

1. (25 points) Figure 1 shows two globes. Listed below is most of the code needed to make the two globes scroll across the screen. Modify the code in PinkGlobe so that the pink globe moves in a circle around the blue globe. (Note: Most import statements have been elided due to space constraints)

```
class Manager(Singleton, pygame.sprite.Sprite):
    def __init__(self):
        pygame.init()
        self.screen = pygame.display.set_mode((640, 480))
        self.backGroup = pygame.sprite.Group()
        self.backGroup.add( Background() )
        self.spriteGroup = pygame.sprite.Group()
        globe = Globe(0.14, 200)
        self.spriteGroup.add( globe )
        self.spriteGroup.add( PinkGlobe(70, 0.1, globe) )

    def play(self):
        done = False
        while not done:
            for event in pygame.event.get():
                if event.type == QUIT:
                    done = True
                elif event.type == KEYDOWN and event.key == K_ESCAPE:
                    done = True
            self.backGroup.update()
            self.spriteGroup.update()
            self.backGroup.draw(self.screen)
            self.spriteGroup.draw(self.screen)
            pygame.display.flip()

class Globe(pygame.sprite.Sprite):
    def __init__(self, speed, y):
        pygame.sprite.Sprite.__init__(self)
        self.image, self.rect = load_image('globe.png', -1)
        self.image = pygame.transform.flip(self.image, 1, 0)
        screen = pygame.display.get_surface()
        self.area = screen.get_rect()
        self.rect.topleft = (0, y)
        self.speed = speed
        self.start_ticks = pygame.time.get_ticks()
    def update(self):
        total_ticks = pygame.time.get_ticks() - self.start_ticks
        left = (self.speed * total_ticks ) % \
            (self.area.width + self.rect.width) - self.rect.width
        self.rect.left = left

class Background(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image, self.rect = load_image('cliffs.bmp', None)
        screen = pygame.display.get_surface()
        self.area = screen.get_rect()
        self.rect.topleft = (0, 0)
    def update(self): pass
```

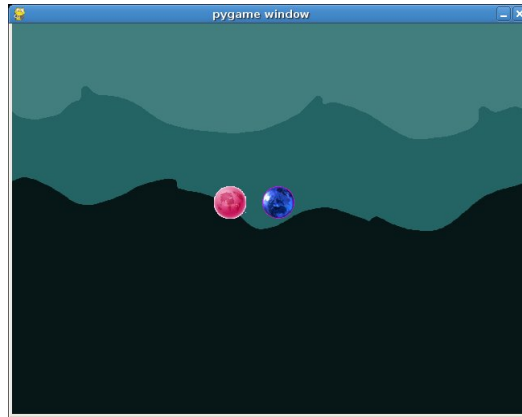


Figure 1: This figure depicts two globes moving across the screen. Globe on left is pink; globe on right is blue.

```
import math
class PinkGlobe(pygame.sprite.Sprite):
    def __init__(self, radius, speed, globe):
        pygame.sprite.Sprite.__init__(self)
        self.image, self.rect = load_image('pinkglobe.png', -1)
        self.image = pygame.transform.flip(self.image, 1, 0)
        screen = pygame.display.get_surface()
        self.screenRect = screen.get_rect()
        self.blueGlobe = globe
        self.x, self.y = (self.blueGlobe.rect.center)
        self.speed = speed
        self.curr_ticks = pygame.time.get_ticks()

    def update(self):
        prev_ticks = self.curr_ticks
        self.curr_ticks = pygame.time.get_ticks()
        ticks = self.curr_ticks - prev_ticks
        incr = self.speed * ticks
        self.x += (self.speed * ticks)
        self.rect.center = (self.x, self.y)
```

2. (25 points) Listed below is the `Manager` class from the previous question, with three lines added to incorporate a scoreboard into the animation. Also, a `Scoreboard` class is listed below, which should list the number of frames. (a) The number of frames is not appearing on the screen. Explain why, and modify the code so that the number of frames does appear on the screen. (b) Modify the code so that your program prints on three lines, (1) the number of elapsed seconds, (2) the (x, y) coordinates of the pink globe, and (3) the (x, y) coordinates of the blue globe.

```
class Manager(Singleton, pygame.sprite.Sprite):
    def __init__(self):
        pygame.init()
        self.screen = pygame.display.set_mode((640, 480))
        self.backGroup = pygame.sprite.Group()
        self.backGroup.add( Background() )
        self.spriteGroup = pygame.sprite.Group()
        globe = Globe(0.14, 200)
        self.spriteGroup.add( globe )
        self.spriteGroup.add( PinkGlobe(70, 0.1, globe) )
        # The following two lines added
        self.scoreBoard = ScoreBoard()
        self.scoreBoard.update()

    def play(self):
        done = False
        while not done:
            for event in pygame.event.get():
                if event.type == QUIT:
                    done = True
                elif event.type == KEYDOWN and event.key == K_ESCAPE:
                    done = True
            self.backGroup.update()
            self.spriteGroup.update()
            # The following line added
            self.scoreBoard.draw(self.screen)
            self.backGroup.draw(self.screen)
            self.spriteGroup.draw(self.screen)
            self.scoreBoard.update()
            pygame.display.flip()

class ScoreBoard(Singleton, pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        pygame.init()
        self.startTicks = pygame.time.get_ticks()
        self.lastTicks = 0
        self.frames = 0
        font = pygame.font.get_default_font()
        self.theFont = pygame.font.SysFont(font, 30)
        self.image = None

    def draw(self, screen):
        screen.blit(self.image, (10, 10))

    def update(self):
        self.frames += 1
        self.image = self.theFont.render(str(self.frames), 1, (255, 255, 255))
```

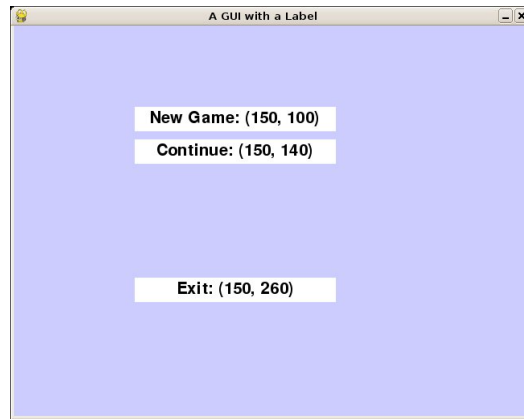


Figure 2: This figure illustrates a gui with three buttons.

3. (25 points) Figure 2 illustrates a gui menu with three buttons. The code listed on this page implements a Label class, and a Button class that inherits attributes of the Label class. The code on the next page illustrates a GuiManager that puts two buttons on the screen.

- (a) The Exit button causes the program to terminate. Explain how this Exit button works.
- (b) Add code to GuiManager so that the third button shown in Figure 2 appears; also, this third button should, when pressed, cause the background to change to magenta (255,0,255).

```
class Label(pygame.sprite.Sprite):
    def __init__(self, txt, size, topleft):
        pygame.sprite.Sprite.__init__(self)
        self.font = pygame.font.SysFont(None, 30)
        self.text = txt
        self.size = size
        self.image = pygame.Surface(self.size)
        self.fgColor = ((0, 0, 0))
        self.bgColor = ((255, 255, 255))
        self.image.fill(self.bgColor)
        self.rect = self.image.get_rect()
        self.rect.topleft = topleft
        self.fontSurface = self.font.render(self.text, True, self.fgColor)
        #center the text
        self.xPos = (self.image.get_width() - self.fontSurface.get_width())/2
        self.yPos = (self.image.get_height() - self.fontSurface.get_height())/2
        self.image.blit(self.fontSurface, (self.xPos, self.yPos))
    def update(self): pass

class Button(Label):
    def __init__(self, txt, size, topleft, callback = None):
        Label.__init__(self, txt, size, topleft)
        self.active = False
        self.callback = callback

    def update(self):
        self.active = False
        if pygame.mouse.get_pressed() == (1, 0, 0):
            if self.rect.collidepoint(pygame.mouse.get_pos()):
                self.active = True
                if self.callback: self.callback()
```

```

def quit(): exit()

class GuiManager(pygame.sprite.Sprite):
    def __init__(self):
        self.screen = pygame.display.set_mode((640, 480))
        pygame.display.set_caption("A GUI with a Label")

        self.background = pygame.Surface(self.screen.get_size())
        self.lightblue = (204, 204, 255)
        self.red = (255, 0, 0)
        self.background.fill( self.lightblue )

        self.sprites = pygame.sprite.Group()
        self.button1 = Button( "New Game: (150, 100)", (250, 30), (150, 100) )
        self.button3 = Button( "Exit: (150, 260)", (250, 30), (150, 310), quit )
        self.sprites.add(self.button1)
        self.sprites.add(self.button3)

    def play(self):
        clock = pygame.time.Clock()
        done = False
        while not done:
            clock.tick(60)
            for event in pygame.event.get():
                if event.type == QUIT: done = True
                elif event.type == KEYDOWN and event.key == K_ESCAPE:
                    done = True
            if self.button1.active :
                self.background.fill(self.red)
            else:
                self.background.fill( self.lightblue )
            self.screen.blit(self.background, (0, 0))
            self.sprites.clear(self.screen, self.background)
            self.sprites.update()
            self.sprites.draw(self.screen)
            pygame.display.flip()

```

4. (25 points) The typical approach utilized to incorporate levels into a video game is to use a stack. The various levels are pushed onto the stack and, to play a level, the function(s) that are required to play the level are invoked for the level at the top of the stack. Then, when a level is completed, it is popped off of the stack and the process of calling the function(s) that are required to play the level at the top of the stack are, once again, invoked.

- (a) Write a class `PlayLevels` that uses a list to implement a stack of levels. The `append` and `pop` methods in list can be used to implement you stack: `append` will push onto the top of the stack and `pop` will both remove the top item, and return it. In the constructor for `PlayLevels` you should initialize the list and push two levels onto the stack: a level to implement the menu system, described in question #3, and the scrolling globes example, described in question #1.
- (b) Note, the `Exit` button in the gui will terminate the program. Thus, fix the `Continue` button that you wrote in question #2, so that you will smoothly go into the globe animation.

Your main routine should be as follows:

```
if __name__ == "__main__":  
    levels = Levels()  
    levels.play()
```