***MESSAGE SENDING:***

**import** java.io.IOException;

**public** **class** MessageSender {

Connection conn = ConnectionManager.*getConnection*();

**public** **void** sendPostAlert(String filename) {

ArrayList<String> list\_gcm\_id = **new** ArrayList<String>();

**try** {

Statement st = conn.createStatement();

ResultSet rs = st.executeQuery("SELECT \* FROM usertable");

**while** (rs.next()) {

list\_gcm\_id.add(rs.getString("gcm\_id"));

}

} **catch** (SQLException e) {

e.printStackTrace();

}

System.*out*.println(list\_gcm\_id+"\n");

**if** (list\_gcm\_id.size() == 1) {

Sender sender = **new** Sender(

"AIzaSyB36xhJBNHfDKwoq8\_qAVkJ8Wg9uLoBE2c");

Message message = **new** Message.Builder().collapseKey("1")

.timeToLive(3).delayWhileIdle(**true**)

.addData("filename", filename).build();

**try** {

Result result = sender.send(message, list\_gcm\_id.get(0), 3);

System.*out*.println(result.toString());

} **catch** (IOException e) {

e.printStackTrace();

}

} **else** **if** (list\_gcm\_id.size() != 0 && list\_gcm\_id.size() > 1) {

Sender sender = **new** Sender(

"AIzaSyB36xhJBNHfDKwoq8\_qAVkJ8Wg9uLoBE2c");

Message message = **new** Message.Builder().collapseKey("1")

.timeToLive(3).delayWhileIdle(**true**)

.addData("filename", filename).build();

**try** {

MulticastResult multicastResult = sender.sendNoRetry(message,

list\_gcm\_id);

System.*out*.println(multicastResult.toString());

} **catch** (IOException e) {

e.printStackTrace();

}}}

**MOTION DETECTION CONTROL:**

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** javax.swing.\*;

**import** javax.swing.event.\*;

**import** javax.media.Control;

**public** **class** MotionDetectionControl **implements** Control, ActionListener, ChangeListener {

Component component;

JButton button;

JSlider threshold;

MotionDetectionEffect motion;

**public** MotionDetectionControl(MotionDetectionEffect motion) {

**this**.motion = motion; }

**public** Component getControlComponent () {

**if** (component == **null**) {

button = **new** JButton("Debug");

button.addActionListener(**this**);

button.setToolTipText("Click to turn debugging mode on/off");

threshold = **new** JSlider(JSlider.*HORIZONTAL*,0, motion.THRESHOLD\_MAX,

motion.THRESHOLD\_INIT);

threshold.setMajorTickSpacing(motion.THRESHOLD\_INC);

threshold.setPaintLabels(**true**);

threshold.addChangeListener(**this**);

Panel componentPanel = **new** Panel();

componentPanel.setLayout(**new** BorderLayout());

componentPanel.add("East", button);

componentPanel.add("West", threshold);

componentPanel.invalidate();

component = componentPanel;

} **return** component;}

**public** **void** actionPerformed(ActionEvent e) {

Object o = e.getSource();

**if** (o == button) {

**if** (motion.debug == **false**)

motion.debug = **true**;

**else** motion.debug = **false**;

}}

**public** **void** stateChanged(ChangeEvent e) {

Object o = e.getSource();

**if** (o == threshold) {

motion.blob\_threshold = threshold.getValue()\*1000;

}

}}

**MOTION DETECTION EFFECT:**

**import** java.awt.Dimension;

**public** **class** MotionDetectionEffect **implements** Effect {

**public** **static** Player *player* = **null**;

**public** Buffer buf;

**public** BufferToImage btoi;

**public** Image img;

**public** **int** OPTIMIZATION = 0;

**boolean** tag=**true**;

**public** **int** THRESHOLD\_MAX = 10000;

**public** **int** THRESHOLD\_INC = 1000;

**public** **boolean** debug = **false**;

**public** **int** THRESHOLD\_INIT = 5000;

**public** **int** blob\_threshold = THRESHOLD\_INIT;

**private** **final** **static** **int** *INITIAL\_SQUARE\_SIZE* = 5;

**public** **final** **static** Format[] *supportedFormat* = **new** Format[] { //

**new** RGBFormat(**null**, //

Format.*NOT\_SPECIFIED*, //

Format.*byteArray*, //

Format.*NOT\_SPECIFIED*, //

24, //

3, 2, 1, //

3, Format.*NOT\_SPECIFIED*, //

Format.*TRUE*, //

Format.*NOT\_SPECIFIED*) };

**private** Format inputFormat;

**private** Format outputFormat;

**private** Format[] inputFormats;

**private** Format[] outputFormats;

**private** **int**[] bwPixels;

**private** **byte**[] bwData;

**private** **boolean** visualize = **true**;

**private** **boolean** serverActive = **true**;

**private** **boolean** updateRequested;

**private** **int** avg\_ref\_intensity;

**private** **int** avg\_img\_intensity;

**private** RGBFormat vfIn = **null**;

**private** **int**[] threshs = { 50, 100, 150, 200 };

**private** **int** det\_thresh = threshs[1];

**private** **int**[] colors = { 0x00FF0000, 0x00FF9900, 0x00FFFF00, 0x00FFFFFF };

**private** **int**[] newImageSquares = **null**;

**private** **int**[] oldImageSquares = **null**;

**private** **int**[] changedSquares = **null**;

**private** **int** numberOfSquaresWide;

**private** **int** numberOfSquaresHigh;

**private** **int** numberOfSquares;

**private** **int** sqSide = *INITIAL\_SQUARE\_SIZE*;

**private** **int** sqArea = 0;

**private** **int** sqWidthLeftover = 0;

**private** **int** sqHeightLeftover = 0;

**private** **int** pixelSpace = 0;

**private** **int** imageWidth = 0;

**private** **int** imageHeight = 0;

**private** **int** imageArea = 0;

**public** MotionDetectionEffect() {

inputFormats = **new** Format[] { **new** RGBFormat(**null**, Format.*NOT\_SPECIFIED*,

