

FPO

Cognitive Gaming

Wei Cai, Yuanfang Chi, and Victor C.M. Leung, *The University of British Columbia*

As intelligent networked computing becomes pervasive, an emerging trend is to apply a cognitive computing paradigm to video game design and development. This article describes the concept of cognitive gaming and discusses its enabling architecture.

Recent hot topics in research, including the Internet of Things, cloud computing, and virtual reality (VR), can all benefit from powerful cognitive systems able to perceive human and environmental variables, make intelligent decisions, and give feedback.¹ Video games are a virtualized and enhanced reflection of reality. Focused on providing virtual experiences to human players, gaming systems are emerging as the earliest adopters of cognitive capability. Furthermore, games are the simplest platforms for demonstrating intelligent technologies.

In fact, public interest in artificial intelligence (AI) was raised by a Go game between Google's

AlphaGo program and Lee Sedol, a South Korean professional Go player ranked as a 9 dan. AlphaGo's unexpected win made it famous worldwide. The program builds neural networks that guide the optimal solution search procedure, which is a learning approach through dataset training rather than predetermined strategies specified by the designer. Recently, Master, the upgraded version of AlphaGo, won 60 straight games by defeating a litany of world champions, which further proved AI's advantage.

Here, we investigate the design and development of cognitive gaming, survey the current application of cognitive computing in the game industry, and discuss its future.

Cognitive Gaming Definition

The term “cognitive gaming” carries different meanings in diverse contexts. For example, it can be used to refer to games that test players’ brains or challenge their cognitive abilities. It is even adopted as the name of a professional e-sports team based in the US. In this article, we define cognitive gaming as the integration of the cognitive computing paradigm¹ in video games. Figure 1 depicts the architectural framework of cognitive gaming, which consists of three fundamental elements:

- *input*—the interface that perceives detailed environmental data and players’ information in real time and delivers this data to a cognitive analytical engine as quickly as possible;
- *cognitive engine*—the analytical component that consists of data mining, decision making, and resource integration components; it cooperatively leverages rich resources in the cloud and applies reasoning, inducting, and learning methodologies to support cognitive capacity; and
- *output*—the interface that provides various representations for players, including the video display, vision mixture, vibration, and audio.

As a software system that provides real-time interaction with players, the system design should be based on low-latency principles.² The application of cognitive gaming can be classified into two categories: cognitive game content generation and cognitive game system optimization. We discuss these two directions in the following sections.

Cognitive Game Content Generation

Providing intelligent responses to players’ commands is the fundamental requirement of a gaming system. Under this circumstance, computer-based AI is a frequently mentioned principle in traditional game design, even though most implementations are simple scripts that let a nonplayer character (NPC) provide predefined feedback to players’ input. Recently, advanced AI

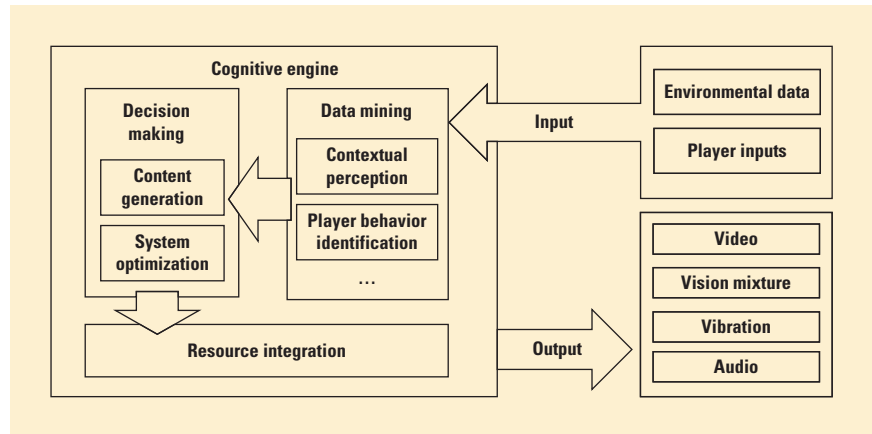


Figure 1. The architectural framework of cognitive gaming. This architecture has three fundamental elements: input, a cognitive engine, and output.

algorithms have started to provide sophisticated responses. For instance, AlphaGo represents the most cutting-edge decision making in chess placement in a Go game.

