Teaching Game AI as an Undergraduate Course in Computational Media

Adam M. Smith, Daniel Shapiro
Department of Computational Media
University of California Santa Cruz
1165 High St. SOE 3, Santa Cruz, California 95064
{amsmith,dgs}@ucsc.edu

Abstract

We need to teach AI to students in and outside of traditional computer science degree programs, including those designer-engineer hybrid students who will design and implement games or engage in technical games research later. The need to rethink AI curriculum is pressing in a design education context because AI powers many emerging practical techniques such as drama management, procedural content generation, player modeling, and machine playtesting. In this paper, we describe a 5-year experimental effort to teach a Game AI course structured around a broad and expanding set of roles AI can play in game design (e.g., Adversary and Actor, as well as Design Assistant and Storyteller). This course sets up computer science and computer game design students to transform practices in the game industry as well as create new forms of media that were previously unreachable. Our students gained mastery over the relevant techniques and further demonstrated (via novel prototype systems) many new roles for AI along the way.

Introduction

As AI technology matures, there is an increasing need to consider how engineered systems integrate with human experiences. However, these topics are typically pursued by people with distinct interests and backgrounds. Students with design interests are often ill-prepared to work with complex computational systems, while students with technical backgrounds are similarly ill-prepared to work in areas where human interaction and experience are key factors. In the practice of game design, this concern is unavoidable as the projects require communication between technical and artistic contributors. As a result of this tension between disciplines, AI has so far been narrowly applied in the gaming industry. This situation creates an important opportunity for education to develop design-literate engineers and engineering-capable designers.

In 2014, UC Santa Cruz established a new Department of Computational Media to serve as an academic home for a new interdisciplinary field that is concerned with computation as a medium for creative expression (Stephens 2014). This department hosts the computer game design degree program which was previously administered through the Department of Computer Science. The background and interests of students in this program differ from that of other STEM disciplines in that they directly connect to the arts and social sciences via their focus on human experiences. With the establishment of the new department, there was a need to reboot several existing computer science classes under the banner of computational media.

This experience report describes a 5-year effort in teaching an upper-division artificial intelligence course in the context of an interdisciplinary technical degree program. We contribute a new course design including programming and reading assignment suggestions, evaluated by the diversity of student-initiated final projects. We enumerate several roles for AI in interactive systems including many that fall outside the range of traditional AI textbooks, and we show that students can be engaged in the process of inventing and demonstrating additional roles for AI themselves. This demonstrates the possibility to teach a complex engineering subject to a mixed audience of undergraduates by adopting an applications perspective.

Background

Our new Game AI course design succeeds an earlier design, first offered in 2006, that primarily emphasized coverage of concepts and practices in use by the game industry. With the establishment of the new department, the Game AI course was pulled in several directions: it could continue to distill the industry sense of Game AI, it could introduce students to the role of AI in technical games research (a focus area amongst the faculty of the new department), or by default it could follow the outlines offered by textbooks of Game AI.

AI in the Gaming Industry

The premier industry conference for video game developers is the Game Developers Conference (GDC). The kinds of knowledge shared at this conference are very broad, ranging from technical to artistic and business-related. The conference regularly hosts an event called the GDC AI Summit focused specifically on current and emerging applications of AI game development. The summit is organized by (and

has an attending audience largely overlapping with) the AI Game Programmers Guild. These professional bodies define one viewpoint on Game AI: Game AI should be concerned with whatever specialized knowledge AI game programmers need to do their job or is shared on online forums for the industry like Gamasutra.

Although this community is only very loosely connected to the world of academic AI research, there is a long history of academic ideas making their way into game projects and for ideas originally developed in an industry context to be documented and disseminated in the academic literature. The game *F.E.A.R.* (Monolith Productions 2005) (and many more including *Deus Ex: Human Revolution* (Eidos Montréal 2011)) makes use of Goal-Oriented Action Planning (GOAP), a STRIPS-like planning architecture (Nilsson 1998), to control non-player characters. *Halo 2* (Bungie 2004) employed behavior trees, a reactive/dynamic planning architecture (Isla 2005), for a similar purpose. The game *Left 4 Dead* (Valve South 2008) included a "Director" system for drama management, the practice of procedurally manipulating the game world to influence the player's experience without overtly controlling the game's non-player characters. Other systems for technical authoring of a game's narrative, such as the Versu system underlying *Blood & Laurels* (Short 2014), are now documented in academic AI journals. The use of procedural content generation, such as for generating the vast in-game worlds of *Dwarf Fortress* (Bay 12 Games 2006), *Minecraft* (Mojang 2011), or *No Man's Sky* (Hello Games 2016), is now considered an AI application by the academic community (Yannakakis and Togelius 2018b). Systems for player intent recognition, such as those in *Bioshock Infinite* (Irrational Games 2013), have clear connections to academic AI research on goal recognition. The categories of academic AI applied in games and readable-as-intelligent systems used in commercial games that are then accepted by the academic community are co-evolving.

