2012

Space Shooter

Project Management Plan

COP 4331: Processes for Object Oriented Software Development
Fall – 2012

Team 2
9/3/2012

Table of Contents

Prefatory Information

Modification history:

Version	Date	Who	Comment
v0.5	09/03/2012	Joshua Thames	Initially unfinished compilation
v0.6	9/5/12	Joshua Thames	Modification history – Plan for tracking/controlling progress
v1.0	9/6/2012	Christopher Lin	First Completed Version
v1.1	9/11/2012	Joshua Thames	Formatting
v1.2	9/13/2012	Christopher Lin	Formatting
v1.3	9/18/2012	Joshua Thames	Formatting, update contents of organization, tools, config man., and QA
v1.4	9/18/2012	Joshua Thames	Formatting, update contents of work packages, Progress Metrics
v1.5	10/8/2012	Joshua Thames	Updating due-dates, Members responsibility
V1.6	10/8/2012	Chris Lin	Updated pert chart/table of work packages

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

Project Overview

This project will be a 2D space shooter game in which the user can control an upgradable “space ship”. The game will feature an interactive environment with asteroids moving across the screen at various times, each with their own gravitational pull. The environment will also feature Enemy NPC ships which will shoot at and possibly collide with the user’s ship. The user will be able to shoot at other ships to gain currency to upgrade hull, weapons, engines, and thrusters. Each upgrade will change the appearance of the user’s ship in some way.

Reference Documents

- Concept of Operations
- Test Plan

Applicable Standards

Coding Standard

- All functions must have comments including input parameters, output parameters, and a short description of their function. If the function cannon be described in a simple sentence then the logic needs to be re-worked and reduced.
- All functions must have a corresponding testing function to ensure reliability. The testing functions will be added to a regression suite
- Every file in the build that includes code must have an appropriate commented heading as follows:

```
/*************************************************************************
*************************************************************************
*
*     ============    Team 2  -  Project 6: Space Shooter    ============
*
*      AUTHOR(S):
*      DATE OF FIRST VERSION:
*      UNIT TESTING COMPLETED: <Yes/No>
*      STATUS OF UNIT TESTING: <Working/not working/any known bugs?>
*      INTEGRATION TESTING COMPLETED: <Yes/No>
*      STATUS OF UNIT TESTING: <Working while integrated in whole system,
*            known bugs with integration testing?>
*                     problems and where?>
*      FILE NAME:
*      DATE LAST MODIFIED:
*      PURPOSE OF FILE IN THE SYSTEM: <short and concise descriptions tab
*                     them out if necessary>
*
*************************************************************************
*************************************************************************/
```

- Major end Braces } should have a comment as to what it ends. Ex. `} // end main()`
- Naming conventions: all variable names, functions, classes, header files and source files must have an underscore between words to increase readability. Classes must use capital letter for the first letter in each word. Ex. `test_function()` Ex. `My_Class`
- Must comment the logic of the code using whitespace and tabs as necessary

Document Standard

The minimum standard for grading is provided according the UCF Software Engineering courses templates for artifacts. The specific standards for each artifact are as follows:

- Each document must have a cover page that includes the year, project name, document title, course name and number, and date of first version of the document
- Each document must have a “Table of Contents”
- Each document must have a major title (Heading 1 in Microsoft Word 2010) called “Prefatory Information” that includes: modification history, team name, team members
- Paragraph and sentence font size must be 11

- Paragraph and sentence font must me Calibri
- Heading fonts will be Cambria and the sizes will change depending on specific heading
- All documents must include a Header called “Contrast (odd)” in Microsoft Word 2010
- One or two new lines after each heading depending on readability
- Each file name must have its appropriate version number corresponding to the modification history table under “Prefatory Information” Ex.: Project Management Plan_v1.0
- Whenever a list of 3 or more is used, implement a bullet point format
- No periods should be used for tabbed lists unless it is one or more complete sentences
- Must use standard English grammar rules for documentation (except tabbed lists)
- Must use proper spelling and spell checking

Artifact Size Metric

The Artifact Size Metric will be measured in general by the working functionality of the program since we are using the Incremental Process. Each functionality of the program will be measured objectively by the class characteristics and complexity. Each class will be measured in complexity by the number of methods used in that class and estimation of the lines of code per class.

