Lab 1:
Introduction to UNIX and LaTeX

1.1 Introduction

Much of the work for this course will be performed on computers using the Linux operating system, which is part of a family of systems sometimes called "UNIX-like." For students accustomed to windowing systems, where most tasks are performed with pointing, clicking, and dragging, the command-line driven nature of a UNIX system can be frustrating at first, but it will reveal its usefulness as the semester progresses.

Along with this new operating system comes a new document preparation system - LaTeX. Unlike WYSIWYG (what you see is what you get) editors most students are familiar with, LaTeX compiles a document by reading several source files and by being told what a particular item is supposed to *be*, rather than how it is supposed to *look*. For example, the first command in a new LaTeX document tells the program what class of document you want (possibilities include a book, a report, or an article). With this information, LaTeX can determine the overall look and feel of the finished result without the user having to spend any time on the specific formatting of the document. While there is a fairly steep learning curve to get started with LaTeX, the results are more than worth it.

The purpose of this assignment is to familiarize you with the UNIX system, how you will access UNIX machines from Windows-based clusters and how you will document your work in the class using the LaTeX system. By the end of the lab, you will have the necessary resources to organize your file system, to copy required files from Dr. G's account into your own, and to create a LaTeX document that contains most of the features found in all your lab and recitation reports this semester.

1.2 Resources

The additional resources required for this assignment include:

- Books: neither
- Pratt Pundit Pages: `Emacs`, `LaTeX`, `PuTTY`, `UNIX`, `UNIX Tutorial`, and `X-Win 32`; Apple users will also want to reference `X11R6`
- Lab Manual Appendices: A - "LaTeX Information" and B - "Creating LaTeX Arrays, Tables, and Figures."

1.3 UNIX

The following parts of the lab manual are focused on teaching you the basic commands in UNIX and on getting your OIT account ready for doing work in the course. There is no assignment for this particular part of the lab, but you must follow the directions carefully to prepare for the work to be done in future labs. Some of the items in this section refer to Pratt Pundit pages, so go ahead and start the Firefox browser and point it to:

```
http://pundit.pratt.duke.edu
```

1.3.1 Accessing UNIX from the Laboratory

All the programs we will use for this class are already installed on the UNIX clusters across campus. These machines can be directly accessed at Teer and Hudson. However, you can also run programs remotely on these machines, and that will be the method used for the laboratory since the lab computers are running Windows and are not a part of the public computing system. Essentially, you will need to follow four steps to gain remote access to one of these machines:

(a) Run software to allow your computer to handle graphical information from a remote host. For Windows-based machines, this means running a program called **X-Win 32** that can interpret graphical commands from a UNIX-based source. This software has already been installed on the machines in the lab - go to the **Start** button and run the X-Win 32 program. An X-Win icon should appear in the lower right corner of the screen after the X-Win splash screen goes away.

(b) Tell X-Win which machines are allowed to send graphics to your screen. To do this, right-click the X-Win icon at the lower right and select **XConfig**. Now click on the **Security** tab. In the `Allow by Address` section, select `Allow all host addresses`, then click the `OK` button.

(c) Connect to the remote host through a terminal window. You will be accessing the remote host through what is called a terminal by using a program called **PuTTY**. This terminal will allow you to issue commands to the remote machine from your PC. Again, go to the **Start** button and find the PuTTY program (within the `PuTTY` folder). Information on how to connect using PuTTY can be found on its Pratt Pundit page. Read the instructions on that page and connect to a machine. Note that there are 21 machines in Hudson and 45 machines in Teer, so `hudson1.oit.duke.edu` through `hudson21.oit.duke.edu` and `teer1.oit.duke.edu` through `teer45.oit.duke.edu` should be available. Sometimes, however, one or more of these machines will be down for maintenance - if that is the case, simply pick a different number.

To test the connection, you should try to bring up something graphical. Among the quickest programs to run is `xterm` so type `xterm` in the SSH window and see if a new terminal pops up. If it does, type `exit` *in the new xterm window, not PuTTY.* If it does not, check to see if any of the steps above were done incorrectly. Common mistakes include forgetting to start the X-Win 32 program or spelling errors in the `setenv` command. If you are sure you have performed the steps above correctly but it still does not work, ask a TA for help.

