Computer Games

Frederick W.B. Li

E-Mail: Frederick.Li@durham.ac.uk

Department of Computer Science, University of Durham, United Kingdom

I. INTRODUCTION

Computer (and video) games gain significant interests in recent years. Particularly, in US, total game sales for 2005 were over 10.5 billion US dollars, which makes an improvement of 6% over the game sales in 2004. Demographically, 69% of American heads of households play computer and video games. While in UK, which owns the world's third largest computer game market, 82% of 9- to 19-year-old youngsters own a game console. In this article, we are going to discuss the existing technologies and the design issues of computer games. As computer and video games are closely related and technologically comparable with each other, without loss of generality, our discussion will use the term *computer game* to cover both of them.

A computer game is an interactive entertainment running on computers, game consoles or some electronic devices, such as mobile phones and PDAs. Typically, a computer game accepts input from game players through keyboard, mouse, joystick or various types of game controllers to let the players control and interact with game objects. On the other hand, a computer game offers game players different forms of feedbacks, including visual, audio and tactile, which are revealed through computer / TV monitor, speaker and force-feedback game controllers, respectively. Technically, developing a computer game is very challenging as it involves not only tremendous artwork furnishings from animators or artists, but more critically, it demands the hard core supports from different disciplines of computer science to address various technical issues. In this article, we will take a look of the development in computer games. In particular, we will discuss the technical issues in game development. The rest of the article will be organized as follows. Section II presents a brief history of computer games. Section III elaborates the development of 2D and 3D games, multiplayer online games and the handheld and mobile games. Section IV shows the details of the critical elements involving game development. Section V elaborates the details in developing a game engine. Section VI discusses the modern game design issues. Section VII concludes this paper and depicts the future trend of computer games.

II. GAME HISTORY

Computer game has a long history, in which we can trace back its root from 1947, when Thomas T Goldsmith Jr and Estle Ray Mann designed the first game for playing on a cathode ray tube in US. Here, we are not intended to elaborate the full computer game history. Instead, we focus on the technological evolution of computer game and technical issues arisen during the change. During 1960s to 1970s, games developed were simple, primitive and mainly in two-dimension (2D). Many games, in particular of different types, were developed. Two of the most unforgettable examples are Space Invaders and Pac-Man. In addition, handheld game devices were also developed, despite most of them are hard coded to run only a single game. For game playing, game players used

simple button controllers, keyboards or joysticks to control their game characters. The main forms of feedback were offered through screen displays with limited color and resolution as well as simple sound outputs. In 1980s, there was a major growth in computer game technologies. For hardware, a variety of personal computers and game consoles were developed. For software, three-dimensional (3D) games and network games were first developed. In addition, different forms of input and output devices were developed. These included color monitors, sound cards and various types of gamepads. They offer game players better game feedbacks and greater flexibility in controlling game characters. In 1990s, games developed planted the seeds of today's game development. Many classic game types, including first-person shooters (FPS), real-time strategy (RTS), daily life simulators and graphical multiplayer games, were developed during this period. Also, there was a trend for developing 3D games. Nowadays, many new games are developed based on these classic games. The major difference of the new games from the classic ones is that the new games are mainly in 3D. Hence, hardware graphics accelerators were urged to develop to support real-time rendering of 3D game content. To take the advantage that human visual sense has a dominate influence of a game player to determine whether a game is good or not, game companies put their very first priority in enhancing the graphics output for their games. They put advanced graphics content, such as high detailed 3D models, texture images and various special effects, into the games. Besides, multimedia elements, such as videos and songs, are also used to enrich the game content. However, such arrangement increases the complexity of the hardware requirement of running computer games. It also demands the development of efficient algorithms to manage such a variety of game content. To optimize both the man-power and time spent in game development, game developers begin to make use of ready-made game engines (Please refer to section V), which comprise many useful tools to support general game functions, to build their games.

When we go through the history of computer games, we note that during the early stage, games were mainly simple and small. They could be generally handled by simple computation and graphics processing power. Hence, there was not really any stringent requirement putting on the development of these games. Later, when game developers turned their focus to 3D games, working out both hardware and software solutions in supporting real-time rendering of 3D graphics has subsequently become a critical part of game development. Besides, game physics and game artificial intelligence (AI) also played an important part of the games. While game physics provides support to collision detection and motion control of game characters, game AI offers autonomy to the non-person controlled game characters to govern how these characters are behaved in the game environment. Recently, as multiplayer online games begin to dominate the game market, issues including network latency, system scalability and security are turned out consequently. Eventually, these make the game development facing various technological design issues from different disciplines in computer sciences.

