```
package com.capstone.ocelot;

import java.io.File;

@SuppressWarnings("deprecation")
@TargetApi(11)
public class SoundBoardActivity extends Activity implements TextToSpeech.OnInitListener{

    ArrayList<SoundBoardItem> mGridItems;
    ArrayList<SoundBoardItem> mSequenceItems;
    int mSequenceLocation = 0;
    SoundBoardItem mCurrentItem;
    Iterator<SoundBoardItem> sequenceIterator;
    MediaPlayer mPlayer;

    //New Item Window
    ImageButton picButton;
    Uri newItemURI;

    //int MY_DATA_CHECK_CODE = 0;
    static final int PICK_IMAGE = 1;
    String userName = "Jesse";
    private TextToSpeech ttsPlayer;
    ScrollView scrollView;
    Gallery sequenceView;
    GridView gridView;
    boolean isDeleteMode = false;

    //View Adapters
    SoundBoardSequenceAdapter seqAdapter;
    SoundBoardGridAdapter gridAdapter;

    void SequenceNext(){
        AdvanceCurrentLocation();
        LoadNextSound();
    }

    OnCompletionListener MediaCompletionListener = new OnCompletionListener() {
        public void onCompletion(MediaPlayer mp) {
            SequenceNext();
        }
    };

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //Create Directory to store data in; ignore if already created
        File directory = new File(Environment.getExternalStorageDirectory() + File.separator +
"vocalot");
        directory.mkdirs();

        //Set and load the soundboardItems
        //TODO This will need to be extended so that we can load from a database!
        LoadState(); //Load the initial items to the grid
```

```
        //Setup the gridview adapter
        gridView = (GridView) findViewById(R.id.gridview);
        gridAdapter = new SoundBoardGridAdapter(this);
        gridView.setAdapter(gridAdapter);

        //Setup the Listener for the SequenceBar Container
        mSequenceItems = LoadSequenceBoard();
        scrollView = (ScrollView) findViewById(R.id.seqscrollview);
        scrollView.setOnDragListener(new MyDragListener());

        //Create the TTS Device
        ttsPlayer = new TextToSpeech(this, this);

        //Accelerometer Support
        mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        mSensorManager.registerListener(mSensorListener,
mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER), SensorManager.SENSOR_DELAY_NORMAL);
        mAccel = 0.00f;
        mAccelCurrent = SensorManager.GRAVITY_EARTH;
        mAccelLast = SensorManager.GRAVITY_EARTH;

        sequenceView = (Gallery) findViewById(R.id.seqgallery);
        seqAdapter = new SoundBoardSequenceAdapter(this);
        sequenceView.setAdapter(seqAdapter);
        sequenceView.setOnItemClickListener(new OnItemClickListener() {  //Play the sound
associated with the object.
            public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                    long arg3) {
                //Reset the position of the sequence bar
                mSequenceLocation = arg2;
                sequenceView.setSelection(mSequenceLocation);
                sequenceView.invalidate();
                sequenceIterator = mSequenceItems.iterator();
                LoadNextSound(arg2);
            }
        });

        sequenceView.setOnItemLongClickListener(new OnItemLongClickListener() {
            @Override
            public boolean onItemLongClick(AdapterView<?> parent, View view,
                    int position, long id) {
                mSequenceItems.remove(position);
                updateSequenceBar();
                return false;
            }
        });
    }

    public void UpdateGrid() {
        Collections.sort(mGridItems);
        gridAdapter.notifyDataSetChanged();
        gridView.invalidateViews();
        gridView.setAdapter(gridAdapter);
    }

    public ArrayList<SoundBoardItem> getGridItems(){
```

```
    return mGridItems;
}

public SoundBoardItem getGridItem(int index){
    return mGridItems.get(index);
}

public ArrayList<SoundBoardItem> getSequenceitems(){
    return mSequenceItems;
}

public SoundBoardItem getSequenceItem(int index){
    return mSequenceItems.get(index);
}

public int getGridSize(){
    return mGridItems.size();
}

public int getSequenceSize(){
    return mSequenceItems.size();
}

public void removeSequenceItem(int position){
    mSequenceItems.remove(position);
}

public void setCurrentItemFromGrid(int position){
    setCurrentItem(mGridItems.get(position));
}

public void setCurrentItemFromSequence(int position){
    setCurrentItem(mSequenceItems.get(position));
}

public void setCurrentItem(SoundBoardItem currentItem){
    mCurrentItem = currentItem;
}

public SoundBoardItem getCurrentItem(){
    return mCurrentItem;
}

public void AdvanceCurrentLocation(){
    if (mSequenceLocation < mSequenceItems.size() - 1){
        mSequenceLocation++;
        sequenceView.setSelection(mSequenceLocation);
        sequenceView.invalidate();
    }
}

public void addSequenceItem(){
    if (mCurrentItem != null)
        mSequenceItems.add(mCurrentItem);
    updateSequenceBar();
}
```

```
public void addGridItem(){
    if (mCurrentItem != null)
        mGridItems.add(mCurrentItem);
    VerifySoundBank(mGridItems);
    UpdateGrid();
}

class MyDragListener implements OnDragListener {
    public boolean onDrag(View v, DragEvent event) {
        switch (event.getAction()) {
        case DragEvent.ACTION_DROP: //If the drag ended on the sequence bar
            addSequenceItem();
