

COMPUTER AND INFORMATION SCIENCE (EG) {CIS}

COMPUTER SCIENCE & ENGINEERING (CSE)

Undergraduate Courses

099. Undergraduate Research/Independent Study. (C) A maximum of 2 c.u. of CSE 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.

L/R 110. Introduction to Computer Programming (with Java, for beginners). (C)

How do you program computers to accomplish tasks? How do you break down a complex task into simpler ones? CSE 110 is a "Java lite" course that covers the fundamentals of object-oriented programming such as objects, classes, state, methods, loops, arrays, inheritance, and recursion using the Java programming language.

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.

L/R 120. Programming Languages and Techniques I. (C)

This will be a fast-paced introduction to the fundamental concepts of programming, with Java as the main experimental vehicle. We assume some previous programming experience at the level of a high school computer science class. If you got at least 4 in the AP Computer Science A or AB exam, you will do great. However, we do not assume you know Java. Basic experience with any programming language (for instance C, C++, VB, PHP, Perl, or Scheme) will be sufficient. A quiz will be given in the second week of class to test your programming knowledge so that you can decide whether the class is for you. If you have never programmed before, you should take CIS 110 first. We will mainly use Java and the DrJava programming environment, but we will also experiment with Python, a higher-level language.

L/R 121. Programming Languages and Techniques II. (B) Prerequisite(s): CIS 120, CIS 260 is a pre or co-requisite for but will be strictly a pre-requisite effective Fall 2009.

This is a course about Algorithms and Data Structures using the JAVA programming language. We introduce the basic concepts about complexity of an algorithm and methods on how to compute the running time of algorithms. Then, we describe data structures like stacks, queues, maps, trees, and graphs, and we construct efficient algorithms based on these representations. The course builds upon existing implementations of basic data structures in JAVA and extends them for the structures like trees, studying the performance of operations on such structures, and their efficiency when used in real-world applications. A large project introducing students to the challenges of software engineering 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.

140. (COGS001, LING105, PHIL044, PPE 140, PSYC107) Introduction to Cognitive Science. (A)

Prerequisite(s): An introductory course in Computer Science, Linguistics, Neuroscience, Philosophy or Psychology.

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

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 primitive 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 260. Mathematical Foundations of Computer Science. (B)

What are the basic mathematical concepts and techniques needed in computer science? This course provides an introduction 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.

L/R 261. Discrete Probability, Stochastic Processes, and Statistical Inference. (B) Prerequisite(s): CSE 260 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): CSE 260.

The course provides an introduction to the theory of computation. The treatment is mathematical, but the point of view is that of Computer Science. Broadly speaking, the theory of computation consists of three overlapping subareas: (1) formal languages and automata; (2) computability and recursive function theory; (3) complexity theory. The course will focus mostly on (1) and (2). The topics covered include finite automata and regular languages, context-free languages, Turing machines, Church's Thesis, undecidability, reducibility and completeness, time complexity and NP-completeness.

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): CSE 120, 121, 260, 262.

How do you optimally encode a text file? How do you find shortest paths in a map? How do you design 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 flows; NP-completeness theory; and approximation algorithms.

330. Design Principles of Information Systems. (A) Prerequisite(s): CSE 121 and 260.

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. Datalog and recursive queries. Views and data integration. Overview of system level issues: physical data organization, 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): CSE 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 design. 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.

340. Problem Solving and Programming. (M) Prerequisite(s): CSE 120, 121.

This course is about the principles of programming languages. It studies programming language concepts by implementing a sequence of interpreters, compilers, and type checkers, each one introducing a new language concept. The goal of this course is threefold: By studying the concepts and abstractions of high-level programming languages, students should be able to use them more effectively. Second, by learning how the features of high-level programming languages are implemented, students should be able to program more expressively in low-level languages. Finally, by understanding the principles behind programming language design, students should be able to create, evaluate and compare programming languages.

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

You know how to program, but do you know how to write programs that understand and generate other programs? This is the focus of CSE 341. In addition to traditional programming language implementation topics (such as lexing, parsing, grammars, symbol tables, code generation, optimization, garbage collection, and object-oriented implementation), this course also explores the more general problem of reasoning about computation (e.g., for detecting bugs or security constraint violations). CSE 341 includes a substantial and rewarding Java programming project to develop a fully operational compiler for a Java-like object oriented programming language.

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

Large systems versus small programs. Problems of scale. Software life-cycle: design phase, implementation phase, testing, maintenance. Software re-use. Tools/Toolkits/Libraries. Programming as a group activity. Support tools, e.g., SCCS and RCS. Standards. Software readability and structure. Reading code. Style sheets. Software Testing: role in process, test cases, testers. Documentation. Embedded documentation and external documentation.

