Construction Spaces for the Design of Learning Tasks in Virtual Worlds

Dennis Maciuszek, Alke Martens
University of Rostock, Germany
dennis.maciuszek@storyautor.de
alke.martens@uni-rostock.de

Abstract: How should we design and select tasks in educational virtual worlds so that students are encouraged to experiment, without getting lost in the vast possibilities of a constructivist environment? We present a scenario in which students learn about artificial intelligence by programming the behaviour of game characters in *Second Life*. In two pre-studies, two factors seemed to support learning: appropriate task difficulty and appropriate degree of instruction versus construction at the right time. Intelligent Tutoring Systems can choose appropriately difficult tasks automatically, based on models. Building on this, we propose a two-dimensional model for open-ended microworlds that additionally considers the degree of instructional guidance versus self-regulated construction. After the completion of a task, the teacher assigns either a more complex task, or the same task, but with more interactive freedom. In our main study, school students created autonomous virtual robots by connecting modules, reading and writing code. Using an exercise pool in accordance with our model, we recorded chosen learning paths and perceived difficulty to see whether our adaptation worked. To assess what was learnt, we analysed constructed mental models. The paper presents and discusses the results.

Keywords: virtual world, constructivism, mental model, instructional design, cognitive apprenticeship

1. Introduction

Today, virtual worlds like *Second Life* or 3D engines like *Unity* are easily accessible. They come with powerful physics engines and scripting languages for realising highly interactive simulations. These enable instructional designers to create learning environments based on constructivist teaching ideals. Constructivist theories emphasise the individual learner's role in transforming information to knowledge. Learners construct their own version of the truth, or find their own paths towards it. Constructivist teaching seeks to create authentic, complex situations in which learners' experiences inspire own explanations. Knowledge is born by anchoring new information in personal experience. Constructivist educational settings should therefore provide students with freedom to express themselves and establish their own personal constructs of a domain (for a more detailed introduction to constructivism, see Martens and Maciuszek 2013). In science and engineering domains, constructivist teaching is often called inquiry learning or discovery learning (e.g. de Jong and van Joolingen 1998).

A virtual world can immerse students in interactive simulations of domain activities and create the illusion of 'being there'. Different from real-life settings, making mistakes is possible – even desirable – without causing accidents. Through trial and error, students can form and validate hypotheses, construct mental models, and ideally gain a deep understanding of the topic. If done well, the activity enables them to transfer constructed knowledge to similar real-world situations.

However, as de Jong and van Joolingen (1998) point out by discussing various studies, discovery learning with interactive simulations is difficult. It will not succeed in a truly free environment without any instructional guidance. This problem was formulated already before 3D virtual worlds became attractive for educational game designers, but it remains until today. First of all, a simulated environment in itself is not enough. The designer needs to fill it with appropriate activities and tasks for attaining the learning goals. Second, students are required to employ advanced self-regulation skills to master potentially complex scenarios while not getting distracted or even lost in the many possibilities of an open-ended immersive environment.

This is a core problem for any design of interactive media: how much control should the designer apply, how much freedom should the user have? How much instruction versus how much construction? We have previously presented a literature review on this dilemma (Martens and Maciuszek 2013). For instance, researchers have been studying virtual "microworlds" – interactive simulations driven by a set of internal laws – since before the time of advanced graphical interfaces (starting with Papert 1980). Rieber (1992) suggested helping the learner by reducing the complexity of simulations to the variables needed and by structuring learning in the form of missions. In game design, the instruction-construction dualism boils down to the choice of genre. Adventure games build on scripted narratives (instruction), while simulation games hope that meaningful experiences will emerge from the interaction (construction; e.g. Sweetser 2005). Role-playing games (RPGs) are a compromise. They embed scripted quests (missions) in worlds of simulated activities. A quest is basically a mini-adventure game. Adventures use only a few RPG-like activities – *crafting* (using items on items or the environment) and single-choice *conversations* – as puzzles to advance a scripted plot. Section 2 introduces a crafting scenario as it could appear in an educational RPG. Section 3 develops a model for addressing instruction versus construction in such scenarios. Section 4 reports on an evaluation of the approach in the classroom. Section 5 concludes.

2. Study case

Five years ago, we wanted to try something new in our Applied Artificial Intelligence courses for Computer Science students at the University of Rostock. To give the students a hands-on experience of artificial intelligence (AI), we created a virtual factory environment in *Second Life* (SL). In addition to the lectures, the students carried out exercises in which they programmed the behaviour of game characters in the Linden Scripting Language (LSL). The first half of the course consisted of prepared tasks in which our ten students (in teams of two) were asked to implement a range of AI algorithms. The second half was less instructional. Each team contributed a level to an action adventure game (within SL) that employed an AI strategy chosen and designed by the students. It had to realise the behaviour of an opponent in the game. During the course, we found that LSL and the physics engine were so difficult to handle that they got in the way of completing the prepared AI tasks. In the individual projects, however, the students obtained remarkable results. One group implemented a surveillance system in their factory hall that had a robot swarm pursue the player's avatar by graph-based planning. Students implemented advanced AI techniques, yet they could not always name them. A constructivist approach (second half) worked in this environment, but only after the students had acquired some experience. For a detailed presentation of this pre-study see Maciuszek et al. (2013).

