# Multimodal Interfaces

## Multimodal Bejeweled

BAERISWYL Roman
roman.baeriswyl@unifr.ch

ROSSIER Blaise
blaise.rossier@unifr.ch

## ABSTRACT
This is a project for the course Multimodal Interfaces, JMCS, University of Fribourg, Switzerland. The widely known puzzle game Bejeweled uses traditionally the mouse as an input modality. This project has the goal to augment the number of modalities, like gesture and voice. To play the game, multiple modality combinations will be possible and tested against each other.

## Keywords
Multimodal, Bejeweled, Speech Recognition, Gesture Recognition, Leap Motion

## 1. INTRODUCTION
Human-Computer Interaction is done via interfaces. Computers are always evolving and becoming more powerful. However, the interaction with them hasn't changed much over time. Different Graphical User Interfaces (GUI's) evolve, but follow the same paradigm.

Most software uses **Windows** as a primary container. Those windows are displayed and visible on the screen. Users can interact with them by using a **Pointing device**. The mouse is the most popular one. **Menus** are used to organize and facilitate command selections. The mouse and the keyboard are mainly use to interact with menus. Because sometimes a picture is more expressly than words, **Icons** have been introduced into GUI's. They are little pictograms that can be seen as another visual command selector. Taken together, this is the **WIMP** interface [6].

This interface is already over 40 years old but still widely used. Computers evolve but the input devices stay the same. Keyboard and mouse are nowadays still the most important interaction tools [2].

Still, there are plenty of ways to communicate with a computer; those ways are called modalities. Humans use several modalities to communicate together. This could also be the case with Human Computer Interaction (HCI), but traditional devices stay the norm. Maybe in a near future?

The aim of this project is to take an existing game, a game that uses traditional modalities, and replace those old robust modalities with new, more advanced ones. The project should focus on the modalities, not on the game logic or graphics.

The next chapter will present the chosen game. Then the old and new modalities are presented and discussed. A next section explains briefly how the game is designed to support those new modalities. The new version is then tested with users, because they are a keystone in software development. They should be taken into account as soon as possible. Finally, the last part discusses the results and lessons learnt with this project.

## 2. THE GAME
Because the aim of this project is to work with new modalities and not with the logic behind an application, an existing game implementation is taken and adapted. The chosen game is a widely known one, called **Bejeweled**. There are a lot of different implementations available on the net, the choice was made for a Java implementation [7], because of its popularity.

### 2.1 Presentation
Bejeweled is a tile-matching puzzle [4]. Figure 1 shows the game board.

**Figure 1: Game board**

The aim of this game is to swap two adjacent tiles in order to form a chain of at least three similar tiles. Chains can

be horizontally or vertically created. When such a chain is created, it will disappear, score points and new tiles will appear. Cascades are also possible, meaning that new appearing tiles create chains themselves.

Figure 2 shows the evolution of the game board if a chain of at least three tile is created.

**Figure 2: When a chain is created**

The gamer scores points as illustrated in Listing 1. Two different variables are considered. First one is the *combo* which memorizes the longest created chain. Second one is the score depending on chain size.

```
combo += chain.size();
score += chain.size()*10;
```

**Listing 1: Score**

In the current Bejeweled version, tiles are selected using the mouse (yellow square on Figure 2). The user selects a tile with the mouse (click) and can click then on an adjacent one to swap the two tiles. If a tile is clicked that is not adjacent to the first selected one, the last tile is marked as first one.

## 2.2 How to play

The original Bejweled version uses one modality to select and swap tiles. This modality is the mouse. To play the game, two actions are needed:

1. select a first tile (**T1**);
2. select a second (adjacent) tile (**T2**) to swap with the first one.

Those two actions together form the **task of playing the game**. This project focuses on this task; for game window control (minimize/close/move) and game state control (start new game) the default modalities mouse and keyboard will continued to be used.

# 3. MODALITIES

Modalities are ways of communicating with an application. This section discusses the chosen modalities for this project from the system side as well as the human side.

