Using a Multi-Touch Surface to Create Conversation Between Two Different Languages

Undergraduate Honours Thesis
Faculty of Science (Computing Science)
University of Ontario Institute of Technology

Tony Tran

Supervisor:
Dr. Christopher Collins

April 13th, 2011

Abstract

As multi-touch surfaces begin to become a mainstream feature in many electronics, many innovations are being made for such a feature. In this thesis, the feature we are implementing for multi-touch surfaces is to aid in the learning of a different language. Through this we can eliminate the barrier which disallows two different languages to communicate with each other. To do this we include a game element to the learning objective. The first two prototypes that were created use the classic game “Pong” to allow the learners to experience fun while learning. The way these two prototypes deliver the learning elements are different, where as one simply ask for the approval of the player, while the other asks for the player to construct the sentence themselves. Afterwards the player is allowed to proceed with their game of “Pong” and the process repeats. The third prototype is a simple word game where the users match the words with their direct translation. With this, the goal of bridging two different languages to have a conversation should be met with this thesis.

Table of Contents

1. Introduction

The idea behind this thesis was driven by the interactivity that is possible with multi-touch surfaces. In this thesis we will be examining and developing for the multi-touch capabilities of the SMART table, which allows for interaction amongst multiple users. Because of this interactivity, it allows for collaboration while using the SMART table. We hope that this thesis will make full use of this, and can inspire future works to follow in the same ideal. With the plentiful amount of different languages that exists today, there also exist a lot of barriers which stop different languages to have a conversation. This makes dialogue between two different languages difficult to occur, causing the two speakers to never understand each other. The only way around this, is to make each speaker familiar with the others language, hence the idea of this thesis came to be.

1.1 Problem Statement and Motivation

This thesis presents many problems that must be overcome. One of the problems is dealing with the design of the program. As stated before, the SMART table allows for collaboration amongst its users, however that solely relies on the way the program is designed. Another problem is to decide how to incorporate the learning elements with the game elements. How are we going to balance the two so that the users are motivated to continue learning, and at the same time ensure that they are learning at all? Finally, the main problem regarding the purpose behind this entire thesis must be answered. How are we going to lead these two speakers with this program to the point where a dialogue is possible between them?

1.2 Goals

As stated before, this thesis aims to bridge the barriers between languages through the collaboration that is created while using the SMART Table. To do this, the program that is being created must incorporate the ideal of users interacting with each other along with the table. In a sense we have to build this program to promote teamwork, however at the same time give the competitive spirit of a game. Also, because this is incorporating a game, we have to design the program that is fun that the user is motivated to continue playing. The difficulty of the game should be measured carefully. If we are unable to lead the speaker from a novice level at the native language to someone who can understand a fair amount, the game becomes pointless defeating the purpose of this thesis. As before, we are going to be using the basis of the Pong game in order to deliver both the competitive and enjoyment feeling to the user. In the end, we hope to create something that encompasses all these traits in order to be considered successful.

Therefore are goals, as stated before are too:

- Combine the learning experience with the enjoyment of a game
- To take full use of the collaboration that a SMART table can provide
- Ensure the user has learnt something from their success and failures leading to the ability to understand the native language

1.3Users/Stake holders

This thesis is dependent on the people that use it. If the players who use this program do not learn anything from it, than we have failed. The players will also be playing in pairs, because they are trying to learn each other's language. These conversation partners are what drive the purpose of this thesis. Hence we must create our prototypes around their needs, therefore making them our primary stake holder. Their aim is to become familiar with a certain language, which requires them to learn common words or phrases which is the very aim of this thesis. We also wish to make the learning experience enjoyable and motivating the player to continue learning as well. This also leads to institutions using this method to teach different languages. They become our secondary stake holder, because they are integrating our work to their teachings, however they are not the ones directly using it. With our stake holders identified, we are able to proceed on and design our thesis to suit their requirements.

1.4 Thesis Overview

In Section 2, we will be looking at the design concepts that came up and inspired the production of the prototypes. This is going to range from the initial blue prints, to what was actually done, and the reasons for the various changes. Each prototype was aimed at solving certain problems to help us reach our goals, therefore we shall individually look at what lead to the design choices. Section 3 will cover the implementation of the program, where we analyze the choices that I had made while coding this thesis, the data structure I used, and the structure of the code. The prototypes maybe similar in some ways, however they differ enough that we should individually examine them. In Section 4 we will go over the feedback this thesis work has

gotten from the users we had testing the program. Finally, after going through the feedback we will look at what goals were achieved with this project, a summary of this thesis, and any future work that could be based off this design.

2. Design Approach

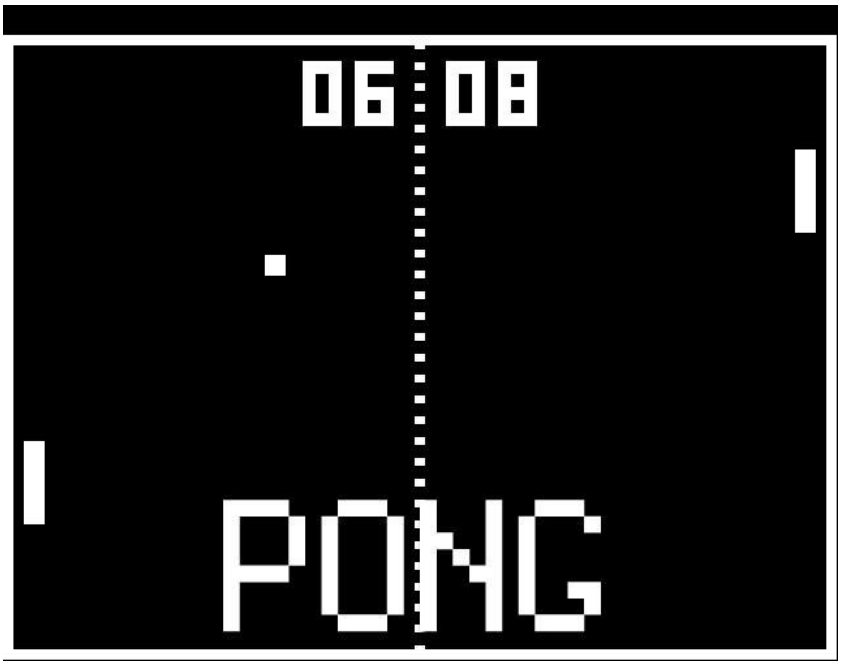

Figure 1: The original Pong Game being played. [PO72]
Image obtained from (http://blogs.rockyview.ab.ca/wp-content/uploads/2011/04/pong.jpg)

Pong[PO72] is a simulation of the real life game table tennis (as shown in Figure 1). Both players are given a paddle, and are in charge of keeping a ball in play which is done by hitting the ball with the paddle. If one player were to miss contacting the ball with the paddle, they would concede a point to the other player. However in this thesis, we must add a learning element to this game, the element which teaches the user the language of the other. In order to do this we concluded that the best way would to create an obstacle before the player is able to move their respective paddle. This will force the players to go through the learning objective first, and if successful, are rewarded to continuing the play of the game. Also the more that the player succeeds, the more difficult the objectives become allowing the player to become strong in the language. The first two prototypes use this process to achieve their goal, however their methods of presenting the objectives are different. The third prototype better integrates the

game play and learning objectives. It does not use the Pong base game to provide the game element, but instead uses simple matching word game. All prototypes are able to standalone as their own program, but all spawned from the same motivation to bridge languages.

