# Bug report for ENGF0002 CA 3.0 python

**By Yadong Liu (Adam)**

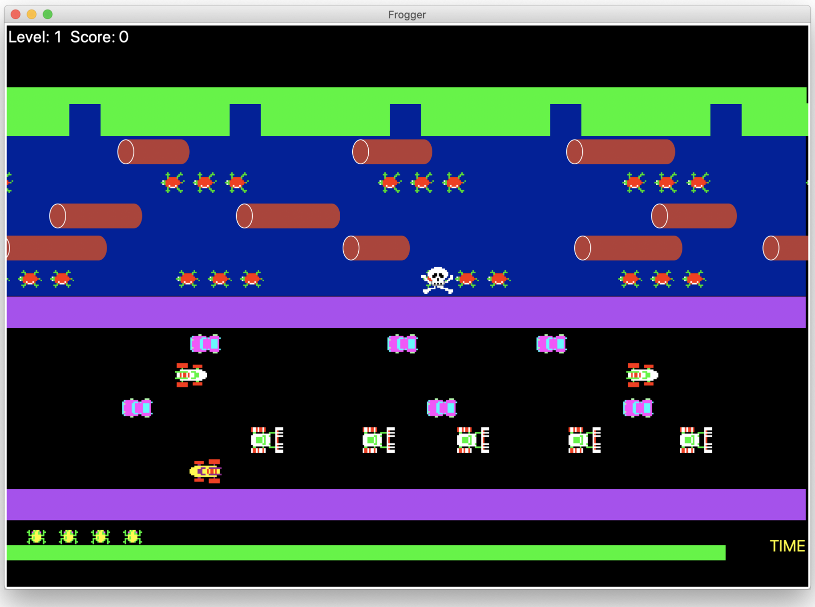
## 0. Basic game introduction

The game is programmed using tkinter. The objective of the game is to cross the road, avoiding the cars, cross the river by jumping on the logs or turtles, and make it to one
of the five frog homes at the top of the screen.

## 1. Issues

### Bug 1: User can not cross the river (frog dies) even on top of a turtle

(screenshot with bug)



#### Reason:

In "fr_model.py" check_frog_crossing_river function. Inside loop for checking if frog is on any of log, the variable name on_log was typed wrongly as "on_long". Hence "on_log" will always be False and frog will die once it has x,y coordinates inside river.

#### Fix:

```
# Correct variable name on_log
for log in self.logs:
    if log.contains(self.frog):
        #Bug1: wrong variable name
    #on_long = log
    on_log = log
    break
```

## Bug 2: Once a frog is dead, the rest of frogs will also die one by one at a rate of one second

### Reason:

In "fr_model.py" after a frog dead. if there are remaining frogs, the program pause for 1sec and unpause after calling method "self.new_life( )" inside class model.
In method "new_life", the method only updated lives of frogs but not reset the position of frog. Hence the next frog will have the same position as previous frog ( Which will cause frog die).

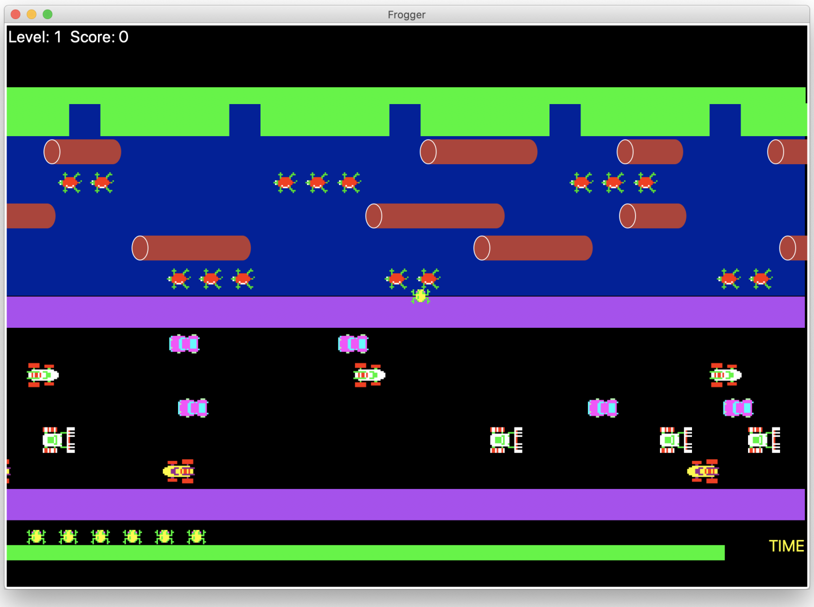
### Fix:

