COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

099. Undergraduate Research/Independent Study. (C) A maximum of 2 c.u. of CIS 099 may be applied toward the B.A.S. or B.S.E. degree requirements.

An opportunity for the student to become closely associated with a professor (1) in a research effort to develop research skills and techniques and/or (2) to develop a program of independent in-depth study in a subject area in which the professor and student have a common interest. The challenge of the task undertaken must be consistent with the student's academic level. To register for this course, the student must submit a detailed proposal, signed by the independent study supervisor, to the SEAS Office of Academic Programs (111 Towne) no later than the end of the "add" period.

101. Introduction to Computer Science. (A) Corequisite(s): Math 104 or Math 150.This counts as a Formal Reasoning course for College students.

This course is an introduction to the basic principles and great ideas of computer science intended for non-engineering students. It covers some of the essential topics of contemporary computer science from a mathematical perspective. No programming experience necessary. Open to all non-SEAS students. SEAS students may not take for credit toward their engineering degree.

106. (ANTH258) Visualizing the Past.. (C)

Most people's information about the Past is drawn from coffee table picture books, popular movies, video games, documentaries about discoveries of "ancient, mysterious, and lost" civilizations, and tours often lead by guides of limited or even dubious credentials. How are these ideas presented, formed, and circulated? Who creates and selects the information presented in this diverse media? Are these presentations accurate? Do they promote or hurt scientific explanations? Can the artistic, aesthetic, and scientific realms be bridged to effectively promote and interpret the past? How can modern technologies be applied to do a better job at presenting what is difficult to experience firsthand? This class will focus on case studies, critiques, and methods of how archaeology and the past are created, presented and used in movies, museums, games, the internet, and art.

Each year, the studio-seminar focuses on a project. In addition to exploring general concepts of archaeology and the media, students will work in teams to produce an interactive, digital media exhibit using the latest modeling visualization programs for presenting the sacred landscape of the Inca capital of Cuzco, Peru. Cuzco is one of the most important UNESCO World Heritage sites and visited by nearly a million tourists a year. Potential class projects include fly-throughs of architectural and landscape renderings, simulations of astronomy and cosmology, modeling of human behavior within architectural and landscape settings, and study artifacts in the Penn Museum.

L/R 110. Introduction to Computer Programming. (C) See the CIS 110 website for information about registration in recitations and permission to register for closed sections of CIS 110. Counts as a Formal Reasoning course for College students.

Introduction to Computer Programming is the first course in our series introducing students to computer science. In this class you will learn the fundamentals of computer programming in Java, with emphasis on applications in science and engineering. You will also learn about the broader field of computer science and algorithmic thinking, the fundamental approach that computer scientists take to solving problems.

112. (PPE 112) Networked Life. (C)

How does Google find what you're looking for... and exactly how do they make money doing so? What properties might we expect any social network (such as the Penn Facebook) to reliably have, and are there "simple" explanations for them? How does your position in a social or economic network (dis) advantage you, and why? What might we mean by the economics of spam? What do game theory and the Paris subway have to do with Internet routing? Networked Life looks at how our world is connected -- socially, economically, strategically and technologically -- and why it matters.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

L/R 120. Programming Languages and Techniques I. (C) This counts as a Formal Reasoning course for College students.

A fast-paced introduction to the fundamental concepts of programming and software design. This course assumes some previous programming experience, at the level of a high school computer science class or CIS110. (If you got at least 4 in the AP Computer Science A or AB exam, you will do great.) No specific programming language background is assumed: basic experience with any language (for instance Java, C, C++, VB, Python, Perl, or Scheme) is fine. If you have never programmed before, you should take CIS 110 first.

L/R 121. Programming Languages and Technigues II: Data Structures in Java. (B) Prerequisite(s): CIS 120, CIS 160.

This is an introductory course about Basic Algorithms and Data Structures using the Java programming language. We introduce elementary concepts about the complexity of an algorithm and methods for analyzing the running time of software. We describe data structures like stacks, queues, lists, trees, priority queues, maps, hash tables and graphs, and we discuss how to implement them efficiently and how to use them in problems-solving software. A larger project introducing students to some of the challenges of software development concludes the course.

125. (EAS 125) Technology and Policy.

Have you ever wondered why sharing music and video generates such political and legal controversies? Is information on your PC safe and should law enforcement be able to access information you enter on the Web? Will new devices allow tracking of your every move and every purchase? CIS 125 is focused on developing an understanding of existing and emerging technologies, along with the political, societal and economic impacts of those technologies. The technologies are spread across a number of engineering areas and each of them raise issues that are of current concern or are likely to be a future issue.

L/R 140. (COGS001, LING105, PHIL044, PPE 140, PSYC207) Introduction to Cognitive Science. (A) This counts as a Formal Reasoning course for College students.

How do minds work? This course surveys a wide range of answers to this question from disciplines ranging from philosophy to neuroscience. The course devotes special attention to the use of simple computational and mathematical models. Topics include perception, learning, memory, decision making, emotion and consciousness.

L/R 160. Mathematical Foundations of Computer Science. (B) Corequisite(s): CIS 120.

What are the basic mathematical concepts and techniques needed in computer science? This course provides an introduction to proof principles and logics, functions and relations, induction principles, combinatorics and graph theory, as well as a rigorous grounding in writing and reading mathematical proofs.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

L/R 190. C++ Programming. (C) Prerequisite(s): CIS 240.

This course will provide an introduction to programming in C++ and is intended for students who already have some exposure to programming in another language such as Java, C++ provides the programmer with a greater level of control over machine resources and are commonly used in situations where low level access or performance are important. This course will illuminate the issues associated with programming at this level and will cover issues such as explicit memory management, pointers, the compilation process and debugging. The course will involve several programming projects which will provide students with the experience they need to program effectively in these languages. This course assumes programming experience equivalent to CIS 110, CIS 120 or ESE 112.

L/R 191. Using and Understanding Unix and Linux. (C) Prerequisite(s): CIS 110 or equivalent.

Unix, in its many forms, runs much of the world's computer infrastructure, from cable modems and cell phones to the giant clusters that power Google and Amazon. This half-credit course provides a thorough introduction to Unix and Linux. Topics will range from critical basic skills such as examin and editing files, compiling programs and writing shell scripts, to higher level topics such as the architecture of Unix and its programming model. The material learned is applicable to many classes, including CIS 240, CIS 331, CIS 341, CIS 371, and CIS 380.

L/R 192. Python Programming. (C) Prerequisite(s): CIS 120 or ESE 112.

Python is an elegant, concise, and powerful language that is useful for tasks large and small. Python has quickly become a popular language for getting things done efficiently in many in all domains: scripting, systems programming, research tools, and web development. This course will provide an introduction to this modern high-level language using hands-on experience through programming assignments and a collaborative final application development project.

193. C# Programming. (C) Prerequisite(s): CIS 110.

C# is the premier programming language for the .NET framework. Over the last decade, the language has evolved to meet the needs of a variety of programming styles while supporting the ever-growing capabilities of the the .NET runtime and libraries. This course provides a thorough introduction to the C# language and the .NET framework, building on the skills gained in the introductory programming courses (CIS 110, CIS 120, or ESE 112). In addition to providing the student with a solid background in C#, this course also explores topics that the .NET platform exposes such as object oriented design, .NET runtime internals, and others based on class interest. A series of short, weekly homework assignments reinforces the concepts introduced in class and a group-based final project of the students' design allows them to apply their C# knowledge toward a substantial problem.

L/R 194. Haskell. (C)

L/R 195. IPhone App Development. (C)

L/R 196. Ruby on Rals Web Develp.

