A Study of Kinesthetic Learning Activities Effectiveness in Teaching Computer Algorithms Within an Academic Term

Matthew Lai, Danny Luong, and G. Young
Computer Science Department, California State Polytechnic University, Pomona, CA 91768

Abstract

Kinesthetic learning is a teaching method that involves students' physical interaction among each other and the environment. The typical method of teaching, a classroom with an instructor talking and students listening and taking notes, has been the norm for centuries. This research attempts to show that the kinesthetic learning activities(KLA) approach can be a viable alternative. In this study, the performance of students from an undergraduate level computer science course, Parallel Processing ,is considered. In the Spring 2014 quarter, the class was divided into two groups. Each group was alternatively taught using KLA and traditional methods, allowing us to gauge the effectiveness of the KLA approach in one quarter. By performing this test over the course of a single quarter, we hope to more definitively show the effectiveness of a KLA approach to teaching students. The students' gained knowledge was measured through pre / post tests. We hypothesized that the KLA approach would be as efficient as the traditional lectures if not more efficient. The data that was collected from these tests favor our hypothesis.

Keywords: *kinesthetic, learning styles, teaching, computer science, algorithms.*

1. Introduction

In the present day, technology is everywhere and in the hands of everyone. Although the benefits of technology in the learning process are immeasurable, technology itself has many drawbacks. One of these drawbacks is the increase in distraction among students. In most classroom settings, students just sit and listen to an instructor. It is easy for them to lose focus by checking social media or chatting with their friends. In addition, several studies suggest that students' attention during a lecture may last only up to fifteen minutes [9]. Therefore, it is challenging for some students to maintain concentrationin a standard lecture.

People have different learning styles. According to Neil Fleming's model, there are four types of learners: The first group is visual learners who prefer learning through symbolic representations, using graphs and charts to obtain knowledge. Aural learners are those who perceive more knowledge when they are listening to a lecturer, and they have no problems learning in traditional lecture-based environments. There are also learners who prefer learning through text, either reading a book or writing notes. The last type is the kinesthetic learners who gain more knowledge through physical simulations for the concepts they are learning [4].

Active learning or kinesthetic learning approach is a solution for those who have trouble paying attention for a long period of time because it requires students to be physically active during the lecture. Since computer science topics are theoretical and intangible, grasping the concepts may be challenging for the students. Therefore, kinesthetic learning activities can be helpful and efficient.

1.1. Active learning characteristics

Active learners tend to learn better through physical activity. Hence, they prefer performing arts and athletics more than studying theoretical science. They get distracted easily in a traditional lecture environment. They perfectly apply to the Chinese proverb which says "*I hear and I forget. I see and I remember. I do and I understand.*" These learners gain more knowledge when they use their hands and bodies. The KLA teaching method provides a suitable learning atmosphere because it forces learners to pay full attention in their movements. For example, a role playing KLA allows learners to imagine a situation where a problem arises, and they act like the solving agent. Another example is playing a game in the classroom like playing the tower of Hanoi puzzle. Kinesthetic learning gives an exciting learning experience for all learners, especially active learners.

1.2. Active learning techniques

Faust and Paulson [3] have categorized active learning techniques into six types: First, exercises for individual students, which include quizzes that require the students to be in the classroom and pay attention to the lecture. Second, questions and answers, which is a method that allows the students to ask questions anonymously using the fish bowl technique, and the rest of the students answer these questions. This method empowers the students' role in their own learning process. Third, immediate feedback, which is exactly as its name suggests; it gives the professor an opportunity to measure the class's understanding as whole, and it is a helpful method that can be used in large classes. Fourth, critical thinking motivators, is a method that aims to get the students to think about the concept before they actually learn it using puzzles. The tower of Hanoi puzzle is a good example of this technique. The concept can be explained to students before they attempt to solve the puzzle. The learning process actually happens when they solve the puzzle with the algorithm that

they have been provided. The fifth active learning type is share/pair. This method forces students to work together as pairs by exchanging their thoughts and opinions and explaining unclear concepts. An example of this learning type is pair programming which always produces high quality software [10]. The last technique is the cooperative-learning strategy, which requires students to be divided into groups. Each group is responsible of solving a given problem and explaining it to the rest of the class. This method enhances students' communication and social skills.

