```
################################################################################
# STEP 1 - Game Loop
################################################################################
# Create a blank window with a game loop.
################################################################################

import pygame
import pygame.mouse
from pygame.locals import *
import sys
import directions
import game


pygame.mixer.pre_init(22050, -16, 2, 256)
pygame.init()
pygame.mixer.init()

screen_size = (640, 480)
screen = pygame.display.set_mode([screen_size[0] - 1, screen_size[1] - 1])
g = game.Game(screen_size, screen)


while True:
	g.clock.tick(1000/30)

	for event in pygame.event.get():
		# Pay attention if the user clicks the X to quit.
		if event.type == pygame.QUIT:
			sys.exit()

# End of game loop.
```

```
################################################################################
# STEP 2 - Sky
################################################################################
# Draw the sky on the screen.
################################################################################

import pygame
import pygame.mouse
from pygame.locals import *
import sys
import directions
import game


pygame.mixer.pre_init(22050, -16, 2, 256)
pygame.init()
pygame.mixer.init()

screen_size = (640, 480)
screen = pygame.display.set_mode([screen_size[0] - 1, screen_size[1] - 1])
g = game.Game(screen_size, screen)

while True:
       g.clock.tick(1000/30)

       for event in pygame.event.get():
              # Pay attention if the user clicks the X to quit.
              if event.type == pygame.QUIT:
                     sys.exit()

       # Draw the sky.
       g.sky.render(g.screen)

       # Put the scene on the monitor.
       pygame.display.update()

# End of game loop.
```

```
###############################################################################
# STEP 3 - Asteroids
###############################################################################
# Draw the asteroid field on the screen.
###############################################################################

import pygame
import pygame.mouse
from pygame.locals import *
import sys
import directions
import game


pygame.mixer.pre_init(22050, -16, 2, 256)
pygame.init()
pygame.mixer.init()

screen_size = (640, 480)
screen = pygame.display.set_mode([screen_size[0] - 1, screen_size[1] - 1])
g = game.Game(screen_size, screen)

while True:
       g.clock.tick(1000/30)

       for event in pygame.event.get():
              # Pay attention if the user clicks the X to quit.
              if event.type == pygame.QUIT:
                     sys.exit()

       # Draw the sky.
       g.sky.render(g.screen)

       # Create new asteroids if needed
       g.field.create_asteroids()
       # Move and animate asteroids
       g.field.move_asteroids()
       # Draw asteroids
       g.field.render(g.screen)

       # Put the scene on the monitor.
       pygame.display.update()

# End of game loop.
```

```
################################################################################
# STEP 4 - Earth
################################################################################
# Draw Earth in the center of the screen
################################################################################

import pygame
import pygame.mouse
from pygame.locals import *
import sys
import directions
import game


pygame.mixer.pre_init(22050, -16, 2, 256)
pygame.init()
pygame.mixer.init()

screen_size = (640, 480)
screen = pygame.display.set_mode([screen_size[0] - 1, screen_size[1] - 1])
g = game.Game(screen_size, screen)

while True:
	g.clock.tick(1000/30)

	for event in pygame.event.get():
		# Pay attention if the user clicks the X to quit.
		if event.type == pygame.QUIT:
			sys.exit()

	# Draw the sky.
	g.sky.render(g.screen)

	# Draw the earth
	g.earth.render(g.screen)

	# Create new asteroids if needed
	g.field.create_asteroids()
	# Move and animate asteroids
	g.field.move_asteroids()
	# Draw asteroids
	g.field.render(g.screen)

	# Put the scene on the monitor.
	pygame.display.update()

# End of game loop.
```

