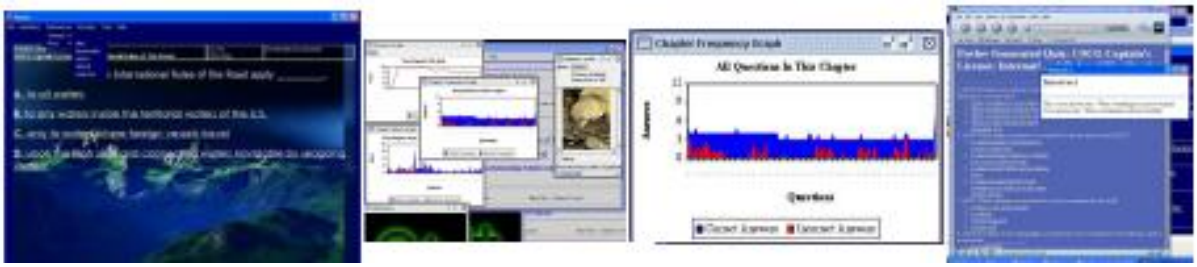
Pavlov Documentation

1. About

1.1. Pavlov Home

1.1.1. What is Pavlov?



Pavlov Screenshot

We've all made decks of flashcards and slogged through them to study (oftentimes boring) material. Wouldn't it be great if the flashcards magically shuffled themselves to optimize your study time? Wouldn't it be great if they sung to you and danced and entertained you while you studied?

This is Pavlov - efficient, entertaining study.

Pavlov uses easy-to-write "pluglets" for feedback mechanisms (entertainment) and question choice strategies (efficiency). Just pop a new pluglet in the right directory to expand your Pavlov universe.

1.1.2. PAVLOV 1.1.2 Now Available

(30 Jun 04) Download the latest version of Pavlov and enjoy -- it's FREE SOFTWARE. New features include:

- 165,000 questions in sample language learning books are now available at http://pavlov.sf.net
- Books and chapters are now sorted alphabetically
- Several improvements to the "Incorrect Answer Dialog"

- Startup is quicker
- Pavlov consumes memory less quickly when dealing with huge books
- Export a quiz to any file format using pluggable templates. Included templates include a simple Web Application and two static HTML examples.
- HTML+ scriptable feedback pluglets
- HTML+ scriptable application skins

If you murder a test using Pavlov, please drop us a note! If you like the program, please rate it at freshmeat.net.

1.1.3. 165,000 Language Questions Now Available

The Pavlov Project, with help from The JDict project and the European software company Majstro Aplikajoj are pleased to provide a set of huge language learning question books. These books are "bidirectional," so that, for example, the "English/French" book could be used by an English speaker to learn French or a French speaker to learn English. The files can be downloaded at the Pavlov Project Page.

- German/Spanish: 16,000 questions: pav-de-es_2a
- German/French: 27,000 questions: pav-de-fr_2a
- German/Russian: 4,000 questions: pav-de-ru_2a
- German/Slovak: 1,400 questions: pav-de-sk_2a
- German/Turkish: 6,300 questions: pav-de-tr_2a
- English/German: 39,846 questions: pav-en-de_2a
- English/French: 30,408 questions: pav-en-fr
- English/Hawaiian: 200 questions: pav-en-ha
- English/Hebrew: 700 questions: pav-en-he
- English/Italian: 13,000 questions: pav-en-it
- English/Latin: 10,000 questions: pav-en-la
- English/Mandarin Chinese: 4,000 questions: pav-en-ma
- English/Modern Greek: 3,000 questions: pav-en-ng
- English/Russian: 4,300 questions: pav-en-ru_2a
- English/Zulu: 3,500 questions: pav-en-zu
- French/Russian: 3,100 questions: pav-fr-ru_2a

A much larger set of books should be available in September 2004. All books are in XML/Unicode format and distributed under the GNU General Public License.

1.1.4. Where's The Project Manager?

The Pavlov Project Manager, T.J. Willis, will be in Alaska, probably completely cut off from the Internet during July and August 2004. Direct emails will almost certainly not be answered during this time. Questions may be sent to the Pavlov Developer's Mailing List

pavlov-devel@lists.sourceforge.net.

1.2. Pavlov News

1.2.1. PAVLOV 1.1.2 Now Available

(30 Jun 04) Download the latest version of Pavlov and enjoy -- it's FREE SOFTWARE. New features include:

- 165,000 questions in sample language learning books are now available at http://pavlov.sf.net
- Books and chapters are now sorted alphabetically
- Several improvements to the "Incorrect Answer Dialog"
- Startup is quicker
- Pavlov consumes memory less quickly when dealing with huge books
- Export a quiz to any file format using pluggable templates. Included templates include a simple Web Application and two static HTML examples.
- HTML+ scriptable feedback pluglets
- HTML+ scriptable application skins

If you murder a test using Pavlov, please drop us a note! If you like the program, please rate it at freshmeat.net.

1.2.2. 165,000 Language Questions Now Available

The Pavlov Project, with help from The JDict project and the European software company Majstro Aplikajoj, is pleased to provide a set of huge language learning question books. These books are "bidirectional," so that, for example, the "English/French" book could be used by an English speaker to learn French or a French speaker to learn English. The files can be downloaded at the Pavlov Project Page.

- German/Spanish: 16,000 questions: pav-de-es_2a
- German/French: 27,000 questions: pav-de-fr_2a
- German/Russian: 4,000 questions: pav-de-ru_2a
- German/Slovak: 1,400 questions: pav-de-sk_2a
- German/Turkish: 6,300 questions: pav-de-tr_2a
- English/German: 39,846 questions: pav-en-de_2a
- English/French: 30,408 questions: pav-en-fr
- English/Hawaiian: 200 questions: pav-en-ha
- English/Hebrew: 700 questions: pav-en-he
- English/Italian: 13,000 questions: pav-en-it
- English/Latin: 10,000 questions: pav-en-la

- English/Mandarin Chinese: 4,000 questions: pav-en-ma
- English/Modern Greek: 3,000 questions: pav-en-ng
- English/Russian: 4,300 questions: pav-en-ru_2a
- English/Zulu: 3,500 questions: pav-en-zu
- French/Russian: 3,100 questions: pav-fr-ru_2a

A much larger set of books should be available in September 2004. All books are in XML/Unicode format and are distributed under the GNU General Public License.

1.2.3. Where's The Project Manager?

The Pavlov Project Manager, T.J. Willis, will be in Alaska, probably completely cut off from the Internet during July and August 2004. Direct emails will almost certainly not be answered during this time. Questions may be sent to the Pavlov Developer's Mailing List pavlov-devel@lists.sourceforge.net.

1.2.4. Pavlov User Exchange Online

(27 May 04) We've hacked together a facility to allow users to exchange books, themes, skins, quiz export templates, and velocity feedback pluglets using sourceforge's "Patch" facility. Look for the "Pavlov Exchange" links on the navigation bar to the left for direct links.

1.2.5. Coding Conventions and Developer Info Online

(17 Mar 04) A document for developers discussing tools, coding conventions, and best practices is now available on the Pavlov site.

1.2.6. Pavlov 1.0 Tutorial is Online

(18 Feb 04) Want to check out how it works before you download it? Have you used it and want to explore the advanced features? Check out our virtual tour.

1.2.7. BEE Editor Beta Release

BEE Logo

(16 Feb 04) BEE is a tool to easily and efficiently create questions for Pavlov and store them in the framework of books and chapters provided by the Pavlov Book XML Document Type Definition. The best thing is that you don't have to know anything about XML to use BEE!

Of course, as an open standard, you can use any sort of XML editor you want (even WordPad in Windows, if you really want) to edit Pavlov Book files. But BEE is specially designed to let you spend less time editing and more time learning (or making your victims learn).

1.2.8. BEE Editor Tutorial

(8 Feb 04) Take a virtual tour of the BEE Editor Preview Release online. Available HTML format.

1.2.9. PAVLOV 1.1B1 API Documentation Online

(17 May 04) Pavlov is not just a program, it's software. If you're a Java developer and you want to do some of the things Pavlov does in your own program, you can use the Application Programmer's Interface in your own programs under the terms of the GNU General Public License. Note that the API docs are linked to the 1.1B1 source code: click a method name to see the source code.

1.2.10. How To Write Velocity Pluglets

(18 May 04) The soup-to-nuts guide on how to write a Velocity Pluglet for Pavlov. If you can write a HTML page, you can write a Pluglet, and customize the bejesus out of your learning experience.

1.3. Installing Pavlov

1.3.1. Installing Pavlov

Note:
Updated 22 May 04 For 1.1X Releases

First off, you have to have Java (version 1.5.0B1 or later) installed on your computer. You can download it at http://www.java.com.

Note:
Pavlov 1.0 only required Java 1.4.

Now download Pavlov. Pavlov releases are distributed in familiar-acting installers. The full installation is about 6 MB, so you should have 12 MB of available disk space to install it. Pavlov takes about 8MB of RAM to run, but big pluglets can increase that to around 30MB of RAM.

Download the appropriate installer. The ".jar" installer runs on any java-capable computer. On PC's and Macs you should be able just to double-click the ".jar" installer and be off and running.

> **Note:**
> Experience has shown that the ".jar" installer works slightly better than the ".exe" installer. (For you experts: the .exe asks the user to type in the location of java.exe if it can't find it, which might be a bit intimidating.)

Since the 1.0 release, the installer will create two files "Pavlov.bat" and "Pavlov_guest.bat" (.sh in Unix). You can double-click on these to run Pavlov. There is also a Bee.bat file for creating books of your own questions.

> **Note:**
> Double-clicking the "pavlov-1.XX.jar" file probably won't work, as it won't load all the jars in Pavlov's "lib" subdirectory.

If it doesn't work, your computer probably doesn't know where to find the java program (java.exe on windows). You can edit the run.bat or run.sh files to tell your computer where to find java.

> **Note:**
> If Pavlov starts to run, then freaks out, you're probably using an older version of Java. (If you run Pavlov from a console, it's not unusual to see a bit of "spam," especially in the Beta releases.)

If you're having problems with the installer or understanding the documentation, try our Forum to see if someone else has already answered the question, or tell us what's going on and we can try to help. Nobody here is going to give you a hard time for asking a question.

1.4. Pavlov Project Help Wanted Page

1.4.1. Contributing (your time) to Pavlov

There are many many ways that you, yes *you* can get involved in the Pavlov project. Here are a few:

1.4.2. Be an Active User

If you could get your browser to this web page, you should be plenty savvy enough to use Pavlov productively and think of things that need to be done, and ways it can be used in your world.

- Show it to your teacher.
- Submit your questions to this site so others can benefit from your work.
- Write an essay on Pavlovian Conditioning or Operant Conditioning.
- Look in the forum and see if you can help other users get started, or make suggestions.
- Let us know if you nailed the spelling test or your dissertation defense with Pavlov's help.

The sky's the limit!

1.4.3. Write a Pluglet

Pavlov's strengths lie in the pluggable nature of its feedback and question selection strategies. Pavlov would be a lot more user-friendly, and useful, with 100 pluglets available instead of 15. Someone new to programming in Java or programming period could very well write pluglets. I'd recommend this: go to java.sun.com and download Java (the name will be something like "J2SE1.5.0B..."). Look through their documentation on how to write an Applet. (Applets and Pluglets are not the same thing, but they're kissing cousins.) Then come back here and look at the Pluglet documentation. A lot of things will look familiar. Then do what every new programmer does when in a new environment: read the documentation and experiment. It's a lot of fun.

Note:

With Pavlov version 1.1 you don't have to know Java to write a pluglet. Look for the documentation on how to write a Velocity Pluglet -- use HTML and a little programming to make a feedback pluglet!

1.4.4. Do Science

I'd love to have a professional educator or psychologist do a case study on Pavlov.

1.4.5. Hack The Core

There is need for experienced programmers to improve and expand the core Pavlov code. It's mostly in good shape, but there are API modifications and implementation cleanups lurking around that another pair of eyes would see before I would. Expansion brings up some really interesting problems, like how to use AbstractUIFactory to run Pavlov on a Palm-Pilot, how to back data to a RDBMS, how to implement in various client-server settings, etc. There are cobwebs in some packages that really need to be swept away.

1.4.6. Steal This Code

Well, use it in accordance with the GNU General Public License, but it's almost the same thing. The best way to see if an API is flexible enough is to try to use it in a completely different setting. I've had to make about 6 changes in writing BEE, which is almost exactly the same application as Pavlov.

1.5. A Brief History of Pavlov

1.5.1. Pavlov's History

Pavlov has been in private development for 4 years. Bored with writing and flipping flash cards for the United States Coast Guard's Captain's license, I wrote a Perl Script to do it and hacked in statistical choice of questions and question history logging.

After a hard disk crash in 2000, I (grudgingly) reimplemented it in Java.

Noticing how successful HotOrNot.com was at keeping users (i.e. me and my neighbor Josh) clicking their buttons, the idea occured to me to have my program show pictures of bikini-babes when the user got correct answers. This proved extremely successful with my Navy shipmates who borrowed Pavlov to study for advancement examinations and Enlisted Surface Warfare boards.

As the program grew in complexity, I went through several stages of refactoring and API refinement. Writing plugin support for "question selection strategies" and "feedback pluglets" allowed me to simultaneously decrease the core source code complexity and increase its feature set. Now, it's not "bikini-babe-centric," anyone who can write a normal applet can use his imagination and write a feedback pluglet to provide positive and negative support based on a wide variety of historical data that's stored about the user.

Note:

With Pavlov 1.1X, you don't need to be able to write a Java program to create a pluglet. Now you can do it using HTML and some minor programmatic content using the Velocity Template Language.

Question selection strategies, similarly, are only a matter of a programmer using his imagination, writing a simple program, and dropping the class or JAR file into the `pluglets/strategy/` directory.

The core source is hovering around 130 Java classes and 14,000 lines of code. Heavily used technologies include XML, Swing, and runtime class loading. It has been the target of several bouts of intense refactoring. Abstraction and design patterns are the rule rather than the exception throughout the code.

Note:

> With 1.1B1 it's more like 100 classes and 16,000 lines of code.

Pavlov's evolution has resulted in several packages suitable for general use that are available under the GNU General Public License (GPL).

- `pavlov.randommedia` is a (more-or-less) polished API for picking random or sequenced media objects (images/sound/html/etc) from directory trees.
- `net.sourceforge.steelme` (formerly `pavlov.themes`) is a polished API for user customizable GUI color and font themes.
- `pavlov.pluglets` would be very easy to use in any other Swing program to provide pluggable tools.
- `pavlov.startup` provides slick & GPL'd startup windows for Java applications.

My main goals in sourceforging this project are:

1. make the Pavlov program and API's available to users and programmer-users,
2. get suggestions/assistance on problems remaining in the code and/or architecture,
3. provide a home for pluglet developers to create and distribute feedback & question strategy pluglets,
4. to provide a home for the collection and distribution of content: Library files of questions, and,
5. to encourage different implementations for abstract classes, i.e. DBMS storage of questions and statistics, a servlet Abstract UIFactory implementation, PDA-friendly implementations, etc.

If you've read this far, you are either a very bored person or a potential contributor. If you'd like to take part of Pavlov and run with it, whether it be part of the core, pluglets, writing a "book" of questions, documentation, a study on Pavlov's efficacy in the classroom, or whatever, there's plenty of room in the project for you. Any contributor who would like to get his or her name mentioned has but to ask.

1.6. Download

2. Other Pavlov Sites

3. How To

3.1. Pavlov Code Conventions

3.1.1. Pavlov for Developers

As more people start working with the Pavlov code, it's important to recognize and maintain

the strengths in the code and to attack its weaknesses. If something in this document doesn't make sense to you, that's OK -- do a quick web search to try to get an explanation. If that doesn't work, feel free to post a message on the Pavlov forums.

3.1.2. Tools

The following tools are necessary or useful to have installed on your computer:

- Java Development Kit JDK1.5.0B1 or later (needed to build Pavlov)
- Ant (needed to compile Pavlov)
- A CVS client of your choice (needed to check code in and out)
- JUnit (only needed to write/run tests)
- A reasonable text editor or integrated developement environment

> **Note:**
> If you don't intend to recompile Pavlov (i.e. if you aren't a programmer), you can get JRE1.5.0B1 or later -- the Java Runtime Environment. This is much smaller than JDK1.5.0B1.

