Project Portfolio

Haptic virtual temporal bone dissection simulation.

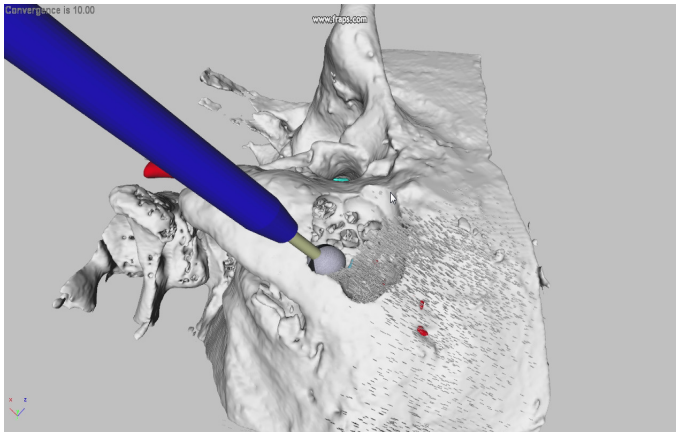
I developed this simulation to train ENT surgeons to perform inner ear surgery. The simulation uses a haptic feedback device, the Phantom Omni for the drill input and is rendered in stereo 3D using the NVidia stereo API.

The bone is represented by voxels, which represent if an area of space contains anatomy or does not. The voxels make collision detection fast but require significant processing for visual realism.

The simulation space is spatially subdivided into cubes and each cube is able to execute the Marching Cubes algorithm and then Laplacian HC Smoothing independently of each other. Thus it is possible to process the cubes separately on multiple cores. The cubes are connected to each other using a low overhead multithreaded staging system, so the entire model looks seamless.

The simulator includes other features such as: an improved haptic system for greater accuracy and stability, simulated dust, stereoscopic 3D rendering, bone transparency, various training tools such as an undo system, and the ability to replay an experienced surgeon's simulation for student training.

The simulation uses a custom engine designed from the ground up using C++ and DirectX.

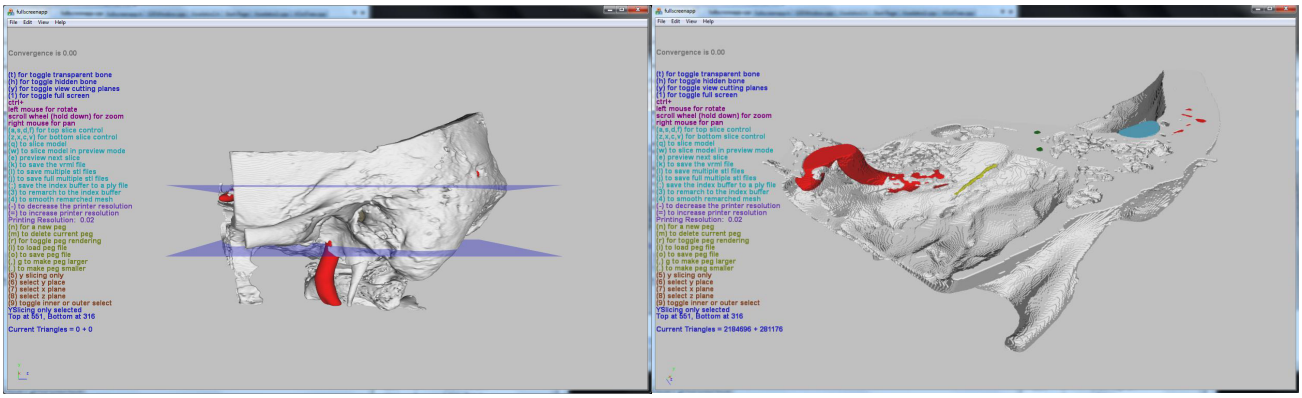


- Video of the simulation http://youtu.be/qIi9hvZL0hE
- Video of the CTV news report of the lab http://youtu.be/r6ur_E4FlnA
- PowerPoint slide of presentation https://drive.google.com/open?id=0B0IHjng8Lgb_akJuRENhY2VELTQ

Bone Slicing and 3D Printing

I developed software to take a 3D model generated from a CT scan and slice the 3D model to allow the interior to be printed properly using a 3D printer. Without the slicing, the interior of the bone would contain trapped dust from the 3D printer.

