Rochester Institute of Technology

A Thesis submitted to the Faculty of the
College of Imaging Arts and Sciences
in candidacy for the degree of
Master of Fine Arts
Computer Graphics Design

Beauty of Taiwan

by Yu-Chung Chang

May 1, 2011

Approvals

Chief Advisor: Chris Jackson, Associate Professor, Computer Graphics Design

Signature of Chair Advisor Date

Associate Advisor: Dan DeLuna, Assistant Professor, Computer Graphics Design

Signature of Associate Advisor Date

Associate Advisor: Nancy Doubleday, Associate Professor, Information Technology

Signature of Associate Advisor Date

School of Design Chairperson:
Patti Lachance, Associate Professor, School of Design

Signature of Administrative Chairperson Date

Reproduction Granted:

I, YU-CHUNG CHANG, hereby grant permission to Rochester Institute of Technology to reproduce my thesis documentation in whole or part. Any reproduction will not be for commercial use or profit.

Signature of Author Date

Inclusion in the RIT Digital Media Library Electronic Thesis and Dissertation (ETD) Archive:

I, YU-CHUNG CHANG, additionally grant to Rochester Institute of Technology Digital Media Library the non-exclusive license to archive and provide electronic access to my thesis in whole or in part in all forms of media in perpetuity. I understand that my work, in addition to its bibliographic record and abstract, will be available to the worldwide community of scholars and researchers through the RIT DML. I retain all other ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles and books) all or part of this thesis. I am aware that Rochester Institute of Technology does not require registration of copyright for ETDs. I hereby certify that, if appropriate, I have obtained and attached written permission statements from owners of each third party copyrighted matter to be included in my thesis. I certify that the version I submit is the same as that approved by my committee.

Signature of Author Date

Abstract

Taiwan Images
Yu-Chung Chang

The term “Multimedia” refers to not only the materials used in creating art, but also the devices used for presenting the content. In the present time, artists can practice computer softwares as the media. They use audio and video tracks to display artwork. Various formats of technological or digital multimedia enhance the user’s experience in entertainment or art, and transcend everyday experience.

“Beauty of Taiwan”, my MFA Thesis, is a Flash-based program which offers users interactivity to control the way to view art piece as used in video games. This program allows the presented film, made from my pencil drawings on the images of Taiwan, to be affected by the viewer’s movements. Starting from when he/she first enters the affected zone situated in front of the screen, the program detects his/her initial location, and the film starts playing depending on his/her followup movements.

The opening has a sequence of images flowing in from either side of the screen, according to where the viewer enters the affected zone in relation to the screen. If the viewer enters the center of the zone, or if more than one viewer comes in from both sides at the same time, the film starts playing from the center of the screen. If no one is in the zone, an animated logo is set to loop as a screen saver.

Setting up a camera that collects proper amount of information is very critical. Through Flash, I’m able to locate viewer’s movements. They are caught by the camera. That information is then transferred from the captured images and becomes the controller for the viewer to navigate through the film.

By having the interactivity between the viewer and the exhibited artworks, the brilliant artworks that used the traditional materials, are no longer just static images hung on the walls of museums, or of our own homes. In my project, the art piece now also attract instant conversations and simultaneous reactions between the creator and the audiences. The interactivity brings out the fun, communication and humanity to the creator, the audiences and the artworks.

Used Softwares:
Flash and After Effects

Related Professions:
Motion Graphics and Interactive Art

see “Beauty of Taiwan” at
http://taiwan.cheerevelyn.com/

Content

Introduction

In the past, people have had no choice but to browse through each static art work one by one in the museums. Have you ever wondered about making the artwork change, move, and walk alongside with the viewers? The central idea of this project is to display my artwork in a novel and interactive way so that viewers will have new surprises for every single step. For example, when someone walks from the left side of a prolonged screen toward right, a short film will be displayed from the left to right along with the viewer and vise versa.

This is a Flash-based interactive art work. It utilizes a camera to detect people's position, and then randomly choose one film out of four topics to be displayed. All the four topics are the unique features of my motherland Taiwan and contrast much to my oversea experience in United States- Hakka (a race in Taiwan), transportation, aboriginal, and folk beliefs. Each topic is presented by two thirty -second films and each short film is composed of pencil drawing images.

Similar to the aforementioned left and right starts, the film is set to start from the middle if the viewer initiates the contact in the middle, or if two viewers simultaneously walk in from opposite directions. A recurrent animation of a Taiwan logo will appear in the center if no one is under the radar of the camera. Every feature works if one viewer is replaced by a group.

Noticing the change on the screen display, viewers will immediately realize that their positions play a role in what they will see next. Viewers get entertained and linger in front of the screen to enjoy more artwork. Artists tell stories via creating artwork. By incorporating the new techniques in interactive art, I long for providing the viewers a whole new experience while listening to my story of Taiwan.

My artwork is like stories that an artist wants to tell, and they can be told in a different and special way that combines digital multimedia. In this project, my artwork are going to be shown according to viewers' positions. My artwork appear like scenes around when people walk. This way makes people interested in spending time watching and feeling my artwork.

Review of Literature

The Art and Science of Interface and Interaction Design
by Christa Sommerer, L. C. Jain, Laurent Mignonneau
Springer, 2008

This book introduces interactive art or design of many forms such as games, web pages, shows in a public space or a semi-pubic space, etc. Discussed in this book, many interactive art or interaction design projects often involve digital process. The early developed technology has greatly influenced new design.

Information Arts: Intersections of Art, Science, and Technology
by Stephen Wilson
MIT Press, 2003

Through the point of view of an artist, the book talks about the relation between art and science. It also introduces different forms of technology in which art is expressed. In the form of computers, the ways to activate an art show include motion, gesture, touch and tactility, gaze, face recognition, balance, walking, bicycling, breath, etc.

