Software Development Proposal

Prepared for: Kaplan Early Learning Company

Prepared by: Duane McMahon (Acme Corporation)

24/11/2017

Acme Corporation, a software development (IT) business with an address at 5 Alphabet Street, McKinnon, Victoria (hereinafter also referred to as “Acme”) is excited to provide this software development proposal to Kaplan Early Learning Company, a leading international provider of products and services that enhance children's learning (hereinafter referred to as “Kaplan”) in support of Kaplan Early Learning Company’s *Computer Vision* project (hereinafter referred to as the “IT Project”).

By the signature of Acme Corporation’s authorised representative hereunder, this software development proposal constitutes Acme Corporation’s formal offer to provide the services and/or deliverables described herein on the terms and conditions described herein.

This IT Project proposal will be valid for a period no longer than 8 weeks following the date first set forth above.

Acme Corporation

Duane McMahon

24/11/2017

________________________________ ________________

Duane McMahon
Technical Lead

1. Project Overview

The IT Project is intended to provide early English language learners with a simple game to play at any fresh fruit and vegetable store. The elements of the game are four words and four icons. Each word is the name of a fruit and each fruit has its own icon. The words and images will be paired when a fruit that has been currently tasked for identification is present in the camera view. The fruit may or may not be tasked in any order ('randomly') for any game. The game is completed successfully when each fruit has been paired with its representative icon.

Assumptions and Constraints.

The project requires approximately eight weeks to complete. The purpose of the program is to demonstrate proof-of-concept (i.e. of automated image recognition) on any recently manufactured smartphone running android lollipop or above.

2. Obstacles

The primary obstacle is the developer's lack of experience building applications on the Android mobile platform (or any other mobile platform). The developer has NO experience building mobile apps.

3. Technical Obstacles

OpenCv

The primary technical obstacle has been identified as the integration of the computer vision framework into the android code base and making use of the framework's application programming interface to control the camera and process camera frames by applying trained pattern recognition algorithms.

'CameraBridgeViewBase' is the basal class in OpenCv for implementing interaction with the camera. The application grabs preview frames from the camera in real time, and then renders them in full screen mode using either OpenCV's Java or native camera API (the framework also allows the user to switch between the Java and native modes). This project uses the Java wrapper (OpenCv android sdk) to call OpenCv native (c++) code.

A decision will need to be made as to whether to use trained classifiers or to train the classifers(to recognize a banana, orange, etc.). Such a decision belongs properly to the development phase of the project.

