2012

Space Shooter

High Level Design

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

Team 2
10/4/2012

Table of Contents

Prefatory Information

Modification history:

Version	Date	Who	Comment
v1.0	10/04/2012	Joshua Thames	Initial Compilation

Team Name:

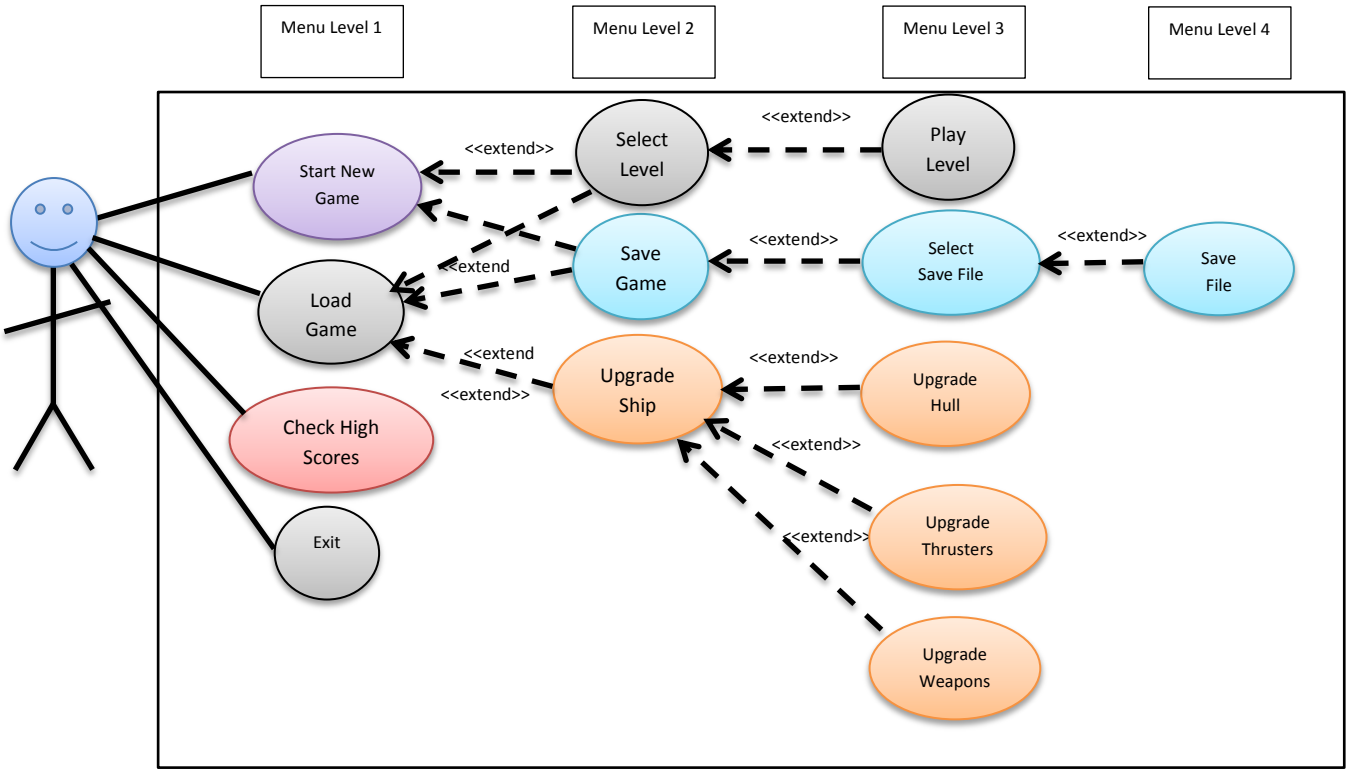
Team 2

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
Josh Thames	jthames88@knights.ucf.edu
Thaddeus Latsa	tlatsa@knights.ucf.edu

High Level Architecture

Use-Case Diagram



This is a Use-Case diagram of our proposed system. Each use case is a menu option that the user is allowed to select. At the top are indicators as to how deep the user is in the menu system. At each menu level, the user will be posed with further options to progress through the menu or return to previous menu level. The user is never forced to pick any of the menu options which is why they have been designated as extensions.

Design Issues

- Reusability
    - This will be a one-time project in which we will not develop further after this class project
    - However, the components in the system will be independent enough so that we can perhaps use the modularity of the system for personal research in the future
- Maintainability
    - Maintain high score tracker within local entity on hard disk
    - Must maintain the project website for downloading software and user notes
    - Once system is completed and before the final package is sent, we will still continue to check for errors and update system appropriately
- Performance
    - Our goal is to maintain 24 frames per second even on low end computers
- Portability
    - Send the .jar or .exe to grade
    - We would like any computer with Java to be able to run this program
    - As of right now, we will not plan for Android or Mac support unless we have an abundance of time
- Prototypes
    - We will be using the incremental phased development approach so each new functionality or class will be a new demo/prototype
- Technical Difficulties
    - Getting graphics/sound and implementing them in the system
        - Thadd and Chris Lin are responsible for graphics and sound
        - They will possibly consult existing Java libraries for graphics and sound
        - For any additional materials beyond their capabilities, Thadd and Chris will consult UCF digital media design students
    - Designing an appropriate physics engine to conform to the system requirements
        - We will search for existing Java libraries to simulate our required physics and modify them for our specific needs
    - Getting Git and Github to work as expected
        - Keep trying to figure out the bugs
        - Consult tech support
- Trade-offs in design architecture
    - No Android or Mac support because of time and complexity
    - Simple ship upgrade methods as opposed to a more detailed and graphic intensive alternatives. We would like to use detailed graphics to represent different ships and weapon options, however, because of time demands and possible conflicts with

collision detection, we will opt for simple ship model representations. If time and expertise allows, we will use higher quality models, sounds, etc.
    - We are going to use pre developed Java libraries and modify them to our needs. We would all like the experience in developing our own.
- Rationale for this architecture
    - We are going to use the incremental phased approach which will ensure that we have one functional/working system at all times, improve that system with new functionality, insuring that new functionality will be working before we move to the next phase.
    - There is a limited amount of time, limited experience, and many things that we desire to implement. Our highest priority is getting the minimum requirements done as quickly and efficiently as possible. We wish to focus most of our energy on the design process, therefore, it is counterproductive implement many features outside of the minimum requirements.
- Risks with this solution
    - Using too much of another developers code could result in lazy programming techniques, poor code documentation, or plagiarism.
    - We could get hung up on developing one facet of the program and not move to the next in time to complete the assignment.