Department: Head
Editor: Name, xxxx@email

Artificial Intelligence and Mobile Programming Courses for a Video Game Development Program in Chile

Nicolas A. Barriga
Escuela de Ingeniería en Desarrollo de Videojuegos y Realidad Virtual, Facultad de Ingeniería, Universidad de Talca, Campus Talca, Chile. nbarriga@utalca.cl

Felipe Besoaín
Escuela de Ingeniería en Desarrollo de Videojuegos y Realidad Virtual, Facultad de Ingeniería, Universidad de Talca, Campus Talca, Chile. fbesoain@utalca.cl

***Abstract*— We present our curriculum design process for two third-year undergraduate courses: *Artificial Intelligence for Video Games* and *Mobile Device Programming*. These are part of a 4.5 year program in *Video Game Development and Virtual Reality Engineering*. We explore the range of possible content, usually aimed at traditional computer science or software engineering students or practitioners, and extract and adapt it to our particular program. Students developed high quality apps, achieved good standings in AI competitions, and some even published peer-reviewed articles. We believe the approach presented in this article is broadly applicable, but can be specially useful to instructors creating courses for uncommon and innovative programs.**

■ **IN RECENT YEARS** several attempts have been made at using video games to teach general Artificial Intelligence courses [1], [2]. However, there hasn't been much discussion on how to teach *Video Game AI*. Video game AI techniques, as used in industry, have some overlap with standard academic AI topics, such as best-first search. Still, traditional AI courses rarely teach the basic methods of video game AI, such as: decision making via finite-state machines (FSMs), behavior trees (BTs) or utility; hierarchical pathfinding; or procedural content generation (PCG).

The spread of information and communication technology (ICT) in the population has increasingly grown, in particular mobile phone technology. Smartphone technology continues to be the principal motor of growth for ICT. According to Global Mobile Market Report (newzoo.com), smartphone penetration in developed countries is around 70% to 80%, while in developing countries it is roughly between 25% and

50%, and growing steadily. Mobile devices are a key market for video games accounting for 45% of the USD$152Bn global video game revenue, and 76% of the USD$92Bn global mobile apps revenue. In this context, knowing how to develop mobile applications is an important skill to develop during this undergraduate program.

This work describes our approach for creating and teaching two undergraduate courses: 1) Artificial Intelligence for Video Games; and 2) Mobile Device Programming. The curriculum has been designed in a way that the second course reinforces the first, through projects and practical applications, where the students apply all the theoretical algorithms, learned in the first course, in the development of mobile applications. It draws inspiration from both academic and industry literature, as well as from direct communication with game AI professionals.

PROGRAM DESCRIPTION

The *Video Game Development and Virtual Reality Engineering* undergraduate program is aimed at students who want to develop their creativity and technological skills with applications in virtual environments.

The program presents a multidisciplinary graduate profile with the aim of providing students with the competencies needed for the creation of video games with an emphasis on informatics, software development and the application of information and communications technology to this area. The curriculum blends the fields of engineering and design to instruct professionals with scientific and creative competencies for working in the interactive entertainment and virtual simulation industry, developing entrepreneurship, and addressing diverse areas such as recreation, simulation, education, and training, among others. The graduates will have a solid foundation in basic design and computer sciences, with a modern, proactive focus.

The graduate's distinctive characteristic will be reflected in his/her entrepreneurship and leadership skills for executing innovative solutions using information technologies to develop videogames and applications, incorporating teamwork skills. Moreover, the graduate's multidisciplinary training will allow him/her to contribute in the development and management of creative industry and software projects, as well as to take on emerging technologies and new areas.

COURSE AUDIENCES AND SCOPE

The curriculum of the *Video Game Development and Virtual Reality Engineering* undergraduate program is 4.5 years long, divided into three blocks. First, two years of basic science and foundation courses of each discipline (design, computer science, general education, foreign Language), leading to an associate's degree. Second, two years of specialized and detailed instruction, complementing the first block, leading to a bachelor's degree. Lastly, with the final degree project, skills related to innovation and entrepreneurship are stressed, and a first professional degree is obtained. The program summary [3] shows pre-requisites for the courses described in this article.

