IJDI-ERET

INTERNATIONAL JOURNAL OF DARSHAN INSTITUTE ON ENGINEERING RESEARCH AND EMERGING TECHNOLOGIES
Vol. 11, No. 1, 2022
www.ijdieret.in

(Research Article)

Challenges in Educational Game Development

A. Ghosh[1*]

[1*]*Department of Electrical Engineering and Computer Science, Florida Atlantic University Boca Raton, Florida, USA*

Abstract

Games in the twenty-first century have the potential to be important educational tools. Today's kids are easily distracted, and game-based learning is the panacea because it has the potential to fully immerse them in the subject and curriculum. Not only can game-based learning bring a course to life by engaging and motivating students, but it may also serve as a platform for critical thinking, creativity, quick feedback, and collaboration. Different learning styles of pupils are one of the most difficult obstacles in education, yet game-based learning can easily overcome this. Different genres of games include action, adventure, fighting, puzzles, role-playing, simulation, sports, and strategy. Game designers may be able to choose the genre that is best suited for effective learning. Even with all the benefits of game-based learning, some obstacles remain, such as teachers' resistance to change or poor educational game design. Some teachers may be intimidated by students who are often far ahead of them in their use of technology. In this aspect, there is a significant conceptual generation gap, and planning, developing, and implementing games in curricula can be costly. While some games can be repurposed for educational purposes, many cannot be repurposed to satisfy student expectations. Because of the large appeal of games, many educators are introducing a wide variety of concepts through gaming. By teaching students through educational games, teachers can engage students, and students have fun while learning. However, from a large number of game engines and frameworks dedicated to educational game development, determining which tool to employ may be problematic. This literature review identifies the research on game development, the challenges from educators' perspectives, and the possible technical barriers developers must overcome to develop truly engaging educational materials through gaming.

Keywords: Computer Science Education, Game Development Platforms, Serious Games, Game Engine

1. Introduction

The power of computer games to captivate and engage players for a specific purpose such as to develop new knowledge or skills (Corti, 2006) and foster greater engagement within an educational setting has the potential to yield higher academic achievement (Shute et al., 2009). Players are immersed in virtual worlds filled with stunning graphics, storylines, sound, and video. McKee (1992) and Billen (1993) argue that games affect cognitive functions, and motivation, and remove players from the "real world". Games motivate users intrinsically by stimulating curiosity (Thomas and Macredie, 1994). This may be due to the presence of challenges and elements of fantasy (Malone 1980, 1981a, b), novelty, and complexity (Carroll, 1982; Malone, 1984; Malone and Lepper, 1987; Rivers, 1990).

Learning that is fun appears to be more effective (Lepper and Cordova, 1992). Also, Quinn (1994, 1997) argues that the combination of fun elements with aspects of the instructional design includes motivational learning and interactive components for games to benefit educational practice. According to Malone (1981a, b), three elements (fantasy, curiosity, and challenge) contribute to the fun in games.

Though the research on educational games throughout the world is continuously growing, the integration of games in teaching is still somewhat an unexplored area of study and has received limited attention in educational research (Girard et al., 2013; Hanghøj et al., 2011; Ke, 2009; Meyer et al., 2011; Tzuo et al., 2012; Watson et al., 2011). Hanghøj et al. (2011) observed that many of the studies provide limited descriptions of teachers' pedagogical activities, such as the pedagogical choices and considerations that teachers make when they teach with games. Hanghøj et al. (2011) argue that research on games in education has mostly sought to either measure the learning outcomes of game-based learning or to 'identify the inherent learning potential of game designs.

In this paper, we present a review of game development

[*]*Corresponding Author: e-mail: aghosh2016@fau.edu, Tel-+1-3612280612*
ISSN 2320-7590 (Print) 2583-3863 (Online)

platforms that educators use as teaching tools. This article blends research on game development challenges from the educator's perspective and the developer's perspective with the probable solutions to overcome technical barriers to educational game development. This literature review can benefit both educators and developers for the future development of teaching materials that engage students in education through gaming.

