Picasso Orchestra 2.0
Systemdokumentation

Martin Engler - Projektmanager
martin.engler@fh-hagenberg.at

Alexander Brugger
alexander.brugger@fh-hagenberg.at

Dietmar Hatzenbichler
dietmar.hatzenbichler@fh-hagenberg.at

Robert Leitner
robert.leitner@fh-hagenberg.at

Inhaltsverzeichnis

1 Infos zu Applikation und Programmen

1.1 Zur Erstellung verwendete Programme

Zur Erstellung der Adobe Flash-Applikation „PicassoOrchestra 2.0" wurde in erster Linie die Programmierumgebung Adobe FlexBuilder3 verwendet, worin grundsätzlich die Formate MXML und ActionScript3 zur Anwendung kommen. Das daraus generierte Produkt wird als .swf-file in einem .htlm-file mittels CSS-Formatierung kompiliert. Für das Interface wurden die Softwarelösungen Adobe PhotoShop CS4 und Adobe Illustrator CS4 herangezogen, wobei das Adobe FlexSkiningComponent-Script für CS4 die Brücke zwischen Design- und Programmierumgebung schlägt.

1.2 Zur Verwendung benötigte Programme

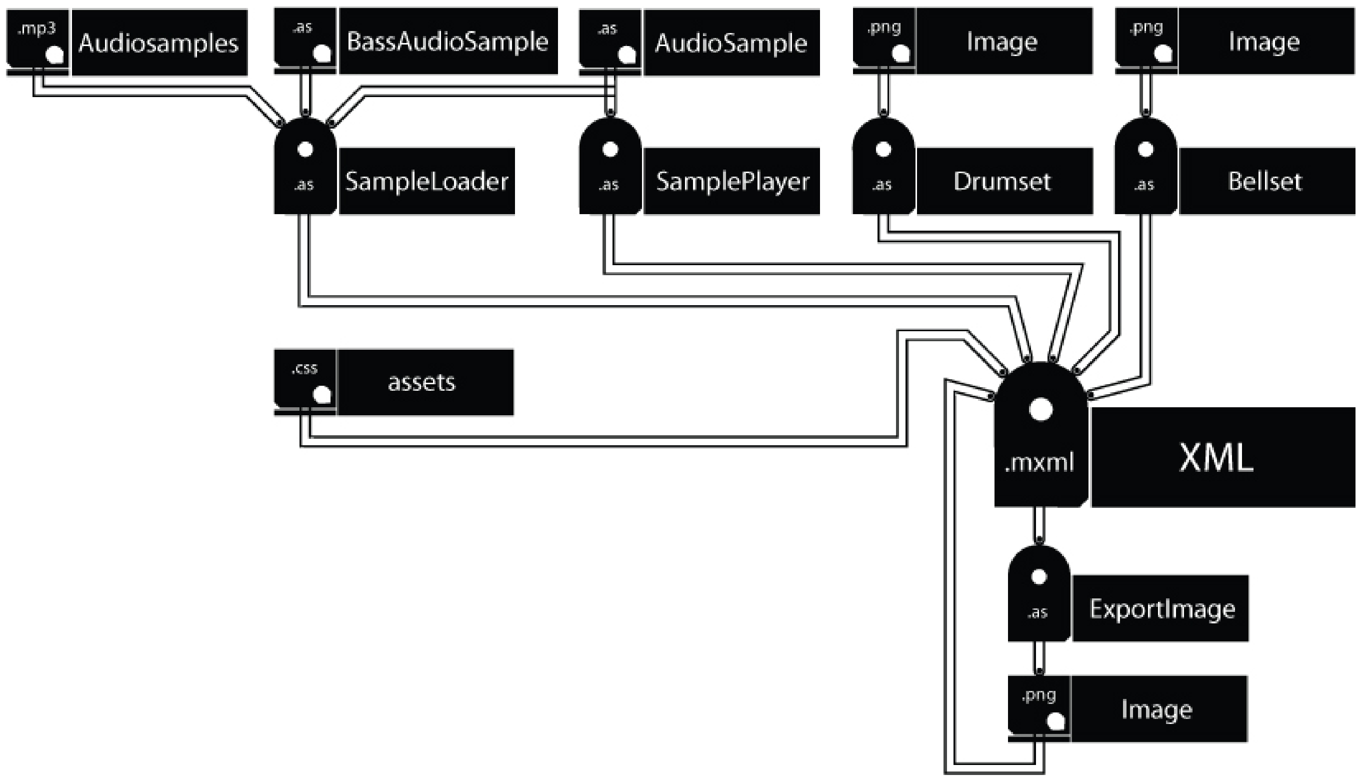
Um die Applikation problemlos verwenden zu können wird einer der untenstehenden Internetbrowser samt installiertem Adobe FlashPlayer10 vorausgesetzt. Das Update auf Adobe FlashPlayer10 ist von Nöten, da gegenüber dem Adobe FlashPlayer 9 wesentiche Erneuerungen im Handling von Audiomaterial implementiert sind.
Liste der Internetbrowser:
Mozilla FireFox 2.X/3.X – funktioniert
IE 6.X - ???
Opera 9.X – funktioniert
Safari 3.X - ???

1.3 Einschränkungen

Eine in der Flash-Community wohlbekannte, jedoch bis zum heutigen Tage nicht ausgemärzte Einschränkung bezieht sich auf die Scroll-Funktion, welche auf Macintosh-Maschinen nicht möglich ist. Somit ist die Funktionalität der Applikation in geringem Maße für OS 9/OS X – User begrenzt. Weiteres hierzu lesen Sie bitte in der User-Dokumentation.

2 Programmaufbau

2.1 Klassenhierarchie



2.2 Klassen und Funktionen – Zusammenspiel

In der Init()-Funktion werden die Variablen angelegt, grafische Komponenten werden geladen und die entsprechenden Listener hinzugefügt.
Der SampleLoader wird angelegt (dieser ist verantwortlich für das Laden der MP3s vom Server und das Umwandeln in ein/mehrere ByteArray(s)). Auf diesen wird mit Event.COMPLETE()ein Listener gelegt, der zum vollständigen Laden der Audiodateien benötigt wird.
Nach vollständigem Laden wird der Player angelegt wobei ByteArrays in SampleData zur Weiterverarbeitung umgewandelt werden.

2.2.1 Drumset

Drumset erbt von der in Flex/AS zur Verfügung gestellten Image-Klasse. Jedes Percussion-Icon, dass auf die Bühne gezogen wird, ist ein Drumset und beinhaltet das Icon selbst, die Volume, den Pitch und die Position auf der Bühne an sich.

Drumset.as

```
package Tools
{
		import mx.controls.Image;

		public class Drumset extends Image
		{
				private var loudness:Number = 0.5;
				public var newLoudIcon:Image = new Image();
				public var xwertmitteSnap:uint;

				//public var glowImage:Glow = new Glow();


						[Embed(source='../pics/Mute.png')]
	public var Mute:Class;
					[Embed(source='../pics/Loud1.png')]
	public var Loud1:Class;
	[Embed(source='../pics/Loud2.png')]
	public var Loud2:Class;
	[Embed(source='../pics/Loud3.png')]
	public var Loud3:Class;
	[Embed(source='../pics/Loud4.png')]
	public var Loud4:Class;
	[Embed(source='../pics/Loud5.png')]
	public var Loud5:Class;

				public function Drumset()
				{
						super();
						// LautstärkeIcon wird initialisiert
						newLoudIcon.width = 70;
						newLoudIcon.height = 70;
						newLoudIcon.name = „Loudness“;
						newLoudIcon.x -= (newLoudIcon.width/2)-25; //Die minus 25 ist di Hälfte der breite des Icons
						newLoudIcon.y -= (newLoudIcon.height/2)-25; //Die minus 25 ist di Hälfte der höhe des Icons
						newLoudIcon.source = Loud3;
						this.addChild(newLoudIcon);
				}




		public function setLoudness(newloud:Number):void{
				if(newloud >= 0 && newloud <= 1 )
				this.loudness = newloud;
		}
		public function setLoudnessPLUS():void{
				trace(„loudness:“+loudness);
				if(loudness+0.1 < 1){
				this.loudness += 0.1;
				}else{
						this.loudness = 1;
				}
		}
		public function setLoudnessMINUS():void{
				trace(„MINUS“);
				if(loudness-0.1 > 0){
				this.loudness -= 0.1;
				}else{
						this.loudness = 0;
				}


		}


		public function getLoudness():Number{
				return this.loudness;
		}

		public function getLoudIcon():void{
				if (this.loudness == 0){
				newLoudIcon.source = Mute;
		   }
				else if (this.loudness > 0 && this.loudness <= 0.2){
				newLoudIcon.source = Loud1;
				}
				else if (this.loudness > 0.2 && this.loudness <= 0.4){
				newLoudIcon.source = Loud2;
				}
				else if (this.loudness > 0.4 && this.loudness <= 0.6){
				newLoudIcon.source = Loud3;
				}
				else if (this.loudness > 0.6 && this.loudness <= 0.8){
				newLoudIcon.source = Loud4;
				}
				else if (this.loudness > 0.8 && this.loudness <= 1){
				newLoudIcon.source = Loud5;
				}
		}
		public function calcPitchwithPos(position:uint):Number{
				var pitcha:Number = (4-((position/250)+1))/2;  //komische Rechnung damit max Werte von 0.5-1.5
				return pitcha;
		}
		public function calcPoswithPitch(pitch:Number):uint{
				var pos:uint = (((4-(2*pitch))-1)*250);  //komische Rechnung damit korrekten Positionswerte
				return pos;
		}

		}
}
```

2.2.2 Bellset

Ähnlich wie Drumset ist das Bellset zuständig für zusätzliche Sounds, die durch einen selbst implementierten Algorithmus erzeugt werden.

