Constructivist Simulations for Path Search Algorithms

Alan Blair [a], David Collien [b], Dwayne Ripley [c], and Selena Griffith [a]

[a] UNSW, [b] OpenLearning, [c] University of Sydney

Corresponding Author Email: *blair@cse.unsw.edu.au*

SESSION

C1: Integration of Theory and Practice in the Learning and Teaching Process

CONTEXT We have augmented a large Artificial Intelligence course with a blended learning component, using a social constructivist cloud-based learning platform. This afforded the opportunity to redesign the course delivery by using a flipped tutorial approach where students engage in online activities and interactions which are then consolidated in a face-to-face tutorial. The main focus of this paper is a maze widget which was developed for learning about path search algorithms as part of these online activities.

PURPOSE To create a widget that would offer both constructive and interactive learning experiences in order to help students better conceptualize and understand the relative strengths and weaknesses of nine different path search algorithms.

APPROACH The maze widget allowed students to step through and visualize an algorithm selected from a drop-down menu on randomly generated mazes, as well as creating their own mazes using an online editing tool, and posting them to online galleries where students could view, comment upon, and interact with submissions from other students. Qualitative student feedback was collected to gauge the extent to which this maze widget helped students with learning path search algorithms.

RESULTS Students posted a great variety of mazes to the gallery and wrote descriptions which seem to indicate not only a deeper understanding of the material than was previously possible, but also a better student learning experience.

CONCLUSIONS We found that a carefully designed interactive widget can help students to learn a challenging topic through exploration and interaction, and reach a deeper understanding of the fundamental concepts. A collaborative and iterative course design process informed by educational theory can lead to innovative approaches in engineering education, which make learning experiences more engaging and effective.

KEYWORDS Path Search Algorithms, Constructivist Simulations, ICAP Framework

Introduction

This paper describes our experiences in the redesign of a large Artificial Intelligence course, including migration from face-to-face delivery to blended learning with flipped tutorials. We include background and motivation, some of the innovations we introduced, general observations about student experience, and a detailed description of a maze widget which we developed to help students learn about path search algorithms.

Collaborative Course Design and Development

We recently undertook to redesign a large Artificial Intelligence course at UNSW – motivated by the need to accommodate a significant increase in student enrolments, as well as a desire to enhance the educational experience with online materials, innovative presentation and specially designed widgets. Funding support was obtained through a new University initiative to explore digital uplift through collaboration with external education technology providers. In our case, the technology provider was OpenLearning, a cloud-based social constructivist learning platform with a focus on activity-centered learning in a student-centered online collaborative learning environment.

An academic who had been teaching the course for a number of years in a more traditional face-to-face format acted as subject matter expert (SME), and collaborated on all aspects of the course redesign and delivery with an instructional designer (ID) from OpenLearning who was pursuing postgraduate studies in the learning sciences. The maze widget was created by a software engineer (SE) from OpenLearning with feedback and guidance from the SME. The SE, who was a student in the SME's course some years earlier, had course design experience, a theoretical background in education, and expertise in artificial intelligence.

The team collaborated remotely to contribute content, design, coding and development of the course, using tools such as Slack, Google Drive and the online learning platform itself, which were chosen for being both easy to learn and effective at supporting collaboration. The SME and ID met weekly over several months to plan, co-develop, and finalize course design and content, while the SE advised on technical issues and worked intensively on the maze widget.

Blended Learning with Flipped Tutorials

In total, 260 undergraduate and 300 postgraduate students were enrolled in the course. There was great diversity among these students in regard to prior study, coding experience, relevant general knowledge and socio-cultural background. In order to meet the needs of such a large and diverse group of students, and keep them engaged and motivated, we designed a blended learning model which expanded upon the existing slide decks and lecture recordings with short tablet-style walkthrough videos, links to extension material, and online activities that formed a flipped tutorial model.

Lectures were scheduled on Thursday evenings, and the material from each lecture was consolidated in smaller "flipped" tutorial groups (problem sections) of 25 students, which were all scheduled on Thursday or Friday of the following week. Course content was released in advance of the lecture in the form of text, images and short explanatory videos. Students were expected to work through a number of online activities in the collaborative learning platform during the week between each lecture and the corresponding tutorial.

