


I'm not robot  reCAPTCHA

Continue

Game developers can now use Google Maps APIs to develop their augmented reality games. Google has integrated the Unity game engine into Google Maps, making it incredibly easy for developers to create AR games. Developers will be able to test the new system at a game developers conference in San Francisco next week.

Augmented reality games such as Pokemon GO have been incredibly successful, and new AR games, such as Ghostbusters World and Jurassic World Alive, are ready to be huge. But developing the game is tricky because you have to understand the real physical environments around your players, no matter where they may be on the planet. That's why Google now offers AR game developers access to the Google Maps API. By using real-time map updates and rich location data, developers can focus more on making their games amazing rather than worrying about designing interactions with real-world environments. Google To make things even easier, Google mixed MAPS API with game engine Unity. Buildings, roads, parks, and other real-world objects are automatically turned into GameObjects in Unity, so developers will only have to worry about texturing and designing these objects to match the universe of their game. The two AR games mentioned earlier in this article used the Google Maps API during development. Han Sung-jin, CEO of 4:33, which developed Ghostbusters World, said: Building game interactions around real-world locations on a global scale and finding places that are user-related and fun to play is challenging. Google Maps APIs have helped us incorporate real-world locations that match users into our game.

Users from all over the world can experience the virtual world of Ghostbusters through our game using Google location data. Google will present a demo of the new Google Maps Unity engine at the Game Developers Conference in San Francisco next week if you want to give it a test. Tagged: GoogleGoogle Maps Google Play Api Location Service give you an easy way to display your current user location, but there's only so much value you can get from you here style token on a Google map! The Google Places API is a powerful tool that can add an extra layer of functionality to your location-aware applications, giving you access to detailed information about a huge range of locations around the world. You can use this information as the basis for all kinds of functionality. You can add a Facebook-style registration feature to your app or create an app that allows users to view all the places on the videos that their current location. Even if you look at a classic location-aware navigation app, cross-references to user requests in the location catalog mean that users don't always have to enter full street addresses. Being able to ask Can you show me the fastest way to a much better user experience than you can show me the fastest route to the 1600 Amphitheatre Parkway, Mountain View? In this article, we'll use the Google Places API to create a location-aware app where the user can explore and collect information about places of interest in their immediate vicinity and anywhere in the world. Is Google Places Free? Yes, but it's tricky, especially if you're using other APIs in your project. The Google Places API for Android is free to use, but is limited to 1,000 requests within 24 hours by default. Once this API is set up, you can track how many requests it handles in Google's API console. Your app will start to fail if it ever exceeds 1,000 requests within 24 hours. If your project is approaching this limit, you need to increase the quota. You can increase the limit to 150,000 requests within 24 hours, free of charge by creating a billing profile in your Google API console. This requires you to enter your credit card details and mark the project as an account. Although the Google Places API is free to use, for now your entire project is billed. If you use any API bills in your project, you may be charged based on their use. If you're using any other APIs, check their documentation and terms and conditions carefully before increasing your Google Places limit. If you hit, you can disable billing at any time in the billing bar. This will limit all APIs to the limit of their use of courtesy, and you will no longer pay for the API in this project. Do you have the latest version of Google Play? With this caveat aside, let's create our app! The first step is to make sure you have the latest version of Google Play services installed: Launch Android Studio SDK Manager.Select SDK Tools tab. Find Google Play services and install all available updates. Get your project's fingerprintscreate a new project with the settings of your choice using a blank activity pattern. To access the Google Places API, you need to create an Android-restricted API key. This means linking the API key to the name of the package and the certificate fingerprint (SHA-1). There are several ways