L/R 197. JAVASCRIPT. (C)

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

240. Introduction to Computer Architecture. (A) Prerequisite(s): CIS 110 or equivalent experience.

You know how to program, but do you know how computers really work? How do millions of transistors come together to form a complete computing system? This bottom-up course begins with transistors and simple computer hardware structures, continues with low-level programming using primative machine instructions, and finishes with an introduction to all aspects of computer systems architecture and serves as the foundation for subsequent computer systems courses, such as Digital Systems Organization and Design (CIS 371), Computer Operating Systems (CIS 380), and Compilers and Interpreters (CIS 341).

The course will consider the SPARC architecture, boolean logic, number systems,and computer arithmetic; macro assembly language programming and subroutine linkages; the operating system interface and input/output; understanding the output of the C compiler; the use of the C programming language to generate specific assembly language instructions.

L/R 261. Discrete Probability, Stochastic Processes, and Statistical Inference. (A) Prerequisite(s): CIS 160 or equivalent.

This course tightly integrates the theory and applications of discrete probability, discrete stochastic processes, and discrete statistical inference in the study of computer science. The course will introduce the Minimum Description Length Paradigm to unite basic ideas about randomness, inference and computation. Students will be expected to use the Maple programming environment in homework exercises which will include numerical and symbolic computations, simulations, and graphical displays.

L/R 262. Automata, Computability, and Complexity. (A) Prerequisite(s): CIS 160.

This course explores questions fundamental to computer science such as which problems cannot be solved by computers, can we formalize computing as a mathematical concept without relying upon the specifics of programming languages and computing platforms, and which problems can be solved efficiently. The topics include finite automata and regular languages, context-free grammars and pushdown automata, Turing machines and undecidability, tractability and NP-completeness. The course emphasizes rigorous mathematical reasoning as well as connections to practical computing problems such as test processing, parsing, XML query languages, and program verification.

277. Introduction to Computer Graphics Techniques. (C) Prerequisite(s): CIS 120.

This course is focused on programming the essential geometric and mathematical concepts underlying modern computer graphics. Using 2D and 3D implementations, it covers fundamental topics on scene graphs, computational geometry, graphics algorithms, and user interface design. Programming languages introduced include C++, OpenGL, FLTK and Python.

320. Introduction to Algorithms. (B) Prerequisite(s): CIS 120, 121, 160, 262.

How do you optimally encode a text file? How do you find shortest paths in a map? How do you desgin a communication network? How do you route data in a network? What are the limits of efficient computation? This course gives a comprehensive introduction to design and analysis of algorithms, and answers along the way to these and many other interesting computational questions. You will learn about problem-solving; advanced data structures such as universal hashing and red-black trees; advanced design and analysis techniques such as dynamic programming and amortized analysis; graph algorithms such as minimum spanning trees and network flos; NP-completeness theory; and approximation algorithms.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

330. Design Principles of Information Systems. (A) Prerequisite(s): CIS 121 and 160.

Introduction to database management systems and principles of design. The Entity-Relationship model as a modeling tool. The relational model: formal languages, the industry standard SQL, relational design theory, query optimization. Storing and querying XML data. Recursive queries, Views and data integration. Overview of system level issues: physical data organizaiton, indexing techniques, and transactions. Connecting databases to the Web. Course work requires programming in several different query languages, several written homeworks and a team project.

334. Advanced Topics in Algorithms. (M) Prerequisite(s): CIS 320.

Can you check if two large documents are identical by examining a small number of bits? Can you verify that a program has correctly computed a function without ever computing the function? Can students compute the average score on an exam without ever revealing their scores to each other? Can you be convinced of the correctness of an assertion without ever seeing the proof? The answer to all these questions is in the affirmative provided we allow the use of randomization. Over the past few decades, randomization has emerged as a powerful resource in algorithm desgin. This course would focus on powerful general techniques for designing randomized algorithms as well as specific representative applications in various domains, including approximation algorithms, cryptography and number theory, data structure design, online algorithms, and parallel and distributed computation.

341. Compilers and Interpreters. (M) Prerequisite(s): Two semesters of programming courses, e.g., CIS 120-121, and CIS 240.

You know how to program, but do you know how to implement a programming language? In CIS341 you'll learn how to build a compiler. Topics covered include: lexical analysis, grammars and parsing, intermediate representations, syntax-directed translation, code generation, type checking, simple dataflow and control-flow analyses, and optimizations. Along the way, we study objects and inheritance, first-class functions (closures), data representation and runtime-support issues such as garbage collection. This is a challenging, implementation-oriented course in which students build a full compiler from a simple, typed object-oriented language to fully operational x86 assembly. The course projects are implemented using OCaml, but no knowledge of OCaml is assumed.

350. Software Design/Engineering. (M) Prerequisite(s): CIS 240.

You know how to write a "program". But how do you create a software "product" as part of a team, with customers that have expectations of functionality and quality? This course introduces students to various tools (source control, automated build systems, programming environments, test automation, etc.) and processes (design, implementation, testing, and maintenance) that are used by professionals in the field of software engineering. Topics will include: software development lifecycle; agile and test-driven development; source control and continuous integration; requirements analysis; object-oriented design and testability; Android application development; software testing; refactoring; and software quality metrics.

368. User Interfaces and the Web. (C) Prerequisite(s): CIS 110, CIS 120, CIS 121, CIS 277.

This course will teach the fundamentals of Human-Computer Interaction (theory, design, implementation, experimentation, evaluation) in the context of current web interaction mechanisms, technologies, and applications. The course content will emphasize and leverage open source technologies to design, prototype, implement, and test user-interfaces and functionality in the context of today's most intriguing web trend, social networking.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

371. Computer Organization and Design. (B) Prerequisite(s): CIS 240.

This is the second computer oganization course and focuses on computer hardware design. Topics covered are: (1) basic digital system design including finite state machines, (2) instruction set design and simple RISC assembly programming, (3) quantitative evaluation of computer performance, (4) circuits for integer and floating-point arithmatic, (5) datapath and control, (6) micro-programming, (7) pipeling, (8) storage hierarchy and virtual memory, (9) input/output, (10) different forms of parallelism including instruction level parallelism, data-level parallelism using both vectors and message-passing multi-processors, and thread-level parallelism using shared memory multiprocessors. Basic cache coherence and synchronization.

380. Computer Operating Systems. (A) Prerequisite(s): CIS 240.

This course surveys methods and algorithms used in modern operating systems. Concurrent distributed operation is emphasized. The main topics covered are as follows: process synchronization; interprocess communication; concurrent/distributed programming languages; resource allocation and deadlock; virtual memory; protection and security; distributed operation; distributed data; performance evalaution.

390. Machine Perception. (M) Prerequisite(s): MATH 240, PHYS 150 or MEAM 110/147.

The rapidly evolving field of robotics includes systems designed to replace, assist, or even entertain humans in a wide variety of tasks. Recent examples include planetary rovers, robotic pets, medical surgical-assistive devices, and semi-autonomous search-and-rescue vehicles. This introductory-level course presents the fundamental kinematic, dynamic, and computational principles underlying most modern robotic systems. The main topics of the course include: coordinate transformations, manipulator kinematics, mobile-robot kinematics, actuation and sensing, feedback control, vision, motion planning, and learning. The material is reinforced with hands-on lab exercises including basic robot-arm control and the programming of vision-guided mobile robots.