2. Comparison of Different Gaming Platforms

Games, in general, can be classified into seven categories: adventure, role-playing games, shooting games, simulation, puzzles, strategy, and sports games (Rugelj, 2015). Games in education also include quiz games, word games, puzzle games with branching scenarios, various simulations with tasks, simulation environments, and role-playing games (Horton, 2012). There are many types of software tools for developing computer games, so it is a challenging task to choose the appropriate game engine for educational games. Choosing the correct tool presents many dilemmas because they all have strengths as well as weaknesses. Some tools are fast for development, some may have performance issues, some require programming skills, and some come with a developer-friendly interface (Peng, 2016). This paper evaluates and briefly describes different popular game development tools used in education.

2.1 PC-based Games: One of the benefits of using PC-based games is their ability to present a scenario with many interdependent variables in a manner that is accessible to novices. Most of these games can be played as both single and multiplayer as well as be compatible with multiple platforms (desktop, mobile, and play station). For PC-based games, each student needs access to a computer, and probably the most significant challenge with those games is finding course content-related game assets including everything that can go into a game, including 3D models, sprites, sound effects, music, code snippets and modules, and even complete projects that can be used by a game engine. Some game engines provide free asset stores to download, but it is difficult to find all assets required for developing the game. In that case, developers need to create their assets with the help of Photoshop or Maya, import the models, and program them.

Various PC-based game engines are available in the market. Among them, Unity is one of the most popular and widely used game engines. This tool allows the deployment of games to a wide variety of platforms such as mobile phones, consoles, and iPad. Unity has fast development speeds. It follows agile game creation and facilitates quick prototyping and continuous releases. There are rich resources of assets that can be imported easily and support most image, audio, video, and text formats. It has an excellent integrated level editor with supporting JavaScript and C# for scripting. It offers superior support for debugging processes at runtime. Large Unity 3D communities and marketplaces offer built components for sound, assets for physics, rendering, controls, etc. Multi-threading, collections, input/output (I/O), and expressive language integrated query (LINQ) library functions using MonoDevelop are used as script hosts.

Unity forces the developers to start from scratch with each game as it doesn't allow any template. Unity graphics are lagging compared to other engines like Unreal Engine 4. The fact that most of the game engines including Unity and Unreal Engine 4 have expensive licenses can be a limitation for freelance developers and small development groups in the educational field. As it needs a large memory, this may cause out-of-memory (OOM) errors in mobile devices and debugging issues. As no source codes are provided, performance issues are hard to find, address and fix. It is expensive if someone needs to use entire features (render textures, stencil buffer support, etc.). There are additional costs to purchase Mobile Pro licenses for effective deployment. The outdated version of MonoDevelop restricts the game to a certain size and constant struggle with garbage collection (GC). Unity has no support for HTML5 for now. Switching built targets requires re-importing everything and is time-consuming when working on a multiplatform game.

2.2 Web-based Games: Relatively quick development cycles have made browser-based games the platform of choice among educators hoping to educate and influence a wide audience. Many of these games can be used to introduce a problem or begin a discussion in courses that cover social issues, current events, and activism. Though usually not as advanced as PC-based games, they are easy to program and learn. System requirements are likewise minimal, requiring only a modern browser. Developers can create web-based games in two ways – directly using HTML5 or integrating animations and videos into the web using the adobe flash multimedia platform.

HTML5 works directly in all browsers (for example, Google Chrome, Mozilla Firefox, Opera, etc.) without any additional plugins for audio, video, and animation. It requires easily readable source code for JavaScript to run the game on any device like a computer, smartphone, or tablet which supports HTML5. HTML5 code writing is relatively easy to learn. Users can play the games without installation. HTML5 allows advanced features such as multiplayer, GPS, camera, and accelerometer in modern devices with different resolutions, screen sizes, aspect ratios, and guidelines. Users have complete control of the devices and their screen space.

