Lecture: 18-1

- Prerequisites for this lecture are: 17-1, 17-2 and 17-3.

File export and integrating with chisel part 1

- would be good to modify touchmap so that it
  - exports (saves the map file)
  - runs chisel and/or runs doom3

Exporting the file

- would be good to keep a list of assets
  - only write out the assets the map is using
-
```
asset_list = []  # list of assets
asset_desc = {}  # dictionary of asset descriptions
asset_count = {}  # how many of each asset are we using?
```


Asset set functions

-
```
def include_asset (a, desc):
    global asset_list, asset_desc, asset_count
    if not (a in asset_list):
        asset_list += [a]
    asset_desc[a] = desc
    if asset_count.has_key (a):
        asset_count[a] += 1
    else:
        asset_count[a] = 1
```

Asset set functions

```
def exclude_asset (a):
    global asset_list, asset_count
    if asset_count.has_key (a):
        asset_count[a] -= 1
        if asset_count[a] == 0:
            del asset_count[a]
            asset_list.remove (a)
```


Export callback function

```
def myexport (name, tap):
    pygame.display.update ()
    save_map (current_map_name)

def save_map (name):
    f = open (name, "w")
    f = write_assets (f)
    f.write ("\n")  # add blank line for eye candy
    f = write_map (f)
    f.close ()
```


Writing assets to the file

```
def write_asset (f, a):
    s = "define %s %s\n" % (a, asset_desc[a])
    f.write (s)
    return f

def write_assets (f):
    for a in asset_list:
        f = write_asset (f, a)
    return f
```

- notice that the file object `f` is returned, this ensures that future writes occur in order!
  - unambiguous code (will work if an object were passed by value rather than by reference)

Writing assets to the file

```
def write_map (f):
    m = ""
    mdict = {"v":"#", "h":"#", "-":".", "|":".", " ":" ",
    x, y = cell_array.high ()
    for j in range (y):
        for i in range (x):
            m += mdict[cell_array.get (i, j)]
        m += "\n"
    f.write (m)
    return f
```

hellknight callback

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

def assets ():
    return [touchgui.image_tile (private_list ("hellknigh
                                 touchgui.posX (0.95), to
                                 100, 100, hellknight),
            touchgui.image_tile (private_list ("tick"),
                                 touchgui.posX (0.95), to
                                 100, 100, myquit)]
```


Changing cellback

```
def cellback (param, tap):
    global clicked, cell_array, button_array, double_tapp
    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)
```


Changing cellback

```
    if old == " ":
        # blank -> next_tile
        function_create[next_tile] (button)
    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
    elif old in ["H", "S"]:
        # remove asset
        button.to_blank ()
        cell_array.set_contents (x + xoffset, y + yoffset
        exclude_asset (old)
```


Extending class button

```
    def to_hell (self):
        self._tile.set_images (private_list ("hellknight"))
    def to_spawn (self):
        self._tile = touchgui.text_tile (black, light_grey, white,
                                         'S', self._size,
                                         self._x, self._y,
                                         self._size, self._size, de
    def spawn_to_blank (self):
        self._tile = touchgui.image_tile (blank_list ("wallv", self
                                          self._x, self._y,
                                          self._size, self._size, c
```

Reverting a cell to a blank

```
def delspawn (param, tap):
    global clicked, cell_array, button_array, double_tapp
    clicked = True
    mouse = pygame.mouse.get_pos ()
    x, y = get_cell (mouse)
    button = button_array.get (x + xoffset, y + yoffset)
    button.spawn_to_blank ()
```