In this article, we extend cognitive computing to the broader aspect of procedural generation,³ in which the gaming system can intelligently generate dynamic game content.⁴ In this approach, the game engine predefines only the rules and elements that generate future game content, which will be subject to players’ behaviors and environments. For example, the simplest and earliest approach is based on pseudorandom number generation, which is used by *Elite*, a space exploration game, to create a very large universe. It falls in the category of game-space generation. Another example is rhythm-based level generation, which generates different levels for 2D platformers (such as *Super Mario Bros.*) based on the pattern of a player’s hand movements when playing the game.³ Furthermore, variations in scenes, weapons, characters, obstacles, and other design features could be generated using a genetic algorithm and verified by a game-playing agent.⁴ The purpose is to give gamers a more immersive experience when they are engaged in a gaming session.

On top of traditional approaches, cognitive gaming leverages richer sources of content and explores environmental data from a deeper gradation in real time. These approaches will be applied in all perspectives, including to augmented reality (AR), sensors, and players’ behavior. For instance, a cognitive version of *Super Mario Bros.* might mix the Mushroom Kingdom with players’ vision, so that Mario might jump from one edge to another surface in reality.

Instead of preset bricks, coins, mushrooms, and tortoises, these items might show up according to players' surrounding environmental information, which can be deducted from multiple sources. An example could be the traffic information returned from the Bing Maps API. A cognitive game system could create more enemies when players are in crowded traffic. Also, Mario's speed might also be subjected to the movement of a player's terminal. Given the playing scenario of a fast-moving train, the accelerometer in the gaming device might trigger a high-speed mode for Mario that enables some hidden warp zones. Furthermore, players' social network status, clicks, and swipes during gaming sessions and the rhythm of Mario's jumps could also be potential data for content generation. The game scene might be brighter if a player is identified as being in a negative mood, whereas his or her fast pace might result in a burst of enemy attacks.

Augmented Reality

An AR game can be considered an advanced version of VR.⁵ The design principle for VR games is to track players' eyes and render corresponding game images, thus making the depth of field more realistic. In contrast, AR games capture videos from players' field of vision (or simply use mobile devices' back camera to record videos), analyze the videos' content, and then dynamically generate virtual content to be displayed to players' eyes. A well-designed AR system can mix virtual items with reality, leading players to see an indistinguishable vision of their world.

An AR game is a typical application of cognitive computing. Industry giants have been investigating the possibility of AR games. For example, the *world* Android game on Google Project Tango lets players decorate their space with fantastical objects and then see how they interact in silly, playful ways (www.funomena.com/world). Similarly, the *RoboRaid* game on Microsoft HoloLens gives players a mixed reality, first-person shooting experience, in which target enemies are coming at the player from every possible direction in his or her physical environment (bit.ly/2q8gqre). Certainly, real-time video pattern recognition is the most challenging issue to be addressed in AR games. State-of-the-art computer

vision techniques still have immature image analysis accuracy, not to mention high costs and long delays in data processing. Research projects, such as Google Cloud Vision and Microsoft Cognitive Services, are looking for breakthroughs in this area with help from machine learning technologies.

Sensors

Another cognitive gaming approach leverages all kinds of available sensors to harvest environmental data from around game players and use it as the input for content generation.⁶ A classic case is to locate players with GPS and adopt the location information as an important parameter for gaming content. For example, *PokemonGo* (www.pokemongo.com), a recently popular mobile game, produces different pokemons (pocket monsters) such as magikarp, a special carp fish, according to players' locations. A magikarp will be available only when the gamer approaches a waterfront, such as a lakeside or harbor side. A similar concept was implemented in an earlier Android game, *Ingress* (www.ingress.com). Besides GPS, the trend in cognitive gaming is to extend the application of information usage to all kinds of sensors in mobile devices, including gyroscopes and accelerometers. In fact, these sensors have been adopted as input equipment for a long time—for example, in the Wii game console produced by Nintendo. However, in cognitive gaming, these sensory data should target content generation. For example, when a player is traveling in a vehicle, the acceleration his or her mobile device senses can be used as a parameter for game scenario creation.