2.1 Background

Figure 2: Some of the software that shipped with the system, all are built around the touch interface and to promote collaboration[SmTa]

The Smart Table is supplied with its own games that support learning as shown in Figure 2. When we took a look at these small games, we found that all these games are built to support collaboration between all the users. One of the games was a simple addition game, asking players to touch the screen as many fingers that the sum of the question asks for.

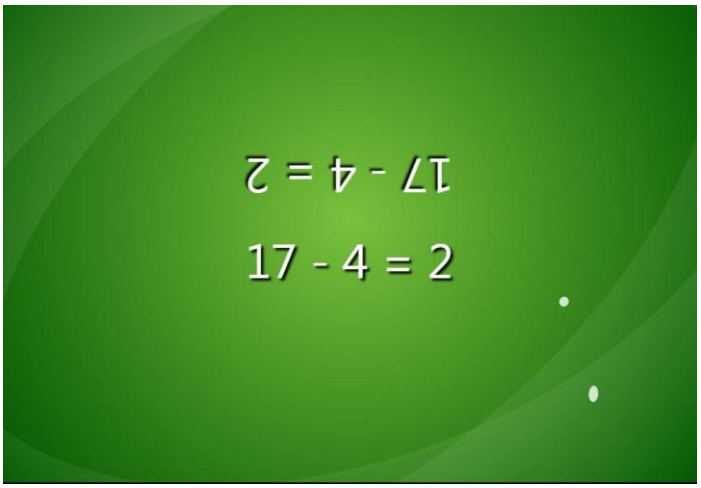

Figure 3: Here 13 fingers are needed to touch the screen, therefore the player requires assistance from those playing along[SmTa]

Depending on how many people are present, the table will often ask for more than one person's limited amount of fingers to touch the table (example being in shown in Figure 3. Because of this, collaboration is needed amongst the player in order to meet the required number of touches. This collaboration is what we want present in our thesis, to promote learning between both users and allowing them to interact with each other to, not only learn from the program, but from each other as well.

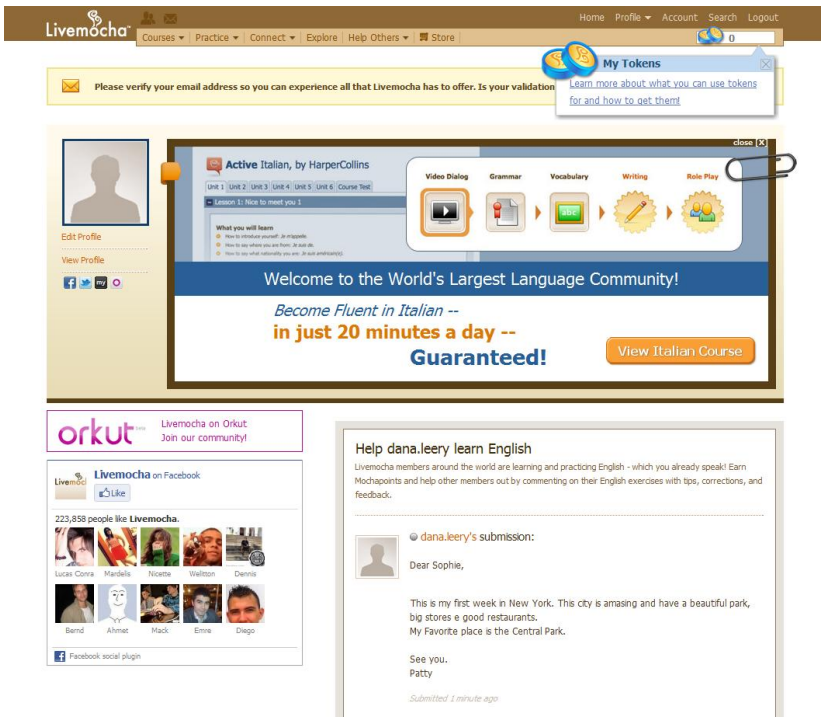

Figure 4:Livemocha's main screen, a service that allows users to learn new languages through their tutorials and such.[LiMo07]

Another thing to observe is other language games. From what we've researched, most language learning games require the user to play against an A.I. Livemocha is a site that offers users the chance to learn another language (main site shown in Figure 4) [LiMo07]. This is done through the tutorial material they offer, they many practice methods, and a chance to chat with other people learning the same language.

Figure 5: One of the mini-games that are offered by Livemocha, a simple matching game. This one being based around colours[LiMo07]

They also feature practice mini-games, which allows the user play simple games that at the same time allow players to practice what they have learned(Figure 5). Upon writing this thesis, the only two games were simply matching words to the translated meaning to another word.

Figure 6: The results of the game, which felt like a quiz[LiMo07]

While simple, we feel that it does not offer an actual game play element, and is simply just a quiz with a designated time limit, simply because the punishment is just scoring lower(shown in Figure 6). Other language games that were found were similar to what is offered by Livemocha, where the 'game play' is simple and does not allow for a lot of interaction.

Therefore, after looking at these examples, the prototypes that are designed will need to have the collaboration element that the Smart Table games have. Not only that but they must also contain actual game play elements that will motivate the player to continue learning. Finally, the game must also teach, and ensure that the player learns and grows stronger in the language.

2.2 Prototype 1(Pong)

Being the first prototype, we first had to think of a way to create the objective so that it is quick enough to be played along Pong but still providing the learning aspect. If the objective takes too long to accomplish, the flow of Pong will be taken away completely. Therefore our first objective was to going to be a quick action.

Figure 7: The design for Pong, where two sentences appear for the player to choose

So then came the idea of having the players choose a valid sentence for the conversation. As shown in Figure 7, the player will be given a choice between sentences that are structured incorrectly and a sentence which is structured correctly. When in play, the player must choose the correctly structured sentence.

