Software Workshop Team Java (06-08165) 2010/11, Dr. E. Thompson

Project Report:
Space Runner Game

Team B3:
Daniel Cecil
Jere Ketonen
David Saunders
Michal Staniaszek

March 29, 2011

Work Breakdown

Coding	**Daniel**	**Jere**	**David**	**Michal**
Datastructures	Contribution: 100%	0%	0%	0%
...	...	...	...	...

Report	**Daniel**	**Jere**	**David**	**Michal**
Introduction	Chapter 1: 100%	0%	0%	0%
...	...	...	...	...

Contents

Abstract

You should write a one-page abstract written as an "Executive Summary". It should be written for someone who is familiar with the Team Java module, so that there is no need for background or generalities. Rather, you should explain what is special about your project, and what you claim to have achieved. (One page maximum.)

Chapter 1

Introduction

Give a brief overview and guide the reader to the important points in the remaining sections.

This template for your report also contains some examples of how to use some LaTeX elements and commands. In particular, there are examples for tables, how to include figures, and various environments for bullet points or enumerations.

For further information (and here is how to do a bullet list):

- look on the Team Java web page,
  `http://www.cs.bham.ac.uk/internal/courses/team-java/current`
  (Click on "Guidance".)
- google "latex"
- look at the LaTeXbook [**?**]

This is how to do numbered lists:

1. First point
2. Second point
3. ...

This is how to do sections:

1.1 Some topic

1.2 Another topic

If you set a label with the `label` command, you can then use the `ref` command to refer to that section – e.g. section 1.1. This means you don't need to know about section numbers. Note that `~` means a space that cannot be broken across lines.

Chapter 2

Requirements

2.1 Functional User Requirements

This section outlines the functional requirements which the system will be tested against. Functional requirements are what the system is expected to do and how to user interacts with the software. Each requirement is split into sub-requirements for ease of understanding and clarity.

1. **The human player is able to control one's spaceship**
   This is a core requirement needed to be able to play the game successfully in at least a single player mode without any crashes or bugs.
   (a) The user is able to use either a mouse or the keyboard's arrow keys to move the spaceship.
   (b) The user's spaceship is able to move freely along the x and y coordinates but not leave the frame's boundaries.
   (c) The user will be able to hold more than one key for diagonal movement where the movement speed much be normalised.
   (d) The user's spaceship will be able to be represented graphically on the screen.
2. **The human player will be able to shoot**
   The aim of the game is to enable the user to destroy enemy ships and therefore shooting is a must-have requirement of the game.
   (a) The user will be able to shoot by tapping or holding the spacebar or the mouse button (left click).
   (b) The user's spaceship will have a type of weapon to use, this can be changed during the game (if implemented - dependant on future requirement).
   (c) The user's shot will follow a set path forwards (negative y-coordinate movement).
   (d) Shots which leave the frame's boundary will be removed from the game state.
3. **Enemies will be created to be destroyed by the user**
   This is a core functional requirement that needs to be implemented to enable the user to progress through the game by shooting down opposing units.
   (a) Enemies will be spawned at set locations on the screen.
   (b) Enemy units are to have a set health limit.
   (c) Enemies will be able to be shot by any user spaceship.
   (d) Enemies are to be distinguishable from friendly user spaceships by using different shapes or graphics.

(e) The enemy unit's health will be decreased when a user's shot collides with the enemy.

(f) Enemy's will 'die' once all their health have been depleted.

4. **Enemies are to be able to return a level of resistance**
   This requirement is needed to make the game more interesting by introducing the possibility of a player 'death'

   (a) Enemies are able to return shots towards the human players with the use of different weapons.

   (b) Enemies are able to move in certain paths (zig-zag, diagonal, straight, side-to-side).

   (c) If the player collides with an enemy the player will 'die'

   (d) 'Boss' enemies are to be introduced which fire more shots and have more health.

5. **The game is to run continuously with set events occurring at regular intervals**
   This requirement ensures the game runs smoothly and that something will always happen. For example, to stop the incidence of no more enemies being spawned (so the game is playable).

   (a) The game is to implement a Timer class.

   (b) Enemies will always be spawned at set intervals during the game. These can be changed to be more or less frequent (if future requirement is implemented).

   (c) Each tick of the timer will move enemies, player units (depending on user input) and projectiles.

   (d) The game panel will be redrawn at every tick of the timer.