391. Introduction to Artificial Intelligence. (M) Prerequisite(s): CIS 121 and CIS 262.

Artificial Intelligence is considered from the point of view of a resource-limi knowledge-based agent who must reason and act in the world. Topics include log automatic theorem proving, search, knowledge representation and reasoning, natural language processing, probabilistic reasoning, and machine learning. Programming assignments in Python.

398. Quantum Computer and Information Science. (C) Prerequisite(s): CIS 260, 262 and Math 240.

The purpose of this course is to introduce undergraduate students in computer computer science and engineering to quantum computers (QC) and quantum information science (QIS). This course is meant primarly for juniors and seniors in Computer Science. No prior knowledge of quantum mechanics (QM) is assumed. Enrollment is by permission of the instructor.

400. Senior Project. (A) Prerequisite(s): Senior standing or permission of instructor.

The goal of the senior design coruse is to provide students with an opportunity to define, design and execute a significant project. Project subjects may revolve around software, hardware or computational theory. Students must have an abstract of their Senior Project, which is approved and signed by a Project Advisor early in the Fall semester. The project is expected to span two semesters; students must enroll in CIS 401 during the second semester. At the end of the first semester, students are required to submit an intermediate report and give a presentation describing their project and progress. Grades are based on technical writing skills (as per submited report) presentation skills and progress on the project. These are evaluated by the Project Adviser and the Course Instructor.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

401. Senior Project. (B) Prerequisite(s): CIS 400, senior standing or permission of instructor.

Continuation of CIS 400. Design and implementation of a significant piece of work: software, hardware or theory. Students are required to submit a final written report and give a final presentation and demonstration of their project. Grades are based on the report, the presentation and the satisfactory completion of the project. These are evaluated by the Project Adviser and the Course Instructor.

419. (CIS 519) Introducation to Machine Learning. (C)

Machine learning has been essential to the success of many recent technologies, including autonomous vehicles, search engines, genomics, automated medical diagnosis, image recognition, and social network analysis, among many others. This course will introduce the fundamental concepts and algorithms that enable computers to learn from experience, with an emphasis on their practical application to real problems. This course will introduce supervised learning (decision trees, logistic regression, support vector machines, Bayesian methods, neural networks and deep learning), unsupervised learning (clustering, dimensionality reduction), and reinforcement learning. Additionally, the course will discuss evaluation methodology and recent applications of machine learning, including large scale learning for big data and network analysis.

430. Introduction to Human Language Technology. (A) Prerequisite(s): CIS 121.

This course is an automatic summarization that can help alleviate the information overload problem caused by the unprecedented amount of online textual information. The building of a summarization system requires good understanding of the properties of human language and the use of various natural language tools. In this course we will build several summarization systems of increasing complexity and sophistication. In the process we will learn about various natural language processing tools and resources such as part of speech tagging, chunking, parsing, Wordnet, and machine learning toolkits. We will also cover probability and statistics concepts used in summarization, but also applicable to a wide range of other language-related tasks.

441. (CIS 541) Embedded Software for Life-Critical Applications. (C) Prerequisite(s): CIS 240 or equivalent; ESE 350 recommended.

This course is focused on cyber physical systems with emphasis on real-time issues. Cyber physical systems are integrations of computation and communication with physical processes. Embedded computers monitor and control physical processes in real-time. As these embedded computer transformed from word processors to global communications devices for information gathering and sharing, embedded computers will change from small self-contained systems to cyber-physical systems by sensing,

The course is to study principles, methods, and techniques for building high-assurance cyber-physical systems. Topics will include requirements capture and modeling, mental models, assurance cases, hazard analysis, real-time programming and communication, real-time scheduling and virtual machines, feedback control in computer systems, verification and validseries of projects that will implement safety-critical embedded systems (e.g., pacemaker, infusion pump).

450. (CIS 550) DATABASE & INFO SYSTEMS. (A)

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

455. (CIS 555) Internet and Web Systems. (C) Prerequisite(s): Familiarity with threads and concurrency, strong Java programming skills.

This course focuses on the challenges encountered in building Internet and web systems: scalability, interoperability (of data and code), security and fault tolerance, consistency models, and location of resources, services, and data. We will examine how XML standards enable information exchange; how web services support cross-platform interoperability (and what their limitations are); how to build high-performance application servers; how "cloud computing" services work; how to perform Akamai-like content distribution; and how to provide transaction support in distributed environments. We will study techniques for locating machines, resources, and data (including directory systems, information retrieval indexing, ranking, and web search); and we will investigate how different architectures support scalability (and the issues they face). We will also examine ideas that have been proposed for tomorrow's Web, and we will see some of the challenges, research directions, and potential pitfalls. An important goal of the course is not simply to discuss issues and solutions, but to provide hands-on experience with a substantial implementation project.

This semester's project will be a peer-to-peer implementation of a Googe-style search engine, including distributed, scalable crawling; indexing with ranking; and even PageRank. As a side-effect of the material of this course you will learn about some aspects of large-scale software development assimilating large APIs.

460. (CIS 560) Computer Graphics. (A) Prerequisite(s): One year programming experience (C, JAVA, C++).

A thorough introduction to computer graphics techniques, covering primarily 3D modeling and image synthesis. Topics cover: geometric transformations, geometric algorithms, software systems (OpenGL), 3D object models (surface and volume), visible surface algorithms, image synthesis, shading and mapping, ray tracing, radiosity, global illumination, photon mapping, anti-aliasing and compositing.

462. (CIS 562) Computer Animation. (C) Prerequisite(s): Previous exposure to major concepts in linear algebra (i.e. vector matrix math), curves and surfaces, dynamical systems (e.g. 2nd order mass-spring-damper systems) and 3D computer graphics has also been assumed in the preparation of the course materials.

This course covers core subject matter common to the fields of robotics, character animation and embodied intelligent agents. The intent of the course is to provide the student with a solid technical foundation for developing, animating and controlling articulated systems used in interactive computer game virtual reality simulations and high-end animation applications. The course balances theory with practice by "looking under the hood" of current animation systems and authoring tools and exams the technologies and techniques used from both a computer science and engineering perspective. Topics covered include: geometric coordinate systems and transformations; quaternions; parametric curves and surfaces; forward and inverse kinematics; dynamic systems and control; computer simulation; keyframe, motion capture and procedural animation; behavior-based animation and control; facial animation; smart characters and intelligent agents.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

477. (LING549) Mathematical Methods/Techniques for Linguistics and Natural Language Processing. (M) Prerequisite(s): PHIL 006 or instructor's permission.

Basic concepts of set theory, relations and functions, properties of relations. Basic concepts of algebra. Grammars, languages, and automata- finite state grammars, regular expressions, context-free and context-sensitive grammars, unrestricted grammars, finite automata, pushdown automata and other related automata, Turing machines, Syntax and semantics of grammar formalisms. Strong generative capacity of grammars, Grammers as deductive systems, parsing as deduction. Relevance of formal gammars to modeling biological sequences. The course will deal with these topics in a very basic and introductory manner--ideas of proofs and not detailed proofs, and more importantly with plenty of linguistic examples to bring out the linguistic relevance of these topics.

The course will deal with these topics in a very basic and introductory manner--ideas of proofs and not detailed proofs, and more importantly with plenty of linguistic examples to bring out the linguistic relevance of these topics.

480. Real-Time and Embedded Systems. (M) Prerequisite(s): CIS 380, some network programming experience is desirable.

Ever increasing availability of inexpensive processors connected by a communication network has motivated the development of numerous concepts and paradigms for distributed real-time embedded systems. The primary objectives of this course are to study the principles and concepts of real-time embedded computing and to provide students hands-on experience in developing embedded applications. This course covers the concepts and theory necessary to understand and program embedded real-time systems. This includes concepts and theory for real-time system design, analysis, and certification; programming and operating systems for embedded systems; and concepts, technologies, and protocols for distributed embedded real-time systems.

