CoSMoS in the Interactive Simulation Curriculum

Sebastian von Mammen, Sarah Edenhofer, and Jörg Hähner

University of Augsburg, Augsburg, Germany

Abstract. Animations at first, then real-time computer graphics and human-computer interaction techniques have made interactive simulations possible. Nowadays, they play an enormously important role in training the operation of complex technology such as aircraft, and they have achieved a remarkable share in the computer gaming industry. The fast emergence of virtual and augmented reality solutions promises an even wider spread and a greater impact for interactive simulations in the near future. Due to the multifaceted nature of interactive simulations in terms of confluent scientific fields, due to the underlying iterative and agile development processes, and last but not least due to the inherently central human factors, we have been integrating the CoSMoS process of complex system modelling and simulation into our course curriculum on interactive simulation for computer science graduate students. In this work, based on an overview of the contents and the logistics of the course, we present our conceptual efforts towards this goal. We emphasise the role of the CoSMoS process, discuss its impact on the students' projects, and we provide concrete examples.

1 Introduction

Since the first human-in-the-loop simulators entered the market in the 1980s [46], interactivity has evolved into an increasingly important aspect of scientific simulations. Nowadays, established mathematics frameworks such as Mathematica, Maple or Matlab provide ample support for visualisation routines and interactive parametric exploration of any devised models, whereas development frameworks such as Unity3D, Unreal Engine or CryEngine that primarily target the computer gaming market are offered and marketed in the context of simulations as well. Due to the fast-paced strides towards ubiquity of virtual and augmented reality systems [61], for instance by utilising widely availably smart phones, we expect an even more accelerated spread of interactive simulations in the near future.

Numerous areas of computer science feed into the development of an interactive simulation—human-computer interaction, real-time computer graphics, visualisation, modelling and simulation approaches, etc. The according methodologies and techniques are deployed to make a simulation model accessible to the user. In addition to translating reality into an adequate domain model and further into a suitable computational representation, or platform model, the

creators of an interactive simulations are confronted by an abundance of user-related interfacing challenges. In all brevity, they need to translate the user's wishes into effective commands of control and model changes, and they need to translate the matter-of-fact results of the simulation process into visualisations (mostly), that are quickly understood and capture rather than lose the user's attention. To render the trade even more challenging, all of these translations need to happen at rather high rates that provide for an uninterrupted interaction experience.

Motivated by their great and growing importance, we set out to teaching students foundational knowledge about interactive simulations. In particular, we designed a university course to empower computer science graduate students with an interwoven in-depth apprehension of methods in the associated fields. Thus, the students acquire knowledge to evaluate and skills to contribute to the design and the programmatic implementation of interactive simulations. In this work, we present our course concept, focusing on the role of the CoSMoS process of complex system modelling and simulation. Based on a description of our course concept (Section 2), we highlight the role of the CoSMoS process in the curriculum as a whole, and with respect to the accompanying student projects, in particular, in Section 3. Next, we present several select student projects (Section 4), also shedding light on the development processes the student went through throughout the term. We conclude this work with a summary of our findings and an outlook on future work on CoSMoS for interactive simulations.

2 Synopsis of an Interactive Simulation Course

An interplay of a variety of computer science disciplines provides the foundation for interactive simulation. Accordingly, the contents of a course on interactive simulation greatly vary dependent on the expected knowledge base of the students as well as complementary courses offered by the hosting institution. In our case, we devised a university course suitable for master students in computer science and closely related programmes of study. The course runs for four months, staging a 2-hours-lecture and a 2-hours-tutorial each week. In combination with the allotted project work, the course demands for a total workload of 150 hours.

In the following paragraphs, we summarise the contents of nine provided lecture units. After an introduction to the subject matter, we teach the CoSMoS approach to modelling and simulation. Next, foundations of computer graphics are conveyed, as well as a mathematical display of real-time physics computation models and algorithms. Visualisation methods and an introduction to human-computer interaction techniques complete the first block of basic lecture units.

The second block of advanced lecture units focusses on model representation and process optimisation both of which are important constituents of interactive simulation technology. After presenting the foundations of discrete event simulation and an array of computational representations, popular conservative and approximative acceleration mechanisms in the realm of interactive applications are discussed. As the versatility and the transferability of an agent-based

modelling (ABM) approach is rather unique but can easily result in costly computations, we commit another lecture unit to introducing novel research concepts that promise to scale ABM to reach interactive performances.

2.1 A Short History of Human-in-the-Loop Systems

The history of interactive simulation begins with efforts to enhance existing simulation data by means of interactive custom animations. We present an according example, a SIMAN job shop simulation model of an automatic guided vehicle system visualised by the CINEMA animation system [49]. The optimisation of industrial workflows was the most compelling argument for such animation systems in the 1980s. At the time, the market offered an array of simulation animation tools, including Model Master, XCell, and Performance Analysis Workstation. Next, solutions were offered that tightly coupled interactive visualisation with the underlying simulation. See-Why was one of these packages that promised Visual Interaction Simulation (VIS). An according example allowing the configuration of a locomotive servicing centre is shown [47]. Definitions of basic terms such as model [67], simulation [4], and the early-conceived notion of interactive simulation ('on-line simulation') [31] follow the introductory historic examples. The distinctive feature of interactive simulations is the possibility of human influence during the simulation process, typically referred to Human-in-the-Loop systems [54]. We look at the taxonomy of interactive simulations, their advantages over stand-alone simulations, established fields of application, technological challenges, and their historic evolution in respect to programming paradigms, languages, and interfaces. The basic steps taken in a simulation project, especially under consideration of interactivity, and several examples of state-of-the-art interactive simulation systems round off this lecture unit. The examples are organised to guide the students from comprehensive immersive solutions with special hardware configurations (driving and flight simulators) to software-only solutions, which are the focus of the lecture.

