print("--Running Pycraft")

try: # checks all modules are installed

import pygame # the modules to import:

from pygame.locals import \* # loads EVERYTHING that pygame requires

pygame.init() # initiates pygame

from OpenGL.GL import \* # imports sections of opengl

from OpenGL.GLU import \*

from OpenGL.GLUT import \* # end

import tkinter as tk # imports tkinter (press q for dev menu)

import time

import random

import sys

import os # imports misc and python modules (builtin)

from PIL import Image # used for better skybox handling

import pyautogui # used for taking screenshots for savepoints and pauses and inv back (not imp)

import pywavefront # used to load and understand .obj files (very handy)

import psutil # used to tell the code some system data

import timeit

except: # is a module is not installed then

print("An error occured...") # tell the user...

import FinalStartup # ...and run the installer (To FIX)

else: # if the modules are installed then:

contIG = 0 # ingnore this and move on to the rest of the program

version = "20p1009\_20a" # what version of pycraft are we up to:

clock = pygame.time.Clock() # starts running pygame's clock function

rendis = 60 # render distance is set to 60 by default

FPS = 120 # the Frames Per Seccond is set to 60

FOV = 70 # the Field of View is set to 70

cameraANGspeed = 3 # CameraANGspeed (speed of camera panning) is set to 0.5

aa = True # anti aliasing

RenderFOG = True # render fog

FanSky = True # fancy skybox, moons, sun, stars, celestial events

FanPart = True # fancy particles, leaves, rain, walking, dust, HDR (?)

sound = True # sound on/off

soundVOL = 100 # sets the sound volume (%)

music = True # sets the music on/off

musicVOL = 75 # sets the music volume (%)

devmode = 1 # turns off optional features by default

width, height = 1200, 720 # defines the size of the display (res~720)

vertices = ((1, -1, -1), (1, 1, -1), (-1, 1, -1), (-1, -1, -1), (1, -1, 1), (1, 1, 1), (-1, -1, 1), (-1, 1, 1)) # defines the coordinates for the cube

edges = ((0,1), (0,3), (0,4), (2,1), (2,3), (2,7), (6,3), (6,4), (6,7), (5,1), (5,4), (5,7)) # and the edges of the cube

colors = ((0,1,0), (1,1,0), (1,0,0), (1,0,0), (1,1,1), (1,0,0), (1,0,0), (1,0,0), (1,0,0), (1,0,0), (1,0,0), (1,0,0)) # and the colour of the cube

surfaces = ((0,1,2,3), (3,2,7,6), (6,7,5,4), (4,5,1,0), (1,5,7,2), (4,0,3,6)) # and the faces of the cube

ground\_surfaces = (1,2,3,4, 5) # the faces of the ground

ground\_vertices = ((-100000,-150,-100000),(-100000,-150,100000),(100000,-150,-100000), (100000, -150, 100000)) # the corners of the ground

def Start(): # this procedure creates an intro, and checks all modules are installed.

data = open("D:\\PYGAME\\Data\_Files\\data.txt","r+") # opens a file that is used to test weather the program's on its first run

if sys.platform == 'win32' or sys.platform == 'win64':

os.environ['SDL\_VIDEO\_CENTERED'] = '1'

if data.read() == "False": # is the program is on its first run...

data.seek(0) # navigates to the first character in the file (index = 0)

data.truncate() # this secects all the data in the file...and clears it.

data.close() # saves the file

data = open("D:\\PYGAME\\Data\_Files\\data.txt","r+") # re opens it

data.write("True") # this tells the program when its next run that this is not the first run

data.close() # this clears the file from the program as it is no longer needed, freeing up 4 bytes of RAM

Display = pygame.display.set\_mode((width, height)) # creates a GUI with the size of width and height

fade = pygame.Surface((width, height)) # creates a temporary GUI with the size; width, height

LoadingImage = pygame.image.load("D:\\PYGAME\\Resources\\General\_Resources\\Pycraft\_Short\_Loading.JPG") # loads a image (Loading v2.jpg) to RAM.

Display.blit(LoadingImage, (0,0)) # renders the image (see prv line) to the GUI, Display at the position (0,0), top left

wallpaper = pygame.image.load("D:\\PYGAME\\Resources\\General\_Resources\\Pycraft\_Background.jpg")

pygame.display.set\_caption(f"Pycraft: {version} | Welcome") # sets the Display caption to Pycraft: {version} | Welcome

PresentsFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 20) # loads the font; Book Antiqua and stores it in the variable PresentsFont

PycraftFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 60)

NameFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 45) # END OF FONT LOADING

Display.fill([0,0,0]) # sets the display colour to black

pygame.time.wait(2000) # waits the program for 2000 milliseconds (2 seconds)

name = NameFont.render("Thomas Jebson",True,(255,255,255)) # loads Thomas Jebson into the RAM with the font NameFont

Display.blit(name,(480,360)) # renders the name variable to the screen at the position, (480,360)

pygame.display.flip() # updates the GUI to add the current items

pygame.time.wait(2000)

pygame.event.get() # makes the program not crash

name = PresentsFont.render("presents",True,(255,255,255))

Display.blit(name,(560,400))

pygame.display.flip()

pygame.time.wait(4000)

pygame.event.get()

Display.fill([0,0,0])

name = PycraftFont.render("Pycraft",True,(255,255,255))

Display.blit(name,(500,360))

pygame.display.flip()

pygame.time.wait(2000)

pygame.event.get() # END OF FONT RENDERING ANIMATION

for a in range(360): # run this 360 times:

Display.fill([0,0,0]) # sets the display to black

Display.blit(name,(500,(360-a))) # re-renders the font to the GUI Display, changing the y axis each time

pygame.display.flip() # refreshes the GUI to show the changes

pygame.time.wait(5) # pauses for 5 milliseconds

pygame.event.get() # makes the game not crash

for b in range(0,300): # run this program 300 times

fade.set\_alpha((300-b)) # sets the seccond surface's alpha value (transparancy) to change

Display.fill((0,0,0)) # sets the display to black

Display.blit(wallpaper, (0,0)) # loads the wallpaper

Display.blit(fade, (0,0)) # hides the wallpaper by the transparancy value

pygame.display.flip() # updates the GUI

pygame.time.wait(5) # waits 5 milliseconds

pygame.event.get() # makes the program not crash

return True # tells the program that this is now finished and to get on with other things

elif data.read() == "True": # if the file reads True

data.close() # closes the file to clear up a bit of RAM

Display = pygame.display.set\_mode((width, height)) # creates a display GUI that displays the loading screen until HS is loaded

pygame.display.set\_caption(f"Pycraft: {version} | Loading") # sets caption to this

LoadingImage = pygame.image.load("D:\\PYGAME\\Resources\\General\_Resources\\Pycraft\_Short\_Loading.JPG") # loads the image to RAM

Display.blit(LoadingImage, (0,0)) # loads the image to the display

pygame.display.flip() # updates the display

return True # tells the program to move on

else: # if the file reads something else then it will run the startup again

data.seek(0) # goes to the position to character 0

data.truncate() # clears the data file

data.close() # saves the file

data = open("D:\\PYGAME\\Data\_Files\\data.txt","r+")

data.write("False") # and sets it to false so the program re-runs

data.close() # cleans up the RAM

Display = pygame.display.set\_mode((width, height)) # does the same thing now as the previus section, loading the loading screen

pygame.display.set\_caption(f"Pycraft: {version} | Loading")

LoadingImage = pygame.image.load("D:\\PYGAME\\Resources\\General\_Resources\\Pycraft\_Short\_Loading.JPG")

Display.blit(LoadingImage, (0,0))

pygame.display.flip()

return False # returns false so the program knows to re-run this

def settings(rendis, FPS, FOV, cameraANGspeed, devmode, aa, RenderFOG, FanSky, FanPart, sound, soundVOL, music, musicVOL): # defines the setting procedure

Display = pygame.display.set\_mode((width, height)) # sets the GUI size

LoadingImage = pygame.image.load("D:\\PYGAME\\Resources\\General\_Resources\\Pycraft\_Short\_Loading.JPG") # load the loading screen (ironic?)

Display.blit(LoadingImage, (0,0)) # displays the image to the GUI

pygame.display.flip() # updates the display

if devmode == 10 or devmode-10 == 0: # if developer mode is enabled

FCol = (179, 204, 255)

pygame.display.set\_caption(f"Pycraft: {version}: Settings | Developer Mode") # sets the caption to developer mode for the user

else:# if the developer mode is not enabled then it is set to default

FCol = (255,255,255)

pygame.display.set\_caption(f"Pycraft: {version}: Settings") # sets the display caption

VersionFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 15) # loads the version font for the pre-sets

MainTitleFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 60) # loads the main title font

wallpaper = pygame.image.load("D:\\PYGAME\\Resources\\General\_Resources\\Pycraft\_Background.jpg") # loads the background image (feature coming here)

LOWFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf",15) # loads the fonts for the pre-sets

MEDIUMFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 15)

HIGHFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 15)

ADAPTIVEFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 15) # END OF PRE-SET FONT LOADING

DataFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 15)

data1 = []

data2 = []

data3 = []

run = 0

rerun = 0

TempMx, TempMy = 0,0 # used to get the previous mouse coordinates

Mx, My = 0,0 # sets the current mouse pos to zero, zero

mousebuttondown = False # the mouse button (left) is not down

while True: # main settings loop

TempMx, TempMy = Mx, My

Mx, My = pygame.mouse.get\_pos() # gets the current mouse position

eFPS = clock.get\_fps() # gets the current fps

run += 1

for event in pygame.event.get(): # detects events, (keypresses, display interactions, mousebutton presses, ect.)

if event.type == pygame.QUIT: # if the 'x' in the corner is pressed then;

return rendis, FPS, FOV, cameraANGspeed, aa, RenderFOG, FanSky, FanPart, sound, soundVOL, music, musicVOL

elif event.type == pygame.KEYDOWN: # detects keypresses

if event.key == pygame.K\_SPACE and devmode < 10: # if developer mode is getting enabled...

devmode += 1 # increases the devmode value

if devmode >= 5 and devmode <= 9: # if devmode is getting enabled then

pygame.display.set\_caption(f"Pycraft: {version}: Settings | you are: {10-devmode} steps away from being a developer") # tells the user that they are enabling devmode

elif devmode == 10: # if devmode is enabled then

pygame.display.set\_caption(f"Pycraft {version}: Settings | Developer mode | V: 0,00 | FPS: {clock.get\_fps()} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}") # tells the user

FCol = (220,220,255)

else:# if the developer mode is not enabled then it is set to default

FCol = (255,255,255) # sets the pycraft screen to white if not in devmode

pygame.display.set\_caption(f"Pycraft: {version}: Settings") # sets the display caption

if event.key == pygame.K\_x: # detects if 'x' key is pressed

devmode = 1 # restes the devmode value

FCol = (255,255,255) # sets the title colour to white

pygame.display.set\_caption(f"Pycraft: {version}: Settings") # sets the caption to default

if event.key == pygame.K\_q: # detects if 'q' key pressed

DataWindow = tk.Tk() # sets the tkinter root

DataWindow.title("Player Information") # sets the display (window) caption

DataWindow.configure(width = 500, height = 300) # sets the window size (not needed (?))

