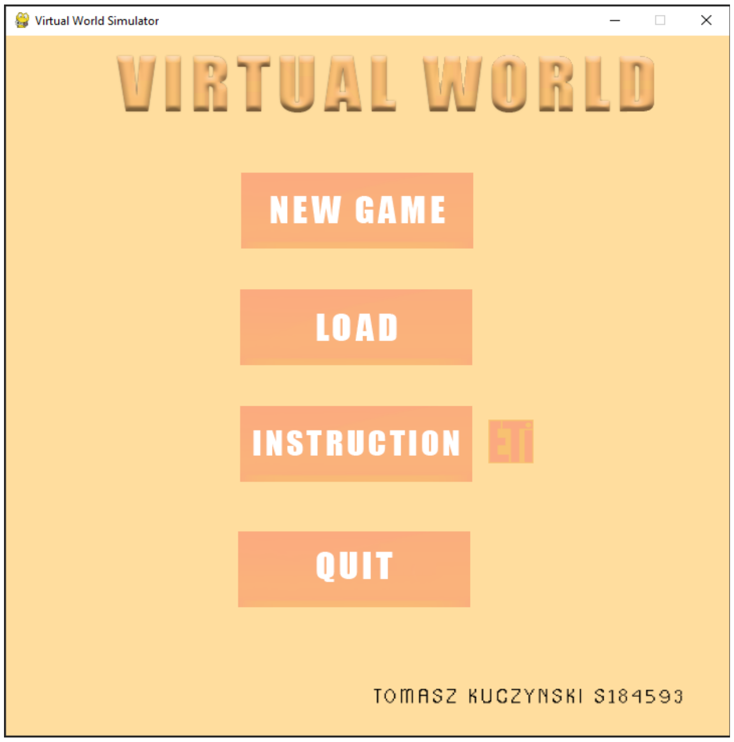
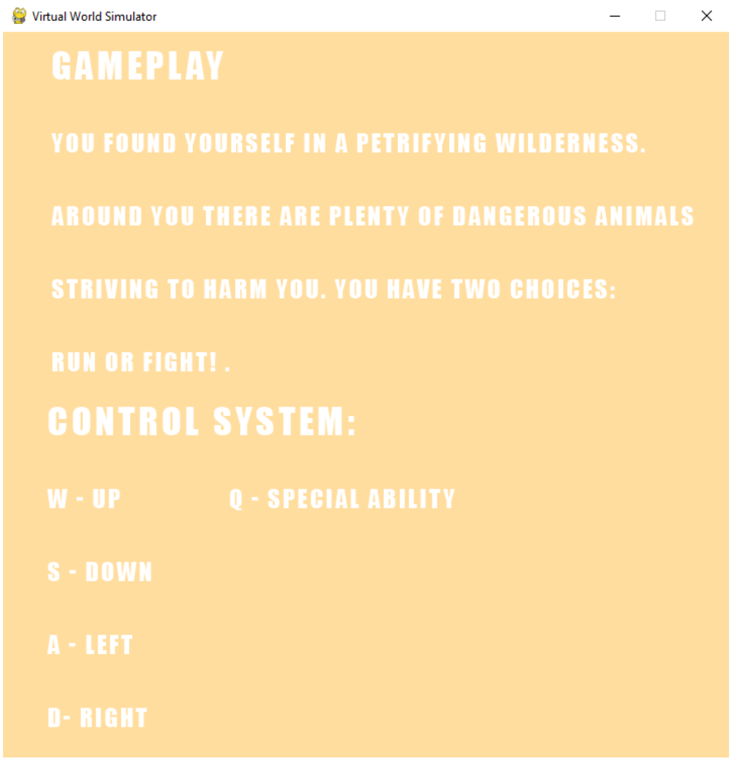
Author: Tomasz Kuczyński

# OBJECT PROGRAMMING VIRTUAL WORLD SIMULATOR PYTHON PROJECT

## 1.Implementation of the world and its visualization




Virtual World Simulator

GAMEPLAY

YOU FOUND YOURSELF IN A PETRIFYING WILDERNESS.

AROUND YOU THERE ARE PLENTY OF DANGEROUS ANIMALS

STRIVING TO HARM YOU. YOU HAVE TWO CHOICES:

RUN OR FIGHT! .