For every lab session (except DAQ labs), you will need to make sure X-Win is running, open up at least one PuTTY terminal, and issue the `setenv` command for each terminal you open. You may also need to check the Security tab information though this *should* not change throughout the semester.

Note that for the recitations - and any time your are sitting directly at one of the UNIX machines - you will simply need to right-click and select `Open Terminal` to get a terminal window. Also, for Mac users, the command you will use to connect to a remote machine within the `X11R6` program takes care of setting the display, so you will not be using the `setenv` command either. More information about `X11R6` is available on its Pratt Pundit page.

1.3.2 UNIX Demonstration

During lab, we will run through several of the UNIX commands documented in the Pratt Pundit page `UNIX Tutorial`. At this point in the lab (which is to say, now that you have an open terminal window and the OIT machine you are remotely connected to knows to send graphical information your way), go ahead and navigate to that page on Pratt Pundit. You will also want to get out something to take notes on and with. Wait for the instructor to begin a hands-on tutorial of the various commands in UNIX. At the end of this session, you will have seen the passive commands to determine where you are in the file structure, how to move around, and how to determine what files and folders exist at various locations. Later during this lab, you will learn active commands to make changes to the file structure and to copy, move, and delete files.

1.3.3 Setting up your UNIX Account

For this course, you will be using space given to you by OIT for use on the public system. In order to allow the TAs and instructors to help you with any programming or documentation problems that may arise, you will need to give us access to part of your account - specifically the space used for this course. This access will allow us to help you with EGR 53 issues, but will be limited to one directory only. This access will also allow us to make sure all students are working on their assignments individually. Failure to assign the proper access to this folder may result in a delay in grading as well as a loss of points on your overall lab grade.

Now that you know a little bit about how to see where you are in the file tree and how to navigate from place to place, you can set up the folder for this course. The following list demonstrates how - note that the ␣ character is used to indicate a space.

- Change into your home directory.

  ```
  cd
  ```

  Remember - you can check to see where you are with the `pwd` command.

- Make a directory called `EGR53` in your home directory.

  ```
  mkdir␣EGR53
  ```

  Because you have given the `mkdir` command a relative path (i.e. there is no slash or path shortcut at the start) it will create the directory inside whatever directory you are in - which should be your home directory.

  - Note: Capitalization (or the lack thereof) is important
  - Hint: To ensure you have done this correctly, list the contents of your current directory by typing `ls`. The EGR53 directory should be there (as well as some other things).

- Give the TAs and professors of this course access to your EGR53 directory using the AFS commands. This may be the only time in the course (or your Duke career) you are required to use the AFS commands, so they are not explained in detail here. More information is available in the UNIX Tutorial.

  For now, simply type:

  ```
  fs␣setacl␣~/EGR53/␣mrg:egr53ta␣all
  fs␣listacl␣~/EGR53/
  ```

  The first command sets the access list for your `EGR53` directory and the second displays the access list for that directory. After the second command is issued, the terminal should respond with a listing of who has access to your `EGR53` directory. As an example, the listing for a student with a NET ID of `abcxyz` might be:

  ```
  abcxyz rlidwka
  mrg:egr53ta rlidwka
  system:administrators rlidwka
  system:anyuser l
  ```

  Be sure that one of the entries should read that `mrg:egr53ta` has `rlidwka` access (as in the second line above). If this is not the case, stop now and let a TA or the instructor know. It is **critical** that you have your account set with the proper permissions before continuing on. Common mistakes include spelling `setacl` or `listacl` wrong or adding a space after the tilde (∼) in either command (there should be a slash directly after the tilde).

Congratulations! Your EGR53 directory has been created and has been set up for the TAs and professors to read and grade your work. With that finished, for the rest of the semester, you must begin any work session for this class by changing into your `EGR53` directory. One reason for this is the way AFS grants permissions. Any new folder that is created inherits the permissions of the folder in which it is created. For this course, all folders you create from this moment on should be created from within the `EGR53` directory, thereby granting the group `mrg:egr53ta` access to it and its contents. If you make a mistake and create a directory *outside* this space, *even if you move it to* `EGR53` *later*, it will have the old permissions it inherited from wherever it was created, and `mrg:egr53ta` will not be able to access it. You would then have to repeat the `fs` commands above, replacing the `~/EGR53/` part with a path to whichever directory you need to fix. It is much easier just to start within `EGR53` and not have to worry about it!