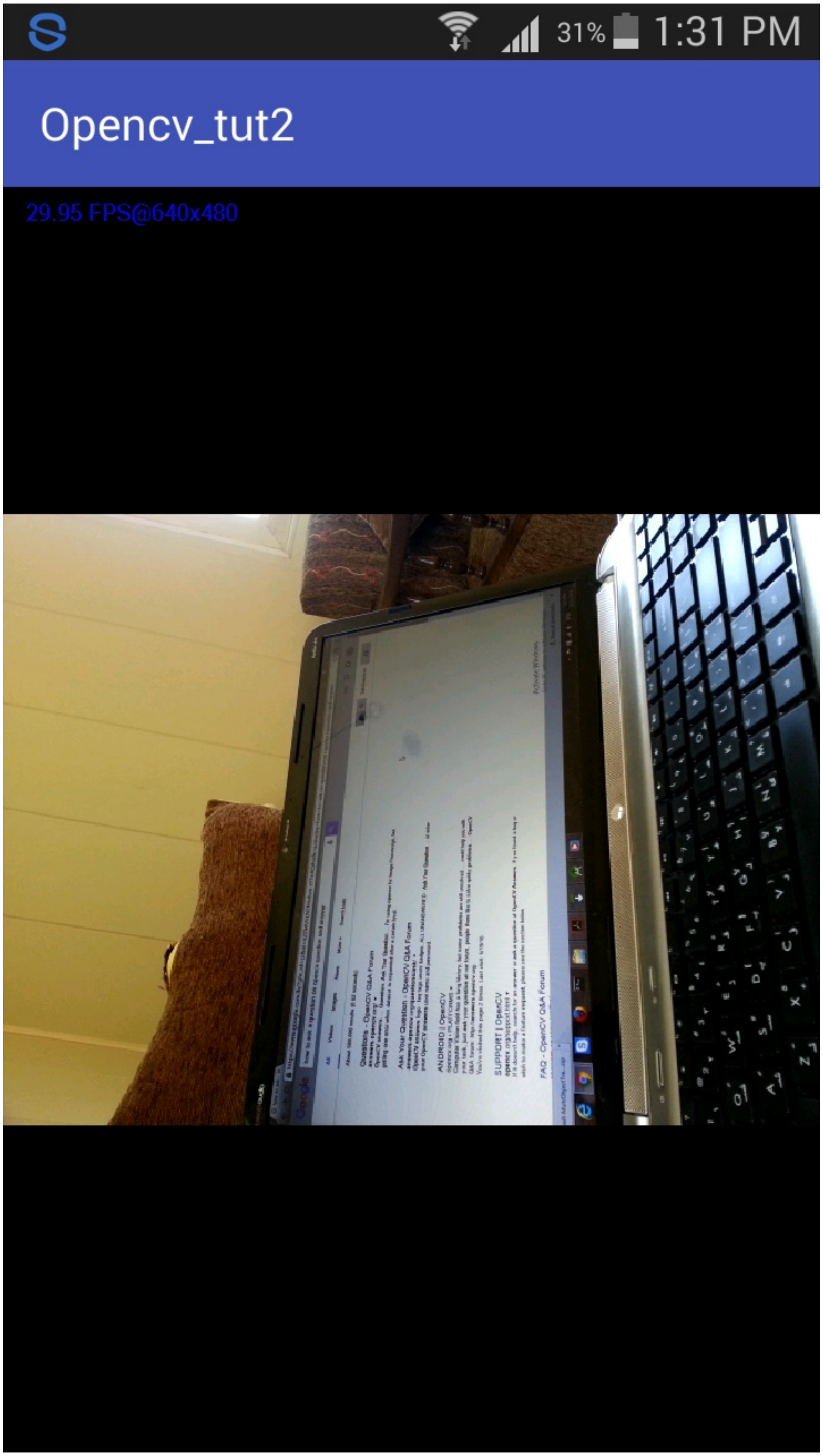
Device Capability Requirements

Please note the following special permissions in the AndroidManifest.xml file that allow access to the device camera:

<uses-permission android:name="android.permission.CAMERA"/>
<uses-feature android:name="**android.hardware.camera**" android:required="false"/>
<uses-feature android:name="**android.hardware.camera**.autofocus" android:required="false"/>
<uses-feature android:name="**android.hardware.camera**.front" android:required="false"/>
<uses-feature android:name="**android.hardware.camera**.front.autofocus" android:required="false"/>

The significance of the xml attributes highlighted above is that the OpenCv framework uses the deprecated (but still most commonly used) android 'camera' to control the camera.

An anomaly that anyone developing an android app using OpenCv will quickly be challenged by, is that by default the image on the screen will appear rotated to the left by 90 degrees:



The OpenCv class that controls the camera is deliberately designed to appear in a landscape orientation, and the fact will have ramifications for UI design for this project. Decisions in relation to this belong properly, however, to the development phase of the project.

4. Industry and Market Risks

The Google Play store is filled with apps which are free to use; the possibility of a novice programmer offering something unique to the market that is consequently a money spinner and that may be designed, coded, and realised within an eight week time frame is - not the intention of the developer. The IT Project is intended to be a free, proof-of-concept software application designed for the purpose of demonstrating an android phone performing image recognition.

Please note that the developer acknowledges *convolutional neural network* algorithms (i.e. Deep Learning algorithms) outperform the *cascade classification* algorithms applied by OpenCv. Presumably the framework will eventually incorporate recent developments in the area of computer science.

5. Budgetary Risks

There are no budgetary risks. All tools used to develop the project are free (no capital expenditure required).

Technology Requirement (Software and Hardware)

6. Hardware

The hardware upon which the IT Project is developed is a Motorola Moto Z Play running the Nougat android platform.

7. Software

The open source OpenCV computer vision framework version 3.3.1.
The minimum SDK targeted by the application is **16**.
The compiled SDK version is **26**.
The IDE used for development is Android Studio.
Balsamiq was used to generate mockups.
Visual Paradigm was used to generate class diagrams from project code.

8. Milestones

1. **Requirements**

   Proposed Mobile App Introduction/Description (e.g. proposal document).
   Device Capability Requirements (e.g. features in Manifest.xml)
   User Interface Prototype Documents (e.g.. mockups)

2. **Analysis**

   Analysis Classes (i.e. class diagrams with brief Description)

3. **Development**

   Components
   Builds and Executables

4. **Testing**

   As required.

5. **Deployment**

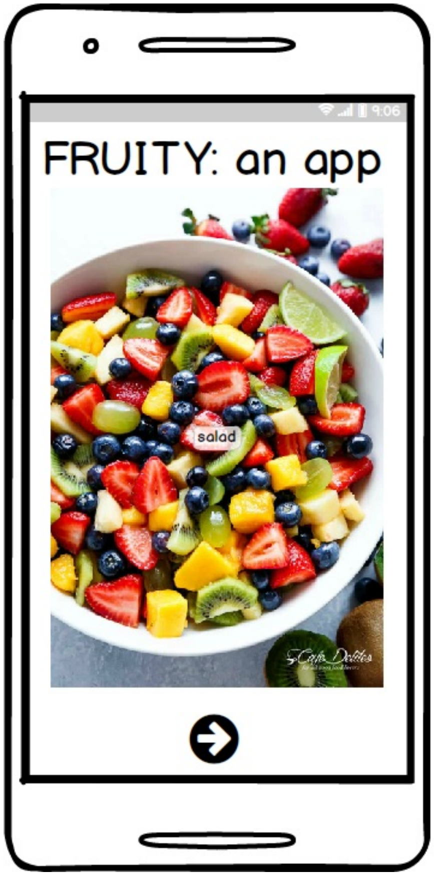
   The program will be demonstrated to a restricted audience late January 2018 using a release-ready APK file that is signed and hosted at http://www.droidbin.com/

