**Group Beta**

**Software Engineering**

**Project name: Mp3 Player**

**Introduction**

Building the music player will involve using the ContentResolver class to retrieve tracks on the device, the MediaPlayer class to play audio . We will also use a Service instance to play audio when the user is not directly interacting with the app.

Now We're going to build a simple mp3 player which you can embed in any webpage. In this part we will know what we have used and what will need to build the mp3 player. Graphics, code, library and lot of things we will discuss in this file. Here we go..

The source code includes one mp3 file that we'll need: Tarzan's mighty yell :). we can also use our own mp3 .

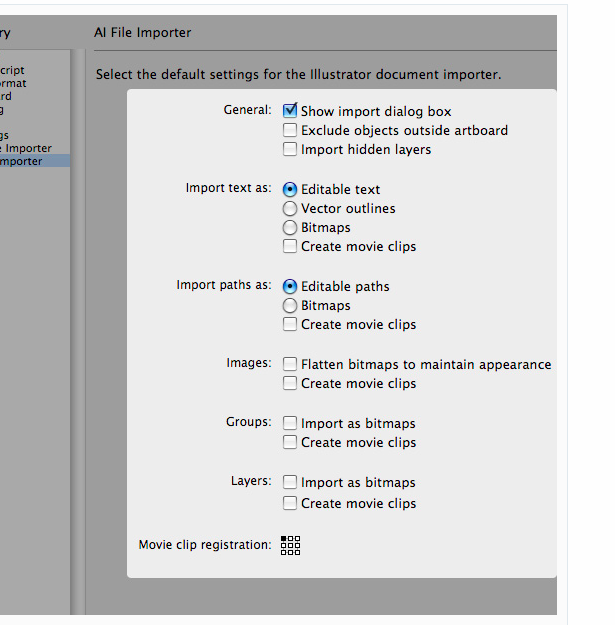
Also in the source code you'll find the directory "ai" which contains the simplemp3player.ai file. This file includes all the graphics for the player.

Lastly I've included font that we need. It's called 04b03 and it's a bitmap font. Install it and you're ready to go.

We need many things to do this project run. We have to create some graphics like illustrator file. Our step should be like this:

* We need to create a new Flash Actionscript 3.0 document in android studio

Flash > Preferences > AI file importer on Mac or Edit > Preferences > AI file importer on Windows.(Android Studio)



* Import illustrator file

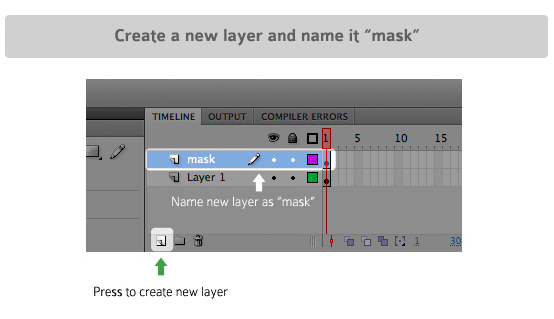
Now we're ready to import the Illustrator file. Go to File > Import > Import to Stage. Locate ".ai" from the source package "ai" directory

* Create a button sprite

we probably noticed play and pause buttons in the last image. Those are going to be our button sprite that controls the sound. All the buttons are grouped into one group. We Give it the name "buttons" and make sure it selected MovieClip as the type. Set the registration to the top left corner and click OK. And we Switch to the Properties panel and give the object a instance name "buttons".

* Manual Masking

Now our button sprite is almost ready to rock. All we need to do is mask it so that it shows just one button at a time.Double-click buttonSprite movieclip. Create a new layer above the active layer and name it "mask". Select the mask object and cut it using "Cmd + X" on Mac. Select the mask layer and paste using "Cmd + V" on Mac .It doesn't matter where mask object is because next we align it to right place. The "Align top edge" and "Align left edge" buttons and mask object should now be in the correct position, the top left corner of movieclip.The only thing remaining to do is mask the buttons. Right-click above the mask layer and choose Mask.



* Creating display

Now forget the buttons for a while and focus on the MP3 player's display. Now we will work on how to build this display element manually. let's do it manually this time. First we select the mask element from mask layer. And Convert it to a movieclip and give it the name "name". Use this name as the instance name too. The mask for display is ready

* Text field to display

We need textfields to show our information (artist, song name, loading percentage and song length). We'll create textfields manually to the stage.