2.2 The CoSMoS Process & Gamification

The orthogonal relationship between descriptive and defining models precedes the remaining contents of this lecture that primarily aims at the process of modelling and simulating complex systems. Examples for seemingly disparate approaches are provided that put different weights on these respective modelling purposes: In detail, these are understanding complex behaviours of real-world systems, simulating complex system themes, engineering complex algorithms, and engineering complex systems [51]. We consider the means of scientific instrumentation (extrapolation, conversion, augmentation) to become aware of its limits and limitations [25] and to define the products of the CoSMoS modelling and simulation cycle. The definition of these products motivates an elaborate discussion of the phases of discovery, development and exploration [2]. Interactive simulations need to engage their users. The relatively novel paradigm of

turning burdensome chores into games suits this challenge well. Hence, we proceed with the presentation of game definitions and, more specifically, aspects of development of computer games [42]. The short history of serious games (starting in the early 2000s) is summarised [21, 59] and representative examples are demonstrated (e.g. [9]). Their concrete successes in terms of engagement are discussed and a comprehensive list of game design elements [11] is presented that can be utilised to 'gamify' an interactive simulation. All of these elements can be derived from the cornerstones of intrinsic motivation, namely relatedness, competence, and autonomy which are explained as well [33]. In the context of interactive simulations, these aspects can be considered during the discovery phase, whereas gamification typically takes place during the development phase of the CoSMoS process.

2.3 Computer Graphics Foundations

The increasing availability of dedicated graphic processing units (GPUs) promotes the utilisation of a standardised 3D rendering pipeline for any kind of visualisation needs, whether 2D or 3D, vector-based, or otherwise. Therefore, this lecture units seeks to empower the students with a basic understanding of this rendering pipeline [1, 42, 63]. Basic concepts that are presented in this lecture unit are: object definitions based on geometric primitives and various kinds of textures, the view reference, spatial transformations (also introducing quaternions in the context of rotational operations), basic types of lighting, shading, and light sources, shadow definitions and different implementation techniques such as shadow maps. A short walkthrough of generating 3D graphic assets suited for real-time rendering rounds off this lecture unit.

2.4 Real-time Physics

This lecture seeks to provide a solid grip on real-time capable approaches to simulate physical processes [8, 24]. We have a quick look at the taxonomy of the vast field of physics simulation [14] but we focus on real-time methods of forward dynamics, covering three categories: rigid-body dynamics [7, 15], soft-body dynamics [13], and particle physics [38]. Next to the general laws of motion, we look at non-penetration constraints, collision resolution and friction forces, and complementary constraints in the context of rigid body simulation. For calculating the respective forces, we present the penalty force method, Lagrange multipliers, impulse-based simulation, and reduced coordinate formulation as well as the Coulomb friction model. We introduce a taxonomy of constraints and explain how they can serve as representation of mechanical joints of articulated bodies [6]. We follow the steps to transform the resulting differential algebraic inequalities into an efficiently solvable linear complementary problem. We conclude the integration of forces with a brief recap of basic methods of numeric integration, starting with Euler and Runge-Kutta. We discuss algorithms for efficient collision detection and contact point generation, e.g. [37, 18]. We then widen the

scope of this lecture unit, looking at one specific approach to computing incompressible deformable mesh dynamics that is superior to alternative approaches in terms of efficiency and accuracy [13]. Finally, we introduce to real-time particle physics, explaining particle approximation functions based on the notion of kernel functions [38], culminating in recent advancements in unified real-time physics simulation [39].

2.5 Visualisation Methods

To a large extent, interactive simulations imply some kind of visualisation of the underlying models and the emerging simulation processes. In this lecture unit, we emphasise the necessity to consider human perception and information processing when crafting the platform model of an interactive simulation and we provide an overview of foundational visualisation techniques. We follow the structure of numerous textbooks on this subject matter and motivate the discussion on the human vision apparatus by providing several examples of optical illusions [62]. In a 7-step guideline, we establish an idea of the selection of the proper visualisation method embedded in the context of data acquisition and the intended modes of interactions [17]. Qualitatively, visuals can be measured in terms of novelty, informativeness, efficiency and aesthetics [58]. We shed light on various scientific, multidimensional, multivariate visualisation methods [65], before we turn to visualisation techniques that allow for the immersive augmentation of simulation contents, such as examples of flow visualisation [43], graph-based visualisations [34], or the transformation of volumetric (4D) data into 3D surfaces [30].

2.6 Human-Computer Interaction Techniques

The design of interactive simulations necessitates an interface between human and computer. This lecture unit provides the necessary background, starting with a brief history of HCI research [44]. Human interaction requires the user to process and translate sensed information into motor activity [12]. In general, user interactions can be classified as operations of selection, manipulation, navigation, and system control [42]. As any other design task, the design of interactions is the result of a tradeoff between multiple goals and constraints [16]. We present a top-down approach to designing interaction scenarios that starts with the definition of an application's requirements and arrives at individual interaction tasks. We have a brief look at multimodal approaches, e.g. affective, perpetual, attentive, and enactive interfaces [29], and we explore the modes of interaction of an embedded multimodal prototype game [19]. We quickly step through established and emerging immersive hardware technologies, including devices of motion sensing and object tracking capabilities. We convey a general understanding for the hard latency limitations of interaction hardware and we provide recipes for rather general issues that arise in real-world sampling, i.e. noisy sensing and the state estimation problem [63, 42].

