A Media-Rich Curriculum for Modeling and Simulation
Karen Doore, David Vega, Paul Fishwick
Creative Automata Laboratory
University of Texas at Dallas
800 W. Campbell Rd., AT10
Richardson, TX 75080
{kdoore, dxv130830, paul.fishwick}@utdallas.edu

ABSTRACT
We discuss a novel approach for teaching Modeling and Simulation (M&S) using a rich, multi-media focus with an emphasis on student construction and creative representation of simulations for dynamic systems. The increased use of M&S to enhance understanding and analyze problems across an ever-widening range of disciplines means that the diversity and number of professionals who will work with simulations is also increasing. Therefore, it is important that expanded opportunities exist for a wide range of students to take M&S courses as part of their secondary or post-secondary education. In addition to introducing M&S to new audiences, it is important to consider how professionals from these diverse disciplines can enhance and improve the quality and effectiveness of simulations. Courses, which target non-traditional simulation students, can broaden the diversity of expertise within the M&S community. We introduce an M&S course that uses Max/MSP software, which is familiar for many multimedia students and faculty. The course targets a mixed class of graduate students from Art and Technology, Computer Science, and Engineering programs.

Categories and Subject Descriptors
K.3.2 [Computers and Education]: Computer and Information Science Education

General Terms
Human Factors

Keywords
Max/MSP, Modeling and Simulation, Education, Data flow, Petri nets, Creativity, IoT, Microcontrollers.

1. INTRODUCTION
With increased use of modeling and simulation across diverse domains, there is a growing need to expand the offering of university courses in modeling and simulation concepts to a wide range of students. Modeling and Simulation (M&S) can be considered a relatively young discipline, and one that is highly multi-disciplinary; this complicates the determination of what should be taught in a graduate-level M&S course [31]. Padilla et al. [36] discuss the diversity of subject areas and fields within the discipline of Modeling and Simulation, while arguing that it does have enough coherence as a discipline to justify establishing M&S as a science in its own right. The authors provide a broad overview of M&S as a formal academic discipline with three main components: problem situations, ontological, teleological and epistemological constraints of problem situations, and computational constraints [36].

Fishwick [10] defines computer simulation as ‘the discipline of designing a model of a system, executing the model, and analyzing the execution output.’ He argues that learning simulation should embody the principle of ‘learning by doing’; he further emphasizes that “in order to learn about a system, we must first build a model and then make it run.” This raises the question of what pedagogical content should be taught to students interested in learning M&S. While there are many factors that should be considered when designing new curricula, there is a growing consensus within the Computing Education Research community that curricula, which engage learners through contextually relevant materials and practices, can often improve learning outcomes [17, 28, 43].

Our course has some unique curriculum-design constraints: how can we teach Modeling and Simulation to graduate students who have an extreme disparity in their prior knowledge and experience with regards to mathematics, computing, technical and artistic fluency? Our goal is to challenge all students to approach modeling and simulation from new perspectives with an explicit focus on fostering what Cropley [5] has termed ‘functional creativity.’ For computer science graduate students, developing simulation programs using a visual data-flow programming paradigm, in addition to a focus on creating sensory-rich simulations requires a different mindset compared to traditional computing courses. For Art and Technology students, the multi-sensory creative approach is well within their comfort zone, as well as using visual data-flow languages, however, our goal was to challenge them to approach the creative process from the perspective of modeling & simulation foundational concepts. Concepts such as message passing, discrete events, state-transitions, feedback, concurrency, parallelism, with an explicit focus on process dynamics are typically not taught in most media art courses, with the possible exception of audio-focused media courses. The software used in our course, Max/MSP, is designed for artists, educators, researchers, working with audio, visual media, and physical computing [29].

We provide an overview of traditional M&S education approaches, which range from teaching applied simulation to curricula with a more theoretical focus. This is followed by a review of some alternative curriculum approaches. Then we examine functional creativity and educational approaches to foster creativity in technical disciplines. We include an overview of data flow programming, data-flow visual languages, and then give a brief overview of Max/MSP software, which was utilized for the
course. A survey and discussion of several student projects will provide examples of the benefits of this media-rich approach for teaching modeling and simulation. Finally, we discuss possible extensions to this curriculum, which includes integration between Max/MSP and the Arduino IDE to control microcontrollers like the Arduino Uno.

2. M&S EDUCATION

The rapid expansion and use of simulation across domains, coupled with improvements in ease of use of simulation software systems creates opportunities for teaching M&S to a more diverse target audience [34, 46]. A review of the M&S education literature reveals that there are significant changes in target audiences for simulation courses due to the growing use of simulation in industry [14, 46]. Ideally, curriculum design should be tailored to support the unique learning objectives for each of these diverse groups. Much of the M&S education literature about curriculum and pedagogy seems to be focused on courses taught in engineering, computing, and business schools [14, 18, 25]. Kincaid et al. [23] did a review of M&S education literature of graduate-level courses and found that ‘these programs are usually focused on the mathematical, hardware, or management aspects of M&S.’ We argue that many students from diverse disciplines across the university campus will benefit from courses in M&S; however, these courses should be tailored to support this more diverse set of student interests and background knowledge. In addition, the M&S community can benefit from this type of course, as it may drive interest in M&S careers, where these future artists, educators, designers, and user-experience professionals can use their expertise to enhance M&S products.

