1-1-2013

Mobile games with intelligence: a killer application?

Philip Hingston
Edith Cowan University

Clare Bates Congdon

Graham Kendall

10.1109/CIG.2013.6633660

Mobile Games with Intelligence: a Killer Application?

Philip Hingston
School of Computer and Security Science
Edith Cowan University
Australia
p.hingston@ecu.edu.au

Clare Bates Congdon
The Department of Computer Science
University of Southern Maine
Portland, ME, USA
congdon@usm.maine.edu

Graham Kendall
School of Computer Science
University of Nottingham, UK and
University of Nottingham,
Malaysia Campus
graham.kendall@nottingham.ac.uk

***Abstract*—Mobile gaming is an arena full of innovation, with developers exploring new kinds of games, with new kinds of interaction between the mobile device, players, and the connected world that they live in and move through. The mobile gaming world is a perfect playground for AI and CI, generating a maelstrom of data for games that use adaptation, learning and smart content creation. In this paper, we explore this potential killer application for mobile intelligence. We propose combining small, light-weight AI/CI libraries with AI/CI services in the cloud for the heavy lifting. To make our ideas more concrete, we describe a new mobile game that we built that shows how this can work.**

I. Introduction

Games are an appealing application to showcase AI (Artificial Intelligence) and CI (Computational Intelligence) approaches because they are popular and ubiquitous, attracting a diverse range of users.

Mobile games are easier to bring to market than commercial (large scale) video games. This makes them a practical choice for development and study in an academic environment, using relatively small teams of academics and students, who are able to work on relatively low budgets. For example, the small screen size and lack of powerful graphics hardware typical of mobile devices means that simple graphics, often only 3 or 4 inches in size, are expected, so that large teams of highly skilled artists and 3D modellers are not required.

Mobile devices usually provide a wider variety of input data (touch, location, images, video, sound, acceleration, orientation, personal data, data from/about other users etc.) than is normally available on a desktop or laptop computer and offer a full range of output options (images, video, animation, sound, vibration, wireless, bluetooth, infrared) as well. In addition, the popularity of mobile devices allows developers to recruit large numbers of casual users, whose interactions provide another potentially large data source for game data mining, using techniques such as those described in [1]. Novel game mechanics and interaction methods might be made possible by processing these input data using AI and CI methodologies.

Computational power, memory and battery life present potential obstacles to intensive AI/CI-based games, and some potential designs will require offloading some of the computation to servers. It might also be difficult to implement large-scale, complex game worlds due to the limited resources that are available. There are also significant challenges in developing AI/CI libraries that can work with low memory, limited battery power etc., adapting or developing AI/CI methods to work effectively in games that are played in short bursts, using unreliable communications, and providing real-time responses. However, these constraints provide significant research opportunities.

Mobile devices are still "young" enough to provide opportunities for developers to implement innovative products without having to employ large specialist teams (e.g. graphic designers, musicians etc.), although some specialists are still required of course. However, devices are becoming more capable – for example, the original iPhone had a screen resolution of 480x320 pixels, a single 2 Megapixel still camera, and a storage capacity of 4-8 GB, while the iPhone 5 is 1136x640 pixels, has two 8 Megapixel cameras and can record 1080p HD video at 30 fps, has a storage capacity of 16-64 GB, and has in-built voice recognition. Applications are also becoming more sophisticated — for example, technologies like Web3D and game engines like Unity3D are bringing 3D graphics to mobile platforms [2]. Inevitably, game players will come to expect more and more from mobile games, so the opportunity for small players and enthusiasts will not last long (perhaps several years, in our estimation). Those who are interested in this area might want to explore, and capitalize, on those opportunities now. Moreover, AI/CI both provide significant opportunities both in terms of research challenges and also to make the games more interesting and more fun to play. We would like to see the research community take up the challenge to showcase what can be done with the limited resources available on mobile devices, but also utilizing the larger number of sensors (e.g. movement detection) and other options (e.g. location awareness) which are not available on traditional "living room" game consoles.

