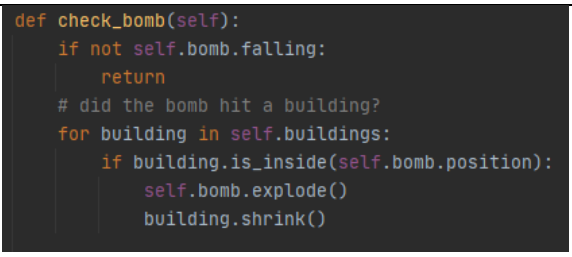
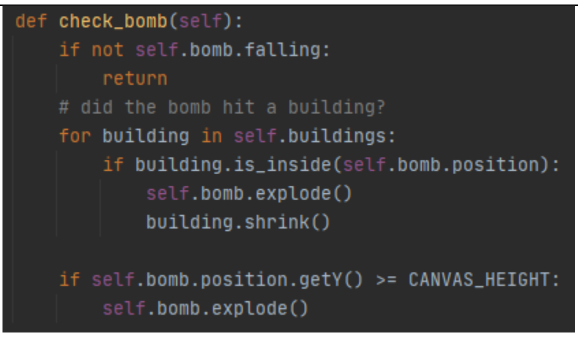
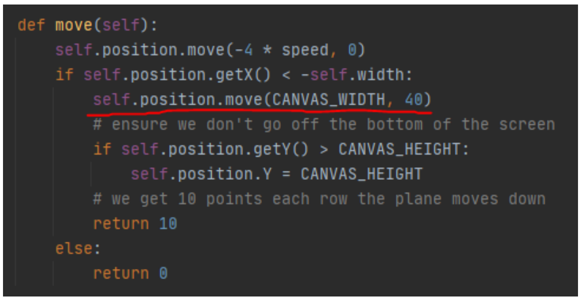
Bug Summary	Description	Intended Behaviour	How to Reproduce
If a bomb is missed, no more can be dropped	If a bomb misses a building (hits bottom of the screen), no more can drop from the plane, even after crossing the screen.	More bombs should be able to be dropped after 1 is missed and hits the bottom of the screen.	Drop a bomb so it falls between 2 buildings (doesn't matter which buildings). Once it hits the bottom of the screen and disappears, press space to try and drop more – no more will drop.
Not possible to hit the first building with a bomb	As the plane starts too far left on the screen (after resetting and returning to the right hand side of the screen), the first bomb cannot physically hit the first building.	Plane should start further right (offscreen) so that the player can hit the first building with a bomb.	As soon as the plane reappears on the right-hand side of the screen, press space. The bomb will hit the 2nd building, despite being dropped at the earliest possible moment.
Game over occurs without hitting any buildings	Once the plane has crossed the screen 5 times, without hitting any buildings the game ends	Game should only end if the plane hits a building	Wait for the plane to pass through the screen several times without hitting any buildings. Game will give "Game Over" message.
Game over occurs when the plane hits the ground	Once the plane is at "ground level" game over occurs	Should instead go to the next level	Destroy all buildings and wait until plane reaches ground level, upon which "game over" is displayed
Bomb can appear to hit side of building but does not register (in program) as a hit	If a bomb hits the corner/side of a building, it will continue falling (despite apparently hitting the building)	Should register as a hit to the building (bomb should disappear, building should reduce in height, points should be given)	Hit a building at the corner, the bomb will appear to hit the building but will continue travelling down through the side of the building.

<table>
<tr><th>Bug Summary</th><th>Bugged code</th><th>Fixed code</th></tr>
<tr><td>If a bomb is missed, no more can be dropped</td><td>
<pre>
def check_bomb(self):
    if not self.bomb.falling:
        return
    # did the bomb hit a building?
    for building in self.buildings:
        if building.is_inside(self.bomb.position):
            self.bomb.explode()
            building.shrink()
</pre>
In the "check_bomb" function it will only reset "falling" to false if the bomb hits a building, hence if the bomb hits the bottom of the screen (not hitting a building) it will still be considered "falling"</td><td>
<pre>
def check_bomb(self):
    if not self.bomb.falling:
        return
    # did the bomb hit a building?
    for building in self.buildings:
        if building.is_inside(self.bomb.position):
            self.bomb.explode()
            building.shrink()

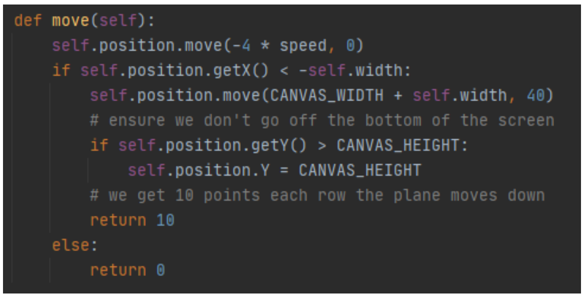
    if self.bomb.position.getY() >= CANVAS_HEIGHT:
        self.bomb.explode()
</pre>
"if" statement added to check if the bomb's y position exceeds the canvas height (hence the bomb has hit or passed the bottom of the screen)</td></tr>
</table>