Bellset.as

```
package Tools
{
		import mx.containers.Panel;
		import mx.controls.Image;



		public class Bellset extends Image
		{
				private var loudness:Number = 0.5;
				public var xwertmitteSnap:uint;
				private var v1:Panel;

				public function Bellset(v1:Panel)
				{
						super();
						this.v1 = v1;
				}

				public function calcStartPosSech(canvasstartX:int):uint{
		var snapPosX:uint = ((this.x)-canvasstartX)+(this.width/2);
		snapPosX = (snapPosX/50)+1;
		snapPosX = snapPosX-1;
		if(snapPosX < 0 )snapPosX = 0;
		if(snapPosX > 15)snapPosX = 15;
		return snapPosX;

				}


				public function calcQuadrant():uint{
								var quad:uint;
		var snapPosX:uint = ((this.x)-v1.x)+(this.width/2);
		snapPosX = (snapPosX/50)+1;
		xwertmitteSnap = snapPosX-1;

		if(xwertmitteSnap < 8){ //LINKS
						if(this.y < ((v1.height/2)+v1.y)){ //UNTEN
								quad = 0;
						}else{		// OBEN
								quad = 2;
						}
				}else{		//RECHTS
						if(this.y < ((v1.height/2)+v1.y)){ //UNTEN
								quad = 1;
						}else{		// OBEN
								quad = 3;
						}
				}

		return quad;
				}

				public function calcVolume():Number{

								var volume:Number;
								var mitteCanvas:uint=(v1.height/2)+v1.y;
		if(((this.y+((this.height)/2)) - mitteCanvas)<0){//ober mitte
		volume = mitteCanvas -((this.y+((this.height)/2))); //x ist der abstand zur mitte
		}else{ // unter mitte
		volume = ((this.y+((this.height)/2)))-mitteCanvas; //x ist der abstand zur mitte

		}
								return 1-(volume/250);
				}

								public function clone(belltoClone:Bellset):void{
										belltoClone.v1 = this.v1;
										belltoClone.xwertmitteSnap = this.xwertmitteSnap;
										belltoClone.x = this.x;
										belltoClone.y = this.y;
										belltoClone.source = this.source;
				}
}
}
```

2.2.3 ExportImage

Diese Klasse sorgt dafür, dass die Stage mit den daruf gezogenen Icons mit Hilfe der ImageSnapshot-Klasse als .jpg-file in der unteren BottomBar abgelegt werden. Damit kann der User Patterns abspeichern und nach Bereinigung eines noch auftretenden Serverproblems zu einem späteren Zeitpunkt wieder aufrufen. Des weiteren hat der User die Möglichkeit, die Stage als .jpg-file lokal auf dem Rechner zu speichern und so, unabhängig von einer Internetanbindung, die Wahl auch weiter an Sounds/ Patterns zu experimentieren.

ExportImage.as

```
package Tools
{
    import flash.display.*;
    import flash.events.Event;
    import flash.geom.Point;
    import flash.geom.Rectangle;
    import flash.net.FileReference;
    import flash.utils.ByteArray;
    import flash.utils.Timer;

    import mx.graphics.ImageSnapshot;
    import mx.graphics.codec.JPEGEncoder;
    import flash.utils.setTimeout;

    public class ExportImage
    {



        public static function createImageBA(ScreenshotArray:Array):BitmapData{

            var imageBitmapData:BitmapData = ImageSnapshot.captureBitmapData((ScreenshotArray[0] as IBitmapDrawable));


             for(var i:int=1; i<ScreenshotArray.length; i++){
                if(ScreenshotArray[i] != null){

                    //FUNZT NET MIT LOUD ICONS
                    if(ScreenshotArray[i] is Drumset){
                        var imageBitmapDataMerge:BitmapData = ImageSnapshot.captureBitmapData((ScreenshotArray
[i] as IBitmapDrawable));
                        var imageBitmapDataMergeLoud:BitmapData = ImageSnapshot.captureBitmapData((Screenshot
Array[i].newLoudIcon as IBitmapDrawable));
                        imageBitmapDataMerge.merge(imageBitmapDataMergeLoud, new Rectangle(0, 0, 70, 70),
                        new Point(ScreenshotArray[i].x-ScreenshotArray[0].x, ScreenshotArray[i].y-
ScreenshotArray[0].y), 256, 256, 256, 1);

                        imageBitmapData.merge(imageBitmapDataMerge, new Rectangle(0, 0, 70, 70),
                        new Point(ScreenshotArray[i].x-ScreenshotArray[0].x, ScreenshotArray[i].y-
ScreenshotArray[0].y), 256, 256, 256, 1);
                //SO EIN SHIT ABER AUCH
                    }else{
                        var imageBitmapDataMerge:BitmapData = ImageSnapshot.captureBitmapData((ScreenshotArray
[i] as IBitmapDrawable));
                        imageBitmapData.merge(imageBitmapDataMerge, new Rectangle(0, 0, 50, 50),
                        new Point(ScreenshotArray[i].x-ScreenshotArray[0].x, ScreenshotArray[i].y-
ScreenshotArray[0].y), 256, 256, 256, 1);

                    }


                }
             }

            return imageBitmapData;
        }
        public static function createImage(ScreenshotArray:Array):Bitmap{

            var imageBitmapData:BitmapData = ImageSnapshot.captureBitmapData((ScreenshotArray[0] as IBitmapDrawable));


             for(var i:int=1; i<ScreenshotArray.length; i++){
                if(ScreenshotArray[i] != null){

                    //FUNZT NET MIT LOUD ICONS
                    if(ScreenshotArray[i] is Drumset){
```

```
                                                        var imageBitmapDataMerge:BitmapData = ImageSnapshot.captureBitmapData((ScreenshotArray
[i] as IBitmapDrawable));
                                                        var imageBitmapDataMergeLoud:BitmapData = ImageSnapshot.captureBitmapData((Screenshot
Array[i].newLoudIcon as IBitmapDrawable));
                                                        imageBitmapDataMerge.merge(imageBitmapDataMergeLoud, new Rectangle(0, 0, 70, 70),
                                                        new Point(ScreenshotArray[i].x-ScreenshotArray[0].x, ScreenshotArray[i].y-
ScreenshotArray[0].y), 256, 256, 256, 1);

                                                        imageBitmapData.merge(imageBitmapDataMerge, new Rectangle(0, 0, 70, 70),
                                                        new Point(ScreenshotArray[i].x-ScreenshotArray[0].x, ScreenshotArray[i].y-
ScreenshotArray[0].y), 256, 256, 256, 1);
                                        //SO EIN SHIT ABER AUCH
                                                }else{
                                                        var imageBitmapDataMerge:BitmapData = ImageSnapshot.captureBitmapData((ScreenshotArray
[i] as IBitmapDrawable));
                                                        imageBitmapData.merge(imageBitmapDataMerge, new Rectangle(0, 0, 50, 50),
                                                        new Point(ScreenshotArray[i].x-ScreenshotArray[0].x, ScreenshotArray[i].y-
ScreenshotArray[0].y), 256, 256, 256, 1);

                                                }


                                        }
                                }
                                   var bitmap1:Bitmap = new Bitmap(imageBitmapData); // Bild erzeugen Bitmap
                var jpg:JPEGEncoder = new JPEGEncoder(); // Jpeg decoder erzeugen
                var ba:ByteArray = jpg.encode(imageBitmapData); // Byte Array erzeugen zum lokalen abspeichern
                                          var file:FileReference = new FileReference();

                                 file.save(ba, „bild2.jpg“);

                                return bitmap1;
                        }
                        public static function saveImage(ScreenshotArray:Array):Bitmap{

                                var imageBitmapData:BitmapData = ImageSnapshot.captureBitmapData((ScreenshotArray[0] as IBitmapDrawable));


                                for(var i:int=1; i<ScreenshotArray.length; i++){
                                        if(ScreenshotArray[i] != null){

                                                //FUNZT NET MIT LOUD ICONS
                                                if(ScreenshotArray[i] is Drumset){
                                                        var imageBitmapDataMerge:BitmapData = ImageSnapshot.captureBitmapData((ScreenshotArray
[i] as IBitmapDrawable));
                                                        var imageBitmapDataMergeLoud:BitmapData = ImageSnapshot.captureBitmapData((Screenshot
Array[i].newLoudIcon as IBitmapDrawable));
                                                        imageBitmapDataMerge.merge(imageBitmapDataMergeLoud, new Rectangle(0, 0, 70, 70),
                                                        new Point(ScreenshotArray[i].x-ScreenshotArray[0].x, ScreenshotArray[i].y-
ScreenshotArray[0].y), 256, 256, 256, 1);

                                                        imageBitmapData.merge(imageBitmapDataMerge, new Rectangle(0, 0, 70, 70),
                                                        new Point(ScreenshotArray[i].x-ScreenshotArray[0].x, ScreenshotArray[i].y-
ScreenshotArray[0].y), 256, 256, 256, 1);
                                        //SO EIN SHIT ABER AUCH
                                                }else{
                                                        var imageBitmapDataMerge:BitmapData = ImageSnapshot.captureBitmapData((ScreenshotArray
[i] as IBitmapDrawable));
                                                        imageBitmapData.merge(imageBitmapDataMerge, new Rectangle(0, 0, 50, 50),
                                                        new Point(ScreenshotArray[i].x-ScreenshotArray[0].x, ScreenshotArray[i].y-
ScreenshotArray[0].y), 256, 256, 256, 1);

                                                }


                                        }
                                }
                                var bitmap1:Bitmap = new Bitmap(imageBitmapData); // Bild erzeugen Bitmap

                                return bitmap1;
                        }



                        public static function makeXML():void{
                                /*geht noch net ganz
                                var file2:FileReference = new FileReference();
                                                          var xmlFile:XML =  <image>
                                                                                <filename>bild1.jpg</filename>
                                                                                </image>;

                                                                  xmlFile.appendChild(„bild2.jpg“);
                file2.save(xmlFile,‘images.xml‘);

        */
                        }



                }

}
```

2.2.4 assets.css

Das assets.css-file skinned die Standardkomponenten der Applikation. Es bindet sämtliche FontFamilies mit in das kompilierte .swf-file ein. Weiters kann es, einmal erstellt, beliebig in jeglicher MXML/Flex-Klasse eingebunden werden. Mit wenigen Klicks kann so das Design der App völlig abgeändert werden.