JavaScript, the scripting language of HTML5, lacks many features like inheritance, interfaces, and member access via public, private, protected, and custom namespaces. These features are ideal for gaming. In the HTML5 gaming platform, asset integration and the use of an integrated development environment (IDE) are difficult. For mobile game development, there is partial support for the Web Graphics Library (WebGL) for rendering 3D graphics. Whenever a feature is changed or added in iOS, Android, or Windows, the

HTML5 development tool must reflect or consider the modification and make the required alterations in the code. As developers must write code for each platform, sometimes it takes too much time to complete the gaming application.

On the other hand, Adobe Flash is a tool for developing animations over the web. It is also well suited for video games. Adobe Flash is divided into two main parts. Designing animations can be done with drag and drop entirely through an integrated development environment (IDE). Flash programming is done with the Action Script programming language. Flash files are compatible with almost all web browsers using OS-independent flash plugins. Flash games' file sizes are smaller than those of games developed on other platforms. Flash games can talk to the server seamlessly. Flash programming is easier to learn, and developers can start creating games in a very short time compared to other languages; perhaps this is one of the most attractive reasons for choosing Flash.

Flash plugins have very weak support for mobile devices; iOS does not support Flash. It has low efficiency on mobile devices. While a large proportion of Web browsers have the plug-in pre-installed, Flash does require a plug-in with low performance because of its small size. Flash underperforms in the speed of code execution and graphics rendering. Flash doesn't provide native support for real 3D engines or any sort of texture mapping. One of the most important and frequently used buttons on a Web browser is the back button. But a Flash site typically removes that functionality. Flash increases the time it takes the homepage to load which may discourage the users. Audio files embedded in Flash increase download time.

Table 1. Summary of the comparison for different gaming platforms explained above.

			Web-based Gaming Platform		PC-based Gaming Platform
			HTML5	Flash	Unity
Audiovisual fidelity	Rendering	Texturing	Basic	Basic, Bump mapping	Basic, Bump mapping, Procedural
		Lighting	Per Pixel	Per-vertex, Per Pixel	Per-vertex, Per Pixel
		Shadows	Yes	Yes	Projected planar
		Special Effects	No	Yes	Environmental Mapping, Particle Systems, Bill Boarding, Lens Flares
	Animation		Yes	Yes	Forward Kinematics Keyframe Animation, Skeletal Animation Morphing, Animation Blending
	Sound		2D Sound, 3D Sound, Streaming Sound	2D Sound, 3D Sound, Streaming Sound	2D Sound, 3D Sound, Streaming Sound
	Virtual Environment	AR/VR/MR	Basic	Medium	High
Functional fidelity	Scripting	Script	Yes	Yes	Yes
		Object Model	Yes	Yes	No
	Supported AI Techniques	Collision Detection	Yes	Yes	Yes
		Path Finding	Yes	Yes	Yes
		Decision Making	No	No	Yes
	Physics	Basic Physics	Yes	Yes	Yes
		Rigid Body	No	Yes	Yes
		Vehicle Dynamics	No	No	Yes
Composability	Import/Export Content	CAD platforms supported	No	3ds Max, Maya	3ds Max, Maya
		Import/Export Limitations	Yes	No	No
		Content Availability	High	Medium	Medium

	Developer Toolkits	SDK/GDK	Yes	No	No
Accessibility	Learning Curve		High	Medium	Medium
	Docs and Support	Docs Quality	Docs and Tutorials	Docs and Tutorials	Docs and Tutorials
		Technical Support	No	Yes	Yes
		Community Support	No	Yes	Yes
	Licensing		Free	Free	Indie and Pro Version
	Cost		Free	Depends on vendor	149 and 1099 Euros respectively
Networking and Heterogeneity	Networking	Client-Server	Yes	Yes	Yes
		Peer-to-Peer	No	Yes	No
	Heterogenic	Multiplatform	Yes	iPhone does not support	Yes
		Multiplayer	Yes	Yes	Yes

