GIOCATORE 1

PUNTEGGIO 2500

GIOCATORE 2

CASTLE WAR

BY SANDY JIANG

START MENU SIGN IN

HOME PAGE

```
def home_screen(play_text, load_text):
    pygame.draw.rect(game_display, (56,207,56), (0, 0, screen_width, 350))
    pygame.draw.rect(game_display, (151,230,151), (0, 0, screen_width, 300))
    pygame.draw.rect(game_display, (213,245,213), (0, 0, screen_width, 200))
    play_button = pygame.Rect(260, 230, 250, 50)
    load_button = pygame.Rect(640, 230, 250, 50)
    mouse_pos = pygame.mouse.get_pos()

    if play_button.collidepoint(mouse_pos):
        pygame.draw.rect(game_display,(175, 225, 175), play_button)

    else:
        pygame.draw.rect(game_display,(0,156,75), play_button)

    if load_button.collidepoint(mouse_pos):
        pygame.draw.rect(game_display,(175, 225, 175), load_button)

    else:
        pygame.draw.rect(game_display,(0,156,75), load_button)

    play_button_border= pygame.draw.rect(game_display, (70,70,70), play_button,3)
    load_button_border= pygame.draw.rect(game_display, (70,70,70), load_button,3)

    game_display.blit(play_text, (315, 240))
    game_display.blit(load_text, (690, 240))
    game_display.blit(castle_war, (390, 80))

    return play_button, load_button
```

```python
def draw_background(red_turn, pause, save, red_players_keys, blue_players_keys):

    #background
    game_display.fill((255,255,255))
    if red_turn:
        red_rect = pygame.Rect(0, 0, screen_width//2, screen_height)
        pygame.draw.rect(game_display, (255,232,236), red_rect)

    else:
        blue_rect = pygame.Rect(screen_width//2, 0, screen_width//2, screen_height)
        pygame.draw.rect(game_display, (226, 220, 255), blue_rect)

    grass=pygame.Rect(0, 300, screen_width, 50)
    pygame.draw.rect(game_display, (80, 200, 120), grass)

    #red castle
    game_display.blit(red_mine, (10,263))
    game_display.blit(red_barrack, (92,250))
    game_display.blit(red_tower, (185,140))
    game_display.blit(red_wall, (180,220))

    #blue castle
    game_display.blit(blue_mine, (1082,263))
    game_display.blit(blue_barrack, (996,250))
    game_display.blit(blue_tower, (913,140))
    game_display.blit(blue_wall, (908,220))

    #keys
    game_display.blit(red_players_keys, (15, 308))
    game_display.blit(blue_players_keys, (15, 326))
    game_display.blit(pause, (440, 10))
    game_display.blit(save, (430, 32))
```

COUNTDOWN

```
def countdown(turn_start_time2, num_sec, archer_turn_trained, sword_turn_trained, train_one_at_time, cost_message, colour, red_turn, textfont4):
     if pygame.time.get_ticks() - turn_start_time2 >= 1000:
          num_sec-=1
          turn_start_time2 = pygame.time.get_ticks()

     if num_sec>=0:
          if colour=='red':
               seconds=textfont4.render(str(num_sec) +"s", 1, (255,114,118))
               game_display.blit(seconds, (500,50))
          if colour=='blue':
               seconds=textfont4.render(str(num_sec) +"s", 1, (28, 125, 213))
               game_display.blit(seconds, (630,50))
     else:
          if colour=='red':
               red_turn=False
          if colour=='blue':
               red_turn=True
          if archer_turn_trained>0:
               archer_turn_trained+=1
          if sword_turn_trained>0:
               sword_turn_trained+=1
          train_one_at_time=False
          cost_message=False

     return  archer_turn_trained, sword_turn_trained, train_one_at_time, cost_message, num_sec, turn_start_time2, red_turn
```

5s 4s 3s 2s 1s 0s

5s 4s 3s 2s 1s 0s

TRAINING UNITS

```
if event.key == pygame.K_q:
     if red_init_resources<worker_cost:
          red_cost_message=True

     elif red_worker_turn_trained>0 and red_worker_turn_trained<worker_train:
          red_train_one_at_time=True

     else:
          red_worker_turn_trained=train_red_units(red_minus_Cost_worker, worker_cost, red_worker_turn_trained)
```

```
def train_red_units(minus_Cost_unit, unit_cost, unit_turn_trained):
     global red_train_one_at_time, red_cost_message, minus_points_displayed, red_init_resources, minus_points_start_time, red_turn
     red_train_one_at_time=False
     red_cost_message=False
     minus_points_displayed = minus_Cost_unit
     red_init_resources-=unit_cost
     minus_points_start_time = pygame.time.get_ticks()
     unit_turn_trained+=1
     red_turn=False
     return unit_turn_trained
```

```
 if red_train_one_at_time:
      game_display.blit(AltMessage_train, (20,40))
```

resources 200

This unit is already training choose another one

```
if red_cost_message:
     game_display.blit(AltMessage_cost, (20,50))
```

resources 30

You don't have enough resources

MODIFYING RESOURCES

```
#show cost
if minus_points_displayed:
     current_time = pygame.time.get_ticks()
     time_passed = current_time - minus_points_start_time

     if time_passed < 1000:
          if minus_points_displayed==red_minus_Cost_worker or minus_points_displayed==red_minus_Cost_archer or minus_points_displayed==red_minus_Cost_sword:
               game_display.blit(minus_points_displayed, (120,35))
          if minus_points_displayed==blue_minus_Cost_worker or minus_points_displayed==blue_minus_Cost_archer or minus_points_displayed==blue_minus_Cost_sword:
               game_display.blit(minus_points_displayed, (1052,35))
     else:
          minus_points_displayed=None
```

```
def show_added_resources(num_Workers_toMine, colour, plus_resources_added, turn_start_time, init_resources, worker_prod, textfont):
     if num_Workers_toMine>0:
          plus_resources=num_Workers_toMine*worker_prod
          text_plus_resources=textfont.render("+" + str(plus_resources), 1, (80, 200, 120))
          if pygame.time.get_ticks() - turn_start_time < 1500:
               if colour=='red':
                    game_display.blit(text_plus_resources, (165,15))
               if colour=='blue':
                    game_display.blit(text_plus_resources, (1095,15))
          if not plus_resources_added:
               init_resources += plus_resources
               plus_resources_added = True
     return plus_resources_added, init_resources
```

MENU 01 07 25

PAUSED SCREEN

```
#paused screen
if paused:
     pause_game(resume_text)
     if red_turn:
          pygame.draw.rect(game_display,(255,232,236), (425,17,150,20))
          pygame.draw.rect(game_display,(255, 255, 255),(575,17,150,20))
     else:
          pygame.draw.rect(game_display,(226, 220, 255), (575,17,150,20))
          pygame.draw.rect(game_display,(255, 255, 255), (425,17,150,20))
```

```
if event.type == pygame.KEYDOWN:

     if event.key == pygame.K_SPACE:
          if paused:
               paused = False
          else:
               paused = True
```