The aim of this short paper is to outline the limitations of mobile computing, with respect to utilizing AI/CI, but also to draw out some of the potential advantages that can be exploited now (or certainly in the years to come) as the convergence of technology continues and offers greater opportunities than are available at present.

The rest of the paper is presented as follows. In the next section, we present the (limited) work that has been carried out on AI/CI for mobile devices. In Section III we lay out what we

believe are the defining characteristics of mobile environments. In Section IV we outline the challenges faced when using mobile devices. Section V presents the opportunities that arise when using a mobile device, rather than a desktop, console, or other *stationary* computer. In Section VI we provide some insight as to what AI/CI can offer mobile computation. We also outline some possible projects that would be feasible at this time, as well as some thoughts as to what might be possible in the next 5-10 years. In Section VII, we describe a new mobile puzzle game that we built to illustrate some of these ideas (and also for the fun of doing it!) Section VIII concludes the paper.

II. Prior Work

We were able to find only a limited amount of work that considers AI/CI in mobile games and there seems to be limited scientific literature about using AI/CI on mobile devices at all. In this section, we summarize the few papers we did find, on AI/CI for games as well as for non-games on mobile devices.

In one gaming example, Aiolli and Palazi [3] adapted an existing machine learning algorithm to enable it to work within the reduced computational resources of a mobile phone. Their target was the game "Die guten und die bösen Geister", which is a board game requiring the player to identify which game pieces (Ghosts) are "good" and which are "bad". Therefore, an AI opponent for the game would need to be able to perform a simple classification task. The more usual classification algorithms were rejected on the basis of requiring too much memory or too much computation. Instead the authors opted for a very simple system based on two prototype feature vectors, one for good and one for bad ghosts. Unfortunately, they did not report any comparison of performance of this simple scheme over more complex classifiers, but the point to note is that for such applications, there is a trade-off to evaluate between accuracy and computational resource requirements. There was also no evaluation of the different schemes in terms of player satisfaction.

In a more recent example by Jordan et al. [4], the authors report on a research prototype *BeatTheBeat*, in which game levels are matched to background music tracks based on features extracted from the audio signal, and these are allocated to cells on a game board using a self-organising map.

In a paper discussing the potential uses of AI methods in serious mobile games, Xin [5] suggests that, while AI methods could add value to such games, computational requirements might require a client-server solution, offloading the AI to a server.

Although not focusing on games, Kruger and Malaka [6] argue that AI has a role in solving many of the challenges of mobile applications, including:

- Location awareness;
- Context awareness;
- Interaction metaphors and interaction devices for mobile systems;
- Smart user interfaces for mobile systems;
- Situation-adapted user interfaces.

Their paper introduces a special issue of the journal Applied Artificial Intelligence containing articles describing the state of the art as it was in 2004. Many of these same challenges may provide opportunities for novel mobile game concepts based on AI/CI.

In [7], Baltes et al. describe their experience with implementing high-level real-time AI tasks such as vision, planning and learning for small robots, using smart phones to provide sensing, communication and computation. Although the application is not to games, and their aims are different from ours, many of the research challenges in terms of implementing AI solutions with limited computational resources are similar. Their robots' agent architectures are based on behaviour trees described using a XML schema, and translated off-line into efficient C code. Behaviour trees are also an increasingly popular approach to agent design for games, in both academic research (see, for example [8]), and in commercial games such as the first-person shooter Halo2 [9]. Vision is based on fast approximate region detection. A standard algorithm was found to be too slow and was modified to take advantage of specific domain knowledge (e.g. expected object colors). Another high-level task that they tackled was multi-agent simultaneous location and mapping (SLAM). Once again, the task was simplified by taking advantage of the structured environment (robot soccer). BlueTooth was used to share information between agents. A particle filter method was used to maintain estimates of the robots' poses, with a limited particle population size dictated by the available memory. We see that the researchers used a variety of strategies to cope with the limitations of the computing platform: offline preprocessing, modification and simplification of algorithms for specific tasks and environments, and sharing of information between mobile devices. We expect that some of the same strategies and even some of the same algorithms will be applicable in both the robotics and games domains.