The course will cover a variety of existing systems and technologies, e.g., real- machines, architectural description anguage, formal meth and logical-time programming paradigms, and certification The course requires active student participation in-group projects. Each group will be responsible for the design and implementation of a life-critical embedded system such as a pacemaker. The group projects are intended to complement the learning of principles and concepts through the application of theory in practice and the development of experimental skills in building embedded applications.

482. (CIS 582) Logic In Computer Science. (C) Prerequisite(s): CIS 160.

Logic has been called the calculus of computer science as it plays a fundamental role in computer science, similar to that played by calculus in the physical sciences and traditional engineerng disciplines. Indeed, logic is useful in areas of computer science as disparate as architecture (logic gates), software engineerng (specification and verification), programming languages (semantics, logic programming), databases (relational algebra and SQL), artificial intelligence (automatic theorem proving), algorithms (complexity and expressiveness), and theory of computation (general notions of computability). CIS 482 provides the students with a thorough introduction to mathematical logic, covering in depth the topics of syntax, semantics, decision procedures, formal proof systems, and soundness and completeness for both propositional and first-order logic. The material is taught froma computer science perspective, with an emphasis on algorithms, computational complexity, and tools. Projects will focus on problems in circuit design, specification and analysis and protocols, and query evaluation in databases.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

497. DMD Senior Project. (C) Prerequisite(s): Senior Standing or Permission of the Instructor.

The goal of this course is to provide an opportunity for seniors to define, desand execute a project of your own choosing that demonstrates the technical skiland abilities that you have acquired during your 4 years as undergraduates. Evaluation is based on selecting an interesting topic, completing appropriate research on the state of the art in that area, communicating your objectives i writing and in presentations, accurately estimating what resources will be reqto complete your chosen task, coding necessary functionality, and executing your plan.

500. Software Foundations. (C) Prerequisite(s): CIS 121, 160, and 262 (or equivalents); plus substantial mathematical maturity (at least two additional undergraduate courses in math or theoretical CS). Undergraduate-level coursework in programming languages, compilers, functional programming, or logic is helpful but not required.

This course introduces basic concepts and techniques in the foundational study of programming languages. The central theme is the view of programs and programming languages as mathematical objects for which precise claims may be made and proved. Particular topics include operational techniques for formal definition of language features, type systems and type safety properties, polymorphism, constructive logic, and the Coq proof assistant. This course is appropriate as an upper-level undergraduate CIS elective. Undergraduates who have satisfied the prerequisites are welcome to enroll. No permission from the instructor is needed.

L/R 501. Computer Architecture. (C) Prerequisite(s): Knowledge of computer organization and basic programming skills.

This course is an introductory graduate course on computer architecture with an emphasis on a quantitative approach to cost/performance design tradeoffs. The course covers the fundamentals of classical and modern uniprocessor design: performance and cost issues, instruction sets, pipelining, superscalar, out-of-order, and speculative execution mechanisms, caches, physical memory, virtual memory, and I/O. Other topics include: static scheduling, VLIW and EPIC, software speculation, long (SIMD) and short (multimedia) vector execution, multithreading, and an introduction to shared memory multiprocessors.

L/R 502. Analysis of Algorithms. (C) Prerequisite(s): CIT 594 or equivalent.

An investigation of paradigms for design and analysis of algorithms. The course will include dynamic programming, flows and combinatorial optimization algorithms, linear programming, randomization and a brief introduction to intractability and approximation algorithms. The course will include other advanced topics, time permitting.

505. Software Systems. (C) Prerequisite(s): Undergraduate-level knowledge of Operating Systems and Networking, programming experience (CIT 594 or equivalent).

This course provides an introduction to fundamental concepts of distributed systems, and the design principles for building large scale computational systems. Topics covered include communication, concurrency, programming paradigms, naming, managing shared state, caching, synchronization, reaching agreement, fault tolerance, security, middleware, and distributed applications. This course is appropriate as an upper-level undergraduate CIS elective.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

510. (CIS 410) Curves and Surfaces: Theory and Applications. (M) Prerequisite(s): Basic knowledge of linear algebra, calculus, and elementary geometry. CIS 560 is not required.

The course is about mathematical and algorithmic techniques used for geometric modeling and geometric design, using curves and surfaces. There are many applications in computer graphics as well as in robotics, vision, and computational geometry. Such techniques are used in 2D and 3D drawing and plot, object silhouettes, animating positions, product design (cars, planes, buildings), topographic data, medical imagery, active surfaces of proteins, attribute maps (color, texture, roughness), weather data, art, etc. Three broad classes of problems will be considered: approximating curved shapes, using smooth curves or surfaces. Interpolating curved shapes, using smooth curves or surfaces. Rendering smooth curves or surfaces.

511. Theory of Computation. (C) Prerequisite(s): Discrete Mathematics, Automata theory or Algorithms at the undergraduate level.

Review of regular and context-free languages and machine models. Turing machines and RAM models, Decidability, Halting problem, Reductions, Recursively enumerable sets, Universal TMs, Church/Turing thesis. Time and space complexity, hierarchy theorems, the complexity classes P, NP, PSPACE, L, NL, and co-NL. Reductions revisited, Cook-Levin Theorem, completeness, NL = co-NL. Advanced topics as time permits: Circuit complexity and parallel computation, randomized complexity, approximability, interaction and cryptography.

515. Fundamentals of Linear Algebra and Optimization. (C) Prerequisite(s): Undergraduate course in linear algebra, calculus.

This course provides firm foundations in linear algebra and basic optimization techniques. Emphasis is placed on teaching methods and tools that are widely used in various areas of computer science. Both theoretical and algorithmic aspects will be discussed.

SM 518. (MATH571, PHIL412) Topics in Logic; Finite Model Theory and Descriptive Complexity. (C)

This course will examine the expressive power of various logical languages over the class of finite structures. The course begins with an exposition of some of the fundamental theorems about the behavior of first-order logic in the context of finite structures, in particular, the Ehrenfeucht-Fraisse Theorem and the Trahktenbrot Theorem. The first of these results is used to show limitations on the expressive power of first-order logic over finite structures while the second result demonstrates that the problem of reasoning about finite structures using first-order logic is surprisingly complex. The course then proceeds to consider various extensions of first-order logic including fixed-point operators, generalized quantifiers, infinitary languages, and higher-order languages. The expressive power of these extensions will be studied in detail and will be connected to various problems in the theory of computational complexity. This last motif, namely the relation between descriptive and computational complexity, will be one of the main themes of the course.

519. (CIS 419) Introduction to Machine Learning. (C)

Machine learning has been essential to the success of many recent technologies, including autonomous vehicles, search engines, genomics, automated medical diagnosis, image recognition, and social network analysis, among many others. This course will introduce the fundamental concepts and algorithms that enable computers to learn from experience, with an emphasis on their practical application to real problems. This course will introduce supervised learning (decision trees, logistic regression, support vector machines, Bayesian methods, neural networks and deep learning), unsupervised learning (clustering, dimensionality reduction), and reinforcement learning. Additionally, the course will discuss evaluation methodology and recent applications of machine learning, including large scale learning for big data and network analysis.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

L/R 520. Machine Learning. (A) Prerequisite(s): Elementary probability, calculus, and linear algebra. Basic programming experience.

