John Romero Programming Proverbs

- 5. “We are our own best testing team and should never allow anyone else to experience bugs or see the game crash. Don’t waste others’ time. Test thoroughly before checking in your code.”

- John Romero, “The Early Days of Id Software - John Romero @ WeAreDevelopers Conference 2017”

Implementation of Touchmap 0.2

- last week we adpated `touchmap-0.1` to use an indirect function call to implement a spawn button
  - this worked and could be extended to implement many other buttons

- however it is an ugly implementation
  - although it was very minimal (in terms of code line changes)

- it would be cleaner to adopt an object oriented approach
  - albeit the number of lines of code will increase

Implementation of Touchmap 0.2

- one of the major issues of a `touchmap` implementation is the ability to change a tile in the grid
  - currently it changes:
    - a blank to a wall
    - a wall to a door
    - a door to a blank

- we need a spawn, hellknight, ammo, ticks etc
  - the indirect function call can do this but it will result in messy code

Touchmap 0.2

- removes the indirect function call
  - but introduces a new class `button`
  - and about 20 lines of extra code

Touchmap 0.2

- `touchmap-0.2` can be downloaded using:

- ```
  $ cd
  $ cd Sandpit
  $ wget http://floppsie.comp.glam.ac.uk/download/targz/touchmap-0.2.tar.gz
  $ tar zxf touchmap-0.2.tar.gz
  $ ls touchmap-0.2
  ```
  ```

Touchmap 0.2

- and can be built using:

```
$ cd
$ cd Sandpit
$ rm -rf build-touchmap
$ mkdir build-touchmap
$ cd build-touchmap
$ ../touchmap-0.2/configure
$ make
```

- and you can run it via:

```
$ cd
$ cd Sandpit/build-touchmap
$ ./localrun.sh touchmap.py
```

Touchmap 0.2

Sandpit/touchmap-0.2/touchmap.py

```
class button:
    def __init__ (self, x, y, size):
        self._x = x
        self._y = y
        self._size = size
        self._tile = touchgui.image_tile (blank_list ("wallv", size),
                                           x, y,
                                           size, size, cellback)
    def to_blank (self):
        self._tile.set_images (blank_list ("wallv", cell_size))
    def to_wall (self):
        self._tile.set_images (wall_list ("v", cell_size))
    def to_spawn (self):
        self._tile = touchgui.text_tile (dark_grey, light_grey, white, mid_grey,
                                         's', self._size,
                                         self._x, self._y,
                                         self._size, self._size,
                                         worldspawn, "worldspawn")
    def get_tile (self):
        return self._tile
```

Touchmap 0.2

- **Sandpit/touchmap-0.2/touchmap.py**

```
blank_t, wall_t, door_t, spawn_t = range (4)  # enumerated types
next_tile = wall_t

function_create = {blank_t:create_blank,
                   wall_t:create_wall,
                   door_t:create_door,
                   spawn_t:create_spawn}
```

- note the `next_tile` is now a variable which contains an enumerated value
  - denoting the cell type to be created on the grid

- notice the dictionary of enumerated values associated with functions

Touchmap 0.2

Sandpit/touchmap-0.2/touchmap.py

```
def cellback (param, tap):
    global clicked, cell_array, button_array, double_tapped_cell
    clicked = True
    mouse = pygame.mouse.get_pos ()
    x, y = get_cell (mouse)
    old = cell_array.get (x + xoffset, y + yoffset)
    button = button_array.get (x + xoffset, y + yoffset)
    if old == " ":
        # blank -> next_tile
        function_create[next_tile] (button)
        cell_array.set_contents (x + xoffset, y + yoffset, "v")
    elif old == "v":
        # wall -> door
        button.to_door ()
        cell_array.set_contents (x + xoffset, y + yoffset, "|")
    elif old == "|":
        # door -> blank
        button.to_blank ()
        cell_array.set_contents (x + xoffset, y + yoffset, " ")
```

Touchmap 0.2

Sandpit/touchmap-0.2/touchmap.py

```
def create_blank (button):
    button.to_blank ()

def create_wall (button):
    button.to_wall ()

def create_door (button):
    button.to_door ()

def create_spawn (button):
    global next_tile
    button.to_spawn ()
    next_tile = wall_t
```

Hints on how to implement the header for the output map

- we need a mechanism to allow the program to remember which assets have been used
- then just prior to writing the file the program needs to iterate over the list and emit the appropriate header info

Tutorial

- download the `touchmap-0.2` code
- comment each function/method in `touchmap-0.2/touchmap.py`
- run the code
  - notice the colour scheme of the spawn button
  - change the colour of the spawn button so the default screen background blends in with the background of the button
- change the `return` button to an `export` button

Tutorial

- see if you can add a hellknight button
- fix the spawn button to that it adds an `s` when the `export` button is pressed
- see if you can find a way to add the correct header to the exported text

Tutorial

- ie, check the example

- **Sandpit/chisel/maps/simple.txt**

```
define 1 room 1
define 2 room 2
define 3 room 3
define s worldspawn
define o monster monster_demon_imp
define n monster monster_demon_hellknight
define i light
define a ammo ammo_shells_large 16
```

- add a light button

- add a room button