Phase 3 Refactoring

Code Smell 1 - Dispensable (Duplicate Code) - (New_game.py)

In our initial implementation of a dropdown feature for our new game menu in NewGame.py, we implemented each of our select country buttons using Pygame rectangles. Since our game involves five players (four players + AI), most of the code written to draw the rectangles with the appropriate text, making them clickable, and replacing the text of the original select country button was highly repetitive and violated the principle of DRY code. As the number of options increased, we began accumulating duplicated code that was very hard to read, update, and understand. We used this technique as it was consistent with other ways we fixed duplicated code throughout our game. In order to refactor this code, we implemented a selectFlag() function, which handles the task of setting the text of the original select country button and checks for mouse clicks. By implementing this function, we significantly reduced the amount of duplicated code in our program. However, one thing to note about this fix is that upon creating selectFlag(), we noticed it contained too many parameters. This is indicative of a bloater code smell; therefore, through using refactoring to fix duplicated code, introducing this function outweighed keeping the old version of the code.

Code Smell 2 - Dispensable (Duplicate Code)

Before refactoring this Duplicate Code Smell, any instance of writing text to the screen required five lines of code. These five lines are: setting the font, generating the text, getting a rectangle, centering the text on the rectangle, and blitzing the object to the screen. Since our How To Play menu in How_To.py has multiple tips with two lines each, these five lines are repeated numerous times, only differing in the sentence, the x and y positions, font size, and colour. Text was also displayed on other menus (Main_menu.py, New_game.py,) and in the game, which also re-used these similar five lines of code. We solved this duplication smell by creating a class with a method that takes a string, the x and y position, font size, and colour. We decided it was worth creating another class over other methods because we found this code was duplicated over multiple files, and it would improve our code as a whole using this method. Our code was improved because of those five lines to 1 in many instances throughout our code.

```
class Text():

    def __init__(self):
        resolution=(800, 800)
        self.screen=pygame.display.set_mode(resolution)

    def print_sentance(self,string,x_pos,y_pos,font_size,colour):
        #Refactoring Function
        text_font = pygame.font.Font('freesansbold.ttf', font_size)
        Label = text_font.render(string,
                True, (0,0,0), colour)
        Label_Rect = Label.get_rect()
        Label_Rect.center = (x_pos, y_pos) #text position
        self.screen.blit(Label,Label_Rect)
```

Figure 2.1 - display.py Text Class created to solve duplicate code used in multiple classes

```
text.print_sentance("How To Play Pandemic: The Triva Board Game",400,75,25,color)
text.print_sentance("To begin a player turn click the 'Roll' Button to roll the 3-Sided dice. Then use your cursor to",410,162,15,color)
text.print_sentance("move your flag forward within your lane.",255,200,15,color)
text.print_sentance("Upon crossing a Trivia Cell, the player is prompted with a random trivia question. To answer",415,262,15,color)
text.print_sentance("the trivia question, simply click on any of the four multiple-choice answers.",375,302,15,color)
text.print_sentance("Within each players lane there are hidden chance cards that can either give one point or",403,362,15,color)
text.print_sentance("deduct one point.",170,392,15,color)
text.print_sentance("Players are able to spend these points by clicking the 'Buy Upgrades' button and choosing",415,462,15,color)
text.print_sentance("either of the 3 options.",192,492,15,color)
text.print_sentance("To win the game a player must reach the right side of the board before the timer reaches 0.",407,562,15,color)
```

Figure 2.2 - How_To.py Run() **After Refactoring** (*Was 50+ lines of code prior*)