2.1 M&S Curriculum Design Factors

We have embraced a learner-centered approach to curriculum design; a similar approach has helped influence how introductory computing courses are being designed to target a wider range of student interests and majors [16, 17, 43]. Szczerbicka et al. define a ‘simulationist’ as someone who fits within an extremely diverse range of roles, who works with simulations; then the authors discuss the diversity of educational needs for ‘simulationists’ [46]. Greenwood and Beverstock [14] cite seven reasons why changes should be implemented to improve M&S curriculum at the university level. Loper et al. [27], discuss the wide range of difficulties associated with designing curriculum to support growing need for an M&S educated workforce. Complicating the curriculum design process is the diversity in M&S approaches, such as discrete event simulation (DES), agent based simulation, aggregate approaches to simulation, and hybrid approaches [34]. For each of the simulation categories just mentioned, there are a broad range of software simulation systems, some of which are designed to support teaching / learning while others designed for applied simulation analysis including enterprise-type applications [45].

2.1.1 Education for Applied Simulation

Greenwood and Beverstock [14] suggest curriculum changes that focus on the importance of teaching applied simulation in contrast to more theoretical approaches. They highlight the fact that the role of simulation in the workplace has expanded and that simulation is now an integral part of decision support systems, where most users of simulation systems won’t be simulation experts. These ‘intermediate’ [14] users will more likely be domain experts, using simulation as an applied tool to support analysis and decision-making. In contrast, Jain [18] expresses concern that simulation courses in many business schools primarily focuses on teaching the use of simulation software rather than teaching students how to build simulations.

2.1.2 Conceptual Modeling Focus

Jové et al. [20] discuss the importance of conceptual models when teaching Discrete Event Simulation (DES) to engineering and computer science students. They recommend a curriculum based on the use of Petri nets as a modeling language to allow students to develop a conceptual model of a system and then to map that to a simulation model implementation, using Arena software. They further note that for these highly technical CS and engineering students, it is critical that they understand fundamental modeling constructs rather than “just teaching the use of certain software” [20]. Jové et al. have identified an important learning objective: ‘students must understand the difference between a conceptual model and the implementation of the model.’ The Petri net model structure provides a visual language to represent a DES system, where each visual element in a Petri net model corresponds directly to a distinct discrete event simulation concept [52]. In our course we also focused on theoretical and foundational aspects of M&S, this included a unit that focused on the use of Petri nets to model a range of systems.

2.1.3 Blended Approaches

The above approaches represent endpoints within a range curriculum approaches; either a purely applied simulation focus, contrasted with a purely theoretical approach. Kress et al. [25] present a blended approach for a course that targets non-engineering graduate students. They note that these graduate students are primarily from mathematics or science disciplines, and these students typically have extensive experience working with mathematical modeling, numerical simulations, partial differential equations, etc., however they often lack experience with discrete event approaches to simulation. Their goal for the course is ‘to prepare these students to work effectively in a typical industry or government-supported simulation modeling group [25].’ Accordingly, their curriculum begins with discrete event fundamentals, including a comparison between discrete-event and continuous models, then progresses to cover modeling theory, techniques and applications which were implemented with ExtendSim software. While their target audience has diversity of prior knowledge based on discipline, it appears that they would have similar depth of knowledge of fundamentals of mathematical modeling, to enable the course to cover such a broad range of content in a technically rigorous manner. The target audience for our course lacks the shared depth of knowledge of mathematical modeling concepts, this the primary constraint impacting our curriculum design.

2.1.4 Simulation and Instructional Design

One important aspect of the changing face of Modeling and Simulation is the increased use of simulation for training and instruction. Szczerbicka et al. identify training and education simulation as a major component of the M&S domain, both for training in other disciplines as well as for educating ‘simulationists’ [46]. Kincaid and Westerlund [23] provide a discipline-oriented taxonomy of M&S knowledge for the purpose of showing the ‘breadth of knowledge and skills’ of various M&S professionals. The authors specifically discuss the need for professionals in M&S to develop skills in human factors and instructional technology; these skills include UI design, usability,
media selection, task analysis, learning theory, to name a few [23]. The design and development of simulations for training for use in areas like medical training, emergency response training, cultural training, or workplace training, requires collaborative teams of professionals with a diverse range of skills.

2.1.5 Multi-Modal Interactive Design

Improvements in technology provide opportunities to create enhanced, media-rich simulations. In order to integrate these multi-sensory components, designers with multi-media expertise will need to collaborate with simulation developers. For example, Taylor et al. [47] are incorporating sonification capabilities in a fluid dynamics simulator, which they’ve named the ‘Virtual Wind Tunnel’. They used Max/MSP software to design and implement their sonification strategy with the goal that integration of multimodal interaction might enhance user’s understanding of complex simulation data [47]. Future iterations of their project will include haptic user-interaction. In a separate project, Taylor and colleagues [48] have used Max/MSP to create an artistic participatory performance as a means to understand collaborative creative behavior, which could be integrated into the design of interactive systems. They also used Max/MSP to integrate remote tangible interfaces that participants used for interaction with the system.

As the art of simulation design evolves to include more diverse media types, it will be important that the design process includes creative artists and designers who can provide enhanced aesthetics for multi-sensory interfaces. For example, Bak et al [2] use Max/MSP software in a course designed to ‘provide Interaction Design students with tools and skills to develop complex multimodal, embodied experiences.’ We contend that it’s not enough to have simulations that present accurate measures of a dynamic system; these systems should be designed to provide an engaging, realistic, multi-sensory, immersive experience. Dance et al. notes that it is important for training to accurately reflect the real-world scenario to support deep learning from the training session [8]. Shams et al. have conducted research that indicates that ‘multi-sensory training protocols can better approximate natural settings and are more effective for learning [41]. Our course used a multi-sensory and user-interactivity focus for project assignments. This emphasis was designed to highlight the importance of these factors on end-user understanding of the modeled system, and to encourage students to design system representations from an aesthetic, integrative, and reflective perspective. We argue that an important component of M&S education is for students to develop an understanding of simulation user-experiences in order that they might embrace these aesthetics when they are simulation professionals.

2.2 Insights from Computing Education

While teaching Modeling and Simulation is not the same as teaching introductory programming courses, research in field of Computing Education Research (CER) can inform curriculum design for modeling and simulation courses since there is broad overlap in the nature of concepts. For example, programming and M&S courses have the common goal of teaching students how to design and implement some form of a system based on abstract modeling concepts [40, 9]. The abstract nature of the concepts in these courses means that the software and context of what is taught or modeled can vary greatly across a wide continuum [26, 45]. Curriculum design for these courses is often, at least partially, based on use of a specific software system or language, where trade-offs between factors such as: ease-of-use, complexity, cost, and representational expressiveness, have an impact on the how the course is taught [28, 45]. Factors such as problem context, software, and language, can have a large influence on student interest, engagement, and learning outcomes, particularly based on the level of background knowledge and technical fluency of the learner [24,4].

2.2.1 Computing Education for Diverse Audiences

In addition, there has been a recent increase in CS Education research (CER) that has focused on teaching computing concepts to non-majors, where motivation, relevance, context, and representational format are common research themes [16, 17, 28]. The goal of teaching computing concepts to non-experts, with a goal of increased general computing literacy can also provide a lens to improved methods for teaching modeling and simulation to students who will use modeling and simulation in profession as a tool for problem solving, rather than with a goal of becoming a simulation expert. In addition, after graduation, these students may become designers of simulations, where their role is to enhance user experience.

2.2.2 M&S in Computing Education

Another area of research within the CER research community is the use of modeling and simulations to teach computing concepts for the purpose of introducing students to computing. Researchers, including Stroup and Wilemsky [44, 53] often use contextualized simulations to teach abstract concepts like cause and effect, distributed or decentralized systems, feedback, rule-based systems, emergent behavior and complex systems. Lin and Tater [26] have studied the difficulties that even advanced CS students have with understanding parallelism, concurrency and coordination concepts. They recommend teaching students to explicitly model coordination, yet also note that for students without strong programming skills, teaching these concepts using traditional programming language constructs such as Java threads can be difficult [26]. We have taken inspiration from computing education research, with a focus on teaching foundational aspects of modeling and simulation, with a necessary focus on connecting these fundamental abstract concepts in a context that is relevant to both media artists and computing graduate students.

2.2.3 Multi-sensory Teaching and Learning

Katai and Toth [21] have conducted research to explore the benefits of teaching abstract computing concepts to undergraduate students using a multi-sensory approach with a well-designed integration of technology and arts. Their study integrated music, video, graphical representations and choreographed dances to provide students with a multi-faceted, sensory experience for a variety of sorting algorithms. Their results indicate that using a multi-sensory, art-integrated method improves students’ skill to analyze and design algorithms. They also referred to their prior research study using similar teaching methodology. They found similar improvement in another difficult-to-teach computing concept, resulting in students’ improved skill to analyze, design and implement recursive procedures and functions [21]. This improved understanding of abstract concepts through integration of sensory-rich media provides motivation for our use of Max/MSP as the software component for our course.
3. CREATIVITY IN M&S EDUCATION

Cropley suggests that fostering creativity within the domain of Modeling and Simulation can result in improved innovation in both the M&S domain as well as in engineering domains since “the engineering process is supported and enabled by modeling and simulation” [7]. According to the author, creativity has become a valued and ‘vital component of engineering practice’ and is seen as a key to innovation, where engineering creativity is articulated as engineering which ‘results in the output of novel useful products.’ Kaufman [22] notes, ‘creativity has been described as the most important economic resource of the 21st century.’ Dance and Fishwick [8] note that the ultimate goal in the design of a model is to achieve effective communication of system behaviors to improve understanding of relationships within the model. The authors further argue: ‘A model that is aesthetically pleasing will result in enjoyment and improved utilization’ [8]. Fishwick urges deeper integration of aesthetic considerations within computing cultures to enrich computing experiences [12]. The notion that a primary role of models and simulations is to function as artifacts for supporting communication means the creative aspects of M&S should be an important part of curriculum design decisions. For these reasons, we argue that it is important to include creativity and aesthetic design as components in an M&S course.