assets.css

```
Application
{
   background-image: Embed(source=“assets/BGStage.jpg“);
   background-position: center;
   background-blend:    multiply;
}

Button.clearButton
{
             disabledSkin: Embed(source=“assets/Button_clearButton_disabledSkin.png“);
             downSkin: Embed(source=“assets/Button_clearButton_downSkin.png“);
             overSkin: Embed(source=“assets/Button_clearButton_overSkin.png“);
             upSkin: Embed(source=“assets/Button_clearButton_upSkin.png“);
}
Button.PauseButton
{
             disabledSkin: Embed(source=“assets/Button_PauseButton_disabledSkin.png“);
             downSkin: Embed(source=“assets/Button_PauseButton_downSkin.png“);
             overSkin: Embed(source=“assets/Button_PauseButton_overSkin.png“);
             upSkin: Embed(source=“assets/Button_PauseButton_upSkin.png“);
}
Button.playButton
{
             disabledSkin: Embed(source=“assets/Button_playButton_disabledSkin.png“);
             downSkin: Embed(source=“assets/Button_playButton_downSkin.png“);
             overSkin: Embed(source=“assets/Button_playButton_overSkin.png“);
             upSkin: Embed(source=“assets/Button_playButton_upSkin.png“);
}
DragManager
{
             copyCursor: Embed(source=“assets/DragManager_copyCursor.png“);
             moveCursor: Embed(source=“assets/DragManager_moveCursor.png“);
             rejectCursor: Embed(source=“assets/DragManager_rejectCursor.png“);
}

Panel
{
             borderSkin: Embed(source=“assets/Panel_borderSkin.png“);
             closeButtonDisabledSkin: Embed(source=“assets/Panel_closeButtonDisabledSkin.png“);
             closeButtonDownSkin: Embed(source=“assets/Panel_closeButtonDownSkin.png“);
             closeButtonOverSkin: Embed(source=“assets/Panel_closeButtonOverSkin.png“);
             closeButtonUpSkin: Embed(source=“assets/Panel_closeButtonUpSkin.png“);
             controlBarBackgroundSkin: Embed(source=“assets/Panel_controlBarBackgroundSkin.png“);
             fontFamily: Scribblicious;
             fontSize: 31;
             color: #171717;
             titleStyleName: panelTitle;
             paddingTop: 100;
             paddingBottom: 100;
}

@font-face
{
             fontFamily: Scribblicious;
             fontWeight: normal;
             fontStyle: normal;
             src: local(„Scribblicious“);
}
.panelTitle
{
             paddingLeft: 100;
             paddingRight: 100;
             fontFamily: Scribblicious;
}
Slider.volumeSlider
{
             thumbDisabledSkin: Embed(source=“assets/Slider_volumeSlider_thumbDisabledSkin.png“);
             thumbDownSkin: Embed(source=“assets/Slider_volumeSlider_thumbDownSkin.png“);
             thumbOverSkin: Embed(source=“assets/Slider_volumeSlider_thumbOverSkin.png“);
             thumbUpSkin: Embed(source=“assets/Slider_volumeSlider_thumbUpSkin.png“);
             trackHighlightSkin: Embed(source=“assets/Slider_volumeSlider_trackHighlightSkin.png“);
             trackSkin: Embed(source=“assets/Slider_volumeSlider_trackSkin.png“);
}
Button
{
             disabledSkin: Embed(source=“assets/Button_disabledSkin.png“);
             downSkin: Embed(source=“assets/Button_downSkin.png“);
             overSkin: Embed(source=“assets/Button_overSkin.png“);
             upSkin: Embed(source=“assets/Button_upSkin.png“);
             fontFamily: Scribblicious;
             fontSize: 24;
             paddingTop: 0;
             paddingBottom: 7;
             color: #000000;
             leading: 2;
}
@font-face
{
		fontFamily: Scribblicious;
		fontWeight: bold;
		fontStyle: normal;
		src: local(„Scribblicious“);
}
Button.export
{
		disabledSkin: Embed(source=“assets/Button_export_disabledSkin.png“);
		downSkin: Embed(source=“assets/Button_export_downSkin.png“);
		overSkin: Embed(source=“assets/Button_export_overSkin.png“);
		upSkin: Embed(source=“assets/Button_export_upSkin.png“);
}
Button.save
{
		disabledSkin: Embed(source=“assets/Button_save_disabledSkin.png“);
		downSkin: Embed(source=“assets/Button_save_downSkin.png“);
		overSkin: Embed(source=“assets/Button_save_overSkin.png“);
		upSkin: Embed(source=“assets/Button_save_upSkin.png“);
}
```

2.2.5 AudioSample

Die AudioSample-Klasse besteht aus einem ByteArray, dass ein Soundfile (MP3) representiert. Sie besitzt die Modifikatoren Volume und Pitch. Weiters sorgen diverse implementierte Funktionen für die Aufbereitung der ByteArray-Daten, was ausschlaggebend für die Performance der Applikation ist.

AudioSample.as

```
package Tools
{
	import flash.display.Sprite;
	import flash.utils.ByteArray;

	/**
	 * @author Martin Engler
	 */

	public class AudioSample extends Sprite
	{
		private var _originalSample:ByteArray;
		public var _actualSample:ByteArray;
		private var _volume:Number; // between 0 and 1
		private var _pitch:Number;
		public var debugcounter:int =0;

		public function AudioSample(sample:ByteArray)
		{
			_originalSample = new ByteArray();
			_actualSample = new ByteArray();
			_volume = new Number(0.5);
			_pitch = new Number(1);

			_originalSample = sample;

			for(var i:int =0; i<sample.length; i++){
				_actualSample[i] = sample[i];
			}
			//clearSound();
		}

		public function clearSound():void
		{
			var cnt:int = 0;
			var help:Number = 1-(_pitch-0.5);
			var help2:int = (50000 - 5000*help)/8;
			var newPos:int = help2*8;

			var fade:Number = 0.9;
			var x:Number=1;
			var newX:int;
			_actualSample.position = _actualSample.length - 5000;
			var workBytes:ByteArray = new ByteArray();
			workBytes.writeBytes(_actualSample);
			workBytes.position = 0;

			var checkL:Number = 0;
			var checkR:Number = 0;
			var smallestL:Number = 1;
			var smallestR:Number = 1;
			var smallPosL:int = _actualSample.length - 5000;
			var smallPosR:int = _actualSample.length - 5000;
			//trace(„pitch: „+_pitch + „ | help „ + help +“ | pos before: „ + _actualSample.position);
			while(_actualSample.position<_actualSample.length)
			{
				checkL = _actualSample.readFloat();
				checkR = _actualSample.readFloat();
				//trace(„pos: „ + _actualSample.position);
				if (checkL > 0 && checkL < smallestL){
					smallestL = checkL;
```

```
                        smallPosL = _actualSample.position -8;
                }
                if (checkR > 0 && checkR < smallestR) {
                        smallestR = checkR;
                }

                if(checkL==0 && checkR==0){
                        break;
                }
        }
        //trace(„smallestL: „ + smallestL + „ | smallestR: „ + smallestR + „ | pos: „ + smallPosL);

        if(!_actualSample.bytesAvailable){
                //trace(„kein nulldurchgang gefunden“);
                _actualSample.position = smallPosL;
        }

        while(_actualSample.position < 65535)
        {
                _actualSample.writeFloat(0);
                _actualSample.writeFloat(0);
        }

        //trace(„posAfter: „ + _actualSample.position);
}

public function clearStart():void
{
        var cnt:int = 0;
        _actualSample.position = 0
        var thisSample:Number;
        while(_actualSample.bytesAvailable)
        {
                cnt++;
                thisSample = _actualSample.readFloat();
                trace(„thisSample: „ + thisSample + „ | cnt: „ +cnt);
                if(thisSample >= 0 && thisSample < 0.01) break;
                _actualSample.readFloat();
        }
        trace(„length: „ + _actualSample.length + „ | breakPos: „ + _actualSample.position);
        _actualSample.position = 0;
        var i:int=0;

        while(i<(cnt+1))
        {
                _actualSample.writeFloat(0);
                _actualSample.writeFloat(0);
                i++;
        }
}

// volume wird gesetzt und gleich ins bytearray hineingerechnet
public function volume(vol:Number):void
{
        if (vol>1) vol = 1;
        if (vol<0) vol = 0;
        _volume = vol;

        var returnBytes:ByteArray = new ByteArray();
        var workBytes:ByteArray = new ByteArray();
        workBytes.writeBytes(_actualSample);
        workBytes.position = 0;

        while(workBytes.bytesAvailable){
                returnBytes.writeFloat(workBytes.readFloat()*vol);
        }
        returnBytes.position = 0;
        _actualSample.position = 0;

        _actualSample.writeBytes(returnBytes);
}

// pitch wird gesetzt und sofort berechnet
public function pitch(scale:Number):void
{
        if(scale < 0.5) scale = 0.5;
        if(scale > 1.5) scale = 1.5;
        _pitch = scale;
        var skipCount:Number = 0;
        var returnBytes:ByteArray = new ByteArray();
        var workBytes:ByteArray = new ByteArray();
        workBytes.writeBytes(_actualSample);

        workBytes.position = 0;
        returnBytes.position = 0;
        var step:int;
        var skipRate:Number;

        if(scale>=1){
                skipRate = 1 + (1 / (scale - 1));
                step = 8;
        } else if(scale > 0 && scale < 1){
                skipRate = 1 + (1/(1-scale));
                step = -8;
        }
        //trace(„scale: „ + scale)
        //trace(„skipRate: „ + skipRate);
        while(workBytes.bytesAvailable)
        {
                skipCount++;
                if(workBytes.length == returnBytes.length) break;

                if (skipCount <= skipRate)
                {
                        returnBytes.writeFloat(workBytes.readFloat());
                        returnBytes.writeFloat(workBytes.readFloat());
                }
```

```
                                        else
                                        {
                                                workBytes.position += step;
                                                skipCount = skipCount - skipRate;
                                        }
                                }

                                trace(„pitchRetPos: „ + returnBytes.position + „ | pitchRetLength: „ + returnBytes.length);
                                _actualSample = new ByteArray();
                                _actualSample.position = 0;
                                returnBytes.position = 0;
                                _actualSample.writeBytes(returnBytes);
                                clearSound();
                        }

                        // merged 2 bytearrays auf eins zusammen (mit gleichen anteilen [man könnts auch 2:1 oder so mixen])
                        public function merge(sample:AudioSample):AudioSample
                        {
                                var merged:ByteArray = new ByteArray();
                                var sampleBytes:ByteArray = sample.getByteArray();
                                _actualSample.position = 0;
                                sampleBytes.position = 0;
                                //trace(„actual: „ + _actualSample.length + „ sample: „ + sampleBytes.length );
                                while(_actualSample.bytesAvailable)
                                {
                                        merged.writeFloat(_actualSample.readFloat() + sampleBytes.readFloat());
                                }
                                merged.position = 0;
                                return new AudioSample(merged);
                        }

                        public function clone():AudioSample
                        {
                                return new AudioSample(this._actualSample);
                        }

                        // wird in merge() verwendet
                        public function getByteArray():ByteArray
                        {
                                var returnByteArray:ByteArray = new ByteArray();
                                returnByteArray.writeBytes(_actualSample);
                                return returnByteArray;
                        }
                }
}
```

2.2.6 SampleLoader

Der SampleLoader wird in der Applikation nur 1x erzeugt. Es werden MP3s vom Server oder der lokalen Festplatte geladen, in ein ByteArray umgewandelt und in weiterer Folge daraus die Audiosamples generiert.

