Lecture: 16-1

- Prerequisites for this lecture are: 15-1, 15-2 and 15-3.

Implementation of Touchmap

- these notes will show the structure of `touchmap.py`
- they also will describe `touchgui.select`
- they will show you how extend touchmap
  - creating an `export` function
  - create a `worldspawn` entity
- also show you how to add your own graphics into the library

Implementation of Touchmap

- touchmap is implemented in a single file
- uses a similar structure to the demo programs in `touchgui`

Implementation of Touchmap

touchmap-0.1/touchmap.py

```
def main ():
    global players, grid, cell_size

    pygame.init ()
    if full_screen:
        gameDisplay = pygame.display.set_mode ((display_width, \
                                                display_height), FULLSCREEN)
    else:
        gameDisplay = pygame.display.set_mode ((display_width, display_height))

    touchgui.set_display (gameDisplay, display_width, display_height)
    controls = buttons ()

    gameDisplay.fill (touchguipalate.black)
    while True:
        grid = button_grid (cell_size)
        forms = controls + grid
        touchgui.select (forms, event_test, finished)

main ()
```

touchgui.select

- **touchgui/touchgui.py**

```
def select (forms, event_test, finished = None, timeout = -1):
    if timeout == -1:
        _blocking_select (forms, event_test, finished)
    else:
        _nonblocking_select (forms, event_test, finished, timeout)
```

- two optional parameters: `finished` and `timeout`
- if `timeout` is absent then it calls a blocking version of select
  - in which the process will block until an event occurs
  - this is efficient, but forces the main program to be entirely event based
    - furthermore all events must go through the `touchgui/pygame` event queue

touchgui.select

- sometimes you might want to write programs which use a mixture of event based and some polling

- for example the cluedo server program
  - tests the gui briefly and then checks the network stack for input and rotates icons
    - ideally it would be good to join the network stack to the pygame input event queue and timers
    - in practice this is hard to configure, and `touchgui.select` allows a pragmatic (less efficient) solution
    - can *poll* both

Cluedo server example code

```
    offset = 0
    while not selection_complete:
        s, rpc = getRPC (s)
        processRPC (s, rpc)
        playerIcons = positionIcons (players, [.5, .5], .2, offset)
        forms = playerIcons + playerIconsStatic
        gameDisplay.fill (palate.black)
        touchgui.select (forms, event_test, selected, 10)
        offset = (offset + 1) % 360
    return players
```

touchgui.select

- redraws all tiles in forms.
- `finished` is polled to see if the function should return
  - `finished` is a parameterless function which returns `True` or `False`
- `timeout` is the maximum no. of milliseconds the function can poll.
  - `timeout` is optional and defaults to -1 if absent
- `finished` is also optional

touchgui.select

- Pre-condition
  - forms is a list of tiles.
  - `event_test`is which has a single parameter `(event)`
  - `event_test` does not return a value

- Post-condition: None.