III. TYPE OF GAMES

A. 2D and 3D Games

Technologically, computer games can be broadly categorized into 2D and 3D games. 2D games generally manage the game environment into a logical 2D space, where the game objects can be moving and interacting around. Practically, a majority of 2D games, such as Pac-Man, Load-Runner and Mario, can be implemented using a simple tile-based concept [TILE]. It is by partitioning the game environment into cells, which are hosted in a 2D array. Then, different states can be assigned to individual array element to logically represent different game objects and game scene elements in the game environment. When some objects are moving around in the game environment, the corresponding elements of the 2D array are then updated to reflect the change. To render game objects or game scene elements, it can be done as easy as using simple graphics primitives, such as line, point and polygon, or alternatively, using picture images to present the game objects or game scene elements in a more impressive and realistic way.

As the successor of 2D games, 3D games offer greater attractions to game players in terms of game interaction and visual effect. The game environment of such games is hosted in a 3D space. As one more dimension is provided, game players have been offered with a greater degree-of-freedom (DOF) in controlling their game characters and interacting with other game objects. They can also navigate and interact in the game environments with a variety of camera views. In addition, the 3D setting of these games also gives a variety of advanced computer graphics effects a sufficient environment for implementation, which can then be visualized by the game players. Examples of the graphics effects range from some low-level graphics techniques, such as lighting and shadowing, to some high-level graphics techniques, which cover a wide variety of natural phenomenon simulations [Deus01]. In addition, computer animation techniques [Pare01] can also be employed to make the game objects move in a realistic way. Unlike 2D games, which can natively be displayed by most of the 2D display controllers, 3D games need a rendering process to covey 3D game content to present on 2D display devices. In light of supporting interactive game playing, the rendering process must be carried out with a frame rate of 25-60 frames per second, which means 25-60 pictures should be produced within every second for the relevant portion of the game content for display purpose. To support such performance, the rendering process is often needed to carry out by hardware graphics accelerators.

B. Multiplayer Online Games

Multiplayer online games have been developed as early as in the late 1980s. The uniqueness of this game type is that it connects people from geographically dispersed locations to a common game environment for game playing. One of the early developed online games was the *Modem Wars*, which was a simple 2D game designed for the personal computer - Commodore 64. The game connected two game players using modems into a shared game environment. During game playing, game players can interactively move the game items around the game environment to attack any enemy items located within certain pre-defined range. Despite the game design was

simple, it set a good starting for the multiplayer online game development.

Nowadays, multiplayer online games become one of the most popular game types. Unlike the old days, instead of running such games on peer computers and connecting the game machines through modems, the new online games are generally hosted in some server machines running on the Internet, which are referred as *game servers*. Game players from different geographical locations can connect to the games through some broadband network connections using their preferred game platforms, which can be a computer or a game console. Such game platforms are referred as *game clients*. The first commercial multiplayer online game of this type was, *Meridian 59*, published in 1996 by 3DO [3DO]. Thereafter, some major game types include first person shooting (FPS) game, real-time strategy (RTS) and role-playing game (RPG), had become dominating the market of multiplayer online games. Technically, the main responsibility of a game server is to keep track the actions issued by the game clients and to propagate these actions and the state updates of the game environment to the relevant game clients. The game server also needs to build up a profile for each game player to store the game state and the possessions of the player, such that the players can keep on playing the game in future based on the saved profile. Today, most of the game developers focus on two major issues when develop multiplayer online games. First, they try to seek good game server architectures for hosting games to allow a larger number of game players to join and play around their games. Second, to attract more people to play their games, the game developers take advantage of the advancement in computer graphics technologies to make their game have very impressive 3D graphics presentations and visual effects.

C. Handheld Games

In contrast to computer-based or game console-based games, handheld games are run on machines with small machine size. It allows people to carry along anywhere and play around with it at any time when they are free. Generally, such machines can be referred to dedicated handheld game consoles, personal digital assistants (PDAs) or mobile phones. Due to the hardware limitation, such game devices are often suffered from small in screen size and limited in processing power and storage space, as well as the problem of short battery life. These problems do not only impose difficulties to handheld game development, they also make some people reluctant to play handheld games. Fortunately, these shortcomings have been addressing or have ways to workaround during recent years.