SampleLoader.as

```
package Tools
{
        import flash.display.Sprite;
        import flash.events.Event;
        import flash.media.Sound;
        import flash.utils.ByteArray;

        /**
         * @author Martin Engler
         */
        public class SampleLoader extends Sprite

        {
                static public const BLOCK_SIZE: int = 8192;

                [Embed(source=‚sound/basementBass_f1.mp3‘)] public var _bassFCls:Class;
                [Embed(source=‚sound/basementBass_g1.mp3‘)] public var _bassGCls:Class;
                [Embed(source=‚sound/basementBass_a1.mp3‘)] public var _bassACls:Class;
                [Embed(source=‚sound/basementBass_h1.mp3‘)] public var _bassHCls:Class;
                [Embed(source=‚sound/basementBass_c2.mp3‘)] public var _bassCCls:Class;

                [Embed(source=‚sound/empty_logic.mp3‘)] public var _emptyCls:Class;
                [Embed(source=‚sound/bassdrumShort.mp3‘)] public var _BDCls:Class;
                [Embed(source=‚sound/snare_logic.mp3‘)] public var _SDCls:Class;
                [Embed(source=‚sound/hihatShort.mp3‘)] public var _HHCls:Class;


                private var _BD: Sound;
                private var _SD: Sound;
                private var _HH: Sound;
                private var _empty: Sound;

                private var _BDBytes:ByteArray;
                private var _SDBytes:ByteArray;
                private var _HHBytes:ByteArray;
                private var _emptyBytes:ByteArray;

                private var _loadBell:Sound;
private var _bells:Array;

private var _loadBassF:Sound;
private var _loadBassG:Sound;
private var _loadBassA:Sound;
private var _loadBassH:Sound;
private var _loadBassC:Sound;

private var _bassF:Array;
private var _bassG:Array;
private var _bassA:Array;
private var _bassH:Array;
private var _bassC:Array;

private var _bassAudioSample:BassAudioSample;

private var _loadString:Sound;
private var _strings:Array;

// konstruktor, lädt alle samples
public function SampleLoader():void
{
		_loadBassF = new _bassFCls() as Sound;
		_loadBassG = new _bassGCls();
		_loadBassA = new _bassACls();
		_loadBassH = new _bassHCls();
		_loadBassC = new _bassCCls();

		_empty = new _emptyCls() as Sound;
		_BD = new _BDCls() as Sound;
		_HH = new _HHCls();
		_SD = new _SDCls();
}

// die getters geben die sounds als bytearrays zurück
public function getBD():AudioSample
{
		var bd:AudioSample = new AudioSample(_BDBytes);
		return bd;
}

public function getSD():AudioSample
{
		return new AudioSample(_SDBytes);
}

public function getHH():AudioSample
{
		return new AudioSample(_HHBytes);
}

public function getEmpty():AudioSample
{
		var c:int = 0;
		_emptyBytes.position = 0;
		return new AudioSample(_emptyBytes);
}

public function getBellsArray(startPos:int, quadrant:int, vol:Number):Array
{
		_bassAudioSample.modifyBassArray(startPos, quadrant, vol);
		trace(„bellsarray“);
		var workArray:Array = _bassAudioSample.getArray();

		return workArray;
}

public function getStringsArray(startPos:int, quadrant:int):Array
{
		var workArray:Array = cloneArray(_strings);
		calculateSoundArray(workArray, startPos, quadrant);
		return workArray;
}

private function calculateSoundArray(array:Array, startPos:int, quadrant:int):void
{
		var returnArray:Array = new Array(16);
		var y:int = startPos;
		var startPitch:Number;
		var startVolume:Number;
		var pitchOperator:int;
		var volumeOperator:int;

		switch(quadrant){
			case 0:
				startPitch = 0.5;
				startVolume = 1;
				pitchOperator = +1;
				volumeOperator = -1;
			case 1:
				startPitch = 0.5;
				startVolume = 0;
				pitchOperator = +1;
				volumeOperator = +1;
			case 2:
				startPitch = 1.5;
				startVolume = 0;
				pitchOperator = -1;
				volumeOperator = +1;
			case 3:
				startPitch = 1.5;
				startVolume = 1;
				pitchOperator = -1;
				volumeOperator = -1;
		}

		var thisSample:AudioSample;
		for(var i:int = 0; i < 16; i++){
					thisSample = array[i] as AudioSample;
					startPitch += (1/16)*pitchOperator;
					startVolume += (1/16)*volumeOperator;
					thisSample.pitch(startPitch);
					thisSample.volume(startVolume);
					returnArray[y] = thisSample;
					y=(y+1)%16;
				}

				array = returnArray;
			}

			// erzeugt aus allen sounds bytearrays
			public function extract():void
			{
				_strings = new Array(16);
				_bells = new Array(16);
				_bassF = new Array(16);
				_bassG = new Array(16);
				_bassA = new Array(16);
				_bassH = new Array(16);
				_bassC = new Array(16);

				var thisByteArray:ByteArray;

				for(var i:int =0; i<16; i++)
				{
					thisByteArray=new ByteArray();
					//_loadBell.extract(thisByteArray, BLOCK_SIZE, BLOCK_SIZE*i);
					_bells[i] = new AudioSample(thisByteArray);

					thisByteArray=new ByteArray();
					//_loadString.extract(thisByteArray, BLOCK_SIZE, BLOCK_SIZE*i);
					_strings[i] = new AudioSample(thisByteArray);

					thisByteArray=new ByteArray();
					_loadBassF.extract(thisByteArray, BLOCK_SIZE, BLOCK_SIZE*i);
					_bassF[i] = new AudioSample(thisByteArray);

					thisByteArray=new ByteArray();
					_loadBassG.extract(thisByteArray, BLOCK_SIZE, BLOCK_SIZE*i);
					_bassG[i] = new AudioSample(thisByteArray);

					thisByteArray=new ByteArray();
					_loadBassA.extract(thisByteArray, BLOCK_SIZE, BLOCK_SIZE*i);
					_bassA[i] = new AudioSample(thisByteArray);

					thisByteArray=new ByteArray();
					_loadBassH.extract(thisByteArray, BLOCK_SIZE, BLOCK_SIZE*i);
					_bassH[i] = new AudioSample(thisByteArray);

					thisByteArray=new ByteArray();
					_loadBassC.extract(thisByteArray, BLOCK_SIZE, BLOCK_SIZE*i);
					_bassC[i] = new AudioSample(thisByteArray);
				}

				_bassAudioSample = new BassAudioSample(_bassF, _bassG, _bassA, _bassH, _bassC);

				_BDBytes = new ByteArray();
				_SDBytes = new ByteArray();
				_HHBytes = new ByteArray();
				_emptyBytes = new ByteArray();


				_BD.extract(_BDBytes, BLOCK_SIZE);
				_SD.extract(_SDBytes, BLOCK_SIZE);
				_HH.extract(_HHBytes, BLOCK_SIZE);
				_empty.extract(_emptyBytes, BLOCK_SIZE);

				trace(„Emty: „ + _emptyBytes.length);
				trace(„Emty: „ + _BDBytes.length);
				trace(„Emty: „ + _SDBytes.length);
				trace(„Emty: „ + _HHBytes.length);
				trace(„thisBA: „ + thisByteArray.length);
				var ev:Event = new Event(„extracted“);

				dispatchEvent(ev);
			}

			private function cloneArray(array:Array):Array
			{
				var returnArray:Array = new Array(array.length);

				for(var i:int =0; i<array.length; i++)
				{
					var thisSample:AudioSample = array[i] as AudioSample;
					if(i==15) thisSample.clearSound();
					returnArray[i] = thisSample.clone();
				}
				return returnArray;
			}
		}
	}
```

2.2.7 SamplePlayer

In der Applikation könnten mehrere SamplePlayer-erzeugt werden, benötigt werden laut derzeitigem Stand SamplePlayer. Das Grundgerüst ist ein 16-stelliges Array, das die Samples speichert. In dem SamplePlayer wird ein leerer Sound auf Auslösen des Play-Buttons abgespielt, der das SampleData-Event auslöst. Im SampleData-Event wird dieser leere Sound ständig vom 16-stelligen Array aus befüllt.

SamplePlayer.as

```
package Tools
{
		import flash.display.Sprite;
		import flash.events.SampleDataEvent;
		import flash.media.Sound;
		import flash.media.SoundChannel;
		import flash.media.SoundTransform;

		/**
		 * @author Martin Engler
		 */

		public class SamplePlayer extends Sprite
		{
				static public const BLOCK_SIZE: int = 8192;

				private var _volume:Number = 1;
				private var _sound: Sound; //sound, der dynamisch gefüttert wird
				private var _samples:Array; //AudioSample Array
				private var _counter:int;
				private var _playTrue:Boolean;
				private var _debugcount:int=0;
				private var _soundChannel:SoundChannel;
				private var _loader:SampleLoader;
				private var _samplesInBytes:Array;

				// gibt den counter zurück
				public function get counter():int
				{
						return _counter;
				}

				//gibt das samles array zurück
				public function get samples():Array
				{
						var returnArray:Array = cloneArray(_samples);
						return returnArray;
				}

				public function SamplePlayer(sample:AudioSample)
				{
						_counter = 0;
						trace(„new player“);
						_samplesInBytes = new Array(16);
						_samples = new Array(16);
						var sampleArray:Array = new Array(16);
						for(var i:int =0; i<16; i++)
						{
								_samples[i] = sample;
								sampleArray[i]=sample;
						}
						_soundChannel = new SoundChannel();
						_sound=new Sound();
						_playTrue = new Boolean(true);
						changeSamples(sampleArray);
				}

				// alle 16 samples austauschen
				public function changeSamples(samples:Array):void
				{
						_samples = cloneArray(samples);
						for(var i:int =0;i<16;i++)
						{
								var thisSample:AudioSample;
								thisSample = _samples[i] as AudioSample;
								_samplesInBytes[i] = thisSample._actualSample;
						}
				}

				// ein sample an bestimmter position löschen
				public function changeSampleAt(pos:int, sample:AudioSample):void
				{
						if(pos>=0 && pos<=15)
						{
								_samples[pos] = sample;
								_samplesInBytes[pos] = sample._actualSample;
						} else {
								trace(„error: pos nicht innerhalb der grenzen [pos: „ + pos + „]“);
						}

				}

				// alle 16 stellen auf mit übergebenem sample füllen
				public function resetExtraFuerDidi(sample:AudioSample):void
				{
```

```
                        var sampleArray:Array = new Array(16);
                        for(var i:int =0; i<16; i++)
                        {
                                sampleArray[i]=sample.clone();
                        }
                        changeSamples(sampleArray);
                }

                // sound wird abgespielt
                public function play():void
                {
                        _soundChannel.soundTransform = new SoundTransform(_volume);
                        _playTrue = true;
                        _sound.addEventListener( SampleDataEvent.SAMPLE_DATA, sampleData );
                        _sound.play(); // der „leere" sound läuft ständig durch
                }

                //sound wird gestoppt
                public function stop():void
                {
                        _playTrue = false;
                        _counter = 0;
                        _sound.removeEventListener( SampleDataEvent.SAMPLE_DATA, sampleData );
                }

                //sampledataevent, wird aufgerufen wenn _sound nach neuen samples verlangt
                private function sampleData( event: SampleDataEvent ): void
                {
                        trace(„sampledata: „ + _counter);
                        event.data.writeBytes(_samplesInBytes[_counter]); // _sound mit aktuellem sample füttern

                        _counter++;
                        if (_counter==16) _counter = 0;
                }

                //clont ein array, das mit audiosamples gefüllt ist
                private function cloneArray(array:Array):Array
                {
                        var returnArray:Array = new Array(array.length);
                        for(var i:int =0; i<array.length; i++)
                        {
                                var thisSample:AudioSample = array[i] as AudioSample;
                                returnArray[i] = thisSample.clone();
                        }
                        return returnArray;
                }

                }
        }
```

2.2.8 XML

XML beschreibt Oberfläche, die das Interface verwaltet und von der, je nach Interaktion des Users aus (Drag&Drop, Klicks, etc.), die diversen Methoden aufgerufen werden. Die XML-Klasse ist somit das Kernstück der Applikation „PicassoOrchestra 2.0"

