Technical Basics: Written Presentation
HE Jun

<u>Short presentation of the main idea:</u>
The gernal idea of my application is to modify classic Conway's Game of Life with Pygame. Based on the rule of cellular automaton, Game of Life is a two-dimensional zero-player game which is controlled by the computer itself.

The concept of this game is demonstrating the life of cells in the grid: alive or dead. The initial number (pattern) of cells is added or deleted by player, and once it is started, cells' life will be determined by their neighbours (generally eight neighbours). [1]

1. Under population: if a live cell has less than 2 live neighbours, it will die.
2. Over population: if a live cell has more than 3 live neighbours, it will die.
3. Stabilization: if a live cell has two or three live neighbours, it will remain alive.
4. Reproduction: if a dead cell has three live neighbours, it will become alive.

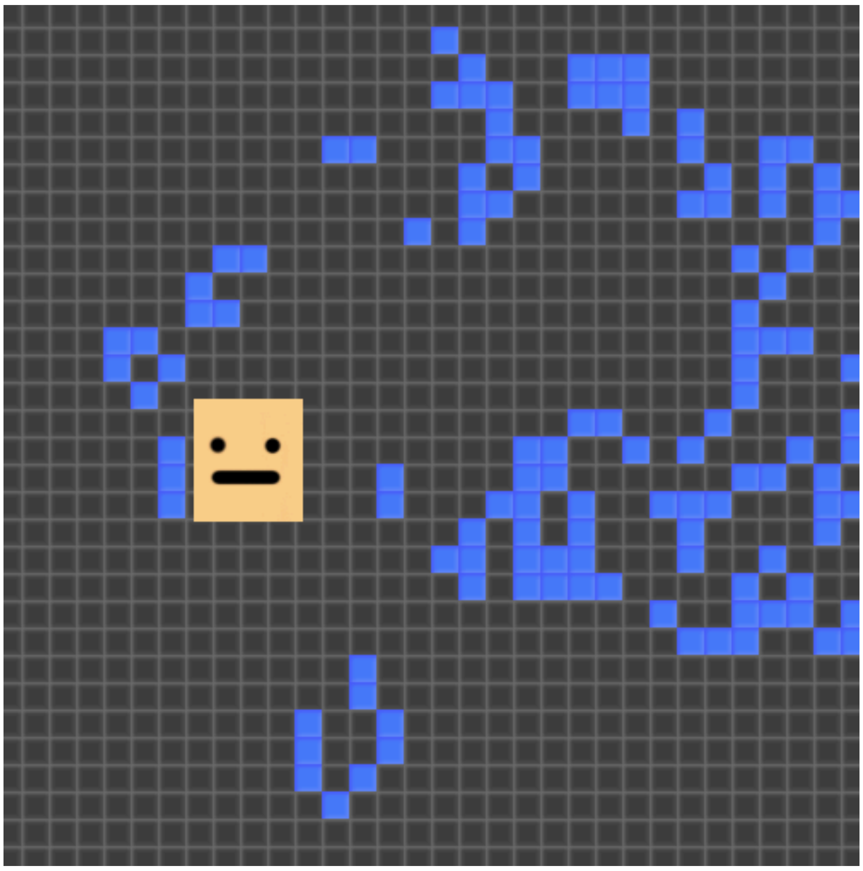
The original game can be played here: http://pmav.eu/stuff/javascript-game-of-life-v3.1.1/

<u>Description of functionalities:</u>
Even though the link of the game provided above was generated by Javascript, it is possible to make it with Python Pygame. When the game is running, the computer will generate many different types of amazing patterns or oder (some are being symmetrical) which is hard to make purly by human hand. The process of generating pattern is automatic based on the rule of cellular automaton. Based on the source code founded on internet, my application will try to add more user interaction and feature in it:

1. Adding a moving character (using Drawbot to create a gif of a running little man) to the game interface. After the game started, the character will move from the left to the right and move down at the same time. The player can lift the character up by pressing "up" keyboard.
2. The player have to protect the character from being blocked or touched by the living cells and ensure it not falls to the ground.
3. Beautify the game background (the grid) and cells by applying Drawbot. Adding music in it.
4. The things that control the character movement or cell placement do not need to be the mouse or keyboard. They could be other physical stuff such as real game controller or even sensor of human body movement in real life.
5. The game can be played by one to three players:
    1. Single player mode: clicking keyboard "a" to randomize the map and start the game.
    2. Two players mode: one can design the map before game started (thinking how to distribute the living cells for blocking the character), and the other will control the character.