Early in the conception of the *Video Game Development and Virtual Reality Engineering* program we decided that harmonization between courses would play a large role in the design process. We coordinated several small clusters of courses, such as the one presented in this article, as well as larger groups. Each block includes a workshop course with a high credit load and 7 to 8 hours in a laboratory, working in a project that ties all the competencies learned in the previous courses (see courses in yellow in [3]). For example, the first of these workshops (*2D Game Programming Workshop*) caps and integrates the knowledge, skills and competencies acquired in three product design courses, three programming courses and three video game development courses. This is also considered a checkpoint were professors can check if the students are developing the competencies and skills properly or if they need some support on their progress.

Students are also exposed to other early experiences of course articulation, such as making one game project for three courses in the same semester, as seen on the courses in green in [3]. In this case, each course contributes with their disciplinary competency: programming, concept design, videogames basics.

Taking these workshop and introductory courses as an example, we designed a coordinated process between *Artificial Intelligence for Video Games* and *Mobile Device Programming* courses, aimed at offering students the chance to apply and develop their competencies in a related area, going from theory to practice.

Artificial Intelligence for Video Games is a one semester, 6 European Credit Transfer and Accumulation System (ECTS) credits [4] (1 credit is around 25 to 30 hours) long course. Our students are in the fifth semester of the 4.5 years *Video Game*

Development and Virtual Reality Engineering undergraduate program. They have already taken several computer science courses, such as structured programming, object-oriented programming, databases, algorithms and data structures, as well as various video game development courses and workshops. *Mobile Device Programming*, a 4 ECTS credits course, is delivered the following semester.

Our aim is to cover AI techniques currently in use in video games (e.g, FSMs, BTs, A*, constructive PCG), newer methods that are slowly finding their way into commercial games (e.g., Monte-Carlo Tree Search, MCTS) and novel techniques that might be useful in the close future (e.g., machine learning and search-based PCG). Due to the expanse of the subject matter at hand, we can't cover every topic in detail. We will use a mix of guided programming activities (labs), solo programming assignments, research assignments, readings and lectures. The content deemed most relevant to the students' future careers will be taught in depth, using all four methods, while topics whose application to video games is more speculative will be given less attention (e.g., by only assigning a reading).

On the other hand, in the mobile devices course, we aim to study the growth of this technology and its applications for programming mobile applications. In this context, we work with Android OS due to its open-source license and SDK available to the developer community. The content goes from building a strong foundation in the Kotlin language (e.g, by resolving problems without a GUI, using only the core language), then understanding the Android OS and applications architecture, to programming apps that require access to sensors like accelerometers, GPS, light, etc. and services such as databases, Firebase among others. With this in mind, students build a portfolio of developed apps with different functionalities. They will later develop a final course project.

In the case of the two courses presented here, we have decided to coordinate the more theoretic artificial intelligence and the largely practical mobile devices courses by having the programming assignments in the latter use AI algorithms. We have settled on single- and two-player board games in a mobile platform, which allows us to include single-agent search, pathfinding and/or adversarial search algorithms.

CONTENT

In this section, we present a compilation of the most relevant content we considered, as well as the content we finally selected for each course.

Artificial Intelligence for Video Games

We consider content covered by game AI books targeted at industry professionals, books covering academic game AI research, as well as some discussions in the AI Game Programmers Guild mailing list. For details, see **Table 1**.

Millington and Funge's *Artificial Intelligence for Games* [5] is the *de facto* standard textbook for game AI professionals. It covers a wide range of topics in great detail, while focusing on the methods themselves, rather than on the applications.

Mat Buckland's *Programming Game AI by Example* [6] is much more application oriented. It starts from example games, and pulls in the necessary knowledge, leading to a more applied, but less systematic and detailed coverage of the topics.