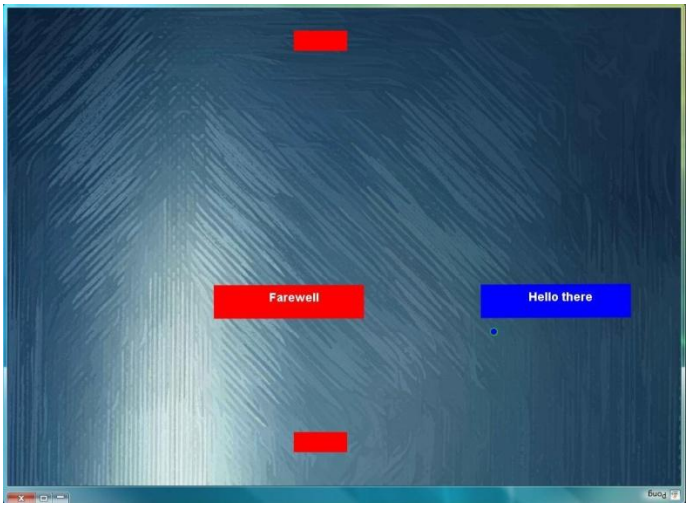

Figure 8: User is being informed that the wrong choice has been made

Only then can the player move the paddle to hit the ball back and continue the game. If the player chooses the wrong sentence, he would be informed that he has chosen the wrong sentence(as seen on Figure 8), and will not be able to move his paddle until he has chosen the right sentence. By then, the player the player will understand the mistake he has made than, and will learn from their mistake. After making the correct choice, the other player is now presented with the Objectives. These Objectives however are the reply to the sentence that the other player has chosen, therefore almost simulating a conversation. As the players continue through the game, they will find that they are taking turns having a conversation with questions and replies, however in their respective language that they are trying to learn.

There are times when the player is able to go through the objective through trial and error. For testing reasons, Pong only offers one wrong sentence, and one correct sentence, therefore giving the player a 50% chance of getting the question right through guessing. Ideally there would be more choices to slow the player down, and perhaps a freeze on the player's action to penalize them for making the incorrect choice. However, the fact that the player is still able to guess their answer is demeaning and defeats the purpose of this thesis.

2.3 Prototype 2 (PongBeta)

After creating the first prototype, the feeling of the player being able to go through the objective via trial and error was unsettling. The purpose of this thesis was to teach the player the respective language they were aiming for. By guessing at the correct answer we felt they had learned nothing. Therefore the idea for the second prototype was to eliminate that factor.

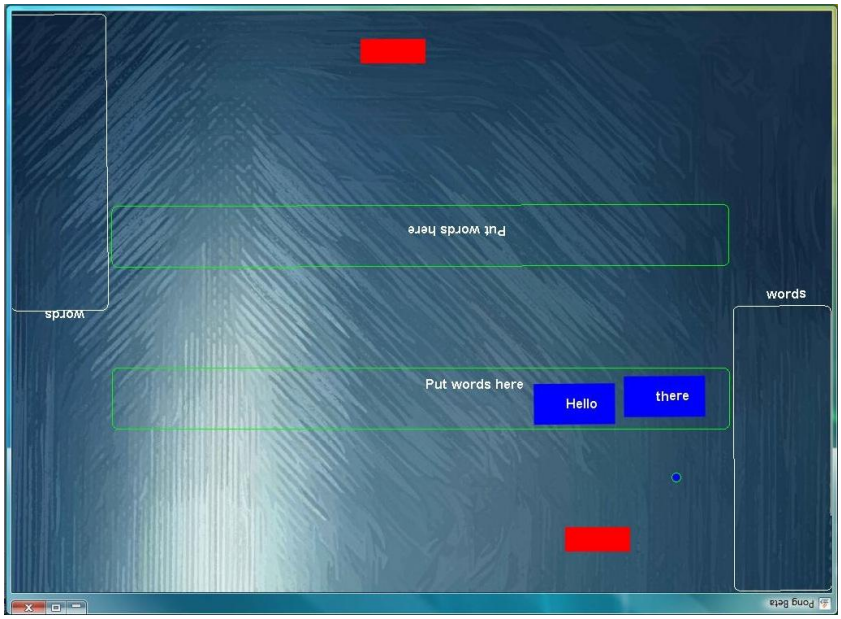

Figure 9: PongBeta in play. The player has correctly placed the words in the right order and is moving the paddle

This gave way to the idea of the player being able to construct their own sentence. The way this worked was that in one corner, all the words would appear in different blocks. The player would have to place these blocks within a designated area and in the correct order to form a proper sentence as shown in Figure 9. Only then, can the player move their paddle to continue their game of pong. To further take away from allowing the player to guess, the game will give no indication whether the player has formed a correct sentence. This way, when the player forms the sentence, they will know for sure if they have formed the correct sentence. Otherwise, the paddle will not move at all and the player must figure out their mistake. This

may take away from the flow of the game, because it takes much longer to form a sentence than it does to simply pick one. However to compensate for that, the velocity of the ball was reduced in order to give the player time to create their sentence.

2.4 Prototype 3 (Dueldisk)

Unlike Prototypes 1 & 2, this prototype does not rely on the game element of Pong. Instead it drew inspiration from real life games such as Dodge ball. The goal of the game is to successfully defend you from offensive assaults from the other player, and at the same time present the same offensive front as the other player.