III. Characteristics of a mobile environment

Our working definition of a mobile device for game playing is a device that is networked, and is small enough to be mobile, yet still provides a platform for general computation. In the future, one might imagine that many kinds of mobile devices might be used in games. For example, a car's GPS system might be used in a futuristic version of a scavenger hunt car rally (scavenger hunt games for mobile phones already exist - e.g. SCVNGR, textClues). However, at the present time, we are chiefly thinking of smart phones and tablets.

While computational resources (CPU, memory, persistent storage) are available on these devices, they are all limited in comparison to standard platforms, and limited battery power is an additional consideration.

On the plus side, these devices usually have a number of other features that are often not available, and especially not all together, on "standard" gaming platforms:

- **location services** - whether by GPS, WiFi or cell tower triangulation;
- **personal ownership** - generally one person is more or less the sole user of a particular device;
- **Internet access** - to data, services and other users;

- **multiple modes of connectivity** - WiFi, Bluetooth, 3G/4G may be provided, and it is expected that connectivity will not be continuously available;
- **a range of non-standard sensors** - touch screen, camera (for image capture and subsequent processing), microphone will probably be provided, and others may be, such as a gyroscope, accelerometer and video camera;
- **non-standard outputs** - a small screen, some sound, possibly vibration;
- **other app data** - apps may be able to share data, especially with social media platforms.

Also, usage patterns for these devices are often different from those on standard game platforms such as PCs - games are often played in short bursts (waiting for a meeting, on a bus/train etc.), and gameplay may be interruptible.

In designing and implementing games for mobile devices, these differences combine both to provide challenges, which AI and CI have the potential to solve, and to provide opportunities for novel game concepts based on or supported by AI and CI methods.

IV. Challenges when using AI/CI on mobile devices

Mobile devices introduce a number of constraints to game design:

- Limited CPU and memory (although this is always improving);
- Small screen size limits graphical complexity (although the number of pixels is increasing, the physical size remains a limitation);
- The reality that these devices are typically used when running on a battery further encourages limiting CPU and memory usage beyond what is physically available on these devices;
- However, real-time responses are often called for with mobile devices;
- Connectivity issues must be kept in mind, as devices may lose signal either while out of range of a cell tower or due to a user opting not to pay for wi-fi access at a given location.

It is our thinking that these challenges provide interesting constraints when designing AI/CI-based games, as will be discussed in the next section.

V. Opportunities when using AI/CI on mobile devices

There are some limitations to using mobile devices for gaming (such as small screen size, limited battery life, less powerful processors etc.) but there are also many opportunities for utilizing mobile devices, which are not present on other platforms. We briefly mentioned some of these in the introduction, but in this section we discuss these opportunities in a little more detail.

A. Small screen

Having a smaller screen could be seen as a limitation but it could also be viewed as an opportunity. Having limited graphic capabilities means that the programmers may not have to focus as much on this aspect of the system as would be the case if you were designing a system that had a large screen, high resolution and a powerful graphics processor to assist with the processing required in rendering the screen (although screen resolutions are improving and mobile phone GPUs are becoming more powerful). If a programmer's (or researcher's) skills are in AI/CI, then having a platform which is relatively easy to program could be an advantage as you are able to focus on the AI/CI, without having to be so concerned about the graphics. This may also reduce the need for artists on the project team. Of course, as technology continues to develop, the advantages that we outline here will gradually diminish, and the quality of graphics and art will become a higher priority.

B. Location awareness

A static computer, by its nature, is stationary, and this could be seen as one of its major limitations. A gaming device that is able to be in different geographical locations at different times, opens up a range of possibilities that were not available even a few years ago. It is obvious that having devices that can be moved around offers many opportunities but the focus of this paper is to look at those opportunities from an AI/CI point of view. AI/CI could be utilized in a variety of ways. As the player roams around the game (both physically and within the game world) the AI/CI agent could tailor the game playing experience to meet the expectations of the players.