DataWindow.configure(bg="lightblue") # sets the background colour

VersionData = f"Pycraft: {version}" # adds the curent version if caption fails

CoordinatesData = f"Coordinates: x: {Mx} y: {My} z: 0.0 FacinE: 0.0,0.0,0.0" # gives information about the coordinates of the user

FPSData = f"FPS: Actual: {eFPS} Max: {FPS}" # gives the FPS unsimplified before menu opens (pauses everything)

VersionData = tk.Label(DataWindow, text=VersionData) # loads this to the currently active Tk window

CoordinatesData = tk.Label(DataWindow, text=CoordinatesData) # loads the coordinates data to the GUI

FPSData = tk.Label(DataWindow, text=FPSData) # loads the fps data to the window

VersionData.grid(row = 0, column = 0, columnspan = 2) # coordinates are given

CoordinatesData.grid(row = 1, column = 0, columnspan = 2)

FPSData.grid(row = 2, column = 0, columnspan = 2)# END OF COORDINATES SPECIFIED

DataWindow.mainloop() # Tkinter will run what happens next until

DataWindow.quit() # it is ordered to quit

if event.key == pygame.K\_x: # detects if x key is pressed

devmode = 1 # resets devmode to 1

FCol = (255,255,255) # resets the font colour

pygame.display.set\_caption(f"Pycraft: {version}: Settings") # and the caption

elif event.type == pygame.MOUSEBUTTONDOWN: # if the mouse button is down (left)

mousebuttondown = True # this variable is set to True

elif event.type == pygame.MOUSEBUTTONUP: # if the mouse button is up (left)

mousebuttondown = False # this variable is set to False

titletFont = MainTitleFont.render("Pycraft", aa, FCol) # main title with colour defined with developer mode

RendertFont = VersionFont.render(f"Render Distance: {rendis}",aa, (255,255,255)) # setting title plus data

FPStFont = VersionFont.render(f"FPS: Actual: {eFPS} Max: {int(FPS)}",aa, (255,255,255))

FOVtFont = VersionFont.render(f"FOV: {FOV}",aa, (255,255,255))

CamRottFont = VersionFont.render(f"Camera Rotation Speed: {round(cameraANGspeed,1)}",aa, (255,255,255)) # END OF SETTING TITLE + DATA

ModetFont = VersionFont.render("Mode; , , , .",aa, (255,255,255)) # gives the pre defined settings a location

LOWtFont = LOWFont.render("LOW",aa, (255,255,255)) # low preset font

MEDIUMtFont = MEDIUMFont.render("MEDIUM",aa, (255,255,255)) # medium preset font

HIGHtFont = HIGHFont.render("HIGH",aa, (255,255,255)) # high preset font

ADAPTIVEtFont = ADAPTIVEFont.render("ADAPTIVE",aa, (255,255,255)) # adaptive preset font

AAtFont = VersionFont.render(f"Anti-Aliasing: {aa}",aa, (255,255,255)) # setting title plus data part 2

RenderFogtFont = VersionFont.render(f"Render Fog: {RenderFOG}",aa, (255,255,255))

FancySkytFont = VersionFont.render(f"Fancy Skies: {FanSky}",aa, (255,255,255))

FancyParticletFont = VersionFont.render(f"Fancy Partices: {FanPart}",aa, (255,255,255))

SoundtFont = VersionFont.render(f"Sound: {sound}",aa, (255,255,255))

SoundVoltFont = VersionFont.render(f"Sound Volume: {soundVOL}",aa, (255,255,255))

MusictFont = VersionFont.render(f"Music: {music}",aa, (255,255,255))

MusicVoltFont = VersionFont.render(f"Music Volume: {musicVOL}",aa, (255,255,255)) # END OF SETTING TITLE + DATA 2

Display.blit(wallpaper, (0,0)) # blits the background image to the screen

Display.blit(titletFont, (500,0)) # then the title font

Display.blit(RendertFont, (0,100)) # then the setting data part 1

Display.blit(FPStFont, (0,150))

Display.blit(FOVtFont, (0,200))

Display.blit(CamRottFont, (0,250))

Display.blit(ModetFont, (0,50)) # then the pre set modes

Display.blit(LOWtFont, (48,50))

Display.blit(MEDIUMtFont, (90,50))

Display.blit(HIGHtFont, (165,50))

Display.blit(ADAPTIVEtFont, (215,50)) # then continues with the settings

Display.blit(AAtFont, (0,300))

Display.blit(RenderFogtFont, (0,350))

Display.blit(FancySkytFont, (0,400))

Display.blit(FancyParticletFont, (0,450))

Display.blit(SoundtFont, (0,500))

Display.blit(SoundVoltFont, (0,550))

Display.blit(MusictFont, (0,600))

Display.blit(MusicVoltFont, (0,650))

render\_rect = pygame.Rect(50, 130, 450, 10) # x,y, width, height.

FPS\_rect = pygame.Rect(50, 180, 450, 10)

FOV\_rect = pygame.Rect(50, 230, 450, 10)

CAM\_rect = pygame.Rect(50, 280, 450, 10)

sound\_rect = pygame.Rect(50, 580, 450, 10)

music\_rect = pygame.Rect(50, 680, 450, 10)

aa\_rect = pygame.Rect(50, 330, 50, 10)

RenderFOG\_Rect = pygame.Rect(50, 380, 50, 10)

Fansky\_Rect = pygame.Rect(50, 430, 50, 10)

FanPart\_Rect = pygame.Rect(50, 480, 50, 10)

sound\_Rect = pygame.Rect(50, 530, 50, 10)

music\_Rect = pygame.Rect(50, 630, 50, 10)# end of pygame.Rect()

pygame.draw.rect(Display, (83,83,83), render\_rect, 0) # draws the rectangle to the display with a dark grey, filled

pygame.draw.rect(Display, (83,83,83), FPS\_rect, 0)

pygame.draw.rect(Display, (83,83,83), FOV\_rect, 0)

pygame.draw.rect(Display, (83,83,83), CAM\_rect, 0)

pygame.draw.rect(Display, (83,83,83), sound\_rect, 0)

pygame.draw.rect(Display, (83,83,83), music\_rect, 0)

pygame.draw.rect(Display, (83,83,83), aa\_rect, 0)

pygame.draw.rect(Display, (83,83,83), RenderFOG\_Rect, 0)

pygame.draw.rect(Display, (83,83,83), Fansky\_Rect, 0)

pygame.draw.rect(Display, (83,83,83), FanPart\_Rect, 0)

pygame.draw.rect(Display, (83,83,83), sound\_Rect, 0)

pygame.draw.rect(Display, (83,83,83), music\_Rect, 0)# end of pygame.draw.rect()]

if mousebuttondown == True:

if My > 130 and My < 140: # if the mouse is in these positions and mousebuttondown is True

if Mx > TempMx and rendis < 490: # if player tries to slide the slider (ironic?)

rendis += 1 # rendis is raised

elif Mx < TempMx and rendis > 60: # if the player tries to lower the rendis

rendis -= 1 # rendis is lowered

elif Mx == TempMx: # if player hovers over the circle

rendis = rendis # nothing happens

if rendis < 60: # however if rendis is less than 60

rendis = 60 # this is the minimum so stops it from going lower

elif rendis > 490: # the opposite is also true

rendis = 490

if My > 180 and My < 190: # same as before but for the FPS slider

if Mx > TempMx and FPS < 445: # with a maximum refresh rate of 240 as tech limited

FPS += 1

elif Mx < TempMx and FPS > 15: # below this number things begin to break...

FPS -= 1

if FPS < 15:

FPS = 16

elif FPS > 445:

FPS = 444

if My > 230 and My < 240:

if Mx > TempMx and FOV < 98:

FOV += 1

elif Mx < TempMx and FOV > 12:

FOV -= 1

if FOV < 12:

FOV = 13

elif FOV > 98:

FOV = 97

if My > 280 and My < 290:

if Mx > TempMx and cameraANGspeed < 5.0:

cameraANGspeed += 0.1

elif Mx < TempMx and cameraANGspeed > 0.0:

cameraANGspeed -= 0.1

if cameraANGspeed > 5.0:

cameraANGspeed = 4.9

elif cameraANGspeed < 0:

cameraANGspeed = 0.1

if My > 580 and My < 590 and sound == True:

if Mx > TempMx and soundVOL < 100:

soundVOL += 1

elif Mx < TempMx and soundVOL > 0:

soundVOL -= 1

if soundVOL > 100:

soundVOL = 100

elif soundVOL < 0:

soundVOL = 0

if My > 680 and My < 690 and music == True: # there is no point adjusting the music vol when there is none!

if Mx > TempMx and musicVOL < 100:

musicVOL += 1

elif Mx < TempMx and musicVOL > 0:

musicVOL -= 1

if musicVOL > 100:

musicVOL = 100

elif musicVOL < 0:

musicVOL = 0

if My > 330 and My < 340: # END OF SLIDER, start of on/off switches, anti-aliasing

if aa == True: # if on switch off

aa = False # switches off

mousebuttondown = False # stops the on/off button breaking

elif aa == False: # if off switch on

aa = True # switches on

mousebuttondown = False # stops the on/off button from breaking

if My > 380 and My < 390:

if RenderFOG == True:

RenderFOG = False

mousebuttondown = False

elif RenderFOG == False:

RenderFOG = True

mousebuttondown = False

if My > 430 and My < 440:

if FanSky == True:

FanSky = False

mousebuttondown = False

elif FanSky == False:

FanSky = True

mousebuttondown = False

if My > 480 and My < 490:

if FanPart == True:

FanPart = False

mousebuttondown = False

elif FanPart == False:

FanPart = True

mousebuttondown = False

if My > 530 and My < 540:

if sound == True:

sound = False

mousebuttondown = False

elif sound == False:

sound = True

mousebuttondown = False

if My > 630 and My < 640:

if music == True:

music = False

mousebuttondown = False

elif music == False:

music = True

mousebuttondown = False

if My >= 40 and My <= 70 and Mx >= 40 and Mx <= 80:

LOWFont.set\_underline(True)

if mousebuttondown == True:

rendis = 60

FPS = random.randint(15,30)

aa = False

RenderFOG = False

FanSky = False

FanPart = False

mousebuttondown = False

else:

LOWFont.set\_underline(False)

if My >= 40 and My <= 70 and Mx >= 90 and Mx <= 155:

MEDIUMFont.set\_underline(True)

if mousebuttondown == True:

rendis = 80

FPS = random.randint(30,60)

aa = True

RenderFOG = False

FanSky = True

FanPart = False

mousebuttondown = False

else:

MEDIUMFont.set\_underline(False)

if My >= 40 and My <= 70 and Mx >= 165 and Mx <= 205:

HIGHFont.set\_underline(True)

if mousebuttondown == True:

rendis = 100

FPS = random.randint(60, 120)

aa = True

RenderFOG = True

FanSky = True

FanPart = True

mousebuttondown = False

else:

HIGHFont.set\_underline(False)

if My >= 40 and My <= 70 and Mx >= 215 and Mx <= 300:

ADAPTIVEFont.set\_underline(True)

if mousebuttondown == True:

mousebuttondown = False

rendis = (psutil.cpu\_freq(percpu=True)[0][2])/20

FPS = (psutil.cpu\_freq(percpu=True)[0][2])/35

if (psutil.cpu\_freq(percpu=True)[0][2])/10 > 300 and psutil.virtual\_memory().total > 8589934592:

aa = True

RenderFog = True

FanSky = True

FanPart = True

elif (psutil.cpu\_freq(percpu=True)[0][2]) > 200 and psutil.virtual\_memory().total > 4294967296:

aa = True

RenderFog = True

FanSky = True

FanPart = False

elif (psutil.cpu\_freq(percpu=True)[0][2]) > 100 and psutil.virtual\_memory().total > 2147483648:

aa = False

RenderFog = False

FanSky = True

FanPart = False

elif (psutil.cpu.frequ(percpu=True)[0][2]) < 100 and (psutil.cpu.freq(percpu=True)[0][2]) > 75 and psutil.virtual\_memory().total > 1073741824:

aa = False

RenderFog = False

FanSky = False

FanPart = False

else:

ADAPTIVEFont.set\_underline(False)

pygame.draw.circle(Display, (255,255,255), (int(rendis), 135), 9) # draws the slider outside (plan)

pygame.draw.circle(Display, (83,83,83), (int(rendis), 135), 6) # draws the slider inside

pygame.draw.circle(Display, (255,255,255), (int(FPS+45), 185), 9)

pygame.draw.circle(Display, (83,83,83), (int(FPS+45), 185), 6)

pygame.draw.circle(Display, (255,255,255), (int(FOV\*5), 235), 9)

pygame.draw.circle(Display, (83,83,83), (int(FOV\*5), 235), 6)

pygame.draw.circle(Display, (255,255,255), (int(cameraANGspeed\*89)+45, 285),9)

pygame.draw.circle(Display, (83,83,83), (int(cameraANGspeed\*89)+45, 285), 6)

pygame.draw.circle(Display, (255,255,255), (int(soundVOL\*4.4)+50, 585), 9)

pygame.draw.circle(Display, (83,83,83), (int(soundVOL\*4.4)+50, 585), 6)

pygame.draw.circle(Display, (255,255,255), (int(musicVOL\*4.4)+50, 685), 9)

pygame.draw.circle(Display, (83,83,83), (int(musicVOL\*4.4)+50, 685), 6)

if devmode == 10 or devmode-10 == 0:

pygame.display.set\_caption(f"Pycraft: {version}: Settings | Developer mode | V: 0,0,0 | FPS: {clock.get\_fps()} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}")

else:

pygame.display.set\_caption(f"Pycraft: {version}: Settings")

# ani aliasing selection

if aa == True: # sets the dial/indicator position to 90,335 (x,y) when on

pygame.draw.circle(Display, (255,255,255), (90, 335), 9)

pygame.draw.circle(Display, (83,83,83), (90, 335), 6)

elif aa == False: # sets the dial/indicator position to 60,335 (x,y) when off

pygame.draw.circle(Display, (255,255,255), (60, 335), 9)

pygame.draw.circle(Display, (83,83,83), (60, 335), 6)

# render fog selection

if RenderFOG == True:

pygame.draw.circle(Display, (255,255,255), (90, 385), 9)

pygame.draw.circle(Display, (83,83,83), (90, 385), 6)

elif RenderFOG == False:

pygame.draw.circle(Display, (255,255,255), (60, 385), 9)

pygame.draw.circle(Display, (83,83,83), (60, 385), 6)

# fancy skies selection

if FanSky == True:

pygame.draw.circle(Display, (255,255,255), (90, 435), 9)

pygame.draw.circle(Display, (83,83,83), (90, 435), 6)

elif FanSky == False:

pygame.draw.circle(Display, (255,255,255), (60, 435), 9)

pygame.draw.circle(Display, (83,83,83), (60, 435), 6)

# fancy particles selection

if FanPart == True:

pygame.draw.circle(Display, (255,255,255), (90, 485), 9)

pygame.draw.circle(Display, (83,83,83), (90, 485), 6)

elif FanPart == False:

pygame.draw.circle(Display, (255,255,255), (60, 485), 9)

pygame.draw.circle(Display, (83,83,83), (60, 485), 6)

# sound selection

if sound == True:

pygame.draw.circle(Display, (255,255,255), (90, 535), 9)

pygame.draw.circle(Display, (83,83,83), (90, 535), 6)

elif sound == False:

pygame.draw.circle(Display, (255,255,255), (60, 535), 9)

pygame.draw.circle(Display, (83,83,83), (60, 535), 6)

# music selection

if music == True:

pygame.draw.circle(Display, (255,255,255), (90, 635), 9)

pygame.draw.circle(Display, (83,83,83), (90, 635), 6)

elif music == False:

pygame.draw.circle(Display, (255,255,255), (60, 635), 9)

pygame.draw.circle(Display, (83,83,83), (60, 635), 6)

if run >= 1000:

run = 0

rerun += 1

if rerun >= 1:

try:

data1[run] = ([((run/5)+1000), ((400-eFPS)-250)])

data2[run] = ([((run/5)+1000), ((400-((psutil.cpu\_percent())))-250)])

data3[run] = ([((run/5)+1000), ((100-psutil.virtual\_memory().percent)\*2)-25])

except:

contIG = 0

else:

contIG = 0

else:

data1.append([((run/5)+1000), ((400-(eFPS)-250))])

data2.append([((run/5)+1000), ((400-((psutil.cpu\_percent())))-250)])

data3.append([((run/5)+1000), ((100-psutil.virtual\_memory().percent)\*2)+25])

if devmode == 10: # checks if devmode is equal to 10

dev\_Rect = pygame.Rect(1000,0,200, 200)

pygame.draw.rect(Display, (0,0,0), dev\_Rect)

if run >= 10:

pygame.draw.lines(Display, (0,255,0), False, (data2))

pygame.draw.lines(Display, (255,0,0), False, (data1))

pygame.draw.lines(Display, (0,0,255), False, (data3))

pygame.draw.line(Display, (255,255,255), (((run/5)+1000), 20), (((run/5)+1000), 200))

runFont = DataFont.render(f"{psutil.virtual\_memory().percent} | {str(psutil.cpu\_percent())} | {str(run)} | {str(rerun)} | {str(round(eFPS, 2))}", False, (255,255,255)) # stores the advanced data to be used when devmode is enabled

Display.blit(runFont, (1000,0)) # displays the data in the top left

pygame.display.flip() # updates the display

clock.tick(FPS) # sets the refresh rate to cap out at 30 fps

def Character\_Customiser(devmode): # sets up the character customiser, (not finished), (girl, women, skin colour?)

Home\_Screen(devmode) # as nothing is happening here it takes you back to the home screen

clock.tick(30) # continues the 30 fps rule

def Achievements(devmode): # sets up the acievements (targets, score?)

Home\_Screen(devmode) # goes back to the home screen

clock.tick(30) # just continues

def Credits(devmode, aa): # loads the credits menu

Display = pygame.display.set\_mode((width, height))# sets the GUI size

LoadingImage = pygame.image.load("D:\\PYGAME\\Resources\\General\_Resources\\Pycraft\_Short\_Loading.JPG") # load the loading screen (ironic?)

Display.blit(LoadingImage, (0,0)) # displays the image to the GUI

pygame.display.flip() # updates the display

if devmode == 10 or devmode-10 == 0: # if developer mode is enabled

FCol = (179, 204, 255) # sets the colour of the title font + devmode dependant features

pygame.display.set\_caption(f"Pycraft: {version}: Credits and Change-Log | Developer Mode | V: 0,00 | FPS: {clock.get\_fps()} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}") # sets the caption to developer mode for the user

else:# if the developer mode is not enabled then it is set to default

FCol = (255,255,255)

pygame.display.set\_caption(f"Pycraft: {version}: Credits and Change-Log") # sets the display caption

VersionFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 15) # loads the version font for the pre-sets

MainTitleFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 60) # loads the main title font

DataFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 15)

wallpaper = pygame.image.load("D:\\PYGAME\\Resources\\General\_Resources\\Pycraft\_Background.jpg") # loads the background image (feature coming here)

Mx, My = pygame.mouse.get\_pos() # gets the mouse position

data1 = []

data2 = []

data3 = []

run = 0

rerun = 0

while True: # main game loop

eFPS = clock.get\_fps() # gets the current FPS limmited to 30

Mx, My = pygame.mouse.get\_pos() # gets the mouse position

rand = 0

run += 1

for event in pygame.event.get(): # detects events, (keypresses, display interactions, mousebutton presses, ect.)

if event.type == pygame.QUIT: # if the 'x' in the corner is pressed then;

Home\_Screen(devmode, rendis, FPS, FOV, cameraANGspeed, aa, RenderFOG, FanSky, FanPart, sound, soundVOL, music, musicVOL) # goes back to the home screen

elif event.type == pygame.KEYDOWN: # detects keypresses

if event.key == pygame.K\_SPACE and devmode < 10: # if developer mode is getting enabled...

devmode += 1 # increases the devmode value

if devmode >= 5 and devmode <= 9: # if devmode is getting enabled then

pygame.display.set\_caption(f"Pycraft: {version}: Credits and Change-Log | you are: {10-devmode} steps away from being a developer") # tells the user that they are enabling devmode

elif devmode == 10: # if devmode is enabled then

pygame.display.set\_caption(f"Pycraft: {version}: Credits and Change-Log | Developer mode | V: 0,00 | FPS: {clock.get\_fps()} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}") # tells the user

FCol = (220,220,255)

else:# if the developer mode is not enabled then it is set to default

FCol = (255,255,255) # sets the pycraft screen to white if not in devmode

pygame.display.set\_caption(f"Pycraft: {version}: Credits and Change-Log") # sets the display caption

if event.key == pygame.K\_q: # detects if 'q' key pressed

DataWindow = tk.Tk() # sets the tkinter root

DataWindow.title("Player Information") # sets the display (window) caption

DataWindow.configure(width = 500, height = 300) # sets the window size (not needed (?))

