator;
~~~~

```
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

import com.jibestream.jibestreamandroidlibrary.elements.Element;
import com.jibestream.jibestreamandroidlibrary.elements.ParkingLot;
import com.jibestream.jibestreamandroidlibrary.elements.Unit;
import com.jibestream.jibestreamandroidlibrary.intentFilters.IntentFilterMap;
import com.jibestream.jibestreamandroidlibrary.intentFilters.IntentFilterTouch;
import
com.jibestream.jibestreamandroidlibrary.intentFilters.IntentFilterWaypoint;
import com.jibestream.jibestreamandroidlibrary.intents.IntentWaypoint;
import com.jibestream.jibestreamandroidlibrary.main.Building;
import com.jibestream.jibestreamandroidlibrary.main.VenueData;
import com.jibestream.jibestreamandroidlibrary.mapBuilderV3.dataObjects.Zone;
import
com.jibestream.jibestreamandroidlibrary.mapBuilderV3.textDirections.TextDirecti
onInstruction;
import com.jibestream.jibestreamandroidlibrary.main.EngineView;
import com.jibestream.jibestreamandroidlibrary.main.M;
import
com.jibestream.jibestreamandroidlibrary.mapBuilderV3.dataObjects.Category;
import
com.jibestream.jibestreamandroidlibrary.mapBuilderV3.dataObjects.Destination;
import
com.jibestream.jibestreamandroidlibrary.mapBuilderV3.dataObjects.MapFull;

import java.util.ArrayList;

public class LiveMapWithStaticMapActivity extends AppCompatActivity {
private static final String TAG = "LiveMapWithStaticMapActivity";
private ToggleButton aToggleButton;
private Button framePathButton;
private Button frameMapButton;
private Button resetCameraButton;
private Button frame40percentButton;
private Button frameCustomButton;
private Button frameRandDestButton;
private Button leftButton;
private Button rightButton;
private Button zoomInButton;
private Button zoomOutButton;
private Button rotateButton;
private Button toClosestAmenityButton;
private Button highlightCategoryButton;
private Button highlightUnitButton;
private Button showTextDirectionsButton;
private Button imageButton;
private Button showZonesButton;
private TextView fromView;
private TextView toView;
private TextView levelView;
private TextView debugView;
private ImageView mapImageView;

private String lastSearchQuery;
private ArrayList<Integer> selectedItems = new ArrayList();
private volatile M m;
private volatile M mImage;
protected Context context;
private EngineView engineView;
public VenueData venueData;
public String fileName;
//
public BroadcastReceiver broadcastReceiverMap = new BroadcastReceiver() {
  @Override
  public void onReceive(Context context, Intent intent) {

    MapFull currentMap = m.getCurrentMap();
    if (currentMap == null) return;
    final String s = currentMap.map.name;
    setTitle(s);
    runOnUiThread(new Runnable() {
      final TextView tv = levelView;

      @Override
      public void run() {
        tv.setText(s);
      }
    });
  }
};
private BroadcastReceiver broadcastReceiverWaypoint = new BroadcastReceiver() {
  //region onReceive
  @Override
  public void onReceive(Context context, Intent intent) {
    final IntentWaypoint waypointEvent = (IntentWaypoint) intent;
    final Destination[] destinationsFrom =
Building.getDestinationsOfWaypoint(m, waypointEvent.from);
    final Destination[] destinationsTo = Building.getDestinationsOfWaypoint(m,
waypointEvent.to);
    runOnUiThread(new Runnable() {
      @Override
      public void run() {
        if (destinationsFrom != null && destinationsFrom.length > 0 &&
destinationsFrom[0] != null) {
          fromView.setText(destinationsFrom[0].name);
        } else {

com.jibestream.jibestreamandroidlibrary.mapBuilderV3.dataObjects.Amenity[]
amenities =
              Building.getAmenitiesOfWaypoint(m, m.getFromWaypoint());
          if (amenities != null && amenities.length > 0 && amenities[0] !=
null) {
            String display = amenities[0].bean.description;
            fromView.setText(display);
          } else {
            fromView.setText("please set..");
          }
        }
        if (destinationsTo != null && destinationsTo.length > 0 &&
destinationsTo[0] != null) {
          toView.setText(destinationsTo[0].name);
        } else {