Format.*byteArray*, Format.*NOT\_SPECIFIED*, 24, 3, 2, 1, 3,

Format.*NOT\_SPECIFIED*, Format.*TRUE*, Format.*NOT\_SPECIFIED*) };

outputFormats = **new** Format[] { **new** RGBFormat(**null**,

Format.*NOT\_SPECIFIED*, Format.*byteArray*, Format.*NOT\_SPECIFIED*,

24, 3, 2, 1, 3, Format.*NOT\_SPECIFIED*, Format.*TRUE*,

Format.*NOT\_SPECIFIED*) };

}

**public** Format[] getSupportedInputFormats() {

**return** inputFormats;

}

**public** Format[] getSupportedOutputFormats(Format input) {

**if** (input == **null**) {

**return** outputFormats;

}

**if** (matches(input, inputFormats) != **null**) {

**return** **new** Format[] { outputFormats[0].intersects(input) };

} **else** {

**return** **new** Format[0];

}

}

**public** Format setInputFormat(Format input) {

inputFormat = input;

**return** input;

}

**public** Format setOutputFormat(Format output) {

**if** (output == **null** || matches(output, outputFormats) == **null**)

**return** **null**;

RGBFormat incoming = (RGBFormat) output;

Dimension size = incoming.getSize();

**int** maxDataLength = incoming.getMaxDataLength();

**int** lineStride = incoming.getLineStride();

**float** frameRate = incoming.getFrameRate();

**int** flipped = incoming.getFlipped();

**int** endian = incoming.getEndian();

**if** (size == **null**)

**return** **null**;

**if** (maxDataLength < size.width \* size.height \* 3)

maxDataLength = size.width \* size.height \* 3;

**if** (lineStride < size.width \* 3)

lineStride = size.width \* 3;

**if** (flipped != Format.*FALSE*)

flipped = Format.*FALSE*;

outputFormat = outputFormats[0].intersects(**new** RGBFormat(size,

maxDataLength, **null**, frameRate, Format.*NOT\_SPECIFIED*,

Format.*NOT\_SPECIFIED*, Format.*NOT\_SPECIFIED*,

Format.*NOT\_SPECIFIED*, Format.*NOT\_SPECIFIED*, lineStride,

Format.*NOT\_SPECIFIED*, Format.*NOT\_SPECIFIED*));

**return** outputFormat;}

**public** **synchronized** **int** process(Buffer inBuffer, Buffer outBuffer) {

**int** outputDataLength = ((VideoFormat) outputFormat).getMaxDataLength();

validateByteArraySize(outBuffer, outputDataLength);

outBuffer.setLength(outputDataLength);

outBuffer.setFormat(outputFormat);

outBuffer.setFlags(inBuffer.getFlags());

**byte**[] inData = (**byte**[]) inBuffer.getData();

**byte**[] outData = (**byte**[]) outBuffer.getData();

**int**[] sqAvg = **null**;

**int**[] refsqAvg = **null**;

vfIn = (RGBFormat) inBuffer.getFormat();

Dimension sizeIn = vfIn.getSize();

**int** pixStrideIn = vfIn.getPixelStride();

**int** lineStrideIn = vfIn.getLineStride();

imageWidth = (vfIn.getLineStride()) / 3;

imageHeight = ((vfIn.getMaxDataLength()) / 3) / imageWidth;

imageArea = imageWidth \* imageHeight;

**int** r, g, b = 0;

**if** (oldImageSquares == **null**) {

changeSqSize(*INITIAL\_SQUARE\_SIZE*);

updateRequested = **true**;

}System.*arraycopy*(inData, 0, outData, 0, outData.length);

bwPixels = **new** **int**[outputDataLength / 3];

**for** (**int** ip = 0; ip < outputDataLength; ip += 3) {

**int** bw = 0;

r = (**int**) inData[ip] & 0xFF;

g = (**int**) inData[ip + 1] & 0xFF;

b = (**int**) inData[ip + 2] & 0xFF;

bw = (**int**) ((r + b + g) / (**double**) 3);

bwPixels[ip / 3] = bw;

}

**if** (updateRequested) {

updateRequested = **false**;

updateSquares();

**return** *BUFFER\_PROCESSED\_OK*;

} **else** {updateSquares();

oldNewChange();

**int** c = 0;

**for** (**int** i = 0; i < changedSquares.length; i++) {

**if** (changedSquares[i] > det\_thresh) {

c++;

}}

**if** (c > 40 && serverActive && tag) {

tag = **false**;

Timer t = **new** Timer();

t.schedule(**new** TimerTask(){

**public** **void** run(){

tag =**true**;}

},1\*1000);

Date date = **new** Date();

DateFormat formatter = **new** SimpleDateFormat("MMMddHHmmss");

String s = formatter.format(date.getTime()) + ".jpg";

String filename=s;

s = "C:\\Program Files (x86)\\Apache Software Foundation\\Tomcat 7.0\\webapps\\LoginRegistration\\images\\" + s;

System.*out*.println(s);

**try** {

buf = outBuffer;

btoi = **new** BufferToImage((VideoFormat) buf.getFormat());

img = btoi.createImage(buf);

MessageSender send=**new** MessageSender();

send.sendPostAlert(filename);

*saveJPG*(img, s);

}

**catch** (Exception ex) {

ex.printStackTrace();

}

System.*out*.println("Motion detected (motion at " + c + "areas");}

**if** (visualize) {

**for** (**int** i = 1; i <= numberOfSquares; i++) {

**if** ((changedSquares[i - 1] > threshs[0])) {

**if** (((i % numberOfSquaresWide) != 0)&& (numberOfSquares - i) > numberOfSquaresWide)

**int** begin = ((((i % numberOfSquaresWide) - 1) \* sqSide) + ((i / numberOfSquaresWide)

\* imageWidth \* sqSide)) \* 3

**if** (changedSquares[i - 1] > threshs[3]) {

b = (**byte**) (colors[3] & 0xFF);

g = (**byte**) ((colors[3] >> 8) & 0xFF);

r = (**byte**) ((colors[3] >> 16) & 0xFF);

