Pyglet

Approximately what you can learn in an evening

Neil Muller

31st August 2013

Introducing Pyglet

- “a cross-platform windowing and multimedia library for Python.”

Introducing Pyglet

- “a cross-platform windowing and multimedia library for Python.”
  - i.e. The other game framework for python
- Fewer dependancies than pygame, and better bundled installation

Introducing Pyglet

- “a cross-platform windowing and multimedia library for Python.”
  - i.e. The other game framework for python
- Fewer dependancies than pygame, and better bundled installation
  - Minimal dependencies on Windows and MacOS, which is a great plus
  - Python obviously
  - Only AVBin as an additional dependency, and that’s optional

Introducing Pyglet

- “a cross-platform windowing and multimedia library for Python.”
  - i.e. The other game framework for python
- Fewer dependancies than pygame, and better bundled installation
  - Minimal dependencies on Windows and MacOS, which is a great plus
  - Python obviously
  - Only AVBin as an additional dependency, and that's optional
  - Rather more dependancies on Linux (OpenGL libraries, etc), but they're usually available anyway

Some Examples

- Hello World and friends

The important differences from pygame

- Graphics based entirely on OpenGL
  - Surface is always an OpenGL context
  - All the underlying graphics operations are OpenGL

The important differences from pygame

- Graphics based entirely on OpenGL
  - Surface is always an OpenGL context
  - All the underlying graphics operations are OpenGL
  - Not a higher level wrapper than PyOpenGL, though

The important differences from pygame

- Graphics based entirely on OpenGL
  - Surface is always an OpenGL context
  - All the underlying graphics operations are OpenGL
  - Not a higher level wrapper than PyOpenGL, though
- Less clunky event loop customisation that pygame
- Supports a significantly wider variety of sound and video formats by using AVBin

The important differences from pygame

- Graphics based entirely on OpenGL
  - Surface is always an OpenGL context
  - All the underlying graphics operations are OpenGL
  - Not a higher level wrapper than PyOpenGL, though
- Less clunky event loop customisation that pygame
- Supports a significantly wider variety of sound and video formats by using AVBin
  - Breaks in much more interesting ways if it uses AVBin

The important differences from pygame

- Graphics based entirely on OpenGL
  - Surface is always an OpenGL context
  - All the underlying graphics operations are OpenGL
  - Not a higher level wrapper than PyOpenGL, though
- Less clunky event loop customisation that pygame
- Supports a significantly wider variety of sound and video formats by using AVBin
  - Breaks in much more interesting ways if it uses AVBin
- Nifty Procedural sound module

The important differences from pygame

- Graphics based entirely on OpenGL
  - Surface is always an OpenGL context
  - All the underlying graphics operations are OpenGL
  - Not a higher level wrapper than PyOpenGL, though
- Less clunky event loop customisation that pygame
- Supports a significantly wider variety of sound and video formats by using AVBin
  - Breaks in much more interesting ways if it uses AVBin
- Nifty Procedural sound module
- Less game-focussed than pygame
  - No equivalent for several pygame features (sprite collisions, etc).
  - Better starting point for widget work (provides labels and various text management utilities for the low level stuff)

The important differences from pygame

- Graphics based entirely on OpenGL
  - Surface is always an OpenGL context
  - All the underlying graphics operations are OpenGL
  - Not a higher level wrapper than PyOpenGL, though
- Less clunky event loop customisation that pygame
- Supports a significantly wider variety of sound and video formats by using AVBin
  - Breaks in much more interesting ways if it uses AVBin
- Nifty Procedural sound module
- Less game-focussed than pygame
  - No equivalent for several pygame features (sprite collisions, etc).
  - Better starting point for widget work (provides labels and various text management utilities for the low level stuff)
  - Widgets are still entirely roll your own, though

Resource Handling

- Resources loaded from a specified search path
  - Filenames are checked for case, even on case-insenstive file systems
  - Resources specified only by filename, with a first found wins policy

Resource Handling

- Resources loaded from a specified search path
  - Filenames are checked for case, even on case-insenstive file systems
  - Resources specified only by filename, with a first found wins policy

  - Can search zip files and other funky tricks like that, for ease of distribution
  - Can load images, sounds, fonts, arbitary files