3.1.3. Readable Code

Let's start with the following code conventions:

1. Avoid special characters (smart-quotes, copyright signs, etc) at all costs
2. Indent your code so that it's readable
3. Avoid going over 80 characters per line
4. Label problem areas in code with `"//FIXME: Description of problem"`
5. Name temporary variables better than TJ does

The following depend on your IDE. When Ant's checkstyle task can handle tiger (jdk1.5) code, it will become much easier to enforce.

- Avoid uploading source with TAB characters
- Avoid uploading source with MS-DOS linebreaks ("\r")

3.1.4. Good Code

Try to adhere to these rules as much as possible. If you see an instance where a rule is violated, at least add a `FIXME` or else fix the problem. Many of these come from Joshua Bloch's book "Effective Java Programming Language Guide."

1. [Bloch01 #47] Don't ignore exceptions
2. [Bloch01 #12] Minimize the accessibility of classes and members
3. [Bloch01 #23] Check parameters for validity.
4. [Bloch01 #38] Adhere to generally accepted naming conventions

5. [Bloch01 #29] Minimize scope of local variables
6. [Bloch01 #28] Write doc comments for all exposed API elements
7. [Bloch01 #34] Refer to objects by their interfaces
8. [Bloch01 #33] Beware the performance of string catenation
9. [Bloch01 #15] Design and document for inheritence or else prohibit it
10. Use final variables and parameters where possible.
11. Use tiger-style looping as possible
12. Use tiger-style typechecking as possible
13. Strive to decrease `javac -Xlint` warnings
14. Write unit tests as possible
15. Use assertions to check preconditions and postconditions
16. Use log4j instead of `System.out.println`
17. If a method doesn't depend on the object's state, make it static

3.1.5. Good Documentation

Even with a great deal of recent effort, Pavlov's documentation is not in great shape. Check out How_To_Write_Doc_Comments_for_Javadoc to learn how to do it right. Apart from JavaDocs, it's good to have good, normal comments in your code, like:

```
070  /**
071   * Returns the next question answered fewest times or null if problem.
072   *
073   * @return a <code>QuestionData</code> value
074   */
075  public QuestionData getQuestion(Vector<QuestionData> exclusion) {
076
077    QuestionData q = null;
078
079    int lowest = 10000000; // FIXME: should be Integer.MAX_VALUE
080
081    for(QuestionData x: questions) {
082        if(exclusion!=null)
083            if(exclusion.contains( x)) // if this question is in the exclusion vecto
084                continue; // go to next question
085        if (x.getTotal() < lowest) {  // this one's been asked fewest times
086            lowest = x.getTotal();
087            if(lowest==0) return x;
088            q = x;
089        }
090    }
091    return q; // question with fewest answers or null
092  }
```

3.1.6. Hey! Let's look at that example!

Since it's sitting there, let's make some comments on it.

- Line 71, "if problem" should be explained better
- Line 73, not bad, has return value documented
- Line 75, uses tiger style type checking on the vector (rule 12)
- Line 75, exclusion isn't changed by this method, should be a final parameter (rule 10)
- Line 79, the FIXME reminds us to come back to that later (a randomly selected big number is not as good as the biggest number possible)
- Line 81, tiger style looping (rule 11), good
- Lines 82-83 should be combined into one if statement
- Lines 82-84, if blocks should have braces (C programmers do this all the time)
- Lines 85-86, should calculate `x.getTotal()` once and store in (final) temp var (rule 10)
- Line 75, could substitute `AbstractList` for `Vector` ((tricky) rule 7)

This is what "code cleaning" is about, and while it's not as glamourous as some other aspects of development, it cuts down on bugs, makes the code faster, and shows everybody that you're smarter than the original developer. :)

3.1.7. Design

Object Oriented Design (OOD) isn't a newbie topic, but an understanding of some design patterns will make parts of the Pavlov code easier to understand. You can read about these on the Web, in the famous "Gang Of Four" book "Design Patterns", or [Stelting02]. The following patterns are often used.

- Strategy (aka Method-Object). Example: user.BasicStrategy.
- Template Method. Used all over the place.
- Observer (aka Listener). Example: event.AnswerListener.
- Model-view-controller. Almost the entire user interface.
- Abstract Factory. main.AbstractTemplateKit is one example.
- (Static) Factory Method. Used often.
- Builder. main.standalone.SwingUIBuilder aspires to be a real Builder.
- Iterator. Many, many classes use iterators.
- Callback. Example: LoginController calls AbstractPavlovApplication back.
- Composite. Used often.
- Plugin (aka Create-A-Bunch-Of-Strategy-Classes-From-Files-In-A-Directory-At-Runtime-Pattern). Probably a dozen cases of this throughout the code, from skins, to themes, to question selection strategies, to feedback pluglets.

3.1.8. Baby Steps

To many programmers, this document, not to mention the size of the Pavlov code base may

seem overwhelming. When you make a task for yourself, try to make it small enough to handle in one session. For example the task "I'm going to decrease the scope of all the variables in pavlov" is too big. I often pick a subpackage, like pavlov.user or pavlov.library and bang on it, compile, get rid of Xlint warnings, run the program for a while, then check my code in. The longer your code is different than the CVS repository, the more of a chance of a check-in collision (count yourself lucky if you don't know what that means) or breaking the build. These problems are bound to happen, but we should try to minimize them as much as possible.

3.1.9. Keep it Friendly

I started working on Open Source projects in about 1990. Some projects were great because the developer community was talented and supportive of each other -- everybody learned a lot, had fun, and made cool software. In others, conflicts arose and things got nasty. (I remember one morning when a developer at Lawrence Livermore National Laboratory got a 80-gigabyte email for breaking a build.) As this community grows, I require that it remain courteous. On the other hand, sometimes people won't be able to help in what you feel is a timely manner, and you may feel offended. One of the things that project work like this teaches aspiring programmers is to be thick skinned and self reliant.

3.1.10. Recommended Environment

If you're a seasoned developer, you probably have strong views about what development environment you use. I don't have any interest in getting into an argument like "xjpicovim rules and winusurper sucks!" But, if you are starting out from scratch, I can make a couple of suggestions.

Install linux! (I'm almost kidding.)

Linux is a great environment, it's free-no-cost and free-liberty. Blah, blah, blah. It takes some work and some disk space to get it installed and it takes a while to get used to it. But in the long term, its worth it.

But, you want to get started hacking out code quick. Cygwin-X provides a set of tools that make your computer "feel like" Linux with XWindows and it's pretty painless to install. You can get all the tools you need to work with sourceforge (ftp, ssh, scp, cvs) and great time-tested editors like XEmacs and vim. Getting to know these tools, especially if you're a college student (in math, physics, computer science, engineering), is really, really useful.

There are some open-source Java IDE's that are probably worth looking at, but I don't have an informed opinion on them. At the least, they should provide text highlighting, auto indenting, and CVS support.

Note:

Since I wrote this, I've downloaded and installed jEdit. When its plugins begin to support JDK1.5 better, I'll be able to wholeheartedly recommend it. It rocks for JDK1.4 code. It's also very good for XML code (such as editing Pavlov books.)

3.1.11. Bibliography

These are books I've drawn (heavily) on in preparing this document.

[Bloch01] Bloch, Joshua. *Effective Java Programming Guide*. Addison-Wesley, Boston, MA, 2001.

[Stelting02] Stelting, Stephen and Maassen, Olav. *Applied Java Patterns.* Sun Microsystems Press, Palo Alto, CA, 2002.

[Fowler04] Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D. *Refactoring: Improving the Design of Existing Code*. O'Reilly and Associates. 2004.

[Simmons04] Simmons, Robert. *Hard Core Java*. O'Reilly and Associates. 2004.

3.2. Pavlov For Developers FAQ

3.2.1. External Tools

These are some of the tools that are used in and around the Pavlov project. You don't necessarily need any of them to be an active developer, although Ant and a CVS client are pretty close to necessary.

Note:

The mention of any software in this document is not meant to imply that that software's creators endorse Pavlov. This document is instructional, not promotional in nature. This note addresses an item in some versions of the BSD and Apache licenses.

3.2.1.1. Apache Ant

Apache Ant Project

Ant simplifies compiling Java files to class files, and a host of other, similar tasks. It's basically a Java-centric replacement of the UNIX "Make" command. Ant uses an XML file, usually called "build.xml" to get rules for compiling a whole tree of files at once.

3.2.1.2. Apache Forrest

Apache Forrest Project

Apache Forrest is used to maintain the Pavlov web site. All the pretty HTML and links and stuff is generated automatically by Forrest. The web site documents, like this one, are generated from XML documents which focus on content rather than form.

3.2.1.3. Apache Log4j

Apache Log4j Site

Log4J is an open-source system for logging, i.e. reporting errors, warnings, informational messages, etc. It replaces "System.out.println" for debugging code, while adding a great deal of extra functionality.

3.2.1.4. CVS

CVS For New Users

Jim Blandy's Intro To CVS

sourceforge.net's CVS info (See section F)

CVS stands for "Concurrent Versioning System." It allows several people to edit a file or set of files at the same time and minimizes "collisions."

In the old days, people used to use RCS (Revision Control System), where they would check out a file, work on it, and then check it in. Nobody else could edit the file while it was checked out. People would go on holiday with a file checked out and return to find their house burnt down by a mob of angry co-contributors.

CVS takes a little getting used to, and is by no means the easiest program in the world to get installed. It is, however, one of the most useful programs in the world to use for collaborative development.

3.2.1.5. emacs, jedit, vim, joe, pico, xedit, yadda yadda yadda

GNU Emacs Site

jEdit Site

These are text editors you'll hear referred to every once in a while. While it's possible to edit a Java program file, XML document, HTML document, etc in a commercial word processor, text editors such as these are geared towards doing it more efficiently and cleanly. Developers get extremely attached to their text editors. Few people know this, but the siege

at Troy was started when Paris told Menelaus that vim was better than emacs. Zeus, the thunder maker and supreme god, knew that emacs was best and gave victory to the Greeks.

3.2.1.6. ICQ (I seek you)

ICQ Site

ICQ (say it out loud) is a free communication tool, that includes functionality like instant messaging. ICQ has been vital and free for at least 8 years. ICQ is recommended for project members.

3.2.1.7. JUnit

JUnit.org

JUnit is a framework for writing unit tests. Avoiding formality, a unit test simply tests a chunk of code, usually a method, to determine if it functions correctly in a variety of situations. JUnit has been found to be extremely useful when multiple developers are working on a program, and is considered by many to be a cornerstone of eXtreme Programming. Pavlov's unit tests, such as they are, are located in the test subdirectory.

3.2.1.8. UMLGraph

UMLGraph Site

UMLGraph generates UML diagrams from doclet tags. If you see a doclet tag in Pavlov code that you're not familiar with, it's likely a UMLGraph tag. It's a back-burner project to get good UML diagrams from UMLGraph.

3.2.1.9. VAInstall

VAInstall Site

VAInstall is a GPL-ed program that creates installers, including all of Pavlov's installers. The configuration files that specify how the Pavlov installers are created are located in the pavlov/vai directory in the CVS repository.

3.2.1.10. PMD, JavaStyle, CheckStyle, Jalopy, and So Forth

PMD Site

JavaStyle Page

CheckStyle Site

I'm grouping these together as "code improvement" tools. They analyze your code and either reformat it or make recommendations for improving it. I'm not aware of any similar tool that works with JDK1.5 constructs, and many have problems with 1.4 code (which is to say asserts). The picosecond that any of these tools starts to work with 1.5 code, I will start using it again.

3.2.2. Bundled API's

It's a hallmark not only of laziness, but of good design to not reinvent the wheel. Since Pavlov is distributed under the GNU General Public License, it can leverage software distributed under several open source licenses. Here is a quick intro to them.

> **Note:**
> The mention of any software in this document is not meant to imply that that software's creators endorse Pavlov. This document is instructional, not promotional in nature. This note addresses an item in some versions of the BSD and Apache licenses.

3.2.2.1. jCharts

jCharts Project

jCharts is an Apache-Licensed API for plotting line-graphs, scatter-plots, pie charts, and so forth. It's used by several Pavlov pluglets. It's lightweight, quick, and the output is very pretty. jCharts replaced Pavlov's "EasyGraph" module around January 2004. jCharts is a trademark of Nathaniel G. Auvil.

3.2.2.2. JavaHelp/jh.jar

JavaHelp Site at Sun

JavaHelp is software written by Sun Microsystems to provide a familiar help system in Java programs. Click "Help Topics" in Pavlov or Bee to see it in action.

3.2.2.3. sillyview

sillyview site

sillyview is an API that spun off from Pavlov. It uses Velocity (cf below) to provide a model-view-controller framework based on template files.

3.2.2.4. steelme

steelme site

steelme is another API spun off from Pavlov. In short, it provides the Themes submenu of the Prefrences menu. In shorter, it lets Pavlov use pretty colors and fonts (or ugly colors and fonts).

3.2.2.5. Apache Jakarta's Velocity

Velocity Site

Velocity is a very, very pretty piece of software created by the folks at the Apache Jakarta project. It's hard to put velocity in a nutshell, but here's my try: you start with a text "template file." It could be XML, HTML, RTF, PDF, or any sort of text file -- it doesn't matter.

Now, say there are some "tokens" or string values in the template file you want to replace with a set of values determined in. Velocity lets you do this in a very cool way. You can pass any sort of Java object "into" the template, evaluate methods, do loops and evaluate conditionals.

Pavlov uses HTML templates in many places. The main quiz screen is specified by a HTML template. So is the library view on the left side of the main screen. So is the login widget, the "Export Quiz" widget, and so forth. Velocity Pluglets are simply HTML templates that are given information about the user's current state every time he answers a question.

Templates are converted to Swing entities by the sillyview package. sillyview can also render these templates to other environments, such as servlets.

3.2.2.6. Apache Xerces2 Java (xercesImpl.jar/xml-apis.jar)

Xerces2 Java Site

Pavlov uses XML heavily. Xerces is a free API for parsing (and doing other stuff to) XML files. Hence, Pavlov uses Xerces heavily. QED.

3.2.3. What the Heck?!?

You just started looking through the code, and it looks more like C++ than Java? See a "cat" or an "rb" and wonder what it is?

3.2.3.1. What's a "cat" and why is it everywhere?

cat is the standard name for a log4j Category or Logger, an object that can create error, warning, info, debug messages and so forth.

3.2.3.2. What are asserts all about?

JDK 1.4 and later provides assertions. These are checks, usually of preconditions and postconditions in methods, that can be enabled or disabled at runtime. People spend a lot of time arguing about best-practices with assertions. It's generally agreed on that they shouldn't be used to check that method parameters are valid, but you may find places in Pavlov where they're used this way. If you do stumble upon an assert used to check parameters, tag it with a FIXME.

3.2.3.3. What's an "rb" and why is it everywhere?

ResourceBroker is a wrapper with some convenience methods for the java.util.ResourceBundle class. It allows Strings to be stored in properties files instead of being hardcoded into the Java program. This makes it easy to a) tweak strings, and b) internationalize the program.

3.2.3.4. What's this HashMap<String, Object> stuff?

As of JDK1.5, Java allows strong typing in its Collections framework. This is great if you have concerns that somebody might put a CurryChicken object in your vector of IceCream objects. I think we all know what havoc can ensue when you're expecting IceCream and get CurryChicken instead.

3.2.3.5. What's this for(String x : vec) stuff?

Another JDK1.5 feature. Say you have a Vector<String> vec, and you want to loop through it. You could make an Iterator, an Enumeration, loop through it with elementAt(), and so forth. (Furthermore, in 1.4 you'd have to cast your loop variable to String). This construct does all that for you, decreasing the amount of code and the possibility of little mistakes.

3.2.3.6. What are FIXME's for?

A FIXME comment is used to point out that something is implemented poorly, bug-prone, inefficent, or so forth. Developers can search for the string FIXME when looking for something to fix. If you see suspicious code, feel free to plop a FIXME with a brief description of the problem.

You'll also see FIXED comments, which should replace the FIXME when the problem is

corrected. FIXED comments should last "for a while" but should be removed sooner or later. Later is probably better than sooner.

