Fruit Catcher

GameDoc

```
Some fruit will fly up the screen.
You must try to destroy the fruit by clicking on it.
Get the high score to win!
```

Let's begin!

1. Your teacher will have prepared the testing field for you to use. On your screen you should see a blank "game.py" window. Let's work inside!
2. Everything you need has been prepared for you! You do not need to program any special classes or set the screen up. We will do that together!

Modules

Let's start by importing all of the modules we will need.

```
import screen_setting
import fruit
import scoreboard
import random
```

The `screen_setting` module handles our display. The `fruit` module is our fruit basket. The `scoreboard` module handles our score display. The `random` module will let us get random numbers.

Initialize our Objects

Let's create our game window and our scoreboard.

```
game_win = screen_setting.Game_Screen()
scoreboard = scoreboard.Scoreboard(game_win.surface, 0, 0)
```

Next, let's create our fruit basket lists. We will create a list of fruits and also create an empty list.

```
fruits = ["apple", "avocado", "banana", "cherry", "grapes", "orange",
"strawberry"]
fruit_list = []
```

Later, we will need to be able to tell Python which fruit we want. For now, we just want to mark our spot, like we would do when we go to a crowded restaurant and reserve a table before we order.

```
this_fruit = None
```

So far so good!

We have two initializers to create. Let's make our first one!

```
def init():
    global fruit_list
    for i in range(0, len(fruits)):
        fruit_list.append(fruit.Fruit(game_win.surface,
fruit_type=fruits[i]))
    scoreboard.score = 0
```

Here, we use the `global` keyword to access the `fruit_list`. This is where we fill our basket.

Within the `for` loop, which is set between values 0 and the length of the `fruits` list, we `append` to the `fruit_list` all of the `fruits` by creating objects from `fruit.Fruit.` We will use all of the fruit in the `fruits` list and fill our basket.

We also set the score to 0.

Next, let's get some fruit.

```
def get_fruit():
    global this_fruit
    this_fruit = fruit_list[random.randrange(0, len(fruit_list))]
    this_fruit.posY = 500
    this_fruit.posX = random.randrange(0,
this_fruit.fruit_sprite.get_width())
    this_fruit.isVisible = True
    this_fruit.set_speed()
```

Here, we are going to use the `global` keyword to access `this_fruit`.

1. We set `this_fruit` to a random fruit in the `fruit_list` basket.
2. We set the y position to 500, just below the screen.
3. We set the x position to a random point between the left edge of the screen and the right edge, minus the width of the fruit's sprite image.
    1. We do this because we don't want part of the fruit on the screen or a length of time where the screen is blank. It doesn't make for a fun game that way.
4. We set the visibility of the fruit to `True`.
5. We set the speed of the fruit.

Next, let's update our scoreboard.

```
def update_score():
    if not this_fruit.isVisible:
        scoreboard.score += 1
        scoreboard.say(scoreboard.score)
```

This is triggered whenever fruit is clicked and we take the fruit.

Great Job! Almost finished!

Now, let's do the main game function called `run()`

```
def run():
    init()
    get_fruit()
    while True:
        game_win.update()
        this_fruit.update()
        this_fruit.draw()
        scoreboard.draw()
        update_score()
        if this_fruit.posY >= 500 or not this_fruit.isVisible:
            get_fruit()
            continue
```

When we run `run()`, this is what happens:

1. We run `init()`
2. Then, `get_fruit()` runs
3. Next, under `while True`, we have our game loop.
    1. We update the game window's display every frame.
    2. We update the fruit, every frame.
    3. We draw the fruit on the screen, which is updated every frame.
    4. We draw the score on the screen.
    5. We update the score, every frame, so that it gives us a new score.
    6. If the fruit falls below the edge of the screen, `or` the fruit is invisible, we get a new fruit.
    7. We call `continue` so that the loop repeats.

Great Work! Now, Let's Finish the Script!

The next piece of script is important. Because we have many modules, we need to tell Python that this script is *not* a module, but a main script. We do so like this:

```
if __name__ == '__main__':
    run()
```

Press **CTRL+S** to save.

Then, press **F5** to run!

How to Play!

This is how you programmed the game to work.

- Click the fruit in the center. Can you capture it?
- If you can capture the fruit, you get a point!
- When you're finished, press ESC to exit the game!

Challenge

Can you find a way to customize the game to your style? Don't be afraid to take a look at the modules' `.py` files and see their code for yourself. Perhaps you may want to make it cooler?

Some Ideas:

- Add a 'game over' condition for loss or win
- Add sound

If you are unsure about how to do any of those two things, come join us at IT KiDS where you will learn how to do that, and much, much more!!

If you had fun making this game, join us at IT KiDS!

You will learn how to make games like this, and much more! If Python programming interests you, but you don't know what's happened while you typed this out, that's perfectly okay! IT KiDS has an Introduction to Python course that's part of the Pygame curriculum! Please don't hesitate to ask about it!

We look forward to seeing you soon!!