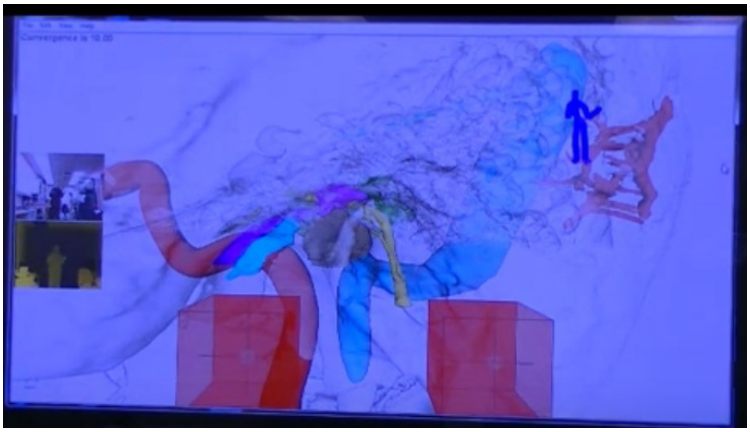
The software was developed using C++ and DirectX. Other than the code for the marching cubes algorithm it was developed from scratch.



- Demo Video of the slicing https://youtu.be/xyi05WjOQMM
- Manual of Slicing Software https://drive.google.com/open?id=0B0IHjng8Lgb_NGVQMzFkZHFuNkk

Kinect gesture based stereoscopic 3D anatomy display

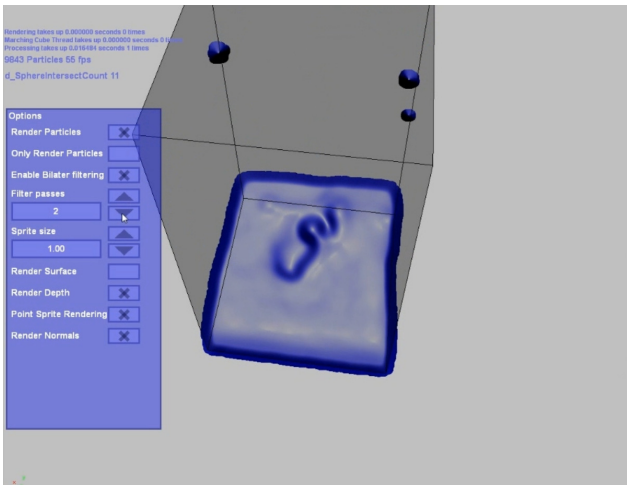
One of my first projects in the simulation lab was to develop stereo 3D visualization software. We had various different versions with various anatomy. One version I connected the Kinect to the visualization. I wrote all the interfacing code.



- Video of the Kinect demo http://youtu.be/oSh93OTNXZY

Fluid Simulation

A long term goal of the lab was to have a soft tissue simulation. Unfortunately not enough time was available to develop it but there are two demos.

