

# 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

slide 3  
gaius

## 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?
```

slide 4  
gaius

## 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
```

## 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
```

- 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)

## 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 ("hellknight",
        touchgui.posX (0.95), to
        100, 100, hellknight),
        touchgui.image_tile (private_list ("tick"),
        touchgui.posX (0.95), to
        100, 100, myquit)]

```

## Changing callback

```

def callback (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 callback

```

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 ("wall", 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 ()
```

## test.txt created by touchmap

```
define S worldspawn
define H monster monster_demon_hellknight

#####
# S #
#   #
#   #
# H #
#####
```

## Introducing new images into touchmap

- place the new images in  
\$HOME/Sandpit/touchmap-0.2/images
- alter \$HOME/Sandpit/Makefile.am to copy this image into the build directory

\$HOME/Sandpit/Makefile.am

```
all: doorh.png doorv.png doorh-bw.png doorv-bw.png hinge.hellkn.
    wallh.png wallv.png wallh-bw.png wallv-bw.png
hellknight.png: $(srcdir)/images/hellknight.png
    convert -resize 100x100 $< $@
tick.png: $(srcdir)/images/tick.png
    convert -resize 100x100 $< $@
```

## Tutorial

- integrate these changes into your touchmap code
- add a room button, so that the room information is generated, ie:

```
define S worldspawn
define H monster monster_demon_hellknight
define 1 room 1

#####
# S #
#   #
# 1 #
# H #
#####
```

## Tutorial

- your room button should keep a record of the rooms allocated
  - remember which rooms are deleted and reuse the number
  
- add other assets into touchmap
  - lights, tick, ammo, rockets, rocket launcher
  - [Doom3 asset list](https://doom.fandom.com/wiki/Doom_3) ([https://doom.fandom.com/wiki/Doom\\_3](https://doom.fandom.com/wiki/Doom_3))
  
- add a doom3 button which will shutdown touchmap and then run doom3
  - make this button also run chisel to generate the tiny.map before running doom3