
HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY
SCHOOL OF COMPUTER SCIENCE
FALL 2017

HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

SCHOOL OF COMPUTER SCIENCE FALL 2017

MSc students can choose courses from either the BSc or MSc routes, but you must get permission from your home university and the department at Reykjavik University. It is not possible for BSc students to take their final thesis project during their exchange studies in the School of Computer Science.

Department Contact:

Please contact the department for information regarding; courses and course selection.

Undergraduate/ BSc/ BA / First cycle
Graduate /MSc/ Masters/ Second Cycle

Undergraduate/BSc
Graduate /MSc
Sigurbjörg Ásta Hreinsdóttir
astahr@ru.is

COURSES OVERVIEW

UNDERGRADUATE COURSES:

T-515-NOTH USER-CENTERED SOFTWARE DEVELOPMENT — COMPUTER SCIENCE

LEVEL: BSC ECTS: 6 | **PREREQS: Software and Design**

CONTENT: The course will focus on teaching methods for analysing, designing and evaluating software systems anticipating the users aspects in software development. Students will gain skills in using particular methods for analysing, design and evaluation user interfaces. Furthermore, other methods for analysing, designing and evaluating user interfaces will be described. Research on user centred software development methods will be described, when it is best to use each method and how practioners have ranked the methods. The integration of user centred software development methods into Scrum will be discussed, and the integration of user experience into lean software development. Furthermore experiences from industry will be a part of the course.

LEARNING OUTCOMES:

Knowledge: Be familiar with several methods for analysing the users´ needs for software systems
Be familiar with evaluation with and without the participation of users
Be familiar with guidelines for good user interface design
Be familiar with the integration of user-centered design methods in the Scrum software process
Skills: Be able to make a vision (ie, Visioning) for a software project and explain it
Be able to analyse the context of use for software systems
Be able to perform contextual inquiries and derive the results from affinity diagram
Be able to design an interface that is based on the relationship schema and test it with users
Be able to perform formal user evaluations
Competence: Be able to choose which user-centred design methods are suitable in different cases
Know the advantages and disadvantages of user centred design methods

TEACHING & ASSESSMENT: 3 lectures per week, one open session and two problem solving sessions. Usually there will be 2 lectures on Tuesdays and one on Thursdays. All the lectures will be recorded. In the open section, the students will suggest subjects to cover or state questions for the lecturer. In the problem solving sessions the students will collaborate on working on projects. There will often be individual assessment of involvement in these sessions that contribute to the final assessment in the course. Assignments and final exams. There will be 4 group projects with 3 - 4 students in each group. Two projects are 10% and two 15% (50% in total). Additionally there will be individual assignments, most of them 1%, that will add up to 10% in total. The written exam is 40% of the final grading.

T-513-CRNU CRYPTOGRAPHY AND NUMBER THEORY — COMPUTER SCIENCE

LEVEL: BSC ECTS: 6 | **PREREQS: Discrete Math, Calculus and Statistics, Algorithms**

CONTENT: We treat the basics of cryptography and number theory. We start with some classical ciphers and the the tools from number theory necessary for doing cryptography. We cover symmetric and asymmetric ciphers. Some topics from groups, rings and fields will be introduced and used, especially when we look at elliptic curve cryptography. There will be some programming exercises in addition to standard mathematical homework. You will use the programming language Sage to program with.

LEARNING OUTCOMES:

Knowledge:
- Know the purpose of cryptography and its uses throughout history
- Know the basics of number theory, especially relating to cryptography
- Know the Sage programming language, especially how to implement algorithms from number theory and cryptography
- Know the most common algorithms used in cryptography, e.g., the RSA public key system
- Know the basics of information theory
- Know the basics of finite fields and how they are used in cryptography
- Know the basics of elliptic curves and how they are used in cryptography
- How cryptography is applied, e.g., in multi-party computation, zero knowledge proofs, digital cash and voting systems

Skills:
- Use simple cryptographic methods to encrypt short texts by hand
- Write code in Sage to use powerful cryptographic methods to encrypt text
- Solve number theoretic problems, with pencil and paper, as well as with Sage
- Apply their knowledge of number theory to solve problems in other mathematical courses, especially where algebra is needed
- Use Sage as a tool in other programming and mathematical courses, for testing conjectures, drawing graphs, etc.

