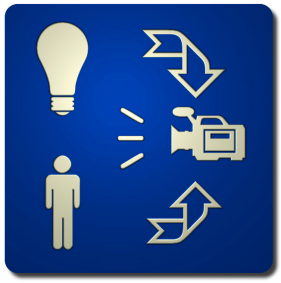
INTERACTIVE WEBCAM PACKAGE DOKUMENTATION

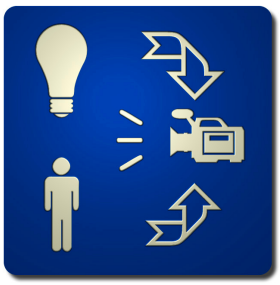


Seit 2006/2007 spiele ich (Florian Weil / der hess) ein bissl mit bildbasierter Interaktion herum. Daraus entstanden ist das hier vorliegende Interactive Webcam Package für ActionScript 2 (Adobe Flash).

Das Interactive Webcam Package soll Flash Programmierer bei ihrer Entwicklung von bildbasierter Interaktion unterstützen und so den dazu gehörigen Aufwand hoffentlich auf ein erträgliches Maß senken.
Momentan sind eine devicelose und eine deviceabhängige Interaktion möglich. Bei der devicelosen Interaktion interagiert man mit seinen Körper. Technisch wurde dieses Verfahren mittels MotionDetection und Differenzkeys umgesetzt.

Die deviceabhängige Interaktion erfolgt mittels einer Lichtquelle (z.B. Taschenlampe) Die Lichtquelle wird im Bild getrackt und als Ergebnis erhält man ein Rechteck mit Positions- und Größenangabe der Lichtquelle zurück. Dieses Tracking funktioniert auch mit mehreren Lichtquellen, so dass man z.B. ein Stab visualisieren kann mit 2 Lichtquellen.

Die CamButtonManager – Klasse



Die CamButtonManager Klasse erstellt und verwaltet Buttons bzw. Hotpoints im Bild. Die Klasse überprüft ob der Bereich im Bild aktiv ist oder nicht und führt dem entsprechend ein Ereignis des CamButtons aus. Die CamButtonManager Klasse arbeitet in zwei verschiedenen Modi. Der CamButtonManager.STATIC Modus arbeitet mit einem Differenzkey, der CamButtonManager.DYNAMIC Modus dagegen arbeitet mit dem Motion Detection Prinzip.
Bei Modi haben ihre Vor- und Nachteile, die in der unteren Tabelle zu sehen sind.

CamButtonManager.STATIC	***CamButtonManager.DYNAMIC***
	
Vorteile: - klare Lokalisierung der Peson im Bild - Gute Basis für Kollisionsanwendungen - Ereignis onHandOver() funktioniert zuverlässiger	**Vorteile:** - Sehr störungsrobust gegenüber Belichtungsänderung (Blendenautomatik) - Nur bewegungsreiche Bereiche im Bild werden angezeigt
Nachteile: - Sehr anfällig für Licht- und Schärfeänderungen im Bild - Funktioniert bei Kameras mit Automatik nur sehr schlecht - Schlechte Handhabung für Endnutzer, weil ein Snapshot (Bild) ohne Person erstellt werden muss	**Nachteile:** - Perfomancehungriger gegenüber des STATIC-Modus - onHandOver() und onHandOff() Ereignis nur bedingt verwendbar, weil wenn der Körper über den Buttonbereich stehen bleibt, verschwindet die Aktivität.

Variablen und Methoden der Klasse CamButtonManager

. static CamButtonManager.STATIC	Statische Variable für den Static-Modus
. static CamButtonManager.DYNAMIC	Statische Variable für Dynamic Modus
.CamButtonManager(webcamSignal:MovieClip, camMode:Number)	Konstruktor
.createCamButton(xpos:Number,ypos:Number, width:Number,height:Number):CamButton	Erzeugt einen CamButton
.addCamButton(button:CamButton):Boolean	Fügt einen schon bestehenden Button der Liste wieder hinzu
.removeCamButton(button:CamButton):Boolean	entfernt einen Button aus der Liste
.doSnapshot():Void	Erstellt ein Snapshotbild für den Static-Modus
.startAnalyze():Void	Startet den Analyse Modus der Buttons
.getBitmapData():BitmapData	Gibt das Analysebild zurück
.getMode():Number	Gibt den aktuellen Modus des CamButtonManagers zurück
.setMode(modus:Number)	Mithilfe der statischen Variablen STATIC und DYNAMIC kann man hier den Analysemodus zur Laufzeit verändern
.getAccuracy():Number	Gibt den Wert der Button-empfindlichkeit (Zahl zwischen 0 und 100) zurück.
.setAccuracy(value:Number):Void	Setzen der Buttonempfindlichkeit. Wert muss zwischen 0 und 100 sein
.getQuantityFrames():Number	Frameanzahl der Differenzüberlagungen beim DYNAMIC Modus
.setQuantityFrames(number:Number):Void	Setzen der Frameanzahl der Differenzüberlagungen im DYNAMIC Modus. Werte zwischen 0 und 15 werden empfohlen