Player Behavior

Collecting players' information for game content generation will certainly become the trend in cognitive gaming. This is an extension of the personalization approach for targeted advertising, which tracks consumers' behavior and delivers appropriate and accurate advertisements to them. The Internet's rich information allows a machine to quickly understand game players' characteristics. Game systems can analyze gamers' behavior models, such as selection of game genres, length of gaming sessions, strategy during the sessions,⁷ and even the key strike frequency and mouse movement traces, to deduce

a particular gamer's characteristics and preferences, then generate content to meet his or her needs. If players connect their accounts to their social networks, the gaming system can even collect and analyze their personal profiles and social feeds to give them a unique, personalized gaming experience. The game rating system can also be implemented in this paradigm: teen players might receive different content than adults, such as that with restrictions on violence, blood and gore, sexual content, or strong language. Furthermore, cognitive computing is a potential solution for difficulty settings, the most critical challenge in game design. An excellent game design should post appropriately difficult tasks, setting solvable but challenging targets to engage players step by step. Existing practice relies on experienced game designers, which isn't suitable for different levels of players. In the future, cognitive computing will take over this task and provide fine-tuning services for game difficulties, thus enabling customized gaming experiences for different players with different skill levels.

Opportunities and Issues

Building cognitive services specifically for game content is never easy because it involves complicated mathematic models and requires large amounts of data as training sets for machine learning algorithms. Leading enterprises running large machine learning clusters are way ahead in this new playing field. For instance, IBM Japan has announced *Sword Art Online* (ibm.co/2rEu6Ly), which will utilize the IBM Watson cloud to enable cognitive game design.

Does this mean that small game companies, especially startups, have no chance in this evolution? Not really. IT franchises would like developers to be attached to their platforms and create rich content for their marketplace, so that they can continuously receive high revenue from infrastructure services. Hence, they share their core services through customized APIs. This opens up opportunities for startups to leverage these rich resources with a pay-as-you-go model. From the developers' perspective, integrating cognitive computing services from public cloud providers is a shortcut.

Figure 2 illustrates how a game development leverages several services from the Google cloud platform and the IBM Bluemix cloud.

For instance, the player's terminal can capture video and voices, along with traditional input from players' keyboard and mouse actions. In this case, the environmental videos can be analyzed by pattern recognition algorithms, such as the Google Cloud Vision service, to extract context information. For voice, the cloud might predict players' emotions through a tone analyzer, while the verbal content can be converted to text-based messages for further semantic analysis, which will be performed with natural language processing algorithms.

Considering that more and more games are operated across multiple countries, cloud translation services for text information are also important elements in future game development. After the cognitive content generator gathers such information, dynamic content generation can be conducted with the help of AI algorithms and external resources, including historic gaming data, related multimedia content from the Internet, players' social network updates, and so on. As the figure illustrates, cognitive services from the public cloud, such as IBM Watson Conversation, can also be imported as external resources. An illustrative scenario will be creating an NPC to imitate a real person and chat with players in game scenes. As the output of the cognitive content generator, the response would be in the form of scene renderings, game element behaviors, and varied dynamic design elements. As depicted in Figure 2, text responses can be transformed into voice responses with help from the IBM text-to-speech service.

However, adopting public cloud services also introduces uncertainty to the gaming system because the quality of service (QoS) from these clouds is out of developers' control; in particular, today's service level agreements (SLAs) are still coarse-grained and hard to audit. Hence, how to guarantee QoS in accessing these cloud services is still a critical challenge. Moreover, how to customize existing cloud services for specific gaming needs is an open issue. For instance, regarding the chatting NPC example described in Figure 2, it is still difficult to create different NPCs with different story backgrounds in different narratives, because it is challenging to create separate artificial neural networks for different NPCs, which must be fed with distinct knowledge data and profiles.