DataWindow.configure(bg="lightblue") # sets the background colour

VersionData = f"Pycraft: {version}" # adds the curent version if caption fails

CoordinatesData = f"Coordinates: x: {Mx} y: {My} z: 0.0 FacinE: 0.0,0.0,0.0" # gives information about the coordinates of the user

FPSData = f"FPS: Actual: {eFPS} Max: {FPS}" # gives the FPS unsimplified before menu opens (pauses everything)

VersionData = tk.Label(DataWindow, text=VersionData) # loads this to the currently active Tk window

CoordinatesData = tk.Label(DataWindow, text=CoordinatesData) # loads the coordinates data to the GUI

FPSData = tk.Label(DataWindow, text=FPSData) # loads the fps data to the window

VersionData.grid(row = 0, column = 0, columnspan = 2) # coordinates are given

CoordinatesData.grid(row = 1, column = 0, columnspan = 2)

FPSData.grid(row = 2, column = 0, columnspan = 2)# END OF COORDINATES SPECIFIED

DataWindow.mainloop() # Tkinter will run what happens next until

DataWindow.quit() # it is ordered to quit

if event.key == pygame.K\_x: # detects if x key is pressed

devmode = 1 # resets devmode to 1

FCol = (255,255,255) # resets the font colour

pygame.display.set\_caption(f"Pycraft: {version}: Credits and Change-Log") # and the caption

TitleFont = MainTitleFont.render("Pycraft", aa, FCol) # main title with colour defined with developer mode

Credits1 = VersionFont.render("Head of Development: Thomas Jebson", aa, (255,255,255)) # start of long font for credits and change-log

Credits2 = VersionFont.render("Music By: Thomas Jebson", aa, (255,255,255))

Credits3 = VersionFont.render("Other Programmers:", aa, (255,255,255))

Credits4 = VersionFont.render(" - Pygame: illume, pygameci, takowl", aa, (255,255,255))

Credits5 = VersionFont.render(" - PyOpenGL: mcfletch", aa, (255,255,255))

Credits6 = VersionFont.render(" - OpenGL: Silicon Graphics, Khronos Group", aa, (255,255,255))

Credits7 = VersionFont.render(" - Python: Python Software Foundation", aa, (255,255,255))

Credits8 = VersionFont.render(" - Numpy: ahaldane, charlesr.harris, matthew.brett, mattip, rgommers, teoliphant", aa, (255,255,255))

Credits9 = VersionFont.render(" - PIL / pillow / Python Imaging Libary: aclark, hugovk, radarhere, wiredfool", aa, (255,255,255))

Credits10 = VersionFont.render(" - PyAutoGUI: AlSweigart", aa, (255,255,255))

Credits11 = VersionFont.render(" - Matplotlib: ivanov, matthew.brett, mdbloom, QuLogic, Thomas.Caswell", aa, (255,255,255))

Credits12 = VersionFont.render(" - Kiwisolver: mdartiaih, sccolbert", aa, (255,255,255))

Credits13 = VersionFont.render(" - Six: gutworth", aa, (255,255,255))

Credits14 = VersionFont.render(" - cycler: matthew.brett, mdbloom2, Thomas.Caswell", aa, (255,255,255))

Credits15 = VersionFont.render(" - pyparsing: ptmcg", aa, (255,255,255))

Credits16 = VersionFont.render(" - python-dateutil: dateutilbot, jarondl, pganssle, tpievila", aa, (255,255,255))

Credits17 = VersionFont.render(" - mouseinfo: AlSweigart", aa, (255,255,255))

Credits18 = VersionFont.render(" - pygetwindow: AlSweigart", aa, (255,255,255))

Credits19 = VersionFont.render(" - pymsgbox: AlSweigart", aa, (255,255,255))

Credits20 = VersionFont.render(" - pyperclip: AlSweigart, cblgh", aa, (255,255,255))

Credits21 = VersionFont.render(" - pyrect: AlSweigart", aa, (255,255,255))

Credits22 = VersionFont.render(" - pyscreeze: AlSweigart", aa, (255,255,255))

Credits23 = VersionFont.render(" - pytweening: AlSweigart", aa, (255,255,255))

Credits24 = VersionFont.render(" - pywavefront: einarf, greenmoss", aa, (255,255,255))

ChangeLog1 = VersionFont.render("Pre-release (alpha, a):", aa, (255,255,255))

ChangeLog2 = VersionFont.render(" - 1 - Created New presentation and experimented with a concept idea about minecraft with curves, along with a story-line and side projects!", aa, (255,255,255))

ChangeLog3 = VersionFont.render(" - 2 - Started an online search for a 3d rendering engine and found pythonprogramming.net", aa, (255,255,255))

ChangeLog4 = VersionFont.render(" - 3 - started making and understaning OpenGL and my video game", aa, (255,255,255))

ChangeLog5 = VersionFont.render(" - 4 - started making multiple cubes on the screen and a jump animation along with w, a, s, d keys for movement", aa, (255,255,255))

ChangeLog6 = VersionFont.render(" - 5 - New naming system, day of the month, first letter of the game, version, month, -, year, version (a/b)", aa, (255,255,255))

ChangeLog7 = VersionFont.render(" - 28p0506-20a - added a home screen, credits and changelog along with buttons", aa, (255,255,255))

ChangeLog8 = VersionFont.render(" - 04p0607-20a - added a settings menu, started on sounds and a (very) rudimentary physics engine, it's been 7 days since programming started", True, (255,255,255))

ChangeLog9 = VersionFont.render(" - 07p0707-20a - completely re-programmed the entire program, cleaned it up, comments and +10 FPS", aa, (255,255,255))

ChangeLog10 = VersionFont.render(" - 17p0807-20a - added skybox, better keypresses, started work on OpenGL .obj rendering and CAD of map", aa, (255,255,255)) # END OF LONG FONT FOR CREDITS AND CHANGE-LOG

Display.blit(wallpaper, (0,0)) # start of adding data to the Display

Display.blit(TitleFont, (500,0))

Display.blit(Credits1, (0,100))

Display.blit(Credits2, (0,115))

Display.blit(Credits3, (0,130))

Display.blit(Credits4, (0,145))

Display.blit(Credits5, (0,160))

Display.blit(Credits6, (0,175))

Display.blit(Credits7, (0,190))

Display.blit(Credits8, (0,205))

Display.blit(Credits9, (0,220))

Display.blit(Credits10, (0,235))

Display.blit(Credits11, (0,250))

Display.blit(Credits12, (0,265))

Display.blit(Credits13, (0,280))

Display.blit(Credits14, (0,295))

Display.blit(Credits15, (0,310))

Display.blit(Credits16, (0,325))

Display.blit(Credits17, (0,340))

Display.blit(Credits18, (0,355))

Display.blit(Credits19, (0,370))

Display.blit(Credits20, (0,385))

Display.blit(Credits21, (0,400))

Display.blit(Credits22, (0,415))

Display.blit(Credits23, (0,430))

Display.blit(Credits24, (0,445))

Display.blit(ChangeLog1, (0,470))

Display.blit(ChangeLog2, (0,485))

Display.blit(ChangeLog3, (0,500))

Display.blit(ChangeLog4, (0,515))

Display.blit(ChangeLog5, (0,530))

Display.blit(ChangeLog6, (0,545))

Display.blit(ChangeLog7, (0,560))

Display.blit(ChangeLog8, (0,575))

Display.blit(ChangeLog9, (0,590))

Display.blit(ChangeLog10, (0,605)) # END OF ADDING DATA TO THE DISPLAY

if devmode == 10 or devmode-10 == 0:

pygame.display.set\_caption(f"Pycraft: {version}: Credits and Change-Log | Developer mode | V: 0,0,0 | FPS: {clock.get\_fps()} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}")

else:

pygame.display.set\_caption(f"Pycraft: {version}: Credits and Change-Log")

if run >= 1000:

run = 0

rerun += 1

if rerun >= 1:

try:

data1[run] = ([((run/5)+1000), ((400-eFPS)-250)])

data2[run] = ([((run/5)+1000), ((400-((psutil.cpu\_percent())))-250)])

data3[run] = ([((run/5)+1000), ((100-psutil.virtual\_memory().percent)\*2)+25])

except:

rand = 0

else:

rand = 0

else:

data1.append([((run/5)+1000), ((400-eFPS)-250)])

data2.append([((run/5)+1000), ((400-((psutil.cpu\_percent())))-250)])

data3.append([((run/5)+1000), ((100-psutil.virtual\_memory().percent)\*2)+25])

if devmode == 10: # checks if devmode is equal to 10

dev\_Rect = pygame.Rect(1000,0,200, 200)

pygame.draw.rect(Display, (0,0,0), dev\_Rect)

if run >= 10:

pygame.draw.lines(Display, (0,255,0), False, (data2))

pygame.draw.lines(Display, (255,0,0), False, (data1))

pygame.draw.lines(Display, (0,0,255), False, (data3))

pygame.draw.line(Display, (255,255,255), (((run/5)+1000), 20), (((run/5)+1000), 200))

runFont = DataFont.render(f"{psutil.virtual\_memory().percent} | {str(psutil.cpu\_percent())} | {str(run)} | {str(rerun)} | {str(round(eFPS, 2))}", False, (255,255,255)) # stores the advanced data to be used when devmode is enabled

Display.blit(runFont, (1000,0)) # displays the data in the top left

pygame.display.flip() # updates the display

clock.tick(FPS) # limmists the FPS to 30

def Home\_Screen(devmode, rendis, FPS, FOV, cameraANGspeed, aa, RenderFOG, FanSky, FanPart, sound, soundVOL, music, MusicVOL): # creates the home screen module used after the startup

data = open("D:\\PYGAME\\Data\_Files\\data.txt","r+")

print(data.read())

if data.read() == "True":

contIG = 0

elif data.read() == "False":

Start()

else:

Start()

data.close()

if sys.platform == 'win32' or sys.platform == 'win64':

os.environ['SDL\_VIDEO\_CENTERED'] = '1'

Display = pygame.display.set\_mode((width, height)) # sets the window size

LoadingImage = pygame.image.load("D:\\PYGAME\\Resources\\General\_Resources\\Pycraft\_Short\_Loading.JPG") # loads the loading screen

Display.blit(LoadingImage, (0,0)) # renders the loading screen to the display at the position (0,0)

pygame.display.flip() # updates the display

if devmode == 10 or devmode-10 == 0: # if developer mode is enabled

FCol = (179, 204, 255) # sets the colour of the title font + devmode dependant features

pygame.display.set\_caption(f"Pycraft: {version}: Home Screen | Developer Mode | V: 0,0,0 | FPS: {clock.get\_fps()} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}") # sets the caption to developer mode for the user

else:# if the developer mode is not enabled then it is set to default

FCol = (255,255,255)

pygame.display.set\_caption(f"Pycraft: {version}: Home Screen") # sets the display caption

icon = pygame.image.load("D:\\PYGAME\\Resources\\General\_Resources\\Icon.jpg")

pygame.display.set\_icon(icon)

hover1 = False # sets the underline value of the first button font to false

hover2 = False # sets the underline value of the seccond button font to false

hover3 = False # sets the underline value of the third button font to false

hover4 = False # sets the underline value of the fourth button font to false

hover5 = False # sets the underline value of the fith button font to false

mousebuttondown = False # used to tell the if statements later on weather the mouse button is down or not

wallpaper = pygame.image.load("D:\\PYGAME\\Resources\\General\_Resources\\Pycraft\_Background.jpg") # loads the background image (feature coming here)

MainTitleFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 60) # loads the title / heading font

SideFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 24) # loads the "By Thomas Jebson" font

VersionFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 15) # loads the font that displays the version

ButtonFont1 = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 30) # play

ButtonFont2 = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 30) # settings

ButtonFont3 = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 30) # character custom

ButtonFont4 = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 30) # achievements

ButtonFont5 = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 30) # credits and change-log

DataFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 15) # loads the font used for the graph

data1 = [] # stores the FPS

data2 = [] # CPU Usage

data3 = [] # RAM usage

run = 0 # defines how many times the program has run

rerun = 0 # defines how many times the run has been greater than 2000

LoadingGameImage = pygame.image.load("D:\\PYGAME\\Resources\\General\_Resources\\Pycraft\_Long\_Loading.JPG")

if devmode == 10 or devmode-10 == 0: # if developer mode is enabled

FCol = (179, 204, 255) # sets the colour of the title font + devmode dependant features

pygame.display.set\_caption(f"Pycraft: {version}: Home | Developer Mode | V: 0,0,0 | FPS: {clock.get\_fps()} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}") # sets the caption to developer mode for the user

else:# if the developer mode is not enabled then it is set to default

FCol = (255,255,255)

pygame.display.set\_caption(f"Pycraft: {version}: Home") # sets the display caption

while True: # main game loop

eFPS = clock.get\_fps() # gets the refresh rate

Mx, My = pygame.mouse.get\_pos() # gets the mouse position

run += 1 # increases the run value by 1

rand = 0 # rand is a temporary variable use to store the number of errors in a loop

PycraftTitle = MainTitleFont.render("Pycraft", aa, (FCol)) # loads the title font text

Name = SideFont.render("By Thomas Jebson", aa, (FCol)) # loads the creator name text

Version = VersionFont.render(f"Version: {version}", aa, (FCol)) # loads the version text

Play = ButtonFont1.render("Play", aa, (255,255,255)) # loads the play text

Settings = ButtonFont2.render("Settings", aa, (255,255,255)) # loads the settings text

Character\_Customisations = ButtonFont3.render("Character Customisations", aa, (255,255,255)) # loads the char custom text

Achievements = ButtonFont4.render("Achievemets", aa, (255,255,255)) # loads the achievements text

Credits\_and\_Change\_Log = ButtonFont5.render("Credits and Change\_Log", aa, (255,255,255)) # loads the chedits font

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

quit()

elif event.type == pygame.KEYDOWN: # detects keypresses

if event.key == pygame.K\_SPACE and devmode < 10: # if developer mode is getting enabled...

devmode += 1 # increases the devmode value

if devmode >= 5 and devmode <= 9: # if devmode is getting enabled then

pygame.display.set\_caption(f"Pycraft: {version}: Home | you are: {10-devmode} steps away from being a developer") # tells the user that they are enabling devmode

elif devmode == 10: # if devmode is enabled then

pygame.display.set\_caption(f"Pycraft: {version}: Home | Developer mode | V: 0,0,0 | FPS: {clock.get\_fps()} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}") # tells the user

FCol = (220,220,255)

else:# if the developer mode is not enabled then it is set to default

FCol = (255,255,255) # sets the pycraft screen to white if not in devmode

pygame.display.set\_caption(f"Pycraft: {version}: Home") # sets the display caption

if event.key == pygame.K\_q: # detects if 'q' key pressed

DataWindow = tk.Tk() # sets the tkinter root

DataWindow.title("Player Information") # sets the display (window) caption

DataWindow.configure(width = 500, height = 300) # sets the window size (not needed (?))

DataWindow.configure(bg="lightblue") # sets the background colour

VersionData = f"Pycraft: {version}" # adds the curent version if caption fails

CoordinatesData = f"Coordinates: x: {Mx} y: {My} z: 0.0 FacinE: 0.0,0.0,0.0" # gives information about the coordinates of the user

FPSData = f"FPS: Actual: {eFPS} Max: {FPS}" # gives the FPS unsimplified before menu opens (pauses everything)

VersionData = tk.Label(DataWindow, text=VersionData) # loads this to the currently active Tk window

CoordinatesData = tk.Label(DataWindow, text=CoordinatesData) # loads the coordinates data to the GUI

FPSData = tk.Label(DataWindow, text=FPSData) # loads the fps data to the window

VersionData.grid(row = 0, column = 0, columnspan = 2) # coordinates are given

CoordinatesData.grid(row = 1, column = 0, columnspan = 2)

FPSData.grid(row = 2, column = 0, columnspan = 2)# END OF COORDINATES SPECIFIED

DataWindow.mainloop() # Tkinter will run what happens next until

DataWindow.quit() # it is ordered to quit

if event.key == pygame.K\_x: # detects if x key is pressed

devmode = 1 # resets devmode to 1

FCol = (255,255,255) # resets the font colour

pygame.display.set\_caption(f"Pycraft: {version}: Home") # and the caption

if event.type == pygame.MOUSEBUTTONDOWN: # if the mouse button down

mousebuttondown = True # mouse button down is set to True (yes)

if event.type == pygame.MOUSEBUTTONUP: # if the mouse button is up

mousebuttondown = False # this variable is set to no (False)

ButtonFont1.set\_underline(hover1) # applies an underline value to each button

ButtonFont2.set\_underline(hover2) # when hovering over it

ButtonFont3.set\_underline(hover3)

ButtonFont4.set\_underline(hover4)

ButtonFont5.set\_underline(hover5)

if devmode == 10 or devmode-10 == 0:

pygame.display.set\_caption(f"Pycraft: {version}: Home | Developer mode | V: 0,0,0 | FPS: {clock.get\_fps()} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}")

else:

pygame.display.set\_caption(f"Pycraft: {version}: Home")

# sets the hover value to True (underlined) when, (eh), hovering over the font

if My >= 202 and My <= 247:

hover1 = True

if mousebuttondown == True: # if the button is clicked

Display.blit(LoadingGameImage, (0,0)) # renders the loading screen

pygame.display.flip() # updates the display

if devmode == 10 or devmode-10 == 0:

pygame.display.set\_caption(f"Pycraft: {version}: Playing | Developer mode | V: 0,0,0 | FPS: {clock.get\_fps()} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}")

else:

pygame.display.set\_caption(f"Pycraft: {version}: Playing")

main(rendis, FPS, FOV, cameraANGspeed, devmode, aa, RenderFOG, Display, LoadingGameImage) # loads the command that happens when clicked

else: # if you are not hovering over the font

hover1 = False # hover os set to false

if My >= 252 and My <= 297: # this repeats for each button

hover2 = True

if mousebuttondown == True:

Display.blit(LoadingImage, (0,0))

pygame.display.flip()

rendis, FPS, FOV, cameraANGspeed, aa, RenderFOG, FanSky, FanPart, sound, soundVOL, music, MusicVOL = settings(rendis, FPS, FOV, cameraANGspeed, devmode, aa, RenderFOG, FanSky, FanPart, sound, soundVOL, music, musicVOL)

mousebuttondown = False

else:

hover2 = False

if My >= 302 and My <= 347:

hover3 = True

if mousebuttondown == True:

print("Character Custom")

Display.blit(LoadingImage, (0,0))

pygame.display.flip()

# Char\_Custom(FPS)

else:

hover3 = False

if My >= 402 and My <= 447:

hover4 = True

if mousebuttondown == True:

print("Achievements")

Display.blit(LoadingImage, (0,0))

pygame.display.flip()

#Achievements(FPS)

else:

hover4 = False

if My >= 352 and My <= 397:

hover5 = True

if mousebuttondown == True:

Display.blit(LoadingImage, (0,0))

pygame.display.flip()

Credits(devmode, aa)

else:

hover5 = False

Display.blit(wallpaper, (0,0)) # starts blitting images and font to the home screen

Display.blit(PycraftTitle, (500,0))

Display.blit(Name, (0,690))

Display.blit(Version, (1050, 700))

Display.blit(Play, (545, 200))

Display.blit(Settings, (525, 250))

Display.blit(Character\_Customisations, (430, 300))

Display.blit(Achievements, (490, 400))

Display.blit(Credits\_and\_Change\_Log, (440, 350))

if run >= 1000:

run = 0

rerun += 1

if rerun >= 1:

try:

data1[run] = ([((run/5)+1000), ((400-eFPS)-250)])

data2[run] = ([((run/5)+1000), ((400-((psutil.cpu\_percent())))-250)])

data3[run] = ([((run/5)+1000), ((100-psutil.virtual\_memory().percent)\*2)+25])

except:

rand = 0

else:

rand = 0

else:

data1.append([((run/5)+1000), ((400-eFPS)-250)])

data2.append([((run/5)+1000), ((400-((psutil.cpu\_percent())))-250)])

data3.append([((run/5)+1000), ((100-psutil.virtual\_memory().percent)\*2)+25])

if devmode == 10: # checks if devmode is equal to 10

dev\_Rect = pygame.Rect(1000,0,200, 200)

pygame.draw.rect(Display, (0,0,0), dev\_Rect)

if run >= 10:

pygame.draw.lines(Display, (0,255,0), False, (data2))

pygame.draw.lines(Display, (255,0,0), False, (data1))

pygame.draw.lines(Display, (0,0,255), False, (data3))

pygame.draw.line(Display, (255,255,255), (((run/5)+1000), 20), (((run/5)+1000), 200))

runFont = DataFont.render(f"{psutil.virtual\_memory().percent} | {str(psutil.cpu\_percent())} | {str(run)} | {str(rerun)} | {str(round(eFPS, 2))}", False, (255,255,255)) # stores the advanced data to be used when devmode is enabled

Display.blit(runFont, (1000,0)) # displays the data in the top left

pygame.display.flip()

clock.tick(FPS)

def Set\_Vertices(max\_distance,x,y,z): # adds a subprogram to define the vertices of a cube

x\_value\_change = x+x # appends the x value to the variable, x\_value\_change

y\_value\_change = y # adds the y value to the variable, y\_value\_change

z\_value\_change = z # adds the z value to the variable, z\_value\_change

new\_vertices = [] # creates a new empty array called new\_vertices

for vert in vertices: # repeats this loops for then lengh of the variable/array vertices

new\_vert = [] # creates another blank array 2D

new\_x, new\_y, new\_z = vert[0] + x\_value\_change, vert[1] + y\_value\_change, vert[2] + z\_value\_change # defines variables new\_x, new\_y, new\_z and runs an equation for each vertex

new\_vert.append(new\_x), new\_vert.append(new\_y), new\_vert.append(new\_z) # new vert is appended the results from the equations

new\_vertices.append(new\_vert) # appends the results to the array first defined

return new\_vertices # and outputs the results

def Ground(): # creates the "ground"

glBegin(GL\_QUADS) # opens an OpenGL file to store the data in the 3D "world"

x = 0 # creates a empty variable x

for vertex in ground\_vertices: # creates a rectangle with the points in the array ground\_vertices

x+=1 # x increases by 1 every time the loop runs

glColor3fv((0,0,255)) # sets the vertex point colour

glVertex3fv(vertex) # tells OpenGL the point / array

glEnd() # closes the openGL file opened on line 538

def Cube(vertices): # creates the "cubes"

glBegin(GL\_QUADS) # opens the previously open oGL file

for surface in surfaces: # at each face...