Der Konstruktor

Dem Konstruktor ein MovieClip und ein CamMode übergeben. Der MovieClip muss das Webcambild enthalten und sollte von der Auflösung ein Vielfaches von 80px X 60px haben (z.B. 320x240px oder 640x480px). Programmiert wurde das System auflösungsunabhängig, daher sollte es eigentlich mit jeder Bildauflösung funktionieren.
Der CamMode wird mit den statischen Variablen der Klasse CamButtonManager übergeben.

Beispielcode:

```
import de.derhess.iaCam.CamButtonManager;
import de.derhess.iaCam.CamButton;

var manager:CamButtonManager = new
CamButtonManager(webcamBild, CamButtonManager.STATIC)

oder

var manager:CamButtonManager = new
CamButtonManager(webcamBild, CamButtonManager.DYNAMIC)
```

CamButtons erstellen

CamButtons sind rechteckige Bereiche im Bild. Wenn diese Bereiche aktiv sind (weisse Pixel im Analysebild) und den Buttonempfindlichkeitswert (Standardmäßig: 80%) übersteigen, dass heisst wenn über 80% der Pixel in diesem Bereich weiss sind, ist der Button aktiv.
CamButtons können nur mit der CamButtonManager Klasse erstellt. Die Implementierung orientiert sich an dem Factory-Design Pattern.
Als Parameter übergibt man der createCamButton() Funktion eine X und Y Koordinate (der Ankerpunkt liegt immer links oben), sowie eine Breite und Höhe.

Beispielcode:

```
import de.derhess.iaCam.CamButtonManager;
import de.derhess.iaCam.CamButton;

...

var button1:CamButton = manager.createCamButton(5,5,30,30);
var button2:CamButton = manager.createCamButton(50,65,50,50);
```

CamButtons entfernen

Wenn man einen Button nicht mehr benötigt entfernt man ihn einfach mit der Funktion removeCamButton(). Als Parameter übergibt man die entsprechende CamButton-Instanz. Ist diese Instanz in der ButtonListe enthalten, wird der Button aus dieser Liste gelöscht und man erhält den Wert *true* zurück.

Beispielcode:

```
import de.derhess.iaCam.CamButtonManager;
import de.derhess.iaCam.CamButton;

...

var button:CamButton = manager.createCamButton(5,5,30,30);

if(manager.removeCamButton(button))
     trace(„Löschung erfolgreich“);
else
     trace(„Löschung nicht erfolgreich“);
```

CamButtons wieder hinzufügen

Aus der CamButtonManager gelöschte Buttons können dem CamButtonManager wieder mit der Methode addCamButton(alterButton:CamButton) zugefügt werden. Wenn dieser Button schon vorhanden ist erhält man den Wert *false* zurück.

Die Methode startAnalyze()

Die Methode startAnalyze() startet die Bildanalyse und muss periodisch ausgeführt werden. Die Funktion kann in einer Funktion setInterval(…) oder in einem MovieClip.onEnterFrame stehen.

Beispielcode:

```
import de.derhess.iaCam.CamButtonManager;
import de.derhess.iaCam.CamButton;

var manager:CamButtonManager = new
CamButtonManager(webcamBild, CamButtonManager.DYNAMIC)

var button:CamButton = manager.createCamButton(5,5,30,30);

_root.onEnterFrame = function () {
     manager.startAnalyze
}
```

Die Methode doSnapshot()

Die Methode doSnapshot() muss in Verbindung mit dem STATIC-Modus vorher ausgeführt werden. Denn mit der Hilfe von doSnapshot() wird ein Frame (Bild) zwischengespeichert. Dieses Bild wird nacher bei der Bildanalyse mit dem neusten Frame verrechnet. Beim Ausführen dieser Funktion sollte der User nicht im Bild sein, damit der Differenzkey richtig funktionieren kann.

