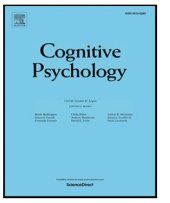
Cognitive Psychology 129 (2021) 101410

Contents lists available at ScienceDirect

Cognitive Psychology

journal homepage: www.elsevier.com/locate/cogpsych



Discovering skill

John R. Anderson *, Shawn Betts, Daniel Bothell, Christian Lebiere

Department of Psychology, Carnegie Mellon, United States

ARTICLE INFO

Keywords:
Skill acquisition
Cognitive architecture
Causal inference
Discovery learning
Video games

ABSTRACT

This paper shows how identical skills can emerge either from instruction or discovery when both result in an understanding of the causal structure of the task domain. The paper focuses on the discovery process, extending the skill acquisition model of Anderson et al. (2019) to address learning by discovery. The discovery process involves exploring the environment and developing associations between discontinuities in the task and events that precede them. The growth of associative strength in ACT-R serves to identify potential causal connections. The model can derive operators from these discovered causal relations just as does with the instructed causal information. Subjects were given a task of learning to play a video game either with a description of the game's causal structure (Instruction) or not (Discovery). The Instruction subjects learned faster, but successful Discovery subjects caught up. After 20 3-minute games the behavior of the successful subjects in the two groups was largely indistinguishable. The play of these Discovery subjects jumped in the same discrete way as did the behavior of simulated subjects in the model. These results show how implicit processes (associative learning, control tuning) and explicit processes (causal inference, planning) can combine to produce human learning in complex environments.

1. Introduction

Exploratory or discovery-based learning is a common way of learning to use computer interfaces and apps (e.g. Garett, Chiu, Zhang, & Young, 2016; Grossman, Fitzmaurice, & Attar, 2009). Much effort has been put into designing these interfaces so that they can be easily learned by exploration (Shneiderman, 2002). Video games are one form of technology that is often learned by discovery (along with other methods that bypass direct instruction such as watching others —see Hamlen (2011)) and games are often designed to facilitate this kind of non-instructional learning. This paper contrasts instruction-based learning with discovery learning in the case of a video game.

Direct instruction, whether in formal educational settings or informal interaction with others, is typically reserved for things that are not easy to discover on their own. A controversy in the educational literature is whether even learning of topics taught in school settings might be improved by letting the learner discover the knowledge rather than provide direct instruction (Mayer, 2004). This question receives particular attention in studies of mathematical learning (Cobb, Yackel, & Wood, 1992). Another distinction that is prominent in mathematics is whether students should develop their skills by trying to solve problems on their own or by studying examples of problem solutions (for a review see Lee & Anderson, 2013). Examples of observing how a problem is solved can be seen as something midway between receiving direct instruction and being left to discover how to solve the problem on one's own. It is argued that studying examples can reduce memory load and wasted search, but still provide some of the supposed benefit of the discovery learning approach.

* Correspondence to: 5000 Forbes Ave., Pittsburgh, PA 15213, United States.
E-mail address: ja@andrew.cmu.edu (J.R. Anderson).

https://doi.org/10.1016/j.cogpsych.2021.101410
Received 3 February 2021; Received in revised form 18 June 2021; Accepted 27 June 2021
Available online 9 July 2021

Table 1
Two-by-two representation of pairings of presence of cause and presence of effect (a,b,c,d are frequencies).

Cause	Effect	
	Yes	No
Yes	a	b
No	c	d

One thing that has been missing in discussions of learning by discovery in education and other domains is a detailed process model of how learning by discovery happens. The deep learning efforts in AI can be seen as providing process models of how learning by discovery happens in many domains including video games, where human or super-human levels of performance are being achieved (Mnih et al., 2015). However they are not plausible process models of human learning, at least in the case of video games. Studies of human learning of video games without instruction have found subjects to be able to learn with many fewer games of experience than deep learning (10s of games rather than 1000s) even if they might not achieve the same final levels (Tessler, Goodman, & Frank, 2017; Tsividis, Pouncy, Xu, Tenenbaum, & Gershman, 2017). Perhaps to confirm the obvious, these studies also show that subjects achieve high proficiency even faster if given instruction on the game. A factor influencing success of human learning without instruction is the prior assumptions subjects have about how objects in the game should behave. For instance, Dubey, Agrawal, Pathak, Griffiths, and Efros (2018) showed that human learning is hindered when the behavior of components of the game are changed either to hide their interpretation (e.g., replacing characters by colored blocks) or to be misleading (e.g., rendering a ladder that needs to be ascended by fire).

Our hypothesis is that human ability to play video games stems from an understanding of the causal structure of the games, whether from instruction or discovery (for a similar proposal see Ho, Yang, Quieta, Krishnamurthy, & Liausvia, 2019). Instructions for such games frequently describe how the game works and leave it largely to the learner to determine what needs to be done to achieve success in the game (e.g., see instructions in Appendix A). Players learning to play the game without instruction have to discover the game's causal structure. In some games (e.g. chess) it can be a major planning challenge to go from knowledge of rules of the game to a successful strategy for play, but in many of the video games that have been studied that translation is straightforward. In these video games once the causal structure is understood, it is clear how to play the game, although it can remain a learning challenge to master the motor actions to convert that policy into fluid game play. Having a causal model is also critical in learning to use many devices like a microwave oven and software products, even if the user's model is shallow (Polson & Lewis, 1990). In contrast to video games, using these devices often is not a motor challenge, and so when one understands the device one immediately achieves desired levels of performance.

The last few decades have seen a great deal of research and theory into causal learning (for a review, Holyoak & Cheng, 2011). All theories of causal learning assume that two preconditions to making an inference of a relationship between the cause and effect are that the cause occur before the effect and that the cause increase the probability of the effect. Typically, there is also an emphasis that the cause and effect be close temporally and spatially (Shanks, Pearson, & Dickinson, 1989), although as demonstrated by the counter-examples in Anderson (1991) there can be exceptions to such temporal and spatial constraints. Much of the research testing various theories of causal inference has involved presenting subjects a series of cases where a single possible cause is present or not and a single effect either occurs or not (Perales & Shanks, 2007). Subjects are typically asked to make a judgment of the causal strength between cause and effect on a scale from −100 to 100 where negative values mean the cause prevents the effect. Their experiences can be described in a 2 × 2 table like Table 1 where a, b, c, and d denote the frequencies of the 4 possible pairings of presence or absence of cause and effect. Two striking effects that are obtained (Anderson & Sheu, 1995) are that subjects are largely insensitive to the magnitudes of a–d but only the relative proportions and that the quantities in the presence of the cause (a and b) have a greater effect than in the absence (c and d). Neither would be considered normative from the perspective of a chi-square test of contingency. Whether these effects are considered normative or not turns on how one interprets the causal judgments that are being made.

Our approach to learning the causal structure of video games will similarly depend on an accumulation of information like Table 1, but with two important differences. First, there are many potential causes and many effects, creating an information-processing challenge in determining what causes what — effectively, the 2 × 2 Table 1 becomes a matrix of m potential causes and n effects. Second, while the subjects′ belief about a potential cause–effect relationship may vary on a continuous scale, there are also discrete points where subjects treat the connection as actionable and transform their play to reflect a belief in a cause–effect relationship. We will provide evidence for such discrete changes in subject game play.

1.1. Autoturn Space Fortress

This research uses a video game that originated from the Pygame implementation of Space Fortress (Destefano, 2010) that used a keyboard rather than the joystick that was used in the original version of Space Fortress (Donchin, 1989). We developed a simplified version of this Space Fortress implementation that was described in Anderson, Betts, Bothell, Hope, and Lebiere (2019) along with a detailed cognitive model. That model captured how subjects given instruction became fluent in playing the game over the course

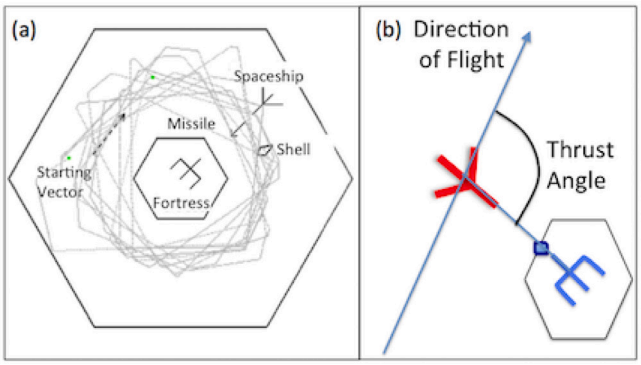


Fig. 1. (a) The Space Fortress screen, showing the inner and outer hexagon, a missile fired at the fortress, and a shell fired at the ship. The distance from the center (fortress) to the corners of the outer hexagon is 200 pixels and the distance to the corners of the inner hexagon is 40 pixels. The ship starts 120 pixels to the left of the center, flying at 30 pixels per second, parallel to the upper left side of the hexagon. The dotted lines illustrate an example path during one game. (b) A schematic representation of critical values for firing and flight control.

of 20 3-minute games. The model successfully reproduced the learning of subjects over a number of variants of the Space Fortress game, in each case given instruction on the game (see also Gianferrara, Betts, & Anderson, 2020; Seow, Betts, & Anderson, 2019).

In this paper we chose the variant of Space Fortress called Autoturn. Fig. 1a illustrates the critical elements of the game. Players are instructed to fly a ship between the two hexagons. This version is called Autoturn because, as the ship flies, it automatically turns so that it is always aimed at the fortress. Subjects are firing missiles at a fortress in the middle, while trying to avoid having their ship hit by shells from the fortress. The ship flies in a frictionless space. To navigate, the player must combine thrusts in various directions to achieve a path around the fortress. Mastering navigation in the Space Fortress environment is challenging even when explained: while subjects are all video game players, most have no experience in navigating in a frictionless environment. In the Autoturn variant of the game, flying the ship is simpler than in regular Space Fortress because the ship is always aimed at the fortress and subjects do not have to turn it. The ship begins each game at the position of the starting vector in Fig. 1a, flying at a moderate speed in the direction of the vector. To avoid having their ship destroyed, subjects must avoid hitting the inner or outer hexagons, and they must fly fast enough to prevent the fortress from aiming at the ship and shooting it with a shell. When subjects are successful the ship goes around the fortress in a clockwise direction. They can destroy the fortress by shooting missiles at it to build up its vulnerability and then issuing a Kill Shot (two shots in rapid succession). If the fortress is destroyed it leaves the screen for 1 s before respawning. If the ship is destroyed it respawns after 1 s in the starting position flying along the starting vector. This version of the game eliminated much of the complexity of scoring in the original game and just kept three rules:

1. Subjects gained 100 points every time they destroyed the fortress.
2. Subjects lost 100 points every time the ship was destroyed
3. To reinforce accurate firing, every fire costs 2 points.

To keep subjects from being discouraged early on, their score never went negative. The replay site (http://andersonlab.net/reconstruction/) offers examples of game play from the study of Anderson, Betts, Fincham, Hope, and Walsh (2020).

Good performance involved mastering two skills — destroying the fortress and flying the ship in the frictionless environment. To destroy the fortress, one must shoot at the fortress (using the spacebar key) and build up its vulnerability (displayed at the bottom of the screen). When the vulnerability reaches 10 or more, subjects can destroy the fortress by quickly firing a pair of shots (which together are the Kill Shot). Each fire increases the fortress's vulnerability by one, provided the fires are paced at least 250 ms apart. If the inter-fire interval is less than 250 ms apart, the vulnerability is reset to 0 and one must begin the buildup of vulnerability anew. While subjects could easily make sure the shots building up vulnerability are at least 250 ms apart by putting long pauses between them, this would reduce the number of fortresses destroyed and points gained per game. Thus, subjects are motivated to pace the fires as close to 250 ms as they can without going below that 250 ms threshold. In contrast to the fires that build up the vulnerability, the pair of shots that constitute the Kill Shot must be less than 250 ms apart.

Since the ship is always aimed at the fortress, subjects do not need to turn their ship as in the original version of Space Fortress. Subjects' average ship speed is a little over 1 pixel per **game tick** (the game updates every 30th of a second). To navigate around the fortress, they must press the thrust key (w) at appropriate times and for appropriate durations. The direction of the ship after a thrust is determined by a vector sum of the current flight velocity and the acceleration that is added in the direction that the ship is pointing (which is always at the fortress). The acceleration is determined by how long they hold the thrust key down. Every game tick the thrust key is held down adds .3 units of speed in the current orientation of the ship. As an example, suppose the ship is flying at 1.2 pixels per game tick, the angle between aim and ship direction (Thrust Angle in Fig. 1b) is 120 degrees, and the thrust key is held down for 4 game ticks. This will produce a velocity change of 1.2 pixels in the direction the ship is aimed. The resulting trajectory would still have a velocity of 1.2 pixels per game tick (more if the thrust angle was less than 120 degrees, less if it was more), and would now be in a direction that bisected the thrust angle. Thrusts at the wrong time or for the wrong duration can lead

to death of the ship, which happens if the ship hits the inner or outer hexagons or if the ship flies such that its angular velocity is slow and the fortress can shoot it.