TEACHING & ASSESSMENT: Assessment Methods: 50% homework (4 best of 5 count) and 50% written final exam.
Textbook: An Introduction to Mathematical Cryptography, By: Hoffstein, Pipher and Silverman ISBN Number: 978-1-4939-1710-5

T-519-STOR THEORY OF COMPUTATION — COMPUTER SCIENCE

LEVEL: BSc **ECTS: 6** **PREREQS: Discrete Math, Algorithms**

CONTENT: The main topic of this course is the theoretical basis of computer science. Various types of finite automata are introduced and connected to the formal definition of a programming language. Turing machines are introduced as a theoretical model for computation. Computability is discussed and the classification of solvable and unsolvable problems. Finally there is a discussion of complexity classes and the classification of algorithmically hard and easy problems.

LEARNING OUTCOMES:

Knowledge:
- Know deterministic & undeterministic Finite Automata (DFA and NFA), regular languages & their most important properties.
- Know what it means that two such automata are equivalent.
- Know the Pumping Lemma for regular languages.

Skills
- draw DFAs and NFAs and describe in words the language they accept.
- Draw an NFA (or DFA) for a simple regular language from the description of that language.
- Describe the strings in a regular language from a regular expression that describes it.
- Write a regular expression that describes a simple regular language based on its description in words.

Competence:
- Use finite automata and their properties in computer science.
- Use the properties of context free grammars in programming.

TEACHING & ASSESSMENT: Home Assignments: 35%, Midterm: 15%, Final exam: 50%

T-603-THYD COMPILERS — COMPUTER SCIENCE

LEVEL: BSC ECTS: 6 **PREREQS: Programming Languages.**

CONTENT: Compilers are the most important part of a programming development environment. The course defines the function and objective of a compiler. Lexical analysis of programs is discussed in detail, regular expression and finite automatons defined and the use of Lex introduced. Top-down and bottom-up approaches in parsing are discussed precisely and the use of Yacc introduced. Implementation of error handling illustrated, particularly semantic analysis. Finally, code generation is covered. Construction of a compiler will be a large component of the course.

LEARNING OUTCOMES: Knowledge
- Understand the structure and design of compilers
- Understand the role and function of lexical analyzers, parsers and code generators
- Be able to get a theoretical foundation necessary for compiler construction
- Be able to design and build a simple compiler

Skills
- Be able to use regular expressions and finite machines when doing lexical analyzation
- Be able to use fragmented grammar and both above- and bottom up parsing methods
- Be able to use the software that makes lexical analyzers and parsers

TEACHING & ASSESSMENT: Compiler project (in 3 parts): 37%, Home exercises (3): 18%, Final exam: 45%,

T-409-TSAM COMPUTER NETWORKS — COMPUTER SCIENCE

LEVEL: BSc **ECTS: 6** **PREREQS: Data Structure, Computer Architecture**

CONTENT: We begin with a short overview of network systems and services. Then we will focus will be on the layers of the OSI and IETF models. The following network layers will be studied in the details: application layer (WWW, HTTP, DNS, SMTP, FTP etc), transport layer (UDP and TCP), network layer (link state routing and distance vector routing, IP, IP-addresses, link layer (MAC, Ethernet, Hubs and switches). Finally an introduction to more specific topics such as mobile networks, multimedia networking, and network security will be given.

LEARNING OUTCOMES:
- Explain the layers of the OSI and IETF network protocol stack and their interactions.
- Explain basic application layer protocols such as HTTP, SMTP, and P2P applications.
- Discuss and analyse the transport layer protocols TCP and UDP.
- Explain details of the IP protocol. Understand link-layer protocols and technology . Explain basic security terminology, security threats in networks, countermeasures, symmetric and public key
- Discuss performance assumptions and scalability in computer systems and networks.cryptography.
- Partition a network into subnets according to user specifications.
- Explain and analyse traces of real-world network traffic.
- Work with application layer networking interfaces.
- Understand how to write network services securely.
- Write simple network service using network primitives.
- Write secure network applications.

TEACHING & ASSESSMENT: Assessment Methods: Homework (24%), programming assignments (30%) and final exams (46%). Textbook: Computer Networking - A Top-Down Approach By: James F Kurose , Keith W Ross ISBN: 9780132856201

T-504-ITML INTRODUCTION TO MACHINE LEARNING — COMPUTER SCIENCE

LEVEL: BSC ECTS: 6 | **PREREQS: Algorithms, Calculus and Statistics, Discrete Math**

CONTENT: This course presents an overview of the field of machine learning, which deals with finding patterns and rules in large datasets. Such rules can then be used to predict outcomes of future events, for example with the aim of improving decision making in a wide range of business and manufacturing disciplines. In this course we will study machine learning techniques for classification and clustering as well as other selected techniques. In addition to introducing the underlying theory the the methods will be used to solve practical problems.