Technical Games Research

Technical games research is an emerging topic spanning existing fields of computing including artificial intelligence, human-computer interaction, computer graphics, and others (Nelson 2019). Technical games work related to AI is often presented at the AAAI Conference on Artificial Intelligence in Interactive Digital Entertainment (AIIDE) or the IEEE Conference on Computational Intelligence in Games (CIG, recently renamed to the IEEE Conference on Games). It is archived in journals such as the IEEE Transactions on Computational Intelligence and AI in Games (TCIAIG, recently renamed to the IEEE Transactions on Games). Additionally, AI-related work is regularly presented at cross-disciplinary conferences such as the International Conference on Digital Storytelling (ICIDS) or International Conference on the Foundations of Digital Games (FDG).

An alternate view of Game AI emerges from technical games research: Game AI should be concerned with whatever specialized knowledge is used by these academics and disseminated in their peer-reviewed publication venues. This view is distinct from the general sense of AI captured by the AAAI Conference on Artificial Intelligence or the Journal of AI Research which, while very broad, usually would not consider drama management, procedural content generation, player modeling, and machine playtesting.

Textbooks of Game AI

Around the world, several universities offer Game AI classes that are based on a foundational distinction between (industry-based) Game AI and Academic AI. While textbooks such as Russell & Norvig's well-known *Artificial Intelligence: A Modern Approach* (Russell and Norvig 2016) often anchor general AI courses in computer science programs, Funge's *Artificial Intelligence for Computer Games: An Introduction* (Funge 2004) is often used to anchor Game AI courses (usually at institutions offering technical degree programs specifically related to game design). Indeed, the Game AI versus Academic AI distinction is introduced in the book's first chapter. Upon completing a course based on Funge's book, students should be able to understand and apply the ideas documented in industry-oriented books such as the *AI Game Programming Wisdom* series (Rabin 2002).

Since the emergence of technical games research (largely taking place after Funge's 2006 distillation of Game AI), there is a clear need for textbooks that cover topics relevant to future AI-oriented technical games researchers. In 2018, Yannakakis & Togelius published the textbook *Artificial Intelligence and Games* (Yannakakis and Togelius 2018a), targeting both graduate and undergraduate students and closely aligning with the slice of technical games research usually considered at the CIG conference and in the TCAIG journal. This book covers the use of games as configurable testbeds for academic AI technologies (usually in the form of game playing agents exemplified by AlphaZero (Silver et al. 2017)) but also as a domain with its own unique challenges. The book introduces procedural content generation and player modeling as peers to the topic of automatically playing games. Although this book also opens with a discussion of the "gap" between Game AI and Academic AI, it suggests this gap is the manifestation of a temporary historical situation before the interest in AI in games broadened beyond controlling non-player characters. In a section called "What We (Don't) Cover in This Book" the authors intentionally leave certain AI topics like planning to academic AI books (like Russel & Norvig's) and others like pathfinding to industry-oriented Game AI books (like Millington & Funge's). For these authors, post-gap AI in Games is not simply the intersection or union of the categories identified earlier. They identify further AI topics popular in technical games research, including computational narrative, but leave these out to control the scope of their book.

Our Game AI course takes the perspective that neither of these approaches (that might be caricatured as preparing future AI programmers for the game industry or future AI researchers in technical games research) is entirely appropriate for our undergraduates in a department of Computational Media. Instead, we propose that a core topic for AI in games is inventing and demonstrating new roles for AI in games. Our course did not employ one specific textbook but instead assigned readings, game playing, and original research papers instead. We intend that our students, whether in industry

or in academia, continually identify new applications of AI in game design, development, run-time execution, or post-deployment analysis.

Course Design

Our Game AI course introduces AI in the context of a newly formed department of Computational Media, whose goal (roughly stated) is to train a generation of design-sensitive engineers and engineering-capable designers. It is a quarter-length (11 week long) class, with a student body consisting of Juniors and Seniors with backgrounds in digital media, arts, computer science, and games. The composition is typically half students from our computer game design degree program and half students from the computer science and engineering degree program (both housed within our School of Engineering). This mixed background presents a challenge for communicating technical material, but also a tremendous opportunity to coordinate students on creative projects that take advantage of their diverse skills. The course has only one prerequisite: completion of the upper-division Algorithms and Abstract Data Types course. No previous exposure to AI in previous courses is assumed, and some students enroll in our institution's upper-division Artificial Intelligence course independently of Game AI.