## 3.1 New modalities

Considering the two actions to swap **T1** and **T2**, three new modalities are introduced: **Keyboard**, **Voice** and **Gesture**.

To do the two actions mentioned earlier, two groups of modalities **M1** and **M2** are introduced.

- **T1** is selected with **M1** and swapped with **T2** using **M1** (one modality);
- **T1** is selected with **M1** and swapped with **T2** using **M2** (two modalities).

where **M1** and **M2** can be:

- **M1**: Mouse XOR Gesture
- **M2**: Keyboard OR Voice

Group **M1** can do both actions, or can be combined with one of **M2** to do the second action.

In other words, there are 6 modality combinations possible, listed here (Table 1).

| | **M1 for action 1** | **M2 for action 2** |
|---|---|---|
| 1 | Mouse | Mouse |
| 2 | Mouse | Keyboard |
| 3 | Mouse | Voice |
| 4 | Gesture | Gesture |
| 5 | Gesture | Keyboard |
| 6 | Gesture | Voice |

**Table 1: 6 interaction possibilities**

Mouse or gesture are suitable to select a first tile (**T1**). Then keyboard or voice could decide with which adjacent tile the selected one has to be swapped.

## 3.2 New Devices and Libraries

The game is implemented in Java using Swing as graphical library. Events are managed via the AWT library. Mouse and keyboard events are thus natively managed by this library.

A traditional computer can work natively with voice signal as input. There is a microphone or at least a microphone entry. Working with gesture is a bit different. There is often a webcam that could be used to record movement, but fortunately there exist special devices dedicated to gesture recognition.

One of those devices is *Leap Motion*. It is a small infrared sensor that can recognize hand and finger movements. Figure 3 illustrates how this new (2010) [5] device looks like.

**Figure 3: Leap Motion device [1]**

For developers, there is a SDK and documentation available for several languages, including Java.

After reading some of the documentation and following demo examples, gesture recognition could be integrated smoothly in this project.

For voice recognition, there exist many libraries capable of. After trying out different approaches, a library called *Sphinx* is used. After adapting a simple demo example, simple vocal commands can be recognized by the system.

The new modalities have to be integrated into the game implementation; all those modalities should collaborate or at least coexist peacefully.

## 3.3 CASE and CARE

When discussing modalities, two different models can be considered. The first one permits to understand the different communication types. This model considers modalities from a system side point of view and contains four different possibilities of multimodal communication:

- **C**oncurent: two independent modalities can be used in parallel for *two distinct tasks*;
- **A**lternate: two combined modalities can be used sequentially for *one task*;
- **S**ynergistic: two combined modalities can be used in parallel for *one task*;
- **E**xclusive: two independent modalities can be used sequentially for *two distinct tasks*.

This is the **CASE** model [3].

According to the 6 possibilities to accomplish the task of swapping two tiles together; this project uses following aspects of the **CASE** model.

- **A**lternate: **T1** is selected with the *mouse*, **T2** is swapped using **M2**;
- **S**ynergistic: **T1** is selected with *gesture*, **T2** is swapped using **M2**;

The difference between the two types of communication is that, with the mouse, the action of selecting a tile ends with a click on it. With the gesture, an user needs to maintain the selection until the tile is swapped.

Second model considers modalities from the human side point of view. This model explains how to use modalities. There are also four different possibilities.

- **C**omplementary: two modalities *have to be* used for one task
- **A**ssignment: only one modality *can be* used for one given task
- **R**edundancy: two modalities *can be* used together for one task, but one is sufficient
- **E**quivalence: two modalities *can be* used for one task

According to this **CARE** model [3], the project considers following aspects.

- **A**ssignment: game windows control, only the mouse can move or resize the window [1]
- **R**edundancy: There are 6 different possibilities two swap two tiles!
- **E**quivalence: There are 4 different possibilities two swap two tiles with two modalities!

The upcoming section is dedicated to the software architecture and choices made to implement the new modalities.

