PyGame Overview
(A Tutorial with digressions)

A Talk by Neil Muller

23rd June 2007

About PyGame

- Wrapper around SDL
- Provides Graphics, sound, font, input handling, provided necessary SDL libraries are available
- Quite popular
  - pygame.org lists well over 100 games
- Used for pyweek games
- LGPL'd license
- Active development community

The Basics (Graphics)

- Hello world

The Basics (Graphics)

- Hello world
- 1st rule of pygame graphics: Everything is a surface

The Basics (Graphics)

- Hello world
- 1st rule of pygame graphics: Everything is a surface
- 2nd rule of pygame graphics: Everything is rectangular

The Basics (Graphics)

- Hello world
- 1st rule of pygame graphics: Everything is a surface
- 2nd rule of pygame graphics: Everything is rectangular
- Managing rectangles is a pain – Sprites and Sprite Groups FTW

The Basics (Input Loop)

- Default is `pygame.event.poll()`

The Basics (Input Loop)

- Default is `pygame.event.poll()`
  - All user interaction creates events
  - Events need to be pulled out of queue and handled

The Basics (Input Loop)

- Default is `pygame.event.poll()`
  - All user interaction creates events
  - Events need to be pulled out of queue and handled
  - Can receive multiple events per clock tick
  - Mouse positions can be discrete

The Basics (Input Loop)

- Default is `pygame.event.poll()`
  - All user interaction creates events
  - Events need to be pulled out of queue and handled
  - Can receive multiple events per clock tick
  - Mouse positions can be discrete
- Can directly query current status

The Basics (Input Loop)

- Default is `pygame.event.poll()`
  - All user interaction creates events
  - Events need to be pulled out of queue and handled
  - Can receive multiple events per clock tick
  - Mouse positions can be discrete
- Can directly query current status
  - Problematic with a slow frame-rate
  - Need to query all events of interest
  - Still need to interact with event system – needed so pygame will repaint, etc.

The Basics (Input Loop)

- Default is `pygame.event.poll()`
  - All user interaction creates events
  - Events need to be pulled out of queue and handled
  - Can receive multiple events per clock tick
  - Mouse positions can be discrete
- Can directly query current status
  - Problematic with a slow frame-rate
  - Need to query all events of interest
  - Still need to interact with event system – needed so pygame will repaint, etc.
- Other ways of interacting with the event queue – `peek` to look ahead, `get` to select event type of interest, `post` to add your on events

The Basics (Input Loop)

- Default is `pygame.event.poll()`
  - All user interaction creates events
  - Events need to be pulled out of queue and handled
  - Can receive multiple events per clock tick
  - Mouse positions can be discrete
- Can directly query current status
  - Problematic with a slow frame-rate
  - Need to query all events of interest
  - Still need to interact with event system – needed so pygame will repaint, etc.
- Other ways of interacting with the event queue – `peek` to look ahead, `get` to select event type of interest, `post` to add your on events
- Can filter events of interest with `set_allowed`, `set_blocked`

Audio Basics

- Sound system consists of mixer, Channel Objects and Sound Objects
- usually only worried about the Sound Object
- WAV only in guaranteed format, but MOD, OGG and MP3 available if SDL_mixer supports them.

Audio Basics

- Sound system consists of mixer, Channel Objects and Sound Objects
- usually only worried about the Sound Object
- WAV only in guaranteed format, but MOD, OGG and MP3 available if SDL_mixer supports them.
- sound playback occurs in background thread
- Channels allow finer control effects -> looping, etc.
  - Can queue up multiple sounds on a Channel
  - Channels can generate events on sound end

Audio Basics

- Sound system consists of mixer, Channel Objects and Sound Objects
- usually only worried about the Sound Object
- WAV only in guaranteed format, but MOD, OGG and MP3 available if SDL_mixer supports them.
- sound playback occurs in background thread
- Channels allow finer control effects -> looping, etc.
  - Can queue up multiple sounds on a Channel
  - Channels can generate events on sound end
- Multiple sounds get mixed together transparently. Each sound can be played by multiple channels
- Sounds get resampled to mixer on load -> Need to reload Sounds if you re-init the mixer with different settings

Audio Basics(cont)

- `pre_init` allows passing settings to mixer if using `pygame.init()`

Audio Basics(cont)

- `pre_init` allows passing settings to mixer if using `pygame.init()`
- Special music object for background music
  - Doesn't load entire sound at once
  - Can only be one music object at a time

Misc other stuff

- `pygame.joystick` and `pygame.event.JOY*` allow you to interface with joysticks
- `pygame.movie`: allows playback of MPEG files much like pygame.mixer
  - Can't play while pygame.mixer is initialised
- `pygame.cdrom` and CD objects for manipulating CDs
- `pygame.color` allows color lookup and some manipulation
- `pygame.cursor`: manipulated the mouse cursor

More about pygame.display

- Can ask for specific modes, but pygame will emulate things that are not available
  - Colour depths
  - special surface types: FULLSCREEN, OPENGL surfaces for pyopengl, etc.