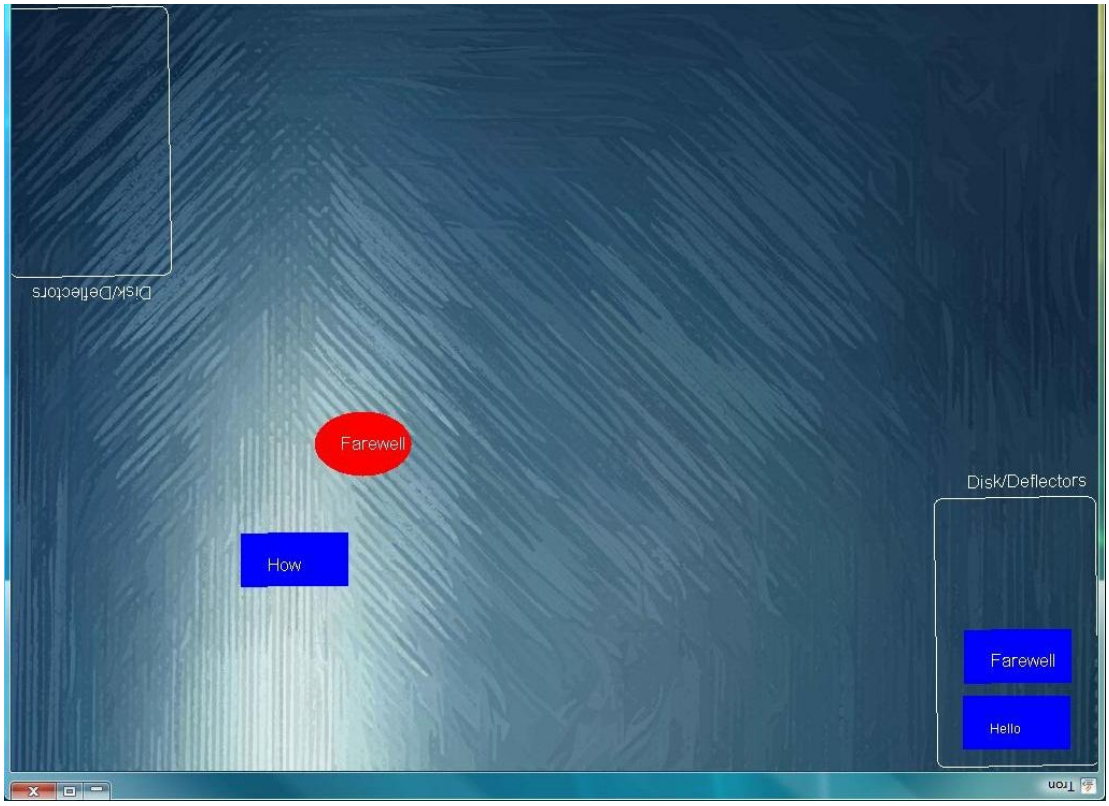

Figure 10:Dueldisk being played, where the player has thrown a disk, and another player is defending themself. Unfortunately the wrong word was chosen.

This starts with a player choosing a disk from selection. This disk will have a single word on it, which is in the language that the other player is unfamiliar with. The player with the chosen disk

then throws that disk at the other player like in Figure 10. On the other side, the player having the disk being thrown at has to make a quick decision to choose a correct deflector in order to successful defend themselves. The deflectors have words on them as well, however they are the in the language the player is familiar with. Therefore, in order to defend themselves, they must translate the word on the disk, to the word of their language in order to know which deflector to choose. As the game proceeds harder words appear, as well as the maximum velocity of the disk is increased, making it harder for players to defend themselves. In terms of the flow of the game, this prototype should be much faster pace than the others. As soon as the other player is able to deflect or fails, new disks with fresh words pop up right away and the process is repeated. At the same time, the learning objective is simplified. Instead of learning entire sentences, we are teaching them single words at a time now allowing for much slower learning. This prototype is perhaps one that is used to introduce players to the beginning concepts of another language.

2.5 Aesthetics

A lot of today's games that are played tend to have a lot of graphics to stimulate the player's experience more. However for this thesis, the time line did not permit us to spend a large amount of time in this area. In the end however, we were able to provide the skeleton of could be built on. Objects that are drawn on the screen are simple circles and rectangles that can be drawn with Java's graphics. We inserted a background to give the player something more pleasing to look at than a plain white background. We also drew boxes and labels to help the player understand what different objects are in the game, allowing them to focus more on the game play and learning mechanics

3. Implementation

For this thesis, we developed the prototypes on the Smart Table. Although the table does come with its own programs and developer kit, it does not allow for more complicated programs and most of the already developed software is aimed at a young age. Therefore in order to develop for the Smart Table, we had to use the program which provides our thesis with the touch coordinates.

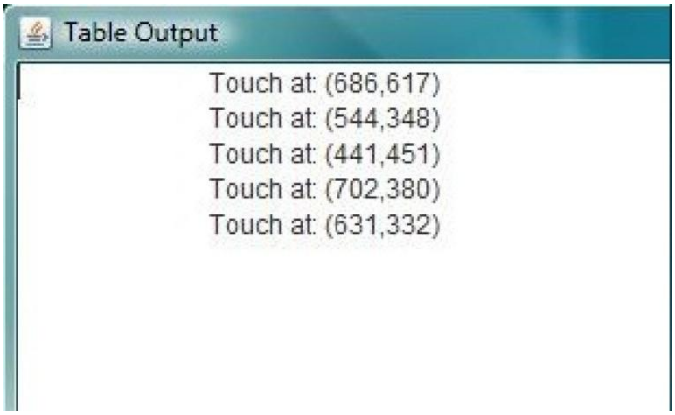

Figure 11: A simple program that reads the XML file that is generated when a touch occurs [TaCo01]
Provided by Dr. Collins and his work on Vislink (VisLi07]

In order to do this, we run a separate program which takes the values that the Smart Table has recorded for touches, and convert them to a XML format from which our program can read from and use. This process was provided by Dr. Collins, Mark Hancock and Luc Vlaming and their work on Vislink(VisLi07] which also requires the need for this tool kit to provide touches. Because Vislink was programmed in Java, we choose to program this thesis in Java as well for simplicity sake. This is because we can draw their method of using touches for their own usage, and apply it to this thesis. In the end, most of the touch recognition code was learned by

through these examples which were all programmed in Java. With these tools, we were able to create the 3 prototypes

3.1 Data Structure

As stated before, the purpose of this thesis was to bridge languages in order to create conversation between 2 people. With the many languages that exist, this thesis should be able to support all of them.

```
<?xml version="1.0" encoding="UTF-8"?>
<Example>
    <Sentence>Hello</Sentence>
    <Sentence>Hi</Sentence>
    <Sentence>Good Bye</Sentence>
    <Sentence>Farewell</Sentence>
    <Sentence>How do you do?</Sentence>
    <Sentence>How are you?</Sentence>
    <Sentence>Going is it how?</Sentence>
    <Sentence>You do how?</Sentence>
    <Sentence>I'm fantastic</Sentence>
    <Sentence>I'm feeling good</Sentence>
    <Sentence>Good is feeling I'm</Sentence>
    <Sentence>Fantasitic am I are</Sentence>
    <Sentence>Thats good to hear</Sentence>
    <Sentence>Thats great</Sentence>
    <Sentence>Hearing to that good</Sentence>
    <Sentence>Great is now that</Sentence>
    <Sentence>Weather is crazy out there, isn't it?</Sentence>
    <Sentence>Isn't the weather crazy out there? </Sentence>
    <Sentence>Weather crazy out isn't out the?</Sentence>
    <Sentence>Absolutely out crazy weather out isn't?</Sentence>
    <Sentence>Yeah, caused a lot of traffic</Sentence>
    <Sentence>Yup, really slowed me down in getting here</Sentence>
    <Sentence>Traffic caused yeah a of lot</Sentence>
    <Sentence>Really yup slowed, me in down</Sentence>

</Example>
```

Figure 12: An example of the dialogue that occurs in the XML file that is used in this thesis

Therefore we store the scripts for the prototypes in XML files such as what is shown in Figure 12, where it can be read in. Different XML files will hold different languages, however the

context and actual dialogue being spoken is the same. Example being that both dialogues will have a sentence saying “Hello, how are you?”, but will be represented in another language. Once the program reads the XML file, it parses the needed sentences from it and stores them in array lists in the Sentence class. For testing purposes these prototypes only read from one XML file which holds the dialogue in English, however it is possible to read in another XML file at the same time. Ideally, each player is reading in a different XML file, hence a different language is presented on each side. From there, the program will continue to read different sentences from the array lists whenever a player has completed a task.