Not possible to hit the first building with a bomb



```
def move(self):
    self.position.move(-4 * speed, 0)
    if self.position.getX() < -self.width:
        self.position.move(CANVAS_WIDTH, 40)
        # ensure we don't go off the bottom of the screen
        if self.position.getY() > CANVAS_HEIGHT:
            self.position.Y = CANVAS_HEIGHT
        # we get 10 points each row the plane moves down
        return 10
    else:
        return 0
```

When the plane reaches the left hand side of the screen, it currently resets to the canvas width (right hand side), however the plane is then drawn entirely on screen and hence is too far left to drop a bomb on the first building.

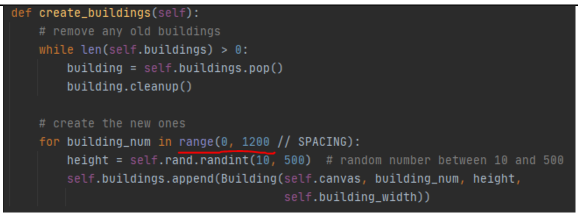


```
def move(self):
    self.position.move(-4 * speed, 0)
    if self.position.getX() < -self.width:
        self.position.move(CANVAS_WIDTH + self.width, 40)
        # ensure we don't go off the bottom of the screen
        if self.position.getY() > CANVAS_HEIGHT:
            self.position.Y = CANVAS_HEIGHT
        # we get 10 points each row the plane moves down
        return 10
    else:
        return 0
```

Plane now resets to canvas width + the width of the plane, hence it now starts offscreen and allows the user to drop a bomb on the first building.

Another bug created – cannot completely destroy buildings (too close to base of screen)

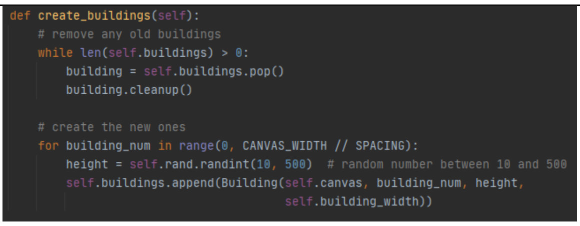
Game over occurs without hitting any buildings



```
def create_buildings(self):
    # remove any old buildings
    while len(self.buildings) > 0:
        building = self.buildings.pop()
        building.cleanup()

    # create the new ones
    for building_num in range(0, 1200 // SPACING):
        height = self.rand.randint(10, 500)  # random number between 10 and 500
        self.buildings.append(Building(self.canvas, building_num, height,
                                       self.building_width))
```

As the range is (0, 1200), 1200 is 200 more than the width of the screen, hence another invisible building is generated offscreen that the players cannot hit (as they cannot see it/bombs cannot drop offscreen), yet the plane's position can still hit it briefly when resetting the plane position to the right of the screen.

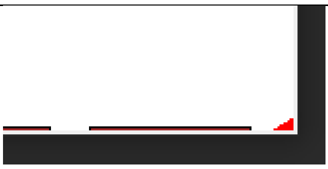


```
def create_buildings(self):
    # remove any old buildings
    while len(self.buildings) > 0:
        building = self.buildings.pop()
        building.cleanup()

    # create the new ones
    for building_num in range(0, CANVAS_WIDTH // SPACING):
        height = self.rand.randint(10, 500)  # random number between 10 and 500
        self.buildings.append(Building(self.canvas, building_num, height,
                                       self.building_width))
```

Range is now the width of the canvas, hence now buildings cannot generate off the left hand side of the screen.

Game over occurs when the plane hits the ground

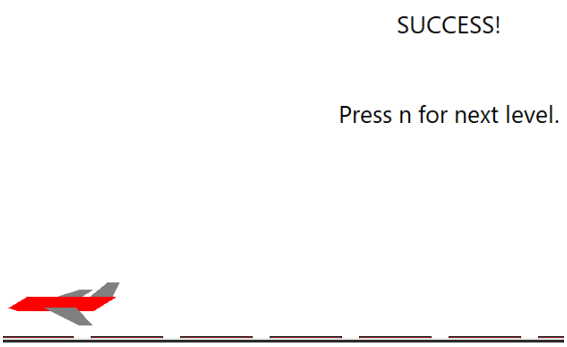


The building still exists with height 0, so when it travels across the ground it hits the "building" (despite the building being effectively destroyed).

