Pixel Toy

Functions

Drawing

```
drawRectangle(x, y, width, height [, rotation])
drawLine(x1, y1, x2, y2)
drawPoint(x, y)
drawString(x, y, string [, rotation] [, size] [, centered])
drawCircle(x, y, radius [, rotation])
drawEllipse(x, y, radiusX, radiusY [, rotation])
drawPolygon(point1x, point1y, point2x, point2y, ... [, rotation = x])
drawRegularPolygon(x, y, order, size, [, rotation])
```

```
setWireFrame( boolean )
```

```
loadImage(src, smooth = True, animatedImageCountX = 1)
```

Loads an image from the specified location on disk.

Optional arguments:

smooth: smooth the image when drawing. True by default

animatedImageCountX:

Number of animated image frames the image file contains. 1 by default.

Example:

```
image = loadImage('res/man1.png')
```

Be sure to load images before starting the main loop, otherwise you're loading the same image 60 times per second!

```
drawImage(image, x, y, width, height)
```

Draws an image that was loaded by loadImage. The remaining parameters are used similar to the drawRectangle() function.

```
image.rotate(rotationDegrees)
```

Rotates the image by the specified amount. Next time the image is drawn it is rotated by the new rotation.

```
image.setRotation(rotationDegrees)
```

Sets the rotation of the image

```
image.draw()
```

Same effect as drawImage.

```
image.nextAnimationFrame()
```

Shows the next frame of the animated image

```
image.resetAnimation()
```

Start the image animation over from the start.

```
useColour(r, g, b [, a])
```

After calling this function, anything you draw will use this colour.
Until you update the colour again, of course.
The alpha component is optional.

```
setBackgroundColour(r, g, b)
```

Set the background colour

`newFrame()`

Call this function when you are ready to draw a new frame.

`GL Matrix interface:`

Everything drawn between a GLpush call and a GLpop call will be translated/rotated/scaled depending on which GL function you call.

Example:

```
GLpush()
GLtranslate(10, 0)
drawCircle(10, 10, 5)  #circle will actually be drawn at (20, 10)
GLpop()
```

`GLpush()` Pushes a matrix

`GLpop()` Pops a matrix

`GLtranslate( dx, dy )` Translates everything drawn by (dx, dy)

`GLrotate( angle )` Rotates everything drawn around the bottom left corner

`GLscale( sx, sy )` Scales everything drawn by (sx, sy)

<u>Audio</u>

`setListenerPosition(x, y, z)`

Sets the Listeners position

`setListenerVelocity(x, y, z)`

Sets the Listeners velocity

`setListenerOrientation(x, y, z, upaxis = "z")`

Sets the Listeners orientation.

Parameters:

- x, y, z: The forward vector
- upaxis: can be “x”, “y” or “z”. Determines which axis is the up vector

`rewindAllSources()` rewinds all sources.

`stopAllSources()` stops all sources.

`pauseAllSources()` pauses all sources.

`setVolume(volume)`

Sets the master volume, can go up to infinity but not advised to go above 15.

`Source( path, type = “static”, loop = False )`

Returns a new Source object. Can play either .ogg files or .wav files.

Optional arguments:

- type: Can be either “static” or “stream”
  - “static”: The entire sound file with be loaded on the disk
  - “stream”: Chunks of the sound file will be loaded bit by bit while playing (good for music files)
- loop: whether or not the source should loop

Examples:

```
source = Source('res/heal.ogg') #source will not loop or stream
source = Source('res/heal.ogg', True)  # source will loop but not stream
source = Source('res/heal.ogg', “stream”)  # source will stream the ogg file
source = Source('res/heal.ogg', “stream”, True)  # source will stream and loop
```

Be sure to load sources before starting the main loop, otherwise you're loading a source 60 times per second and openAL will crap itself!

`source.setPitch(pitch)`
Sets the pitch of the source

`source.setVolume(volume)`
Sets the volume of the source (volume will be clamped between 0 and 1)

`source.setPosition(x, y, z)`
Sets the position of the source

`source.setDirection(x, y, z)`
Sets the direction of the source

`source.setVelocity(x, y, z)`
Sets the velocity of the source

`source.setLooping(enabled)`
Sets whether or not the source should loop

`source.setRelative(enabled)`
Sets whether or not the source's position is relative to the listene

`source.setCone(innerAngle, outerAngle, outerVolume)`
Sets a directional volume cone for the source. Combined with setDirection the cone angles allow for the sources's volume to vary depending on the sources direction.
Parameters:
- innerAngle: The angle from the source's direction where if the listener is within the cone created by the angle, it hears the sound at normal volume.
- outerAngle: If the listener is between the cones defined by this angle and innerAngle, it will hear the sound with a volume between the normal volume and the outerVolume.
- outerVolume: The source's volume when the listener is outside both the inner and outer cone angles.

`source.play()` Plays the source.
`source.stop()` Stops the source and rewinds it.
`source.pause()` Pauses the source.
`source.rewind()` Makes the source start at the beginning.

`The Getters:` Returns whatever their description says
`source.isLooping(), source.isPlaying(), source.isStopped(), source.isPaused(), source.isRelative(), source.getVolume(), source.getPitch(), source.getPosition(), source.getVelocity(), source.getDirection(), source.getCone()`

Input

`isLeftMouseDown()`

Returns True if the left mouse button is pressed. False if it is not.

`isRightMouseDown()`

Returns True if the right mouse button is pressed. False if it is not.

`getMouseWheelDelta()`

Returns the number of steps that the mouse wheel has been moved up since the previous call to this function.

`getTime()`

Returns a counted time in seconds with no specific start time, use it only for telling the difference between two different times.

`isKeyDown(key)`

Returns True or False depending on whether the specified key is pressed.
Example: check if the 'e' key is pressed:

```
if isKeyDown('e'):
    print 'e'
```

Here's a list of all possible keys:

Letters: **`a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z`**

Numbers: **`0, 1, 2, 3, 4, 5, 6, 7, 8, 9`**

Numpad: **`numpad1, numpad2, numpad3, numpad4, numpad5, numpad6, numpad7, numpad8, numpad9, numpad0, numpadequals, numpadenter, numpadcomma, decimal, divide, multiply, subtract, add`**

Function keys: **`f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16, f17, f18, f19`**

Punctuation: **`colon, comma, period, slash, semicolon, space, apostrophe, grave, underline, lbracket, rbracket, circumflex, backslash, tab, minus, equals`**

Editing keys: **`insert, delete, rshift, lshift, rcontrol, lcontrol, backspace, capslock, numlock, scroll, ralt, lalt`**

Navigation keys: **`up, left, right, down, end, home, escape, enter, pageup, pagedown`**

System keys: **`clear, sysrq, function, pause, stop, section, rwindows, lwindows`**

`random()`

Returns a random value between 0.0 and 1.0

`quit()`

Exit the program immediately

Variables

These variables contain some utility values. They are updated every time you call the newFrame() function.

`_mouseX`	The x component of the mouse position on the screen
`_mouseY`	The y component of the mouse position on the screen
`_screenWidth`	The width of the screen
`_screenHeight`	The height of the screen
`_deltaTime`	Time in seconds since last call of newFrame()