x = 0 # x is reset to zero

for vertex in surface: # at each corner

x+=1 # x is increased by an increment of 1

glColor3fv(colors[x]) # sets the colour of the vertice to the specified color

glVertex3fv(vertices[vertex]) # and all the data is sent to OpenGL in the array, glVertex3fv

glEnd() # closes the oGL file

'''glBegin(GL\_LINES) # hurts performance (50%) but creates the cube outline

for edge in edges: # repeats the next loop for each face in the object

for vertex in edge: # repeats the next loop for each edge

glVertex3fv(vertices[vertex]) # tells opengl the data needed to create the outline for the cube

glEnd()''' # closes the oGL file (oGL = openGL)

def MapModel(Map, Map\_scale, Map\_trans): # used to create the model

glPushMatrix() # creates a new matrix

glScalef(\*Map\_scale) # and creates a scale vector (\*Map\_scale)

glTranslatef(\*Map\_trans) # and translates the mesh into view

time = "day"

if time == "day":

glColor3fv((0,255,0))

elif time == "night":

glColor3fv((0,128,0))

for mesh in Map.mesh\_list: # then creates the mesh 3D locations

glBegin(GL\_TRIANGLES) # loads a oGL file for triangle handling

for face in mesh.faces: # then creates a face location in 3D locations in oGL "world"

for vertex\_i in face: # defines the vertex of each face

glVertex3f(\*Map.vertices[vertex\_i]) # tells OGL everything

glEnd() # closes the oGL file GL\_TRIANGLES

glPopMatrix() # adds the entire thing to the 3D "world"

'''def SunModel(Sun, Sun\_scale, Sun\_trans, Sun\_pos\_x, Sun\_pos\_y, Sun\_pos\_z): # used to create the model

glPushMatrix() # creates a new matrix

glScalef(\*Sun\_scale) # and creates a scale vector

glTranslatef(Sun\_pos\_x, Sun\_pos\_y, Sun\_pos\_z) # and translates the mesh into view (2Sun\_trans)

glColor3fv((255,255,0))

for mesh in Sun.mesh\_list: # then creates the mesh 3D locations

glBegin(GL\_TRIANGLES) # loads a oGL file for triangle handling

for face in mesh.faces: # then creates a face location in 3D locations in oGL "world"

for vertex\_i in face: # defines the vertex of each face

glVertex3f(\*Sun.vertices[vertex\_i]) # tells OGL everything

glEnd() # closes the oGL file GL\_TRIANGLES

glPopMatrix() # adds the entire thing to the 3D "world"'''

def CharacterModel(PlayerModel, PlayerModel\_scale, PlayerModel\_trans, PlayerModel\_pos\_x, PlayerModel\_pos\_y, PlayerModel\_pos\_z): # used to create the model

glPushMatrix() # creates a new matrix

glScalef(\*PlayerModel\_scale) # and creates a scale vector

glLoadIdentity()

glTranslatef(PlayerModel\_pos\_x, PlayerModel\_pos\_y, PlayerModel\_pos\_z) # and translates the mesh into view (2PlayerModel\_trans) (PlayerModel\_pos\_x, PlayerModel\_pos\_y, PlayerModel\_pos\_z)

glColor3fv((255,255,255))

for mesh in PlayerModel.mesh\_list: # then creates the mesh 3D locations

glBegin(GL\_TRIANGLES) # loads a oGL file for triangle handling

for face in mesh.faces: # then creates a face location in 3D locations in oGL "world"

for vertex\_i in face: # defines the vertex of each face

glVertex3f(\*PlayerModel.vertices[vertex\_i]) # tells OGL everything

glEnd() # closes the oGL file GL\_TRIANGLES

glPopMatrix() # adds the entire thing to the 3D "world

'''def LoadMapTexture():

Texture = Image.open("D:\\PYGAME\\Resources\\G3\_Resources\\map\\Grass 03 seamless.JPG")

MapTexture = Texture.tobytes()

glGenTextures(7)

glBindTexture(GL\_TEXTURE\_2D, 7)

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP) # glues the corners to the face vertices top left

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP) # glues the other side to the face bottom right.

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR) # makes sure the texture moves and rotates correctly

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR)

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, 512, 512, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, MapTexture) # blits the texture to the cube (renders)'''

def LoadSkyBox(aa):

if aa == True:

im1 = Image.open("D:\\PYGAME\\Resources\\G3\_Resources\\skybox\\front.jpg").rotate(180).transpose(Image.FLIP\_LEFT\_RIGHT).resize((512,512), Image.ANTIALIAS) # loads the image specified, rotates it, then flips it from left to right then resizes it to fit the cube (google a skybox if you don't understand)

texture1 = im1.tobytes() # converts the loaded and edited image to denary

im2 = Image.open("D:\\PYGAME\\Resources\\G3\_Resources\\skybox\\left.jpg").rotate(180).resize((512,512)) # loads the image specified and rotates it 180' and resizes it to fit the cube

texture2 = im2.tobytes() # and again.

im3 = Image.open("D:\\PYGAME\\Resources\\G3\_Resources\\skybox\\top.jpg").rotate(180).transpose(Image.FLIP\_LEFT\_RIGHT).resize((512,512), Image.ANTIALIAS)

texture3 = im3.tobytes()# and again..

im4 = Image.open("D:\\PYGAME\\Resources\\G3\_Resources\\skybox\\back.jpg").rotate(180).transpose(Image.FLIP\_LEFT\_RIGHT).resize((512,512), Image.ANTIALIAS)

texture5 = im4.tobytes() # and again...

im5 = Image.open("D:\\PYGAME\\Resources\\G3\_Resources\\skybox\\right.jpg").rotate(180).resize((512,512), Image.ANTIALIAS)

texture4 = im5.tobytes() # and again....

im6 = Image.open("D:\\PYGAME\\Resources\\G3\_Resources\\skybox\\bottom.jpg").resize((512,512), Image.ANTIALIAS)

texture6 = im6.tobytes() # and again.....

if aa == False:

im1 = Image.open("D:\\PYGAME\\Resources\\G3\_Resources\\skybox\\front.jpg").rotate(180).transpose(Image.FLIP\_LEFT\_RIGHT).resize((512,512)) # loads the image specified, rotates it, then flips it from left to right then resizes it to fit the cube (google a skybox if you don't understand)

texture1 = im1.tobytes() # converts the loaded and edited image to denary

im2 = Image.open("D:\\PYGAME\\Resources\\G3\_Resources\\skybox\\left.jpg").rotate(180).resize((512,512)) # loads the image specified and rotates it 180' and resizes it to fit the cube

texture2 = im2.tobytes() # and again.

im3 = Image.open("D:\\PYGAME\\Resources\\G3\_Resources\\skybox\\top.jpg").rotate(180).transpose(Image.FLIP\_LEFT\_RIGHT).resize((512,512))

texture3 = im3.tobytes()# and again..

im4 = Image.open("D:\\PYGAME\\Resources\\G3\_Resources\\skybox\\back.jpg").rotate(180).transpose(Image.FLIP\_LEFT\_RIGHT).resize((512,512))

texture5 = im4.tobytes() # and again...

im5 = Image.open("D:\\PYGAME\\Resources\\G3\_Resources\\skybox\\right.jpg").rotate(180).resize((512,512))

texture4 = im5.tobytes() # and again....

im6 = Image.open("D:\\PYGAME\\Resources\\G3\_Resources\\skybox\\bottom.jpg").resize((512,512))

texture6 = im6.tobytes() # and again.....

# clamps the texture 1 (img1, front) to the cube face

glGenTextures(1) # loads the texture

glBindTexture(GL\_TEXTURE\_2D, 1) # binds the texture to the cube

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1) # alligns the texture to the cube

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP) # glues the corners to the face vertices top left

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP) # glues the other side to the face bottom right.

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR) # makes sure the texture moves and rotates correctly

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR)

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, 512, 512, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, texture1) # blits the texture to the cube (renders)

# clamps the texture 2 (img2, left) to the correct face

glGenTextures(2)

glBindTexture(GL\_TEXTURE\_2D, 2)

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR)

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, 512, 512, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, texture2)

# clamps the texture 3 (img3, top) to the correct face

glGenTextures(3)

glBindTexture(GL\_TEXTURE\_2D, 3)

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR)

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, 512, 512, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, texture3)

# clams the texture 4 (img4, back) to the correct face

glGenTextures(4)

glBindTexture(GL\_TEXTURE\_2D, 4)

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR)

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, 512, 512, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, texture4)

# clamps the texture 5 (img 5, right) to the correct face

glGenTextures(5)

glBindTexture(GL\_TEXTURE\_2D, 5)

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR)

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, 512, 512, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, texture5)

# clamps the texture 6 (img6, bottom) to the correct face

glGenTextures(6)

glBindTexture(GL\_TEXTURE\_2D, 6)

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR)

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, 512, 512, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, texture6)

def DrawMapTexture():

glEnable(GL\_TEXTURE\_2D)

glDisable(GL\_DEPTH\_TEST)

glColorfv(1,1,1)

glBindTexture(GL\_TEXTURE\_2D, 7) # binds the texture to the ground

glBegin(GL\_QUADS) # opens the oGL file

glTexCoord2f(0, 0) # sets the image coords to 0,0

glVertex3f(-10.0, 0.0, -10.0) # loads the image at the points

glTexCoord2f(1, 0) # takes a 3D location and converts the coords into a 2D location

glVertex3f(10.0, 0.0, -10.0)

glTexCoord2f(1, 1)

glVertex3f(-10.0, 0.0, 10.0)

glTexCoord2f(0, 1)

glVertex3f(10.0, 0.0, 10.0)

glEnd() # closes the file

glBindTexture(GL\_TEXTURE\_2D, 0)

def DrawSkyBox():

glEnable(GL\_TEXTURE\_2D) # allows 2D images

glDisable(GL\_DEPTH\_TEST) # does not include the skybox into the translation vector

glColor3f(1,1,1) # sets the colour of all the surface to white (clears the cube of all texture)

# c1 Front

glBindTexture(GL\_TEXTURE\_2D, 1) # binds the texture to the cube

glBegin(GL\_QUADS) # opens the oGL file

glTexCoord2f(0, 0) # sets the image coords to 0,0

glVertex3f(-10.0, -10.0, -10.0) # loads the image at the points

glTexCoord2f(1, 0) # takes a 3D location and converts the coords into a 2D location

glVertex3f(10.0, -10.0, -10.0)

glTexCoord2f(1, 1)

glVertex3f(10.0, 10.0, -10.0)

glTexCoord2f(0, 1)

glVertex3f(-10.0, 10.0, -10.0)

glEnd() # closes the file

glBindTexture(GL\_TEXTURE\_2D, 0) # empties the RAM of the image as it is now stored in an oGL file

# c2 Left Side

glBindTexture(GL\_TEXTURE\_2D, 2)

glBegin(GL\_QUADS)

glTexCoord2f(0, 0)

glVertex3f(-10.0, -10.0, -10.0)

glTexCoord2f(1, 0)

glVertex3f(-10.0, -10.0, 10.0)

glTexCoord2f(1, 1)

glVertex3f(-10.0, 10.0, 10.0)

glTexCoord2f(0, 1)

glVertex3f(-10.0, 10.0, -10.0)

glEnd()

glBindTexture(GL\_TEXTURE\_2D, 0)

# c3 Top

glBindTexture(GL\_TEXTURE\_2D, 3)

glBegin(GL\_QUADS)

glTexCoord2f(0, 0)

glVertex3f(-10.0, 10.0, -10.0)

glTexCoord2f(1, 0)

glVertex3f(10.0, 10.0, -10.0)

glTexCoord2f(1, 1)

glVertex3f(10.0, 10.0, 10.0)

glTexCoord2f(0, 1)

glVertex3f(-10.0, 10.0, 10.0)

glEnd()

glBindTexture(GL\_TEXTURE\_2D, 0)