1.3. Benefits

The typical classroom setting, where students are just passively sitting and listening, leads students to enter their "relaxed zone," where it is much harder for them to maintain full attention. On the other hand, a kinesthetic activity fills the room with energy and excitement, and it makes students see things from a new angle. Moreover, a KLA also helps students to develop interpersonal skills, since it requires them to communicate with each other. It helps timid students to interact with their classmates in an observed environment. As reported by the National Training Laboratories' pyramid of learning, students gain only 5% of the information given to them in the form of lectures while learn-by-doing retains 75% of the knowledge. Furthermore, students retain 90% of what they have learned when they teach each other [7].

1.4. Obstacles

Despite the numerous advantages of active learning, some instructors are reluctant to adapt this teaching method for several reasons. For example, how an instructor decides to manage their time has a major influence of thewhole learning process and its outcome. An instructor must account for the preparation and execution time, as well as students' responses to such new learning strategy. Some instructors think that they do not have enough time to cover all the topics they have assigned for a certain quarter, and active learning can reduce the amount of available time. As a result, an instructor may conclude that lecturing is more convenient for delivering the information. However, a scenario where an instructor explains and students listen does not guarantee that students will be able to absorb the knowledge. Students may leave their classroom with some bits and pieces of a lecture in their notebooks and with nothing in their heads. Also, there is a big chance that a number of students may stop the instructor to ask questions, and the instructor might end up not covering every topic that needs to be covered. In addition, in a traditional lecture, the teacher has more control of the class where in an active learning session; students are more involved with how the course is run. With this in mind, an instructor may be ambivalent to use KLA as new problems with classroom management may arise.

2. Past Research

In the past decade, kinesthetic learning activities were common in preschools and elementary schools. In 2004, Tammy Nguyen did a study named "Do kinesthetic strategies influence students' achievement?" She studied the effectiveness of teaching mathematics kinesthetically for first-graders. She taught them addition and subtraction using both a traditional method and a kinesthetic method using hand signals, and she tested them. The results showed that the kinesthetic test scores were higher than the non-kinesthetic test scores. Also, she noticed that the students were excited about participating in the KLA's. Though significant for its findings on the benefits of kinesthetic learning at the preschool and elementary level, this study by itself cannot prove that kinesthetic learning is more effective than traditional lecturing in general [6].

In 2009, Katherine Gunion taught middle school students the concept of recursion in a KLA fashion. She held a 7-weeks-after-school program for the students and used 6 different KLAs. She wanted to answer three questions:

1- Can students identify recursion?

2- Can the students understand and apply recursion?

3- What is the effect of understanding recursion on their attitudes towards computer science?

Her answers were as follows: most of the students were able to identify recursion, but few of them were able to apply it on solving different problems instead of solving them sequentially. For the last question, the different data collecting methods she used showed that the students enjoyed the activities, and half of them came back for the next offering of the program, but this still does not prove our goal [5].

Similarly, for college students, there were several publications about kinesthetic learning activities in the computer science field. "Human cons cell jeopardy" is such a KLA developed by Begel, Garcia, and Wolfman to introduce any programming course [2]. In 2007, Sivilotti and Pike developed a set of KLAs for both undergraduate and graduate students to teach them the concepts of distributed systems [8]. In general, there are only a few KLAs that have been designed for computer science education at the university level.

3. Research Goal and Methodology

In this study, we applied kinesthetic learning strategies on students enrolled in a Parallel Processing course in the Computer Science department at California State Polytechnic University, Pomona (Cal Poly Pomona) to test the efficiency of the KLAs compared to traditional lectures. Our hypothesis says that kinesthetic learning will be as effective as traditional lecturing, if not more so. It is important to mention that this research has been revised and approved by the Cal Poly Pomona Institutional Review Board (IRB): protocol #14-0004,

and it has met the federal and state regulations and Cal Poly Pomona policies regarding the safety of the human subjects who participated in this research.

