Teaching Introductory Artificial Intelligence with Pac-Man

John DeNero and **Dan Klein**
Computer Science Division
University of California, Berkeley
{denero, klein}@cs.berkeley.edu

Abstract

The projects that we have developed for UC Berkeley's introductory artificial intelligence (AI) course teach foundational concepts using the classic video game *Pac-Man*. There are four project topics: state-space search, multi-agent search, probabilistic inference, and reinforcement learning. Each project requires students to implement general-purpose AI algorithms and then to inject domain knowledge about the Pac-Man environment using search heuristics, evaluation functions, and feature functions. We have found that the Pac-Man theme adds consistency to the course, as well as tapping in to students' excitement about video games.

Introduction

Because of the great breadth of the field of artificial intelligence (AI) today, introductory AI courses typically include a heterogenous collection of topics and techniques. For this reason, AI courses risk appearing incoherent to students, who can find it challenging to understand how different AI topics relate to each other. We have found that assigning a tightly integrated series of programming projects has lent unity to our AI course by providing students with a common platform for solving many different kinds of AI problems. In this paper, we describe a series of projects that use Pac-Man as a problem-solving environment for teaching state-space search, adversarial search, Markov decision processes, reinforcement learning, and probabilistic tracking.

We chose Pac-Man for several reasons. First, it's fun to play and watch, as it taps into students enthusiasm for video games and retro pop culture. Second, we sought a domain that would support deterministic, stochastic, partially informed, and adversarial problem settings. Finally, we wanted an environment that was both intuitive and rich. Pac-Man is intuitive in the sense that it consists of objects moving around on a grid, a setting that students can easily map onto the general definitions of search problems and Markov decision processes. Pac-Man is rich in the sense that very challenging AI problems arise; just eating all of the food dots in as few steps as possible is a non-planar traveling salesman problem.[1]

[1]The TSP is non-planar even though the food dots (cities) are arranged on a planar grid because of the maze walls.

Our projects are made consistent not only by their Pac-Man theme, but also by the common design goals that have guided each project's development. These goals include providing students with built-in visualization of the behavior of their algorithms, opportunities for creative extensions, and an effective balance of cooperative and competitive incentives. The remainder of this paper consists of first describing the goals that cross-cut the entire series of projects, then describing each component project. We conclude by describing the beneficial impact of these projects on our course. The Pac-Man projects are now available to the computer science community for educational use and have already been adopted by several other universities. We are happy to assist others in using, refining, and extending these projects so that other courses can realize the same benefits that ours has.

Design Goals

Our undergraduate AI course has two primary learning objectives. We expect our students to be able to (1) precisely define different classes of AI problems and the core algorithmic techniques that apply to them, and (2) apply those techniques effectively to specific problem instances, adding domain knowledge in appropriate ways to increase the effectiveness of general techniques.

Our projects support these learning objectives via five design goals that have guided their development. We believe that many of these project design principles should apply to computer science programming projects in general, but together they make a particularly effective recipe for AI projects that mix theory and applications. For each of the five goals below, we motivate the goal and explain how it manifests in the projects.

Engagement: Programming projects motivate students by addressing problems that students find intriguing, challenging, and fun. Targeting video games is an easy way to get our students' attention. However, video games designed for pedagogical purposes can appear dull or underengineered relative to commercial games. We invested time into ensuring that our implementation of Pac-Man was a faithful enough approximation to the original that it was fun to play.

