Space Shooter

Project Legacy

COP 4331: Processes for Object Oriented Software Development
Fall 2012

Table of Contents

Prefatory Information

Modification history:

Version	Date	Who	Comment
v0.5	11/23/2012	Joshua Thames	Initially unfinished compilation
V1.0	11/24/2012	Joshua Thames	Finished version – need to check with ta concerning the inclusion of charts

Team Name:

Team 2

Team Members:

Name	Email address
Andre Meireles	andre.meireles@knights.ucf.edu
Alex Banke	banke@knights.ucf.edu
Christopher Margol	margol_chris@knights.ucf.edu
Chris Lin	christophercklin@gmail.com
Joshua Thames	jthames88@knights.ucf.edu
Thaddeus Latsa	tlatsa@knights.ucf.edu

Roles

The intent of this section is to describe to the reader who did what for the completion of the project.

Item	Person/s Responsible	Contri-bution	Notes
Concept of Operations:	Joshua Thames	100%	
Project Management Plan:	Joshua Thames Chris Lin	60% 40%	Joshua did the top 60% of the documentation and finalized the report while Chris did that latter portion
Software Requirement Specification:	Alex Banke Joshua Thames	99% 1%	As the lead programmer, Alex was solely responsible for this document while Josh check for formatting consistency
High Level Design:	Joshua Thames	95%	Joshua wrote the document and made the final decisions but not without the collaboration with other group members
Detailed Design:	Thaddeus Latsa Joshua Thames	99% 1%	Thad wrote the document and collaborated with the team during a group meeting - Josh double checked for consistency
Web Management	Chris Margol	100%	Chris was 100% responsible for the maintenance of the website minus the update to individual pages
Test Plan	Andre Mereles Joshua Thames	99% 1%	Andre developed and wrote the test plan while Josh checked for consistency
Test Results	Andre Mereles Joshua Thames	99% 1%	Since Andre developed and wrote the test plan, he was the one responsible for executing it
Project Legacy	Joshua Thames	100%	Written by Josh
Build Instructions	Chris Lin	100%	Written by Chris Lin
User's Manual	Thaddeus Latsa		Written by Thad
Documentation	Joshua Thames Alex Banke Thaddeus Latsa Andre Mereles Chris Lin	30% 10% 20% 20% 20%	The corresponding responsibilities are listed above
Coding	Alex Banke Chris Margol Andre Mereles	45% 27.5% 27.5%	It was not originally intended to be divided in this manner but because of the specialization and expertise of these individuals, they became the programmers for the project

Analysis

The intent of this section is the review the end product concerning our project. The primary material that will be taken into account is our adherence to the planned routine and how closely we were able to adhere to that plan.

Assessment of the Quality of the Final Product:

Our final product works as expected under all known circumstances. Our product works under Windows 7 and Mac OS X Mountain Lion operating systems. The goal was to allow the user to experience a seamless transition from reality to a 2D simulation of space shooting type environment without any programming "bugs" to distract the player. I believe we have accomplished that goal considering the time allotted.

Recommended Use of the Final Product:

The intended use of the product is simply for entertainment. Our product is not designed to accurately simulate the effects of gravity or forces of movement in a frictionless flight (without air there is no friction) in any way, however, the player may find that the actual game play is "realistic" in some ways. We did not wish to recreate the 2D space shooter genre but did implement a few new features that the user may enjoy.

Known Problems:

There are no known problems at this time. All the features that were in the minimum requirements have been met; however, there are some aspects about the final product that could be improved. One of which is the intermittent flashing of some of the sprites when there are a lot different items displayed on the screen at once.

Adherence to Project Plan:

First concerning how we deviated from the plan: I believe that we fell very short of the configuration management and expected depth and breadth of the necessary programming. We were somewhat lacking in these areas: coding standards, communication, estimation of time to complete certain programing modules, and the software lifecycle process was only loosely adhered to because of some difficulties in programming.

