

Category: Computer Science

Table Number: H0407

Student Name: Nikolaos Liidakis

Team Members (if any):

Project Title: Facial Recognition

Abstract: The programming objective of this experiment was to create a facial recognition algorithm that could effectively identify a member within a social network. The goal of the algorithm was to take an uploaded image by a user and through a process of pixel by pixel manipulation display the most related images within the network database. Each uploaded image would be transformed into a matrix of RGBA pixel values and this matrix would be simplified into a matrix of linearly independent vectors. This matrix would then be compared to a predetermined average face. If the matrix passed a specific threshold it would then be compared to individual images; the images with the most similar numeric value to the image would be outputted. The user would then have the option of choosing one of the outputted images or cancelling the search altogether. The program's purpose was to give individuals a way to reconnect with friends from the past by allowing them to input old pictures and use these images to search the database. The experiment began by first creating a successful algorithm that was able to give a value of uniqueness to each image. This uniqueness value would then be compared to the uniqueness value of the pictures in the database. The facial algorithm was effective in determining that the image was a face in over ninety percent of the trials, and it was able to output the correct corresponding image from the database eighty percent of the time. This data was effective in determining which faces in the network were most similar to the inputted image.

Category: Computer Science

Table Number: H0405

Student Name: Heather Watson

Team Members (if any):

Project Title: Computer Programming 101

Abstract: I wanted to know if I could program a calculator with the four basic operations and parentheses. I predicted that I would be able to. I programmed it mostly with trial and error. I thought of different ways to program it and decided which way would work the best. Then I would test it to make sure that it was working the way I wanted it to. If it wasn't, I tried to come up with a way to fix the problem. Most of the time I just fixed something I had programmed. The calculator I programmed works and is accurate.

Category: Computer Science

Table Number: H0402

Student Name: Joseph Ivie

Team Members (if any):

Project Title: The EyeMouse

Abstract: My engineering goal was to create a program that could use a web cam looking at an eye and figure out where it was looking, and finding the best one in speed and accuracy. **Hypothesis** The following method will work best: Find darkest point> find edges using fill algorithm> use the edges of a circle equation to find some centers> average the centers> use calibration points to determine the spot on the screen. **Methods** The program can measure accuracy by finding the average distance to the center, and time is measured by using a timer inside the program. **Results** Find darkest point> find edges using fill algorithm that avoids color> average the edge points> use calibration points to determine the spot on the screen.

Category: Computer Science

Table Number: H0401

Student Name: Jordan Conlin

Team Members (if any):

Project Title: Multi-Cored Overkill

Abstract: Video games are the greatest thing since the sliced bread, yet they still have limitations, such as loading time, computer opponent intelligence, & graphics. The purpose of this project is to show that modifying the source code of a game engine to accommodate multi-threading will improve a games performance when run on a multi-core computer. This modification would allow a game to load faster & have better computer opponents. My problem is: Would it be feasible to modify an existing game engine to be multi-threaded? And would the games that use that engine run faster? My Hypothesis is that it would be feasible to modify an existing game engine to use all of the processors in a multi-core CPU I began by running a test program written in Java on multiple computers to see how each computer performed with 200+ threads. It was no surprise that a quad core CPU was much faster than single core CPU using Java. After more research, I found a game engine written in C++, called Ogre. To have a baseline for this test, I used a quad-core gaming machine. I then increased the workload of the game until it was running extremely slow on the gaming machine. The main problem I ran into with this experiment is that once I tried to insert multiple threads into the engine, I couldn't find any place to insert them without affecting the timing of the game. After consulting with my mentor I came to the conclusion that small modification would not work, it would be more practical to rewrite the entire game engine. After more research I found the concept of parallel processing, where you write code that is designed to run on parallel processors. To prove this out, I created three simple games using Java, each with a different game loop to see which implementation gave the best performance. My conclusion is that I reject my hypothesis because it is not feasible to modify existing game engines to be multi-threaded. I did, however, find that parallel processing will require that game loops will need to be rewritten to provide support for multi-core CPU's.

Category: Computer Science

Table Number: H0404

Student Name: Bridger Maxwell

Team Members (if any):

Project Title: Creating a Computer System For Simulation-Based Education

Abstract: Simulations can be a powerful tool for education, as shown by the Christa McAuliffe Space Education Center. Simulations employ a unique integration of role play, active learning, and narrative to create an environment where students can exercise problem solving, apply principles learned in the classroom, and build teamwork while having a great time. However, building a simulator is beyond the budget and resources of most schools. One of the most expensive, and complicated, aspects of a simulator is the computer system which the students interact with. Software that can run a simulation is not commercially available, and creating such software can be enormously expensive. Additionally, much can be learned from the current software at the CMSEC. My goal was to create a computer system that could be used in schools wishing to create their own simulators. I both learned from and improved the software currently used at the CMSEC. My project had many engineering goals in the areas of networking, extensibility, maintenance required, themeability, and educational content. The finished product is a family of applications, including a server for keeping all of the stations in a simulator synchronized, a client which bootstraps and themes a station, several content screens, and hardware controllers for networking tactile controls or automating lighting. Additionally, I created several tools to facilitate the development of additional content which can be later added to the simulator.

Category: Computer Science

Table Number: H0403

Student Name: Nicholas Day

Team Members (if any):

Project Title: Encryption System Development

Abstract: Question: How hard is it to build a basic encryption system? Hypothesis: I believe I can create an application that encrypts, decrypts, and cracks the encryption on text files within a reasonable time frame. Procedure: I programmed the application in Java starting last December. The Caesar encryption function of the program takes all the characters in a text file and shifts them forward a number of characters based on the hard-coded Java integer of the selected shift character before saving the output to a new file. The Vigenère encryption function does the same thing but uses different shifts according to the value of the current letter in a keyphrase. The decryption functions work about the same but they shift characters backwards. The Caesar cracking function in the program counts the number of common words in each possible shift and saves the shift that has the most (the one that should be in English). Sixteen text files were used to test every function: four digitized books (The Adventures of Huckleberry Finn, A Tale of Two Cities, The Adventures of Sherlock Holmes, and Hamlet), four articles from PC World, four articles from The Salt Lake Tribune, and four articles from Wikipedia). Results: I successfully completed the encryption and decryption parts of the application but I haven't been able to fully complete the Caesar cipher cracking part; however, I will be working on my project for the next two weeks attempting to fix every bug the program currently has.

Central Utah Science & Engineering Fair 2009
Senior Division

Category: Computer Science
Number: H0406

Table

Student Name: Chase Jensen
Team Members (if any):

Project Title:

Abstract: