|  |
| --- |
|  |
| UBC eHealth Summer Camp |
| App #2: EMR Plug-In: Remote Patient Observation App |
|  |



|  |
| --- |
| © UBC eHealth Summer Camp; Matthew Toom; 2011 |

## EMR

An electronic medical record (EMR) is a computerized medical record created in an organization that delivers care, such as a hospital or physician's office.

## Case Study

A patient has travelled to some exotic jungles in the world. When she comes home, she presents to a clinic with a very unusual purple rash, ostensibly caused by an infection. The doctor has no idea what the rash is, so the patient is referred to a dermatologist (i.e. a doctor who specializes in skin problems). A week later, the patient comes into the dermatologist’s office but the rash is now gone! The patient tries to describe the rash but the dermatologist doesn’t know what to think because the description is not good enough. The dermatologist and is now worried that it will be difficult to determine what kind of infection the patient has!

## EMR Plug-In: Remote Patient Observation App

We are going to turn our Google Nexus One into a diagnostic device that can record observations about a patient and add data to their medical file. The idea of this project is that a patient or doctor can take photos that constitute medical observations. The App then links the images into a patient’s EMR, like a plug-in.

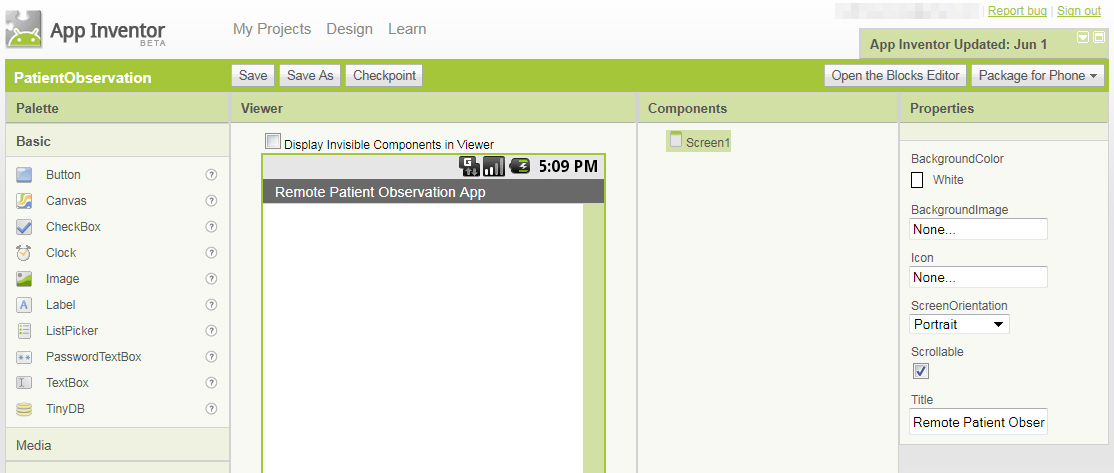
In order to do this, we will need to work with web-based components in the Google AppInventor as well as the Google Nexus One’s camera device!

**Login to AppInventor & Create a Project**

Log into AppInventor at <http://appinventor.googlelabs.com/> and create a project called *PatientObservation*.