3.2 Prototype 1 (Pong)

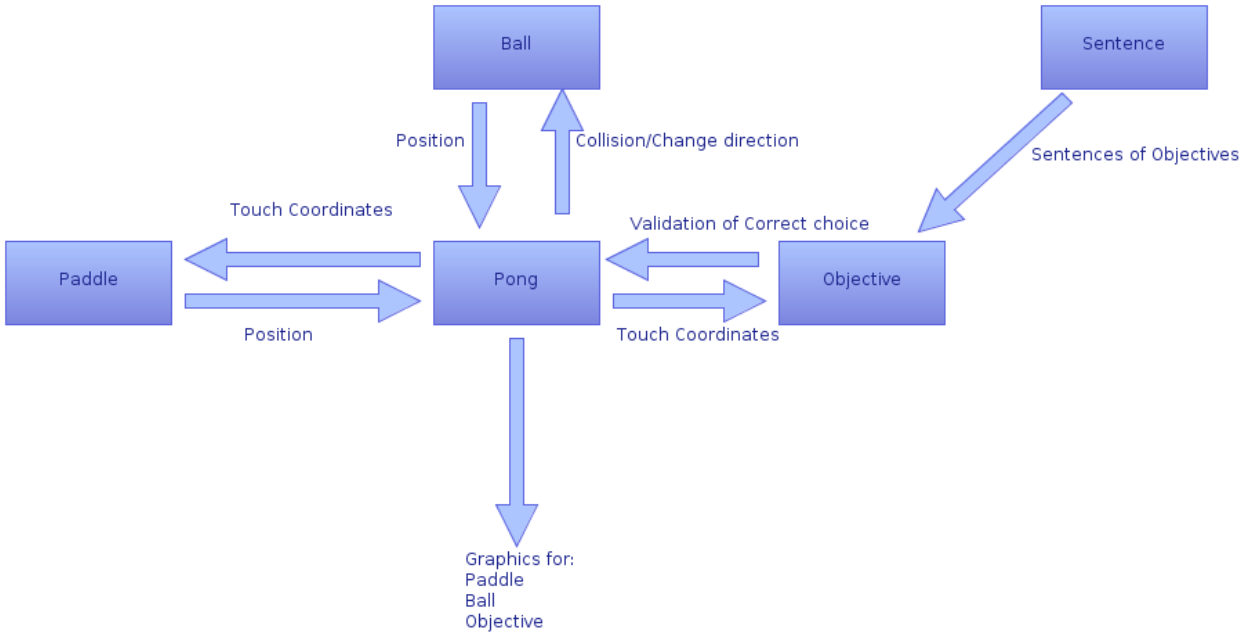

Figure 13: Data flow diagram for Pong, the touches originate from Pong, and are passed along to the other classes to run the game

Prototype 1 is separated into many class files. Pong is the executable class which interact with other classes to create the game. As shown in figure 13, Pong registers the touches, runs the

game play and learning elements and ensure draws all of the graphics for the game. Touches are handled in the same way in SampleTableCode, where the program is constantly updating the touch points while performing the necessary actions as long as there is one touch. Also if no touches are registered, specific actions are executed as well. The program will constantly being running the loop, updating to see if new touches have occurred, or to update touches. Because of this, we have inserted into this area of code all the functions which check to see if any of the objects on screen are being touched. In each of the objects (Paddle and Objective), there is a function to see if the player has touched the object. When the player touches the table, the program generates co-ordinates of the touch position and we pass those coordinates to the functions and from there, validate the contact. After seeing if these coordinates are within the area where the object is placed, the necessary actions are executed, whether it is moving and object, or advancing the game forward. The Pong class is also in charge of controlling the speed of the ball, only allowing the ball function that updates to only be run every so often milliseconds. This function checks to see if the ball has made contact with any of the paddles every time it updates the ball position as well. We have also made it so that the ball position is updated more frequently as the game draws longer, causing the ball to move faster.

The Objective class controls the obstacle the player has to go through in order to move their paddle. In this prototype, each player has two Objectives where one represents the correct sentence, while the other containing the wrong sentence which is represented with a boolean parameter. These sentences are taken from the array list that has read the XML files. The objective class than draws both the rectangle and the word onto the screen, and is

constantly checking to see if a touch has occurred in the rectangle. If the player were to touch the correct sentence, Pong will validate their actions and allow the player to move their paddle. However, if the player were to touch the wrong paddle, the colour of the rectangle will turn red informing the player has chosen the wrong sentence.

The Paddle and Ball are simple classes which only deal with updating their position and dealing with the collisions that can occur in the game. These collisions range from the ball being hit by the paddles, or the ball hitting the side walls. In each of these instances, either the X or Y direction of the ball is reversed. The paddles contain functions that check to see if the touches are within the area of the rectangle they draw, and repositions the rectangle to the new touch coordinate if there