Scaffolding: We provide substantial portions of each project's implementation to our students: the graphics,

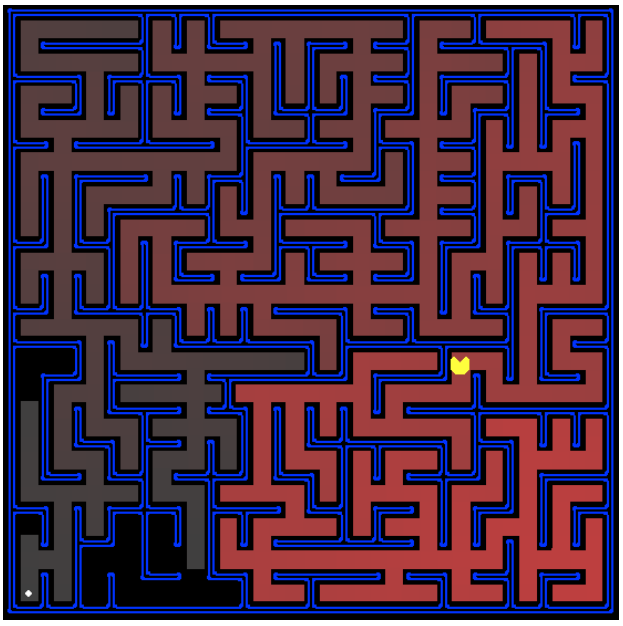

Figure 1: The search project visualizes an execution trace of each search algorithm when applied to the simple case of finding a path through a maze to a fixed location. Above, the states explored in breadth-first graph search are colored, with their shade indicating the order in which they were dequeued from the fringe.

game mechanics, object-oriented modularization, utility classes, and agent interfaces are all provided. Moreover, we ensure that each project already "runs" out of the box by providing implementations for rudimentary baseline agents (e.g., a Pac-Man controller that always moves left until it hits a wall). We have found that this amount of code scaffolding allows students to focus primarily on the AI topics, rather than struggling to implement software from scratch based only on a specification.

Visualization: We have designed each project so that as soon as students implement an algorithm, they can visualize its behavior. For example, as soon as they implement uniform-cost search, they can watch Pac-Man walk efficiently through a maze. Providing visual feedback is rewarding, and it additionally provides students with useful information as they debug their implementations. Where possible, we also visualize aspects of the execution trace of their algorithms. For example, we show the states explored by their search algorithms in the order that they are dequeued from the fringe, as in Figure 1.

Consistency: All of the projects use a common code base that provides a consistent interface for implementing Pac-Man agents and querying the current game state. Sharing code across projects dramatically reduces the start-up time and confusion in later projects.

Grading: The fixed scaffolding allows us to grade large portions of the projects programmatically using scripts. This grading approach has several advantages: it saves time for the course staff, but also allows us to provide immediate feedback to students when they submit their projects. We automatically e-mail them requesting a resubmission if their projects generate syntax errors, throw exceptions, or fail simple test cases.

These design goals have guided the *format* of our projects. We also focused on the following two design goals to guide our decisions in selecting project *content*.

Separating Algorithms from Problem Instances

The usefulness of most artificial intelligence techniques comes from the fact that the same algorithms, and in fact the same implementations, can operate on a wide variety of problem instances. One risk of focusing on a common problem setting throughout our course is that students may fail to appreciate that the techniques also apply much more broadly than simply to playing Pac-Man.

To emphasize the generality of AI techniques, we instruct students to provide problem-agnostic algorithm implementations. For instance, they write search and reinforcement learning algorithms against general interfaces for search problems and Markov decision processes, respectively. To enforce generality, we including multiple different problem instantiations for some projects. For example, the students' search code won't receive full credit unless it solves the 8-puzzle in addition to Pac-Man related search problems. Similarly, their reinforcement learning code must apply to the grid world from our course textbook (Russell and Norvig 2003) and a simulated robot controller, as well as Pac-Man.

We also believe that students should gain experience in adding domain-specific knowledge to their AI algorithms (learning objective 2) in the form of search heuristics, search problem definitions, evaluation functions for adversarial games, and features for linear approximations of expected reward. One benefit of reusing the Pac-Man environment across projects is that students quickly become domain experts on our implementation of Pac-Man. This consistency reduces the risk that students will fail at the domain-specific aspects of our projects simply because they didn't understand the domain well. Another benefit is that students can directly compare the formats of domain knowledge consumed by different AI techniques because they all address a common domain.