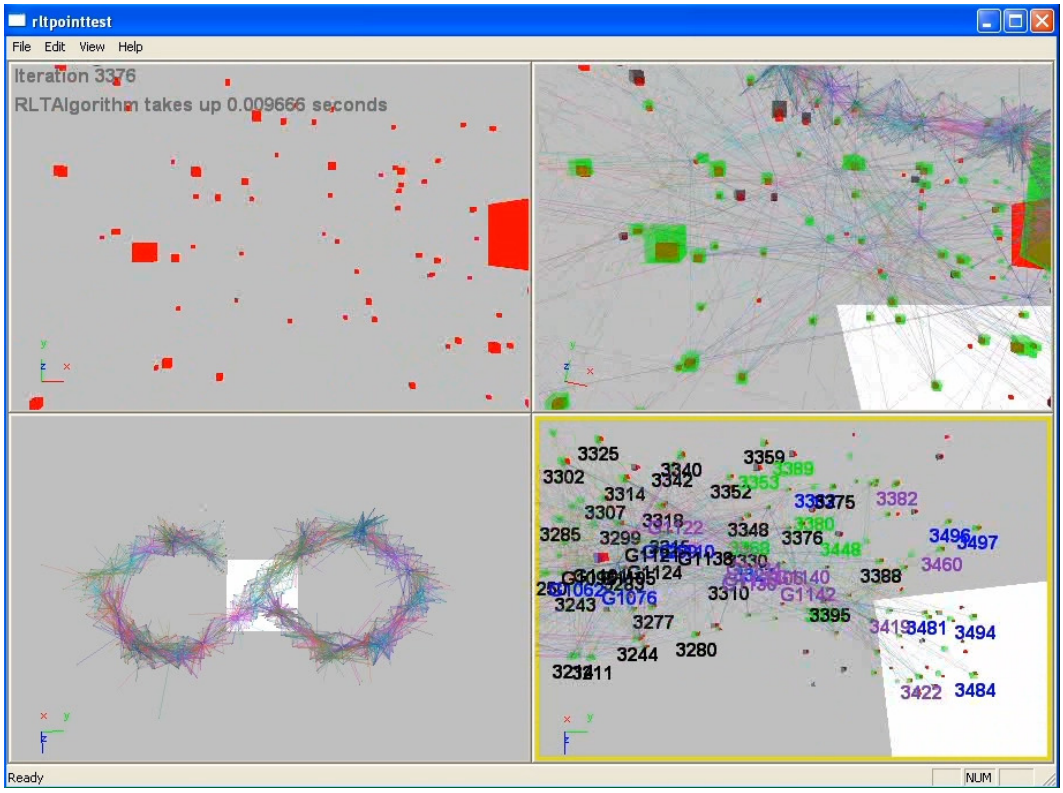


- Video of fluid simulation using marching cubes https://youtu.be/JESZkka6tDs
- Video of fluid simulation using GPU rendering https://youtu.be/X358x3_HZHs

PhD Thesis

My PhD thesis was developing the Relative Mapping algorithm. It is a new way of looking at the Simultaneous Localization and Mapping (SLAM) problem that does not use Probability, Iterative Closest Point, or Scan Matching techniques. A map of landmarks is generated by using the average relative location difference between landmarks. This means the algorithm does not use any known, estimated or predicted movement of position data. In addition, the Relative Mapping Algorithm has the capability to identify dynamic landmarks using a binning algorithm. The algorithm is shown to have a fast constant time $O(n_a \log n_a)$ computation complexity where n_a is the average quantity of points that are visible. In limited testing, the accuracy of the Relative Mapping Algorithm is shown to be comparable to the Extended Kalman Filter.

More information of the algorithm including videos are available on my web page at: http://www.jkrobots.com/. A copy of my thesis and a video of my thesis presentation rehearsal are available at http://www.jkrobots.com/thesis/.