We established the following learning objectives for the course: (1) students can apply and modify common Game AI techniques in programming tasks; (2) students understand that AI can be employed in a diverse set of roles within games; and (3) students can invent new roles for AI in Games, and demonstrate them in novel prototypes.

The first objective ties the desired level of mastery to the background of the student body. It defines the goal as giving students the power to apply AI techniques in games, versus the ability to implement the technique or understand the surrounding theory. The second objective clarifies a difference between this course and the way Game AI has traditionally been taught; rather than focus on using AI to create smart non-player characters (NPCs), we adopt a mission to broaden students' conception of the roles AI can perform. This stance clearly reflects a bias on our part that is partially, but not completely supported by the current state of the art in the game industry, but it's heartfelt, and consistent with the academic objective of improving the landscape of games and interactive experiences in the future. Our third objective emphasizes the creative use of AI techniques, which is essential in the context of a design-oriented curriculum.

We refined these learning objectives into a set of *themes* that structured course content: Core AI Metaphors; AI as "X" in Games (where X is a role for AI); Examining the AI in Existing Games; and Creatively Applying AI in Games.

For purposes of an applications-oriented class, we defined **Core AI Metaphors** as AI technologies and algorithms with implementations available on the web that are most easily applicable in games. The selection involves something of a judgment call, and the ordering is also important as the students draw on the palette of methods presented in the first half of the course when framing their creative projects. Lectures described algorithms at the pseudo-code level, and worked examples primarily in game settings. We covered forward search (breadth first and depth-first search, Dijkstra's forward search, greedy best first search, A*), constraint satisfaction, Monte Carlo Tree Search, finite and hierarchical finite state machines, rule application, behavior trees, reactive planning languages, goal oriented action planning, utility models, genetic algorithms, decision tree learning, perceptron learning, steering behaviors, answer set programming, deep learning for classification, and deep reinforcement learning, with all but two or three discussed prior to the start of student projects.

We presented a variety of **roles** for the use of AI in games. These included AI as: Actor (agents that act in a way that is readable as intentioned behavior consistent with a fictional characterization); Author (agents that configure settings, plots, and characters to assemble a plausible narrative); Design Assistant (agents that provide feedback on human-authored design elements, often via simulation); Designer (agents that generate design elements that would otherwise be hand-authored); Drama Manager (agents that adapt continuously the game world to shape the game's narrative and player experience); Adversary/Villain (agents that either play to win or to lose under precisely described conditions); Student/Learner (agents that continually grow or adapt in response to player feedback); and Referee (agents in the same role of game masters in table-top games).

Many of these themes are illustrated in **commercially successful AAA and Indie games** (such as in those examples given in the Background section above). Overall, the coverage of AI themes in existing games is somewhat sparse, and this fact was one of our motivations for composing the course. We believe that AI represents a largely unexploited resource in game design, and this course is a practical step in addressing that gap.

We used the theme of unpacking the AI in published games to ground our technology lectures in concrete settings. We found that students were tremendously motivated by these examples, as they unveiled mysteries behind their own game playing experiences and showed depth behind games they had not previously considered.

The source material for these lectures was somewhat hard to obtain as companies are not necessarily motivated to reveal their underlying technology. *Postmortem* talks (where designers explained what worked and what did not after their game had been commercially released) were a great resource here, as were occasional debriefs by game authors on the web or as guest speakers.

The final theme called for students to actively **employ AI in games in a creative fashion**. This goal was the source of the project-oriented structure of the class, and it dictated the broad syllabus of the class. In particular, we needed to communicate AI metaphors in roughly the first half of the term in order to leave time for a substantial creative project based on those techniques in the last 4-5 weeks of the quarter. This divided the lecture content into a closely guided tour through AI techniques, followed by a more open presentation of interest topics. Most guest lecturers spoke during this period. This schedule also impacted the assignment structure, with multiple readings and 6 weekly programming tasks starting on day 1 of the class, but minimal readings and no program-

ming assignments later.

We employed programming assignments to communicate selected AI roles in detail (described later). Students performed each of these assignments in two-member teams. This decision served multiple goals; it let us create more valuable (but more challenging) assignments, it let us scope projects to enable contributions by both the technology-focused and design-oriented students in the class, it fostered peer-instruction, it reflects our belief that pair-coding is easier than solo work, and it reflects the reality that game development in industry is invariably conducted in teams, which makes teamwork a necessary skill for undergraduates in a game-oriented curriculum. We encountered some resistance to this team-coding requirement. The amount varied from very little to moderate grumbling during the term, but the policy received positive student summations at the end. Overall, we attempted to thermalize inequalities among teams by requiring that students not pair with someone else they had already paired with in recent assignments.