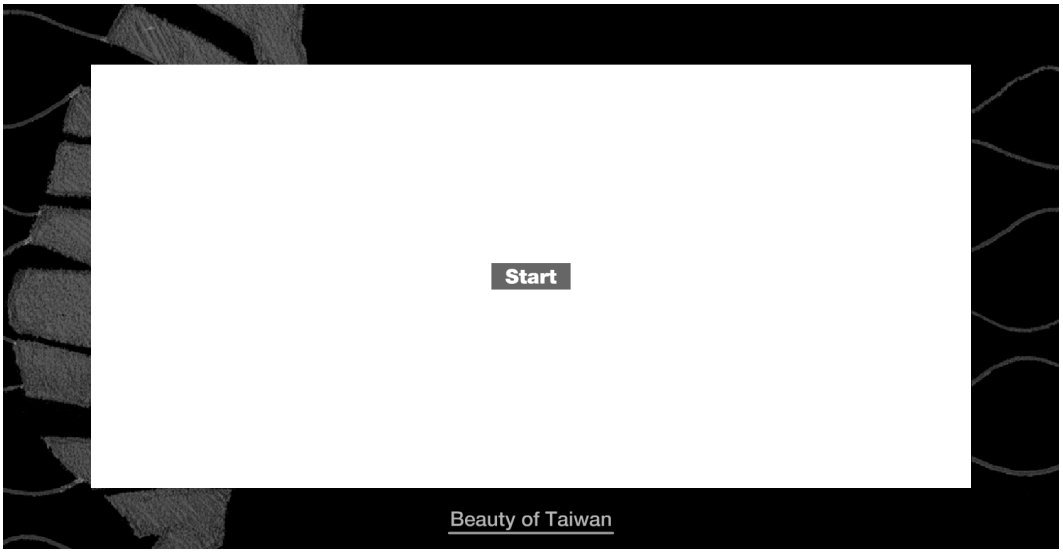
The Language of New Media
by Lev Manovich
MIT Press, 2002

The author discusses the term, new media, from the angles of theories and histories. New media, which often combines different apparatuses, differs from the single word, media. It strengthens the interaction between watchers and a screen, and adds physical controls to emotional communication. New media has affects the interface design, representation space, etc.

Process

I. Operation of Flash Codes

Through a camera, Flash detects the position of people's movements. An operator first clicks the button, Start, which is a one time click during the whole experiencing, to start using this program.



The following is an opening film which is around ten seconds, and then, Flash detects if there are people in front of the camera, and responds with different films starting from different sides according to people's positions.

Following people's movements, motion graphics display relative parts. Down below a film, there is a brief introduction about the film. There are two thin bars above and below the area where a film shows. The two bars' length and positions change with the range over which people are distributed, and they supply a hint of the interaction between people and a film.

After showing about thirty seconds is a ten-second break time. There is a sign, Come Here, attracting and directing people to come front to interact with this program.Then, Flash detects people's positions and starts all over again. If there is no one there, the sign, Come Here, will also appear.

A. Detect and Follow One or More People

Flash gets continuous video images of the viewers from a camera, and captures them in a millisecond. Meanwhile, it also compares the differences between each image, and displays them as green dots on the screen. The x positions of the green dots are traced in an array.

Depending on where the green dots appear on the screen, the film is arranged to start playing either from the left, right, or center of the screen. During the screening, the images unfold to the followup movements of the green dots. Flash knows that the greens dots are representing either one or multiple viewers.

When there's only one viewer entering the affected zone, a sequence of images starts from the same side, either the left or the right, as the green dots appeared. Things get complicated when there are more than one entry. Since the viewers can easily come from any directions, the film starts from the center of the screen, and the sequence of images unfolds symmetrically to the furtherest green dots detected from the center. On the other hand, if the film starts from either the left or right side of the screen, the sequence of images unfolds to the furtherest green dots detected from the starting side.

Flash detects the initial positions of the green dots at the opening of the film, and also at the end of each break. For every thirty seconds, the film takes a ten-second break. An animated logo loops through the break until the next show. However, if there are no green dots detected by Flash, the loop remains active and the film is not played.

B. Problems of Flash and Solutions

1. Flash Takes Times to Start Up

When starting the program, a window pops out and asks viewer's permission on camera access. The viewer clicks on the "allow" button and Flash starts initializing the camera. However, the initialization takes a few seconds in order for Flash to collect the correct information.

Solutions - Use an opening film to fill the time gap. I create a greeting shorts to cover for Flash.

2. Flash Misses Capturing Sometimes

In order to have a smooth viewing experience, It is very critical to have images on every Flash keyframe. Unfortunately, that is not always the case. When no green dots are detected by Flash, it returns a null value. Not only there is no array, there are also no images being captured. Although we can't finger point the lack of images when it happens, due to persistence of vision, the film flicks before our eyes.

Solutions - Create two sets of variables- MAX, MIN and keepMAX, keepMIN. Every time when Flash captures images and generates a valid array representing x positions of all green dots, MAX (MIN) equals to the biggest (smallest) value. At the same time, the variable, keepMAX (keepMIN), equals to MAX (MIN.) While Flash does not capture images and generates no array, MAX (MIN) equals to the value of keepMAX (keepMIN) that just recorded down. In this way, null values are avoided.

C. The Making of Short Films

1. Display Motion Graphics

There are three ways to display motion graphics of a short film. Each way performs a little bit differently, and has its own features.

Take a ball as an example. Make a ball scale from very small to very big and then keep rotating in its position.

a. The first way is to create all the motion in the timeline. However, since a short film displays according to people's position, and people's bodies actually shake a little bit, the film will shake, keep going forword and backward. It looks unstable this way.

b. The second way is to make all the motion be embed in a symbol and then put the symbol on the stage in a proper frame of the timeline. The ball starts scaling and rotating when users go to the relative position.

However, when users go backward, the ball will not scale down but suddenly disappear on the stage.

c.The third way is to make the motion of scaling happen in the timeline of the stage, and make the motion of rotating in a symbol. In this way, when people walk backward, they can see the ball scaling down and disappear rather than disappearing rapidly.

The best solution is the the third way. Most of the motion graphics in this project are made in this way.

2. Relation Between People's Position and Flash Timeline

Each of the subjects has three films that start from the left, right and center. To make appearances of motion graphics corespond to people's positions, it is important to undersand how Flash timeline works.