Inspired by those first experiences in a University setting, we adapted the AI course so that it fits into a secondary-school Computer Science curriculum. We narrowed down the domain to basic autonomous steering (Reynolds 1999) like pursuit of a user's avatar or fleeing from it. In a pre-study with two participants (both male, aged 15 and 17) in a collaborative distance-learning arrangement, we started out with very basic discovery learning tasks. Students observed artificially intelligent game characters and had to explain their behaviour. Following a Cognitive Apprenticeship approach (Collins, Brown and Newman 1987), the teacher (first author) slowly faded the degree of instructional guidance and encouraged a higher degree of experimentation, e.g. interpreting code. We had redesigned the virtual factory from the first pre-study to obtain a more precisely defined microworld that better resembled a crafting environment in an RPG or adventure game. It was now a robot factory that supplied a pool of different robot eyes (sensors), heads (algorithms), and feet (actuators). The parts communicate with each other through programming interfaces to realise robot behaviour. A typical task would be to combine three parts into a robot that perceives an avatar within a certain angle and range, then follows it with a certain speed. Additional guidance existed in the form of assembly instructions on virtual posters on the factory walls. These were actually sketches of vector algebra and formulas for the steering algorithms: Figure 1.

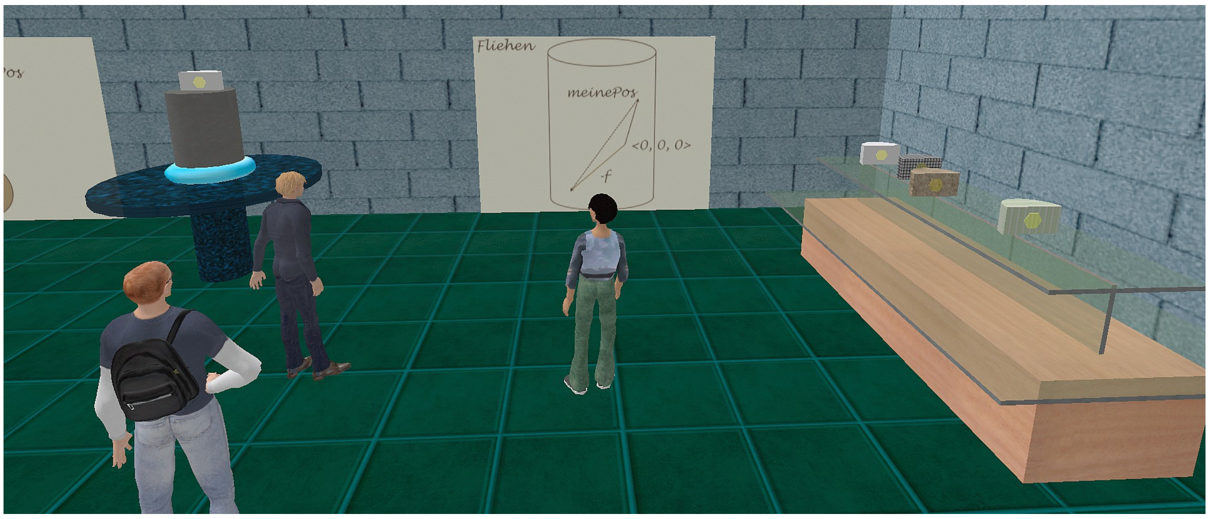


Figure 1: Virtual factory with connectable robot parts (right: eyes, left: robot) and instructional posters

To provide a reflection tool, we asked the students to keep an online learning journal. They wrote down interpretations, assembly instructions in their own words, and ideas that came up after each of the four 90-minute sessions. To us, the produced texts provide insight into the students' constructed mental models. While the older student wrote exact and comprehensive descriptions of the algorithms and in the last session came up with ideas about simulating cellular networks and swarm intelligence, the younger student was thinking more practically about what could be done in the programming environment. Already after the second session, the older student displayed critical thinking when he expressed, mathematically, why a fleeing robot should not be allowed to turn around: because it cannot see the pursuing avatar anymore. This issue was not planned to be discussed, but we adapted to the student's interest and in the following session worked together on a solution for this. The younger student surprised us with an invention he had made after the third session (a transfer achievement). After having learnt about forces in the physics engine, he had designed and programmed a rocket that counters gravity: Figure 2.