            break;
        default:
            break;
        }
        return true;
    }
}

public void updateSequenceBar(){
    sequenceView.setAdapter(seqAdapter);
    sequenceView.invalidate();
}

public void removeGridItem(int position){
    mGridItems.remove(position);
    UpdateGrid();
}

public void LoadNextSound(){
    SoundBoardItem item;
    if (sequenceIterator.hasNext()){
        item = sequenceIterator.next();
        mPlayer = item.getMediaPlayer(getBaseContext());
        mPlayer.setOnCompletionListener(MediaCompletionListener);
        mPlayer.start();
    } else {
        mPlayer.release();
    }
}

public boolean isDeleteMode(){
    return isDeleteMode;
}

//Advance a number of positions so that you are on the one that the user has clicked.
public void LoadNextSound(int position){
    for (int i = 0; i < position; i++){
        sequenceIterator.next();
    }
    LoadNextSound();
}

public Uri getUriForId(int resId){
    Resources resources = getBaseContext().getResources();
    return Uri.parse(ContentResolver.SCHEME_ANDROID_RESOURCE + "://" +
```

```
            resources.getResourcePackageName(resId) + '/' +
            resources.getResourceTypeName(resId) + '/' +
            resources.getResourceEntryName(resId) );
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.activity_default, menu);
    return true;
}

private void SaveState(){
    ObjectOutputStream os;
    try {
        FileOutputStream fos = this.openFileOutput("database.ser", Context.MODE_PRIVATE);
        os = new ObjectOutputStream(fos);
        os.writeObject(mGridItems);
        os.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

@SuppressWarnings("unchecked") //The object is going to be correct, no need to check it.
private void LoadState(){
    mGridItems = new ArrayList<SoundBoardItem>();
    try {
        FileInputStream fis = this.openFileInput("database.ser");
        ObjectInputStream is = new ObjectInputStream(fis);
        try {
            mGridItems = (ArrayList<SoundBoardItem>) is.readObject();
            if (mGridItems.size() == 0){
                LoadDefaultGrid();
            }
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        is.close();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

 /* put this into your activity class */
  private SensorManager mSensorManager;
  private float mAccel; // acceleration apart from gravity
  private float mAccelCurrent; // current acceleration including gravity
  private float mAccelLast; // last acceleration including gravity

  private final SensorEventListener mSensorListener = new SensorEventListener() {

    public void onSensorChanged(SensorEvent se) {
      float x = se.values[0];
      float y = se.values[1];
```

```
          float z = se.values[2];
          mAccelLast = mAccelCurrent;
          mAccelCurrent = (float) Math.sqrt((double) (x*x + y*y + z*z));
          float delta = mAccelCurrent - mAccelLast;
          mAccel = mAccel * 0.9f + delta; // perform low-cut filter
        }

//      onshak

        public void onAccuracyChanged(Sensor sensor, int accuracy) {
        }
      };

      @Override
      protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(mSensorListener,
mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER), SensorManager.SENSOR_DELAY_NORMAL);
      }

      @Override
      protected void onPause() {
        mSensorManager.unregisterListener(mSensorListener);
        super.onPause();
      }


    private void clearSequenceBar(){
        mSequenceItems = new ArrayList<SoundBoardItem>();
        updateSequenceBar();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle item selection
        switch (item.getItemId()) {
        case R.id.new_game: //TODO Add 'shake and clear'
http://stackoverflow.com/questions/2317428/android-i-want-to-shake-it
//          showNewGameDialog();
            if (mGridItems.size() == 0){
                LoadDefaultGrid();
            }
            clearSequenceBar();
            return true;
        case R.id.new_item:
            showNewItemDialog();
            return true;
        case R.id.delete_mode:
            if (isDeleteMode) {
                endDeleteMode(item);
            } else {
                showDeleteModeDialog(item);
            }
            return true;
        case R.id.save:
            SaveState();
            return true;
```

```
        case R.id.load:
            LoadState();
            UpdateGrid();
            return true;
        case R.id.help:
            AlertDialog.Builder dlgAlert  = new AlertDialog.Builder(this);
            dlgAlert.setMessage("If You Need Support Please Email or Call me at:\nPhone - (978)
257-1697\nEmail - JesseCWhitworth@gmail.com");
            dlgAlert.setTitle("Support Information");
            dlgAlert.setPositiveButton("OK", null);
            dlgAlert.setCancelable(false);
            dlgAlert.create().show();
            return true;
        default:
            return super.onOptionsItemSelected(item);
        }
    }

    private void endDeleteMode(MenuItem item){
        item.setTitle(R.string.delete_mode);
        isDeleteMode = false;
        UpdateGrid();
    }

    String sanitizePath3gp(String path) {
        if (!path.startsWith("/")) {
          path = "/" + path;
        }
        if (!path.contains(".")) {
          path += ".3gp";
        }
        return Environment.getExternalStorageDirectory().getAbsolutePath() + path;