MILESTONE	TASKS	DATE
Requirements (Assessment Item 01)	Proposal Document including device requirements, UI mockups.	24/11/2017
Analysis (Assessment Item 01)	Class Diagram	24/11/2017
Development	Code and test the project	24/01/2018 (???)
Deployment (Assessment Item 04)	Final presentation and demonstration	TBA

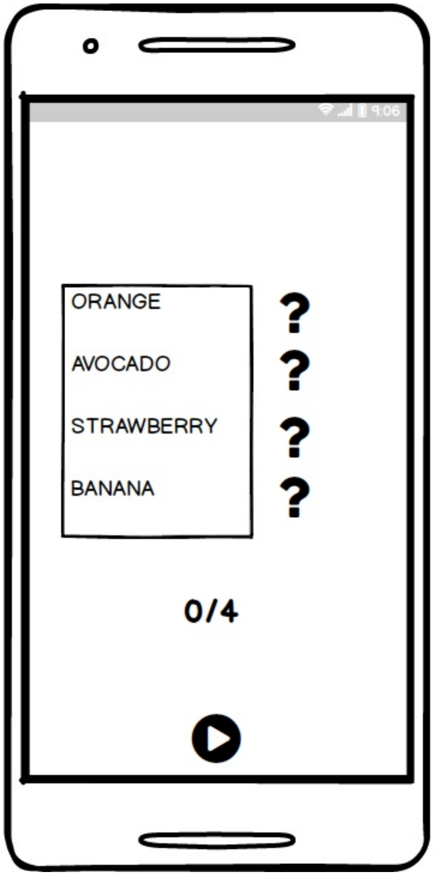
9. Documentation

Mockups of the user interface:

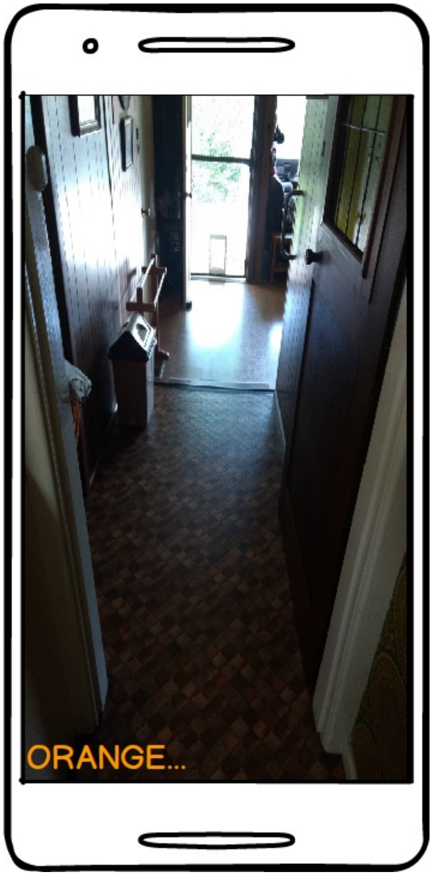
[#1]



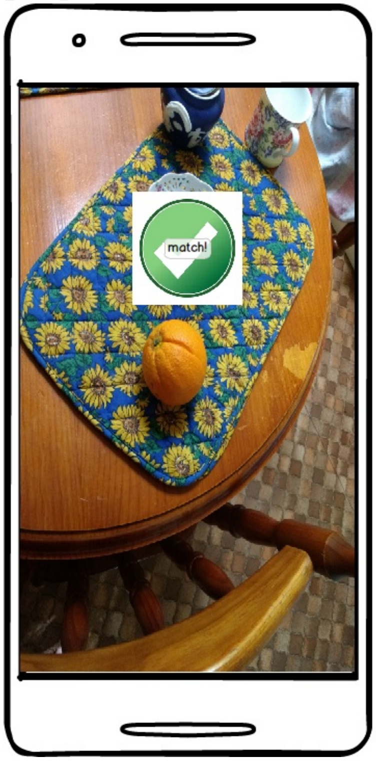
[#2]



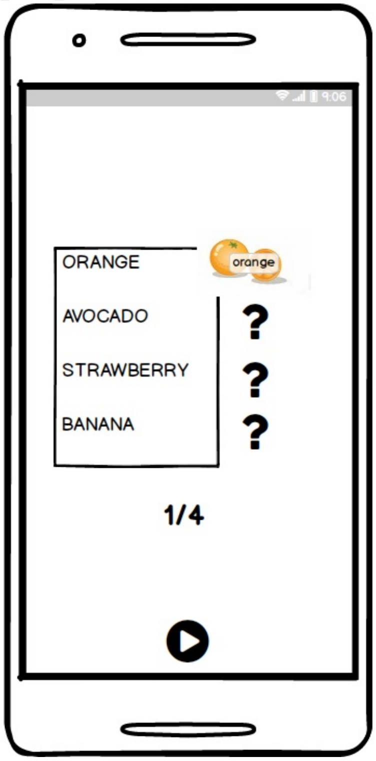
[#3]