Project Team Organization

These are the primary roles of each of the members. We did have one member arrive later in the planning that expected so he did not have many responsibilities during deliverable one but his role will become clearer as time goes on. The organization will be handled by the Leads (Programming and Project) but each person will have coding, documentation, maintenance, testing, and reporting activity duties to some degree or another.

Primary Roles

Joshua Thames – Project Manager

- Coordinate the overall structure and logistics of the team decision making
- Designate tasks to team members
- Organize meetings
- Lead meetings with the Lead Coder
- Impress deadlines on the team
- Organize and analyze documents
- Analyze risks with Lead Programmer and make adjustments as necessary
- Maintain Project Management Report
- Maintain High Level Design
- Maintain individual time and activity logs

Alex Banke – Lead Programmer

- Lead the team in the organization and execution of all major programming related tasks
- Assign programming tasks to all team members
- Maintain Incremental Software Design pattern
- Facilitate decision making
- Lead meetings with Project Manager
- Keep team on track in regards to coding progress and metrics
- Maintain individual time and activity logs

Christopher Margol – Web Host Manager

- Maintain current and up to date website content
- Ensure all document links are working
- Ensure website is at functional minimum requirements
- Maintain individual time and activity logs

Chris Lin – Media, Documents, Management

- Maintain PERT chart
- Acquire sound samples and graphics
- Maintain individual time and activity logs

Andre Meireles – Lead Tester

- Maintain code testing standards
- Execute Regression suite
- Ensure code is reaching minimum benchmarks
- Maintain documents pertaining to testing standards
- Maintain status of overall program correctness
- Maintain individual time and activity logs

Thaddeus Latsa – Media, Documents, Detailed Design

- Acquire sound samples and graphics
- Ensure document standardization
- Ensure coding standardization
- Maintain Detailed Design Document
- UML Diagrams
- Maintain individual time and activity logs

Communication

Our team will maintain communication in these ways:

- Maintain activity logs on the project website

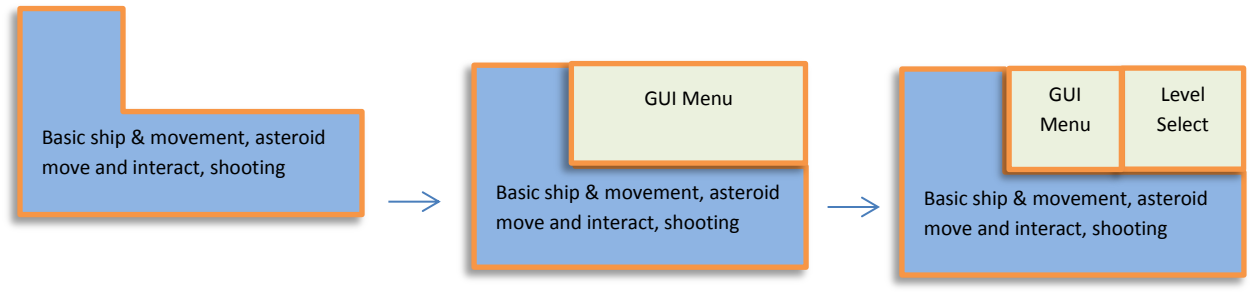
- Weekly during team meetings
- Emails for meeting discussions
- Via cell phone for other important discussions and requests during the week

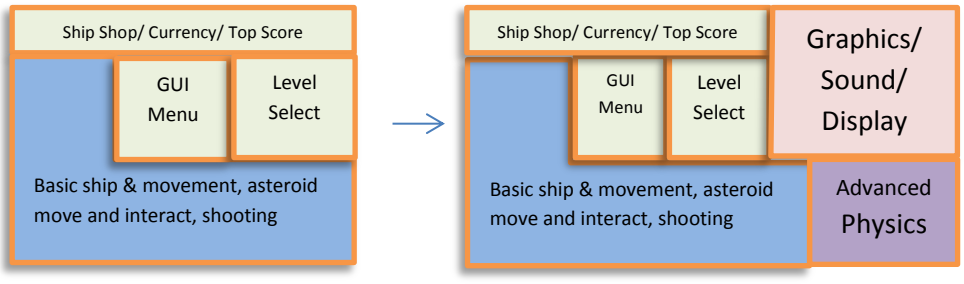
Deliverables

Artifact	Due Dates
Meeting Minutes	Once a week
Individual Logs	Once a week
Group Project Management Reports	Once every two week
Concept of Operations	9/18/2012
PMP	9/18/2012
SRS	9/18/2012
Test Plan	9/18/2012
High-Level Design	10/9/2012
Detailed Design	10/9/2012
Presentation 1	10/9/2012
User's Manual	11/27/2012
Final Test Results	11/27/2012
Source, Executable, Build Instructions	11/27/2012
Project Legacy	11/27/2012