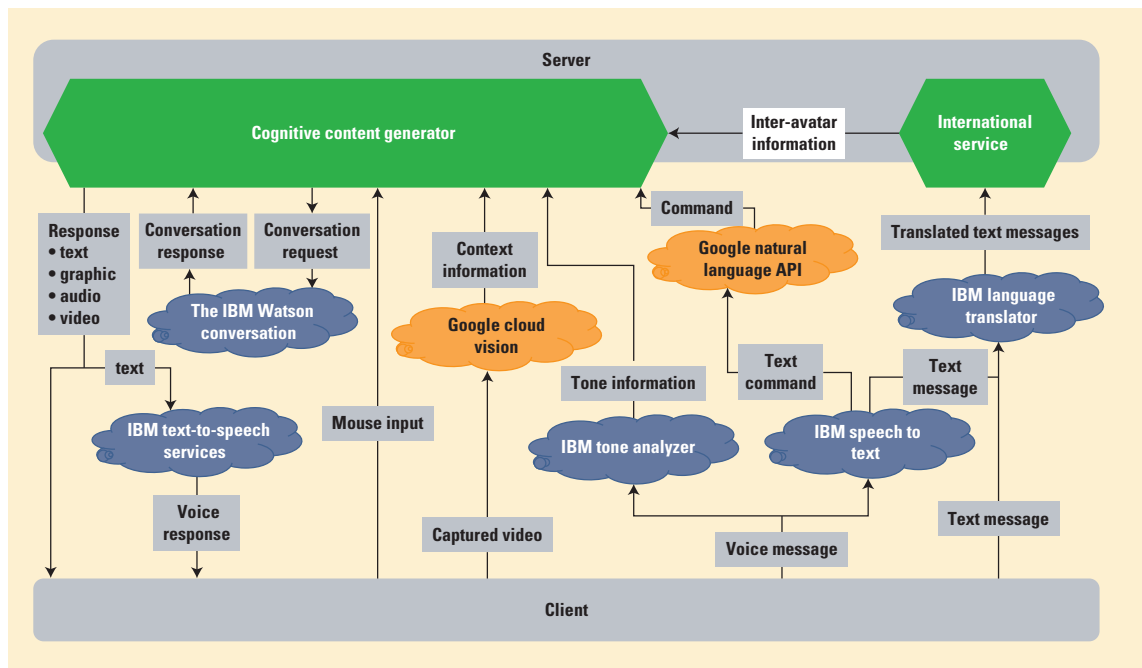


Figure 2. A cognitive gaming system integrated with public cloud services. Raw inputs (video and voice) are processed by cloud services before reaching the cognitive content generator.

Cognitive Game System Optimization

Optimization is a never-ending task for quality-of-experience (QoE)-critical systems such as video games. By optimizing the scenario rendering and computational workload distribution, the cognitive gaming system is expected to deliver a better QoE with the same hardware foundations. As a dynamic human-computer interaction system, different players' distinct behaviors and performances in different game scenarios pose a stringent requirement in game system design and optimization. These issues become even more complicated in multiplayer games because the variety of environmental (for example, network or terminal) parameters brings in more uncertainty. The self-adaption feature of cognitive computing makes it a powerful methodology to address these issues.

Scenario Rendering

Modern video games pursue realistic 3D game scenarios and avatar rendering, imposing a high demand on computational resources. Different from other applications, scenario rendering for video games is highly latency-sensitive. The experience of most game players will suffer from any response delay higher than 200 ms,⁸ while

VR games have an even tighter restriction on the delay. Although the computation performance of PCs and mobile devices has improved greatly in recent years, it is no match for highly sophisticated graphic algorithms that demand abundant computational power, especially in mobile devices with limited computational resources. This becomes a critical issue in game development and deployment. Consequently, many advanced games provide video options for gamers, enabling them to customize the quality of scenario rendering according to their hardware capacity. However, these manual settings with a large number of video options confuse some players, whereas others do not even realize that such functions exist. Furthermore, online and cross-platform games also limit the usage of these settings due to the variety in network and terminal hardware.

To this end, cognitive computing can be adopted to optimize scenario rendering. A leading research work on this topic is a comprehensive study of cloud gaming that hosts game engines in cloud servers, renders game scenarios remotely in the cloud, and streams the gaming video to players through the Internet.⁹

Under this circumstance, network quality becomes the key element that determines system performance, given that the bottleneck is the real-time video transmission. Therefore, the early-stage work focuses on adapting video to network parameters—for instance, one scheme monitors network QoS parameters and adapts scenario rendering parameters such as scenario depth, scenario details, and video quality to improve the perceived gaming experience.¹⁰ Another idea perceives avatars' interaction with game scenarios and determines different importance levels of objects to be rendered in the scene.¹¹ Under weak network or hardware conditions, the proposed system would render only critical object sets in game scenes to minimize the workload for scenario rendering. Both of these works can be easily extended to conventional video games. More interestingly, players' behavior can also be included as a source of parameters in determining optimal rendering solutions. For example, foveated rendering tracks players' eyes and selectively renders the important areas around their gaze.¹² By reducing the rendering quality of vision edges, this approach significantly reduces the target frame rate and increases resolution compared to traditional real-time renderers.