The first handheld game console is *Tic Tac Toe*, which was made in 1972. Similar to most of the early handheld game consoles, it came with one hard-coded game only. This limitation had been lasting until 1979, when the first handheld game console with changeable cartridges, *Microvision*, was developed. In general, as handheld game consoles at that period were suffered from the problems of small in screen size and limited in battery life, handheld games had not received a truly great success. Until the release of Game Boy [Nintendo] in 1989, which came with a monochrome display with improved resolution, used re-chargeable batteries and had a long list of game cartridges for game players to pick and play, handheld games became to attract significant amount of game players. More importantly, Game Boy virtually set the “design standard” for today’s game consoles. In addition,

in 1998, the color display version of Game Boy was released to further improve the attractiveness of handheld game consoles. Nevertheless, up till the release of Game Boy and its color display version, as the processing power of the handheld game consoles was still quite limited, most of the games developed for the handheld game consoles were still essentially 2D games.

Starting from 2000, there is a dramatic development in handheld game consoles, particularly in terms of computation and graphics processing power. In light of this improvement, 3D games had been engaged to these devices, an example of this was in Game Boy Advance [Nintendo]. On the other hand, useful accessories, such as network connections, external memory storages and new types of input devices were added to the game consoles. Regarding to the network capability, examples could be found in Nokia N-Gage [Nokia] and Nintendo DS [Nintendo]. This made multiplayer online games be possibly supported. To extend the storage capacity, Sony had made use of UMD disks and Memory Stick Duo as a media to extend the amount of storage of its newest handheld game console, Sony PSP [Sony]. For input device, Nintendo DS adopted a touch screen approach, where game players could use stylus or even the player's finger as an input method to control the games objects.

On the other hand, similar to dedicated handheld game consoles, PDAs and mobile phones are also featured with high mobility, this makes such devices become alternate platforms for handheld games. More importantly, from the business point of view, putting games on essential devices, such as PDA or mobile phones, is favorable as this frees people from investing or carrying addition game devices for entertainment. In addition, mobile phones and modern PDAs also natively come with network capability to provide a critical support for running online games. However, before PDAs and mobile phones become substantial handheld game devices, the technical problems of these devices, such as small in screen size and limited in storage space, which could also be found in dedicated handheld game consoles, must be solved.

IV. GAME DEVELOPMENT PROCESS

Nowadays, making a game no longer focuses only on working out the game logic or game features and the graphical display for the game. Depending on the resource availability and the business strategy of a game company, a variety of associated tasks may also involve in the modern game development process. These tasks include game hardware and software development, media construction, localization and even the handling of the cultural and social issues:

- **Game Hardware**: This refers to the development of game consoles and game input / output devices. Such development may usually introduce new attractions to game players. It also offers game companies niches in developing proprietary hardware specific games or obtaining license fee from developers who develop games on such game hardware. However, as developing game hardware usually requires quite a significant amount of investment in terms of man-power, time and money, only large game companies can afford such development. More technical design issues on game hardware will be discussed in section VI.
- **Game Software**: Game software refers to the technical part of the game development process. It involves

the development of various game software components, which may include some hard core game functionality, such as game content management and rendering, game animation, game artificial intelligence (AI) and game physics. In addition, it may also involve the development of game networking and game security, which depends on the type of the game for development. More details on this will be discussed in section V and section VI.

- **Media Construction**: Media construction refers to the artistic part of the game development process. It involves the development of game content by using different types of media, which may include image, 2D/3D graphics model, audio, video and motion capture information [Meye92]. As media offers the presentation of game content, which determines how game players perceive a game, media construction becomes an inevitable part of the game development process. Nowadays, many game companies have been investing a significant amount of resources for the media construction process.
- **Localization**: Localization is the process to turn a computer game into a country or a target market specific version. This helps a computer game to boarder its market share and helps introduce country or market specific attractions to the game players. The localization process can be done as simple as by conveying the language of the user interface and the textual content of a game. In a more complicated way, we may further change the game characters or other game media content to a country or market specific ones. Furthermore, we may even consider altering the storyline of a computer game to suit the culture or custom of the country or target market specific game players.
- **Cultural and Social Issues**: During recent years, there is a rising concern in the cultural and social effect of computer games on our human, especially, on the youngsters. On the one hand, more and more people are getting addicted to computer game playing, particularly after the release of multiplayer online games. This likely has a bad effect to the academic performance of addicted student game players. It may also significantly reduce the amount of time for people to participate in social activities. On the other hand, the release of game titles with violent and sexual game content imposes negative ethical effect to the young people. Although there is not a consensus on the handling of the cultural and social issues of computer games, the game companies should try their best to maintain a good attitude in addressing these issues during the game development process.