In method "new_life" in class model, add a function call to reset position for next frog

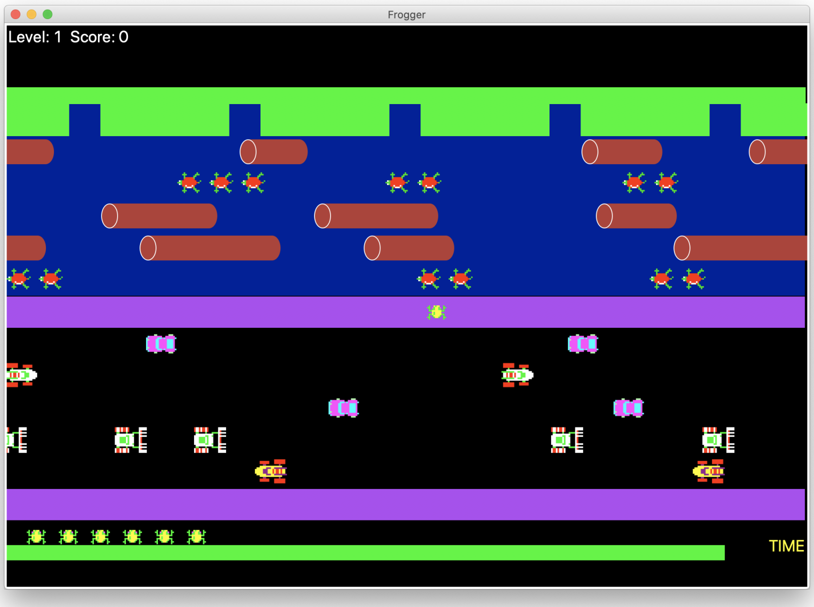
```
    def new_life(self):
        #Bug2: didn't reset postion of frog after a frog is dead
        self.frog.reset_position()
        self.controller.update_lives(self.lives)
```

## Bug 3: Frog can be moved into a strange position when quickly stroke up/down + left/right

(bug screenshot)



(Normal)



## Reason:

In "fr_model.py" frog.move_frog method. there isn't any check for current status for frog before starting a new movement. Hence combine movement was allowed in the game (which shouldn't) and that causes frog to have a coordinate on half of a grid.

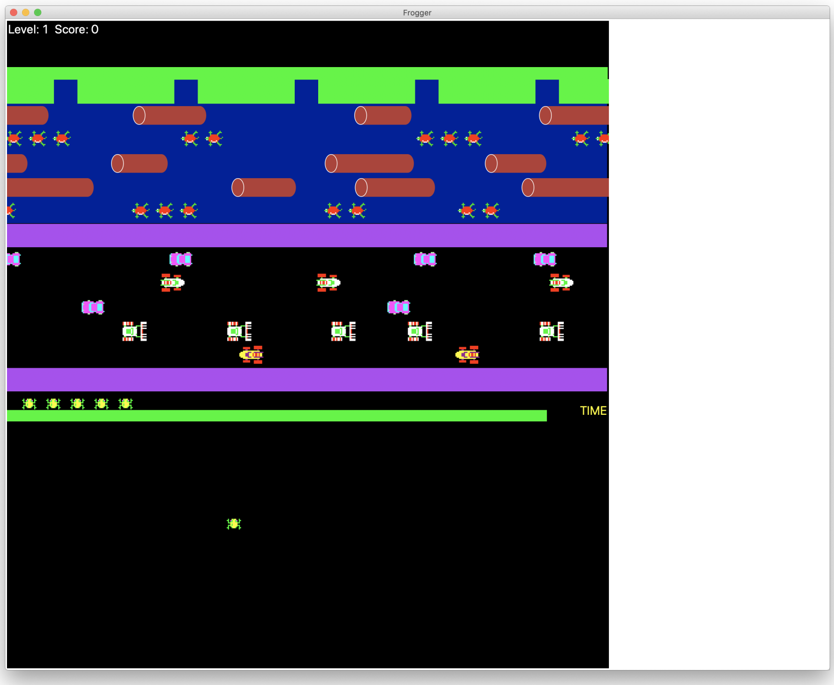
## Fix:

