Game Architecture for Business Simulation Games in XNA: The VTeam case

Bruno Jamir e Silva       Raphael L. B. de Barros       Diogo C. Lemos       Danielle R. D. da Silva
Patricia R. Tedesco       Geber L. Ramalho

Universidade Federal de Pernambuco

Abstract

Business simulation is a kind of video game aimed at reproducing a real business environment where the player plays the role of a manager, making decisions in order to make the business prosper. However, in several related researches about game development and architecture definition, business simulation games are rarely mentioned. Thus, this paper presents a core architecture for business simulation games, based on their common requirements and interface features. The proposed architecture was applied in a real game development project in order to validate it.

Keywords: Business, Simulation, Game, XNA, Architecture

Authors’ contact: brunojamir@gmail.com, rlbbrb, dcl, drds, pcart, glr@cin.ufpe.br

1. Introduction

Video games are no longer an exclusive hobby for children and teenagers. Every year, games interest more adults [Esa 2009]. One particular kind of game that attracts this new public is business simulation games. These are designed to simulate a real business environment where the player plays the role of the manager, making decisions in order to make the business prosper [Laramée 2002]. To attract this public, these games are designed to keep the familiarity with real world tools, such as management software and information systems.

Although there are several related works about game development process and architecture definition, business simulation games are rarely mentioned. This paper presents a core architecture for business simulation games, based on common requirements and interface features present in this kind of game. The proposed architecture main goal is to simplify the development process of business simulation games, providing scalable and efficient core features.

To validate the proposed architecture, a business simulation game called VTeam was developed over it [VTeam 2009]. This game simulates a software development environment, and the player plays the role of project manager. The player can attribute tasks, resolve conflicts between the team members, give some feedback to development team, etc. To provide a believable simulation, the game characters were implemented as synthetic actors [Silva 2009].

This paper is organized as follows. Section 2 explains the main business simulation games features. Section 3 shows the proposed architecture for business simulation games. Section 4 details the application of the proposed architecture in the development of a real game. Finally, section 5 our conclusions and suggestions for further work.

2. Business Simulation Games

Business simulation games focus on economic process management using a simplification of the business environment. These games can be described as construction simulations, when they focus on building elements to achieve a goal, or management simulation, when the game elements are already set and the actions are focused on resource management [Laramée 2002].

Business simulation games often share similar features, related to how the information is displayed and how the way the player interacts. Thus, for the development of business simulation games the following requirements must be met:

[R1] Mouse based actions: since all actions of the game are triggered by the mouse cursor, the player should be able to perform actions of selection, click and drag & drop intuitively with an immediate visual response.

[R2] Multiple (simultaneous) screens: the large number of actions available to the user and the need to display different groups of information requires an intelligent grouping of interface items. To address this requirement, the game must provide the player with several sub-screens that can be accessed from the main game screen.

[R3] Several interface components: the player controls the system through interface items arranged in the screens, such as buttons, checkboxes and tab panels. Thus a standardization of these elements is necessary in order to both simplify the development and maintain the usability cohesion of the system as a whole.
[R4] Base Simulation: in such games a large amount of calculation is processed in the background to simulate a real environment. Such calculations can be related to economics, statistics, and artificial intelligence formulas. This should be executed in parallel to the interface processes, in order to avoid jeopardizing the visual performance.

3. XNA Architecture for Business Simulation Games

Based on business simulation games' features and the architectural pattern offered by XNA Framework starter kits, a general architecture for business simulation games will be proposed in this section. Such architecture consists of 3 main modules: the graphics module, the services module, and the content loading module. The first component is the responsible for displaying the game elements to player. In other words, it controls the screen presentation, and the drawing of its visible elements. It is composed by screens, user interface components (like buttons, text boxes, etc.) and simulation views. These are specific visualizations of the game’s simulated world.

The services module is responsible for providing resources for the game, helping in its implementation. The components of the services module were called managers. These managers are well-defined and almost independent modules. Thus, these managers respond to specific requests of the game’s screens and their components. Examples of managers include: audio manager, input manager, and simulation manager, that simulate the game rules and artificial intelligence.

Finally, the content loading module is responsible for loading the resources that will be used in the game. It is responsible for loading image files, sound files and XML files for game configurations. The content loading module is composed by well-defined responsibility elements. Thus, an element may be the loader of character’s animations, while another is responsible for loading the map that defines the character’s positions in the world of game, for example.

In Figure 1, it is possible to correlate the created architecture with the requirements of the simulation games, mentioned in section 2 of this paper. Therefore, the way by which the requirements were covered by the architecture is explained below:

[R1] Mouse based actions: the interaction in a business simulation game is mostly done with the mouse. Therefore, the way the input is treated should be simple and scalable. Instead of pooling, we used an approach with events and delegates, available in the .NET Framework. In this approach, the components do not make consecutive requests for an input change. Instead they are registered as listeners of input events, raised by a sender class. For example, if the player selects the “New Game” option in the main screen, the mouse click action will be captured by the input manager and passed to the active screen. Upon receiving this event, the screen will decide what action to take.

[R2] Multiple (simultaneous) screens: to control frequent transitions between the game screens a Screen Manager was developed. This class is responsible for the resource allocation (and deallocation) of involved screens, and for controlling the effects of screen transitions.