Table 1. Topics covered in Video Game AI books.

Topic	Details
Artificial Intelligence for Games [5]	
Movement	Kinematics, steering, groups.
Pathfinding (PF)	Representation, search, hierarchical PF.
Decision making	FSMs, BTs, fuzzy logic, goal-driven, rule based, scripting.
Tactics and strategy	Waypoints, terrain analysis, tactical PF, coordination.
Learning	Hill-climbing, Machine Learning (ML), Reinf. Learning (RL).
Tree search	Minimax, optimizations.
Execution management	Scheduling, anytime algorithms, Level of Detail.
World interfacing	Polling, events.
Programming Game AI by Example [6]	
Movement	Steering, groups.
Decision making	FSMs, scripting, goal-driven agents, fuzzy logic.
Pathfinding	Representation, search, smoothing.
Artificial Intelligence and Games 7]	
Decision making	FSMs, BTs, utility, evolutionary algorithms (EA).
Tree search	Minimax, MCTS.
Learning	ML, RL.
PCG	Search-based, solver, grammar, cellular automata, noise and fractals, ML.
Modelling players	ML.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2020.2986758, Computing in Science & Engineering

A recent book coming from academia is Yannakakis and Togelius' *Artificial Intelligence and Games* [7]. Here, the focus is clearly on the methods, in particular newer methods coming from academia, most of which have not been used in commercial games. It is written as a survey of the subject, giving a broad overview, but glossing over the implementation details. It contains numerous references for those wanting to dig deeper.

We have not covered some closely related, but very subject specific books, such as Dave Mark's *Behavioral Mathematics for Game AI* [8] on utility-based AI, Brian Schwab's *AI Game Engine Programming* [9] and Shaker et al.'s *Procedural Content Generation in Games* [10].

Finally, we compiled our own personal communications with professional video game programmers, as well as discussions that have taken place in the AI Game Programmers Guild, a forum for industry professionals. Most developers agree on the broad topics of pathfinding, decision making and PCG, with some differences in specific algorithms. For example, A* and FSMs are seen as *old*—by some—and superseded by hierarchical pathfinding and BTs, while others prefer to include them, to build up from the basics. Adoption of other topics is more fragmented: movement and animation are seen as important, but they are usually covered in other courses in game programming programs; adversarial search and machine learning are considered niche areas, not part of the regular AI programmer's toolbox, but useful tools if you have them.

Selected AI Content

In this section we present the topics we have chosen to cover. We dismissed topics already seen by our students, such as movement and animation. We start with a review of basic search algorithms, since a working knowledge of them is crucial to later subjects. We then move to pathfinding and decision making, of which there is an agreement both in the literature and in the practitioner community. We then spend a significant amount of time on tree search algorithms. We believe that they have a big niche of current applications in board games, as well as potential new uses in the near future in card and strategy games. We finish the semester with an overview of PCG, both using traditional methods and emerging AI-based ones. A summary can be viewed in **Table 2**.

We believe this selection of content provides our students with a good balance of job-ready skills and a broad algorithmic understanding of the field.

Table 2. Video Game AI course topics.

Topic	Weeks	Details
Search	2	Trees. Graphs. Breadth-First Search. Depth-First Search. Dijkstra.
Pathfinding	3	Map representations. A*. JPS+.
Decision making	4	Decision trees. Finite-State Machines. Behavior Trees. Utility.
Tree Search	3	Game tree search. Search optimizations. Evaluation functions. Monte-Carlo Tree Search.
PCG	2	Traditional methods: constructive, fractals, noise, grammars. AI methods: search-based, ML.

AI Content Delivery

The AI content was delivered using a flipped classroom approach. The approach consisted of weekly readings−mostly chapters from the Game AI Pro [11] and AI Game Programming Wisdom [12] book series−followed by short questionnaires at the beginning of each class to assess students' understanding of the material. Then, the instructor would fill in any perceived gaps in knowledge and follow with group exercises. Lab time was spent, on alternating weeks, on guided programming tasks or individual programming assignments.