LEARNING OUTCOMES:

Knowledge:

- Know how data mining is carried out
- Recognize different types of training data and how to deal with common problems that arise, such as incomplete data
- Be familiar with key algorithms and models used for classification, including decision trees, set of rules, Naive Bayes, neural networks and support vector machines
- Know the basic algorithms used with clustering, including K-means
- Know the basic algorithms used to find relationships in data (e.g., association analysis)
- Be familiar with basic ideas behind evolutionary and reinforcement learning

Skills:

- Use software tools and programming libraries for data mining to categorize and cluster data
- Be able to set up problems and apply data mining techniques to solve them

Competence:

- Be able to determine the mechanical data learning strategies best suited to the solution of various practical problems, and ready to use data mining tools and libraries to their solution

TEACHING & ASSESSMENT: Assessment Methods: Homework assignments and in-class quizzes (20 %), Programming assignments (30 %), Final Exam (50%) Textbook: Introduction to Data Mining, By: P, Tan, M Steinback and V Kumar ISBN Number: 978-0-321-42052-7

T-511-TGRA COMPUTER GRAPHICS — COMPUTER SCIENCE

LEVEL: BSc ECTS: 6 | **PREREQS:**

CONTENT: Computer graphics is an increasing part of today's programmer projects. The first part of this course covers the use of the OpenGL library, vector tools for graphics, transformations of objects and polygonal meshes. The second part deals in more detail with three-dimensional drawing with emphasis on perspective, depth, light and colour. In the end, several issues regarding the implementation of a renderer are presented, in addition to curve and surface design. During the course students build several programs related to the course material.

LEARNING OUTCOMES:

Knowledge:

- Be familiar with the algorithms and calculations used when three-dimensional images are drawn on screen in real time (pipeline graphics), including, model transformations, perspective transformations, lighting, shading, clipping and rasterization.
- Be familiar with methods in OpenGL that implement these algorithms and calculations and how they are used in graphics applications such as computer games (OpenGL pipeline).
- Know how the flow in a graphical real-time application (i.e. computer game) is implemented, with respect to input, movement and drawing

Skills:

Be able to use the OpenGL standard to draw a three-dimensional image on a screen?

- Be able to implement a drawing loop which draws a motion picture, frame by frame, in real time.
- Be able to implement a programming loop that receives input and output, moves things, makes decisions and draws each frame with respect to camera angles and objects in a three-dimensional space.

Competence:

- Be able to implement three-dimensional video games and real-time animations with the OpenGL standard.

TEACHING & ASSESSMENT: Assessment Methods: Programming Assignments: 50%, Problem hand-ins: 10%, Final Exam: 40%. Textbook: Introduction to Computer Graphics By: David J. Eck.

E-402-STFO MATHEMATICAL PROGRAMMING — COMPUTER SCIENCE

LEVEL: BSC ECTS: 6
3 WEEK COURSE

PREREQS: Discrete Mathematics, Calculus and Statistics, Algorithms

CONTENT: Mathematics is generally discovered through experiments. Traditional tools for such experiments are pen and paper, and, of course, the mind. A (historically) recent addition to these tools is the computer. We will look at problems from several areas of mathematics and, in particular, how programming can be used as a means to better understand and ultimately solve those problems. This will involve designing and implementing algorithms, experimentation to make conjectures, and deductive/formal mathematics to prove conjectures. For programming we will use python/sage.

LEARNING OUTCOMES:

Knowledge

- After the course the students should know how computers and algorithms are used in research, both in mathematics and computer science.
- Students recognize linear programming as a method for solving problems
- Students recognize dynamic programming as a method for solving problems
- Students recognize search methods as a method for solving problems
- Students recognize brute force and other common solution methods for solving problems
- Students know several objects from discrete mathematics, such as permutations, graphs, games (like the Game of Life) and finite surfaces (tori and the Klein bottle).

Skills

- Students can use a computer to test conjectures and run simulations.
- Students can use dynamic programming to solve problems
- Students can use linear programming to solve problems
- Students can use search and other common methods to solve problems
- Students can choose an appropriate method to deal with different problems
- Students can prove certain problems by hand, where running simulations is to time-consuming.