## Set Title for the Project

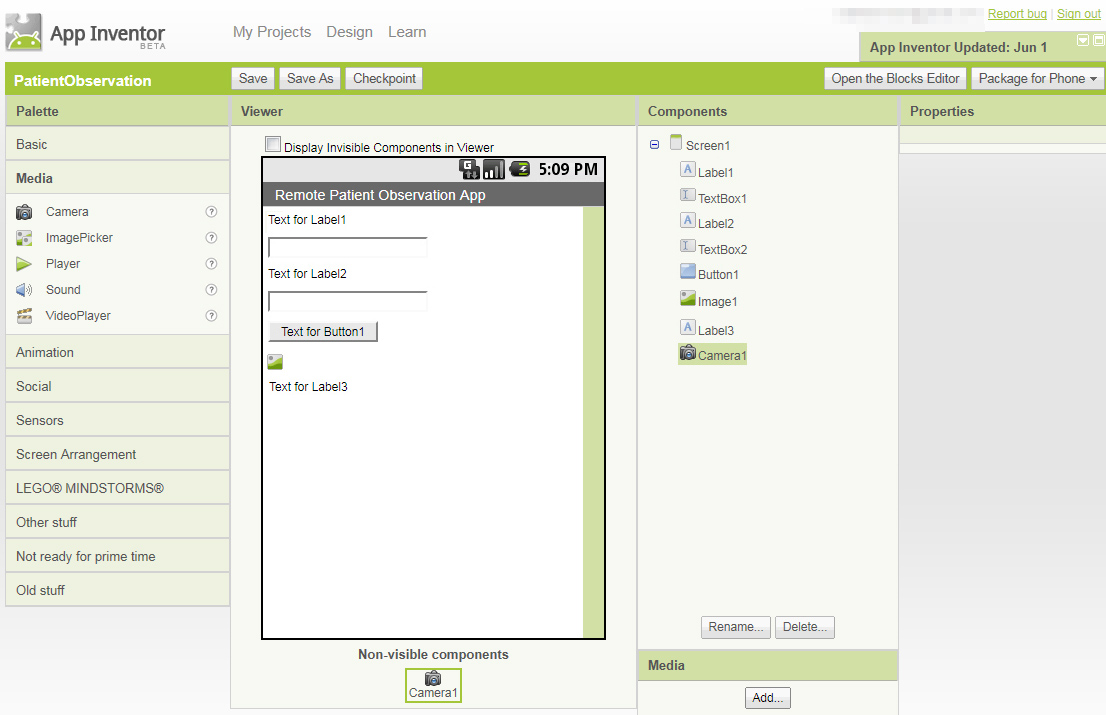
Under *Components*, select *Screen1* and then change the title to *“Remote Patient Observation App”.* Set *ScreenOrientation* to *Portrait*.



## Set up the Design

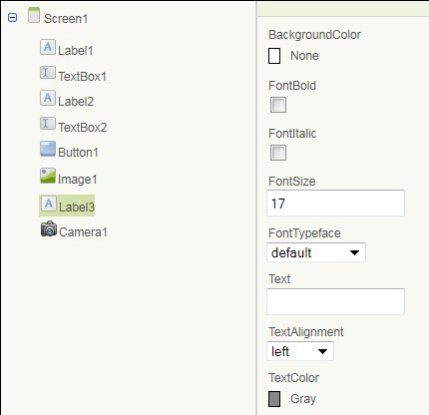
We want to build an App that will allow the user to write about an observation for a patient and then take a photo so submit to their medical record.

The first step to doing this is to designing the user interface. We will need to add a *Label*, *TextBox*, *Label*, *TextBox*, *Button*, *Image* and *Label* in that order, from top to bottom. Add each item by dragging it form the *Basic* section of the palette into the canvas. Finally, from the *Media* section of the canvas, drag in a *Camera* object. It is a *Non-Visible* object. Now, you should have the following:

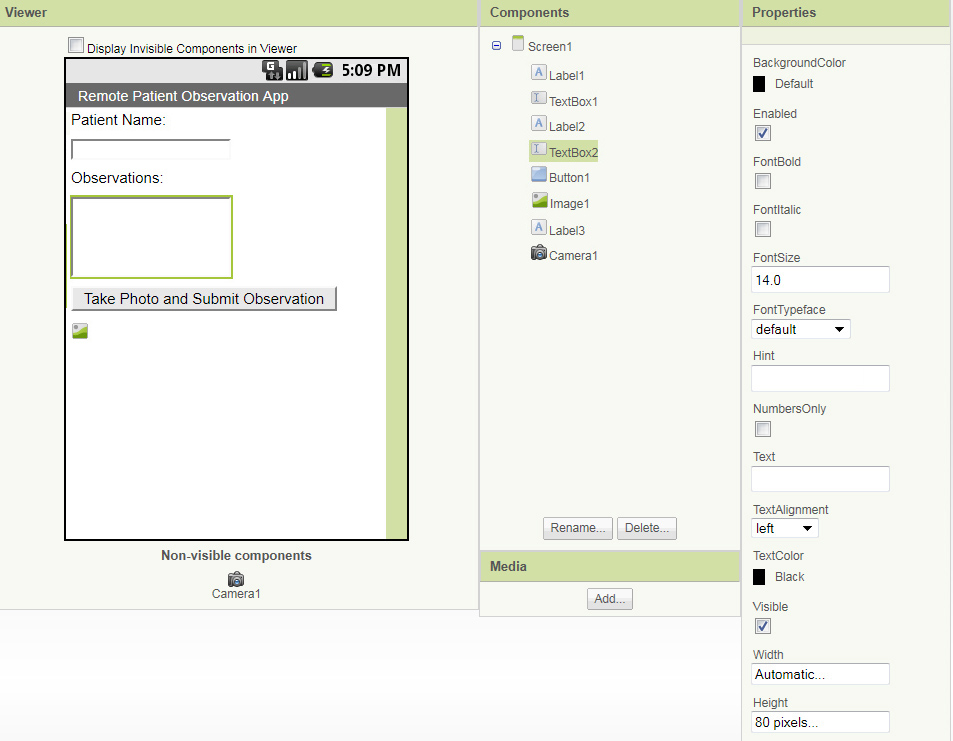


Notice the *Camera* object in the figure above.

In the image above, we will need to change the caption of the *Label* objects and the caption of the *Button*. The first two labels will read “Patient Name:” and “Observations:”. The *Button* should say “Take Photo and Submit Observation”. If you want, increase the TextSize of all your *Label* objects and your *Button* up to 17, from the default of 14.

The last *Label* object should be set to have a blank caption (i.e. *Text* attribute) and the *TextColor* attribute should be set to *Gray*. This last *Label* will be used to display the status of our communication with the server. *(See the example to the right).*

For both *TextBox* objects, set the *Hint* attribute to empty. For the second *TextBox* only, set the *Height* to 80 pixels as is shown below:



## Load the Blocks Editor

Now we are ready to set up the programming for the App! Click on *“Open the Blocks Editor”* and we will begin the programming steps.

## Phase 1

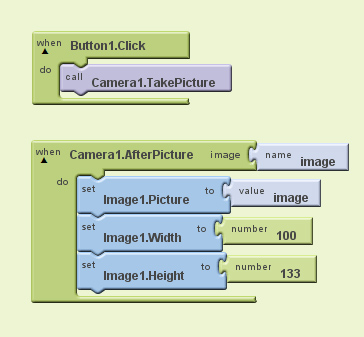
We will begin by hooking up the phone’s camera so that you can take a picture and load it into your App.

## Setting up the Camera

It’s really easy to set up the phone’s camera with your App. Add the following puzzle pieces into the Blocks Editor:

|  |  |  |
| --- | --- | --- |
| Puzzle Piece | Path | What Does it Do? |
| Button.Click | My Blocks > Button1 | Capture the event when your Button is clicked. |
| Camera.TakePicture | My Blocks > Camera1 | Tells the App to take a picture using the Camera. |
| Camera.AfterPicture | My Blocks > Camera1 | Handles what to do with the picture once it is taken. |
| set Image.Picture | My Blocks > Image1 | Sets the Image’s picture. |
| set Image.Width | My Blocks > Image1 | Sets the Image’s width. |
| set Image.Height | My Blocks > Image1 | Sets the Image’s height. |
| image value | My Blocks > My Definitions | The image data. |
| number | Built-In > Math | A number. |

Now, you can set the pieces up so that everything looks exactly like this:



## Test your App

Now, try testing the App by going to “*Connect to Device…”* and selecting your phone’s serial number from the drop-down list. When the App starts, click the *Button* and then use the Google interface to take a photo. If you are happy with your picture press OK. The phone then transfers the image back to your App which displays it on the screen!