[R3] Different interface components: business simulation games have an interface similar to general-purpose applications. Therefore, it is necessary to provide user interface components such as buttons, menus, and text boxes. To meet this requirement, the proposed architecture has these graphical components at the graphics module. However, there are some graphical objects that cannot be represented by simple interface components (e.g. a window of the world filled with characters). In this architecture these components will be considered as Simulation Views.

[R4] Base Simulation: to simulate the game rules, some managers can be developed in the service module. For example, if the game aims to simulate financial transactions, it is possible to create a manager that simulates the behavior of a stock exchange.

These requirements are found in most business simulation games. In order to validate the consistency of the proposed architecture, we have developed the
VTeam game, which simulates a software development environment. In this game, the player plays the role of team manager. The details concerning the architecture created for VTeam will be explained in the next section.

4. Business Simulation Game Architecture: The VTeam Case

A good example of use of Business Simulation games is in serious games. These games have been adopted in various universities as a support tool for learning business management. For instance, the games Virtual-U [Virtual University 2009] and Capitalism [Capitalism 1996] are games adopted at MBA courses to learn the complex process to manage and to build a business. This is also proposed by the Virtual Team Project that has been developing the VTeam business game.

5.1 The Game VTeam

Virtual Team (VTeam, in short) is a business simulation game developed with the purpose of training project managers, focusing mainly on people management and human resources. This game is an extension of the prototype developed previously by the Smartsim project [Smartsim 2006] including new features and business conflict scenarios to be learned by the player. VTeam's goal is to provide a greater experience of organizational, methodological, cultural and personal processes for the manager being trained.

This game offers to both instructor and player a richer experimentation scenario when compared to traditional methods and simulators used to teach those concepts. Thus, in order to provide a good simulation and give the player a proper learning experience, the VTeam game flow covers the following phases:

- **Planning**: a project plan is presented to the player. This plan is composed by a set of tasks, an initial budget, and some goals to be achieved. The player also has to contract the team selecting the characters to fill in the roles of programmer, architect and analyst.

- **Start up**: at this phase the player can interfere directly with the project through the environment were the characters interact with each other. The player can provide feedback, give rewards and assign project tasks for each of the characters from this phase on.

- **Development**: the player must follow the development of the tasks and solve conflicts that may occur between the characters. He has also to keep the client informed of the project status and interact with the team in order to optimize their job.

- **Finish**: once all the tasks are done and the goal is achieved (or the project time and budget are over), the game ends and a detailed result screen is presented, providing information allowing both the trainer and the trainee to reflect over the game performance.

Based on the presented requirements (in section 2) it was possible to apply the proposed architecture in VTeam game. Some details of this instantiation will be presented in next sections.

5.2 VTeam Game Screens

Each of the screens contains a set of user interface components grouped on a panel. Each screen is responsible for drawing and controlling the input of these components. Besides, the screens control the navigation between themselves requesting the `ScreenManager` to change the active current screen. Figure 2 shows some interface components at the character status game screen panel.

![Figure 2: Interface components on a game screen panel](Image)

5.3 World View

Besides the user interface components, the Ingame Screen (Figure 3) shows an area portraying the scene: a software development office where the characters interact. This world view is too complex to be a simple interface component, so it is implemented as a class itself.

![Figure 3: Ingame screen](Image)

The Ingame Screen is composed by several Objects Views which display each of the objects in the scene, like the characters, computers, furniture and the background. These are updated and drawn by the World View. The characters are implemented as
5.4 Simulation

In order to create a believable software development environment, the characters of the game should have distinct skills and believable behaviors. In order to implement this feature, the characters are represented by Synthetic Actors: intelligent agents provided of emotions, personality and beliefs [Silva 2009]. These agents are focused on creating an illusion of life providing a good credibility to the player, rather than the usual intelligent agents approaches which aim at problem solving.

The logic behind these synthetic actors is controlled by the AI Manager, which constantly updates the state of the characters based on the player actions, the world’s current state and the inference engine. The message exchange between game and inference engine is shown on Figure 4.

![Figure 4: Communication with the inference engine](image)

5.4 Resource Loading

The VTTeam is a game directed to train project manager, and its sessions are usually conducted by an experienced instructor. Then, some customizations must be allowed. To provide this functionality, some features of the game can be changed by the user, via external tools such as the Level Editor. This tool is able to modify the XML file that describes game level, characters’ personality and skills, etc.

6. Conclusion

Business simulation games have particular features that differ from the majority of conventional games. These features demand a customized architecture covering the requirements of this kind of game. Therefore, this paper provides to game independent developers an efficient and scalable architecture to develop business simulation games in XNA. This architecture was tested in a real game project, called VTTeam.

However, the architecture should be validated in other projects, since the VTTeam project does not cover all of its aspects. For instance, in this project there is only one view for each model, since there is a single view of the world view. In other words, the only visualization of the game’s world is the room, where the characters work in the project. Thus, in a VTTeam’s extension another visualization could be available. For example, the project meeting room where the manager (player) could see the characters interactions and facial expressions.

Acknowledgements

First of all the authors would like to thank the VTTeam development team and project coordinators for all the effort and support. Also thanks to the project partners and sponsors: FINEP, Jynx Playware, Valença & Associados, Qualiti, CNPq, FADE-UFPE and Clín-UFPF.

References


