THE SQUARE DANCE FRAMEWORK - A FRAMEWORK APPROACH FOR MODULE-BASED SERIOUS GAMES

Alexander Waldmann, Juan Haladjian, Damir Ismailović, Bernd Brügge
Technische Universität München, Munich, Germany
{waldmann, juan.haladjian, damir.ismailovic, bernd.bruegge}@cs.tum.edu

ABSTRACT

In educational serious games, learning is encouraged through gaming experience. A major goal in modern serious games is to dynamically adapt the learning content to the player, in order to enrich the learning experience. This paper presents a framework for the development of module-based games, used throughout a project at the Technical University Munich for the development of a serious game composed of multiple mini-games. It defines and attempts to bridge the conceptual gap between classical game and serious game development. For this purpose, it employs elements from the game development domain as well as state of the art technologies for feedback through adaptivity.

KEYWORDS

Serious Game Engineering, Adaptivity, Framework

1. INTRODUCTION

Serious games allow the players to take advantage of the fun experienced in a game and learn through their experience.

Many serious games consist of different modules, which can be scenarios, levels, etc. In order to optimize the game's development time, the different modules are split among the developers, who work on each module independently. In the end, the different modules need to be integrated, which often requires a big effort.

Well known game development frameworks like *Unity*, *Gamemaker* or *TangoPhysics* (Haladjian 2012) ease the development task by providing ready-to-use game elements, like images, sounds, particle systems, behaviors, etc. However, they do not deal with higher-level functionality like the ability to define a game's structure.

We introduce a framework for defining a game's structure as a collection of modules. Modules may be any part of a game, typically a scene or a level. Our framework lets users define modules' structure and dependencies between different modules. Different modules can then be developed independently and integrated afterwards. Module integration conflicts are reduced thanks to a common module structure, enforced to every module through our framework.

The purpose of this paper is twofold: we first present a case study where we analyze the development process of a module-based serious game, which highlights the need for our framework. We then introduce Square Dance, a framework for the collaborative creation of module-based serious games.

2. BACKGROUND

There is no single definition for the term "Serious Games". Different definitions stress for instance pedagogical or technical aspects. As (Susi 2007) state, "Even a brief survey of the literature soon reveals that there seems to be as many definitions available as there are actors involved". Due to this lack of one unambiguous definition of the problem domain, the understanding of serious games in the context of this project relied on the conceptual model of (Yusoff 2009). This model illustrates the divergence of concepts

(game concepts vs. pedagogical concepts) in the domain of serious games. (Yusoff 2009) identify elements from the game application domain as well as the teaching domain. The teaching domain is simplified as „Capability, instructional content, intended learning outcomes, learning activity and reflection". The gaming domain is reduced to „Game mechanics, games genre, game achievement and game attributes". Although the relationship between these concepts is described in detail, the model lacks an approach to overcome the conceptual gap between the two fields. This gap has to be closed in order to implement a serious game.

3. CASE STUDY

We developed an educational serious game in a period of two years of time. The purpose of the game is teaching mathematics (addition, subtraction, counting, number drawing) to pre-school children. The game consisted of six different levels or mini-games. Each mini-game was given to different groups of students as part of a lab course at the Technical University of Munich. The project leader then adapted the different mini-games, integrated them into a single application, and delivered the resulting game.

The integration of the mini-games into one product was difficult to handle. The project allowed us to identify two major issues. The first one is related to the quality of the game created. The resulting project had a high amount of repeated non-consistent functionality. A good example was the game's menu. Each mini-game had it's own menu, implemented in different ways. Adaptivity represented another issue. The way in which the adaptivity framework was used by different students, lead to the creation of mini-games with an heterogeneous difficulty level. Some games were too hard for it's target users, others were too easy. These issues were the most time consuming, as different parts of every mini-game had to be reimplemented.

The second issue is related to the working methodology followed. Communication often caused time overhead, as the usage of frameworks, best practices and mini-game conventions had to be explained to every team member.