2.7 Discrete-Event Simulation

In this lecture unit we provide an overview of computational representations as well as modelling and simulation approaches. We start out with explaining the basic terminology of discrete-even stimulation (DES) in the context of previous lecture units, especially those described in Sections 2.1 and 2.4 [3]. To this end, hybrid simulation and combined simulation concepts are of great importance. We roughly trace the history of this seminal field in terms of DES software packages and languages [45]. Three 'world views' on DES (event scheduling, activity scanning, and process interaction) serve as the starting point for our venture into historic approaches. We meticulously describe the elements of a DES and provide a glimpse at charts already used for engineering DES back in the 1960s. These diagrams (activity cycles, wheel chart, flow charts) are our point of departure towards other computational representations commonly used for modelling and simulation: Finite state machines, UML transition diagrams, Petri nets [50], artificial chemistries [5], cellular potts [28], cellular automata [68], random boolean networks [32], boids [53], L-Systems [52], swarm grammars [40], and the general approach of agent-based modelling [10, 66].

2.8 Acceleration Algorithms

Low latency requirements of the interaction interface (Section 2.6) as well as the desire to serve large, complex models for interactive exploration at real-time demand for the utilisation of sophisticated acceleration algorithms. Computer graphics and real-time physics are currently occupying this niche and this lecture unit aims at exhibiting their commonly used, highly efficient approaches [1].

It is divided into four parts: First, we focus on bounded volume hierarchies (BVHs) and binary space partitioning trees (BSPs). While BVHs are built bottom-up based on bounding volumes that enclose geometries or other bounding volumes recursively, BSPs are generated top-down by recursive division of the simulation space. We also provide guidelines to coping with (a) mobile and (b) deformable objects [35, 36, 60]. Second, we explain different culling techniques which ensure that graphical objects are not pushed through the rendering pipeline (Section 2.3), if their contributions to the final rendering are marginal or not existing. Examples are surfaces that lie outside of the view frustum, those that are hidden behind objects, or those that are so far away from the camera that they could hardly be seen on the screen. Third, we present approaches that lower the level of detail (LOD) of the rendered objects to the match the actual needs—as opposed to always rendering at the highest possible level of detail [23]. An example of LOD is the number of triangles of discrete geometries which can, for instance, be selected according to the distance to the camera. We conclude this lecture unit showing stochastic acceleration algorithms for collision detection.

2.9 Dynamic Model Abstraction

Motivated by the outlook on large, multi-scale, multi-representation simulations [26], we tackle the issue of ever-growing computational complexity by means of dynamic model abstraction techniques in this lecture unit. Particularly, we focus on adaptive optimisation of agent-based models, as they can serve as a generic computational representation. Concerning the immense computational costs running large-scale simulations, we discuss the limitation of different model aspects and how they could improve efficiency. We conclude this investigation with the realisation that if we want to model and compute natural systems, we need to consider dynamic systems with dynamic interaction topologies (DS^2) [20]. In addition to hardware-based solutions (e.g. [27]), we promote dynamic model adaptation. Agent compression identifies and subsumes clusters of similar agents [57]. The dynamic extension of this approach considers container agents to maintain similar agents and to offer the possibility to remove or add individuals on demand. Compression managers are responsible for (a) organising the container agents and their contents, and (b) representing the compressed agents to the remainder of the model [64]. Taking this idea even one step further, we provide the detailed steps of the self-organised middle-out abstraction approach [41] and we show its capabilities with respect to a decentralised, agent-based blood-coagulation simulation [55, 56].

3 CoSMoS' Central Role

The CoSMoS process is introduced right after a general introduction to the course (see Section 2.2), as it provides a flexible, yet focussed guideline for all phases of the development of interactive simulations. In this section, we first detail a way of applying the CoSMoS process to student projects, following the explanations in [2]. Second, we present a course infrastructure to realise this approach.

3.1 CoSMoS for Interactive Simulation

We discuss the three phases of the CoSMoS cycle (discovery, development, exploration) in the context of interactive simulation based on the five activities performed during each phase: scoping, modelling, experimenting, documenting, and interacting.

During the *discovery phase*, the greatest challenge to the students is the primary need to settle on an application domain and to define the goals of the interactive simulation, e.g. teaching contents or providing for a scientific exploration tool. Although the students appreciate the opportunity to freely chose an application domain for their projects, they seem to be more comfortable when provided with a theme, for instance biology. The only constraints regarding their choice is the projects' evaluation based on the following aspects, which are set to ensure their usefulness and the comparability.

Science The model that drives the resultant prototype has to be scientific, i.e. it has to be based on scientifically published results. A CoSMoS compliant development process certainly supports this endeavour. In addition, the modelling domain, the validity of the modelled system, its degree of innovation, and the computational representation and algorithms used give strong indications for a scientific approach.

Gamification The prototype has to motivate the user to interact and explore the simulation space. One can try capturing this aspect quantitatively by describing interaction possibilities, user guidance, usage of game elements, and the factors of intrinsic motivation as referenced in Section 2.2.

Complexity Interacting with the prototype should be rewarding in itself, i.e. it should convey insights with respect to the underlying scientific model. The model complexity defines the scope of potentially educational contents, given the conveyed complexity considers the full extent of the underlying model.

Aesthetics An interactive simulation has to be aesthetic, not only to efficiently convey information to the user but also to motivate their involvement. Aesthetics can be promoted following established design principles, by utilising beautiful visual assets, and by combining them in novel ways.

Any steps towards desirable domain attributes, concrete domains, and even concrete goals and an application concept, necessitate answering questions about the projects' criticality, their limitations, and their measurable success. In the context of interactive simulations, the answers typically stress the relationship between the software and the user. The utility for the user, for instance, not only considers a final simulation result but also the benefit of pro-active participation in the simulation process. Accordingly, limitations are not only considered regarding the accuracy and efficiency of the simulation but especially with respect to the degrees of freedom exploitable by the user and the quality of the communication between the user and the simulation, including aspects such as clarity and attractiveness. The modelling activity during the discovery phase is rather limited in the scope of a term-long project. Despite the abundance of scientific data accessible through online libraries and the large repositories of computational libraries and tools for numerous scientific domains, comprehending the elements and their relationships of a previously unstudied field is a rather difficult task. For this reason, and also to provide the necessary degree of autonomy to intrinsically motivate the students, we allow the students to decide on a concrete domain and goal by themselves; based on supervisory feedback on a written proposal and classroom presentations with subsequent discussions, the core ideas can then be quickly translated into first proof-of-concept prototypes. The discovery phase is decisively shaped by documentation activity—from coarse to fine grained searches for references and tools, through merging sources, assumptions and ideas into a concept proposal that includes an early domain model, to creating a first prototype that provides evidence for the created line of argument.