This course covers the foundations of statistical machine learning. The focus is on probabilistic and statistical methods for prediction and clustering in high dimensions. Topics covered include SVMs and logistic regression, PCA and dimensionality reduction, and EM and Hidden Markov Models.

L/R 521. Fundamentals of AI. (C) Prerequisite(s): Students are expected to have the following background: Basic algorithms, data structures and complexity (dynamic programming, queues, stacks, graphs, big-O, P/NP); Basic probability and statistics (random variables, standard distributions, simple regression); Basic linear algebra (matrices, vectors, norms, inverses); Reasonable programming skills.

Modern AI uses a collection of techniques from a number of fields in the design of intelligent systems: probability, statistics, logic, operations research, optimal control and economics, to name a few. This course covers basic modeling and algorithmic tools from these fields underlying current research and highlights their applications in computer vision, robotics, and natural language processing.

526. Machine Translation. (C)

Google translate can instantly translate between any pair of over fifty human languages (for instance, from French to English). How does it do that? Why does it make the errors that it does? And how can you build something better? Modern translation systems like Google Translate and Bing Translator learn how to translate by reading millions of words of already translated text, and this course will show you how they work. The course covers a diverse set of fundamental building blocks from linguistics, machine learning, algorithms, data structures , and formal language theory, along with their application to a real and difficult problem in artificial intelligence.

530. Computational Linguistics. (A)

Computational approaches to the problem of understanding and producing natural language text and speech, including speech processing, syntactic parsing, semantic interpretation, discourse meaning, and the role of pragmatics and world knowledge. The course will examine both rule-based and corpus-based techniques. It is recommended that students have some knowledge of logic, basic linguistics, and/or programming.

534. (CIS 434) Multicore Programming and Architecture. (C) Prerequisite(s): CIS 371 or CIS 501, and significant programming experience.

This course is a pragmatic examination of multicore programming and the hardware architecture of modern multicore processors. Unlike the sequential single-core processors of the past, utilizing a multicore processor requires programmers to identify parallelism and write explicitly parallel code. Topics covered include: the relevant architectural trends and aspects of multicores, approaches for writing multicore software by extracting data parallelism (vectors and SIMD), thread-level parallelism, and task-based parallelism, efficient synchronization, and program profiling and performance tuning. The course focuses primarily on mainstream shared-memory multicores with some coverage of graphics processing units (GPUs). Cluster-based supercomputing is not a focus of this course. Several programming assignments and a course project will provide students first-hand experience with programming, experimentally analyzing, and tuning multicore software. Students are expected to have a solid understanding of computer architecture and strong programming skills (including experience with C/C++).

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

535. (BIOL535, GCB 535) Introduction to Bioinformatics. (A)

The course covers methods used in computational biology, including the statistical models and algorithms used and the biological problems which they address. Students will learn how tools such as BLAST work, and will use them to address real problems. The course will focus on sequence analysis problems such as exon, motif and gene finding, and on comparative methods but will also cover gene expression and proteomics.

537. (BE 537) Biomedical Image Analysis. (C) Prerequisite(s): Mathematics through multivariate calculus (Math 241), programming experience, as well as some familiarity with linear algebra, basic physics, and statistics.

This course covers the fundamentals of advanced quantitative image analysis that apply to all of the major and emerging modalities in biological/biomaterials imaging and in vivo biomedical imaging. While traditional image processing techniques will be discussed to provide context, the emphasis will be on cutting edge aspects of all areas of image analysis (including registration, segmentation, and high-dimensional statistical analysis). Significant coverage of state-of-the-art biomedical research and clinical applications will be incorporated to reinforce the theoretical basis of the analysis methods.

L/R 540. Principles of Embedded Computation. (A) Prerequisite(s): This course assumes mathematical maturity, commensurate with either ESE 210 (Introduction to Dynamical Systems), or CIS 262 (Introduction to Theory of Computation). It is suitable for students who have an undergraduate degree in computer science, or computer engineering, or electrical engineering. It is also suitable for Penn undergraduates in CIS or CE as an upper-level elective.

This course is focused on principles underlying design and analysis of computational elements that interact with the physical environment. Increasingly, such embedded computers are everywhere, from smart cameras to medical devices to automobiles. While the classical theory of computation focuses on the function that a program computes, to understand embedded computation, we need to focus on the reactive nature of the interaction of a component with its environment via inputs and outputs, the continuous dynamics of the physical world, different ways of communication among components, and requirements concerning safety, timeliness, stability, and performance. Developing tools for approaching design, analysis, and implementation of embedded systems in a principled manner is an active research area. This course will attempt to give students a coherent introduction to this emerging area. This course is appropriate as an upper-level undergraduate CIS elective.

541. (CIS 441) Embedded Software for Life-Critical Applications. (C) Prerequisite(s): CIS 240 or equivalent, ESE 350 is recommended.

This course is focused on cyber physical systems with emphasis on real-time issues. Cyber physical systems are integrations of computation and communication with physical processes. Embedded computers monitor and control physical processes in real-time. As these embedded computer transformed from word processors to global communications devices for information gathering and sharing, embedded computers will change from small self-contained systems to cyber-physical systems by sensing,

The course is to study principles, methods, and techniques for building high-assurance cyber-physical systems. Topics will include requirements capture and modeling, mental models, assurance cases, hazard analysis, real-time programming and communication, real-time scheduling and virtual machines, feedback control in computer systems, verification and validseries of projects that will implement safety-critical embedded systems (e.g., pacemaker, infusion pump).

542. Embedded Systems Programming. (C) Prerequisite(s): C fluency.

This course explores techniques for writing correct and efficient embedded code. Topics include C/C++ idioms, data abstraction, elementary data structures and algorithms, environment modeling, concurrency, hard real time, and modular program reasoning.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

550. (CIS 450) Database and Information Systems. (A) Prerequisite(s): CIT 594 and CIT 592 or equivalent.

Introduction to the theory and practice of data management systems, including databases and data integration. The Entity-Relationship approach as a modeling tool. The relational model, algebra and calculus. Database design and relational normalization. Views and their role in security and integration. Physical data organization and indexing structures. Query execution and optimization. Updates and integrity: transaction management, concurrency control and recovery techniques. XML and database-backed Web sites.

551. (TCOM551) Computer and Network Security. (B) Prerequisite(s): TCOM 512 or equivalent required; CIS 500 recommended.

This is an introduction to topics in the security of computer systems and communication on networks of computers. The course covers four major areas: fundamentals of cryptography, security for communication protocols, security for operating systems and mobile programs, and security for electronic commerce. Sample specific topics include: passwords and offline attacks, DES, RSA, DSA, SHA, SSL, CBC, IPSec, SET, DDoS attacks, biometric authentication, PKI, smart cards, S/MIME, privacy on the Web, viruses, security models, wireless security, and sandboxing. Students will be expected to display knowledge of both theory and practice through written examinations and programming assignments.

552. Advanced Programming. (C) Prerequisite(s): Four courses involving significant programming and a discrete mathematics or modern algebra course. Enrollment by permission of the instructor only.

The goals of this course are twofold: (1) to take good programmers and turn them into excellent ones, and (2) to introduce them to a range of modern software engineering practices, in particular those embodied in advanced functional programming languages.

553. Networked Systems. (C) Prerequisite(s): CIS 121 or equivalent, or permission of the instructor.

This course provides an introduction to fundamental concepts in the design and implementation of networked systems, their protocols, and applications. Topics to be covered include: Internet architecture, network applications, addressing, routing, transport protocols, network security, and peer-to-peer networks. The course will involve written assignments, examinations, and programming assignments.. Students will work in teams to design and implement networked systems in layers, from routing protocols, transport protocols, to peer-to-peer networks.