[1] Wikipedia, Conway's Game of Life, https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

3. Three players mode: Two of them's roles are the same as two players mode, the third player will be responsible for mapping the cells after the game started to help the character pass sucessfully.



A screenshot of the Game of Life: the blue rectangles are living cells, the dark rectangles are the dead cells, and the yellow face is the character.

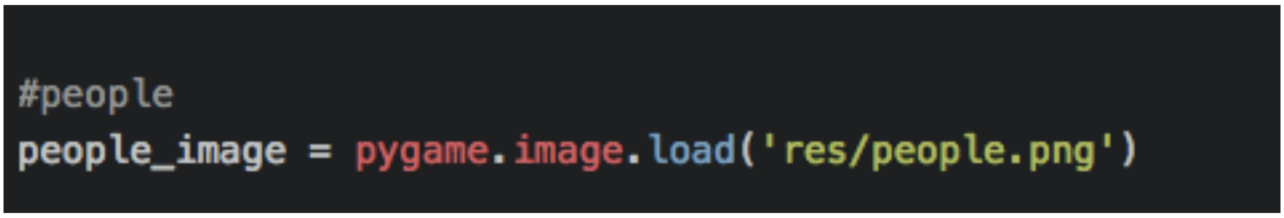
After being modified and improved, the Game of Life will become more interesting and playable. It is half human control and half machine algorithm, so the result somehow is out of player's expectation. The game enables players to not only have fun during leisure time but also experience the beauty of mathematics in a new way, sometimes the players even need to think about how to use mathematics to win the game.

<u>Description of the main structure:</u>

Five pages will be displayed: introduction page, one player mode page, two players mode page, three players mode page, and game over page. The introduction page is the home page which introduces the rules of game and allows players to select modes by pressing different button. In the three modes, which means the game is running or preparing to run, players can go back to home page and check the rules whenever they want by pressing the "Home" and "Help" button. When players win the game, or there is a bug, the page will jump to the game over page. The game over page contains the “Restart” and “Home” button for replaying the previous mode or returning to the introduction page.

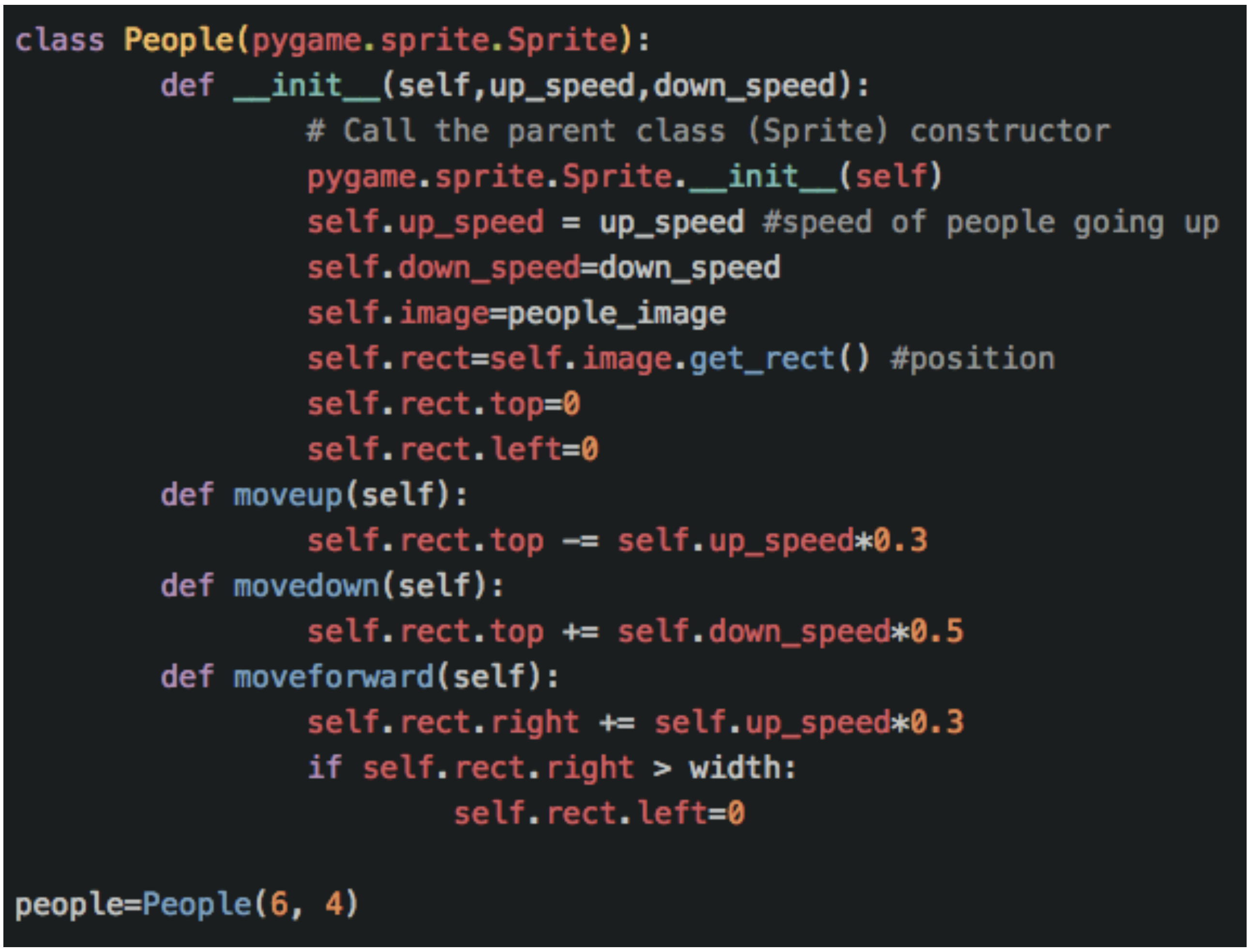
<u>Description of used modules/libraries:</u>
A source code of Conway’s Game of Life was found in Pygame official website.[2] The code is written for python 2 environment, so I modified the code to make it suitable for python 3. For example, I replaced range with xrange.