V. GAME ENGINE

Making computer game is a complicated task. From the hardware perspective, when constructing a game, game developers may need to deal with a wide range of hardware and software platforms as well as work hard on a number of game components. More specifically, a computer game may need to be designed running on different game platforms, including computer, game consoles, which are usually controlled by different operating systems. Even under the same hardware platform, the game may need to be rendered by different graphics accelerators, and relied on different graphics application programming interfaces (APIs) to drive the graphics accelerators.

From the software perspective, when developing a computer game, game developers typically need to work on a number of game components, in which the most essential ones include game content management and rendering, game animation, game artificial intelligence (AI) and game physics. Working out these components generally involves a lot of efforts and is very time consuming.

To minimize the complexity of game development by hiding the differences in various game platforms and help game developers put their focus on developing high level game logics and game features, game engine has been developed, which comprises a set of basic game building blocks and provide a high-level and unified abstraction for both low-level graphics APIs and hardware game platforms. With the help of game engine, the investment of game development, in terms of time, man power and cost, can significantly be reduced. Reputable examples of game engine include Unreal Engine 3 [UE] and RenderWare [RW]. In practice, there are not any standards or rules to govern the exact game components to be included in a game engine. Game engine developers have a great flexibility to select the appropriate set of components to make their own engines. However, there are some major game components that are essential to most of the games and hence, should better be included to develop a game engine:

- **Game content management and rendering**: Game content management and rendering is one of the most core parts of a computer game. It comprises techniques to manage game content in a way to support efficient content retrieval and processes for making the game content to be displayable on the output device of the game. For game content management, most of the game engines adopt a scene graph approach [PHIGS89], where the game objects and the graphics primitives are hierarchically maintained with a tree structure. As such tree structure implicitly links up the game objects according to their spatial relationship, this provides sufficient information for a game to pick out closely located game objects to support game rendering and game object interaction evaluation. On the other hand, for game content rendering, particularly when dealing with 3D game content, there will be a significant amount of processes as well as a variety of options to go through for converting game content from the 3D representation into the 2D one for display. These processes and major options are shown as follows:

    1. *Standard graphics rendering processes* [Hear04], such as perspective transformation, clipping, hidden surface and back face removal, are fundamental to render 3D game content into 2D images for display. As these processes are usually natively come with the hardware graphics accelerators, game engine developers need not to develop their own algorithm to support these processes. Instead, consider that the standard graphics rendering processes typically run on various hardware graphics accelerators, which are controlled by different low-level graphics APIs, such as OpenGL and DirectX, game engine developers may better work out high-level unified abstraction on these graphics APIs and hardware to help reduce the effort of game developers to construct games for a variety of game platforms.

    2. *Shading and texturing* are the major processes to fix the appearance of individual game object. The

shading process takes in the lighting and the object material information to evaluate the appearance of a game object by applying certain empirical lighting models. Common options of shading include flat shading, Gouraud shading and Phong shading. They offer different degree of realism to the rendered game objects. On the other hand, texture mapping adds or modifies detail on the game object surface. Basic options of texture mapping include basic texture mapping, multi-texturing and mip-mapping, which arranges captured or artificial images to add on the surface of a game object. Advanced options of texture mapping include bumping mapping and displacement mapping. They make use of certain specially designed "texture maps", which comprise geometry modifiers rather than generic images, to modify the appearance of the geometric details over the surface of a game object.

3. *Advanced rendering options* can be taken to further enhance the realism of the overall game environment. They include but are not limited to reflection, motion blur and shadowing. Adopting these options definitely help attract game players' interests as they make the rendered game environment look more realistic by adding details of some natural phenomenon to the game scene. However, as such rendering options generally require time consuming add-on procedures to be executed on top of the standard graphics rendering processes, taking such options would likely degrade the rendering performance of a computer game significantly. Therefore, game developers should better put these options as optional choices for game players to take rather than set them as mandatory game features.