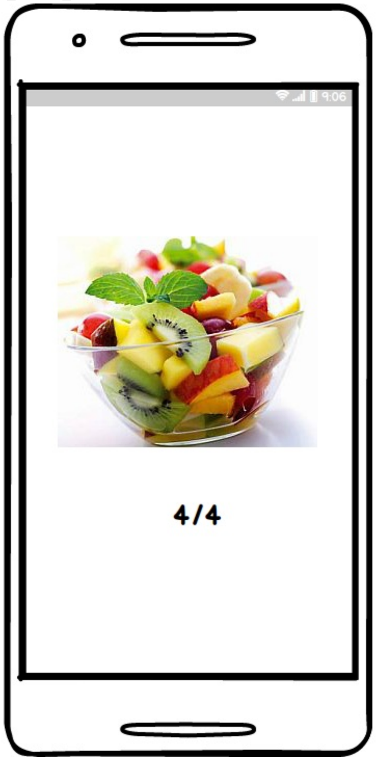
[#4]



[#5]

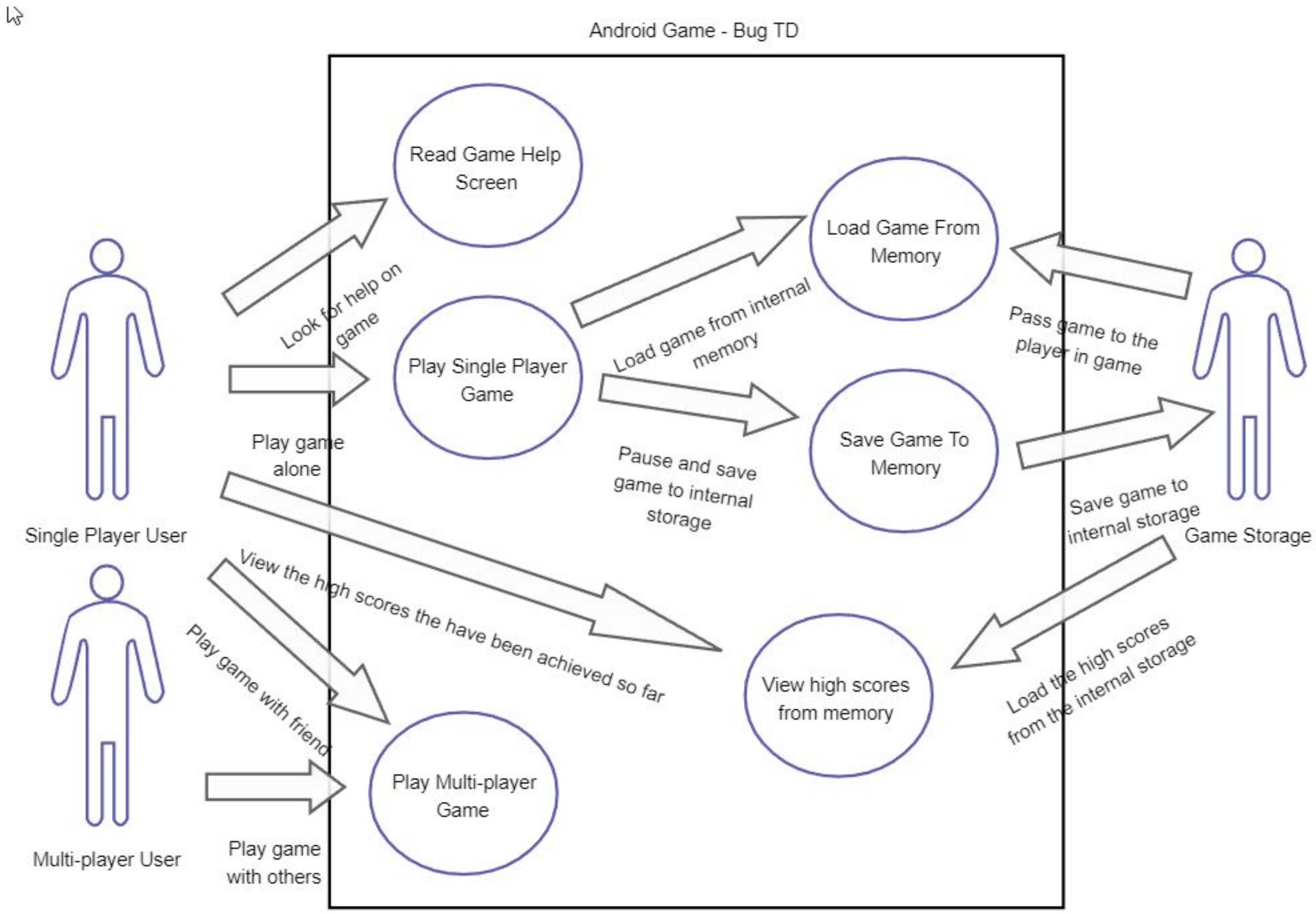


[#6]