Beispielcode:

```
import de.derhess.iaCam.CamButtonManager;
import de.derhess.iaCam.CamButton;

var manager:CamButtonManager = new
CamButtonManager(webcamBild, CamButtonManager.STATIC)

var button:CamButton = manager.createCamButton(5,5,30,30);

//Webcambild ohne User
manager.doSnapshot();

_root.onEnterFrame = function () {
     manager.startAnalyze
}
```

Die Methode getBitmapData()

Die Methode getBitmapData() gibt das aktuelle BitmapData Bild zurück. Meistens in der Größe 80x60px

Alles rund um Accuracy / Buttonempfindlichkeit

Mit der Funktion setAccuracy() könnt ihr die Buttonempflindlichkeit zur Laufzeit verändern. Stanardmäßig ist der Wert auf 80 eingestellt. Bei diesem Wert handelt es sich um einen Prozentwert, wie viele Pixel in der Buttonfläche aktiv sein müssen (weisse Pixel)

Beispielcode:

```
import de.derhess.iaCam.CamButtonManager;
import de.derhess.iaCam.CamButton;

var manager:CamButtonManager = new
CamButtonManager(webcamBild, CamButtonManager.STATIC)

manager.setAccuracy(60);

trace(„Button Empfindlichkeit: „ + manager.getAccuracy());
```

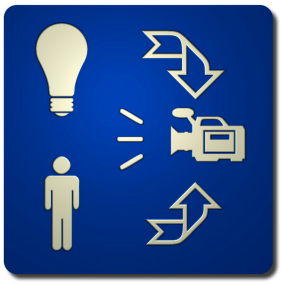
Alles rund um QuantityFrames / Motion Detection

Die Funktionen rund um QuantityFrames können im MotionDetection Modus (CamButtonManager.DYNAMIC) eingesetzt werden. Mit der Funktion setQuantityFrames(anzahlFrames) kann eingestellt wie lange auf Bewegung reagiert werden kann. Standardwert sind 5 Frames. Allgemein kann man empfehlen nicht über 15 Frames zu gehen, da dann die Verzögerung der Bewegung zu lange andauert.

Beispielcode:

```
import de.derhess.iaCam.CamButtonManager;
import de.derhess.iaCam.CamButton;

var manager:CamButtonManager = new
CamButtonManager(webcamBild, CamButtonManager.DYNAMIC)

manager.setQuantityFrames(3);

trace(„Anzahl QuantityFrams: “ +manager.getQuantityFrames());
```

Die CamButton – Klasse



Die CamButton Klasse beinhaltet die Position und Größe des CamButtons bzw. die Position und Größe der HotArea im Webcam Bild. Je nach Zustand des CamButtons werden die Ereignisse onHand(), onHandOver() und onHandOff() ausgeführt. Der CamButton kann nur über den CamButtonManager.createCamButton() erstellt werden.

Variablen und Methoden der Klasse CamButton

.CamButton(xpos:Number,ypos:Number, breite:Number, hoehe:Number,scale:Number)	Konstruktor
.move(xpos:Number,ypos:Number):Void	Verschieben des CamButton
.changeSize(breite:Number,hoehe:Number):Void	Verändern der Größe
. getPosition():Point	Koordinate des CamButtons
. getCamButtonRect():Rectangle	liefert alle geometrischen Information des CamButton
. getAnalyzeRect():Rectangle	Geometrische Information über den CamButton im Analysebild des CamButtonManagers

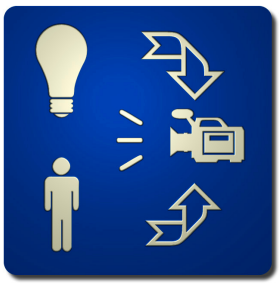
Mit Ereignissen der CamButtons arbeiten

Ein CamButton kann 3 verschiedene Ereignisse werfen. Das Ereignis onHand() wird ausgeführt wenn der Button vorher unaktiv war und jetzt aktiv wird. Z.B. durch das erste Berühren der Hand des CamButtons.
Das zweiter Ereignis onHandOver() wird ausgeführt, wenn der Button bereits aktiv war und noch ist. Dieses Ereignis tritt meisten auf wenn sich die Hand weiterhin über den Button bewegt.
Das dritte Ereignis onHandOff() tritt ein wenn der CamButton vorher aktiv war und jetzt nicht mehr aktiv ist. Das tritt ein wenn z.B. die Hand über den Button verschwindet.

Beispielcode:

```
import de.derhess.iaCam.CamButtonManager;
import de.derhess.iaCam.CamButton;

var manager:CamButtonManager = new
CamButtonManager(webcamBild, CamButtonManager.STATIC);

var button1:CamButton = manager.createCamButton(5,5,30,30);
var button2:CamButton = manager.createCamButton(50,65,50,50);

button1.onHand = function () {
     trace(„Button1 onHand“);
}

button1.onHandOver = function() {
     trace(„Button1 onHandOver“);
}

button1.onHandOff = function() {
     trace(„Button1 onHandOff“);
}

manager.doSnapshot();

_root.onEnterFrame = function () {
     manager.startAnalyze();
}
```

Die LightTracking - Klasse



Die LightTracking Klasse ist eigentlich nur eine Tracking Klasse.Es wird die Position und Größe einer oder mehrerer Lichtquelle ermittelt. Das Tracking erfolgt mittels einem Helligkeitsschwellenwert und dieser kann zur Laufzeit verändert werden.
Beim Tracking von mehreren Lichtern wird versucht das Trackingergbnis so aufzubereiten, dass alle Lichtquellen immer den gleichen Index im Ergebnis Array haben. Diese Sortierung erfolgt mittels Distanzberechnung zum vorherigen Trackingergbnis. In der Praxis hat sich das Verfahren als sehr brauchbar erwiesen.

Variablen und Methoden der Klasse LightTracking

. static LightTracking.ONE_LIGHT	Statische Variable für das Tracken einer Lichtquelle
. static LightTracking.MORE_LIGHTS	Statische Variable für das Tracken mehrer Lichtquellen
.LightTracking(webcamSignal:MovieClip, schwellenwertHelligkeit:Number)	Konstruktor
. startTracking(modus:Number):Void	Starten der Bildanalyse und als Übergabeparameter muss eine der statischen LightTracking Variablen übergeben werden
. getBitmapData():BitmapData	Gibt das Analysebild zurück
. getThreshold():Number	Gibt den Schwellenwert des Helligkeitstracking zurück
. setThreshold(schwellenwert:Number):Void	Ändern des Helligkeitsschwellenwertes. Muss zwischen 0 und 255 sein
. addEventListener(ArtEreignis:String, Listener:Obeject):Void	Anhängen eines Listeners
. removeEventListener(ArtEreignis:String, Listener:Obeject):Void	Entfernen eines Listeners
.Ereignisse:	
.onNoLightFound(event:Object)	Wenn kein Licht vorhanden ist im ONE_LIGHT Modus. Man erhält die LightTracking Instanz
.onNoLightFound(event:Array)	Wenn kein Lichtvorhanden ist im MORE_LIGHTS Modus. Man erhält ein Array gefüllt mit Rechteck-Objekten der Lichter

.onTrackingResult(event:Rectangle)	Trackingergbnis im ONE_LIGHT Modus. Das Ergebnis wird in einem Rechteck Objekt übergeben
.onTrackingResult(event:Array)	Tracking Ergebnis im MORE_LIGHTS
.onAddLight(event:Number)	Wird ausgeführt wenn ein oder mehrere Lichter hinzugefügt werden. Es wird die Zahl der hinzugefügten Lichtquellen übergeben
.onRemoveLight(event:Number)	Wird ausgeführt wenn ein oder mehrere Lichter entfernt werden. Es wird die Zahl der entfernten Lichtquellen übergeben

Der Konstruktor

Der Konstruktor der LightTracking Klasse benötigt ein MovieClip in dem das Webcamsignal enthalten ist. Die Auflösung des Webcam Bildes sollte ein Vielfaches von 80x60px (z.B. 320x240, 640x320) sein. In Test ging es auch mit anderen Auflösungen und anderen Seitenverhältnisse.
Der zweite Parameter muss der Helligkeitsschwellenwert sein. Der Wert muss zwischen 0 und 255 sein. Von diesem Wert ist es abhängig ab welchen Helligkeitswert die Lichtquelle erkannt wird. Werte zwischen 230 und 240 lieferten gute Ergebnisse.