Mobile Device Programming

Similar to the AI for video games course, for mobile device programming we considered content covered by several Android development books targeted at academic and industry professionals, and also the official Android development website.

Dawn and David Griffiths' *Head First Android Development: A Brain-Friendly Guide* [13], is particularly interesting because of their approach in the *Head First* series that is based on cognitive science and learning theory, using a visually rich

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2020.2986758, Computing in Science & Engineering

format to engage the reader rather than a text-heavy approach. This approach proves beneficial to engaging students and enhancing their learning process, both inside and outside of the classroom.

Neil Smyth's *Android Studio 3.0 and 3.5 Development Essentials: Kotlin Edition* [14], [15] is a complete book that presents Android Studio and does the walkthrough to set up an Android development and testing environment. It continues with an introduction to programming in Kotlin, including data types, flow control, functions, lambdas, and object-oriented programming.

Ian F. Darwin's *Android Cookbook: Problems and Solutions for Android Developers* [16], introduces work with accelerometers and other Android sensors, the use of various gaming and animation frameworks, the storage and retrieval of persistent data in files and embedded databases, and the access of RESTful web services with JSON and other formats.

Android has had a considerable growth rate, from the very first versions to the last one, improving their API and frameworks, and also changing their main programming language from JAVA to Kotlin. Therefore, as a technical resource, the official Android development website is in fact the best source of up-to-date examples and descriptions of each concept, process or SDK related topic.

Selected Mobile Device Content

In this section, we present the topics we have chosen to cover. We dismissed topics already seen by our students, such as the object-oriented programming paradigm and UI design.

The course is planned as follows, with three main units: 1) Kotlin language essentials, which covers the technical aspect of the language with several analyses of different apps, smart context, and ubiquitous computing topics. 2) We start understanding the architecture of Android OS and its apps, activity lifecycle, and an incremental exploration from GUI, events, states, access to sensors, and external services. 3) The main approach in this unit is to explore a solution to a problem proposed by the student, or selected from instructor suggested topics, that can be addressed with a mobile app. We expect to complement and integrate all the previous competencies and skills. A summary is presented in **Table 3**.

Unit 1

This unit introduces the essentials to the new programming language, with the aim of solving technical problems. We finish with a project where students have to implement a game that requires an AI algorithm. These algorithms where previously taught, both theoretically and practically, and the goal here is to have a first approach to solving problems in console with Kotlin.

Table 3. Mobile Device Programming topics.

Topic	Weeks	Details
Unit 1	4	Kotlin essentials Control structures. Loop control. Data structures. Functions. MultiThreading. Project with AI algorithm.
Unit 2	6	Android essentials. Android Studio editor. Android architecture. Activity lifecycles GUI. Working with managers (sensors, preferences, etc). Working with Webservices. Working with storage. App portfolio.
Unit 3	4	Integration project. Integration project and refinements.

Unit 2

Since the technical aspects of Kotlin programming are covered, we start integrating mobile device specific aspects. We create different apps, incrementally adding new topics, with the aim of introducing a general approach to application development using different resources. Here, a second integration comes with the implementation of the first project, now in a mobile environment, where they need to take care of the GUI design, events, and states of activity, among other important aspects.

Unit 3

The last unit aims to serve as an integration arena for solving a problem that requires the use of external services. It allows students to refine their previously acquired knowledge while encouraging their autonomy.

The integration of knowledge and competencies is very important for courses to provide the basic skills for the creation of products that require a multidisciplinary point of view. Even though mobile apps can be developed by a single developer, professional development considers many aspects besides programming. Larger teams manage by pooling together people with different skill sets. Our program stresses this multidisciplinary aspect, by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCSE.2020.2986758, Computing in Science & Engineering

having students undertake User experience (UX) and design aspects as well as technical and algorithmic aspects.