3.2.4. Miscellany

3.2.4.1. The Coding Standards seem confusing/draconian/etc...

The Coding Standards document is a mix of simple and advanced topics, which is probably not a good idea. You don't have to understand the Model-View-Controller pattern to tweak some code. CS should not make you want to jump out of a window or be nervous about contributing. It should provide guidance, especially when you're getting close to committing some changes.

If you are nervous about breaking the coding standards, you may take comfort in the fact that the chunk of code that was lambasted at length to illustrate the standards was written by the project manager. Nobody's perfect.

If there's something in the CS that's patently confusing, ask about it on the pavlov-devel mailing list.

3.2.4.2. I Can't Get CVS to Check Out Pavlov

There are a buzillion CVS clients out there, and I use precisely one (cvs/openssh for cygwin/linux). I think, especially if you use jEdit, the GruntsPud CVS Client is the second easiest one in the world to set up. The key is to get the CVSROOT and CVS_RSH variables set up right.

If, while you're setting up, you see an option named "pserver," it shouldn't be there. In short, pserver is bad, ext is good. Everybody join hands and repeat: "pserver is bad, ext is good." You have to have some sort of secure socket or secure shell support to use CVS at sourceforge if you plan on checking something in. Many clients include some ssh support, you just have to figure out how to make it work.

That said, here's what a CVS session looks like in CygWin and Linux:

```
$CVSROOT = :ext:YOURSFNAME@cvs.sourceforge.net:/cvsroot/pavlov
export CVSROOT
$CVS_RSH = /usr/bin/ssh
export CVS_RSH

then, you'd check out the repository once:

mkdir ~/foo
```

```
cd ~/foo
cvs co .  ( "period" for everything -- cvs co net for just core source)

then edit a file (say, put a space in a comment or something)
cd ~/foo
vi net/sourceforge/pavlov/event/AnswerEvent.java

then checkin
cd ~/foo
cvs ci

then (say the next day) apply diffs from the server to your copy of the code
cd ~/foo
cvs update

or, you could just update the event directory:
cd ~/foo/net/sourceforge/pavlov/event
cvs update
```

3.2.4.3. What's tiger?

Tiger is slang for JDK1.5X/JRE1.5X.

3.2.4.4. Where do I find stuff to do?

Here's some pointers:

- Coding Standards
- Contributing I
- Contributing II
- Current Feature Requests
- Mailing List Archives
- Join the Pavlov Mailing List

I'm philosophically against "assigning projects," it stifles innovation and lowers morale. But, sooner or later you will get a message like "hey, can you take a look at X?" that's just a sign that the other developer thinks you have good eyes. If you're still having trouble finding something to work on, send an email to the list and ask if anybody needs help with something.

3.2.4.5. Can a GPL Project use Apache-Licensed Modules?

According to Apache, yes. There are rumblings that GNU doesn't agree, but I've yet to see anything concrete. There's never been any beef with using BSD modules in a GPL project, so I can't really see what the issue would be. I'm a rabid GPL zealot, but I really can't see what the issue would be.

3.2.4.6. Shouldn't it be called Skinner?

Maybe skinlov or something. Pavlov makes sense from a meta-viewpoint: conditioning a positive response to studying is Pavlovian. The operant conditioning gets the user to learn questions, the Pavlovian aspect conditions the user to respond in a positive manner to the stimulus of "study time."

3.2.4.7. Is it "BEE" or "Bee"

Yes, it is "BEE" or "Bee." BEE is, more or less, an acronym for "Book Editing Environment," so, it should be capitalized, but is often not in practice.

3.2.5. Philosophy

3.2.5.1. What's the goal of the Pavlov Project?

To provide a comfortable, customizable environment for efficient study.

The development of the emacs text editor has deeply affected my goals with Pavlov. My dog could write a text editor -- if there aren't 10,000 text editors out there, I'll eat my hat. What makes emacs special is that it is, and has been, so customizable. As an end user, without touching the core source code, without writing a line of C code, you can dramatically affect how emacs looks and how it works. The result is that the user community adapts emacs to solve the problems they have with it. They share their adaptations, and emacs en-grande develops organically.

This isn't meant to detract in any way from the amazing way that the emacs core has grown with it's staggering network of coders. I'm just saying that the magic bit is the customizability.

Here are some ways that we've strived to make Pavlov customizable:

- themes (color/font schemes) with a easy-to-use theme editor
- skins HTML/VTL documents that specify the user interface
- book XML files: standardized question content
- feedback pluglets: little java programs that respond to the users progress
- velocity feedback pluglets: feedback pluglets without Java
- strategy pluglets: little java programs that choose which questions to ask
- quiz export: VTL documents to export quizzes to any sort of text file
- tool pluglets: pluglets for other sorts of operations

A lot of effort is going into releasing a Java servlet version of Pavlov. This will decouple

using the program from installing it, and should increase the user base by orders of magnitude.

3.2.5.2. Why write another "flashcard" program?

There wasn't one that did what I wanted. I'd say 25% of the ones out there have their questions hardcoded. Another 25% can't handle media, like pictures and sounds, embedded in their questions. I'm not aware of another one that supports pluggable feedback mechanisms, pluggable question selection strategies, skins, and so forth.

3.2.5.3. Get to Know Everybody

I know I'm repeating myself here, but it bears repeating. A lot of the joy of collaborative development comes from talking to other developers, discussing ideas, sharing victories and defeats, and socializing between builds. Coaching a new developer can be particularly rewarding. 'nuff said.

3.2.5.4. Being Nice

As computer scientists, engineers, mathematicians, and so forth, we tend to be a bit abrupt when someone asks a question. This isn't conducive to creating a vital community. There is no place in the Pavlov community for flaming users or other developers. Don't do it.

3.2.6. Directory Structure

Pavlov has many, many directories. A current developer directory tree, discounting build directories, CVS directories, and Javadocs has 106 directories. This document is almost guaranteed to not keep up with changes in the directory structure, but should provide a good basic idea of what is where.

- PAVLOV: startup scripts, Licenses, Readme, ant's build.xml
  - BUILD: where ant creates a build image
  - DIAGRAMS: question images
  - FOR: website content in Forrest format
  - LIB: holds jars for bundled API's
  - LIBRARY: holds Book XML files -- where the questions live
  - PLUGLETS: where feedback, strategy, tool, and velocity pluglets live.
    - FEEDBACK:feedback pluglet classes/jars
    - STRATEGY: question selection strategy classes/jars
    - TOOLS: tool pluglet classes/jars
    - PLUGSRC: pluglet source code
    - BIN: images, sounds, etc for pluglets before deployment in Jar files

    - VELOCITY: velocity pluglet template files
- RESOURCES: application-level image files, Icons, About, etc.
    - AUDIO: files for random audio pluglet
    - BEEHS: bee JavaHelp files
    - HCTEMPLATES: velocity templates for quiz export
    - HS: pavlov JavaHelp files
    - ICONS: icon images for Bee and Pavlov.
    - PROPERTIES: original ResourceBundles. Copied by ant into pavlov.jar.
    - RANDOM: images for RandomImage pluglet
    - SEQUENCE: images for Sequenced Image (aka Filmstrip) pluglet
    - SKINS: velocity templates and resources for skins
    - THEMES: installed steelme themes
    - VIEWS:Deprecated, functionality will be replaced by skins/default
- TEST: holds JUnit test suite
- USERS: holds per-user history files.
- VAI: holds config files and scripts for VAInstaller
- NET: top of source tree. See JavaDocs for source tree descriptions.

3.3. Writing A Velocity Pluglet

3.3.1. About This Document

One of the big improvements in Pavlov 1.1 is the leveraging of the Apache Jakarta Project's Velocity Template Engine to create templated Pluglets and Skins. This document describes how to write a Velocity Feedback Pluglet for Pavlov.

3.3.2. Step 1: Find The VelocityPluglet Directory

Let's assume you already have Pavlov 1.1B1 or later installed. Find the directory it was installed in. Now, there will be a subdirectory called `pluglets` and its subdirectory of `pluglets/velocity`. In the 1.1BX release, there are two subdirectories here called `aliengrades` and `navylife`. Every subdirectory of `pluglets/velocity` is scanned at startup for a file named `index.vm`. Those with a valid `index.vm` are accepted and added to the `Feedback Menu` in Pavlov. So, for example, on my machine, I have

```
/home/haus/pup/pluglets/velocity/aliengrades/index.vm
/home/haus/pup/pluglets/velocity/navylife/index.vm
```

let's take a look at one of these.

3.3.3. Step 2: Look at an Example

We open up the file `pluglets/velocity/aliengrades/index.vm` and see something like this:

```
01   <HTML>
02   <HEAD></HEAD>
03   <BODY BGCOLOR="black">
04
05   <TABLE HEIGHT="223" WIDTH="304">
06   <TR>
07   <TD HEIGHT="223" WIDTH="304">
08   #set ( $right = $ANSWER_EVENT.getNumberOfCorrectAnswers() )
09   #set ( $tot= $ANSWER_EVENT.getNumberOfAnswers() )
10   #set ( $perc  = 100 * $right/$tot )
11
12   #if ( !$perc )
13     #set ($img ="logo.jpg")
14   #elseif ( $perc < 59 )
15     #set ($img ="f.jpg")
...stuff deleted...
16   #else
17      #set ($img ="aplus.jpg")
18   #end
19   <IMG SRC ="$BASE_URL$img">
20   </TD>
21   </TR>
22   </TABLE>
23
24   </BODY>
25   </HTML>
```

Let's make a few observations:

- This is a well formed HTML document (Java doesn't like bad HTML)
- It has some funny stuff with pound "#" signs (this is Velocity for "do something")
- It has some funny stuff with dollar "$" signs (this is Velocity for "value of something")
- `$ANSWER_EVENT` shows up a lot (Pavlov sends an `ANSWER_EVENT` to the `VelocityModel` every time the user answers a question. `ANSWER_EVENT` has a lot of fun information in it, see http://pavlov.sourceforge.net/api-docs/net/sourceforge/pavlov/event/AnswerEvent.html for all the goodies.
- On lines 8 and 9 we get the number of correct answers and number of total answers in the current quiz
- In line 10, we get `perc`, which is a percentage from 0 to 100 representing the score in the current quiz
- In line 12, we see if `$perc` exists. If it does not exist there have been no answers in this quiz, and we set `$img` to a default value

- In lines 14-18 we choose an image to display based on the value of $perc
- In line 19 we display the image.
- Note the `$BASE_URL` variable in line 19. This lets you use resources in the same directory as `index.vm`.
- The rest of the code just closes the table and document.
- Since we know about how big this HTML page is going to be, we set the table height and width. This makes life a little easier on Java. If we do not know the size, the Java components can figure it out on their own.

> **Note:**
> Java doesn't like some META tags, and it hates bad HTML. If you have problems debugging a VelocityPluglet, try to remove META tags and simplify the HTML.

3.3.4. Step 3: Use Advanced Velocity Features

The best source for information on advanced Velocity features is at Apache Jakarta Project's Velocity Site. There are some neat tricks on display there.

3.3.5. Step 4: Get More Information From Pavlov

Pavlov passes a great deal of information to your template using the AnswerEvent class. You can get information from the `AnswerEvent` by using it's methods. A few examples:

$ANSWER_EVENT.getQuestionData().getPercentageCorrect()	Percentage of correct answers over whole user history
$ANSWER_EVENT.getUser().getName()	User's login name.
$ANSWER_EVENT.getBookData().getTitle()	Title of book in use
$ANSWER_EVENT.getChapterData().getStrategyName()	Name of the question selection strategy in use
$ANSWER_EVENT.getChapterData().getQuizCollectionData().getNumberOfQuizzes()	Number of quizzes this user has taken on this chapter
$ANSWER_EVENT.getChapterData().getQuizCollectionData().getTotalAsked()	Total number of questions this user has answered over all quizzes on this chapter

The point here is not to list all the possible types of information you can glean from the system, but to illustrate that there is a lot to choose from.

3.3.6. Step 5: Publish

When you get a feedback pluglet that you like, send it to the Pavlov project. We'll need copyright information and have to filter out obscenities, but other than that, we'll be happy to

make your pluglet available. That way, others can not only use your pluglet, but learn from it as well!

3.4. Writing A Custom Skin for Pavlov

3.4.1. Writing a Skin for Pavlov

One of the big improvements in Pavlov 1.1 is the leveraging of the Apache Jakarta Project's Velocity Template Engine to create templated Pluglets and Skins. This document describes how to write a custom Skin for Pavlov.

3.4.2. Step 1: Find The Skins Directory

Let's assume you already have Pavlov 1.1B1 or later installed. Find the directory it was installed in. Now, there will be a subdirectory called `resources` and its subdirectory of `resources/skins`. In the 1.1B1 release, there are several subdirectories of skins, like "cipher" and "euphoria." Every subdirectory of `skins` is scanned at startup for subdirectories that have velocity template files named `Library.vm`, `Logon.vm`, `QuizPanel.vm`, and `Welcome.vm`. Pavlov also looks for a file called `Theme.theme` which specifies the fonts and colors to use for the skin.

> **Note:**
> Don't spend any time working on the Logon and Welcome templates. They need to be there, but aren't really used. They won't need to be there in a near-future release.

Let's take a look at one of these examples.

3.4.3. Step 2: Look at an Example

We open up the file QuizPanel.vm in the cipher directory and see something like this:

```
01 <html>
02 <head>
03  <title>pavlov quiz: $BOOK_NAME: $CHAPTER_NAME</title>
04 </head>
05 <body text="#ffffff" bgcolor="#000000" link="#ffffff" vlink="#ffffff"
   alink="#ffffff" background="${BASE_URL}cipher.JPG">
06 pavlov quiz: $BOOK_NAME: $CHAPTER_NAME
07
08 <HR>
09  <table width="100%" height="400" border="1">
10    <tbody>
11      <tr width="100%" height="200">
12
```

```
13 #if( $QUESTION_IMAGE!="")
14       <td valign="Top" height="200" colspan="1">
15        <div class="P"><big>$QUESTION_TEXT</big></div>
16        </td>
17        <td><img src="$QUESTION_IMAGE" align="Left">
18        </td>
19
20 #else
21        <td colspan="2" height="200" >
22        <div class="P"><big>$QUESTION_TEXT</big><</div>
23        </td>
24
25 #end
26        </tr>
27      <tr>
28        <td colspan="2">
29        <table width="100%" border="0" height="200">
30          <tbody>
31            <tr>
32              <td height="100%"><!-- start a -->
33              <div class="P"><big><a href="${BASE_URL}$A_BUTTON"><b>
34       A.</b></a>
35  $A_ANSWER </big></div>
36              <hr>
... stuff deleted...
37             </td>
38            </tr>
39          </tbody>
40        </table>
41      </tbody>
42   </table>
43</body>
44</html>
```

Let's make a few observations:

- This is a well formed HTML document (Java doesn't like bad HTML)
- It has some funny stuff with pound "#" signs (this is Velocity for "do something")
- It has some funny stuff with dollar "$" signs (this is Velocity for "value of something")
- `$BOOK_NAME` and `$CHAPTER_NAME` let you reference the name of the book and chapter the user is quizzing on.
- On lines 13 to 18, we display the question's image along with the question's text if the image exists
- On lines 20 to 25, we display the question's text if no image exists for the question
- On lines 33 to 35, we display answer A as well as providing a link that Pavlov uses to determine if it is the right answer or the wrong answer
- Note the `$BASE_URL` variable in line 5. This lets you use resources in the same directory as `QuizPanel.vm`.
- The rest of the code just closes the table and document.

- Since we know about how big this HTML page is going to be, we set the table height and width. This makes life a little easier on Java. If we do not know the size, the Java components can figure it out on their own.
- Feel free to use CSS Stylesheets, either inlined with the `<STYLE>` tag or loaded with a `<LINK>` tag. Remember to use `$BASE_URL` to help the system find linked files.

> **Note:**
> Java doesn't like some `META` tags (charset specifiers seem to be the primary offenders), and it hates bad HTML. If you have problems debugging a VelocityPluglet, try to remove `META` tags and simplify the HTML.