} **else** **if** (changedSquares[i - 1] > threshs[2]) {

b = (**byte**) (colors[2] & 0xFF);

g = (**byte**) ((colors[2] >> 8) & 0xFF);

r = (**byte**) ((colors[2] >> 16) & 0xFF);

} **else** **if** (changedSquares[i - 1] > threshs[1])

b = (**byte**) (colors[1] & 0xFF);

g = (**byte**) ((colors[1] >> 8) & 0xFF);

r = (**byte**) ((colors[1] >> 16) & 0xFF);

} **else** {

b = (**byte**) (colors[0] & 0xFF);

g = (**byte**) ((colors[0] >> 8) & 0xFF);

r = (**byte**) ((colors[0] >> 16) & 0xFF);

}

**for** (**int** k = begin; k < (begin + (sqSide\* imageWidth \* 3));

k = k+ (imageWidth \* 3)) {

**for** (**int** j = k; j < (k + (sqSide \* 3)); j = j + 3) {

**try** {

outData[j] = (**byte**) b;

outData[j + 1] = (**byte**) g;

outData[j + 2] = (**byte**) r;

} **catch** (ArrayIndexOutOfBoundsException e) {

System.*out*.println("Nullpointer: j = "+ j + ". Outdata.length = "+ outData.length);

System.*exit*(1);

}

}

}}}}}}

**return** *BUFFER\_PROCESSED\_OK*;

}

**public** **static** **void** saveJPG(Image img, String sav) {

BufferedImage bi = **new** BufferedImage(img.getWidth(**null**),

img.getHeight(**null**), 1);

Graphics2D g2 = bi.createGraphics();

g2.drawImage(img, **null**, **null**);

FileOutputStream out = **null**;

**try** {

out = **new** FileOutputStream(sav);

} **catch** (FileNotFoundException io) {

System.*out*.println("File Not Found");

}

JPEGImageEncoder encoder = JPEGCodec.*createJPEGEncoder*(out);

JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bi);

param.setQuality(0.5F, **false**);

encoder.setJPEGEncodeParam(param);

**try** {

encoder.encode(bi);

out.close();

} **catch** (IOException io) {

System.*out*.println("IOException");

}

}

**public** String getName() {

**return** "Motion Detection Codec";

}

**public** **void** open() {

}**public** **void** close() {

} **public** **void** reset() {

}

**public** Object getControl(String controlType) {

System.*out*.println(controlType);

**return** **null**; }

**public** Object[] getControls() {

**return** **null**;

}

**public** Format matches(Format in, Format outs[]) {

**for** (**int** i = 0; i < outs.length; i++) {

**if** (in.matches(outs[i]))

**return** outs[i];

}

**return** **null**;

}

**byte**[] validateByteArraySize(Buffer buffer, **int** newSize) {

Object objectArray = buffer.getData();

**byte**[] typedArray;

**if** (objectArray **instanceof** **byte**[]) {

typedArray = (**byte**[]) objectArray;

**if** (typedArray.length >= newSize) {

**return** typedArray;

}

**byte**[] tempArray = **new** **byte**[newSize];

System.*arraycopy*(typedArray, 0, tempArray, 0, typedArray.length);

typedArray = tempArray;

} **else** {

typedArray = **new** **byte**[newSize];

}

buffer.setData(typedArray);

**return** typedArray;

}

**private** **void** setPixelSpace(**int** newSpace) {

pixelSpace = newSpace;

}

**private** **void** changeSqSize(**int** newSide) {

sqSide = newSide;

sqArea = newSide \* newSide;

**int** wid = (imageWidth / sqSide);

**int** hei = (imageHeight / sqSide);

sqWidthLeftover = imageWidth % sqSide;

sqHeightLeftover = imageHeight % sqSide;

**if** (sqWidthLeftover > 0) {

wid++;

}

**if** (sqHeightLeftover > 0) {

hei++;

}

numberOfSquaresWide = wid;

numberOfSquaresHigh = hei;

numberOfSquares = wid \* hei;

newImageSquares = **new** **int**[numberOfSquares];

oldImageSquares = **new** **int**[numberOfSquares];

changedSquares = **new** **int**[numberOfSquares]; }

**private** **int** averageInSquare(**int** startX, **int** startY, **int** sqWidth,

**int** sqHeight) {

**int** average = 0;

**for** (**int** i = 0; i < sqHeight; i = i + 1 + pixelSpace) {

**for** (**int** j = 0; j < sqWidth; j = j + 1 + pixelSpace) {

average += bwPixels[(((startY + i) \* imageWidth) + (startX + j))];

}

}

average = average / (sqWidth \* sqHeight);

**return** average;

}

**private** **void** updateSquares() {

System.*arraycopy*(newImageSquares, 0, oldImageSquares, 0,

newImageSquares.length);

**int** sqCount = 0;

**for** (**int** j = 0; j < (imageHeight); j = j + sqSide) {

**for** (**int** i = 0; i < (imageWidth); i = i + sqSide) {

**if** (i <= (imageWidth - sqSide) && j <= (imageHeight - sqSide)) {

newImageSquares[sqCount] = averageInSquare(i, j, sqSide,

sqSide);

} **else** **if** (i > (imageWidth - sqSide)

&& j <= (imageHeight - sqSide)) {

newImageSquares[sqCount] = averageInSquare(i, j,

sqWidthLeftover, sqSide);

} **else** **if** (i <= (imageWidth - sqSide)

&& j > (imageHeight - sqSide)) {

newImageSquares[sqCount] = averageInSquare(i, j, sqSide,

sqHeightLeftover);

} **else** **if** (i > (imageWidth - sqSide)