3. Challenges in Educational Game Design and Probable Solutions

Game development presents its own set of obstacles, but Game-Based Learning also faces the following implementation issues:

- Are there any computer labs where students can play games?
- Are they configured correctly?
- Are they available for the long hours required for gameplay?
- Is the necessary equipment, such as headphones, speakers, and special consoles, available?
- Is there any technical or game-play assistance accessible for the game?
- Is gaming part of the curriculum or an afterthought? (Oblinger, 2006)

3.1 The Educator Perspective:

3.1.1 Identifying Correct Challenges: Games are supposed to be challenging. This requires a clear understanding of the difference between good challenges and frustrating usability problems. Most productivity tools struggle to find a balance between providing a powerful enough set of tools for technologically savvy educators, and a gradual enough learning curve for the novices. However, no designer consciously chooses to make a productivity tool more difficult. After all, doing so would go against the main design goal of making users productive. The ideal tool enables users to experience challenges only in terms of expressing their creativity. For games, learning the goals, strategies, and tactics to succeed is part of the fun. Unfortunately, it is not always clear which tasks should be intuitive (i.e., easy to use) and which ones should be challenging. Input from users becomes necessary to distinguish good challenges from incomprehensible design.

3.1.2 Addressing Different Skill Levels: Unfortunately, all players don’t have the same skill level in terms of gaming experience or talent. Frequent failure can be a turnoff. A success that comes too easily can also become repetitive. Games must address the problem of meeting all players with the correct level of challenge. Tuning a game to the right challenge level is called “game balancing”.

There are many ways to balance the difficulty of the game. The most obvious way is to let players choose the difficulty themselves. Many games offer the choice of an easy, medium, or hard difficulty level. While this seems like a simple solution, it is not simple to identify exactly how easy the easiest level should be. Players want to win, but they do not want to be patronized. Too easy is boring and too hard is unfair. Either experience can make a person-cease playing. For that, the educators need to create different contents of the game story according to the hardness of the game. Another approach to varying skill levels is to require explicit instruction that helps all users become skilled in the game. For example, before starting a cannon fire game simulation, a demonstration of projectile motion and its formula would be helpful for the user.

3.1.3 Rewarding Players Appropriately: Explicit or slow reinforcement schedules may cause users to lose motivation and quit playing a game. Because playing a game is voluntary, games need to quickly grab the user’s attention and keep them motivated to come back again and again. One way to accomplish this is to reward players for continued play. The rewards may be in the form of points, giving more lives, badges, or boosting power.

3.1.4 Combining Entertaining Game Design with Educational Goals and Curricula: Game-Based Learning must achieve specific learning outcomes to be instructional and beneficial in the classroom. Adapting games to these diverse outcomes necessitates a wide range of difficulty levels, numerous "quests or challenges," and several steps or levels to "win" the game. Most children in today's schools will undoubtedly be digital natives, having grown up with the Internet, cell phones, and easily accessible computers. However, this does not apply to all learners, nor does it reflect each learner's specific talent or gaming expertise. It's difficult to adapt a game to accommodate these changes.

Promoting the concept of game-based learning. It is difficult to persuade academics and, especially, parents to learn. GBL is still associated with gaming or, as some parents see it, a "waste of time." In an educational context where accountability and outcomes are important and needed, the stigma associated with games and play is tough to remove.

3.2 The Developer Perspective:

3.2.1 Balancing Different Skill Levels: A contemporary game may use several input devices (such as keyboards, gamepads, and joysticks) and may include different fields of technical knowledge for advanced physics, they need 3D graphics, artificial intelligence, sound, and complex strategies. Each aspect of the game may require all skills of a programmer and, in many cases, several programmers. The number of programmers required for each module depends on the level of programmers' skills, mostly by the type of game being developed. Developers involved come from different disciplines, which might make effective communication among them challenging.