3.4.4. Step 3: Set up A Default Theme

When you've got your HTML how you want it, you've probably made some decisions on what sort of fonts and colors you want the skin to use. You can ensure that the rest of the Pavlov application uses these fonts and colors by providing a file `Theme.theme` in your skin directory. You can create this file by hand or using the "Theme Editor" available in Pavlov. (If you use the theme editor, you'll have to save your theme, then copy if from `resources/themes/YourTheme.theme` to `resources/skins/yourskin/Theme.theme`.)

> **Note:**
> You can skip the next paragraph if you're planning to use the ThemeEditor to create your themes.

As an introduction to themes, let's take a look at the `Binary.theme` file. The colors are comprised of triplets of numbers from 0-255 describing the amount of red, green, and blue in the color. Attributes like fontSize, name, and fontName are pretty self-explanatory. The best way to understand what parts of the GUI correspond to the various colors is to use the Theme Editor in Pavlov.

```
#Tue May 11 20:41:04 EDT 2004
name=Binary
fontName=Microsoft Sans Serif
text2=153,153,153
text1=255,255,255
black=255,255,255
fontStyle=1
white=0,0,0
primary3=0,0,0
secondary3=31,31,31
primary2=0,0,0
secondary2=204,204,204
primary1=204,204,204
secondary1=0,0,0
fontSize=14
```

3.4.5. Step 4: The Other Template Files

The files `Logon.vm` and `Welcome.vm` shouldn't be part of the skin system, but they have to be there for now. You shouldn't spend any effort customizing these templates. The `Library.vm` file controls how the books and chapters containing questions are displayed to the user. You should customize `Library.vm` to agree with your `QuizPanel.vm` template.

3.4.6. Step 5: Use Advanced Velocity Features

The best source for information on advanced Velocity features is at Apache Jakarta Project's Velocity Site. There are some neat tricks on display there.

3.4.7. Step 6: Get More Information From Pavlov

Pavlov passes a great deal of information to your template. The default skin in `skins/default` displays all the information Pavlov passes. If you want to display more or less information, feel free.

3.4.8. Step 7: Publish

When you get a skin that you like, send it to the Pavlov project. We'll need copyright information and have to filter out obscenities, but other than that, we'll be happy to make your skin available. That way, others can not only use your skin, but learn from it as well!

3.4.9. Known Bugs and Issues

- Selecting a skin before starting a quiz causes some (minor) problems in how the QuizPanel is drawn.
- It's a really good idea to use HTML tables to enclose the content in `Library.vm` and `QuizPanel.vm`, and to set the width and height of these tables. 100% is a good setting for table width. Using exact numbers like width="400" height="400" is a good idea.

3.5.

4. Pavlov Exchange

5. CVS Repository

6. Whole Site

7. Pavlov Tour

7.1. Starting Pavlov

7.1.1. Slide 1: Starting Pavlov

This is what Pavlov looks like when you start it up. On the left, you see the "Quiz Selector" displaying books and chapters of questions available for you to study.

Starting Pavlov

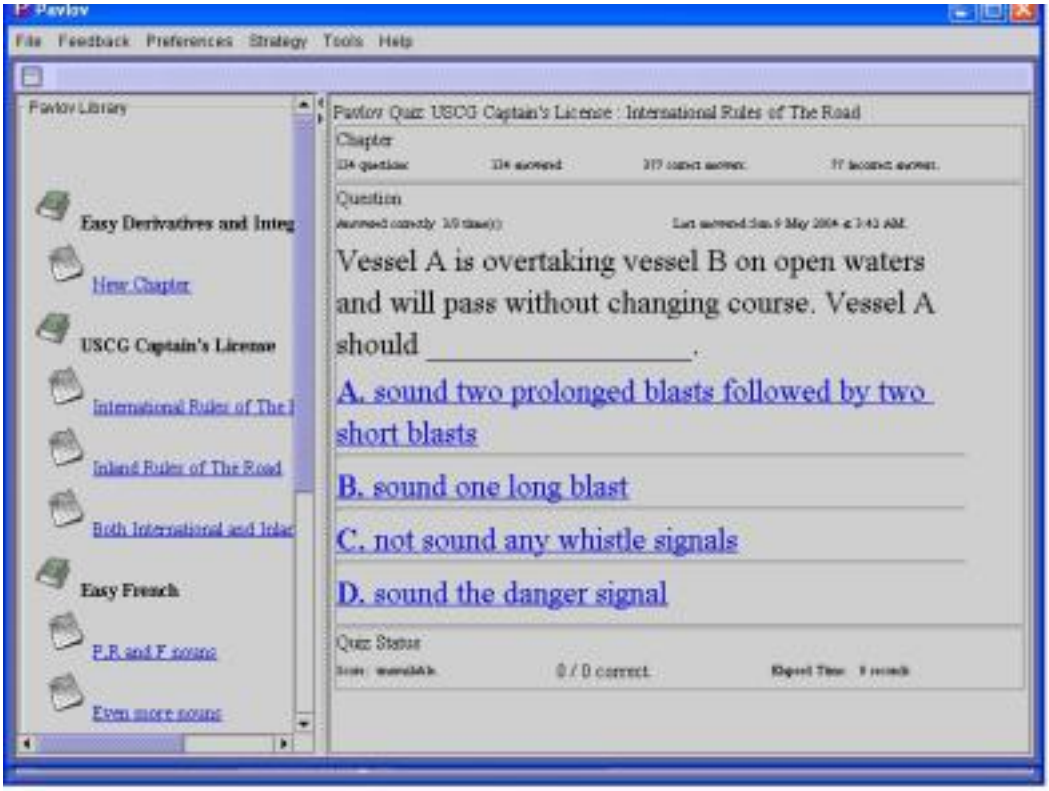
7.2. Choosing A Quiz

7.2.1. Slide 2: Choosing a Quiz

Click on a chapter to start a quiz.

Note:

Apparently, when starting the first quiz, you may have to double-click the link as of Pavlov 1.1B1.

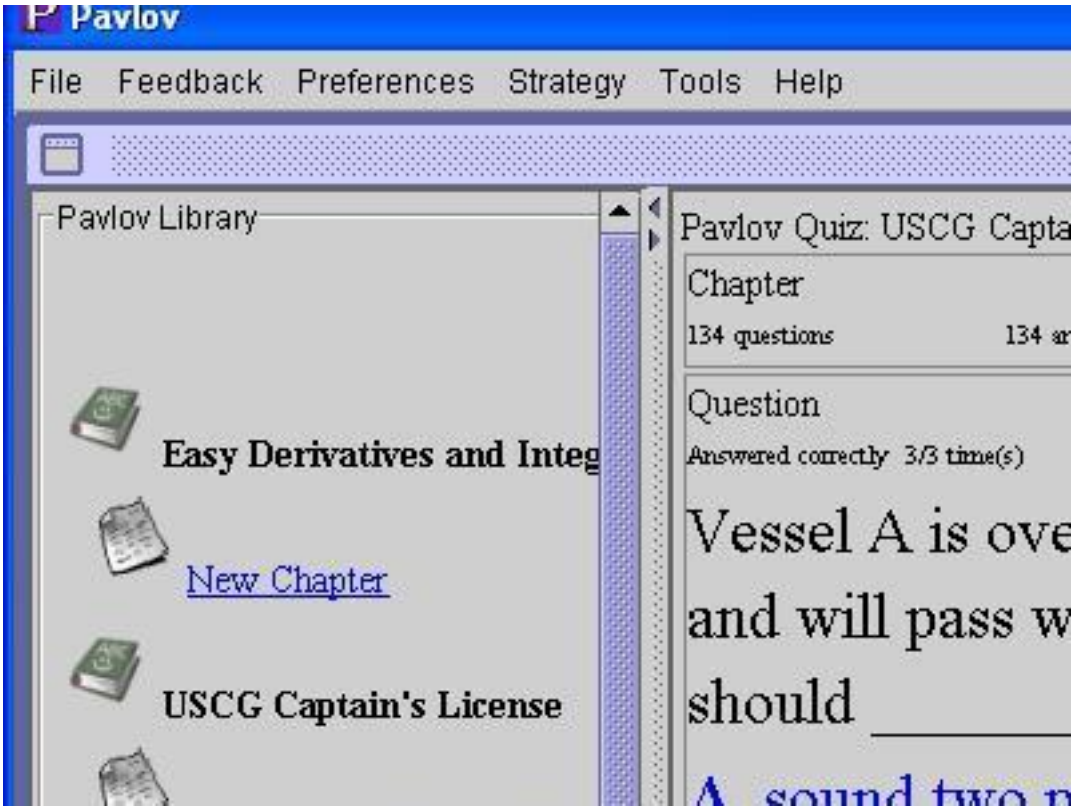


pavlov screenshot

7.3. Starting a Quiz

7.3.1. Slide 3: Starting a Quiz

Starting a quiz pops up the Quiz screen. You are presented with a question and four possible answers. Notice that Pavlov keeps track of how you have done on this question in the past. I've answered this question correctly three of three times.



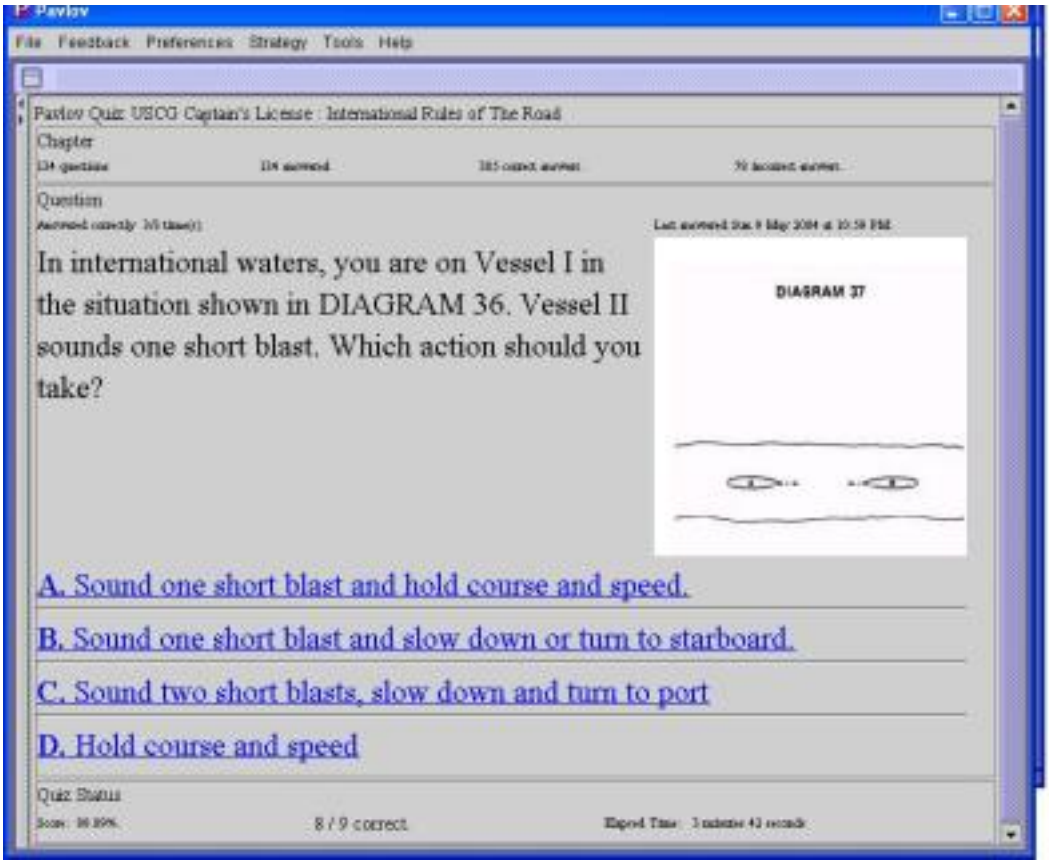
pavlov screenshot

7.4. Continuing A Quiz

7.4.1. Slide 4: Continuing a Quiz

You can minimize the quiz selector using the splitpane control (the arrows between the quiz selector and the quiz panel). Note that Pavlov handles question images nicely.

If you look closely at the bottom of the image, you'll see how this skin keeps track of score and elapsed time. Information about my progress in the chapter is displayed towards the top.



pavlov screenshot

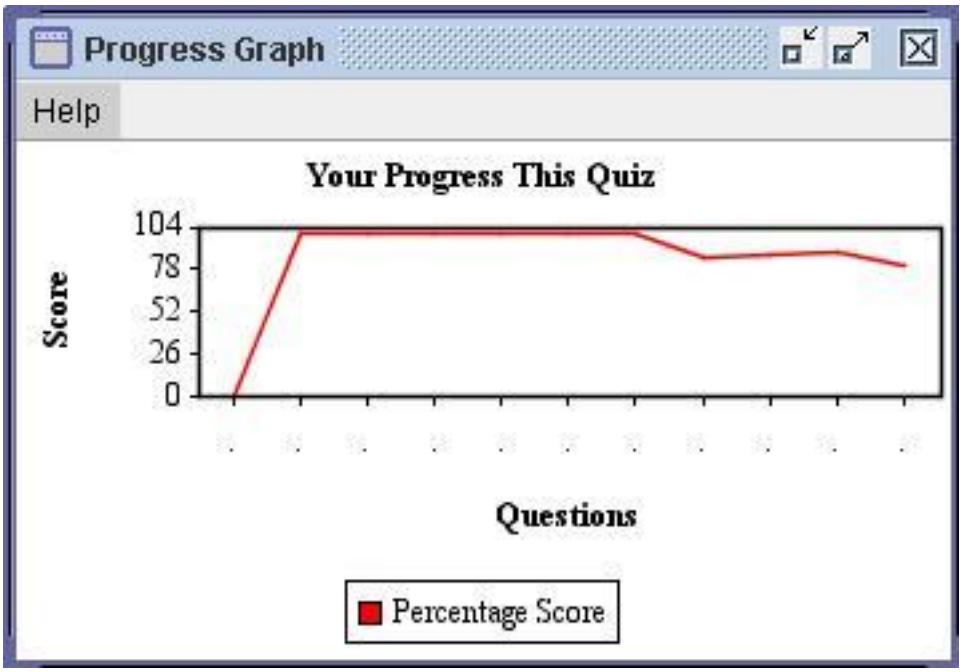
7.5. Monitoring Your Progress

7.5.1. Slide 4A: Monitoring Your Progress

Pavlov offers several ways to monitor your progress as you work on a chapter. Most use information from every question you've ever answered in a quiz. Most are graphical. *Most of the following tools are available from the "Feedback" menu.*

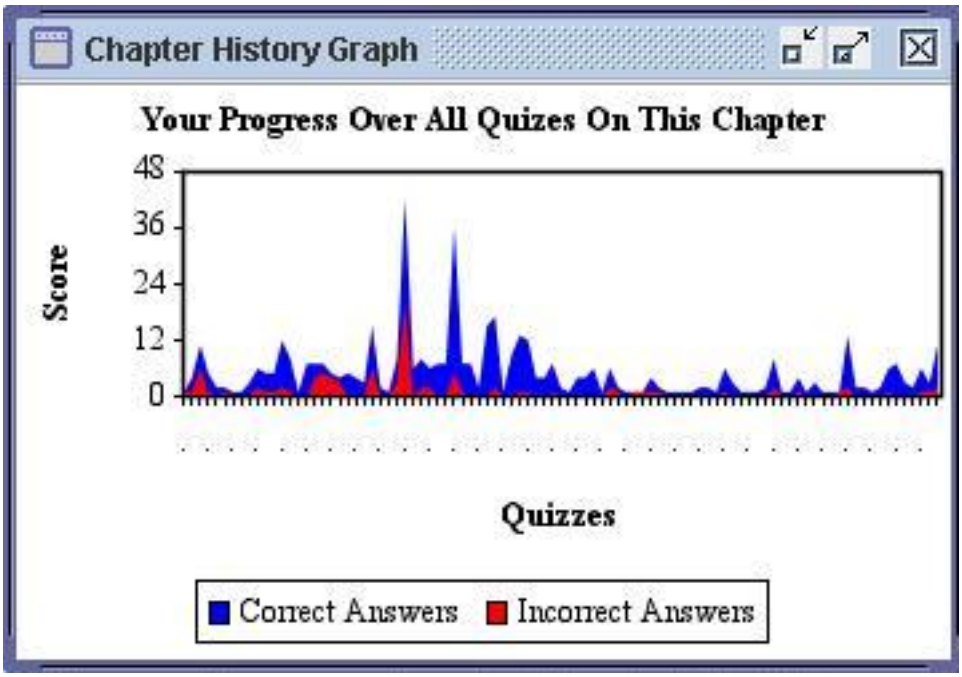
7.5.2. Quiz-Centric Progress

The most basic statistic in a quiz is your score. Watching your score go up and down gives you a feel about how you're doing. The "Progress Graph" shows exactly this:



pavlov screenshot

You can monitor how you have progressed over every quiz you've taken in this chapter by watching the "History Graph:"