554. Programming Paradigms. (C) Prerequisite(s): CIS 121 or CIT 594 or equivalent.

Achieving mastery in a new programming language requires more than just learning a new syntax; rather, different languages support different ways to think about solving problems. Not all programming languages are inherently procedural or object-oriented. The intent of this course is to provide a basic understanding of a wide variety of programming paradigms, such as logic programming, functional programming, concurrent programming, rule-based programming, and others.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

555. (CIS 455) Internet and Web Systems. (C) Prerequisite(s): Familiarity with threads and concurrency, strong Java programming skills.

This course focuses on the challenges encountered in building Internet and web systems: scalability, interoperability (of data and code), security and fault tolerance, consistency models, and location of resources, services, and data. We will examine how XML standards enable information exchange; how web services support cross-platform interoperability (and what their limitations are); how to build high-performance application servers; how "cloud computing" services work; how to perform Akamai-like content distribution; and how to provide transaction support in distributed environments. We will study techniques for locating machines, resources, and data (including directory systems, information retrieval indexing, ranking, and web search); and we will investigate how different architectures support scalability (and the issues they face). We will also examine ideas that have been proposed for tomorrow's Web, and we will see some of the challenges, research directions, and potential pitfalls. An important goal of the course is not simply to discuss issues and solutions, but to provide hands-on experience with a substantial implementation project.
This semester's project will be a peer-to-peer implementation of a Googe-style search engine, including distributed, scalable crawling; indexing with ranking; and even PageRank. As a side-effect of the material of this course you will learn about some aspects of large-scale software development assimilating large APIs, thinking about modularity, reading other people's code, managing versions, debugging, etc.

558. (LING525) Computer Analysis and Modeling of Biological Signals and Systems. (B) Prerequisite(s): Undergraduate-level knowledge of linear algebra.

A graduate course intended to introduce the use of signal and image processing tools for analyzing and modeling biological systems. We present a series of fundamental examples drawn from areas of speech analysis/synthesis, computer vision, and modeling of biological perceptual systems. Students learn the material through lectures and via a set of computer exercises developed in MATLAB.

559. Programming and Problem Solving. (C) Prerequisite(s): Proficiency in Java. CIS 320 or CIS 502, or equivalent.

This course develops students problem solving skills using techniques that they have learned during their CS training. Over the course of the semester, students work on group projects in which they use programming techniques to solve open-ended problems, e.g. optimization, simulation, etc. There are no "correct" answers to these problems; rather, the focus is on the four steps of the problem solving process: algorithmic thinking; programming; analysis; and communication.

560. (CIS 460) Computer Graphics. (A) Prerequisite(s): One year programming experience (C, JAVA, C++).

A thorough introduction to computer graphics techniques, covering primarily 3D modeling and image synthesis. Topics cover: geometric transformations, geometric algorithms, software systems (OpenGL), 3D object models (surface and volume), visible surface algorithms, image synthesis, shading and mapping, ray tracing, radiosity, global illumination, photon mapping, anti-aliasing and compositing.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

562. (CIS 462) Computer Animation. (C) Prerequisite(s): Previous exposure to major concepts in linear algebra (i.e. vector matrix math), curves and surfaces, dynamical systems (e.g. 2nd order mass-spring-damper systems) and 3D computer graphics has also been assumed in the preparation of the course materials.

This course covers core subject matter common to the fields of robotics, character animation and embodied intelligent agents. The intent of the course is to provide the student with a solid technical foundation for developing, animating and controlling articulated systems used in interactive computer games, virtual reality simulations and high-end animation applications. The course balances theory with practice by "looking under the hood" of current animation systems and authoring tools and exams the technologies and techniques used from both a computer science and engineering perspective. Topics covered include: geometric coordinate systems and transformations; quaternions; parametric curves and surfaces; forward and inverse kinematics; dynamic systems and control; computer simulation; keyframe, motion capture and procedural animation; behavior-based animation and control; facial animation; smart characters and intelligent agents.

563. Physically Based Animation. (C) Prerequisite(s): Students should have a good knowledge of object-oriented programming (C++) and basic familiarity with linear algebra and physics. Some background in computer graphics is helpful.

This course introduces students to common physically based simulation techniques for animation of fluids and gases, rigid and deformable solids, cloth, explosions, fire, smoke, virtual characters, and other systems. Physically based simulation techniques allow for creation of extremely realistic special effects for movies, video games and surgical simulation systems. We will learn state-of-the-art techniques that are commonly used in current special effects and animation studios and in video games community. To gain hands-on experience, students will implement basic simulators for several systems. The topics will include: Particle Systems, Mass spring systems, Deformable Solids & Fracture, Cloth, Explosions & Fire, Smoke, Fluids, Deformable active characters, Simulation and control of rigid bodies, Rigid body dynamics, Collision detection and handling, Simulation of articulated characters, Simulated characters in games. The course is appropriate for both upper level undergraduate and graduate students.

564. Game Design and Development. (C) Basic understanding of 3D graphics and animation principles, prior exposure to scripting and programming languages such as Python, C and C+ +..Prerequisite(s): Basic understanding of 3D graphics and animation principles, prior exposure to scripting and programming languages such as Python, C and C++.

The intent of the course is to provide students with a solid theoretical understanding of the core creative principles, concepts, and game play structures/schemas underlying most game designs. The course also will examine game development from an engineering point of view, including: game play mechanics, game engine software and hardware architectures, user interfaces, design documents, playtesting and production methods.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

565. GPU Programming and Architectur. (C) Prerequisite(s): CIS 460 or CIS 560, and familiarity with computer hardware/systems. The hardware/systems requirement may be met by CIS 501; or CIT 593 and 595; or CIS 240 (with CIS 371 recommended); or equivalent coursework.

This course examines the architecture and capabilities of modern GPUs. The graphics processing unit (GPU) has grown in power over recent years, to the point where many computations can be performed faster on the GPU than on a traditional CPU. GPUs have also become programmable, allowing them to be used for a diverse set of applications far removed from traditional graphics settings. Topics covered include architectural aspects of modern GPUs, with a special focus on their streaming parallel nature, writing programs on the GPU using high level languages like Cg and BrookGPU, and using the GPU for graphics and general purpose applications in the area of geometry modeling, physical simulation, scientific computing and games. Students are expected to have a basic understanding of computer architecture and graphics, and should be proficient in OpenGL and C/C++. This course is appropriate as an upper-level undergraduate CIS elective.

568. Game Design Practicum. (C) Prerequisite(s): CIS 462/562, CIS 277 or CIS 460/560. Corequisite (s): CIS 564.

The objective of the game design practicum is to provide students with hands on experience designing and developing 3D computer games. Working in teams of three or four, students will brainstorm an original game concept, write a formal game design document then develop a fully functional prototype consisting of a playable level of the game. In addition to creation of original art and animation assets for the game, technical features to be designed and implemented include a nove l game mechanic and/or user interaction model, game physics (i.e. particle systems and rigid body dynamics), character animation, game AI (i.e. movement control, path planning, decision making, etc.), sound effects and effects and background music, 2D graphical user interface (GUI) design and optional multiplayer networking capabilities. Consistent with standard industry practices, game code and logic will be written using C++ and popular scripting languages such as Python and Lua.
State-of-the-art game and physics engine middleware also will be used to expose students to commercial-grade software, production methodologies and art asset pipelines. As a result of their game development efforts, students will learn first hand about the creative process, design documentation, object-oriented software design and engineering, project management (including effective team collaboration and communication techniques), design iteration through user feedback and play-testing, and most importantly, what makes a game fun to play.

