```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
1   #!/usr/bin/python
2   # Forest Bomber
3   # Code Angel
4
5   import sys
6   import os
7   import pygame
8   from pygame.locals import *
9
10  # Define the colours
11  WHITE = (255, 255, 255)
12  PURPLE = (96, 85, 154)
13  LIGHT_BLUE = (157, 220, 241)
14  DARK_BLUE = (63, 111, 182)
15  GREEN = (57, 180, 22)
16
17  # Define constants
18  SCREEN_WIDTH = 640
19  SCREEN_HEIGHT = 480
20  SCOREBOARD_MARGIN = 4
21  LINE_HEIGHT = 18
22  BOX_WIDTH = 300
23  BOX_HEIGHT = 150
24
25  TOTAL_LEVELS = 4
26  MAX_TREES = 12
27  TREE_SPACING = 40
28  FIRST_TREE = 140
29  GROUND_HEIGHT = 8
30  TREE_OFF_GROUND = 4
31
32  PLANE_START_X = 0
33  PLANE_START_Y = 54
34
35  # Setup
36  os.environ['SDL_VIDEO_CENTERED'] = '1'
37  pygame.mixer.pre_init(44100, -16, 2, 512)
38  pygame.mixer.init()
39  pygame.init()
40  game_screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
41  pygame.display.set_caption('Forest Bomber')
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
clock = pygame.time.Clock()
font = pygame.font.SysFont('Helvetica', 16)

# Load images
background_image = pygame.image.load('background.png').convert()
tree_image = pygame.image.load('tree.png').convert_alpha()
burn_tree_image = pygame.image.load('burning_tree.png').convert_alpha()
plane_image = pygame.image.load('plane.png').convert_alpha()
burn_plane_image = pygame.image.load('burning_plane.png').convert_alpha()
bomb_image = pygame.image.load('bomb.png').convert_alpha()

# Load sounds
explosion_sound = pygame.mixer.Sound('explosion.ogg')
tree_sound = pygame.mixer.Sound('tree_explosion.ogg')

# Initialise variables
level = 1
score = 0
hi_score = 0
speed_boost = 0

plane_exploded = False
level_cleared = False
plane_front = 0
plane_explode_sound_played = False


bomb_dropped = False
bomb = bomb_image.get_rect()

plane = plane_image.get_rect()
plane.x = PLANE_START_X
plane.y = PLANE_START_Y

tree = tree_image.get_rect()
tree.y = SCREEN_HEIGHT - tree.height - TREE_OFF_GROUND

burning_tree = 0
tree_timer = 0

burning_trees = []
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
83
84  # Set up different forests for each level
85  forest_1 = ['T', '-', 'T', '-', '-', '-', 'T', '-', '-', '-', '-', 'T']
86  forest_2 = ['-', 'T', '-', '-', 'T', '-', 'T', '-', 'T', 'T', '-', 'T']
87  forest_3 = ['T', 'T', '-', '-', 'T', '-', 'T', 'T', 'T', 'T', '-', '-']
88  forest_4 = ['T', 'T', '-', '-', 'T', 'T', 'T', '-', 'T', 'T', 'T', '-']
89  forest = list(forest_1)
90
91  # Main game loop
92  while True:
93
94      for event in pygame.event.get():
95
96          # Space key pressed, drop bomb
97          key_pressed = pygame.key.get_pressed()
98          if key_pressed[pygame.K_SPACE]:
99              if bomb_dropped is False and level_cleared is False and plane_exploded is False:
100                 bomb_dropped = True
101                 bomb.x = plane.x + 15
102                 bomb.y = plane.y + 10
103
104         # Return key at end of game / level pressed
105         elif key_pressed[pygame.K_RETURN]:
106
107             # Plane has exploded or all levels completed - so go back to start
108             if plane_exploded is True or (level == TOTAL_LEVELS and level_cleared is True):
109                 plane_exploded = False
110                 plane_explode_sound_played = False
111                 score = 0
112                 speed_boost = 0
113                 level = 1
114                 forest = list(forest_1)
115                 plane.x = PLANE_START_X
116                 plane.y = PLANE_START_Y
117                 level_cleared = False
118
119             # Level cleared - go up 1 level and load a new forest
120             elif level_cleared is True:
121                 level += 1
122                 level_cleared = False
123
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
124                 if level == 2:
125                     forest = list(forest_2)
126                 elif level == 3:
127                     forest = list(forest_3)
128                     speed_boost = 1
129                 else:
130                     forest = list(forest_4)
131                     speed_boost = 1
132
133                 plane.x = PLANE_START_X
134                 plane.y = PLANE_START_Y
135
136             # User quits
137             if event.type == QUIT:
138                 pygame.quit()
139                 sys.exit()
140
141         # Update plane location
142         if level_cleared is False and plane_exploded is False:
143             plane.x = plane.x + 5 + speed_boost
144
145             if plane.x >= SCREEN_WIDTH:
146                 plane.x = 0
147                 plane.y += 100
148
149         # Update bomb location
150         if bomb_dropped is True:
151             bomb.y += 5
152             bomb.x += 3
153
154             if bomb.y > SCREEN_HEIGHT:
155                 bomb_dropped = False
156
157             if bomb.x > SCREEN_WIDTH:
158                 bomb_dropped = False
159
160             # Check if bomb has hit a tree
161             for column, forest_item in enumerate(forest):
162                 if forest_item == 'T':
163                     tree.x = FIRST_TREE + column * TREE_SPACING