Promoting Collaboration and Competition

Students vary in how they react to collaborative and competitive scenarios. Some students are motivated by competition, while others prefer to be evaluated in absolute terms. Most learn effectively in collaborative context, while some prefer to work alone. In designing the Pac-Man projects, we have aimed to foster but not force collaboration in the core work, while at the same time providing optional competitions. In this way, we have been able to address the learning needs of a wide variety of students.

We designed our projects to be completed in pairs, but we allow students to work alone with no reduction in project requirements. This structure naturally encourages group work

without mandating it. We also found that grading on an absolute scale strongly promotes informal collaboration. For example, we use a newsgroup for course communication; the willingness of students to answer each other's questions increased dramatically when we switched from a relative to an absolute grading system. We explicitly promote an awareness that the competitive aspects of the projects are restricted to optional contests.

That said, several projects include an optional competition that extends the required material. For instance, the search project concludes with a contest to find the shortest path in an intractable traveling salesman problem, after they have solved easy instances exactly using A*. Similarly, the reinforcement learning project concludes with a feature design challenge. These contests are much more open-ended in character than the main projects, but are evaluated by clear quantitative measures that ensure fairness and transparency in our evaluation.

Although we offer only a small amount of extra credit for winning a contest, participation can range as high as 50% of the class. In particular, we have found that contests which directly extend required projects attract more participation than stand-alone contests: extensions require no additional start-up cost for the students and often allow students to implement ideas that occurred to them during the required portions of a project. The open-ended character allows students to think creatively about solving hard problems, and the competitive framework allows our strongest students to demonstrate their talents. We also view optional contests as opportunities for students to explore AI ideas beyond those required for the course.

We recognize the winners of our contests in lecture and describe their solutions to the class. Even the students who do not choose to participate in the contests enjoy learning about the most effective solutions to these challenging problems. Because the contests are optional and have little impact on the grading of the course, student reactions to the contests have been extremely positive.

Project Descriptions

Each project consists of a detailed description and a large amount of scaffolding code with clear directions for students on where to add their implementations of algorithms, heuristics, etc. We describe the projects in the order that we assign them in our course.

State-Space Search

Students implement depth-first, breadth-first, uniform cost, and A* search algorithms. These algorithms are used to solve navigation and traveling salesman problems in the Pac-Man world. In particular, we provide a search problem definition for navigating to a particular position in a maze. Students create a new search problem definition for finding a path that passes through all of the food dots in the maze: a much more challenging problem. They design and implement search heuristics for both a simple special case of the all-dots search problem and the full problem itself.

Although many students have seen uninformed search methods in previous classes, they often struggle to implement the graph search algorithms that this project requires. To aid their development, we provide a visualization of the exploration pattern of the search space for the simple maze navigation search problem, as shown in Figure 1.

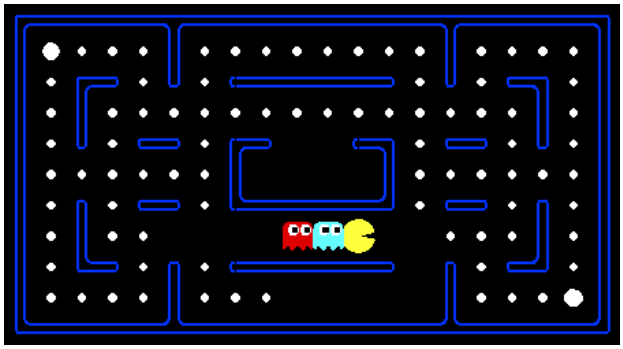

Figure 2: The multi-agent search project emulates classic Pac-Man, but using smaller layouts and clearly parameterized non-deterministic ghost behaviors.

