<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper presented at *2010 IEEE Conference on Computational Intelligence and Games, CIG2010; Copenhagen; 18 August 2010 through 21 August 2010*.

Citation for the original published paper:

Syberfeldt, A., Syberfeldt, S. (2010)

A serious game for understanding artificial intelligence in production optimization.

In: *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, CIG2010* (pp. 443-449). IEEE

<http://dx.doi.org/10.1109/ITW.2010.5593321>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-4543>

A serious game for understanding artificial intelligence in production optimization

Anna Syberfeldt and Sanny Syberfeldt

Abstract— This paper describes a serious game that can be used to teach and demonstrate production optimization using artificial intelligence techniques. The game takes place in a physical Lego factory and is designed to be an enjoyable learning experience.

I. INTRODUCTION

This paper describes a game for teaching the concept of production optimization and how it can be improved using artificial intelligence (AI) techniques. The goal of production optimization is to increase the efficiency of a production process, such as construction or assembly, a vital activity for industries aiming to make profit on a competitive market. Production processes are often too complex to effectively optimize manually due to combinatorial relationships, uncertainty factors, and nonlinearities. AI techniques have been shown to be powerful in optimizing a wide range of production problems, such as engine construction, robot control, and resource planning [1-3].

Production optimization using AI techniques is one of the research areas at the University of Skövde, Sweden, and the game was designed to demonstrate research problems and results to the general public and to students. Since the game has an educational purpose in addition to entertainment, it is a *serious game* [4]. Specifically, it is an *educational simulation* [5] with goal-oriented tasks and reward structures. These game concepts are used to encourage playful experimentation and exploration of the system, hopefully improving the simulation's educational potential.

The simulation shows a physical production process in the form of a miniature factory built using Lego bricks and Lego's computerized building system Mindstorms NXT. The factory refines "bubble gums" (represented by marbles) by means of a production process consisting of three operations; flavoring, drying, and polishing. The operations are made-up and do not correspond to real bubble gum production. Setting up a real production process is too expensive and complicated, and is unnecessary for the game's purpose.

The player assumes the role as production manager for the factory, with privileges to change the production process. The player's objective is to change the process in a way that

makes the production as efficient as possible in order to maximize profit. During the course of the game, the effect of a change in the production process becomes immediately visible in order to guide the player in the configuration task.

In the next section, the details of the factory and the production process are further described.

II. THE LEGO FACTORY

A. General construction

Figure 1 shows the layout of the factory, which is made up of three production stations built using Lego and Lego Technic. The base of each production station is about 40x40 centimeters, and in total the factory is about 1.5 meters in height and 1 meter in width. The stations are placed above each other in a spiral and are connected with PVC tubes, allowing gravity to transport the bubble gums between stations. A conveyor belt transports the bubble gums from the last station to the first, making the system a closed loop and removing the need to feed new marbles to the system at runtime. The system can thus run indefinitely without manual intervention. The conveyor belt consists of 400 individual caterpillars, also from the Lego series. An XL motor from the Lego Power Function series drives the belt.

B. Production stations

The three production stations in the factory simulate refinement of bubble gums. The stations are built to be entertaining to watch and easy to understand, rather than realistic. All three production stations are controlled using Lego Mindstorm NXT 2.0 computers (also called "intelligent bricks"). The NXT computer has a 32-bits micro processor, four input sensors, and three motor outputs. These sensor inputs and motor outputs are used to control the production stations.

The general design is the same for all three stations. When a bubble gum arrives at the station, a light sensor reads the color of the gum. If the specific color should be processed in the station, it is placed in a buffer in front of the station – otherwise it passes by the station. The buffer holds bubble gums that await processing, similarly to a real production process. The space of a buffer (how many gums it can hold) is variable and can be changed by the production manager (the player). Each buffer has a minimum size of 1 and a maximum size of 5.

Dr. Anna Syberfeldt is with Intelligent Automation within the Virtual Systems Research Center at the University of Skövde, Skövde, Sweden (corresponding author to provide phone: +46(0)500448577; e-mail: anna.syberfeldt@his.se).

Dr. Sanny Syberfeldt is with the InGameLab within the Virtual Systems Research Center at the University of Skövde, Skövde, Sweden (e-mail: sanny.syberfeldt@his.se).