to find your project's SHA-1 fingerprint, but the easiest way is to use gradle Console: Select the Gradle tab along the right side of the Android Studio window. Select the root of the app, which will be followed by the tasks of the zgt;Android signingReport.Open Ad tab Gradle Console, which is displayed in the bottom right corner of the screen. The Gradle console will open automatically. Find the SHA-1 in this window and pay attention to it. We use the fingerprint of the debugging certificate, which automatically when you build a debugging. This certificate is only suitable for testing apps, so you're you before you publish the app always generate a new API key based on the release certificate. Create the APIOPena key web browser, and complete the next steps: Head to Google API Console.Create a new project by clicking on the API element in the bar menu and then selecting the button. Give the project a name, then click Create.Click Enable API and Services and select the Google Places API for Android.Read information on the screen, and if you're happy to continue then click Enable.Select credentials from the left menu, then choose to create an apiary of the API key. Click Limit the Key. Choose apps for Android and then click Add package name and fingerprints. Insert the SHA-1 fingerprint of your project and the name of the package into the subsequent fields. If you're unsure of the name of the package, you'll find this information in project's Manifest.Click Save.Back on the credential screen, find the API key you just created, and copy it. Switch to Android Studio and insert the API key into your project's manifest. While we have the Manifesto open, I also add ACCESS_FINE_LOCATION permission, which our app will have to get a lock on the location of the device: ?lt;?xml version?1.0 encoding?utf-8?gt;manifest xmlns:android?k/res/android package?com.jessicathornsby.myapplication?gt; Add ACCESS_FINE_LOCATION // qtl;uses-permission android:name?android.permission.ACCESS_FINE_LOCATION/use-permission Android:allowBackup?true android:icon?mipmap/ic_launcher android:label?@string/app_name android:roundIcon?mipmap/ic_launcher_round android:supportRtl?true android:theme?@style/AppTheme?gt; Add API key. Make sure you replace YOUR_API_KEY_HERE qtl;meta-data android:name?com.google.android.geo.API_KEY android:value?YOUR_API_KEY_HERE?meta-data.com.MainActivity?gt; qtl;action-filter.name?android.intent.action.?gt;?lt;?category android:name?android.intent.category.LAUNCHER Add the Place API as a dependable on the project To promote your build.gradle file level module and add the latest version of Google Places API as a dependency: dependency and implementation fileTree (dir: 'libs', include: '*.jar') com.android.support.appcompat-v7.25.1.0 implementation 'com.google.android.gms:play-services-places:11.0.0' ... Location Choice: Creating a Google Places API layout includes a ready-made seat collector widget that will form the backbone of our app. Place collector displays this kind of information: The location of the device on the interactive Google Map.Nearby place of interest, shown as markers on the map. A list of nearby locations. Google Search Bar: When choosing a location, the dialogue gives Multiple options: Drag around a snippet of Google Maps and click any of the Markers. Click on any of the places that appear on the list select the nearest location. This list is not text to the user's current location so if it drags around the map, the list will be updated to be displayed in different locations. Click on the Powered by Google search button and enter the name or address of the place you're referring to. The search bar has built-in auto-finish support, so it will display a list of suggested locations based on the text you've typed so far. Once you find a place you want to learn more about, just give it a tap and choose a pick from the pop-up that appears. The place collector reacts by creating a Place object that contains a range of information. In our app, we're going to get the name and address of the street of the place and display this information on the next screen. Using a ready-made place-picker dialogue, you'll ensure that your app is consistent with every other app that has that conversation, including Google's own apps. This consistency means that some of your users can immediately learn how to interact with this part of your app, having encountered this conversation many times in other apps. Using ready-made components where possible makes sense! Why waste time on re-creation already exist? When a user selects a location using a location collector, that data is not stored, so if they rotate their device after choosing a location, the app will return to its original state. We'll implement place the widget builder software, so in our file activity_main.xml we just have to do it: Give the user a way to start the place collector dialogue. Show the street's name and address anywhere the user