Stuttgart Media University

Course Catalog
Games Technology

Course Title	**Theory of Game Development**
	Bachelor Level Course (Typically taken during 2nd or 3rd year)
Course No	113520
Lecturers name	Prof. Dr. Stefan Radicke
Teaching language	English
Credits (ECTS)	6
Teaching/learning methodology	lecture
Total workload	180 hours
Contact hours per week	45 hours teaching time
Type of exam	Written Exam
Learning outcomes	- See games through the eyes of a professional game developer. This includes the ability to draw valid conclusions about the development process, the business model, cultural influences, and the underlying technology. - Ability to analyze video games from different points of view (e.g. programmer, designer, artist, publisher, politician…) - Understanding of the hard- and software used to develop video games and how they impact the development process. This includes implications on the runtime performance as well as financial and budget considerations. - Ability to reflect and scientifically discuss (within small groups) various game development-related subjects such as design, business, youth protection, hardware, and software. Presentation and explanation of the findings before the class. - Understanding of the various career options in the games industry. - Acquiring fundamental knowledge for advanced courses.
Abstract	The lecture gives a comprehensive overview about game development and does not require a lot of prior knowledge. The students get the opportunity to give one 10-15 minute long presentation over the course of the semester. Doing so will be rewarded with a few bonus points for the written examination. The exact dates and topics of these presentations will be assigned during the lectures.
Contents/ Indicative syllabus	- Genres and History - Business Aspects - Youth Protection - Hardware - Game Engines - Game Objects - Human Interface Devices - Virtual Reality - Multiplayer - Artificial Intelligence Please note: Detailed syllabi are not a standard in German universities; students should expect to be informed of assignments verbally and/or via an online learning platform, i.e. Moodle.
Reading Materials	- Jeff Lander, Jason Gregory, Game Engine Architecture, Second Edition, Taylor & Francis Ltd., 2014, ISBN 978-1466560017

	- Robert Nystrom, Game Programming Patterns, Genever Benning, 2014, ISBN 978-0990582908 - Ron White, How Computers Work, 10th Edition, Que, 2014, ISBN 978-0789749840 - Mike McShaffry, David Graham, Game Coding Complete, Course Technology, 2012, ISBN 978-1133776574 - Ian Millington, John Funge, Artificial Intelligence for Games, Morgan Kaufmann, 2009, ISBN 978-0123747310 - Jesse Shell, The Art of Game Design, Morgan Kaufmann, 2008, ISBN 978-0123694966

Course Title	Practical Course in Game Development
	Bachelor Level Course (Typically taken during 2nd or 3rd year)
Course No	671131
Lecturers name	Prof. Dr. Stefan Radicke
Teaching language	English
Credits (ECTS)	8
Teaching/learning methodology	Practical project
Total workload	240 hours
Contact hours per week	As discussed. Team meetings and meetings with the professor on a regular basis
Type of exam	Group Project and Presentation
Learning outcomes	- Practical use of theoretical knowledge in context of a large game project. - Teamwork and communication skills within a large-scale project team of over 30 students. - Structured and independent work capabilities. - Strategic planning, reflection and results evaluation. - Experienced students also get the opportunity to take leadership roles. This includes making important project decisions, managing and assigning tasks, and some supervisory functions.
Abstract	This course gives all participating students the opportunity to work together on a video game production in a large-scale project team under realistic circumstances. There will be two must-attend kickoff-events.
Contents/ Indicative syllabus	Please note: Detailed syllabi are not a standard in German universities; students should expect to be informed of assignments verbally and/or via an online learning platform, i.e. Moodle.

Course Title	**Game Engine Programming**
	Bachelor Level Course (Typically taken during 2nd or 3rd year)
Course No	113521
Lecturers name	Prof. Dr. Stefan Radicke
Teaching language	English
Credits (ECTS)	6
Teaching/learning methodology	Lecture and practical work
Total workload	180 hours
Contact hours per week	45 hours teaching time Meetings as necessary
Type of exam	Practical work + oral exam
Learning outcomes	- Students gain the ability to design effective and reusable software architectures for high-performance environments. This includes the utilization of common, as well as game-specific software design patterns, and interface abstraction techniques to build a flexible, customizable and reusable game engine from the ground up. - Detailed knowledge of game engine subsystems and their inter-dependencies are attained. Participants acquire skills to plan and implement reference counting objects, container classes, memory management utilities, mathematical structures, accurate timing routines, resource loading and streaming systems, multi-threaded algorithms, and the integration of a 3D environmental sound framework. - Students gain skills to analyze, evaluate, and assess the time-, performance-, and memory-characteristics of high-performance real-time software systems. This includes deep knowledge of several well-known algorithms, their specific use-cases and respective optimization techniques. In this context, a deep understanding and appreciation of the performance characteristics of modern multi-core architectures is earned. Students can write and optimize algorithms for improved cache-efficiency, memory utilization, task concurrency, and data parallelism. - Students acquire the ability to competently and confidently develop large, interdependent software systems utilizing a test-driven approach. They can both realize subsystem functionality based on given unit-test scenarios as well as plan and execute their own automated test cases.
Abstract	Students learn how to implement a flexible and reusable game engine from scratch using C++14 and the Windows 10 SDK. The architectural aspects, as well as the theory behind the most important core engine systems, are covered in great detail. Special emphasis is put on engine support systems, memory management, time measurement, maths, resource management and multi-core architectures. Another major aspect of the course is how common software design patterns like singleton, factory, facade, iterator, extractor, and others can be utilized in practical software development, to ensure both flexibility and high run-time performance. The students also get the chance to apply their knowledge in numerous practical exercises. Each implemented component will be evaluated through unit tests to ensure proper functionality. If executed well, the individual exercise modules can be combined to build a complete game engine.
Contents/ Indicative syllabus	Basics: - C++ for Java Programmers - Engine Framework - Reference Counting - Container Classes - Core Systems - Memory Management - Memory Tracking - Game Loop - Math for Games

	- In-game Camera - Resource Management Advanced Topics: - Multithreading - Multithreaded Jobs - Audio Programming using FMOD Please note: Detailed syllabi are not a standard in German universities; students should expect to be informed of assignments verbally and/or via an online learning platform, i.e. Moodle.
Reading Materials	- Mike McShaffry, David Graham, Game Coding Complete, Course Technology, 2012, ISBN 978-1133776574 - Jeff Lander, Jason Gregory, Game Engine Architecture, Second Edition, Taylor & Francis Ltd., 2014, ISBN 978-1466560017 - Robert Nystrom, Game Programming Patterns, Genever Benning, 2014, ISBN 978-0990582908 - Andrei Alexandrescu, Modern C++ Design, Generic Programming and Design Patterns Applied, Addison-Wesley Longman, Amsterdam, 2001, ISBN 978-0201704310 - Erich Gamma, Richard Helm, Ralph E. Johnson, Design Patterns. Elements of Reusable Object-Oriented Software, Addison-Wesley Longman, Amsterdam, 1994, ISBN 978-0201633610

Course Title	**Gameplay Programming**
	Bachelor Level Course (Typically taken during 2nd or 3rd year)
Course No	113522
Lecturers name	Prof. Dr. Stefan Radicke
Teaching language	English
Credits (ECTS)	6
Teaching/learning methodology	Lecture and practical work
Total workload	180 hours
Contact hours per week	45 hours teaching time Meetings as necessary
Type of exam	Practical work
Learning outcomes	- Ability to pick up any modern game engine quickly and easily. - Analysing, evaluating, and understanding mechanics of various games. - Building fun mechanics utilizing solid and robust gameplay systems. - Realising communication and meaningful interaction between systems.
Abstract	Students learn how to program core gameplay mechanics using the Unreal Engine 4 (UE4). These include movement, animation, interaction, decision-making and combat. For the examination, students are required to pitch, plan, implement, playtest, and re-iterate a gameplay prototype.
Contents/ Indicative syllabus	- UE4 Gameplay Framework - UE4 Blueprints Visual Scripting - UE4 C++ Programming - Collision and Physics - Character Movement - Camera Systems - Game States - Character Animation - Performance Optimization - Randomness - Artificial Intelligence - Multiplayer Please note: Detailed syllabi are not a standard in German universities; students should expect to be informed of assignments verbally and/or via an online learning platform, i.e. Moodle.
Reading Materials	- Jesse Schell, The Art of Game Design, 2nd revised edition, Taylor & Francis Ltd, 2014, ISBN 978-1466598645 - Robert Nystrom, Game Programming Patterns, Genever Benning, 2014, ISBN 978-0990582908 - Mike McShaffry, David Graham, Game Coding Complete, Course Technology, 2012, ISBN 978-1133776574 - Ian Millington, John Funge, Artificial Intelligence for Games, Morgan Kaufmann, 2009, ISBN 978-0123747310