    }

    String sanitizePathWav(String path) {
        if (!path.startsWith("/")) {
          path = "/" + path;
        }
        if (!path.contains(".")) {
          path += ".wav";
        }
        return Environment.getExternalStorageDirectory().getAbsolutePath() + path;
    }

    String sanitizePathSer(String path) {
        if (!path.startsWith("/")) {
          path = "/" + path;
        }
        if (!path.contains(".")) {
          path += ".ser";
        }
        return Environment.getExternalStorageDirectory().getAbsolutePath() + path;
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == PICK_IMAGE) {
```

```
            if (resultCode == RESULT_OK) {
                if (data != null){
                    mCurrentItem.setIconResourceId(data.getData());
                }
                addGridItem();
                SaveState(); //Always save the state after you add a new item.
            }
        }
    }

    //NEW ITEM
    void showNewItemDialog(){
        final AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setCancelable(false);
        LayoutInflater inflater = getLayoutInflater();
        ViewGroup parent = (ViewGroup) inflater.inflate(R.layout.popup, null);

        final EditText dItemName = (EditText) parent.findViewById(R.id.item_name);
        final ImageButton recordButton = (ImageButton) parent.findViewById(R.id.record_button);
        recordButton.setEnabled(false);
        final ImageButton playButton = (ImageButton) parent.findViewById(R.id.play_button);
        playButton.setEnabled(false);

        final CheckBox imageCheckBox = (CheckBox) parent.findViewById(R.id.image_checkbox);

        //Setup Description Text
        dItemName.addTextChangedListener(new TextWatcher() {
            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
            }

            @Override
            public void beforeTextChanged(CharSequence s, int start, int count,
                    int after) {
            }

            @Override
            public void afterTextChanged(Editable s) {
                if(s.length() > 0){
                    recordButton.setEnabled(true);
                    playButton.setEnabled(true);
                    mCurrentItem = new SoundBoardItem(getBaseContext(),
dItemName.getText().toString());
                } else {
                    recordButton.setEnabled(false);
                    playButton.setEnabled(false);
                }
            }
        });

        //Setup Media Recorder
        AudioManager audioManager =
(AudioManager)getBaseContext().getSystemService(Context.AUDIO_SERVICE);
        audioManager.setMode(AudioManager.MODE_NORMAL);

        final MediaRecorder mediaRecorder = new MediaRecorder();
        mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
```

```
        mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.AMR_NB);
        mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
        mediaRecorder.setAudioChannels(1);
        mediaRecorder.setAudioSamplingRate(44100);
        mediaRecorder.setMaxDuration(3000);

        //TODO http://www.benmccann.com/dev-blog/android-audio-recording-tutorial/
        recordButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                String finalPath = sanitizePath3gp("/vocalot/" + dItemName.getText().toString());
                mediaRecorder.setOutputFile(finalPath);
                mCurrentItem.setSoundResourceId(finalPath);
                try {
                    mediaRecorder.prepare();
                } catch (IllegalStateException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                mediaRecorder.start();
                recordButton.setEnabled(false); //Disable the button afterwards
            }
        }); //TODO This should enable the play button, not necessarily the description field.

        //Setup Media Player
        playButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                mPlayer = mCurrentItem.getMediaPlayer(getBaseContext());
                mPlayer.start();
            }
        });

        builder.setView(parent);
        builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int whichButton) {
                dialog.dismiss();
                if (imageCheckBox.isChecked()){
                    Intent BrowsePictureIntent = new Intent();
                    BrowsePictureIntent.setType("image/*");
                    BrowsePictureIntent.setAction(Intent.ACTION_GET_CONTENT);
                    startActivityForResult(Intent.createChooser(BrowsePictureIntent, "Select
Picture"), PICK_IMAGE);
                } else {
                    addGridItem();
                }
            }
        });
        builder.setNegativeButton("CANCEL", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
                UpdateGrid();
            }
        });
```

```
    builder.create().show();
}

//NEW GAME CREATION
void showNewGameDialog(){
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setCancelable(true);
    builder.setTitle("New Game");
    builder.setInverseBackgroundForced(true);
    builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            clearSequenceBar();
            if (mGridItems.size() == 0){
                LoadDefaultGrid();
            }
            dialog.dismiss();
            UpdateGrid(); //Update the grid, sorted by number of plays.
