TDT4240 Programvarearkitektur
X2

Julius Buset Asplin
Emil Grunt
Hvard Kindem
Dimitry Kongevold
Nicolay Thafvelin
Milos Zlatkovic

April 27, 2012

Contents

1 Introduction

This document describes product delivered by team X2 in the courseTDT4240 Software Architecture. The document contains: design and implementation details, tests run on the final product and during the development, the Users manual and the list of inconsistencies between implementation and the architecture.

The main attribute that have been focus under development of the project is modifiability. Game, as in now, is just an engine with the possibilities to add new features, characters, maps and power ups. Those opportunities could be explored by an advanced user as well as a developer, or future support personnel, by adding an XML file to a specific folder.

2 Implementation Details

We originally decided to use the the architectures: ModelView Controller(MVC), Service Oriented Architecture(SOA), Observer Patterns(OP) and State Machine.

2.1 Model View Controller

MVC is the architecture that we have focused the most on. This makes it easy for the developers to get an overview of the project.

Models contains all the data in the game. This consists of game data, character data, character-, cursor- and map data. As we areusing the MVC architecture, these classes only contain raw data, nothing to view them at all.

Views contains everything that can be displayed. The different views then get their data from the models and display them accordingly. Defined as views, we have sprites, images, text viewing and the menu view.

The controllers handles all the data from the models handles all updates and uses the views to display the result.

Models
- Box
- Character
- CursorModel
- GameModel
- GameOptions
- Map
- MenuEntry
- Move
- Player
- PowerUp
- Weapon

Views
- ImageTexture
- IView
- MenuView
- Sprite
- TextBox

Controllers
- CameraController
- CharacterController
- CursorController
- GamepadController
- GamePlayController
- IController
- MapController
- MenuController
- OverlayMenuController
- SoundController

2.2 Service Oriented Architecture

We found that this was a nice addition to MVC. We have implemented a class, ControllerViewManager which acts as an interface for the controllers and views. This class then handles all of the world updating as well as the rendering of the game and handling of sprites.

By using this, we dont need to worry about adding or removing components and makes it very easy to add more functionality to the game, as it then just has to be added to the SOA interface.

2.3 Observer pattern

2.4 State Machine

3 User's Manual

3.1 Requirements

- Visual Studio 2010 [1]
  As the project is built using Visual Studio 2010, this recommended. It is

possible to compile the project in earlier versions as well, but this means rebuilding the project structure or convert it.

- XNA Game Studio 4 [2]
  This is required.

3.2 Getting started

To compile and run the project, open it up in Visual Studio and simply press F5. This will compile and run the program. The first screen you will see is the character selection screen. Here you can select your characters by moving your pointer with the XBox control pad or the W, A, S, D keys for player 1 and the arrow keys for player 2. Player 1 select his or her character by pressing v and player 2 by pressing j. After selecting your characters, start the game by pressing return. Here you can move around with the XBox control pad or the W, A, S, D keys for player 1 and the arrow keys for player 2. You can attack using v for player 1 and j for player 2.

3.3 Gameplay

The goal of the game is to knowck the opposing player(s) off the map. This is done by attacking them and usingthe knockback effect of the attack to get them off. Hitting your opponen(s) will increase the power of the knockback and further improve your ability to kick them off the map. When a player is kicked off the map, he will respawn after a few seconds, loosing a ”life”.

4 Test Report

Modifiability 1: Add new character.	
Executor:	
Date:	
Stimuli:	To add a new character up should be easy and possible to do with out re compile
Expected response:	A new character should be visible in character selection menu. With out the necessarily to re compile
Executed actions:	Added a new character XML file in a specific folder
Observed response:	
Evaluation:	

Modifiability 2: Add new map.	
Executor:	
Date:	
Stimuli:	To add a new map up should be easy and possible to do with out re compile
Expected response:	A new Map should be possible to chose in Map selection menu. With out the necessarily to re compile
Executed actions:	Added a new map XML file in a specific folder
Observed response:	
Evaluation:	