# 4. SOFTWARE ARCHITECTURE

Different modalities have to be implemented and interact together. As already mentioned, the manufacturer of the Leap Motion device provides a SDK for it. For voice recognition a library called Sphinx is used.

## 4.1 Implementation

This project is written in Java entirely. The new modalities are implemented as Listener.

### *4.1.1 Keyboard*

The implementation for the keyboard is trivial, it's a simple KeyListener for the buttons left, right, up and down.

[1] assuming that the desktop environment uses a floating windows manager and not a tiling one!

#### 4.1.2 *Gesture*

The Leap Motion SDK provides gesture recognition. Hand and finger positions are calculated dynamically. Specific gestures like swipes, circles or screen tapping are also recognized.

The average finger position defines the tile an user is pointing at. It is highlighted with a red square (Figure 4) on the gameboard. The yellow one is dedicated for the mouse!

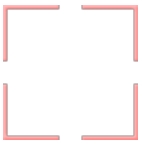

**Figure 4: Red Square to indicate current finger focus position**

The chosen implementation doesn't allow to store a selection like it is done with a mouse click. Instead, an user has to maintain the selection until the two tiles **T1** and **T2** are swapped.

The gesture module is then implemented like a MouseListener. But instead of providing the xy-coordinates of a mouse pointer, it provides the xy-coordinates of the average finger position.

Since the tiles need also to be swapped with gesture only, the swipe gesture is chosen to accomplish this. In the direction an user swipes his hand, the current focused tile will be swapped with the adjacent one.

#### 4.1.3 *Voice*

Sphinx is a library for live voice recognition. For this project, a simple vocabulary of four words is used: left, right, up and down.

The module is added like a KeyListener. As soon as the system recognizes one of these vocal commands, it launches the according button pressed action. The system will then swap the current focused or selected tile in the direction the command indicates. This modality is used as the one using keyboard!

## 5. TESTING

With the new implemented modalities, testing has to be done. Users are important in this step. This section explains how the application has been tested with them and presents the obtained results.

### 5.1 Two alternatives

The equivalent/redundant modalities of the **CARE** model can be compared; solutions using one modality is compared with solutions using two modalities for the task of swapping two tiles.

The two tested alternatives (**A1** and **A2**) are the following.

1. **T1** is selected with **M1** and swapped using **M1** (one modality);
2. **T1** is selected with **M1** and swapped using **M2** (two modalities).

where **M1** and **M2** can be:

- **M1**: Mouse XOR Gesture
- **M2**: Keyboard OR Voice

### 5.2 Testable Hypothesis

For this project following hypothesis is formulated:

$$\mathbf{M1} + \mathbf{M2} > \mathbf{M1} + \mathbf{M1}$$

Two modalities for a given task (two actions) are more intuitive, usable, effective than only one modality.

### 5.3 Evaluation Plan

The game is tested with 4 users forming two group of 2 users. Each group test all 6 possible combinations of modalities (within group). First group begins with alternative **A1** and ends with alternative **A2**. The second group does the reverse.

Each user tests a given combination during 60 seconds.

Following variables are considered for this experiment:

- **Independent**: combination of modalities
- **Dependent**: score, appreciation, recognition errors (manually observed)

An recognition error is an error if a given command has to be repeated because it is not understood or wrongly interpreted by the system.

### 5.4 Results

Following results have been collected (Figure 5).

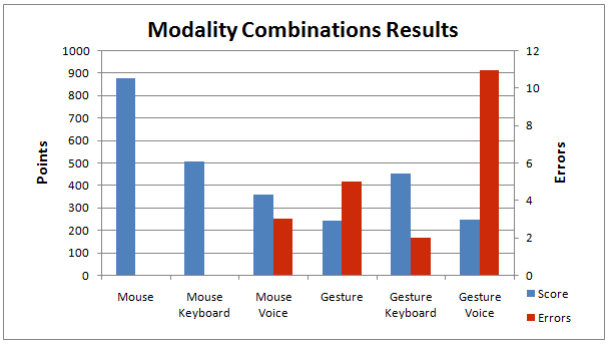


**Figure 5: Test results: average scores and errors for different modality combinations**

The traditional modalities mouse and keyboard scored the most points, by having zero recognition errors. The second most points could be achieved by using one traditional modality and one of the new ones (mouse & voice, gesture & keyboard). For both of these modality combinations, there were several recognition errors.