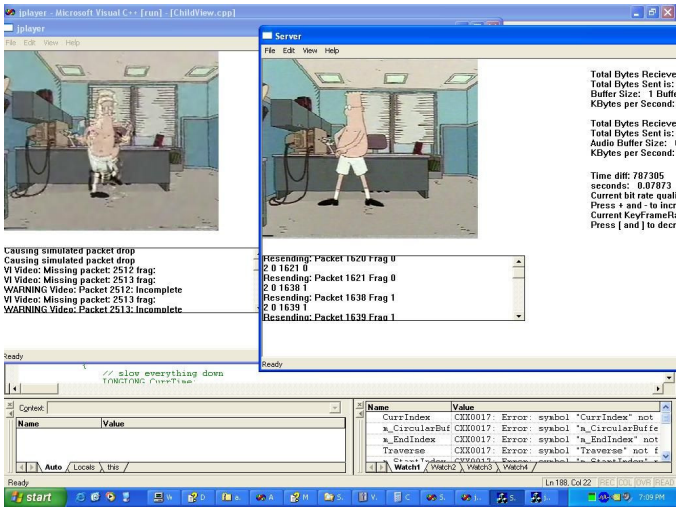


Master's Thesis

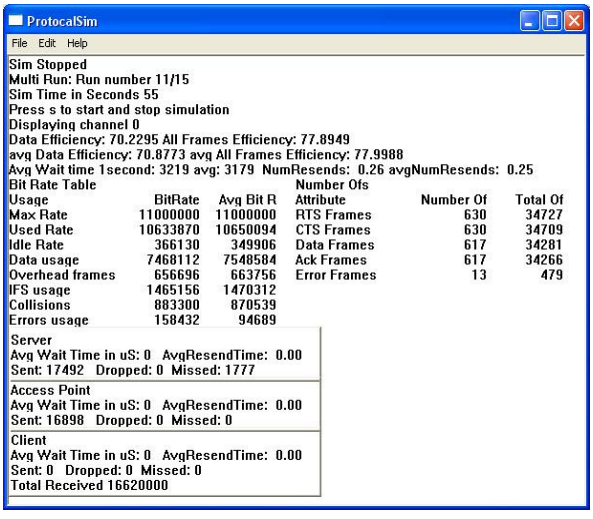
My Masters thesis was on wireless media streaming. It consisted of two parts. The first part was a wireless network simulator. The simulator used a software engineering approach of modelling each component of the network stack as an object oriented class. Simulated results were compared to theoretical results to verify the accuracy of the simulator. One of the test programs consisted of a mock file transfer protocol in which actual data is sent through the network stack. A client process was able to receive a file from a server process. The simulator also has the capability of simulating multiple channels by simply making more instances of the simulator's physical layer and a process's network interface.

The second part was on developing a transport layer to stream wireless media on an actual network. The transport layer was able to perform resend requests in case transmission errors occurred. This was used in conjunction with simulated transmission errors to observe the effects of streaming media with various quantities of packet loss.

A copy of the thesis available at:
http://www.jkrobots.com/misc/misc.html#Masters%20Thesis



Video streaming server/client GUI. Wrote transport code and DirectShow interface.

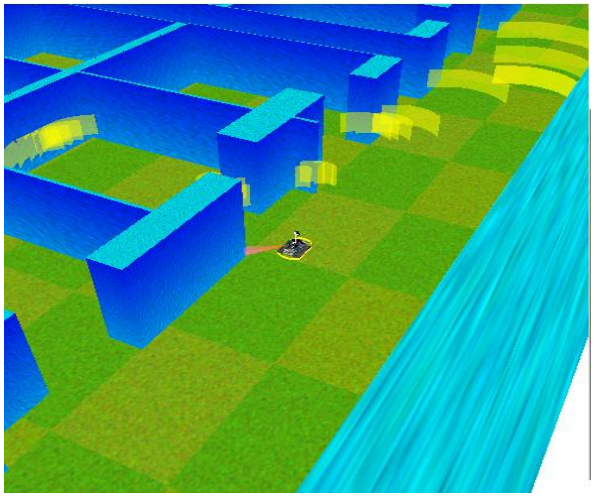


Network protocol simulator

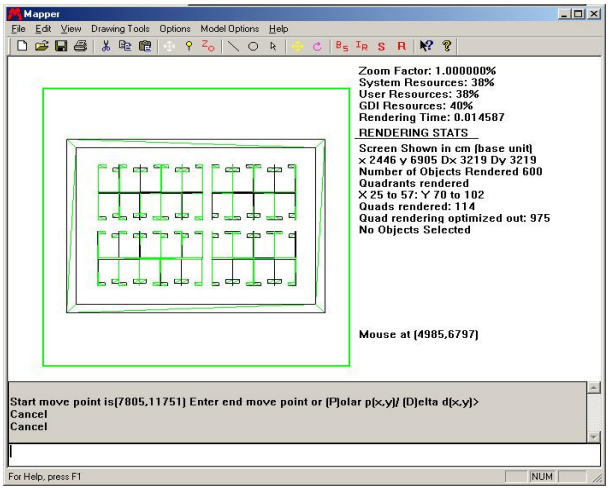
Undergraduate Thesis

My undergraduate thesis was an autonomous mobile robot that I created from a radio controlled car chassis using a microcontroller board, and various off the shelf sensors. A simulator was created that contained the same application program interface (API) as the embedded operating system that I also created. Using the simulator, I developed a wall following Artificial Intelligence (AI) that used infrared sensors to follow walls and a sonar to detect hallways. The approach of using a simulator was validated by the fact that the AI code was ported to the actual robot, tested, and then video was taken of it working within a single day.