Based on this source code, I added the character by applying pygame.image.load.



```python
#people
people_image = pygame.image.load('res/people.png')
```

I also used “pygame.sprite.Sprite” to manage the movement of the character with the logic that “new position of the character = original position + or - speed”.

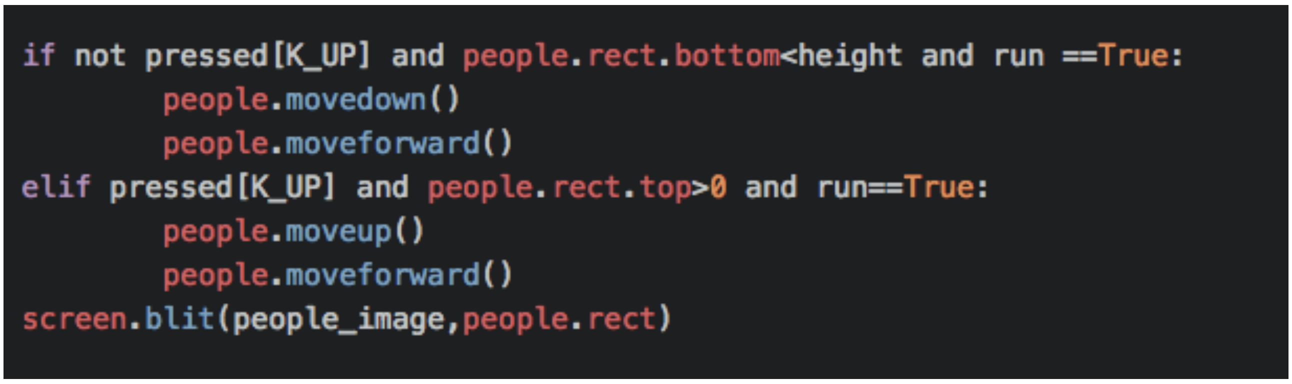


```python
class People(pygame.sprite.Sprite):
        def __init__(self,up_speed,down_speed):
                # Call the parent class (Sprite) constructor
                pygame.sprite.Sprite.__init__(self)
                self.up_speed = up_speed #speed of people going up
                self.down_speed=down_speed
                self.image=people_image
                self.rect=self.image.get_rect() #position
                self.rect.top=0
                self.rect.left=0
        def moveup(self):
                self.rect.top -= self.up_speed*0.3
        def movedown(self):
                self.rect.top += self.down_speed*0.5
        def moveforward(self):
                self.rect.right += self.up_speed*0.3
                if self.rect.right > width:
                        self.rect.left=0


people=People(6, 4)
```