3.3 Prototype 2 (PongBeta)

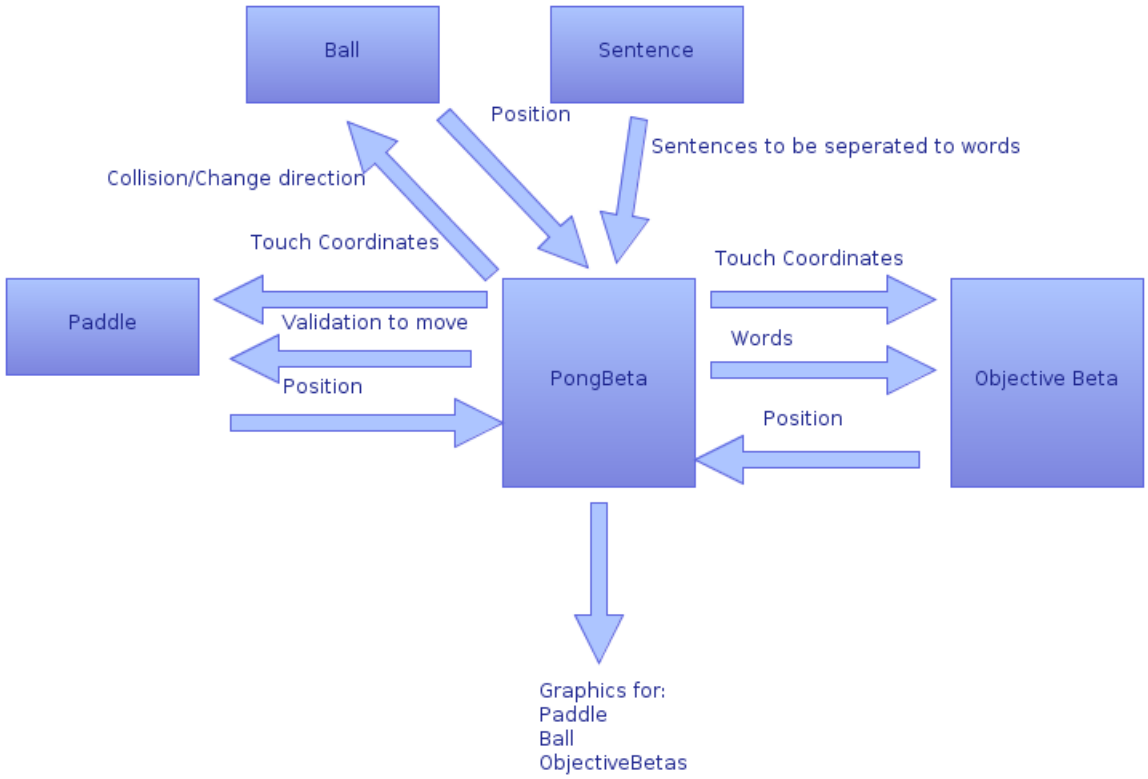

Figure 14: Similar to Pong, however PongBeta has much more responsibility over the game

Because PongBeta still uses the Pong game as a base, a lot of the code is recycled from Pong such as the Paddle, Ball, and Sentence classes. What have changed are the Objective and the Pong class, where they had to be modified in order to fit the new concept. The PongBeta class, as shown in Figure 14, still deals with the game play elements, drawing and such. However it is also in charge of splitting the sentence into words and creating the ObjectiveBeta class around those words. Since the amount of ObjectiveBeta depends on the amount of words in the sentence, we decided to store all the ObjectiveBetas into an array list for easier management. Prototype 2 also needed an algorithm in order to check to see if the Objective's are in the right order, which made the array list more essential to make things easier.

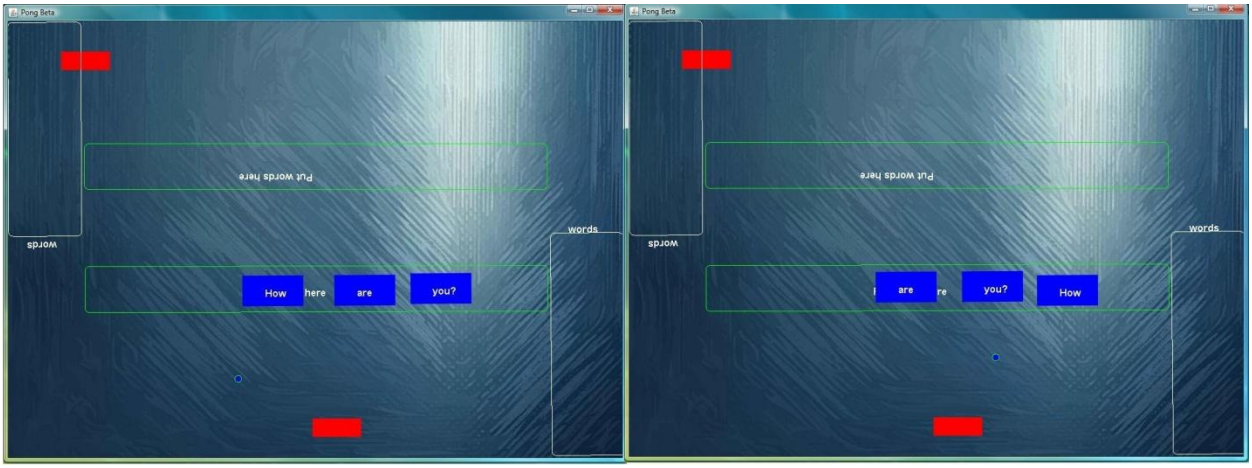

Figure 15: On the left screen, the score would be 2, allowing the paddle to move
On the right screen, the score would be 1, which is not sufficient enough to move the paddle

To do this we loop through the array list of Objective's and score every time an Objective's position is correctly behind another Objective. This is done by comparing the X values of the ObjectiveBeta's to see their orientation on the screen. Also, there is a certain range of Y values where PongBeta will check for the X values. Once the score reaches a certain number, it means that all the words are in correct order(See Figure 15 for example). Only then can the player move their paddle to hit the ball back.

Having proved functional, the method to check if an Objective was being touched in Pong was reused again in PongBeta. In this prototype though, the player can move the ObjectiveBeta to any position on screen. Though it is simple to update the position of the ObjectiveBeta's rectangle, we hit a new problem with this concept.

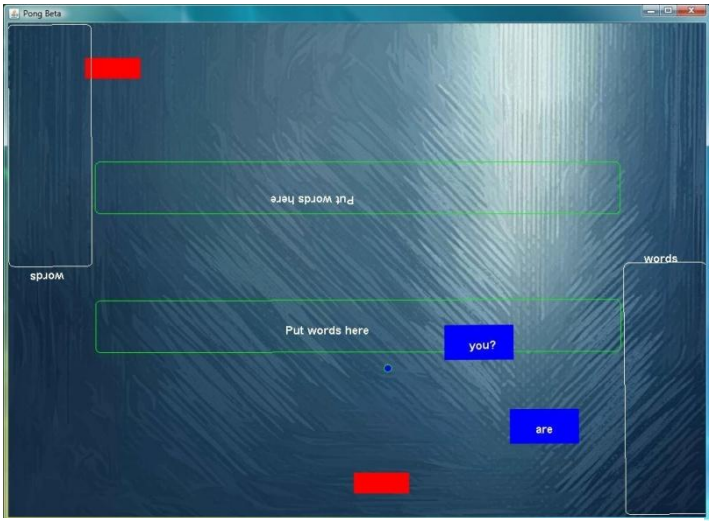

Figure 16: The object holding the word "How" is behind "you?

Because we're always checking to see if any Objective is being touched, if the player is moving one ObjectiveBeta, and so happens to touch another, the two ObjectiveBetas would overlap each other not allowing for the player to separate them(such as what is shown in Figure 16). This does not allow for the player to correctly form their sentences, and thus causing a problem. Many solutions were thought up of, such as only allowing one ObjectiveBeta to be touched at a time, or setting flags to designate certain touches were dedicated to certain objects on screen. However the one that was chosen was to have collision detection for the Objectives. What would happen, when two Objectives are close, is that the Objective not being touched would be pushed away, hence, never being touched and combining. Also if the player were to touch two Objectives at once and bring them together, they will push each other away.