There have been a couple of efforts to apply deep reinforcement learning to Space Fortress (Agarwal, Hope, & Sycara, 2018; van Oijen, Roessingh, Poppinga, & García, 2019). The vulnerability feature of Autoturn makes it particularly challenging. Each shot needed to build up vulnerability loses 2 points and so might seem a bad thing to do. Then when vulnerability reaches 10, the rules of good game play suddenly switch and a double-shot becomes the right thing to do. The fact that shots to build up vulnerability cost points but eventually lead to a big gain is particularly difficult for deep reinforcement learning efforts

1.2. The discovery manipulation

To provide comparison with the deep reinforcement efforts Anderson et al. (2019, Appendix B) report a pilot study examining whether subjects could learn Autoturn without instruction. 20 subjects were just told that there were two relevant keys (the w-key and the spacebar key) and that there were points to be gained (see instructions in A.13). We were surprised that about half the subjects seemed to be able to learn the game, if more slowly than we had observed in studies of instructed subjects. It was not too surprising that subjects learned to fly observing what the ship did in response to pressing the thrust (w) key and observing deaths of the ship. What surprised us (perhaps biased by our knowledge of the problems in ongoing deep-reinforcement learning efforts) was that subjects learned about the vulnerability threshold for a Kill Shot. Given that Kill Shot at any vulnerability over 10 would work, we were even more surprised that the successful subjects were learning that the vulnerability threshold was exactly 10. Therefore, we decided to run an experiment that sampled Discovery and Instruction subjects from the identical pool and introduce a manipulation that we thought would offer insight as to how subjects were learning about vulnerability.

A number of aspects of Autoturn suggest how it should be played and perhaps provide clues to the critical role of vulnerability. The shape of the ship suggests a flying object and that it should move when thrust is applied in the direction it is pointing (and Discovery subjects see it as such). The shells suggest shooting while explosions suggest both good and bad events (and Discovery subjects see these events as such). Such games often come with a concept like an opponent's health that must be diminished to destroy it, or some kind of special power that must be built up to defeat the enemy. Some sort of special action is often required to destroy the opponent when the threshold has been reached.

We created a version of the game that removed these suggestive features: The ship was replaced by a round ball that moved according to the same rules as in Autoturn.[1] There were no missiles, shells, explosions and no fortress. The only effects of shooting were on the counter and points at the bottom of the screen (see game illustration in A.12). Killing the ball simply meant that it would disappear for a second and then reappear at the start position taking the same trajectory over the screen as the ship would. We called this version of the game No-Semantics because we had removed the aspects of the game that suggested the flying and shooting aspects that provide meaning to Space Fortress. For purposes of contrast in this paper, we will refer to the original game as the Semantics version.

No-Semantics subjects saw the game in different terms even if the same key actions would result in the same trajectory, vulnerability, and point changes in either game. Below is how a successful No-Semantics Discovery subject described the game:

> "The ball will move on its own, but pressing W will change its direction (slightly to the right of its current direction). Don't let the ball touch the lines, or you'll lose 100 points. While moving the ball, tap the spacebar somewhat quickly (but not too quickly, or it'll reset). Once it reaches 10, press it twice very quickly. That will 100 points. Pressing spacebar consumes 2 points."

Contrast this with a description from a successful Semantics Discovery subject:

> "The goal of the game was to destroy the object in the middle of the screen. In order to destroy the object the player had to shoot at the object slowly 10 times, then shoot quickly after that. Points were subtracted after every shot, so one would want to minimize the number of shots taken. It was a good idea to fly closely to the middle object, otherwise the object who shoot at, and potentially destroy, the player."

We thought the Semantics manipulation was line with manipulation of game elements (e.g., changing a ladder to a wall of fire) that Dubey et al. (2018) had shown to have strong effects on learning by discovery.

2. Methods

2.1. Subjects

200 subjects were recruited on Mechanical Turk and 50 were randomly assigned to the 4 conditions created by crossing the factor of instruction versus no instructions and the factor of Semantics versus No-Semantics. Subjects were paid a base amount of

[1] In Autoturn the ship was always aimed at the fortress and the effect of a thrust did not depend on how player oriented the ship in contrast to the original Space Fortress. This made possible the use of a ball that was not pointed in a direction.

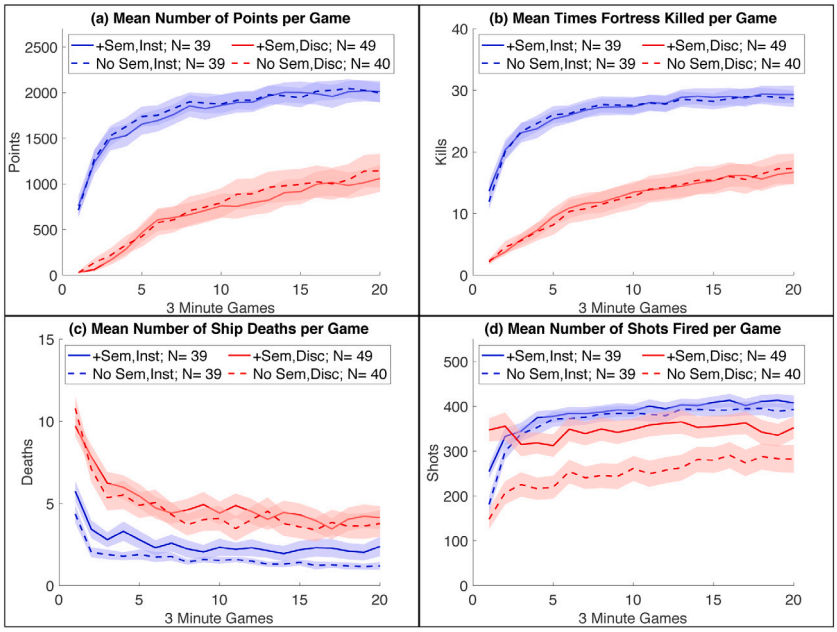


Fig. 2. Change in 4 performance measures over the 20 games for subjects in the 4 conditions. See text for discussion. The shaded areas represent +/−1 standard deviation of the mean.

$4 plus an amount that depended on their scores. We adjusted the conversion of points into pay such that subjects in all conditions averaged about $6 in performance bonus.

In their post-experiment questionnaires some subjects (they all had US IP addresses) were producing awkward and short answers (e.g., "Yes. i am were able to over come", "Very well understand this game"). Poor English comprehension is problematic when we want to study the effect of understanding the instructions. To have an objective measure rather than making subjective English quality judgments of questionnaire answers, we used the length of the answers (which seemed strongly related to English quality of answer). Using a threshold of 225 characters, 33 subjects were rejected leaving 39 subjects in the Semantics Instruction Condition, 39 in the No-Semantics Instruction condition, 49 in the Semantics Discovery condition, and 40 in the No-Semantics, Discovery Condition. The mean length of the answers was 117 characters for the excluded subjects and 863 characters for the included subjects. 17 of the 22 excluded subjects in the two conditions with instruction scored less than 300 points on the final game whereas this was only true for 5 of the 78 included subjects who received instruction. The 167 included subjects ranged from 18 to 56 years of age with a mean of 31 years. There were 110 males and 57 females.

2.2. Procedure

The actual experiment is available to run in the four conditions at out website (http://andersonlab.net/demos/autoturn2x2/). The exact physical details of the Space Fortress game are described in Anderson et al. (2019). The experiment began with the filling out of a consent form and collection of demographic information. Then subjects read the instructions appropriate to their condition (see Appendix A). After reading the instructions subjects played 20 3-minute games of Autoturn. Game play was followed by a questionnaire that asked them to describe what they understood about the game, how they played it, and whether the game reminded them of any other games they had played.

2.3. Model runs

There was a single model for both conditions, which will be described after the results. We ran 200 simulated subjects with instruction and 200 simulated subjects without instruction, varying the same parameters that control rate of learning as in Anderson et al. (2019) —see Appendix B for details. The model interacts with the same game software as the subjects producing identically structured records of game play and we can apply the same analyses to model data and subject data.

3. Results

Fig. 2 shows some measures that capture the effects of the 4 conditions created by crossing the two factors (Instruction versus Discovery; Semantic vs No Semantics). In terms of points earned (Part a), there is large benefit of instruction ($F(1,163)=95.05$, p

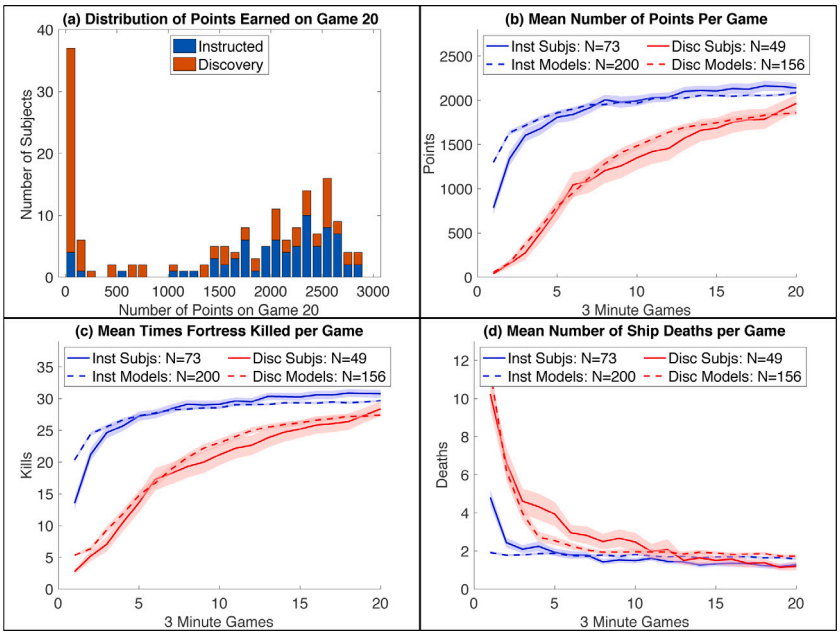


Fig. 3. (a) Distribution of number of points gained on the 20th game. (b)—(d) performance measures for subjects and models who exceed 300 points on game 20. The shaded areas represent +/ − 1 standard deviation of the mean.

< .0001), no effect of semantics (F(1,163)=0.12), nor an interaction (F(1,163)=0.01). Similar patterns are seen in the two major factors determining points, number of fortresses killed (Part b) and number of ship deaths (Part c): large effects of instruction (F(1,163)=99.07 for kills and 29.58 for deaths, both p< .0001), no effect of semantics (F(1,163)=.00 and F(1,163)=1.65, p .20), and no interaction (F(1,163)=0.00 and F(1,163)=0.60). It might appear that Instruction subjects with semantics make more errors (blue solid line in Fig. 2c — average over games of 2.57) than Instruction subjects without semantics (blue dotted line — average 1.66), but a t-test of this difference falls short of significance (t(76)=1.90, p > .05).[2] Let it be thought that semantics had no significant effects, part (d) of Fig. 2 shows number of shots fired per game, where there is not only an effect of instruction (F(1,163)=15.05, p < .0005) but also an effect of semantics (F(1,163)=9.06, p < .005) and an interaction between the two (F(1,163)=4.13, p < .05). The Semantics Discovery subjects can observe missiles fire from their ship and this visual effect induces many more shots (average over games of 346) than the No-Semantics Discovery subjects (average of 251) — a difference which is quite significant (t(87)=2.94, p < .005). However, this does not result in more kills because many more of the shots reset the vulnerability for Discovery Semantics subjects (an mean of 85.2 resets per game) than the No-Semantics Discovery subjects (an mean of 14.4 resets - t(76)=3.08, < .005). Further analyses will average over the Semantics vs No Semantics dimension because it failed to have an effect on subject success.

Part (a) of Fig. 3 shows the distribution of number of points subjects earned on game 20. There is a wide distribution of success for both subjects receiving instruction and those left to discover how the game worked. Probably many factors are behind these individual differences. Motivation is a possible factor in all conditions, probably more so when subjects face the frustration of having to discover how to play the game. Given that the game was played over the Internet some subjects may have had problems with their machines. To explore some of the sources of individual differences, we used demographic reports of subjects (given before they played) as to age, gender (110 males, 56 females), length of experience with video games (coded as more than 8 years or less, 73 subjects < 8 years, 93 more), and platform (coded as personal computer versus other such as console or mobile device, 82 vs. 84 subjects). We entered these into a linear regression model along with whether they had instruction or not (78 vs. 88 subjects). The only variable not to be significantly related to score was gender (t(160)=1.65, p> .1). Of course, there was a large positive effect of instruction (t(160)=6.39, p < .0001). Performance dropped with age at the rate of 35 points per year (t(160)=−3.47, p < .001), was 506 points greater for those who had played video games for more than 8 years (t(160)=3.12, p<.005), and 272 points greater for those who were playing on a personal computer (t(160)= 2.01, p< .05). While there are these overall effects, it needs to be stressed that collectively (instruction included) they only accounted for 30% of the variance and subjects could be found that were exceptions to all trends.