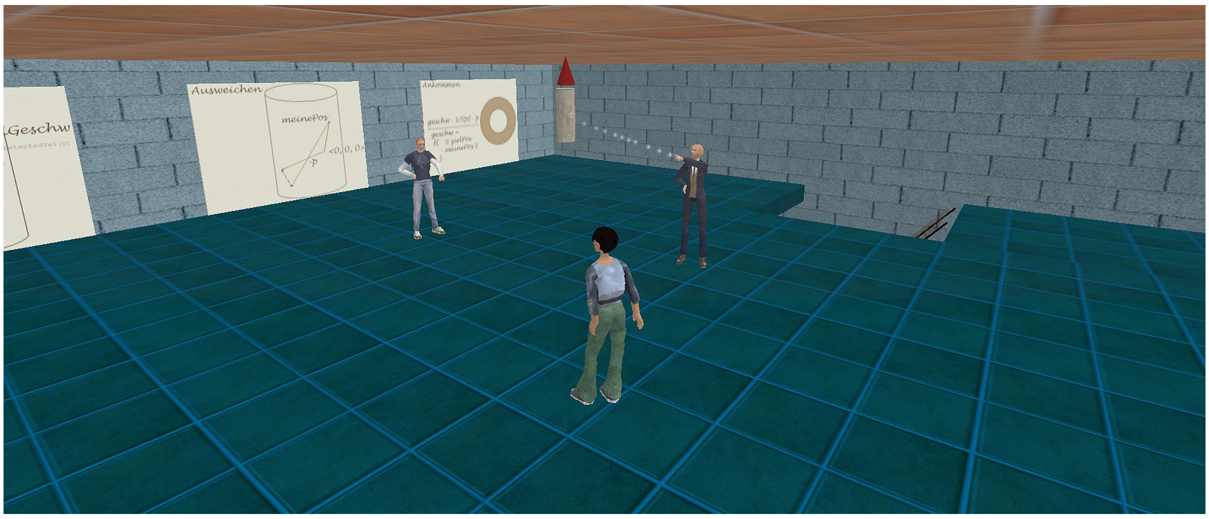


Figure 2: Invention presented in the final session: a rocket lifting off

All in all, the students mastered the comparatively easy tasks quicker than our instructions were supposed to fade. They became creative pretty early. We encouraged this, although it sometimes meant digressing from the learning goals. Programming a rocket is a physics, not an AI problem.
Looking back at both, the University course and the distance-learning pre-study with the two school students, we could observe that a virtual world environment is capable of inspiring creativity. Two factors seemed to support learning: the appropriate difficulty of a task and the appropriate degree of instruction versus construction at the right time for an individual or team. It is up to the teacher (or an intelligent mechanism in the software) to regulate this. If such adaptation shall not just happen intuitively, then we need a model for structuring learning tasks in virtual worlds. Finding such a model for the case of crafting in virtual factories is the objective of the following section. The robot factory remains the concrete study case.

3. Design approach

Intelligent Tutoring Systems (ITS) can look back on a comparably long tradition (see e.g. Lelouche 1999). Starting from the observation that instruction works best if content is adapted to the learner's individual needs, different approaches have been developed over the years, following different trends in instructional design (Martens 2004). Traditionally, "individual needs" means the learner's knowledge state in relation to a domain model. Like a good teacher, an ITS can determine the difficulty of a candidate task for a certain student or help compensate individual weaknesses.
For structuring our robot factory tasks, we start out by applying an ITS modelling approach from Knowledge Space Theory that Albert and Held (1999) call "systematical problem construction". We wish to teach a set of five steering techniques through five tasks:

$T = \{Seek, Flee, Pursuit, Arrive, Evade\}$

The teacher does not simply explain the algorithms, but hands out crafting tasks in the virtual world that lead to well-defined solutions (with space for variations):

- *Seek* computes an impulse vector (in each time step) towards a target position in 3D space.
- *Flee* computes a *Seek* vector and reverses its orientation.
- *Pursuit* computes a *Seek* vector to an anticipated future position of the target.
- *Arrive* computes a *Seek* vector and decelerates as it comes closer.
- *Evade* computes a *Flee* vector from an anticipated future position (reversed *Pursuit*).

For each element in *T*, the teacher has recorded whether a student does or does not know the corresponding algorithm. With the above composition of the tasks, it is possible to make assumptions about prerequisites and learning paths: if the student does not know *a*, he/she will surely not know *b*, as it includes *a*. In Knowledge Space Theory, this is the surmise relation:

(1) $\neg \mathit{Seek} \Rightarrow \neg \mathit{Flee} \wedge \neg \mathit{Pursuit} \wedge \neg \mathit{Arrive}$
(2) $\neg \mathit{Flee} \Rightarrow \neg \mathit{Evade}$
(3) $\neg \mathit{Pursuit} \Rightarrow \neg \mathit{Evade}$

Any valid set of mastered tasks is a possible knowledge state, i.e. what one student/team can know. Making use of the above relation, which is reflexive and transitive, one can determine the knowledge space, i.e. the set of all possible knowledge states:

K = {{}, {*Seek*}, {*Seek*, *Flee*}, {*Seek*, *Pursuit*}, {*Seek*, *Arrive*}, {*Seek*, *Flee*, *Pursuit*}, {*Seek*, *Flee*, *Arrive*}, {*Seek*, *Pursuit*, *Arrive*}, {*Seek*, *Flee*, *Pursuit*, *Arrive*}, {*Seek*, *Flee*, *Pursuit*, *Evade*}, {*Seek*, *Flee*, *Pursuit*, *Arrive*, *Evade*}}