In Spring 2014, the students who attended the CS370 Parallel Processing class participated in this study. Similar to our previous study from the Winter and Spring quarters of 2014 [1], we presented the groups of students to both a traditional lecture on the selected topics as well as a KLA lecture. However, for this path of research, our study group consisted of only a single class from a single quarter. The chosen topics for this study were parallel pipeline algorithms for addition, prime number generation, insertion sort, and systems of linear equations and parallel algorithms for merge sort, bubble sort, radix sort, and shear sort. Prior to the lectures, the Parallel Processing class was given multiple pretests to gauge their background knowledge about these topics. The class was then divided into two groups and each group was alternatively taught using traditional and KLA methods over two sessions. The students were given a posttest after each lecture to determine their understanding of the topics after learning from either the traditional lessons or the KLA activities. To this end, we developed kinesthetic learning activities for the listed topics to study the benefits of using kinesthetic approach in Computer Science.

3.1. Parallel Pipeline Algorithms

3.1.1. Addition

For the addition pipeline KLA, we performed the calculations such that each participating student simulates a processor in a linear series of processors. The first student receives an instance of addition and performs an operation. That processor passes on the instance to the next processor in the pipeline while simultaneously receiving another instance of addition. Along the pipeline, the next processors performs an operation before passing on their results to the next processor. This demonstrates one of the usages of pipeline processors in processing multiple instances of the same operation, in this case multiple separate instances of addition.

3.1.2. Prime Number Generation

For our design of a KLA for prime number generation in a pipeline, we again have the students each act as a processor in a linear series of processors. Numbers of increasing value, beginning with 2, are passed one by one to the first processor in the series. As each processor receives a number, they do the one of the following:

1. If the student does not possess a number prior to receiving the new number, they retain the new number and perform no further actions for the round.
2. If the student does possess a number and the number that they receive is not a multiple of the number they possess, then the student passes the number that they have received to the next student at the beginning of the next round.
3. If the student does possess a number and the number that they receive is a multiple of the number that they possess, then they immediately drop the number that they received.

This process can continue until the instructor decides to end the KLA or the final processor in the pipeline obtains a number.

3.1.3. Insertion Sort

The insertion sort KLA that we implemented is a simple exercise. Once again the students are arranged such that they are each a processor in a pipeline series of processors. A number is fed to the first processor of the pipeline. If a processor receives a number and he does not possess a number then the processor retains the number and performs no further actions. If the processor receives a number and he does already possess a number, then the processor compares the two numbers.In the next round, it passes the smaller number to the next processor in the pipeline. At the end of the KLA the students should possess a series of numbers sorted in descending order starting from the head of the pipeline.

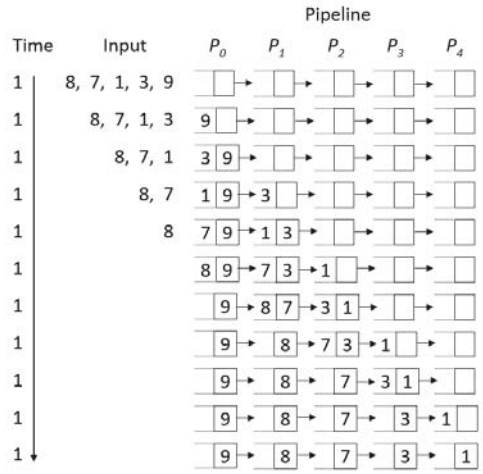

Figure 3.1 Image demonstrating pipeline insertion sort.

3.1.4. Systems of Linear Equations

A KLA for teaching how a system of linear equations can be solved using a pipeline of processors can be run as follows. Each student represents one stage of the system of linear equations pipeline algorithm and each stage is responsible for one equation. When a stage receives a variable that was previously unknown, it copies down the number and passes the number to the next stage in the pipeline. It then uses this variable to attempt to calculate its own unknown variable. If the processor is able to solve for the unknown variable, it passes it on to the next processor in the pipeline. This shows how a pipeline can be used to pass information further along the pipeline, even before the processor has completed all of its internal operations.