# c4 Right Side

glBindTexture(GL\_TEXTURE\_2D, 4)

glBegin(GL\_QUADS)

glTexCoord2f(0, 0)

glVertex3f(10.0, -10.0, 10.0)

glTexCoord2f(1, 0)

glVertex3f(10.0, -10.0, -10.0)

glTexCoord2f(1, 1)

glVertex3f(10.0, 10.0, -10.0)

glTexCoord2f(0, 1)

glVertex3f(10.0, 10.0, 10.0)

glEnd()

glBindTexture(GL\_TEXTURE\_2D, 0)

# c5 Back

glBindTexture(GL\_TEXTURE\_2D, 5)

glBegin(GL\_QUADS)

glTexCoord2f(0, 0)

glVertex3f(10.0, -10.0, 10.0)

glTexCoord2f(1, 0)

glVertex3f(-10.0, -10.0, 10.0)

glTexCoord2f(1, 1)

glVertex3f(-10.0, 10.0, 10.0)

glTexCoord2f(0, 1)

glVertex3f(10.0, 10.0, 10.0)

glEnd()

glBindTexture(GL\_TEXTURE\_2D, 0)

# c6 Bottom

glBindTexture(GL\_TEXTURE\_2D, 6)

glBegin(GL\_QUADS)

glTexCoord2f(0, 0)

glVertex3f(-10.0, -10.0, -10.0)

glTexCoord2f(1, 0)

glVertex3f(10.0, -10.0, -10.0)

glTexCoord2f(1, 1)

glVertex3f(10.0, -10.0, 10.0)

glTexCoord2f(0, 1)

glVertex3f(-10.0, -10.0, 10.0)

glEnd()

glBindTexture(GL\_TEXTURE\_2D, 0)

glEnable(GL\_DEPTH\_TEST)

def main(rendis, FPS, FOV, cameraANGspeed, devmode, aa, RenderFOG, Display, LoadingGameImage): # what requirements are needed to run this function, (are specified here)

LoadingPercent = 0

line = []

LoadingPercent += 100

line.append((LoadingPercent, 620))

LoadingPercent += 1

line.append((LoadingPercent, 620))

Display.blit(LoadingGameImage, (0,0))

pygame.draw.lines(Display, (153, 153, 153), False, line)

LoadingFont = pygame.font.Font("D:\\PYGAME\\Fonts\\Book Antiqua.ttf", 15)

Init = LoadingFont.render("Initiating", aa, (153, 153, 153))

Display.blit(Init, (600,640))

pygame.display.flip()

run2 = True # used in the (unimplemented) loading screen loading bar

percent = 0 # sets the loading percentage to 0%

Jump = False # tells the game weather to run

JumpID = 0 # part of the jump animation where the game makes sure that the camera returns to its original position

totalMoveX = 0# ?

CubeNum = 10 # the number of cubes allowed to be rendered into the game

FOV = 70 # the FOV of the player's camera

MouseUnlock = False # defines weather the mouse is tracked or not

CordX = 0 # sets the x position of the players head

CordY = 0 # sets the y position of the players head

CordZ = 0 # sets the z position of the players head

x\_move = 0 # ?

y\_move = 0 # ?

max\_distance = 0 # ?

z = 0 # ?

run = 0 # defines how mny times the game has been run

y = -10 # ?

z = 1 # ?

x = 0 # ?

changeCamX = 0 # ?

changeCamY = 0 # ?

changeCamZ = 0 # ?

cube\_dict = {}

display = (1200,720)

# pygame is initialised

# diaplay is created, 1200, 720

# the display is then configuered to allow OpenGL

LoadingPercent += 200

line.append((LoadingPercent, 620))

Display.blit(LoadingGameImage, (0,0))

pygame.draw.lines(Display, (153, 153, 153), False, line)

Init = LoadingFont.render("Loaded Variables", aa, (153, 153, 153))

Display.blit(Init, (600,640))

pygame.display.flip()

if devmode == 10 or devmode-10 == 0:

pygame.display.set\_caption(f"Pycraft: {version}: Playing | Developer mode | V: 0,0,0 | FPS: {clock.get\_fps()} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}")

else:

pygame.display.set\_caption(f"Pycraft: {version}: Playing")

MouseUnlock = False

pygame.mouse.set\_pos(600, 360)

Sun\_pos\_x, Sun\_pos\_y, Sun\_pos\_z = 0,10,-20

for j in range(CubeNum):

y += random.randint(-1,1)/10

cube\_dict[j] = Set\_Vertices(max\_distance,x,y,z)

if j >= 20:

x += 1

if x == 20:

x = 0

z += 2

else:

x = j

LoadingPercent += 200

line.append((LoadingPercent, 620))#

Display.blit(LoadingGameImage, (0,0))

pygame.draw.lines(Display, (153, 153, 153), False, line)

Init = LoadingFont.render("Loaded Logic", aa, (153, 153, 153))

Display.blit(Init, (600,640))

pygame.display.flip()

if devmode == 10 or devmode-10 == 0:

pygame.display.set\_caption(f"Pycraft: {version}: Playing | Developer mode | V: 0,0,0 | FPS: {clock.get\_fps()} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}")

else:

pygame.display.set\_caption(f"Pycraft: {version}: Playing")

#LoadMapTexture()

counter = 0

rotationvectX, rotationvectY = 0,0

# loads the map

pygame.event.get()

LoadingPercent += 100

line.append((LoadingPercent, 620))

Display.blit(LoadingGameImage, (0,0))

pygame.draw.lines(Display, (153, 153, 153), False, line)

Init = LoadingFont.render("Set Captions; Beginning Resource Loading", aa, (153, 153, 153))

Display.blit(Init, (600, 600))

pygame.display.flip()

Map = pywavefront.Wavefront("D:\\PYGAME\\Resources\\G3\_Resources\\map\\mapv2.obj", create\_materials=True, collect\_faces=True) # map v2.obj

Map\_box = (Map.vertices[0], Map.vertices[0])

for vertex in Map.vertices:

min\_v = [min(Map\_box[0][i], vertex[i]) for i in range(3)]

max\_v = [max(Map\_box[1][i], vertex[i]) for i in range(3)]

Map\_box = (min\_v, max\_v)

Map\_size = [Map\_box[1][i]-Map\_box[0][i] for i in range(3)]

max\_Map\_size = max(Map\_size)

Map\_size = 10000 # 10000

Map\_scale = [Map\_size/max\_Map\_size for i in range(3)]

Map\_trans = [-(Map\_box[1][i]+Map\_box[0][i])/2 for i in range(3)]

LoadingPercent += 200

Display.blit(LoadingGameImage, (0,0))

line.append((LoadingPercent, 620))

pygame.draw.lines(Display, (153, 153, 153), False, line)

Init = LoadingFont.render("Loaded Map", aa, (153, 153, 153))

Display.blit(Init, (600, 600))

pygame.display.flip()

if devmode == 10 or devmode-10 == 0:

pygame.display.set\_caption(f"Pycraft: {version}: Playing | Developer mode | V: 0,0,0 | FPS: {clock.get\_fps()} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}")

else:

pygame.display.set\_caption(f"Pycraft: {version}: Playing")

# loads the sun

pygame.event.get()

'''Sun = pywavefront.Wavefront("D:\\PYGAME\\Resources\\G3\_Resources\\Sun\\Sun.obj", create\_materials=True, collect\_faces=True)

Sun\_box = (Sun.vertices[0], Sun.vertices[0])

for vertex in Sun.vertices:

min\_v = [min(Sun\_box[0][i], vertex[i]) for i in range(3)]

max\_v = [max(Sun\_box[1][i], vertex[i]) for i in range(3)]

Sun\_box = (min\_v, max\_v)

Sun\_size = [Sun\_box[1][i]-Sun\_box[0][i] for i in range(3)]

max\_Sun\_size = max(Sun\_size)

Sun\_size = 1

Sun\_scale = [Sun\_size/max\_Sun\_size for i in range(3)]

Sun\_trans = [-(Sun\_box[1][i]+Sun\_box[0][i])/2 for i in range(3)]'''

LoadingPercent += 100

Display.blit(LoadingGameImage, (0,0))

line.append((LoadingPercent, 620))

pygame.draw.lines(Display, (153, 153, 153), False, line)

Init = LoadingFont.render("Loaded Sun", aa, (153, 153, 153))

Display.blit(Init, (600, 600))

pygame.display.flip()

pygame.event.get()

PlayerModel = pywavefront.Wavefront("D:\\PYGAME\\Resources\\G3\_Resources\\Player\\Man v2.obj", create\_materials=True, collect\_faces=True)

PlayerModel\_box = (PlayerModel.vertices[0], PlayerModel.vertices[0])

for vertex in PlayerModel.vertices:

min\_v = [min(PlayerModel\_box[0][i], vertex[i]) for i in range(3)]

max\_v = [max(PlayerModel\_box[1][i], vertex[i]) for i in range(3)]

PlayerModel\_box = (min\_v, max\_v)

PlayerModel\_size = [PlayerModel\_box[1][i]-PlayerModel\_box[0][i] for i in range(3)]

max\_PlayerModel\_size = max(PlayerModel\_size)

PlayerModel\_size = 10

PlayerModel\_scale = [PlayerModel\_size/max\_PlayerModel\_size for i in range(3)]

PlayerModel\_trans = [-(PlayerModel\_box[1][i]+PlayerModel\_box[0][i])/2 for i in range(3)]

LoadingPercent += 100

Display.blit(LoadingGameImage, (0,0))

line.append((LoadingPercent, 620))

pygame.draw.lines(Display, (153, 153, 153), False, line)

Init = LoadingFont.render("Loaded Player", aa, (153, 153, 153))

Display.blit(Init, (600, 600))

pygame.display.flip()

run = 0

pygame.event.get()

pygame.mouse.set\_pos(600,360)

Total\_move\_x, Total\_move\_y, Total\_move\_z = 0,0,0

if devmode == 10 or devmode-10 == 0:

pygame.display.set\_caption(f"Pycraft: {version}: Playing | Developer mode | V: 0,00 | FPS: {clock.get\_fps()} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}")

else:

pygame.display.set\_caption(f"Pycraft: {version}: Playing")

WKeyPressed, AKeyPressed, SKeyPressed, DKeyPressed = False, False, False, False

stop = False

stop1 = False

counterForWeather = 1

weather = 0

LoadingPercent += 100

Display.blit(LoadingGameImage, (0,0))

line.append((LoadingPercent, 620))

pygame.draw.lines(Display, (153, 153, 153), False, line)

Init = LoadingFont.render("Finished Resource Loading; Rendering", aa, (153, 153, 153))

Display.blit(Init, (600, 600))

pygame.display.flip()

pygame.display.set\_mode((width, height), DOUBLEBUF|OPENGL)

LoadSkyBox(aa)