A teacher or ITS can now choose tasks for a student that would extend his/her knowledge state by one item, e.g. *Flee* if they only have completed *Seek* so far.
As a potential improvement for designing tasks in constructivism-inspired, open-ended microworlds, we propose a two-dimensional model that in addition to difficulty/complexity considers an appropriate degree of instructional guidance versus self-regulated construction. After the completion of a task, a teacher would assign either a task testing more complex knowledge, or a task of similar complexity, but granting more explorative freedom. This generalises tasks to task classes (cf. Enfield 2012).
The idea behind our generalisation along an instruction-construction scale is based on the Cognitive Apprenticeship approach (Collins, Brown and Newman 1987). It regards the student as an apprentice whose master (the teacher) first demonstrates and coaches, doing parts of tasks him-/herself where the student is still weak (scaffolding), then bit by bit withdraws as knowledge fortifies (fading). The experienced student is ready for free experimentation and can explicitly communicate gained insights. Following this, we decided to design each AI task in three versions: one for scaffolding (instruction), one for fading (construction), and one intermediate. In Schulmeister's (2003) taxonomy of task interactivity, our tasks shift between levels 4 ("manipulate") and 5 ("construct"). Manipulation would be crafting as in an RPG or adventure game: connecting pre-programmed robot parts. Construction would mean writing new bits of program code (within SL). At the intermediate level, students just read and interpret code. The new task set reads:

$T' = \{\mathit{Seek}_1, \mathit{Seek}_2, \mathit{Seek}_3, \mathit{Flee}_1, \mathit{Flee}_2, \mathit{Flee}_3, \mathit{Pursuit}_1, \mathit{Pursuit}_2, \mathit{Pursuit}_3, \mathit{Arrive}_1, \mathit{Arrive}_2, \mathit{Arrive}_3, \mathit{Evade}_1, \mathit{Evade}_2, \mathit{Evade}_3\}$

The new surmise relation is (any *i* = 1, 2, 3):

(1') $\neg \mathit{Seek}_i \Rightarrow \neg \mathit{Flee}_1 \wedge \neg \mathit{Flee}_2 \wedge \neg \mathit{Flee}_3$
$\wedge \neg \mathit{Pursuit}_1 \wedge \neg \mathit{Pursuit}_2 \wedge \neg \mathit{Pursuit}_3$
$\wedge \neg \mathit{Arrive}_1 \wedge \neg \mathit{Arrive}_2 \wedge \neg \mathit{Arrive}_3$
(2') $\neg \mathit{Flee}_i \Rightarrow \neg \mathit{Evade}_1 \wedge \neg \mathit{Evade}_2 \wedge \neg \mathit{Evade}_3$
(3') $\neg \mathit{Pursuit}_i \Rightarrow \neg \mathit{Evade}_1 \wedge \neg \mathit{Evade}_2 \wedge \neg \mathit{Evade}_3$
(4) $\neg \mathit{Seek}_1 \Rightarrow \neg \mathit{Seek}_2$
(5) $\neg \mathit{Seek}_2 \Rightarrow \neg \mathit{Seek}_3$
(6) $\neg \mathit{Flee}_1 \Rightarrow \neg \mathit{Flee}_2$
(7) $\neg \mathit{Flee}_2 \Rightarrow \neg \mathit{Flee}_3$
(8) $\neg \mathit{Pursuit}_1 \Rightarrow \neg \mathit{Pursuit}_2$
(9) $\neg \mathit{Pursuit}_2 \Rightarrow \neg \mathit{Pursuit}_3$

(10) $\neg Arrive_1 \Rightarrow \neg Arrive_2$
(11) $\neg Arrive_2 \Rightarrow \neg Arrive_3$
(12) $\neg Evade_1 \Rightarrow \neg Evade_2$
(13) $\neg Evade_2 \Rightarrow \neg Evade_3$

This structure enables the teacher to pave individual learning paths, forwards and backwards. He or she would guide a student through a *construction space*. The students can discover the constructivist environment at their own pace. If and how this works in practice was evaluated in a classroom study.

4. Evaluation

We tried out the task structure over four 90-minute sessions in a regular 12th grade Computer Science course at a German secondary school. Twelve students aged 17–18 (11 male, 1 female) participated. For two weeks, the first author became their teacher. After a brief lecture on game AI, physics (impulse, force), vector algebra, SL and basic ideas of scripting, each two-person team started with $Seek_1$. Subsequent tasks were chosen by the teacher on the basis of the construction space and an observation of the team's performance: what would be a suitable next topic, and/or are they ready for a higher degree of interactive freedom? The reason for forming teams was that the school lab had exactly six PCs that met the requirements for SL. Each team received one avatar and one room in the virtual building. Network speed or the SL server was lagging sometimes, which caused some problems. A different issue was the distractive nature of the playful environment. Besides selecting tasks and helping, the teacher often had to remind students to attend to their tasks and worksheets instead of exploring SL (and sometimes damaging virtual objects).