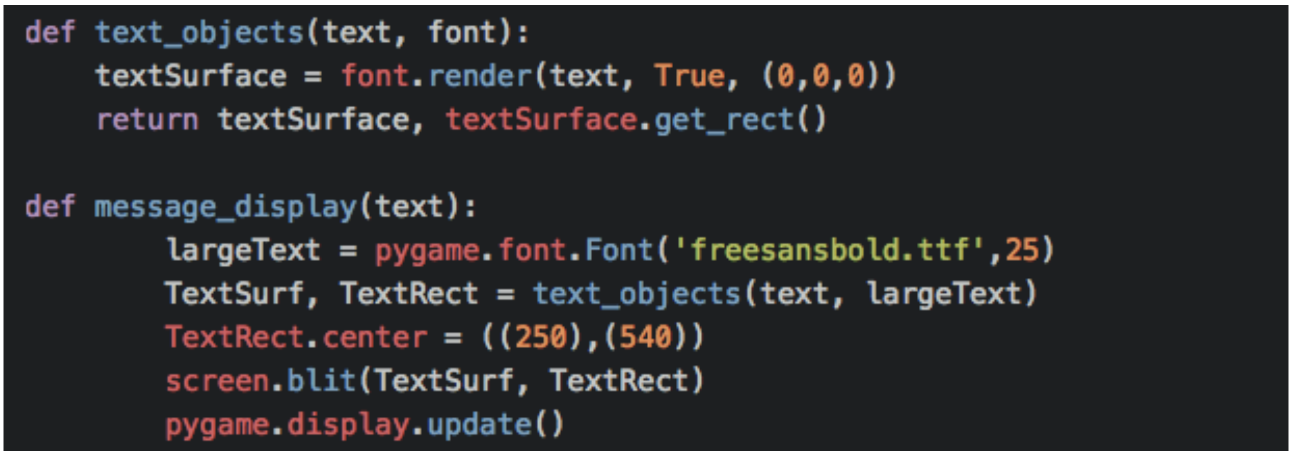
[2] Conway’s Game of Life by Nick Jarvis and Nick Wayne: https://www.pygame.org/project/2899/4734

For displaying the character and making the movement interaction with keyboard, I used "if-elif" statement to check whether the up keyboard is pressed or not and the character is touch the boundaries or not. I also found the similar code from the internet[3], but it is running when the file is opened. I want the character to move when the game is started instead of the file is opened, so I added one more condition to the statement, which is "run == True".



```
if not pressed[K_UP] and people.rect.bottom<height and run ==True:
        people.movedown()
        people.moveforward()
elif pressed[K_UP] and people.rect.top>0 and run==True:
        people.moveup()
        people.moveforward()
screen.blit(people_image,people.rect)
```

Next, I extended the height of the canvas (the game interface size remains the same) for adding introduction or button. The code of displaying text and button is token from a pygame tutorial.[4] In the future, the interactive button will be added here.



```
def text_objects(text, font):
    textSurface = font.render(text, True, (0,0,0))
    return textSurface, textSurface.get_rect()

def message_display(text):
        largeText = pygame.font.Font('freesansbold.ttf',25)
        TextSurf, TextRect = text_objects(text, largeText)
        TextRect.center = ((250),(540))
        screen.blit(TextSurf, TextRect)
        pygame.display.update()
```

Possible difficulties and solutions:

The next step of my project is to think about how to make the character interact with the cells. I am thinking about continue using the "pygame.sprite" by applying "spritecollide (sprite, group, dokill, collided = None)". However, to make that work I have to change the cells into sprite group. There might be another solution: instead of using pygame sprite, using "if" statement to detect the position of the character and living cells.

