Digtal STB Game Portability Based on MVC Pattern

HUANG Jun
Department of Computer Science and Technology
China Jiliang University
Hangzhou, Zhejiang, China

CHEN Guangping
Department of Computer Science and Technology
China Jiliang University
Hangzhou, Zhejiang, China

Abstract—**Due to the tight coupling between the game and the STB environment, the ability to migrate the game to different STB environments is greatly restricted. We first analyzed the STB game software porting difficulties, and then put forward a solution to address the causes identified, presented an architecture based on Model-View-Controller pattern. The architecture managed to allow the same game to be ported to different STB environment without or with a little modifying the game. Lastly the architecture performance which revealed the architecture performance strengths and weaknesses was evaluated.**

Keywords-set-top box; game portabilit; game architecture; MVC mode

I. INTRODUCTION

Set-top boxes (STB) employed in Digital Video Broadcast Networks in the past were mainly used for digital TV distribution. They were simply designed to receive and decode digital encoded video-/audio-data streams [1].With the development of digital television, the STB technology has been developed continually, and the accession of family digital television to the Internet becomes popular. It has wide prospects for the network interacting games based on the STB. By October 2007, more than 80 million games had been played and more than ten million households had played the game [2].

Taking into account the international criterion about STB have not yet unified, the STBS produced by different companies varied greatly in the architecture of hardware and software, design patterns and techniques used by companies and its powerful cross-platform performance has been greatly limited. In order to solve the problem of game cross platform for different STB hardware and software environment and portability from one STB environment to another STB, make the game design and development can adapt to different environments and standards on STB, the game platform running on the STB has to be excellent maintainability, scalability, reliability and reusability. In the paper, we studied the gaming cross platform in its portability, and proposed a MVC design pattern which is portable and can cross different hardware and software environment on the STB.

II. STB GAME PORTING DIFFICULTIES

Set-top box is an interactive device which integrates the video and audio decoding capabilities of television with a multimedia application execution environment. It provides a user friendly interface offering personalized multimedia services and regular cable TV service.

A significant amount of different STB devices is produced and sold because there are segments of the market with distinct needs and financial resources. Therefore, the game developer needs to adapt the games so that these comply with the specific requirements of each target device. Even focusing on the J2ME universe[3], which is currently the most used platform form developing STB games, porting demands significant effort from the development team due to several challenges followed:

1) Differences in STB hardware structure: The physical components of a STB can be generally categorized into five subsystems [4]: Front End, Demultiplexor and Descrambler, Decoder, Microcontroller, and Output Encoder. A conceptual view of modern STB architecture is depicted in figure 1. However, different vendor products and different STB grades are quit varied in the hardware architecture and sometimes are not compatible each other. In general, these different STB hardware architecture can be summarized mainly three types: the structure based ASIC, the structure based digital multimedia signal processor and structure based X86；

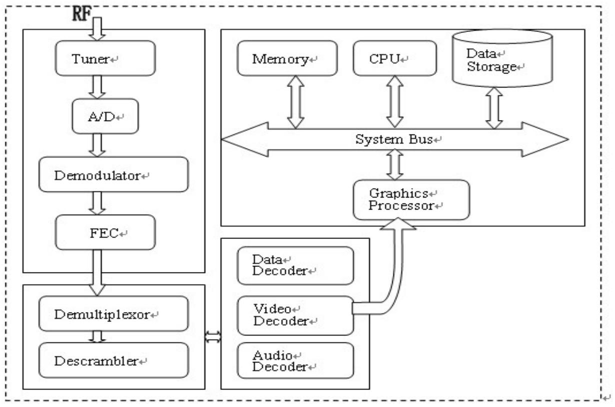

Figure 1. Conceptual View of STB Hardware Architecture

2) Differences in operating system used: STB software architecture generally has four layers: hardware drives, operating system, middleware and service applications. The operating system in a STB manages all the hardware resources and keeps them accessible to applications through API. Event-driven model is wildly used in STB operating system design to fulfill the requirement of real time. Though a shared open system would be beneficial for both vendors and customers, there is not standard set-top real-time operating system by now. Many broadcasters and consumer electronic manufacturers develop their own operating systems for STB, and currently the existing STB operating systems mainly are: Power TV OS, VxWorks, pSoSystem, Microware's DAVID OS-9, Microsoft's Windows CE, Embedded Linux, etc.