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

This is the second computer organization 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 arithmetic, (5) datapath and control, (6) micro-programming, (7) pipelining, (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.

372. Computer Organization and Design Lab. (B) Corequisite(s): CSE 371.

Laboratory for CSE 371. In this laboratory section, students gain experience with digital design techniques by designing and implementing actual circuits using Verilog HDL and FPGAs. Five assignments culminate in the design and simulation of a complete 16-bit integer pipelined CPU.

380. Computer Operating Systems. (A) Prerequisite(s): CSE 240 or EE 300.

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 evaluation.

381. Computer Operating System Lab. (A) Corequisite(s): CSE 380.

This course is a semester long project to design and implement your own operating system. Typical components include a process management system, a command interpreter, and a file management system.

390. (MEAM420, MEAM520) 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): CSE 121 and CSE 262; CSE 341 strongly recommended.

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

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

The purpose of this course is to introduce undergraduate students in computer science and engineering to quantum computers (QC) and quantum information science (QIS). This course is meant primarily for juniors and seniors in CSE. 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 course 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 CSE 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 submitted report) presentation skills and progress on the project. These are evaluated by the Project Adviser and the Course Instructor.

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

Continuation of CSE 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.

410. (CIS 510) Curves and Surfaces: Theory and Applications. (M)

The course introduces mathematical and algorithmic techniques for geometric modeling with applications to computer graphics and computer animation. The course covers implicit and parametric curves; implicit and parametric surfaces; polygonal surfaces; polygonal surface simplification, decomposition, and parametrization; and surface reconstruction from point sets.

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

Automatic summarization 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.

434. (CIS 534) Introduction to Parallel Processing. (C)

This course is a pragmatic introduction to parallel and distributed programming. It targets science and engineering students with basic programming skills, and prepares them for parallelizing existing sequential programs or optimizing the performance of existing parallel codes. The course teaches how to program with widely used parallel programming interfaces such as Pthread, MPI, OpenMP, HPF and RMI. In addition, the course covers enough information on common parallel architectures, so that the students can optimize the programs for different platforms.

455. (CIS 555) Internet and Web Systems. (C) Prerequisite(s): CIS 330, CIS 380 recommended.

This course focuses on Internet and Web technologies and the underlying principles of distributed systems, information retrieval, and data management. The material covered will include web and application server architectures, SML and semistructured data, schema mediation, document indexing and retrieval, peer-to-peer systems, distributed transactions and remote procedure calls. The course has a substantial group implementation project.

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.

461. (CIS 561) Computer Modeling and Animation Applications. (C) Prerequisite(s): CSE 120, 121 or equivalent experience and concurrent or past enrollment in CSE 460/560.

This project-based course is designed to provide a comprehensive introduction to the application of computer graphics in a laboratory setting. Course materials and labs will facilitate understanding issues and trends in 3D computer graphics. Students will develop a facility with fundamental 3-D models and modeling software through a series of projects. The course will offer students a technical understanding of Polygonal and Spline based modeling, alternative and standard methods of 3-D model import and export, and model conversion. It will also cover procedural and scripting methods, techniques, and conventions for creating models and shaders that will function properly for rendering and animation. Practical application of topics covered in CSE 460/560 include; geometric transformations, hierarchies, articulation, modeling, blend shapes, vertex weighting, and animation. Experiments with various animation methods include: dynamics, forward and inverse kinematics, surface deformations, keyframe interpolation, motion capture, procedural animation, and facial animation. The course will be laboratory based and will use industry standard software.

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 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 examines 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.

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, Grammars as deductive systems, parsing as deduction. Relevance of formal grammars 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-time machines, architectural description language, formal method 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): CSE 260.

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). CSE 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 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 and protocols, and query evaluation in databases.

COMPUTER & INFORMATION SCIENCE (CIS)

Graduate Courses

500. Software Foundations. (C) Prerequisite(s): Undergraduate-level course in programming languages or compilers; significant programming experience.

This course introduces basic concepts and techniques in the foundational study of programming languages. The central theme is the view of individual programs and whole languages as mathematical objects about 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 and subtyping, foundations of object-oriented programming, and mechanisms supporting information hiding and programming in the large.

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.

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

An investigation of several major algorithms and their uses in areas including list manipulation, sorting, searching, selection and graph manipulation. Efficiency and complexity of algorithms. Complexity Classes.

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. Topics covered include communication, concurrency, programming paradigms, naming, managing shared state, caching, synchronization, reaching agreement, fault tolerance, security, middleware, and distributed applications.

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): Basic notions of discrete algebra.