RESULTS

In the AI course, a programming assignment was linked to external AI competitions at the IEEE Conference on Games (IEEE CoG). One of the groups placed in fifth position, out of twelve participants, in the µRTS competition. It is worth noting that their entry beat the previous year's winner, and that all higher placed entries were submitted by graduate students.

The same course had a research assignment, where students were asked to write a report on a topic of their choosing, related to video game AI. Two of those reports [17], [18] were later turned into exploratory surveys and accepted for presentation at the IEEE Chilecon 2019 conference. A third one is currently under review at a journal.

In the *Mobile Device Programming* course, students go on to develop several applications. In the first unit, they develop Kotlin console apps. **Figure 1** shows a Microrobots puzzle being solved using a single agent search or pathfinding algorithm from the AI course. The game consists on finding a path from a starting position to a destination, moving on cardinal directions, between squares with the same number and/or color.

On the second unit, they developed a full-fledged game application, Connect4, shown in **Figure 2**, which requires an adversarial tree search algorithm.

DISCUSSION

These syllabi are specifically tailored to our students. For students with a greater algorithmic knowledge, the first section of the AI course, on search can be skipped. Conversely, students new to game programming will probably benefit from a section on movement and animation.

If the program curriculum included a second AI course, we would move PCG and tree search there. That second course would also contain machine learning and reinforcement learning modules, as well as player modeling. The extra time in the first course would be used for expanding our coverage of pathfinding algorithms, specifically world representations, and adding single-agent planning algorithms to our decision-making module.

Figure 1. Example of the Microrobots console mobile application developed in Kotlin.

Figure 2. Example of a Connect 4 mobile application developed in the course.

Likewise, the mobile device programming course is also tailored to our students. The two main considerations were: 1) algorithmic knowledge and previous programming courses (structured and object-oriented OO); and 2) courses in the same semester, which are: *Operating Systems and Networking* and *Usability and Interfaces.*

The first unit focuses on Kotlin's essentials, considering our students already know the OO paradigm and JAVA language. If the first OO programming course was taught in Kotlin, we could probably skip this unit.

The second unit, which focuses on Android essentials, can be extended depending on the goals of the program. In this case, we aim for giving the students a strong basic foundation that allows them to keep learning and improving their skills and competencies autonomously. It would be very interesting to include topics such as Media resources, OpenStreetMap (OSM) and Google maps (in detail), Android instant apps, Google Play store publishing, or cloud storage, among others. But, by giving students the freedom to choose their own project topics, we

allow them to explore some of these topics or just deepen their knowledge of previous ones.

Finally, it is worth noting that the courses presented here are a good complement to our department's main research fields: ubiquitous and pervasive computing, mHealth, uHealth, gamification, user experience, video game development and video game AI. Even more to the point, these courses are a very good match for the authors' research topics, as evidenced by their latest papers on gamification of a mobile app for HIV prevention [19] and on using supervised learning, search algorithms and reinforcement learning for video game AI [20].

CONCLUSIONS

Our program covers the major sources of game AI content by professional practitioners along with academic researchers in the field. Where we found significant overlap, we added it to our syllabus, with some exceptions where our students had already covered the topics in previous courses. On subjects in which there was some disagreement, we have opted for content we expect will become more relevant to AI practitioners in the video game industry in the near future. We have also made suggestions on how to adapt this course to your own program curricula, by taking into account previous courses, or the possibility of more than one game AI course.

In the same way, for mobile devices, the course allows the students to create a strong foundation for developing mobile apps. But it's important to take into consideration the convergence of knowledge of all the previous and parallel courses, which allow explaining easily some important parts of the technical point of view of programming. Having as a tool the integration of previous topics (i.e. AI algorithms), allows the students to develop projects to familiarize themselves with a new language, while separating the technical challenges, from the algorithmic ones. Conversely, the earlier course can establish strong theoretical foundations without the need to develop full-fledged applications. Lastly, the students learn to iterate the same solutions, from algorithms, to basic implementations, to mobile applications, making mental connections from the computer science, software engineering, and product design aspects of application development.