1.4 LaTeX

For most of your labs and recitations, you will create a LaTeX document that presents your problem statements, codes, data, figures, and conclusions together in a professional manner. Most people in the course have never worked with a document preparation system like LaTeX before, and so it is important to both introduce the program as well as justify its use.

1.4.1 Assignment Overview

For this lab, your assignment is to make a duplicate of a document posted as a part of the assignment. Not a *photo-copy*, mind you, but a replica created by using LaTeX commands. This document is of class `article` with the package `amstex` (among others) added to it. To complete this assignment, you can use information contained within this lab handout, information in Appendices A and B, and the handouts posted at

`http://pundit.pratt.duke.edu/wiki/LaTeX`

especially the "Not So Short Guide..." and the "Short Math Guide..." in the `External Links` section. To view the assignment, you can go to the course web page at

```
http://classes.pratt.duke.edu/EGR53F09
```

navigate to the Handouts and click on the link for the `Lab 1 Assignment`.

The commands you will need to use can be found several places - certainly they are in the optional Kopka and Daly book, but you can also refer to Appendices A and B of this manual as well as a myriad of online tutorials and references.

Note that for the title block, you should put whichever section you are in instead of *Section 1* and you should use your own name and NET ID. In future weeks, you will also be adding a letter to your section number to specify which grading group you are in.

1.4.2 Getting Started and Report Skeletons

You will be given a skeleton `.tex` file for this lab (and for almost every lab). This skeleton will be copied form the public course directory on Dr. G's account. Generally, for labs, you will follow the following steps:

1. Make sure X-Win 32 is running, start PuTTY, and connect to your favorite machine.
2. Switch into your `EGR53` directory and create a `lab1` directory inside it:

   ```
   cd ~/EGR53
   mkdir lab1
   ```

3. Switch to your `~/EGR53/lab1` directory:

   ```
   cd lab1
   ```

4. Copy all relevant files from Dr. G's public `lab1` directory, including the skeleton for the report:

   ```
   cp -i ~mrg/public/EGR53/lab1/* .
   ```

   Do not forget the "." at the end.

1.4.3 Starting the Document

Refer to Appendix A for instructions on writing, saving, and processing the LaTeX document. For this assignment, the *base* name of your file (i.e. the file name without any of the extensions) will be `lab1`. You will be creating a text document named `lab1.tex` that contains all the code LaTeX needs to process the document. The name of the skeleton file is `Lab1Sample_F09.tex` so you will want to first make a copy of this file and call it `lab1.tex`:

```
cp␣Lab1Sample_F09.tex␣lab1.tex
```

Once that is completed, there are a few basic steps to remember:

- Open the source file by typing

  ```
  emacs␣lab1.tex␣&
  ```

  in the xterm window. Do not forget the &! You will want to be able to have the emacs window open while *also* being able to run LaTeX on the file in the terminal window.

- A header must be included, or LaTeX will get confused. This includes at least the class of the document as well as a beginning and an end.

- Use the `latex, kdvi,` and `dvips` commands to process, view, and print your document. You can also use `kdvi`'s print feature to print your document or `dvips` to produce a PostScript document that can be viewed with `kghostview`.

During lab, we will go through the complete process contained in Appendix A to start your document and produce an equation. You will then be allowed to work on completing the rest of the assignment.

1.4.4 Document Components

The following subsections give information, hints, and tips for completing the different parts of the assigned document: header, title block, table of contents, equations, tabular and array environments, lists, text files, and figures.

Header

The header of a LaTeX document provides information about the size of the paper as well as about any packages that need to be loaded to define commands not found in basic LaTeX. The header for LaTeX files for this class will generally be:

```
\documentclass{article}
\usepackage{amsmath}    % loads AMS-Math package
\usepackage{epsfig}     % allows PostScript files
\usepackage{listings}   % allows lstlisting environment
\usepackage{moreverb}   % allows listinginput environment
\usepackage{vmargin}    % allows better margins
\setpapersize{USletter} % sets the paper size
\setmarginsrb{1in}{0.5in}{1in}{0.2in}{12pt}{11mm}{0pt}{11mm} %sets margins
\begin{document}
```

Note the descriptions of the packages are given in comments - and that comments in LaTeX files are set off using the percent symbol.