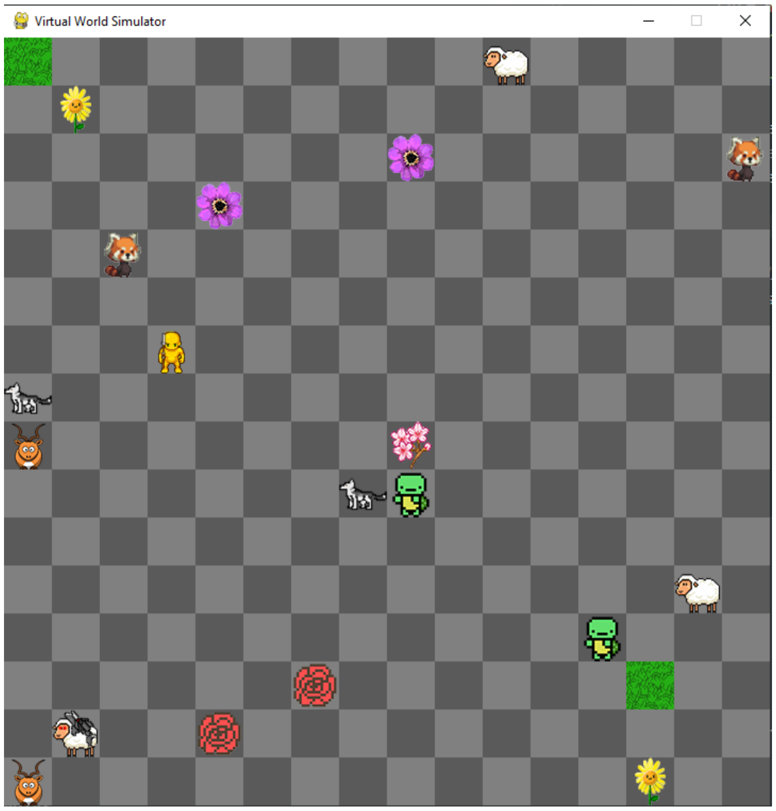
CONTROL SYSTEM:

W - UP Q - SPECIAL ABILITY

S - DOWN

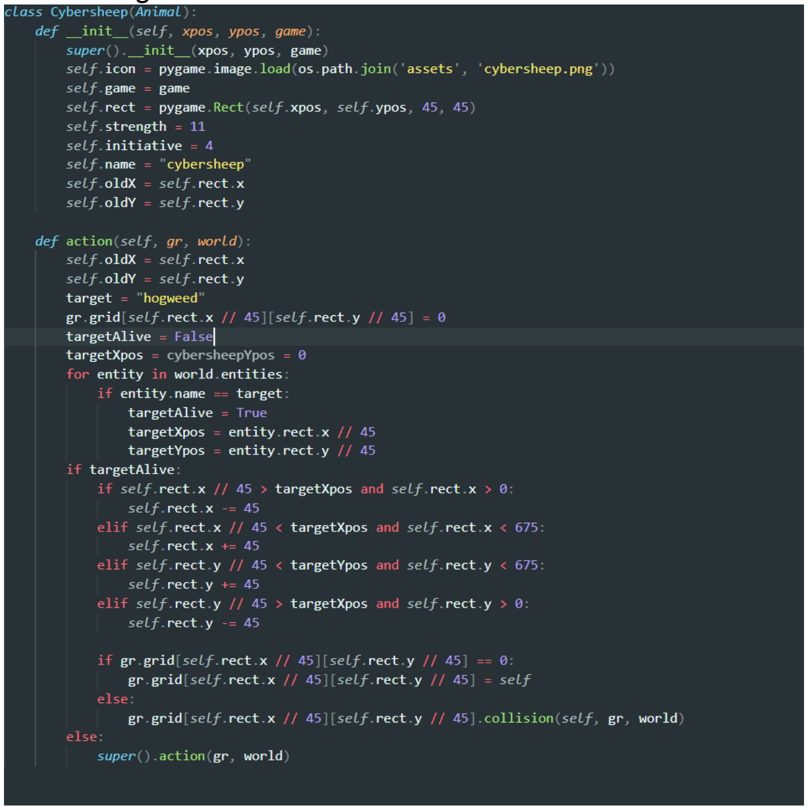
A - LEFT

D- RIGHT



**2. Implementation of all required animals (including cyber-sheep)**

- All of the organisms were implemented as visible on the screenshot. The game was initiated with 2 instances of every kind of organism.
- All of the animals classes are in a file Animal.py
- Example of the cybersheep Class: It is possible to select a target for the cybersheep and in this case it is set to hogweed as it was required. If the target is alive then whenever sheep moves in the action function it moves towards that target. If the target is dead then cybersheep behaves like a regular sheep by calling its parent class action function.



