17.3 Controlling multiple players

In a multimedia application it may be required to synchronize many Players. For example an applet may have to play an animation along with a background music. This implies that whenever the Players corresponding to the animation video clip is started the Player playing the background music should also be started and similarly both should be stopped simultaneously.

Another example is an application showing an documentary in which a video clip (with no audio content) has to be played in synchronization with a background voice giving the explanation for the documentary. Here a user would like to position the media time of the video clip at any arbitrary media time and then playback. The application should take care that the media time of the audio Player playing the background voice is also correspondingly set in synchronization with that of the video. Similarly if the playback rate of one player is modified the other player's rate should also be set accordingly. In these applications multiple Players have to be synchronized. One way to achieve the synchronization is as follows.

(a) First set the TimeBase of all the Players to be the same.
(b) Whatever playback control you wish to do, the program should invoke the corresponding Player method (start, stop, setMediaTime etc) on all the Players that are to be synchronized. The same is true even if a user controls the playback through a GUI.

Setting the TimeBase of all the Players to be the same is essential because unless the TimeBase of all the synchronizing Players are same, the methods setRate and setMediaTime cannot synchronize the Players. To set the TimeBase of all the Players get the TimeBase from one Player using the getTimeBase() method and set the TimeBase of the rest of the Players using the method setTimeBase().

However the above method of synchronizing the Players by explicitly invoking Player methods on all the Players is cumbersome. To solve this problem JMF allows a Player to take control over other controllers. Suppose we want a Player to be controlled by another Player. Let us call the Player to be controlled as the managedPlayer. Let us call the Player, which takes over the control as the managingPlayer. To let the managingPlayer to take control over the managedPlayer we invoke the method addController() as follows,

managingPlayer.addController(managedPlayer).

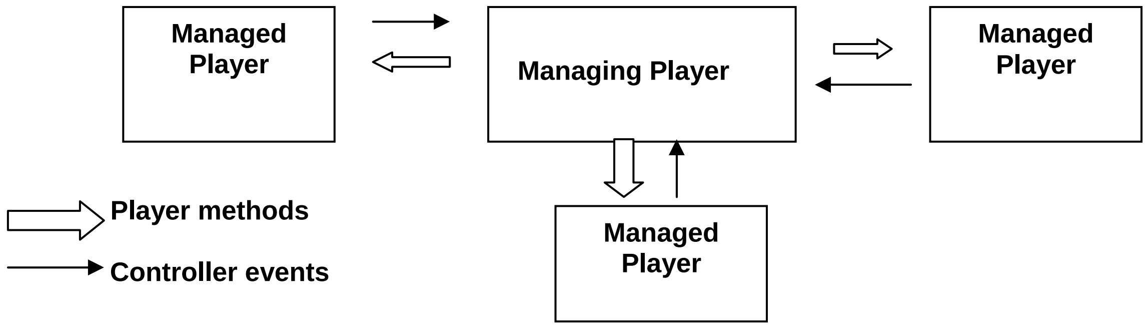

Figure 17.3 : Using one Player to control other Players.

While invoking this method the managedPlayer should be in the Realized state. If a Player method is invoked on the managingPlayer the corresponding method gets automatically invoked on the managedPlayer. A Player can control any number of Controllers using the above procedure. The managingPlayer posts a completion event only after all its managedPlayers have posted that particular event. The following program multipleControllers.java illustrates how a Player can be used to control other Players.

```
/* Program multipleControllers.java illustrates how a Player can be used to synchronize and
control other Players.
*/
import JMFUtil.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;
import java.net.*;
import javax.media.*;
import javax.media.protocol.*;
import java.awt.event.*;

public class multipleControllers {

	MediaLocator ml1= null;
	MediaLocator ml2 = null;
	Player player1 = null;
	Player player2 = null;
	static int numberofPlayers =0;
	Player masterPlayer = null;

	public  multipleControllers(){

		MediaLocator ml1 = captureDev.getAudioML();
		MediaLocator ml2 = captureDev.getVideoML();
		//MediaLocator ml3 =  new MediaLocator(
		"file://d:/book/program/media/hello.wav");

		masterPlayer = createPlayer( ml1);
		Player player1 = createPlayer( ml2);
		//Player player2 = createPlayer (ml3);

		try{
			masterPlayer.addController( player1);
			//masterPlayer.addController( player2);
		} catch( IncompatibleTimeBaseException e) {
			System.out.println( e);
		}
		masterPlayer.start();

		JFrame controlFrame = new JFrame("Multiple Controllers \n");
		JTextArea text = new JTextArea("Close this frame to stop all Controllers");
		controlFrame.getContentPane().add( text, BorderLayout.CENTER);
		controlFrame.addWindowListener(new WindowAdapter() {
```

```
                        public void windowClosing(WindowEvent we) {
                              if( masterPlayer != null){
                                    masterPlayer.stop();
                                    masterPlayer.close();
                              }
                              System.exit(0);
                        }
                  } );
                  controlFrame.pack();
                  controlFrame.setVisible(true);
}

  public static Player createPlayer( MediaLocator ml) {

                  JFrame frame = new JFrame();
                   Player player = null;
            try{
                   player = Manager.createRealizedPlayer(ml );
            }catch( java.io.IOException ioe)   { System.out.println(ioe);
            }catch( NoPlayerException npe  )   { System.out.println( npe);
            }catch( CannotRealizeException cre){ System.out.println(cre);
            }

            Component visual = player.getVisualComponent();
            if( visual != null) {
                  frame.getContentPane().add( visual, BorderLayout.CENTER);
            }

            Component  control = player.getControlPanelComponent();
            if( control != null) {
                  frame.getContentPane().add(control, BorderLayout.SOUTH);
            }

            Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
            int height = 100 *  numberofPlayers;
            numberofPlayers =  numberofPlayers +1;
            frame.setLocation(  d.width/2, height);
            frame.addWindowListener(new WindowAdapter() {
                  public void windowClosing(WindowEvent we) {
                        //player.stop();
                        //player.close();
                        //System.exit(0);
                  }
            });

            frame.pack();
            frame.setVisible(true);
            return player;
  }
```

```
	public static void main ( String args[] ) {
		multipleControllers mc = new multipleControllers();
	}
}
```

In the above program we get the MediaLocator of the audio capture device and the video capture device. We then create two Players for the capture devices using the method createPlayer(). The method createPlayer() creates a Realized Player and also displays the visual and control components of the Player. The video Player is added to the audio Player using the addController() method. This implies that the audio Player will control the video Player.

When the application ends all the resources held by the Players should be released by calling the Player.close() method. To let the user indicate when he/she wants to close the application we create a JFrame entitled “Multiple Controllers”. When this frame is closed the close() method is invoked on the master Player which will thereby close all the Controlled Player.

Let us test whether the audio Player controls the video Player. If you pause the audio Player using its control panel you should notice that the video player also gets paused. Try restarting the audio player and you should notice that the video Player gets restarted. Verify that pausing and restarting the video Player has no effect on audio Player.

Note that you are not supposed to invoke the Player methods on the controlled Player directly. The controlled Player should be controlled only through the master Player. If you do not follow this rule you may encounter problems and it is brought out by the following interesting experiment.

(I) Pause the masterPlayer. This will automatically pause the slave Player.
(ii) Now restart the slave Player.
(iii) Then you restart the master Player.

Restarting the master Player will attempt to restart the slave Player that has already been started. This results in the following error.

```
javax.media.ClockStartedError : syncStart cannot be called on an already started controller.
```