6. **The game will be able to be multiplayer across the network**
   This requirement is necessary to fulfil the assessment criteria allowing for a second human user to play in co-operative mode with each other against the computer enemies.

   (a) Each human user will be able to select whether they will act as the host or the client PC.

   (b) Clients will be able to enter the Host's IP address to connect.

   (c) The host's game will start immediately after selection with clients dropping into the game at a set spawn point.

   (d) The game must be able to support at least two human players and a maximum of eight players (7 clients).

   (e) All users must be connected to the same LAN network.

   (f) All users must have similar game information on their screens (player, enemy and projectile positions).

   (g) With each tick of the timer (requirement 5) each player's screen will be updated with network data from the host.

7. **The game must have a terminating clause**
   This requirement ensures that the game will end at some point.

   (a) Once a player's health has been depleted, that unit will 'die' and be removed from the game allowing other player's to carry on playing.

   (b) If a player collides with an enemy unit they will also 'die'

   (c) The player's score will be displayed on termination and if high enough will be recorded in a high scores table.

8. **The game will include a Graphical User Interface (GUI)**
   This allows all users to be able to start the necessary game type as well as actually play the game with the information displayed on the screen.

(a) The game will run from a single frame.

(b) Panel's are to be added to the frame: Menu, Game, Gameover.

(c) The Menu Panel is to feature buttons corresponding to various game types and options.

(d) The Menu Panel is to be accessible from within the game (Esc key).

(e) The Game is to be able to be paused using the 'P' key.

(f) The window is to be resizable allowing for full-screen play.

(g) The Game Panel is to feature a scrolling 'star-like' background (black with white stars).

(h) The game objects (players, enemies and projectiles) are to be represented by shapes or sprites (graphics).

2.1.1 Attributes

Attribute	**Requirement No.**	**Comment**
Status	All	Approved - development started
Priority	1, 2, 3, 4abc, 5, 6, 7, 8abch - Mandatory. 4d, 8defg - Important	Mandatory requirements must be implemented.
Effort	All	Deadline for code: 22/03/11 (10 weeks). Estimated 40 person-weeks for first release.
Risk	All	Medium probability of risk occurring. Large impact if assessment is not complete. High risk with networking code due to lack of experience.
Target Release	1, 8a, 8h 2 3b-f 4, 5, 7 3a, 8b-g 6	v0.1 v0.2 v0.3 v0.4 v0.5 v0.6
Assigned To	All	4 x Team Members. Requirements and tasks to be distributed at weekly meetings.

2.2 Non-functional User Requirements

This section outlines the non-functional requirements. These requirements relate to the quality of the product and testing requires opinions and qualitative methods rather than quantitative feedback. The project has been split into different categories of requirement.

Usability

9. **To provide a simple, easy to use system in order to play the game**

(a) A novice to the game should be able to gain understanding and play the game within 10 minutes of first playing.

(b) Expert gamers should be able to grasp game concept within 1-2 minutes of playing.

(c) The game should have a clean look and feel.

(d) The user should feel in control of their spaceship with smooth movement and quick reactions.

(e) The menu should have a standard, organised layout with minimal pages

(f) The game should have a professional look

Efficiency

10. **The game should be constantly quick to respond**

    (a) The game should run at a constant quick speed without any lag.

    (b) The user's input should have an almost instant effect on the game.

    (c) Network play should be stable for 95% of the time.

Dependability

11. The game should run first time, all of the time as single player or host.

12. Network clients should be able to drop-into the host's game within 5 seconds.

Environmental

13. The game should run on the platforms detailed in the below section (system requirements).

Development

14. The project should be developed using appropriate software engineering practises within the time frame for the assessment.

15. The project must be written in the Java programming language.

Operational

16. The multiplayer game must be able to run on any LAN with the IP addresses given.

17. High scores will remain stored in each copy of the game until they are overridden by a higher score.

2.3 System Requirements

This section details the requirements and physical devices needed to play the game successfully.