Title Block

The code for the title block is given below and in the skeleton. Note the different font sizing and line spacing elements:

```
\begin{center}
\rule{6.5in}{0.5mm}\\~\\
{\bf \large EGR 53L -- Fall 2009}\\~\\
{\huge \bf \LaTeX~Assignment}\\~\\
Anne O. Nymous (aon)\\
Lab Section 1, Tuesday 8:30-11:20\\
Due XX XX, 2009\\~\\
{\small I have adhered to the Duke Community Standard in completing
  this assignment.  I understand that a violation of the Standard can
  result in failure of this assignment, failure of this course, and/or
  suspension from Duke University.}
%% YOU WILL SIGN THE DOCUMENT HERE
~\\~\\~\\~\\
\rule{6.5in}{0.5mm}\\
\end{center}
```

The code above is explained as follows:

1. Start a centered environment with `\begin{center}`.
2. Add horizontal lines of a certain width and thickness with the `rule` command and two arguments. Note here that LaTeX understands units - in fact, for many different arguments, LaTeX requires units.
3. Change to **bold** font with the `bf` command. The group of characters to be bold-ed is surrounded by brackets which tells LaTeX the extent of text to put in bold. This particular construction is used several times throughout this document and is handy for both font faces and font sizes.
4. Change font sizes with `large`, `huge`, and `small`; there are other sizes described in the LaTeX book.
5. Use two backslashes (\\) to move to the next line.
6. A ~ will create a space between LaTeX and Assignment. The code for LaTeX is `\LaTeX`.
7. Use the ~\\ on a line by itself or at the end of a line to create a space between two lines. Be careful here - LaTeX will not allow you to go to a new line if the current line does not contain anything. The ~ puts a "hard" space on a line, after which LaTeX will let you go to the next line.
8. Be sure to put in your own name, lab section, and lab time. Eventually you will also be assigned a lab subsection to include.
9. End the centered environment with `\end{center}`. Every environment needs a distinct beginning and end.

Automated Lists - Table of Contents and List of Figures

LaTeX can automatically generate several different kinds of list, including a table of contents and a list of figures. The commands are very easy:

```
\tableofcontents
\listoffigures
```

To use them, you must first make sure that the commands show up in the right place in your document (in this case, just underneath the end of the `center` environment. Second, you must remember to run LaTeX at least three times when processing the file. The first time through a document, LaTeX will not have any idea what (or where) the contents actually are so the commands will not produce anything. The second time

through, the commands will produce the appropriate lists, but in the process of doing so may push items to different pages, resulting in an *incorrect* table of contents or list of figures. The third time through should *generally* correct those issues. For extremely large documents, even more passes with LaTeX may be required. Conversely, if a document only has minor edits, the previous pagination information might still be valid. Still, before turning in a final lab report, be sure to have run LaTeX at least a final three times.

Equations

For this lab report, the *substance* of the document will start on its own page, so add the command

```
\pagebreak
```

under the list of figures to move to the next page. Note the space under this command in the skeleton - this is important. There are several LaTeX commands that will wait to execute until a paragraph or section of the document is completed - the page break is one of them. Putting the blank space in the code tells LaTeX that the section of the document is finished and the page break should go here.

Once you have started with a new page, the next section of the assignment will use the `align` environment to line up the equals signs of four equations. Note that the `align` environment is a math environment *in and of itself* so we do *not* need to explicitly enter math mode using dollar signs or other delimiters. See Appendix A for more information about math environments. Generally, an environment such as `align` is used for large blocks of math (equation lists, arrays, etc.) while the inline math delimiters (the $) are used for inserting a smaller mathematical or symbolic element - such as in saying "A may equal πr^2 but pies are round" which is coded as:

```
$A$ may equal $\pi r^2$ but pies are round
```

One **extremely important** note about the `align` environment - it will *not* allow LaTeX to process the file if there are empty lines between the `begin` and `end` commands. In other words, the following will work:

```
\begin{align*}
\alpha&=\beta \\
\gamma&=\delta
\end{align*}
```

and will produce:

$$\alpha = \beta$$
$$\gamma = \delta$$

while the following, seemingly identical code, will fail:

```
\begin{align*}
\alpha&=\beta\\

\gamma&=\delta
\end{align*}
```

Specifically, the error given is:

```
Runaway argument?
 \alpha &=\beta
! Paragraph ended before \align* was complete.
```

Other items of note:

1. \begin{align*} - used to start an unnumbered **align** environment. The $*$ suppresses the numbering of the equations.