PicassoOrchestra.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
        layout="absolute"
        applicationComplete="init()"
        borderStyle="none"
        backgroundColor="#F2F2F2"
        backgroundGradientAlphas="[1.0, 1.0]"
        backgroundGradientColors="[#CCC5C5, #E5DFDF]"
        cornerRadius="20" width="1071"
        height="808" verticalAlign="middle">
          <mx:Script>

    <![CDATA[
        import mx.events.SliderEvent;
        import Tools.Drumset;
        import Tools.Bellset;
        import Tools.SampleLoader;
        import Tools.SamplePlayer;
        import Tools.ExportImage;
        import Tools.AudioSample;
        import com.pixelbreaker.ui.osx.MacMouseWheel;

      //Import classes so you don't have to use full names.
      import mx.managers.DragManager;
      import mx.core.DragSource;
      import mx.events.DragEvent;
      import mx.rpc.events.ResultEvent;
      import flash.events.MouseEvent;
      import flash.display.Loader;
      import flash.display.Bitmap;
                        import flash.display.BitmapData;
                        import mx.controls.Image;
                        import mx.graphics.codec.JPEGEncoder;
```

```
                    import flash.utils.ByteArray;

                    //Damit Object serialisiert werden kann
                    registerClassAlias(„Drumset“,Drumset);
                    registerClassAlias(„Bellset“,Bellset);
                    registerClassAlias(„AudioSample“,AudioSample);
                    // CLONE OBJECT BYTEARRAY basis
                    ///////////////////////////

                    //VARIABLES:
                    private var _loader:SampleLoader;
                    private var _player:SamplePlayer;
                    private var _playerBell:SamplePlayer;

                    private var percussionArray:Array;
                    private var _kickAudio:Array;
                    private var _snareAudio:Array;
                    private var _hihatAudio:Array;

                    private var test:int = 0;
                    private var countervar:int;
                    private var bellIcon:Bellset;
                    //private var synthStringIcon:Bellset;
                    private var countbildanzeige:int = 0;

                    //VARIABLES FOR SAVED IMAGES

                    private var percussionArray_img1:Array;
                    private var _kickAudio_img1:Array;
                    private var _snareAudio_img1:Array;
                    private var _hihatAudio_img1:Array;
                    private var _samples_img1:Array;
                    private var _samplesBell_img1:Array;
                    private var bellIcon_img1:Bellset = new Bellset(v1);

                    private var percussionArray_img2:Array;
                    private var _kickAudio_img2:Array;
                    private var _snareAudio_img2:Array;
                    private var _hihatAudio_img2:Array;
                    private var _samples_img2:Array;
                    private var _samplesBell_img2:Array;
                    private var bellIcon_img2:Bellset = new Bellset(v1);

                    private var percussionArray_img3:Array;
                    private var _kickAudio_img3:Array;
                    private var _snareAudio_img3:Array;
                    private var _hihatAudio_img3:Array;
                    private var _samples_img3:Array;
                    private var _samplesBell_img3:Array;
                    private var bellIcon_img3:Bellset = new Bellset(v1);
                    ///////////////////////////
public function init():void{
    this.loadetr
    _kickAudio = new Array(16);
                    _snareAudio = new Array(16);
                    _hihatAudio = new Array(16);
                    percussionArray = new Array();

    _loader  = new SampleLoader();
    _loader.addEventListener(„extracted“, forward);
    _loader.extract();

    bellIcon = new Bellset(v1);
    textfield.text = „Application started“;
    //var AllDrums:Drumset;
    trace(this.numChildren);
    createKickIcon();
    createSnareIcon();
    createHiHatIcon();
    createBellIcon();
                    v1.addEventListener(MouseEvent.CLICK, deselectall);
                    MacMouseWheel.setup( this.stage );
                    stage.addEventListener(KeyboardEvent.KEY_DOWN, iconKeyEvent);
                    volumeslider.value = 3;
                    volumeslider.alpha = 0;
  pitchslider.alpha = 0;

}

public function forward(event:Event):void{
                    _player = new SamplePlayer(_loader.getEmpty());
                    _playerBell = new SamplePlayer(_loader.getEmpty());
            }

            public function imageAbspeichern():void{

                    if(countbildanzeige == 0){

                                    _kickAudio_img1 = new Array(16);
                                    _snareAudio_img1 = new Array(16);
                                    _hihatAudio_img1 = new Array(16);
                                    percussionArray_img1 = new Array();
                                    _samples_img1 = new Array();
                                    _samplesBell_img1 = new Array();
                                    bellIcon.clone(bellIcon_img1);

                                    _kickAudio_img1 =  cloneAudioArray(_kickAudio);
                                    _snareAudio_img1 = cloneAudioArray(_snareAudio);
                                    _hihatAudio_img1 = cloneAudioArray(_hihatAudio);
                                    percussionArray_img1 = cloneArray(percussionArray);
                                    _samples_img1 = cloneAudioArray(_player.samples);
                                    _samplesBell_img1 = cloneAudioArray(_playerBell.samples);
                                    //_samples_img1 = cloneArray(_player.samples);
                                    imagePLatzhalter.source = ExportImage.saveImage(createObjectArrayforScreenshot());
                                    countbildanzeige++;

                    }else if(countbildanzeige == 1){
```

```
                                                _kickAudio_img2 = new Array(16);
                                                _snareAudio_img2 = new Array(16);
                                                _hihatAudio_img2 = new Array(16);
                                                percussionArray_img2 = new Array();
                                                _samples_img2 = new Array();
                                                _samplesBell_img2 = new Array();

                                        imagePLatzhalter0.source = ExportImage.saveImage(createObjectArrayforScreenshot());
                                                bellIcon.clone(bellIcon_img2);
                                                _kickAudio_img2 =  cloneAudioArray(_kickAudio);
                                                _snareAudio_img2 = cloneAudioArray(_snareAudio);
                                                _hihatAudio_img2 = cloneAudioArray(_hihatAudio);
                                                _samples_img2 = cloneAudioArray(_player.samples);
                                                _samplesBell_img2 = cloneAudioArray(_playerBell.samples);
                                                percussionArray_img2 = cloneArray(percussionArray);
                                                //_samples_img2 = cloneArray(_player.samples);
                                                countbildanzeige++;

                                }else if(countbildanzeige == 2){
                                        textfield.text = textfield.text + „\n Bild Position 3 speichere“;
                                                _kickAudio_img3 = new Array(16);
                                                _snareAudio_img3 = new Array(16);
                                                _hihatAudio_img3 = new Array(16);
                                                percussionArray_img3 = new Array();
                                                _samples_img3 = new Array();
                                                _samplesBell_img3 = new Array();
                                                bellIcon.clone(bellIcon_img3);
                                        imagePLatzhalter1.source = ExportImage.saveImage(createObjectArrayforScreenshot());

                                                _kickAudio_img3 =  cloneAudioArray(_kickAudio);
                                                _snareAudio_img3 = cloneAudioArray(_snareAudio);
                                                _hihatAudio_img3 = cloneAudioArray(_hihatAudio);
                                                _samples_img3 = cloneAudioArray(_player.samples);
                                                _samples_img3 = cloneAudioArray(_playerBell.samples);
                                                percussionArray_img3 = cloneArray(percussionArray);
                                                //_samples_img3 = cloneArray(_player.samples);
                                                countbildanzeige = 0;
                                }
                        }

                        public function imageExport():void{
                                ExportImage.createImage(createObjectArrayforScreenshot());
                        }

                        public function createObjectArrayforScreenshot():Array{
                                var ScreenshotArray:Array = new Array();

                                //Stage reinschreiben
                                ScreenshotArray.push(this.v1);
                                //

                                //Percussions reinschreiben
                for(var i:int=0; i<percussionArray.length; i++){
                        if(percussionArray[i] != null){
                                 if(!(percussionArray[i].x > toolbox.x && percussionArray[i].x < (toolbox.x+toolbox.width))){ // ABFRAGE OB ICON NICHT IN TOOL-
BOX
                                                ScreenshotArray.push(percussionArray[i]);
                                 }
                        }
                }

        ///////////////////////////////7
        //BellIcon reinschreiben
         if(!(bellIcon.x > toolbox.x && bellIcon.x < (toolbox.x+toolbox.width))){ // ABFRAGE OB ICON NICHT IN TOOLBOX
                        ScreenshotArray.push(bellIcon);
         }
        ////////////////////////////////

                                return ScreenshotArray;
                        }

        public function onEnterFrame(event:Event):void{
            countervar = ((_player.counter)+11)%16;

                for(var i:int=0; i<percussionArray.length; i++){
                        //textfield.text = textfield.text.toString() + „\n icon:“+(percussionArray[i] as Drumset).xwertmitteSnap;
                                if(percussionArray[i]!= null){
                                                        if(percussionArray[i].width > 50 && (percussionArray[i] as Drumset).xwertmitteSnap != counter-
var ){
                                                                //percussionArray[i].scaleX=1;
                                                                //percussionArray[i].scaleY=1;
                                                                percussionArray[i].width --;
                                                                percussionArray[i].height --;
                                                                //textfield.text = textfield.text.toString() + „\MINUS ICON!“;
                                                                }else if( percussionArray[i].x > toolbox.x+toolbox.width && (percussionArray[i] as
Drumset).xwertmitteSnap == countervar  ){
                                                        percussionArray[i].width ++;
                                                        percussionArray[i].height ++;
                                                        //percussionArray[i].scaleX=2;
                                                        //percussionArray[i].scaleY=2;
                                                        //textfield.text = textfield.text.toString() + „\nPLUS ICON!“;
                                                }
                }
                }
        }

         public function createBellIcon():void{
              bellIcon.source = bellIconC;
              bellIcon.x =  toolbox.x + 120;
              bellIcon.y = toolbox.y + 65;
              bellIcon.width = 50;
              bellIcon.height = 50;
              bellIcon.name = „BellIcon“;
               bellIcon.toolTip = „BellIcon“;
                                        bellIcon.addEventListener(MouseEvent.MOUSE_MOVE,mouseOverHandler);
                                        bellIcon.addEventListener(MouseEvent.CLICK, clickedIcon);
                                        bellIcon.alpha =1;
              this.addChild(bellIcon);
         }

         public function createKickIcon():void{
                        var newImage:Drumset= new Drumset();
              newImage.source = kickIcon;
              newImage.x = toolbox.x + 30;
              newImage.y = toolbox.y + 65;
              newImage.width = 50;
              newImage.height = 50;
              newImage.name = „KickIcon";
               newImage.toolTip = „Kickdrum";
                                          newImage.addEventListener(MouseEvent.MOUSE_MOVE,mouseOverHandler);
                                  newImage.addEventListener(MouseEvent.CLICK, clickedIcon);
                                        newImage.addEventListener(MouseEvent.MOUSE_WHEEL, scaleIcon);
              this.addChild(newImage);
               percussionArray.push(newImage);
         }

         public function createSnareIcon():void{
                        var newImage:Drumset= new Drumset();
              newImage.source = snareIcon;
              newImage.x = toolbox.x + 30;
              newImage.y = toolbox.y + 135;
              newImage.width = 50;
              newImage.height = 50;
              newImage.name = „SnareIcon";
               newImage.toolTip = „Snare";
                                        newImage.addEventListener(MouseEvent.MOUSE_MOVE,mouseOverHandler);
                                        newImage.addEventListener(MouseEvent.CLICK, clickedIcon);
                                newImage.addEventListener(MouseEvent.MOUSE_WHEEL, scaleIcon);
              this.addChild(newImage);
               percussionArray.push(newImage);
         }
         public function createHiHatIcon():void{
                        var newImage:Drumset= new Drumset();
              newImage.source = hiIcon;
              newImage.x = toolbox.x + 30;
              newImage.y = toolbox.y + 205;
              newImage.width = 50;
              newImage.height = 50;
              newImage.name = „HiHatIcon";
               newImage.toolTip = „HiHat";
                                newImage.addEventListener(MouseEvent.MOUSE_MOVE,mouseOverHandler);
                                newImage.addEventListener(MouseEvent.CLICK, clickedIcon);
                                newImage.addEventListener(MouseEvent.MOUSE_WHEEL, scaleIcon);
              this.addChild(newImage);
               percussionArray.push(newImage);
         }

