Syllabus SI4

Cursus d'Ingénieur en Informatique de Polytech Nice Sophia

Département Sciences Informatiques

Année universitaire 2026 - 2027

Semestre S7

Six unités d'enseignement

EIENAL7	**Algorithmique et complexité**	CM 12h	TD 30h	HNE 0h

Cours proposé en :

SI4

Responsable : **Florian Bridoux (Florian.BRIDOUX@univ-cotedazur.fr)**

Résumé :

Dans ce cours, nous étudions la calculabilité et la complexité algorithmique.

Nous nous intéressons aux questions suivantes : comment formaliser la notion de calcul ? Quels problèmes peut-on résoudre avec un ordinateur ? Comment comparer la complexité de deux problèmes informatiques?

Prérequis :

Les étudiants doivent être familiers avec

- la base de l'algorithmique (pseudo code) ;
- les bases de la théorie des langages (traité dans le cours « Langages, Compilation, Automates » en SI3).

Objectifs :

À la fin de ce cours les étudiants devront savoir:

- Comprendre et savoir dessiner des machines de Turing.
- Comprendre l'universalité du calcul des machines de Turing.
- Déterminer la bonne classe de complexité de problèmes de décisions simples en proposant un algorithme en pseudo code qui les décide (avec les bonnes propriétés selon la classe).
- Ils devront aussi pouvoir prouver qu'un problème est C-difficile pour une classe C via des réductions ou via le théorème de Rice.

Contenu :

*Machine de Turing : universalité du calcul, Thèse Church-Turing

*Calculabilité : théorème de Rice, énumérable et récursivement énumérable (appelé décidable et semi-décidable dans ce cours)

*Complexité en temps et en espace, classes de complexités : P, NP, coNP, PSPACE, EXPTIME, NEXPTIME…

*Questions ouvertes P = NP ?NP = coNP ? NP=PSPACE ? Question fermées : P!= EXPTIME.

Références :

- *Complexite algorithmique, 2014, Sylvain Perifel*
- *Langages formels, Calculabilité et Complexité, 2007, Olivier Carton*

Acquis :

Evaluation :

2 ou 3 contrôles continus.

EIIN714	« Programmation Multi Paradigmes en C++ »	CM 12h	TD 30h	HNE 20h

Responsable : **Sylvain.Lippi @univ-cotedazur.fr** et **équipe Amadeus de Mr Kevin YEUNG**

Résumé :

Le langage C++ est un langage de programmation à objets mais plus généralement multi-paradigmes qui est largement utilisé à la fois dans le monde industriel et académique. Il a été introduit originellement comme une extension du langage C mais a subi de nombreuses évolutions qui permettent aujourd'hui de nombreux styles de programmation avec un niveau d'abstraction élevé éloigné du C originel. On peut résumer ses avantages au triptyque : vitesse d'exécution / haut niveau de portabilité / grande disponibilité de bibliothèques. Cependant, le prix à payer est qu'il s'agit d'un langage complexe et qui exige une expérience importante avant d'acquérir une pratique satisfaisante. On exhibera en particulier les bogues typiques du programmeur C++ novice pour mettre en valeur les bonnes pratiques à adopter.

Prérequis :

- Langage C
- Fondamentaux de la programmation objet

Objectifs :

- Concepts de base de la programmation objets (classes, héritage, polymorphisme) appliqués au langage C++
- Mécanismes implicites du langage : constructeurs/destructeurs, conversions, opérateurs
- Gestion explicite de la mémoire
- Elements essentiels de la STL (Standard Template Library)
- Application de la généricité

Contenu :

- Sessions 1 à 5 : C++ de base
  - Concepts de base de la programmation orientée objet appliqués au langage C++
- Session 6-et suivantes: Modern C++

  Session 6:

  - Pourquoi le C++ moderne aujourd'hui ? Contextualisation
  - Rappel sur les Classes / Constructeurs / Destructeurs
    - Destructeurs
    - comportement par défaut par exemple sur les copies d'objets
    - Existence de pointeurs en C++, redéfinition de constructeur de copie
    - Introduction aux "smart pointers" comme résolution
    - Special members (copy)
    - new / delete
    - Introduction aux exceptions (destructeur noexcept, exception dans le constructeur, exception safety)

Session 7:

- Special members (move)
- Rule of 0, 3, 5

Session 8:

- Initialisation des objets (différentes manières d'initialisation, pièges à éviter)
  - value initialization
  - copy initialization
  - uniform initialization
  - Value initialization, initialization list, default constructor,...
- Smart pointer plus en détails
- Le concept "guard" en C++
- Composition

Session 9:

- Basics of STL (vectors, strings, algorithms, ...)
- Exception handling of STL functions
- Programmation Générique : Template

Session 10:

- std::all_of / std::any_of / std::none_of
- std::accumulate (reduce) / std::transform (map)
- lambda expressions
- C++20 ranges

Références :

- C++2011 Primer : Lippman, Lajoie, Moo
- A tour of C++ : Bjarne Stroustrup

Evaluation :

- TDs notés
- Examen de mi-semestre
- Examen final

EIIN716	Conception Logicielle (S7)	CM 12h	TD 30h	HNE 12h

Responsable : **Mireille Blay-Fornarino** (Mireille.blay@univ-cotedazur.fr)