Fig. 1. Lego factory.

The first production station simulates flavoring. In this station (Fig. 2), the bubble gum is placed in a spoon-like construction and dropped into a container, which is then covered with a lid for a few seconds. When the lid opens, the spoon lifts up the bubble gum and drops it into the tube. The bubble gum then proceeds to the second station.

Fig. 2. First production station: flavoring.

The second station simulates drying. In this station (Fig. 3), a bubble gum is placed at the end of an arm that spins quickly in order to air dry the bubble gum. The bubble gum is then put into the tube and rolls to the third station.

Fig. 3. Second production station: drying.

The third station simulates polishing. In this station (Fig. 4), the bubble gum is placed under a rotating brush. After polishing, the bubble gum rolls to the conveyor belt and is transported back to the first station.

Fig. 4. Production station 3: polishing.

C. Production process

The system handles four different types of bubble gums: blue, red, green, and yellow. The various types are processed differently, meaning, for example, that a blue bubble gum should pass through certain stations while skipping others. A red bubble gum may follow different rules. The production process is further complicated by the fact that for a specific type of bubble gum there is a set of different legal process configurations, as shown in Table I. A blue bubble gum, for example, could go through the stations for flavoring and drying, or through the stations for flavoring and polishing, or through only the polishing station. Only one of these three configurations can be used, and which one to use up to the production manager (the player) to decide.

TABLE I
POSSIBLE CONFIGURATIONS OF THE PRODUCTION FLOW

Blue	Red	Green	Yellow
Flavor & dry OR Flavor & polish	Flavor OR Dry OR Polish	Flavor & dry OR Dry & polish OR Flavor & polish	Flavor & polish OR Dry & polish

D. Software

As previously mentioned, the production stations are controlled using NXT computers. These computers run software written in NXT-G, Lego’s official programming language for the NXT platform (see mindstorms.lego.com for more information). The NXT-G programs read sensor input (e.g. from a light sensor) and react to this using motor outputs (e.g. changes a switch to steer the direction of the bubble gum). A sensor input is also communicated to a higher-level control system run on a PC, in order to collect data (such as how many bubble gums that have passed through a station). The higher-level control system is written in C#, and communication between this system and the NXT computers take place using the Bluetooth protocol. The communication is two-ways; the higher-level system forwards user commands in form of configuration changes to the NXT computers. Such configuration changes are provided by the user using a touch screen attached next to the factory.

III. GAME DESCRIPTION

A. Game play

As the factory production manager, the player’s task is to find the best configuration possible for the production process in order to maximize profit. The player can change two aspects of the production process; the production flow, and the buffer sizes. To make production efficient, the production flow and the buffers should be configured in a way that minimizes work-in-process and maximizes throughput. Work-in-process is the number of bubble gums currently being processed, while throughput is the number of bubble gums that the plant outputs per minute.

Work-in-process and throughput are two important objectives in real production scenarios. The amount of work-in-process in the production system should be kept as small as possible in order to avoid capital being tied up in partially completed products. At the same time, the throughput rate should be kept as high as possible since the higher the throughput rate, the better the productivity and profitability.

Achieving a low work-in-process and a high throughput rate is not trivial, neither in reality nor in the bubble gum factory. With large buffers, the throughput increases since machine starvation is avoided and the impact of the interference caused by variability in processing times decreases. However, with large buffers the work-in-process also increases since a large number of bubble gums are kept in the system. Conversely, small buffers decrease both the

work-in-process and the throughput. Further complexity is added by the fact that it is not only the buffer sizes that are variable, but also the process flow (Table 1). Predicting the effect of various configurations of the buffers and the process flow is not possible, but each of them must be tested in order to find best one. Testing all 4500 possible configurations is impossible in practice. One of the goals of the game is to make the player realize the infeasibility of exhaustive testing and the benefits of AI-based optimization.

B. Reward system

The player wins the game if the optimal configuration of the factory is found. However, as discussed in the previous section, the game is designed so that winning is almost impossible. The idea is that the player should realize the complexity of production optimization and that it cannot be effectively carried out by a human.