Concerning how we did well, I believe that each individual maintained diligence in getting his own responsibilities done with excellence, generally speaking. The code itself is written well, with no known errors and the documentation is professionally done, adhering to the specifications.

Finally, concerning the root of the problems, I believe that configuration management could have gone much better if the person responsible what have gotten a head start and completed his duties in a more timely manner. The learning curve for our configuration management tool (git) was high and required a large base knowledge to even properly use the most basic functionality well. Only one person learned the configuration management tool well enough to use without supervision. The other areas were lacking primarily because of our limited experience with group coding across different networks and different locations, i.e., we were not programming in the same office.

Defect Analysis:

Defect	Found by	Found During	Solved During
Selection oval does not correspond to menu item	Alex	Release 2	Release 2
Selection oval goes out of bounds	Alex	Release 2	Release 4
Ship only moves one button at a time	Several members	Release 3	Release 3
Images flash on screen	Several members	Release 3	Release 3
Ship flashes wrong upgrades	Chris Margol	Release 3	Release 3

Quality Assurance:

The quality assurance was done in a sufficient manner. Each of the coders did his own unit testing before moving to integration with the system. The programmers went into testing with the intent to "break the program" as to find as many errors as soon as possible. Instead of letting the errors pile up, we decided to tackle each one early and not move on until it is working as expected. This practice ensured a quality product at the end.

Configuration Management:

Our CM could have been much better. As I mentioned above, git is a difficult program to use "out of the box", there are a lot of ways and a lot of different things that can go wrong when several users are attempting to use it across different locations and with very limited or no knowledge. The simplest tasks like pushing would usually go well but often the person responsible for CM would have to step in and correct significant errors.

I propose to the reader that CM be handled by an expert and that **each** user involved with coding have a strong base knowledge of all the common functions of the CM tool while the expert has a very advanced knowledge and experience base.

Suggestions for the Future:
Concerning Technical Process:

1) A few key improvements to our project could have drastically positive results in the future. The primary improvement our team could have made is making a more thorough detailed design of our entire system. Our detailed design lacked sufficient depth to be of use which left the programmers scrambling to catch up as the deadline approached. It is an imperative to find out **what** exactly needs to be coded along with **how** to go about coding the function. This will force the designers to be more familiar with the specifics of the program which should result in more accurate time estimates.

2) More decisive team leader. The approach with a single chief programmer is a good model but that individual has to have excellent communication, planning, and programming skills. I believe that the

project manager and lead programmer combo is a good model but only if each have great leadership and programming skills.

3) Also, CM became a problem and we ended up using dropbox for the bulk of moving code from one programmer to the next. CM is very important and can drastically hinder the progress of code writing if not everyone has the latest version of the code to work on. This can result in two or three members trying to develop the same feature simultaneously or some individuals trying to fix a bug that another person has already solved.

What I Would Change:

1) Time to start coding
2) Depth of detailed design
3) From 2 lead design (lead programmer and project manager) to single chief programmer
4) A more simple CM approach/tool

If 10x the Size:

1) Leadership model
2) Configuration management tool
3) User's knowledge of CM
4) Depth of understanding and research before we start coding
5) Communication could drastically be improved to include a bi-weekly check in concerning progress and problems. There was little accountability except perhaps the day before the coding assignment was due.
6) Also, some team members were vastly more involved which resulted in their overload and other's lack of contribution.
7) There was a lot of interdependency on the development of code modules/functionality that hindered parallel progress. This would have to be de-coupled for a larger project

If 100x the Size:

1) All of the above modifications
2) More thorough accountability, check-ins, progress reports. It was common that errors would arise but no one would say anything until the day the assignment was due making it nearly impossible to get the functionality working on time.
3) Several chief programmers would be required for a program 100x the size. I would suggest several teams of specialization but also one expert team of programmers to oversee and ensure that the project is not going out of scope and is progressing.
4) I would suggest that the programmers responsible for particular modules would also play a significant role in the design of the module so that he is more familiar with it when coding begins.