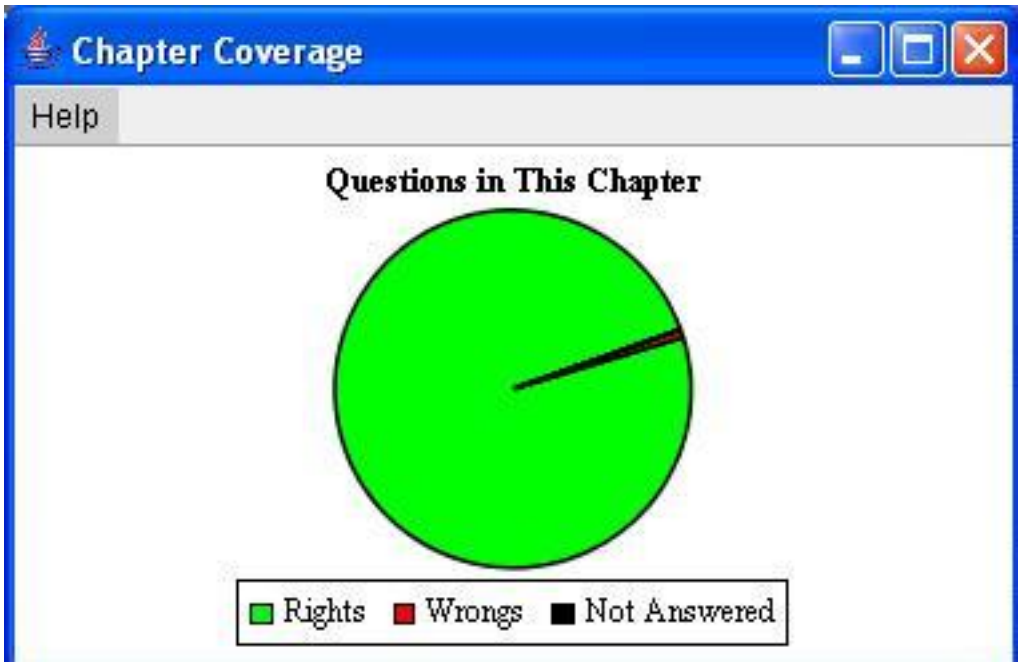


pavlov screenshot

The Red areas represent the number of incorrect answers in that quiz, and blue the correct answers.

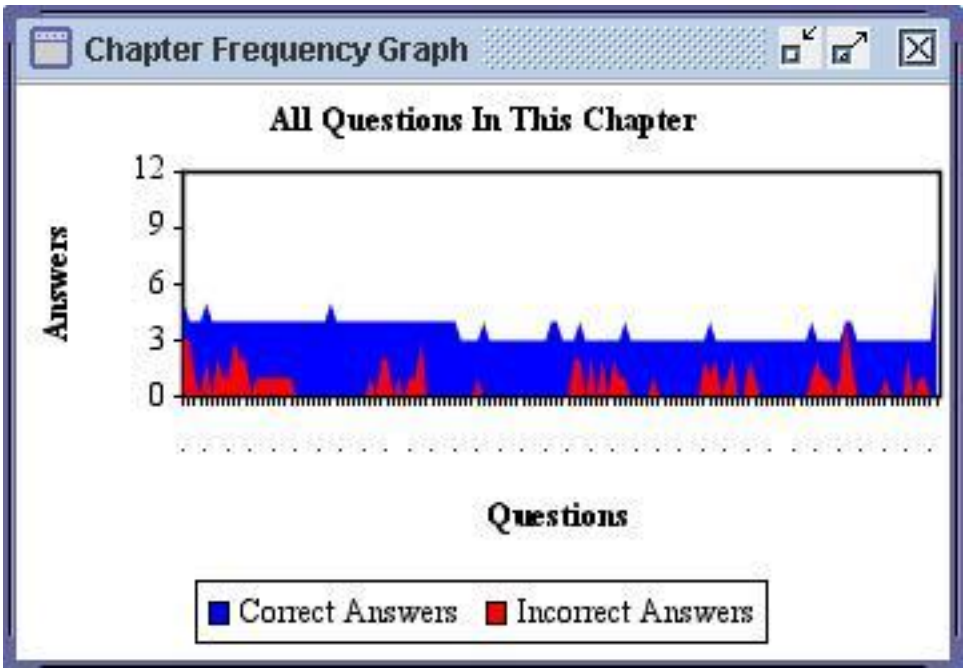
7.5.3. Chapter-Centric Progress

The simplest monitor of how you've fared against a chapter in all your quizzes is the "Coverage Graph." This breaks the chapter into 3 pieces: questions you've answered right at least once(green), questions you've answered at least once with no correct answers(red), and questions you've never answered (black). Here's what one looks like:



pavlov screenshot

The "Frequency Graph" gives you a much more detailed look at how you're attacking the chapter. It lists all the questions across the bottom of the graph. Red areas represent incorrect answers and blue areas represent correct answers.

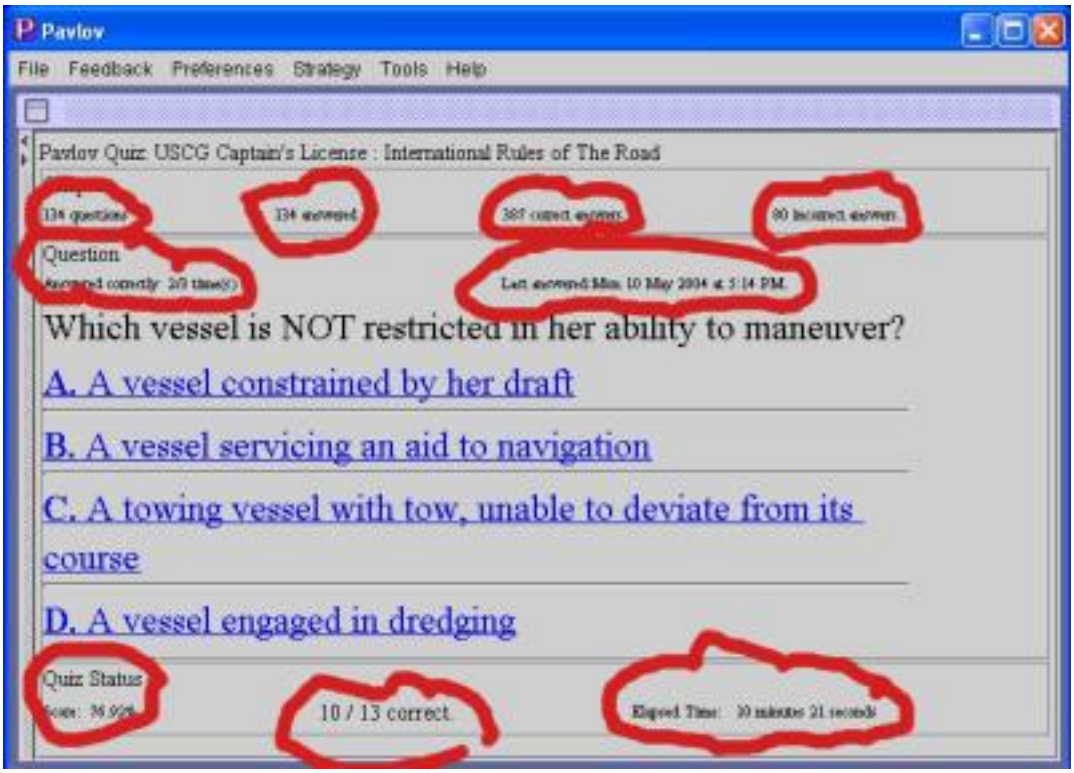


pavlov screenshot

Analyzing this graph can give you a lot of insight into how question selection strategies work, and more importantly, how much you know about the information in this chapter.

7.5.4. Textual Progress Monitoring

Arranged on the quiz panel are various indicators of how you are progressing on the quiz. Which pieces of information are displayed, and where, depends on the designer of the skin you are using. This skin was obviously designed by someone obsessed with numbers.

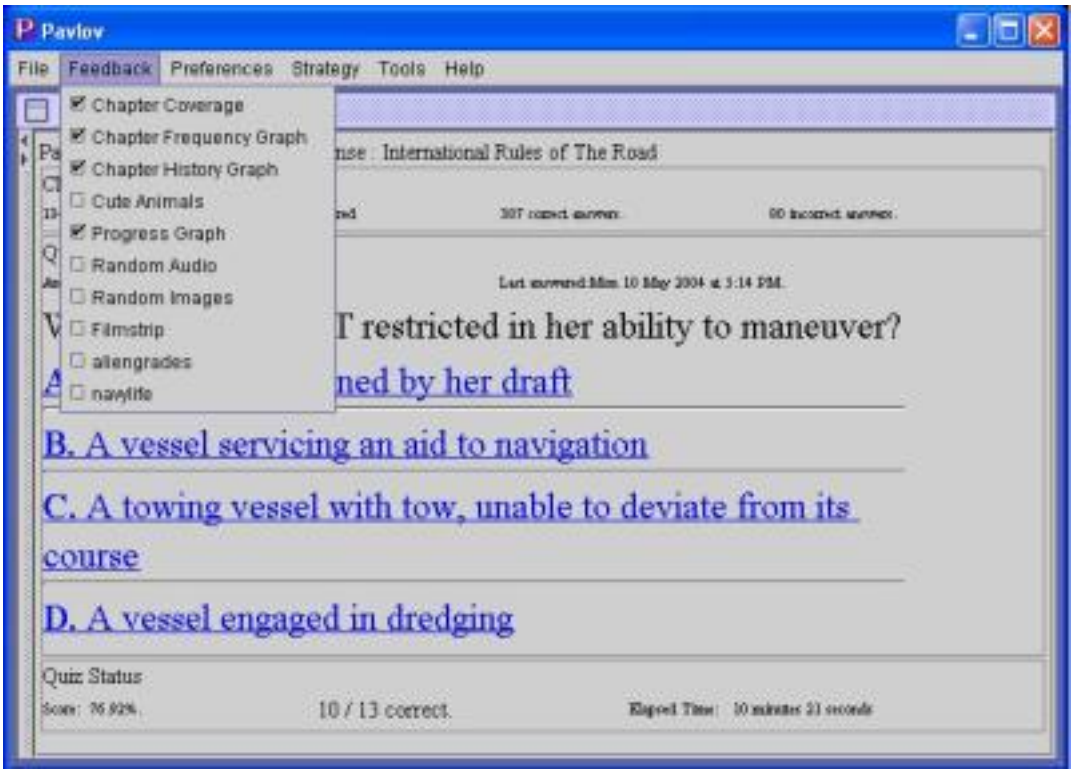


pavlov screenshot

7.6. Using Feedback Pluglets

7.6.1. Slide 5: Using Feedback Pluglets

Now it's time to have fun. Open the "Feedback" menu and look at the pluglets that you have installed. Selecting a pluglet activates it, deselecting it deactivates it.

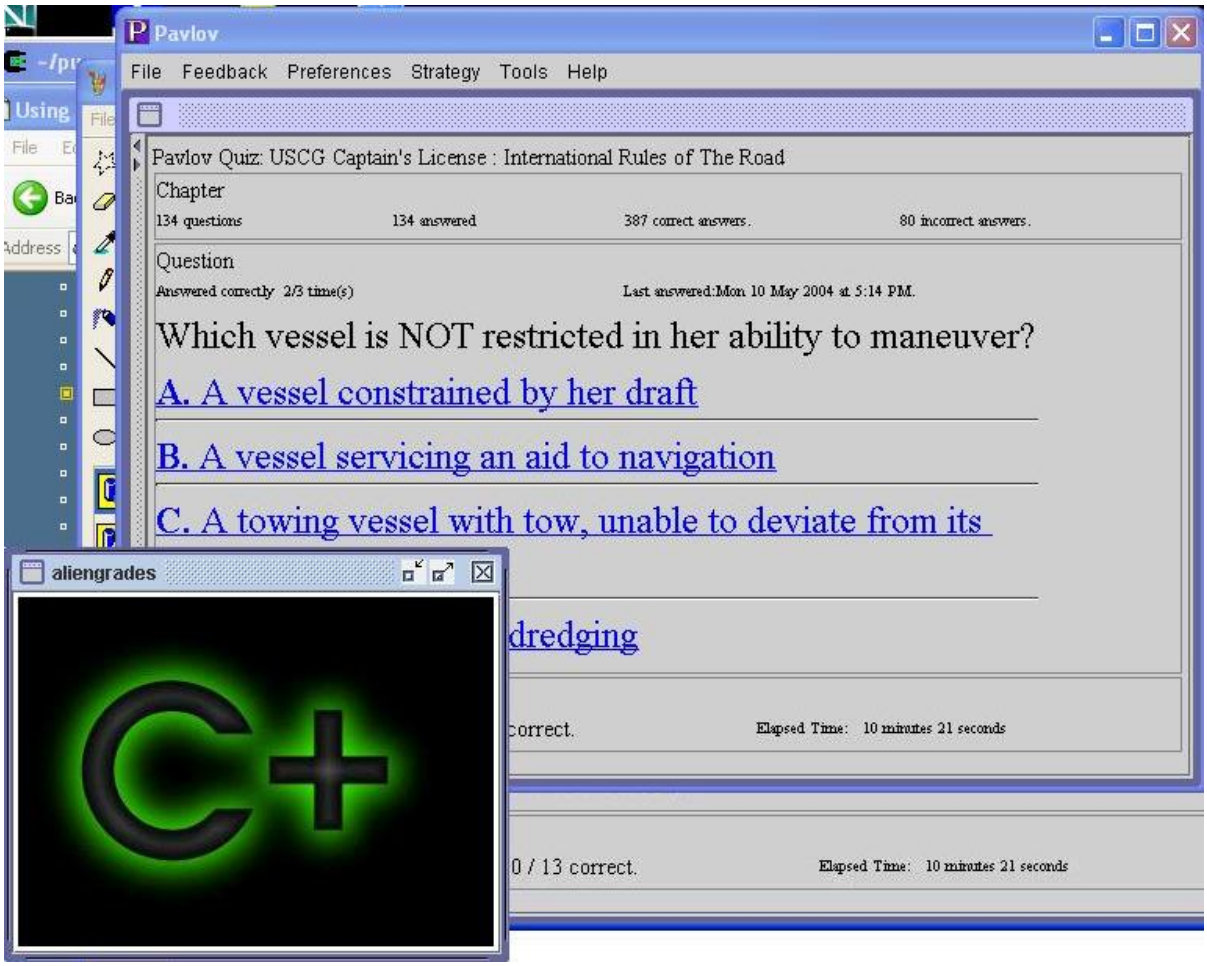


pavlov screenshot

7.7. Using Feedback Pluglets

7.7.1. Slide 6: Using Feedback Pluglets

Let's activate the "Alien Grades" pluglet. This assigns you a grade based on the percentage of your correct answers.