WORKERS

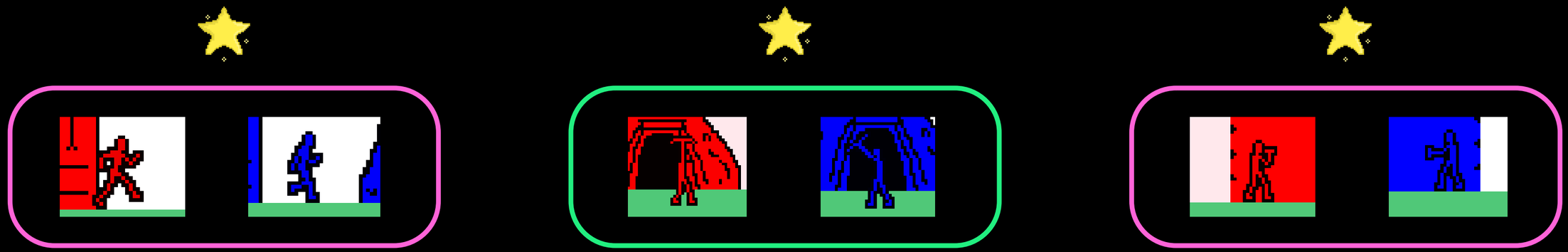
```
class Worker(pygame.sprite.Sprite):
    def __init__(self, initial_pos, colour):
        super().__init__()
        self.index = 0
        self.image=self.ready
        self.rect=self.image.get_rect()
        self.speed = 5
        self.rect.x, self.rect.y=initial_pos
        self.move_toMine=False
        self.move_toWall=False
        self.digging=False
        self.repairing=False
        self.sprite_ready=True
```




```
def update(self):

    if self.move_toMine:
        if self.index >= len(self.run):
            self.index = 0

        self.image = pygame.transform.flip(self.run[self.index], True, False)
        self.index += 1
        if self.colour=='red':
            if self.rect.x<=random.randint(25,50):
                self.digging=True
                self.move_toMine=False
            self.rect.x -= self.speed

        if self.colour=='blue':
            if self.rect.x>=random.randint(1092,1117):
                self.digging=True
                self.move_toMine=False
            self.rect.x += self.speed


    elif self.digging:
        if self.index >= len(self.dig):
            self.index = 0
        self.image = self.dig[self.index]
        self.index += 1
  elif self.move_toWall:
      if self.index >= len(self.run):
          self.index = 0
      self.image = self.run[self.index]
      self.index += 1
      if self.colour=='red':
          if self.rect.x>=random.randint(167,177):
              self.repairing=True
              self.move_toWall=False
          self.rect.x += self.speed
      if self.colour=='blue':
          if self.rect.x<=random.randint(955,965):
              self.repairing=True
              self.move_toWall=False
          self.rect.x -= self.speed

  elif self.repairing:
      if self.index >= len(self.repair):
          self.index = 0
      self.image = self.repair[self.index]
      self.index += 1
```

```python
class RedWorker(Worker):
    def __init__(self):
        self.run=[pygame.image.load('sprites/player1/worker/run-0.png'),
                  pygame.image.load('sprites/player1/worker/run-1.png'),
                  pygame.image.load('sprites/player1/worker/run-2.png'),
                  pygame.image.load('sprites/player1/worker/run-3.png'),
                  pygame.image.load('sprites/player1/worker/run-4.png'),
                  pygame.image.load('sprites/player1/worker/run-5.png')]
        self.ready=pygame.image.load('sprites/player1/worker/ready.png')
        self.repair=[pygame.image.load('sprites/player1/worker/repair-0.png'),
                     pygame.image.load('sprites/player1/worker/repair-1.png'),
                     pygame.image.load('sprites/player1/worker/repair-2.png'),
                     pygame.image.load('sprites/player1/worker/repair-3.png'),]
        self.dig=[pygame.image.load('sprites/player1/worker/dig-0.png'),
                      pygame.image.load('sprites/player1/worker/dig-1.png'),
                      pygame.image.load('sprites/player1/worker/dig-2.png'),
                      pygame.image.load('sprites/player1/worker/dig-3.png'),
                      pygame.image.load('sprites/player1/worker/dig-4.png'),
                      pygame.image.load('sprites/player1/worker/dig-5.png'),
                      pygame.image.load('sprites/player1/worker/dig-6.png'),
                      pygame.image.load('sprites/player1/worker/dig-7.png'),
                      pygame.image.load('sprites/player1/worker/dig-8.png')]

        self.initial_pos = (random.randint(92, 154),280)
        self.colour='red'
        super().__init__(self.initial_pos, self.colour)
```

```python
class BlueWorker(Worker):
    def __init__(self):
        self.run=[pygame.image.load('sprites/player2/worker/run-0.png'),
                  pygame.image.load('sprites/player2/worker/run-1.png'),
                  pygame.image.load('sprites/player2/worker/run-2.png'),
                  pygame.image.load('sprites/player2/worker/run-3.png'),
                  pygame.image.load('sprites/player2/worker/run-4.png'),
                  pygame.image.load('sprites/player2/worker/run-5.png')]
        self.ready=pygame.image.load('sprites/player2/worker/ready.png')
        self.repair=[pygame.image.load('sprites/player2/worker/repair-0.png'),
                     pygame.image.load('sprites/player2/worker/repair-1.png'),
                     pygame.image.load('sprites/player2/worker/repair-2.png'),
                     pygame.image.load('sprites/player2/worker/repair-3.png'),]
        self.dig=[pygame.image.load('sprites/player2/worker/dig-0.png'),
                      pygame.image.load('sprites/player2/worker/dig-1.png'),
                      pygame.image.load('sprites/player2/worker/dig-2.png'),
                      pygame.image.load('sprites/player2/worker/dig-3.png'),
                      pygame.image.load('sprites/player2/worker/dig-4.png'),
                      pygame.image.load('sprites/player2/worker/dig-5.png'),
                      pygame.image.load('sprites/player2/worker/dig-6.png'),
                      pygame.image.load('sprites/player2/worker/dig-7.png'),
                      pygame.image.load('sprites/player2/worker/dig-8.png')]
        self.initial_pos= (random.randint(996, 1058), 280)
        self.colour='blue'
        super().__init__(self.initial_pos, self.colour)
```