1. The game is to be written in the Java programming language.

2. All user PCs will need to follow the system requirements for Java 6

    (a) Windows 7, Vista, XP, 2000, Server 2008, Server 2003. All 32 and 64-bit operating systems

    (b) Mac OS X

    (c) Most Linux distributions

    (d) Solaris

3. Keyboard and Mouse are required (with Western layout)

4. For multiplayer, all users will need to be connected to a local area network (LAN) at least

5. Monitor and graphics card with at least a resolution of 800 x 600

Chapter 3

Design

This section is crucial. Describe the overall structure of your program at a suitably high level of abstraction. For instance, UML diagrams or informal box-and-arrow diagrams can be used to describe program structure. Be sure to describe the MVC structure used. Note that code listings or screenshots are not appropriate here. An important point is how you have divided the project into modules that different team members can work on, and how these are then integrated. For example, you could use interfaces to describe a clean boundary between modules, so that some team members use the functionality provided by the interface, while another team member implements it. Bear in mind Software Engineering principles of good design like coherence and coupling.

Chapter 4

Validation and Testing

Your applet is expected to be in good working order and do something useful. This chapter is important, because it describes how you assure yourself that that is the case.

Testing is so you can be confident that the software is robust and bug-free. What was your strategy for that? How did you plan unit testing (for components) and integration testing (for the whole applet)? How did the prototype fit in?

Validation is to check that in the end the applet is useful and pleasant to use. What was your strategy for that? Have you tried it out with teachers or students? What kind of rolling validation did you use for the evolutionary part (developing the GUI)?

Establish what your project can handle successfully, and what its limitations are. Use meaningful examples, not lists of trivial cases.

Chapter 5

Project Management

The week-to-week management of your project is documented in your weekly progress reports, so you do not need to repeat that information here. A number of important points that you need to address are:

- What are the main components of the project (or *workpackages*) that have to be completed for it to succeed?
- What software process model (e.g. waterfall or evolutionary) did you use for *individual workpackages*?
- How did you allocate team members to particular tasks?
- How did you coordinate what team members were working on?
- How did you communicate the relevant technical information in the team?

Note that a clean design makes all of these easier, so you could refer to your design section, i.e. chapter 3, where appropriate. You can also discuss difficulties in the team interactions if there were any, and how you dealt with them.

You can, for example, include a chart detailing the time management of your project as shown in figure 5.1. Explain the single workpackages in the text:

Workpackage 1: Requirements specification (Week 1)

Workpackage 2: Something else... (Week 2—3)

Workpackage 3: ...

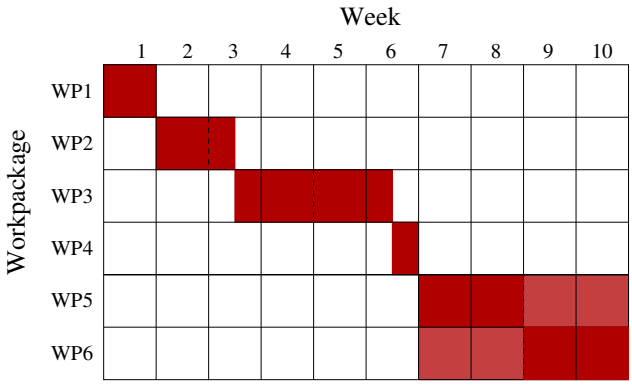

Figure 5.1: Example timeplan of a project.

This is only an example chart, of course; your own timeplan will probably look differently. Note that different workpackages can be worked on in parallel!

Figures are included in LaTeX as `.eps` files. In Linux you can use `convert` to convert any type of graphics file to `.eps` format.

Chapter 6

Conclusions

Evaluate what you have achieved in your project in objective terms (not just things like "we are all happy with the result"). What are the strenths and limitations of your project? If you had more time, what could you add? If you could do it all over again, what would you do differently? Are there general things about software or team management that you have learnt in this project that you could apply if you were going to work on a (perhaps completely different) team project in the future?

Acknowledge any code you have used (if any), and also any that was generated automatically, e.g. by wizards.

Give correct and complete bibliographic information for any sources cited. See the local referencing guide. For instance cite which books on software engineering you have used, for instance [**?**]. Use the `report.bib` file to manage your citations.

When you *quote* material from other sources, you must be absolutely clear *at the point where you quote it* exactly which of your material is quoted and what the source is. As explained in [**?**],

"Direct quotation is not particularly common in scientific writing, as it is generally not the words that matter, but the meaning. Normally it is preferable to rewrite someone else's ideas in your own words, often changing the terminology and other superficial details to suit the new context.

However, in circumstances where it is appropriate to make direct use of the words of another person, those words should normally be included within quotation marks and a reference to the source of the words given in the usual way."