Add a condition check for frog status before starting a new movement.

```
    def move_frog(self, dir):
        if self.game_running and not self.paused:
            # Bug3: Movement should Only done one by one (should not allow
combine key)
            if self.frog.moving:
                return
            self.frog.move(dir)
```

## Bug 4: Frog can go outside canvas (in height)

(screenshot of bug)



### Reason:

In "fr_model.py" There was no restriction on moving frogs in y coordinates.

### Fix:

```
# In class frog, add a method to get initial x,y coordinate

    def get_start_position(self):
        return self.start_position

# In check_frog(self), add a condition to make sure frog does not go below it's
start position

    (startx, starty) = self.frog.get_start_position()
    if y < 0 or y > starty:
        self.died()
      return
```

## Bug 5: All things got frozen when restart from a "game over" condition

### Reason:

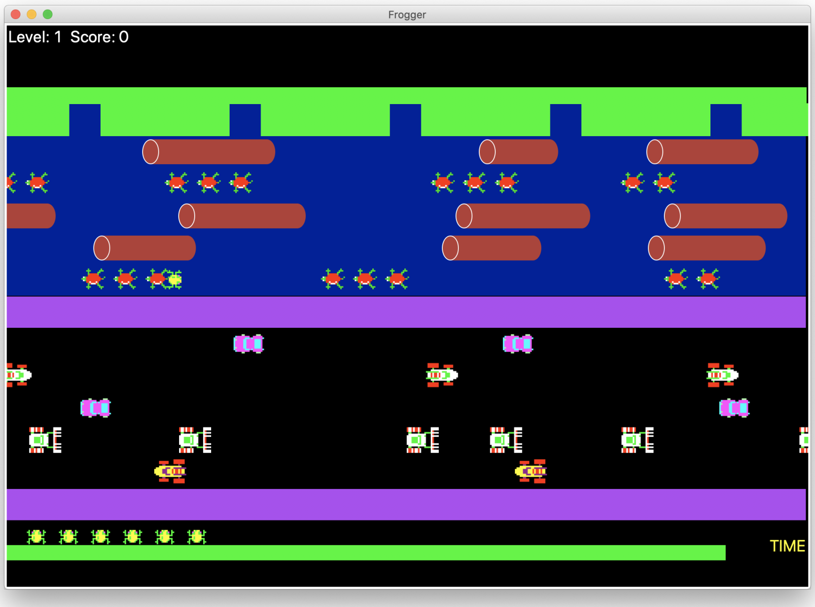
In "fr_model.py" when game is over, variable "game_running" is set to False. However, there is nothing for changing running status inside restart function. Hence, after game_running is set to False, even we press the key for restart, the game is still in a state of not running. Which causes everything to be "frozen" in user side.

### Fix:

```
    # Inside class Model()
def restart(self):
    self.level = 1
    self.score = 0
    self.reset_level()
    self.dont_update_speed = True
    #Bug 5: Makesure game is running after restarting
    self.game_running = True
```

## Bug 6: Frog can stand behind last turtle but not on the top one

(Bug screenshot)



### Reason:

In "fr_model.py" class Model, "contains(self, frog)". The x coordinate of turtle used to check is the centre of first turtle. Hence when frog is on left of centre of first turtle, frog is dead even we see it is on the turtle.