        }
    });
    builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
            UpdateGrid();
        }
    });
    builder.create().show();
}

//DELETE MODE
void showDeleteModeDialog(final MenuItem menuItem){
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setCancelable(true);
    builder.setTitle("Delete Items?");
    builder.setInverseBackgroundForced(true);
    builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            isDeleteMode = true;
            menuItem.setTitle(R.string.delete_mode_off);
            dialog.dismiss();
            UpdateGrid();
        }
    });
    builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            isDeleteMode = false;
            menuItem.setTitle(R.string.delete_mode);
            dialog.dismiss();
            UpdateGrid();
        }
    });
    builder.create().show();
}

private ArrayList<SoundBoardItem> LoadSequenceBoard(){
    if (mSequenceItems != null)
        return mSequenceItems;
    ArrayList<SoundBoardItem> mLoadItems = new ArrayList<SoundBoardItem>();
```

```
        return mLoadItems;
    }

    @Override
    public void onInit(int status) {
        if (status == TextToSpeech.SUCCESS) {
            int result = ttsPlayer.setLanguage(Locale.US);
            if (result == TextToSpeech.LANG_MISSING_DATA || result ==
TextToSpeech.LANG_NOT_SUPPORTED) {
                Toast.makeText(this, "Language not supported", Toast.LENGTH_LONG).show();
                Log.e("TTS", "Language is not supported");
            }
            //ttsPlayer.speak("Welcome" + userName, TextToSpeech.QUEUE_ADD, null);
            VerifySoundBank(mGridItems);
            VerifySoundBank(mSequenceItems);
        } else {    //TTS is not initialized properly
            Toast.makeText(this, "TTS Initilization Failed", Toast.LENGTH_LONG).show();
            Log.e("TTS", "Initilization Failed");
        }
        UpdateGrid(); //Do a final update once everything is loaded.
    }

    private void VerifySoundBank(ArrayList<SoundBoardItem> soundBank){
        for (SoundBoardItem item : soundBank){
            if (!item.hasSound()){
                HashMap<String, String> myHashRender = new HashMap<String, String>();
                String destFileName = sanitizePathWav("/vocalot/" +
item.getDescription()).replaceAll("\\s",""); //Remove all white space on the file name as well
                myHashRender.put(TextToSpeech.Engine.KEY_PARAM_UTTERANCE_ID,
item.getDescription());
                ttsPlayer.synthesizeToFile(item.getDescription(), myHashRender, destFileName);
                item.setSoundResourceId(destFileName);
            }
        }
    }

    private void LoadDefaultGrid(){

        mGridItems = new ArrayList<SoundBoardItem>();

        SoundBoardItem s = new SoundBoardItem(this, "Cougar");
        s.setIconResourceId(R.drawable.cougar);
        s.setSoundResourceId(R.raw.cougar);
        mGridItems.add(s);

        s = new SoundBoardItem(this, "Chicken");
        s.setIconResourceId(R.drawable.chicken);
        s.setSoundResourceId(R.raw.chicken);
        mGridItems.add(s);

        s = new SoundBoardItem(this, "Dog");
        s.setIconResourceId(R.drawable.dog);
        s.setSoundResourceId(R.raw.dog);
        mGridItems.add(s);

        s = new SoundBoardItem(this, "Elephant");
        s.setIconResourceId(R.drawable.elephant);
```

```
        s.setSoundResourceId(R.raw.elephant);
        mGridItems.add(s);

        s = new SoundBoardItem(this, "Hug Me");
        mGridItems.add(s);

        s = new SoundBoardItem(this, "Friend");
        mGridItems.add(s);

        s = new SoundBoardItem(this, "Love");
        mGridItems.add(s);

        s = new SoundBoardItem(this, "I want");
        mGridItems.add(s);

        s = new SoundBoardItem(this, "Dancing");
        mGridItems.add(s);

        s = new SoundBoardItem(this, "Hungry");
        mGridItems.add(s);

        s = new SoundBoardItem(this, "I am");
        mGridItems.add(s);

        s = new SoundBoardItem(this, "Happy");
        mGridItems.add(s);

        s = new SoundBoardItem(this, "My");
        mGridItems.add(s);
    }
}
```