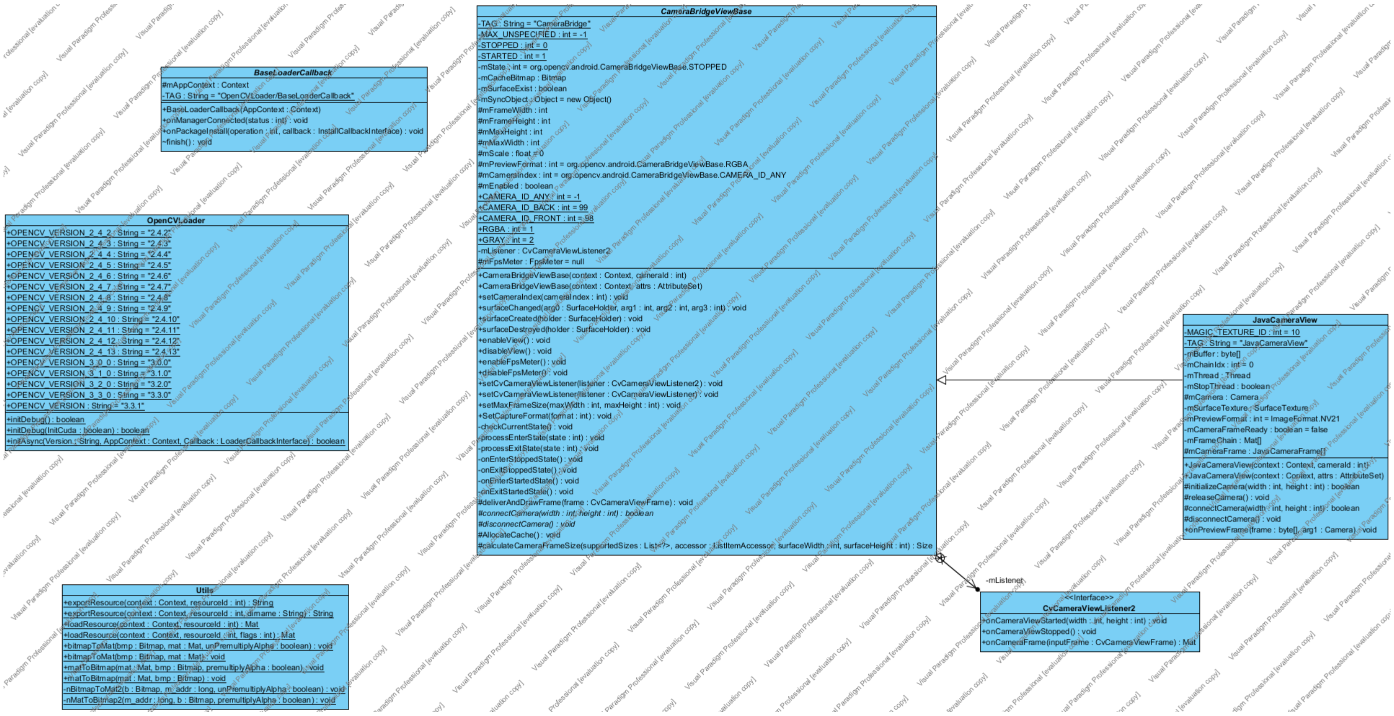
Reading row by row from left to right, the main 'flow' of the app should be apparent. The sequence of screens from 2 through 5 is iterated for each fruit until all 4 fruits have been identified - culminating finally in a 'success' screen (#6). The game does not permit alternative flows - the only option is to 'quit' from a menu item in portrait mode (i.e. from screens 1, 2, 5, and 6). The activity indicated in screens 3 and 4 is done in a windowless, title-less landscape mode.

A use case diagram illustrating known requirements for The IT Project has been considered for submission and rejected for the reason that it will not add to the sum of knowledge about the project. A diagram has been included below however to illustrate, by way of contrast, what requirements may be captured in a use case diagram, as shown in the following example depicting a more complex subject:

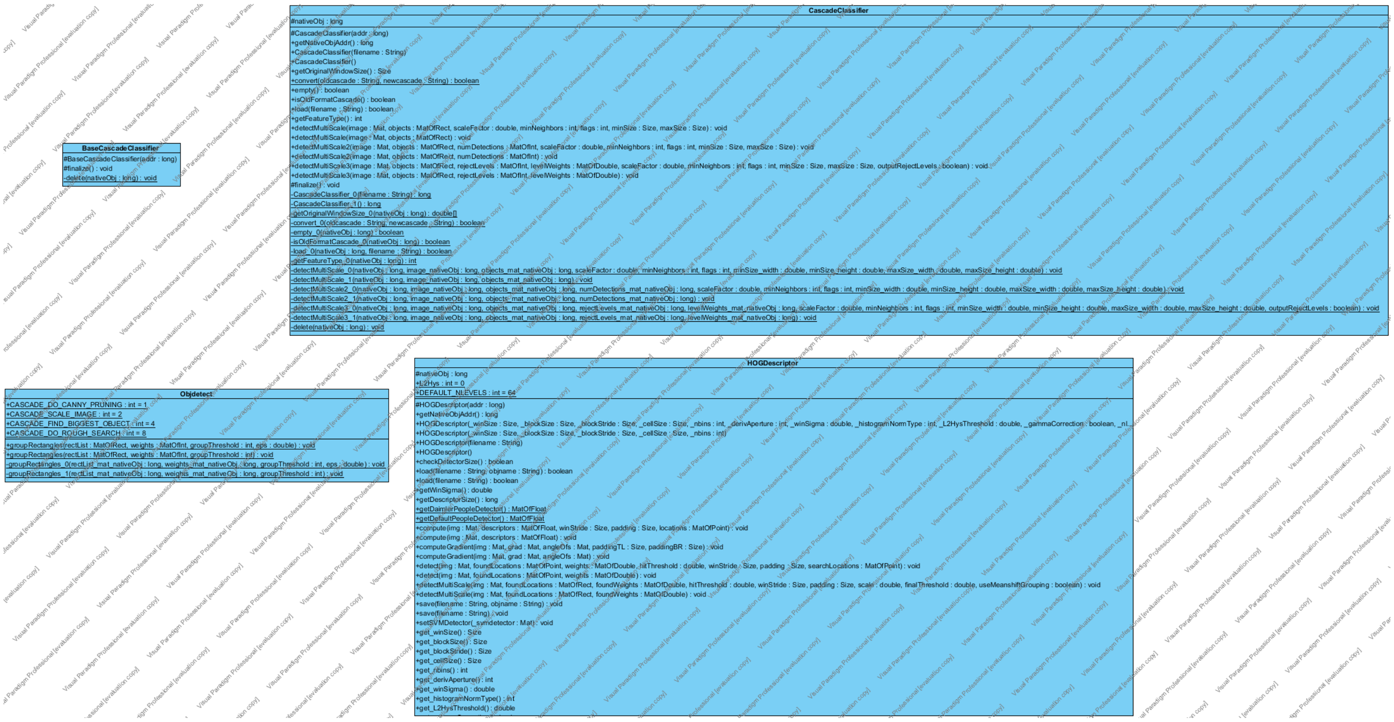

Android Game - Bug TD
Read Game Help Screen
Look for help on game
Single Player User
Play Single Player Game
Play game alone
Load game from internal memory
Load Game From Memory
Pass game to the player in game
Pause and save game to internal storage
Save Game To Memory
Save game to internal storage
Game Storage
View the high scores the have been achieved so far
View high scores from memory
Load the high scores from the internal storage
Multi-player User
Play game with friend
Play Multi-player Game
Play game with others

Analysis classes

The following classes in the android package of OpenCv will certianly be used (please use zoom feature in the pdf viewer):