* Documents class file

All we need to do is include our new loader-textfield and playerTexts-movieclip in one movieclip so we can control texts as and when we like.

* Masking with code
* Methods , Button Modes and BS
* And Finally the code we will use

**This method is our main method.**

{

mp3File=new Sound();

mp3File.addEventListener(ProgressEvent.PROGRESS,mp3FileLoading);

mp3File.addEventListener(Event.COMPLETE,mp3FileLoaded);

mp3File.addEventListener(IOErrorEvent.IO\_ERROR, errorLoadingSound);

mp3File.addEventListener(Event.ID3, getID3Data);

}

This code describe our main method. What it does is load the mp3-file, declare necessary EventListeners and slide the preloader to display.

**mp3FileLoading Event Method**

{

private function mp3FileLoading(e:ProgressEvent):void

{

var currentPercent:Number = Math.round(e.bytesLoaded / e.bytesTotal \* 100);

display.preloader.text = 'LOADING...' + currentPercent + '%';

if (currentPercent > 50 && id3InfoAvailable)

{

Tweener.addTween(display.preloader, {x: 200, time:1, onComplete:playMP3, onCompleteParams:[false, 0]});

mp3File.removeEventListener(ProgressEvent.PROGRESS, mp3FileLoading);

}

}

The "mp3FileLoading" method shows a percentage value of how much the mp3-file has been loaded. It also starts the song whenthe percentage climbs higher than 50%.

currentPercent is for counting the percentage value. We round it up with the "Math.round" method.

Line 4 writes the percentage value to our textfield so the user can see what's happening with loading.

Inside the "if" statement we check if the percentage is above 50 and the "id3InfoAvailable" variable is true.

If both return true, the "if" statement executes everything inside the brackets.When the if statement is true, we tween the "preloader" textfield away. When the tween is complete, Caurina calls the "playMP3" method and passes two properties to it.

## ErrorLoadingSound Event Method

{

private function mp3FileLoaded(e:Event):void

{

mp3File.removeEventListener(Event.COMPLETE, mp3FileLoaded);

mp3ile.removeEventListener(ProgressEvent.PROGRESS, updateBufferField);

Tweener.addTween(display.playerTexts.buffer, {x: 200, time:1});

}

This error handling is really basic. It only traces errors if errors occur.

This is classes we used in our code ……



Everything should be quite straight forward. In the last line we import Caurina Tweener.

We have lot of variables,

All variable are private. Only the simple mp3 player class can use them

{

private var mp3File:Sound;

private var mp3FilePosition:SoundChannel;

}

The Mp3 file sound object stores our Tarzan.Mp3 sounds. Mp3 file position is a soundChannel object we need it to tell us current mp3 player position in milliseconds. With this object we can also stop the mp3 file.

**{**

private var id3Data:ID3Info;

private var id3InfoAvailable=false;

}

The id3Data object stores mp3 sounds id3 information. For example, the song name and name of the artist. id3InfoAvailable tells our preloader that id3 info is available so code can start to play mp3File.

{

private var upldateTime:Timer;

}

updateTime is our timer object that updates the display times-textfield. You can find more info about the Timer class here. I'll go through this later.

{

private var playingSound:Boolean=false;

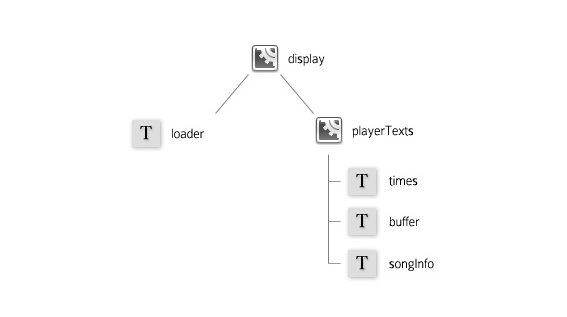
private var soundPosition:Number;

private var songReachedEnd:Boolean=false;

}

playingSound is simply a boolean variable that tell us if mp3File is currently playing. soundPosition stores a numeric value when users push the Pause button so we can continue playing music from the paused position when user clicks the Play button. songReachedEnd indicates that song has finished playing.

This is the diagram of mp3 player : how this app work in displays



**Conclusion**

This is our project what we will complete very soon. We are working on it every day . we are facing lots of probles. This is very difficult for us to do this kind project. We never did before this kind of project. I really warmed to make it happen.