ATTACK UNITS

```
class AttackUnit(pygame.sprite.Sprite):

    def __init__(self, initial_pos, max_health, colour):
        super().__init__()
        self.index = 0
        self.image=self.ready
        self.rect=self.image.get_rect()
        self.rect.x, self.rect.y=initial_pos
        self.unleash = False
        self.falling = False
        self.current_health = max_health
        self.max_health = max_health
        self.health_bar_length = 25
        self.health_ratio = self.max_health / self.health_bar_length
        self.colour=colour
        self.sprite_ready=True

    def update(self):
        self.health_bar()

        if self.unleash:
            if self.index >= len(self.run):
                self.index = 0
            self.image = self.run[self.index]
            self.index += 1
            if self.colour=='red':
                self.rect.x += self.speed
            if self.colour=='blue':
                self.rect.x -= self.speed

        elif self.falling:
            if self.index >= len(self.fallen):
                self.kill()
            else:
                self.image = self.fallen[self.index]
                self.index += 1
```

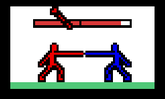
```
    def get_damage(self, amount):
        if self.current_health>0:
            self.current_health-=amount
        else:
            self.current_health=0

    def health_bar(self):
        pygame.draw.rect(game_display, (217, 55, 55), (self.rect.x-2, self.rect.y-10, self.current_health/self.health_ratio, 4))
        pygame.draw.rect(game_display, (0,0,0), (self.rect.x-2, self.rect.y-10, self.health_bar_length, 4),1)
```

```python
class RedKnight(AttackUnit):
    def __init__(self):
        self.attack=[pygame.image.load('sprites/player1/sword/attack-0.png'),
            pygame.image.load('sprites/player1/sword/attack-1.png'),
            pygame.image.load('sprites/player1/sword/attack-2.png'),
            pygame.image.load('sprites/player1/sword/attack-3.png'),
            pygame.image.load('sprites/player1/sword/attack-4.png'),
            pygame.image.load('sprites/player1/sword/attack-5.png'),
            pygame.image.load('sprites/player1/sword/attack-6.png'),
            pygame.image.load('sprites/player1/sword/attack-7.png')]
        self.fallen=[pygame.image.load('sprites/player1/sword/fallen-0.png'),
             pygame.image.load('sprites/player1/sword/fallen-1.png'),
             pygame.image.load('sprites/player1/sword/fallen-2.png'),
             pygame.image.load('sprites/player1/sword/fallen-3.png'),
             pygame.image.load('sprites/player1/sword/fallen-4.png'),
             pygame.image.load('sprites/player1/sword/fallen-5.png')]
        self.ready=pygame.image.load('sprites/player1/sword/ready.png')
        self.run=[pygame.image.load('sprites/player1/sword/run-0.png'),
           pygame.image.load('sprites/player1/sword/run-1.png'),
           pygame.image.load('sprites/player1/sword/run-2.png'),
           pygame.image.load('sprites/player1/sword/run-3.png'),
           pygame.image.load('sprites/player1/sword/run-4.png'),
           pygame.image.load('sprites/player1/sword/run-5.png'),
           pygame.image.load('sprites/player1/sword/run-6.png'),
           pygame.image.load('sprites/player1/sword/run-7.png'),
           pygame.image.load('sprites/player1/sword/run-8.png'),
           pygame.image.load('sprites/player1/sword/run-9.png'),
           pygame.image.load('sprites/player1/sword/run-10.png'),
           pygame.image.load('sprites/player1/sword/run-11.png')]
        self.initial_pos = (random.randint(92, 154),280)
        self.attacking=False
        self.max_health=90
        self.colour='red'
        self.speed = 5
        super().__init__(self.initial_pos, self.max_health, self.colour)

    def update(self):
        super().update()
        if self.attacking:
            if self.index >= len(self.attack):
                self.index = 0
            self.image = self.attack[self.index]
            self.index += 1
```

```python
class BlueKnight(AttackUnit):
    def __init__(self):
        self.attack=[pygame.image.load('sprites/player2/sword/attack-0.png'),
           pygame.image.load('sprites/player2/sword/attack-1.png'),
           pygame.image.load('sprites/player2/sword/attack-2.png'),
           pygame.image.load('sprites/player2/sword/attack-3.png'),
           pygame.image.load('sprites/player2/sword/attack-4.png'),
           pygame.image.load('sprites/player2/sword/attack-5.png'),
           pygame.image.load('sprites/player2/sword/attack-6.png'),
           pygame.image.load('sprites/player2/sword/attack-7.png')]
        self.fallen=[pygame.image.load('sprites/player2/sword/fallen-0.png'),
             pygame.image.load('sprites/player2/sword/fallen-1.png'),
             pygame.image.load('sprites/player2/sword/fallen-2.png'),
             pygame.image.load('sprites/player2/sword/fallen-3.png'),
             pygame.image.load('sprites/player2/sword/fallen-4.png'),
             pygame.image.load('sprites/player2/sword/fallen-5.png')]
        self.ready=pygame.image.load('sprites/player2/sword/ready.png')
        self.run=[pygame.image.load('sprites/player2/sword/run-0.png'),
           pygame.image.load('sprites/player2/sword/run-1.png'),
           pygame.image.load('sprites/player2/sword/run-2.png'),
           pygame.image.load('sprites/player2/sword/run-3.png'),
           pygame.image.load('sprites/player2/sword/run-4.png'),
           pygame.image.load('sprites/player2/sword/run-5.png'),
           pygame.image.load('sprites/player2/sword/run-6.png'),
           pygame.image.load('sprites/player2/sword/run-7.png'),
           pygame.image.load('sprites/player2/sword/run-8.png'),
           pygame.image.load('sprites/player2/sword/run-9.png'),
           pygame.image.load('sprites/player2/sword/run-10.png'),
           pygame.image.load('sprites/player2/sword/run-11.png')]
        self.initial_pos = (random.randint(996, 1058),280)
        self.attacking=False
        self.max_health=90
        self.colour='blue'
        self.speed = 5
        super().__init__(self.initial_pos, self.max_health, self.colour)

    def update(self):
        super().update()
        if self.attacking:
            if self.index >= len(self.attack):
                self.index = 0
            self.image = self.attack[self.index]
            self.index += 1
```