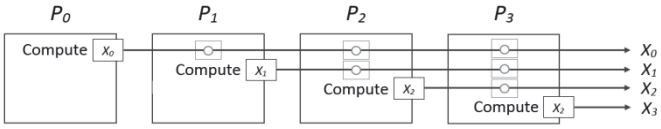

Figure 3.2 Image demonstrating Pipeline System of Linear Equations

3.2. Parallel Sorting Algorithms

3.2.1. Merge Sort

Because of the nature of merge sort, a KLA for running merge sort in parallel is fairly easy to design but not necessarily easy to implement with the students. In our design for a KLA of parallel merge sort, we have each student represent a single processor. We begin the KLA having already divided the array into the smallest elements for the algorithm.Each number of the array is represented by a separate piece of paper. During each round, each student only performs one comparison, during which it compares the numbers as required by the merge sort algorithm and swap them as necessary. Each subarray of numbers is then passed to the next processor in thetree and the merges are performed in parallel. At the end of the process, one of the students performs the final merging of the two largest subarraysat which point, there is a final sorted array.

3.2.2. Bubble Sort

Parallel Bubble sort, also known as Odd-Even Transposition Sort, is a relatively simple sorting algorithm designed to work on parallel processors. It is a comparison sort similar to a bubble sort that compares, alternately, the odd-even and even-odd adjacent pairs and swaps them if they are in the wrong order. The KLA we designed for this course utilizes each student as an individual processor. Numbers on adhesive paper are stuck to the wall and the students alternately compare their assigned even and odd pair then the odd and even pair and switches their placement as necessary. When no swaps are made in a given round, the sort is finished.The resulting array of numbers should be in sorted order.

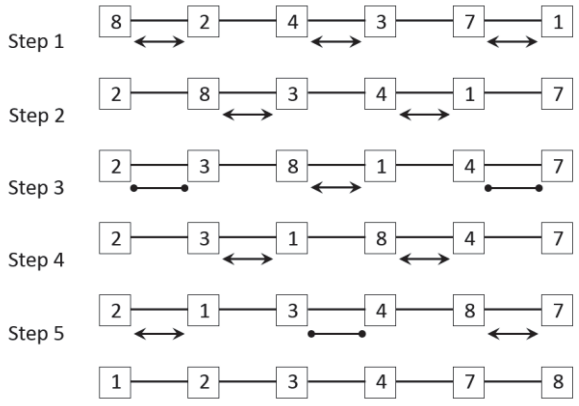

Figure 3.3 Image demonstrating Odd-Even Transposition Sort

3.2.3. Radix Sort

Radix sort is a non-comparative integer sorting algorithm that sorts each integer by grouping them according to individual digits of each integer. In the KLA we designed to demonstrate parallel radix sort, we utilized plastic cups as the "buckets" into which each element is placed. We used ping pong balls marked with integers to represent the integers for easier handling. For ease of sorting, we used integers of base four and maximum integer length of three. Each student acts as a processor and are arranged such that the first order of students, in our case only one student, processed each integer and passed the integer to the appropriate second order of students according to their most significant digit. Each succeeding order of students processed their given integers according to succeedingly less significant digits and passed them onto the next order of students until the least significant digit was processed and placed into the appropriate plastic cup. If the ping pong balls were removed from the cups in order, then the integers were shown to be sorted.

3.2.4. Shear Sort

Shear sort is an algorithm designed for parallel processing. Shear sort sorts a two-dimensional array into a snake like order within the array. It alternatively sorts the array row-wise, where the odd rows are sorted in ascending order and the even rows are sorted in descending order, and column-wise where all columns are sorted in ascending order downwards. For our KLA, we have the students stand together as an *n* by *n* two-dimensional array. Each student in the array holds a paper with an integer printed on it. There are then *n* student processors who alternates between sorting the array row-wise and column-wise. When there are no changes made in a round, the array is sorted in a snake like fashion.

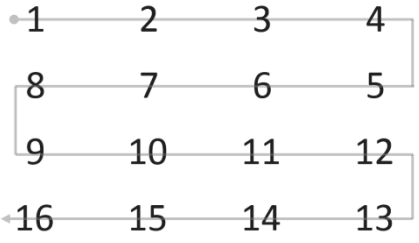

Figure 3.4 Image demonstrating array sorted in snake-like order

4. One Quarter Methodology

In what we believe to be a unique study, we have performed this KLA research using a single class of students to attempt to better gauge the effectiveness of KLA teaching methods. We have not found this particular line of research in any of the existing literature on KLA studies.We present our method of evaluating KLA effectiveness utilizing a single body of students, conducted over the course of a single quarter. To do this we had to solve a small number of problems.

