Algorithm’s

ScreenMsg(screen,x,y,font,text,colour)

**BEGIN**

Text=font.render(text,colour)

Screen.blit(text,(x,y))

**END**

RectButton(screen,x,y,w,h,colour,events,text,font,fcolour)

**BEGIN**

Text=font.render(text,fcolour)

Draw.rect(screen,colour,(x,y,w,h))

rect=Rect(x,y,w,h)

screen.blit(text,(x+w/2-text.width()/2,y+h/2-text.height()/2))

**FOR** event=events[i] **TO** length(events):

**IF** event.type == Mouseclick():

**IF** rect.collision(mouse.pos())=True:

**RETURN** True

**ENDIF**

**ENDIF**

**NEXT i**

**RETURN** False

**END**

ImgButton(screen,img,x,y,events)

**BEGIN**

Screen.blit(img,(x,y))

Rect = py.rect(x,y,img.width(),img.height())

**FOR** event=events[i] **TO** length(events):

**IF** event.type == Mouseclick():

**IF** rect.collision(mouse.pos())=True:

**RETURN** True

**ENDIF**

**ENDIF**

**NEXT i**

**END**

Selectlevel()

**BEGIN**

Tick(10)  
 events=py.event.get()

**FOR** event=events[i] **TO** length(events):

**IF** event.type == Quit():

Pygame.quit()

**ENDIF**

**NEXT i**

Screen.fill(GRAY)

Counter=0

X=0

Y=0

**WHILE** X != 3:

**WHILE** Y != 5:

Counter=Counter+1

Click=rectbutton(screen,175+160\*y,300+100\*x,50,50,BLACK,events,str(cou),myfont,WHITE)

**IF** Click == True:

RETURN(Counter)

**END IF**

**END WHILE**

**END WHILE**

Pygame.display.flip()

**END**

This algorithm does the main game logic and also the

Gamescreen(LevelNum, LevImageRes, LevImagePlay)

**BEGIN**

Tick(100)

Global PrevClickedResImg

Global PrevClickedRes

Global CorrectSpot

Global Score

events=py.event.get()

**FOR** event=events[i] **TO** length(events):

**IF** event.type == Quit():

Pygame.quit()

**ENDIF**

**NEXT i**

Screen.fill(GRAY)

Playcount=0

Tog1 = False

J1=0

J2=0

**While** J1 != 5:

**While** J2 != 5:

IF LevImagePlay[Playcount].img\_selected == False:

Tog1=imgbutton(screen,LevImagePlay[Playcount].image[LevImagePlay[Playcount].img\_count],LevImagePlay[Playcount].x,LevImagePlay[Playcount].y,events

**END IF**

**IF** LevImagePlay[Playcount].img\_selected == True:

tog1=imgbutton(screen,GreyBox,LevImagePlay[Playcount].x,LevImagePlay[Playcount].y,events)

**END IF**

**IF**  Tog1 == True:

if LevImagePlay[Playcount].img\_count == prevclickedResImg:

LevImagePlay[Playcount].img\_selected = False

LevImageRes[prevclickedRes].correct=True

Correctspot[prevclickedRes]=True

score=score+10

**END IF**

**ELSE**:

score=score-10

**END ELSE**

J2=J2+1

**END WHILE**

J1=J1+2

**END WHILE**

Tog=False

Coun=0

I1 =0

I2=0

**WHILE** I1 != 5:

**WHILE** I2 != 5:

**IF** LevImageRes[coun].img\_selected == False:

tog=imgbutton(screen,LevImageRes[coun].image[LevImageRes[coun].img\_count],LevImageRes[coun].x,LevImageRes[coun].y,events)

**END IF**

**IF** LevImageRes[coun].img\_selected == True **OR** LevImageRes[coun].correct==True:

screen.blit(GreyBox,(LevImageRes[coun].x,LevImageRes[coun].y))

**END IF**

**IF** tog == True:

LevImageRes[prevclickedRes].img\_selected = False

prevclickedRes=coun

prevclickedResImg=LevImageRes[coun].img\_count

LevImageRes[coun].img\_selected = True

**END IF**

I2=I2+1

**END WHILE**

I1=I1+1

**END WHILE**

ScreenMsg(screen,400,100,yfont,”Logic Puzzler”,ORANGE)

ScoreMsg=”Score: “ +score

ScreenMsg(screen,100,100,myfont,ScoreMsg,BLACK)

Py.display.flip()

**IF** all(correctspot) == True:

**IF** PrevClickedRes == 26:

**RETURN**(Score)

**END IF**

PrevclickedRes = 26

**END IF**

**RETURN**(-100000000000)

**END**

Scorescreen(score,scorel)

**BEGIN**

Tick(10)

**FOR** event=events[i] **TO** length(events):

**IF** event.type == Quit():

Pygame.quit()

**ENDIF**

**NEXT i**

Screen.fill(GRAY)

I=0

**WHILE** I != 5:

screenMsg(screen,100,100\*I,yfont,scorel[i],RED)

i=i+1

**END WHILE**

BTS=Rectbutton(screen,20,10,50,15,BLACK,events,”Back to Level’s”,yfont,ORANGE)

**IF** BTS == True:

**RETURN**(“BackToLevel”)

**END IF**

**END**

This algorithm controls all the logic for switching screens and setting up the necessary data structures for the different screens.