KNIGHTS

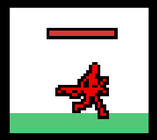


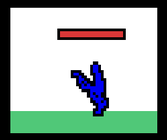


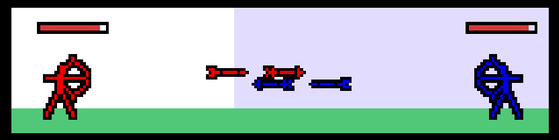
```python
class RedArcher(AttackUnit):
    def __init__(self):

        self.run=[pygame.image.load('sprites/player1/bow/run-0.png'),
                  pygame.image.load('sprites/player1/bow/run-1.png'),
                  pygame.image.load('sprites/player1/bow/run-2.png'),
                  pygame.image.load('sprites/player1/bow/run-3.png'),
                  pygame.image.load('sprites/player1/bow/run-4.png'),
                  pygame.image.load('sprites/player1/bow/run-5.png'),
                  pygame.image.load('sprites/player1/bow/run-6.png'),
                  pygame.image.load('sprites/player1/bow/run-7.png'),
                  pygame.image.load('sprites/player1/bow/run-8.png'),
                  pygame.image.load('sprites/player1/bow/run-9.png'),
                  pygame.image.load('sprites/player1/bow/run-10.png'),
                  pygame.image.load('sprites/player1/bow/run-11.png'),]
        self.ready=pygame.image.load('sprites/player1/bow/ready.png')
        self.shoot=[pygame.image.load('sprites/player1/bow/shoot-0.png'),
                    pygame.image.load('sprites/player1/bow/shoot-1.png')]
        self.fallen=[pygame.image.load('sprites/player1/bow/fallen-0.png'),
                     pygame.image.load('sprites/player1/bow/fallen-1.png'),
                     pygame.image.load('sprites/player1/bow/fallen-2.png'),
                     pygame.image.load('sprites/player1/bow/fallen-3.png'),
                     pygame.image.load('sprites/player1/bow/fallen-4.png'),
                     pygame.image.load('sprites/player1/bow/fallen-5.png')]
        self.initial_pos=(random.randint(92, 154), 280)
        self.max_health=60
        self.shooting=False
        self.colour='red'
        self.speed = 8
        super().__init__(self.initial_pos, self.max_health, self.colour)


    def update(self):
        super().update()
        if self.shooting:
            if self.index >= len(self.shoot):
                self.index = 0
            self.image = self.shoot[self.index]
            self.index += 1
```

```python
class BlueArcher(AttackUnit):
    def __init__(self):
        self.run=[pygame.image.load('sprites/player2/bow/run-0.png'),
                  pygame.image.load('sprites/player2/bow/run-1.png'),
                  pygame.image.load('sprites/player2/bow/run-2.png'),
                  pygame.image.load('sprites/player2/bow/run-3.png'),
                  pygame.image.load('sprites/player2/bow/run-4.png'),
                  pygame.image.load('sprites/player2/bow/run-5.png'),
                  pygame.image.load('sprites/player2/bow/run-6.png'),
                  pygame.image.load('sprites/player2/bow/run-7.png'),
                  pygame.image.load('sprites/player2/bow/run-8.png'),
                  pygame.image.load('sprites/player2/bow/run-9.png'),
                  pygame.image.load('sprites/player2/bow/run-10.png'),
                  pygame.image.load('sprites/player2/bow/run-11.png'),]
        self.ready=pygame.image.load('sprites/player2/bow/ready.png')
        self.shoot=[pygame.image.load('sprites/player2/bow/shoot-0.png'),
                    pygame.image.load('sprites/player2/bow/shoot-1.png')]
        self.fallen=[pygame.image.load('sprites/player2/bow/fallen-0.png'),
                     pygame.image.load('sprites/player2/bow/fallen-1.png'),
                     pygame.image.load('sprites/player2/bow/fallen-2.png'),
                     pygame.image.load('sprites/player2/bow/fallen-3.png'),
                     pygame.image.load('sprites/player2/bow/fallen-4.png'),
                     pygame.image.load('sprites/player2/bow/fallen-5.png')]

        self.initial_pos = (random.randint(996, 1058),280)
        self.shooting=False
        self.max_health=60
        self.colour='blue'
        self.speed = 8
        super().__init__(self.initial_pos, self.max_health, self.colour)

    def update(self):
        super().update()
        if self.shooting:
            if self.index >= len(self.shoot):
                self.index = 0
            self.image = self.shoot[self.index]
            self.index += 1
```

ARCHERS







MENU 01 07 25

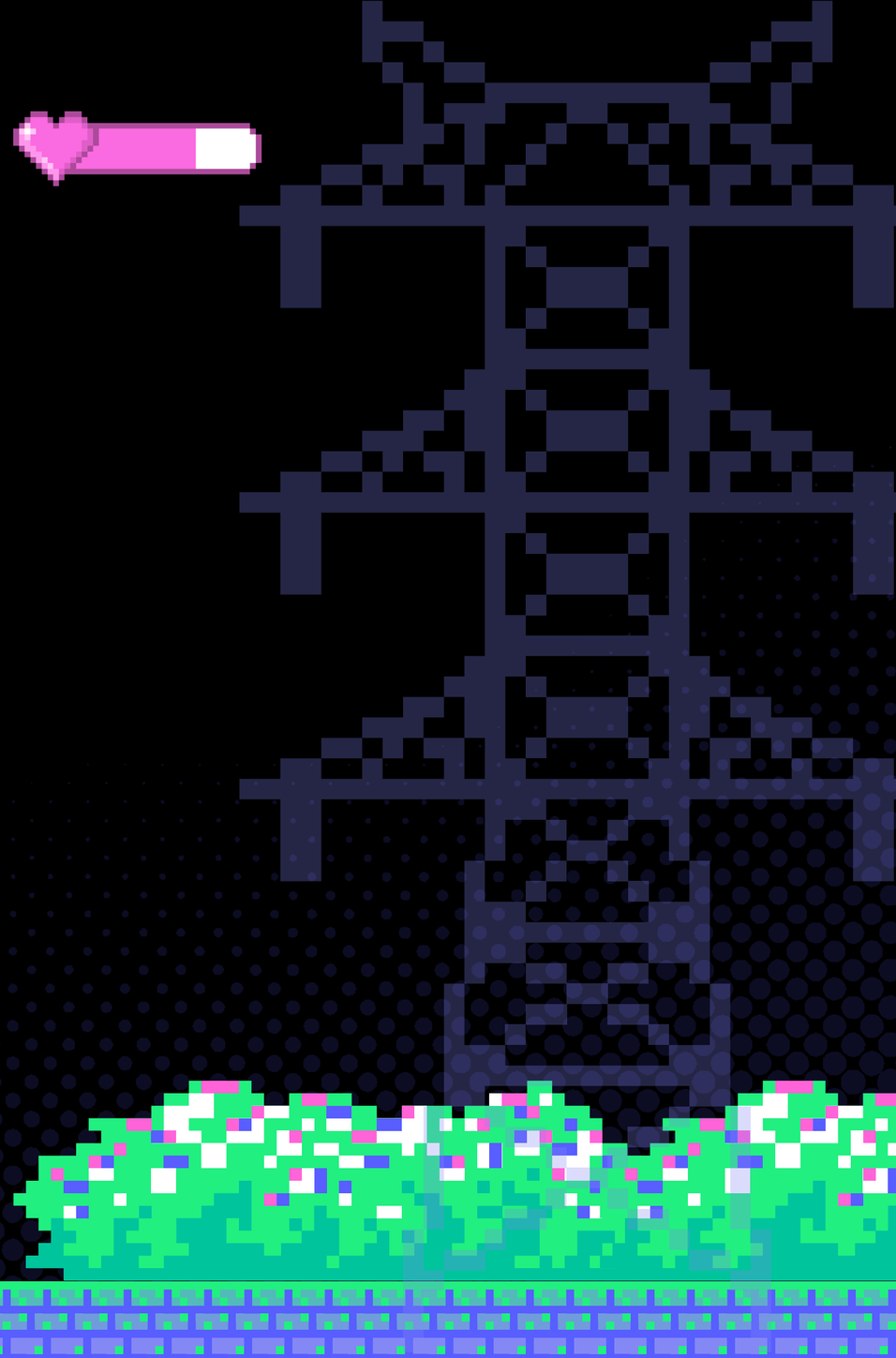
ARCHERS' ARROWS

```
class ArcherArrows(pygame.sprite.Sprite):
     def __init__(self):
          super().__init__()
          self.index = 0
          self.image=self.horizontal[0]
          self.rect=self.image.get_rect()
          self.speed=20
          self.rect.x=self.pos_x
          self.rect.y=self.pos_y

     def update(self):

          if self.index >= len(self.horizontal):
               self.index = 0
          self.image = self.horizontal[self.index]
          self.index += 1
          if self.colour=='red':
               self.rect.x += self.speed
          if self.colour=='blue':
               self.rect.x -= self.speed
```

```
class RedArcher_Arrows(ArcherArrows):
     def __init__(self, pos_x, pos_y):
          self.horizontal=[pygame.image.load('sprites/player1/bow/arrowhor-0.png'),
               pygame.image.load('sprites/player1/bow/arrowhor-1.png')]
          self.pos_x=pos_x
          self.pos_y=pos_y
          self.colour='red'
          super().__init__()


class BlueArcher_Arrows(ArcherArrows):
     def __init__(self, pos_x, pos_y):
          self.horizontal=[pygame.image.load('sprites/player2/bow/arrowhor-0.png'),
               pygame.image.load('sprites/player2/bow/arrowhor-1.png'),]
          self.pos_x=pos_x
          self.pos_y=pos_y
          self.colour='blue'
          super().__init__()
```