During the *development phase*, documentation about the students' activities is similarly important. However, to a great extent, it coincides with the devel-

opment of the platform model, an accompanying commented code base, and its transcription for a given simulation platform. To help reduce the burden that a comprehensive interactive simulation project incurs, we diminished the scoping activity of the development phase and taught about various tools of the trade for interactive simulation development—ranging from 3D asset creation over scripting and high-level, component based model compositions to utilising third-party plugins and libraries for the targeted development environments. In frequent presentation and feedback sessions, we ensured that domain elements were properly represented and domain behaviours were not directly encoded in the models. Adding instrumentation to the platform model plays an important role for interactive simulations. This step should closely follow the interaction concept developed as an extension of the usual domain model, i.e. one that encompasses the user as a special model element. Nevertheless, the targeted simulation platform may provide a rather special interaction infrastructure. For example, the ubiquity of mobile, multi-touch platforms equipped with relatively weak processing capabilities competes with the processing power, storage capacity, and extensibility of desktop systems. Clearly, any specific interaction platform demands for individual adjustments of the platform model to realise both the interaction and the simulation concept. Experimentation in the development phase begins with the first prototype supporting preliminary user interactions. At later stages of the development phase, it increasingly involves feedback from testers not directly involved in the development work. Beyond honing the visualisation, consistent design, usability and the scalability of their platform models in terms of parameter settings, numbers of interacting agents, increasing levels of difficulty and the fine line between balanced, rewarding interaction and user boredom and frustration.

During the final stage of the course project, the *experimentation phase*, the students focus on logging and analysing user responses to their simulations. To keep the amount of work at a level reasonable in the context of our course, the students are asked to try each others' simulations and to ask their friends and relatives to provide them with some preliminary feedback. This exposure typically already provides comprehensive insights into the users' general interest in the topic, their opinion about model complexity and aesthetics, and whether they think it is educational. Based on these evaluations, the students are encouraged to hone their software and to launch more comprehensive online surveys. However, these more rigorous steps are not mandatory course stipulations. Nevertheless, the gathered preliminary data in combination with the initial motivation of their projects, the development processes and the implementation results, serves as an extensive basis for fleshing out their final report. It culminates in conceptual improvement that could instigate the next development cycle.

3.2 Course Project Infrastructure

Above, we already touched upon the students' deliverables and how their realisation is backed by the CoSMoS process. Now, we briefly present the logistical

infrastructure of the course setup to support the traversal of the CoSMoS process throughout the term.

During the first lecture, the students are first informed about the course contents and its stipulations. For the remainder of the lecture, we present and explain several examples of possible project concepts. Although the students may conceive a project idea completely on their own, providing examples proved important to communicate the expected scope and the imparted opportunities. Within ten days' time, teams of two students need to author a proposal of their projects. On two pages (ACM double-column format), the students need to motivate, present and detail their concepts. Hereby, the envisioned user experience plays an important role as it ties different aspects of the envisioned simulation together and it implicitly underlines its goal. From a CoSMoS perspective, the project proposal is part of the documentation activity of the discovery phase. As such it serves not only as a platform for the students to substantiate their initial ideas and consistently brush up their findings but also to communicate their concept to the instructors.

At the time of the proposal submission, a second lecture unit has introduced the general topic of the course (Section 2.1) and a first tutorial session has familiarised the students with the development environment that we recommend (in previous years, we recommended Unity3D). The day after the submission of the proposals, the students are asked to present their concepts in short 3-minute presentations during the tutorial session. In this way, all the students in the course would gain an overview of their peers' projects and learn about new ideas, possibly even about the usage of previously unknown code snippets, etc. The quick start into the projects and presentations early in the term help the students build up momentum for their projects. In fact, until the last few weeks of the term, the students would present the state of their projects bi-weekly. This fosters a certain sense of togetherness and it ensures guidance to maintain high productivity and to avoid frustration.

Two weeks before the end of the term, final reports are due (six pages, ACM double-column format) that should ideally condense the documentation recorded throughout the whole term. One week later, the students need to submit their projects, including batches of slides for the final presentations which are given in front of faculty and students of the whole department. The audience is asked to vote for the best entry in terms of the generic project criteria: science, complexity, gamification, and aesthetics (Section 3.1). A 15-minute brief oral exam at the end of the term makes sure that the students have learned and understood the diverse contents of the course and their relationships.

4 Select Student Projects

In this section, we present select student projects that were developed in two iterations of our interactive simulation course. First, we describe some of the outcomes exemplarily. Second, we shed light on the CoSMoS-driven development process of a specific project.