**Main()**

**BEGIN**

**IF** event == quit:

Pygame.quit

**END IF**

Txtbx.update(event)

Txtbx.draw(screen)

Name=txtbx.value

Start=imgbutton(screen,startbutt,350,100,events)

**IF** start == True **AND** name != “”:

Select=True

**END IF**

**IF**  start == True **OR** errorcount[0] == True **AND** name == “” and errorcount[0] != 100:

Errorcount[1] = True

screenMsg(screen, screen\_width/2, screen\_height/2, yfont, “INPUT A NAME”, ORANGE)

errorcount+=1

**END IF**

**if** errorcount[0] == 1000 **AND** errorcount[1] == True:

errorcount[0]=0

errorcount[1]=False

**END IF**

**WHILE** select **AND** levnum == 16:  
 **IF** event == quit:

Pygame.quit

**END IF**

Levnum=Selectlevel()

**END WHILE**

**IF** levnum != 16:

**DEFINE** imcount as an array with 25 incremented integers in it.

Shuffle imcount

Count=0

**FOR** i **TO** 5:

**FOR** j **TO** 5:

Levimagetemp=Images()

Add image to LevImageTemp.image

Set LevImageTemp.imagecount to count

Increment LevImageTemp.x with offset of 51

Increment LevImageTemp.y with offset of 51 plus 440

LevImageTemp.img\_selected=False

LevImageTemp add to Levimageres

Count+=1

**END FOR**

**END FOR**

Count=0

**FOR** i = 0 **TO** 5:

**FOR** j = 0 **TO** 5:

LevImageTemp=images()

Add image to LevImageTemp.image

Set LevImageTemp.imagecount to count

Increment LevImageTemp.x with offset of 51 plus 400

Increment LevImageTemp.y with offset of 51 plus 440

LevImageTemp.img\_selected=False

Add Levimagetemp to Levimageply

**END FOR**

**END FOR**

**WHILE** lvnum:

Gs=Gamescreen(Levnum,Levimageres,Levimageply)

**IF** gs != -100000:

Lvum=False

Gsgo=True

Create an array of arrays from the Scores.csv file as “Data”

Check if the score is higher than atleast one of the existing scores, if so replace it and move on to next line

Save data to Scores.csv

**END IF**

**WHILE**  GSGO=true:

Scorescreen()

Update the game screen

**END**

Mainline()

**BEGIN**

set framerate to 100

**IF** event = Quit:

Pygame.quit

**END IF**

Main()

**DATA DICTIONARY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DATA NAME** | **DATA TYPE** | **SIZE** | **SCOPE** | **DESCRIPTION** |
| Screen\_height | Integer |  | GLOBAL | Defines screen height |
| Screen\_width | integer |  | GLOBAL | Defines screen width |
| Screen | Pygame display object |  | GLOBAL | Is the screen object |
| Lock | Pygame Clock object |  | GLOBAL | Sets the clock object in order to set framerate |
| Running | Boolean |  | GLOBAL | Defines whether the program runs or doesn’t |
| Txtbx | Eztext text input object |  | GLOBAL | Allows capture of user text input |
| Startbutt | Pygame image object |  | GLOBAL | Image for the start button |
| RED | Tuple |  | GLOBAL | Easy to call for the colour red |
| BLUE | Tuple |  | GLOBAL | Easy to call for the colour blue |
| GREEN | Tuple |  | GLOBAL | Easy to call for the colour green |
| BLACK | Tuple |  | GLOBAL | Easy to call for blacl |
| WHITE | Tuple |  | GLOBAL | Easy to call for white |
| ORANGE | Tuple |  | GLOBAL | Easy to call for the colour orange |
| prevclickedRes | Integer |  | GLOBAL | Used for caching the latest clicked resource image |
| prevclickedPlay | Integer |  | GLOBAL | Used for caching the latest clicked play image |
| Correctspot | Array |  | GLOBAL | Stores a list of images in the correctspot |
| Score | Integer |  | GLOBAL | Stores the current user score |
| Name | String |  | GLOBAL | Stores the user’s name |
| GreyBox | Pygame image object |  | GLOBAL | Used as a placeholder for a clicked image |
| Errorcount | Array |  | GLOBAL | Used for checking if an error should be displayed or not |
| Xv | Text object |  | LOCAL to screenmsg func | Text object to be blitted |
| Rect | Rectangle object |  | LOCAL to rect button func | Allows for easier collision detection |
| Butt | Rectangle object |  | LOCAL to imgbutton func | Allows for easier collision detection |
| Events | Pygame event queue |  | LOCAL to selectlevel | Allows for getting the event queue |
| Cou | Integer |  | LOCAL to selectlevel | Counts the amount of times the for loops have looped |
| Events | Pygame event queue |  | LOCAL to Gamescreen | Allows for getting the event queue |
| Playcount | Integer |  | LOCAL to Gamescreen | counts the amount of times the nested for loops have looped |
| LevImagePlay | Array of Records |  | LOCAL to Gamescreen | Stores information and images required for the logic to work |
| Tog1 | Boolean |  | LOCAL to Gamescreen | Is assigned to a imgbutton object to let each of the images be buttons |
| Coun | Integer |  | LOCAL to Gamescreen | counts the amount of times the nested for loops have looped |
| Tog | Boolean |  | LOCAL to Gamescreen | Is assigned to a imgbutton object to let each of the images be buttons |
| ScoreMsg | String |  | LOCAL to Gamescreen | Is the string that is used in making scomsg |
| Scomsg | Pygame text object |  | LOCAL to Gamescreen | Is the text object that is blitted to the screen |
|  |  |  | 88888888888888888 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |