ENGR 105 – Introduction to Scientific Computing
Final Project
Code due by 5pm on Wednesday 12/17/2014

*This project may be completed using pair programming. You can work with any partner of your choice – even persons you have programmed with twice before on class assignments.

Quick Summary of Project due Dates and Deliverables

Proposal due **11/19** by **11:59pm**

Project code, project summary, presentation materials due **12/17** by **5pm**

Project presentation of duration 3 - 7 min. on **12/18** from **12-2pm** or **3-5pm**

General Instructions

You will complete one of the following for the final course project:

- A project that you define (*e.g*. puzzle solver, atomistic simulation)
- A project that replicates portions of, fully replicates, extends upon, or is a derivative of an existing program written in a different programming language (*e.g*. a flash or iPad/iPhone/Android game).
- A project that is an extension of previous HW assignment problem (*e.g*. Billiard Ball problem with multiple balls)
- A project or derivative of a project from the list below.

You project should be significantly more challenging than your HW assignments. You are encouraged to build upon the skills you have learned throughout ENGR 105 or to amass additional skills through self-directed discovery of Matlab programming techniques.

You are advised to start your project early.

You can consult with the instructor, TA, or your peers for help with your project.

Deliverables

Project Proposal

You are to produce a document that:

1) Assigns a name of less than six words to your project (title of the document).

2) Identifies the person(s) involved in the coding process (underneath title).

3) Outlines a problem statement or statement of intent for your project.

4) Provides a high level description of the program(s) or working principles of your program.

5) Discusses any potential challenging aspects of the code.

6) Assigns a predicted difficulty rating (scale of 0-100) to the project. See the example projects to determine the complexity of your project in comparison to potential projects.

The purpose of the proposal is to convince the instructor that your project is of sufficient complexity, well-motivated, and achievable (*i.e.* complexity doesn't exceed the time allotted to the project/capabilities of programming team). Your document should be concise, yet complete. You are encouraged to keep the written portion to 1 page or less.

You are not expected to code your program(s) before submitting your proposal.

You will receive feedback from your instructor indicating "go" or "no-go". This feedback may include commentary or advice. You may be instructed to re-submit your proposal if your proposed project lacks sufficient complexity, is poorly motivated, or is too complicated for the time frame of the project ("no-go"). The project proposal will be graded in a binary fashion – on time or not on time.

Upload your project proposal to the proposal assignment callout on Canvas as a .docx, .doc, or .pdf file (.pdf encouraged) by 11:59pm on 11/19.

Project Code and Project Summary

You will submit your project code, a summary of your project (see below), and any additional presentation materials (see below) to the final project assignment callout on Canvas by 5pm on 12/17/2014. Project code or additional presentation materials not uploaded by this due date will not be included in the presentations – which could affect your final grade. All of the documents should be uploaded as a **single zipped folder named according to <Last name 1>_<First name 1>_<Last name 2>_<First name2>.zip**. For example, John Schmidt and Anne Thomas would upload the file "Schmidt_John_Thomas_Anne.zip". Extraneous files (e.g. autosave files or development files) should not be included in your upload.

It should be clear how to interact with your Matlab program(s) from the program naming conventions and/or by the inclusion of a “readme.txt” file. For instance, if you code Pong you could include an m-file that launches the project called “Start_Pong” or “Start_Here”.

You will include a project summary in your code .zip file. Your document should be concise, yet complete. You are encouraged to keep the written portion to 1 page or less (not including figures).

The project summary can be derived from your project proposal. However, it should reflect any changes to the intent or outcome of the project. This document should address the following:

1) Assign a name of less than six words to your project (title of the document).

2) Identify the person(s) involved in the coding process (underneath title).

3) Include an image that “describes” or encompasses your project.

4) Describe the intent of your project.

5) Provide a high level description of the program(s), working principles, and/or solution methodologies.

6) Discuss any challenges encountered during the coding process.

7) Describe any functionality described in the project proposal that was omitted from the final submission and the reason for omitting such functionality.

Presentation

You will present your project to the class with a 3 to 5 minute long presentation on 12/18/2014 from 12-2pm or 3-5pm (our final exam periods). The time limit of the presentation will be determined to the number of coding teams and will be communicated to you at least one week prior to the presentation date. Additionally, signups for the presentation time period (12-2pm or 3-5pm) will occur at least one week prior to the final. Any materials to be used during your presentation (*i.e*. code, powerpoint files, movies, etc.) should be included with your .zip code folder that you will upload to Canvas by 5pm on 12/17/2014.

Your project presentation will be a chance for you to show off all of your hard work. This presentation may include running your code (*e.g*. playing a game of Pong), describing the functionality of your code through a powerpoint presentation, and/or playing a movie of the outcome of your code (*e.g*. movie of an atomistic simulation that would take too long to run during the presentation time). Your presentation should leave the audience with a sense of what your code does and, potentially, how it does it.

Grading

Your final grade will be an amalgam/convolution of the following:

1) The clarity of your code (see below).

2) The difficulty / complexity of the project – it is expected that projects undertaken with pair programming will be of higher complexity than individual projects.

3) The uniqueness of your problem/project statement and solution method.

4) The clarity of your project summary.

5) A successful demonstration of your project.

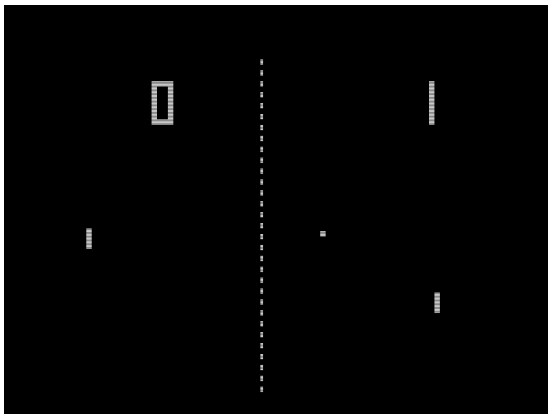
A significant part of your grade on this assignment will be based on how well your code is formatted and organized. You want to pay attention to the following elements:

- Comments: Use comments to document the structure of your code and your thought process. You can think of your comments as pseudo code explaining what each stage of your program does. A good rule is that you should have one line of comment for every three lines of code. Functions or subfunctions should include headers describing the inputs, outputs, and use of the function.
- Use of multiple functions/subfunctions: It is best practice to break a complex problem (code) into several functions or subfunctions. This aids debugging and results in code that is more easily understood by outside parties (*i.e*. your pair programming partner or your instructor, who is grading the final projects).
- Indentation and formatting: Messy code is hard to read and hard to debug. Make use of Matlab's Smart Indent functionality to automatically analyze and format your code in proper Matlab style.
- Variable and function names: The names that you give things are another form of documentation; meaningful names make your code easier to read, understand, and debug.

Example Projects

Several example projects from previous semesters are outlined below. Difficulty levels (scale of 0 – 100) have been included for your reference. Some of the projects include descriptions of flourishes (“extra credit”, “kudos”, “awesomeness”) that could improve your potential score. Recall, you need not select a project from the list below.

Pong: (Difficulty: 30 - 70)



Before Playstation, Xbox, or Angry Birds, there was Pong, at one time the ultimate in video game entertainment. Refer to the Wikipedia entry for this game (http://en.wikipedia.org/wiki/Pong) if you are unfamiliar with the rules/play. Pong is not so different in structure from the Billiard Ball simulation that you developed earlier in the course. Both involve a ball moving and colliding with obstacles. The difference in this case is that two of the obstacles are under user control. The goal of this project is to develop your own version of Pong in Matlab using the keyboard as input. For example, one player could move the left paddle up and down using the w and s keys while the right paddle is controlled with the up and down arrow keys. You would need to make use of callback functions. You will likely find the `plot()`, `set()`, `line()`, and `text()` commands useful in your program. You can use Matlab’s online help to get more details on the many options these functions offer. Your program should allow you to play a game to 11 keeping track of score appropriately.

The general structure of the code is outlined below in pseudo code

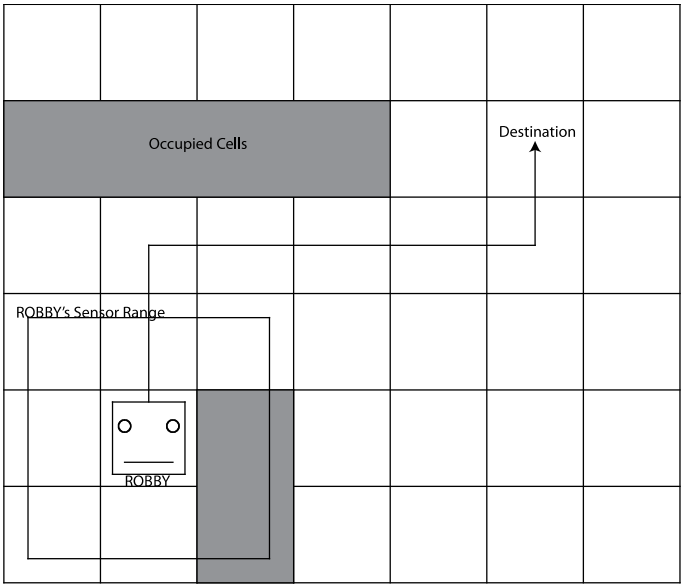
While not done
- Update ball state (position and velocity) taking into account collisions with walls and paddles
- Check for scoring and handle appropriately
- Update display

Note that psuedo code implicitly assumes that capturing the user input and moving the paddles is being handled with callback functions, which removes that functionality from the main loop.

You should handle collisions in your simulation by predicting whether the ball will collide with a paddle or wall during a simulation timestep, simulating forward to that impact point and then restarting the simulation step from there after reflecting the balls velocity to account for the collision.

For extra awesomeness (higher difficulty rating) you could consider adding features like spin or gravity to the ball flight and/or provide a single player mode where the computer controls one of the paddles. Additionally, you are encouraged to use velocity control for paddle movement as it will result in smooth gameplay.

Robby the Robot: (Difficulty: 30 - 50)



In this project you will write your own autonomous robot navigation system based on the grassfire (or other suitable) algorithm. The figure above shows our hapless robot, Robby, who is wandering around in a grid world trying to make it to the destination cell. Each cell in the grid is either filled with an obstacle or empty. At first Robby doesn't know which cells are occupied. However, he is equipped with a sensor that allows him to sense his eight neighboring cells and determine whether or not they are occupied. He also knows exactly where he is at every point in time so as he moves around he can build up his own map of the world which will get better as he sees more of the world. Robby's job in life is to get to the destination cell. Your job is to write a program that will get him there. More specifically you are asked to write a function with the following signature

```
function Robby(true_map, initial_row, initial_col, dest_row,
dest_col)
```

The `true_map` input is a logical array where the true entries correspond to empty cells and the false entries to occupied cells, `intial_row` and `initial_col` indicate Robby's starting point while `dest_row` and `dest_col` denote the coordinates of his final destination.

The Robby function could run the simulation according to the following pseudo code.

While not done
    Update Robby's personal map of the world based on his current sensor readings of his eight neighboring cells.
    Plan a path from Robby's current position to the goal based on his personal map (not the true map which you know but Robby doesn't)
    Take a step along the planned path
    Update the display showing where Robby is in the world.

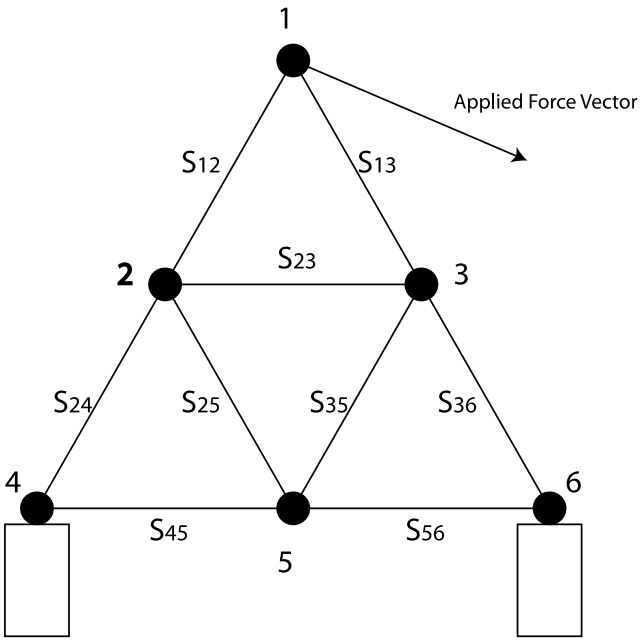
The program should also exit with an appropriate message if it decides that the robot cannot get there at all.