C. Interaction with other players

Having a capability such as Bluetooth provides opportunities to meet with other players that are in a similar location, but you were not aware that they were there. This would be useful in locations such as a city center but imagine how many people are potentially with a few feet of you at a sporting event or a concert. Once the application had identified potential game 'buddies' the AI/CI could be used to validate the other person's skill level, whether they are actually a match for you to play with etc. A lot of innovation in gameplay is taking place in the mobile market. A couple of examples are Fingle (a bit like the classic ice-breaking game, Twister, but for hands on a tablet - http://fingleforipad.com/) and Swordfight (an example of a Phone-to-Phone Mobile Motion Game [10]).

D. Social media

Mobile platforms already take advantage of the many social platforms that are available. Facebook and Twitter are probably the most well known but there are hundreds, if not thousands, of other platforms that offer users the ability to communicate with one another. Indeed many people, we suspect, use their phone more for texting and updating their status rather than for making phone calls. If a networked, mobile platform is used for game playing, users might want to update their various social networking sites with the games they are playing, their progress, their high scores, who they are playing with etc. This could place a burden on the user who does not have

the time to disseminate all this information, but still wishes it to be known. AI/CI could be used to learn when/what the user wishes to update to social networking sites. For example, a user might always tweet a new high score, but not update their facebook page. Another user might keep a certain person regularly updated about their progress through a game via social media messages aimed at just that user. The challenge is to learn what to update and when, and provide the API (Applications Programming Interface) to the various social media feeds, many of which already exist.

E. AI/CI libraries for use in mobile games

The limited CPU and memory resources typically available on mobile devices suggest the need for AI and CI libraries specifically designed for mobile applications. Two approaches come to mind. Firstly, for applications that require execution on the device itself, stripped down and simplified implementations of common algorithms would be useful. On the other hand, for applications where a client-server model is appropriate, cloud or web service based implementations would be a good solution.

In the academic literature, we could not find any examples of the first kind of any substance. However, there are many examples of small libraries from the open-source community that could provide a good starting point. Many of these examples are implemented in Lua, a scripting-like language with object-oriented capabilities that is commonly used for games. Some examples are Abalhas, which is a PSO implementation in Lua (by Alexandre Erwin Ittner, available at http://ittner.github.com/abelhas/), LuaFuzzy, a fuzzy logic library written in Lua (http://luaforge.net/projects/luafuzzy/) and LuaFann, a fast artificial neural net implementation (http://luaforge.net/projects/luafann). One could perhaps envisage a collection of small, modular library components, written in Lua, and covering these AI and CI technologies, along with others such as evolutionary algorithms, a Lua version of OpenSteer, an A^* implementation, a lightweight rule-based system library perhaps based on CLIPS, and so on.

Of course, this is only one possible development path. For example, web-based development using JavaScript in conjunction with native code, as discussed by Charland et al. [11] is another possibility. There are also existing open-source AI and CI codes, such as JMLR MLOSS (http://jmlr.csail.mit.edu/mloss/), implemented in various languages such as C++, Java or Python. While there may be issues such as size and portability to overcome, much of this could also be utilised : we point out the Lua pathway as one that might work particularly well for mobile games. In the commercial arena, Unity3d provides some AI capabilities, such as path-finding, and AI plug-ins are becoming available.

There are also examples of cloud-based implementations of AI and CI technologies that might be utilised in a client-server approach for mobile games. For example, there is Merelo et al.'s cloud-based evolutionary algorithm [12], Li's cloud-based fuzzy system [13] and Haqquni et al.'s cloud-based neural network system [14]. The Apache Mahout project aims to provide scalable, distributed machine learning (http://mahout.apache.org/), including clustering, classification, collaborative filtering and pattern mining.

