Google Maps & GPS

Junchao Zhou Matthew Swartwout James Zhang

October 26, 2015

1 Homework 1

Homework 1 requires us to make TAKE A PICTURE button functional. After clicking this button, a camera activity should be opened which allows us to take a photo and save it to a specific directory under external storage. In order to open another activity in the current activity, we need to use Intent class to tell our current activity what operation we want to perform. In the function below

```
private  void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    File tempdir = null;
    if (takePictureIntent.resolveActivity(getPackageManager())!=null){

        try {
            tempdir = createImageFile();
        } catch (IOException e){
            e.printStackTrace();
        }
        if ( tempdir != null  ){
            takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(tempdir));
        }

        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE_CODE);
    }
}
```

MediaStore.ACTION_IMAGE_CAPTURE is a standard intent action, which can be used to have the camera application capture an image and return it. Then we need to specify a storage directory for saving the image. We use a function createImageFile() to return the storage path for the taken photo and use putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(tempdir)) to save the image to the file path specified by the Uri.

```
//create the desired image name and directory to save for the next taken photo
private File createImageFile() throws IOException {
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
    String imageFileName = "JPEG_" + timeStamp + "_";
    File storageDir = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DCIM + "/Camera");
    if (storageDir == null) {
        storageDir.mkdir();
    }
    File image = File.createTempFile(imageFileName,".jpg",storageDir);
    return image;
}
```

However, to get the thumbnails and display them on the map, we cant use putExtra() because once the image is saved by using putExtra(), then in

onActivityResult() function data.getExtras() will return null. So we need other approach to save the thumbnail.

2 Homework 2

For Homework 2, we used the application to take photos of ten different buildings around campus. For each photo, the application saves the timestamp and the GPS coordinates(latitude and longitude) into a CSV file in the DCIM folder. The Android Camera encodes each photo in data of type Intent. This data is used by the onActivityResult method

If the photo was successfully captured, the method saves the timestamp and the last available coordinate data to the CSV file. The method then sets the photo data as an imageBitmap, which is compressed in to a .PNG file. Lastly, the method adds a marker to the Google Map mMap with the coordinates for the marker position and the photo as the icon.

3 Homework 3

For getting the thumbnails, we implemented two separate methods. The first creates a marker and gets the thumbnail for a photo immediately after it is taken. The second checks to see if any thumbnails exist when the app is first launched (i.e. photos were taken with the app on a previous execution) and creates those markers.

For the first example, creating a marker immediately after the photo is taken, we added code to onActivityResult. After the CSV file is written to, we added the example code given to getExtras and extract the Bitmap. Then the file is named JPEG_TIMESTAMP_.jpg and stored in the DCIM/Camera directory. Finally, a marker is added to the map with the thumbnail at the latitude and longitude coordinates.

For the second example, creating markers that were previously taken, we added code to setUpMap. First we set up a BufferedReader from our Homework2.csv file. Then, we loop through each line of the file, knowing that each line consists of TIMESTAMP, LATITUDE, LONGITUDE. We read the TIMESTAMP and create a filepath based off that. Then we add a marker with that thumbnail, at the corresponding Latitude and Longitude for that Timestamp.

4 Homework 4

For this section we had to create a dialog to edit the Title and Snippet of the markers. The first step of this is in onCreate to create an onMarkerListener that registers when a marker is clicked and launches a MarkerInfoDialog. The MarkerInfoDialog gets the title and snippet of the marker, and creates an AlertDialog Builder and an inflater. It has an onClick listener that can detect when

a user presses the OK or Cancel button, and when the OK button is pressed it saves the edited Title and Snippet values to the marker.

5 Screenshots

6 Division of Work