4. FRAMEWORK

The framework is divided in two components. The Director and the Module. The first one takes care of managing the control flow. The second one is responsible for the high-level structure of games (so called mini-games, or levels). All modules register at the Director by supplying themselves via the registerModule() method. To create a module a developer constructs a subclass of the Module class (see Figure 1). The Module class supplies the subclasses with a set of hook methods, which get triggered if the director decides to hand over control.

The Director decides if and which module can be started and if the control can be handed over to it if certain requirements are fullfilled (e.g. a score has been reached, see Figure 1).

Such a requirement has to be a subclass of the Requirement class (see Figure 1) and implement the hook method "requirementFullfilled". Through this workflow developers are capable of controlling when their modules get invoked.

The creation of modules is eased by a graphics framework, in the concrete implementation Cocos2D that focuses on animations and movements of objects.

In addition to the graphics library, an Overlay class (see Figure 1) is introduced to create a consistent style within the independently developed modules. This Overlay class eases the creation of controls and optical elements that are used throughout all other modules in the same way. Abstract factories supply each developer with the same set of controls throughout all modules. This approach aims to increase the usability and the conceptual model in the mind of the learner.

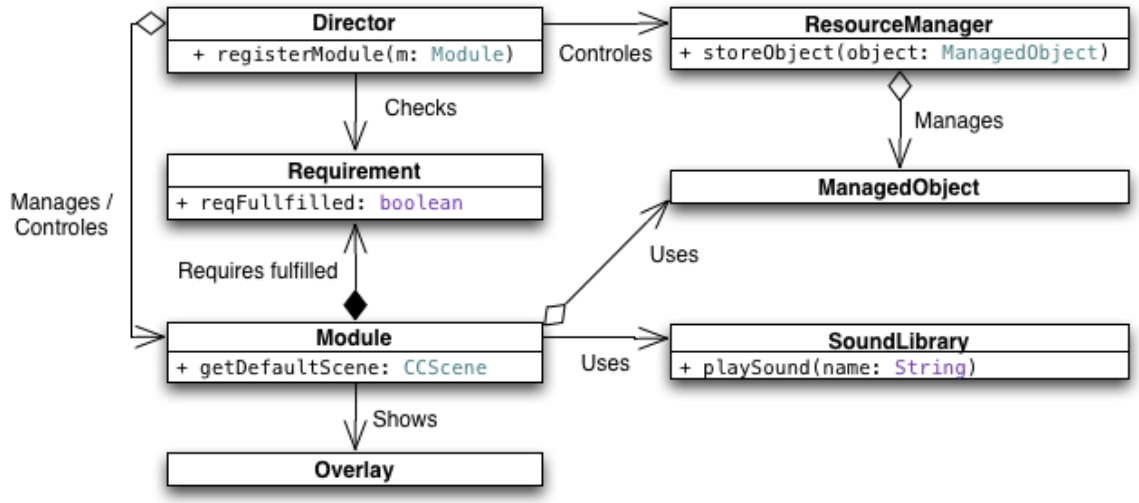

Figure 1: The 7 base classes for the Square Dance Framework.

The ManagedObject class (see Figure 1) can be used to create very simple objects (which appear in the game world) and share them with other modules. It also conforms to the interfaces described in (Ismailovic 2011) for the adaptivity of the game-world and allows the game to change behavior, according to the players success. A game object can be registered and saved in a resource manager, which persists data if needed.
To use this feature, developers simply subclass the ManagedObject class and call a store method in the ResourceManager class (see Figure 1). This concept is similar to the concept in Apples CoreData.
Through this approach the Resource Manager allows the sharing of Managed Objects between modules to create for example a shared inventory.

4.1 Architecture

The architecture is based on strict layering, yet it is not enforced in the framework. The dependencies of the modules on the graphical framework, as well as on the adaptivity framework are shown in Figure 2.