Beispielcode:

```
import de.derhess.iaCam.LightTracking;

// eine Lichtquelle tracken
var track:LightTracking = new LightTracking(webcamBild,
LightTracking.230);
```

Starten des Trackings

Mit der Funktion startTracking() startet man die Analyse des Webcambildes. Diese Funktion muss periodisch ausgeführt werden. Das heißt die Funktion kann per setInterval() oder durch das Ereignis onEnterFrame periodisch ausgeführt werden. Als Übergabeparameter wird einer der zwei statischen LightTracking Variablen benötigt.
Das Tracken mehrerer Lichtquellen ist performancehungriger und sollte bei der Verwendung der LightTracking Klasse beachtet werden

Beispielcode:

```
import de.derhess.iaCam.LightTracking;

// eine Lichtquelle tracken
var track:LightTracking = new LightTracking(webcamBild,230);

_root.onEnterFrame = function () {
     // Ein Licht tracken
     track.startTracking(LightTracking.ONE_LIGHT);

     oder

     // Mehrere Lichter tracken
     track.startTracking(LightTracking.MORE_LIGHTS);
}
.
```

Mit Ereignissen arbeiten

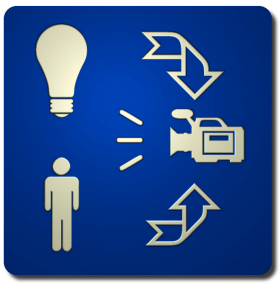
Als Beispiel für das Arbeiten mit dem Ereignissen der LightTracking Klasse sollte ein Beispiel im MORE_LIGHTS Modus ausreichen.

Beispielcode:

```
import de.derhess.iaCam.LightTracking;

// Listener Objekt erstellen
var listener:Object = new Object();

listener.onNoLightFound = function(event:Array) {
     trace(„Kein Licht im Bild vorhanden“);

     for(var i:Number = 0; i < event.length; i++) {
          trace(„altes Trackingergebnis: „+ event[i].x +
                                        „:“ + event[i].y);
     }
}
```

```
listener.onTrackingResult = function(event:Array) {
     trace(„Anzahl Lichtquellen: “ + event.length);

     for(var i:Number = 0; i < event.length; i++) {
          trace(„Trackingergebnis Position: „+ event[i].x +
                                        „/“ + event[i].y);

          trace(„Trackingergebnis Size: „+ event[i].width +
                                        „/“ + event[i].height);

     }
}

listener.onAddLight = function(event:Number) {
     trace(„Anzahl neuer Lichtquellen: “ + event);

}

listener.onRemoveLight = function(event:Number) {
     trace(„Anzahl entfernter Lichtquellen: “ + event);

}

// LightTacking Instanz erstellen
var track:LightTracking = new LightTracking(webcamBild,230);

track.addEventListener(„onNoLightFound“,listener);
track.addEventListener(„onTrackingResult“,listener);
track.addEventListener(„onAddLight“,listener);
track.addEventListener(„onRemoveLight“,listener);



// Mehrere Lichtquelle tracken
_root.onEnterFrame = function () {
     track.startTracking(LightTracking.MORE_LIGHTS);
}
```

Planung Für Die Zukunft



In Zukunft möchte ich noch den Algorithmus des LightTrackings in Sachen Perfomance verbessern. Zusätzlich wäre ich sehr glücklich darüber, wenn ich noch in Zukunft ein ColorTracking implementieren könnte, wie es GSkinner bei seinem CamWriter programmiert hat. (siehe http://incomplet.gskinner.com/index2.html#camwriter)

Ein wohl nicht umsetzbarer Wunsch mit ActionScript wäre eine Gestenerkennung mit Fingerzeichen.

Siehe:
http://www.uni-koblenz.de/~cg/Diplomarbeiten/DABreuer.pdf
http://www.techfak.uni-bielefeld.de/ags/ai/publications/master-theses/Gaertner2005-DIP.pdf
http://www.pt-it.pt-dlr.de/_media/01_rogalla.pdf

Wenn jemand da draußen Interesse hat das Interactive Webcam Package mit weiter zu entwickeln, freue ich mich über jede Mail (info@derhess.de). Desweiteren bin ich auch über jeden Tipp / Idee sehr dankbar!!! ;-)

Nutzungsbedingungen

Das Interactive Webcam Package darf für private und Testzwecke uneingeschränkt benutzt und modifiziert werden.
Bei kommerzieller Nutzung bitte eine Mail an info@derhess.de schreiben um die Nutzungsdetails zu besprechen.

Vielen Dank!