```
################################################################################
# STEP 5 - Mouse Control
################################################################################
# Control Earth's aiming system with the mouse.
################################################################################

import pygame
import pygame.mouse
from pygame.locals import *
import sys
import directions
import game


pygame.mixer.pre_init(22050, -16, 2, 256)
pygame.init()
pygame.mixer.init()

screen_size = (640, 480)
screen = pygame.display.set_mode([screen_size[0] - 1, screen_size[1] - 1])
g = game.Game(screen_size, screen)

while True:
	g.clock.tick(1000/30)

	for event in pygame.event.get():
		# Pay attention if the user clicks the X to quit.
		if event.type == pygame.QUIT:
			sys.exit()

		# Watch for mouse movement.  Tell the earth where the mouse is pointing.
		if event.type == pygame.MOUSEMOTION:
			mousepos = pygame.mouse.get_pos()
			a = directions.angle_of_point(mousepos, g.earth.origin)
			g.earth.set_angle(a)

	# Draw the sky.
	g.sky.render(g.screen)

	# Draw the earth
	g.earth.render(g.screen)

	# Create new asteroids if needed
	g.field.create_asteroids()
	# Move and animate asteroids
	g.field.move_asteroids()
	# Draw asteroids
	g.field.render(g.screen)

	# Put the scene on the monitor.
```

```
        pygame.display.update()

# End of game loop.
```

```
###############################################################################
# STEP 6 - Keyboard states
###############################################################################
# Read key states from the keyboard.
# Fire a weapon while a key is pressed.
###############################################################################

import pygame
import pygame.mouse
from pygame.locals import *
import sys
import directions
import game


pygame.mixer.pre_init(22050, -16, 2, 256)
pygame.init()
pygame.mixer.init()

screen_size = (640, 480)
screen = pygame.display.set_mode([screen_size[0] - 1, screen_size[1] - 1])
g = game.Game(screen_size, screen)

while True:
	g.clock.tick(1000/30)

	for event in pygame.event.get():
		# Pay attention if the user clicks the X to quit.
		if event.type == pygame.QUIT:
			sys.exit()

		# Watch for mouse movement.  Tell the earth where the mouse is pointing.
		if event.type == pygame.MOUSEMOTION:
			mousepos = pygame.mouse.get_pos()
			a = directions.angle_of_point(mousepos, g.earth.origin)
			g.earth.set_angle(a)

	# Check the keyboard for button states. (These buttons can be held down.)
	keys = pygame.key.get_pressed()
	btn_f = keys[K_f]

	# Fire weapons (for these weapons, you can hold down the button).
	if (btn_f):
		g.weapons.fire_gun(g.earth.angle, g.earth.launch_position)

	# Draw the sky.
	g.sky.render(g.screen)

	# Draw the earth
	g.earth.render(g.screen)
```

```
        # Move and animate weapons
        g.weapons.move_weapons()

        # Draw weapons
        g.weapons.render(g.screen)

        # Create new asteroids if needed
        g.field.create_asteroids()
        # Move and animate asteroids
        g.field.move_asteroids()
        # Draw asteroids
        g.field.render(g.screen)

        # Put the scene on the monitor.
        pygame.display.update()

# End of game loop.
```

```
################################################################################
# STEP 7 - Keyboard key presses
################################################################################
# Watch for key presses on the keyboard.
# Fire a weapon once for each key press.
################################################################################

import pygame
import pygame.mouse
from pygame.locals import *
import sys
import directions
import game


pygame.mixer.pre_init(22050, -16, 2, 256)
pygame.init()
pygame.mixer.init()

screen_size = (640, 480)
screen = pygame.display.set_mode([screen_size[0] - 1, screen_size[1] - 1])
g = game.Game(screen_size, screen)

while True:
	g.clock.tick(1000/30)

	btn_d = False
	btn_s = False
	btn_a = False

	for event in pygame.event.get():
		# Pay attention if the user clicks the X to quit.
		if event.type == pygame.QUIT:
			sys.exit()

		# Watch for mouse movement.  Tell the earth where the mouse is pointing.
		if event.type == pygame.MOUSEMOTION:
			mousepos = pygame.mouse.get_pos()
			a = directions.angle_of_point(mousepos, g.earth.origin)
			g.earth.set_angle(a)

		# Check the keyboard for keypresses. (These buttons must be pressed 
repeatedly.)
		if event.type == pygame.KEYDOWN:
			if (event.key == K_d):
				btn_d = True
			if (event.key == K_s):
				btn_s = True
			if (event.key == K_a):
				btn_a = True


	# Check the keyboard for button states. (These buttons can be held down.)
	keys = pygame.key.get_pressed()
	btn_f = keys[K_f]