MENU
01
07
25
TOWERS' ARROWS
class TowerArrows(pygame.sprite.Sprite):
def __init__(self, initial_pos):
super().__init__()
self.index = 0
self.image = self.diag_arrows[0]
self.rect = self.image.get_rect()
self.rect.x, self.rect.y=initial_pos
self.speed = 30
self.angle=0
self.diagonal=False
self.vertical=False
def update(self):
if self.diagonal:
if self.index >= len(self.diag_arrows):
self.index = 0
self.image = self.diag_arrows[self.index]
self.index += 1
self.x_diff = self.dest_x - self.rect.x
self.y_diff = self.dest_y - self.rect.y
self.angle = math.atan2(self.y_diff, self.x_diff)
self.rect.x+=math.cos(self.angle)*self.speed
self.rect.y+=math.sin(self.angle)*self.speed
if self.vertical:
if self.index >= len(self.vert_arrows):
self.index = 0
self.image = self.vert_arrows[self.index]
self.index += 1
self.rect.y+=self.speed
class RedTower_Arrows(TowerArrows):
def __init__(self, dest_x, dest_y):
self.diag_arrows=[
pygame.image.load('sprites/player1/bow/arrowdiag-0.png'),
pygame.image.load('sprites/player1/bow/arrowdiag-1.png')]
self.vert_arrows=[pygame.image.load('sprites/player1/bow/arrowvert-0.png'),
pygame.image.load('sprites/player1/bow/arrowvert-1.png')]
self.initial_pos= (232,180)
self.dest_x=dest_x
self.dest_y=dest_y
super().__init__(self.initial_pos)
class BlueTower_Arrows(TowerArrows):
def __init__(self, dest_x, dest_y):
self.diag_arrows=[
pygame.image.load('sprites/player2/bow/arrowdiag-0.png'),
pygame.image.load('sprites/player2/bow/arrowdiag-1.png')]
self.vert_arrows=[pygame.image.load('sprites/player2/bow/arrowvert-
pygame.image.load('sprites/player2/bow/arrowvert-1.png')]
self.initial_pos=(900,180)
self.dest_x=dest_x
self.dest_y=dest_y
super().__init__(self.initial_pos)

```
def Red_TowerHealth():
     global redTower_current_health
     health_bar_length=100
     health_ratio = max_health / health_bar_length
     pygame.draw.rect(game_display, (217, 55, 55), (160, 120, redTower_current_health/health_ratio, 10))
     pygame.draw.rect(game_display, (0,0,0), (160, 120, health_bar_length, 10),1)

def Blue_TowerHealth():
     global blueTower_current_health, max_health
     health_bar_length=100
     health_ratio = max_health / health_bar_length
     pygame.draw.rect(game_display, (217, 55, 55), (888, 120, blueTower_current_health/health_ratio, 10))
     pygame.draw.rect(game_display, (0,0,0), (888, 120, health_bar_length, 10),1)

def Tower_getDamage(colour, amount):
     global redTower_current_health, blueTower_current_health
     if colour=='red':
          if redTower_current_health>0:
               redTower_current_health-=amount
          if redTower_current_health<0:
               redTower_current_health=0
     if colour=='blue':
          if blueTower_current_health>0:
               blueTower_current_health-=amount
          if blueTower_current_health<0:
               blueTower_current_health=0

def Tower_getHealth(colour, amount):
     global redTower_current_health, blueTower_current_health, max_health
     if colour=='red':
          if redTower_current_health<max_health:
               redTower_current_health+=amount
          if redTower_current_health>=max_health:
               redTower_current_health=max_health

     if colour=='blue':
          if blueTower_current_health<max_health:
               blueTower_current_health+=amount
          if blueTower_current_health>=max_health:
               blueTower_current_health=max_health
```