Videos and pictures of the robot are available at:
http://www.jkrobots.com/JProject/Robot.htm



2D Robot simulation drawn in 3D
Robot sending out sonar pulses



2D Map editor

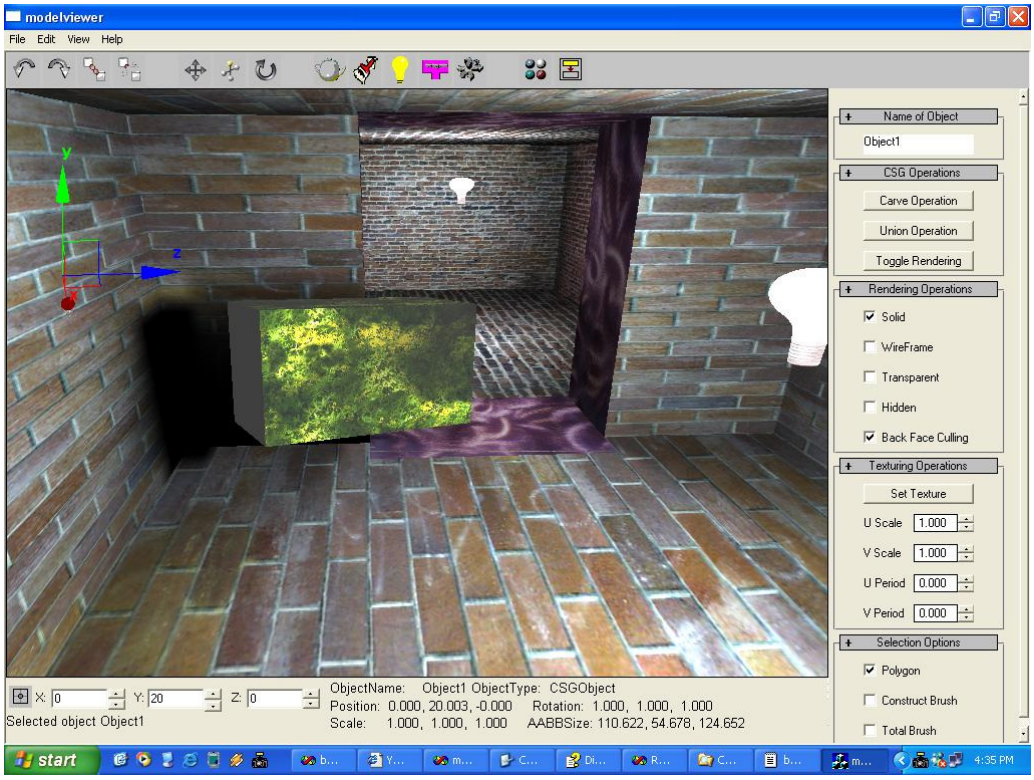
Constructed Robot

Robot navigating a hallway

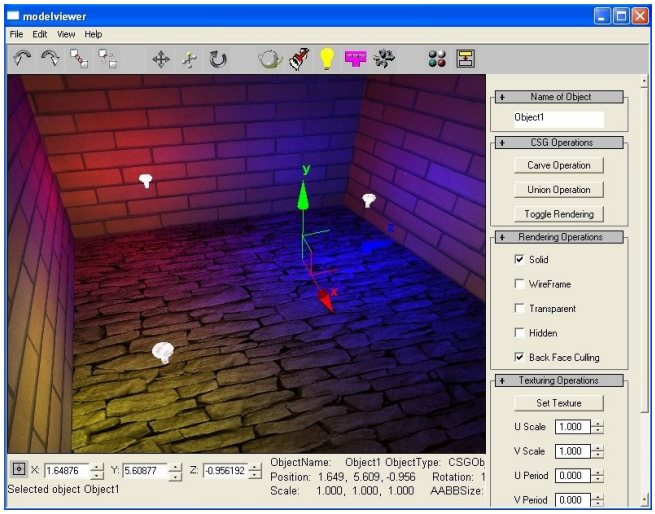
3D Map Editor

Developed a 3D map editor using C++, Visual Studios and DirectX. Other than the BSP/Vis code the editor was programmed from scratch including the callback architecture used for the GUI.

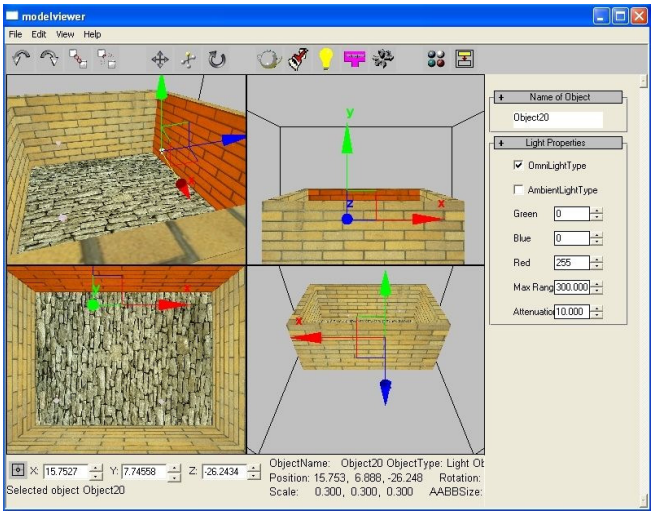
More pictures at: http://www.jkrobots.com/simhistory/simhistory.html