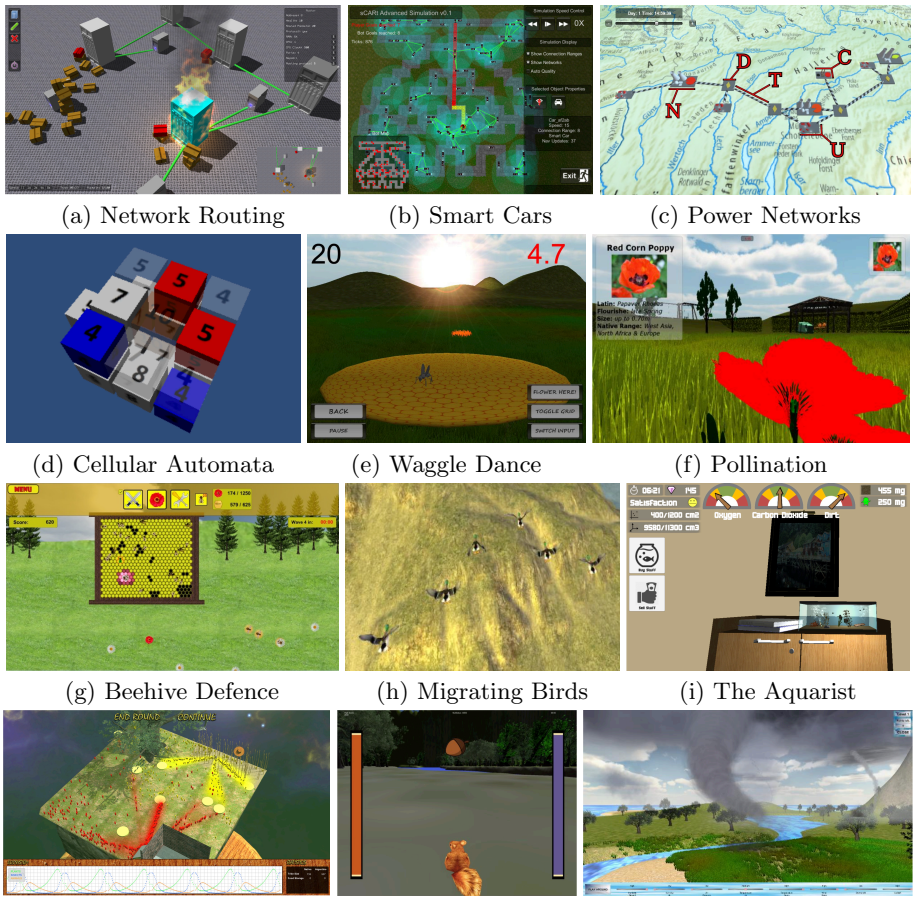
4.1 Examples

During the first iteration of the course, the majority of the students chose "technical systems" topics such as routing in communication networks, smart cars, and power networks. Figure 1(a)-(c) shows according screenshots. The user is tasked to build and maintain power or communication infrastructures to ensure their proper functionality. In the network routing and the smart car example, the user also had to guide the network activity itself by laying out flow paths of the respective traffic. Some students also journeyed towards biological themes such as cellular automata as seen in Figure 1(d). Here, a game of life variant served as the basis for a two-person game with the goal of conquering as much space as possible solely by adjusting the cells' rules. During the second iteration of the course, we proactively advertised biological and natural phenomena as an exciting and multifaceted field to motivate the student projects—yet, they were still free to take their projects into other directions. As a result, three groups let their projects revolve around bees (we had not motivated this trend), see Figure 1(e)-(g). In the first one, the user had to guide a bee's waggle dance to point its peers to the location of a food source outside the hive. Figure 1(f) shows a screenshot of a bee simulation that focusses on the challenge of gathering nectar and thereby helping flowers pollinate. Lastly, a complex real-time strategy simulation is presented in which bees need to gather resources, maintain their hive and defend it against wasp intruders. Other examples included the user-guided migration of a flock of geese (Figure 1(h)) or the establishment of a fine balance of interdependent inhabitants in a simulated aquarium (Figure 1(i)). The interdependency of species provided the basis of yet another title where a new ant species threatens to overrun a native species and the user is tasked to maintain a balance by building barriers or proactively diminishing one or the other ant population (Figure 1(j)). Focussing on solitary species, a squirrel simulator offered the experience of sharing a rodent's worries: collecting, burying, and finding enough nuts to survive the winter season (Figure 1(k)). The importance of climate also inspired "Cloud Computing", where a user was tasked to set the environmental conditions in such a way that certain weather phenomena such as rain or tornados would emerge (Figure 1(l)).

The set of presented examples emphasises the flexibility of the course project in terms of contents, perspectives and goals of the student project while addressing the project requirements as outlined above (Section 3.1). Next, we dive into one specific project and shed light on how the CoSMoS process informed its development.

4.2 A CoSMic Case Study: "Drink & Drive"

One student team decided on creating a serious game about the negative effects of alcohol on traffic participants. They understood that although some accurate simulators exist for this purpose (e.g. [22]), they don't provide for a stimulating, engaging experience. At first the students were hesitant whether their idea was acceptable as it attempted to approach a serious topic in an engaging, fun way.



(a) Network Routing (b) Smart Cars (c) Power Networks

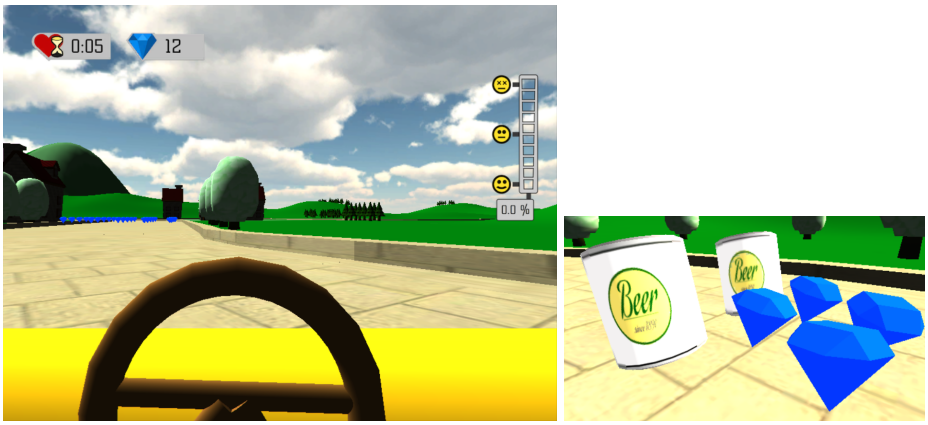
(d) Cellular Automata (e) Waggle Dance (f) Pollination

(g) Beehive Defence (h) Migrating Birds (i) The Aquarist

(j) Invasive Species (k) Squirrel Simulator (l) “Cloud Computing”

Fig. 1. Screenshots of interactive simulations developed as student projects in two iterations of the course.