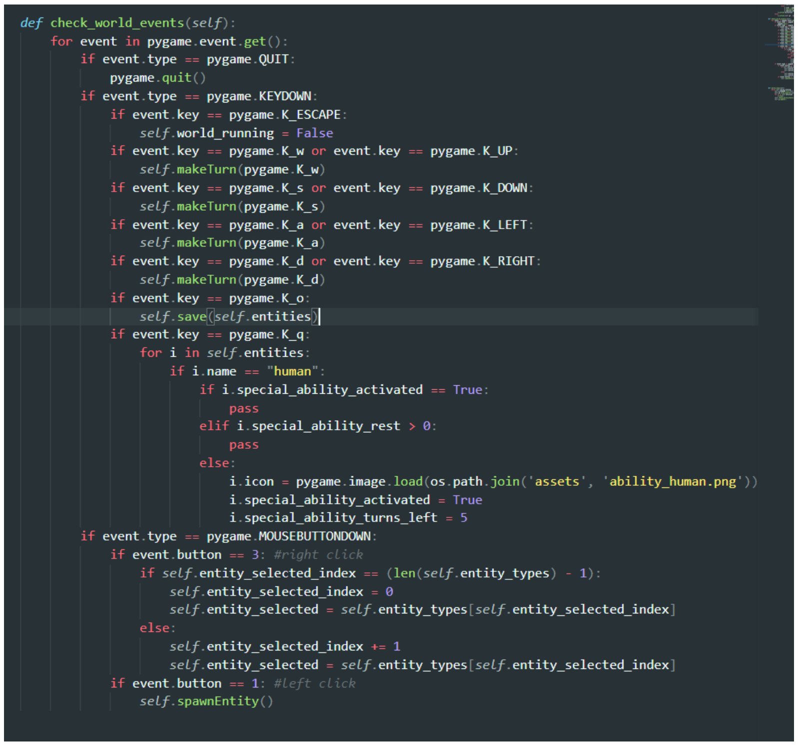
```python
class Cybersheep(Animal):
    def __init__(self, xpos, ypos, game):
        super().__init__(xpos, ypos, game)
        self.icon = pygame.image.load(os.path.join('assets', 'cybersheep.png'))
        self.game = game
        self.rect = pygame.Rect(self.xpos, self.ypos, 45, 45)
        self.strength = 11
        self.initiative = 4
        self.name = "cybersheep"
        self.oldX = self.rect.x
        self.oldY = self.rect.y

    def action(self, gr, world):
        self.oldX = self.rect.x
        self.oldY = self.rect.y
        target = "hogweed"
        gr.grid[self.rect.x // 45][self.rect.y // 45] = 0
        targetAlive = False
        targetXpos = cybersheepYpos = 0
        for entity in world.entities:
            if entity.name == target:
                targetAlive = True
                targetXpos = entity.rect.x // 45
                targetYpos = entity.rect.y // 45
        if targetAlive:
            if self.rect.x // 45 > targetXpos and self.rect.x > 0:
                self.rect.x -= 45
            elif self.rect.x // 45 < targetXpos and self.rect.x < 675:
                self.rect.x += 45
            elif self.rect.y // 45 < targetYpos and self.rect.y < 675:
                self.rect.y += 45
            elif self.rect.y // 45 > targetXpos and self.rect.y > 0:
                self.rect.y -= 45

            if gr.grid[self.rect.x // 45][self.rect.y // 45] == 0:
                gr.grid[self.rect.x // 45][self.rect.y // 45] = self
            else:
                gr.grid[self.rect.x // 45][self.rect.y // 45].collision(self, gr, world)
        else:
            super().action(gr, world)
```

## 3. Implementation of all plants

Implementation of all the plants is visible on the gameplay screenshot.
All of the plants classes are in a file Plant.py

## 4. Implementation of a Human which can be controlled by arrow keys.

Human is the yellow guy visible on the gameplay screenshot.

