Final Project Documentation

Project topic:

Defense Homeland

Game Background:

You are a Great Warrior in Ancient Rome. Rome has been attacked by Croco Monster. Now is the time to show your power.

Game objective:

Defeat Croco Monster

Controls:

Use arrow keys to move the character.
Press SPACE to attack

Description:

This game design is mainly focus on the fighting. It is an optimized version. There is no bug in gaming. No animation conflict or mess.

Game Sprites

- ### Character

  Moving by arrow keys, attacking by SPACE button
  When leaving the screen, it will be kept inside the screen.
  There is no collision function. Instead, there are two detection functions to keep calculating the distance between enemy and character, and the face direction to the enemy. When the distance and direction is accepted by the character attack function, the damage will be valid.
  There are 5 animation states: Stand, Walk, Attack, Been Hit and Die.
  There are lockers and flags to maintain the synchronization of animations, so it will not mess up in different states.
  When character is attacked, it goes to “been hit” state. The locker will lock the key listener until the “been hit” animation end (About 10 frames). While the locker is True, character does not allow to do anything.

```
#stopped  0
#walking  1
#attack   2
#been hit 3
#die      4
self.state = 0
self.movable = True
self.attackLock = False
self.gameover = False
```

  Also, when character attacks, it goes to “attack” state. The locker will lock the

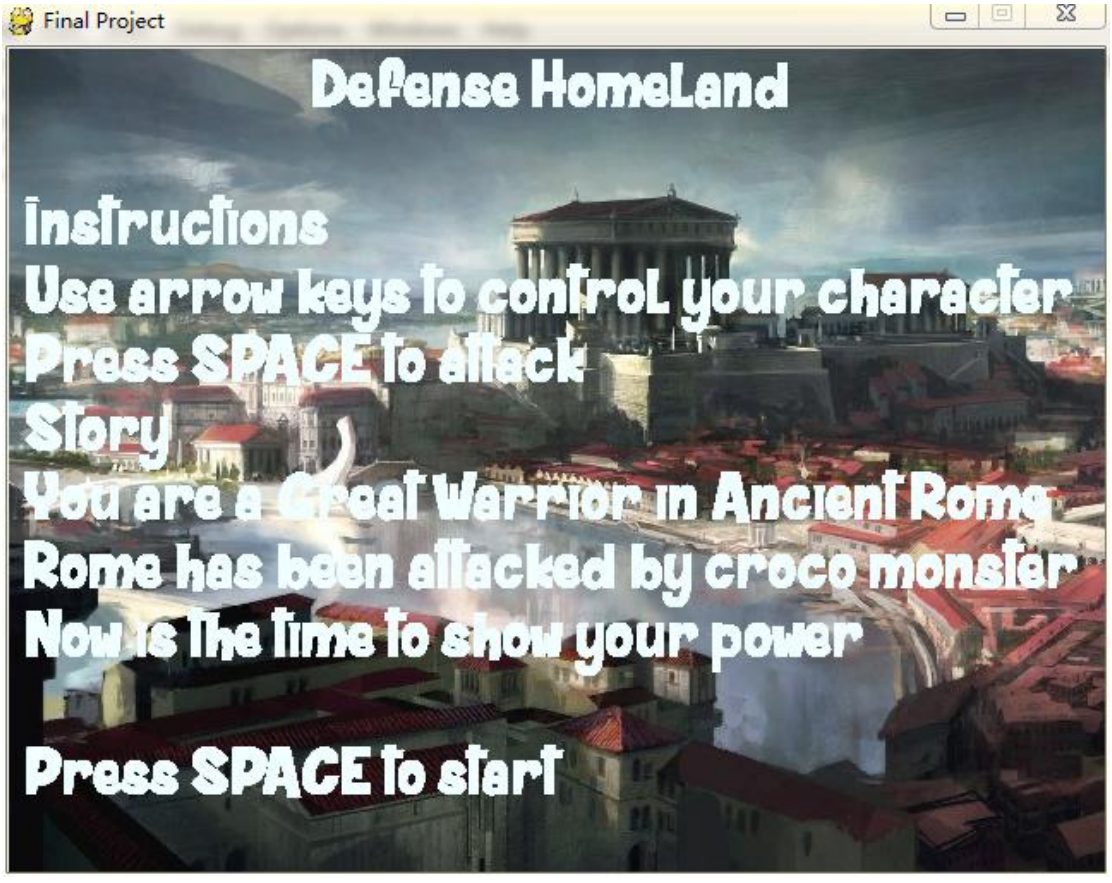
key listener until the “attack” animation end (About 10 frames).
Also for “die” state.
There are sound effects when character attacks, been hit and die.