- **Game AI**: Game AI [Mill05] is a way to give "lives" to the non-person controlled game characters (NPCs) of the game environment, it directs the way of the NPC to interact with the game environment or other game objects. Putting different game AI to a NPC can assign different behaviors to the NPC. In fact, one of the major reasons for computer game to be so attractive is that game players can find different challenges and funs when play against the NPCs inside the game environment. To implement game AI, two major options are available:

  1. *Reactive techniques* are widely adopted in many computer games, as they are fully deterministic. Examples of these techniques include scripts, rule-based systems and finite-state machines. Such techniques take in some given game parameters or game states, which are then evaluated through pre-defined rules to produce deterministic results. Practically, reactive techniques are good for implementing high-level tactical decisions.

  2. *Planning techniques*, in contrast, are non-deterministic. From a given situation, multiple actions can be taken depending on the current goal or some selected factors. A planning algorithm can scan through the possible options and find the sequence of actions that matches the goal or the selected factors. For instance, A* is the most reputable example of planning techniques. Practically, planning techniques are good at helping search best possible path for a game object to navigate in

the game environment.

- **Game Physics**: Game physics [Eber03] is developed based on the laws of physics to govern how each individual game object reacts with the game environment or other game objects. It also offers a way to support the simulation of some natural phenomenon. Typically, the reaction of a game object can be determined using mass, velocity, friction, gravity or some other selected physical properties. In practice, game physics can be natively applied to help generate realistic response for the collision or interaction of the game objects. Alternatively, game physics can be applied to drive the motion of a large amount of tiny particles for simulating some natural phenomenon, such as the flow of smoke and water, fire blaze, snow and cloud.

- **Animation**: Animation [Pare01] is a technique to drive the motion of the body of the game characters. Typically, there are several ways to produce animation in computer games. They are shown as follows:

    1. *Key-framing* [Burt76] requires animators to define and draw key-frames of a motion sequence of the game character to be animated. However, manipulating and coordinating the limbs of a game character via key-framing is a complicated and tedious task. In addition, it is also difficult to produce realistic and natural looking motions with key-framing.

    2. *Inverse kinematics* [Wang91] computes the pose of a game character from a set of analytically constrained equations of motion. It can generally produce physically realistic motions.

    3. *Motion capture* [Meye92] acquires movement information from live objects. The captured position and orientation information from motion capture can then be applied to game characters to drive their motions. This approach has been widely accepted as it helps produce realistic and natural looking character animations.

VI. MODERN GAME DESIGN ISSUES

Since the release of the first computer game, computer games have becoming one of our major entertainments. During the years, as game hardware becomes much and much powerful, the game players' expectation on both the visual quality and the performance of computer games has then been increased significantly. Such expectation has partially been tackled by the release of new computer graphics algorithms and hardware. However, due to the advancement in computer network technologies and the popularity in multiplayer online games, game design issues are no longer restricted to computer graphics or input/output devices related ones. Instead, the game design issues have become multi-disciplined and are much challenging than ever. The following shows the modern technical design issues that game developers should need to pay attention on when developing their new games:

- **Advancement in Hardware Graphics Accelerator**: Owning to the increase in the demand for high visual quality and high realism of the graphics output of computer games, a brand-new type of hardware

graphics accelerator – Graphics Processing Unit (GPU) [Phar05] has been developing, which offers a dramatic improvement in the rendering performance at the game clients. Such improvement is due to the fact that GPU allows the major rendering tasks, including the standard and the advanced game rendering tasks as well as game physics, to be executed in parallel rather than carries tem out in sequential manner as in the traditional hardware graphics accelerator. To take advantage of such advancement in hardware graphics accelerator, game engine developers should seek ways to parallelize the algorithms used for implementing the game engine components.