The general idea of the flipped classroom is to use technology to introduce content in advance, in order to free up class time for an expanded range of learning activities (Roehl et al., 2013). In consideration of the large class size, we developed a *flipped tutorial* approach in which students are introduced to the content during the lectures, develop their understanding through online activities and interactions during the week, and consolidate their understanding of the material through smaller group discussions and reflection in the face-to-face tutorials at the end of the week. In addition to the tutors (teaching assistants), an online facilitator (who was a former student in the course) was recruited to create social presence and support/guide learners as they progressed through the course.

Path Search Algorithms

One theme that is fundamental in many areas of engineering education is the interplay between algorithms and problem instances. A variety of competing algorithms or techniques are presented for solving a certain class of problems; but, some of these algorithms may be better suited than others to a particular situation, depending on certain attributes of the problem instance. In this paper we particularly focus on path search algorithms. The students are first introduced to these algorithms in Weeks 3 and 4 of our course, and the algorithms are demonstrated by tracing through examples from the textbook (Russell & Norvig, 2016) which involve finding a path between specified cities in a map of Romania. Students are then expected to apply the same algorithms to more complex domains such as solving a 4-by-4 sliding tile puzzle. In previous offerings of the course, a typical tutorial question might be: "Describe a search space in which breadth-first-search performs much worse than depth-first-search". However, we found that students were generally unable to come up with their own answers to this type of question; and, even when told an answer by the tutor, they still had difficulty conceptualizing the relative strengths and weaknesses of the various algorithms.

Traditional textbook instruction is limited as it can contain knowledge that is decontextualized and explicit. Beyond simply delivering an exposition of each algorithm, we felt that students would gain a deeper understanding if they were able to create their own problem instances, apply the various algorithms, and explore for themselves why certain kinds of test cases would cause one algorithm to perform particularly well, while another may consume excess time or memory, or produce a poor quality solution. Simulations were chosen as a tool because they allow knowledge to be experienced contextually, so implicit domain knowledge is learned, which "will foster deeper understanding and make the information more accessible in appropriate problem-solving contexts" (Taylor & Chi, 2006).

We therefore designed a purpose-built maze widget to support the online activities for Weeks 3 and 4 of the course – which covered Uninformed and Heuristic Path Search Algorithms, respectively. The aim of the widget is to facilitate conceptualization of the algorithms and exploration of the interplay between algorithms and problem instances, thus bridging the gap between simple examples that can be worked through by hand, and complex domains which are too large or multi-dimensional to be directly visualized. This approach builds upon the observation that "the move from a static model in an inert medium, like a drawing, to dynamic models in interactive media that provide visualization and analytic tools is profoundly changing the nature of inquiry in mathematics and science" (Brown et al., 1999).