## Phase 2

Now that we are taking pictures with the phone and the Design for the interface is set up, let’s transfer the data to a server it can be viewed remotely.

## Set up the Web Programming to Upload the Image

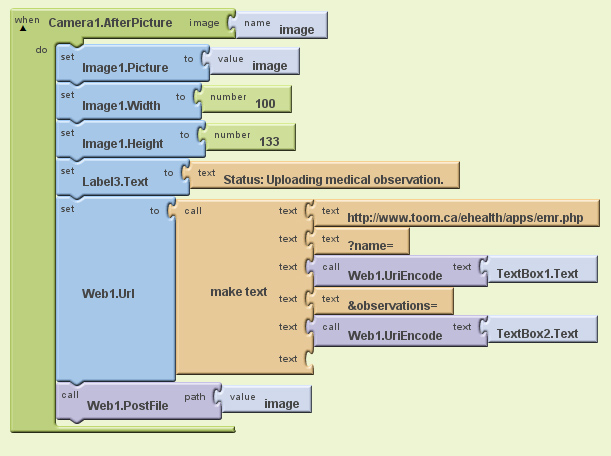
It’s really easy to set up the phone’s camera with your App. This time, we won’t show you exactly where to get all the pieces, so you will have to find them on your own!

Go back to the Designer and add a *Web* object (which you can find under “Other Stuff” in the palette). To add it, drag it and drop it onto your canvas.

Now you will need to update your *Camera1.AfterPicture* puzzle piece so that it does the following new tasks:

1. Updates the status label to tell the user that the medical observation is uploading.
2. Sets the URL of the *Web* object, which is used to transfer the name and observation about the patient. You must use *UriEncode* calls here!
3. Finally, it uploads the actual image to the server.

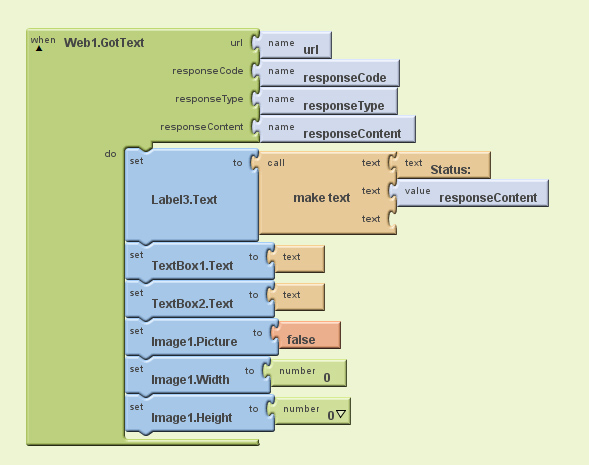
To do this, set up the following blocks:



Alright, now when the upload is complete, we want to do the following tasks:

* Notify the user that the upload has been completed using the status *Label*.
* Remove text from the Name *TextBox* and the Observations *TextBox*.
* Remove the Image from the screen of the phone.
  + To do this, we set the *Image.Picture* value to false.
* Set the *Width* and *Height* of the Image to zero.

In order to do these tasks, add in a *Web.GotText* object which will let us know when the upload is complete. Here is exactly how you do it:



## Test your App

Now, try testing the App by going to “*Connect to Device…”* and selecting your phone’s serial number from the drop-down list. When the App starts, fill out your patient’s name and give some information about the observations. Then, take a photo of your patient. The phone will automatically upload your observations complete with an image! You should get a message that confirms your upload was completed successfully. Otherwise, you will get an error message. To view the observations in your web browser, go here:

<http://www.toom.ca/ehealth/apps/emr_view.php>