More about pygame.display

- Can ask for specific modes, but pygame will emulate things that are not available
  - Colour depths
  - special surface types: FULLSCREEN, OPENGL surfaces for pyopengl, etc.
- mode_ok allows checking if modes are available

More about pygame.display

- Can ask for specific modes, but pygame will emulate things that are not available
  - Colour depths
  - special surface types: FULLSCREEN, OPENGL surfaces for pyopengl, etc.
- mode_ok allows checking if modes are available
- `pygame.display.iconify`, `pygame.display.set_icon`, `pygame.toggle_fullscreen`

More about pygame.display

- Can ask for specific modes, but pygame will emulate things that are not available
  - Colour depths
  - special surface types: FULLSCREEN, OPENGL surfaces for pyopengl, etc.
- mode_ok allows checking if modes are available
- `pygame.display.iconify`, `pygame.display.set_icon`, `pygame.toggle_fullscreen`
- Double buffering $\Longrightarrow$ mainly useful for pyopengl

More about pygame.display

- Can ask for specific modes, but pygame will emulate things that are not available
  - Colour depths
  - special surface types: FULLSCREEN, OPENGL surfaces for pyopengl, etc.
- mode_ok allows checking if modes are available
- `pygame.display.iconify`, `pygame.display.set_icon`, `pygame.toggle_fullscreen`
- Double buffering $\Longrightarrow$ mainly useful for pyopengl
- `pygame.set_gamma` $\Longrightarrow$ simplistic gamma control, `pygame.set_gamma_ramp` -> fairly fine grained control (256 element sequences for each colour band)
  - Not all modes support gamma though

Sprites and More Sprites

- Fundamental Objects: sprites and Groups

Sprites and More Sprites

- Fundamental Objects: sprites and Groups
  - typically, sprites $\equiv$ game sobjects
  - groups $\equiv$ object properties

Sprites and More Sprites

- Fundamental Objects: sprites and Groups
  - typically, sprites $\equiv$ game sobjects
  - groups $\equiv$ object properties
- Group actions (adding, deleting, checking emmebership) quick

Sprites and More Sprites

- Fundamental Objects: sprites and Groups
  - typically, sprites $\equiv$ game sobjects
  - groups $\equiv$ object properties
- Group actions (adding, deleting, checking emmebership) quick
- Can manage group memebership either rom group methods or sprite methods

Sprites and More Sprites

- Fundamental Objects: sprites and Groups
  - typically, sprites $\equiv$ game sobjects
  - groups $\equiv$ object properties
- Group actions (adding, deleting, checking emmebership) quick
- Can manage group memebership either rom group methods or sprite methods
- Groups are sequences $\Longrightarrow$iteration, len, etc. all possible

More Surface stuff

- Can happily scribble on Surfaces
  - pygame.draw provides lines, arcs, ellipses and such
  - rectangles can be obtained by pygame.draw.rect

More Surface stuff

- Can happily scribble on Surfaces
  - pygame.draw provides lines, arcs, ellipses and such
  - rectangles can be obtained by pygame.draw.rect
  - draw.rect will draw filled rectangles when asked
  - but Surface.fill() should always be as fast or faster

More Surface stuff

- Can happily scribble on Surfaces
  - pygame.draw provides lines, arcs, ellipses and such
  - rectangles can be obtained by pygame.draw.rect
  - draw.rect will draw filled rectangles when asked
  - but Surface.fill() should always be as fast or faster
  - Can query individual pixels (`get_at` and `set_at`)

More Surface stuff

- Can happily scribble on Surfaces
  - pygame.draw provides lines, arcs, ellipses and such
  - rectangles can be obtained by pygame.draw.rect
  - draw.rect will draw filled rectangles when asked
  - but Surface.fill() should always be as fast or faster
  - Can query individual pixels (`get_at` and `set_at`)
- Transform operations:
  - Scale, rotate, chop, flip
  - Unusual transforms: rotozoom, scale2x
  - rotozoom is known to cause artifacts in some circumstances