	# Fire weapons (for these weapons, you can hold down the button).
```

```
        if (btn_f):
                g.weapons.fire_gun(g.earth.angle, g.earth.launch_position)
        if (btn_d):
                g.weapons.fire_multigun(g.earth.angle, g.earth.launch_position)
        if (btn_s):
                g.weapons.fire_missile(g.earth.angle, g.earth.launch_position)
        if (btn_a):
                g.weapons.fire_multimissile(g.earth.angle, g.earth.launch_position)

        # Draw the sky.
        g.sky.render(g.screen)

        # Draw the earth
        g.earth.render(g.screen)

        # Move and animate weapons
        g.weapons.move_weapons()

        # Draw weapons
        g.weapons.render(g.screen)

        # Create new asteroids if needed
        g.field.create_asteroids()
        # Move and animate asteroids
        g.field.move_asteroids()
        # Draw asteroids
        g.field.render(g.screen)

        # Put the scene on the monitor.
        pygame.display.update()

# End of game loop.
```

```
################################################################################
# STEP 8 - Asteroid Collisions
################################################################################
# Check for collisions between asteroids and weapons
################################################################################

import pygame
import pygame.mouse
from pygame.locals import *
import sys
import directions
import game


pygame.mixer.pre_init(22050, -16, 2, 256)
pygame.init()
pygame.mixer.init()

screen_size = (640, 480)
screen = pygame.display.set_mode([screen_size[0] - 1, screen_size[1] - 1])
g = game.Game(screen_size, screen)

while True:
	g.clock.tick(1000/30)

	btn_d = False
	btn_s = False
	btn_a = False

	for event in pygame.event.get():
		# Pay attention if the user clicks the X to quit.
		if event.type == pygame.QUIT:
			sys.exit()

		# Watch for mouse movement.  Tell the earth where the mouse is pointing.
		if event.type == pygame.MOUSEMOTION:
			mousepos = pygame.mouse.get_pos()
			a = directions.angle_of_point(mousepos, g.earth.origin)
			g.earth.set_angle(a)

		# Check the keyboard for keypresses. (These buttons must be pressed
repeatedly.)
		if event.type == pygame.KEYDOWN:
			if (event.key == K_d):
				btn_d = True
			if (event.key == K_s):
				btn_s = True
			if (event.key == K_a):
				btn_a = True
```

```
        # Check the keyboard for button states. (These buttons can be held down.)
        keys = pygame.key.get_pressed()
        btn_f = keys[K_f]

        # Fire weapons (for these weapons, you can hold down the button).
        if (btn_f):
                g.weapons.fire_gun(g.earth.angle, g.earth.launch_position)
        if (btn_d):
                g.weapons.fire_multigun(g.earth.angle, g.earth.launch_position)
        if (btn_s):
                g.weapons.fire_missile(g.earth.angle, g.earth.launch_position)
        if (btn_a):
                g.weapons.fire_multimissile(g.earth.angle, g.earth.launch_position)

        # Draw the sky.
        g.sky.render(g.screen)

        # Draw the earth
        g.earth.render(g.screen)

        # Look for collisions before drawing anything.
        # First, look for collisions between asteroids and weapons.
        clist = g.collision_handler.get_weapon_collisions(g.field, g.weapons)
        if (len(clist) > 0):
                for c in clist:
                        asteroid, weapon = c
                        g.collision_handler.handle_weapon_collision(asteroid, weapon, g.field,
g.weapons)

        # Move and animate weapons
        g.weapons.move_weapons()

        # Draw weapons
        g.weapons.render(g.screen)

        # Create new asteroids if needed
        g.field.create_asteroids()
        # Move and animate asteroids
        g.field.move_asteroids()
        # Draw asteroids
        g.field.render(g.screen)

        # Put the scene on the monitor.
        pygame.display.update()

# End of game loop.
```

```
###############################################################################
# STEP 9 - Earth Collisions
###############################################################################
# Check for collisions between asteroids and Earth
###############################################################################

import pygame
import pygame.mouse
from pygame.locals import *
import sys
import directions
import game


pygame.mixer.pre_init(22050, -16, 2, 256)
pygame.init()
pygame.mixer.init()

screen_size = (640, 480)
screen = pygame.display.set_mode([screen_size[0] - 1, screen_size[1] - 1])
g = game.Game(screen_size, screen)

while True:
	g.clock.tick(1000/30)

	btn_d = False
	btn_s = False
	btn_a = False

	for event in pygame.event.get():
		# Pay attention if the user clicks the X to quit.
		if event.type == pygame.QUIT:
			sys.exit()

		# Watch for mouse movement.  Tell the earth where the mouse is pointing.
		if event.type == pygame.MOUSEMOTION:
			mousepos = pygame.mouse.get_pos()
			a = directions.angle_of_point(mousepos, g.earth.origin)
			g.earth.set_angle(a)

		# Check the keyboard for keypresses. (These buttons must be pressed
repeatedly.)