Since in this project, most motion graphics are made in the third way mentioned above- motion tweens are embed in symbols. Objects on the stage can only display and perform well when the order of their appearing timing in the time line is from the beginning (left) toward the end (right.) Making the playhead play backward will result in flashing images, and that symbols all play from their beginning at the same time, rather than starting one by one when people walk by. Therefore, the making of the three displaying ways is different.

a. Making films starting from the left is the easiest among all. Objects near the left side of the stage start earlier in the main timeline. When people enter from the left side, motion graphics play in order.

b. On the contrary, when making films starting from the right, objects near the right side of the stage start earlier in the main timeline. When people enter from the right side, the playhead is demanded to move from the left side of the main timeline. Thus motion graphics show normally.

c. When making films starting from the middle, the first half of the main timeline is empty, and all the motion graphics are set in the last half of the main timeline. Objects near the middle of the stage start earlier in the main timeline, and objects near the two sides of the stage start later in the main timeline.

When people are in front of the camera, there are another three situations to consider. When people are in the right half area of the screen, the playhead in the main timeline moves from the middle to the right depending on how far people are from the middle. When people are in the left half area of the screen, people's position, the furtherest point is reflected to get a value that is able to demand the playhead in the main timeline still moves from the middle to the right. When people's area covers the middle of the screen, if the right part is bigger than the left part, the playhead moves according to the furtherest point of people's area, and if the left part is bigger than the right part, the point needs to be reflected to get a proper value for the playhead to move.

D. Set a Camera

It is very important to set a camera in a proper way. The wider range a camera can cover or the further a camera can be from people, the better this project can perform. A small range or being too close to people will cause too many green dots appearing on the screen at one time. As a result, it is not obvious that motion graphics are folloing people.

II. Art Works in Short Films

The contents of short films are pencil drawing images. I had learned fine art for a long time. Now, combining with computer graphics design, the steady images become motion graphics. The topic is Beauty of Taiwan, and the four subjects are transportation, folk beliefs, aboriginal, and Hakka (a race in Taiwan.) For each subject, there are two short films. To make each subject more impressive, each film is given a topic, and a slogan is added in each film to make the topic and meaning clear.

A. Hakka

1. Blue Blouse

Hakka people's traditional blue blouse has beautiful patterns and a special form. It was designed to make it easy and comfort for women while they worked on a farm. Now it represents the diligent spirit of Hakka.

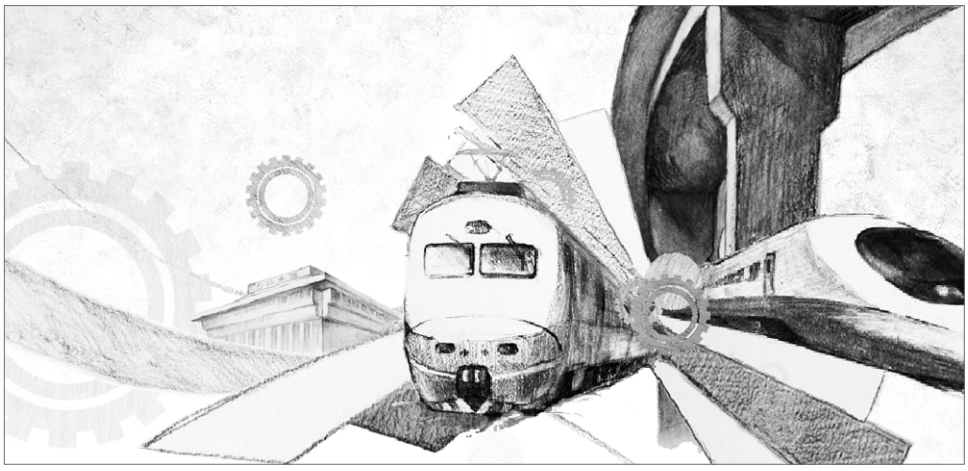
2. Tung Blossom

Hakka people originated in mainland China, and parts of them moved to Taiwan many years ago. Tung trees once were Hakka people's industrial crops. Tung blossom in May is part of their memory. It is now a symbol of Hakka.

B. Transportation

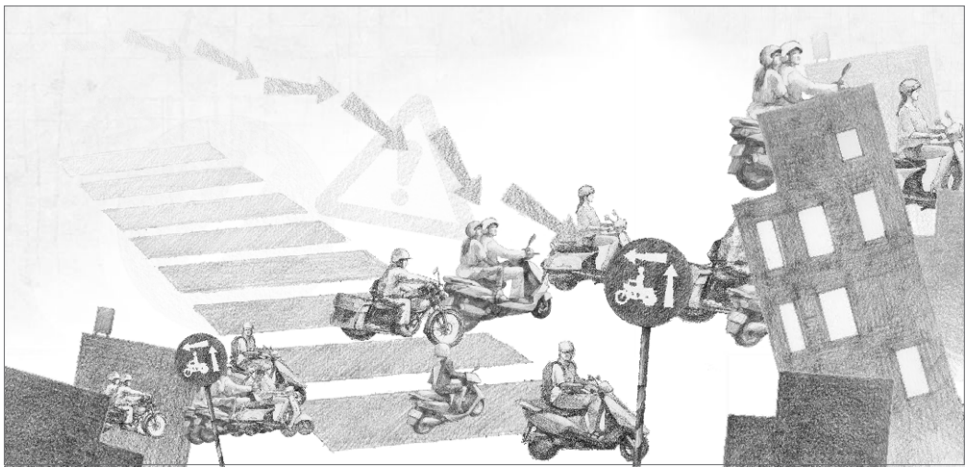
1. Taiwan Railway and HSR

Taiwan Railway has served many years, and it has played a very important role in the development of industry in Taiwan. In 2007, Taiwan High Speed Rail (HSR) was completed and opened to give more transportation. Now, it only takes four hours to travel from the north to the south. Taiwan High Speed Rail has made it more convenient in travel and business affairs.



2. Motorcycles

Taiwan is crowded. On the streets, besides cars, there are a lot of motorcycles. They stop behind traffic lights, and when the lights turn to green, they rush out rapidly.



