Final Year Interim Project Report

Scheduling Disciplines Learning Tool

Dominic Carr

A thesis submitted in part fulfilment of the degree of

BA/BSc (hons) in Computer Science

Supervisor: Dr Felix Balado

UCD School of Computer Science and Informatics
College of Engineering Mathematical and Physical Sciences
University College Dublin

February 11, 2010

Table of Contents

Abstract

This project involves the creation of a learning tool to build and enhance computer science students understanding of process scheduling and several of the fundamental scheduling disciplines used in operating systems to enable this. This will be done through an interactive visual e-learning tool. Scheduling disciplines such as Shortest Job First and Round Robin will be simulated. This report will detail and provide summary of the background research which has been conducted as part of the project. Therein the core concepts associated with the project, such as processes, scheduling disciplines etc. will be explained. Following this details of the project implementation process thus far will be provided. Finally a set of project milestones will be given and a timeline for the projects completion detailed.

Project Specification

Subject: Learning tool for visualizing scheduling disciplines
Prerequisites: Operating Systems I
Project Type: Design and Implementation
Software Requirements: Java, C/C++, or any other suitable language
Hardware Requirements: None
Preassigned: No

Description

General Information: The objective of this project is to develop a visual learning tool to demonstrate process scheduling in an operating system. The tool should be interactive, so that the user can follow the evolution in time of the queue in detail.

Mandatory: Design and implementation of a GUI that will allow the user to choose between several scheduling disciplines. The program underlying this GUI will then generate an interactive animation showing how processes arriving at the queue are scheduled and processed. The evolution over time of relevant parameters will be displayed: average waiting time, average turnaround, average queue size, etc. The system must implement at least two scheduling disciplines: First Come First Served (FCFS) and Shortest Job First (SJF). The processes arriving at the queue will be a tunable random mixture of CPU-bound and I/O-bound processes.

Discretionary: Implement the Shortest Remaining Time First (SRTF) and Round Robin (RR) scheduling disciplines.

Exceptional: Implement a module able to create priority queues.

Reading: Lecture notes of Operating Systems I, and references therein.

Chapter 1: **Introduction**

1.1 Project Overview

This project is concerned with the creation and optimization of a learning tool to impart and enhance a students knowledge and understanding of the workings and performance of several operating system (OS) scheduling disciplines. Scheduling is a core issue in Operating System design, and as such is a key concept which students must learn. The project will involve the modeling of a CPU scheduler, processes, queues and various scheduling disciplines and the visualization of the operation of these scheduling disciplines.

The scheduling disciplines which will be simulated are Shortest Job First, First Come First Served, Shortest Remaining Time First and Round Robin with the probable inclusion of a priority queue based discipline. The key points of these disciplines are discussed in the second section of this report. The project will be implemented entirely in the Java programming language.

This learning tool will be composed of a graphical application in which the users can select a scheduling discipline to simulate and modify various parameters such as the simulation speed and the ratio of I/O and CPU bound processes. The student will be able to view the evolution of the queues and system performance over time. The transition of the processes between their various states will be animated and the relevant information regarding each process' properties will be displayed.

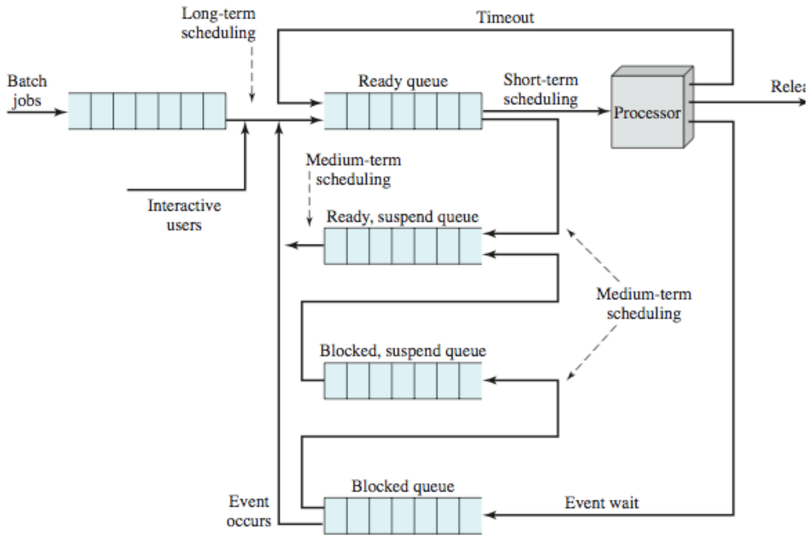

Figure 1.1: Taken from [1] shows many of the elements which will be animated and visualized in the finalized application

1.2 Report Overview

This report is primarily concerned with detailing the background research conducted, and work done thus far towards the development of a fourth year project, in which a learning tool is being developed to educate students about process scheduling and the disciplines (algorithms) employed to facilitate this.