The distribution in Fig. 3a is bimodal, with some subjects displaying varying degrees of success and some subjects appearing quite lost even after 20 games. Many subjects scored 0 points in their last game and others had less than 100 points (a result that can happen with totally random play). As a measure of understanding how to play the game, we selected subjects who scored more than 300 points on that last game. This resulted in excluding 45 subjects whose mean score was 38 points. Only one of these

[2] All t-tests reported are 2-tailed.

subjects scored more than 150 points and that subject scored 248 points (the worst included subject scored 460 points). These excluded subjects came from all conditions but unevenly — 3 of 39 subjects in Semantics Instruction, 2 of 39 in No-Semantics Instruction, 22 of 49 in Semantics Discovery, 18 of 40 in No-Semantics Discovery. Parts (b) — (d) show performance statistics for the included subjects. While still remaining, the differences between Instruction and Discovery subjects have been substantially reduced, particularly by the 20th game. The benefit of instruction on the 20th game is marginal for points ($t(120)=1.70$, $p < .10$) and kills ($t(120)=1.99$, $p < .05$) and non-existent for deaths ($t(120)=.74$). Four of the top 8 subjects are from each group. The very best game-20 score (2876 points) was obtained by a Discovery subject. Discovery and Instruction subjects seem to be reaching the same asymptotic performance, but Discovery subjects are slower in achieving that.

As noted in the legends of the figures, all 200 of the Instruction models reached the 300-point threshold for inclusion but only 156 of the Discovery models. The next section describes the model for the two conditions and data that supports these models. Our focus will be on the successful subjects and models. Appendix C provides a description of the behavior of the 40 unsuccessful Discovery subjects and how this compares with the behavior of the 44 unsuccessful Discovery models.

4. Model of learning from instruction and discovery

Given the lack on any consequential difference between the semantics and no-semantics condition, we concluded that the basis for learning was not in prior assumptions that subjects might have about how video games work. Rather, their success must be based on observing the basic mechanics of the two games, which were identical. We hypothesized that subjects were making causal inferences not unlike those that subjects make in studies that look directly at causal inference, and that they were deciding how to play the game from this causal understanding.

We took the model described in Anderson et al. (2019), which successfully modeled learning from instruction, and augmented it to be rooted in a causal understanding of that game. To summarize that 2019 model (see Appendix B in this paper for more detail): it started with a declarative representation of the instructions about when to do what. This produced slow performance initially, but over time the model built direct-action rules that directly perform the actions in the appropriate situations bypassing the need for declarative retrievals. These direct-action rules respond directly to states of the game with requests for motor actions. Besides speed, critical factors in Autoturn are learning when to thrust and when to fire. A Controller module has been implemented within ACT-R that explores a range of values for when to fire and when to thrust and converges on appropriate settings, which the model comes to exploit. The creation of direct-action rules and the learning of control values for action underlie the improvement with practice in the model. The behavior of the model is similar to subjects in part because it uses established ACT-R settings (on the basis of prior experiments) for the timing and variability of mental steps and motor execution.

We augmented the 2019 model to work in both the Instruction and Discovery conditions. Below we describe how this model behaved when given instructions, and then in more detail how it behaved when it was not given instructions.

4.1. Learning from instruction

The model in Anderson et al. (2019) began with operators coded in declarative memory that specified how to play the game. However, inspection of the instruction in Appendix A reveals that subjects are not told how to play the game. Rather, the instructions describe the causal structure of the game and leave for the subject the planning step of determining what this means for playing the game. For instance, the instructions specify that the w-key will send the ship inward and that touching a border will result in death. The instructions leave it for subjects to decide that they should hit the w-key to avoid the border. They tell the subject that each shot will increase the vulnerability of the fortress (unless it is a double-shot) and that double-shot when vulnerability reaches 10 will destroy the fortress, but again leave it to the subject to infer what shot pattern to adopt. Therefore, we provided the model with a representation of the causal relationships described in the instructions and the model created the operators that specified when to take what actions. Specifically, we provided the model with declarative representations of the following 6 causal relationships:

1. Pressing the w-key causes the ship (ball) to move inward.
2. A single tap of the spacebar increases the vulnerability (counter) by 1.
3. If the ship (ball) hits the border it will go back to the start position.
4. A double tap of the spacebar resets the vulnerability (counter) to 0.
5. If the ship (ball) goes back to the start position there is a 100-point loss.
6. A double tap when vulnerability (counter) is 10 or greater results in a 100-point gain.

The actual instructions contain more causal information, but this is what the augmented model needed to construct the set of operators to play the game, i.e. those that had been given to the model in Anderson et al. (2019).

Fig. 4a represents the operators that the model constructs from this causal knowledge given the goals of avoiding point loss and achieving point gains. Each of the 7 boxes in the figure corresponds to an operator that the model builds in declarative memory.[3] The top operator 1 in Fig. 4a tests whether the ship has drifted too close to the outer hexagon and if it has, control will pass to

[3] This is a simpler operator structure than in Anderson et al. (2019) because that model also dealt with regular Space Fortress where the player had to turn the ship.

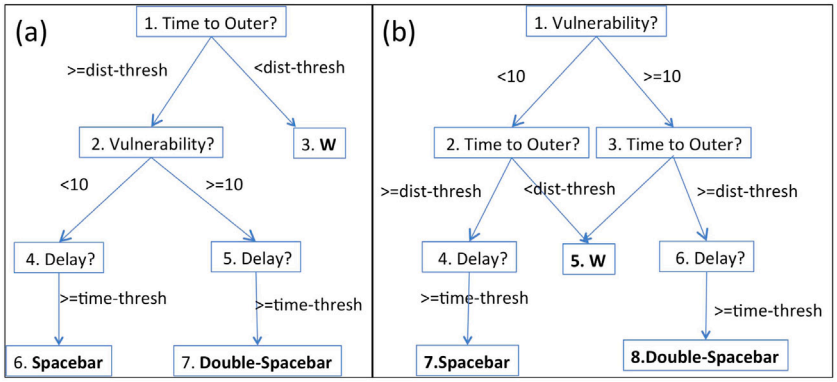


Fig. 4. The logical structure of the operators derived from a causal understanding of Autoturn. Each box corresponds to an operator. The model starts at the top operator and performs tests until it gets to the action operators at the bottom. Part (a) shows the operator structure created in the Instruction condition where models start with a complete causal structure. Part (b) shows the operator structure that successful models build up in the Discovery condition.

operator 3 that presses the w-key to thrust away (operator box to the right)[4]. If the ship has not drifted too close to the border, control passes operator 2 which tests if the fortress is ready to be killed. The Delay? operators (4 and 5) test if enough time has passed, allowing it to either take a single-shot (operator 6) or a double-shot (operator 7) depending on vulnerability. Control goes back to the top box after each of the action operators (3, 6, and 7).

While the operators specify what to do, this is not enough to achieve high performance. Acquisition of the needed skill was the focus in Anderson et al. (2019). That model learned from experience the right values for three parameters that control its behavior: when to thrust to avoid drifting into the outer hexagon, how long to thrust, and how far apart to keep its shooting to avoid resetting the vulnerability. Much of the focus in Anderson et al. (2019) was on the new ACT-R Controller module that learned these values through a process of control tuning. The other kind of learning involves production compilation, which has been a critical part of many ACT-R models of skill acquisition (e.g. Anderson et al., 2004; Taatgen & Anderson, 2002). It is the process by which special rules are acquired for performing a task. In the current situation ACT-R has to initially retrieve the operators and determine how to apply them, but it gradually learns rules that skip operators in that tree. Eventually, it learns direct-action rules that respond directly to game situations with motor actions.

The improvement in performance over games of the Instruction model in Figs. 3b–3d is a result of this combination of control tuning and production compilation. Asymptotically the model reaches the same level of performance as the successful subjects. However, for the first few games the model is somewhat better than the subjects, killing more fortresses and dying less frequently. We could have slowed down learning by parameter adjustments, but our goal is not to get an exact fit to this group of subjects but to show that the model captures the characteristics of their learning given a prior set of parameters (those determined in Anderson et al., 2019)[5]

4.2. Learning by discovery

Fig. 4b shows the operator structure that eventually emerges in successful Discovery models. It differs from Fig. 4a in that the top test is for vulnerability. This is because, as will be explained, Discovery models learn about vulnerability late when they have built the simpler test-distance-and-shoot subtree on the left (operators 2, 4, 5, and 7). Thus, when vulnerability is discovered this existing operator subtree has to be put under the vulnerability test[6] and another subtree is created for the Kill Shot (operators 3, 5, 6, and 8). Once the operators in Fig. 4b have been created, a model's learning will proceed just as in the Instruction model and the same direct-action rules will develop. The direct-action rules are the same whether they are learned from the operators in Fig. 4a or Fig. 4b. Thus, asymptotically the behavior in successful Discovery models will be the same as in successful Instruction models.

We believe the processes by which the Discovery model comes to the operators in Fig. 4b are involved in discovery learning in many situations. The next 6 subsections describe these mechanisms in detail and how they apply in the current context. Section 4.2.1 describes how the model begins exploring the game and noticing discontinuities in game flow. Section 4.2.2 describes how associations grow among the things it notices and how the strength of these associations points the model to causal inferences. Section 4.2.3 reviews the success of the model at identifying causal relationships and how that success relates to success of game

[4] This and other tests in Fig. 4 are achieved by comparing a value in the Game-State or Temporal buffer, representing current state, with a value in the Goal buffer that is the threshold value — see Appendix B.

[5] A different group of subjects would have required different parameter settings. There were subjects in the current pool that started out better than the model average and others that started out much worse.

[6] Given that declarative structures cannot be deleted in ACT-R, attempting to insert the vulnerability test under the distance test would cause interference among operators.

play. Section 4.2.4 describes how these causal inferences result in the construction of the operators in Fig. 4b. Section 4.2.5 describes how the construction of these operators results in discontinuities in the models' play and how this corresponds to discontinuities in subjects' play. The last subsection provides a detailed analysis of how subjects and models learn that 10 is the exact threshold for a kill shot.

4.2.1. Exploratory behavior and noticing discontinuities

In a continuously changing environment like Autoturn we hypothesize that discovery learning is driven by noticing discontinuities in the flow of the game. The model experiments with the two keys, w and spacebar, mentioned in the minimal discovery instructions (see Appendix A), pressing them at random times looking for discontinuities. When it detects a discontinuity, it looks for something that might have caused it.

The mean rate of the exploratory key presses was set to match approximately the keying behavior of Discovery subjects in the first 30 s of play, when presumably subjects are still exploring.[7] These subjects average 16.0 presses of W (thrust) and 36.5 presses of spacebar in these first 30 s. The model waited variable and random amounts of time to press the W key and a different variable and random times to the press the spacebar. For the thrust key the mean wait was 1.5 s (uniform between 1 and 2) resulting in an average of 15 presses in the first 30 s.[8] For the shot key the mean wait time varied between .1 and 5 s (according to a truncated exponential distribution) resulting in an average of 29.6 shots in the first 30 s.

The model could detect 6 discontinuities in game flow, which correspond to the 6 causal relationships given to the Instruction model. These effects are listed below and their median frequency in the first game for successful subjects and models:

1. Ship changes direction of motion: Subjects 75, Models 82.
2. Vulnerability increases: Subjects 140, Models 218.
3. Ship dies and returns to starting trajectory (a special case of 1 because ship direction changes): Subjects 9, Models 11.
4. Vulnerability resets to 0: Subjects 14, Models 20.
5. The fortress is killed and there is a big point gain (a special case of 4 because the vulnerability is also reset to 0): Subjects 2, Models 5.
6. The ship dies and there is a big point loss (a special case of 4). Early in the game this is relatively rare because the player has no accumulated points to lose (and points can never be negative): Subjects 1, Models 3.[9]

Note that the most important effects, 5 and 6, are the least common.

Given the sheer number of these effects, it is unreasonable to suppose subjects would detect and encode them all. To reflect this, each model was seeded with a **Noticing Probability**, which is the probability that a big point change would be detected and encoded. The Noticing Probability for each model was randomly selected from the interval 0 to 1. It seemed reasonable to suppose subjects would fail to notice a much larger proportion of the first 4 types of changes. The probability of noticing other ship deaths (when there was not a point loss) was set to 1/4 of the probability of noticing big-point events and the probability of noticing the more frequent effects (1, 2, and 4) to 1/8 of the big point probabilities. These are not carefully fit probabilities but reflected the principle that less frequent and more important effects would be noticed more often. The random choice of noticing probabilities resulted in a range of models that attended to and tried to explain different numbers of critical events. We hypothesize subjects also varied in how attentive they were to critical changes in game state. Appendix C presents evidence that the unsuccessful subjects tended to notice fewer of the critical relationships in the game.

Once an effect was noticed the model considered what happened in the past half-second preceding that event and selected the most recent event as a potential cause (Contiguity Principle). Potential causes were presses of the w-key and the spacebar key. If the gap between two w's or two spacebars was less than 250 ms, the pair was perceived it as a double-w or a double-spacebar. The cause might also be something that happened in the game other than any action on the model's part. While the number of things in the game that might be the cause of an event is very large if not infinite (for instance the ship passing a specific location on the screen might cause a vulnerability reset), the game is so sparse that little beyond the key presses are plausible as causes. The one other thing the model considered as a possible cause was touching the border. So, in total, the set of possible causes of an event were just single press of spacebar, double press of spacebar, single press of w, double press of w, and touching a border. Note that unlike the typical causal inference study, where subjects are focused on one potential cause and one potential effect, there are 30 (5 causes by 6 effects) pairings to sort out. Such a large space of causal connections is typical of many real situations where people learn by exploration.

[7] Only 6 of the 49 subjects have managed to kill a fortress (once each) in the first 30 s of game 1 and only 4 of those have any points left from their one kill at the end of the 30 s. All but 2 subjects have already died at least once (average of 2.96 deaths in the first 30 s). In contrast all 49 subjects kill at least 1 fortress in the first 30 s of game 20 (average 4.42) and only 1 subject dies in those 30 s. They average 338 points in the first 30 s of the last game (in contrast to the average of 6 points in the first 30 s of the first game).