com.jibestream.jibestreamandroidlibrary.mapBuilderV3.dataObjects.Amenity[]
amenities =
              Building.getAmenitiesOfWaypoint(m, m.getToWaypoint());
          if (amenities != null && amenities.length > 0 && amenities[0] !=
null) {
            String display = amenities[0].bean.description;
            toView.setText(display);
          } else {
            toView.setText("please set..");
          }
        }
      }
    });
  }
  //endregion
};
//


boolean isBound = false;
private ServiceConnection serviceConnection = new ServiceConnection() {
  @Override
  public void onServiceConnected(ComponentName name, IBinder service) {
    Log.i(TAG, "onServiceConnected");
    MService.MBinder binder = (MService.MBinder) service;
    m = binder.getM();
    isBound = true;
    m.setEngineView(engineView);
    m.onResume();
    //
    LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(context);
    localBroadcastManager.registerReceiver(broadcastReceiverMap, new
IntentFilterTouch(m, IntentFilterMap.ACTION));
    localBroadcastManager.registerReceiver(broadcastReceiverWaypoint, new
IntentFilterTouch(m, IntentFilterWaypoint.ACTION));
  }
  @Override
  public void onServiceDisconnected(ComponentName name) {
    isBound = false;
    //
    m.onPause();
    LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(context);
    localBroadcastManager.unregisterReceiver(broadcastReceiverMap);
    localBroadcastManager.unregisterReceiver(broadcastReceiverWaypoint);
  }
};

@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  //
  context = getApplicationContext();
  //
  //Remove notification bar
  this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
  //Keep screen on...
  getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
  //Set activity content from layout resource
  setContentView(R.layout.activity_maplive_with_mapstatic);
  //
  engineView = (EngineView) findViewById(R.id.engineview);
  //
  Intent intent = new Intent(this, MService.class);
  startService(intent);
  bindService(intent, serviceConnection, Context.BIND_AUTO_CREATE);
}

@Override
protected void onResume() {
  super.onResume();
  //
  if (m != null) m.onResume();
  //
  findViews();

  addButtonListeners();
}

@Override
protected void onPause() {
  super.onPause();
  //
  if (m != null) m.onPause();
}
```

```
@Override
protected void onDestroy() {
  unbindService(serviceConnection);
  super.onDestroy();
}

private void findViews() {
  // Reference buttons
  aToggleButton = (ToggleButton) findViewById(R.id.aToggleButton);
  frameMapButton = (Button) findViewById(R.id.frameMapButton);
  framePathButton = (Button) findViewById(R.id.framePathButton);
  frame40percentButton = (Button) findViewById(R.id.frame40percentButton);
  frameCustomButton = (Button) findViewById(R.id.frameCustomButton);
  frameRandDestButton = (Button) findViewById(R.id.frameRandDestButton);
  resetCameraButton = (Button) findViewById(R.id.resetCameraButton);
  zoomInButton = (Button) findViewById(R.id.zoomInButton);
  zoomOutButton = (Button) findViewById(R.id.zoomOutButton);
  rotateButton = (Button) findViewById(R.id.rotateButton);
  leftButton = (Button) findViewById(R.id.leftButton);
  rightButton = (Button) findViewById(R.id.rightButton);
  showTextDirectionsButton = (Button)
findViewById(R.id.showTextDirectionsButton);
  toClosestAmenityButton = (Button) findViewById(R.id.toClosestAmenityButton);
  highlightCategoryButton = (Button)
findViewById(R.id.highlightCategoryButton);
  highlightUnitButton = (Button) findViewById(R.id.highlightUnitButton);
  imageButton = (Button) findViewById(R.id.imageButton);
  showZonesButton = (Button) findViewById(R.id.showZonesButton);
  levelView = (TextView) findViewById(R.id.levelView);
  debugView = (TextView) findViewById(R.id.debugView);
  fromView = (TextView) findViewById(R.id.fromView);
  toView = (TextView) findViewById(R.id.toView);
  mapImageView = (ImageView) findViewById(R.id.mapImageView);
}