3) *Differences in the middleware used:* The middleware layer abstracts the interaction between application and operating systems [4]. With the help of middleware, set-top applications do not need to care about the details of operating system and hardware configurations. Virtual machine is an important form of middleware in STB environment. A virtual machine not only abstracts the behavior and function of the hardware, but also defines the common appearance and representation of application. The most common used virtual machine is Java Virtual Machine (JVM). In STB middleware market, there are many kind of mature products, such as Opentv EN2 、Liberate TV Navigator for DTV, EnreachTV for DTV, Canel+ Mediahighway, IB Browser and so on, all of which have occupied a certain share of market and not compatible.

4) *Despite the STB manufacturers efforts* to make their devices totally compatible with the J2ME standard specification, most device manufactures supply proprietary APIs which extend standard J2ME functionalities. These innovations could be not used to support porting. However, some games frequently rely on such native APIs and the game developer have to take advantage to build professional and powerful games, which also make game porting very expensive and complex.

As a result of above factors, developers are frequently forced to develop up to dozens of variations of a single game, optimized for different types of devices [3]. This further complicates the game development process, thus very likely having a negative impact on the quality of the resulting software, because these variations usually involve modifications scattered across various artifacts. Accordingly, providing consistent maintenance of these variations becomes a more expensive and error-prone task, as the functional common core is normally dispersed across such variations.

Portability is the ability to migrate the game to another STB environment without having to redevelop or modify the game specifically to suit that environment. In order to address hardware platform crossing and games porting problems of the STB games and the game on STB is highly independent of any specific STB hardware and OS platform, we have conducted a research on game porting of STB and put forwards a game platform design architecture based on the MVC model, which was proved to be effective, and make games portable between different STB device.

III. Tight coupling Problem Analysis

The typical game development approach can be grouped into three main steps [5]. First, game developer design and build game data resource by model building tools, sound design tools and so on. Then they use the level editor to create the game environment using data created in modeling and sounding tools, etc. Finally, the game behavior, which controls the game, is added by direct access to the game API or by using a scripting language or by using an editor. The design contents include event handling, physical design, rule design, graphic engine design, sound engine design and preparation of game data.

In the game development, the key factor which causes STB games tightly coupled to the STB environment they were run on is that when a game is being developed and the game state is asked for to be put in its own game state format which should complied with .its relying STB environment and private data-statue format. The another factor is that the game requires the objects represented should comply with its relying STB environment and is only accessible through the its own interface The consequence of using the game model would mean that the manipulation of the game state would always be dependent on the game model interface which would correspondingly mean that it would have to be carried along with the game when moving to another STB environment. The last is the game requires the behavior to be specified in the game STB own language or format.

Accordingly, the main research focus on STB game portability.is to make clear the currently popular hardware and software environment used in different STB devices, research the technology of operating system and middleware used in building game logic, game object model and game state, their effects on game dependencies on STB environment that cause tight coupling to exist between the game and the STB environment. By using J2ME popular methods, such as messaging control mechanism, scripting design mapping, API and so on, a new mechanism is required to design to reduce these tight coupling and increase game portability in different STB platform.

In addition, reducing software dependencies is needed to have the game state living outside the STB environment and find a way to communicate between the two states. Not to have a game model that is only accessible through the specified interface but have a game model that can be accessed beyond the specified interface. This could then be used by modifying the game state.

IV. New Architecture Based on MVC Model

Model-View-Controller ("MVC") is the BluePrints recommended architectural design pattern for interactive applications. MVC organizes an interactive application into three separate modules: one for the application model with its data representation and game logic, the second for views that provide data presentation and user input, and the third for a controller to dispatch requests and control flow. The MVC design pattern provides a host of design benefits and separates design concerns (data persistence and behavior, presentation, and control), decreasing code duplication, centralizing control, and making the application more easily modifiable and portable [6].

In the STB game architecture design, which can cross and port across different STB platform, the MVC model is recommended, in witch multiple views can exist for the same model, any modification carried out by one of the views is visible to other views [10]. According MVC architectural design pattern, we divided the STB game architecture into three subsystems: the game space, the adapters, and the STB environment. The game space consists of game state, game model, and behavior model. The game space has components like Application Programming Interface (API), scripting interpreter, sockets, and persistent storage. These are used to manage the game and communicate it to the STB environment. Figure 2 shows our new architecture model.

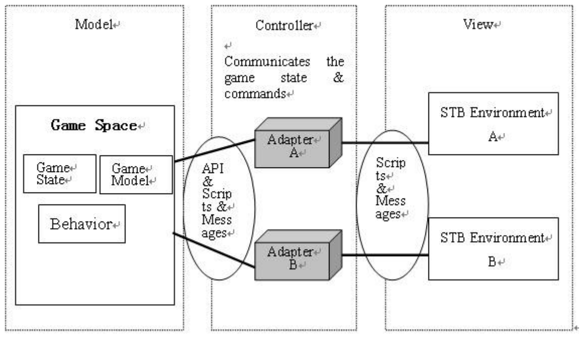

Figure 2. MVC pattern architecture for STB game platform

In fact, our architecture does not follow the MVC pattern strictly as it breaks the direct link between the view and the model and only allows communication through the controller. This decision was deliberate to remove one of the liabilities of using MVC which results in close coupling between the views and the model [7]. Therefore the approach can be considered as a variant of MVC. The corresponding elements to the game state objects still have to be added to the environment and this is done through the level editor.

As the STB games greatly rely on the existing Internet for data transmission, the problems such as network transmission delay, mistakes occurred and so on would be another key question. With consideration of the long network latency taking a serious impact on the network game, it is critical to solve the game state synchronization between the game and game space. In the architecture, we use messaging as a communication mechanism between the game space and the STB environment via the adapters. The messaging role is to allow synchronization of the two game states and allow for actions triggered by the behavior to be carried out to the STB environment. The messaging mechanism used is synchronous which aims at limiting the delay that is caused by the synchronous mechanism [8]. The messages arriving from the STB environment are in a pre-determined format whereas the messages delivered to the STB environment. are formatted in its scripting language. If a specialized behavior model is used messaging can be used to communicate between it and the game space. The format of the messages is dependent on the behavior model. For example, a communication may begin with the STB environment. sending the updates to the adapter. The adapter converts them into scripts or direct API calls which are then used to update the game space. When the game space needs to communicate with the STB environment, it notifies the adapter of the changes that need to be communicated. The adapter formats these into the scripting language and sends them to be executed. The separation and the communication mechanisms allow the STB game to exist independently of the STB environment. The effect this has on portability is that when migrating to a new STB environment, the elements in the game space (i.e., the game state, object model, and game logic) can stay intact. Contrasting this to migrating a game developed using the typical game development approach, which often require all three elements to be created again.

Comparing our approach with the typical game development approach, the first difference is the creation of the game objects, which is split over the STB environment and the game space due to the two types of game objects. The first type is the game objects that have to have representations inside the STB environment to provide visual representations. These require real-time processing in the STB environment, and it is impractical to communicate every frame from the game space to the STB environment. Therefore, these objects have to be created in the STB environment as well as the game space and only updates are communicated. The second type of game objects is the ones that do not have representations inside the STB environment, such as the action, interaction, and reaction objects. These objects can be created in the game space only. The object model creation is similarly split over the STB environment and the game space. The second difference is creating the game logic in the game space rather than the STB environment. The third difference is creating the adapter which handles the communication between the game space and the STB environment.

V. PERFORMANCE EVALUATION

Ideally, the efficient porting game must fulfill the following requirements [9]. First it should be scalable, once the core game is fully developed; porting to a new STB environment shall not be an extremely complicated and time consuming. Second, it should provide a good level of maintainability: new features or bug fixes shall be easily replicated through all different versions of the game; Lastly, It should be adaptive, so developers can extract the most out of the particular features of each STB device.

In our architectural evaluation, we mainly focus on attributes: portability and performance. By using unstructured methods, we made an experimental performance analysis on the STB game portability and performance test. Portability is ranked with high for importance because it is the main quality attribute for the success of the architecture. Our evaluation is to port one game formerly developed on the one type of low-end STB produced by China JIUZHOU manufacturer to the HUAWEI DTV-STB environment without modifying the software core contents and being constrained to modifying the adapter. It found that the separation using the MVC pattern allows for multiple views for the same model. However, the evaluation found that portability could be undermined if the STB environment does not fully expose the required functionality through scripting since the adapter relies on scripting for communication.

Aim of Our game performance analysis was to find the average reduction in frames per-second (fps) due to the use of MVC model. To get a performance indicator, two performance tests were run to contrast the overheads of a game developed with the typical development approach to one developed using our new model. The performance overheads measured were: fps, and network. The average reduction in fps was 11% when following the MVC approach as showed in Figure 3. From the figure, we can see that the average fps reduction is relatively large for a simple STB game. However for a relatively large game (such as code size is more then 10 million), the fps reduction was only about 6%. Of course, only through the above simple tests can not fully explain the problems. In future, the more tests would be needed to be performed to get a better indication of how this reduction will scale with the game size. The evaluation revealed that the data integrity across the different game states was at risk. This is due to the delays that might occur because of the separation as a result of the use of the MVC pattern which add an overhead for exchanging information. Initial tests revealed no problems, but further tests are required before this can be established with certainty. In addition, there is a danger if the message load increases that the game space becomes the bottleneck in the architecture.