### Fix:

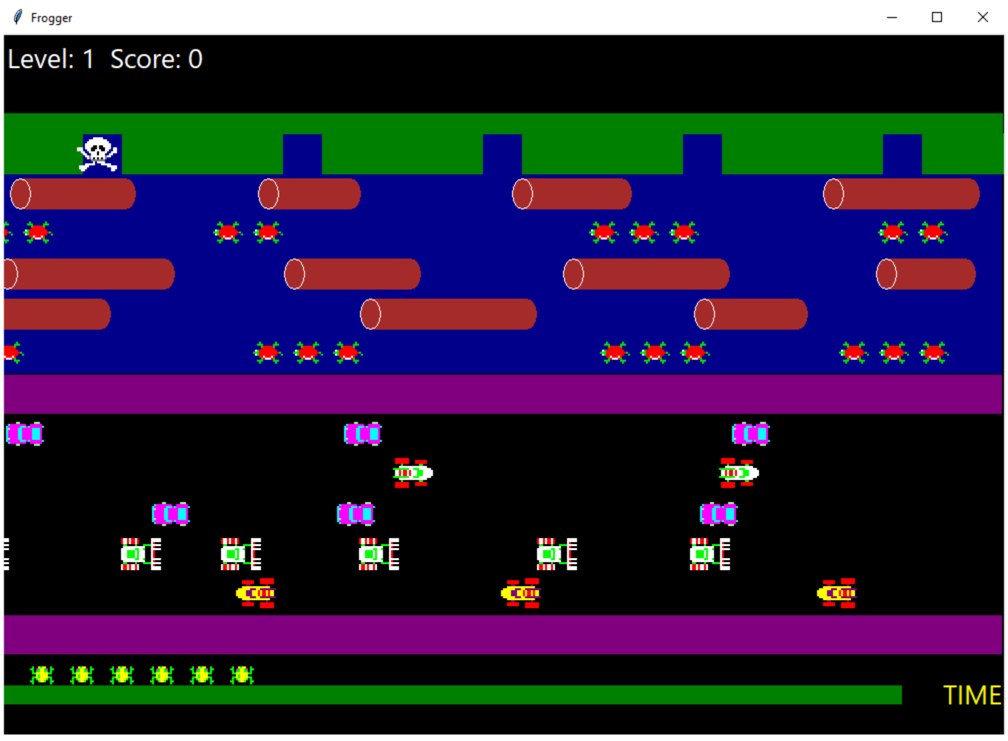
Since turtle and log are both river objects. The "contains" function works for logs. Hence it is better to give an offset to draw turtle.

```
    #In fr_view.py class TurtleView
    def draw(self):
        #We move turtle's centre to right at an offset of GRIDE_SIZE//2 so in
contains function, x coordinate will                      #become first turtle's
head
        #After (x,y) = self.turtle.get_position()
        x = x + GRID_SIZE//2

    def redraw(self, time_now):
        #After (x,y) = self.turtle.get_position()
        x = x + GRID_SIZE//2
```

## Bug 7: Frog dead even it has successfully entered the first house (Count from L to R)

(Bug screenshot)



### Reason:

In "fr_model.py" the function didn't append x coordinate of most left home. Hence homes_x starts from second house count from left.

