John Romero Programming Proverbs

- 7. “Use a development system that is superior to your target.”

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

File export and integrating with chisel part 2

- add a doom3 button which runs `chisel` and `doom3`
- firstly it needs to compile the `.txt` file into a `.map` file
- secondly it needs to dmap tile `.map` into the doom3 format

File export and integrating with chisel part 2

Sandpit/touchmap/touchmap.py

```
def buttons ():
    return [touchgui.image_tile (button_list ("power"),
                                 touchgui.posX (0.95), touchgui.posY (1.0),
                                 100, 100, myquit),
            touchgui.image_tile (button_list ("export"),
                                 touchgui.posX (0.0), touchgui.posY (1.0),
                                 100, 100, myexport),
            touchgui.image_tile (private_list ("doom3"),
                                 touchgui.posX (0.05), touchgui.posY (1.0),
                                 100, 100, mydoom3),
            touchgui.image_tile (button_list ("smaller"),
                                 touchgui.posX (0.0), touchgui.posY (0.10),
                                 100, 100, myzoom, True),
            touchgui.image_tile (button_list ("larger"),
                                 touchgui.posX (0.95), touchgui.posY (0.10),
                                 100, 100, myzoom, False)]
```

File export and integrating with chisel part 2

Sandpit/touchmap/touchmap.py

```
def mydoom3 (param, tap):
    pygame.display.update ()  # flush all graphic changes to screen
    pygame.time.delay (toggle_delay * 2) # pause
    pygame.quit ()            # shutdown pygame
    dmap ()                   # run chisel and dmap doom3 compile
    exec_doom_map ()          # now run doom3
    quit ()                   # quit python
```

Sandpit/touchmap/touchmap.py

```
def dmap ():
    os.system ("d3 +dmap tiny.map +quit")

def exec_doom_map ():
    os.system ("d3 +map tiny.map")
```

File export and integrating with chisel part 2

- maybe you can improve the code so that it checks whether it needs to export the map

- the whole area of check pointing and saving maps has not been addressed
  - you might want to consider this aspect of the touchmap

Adding a monster tick

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

```
blank_t, wall_t, door_t, spawn_t, hell_t, tick_t = range (6)  # enumerated types
```

Adding a monster tick

Sandpit/touchmap/touchmap.py

```
class button:
    ...
    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_door (self):
        self._tile.set_images (door_list ("v", cell_size))
    def to_hell (self):
        self._tile.set_images (private_list ("hellknight"))
    def to_tick (self):
        self._tile.set_images (private_list ("tick"))
    ...
```

Adding a monster tick

Sandpit/touchmap/touchmap.py

```
def tick (name, tap):
    global next_tile
    pygame.display.update ()
    if tap == 1:
        next_tile = tick_t

def assets ():
    return [touchgui.image_tile (private_list ("hellknight"),
                                 touchgui.posX (0.95), touchgui.posY (0.9),
                                 100, 100, hellknight),
            touchgui.image_tile (private_list ("tick"),
                                 touchgui.posX (0.95), touchgui.posY (0.8),
                                 100, 100, tick)]
```

Adding a monster tick

Sandpit/touchmap/touchmap.py

```
def create_tick (button):
    global next_tile
    mouse = pygame.mouse.get_pos ()
    x, y = get_cell (mouse)
    button.to_tick ()
    include_asset ('T', "monster monster_demon_tick")
    cell_array.set_contents (x + xoffset, y + yoffset, "T")
    next_tile = wall_t

function_create = {blank_t:create_blank,
                   wall_t:create_wall,
                   door_t:create_door,
                   spawn_t:create_spawn,
                   hell_t:create_hell,
                   tick_t:create_tick}
```

Adding a monster tick

Sandpit/touchmap/touchmap.py

```
def cellback (param, tap):
    global clicked, cell_array, button_array, double_tapped_cell
    clicked = True
    ...
    elif old == "|":
        # door -> blank
        button.to_blank ()
        cell_array.set_contents (x + xoffset, y + yoffset, " ")
    elif old in ["H", "S", "T"]:
        # remove asset
        button.to_blank ()
        cell_array.set_contents (x + xoffset, y + yoffset, " ")
        exclude_asset (old)
    ...
```

Adding room numbers

- ideally would like the button to remember which room has been allocated
  - touchmap should reuse old deleted room numbers appropriately

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

```
blank_t, wall_t, door_t, spawn_t, hell_t, tick_t, room_t = range (7)
...
rooms_available = []  # any room number which was deleted is placed here
next_room = 1  # the next available room number to be used.
```

Adding a to_room method to the button class

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

```
def to_room (self, room):
   self._tile = touchgui.text_tile (black, light_grey, white, mid_grey,
                                     room, self._size,
                                     self._x, self._y,
                                     self._size, self._size, delroom, "room")
```

- require a specific `delroom` callback to remember the room number for next time a room is created
  - alternatively we could use `cellback`, however `cellback` would become much more complex

delroom

Sandpit/touchmap/touchmap.py

```
def delroom (param, tap):
    global clicked, cell_array, button_array, double_tapped_cell, rooms_available
    clicked = True
    mouse = pygame.mouse.get_pos ()
    x, y = get_cell (mouse)
    button = button_array.get (x + xoffset, y + yoffset)
    button.spawn_to_blank ()
    rooms_available += [cell_array.get (x + xoffset, y + yoffset)]
    cell_array.set_contents (x + xoffset, y + yoffset, " ")
```

myroom

Sandpit/touchmap/touchmap.py

```
def myroom (name, tap):
    global next_tile
    pygame.display.update ()
    if tap == 1:
        next_tile = room_t
```

myroom

Sandpit/touchmap/touchmap.py

```
def glyphs ():
    return [touchgui.text_tile (black, mid_grey, white, light_grey,
                                'S', touchgui.unitY (0.05),
                                touchgui.posX (0.5), touchgui.posY (1.0),
                                100, 100, worldspawn, "worldspawn"),
            touchgui.text_tile (black, mid_grey, white, light_grey,
                                'room', touchgui.unitY (0.05),
                                touchgui.posX (0.45), touchgui.posY (1.0),
                                100, 100, myroom, "room")]
```

Tutorial

- try integrating this code into your touchmap
- you might need to work through the previous weeks tutorial
- do not be afraid to deviate away from the notes or improve on the ideas
- optional extra, if you finish this task see if you can modify touchgui
  - so that it allows a text tile to be created with a background
    - but this button changes the foreground text, rather than the background colour

Tutorial

- consider how the program might save the map every 5 or so changes
  - make this configurable
  - how might you implement a back/forward button which switches between maps?