As mentioned earlier, we employed on-line readings vs textbooks as primary course materials. Some readings were very broad and covered high level themes, e.g., from the "Turing Tantrums: AI Devs Rant" session at the GDC AI Summit (Sunshine-Hill 2015) or selections from the "Photoshop of AI" debate (Mark 2009) in various venues, while others were technology summaries.

The detailed course structure consisted of 6-7 weekly programming assignments (done in teams), about 15 readings with associated questionnaires, a midterm in some years, and a major creative project (performed in teams of 3-4) in the last 4-5 weeks of the class. Their relative weights for grading were 20% reading, 50% programming, and 30% project in years without a midterm, and 10% reading, 30% programming, 30% midterm, and 30% project otherwise.

Programming Assignments

This section reviews the breadth of programming assignments utilized in our offerings of the course. Not every assignment was utilized every year and not every assignment is listed here. Most assignments built on instructor-provided base code and downloadable packages for the Python language. Where the previous design for this course emphasized from-scratch C++ programming to orient students to industry practices, our new design emphasized finding and building on top of existing software infrastructure to let students quickly demonstrate new roles for AI in games.

Navmesh Pathfinding: Navigation meshes (Tozour and S. Austin 2003) are a Game AI technique for abstractly representing the traversable spaces in a game world. Just after a (video) reading assignment on how to use a new algorithm to outperform the traditional A* graph search algorithm (Rabin 2015a), we asked students to demonstrate their own massive speedups over A* using an alternate world representation. Given a 4,096 by 4,096 pixel dungeon map, this assignment required students to find paths between interactively selected waypoints with pixel-level precision. Students implemented variations of A* (e.g. including bidirectionality) that worked on a rectangular decomposition of the free space in the input dungeon map where the cost of traversing a rectangle

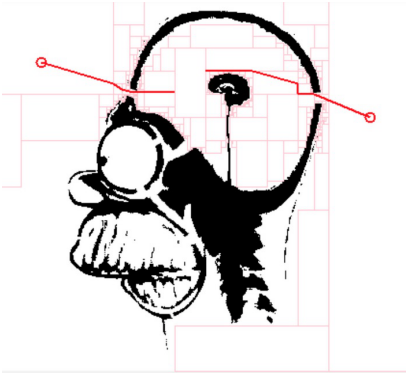

Figure 1: A student-selected test map for the navmesh pathfinding assignment. Students were required to generate their own debugging visualizations, as such visualizations are a common topic at the GDC AI Summit.

depended on the dynamically chosen enter and exit points on the perimeter. Students were also asked to find or design their own map to demonstrate edge cases of their implementations (Figure 1). The base code implemented the recursive subdivision logic as this assignment emphasized pathfinding with navmeshes rather than navmesh generation.

Ultimate TicTacToe: We used the game of *Ultimate TicTacToe* to teach Monte Carlo Tree Search (Browne et al. 2012). Unlike its progenitor, this game is hard enough to engage the adult mind; play occurs on a 3x3 grid of TicTacToe boards, where a move in square i, j of any board forces the next player to choose a move on board i, j if possible. The game is won by securing a three-in-a-row line of wins on the 3x3 grid. Students were given an Ultimate TicTacToe implementation with appropriate operations, an MCTS node abstraction, and two game-playing bots (one that selects random moves, and another that samples x games for every available move, plays randomly, and returns the move with the highest expected turnout. Given these tools, students were asked to implement two versions of MCTS and conduct source-of-power experiments. They assessed the impact of tree size on win rate for a random-rollout player, and the impact of heuristic vs random rollout on win rate given a maximum tree size (with and without a time bound). Ultimate TicTacToe is also sometimes used in AI courses that do not have a games focus (Neller et al. 2019).

MiniRTS: We implemented a miniature real-time strategy game and asked students to implement controllers for two kinds of non-player characters. In contrast to the MicroRTS framework (Ontanón 2013) used in previous academic game playing AI competitions (turn-based, with complex economic mechanics), this game emphasized the role of AI as actor. The student-created non-player characters needed to demonstrate a modest (design-appropriate) level of autonomy while being highly responsive to player commands and being readable as fictional workers and warriors. Although paired readings primed students to think about finite state machine (FSM) behavior representations, this assignment encouraged students to create a functional

but tangled mess of if-else blocks with ad-hoc conditions. We wanted students to understand the tension between expressive-and-messy and clean-but-restrictive forms of behavior authoring seen in Game AI practice.

Planet Wars: We assigned a 1-week programming exercise with Behavior Trees set in Google Planet Wars, an interstellar conflict simulation with mechanics for starship generation, conflict resolution, and travel time between planets. We supplied a behavior tree implementation, a game interface, plus five game-playing bots, and asked students to compose a single behavior tree that won against all five bots on a randomly chosen set of star maps. We also ran a round-robin tournament among student entries, which students found quite motivating (we also gave extra credit to the winning team). We observed that simple/bold strategies tended to win over intricate casing logic.