- **Enemy**

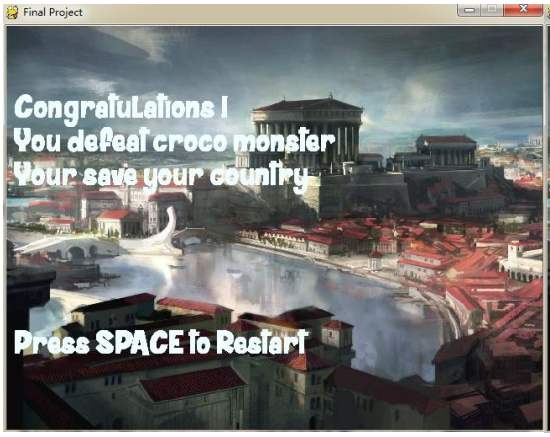
  It is created when game starts.
  It has a chasing function that keeps turning the direction to face the character. There is a turning chance number to control the reaction.
  Also, there is a distance detection function for the enemy. When the distance to character is accepted by the attack function, enemy will perform attack. There is an attack chance number to control the reaction.
  There are also 5 animation states: Stand, Walk, Attack, Been Hit and Die.
  There are also lockers and flags to maintain the synchronization of animations.
  When enemy is attacked, it goes to “been hit” state. The locker will lock the key listener until the “been hit” animation end (About 10 frames). While the locker is True, enemy does not allow to do anything.
  Also, when enemy attacks, it goes to “attack” state. The locker will lock the key listener until the “attack” animation end (About 10 frames).
  Also for “die” state.
  There are sound effects when enemy attacks, been hit and die.

Game Interface:

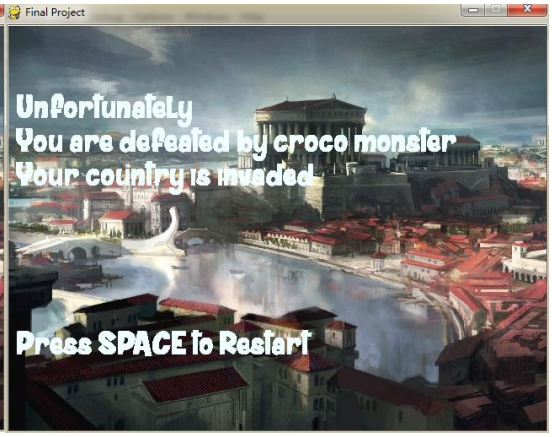
Game start scene



Game Win scene



Game over scene

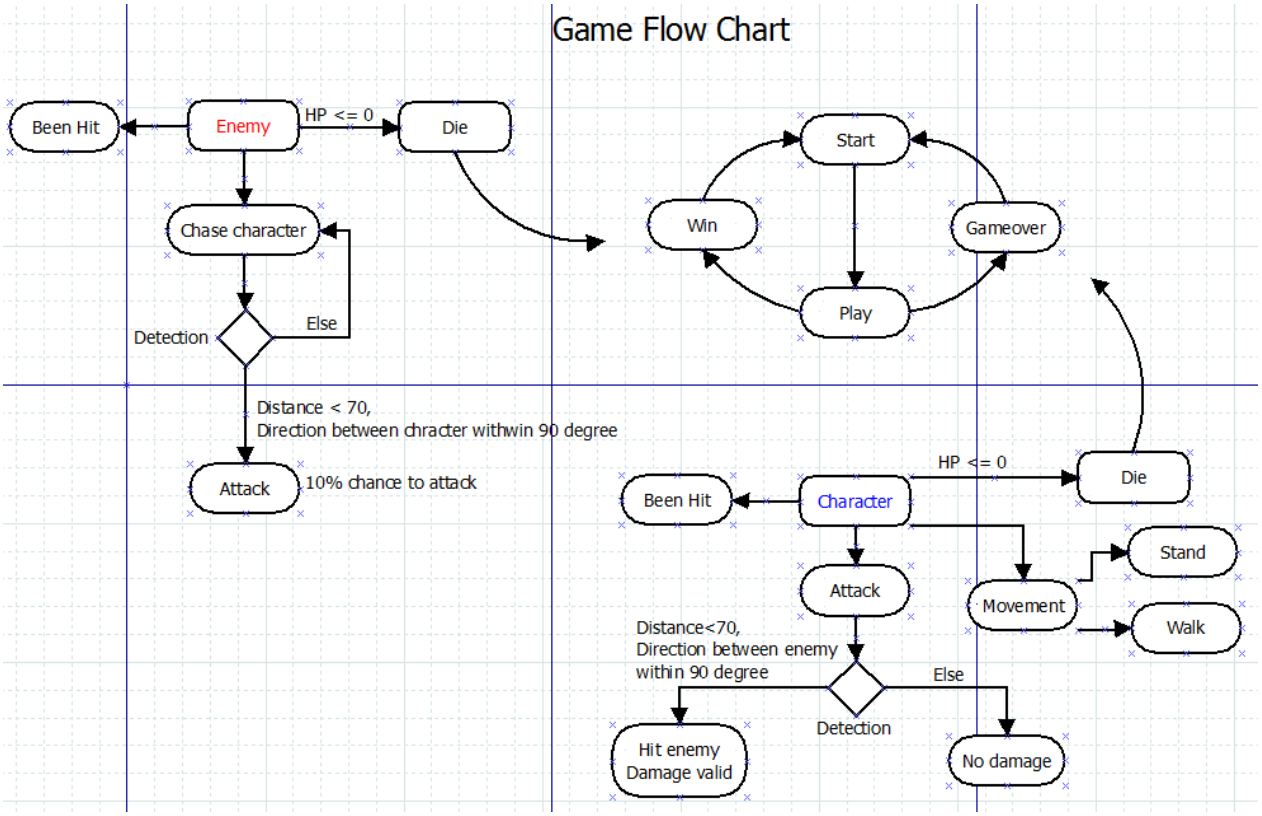


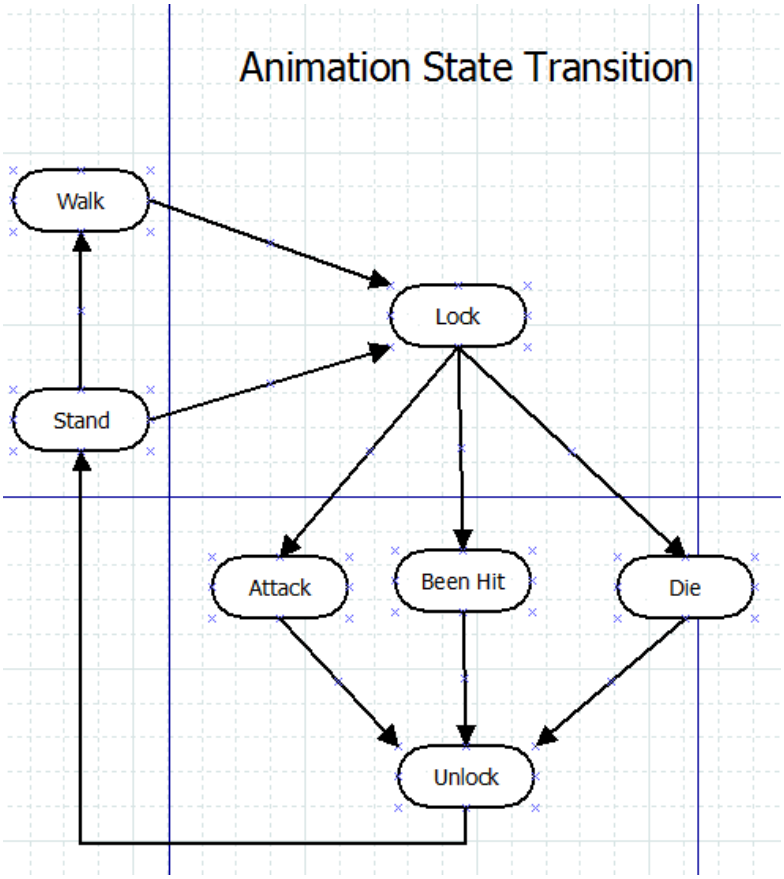
Game play scene

Left Health Bar: character

Right Health Bar: Croco Monster

Game Flow Chart





Critical Functions:

Animation function under character class: assign state and flag to different animations

```
    def animation(self)
#character attack enemy
def characterAttack(character,enemy,eHP,d):

#enemy attack character
def enemyAttack(character, enemy, cHP, d):

#enemy chase character
def chase(character, enemy, d):
```

```
    def animation(self):
        self.pause -= 1
        if self.pause <= 0:
            self.pause = self.delay

            #stopped
            if self.state == 0:
                self.image = self.stoppedList[self.dir]

            #walking
            if self.state == 1:
                self.frame += 1
                if self.frame > 7:
                    self.frame = 0
                self.image = self.walkingList[self.dir][self.frame]
            #attack
            if self.state == 2:
                self.movable = False
                self.frame += 1
                if self.frame < 13:
                    self.image = self.attackList[self.dir][self.frame]
                else:
                    self.attackLock = False
                    self.state = 0
```

```
            self.movable = True
        #been hit
        if self.state == 3:
            self.movable = False
            self.frame += 1
            if self.frame < 9:
                self.image = self.beenHitList[self.dir][self.frame]
            else:
                self.state = 0
                self.movable = True
        #die
        if self.state == 4:
            self.movable = False
            self.frame += 1
            if self.frame < 11:
                self.image = self.dieList[self.dir][self.frame]
            else:
                pygame.time.wait(2000)
                self.gameover = True
```