2. Insert the & symbol before each equals sign. This symbol tells the `align` environment to line up each equation by aligning the character after the & symbol. Since the first equation has more than one equals sign, make sure the correct one is lined up. LaTeX and the `align` environment can align lines based on letters, numbers, or symbols. For example:

$$\begin{aligned} 012345f678 \\ ab3d \\ 6+1&=7 \\ 6&+1=7 \\ \text{left side}&\int\text{right side} \\ \text{missing an ampersand} &= \text{stuff gets shoved left} \end{aligned}$$

   which is produced with:

```
\begin{align*}
012345&f678\\
ab&3d\\
6+1&=7\\
6&+1=7\\
\mbox{left side}&\int\mbox{right side}
\mbox{missing an ampersand}=\mbox{stuff gets shoved left}
\end{align*}
```

   has the first five lines aligned on the f, 3, =, +, and $\int$, respectively. The sixth line, having no & to align on, is treated as being left-hand side only. Note also the fourth and fifth lines, where the exact same equation is lined up differently based on where the & is placed.

3. For the first equation, you will need the commands \int (integral), \infty (infinity), ^ and _ (for the limits of the integral; see text), and \tau (Greek letter). Remember that spaces can be *forced* in math mode with the $\sim$.

4. For the second equation, you will use

   - \frac{a}{b} to make a fraction $\frac{a}{b}$, where a and b represent the codes for the numerator and denominator, respectively.
   - \lim to keep the term lim out of italics (versus *lim*, which is incorrect). The Kopka and Daly book has a list of math terms that should not be displayed in italics.
   - \rightarrow to produce a...right arrow.[1]
   - \Delta (capital D is important; \Delta=Δ while \delta=δ). Some upper-case Greek letters are the same as their Latin equivalent and thus do not have commands. For example, ζ is given by \zeta but Z is merely an upper-case Z - there is no \Zeta command. Also, lower case omicron is simply an o.

[1] Any guesses what \longrightarrow produces?

5. For the third equation, you will need to create an array.

   - Use \begin{array}{cc} to begin an **array** environment with two columns, both of which are centered (the **cc** part), and \end{array} to end an **array** environment, regardless of what columns there are. Check Appendix B for instructions on creating the array.
   - You will also need to surround the array with brackets and parentheses, using \left[, \right] and \left(, \right), respectively.
   - Other commands:
     - \mbox{text} - used to place regular text in an equation (instead of using italics and math spacing)
     - the subscript, _{text} to put the content of text as a subscript of the character preceding the underline character. If there is a single character in a subscript or superscript, you can omit the curly braces, but if not, you *must* make sure that the entire subscript or superscript is surrounded with curly braces. If not, LaTeX will *assume* that only the first character is included. For example:

Code	Result
x_1^3	x_1^3
x_12^34	$x_1 2^3 4$
x_{12}^34	$x_{12}^3 4$
x_12^{34}	$x_1 2^{34}$
x_{12}^{34}	x_{12}^{34}

6. For the fourth equation, you will need the following additional commands:

   - \pm - plus/minus
   - \sqrt{a} - used to put a under a square root (i.e. $\sqrt{a}$)

7. Finish with \end{align*} which ends an unnumbered **align** environment

Tabular and Array Environments

Tabular LaTeX has commands that allow you to organize text or math into arrays. You will use the tabular command to create the primarily-text table of chemical equations. Note that since the tabular command does *not* start math mode, you will need to use dollar signs ($) to surround commands such as Greek letters and subscripts - these only work in math mode and if they are missing, LaTeX will give you an error while processing the file. Also, to center the tabular environment, you will want to put it inside a center environment.

1. Begin with \begin{tabular}.
2. Provide the justification argument. We want to create a table with a right-justified first column, a left-justified second column, and a line between them. Use Appendix B to figure out how to do this.
3. Follow the instructions in Appendix B for using tabular. Remember to include the & between columns, \\ at the end of each line, and \hline where it is needed.
4. Note that the first equation can be completely surrounded by the dollar signs, but the second equation cannot. The H and O are not in italics so make sure they are not surrounded by dollar signs. Another option is to use \mbox{text}.
5. Note the typewriter text. Use {\tt yourtext}.
6. Finish with \end{tabular}.