VI. What can AI/CI offer for games on mobile devices

A. Procedural Content Generation

Using AI/CI methods for Procedural Content Generation (PCG) in games is an active research area with some notable successes in recent years. Spore is one high-profile example in the commercial arena. We argue that several factors make mobile games well suited for PCG. Firstly, in terms of typical length of play sessions and complexity of typical game environments, mobile games are smaller in scale than many standard video games. This should mean that PCG is achievable with limited computational resources, and could be done locally on the device, without having to offload the task to a server. Second, some of the more interesting AI/CI methods for PCG make use of player preferences, either in real-time or offline. Mobile games with many players would have a ready source for the training data needed to drive these systems.

For example, Interactive Evolutionary Computation (IEC) [15] is a CI technique that could be very well suited for mobile games. Hastings et al. have applied this technique successfully in *Galactic Arms Race* [16]. This game features weapons defined by particle systems controlled by a kind of neural network called a *Compositional Pattern Producing Network*, and these are evolved using *cgNEAT*, a version of *Neuro-Evolution by Augmenting Topologies* [17], where fitness is determined by popularity of player choices in the game. The authors coined the term *collaborative content evolution* to describe this approach.

The mobile game-playing population could provide an ideal environment for collaborative content evolution, with a large pool of players, playing many short game sessions, providing a very large number of judgements to feed into fitness calculations. Crowd-sourcing used in this way should enable content to evolve rapidly, giving game players a constantly novel, changing game experience, guided by the preferences of the players themselves.

B. Personalisation and customisation

Recently, CI techniques are being used to adapt gameplay to optimise player satisfaction in real time. For example, Yannakakis and Hallam reported success with using neural network based user models adjusted in real time to improve player satisfaction in "playware" games [18]. Using the kind of lightweight libraries proposed in Section V-E, this kind of gameplay adaptation and other customisation could be added to mobile games, and neural networks and other machine learning methods have already been proven to be effective for adaptation in other, non-mobile games.

C. Ubiquitous games etc.

The terms *ubiquitous or pervasive* computing have been in use for some time now. As far back as 2002, these terms were also applied to games (see e.g. [19]). There's obviously a considerable overlap between these kinds of games and mobile games — mobile devices provide the means of achieving ubiquity/pervasiveness. A related concept is that of the *augmented reality game*. Here too, modern mobile devices have the camera, audio, and display capabilities to support

augmented reality applications. For ubiquitous games, real-time adaptation with CI algorithms running on the device, could be combined with periodic sychronisation with a cloud-based repository, so that the learned personal profile can be shared across locations and devices. For augmented reality games, either a generic light-weight augmented reality library or perhaps some application specific implementation in the style of Baltes et al. [7], could be used.

VII. An example game: InfiniteWords

InfiniteWords is an example game created to illustrate a design pattern for one kind of "smart mobile game". The idea of the pattern is to combine Procedural Content Generation to create game content, with smart filtering (for example, Collaborative Filtering – see [20] for a recent survey), to produce an infinite supply of high quality content. This idea is similar to collaborative content evolution, except that there is no evolution as such (new puzzles are randomly generated without any direction), and that the filtering can take account of individual player preferences rather than simply overall popularity. The game is similar to a number of commercial mobile word games that are popular at the moment – word-guessing games with images as clues, except that in those games, the puzzles have to be created by hand and provided by the game company.

The components of InfiniteWords are shown in Fig. 1. The game logic resides on a mobile device, along with a puzzle generator (the PCG) and two queues - one for puzzles that the player is yet to play, and one for player ratings for puzzles that the player has played. These queues reduce player waiting time by allowing network tasks to take place during thinking time, and also allow the game to function for a time without network access (until the puzzle queue is exhausted). Two web services are used to enable the game. An image server provides tagged images (the tags are assigned manually), and a "recommendation server" collates player rating information from all players and carries out the smart filtering.

