Programming Proverbs

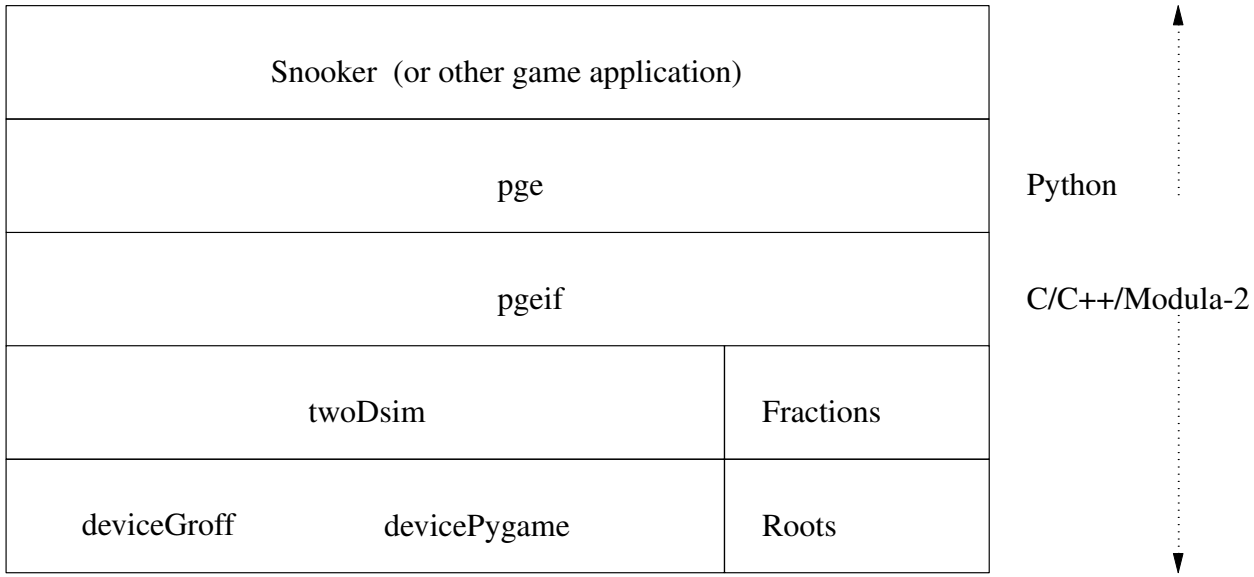
- 15. “Avoid tricks.”

- Henry F. Ledgard, “Programming Proverbs: Principles of Good Programming with Numerous Examples to Improve Programming Style and Proficiency”, (Hayden Computer Programming Series), Hayden Book Company, 1st edition, ISBN-13: 978-0810455221, December 1975.

Internals of PGE (Python)

- during this lecture we will start to look at the internals of PGE
- we will concentrate on the Python module `python/pge.py`
- we can see that this sits near the top of the various software levels of our game

Internals of PGE (Python)



pge.py

- in the last lecture we saw how foreground and background objects are maintained in `python/pge.py`
- we also saw how objects were created and are checked at runtime for type consistency
- in this lecture we will examine how integrates with Pygame

Obtaining the source to pge

- open up and command line terminal and type:

```
$ mkdir -p $HOME/Sandpit
$ cd $HOME/Sandpit
$ git clone https://github.com/gaiusm/pge
```

- the files for the `pge` package will be available in `pge`

Source directory structure of the pge package

- the key directories are:
- `pge/c`
  - C source code
- `pge/m2`
  - Modula-2 source code
- `pge/i` swig interface (PGE API definition)
- `pge/python`
  - python code, (`pge.py` and Python tools, such as `pgeplayback` and `max2code`)

Key configuration files

- `pge/configure.ac`
  - source code for the classic `configure` command
- `pge/Makefile.am`
  - source code for `Makefile` in the top directory of the build tree
- `pge/c/Makefile.am`
  - source code for `c/Makefile` in the build directory
- `pge/m2/Makefile.am`
  - source code for `m2/Makefile` in the build directory