This is because when we are updating the positions, we check to see all the other Objectives that are present, and see if any are to close. Depending on which corner the object is nearby, the X and Y values will be changed so that the object moves away from the touched object.

3.4 Prototype 3 (dueldisk)

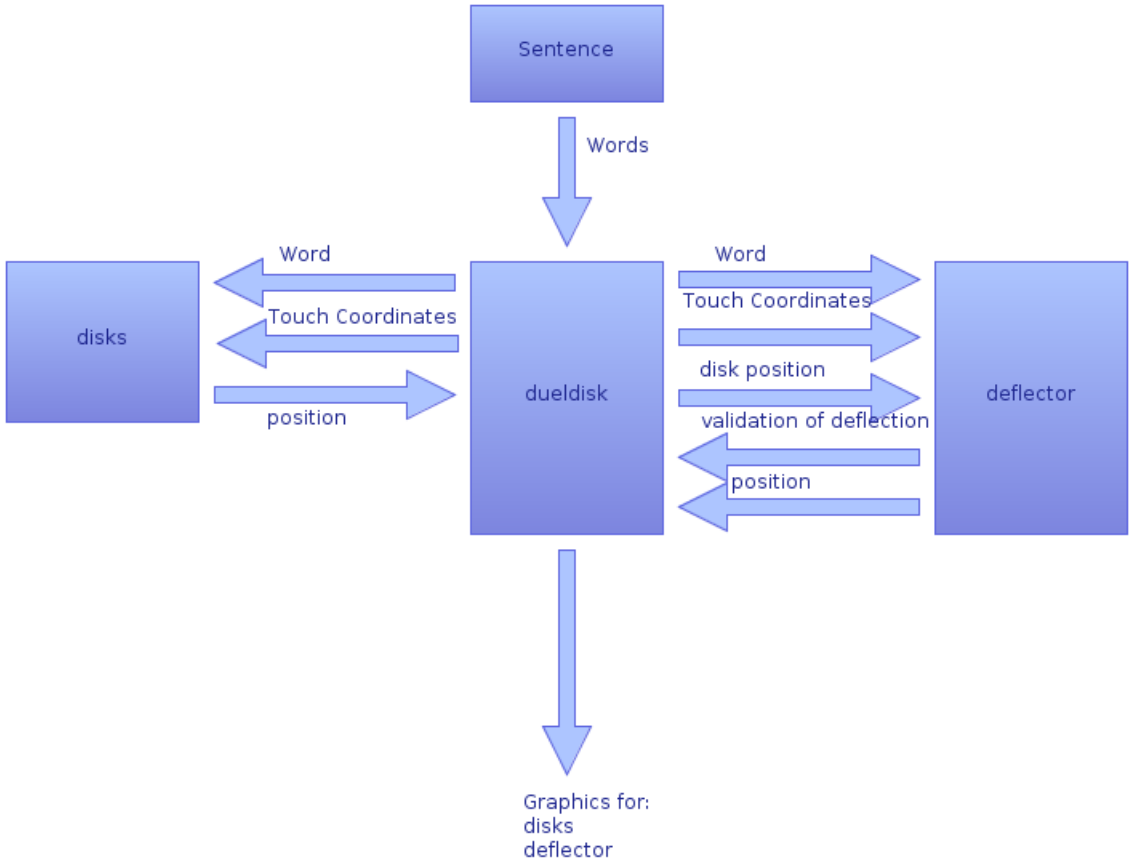

Figure 17: Data flow diagram for Dueldisk

Because Prototype 3 uses a completely different game play mechanic, most of the code in the previous prototypes can be used in order to create this prototype. The prototype still follows the same class structure (shown in Figure 17) as the other prototypes however. Once again, a single class (dueldisks) is in charge of drawing the graphics and controlling the game play. The other classes (deflector, disks, and Sentences) are in charge of the Objects that are drawn on

screen, controlling their positions and seeing if they are being touched. Together they interact with each other in order to create this prototype.

The class disk is in mainly in charge of updating the position and the movement of the disk on screen. The drawing and the updating of the position is similar to the Paddle class in the previous prototype. However, since the player is now going to be able to throw the disks now, an algorithm to calculate the velocity of the disks is needed. To do this, we save the positions of the last 10 coordinates where the user has touched the disks. From there we find the differences between those 10 coordinates, and next we find the averages of those differences. This will give the velocity of the disks, which is taking into account how fast the user is moving the disks. The table however updates the touches frequently, causes some of the touches that are recorded are of the same position. This makes the difference between some of the touches zero, therefore affecting the calculated average significantly making the velocity slow. Hence we most disregard those differences, and only calculate values that are not zero which will give a closer representation of the player's movement.

When to tell disk to calculate the velocity was also another issue we had to deal with. At first, we simply checked to see if check to see if any deflector or disk was being touched. If a disk that was being touched is no longer being touched, it would mean that player has thrown the disks. Therefore if that touch has disappeared, dueldisk would make the disk run the function to calculate the velocity. However if a deflector would be touched between the period of a disk being thrown, it causes the velocity not to be calculated until the player lets go of the

deflector. Due to the nature of how the table updates touches, we choose to calculate the speed outside of the loop where the table is always looking for touches. By creating flags, we can control when the disks should calculate the velocity which is at the correct moment when the player releases the disks.

As stated before, this prototype is a word game, therefore a different XML file had to be created to suit this trait. This file is once again is read in by Sentence, and is saved in array list which is accessed by dueldisk whenever necessary. Generally dueldisk will only read from this array lists at the beginning of the game, or whenever the player succeeds at defending themselves and require new words. This is when dueldisks reads the array lists of words, clears the array lists of the current disk and objectives, and creates new objects surrounding the new words, and stores them in the array lists.

4. User Evaluation

This thesis is heavily reliant on how the player conceives the program. This is because it is their problem we are trying to solve with this thesis, therefore making them a pivotal part in this. Hence we looked for feedback on the prototypes so that we will be able to further improve and perhaps learn from our failures that they have found. The evaluators that were asked to participate were people who had to learn English as their secondary language. The thesis was explained to the evaluators, and then they were presented with the prototypes with a small explanation of how to play. Afterwards they were encourage to play and were allowed to ask for guidance whenever they were confused.

