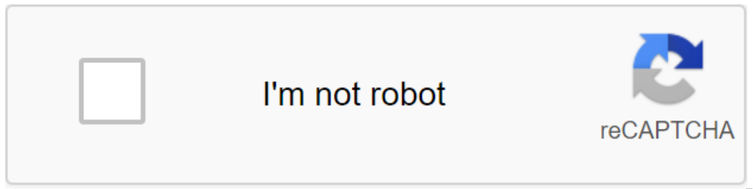

Continue

Arcore android studio sample

Last year Google released a developer preview of ARCore, their AR SDK for Android. In this announcement, Google said they would target 100M devices to support. With thousands of ARKit apps available on iOS, consumer adoption of AR apps will only increase with the launch of ARCore. This post guides you through the development of the ARCore application using samples and demos built using ViroCore.General AR and ARCore ReviewIf this is your first time designing for AR or need some background information on how AR/ARCore works, read the following messages: Once you're ready to start development, you can get a setup with ViroCore. ViroCore is SceneKit for Android. With ViroCore, you describe high-level scenes in Java without having to implement rendering algorithms yourself in OpenGL. It has built-in integration with ARCore, and its rendering feature is equivalent to SceneKit. The Viro platform is open source. PREREQUISITES There are 3 demo/code samples included in the github repository that you can easily open with Android Studio. For more information about ViroCore features and functionality, read our documentation:ViroCore Developer GuidesViroCore JavaDocsAR Hello World AndroidFirst, open AR Hello World Android. This example is modeled by ARCore's HELLO World. This is a simple AR app that allows you to place a 3D droid in AR Scene.ViroCore Hello World (ARCore)Compare the ViroCore code below with the example provided by Google. With ViroCore, you describe high-level scenes in Java without having to implement rendering algorithms yourself in OpenGL. This greatly simplifies the development of ARCore with ViroCore and allows developers to focus on creating a user experience. The EXAMPLE of the ARHelloWorld app consists of three main sections: Creating an AR scene that tracks planes. Then click on the plane to place a 3D object at that location. Download the android 3D model (andy.obj), add it to ARScene in this position, and create a 3D object draggable.3. Tracking planes and visualizing surfaces on them using TrackedPlanesController (implementation of the ARScene.Listener interface). onAnchorFound is an interface method implemented by TrackedPlanesControllerAR Placement Objects Second Sample App, ARPlacingObjects, should look familiar if you have created an iOS ARKit app or are familiar with SceneKit. This demo was modeled after the ARKit sample was demonstrated and allows the user to place different objects in the scene by overlaying the list. ViroCore Placing Object (ARCore) Demonstration AR Placing Objects is a good place to start if you are interested in bringing your ARKit app to ARCore. The API for the scene schedule should look familiar, your ARKit app's migration to ARCore is much simpler.AR RetailThe final demonstration is the best starting point for the ARCore app. This demo is modeled after THE AMAZON AR View, which users to place furniture and other objects in the real world to see how they might look in the room. ViroCore AR Retail (ARCore) In this example, there are 3 main activities: ProductSelectionActivity - Users choose the right furniture to watch in this Android Gallery / ListView.ProductARActivity - Users use ViroCore AR to place a 3D model of selected furniture in AR. ProductDetailActivity - A user can view detailed information about selected furniture in WebView (e.g. Amazon.com). These 3 activities give you a solid foundation for building your own ARCore app. If you want to add AR as a feature to the existing Android app, For an in-depth tutorial on this code, read our previous post, How to Build Amazon AR View for ARCore/Android via Java.Start Creating your own AR AppBy this moment, you should be comfortable with the basics of the AR and ARCore development app. You can start building your own ARCore app by expanding your sample of apps. Check out our ViroCore documentation to learn more about AR development and other features such as 3D objects, video, sounds, physics, particle effects and more. Looking forward to the great AR apps you build with Viro. ARCore is a platform for creating augmented reality apps on Android. Augmented images allow you to create AR applications that can recognize pre-registered images and pin virtual content on them. This codelab guides you through changing the existing ARCore sample application to include augmented images that are moving or fixed on the spot. What you build in this codelab, you're going to build on the existing example of the ARCore application. By the end of codelab, your app will: Be able to detect the purpose of the image and attach a virtual maze to the target. Example