// Psudocode version 2.0.0

////////////////////////////////////////////////////////////////////////////////////////

// Menu part that is added to the base game

////////////////////////////////////////////////////////////////////////////////////////

// imports needed

pygame, sys, random, time, webbrowser, os

// globals added

// For the title size

TITLEFONT = pygame.font.Font(font = "freesansbold.ttf", size =40)

// For the text using in 2.4.1

SMALLFONT = pygame.font.Font(font = "freesansbold.ttf", size =20)

// colors

Screen\_color = black

Text color = white

// menu function

Def menu(game\_over=False):

// added in version 2.1.0

// how to play button / game instruction button added in version 2.2.0 and 2.2.1

If game\_over:

Start button text = font render (“play again”, color = blue)

Quit button text = font render (“quit”, color = green)

Instruction button text = font render (“game instruction”, color = red)

Title = TITLEFONT render (“game over you lost”, text\_color)

Else:

Start button text = font render (“Start”, color = blue)

Quit button text = font render (“quit”, color = green)

Instruction button text = font render (“how to play”, color = red)

Title = TITLEFONT render (“Welcome to Agar.io!”, text\_color)

Button width = 500

Button height = 150

Border width = 100

Top border height = 80

Start button rect = pygame rect (value, value, button width, button height)

Quit button rect = pygame rect (value , value, button width, button height)

Instruction button rect = pygame rect (value, value, button width, button height)

// This is where the stand-in background come in version 2.3.1

Stand ins = []

For in range(30)

New stand in = create( random randint (-100, 1000), random randint (-100, 1000), random color, size between 10 to 50, “stand\_in”)

Stand\_ins list adds new stand in

// start of loop

Start game = False

While not start game:

For event in pygame event get()

If event type == quit

Pygame quit()

Sys exit()

Elif event type == MOUSEBUTTONDOWN and event button == 1:

Mouse pos = pygame mouse get\_pos()

If start button has collision with mouse point:

Start game = True

Elif quit button has collision with mouse point:

Pygame quit()

Sys exit()

Elif instruction button collision with mouse point:

Webbrowser open new tab ( filename )

//Filename will look like ‘File:///’ + os.getcwd() + '/' + 'Manual.html'

// for resizing screen and keeping track of the screen

WIDTH, HEIGHT = pygame display get\_surface() get\_size()

// screen color

SCREEN fill color = screen\_color

//draws stand\_on screen

For stand\_in in stand\_ins:

Stand-in draw on screen with the x and y position that is stored in the stand-in object

// top border on the menu screen

Top border rect = pygame Rect (0, 0, WIDTH, top border height)

// draw the rect objects (buttons)

Pygame draw rect (SCREEN, color = grey, top border rect)

Pygame draw rect (SCREEN, color = dark grey, start button rect, border width)

Pygame draw rect (SCREEN, color = dark grey, quit button rect, border width)

Pygame draw rect (SCREEN, color = dark grey, instruction button rect, border width)

// setup the centering for the buttons

Title rect = title get\_rect(center=(WIDTH / 2, (HEIGHT / 2) - 200))

Start button rect = start button text get\_rect(center=(WIDTH / 2, HEIGHT / 2))

Quit button rect = quit\_button text get\_rect(center=(WIDTH / 2, (HEIGHT / 2) + 200))

Instruction button rect = instruction button text get\_rect(center=(WIDTH / 2, (HEIGHT / 2) + 100))

// draws to screen

SCREEN blit ( start button text, start button rect)

SCREEN blit ( quit button text, quit button rect)

SCREEN blit (instruction button text, instruction button rect)

SCREEN blit (title, title rect)

// update display

Pygame display update()

Return start game

Start game = menu(game\_over=False)

If start game:

// building objects for game and then runs game loop

// The game is part of version 1.0.0

//in the game there was pygame event types that were added

For event in pygame event get():

// … this is what is added to the version 1.0.0 from version

// from version 2.4.0 and 2.4.1

if event.type == pygame pressed and event.key == pygame m key:

start game = menu(game\_over=False)

if event.type == pygame key pressed and event.key == pygame k key:

game\_over = true

//if gameover had changes made

If game over:

// game over message was taking out with version 2.1.0, which added the game over menu.

Start game = menu(game\_over=True)

If start game:

Balls list empty

Bots list empty

for i in range(food\_balls):

new\_ball = Create(random.randint(-map\_size, map\_size), random.randint(-map\_size, map\_size), (random color), 5, ball")

balls.append(new\_ball)

// create the bots on the screen again.

for i in range(bot\_max):

new\_bot = Create(random.randint(-map\_size, map\_size), random.randint(-map\_size, map\_size), (random color), random.randint(bot\_min\_size, bot\_max\_size), "bot")

bots.append(new\_bot)

// set game over to false

game\_over = false

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// around the end of the game loop

// add information for the player on the screen for the key binds

advice = SMALLFONT render("Press m on keyboard to go back to main menu or press k on keyboard to reset the game", text\_color)

SCREEN blit(advice, (0, HEIGHT - 30))