Competence

- Students can use the Sage computer algebra system to assist them in other courses
- Students recognize which kind of problems can be solved with the solution methods treated in the course

GRADUATE COURSES

T-809-DATA DATAMINING AND MACHINE LEARNING — COMPUTER SCIENCE

LEVEL: MSC ECTS: 8

PREREQS: BSC DEGREE – COMPUTER SCIENCE

CONTENT: Pattern recognition system, classifier design cycle and learning. Statistical pattern recognition, Bayesian decision theory, maximum likelihood and Bayesian parameter estimation. Linear models for classification. Principal component analysis. Multilayer neural networks. Nonparametric methods: k-nearest neighbours and Parzen kernels. Kernel methods and support vector machines. Unsupervised classification, K-means clustering, Gaussian mixture models and expectation maximization. Combination of classifiers, bagging and boosting.

LEARNING OUTCOMES:

Knowledge: After the course the students should be able to recall, describe and define, the following terms: Pattern recognition system, classifier design cycle and learning. Statistical pattern recognition, Bayesian decision theory, maximum likelihood and Bayesian parameter estimation. Linear models for classification. Principal component analysis. Multilayer neural networks. Nonparametric methods: k-nearest neighbours and Parzen kernels. Kernel methods and support vector machines.

Skills: After the course the students should be able to apply the data mining methods and implement the machine learning algorithms presented in the course using standard programming languages such as Python or Matlab and software packages such as scikit-learn and Weka.

Competence: After the course the students should be able to design a suitable machine learning algorithm for a real world problem, evaluate its performance, compare different designs and implementations and interpret the results. The students should also be able to present findings and new results in the subject.

TEACHING & ASSESSMENT: 15% Quizzes 8-10 short computerized quizzes based on material from lectures and readings of which 50% is the outcome of the quiz and 50% is participation in the discussion afterwards. 10% Papers Two overview exams 35% Homework 4-5 papers that include computer exercises, problem solving and academic research 40% Course project One group project of 2-3 people (see detail in hand-out)

T-730-ASEN ADVANCED SOFTWARE ENGINEERING — COMPUTER SCIENCE

LEVEL: MSC **ECTS: 8** **PREREQS: BSC DEGREE – COMPUTER SCIENCE**

CONTENT: The focus of this course is on modelling and model checking reactive and concurrent systems, with a formal basis as well as a practical point of view. We start with formal semantics of concurrent models as basic transition systems presented by Manna and Pnueli. Then we introduce model checking and temporal logics of LTL and CTL using Huth and Ryan book. The actor-based language Rebeca (and possibly the coordination language Reo) and its (their) formal semantics will be covered next. We will also study Petri Nets. While introducing the models we will also look at the verification tools which support them. We will work with the widely used model checkers of NuSMVand Spin and also supporting tools of Rebeca and Timed Rebeca. A proper medium-size case study shall be selected and studied by the students and then modelled and model checked by one of the languages and tools which are introduced in the course. Model checking algorithms and temporal logic are not fully covered in this course.

LEARNING OUTCOMES:
- Know about transition systems as semantics of concurrent models
- Know about LTL and CTL as property languages for model checking
- Know Petri nets, coordination lanaguage Reo and actor-based language Rebeca and Timed Rebeca
- Be able to work with NuSMV, Spin model checkers, Afra (not an expert in)
- Be able to find their way in choosing a proper modelling language and analysis method for their application (not necessarily the best one!)

T-701-REM4 RESEARCH METHODOLOGY — COMPUTER SCIENCE

LEVEL: MSC **ECTS: 8** **PREREQS: BSC DEGREE – COMPUTER SCIENCE**

CONTENT: The course has two main aims. First, to provide the student with a compact foundation for the principles of scientific knowledge discovery, its foundational principles and concepts, the use of statistics for scientific investigation, and philosophical context out of which science came. Second, to provide the student experience in writing scientific text, focusing on scientific papers with results and data.

LEARNING OUTCOMES:
1 - Explain research, research methodologies, research in computer science;
2 - Choose a research subject and conduct a research project;
3 - Effectively write technical reports, papers, theses, proposals;
4 - Identify and summarize key points and issues in a body of scientific research, and present in a concise manner;
5 - Properly read and review a technical paper;
6 - Understand and explain professional and scientific ethics: credit, conflict of interest, and misconduct.

TEACHING & ASSESSMENT: The course is based on assignments, readings, and lectures. Students are encouraged to bring questions about each lecture´s topic to class, and raise them as the material is covered. This improves comprehension and incremental building of the student´s understanding.