The upper and bottom layer are developed as part of the framework. The middle layer is implemented as a set of mini-games, their corresponding requirements and objects. The upper layer shows the Director. This component appears as the managing component for the modules.

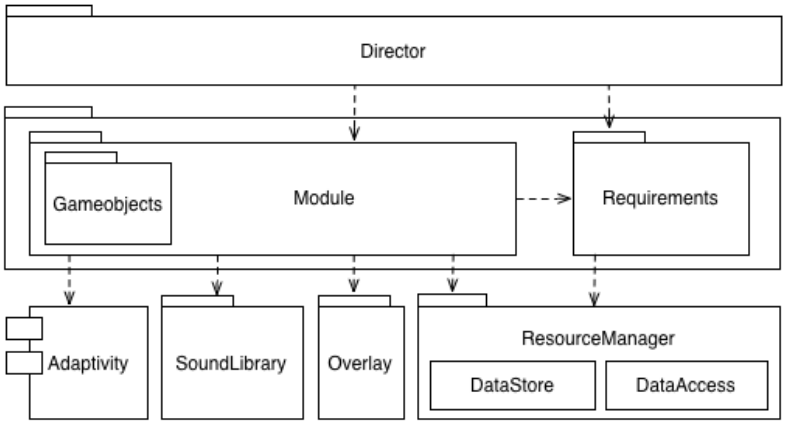

Figure 2: Components and architecture of the framework

It uses the inversion of control principal (the framework calls the modules, not the modules the framework) only to some degree and passes the control completely to the module if possible. This promotes the development of the modules, as there are not numerous hook methods needed in the module.

5. CONCLUSIONS

The development of serious games is an active and rapidly growing research area. With the Square Dance Framework, we intend to make the development of such games easier.

Different developers either build modules based on different structures, which makes their integration difficult, or they invest a considerable effort in communication in order to make the different modules compatible. Our framework enforces the usage of common modules' interfaces, making integration of different modules easier.

Adaptive games guide the users with individual feedback in their decision making process. The adaptivity of the content (difficulty, behavior of the environment) is based on the learner's behavior and success in completing a task. The framework delegates this problem to the adaptivity framework (Ismailović *2011*). The framework needed to be adapted to the game would tend to use the adaptivity framework in different ways, leading to inconsistencies among the different modules. Our framework provided a higher-level common interface to the adaptivity framework, which enforced the way in which the framework should be used by each module.

When faced with the design of a game the structure of objects inside the game world can become complex. Consequently a basic object composition was introduced, allowing the simple creation of objects in the game and the sharing between the modules later on (e.g. a trophy or an inventory item in the game) at runtime. Those objects are the primary connection between the adaptivity framework and the game. They connect the game world with the adaptivity algorithms.

To allow the use of best practices, such as how to efficiently handle animations and movements of graphical objects, a number of well-known libraries were introduced. In this project, the „cocos2d" library was used, yet the concept of the framework is not bound to this game framework.

In combination with frameworks like TangoPhysics, the Square Dance framework allows the development of full-blown serious games from a high level of abstraction.

Still, research in this field needs to be intensified. In the future, different game design patterns will be supported by our framework, not only the module-based one.

REFERENCES

Cocos2D, 2010. *Learn iPhone and iPad cocos2d Game Development*,
Apress, USA, New York City

Haladjian J. et al, 2012. A quick prototyping framework for adaptive serious games with 2D physics on mobile touch devices. *IADIS Mobile Learning 2012 (ML 2012)*

Ismailović D., 2011. *Adaptivity in Serious Games*. PhD thesis, Chair for Applied Software Engineering, Technische Universität München, Munich.

Susi T. et al., 2007. Serious Games - An Overview, Technical Report HS-IKI-TR-07-001, School of Humanities and Informatics, University of Skövde, Sweden

Yusoff A. et al, July 2009. A Conceptual Framework for Serious Games.
Advanced Learning Technologies, 2009. ICALT 2009. Ninth IEEE International Conference. Riga, Latvia, pages 21–23.