For large mazes, we know of no heuristic that is sufficiently tight to enable A* to efficiently find the shortest path that consumes all food dots. The optional contest extension for the search project asks students to find the shortest path they can in a large maze, either through inadmissible heuristics, greedy techniques, or whatever extensions they devise on their own.

Multi-Agent Search

The second project models Pac-Man as both an adversarial and a stochastic search problem. Students implement minimax and expectimax algorithms, as well as designing evaluation functions for limited-depth search. This project begins by asking students to design a reflex agent, so that they can compare the behaviors that emerge from adversarial and stochastic search with the behaviors that they specified by hand using their knowledge of the domain. A screenshot for this project appears in Figure 2

We challenge students in this project to generalize the two-player adversarial search algorithms that we present in the textbook and in lecture to the case where there is one maximizer (Pac-Man) and many minimizers (the ghosts), which act in a fixed sequential order. Rather than attempting to visualize the game trees explored by their algorithms in this project, we tell the students what behavior they should observe in some very simple test scenarios to help them debug their implementations.

We grade their solutions by ensuring that all actions they choose are optimal under the modeling assumptions relevant to each question (stochastic or adversarial, with a fixed limited depth). In the case of this project, automatically grading their submissions with scripts is much more effective than manually inspecting their code.

The optional contest for this project asks students to design the best evaluation function they can for playing Pac-Man. We suggest that they define their evaluation functions

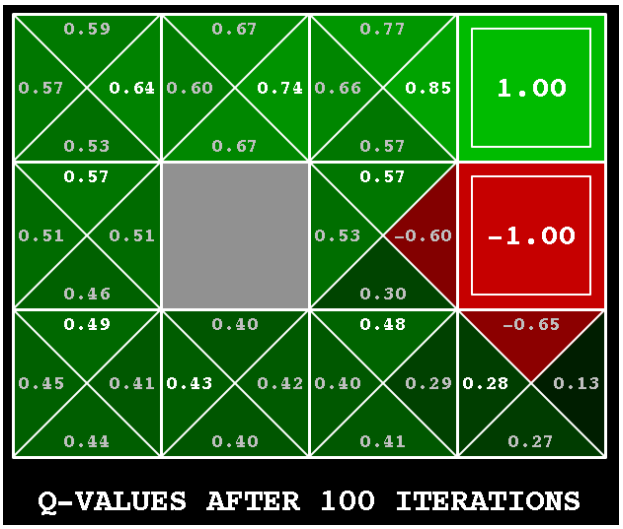

Figure 3: Our graphical output for the gridworld environment includes overlays for values, q-values, and agent simulations. We also support manual control via the keyboard, so that students can set up particular scenarios and see the effects of q-learning updates.

as linear functions of features so that they can experiment with learning feature weights in the following project.

Reinforcement Learning

Students implement model-based and model-free reinforcement learning algorithms, applied to the gridworld environment that is used as a running example in Artificial Intelligence, A Modern Approach (Russell and Norvig 2003). They then apply their code to both Pac-Man and a simulation of a crawling robot. The project requires students to implement value iteration, q-learning, and q-learning with linear function approximation. Students develop their algorithms against a fully functional Markov decision process implementation of gridworld that includes both graphical and text output. The graphical output visualizes the state of their learning algorithms, as in Figure 3.

The optional contest for this project asks students to design feature functions for an approximate q-learning Pac-Man agent that can be effectively trained in only a few training iterations. Students find that they cannot always reuse all of the features they designed for the multi-agent project because of approximation errors during learning.

Probabilistic Tracking

Teaching probabilistic inference in the context of Pac-Man required us to invent a new game variant with partial information. In this project, Pac-Man is hunting ghosts that are unobserved, but which give noisy observations about their relative location to the Pac-Man agent. Probabilistic inference for hidden Markov models and dynamic Bayes nets track the movement of these invisible ghosts. Students implement exact inference using the forward algorithm and approximate inference via particle filters.

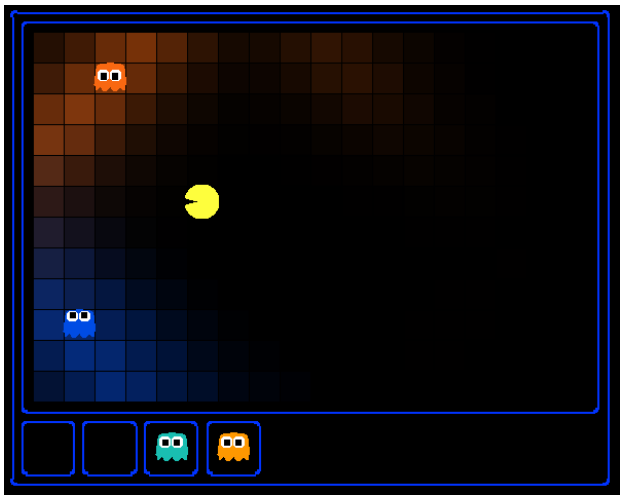

Figure 4: In the probabilistic tracking project, students compute belief distributions over the location of unobserved stochastic ghosts, which we visualize as heat maps overlaying the game board.

We visualize the output of student models by computing the marginal distributions over each ghosts' position and overlaying those distributions onto the Pac-Man game world, as shown in Figure 4. The visualization is particularly useful when students compare the behavior of exact and approximate inference. There is currently no contest extension for this project because at this point in our course we instead encourage students to focus their energy on the integrative multi-agent contest below.

Integrative Multi-Agent Contest

Our students' enthusiasm for the contest extensions of the early projects prompted us to develop a larger-scale contest that integrated several techniques from the course. The multi-agent contest focuses on a head-to-head capture-the-flag-style variant of Pac-Man in which each student entry must coordinate multiple agents that shift between ghosts and Pac-Man to simultaneously defend their own dots and try to eat the opponent dots. Figure 5 shows an example state from the game. To encourage the use of probabilistic inference, we hide the positions of opponent agents and provide only noisy relative position observations. Effective entries for this contest combine search, adversarial search, and probabilistic tracking, and exploit various domain-specific strategies.

We announce this project early in the course as an example of the kind of problem that requires integrating multiple AI techniques. Although the contest is optional and rarely has a material effect on student grades, over a third of the students usually participate, and the winning entries are always impressive. We allow teams of up to five students to work together. We have successfully held live play-offs that result in several hours of shouting, cheering, and frantic code editing as student teams pit their entries against one another to determine a contest winner. We also provide infrastruc-