URL: **https://lms.univ-cotedazur.fr/2023/course/view.php?id=11330**

Résumé :

Cet enseignement a pour but d'apprendre à développer une application de qualité en prenant en compte les exigences des utilisateurs. Cet enseignement abordera la spécification via des Use cases et des histoires utilisateurs, la conception orientée objets via UML et les designs patterns, et l'implémentation en utilisant des paradigmes avancés composants, aspects, …

Prérequis :

- Principes de la programmation par objets
- Programmation en java
- Bases de la gestion de projets

Objectifs :

- Savoir analyser et concevoir « objet » – Connaître l'essentiel de la notation UML
- Apprendre les User Stories – et les tests comportementaux associés (Utilisation de CUCUMBER)
- Connaître les patrons de conception les plus courants – savoir quand les appliquer et ne pas les appliquer
- Comprendre les architectures à base de composants et comprendre les enjeux – Focus sur l'injection de dépendances et les interfaces
- Mécanismes avancés pour la séparation des préoccupations : aspects, log, …

Contenu :

- Les cours présentent les spécificités des différents points présentés dans les objectifs.
- Les séances de TD sont basées sur un projet fil rouge en groupe permettant d'appréhender un projet de sa conception à sa maintenance en mettant en jeux chacun des principes énoncés en cours.
- Le planning prévisionnel précise certains points.

Références :

Voir https://lms.univ-cotedazur.fr/2022/course/view.php?id=1402&section=4

Compétences :

- CG1.2 Maîtriser les liens entre les disciplines et transposer les mêmes concepts d'un domaine à un autre, être capable de collaborer avec des spécialistes de disciplines connexes-- Niveau : Applications
- CG2.3 Maîtriser les différents aspects des systèmes d'information (fonctionnels, organisationnels, techniques), de leur conception à leur mise en œuvre et leur intégration tant d'un point de vue conceptuel qu'appliqué. Niveau: Applications

- CG3.1 Concevoir des modèles, systèmes et process en utilisant des méthodologies d'analyse, de conception et de modélisation, en connaissant leurs limites et sans perdre le sens de la réalité et du concret. Niveau : Maîtrise

Acquis :

- Savoir modéliser en UML une application à partir des spécifications littérales.
- Savoir exprimer sous la forme d'user story et des tests associés les fonctionnalités attendues du système et mettre en place les tests qui vérifient que les exigences sont respectées
- Savoir concevoir une architecture orientée objets qui utilise à bon escient les design patterns idoines.
- Savoir appliquer des concepts avancés de programmation dans son développement.

Evaluation :

Modalités d'évaluation			
QCM		**Nombre**	
Sur Devoir sur table	Oui : 40%	**Nombre**	À affiner
Sur Rendu de Projet	Oui : 40%	**Nombre**	À affiner
Sur Rendu de TP		**Nombre**	
Sur entretien individuelle		**Nombre**	
Autres	Oui : 20% (à l'étude évaluation par les pairs ; présentation ; tests en ligne)		

Planning prévisionnel :

Semaines 1 à 3 : (13/9, 20/9, 27/9) Introduction générale au génie logiciel dans l'ensemble du cycle de vie du logiciel et positionnement des enseignements du module. – Introduction à UML : les principes. Les éléments de syntaxe seront étudiés plus précisément en TD.

Semaines 4 à 5 : (4/10, 11/10) Éléments avancés pour un développement dirigé par les besoins : User stories, BDD, Estimations.

Semaines 6, 7, 9 : (18/10, 25/10, 8/11) Des principes d'affectation des responsabilités aux design patterns.

Semaines 10 à 13 : (15/11, 22/11, 29/11, 6/12) Paradigmes avancés de programmation : Des objets aux composants, injection de dépendances, aspects, et variabilité des logs.

EIEIN709	Deep Learning 1	CM 12h	TD 30h	HNE 6h

Cours proposé en : SI4

Responsable : **Diane Lingrand** (diane.lingrand@univ-cotedazur.fr)

Résumé :

Les réseaux de neurones : du neurone artificiel aux réseaux profonds.

Ce cours a pour but de donner une vision complète et précise des réseaux de neurones qui sont maintenant omniprésents dans notre monde (recommandation, traducteur, génération d'image, complétion de code...).

Nous partirons du neurone artificiel pour aller vers les réseaux de neurones simples et passer ensuite aux réseaux de neurones profonds. Dans une seconde partie, nous étudierons les réseaux utilisant des relations spatiales, notamment les CNNs pour les images, et des relations séquentielles (RNN/LSTM et TCN) et les modèles génératifs (VAE, GAN). Ce cours est un pré-requis pour le cours en S8, 'Modèles fondation'.

Les travaux pratiques seront effectués en python 3 avec la librairie pytorch.

Les mises en application concerneront aussi bien la détection et la reconnaissance d'objets dans les images, la segmentation d'images, que la classification de texte et la détection d'anomalies dans les séries temporelles. On terminera avec des applications à la génération de texte et d'images.

Prérequis :

Cours d'introduction à l'apprentissage automatique en SI3 (données numériques).

Programmation python.