Code Smell 3 - Magic Numbers

Before refactoring our code, our code was filled with many numbers that were undefined. We, as programmers, knew about the function of these numbers, but it would be unclear to a tester or someone who has not followed our project closely. While making the new_game menu, the board, and the settings menu, we made a lot of UI objects, such as buttons, titles etc. The colours we used to make these objects could only be seen if the code was run, but now after defining the colours, they can be understood from the code itself. This fix was used extensively throughout the code. Here are some examples of colour magic numbers in our code :

```
def create_upgrade_button(self):
    color=(255,0,0)
    diceButton= Rect(30,30,150,50)
    diceButton.move_ip(595,280)
    pygame.draw.rect(self.screen,color,diceButton)
    font=pygame.font.Font('freesansbold.ttf',20)
    text=font.render('Buy Upgrades',True,self.NAVY_BLUE)
    textRect=text.get_rect()
    textRect.center=(700,335)
    self.screen.blit(text,textRect)

def create_upgrade_first(self):
    color=(255,0,0)
    diceButton= Rect(30,30,150,30)
    diceButton.move_ip(595,340)
    pygame.draw.rect(self.screen,color,diceButton)
    font=pygame.font.Font('freesansbold.ttf',13)
    text=font.render('1:Move Self Forward 1',True,self.NAVY_BLUE)
    textRect=text.get_rect()
    textRect.center=(700,387)
    self.screen.blit(text,textRect)

def create_upgrade_second(self):
    color=(255,0,0)
    diceButton= Rect(30,30,150,30)
    diceButton.move_ip(595,380)
    pygame.draw.rect(self.screen,color,diceButton)
    font=pygame.font.Font('freesansbold.ttf',13)
    text=font.render('1:Move Player Back 1',True,self.NAVY_BLUE)
    textRect=text.get_rect()
    textRect.center=(697,425)
    self.screen.blit(text,textRect)

def create_upgrade_third(self):
    color=(255,0,0)
    diceButton= Rect(30,30,150,30)
    diceButton.move_ip(595,420)
    pygame.draw.rect(self.screen,color,diceButton)
    font=pygame.font.Font('freesansbold.ttf',13)
    text=font.render('3:Move Player to Start',True,self.NAVY_BLUE)
    textRect=text.get_rect()
    textRect.center=(700,465)
    self.screen.blit(text,textRect)
```

```
for event in pygame.event.get():
            SKY_BLUE = (118, 197, 234)
            BLUE = (118, 158, 234)
            PASTEL_BLUE = (120, 118, 234)
            DODGER_BLUE = (66, 97, 252)

            if event.type == pygame.MOUSEBUTTONDOWN:
                if OPTION_1.user_click(pygame.mouse.get_pos()):
                    DropDownMenu.draw_option(self, self.screen, "Player 1 - Canada", 125, 45, 275, 200, color = SKY_BLUE )
                    DropDownMenu.draw_option(self, self.screen, "Player 1 - China", 125, 45, 400, 200, color = BLUE)
                    DropDownMenu.draw_option(self, self.screen, "Player 1 - Mexico", 125, 45, 275, 245, color = PASTEL_BLUE)
                    DropDownMenu.draw_option(self, self.screen, "Player 1 - India", 125, 45, 400, 245, color = DODGER_BLUE)

                DropDownMenu.selectFlag(self, dropdown_button, dropdown_button2, dropdown_button3, dropdown_button4, OPTION_1, text, 1, listName)

            if event.type == pygame.MOUSEBUTTONDOWN:
                if OPTION_2.user_click(pygame.mouse.get_pos()):
                    DropDownMenu.draw_option(self, self.screen, "Player 2 - Canada", 125, 45, 275, 350, color = SKY_BLUE)
                    DropDownMenu.draw_option(self, self.screen, "Player 2 - China", 125, 45, 400, 350, color = BLUE)
                    DropDownMenu.draw_option(self, self.screen, "Player 2 - Mexico", 125, 45, 275, 395, color = PASTEL_BLUE)
                    DropDownMenu.draw_option(self, self.screen, "Player 2 - India", 125, 45, 400, 395, color = DODGER_BLUE)

                DropDownMenu.selectFlag(self, dropdown_button5, dropdown_button6, dropdown_button7, dropdown_button8, OPTION_2, text2, 2, listName)

            if event.type == pygame.MOUSEBUTTONDOWN:
                if OPTION_3.user_click(pygame.mouse.get_pos()):
                    DropDownMenu.draw_option(self, self.screen, "Player 3 - Canada", 125, 45, 275, 500, color = SKY_BLUE)
                    DropDownMenu.draw_option(self, self.screen, "Player 3 - China", 125, 45, 400, 500, color = BLUE)
                    DropDownMenu.draw_option(self, self.screen, "Player 3 - Mexico", 125, 45, 275, 545, color = PASTEL_BLUE)
                    DropDownMenu.draw_option(self, self.screen, "Player 3 - India", 125, 45, 400, 545, color = DODGER_BLUE)

                DropDownMenu.selectFlag(self, dropdown_button9, dropdown_button10, dropdown_button11, dropdown_button12, OPTION_3, text3, 3, listName)

            if event.type == pygame.MOUSEBUTTONDOWN:
                if OPTION_4.user_click(pygame.mouse.get_pos()):
                    DropDownMenu.draw_option(self, self.screen, "Player 4 - Canada", 125, 45, 275, 650, color = SKY_BLUE)
                    DropDownMenu.draw_option(self, self.screen, "Player 4 - China", 125, 45, 400, 650, color = BLUE)
                    DropDownMenu.draw_option(self, self.screen, "Player 4 - Mexico", 125, 45, 275, 695, color = PASTEL_BLUE)
                    DropDownMenu.draw_option(self, self.screen, "Player 4 - India", 125, 45, 400, 695, color = DODGER_BLUE)

                DropDownMenu.selectFlag(self, dropdown_button13, dropdown_button14, dropdown_button15, dropdown_button16, OPTION_4, text4, 4, listName)
```

Even when making the timer, we had magic numbers that were difficult to understand. These were useful in calculating the remaining time. A proper constant was assigned to these numbers too. Here is an example :

```
    seconds = (int(seconds)) #print how many seconds
    maxTime=330
    if(mainmenu1.loadGame==1):
        remainingseconds=gameData['Timer']
    remainingseconds = maxTime - seconds
    seconds=int((pygame.time.get_ticks()-remainingseconds)/1000) #calculate how many seconds have elapsed
    board.timer_seconds(remainingseconds)
    gameData['Timer']=remainingseconds
    if(remainingseconds == 0):
        board.display_seconds_window()
```