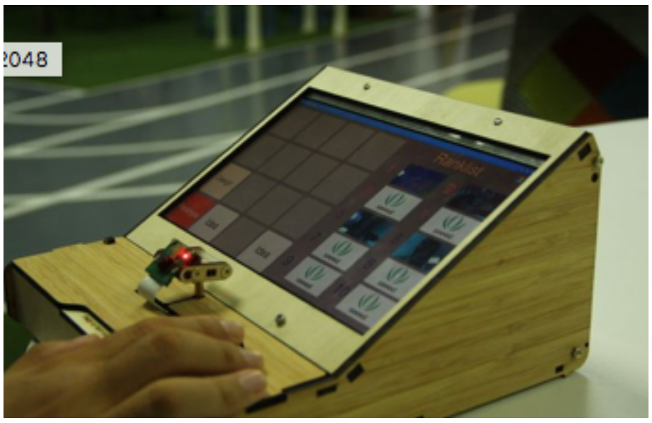
Another present difficulties is avoiding bugging. Because the living cells is keeping moving quickly, it is possible that the cells and character overlap. The solution might be

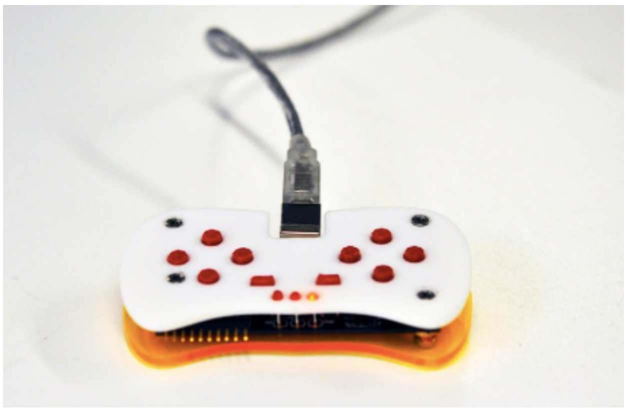
[3] Ball movement and interaction: https://zhuanlan.zhihu.com/p/40807732

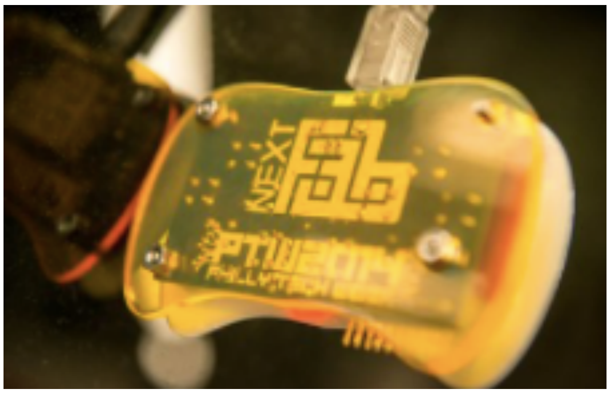
[4] text display: https://pythonprogramming.net/displaying-images-pygame/

slow down the speed or frame but I don't want to make the game too easy, so the balance between the error rate and complexity should be considered carefully. Maybe few trials should be made and played by different people, and I cam interview them to improve my game.

I was informed that in the next semester class we will learn how to visualise date and work with hardware. Even though I don't have any knowledge or experience about that before, I really want to extend my project with hardware (light, audio, or mechanical trigger). I want to enhance player's experience by replacing computer keyboard with classic hardware (game controller). I searched the inspiration in the internet. It might possible to make a game machine, for example: controlling Android phone with game boy[5], making a retro machine[6], or DIY a raspberry Game[7]. Playing game with biofeedback is also very cool but sounds too hard.[8] If the examples above are too hard to make, making a USB game[9] controller might be easier.







[5] Game boy Android gamepad: https://www.instructables.com/id/Game-Boy-Android-Gamepad/

[6] https://www.instructables.com/id/Retropie-Arcade-Game-Machine/

[7]DIY a raspberry Game: https://www.instructables.com/id/DIY-a-Raspberry-Game-2048/

[8] Playing game with Biofeedback: https://www.instructables.com/id/USB-Biofeedback-Game-Controller/

[9] Making USB game controller: https://www.instructables.com/id/Making-a-USB-Game-Controller/

References:

1. source code1 game of life: https://www.pygame.org/project/2899/4734
2. source code2 text display: https://pythonprogramming.net/displaying-images-pygame/
3. source code3 moving object (in Chinese): https://zhuanlan.zhihu.com/p/40807732
4. Game boy Android gamepad: https://www.instructables.com/id/Game-Boy-Android-Gamepad/
5. DIY a raspberry Game: https://www.instructables.com/id/DIY-a-Raspberry-Game-2048/
6. Playing game with Biofeedback: https://www.instructables.com/id/USB-Biofeedback-Game-Controller/
7. Making USB game controller: https://www.instructables.com/id/Making-a-USB-Game-Controller/