private void addButtonListeners() {
  aToggleButton.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
{
      if (isChecked) {
        // The toggle is enabled
      } else {
        // The toggle is disabled
      }
    }
  });

  final ValueAnimator valueAnimatorRotCam = new
ValueAnimator().setDuration(333);
  valueAnimatorRotCam.setInterpolator(new AccelerateDecelerateInterpolator());
  valueAnimatorRotCam.addUpdateListener(new
ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator animation) {
      m.camera.setRoll((Float) animation.getAnimatedValue());
    }
  });
  rotateButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      if (valueAnimatorRotCam.isRunning()) valueAnimatorRotCam.end();
      float roll = m.camera.getRoll();
      valueAnimatorRotCam.setFloatValues(roll, roll - 45);
      valueAnimatorRotCam.start();
//      m.camera.addRoll(-45);
    }
  });
  final ValueAnimator valueAnimatorCamZoomStep = new
ValueAnimator().setDuration(222);
  valueAnimatorCamZoomStep.setInterpolator(new
AccelerateDecelerateInterpolator());
  valueAnimatorCamZoomStep.addUpdateListener(new
ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator animation) {
      m.camera.setScale((Float) animation.getAnimatedValue());
    }
  });
  zoomInButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      if (valueAnimatorCamZoomStep.isRunning()) valueAnimatorCamZoomStep.end();
      float scale = m.camera.getScale();
      valueAnimatorCamZoomStep.setFloatValues(scale, scale - scale * 0.5f);
      valueAnimatorCamZoomStep.start();
//      m.camera.zoomIn();
    }
  });
  zoomOutButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      if (valueAnimatorCamZoomStep.isRunning()) valueAnimatorCamZoomStep.end();
      float scale = m.camera.getScale();
      valueAnimatorCamZoomStep.setFloatValues(scale, scale + scale * 0.5f);
      valueAnimatorCamZoomStep.start();
//      m.camera.zoomOut();
    }
  });
  frameMapButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      m.camera.zoomOutMax();
    }
  });
  framePathButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      Building.cameraToPath(m, m.camera);
    }
  });
  final ValueAnimator valueAnimator40 = new ValueAnimator().setDuration(667);
  valueAnimator40.setInterpolator(new AccelerateDecelerateInterpolator());
  valueAnimator40.addUpdateListener(new ValueAnimator.AnimatorUpdateListener()
{
    @Override
    public void onAnimationUpdate(ValueAnimator animation) {
      m.camera.framePerCent((Float) animation.getAnimatedValue());
    }
  });
  frame40percentButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      float cent = m.camera.getScaleUnit() * 100f;
      float centTo = cent + 40f;
      centTo = Math.min(centTo, 100f);
      centTo = Math.max(centTo, 0.1f);
      valueAnimator40.setFloatValues(cent, centTo);
      valueAnimator40.start();
    }
  });
  frameCustomButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      final RectF rectOfClassesInstances =
Building.getRectOfClassesInstances(m, new Class[]{ParkingLot.class},
m.getCurrentMapIndex());
      float sqrt = (float) Math.sqrt(rectOfClassesInstances.width() *
rectOfClassesInstances.width() + rectOfClassesInstances.height() *
rectOfClassesInstances.height());
      sqrt = sqrt * -0.5f;
      RectF rectF = new RectF(
          rectOfClassesInstances.centerX(),
          rectOfClassesInstances.centerY(),
          rectOfClassesInstances.centerX(),
          rectOfClassesInstances.centerY()
      );
      rectF.inset(sqrt, sqrt);
      m.camera.zoomTo(rectF, 0);
    }
  });
  frameRandDestButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      final Destination[] destinations = m.venueData.destinations;
      final double random = Math.random() * (destinations.length - 1f);
      final Destination destination = destinations[((int) random)];
      final ArrayList<Element> elements = Building.getElementsOfDestination(m,
destination.id);
      if (elements.size() == 0) return;
      Element el = elements.get(0);
      m.setLevel(m.venueData.maps[el.getLevel()]);
      m.camera.zoomTo(el.getBBox(), 10);
    }
  });
  resetCameraButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      m.camera.reset();
    }
  });
  rightButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      if (valueAnimatorCamZoomStep.isRunning()) valueAnimatorCamZoomStep.end();
      float scale = m.camera.getScale();
      valueAnimatorCamZoomStep.setFloatValues(scale, scale * 0.95f, scale -
scale * 0.05f, scale);
      valueAnimatorCamZoomStep.start();
      m.nextLevel();
    }
  });
  leftButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      if (valueAnimatorCamZoomStep.isRunning()) valueAnimatorCamZoomStep.end();
      float scale = m.camera.getScale();
      valueAnimatorCamZoomStep.setFloatValues(scale, scale * 0.95f, scale -
scale * 0.05f, scale);
      valueAnimatorCamZoomStep.start();
      m.prevLevel();
    }
  });
  toClosestAmenityButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      showAmenitiesDialog(findClosestAmenity);
    }
  });
  highlightCategoryButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      showCategoryDialog();
    }
  });
  highlightUnitButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      showDestinationsDialog(highlightDestination);
    }
  });
  showTextDirectionsButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      showTextDirections();
    }
  });
  imageButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      final Bitmap bitmap = Bitmap.createBitmap(mapImageView.getWidth(),
mapImageView.getHeight(), Bitmap.Config.ARGB_8888);
      Canvas canvas = new Canvas(bitmap);
      m.renderToCanvas(canvas, new M.RenderCallback() {
        @Override
        public void onRender() {
          runOnUiThread(new Runnable() {
            @Override
            public void run() {
              mapImageView.setImageBitmap(bitmap);
            }
          });
        }
      });
    }
  });
  showZonesButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      showZonesDialog();
    }
  });
}