ATTACKING FUNCTIONS

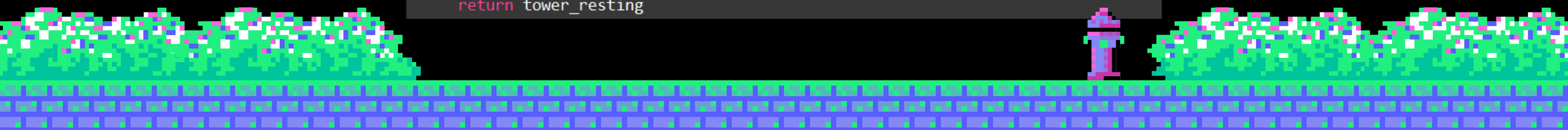
```
def actions(unitType, min_distance, aSprite, closest_enemy, damage, attack_range, tower_distance, TowerColour):
     attacking=False
     shooting=False
     if min_distance <= attack_range and not aSprite.sprite_ready:
          if unitType=='knight':
               aSprite.attacking=True
               attacking=True
          if unitType=='archer':
               aSprite.shooting=True
               shooting=True
          aSprite.unleash = False

          if min_distance==tower_distance:
               Tower_getDamage(TowerColour, damage)
     else:
          attacking=False
          shooting=False

     if closest_enemy and (shooting or attacking):
          closest_enemy.get_damage(damage)

     if  min_distance > attack_range and not aSprite.sprite_ready:
          if unitType=='knight':
               aSprite.attacking = False
          if unitType=='archer':
               aSprite.shooting = False
          aSprite.unleash = True

     if aSprite.current_health<=0:
          if unitType=='knight':
               aSprite.attacking = False
          if unitType=='archer':
               aSprite.shooting = False
          aSprite.unleash = False
          aSprite.falling=True
```

```
def compute_min_distance(aSprite, min_distance,  enemy_group, closest_enemy):
     for enemy in enemy_group:
          distance = abs(enemy.rect.x - aSprite.rect.x)
          if distance < min_distance:
               min_distance=distance
               closest_enemy=enemy
     return min_distance, closest_enemy
```

```
def compute_tower_min_distance(wall_x, aSprite, sprite_min_distance, closest_enemy):
     tower_distance=abs(wall_x - aSprite.rect.x)
     if tower_distance < sprite_min_distance:
          sprite_min_distance=tower_distance
          closest_enemy=None
```

RESTING FUNCTIONS

```
def enable_resting_knight(aSprite, sprite_resting, previous_time):
     if aSprite.attacking:
          if aSprite.index >= len(aSprite.attack):
               sprite_resting=True
               previous_time=pygame.time.get_ticks()
     return sprite_resting, previous_time

def enable_resting_tower(num_arrows, tower_resting, tower_previous_time):
     if num_arrows>=10:
          num_arrows=0
          tower_resting=True
          tower_previous_time=pygame.time.get_ticks()
     return num_arrows, tower_resting, tower_previous_time

def resting(group, unitType, previous_time, sprite_resting, sprite_rest_time):
     for aSprite in group:
          if unitType=='knight':
               aSprite.attacking=False
          if unitType=='archer':
               aSprite.shooting=False
     current_time = pygame.time.get_ticks()
     if current_time-previous_time >= sprite_rest_time:
          sprite_resting = False
     return sprite_resting

def towerResting(previous_time, tower_resting):
     current_time = pygame.time.get_ticks()
     if current_time - previous_time >= tower_rest:
          tower_resting = False
     return tower_resting
```



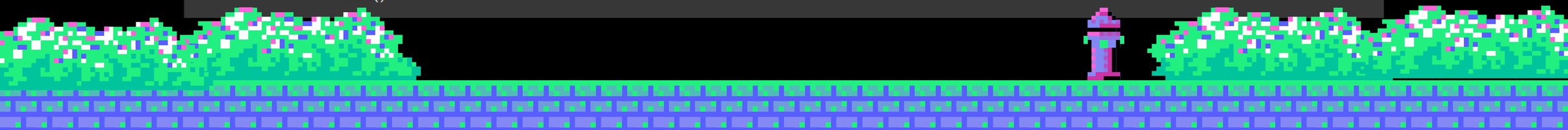
```python
def RedKnight_attackMode(RedKnight_group, BlueArcher_group, BlueKnight_group):
    global red_knight_resting, red_knight_previous_time
    red_knight_tower_distance=float('inf')

    if 'red_knight_resting' not in globals():
       red_knight_resting = False

    if not red_knight_resting:

        for red_knight in RedKnight_group:
            closest_enemy=None
            red_knight_min_distance=float('inf')

            red_knight_min_distance, closest_enemy = compute_min_distance(red_knight, red_knight_min_distance, BlueArcher_group, closest_enemy)
            red_knight_min_distance, closest_enemy = compute_min_distance(red_knight, red_knight_min_distance, BlueKnight_group, closest_enemy)
            red_knight_min_distance, red_knight_tower_distance, closest_enemy=compute_tower_min_distance(blue_wall_x, red_knight, red_knight_min_distance, closest_enemy)

            actions('knight', red_knight_min_distance, red_knight, closest_enemy,knight_damage, knight_range, red_knight_tower_distance, 'blue')

            red_knight_resting, red_knight_previous_time=enable_resting_knight(red_knight, red_knight_resting, red_knight_previous_time)

    else:#resting
        red_knight_resting=resting(RedKnight_group, 'knight', red_knight_previous_time, red_knight_resting, knight_rest)
```

```python
def RedArcher_shootMode(RedArcher_group, BlueArcher_group, BlueKnight_group, RedArrows_group):
    global red_archer_resting, red_archer_previous_time
    red_archer_archer_distance = float('inf')
    red_archer_knight_distance = float('inf')

    if 'red_archer_resting' not in globals():
       red_archer_resting = False

    if not red_archer_resting:
        for red_archer in RedArcher_group:
            closest_enemy=None
            red_archer_min_distance = float('inf')

            red_archer_min_distance, closest_enemy = compute_min_distance(red_archer, red_archer_min_distance, BlueKnight_group, closest_enemy)
            red_archer_min_distance, closest_enemy = compute_min_distance(red_archer, red_archer_min_distance, BlueArcher_group, closest_enemy)
            red_archer_min_distance, red_archer_tower_distance, closest_enemy=compute_tower_min_distance(blue_wall_x, red_archer, red_archer_min_distance, closest_enemy)

            actions('archer', red_archer_min_distance, red_archer, closest_enemy, archer_damage, archer_range, red_archer_tower_distance, 'blue')

            if red_archer.shooting:
                RedArrows_group.add(RedArcher_Arrows(red_archer.rect.x + 17, red_archer.rect.y + 5))
                if red_archer.index >= len(red_archer.shoot):
                    red_archer_resting = True
                    red_archer_previous_time = pygame.time.get_ticks()

    else:#resting
        red_archer_resting=resting(RedArcher_group, 'archer', red_archer_previous_time, red_archer_resting, archer_rest)

    ArcherArrows(RedArrows_group, BlueArcher_group, 'red')
    ArcherArrows(RedArrows_group, BlueKnight_group, 'red')
```