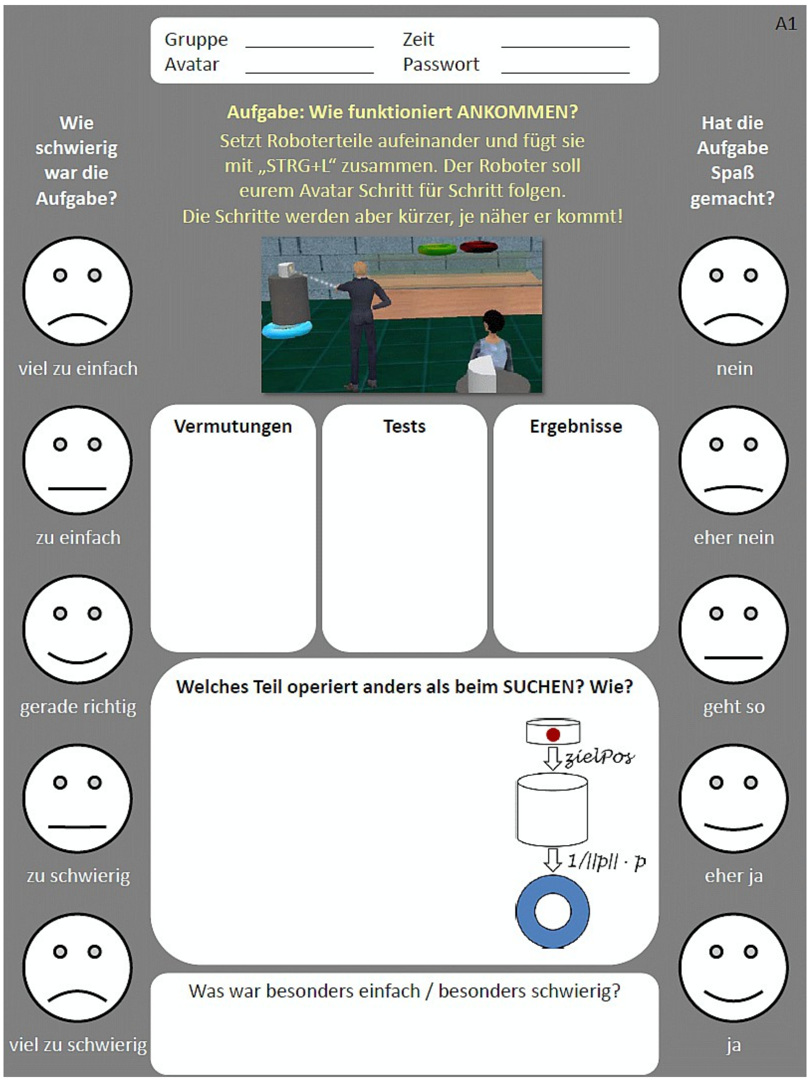
4.1 Hypotheses

The main objective of the study was to evaluate our approach to balancing instruction versus construction in virtual factories for Computer Science education: paving individual learning paths through construction spaces. We hoped to find evidence for the following:

- By traversing construction spaces, students are always working on tasks that pose a medium challenge.
- Working on such tasks is enjoyable.
- Students engage in discovery learning and form correct mental models.
- Later, when working on tasks allowing more freedom, students have creative ideas.

4.2 Method

Gaining access to students' constructed mental models requires a qualitative data collection strategy. Thinking-aloud protocols are typical in cognitive psychology research. But those work in a laboratory setting, not in a classroom with lots of talking, collaborating people. Observation of the collaborative dialogues would have been a possibility. But this would only collect communication in the instant of problem solving, not the reflection part. In addition, any surveillance might be experienced as intrusive. On the other hand, post-activity interviews or questionnaires would come in too late. Students would not recall their thoughts.
A tool from ethnographic research captures thoughts as they happen, without intruding in a person's natural activities: cultural probes (e.g. Crabtree 2003). A cultural probe can be a diary, a photo camera, or similar artefacts that a participant uses to document his/her daily life. We thought that a natural tool in a school student's daily routine would be paper worksheets being filled in while working on tasks. So, we created a pool of worksheets: one for assessing previous knowledge, a final one for assessing the state after the course (including space for transfer, ideas, opinions), and one for each task during the main phase: Figure 3.



A1

Gruppe __________ Zeit __________
Avatar __________ Passwort __________

Wie schwierig war die Aufgabe?

viel zu einfach
zu einfach
gerade richtig
zu schwierig
viel zu schwierig

Aufgabe: Wie funktioniert ANKOMMEN?
Setzt Roboterteile aufeinander und fügt sie mit „STRG+L" zusammen. Der Roboter soll eurem Avatar Schritt für Schritt folgen. Die Schritte werden aber kürzer, je näher er kommt!

Hat die Aufgabe Spaß gemacht?

nein
eher nein
geht so
eher ja
ja

Vermutungen	Tests	Ergebnisse

Welches Teil operiert anders als beim SUCHEN? Wie?