&& j > (imageHeight - sqSide)) {

newImageSquares[sqCount] = averageInSquare(i, j,

sqWidthLeftover, sqHeightLeftover);

}

sqCount++;

}

}

}

**private** **void** oldNewChange() {

**for** (**int** i = 0; i <= (numberOfSquares - 1); i++) {

**int** difference = Math.*abs*((newImageSquares[i])

- (oldImageSquares[i]));

changedSquares[i] = difference;

}}

**public** **synchronized** **void** updateModel(**boolean** visualize,

**boolean** serverActive, **boolean** simplified, **int**[] threshs,

**int**[] colors, **int** sqSide, **int** det\_thresh) {

**this**.visualize = visualize;

**this**.serverActive = serverActive;

**if** (sqSide != **this**.sqSide)

changeSqSize(sqSide);

**if** (!simplified) {

System.*out*.println((colors == **null**) + " " + (**this**.colors == **null**));

System.*arraycopy*(colors, 0, **this**.colors, 0, colors.length);

System.*arraycopy*(threshs, 0, **this**.threshs, 0, colors.length);

**this**.det\_thresh = det\_thresh;

System.*out*.println("New det\_threhsh: " + **this**.det\_thresh);

}

updateRequested = **true**;

}

**public** **boolean** isVisual() {

**return** visualize;

}

**public** **int**[] getThreshholds() {

**return** threshs;

}

**public** **boolean** isServerActive() {

**return** serverActive;

}

**public** **int**[] getColors() {

**return** colors;

}

**public** **int** getSqSide() {

**return** sqSide;

}

}

**TEST MOTION DETECTION:**

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** javax.media.\*;

**import** javax.media.control.TrackControl;

**import** javax.media.Format;

**import** javax.media.format.\*;

**import** javax.media.protocol.\*;

**import** javax.media.datasink.\*;

**import** javax.media.control.\*;

**public** **class** TestMotionDetection **extends** Frame **implements** ControllerListener {

Processor p;

DataSink fileW = **null**;

Object waitSync = **new** Object();

**boolean** stateTransitionOK = **true**;

**public** TestMotionDetection() {

**super**("Test Motion Detection");

}

**public** **boolean** open(MediaLocator ds) {

**try** {

p = Manager.*createProcessor*(ds);

} **catch** (Exception e) {

System.*err*.println("Failed to create a processor from the given datasource: " + e);

**return** **false**;

}

p.addControllerListener(**this**);

p.configure();

**if** (!waitForState(p.*Configured*)) {

System.*err*.println("Failed to configure the processor.");

**return** **false**;

}

p.setContentDescriptor(**null**);

TrackControl tc[] = p.getTrackControls();

**if** (tc == **null**) {

System.*err*.println("Failed to obtain track controls from the processor.");

**return** **false**;

}

TrackControl videoTrack = **null**;

**for** (**int** i = 0; i < tc.length; i++) {

**if** (tc[i].getFormat() **instanceof** VideoFormat) {

videoTrack = tc[i];

**break**;

}

}

**if** (videoTrack == **null**) {

System.*err*.println("The input media does not contain a video track.");

**return** **false**;

}**try** {

Codec codec[] = { **new** MotionDetectionEffect(), **new** TimeStampEffect()};

videoTrack.setCodecChain(codec);

} **catch** (UnsupportedPlugInException e) {

System.*err*.println("The processor does not support effects.");

}

p.prefetch();

**if** (!waitForState(p.*Prefetched*)) {

System.*err*.println("Failed to realize the processor.");

**return** **false**;

}

setLayout(**new** BorderLayout());

Component cc;

Component vc;

**if** ((vc = p.getVisualComponent()) != **null**) {

add("Center", vc);

}

**if** ((cc = p.getControlPanelComponent()) != **null**) {

add("South", cc);

}

p.start();

setVisible(**true**);

addWindowListener(**new** WindowAdapter() {

**public** **void** windowClosing(WindowEvent we) {

p.close();

System.*exit*(0);

}

});

p.start();

**return** **true**;

}

**public** **void** addNotify() {

**super**.addNotify();

pack();

}

**boolean** waitForState(**int** state) {

**synchronized** (waitSync) {

**try** {

**while** (p.getState() != state && stateTransitionOK)

waitSync.wait();

}

**catch** (Exception e) {}

}

**return** stateTransitionOK;

}

**public** **void** controllerUpdate(ControllerEvent evt) {

System.*out*.println(**this**.getClass().getName()+evt);

**if** (evt **instanceof** ConfigureCompleteEvent ||

evt **instanceof** RealizeCompleteEvent ||

evt **instanceof** PrefetchCompleteEvent) {

**synchronized** (waitSync) {

stateTransitionOK = **true**;

waitSync.notifyAll();

}

} **else** **if** (evt **instanceof** ResourceUnavailableEvent) {

**synchronized** (waitSync) {

stateTransitionOK = **false**;

waitSync.notifyAll();

}

} **else** **if** (evt **instanceof** EndOfMediaEvent) {

p.close();

System.*exit*(0);

}

}

**public** **static** **void** main(String [] args) {

String url = "vfw://0";

**if** (url.indexOf(":") < 0) {

*prUsage*();

System.*exit*(0);

}

MediaLocator ml;

**if** ((ml = **new** MediaLocator(url)) == **null**) {

System.*err*.println("Cannot build media locator from: " + url);

System.*exit*(0);

}

TestMotionDetection fa = **new** TestMotionDetection();

**if** (!fa.open(ml))

System.*exit*(0);

}

**static** **void** prUsage() {

System.*err*.println("Usage: java TestMotionDetection <url>");

}

}

**TIME STAMP EFFECT:**

**import** javax.media.\*;

**import** javax.media.format.\*;

**import** java.awt.\*;

**public** **class** TimeStampEffect **implements** Effect {

Format inputFormat;

Format outputFormat;

Format[] inputFormats;

Format[] outputFormats;

java.text.SimpleDateFormat sdf;