1.2.1 Section 2

In the second section of this report, details of the background research carried out to date will be given. Information will be provided regarding the theoretical research carried out as regards, processes and their behavior, process scheduling, scheduling disciplines and their performance evaluation criteria and how they are typically visualized. Further to this, details will be given about practical research into the implementation language, Java. Some of the useful Java classes which will be used in the project implementation will be discussed. There will also be some discussion of research conducted into the the creation of interactive animations and the software necessary to create the custom imagery needed for the animations and other components of the Graphical User Interface.

1.2.2 Section 3

In the third section of the report a progress report regarding the project implementation thus far will be given. Therein there will be a breakdown of the project implementation into a series of separate, important milestones. The completion of any milestone will mark an important moment in the project development.

Details will be given as regards the milestones which have been completed thus far, and the work that was required to complete them will be briefly elaborated upon. Further to this a detailed plan will be summarized providing information as to approximately when each of the remaining milestones will be completed.

Subsequently some details will be given of the authors plan of action, should the development of the project proceed more quickly than expected. Finally there will some details given on the measures put in place in case of unforeseen difficulties.

Chapter 2: **Background Research**

The primary source for the theoretical background research necessary for this project was [1] by William Stallings, primarily chapter 9 in which Operating System (OS) process scheduling and the various scheduling disciplines under consideration, and the criteria for their assessment, are explained in detail. The knowledge acquired therein was supplemented by [2] by Andrew S. Tanenbaum and the lecture notes of the UCD module Operating Systems 1 (COMP30090) by Dr. Felix Balado and Dr. Tahar Kechadi [3].

2.1 Scheduling Disciplines

2.1.1 What is a scheduling discipline?

In an operating system there are frequently many processes vying for use of the Central Processing Unit (CPU). As such a choice must be made as to which process executes next, this is handled by the short term scheduler which employs an algorithm of varying complexity, the scheduling discipline, to choose the next process to execute. Typically there are two other levels of scheduling: long-term scheduling and medium-term scheduling. The long-term scheduler handles the addmission of processes to the ready queue. The medium term scheduler decides which processes are to be held in main memory (this places a process in the suspended blocked or suspended ready queues).

In the following subsections the five scheduling disciplines which will be implemented and simulated as part of this project will be discussed and some details will be provided as to proposed implementation strategies. Firstly the two distinct categories of scheduling disciplines will be explained.

2.1.2 Non-preemptive Scheduling

In non-preemptive scheduling disciplines, decisions relating to scheduling only occur when a process voluntarily hands over control of the CPU. For example a scheduling decision will be made when the running process finishes its CPU burst or exits the system.

2.1.3 Preemptive Scheduling

In these kind of disciplines in addition to decisions being made when a process relinquishes the CPU, scheduling decisions may also be made based on other external factors such as the arrival of a new process at the ready queue or if the current process is running for too long etc. When these events occur the current process may be preempted.

2.1.4 First Come First Served

This is the simplest and most easily understood scheduling discipline. It is a non-preemptive algorithm. Processes are simply assigned to the CPU in the order that they arrive at the queue of ready processes. After the completion of the current process' CPU burst (or if an I/O interrupt occurs) the process at the head of the queue is executed. Newly arriving process' are placed at the end of the queue as are processes transitioning from blocked to ready i.e. from the blocked queue. As this is the simplest scheduling discipline it was implemented first providing a solid basis for the implementation of the other disciplines, this has, and will, allow for testing of the interactive animation and the other GUI components before development progresses to the other disciplines.

2.1.5 Shortest Job First

In this conceptually simple scheduling discipline, as the name suggests, the scheduler chooses the process on the ready queue with the shortest execution time to run next [2]. This is a non-preemptive algorithm (although it has a preemptive variant discussed later). This discipline either makes the assumption that the run time of a process is known in advance or that each process' next burst time must be estimated based on prior behavior [1]. For the purpose of this project the burst times of processes will be generated randomly within a range of values and a process' next burst time will be estimated, by the scheduler, using formulae detailed below.