C. Aboriginal

1. Totems and Patterns

Besides races that came from mainland China hundreds years ago, there are fourteen races that originated in Taiwan. Each race has its own culture, belief, activities. They feature in totems and patterns on clothes, appliance, sculptures, etc.

2. Harvest Festival

Most of the races hold ceremonies for harvest and the hunt. Now they have joint ceremonies for conveying and passing their cultures.

D. Folk Beliefs

1. Worship

Many people believe in Buddhism and Taoism. Besides, in many places, there are Village Gods, who protect people in health and wealth. There are many temples and many different gods and goddesses. Worship is a part of daily life and also a yearly activity.

2. Food for Festivals

In Taiwan, there are many festivals on the lunar calendar. In the Chinese New Year, there are meals cooked with fish, chicken, pork, etc. Foods have their meanings, shuch as fish refering to “always getting more than you wish for” for its produnciation in Mandarin. In summer, there is the Dragon Boat Festival. People eat rice dumpling to commemorate Qu-yu-an, a patriot in history. Riding boats for finding his body now becomes the competition of riding dragon boats. In fall, in a full moon night, people get together with families and friends watching the beautiful full moon, and eat moon cakes to celebrate the Moon Festival.

Summary

With the development of technology, many aspects of our lives, including art, are influenced by computers. Rather than hanging pictures on walls, the form of showing art works can be different. This project is to make a Flash based program that uses a computer to make an interactive way to show my art works whose topic is Beauty of Taiwan.

My artwork, which are pencil drawings, are made into short films. In front of a large screen, people's movements affect the starting position of the playing of a random short film. When the position of people's movements is in the left (right) side of the screen, a short film that starts from the left (right) plays. When the position of people's movement is in the center of the screen or when people enter the area of the screen on both sides at the same time, a random film starts playing from the center. When no one is in front of the screen, a logo animation shows and it loops like a screen saver. Besides, this project can respond to not only one user but also multi-people.

I drew for the four subjects that I am concerned about- transportation, folk beliefs, aboriginal, and Hakka (a race in Taiwan.) Each subject has two short films. Taiwan is a small and crowded island. A large amount of motorcycles are the feature of Taiwan's traffic. Taiwan is a multi-culture island, and there are many traditional activities and many different religions. The ten races that originated in Taiwan have their own beautiful cultures. Hakka people immigrated to Taiwan hundreds years ago. I am a Hakkanese, and I am proud of Hakka people's delicious food, and Hakka people's features- diligence and frugality.

To make short films, some images are imported into the software, After Effects, and then the short films are imported into Flash to be ready to play, and some images are imported into Flash and made into films in Flash. Flash is the software that shows those short films in an interactive way. It detects the position of people's movements by capturing and comparing images through a camera, and plays a random short film or the looping logo animation according to the value returned from those images. The camera is set to focus on a proper area of the ground in front of the screen to get simple images and clear data for Flash to judge.

Making use of the relation between art works and watchers (users) can make it more interesting and attractive to show an artist's work. In this project, people can find out that they are able to affect the playing of the short films. When they watch my art works, they also have fun with the form of showing art works. Combining digital multimedia, this project makes it fascinating to share my feelings and thoughts in my art works.

Conclusion

Artists express feelings and thoughts through their works. While the works are important, how they are displayed are equally weighted. The presentation decides the relationship between the art piece and its audience. An well-designed presentation for the piece has higher exposure to the general public, and it can gain greater acceptance from the right audience.

In the late 1970s, the term "Multimedia" was used to describe presentations consisting of multi-projector slide shows timed to an audio track. In the past 20 years, it refers to an electronically delivered combination of media including video, still images, audio, text in such a way that can be accessed interactively. Enhanced levels of interactivity are made possible in showcasing art projects, including my own, by using the software Flash.

I'm a Hakkanese from Taiwan, a multicultural Asian country that has mixed traditions and religions from its historical background. "Hakka", "Transportation", "Aboriginal" and "Folk Beliefs" are the four selected subjects for my film. They are the images in my head when I think of Taiwan. I sketch my ideas first and make the final drawings into eight motion graphic shorts. "Hakka" is about the diligent and frugal culture of the Hakkanese who immigrated to Taiwan hundreds of years ago. "Transportation" shows the traffic in Taiwan and it represents the energy of its residents. "Aboriginal" refers to the indigenous people of Taiwan who are Austronesian people while the majority is Han Chinese. Their languages and cultural identities are distinctive and unique. There are fourteen tribes recognized by the government of Taiwan. "Folk Beliefs" is about the various religions that the Taiwanese (Han Chinese, Hakkanese and the aboriginal) people follow.

The interactivity between my audience and the art piece allows the audience to trigger the shorts and to progress the film based on their movements. A randomly selected sequence of images starts playing as the initial movement is detected. From then on, the film literally unfolds and follows the audience's physical locations in relation to the screen.

Normally the audiences observe art pieces in a distance. Once they lost interests and walk away, the distance becomes infinite and the greeting ends. I use this mentality and turn the distance into a game between the piece and my audiences. Their locations matter to their interactive experience with the art work. Once they realize that they have control over the image arrangements and displays, they are happy to examine that power. Meanwhile, the artist renews his/her chance of continuing the storytelling process by every interactive act from the audience. This new dynamic can ensure the hard work, created by those behind-the-scene artists such as myself, to be fully delivered to the public eyes.