Minecraft Crafting Planner: In Game AI, graph search algorithms (e.g. A*) are almost always used to search two or three-dimensional spaces for the purposes of navigation. To break this association and open up student minds to new roles for AI, we asked students to repurpose their earlier graph search code to produce action plans. Inspired by the crafting mechanics of the game Minecraft (a game popular amongst students in which the player can, for example, convert three units of Planks and two units of Sticks into one unit of Wooden Pickaxe in the presence of a Crafting Table), students implemented a state-space planning approach where states were represented by resource inventory tuples (how many units of which kinds of items the player possessed). Students were given a file defining the vocabulary of items and available crafting recipes (a planning domain) as well as a sequence of planning problems (an initial inventory state and a set of minimum items counts for goal states). In order to be able to complete the full sequence of challenge problems within a time limit, students needed to use domain knowledge in order to implement heuristics, state pruning operators, or operator dependency analysis. Students were encouraged but not required to implement optimal planners (which minimized the sum of per-recipe action costs).

Instant Feedback Level Design Tool: Paired with readings on mixed-initiative level design tools emerging from the technical games research literature (Smith, Whitehead, and Mateas 2010; Butler et al. 2013), students were asked to implement AI for the Design Assistant role (Figure 2). Given base code that implemented a simplified paint program, students again adapted their graph search code to quickly compute reachability of every point in the game's world to the initial state each time the user painted a new detail onto the map. Inspired by the Metroidvania genre of games, the player's movement ability in this game incrementally unlocks as they collect more special items (requiring traversing the same geometric space multiple times with different inventory states). A key design concern for Metroidvania games is *sequence breaking* – can the player reach certain locations too soon by collecting special abilities in an unexpected order? Although planning was still used to find paths for a hypothetical player in this assignment, it emphasized a design-time role for AI in the context of game that might not utilize any path-planning at run-time.

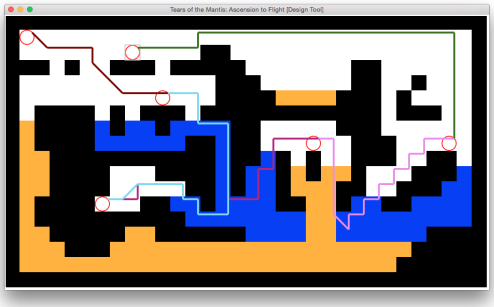

Figure 2: Reachability feedback in the level design tool.

Evolutionary Platformer Level Generation: We used a level design task set in the Super Mario Bros game to teach genetic algorithms. We provided students with a skeleton genetic algorithm implementation, a game interface, helper functions and a base fitness metric, and two genome representations ; a fixed length grid encoding of a Mario level, (Summerville et al. 2016) and a variable-length sequence of design elements (Sorenson, Pasquier, and DiPaola 2011) like a hole of width w in the ground plane, or ascending/descending stairs with some height and direction. We asked students to implement their choice of crossover mutation and successor population generation functions, refine the base fitness metric per their own criteria, and employ both representations to evolve levels. Students submitted their favorite level from each encoding, with the added instruction that they should be playable.

Machine learning (including the use of neural networks) is playing an increasing role in AI-related technical games research, e.g., for machine playtesting (Keehl and Smith 2019) and content generation (Volz et al. 2018). However, it was not represented in our programming assignments, and there is little[1] industry adoption of machine learning in Game AI. Nevertheless, we feel that future iterations of this course should demonstrate a role for machine learning, particularly where it might seem unexpected, such as in narrative generation (Ammanabrolu et al. 2019).

Reading Assignments

Our reading assignments were highly varied in their sources and formats. We sampled chapters from Game AI textbooks (reviewed earlier), from industry-oriented books in the *Game AI Programming Wisdom* (Rabin 2002) and *Game AI Pro* (Rabin 2015b) series, from postings on Gamasutra, from videos in the GDC Vault,[2] or from independent websites featuring key interactive explanations (Patel 2014). Depending on the scheduling of the class offering relative to the annual Game Developers Conference, it was sometimes possible to create reading assignments on extremely fresh material (emphasizing the importance of taking an expansive view on Game AI over mastering any one technique).

[1] *Black and White* is a notable exception (Wexler 2002).
[2] https://www.gdcvault.com/

After reading (or watching) the assigned source material, students were required to complete a brief response assignment. The format of this assignment varied between offerings, sometimes including multiple choice questions, short answer questions, or short essay questions. Some anonymized responses to readings would be discussed in the following lecture to clarify misconceptions or to encourage students to share their own stories (acknowledging students diverse background experiences as a source of knowledge).