T-763-INAR INTELLIGENT NARRATIVE TECHNOLOGIES — COMPUTER SCIENCE

LEVEL: MSC **ECTS: 8** **PREREQS: BSC DEGREE – COMPUTER SCIENCE**

CONTENT: The ability to create and revise stories is fundamental to human interaction, but computers are still in their infancy of doing either very well. In this course, we will explore computational storytelling from the perspective of Artificial Intelligence. We will read and analyze key papers from the literature, discuss how such technologies might be applied in different domains (e.g., computer games or training simulations), and obtain hands-on experience by building prototypes that extend previous research.

LEARNING OUTCOMES: Upon completion of the course, students should be able to:
- Discuss current challenges in computational storytelling
- Describe a variety of algorithms and techniques for addressing those challenges
- Present and critique related research, both orally and in writing
- Pursue original research that extends the concepts discussed in class
- Write a conference-level research report
- Identify computational storytelling projects that could be pursued as thesis research

TEACHING & ASSESSMENT: This course will combine presentations, discussions, and brainstorming in class with a hands-on term project. We will read and analyse key papers from the literature, discuss how such technologies might be applied in different domains (e.g., computer games or training simulations), and obtain hands-on experience by building prototypes that extend previous research.

T-785-IPIN INDOOR POSITIONING AND INDOOR NAVIGATION — COMPUTER SCIENCE

LEVEL: MSC **ECTS: 8** **PREREQS: BSC DEGREE – COMPUTER SCIENCE**

CONTENT: Problem of positioning and navigation, coordinate systems, maps - Range-free position estimation, models of radio frequency propagation, fingerprinting - Range-based position estimation, measurement methods, algorithms, dilution of precision - Ineartual measuring units, dead reckoning - Point estimation, error analysis, and Cramér-Rao Lower Bound - Simultaneous localisation and mapping (SLAM) - Navigation algorithms - Applications, for example in marketing and shop design, ambient assisted living, robotics

LEARNING OUTCOMES: On completion of the course the students will be able to:
- Explain the position estimation and navigation problem
- Explain algorithms for range-free and range-based position estimation
- Understand and apply principles of point estimation to positioning problems
- Explain algorithms for simultaneous localisation and mapping
- Explain navigation algorithms
- Know and understand criteria for selecting suitable navigation and position estimation methods
- Understand applications in marketing, health care, robotics

TEACHING & ASSESSMENT: 50% exercises, 50% written exam

T-810-OPTI OPTIMIZATION METHODS — COMPUTER SCIENCE

LEVEL: MSC **ECTS: 8** **PREREQS: BSC DEGREE – COMPUTER SCIENCE**

CONTENT: This course introduces the principal algorithms for linear, network, discrete, nonlinear, dynamic optimization and optimal control. Emphasis is on methodology and the underlying mathematical structures. Topics include the simplex method, network flow methods, branch and bound and cutting plane methods for discrete optimization, optimality conditions for nonlinear optimization, interior point methods for convex optimization, Newton´s method, heuristic methods, and dynamic programming and optimal control methods.

T-720-DEEP TOPICS IN DEEP LEARNING — COMPUTER SCIENCE

LEVEL: MSC **ECTS: 8** **PREREQS: BSC DEGREE – COMPUTER SCIENCE**

CONTENT: The goal of the course is to learn about Deep Learning methods and tools. The format of the course is a mainly reading course. We will start with a few lectures with an overview of the relevant material from Machine Learning. Then each student presents either a paper or a tool in a public talk/presentation. We will assign further students to study each paper/tool to encourage insightful discussions after each presentation.

LEARNING OUTCOMES:

Knowledge:
-Explain the basic concepts of deep learning
-Compare different neural network architectures
-Explain how deep neural networks are trained
Compare different software packages available for deep learning

Skills:
- Use one particular software package for deep learning for a small scale project

Competence:
- Summarize and critically evaluate research papers in deep learning

TEACHING & ASSESSMENT: grade composition: 30% paper presentation, 30% tool presentation, 15% reviewer/supporter for an active role in discussion about the paper both in class and after the presentation 25% participation in the course. For the participation grade we will look at attendance and active participation in the discussion both in class and after the public presentation.