Modifiability 3: Add new power up.	
Executor:	
Date:	
Stimuli:	To add a new power up should be easy and possible to do with out re compile
Expected response:	A new power up should have the possibility to drop in game. With out the necessarily to re compile
Executed actions:	Added a new power up XML file in a specific folder
Observed response:	
Evaluation:	

Modifiability 4: Replace a controller method.	
Executor:	
Date:	
Stimuli:	The change in controller class should not affect other, functionalities.
Expected response:	The program should use the new controller method after recompile the game.
Executed actions:	Changed the name of the controller class in the code
Observed response:	
Evaluation:	

Performance 1: Frame rate	
Executor:	
Date:	
Stimuli:	The game should appear smooth for the user.
Expected response:	The frame rate should not go below 30fps.
Executed actions:	Play the game for 20 min.
Observed response:	
Evaluation:	

Performance 2: Game delay	
Executor:	
Date:	
Stimuli:	The player should not notice any waiting between the pressed button and the executed action on the screen.
Expected response:	The action should be done immediate after clicking.
Executed actions:	Start the game and execute actions/game play.
Observed response:	
Evaluation:	

Portability 1: Run on XBox	
Executor:	
Date:	
Stimuli:	The game should be runable on Xbox 360.
Expected response:	The game should run on Xbox as good as on PC.
Executed actions:	Uploaded game to the Xbox and started it.
Observed response:	
Evaluation:	

Testability: Beta	
Executor:	
Date:	
Stimuli:	The game should be tested as created. for faster detections of bugs.
Expected response:	The game should be able to run with out all of the features before release date.
Executed actions:	During the developing Simple tests have been executed.
Observed response:	It was possible to run tests on the unfinished version of the game a week before delivery.
Evaluation:	Passed.

5 Relationship with the architecture

The architecture description could be found in ArchitecturalDoc.pdf file. The implementation has some inconsistencies with the planed architecture, caused by several different reasons. This chapter lists them and tries to explain the developers view and logic behind those changes. Model View Controller (MVC) In the architectural description modifiability is one of the primal attributes for the architecture, and had lead to the choice of MVC. Through the implementation we can observe some inconsistencies with the MVC architectural pattern. The reasons for these inconsistencies are caused majorly by the use of FarseerPhysics COT in the project. Development View and Control-Float Control-float of the program has been changed to fit better with FarseerPhysics engine, and now is different from the sequence diagram presented by the Scenario View and Development View in the architectural description, because of the great damage from the architectural-drift. Possible reasons for inconsistencies As the development team worked with the XNA framework, there were difficulties fitting MVC into XNAs pipe and filter architecture. Though the Architecture document gave a possible solution for how to implement MVC inside a pipe and filter pipeline, the difficulties only got worse with FarseerPhysics engine, resulting into an architectural-drift. This could be avoided earlier in the process if the architects had more experience and understanding of XNA and used COTs. If the development view was more strict and detailed, so it could give more support to the developers, the possibility of an architectural-drift and erosion would be considerably lower, resulting in a much more consisting product.

6 Problems, issues and points learned

The main problem we have experienced during this project was the lack of time and experience. As well as communication problems with inn the group and different programming experience and habits. As our first try as software architects we underestimated the value of creating a detailed architecture and fast-forward towards the implementation. This resulted in an architectural-drift and erosion lowering the required quality of the product and increasing the implementation time. With a better planed out and detailed architecture, many of issues and problems we have encountered would be solved. For instance the use of the FarseerPhysics engine that we thought would save some of implementation time ended up causing an architectural drift. Next time well be more careful in our choice and usage of COTs. And gather enough knowledge and understanding of bough developing platform and COT, before taking a next step. If we were to start again from scratch we would try to stay closer to chosen architecture and use more time to think over the product. We would choose our architectural/design patterns more wisely a consideration to the environment/platform well use. As well as create more detailed views, so it will be more informative for the stakeholders. Specially the developing view.

References

[1] Visual Studio — MSDN, *Microsoft*, 2012, http://msdn.microsoft.com/en-us/vstudio/aa718325

[2] Download: Microsoft XNA Game Studio - MSDN, *Microsoft*, 2012, http://www.microsoft.com/en-us/download/details.aspx?id=23714#overview