BaseLoaderCallback
#mAppContext : Context
-TAG : String = "OpenCVLoader/BaseLoaderCallback"
+BaseLoaderCallback(AppContext : Context)
+onManagerConnected(status : int) : void
+onPackageInstall(operation : int, callback : InstallCallbackInterface) : void
~finish() : void
OpenCVLoader
+OPENCV_VERSION_2_4_2 : String = "2.4.2"
+OPENCV_VERSION_2_4_3 : String = "2.4.3"
+OPENCV_VERSION_2_4_4 : String = "2.4.4"
+OPENCV_VERSION_2_4_5 : String = "2.4.5"
+OPENCV_VERSION_2_4_6 : String = "2.4.6"
+OPENCV_VERSION_2_4_7 : String = "2.4.7"
+OPENCV_VERSION_2_4_8 : String = "2.4.8"
+OPENCV_VERSION_2_4_9 : String = "2.4.9"
+OPENCV_VERSION_2_4_10 : String = "2.4.10"
+OPENCV_VERSION_2_4_11 : String = "2.4.11"
+OPENCV_VERSION_2_4_12 : String = "2.4.12"
+OPENCV_VERSION_2_4_13 : String = "2.4.13"
+OPENCV_VERSION_3_0_0 : String = "3.0.0"
+OPENCV_VERSION_3_1_0 : String = "3.1.0"
+OPENCV_VERSION_3_2_0 : String = "3.2.0"
+OPENCV_VERSION_3_3_0 : String = "3.3.0"
+OPENCV_VERSION : String = "3.3.1"
+initDebug() : boolean
+initDebug(InitCuda : boolean) : boolean
+initAsync(Version : String, AppContext : Context, Callback : LoaderCallbackInterface) : boolean
Utils
+exportResource(context : Context, resourceId : int) : String
+exportResource(context : Context, resourceId : int, dirname : String) : String
+loadResource(context : Context, resourceId : int) : Mat
+loadResource(context : Context, resourceId : int, flags : int) : Mat
+bitmapToMat(bmp : Bitmap, mat : Mat, unPremultiplyAlpha : boolean) : void
+bitmapToMat(bmp : Bitmap, mat : Mat) : void
+matToBitmap(mat : Mat, bmp : Bitmap, premultiplyAlpha : boolean) : void
+matToBitmap(mat : Mat, bmp : Bitmap) : void
-nBitmapToMat2(b : Bitmap, m_addr : long, unPremultiplyAlpha : boolean) : void
-nMatToBitmap2(m_addr : long, b : Bitmap, premultiplyAlpha : boolean) : void
CameraBridgeViewBase
-TAG : String = "CameraBridge"
-MAX_UNSPECIFIED : int = -1
-STOPPED : int = 0
-STARTED : int = 1
-mState : int = org.opencv.android.CameraBridgeViewBase.STOPPED
-mCacheBitmap : Bitmap
-mSurfaceExist : boolean
-mSyncObject : Object = new Object()
#mFrameWidth : int
#mFrameHeight : int
#mMaxHeight : int
#mMaxWidth : int
#mScale : float = 0
#mPreviewFormat : int = org.opencv.android.CameraBridgeViewBase.RGBA
#mCameraIndex : int = org.opencv.android.CameraBridgeViewBase.CAMERA_ID_ANY
#mEnabled : boolean
+CAMERA_ID_ANY : int = -1
+CAMERA_ID_BACK : int = 99
+CAMERA_ID_FRONT : int = 98
+RGBA : int = 1
+GRAY : int = 2
-mListener : CvCameraViewListener2
#mFpsMeter : FpsMeter = null
+CameraBridgeViewBase(context : Context, cameraId : int)
+CameraBridgeViewBase(context : Context, attrs : AttributeSet)
+setCameraIndex(cameraIndex : int) : void
+surfaceChanged(arg0 : SurfaceHolder, arg1 : int, arg2 : int, arg3 : int) : void
+surfaceCreated(holder : SurfaceHolder) : void
+surfaceDestroyed(holder : SurfaceHolder) : void
+enableView() : void
+disableView() : void
+enableFpsMeter() : void
+disableFpsMeter() : void
+setCvCameraViewListener(listener : CvCameraViewListener2) : void
+setCvCameraViewListener(listener : CvCameraViewListener) : void
+setMaxFrameSize(maxWidth : int, maxHeight : int) : void
+SetCaptureFormat(format : int) : void
-checkCurrentState() : void
-processEnterState(state : int) : void
-processExitState(state : int) : void
-onEnterStoppedState() : void
-onExitStoppedState() : void
-onEnterStartedState() : void
-onExitStartedState() : void
#deliverAndDrawFrame(frame : CvCameraViewFrame) : void
#connectCamera(width : int, height : int) : boolean
#disconnectCamera() : void
#AllocateCache() : void
#calculateCameraFrameSize(supportedSizes : List<?>, accessor : ListItemAccessor, surfaceWidth : int, surfaceHeight : int) : Size
JavaCameraView
-MAGIC_TEXTURE_ID : int = 10
-TAG : String = "JavaCameraView"
-mBuffer : byte[]
-mChainIdx : int = 0
-mThread : Thread
-mStopThread : boolean
#mCamera : Camera
-mSurfaceTexture : SurfaceTexture
-mPreviewFormat : int = ImageFormat.NV21
-mCameraFrameReady : boolean = false
-mFrameChain : Mat[]
#mCameraFrame : JavaCameraFrame[]
+JavaCameraView(context : Context, cameraId : int)
+JavaCameraView(context : Context, attrs : AttributeSet)
#initializeCamera(width : int, height : int) : boolean
#releaseCamera() : void
#connectCamera(width : int, height : int) : boolean
#disconnectCamera() : void
+onPreviewFrame(frame : byte[], arg1 : Camera) : void
-mListener
<<Interface>>
CvCameraViewListener2
+onCameraViewStarted(width : int, height : int) : void
+onCameraViewStopped() : void
+onCameraFrame(inputFrame : CvCameraViewFrame) : Mat

The project requires a cascade classifier that instructs OpenCV how to recognize, for example, a banana. The classes that do this work appear in the *objdetec*t package (please use zoom feature in the pdf viewer):