ARCHERS, KNIGHTS
ATTACK MODE

TOWERS' ATTACK MODE

```
def RedTower_shootMode(RedArrows_group, BlueArcher_group, BlueKnight_group):
     red_tower_min_distance=float('inf')
     closest_enemy=None
     global dest_x, dest_y, red_tower_resting, red_num_arrows, red_tower_previous_time, redArrow_archer_collision ,redArrow_knight_collision

     if 'red_tower_resting' not in globals():
          red_tower_resting = False

     if not red_tower_resting:

          for blue_archer in BlueArcher_group:
               red_tower_min_distance, dest_x, dest_y, closest_enemy=get_enemy_position(red_wall_x, blue_archer, red_tower_min_distance, closest_enemy, dest_x, dest_y)

          for blue_knight in BlueKnight_group:
               red_tower_min_distance, dest_x, dest_y, closest_enemy=get_enemy_position(red_wall_x, blue_knight, red_tower_min_distance, closest_enemy, dest_x, dest_y)

          if red_tower_min_distance<=tower_range:
               RedArrows_group.add(RedTower_Arrows(dest_x, dest_y))

          red_num_arrows, red_tower_resting, red_tower_previous_time=enable_resting_tower(red_num_arrows, red_tower_resting, red_tower_previous_time)

          for arrows in RedArrows_group:
               if dest_x>=(red_wall_x+5):
                    arrows.diagonal=True
                    arrows.vertiacl=False
               else:
                    arrows.diagonal=False
                    arrows.vertical=True
               red_num_arrows=TowerArrows(arrows, BlueArcher_group, BlueKnight_group, closest_enemy, red_num_arrows)


     else:#resting
          red_tower_resting=towerResting(red_tower_previous_time, red_tower_resting)

     for arrows in RedArrows_group:
          if redArrow_archer_collision or redArrow_knight_collision:
               arrows.kill()
          if arrows.rect.y>=280:
               arrows.kill()
```



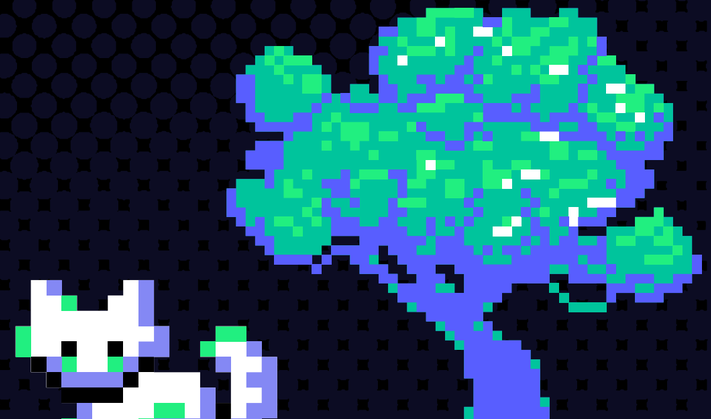
SAVING DATA

01 07 25

```
def store_data(data, red_init_resources, blue_init_resources, red_turn, num_re
    red_worker_turn_trained, red_archer_turn_trained, red_sword_turn_trained,
    RedArcher_group, RedKnight_group, RedArcher_Arrows_group, RedTower_Arrows
    data["red_init_resources"] = red_init_resources
    data["blue_init_resources"] = blue_init_resources
    data["red_turn"] = red_turn
    data["num_redWorkers_toMine"] = num_redWorkers_toMine
    data["num_blueWorkers_toMine"] = num_blueWorkers_toMine
    data["num_redWorkers_toWall"] = num_redWorkers_toWall
    data["num_blueWorkers_toWall"] = num_blueWorkers_toWall
    data["red_worker_turn_trained"] = red_worker_turn_trained
    data["red_archer_turn_trained"] = red_archer_turn_trained
    data["red_sword_turn_trained"] = red_sword_turn_trained
    data["blue_worker_turn_trained"] = blue_worker_turn_trained
    data["blue_archer_turn_trained"] = blue_archer_turn_trained
    data["blue_sword_turn_trained"] = blue_sword_turn_trained
    data["RedWorker_group"] =  store_Workers_attributes(RedWorker_group)
    data["RedArcher_group"] =  store_Archer_attributes(RedArcher_group)
    data["RedKnight_group"] =  store_Knight_attributes(RedKnight_group)
    data["BlueWorker_group"] = store_Workers_attributes(BlueWorker_group)
    data["BlueArcher_group"] = store_Archer_attributes(BlueArcher_group)
    data["BlueKnight_group"] = store_Knight_attributes(BlueKnight_group)
    store_data_spritesCode(data)
```

```
def store_Workers_attributes(group):
    attributes_list=[]
    for sprite in group:
        image_string=pygame.image.tostring(sprite.image, 'RGBA')
        encoded_image=base64.b64encode(image_string).decode('utf-8')
        attributes={
        "rect.x":sprite.rect.x,
        "rect.y":sprite.rect.y,
        "image_width": sprite.image.get_width(),
        "image_height": sprite.image.get_height(),
        "image":encoded_image,
        "sprite_ready":sprite.sprite_ready,
        "index":sprite.index,
        "speed":sprite.speed,
        "move_toMine":sprite.move_toMine,
        "move_toWall":sprite.move_toWall,
        "digging":sprite.digging,
        "repairing":sprite.repairing,
        "colour":sprite.colour}
        attributes_list.append(attributes)
    return attributes_list
```

```
def save_data(data):
    with open("saved_game.txt", "w") as saved_file:
        json.dump(data, saved_file)
```