Bases de mathématiques (produit scalaire, dérivées, fonctions usuelles)

Objectifs :

Contenu :

- Régression linéaire/logistique et le neurone simple
- MultiLayer Perceptron (2 semaines)
- Convolutional Neural Network (2/3 semaines)
- Autoencodeurs - SDAE/VAE - architecture pour segmentation sémantique
- Réseaux de neurones récurrents RNN - LSTM/GRU (2 semaines)
- Exemples adversaires - Réseaux de Neurones Génératifs Antagonistes (GAN)

Références :

Acquis :

En plus des objectifs : maîtrise des librairies numpy, matplotlib, scikit-learn, pytorc.

Evaluation :

Contrôle continu : petits contrôles en amphi, rendu de TP

Contrôle terminal portant sur l'intégralité de la matière (théorie et pratique)

EIENMI7	Middleware and Service Oriented Computing	CM 12h	TD 30h	HNE 12h

Cours proposé en plus de SI4 en :

AL	CyberSec	IA-ID	IHM	IoT-CPS	Ubinet	M1 EIT	M2 EIT	M2 Fintech
						X		X

Responsable : **Benjamin VELLA et Baude Françoise** (Françoise.Baude@univ-cotedazur.fr)

Résumé :

In this course, we study the main approaches for connecting software pieces in a high level manner, using adequate middleware(s). We also investigate the concepts middlewares are built upon.

Practical implementations will be done in Java, and in C#, using middlewares like Java RMI, gRPC, ActiveMQ, and .Net appropriate technologies for web services based interactions. The technologies will be all used in the scope of a semester wide project, mainly centered on .NET Visual Studio IDE, using C# as main language, with Microsoft' Web services technology. Study of heterogeneity in service oriented computing will be illustrated by interconnecting the core C# code with Java, Active MQ / Message Oriented Messaging Middleware.

Prérequis :

- Students must be proficient in Java programming or at least master object oriented models and must be familiar with networking concepts.

Objectifs :

- Understand and apply in practice the key concepts of distributed programming, either in an object oriented synchronous method invocation model, or in an message driven asynchronous model. It also presents and make use of the web services technology, both SOAP and REST based. General goal is to learn how to connect distributed applications in an interoperable way.

Contenu :

RPC and RMI concepts.

- Illustration with Java RMI and gRPC, in particular
  - Naming/Discovery service
  - Service definition including parameters
  - Multi threading management concepts
    - Some basic concurrency management tools offered in Java for instance

Message Oriented Middleware concepts and standards

- JMS and its illustration with Apache Active MQ
- Brief introduction to MQTT if time allows

Dynamic http servers

Clients and Web Servers

- WS-SOAP Clients and Servers
- REST Clients and servers

Illustration in C#, in .NET Visual Studio, using the WCF Framework and the "ABC" service model

Références :

gRPC online resources
Java RMI: Designing & Building Distributed ApplicationsWilliam Grosso
.NET Web Services: Architecture and Implementation, Keith Ballinger · 2003
Java Message Service, Richard Monson-Haefel, David A. Chappell · 2002
ActiveMQ in Action, Dejan Bosanac, Bruce Snyder, Rob Davies · 2011

Acquis :

Basic gRPC or Java RMI applications development and deployment
Client server web services applications
Interoperable services integration, using web services and message queues

Evaluation :

A full semester project, developed in small teams, but defended through an individual oral defense.

EIEISE7	Software Security	CM 12h	TD 30h	HNE 15h

Cours proposé en : SI4

Responsable : **ROUDIER Yves (Yves.Roudier@univ-cotedazur.fr**)

Résumé :

Software is the backbone of today's digital economy, networks, and embedded systems. This course aims at:

- achieving a better knowledge of computer security challenges, i.e., understanding the risks posed by code and software architectures;
- acquiring know-how in secure software engineering, i.e., how to detect vulnerabilities, design a secure architecture and develop secure programs, including cryptography-based countermeasures and access control mechanisms.

In addition to labs, a secure software development project will be carried out throughout the course to better understand secure software lifecycle. Programming will be in C/C++, Python, and Rust.

Prérequis :

- Students should at least be familiar with basic operating system engineering concepts taught in 3rd year (pointers, software execution, stack and heap, malloc, etc.), and ideally proficient in C (or C++).

Objectifs :

- Mastering defensive and secure programming
- Introduction to security testing (mostly fuzzing), DevSecOps, software reviews and penetration testing (mostly reverse engineering), and antivirus and EDRs
- Introduction to cryptography and its use for distributed application security
- Ability to understand and express security requirements and security by design

Références :

- Computer Security, A Hands-On Approach. Third Edition, 2022. Wenliang Du. **ISBN-10 :** 1733003959 / **ISBN-13 :** 978-1733003957

Evaluation :

- One group project
- One individual test

Semestre S8

Dix unités d'enseignement

	Algorithmique 2 (S8)	CM —	TD —	HNE —

Cours proposé en :

SI4

Responsable : Dorian Mazauric (INRIA Sophia Antipolis – Méditerranée)

Résumé :

Algorithmique avancée et médiation scientifique dans le cadre de Terra Numerica.
Ateliers Terra Numerica et algorithmique.
(Re)Découverte de concepts mathématiques/informatique avec des ateliers originaux de médiation et travail sur des projets algorithmiques (en lien avec les ateliers existants).