3.1 Creativity and Innovation in Engineering

Cropley et al. [5, 6, 7] provide an overview of the increase in perceived value of creativity within engineering disciplines. He points to the 1957 Sputnik satellite launch as a turning point, where creativity suddenly became an important consideration in evaluating the failure of American engineering innovation in the context of the global ‘Space Race’. Cropley notes that Gilford, a preeminent psychologist in the 1950’s, identified ‘convergent thinking’ as a main cause of this lack of creativity and innovation of American engineers [7]. ‘Convergent thinking’ [15], a term introduced by Gilford, was thought to be the result of teaching approaches that emphasized rapid acquisition of factual knowledge, rapid recall, having clearly defined goals, logical thinking, student compliance, and other similar processes [7]. This is in contrast to Gilford’s notion of ‘divergent thinking’, which involves ‘branching out from the given’ to explore novel solutions.

3.2 Functional Creativity

Cropley [7] makes a distinction between creativity in the fine arts as compared to engineering creativity. He defines engineering creativity as ‘functional creativity’, where the focus is on creating products to solve problems. Cropley [5] notes that process is central component of functional creativity, and he identifies four dimensions of creativity of engineering products and explains how they form a hierarchy that can be used as a lens to view the likely benefits of enhanced creativity in the M&S domain. Cropley’s [7] four-dimensional model for defining the creativity of engineering products is: relevance and effectiveness, novelty, elegance, and generalizability. These four dimensions can be used as criteria to evaluate the value of creative products, and can provide guidelines as to inspire creativity when designing products that would range from routine to original, and elegant to innovative solutions.

3.2.1 Fostering Functional Creativity

While Cropley provides strong arguments indicating there is a consensus regarding the importance and value of creativity in engineering domains, he also acknowledges that there is a problematic disconnect because few engineering programs recognize the ‘importance of creative thinking as an essential teachable skill’ [5]. The author suggests that many engineering curricula are overly focused on ‘driving toward ever greater breadth and surface learning in technical topics only’, and as a consequence, creativity is not taught or fostered as an integral component of most engineering curricula. As discussed earlier, M&S curriculum designers must also determine how to strike a good balance between covering the breadth or depth of a rapidly expanding discipline.

3.2.2 Simulation-Thinking and Functional Creativity

Cropley and Cropley [6] highlight the notion that links between creativity and engineering are often viewed with an explicit focus on educational changes that could foster creative thinking as an integral part of engineering curricula. Thompson [50] discusses the importance of integrating creative design and analysis in a cohesive manner for engineering curricula. She suggests that simulation courses might provide a good opportunity to have students learn more integrated design-analysis thinking. Thompson coined a new term ‘simulation-thinking’ to capture a more integrated design-analysis thinking process [50]. Kaufman and Beghetto [22] have developed a model of creativity that can provide insight for fostering creativity as part of the learning process. The authors identify ‘mini-c’ as a category of creativity, which recognizes the ‘creativity inherent in students’ unique and personally meaningful insights and interpretations as they learn new subject matter’ [22]. It is this type of creativity that we aim to foster in our course; students in our course are expected to explore creative representation in learning modeling and simulation.

3.3 Education and the Art of M & S

Paul et al. raise the question of whether M&S should be taught as a ‘science’, or whether it’s possible to teach so as to provide students with an appreciation that ‘simulation is more of an art than science [37]’ The authors argue that simulation education should help students recognize that many simulation problems are by their very nature ‘not well understood’, and recommend that creative problem solving should be a critical component of an M&S course. Paul et al. [37] advocate that M&S education should have four general objectives: teach students how to learn, how to think creatively, how to problem solve, and how to be professionals. The authors encourage a course design where students learn that the iterative process of modeling helps clarify problem features, and that ‘simulation models are not about being correct or incorrect,’ they are a vehicle to elicit discussion and debate in order to allow problem owners to develop an enhanced understanding of their problems [37]. This works best when models are perceived as a representation of the modeled system, and that ‘models may be purposefully distorted’ in order to enable discovery of enhanced insights about a system [37]. Our course design reflects a similar perspective of M&S, which emphasizes the importance of models as creative products, which are designed through an iterative constructive process, where creative abstraction used to emphasize particular model features in order to create interest or enhance understanding.

4. COURSE: CREATIVE MODELING AND SIMULATION

An important aspect of our course is the process of designing creative system models. This encourages students to engage in
creative reflection about how the features of their project can be represented so as to provide an interactive experience so that a user will develop an improved understanding of the system and process being modeled. We want students to embrace the idea of models as creative artifacts, which should be designed to provide insight and to communicate details of a system. In addition, we encourage students to create multi-sensory models and simulations that will be interesting and engaging for users exploring their model. Another important aspect of the course is that it was highly interactive, the course was held in a computer lab with a participatory class environment; where students frequently presented their projects in order to share learning, to generate feedback and to encourage discussion on a regular basis.

Our class size it is generally less than 25 students. Duration of the projects varies, we conduct most projects in a 2-phased approach, the initial phase requires students to submit a project proposal. This gives us an opportunity to provide guidance, which is often needed for such open-ended creative projects. Project duration is approximately three weeks, one week for proposal and one-two weeks for the final implementation. We try to interleave projects so as students are completing implementation of one project they are in the ideation stage for their next project proposal.