Although it is virtually impossible to win the game, the player can reach different success levels: gold, silver, or bronze. Each level defines a minimum throughput rate and a maximum amount of work-in-process that must be achieved. The levels are relatively easy to reach in order to motivate the player to continue playing. A player without previous knowledge of production or optimization generally reaches the bronze level after about 5 minutes, the silver level after about 8 minutes, and the gold level after about 10 minutes. When reaching the gold level, the player usually has a basic understanding of the production system and how it can be improved by changing its configuration.

When reaching the gold level, the player may continue the game and try to improve their results in order to take a spot on the high-score list. This list records, in order, the ten best results from all plays so far.

C. Game structure

Before the game starts, the player is briefly introduced to the concepts of production, production optimization, and AI. This is done by presenting a text for the player to read. After the introduction, the game is started and a control panel is shown to the user (Fig. 5). This panel shows control buttons that can be used to configure the bubble gum production (that is, change the buffer sizes and the process flow).

Fig. 5: User control panel.

During play, the current amount of work-in-process and throughput rate is shown in the control panel. Along with these numbers the player's current success level (none, bronze, silver, or gold) is shown, and also the improvements needed to reach the next level.

Work-in-process and throughput are updated immediately when a configuration change is made in order to guide the player in the configuration task. This is possible by calculating the numbers in a simulator instead of using the physical factory. If the physical factory is used, the player must wait several minutes to clearly see the effects of a configuration change. Improving the production process requires extensive experimentation, which the game also aims to encourage. Without the immediate feedback from the simulator the player would likely be bored or frustrated. An exact 1:1 mapping between the simulator and the physical factory is not possible due to imprecision in sensors and motors, but the differences are so small that they are not noticeable for the player.

When the player is satisfied with their configuration and ends the game, the player's results are compared to those of an AI-based optimization algorithm (described in the next section). The AI has been run at beforehand and allocated three seconds of runtime. The time spent by the player is also shown alongside the AI's results. Usually, the AI performs considerably better than the player, despite the short run time. By comparing the results, the player hopefully realizes the need and power of using AI in production optimization.

D. Comparison to previous work

The serious game described in this paper is not the first attempt to teach production concepts. Hunecker [6] presents a game that aims to help the player understand complex production systems. The game uses a resource-based process model in which a process transforms resources from one form to another. A machine may, for example, transform energy into a product. The player's goal is to use the resources in a way that maximizes profit.

Another serious game within the area of production systems is presented by Duin et. al [7]. In this game, a player manages a production system and competes on a global market. The underlying simulation of the game is action-driven; an input from the player drives the simulation towards a specific direction, which is displayed as feedback information to the player. The game aims to teach young management staff strategic decision making in global production.

Comparing the games by Hunecker [6] and Duin et. al [7] to the game presented in this paper, a main difference is that the former games are virtual computer simulations, while the latter is a physical simulation. Another difference is the purposes of the games; Hunecker [6] aims to teach operation of production systems in general, Duin et. al [7] aims to teach decision making, and the game presented in this paper aims to teach production optimization using AI. The details of the AI implementation are further described in the next section.

IV. AI OPTIMIZATION

To automatically optimize the production process, an optimization module has been developed and integrated into the higher-level control system described in the previous section. Many different AI techniques are possible in production optimization, and in this case an evolutionary algorithm has been used. The basics of this type of algorithms are presented in Section IV.A, while Section IV.B describes the implementation details of the evolutionary algorithm implemented for the bubble gum factory.

A. Evolutionary algorithms

The idea behind evolutionary algorithms is to use computational models of evolutionary processes in the design and implementation of problem solving applications. Based on Darwin's theory of "survival of the fittest", candidate solutions to a problem are iteratively refined. Generally, an evolutionary algorithm consists of a genetic representation of solutions, a population-based solution approach, an iterative evolutionary process, and a guided random search.

In evolving a population of solutions, evolutionary algorithms apply biologically inspired operations for selection, crossover and mutation. The operators are applied in a loop, and an iteration of the loop is called a generation. The solutions in the initial population are usually generated randomly. During each generation, a proportion of the solutions in the population is selected to breed offspring for the next generation of the population. Solutions are selected based on their fitness, representing a quantification of their optimality. Typically, solutions with high fitness have a higher probability to be selected, but to prevent premature convergence, it is common that a small proportion of solutions with worse fitness are also selected. From the solutions selected, new solutions are created to form the next generation of the population. For the creation of each new solution, two parent solutions are chosen and through mating (called crossover) an offspring is produced. Occasionally, the new solution can undergo a small mutation in order to keep the diversity of the population large and avoid local minima. A mutation usually involves changing an arbitrary part of a solution with a certain probability.