[8] While 30 s divided by a mean waiting time of 1.5 equals 20 presses, the models are sometimes engaged in another activity when it comes time to make a press and so make their key press later than the waiting time.

[9] When one effect was a special case of another, the effect the model identified was the special case.

Table 2
Pairings noticed by one run of the model for 1 game.

Causes	Effects					
	Big loss	Big gain	Death	Vuln = 0	Direction	Vuln-Inc
Border	0	0	3	0	0	3
D-space	0	1	0	4	0	1
Space	0	0	0	1	0	12
D-w	0	0	0	0	2	0
w	0	0	0	1	9	1
None	0	0	0	0	0	1

4.2.2. Using associative strength for causal inference

In situations where there is a large space of possible cause–effect combinations, strength of associations among potential causes and effects can provide a guide as to which are true causal relationships. Each time the model noticed a discontinuity and a preceding event, it put the preceding event in the goal buffer and retrieved its memory of this type of a discontinuity. This retrieval has two consequences of relevance in ACT-R. First, the retrieval results in a strengthening of the association from the potential cause to the discontinuity. Second, the retrieval takes an amount of time, which the model can use as a measure of the strength of the existing association between the cause and the effect. Thus, consistent pairing of the preceding event and the discontinuity will result in rapid retrieval, which the model can treat as evidence of a causal relationship between cause and effectpast (see Lebiere & Wallach, 2000, for a similar idea). Appendix D develops in detail how the associative strength grows in ACT-R but, to summarize, it provides an estimate of the log likelihood ratio of the preceding event given the discontinuity i:

$$S_{ji} = log\left(\frac{P(j|i)}{P(j|\neg i)}\right) \tag{1}$$

One might think if j is the cause of i, $P(j|i)$ should be 1 and $P(j|\neg i)$ would be 0 since each effect has just one cause in the game and each cause is guaranteed to produce the effect. However, the noise in the game and the model can cause mis-encodings. For instance, vulnerability is reset to zero if and only if the spacebar is pressed twice quickly (less than 250 ms apart) but there are three ways something else can be perceived. First, what is perceived as a single tap of the spacebar can be coded as a double-spacebar by the game engine and vice versa.[10] Second, other potential causes such as pressing the w-key or hitting the border might happen after the double-spacebar but before the reset and then would be treated as the cause of the reset. Third, other discontinuities such as direction change or a ship death could be detected after the double-spacebar and treated as the effect of the double-spacebar.

Table 2 illustrates the pairings that were associated together in one run of the model after one game. Among other things the table illustrates the noise around the connection between a double-spacebar and the vulnerability being reset to 0: There are 4 cases where the model encoded that double-spacebar (D-Space in Table 2 – a potential cause) as followed by a vulnerability reset to 0 (Vuln=0 – a potential effect). However, twice the model noted that something else followed what it perceived as a double-spacebar: once it encoded a big point gain (Big Gain in Table 2) which can happen when the vulnerability reaches 10; once it encoded a vulnerability increment (Vuln-Inc in Table 2) because the game engine treated what it thought was a double-spacebar as a pair of single spacebars. There are also two cases where the model saw something else as preceding a vulnerability reset: once it had mis-encoded the double-spacebar as a pair of single spacebars (Space in Table 2); once a w key (w in Table 2) followed the double-space bar and before the vulnerability reset happened. Such mis-encodings of causes and effects reflect realistic limitations of ones ability to identify the causal structure of the environment given the underlying stochasticity of events and limits on measurements — even scientists make wrong inferences about what is causing what.

Only 6 of the 30 possible cause–effect pairings reflect true causal relationships. The build up of associative strength serves to accumulate evidence about which pairings might be real. Because the model will potentially change its actions on the basis of making a causal attribution it needs a rather strict threshold for attributing a cause–effect relationship. The model accepted a relationship as causal if the time to retrieve the effect primed by the potential cause is less than 25 ms. Given the parameter settings of the model, that threshold corresponds to a strength of association of 1.4. This idea of using retrieval time as a measure of fluency has been used in the past (for instance, Marewski & Schooler, 2011). A similar idea of threshold retrieval activation can be found in Altmann and Gray (2008).

4.2.3. Model success at identifying different causal relationships

Given the variability in what different models notice, models sometimes fail to identify a cause–effect relationship or they identify the wrong relationship, just as we assume is true of some subjects. One thing that influences the probability of success is the Noticing Probability. If a model fails to notice most of the cause–effect contingencies, it will fail to create the operators it needs. Recall that the probability of noticing the big point events varied for the model runs such that any probability between 0 and 1 was equally likely and the Noticing Probabilities for other events were scaled to be a fraction of that probability. Fig. 5a shows the success in

[10] In part this is because of motor noise. Also, because the ship is moving, the interval between when two shots that arrive at the fortress (which the game engine treats as the relevant interval) is not exactly the same as the interval between when the spacebar was pressed. Finally, the game engine encodes events in discrete 30th-of-a-second game ticks that are not aligned with model perception of time.

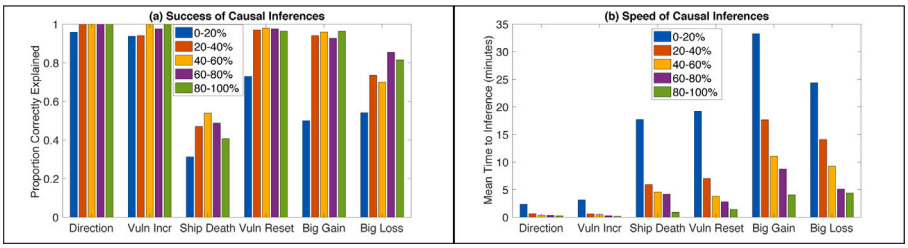


Fig. 5. Success of models at explaining different discontinuities for different ranges of Noticing Probability. (a) Probability of attributing correct cause. (b) Mean time to attribute a cause.

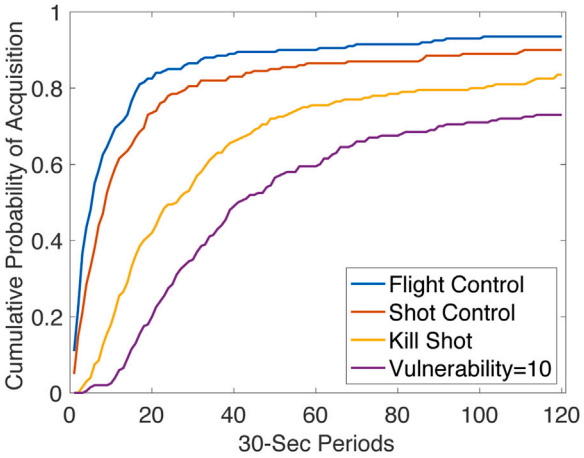


Fig. 6. Cumulative probability of acquiring an operator as a function of 30-second period in the experiment.

explaining the 6 discontinuities for ACT-R models with different ranges of Noticing Probabilities for the big point events. Fig. 5b shows the mean times to make those inferences. The models almost always identified the correct causes of direction changes and vulnerability increments and did so relatively quickly. Less than half of the ship death inferences were made, but this is because many models identified a big point loss as the effect of touching the border and this interfered with recognizing ship death as a distinct event. The models took the longest to recognize the causes of big point gains and losses. Models with a Noticing Probability less that 20% were less successful than models with a higher Noticing Probability. The effect of Noticing Probability greater than 20% was largely on time to make the inferences. Overall, 81% of the discontinuities were explained, 5% were attributed to incorrect sources, and 13% were never explained.

There is a strong relationship between Noticing Probability and the mean number of points achieved in a game (r = .65). The mean Noticing Probability in the 44 models that failed to reach 300 points on the 20th game was .231, whereas for the 156 models that reached the threshold it was .523. Thus, a potential explanation of individual differences in the subjects is differences in how well they attend to critical events in the game and how successful they are in noticing the causes. All the successful models make at least one of the two inferences involving touching the border and all of the other causal inferences. Only 4 of the unsuccessful models make all of these causal inferences.

4.2.4. Construction of operators from causal inferences

Once the model has identified the prerequisite causal relationships it can form operators for game play just as it did from instructions. However, these operators are planned piecemeal as the causal relationships are discovered. Fig. 6 shows the cumulative probability of having acquired various pieces of Fig. 4b as a function of position in the game, broken down into 30-second periods (6 periods per game, 120 such periods over 20 games):

1. Flight Control. Operators 2 and 5 in Fig. 4b are built once the model has made both the causal inference that w leads to inward movement and the inference that touching the border results in either deaths or point losses.
2. Shot Control. Operators 4 and 7 are built once the two inferences are made that spacebar leads to a vulnerability increment and that double-spacebar leads to a reset.
3. Kill Shot. Together the operators for flight control and shot control form the subtree on the left of Fig. 4b. The model typically will operate for some time according to this subtree, trying to avoid deaths and shoot fast enough to rapidly build up vulnerability but avoid resets. This leads to fairly stable behavior where vulnerability builds up but an occasional fast press

of the space bar (encoded as double-spacebar) will produce a big point gain. This enables discovery that double-spacebars can also cause big point gains. When the model makes this causal inference, it constructs operator 1 at the top of Fig. 4b to test for high vulnerability. It copies the left subtree to the right branch, replacing spacebar with a double-spacebar.

4. Vulnerability=10. The vulnerability will be typically higher than 10 when the model discovers that a double-spacebar can produce a big point gain. It remains to be determined what the vulnerability threshold is, if indeed there is a hard threshold. Initially, it uses the Controller to experiment with different thresholds to find settings that maximize the number of kills minus the number of resets. The Controller can hone the threshold down to a range, but it will never produce a hard threshold. However, the model also maintains in working memory what the highest vulnerability was when a double-shot produced a reset and what the lowest vulnerability was when a double-shot resulted in a kill. Typically, by experimentation the model will eventually have generated a double-shot producing a reset when the vulnerability is 9 and a double-shot producing a kill when the vulnerability is 10. Once it has this information in hand it sets 10 as the hard threshold for a double-shot.

By the end of the experiment 142 of the 156 successful models have identified a hard vulnerability threshold. It is possible to score over 300 points without identifying the hard threshold, as the remaining 14 models show. The majority of the subjects figure out that the vulnerability of 10 is the threshold for a double-shot. Looking at post-experiment reports by the subjects describing what they did, 33 of the 49 successful Discovery subjects volunteered the 10-rule (see the earlier reports for examples). Consistent with what they say, they seldom go beyond a vulnerability of 10 before killing the fortress (averaging .32 extra shots per kill in their last game). The remaining 16 Discovery subjects did not mention a 10 rule, but they were not explicitly asked about the rule and perhaps some knew the rule but did not mention it. Still, these 16 subjects average 1.67 extra shots per kill in their last game. The difference in extra shots between the two groups of subjects is quite significant ($t(47)= 3.41$, $p < .005$) and the subjects who report the 10-rule score better on average (2039 points vs. 1726 points, $t(47)= 3.08$, $p < .005$).

The successful behavior of the models depends on switching from random experimentation to a policy of avoiding deaths and building up vulnerability. Only when the models are behaving with some reliability do they get consistent enough information to identify the existence of a Kill Shot. Reliable play requires learning both shot control and flight control and models can learn about both before learning about the point consequences of such control. The models learn shot control because they value building up vulnerability before they connect that to gaining points. To test whether subjects also value building up vulnerability before they appreciate its connection to points, we looked for a reduction in number of resets before they experience their first kill of the fortress: 33 of the subjects took at least a minute to kill their first fortress. We split the time before their first kill into two equal periods, excluding the 30-second period that contained the kill. The average number of resets per minute reduced from 8.8 in the first period to 6.5 in the second period ($t(32)=2.82$, $p < .01$).

While the majority of the models learn flight control from observing point loss, some models experience many ship deaths without point losses because they have not built up any points to lose. 43 of the 156 successful models learn to avoid borders from this experience rather than point loss. There is evidence that some successful subjects adjust their navigation to avoid deaths before inferring that ship deaths mean lost points: We do not know when subjects make that inference about deaths leading to big point losses, but they cannot make it before experiencing such a point loss. Early deaths tend not to result in point losses because subjects have not accumulated any positive points (and points are bounded below by zero). Seventeen of the 49 subjects experience their first death with a point loss later than game 1 and these offer opportunities to test for a reduction of deaths in advance of knowing the point consequences of these deaths. 10 of these subjects experience their first death with a point loss in Game 2. These 10 subjects average 2.3 deaths in the first half of game 1 and 1.2 deaths in the second half (a significant difference – $t(9)=2.81$, $p < .01$). The other 7 subjects experience their first death with a point loss still later (6 in game 3, 1 in game 4). These subjects average 2.8 deaths in game 1 and 1.5 deaths in game 2, a marginally significant difference ($t(6) = 2.40$, $p = .053$, two-tailed). Combined, 16 of the 17 subjects reduce the number of ship deaths before experiencing their first point loss from a ship death ($p < .0005$, sign-test), indicating that they are learning to improve their flight control before seeing any consequences of deaths on points.

