Assignment CA2.0

> Computer Science (Bsc), Chen Yang

Bug #1 - bomb

Summary

> The plane can't drop any more bombs if a bomb is missed.

Description

> The plane can't drop any more bombs if a bomb is missed.
>
> We can firstly track the function that drops the bomb, then we should notice it only drops bomb if `bomb.falling` is false. Therefore, we know we miss some update for the status of *bomb* falling. This is because in `check_bomb` function, it doesn't set a status for `bomb.falling` when a bomb doesn't hit the building.

What should happen

> When the bomb goes through the windows and doesn't hit any building, `bomb.falling` should return to false. And the player can drop the next bomb.

Fix code

> Add an if-statement to check whether the bomb doesn't hit any buildings.

```
    class Bomb():
             '''↓↓↓↓↓new code↓↓↓↓↓'''
        def miss(self):
            self.falling = False
                '''↑↑↑↑↑↑new code↑↑↑↑↑↑'''
```

```
class Display(Frame):
        def check_bomb(self):
            if not self.bomb.falling:

                return
            # if not hit the building reset the status of bomb
            '''↓↓↓↓↓new code↓↓↓↓↓'''
            if self.bomb.position.getY() >= CANVAS_HEIGHT:
                self.bomb.miss()
                return
            '''↑↑↑↑↑↑new code↑↑↑↑↑↑'''
            # did the bomb hit a building?
            for building in self.buildings:
                if building.is_inside(self.bomb.position):
                    self.bomb.explode()
                    building.shrink()
```

Bug #2 - Windows size

> The first building doesn't show in the Canvas windows

Description

> The window size is smaller than the bounding of the building.

What should happen

> CANVAS_WIDTH should greater than the boundary of the building.

Fix code

> - Change `CANVAS_WIDTH = 1000` to `CANVAS_WIDTH= 1400`
> - Add 100 to the building generated postion, in order to make the buildings located at the center of the windows
>   - `self.x = building_num * SPACING + 100`

```
CANVAS_WIDTH = 1400
class Building():
    def __init__(self, canvas, building_num, height, width):
            self.x = building_num * SPACING + 100
```

Bug #3 -Landed problem

> The plane can't be landed

Description

> Firstly, We can track to function *check_plane* which checks the position of the plane, then we should notice it only lands the plane when the bottom of the plane equals to `CANVAS_HEIGHT` and the plane is at the left of the windows.

What should happen

> When the plane is at the left of the windows and exceeds `CANVAS_HEIGHT`, it should be landed.

Fix code

> - Add an if-statement to return False when the height of the building is 0.
> - Change `plane_body_bottom.getY() == CANVAS_HEIGHT` to `plane_body_bottom.getY() >= CANVAS_HEIGHT`

```
class Building():
    def is_inside(self, point):
        '''↓↓↓↓↓new code↓↓↓↓↓'''
        if self.height <= 0:
            return False
        '''↑↑↑↑↑↑new code↑↑↑↑↑↑'''
```

```
class Display(Frame):
    def check_plane(self):
        '''↓↓↓↓↓new code↓↓↓↓↓'''
        if plane_body_bottom.getY() >= CANVAS_HEIGHT and
plane_body_bottom.getX() < 20:
            self.plane_landed()
        '''↑↑↑↑↑↑new code↑↑↑↑↑↑'''
```

Bug #4 - Hitting problem

> Difficult to hit the first building

Description

> Difficult to hit the first building.

What should happen

> The plane should be generated before the first builiding.

Fix code

> - ```
>   self.plane = Plane(self.canvas, CANVAS_WIDTH + 100, 0)  # Plane initialised postion
>   ```

Bug #5 - Explode problem

> The bomb doesn't explode when hit the left side of the building

Description

> The bomb doesn't explode when hit the left side of the building. Basically, this because the `building_inside` function doesn't consider the volume of the bomb

What should happen

> The bomb should explode when hit the boundary of the building.

Fix code

> - Add a function to check whether the bomb is inside the building.

```
    class Building():
        def is_bomb_inside(self, point):
            print(point.X, self.x)
            if point.X < self.x - 5 or point.X > self.x + self.width or point.Y < CANVAS_HEIGHT - self.height:
                return False
            return True
```

Bug or Feature #6 - Restart problem

> Can't restart the game when running the program

Description

> `self.text` should be set an initial value when the game starts.

What should happen

> The player can restart the game anytime.

Fix code

```
class Display(Frame):
    def __init__(self, root):
    '''↓↓↓↓↓new code↓↓↓↓↓'''
        self.text = 0
    '''↑↑↑↑↑↑new code↑↑↑↑↑↑'''

    def restart(self):
        '''↓↓↓↓↓new code↓↓↓↓↓'''
        if self.text:
            self.canvas.delete(self.text)
        '''↑↑↑↑↑↑new code↑↑↑↑↑↑'''
```

Improvement

Pause function

Description

> Add a pause function for user to pause the game by using *messagebox*

```
import tkinter.messagebox


class Game():
    def __init__(self):
        self.pause = False
'''↓↓↓↓↓new code↓↓↓↓↓'''
    def turn_pause(self):

        if self.pause == True:
            tkinter.messagebox.showinfo("Pause", "press enter to resume the game")
            self.pause = False
'''↑↑↑↑↑↑new code↑↑↑↑↑↑'''


class Game():
    def key(self, event):
        '''↓↓↓↓↓new code↓↓↓↓↓'''
        elif event.char == 'p':
        self.pause = True
        self.turn_pause()

    '''↑↑↑↑↑↑new code↑↑↑↑↑↑'''
```

FPS drop problem

> Just increase the rate of checking framerates