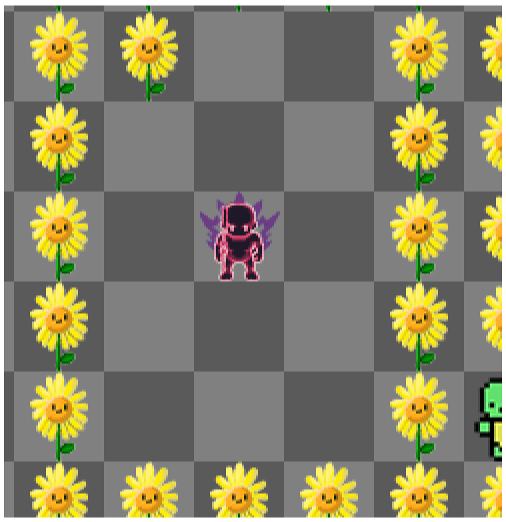


```python
def check_world_events(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                self.world_running = False
            if event.key == pygame.K_w or event.key == pygame.K_UP:
                self.makeTurn(pygame.K_w)
            if event.key == pygame.K_s or event.key == pygame.K_DOWN:
                self.makeTurn(pygame.K_s)
            if event.key == pygame.K_a or event.key == pygame.K_LEFT:
                self.makeTurn(pygame.K_a)
            if event.key == pygame.K_d or event.key == pygame.K_RIGHT:
                self.makeTurn(pygame.K_d)
            if event.key == pygame.K_o:
                self.save(self.entities)
            if event.key == pygame.K_q:
                for i in self.entities:
                    if i.name == "human":
                        if i.special_ability_activated == True:
                            pass
                        elif i.special_ability_rest > 0:
                            pass
                        else:
                            i.icon = pygame.image.load(os.path.join('assets', 'ability_human.png'))
                            i.special_ability_activated = True
                            i.special_ability_turns_left = 5
        if event.type == pygame.MOUSEBUTTONDOWN:
            if event.button == 3: #right click
                if self.entity_selected_index == (len(self.entity_types) - 1):
                    self.entity_selected_index = 0
                    self.entity_selected = self.entity_types[self.entity_selected_index]
                else:
                    self.entity_selected_index += 1
                    self.entity_selected = self.entity_types[self.entity_selected_index]
            if event.button == 1: #left click
                self.spawnEntity()
```

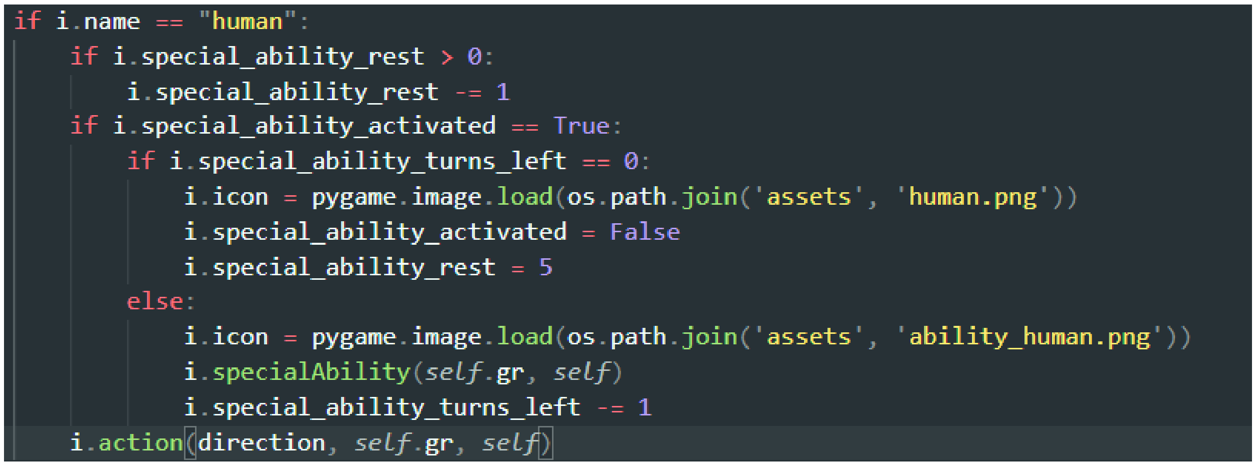
The way human's movement works is whenever W,S,A,D or one of the arrow keys are clicked, the event is directed to the makeTurn function which plays out the turn and when it's humans time time to go, the direction where he is supposed to go is passed to his action function.

