**Memory V1 Algorithm**

main program

create window (**U**)

create Game called game, using window (**U**)

play game (U)

close window (L)

function create window (out: window)

initialize graphics library (L)

open window with title '**Memory**', size 500 by 400 (L)

class Game

window

bg color

pause time

close clicked

continue game

**surface**

**tiles**

**image list**

Game function create (in: window, out: game)

set window using argument

create Color object, called bg color, using 'black' (L)

set pause time to value

set close clicked to false

set continue game to true

**set the surface of the Tile obj(U)**

**load question mark image**

**create empty image list**

**for index in range (0, 8)**

**load the image**

**append to image list**

**image\_list = image\_list + image\_list**

**call create\_grid function**

**Game function create grid(in:self out:)**

**tile width = window width / 5**

**tile height = window height / 4**

**for row in range 0 to 4**

**create a empty row list (L)**

**for column in range 0 to 4**

**set x to column index \* tile width**

**set y to row index \* tile height**

**set tile\_type to random int(0, length of image list -1)**

**image = image\_list[tile\_type]**

**image\_list.remove(tile\_type)**

**value = tile\_type % 8**

**tile=Tile(x, y, tile width, tile height, image, value)**

**add tile to row list (L)**

**add row list to tiles list (L)**

Game function play (in: self)

draw self (U)

while not close clicked

play frame (B)

Game block play frame

handle event on self (**U**)

if continue game

update self (**U**)

decide continue on self (**U**)

draw self (**U**)

pause for pause time (L)

Game function handle event (in: self)

get next event (L)

if type of event equals window close

set close clicked to true

~~if type of event equals~~ **~~???~~** ~~and continue game~~

**~~???~~**

Game function draw (in: self)

fill window using bg color (L)

**for row in tiles:**

**for tile in row:**

**draw tile (U)**

update display (L)

Game function update (in: self)

**pass**

Game function decide continue (in: self)

**pass**

**class Tile**

**image**

**rectangle**

**border\_width = 3 (shared)**

**surface (shared)**

**value**

**fg\_color = pygame.Color(‘black’) (shared)**

**Tile class function set\_surface(in:cls,surface, out:)**

**set surface using argument**

**Tile function create(in: self, x, y ,width, height, image, value out: Tile)**

**set rectangle to Pygame.Rect using x, y, width and height**

**set image using argument**

**set value using argument**

**Tile function draw (in: self)**

**draw black rectangle to be background**

**draw image at rectangle.top+border width, rectangle.left + border height**