- **Game Controller (Input)**: Game controller [Controller] is referred as the primary input device for game players to issue commands or actions to drive game characters or interact in a computer game. Common game controllers include keyboard, joystick and mouse. To use these controllers, game players need to convey their actions getting in minds into controller dependent operations. However, such operations may not always match well with the activities or interactions of one's game playing, especially for the games about human's real life activities, such as driving, dancing, musical instrument playing and shooting. Recently, game developers began to realize that the design of the game controllers could be one of the critical determinants for the success of a computer game. Hence, they have been working out many different game controllers to attract game players' interests and offer better game playing experience. On the one hand, game specific game controllers have been developed, examples include the steering wheel for driving or racing games, the dance platform for dancing games and light guns for shooting games. Through these controllers, game players can act naturally as performing real life activities during their game playing. On the other hand, game developers actively create new types of game controller. Examples include the EyeToy of Playstations [EyeToy] and the wireless controller – Wii Remote of Wii [Wii]. Particularly, EyeToy makes use of a webcam to capture human motion, which virtually forms a game controller for a game player to control and interact with the game. Wii Remote is a wireless game controller with high degree-of -freedoms to let a game player generate a large variety of free motions or perform human's real life actions for game playing. These game controllers also lead to the development of new game types.
- **Game Feedback (Output)**: Game feedback is the way of computer game to give game players response for their actions. It is the one of the most direct ways for game player to feel and to be impressed on how good a computer game is. The primary ways for offering game feedback is to put different graphics effects on the screen and to generate sound effects for responding to game players' actions. However, as more game devices are released, a greater variety of game feedback is made available for game developers to consider putting in their games. Broadly speaking, we may have three ways of game feedbacks. They include tactile, audio and visual feedbacks. For tactile feedback, the most common form can be found in the most of the game pads of modern game consoles. It is typically done by generating vibrations to certain game events or game players' interaction. Another form is the resisting

force used in the steering wheel for the car games. For audio feedback, as new sound compression and speaker technologies are emerging, multi-channel sound feedback is now made available to provide higher sound quality and support 3D sound rendering. In particular, 3D sound offers new type of spatial feedback for objects or interactions in a game environment, which originally can be provided by the visual feedback only. For visual feedback, conventional approach focuses on rendering 3D graphics and visual effects to display on 2D screen. But as stereo display technologies is getting mature and widely available, 3D display is made possible as an alternative option for visual feedback.

- **Scalability**: Multiplayer online games are the most fast growing game type in recent years. They allow a large amount of remote game players to get connected for playing in a shared game environment. Existing multiplayer online games have been implemented with distributed game servers. However, some of the games, such as Quake III Arena [Quake] and Diablo II [Diablo], maintain individual game state for each game server, which is not shared among the servers. This is essentially a set of separated client-server systems running the same game and may not be considered as a real multi-server system. EverQuest [EverQ], in contrast, divides the entire game environment into distinct zones, and maintains these zones by individual game servers. EverQuest allows a client to travel from one zone (game server) to another freely. Ultima Online [Ultima] and Asheron's Call [Asheron] adopt similar approach as EverQuest, but they divide the entire game environment into visually continuous zones. The boundary of each zone is mirrored at neighbor server to reduce the lag problem and to improve interactivity when a user crosses from one zone to another. In addition, Asheron's Call is technically more advanced in that it may dynamically transfer a portion of the zone controlled by a given game server to any other lightly loaded server. Unfortunately, game object coherency is not considered.

- **Network Latency**: A unique characteristic of multiplayer online games is that game updates are required to send to remote game players over the network throughout the game playing sessions to renew the states of the local copies of shared game objects at the game players. More than that, as multiplayer online games involve time-dependent game data and continuous user interaction, the state update events are therefore *continuous* [Mauv04] in nature. However, due to the existence of network latency, such updates are likely arrived at the remote game players after some delay. This leads to state discrepancy of the shared game objects among the game players. This fact opposes the sufficient condition for supporting game player interaction, in which the state updates need to be presented to remote game players either without any delay or at least within a very short period of time.

  To cope with the latency problem, adaptations could be performed at either the user or the system side. For user side adaptation, Final Fantasy XI [FF11], which is one of the popular online games currently available in the market, imposes restrictions to game player. In this game, a player usually receives position updates of other game players with almost a second delay. In order to reduce the effect of such delay, first, players can only attack enemy objects, but not each other. Second, the enemy objects are

designed to move very little while they are under attack by a player. Such game rules significantly limit the game features and the type of games that can be developed. For system side adaptation, a popular approach is to use dead reckoning [DIS98]. With this approach, the controlling game player of a game object runs a motion predictor for the object. Other game players accessing the object also run the same motion predictor to drive the motion of the local copies of the object. The controlling game player is required to keep track of the error between the actual and predicted motions of the object, and sends the updated motion information to the other game players when the error is higher than a given threshold. Although this approach is very simple, it does not guarantee that the state of a shared object could be synchronized among all game players.