         // Embed icon images;
         [Embed(source='../pics/Bell.png')]
         public var bellIconC:Class;
         [Embed(source='../pics/Snare.png')]
         public var snareIcon:Class;
         [Embed(source='../pics/Bass.png')]
         public var kickIcon:Class;
             [Embed(source='../pics/HiHat.png')]
         public var hiIcon:Class;
                         ////////////////////////////////////////////
         private function iconKeyEvent(event:KeyboardEvent):void // DELETES
                                {
                                          if(event.keyCode==46 || event.keyCode==8  )
                                          {
                                                          for(var i:int=0; i<percussionArray.length; i++){
                                        if(percussionArray[i] != null){
                                                                            if (percussionArray[i].alpha == 0.5){
                                                                                    var tempIconname:String = percussionArray[i].name;
                                                                                    this.removeChild(percussionArray[i]);
                                                                                    //Position im Array ausrechnen
                                                                                    var posXnew:int = ((percussionArray[i].x -
v1.x)+percussionArray[i].width/2);  // X Position ausrechnen
                                                        var arraypos:int = posXnew/50; //Durch die 50 damit es max 16 werte gibt
                                                        var whatwasname:uint = 0;
                                                        if(percussionArray[i].name == „KickIcon"){
                                                                _kickAudio[arraypos] = null;
                                                                whatwasname = 1;
                                                        }
                                                        if(percussionArray[i].name == „SnareIcon"){
                                                                _snareAudio[arraypos] = null;
                                                                whatwasname = 2;
                                                        }
                                                        if(percussionArray[i].name == „HiHatIcon"){
                                                                _hihatAudio[arraypos] = null;
                                                                whatwasname = 3;
                                                        }
                                                                                    //textfield.text = textfield.text.toString() + „\nDELETED:" +
_player._samples[arraypos];
                                                                                    _player.changeSampleAt(arraypos,checkPositionForOtherIcons
addDelete(arraypos,whatwasname));
                                                                                    percussionArray[i] = null;

                                                                            }
                                        }
                                }
                                if(bellIcon.alpha == 0.5){
                                        if(playButton.styleName == „playButton"){
                                                this.removeChild(bellIcon);
                                                bellIcon = new Bellset(v1);
                                                                                    _playerBell = new SamplePlayer(_loader.getEmpty());
                                                createBellIcon();
                                        }else{
                                                //startStopPlayer(false);
                                                //playButton.label = „play";
                                                //playButton.removeEventListener(Event.ENTER_FRAME,onEnterFrame);
                                                this.removeChild(bellIcon);
                                                 for(var j:int=0; j<16; j++){
                                                 _playerBell.changeSampleAt(j,_loader.getEmpty());
                                                 }
```

```
                                                        createBellIcon();
                                                       //startStopPlayer(true);
                                                       //playButton.label = „stop";
                                                       //playButton.addEventListener(Event.ENTER_FRAME,onEnterFrame);

                                              }
                                    }
                                                 }
                                    }
                                    private function checkPositionForOtherIconsaddDelete(arraypos:int,whatwasname:int):AudioSample{
                  var maybeothersounds:AudioSample;
                  if (whatwasname == 0){
                           textfield.text = textfield.text.toString() + „\ERROR 401:";
                  }else if(whatwasname == 1){
                           if(_snareAudio[arraypos] != null){
                                                                      maybeothersounds = _snareAudio[arraypos];
                           }
                           else if(_hihatAudio[arraypos] != null){
                                                                      maybeothersounds = _hihatAudio[arraypos];
                           }
                                                            else if(_snareAudio[arraypos] != null && _hihatAudio[arraypos] != null){
                                                                      maybeothersounds = _snareAudio[arraypos].merge(_hihatAudio[arraypos]);
                           }else{
                                    maybeothersounds = _loader.getEmpty();
                           }
                  }else if(whatwasname == 2){
                           if(_kickAudio[arraypos] != null){
                                                                      maybeothersounds = _kickAudio[arraypos];
                           }
                           else if(_hihatAudio[arraypos] != null){
                                                                      maybeothersounds = _hihatAudio[arraypos];
                           }
                                                            else if(_kickAudio[arraypos] != null && _hihatAudio[arraypos] != null){
                                                                      maybeothersounds = _kickAudio[arraypos].merge(_hihatAudio[arraypos]);
                           }else{
                                    maybeothersounds = _loader.getEmpty();
                           }
                  }else if(whatwasname == 3){
                           if(_snareAudio[arraypos] != null){
                                                                      maybeothersounds = _snareAudio[arraypos];

                           }
                           else if(_kickAudio[arraypos] != null){
                                                                      maybeothersounds = _kickAudio[arraypos];
                           }
                                                            else if(_snareAudio[arraypos] != null && _kickAudio[arraypos] != null){
                                                                      maybeothersounds = _snareAudio[arraypos].merge(_kickAudio[arraypos]);
                           }else{
                                    maybeothersounds = _loader.getEmpty();
                           }
                  }

                  return maybeothersounds;
                }


               private function deselectall(event:MouseEvent):void{ // Alle Icons zurücksetzen wenn click auf canvas

                                                    //Durchgehen und alle Selections null setzen
                 for(var i:int=0; i<percussionArray.length; i++){
                  if(percussionArray[i] != null){
                                                            percussionArray[i].alpha = 1;
                  }
                 }

                 bellIcon.alpha =1;
                    volumeslider.alpha = 0;
                     pitchslider.alpha = 0;


               }
               private function clickedIcon(event:MouseEvent):void{ // Alle Icons nullsetzen und neues Icon selektieren

                  //Durchgehen und alle Selections Drumset null setzen wenn Strg nicht gedrückt und wenn BellIcon geklickt

                  if(event.ctrlKey != true){
                          for(var i:int=0; i<percussionArray.length; i++){
                           if(percussionArray[i] != null){
                                                                      percussionArray[i].alpha = 1;
                                                                      volumeslider.alpha = 0;
                                    pitchslider.alpha = 0;
                           }
                          }
                          bellIcon.alpha =1;
                        }

                 ///////////////////////////////////////


                   if(!(event.currentTarget.x > toolbox.x && event.currentTarget.x < (toolbox.x+toolbox.width))){ // ABFRAGE OB ICON NICHT IN TOOLBOX

                     event.currentTarget.alpha = 0.5;

                     if(event.currentTarget is Drumset){
                     volumeslider.value = event.currentTarget.getLoudness();
                     pitchslider.value = event.currentTarget.calcPitchwithPos(((event.currentTarget.y - v1.y)+event.currentTarget.height/2));
                     //SLIDER Visible setzen
                     volumeslider.alpha = 1;
                     pitchslider.alpha = 1;

                     }
                     }
```

```
        }
                private function scaleIcon(event:MouseEvent):void{
                                                //Derzeitigen State herausfinden

                  if(!(event.currentTarget.x > toolbox.x && event.currentTarget.x < (toolbox.x+toolbox.width))){ // ABFRAGE OB ICON NICHT IN TOOLBOX
                                                        //textfield.text = textfield.text.toString() + „\nLoudnesscur:“ + (event.currentTarget as Drumset).getLoud-
ness();

                                                        if (event.delta < 0){
                                                                if(event.currentTarget.getLoudness() > 0 ){
                                                                        event.currentTarget.setLoudnessMINUS();
                                                                }
                                                        }else{
                                                                if(event.currentTarget.getLoudness() < 1 ){
                                                                        event.currentTarget.setLoudnessPLUS();
                                                                }
                                                        }

                                                        event.currentTarget.getLoudIcon();
                                                        volumeslider.value = event.currentTarget.getLoudness();
                                                        prepareSample((event.currentTarget as Drumset),event.currentTarget.x,true);


        }
        }

        // The dragEnter event handler for the Canvas container
        // enables dropping.
        private function dragEnterHandler(event:DragEvent):void {
           if (event.dragSource.hasFormat(„img“))
           {
              DragManager.acceptDragDrop(v1);
           }
        }

        // The dragDrop event handler for the Canvas container
        // sets the Image control‘s position by
        // „dropping“ it in its new location.
        private function dragDropHandler(event:DragEvent):void {

             var oldX:uint = event.dragInitiator.x;              //ursprungsXdamit nur copy wenn von Toolbox in Canvas
             var oldY:uint = event.dragInitiator.y;              //ursprungsXdamit nur copy wenn von Toolbox in Canvas
             var newcreated:Boolean = false;
             var copyornot:Boolean = true;

             ////////////////////////////////////////////////
             // NUR DRUMSET////////////////////////////////////
             ////////////////////////////////////////////////
             ////////////////////////////////////////////////
             if(event.dragInitiator is Drumset){
             //Derzeitige Mausposition auf v1 + die Position des Canvas - Halber größe des Icons +snap auf 16ntel
             var snapPosX:uint = ((v1.mouseX/50)+1); // derzeitige position wird gesechzentelt
             snapPosX = (snapPosX*50)-25; // und bei jedem sechzenhntel gemittelt
             snapPosX = (snapPosX+v1.x)-(event.dragInitiator.width/2);

             //DURCHSCHAUN OB AUF DIESER X POSITION BEREITS EIN GLEICHES ICON LIEGT WENN JA NICHT DRAGGEN
             for(var i:int=0; i<percussionArray.length; i++){
                         if(percussionArray[i] != null){
                                                                if(percussionArray[i].x == snapPosX && event.dragInitiator.name ==
percussionArray[i].name && percussionArray[i].x != oldX  ){
                                                                        copyornot = false;
                                                }
                                        }
                  }
             //////////////////////



             if(copyornot){

             textfield.text = textfield.text.toString() + „\n testxpos:“+  snapPosX;
             Image(event.dragInitiator).x = snapPosX;
           // Image(event.dragInitiator).x = ((v1.mouseX+v1.x)-(event.dragInitiator.width/2));
            Image(event.dragInitiator).y = (v1.mouseY+v1.y)-(event.dragInitiator.height/2);
            textfield.text = textfield.text.toString() + „\n MOVE:  Tool moved!“;