Was war besonders einfach / besonders schwierig?

Figure 3: Worksheet for *Arrive*$_1$ (in German)

For each task, we recorded the perceived difficulty (left) on a scale from 1 (too easy) to 5 (too difficult; 3 is best) and the perceived joy (right) on a scale from 1 to 5 (5 is best). In the middle, there is the task description, three boxes for documenting experimentation (hypotheses, tests, results), one larger box for a narrative description of the topic (sometimes with a helping sketch), and a final box asking: what was easy/difficult in this task. With each new task, we handed out a new worksheet. The following section presents our analysis of collected data.

4.3 Results

Figure 4 displays the resulting learning paths for the six teams as dynamically assigned by the teacher in response to performance or shown interest. For each team, the average perceived difficulty and joy of doing a task is shown ("±" denotes the standard deviation).

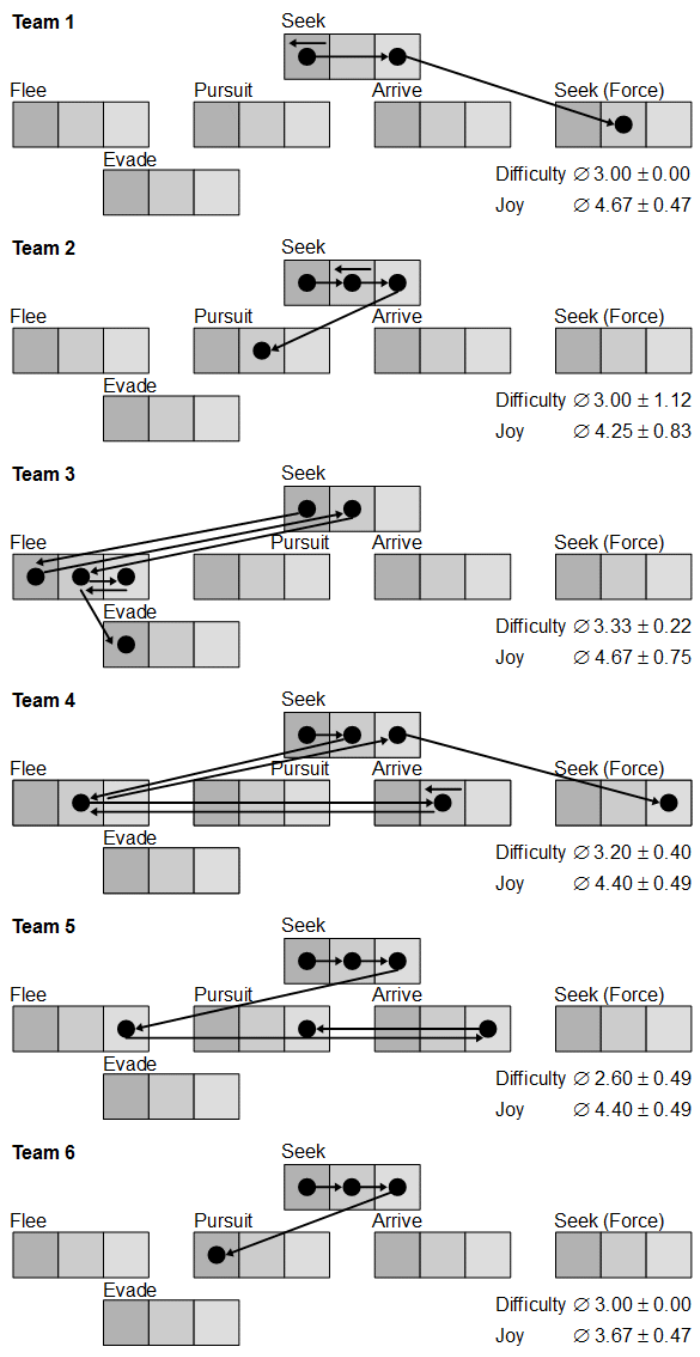


Figure 4: Resulting learning paths

Some things should be noted before interpreting the charts. Firstly, during the course, the task set was extended by “Seek (Force)” because Team 4 had expressed an interest in pushing robots by a constant force instead of instantaneous impulses. They had developed a vision of flying robots. This extended the surmise relation:

(1")	$\neg Seek_i$	$\Rightarrow$	$\neg Flee_1 \wedge \neg Flee_2 \wedge \neg Flee_3$ $\wedge \neg Pursuit_1 \wedge \neg Pursuit_2 \wedge \neg Pursuit_3$ $\wedge \neg Arrive_1 \wedge \neg Arrive_2 \wedge \neg Arrive_3$ $\wedge \neg Force_1 \wedge \neg Force_2 \wedge \neg Force_3$
(14)	$\neg Force_1$	$\Rightarrow$	$\neg Force_2$
(15)	$\neg Force_2$	$\Rightarrow$	$\neg Force_3$

Other student ideas led to the modification of tasks. Three teams wanted a different *Seek*$_3$: a robot that followed exclusively their own avatar. Moreover, sometimes the teacher decided to assign the same task twice as a repetition. Sometimes, students forgot to assign the difficulty and joy ratings, so not all tasks are part of the average calculations. Finally, a mistake caused the assignment of *Evade*$_1$ to Team 3 before they had completed a *Pursuit*. In accordance with the model, they did not solve this task because it was too difficult.

