Chapter 1

Introduction

The first week is an introductino to the class: objectives, syllabus, schedule, project challenges.

Handouts printout of this chapter and schedule (web page)

1.1 Course overview

Duration 30 min.

1.1.1 Objectives

Participation in a large development project, collaboration, s/w engineering issues; concurrency, synchronization.

Prerequisites Object-oriented programming in C++.

1.1.2 Topics

- Requirements, architectural design, code design, documentation, testing (unit-testing), patterns,
- Concurrency, synchronization, distributed systems,
- Design, human factors, collaboration and collaborative tools, ethics, open source development
- etc.

1.1.3 Project

Collaborative class project: specifics will be determined in the course of the class, based on interest and skills represented in the team. The project will likely be of the level of a technical demonstration rather than a fully developed game.

1.1.4 Homeworks

Individual assignments related to project (between 5 and 10 assignments over the semester).

1.1.5 Text and Readings

Optional textbook (not necessary to complete the course successfully) [FS04]:

> John P. Flynt and Omar Salem, ¡em¿Software Engineering for Game Developers¡/em¿, Muska & Lipman, 2004, ISBN-10: 1592001556.

Suggested readings will be listed for specific lectures/topics.

1.1.6 Labs

Labs will be devoted to programming. The project will be designed so that most of the programming required will be carried during lab time (in order to minimize the amount of work needed outside of class).

1.1.7 Grading

- Homework(s) grade (average): 65%
- Final project presentation: 10%
- Final project release (collective grade): 15%
- Class participation: 10%

1.2 Getting to Know the Team

Once the objectives and main ideas of the course are established, it is time for everybody to get to know the team members.

Students takes 1-5 minute each to share:

- Name (by which they would like to be called)
- Programming classes/experience
- Most complex program written
- Experience in group projects (if any)
- Technical interests for the project

Instructor/project manager gives similar and additional information relevant to the project..

1.3 Project

This section takes a closer look at what it takes (in general) to successfully create a software project [FS04] ch.1: *Getting into the Game.*

1.3.1 Challenges

Re-iterate the objectives of the project:
Complex project: more than just writing a function or two.
Short time, work together (parallel) Need to organize!

Software engineering vs. Engineering [FS04], p.6:

"Technique is a process of refinement that characterizes formalized knowledge"

Time, Energy, Efficiency, Reliability, Maintainability, Appropriateness.
Continuous iterative improvement

Integrity (soundness) Consider:

- Scope
- Architecture
- Planning
- Testing
- Implementation

1.3.2 Steps

- Answering risk with design
  Game design and software design (fig.1.10, p.13)
- Design and development
- Class design and implementation
- Refactoring and patterns
- Development and testing
- Documentation
- Learning and capabilities***
- Maintenance and revision***

1.4 Team Organization

This section takes a closer look at what it is going to take **for the team** to successfully create a software project [FS04] ch.15: *Team Work*.

1.4.1 Tasks

What teams do ([FS04] p.546-547):

- Prepare a software development plan.
- Gather technical requirements for the game and summarize these in a software requirements document.
- Design the software system that supports the features of the game and document this design in a design description.
- Construct the software for the game and create documentation for the code.
- Prepare a test plan for the software and conduct the tests that the plan specifies.
- Prepare a deployment plan for the software and deploy the software.

1.4.2 Communication

Core information pathways for teams (from [FS04] p.557-559):

- Vision statement - A statement that the team might compose together that briefly expresses the purpose and goal the team has set for itself in its development effort. A vision statement is the starting point of common understanding.
- **Change control plan** - An agreement among the members of the team about how change to important work, such as requirements and code, are to be made so that everyone on the team can keep track of them.
- **Change proposals** - An agreement about how team members can propose change. This might be a brief form or a convention involving sending an email to a change control coordinator.
- **Top 10 risk list** - a list of the major dangers to the success of the project. everyone on the team contributes to this list, and knowledge of the list allows everyone to know the ways in which the project is most likely to be derailed.
- **Software development plan** - The schedule that everyone is expected to follow during the development effort. It provides a common view of the milestones, the costs and the scope of the project.

- Interface prototype -
- **User documentation** - Information that's intended for the user, such as an online manual. The primary value of user documentation is that it represents the customer or player viewpoint.
- **Requirements specifications** - The starting place for the whole software development project. The software requirements specification contains the plan of how the software system is to address the requirements.
- **Use cases** - Contexts or stated requirements. You can employ use cases in the design effort to define system performance contexts. You can also use them for testing. Developers, documentation specialists, and testers require ongoing access to use cases to solidify their understanding of the project.
- **Quality assurance plan** - A schedule and protocol for reviews. A quality assurance plan differs from a change control plan because it provides a way for members of the team to review each other's work. It sets up a model, for instance, for code reviews.
- **Software architecture** - The principles and quality measures that the software developers should seek to fold into the software. These principles give shape to the software design description. The software design description provides logical, behavioral, user, and deployment views of the system, views that allow everyone to know the scope and development strategy that
- **Integration procedures** - For the software developers, integration procedures are essential for being able to work together on a day-to-day basis
- **Staged delivery plan**
- **Milestone schedules**
- **Coding standards**
- **Test cases**
- **Source code**
- **Incorporated media**
- **Game design document**
- **Build instructions**
- **Detailed design documents**
- **Installation program**
- **Release check list**
- **Project log**
- Project history

1.4.3 Team Management

[FS04] p. 585

1.4.4 Reasons teams fail

[FS04] p. 572

- Upper management abandonment
- **Methodology shortcuts**
- **Worthless methodologies**
- **Loss of scope**
- **Unreallistic schedule**
- **Poor estimations**
- **Too much enthusiasm**
- Mythical man-month
- Team desintegration
- The compoany goes under
- Lack of resources
- Lack of equipment
- Divergence from the project plan