4.1. Constraints and Requirements

Due to the nature of this research, there are certain constraints and requirements that must be understood and met to perform this study.The most important constraint is the requirement of two separate lecture areas. While the students of one group are learning some material with a traditional lecture, the other group is learning the material under the KLA method. We don't want to cross contaminate how the students learn the material. It is then essential that we have access to two separate rooms from which we could lecture the two groups of students separately. In our study, we had access to a room with an adjoining lab area with standing space, perfect for conducting KLA activities. The main classroom was a larger space with writing space where the students could be taught using traditional methods. It is also imperative to keep in mind what equipment is available in each room. The equipment, such as whiteboards, computers, and projectors are essential tools with which students can be taught. In our situation, our large main classroom is a widely used lecture room equipped with all the necessary lecture tools, making it easy to perform traditional lectures. However, the smaller adjacent space, in which we planned to lecture using the KLA method, is a lab area designed for testing hardware and software projects and did not contain a whiteboard or a projector. This classroom's deficiency in teaching ability was mitigated by using a loaned projector cart with which we could project our lectures onto the wall.

The next most important resource required for this study was the instructors who would teach the material. For this study, we needed two comparable instructors who are capable of performing both a traditional lecture and a KLA lecture. Because of these requirements and the time of day at which the classes were held, it was decided that two graduate students would perform the lectures. With this arrangement, we avoid having a more experienced lecturer skew the results in favor of either the KLA or traditional method.

4.2. Planning and Execution

There were several steps in planning the KLA study. The first step was to prepare for the lectures by deciding on the algorithms that would be most appropriate for the class. With the time given, we were able to plan for two class lectures in this KLA study. Pipeline algorithms were a part of the planned class curriculum, so we dedicated a class lecture to various parallel pipeline algorithms. There are also an abundance of parallelizable sorting algorithms which lend themselves well to KLA activities, so we decided to use the second class lecture to instruct the students on sorting algorithms. We then selected four different algorithms for each lecture. Our plan for this study was to split the class into two groups and have each lecturer spend half of a lecture instructing one group using KLA methods for two algorithms and the other half of the lecture instructing the other group of students on the other two algorithms using traditional methods. With this approach each group would receive both KLA and traditional lectures on a given lecture but on different topics and each lecturer would give instruction on two algorithms using both traditional and KLA methods. By organizing the study in such a way, we hoped to find good results on the effectiveness on KLA by comparing a lecturer's effectiveness using traditional methods versus their effectiveness using KLA methods.

The next step was to decide the method with which to divide the class so that we could alternate between the teaching of lessons using KLA and traditional methods. Because this KLA study took place over multiple days, we needed find a method that satisfied a few requirements. Each group of students needs to be of approximately the same size. We also needed a method of dividing the class that would be repeatable so that the students would be in the correct group for each lesson. We considered multiple methods before deciding to have the students split into groups based on whether their birthdays occur on an even or odd date. We labeled the students with odd birth dates as the orange group and provided pre-tests and post-tests on orange paper. We labeled the students with even birth dates as the white group and provided pre-tests and post-tests on white paper. This allowed us to use visual cues to easily differentiate between the two groups. This even and odd birth date approach provides several benefits. Because there are approximately the same number of odd and even dates in a year, we were able to theoretically get groups of about the same size. And because the dates of the students' birthdays remain static, the groups that they would divide into will remain the same through each of the sessions. Another important benefit is that this method is quick and simple for the students to follow, allowing more time for the lessons to occur.

With the planning complete, we could begin our study. In our first interaction with the students the week before the lectures, we briefed the students on our research and explained to them our goal, methodology and what they could expect from the upcoming lectures. We then obtained signed permission from each student in which they agreed to participate in the KLA study. Finally we provided a pre-test to the students to gauge the students' understanding of the algorithms chosen for each lecture so that we would have a baseline with which we could compare our final results.