**public** TimeStampEffect() {

sdf = **new** java.text.SimpleDateFormat("hh:mm:ss MM/dd/yy");

inputFormats = **new** Format[] {

**new** RGBFormat(**null**,

Format.*NOT\_SPECIFIED*,

Format.*byteArray*,

Format.*NOT\_SPECIFIED*,

24,

3, 2, 1,

3, Format.*NOT\_SPECIFIED*,

Format.*TRUE*,

Format.*NOT\_SPECIFIED*)

};

outputFormats = **new** Format[] {

**new** RGBFormat(**null**,

Format.*NOT\_SPECIFIED*,

Format.*byteArray*,

Format.*NOT\_SPECIFIED*,

24,

3, 2, 1,

3, Format.*NOT\_SPECIFIED*,

Format.*TRUE*,

Format.*NOT\_SPECIFIED*)

};}

**public** Format[] getSupportedInputFormats() {

**return** inputFormats;

}

**public** Format [] getSupportedOutputFormats(Format input) {

**if** (input == **null**) {

**return** outputFormats;}

**if** (matches(input, inputFormats) != **null**) {

**return** **new** Format[] { outputFormats[0].intersects(input) };

} **else** {

**return** **new** Format[0];

}}

**public** Format setInputFormat(Format input) {

inputFormat = input;

**return** input;

}

**public** Format setOutputFormat(Format output) {

**if** (output == **null** || matches(output, outputFormats) == **null**)

**return** **null**;

RGBFormat incoming = (RGBFormat) output;

Dimension size = incoming.getSize();

**int** maxDataLength = incoming.getMaxDataLength();

**int** lineStride = incoming.getLineStride();

**float** frameRate = incoming.getFrameRate();

**int** flipped = incoming.getFlipped();

**int** endian = incoming.getEndian();

**if** (size == **null**)

**return** **null**;

**if** (maxDataLength < size.width \* size.height \* 3)

maxDataLength = size.width \* size.height \* 3;

**if** (lineStride < size.width \* 3)

lineStride = size.width \* 3;

**if** (flipped != Format.*FALSE*)

flipped = Format.*FALSE*;

outputFormat = outputFormats[0].intersects(**new** RGBFormat(size,

maxDataLength,

**null**,

frameRate,

Format.*NOT\_SPECIFIED*,

Format.*NOT\_SPECIFIED*,

Format.*NOT\_SPECIFIED*,

Format.*NOT\_SPECIFIED*,

Format.*NOT\_SPECIFIED*,

lineStride,

Format.*NOT\_SPECIFIED*,

Format.*NOT\_SPECIFIED*));

**return** outputFormat;

}

**public** **int** process(Buffer inBuffer, Buffer outBuffer) {

**int** outputDataLength = ((VideoFormat)outputFormat).getMaxDataLength();

validateByteArraySize(outBuffer, outputDataLength);

outBuffer.setLength(outputDataLength);

outBuffer.setFormat(outputFormat);

outBuffer.setFlags(inBuffer.getFlags());

**byte** [] inData = (**byte**[]) inBuffer.getData();

**byte** [] outData = (**byte**[]) outBuffer.getData();

RGBFormat vfIn = (RGBFormat) inBuffer.getFormat();

Dimension sizeIn = vfIn.getSize();

**int** pixStrideIn = vfIn.getPixelStride();

**int** lineStrideIn = vfIn.getLineStride();

**if** ( outData.length < sizeIn.width\*sizeIn.height\*3 ) {

System.*out*.println("the buffer is not full");

**return** *BUFFER\_PROCESSED\_FAILED*;

}

System.*arraycopy*(inData,0,outData,0,inData.length);

Font.*println*(sdf.format(**new** java.util.Date()) + " (math room 205)", Font.*FONT\_6x11*, 10, 20, (**byte**)255,(**byte**)255,(**byte**)255, outBuffer);

**return** *BUFFER\_PROCESSED\_OK*;

}

**public** String getName() {

**return** "TimeStamp Effect";

}

**public** **void** open() {

}

**public** **void** close() {

}

**public** **void** reset() {

}

**public** Object getControl(String controlType) {

**return** **null**;

}

**public** Object[] getControls() {

**return** **null**;

}

Format matches(Format in, Format outs[]) {

**for** (**int** i = 0; i < outs.length; i++) {

**if** (in.matches(outs[i]))

**return** outs[i]; } **return** **null**; }

**byte**[] validateByteArraySize(Buffer buffer,**int** newSize) {

Object objectArray=buffer.getData();

**byte**[] typedArray;

**if** (objectArray **instanceof** **byte**[]) {

typedArray=(**byte**[])objectArray;

**if** (typedArray.length >= newSize ) {

**return** typedArray;

}

**byte**[] tempArray=**new** **byte**[newSize];

System.*arraycopy*(typedArray,0,tempArray,0,typedArray.length);

typedArray = tempArray;

} **else** {

typedArray = **new** **byte**[newSize];

}

buffer.setData(typedArray);

**return** typedArray;

}}

**ANDROID MOBILE SIDE**

**DISPLAY IMAGE:**

**import** java.io.IOException;

**public** **class** DisplayImage **extends** Activity {

Button image\_btn;

EditText inputUrl;

ImageView image\_view;

Bitmap b = **null**;

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*imageex*);

inputUrl = ((EditText)findViewById(R.id.*imageUrl*));

inputUrl.setText(DisplayImage.**this**.getResources().getString(R.string.*ipaddress*)+getSharedPreferences("filename", Context.*MODE\_PRIVATE*).getString("filename", ""));

inputUrl.setSingleLine();

inputUrl.setTextSize(18);

image\_view = (ImageView)findViewById(R.id.*imageView1*);

image\_btn = (Button) findViewById(R.id.*getImageButton*);

image\_btn.setOnClickListener(**new** OnClickListener() {

**public** **void** onClick(View v) {

**if** (TextUtils.*isEmpty*(inputUrl.getText().toString())) {

}

**else** {

**new** Thread(**new** Runnable() {

**public** **void** run() {

**try** {

b = BitmapFactory.*decodeStream*((InputStream)**new**

URL(inputUrl.getText().toString()).getContent());