CascadeClassifier
#nativeObj : long
#CascadeClassifier(addr : long)
+getNativeObjAddr() : long
+CascadeClassifier(filename : String)
+CascadeClassifier()
+getOriginalWindowSize() : Size
+convert(oldcascade : String, newcascade : String) : boolean
+empty() : boolean
+isOldFormatCascade() : boolean
+load(filename : String) : boolean
+getFeatureType() : int
+detectMultiScale(image : Mat, objects : MatOfRect, scaleFactor : double, minNeighbors : int, flags : int, minSize : Size, maxSize : Size) : void
+detectMultiScale(image : Mat, objects : MatOfRect) : void
+detectMultiScale2(image : Mat, objects : MatOfRect, numDetections : MatOfInt, scaleFactor : double, minNeighbors : int, flags : int, minSize : Size, maxSize : Size) : void
+detectMultiScale2(image : Mat, objects : MatOfRect, numDetections : MatOfInt) : void
+detectMultiScale3(image : Mat, objects : MatOfRect, rejectLevels : MatOfInt, levelWeights : MatOfDouble, scaleFactor : double, minNeighbors : int, flags : int, minSize : Size, maxSize : Size, outputRejectLevels : boolean) : void
+detectMultiScale3(image : Mat, objects : MatOfRect, rejectLevels : MatOfInt, levelWeights : MatOfDouble) : void
#finalize() : void
-CascadeClassifier_0(filename : String) : long
-CascadeClassifier_1() : long
-getOriginalWindowSize_0(nativeObj : long) : double[]
-convert_0(oldcascade : String, newcascade : String) : boolean
-empty_0(nativeObj : long) : boolean
-isOldFormatCascade_0(nativeObj : long) : boolean
-load_0(nativeObj : long, filename : String) : boolean
-getFeatureType_0(nativeObj : long) : int
-detectMultiScale_0(nativeObj : long, image_nativeObj : long, objects_mat_nativeObj : long, scaleFactor : double, minNeighbors : int, flags : int, minSize_width : double, minSize_height : double, maxSize_width : double, maxSize_height : double) : void
-detectMultiScale_1(nativeObj : long, image_nativeObj : long, objects_mat_nativeObj : long) : void
-detectMultiScale2_0(nativeObj : long, image_nativeObj : long, objects_mat_nativeObj : long, numDetections_mat_nativeObj : long, scaleFactor : double, minNeighbors : int, flags : int, minSize_width : double, minSize_height : double, maxSize_width : double, maxSize_height : double) : void
-detectMultiScale2_1(nativeObj : long, image_nativeObj : long, objects_mat_nativeObj : long, numDetections_mat_nativeObj : long) : void
-detectMultiScale3_0(nativeObj : long, image_nativeObj : long, objects_mat_nativeObj : long, rejectLevels_mat_nativeObj : long, levelWeights_mat_nativeObj : long, scaleFactor : double, minNeighbors : int, flags : int, minSize_width : double, minSize_height : double, maxSize_width : double, maxSize_height : double, outputRejectLevels : boolean) : void
-detectMultiScale3_1(nativeObj : long, image_nativeObj : long, objects_mat_nativeObj : long, rejectLevels_mat_nativeObj : long, levelWeights_mat_nativeObj : long) : void
-delete(nativeObj : long) : void
BaseCascadeClassifier
#BaseCascadeClassifier(addr : long)
#finalize() : void
-delete(nativeObj : long) : void
Objdetect
+CASCADE_DO_CANNY_PRUNING : int = 1
+CASCADE_SCALE_IMAGE : int = 2
+CASCADE_FIND_BIGGEST_OBJECT : int = 4
+CASCADE_DO_ROUGH_SEARCH : int = 8
+groupRectangles(rectList : MatOfRect, weights : MatOfInt, groupThreshold : int, eps : double) : void
+groupRectangles(rectList : MatOfRect, weights : MatOfInt, groupThreshold : int) : void
-groupRectangles_0(rectList_mat_nativeObj : long, weights_mat_nativeObj : long, groupThreshold : int, eps : double) : void
-groupRectangles_1(rectList_mat_nativeObj : long, weights_mat_nativeObj : long, groupThreshold : int) : void
HOGDescriptor
#nativeObj : long
+L2Hys : int = 0
+DEFAULT_NLEVELS : int = 64
#HOGDescriptor(addr : long)
+getNativeObjAddr() : long
+HOGDescriptor(_winSize : Size, _blockSize : Size, _blockStride : Size, _cellSize : Size, _nbins : int, _derivAperture : int, _winSigma : double, _histogramNormType : int, _L2HysThreshold : double, _gammaCorrection : boolean, _nl...
+HOGDescriptor(_winSize : Size, _blockSize : Size, _blockStride : Size, _cellSize : Size, _nbins : int)
+HOGDescriptor(filename : String)
+HOGDescriptor()
+checkDetectorSize() : boolean
+load(filename : String, objname : String) : boolean
+load(filename : String) : boolean
+getWinSigma() : double
+getDescriptorSize() : long
+getDaimlerPeopleDetector() : MatOfFloat
+getDefaultPeopleDetector() : MatOfFloat
+compute(img : Mat, descriptors : MatOfFloat, winStride : Size, padding : Size, locations : MatOfPoint) : void
+compute(img : Mat, descriptors : MatOfFloat) : void
+computeGradient(img : Mat, grad : Mat, angleOfs : Mat, paddingTL : Size, paddingBR : Size) : void
+computeGradient(img : Mat, grad : Mat, angleOfs : Mat) : void
+detect(img : Mat, foundLocations : MatOfPoint, weights : MatOfDouble, hitThreshold : double, winStride : Size, padding : Size, searchLocations : MatOfPoint) : void
+detect(img : Mat, foundLocations : MatOfPoint, weights : MatOfDouble) : void
+detectMultiScale(img : Mat, foundLocations : MatOfRect, foundWeights : MatOfDouble, hitThreshold : double, winStride : Size, padding : Size, scale : double, finalThreshold : double, useMeanshiftGrouping : boolean) : void
+detectMultiScale(img : Mat, foundLocations : MatOfRect, foundWeights : MatOfDouble) : void
+save(filename : String, objname : String) : void
+save(filename : String) : void
+setSVMDetector(_svmdetector : Mat) : void
+get_winSize() : Size
+get_blockSize() : Size
+get_blockStride() : Size
+get_cellSize() : Size
+get_nbins() : int
+get_derivAperture() : int
+get_winSigma() : double
+get_histogramNormType() : int
+get_L2HysThreshold() : double

10. Contact Us

You can get in touch with the developer in any of the below ways:

By Phone:
0447 303 393 (Cellphone of Duane McMahon)

By Email:
dmcmah09@postoffice.csu.edu.au

If you would like to proceed with our proposal then you can sign the first page and return a copy to us by fax, email or post. In any case please feel free to call us to discuss the quote, request more information or for any other reason.

We look forward to hearing from you soon!