Finite automata (deterministic and nondeterministic) regular graphs, regular expressions, regular grammars, (Nerode congruence), the "pumping lemma", closure properties. Context-free languages. Standard forms: removal of ϵ -rules, chain rules, reduced grammars. Chomsky Normal Form. Context-free languages as fixed points (Ginsburg and Rose's Theorem). Greibach Normal Form (using Rosenkrantz's matrix method). Ogden's Lemma and the "pumping lemma". Pushdown automata (PDA's). Equivalence of PDA's and context-free grammars. Brief sketch of top-down and bottom-up (nondeterministic) parsing. Deterministic PDA's. Closure properties. Partial recursive functions, Turing machines and RAM programs. Primitive recursion. Minimization. Equivalence of the models. Church/Turing's thesis. Acceptable Codings. A Universal RAM program. Undecidability of the halting problem. Recursively enumerable sets (RE sets).

SM 518. (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 Trakhtenbrot 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.

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. Other topics covered include graphical models, dimensionality reduction, neural networks, and reinforcement learning.

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.

530. Computational Linguistics. (A)

Computational approaches to the problem of understanding and producing written and spoken natural language, including speech processing, syntactic parsing, statistical and corpus-based techniques, semantic interpretation, discourse meaning, and the role of pragmatics and world knowledge. It is recommended that students have some knowledge of logic, basic linguistics, and programming.

534. (CIS 434) Introduction to Parallel Processing. (C)

This course is a pragmatic introduction to parallel programming. It is intended for graduate students in computer science, as well as all science and engineering students with an interest in parallel programming. This course prepares the students for parallelizing sequential programs or optimizing the performance of existing parallel codes. After a brief discussion of the basic notions of parallelism, we will discuss several popular models of parallel programming, including Pthreads, MPI, OpenMP, HPF, Linda, and object-oriented parallel programming. We will also study techniques to improve the efficiency of parallel applications on various platforms. The students are required to carry out a major programming project that often originates from their discipline. Assessment will be determined by some combination of projects and exams.

L/L 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.

536. (BIOL536, GCB 536) Computational Biology. (A) Prerequisite(s): Math 104/114 or equivalent, BIOL 221 or equivalent, or permission of the instructor.

Computational problems in molecular biology, including sequence search and analysis, informatics, phylogenetic reconstruction, genetic mapping and optimization.

537. (BE 537) Biomedical Image Analysis. (C) Faculty. Prerequisite(s): Math 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.

550. Database and Information Systems. (A) Prerequisite(s): CIT 591 or equivalent.

Introduction to the theory and practice of database management systems. The Entity-Relationship approach as a modeling tool. The relational model, algebra and calculus. Commercial systems: SQL, Quel and Ingres. Database design and relational normalization. Physical data organization and indexing structures. Updates and integrity: transaction management, concurrency control and recovery techniques. Logic as a data model: Datalog and evaluation techniques. The network model and object oriented approaches.

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.

553. (TCOM512) Networked Systems. (C) Prerequisite(s): CIS 121 (Programming Languages and Techniques II) 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, quality of service, transport protocols, data link protocols, network security, and application level protocols such as peer-to-peer networks and overlay networks. The course will involve written assignments, examinations, and programming assignments.

555. (CIS 455) Internet and Web Systems. (C) Prerequisite(s): At least one year of Java programming. Database and operating systems familiarity recommended.

This course focuses on the issues encountered in building Internet and web systems: scalability, interoperability (of data and code), atomicity and consistency models, replication, and location of resources, services, and data. Topics include other remote procedure calls, caching, replication, and hierarchical structures; distributed consensus and transactions. Of particular focus will be techniques for locating machines, resources, and data (including ranked web search, publish/subscribe systems, directories, and peer-to-peer protocols). This course has a significant project-based component, in order to provide hands-on experience with the ideas and algorithms discussed. Students will construct and validate a large-scale distributed system, with some components developed individually and some in teams.

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.

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

A thorough introduction to computer graphics techniques, including 3D modeling, rendering, and animation. 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, animation techniques, and virtual environments.

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 examines 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): Prerequisite: CIS 460/560; CIS 462/562 or instructor's permission. Students should have a good knowledge of C++, OpenGL and basic familiarity with linear algebra and physics.

This course introduces students to common physically based modeling techniques for animation of virtual characters, fluids and gases, rigid and deformable solids, cloth, explosions and other systems. To gain hands-on experience, students implement basic simulators for several systems. Topics include - Simulating Deformable Objects: Particle Systems, Mass spring systems, Deformable Solids & Fracture, Cloth, Explosions & Fire, Smoke, Fluids, Deformable active characters, Simulating Rigid bodies, Rigid bodies dynamics, Collision detection and handling, Controlling rigid bodies simulation; Simulating Articulated Bodies: Simulated characters in games, Optimization for character animation, Data driven approaches, Dynamic Response for 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++.

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.

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.