} **catch** (MalformedURLException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

}handler.sendEmptyMessage(1);

}}).start();

}}

});}

Handler handler = **new** Handler() {

@Override

**public** **void** handleMessage(Message msg) {

image\_view.setImageBitmap(b);

}};

**private** Drawable ImageOperations(Context ctx, String url) {

**try** {

InputStream is = (InputStream) **this**.fetch(url);

Drawable d = Drawable.*createFromStream*(is, "src");

**return** d;

} **catch** (MalformedURLException e) {

e.printStackTrace();

**return** **null**;

} **catch** (IOException e) {

e.printStackTrace();

**return** **null**;

}

}

**public** Object fetch(String address) **throws** MalformedURLException,IOException {

URL url = **new** URL(address);

Object content = url.getContent();

**return** content;

}

}

**GCM INTENT SERVER:**

**import** org.apache.http.HttpResponse;

**import** org.apache.http.client.HttpClient;

**import** org.apache.http.client.methods.HttpPost;

**import** android.annotation.SuppressLint;

**import** android.app.Notification;

**import** android.app.NotificationManager;

**import** android.app.PendingIntent;

**import** android.content.Context;

**import** android.content.Intent;

**import** android.support.v4.app.NotificationCompat;

**import** android.util.Log;

**import** com.google.android.gcm.GCMBaseIntentService;

**public** **class** GCMIntentService **extends** GCMBaseIntentService {

HttpClient httpClient;

HttpPost httpPost;

HttpResponse httpResponse;

**private** Thread thread;

**public** GCMIntentService() {

**super**(R.string.*registrationno*+"");

} **protected** **void** onError(Context arg0, String arg1) {

Log.*d*("Registration error", "Registration error" + arg1);

}

("NewApi")

@Override

**protected** **void** onMessage(Context arg0, Intent arg1) {

Log.*d*("Gcm message", "" + arg1.getStringExtra("filename"));

arg0.getSharedPreferences("filename", Context.*MODE\_PRIVATE*).edit()

.putString("filename", arg1.getStringExtra("filename"))

.commit();

NotificationManager manager = (NotificationManager) arg0

.getSystemService(*NOTIFICATION\_SERVICE*);

Intent notificationIntent = **new** Intent(getApplicationContext(),

Welcome.**class**);

PendingIntent pendingIntent = PendingIntent.*getActivity*(

getApplicationContext(), 0, notificationIntent,

Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);

NotificationCompat.Builder builder = **new** NotificationCompat.Builder(

arg0).setWhen(System.*currentTimeMillis*())

.setContentText("Motion has been detected. Click to view")

.setContentTitle("Video Survilliance")

.setSmallIcon(R.drawable.*alert*).setAutoCancel(**true**)

.setDefaults(Notification.*DEFAULT\_SOUND*)

.setTicker("Motion detected").setContentIntent(pendingIntent);

Notification notification = builder.build();

manager.notify((**int**) System.*currentTimeMillis*(), notification);

}

**protected** **void** onRegistered(Context arg0, fi**nal** String arg1) {

Log.*d*("Registration received", "Registration received " + arg1);

Gcmpreference.*saveString*(GCMIntentService.**this**, Gcmpreference.*GCM\_ID*, arg1);

Log.*i*("reg",""+Gcmpreference.*getString*(GCMIntentService.**this**, Gcmpreference.*GCM\_ID*, ""));

**protected** **void** onUnregistered(Context arg0, String arg1) {

}

}

**GCM PREFERENCE:**

**import** android.content.Context;

**import** android.content.SharedPreferences;

**import** android.content.SharedPreferences.Editor;

**public** **class** Gcmpreference {

**public** **static** **final** String *GCM\_ID* = "GCM\_ID";

**private** Context context;

**public** **static** **final** String *PREFERENCE\_NAME* = "VIDEO\_SURV";

**public** **static** **final** **int** *MODE* = Context.*MODE\_PRIVATE*;

**public** Gcmpreference(Context context) {

**this**.context = context;

}

**public** **static** **void** saveString(Context context, String key, String value) {

*getEditor*(context).putString(key, value).commit();}

48

**public** **static** String getString(Context context, String key, String defValue) {

**return** *getPreference*(context).getString(key, defValue);

}

**public** **static** SharedPreferences getPreference(Context context) {

**return** context.getSharedPreferences(Gcmpreference.*PREFERENCE\_NAME*,

Gcmpreference.*MODE*);

}

**public** **static** Editor getEditor(Context context) {

**return** *getPreference*(context).edit();

}

}

**IMAGE EXAMPLE:**

**import** java.io.IOException;

**public** **class** ImageExample **extends** Activity {

EditText inputUrl;

Button image\_btn;

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*imageex*);

inputUrl = ((EditText) findViewById(R.id.*imageUrl*));

inputUrl.setText(R.string.*ipaddress*

+ getSharedPreferences("filename", Context.*MODE\_PRIVATE*)

.getString("filename", ""));

inputUrl.setSingleLine();

inputUrl.setTextSize(18);

image\_btn = (Button) findViewById(R.id.*getImageButton*);

}

**private** Drawable ImageOperations(Context ctx, String url) {

**try** {

InputStream is = (InputStream) **this**.fetch(url);

Drawable d = Drawable.*createFromStream*(is, "src");

**return** d;

} **catch** (MalformedURLException e) {

e.printStackTrace();

**return** **null**;

} **catch** (IOException e) {

e.printStackTrace();

**return** **null**;

}

}

**public** Object fetch(String address) **throws** MalformedURLException,

IOException {

URL url = **new** URL(address);

Object content = url.getContent();