We encouraged them to try anyway. Findings about games that had been developed for this purpose, such as [48], further boosted the students' ambitions. These preceding titles had disconnected from the actual problem too much, for instance by assuming a third-person perspective on the driving situation. Quickly, the students realised that their interactive simulation should fill this gap and make their title "Drink & Drive" both fun and educational, so that the target group of soon-to-be drivers and young drivers would engage in and learn about this fundamentally serious topic. The second part of the discovery phase of their project shed light on actual models of impairment of drunk drivers. Its last part posed the greatest challenge: Merging the seemingly conflicting concepts of learning about the severe consequences of drunk driving on the one hand, and the need for user engagement on the other hand. They achieved this by two means. First, they decided to represent the game itself at a level of abstraction different from the effects of alcohol. In particular, the game implemented widely-known "Mario Kart"-style game mechanics and a simple, cartoonish look (Figure 2(a)), whereas the impairment of alcohol was reflected by realistic effects, including the deterioration of clear-sightedness, darkening the edges of the vision, attenuating sounds, and prolonged reaction times (realised by increased simulation speed), see Figure 3. Second, they introduced gamification elements including timed laps and collecting high scores by picking up precious diamonds from the track (Figure 2(b)). However, fundamental game mechanic to engage the users was invented later during the experimentation activity of the development phase. The students laid out the development phase very professionally and, together with the other students, received bi-weekly feedback to stay on track. Knowing that experiments could yield the key to an engaging user experience, the students tested various parameter settings of the driving model, its reactivity to the user input, as well as different interaction modes between the steered vehicle and the environment. From what they learned they were able to invent a mechanism to ensure a challenging and well-directed user experience. In particular, they translated the idea of collectibles on the track to their application domain and positioned beer cans at certain locations (Figure 2(b)). Their uptake would increase the blood alcohol level and driving would be impaired. The impairments would render it difficult to complete a track within a certain amount of time. Given the mechanics of driving, impaired driving, high-scores and time-laps, the students just needed to find the right balance to finish the development phase of their simulation. "Drink & Drive" was voted best entry in the public presentations at the end of last term's interactive simulation course. In addition, it stirred a lot of excitement when it was offered for play as part of the Girls' and Boys' Day at our university. Based on these successes, the students feel that the most fundamental aspect that could drive a second development cycle would be the port of "Drink & Drive" to mobile devices for reaching a greater audience.



(a) (b)

Fig. 2. (a) A first-person default view is reduced to a simple steering wheel dashboard and a few icons that represent the time left to complete the track (the heart icon in the upper-left corner), the achieved score (the diamond icon next to the heart icon), and the alcohol blood concentration (to the right-hand side). (b) Alcoholic beverages and diamonds can be picked up from the road - the first increases the driver's blood alcohol concentration, the latter his score.

0.0‰ 2.0‰ 3.3‰

Fig. 3. The alcohol blood level directly translates to impairments of vision, hearing, and reactivity.

5 Conclusion and Future Work

In this paper, we presented an experience to adapt, teach and apply the CoSMoS process in a graduate computer science course on interactive simulation. We first laid out the multifaceted synopsis of the course before elaborating on the central role of the CoSMoS process in the context of the students' term-long projects. Finally, we briefly presented some of the results of the students' works and expanded on one of them, exemplarily. The scientific claim, the notion of self-organising processes with a focus on the interaction of numerous interwoven parts, as well as the agility of the CoSMoS process lend themselves well for backing interactive simulation projects.

Although both the results and the students' feedback have been rather encouraging regarding the course contents, its layout and its general methodology, we are eager to further improve several aspects. It might, for instance, be beneficial to have certain activities of the different phases of the CoSMoS process take place in groups during the tutorial sessions. Scoping during the discovery phase has repeatedly proven difficult to students. An experienced teacher could guide the process and ensure that multiple options are considered by each group. More generally, we believe the CoSMoS process could still be more tightly integrated in both the lectures and the tutorials, by providing an outlook of its application to the lecture units' contents. For instance, one could illustrate the application of the CoSMoS phases not only to the project as a whole but also to individual aspects such as computer graphics and visualisation—from the goals and ideas of the used assets, the designed environment, over their creation and programming to experimenting with their parameters.

So far, we have not considered building on the CoSMoS process for evaluating the students' works or their performances during the exams, except for considering CoSMoS-supported project criteria (Section 3.1). Yet, the students frequently utilised the structure of the process for classifying and presenting their work. In particular, they frequently referred to its phases and activities during their bi-weekly oral presentations and let their final project reports revolve around them. Hence, one research question that remains is whether and to which extent the individual phases of the CoSMoS process could be coupled a priori with the students' evaluation.

Last but not least, the CoSMoS process could be expanded to even better accommodate the development of interactive simulations. As they are typically designed for learning and training, an according 'engagement model' could, for instance, be an additional, desirable product of the discovery phase, complementing the domain model. It could comprise learning targets, explicitly visualised versus implicitly utilised data, the tasks and mechanics of the interfaces provided for interacting and exploring the domain model, as well as means of motivation, such as gamification elements. In combination with the domain model, such an engagement model would provide for a clear conceptual foundation for the development phase.