As illustrated in [2] the Shortest Job First algorithm is optimal with regards to average process waiting time. This algorithm should be fairly easy to implement. It will be possible to sort the queue of ready processes in order of their projected next burst time as new processes arrive thereby ensuring that the shortest process is always executed next. This could be easily done using the optimized sorting functions from the Java Collections library [12] by providing it with a suitable comparator.

Next burst time calculation
In the case of interactive processes and many batch processes it is simply not feasible to know the next burst time of a process, as such this must be estimated [1] [3]. This is done by using the past behavior of the process as a guideline. The next burst time is typically estimated by way of an exponential average of all the previous burst times of the process [1]. This estimate is calculated as in the following formula, adapted from [3].

$$T_{n+1} = \sum_{k=0}^{(n-1)} (1-\alpha)^k \alpha t_{n-k} + (1-\alpha)^n T_1$$

Where t_n is the length of the process' nth CPU burst and T_{n+1} is the estimated length of the process' $n+1th$ CPU burst and α is bounded $0 \leq \alpha \leq 1$. Assigning different values to α gives varying degrees of weight to the previous burst times. For example if α is assigned the value 1 only the most recent burst time is taken into account [3]. This will be a tunable parameter in the application interface so the user may choose which value to use.

2.1.6 Round Robin

In this nonpreemptive discipline a fixed slice of CPU time, a quantum, is given to each process. If a process is still running when its quantum is up it is preempted and put back at the tail of the queue and the next process runs for the time quantum. In this way the algorithm cycles through the processes in the ready queue in a FIFO manner [1]. If the process finishes execution before the time quantum expires the next process will be run immediately [1].

As discussed at length in [2] the selection of an appropriate quantum is a key issue. it will be interesting to analyze the performance of the Round Robin discipline simulation under various time quantum's. When this algorithm is under simulation there will be additional user interface elements to allow the user to vary the time quantum.

2.1.7 Shortest Remaining Time First

This is the preemptive variant of Shortest Job First scheduling discipline. As such the scheduler always chooses to run the process with the least amount of CPU time remaining [2]. This discipline functions exactly in the manner of Shortest Job First except when a process is added to the ready queue. When this occurs its predicted CPU burst time is checked against the remaining burst time of the current process and if the new process has a shorter burst time the current process is preempted and the new process dispatched. The main implementation consideration for this algorithm is that the scheduler be alerted when a new process arrives and be capable of making a simple comparison between processes.

2.1.8 Priority Queue

In this scheduling discipline a priority, typically a number denoting importance, is given to each process. The scheduler dispatches the process with highest priority to the CPU, when that process relinquishes the CPU (presuming a non-preemptive algorithm) the process of next highest priority is dispatched. Priority based scheduling can be either preemptive or nonpreemptive. A scheduling discipline of this kind can cause processes of lower priority to be starved [1] i.e never receive CPU time. To avoid this problem process priority can be modified over time, this model will be used in the project implementation.

2.2 Processes

A process may be defined in many different ways. The definition most befitting of this project is given in [1], A process is an "entity that can be assigned to and executed on a processor".

As the project brief mentions that there must be a tunable mixture of I/O and CPU bound processes it will be necessary to create two different kinds of process which behave in different ways as regards their burst pattern. This behavior should be easily implemented in Java, via the creation of separate subclasses with varying bounds on the percentage of I/O and CPU time in their total execution time. There will also need to be a significant amount of variability in the process burst times etc so as to give some realism to the simulation. Research was carried out as to how these two kinds of process differ, this is detailed below.

2.2.1 CPU Bursts

These are periods in which a process is executing on the CPU. CPU bound processes have long CPU bursts and proportionally much shorter I/O bursts, CPU bound processes typically consume their allocated CPU time [1].

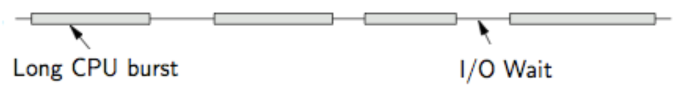

Figure 2.1: CPU Bound process

2.2.2 I/O Bursts

These are periods during which a process is waiting for I/O, for example waiting for user input. I/O bound processes have frequent I/O bursts and as a result they have short CPU bursts times, I/O bound processes typically need to wait for I/O before their allocated CPU has expired [1].