Appendix

Codes in intercam.as

```
1   package {
2
3       import flash.geom.Point;
4       import flash.geom.Rectangle;
5       import flash.geom.ColorTransform;
6
7       import flash.media.Video;
8       import flash.media.Camera;
9       import flash.events.*;
10
11      import flash.display.*;
12      import flash.display.Sprite;
13      import flash.display.Bitmap;
14      import flash.display.BitmapData;
15
16      import flash.utils.Timer;
17
18      import flash.system.fscommand;
19
20      import fl.transitions.*;
21      import fl.transitions.easing.*;
22
23      public class intercam extends Sprite {
24
25              // motion detection stepping
26              private static  var BLOCKS:Number=5;
27
28              // motion detection sensitivity
29              private static  var SENSITIVITY:Number=1000000;
30
31              // bitmapdata to show detected differences for better understanding
32              private var helperData:BitmapData;
33
34              // colortransform to darken old difference points
35              private var helperTransform:ColorTransform;
36
37              // detector
38              private var detector:MotionDetector;
39
40              // video
41              public var video:Video=new Video(1000,750);
42
43              // timer
44              public var timerCountDown:Timer=new Timer(5000);
45
46              public var timerPlayheadDetect:Timer=new Timer(100);
47              public var timerPlayheadMove:Timer=new Timer(500);
48
49              public var timerMovieDetect:Timer=new Timer(100);
50              public var timerMoviePlay:Timer=new Timer(60000);
51
52              public var timerBreakTime:Timer=new Timer(10000);
53
54              // the number of the frame movies go to
55              public var N:Number=new Number;
56              public var KeepN:Number=new Number;
57
```

```
58              // Art Work
59              public var movie:ArtWork = new ArtWork;
60              public var breakTimeMovie:MC_breakTime = new MC_breakTime;
61
62              // circle and its position
63              public var GreenAverage=new circle;
64
65              // area of differences
66              public var areaArrayLength:Number=new Number;
67
68              // the max/min of people's area
69              public var MAX:Number=600;
70              public var MIN:Number=400;
71
72              // code number of conditions of green dots
73              public var conditions:Number=0;
74              public var keepconditions:Number=0;
75
76              // movie frame length and the number of frame
77              public var movieFrameLength:Number=100;
78              public var FN:Number=new Number;
79
80              // A medium between the circle (green dots) and the movie
81              public var Playhead:bar=new bar;
82              public var PlayheadPosition:Number=new Number;
83              public var PlayheadScaleX:Number=new Number;
84              public var newPlayheadScaleX:Number=1;
85
86              // mask of playhead
87              public var MaskPlayhead:MC_mask_playhead=new MC_mask_playhead;
88
89              // BTN to enter and an opening
90              public var enterBTN:BTN_enter=new BTN_enter;
91              public var openingMC:MC_opening=new MC_opening;
92
93              public function intercam() {
94
95                      fscommand("allowscale","true");
96                      fscommand("fullscreen","true");
97
98                      // bitmap to show helperData
99                      var helper:Bitmap=new Bitmap;
100
101                     // getting default camera
102                     var camera:Camera=Camera.getCamera();
103
104                     helperData=new BitmapData(1000,135,true,0x000000);
105                     helperTransform=new ColorTransform(1,1,1,.7,0,0,0,0);
106
107                     helper.bitmapData=helperData;
108                     video.attachCamera(camera);
109
110                     detector=new MotionDetector(video,BLOCKS,SENSITIVITY);
111
112                     // add video, helper, GreenAverage, Movie...
113                     addChild(video);
114                     addChild(helper);
115                     addChild(movie);
116                     movie.gotoAndStop("blank");
117                     movie.x=0;
118                     movie.y=0;
119                     addChild(breakTimeMovie);
120                     breakTimeMovie.visible=false;
121                     breakTimeMovie.x=500;
122                     breakTimeMovie.y=375;
```

```
123                     addChild(GreenAverage);
124                     GreenAverage.visible=false;
125                     addChild(enterBTN);
126                     enterBTN.x=500;
127                     enterBTN.y=375;
128                     addChild(Playhead);
129                     Playhead.x=500;
130                     Playhead.y=375;
131                     Playhead.visible=false;
132                     addChild(MaskPlayhead);
133                     MaskPlayhead.x=0;
134                     MaskPlayhead.y=125;
135                     Playhead.mask=MaskPlayhead;
136
137                     // detecting all the time
138                     stage.addEventListener(Event.ENTER_FRAME,step);
139
140                     // enterBTN
141                     enterBTN.addEventListener(MouseEvent.CLICK,startCountDown);
142             }
143             // detecting
144             public function step(event:Event):void {
145
146                     // differences
147                     var differences:Array=detector.getDifferences();
148                     var GreenAverageX:Number=GreenAverage.x;
149                     var averageX:Number=0;
150
151                     // area of differences
152                     var areaArray:Array=new Array;
153
154                     var keepMAX:Number=MAX;
155                     var keepMIN:Number=MIN;
156
157                     // getting the average position and the max/min of green dots
158                     for (var i=0; i < differences.length; i++) {
159                             averageX+= differences[i].x;
160                             areaArray.push(differences[i].x);
161                     }
162                     averageX=averageX / differences.length;
163
164                     areaArrayLength=areaArray.length;
165
166                     // avoid NaN, avoid flashing movie
167                     if (areaArray.length>0) {
168                             MAX=areaArray[i - 1];
169                             MIN=areaArray[0];
170                             GreenAverage.x=1000-averageX;
171                             GreenAverage.y=100;
172                             GreenAverageX=GreenAverage.x;
173                             keepMAX=MAX;
174                             keepMIN=MIN;
175
176                             // conditions of green dots area
177                             // 1 0 1
178                             var index1:int=areaArray.indexOf(305);
179                             var index2:int=areaArray.indexOf(695);
180                             var index3:int=areaArray.indexOf(50);
181                             var index4:int=areaArray.indexOf(950);
```

```
182                         // 0 0 0 and 1 0 1
183                         if (index1===-1 && index2===-1) {
184                                 // 0 0 0
185                                 if (index3===-1 && index4===-1) {
186                                         if (areaArray.length==0) {
187                                                 conditions=0;
188                                                 keepconditions=conditions;
189                                         }
190                                         // 0 1 0
191                                         if (areaArray.length>0 && MAX<700
                                            && MIN>300) {
192                                                 if (MAX<695 && MIN>305) {
193                                                         conditions=3;
194                                                         keepconditions=conditions;
195                                                 }
196                                                 if (MAX<305) {
197                                                         conditions=2;
198                                                         keepconditions=conditions;
199                                                 }
200                                                 if (MIN>695) {
201                                                         conditions=1;
202                                                         keepconditions=conditions;
203                                                 }
204                                         }
205                                 }
206                                 // 1 0 1
207                                 if (MAX>700 && MIN<300) {
208                                         conditions=3;
209                                         keepconditions=conditions;
210                                 }
211                         } else {
212                                 // 1 0 0
213                                 if (MAX>700 && MIN>=700) {
214                                         conditions=1;
215                                         keepconditions=conditions;
216                                 }
217                                 // 0 0 1
218                                 if (MAX<=300 && MIN<300) {
219                                         conditions=2;
220                                         keepconditions=conditions;