Objectifs :

L'idée serait d'aller jusqu'à créer une ressource (au sens large) et possiblement l'animer. Qu'est ce que cela veut dire ? Il s'agit de s'attaquer à un problème d'algorithmique réputé difficile, et d'illustrer son usage dans un cadre pratique. Puis, de développer une solution informatique qui mette en œuvre le tout, dans un objectif délibérément « grand public ». A titre d'exemple : Fête de la Science : les pouvoirs insoupçonnés du ballon rond (Inria) | WebtimeMedias
Outre l'étude poussée de certains problèmes algorithmiques, l'originalité de cet enseignement est de réfléchir à leur usage au sein d'applications réalistes en allant jusqu'à la mise en œuvre dans un cadre réel. D'autre part, ayant une visée de culture scientifique à destination du grand public, la démarche sera entièrement guidée par l'objectif de produire une « ressource », utilisable dans le contexte d'ateliers qui pourront être déclinés face à des publics variés. La participation des étudiants à de tels ateliers sera également souhaitable. La production de ressources en langue anglaise serait un plus.

Références :

TerraNumerica@Sophia – vers une Cité du Numérique – Terra Numerica (terra-numerica.org)

	Bases de données : format de données et transformations (S8)	CM —	TD —	HNE —

Cours proposé en :

SI4

Responsable : Catherine Faron

Résumé :

Ce cours présente les langages et technologies pour la structuration de bases de données XML et JSON : le langage XML, le langage XPath d'expressions de localisation, le langage XML Schema de définition de modèles de documents XML et le langage XSLT de transformation de documents XML, le langage JSON et le langage JSON Schema de définition de modèles de documents JSON. Une introduction aux langages du web des données (structuration des données en graphes de connaissances) est faite en fin de cette partie.

Objectifs :

- Maîtrise des concepts de documents et données structurés, de modèle de documents, et de transformation de documents.
- Maîtrise des langages XML, XPath, XML Schema, XSLT, JSON, JSON Schema
- Maîtrise de la modélisation de données en XML ou JSON, schémas XML, schémas JSON
- Maîtrise du traitement de données XML avec XSLT

Contenu :

- XML et espaces de nommage
- XPath : Expression de chemins de localisation
- XML Schema : modèles de documents XML
- XSLT : transformation de documents XML
- JSON et JSON Schema et comparaison avec XML et XML Schema

EIN834	Deep Learning 2	CM 12h	TD 30h	HNE 6h

Cours proposé en : SI4

Responsable : **Frédéric Precioso** (frederic.precioso@univ-cotedazur.fr)

Résumé :

Les modèles fondations: des transformers aux modèles fondations

Tout le monde, dans le grand public, connait aujourd'hui ChatGPT, Dall-e, Stable Diffusion ou encore Midjourney. Si vous programmez vous connaissez copilot mais peu d'entre-vous savent comment ces outils fonctionnent, quels sont leurs mécanismes internes.

Le but de ce cours et de donner aux futurs ingénieurs en informatique les bases pour comprendre ces familles de modèles afin de mieux les utiliser ou de les adapter à leurs besoins et d'être capables de suivre les évolutions très rapides qui s'annoncent. Nous ne vous transmettrons pas de listes de trucs et astuces pour mieux interroger ChatGPT (qui serait de toute façon obsolète avec la prochaine version du modèle), mais en vous expliquant dans le détail comment ça fonctionne, vous comprendrez comment manipuler efficacement ces modèles et les prochains à venir.

Les mises en applications viseront des tâches multiples en vision par ordinateur, analyse du langage, analyse du son, et de données multiples (image, vidéo, son, et texte)...

Prérequis :

Cours d'introduction à l'apprentissage automatique en SI3 (données numériques).

Cours de Machine Learning de SI4 S7 : les réseaux de neurones: du neurone artificiel aux réseaux profonds

Programmation python. Librairies matplotlib, scikit-learn, pytorch

Bases de mathématiques (produit scalaire, dérivées, fonctions usuelles)

Objectifs :

Compréhension fine des modèles fondations pour être capable de suivre les évolutions rapides de ce domaine.

Contenu :

- RNN seq2seq - Modèles d'Embedding de texte
- Transformers
- Fine-tuning / prompt engineering / l'humain dans la boucle
- Modèle fondation
- Ressources (données, calculs): biais, éthique, éco-responsabilité (2 semaines)

Références :

Acquis :

En plus des objectifs : maîtrise approfondie de la librairie pytorch

Evaluation :

Contrôle continu : petits contrôles en amphi, rendu de TP

Contrôle terminal portant sur l'intégralité de la matière (théorie et pratique)

EIEIN832	IA Embarquée, Capteurs et Actionneurs	CM 12h	TD 36h	HNE 12h

Cours proposé en :

AL	CyberSec	IA-ID	IHM	IoT-CPS	Ubinet	M1 EIT	M2 EIT	M2 Fintech
				X		X		

Responsable : **Miramond Benoit** (Benoit.MIRAMOND@univ-cotedazur.fr)

Résumé :

Ce cours s'intéresse au déploiement de l'IA dans les applications embarquées. Ce secteur spécifique de l'IA, également appelé Edge AI, est soumis à des contraintes fortes en termes de consommation d'énergie et de puissance de calcul disponible. Déployer des algorithmes de Deep Learning dans de telles conditions fait appel à plusieurs notions : intelligence artificielle, programmation embarquée, systèmes temps réel, capteurs et actionneurs, traitement de signal.