		if event.type == pygame.KEYDOWN:
			if (event.key == K_d):
				btn_d = True
			if (event.key == K_s):
				btn_s = True
			if (event.key == K_a):
				btn_a = True
```

```
        # Check the keyboard for button states. (These buttons can be held down.)
        keys = pygame.key.get_pressed()
        btn_f = keys[K_f]

        # Fire weapons (for these weapons, you can hold down the button).
        if (btn_f):
                g.weapons.fire_gun(g.earth.angle, g.earth.launch_position)
        if (btn_d):
                g.weapons.fire_multigun(g.earth.angle, g.earth.launch_position)
        if (btn_s):
                g.weapons.fire_missile(g.earth.angle, g.earth.launch_position)
        if (btn_a):
                g.weapons.fire_multimissile(g.earth.angle, g.earth.launch_position)

        # Draw the sky.
        g.sky.render(g.screen)

        # Draw the earth
        g.earth.render(g.screen)

        # Look for collisions before drawing anything.
        # First, look for collisions between asteroids and weapons.
        clist = g.collision_handler.get_weapon_collisions(g.field, g.weapons)
        if (len(clist) > 0):
                for c in clist:
                        asteroid, weapon = c
                        g.collision_handler.handle_weapon_collision(asteroid, weapon, g.field,
g.weapons)

        # Next, look for collisions between asteroids and earth.
        eclist = g.collision_handler.get_earth_collisions(g.field, g.earth)
        if (len(eclist) > 0):
                for ec in eclist:
                        asteroid, earth = ec
                        g.collision_handler.handle_earth_collisions(g.field, asteroid, earth)

        # Move and animate weapons
        g.weapons.move_weapons()

        # Draw weapons
        g.weapons.render(g.screen)

        # Create new asteroids if needed
        g.field.create_asteroids()
        # Move and animate asteroids
        g.field.move_asteroids()
        # Draw asteroids
        g.field.render(g.screen)
        # Put the scene on the monitor.
        pygame.display.update()

# End of game loop.
```

```
################################################################################
# STEP 10 - Heads-Up Display (HUD)
################################################################################
# Show the HUD.
################################################################################

import pygame
import pygame.mouse
from pygame.locals import *
import sys
import directions
import game


pygame.mixer.pre_init(22050, -16, 2, 256)
pygame.init()
pygame.mixer.init()

screen_size = (640, 480)
screen = pygame.display.set_mode([screen_size[0] - 1, screen_size[1] - 1])
g = game.Game(screen_size, screen)

while True:
	g.clock.tick(1000/30)

	btn_d = False
	btn_s = False
	btn_a = False

	for event in pygame.event.get():
		# Pay attention if the user clicks the X to quit.
		if event.type == pygame.QUIT:
			sys.exit()

		# Watch for mouse movement.  Tell the earth where the mouse is pointing.
		if event.type == pygame.MOUSEMOTION:
			mousepos = pygame.mouse.get_pos()
			a = directions.angle_of_point(mousepos, g.earth.origin)
			g.earth.set_angle(a)

		# Check the keyboard for keypresses. (These buttons must be pressed
repeatedly.)
		if event.type == pygame.KEYDOWN:
			if (event.key == K_d):
				btn_d = True
			if (event.key == K_s):
				btn_s = True
			if (event.key == K_a):
				btn_a = True
```

```
        # Check the keyboard for button states. (These buttons can be held down.)
        keys = pygame.key.get_pressed()
        btn_f = keys[K_f]

        # Fire weapons (for these weapons, you can hold down the button).
        if (btn_f):
                g.weapons.fire_gun(g.earth.angle, g.earth.launch_position)
        if (btn_d):
                g.weapons.fire_multigun(g.earth.angle, g.earth.launch_position)
        if (btn_s):
                g.weapons.fire_missile(g.earth.angle, g.earth.launch_position)
        if (btn_a):
                g.weapons.fire_multimissile(g.earth.angle, g.earth.launch_position)

        # Draw the sky.
        g.sky.render(g.screen)

        # Draw the earth
        g.earth.render(g.screen)

