Software Architecture

TDT4240

Implementation Document

Group 17 : "XNA videogame focus on modifiability"

Marius Greve Hagen

David Rozas Domingo

Maria Fernandez-Rodriguez

Jarle Lindseth

1 Index

2 Introduction

This document deals with the implementation of the project “SuperPang videogame” with focus on modifiability. A description about how the implementation has covered the mains goal of the design can be found in the following section.

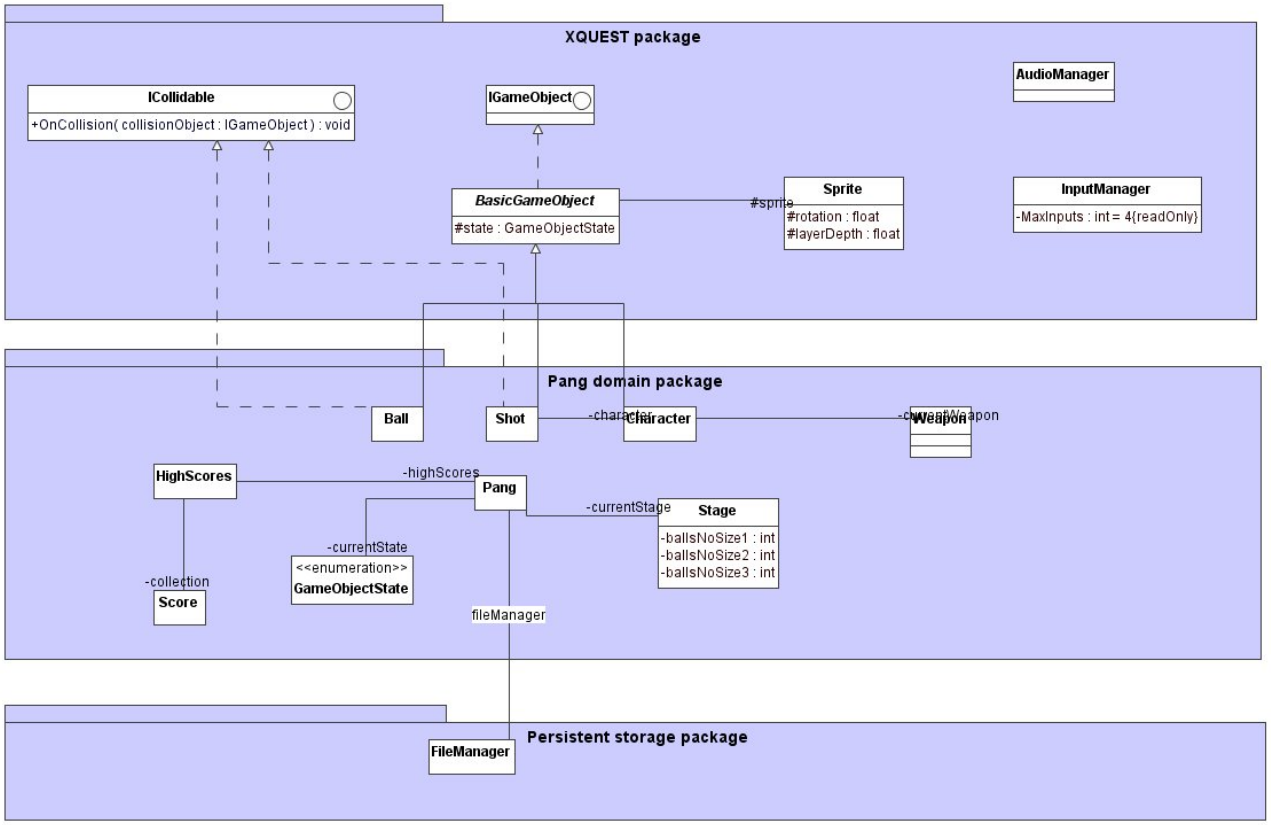
Section four consists of an user manual explaining the content of the main file, how to run the game and how to play with it.

Next, a test report with the most important tests, all of them related with functional and quality requirements.

In the next section a list of inconsistencies with the architecture is showed. The last section refers to the problems, issues and points which have come up during the achievement of this project.

3 Design and implementation details

In order to bound different responsibilities we have decided to dispose our software classes in different levels of abstraction according to the layers pattern.

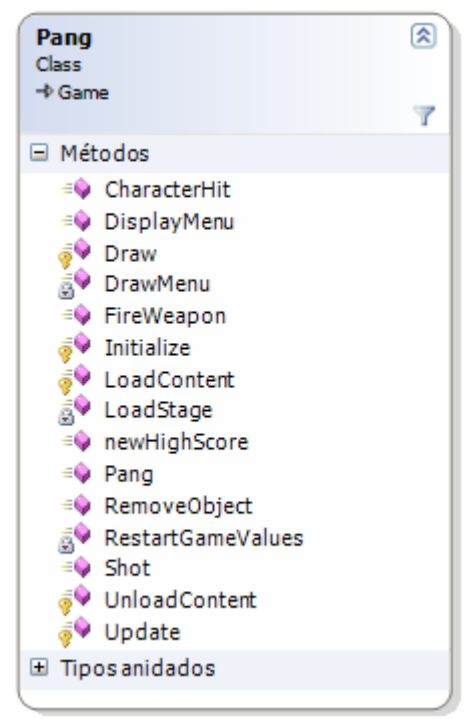


The main logic is residing in *Pang domain package*. It contains eight classes:

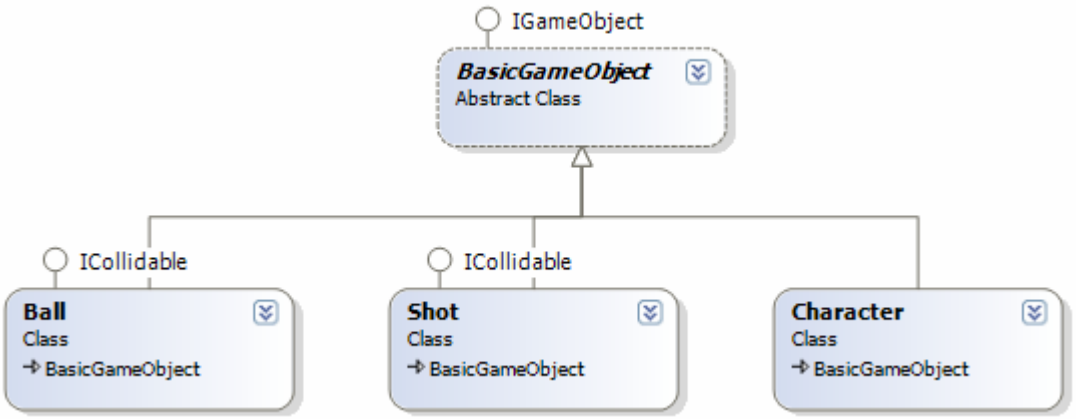
1. Ball
2. Shot
3. HighScores
4. Score
5. Character
6. Weapon
7. Stage
8. Pang

In Pang domain package, the central and main class is *Pang*. This class is in charge of updating properly game states, since when a collision takes place, elements involved in a collision

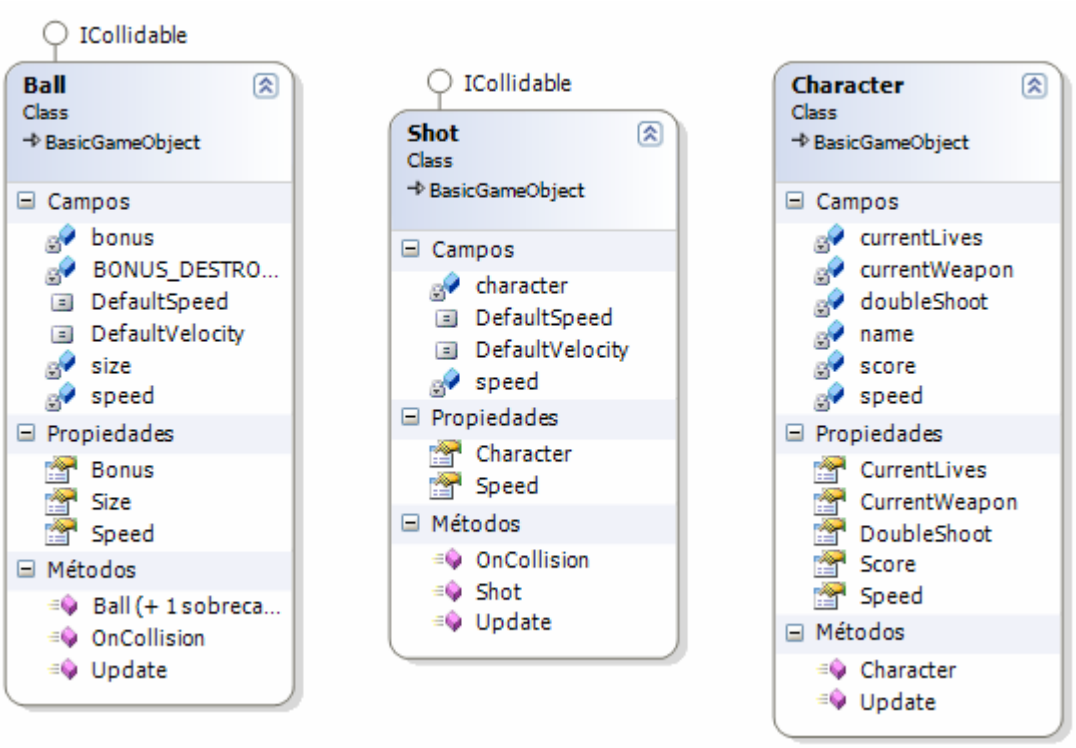
notify *Pang* class. A detailed description of Pang class can be seen in the next figure:



Ball and Shot classes implement ICollidable interface (belonging to XQUEST) and at the same time extend BasicGameObject (XQUEST class) together with Character class. This relation is show in the following figure.

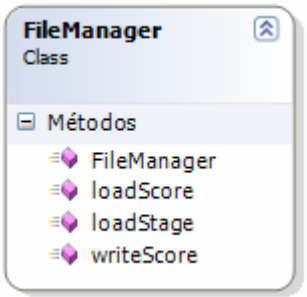


A more detailed description of these three classes is showed in the next figures:

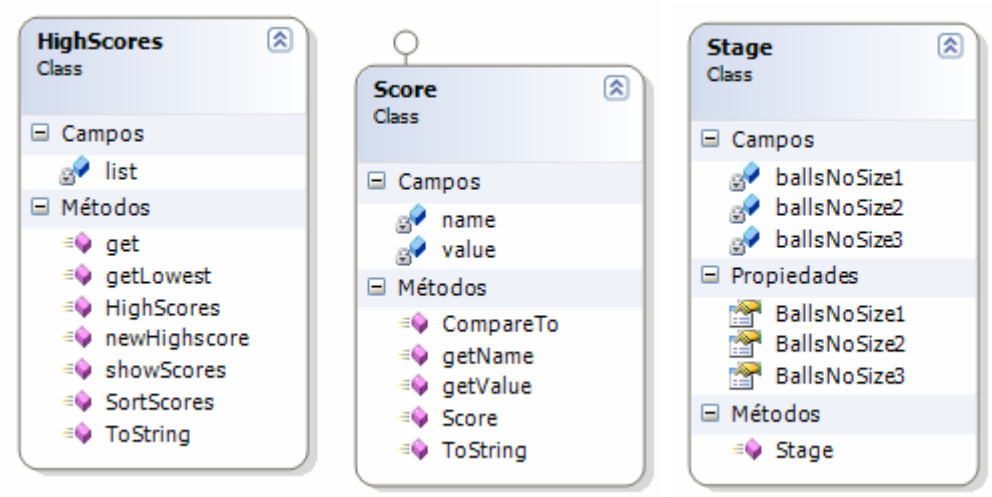


Pang also communicates with *FileManager* class to get and save data from persistent storage and set correct parameters to the *currentStage*. FileManager class resides in the Persistent Storage package.

FileManager class is in charge of store and load from persistent storage to provide to the upper layer configuration settings and data. FileManager is using internally a library called AMS.Profile which provides an API to work with XML documents. This functionality has, undoubtedly, helped us with persistent storage.



This class reads the settings of each stage and provides an Stage class to Pang class. It is also in charge of reading and updating the Higher Scores, which also have their domain model classes, as it is showed accordingly in next figure:


HighScores
Class
Campos
list
Métodos
get
getLowest
HighScores
newHighscore
showScores
SortScores
ToString
Score
Class
Campos
name
value
Métodos
CompareTo
getName
getValue
Score
ToString
Stage
Class
Campos
ballsNoSize1
ballsNoSize2
ballsNoSize3
Propiedades
BallsNoSize1
BallsNoSize2
BallsNoSize3
Métodos
Stage

4 User manual

4.1 What is included in pang.zip?

- exe: Includes the executable version of the game and all the resources necessaries to run it (dll's, images, sounds, configuration files, etc).
- src: Includes the source code of the game, including the file .sln which opens directly the whole solution.
- doc: Includes all the project documentation by phases, in different .pdf files.