4.2.5. Discontinuities in behavior

When the model makes one of these changes to its operator structure, it shows sharp changes in relevant behavioral measures. We divided the games into 30-second periods and identified the period when a particular operator change occurred in each model. Fig. 7 plots 4 behavioral measures for 10 30-second periods before the operator inference, the period of the inference, and 10 periods after an inference. Part (a) shows a sharp drop in deaths with the construction of flight control operators. Part (b) shows a sharp increase in number of shots with the construction of shot control operators. Part (c) shows a sharp increase in number of fortresses killed after the constructing the operators that use vulnerability to make a Kill Shot. The kills continue to rise after the inference because this is followed at some variable period by the vulnerability=10 inference, which makes killing more efficient by avoiding unnecessary shots in building up vulnerability. Part (d) shows a sharp drop in extra shots per kill when the vulnerability=10 inference is made. Extra shots decrease prior to this inference, in part because the model is beginning to get a fix on the right vulnerability value with the Controller.

The fact that the various inferences are reflected in behavioral measures offers a way to determine whether subjects are making these inferences in a way that corresponds to the model. We can fit a 2-state hidden Markov model (HMM) to the behavioral measures and apply the HMM to each subject and assume that the subject has made the inference when they change state.[11] We can

[11] The assumption that there are only two states with respect to a measure is only an approximation as other things are changing that affect the measure as Fig. 7 reveals.

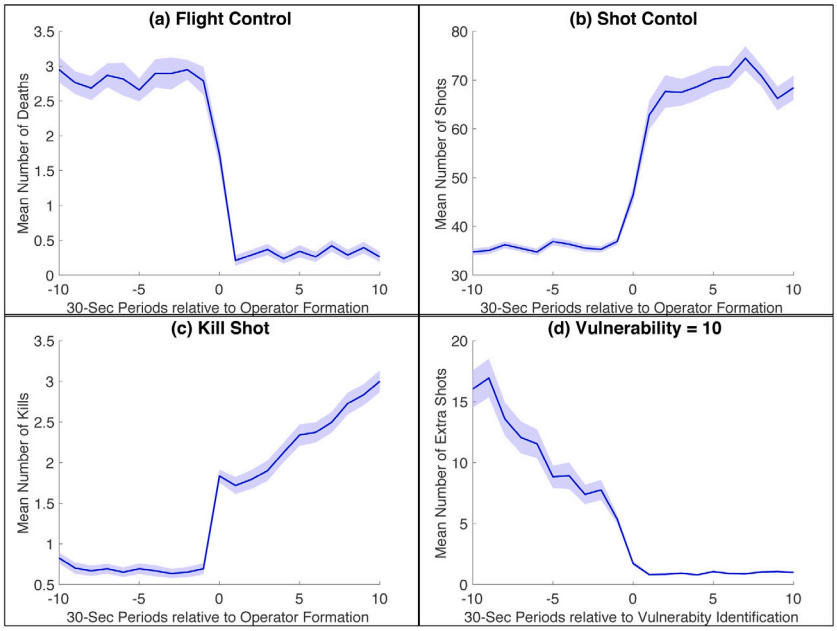


Fig. 7. Counts of events in game play as a function of when the models acquired different critical operators where 0 is the trial of acquisition. The shaded areas represent +/ − 1 standard deviation of the mean.

also fit the same 2-state HMM to the trace of the model's behavior (without using information about when the model actually made the inference). The measures being fitted are counts of events in a 30-second period and they were treated as Poisson distributed with a rate that was the expected mean number of these events in that state. In fitting the HMM, we estimated these two means as well as the probability of starting in state 1 and the probability of transitioning from state 1 to state 2 in going from one 30-second period to another. The estimation maximizes the probability of the observed counts X_j in the 120(j's) periods:

$$\begin{aligned} Prob = s1 \sum_{i=2}^{120} \left((1-t12)^{i-2} \times t12 \times \prod_{j=1}^{i-1} P(X_j|\lambda 1) \times \prod_{j=i}^{120} P(X_j|\lambda 2) \right) \\ + \; s1 \times (1-t12)^{119} \times \prod_{j=1}^{120} P(X_j|\lambda 1) \\ + \; (1-s1) \times \prod_{j=1}^{120} P(X_j|\lambda 2) \end{aligned} \tag{2}$$

where s1 is the probability of starting in state 1, t12 is the probability of transitioning from state 1 to state 2, $\lambda 1$ is mean number of events in state 1, $\lambda 2$ is mean number of events in state 2, and $P(X|\lambda)$ is the probability of observing a count of X from a Poisson distribution with a mean of λ.

The parameter estimates for the fits to model and subject data are given in Table 3. At a general level the parameters are similar between subjects and models — high probability of starting in state 1, low probabilities of transitioning to state 2 in any 30-second period. Similar general comparisons can be made of the mean counts of events in a state. However, the parameters for models and subjects are not identical and one can ask whether the differences are significant. To address this question, we did a split half fit, estimating parameters from the odd subjects to predict the even subjects and comparing this with how well the model parameters predicted the even subjects. Similarly, we estimated parameters from the even subjects and compared how well they predicted the odd subjects versus the model parameters. The number of subjects out of 49 predicted better by the model was 17 for deaths, 15 for shots, 26 for kills, and 25 for extras. The model does slightly better than the expected chance level (24.5) for all kills and extras but is worse for deaths and shots. If a sign test was used, the model is significantly worse for both deaths (17 out of 49, $p < .05$) and shots (15 out of 49, $p < .01$). In both cases the model is making the transition to state 2 faster. We could have slowed this transition by lowering the Noticing Probability for the discontinuities that are relevant to flight and shot control. However, our goal is not to provide a careful fit to data of these particular subjects, but to show that general properties of their behavior can be accounted for in this framework given plausible parameter settings.

Given these parameter estimates and the observed counts in the 30-second periods, we identified for each subject or model run and each measure, the 30-second period when the HMM first had a greater than .5 probability of being in state 2. There tended to be a 30-second period when the probability of being in state 2 took a big jump. The average probability jumped .14 to .82

Table 3
Parameter estimates to the 2-state HMM.

(a) Two state-parameters estimated in fit to model runs				
Behavioral measures	Start in state 1	Transition to state 2	Mean count in state 1	Mean count in state 2
Deaths	1.000	0.131	2.7	0.3
Shots	0.998	0.087	36.7	62.2
Kills	1.000	0.031	0.9	4.3
Extras	1.000	0.023	18.9	1.2
(b) Two-state parameters estimated in fit to subjects				
Behavioral measures	Start in state 1.	Transition to state 2	Mean count in state 1	Mean count in state 2
Deaths	1.000	0.039	1.5	0.2
Shots	0.845	0.026	29.2	67.9
Kills	1.000	0.024	0.7	4.4
Extras	0.816	0.018	23.1	1.5

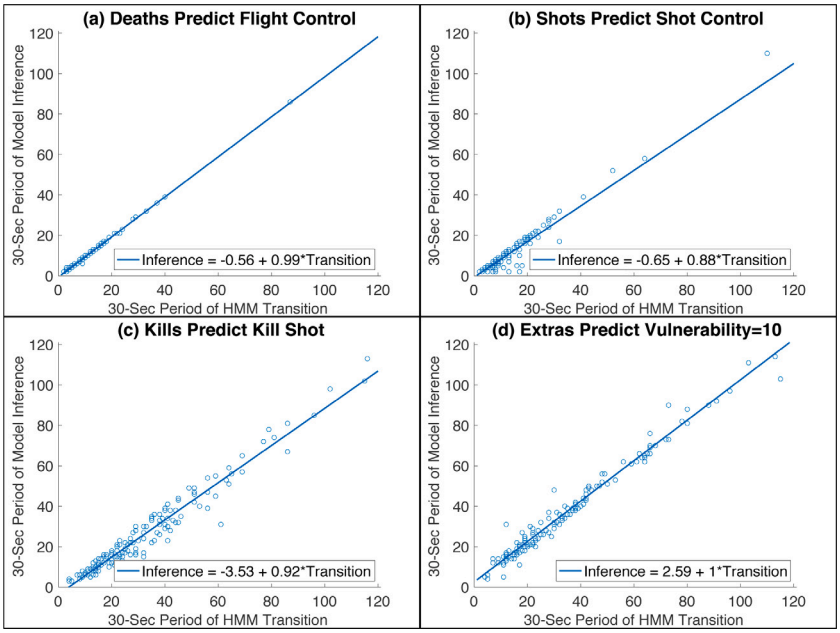


Fig. 8. Correspondence between when the 2-state HMM showed a transition between the 2 states (x-axis) and when the model actually made that inference.

for subjects and from .12 to .86 for the models at this transition point. Fig. 8 shows that this point of HMM transition does well at estimating when the ACT-R model actually made the associated inference. We compared what happened before and after the 30-second period of state transition in subjects and model runs. The model implies there should be a sharp change in the behavioral measures, something like the step functions in Fig. 7. However, subjects would not show such abrupt changes if they were just gradually improving their performance with respect to these measures. Fig. 9 plots for the subjects and the models, the behavioral measures for 10 periods before and after the 30-second period in which the transition happened (similar to the backwards learning curves, e.g., Bower & Trabasso, 1963; Hayes, 1953). In each case both the subject and the model data show discrete behavioral changes at the point of transition. The model has a higher death rate than subjects before the transition in Fig. 9a, consistent with the poor fit of model parameters to subject data. Otherwise model and subjects are similar before and after state switch on the four measures. Successful Discovery subjects do seem to be making model-like inferences about the game as they played it, and these inferences resulted in discrete changes in game play like the model.

The model tends to learn about the Kill Shot only after it has learned shot control and flight control and achieved stable behavior. The HMM shifts to state 2 are consistent with this. Only one of the 156 models achieves state 2 with respect to kills (Fig. 8c) before achieving state 2 for both deaths and shots (Figs. 8a and b). The subject state changes confirm the same thing. Only one of the 49 subjects achieved state 2 with respect to kills before achieving state 2 for both deaths and shots.

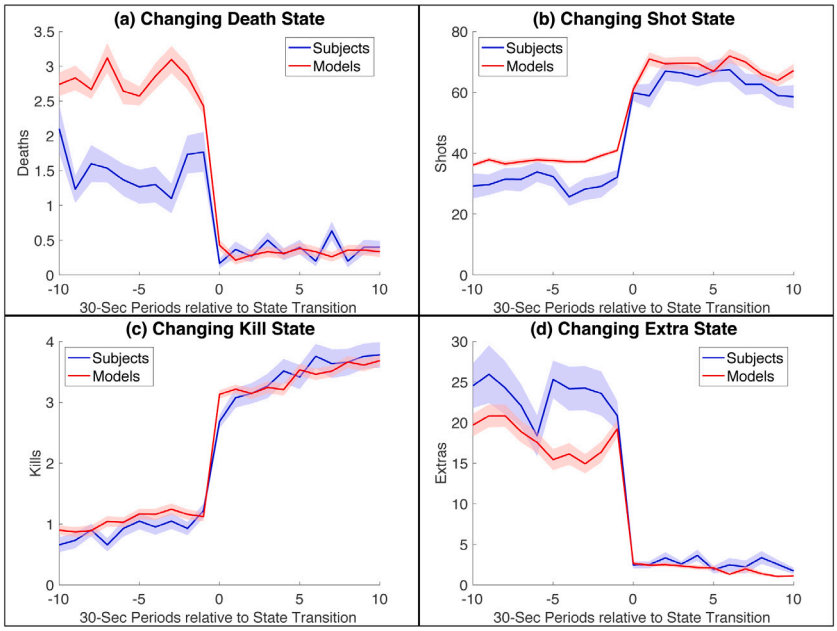


Fig. 9. Counts of events in game play as a function of when the HMM makes a transition to state 2. The shaded areas represent +/ − 1 standard deviation of the mean.

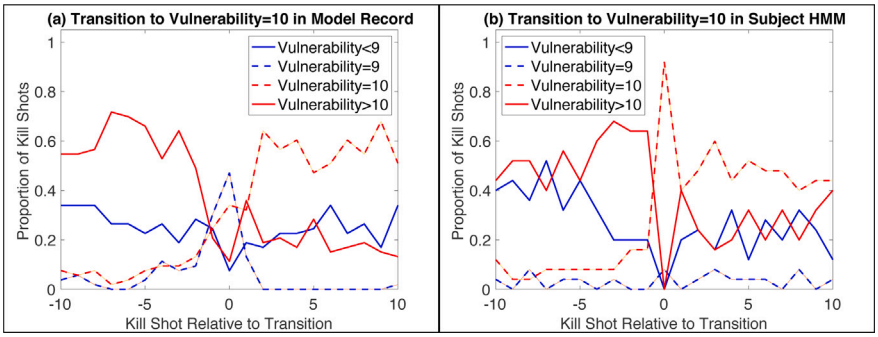


Fig. 10. Proportions of different vulnerabilities at which kill shots were issued relative to the point of insight (x=0) that 10 is the threshold for a kill shot.

4.2.6. Analysis of kill shots around the vulnerability=10 transition

A kill shot (two taps of the spacebar less that 250 ms. apart) resets the vulnerability to 0 whether successful (when vulnerability is 10 or higher) in destroying the fortress or not. Kill shots above or below a vulnerability of 10 provide evidence that the player has not yet grasped that 10 is the right threshold for a kill shot. Kill shots can also occur unintentionally when subjects are just trying to pace their shots to build up the vulnerability but fire a pair of shots so close together that the game engine treats the pair as a kill shot. We examined the vulnerabilities at which kill shots were issued. The pattern of these vulnerabilities provides a basis for comparing subjects and models with respect to how they come to the insight that 10 is the threshold for a kill shot.