T-811-PROB APPLIED PROBABILITY — COMPUTER SCIENCE

LEVEL: MSC **ECTS: 8** **PREREQS: BSC DEGREE – COMPUTER SCIENCE**

CONTENT: Overview and approach: This heuristically and practically motivated course will discuss the computation of probabilities of events, discrete/continuous random variables, conditioning of random variables. In addition, the course will also cover transformations of random variables, markov processes, and the applications of stochastic processes to queuing theory, derivatives/finance, decision theory and game theory.

LEARNING OUTCOMES: Understand the basic concepts of probability distribution functions and their role in the modelling of uncertain outcomes – both in the discrete and the continuous case. Use expectation values, variances and covariances to model various probabilistic phenomena

T-719-STO4 THEORY OF COMPUTATION — COMPUTER SCIENCE

LEVEL: MSC ECTS: 8 | **PREREQS: BSC DEGREE – COMPUTER SCIENCE**

CONTENT: The main topic of this course is the theoretical basis of computer science. Various types of finite automata are introduced and connected to the formal definition of a programming language. Turing machines are introduced as a theoretical model for computation. Computability is discussed and the classification of solvable and unsolvable problems. Finally there is a discussion of complexity classes and the classification of algorithmically hard and easy problems.

LEARNING OUTCOMES:

Knowledge:
- Deterministic and undeterministic Finite Automata (DFA and NFA), regular languages and their most important properties.
- Correspondence between finite automata and regular expressions.
- Know the Pumping Lemma for regular languages.
- Know context free grammars, context free languages and push-down automata and the correspondence between the concepts.
- Know Turing machines and different variants of those.

Skills:
- Describe the strings in a regular language from a regular expression that describes it.
- Write a regular expression that describes a simple regular language based on its description in words.
- Draw a Turing machine and describe the language it accepts.
- Show that a language is decidable by using closure properties for such languages.
- Decide if a language belongs to the complexity class P or NP.

Competence:
- Finite automata and their properties in computer science.
- Reason about how difficult problems are to solve according to their possible decidability and their complexity.

TEACHING & ASSESSMENT: There will be five home assignments, each worth 7%. Midterm: 15% Final exam: 50%. Both the midterm and the final exam will be closed book. For the MSc students, the final exam will consist of two parts. The final written exam will weigh 45% of the final grade. Moreover, there will be an oral exam on Rice´s theorem that will weigh 5% of the final grade. The oral exam will consist of a presentation, lasting at most 15 minutes including questions, using the whiteboard.

T-723-VIEN VIRTUAL ENVIRONMENTS — COMPUTER SCIENCE

LEVEL: MSC ECTS: 8 | **PREREQS: BSC DEGREE – COMPUTER SCIENCE**

CONTENT: This is a comprehensive course in both the theory and practice of Virtual Environments (VEs). Virtual Environments are simulations that engage the senses of users through real-time 3D graphics, audio and interaction to create an experience of presence within an artificial world. VEs are used in a variety of settings, including training, education, health, online collaboration, scientific visualization and entertainment. Their use is becoming more and more pervasive as hardware gets more capable of simulating reality in real-time (including graphics, physics and intelligent behavior). As part of the theoretical overview, the course will introduce the history of VEs, what kind of problems VEs have proven to be best at addressing, what are their shown limitations, what models of human-computer interaction apply to VEs and how these models are evolving and pushing the state-of-the-art in interactivity. The technical portion of the course will lead students through the construction and population of VEs in a very hands-on manner, covering topics such as world representation, real-time graphics and simulation issues, networked environments, avatars and interactive characters, event scripting and AI control, special real-time visual and aural effects and intuitive user interfaces.

LEARNING OUTCOMES:
- Know what constitutes a virtual environment, why they have been created throughout history and how they are used today.
- Be able to think critically about virtual environments as a user interface and design effective environments.
- Understand how humans construct a mental image of their environment using visual cues and how this can be exploited.
- Know the difference between presence and immersion, and understand how these may be measured.
- Understand the principles of effective action in virtual environments, including concepts such as flow, implicit constraints, explicit constraints and contextual action.
- Be familiar with the roles of characters in virtual environments and the common ways to make them autonomous and to animate them.
- Know what an avatar is and understand the issues that relate to level of control.
- Be familiar with the several techniques for constructing visual realism in virtual environments.
- Be able to create an interactive virtual environment in a scripting language and use a scene representation, models, terrain, lights, texturing, physics, animation, heads-up-display and shaders.

TEACHING & ASSESSMENT: Programming Assignments (x2) 20%, Final Project Proposal 5%, Final Programming Project 30%, Final Project Report 5%, Discussion Prep and Lab Work 10%, Final Written Exam 30%,