Crash Intro to GameTeX

"Make sure you have at least a basic understanding... before doing arbitrary global hacking."

This is a short document[1] on how to use the GameTeX template system. It will teach you the basics of how to use it to write and produce a Guild game. This document will not attempt to teach (La)TeX basics, Athena/UNIX basics, or the basics of emacs or your preferred text editor.

What is GameTeX? GameTeX (or GameTeX) is a large macro package built on top of LaTeX, which is a macro package aimed at journal authors, itself written in TeX, a typesetting language/system. If you are a math, science, or engineering student, there is some chance you will write a few papers, journal entries, or a thesis in LaTeX. All of the Guild Camp sheets, including this one, are produced using GameTeX. GameTeX is actually based on ε-TeX, an extension of TeX. On most modern TeX installations (including debathena and anything using TeX Live), e-TeX is the default program used by the `latex` command (though on some platforms, you'll need to use the `elatex` command instead).

To use GameTeX, you need to know the basics of LaTeX (though you shouldn't need to know all the details of programming in TeX). SIPB's *Inessential LaTeX* is a decent place to start learning about LaTeX, and you can find many similar resources on the internet. To edit GameTeX files, use any text editor. If using emacs or some other editor designed for editing code, you may want to turn on syntax highlighting (e.g. font-lock-mode), if it is available for latex.

GameTeX's documentation is almost entirely in the form of README files. There is one for every directory, plus a few extra ones covering special topics. Everything in this document is covered in those files; you should read them. Each of the sheet directories has a `README.tex` file which also doubles as an example sheet. When starting a new sheet, it's good to begin by copying the `_template.tex` file (a much more bare-bones sheet) in that directory, and using `README.tex` as a reference/example.

Getting Started The latest version of GameTeX is available at `http://web.mit.edu/kenclary/Public/Guild/GameTeX/` and from Athena at `~kenclary/Public/Guild/GameTeX/`. That directory has a README file, a copy of the GameTeX tree, and a bzip'ed tarfile of the GameTeX tree. You have two decisions to make right away:

1. How to organize your GameTeX files: you might rename your copy of the GameTeX tree to something like `Thebes/` or `YourGame/`, and put any new directories (`YourGame/Notes/`, `YourGame/Admin/`, etc.) alongside the existing GameTeX directories (`YourGame/Bluesheets/`, `YourGame/Charsheets/`, `YourGame/Lists/`, etc.). This is the most common choice. You might instead want to have the entire GameTeX tree underneath your top-level directory, so `YourGame/GameTeX/` is right next to `YourGame/Notes/`, `YourGame/Admin`, etc. This latter choice may appeal to software engineers working on games with complex systems outside of GameTeX.
2. Your game's classname: by default, the game's classname is "game," meaning your `.cls` file is at `LaTeX/game.cls` and all of your game's latex documents start with `\documentclass[...]{game}`. You may want to change this to something like "yourgame," so your `.cls` file is at `LaTeX/yourgame.cls` and your files start with `\documentclass[...]{yourgame}`. `GameTeX/README` gives a full description of how to do this; the simplest method is to run the `Extras/changeclass.pl` script. `cd` to the `Extras/` directory, make sure `changeclass.pl` has execution permissions, and run:
   ```
   ./changeclass.pl /mit/username/blah/YourGame/ game yourgame
   ```
   substituting the appropriate classname and path to your local copy of the game. It is probably best to do this before doing the initial SVN checkin of your copy of the GameTeX tree.

Next you should visit your `.cls` file, at `LaTeX/yourgame.cls`. It contains a large number of settings you can change, notably including the name and date of your game. Read through the comments in that file and make changes.

[1]last modified 5 June 2012

Your Environment Each GM needs to set up their enviroment to be able to run latex on the game's sheets.[2] This is covered in detail in `GameTeX/README`. This document covers the short version. Open your `.bash_enviroment` file and add these lines:

```
export yourgame=/mit/username/blah/YourGame
export TEXINPUTS=.:$yourgame/LaTeX/:
```

Substitute the appropriate classname and path to your local copy of the game in either case. If you have GameTeX trees for multiple games, you'll want to modify the `TEXINPUTS` line to use all of them, as detailed in `GameTeX/README`.

Once that's done, you'll need to make sure those lines have been executed. Logging out and back in is the most straightforward way to do this.

Lists and Sheets The majority of GameTeX content that you will write for your game will be in one of two forms: prose in sheets and data in list files. Each type of sheet has its own directory, each with a `README.tex` file. The lists are in `Lists/`, in the form of `-LIST.tex` files. You do not run latex on the list files; they get automatically included when you latex a sheet or a production file.

The list files define your game's database of information. In brief, there are "owners," like characters, and "elements," like abilities and bluesheets. Nearly everything in the game has an element or owner macro associated with it, like `\cJamesBond{}` and `\aHacking{}`. Everything that gets printed for the game should be owned by an owner: either a character or a "place" (places own things that don't start in character packets, like signs and items scattered around game). Ownership is specified in the owner's list file (usually `char-LIST.tex` or `place-LIST.tex`), with lines like `\s\MYabils {\aHacking{}\aDriving{}}` in the owner's definition. The syntax for creating and using owners and elements is covered in great detail in `Lists/README`; pay close attention to it, as the vast majority of GameTeX errors come from mistakes in creating and using elements.

Each sheet has an associated owner/element. They are linked both ways: `\cJamesBond` has a `\s\MYsheet{jamesbond.tex}` line and `jamesbond.tex` has a `\name{\cJamesBond{}}` line. Most prose is written into the sheet, while most technical information, like stats and ownership, is in the list file. Ownership (and even element definition) can be done inside character sheets; this is only likely for mempackets, where you may want to write mempacket prose and character prose in the same document. `Charsheets/README.tex` has examples.

Workflow When printing your game, you'll generally use the latex/dvips/lpr workflow. You produce and print a file with:

```
latex filename.tex
dvips filename.dvi
lpr -l filename.ps
```

You may want to preview either the `.dvi` or `.ps` files before printing. Control which printer `lpr` sends the file to with `-P`. Do not commit any `.dvi`, `.aux`, `.log`, or produced `.ps` files into your repository.

Production You print game by latex'ing and printing some or all of the `-PRINT.tex` files in `Production/`. For detailed documentation of this, read `Production/README`. Each kind of document (like charsheets, bluesheets, or ability cards) has an associated `-PRINT.tex` file. Each file produces all of that type of element for the game, sorted by owner, as a single print job.

Between print jobs for each type, you can change the color of paper. When you are done printing, you collate the material for each owner together (by the owner tag in the top corner of each document) and stuff them into packets.

Mailing List `gametex-gms@mit.edu` is for GameTeX announcements and occasional discussion. You can add yourself using blanche, listmaint, or other moira programs.

[2]Some technophobic GMs will believe they are exempt from this. They are wrong, and will regret not being able to latex their files.