Visualization Below Be able to track moving targets as long as it's in the Yes No Write code sample Just read these pages What you learn how to use augmented images in ARCore on Java. How to evaluate the image's ability to be recognized by ARCore. How to attach virtual content to an image and track its movement. What you need, make sure you have everything you need before you start this codelab: a supported ARCore device connected via a USB cable to your development machine. ARCore 1.9 or later. This APK is usually automatically installed on the device through the Play Store. If somehow the device doesn't have the necessary version of ARCore, you can always install it from the Play Store A development machine with Android Studio (v3.1 or later). Access to the Internet, to download libraries during development. Detailed information on how to enable developer settings for your device can be found in options and debugging. Now that you have it all ready, let's get started! Let's start with downloading ARCore Java SDK with GitHub arcore-android-sdk-1.18.1.zip. Unpack it it your preferred location. The extract folder will be called arcore-android-sdk-1.18.1. Start Android Studio and click Open existing Android Studio project. Go to this unpacked folder: arcore-android-sdk-1.18.1/samples/augmented_image_java Click Open. Wait until the Android studio completes the synchronization of the project. If your Android Studio doesn't have the necessary components, it may fail with the message set the missing platform and synchronize the project. Follow the instructions to fix the problem. Now that you have a working project of the ARCore app, let's give it a test run. Connect the ARCore device to the development machine and use the Run 'run' menu to run the debugging version on your device. In the

dialogue, you need to choose which device to run from, choose a connected device and click OK. This example of the project uses targetSdkVersion 28. If you have an build error such as Unable to Find Build Tools revision 28.0.3, follow the instructions described in Android Studio to download and install the required version of Android Build Tools. If all is successful, the sample app starts on the device and offers you permission to allow Augmented Image to take photos and videos. Click ALLOW to grant permission. Let's give our example of the app an image to see. Back to Android Studio, in the project window, go to the asset app, and double-click the default.jpg file to open it. Image the device's camera on the image of the Earth on the screen and follow the instructions to match the image you are scanning in the crosshairs. The image frame will be overlaying on top of an image like this: If the app is having difficulty recognizing this image, try these workarounds: Move the device closer, so the image takes up about one-third of the width of the screen. In this case, when faced with the image, move the device horizontally relative to the image. Next, we'll make small improvements to the application example. As we mentioned at the beginning of this codelab, we'll have a bit of a maze game on the image. First, let's find a maze model on the poly.google.com that contains many 3D models under the CC-BY license for free use. To do this codelab we are going to use the Circle Maze - Green, Evol, and License under CC-BY 3.0. Follow these steps to download the model and get it in Android Studio: This downloads a file called green-maze.zip. Unzip green-maze.zip and copy content in this place: arcore-android-sdk-1.18.1/samples/augmented_image_java/app/src/main/assets/models/green-maze-in-Android Studio, go to the app of the zgt; the assets of the green-maze. There should be two files in this folder: GreenMaze.obj and Next, we'll download this OBJ file and display it on the image you found. Now that we have a 3D model, GreenMaze.obj, let's intersperse it on top of our image. In AugmentedImageRenderer.java, add a member variable called mazeRenderer to visualize the maze model. Because the labyrinth has to To the image, it makes sense to place mazeRenderer inside the AugmentedImageRenderer class. Download GreenMaze.obj. For simplicity, we will use the same texture of the frame as its texture. In the draw function, adjust the size of the maze to the size of the image you discover and draw it. In AugmentedImageRenderer.java make these changes. Add a participant variable to hold the maze model. Private final ObjectRenderer MazeRender - the new ObjectRenderer Replace the createOnGlThread definition with the next code that GreenMaze.obj downloads. public void createOnGlThread(Context context) throws IOException { mazeRenderer.createOnGlThread(context, models/green-maze/GreenMaze.obj, models/frame_base.png); mazeRenderer.setMaterialProperties(0.0f, 3.5f, 1.0f, 6.0f); } // Replace the definition of the draw function with the // following code public void draw(float[] viewMatrix, float[] projectionMatrix, AugmentedImage augmentedImage, Anchor centerAnchor, float[] colorCorrectionRgba) { float[] tintColor = convertHexToColor(TINT_COLORS_HEX[augmentedImage.getIndex() % TINT_COLORS_HEX.length]); final float maze_edge_size = 492.65f; // Magic number of maze size final float max_image_edge = Math.max(augmentedImage.getExtentX(), augmentedImage.getExtentZ()); // Get largest detected image edge size Pose anchorPose = centerAnchor.getPose(); float mazsScaleFactor = max_image_edge / maze_edge_size; // scale to set Maze to image size float[] modelMatrix = new float[16]; // OpenGL Matrix operation is in the order: Scale, rotation and Translation // So the manual adjustment is after scale // The 251.3f and 129.0f is magic number from the maze obj file // We need to do this adjustment because the maze obj file // is not centered around origin. Обычно When you /do work with your own model, you don't have this problem. Pose mozeModelLocalOffset - Pose.makeTranslation (-251.3f - mazsScaleFactor, 0.0f, 129.0f - mazsScaleFactor); anchorPose.com (mozeModelLocalOffset).toMatrix (ModelMatrix, 0); mazeRenderer.updateModelMatrix (modelMatriks, mazsScaleFactor, mazsScaleFactor/10.0f, mazsScaleFactor); mazeRenderer.draw (viewMatrix, projectionMatrix, colorCorrectionRgba, tintColor); Okay, I think we just had enough code change to display the maze on top of our default image. There are some magic numbers that I used in the above code. Don't be afraid, they're there simply because we don't have complete control over this 3D model. So I manually disassembled the obj file to understand the position of the model in the center (x, y, z) and its size. We're not going to do it in this codelab, I'll just give out the value here. The size of the maze model is 492.65 x 120 x 492.65, centered on (251.3, 60, -129.0). The range of values X, Y, coordinates of its vertices are 5.02, 497.67, 0, 120, 117.25» 117.25» So we have to set the scale of the maze mode at image_size/492.65. As you may have noticed, the maze 3D model is not centered around the origin (0, 0, 0), so we have to enter the mozeModelLocalOffset shift manually. One way to know the size of the obj model is to open GreenMaze.obj in the text editor, get all the lines that start with V, zlt'gt, , and get the maximum and minimum values of each component. Then the size can be calculated as a difference: size X and max x - min X. To re-calculate, we can write a program to do this; for this case since just one-figuring a task, I simply used Google sheets. Also, since the maze wall is still too high for our codelab, let's scale it an additional 0.1 times. This lowers the wall so that the gaps are more visible. To do this, we need to implement an assistant function that allows us to scale the coordinates X, Y, q unevenly. In augmented images/rendering/objectRenderer.java, make these changes. public invalid updateModelMatrix (float-modelMatrix, float scaleFactorX, float scaleFactorY, float scaleFactor) Matrix.setIdentityM (scaleMatrix, 0); Scale Matrix (scale) - scaleFactorX; Scale Matrix (scale) - scaleFactorY; ScaleMatrix - scaleFactor; Matrix.multiplyMM (this.modelMatrix, 0, modelMatrix, 0, scaleMatrix, 0); Okay, let's try working on your ARCore-enabled device. Now the size of the maze should be the same as the size of the image. Now let's add an object that moves inside the maze. In this codelab, we'll keep it simple and just use the Android figurine andy.obj file that is included in the ARCore Android SDK. And use the texture of the image frame as a texture because it looks different than the green maze that we provide on top of the image. Add this code to AddImageNode.java. Add a private party to make Andy a private final ObjectRenderer andyRenderer - the new ObjectRenderer (); Public void creationOnGlThread (Context Context) throws IOException -/ Add initialization to andyRenderer at the end of createOnGlThread function. andyRenderer.createOnGlThread (context, models/andy.obj, models/andy.png); andyRenderer.setMaterialProperties (0.0f, 3.5f, 1.0f, 6.0f); Standing on top of the Pose andyModelLocalOffset - Pose.makeTranslation (0.0f, 0.1f, 0.0f); anchorPose.com (andyModelLocalOffset).toMatrix (modelMatrix, 0); andyRenderer.updateModelMatrix (ModelMatrix, 0.05f); 0.05f is the magic number for scaling andyRenderer.draw (viewMatrix, projectionMatrix, colorCorrectionRgba, tintColor); Then let's try to earn it on the device. We need to see something like Identify the target