More Surface stuff

- Can happily scribble on Surfaces
  - pygame.draw provides lines, arcs, ellipses and such
  - rectangles can be obtained by pygame.draw.rect
  - draw.rect will draw filled rectangles when asked
  - but Surface.fill() should always be as fast or faster
  - Can query individual pixels (`get_at` and `set_at`)
- Transform operations:
  - Scale, rotate, chop, flip
  - Unusual transforms: rotozoom, scale2x
  - rotozoom is known to cause artifacts in some circumstances
- Surfaces need to be locked for several operations
  - All functions will lock if needed

More Surface stuff

- Can happily scribble on Surfaces
  - pygame.draw provides lines, arcs, ellipses and such
  - rectangles can be obtained by pygame.draw.rect
  - draw.rect will draw filled rectangles when asked
  - but Surface.fill() should always be as fast or faster
  - Can query individual pixels (`get_at` and `set_at`)
- Transform operations:
  - Scale, rotate, chop, flip
  - Unusual transforms: rotozoom, scale2x
  - rotozoom is known to cause artifacts in some circumstances
- Surfaces need to be locked for several operations
  - All functions will lock if needed
  - But explicit locks possible, and a good idea if doing lots of stuff that needs the Surfed locked

More Surface stuff

- Can happily scribble on Surfaces
  - pygame.draw provides lines, arcs, ellipses and such
  - rectangles can be obtained by pygame.draw.rect
  - draw.rect will draw filled rectangles when asked
  - but Surface.fill() should always be as fast or faster
  - Can query individual pixels (`get_at` and `set_at`)
- Transform operations:
  - Scale, rotate, chop, flip
  - Unusual transforms: rotozoom, scale2x
  - rotozoom is known to cause artifacts in some circumstances
- Surfaces need to be locked for several operations
  - All functions will lock if needed
  - But explicit locks possible, and a good idea if doing lots of stuff that needs the Surfed locked
  - fogetting to unlock the surface is not recommended, though

Interacting with PIL

Who didn't see this coming?

- `pygame.image` only guarantees support for BMP - other formats dependant on SDL compilation options
- `pygame.image` manipulation options quite limited

Interacting with PIL

Who didn't see this coming?

- `pygame.image` only guarantees support for BMP - other formats dependant on SDL compilation options
- `pygame.image` manipulation options quite limited
- pgame Surfaces support `fromstring` and `tostring`, which can be used to interface with PIL

Interacting with PIL

Who didn't see this coming?

- `pygame.image` only guarantees support for BMP - other formats dependant on SDL compilation options
- `pygame.image` manipulation options quite limited
- pgame Surfaces support `fromstring` and `tostring`, which can be used to interface with PIL
- Need to be careful about formats, since that is lost in the conversion

Interacting with Numpy

Seriously, who didn't see this coming?

- `pygame.surfarray`: Gives a array representation of a pygame Surface

Interacting with Numpy

Seriously, who didn't see this coming?

- `pygame.surfarray`: Gives a array representation of a pygame Surface
  - Still based off numeric (darn)

Interacting with Numpy

Seriously, who didn't see this coming?

- `pygame.surfarray`: Gives a array representation of a pygame Surface
  - Still based off numeric (darn)
  - Fortunately casting from numeric array to numpy array trivial
  - Going back is the annoyance, but also trivial
  - Patches for pygame to support numpy directly exist, still need to be applied

Interacting with Numpy

Seriously, who didn't see this coming?

- `pygame.surfarray`: Gives a array representation of a pygame Surface
  - Still based off numeric (darn)
  - Fortunately casting from numeric array to numpy array trivial
  - Going back is the annoyance, but also trivial
  - Patches for pygame to support numpy directly exist, still need to be applied
- `pygame.sndarray`: array representation of a pygame.Sound
  - pygame.sndarry.array(Sound) copies sound to array, pygame.sndarray.make_sound(Array) copies array to a Sound object
  - 1-D for mono, 2-D for stereo
  - Probably most useful for filtering
  - Need to be careful about casting to right format before feeding sndarray (int16 typically, occasionally int8)

Interactions with Threads

- SDL Limitation -> process events and update display in a single thread

Interactions with Threads

- SDL Limitation -> process events and update display in a single thread
- There are some occasionally weird interactions with pygame.time.Clock's - clock's in different threads is often a bad idea.

Interactions with Threads

- SDL Limitation -> process events and update display in a single thread
- There are some occasionally weird interactions with pygame.time.Clock's - clock's in different threads is often a bad idea.
- pygame.fastevent -> pgame.event replacement
  - Meant to be faster in multi-threaded cases
  - Doesn't suffer fixed size event queue issues of standard event queue
  - Not very well documented, and reports of stability and speed rather varied

In Conclusion

- I like it