A map after BSP generation and lighting/shadowing

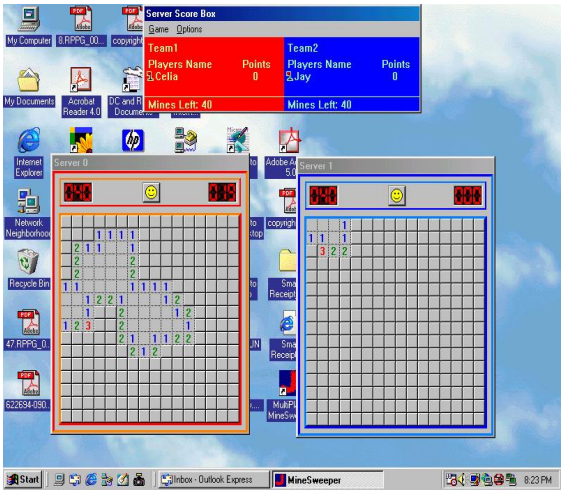


Multiple lighting

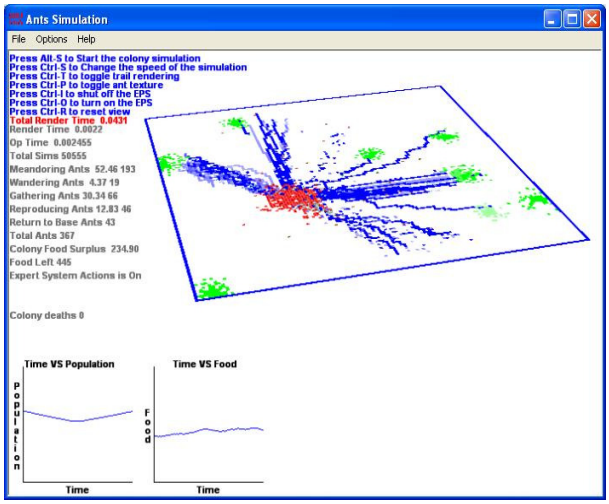


Editor in Multiple Window mode

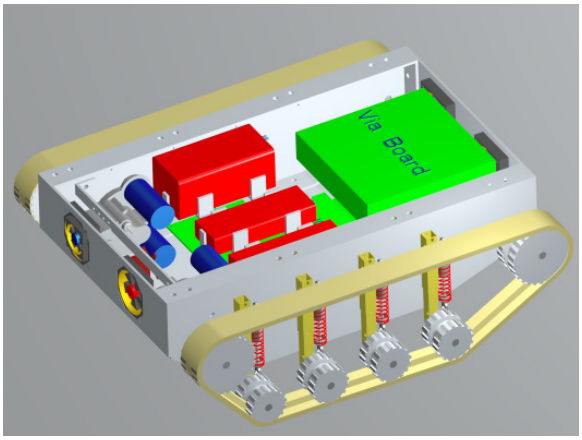
Other Projects



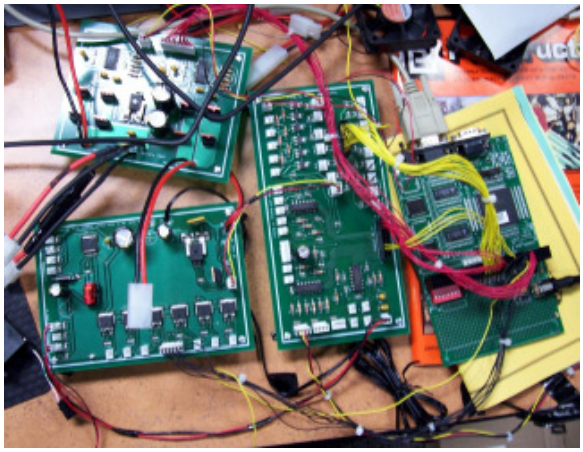
Multiplayer Minesweeper
http://worldminesweepernetwork.com/



Ants simulator
http://www.jkrobots.com/

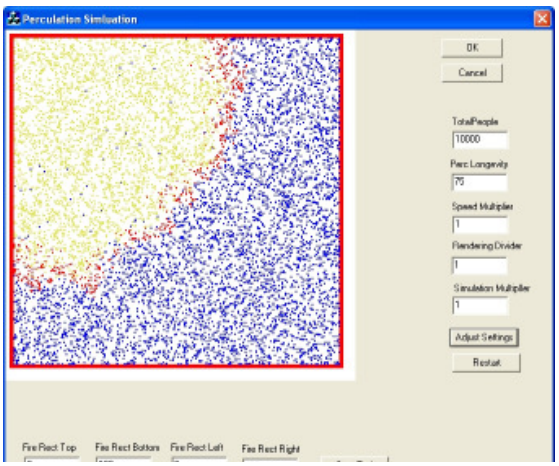


Pro/E Mechanical layout
http://www.jkrobots.com/robot2/Robot2.htm



PCB Design
http://www.jkrobots.com/robot2/Robot2.htm

Education Math Apps
https://itunes.apple.com/us/app/cross-number-discovery-puzzles/id960805283?mt=8
https://itunes.apple.com/us/app/cross-number-discovery-puzzles/id1079840697?mt=8



Percolation simulation
http://www.jkrobots.com/misc/misc.html

Teaching Experience

I have taught several courses at the university level including Introduction to Computer Science, System Engineering, Structures and Algorithms, and Object Orientation. Most of the courses I taught were at the second to third year level. I remember that at that point of my own education I had a good conceptual knowledge but I have not been able to apply that yet.

I often add programming examples to the courses I was teaching in order to allow the students to take the concepts learned in class and apply them to a program that contain a GUI. Some of my favorite examples: http://www.jkrobots.com/courses/

- The model view controller design architecture, shown by extending a single window application to multiple views and models.
- A very simple robot simulation.
- A hospital scheduling system.
- A factory scheduling system.

My favorite parts of teaching are the one on one discussions with students, and observing the different approaches the students may use to solve problems. One year teaching Object Orientation, a student asked if it is possible to use a better approach to creating Object Orientated classes than using multiple if statements. This came up when loading instances of several different kinds of classes from a file. I researched this and in the following class I described a way in Java that it is possible to load an instance of a class by the class name using a class factory.