Guest Lectures

In keeping with the applications focus of the class we maintained an active program of guest lecturers. We invited 2-5 leading AI game developers and researchers each quarter to speak on AI technology and its role in games they had worked on. Guest lecturers included originators of AI technology (steering behaviors), game-AI experts (e.g., Deep Learning, predictive analytics, Answer Set Programming, Procedural Content Generation), AI technology advocates (utility models), and lead AI designers for AAA games (the Sims series). The list varied depending upon whom we knew and could elbow into service, but the inclusion of guest lecturers with a tie to industry strongly affected the texture of the class, and directly addressed the theme of unpacking the AI in existing games.

Creative Projects

We designed the Game AI class to communicate a vision about the role AI can play in games to an audience of Juniors and Seniors with mixed interests in arts and technology. We framed the course as an experiment, both to ourselves and to our students, and now that we have taught it 5 times (once per year for 5 years) we are in a position to examine the results and extract lessons learned. This section evaluates our experience relative to our key learning objectives (listed in the Course Design section above).

The most visible evidence for student mastery of these lessons comes from the final creative projects, which were 4-5 week long tasks, done in teams of 3-5, to creatively apply AI in any aspect of game design, development, play, or evaluation. We have 102 student projects to draw on for this analysis, and we report outcomes in three different ways.

Figure 3 identifies the AI technologies incorporated into student projects. This chart assigns 1 or at most 2 labels to each project that reflect the core AI technology it employed. Students overwhelmingly chose technologies we covered in class; with the exception of the 10 (of 102) projects categorized as *other*, every label in the figure corresponds to a topic we examined in lecture or programming assignments.

The chart clearly shows that students applied a wide variety of AI technologies. Students typically implemented simple algorithms from scratch (like graph search methods), but chose to download and modify packages for more complex techniques instead (like constraint satisfaction). For example, behavior tree projects commonly introduced a new node type and encoded novel character behavior in the result, while goal oriented action planning projects frequently inserted domain specific selection heuristics into a planner

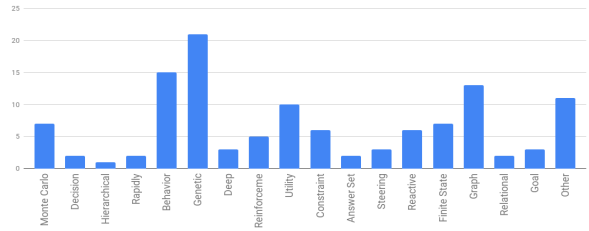

Figure 3: Distribution of AI *technologies* in final projects.

downloaded from the web. Given the breadth of AI mechanisms represented in Figure 3, we conclude that our students collectively demonstrate learning objective #1 - they developed sufficient mastery to apply and modify a variety of AI techniques in programming tasks. Recall that completing other AI courses was neither required nor common for these students.

It is also interesting to examine which techniques most often inspired student projects. Genetic algorithm and behavior tree projects were the most popular, while there are noticeable spikes associated with graph search, utility models, machine learning (collectively), and Monte Carlo Tree Search. From our experience the governing factors are sexiness, familiarity, and accessibility. For example, the popularity of genetic algorithms and behavior trees is explained by the fact that both have native appeal, both were the subject of week-long programming assignments, and both are easy to implement and use. Graph search (the third most popular project focus) is possibly more mundane from a student perspective, but the algorithms are simple and students utilized the concepts in multiple programming assignments. Students also submitted 8 machine learning projects; they clearly found the topic exciting, but it was only discussed in lecture and the techniques have a somewhat higher barrier to entry. Monte Carlo Tree Search has a similar, mixed profile; it is exciting for its success and applicability to games, it was the subject of a programming assignment, but students found it intellectually difficult to absorb. Finally, we note that guest speakers had an effect – in years where a prominent expert on utility theory spoke to the class, there was a strong uptick in utility-theory related projects.

Figure 4 categorizes the same set of student projects by the purpose they pursue using AI. These labels are admittedly subjective. Given that there is no accepted ontology of game purposes or tasks, we chose terms that covered the subject matter of student projects but that would also be familiar to game developers and technical game researchers. This chart illustrates the diversity of student projects. First, they apply AI to topics that span game design, game play, and testing or evaluation. The applications within game design employed AI to construct and modify multiple game elements, such as game music, character avatars, game objects (weapons, furniture), game landscape (both locally, and of entire game worlds), and game levels (their physical layout and contents). Projects emphasizing game play frequently employed AI to encode behaviors on background charac-

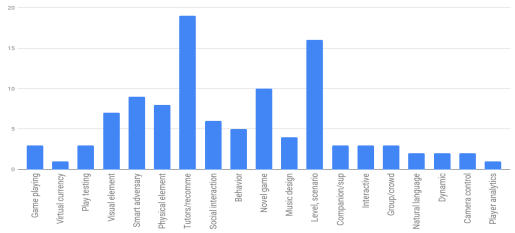

Figure 4: Distribution of the *purpose* of AI in final projects.