Recently, a trajectory preserving synchronization method [Li06] has been developed to tackle the problem. The method runs a reference simulator for each object on the game server. Each of the game players interested in the object, including those that access the objects as well as the owner of the object, will execute a gradual synchronization process on the local copy of the object to align its motion to that of the reference simulator running at the server. The method effectively reduces the discrepancy suffered by remote game players.

- **Game Cheating**: Game cheating generally refers to the activities that modify the game experience to give a game player an unfair advantage over the other players. Common ways of game cheating can be done in terms of user settings, exploits and using external software. To cheat using user settings, one can change game data, game models, game environment or game devices to make one to play a game in an easier way. Exploits, on the other hand, refers to the use of existing bugs of a game, or game features or tricks that are reserved for developers for testing purposes and are unintended to release to game players, to gain certain advantages for game playing. Other than the above, people may develop external software to help modify the normal behavior of a game to let a game player perform game cheating. To cheat in online games, packet tampering is a common way to use. It is by altering the game data sending between the game server and game clients. To prevent game cheating, game companies should take appropriate measures. Typical solutions may include the use of anti-cheating software and banning suspected cheaters from playing a game.

VII. FUTURE TREND AND CONCLUSIONS

Based on the development in computer games, we enumerate and elaborate below a number of emerging issues on the future development of computer games, much of which serves the purpose of highlighting the directions for future research and development. In addition, it is anticipated that many new ones will emerge in time as the technologies evolve. What we have pointed out here is only some important emerging issues and technologies, and we are leaving the readers to comment and decide which ones are the most important ones.

- **On-Demand Transmission**: Nowadays, multiplayer online games are usually set out with a large scale

scene environment together with high quality game objects in order to attract more game players. A standard way to distribute such a large game content is to put it on CDROMs/DVDs for game players to purchase and subsequently install at their local machines. This scenario works fine if players' local machines have sufficient storage to hold the complete set of game content. Clearly, due to limited storage, it is difficult for handheld devices or portable game consoles to access multiplayer online games in this way. To loosen this limitation, a game-on-demand approach [Li04] can be adopted. It is by using a prioritized content delivery scheme to help identify and deliver relevant game content in suitable quality to support users' game playing. In addition, the game-on-demand approach also serves games with very large-scale game scenes and high detailed game objects to be accessible to machines with any kind of network connection speeds. On the other hand, nowadays, game companies typically continue to develop new game scenes and game objects after they have released their games. It is not only because game companies want to gain revenue before they have finished developing their games, they also need to add new attractions to keep motivating people to play their games. With the game-on-demand approach, this process can be done totally transparently to the game players without requiring them to install patches as they do now, as new content can also be streamed progressively during game playing.

- **Web-Based Client Technologies**: Recently, there has been a dramatic development in client technologies for web-based applications. Such technologies are emerging as important development tools and are critical for the future development of web-based computer games. Typical examples include Flash, Java and ActiveX. They help improve the user interface control of web-based applications by getting through the problem of limited in interaction control and webpage reload when a user performs an action. More than that, client technology, likes Flash, serves users with an interactive user interface as well as the multimedia support. The only requirement for adopting these client technologies is the pre-installation of application interpreters and plug-ins. In addition, a more advanced option, Asynchronous JavaScript and XML (AJAX) [Eich06], has been made available since 2005, to provide more interaction control, such as context sensitive menus and window interface, for developers to construct more serious web-based application. AJAX also relaxes the need of pre-installation or maintenance of add-on plug-ins, and requires only a minimal amount of data transmission for application download, as they are driven by light-weighted JavaScript and XML codes. Finally, during run-time, AJAX can collect information from web servers to update only a required part of the application content without the reloading webpage.

- **Security Challenges**: Security control in existing computer games is partially addressed. However, as there is a trend that more and more computer games will be hosted on the Internet, the risk of these games being accessed by unauthorized people and being cracked for cheating will be increased as well. To address this problem, secure socket layer (SSL) could offer a good choice. It provides end-point authentication and communications privacy over the Internet using cryptography. In particular, once a

user has logged into a computer game, such data protection process would be done transparently to the user.