Software Life Cycle Process

We will be following an incremental life cycle process. We chose this process because we felt like it would be easiest to test our program with functions being added one at a time. Our first release will render a basic space environment and the spaceship with basic movement capabilities. Functions to be added later will include shooting, enemies and their gravity, collision/movement algorithm, etc. Our software life cycle diagram is give below:





Tools and Computing Environment

The program is being written with Java using the Eclipse IDE. Since Java runs on the Java Virtual Machine, our compiled code should run on all operating systems with the JVM. Our team will be using the Java Development Kit (JDK) version 6.

During development and testing, we will use Windows 7 primarily – one member uses a Mac. Issues with compatibility between source code will be taken into account for his machine.

The standard compiler has not yet been established. We believe, at the present time, that any compiler that is compatible with Java Eclipse IDE and is able to execute .java files should be cross compatible.

The libraries that will be used have not been established at this point.

Configuration Management

We will use Github to host our source code and keep it accessible and reliable for each programmer. This application will handle the ins and outs of source code version control and changing control. Any changes that need to be made that are not automated will go to Alex Banke for authorization.

Github will handle storage of our code, allowing access across team members, a log of who last updated what, and team permissions. We will be able to lock down the working and tested code so that no one accidently makes an inappropriate change.

Quality Assurance

Andre Meireles will ensure that the application is up to our established standards that are outlined for quality in the Test Plan document. Our results will be reported on weekly or biweekly depending on the stage of development. Bi-weekly at first then more frequently as the deadlines approach.

Risk Management

Risk: A team member withdraws from class
Solution: Project leader and Programming Leader redistribute the responsibilities of the withdrawn student
Risk: Poor communication amongst team members resulting in redundancies and wasted time
Solution: Logging of all group activities and constant communication through texts, emails, etc.

Table of Work Packages, Time Estimates, and Assignments

Deliverable 1

Assignment	Team Member	Estimated time
Concept of Operations	Josh Thames	1 week
Project Management Plan	Josh Thames/Chris Lin	2 week
Software Requirements Specification	Alex Banke	1 week
Test Plan	Andre Meireles	1 week
Web Site	Chris Margol	2 weeks