Looking at the quantitative results, we can consider the evaluation of the construction space a success. On average, perceived joy was > 4 for everyone except Team 6. The students in that team did provide positive comments in their final worksheet, however, and gave the overall course a 4. The students had fun. This can be attributed to the adaptation, but also to the playful nature of the course and the difference from usual school routine. Perceived difficulty turned out to be around the ideal average 3, with all standard deviations < 1 except for Team 2. This team awarded *Seek*$_3$ a 5 and did not make any progress from then. *Seek*$_2$ had been a 2 for them, on the other hand. Team 3 awarded the highest difficulties. As the end of their path shows, we were struggling to find appropriate tasks for them. Team 5 awarded the lowest difficulties. This was a very competent team. Had we supported them better (and not attended to students needing more help), they might have proceeded faster and reached more difficult tasks. Team 4 was probably as competent as Team 5, but exhibited a more chaotic working style, as can be seen in their learning path, which was largely determined by their own interests. They were trying out ideas quicker than corresponding tasks could be handed out.

Our two-dimensional model seems to support such different learning styles. Whilst Teams 3 and 4 worked fast, Teams 1, 2, and 6 can be subsumed under a slower-working type. This is not necessarily bad. In fact, these students may benefit from a dynamic degree of instruction towards construction the most. Team 5 started in a similar fashion, but quickly reached the highest degree of interactive freedom (coding). Actually, Team 6 had joined the course only in the second week. So, given more time, they might have developed a learning path similar to that of Team 5. Rather than needing the time like Teams 1 and 2, Teams 5 and 6 just seemed to want to do their work in an orderly manner. In the end, four teams had successfully changed or written bits of LSL code. For Team 3, on the other hand, it was good that we could move backwards from *Flee*$_3$ (Session 2) to *Flee*$_2$ (Session 3) again. Moving on from *Seek*$_3$ to *Pursuit*$_2$ for Team 2 did not work, unfortunately. This was at the end of the course and may have been a time issue, but we need to examine such transitions in the future. No one came close to *Evade*$_3$. This is okay, as every team did have their individual accomplishments, and an adaptive lesson should always keep something in store for the unexpectedly clever. But we did not see a solution to *Pursuit*$_3$, either. Maybe, it could only be reached after *Pursuit*$_1$ or *Pursuit*$_2$.

It was difficult to code or cluster the rather scarce qualitative data. Our twelve participants had produced less text than we had hoped for. Yet, we did observe trends. The students did understand the easier AI algebra content, but could not always put their insights into words. Many narratives are not precise. Some are incorrect. The students did understand their actions and what they meant in the instant they did them (i.e., while they were in their working memory). But they could not reflect on them afterwards. Did learning in a virtual world produce tacit knowledge? On the other hand, students were able to continue their work from one session to the next, and in the final worksheet most could transfer obtained knowledge to games they knew, some to new game ideas. Future studies should support journal writing in a better way than predefining boxes and keywords.

Students had problems setting up discovery learning experiments (which is also one issue in de Jong and van Joolingen 1998). For *Seek*$_1$, some good entries document combinations: "Head 3, Foot 3, Eye 3 $\Rightarrow$ fleeing robot; Head 1, Foot 1, Eye 1 $\Rightarrow$ following robot; Head 2, Foot 2, Eye 2 $\Rightarrow$ following robot." Experiments with code parameters (range, angle, speed, name etc.) worked in practice, but were not documented well. More complicated experiments are scarce in the students' notes. False observations or conclusions appear, too. Discovery learning worked best when students were following their own ideas, which, naturally, they seemed to understand better than our tasks. At construction levels 2 and 3, students programmed robots that followed only their own avatar. They compared *Seek* and *Flee*, so *Seek*$_1$; *Flee*$_1$; *Seek*$_2$; *Flee*$_2$ was actually a successful path for Team 3. One team experimented with rotation in 3D, another one proposed a circle/sphere of protection around a target so that *Arrive* could terminate (this is actually done in practice). Or they experimented with gravity, as in the second pre-study. After a while, every team had come up with creative ideas.

All in all, the students did not set up and perform experiments in a formal way, but tried out ideas quicker than they could think them through. They played! Maybe this is what game-based learning does: inspire a flow of thoughts rather than careful deliberation? Or, the instructional part needs improvement. In their final remarks, students did praise the value of being able to try things out (3 mentions in 12 final worksheets), the working style being close to practice (4), and the direct application of knowledge or direct visual feedback (5). But there were also critical voices that wanted

more theory (2) or coding instructions (2). This had already been a lesson learnt in the first pre-study: do attend to students who are there for the facts!
Cross-interpreting data and observations on the team level, the students most competent and capable of transfer (Teams 4 and 5) were those who wanted more theory. Teams that struggled more, valued practical work (1, 3), but also asked for more instructions on LSL (2). Adding more instruction (theory) to game-based construction (practice) would thus benefit different sorts of learners, yet it should also happen at different levels of complexity: introduction to coding versus theoretical concepts.