References

1. Akenine-Möller, T., Haines, E., Hoffman, N.: Real-time rendering. CRC Press (2011)
2. Andrews, P.S., Polack, F.A., Sampson, A.T., Stepney, S., Timmis, J.: The cosmos process version 0.1: A process for the modelling and simulation of complex systems. Department of Computer Science, University of York, Tech. Rep. YCS-2010-453 (2010)
3. Banks, J.: Discrete Event System Simulation, 4/e. Pearson Education India (2005)
4. Banks, J., et al.: Handbook of simulation. Wiley Online Library (1998)
5. Banzhaf, W.: Artificial chemistries–towards constructive dynamical systems. Solid State Phenomena 97, 43–50 (2004)

6. Batterman, R.W.: Falling cats, parallel parking, and polarized light. Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics 34(4), 527–557 (2003)
7. Bender, J., Erleben, K., Trinkle, J., Coumans, E.: Interactive simulation of rigid body dynamics in computer graphics. STAR Proceedings of Eurographics (2012)
8. Boeing, A., Bräunl, T.: Evaluation of real-time physics simulation systems. In: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia. pp. 281–288. ACM (2007)
9. Cooper, S., Khatib, F., Treuille, A., Barbero, J., Lee, J., Beenen, M., Leaver-Fay, A., Baker, D., Popović, Z., et al.: Predicting protein structures with a multiplayer online game. Nature 466(7307), 756–760 (2010)
10. Denzinger, J., Winder, C.: Combining coaching and learning to create cooperative character behavior. In: CIG (2005)
11. Deterding, S., Sicart, M., Nacke, L., O'Hara, K., Dixon, D.: Gamification. using game-design elements in non-gaming contexts. In: PART 2————Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems. pp. 2425–2428. ACM (2011)
12. Dix, A.: Human computer interaction. Pearson Education (2004)
13. Diziol, R., Bender, J., Bayer, D.: Robust real-time deformation of incompressible surface meshes. In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. pp. 237–246. ACM (2011)
14. Erleben, K., Sporring, J., Henriksen, K., Dohlmann, H.: Physics-based animation. Charles River Media Hingham (2005)
15. Featherstone, R., Orin, D.: Robot dynamics: Equations and algorithms. In: Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on. vol. 1, pp. 826–834. IEEE (2000)
16. Foley, J.D., Wallace, V.L., Chan, P.: The human factors of computer graphics interaction techniques. Computer Graphics and Applications, IEEE 4(11), 13–48 (1984)
17. Fry, B.: Visualizing data. O'Reilly (2008)
18. Fuhrmann, A., Sobotka, G., Groß, C.: Distance fields for rapid collision detection in physically based modeling. In: Proceedings of GraphiCon 2003. pp. 58–65 (2003)
19. Geiger, C., Fritze, R., Lehmann, A., Stöcklein, J.: Hyui: a visual framework for prototyping hybrid user interfaces. In: Proceedings of the 2nd international conference on Tangible and embedded interaction. pp. 63–70. ACM (2008)
20. Giavitto, J.L.: Topological collections, transformations and their application to the modeling and the simulation of dynamical systems. In: Rewriting Techniques and Applications. pp. 208–233. Springer (2003)
21. Groh, F.: Gamification: State of the art definition and utilization. Institute of Media Informatics Ulm University pp. 39–47 (2012)
22. Hong, I.K., Ryu, J.B., Cho, J.H., Lee, K.H., Lee, W.S.: Development of a driving simulator for virtual experience and training of drunk driving. In: 3rd International Conference on Road Safety and Simulation (2011)
23. Hoppe, H.: Smooth view-dependent level-of-detail control and its application to terrain rendering. In: Visualization'98. Proceedings. pp. 35–42. IEEE (1998)
24. Hummel, J., Wolff, R., Stein, T., Gerndt, A., Kuhlen, T.: An evaluation of open source physics engines for use in virtual reality assembly simulations. In: Advances in Visual Computing, pp. 346–357. Springer (2012)
25. Humphreys, P.: Extending ourselves. Science at Century's End: Philosophical Questions on the Progress and Limits of Science p. 13 (2004)