When the new population is formed, the average fitness will have generally increased. The process of evolving generations continues until a user-defined termination criterion has been fulfilled, for example that the best solutions in the last n generations have not changed.

B. Algorithm implementation

In the bubble gum factory, there are two optimization objectives that must be considered by the evolutionary algorithm: work-in-process, and throughput. The difficulty with this problem is that there is no single optimal solution with respect to both the objectives, as improving the performance of one objective means decreasing the performance of the other. Instead of a single optimum, there is a set of optimal trade-offs between the conflicting objectives, which is usually called *Pareto-optimal* [8]. For

Pareto-optimal solutions there is no other solution that is superior in one objective without being worse in another. Different Pareto fronts can also be identified in a population. Rank 1 includes the Pareto-optimal solutions in the complete population, and rank 2 the Pareto-optimal solutions identified when temporarily discarding all solutions of rank 1, and so on.

In order to manage multiple objectives, specific multi-objective evolutionary algorithms have been suggested. Instead of only seeking a single optimum, these algorithms maintain a set of Pareto-optimal solutions. One commonly used multi-objective evolutionary algorithm is the elitist non-dominated sorting genetic algorithm (NSGA-II) [9]. This is also the algorithm implemented for optimizing the bubble gum production.

In NSGA-II, the selection of solutions for the next generation is done from the union of the parent population and the offspring population (both of size N). The union is sorted in Pareto fronts, and the next generation of the population is formed by selecting solutions from one of the Pareto fronts at a time. The selection starts with solutions in the best Pareto front, then continues with solutions in the second best front, and so on, until N solutions have been selected. All the remaining solutions are discarded. The selection procedure is illustrated in Fig. 6 (adopted from [8]). The detailed implementation of NSGA-II is found in [9].

Fig. 6. General procedure of NSGA-II.

The parameter settings for the implemented NSGA-II algorithms are provided in Table II.

TABLE II
OPTIMIZATION ALGORITHM PARAMETER SETTINGS

Population size	50
Offspring population size	50
Probability of mutation	0.1
Crossover operator	Single-point
Crossover probability	0.8
Parental selection operator	Tournament selection

V. GAME-LIKE EXERCISE USED IN AI COURSE

The Lego factory and the configuration game are used as an assignment in the course “Artificial Intelligence for Industrial Applications” offered by the university. The Lego

factory provides a setting that is close to reality, and optimizing its bubble gum production assists students to deepen their understandings of AI concepts. We also believe that experiential learning opportunities enhance and enrich the learning experience.

In the assignment, the students should construct their own evolutionary algorithm for optimizing the bubble production (that is, buffers and the production flow) with respect to work-in-process and throughput. The exercise is to be solved in groups of 1-2 students. The students are not given any instruction on *how* to construct the evolutionary algorithm, but they need to find this out themselves (so called problem-based learning). They are not allowed to use a code library for solving the task, but must develop the evolutionary algorithm from scratch. Furthermore, the algorithm must be multi-objective, that is, support Pareto optimization.

During the initial stage of the development, the students use only the simulator to evaluate work-in-process and throughput. When they have a first version of a complete algorithm, they start to evaluate the algorithm’s performance also on the physical factory.

To pass the assignment, the algorithm must achieve gold level when run on the physical factory. The algorithm is allocated a maximum of 300 solution evaluations to reach this goal; if it needs more evaluations it is considered to be too inefficient and the student will fail the assignment.

At the end of the course, an optional competition takes place among the students in order to elect the best algorithm. For this competition, the possible configurations of the process flow (Table I) are randomly changed, and the maximum size of each buffer is randomized between one and six. The randomization procedure is unknown to the students before the competition starts. This is to promote algorithms that are designed to be general, and impede those which are customized and perform well only under known circumstances.

VI. EVALUATION

To evaluate the game, three target groups were used: general public, production engineers, and students. The evaluation undertaken is qualitative, rather than quantitative.