Here is a walk-through of InfiniteWords from a player viewpoint:

1) A puzzle is taken from the "puzzle queue" and presented to the player (see Fig. 2). The puzzle consists of four images that relate to a "target word" that the player tries to guess. Random letters are added to the letters that make up the word, giving a total of eight letters, which are then jumbled up and presented on the orange "clue" tiles.
2) The player selects the tiles in the right sequence to spell out what they think is the target word. As tiles are selected, the letters are moved to the blue "solution" tiles, filling from left to right. The player can delete letters using the "X" button, which removes letters from right to left, returning them to the clue tiles.
3) The player can get "hints" by selecting the "?" button. Each time this is selected, another correct letter is added to the solution tiles, filling from left to right. The final letter of the word is not available via hints.
4) When the correct word is spelled, the puzzle is solved, and a "rating" screen appears. The player has to select a rating between one and five stars to continue. At this point, the rating information, along with data on how long the puzzle took to solve, and how many hints were used, is added to the "rating queue".
5) The next puzzle is presented.
6) The player can choose to skip a puzzle, or return to a starting menu (not shown) at any time (except during rating). In this case, the current puzzle is simply discarded.

The puzzle queue holds a fixed number of puzzles that have either been created by the puzzle generator, or recommended by the recommender. Getting a puzzle requires network access for downloading images and communicating with the image server and/or the recommender. A background task keeps this queue full whenever a network is available. If the queue is empty when the player needs a puzzle, then either the recommender is asked for a puzzle, or a new one is constructed by the puzzle generator on the fly. The choice of which method to use is determined randomly. With a large population of players, the system will ony require each player to rate a newly-generated puzzle very occassionally, to keep up a supply of new puzzles. Players will be playing highly-rated puzzles that match their own preferences for the great majority of the time.

The puzzle generator works as follows:

1) A word is selected randomly from a word list (around 10,000 words).
2) The image server is queried for a list of image IDs of images that relate to the target word.
3) If there are less than four images, the generator fails.
4) Otherwise, four images are selected at random, and the image server is asked to send them.

Note that this is simply a random process : player ratings do not influence the generation of new puzzles. Collaborative filtering takes care of the quality issue.

The rating queue temporarily holds the player's rating information for completed puzzles. A background task sends these ratings to the recommendation server when the network is available, before removing them from the queue. These can be ratings for puzzles retrieved earlier from the recommendation server, or for novel puzzles created by the puzzle generator on the player's mobile device. When a rating for a novel puzzle is sent to the recommendation server, that new puzzle is added to the central database, and becomes available to other players. The effect is that the community of players themselves ensure a consistent supply of novel puzzles is maintained, and player ratings are used to "curate" the puzzle collection. Poor puzzles will get low ratings and will eventually cease being recommended (and can then be removed from the database.)

Note that in this example game, puzzles are simply unstructured "items", and the collaborative filtering algorithm is a memory-based one, exploiting similarities between players. For other kinds of procedurally generated content, a model-based collaborative filtering system could be used, which could enable preferences to be used as part of the generation process.