Workload Distribution

The cognitive computing paradigm can also be applied to dynamic workload distribution in gaming systems. Here, workload refers to all kinds of computational needs in game programs. Along with the development of game content and forms, especially those massive multiplayer online role-playing games (MMORPGs), the downloading and rendering of game elements are no longer one-time operations on players' terminals. Instead, progressive downloading and computational offloading become the trend. In progressive downloading implemented by Utomik (utomik.com), the terminal could start a gaming session with a small proportion of the game resources downloaded from the cloud server; the rest of the game then downloads while the game progresses. Progressive downloading requires a cognitive approach to predict a player's progress and determine the data to be delivered in advance. It potentially reduces downstream transmissions, given that there are many branches in narratives.

On the other hand, computational offloading leverages the rich computational resources in the cloud to execute game logic on cloud servers, and then transmits the results to the terminal via the network. In this context, cognitive computing means intelligent decisions in dynamic offloading. CloneCloud, a pioneering study on this topic, enables the elastic execution of general applications between mobile devices and the cloud.¹³ It proposes a framework that combines static program analysis with dynamic program profiling to balance workload and optimize execution time or energy. The future game system should extend the modality of CloneCloud to perceive and predict all environmental parameters involved, and determine players' gaming progress and avatar behavior to perform cognitive offloading for system performance optimization.

Opportunities and Issues

Cognitive scenario rendering and workload distribution should be built on the concept of component-based gaming architecture, given that scenario rendering needs to separate different objects for rendering, and workload distribution requires the workload itself to be migrated from a terminal to the server or other terminals. An obvious challenge for such an architecture is the decomposition complexity or, to be more specific, the decomposition level (for example, data level, task level, or function level). The decomposition level defines the frequency at which components interact with each other and thus the rate of data exchange between them. Because components could be remotely executed, a high data exchange rate (high decomposition level) between remote components could be detrimental to both system performance and communication cost. As the decomposition level varies with game genre, how to find the appropriate level of decomposition remains the biggest challenge.

As Figure 3 depicts, recent work at the University of British Columbia has designed and implemented a testbed for decomposed gaming, following the cognitive computing paradigm.¹⁴ In such a system, video games are decomposed into software components that are cognitive to players' behavior and execution environments in both the terminal and the cloud. The authors developed several game prototypes over the proposed testbed for cognitive workload distribution.

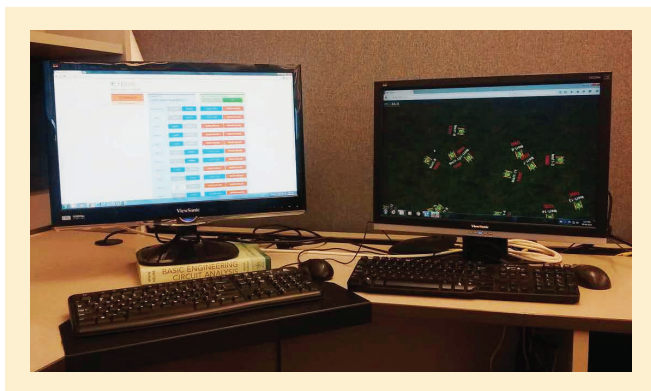



Figure 3. A testbed for decomposed cognitive gaming developed at the University of British Columbia. The left screen displays the workload distribution of different components for the tank game displayed in the right screen.

Preliminary experimental results demonstrated that well-designed cognitive algorithms can help improve the performance of a gaming system—for instance, increase the frame-per-second value in game execution.

The game program is a perfect playground for cognitive computing. Besides the conventional application of using cognitive science as the basis of NPC intelligence, we envision that game content will be partly or entirely generated by a cognitive engine in the near future, involving functions such as AR integration, sensory data usage, and identification of players' behaviors. Also, the adaptive nature of cognitive computing makes it an attractive solution for video games. Cognitive scenario rendering and workload distribution will provide significant optimization for those systems suffering from a varying execution environment. 