We have found that the connection between subsequent courses is well received and culminates in course projects that are more advanced and compelling for both students and instructors. The course design presented in this paper has remained stable for the last two years. Due to its success, we plan on integrating further AI algorithms into the mobile device programming course for next year's offering.

REFERENCES

1. J. DeNero and D. Klein, "Teaching introductory artificial intelligence with pac-man," in Proceedings of the Symposium on Educational Advances in Artificial Intelligence, 2010, pp. 1–5.
2. A. Barella, S. Valero, and C. Carrascosa, "Jgomas: New approach to ai teaching," IEEE Transactions on education, vol. 52, no. 2, pp. 228–235, 2009.
3. F. Besoain, "Video Games Development and Virtual Reality Engineering program summary", https://doi.org/10.6084/m9.figshare.12000588.v1, 2020. .
4. Publications Office of the European Union, ECTS Users' Guide, European Union, 2015.
5. I. Millington and J. Funge, Artificial intelligence for games. CRC Press, 2009.
6. M. Buckland, Programming Game AI by Example, ser. Wordware Game Developers Library. Jones & Bartlett Learning, 2004.
7. G.N. Yannakakis and J. Togelius, Artificial Intelligence and Games. Springer, 2017.
8. D. Mark, Behavioral Mathematics for Game AI, ser. Applied Mathematics. Cengage Learning, 2009.
9. B. Schwab, AI Game Engine Programming, 2nd ed. Cengage Learning, 2008.
10. N. Shaker, J. Togelius, and M.J. Nelson, Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer, 2016.
11. S. Rabin. Game AI Pro: collected wisdom of game AI professionals. AK Peters/CRC Press, 2013-2017.
12. S. Rabin. AI Game Programming Wisdom. Charles River Media, 2002-2008.
13. D. Griffiths and D. Griffiths. Head First Android Development: A Brain-Friendly Guide, 2nd ed. O'Reilly Media, 2017.
14. N. Smyth, Android Studio 3.0 Development Essentials - Kotlin Edition. Payload Media, Inc., 2017.
15. ——, Android Studio 3.5 Development Essentials -Kotlin Edition. Payload Media, Inc., 2019.
16. I.F. Darwin, Android Cookbook: Problems and Solutions for Android Developers. O'Reilly Media, 2017.
17. G.K. Sepulveda, F. Besoain, and N.A. Barriga, "Exploring dynamic difficulty adjustment in videogames," In IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies, 2019.
18. I. Gajardo, F. Besoain, and N.A. Barriga, "Introduction to behavior algorithms for fighting games," In IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies, 2019.
19. F. Besoain, A. Perez-Navarro, C. Jaques Aviñó, J.A. Caylà, N.A. Barriga, P. Garcia de Olalla. UBESAFE: prevention of HIV and others STI by geofencing and contextualized messages with a gamified App. JMIR mHealth and uHealth.
20. N.A. Barriga, M. Stanescu, F. Besoain, M. Buro. Improving RTS Game AI by Supervised Policy Learning, Tactical Search, and Deep Reinforcement Learning IEEE Computational Intelligence Magazine 14 (3), 8-18.

Nicolas A. Barriga is a professor with the School of Video Game Development and Virtual Reality Engineering at Universidad de Talca, Chile. He obtained his Ph.D. at the University of Alberta, Canada, for his work on state and action abstraction mechanisms for RTS games. His current research interests are in the broad area of video game AI.

Felipe Besoain is a professor with the Bioinformatics Department and the School of Video Game Development and Virtual Reality Engineering at Universidad de Talca, Chile. He obtained his Ph.D. Universidad Oberta de Catalunya, Barcelona, Spain, for his work in mobile devices and ubiquitous computing. His research interests are the application of information technologies in intelligent contexts for bioinformatics, agronomy and health areas; and the use of ubiquitous computing and mobile devices, gamification and virtual reality applied to mHealth.