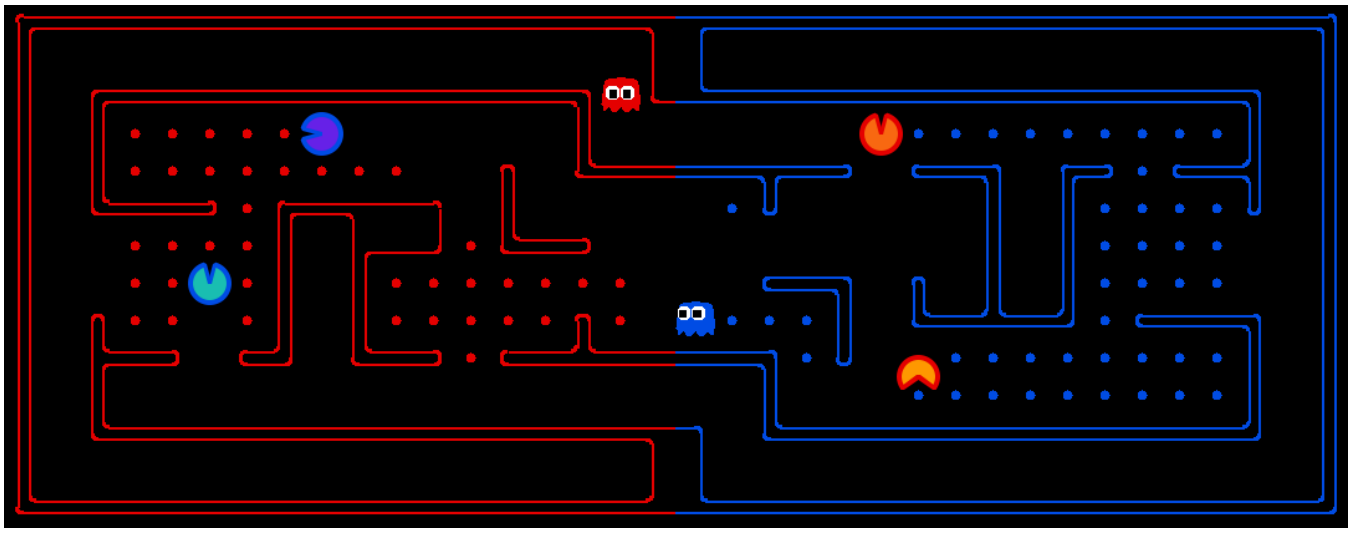

Figure 5: The multi-agent course contest integrates several techniques from the course, including state-space search, adversarial search, and probabilistic tracking. Each team (red vs. blue) controls multiple agents that shift between ghosts (defense) and Pac-Man (offense) as they simultaneously defend their own dots and try to eat the opponent dots.

ture to hold nightly tournaments during the development period so that student teams can gauge the quality of their entries.

Impact of the Projects

The Pac-Man projects represent an important component of a broader redesign of our undergraduate AI course, in which we have removed several topics (e.g., logical reasoning and planning) in order to focus more deeply on the remaining topics and how to apply them. The projects have been effective in motivating students and in creating opportunities for them to apply the general techniques they learn in lecture.

Responses to these projects have been overwhelmingly positive. In response to the survey question, "What worked best for you in this course?" 65% of students listed the course projects. Enrollment in the course increased a dramatic 69% the semester after Pac-Man was first introduced, and has increased by well over 100% in the three years since we began introducing these projects. Along with enrollment, student ratings about the usefulness and effectiveness of the course have also continued to increase as we have refined the projects.

These projects are effective in integrating the lecture and assignment portions of the course. In lecture, Pac-Man makes an excellent platform for demonstrations and examples. After describing a simple example on a small Pac-Man world, we can demonstrate algorithms working on larger problem instances and let students know that they too will build effective solutions for these instances. Describing winning entries for the contests also integrates well with lecture and gives students additional motivation to come to class. It is not unusual that students will applaud after a successful Pac-Man game.

The policies and mechanics we have developed for the projects have also been successful. In particular, we are very happy with grading on an absolute scale so that students will not be discouraged from assisting each other. We now have a very active online newsgroup in which students answer a majority of the questions that are posed. We have also found that automatically grading projects has several advantages, not only in time saved, but in providing students with early feedback about whether or not their code successfully runs on simple test cases. Finally, we have learned that contests provide an excellent outlet for strong students to explore the course material and demonstrate their talents. Of course, contests risk fostering an unproductive learning environment for less competitive students, but we believe that the combination of optionality of the contests and strong encouragement of cooperation for the required projects effectively mitigates any such concerns.

We are very interested in supporting other universities in adopting and extending these projects. Our experience so far in sharing these projects with the faculty of universities has been uniformly positive: at least ten courses around the world have used the projects in some capacity. As adoption increases, we expect that the Pac-Man projects will only continue to improve and expand in their coverage of AI topics. AI is an incredibly exciting field, and we hope that adding a little Pac-Man flavor to courses will help undergraduate students get involved.

References

Russell, S., and Norvig, P. 2003. *Artificial Intelligence, A Modern Approach*. Prentice Hall.