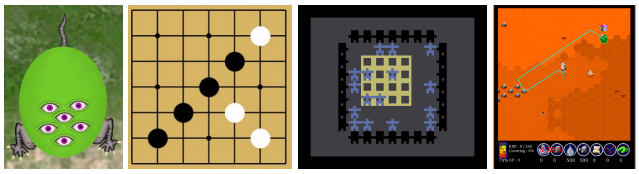

Figure 5: Screenshots of selected final projects.

ters and crowds, companions and support characters, and (of course) smart adversaries, including competition bots. Students also employed AI to enable tutorial and recommendation systems that provided advice on next actions tailored to the player's task, skill level, equipment, or team composition. Relatively few projects applied AI to game evaluation, but these included automated play testers and off-line analyzers correlating player stats with game success. This diversity demonstrates that students mastered learning objective #2; they understood that AI can be employed in a diverse set of roles within games.

Students also applied AI technology in creative ways. For example, a number of projects employed graph search or constraint satisfaction to generate game elements that met requirements, such as solvable mazes and puzzles, or levels with a known difficulty. Students also employed genetic algorithms to meet constraints, for example, to populate environments with creatures adapted to external conditions. Multiple projects focused on reducing authoring effort, for example, to design environments that incorporate aesthetics via player contributions to a fitness function, or to place behaviors on characters by searching across behavior tree structures that achieve a task. Some projects employed AI at a more abstract level to produce a novel game experience, for example, to maximize the feeling of horror by dynamically reconfiguring maze structures, to change a player's time sense by enabling 50 year moves, or to provide more flexible game play through generative narrative. We conclude that students collectively achieved learning objective #3: they were able to invent new roles for AI in games and demonstrate them in novel prototypes, often in novel *AI-based game designs* (Eladhari et al. 2011).

While figures 3 and 4 portray the breadth of student projects and their applications of AI, we can illustrate the depth by discussing a few examples in detail.

Figure 5 (first item) is drawn from a student presentation on a God-game enabled by an application of genetic algorithms. Here, the player manipulates the temperature and rainfall to create biomes, while the creatures evolve until they are well-adapted. The students modified a genetic algorithm with operations for migration and speciation to produce a novel game mechanic. This project demonstrated a lovely merger of technical capability and design sensitivity.

Figure 5 (second item) illustrates a project on tutorial generation for novice-expert game play. The students implemented a bootstrapped deep learning algorithm inspired by AlphaGo Zero that acquired strategies for Go-Moku (5 in a row). Then, they extracted board positions from the MCTS tree with highly divergent success rates dependent upon the move chosen, and presented these as tutorial examples. This combination of deep learning and analysis was a notable technical achievement for undergraduate students.

Figure 5 (third item) shows a Puzzlescript game being automatically playtested by an MCTS-based agent. Students in this project team adapted MCTS code from a programming assignment to be able to automatically play any Puzzlescript game in the wild. This system demonstrated the general game playing abilities and realized them in the form of a Design Assistant role. The tool could generate an animation of how to solve a given puzzle, and those animations could be regenerated after incremental level design or rule design changes. This undergraduate project was a direct follow-up to work in the technical games research literature on Puzzlescript design assistance (Lim and Harrell 2014).

Finally, Figure 5 (fourth item) shows a dynamic tutorial system based on both spatial and abstract planning. Students in this team extended a game that they had created for an earlier game design class by using adaptive AI to replace a fixed tutorial mode of the game. Connecting to the idea of drama management, if the player had not demonstrated a significant game action within a certain timeout, the system would automatically construct and visualize a plan to demonstrate the next simplest game mechanic that the player had not previously exercised (often a crafting recipe). Again, this undergraduate team project was a direct follow-up to academic work on automatically generating instructional scaffolding for educational games (O'Rourke et al. 2015).

Conclusion

This experience report contributes the design of a new Game AI course. Rather than simply servicing industry needs or preparing students for academic research, our course engages students in the process of inventing and demonstrating new roles for AI in games and other interactive media. Through review of students' final creative projects, we have shown how the course was able to prepare students to apply and modify common Game AI techniques, understand that AI can be employed in a diverse set of roles within games, and invent new roles for AI in Games while demonstrating them in novel implemented prototypes.

References

Ammanabrolu, P.; Tien, E.; Cheung, W.; Luo, Z.; Ma, W.; Martin, L.; and Riedl, M. 2019. Guided neural language generation for automated storytelling. In *Proceedings of the*

Second Workshop on Storytelling, 46–55. Florence, Italy: Association for Computational Linguistics.

Bay 12 Games. 2006. Dwarf fortress.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4(1):1–43.