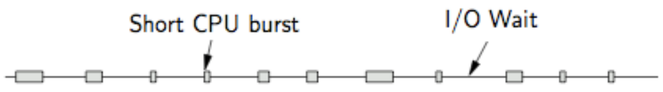

Figure 2.2: I/O Bound process

2.3 Scheduling Parameters

There are many performance evaluation criteria which are employed to assess the performance of a scheduling discipline. For the purposes of this project the most vital of these will be calculated, displayed and updated in the application as time passes.

The various scheduling disciplines discussed previously perform better with regards to certain criteria than others. For example Shortest Job First is optimal with regards to average process waiting time whereas Round Robin performs very well with regards to average response time. The evaluation criteria which will be considered are as follows:

Average Waiting/Response Time
This is the measure of the average interval between a process' submission and the beginning of its execution. This is very important from the users perspective because when a process begins execution they will receive some feedback, so in essence the shorter the average response time the better the user's experience [1].

Average Turnaround Time
This is the measure of the average time taken between process submission and the completion of the process' execution. This is considered as the main criteria for assessment of performance in a batch processing system [2].

Normalized Turnaround Time
This is defined in [1] as the "ratio of turnaround time to service time" where service time is the total amount of CPU time the process required. The formula for its calculation is given below. This criteria characterizes the process' delay relative to the length of its execution time.

$$normalizedturnaround = \frac{turnaround}{servicetime} * \frac{100}{1}$$

Average Queue Size
This is the measure of the average size of the ready queue, It is important as it quantifies the average amount of processes waiting to execute.

Often taken into consideration for evaluation are the maximum and minimum values of the above parameters as well as the standard deviation. As such these values will also be calculated and displayed, or perhaps hidden until the user requests additional statistical detail. The evolution of the above parameters over time will be displayed graphically, most likely by use of a line graph, where the x-axis is the simulation time and the y-axis is the range between the parameters maximum and minimum values. It is important that students be able to understand these parameters and their importance as such some textual explanation will be given.

Throughput
As defined in [1] throughput is the "number of processes completed per unit of time".

Processor Utilization
This is the percentage of the total system time for which the CPU is busy. The formula for utilization calculation is given below. This criteria, while still an important consideration, is less important than those detailed above in regards to a standard single-user system [1].

$$utilization = \frac{busytime}{totaltime} * \frac{100}{1}$$

Processor Idle time
This is the percentage of total system time for which the CPU is idle. It is given by the formula below.

$$idletime = \frac{timeidle}{totaltime} * \frac{100}{1}$$

As to provide some quantification and context to the above parameters the total simulation time will also be displayed.

2.4 Implementation Research

As this project is of highly practical nature it was necessary to conduct a degree of research into the best language and tools that could be used for the implementation component of the project. Several implementation languages were considered such as C++ and Python, however it was decided that Java would be the most appropriate, as it is cross-platform and has extensive and well-documented graphics libraries. C++ and Python have similar capabilities however C++ lacks a platform independent or well-documented graphics library and Python suffers from compatibility issues, related to different releases of the language.

Modules were identified in the Swing [4] and Java2D [11] libraries, which appear to have all the necessary tools to facilitate the creation of a professional quality Graphical User Interface. As discussed below, The Java2D library is particularly suitable for the creation of animations. Proper usage of the object-oriented model of Java will allow for easy extension of the application for the inclusion of more scheduling disciplines if time should allow for this.

2.4.1 Animation In Java

It was necessary that research be conducted into the tools available for the creation of animations within the Java programming language, as this was an area relatively unknown to the author. This involved the creation of some small rudimentary animations and the analysis of the Java2D library documentation contained in [11].

2.4.2 Timer

The Swing Timer class [6] provides a very simple means to model the passage of time in the scheduling simulation. Further to this it allows for variation in the time between updates, so in essence the simulation can be sped up or slowed down, in response to user demands which aids interactivity. This class also provides an easily implemented means of pausing and resuming the application as and when the user requests this.

2.4.3 Observer-Observable

The Java implementation of the Observer-Observable pattern [5], which is included in the standard Java library, allows for easy one way communication between a set of objects and their observer. In the case of this project it allows the processes to communicate with the scheduler e.g. to convey an I/O interrupt so the scheduler can move the process to another queue and free up the CPU.

2.4.4 Image Software

Research was conducted into image editing and creation software. This was necessary as this project has a significant graphical component attached to it and it is highly important that this interface be of a professional quality while still maintaining clarity. This research concluded that Photoshop would be the software most suitable for the construction of the custom graphical elements desired for the project.

2.5 Learning Tool

As the primary goal of this project is the creation of a successful learning tool it was identified early on that there should be a period of user testing. During this period a number of individuals will use the application and assess its ease of use and teaching merit. It is hoped that the feedback received from this will enable the improvement of the tool. The results of the research conducted into the area of beta testing have formed the beginnings of a questionnaire for those testing the prototype. In [9] and [10] some useful user interface design guidelines were provided which informed the layout of components in the initial GUI prototype. The primary lesson learned from these materials was that user interfaces should be developed such that they mirror somewhat other interfaces which the user's are comfortable with.

2.6 Visualisation Techniques

In trying to identify an appropriate visualization for all the components of operating system scheduling the author looked at the various ways in which seminal texts in the field had visualized these components in still imagery. In this way it was hoped a clear visualization could be developed which would be easily comprehensible to students. Many of the texts such as [1] and [2] employed similar imagery, which is simultaneously simple, clear and effective. Various animations of process scheduling were analyzed, particularly helpful was a clear and simple animation accessed at [7], notable also as it is targeted towards individuals perhaps unfamiliar with the subject matter. All of this will feed into the final interactive animation design, which has not as yet been fully finalized. Upon completion of background research into the various scheduling disciplines the conclusion was arrived at that while they operate differently they can be visualized in much the same way with alterations being made in each case to emphasise their unique manner of operation, perhaps through some combination of textual explanation and graphical pointers. Figure 2.3 shows the prototype GUI which has been developed. Processes waiting in the ready queue can be seen as well as a process in execution on the CPU. Also visible are the lines of transition which processes can follow and the various other queues.

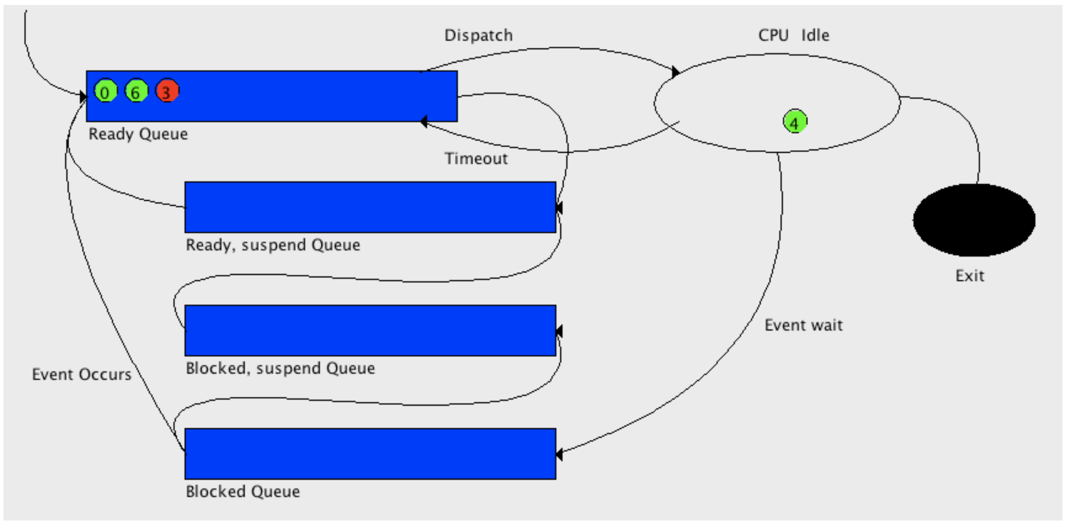

Figure 2.3: Learning tool prototype Graphical User Interface

Chapter 3: **Progress Report**

3.1 Milestones

Before work began on the project a list of project milestones was constructed which correspond to completion of a major component of the project work. This was done in an effort to structure the development of the project in an incremental manner.