Fig. 10a shows the vulnerabilities of kill shots in the model around the point when it adopts vulnerability=10 as the trigger for a Kill Shot. It will do so only after it has identified both that a kill shot at 9 results in a reset without increase in points and that a kill shot at 10 produces a kill and an increase in points. Therefore, we have plotted proportion of kill shots at 9 and 10 separately and aggregated all cases below 9 and separately aggregated all cases above 10. The value of 0 on the x-axis corresponds to the kill shot that occurred just before the transition to using vulnerability of 10 as the threshold for a kill shot. At x=0, the majority of the models (80.3%) made their kill shot at 9 or 10, which provides one of the two critical pieces of information needed to determine that the correct threshold was 10. Occasionally, the critical identification occurs on the previous kill shot but the model did not go through the steps of inference to set the kill policy in place before the next kill shot. After the transition the model shows a big increase in the proportion of kill shots at vulnerability 10. There continues to be some kill shots below 9 and some above 10. The

kill shots below 9 reflect unintended resets as the model tries to get as close to the 250 ms. pacing as it can. Most of the kill shots above 10 are at 11, which occurs because the model fires an extra shot before it registers that the vulnerability has reached 10.

Unlike the model we do not know exactly when subjects start deliberately using a Kill Shot, but the HMM fit for Vulnerability=10 (Fig. 9d) provides a hypothesis as to when this change takes place. Since the HMM fits are only to a 30-second period and only identify the most probable period, these estimates are not as precise or accurate as what we have for the model. Also, some models (and presumably some subjects) make a transition in one 30-second period but do not show the behavioral change that is used by the HMM until the next 30-second period. We looked for the first kill shot at 9 or 10 in the period of HMM transition. If there were none in that period, we looked at last kill shot at 9 or 10 in the 30-second period before that transition.[12] Fig. 10b shows the changes in different categories of kill shots relative to that point. Like the model there is a sharp rise in kill shots at 10 after the transition. However, at the point of transition (x=0) the kill shots are almost all at a vulnerability of 10 unlike the models who have an even mixture of 9s and 10s. At the transition point, 31 models issued kill shots at a vulnerability of 9 at x=0 and 26 issued kill shots at 10. In contrast, just 2 subjects issued kill shots at 9 and 23 subjects issued their kill shot at 10 ($\chi^2(1)=15.55$; $p < .0001$)

In the case of a model, a kill shot at 9 is just as likely to be the piece of evidence that pushes it to adopt vulnerability=10 rule as is a kill shot at 10. In contrast, many subjects seem to be transitioning to using a vulnerability of 10 as the threshold for a kill shot without ever testing that a kill shot at 9 might work. In fact, 12 subjects never try a kill shot at 9 before they have settled into using 10 for their kill threshold (something that never happens in models). One subject never experienced a kill shot at 9 failing to produce a kill in the entire 20 games, but was using 10 as the threshold for a kill shot. It seems that numerical knowledge, such as the base-10 system, is influencing subjects in a way that is not happening in the model.

5. Discussion

The basis for successful performance in both the Instruction and Discovery Conditions was an understanding of the causal structure of the game. The instructions (see Appendix A) described the causal structure of the game, not how to do well in the game. It was relatively easy to go from this information to a strategy (operators in the model, see Fig. 4) for playing the game. The instruction subjects still had the challenge of achieving the speed and precision of action to score well. The Discovery subjects first faced the challenge of figuring out the causal structure of the game. Like many situations where people learn by exploration, part of the challenge was the multiplicity of possible causes and effects. The models, and we assume the humans, varied in how attentive they were to relevant effects. The models used the buildup of associative strength to identify what pairings of possible causes and effects reflected real causal relationships. If and when the right causal relationships were identified, the Discovery models progressed through the same steps as the Instruction models. By the end of the 20 games the successful Discovery subjects and models were approaching the performance of the successful Instructed players.

We had expected that there would be an effect of the Semantics manipulation, but the lack of effect now seems understandable from the perspective of the model. The obvious place for the Semantics manipulation to affect success would be the Noticing Probability for critical discontinuities. While kills and deaths were accompanied by explosions in the Semantics condition, which might grab the subject attention, they were also accompanied by changes in the points determining the pay subjects received, which surely should grab subject attention. The remaining discontinuities involved changes in ship/ball movement and changes in the vulnerability counter (labeled VLNER in Semantics and COUNTER in the no Semantics condition). There is no reason to expect differences in attention to these. Once the models made causal inferences, further improvement in play reflected speed up and tuning of actions, which were unconscious processes. Dubey et al. (2018) did find an effect of changing game elements. Some of their manipulations seemed to change the desirability of certain actions (e.g. climbing a ladder versus climbing flames), which were prerequisites to success in their game. Our model also valued intermediate actions that were perquisites to accumulating points, namely incrementing vulnerability and avoiding deaths. Subjects gave evidence of valuing these things also, even before they experienced point consequences. There was not an apparent effect of the semantics of the game in how much they valued these intermediate effects.

While we failed to find an effect of our semantics manipulation, detailed analysis of the vulnerabilities-at-reset (Fig. 10) did find evidence that subjects prior knowledge did affect their learning. Many subjects treated a vulnerability of 10 as special and when they found that they could get a Kill Shot at this value, they did not bother to explore whether a kill shot at a lower vulnerability would also have worked. The model in contrast did not have the background to treat 10 as special and only came to use a threshold of 10 for the kill shot after it had determined that lower thresholds would not work. This is one indication that the processes by which subjects come to discrete transitions in their behavior may not always have been identical to the model.

Starting from causal knowledge provides a conceptual basis for the rules one is acting by and this can have potential benefits. For instance, Taatgen, Huss, Dickison, and Anderson (2008) compared the kind of list-of-steps instructions that were used in Boeing Flight management system with instruction that focused on the effects of various actions. The list of steps basically specified a sequence of operators. These list-of-steps instructions proved more brittle than the instructions focused on the causal structure (see also Halasz & Moran; Kieras & Bovair, 1984). Failure to encode or remember a step left subjects with the list knowledge floundering, whereas subjects informed of what their actions did could try to fill in the gap. It is quite possible that some of our Instruction subjects

[12] This analysis is restricted to subjects for whom such a kill shot can be found. It is also restricted to subjects for which both a Kill shot transition and a Vulnerability=0 transitions can be identified and for whom the Kill shot transition comes first. It is further restricted to subjects who have at least 10 kill shots before and after the transition. There were 25 subjects who met this constraints. To have comparable model data in Fig. 10a we applied to same restrictions to HMM fits to model runs and there were 71 models that met these constraints.

did not encode all the critical causal information in the game and discovered some of the causal pieces while playing. This might be one of the reasons the Instruction subjects did not start off as well as the Instruction models, which started with a complete encoding of the critical causal relationships.

The distinction between causal knowledge of devices and the action rules for using these devices has similarity to the distinction between conceptual and procedural knowledge in discussions of mathematics education. Educators often propose that mathematical skills will be more robust if grounded in a conceptual understanding and this is one of reasons for advocating discovery-based learning. However, as Rittle-Johnson, Schneider, and Star (2015) argue, the situation is more complex with conceptual knowledge often being shallow and development of procedural skills often promoting deeper conceptual knowledge. In this task, we see that identifying the Kill Shot depends on achieving skill at timing shots and flying safely.

As noted in the Introduction, there has been considerable interest in the mathematics education literature in the potential benefits of using examples as instructional devices. While examples are often accompanied with explanations, examples of problem solutions without explanation have also been used, leaving the student the task of figuring out how they work. Understanding a mathematical example has similarities to understanding a video game by discovery, but in the mathematics case the example has been fashioned by an instructor while the corresponding experiences in the video game are the result of the player's actions. It has been argued (Aleven & Koedinger, 2002) that a major function of verbal instruction when it accompanies the example is to draw learners attention to critical aspects of the example, something that can also be achieved by visual cueing with color (Lee, Betts, & Anderson, 2017). Such attentional manipulations could have their effect in our model by changing the Noticing Probabilities. We would expect that success at discovery in video games would be impacted by manipulations that drew attention to or away from the critical cause–effect contingencies in the game.

One of the challenges in mathematics education, which occurs with any combination of instruction and examples, is that students will learn buggy rules (Siegler, 2009). One of the arguments for a conceptual understanding is that helps students avoid such errors and correct them. Still such errors can become entrenched and present a challenge to overcome. Research on correcting such entrenched errors is limited by many factors, including that one has no control over when and how such errors occur. Video games offer a domain for understanding how one recovers from entrenched rules that does not have this limitation. One can promote erroneous inferences by drawing attention to misleading features or indeed by changing how the game works in the middle of game play. We should note that the current model would not be able to recover from an operator that it has committed to and that it is difficult to change the behavior of any ACT-R model after it has proceduralized its knowledge.

There has been a considerable debate between associative and cognitive model explanations of causal learning (for reviews see Holyoak & Cheng, 2011; Shanks, 2007). The approach here reflects a combination of the two. In a rich multi-cue, multi-effect world it becomes a great challenge to keep straight what the cause–effect connections might be. Here, the build-up of associative strengths among elements, as occurs automatically in ACT-R, is an effective way to detect potential cause–effect relationships. However, once detected the model involves deliberative processes in determining how to respond to these potential causal relationships. Lebiere and Wallach (2000) apply a similar ACT-R combination of implicit associative learning and explicit creation of declarative knowledge to explain the implicit and explicit aspects of sequence learning.

Interestingly, once the models deliberately build up the operator structures (Fig. 4), further improvement depends on the continuous and unconscious processes in ACT-R of production compilation and control tuning. Provided the correct operators are encoded both Instruction and Discovery models will converge to the same ultimate state resulting in identical behavior. Thus, while the formation of explicit knowledge about the game is critical, there are preceding implicit processes leading to the discovery of that knowledge and subsequent implicit processes tuning of that knowledge to achieve high performance.

Acknowledgments

This work was supported by the Office of Naval Research Grant N00014-15-1-2151. We thank Cvetomir Dimov for his comments on the paper. The model, data, and analyses that were used to create the figures in the paper are available at http://act-r.psy.cmu.edu/?post_type=publications&p=32163.

Appendix A. Instructions to subjects in various conditions

Figs. A.11 and A.12 show the instruction screens for subjects in the semantics and no-semantics Instruction conditions. These instructions were not deliberately designed to describe the causal structure even though that is what they do on retrospect. They were streamlined from the instructions in Anderson et al. (2019), which did have some operational game-playing advice. The reason for the streamlining to make it easier to have equivalent information for both the Semantics and No-Semantics conditions. Fig. A.13 shows the same screen was shown to both the semantics and no-semantics Discovery conditions.

Appendix B. Model review

A model used in this paper is implemented in at ACT-R architecture and is available at http://act-r.psy.cmu.edu/?post_type=publications&p=32163. It is the same model as in Anderson et al. (2019) except for the additions described in the paper concerned with causal inference and converting causal relationships into operators. All the parameter settings of the current model are the same as the older model.

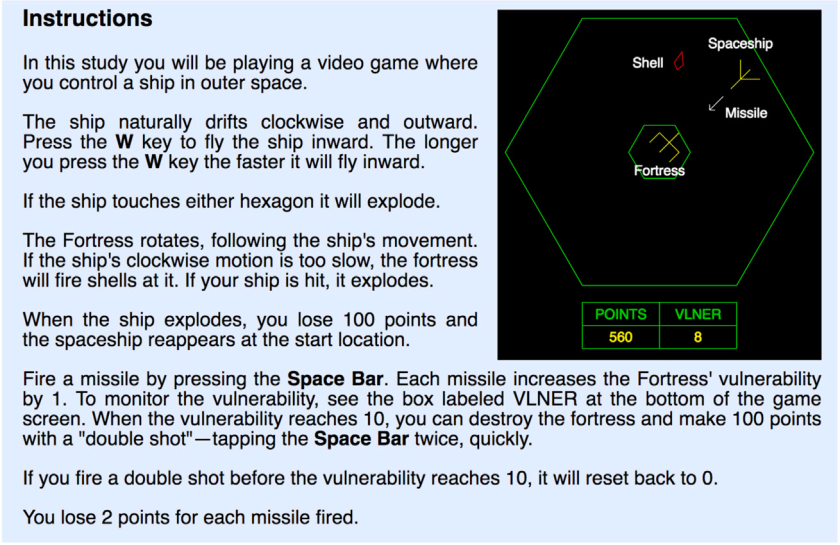


Fig. A.11. Screen shot of instructions given to subjects in the Semantics Instruction condition.

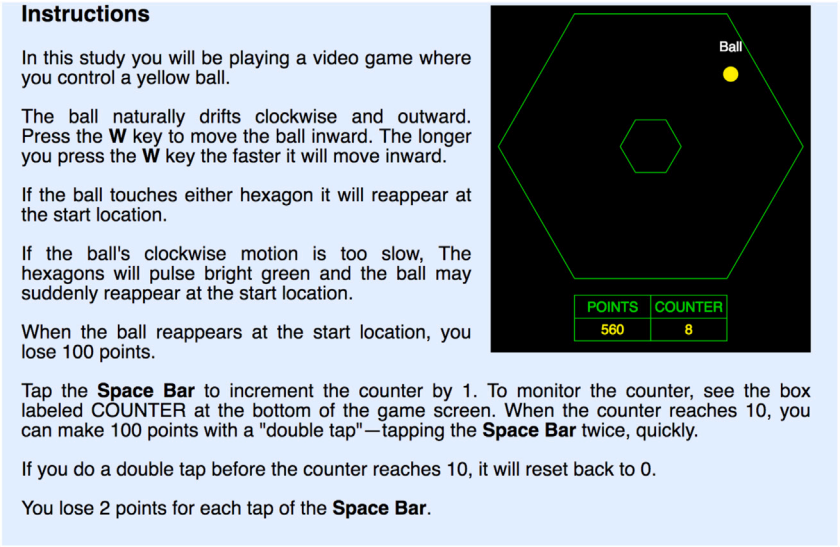


Fig. A.12. Screen shot of instructions given to subjects in the No-Semantics Instruction condition.

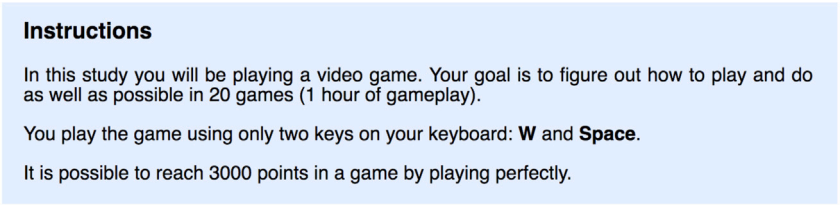


Fig. A.13. Screen shot of instructions given to subjects in the two Discovery conditions.

ACT-R is a collection of independent computational systems, called modules, that run asynchronously and can communicate information among themselves through information placed in limited capacity working memories, call buffers. Fig. B.14 shows the modules of relevance to the performance of the model in this experiment. For a detailed discussion of the module structure of ACT-R

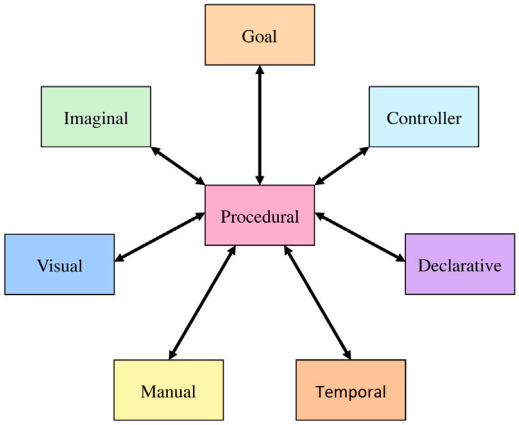


Fig. B.14. The relevant ACT-R modules for playing Space Fortress.

see Anderson (2007) and for a description of their role in models of game playing see Anderson et al. (2019). The review here is briefer.

Procedural: The central piece in any ACT-R model is the procedural module, which consists of production rules, which monitor information in the buffers of other modules and, if certain patterns of information appear, make requests of other modules. The default time for production execution is 50 ms, which places a limit on speed of processing. Early in game play, much of the model's activity involves productions recognizing operator information in the Declarative buffer, making appropriate tests of the state of the game, and then making appropriate requests to the Manual module for actions. With repetition new direct-action rules are acquired that bypass declarative retrieval and directly respond to information in the visual buffer with action requests.

These direct-action rules develop through a process called production compilation (e.g. Anderson et al., 2004; Taatgen & Anderson, 2002): If one production responds to a situation it recognizes with request for memory retrieval and a second production uses the retrieved information to make a request of other modules (for instance, to press a key), these two productions can be collapsed into one production. That production will bypass the retrieval and respond to the situation of the first production with the request of the second, producing a considerable speed up. Much of the early improvement in the Instruction condition is produced by this speed up, which allows the model to quickly react to new situations. It takes repeated compilations of a production for it to build up enough strength (called utility in ACT-R) to compete with an existing rule. The rate of this learning is controlled by a learning parameter α in a simple difference learning equation.

$$Utility_n = Utility_{n-1} + \alpha(Value - Utility_{n-1}) \tag{B.1}$$

In this game with a premium on quick actions, the utility of a production reflects how long it takes to press the next key — the less time the higher the utility of the production. Thus, the quantity $Value$ in the equation is a fixed reward for the key press minus the time from the production firing to the pressing of the key. The value of a newly compiled production starts at 0. Each time it is re-compiled the utility will increase using the value of the first production from which it is compiled. As it approaches the utility of this parent production, stochasticity in production selection will allow it to be selected rather than the parent and then $Value$ will be based on its performance. Because the compiled rule results in faster actions, this value will be greater than the parent production from which it was compiled and will come to dominate. As in Anderson et al. (2019), we varied this learning parameter α to reflect individual differences using the same values: .025, .05, .1, .2, and .3. It is easier to assess the effect of this parameter in the Instruction models where there is not the added variability of when causal inferences are made. Here the means scores for the increasing values of α are 1886 points, 1951 points, 2018 points, 2026 points, and 2028 points.

Visual: We have augmented the standard ACT-R visual module with a game-state buffer that maintains the current position of the screen — i.e., whether ship and fortress are alive, where the ship is if alive, and what the current vulnerability is.

Manual: ACT-R's manual module can issue key presses for specific durations. Durations of the thrust press will control how much thrust is added to the ships position.

Temporal: The time between shots is controlled by ACT-R's Temporal module that ticks at somewhat random and increasing intervals producing the typical logarithmic discrimination for temporal judgments (Taatgen, Van Rijn, & Anderson, 2007).

Declarative: Much of the ACT-R theory is concerned with how information is retrieved from declarative memory and placed in the Declarative buffer. The retrieval time parameter has been set such that the average time to retrieve an operator is 100 ms. Thus, production compilation, which eliminates a retrieval and one production can save about 100+50=150 ms.

Imaginal: The Imaginal buffer serves as a working memory for significant events like resets and deaths, which are used to inform the Controller. It is also used to construct new information like inferred causal relationships and the operators that are created in response.

Goal: The Goal module keeps track of where it is in exploring the game and in interpreting the game strategies in Fig. 4. It also maintains a number of parameters that control which path it takes through that strategy tree. These include how close the

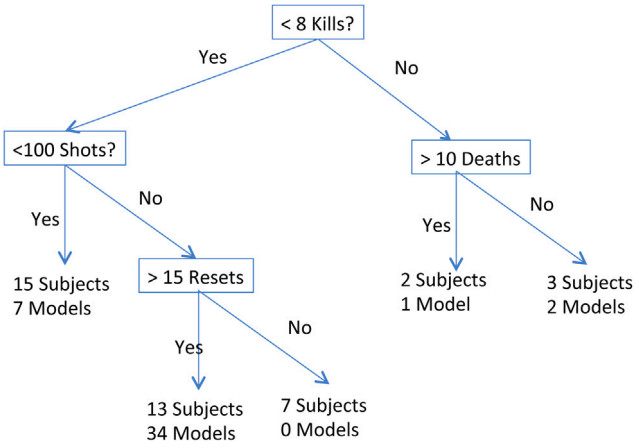


Fig. C.15. A breakdown of the behavioral problems that were preventing subjects and models from achieving 300 points in the 20th game.

model can drift towards the outer border before thrusting, how much time must pass between shots to avoid resets, and what the vulnerability threshold is for issuing a double-shot. It also controls how long to press the thrust key (longer thrusts produce greater changes in flight direction).

Controller: The Controller was the new module that Anderson et al. (2019) introduced to learn good settings for the control parameters in the goal module. In the Instruction condition it has responsibility for setting the time to the outer border at which to thrust, the duration of these thrusts, and the time that must pass between shots. In the Discovery condition it is also responsible for setting the vulnerability threshold until the hard threshold of 10 is discovered. As described in Anderson et al. (2019) the Controller explores a range of potential values looking for the value that gives it the maximum combination of good outcomes and bad outcomes. Each dimension being tracked is set with its own range of values and its own good and bad outcomes. As in Anderson et al. (2019), the Controller for time from border searches a range from 1 to 5 s and counts fortress kills as positive events and ship deaths as bad events. The Controller for duration of thrusts searches a range from 0.067 s to .167 s and also counts kills and deaths as good and bad events. The Controller for shot timing searches from 9 to 18 ticks of the temporal module (approx. 150 to 500 ms) and counts hits as positive events and resets as negative events. The vulnerability Controller used in the Discovery condition is new: it searches the range from 5 to 15 shots and considers kills as positive events and resets without kills as negative events.

As described in Anderson et al. (2019), the Controller tries to estimate a quadratic function V(x) that describe the values of a settings, defined as number of good events minus number bad events, per second as function of values in its range. It selects different settings to try for random intervals (averaging 10 s) according to a probability distribution that reflects a softmax equation:

$$P(x) = \frac{e^{V(x)/temp}}{\int_{min}^{max} e^{V(x)/temp} dx} \tag{B.2}$$

The parameter *temp* is the temperature. The choices will tend to explore the full range of possible values when *temp* is large and exploit the current best estimate when *temp* is small. The temperature decreases as a function of time t according to the function

$$temp(t) = A/(1 + Bt) \tag{B.3}$$

where A is the initial temperature and B scales the passage of time (in the model B is set to 1/180 of a second, reflecting a 180 s game). We ran models with a range of initial temperatures, which were crossed with values of the production learning parameter α (the early results for α settings were averaged over temperature choices). As in Anderson et al. (2019) the initial temperature settings A were .1, .25, .5, 1, and 2 and resulted in mean scores of 1968 points, 1997 points, 2049 points, 1991 points, and 1883 points. As in Anderson et al. (2019) intermediate values are best — small values of temperature can lead to premature convergence on non-optimal values and high values can lead to a failure to converge in time. The standard deviations of these scores (over the 40 models at each initial temperature) are 281, 216, 108, 97, and 128. The high variability at the lower temperatures reflects cases where some models converge on poor values and score poorly and other models converge tightly around optimal values and score well.

Appendix C. Analysis of unsuccessful subjects and models in the discovery condition

We examined the behavioral problems in subjects and models that prevented them from reaching 300 points. In the case of the models we can determine what was causing these behavioral problems, with the possibility that similar factors were behind the subjects' behavioral problems. Fig. C.15 illustrates the breakdown of behavioral reasons for low scores. The most common reason subjects and models failed to reach the 300-point mark is that they did not kill enough fortresses (left branch of Fig. C.15). Since it takes at least 12 shots to destroy a fortress and each shot costs 2 points, the most net points one can get from killing a fortress is

76 points. This means that it takes at least 4 kills to reach 300 points. More realistically, 5 kills is the minimum because of wasted shots from resets, going beyond vulnerability threshold, and shots at the end of a game that do not complete a kill. In addition, if the ship dies even once it will take 1 or 2 further kills to compensate for that 100-point loss. No successful subject or model failed to kill at least 8 fortresses in their last game. On the other hand, 35 of the 40 unsuccessful subjects and 41 of the 44 unsuccessful models failed to reach this threshold. The 35 subjects average 3.0 kills and the 41 models averaged 4.3 kills. 14 of these subjects and 7 of these models had 0 kills.

The left part of Fig. C.15 unpacks why these subjects and models failed to achieve enough kills. No successful subject or model produced less that 100 shots in the final game whereas this was true of 15 subjects and 7 models. The 7 models never fired a shot in the final game, while the 15 subjects averaged only 35 shots (subjects were required to issue some actions for the game to continue). All of these 7 models had made incorrect causal inferences that had incompatible behavioral implications[13] and then gave up either just shooting or both shooting and navigating. The remaining low-kill subjects averaged 325 shots and the remaining low-kill models averaged 363 shots. The most common problem for these subjects and models is that they were too often resetting the vulnerability before reaching 10. No successful subject had more than 15 resets in the final game and there was only 1 successful model with more (it had 17 resets). There were 13 high-reset subjects who averaged 224 resets in the final game (median 43) and the 34 high-reset models averaged 102 resets (median 67). 10 of the 33 models failed to make the causal inferences necessary for constructing the shot control operator and they were experimenting with firing until the end. The other 24 models had failed to converge on a consistent setting for shot control by the 20th game. There were remaining 7 low-kill subjects who had relatively few resets (average of 5.1) but were making many extra shots beyond the threshold for a kill. Successful subjects and models average less than one extra shot per kill while these 7 subjects were averaging a 64.6 extra shots per kill. This meant they lost more points on extras shots than they would get for the kill. While there were no models in this branch of Fig. C.15, models can suffer high levels of extra shots before they discover the concept of vulnerability threshold.

The right part of Fig. C.15 analyzes the 5 subjects and 3 models who failed to reach 300 points and got more than 8 kills (subjects averaged 13 kills and models 10 kills). 2 of these subjects and 1 model were suffering more than 10 deaths, which is more than any successful subject or model. The two subjects suffered 11 and 14 deaths and the model 22 deaths, which took away most the points they were getting for their kills. The model had failed to infer the rule for flight control and we assume that was also true of the 2 subjects. The 3 remaining subjects suffered enough resets (an average of 23.7) and extra shots (an average of 13.3 per kill) to prevent them from reaching 300 points, as was also true of the 2 remaining models (an average of 47.5 resets and 2.6 extra shots per kill). Both of the models had not converged on a safe range for pacing their shots.