For the first lecture, we divided the class according to our even-odd birth date methodology. The orange-odd group was asked to move to the smaller adjacent classroom where they received a lecture, containing KLA elements, on the prime number generation and pipeline insertion sort algorithms. The white-even group remained in the main classroom where they received a traditional lecture on a pipeline addition algorithm and pipeline system of linear equations algorithm. Once the lectures were complete, each group was asked to complete a post-test to measure their understanding on the algorithms in which they just received a lecture on. The instructors then exchanged classrooms where, in the main classroom, the

students would receive a traditional lecture on pipeline prime number generation and pipeline insertion sort, and the students in the adjacent classroom would receive a KLA lecture on pipeline addition and pipeline system of linear equations. Once these lectures were completed another post-test was used to measure their understanding after receiving the second half of the pipeline lecture. In this way, the pipeline lecture was completed.

The second lecture occurred similar to our first lecture. The main difference was that the white-even group was asked to move to the smaller adjacent classroom where they received the KLA lectures. This white group received KLA lectures onthe topics of shear sort, parallel radix sort, odd-even sort, and parallel merge sort algorithms. The orange group received traditional lectures in the main classroom on the same topics. Two post-tests were given to students after each half of the lecture, one before the graduate instructors exchanged classrooms and another one at the end of the lectures, each test gauging the student's knowledge on the two algorithms they had learned in each half of the lecture.

5. Results

With the post-tests and pre-tests of the approximately 25 students in our single course, we can analyze the effectiveness of kinesthetic learning activities in a computer science classroom in comparison to more traditional teaching methods. We found in our study that the KLA methodis more effective at teaching students the selected algorithms and that they had a better understanding of the material after receiving a KLA lesson than they had from the traditional lesson.

Our results show that students who took the post-test after having been instructed with the KLA method scored 17.9% better than when they were instructed with a traditional method. The orange group improved 49.4% on their post-test over their baseline pre-test in parallel sorting when they learned from a traditional method but improved by 59.3% when they learned from a KLA lecture. Similarly the white group improved by 30.8% on their post-test over their pre-test after a traditional lecture and improved by 53.4% after a KLA lecture.

Pre-Test Odd (Orange)		Post-test Odd (Orange)	
Parallel Sorting	Pipeline	Traditional Parallel Sorting	KLA Pipeline
0	0	5	8
0	0	7	9
1	0	10	11
1	0	10	15
1	0	11	17
1	0	12	
2	1	12	
3		15	
		17	
Average	1.125	0.143	11	12
Average Score (out of 20)	0.05625	0.00715	0.55	0.6

Table 5.1 Test Results for Odd-Orange Group

Pre-Test Even (White)		Post-test Even (White)	
Parallel Sorting	Pipeline	KLA Parallel Sorting	Traditional Pipeline
0	0	5	0
0	0	6	1
1	0	10	2
1	0	12	6
2	0	12	6
2	0	15	7
2	0	15	10
3	0	15	11
3	0	15	11
4	0	15	12
5	5	16	
		17	
Average	2.09	0.455	12.75	6.6
Average Score (out of 20)	0.1045	0.0227	0.6375	0.33

Table 5.2 Test Results for Even-White Group

In the pre-test, a definite trend that can be seen is the low scores. This is to be expected as this is the first time many of the students have seen most of these algorithms. The scores of the post-test show an increase in understanding the materials after the lectures.

