

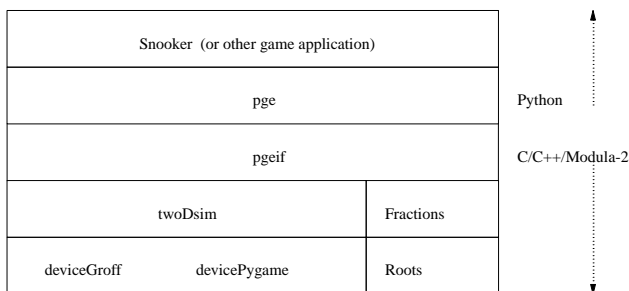
Internals of PGE (Python)

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

slide 3
gaius

Internals of PGE (Python)

■



slide 4
gaius

What does `pge.py` provide?

- for full details see the section [PGE Python API](http://floppsie.comp.glam.ac.uk/Glamorgan/gaius/pge/homepage.html) (<http://floppsie.comp.glam.ac.uk/Glamorgan/gaius/pge/homepage.html>) in the online documentation
- it provides the ability for users to create the following objects
 - colour
 - circle
 - polygon
 - text

What does pge.py provide?

- circle, polygon and text objects appear on the screen at a position
 - they are also given a level
- only objects at level 0 are handled by the physics engine
 - objects declared at any level < 0 are in the background
 - objects declared at any level > 0 are in the foreground
- all objects are drawn in level order

What does pge.py provide?

- `pge.py` front ends the object creation
- it creates a Python object for the `circle`, `polygon`, `colour` and `text` objects
 - the `pge.py` object contains a `type` field which is set to one of: `colour_t`, `box_t`, `circle_t`, `fb_box_t`, `fb_circle_t` or `fb_text_t`
- it then checks the parameters to methods in `pge.py` to ensure that users do not try and create a polygon of colour circle!

What does pge.py provide?

- ```
#
text - place text string, s, at position [x, y] in col
The size will be in font size and placed in the
foreground or background depending upon level.
You are not allowed to place text in level zero
#
def text (x, y, s, c, size, level):
 global device, screen
 c._param_colour ("fourth parameter to text is expected")
 if level == 0:
 _errorf ("not allowed to place in level 0")
 else:
 ob = object (fb_text_t, \
 [x, y, s, size, c._get_pgeif_colour ()],
 _add (ob, level)
 return ob
```

## What does pge.py provide?

- `pge.py` will check to see that users do not try and assign velocity or acceleration to fixed objects
- and ensure that the object still exists

## What does pge.py provide?

```
#
velocity - Pre-condition: an circle or polygon object
which is not fixed and exists at level 0.
Post-condition: assign the velocity (vx, vy)
to this object.
#
def velocity (self, vx, vy):
 self._check_type ([box_t, circle_t], "assign a velocity")
 self._check_not_fixed ("assign a velocity")
 self._check_not_deleted ("a velocity")
 self.o = self._check_same (pgeif.velocity (self.o, vx))
 return self
```

## Managing foreground/background and the physics engine

- pge.py coordinates the foreground, background and physics engine
- a foreground and background circle is never seen by the physics engine
- it is managed locally in pge.py and Pygame
- foreground/background objects have the type field set to: fb\_circle\_t, fb\_box\_t or fb\_text\_t

## Managing foreground/background and the physics engine

```
#
circle - place a circle at coordinate (x, y)
The circle has a radius, r, and is filled with colour,
If the level == 0 it is placed into the physics engine.
A level < 0 is placed into the background.
A level > 0 is placed into the foreground.
#
def circle (x, y, r, c, level = 0):
 c._param_colour ("fourth parameter to box is expected to be a c")
 if level == 0:
 id = pgeif.circle (x, y, r, c._get_pgeif_colour ())
 ob = object (circle_t, id, c, level)
 _register (id, ob)
 else:
 ob = object (fb_circle_t, [x, y, r, c._get_pgeif_colour ()])
 _add (ob, level)
 return ob
```

## Automatic PGE API documentation

- notice the naming convention, most of the internal methods have an underscore prefixed to them
- this allows for the internal methods to be excluded from documentation when it is automatically built
  - by pge/tools/py2html.py
  - this utility generates the API documentation from the comments prior to the method definition
  - it creates a subsection for each method
  - and a function index into the documentation

## Managing foreground/background and the physics engine

```
#
circle - place a circle at coordinate (x, y)
The circle has a radius, r, and is filled with colour,
If the level == 0 it is placed into the physics engine.
A level < 0 is placed into the background.
A level > 0 is placed into the foreground.

def circle (x, y, r, c, level = 0):
 c._param_colour ("fourth parameter to box is expected to be a c
 if level == 0:
 id = pgeif.circle (x, y, r, c._get_pgeif_colour ())
 ob = object (circle_t, id, c, level)
 _register (id, ob)
 else:
 ob = object (fb_circle_t, [x, y, r, c._get_pgeif_colour ()]
 _add (ob, level)
 return ob
```

## Managing foreground/background and the physics engine

- notice the calls to `_register` and `_add`
- the function `_add` adds the object to the appropriate background/foreground level

## Managing foreground/background and the physics engine

```
#
_add - adds an object at foreground/background, level.
A level value of > 0 will be placed into the foreground.
A level value of < 0 will be placed into the background.

def _add (ob, level):
 global foreground, background, levels

 if level > 0:
 if not (level in foreground):
 foreground += [level]
 foreground.sort ()
 else:
 if not (level in background):
 background += [level]
 background.sort ()
 if levels.has_key (level):
 levels[level] += [ob]
 else:
 levels[level] = [ob]
```

## Managing foreground/background and the physics engine

- here there are three main data structures
- foreground and background are lists which contain integer values of active levels
  - the integers are sorted in order
- the dictionary `levels` which uses an integer key to lookup a list of objects

## draw\_foreground

```

_draw_foreground - draws all the foreground objects in

def _draw_foreground ():
 if foreground != []:
 for l in foreground:
 for o in levels[l]:
 o._draw ()
```

## Conclusion

- during this lecture we have started to look at the internals of PGE by concentrating on the Python module `pge.py`
- we have seen how basic objects are created and how parameters are checked and how foreground/background of objects are maintained