### Fix:

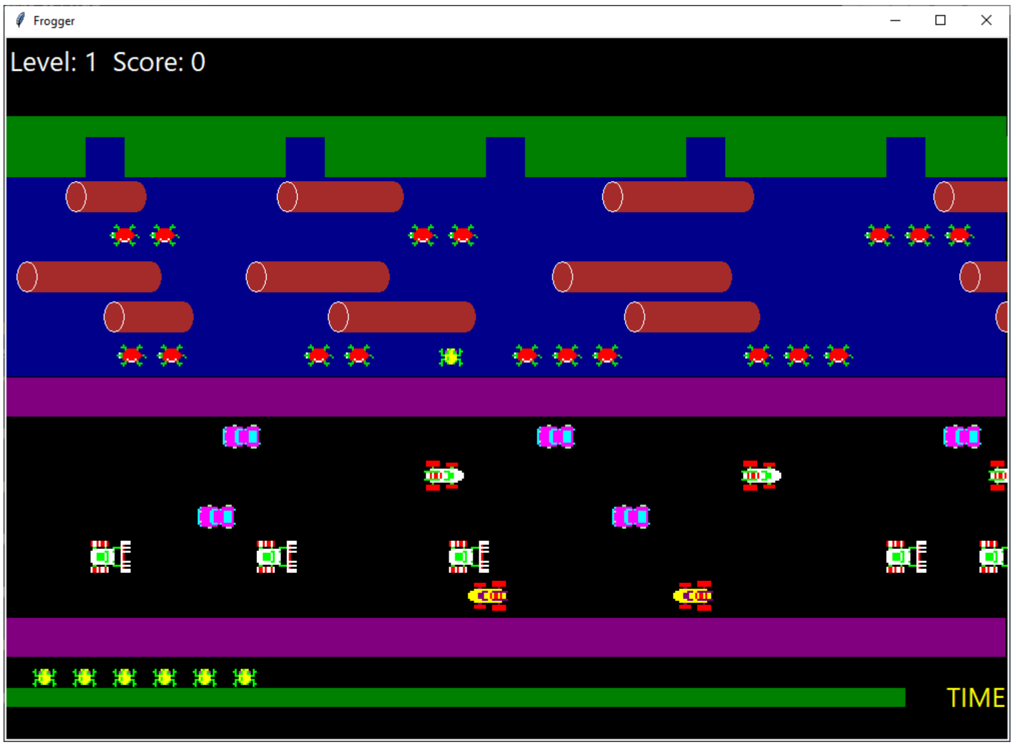
In "fr_model.py" class Model() method "create_homes(self)". Add a statement out side loop to append x coordinate of most left home into list.

```
    def create_homes(self):
        self.frogs_home = 0
        self.homes_x = []
        self.homes_occupied = []
        spacing = (CANVAS_WIDTH - GRID_SIZE*5)//5
        x = (spacing + GRID_SIZE)//2
        #Bug 7: Didn't append the x coordinate for first home (L to R)
        self.homes_x.append(x)
        for i in range(0,6):
            x = x + GRID_SIZE + spacing
            self.homes_x.append(x)
            self.homes_occupied.append(False)
```

## Bug 8: Frog shift when standing on top of turtle

eg. We stand on the first turtle and then when we try to stand on third turtle (same turtle series) frog will shift to first turtle.
Sometimes we have turtle not following turtle and not dead

(Bug screenshot)



### Reason:

- How to trigger this bug: eg. When we have stayed on a turtle, we quickly go back to road and then try to jump frog on same turtle series again.
  The reason of cause this bug is because when we go back to purple block between road and river, "frog.log" is not restore to none, hence while frog is still in range of the same turtle series. "check_frog_crossing_river" will execute "self.frog.move_with(on_log)" directly, which causes problem.

### Fix:

Add condition to restore frog.log to None when frog is on purple blocks

In "fr_model.py" class model "check_frog

```
    def check_frog(self):
        XXXXXXXXXXXXXXXXXXXXXXXXXXX
        XXXXXXXXXXXXXXXXXXXXXXXXXXX
        XXXXXXXXXXXXXXXXXXXXXXXXXXX
        if y >= GRID_SIZE * 10 and y <= GRID_SIZE * 14:
            # frog is on the road
            self.check_frog_crossing_road()

        #Bug 9: Reset log to none when frog goes back to line between road and
river
        elif y == GRID_SIZE * 9:
           self.frog.log = None

        elif y >= GRID_SIZE * 4 and y <= GRID_SIZE * 8:
            # frog is crossing the river
            self.check_frog_crossing_river()
        elif y == GRID_SIZE * 3:
            # frog is attempting to enter home
            self.check_frog_entering_home()
```

## Bug 9: Remain time "frozen"

### Reason:

Initial level time is 120 second. However, green block represent this time has parts outside canvas. Hence, time seems frozen in users' view.

### Fix:

Add condition to restore frog.log to None when frog is on purple blocks

In "fr_view.py" class TimeView( ) "update(self, time_now)"

```

    #Bug 9: simply rescale time bar
    self.bar = self.canvas.create_rectangle(CANVAS_WIDTH - 5*remaining - 100,
GRID_SIZE*16.25,CANVAS_WIDTH - 100, GRID_SIZE*16.75, fill="green")
```

## 2. Improvements

### 1. When remain time is 0, game over

Code:

```
    #in class model, add a method to return remain time
    def check_remaintime(self):
        remain = int(self.end_time - time.time())
        if remain <= 0:
            return False
        return True

    #in class model, "update(self)"
    def update(self):
        if not (self.check_remaintime()):
            self.game_over()
```

## END