Prérequis :

Les étudiants doivent

- avoir un bon niveau en programmation C
- connaître les principes de la représentation des nombres en binaire
- avoir des notions de deep learning.

Objectifs :

- Développer des programmes compacts sur cible embarquée
- Comprendre les contraintes spécifiques au domaine de l'embarqué et des systèmes temps réel
- Sensibiliser à la notion d'IA frugale et la conception de modèles d'IA à faible consommation d'énergie
- Concevoir un projet de capteur intelligent autonome intégrant des algorithmes d'IA au service de l'environnement

Contenu :

- Le module est principalement organisé autour de la réalisation d'un projet de conception de réseau de capteurs intelligents. Ce projet se base sur des cartes à micro-contrôleurs développées au LEAT et sur l'environnement logiciel MicroAI associé pour la compression de réseaux de neurones sur cible embarquée. Les cours fournissent les principes fondamentaux de la programmation embarquée, des systèmes temps réel et de la compression de réseaux de neurones pour l'Edge IA. Les TPs sur cibles embarquées mènent l'étudiant vers la réalisation de son propre projet de capteur intelligent en fin de cours.

Références :

- Introduction aux systèmes embarqués temps réel - Fondamentaux et études de cas: Conception et mise en oeuvre, Emmanuel Grolleau, 2018.
- Quantization and deployment of deep neural networks on microcontrollers, PE Novac, G Boukli Hacene, A Pegatoquet, B Miramond, V Gripon, Sensors 21 (9), 2984, 2021

Acquis :

- Comprendre les contraintes spécifiques au domaine de l'embarqué et des systèmes temps réel
- Comprendre les enjeux et les défis de l'Edge AI à l'heure de l'IoT, des réseaux de capteurs et de la transition énergétique.

Evaluation :

- Une note de projet final et une note de contrôle.

	Embedded AI, Sensors and Actuators	CM _	TD _	HNE _

Summary:

This course focuses on the deployment of AI in embedded applications. This specific sector of AI, also called Edge AI, is subject to strong constraints in terms of energy consumption and available computing resources. Deploying Deep Learning algorithms in such conditions calls for several notions: artificial intelligence, embedded programming, real-time systems, sensors and actuators, signal processing.

Prerequisite:

Students should
- have a good level in C programming
- know the principles of binary number representation
- have notions of deep learning.

Objectives:

- Develop compact programs on embedded targets
- Understand the constraints specific to the embedded domain and real-time systems
- Become aware of the notion of frugal AI and the design of low power consumption AI models
- Design an autonomous smart sensor project integrating AI algorithms to monitor the environment

Content:

- The module is mainly organized around the realization of a smart sensor network design project. This project is based on microcontroller boards developed at LEAT and on the associated MicroAI software environment for the compression of neural networks on embedded targets. The courses provide the fundamentals of embedded programming, real-time systems and neural network compression for Edge AI. Workshops on embedded targets lead the student to the realization of his own smart sensor project at the end of the course.

References:

- Introduction to Real-Time Embedded Systems - Fundamentals and Case Studies: Design and Implementation, Emmanuel Grolleau, 2018.
- Quantization and deployment of deep neural networks on microcontrollers, PE Novac, G Boukli Hacene, A Pegatoquet, B Miramond, V Gripon, Sensors 21 (9), 2984, 2021

Acquired:

- Understand the specific constraints of the embedded and real-time systems domain.
- Understand the issues and challenges of Edge AI in the era of IoT, sensor networks and energy transition.

Assessment:

- A final project grade and a control grade.

EIEIIH8	Creating Interactive Virtual Worlds	CM 12	TD 30	HNE _

Cours proposé en SI4

Responsable : **Hui-Yin WU** (hui-yin.wu@inria.fr)

Résumé :

This course is about creating interactive 3D virtual worlds. We will provide a broad overview of the domain of computer animation and graphics, and 3D interaction design. The first part of the course will focus on 3D modeling and animation technologies, coupled with practical work using Blender. The second part of the course focuses on game engines and interactive 3D, with hands-on development in Unity. The third part then introduces advanced topics in interactive graphics, with students creating their own final project. The course is meant to be a first introduction to 3D animation and virtual environments that can be followed by advanced studies in virtual and augmented realty, computer graphics and animations, 3D video game design, and 3D interaction and visualization.

Prérequis :

- Students must have good programming skills and knowledge of object-oriented programming
- The willingness to make mistakes, explore possibilities, be creative, search for and/or develop solutions
- The need also to own a computer that has sufficient capacities, in order to run software like Blender and Unity

Objectifs :

- Global understanding of the principles of 3D technologies and animations that allow one to go from idea to fully-fledged 3D worlds
- Establish synergy with domains of virtual technologies, design, user experience, and interaction
- Have created an interactive 3D application using Blender and Unity
- Can pursue more specialized or targeted courses in the domain if desired

Contenu : (Total 11h CM 30h TD)

Part 1 – 3D animation and modeling

Week 1:

- Course: Introduction to virtual worlds
- TD: Introduction to Blender and 3D modeling

Week 2:

- Course: The animation pipeline
- TD: Animation and rigging

Week 3:

- Course: Cinematography and lighting
- TD: Cinematography and rendering
- *Evaluation: Individual animation project due*

Part 2 – Interactive 3D technologies

Week 4:

- Course: Game engines and design patterns

- TD: Introduction to Unity – Racing game (I)

Week 5:
- Course: Narrative and AI
- TD: Introduction to Unity – Racing game (II)

Week 6:
- Course: Physics, particles, and deformations
- TD: Introduction to Unity – Racing game (III)

Week 7:
- Course: Procedural content generation
- TD: *Evaluation – Presentation of game design document*

Part 3 – Advanced topics

Week 8:
- Course: Advanced topics – Extended reality and the metaverse
- TD: Group project

Week 9:
- Course: Advanced topics – Immersive analytics
- TD: Group project

Week 10:
- Course: Advanced topics – Crowd simulation
- TD: Group project

Week 11:
- Course: *Evaluation – Exam*
- TD: Group project

Week 12:
- No course
- TD: *Evaluation – Final project presentation*

Références :

- Beane, A. (2012). *3D animation essentials*. John Wiley & Sons.
- Haigh-Hutchinson, M. (2009). *Real time cameras: A guide for game designers and developers*. CRC Press.
- Gregory, J. (2018). *Game engine architecture*. AK Peters/CRC Press.
- Jerald, J. (2015). *The VR book: Human-centered design for virtual reality*. Morgan & Claypool.
- Reeves, W. T. (1998). Particle systems—a technique for modeling a class of fuzzy objects. In *Seminal graphics: pioneering efforts that shaped the field* (pp. 203-220).
- Chandler, T., Cordeil, M., Czauderna, T., Dwyer, T., Glowacki, J., Goncu, C., ... & Wilson, E. (2015). Immersive Analytics. In 2015 Big Data Visual Analytics (BDVA)(2015). In *IEEE, Sept* (pp. 1-8).

Acquis :

- Basic competence in Blender 3D modeling and animation
- Intermediate competence in 3D application development with Unity
- A good global understanding of 3D technologies, approaches, and important topics
- Capacity to propose and prototype solutions involving 3D technologies

Evaluation :

One individual project on 3D animation (10%), one group project (50%), final exam (40%)

EIEIID8	Architecture Logicielle / DevOps avancé (S8)	CM 12h	TD 30h	HNE 12h

Responsable : **Philippe Collet** (Philippe.COLLET@univ-cotedazur.fr)

URL: **https://lms.univ-cotedazur.fr/2025/course/view.php?id=4017**

Résumé :

Ce cours est une introduction à l'architecture logicielle et à des principes de DevOps plus avancés que ceux introduits dans les projets de l'année précédente. Partant des notions de composants logiciels et d'injection de dépendances vus en conception logicielle, le cours introduit les concepts d'interopérabilité, de travail aux interfaces, et de mapping objet-relationnel. La mise en œuvre s'appuie sur une architecture Web N-tiers pour illustrer l'interopérabilité.

Les principes enseignés sont appliqués dans un projet fil rouge réalisé en équipe, demandant une forte autonomie et un apprentissage basé sur l'essai-erreur.

Prérequis :

- Maîtrise de tous les principes de la programmation par objets
- Très bon niveau en Programmation Java
- Maîtrise du génie logiciel (tests unitaires, mock, mesure de la qualité logicielle avec un outil comme Sonar)
- Capacités à travailler en projet de développement agile en équipe (commandes git, pull requests, user stories, Behavioral Driven Development, intégration continue)
- Maîtrise de la modélisation UML
- Notions d'architecture back-end, d'API et de services
- Bases de réseaux : configuration et programmation
- Maîtrise de l'environnement Unix et programmation shell en lignes de commande
- Curiosité et autonomie en développement logiciel (commandes git avancées, pull requests, intégration continue)

Objectifs :

- Comprendre les principes des serveurs d'applications
- Concevoir et réaliser un système à base de composants hétérogènes dans une architecture en couches
- Concevoir et réaliser un mapping objet-relationnel dans une approche par composants et objets
- Concevoir des applications d'entreprise en respectant les bonnes pratiques, notamment de conception d'API
- Construire et déployer une chaine de compilation, intégration et déploiement logiciel en suivant les principes DevOps
- Construire et composer des systèmes et sous-systèmes communicants par containerisation
- Spécifier et mettre en œuvre des tests d'intégration et des tests fonctionnels de bout en bout

Contenu :

- Introduction à l'Architecture Logicielle et aux serveurs d'application
- Architectures 3-tiers (présentation, domaine, persistance)
- Gestion de l'hétérogénéité et de l'interopérabilité
- Tests d'intégration et fonctionnels
- Intégration et déploiement continus, automatisation
- Production et publication d'artéfacts logiciels
- Gestion de la persistance et mapping objet relationnel
- Pipeline et serveur de CI/CD
- Containerisation avancée

Références :

- Beautiful Architecture, Diomidis Spinellis & Georgios Gousios
- Patterns of enterprise application architecture, Martin Fowler
- Software architecture in practice, Bass, Clemens and Kazman
- Software System Architecture, Rozansky and woods

Compétences :

- CG1.2 Maîtriser les liens entre les disciplines et transposer les mêmes concepts d'un domaine à un autre, être capable de collaborer avec des spécialistes de disciplines connexes-- Niveau : Maîtrise
- CG2.3 Maîtriser les différents aspects des systèmes d'information (fonctionnels, organisationnels, techniques), de leur conception à leur mise en œuvre et leur intégration tant d'un point de vue conceptuel qu'appliqué. Niveau: Maîtrise
- CG3.1 Concevoir des modèles, systèmes et process en utilisant des méthodologies d'analyse, de conception et de modélisation, en connaissant leurs limites et sans perdre le sens de la réalité et du concret. Niveau : Maîtrise