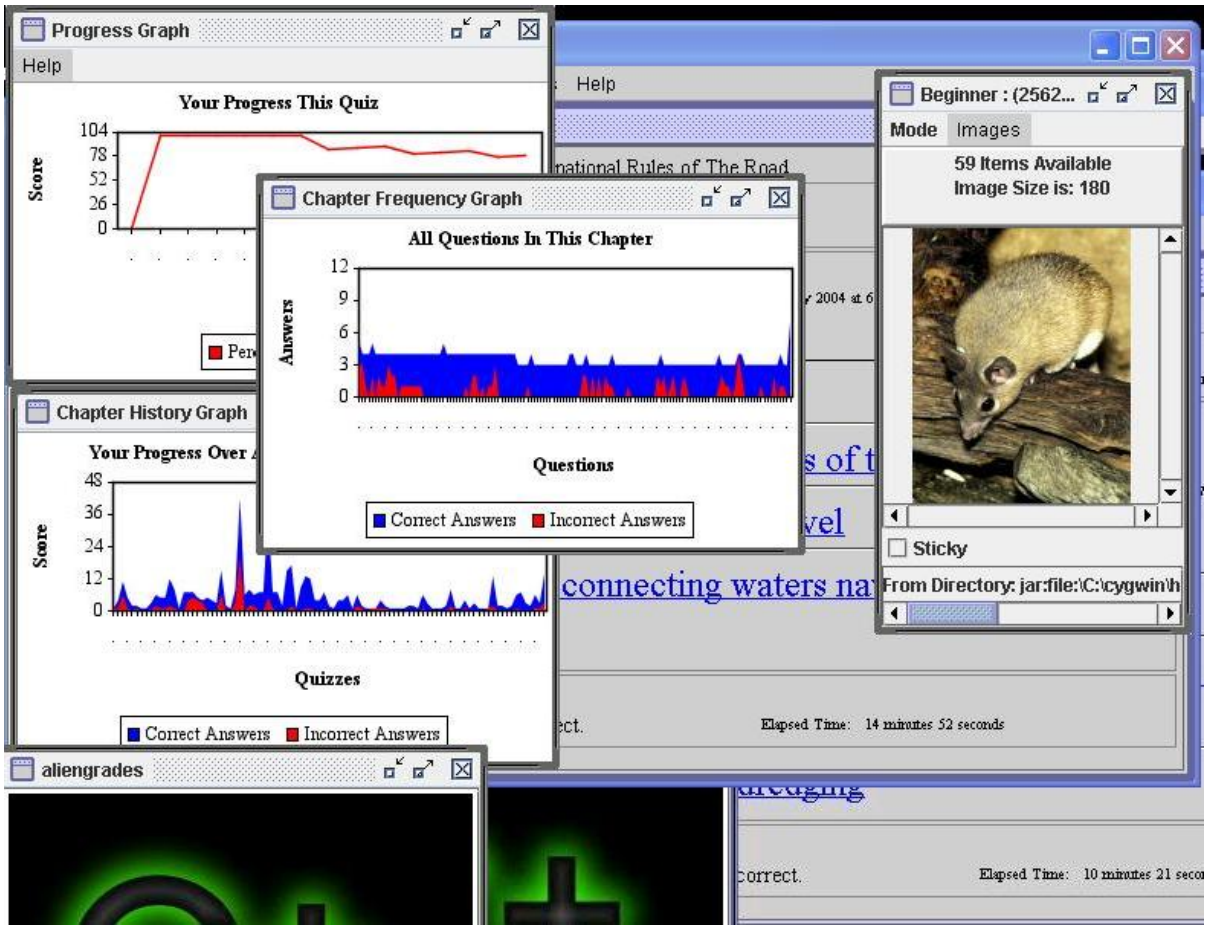


pavlov screenshot

7.8. Using Feedback Pluglets

7.8.1. Slide 7: Using Feedback Pluglets

Maybe "Alien Grades" doesn't excite you. Or maybe you want to use Pavlov to teach a child arithmetic. Looking through the pluglets available, we see that there's one that shows pictures of cute animals (you can plug in images of whatever you want) . If you answer a question correctly, it makes the picture bigger. Incorrect answers make the pictures smaller. That might do the trick...

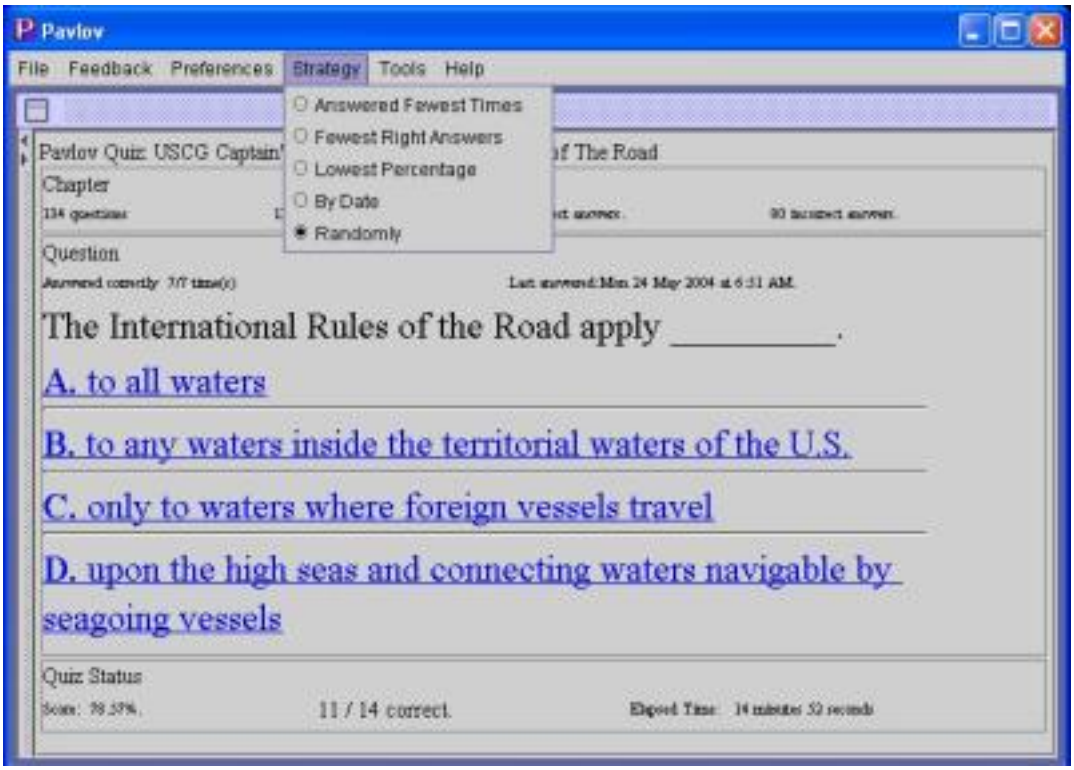


pavlov screenshot

7.9. Using Strategy Pluglets

7.9.1. Slide 8: Using Strategy Pluglets

Pavlov allows you several different ways to pick the questions you study. Choosing "Lowest Percentage" lets you focus on the questions you have the hardest time answering correctly. Choosing "Answered Fewest Times" makes sure you study all the content in the chapter evenly.

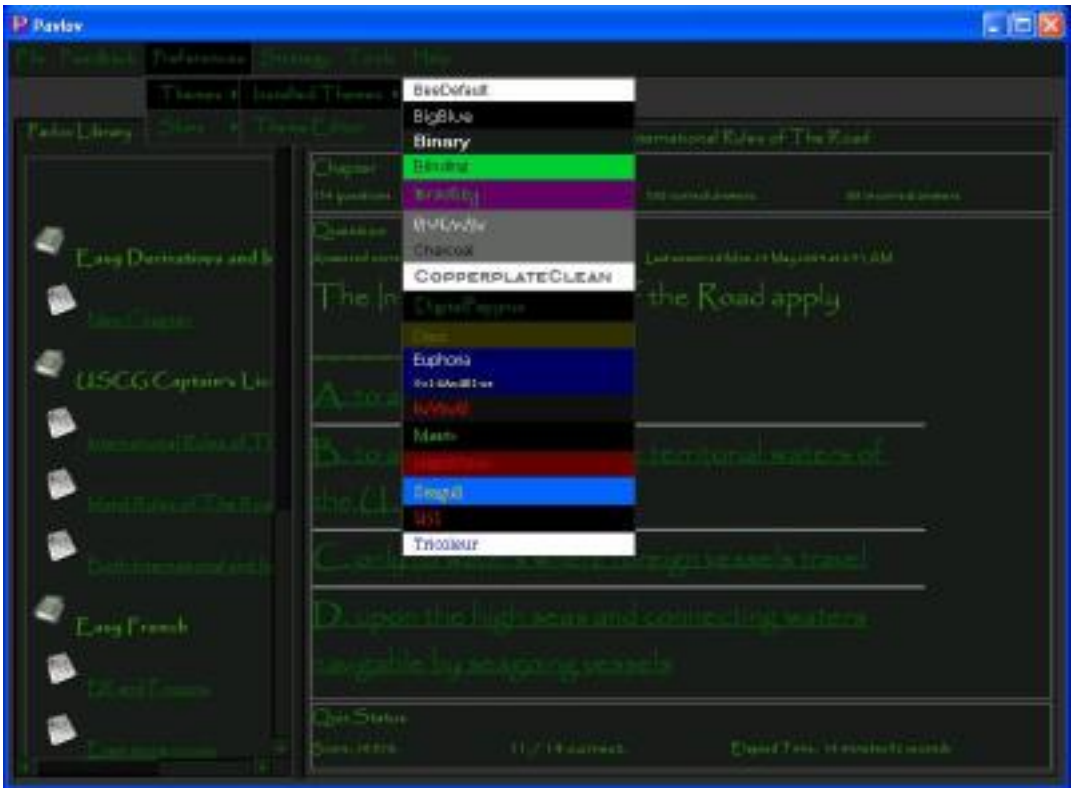


pavlov screenshot

7.10. Using Themes

7.10.1. Slide 9: Using Themes

One of the ways to make a program more comfortable to you is to customize how it looks. The Preferences menu allows you to do this.



pavlov screenshot

7.11. Using Skins

7.11.1. Slide 10: Using Skins

Maybe this sort of visual environment is more soothing to you...

Note:
Artwork (euphoria1024.jpg) by Barbara Lopez http://www.xerraire.com.

pavlov screenshot

7.12. Using Skins

7.12.1. Slide 11: Using Skins

... or maybe you're feeling spacey ...

> **Note:**
> Image property of NASA, used in accordance with their guidelines. See LICENSE_SKINS.txt for more information.

pavlov screenshot

7.13. Using Themes

7.13.1. Slide 12: Using Themes

Or you can create your own custom theme.

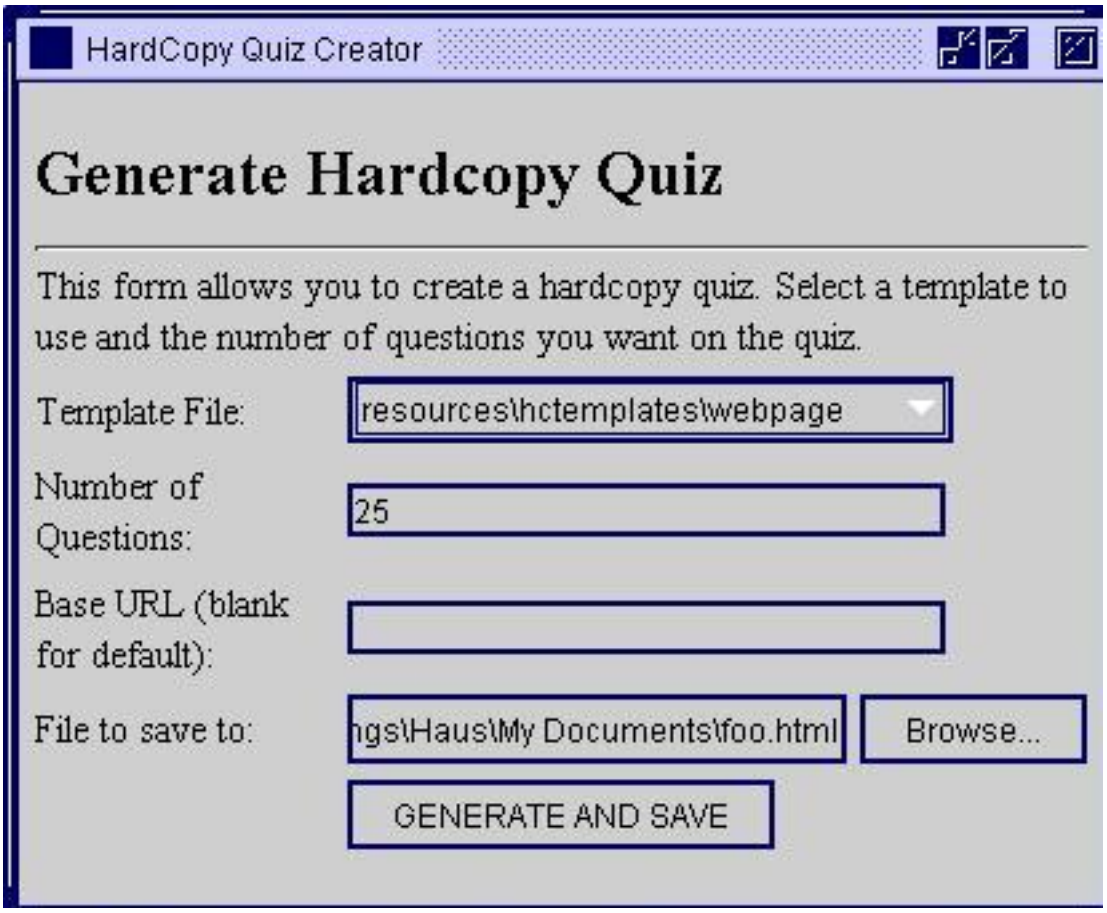
pavlov screenshot

7.14. Exporting a Quiz

7.14.1. Slide 13: Exporting a Quiz

This shows you "Generate Hardcopy Quiz" dialog. Let's generate a simple web-page tutor with 25 questions (using the current chapter and question selection strategy).

Note:
This dialog should be renamed soon.



pavlov screenshot

7.15. Export Quiz

7.15.1. Slide 14: Export Quiz

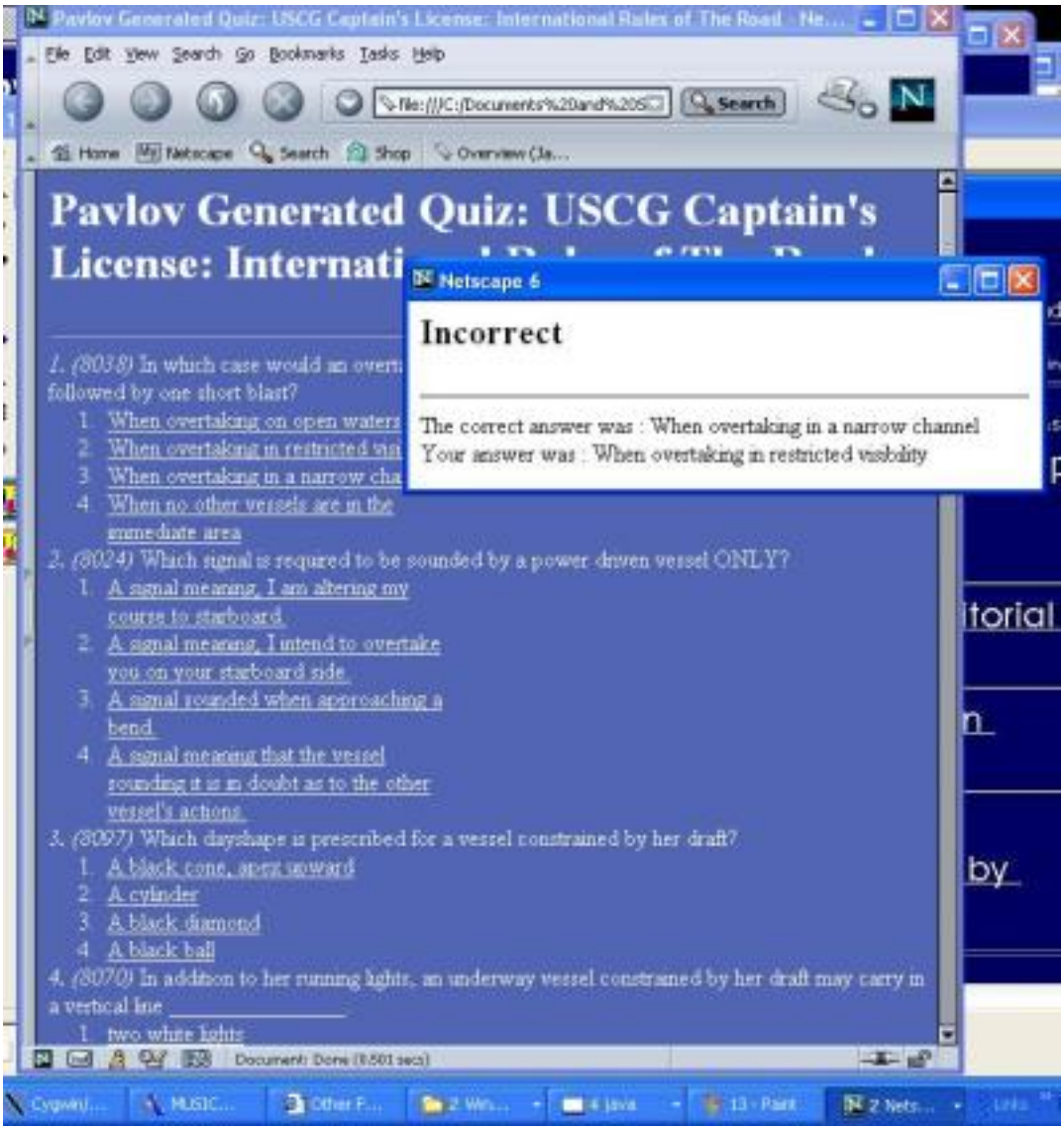
Here's the output from the "Export Quiz" on the previous slide. The user clicks on the answers he selects and is informed by a JavaScript if he was right or not.