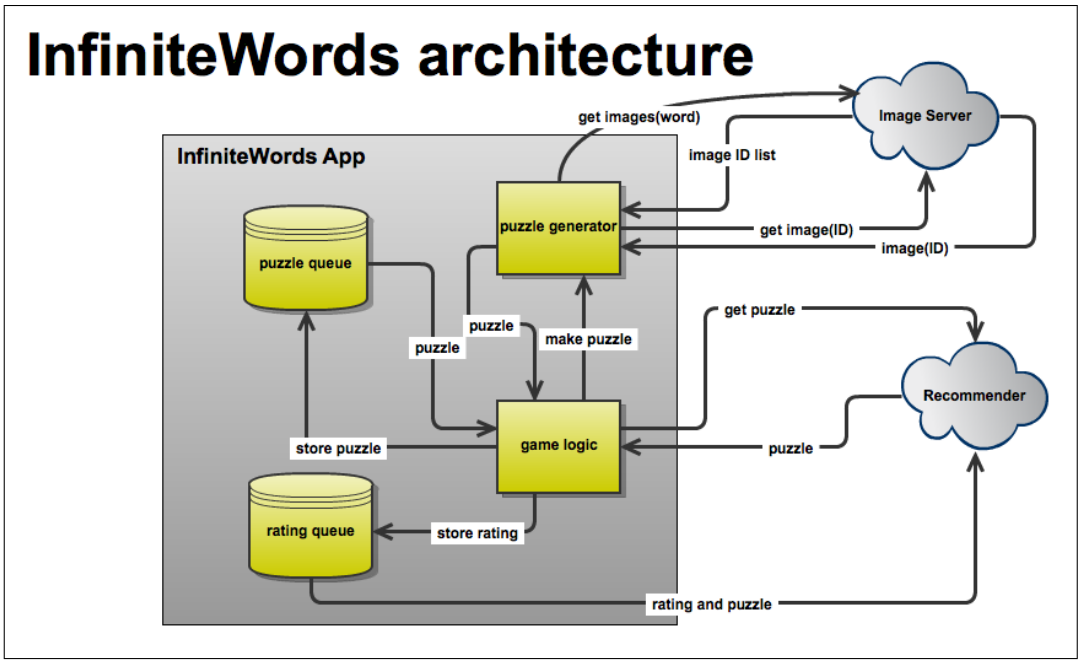

Fig. 1. Architectural diagram of InfiniteWords. The app uses two web services. An image server is used to provide images that relate to some seed word. The app's puzzle generator uses these to construct puzzles. A "recommendation server" is used to provide pre-rated puzzles that suit the player's taste, based on player ratings of puzzles, using collaborative filtering. The app maintains two queues, one for puzzles and one for ratings, so that it can continue to function for a time in the absence of network connectivity.

(a) Puzzle (b) Rating

Fig. 2. InfiniteWords on an iPhone. (a) Shows a puzzle as first presented. The four images relate to a common word, in this case "tribal". The player selects letters from the orange "clue" tiles to move them to the blue "solution" tiles, spelling out what they think is the target word. Hints are available (using the "?" button) - each hint reveals one correct letter. When the player completes the target word, a rating screen becomes available. (b) Shows the rating screen. The player selects a rating between one and five stars. These ratings are used as input to a collaborative filtering algorithm, so that future puzzles are chosen to suit the player's taste.

VIII. Conclusions

Mobile platforms are already widespread and their use is largely for interacting with social media sites and for tweeting. Some people also use them for what they were originally designed for, making phone calls. Game playing is becoming more widespread on these devices, more so on

phones than tablets, with around a third of mobile phone owners reportedly playing mobile games (see, for example http://www.infosolutionsgroup.com/popcapmobile2012.pdf). Computational Intelligence and Artificial Intelligence are not often present in these games, or if present, are unsophisticated. However, there is a window of opportunity where we are able to integrate these technologies into these games, with less of the now usual overhead of having to work with graphic designers, musicians, plot design etc. As mobile platforms develop, the complex, large teams associated with console-based game design are likely to emerge such that it may be more difficult to enter this market. But for the moment there is a great opportunity!

In this short article, we have outlined some of the opportunities and challenges in introducing AI/CI onto mobile platforms. We hope that the research community will take up the many research challenges that exist in this exciting, fast moving area.

ACKNOWLEDGEMENT

Many of the ideas in his paper were first put together by the authors at the Dagstuhl Seminar on Artificial and Computational Intelligence in Games, in May 2012. This paper enhances and builds upon those ideas, which were presented in [21]. We would like to thank the Seminar organisers: Simon Lucas, Michale Mateas, Mike Preuss, Pieter Spronck and Julian Togelius; as well as all the other participants at the seminar for an inspirational experience.

REFERENCES

[1] A. Drachen, C. Thurau, J. Togelius, G. N. Yannakakis, and C. Bauckhage, "Game Data Mining," in *Game Analytics*. Springer London, 2013, pp. 205–253.