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 coursework; prior knowledge of Java required.

BUILDING LARGE INFORMATION SYSTEMS: This course will be a practicum in specifying, designing and documenting, building, testing and administering corporate-sized software projects that invariably have a database component, security and firewall issues, a web-based user interface, and special client programs running as applications elsewhere on a network. The course will examine one or more existing, commercial systems (such as the Blackboard system in use at Penn) and will address conceptual issues surrounding large software systems, such as ways to estimate project size, and ways to integrate different technologies into a maintainable system design. There will be substantial programming assignments using Microsoft Visual Studio .NET languages such as C# to create components of a larger system. Possible projects include a web interface, a client (and possibly server-side programming) for a SOAP/XML based web service, a Windows-based client with graphical interface, and rudimentary SQL database table and view design. The idea of using UML for code generation, and writing "built-in" unit tests for objects to automate future retesting will be examined.

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.

582. (CIS 482) Logic in Computer Science. (C) Prerequisite(s): CIS 260 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 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 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 and protocols, and query evaluation in databases.

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-Mainardi, Bonnet's theorem, lines of curvatures, geodesic torsion, asymptotic lines, geodesic lines, local Gauss-Bonnet theorem).

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, and observability for nonlinear systems; Forbenius' theorem; feedback and input/output 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 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, focussing 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.

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

This course emphasizes the review and understanding of current computer graphics, interaction, and virtual environment research techniques and problems. Research-level topics are based on recent ACM SIGGRAPH papers and special effects techniques, through student-led discussions and both oral and visual presentations. A software project is required.

665. GPU Programming and Architecture. (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 modelling, 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++.

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

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 (CSE 262, CSE 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) 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 modelling of human perception. Topics vary with each offering.

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, e.g., with SLD-resolution, with constraint satisfaction problems, with finite model theory and with automata theory.

700. Computer and Information Science Topics. (M)

One time course offerings of special interest.

899. Independent Study. (C)

For students studying a specific advanced subject area in computer and information science. Involves coursework and class presentations. A CIS 899 course unit will invariably include formally gradable work comparable to that in a normal 500 or 600 level course. This designation 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 40 course units of credit.

996. Research Seminar. (C)

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

999. Thesis/Dissertation Research. (C)

For students working on an advanced research program leading to the completion of master's thesis or PhD dissertation requirements.

COMPUTER & INFORMATION TECHNOLOGY (CIT)

L/R 591. Programming Languages and Techniques I. (C)

Introduction to fundamental concepts of programming and computer science. Principles of modern object-oriented programming languages: abstraction, types, polymorphism, encapsulation, and inheritance. Basic algorithmic techniques and informal complexity analysis. Substantial programming assignments in Java.

L/R 592. Mathematical Foundations of Computer Science. (C)

Foundations: Sets, Functions, Summations, and Sequences. Introduction to algorithms. Counting techniques: The pigeonhole principle, permutations and combinations. Discrete probability. Selected topics from Number theory and/or Graph theory.

593. Introduction to Computer Architecture. (C)

Introduction to fundamental concepts of computer architecture. Programming in C and at least one assembly language as a basis for understanding machine instructions and subroutine linkage conventions. Representation of numbers, characters and other information at machine level, including on virtual machines. Features of current operating systems.

594. Programming Languages and Techniques II. (C) Prerequisite(s): CIT 591 or consent of the instructor.

Basic data structures, including lists, stacks, queues, hash tables, trees, priority queues, and Java Collections. Algorithms, algorithm types, and simple complexity analysis. Development and implementation of program specifications. Software architecture and design methods, including modular program development, correctness arguments, and testing techniques. Concepts illustrated through extensive programming assignments in Java.

L/R 595. Digital System Organization and Design. (C) Prerequisite(s): CIT 593 or equivalent.

Introduction to fundamental building blocks of digital computer hardware such as transistors, logic gates and components built from them, as a basis for understanding how a computer operates at the hardware level. Basic networking, security, and other "under the hood" topics. Use of virtual machines to simulate hardware.

L/R 596. Theory of Computation. (C) Prerequisite(s): CIT 592 or equivalent.

Relations. Finite automata, regular languages, regular grammars, and applications. Pushdown automata, trees, context-free grammars, and applications. Turing machines. Introduction to computability and complexity theory.

597. Programming Languages and Techniques III. (C) Prerequisite(s): CIT 591 or equivalent.

Advanced Java programming and programming tools, with emphasis on developing for the Internet. Java topics will include serialization, synchronization, reflection, advanced I/O, and servlets. This course will cover current Internet-related technologies such as XML and JavaScript, and may include JDBC, UML, PHP, SOAP, and others. Substantial programming assignments, many in Java. May be taken by MCIT and CIS graduate students.