Resource Handling

- Resources loaded from a specified search path
  - Filenames are checked for case, even on case-insenstive file systems
  - Resources specified only by filename, with a first found wins policy

  - Can search zip files and other funky tricks like that, for ease of distribution
  - Can load images, sounds, fonts, arbitary files

  - Also useful text options - plan text & html with formatting
    - Returns these as Document objects for ease of manipulation

Resource Handling

- Resources loaded from a specified search path
  - Filenames are checked for case, even on case-insenstive file systems
  - Resources specified only by filename, with a first found wins policy

  - Can search zip files and other funky tricks like that, for ease of distribution
  - Can load images, sounds, fonts, arbitary files

  - Also useful text options - plan text & html with formatting
    - Returns these as Document objects for ease of manipulation
- Search path is relative to the file being run

Resource Handling

- Resources loaded from a specified search path
  - Filenames are checked for case, even on case-insenstive file systems
  - Resources specified only by filename, with a first found wins policy

  - Can search zip files and other funky tricks like that, for ease of distribution
  - Can load images, sounds, fonts, arbitary files

  - Also useful text options - plan text & html with formatting
    - Returns these as Document objects for ease of manipulation
- Search path is relative to the file being run
  - Except when it sometimes isn't, when it will be relative to the module's __init__.py

Resource Handling

- Resources loaded from a specified search path
    - Filenames are checked for case, even on case-insenstive file systems
    - Resources specified only by filename, with a first found wins policy
    - Can search zip files and other funky tricks like that, for ease of distribution
    - Can load images, sounds, fonts, arbitary files
    - Also useful text options - plan text & html with formatting
        - Returns these as Document objects for ease of manipulation
- Search path is relative to the file being run
    - Except when it sometimes isn't, when it will be relative to the module's __init__.py
- Surprisingly easy to get wrong, as often found when judging pyweek games

Event Handlers

- Usual list of events watched
  - Windows for instance have on_resize, on_draw, on_key_press, on_mouse_motion, on_mouse_press, on_mouse_leave, on_move, etc.
  - EventLoop has on_enter, on_exit, etc.
  - Also timed events, and so forth
- Also a default idle handler, which is responsible for animations, timed events, etc.

Event Handlers

- Usual list of events watched
  - Windows for instance have on_resize, on_draw, on_key_press, on_mouse_motion, on_mouse_press, on_mouse_leave, on_move, etc.
  - EventLoop has on_enter, on_exit, etc.
  - Also timed events, and so forth
- Also a default idle handler, which is responsible for animations, timed events, etc.
- Can easily chain event handlers
  - Event handlers processed in a stack - going top down
    - returning True from an event handler short circuits processing, as expected
  - Since handlers are a stack, easy to remove handlers as well (pop_handler)

Event Handlers

- Usual list of events watched
  - Windows for instance have on_resize, on_draw, on_key_press, on_mouse_motion, on_mouse_press, on_mouse_leave, on_move, etc.
  - EventLoop has on_enter, on_exit, etc.
  - Also timed events, and so forth
- Also a default idle handler, which is responsible for animations, timed events, etc.
- Can easily chain event handlers
  - Event handlers processed in a stack - going top down
    - returning True from an event handler short circuits processing, as expected
  - Since handlers are a stack, easy to remove handlers as well (pop_handler)
- Easy to register custom events
  - Subclass EventDispatcher and register the event

Sprites

- Significantly simpler than pygame sprites
- Mainly a useful way of reusing the same image for multiple objects on screen
  - Several features to help with efficient rendering

Sprites

- Significantly simpler than pygame sprites
- Mainly a useful way of reusing the same image for multiple objects on screen
  - Several features to help with efficient rendering
- Can also use groups for ordering effects

Other notes

- Python 3 port in testing stages (pyglet 1.2alpha has support)
    - recent work has been slow, though

Other notes

- Python 3 port in testing stages (pyglet 1.2alpha has support)
    - recent work has been slow, though
- Support for GLSL still a work in progress
    - Currently doable, but support classes and functions are a bit lacking
        - Consequently, it's all a bit fiddly to get working
    - Various 3rd party modules around to help, though