        # Look for collisions before drawing anything.
        # First, look for collisions between asteroids and weapons.
        clist = g.collision_handler.get_weapon_collisions(g.field, g.weapons)
        if (len(clist) > 0):
                for c in clist:
                        asteroid, weapon = c
                        g.collision_handler.handle_weapon_collision(asteroid, weapon, g.field,
g.weapons)

        # Next, look for collisions between asteroids and earth.
        eclist = g.collision_handler.get_earth_collisions(g.field, g.earth)
        if (len(eclist) > 0):
                for ec in eclist:
                        asteroid, earth = ec
                        g.collision_handler.handle_earth_collisions(g.field, asteroid, earth)

        # Move and animate weapons
        g.weapons.move_weapons()

        # Draw weapons
        g.weapons.render(g.screen)

        # Create new asteroids if needed
        g.field.create_asteroids()
        # Move and animate asteroids
        g.field.move_asteroids()
        # Draw asteroids
        g.field.render(g.screen)
        # Draw the Heads Up Display (HUD)
        g.hud.render(g.screen)

        # Put the scene on the monitor.
        pygame.display.update()

# End of game loop.
```

```
################################################################################
# STEP 11 - Background Music
################################################################################
# Play an MP3 file in the background.
################################################################################

import pygame
import pygame.mouse
from pygame.locals import *
import sys
import directions
import game


pygame.mixer.pre_init(22050, -16, 2, 256)
pygame.init()
pygame.mixer.init()

screen_size = (640, 480)
screen = pygame.display.set_mode([screen_size[0] - 1, screen_size[1] - 1])
g = game.Game(screen_size, screen)

g.sound_system.play_background()

while True:
	g.clock.tick(1000/30)

	btn_d = False
	btn_s = False
	btn_a = False

	for event in pygame.event.get():
		# Pay attention if the user clicks the X to quit.
		if event.type == pygame.QUIT:
			sys.exit()

		# Watch for mouse movement.  Tell the earth where the mouse is pointing.
		if event.type == pygame.MOUSEMOTION:
			mousepos = pygame.mouse.get_pos()
			a = directions.angle_of_point(mousepos, g.earth.origin)
			g.earth.set_angle(a)

		# Check the keyboard for keypresses. (These buttons must be pressed
repeatedly.)
		if event.type == pygame.KEYDOWN:
			if (event.key == K_d):
				btn_d = True
			if (event.key == K_s):
				btn_s = True
			if (event.key == K_a):
```

```
                    btn_a = True


    # Check the keyboard for button states. (These buttons can be held down.)
    keys = pygame.key.get_pressed()
    btn_f = keys[K_f]

    # Fire weapons (for these weapons, you can hold down the button).
    if (btn_f):
        g.weapons.fire_gun(g.earth.angle, g.earth.launch_position)
    if (btn_d):
        g.weapons.fire_multigun(g.earth.angle, g.earth.launch_position)
    if (btn_s):
        g.weapons.fire_missile(g.earth.angle, g.earth.launch_position)
    if (btn_a):
        g.weapons.fire_multimissile(g.earth.angle, g.earth.launch_position)

    # Draw the sky.
    g.sky.render(g.screen)

    # Draw the earth
    g.earth.render(g.screen)

    # Look for collisions before drawing anything.
    # First, look for collisions between asteroids and weapons.
    clist = g.collision_handler.get_weapon_collisions(g.field, g.weapons)
    if (len(clist) > 0):
        for c in clist:
            asteroid, weapon = c
            g.collision_handler.handle_weapon_collision(asteroid, weapon, g.field,
g.weapons)

    # Next, look for collisions between asteroids and earth.
    eclist = g.collision_handler.get_earth_collisions(g.field, g.earth)
    if (len(eclist) > 0):
        for ec in eclist:
            asteroid, earth = ec
            g.collision_handler.handle_earth_collisions(g.field, asteroid, earth)

    # Move and animate weapons
    g.weapons.move_weapons()

    # Draw weapons
    g.weapons.render(g.screen)

    # Create new asteroids if needed
    g.field.create_asteroids()
    # Move and animate asteroids
    g.field.move_asteroids()
    # Draw asteroids
```

```
        g.field.render(g.screen)

        # Draw the Heads Up Display (HUD)
        g.hud.render(g.screen)

        # Put the scene on the monitor.
        pygame.display.update()

# End of game loop.
```