The developers are often segregated into teams by their specialization (programming team, animation team, etc.). This aids in easy information sharing among people of the same discipline but makes communication between groups more complex.

3.2.2 Choosing Game Engines in Cost-Effective Way: Generally, Third-party software is used by Game developers. Developers use game engines to create games for mobile devices, consoles, and personal computers. The core functionality provided by a game engine typically includes a 2D or 3D graphics rendering, a physics engine, animation, scripting, sound, networking, artificial intelligence streaming, memory management, localization support, threading, and may include video support for cinematics. The process of game development is often economized by adapting the same game engine to create different games, or for easy game porting to multiple platforms. This reduces the costs to a great extent, but some issues are to be considered.

A careful decision should be taken on which engines to use as some engines are best used on certain types of games. Developers should be careful enough not to make their games very much like other games made using the same engine. Third-party components must be carefully evaluated to determine which one is the best choice for the job.

3.2.3 Choosing Right Development Tools: Most commercial computer games are written primarily in basic C, C++, and assembly language. Many games, specifically those with highly complex gameplay mechanics, have hardware restrictions. As such, highly optimized code is essential for these games to run at an acceptable frame rate. Due to this reason, compiled code is typically used for critical performance components, such as visual rendering. Also, almost all PC games use either the Open Graphics Library (OpenGL) Application Programming Interface (API)'s, DirectX, or some wrapper library, to interface with hardware devices. Many web-based games use java as it is cross-platform, doesn't require installation by the user usually, and poses fewer security risks when compared to a downloaded executable program. Java is also a preferred language for mobile-based games. Java Script and Adobe Flash are popular development tools for browser-based games.

As games have grown both in size and complexity, middleware is becoming popular within the industry. Compared to standard lower-level APIs such as DirectX and OpenGL, Middleware provides higher and greater level functionality and larger feature sets, such as skeletal animation. Some middleware also supports platform-independent that would make common conversions much easier, for example, Microsoft Windows to PS4.

3.3 Other Challenges:

3.3.1 Developers' Technical Knowledge: To create a worthwhile instructional game, you must first understand the topic and the logic behind it. It's critical to understand the terms and definitions you employ, as well as the context in which you build the game. In a game, you must be honest, since if anything does not make sense, it can easily backfire. We occasionally come across games that use phrases like discrimination or violation of human rights in situations when there was no such thing. This occurs because players incorrectly associate the game's action with the name. If you want to build a serious game, do your homework beforehand.

3.3.2 Neglected Learning Outcomes: We must begin with the most difficult task: setting learning objectives. 2-3 solid learning goals for a game should be set in the same manner that they would be for a school or university classroom lecture. Consider what you want your players to learn, experience, think about, practice, or discover while playing the game. Then write down your realistic and concrete learning objectives and begin developing your game design around them. Avoid going in the opposite direction; otherwise, you can end up with a fantastic mechanic who you adore, but with outdated knowledge.

3.3.3 Lack of Assessment after Game Development: It is simple to fall in love with the theme, mechanic, and specific elements. We tend to build and improve the game environment, characteristics, and aesthetics without constantly checking whether it still contributes to the main learning goals or just complicates the preparation and implementation. Always return to your initial learning goals when creating an educational game to see if this element contributes to them or significantly improves the experience. If the answer is no, do not overburden the game.

3.3.4 Reality vs Imagination: It's fine to create a game set in a galaxy far, far away, or in the past, but it is critical that we can still relate the problems and challenges of those worlds to our lives on planet Earth in the twenty-first century. Players become agitated when there is no logical link between their actions and decisions and the outcomes of the game. Examine your game for inconsistencies and fix any logical flaws. The last thing you want to hear is, "I enjoyed the game, but this could never happen."