221                                 }
222                                 // 0 1 0
223                                 if (MAX<=700 && MIN>=300) {
224                                         conditions=3;
225                                         keepconditions=conditions;
226                                 }
227                                 // 1 1 0
228                                 if (MAX>700 && MIN>=300) {
229                                         conditions=1;
230                                         keepconditions=conditions;
231                                 }
232                                 // 0 1 1
233                                 if (MAX<=700 && MIN<300) {
234                                         conditions=2;
235                                         keepconditions=conditions;
236                                 }
237                                 // 1 1 1
238                                 if (MAX>700 && MIN<300) {
239                                         conditions=3;
240                                         keepconditions=conditions;
241                                 }
242                         }
243                 }
```

```
244                     if (areaArray.length==0) {
245                             conditions=keepconditions;
246                             MAX=keepMAX;
247                             MIN+keepMIN;
248                             GreenAverage.x=GreenAverageX;
249                             GreenAverage.y=100;
250                             GreenAverageX=GreenAverage.x;
251                             keepMAX=MAX;
252                             keepMIN=MIN;
253                     }
254
255                     // darkening prevous state of helperData
256                     helperData.colorTransform(helperData.rect,helperTransform);
257
258                     // loop through difference points
259                     for (var b:String in differences) {
260
261                             var helpRectangle:Rectangle=new Rectangle
                                (differences[b].x,differences[b].y,BLOCKS,BLOCKS);
262
263                             // drawing a green BLOCKS edge sized rectangle
                                at every point
264                             helperData.fillRect(helpRectangle,0xff00ff00);
265                     }
266             }
267             // start counting down
268             public function startCountDown(event:MouseEvent):void {
269                     removeChild(enterBTN);
270                     openingMC.x=500;
271                     openingMC.y=375;
272                     addChild(openingMC);
273                     openingMC.gotoAndPlay(1);
274
275                     // count down
276                     timerCountDown.addEventListener
                        (TimerEvent.TIMER,startEveryThing);
277                     timerCountDown.start();
278             }
279             // start every thing
280             public function startEveryThing(event:TimerEvent):void {
281                     removeChild(openingMC);
282                     Playhead.visible=true;
283                     timerCountDown.reset();
284                     timerCountDown.removeEventListener
                        (TimerEvent.TIMER,startEveryThing);
285                     //start timer to detect people's position
286                     timerMovieDetect.addEventListener
                        (TimerEvent.TIMER,Position);
287                     timerMovieDetect.start();
288             }
289             // position
290             public function Position(event:TimerEvent):void {
291                     timerBreakTime.reset();
292                     timerBreakTime.removeEventListener(TimerEvent.TIMER,Position);
293                     timerMovieDetect.reset();
294                     timerMovieDetect.removeEventListener(TimerEvent.TIMER,Position);
295                     timerMoviePlay.reset();
296                     timerMoviePlay.removeEventListener(TimerEvent.TIMER,breakTime);
297
298                     // ramdom the 8 films
299                     var MN:Number=Math.floor(Math.random()*8+1);
300                     FN=8*movieFrameLength*(conditions-1)
                        +movieFrameLength*(MN-1);
301
```

```
302                     // play movie according to the position
303                     if (areaArrayLength==0) {
304                             timerMovieDetect.addEventListener
                                (TimerEvent.TIMER,Position);
305                             timerMovieDetect.start();
306                             timerPlayheadDetect.reset();
307                             timerPlayheadDetect.removeEventListener
                                (TimerEvent.TIMER,movePlayhead_Left);
308                             timerPlayheadDetect.removeEventListener
                                (TimerEvent.TIMER,movePlayhead_Right);
309                             timerPlayheadDetect.removeEventListener
                                (TimerEvent.TIMER,movePlayhead_Middle);
310                             FN=0;
311                             conditions==0;
312                             Playhead.visible=false;
313                             movie.visible=true;
314                             movie.gotoAndStop("nothing");
315                             breakTimeMovie.visible=false;
316                     }
317                     if (conditions==1) {
318                             timerMoviePlay.addEventListener
                                (TimerEvent.TIMER,breakTime);
319                             timerMoviePlay.start();
320                             timerPlayheadDetect.addEventListener
                                (TimerEvent.TIMER,movePlayhead_Left);
321                             timerPlayheadDetect.start();
322                     }
323                     if (conditions==2) {
324                             timerMoviePlay.addEventListener
                                (TimerEvent.TIMER,breakTime);
325                             timerMoviePlay.start();
326                             timerPlayheadDetect.addEventListener
                                (TimerEvent.TIMER,movePlayhead_Right);
327                             timerPlayheadDetect.start();
328                     }
329                     if (conditions==3) {
330                             timerMoviePlay.addEventListener
                                (TimerEvent.TIMER,breakTime);
331                             timerMoviePlay.start();
332                             timerPlayheadDetect.addEventListener
                                (TimerEvent.TIMER,movePlayhead_Middle);
333                             timerPlayheadDetect.start();
334                     }
335             }
336             // Break Time
337             public function breakTime(event:TimerEvent):void {
338                     timerBreakTime.addEventListener(TimerEvent.TIMER,Position);
339                     timerBreakTime.start();
340                     movie.visible=false;
341                     breakTimeMovie.visible=true;
342                     breakTimeMovie.gotoAndPlay(1);
343                     timerPlayheadDetect.reset();
344                     timerPlayheadDetect.removeEventListener
                        (TimerEvent.TIMER,movePlayhead_Left);
345                     timerPlayheadDetect.removeEventListener
                        (TimerEvent.TIMER,movePlayhead_Right);
346                     timerPlayheadDetect.removeEventListener
                        (TimerEvent.TIMER,movePlayhead_Middle);
347                     timerPlayheadMove.reset();
348                     timerPlayheadMove.removeEventListener
                        (TimerEvent.TIMER,detectPlayheadAgain_Left);
349                     timerPlayheadMove.removeEventListener
                        (TimerEvent.TIMER,detectPlayheadAgain_Right);
```

```
350                     timerPlayheadMove.removeEventListener
                        (TimerEvent.TIMER,detectPlayheadAgain_Middle);
351                     timerMoviePlay.reset();
352                     timerMoviePlay.removeEventListener(TimerEvent.TIMER,breakTime);
353             }
354             // move Playhead
355             public function movePlayhead_Left(event:TimerEvent):void {
356
357                     timerPlayheadDetect.reset();
358                     timerPlayheadDetect.removeEventListener
                        (TimerEvent.TIMER,movePlayhead_Left);
359                     timerPlayheadDetect.removeEventListener
                        (TimerEvent.TIMER,movePlayhead_Right);
360                     timerPlayheadDetect.removeEventListener
                        (TimerEvent.TIMER,movePlayhead_Middle);
361                     timerPlayheadMove.addEventListener
                        (TimerEvent.TIMER,detectPlayheadAgain_Left);
362                     timerPlayheadMove.start();
363
364                     var PlayheadPosition:Number=Playhead.x;
365                     Playhead.visible=true;
366
367                     // Seperate the stage into sections by 50 px
368                     // Playhead goes to the middle of each section every 0.5 sec.
369     var sA:Number=50*(Math.floor(GreenAverage.x/50));
370                     var sB:Number=50*(Math.floor(GreenAverage.x/50)+1);
371
372                     if (GreenAverage.x > sA && GreenAverage.x <= sB) {
373
374                             // change Playhead.x
375                             var TweenPlayheadA=new Tween(Playhead,"x",
                                None.easeIn,Playhead.x,25 + sA,0.5,true);
376
377                             // change Playhead.scaleX