Acquis :

- Comprendre les principes d'interopérabilité sur le Web
- Maîtriser l'injection de dépendances et de l'inversion de contrôle
- Maîtriser un mapping objet-relationnel
- Justifier des choix de décomposition et d'interfaçage de composants
- Justifier des choix sur des styles de service
- Comprendre la différence entre composants stateful et stateless
- Comprendre les principes d'interception et d'intercession dans les architectures logicielles
- Maîtriser la création de tests d'intégration dans une chaine d'intégration continue
- Maîtriser la création de tests fonctionnels de bout en bout sur une architecture Web à base de composants
- Utiliser une chaîne d'intégration et de déploiement continus de niveau industriel
- Appliquer les principes de containerisation d'application
- Maîtriser la composition de containers
- Être capable d'appréhender facilement une nouvelle pile technologique de développement, comme attendu dans les métiers de l'architecture logicielle et du DevOps

Evaluation :

Modalités d'évaluation			
QCM		**Nombre**	
Sur Devoir sur table	Oui	**Nombre**	1
Sur Rendu de Projet	Oui	**Nombre**	2
Sur Rendu de TP		**Nombre**	
Sur entretien individuelle		**Nombre**	
Autres			

EIEIN820	Parallelism	CM 12h	TD 30h	HNE 12h

Cours proposé en plus de SI4 en :

AL	CyberSec	IA-ID	IHM	IoT-CPS	Ubinet	M1 EIT DSC	M2 EIT DSC	M2 Fintech
		X				X	X	X

Responsable : **Baude Françoise** (Françoise.Baude@univ-cotedazur.fr)

Résumé :

In this course, we study multicore and accelerators parallel programming concepts using high-level librairies or languages, after a short introduction on parallel machines architectures. We start by recalling typical problems faced by co-existence of multiple threads (race condtions, deadlocks), and present standard concurrency control methods and synchronisation tools. Practical illustrations are done in Posix threads, OpenMP (in C), and in Cuda (in Python).

We study the main algorithmic model for writing parallel algorithms using a shared-memory and an unbounded number of processors: the PRAM model and its different flavours. Some typical problems that can be solved in an efficient way in parallel are studied. Basic complexity measure tools are presented and applied to the studied algorithms.

A short introduction of Quantum Computing by Amadeus is included to illustrate how potentiality of parallelism can be exploited on quantum computers/accelerators.

Prérequis :

- Students must be proficient in programming, as C and Python, and be aware of multi threading or multi processes basic concepts. Students must know how to apply basic complexity measurements on sequential algorithms.

Objectifs :

Understand and apply in practice the mechanisms of shared memory parallel programming, on multicore, manycore and GPUs. Solve typical easy to parallelize problems using task or data parallelism using standard libraries like Posix threads, OpenMP, or Cuda. General goal is to learn how to face typical problems of multiple threads, and take benefit of multi core or accelerators hardware while understanding the theoretical and practical limitations or difficulties of parallel computing.

Contenu :

- Concurrency control mechanisms in Operating Systems and typical programming languages: mutual exclusion, deadlocks, synchronization basic tools (locks, mutexes, monitors)
- Introduction to parallel architecture models
- Moore law and its limits
- The supercomputers and rankings
- Various sources of concurrency and parallelism: task parallelism, data parallelism, and the corresponding low-level or high-level primitives to manage the parallelism in programming languages
- Posix threads and OpenMP libraries used in the C language or in Java language : concepts, parallel regions, synchronisation, access control to variables, scheduling of tasks on threads

- Limits of parallelisation : Amdhal and Gustafson laws
- Introduction to GPU architectures and their SIMD oriented programming model
- Programmation of kernels with Cuda in Python (Numba)
- Short introduction to distributed memory parallel architectures built on specific networking topologies
- Short introduction of quantum computing

Références :

- The art of multiprocessor programming , 2008, Maurice Herlihy and Nir Shavit
- Using OpenMP Portable Shared Memory Parallel Programming, Chapman, Rost, van des Pas, 2007.
- Parallel Algorithms, Henri Casanova, Arnaud Legrand, Yves Robert, 2007, available at https://www.researchgate.net/publication/241684993_Parallel_Algorithms.
- M. Gengler, S. Ubeda, F. Deprez, Initiation au parallelisme - Concepts, architectures et algorithmes, Masson 1996

Acquis :

How to think in parallel, how to write simple algorithms on the theoretical PRAM models, and thus, how to prepare future concrete parallel implementations, on shared memory parallel architectures, like multicore or GPGPU ones.

Use parallel shared memory parallel programming approaches to accelerate sequential programs

Manage concurrency with synchronisation tools in multi-threaded programs, be there massively parallel or not.