570. Modern Programming Language Implementation. (M) Prerequisite(s): CIS 500. An undergraduate course in compiler construction (CSE 341 or equivalent) is helpful but not required.

This course is a broad introduction to advanced issues in compilers and run-time systems for several classes of programming languages, including imperative, object-oriented, and functional. Particular attention is paid to the structures, analyses, and transformations used in program optimization.

571. (PHIL411) Recursion Theory. (A)

The course covers the basic theory of recursive and recursively enumerable sets and the connection between this theory and a variety of decision problems of interest in a computational setting. The course will then proceed to an exposition of recursion theoretic reducibilities. Elementary results about degrees of unsolvability are established. The theory of arithmetical, analytical, and projective hierarchies will be presented. The study of functionals at this point will provide an entry into the computationally important subject of recursion at higher types. Basic parts of the theory of inductive definitions and monotone operators will be presented. If time and interest permit, this theory will be applied to the analysis of the semantical paradoxes. The course will conclude with an investigation of the lower levels of the analytical and projective hierarchies. Applications to the degrees of unsolvability of various logical systems will be presented, connections between the hierarchies and predicative formal systems will be established, and the relation between the theory of the projective hierarchy and topics in classical descriptive set theory will be indicated.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

SM 572. (PHIL413) Set Theory. (C)

This course is an introduction to set theory. It will begin with a study of Zermelo-Fraenkel set theory (ZF) as a partial description of the cumulative hierarchy of sets. Elementary properties of cardinal and ordinal numbers will be developed in ZF. The inner model of constructible sets will be used to establish the relative consistency of the axiom of choice and the generalized continuum hypothesis with ZF. The method of forcing will be introduced to establish the independence of the continuum hypothesis from ZF and other independence results. Large cardinals and their bearing on the resolution of questions about the continuum will be considered.

573. Software Engineering. (A) Prerequisite(s): CIT 591 and 593, or CIS 120, 121, and 240, or equivalent; proficiency in Java.

Writing a "program" is easy. Developing a "software product", however, introduces numerous challenges that make it a much more difficult task. This course will look at how professional software engineers address those challenges, by investigating best practices from industry and emerging trends in software engineering research. Topics will focus on software maintenance issues, including: test case generation and test suite adequacy; code analysis verification and model checking; debugging and fault localization; refactoring and regression testing; and software design and quality.

580. Machine Perception. (A) Prerequisite(s): A solid grasp of the fundamentals of linear algebra. Some knowledge of programming in C and/or Matlab.

An introduction to the problems of computer vision and other forms of machine perception that can be solved using geometrical approaches rather than statistical methods. Emphasis will be placed on both analytical and computational techniques. This course is designed to provide students with an exposure to the fundamental mathematical and algorithmic techniques that are used to tackle challenging image based modeling problems. The subject matter of this course finds application in the fields of Computer Vision, Computer Graphics and Robotics. Some of the topics to be covered include: Projective Geometry, Camera Calibration, Image Formation, Projective, Affine and Euclidean Transformations, Computational Stereopsis, and the recovery of 3D structure from multiple 2D images. This course will also explore various approaches to object recognition that make use of geometric techniques, these would include alignment based methods and techniques that exploit geometric invariants. In the assignments for this course, students will be able to apply the techniques to actual computer vision problems. This course is appropriate as an upper-level undergraduate CIS elective.

581. Computer Vision & Computational Photography. (M)

This is an introductory course to Computer Vision and Computational Photography. This course will explore three topics: 1) image morphing, 2) image matching and stitching, and 3) image recognition. This course is intended to provide a hands-on experience with interesting things to do on images/ videos. The world is becoming image-centric. Cameras are now found everywhere, in our cell phones, automobiles, even in medical surgery tools. Computer vision technology has led to latest innovations in areas such as Hollywood movie production, medical diagnosis, biometrics, and digital library. This course is suited for students from all Engineering backgrounds, who have the basic knowledge of linear algebra and programming, and a lot of imagination.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

582. (CIS 482) Logic in Computer Science. (C) Prerequisite(s): CIS 160 or CIT 592 or equivalent.

Logic has been called the calculus of computer science as it plays a fundamental role in computer science, similar to that played by calculus in the physical sciences and traditional engineering disciplines. Indeed, logic is useful in areas of computer science as disparate as architecture (logic gates), software engineering (specification and verification), programming languages (semantics, logic programming), databases (relational algebra and SQL), artificial intelligence (automatic theorem proving), algorithms (complexity and expressiveness), and theory of computation (general notions of computability). CIS 582 provides students with a thorough introduction to mathematical logic, covering in depth the topics of syntax, semantics, decision procedures, formal proof systems, and soundness and completeness for both propositional and first-order logic. The material is taught from a computer science perspective, with an emphasis on algorithms , computational complexity, and tools. Projects will focus on problems in circuit design, specification and analysis of protocols, and query evaluation in databases.

597. Master's Thesis Research. (C)

For students working on an advanced research leading to the completion of a Master's thesis.

599. Independent Study for Masters Students. (C)

For master's students studying a specific advanced subject area in computer and information science. Involves coursework and class presentations. A CIS 599 course unit will invariably include formally gradable work comparable to that in a CIS 500-level course. Students should discuss with the faculty supervisor the scope of the Independent Study, expectations, work involved, etc.

601. Advanced Topics in Computer Architecture. (C) Prerequisite(s): CIS 501 or strong performance in CIS 371.

This course will focus on research topics in computer architecture, and include reading and presenting research papers and an optional project. The content will differ with each offering, covering topics such as multicore programmability, datacenter and warehouse-scale computing, security, energy-efficient architectures, etc.

610. (MATH676) Advanced Geometric Methods in Computer Science. (B) Prerequisite(s): CIS 510 or coverage of equivalent material.

The purpose of this course is to present some of the advanced geometric methods used in geometric modeling, computer graphics, computer vision, etc. The topics may vary from year to year, and will be selected among the following subjects (nonexhaustive list): Introduction to projective geometry with applications to rational curves and surfaces, control points for rational curves, rectangular and triangular rational patches, drawing closed rational curves and surfaces; Differential geometry of curves (curvature, torsion, osculating planes, the Frenet frame, osculating circles, osculating spheres); Differential geometry of surfaces (first fundamental form, normal curvature, second fundamental form, geodesic curvature, Christoffel symbols, principal curvatures, Gaussian curvature, mean curvature, the Gauss map and its derivative dN, the Dupin indicatrix, the Theorema Egregium equations of Codazzi-Mainadi, Bonnet's theorem, lines of curvatures, geodesic torsion, asymptotic lines, geodesic lines, local Gauss-Bonnet theorem).

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

613. (ESE 617, MEAM613) Nonlinear Control Theory. (M) Prerequisite(s): A sufficient background to linear algebra (ENM 510/511 or equivalent) and a course in linear control theory (MEAM 513 or equivalent), or written permission of the instructor.

The course studies issues in nonlinear control theory, with a particular emphasis on the use of geometric principles. Topics include: controllability, accessibility, and observability, for nonlinear systems; Forbenius' theorem; feedback and input/outpub linearization for SISO and MIMO systems; dynamic extension; zero dynamics; output tracking and regulation; model matching disturbance decoupling; examples will be taken from mechanical systems, robotic systems, including those involving nonholonomic constraints, and active control of vibrations.

SM 620. Advanced Topics in Artificial Intelligence. (B) Prerequisite(s): CIS 520 or equivalent.