Building pge from source

- you can choose either Modula-2 or C

- ```
  $ cd $HOME/Sandpit
  $ mkdir -p build-pge
  $ cd build-pge
  $ ../pge/configure --enable-langc
  $ make
  ```

- in this case the pge package is built from > 90% of C source files
  ```

Testing your build

- one simple test is to run the trapped example

- ```
  $ cd $HOME/Sandpit/build-pge
  $ ./localrun.sh ../pge/examples/trapped/trapped.py
  ```
  ```

Revisiting pge/python/pge.py

- a potential problem surfaces during the development of pge and its integration with `Pygame`
- `Pygame` controls the input sources: keyboard, mouse, joystick
  - and output devices, screen, audio etc
- internally `Pygame` uses an event queue on which all input events (keypress, mouse button press) are posted
- events are meant to be read by the Pygame user application code

Revisiting pge/python/pge.py

- in order for PGE to be integrated with Pygame we need to ensure that
  - a program does not block waiting for an input event

- otherwise the physics engine will be starved from updating itself in real time

The PGE event queue

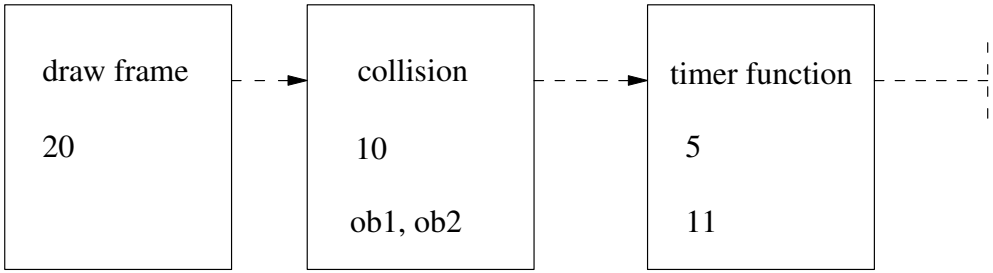
- internally pge also maintains an event queue (different from the `Pygame` event queue!)

- the pge event queue is a time ordered list of future events
  - each event predicting what will happen in the future
  - it might be a draw frame event
  - or a collision event
  - or a timer activation event

- `pge/python/pge.py` coordinates the pge event queue and also the `Pygame` event queue (input source)

The PGE event queue

Relative event Q



- notice the different kinds of events
  - relative time ordered

- although there might be another collision event at, say, time (20+10+1) 31 there is no point predicting it as the event at time 30 might change the world

pge/configure.ac

- is the source file which builds the file `pge/configure`
- it is written in a language called `autoconf` which is compiled into a portable shell script
- `autoconf` allows you to specify dependancies such as the build machine must have certain tools: `awk`, `cpp`, `c++` and `make`
  - and the build machine must also have the `-lpth` library
- it also allows you to add extra configuration arguments
  - ie pge can be built using C sources, or built from Modula-2
  - and one can enable maintainer mode (dont do this unless you know what you are doing!)

Example sections of pge/configure.ac

```
AS_MKDIR_P(c)
AS_MKDIR_P(m2)
AS_MKDIR_P(python)

LT_INIT
...
AC_ARG_ENABLE([maintainer],
[  --enable-maintainer    Turn on maintainer],
[case "${enableval}" in
 yes) maintainer=true ;;
  no) maintainer=false ;;
   *) AC_MSG_ERROR([bad value ${enableval} for --enable-maintainer]) ;;
 esac],[maintainer=false])
AM_CONDITIONAL([MAINTAINER], [test x$maintainer = xtrue])
...

AC_HAVE_LIBRARY(-lpth)
AC_SUBST([langm2])
AC_SUBST([langc])
AC_SUBST([maintainer])
```

Example sections of pge/configure.ac

- we can see that `autoconf` allows us to use a library of routines
  `AS_MKDIR_P`

- and also we can create our own code to drive an option in rule
  `AC_ARG_ENABLE`