- **Computer Games on Grid**: Grid Computing [Fost03] is considered as a hot research area in recent years. Grid connects computers, storages, sensors, and services via fixed-line or wireless Internet (and other networks). The goals of Grid Computing include resource sharing, coordinated problem solving, enabling dynamic virtual organizations/world, and thus service-oriented architecture is considered as a promising direction. Advantages of Grid include data and performance reliability (avoid single point of failure), performance (avoid network and computation bottleneck), and resilience (guarantee existence of data and its backup). In line with the change of new grid concepts, computer games can be considered as one important service to this emerging grid virtual world.

REFERENCES

[3DO] The 3DO Company, URL: http://en.wikipedia.org/wiki/The_3DO_Company.

[Asheron] Asheron's Call, URL: http://www.microsoft.com/games/zone/asheronscall/.

[Burt76] N. Burtnyk and M. Wein, "Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation," *Communications of the ACM*, **19**(10):564–569, 1976.

[Control] Game Controller. URL: http://en.wikipedia.org/wiki/Game_controller.

[Deus01] O. Deusen et. al., "The elements of nature: interactive and realistic techniques," *ACM SIGGRAPH 2004 Course Notes*, 2004.

[Diablo] Diablo II, Starcraft, URL: http://www.blizzard.com/.

[DIS98] DIS Steering Committee, "IEEE Standard for Distributed Interactive Simulation - Application Protocols," *IEEE Standard 1278*, 1998.

[DX] DirectX. URL: http://www.microsoft.com/directx/.

[Eber03] D. Eberly, *Games Physics*, Morgan Kaufman, 2003.

[Eich06] J. Eichorn, *Understanding AJAX*, Prentice Hall, 2006.

[EverQ] EverQuest, URL: http://everquest.station.sony.com/.

[EyeToy] EyeToy. URL: http://www.eyetoy.com/.

[FF11] Final Fantasy XI. URL: http://www.playonline.com/ff11/home/.

[Fost03] I. Foster and C. Kesselman, *The Grid 2*, Morgan Kaufman, 2003.

[Hear04] D. Hearn and M. Baker, *Computer Graphics with OpenGL (3rd Ed.)*, Prentice Hall, 2004.

[Li04] F. Li, R. Lau, D. Kilis, "GameOD: An Internet Based Game-On-Demand Framework," *Proc. of ACM VRST*, pp. 129-136, Nov. 2004.

[Li06] L. Li, F. Li, and R. Lau, "A Trajectory-Preserving Synchronization Method for Collaborative Visualization," *IEEE Trans. Visualization and Computer Graphics (special issue on IEEE Visualization '06)*, **12**(5):989-996, Oct. 2006.

[Mauv04] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg, "Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications," *IEEE Trans. on Multimedia*, **6**(1):47-57, 2004.

[Meye92] K. Meyer, H. Applewhite, and F. Biocca, "A Survey of Position Trackers," *Presence: Teleoperators and Virtual Environments*, **1**(2):173–200, 1992.

[Mill05] I. Millington, *Artificial Intelligence for Games*, Morgan Kaufman, 2005.

[Nintendo] Nintendo. URL: http://www.nintendo.com/.

[Nokia] Nokia. URL: http://www.nokia.com/.

[OGL] OpenGL. URL: http://www.opengl.org/.

[Pare01] R. Parent and R. Parent, *Computer Animation: Algorithms and Techniques*, Elsevier Science & Technology Books, 2001.

[Phar05] M. Pharr and R. Fernando, *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, Addison-Wesley Professional, 2005.

[PHIGS89] *Computer Graphics - Programmer's Hierarchical Interactive Graphics System (PHIGS)*, ISO/IEC 9592-1:1989, American National Standards Institute, New York, NY, 1989.

[Quake] Quake, URL: http://www.idsoftware.com/.

[RW] RenderWare. URL: http://www.csl.com/.

[Sony] Sony. URL: http://www.sony.com/.

[TILE] Tile Based Game. URL: http://www.tonypa.pri.ee/tbw/index.html.

[UE] Unreal Engine 3. URL: http://www.unrealtechnology.com/html/technology/ue30.shtml.

[Ultima] Ultima Online, URL: http://www.uo.com/.

[Wang91] T. Wang and C. Chen, “A Combined Optimization Method for Solving the Inverse Kinematics Problems of Mechanical Manipulators,” *IEEE Trans. on Robotics and Automation*, **7**(4):489– 499, Aug. 1991.

[Wii] Wii. URL: http://wii.nintendo.com/.