3.3.5 Balancing between Rules and Freedom: You rarely want complete chaos created by a lack of game rules, and your players rarely want to mindlessly obey directions. A successful game should provide you both the freedom to make your own decisions and the constraints you need to keep it tough. If you want to make a game where players must follow instructions and have minimal control over the tale, you should make a storybook, comic book, or scripted theatre production instead of a game.

3.3.6 Enough Challenges: Always bear in mind the individuals for whom the game is being created. What are their areas of expertise? What skills do they have and how advanced are they? Remember "the flow," the perfect spot between challenge and skill where you want your target players to be. When making an instructional game, try to think of several difficulty levels so that you may alter it as needed.

3.3.7 Debriefing is Important: The magic happens during debriefing. After a game experience, the process of debriefing and organizing a group discussion is critical for extracting the learning from the game and applying it in the real world. It is the responsibility of the game masters to guide players through the various stages of experiential learning, such as reflection, conceptualization, and possible future application, behavioral or attitude change. Plan your debriefing questions ahead of time and make sure they align with the learning objectives and that the gameplay allows you to ask them coherently.

Debriefing is the hardest element of any learning process, and you can never be completely prepared for it. We recommend that you prepare your questions ahead of time, estimate various game results, and practice debriefing with participants in your head, but most importantly, that you remain flexible and ready to respond to the conversation's flow. What if the group discovered a new learning dimension during the game that is pertinent to your topic? It would be a pity to only ask the questions that had been prepared beforehand. One thing is certain: prepare general questions that will allow you to transition from discussing the game to discussing its relationship to the rest of the world and pointing out what the players can do with this information or experience.

4. Conclusions

While each type of game comes with its own set of advantages and challenges, there are a few guidelines that can help make the introduction of any game into the course successful. It is important from the beginning of the course to be open with students about the reason for choosing the game, expectations for the students, and that, because the course is a pilot effort, an educator will be learning along with them. In any course that varies from the norm, students often express concern about assessment. This is particularly true in a course with a game where students may feel they must “win” to receive a good grade. By making the learning goals clear, an educator can help students see the failures, which are part of any well-designed game. Finally, when constructing the course assignments, making them interesting and challenging is important (Carnegie Mellon University, n.d).

There are a few other tools left out, either because they did not provide more value in the same category as the ones already presented (e.g., Shiva3D), or because they are too exclusive or expensive to be considered (even though they are indeed comparable feature-wise), such as Source Engine or Gamebryo. Choosing an appropriate gaming tool depends on what kind of game is developing, what is the developer’s experience and how much effort he can put into its development.

References

1. Billen, A. (1993). Could it be the end for Super Mario? The Observer 27 June.
2. Carnegie Mellon University. (n.d.). Retrieved from https://www.cmu.edu/teaching/assessment/assesslearning/creatingassignments.html
3. Carroll, J. M. (1982). The adventure of getting to know a computer. IEEE Computer 15 49–58.
4. Corti, K. (2006). Game-based learning; is a serious business application. PIXELearning. Coventry, UK.
5. Girard, C., Ecalle, J., & Magnant, A. (2013). Serious games as new learning tools: How effective are they? A meta-analysis of recent studies. Journal of Computer Assisted Learning, 29, 207–219.
6. Hanghøj, T., & Brund, C. E. (2011). Teachers and serious games: Teachers roles and positioning in relation to educational games. In S. Egenfeldt-Nielsen, B. Meyer, & B. H. Sørensen (Eds.), Serious games in education: A global perspective (pp. 125–136). Aarhus: Aarhus Universitetsforlag.