4.1 Visual Data Flow for Teaching M&S

We have used Max/MSP software for our course, which is a visual data flow programming language. There are numerous reasons why we feel it is a good fit for use in an M&S course. In the sections below we provide a brief overview of some of the benefits for using visual data flow language like Max for an M&S course.

4.1.1 Data Flow Programming Paradigm

While there are numerous variations of flow based programming approaches, in general, they emphasize the movement of data, where the structure of the program is represented as collections of communicating, asynchronous processes [19, 32]. A primary benefit of the data flow paradigm is that the program structure aligns well with implementation requirements for parallel and distributed processing [32]. In addition, the graphical structure of flow-based programs closely resembles the physical processes that it is modeling [19]. Morrison [31] notes that the technology provides a consistent application view from high-level design all the way down to the implementation. Research has shown that having a mental-model with consistent and explicit mappings from the conceptual level to the program implementation level is critical for program comprehension and understanding [13]. In fact, Johnson et al. cite research showing that most software developers ‘naturally think in terms of data flow in the design phase.’ [19]

4.1.2 Data Flow for the Internet of Things

Namiot et al. [33] discuss the similarity between data-flow software models and the physical constraints that impact designs for communication between embedded hardware in both Internet of Things (IoT) and Machine-to-Machine (M2M) systems. They have suggest that the flow-based programming paradigm can provide a simple model for M2M communication, where devices would expose interfaces to expose or consume data rather than a more complex API. They argue that a simple protocol would work well for devices like sensors, which don’t have support for processing program commands. Max/MSP provides a prototyping environment for IoT and M2M systems [47], since it uses a data-flow paradigm and since it also provides firmware for serial communication with microcontrollers. In our current course, we are using Max to create dynamic prototypes for physical IoT systems, and to provide interactive controls for IoT and M2M systems. The ability to create functioning, virtual–dynamic, rapid-prototypes of physical computing systems provides a powerful teaching environment for M&S. This type of project is discussed in detail in section 5.2.2. Figures 7-9 show examples of this type of mapping from conceptual model to max prototype dynamic-model to physical computing representation model.

4.1.3 Data Flow Visual Programming Languages

Researchers in the field of software engineering have recognized many possible benefits for using data flow visual programming languages (DFWPL) [19, 35]. Johnson et al. have noted that the distinction between requirements, program design and coding phases becomes less obvious when using DFWPL, however, the aesthetics and design of the DFWPL environment have a large impact on the ease of use [19]. The authors also note that visualization and animation are essential components of visual programming environments. Nierstrasz et al. made an early (1992) distinction between application engineering and application development, where they envisioned that application developers would be domain experts who would utilize visual scripting to connect compatible software components in order to create applications [35]. They used the term ‘script’ to suggest an analogy between the data-flow visual script application and a theatrical performance.

4.1.4 Multi-Media Data Flow Visual Languages

Nierstrasz et al. [35] developed a multimedia platform, which utilized a data-flow paradigm. They chose a multimedia application domain specifically because they envisioned that their ‘new ideas might fare better in an area’ where inertia and cultural bias against visual scripting wouldn’t negatively impact their project. They utilized a ‘radically different’ approach to OOP application design, as they viewed objects as ‘patterns of communicating agents’, with the goal of developing better software design methodologies, which emphasized component modularity and reuse [35]. Another benefit of application design in the domain of multimedia is that it provides natural parallels because application components actually resemble ‘digital production studio’ physical hardware or software processes, which generate audio events [35]. This strong natural connection between data-flow software engineering, media arts, and physical computing can provide a strong bridge for introducing simulation concepts to media artists and designers. It can also provide a fertile test area for the design of immersive, multi-sensory enhanced, interactive simulation systems.

4.2 Max/MSP Features and Architecture

Max/MSP is a visual-language based software system, which employs a data-flow/control-flow visual syntax. Max uses the visual metaphor of analog electronic music components such as amplifiers, synthesizers, etc., where discrete components are connected via patch cords. Data passes through the patch cords encoding a variety of signal-types such as MIDI, audio, text, and JSON formatted data. Max is designed for creating interactive music compositions, and therefore supports both regular and variable time signals; the metronome object represents discrete time, and this object works well for triggering events. In addition, MSP (signal processing) components provide oscillating audio signals based on digital signal processing components, which can
be used to model dynamic systems. Digital music and media artists use Max/MSP extensively, to create interactive and dynamic compositions including performance art [29].

4.3 Max for Teaching Computing Concepts

Several researchers have discussed the use of Max/MSP software for teaching computing concepts. [2, 9, 28] Manzo [28] notes that the Max data-flow paradigm is intuitive, and the fact that all Max components are objects supports students in learning object-oriented thinking. In addition, Max also provides an API to connect with microcontrollers, and it provides many user-interface components that can be used to control hardware for rapid prototyping of embedded systems. [2, 47] In our M&S course, students were tasked with creating a variety of discrete event models and dynamic simulations. Students were required to incorporate creative visual representation, user interaction, as well as meaningful audio output, using Max objects as construction primitives. Our current iteration of the course has students integrating Max programs with creative physical prototypes of dynamic systems.