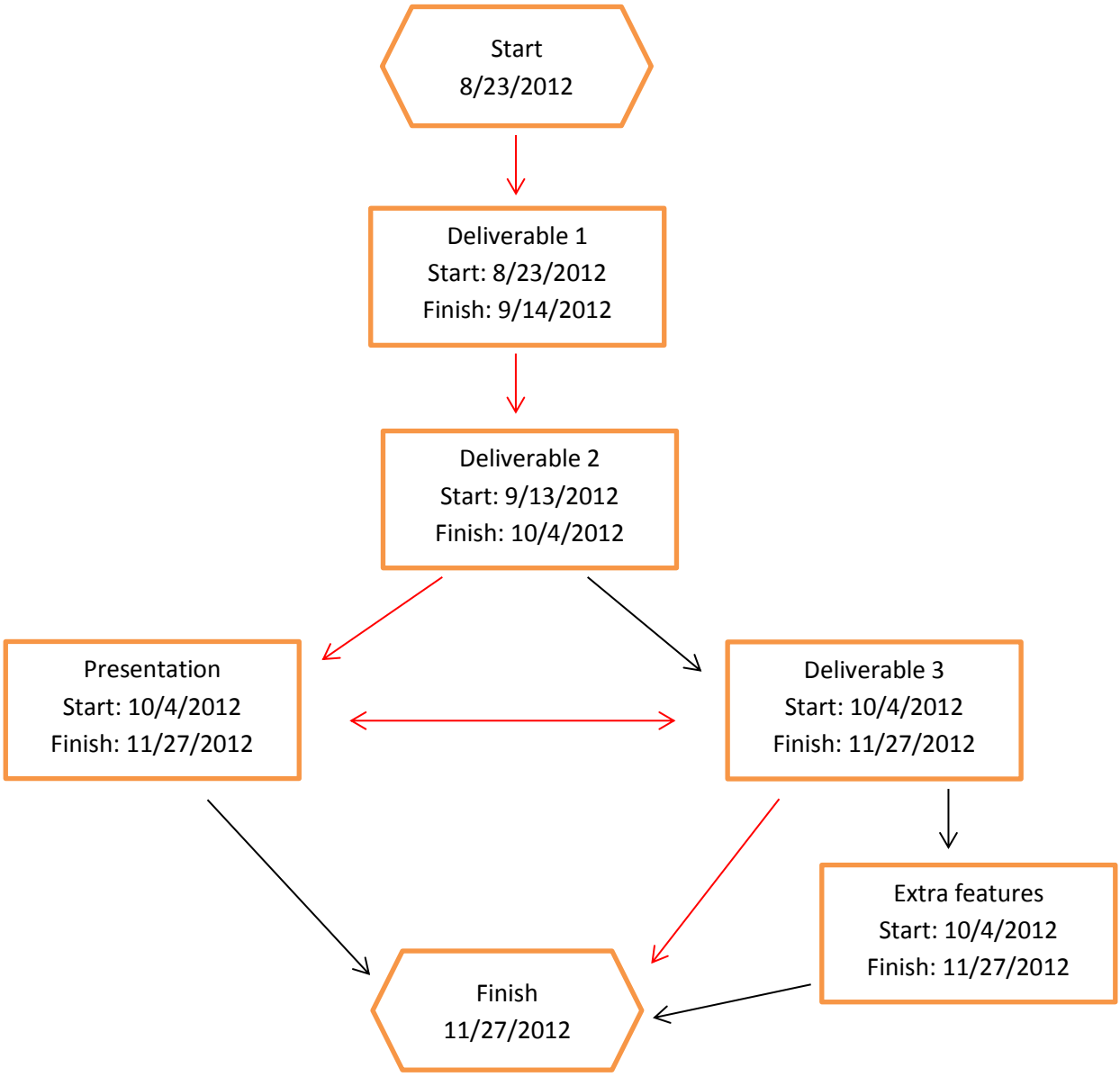
Deliverable 2

Assignment	Team Member	Estimated time
High-level Design	Josh and Alex	3 weeks
Detailed Design	Josh	3 weeks

Deliverable 3

Assignment	Team Member	Estimated time
Test Results	Andre Meireles	1 week
User's Manual	TBA	1 week
Source Code	Team	1.5 months
Build Instructions	TBA	2 weeks
Project Legacy	TBA	1 week
Final Presentation	Group	1.5 weeks

PERT Chart


Start
8/23/2012
Deliverable 1
Start: 8/23/2012
Finish: 9/14/2012
Deliverable 2
Start: 9/13/2012
Finish: 10/4/2012
Presentation
Start: 10/4/2012
Finish: 11/27/2012
Deliverable 3
Start: 10/4/2012
Finish: 11/27/2012
Extra features
Start: 10/4/2012
Finish: 11/27/2012
Finish
11/27/2012

Technical Progress Metrics

Program Requirements		
Functionality	**Expected Size**	**Actual Size**
Start menu	1 Class, 50 lines	TBD
Moveable ship	1 Class, 10 methods	TBD
Enemies	5 Classes, 15 methods	TBD
Upgrade store	1 Class, 10 methods	TBD

Program Requirements
TBDs = 4
Completed = 0
% Completed = 0%

Documents	
Metrics	**Status**
UML Diagrams	8 Class diagrams - Uncompleted
Test plan	Completed
Website	Completed
Software Requirements Specifications	Completed
Detailed Designed	Completed
High Level Design	Completed
User's Manual	TBD
Build Instructions	TBD

Documents
TBDs = 4
Completed = 3
% Completed = 37.5%

Testing Metrics	
Memory usage	No Tests
Response time	No Tests
Any bugs and their causes	No Tests

Plan for tracking, control, and reporting of progress

At a minimum, each team member will post the following information weekly: individual time and activity log, individual status information, individual issues and problems, and individual defect log.

Each week, the project manager will: read and analyze the logs; examine the technical content of the work done to date; examine the technical progress metrics; consider the QA results; reassess the potential project risks; and take corrective action if necessary.

The project manager will issue a Project Management Report on the schedule as indicated in the deliverables section above. At a minimum, the Project Management Report will be generated every two weeks and will include the following information: 1 sentence description of overall status, 1 or 2 sentences of any planned changes to the project plan, graph of planned vs. actual time, graph of planned vs. actual for each technical progress metric, updated PERT chart."