Course Title	**Computer Graphics**
	Bachelor Level Course (Typically taken during 2nd or 3rd year)
Course No	671132
Lecturers name	Prof. Dr. Jens-Uwe Hahn
Teaching language	English
Credits (ECTS)	6
Teaching/learning methodology	Lecture and practical exercises
Total workload	180 hours
Contact hours per week	22.5 hours teaching time 22.5 hours guided practical exercises
Type of exam	Written Exam
Learning outcomes	- Basic knowledge of the classical techniques of generative computer graphics - The ability to translate acquired theoretical knowledge into practical applications. - Basic knowledge in modelling - Basic knowledge in graphics programming
Abstract	This course provides a fundamental introduction to 3D computer graphics in theory and praxis.
Contents/ Indicative syllabus	- Fundamentals in Modelling - Object Representations - Fundamentals of Animation - Projektive Spaces und Homogenous Coordinates - Transformations - Locale Illumination Models - Shading Algorithms - Visibility Calculation - Texturing - Global Illumination: Ratracing und Radiosity - Modelling and Animation in Blender - Graphics Programming in OpenGL - The theoretical knowledge is applied in practical exercises. Please note: Detailed syllabi are not a standard in German universities; students should expect to be informed of assignments verbally and/or via an online learning platform, i.e. Moodle.
Reading Materials	- Alan Watt, Fabio Policarpo: 3D Games - Animation and Advanced Real-time Rendering, Addison-Wesley - Andrew Glassner: Principles of Digital Image Synthesis, Morgan Kaufmann

HOCHSCHULE
DER MEDIEN

Course Title	**Game Physics**
	Bachelor Level Course (Typically taken during 2nd or 3rd year)
Course No	113540
Lecturers name	Prof. Dr. Roland Schmitz
Teaching language	English
Credits (ECTS)	6
Teaching/learning methodology	Lectures and practical work
Total workload	180 hours
Contact hours per week	45 hours teaching time Meetings as necessary
Type of exam	Written exam
Learning outcomes	After completing this module, students will know the essential theoretical foundations and principles on which modern Game Physics Engines are based. In particular, they can: - Set up the equations of motion for moving game objects - Solve the equations of motion approximately - Recognize and treat collisions - Critically evaluate and analyze the performance of existing game physics engines - Communicate and realize own ideas for physics-based games
Abstract	Game Physics is the art (and science) of realistically simulating the physical world inside a game. For about a decade now, physics engines have become an important part of almost any modern game. They range from realistic environments like dust and debris flying around, to physics puzzles and interactive game worlds that became key aspects for whole genres. The lecture consists of a theoretical and a practical part. In the theoretical part, we will look at the basic equations of 2D and 3D rigid body dynamics and how they are solved within a physics engine. In the practice part (given by Andreas Stiegler), students will learn to build their own small 2D or 3D physics engine. Since you cannot do any interesting physics without mathematics, we will also cover some advanced mathematics that goes beyond what you have learnt so far. Yet, we will connect the math formulas and concepts with their applications in a real physics engine, which makes it a lot easier to get a grasp of the concepts.
Contents/ Indicative syllabus	Theoretical Part: - Newton's Laws - 2D Rigid Body Dynamics - 3D Rigid Body Dynamics - Collision Response - We will develop the necessary mathematics (especially vector analysis) along our way. Practical Part: - The Game Loop - Physics Engines - Collision Detection - Building your own (2D or 3D) Physics Engine Please note: Detailed syllabi are not a standard in German universities; students should expect to be informed of assignments verbally and/or via an online learning platform, i.e. Moodle.
Reading Materials	- David Eberly, Game Physics, Morgan Kaufmann 2010 - Ian Millington, Game Physics Engine Development, Morgan Kaufmann 2010