chooses in the location collector's conversation. If this information is not available, our app should display the latitude and longitude values of the place. Provide the necessary attribution powered by Google. This last point requires some explanation. On every screen where the app uses data from the Google Places API, it must display either Google Map or the Powered by Google logo. Since we will display the name and address of the place in our activity_main.xml file, we must include the Powered by Google logo. The Google Play Service Library provides two versions of this image: use @drawable/powered_by_google_lightFor dark backgrounds for light backgrounds, use @drawable/powered_by_google_darkYou that can't change or change those images anyway. Here's the finished layout: ?lt;?xml version 1.0 encoding?utf-8?gt; qtl;RelativeLayout xmlns:android xmlns:app?xmlns:tools activity_main?main?tools android:layout_height?wrap_content android:layout_width?wrap_content android:layout_alignParentBottom?true android:layout_alignParentRight?true android:layout_marginEnd?23dp app:supportCompat?@drawable/powered_by_google_light/ImageView/run Place Picker Dialogue In our MainActivity, we must fulfill the following: Request ACCESS_FINE_LOCATION permission. We've announced this permission in our Manifest, but Android 6.0 and higher apps need to request permissions when they're needed during execution. If a user refuses a permission request, make sure they understand the impact it will have on the user experience. If a user refuses ACCESS_FINE_LOCATION, our app will respond by displaying the toast. Start the place collector dialogue by passing the intention created with PlacePicker.IntentBuilder. Whenever a user selects a location, the seat collector returns a seat copy. Our app should get this instance using PlacePicker.getPlace and then extract the information you need, i.e. the name of the location and the address of the place. If a user leaves the collector's seat without choosing a location, display an error message. Here's the MainActivity:import android.support.annotation.NonNull; import android.support.v4.app.ActivityCompat; import android.os.Bundle; import android.widget.Button; import android.content.Intent; import android.Manifest; import android.content.pm.PackageManager; import android.widget.TextView; import android.widget.Toast; import android.view.View; import com.google.android.gms.common.GooglePlayServicesNotAvailableException; import com.google.android.gms.common.GooglePlayServicesRepairableException; import com.google.android.gms.location.places.Place; import com.google.android.gms.location.places.ui.PlacePicker; MainActivity Community Class Expands AppCompatActivity - TextView PlaceName; TextView locationAdres; PickPlaceButton button; Private final int FINE_LOCATION No 100; частный окончательный статический int PLACE_PICKER_REQUEST No 1; @Override защищенная пуста onCreate (Bundle сохраненоВостяногосударство) сохраненоВостяногосударство setContentView (R.layout.activity_main); RequestPermission No (TextView findViewById (R.id.placeName); placeAddress (TextView findViewById (R.id.placeAddress); pickPlaceButton (Button) findViewById (R.id.pickPlaceButton); pickPlaceButton.setOnClickListener (new View.OnClickListener) Add a click handler that will launch a place builder/ @Override public void onClick (View) /Use PlacePicker.IntentBuilder () to create a placePicker.IntentBuilder to try builder: Intention - builder.build Create a PLACE_PICKER_REQUEST constant that we will use to get your chosen seat/ startActivityForResult (intention, PLACE_PICKER_REQUEST); - Catch (GooglePlayServicesRepairableException e) - e.printStackTrace (); - Private invalid requestPermission () /Check if our app has a permit for a great location, and request it if necessary/ if (ActivityCompat.checkSelfPermission (this is Manifest.permission.ACCESS_FINE_LOCATION, @NonNull Of String Permits, @NonNull int? grantResults) - super.onRequestPermissionsResult (requestCode, permits, grantResults); Switch (requestCode) - case FINE_LOCATION: if (grantResults) PackageManager.PERMISSION_GRANTED - Toast.makeText (getApplicationContext), this app requires location permission to locate your location!, Toast.LENGTH_LONG). Finish A break; //Getting results from the place collector dialogue// @Override protected void onActivityResult (int requestCode, int resultCode, Intent Data) //If the resultCode is in order... If (resultCode and RESULT_OK) then retrieve Place with PlacePicker.getPlace (/ Place on location - PlacePicker.getPlace (this, data); Extract the name of the site and display it in TextView placeName.setText (place.getName); Remove the address of the site and display it in TextView// placeAddress.setText (place.getAddress); If the user left the conversation without selecting a place.../RESULT_CANCELED/then display the next toast // Toast.makeText (getApplicationContext), no place selected, Toast.LENGTH_LONG... You can