**return** content;}}

**MY BROADCAST RECEIVER:**

**import** android.content.BroadcastReceiver;

**import** android.content.Context;

**import** android.content.Intent;

**import** android.util.Log;

**public** **class** MyBroadcastReceiver **extends** BroadcastReceiver{

@Override

**public** **void** onReceive(Context arg0, Intent arg1) {

**if** (arg1.equals("com.google.android.c2dm.intent.REGISTRATION")) {

handleRegistration(arg0, arg1);

Log.*d*("Registration received", "Registration received");}

**else** **if** (arg1.equals("com.google.android.c2dm.intent.RECEIVE")) {

handleMessage(arg0, arg1);

}}

**private** **void** handleMessage(Context arg0, Intent arg1) {

}

**private** **void** handleRegistration(Context arg0, Intent arg1) {

String regis\_id = arg1.getStringExtra("registration\_id");

**if** (regis\_id != **null**) {

Log.*d*("Registration Id", " "+regis\_id);

}

**else** **if** (arg1.getStringExtra("error") != **null**) {

Log.*d*("Registration Error", ""+arg1.getStringExtra("error"));

}

}}

**SERVER IP ADDRESS:**

**package** com.android;

**public** **class** ServerIPAddress {

**static** String *address* = **null**;

**public** **static** String getAddress() {

**return** *address*;

}

**public** **static** **void** setAddress(String address) {

ServerIPAddress.*address* = address;

}

}

**SIGNUP:**

**import** java.io.BufferedReader;

**public** **class** Signup **extends** Activity {

Button signup\_back\_btn, signup\_submit;

EditText signup\_username, signup\_password, signup\_reenter\_password,

email\_edit, phone\_edit;

**boolean** checkUser = **false**;

**private** Context context;

ProgressDialog pd;

HttpClient httpClient;

HttpPost httpPost;

HttpResponse httpResponse;

**int** len;

**int** i;

**int** num=0;

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*signup*);

createWidgetId();

signup\_back\_btn.setOnClickListener(**new** OnClickListener() {

**public** **void** onClick(View v) {

finish();

}

});

signup\_submit.setOnClickListener(**new** OnClickListener() {

@SuppressLint("NewApi")

**public** **void** onClick(View v) {

**if** (signup\_username.getText().toString().isEmpty()) {

Toast.*makeText*(Signup.**this**, "Please Enter your username",

Toast.*LENGTH\_LONG*).show();}

**else** **if** (signup\_password.getText().toString().isEmpty()) {

Toast.*makeText*(Signup.**this**, "Please Enter your password",Toast.*LENGTH\_LONG*).show();

}

**else** **if** (signup\_reenter\_password.getText().toString().isEmpty()) {

Toast.*makeText*(Signup.**this**,"Please Re-Enter your password",

Toast.*LENGTH\_LONG*).show();

}

**else** **if** (email\_edit.getText().toString().isEmpty()) {

Toast.*makeText*(Signup.**this**,"Please Enter your email-address",Toast.*LENGTH\_LONG*).show();

}

**else** **if**(!email\_edit.getText().toString().isEmpty())

{ len=email\_edit.getText().toString().length();

**for**(i=0;i<len;i++)

{

**if**(email\_edit.getText().toString().charAt(i)=='@'){

num++;

}

} **if**(num==0)

{

Toast.*makeText*(getApplicationContext(), "Enter the valid mail.ID", Toast.*LENGTH\_LONG*).show();

}

**else** **if** (phone\_edit.getText().toString().isEmpty()) {

Toast.*makeText*(Signup.**this**,"Please Enter your phone number", Toast.*LENGTH\_LONG*).show();

}

**else** **if**(!phone\_edit.getText().toString().isEmpty()&&phone\_edit.getText().toString().length()!=10)

{

Toast.*makeText*(getApplicationContext(),"Enter The Valid PhoneNumbere",Toast.*LENGTH\_LONG*).show();

}

**else** {

**if** (signup\_password.getText().toString().equals(signup\_reenter\_password.getText().toString())) {

pd = ProgressDialog.*show*(Signup.**this**, "","Logging in...", **false**, **true**);

**new** Thread() {

@Override

**public** **void** run() {

**try** {

httpClient = **new** DefaultHttpClient();

httpPost = **new** HttpPost(Signup.**this**.getResources().getString(R.string.*loginip*));

List<NameValuePair> nameValuePair = **new** ArrayList<NameValuePair>();

nameValuePair.add(**new** BasicNameValuePair("flag", "1"));

nameValuePair.add(**new** BasicNameValuePair("email", email\_edit.getText().toString()));

nameValuePair.add(**new** BasicNameValuePair("password", signup\_password.getText().toString()));

nameValuePair.add(**new** BasicNameValuePair("username", signup\_username.getText().toString()));

nameValuePair.add(**new** BasicNameValuePair("phone", phone\_edit.getText().toString()));

httpPost.setEntity(**new** UrlEncodedFormEntity(nameValuePair));

httpResponse = httpClient.execute(httpPost);

InputStream inputStream = httpResponse.getEntity().getContent();

InputStreamReader inputStreamReader = **new** InputStreamReader(inputStream);

BufferedReader bufferedReader = **new** BufferedReader(inputStreamReader);

StringBuilder stringBuilder = **new** StringBuilder();

String bufferedStrChunk = **null**;

**while** ((bufferedStrChunk = bufferedReader.readLine()) != **null**)

{

stringBuilder.append(bufferedStrChunk);

}

**if** (stringBuilder.toString().trim().equals("yes"))

{

handler.sendEmptyMessage(1);

}

**Else**

{

handler.sendEmptyMessage(2);

}}

**catch** (Exception e)

{

handler.sendEmptyMessage(3);

}

}

}

start();

}

**else**

{

Toast.*makeText*(Signup.**this**,"Re-entered password is wrong!", Toast.*LENGTH\_LONG*).show();

}}}}

});