4.3.1 Creativity and Aesthetics for M&S Education

As Cropley et al. [5] and others have noted, it’s important to nurture creativity, particularly when designing courses situated in technical domains or cultures. Nierstrasz et al. [35] chose a multimedia domain for their software engineering research particularly because they wanted a more creative culture as a test-bed for their application design. We concur with Paul et al. [37] that M&S curricula should encourage creative problem solving in conjunction with traditional M&S concepts. We believe that Max/MSP software provides a solid framework for teaching creative M&S since it uses a visual data-flow format and supports student creation of multi-sensory artistic modeling products. Fishwick suggests that we should endeavor to externalize our mental models as ‘real and virtual models, which have sensory and aesthetic qualities [11].’ He argues that aesthetics should play an important role in modeling and provides a methodology for incorporating aesthetics into the model design process [11].

- Choose a system to be modeled
- Select model types
- Choose an aesthetic
- Define Mapping
- Create Model

Max allows us to extend this model in two important ways. First, Max allows for rich integration of multiple sensory modalities, along with providing a framework that supports hierarchy and dynamic-encapsulation of model layers, where embedded layers can be designed to reveal visual dynamic features. Second, Max supports integration with microcontrollers, which extends opportunities for creating physical model representations. This support for a creative design progression from conceptual to programmatic to physical models creates a fabulous learning opportunity for students. The process of mapping model concepts, components, coordination structure and aesthetics between these model layers provides students with a rich modeling experience to enhance deep learning.

4.3.2 Brief History of Max/MSP

Max was first developed at IRCAM, by Miller Puckette in the late 1980s; it was initially developed for use with the Macintosh computer [29]. Max was commercially available in 1990 and was chosen as ‘Software Innovation of the Year’ by readers of Keyboard Magazine in 1991 [29]. In 1996, Puckett started development of Pure Data, (Pd), which is an open-source version of Max/MSP. ‘Pd was created to further the Max paradigm by extending data processing to applications other than audio and MIDI, such as real-time video and web interaction [39]. An important distinction between Max and other audio software is that Max is not audio editing software, and it is not a simple audio synthesizer. Max can be considered an ‘audio-rendering environment’, where all processing is done in real time and the creative process is more analogous to creating a dynamic instrument. [29] Max provides a visual language interface to specify audio and media synthesis and signal processing algorithms in a real-time, live-code environment [29].

This environment provides immediate feedback and the visual format may make it easier to conceptualize signal processing algorithms [29].

4.3.3 Max Data Flow Objects

Max programs are called patcher files, and a patcher consists of a number of max objects connected together with patch cords. Max has an extremely large library of built-in objects that can be used to build programs. It also provides several ways to create custom objects, including JavaScript, Java and C++ components. Most Max objects have a number of inlets, which can be connected to the outlet from a different patch. The left-most inlet is usually a ‘hot’ inlet, and it is designed to receive control signals indicating that the object should begin executing its behavior. A max object inlet can accept many messages and message-types, depending on arguments defined for that object; in addition, each object also has a large number of attributes, which can be used for initialization and UI-customization. The physical layout of objects on a workspace layer impacts the order of execution, or the ordering of messages sent between objects. Other objects, like route and select, can be used to explicitly control the order of execution.

4.3.4 Encapsulation and Object Hierarchy

Max is designed to support dynamic viewing and interaction with an executing program across any number of object layers. A patcher file can contain several different levels, which represent object hierarchical relationships. So, a patcher file can be embedded as an object within a parent patcher file; it becomes a sub-patcher with it’s own inlets and outlets. Any selection of connected max objects on a workspace can be converted to represent a child sub-patcher object. The basic sub-patcher is represented as a simple rectangular object with the appropriate inlets and outlets as required within the sub-patch for connectivity with the parent patch.

In addition, Max provides the bPatcher object, which allows for creation of customized dynamic graphical components. In student example projects below, we will provide a detailed description of bPatcher objects to show how they enable encapsulation of Max sub-patcher objects to create custom visual-dynamic objects. In essence, each layer within Max represents an object that has a hierarchical relationship with other layers in a patcher program. For each object layer, specific objects can be selected to be visible in a presentation-mode view of the program. In addition, Max patches can be edited while they are in dynamic execution mode. The object-oriented, hierarchical design of Max patchers and sub- patchers, combined with powerful flexibility for controlling and customizing the dynamic user-interface components makes Max an extremely interesting option for teaching M&S concepts.
5. CASE STUDIES

We will present a few course project assignments with detailed overview of student project submissions, to explore the creative diversity of student projects as well as to highlight the capabilities of the Max/MSP software for creating interactive representations and implementations of Discrete Event Simulations. For one project, students created a Max implementation of a Petri net system, which could be used as a toolkit for creating dynamic Petri net (PN) models. In addition, students were required to explore the creative representation and implementation of a non-trivial process that could be simulated using their Petri net system.

5.1 Petri Net Project

Fishwick defines Petri nets as a modeling framework, which is primarily used for ‘studying dynamic concurrent behavior of systems where there is discrete flow [10].’ Peterson notes that a Petri net has static properties that are represented by a directed graph with two distinct node types: circles called places and bars called transitions. The dynamic properties are represented by a token that moves around the graph along directional arcs. [38] Peterson notes that ‘the simplicity and power of Petri nets makes them an excellent tool for working with asynchronous, concurrent systems’ [38].