4.2 How to run the game:

For running the game, it is only necesary to:

- Uncompress pang.zip
- Execute exe/pang_01.exe

4.3 How to play:

- Menu:
  - Select the option with up and down control keys, and confirm with enter.
- Game:
  - Move the character with left and right control keys.
  - Shoot pressing Space

5 Test report

BFR1- There is one character which can move horizontally	
Executor:	Test team
Date:	9th April 2008
Time used:	2 minutes
Evaluation:	Positive, the character moves and it does not go outside the game screen.

BFR2- The player is able to shoot a “laser gun” pressing the space.	
Executor:	Test team
Date:	9th April 2008
Time used:	3 minutes
Evaluation:	Positive, the character can shoot properly

BFR3a - There is one stage, with one ball bouncing.	
Executor:	Test team
Date:	9th April 2008
Time used:	3 minutes
Evaluation:	Negative, the ball bounces but sometimes disappears.

BFR3b - There is one stage, with one ball bouncing.	
Executor:	Test team
Date:	9th April 2008
Time used:	5 minutes
Evaluation:	Positive, the ball now bounces properly.

BFR4 - There is only one kind of ball, which disappears once it has been reached by one of the “laser shots”.	
Executor:	Test team
Date:	9th April 2008
Time used:	5 minutes
Evaluation:	Positive, the collision is detected properly and the object disappears.

M2: Addition of three different sizes of balls, which produces smaller balls if are reached.	
Executor:	Test team
Date:	11th April 2008
Stimuli:	The balls generate smaller ones until they have size one, when in case of being reached, disappear.
Expected response:	Success in 10 of 10 executions.
Observed response:	Success in 10 of 10 executions.
Evaluation	Positive.

M3: Addition of new stages with different number and sizes of balls.	
Executor:	Test team
Date:	12th April 2008
Stimuli:	The game should be able to load 3 different stages whose characteristics are stored in 3 different XML files
Expected response:	Success in 5 of 5 executions.
Observed response:	Success in 5 of 5 executions.
Evaluation	Positive.

[M6]: “Adding a menu”	
Executor:	Test team
Date:	12th April 2008
Stimuli:	The game should be able to display a menu which transits properly over the different states of the game.
Expected response:	Success in 10 of 10 executions.
Observed response:	Success in 10 of 10 executions.
Evaluation	Positive

[M4]: “Addition of high scores stored permanently in XML format”	
Executor:	Test team
Date:	12th April 2008
Stimuli:	The game should be able to display the information about the high scores sorted.
Expected response:	Success in 4 of 4 executions.
Observed response:	Success in 4 of 4 executions.
Evaluation	Positive.

[M5a]: “Insertion of new high score and user name in old arcade style”	
Executor:	Test team
Date:	12th April 2008
Stimuli:	The game should be able to store the user name if it is necessary. It has to display only 5.
Expected response:	Success in 4 of 4 executions.
Observed response:	Success in 0 of 4 executions.
Evaluation	Negative, the game does not remove properly the last one in the list; six names are displayed.

[M5b]: “Insertion of new high score and user name in old arcade style”	
Executor:	Test team
Date:	13th April 2008
Stimuli:	The game should be able to store the user name if necessary. It has to display only 5.
Expected response:	Success in 4 of 4 executions.
Observed response:	Success in 4 of 4 executions.
Evaluation	Positive.

6 Relationship with the architecture

The most important evolution in our development process came from understanding how exactly the XNA and XQuest worked. When understanding how the game loop was implemented it made it possible to finish up the rest of our architecture.

We don't think that our implementation varies from our original architectural description by any significant amount, but we believe that our original architectural description lacked a lot of the understanding of the architecture that we have today. One might say that our implementation and our original architectural description differ so little because our original architectural description was so vague.

A definite weakness for us was a lack of understanding of how we could design the architecture and how we could implement these designs to our advantage. When understanding how to take advantage of the XNA and XQuest it became possible to understand how the rest of the game could be implemented and the rest of the architecture sort of grew out of that.

7 Problems, Issues and points learned

The implementation of this videogame has been interesting, and thanks to the use of XNA and Xquest libraries, we were able to focus on the logic of the videogame saving a lot of time.

We did not have any previous knowledge about the framework, the language, etc., and we think that with more experience we could have been able to create a more complex videogame.

We have seen how important it is to have a well defined and good architectural design at an early stage to help us make clean code that is well structured and easily changed or modified. We especially learned how difficult this can be when your original knowledge on the subject is lacking. For us the problem was a lacking understanding of the XNA, Xquest and the structure of video games in general. But we assume the same is true for any system built for any purpose. Without a general understanding of the workings of a system it is very hard to create a good architecture for it. And without a good architectural design it is impossible to create a great product.