In displaying the world you can use the `imshow()` function to display the `true_map` and Robby's current position. Different colors could be used to indicate the identity of different cells – perhaps using red to indicate Robby's current position. For extra pizzazz, you can think of ways of using colors to indicate what Robby's current personal map looks like – for example using colors to distinguish between free and occupied cells that Robby knows about and ones that he doesn't know about. You could also use the `plot()` function to plot a line showing how the robot has traveled to through the space.

Think up some interesting test cases that you can use to show your code off on demo day. Extra credit is available depending on the complexity of the environment you explore and the speed with which your code runs.

Note: This is a simplified version of the kind of problem that the robots in the DARPA Urban Challenge had to solve.

Interactive Stress Analysis (Difficulty: 40 - 60)

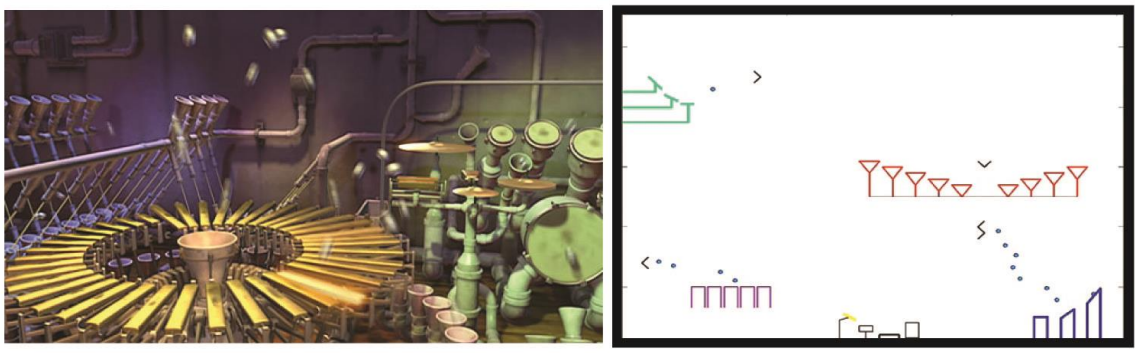


In this project you will analyze a system of equations to determine the stresses and strains in each of the members of the physical system shown above. Your program will allow a user to interactively vary the force vector applied to node 1 and visualize the induced stresses and strains. First, your program needs to create a figure showing the structure given in the figure above. Second, your program needs to provide a mechanism to allow the user to interactively vary the force vector applied at node 1. You can do this using the keyboard or, with a bit more work you can do this via the mouse using a click and drag interface to specify the vector. However you choose to do it you need to also draw a line or arrow indicating the force vector applied at node 1 as shown in the figure. Your program should continually recalculate the force balance system as the load vector is changed and display the results by varying the color of each of the members in the structure. The colors should vary smoothly from blue to red where blue indicates compression and red indicates tension. The more intense the color, the greater the associated force.

In setting up the force balance equations you should assume that node 4 is fixed to the ground so the ground applies any reaction forces required to balance the forces here. Node 6 is resting on the ground so the vertical forces are opposed by ground reaction but no horizontal forces exist. (In other words you get no useful equations for node 4 and one useful equation at node 6 corresponding to horizontal force balance). All of the other nodes in the structure are free standing, *i.e*. not in contact with the ground or any other surface.

For extra awesomeness, you should have your program perform the same analysis on another, more complicated structure of your own design that involves at least 10 members. You will get much kudos if your interface allows you to add load forces to every free node in the structure - not just a single node – or interactively vary the topology and connections of the structure.

Animusic: (Difficulty: 50 - 80)



(Left) A screenshot of the Animusic program in action and (right) a simplified animusic animation inspired by Pipe Dream (http://youtu.be/z5BlekxhV3A).