Baros [3] and also Jové et al. [20] recommend the use of Petri net models to teach fundamental modeling and computing concepts. Baros notes that students often have difficulty when trying to understand concurrent systems behavior. He argues that, when students construct Petri net models, they develop ‘a much more intuitive perception of dependencies (synchronizations) among processes [3].’ Jové et al. conclude that having students construct Petri net models, with a focus on conceptual modeling, helps students understand the distinction between a system, a conceptual model, and the implementation of the model [20].

The figures below are examples of one student’s Petri net project. This student created an implementation of the Dining Philosophers problem (DP) with his Petri net system. The DP problem can be considered a proxy for studying characteristics of human task coordination, where often a high-level task cannot be decomposed into disjoint parts, so several people must coordinate their activities to complete a task [10]. The first image shows a dynamic, creative representation of five philosophers seated at a dinner table. This figure shows the high level, presentation-view of this Max project. The student created user interface controls to allow a user to start/reset the simulation. He also created several dynamic data-widget elements that provide statistics about the simulation as it progresses in real-time. This dynamic interface indicates which philosophers are currently eating, and shows the location of the unused chopstick. In addition, the student integrated an audio track to provide ambient restaurant sounds, in order to create a multi-sensory presentation.

In Figure 2, the inner-workings of the highest-level Max patcher hierarchy for the DP project is shown. The student has used colored panel objects to organize logical sections of patcher code. He has used Max send and receive objects to provide communication between panels; this hides the patch cords from view and creates an aesthetic design to provide improved visual organization of his project. The figure shows three high-level code sections, the yellow panel represents the clock-signal control, the purple panel contains PN-Places, which represent the chopstick resources, and the teal panel contains PN-Transitions, which represent the philosophers. This snapshot image shows that the program is in execution-mode, and Place3 has an active token; this indicates that a chopstick is available. The teal panel, shows that Transition4 is active and highlighted green. This indicates that philosopher4 is currently eating.

This student implemented logic so the PN-transitions have an associated time delay, which corresponds to the time-period when philosophers are eating. Inputs to the philosopher-eating transition are three incoming arcs, which are shown as part of the PN-transition icon; these represent two chopsticks and one food PN-places. There are two outgoing arcs emanating from each PN-transition; these correspond to the chopstick PN-places, so once a transition has completed firing, the two chopstick tokens are returned to their PN-places to enable a different philosopher to have an opportunity to eat.
5.1.1 Petri Nets

The process of creating a PN system may provide students insight into subtleties of this modeling paradigm that might otherwise go unnoticed. Some students used the metro object as a global clock to send a synchronizing control signal to all transitions in order to have them all fire in parallel. Other students designed their PN transition components to insure fairness in transition ordering; to implement this behavior, students used the Max urn object, which generates random numbers without replacement. In Figure 3, which is an embedded sub-patcher designed to control the transition activations, the metro object is connected to an urn object, which controls activation of the transitions. In the image, the output value from the urn object is input to the route object, which provides control-flow capabilities, to activate the selected transition. The bottom of the figure shows that there are six outlets for this sub-patcher, the first outlet is used for testing purposes, while the outer outlets are connected directly to custom designed PN transition objects in the parent patcher. The pink highlighted boxes show that this student has created user-interface elements for the presentation-view of this custom max object. This will show an LED indicator light as each transition is activated via the urn object; currently outlet1 is shown as active.

Figure 3. Random Transition Activation Using Urn Object

5.1.2 Event and Object Coordination Messages

Another subtle concept that students discovered when building Petri net components, was that additional messages are required in order to coordinate verification of the state that corresponds to the presence of tokens for PN places associated with incoming arcs, to determine whether a PN transition can fire. Then, once a transition does fire, additional messaging is required to adjust the token state of each associated PN place. Figure 4 shows that four different types of patch cord messages are input into the PN Place object: Place1. The Max objects ‘s’ and ‘r’ correspond to hidden patch chords, where each send and receive object is connected with some other Max object inlet or outlet, in order to receive or send messages. This layer of message-passing for coordination is often hidden from users when using higher-level simulation software, however these implementation-level details provide concrete demonstration of the importance event orchestration and system architecture in M&S, which non-major students may not have been exposed to prior to this course.

5.1.3 Bpatcher: Custom Visual Dynamic Objects

The PN-place object, which appears as a black circle on a white square in Figures 2, 4 and 5, was created using the Max bpatcher object; the PN-place object icon is a visual interface for an embedded sub-patcher of lower-level logic. Figure 5 shows a partial view of the internal logic for a bpatcher PN-place object, which includes three inlets, and three outlets that correspond to the place UI component from Figure 4. While it is difficult to see object details in this figure, it should be clear that the object-oriented architecture of Max, combined with an extensive library of customizable user-interface elements, provides a powerful and creative environment for designing interactive models and simulations. Students in this course created custom bpatcher objects as the primary means to construct interactive representations for their simulation models.