In Pong, the evaluators found it hard to keep up with the dialogue. This is because they are unable to see that sentence that the other player has chosen, and therefore have no idea what the context of the reply is. They rely on picking only the correctly structured sentence, and not the actual reply to the other player's choice. The evaluators suggested displaying the other player's choice after correctly choosing their sentence. One evaluator suggested making the ball bigger, to give to make it easier to follow and hit back. Evaluators did enjoy this idea and found it to be interesting, and would like the graphics to be improved to give a more game feel.

When playing with PongBeta, evaluators were unanimous on their critique. One of the main things they talked about was the timing. They found that either the ball moved to fast, or that it took them to long to form the sentences, hence not allowing them to hit the ball back with the paddle. Because of this, the users felt frustrated and constantly felt they were losing,

despite putting their best efforts to finish things quickly and being proficient in the language being used. One solution they suggested is to move the paddle back to the middle after deflecting the ball, making it a shorter distance to move to the next position to hit the next volley.

Evaluators seem to have really enjoyed the dueldisk, more so than Pong and PongBeta. They seem to enjoy the simple word matching game, mixed with the competitive style of throwing and deflecting with the ball. Despite some of the bugs that were present at the time of the evaluation, evaluators still seemed to like this idea and would like to see this prototype move past its current stage.

5. Conclusion

After working on this thesis, many things were learned over the period of development. When programming for multi-touch surfaces, a lot of research is needed, and constant planning is need in order to create software that truly takes advantage of the capabilities. After showing the prototypes for user feedback, a lot of interest and encouragement was given. Perhaps this is a sign that this thesis is on the right step.

5.1 Summary

All three prototypes were designed and implemented with the core concept of teaching the language while providing the enjoyable game play element. They were also intended to promote interaction between the players, collaboration almost , to further help the teaching of a language. Upon implementing the designs, many problems were arisen, and solved, where each solution was chosen because it benefitted the multi touch surface. The prototypes are also designed to take in any XML file of any language, making it interchangeable with whatever the players wish. With all this, we hope we have achieved the three goals which this thesis had stated at the beginning.

5.2 Future Work

Although the three prototypes are completed, if one were to continue developing for these programs there is a lot of work that can be done. As of the moment this is being written, all three prototypes are only reading from the English XML file. Although it is possible to read in

another different language, this requires work to create the same dialogue in the same language. More animations can be drawn to give the prototypes more a game feeling. The graphics that are drawn right now are basic and not appealing to the mass public. Hence animations such as after images of moving objects, an explosion when a disk is deflected, or a score to keep track of which player is winning. All these suggestions will make the prototypes feel more like a game.

In prototype 1(Pong), only two options are presented, where one is the correctly structured sentence and the other incorrectly structured. If more work was put into the dialogue, perhaps more options can be present. Perhaps more wrong answers to throw off the player, or another dialogue path that will lead to a different discussion. All this is to further add to the learning element of Pong.

The main problem in Prototype 2(PongBeta), was when 2 words overlapped. Although we fixed that problem with collision detection, it was a crudely implemented program. When updating the position of the Objectives, it will check to see if any other Objectives nearby. If an Objective is to close, the Objective is moved away to clear the way. However it is possible that the pushed Objective to overlap another untouched Objective. The solution to this is to recursively check to see if other Objectives are nearby, however due to the time constraints, this was not able to be tested, and the crude implementation was left.

Prototype 3(dueldisk) was the last prototype to be developed, causing it to be limited to the time that was left to code. The function that creates the velocity for the disk when it is thrown requires some fixing because it can create velocities that are inaccurate to the movement of the player. Also, ideally the velocity of the disk is dependent on the length of the game. The longer the game proceeds, the faster the disk can be thrown. After a certain point in the game though, the player may not be able to read the word on the disks. This is where future work can include an audio component to the game, where an announcer can speak the word that has been chosen, allowing the player to react quick enough to choose the correct deflector.

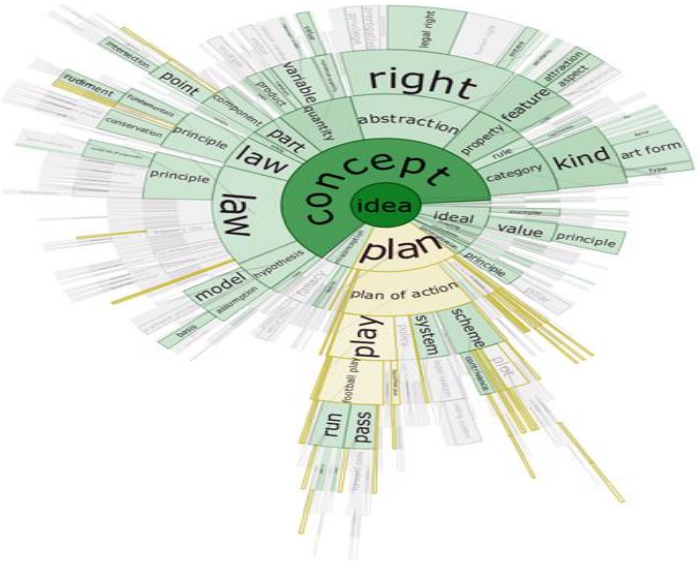

Figure 18: Docuburst Visualization, which is able to fit many words in a small amount of space. Can be modified to show words for dueldisk, allowing for player a larger variety of words [DoBu]

Also the variety of words was limited for this prototype, because of the way the player selected the word. If we were to use a visualization technique such as Docuburst's[DoBu], the player would have a larger variety of words to choose from. Figure 18 shows how words would be displayed if we were to use DocuBurst's style , where the player can choose from any of the words that appear.

Like any good game, it takes a long time to refine the mechanics and the elements that make up the game. This thesis has presented a new way of using multi-touch surfaces, which is to create dialogue between two languages. We have tried to make use of the advantages in which the multi-touch has provided. The different way of interacting with a computer, and the collaboration that can occur with large multi-touch surfaces were taken into account when create the prototypes. Ultimately this thesis was to introduce and inspire future developers who will learn from the success and failures and perhaps further improve on this concept.

6. References

[PO72] Pong video game. Created by Allan Alcorn, Produced by Atari.inc Released 1972

[SmTa] Smart Table Activities, Published by Smart Technologies

[LiMo07] Livemocha online Language learning Website. Featuring language learning games. Created by Shirish Nadkarni and Michael Schutzler. Launched September 24th, 2007

http://www.livemocha.com/

[TaCo01] SampleTableCode.java. Created by Dr. Collins, Dr Chistopher Collins, Mark Hancock and Luc Vlaming, 2001

[VisLi07] Vislink IEEE Information Visualization. Created by Dr. Christopher Collins and Sheelagh Carpendale. University of Toronto & University of Calgary, 2007

[DoBu] DocuBurst: Visualizing Document Content using Language Structure. Created by Dr. Christopher Collins, Sheelagh Carpendale, and Gerald Penn.