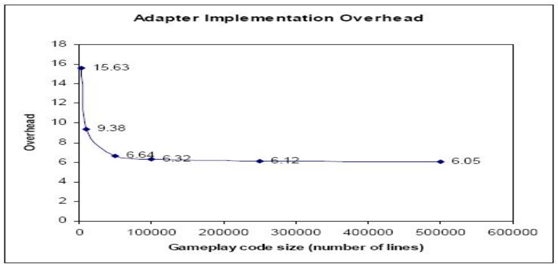

Figure 3. Comparison of MVC-bsed game code numbers and cost

VI. Conclusions

The thesis work has investigated the issue of portability in STB game. We first make the research on the way STB games are developed and identified the problems of the tight coupling that exists which restrict the ability to migrate the STB game to different STB environments. A solution was proposed to address the problems identified and MVC architecture was proposed. The aims are:

1) Improving STB game portability and modifiability, saving developing time of STB game and provide an alternative STB game development route that could be put into practice straightaway. Also can extend the game platform function easily according to future development of game developing technology and digital STB technology;

2) Reducing STB game dependence on the specialized STB environment and realize game logic standardization of and transition standardization between different game logic based on the different STB environment.

The conclusion from this thesis is that when developing a game for STB devices, portability becomes even more of an issue than when developing for the PC market. This is true in the sense that portability becomes more important, since the STB market is extremely diverse and porting applications is a must to stay competitive, but also in the sense that it becomes a lot harder to obtain, because of performance also being of much greater importance in such as STB low-resource environments.

One of the weaknesses in this thesis project is that the experimental results we made in Section V is just for specific STB environments and may have less applicable in other situations due to the scope of this evaluation. Therefore, further work with similar goals, but making more different code size game test on more different STB environments, would create results that could be used in a wider range of situations. In addition, our method complicates game data process and increase network transmitting spending because a lot of data are required to exchange in MVC design model, which also will be our future work for further research.

Acknowledgment

Supported by the ZheJiang Scientific and Technological Project (Grant No.2007C21G2080053).

References

[1] ISO/IEC 13818-1. Information technology – Generic coding of moving pictures and associated audio – System.JTC1/SC29/WG11, November 1994

[2] Yuanzhe (Michael) Cai, Set-top Box Gaming:From Wasteland to Promised Land. www.parksassociates.com/free_data/parks-SettopGaming.pdf, 2010

[3] V. Alves et al. Comparative analysis of porting strategies in j2me games. In Proc. of the 21st IEEE International Conference on Software Maintenance (ICSM 05), pages 123-132,2005.

[4] Yongjun Zhang, A Java 3D Framework for Digital Television Set-top Box.www.tkk.fi/Units/IDC/brocom/sub/terminal/master/Java_stb.pdf

[5] Tay, V. 2005. Massively Multiplayer Online Game (MMOG) — a Proposed Approach for Military Applications. International Conference on Cyberworlds,IEEE, 396-400

[6] Inderjeet Singh, Beth Stearns Designing Enterprise Applications with the J2EE Platform, Second Edition java.sun.com/blueprints/guidelines/designing _application/book.pdf

[7] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M. “Patter-Oriented Software Architecture: A System of Patterns” Volume 1, 1996, John Wiley and Sons, ISBN: 0471958697

[8] Trowbridge, D., Roxburgh, U., Hohpe, G., Manolescu, D., and Nadhan, E. “Integration Patterns: Patterns & Practices”, Microsoft Press, 2004, ISBN: 073561850X

[9] Tarcisio Camara, Rodrigo Lima, Rangner Guimaraes, Alexandre Damasceno. Massive mobile games porting: Meantime study case. In Brazilian Symposium on Computer Games and Digital Entertainment - Computing track, Recife, Brazil, 2006.

[10] HUANG Jun, CHEN Guangping, WANG Jun. Research on portabil ity of Network games based on MVC pattern. Journal of China Jiliang University, Vol . 19 No. 4 Dec, pages 347-350,2008. [In Chinses].