2 potential solutions:

- Completely delete the building object once its height is 0, hence it can no longer be collided with
- Consider the plane landed when it's 1 step above – currently it is landed at the canvas height, it could land at canvas height – 50

I chose solution 2 (solution 1 would be adding an "if" statement so that if the height of the building = 0, function "cleanup" is called to delete the building)

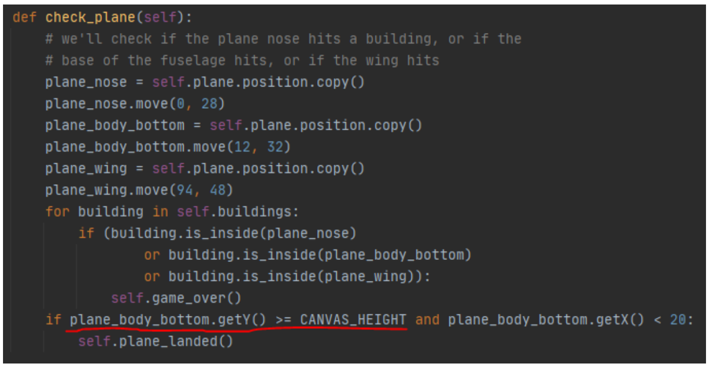


SUCCESS!

Press n for next level.

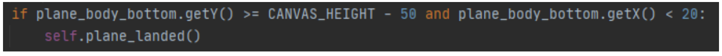
Plane now lands at this height, but as said solution 1 would allow it to land at canvas height if required

<table>
<tr>
<td></td>
<td></td>
<td>



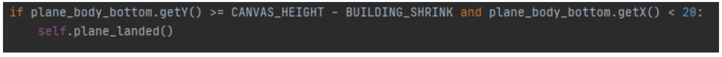
```
def check_plane(self):
    # we'll check if the plane nose hits a building, or if the
    # base of the fuselage hits, or if the wing hits
    plane_nose = self.plane.position.copy()
    plane_nose.move(0, 28)
    plane_body_bottom = self.plane.position.copy()
    plane_body_bottom.move(12, 32)
    plane_wing = self.plane.position.copy()
    plane_wing.move(94, 48)
    for building in self.buildings:
        if (building.is_inside(plane_nose)
                or building.is_inside(plane_body_bottom)
                or building.is_inside(plane_wing)):
            self.game_over()
    if plane_body_bottom.getY() >= CANVAS_HEIGHT and plane_body_bottom.getX() < 20:
        self.plane_landed()
```

Line underlined in red changed to



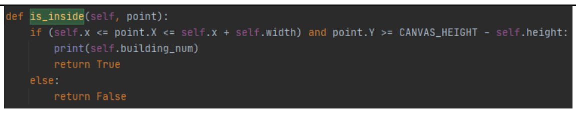
```
if plane_body_bottom.getY() >= CANVAS_HEIGHT - 50 and plane_body_bottom.getX() < 20:
    self.plane_landed()
```

As a further improvement, changed the value of ‘50’ to a constant ‘BUILDING_SHRINK’ so that the amount buildings are reduced by when a bomb hits and the level at which the plane lands are not out of sync.



```
if plane_body_bottom.getY() >= CANVAS_HEIGHT - BUILDING_SHRINK and plane_body_bottom.getX() < 20:
    self.plane_landed()
```

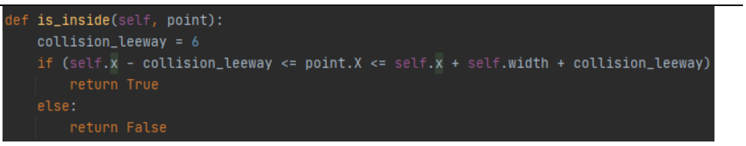
</td>
</tr>
<tr>
<td>Bomb can appear to hit side of building but does not register as a hit</td>
<td>



```
def is_inside(self, point):
    if (self.x <= point.X <= self.x + self.width) and point.Y >= CANVAS_HEIGHT - self.height:
        print(self.building_num)
        return True
    else:
        return False
```

The “is_inside” function requires the point of the bomb to be inside the building, however the bomb image is larger than a single point, hence it can appear the bomb hit the building despite the point not being within it.

</td>
<td>



```
def is_inside(self, point):
    collision_leeway = 6
    if (self.x - collision_leeway <= point.X <= self.x + self.width + collision_leeway)
        return True
    else:
        return False
```

Added a “collision leeway”, hence meaning that if the point is 6 units to either side of the building it will still be considered a hit. This still allows for bombs to be missed between buildings, but it no longer appears that bombs just on the edge travel through the sides of buildings.

</td>
</tr>
</table>