Discussion of problems and techniques in Artificial Intelligence (AI): Knowledge Representation, Natural Language Processing, Constraint Systems, Machine Learning; Applications of AI.

SM 625. Computational Learning Theory. (C) Prerequisite(s): Prior courses in algorithms, complexity and statistics would be helpful but are not necessary.

This course is an introduction to Computational Learning Theory, a field which attempts to provide algorithmic, complexity-theoretic and statistical foundations to modern machine learning. The focus is on opics in computational learning theory for researchers and students in artificial intelligence, neural networks, theoretical computer science, and statistics.

SM 630. Advanced Topics in Natural Language Processing. (C) Prerequisite(s): CIS 530 or equivalent or permission of instructor.

Different topics selected each offering; e.g., NL generation, question-answering, information extraction, machine translation, restricted grammar formalisms, computational lexical semantics, etc.

SM 635. (BIOL537, GCB 537) Advanced Computational Biology. (A) Prerequisite(s): Biol 536 or permission of the instructor.

Discussion of special research topics.

SM 639. Statistical approaches to Natural Language Understanding. (C)

This course examines the recent development of corpus-based techniques in natural language processing, focusing on both statistical and primarily symbolic learning techniques. Particular topics vary from year to year.

SM 640. Advanced Topics in Software Systems. (B) Prerequisite(s): CIS 505 or equivalent.

Different topics selected for each course offering.

SM 650. Advanced Topics in Databases. (B) Prerequisite(s): CIS 550.

Advanced topics in databases: distributed databases, integrity constraints, failure, concurrency control, relevant relational theory, semantics of data models, the interface between programming of languages and databases. Object-oriented databases. New topics are discussed each year.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

SM 660. Advanced Topics in Computer Graphics and Animation. (B) Prerequisite(s): CIS 560 or permission of the instructor.

The goal of the course is to review state-of-the art research in the fields of computer graphics and animation as well as provide students with working knowledge of how to convert theory to practice by developing an associated graphics/animation authoring tool. The course is comprised of primers, lectures, student presentations and the authoring tool group project. Each student will be responsible for presenting one primer and at least two SIGGRAPH papers to the class. Working in teams of two, students will design and develop an authoring tool that that facilitates the creation of a new type of user interaction, animation/simulation capability or 3D graphics special effect. Research papers published in the SigGraph Conference proceedings will provide the basis for the features/functionality/special effects that can be selected for implementation in the authoring tool. Each group will analyze the need and user requirements for the tool they plan to develop, prepare a formal software design document, construct a project work plan, develop the authoring tool functionality and user interface, test the design and demonstratthe authoring of associated content. A plug-in to standard authoring tools such as Maya or Houdini must also be developed to enable importing of appropriate assets and/or exporting of results.

SM 670. Advanced Topics in Programming Languages. (C) Prerequisite(s): CIS 500.

The details of this course change from year to year, but its purpose is to cover theoretical topics related to programming languages. Some central topics include: denotational vs operational semantics, domain theory and category theory, the lambda calculus, type theory (including recursive types, generics, type inference and modules), logics of programs and associated completeness and decidability problems, specification languages, and models of concurrency. The course requires a degree of mathematical sophistication.

673. Computer-Aided Verification. (C) Prerequisite(s): Basic knowledge of algorithms, data structures, automata theory, propositional logic, operating systems, communication protocols, and hardware (CIS 262, CIS 380, or permission of the instructor).

This course introduces the theory and practice of formal methods for the design and analysis of concurrent and embedded systems. The emphasis is on the underlying logical and automata-theoretic concepts, the algorithmic solutions, and heuristics to cope with the high computational complexity. Topics: Models and semantics of reactive systems; Verification algorithms; Verification techniques. Topics may vary depending on instructor.

677. Advanced Topics in Algorithms and Complexity. (A) Prerequisite(s): Consent of the instructor.

This course covers various aspects of discrete algorithms. Graph-theoretic algorithms in computational biology, and randomization and computation; literature in dynamic graph algorithms, approximation algorithms, and other areas according to student interests.

SM 680. Advanced Topics in Machine Perception. (B) Prerequisite(s): A previous course in machine perception or knowledge of image processing, experience with an operating system and language such as Unix and C, and aptitude for mathematics. A previous course in machine perception or knowledge of image processing, experience with an operating system and language such as Unix and C, and aptitude for mathematics.

Graduate seminar in advanced work on machine perception as it applies to robots as well as to the modeling of human perception. Topics vary with each offering.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

682. Friendly Logics. (C)

The use of logical formalisms in Computer Science is dominated by a fundamental conflict: expressiveness vs. algorithmic tractability. Database constraint logics, temporal logics and description logics are successful compromises in this conflict: (1) they are expressive enough for practical specifications in certain areas, and (2) there exist interesting algorithms for the automated use of these specifications. Interesting connections can be made between these logics because temporal and description logics are modal logics, which in turn can be seen, as can database constraint logics, as certain fragments of first-order logic. These connections might benefit research in databases, computer-aided verification and AI. Discussion includes other interesting connections, eg., with SLD-resolution, with constraint satisfaction problems, with finite model theory and with automata theory.

700. Special Topics. (M)

One time course offerings of special interest. Equivalent to a CIS 5XX level course.

800. PhD Special Topics. (C)

One-time course offerings of special interest. Equivalent to CIS seminar course. Offerings to be determined.

899. Doctoral Independent Study. (C)

For doctoral students studying a specific advanced subject area in computer and information science. The Independen t Study may involve coursework, presentations, and formally gradable work comparable to that in a CIS 500 or 600 level course. The Independent Study may also be used by doctoral students to explore research options with faculty, prior to determining a thesis topic. Students should discuss with the faculty supervisor the scope of the Independent Study, expectations, work involved, etc. The Independent Study should not be used for ongoing research towards a thesis, for which the CIS 999 designation should be used.

990. Masters Thesis.

For master's students who have taken ten course units and need only to complete the writing of a thesis or finish work for incompletes in order to graduate. CIS 990 carries full time status with zero course units and may be taken only once.

995. Dissertation.

For Ph.D. candidates working exclusively on their dissertation research, having completed enrollment for a total of ten semesters (fall and spring). There is no credit or grade for CIS 995.

996. Research Seminar. (C)

Introduction to research being conducted in the department. Mandatory for firstyear doctoral students. Taken as fifth course for no credit at no cost.

999. Thesis/Dissertation Research. (C)

For students pursuing advanced research to fulfill PhD dissertation requirements.

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

312. Theory of Networks. (C) Prerequisite(s): ESE 301 and MATH 312.

Networks are ubiquitous in our modern society, playing an increasingly larger role in everyday life. These include: Social Networks such as Facebook or Twitter, infrastructure networks such as the Internet, or energy networks such as the electric grid. Network Science and Engineering is a new discipline that investigates the structure of large complex networks and their behavior and properties, and then designing technologies that control and manipulate their behaviors to bring about greater benefits to society. In this course students will learn some of the basic tools, methods, and algorithms for analysis of networked systems, as well as practical applications of this new science.

112. NETWORKED LIFE. (A)

150. MKT/SOC SYS ON INTERNET.

212. SCALABLE & CLOUD COMP. (A)

L/R 312. THEORY OF NETWORKS. (C)

412. ALGORITHMIC GAME THEORY. (C)

597. MASTER'S THESIS RESEARCH. (C)

599. MASTERS INDEPEND STUDY. (C)

MARKET and SOCIAL SYSTEMS (MKSE)

150. Mkt/Soc Sys on Internet. (C)