Evaluation :

One or two labs using Posix threads or Java concurrency controls, OpenMP; one lab using Cuda or one dedicated written exam; one or two written individual exams (for a total of approximately 80% of the course whole mark)

EIEIN831	Programmation Fonctionnelle	CM 6h	TD 15h	HNE 6h

Cours proposé en plus de SI4 en :

AL	CyberSec	IA-ID	IHM	IoT-CPS	Ubinet	M1 EIT DSC	M2 EIT DSC	M2 Fintech
		X				X	X	X

Responsable : **Urso Pascal** (Pascal.Urso@univ-cotedazur.fr)

Résumé :

In this course, we study the main principles behind functional programming and experiment them using the functional paradigm available in Java.

Prérequis :

- Students must be proficient in programming in particular in Java.

Objectifs :

- Understand and apply in practice the mechanisms of functional programming through dedicated libraries or languages offering functions as firt class citizens
- Design lambdas, high-order functions, understand stream and laziness in function evaluation

Contenu :

- Introduction to Lamda calculus and Functional languages
- Functional programming paradigm in imperative languages
- Specific data types : records, immutable collections, lists (car, cdr)
- Functional interfaces
- Anonymous functions
- High order functions
- Curryfication
- Streams
- Lazy evaluation
- Introduction to Functions as a Service

Références :

Learning Java Functional Programming - Richard M Reese:2015

Acquis :

Use functional style in programming languages, including not purely functional ones as Java

Evaluation :

50% a written individual exam, and 50% for a ranked lab or practical programming test

	Des Réseaux Couches Bases aux Protocoles Internet	CM 12h	TD 42h	HNE 12h

Cours proposé en : SI4

Responsable : LOPEZ PACHEZCO Dino (dino.lopez@univ-cotedazur.fr)

Résumé :

This lecture is composed of three main parts to provide a wide view of the network protocol and administration.

First, we focus mainly on the lower layers of the protocol stack and current trends in IP-based networks. More specifically, we introduce the Layer 2 protocols (Ethernet and Wi-Fi). Then, we introduce the challenges of congestion control on the Internet and the future trends, as well as the importance, advantages, and disadvantages of network virtualization and middleboxes (e.g. NATs).

Since Web-based approaches are among the most popular for distributed software applications, the second part introduces the HTTP protocol and its various features. The focus is on the administrative data of the protocol and the main data formats that are supported.

Finally, we provide a quick introduction to network security. After explaining the principle of attacks at various layers, we introduce classical network security architectures like firewall-protected corporate networks or VPNs. We finally conclude with a presentation of Internet cryptographic protocols and security infrastructures.

Prérequis :

Good knowledge of the CLI and management of Linux systems, IPv4 addressing, routing and layer 2 retransmission. Students must be familiar with networking tools, such as ping, iperf, wireshark, tcpdump. Good knowledge of the most popular networking applications (DNS, DHCP, SSH, SMTP, HTTP, …).

Objectifs :

The student must be aware of the impact that the medium access protocols and congestion might have on the performance of distributed applications. Also, they must understand the complexity that applications will face on the Internet due to the large variety of networking services and network virtualization techniques. Finally, they should be aware of the network security threats they have to address when designing and developing distributed applications.

Contenu :

- The IEEE 802.3 protocol and STP
- The IEEE 802.11 protocol
- Congestion control protocols
- Introduction to Network Function Virtualization (NFV)
- Middleboxes and NATs
- Socket programming
- HTTP and web sockets
- Network attacks on Internet protocols and their analysis
- Firewalls and VPNs
- Internet Security Infrastructures and Cryptographic Protocols (PKIs, key distribution, DNSSEC ...)

Références :

- Computer Networking - James Kurose, Keith Ross. Pearson
- “802.11 Wireless Networks – The definitive guide” - Matthew S. Gast. O’Reilly.

Acquis :

Evaluation :

- Mini QCMs all along the sessions
- written exams

NEW	PS8 : Full Stack Developer	CM 12h	TD 20h	HNE 80h

Cours proposé en :

FISE4

Responsable : **Benjamin Vella** (benjamin.vella@univ-cotedazur.fr)

Résumé :

In this course we will create a website containing a simple online multiplayer game. We will study and implement a full stack of development: modeling and development of a front side in HTML5, CSS3 and JS applying Design, User eXperience, and User Retention concepts; and a back side with a Node.js API and HTTP Server, a NO-SQL DataBase, and Business services such as an Artificial Intelligence able to play the game. Communications will include HTTP(s) and WS(s) requests, and the application will run on Docker containers and be hosted online.

Prérequis :

- Students needs basic understanding of front-end technologies, network communications, and databases.

Objectifs :

- Understanding the different parts of a Full Stack application.
- Building a multi-page website.
- Applying User eXperience and User Retention concepts to any customer-facing project.
- Handling sockets and HTTP requests.
- Implementing real time interactions with the users.
- Compartmentalizing a server's processes to handle users following the same flow at different times.
- Developing an Artificial intelligence for the game.
- Getting basic working knowledge of NO-SQL Databases.

Contenu :

Références :

- HTML/CSS/JS
- Node.Js
- JSON
- Web Sockets
- JWT
- IA (Minimax and Monte Carlo for Connect 4; A Knowledge-based Approach of Connect-Four by Victor Allis)
- ELO

Acquis :

Cf. "Objectifs"

Evaluation :

The project's evolutions will be graded:

- Wireframes (10%)
- Artificial Intelligence (10%)

- Docker Integration (10%)
- Full project before the full-time week (35%)
- Final evaluation at the end of the full-time week (35%)