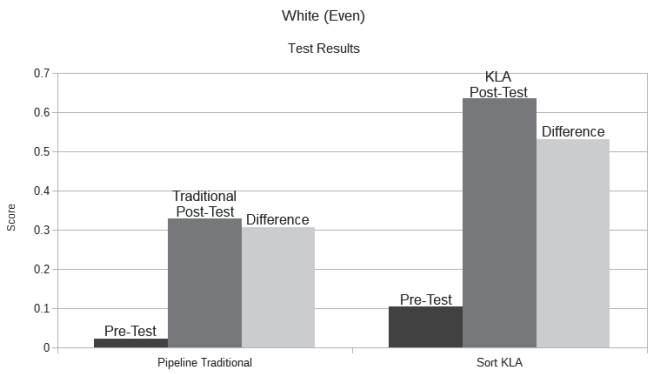

Figure 5.3 Chart Showing Pre-Test and Post-Test Results for the Even-White Group

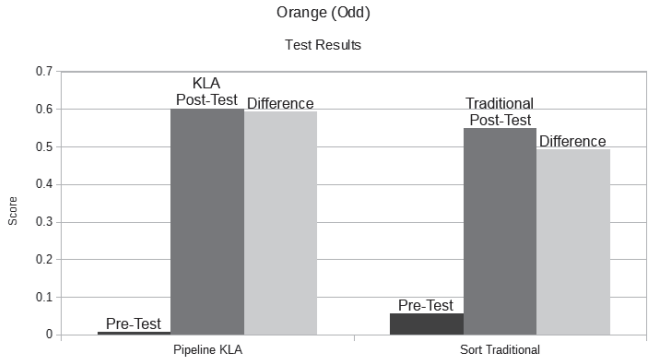

Figure 5.4 Chart Showing Pre-Test and Post-test Results for the Odd-Orange Group

In the two previous figures, you can see that the improvement between the pre-test and post-test is most pronounced after the KLA lectures. The greatest improvements to the students' scores came after they were instructed using the KLA method.

6. Conclusion and Future Work:

From the given figures and results, there appears to be a positive correlation in how well a student learns more from a KLA lectures in comparison to more traditional lectures. Using our methodology, we have shown this result using a single body of students over a single quarter. With the fact that we were able to run this test with two graduate student instructors individually teaching the same algorithms to each student group with both KLA and traditional methods, we can almost eliminate the effects that any difference in teaching skills between instructors might have on the results. With KLA we have a good teaching method that will keep students engaged and focused during class. Through the results of our experiment, we believe that KLA can be a powerful tool to teach students different topics of computer science and increase their understanding of the subject.

In further studies, we hope to be able to research what effects more experienced instructors may have on the score difference between traditional methods and KLA methods. More studies of KLA in the computer science field can only be an improvement to our current state of research as more experimental data on the effects of KLA on teaching students computer science methods will give more confidence to the effectiveness of KLA in computer science classrooms.

REFERENCES

[1] Alraddady, Sara, Danny Luong, and G. Young. 2014. "A Study of Kinesthetic Learning Activities Effectiveness in Teaching Computer Algorithms." *Proceedings of the2014 International Conference on Frontiers in Education: Computer Science & Computer Engineering.* Las Vegas, NV: 400-406.

[2] Begel, Andrew, Daniel D. Garcia, and Steven A. Wolfman. 2004. "Kinesthetic Learning in the Classroom." *35th Proceedings of the Technical Symposium on Computer Science Education*. New York, NY: 183-184.

[3] Faust, Jennifer L., and Donald R. Paulson. 1998. "Active Learning in the College Classroom." *Journal on Excellence in College Teaching*, 9(2): 3-24.

[4] Fleming, Neil D. 2001. "Teaching and Learning Styles: VARK Strategies." Christchurch, New Zealand: Author.

[5] Gunion, Katherine. 2009. "FUNdamentals of CS: Designing and Evaluating Computer Science Activities for Kids." Victoria, BC, Canada: MS thesis, University of Victoria.

[6] Nguyen, Tammy. 2004. "Do Kinesthetic Strategies Influence Students‟ Achievement?" Pomona, CA: MS thesis, California State Polytechnic University, Pomona.

[7] Polovina, Simon. 2011. "About the Learning Pyramid." http://homepages.gold.ac.uk/polovina/learnpyramid/about.htm (accessed December 1, 2013).

[8] Sivilotti, Paolo and Scott M. Pike. 2007. "The Suitability of Kinesthetic Learning Activities for Teaching Distributed Algorithms." *38th Proceedings of the Technical Symposium on Computer Science Education*. New York, NY: 362-366.

[9] Wankat, Phillip C. 2002. "The Effective, Efficient Professor: Teaching Scholarship and Service." Boston, MA: Allyn and Bacon.

[10] Williams, Laurie, Robert R. Kessler, Ward Cunningham, and Ron Jeffries. 2000. "Strengthening the Case for Pair Programming." *IEEE Software.* 17(4): 19-25.