Bungie. 2004. Halo 2.

Butler, E.; Smith, A. M.; Liu, Y.-E.; and Popovic, Z. 2013. A mixed-initiative tool for designing level progressions in games. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, 377–386. ACM.

Eidos Montréal. 2011. Deus ex: Human revolution.

Eladhari, M. P.; Sullivan, A.; Smith, G.; and McCoy, J. 2011. AI-based game design: Enabling new playable experiences. Technical Report UCSC-SOE-11-27, UC Santa Cruz Baskin School of Engineering.

Funge, J. D. 2004. *Artificial intelligence for computer games: an introduction*. AK Peters/CRC Press.

Hello Games. 2016. No man's sky.

Irrational Games. 2013. BioShock Infinite.

Isla, D. 2005. Dude, where's my warthog: From pathfinding to general spatial competence. Invited talk, Artificial Intelligence and Interactive Digital Entertainment (AIIDE).

Keehl, O., and Smith, A. M. 2019. Monster carlo 2: Integrating learning and tree search for machine playtesting. In *2018 IEEE Conference on Games (COG)*. IEEE.

Lim, C.-U., and Harrell, D. F. 2014. An approach to general videogame evaluation and automatic generation using a description language. In *2014 IEEE Conference on Computational Intelligence and Games*, 1–8. IEEE.

Mark, D. 2009. AIIDE 2009 - The Photoshop of AI Debate - Michael Mateas. http://intrinsicalgorithm.com/IAonAI/2009/10/aiide-2009-the-photoshop-of-ai-debate-michael-mateas/.

Mojang. 2011. Minecraft.

Monolith Productions. 2005. F.E.A.R.

Neller, T. W.; Sooriamurthi, R.; Guerzhoy, M.; Zhang, L.; Talaga, P.; Archibald, C.; Summerville, A.; Osborn, J.; Resnick, C.; Oliver, A.; et al. 2019. Model ai assignments 2019. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 9751–9753.

Nelson, M. J. 2019. Institutions active in technical games research. http://www.kmjn.org/game-rankings/.

Nilsson, N. J. 1998. *Artificial intelligence: a new synthesis*. Morgan Kaufmann.

Ontanón, S. 2013. The combinatorial multi-armed bandit problem and its application to real-time strategy games. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*.

O'Rourke, E.; Andersen, E.; Gulwani, S.; and Popović, Z. 2015. A framework for automatically generating interactive instructional scaffolding. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, 1545–1554. ACM.

Patel, A. 2014. Introduction to the A* algorithm. https://www.redblobgames.com/pathfinding/a-star/introduction.html.

Rabin, S. 2002. *AI Game Programming Wisdom*. Rockland, MA, USA: Charles River Media, Inc.

Rabin, S. 2015a. JPS+: Over 100x faster than A*. https://www.gdcvault.com/play/1022094/JPS-Over-100x-Faster-than.

Rabin, S. 2015b. *Game AI pro 2: collected wisdom of game AI professionals*. AK Peters/CRC Press.

Russell, S. J., and Norvig, P. 2016. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Short, E. 2014. Blood & laurels.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676):354.

Smith, G.; Whitehead, J.; and Mateas, M. 2010. Tanagra: A mixed-initiative level design tool. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, 209–216. ACM.

Sorenson, N.; Pasquier, P.; and DiPaola, S. 2011. A generic approach to challenge modeling for the procedural creation of video game levels. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):229–244.

Stephens, T. 2014. UC Santa Cruz creates new Department of Computational Media. https://news.ucsc.edu/2014/10/computational-media.html.

Summerville, A. J.; Snodgrass, S.; Mateas, M.; and Ontanón, S. 2016. The vglc: The video game level corpus. *arXiv preprint arXiv:1606.07487*.

Sunshine-Hill, B. 2015. GDC 2015: "Turing Tantrums: AI Devs Rant!!". https://archive.org/details/GDC2015SunshineHill.

Tozour, P., and S. Austin, I. 2003. Building a near-optimal navigation mesh. In Rabin, S., ed., *AI Game programming wisdom*. Charles River Media. 171–185.

Valve South. 2008. Left 4 dead.

Volz, V.; Schrum, J.; Liu, J.; Lucas, S. M.; Smith, A.; and Risi, S. 2018. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 221–228. ACM.

Wexler, J. 2002. Artificial intelligence in games: A look at the smarts behind lionhead studio's" black and white" and where it can go and will go in the future. *University of Rochester* 19.

Yannakakis, G. N., and Togelius, J. 2018a. *Artificial intelligence and games*, volume 2. Springer.

Yannakakis, G. N., and Togelius, J. 2018b. Game AI panorama. In *Artificial Intelligence and Games*. Springer. 259–277.