Animusic (http://www.animusic.com/) is a neat piece of software that allows designers to create physical simulations that produce music. Your task in this project will be to produce a simplified version of the pipe-dream animation shown on the left. More specifically, you are asked to develop a 2D version that creates sounds by firing projectiles at a set of targets to create 3 or 4 different sounds. The trick is to compute the firing velocities required to hit the various targets from the various spouts. This can be done using trial and error or, better yet, a golden section search. Once you have the trajectories you can store them and simply play them over and over again in your simulation. Then all you need to do is create a sequence of shots that play the musical piece of your preference. You can refer to some of our previous simulation codes (basketball problem) for inspiration.

One simple way to create sounds in Matlab is to create appropriate waveforms – like sine waves - and then play them using the sound function as follows:

```
>> N = 10000; sound(cos((2*pi/N)*1000*(0:N)));
>> N = 10000; sound(cos((2*pi/N)*2000*(0:N)));
>> N = 10000; sound(cos((2*pi/N)*4000*(0:N)));
```

You can vary the frequency of the sound and the duration of the clip to your taste.

You can get extra kudos if you run a highly intricate simulation and/or reproduce an awesome song. Points will be deducted if you simulate “Wrecking Ball” by Miley Cyrus.

Shotgun Sequencing (Difficulty: 30 - 50)

One the first day of class we discussed (very topically) how computers are used to analyze and reassemble DNA sequences through sequence alignment. Below you will find a few links to some interesting and relevant articles on sequence analysis which you should peruse.

http://en.wikipedia.org/wiki/Sequence_alignment
http://www.ornl.gov/sci/techresources/Human_Genome/project/info.shtml
http://en.wikipedia.org/wiki/Nucleotide
http://en.wikipedia.org/wiki/Sequence_assembly
http://en.wikipedia.org/wiki/BLAST

You will implement your own sequence reassembly system for this project. This system will take as input a cell array containing a number of strings – these are fragments of a complete sequence. Your job is to reassemble the fragments to reconstitute the original string.

To do this you will write the following two functions, `align` and `reassemble` which are detailed below:

```
function [out, overlap] = align (str1, str2)
```

This routine should try to merge the two input strings, `str1` and `str2`, together. The output should be the merged string. For example if the inputs were 'abcdefghijklmno' and 'fghijklmnopqrstu' the output should be 'abcdefghijklmnopqrstu'. To do this you need to repeatedly compare the last k characters in `str1` to the first k characters in `str2` to see if they are the same. Furthermore if one string is completely contained in the other your code should return the larger string and indicate the length of the overlap. You can use the `strfind()` function in Matlab to check for this case. Your code should require that the overlap between the two strings is at least 5 characters long. If the two strings cannot be merged the routine should return an empty array. The overlap output indicates the number of characters that overlap between `str1` and `str2`.

You should then use this `align` function to implement the sequence construction scheme described on the first Wikipedia page.

1) Calculate pairwise alignments of all fragments.
2) Merge the two fragments with the largest overlap.
3) Repeat until there is only one fragment remaining or no more merges are found.

You should implement a function with the following signature that takes as input a cell array containing the string fragments and returns as output a cell array containing the final set of merged strings.

```
function out = reassemble (fragments)
```

You are provided with a file called `ShotgunData.mat`. This file contains a set of Matlab variables which you can load into your session using the `load()` command. The file

contains 3 cell arrays – `fragments1`, `fragments2`, and `fragments3` that you can use for testing. The original messages corresponding to the first two cell arrays are provided in the strings `str1` and `str2`. You can compare your answer to these strings to see if your code is working correctly. You may find Matlab's `strcmp()` function useful in this regard. You aren't given the answer for the `fragments3` input but you will know when you are decoding this message correctly.

Other Potential Project Ideas

1) A large-scale atomistic simulation of some physical phenomena.

2) A cell processing toolbox that measures the density, circularity, and spatial distribution of cells in a culture.

3) A Matlab version of your favorite game (e.g. Flappy Bird, Bejeweled, Pac Man, etc.)

4) A Billiard Ball simulator like that in HW6, but with many billiard balls and accounting for the radius of each ball.