378                             PlayheadScaleX=(MAX-MIN)/50;
379                             var TweenPlayheadB=new Tween(Playhead,"scaleX",
                                None.easeNone,newPlayheadScaleX,PlayheadScaleX,0.5,true);
380                             newPlayheadScaleX=PlayheadScaleX;
381
382                             // if people move
383                             if (areaArrayLength>0){
384                             N = Math.floor((1000-MIN)/1000*movieFrameLength)+FN;
385                                     KeepN=N;
386                                     movie.visible=true;
387                                     breakTimeMovie.visible=false;
388                                     movie.gotoAndStop(N);
389                                     PlayheadPosition=Playhead.x;
390                             } else {
391                                     movie.gotoAndStop(KeepN);
392                             }
393                     }
394             }
395             public function detectPlayheadAgain_Left(event:TimerEvent):void {
396                     var newChangeLength:Number=(MAX-MIN)/50;
397                     timerPlayheadDetect.addEventListener
                        (TimerEvent.TIMER,movePlayhead_Left);
398                     timerPlayheadDetect.start();
399             }
400             public function movePlayhead_Right(event:TimerEvent):void {
401
402                     timerPlayheadDetect.reset();
```

```
403                     timerPlayheadDetect.removeEventListener
                        (TimerEvent.TIMER,movePlayhead_Left);
404                     timerPlayheadDetect.removeEventListener
                        (TimerEvent.TIMER,movePlayhead_Right);
405                     timerPlayheadDetect.removeEventListener
                        (TimerEvent.TIMER,movePlayhead_Middle);
406                     timerPlayheadMove.addEventListener
                        (TimerEvent.TIMER,detectPlayheadAgain_Right);
407                     timerPlayheadMove.start();
408
409                     var PlayheadPosition:Number=Playhead.x;
410                     Playhead.visible=true;
411
412                     // Seperate the stage into sections by 50 px
413                     // Playhead goes to the middle of each section every 0.5 sec.
414                     var sA:Number=50*(Math.floor(GreenAverage.x/50));
415                     var sB:Number=50*(Math.floor(GreenAverage.x/50)+1);
416
417                     if (GreenAverage.x > sA && GreenAverage.x <= sB) {
418
419                             // change Playhead.x
420                             var TweenPlayheadA=new Tween(Playhead,"x",
                                None.easeIn,Playhead.x,25 + sA,0.5,true);
421
422                             // change Playhead.scaleX
423                             PlayheadScaleX=(MAX-MIN)/50;
424                             var TweenPlayheadB=new Tween(Playhead,"scaleX",
                                None.easeNone,newPlayheadScaleX,PlayheadScaleX,0.5,true);
425                             newPlayheadScaleX=PlayheadScaleX;
426
427                             if (MAX<1){
428                                     MAX=1;
429                             }
430                             // if people move
431                             if (areaArrayLength>0){
432                                     N = Math.floor(MAX / 1000 * movieFrameLength)+FN;
433                                     KeepN=N;
434                                     movie.visible=true;
435                                     breakTimeMovie.visible=false;
436                                     movie.gotoAndStop(N);
437                                     PlayheadPosition=Playhead.x;
438                             } else {
439                                     movie.gotoAndStop(KeepN);
440                             }
441                     }
442             }
443             public function detectPlayheadAgain_Right(event:TimerEvent):void {
444                     newPlayheadScaleX=(MAX-MIN)/50;
445                     timerPlayheadDetect.addEventListener
                        (TimerEvent.TIMER,movePlayhead_Right);
446                     timerPlayheadDetect.start();
447             }
448             public function movePlayhead_Middle(event:TimerEvent):void {
449
450                     movie.visible=true;
451                     breakTimeMovie.visible=false;
452
453                     timerPlayheadDetect.reset();
454                     timerPlayheadDetect.removeEventListener
                        (TimerEvent.TIMER,movePlayhead_Left);
455                     timerPlayheadDetect.removeEventListener
                        (TimerEvent.TIMER,movePlayhead_Right);
```

```
456                     timerPlayheadDetect.removeEventListener
                        (TimerEvent.TIMER,movePlayhead_Middle);
457                     timerPlayheadMove.addEventListener
                        (TimerEvent.TIMER,detectPlayheadAgain_Middle);
458                     timerPlayheadMove.start();
459
460                     var PlayheadPosition:Number=Playhead.x;
461                     Playhead.visible=true;
462
463                     // Seperate the stage into sections by 50 px
464                     // Playhead goes to the middle of each section every 0.5 sec.
465                     var sA:Number=50*(Math.floor(GreenAverage.x/50));
466                     var sB:Number=50*(Math.floor(GreenAverage.x/50)+1);
467
468                     if (GreenAverage.x > sA && GreenAverage.x <= sB) {
469
470                             // change Playhead.x
471                             var TweenPlayheadA=new Tween(Playhead,"x",
                                None.easeIn,Playhead.x,25 + sA,0.5,true);
472
473                             // change Playhead.scaleX
474                             PlayheadScaleX=(MAX-MIN)/50;
475                             var TweenPlayheadB=new Tween(Playhead,"scaleX",
                                None.easeNone,newPlayheadScaleX,PlayheadScaleX,0.5,true);
476                             newPlayheadScaleX=PlayheadScaleX;
477
478                             // when people are in the middle
479                             if (MIN<500 && MAX>500) {
480                                     // Right > Left
481                                     if ((MAX-500)>(500-MIN)){
482                                             // if people move
483                                             if (areaArrayLength>0){
484                                                     N=Math.floor(MAX/1000
                                                        *movieFrameLength)+FN;
485                                                     KeepN=N;
486                                                     movie.gotoAndStop(N);
487                                             } else {
488                                                     movie.gotoAndStop(KeepN);
489                                             }
490                                     }
491                                     // Right <= Left
492                                     if ((MAX-500)<=(500-MIN)){
493                                             // if people move
494                                             if (areaArrayLength>0){
495                                                     N=Math.floor((1000-MIN)/1000
                                                        *movieFrameLength)+FN;
496                                                     movie.gotoAndStop(N);
497                                                     KeepN=N;
498                                             } else {
499                                             movie.gotoAndStop(KeepN);
500                                             }
501                                     }
502                             }
503
504                             // when people are on the right side
505                             if (MAX<=500) {
506                                     // if people move
507                                     if (areaArrayLength>0){
508                                             N=Math.floor((1000-
                                                MIN)/1000*movieFrameLength)+FN;
509                                             KeepN=N;
510                                             movie.gotoAndStop(N);
511                                     } else {
```

```
512                                             movie.gotoAndStop(KeepN);
513                                     }
514                             }
515                             // when people are on the left side
516                             if (MIN>500) {
517                                     // if people move
518                                     if (areaArrayLength>0){
519                                             N=Math.floor(MAX/1000*movieFrameLength)+FN;
520                                             KeepN=N;
521                                             movie.gotoAndStop(N);
522                                     } else {
523                                             movie.gotoAndStop(KeepN);
524                                     }
525                             }
526                             PlayheadPosition=Playhead.x;
527                     }
528             }
529             public function detectPlayheadAgain_Middle(event:TimerEvent):void {
530                     newPlayheadScaleX=(MAX-MIN)/50;
531                     timerPlayheadDetect.addEventListener
                        (TimerEvent.TIMER,movePlayhead_Middle);
532                     timerPlayheadDetect.start();
533             }
534     }
535 }
```