7. Horton, W. (2012). “Games and simulations,” in E-Learning by Design, vol. 45, no. 5, pp. 323–398.
8. Ke, F. (2009). A Qualitative meta-analysis of computer games as learning tools. In R. F. Ferdif (Ed.), Handbook of research on effective electronic gaming in education (pp. 1–32). New York, NY: Hershey.
9. Lepper, M. R. and Cordova D. I. (1992). A desire to be taught: Instructional consequences of intrinsic motivation. Motivation and Emotion 16 187–208.
10. Malone, T. W. (1980). What makes things fun to learn? A study of intrinsically motivating computer games. Technical Report CIS-7 Xerox PARC, Palo Alto.
11. Malone, T. W. (1981a). Toward a theory of intrinsically motivating instruction Cognitive Science 5 333–369.
12. Malone, T. W. (1981b). What makes computer games fun? Byte 6 258–277.
13. Malone, T. W. (1984). Heuristics for design enjoyable user Interfaces: Lessons from computer games. Human Factors in Computer Systems, Norwood NJ, 1–12.
14. Malone, T. W., and Lepper, M. R. (1987). Making learning fun: A taxonomy of intrinsic motivations for learning in Snow. Aptitude, learning and instruction III: Cognitive and affective process analysis, Erlbaum, Hillsdale, NJ.
15. McKee, V. (1992). Out of sight—and out of mind? Life and Times, The Times.
16. Meyer, B., & Sørensen, H. B. (2011). Methods and design for research in global-oriented game-based language learning. In M. S. Khine (Ed.), Playful teaching, learning games. New tools for digital classrooms (pp. 51–64). Rotterdam: Sense.
17. Oblinger, D. G. (2006). Games and Learning: Digital games have the potential to bring playback to the learning experience. Educause Quarterly, pp. 1-3.
18. Peng, C. (2016). “Introductory game development course: A mix of programming and art,” Proc. - 2015 Int. Conf. Comput. Sci. Comput. Intell. pp. 271–276.
19. Quinn, C. N. (1994). Designing educational computer games in Beattie K, McNaught C and Wills S (eds) Interactive multimedia in University Education: Designing for change in teaching and learning Elsevier Science BV, Amsterdam, 45–57.
20. Quinn, C. N. (1997). Engaging Learning. Instructional Technology Forum, Retrieved from http://itech1.coe.uga/itforum/paper18/paper18.html.
21. Rivers, R. (1990). The role of games and cognitive models in the understanding of complex dynamic systems in Diaper D, Gilmore D, Cockton G and Shackel B (eds) Human-Computer Interaction—Proceedings of INTERACT ’90 Proceedings of INTERACT ’90, Elsevier, North Holland. 87–92.
22. Rugelj, J. (2015). “Serious computer games in computer science education,” EAI Endorsed Trans. Game-Based Learn., vol. 2, no. 6, p. 150613.
23. Shute, V. J., Ventura, M., Bauer, M., and Zapata-Rivera, D. (2009). “Melding the power of serious games and embedded assessment to monitor and foster learning,” In U. Ritterfeld, M. Cody, and P. Vorderer (Eds), Serious Games. Mechanisms and Effects.Routedle Publishers, New York, NY. USA.
24. Thomas, P. and Macredie, R. (1994). Games and the design of human-computer interfaces Educational Technology 31 134–142.
25. Tzuo, P.-W., Ling, J. I. O. P., Yang, C.-H., & Chen, V. H.-H. (2012). Reconceptualizing pedagogical usability of and teachers’ roles in computer game-based learning in school. Educational Research and Reviews, 7, 419–429.
26. Watson, W. R., Mong, C. J., & Harris, C. A. (2011). A case study of the in-class use of a video game for teaching high school history. Computers & Education, 56, 466–474.

Biographical notes

Aritra Ghosh received his Ph.D. in Computer Science from Florida Atlantic University. Prior to joining the Florida Atlantic University, he pursued his Masters' from Texas A&M University-Kingsville. He is working on developing and updating the Community and Engagement Department and Office of Interprofessional Education Website and other media materials for the promotion and support of the University’s community-engaged activities.