TDT4240 Software Architecture

Architectural Description Document

Group 17 : "XNA videogame focus on modifiability"

Marius Greve Hagen
David Rozas Domingo
Maria Fernandez-Rodriguez

1. Index

2. Introduction

This document deals with the Architectural Description of the project “SuperPang videogame”. It starts mentioning the architectural drivers of the project, secondly a description of the stakeholders and concerns is presented followed by a discussion about why a logical architectural viewpoint has been chosen.

The quality tactics we will carry out to achieve the previously mentioned quality requirements are discussed, as well as the limitations we found related to architectural patterns because of the use of XNA. Finally, a brief description of the logical view is provided.

3. Architectural Drivers

In order to identify our architectural drivers, first of all, we should state our main business goals:

3.1 It is desirable that the video game consists of several stages, each of them with different levels of complexity. Complexity can be defined in terms of speed of balls, time to reach the final goal, interaction with more items, allow changes of weapons and any other change which can influence slightly the logic of the program, but not changes the nucleus of the main logic.

3.2 It is expected to include the possibility of allow a two players local session. An online session is also a possible future line of development.

3.3 A player should be able to modify some controls as choosing the keys for movement commands, level of complexity, choose the character to play as through a visual menu, providing as clear as possible interface.

3.4 Characters and items taking part into the game should cover different profiles in order to offer new challenges to the end user.

3.5 Errors in the logic of the program must be easily differentiated from the ones in the underlying infrastructure.

3.6 Constant and platform-independent frame rate.

Relationships established between our architectural drivers and quality attributes are as follows:

- Modifiability (1,2,3).
- Testability (5).
- Usability (3, 4).
- Performance (6).

4. Stakeholders and Concerns

The stakeholders that are taking part in the system are the following ones:

4.1 End user (U)

Mainly, U looks for entertainment, capability to adapt the application to his/her own abilities and tastes. After learning curve, new and exciting possibilities must be provided.

U is also interested in an intuitive and friendly graphical user interface and in an efficient documentation to solve problems, as well as in performance.

4.2 Architect and designers of constituent parts

To resolve resource contention and establish performance and other kinds of runtime resource consumption budgets.

As designers, we have to deal with XNA performance management, find out its limitations and integrate our game functionality taking this into account.

We also have to divide the main functionality of the system in different modules, assigning different responsibilities and defining the communication between the modules.

4.3 Implementors

To provide inviolable constraints (plus exploitable freedoms) on downstream development activities.

As implementors, we will carry out the implementation of the previous design in the programming language level. It is desirable that the previous stage it is clear and precise enough to execute this task without ambiguities.

4.4 Testers and integrators

To specify the correct black-box behavior of the pieces that must fit together.

As testers, we will try to provide mechanisms to find out if the expected results match the obtained results and carry out problem solving if necessary, at a modular level and in the whole system.

4.5 Maintainers

To reveal areas a prospective change will affect.

As maintainers, we have to find out how new functionalities are affecting the system and carry out the necessary changes. We should also be aware about new reported bugs and look for a effective solution.

5. Selection of Architectural Viewpoint

We have considered only the logical view for the moment. The reason why we made this is it is still quite early to see properly the necessary elements in other views.

In later deliveries, new views will be added. So far, the logical view reflects the model game, its classes and relationships. The target audience of this view involves all the stakeholders but the end user.

We are conscious about some issues as a pesistent storage for the configurations file must be included in another module/layer that will be implemented and reflected in the development view.

A proccess view will be also added in next versions of this document when we study more carefully the logic of the program: the relationships between events, methods to implement and so on.

6. Quality Tactics

6.1 Performance:

For the game to perform at a steady frame-per-second rate or have a predictable performance it's important to manage the events and methods in an efficient manner.

The most important task is to use algorithms that compute as little data as possible to maintain this overall efficiency. We might also introduce concurrency to compute data simultaneously, but it's important to keep the threads to a minimum so they don't take up to much of the resources (given that the concurrency is thread-based).

6.2 Testability:

In order to make it easy to test the game architecture for faults and failure it's important to generate a lot of output to control the estimated computation. This could be done by implementing thorough logging and well-defined interfaces. The interfaces should be separated from the implementation.

It should also be possible to test each component separately to easily determine where problems originate from and then make it easier to deal with.

6.3 Modifiability:

In order to make the controller architecture and implementation easy to change for adding or modifying functionality we should first and foremost anticipate expected changes in the architecture. For it to be easier when the changes arrive the system should be module-based and the module should be generalized and given a semantic coherence. We should also hide some information in means of private fields in the code.We could of course limit the number of features, but since we want our game to be scalable we should use interfaces and inherit from them.

7. Architectural Patterns

The architectural patterns are strongly influenced by the use of XNA since it implements the game loop and filter. We think we will find more patterns about the design during future design meetings.

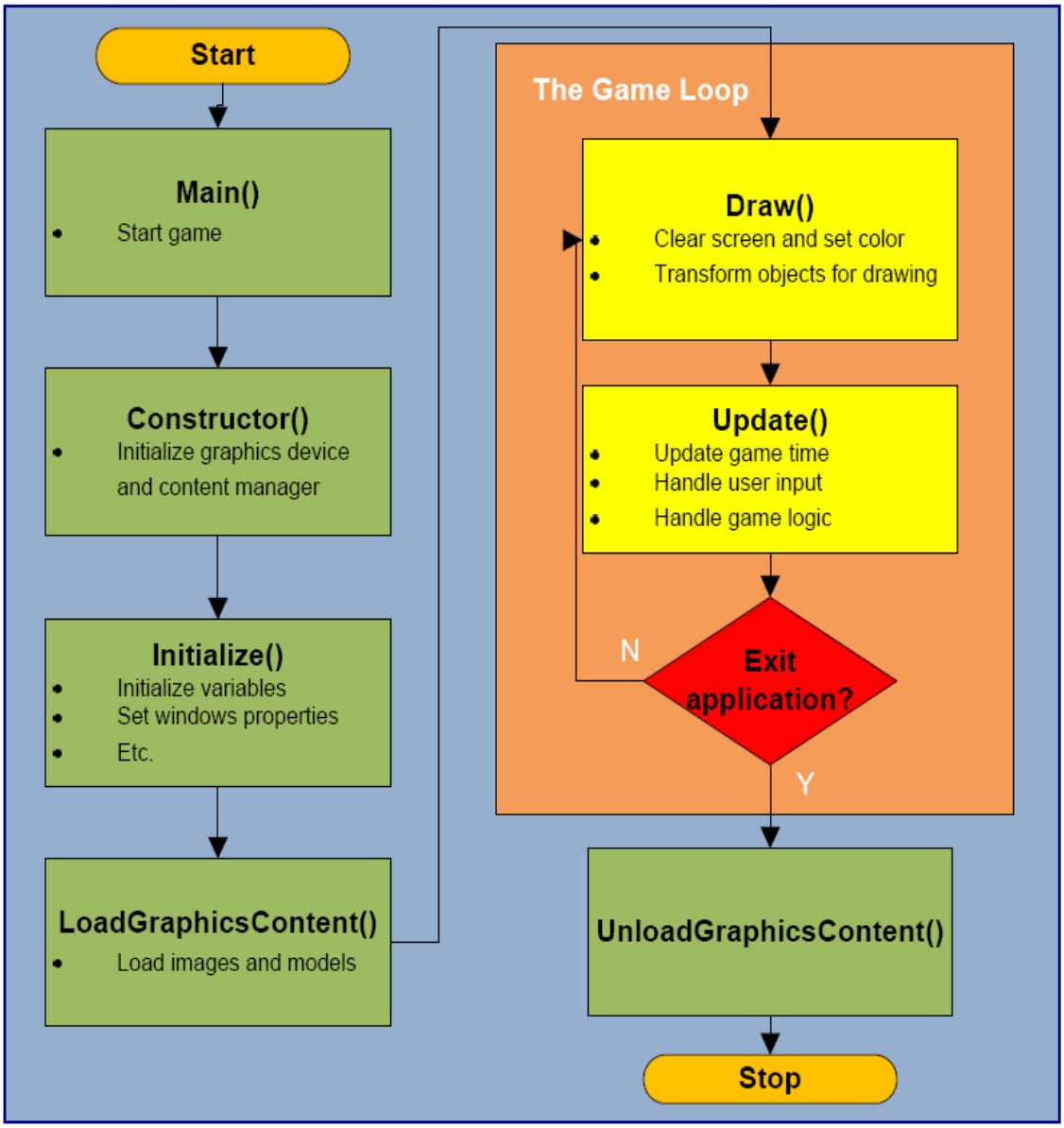

Figure 1: XNA application Model

We have also to study the possible use of XQuest libraries and what this would involve in case we finally decide to use it.

8. Views

Logical view:

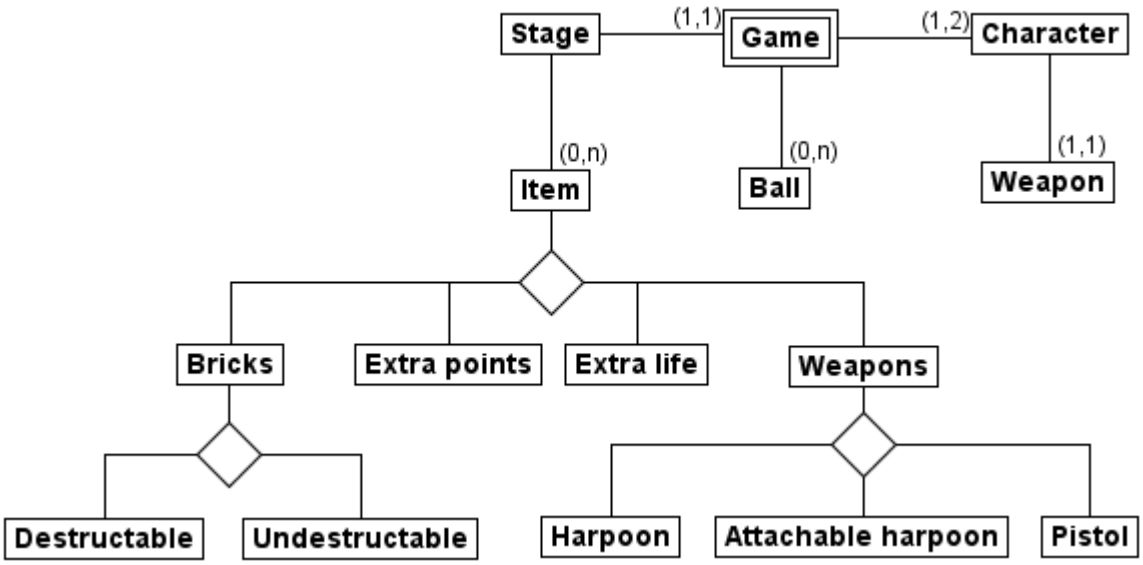

Figure 2: Logic view of the super pang

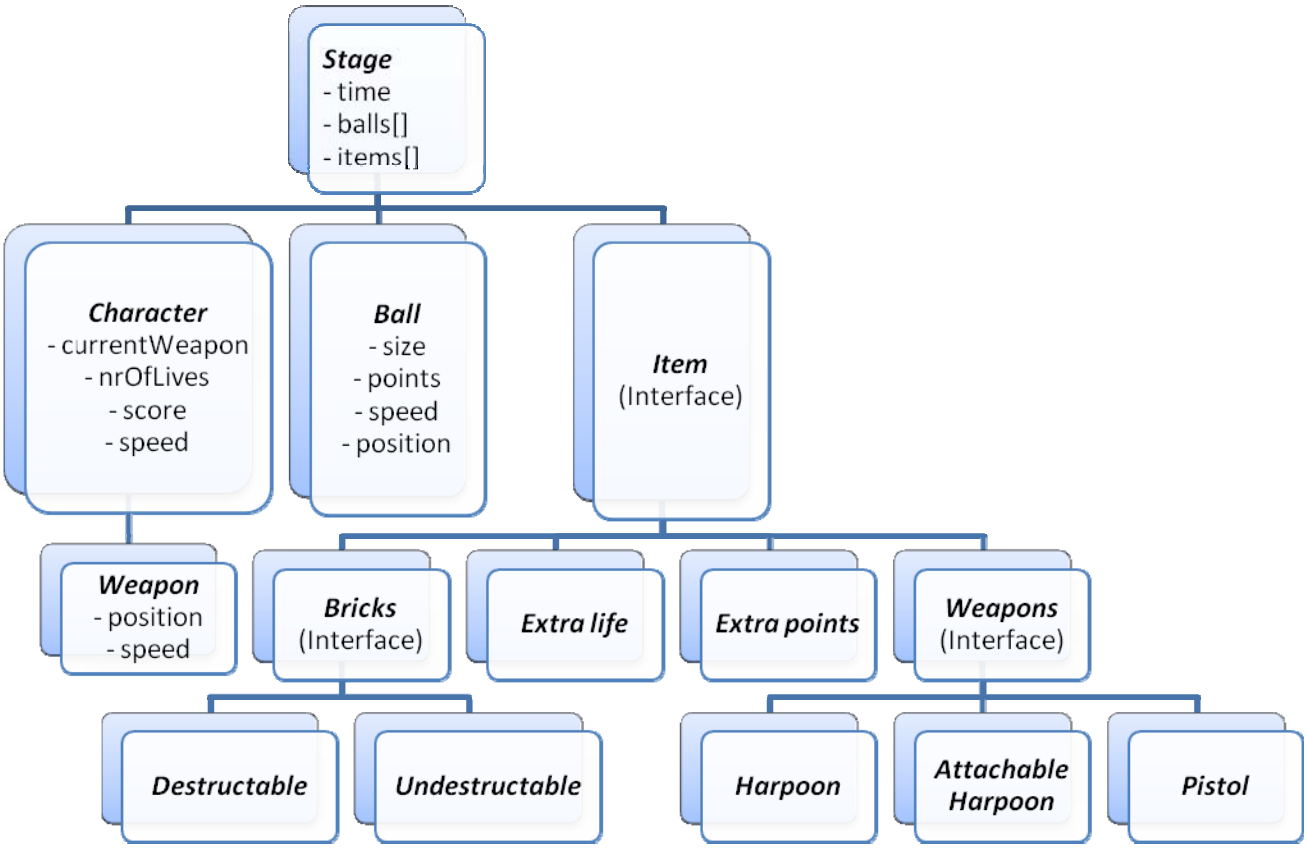

Figure 3: Additional logic view including some attributes

9. References

"Software Architecture in Practice, Second Edition", Len Bass, Paul Clements, Rick Kazman (Tapir).

XNA Intro Slides from TDT4240 course.