Figure 4. Inlet Messages of Place1 Bpatcher Object

Figure 5. Implementation Details of Place Object Sub-Patcher
5.1.4 Creative Audio Representation

Max provides many audio objects, and we encouraged students to integrate creative audio for all of their projects. So while it is difficult to share the creative aspect of student designed audio components, this was a very important part of the course. We had several students with strong music performance and audio design backgrounds, and the audio components of their simulations were quite beautiful. We suggested that students integrate audio in a meaningful way with their simulations; often that corresponded to having students analyze their model to find event triggers for audio events. Figure 6 below shows implementation of a Karplus Strong audio generator, which was triggered in conjunction with a Petri net transition firing. The integration of creative audio for these projects helped students view simulations from a more holistic and multi-sensory perspective, it encouraged reflection on their model to determine how audio could be meaningfully integrated.

5.2 Max: Limitations, Integration with IoT

5.2.1 Limitation: Web Sharing of Programs

Unfortunately, Max patcher programs cannot currently be embedded as interactive components on web pages. This somewhat limits the impact that our student-designed creative simulations can have, because we cannot easily share student projects as interactive online applications. In order to execute a Max program, users must have Max software installed on their system and they must download a patcher file in order to experience and interact with it. However, Max does provide an easy method to share patcher files using a serialized data format that can be copy / pasted in a text editor. In addition, Max has a very robust user community, so there are many resources available online such as videos, tutorials, and an active forum to support users [29]. In addition, within the Max software environment, there is an extensive, well-organized repository of documentation, tutorials, and live-editable code examples for every Max object [29].

5.2.2 Integration with the Internet of Things

Max has an extensive set of objects that provide web integration within a Max patcher. Max components can be used to create user-interfaces to interact and control physical hardware using web protocols. In addition, Max includes firmware to support integration between patcher files and microcontrollers like the Arduino. It is interesting to note that the Arduino microcontroller started as a project for students at the Interaction Design Institute IVREA. One goal of the Arduino project was to create an inexpensive open source microcontroller for student design projects [1]. In our current course, we are using Max to prototype physical simulation models and we are also using Max for designing user-interface controls for these physical, tangible simulation components. Figure 7, below, shows an implementation of the Fibonacci sequence equations, which was created using Max/MSP.

![Figure 6. Karplus Strong Integrated Audio Event](image)

![Figure 7. Max Patcher Fibonacci Model](image)

![Figure 8. BPatcher Logic for X(n-1) Object](image)

Figure 9 shows the physical prototype of a hardware circuit, which was created using five individual Teensy 3.1 microcontrollers [49]. The physical implementation of the circuit design is isomorphic to the Max patcher. Both models implement synchronous architecture, and utilize serial protocol for communication. The metro object in Max, functions as a software clock; it generates a control signal that has a constant (configurable) interval. This architecture is mirrored in the physical circuit where a single Teensy is programmed to behave like the metro object; it sends a control clock signal to each of the
other Teensy boards in order to synchronize the data-flow between autonomous components.

![Figure 9. Teensy Microcontroller Fibonacci Model](image)

A third phase of this project requires the student to creatively embed the physical circuit components in physical objects, which must also have an isomorphic mapping with the Max and electronic hardware prototypes; this final project must be a creative representation of the modeled process. Figure 8, provides a view of the sub-patcher object logic for the X(n-1) Fibonacci mathematical equation component. The Max object hierarchy provides insight into the physical hardware architecture we’ve implemented, the lower-level, custom sub-patcher object logic corresponds directly to the logic that is implemented within a the X(n-1) Teensy board.

### 5.3 CONCLUSION

One goal for our course is for all students to develop a deeper appreciation of the role art and aesthetics in M&S, where multi-sensory representations can facilitate communication and enhance understanding. Another benefit that may result from this blended mix of students from diverse disciplines is that the shared experience and discourse may inspire students in their professional careers to be more effective communicators and collaborators when working on interdisciplinary team projects. We are utilizing Max/MSP software, which is used by professional media artists, as a means to introduce M&S concepts to Art and Technology students, this creates the opportunity for these students to learn M&S using familiar medium in an environment where we embrace their creative exploration. On the other hand, the use of Max/MSP software provides Computer Science students with a unique perspective; they are working alongside creative media artists, using a visual data flow language. This is likely the first experience for CS students using a visual data flow language for creating multi-sensory projects, we anticipate the experience will encourage them to appreciate and integrate aesthetics of computing in their professional careers. We can also note that the goal of enhancing the creativity of CS students may be more of a challenge than the goal of teaching M&S concepts to Arts and Technology students. We enjoy the challenges and opportunity to cultivate and nurture this creative and technical intermixing of academic cultures.

As this is a new approach for teaching this M&S course, it is difficult to measure the success or impact on student career choices. A key question is whether a topic, like modeling and simulation, is being offered for professional development, vocational training, or as a fundamental area of knowledge required by all students, even those outside of engineering. We take the latter view where our aim the introduction to M&S as a disciplinary area. Thus, validation of the approach lay less with professional follow-through and more with an ability to convey modeling and simulation concepts to non-engineers and non-scientists who nevertheless need an understanding of M&S knowledge to be successful in areas that are already known to rely on this knowledge: such as computer music, animation, virtual reality, and game design and implementation.

### 6. ACKNOWLEDGMENTS

We thank the Arts and Technology Department and the Computer Science Department at the University of Texas at Dallas. We also thank the office of the Dean of Engineering and Computer Science for the EEF Fellowship sponsorship of the first author. We also would like to thank Joseph Rowe for permitting the use of his Petri net project to be included in this paper.

### 7. REFERENCES


