Exploring Computational Art and Human-Computer Interaction in an Interactive Game Environment

by
Serena Ngai, 100687943

COMP4905 Honours Project

December, 2009

Supervised by Dr. David Mould

School of Computer Science
Carleton University
Ottawa, Ontario
Abstract

Creative applications of computer science will drive artistic expression and exploration of future innovations. Bridging the gap between technology and art is the focus of computational art and design. This project and report is inspired by the unique combination of computational art, coupled with the study of human-computer interaction in the design and development of the Bubble Breakout game.

Developed using the Processing language and OpenCV library, Bubble Breakout aims to create a more interactive and engaging game experience for users through motion detection. This report will give additional background on the technologies used, motivation and objectives, problems encountered and solutions in designing and developing the project, and final results from the project.
Acknowledgements

I would like to thank Dr. David Mould for his guidance and advice throughout the development of the project, as well as the open source community, specifically the Processing and OpenCV developers and contributors.
Table of Contents

Abstract 2
Acknowledgements 3
Table of Contents 4
List of Figures 5
1 Introduction 6
   1.1 Computational Art & Design 7
   1.2 Human-Computer Interaction 7
   1.3 Processing 8
   1.4 OpenCV Library 8
2 Motivation 9
   2.1 Objectives 10
3 Methodology 11
   3.1 Choosing Computer Vision Library for Motion Detection 11
   3.2 Using OpenCV for Motion Detection 11
   3.3 Creating Bubble Objects from Motion 12
   3.4 Collision Detection and Reaction of Elliptical Objects 14
   3.5 Refining Game Graphics 16
4 Conclusion 17
   4.1 Results 17
   4.2 Contribution to the Field 18
   4.3 Future Work 18
Bibliography 19
Appendix 20
   Project Content 20
   Launch Bubble Breakout Game 20
   Running from Java Application 21
   Running from Source Code 21
List of Figures

**Figure 1:** Blurring and adding threshold for better detection  
**Figure 2:** Bubble objects are created where motion is detected  
**Figure 3:** Creation of bubble objects using a grid during game  
**Figure 4:** Detecting collision  
**Figure 5:** Oversimplified collision reaction  
**Figure 6:** Improved solution for collision reaction  
**Figure 7:** Screenshots and demonstration of game play in action
Introduction

The area of computational art is the fusion of technology, art, and design. This combination provides for more creative, engaging, and meaningful web applications, art exhibitions, and human-computer interaction projects.

Various projects have been done using the technical aspect of computer science to build data-mining web applications interfaced with interactive visualizations and information graphics to analyze and express data that would otherwise be difficult to convey. For example, Jonathan Harris’ work on the We Feel Fine project gathers sentences containing the phrase “I feel” or “I am feeling” from blogs found on the web, to be analyzed according to emotion and demographics and displayed in an interactive web application [Harris and Kamvar, 2006]. The project uses the real-time data readily available on the web to explore candid human emotions from individuals all over the world. Its unique and creative visualizations provide a profound and meaningful ability to resonate and unite human emotions.

This project, Bubble Breakout, is inspired by computational art; it utilizes the Processing language and the OpenCV library to create an interactive game environment, allowing users to connect with the game environment by creating and controlling game objects with user movement. By using computer vision to provide computer input, the project explores human-computer interaction, and provides for a more dynamic game play.

The game was initially inspired by the arcade game Breakout. In this classic one-player video game, the player uses a paddle object to hit a ball towards a wall of blocks. Each time the ball makes contact with the wall of blocks, an area of the layer is shattered. The player earns a different number of points for each layer of bricks eliminated. The objective of the game is to break the entire wall of blocks, earning the highest score. If the ball falls to the ground, the game will end.
Instead of using a paddle object to hit the ball, Bubble Breakout will use motion detection to create bubble objects that “bounce” the ball towards the blocks. A web camera will be used to detect motion of the players’ hand; when motion is detected bubble objects are created in the game. The more motion detected in a specific area, the more bubbles are produced in that area. Each bubble object has a short lifespan of a few seconds, after which they disappear. When the ball object comes in contact with a bubble object, the ball will “bounce” off from the impact. Similar to the original game, the game ends when the ball falls to the ground.

The following sections give a brief background on the technologies used and areas that inspired the project.

1.1 Computational Art & Design

Computational art is an application of computer science that encourages programmers to utilize their technological skills to explore their artistic side. Using the computer as an expressive medium for creativity, developers design algorithms to produce graphical images [Reas and Fry, 2007].

1.2 Human-Computer Interaction

Human-computer interaction is the study of interaction between humans and computers. This area of study examines both software and hardware used, and how the input and output technologies affect interaction [Booth, 1989].
1.3 Processing

Processing is an open source programming language and development environment based on the object-oriented principles of Java. It was designed to “relate software concepts to principles of visual form, motion and interaction… to teach fundamentals of computer programming within a visual context…”. [Reas and Fry, 2007]

1.4 OpenCV Library

OpenCV is an open source library for the use of real-time computer vision for C, C# and Python languages. The particular OpenCV library used in this project was developed specifically for Processing and Java, that supports real-time capture, video file import, basic image treatment, object detection and blob detection. [Cousot and Edric, 2008]
2 Motivation

As a computer scientist with an artistic background I have always wanted to merge the creativity of art and design with the technicalities of computer science. The current curriculum at Carleton University’s School of Computer Science is largely focused on the fundamentals and theories of computer science and less focused on its application in the area of computational art and design. I believe that if we view computer science as a tool for artistic expression and exploration, we can open doors and push boundaries towards more innovative and expressive work. The bridge between development and design provides the opportunity to creatively apply the best of both worlds.

In addition, I am interested in exploring the area of human-computer interaction, another application of computer science not explored in the current curriculum. The conventional input of keyboard and mice in applications are too restrictive and is limiting to the imaginations of both the developer and user. By exploring different means of computer input such as augmented reality, motion detection, and multi-touch surfaces, applications become more inviting and engaging for the user, providing an interactive, organic, and more meaningful experience. The most creative and innovative projects can be born when we remove the limitations of the keyboard stroke and mouse-click. Projects such as the Mud Tub, developed with the Processing language uses mud as computer input and allow users to literally grasp their interactive experience: “...the Mud Tub frees the traditional computer interaction model of it’s rigidity, allowing humans to use their highly developed sense of touch, and creative thinking skills in a more natural way.” [Gerhardt, 2009]

By using computational design and letting the user feel more involved as they perform manipulations through interaction, as opposed to the traditional mouse-click, we able to enrich user experience and take applications of computer science to the next level.
2.1 Objectives

I would like to take the opportunity of this project to investigate applications of computer science in the areas of both computational art and human-computer interaction. I hope to encourage other students in the program to explore different areas of computer science application to get a full appreciation of their degree.

The main objective of this project is to get the opportunity to study other areas and applications of computer science. I will achieve this goal by accomplishing the following objectives:

1. Through the implementation of the project, I will get the opportunity to develop in a new programming language and environment.
2. Through the development of the project, I will become more familiar with the techniques of motion detection using the OpenCV library, as well as investigating the area human-computer interaction.
3. Through research and preparation for the project, I will be able to explore current and past computational art and design projects.
3 Methodology

3.1 Choosing Computer Vision Library for Motion Detection

As the Processing open source community continues to grow, various libraries have been developed to improve graphics, audio, and video capabilities. I chose to use the OpenCV open source computer vision library to handle motion detection for this project because I had previously seen projects that used the library and was interested in learning it myself.

Motion detection can also be achieved by using the built-in Processing Video library, by capturing individual video frames, and comparing pixels to search for motion [Shiffman, 2008]. Other Processing libraries developed more specifically for motion detection include: CbModel, an open source motion detection library [Bartoli, 2009]. As well as Myron, a video capture and computer vision library, used for motion detection, color tracking, glob distinction and pixel addressing [Nimoy, 2009].

3.2 Using OpenCV for Motion Detection

While incorporating the OpenCV library functions to handle motion detection, I initially encountered issues with the camera picking up too much background noise during each frame.

As shown in Fig.1, each time the screen is updated, an image is stored in memory. When the next frame is taken, the absolute difference between the current and previous image is calculated and converted to grayscale. The image is then blurred to eliminate background noise, and a threshold is applied to give a better distinction between black and white pixels. [Best, 2009] The white pixels in the resulting image show the location of the detected motion. I found that by applying a blurred effect,
and adjusting the appropriate threshold to the image, I was able to improve the accuracy of detected motion.

```java
void draw()
{
    ...
    opencv.read();
    opencv.flip(OpenCV.FLIP_HORIZONTAL);
    opencv.absDiff();
    opencv.convert(OpenCV.GRAY);
    opencv.blur(OpenCV.BLUR, 3);
    opencv.threshold(160);
    movement_img = opencv.image();
    ...
    opencv.remember(OpenCV.SOURCE, OpenCV.FLIP_HORIZONTAL);
}
```

**FIGURE 1:** Blurring and adding threshold for better detection

### 3.3 Creating Bubble Objects from Motion

From the resulting image produced by OpenCV, we are able to find the location of motion detected by searching for white pixels in the image. At first, I created bubble objects at the location of every white pixel found, but this resulted in too many bubble objects being created in densely packed areas. In order to distribute the bubble objects more evenly, I implemented a grid system to partition the screen into sections.

The entire screen of pixels is divided up into a grid. If the number of white pixels, where movement may have occurred, inside a grid section is greater than a threshold, a bubble object is created in that grid section. Only one bubble object may exist in each grid section. The figure (Fig. 2) below further illustrates this solution.
This method further eliminates background noise, prevents the bubble objects from clustering together, and distributes the bubble objects to create a more natural and dynamic feel. The images below (Fig.3) illustrate the use of the grid system during game play.
3.4 Collision Detection and Reaction of Elliptical Objects

To handle collision detection of ball and bubble objects, a collision is detected by measuring the distance between the two center points. If the distance between the two center points is less than the sum of the two radii, a collision has occurred [Christopoulos, 2006]. The figure below (Fig.4) illustrates the algorithm used to detect collision between elliptical objects in the game.

Once a collision is detected the two objects of impact must respond accordingly. I attempted several methods to handle collision reaction but found that the movement was too simplified and sluggish. For example, the figure (Fig. 5) below demonstrates one method of collision reaction that simply reversed the x or y direction if a collision was detected [Processing, 2009].

```plaintext
x_position = x_position + { x_speed * x_direction ;
y_position = y_position + { y_speed * y_direction ;

...if collision...
x_direction *= -1;
...or...
y_direction *= -1;
```
This method would work for simple up/down and left/right reactions however in our case, the ball and bubble object could collide at different angles and speeds.

In order to create a more dynamic and interactive experience, I implemented collision physics using methods from the built-in Processing `PVector` class to calculate the vector mathematics. As shown in Fig. 6, when a ball and bubble collision is detected the bubble object will disappear or “pop”, and the ball object will bounce off the bubble, changing its original velocity.

First, the minimum translation distance (MTD) vector is calculated from the location and radius of the two objects. To calculate speed of impact, the dot product of the relative velocity and normalized MTD is calculated. And finally, using conservation of momentum, a collision impulse is calculated and applied to the velocity of the ball object, causing the ball to bounce off. [Stack, 2008]

```java
PVector ball_position = new PVector (x_location, y_location);
PVector bubble_position = new PVector (aBubble.x_location, aBubble.y_location);

/*--Calculate Minimum Translation Distance vector (mtd)--*/
PVector change_in_position = PVector.sub(ball_position, bubble_position);
float change_in_position_mag = change_in_position.mag();
float min_distance = aBubble.radius + radius;
PVector mtd = PVector.mult(change_in_position,((min_distance - change_in_position_mag)/change_in_position_mag));  //Minimum Translation Distance

/*--Calculate speed of impact--*/
PVector ball_velocity = new PVector(x_velocity, y_velocity);
PVector bubble_velocity = new PVector(aBubble.x_velocity, aBubble.y_velocity);
PVector relative_velocity = PVector.sub(ball_velocity, bubble_velocity);
PVector mtd_normal = mtd;
mtd_normal.normalize();
float impact_speed = relative_velocity.dot(mtd_normal);
if(impact_speed > 0)  {  return;  } //If ball is already moving, no change in velocity needed
float coefficient_of_restitution = 0.9;
float i = -(1+ coefficient_of_restitution) * impact_speed;
PVector impulse = PVector.mult(mtd, i);
x_velocity += impulse.x;
y_velocity += impulse.y;
bubble.lifespan = 0;    //Destroy bubble
```

FIGURE 6: Improved solution for collision reaction
This method also allows for more dynamics in object movement, permitting objects to lose and gain speed from gravity and impact.

3.5 Refining Game Graphics

Originally I had intended on creating all game graphics using only Processing code, however I wanted to take advantage of the motion detection aspect of the game by adding a sense of augmented reality with more realistic game objects. Although Processing does provide 3D functions and other libraries to enhance graphics, I ultimately decided to design and create all game objects myself using graphic editing software. The glossy feel of the colorful blocks, polished ball, and transparent bubbles enhance the game play by providing a tangible feel to the game objects.
4 Conclusion

After four months of researching, designing the project and developing the Bubble Breakout game, the following concluding section explains the results accomplished and future work to be investigated.

4.1 Results

As demonstrated in Fig.7 below, all the objectives outlined have been achieved. Through preparation for the project and report, I had the opportunity to learn more about various projects and areas of applications of computer science. As the developer of the project I was able to fully implement all aspects of the game, learning the Processing language and implementing motion detection using the OpenCV library.

Figure 7: Screenshots and demonstration of game play in action
4.2 Contribution to the Field

Although this project does not contribute revolutionary research to the field of computer science, this project would certainly be a great addition to the open source Processing community. The field of computational art and the Processing language is still fairly new and not widely adopted yet; thus this project could be used to promote greater awareness of these creative areas of computer science. In addition, the project would be helpful for other developers interested in developing in the Processing language with the OpenCV library.

4.3 Future Work

To further explore human-computer interaction, the Bubble Breakout game can be modified to use various forms of computer input. For example: to create bubble objects, instead of detecting arbitrary motion, the game can be adapted to detect the motion of a specific color or sensor on a physical device. Similarly, the game can also be modified to detect touch input on a touch-screen surface.

Other areas that could be further explored in this project include game aspects, such as level of difficulty, enhancement of game graphics and addition of sound. I would like to see more work done in the direction of computational art, using more code to create graphical elements.
Bibliography


Appendix

Project Content

In accordance to the Carleton University School of Computer Science Honours Project Guide for Fall 2009, project deliverables include:

- Bubble Breakout source code and Java applications
- Final Project Report in PDF format
- Project abstract
- 640x480px image representing the project

Launch Bubble Breakout Game

There are two options provided to run the Bubble Breakout game: running the exported Java application provided, or running the source code directly from a Processing development environment.

To run the game with either option: Java 1.5 and a web camera must be installed on the operating system. More info on operating systems, Java versions and camera compatibilities can be found here:

- [http://processing.org/reference/troubleshooting/#supported](http://processing.org/reference/troubleshooting/#supported)
- [http://opencv.willowgarage.com/wiki/FullOpenCVWiki#Welcome.2BAC8-OS.OSSpecificStuff](http://opencv.willowgarage.com/wiki/FullOpenCVWiki#Welcome.2BAC8-OS.OSSpecificStuff)
Running from Java Application

Java applications of the game are provided in the project for Windows, Mac OS X, and Linux platforms.

Note: There seems to be some issues with running the Java application on some systems due to a known problem with exporting external video libraries in the Processing environment. If the Java application does not run, try running from source code.

Running from Source Code

If running the source code, the Processing development environment, and OpenCV Processing and Java library must be installed and configured:

- Processing environment: [http://processing.org/download](http://processing.org/download)
- OpenCV Processing and Java Library: [http://ubaa.net/shared/processing/opencv](http://ubaa.net/shared/processing/opencv)