             //Calculate 16ten Position
             if(event.dragInitiator is Drumset){
             var posXnew:int = ((event.dragInitiator.x - v1.x)+event.dragInitiator.width/2);
              (event.dragInitiator as Drumset).xwertmitteSnap = (posXnew/50);
             }
            ////


           //ABGESCHALTENE FUNKTIONALITÄT DOPPELT VERSCHIEBEN
           /*
                                                //Auch alle anderen verschieben wenn makiert nur den initiator ncht(der wir oben verschoben) und nicht wenn von
Toolbox auf canvas
            if(!(oldX < toolbox.x+toolbox.width)){
                                for(var j:int=0; j<percussionArray.length; j++){
                                 if(percussionArray[j] != null && percussionArray[j] != event.dragInitiator){
                                                                if (percussionArray[j].alpha == 0.5 ){          //schauen ob makiert und varible
verschiebungOk ok ob nichts drausen is
                                                                                var tempX:uint = percussionArray[j].x;          //Speichern
falls auerhalb seinen sollte
                                                                var tempY:uint = percussionArray[j].y;//Speichern falls auerhalb seinen sollte
                                                                                percussionArray[j].x += ((v1.mouseX+v1.x)-
(percussionArray[j].width/2)) - oldX;           //alle verschieben
                                                                                percussionArray[j].y += ((v1.mouseY+v1.y)-
(percussionArray[j].height/2))- oldY;

                                                                                        if( percussionArray[j].x < v1.x ||
percussionArray[j].x > v1.x+v1.width ||         //falls außerhalb von canvas
```

```
                                                                                                        percussionArray[j].y < v1.y ||
percussionArray[j].y > v1.y+v1.height){
                                                                                                        percussionArray[j].x = tempX;
                                                                                                        percussionArray[j].y = tempY;
                                                                                                        textfield.text = textfield.text.to-
String() + „\n ERROR: Out of Bounds";
                                                                                                }
                                        }
                                        }
                                    }
        }
        */

        // NUR Neues ICON wenn von Toolbox in Canvas (die -25 wegen)
        if(oldX < (toolbox.x+toolbox.width)-event.dragInitiator.width/2){

                                        if(event.dragInitiator.name == „KickIcon"){
                                         createKickIcon();
                                        }else if(event.dragInitiator.name == „SnareIcon"){
                                         createSnareIcon();
                                        }else if(event.dragInitiator.name == „HiHatIcon"){
                                        createHiHatIcon();
                                        }else{
                                         textfield.text = textfield.text.toString() + „\n Icon unbekannt";
                                        }
                                    newcreated = true;
         textfield.text = textfield.text.toString() + „\n CREATE: New Tool created!";

        }
        ////////////////////////////////////////////

                                                //Dummyobject um Lautstärke herauszufinden
         //textfield.text = textfield.text.toString() + „\n Loudness:"+ dummy.getLoudness();
         var dummy:Drumset;
                        dummy = event.dragInitiator as Drumset;
                                                prepareSample(dummy,oldX,newcreated);
                                                }
         }
         ////////////////////////////////////////////
         ////////////////////////////////////////////
         ////////////////////////////////////////////
         // NUR BELL
         ////////////////////////////////////////////
         ////////////////////////////////////////////
         if(event.dragInitiator is Bellset){
         Image(event.dragInitiator).x = ((v1.mouseX+v1.x)-(event.dragInitiator.width/2));
         Image(event.dragInitiator).y = ((v1.mouseY+v1.y)-(event.dragInitiator.height/2));
         prepareBellSample();
         }
    }
    private function prepareBellSample():void{
         textfield.text = textfield.text.toString() + „\n quad"+ bellIcon.calcQuadrant();
         textfield.text = textfield.text.toString() + „\n Pos"+ bellIcon.calcStartPosSech(v1.x);
          _playerBell.changeSamples(_loader.getBellsArray(bellIcon.calcStartPosSech(v1.x),bellIcon.calcQuadrant(),bellIcon.calcVolume()));
    }

    private function prepareSample(dummy:Drumset,oldX:int,newcreated:Boolean):void{ //Samples werden in den einzelnen Arrays vorbereitet

         textfield.text = textfield.text.toString() + „\n prepareSAMPLE";

        var posXnew:int = ((dummy.x - v1.x)+dummy.width/2);  // X Position ausrechnen
        var seeifXchange:int = (posXnew +v1.x)-dummy.width/2;
        var arraypos:int = posXnew/50; //Durch die 50 damit es max 16 werte gibt
        var posYnew:int = ((dummy.y - v1.y)+dummy.height/2); // YPosition ausrechnen
        var pitcha:Number = dummy.calcPitchwithPos(posYnew);
        //textfield.text = textfield.text.toString() + pitcha;
        var posXold:int = ((oldX-v1.x)+dummy.width/2)/50;
        var mergedsample:AudioSample = _loader.getEmpty(); //Das AudioSample an der neuen STelle
        var mergedsampleOld:AudioSample = _loader.getEmpty(); // Das Audio Sample an der alten STelle

        if(dummy.name == „KickIcon"){
         _kickAudio[arraypos] = _loader.getBD();
         _kickAudio[arraypos].pitch(pitcha);
         textfield.text = textfield.text.toString() + „\n Volume:" + dummy.getLoudness();

         _kickAudio[arraypos].volume(dummy.getLoudness());

         mergedsample = _kickAudio[arraypos];
         if(_snareAudio[arraypos] != null){mergedsample = mergedsample.merge(_snareAudio[arraypos]);}
         if(_hihatAudio[arraypos] != null){mergedsample = mergedsample.merge(_hihatAudio[arraypos]);}

         // Wenn nur im Pitch verschoben nicht null setzen
         //textfield.text = textfield.text.toString() + „\n OLDX:" + seeifXchange;
         //textfield.text = textfield.text.toString() + „\n NEWX:" + oldX ;
         //
                    if (newcreated==false && seeifXchange != oldX){
                                                        _kickAudio[posXold] = null;
                                                        }
        }
        else if(dummy.name == „SnareIcon"){
         _snareAudio[arraypos] = _loader.getSD();
         _snareAudio[arraypos].pitch(pitcha);
         _snareAudio[arraypos].volume(dummy.getLoudness());

         mergedsample = _snareAudio[arraypos];
         if(_kickAudio[arraypos] != null){mergedsample = mergedsample.merge(_kickAudio[arraypos]);}
         if(_hihatAudio[arraypos] != null){mergedsample = mergedsample.merge(_hihatAudio[arraypos]);}

                    if (newcreated==false && seeifXchange != oldX ){
                                                        _snareAudio[posXold] = null;
                                                        }
        }
        else if(dummy.name == „HiHatIcon"){
         _hihatAudio[arraypos] = _loader.getHH();
         _hihatAudio[arraypos].pitch(pitcha);
         _hihatAudio[arraypos].volume(dummy.getLoudness());
    mergedsample = _hihatAudio[arraypos];
    if(_snareAudio[arraypos] != null){mergedsample = mergedsample.merge(_snareAudio[arraypos]);}
    if(_kickAudio[arraypos] != null){mergedsample = mergedsample.merge(_kickAudio[arraypos]);}

            if (newcreated==false && seeifXchange != oldX ){
                                                    _hihatAudio[posXold] = null;
                                                    }
   }else{
    textfield.text = textfield.text.toString() + „\n Icon unbekannt";
   }
  //_player._samples[posXold] = new AudioSample(_loader.getEmpty()); //Wenn verschoben wird alte Position löschen
  _player.changeSampleAt(arraypos,mergedsample); //Neues FIle wird geschrieben
  if(newcreated == false && seeifXchange != oldX){
  //Hier wird wenn weggedragt wurde wieder der Sound ohne das weggedraggte File berechnet
   if(dummy.name == „KickIcon"){
                if(_snareAudio[posXold] != null){
                            mergedsampleOld = _snareAudio[posXold];
                            if(_hihatAudio[posXold] != null){
                                        mergedsampleOld = mergedsampleOld.merge(_hihatAudio[posXold]);
                            }
                }
                if(_hihatAudio[posXold] != null){
                            mergedsampleOld = _hihatAudio[posXold];
                }
   }else if(dummy.name == „SnareIcon"){
                if(_kickAudio[posXold] != null){
                            mergedsampleOld = _kickAudio[posXold];
                            if(_hihatAudio[posXold] != null){
                                        mergedsampleOld = mergedsampleOld.merge(_hihatAudio[posXold]);
                            }
                }
                if(_hihatAudio[posXold] != null){
                            mergedsampleOld = _hihatAudio[posXold];
                }
   }else if(dummy.name == „HiHatIcon"){
                if(_kickAudio[posXold] != null){
                            mergedsampleOld = _kickAudio[posXold];
                            if(_snareAudio[posXold] != null){
                                        mergedsampleOld = mergedsampleOld.merge(_snareAudio[posXold]);
                            }
                }
                if(_snareAudio[posXold] != null){
                            mergedsampleOld = _snareAudio[posXold];
                }
   }
  _player.changeSampleAt(posXold,mergedsampleOld);
  }
}

// The mouseMove event handler for the Image control
// initiates the drag-and-drop operation.
private function mouseOverHandler(event:MouseEvent):void
{
   var dragInitiator:Image=Image(event.currentTarget);
   var ds:DragSource = new DragSource();
   ds.addData(dragInitiator, „img");
   // The drag manager uses the Image control
   // as the drag proxy and sets the alpha to 1.0 (opaque),
   // so it appears to be dragged across the Canvas.
    var imageProxy:Image = new Image();

    if(event.currentTarget.name == „KickIcon"){
     imageProxy.source = kickIcon;
    }
    else if(event.currentTarget.name == „SnareIcon"){
     imageProxy.source = snareIcon;
    }
   else if(event.currentTarget.name == „HiHatIcon"){
     imageProxy.source = hiIcon;
   }
   else if(event.currentTarget.name == „BellIcon"){
     imageProxy.source = bellIconC;
   }else{
    textfield.text = textfield.text.toString() + „\n Icon unbekannt";
   }

   imageProxy.height=30;
   imageProxy.width=30;
   DragManager.doDrag(dragInitiator, ds, event, imageProxy, -5, -5, 1.00);
}

//Startet und Stopt die Player // True startet; //False stoppt
public function startStopPlayer(state:Boolean):void{
    if(state == true){
                _playerBell.play();
                                                    _player.play();
                                                    textfield.text = textfield.text.toString() + „\n Start ALL PLAYERS";
    }else{
                _playerBell.stop();
                                                    _player.stop();
                                                    textfield.text = textfield.text.toString() + „\n STOP ALL PLAYERS";
    }
}

                // startet und stoppt die musik
                public function playButtonHandler():void
                {
                            if(playButton.styleName == „playButton"){