Codes in MotionDetector.as

```
1   package {
2
3       import flash.geom.Point;
4       import flash.geom.Matrix;
5       import flash.media.Video;
6       import flash.media.Camera;
7
8       import flash.display.BitmapData;
9       import flash.display.BlendMode;
10      import flash.display.Sprite;
11      import flash.display.Bitmap;
12
13      public class MotionDetector {
14
15              // the subject of motion detection
16              private var video:Video;
17
18              // to store previous state
19              private var oldData:BitmapData;
20
21              // to store actual state
22              private var newData:BitmapData;
23
24              // to store actual state
25              private var tempImage:BitmapData;
26
27              // stepping blocks
28              private var blockSize:Number;
29
30              // detection sensitivity
31              private var sensitivity:Number;
32
33              public function MotionDetector
                (argVideo:Video,argBlockSize:Number,argSensitivity:Number) {
34
35                      video=argVideo;
36                      blockSize=argBlockSize;
37                      sensitivity=argSensitivity;
38
39                      oldData=new BitmapData(video.width,video.height,false);
40                      newData=new BitmapData(video.width,video.height,false);
41                      tempImage=new BitmapData(video.width,video.height,false);
42
43              }
44
45              public function getDifferences():Array {
46
47                      // capturing new state
48                      newData.draw(video);
49
50                      var differences:Array=new Array  ;
51
52                      // looping through points with stepping
53                      for (var px:int=0; px < newData.width; px+= blockSize) {
54
55                              for (var py:int=0; py < newData.height;
                                py+= blockSize) {
56
57                                      //getting previous and actual pixel color data
58                                      var oldPixel:uint=oldData.getPixel(px,py);
59                                      var newPixel:uint=newData.getPixel(px,py);
```

```
60
61                                      // checking difference threshold
62                                      if (Math.abs(newPixel - oldPixel) > sensitivity)
63                                      {
64                                              // if bigger than sensitivity, storing point
65                                              differences.push(new Point(px,py));
66                                      }
67                              }
68                      }
69
70                      // save previous state
71                      oldData.copyPixels(newData,newData.rect,new Point(0,0));
72                      return differences;
73
74              }
75      }
76  }
```

Bibliography

Gary Rosenzweig. ActionScript 3.0 Game Programming University. QUE CORP, 2007

Manovich, Lev. The Language of New Media. MIT Press, 2002

Rich Shupe, Zevan Rosser. Learning ActionScript 3.0: A Beginner's Guide. O'Reilly, 2007

Sommerer, Christa, L. C. Jain, Mignonneau, Laurent. The Art and Science of Interface and Interaction Design. Springer, 2008

Wilson, Stephen. Information Arts: Intersections of Art, Science, and Technology. MIT Press, 2003

Yang, Dong-Yu. Flash ActionScript 3.0. Flag Publishing Co., Ltd., 2008