References

1. D.S. Modha et al., "Cognitive Computing," *Comm. ACM*, vol. 54, Aug. 2011, pp. 62–71.
2. M. Claypool and K. Claypool, "Latency and Player Actions in Online Games," *Comm. ACM*, vol. 49, no. 11, 2006, pp. 40–45.
3. M. Hendriks et al., "Procedural Content Generation for Games: A Survey," *ACM Trans. Multimedia Computing, Communications, and Applications*, vol. 9, no. 1, 2013, article no. 1.
4. L.T. Pereira et al., "Learning to Speed Up Evolutionary Content Generation in Physics-Based Puzzle Games," *Proc. 28th IEEE Int'l Conf. Tools with Artificial Intelligence*, 2016, pp. 901–907.
5. A. Phillips, "Games in AR: Types and Technologies," *Proc. IEEE Int'l Symp. Mixed and Augmented Reality—Arts, Media, and Humanities*, 2009, pp. 9–10.
6. R. Shea et al., "Location-Based Augmented Reality with Pervasive Smartphone Sensors: Inside and Beyond Pokemon Go!," *IEEE Access*, to appear, 2017; doi:10.1109/ACCESS.2017.2696953.
7. W. Cai et al., "Quality of Experience Optimization for Cloud Gaming System with Ad-Hoc Cloudlet Assistance," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 25, no. 12, 2015, pp. 1–14.
8. M. Jarschel et al., "Gaming in the Clouds: QoE and the Users Perspective," *Mathematical and Computer Modelling*, vol. 57, no. 11, 2013, pp. 2883–2894.
9. W. Cai et al., "The Future of Cloud Gaming," *Proc. IEEE*, vol. 104, no. 4, 2016, pp. 687–691.
10. S. Wang and S. Dey, "Rendering Adaptation to Address Communication and Computation Constraints in Cloud Mobile Gaming," *Proc. IEEE Global Telecommunications Conf.*, 2010; doi:10.1109/GLOCOM.2010.5684144.
11. I.S. Mohammadi, M.R. Hashemi, and M. Ghanbary, "An Object-Based Framework for Cloud Gaming using Player's Visual Attention," *Proc. IEEE Int'l Conf. Multimedia & Expo Workshops*, 2015; doi:10.1109/ICMEW.2015.7169781.
12. A. Patney et al., "Towards Foveated Rendering for Gaze-Tracked Virtual Reality," *J. ACM Trans. Graphics*, vol. 35, no. 6, 2016, article no. 179.
13. B.G. Chun et al., "CloneCloud: Elastic Execution between Mobile Device and Cloud," *Proc. 6th Conf. Computer Systems (EuroSys)*, 2011, pp. 301–314.
14. W. Cai et al., "MCG Test-Bed: An Experimental Test-Bed for Mobile Cloud Gaming," *Proc. 2nd Workshop Mobile Gaming (MobiGames)*, 2015, pp. 25–30.

Wei Cai is a postdoctoral research fellow in the Department of Electrical and Computer Engineering at the University of British Columbia (UBC), Canada. His research areas include cloud computing, cognitive systems, Internet of Things, finance technology, and software engineering. Cai received a PhD in electrical and computer engineering from UBC. He is a member of IEEE. Contact him at weicai@ece.ubc.ca.

Yuanfang Chi is an application engineer at Oracle. Her research areas include cognitive computing, cloud computing, and real-time software architecture. Chi received a master's of applied science in electrical and computer engineering from the University of British Columbia, Canada. She is a student member of IEEE. Contact her at yuanchi@ece.ubc.ca.

Victor C.M. Leung is a professor of electrical and computer engineering and holder of the TELUS Mobility Research Chair at the University of British Columbia (UBC), Canada. His research interests are in the broad areas of wireless networks and mobile systems. Leung received a PhD in electrical engineering from UBC. He is a fellow of IEEE, the Royal Society of Canada, the Canadian Academy of Engineering, and

the Engineering Institute of Canada. Contact him at vleung@ece.ubc.ca.



Read your subscriptions through the
myCS publications portal at

<http://mycs.computer.org>