                                        //SoundMixer.soundTransform = new SoundTransform(1); MASTER VOLUME
                                        startStopPlayer(true);
                                        playButton.styleName = „PauseButton";
                                        playButton.addEventListener(Event.ENTER_FRAME,onEnterFrame);

                            } else {
                                        startStopPlayer(false);
                                        playButton.styleName = „playButton";
                                        playButton.removeEventListener(Event.ENTER_FRAME,onEnterFrame);
```

```
                        }
                }

                // startet und stoppt die musik
                public function clearButtonHandler():void
                {
                        for(var i:int=0; i<percussionArray.length; i++){
                        if(percussionArray[i] != null){
                                this.removeChild(percussionArray[i]);
                        }
                        }
                        this.removeChild(bellIcon);
                        startStopPlayer(false);
                        playButton.removeEventListener(Event.ENTER_FRAME,onEnterFrame);
                        playButton.styleName = „playButton“;
                        //forward(new Event(Event.COMPLETE)); // SOLANGE FORWARD MIT TIMER HIER NOCH MAL AUFRUFEN
                        init();

                }

                public function resetIcons():void
                {
                        for(var i:int=0; i<percussionArray.length; i++){
                                if(percussionArray[i] != null){
                                        this.removeChild(percussionArray[i]);
                                }
                        }
                        this.removeChild(bellIcon);
                        startStopPlayer(false);
                        playButton.removeEventListener(Event.ENTER_FRAME,onEnterFrame);
                        playButton.styleName = „playButton“;
                }

    public function changeSliderHandler(event:SliderEvent):void{ // Slider für Lautstärke für Extrawurst MAC
                  for(var i:int=0; i<percussionArray.length; i++){
                if(percussionArray[i] != null){
                                                if(percussionArray[i].alpha == 0.5){
                                                        percussionArray[i].setLoudness(event.value);
                                                        percussionArray[i].getLoudIcon();
                                                        prepareSample(percussionArray[i],percussionArray[i].x,true);
                                                }
                }
                }
    }
     public function changeSliderHandlerPitch(event:SliderEvent):void{ // Slider für Pitch
                  for(var i:int=0; i<percussionArray.length; i++){
                if(percussionArray[i] != null){
                                                if(percussionArray[i].alpha == 0.5){
                                                        percussionArray[i].y = ((((percussionArray[i].calcPoswithPitch(event.
value))+v1.y))-(percussionArray[i].height/2));

                                                        prepareSample(percussionArray[i],percussionArray[i].x,true);
                                                }

                }
                }
    }

    private function doSaveServer():void
    {
                var jpg:JPEGEncoder = new JPEGEncoder(); // Jpeg decoder erzeugen
      var ba:ByteArray = jpg.encode(ExportImage.createImageBA(createObjectArrayforScreenshot())); // Byte Array erzeugen zum lokalen abspeichern
                        sendByteArray(ba);

    }

    private function sendByteArray(myByteArray:ByteArray):void
    {
        textfield.text = „ba: „ +  myByteArray.length;
        var request:URLRequest = new URLRequest(‚upload.php‘);
          var loader: URLLoader = new URLLoader();
      request.contentType = ‚application/octet-stream‘;
      request.method = URLRequestMethod.POST;
      request.data = myByteArray;
      loader.addEventListener(Event.COMPLETE, phpLoaderComplete);
                loader.load(request);
        }

        private function phpLoaderComplete(e:Event):void
        {
                textfield.text  = „phploadercomplete“;
        }

        private function cloneArray(source:Array):Array{
                var myArrayClone:Array = new Array();

                        for(var i:int=0; i<source.length; i++){
                                                var byteArray:ByteArray = new ByteArray();
                                                byteArray.writeObject(source[i]);
                                                byteArray.position = 0;

                                                        if(source[i] is Drumset){
                                                                textfield.text  =textfield.text+ „Drumset“;
                                                                myArrayClone[i] = ((byteArray.readObject()) as Drumset); //da registiert
funzt es

                                                                //NUN wieder das richtige Icon zuordnen
                                                        if(myArrayClone[i].name == „KickIcon“){
                                                                myArrayClone[i].source = kickIcon;
                                                        }else if(myArrayClone[i].name == „SnareIcon“){
                                                                myArrayClone[i].source = snareIcon;
                                                        }else if(myArrayClone[i].name == „HiHatIcon“){
                                                                myArrayClone[i].source = hiIcon;
                                                        }
                                                                myArrayClone[i].addEventListener(MouseEvent.MOUSE_
MOVE,mouseOverHandler);

                                                        myArrayClone[i].addEventListener(MouseEvent.CLICK, clickedIcon);
                                                        myArrayClone[i].addEventListener(MouseEvent.MOUSE_WHEEL, scaleIcon);
                                                }

                        }
                return myArrayClone;
```

```
            }

            public function cloneAudioArray(source:Array):Array{
                    var myArrayClone:Array = new Array(16);
                    for(var i:int=0; i<source.length; i++){
                            if(source[i] is AudioSample){
                                            myArrayClone[i] = source[i].clone();
                            }
                    }
                    return myArrayClone;
            }

            private function loadImage1(event:MouseEvent):void{
                                    trace(„LOAD“);
                                    resetIcons();
                                    percussionArray = new Array();
                                    percussionArray = cloneArray(percussionArray_img1);

                                    _kickAudio  = cloneAudioArray(_kickAudio_img1);
                                            _snareAudio = cloneAudioArray(_snareAudio_img1);
                                            _hihatAudio = cloneAudioArray(_hihatAudio_img1);
                                            _player.changeSamples(cloneAudioArray(_samples_img1));
                                            _playerBell.changeSamples(cloneAudioArray(_samplesBell_img1));
                                            for(var i:int=0; i<percussionArray.length; i++){
                                                    if(percussionArray[i] != null){
                                                            this.addChild(percussionArray[i]);
                                                    }
                                            }
                                            bellIcon_img1.clone(bellIcon);
                                            this.addChild(bellIcon);
            }
        private function loadImage2(event:MouseEvent):void{
                                    trace(„LOAD1“);
                                    resetIcons();
                                    percussionArray = new Array();
                                    percussionArray = cloneArray(percussionArray_img2);
                                    _kickAudio  = cloneArray(_kickAudio_img2);
                                            _snareAudio = cloneArray(_snareAudio_img2);
                                            _hihatAudio = cloneArray(_hihatAudio_img2);
                                            _player.changeSamples(cloneAudioArray(_samples_img2));
                                            _playerBell.changeSamples(cloneAudioArray(_samplesBell_img2));
                                            //_player.changeSamples(_samples_img1);
                                            for(var i:int=0; i<percussionArray.length; i++){
                                                    this.addChild(percussionArray[i]);
                                            }
                                            bellIcon_img2.clone(bellIcon);
                                            this.addChild(bellIcon);
            }
            private function loadImage3(event:MouseEvent):void{
                                    trace(„LOAD2“);
                                    resetIcons();
                                    percussionArray = new Array();
                                    percussionArray = cloneArray(percussionArray_img3);
                                    _kickAudio  = cloneArray(_kickAudio_img3);
                                            _snareAudio = cloneArray(_snareAudio_img3);
                                            _hihatAudio = cloneArray(_hihatAudio_img3);
                                            _player.changeSamples(cloneAudioArray(_samples_img3));
                                            _playerBell.changeSamples(cloneAudioArray(_samplesBell_img3));
                                            //_player.changeSamples(_samples_img1);
                                            for(var i:int=0; i<percussionArray.length; i++){
                                                    this.addChild(percussionArray[i]);
                                            }
                                            bellIcon_img3.clone(bellIcon);
                                            this.addChild(bellIcon);
            }

               ]]>
    </mx:Script>
    <mx:Canvas width=“1050“ height=“768“ x=“10“ y=“24“>
                    <mx:Button id=“playButton“ click=“playButtonHandler()“ styleName=“playButton“ width=“28.5“ height=“26.4“ horizontalCenter=“-423“
verticalCenter=“4“/>

            <!-- The Canvas is the drag target -->
            <mx:Panel id=“v1“
               width=“800“ height=“500“
               borderStyle=“outset“
               backgroundColor=“#DDDDDD“
               dragEnter=“dragEnterHandler(event);“
               dragDrop=“dragDropHandler(event);“ borderThickness=“3“ cornerRadius=“20“ borderColor=“#2B8BCE“ horizontalCenter=“106“ verticalCen-
ter=“-25“>
               <!-- The image is the drag initiator. -->
            </mx:Panel>

            <mx:Panel backgroundColor=“#DDDDDD“ borderColor=“#605757“  id=“toolbox“ width=“180“ height=“258“ layout=“absolute“ title=“Toolbox“ cornerRa-
dius=“20“ horizontalCenter=“-392“ verticalCenter=“-146“>
            </mx:Panel>
            <mx:TextArea width=“180“ height=“197“ id=“textfield“ editable=“false“ backgroundColor=“#DDDDDD“ cornerRadius=“20“ horizontalCenter=“-392“
verticalCenter=“213“ />
            <mx:HSlider id=“pitchslider“ toolTip=“Pitch“ maximum=“1.5“ minimum=“0.5“ styleName=“volumeSlider“ change=“changeSliderHandlerPitch(event)“
width=“122“ horizontalCenter=“-398“ verticalCenter=“78“/>
            <mx:Button id=“clearButton“ width=“33.6“ toolTip=“clear“ click=“clearButtonHandler()“  styleName=“clearButton“ height=“26.4“ horizontalCen-
ter=“-379“ verticalCenter=“4“/>
            <mx:Button id=“saveButton“ click=“imageExport()“ styleName=“export“ horizontalCenter=“373“ verticalCenter=“301“/>
            <mx:Image id=“imagePLatzhalter“ width=“200“ height=“125“ horizontalCenter=“-194“ verticalCenter=“300“ click=“loadImage1(event)“/>
            <mx:Image id=“imagePLatzhalter0“ width=“200“ height=“125“ horizontalCenter=“14“ verticalCenter=“300“ click=“loadImage2(event)“/>
            <mx:Image id=“imagePLatzhalter1“ width=“200“ height=“125“ horizontalCenter=“222“ verticalCenter=“300“ click=“loadImage3(event)“/>
            <mx:Button click=“imageAbspeichern()“ styleName=“save“ horizontalCenter=“373“ verticalCenter=“256“/>
            <mx:Button id=“saveButton0“ label=“ExpXML“ click=“ExportImage.makeXML()“ horizontalCenter=“472“ verticalCenter=“301“/>

            <mx:HSlider id=“volumeslider“ minimum=“0“ maximum=“1“ change=“changeSliderHandler(event)“  toolTip=“Volume“ width=“121.96666“
styleName=“volumeSlider“ height=“20.933334“ horizontalCenter=“-399“ verticalCenter=“50“/>
            <mx:Text text=“Picassor Orchestra&#xa;“ width=“634“ styleName=“panelTitle“ fontSize=“60“ color=“#565656“ horizontalCenter=“5“ verticalCen-
ter=“-299“/>

    </mx:Canvas>
                  <mx:Style source=“assets.css“/>
</mx:Application>
```