Pseudo code:

```
import pygame
import math


Set up the display by 640 x 480

pygame.init()
screen = pygame.display.set_mode((640,480))

Make a class called Susie and initialize


Set up the image by loading sussane.png
Use convert_alpha()
And transform the image to the size of 75 x 150


create the corresponding rect
Set the x value to 320
Set the y value to 240

create the ability to move by making variables such as self.dx, self.dy, self.speed, self.angle,
self.angle_rad, self.d1 (cos), self.d2 (sin)

Create a variable called self.current_state and assign it an integer 1. This is to track which
update method to use

Define update:

if self.current_state is 1, call self.update1()
elif self.current_state is 2, call self.update2()
elif self.current_state is 3, call self.update3()
elif self.current_state is 4, call self.update4()
elif self.current_state is 5, call self.update5()

def update1(self): (going right)
Add self.dx to self.rect.centerx
if self.rect.right > screen.get_width():
Call update2()

def update2(self): (going left)
subtract self.dx from self.rect.centerx
if self.rect.left < 0:
```

```
Call update 3()

def update3(self): (Going down)
Add self.dy to self.rect.centery
if self.rect.bottom > screen.get_height():
Call update4()

def update4(self): (Going up)
Remove self.dy from self.rect.centery
if self.rect.top < 0:
Call update5()

def update5(self): (Moving at 30 degree)
Add self.d1 to self.rect.centerx
Add self.d2 to self.rect.centery

if self.rect.right > screen.get_width():
        self.rect.right = screen.get_width() (to stay in the screen/boundary)
Reverse x-direction to stay in the boundary while remaining in the boundary

if self.rect.left < 0:
        self.rect.left = 0 (To stay in the screen/boundary)
Reverse x-direction to stay in the boundary while remaining in the boundary

if self.rect.bottom > screen.get_height():
        self.rect.bottom = screen.get_height() (To stay in the boundary)
Reverse y-direction to stay in the boundary while remaining in the boundary

     if self.rect.top < 0:
        self.rect.top = 0 (To stay in the boundary)
Reverse y-direction to stay in the boundary while remaining in the boundary
```

def main():

```
Set up the caption
Load “heart.jpg” and set it as the background

Instantiate the sprite
Set the timer
Event handling
Refresh the display
```

```
if __name__ == "__main__":
    main()
    pygame.quit()
```