}

**private** **void** createWidgetId()

{

signup\_submit = (Button) findViewById(R.id.*signup\_submit*);

signup\_back\_btn = (Button) findViewById(R.id.*signup\_back\_btn*);

signup\_username = (EditText) findViewById(R.id.*signup\_username*);

signup\_password = (EditText) findViewById(R.id.*signup\_password*);

signup\_reenter\_password = (EditText) findViewById(R.id.*signup\_reenter\_password*);

email\_edit = (EditText) findViewById(R.id.*email\_edit*);

phone\_edit = (EditText) findViewById(R.id.*phone\_edit*);

}

**private** Handler handler = **new** Handler() {

**public** **void** handleMessage(Message msg)

{

**switch** (msg.what)

{

**case** 1:

pd.dismiss();

AlertDialog.Builder alert = **new** AlertDialog.Builder(Signup.**this**);

alert.setCancelable(**false**);

alert.setTitle("Registration successfull !");

alert.setMessage("You have been successfully registered with Email");

alert.setPositiveButton("Done",**new** DialogInterface.OnClickListener()

{

**public** **void** onClick(DialogInterface dialog, **int** which)

{

GCMRegistrar.*register*(Signup.**this**, R.string.*registrationno*+"");

finish();

}});

alert.show();

**break**;

**case** 2:

pd.dismiss();

Toast.*makeText*(Signup.**this**, "Your username and password is wrong !", Toast.*LENGTH\_LONG*).show();

**break**;

**case** 3:

pd.dismiss();

Toast.*makeText*(Signup.**this**, "Please check your internet connection or URL!",

Toast.*LENGTH\_LONG*).show();

**break**;

}

}

};

}

**WELCOME:**

**import** java.io.BufferedReader;

**public** **class** Welcome **extends** Activity {

Button submit, reset, exit, signup;

EditText txt\_user, txt\_pass;

SQLiteDatabase db;

**boolean** user = **false**, pass = **false**;

ProgressDialog pd;

HttpClient httpClient;

HttpPost httpPost;

HttpResponse httpResponse;

**private** Context context;

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

createWidgetId();

submit.setOnClickListener(**new** OnClickListener() {

@SuppressLint("NewApi")

**public** **void** onClick(View v) {

**if** (((!txt\_user.getText().toString().isEmpty()))

&& ((!txt\_pass.getText().toString().isEmpty()))) {

pd = ProgressDialog.*show*(Welcome.**this**, "", "Logging in...",**false**, **true**);

**new** Thread() {

@Override

**public** **void** run() {

**try** {

httpClient = **new** DefaultHttpClient();

httpPost = **new** HttpPost(Welcome.**this**.getResources().getString(R.string.*loginip*));

List<NameValuePair> nameValuePair = **new** ArrayList<NameValuePair>();

nameValuePair.add(**new** BasicNameValuePair("flag", "2"));

GCMRegistrar.*register*(Welcome.**this**, "71702409084");

nameValuePair.add(**new** BasicNameValuePair("email",txt\_user.getText().toString()));

nameValuePair.add(**new** BasicNameValuePair("password", txt\_pass.getText().toString()));

nameValuePair.add(**new** BasicNameValuePair("gcm\_id", Gcmpreference.*getString*(Welcome.**this**, Gcmpreference.*GCM\_ID*, "")));

httpPost.setEntity(**new** UrlEncodedFormEntity(nameValuePair));

httpResponse = httpClient.execute(httpPost);

InputStream inputStream = httpResponse.getEntity().getContent();6

InputStreamReader inputStreamReader = **new** InputStreamReader(inputStream);

BufferedReader bufferedReader = **new** BufferedReader(inputStreamReader);

StringBuilder stringBuilder = **new** StringBuilder();

String bufferedStrChunk = **null**;

**while** ((bufferedStrChunk = bufferedReader.readLine()) != **null**) {

stringBuilder.append(bufferedStrChunk);

}

System.*out*.println("Login :"+ stringBuilder.toString());

**if** (stringBuilder.toString().trim().equals("yes")) {

handler.sendEmptyMessage(1);

} **else** {

handler.sendEmptyMessage(2);

}

} **catch** (Exception e) {

handler.sendEmptyMessage(3);

}}}.start();

} **else** {

Toast.*makeText*(getApplicationContext(),"Username or Password fields can't be empty",Toast.*LENGTH\_LONG*).show();

}}});

reset.setOnClickListener(**new** OnClickListener() {

**public** **void** onClick(View v) {

txt\_user.setText("");

txt\_pass.setText("");

}});

signup.setOnClickListener(**new** OnClickListener() {

**public** **void** onClick(View v) {

startActivity(**new** Intent(Welcome.**this**, Signup.**class**));

}

});

exit.setOnClickListener(**new** OnClickListener() {

**public** **void** onClick(View v) {

finish();

}});

}

**private** Handler handler = **new** Handler() {

**public** **void** handleMessage(Message msg) {

**switch** (msg.what) {

**case** 1:

pd.dismiss();

GCMRegistrar.*register*(Welcome.**this**, R.string.*registrationno*+"");

Intent intent = **new** Intent(Welcome.**this**, DisplayImage.**class**);

startActivity(intent);

**break**;

**case** 2:

pd.dismiss();

Toast.*makeText*(Welcome.**this**,

"Your username and password is wrong !",

Toast.*LENGTH\_LONG*).show();

**break**;

**case** 3:

pd.dismiss();

Toast.*makeText*(Welcome.**this**,

"Please check your internet connection or URL!",

Toast.*LENGTH\_LONG*).show();

**break**;}}};

**private** **void** createWidgetId() {

submit = (Button) findViewById(R.id.*submit*);

signup = (Button) findViewById(R.id.*singup*);

exit = (Button) findViewById(R.id.*exit*);

reset = (Button) findViewById(R.id.*reset*);

txt\_user = (EditText) findViewById(R.id.*txt\_userName*);

txt\_pass = (EditText) findViewById(R.id.*txt\_pass*);

}

@Override

**public** **void** onDestroy() {

**super**.onDestroy();

}

}