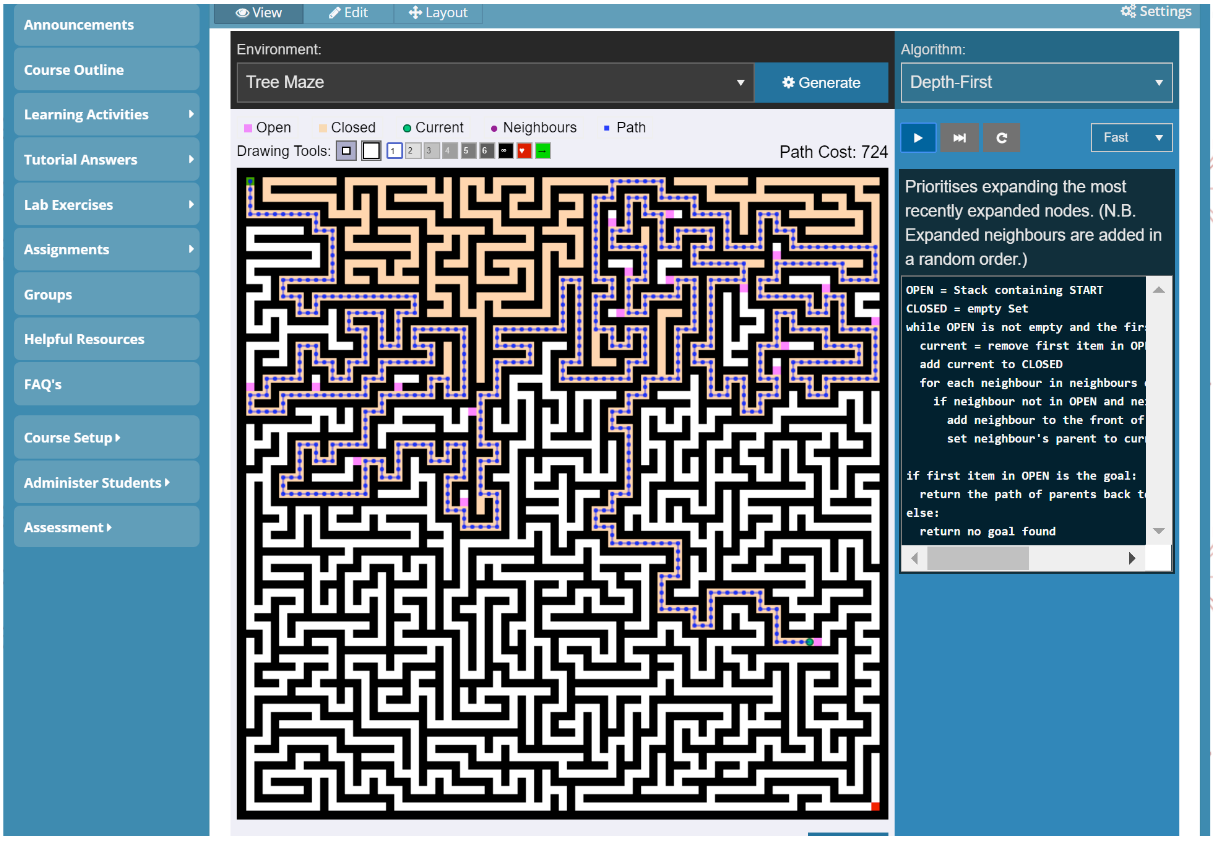
Maze Widget

The maze widget allows students to generate a grid maze and then select from nine different path search algorithms, of which five are uninformed (depth-first, breadth-first, iterative deepening, cost-directed, and bi-directional cost-directed searches) and four are heuristic or "informed" search algorithms (Greedy, A-star, iterative-deepening A-star, and bi-directional

A-star). For the heuristic search algorithms, students can additionally choose between two different heuristics (Euclidean distance and Manhattan distance). The widget was implemented in a flexible way which allowed the algorithms to be introduced over two separate activities within the course – the first to introduce the uninformed searches, and the second to build upon those concepts with the heuristic searches.

The widget code was written for the Web in ES6, and included implementations of each of the algorithms. The implementation allows students to run the algorithm at varying speeds, or step through one line at a time, so the algorithm can be visualized for immediate reflection. Pseudocode for the selected algorithm is displayed alongside the simulation, thus allowing the student to easily follow the logic of the algorithm as each step is executed and displayed by the simulator.

The widget design went through three main iterations. The first conceptualized its use as a visualization tool. Students would first be asked to select a certain style of maze (tree maze, graph maze, concentric tree maze, concentric graph maze, alternating squares or empty) from a drop-down menu, and then press a button to generate a random maze in the specified style (Figure 1). They could then select an algorithm from another drop-down menu and step through the algorithm, or let it play out automatically – with a speed control. This would provide students with immediate reflection on how the algorithms execute within the grid-maze, and an opportunity to compare and contrast the different approaches.



Environment: Tree Maze

Algorithm: Depth-First

Path Cost: 724

Prioritises expanding the most recently expanded nodes. (N.B. Expanded neighbours are added in a random order.)

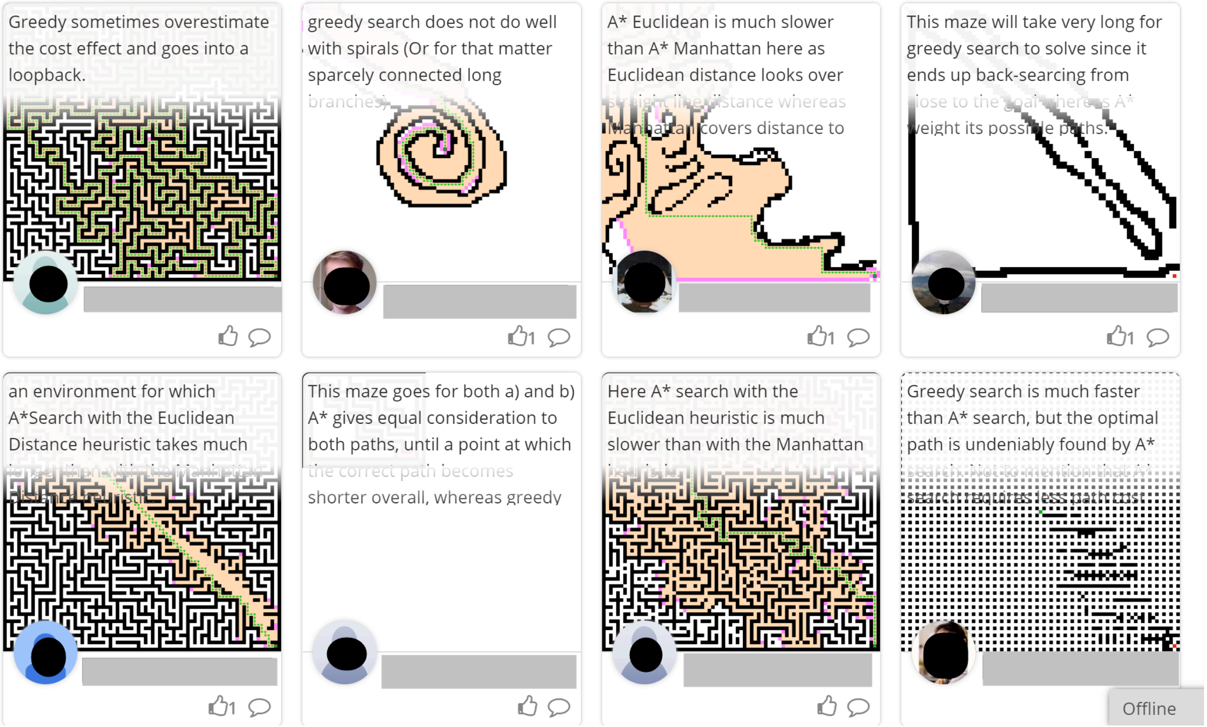
```
OPEN = Stack containing START
CLOSED = empty Set
while OPEN is not empty and the fir
  current = remove first item in OP
  add current to CLOSED
  for each neighbour in neighbours
    if neighbour not in OPEN and ne
      add neighbour to the front of
      set neighbour's parent to cur

if first item in OPEN is the goal:
  return the path of parents back t
else:
  return no goal found
```

Figure 1: Maze widget simulating a depth-first search on a randomly generated maze

A second iteration of the design enabled active and constructive experimentation. The widget was augmented with maze editing functionality, using an interface which is similar to a simple 'paint' program. By selecting colours from a palette, the student could move the start location (green), add one or more goal locations (red) or "paint" the maze with obstacles (black) or empty spaces (white). This allowed for more open-ended exploration of path search algorithms which afforded a more constructive learning experience. Students could actively explore and experiment, and by reflectively analyzing the resulting behaviour of the simulation, build analogies/schemas/concepts for how each algorithm behaved in customized test cases, and how to construct situations which illustrated the strengths and weaknesses of each search algorithm.

The purpose of the third iteration was to promote and enable a community of sharing. The widget was further augmented to allow the sharing of student-created mazes (Figure 2). This also included the ability to tie into the platform's social sharing tools to share and showcase different scenarios, enabling students to experiment with and further modify each other's creations in simulation, with the opportunity for community feedback and commentary.



Greedy sometimes overestimate the cost effect and goes into a loopback.

[redacted] [redacted]

greedy search does not do well with spirals (Or for that matter sparcely connected long branches)

[redacted] [redacted]

A* Euclidean is much slower than A* Manhattan here as Euclidean distance looks over straight line distance whereas Manhattan covers distance to

[redacted] [redacted]

This maze will take very long for greedy search to solve since it ends up back-searcing from close to the goal whereas A* weight its possible paths.

[redacted] [redacted]

an environment for which A*Search with the Euclidean Distance heuristic takes much longer than with the Manhattan Distance heuristic.

[redacted] [redacted]

This maze goes for both a) and b) A* gives equal consideration to both paths, until a point at which the correct path becomes shorter overall, whereas greedy

[redacted] [redacted]

Here A* search with the Euclidean heuristic is much slower than with the Manhattan heuristic.

[redacted] [redacted]

Greedy search is much faster than A* search, but the optimal path is undeniably found by A* search. Not to mention that A* search requires less path cost

[redacted] [redacted]

Figure 2: Gallery view of student-created mazes and interactions

Activities and Discussion Points

Students were encouraged to work through online activities and discuss their findings both on-line and in the face-to-face tutorials at the end of the week. Discussion points for the flipped tutorial on Uninformed Path Search Algorithms (Week 3) included:

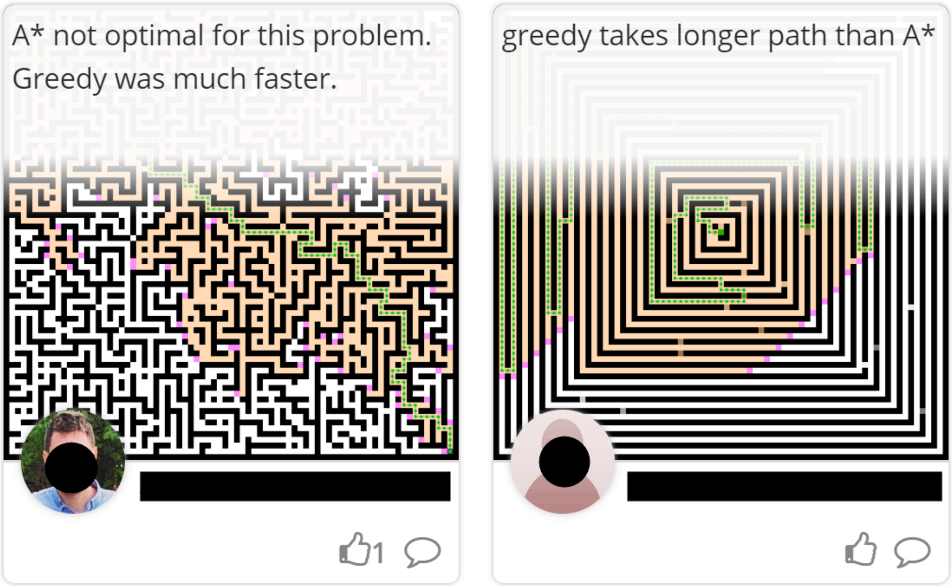
- Compare the speed of Depth First Search (DFS) and Breadth First Search (BFS) on random tree mazes and concentric graph mazes
- Create a maze for which BFS finds a solution considerably faster than DFS
- Create a maze for which DFS finds a solution considerably faster than BFS
- Create a maze for which bi-directional search is faster than BFS
- Run Iterative Deepening Search (IDS) on a random maze and explain why it is slow
- For what type of problem (not a maze) would IDS be faster than BSF and DFS?

Discussion points for the tutorial on Heuristic Path Search Algorithms (Week 4) included:

- Create a maze for which Greedy Search takes much longer than A*Search
- Create a maze for which Greedy Search produces a path that is much longer than the optimal path
- Create a maze for which A*Search with the Euclidean distance heuristic takes much longer than with the Manhattan distance heuristic
- Create a maze that is interesting for some other reason

Pedagogical Approach

The OpenLearning platform provided the design team the opportunity to augment the traditional lecture approach, and to provide opportunities for more engaging and collaborative learning experiences. The design team used the ICAP framework (Chi & Wylie, 2014) to provide a guideline for the activities we adapted for the flipped tutorial model. The ICAP hypothesis posits that "engagement behaviors can be categorized and differentiated into one of four modes: Interactive, Constructive, Active, and Passive. The ICAP hypothesis predicts that as students become more engaged with the learning materials, from passive to active to constructive to interactive, their learning will increase" (Wiggins et al., 2017). This has been proven to be effective in engineering education (Menekse et al., 2013), and was central to the development of our maze widget, which in addition to helping students visualize and contextualise path search algorithms, provided opportunities for constructive and interactive learning experiences.



A* not optimal for this problem. Greedy was much faster.

[redacted]

greedy takes longer path than A*

[redacted]

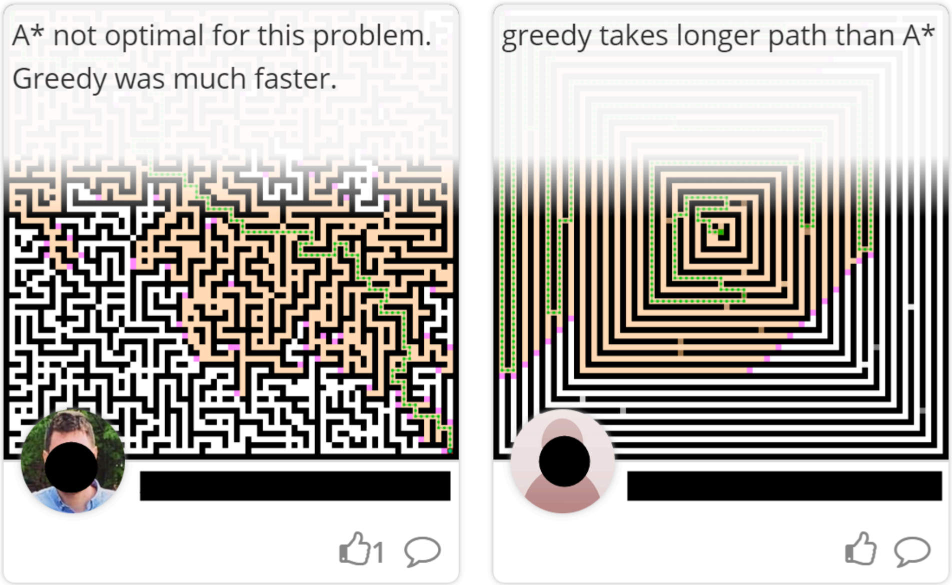
Figure 3: Students sharing mazes which illustrate the relative strengths and weaknesses of Greedy Search and A* Search

The functionality of the maze widget allows students to explore the same algorithm on different kinds of problem instances (Figure 3) allowing for analogical encoding by drawing a comparison across examples (Gertner et al., 2003) thus leading to a deeper understanding of the algorithms.

Student Feedback

Even though it was only a formative task for group discussion and self-evaluation, the maze widget activities proved to be very popular with the students, with many students making multiple contributions and insightful comments.

Our findings from observing the student contributions on the platform, coupled with the qualitative student feedback supports the finding in (Menekse et al., 2013) that “the ICAP hypothesis provides a comprehensive methodology to create and design materials and activities that will promote effective learning in engineering classrooms”.



A* not optimal for this problem. Greedy was much faster.

[redacted] [redacted]

greedy takes longer path than A*

[redacted] [redacted]

Figure 3: Students sharing mazes which illustrate the relative strengths and weaknesses of Greedy Search and A* Search

Qualitative feedback was incorporated into a reflection page at the end of each module. Following the first use of the Maze widget in week 3, Uninformed Path Search, students were given an opportunity to provide any insights about the blended module. The results show that students benefited from the visualization of the maze widget, and the widget helped students gain a deeper conceptual and practical understanding of the algorithms.

Across all comments on the platform, there was a lot of feedback on the maze widgets, and all of it was positive. Voluntary feedback from students included:

> “The maze search widget was helpful in learning about search strategies, I really enjoyed seeing how the algorithms work.”
>
> “This week's activities gave me a more clear idea about how Breadth First Search and Depth First Search perform in different types of mazes. It was something I had a brief idea about but never visualised it.”
>
> “I enjoyed to maze widget – it’s a really great representation of all the different algorithms and really helped to solidify my understanding!”
>
> “the maze apps makes learning it more interactive less theoretical”
>
> “I thought the maze app was a brilliant piece of kit, well done guys.”

A deeper understanding of specific path search strategies gained by students interacting with the maze widget can be seen in the following student comments:

> "This week I enjoyed further studying search algorithms, and learnt quite a lot about depth-first search. Primarily derived from it drunkenly wandering about the maze field."
>
> "I did not enjoy having to watch the maze-solving widget run using different search algorithms – sometimes it was frustrating seeing how close the algorithm was to the solution but how long it was taking to actually find it (of course this ultimately provides helpful insight into each algorithm)!!"