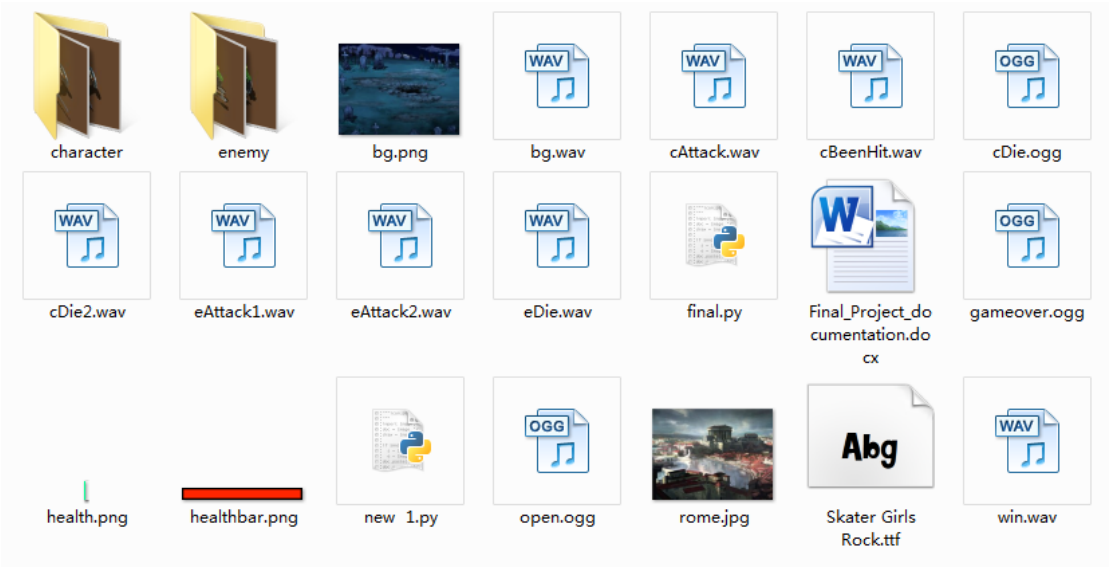
```
#character attack enemy
def characterAttack(character,enemy,eHP,d):
    #get the coordinate of two sprites
    cx = character.rect.centerx
    cy = character.rect.centery
    ex = enemy.rect.centerx
    ey = enemy.rect.centery
    #Character face detection
    #first quadrant
    #cx-ex>0 and cy-ey<0 and(s or sw or w)
    #Second quadrant
    #cx-ex<0 and cy-ey<0 and(s or se or e)
    #Third quadrant
    #cx-ex<0 and cy-ey>0 and(n or ne or e)
    #Fourth quadrant
    #cx-ex>0 and cy-ey>0 and(n or nw or w)

    cHitFlag = False
    #when character in attact state and distance within 70
    if character.state == 2 and d < 70 and enemy.state != 3 and enemy.state !=
4:
        if cx-ex >= 0 and cy-ey <= 0:
            if character.dir == 6 or character.dir == 5 or character.dir
== 4:
                cHitFlag = True
        if cx-ex<0 and cy-ey<=0:
            if character.dir == 6 or character.dir == 7 or character.dir
== 0:
                cHitFlag = True
        if cx-ex<0 and cy-ey>0:
            if character.dir == 2 or character.dir == 1 or character.dir
== 0:
                cHitFlag = True
        if cx-ex>=0 and cy-ey>0:
            if character.dir == 2 or character.dir == 3 or character.dir
== 4:
                cHitFlag = True
    #character attack can hit enemy
    if cHitFlag is True:
        character.sndBeenHit.play()
        eHP -= 10
        if eHP <= 0:
            enemy.frame = 0
            enemy.state = 4
```

```
            enemy.sndDie.play()
        else:
            character.attackLock = True
            enemy.frame = 0
            enemy.state = 3

    return eHP
```

```
#enemy attack character
def enemyAttack(character, enemy, cHP, d):
    if d < 70 and cHP > 0:
        #enemy attack reaction
        attachRate = random.random()
        if attachRate > 0.9:
            enemy.movable = False
            enemy.frame = 0
            enemy.state = 2
            enemy.sndAttack.play()
            cHP -= 20
            if cHP <= 0:
                character.sndDie.play()
                character.movable = False
                character.frame = 0
                character.state = 4
            else:
                character.movable = False
                character.frame = 0
                character.state = 3
    return cHP
```

```
#enemy chase character
def chase(character, enemy, d):
    if d < 70:
        enemy.ifChase = False
        enemy.state = 0
    else:
        enemy.ifChase = True
        enemy.state = 1
    #enemy chase reaction
    chaseRate = random.random()
    if chaseRate > 0.9:
        #enemy face detection
        #first quadrant
        #ex-cx>0 and ey-cy<0 and sw
        #Second quadrant
        #ex-cx<0 and ey-cy<0 and se
        #Third quadrant
        #ex-cx<0 and ey-cy>0 and ne
        #Fourth quadrant
        #ex-cx>0 and ey-cy>0 and nw
        cx = character.rect.centerx
        cy = character.rect.centery
        ex = enemy.rect.centerx
        ey = enemy.rect.centery
        if ex-cx>0 and ey-cy<0:
            enemy.dir = 5
        if ex-cx<0 and ey-cy<0:
            enemy.dir = 7
        if ex-cx<0 and ey-cy>0:
            enemy.dir = 1
        if ex-cx>0 and ey-cy>0:
            enemy.dir = 3

        if ex-cx==0 and ey-cy<0:
            enemy.dir = 6
        if ex-cx<0 and ey-cy==0:
            enemy.dir = 0
        if ex-cx==0 and ey-cy>0:
            enemy.dir = 2
        if ex-cx>0 and ey-cy==0:
            enemy.dir = 4
```

Related Files



Future improvement

1. Add levels
2. Add a Boss in each level.
3. Add more enemies
4. Add power up bonus (increase attack range, damage, defense and so no)
5. Add more skills.
6. Add run movement
7. Make the background movable. Game window will always focus on the character. Background move along with character.
8. Multiplayer mode.