By using only gesture, there were even more errors. The worst combination in terms of score and errors, was obtained by using the two modalities gesture and voice together. The errors of both of these modalities combined did raise the error rate significantly. This high error rate did impact the score accordingly; the lowest score for all combinations.

In terms of user appreciation, the combinations with one traditional and one new modality have been enjoyed the most. They may not be flawless like the traditional modalities, but are a new, refreshing way of interaction. Gesture alone or in combination with voice was too error prone and therefore not appreciated.

The testable hypothesis said that combining two modalities will improve performance and satisfaction compared to alternatives using only one modalities. The answer is not obvious. One modality taken alone could gives good results (mouse) or bad ones (gesture). The same observation can be done with two modalities. Gesture and keyboard together could give good results but gesture used with voice together is not very powerful neither appreciated.

# 6. DISCUSSION

Looking at the number of recognition error for the different modality combinations, the modules for voice and gesture can and should be improved further.

The voice recognition implemented is very sensitive. Noise and words, that are not game commands [2], are problematic for the voice recognition module. Often they get mistakenly interpreted as a game command. By applying filters and improving the vocabulary of the speech recognition system, this modality could be improved. Also the recognition speed is not optimal yet. By using this modality, an user is already penalized in terms of points per time.

The developed system for gesture recognition has several issues. Firstly, recognition is continuous: to focus a tile, the hand and finger position has to be maintained. This can get uncomfortable after some time. The other problem lies in the swiping gesture used to swap tiles. By performing a swipe, the other hand which focuses a tile, may displace itself by a small amount. As a result, the focused tile my change, leading to an unintended action.

Especially for the gesture modality, there is much training needed for a given user to perform well with it. Most users have years of training with the mouse, but have for most never interacted with a Leap Motion device. With more practice, the scores for modality combinations with gesture should improve.

Also user appreciation is here more important than score, this is a game and should be enjoyed. Additionally, score depends of others, mostly uncontrolled parameters, like chain reactions as an example!

[2]Up|down|left|right

# 7. CONCLUSION

In a game where the goal is to score the most points in the fastest way, the traditional modalities mouse and keyboard still seem to work best.

However, the combination with new modalities can be enjoyable and was appreciated by most users. The number of system recognition errors is still high for the modalities voice and gesture. By improving those modules, the error rate should decrease, score and user appreciation increase accordingly.

There are no good or bad modalities, but good or bad choices! The context determines which modality is the most suitable for a given task.

# 8. DISTRIBUTION OF WORK

Table 2 shows the distribution of work.

| **Blaise** | **Roman** | **Both** |
|---|---|---|
| Speech Recognition | Gesture Recognition | Filming |
| Design | Fusion | Testing |
| Video Montage | Results | Report |
| | | Presentation |

**Table 2: Distribution of work**

# 9. REFERENCES


[1] Image. Leap Motion, May 2014. `http://i.pcworld.fr/1260021-leap-motion.png`.

[2] D. Lalanne. Multimodal Interaction Introduction, University of Fribourg, Mars 2014.

[3] D. Lalanne. Multimodal Interfaces, fusion, fission and systems architecture, University of Fribourg, Mars 2014.

[4] Wikipedia. Bejeweled, May 2014. `http://en.wikipedia.org/wiki/Bejeweled`.

[5] Wikipedia. Leap Motion, May 2014. `http://en.wikipedia.org/wiki/Leap_Motion`.

[6] Wikipedia. WIMP, May 2014. `http://en.wikipedia.org/wiki/WIMP_(computing)`.

[7] xihan. Bejeweled implemented in Java, January 2012. `https://github.com/xihan/Bejeweled`.