1. Background reading completed
2. Project problem domain modelled, class hierarchy with skeleton methods created, communication mechanisms in place.
3. Rudimentary discipline selection and parameter selection Graphical User Interface (GUI) completed
4. Textual implementation of Round Robin and First Come First Served
5. Prototype animation working with Round Robin and First Come First Served and final report started
6. Implementation of Shortest Job First, Priority Queue scheduling and Shortest Remaining Time First and additional user testing. user testing undertaken and evaluation of user response.
7. Completed interactive animation
8. Completed final application
9. Final report completed

3.2 Progress

At this juncture in the project development process milestones 1-4 have been completed. All the required reading for the project has been undertaken, and can be referred back to as necessary. Of course as complexities appear more reading may be necessary. The project domain has been modeled in Java with a full hierarchical class and communication structure in place, this has been well documented (by way of Javadoc). This design has been done in an object-orientated manner to remove repetition of code and to allow easy extension for the discretionary and exceptional criteria of the project specification.

A Graphical User Interface (GUI) has been constructed which allows the user to tune a wide variety of scheduling parameters e.g. the percentage of I/O bound versus CPU bound processes in the simulation, the time quantum, the simulation speed etc. Additionally the user may choose a scheduling discipline to simulate. This GUI also supports the display of the relevant performance evaluation criteria. Milestone 4 is complete with the implementation of the First Come First Served and Round Robin disciplines complete.

Additionally a significant portion of work has been completed on milestone 5 as regards the GUI components for the representation of processes, queues, I/O waits and the CPU. A plan for the testing of an initial prototype with several users has also been devised. The final report is in the planning stages.

3.3 Remaining Work

A significant portion of the project work remains to be completed. The remaining scheduling disciplines must be fully coded and rigorously tested, and they must be fully integrated with the scheduler and the existing GUI. The evaluation criteria will need to be calculated and their display updated. The interactive animation needs to be finalized and implemented, debugged and integrated.

A plan will need to be devised for the final report and work will begin on adapting the research included herein for inclusion in the final report, this research may need to be expanded upon. There will need to be time allocated for user testing and review of the received feedback. This feedback may result in some changes being made to the user interface as regards the manner in which information is displayed or the degree of explanation given.

3.4 Timeline

It is my aim that the remainder of milestone 5 be fully completed by Sunday February 28th. I envision that milestone 6 will be completed by the end of March or early April with the completion of milestone 7 following shortly there after. Milestone 8 will be completed by second last week of April, which will create a buffer against any unforeseen difficulties which may arise in the final, crucial, stages of the implementation process.

Over this time period work will also be conducted working towards the last milestone, the completion of the Final Report, It is my hope to have an early draft of this report completed by late April for review by my project co-ordinator. It is hoped the penultimate draft of the report will be completed by May 2nd at the latest so as to allow time for any last minute changes or corrections deemed necessary by my project co-ordinator or myself.

If at any time development proceeds more quickly than expected then more time will be allocated to non-essential tasks such as GUI fine tuning, the introduction of keyboard nemonics or the creation of a range of application deployment options (such as a mac OSX application bundle, executable Jar file or a Java applet [8]). I feel these additions could strengthen the project as a learning tool for students.

Similarly if development proceeds more slowly at any point due to unforeseen problems more effort and time will be expended in order to complete that particular task. The timeline as envisioned above allows for some leeway as regards the completion of most milestones, this is due to the fact that it is very difficult to plan for problems in a project of this size.

Bibliography

[1] William Stallings. *Operating Systems: Internals and Design Principles.* 781 pages. ISBN: 0138874077. Prentice Hall, 1997.

[2] Andrew S. Tanenbaum. *Modern Operating Systems.* 1104 pages. ISBN: 0136006639. Prentice Hall, 2007.

[3] Dr. Felix Balado and Dr. Tahar Kechadi. *Operating Systems 1 Lecture Notes.*

[4] James Elliott et al. *Java Swing, Second Edition.* 1280 pages. ISBN:0596004087. O'Reilly Media, 2002.

[5] http://java.sun.com/j2se/1.4.2/docs/api/java/util/Observable.html

[6] http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/Timer.html

[7] http://computer.howstuffworks.com/operating-system7.htm

[8] http://developer.apple.com/mac/library/documentation/Java/Conceptual/Java14Development/03-JavaDeployment/JavaDeployment.html

[9] http://developer.apple.com/Mac/library/documentation/UserExperience/Conceptual/AppleHIGuideli

[10] Joel Spolsky. *User Interface Design for Programmers* 159 pages. ISBN: 1893115941. Apress, 2001.

[11] Jonathan Knudsen. *Java 2D Graphics* 339 pages. ISBN: 1565924843. O'Reilly Media, 1999.

[12] John Zukowski. *Java Collections* 420 pages. ISBN: 1893115925. Apress, 2001.