[2] K. Curran and C. George, "The Future of Web and Mobile Game Development," *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, vol. 1, no. 1, pp. 25–34, 2012. [Online]. Available: http://iaesjournal.com/online/index.php/IJ-CLOSER/article/view/233

[3] F. Aiolli and C. E. Palazzi, "Enhancing Artificial Intelligence on a Real Mobile Game," *International Journal of Computer Games Technology*, 2009.

[4] A. Jordan, D. Scheftelowitsch, J. Lahni, J. Hartwecker, M. Kuchem, M. Walter-Huber, N. Vortmeier, T. Delbrugger, U. Guler, I. Vatolkin, and M. Preuss, "Beatthebeat music-based procedural content generation in a mobile game," in *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, 2012, pp. 320–327.

[5] C. Xin, "Artificial Intelligence Application in Mobile Phone Serious Game," in *Education Technology and Computer Science, 2009. ETCS '09. First International Workshop on*, vol. 2, march 2009, pp. 1093 –1095.

[6] A. Kruger and R. Malaka, "Artificial Intelligence Goes Mobile," *Applied Artificial Intelligence: An International Journal*, vol. 18, no. 6, pp. 469–476, 2004.

[7] J. Baltes and J. Anderson, "Complex AI on Small Embedded Systems: Humanoid Robotics using Mobile Phones," in *AAAI Spring Symposium Series*, 2010. [Online]. Available: https://www.aaai.org/ocs/index.php/SSS/SSS10/paper/view/1136/1403

[8] C.-U. Lim, R. Baumgarten, and S. Colton, "Evolving behaviour trees for the commercial game DEFCON," in *Applications of Evolutionary Computation*. Springer, 2010, pp. 100–110.

[9] D. Isla, "Managing Complexity in the Halo 2 AI System," in *Proceedings of the Game Developers Conference*, 2005.

[10] Z. Zhang, D. Chu, X. Chen, and T. Moscibroda, "SwordFight: enabling a new class of phone-to-phone action games on commodity phones," in *MobiSys*, 2012, pp. 1–14.

[11] A. Charland and B. Leroux, "Mobile application development: web vs. native," *Commun. ACM*, vol. 54, no. 5, pp. 49–53, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1941487.1941504

[12] J. J. M. Guervós, M. G. Arenas, A. M. Mora, P. A. Castillo, G. Romero, and J. L. J. Laredo, "Cloud-based Evolutionary Algorithms: An algorithmic study," *CoRR*, vol. abs/1105.6205, 2011.

[13] Z. Li, Y. Wang, J. Yu, Y. Zhang, and X. Li, "A Novel Cloud-Based Fuzzy Self-Adaptive Ant Colony System," in *Proceedings of the 2008 Fourth International Conference on Natural Computation - Volume 07*, ser. ICNC '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 460–465. [Online]. Available: http://dx.doi.org/10.1109/ICNC.2008.696

[14] A. Huqqani, X. Li, P. Beran, and E. Schikuta, "N2Cloud: Cloud based neural network simulation application," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, july 2010, pp. 1 –5.

[15] H. Takagi, "Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation," *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1275–1296, 2001.

[16] E. J. Hastings, R. K. Guha, and K. O. Stanley, "Automatic Content Generation in the Galactic Arms Race Video Game," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 4, pp. 245–263, 2009.

[17] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[18] G. Yannakakis and J. Hallam, "Real-Time Game Adaptation for Optimizing Player Satisfaction," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 1, no. 2, pp. 121 –133, june 2009.

[19] S. Bjrk, J. Holopainen, P. Ljungstrand, and R. Mandryk, "Special Issue on Ubiquitous Games," *Personal and Ubiquitous Computing*, vol. 6, pp. 358–361, 2002. [Online]. Available: http://dx.doi.org/10.1007/s007790200040

[20] X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, vol. 2009, p. 19, 2009.

[21] C. Congdon, P. Hingston, and G. Kendall, "Artificial and Computational Intelligence for Games on Mobile Platforms: A Position Paper," in *(To appear) Artificial and Computational Intelligence in Games (Dagstuhl Seminar 12191)*. Dagstuhl Follow-Ups, 2013.