//
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  // Inflate the menu items for use in the action bar
  MenuInflater inflater = getMenuInflater();
  inflater.inflate(R.menu.main_activity_actions, menu);

  MenuItem searchItem = menu.findItem(R.id.action_search);
  final SearchView searchView = (SearchView) searchItem.getActionView();
  final Menu m = menu;
  searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String query) {
      lastSearchQuery = query;
      searchView.setIconified(true);
      searchView.clearFocus();
      (m.findItem(R.id.action_search)).collapseActionView();
      showDestinationsDialog(searchDestination);
      return true;
    }

    @Override
    public boolean onQueryTextChange(String newText) {
      return false;
    }
  });

  return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
  // Handle presses on the action bar items
  switch (item.getItemId()) {
    case R.id.action_amneties_visibilty:
      showAmenitiesVisibilityDialog();
      return true;
    case R.id.action_from:
      showDestinationsDialog(fromDestination);
      return true;
    case R.id.action_to:
      showDestinationsDialog(toDestination);
      return true;
    case R.id.action_search:
//      listView.setVisibility(View.VISIBLE);
      return true;
    case R.id.action_visibility_elements:
      showElementsVisibilityDialog();
      return true;
    default:
      return super.onOptionsItemSelected(item);
  }
}

private static final int findClosestAmenity = 0;

private void showAmenitiesDialog(final int trigger) {
  AlertDialog.Builder builder = new AlertDialog.Builder(this);
  switch (trigger) {
    case findClosestAmenity:
      builder.setTitle("Closest Amenity of...");
      break;
  }
  final ArrayAdapter<String> stringArrayAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1);
  for (com.jibestream.jibestreamandroidlibrary.mapBuilderV3.dataObjects.Amenity
amenity : m.venueData.amenities) {
    stringArrayAdapter.add(amenity.bean.localizedText);
  }
  builder.setNegativeButton(android.R.string.no, new
DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
      // do nothing
    }
  });
  builder.setAdapter(stringArrayAdapter, new DialogInterface.OnClickListener()
{
    @Override
    public void onClick(DialogInterface dialog, int which) {
      String strName = stringArrayAdapter.getItem(which);
      Toast.makeText(context, strName, Toast.LENGTH_SHORT).show();
      switch (trigger) {
        case findClosestAmenity:

com.jibestream.jibestreamandroidlibrary.mapBuilderV3.dataObjects.Amenity
amenity = m.venueData.amenities[which];
          Building.wayfindToClosestAmenity(m, amenity.bean.componentId);
          break;
      }
    }
  });
  builder.show();
}

private static final int fromDestination = 0;
private static final int toDestination = 1;
private static final int highlightDestination = 2;
private static final int searchDestination = 3;

private void showDestinationsDialog(final int destinationWay) {
//  final AlertDialog.Builder builder = new AlertDialog.Builder(this);
  final AlertDialog.Builder builder = new
AlertDialog.Builder(getSupportActionBar().getThemedContext());
  final ArrayAdapter<String> destinations = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1);
  ArrayList<Destination> destinationsByQuery = null;
  switch (destinationWay) {
    case fromDestination:
    case toDestination:
    case highlightDestination:
      for (Destination destination : m.venueData.destinations) {
        destinations.add(destination.name);
      }
      break;
    case searchDestination:
      destinationsByQuery = Building.getDestinationsByQuery(m, lastSearchQuery,
0);
      for (Destination destination : destinationsByQuery) {
        destinations.add(destination.name);
      }
      break;
  }
  if (destinations.getCount() == 0) return;
  builder.setNegativeButton(android.R.string.no, new
DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
    }
  });
  final ArrayList<Destination> finalDestinationsByQuery = destinationsByQuery;
  builder.setAdapter(destinations, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
      String strName = destinations.getItem(which);
      Destination d;
      Toast.makeText(context, strName, Toast.LENGTH_SHORT).show();
      switch (destinationWay) {
        case fromDestination:
          builder.setTitle("From");
          m.setFromWaypoint(Building.getWaypointsOfDestination(m,
m.venueData.destinations[which])[0]);
          break;
        case toDestination:
          builder.setTitle("To");
          m.setToWaypoint(Building.getWaypointsOfDestination(m,
m.venueData.destinations[which])[0]);
          break;
        case highlightDestination:
          builder.setTitle("Highlight Destination");
          d = m.venueData.destinations[which];
          boolean b = Building.cameraToUnit(m, d.id, m.camera);
          if (false) return;
          Building.setHighlightStateByType(m, Unit.class, false);
          Building.setLevelByDestinationID(m, d.id);
          Building.setHighlightOnUnitsByDestinationID(m, d.id, true);
          break;
        case searchDestination:
          builder.setTitle("Search results");
          d = finalDestinationsByQuery.get(which);
          Building.setHighlightStateByType(m, Unit.class, false);
          Building.setLevelByDestinationID(m, d.id);
          Building.cameraToUnit(m, d.id, m.camera);
          Building.setHighlightOnUnitsByDestinationID(m, d.id, true);
          break;
      }
    }
  });
  builder.show();
}

private void showCategoryDialog() {
  AlertDialog.Builder builder = new AlertDialog.Builder(this);
  builder.setTitle("Select Category.");
  final ArrayAdapter<String> categoriesAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1);
  for (Category category : m.venueData.categories) {
    categoriesAdapter.add(category.name);
  }
  builder.setNegativeButton(android.R.string.no, new
DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
      // do nothing
    }
  });
  builder.setAdapter(categoriesAdapter, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
      String strName = categoriesAdapter.getItem(which);
//      Toast.makeText(context, strName, Toast.LENGTH_SHORT).show();
      Category category = m.venueData.categories[which];
      Building.unHighlightUnits(m);
      Building.highlightUnitsByCategory(m, category);
    }
  });
  builder.show();
}

private void showElementsVisibilityDialog() {
  selectedItems.clear();
  AlertDialog.Builder builder = new AlertDialog.Builder(this);
  builder.setTitle("Elements Visibility");
  String[] names = new String[]{"Labels", "Amenities", "Map Labels",
"Destination Labels", "Kiosks", "Obstacles", "Escalators", "Elevators",
"Stairs", "Units", "ParkingLots", "StreetsMajor", "StreetsMinor",
"StreetsSmallAlleys", "MallBoundary", "Background", "Restrooms", "Corridors",
"Kiosks", "HeadsUpDisplay"};
  final boolean[] states = m.getElementsActivness();
  builder.setMultiChoiceItems(names, states,
      new DialogInterface.OnMultiChoiceClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which, boolean
isChecked) {
          if (isChecked) {
            // If the user checked the item, add it to the selected items
            selectedItems.add(which);
          } else if (selectedItems.contains(which)) {
            // Else, if the item is already in the array, remove it
            selectedItems.remove(Integer.valueOf(which));
          }
        }
      });
  builder.setNegativeButton(android.R.string.no, new
DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
      // do nothing
    }
  });
  builder.setPositiveButton(android.R.string.ok, new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
      for (int i = 0; i < selectedItems.size(); i++) {
        int integer = selectedItems.get(i);
        states[integer] = true;
      }
      m.setElementsActivness(states);
    }
  });
  builder.show();
}

private void showAmenitiesVisibilityDialog() {
  selectedItems.clear();
  AlertDialog.Builder builder = new AlertDialog.Builder(this);
  builder.setTitle("Amenities Visibility");
  String[] names = new String[m.venueData.amenities.length];
  final boolean[] states = m.getAmenitiesVisibility();
  for (int i = 0; i < m.venueData.amenities.length; i++) {
    names[i] = m.venueData.amenities[i].bean.localizedText;
  }
  builder.setMultiChoiceItems(names, states,
      new DialogInterface.OnMultiChoiceClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which, boolean
isChecked) {
          if (isChecked) {
            // If the user checked the item, add it to the selected items
            selectedItems.add(which);
          } else if (selectedItems.contains(which)) {
            // Else, if the item is already in the array, remove it
            selectedItems.remove(Integer.valueOf(which));
          }
        }
      });
  builder.setNegativeButton(android.R.string.no, new
DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
      // do nothing
    }
  });
  builder.setPositiveButton(android.R.string.ok, new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
      for (int i = 0; i < selectedItems.size(); i++) {
        int integer = selectedItems.get(i);
        states[integer] = true;
      }
      m.setAmenitiesVisibility(states);
    }
  });
  builder.show();
}

private void showTextDirections() {
  TextDirectionInstruction[] textDirectionInstruction =
m.getTextDirectionInstruction();
  if (textDirectionInstruction == null) return;
  AlertDialog.Builder builder = new AlertDialog.Builder(this);
  builder.setTitle("Text Directions");

  final ArrayAdapter<String> deAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1);
  for (TextDirectionInstruction instruction : textDirectionInstruction) {
    deAdapter.add(instruction.direction);
  }
  builder.setPositiveButton(android.R.string.ok, new
DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
      // do nothing
    }
  });

  builder.setAdapter(deAdapter, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
      String strName = deAdapter.getItem(which);
//      Toast.makeText(context, strName, Toast.LENGTH_SHORT).show();
    }
  });
  builder.show();
}

private void showZonesDialog() {
  if (m.venueData.zones == null || m.venueData.zones.length == 0) return;
  AlertDialog.Builder builder = new AlertDialog.Builder(this);
  builder.setTitle("Select Zone.");
  final ArrayAdapter<String> zoneAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1);
  for (Zone zone : m.venueData.zones) {
    zoneAdapter.add(zone.zoneDetails[0].zoneName + " id:" + zone.zoneId);
  }
  builder.setNegativeButton(android.R.string.no, new
DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
      // do nothing
    }
  });
```

```java
  builder.setAdapter(zoneAdapter, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
      Zone zone = m.venueData.zones[which];
      Building.unHighlightUnits(m);
      ArrayList<Unit> units = Building.highlightUnitsByZone(m, zone);
      RectF rectF = new RectF();
      for (int i = 0; i < units.size(); i++) {
        rectF.union(units.get(i).getBBox());
      }
      m.camera.zoomTo(rectF, 50);
    }
  });
  builder.show();
}
}
```