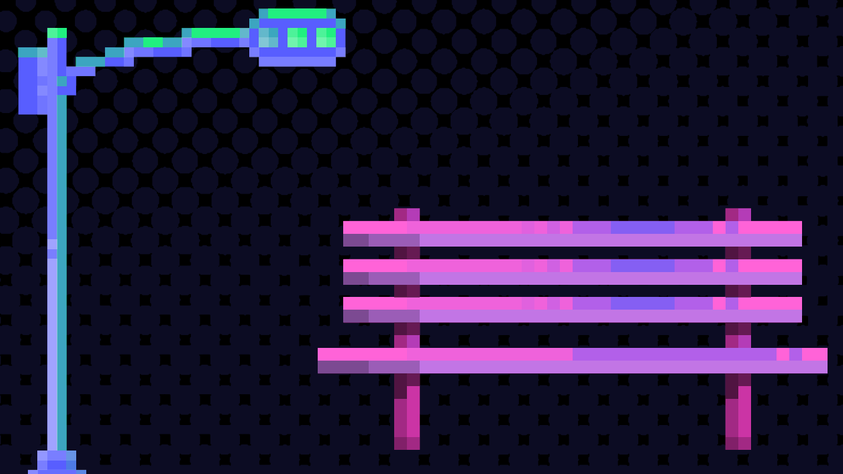


LOADING DATA

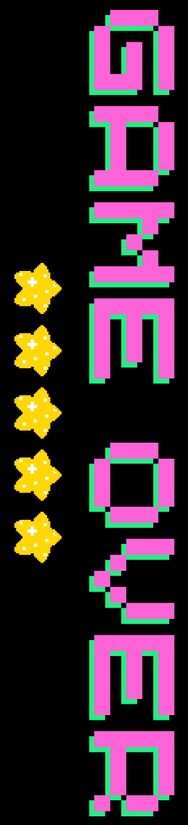
```
def load_data():
    try:
        with open ("saved_game.txt") as saved_file:
            data=json.load(saved_file)
            return data
    except FileNotFoundError:
        print("no data found")
        return None
```

```
def getData_values(data):
    global red_init_resources, blue_init_resources, red_turn, num_red
    global red_worker_turn_trained, red_archer_turn_trained, red_swor
    global RedKnight_group, RedArcher_Arrows_group, RedTower_Arrows_g
    red_init_resources = data.get("red_init_resources")
    blue_init_resources = data.get("blue_init_resources")
    red_turn = data.get("red_turn")
    num_redWorkers_toMine = data.get("num_redWorkers_toMine")
    num_blueWorkers_toMine = data.get("num_blueWorkers_toMine")
    num_redWorkers_toWall = data.get("num_redWorkers_toWall")
    num_blueWorkers_toWall = data.get("num_blueWorkers_toWall")
    red_worker_turn_trained = data.get("red_worker_turn_trained")
    red_archer_turn_trained = data.get("red_archer_turn_trained")
    red_sword_turn_trained = data.get("red_sword_turn_trained")
    blue_worker_turn_trained = data.get("blue_worker_turn_trained")
    blue_archer_turn_trained = data.get("blue_archer_turn_trained")
    blue_sword_turn_trained = data.get("blue_sword_turn_trained")
    getData_values_spritesCode(data)
    getData_RedWorker_attributes(data, RedWorker_group)
    getData_BlueWorker_attributes(data, BlueWorker_group)
    getData_RedArcher_attributes(data, RedArcher_group)
    getData_BlueArcher_attributes(data, BlueArcher_group)
    getData_RedKnight_attributes(data, RedKnight_group)
    getData_BlueKnight_attributes(data, BlueKnight_group)
```

```
def getWorker_attributes(sprite, worker):
    worker.rect.x = sprite.get("rect.x")
    worker.rect.y = sprite.get("rect.y")

    encoded_image = sprite.get("image")
    image_string = base64.b64decode(encoded_image)
    image_width=sprite.get("image_width")
    image_height=sprite.get("image_height")
    worker.image = pygame.image.fromstring(image_string, (image_width, image_height), 'RGBA')

    worker.sprite_ready = sprite.get("sprite_ready")
    worker.index = sprite.get("index")
    worker.speed = sprite.get("speed")
    worker.move_toMine = sprite.get("move_toMine")
    worker.move_toWall = sprite.get("move_toWall")
    worker.digging = sprite.get("digging")
    worker.repairing = sprite.get("repairing")
    worker.colour = sprite.get("colour")
```



```
def gameOver_text(winner, home_text, RedWorker_group, RedArcher_group, RedKnight_group, RedArcher_Arrows_group, RedTower_Arrows_group,
                  BlueWorker_group, BlueArcher_group, BlueKnight_group, BlueArcher_Arrows_group, BlueTower_Arrows_group):
    if winner=='red':
        game_display.blit(RedWins, (370, 100))
    if winner=='blue':
        game_display.blit(BlueWins, (370, 100))
    if not winner:
        game_display.blit(draw_text, (500, 100))

    button = pygame.Rect(500, 200, 130, 40)
    mouse_pos = pygame.mouse.get_pos()
    if button.collidepoint(mouse_pos):
        pygame.draw.rect(game_display,(100,100,100), button)

        RedWorker_group.empty()
        RedArcher_group.empty()
        RedKnight_group.empty()
        RedArcher_Arrows_group.empty()
        RedTower_Arrows_group.empty()

        BlueWorker_group.empty()
        BlueArcher_group.empty()
        BlueKnight_group.empty()
        BlueArcher_Arrows_group.empty()
        BlueTower_Arrows_group.empty()
    else:
        pygame.draw.rect(game_display,(50,50,50),button)
    game_display.blit(home_text, (535, 205))
    return button
```




RESOURCES

HTTPS://WWW.GEEKSFORGEEKS.ORG/PYTHON-DISPLAY-TEXT-TO-PYGAME-WINDOW/
HTTPS://WWW.YOUTUBE.COM/WATCH?V=POELPWWQXEK&LIST=PLI8CNIMH7OBFN3CX8K_GSBVIROVHKI2GZ
HTTPS://WWW.PROGRAMIZ.COM/PYTHON-PROGRAMMING/GLOBAL-KEYWORD
HTTPS://WWW.PYGAME.ORG/DOCS/REF/DRAW.HTML?HIGHLIGHT=PYGAME%20RECT#PYGAME.DRAW.RECT
HTTPS://WWW.PYGAME.ORG/DOCS/REF/TIME.HTML?HIGHLIGHT=TIME#MODULE-PYGAME.TIME
HTTPS://WWW.YOUTUBE.COM/WATCH?V=4TFZJHW0J-8 HTTP://WWW.CODINGWITHRUSS.COM/PYGAME/SPRITE-CLASS-AND-SPRITE-GROUPS-EXPLAINED/
HTTPS://WWW.REDDIT.COM/R/PYGAME/COMMENTS/13FSYY7/GET_SPRITES_FROM_GROUP/
HTTPS://YOUTU.BE/MYAXPA_EZS0 HTTPS://WWW.PYGAME.ORG/DOCS/REF/SPRITE.HTML
HTTPS://YOUTU.BE/HDU8MCALY4E HTTPS://WWW.PYGAME.ORG/DOCS/REF/EVENT.HTML
HTTPS://CODERSLEGACY.COM/PYTHON/PYGAME-SPRITE-COLLISION-DETECTION/
HTTPS://YOUTU.BE/JMPA7TU_0MS
HTTPS://API.ARCADE.ACADEMY/EN/2.6.0/EXAMPLES/SPRITE_BULLETS_ENEMY_AIMS.HTML
HTTPS://WWW.CODEPILE.NET/PILE/XVDLGQV1
HTTPS://WWW.PYGAME.ORG/DOCS/REF/SPRITE.HTML#PYGAME.SPRITE.SPRITECOLLIDE
HTTPS://WWW.YOUTUBE.COM/WATCH?V=RSL87L0OXDE HTTPS://WWW.YOUTUBE.COM/WATCH?V=0UM-JEQ7FAA
HTTPS://GITHUB.COM/PICKRV/PROGRAMMINGKNOWLEDGE/BLOB/MASTER/BUTTON.PY
HTTPS://WWW.YOUTUBE.COM/WATCH?V=__MZ0-S8PPM&T=9S
HTTPS://WWW.PYGAME.ORG/DOCS/REF/IMAGE.HTML#PYGAME.IMAGE.TOSTRING
HTTPS://STACKOVERFLOW.COM/QUESTIONS/68585252/HOW-TO-CHECK-IF-A-GLOBAL-VARIABLE-EXISTS-AND-IF
-NOT-THEN-DEFINE-IT-AS-GLOBAL HTTPS://WWW.GEEKSFORGEEKS.ORG/PYGAME-FLIP-THE-IMAGE/
HTTPS://WWW.GEEKSFORGEEKS.ORG/PYGAME-FLIP-THE-IMAGE/