Array For the velocity equations table, you will be using the `array` environment instead of a `tabular` command, meaning you will need to be in math mode before starting the `array`. The `array` environment is used for mainly math-based entries. The easiest way to get into math mode for this will be to use the `align*` environment. One convenient side effect of this is that items in an `align*` environment are automatically centered.

1. First, increase the inter-row spacing by a factor of 1.5 with the `\arraystretch` command. Refer to Appendix B.
2. Tell LaTeX to begin an unnumbered `align*` environment to start math mode.
3. Use `\begin{array}` and provide the justification argument. We want an array in which both of the columns are centered, and there are lines between the columns and around the sides.
4. Remember that it is necessary to use `\mbox` whenever normal text is inserted into the array, such as in "Equation" and the "Description" column.
5. Note the double line between the first two rows.
6. Use the & to separate columns and \\ at the end of each row.
7. You will also need these commands:
   - `\hat{a}, \vec{a}` - math accents as described in Kopka and Daly
   - `\imath` and `\jmath` - used to produce $\imath$ and $\jmath$ for cases where a hat, tilde, vector, or other symbol would be placed above an i or j rather than the typical dot. That is, generate $\hat{\imath}$ as opposed to $\hat{i}$. There is no such thing as `\kmath` since k is not dotted.
8. End the array and end math mode.

Lists

LaTeX has several commands that allow you to make lists. Of these, the most common are `enumerate` and `itemize` where the former will list with numbers or letters while the latter will use bullets. In each case, the `item` command is used to start a new part of the list.

1. Type the first line as you see here ("Things I learned...")
2. Type `\begin{itemize}` to begin the bulleted listing environment.
3. Before each bullet point, type `\item`.
4. Type the first entries as you see here then make sure you add your own answers for the last three bullet.
5. End with `\end{itemize}`.

Appendices

Your lab reports for this class will generally contain two appendices - one with the codes you used to solve the different problems and one with all the figures generated to assist in analyzing the data. The code for these appendices will be included in the skeleton files and are described below.

Text Files

LaTeX can be "taught" to directly import text files while processing. This is superior to "cutting and pasting" your code files because with the import method, LaTeX will automatically use the most recent version of your file. Furthermore, if you end up changing your files later, all you have to do is re-run LaTeX rather than spend time trying to replace whatever code you copied into your `.tex` document. The code for importing two sample text files is already given in the lab skeleton as:

```
\appendix
\section{Codes}
\subsection{Listing of sample header for original code}
\listinginput[1]{1}{Header1.m}

\subsection{Listing of sample header for modified code}
\listinginput[1]{1}{Header2.m}
```

Note that the `appendix` command serves to both re-start the section counter and to tell LaTeX to letter the sections rather than number them. The `listinginput` command comes from the `moreverb` package that was loaded in the header of the document. The arguments are the line numbering interval ([1] means number every line), the line to start numbering ({1} means start with the first line of code) and the path to the file to import.

Figures

LaTeX has special environments that will allow you to give it information to be displayed without specifying exactly where in the flow of a document it will go - these are called *floating* environments. As examples of floating environments, if you are reading a newspaper or a textbook, you will often notice that Tables and Figures may show up at the top of a page, at the bottom, or somewhere in between. They will be referenced in the text but may not appear at the exact location in the document where the reference appears.

The main floating environments in LaTeX are the `table` and the `figure`. For your lab reports in this class, you will use the `figure` environment in an appendix so that all the figures will be contained at the end of a document. This is because the lab reports for this class will generally not have enough text to surround the figures in a meaningful way - trying to include several figures without text to surround them will make the figures show up in odd places, generally far, far away from their relevant text. Sample code for including figures in the appendix is also given in the lab skeleton; specifically:

```
\section{Figures}

\begin{figure}[htb]
\begin{center}
\epsfig{file=drawing.eps, width=3in, angle=-90}
\caption{Drawing from ME 150L test.}
\end{center}
\end{figure}

\begin{figure}[htb]
\begin{center}
\epsfig{file=SampleFigure.eps, width=2.5in}
\caption{Sample MATLAB figure.}
\end{center}
\end{figure}
```

Appendix A has more information about figures and floating environments.