# creates the perspective of the player, incl FOV, render-distance

gluPerspective(FOV, (display[0]/display[1]), 1, 1000000)

while True:

example\_FPS = clock.get\_fps()

if devmode == 10 or devmode-10 == 0:

pygame.display.set\_caption(f"Pycraft: {version}: Playing | Developer mode | V: {Total\_move\_x, Total\_move\_y, Total\_move\_z} | FPS: {round(example\_FPS,1)} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())}")

else:

pygame.display.set\_caption(f"Pycraft: {version}: Playing")

mX, mY = pygame.mouse.get\_pos()

x = glGetDoublev(GL\_MODELVIEW\_MATRIX)

camera\_x = x[3][0]

camera\_y = x[3][1]

camera\_z = x[3][2]

run += 1

counter += 1

# pygame events

for event in pygame.event.get():

if event.type == pygame.QUIT:

StartMenu = pygame.display.set\_mode((width, height))

LoadingScreen = pygame.image.load("D:\\PYGAME\\Resources\\General\_Resources\\Pycraft\_Short\_Loading.JPG")

StartMenu.blit(LoadingScreen, (0,0))

pygame.display.flip()

Home\_Screen(devmode, rendis, FPS, FOV, cameraANGspeed, aa, RenderFOG, FanSky, FanPart, sound, soundVOL, music, musicVOL)

# get keypresses

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_a:

AKeyPressed = True

if event.key == pygame.K\_d:

DKeyPressed = True

if event.key == pygame.K\_e:

import Inventory

if event.key == pygame.K\_w:

WKeyPressed = True

if event.key == pygame.K\_s:

SKeyPressed = True

if event.key == pygame.K\_SPACE and Jump == False:

Jump = True

if event.key == pygame.K\_ESCAPE:

# locks and unlocks the mouse to the canvas

if MouseUnlock == True:

MouseUnlock = False

elif MouseUnlock == False:

MouseUnlock = True

if event.key == pygame.K\_q:

# setup extra window, root, title, size, background colour

DataWindow = tk.Tk()

DataWindow.title("Player Information")

DataWindow.configure(width = 500, height = 300)

DataWindow.configure(bg="lightblue")

# data is defined so it can be written to the display

versionData = "Pycraft: 27p085-20a: Playing"

versionData, CoordinatesData, FPSData, RendisData, FOVData = f"Pycraft: {version}", f"Coordinates: x: {round(camera\_x,3)} y: {round(camera\_y,3)} z: {round(camera\_z, 3)}", f"FPS: {example\_FPS}", f"Render distance: {rendis}", f"FOV: {FOV}"

# this data is then stored as a tkinter label

versionData, CoordinatesData, FPSData, RendisData, FOVData = tk.Label(DataWindow, text = versionData), tk.Label(DataWindow, text = CoordinatesData), tk.Label(DataWindow, text = FPSData), tk.Label(DataWindow, text = RendisData), tk.Label(DataWindow, text = FOVData)

# which is then rendered on the display at the required locations

versionData.grid(row = 0, column = 0, columnspan = 2)

CoordinatesData.grid(row = 1, column = 0, columnspan = 2)

FPSData.grid(row = 2, column = 0, columnspan = 2)

RendisData.grid(row = 3, column = 0, columnspan = 2)

FOVData.grid(row = 4, column = 0, columnspan = 2)

DataWindow.mainloop()

DataWindow.quit()

pygame.mouse.set\_pos(600,360)

if event.type == pygame.KEYUP:

if event.key == pygame.K\_w:

WKeyPressed = False

if event.key == pygame.K\_a:

AKeyPressed = False

if event.key == pygame.K\_s:

SKeyPressed = False

if event.key == pygame.K\_d:

DKeyPressed = False

# zoom / change FOV

if event.type == pygame.MOUSEBUTTONDOWN:

if event.button == 4:

glTranslatef(0,0,1.0)

FOV += 1

if event.button == 5:

glTranslatef(0,0,-1.0)

FOV -= 1

if WKeyPressed == True:

if stop == False:

time = example\_FPS\*3

stop = True

if time >= 0:

Total\_move\_z += 3.5

elif time <= 0:

Total\_move\_z += 6

time -= 1

else:

stop = False

if AKeyPressed == True:

Total\_move\_x += -3.5

if SKeyPressed == True:

Total\_move\_z += -3.5

if DKeyPressed == True:

Total\_move\_x += 3.5

# jump animation

if Jump == True:

JumpID += 1

if JumpID <= 20:

JumpID += 1

Total\_move\_y -= 0.01

if JumpID >= 21:

JumpID += 1

Total\_move\_y += 0.01

if JumpID >= 40:

Jump = False

JumpID = 0

if MouseUnlock == False:

if mX >= 620:

glRotatef(cameraANGspeed,0,1,0) # 0.5,0,1,0

rotationvectX += 0.5

# auto-loop

if mX <= 580:

glRotatef(-cameraANGspeed,0,1,0) # -0.5,0,1,0

rotationvectX += -0.5

# auto-loop

if mY <= 380 and rotationvectY < 50:

glRotatef(-cameraANGspeed,1,0,0) # -0.5,1,0,0

rotationvectY += 0.5

#up

if mY >= 340 and rotationvectY > -50:

glRotatef(cameraANGspeed,1,0,0) # 0.5,1,0,0

rotationvectY += -0.5

'''if run2 == True:

percent += 20

pygame.display.set\_caption(f"Pycraft: {version}: Playing, {percent}%")'''

'''if int(Sun\_pos\_y) >= 0 and int(Sun\_pos\_y) <= 100 and Temp1 == 0:

Sun\_pos\_y += 0.1

if int(Sun\_pos\_y) == 100:

Temp1 = 1

elif int(Sun\_pos\_y) >= 0 and int(Sun\_pos\_y) <= 100 and Temp1 == 1:

Sun\_pos\_y -= 0.1

if int(Sun\_pos\_y) == 0:

Temp1 = 0

if int(Sun\_pos\_x) >= 0 and int(Sun\_pos\_x) <= 100 and Temp2 == 0:

Sun\_pos\_x += 0.1

if int(Sun\_pos\_x) == 100:

Temp2 = 1

elif int(Sun\_pos\_x) >= 0 and int(Sun\_pos\_x) <= 100 and Temp2 == 1:

Sun\_pos\_x -= 0.1

if int(Sun\_pos\_x) == 0:

Temp2 = 0'''

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT)

x = glGetDoublev(GL\_MODELVIEW\_MATRIX)

glDisable(GL\_DEPTH\_TEST)

glPushMatrix()

glDepthMask(GL\_FALSE)

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT)

glTranslatef(-camera\_x,camera\_y,camera\_z)

glDisable(GL\_DEPTH\_TEST)

DrawSkyBox()

glEnable(GL\_DEPTH\_TEST)

glDepthMask(GL\_TRUE)

glPopMatrix()

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL)

if devmode == 10 or devmode-10 == 0:

pygame.display.set\_caption(f"Pycraft: {version}: Playing | Developer mode | V: {Total\_move\_x, Total\_move\_y, Total\_move\_z} | FPS: {round(example\_FPS,1)} | MemUsE: {psutil.virtual\_memory().percent} | CPUUsE: {str(psutil.cpu\_percent())} | weather: {weather} changeIN: {round(counterForWeather/FPS)}/300")

else:

pygame.display.set\_caption(f"Pycraft: {version}: Playing")

glTranslatef(Total\_move\_x, Total\_move\_y, Total\_move\_z)

PlayerModel\_pos\_x, PlayerModel\_pos\_y, PlayerModel\_pos\_z = -Total\_move\_x, -Total\_move\_y, -Total\_move\_z

Total\_move\_x, Total\_move\_y, Total\_move\_z = 0,0,0

MapModel(Map, Map\_scale, Map\_trans)

glDisable(GL\_DEPTH\_TEST)

CharacterModel(PlayerModel, PlayerModel\_scale, PlayerModel\_trans, PlayerModel\_pos\_x, PlayerModel\_pos\_y, PlayerModel\_pos\_z)

glEnable(GL\_DEPTH\_TEST)

#SunModel(Sun, Sun\_scale, Sun\_trans, Sun\_pos\_x, Sun\_pos\_y, Sun\_pos\_z)

depthMapFBO = ""

glGenFramebuffers(1, depthMapFBO)

Ground()

if stop1 == False:

counterForWeather = 1

stop1 = True

counterForWeather = counterForWeather + 1

if counterForWeather/FPS >= 300:

weather = random.randint(0,2)

stop1 = False

if RenderFOG == True and weather == 2:

glEnable(GL\_FOG)

glFogfv(GL\_FOG\_COLOR,(0.8,0.8,1,1)) # 0.5, 0.5, 0.7, 1

glFogi(GL\_FOG\_MODE, GL\_LINEAR)

glFogf(GL\_FOG\_START, 160) # 160

glFogf(GL\_FOG\_END, 300) # 2000

glFogf(GL\_FOG\_DENSITY, 0.5) # 0.35

elif RenderFOG == True and weather == 1:

glEnable(GL\_FOG)

glFogfv(GL\_FOG\_COLOR,(0.5,0.5,1,1)) # 0.5, 0.5, 0.7, 1

glFogi(GL\_FOG\_MODE, GL\_LINEAR)

glFogf(GL\_FOG\_START, 160) # 160

glFogf(GL\_FOG\_END, 900) # 2000

glFogf(GL\_FOG\_DENSITY, 0.5) # 0.35

elif RenderFOG == True and weather == 0:

glEnable(GL\_FOG)

glFogfv(GL\_FOG\_COLOR,(0.5,0.5,0.7,1)) # 0.5, 0.5, 0.7, 1

glFogi(GL\_FOG\_MODE, GL\_LINEAR)

glFogf(GL\_FOG\_START, 160) # 160

glFogf(GL\_FOG\_END, 3000) # 2000

glFogf(GL\_FOG\_DENSITY, 0.5) # 0.35

elif RenderFOG == False:

glDisable(GL\_FOG)

glEnable(GL\_LIGHTING)

glEnable(GL\_LIGHT0)

glLightfv(GL\_LIGHT0,GL\_POSITION,(Sun\_pos\_x, Sun\_pos\_y, Sun\_pos\_z))

glLightfv(GL\_LIGHT0,GL\_AMBIENT,(1,0,1,1))

glLightfv(GL\_LIGHT0,GL\_DIFFUSE,(1,0,1,1))

glLightfv(GL\_LIGHT0,GL\_SPECULAR,(1,0,1,1))

glEnable(GL\_COLOR\_MATERIAL)

glColorMaterial(GL\_FRONT\_AND\_BACK,GL\_AMBIENT\_AND\_DIFFUSE)

glMaterial(GL\_FRONT\_AND\_BACK,GL\_SPECULAR,(0,1,0,1))

glMaterial(GL\_FRONT\_AND\_BACK,GL\_EMISSION,(0,1,0,1))

glEnable(GL\_FRAMEBUFFER\_SRGB)

if aa == True:

glEnable(GL\_MULTISAMPLE)

elif aa == False:

glDisable(GL\_MULTISAMPLE)

glShadeModel(GL\_SMOOTH)

pygame.display.flip()

'''myScreenshot = pyautogui.screenshot()

myScreenshot.save(r"E:\img.png")'''

clock.tick(FPS)

Home\_Screen(devmode, rendis, FPS, FOV, cameraANGspeed, aa, RenderFOG, FanSky, FanPart, sound, soundVOL, music, musicVOL)