Note:

This "app" might break if there are quotes or single-quotes in the answers.

As with almost everything in Pavlov, you can specify the output with a Velocty Template Language template. It would be interesting to generate DocBook documents that can be

converted to TeX, Postscript, PDF, etc...



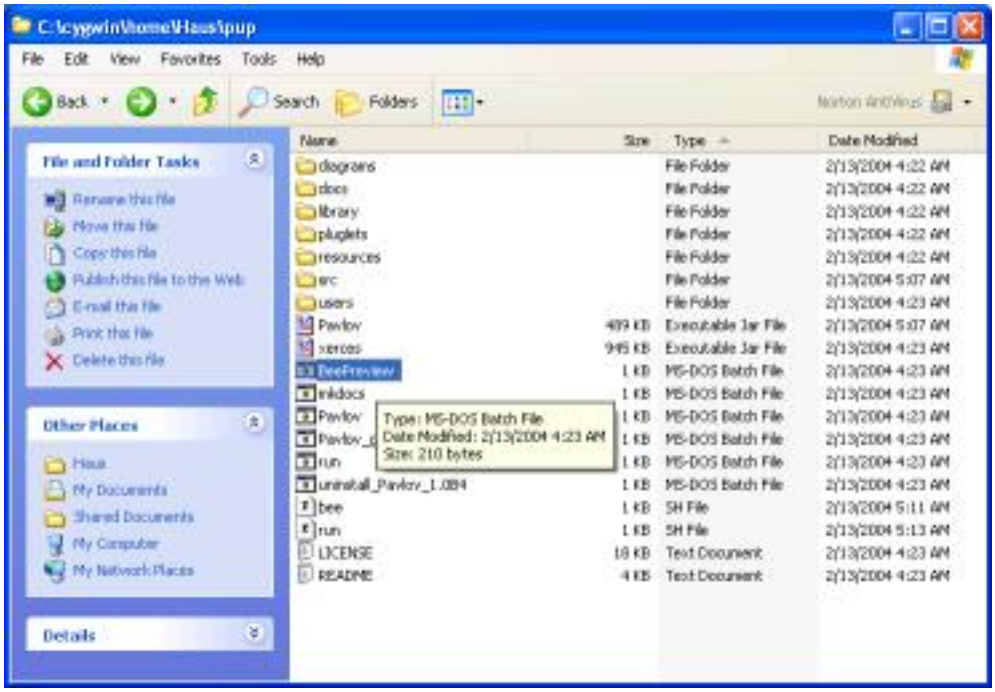
pavlov screenshot

8. BEE Editor Tour

8.1. Starting Bee

8.1.1. Slide 1: Starting Bee

You can start Bee by clicking on the script (batch file, or shell script on Unix) in the Pavlov directory.



bee screenshot

8.2. Bee Startup

8.2.1. Slide 2: Bee Startup

When you start Bee, you have the options to create a new book, edit an existing book, or quit.

bee screenshot

8.3. Create A Book

8.3.1. Slide 3: Create A Book

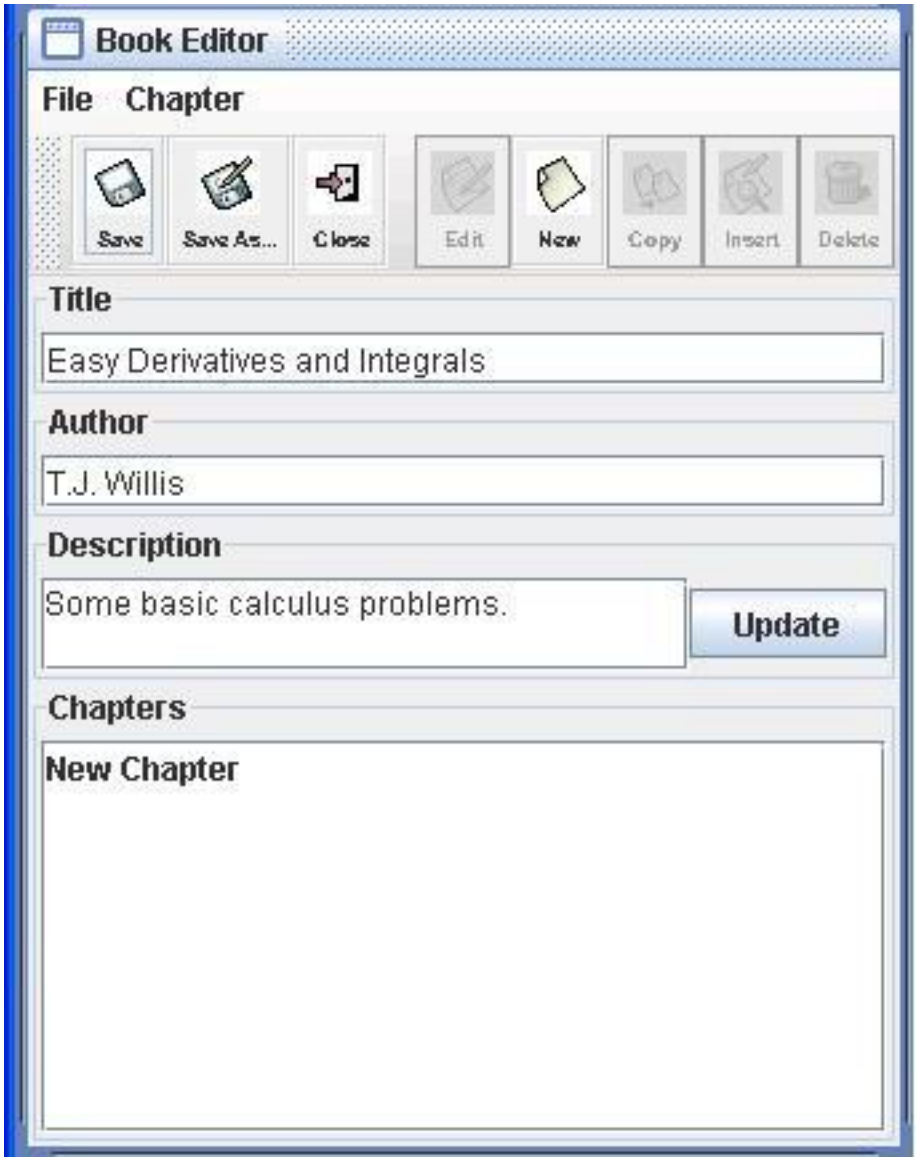
You can use the wizard interface to create a book.

bee screenshot

8.4. The Book Editor

8.4.1. Slide 4: The Book Editor

The Book Editor allows you to edit the author, title, and description of the book, as well as editing the book's chapters.

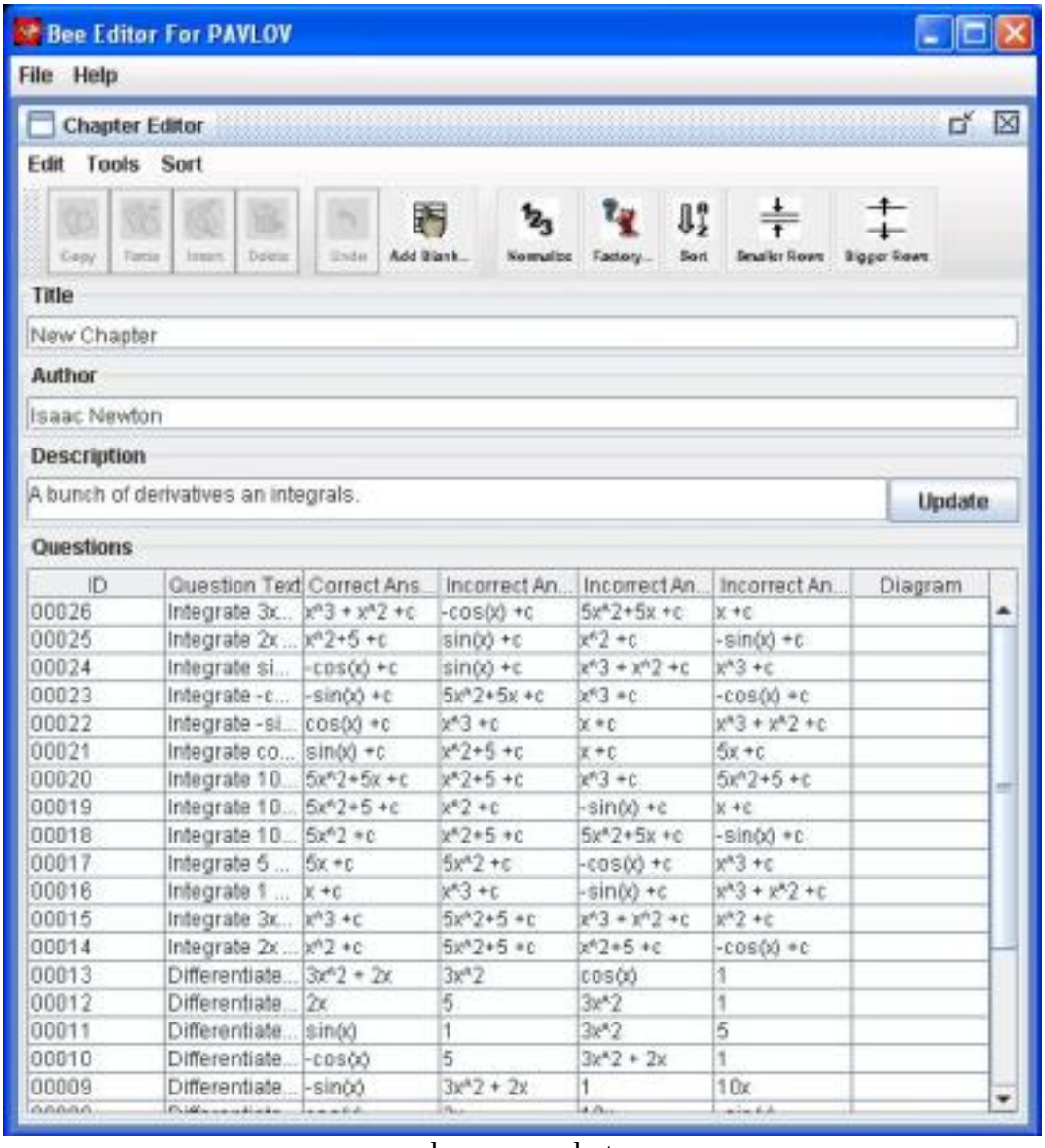


bee screenshot

8.5. The Chapter Editor

8.5.1. Slide 5: The Chapter Editor

This shows the Chapter Editor, which allows you to edit the Chapter's author, title, and description, as well as all its questions.

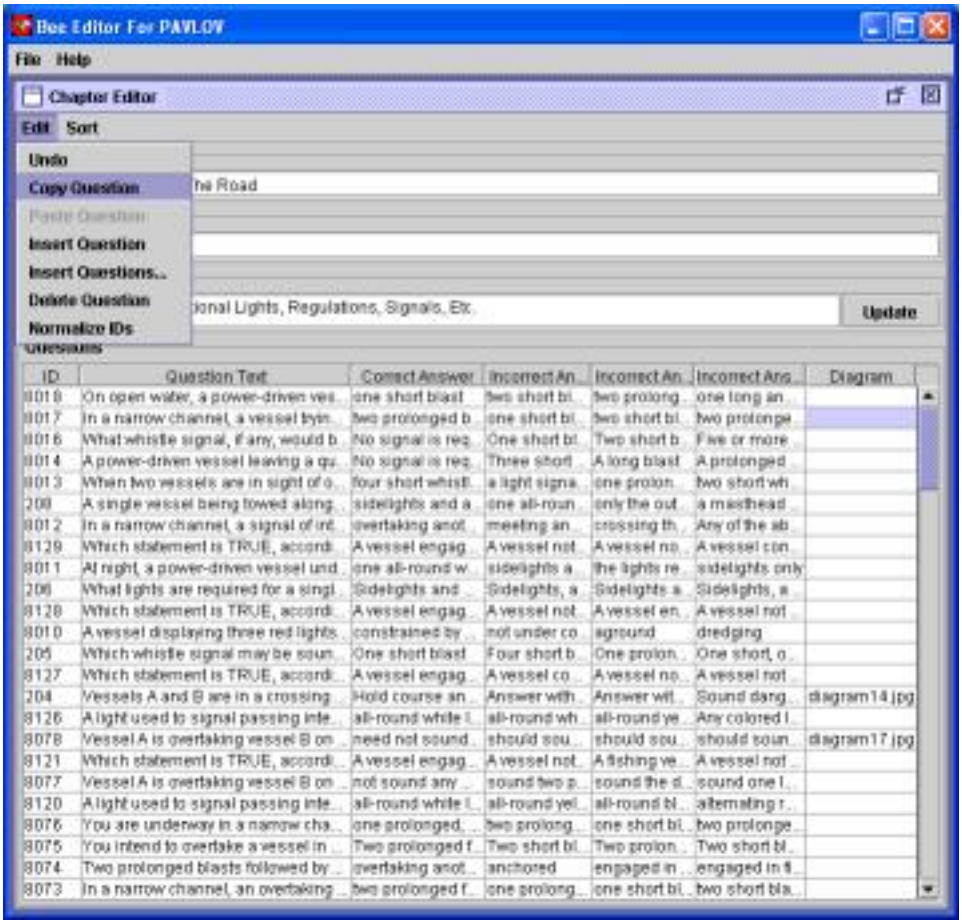


bee screenshot

8.6. The Chapter Editor

8.6.1. Slide 6: Chapter Editor

The edit menu allows you to copy, paste, and insert, and delete questions, as well as undoing your last edit and normalizing question ID's.



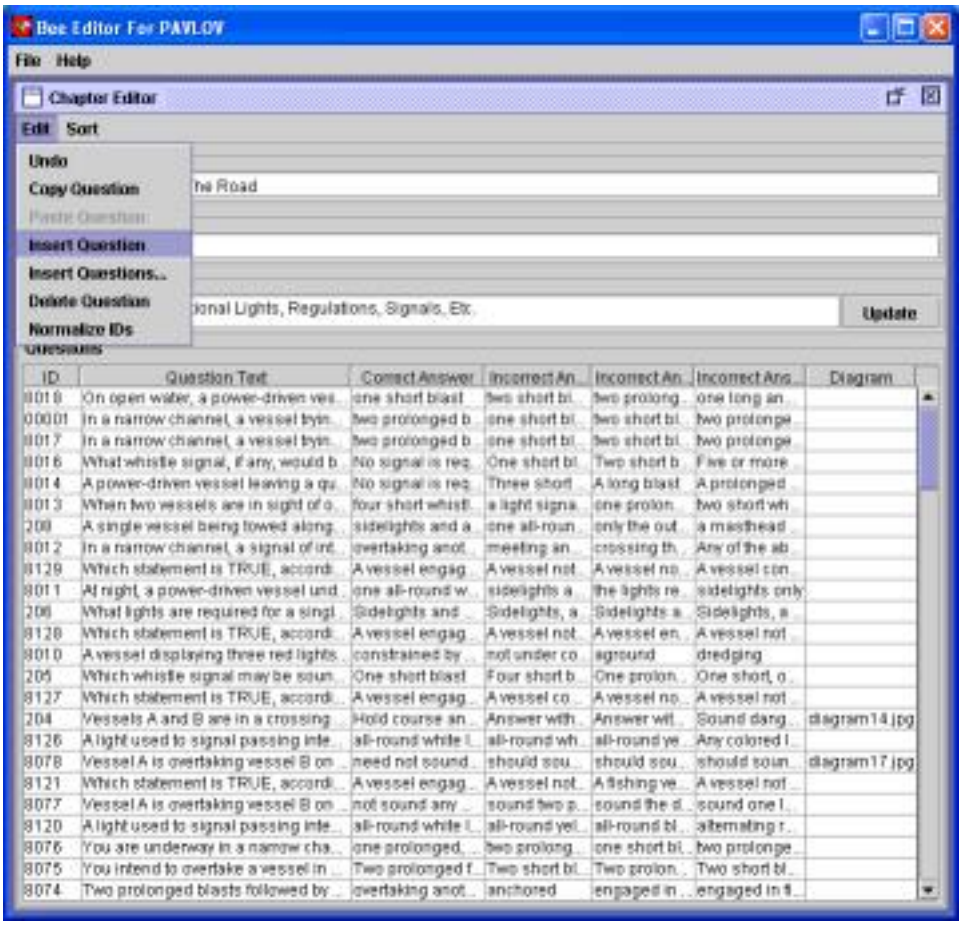
bee screenshot

8.7. Inserting Questions

8.7.1. Slide 7: Inserting

When inserting questions, Bee ensures that you have a unique question ID, minimizing the

possibility of duplicate IDs.

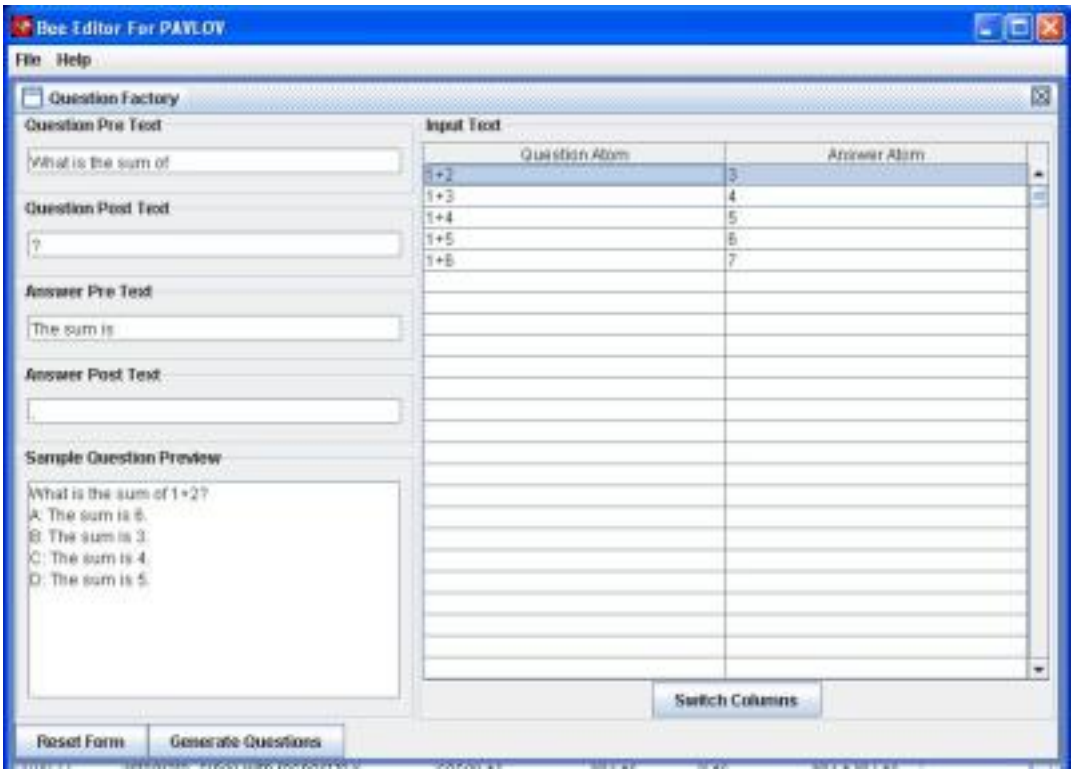


bee screenshot

8.8. The Question Factory

8.8.1. Slide 8: The Question Factory

If you're creating a bunch of similar questions, consider using the question factory. This saves you a lot of time and repetitive typing. Incorrect answers are filled in automatically. Keep in mind that Bee just compares answers character by character to pick wrong answers. So, "Richard Nixon" could be chosen as a wrong answer, where "Richard M. Nixon" was the right answer.

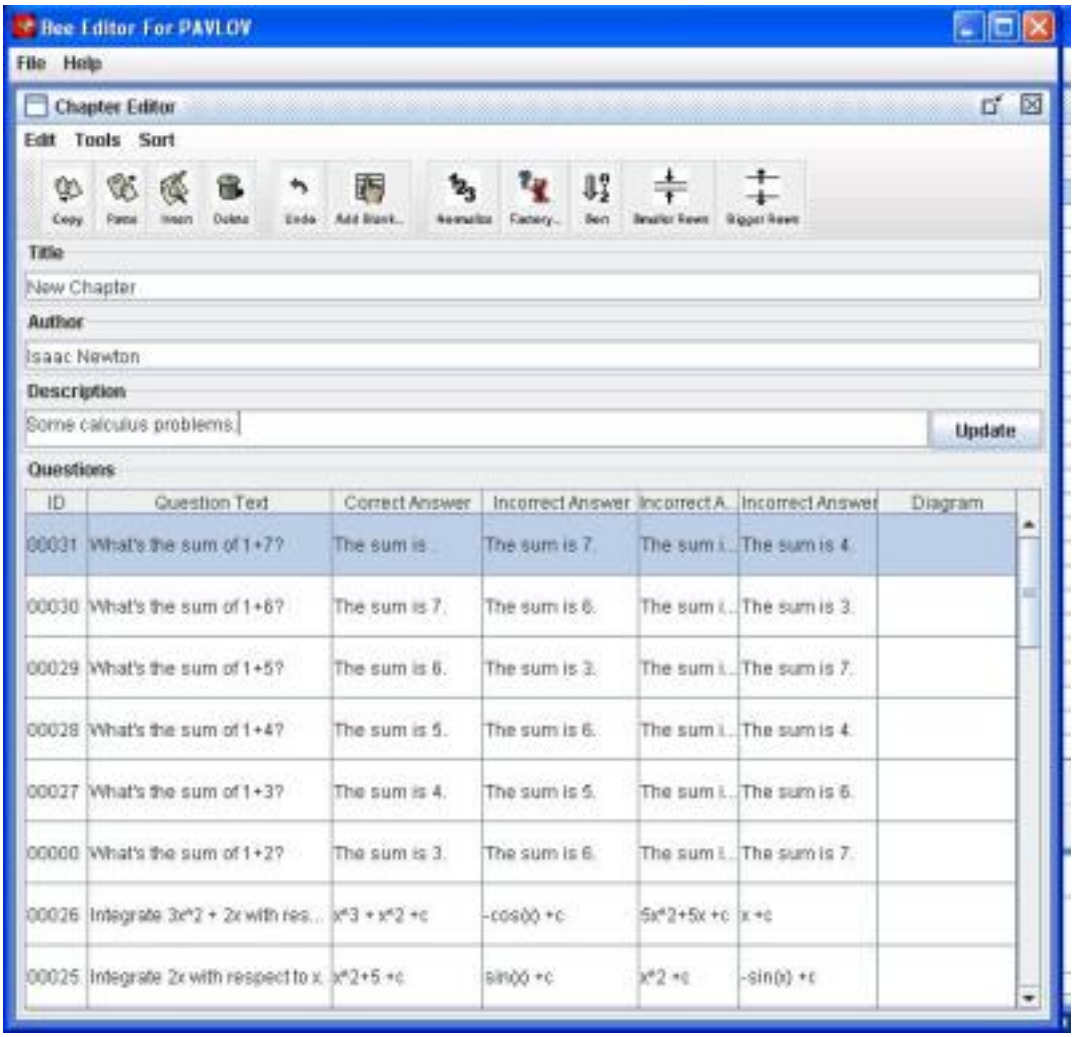


bee screenshot

8.9. Question Factory Output

8.9.1. Slide 9: Question Factory Output

This slide shows the questions generated by the question factory session in the previous slide. Notice how much redundant typing was saved.

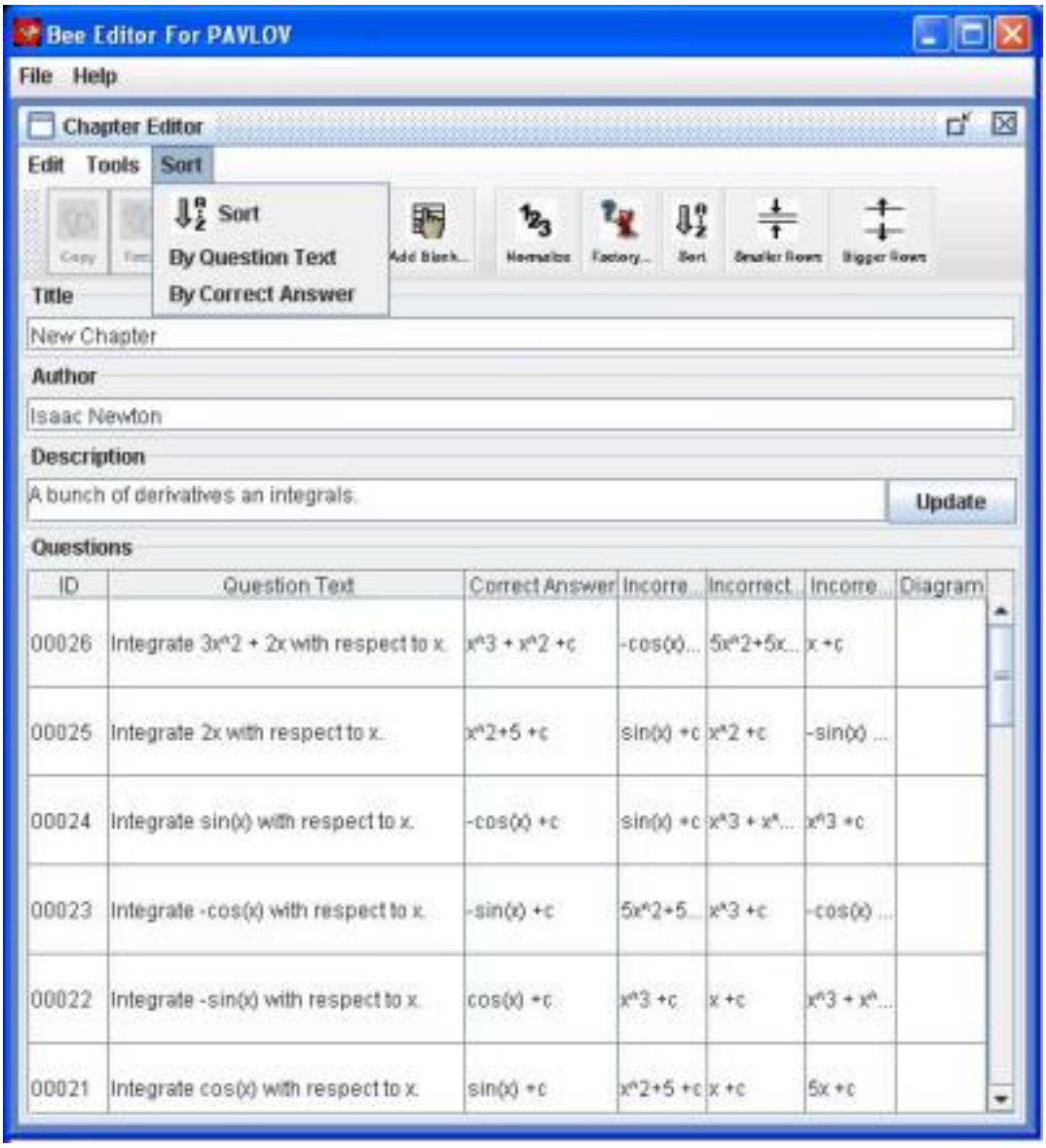


bee screenshot

8.10. Sorting Questions

8.10.1. Slide 10: Sorting Questions

You can sort all the questions by various criteria.

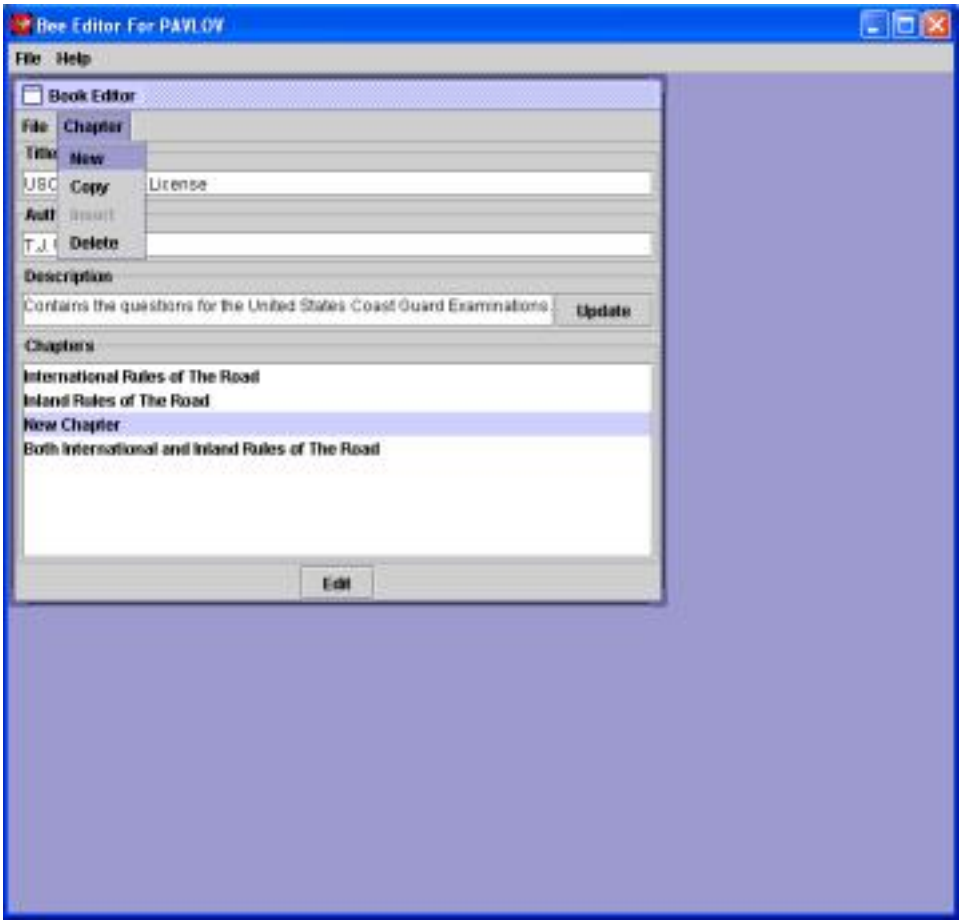


bee screenshot

8.11. More About The Book Editor

8.11.1. Slide 11: More About The Book Editor

Back to the Book Editor, notice that you can copy, insert, and delete whole chapters at once. You can also create new chapters from here.



bee screenshot

8.12. Bee is A Beta Release!

8.12.1. Slide 12: Bee is A Beta Release!

Note:

BEE is now a Beta Release with no problems reported. However the cautions in this document still are prudent to observe.

This is a very early release of Bee. It might behave strangely. If you use it, please work on

copies of the book files *not the original book files*

Here's the The Free Dictionary's definition:

alpha software

Noun 1. alpha software - a first release of a software product that is usually tested only by the developers

If you find problems or would like certain features, please feel free to post in the forums at the Pavlov project page.