26. Hunter, P.J., Borg, T.K.: Integration from proteins to organs: the physiome project. Nature Reviews Molecular Cell Biology 4(3), 237–243 (2003)
27. Husselmann, A., Hawick, K.: Spatial data structures, sorting and gpu parallelism for situated-agent simulation and visualisation. In: Proc. Int. Conf. on Modelling, Simulation and Visualization Methods (MSV'12). pp. 14–20 (2012)
28. Izaguirre, J.A., Chaturvedi, R., Huang, C., Cickovski, T., Coffland, J., Thomas, G., Forgacs, G., Alber, M., Hentschel, G., Newman, S.A., et al.: Compucell, a multi-model framework for simulation of morphogenesis. Bioinformatics 20(7), 1129–1137 (2004)
29. Jaimes, A., Sebe, N.: Multimodal human–computer interaction: A survey. Computer vision and image understanding 108(1), 116–134 (2007)
30. Jeong, W.K., Beyer, J., Hadwiger, M., Blue, R., Law, C., Vázquez-Reina, A., Reid, R.C., Lichtman, J., Pfister, H.: Ssecrett and neurotrace: interactive visualization and analysis tools for large-scale neuroscience data sets. Computer Graphics and Applications, IEEE 30(3), 58–70 (2010)
31. Jones, M.M.: On-line simulation. In: Proceedings of the 1967 22Nd National Conference. pp. 591–599. ACM '67, ACM, New York, NY, USA (1967), http://doi.acm.org/10.1145/800196.806028
32. Kauffman, S.: The origins of order. Oxford Univ. Press New York (1993)
33. Koster, R.: Theory of fun for game design. O'Reilly Media, Inc. (2010)
34. Krause, J., Spicker, M., Wörteler, L., Schäfer, M., Zhang, L., Strobelt, H.: Interactive visualization for real-time public transport journey planning. In: SIGRAD. pp. 95–98 (2012)
35. Larsson, T., Akenine-Möller, T.: Collision detection for continuously deforming bodies. Eurographics (2001)
36. Larsson, T., Akenine-Möller, T.: A dynamic bounding volume hierarchy for generalized collision detection. Computers & Graphics 30(3), 450–459 (2006)
37. Lindemann, P.: The gilbert-johnson-keerthi distance algorithm. Algorithms in Media Informatics (2009)
38. Liu, M., Liu, G.: Smoothed particle hydrodynamics (sph): an overview and recent developments. Archives of computational methods in engineering 17(1), 25–76 (2010)
39. Macklin, M., Müller, M., Chentanez, N., Kim, T.Y.: Unified particle physics for real-time applications. ACM Trans. Graph. 33(4), 153:1–153:12 (Jul 2014), http://doi.acm.org/10.1145/2601097.2601152
40. von Mammen, S., Jacob, C.: The evolution of swarm grammars-growing trees, crafting art, and bottom-up design. Computational Intelligence Magazine, IEEE 4(3), 10–19 (2009)
41. von Mammen, S., Steghöfer, J.P., Denzinger, J., Jacob, C.: Self-organized middle-out abstraction. In: Self-Organizing Systems, pp. 26–31. Springer (2011)
42. McGuire, M.S., Jenkins, O.C.: Creating games: Mechanics, content, and technology. AK Peters Limited (2009)
43. McLoughlin, T., Laramee, R.S., Peikert, R., Post, F.H., Chen, M.: Over two decades of integration-based, geometric flow visualization. In: Computer Graphics Forum. vol. 29, pp. 1807–1829. Wiley Online Library (2010)
44. Myers, B.A.: A brief history of human-computer interaction technology. interactions 5(2), 44–54 (1998)
45. Nance, R.E.: A history of discrete event simulation programming languages. In: History of programming languages—II. pp. 369–427. ACM (1996)
46. Narayanan, S., Kidambi, P.: Interactive simulations: History, features, and trends. In: Human-in-the-Loop Simulations, pp. 1–13. Springer (2011)

47. O'Keefe, R.: Simulation and expert systems-a taxonomy and some examples. Simulation 46(1), 10–16 (1986)
48. Parker, J.R., Sorenson, N., Esmaeili, N., Sicre, R., Gil, P., Kochlar, V., Shyba, L., Heerema, J.: The booze cruise: Impaired driving in virtual spaces. IEEE Computer Graphics and Applications 29(2), 6–10 (2009)
49. Pegden, L.A., Miles, T.I., Diaz, G.A.: Graphical interpretation of output illustrated by a siman manufacturing system simulation. In: Proceedings of the 17th conference on Winter simulation. pp. 244–251. ACM (1985)
50. Petri, C.A., Reisig, W.: Petri net. Scholarpedia 3(4), 6477 (2008)
51. Polack, F.A., Andrews, P.S., Sampson, A.T.: The engineering of concurrent simulations of complex systems. In: Evolutionary Computation, 2009. CEC'09. IEEE Congress on. pp. 217–224. IEEE (2009)
52. Prusinkiewicz, P., Lindenmayer, A., Hanan, J.S., Fracchia, F.D., Fowler, D.R., de Boer, M.J., Mercer, L.: The algorithmic beauty of plants, vol. 2. Springer-Verlag New York (1990)
53. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: ACM SIGGRAPH Computer Graphics. vol. 21, pp. 25–34. ACM (1987)
54. Rothrock, L., Narayanan, S.: Human-in-the-loop Simulations: Methods and Practice. Springer (2011)
55. Sarraf Shirazi, A., Davison, T., von Mammen, S., Denzinger, J., Jacob, C.: Adaptive agent abstractions to speed up spatial agent-based simulations. Simulation Modelling Practice and Theory 40, 144–160 (2014)
56. Sarraf Shirazi, A., von Mammen, S., Jacob, C.: Abstraction of agent interaction processes: Towards large-scale multi-agent models. Simulation 89(4), 524–538 (2013)
57. Stage, A.R., Crookston, N.L., Monserud, R.A.: An aggregation algorithm for increasing the efficiency of population models. Ecological modelling 68(3), 257–271 (1993)
58. Steele, J., Iliinsky, N.: Beautiful visualization. O'Reilly Media, Inc. (2010)
59. Susi, T., Johannesson, M., Backlund, P.: Serious games: An overview (2007)
60. Teschner, M., Kimmerle, S., Heidelberger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., Cani, M.P., Faure, F., Magnenat-Thalmann, N., Strasser, W., et al.: Collision detection for deformable objects. In: Computer Graphics Forum. vol. 24, pp. 61–81. Wiley Online Library (2005)
61. Van Krevelen, D., Poelman, R.: A survey of augmented reality technologies, applications and limitations. International Journal of Virtual Reality 9(2), 1 (2010)
62. Ward, M., Grinstein, G.G., Keim, D.: Interactive data visualization: Foundations, techniques, and applications. AK Peters (2010)
63. Watt, A.H., Policarpo, F.: 3D games: real-time rendering and software technology, vol. 1. Addison-Wesley (2001)
64. Wendel, S., Dibble, C.: Dynamic agent compression. Journal of Artificial Societies and Social Simulation 10(2), 9 (2007)
65. Wong, P.C., Bergeron, R.D.: 30 years of multidimensional multivariate visualization. In: Scientific Visualization. pp. 3–33 (1994)
66. Wooldridge, M.: An introduction to multiagent systems. Wiley. com (2008)
67. Woolfson, M.M., Pert, G.J.: An introduction to computer simulation. Oxford University Press Oxford (1999)
68. Wuensche, A.: Discrete dynamics lab. In: Artificial life models in software, pp. 215–258. Springer (2009)