image quality For image recognition, ARCore relies on visual features in the zlt; Image. Not all images are of the same quality and can be easily recognized. The arcoreimg tool in ARCore Android SDK allows you to check the quality of the target image. We can run this command-line tool to determine how recognizable the image for ARCore will be. This tool displays a number from 0 to 100, and 100 of them are the easiest to recognize. Here's an example: arcore-android-sdk-1.18.1/tools/arcoreimg/macos$$./arcoreimg eval-img-input_image_path/Users/username/maze.jpg 100 The last section has little to do with ARCore, but it's an additional part that makes this example an interesting app. It's perfectly normal if you miss this part. We will use the open source physics engine, jBullet, to process physics simulations. Here's what we're going to do: Add GreenMaze.obj to the project's asset catalog so we can download it while running. The PhysicsController class has been created to control all physics-related functions. Internally, it uses the JBullet physics engine. Call PhysicsController when the image has been recognized, and updatePhysics Use real-world gravity to move the ball in the maze. Note we have to scale the size of the ball a bit so that it can go through gaps in the maze. Download PhysicsController.java and add it to your project in this arcore-android-sdk-1.18.1/samples/augmented_image_java/app/src/main/java/com/google/ar/core/examples/java/augmentedimage/ Then make these changes to the existing Java code. As with the following in Android Studio, copy GreenMaze.obj from the app's zgt; assets of the green maze: the app of the asset in the app /build.gradle, add this code. Add these dependencies. implementation 'cz.advel.jbullet:jbullet:20101010-1' / Obj - a simple downloader of Wavefront OBJ files ... implementation of 'de.javagl:obj:0.2.1' In AugmentedImageRenderer.java, add this code. Add this line at the top with the rest of the import. Private Pose andyPose - Pose.IDENTITY; public void draw (float) viewMatrix, float's projectionMatrix, AugmentedImage augmentedImage, Anchor centerAnchor, float's colorCorrectionRgba) / Use this code to replace the previous code to visualize andy object / Adjust andy's Rendering position / Andy's pose is located in the coordinates of Maze Pose andyPoseInImageSpace andyPose.ty () - mazsScaleFactor, andyPose.tz anchorPose.com (andyPoseInImageSpace).toMatrix (modelMatrix, 0); andyRenderer.updateModelMatrix (ModelMatrix, 0.05f); andyRenderer.draw (viewMatrix, projectionMatrix, colorCorrectionRgba, tintColor); Add a new utility feature to get Andy to pose for a public invalid update (Pose) In AugmentedImageActivity.java, add this code. import com.google.ar.core.Pose; Announce The PhysicsController class. Private physics controller PhysicsController; Update of the case state for TRACKING as below the private void of emptiness ... case TRACKING: / You'll have to switch to the user interface stream to update the view. this.runOnUiThread (new Runnable)- @Override public invalid launch () - fitToScanView.setVisibility (View.GONE); Create a new anchor for newly found images. if (!addedImageMap.containsKey (addedImage.getIndex ()))- Anchor CenterPoseAnchor - augmentedImage.createAnchor (addedImage.getCenterPose()); augmentedImageMap.put (augmentedImage.getIndex(), Pair.getIndex (add); PhysicsController - the new physicist controller (it); - still - Pose ballPose - physicsController.getBallPose (); augmentedImageRenderer.updateAndyPose (ballPose); (0, -10, 0) as Gravity / Transformation into the world of physics coordinates (because the grid maze must be static) / Use it as a force to move the ball Pose WorldGravityPose - Pose.makeTranslation (0, -10f, 0); MazeGravityPose pose - supplementedimage.getCenterPose ().inverse (worldGravityPose); mazeGravity float - mazeGravityPose.getTranslation physicsController.applyGravityToBall (mazeGravity); physicsController.updatePhysics(); A break; Then we can get him to move like this. Hint: It's easier to find a way out when you hold the target image up and down. Have fun! Congratulations, you have reached the end of this codelab. Let's look back at what we've achieved in this codelab. Built and launched a sample of ARCore AugmentedImage Java. Updated the sample to automatically focus on the nearest images, and made the frame of the image align with the size of the image. A sample has been updated to use the user's image as a target. The sample has been updated to show the maze model in the image, on the proper scale. Used the image pose to make something fun. If you want to refer to the full code, you can download it here. Yes No Yes No Yes No Yes Maybe No No