A. General public

The game was demonstrated to the general public during the opening of a new research building at the university. The opening had several hundreds of visitors, of which about fifty took a closer look at the Lego factory. However, only a few of these were actually willing to play the game. Most of the spectators expressed that the game seemed too complicated, and preferred to just watch the Lego factory and taste the bubble gums that were offered. However, while the spectators were watching the Lego and chewing their bubble gums, it was possible for the researchers to talk about the concept of AI-based production optimization and the university’s research in the field. So although the game as such was clearly no success for the general public, the aim of

the Lego factory – namely to communicate research problems and results – was fulfilled nevertheless.

B. Production engineers

The game was presented to a group of six production engineers from the industry during a visit to the university. The engineers showed great interest in the game and all of them wanted to try it out. They easily understood the production set-up, and had no problems to understand how to play the game. All of them played until they achieved gold level, which took between 4-10 minutes. The last players needed considerably shorter time to achieve good results compared with the first ones, since they had watched and analyzed the production process for a longer time.

When asked about their game experience, the production engineers pointed out that they clearly recognized the optimization problem from their own production sites, and that the scenario was realistic. They also stated that the game had provided an eye-opener for automatic optimization in general, and specifically for AI-based optimization. However, they emphasized that the information provided by the game about AI-techniques in production optimization was too basic. They expressed that more details and a few “success stories” from industrial applications would be needed in order to convince them to seriously consider introducing AI-based optimization in their own production.

C. University students

As described in Section V, an assignment based on the game is part of the course “Artificial Intelligence for Industrial Application”. The course has run once since the game was constructed with 13 participating students. After the course had ended, the students were asked to share their opinions about the assignment by answering three questions in writing. The first questions asked about positive aspects, the second one about negative aspects, and the third one about possible improvements to the assignment.

On the positive side, a majority of the students mentioned the experience of working with a physical system, and several of the students expressed that this was their first time doing so. They liked the opportunity to experiment and put the theories taught into practice. Furthermore, the students pointed out the game elements as encouraging and motivating in solving the assignment. All of the students believed that the assignment had given them useful knowledge when it comes to AI-based production optimization.

On the negative side, some of the students mentioned the open-ended nature of the assignment. As described in Section V, the students were not provided any instructions on *how* to solve the task, but they needed to find this out themselves (problem-based learning). However, having no clear path to go was experienced as frustrating to some of the students.

VII. CONCLUSIONS AND FUTURE WORK

This paper describes a serious game for teaching the concept of production optimization and how it can be improved using artificial intelligence (AI) techniques. Production

optimization using AI techniques is one of the research areas at the University of Skövde, Sweden, and the game was designed to demonstrate research problems and results to the general public and to students. As far as the authors are aware of, it is the first serious game for AI-based production optimization.

The game is meant to give the player an understanding of production optimization, the difficulties of production optimization, and the power of AI techniques for this task. Initial observations from evaluating the game show that the intended learning outcomes are fulfilled to a certain degree, but that much work remains to improve the game experience. Some ideas how this can be done are discussed in the remainder of this section.

Right now there is only one game mode, although the players may vary from people that hardly ever heard of production, to production engineers with long experience from working in industry. A better design would probably be to have two game modes, one for domain experts, and one for newcomers. In the former, more emphasis could be put on technical issues regarding AI-based optimization and how AI techniques could be used for different optimization problems. More focus must also be put on convincing the players to evaluate using the techniques themselves at the production site. With respect to the beginner mode, the focus could instead be popular scientific and the concepts could be introduced on a very basic level. To lower the barrier for trying the game, the beginner could start with a very simple task, such as configuration of only one station. The task could then gradually become more challenging as the player becomes more skillful.

To increase interest for playing the game, the plain graphical user interface needs to be more attractive. Today’s interface is “designed” by a programmer, and a graphics artist or user interface expert could certainly do much better. The user interface could also be improved with respect to player assistance. Right now, the only assistance the player gets during the game is two numbers; one for throughput, and one for work-in-process. Players – especially beginners – probably need more guidance than just two numbers, such as tips about what to try in the configuration task. Player feedback could also be improved, for example by complementing the numbers with two bars that clearly show the current achievement level and requirements for reaching the next level. Sound might also be used as feedback to indicate good or bad player actions.