164
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
                if bomb.colliderect(tree):
                    forest[column] = 'B'
                    bomb_dropped = False
                    burning_trees.append(column)
                    tree_timer = 10
                    score += 10 * level
                    tree_sound.play()

    # Update burning trees tree status
    if tree_timer > 0:
        tree_timer -= 1
        if tree_timer == 0:
            for column in burning_trees:
                forest[column] = '-'
            del burning_trees[:]

    # Plane on ground level
    if plane.y >= SCREEN_HEIGHT - plane.height - GROUND_HEIGHT:
        plane_front = plane.x + plane.width

        # Edge of the screen reached so level cleared
        if plane_front >= SCREEN_WIDTH:
            level_cleared = True

        # Check to see if plane has collided with a tree
        else:
            for column, forest_item in enumerate(forest):
                if forest_item == 'T' or forest_item == 'B':
                    tree_left = FIRST_TREE + column * TREE_SPACING
                    if plane_front >= tree_left:
                        plane_exploded = True

    # If score is greater than high score, then new high score
    if score > hi_score:
        hi_score = score

    # Draw background
    game_screen.blit(background_image, [0, 0])

    # Draw forest
    for column, forest_item in enumerate(forest):
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
206             tree.x = FIRST_TREE + column * TREE_SPACING
207             if forest_item == 'T':
208                 game_screen.blit(tree_image, [tree.x, tree.y])
209             elif forest_item == 'B':
210                 game_screen.blit(burn_tree_image, [tree.x, tree.y])
211
212         # Draw plane
213         if plane_exploded is False:
214             game_screen.blit(plane_image, [plane.x, plane.y])
215         else:
216             plane.y = SCREEN_HEIGHT - burn_plane_image.get_height() - TREE_OFF_GROUND
217             game_screen.blit(burn_plane_image, [plane.x, plane.y])
218
219         # Draw bomb
220         if bomb_dropped is True:
221             game_screen.blit(bomb_image, [bomb.x, bomb.y])
222
223         # Display scoreboard - score, level, high score
224         scoreboard_background_rect = (0, 0, SCREEN_WIDTH, LINE_HEIGHT + 2 * SCOREBOARD_MARGIN)
225         pygame.draw.rect(game_screen, LIGHT_BLUE, scoreboard_background_rect)
226
227         score_text = 'Score: ' + str(score)
228         text = font.render(score_text, True, PURPLE)
229         game_screen.blit(text, [SCOREBOARD_MARGIN, SCOREBOARD_MARGIN])
230
231         hi_text = 'Hi Score: ' + str(hi_score)
232         text = font.render(hi_text, True, PURPLE)
233         text_rect = text.get_rect()
234         game_screen.blit(text, [SCREEN_WIDTH - text_rect.width - SCOREBOARD_MARGIN, SCOREBOARD_MARGIN])
235
236         level_text = 'Level: ' + str(level)
237         text = font.render(level_text, True, PURPLE)
238         text_rect = text.get_rect()
239         game_screen.blit(text, [(SCREEN_WIDTH - text_rect.width) / 2, SCOREBOARD_MARGIN])
240
241         # End of game / level message
242         if plane_exploded is True or level_cleared is True:
243
244             if plane_exploded is True:
245                 text_line_1 = font.render('GAME OVER', True, WHITE)
246                 text_rect_1 = text_line_1.get_rect()
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
247
248             text_line_2 = font.render('RETURN for new game', True, WHITE)
249             text_rect_2 = text_line_2.get_rect()
250
251             if plane_explode_sound_played is False:
252                 explosion_sound.play()
253                 plane_explode_sound_played = True
254
255
256         elif level == TOTAL_LEVELS:
257             text_line_1 = font.render('GAME OVER - ALL LEVELS CLEARED', True, WHITE)
258             text_rect_1 = text_line_1.get_rect()
259
260             text_line_2 = font.render('RETURN for new game', True, WHITE)
261             text_rect_2 = text_line_2.get_rect()
262
263         else:
264             text_line_1 = font.render('LEVEL ' + str(level) + ' CLEARED', True, WHITE)
265             text_rect_1 = text_line_1.get_rect()
266
267             text_line_2 = font.render('RETURN for new level', True, WHITE)
268             text_rect_2 = text_line_2.get_rect()
269
270         # Display message box to sit text over
271         msg_bk_rect = ((SCREEN_WIDTH - BOX_WIDTH) / 2, (SCREEN_HEIGHT - BOX_HEIGHT) / 2, BOX_WIDTH, BOX_HEIGHT)
272         pygame.draw.rect(game_screen, DARK_BLUE, msg_bk_rect)
273
274         # Display 2 lines of text, centred
275         game_screen.blit(text_line_1, [(SCREEN_WIDTH - text_rect_1.width) / 2,
276                                        (SCREEN_HEIGHT - text_rect_1.height) / 2 - LINE_HEIGHT])
277         game_screen.blit(text_line_2, [(SCREEN_WIDTH - text_rect_2.width) / 2,
278                                        (SCREEN_HEIGHT - text_rect_2.height) / 2 + LINE_HEIGHT])
279
280     pygame.display.update()
281     clock.tick(30)
282
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```