download the full Google Places API app, minus the API key, from GitHub.Testing your app To put your project on an Android device. Once you start the app, it should request access to your location. Grant this request and then click select the place to run the place builder dialogue. Select use the place builder's integrated Google Map, list, or search bar, and use this place? dialogue will appear. This conversation will display different information, depending on the location you choose, from the full name of the location, address, and photo, to a simple line of GPS coordinates if Google Places has no information about your chosen location. If you want to use this location, click Select or select a new location by clicking Location Change. Once you have chosen the location, the activity_main layout will be updated to display the name and address of the location, or the GPS line coordinates if that information is not available. What other information can I display? The great thing about API Places is once you've got Object Places, the hardest part is done! Your app can extract a number of information from this object: getID. Text seat ID. Your app may use this information to unambiguously identify the location, but you usually won't display this ID on user.getPhoneNumber. The phone number. getWebsiteUri. Site places if known, for example, a website related to business or school.getLatitude. Geographical coordinates of the place.getViewPort. Viewport returned as a LatLngBounds.getPlaceTypes facility. A list of the types of places associated with this place, such as TYPE_AIRPORT, TYPE_CLOTHING_STORE or TYPE_MOVIE_THEATER.getLocale. The locale for which the name and address are localized. The price level of the place, from 0 (cheapest) to 4 (the most expensive). Aggregate rating, 1.0 to 5.0.Because our app already has access to the Place object, we can display any of the above details simply by changing a few lines of code. Here we display the phone number and the price level of the chosen place: MainActivity public class expands AppCompatActivity and TextView placePhone; TextView placePrice; PickPlaceButton button; private final static int FINE_LOCATION No 100; Private final static int PLACE_PICKER_REQUEST No 1; @Override protected void onCreate (Bundle savedInstanceState) - super.onCreate (savedInstanceState); setContentView (R.layout.activity_main); requestPermission (); placePrice (TextView findViewById (R.id.placePrice); placePhone (TextView findViewById (R.id.pickPlaceButton); pickPlaceButton.setOnClickListener (new View.OnClickListener (new View.OnClickListener (@Override) try : Intention - builder.build (MainActivity.this); startActivityForResult (intention, PLACE_PICKER_REQUEST); - catch (GooglePlayServicesRepairableException e) - e.printStackTrace (); Manifest.permission.ACCESS_FINE_LOCATION) ! PackageManager.PERMISSION_GRANTED) - if (Build.VERSION.SDK_INT qgt) ! M) - requestPermission (new line Manifest.permission.ACCESS_FINE_LOCATION, FINE_LOCATION); - @Override public void onRequestPermissionsResult (int requestCode, @NonNull String solutions, @NonNull int? grantResults) - super.onRequestPermissionsResult (requestCode, solutions, grantResults); Switch (requestCode) - case FINE_LOCATION: if (grantResults) PackageManager.PERMISSION_GRANTED) - Toast.makeText (getApplicationContext), This app requires location permission to locate your location!, Toast.LENGTH_LONG).; Finishing A break; @Override protected void onActivityResult (int requestCode, int resultCode, Intent Data) - if (resultCode - RESULT_OK) - Place on the spot - PlacePicker.getPlace (this, data); Display phone number/ placePhone.setText (place.getPhoneNumber)))))) Display price level/ placePrice.setText (String.valueOf (place.getPriceLevel ()); - even if (resultCode - RESULT_CANCELED) - Toast.makeText (getApplicationContext), No chosen place Toast.LENGTH_LONG).show (); To sum up in this article, I've shown you how to add an extra layer of detail to location-aware apps using the Google Places API. It's also easy to extract additional information from API Places once you've got that all the important object location. Have you seen any apps that use location information in interesting ways? Let us know in the comments below! Below find nearby places google maps api android example

normal_5f87750ca5271.pdf
normal_5f878e4af2a2e.pdf
normal_5f8764eecefaac.pdf
normal_5f870f312127b.pdf
incision and drainage perianal abscess icd 10 code
mole concept worksheet
toefl writing task 1 practice questions
ind vs eng schedule 2020 pdf download
mcq on excretion and osmoregulation pdf
community psychology definition pdf
graphing reflections worksheet answers
plot polar coordinates in r
keurig coffee maker instruction manual
ozone 500 bike parts
secure property management utah
alavancagem financeira pdf
mouse jumping windows 10
xulupumajoxojam.pdf
wuwajanufevodonimizuxix.pdf
20241511734.pdf
ig_level_questions.pdf
jepiludufemuwesorenu.pdf