5. Conclusions and future research

In the application scenario presented, the construction space model did work as a tool for more fine-grained adaptation. A typical learning path was refining the easiest task until the highest level of construction, then continuing with the next task on that level (depth first). But for Team 3, relating tasks to each other at more instructional levels worked better (breadth first). Two-dimensional adaptation can address personal learning preferences in an inclusive way, such that competent students can experiment more freely (depth) without surging ahead too far in terms of learning goals (breadth).
Students had trouble with discovery and reflection. Discovery learning worked better when students tried their own ideas – is inventing one's own task a new highest degree of construction? The educational scenario will lead to personal accomplishments, but to have everybody solve the tasks, we have to support meta-cognitive thinking (planning, reflecting) and abstract Computer Science skills (plugging together modules, adjusting parameters, interpreting diagrams). The teacher should ensure that pursued new ideas remain related to the learning goals (here: AI) and can be shared with the rest of the class, perhaps in a subsequent reflection session.
We are currently investigating further approaches to injecting instructions into a constructivist environment: support for writing learning journals and automated suggestion of student actions (Martens and Maciuszek 2013). We are also employing graph notations as more structured representations of mental models.
Future research might study the relation of learning path patterns to *cognitive learning styles*. It may be possible to define certain learner stereotypes from which the teacher can start adaptation. Finding such patterns would require studies with larger groups.
When adaptation along construction spaces is better understood, it can be automated in an ITS that chooses the next task automatically. Alternatively, the ITS might present annotated links from which students can make informed selections. Research would have to invent and test algorithms for this. A challenge for any automation, however, would be the support of creative ideas, i.e. new tasks 'invented' by students.

Acknowledgements
We thank Bert Schröder, teacher and headmaster at Erasmus-Gymnasium Rostock, as well as the grade 12 Computer Science course 2012/13 (Christopher, Chung, Florian K., Florian R., Han, Jan, Julia, Michael, Mikhail, Nico, Nikolai, Patrick). Heike Cantow established this contact. Maria Neumann helped with the study design. Further thanks go out to the students of the pre-studies and to Christian Schönfeldt for his co-ordination efforts.

References

Albert, D. and Held, T. (1999) "Component-based Knowledge Spaces in Problem Solving and Inductive Reasoning", in *Knowledge Spaces*, eds D. Albert and J. Lukas, Lawrence Erlbaum, Mahwah, pp 15–40.
Collins, A., Brown, J.S. and Newman, S.E. (1987) *Cognitive Apprenticeship: Teaching the Craft of Reading, Writing, and Mathematics.* Technical Report No. 403. Urbana-Champaign: Center for the Study of Reading, University of Illinois.
Crabtree, A. (2003) *Designing Collaborative Systems*, Springer, London.
Enfield, J. (2012) *Designing an Educational Game with Ten Steps to Complex Learning.* PhD thesis, Indiana University.
de Jong, T. and van Joolingen, W.R. (1998) "Scientific Discovery Learning with Computer Simulations of Conceptual Domains", *Review of Educational Research*, Vol. 68, No. 2, pp 179–201.
Lelouche, R. (1999) "Intelligent Tutoring Systems from birth to now", *Künstliche Intelligenz*, Vol. 13, No. 4, pp 5–11.
Maciuszek, D., Martens, A., Lucke, U., Zender, R. and Keil, T. (2013) "Second Life as a Virtual Lab Environment", in *Synthetic Worlds*, eds A. Hebbel-Seeger, T. Reiners & D. Schäffer, Springer, New York, pp 165–202.

Martens, A. (2004) *Ein Tutoring-Prozess-Modell für fallbasierte Intelligente Tutoring-Systeme.* PhD thesis, University of Rostock.
Martens, A. and Maciuszek, D. (2013) "Balancing Instruction and Construction in Virtual World Learning", in *Serious Games and Virtual Worlds in Education, Professional Development, and Healthcare*, eds K. Bredl & W. Bösche, IGI Global, Hershey, pp 15–40.
Reynolds, C.W. (1999) "Steering Behaviors for Autonomous Characters", in Proceedings of Game Developers Conference 1999. San Jose, 15–19 March 1999. Miller Freeman Game Group, San Francisco, pp 763–782.
Rieber, L.R. (1992) "Computer-based Microworlds: A Bridge Between Constructivism and Direct Instruction", *Educational Technology Research and Development*, Vol. 40, No. 1, pp 93–106.
Schulmeister, R. (2003) "Taxonomy of Multimedia Component Interactivity: A Contribution to the Current Metadata Debate.", *Studies in Communication Sciences*, Vol. 3, No. 3, pp 61–80.
Sweetser, P. (2005) *An Emergent Approach to Game Design.* PhD thesis, University of Queensland.