The AI part of the game is also an important aspect to improve. A visualization of the AI’s problem solving process at the end of the game, rather than just presenting its final result, might enhance the player’s understanding and increase the credibility. Another idea to enhance the understanding is to allow the player to influence the AI by changing its parameters (such as mutation rate and crossover for an evolutionary algorithm).

Besides implementation issues, it is important to improve the evaluation of the game. So far, only a qualitative evaluation of limited scope has been done. Furthermore, no formal method was used for the evaluation, and no concrete

measurements were applied. The next step of this work is therefore to perform a solid evaluation. Such an evaluation should probably be based on one of the comprehensive frameworks suggested in the literature. One such framework for serious games that could possibly be used is the one suggested by Connolly et al [10]. The major aspects evaluated in this framework are learner performance, motivation, learner perception, attitudes that can affect the effectiveness of the game, learner preferences (such as learning styles), collaboration, and environment. Wouters et al (2009) report that the application of the framework to 28 different serious games indicates that the framework is useful.

In this context, it can be mentioned that evaluation is a weak part not only of the work presented in this paper, but of serious games in general. Connolly et al (2008) report that an analysis of 1400 papers of serious games reveals that only 72 of the studies (approximately 5%) apply some kind of evaluation measurement. The reason behind this low number is an interesting aspect to look further into. Could it be that the evaluation frameworks proposed in the literature are not generally applicable? Or that game developers do not know how to use them in practice? Or something else? A deeper analysis of these questions would be interesting, especially with a focus on games based on hardware simulations.

ACKNOWLEDGMENTS

The authors want to thank Ingemar Karlsson and André Geertsens for their help with the Lego building process.

REFERENCES

- [1] J. Boesel, R. Bowden, F. Glover, J. Kelly, and E. Westwig. Future of simulation optimization, Proceedings of the 2001 Winter Simulation Conference, Arlington, VA, 2001, pp. 1466–1470.
- [2] Y. Ong, P. Nair, A. Keane and K. Wong. Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems, Knowledge Incorporation in Evolutionary Computation, 2004, pp. 307–332.
- [3] M. Emmerich, N. Beume, and B. Naujoks. An EMO algorithm using the hypervolume measure as selection criterion, Proceedings of Evolutionary Multi-Criterion Optimization, Vol. 3410 of Lecture Notes in Computer Science, Publisher Springer Berlin / Heidelberg, Guanajuato, Mexico, 2005.
- [4] D. Michael and C. Sande. Serious Games: Games That Educate, Train, and Inform. Course Technology PTR, 2005.
- [5] C. Aldrich. The Complete Guide to Simulations and Serious Games: How the Most Valuable Content Will be Created in the Age Beyond Gutenberg to Google. John Wiley & Sons Ltd, 2009.
- [6] F. Hunecker. A generic process simulation-model for Serious Games in manufacturing, Proceedings of SAGSET 2008 - 38th Annual Conference. Teaching and Learning through Gaming and Simulation, University of Nottingham, United Kingdom, 2008.
- [7] Heiko Duin, Manuel Oliveira, Ali Saffarpour, 2007, "A Simulation Model for Virtual Manufacturing Environments for Serious Games", in 13th International Conference on Concurrent Enterprising, K. S. Pawar, K.-D. Thoben, & M. Pallot, eds., 4-6 June 2007, Sophia-Antipolis, France.
- [8] K. Deb. Multi-Objective Optimization using Evolutionary Algorithms, second edn, JohnWiley & Sons Ltd, 2004.
- [9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan . A fast and elitistmulti-objective genetic algorithm NSGA-II, KanGAL Report 2000001, Indian Institute of Technology Kanpur, India, 2000.
- [10] T.M. Connolly, M. Stansfield, and T. Hainey, Towards the Development of a Games-Based Learning Evaluation Framework. In Thomas Connolly & Mark Stansfield (Eds.), Games-based learning advancements for multi-sensory human computer interfaces, pp. 251-273. Hershey: Information Science Reference, 2009.
- [11] P.J.M. Wouters, E.D. van der Spek, and H. van Oostendorp. Current practices in serious game research: A review from a learning outcome perspective. In Thomas Connolly & Mark Stansfield (Eds.), Games-based learning advancements for multi-sensory human computer interfaces, pp. 232-250. Hershey: Information Science Reference, 2009.