Feedback from another student illustrates how the maze widget helped to further develop a practical understanding of a search algorithm:

> "The Romania problem has helped me to understand different uninformed search strategies. The maze search widget gave me a better insight of how uninformed searches are linked to the real world."

Another student provided feedback that identifies the strengths of using a simulation to visualize algorithms over traditional teaching approaches:

> "I felt that I learnt a lot in regards to search patterns – really wished I did this course with COMP1927 – was so lost during it –> this is by far amazing! :D"

Overall, we found that the OpenLearning platform, and specifically the maze widget simulation are very well-suited for engineering education. The social constructivist platform created opportunities for students to learn collaboratively, and the qualitative feedback from the students indicates that the maze widget helped students to visualize, conceptualize, and understand how to apply the algorithms.

Next Steps

An iterative approach has been taken to the maze widget design, and to the overall course design as well. Quantitative data from the course analytics has been analyzed, and qualitative feedback from students has been reviewed. Reflections on the design included some technical adaptations, as well as providing more student control, better use of the online tutorial group function, and the inclusion of some more tablet-style walkthrough videos for difficult technical components of the course. There are plans to make the maze widget available outside of the course, and the design team may experiment with a productive failure approach (Kapur, 2012) outside of the course, and potentially adapt sequencing based on what is learned.

The success of the blended learning and flipped tutorial approach has led us to consider modifying the course so it can stand alone as an online offering. It should be noted that this would still require an online presence for the lecturer and chosen course facilitators. We also plan to gather more qualitative and quantitative data around the final assignment, designed by the SE together with feedback and guidance from the SME, which required students to apply their knowledge of path search algorithms gained from the maze widget to solve a game-based 'Adventure Land' scenario.

Conclusion

Our experiences supported the use of the ICAP framework, which "can provide specific guidelines for how to create lessons that incorporate overt behaviors that are associated with higher levels of engagement and their associated knowledge-change processes" (Chi & Wylie, 2014) and we suggest that it is a useful tool for appraising course design and delivery.

Simulations like the maze widget could potentially be applied in any area involving the interplay between algorithms and problem instances. Potential applications include: sorting, storage & retrieval, graph and tree algorithms, string & text processing, machine learning, constraint satisfaction, game theory, data compression, signal and image processing, circuit design, network routing and switching, and cryptography.

Our collaborative and innovative design process highlighted integration in engineering on multiple levels – academic and industry, teacher and (former) student, theory and practice – thus illustrating how an engineering approach to education can make learning experiences more engaging and effective.

References:

Brown, A. L., Bransford, J., Cocking, R. R., & National Research Council (U.S.). Committee on Developments in the Science of Learning. (1999). *How people learn: Brain, mind, experience, and school*. Washington, D.C: National Academy Press.

Chi, M. T. H., & Wylie, R. (2014). The ICAP framework: Linking cognitive engagement to active learning outcomes. *Educational Psychologist, 49*(4), 219-243.

Gentner, D., Loewenstein, J., & Thompson, L. (2003). Learning and transfer: A general role for analogical encoding. *Journal of Educational Psychology, 95*(2), 393-408.

Kapur, M., & Bielaczyc, K. (2012). Designing for productive failure. *Journal of the Learning Sciences, 21*(1), 45-83.

Menekse, M., Stump, G. S., Krause, S., & Chi, M. T. H. (2013). Differentiated overt learning activities for effective instruction in engineering classrooms: Differentiated overt learning activities. *Journal of Engineering Education, 102*(3), 346-374.

Roehl, A., Reddy, S. L., & Shannon, G. J. (2013). The flipped classroom: An opportunity to engage millennial students through active learning strategies. *Journal of Family and Consumer Sciences, 105*(2), 44.

Russell, S.J. & Norvig, P. (2016). *Artificial intelligence: A modern approach* (3rd Global; ed.). Harlow, Essex, England: Pearson Education Limited.

Wiggins, B. L., Eddy, S. L., Grunspan, D. Z., & Crowe, A. J. (2017). The ICAP active learning framework predicts the learning gains observed in intensely active classroom experiences. *AERA Open, 3*(2).