We fit the same HMM models to these subjects that we had fit to the good subjects, constraining the parameters to correspond to those of the good subjects. All but 14 of the subjects were credited with mastering at least 1 of the 4 rules: 18 were credited with achieving flight control, 17 with shot control, 4 with acquiring the Kill Shot, and one subject with learning the vulnerability=10 threshold. The 4 subjects credited with the Kill Shot are all in the right branch of Fig. C.15. One had also learned the vulnerability=10 rule but had not mastered flight control and had too many deaths. The other 3 had not mastered the vulnerability=10 rule and had too many extra shots. This pattern is like what we see for the 44 poor performing models where all but 11 master at least one rule: 31 master flight control, 27 master shot control, 9 master the Kill Shot, and 4 learn the Vulnerability=10 threshold. It seems that like the models, subjects perform poorly because they do not correctly identify all the rules and display the same ordering of difficulty over the 4 rules. As discussed in the main text, the Noticing Probabilities were lower for unsuccessful models (.231) than for successful models (.523). The data from the unsuccessful subjects are consistent with the view that their difficulties stem from failing to notice critical relationships in the game.

Appendix D. Associative strength

This section describes in detail how associative strength builds between factors (preceding events) and effects (observed discontinuities). Each effect of interest is stored as a chunk in ACT-R's declarative memory. Each time a factor-effect pairing is noticed the factor is placed in the goal buffer and the effect chunk is retrieved. This retrieval will increase the strength of association between the factor and the effect. If that associative strength reaches a threshold the inference will be made that the factor is the cause of the discontinuity.

Thus, the inference of a causal relationship is determined by the associative learning mechanisms in ACT-R. In ACT-R the associative strength between a source j (in this case a factor) and a chunk i (in this case a discontinuity) is an estimate of the log likelihood ratio that the cue would be present if the effect is needed:

$$S_{ji} = log\left(\frac{P(j|i)}{P(j|\neg i)}\right) \tag{D.1}$$

In the experiment, this is a measure of how good a predictor the factor is of the effect.

Associative learning has not been a part of recent ACT-R architectures, but we have updated the current architecture to be close to the implementation that was part of the original ACT-R architecture (Anderson & Lebiere, 1998). The current implementation

[13] The 7 problematic causal inferences all involved attributing one correct and one incorrect effect to the same cause. There was one case of inferring that a double-spacebar caused both a big point gain and a big point loss, one case of inferring that a single spacebar both incremented vulnerability and caused a ship death, two cases of inferring that hitting the border both caused a ship death and a big point gain, and three cases of inferring that a single spacebar caused both vulnerability increment and big point gain.

Table D.4
Example calculations of S_{ji}.

(a)				
	Vuln-Inc & Spacebar	Vuln-Inc	˜Vuln-Inc & Spacebar	˜Vuln-Inc
Frequency	37	39	2	408
Decayed	5.50	5.80	0.30	60.65
Plus prior	5.60	6.80	0.40	61.65
	P(Spacebar| Vuln-Inc) = 0.824		P(Spacebar| ˜Vuln-Inc) = 0.006	
Association strength between Spacebar and Vuln-Inc = 4.85.				

(b)				
	Vuln-Zero & Spacebar	Vuln-Zero	˜Vuln-Zero & Spacebar	˜Vuln-Zero
Frequency	1	4	38	443
Decayed	0.15	0.59	5.65	65.86
Plus prior	0.25	1.59	5.75	66.86
	P(Spacebar| Vuln-Zero) = 0.156		P(Spacebar| ˜Vuln-Zero) = 0.086	
Association strength between Spacebar and Vuln-Zero = 0.6.				

involved rules for estimating the log likelihood that defines strength of association: $P(j|i)$ is a Bayesian estimate based on the frequency that i and j occur together and the overall frequency of i:

$$P(j|i) = \frac{p \times W + Freq(i\&j)}{W + Freq(i)} \tag{D.2}$$

The term p in this equation is a prior estimate for the unconditional probability of a factor and W is the weight placed on that prior. Thus, the probability estimate starts at p and moves to the empirically observed proportion as experience accumulates. Similarly, the estimate of $P(j|\neg i)$ is based on the frequency that j occurs without i together and the overall frequency that i does not occur:

$$P(j|\neg i) = \frac{p \times W + Freq(\neg i\&j)}{W + Freq(\neg i)} \tag{D.3}$$

The frequencies in this formulation can be related to the quantities in causal framework of Table 1: $Freq(i\&j)$ is a, $Freq(i)$ is $a + c$, $Freq(\neg i\&j)$ is b, $Freq(\neg i)$ is $b + d$. Note that $\neg i$ includes many things — all the other chunks that the system may create, not just those involved in causal inference. In particular, it includes the operator chunks that the model initially retrieves to guide it actions.

Unlike the original formulation of associative strength, the frequencies have a decay component such that older experiences have less of an effect than more frequent experiences. Thus, the frequency of a category X is calculated

$$F(X) = \sum_{k \, \epsilon \, Xevents} t_k^{-d} \tag{D.4}$$

where $Xevents$ are the experiences of category X, t_k is the time since the event, and d is a decay rate conventionally set to .5 in the ACT-R model. As an illustration, Table D.4 shows, for a moment in time for one model run, the calculation of the strength of association between spacebar and vuln-inc (an increment in vulnerability) and, for comparison, the calculation of the strength of association between spacebar and vuln-zero (resetting vulnerability to zero). It gives the actual frequency of the events and then the decayed values given that 181 s transpired over the experiment and the decay rate is .5.[14] We set W to be 1 and $p = .1$ which produces the "Plus Prior" quantities. The probabilities are calculated by Eqs. (D.2) and (D.3), which result the strengths calculated by Eq. (D.1). The greater associative strength between spacebar and vuln-inc reflects both the much higher probability of spacebar being perceived as the predecessor to vuln-inc than to vuln-zero and the much lower probability of spacebar being perceived in the absence of vuln-inc than vuln-zero. In the model the threshold for attributing a causal relationship is a strength of association of 1.4. Thus, a causal relationship would be attributed between spacebar and vuln-inc but not between spacebar and vuln-zero.

References

Agarwal, A., Hope, R., & Sycara, K. (2018). Challenges of context and time in reinforcement learning: Introducing space fortress as a benchmark. ArXiv preprint arXiv:1809.02206.

[14] This uses the approximate formula for calculated decayed summations which is often used in ACT-R models for efficiency reasons:

$$\sum_{i=1^n} t_k^{-d} \approx \frac{n}{1-d} \times T^{-d} \tag{D.5}$$

where T is the period over which the n events are distributed. This is a reasonable approximation when the events are fairly evenly distributed over the interval of the task.

Aleven, V. A., & Koedinger, K. R. (2002). An effective metacognitive strategy: Learning by doing and explaining with a computer-based cognitive tutor. *Cognitive Science*, *26*(2), 147–179.
Altmann, E. M., & Gray, W. D. (2008). An integrated model of cognitive control in task switching. *Psychological Review*, *115*(3), 602.
Anderson, J. R. (1991). Is human cognition adaptive? *Behavioral and Brain Sciences*, *14*, 471–484.
Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford University Press.
Anderson, J. R., Betts, S., Bothell, D., Hope, R., & Lebiere, C. (2019). Learning rapid and precise skills. *Psychological Review*, *126*(5), 727–760.
Anderson, J. R., Betts, S., Fincham, J. M., Hope, R., & Walsh, M. W. (2020). Reconstructing fine-grained cognition from brain activity. *Neuroimage*, *221*, Article 116999.
Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036.
Anderson, J. R., & Lebiere, C. J. (1998). *The atomic components of thought*. Psychology Press.
Anderson, J. R., & Sheu, C.-F. (1995). Causal inferences as perceptual judgments. *Memory & cognition*, *23*(4), 510–524.
Bower, G., & Trabasso, T. (1963). Concept identification. In *Studies in mathematical psychology*. Stanford University Press.
Cobb, P., Yackel, E., & Wood, T. (1992). A constructivist alternative to the representational view of mind in mathematics education. *Journal for Research in Mathematics Education*, 2–33.
Destefano, M. (2010). *The mechanics of multitasking: The choreography of perception, action, and cognition over 7.05 orders of magnitude*. Rensselaer Polytechnic Institute.
Donchin, E. (1989). The learning strategies project: Introductory remarks. *Acta Psychologica*, *71*(1–3), 1–15.
Dubey, R., Agrawal, P., Pathak, D., Griffiths, T. L., & Efros, A. A. (2018). Investigating human priors for playing video games. ArXiv preprint arXiv:1802.10217.
Garett, R., Chiu, J., Zhang, L., & Young, S. D. (2016). A literature review: website design and user engagement. *Online Journal of Communication and Media Technologies*, *6*(3), 1.
Gianferrara, P. G., Betts, S., & Anderson, J. R. (2020). Time-related effects of speed on motor skill acquisition. In *Proceedings of the 18th international conference on cognitive modelling*.
Grossman, T., Fitzmaurice, G., & Attar, R. (2009). A survey of software learnability: metrics, methodologies and guidelines. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 649–658).
Halasz, F. G., & Moran, T. P. (1983). Mental models and problem solving in using a calculator. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 212–216).
Hamlen, K. R. (2011). Childrens choices and strategies in video games. *Computers in Human Behavior*, *27*(1), 532–539.
Hayes, K. J. (1953). The backward curve: a method for the study of learning. *Psychological Review*, *60*(4), 269.
Ho, S.-B., Yang, X., Quieta, T., Krishnamurthy, G., & Liausvia, F. (2019). On human-like performance artificial intelligence: A demonstration using an atari game. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 9–12).
Holyoak, K. J., & Cheng, P. W. (2011). Causal learning and inference as a rational process: The new synthesis. *Annual Review of Psychology*, *62*, 135–163.
Kieras, D. E., & Bovair, S. (1984). The role of a mental model in learning to operate a device. *Cognitive Science*, *8*(3), 255–273.
Lebiere, C., & Wallach, D. (2000). Sequence learning in the ACT-R cognitive architecture: Empirical analysis of a hybrid model. In *Sequence learning* (pp. 188–212). Springer.
Lee, H. S., & Anderson, J. R. (2013). Student learning: What has instruction got to do with it? *Annual Review of Psychology*, *64*.
Lee, H. S., Betts, S., & Anderson, J. R. (2017). Embellishing problem-solving examples with deep structure information facilitates transfer. *The Journal of Experimental Education*, *85*(2), 309–333.
Marewski, J. N., & Schooler, L. J. (2011). Cognitive niches: an ecological model of strategy selection. *Psychological Review*, *118*(3), 393.
Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning? *American Psychologist*, *59*(1), 14.
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.
Perales, J. C., & Shanks, D. R. (2007). Models of covariation-based causal judgment: A review and synthesis. *Psychonomic Bulletin & Review*, *14*(4), 577–596.
Polson, P. G., & Lewis, C. H. (1990). Theory-based design for easily learned interfaces. *Human–Computer Interaction*, *5*(2–3), 191–220.
Rittle-Johnson, B., Schneider, M., & Star, J. R. (2015). Not a one-way street: Bidirectional relations between procedural and conceptual knowledge of mathematics. *Educational Psychology Review*, *27*(4), 587–597.
Seow, R. Y. T., Betts, S., & Anderson, J. R. (2019). Transfer effects of varied practice and adaptation to changes in complex skill acquisition. In *Proceedings of the 17th international conference on cognitive modelling* (pp. 222–227).
Shanks, D. R. (2007). Associationism and cognition: Human contingency learning at 25. *Quarterly Journal of Experimental Psychology*, *60*(3), 291–309.
Shanks, D. R., Pearson, S. M., & Dickinson, A. (1989). Temporal contiguity and the judgement of causality by human subjects. *The Quarterly Journal of Experimental Psychology*, *41*(2), 139–159.
Shneiderman, B. (2002). Promoting universal usability with multi-layer interface design. *ACM SIGCAPH Computers and the Physically Handicapped*, (73–74), 1–8.
Siegler, R. (2009). Implications of cognitive science research for mathematics education. *Colección Digital Eudoxus*, (8).
Taatgen, N. A., & Anderson, J. R. (2002). Why do children learn to say broke? A model of learning the past tense without feedback. *Cognition*, *86*(2), 123–155.
Taatgen, N. A., Huss, D., Dickison, D., & Anderson, J. R. (2008). The acquisition of robust and flexible cognitive skills. *Journal of Experimental Psychology: General*, *137*(3), 548.
Taatgen, N. A., Van Rijn, H., & Anderson, J. (2007). An integrated theory of prospective time interval estimation: The role of cognition, attention, and learning. *Psychological Review*, *114*(3), 577.
Tessler, M. H., Goodman, N. D., & Frank, M. C. (2017). Avoiding frostbite: It helps to learn from others. *Behavioral and Brain Sciences*, *40*.
Tsividis, P. A., Pouncy, T., Xu, J. L., Tenenbaum, J. B., & Gershman, S. J. (2017). *Human learning in Atari*. Association for the Advancement of Artificial Intelligence.
van Oijen, J., Roessingh, J. J., Poppinga, G., & García, V. (2019). Learning analytics of playing space fortress with reinforcement learning. In *International conference on human-computer interaction* (pp. 363–378). Springer.