## 5. Implementation of Human's special ability

It is possible for the human to use his special ability, which in this case is purification, by pressing the Q button on the keyboard. The human changes his color to purple and for the next 5 turns destroys everything around him. Then he needs to take a rest for another 5 turns before he uses it again.
Example screenshot of human using his special ability:



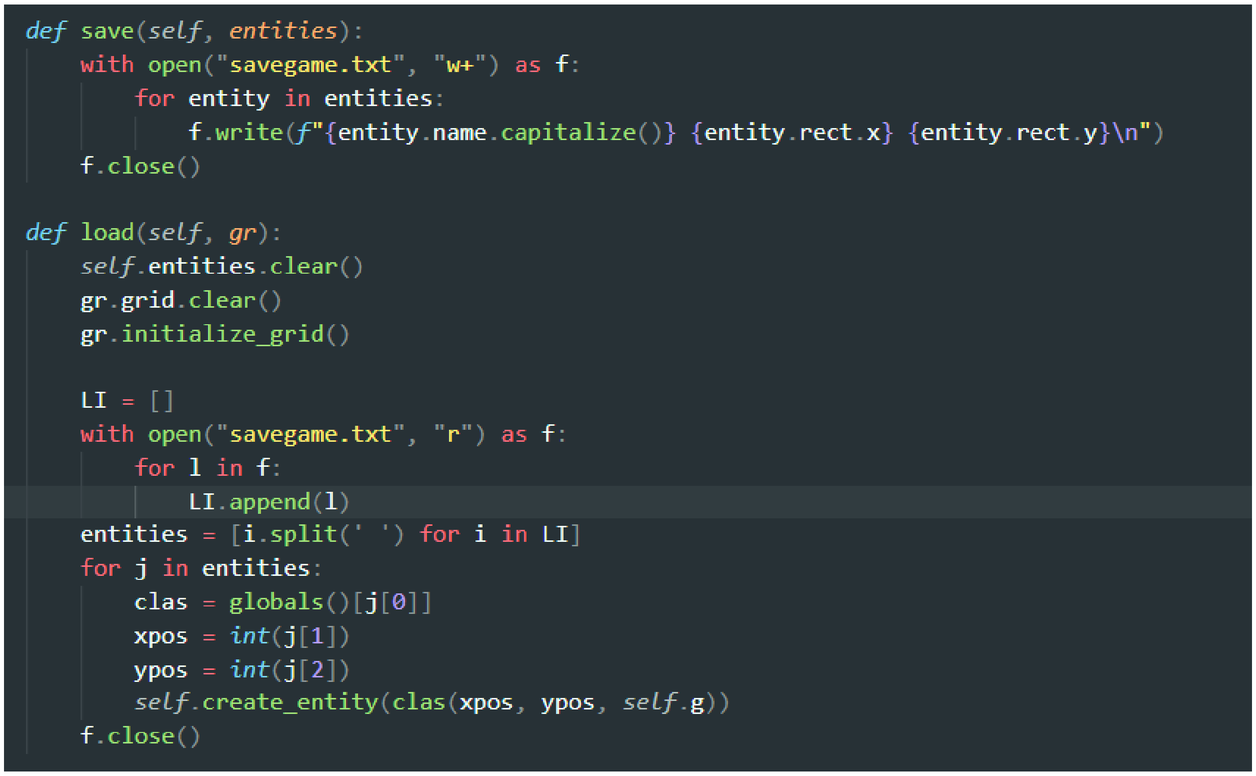
Special ability can be used anytime and when it comes to the humans turn then the count is being kept track of.
Code example:



```
if i.name == "human":
    if i.special_ability_rest > 0:
        i.special_ability_rest -= 1
    if i.special_ability_activated == True:
        if i.special_ability_turns_left == 0:
            i.icon = pygame.image.load(os.path.join('assets', 'human.png'))
            i.special_ability_activated = False
            i.special_ability_rest = 5
        else:
            i.icon = pygame.image.load(os.path.join('assets', 'ability_human.png'))
            i.specialAbility(self.gr, self)
            i.special_ability_turns_left -= 1
    i.action(direction, self.gr, self)
```

**6. Implementation of saving and loading state of the world to file and from file.**

It is possible to save the game state by clicking the ‘o’ button on the keyboard.
In order to load the game, the button in the main menu has to be used.
Code for loading and saving:

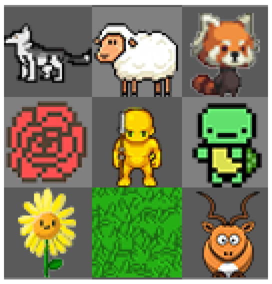


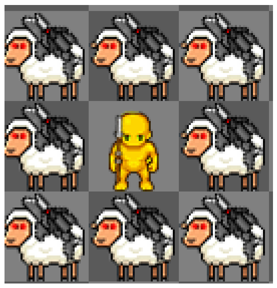
```python
def save(self, entities):
    with open("savegame.txt", "w+") as f:
        for entity in entities:
            f.write(f"{entity.name.capitalize()} {entity.rect.x} {entity.rect.y}\n")
    f.close()

def load(self, gr):
    self.entities.clear()
    gr.grid.clear()
    gr.initialize_grid()

    LI = []
    with open("savegame.txt", "r") as f:
        for l in f:
            LI.append(l)
    entities = [i.split(' ') for i in LI]
    for j in entities:
        clas = globals()[j[0]]
        xpos = int(j[1])
        ypos = int(j[2])
        self.create_entity(clas(xpos, ypos, self.g))
    f.close()
```

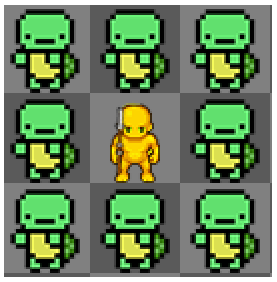
The game state is being saved in savegame.txt file. Loading spawns back up the saved entities in the positions that they should be occupying.

**7.Implementation of adding a new organism to the word by clicking on the free map cell. It should be possible to add organism of any kind**

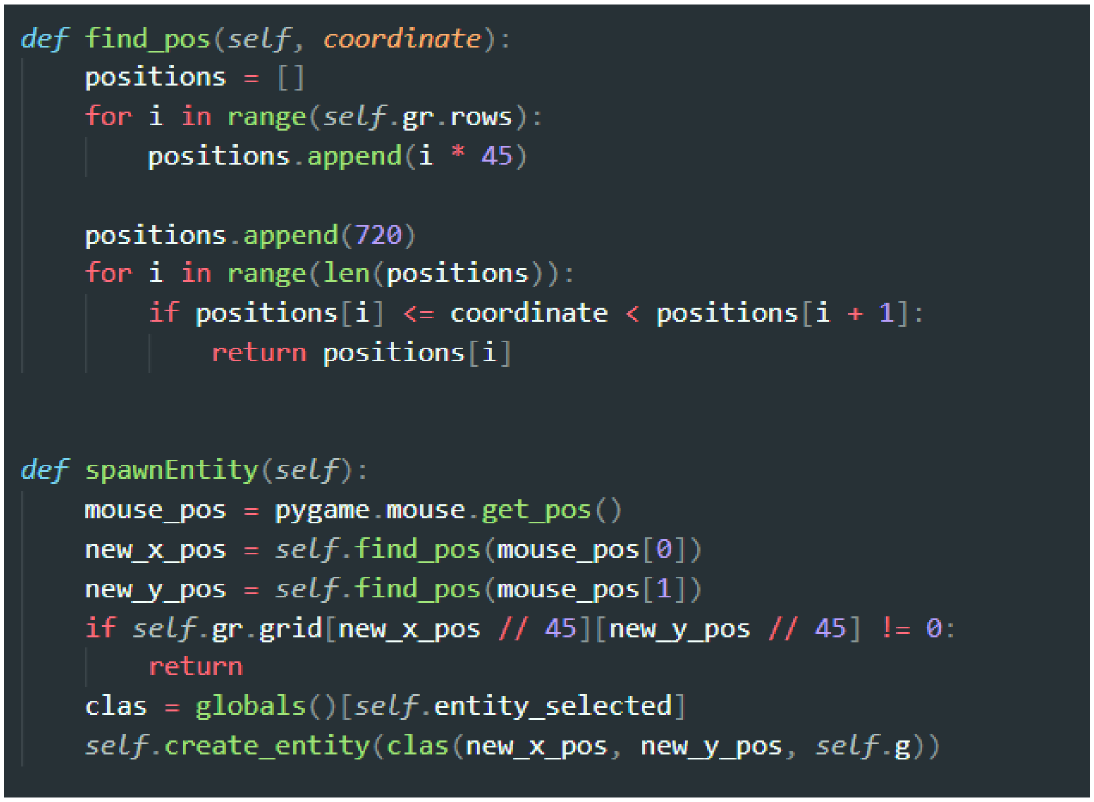
It is possible to add a new organism to the world by clicking on a free map cell with a left mouse button. All of the animals can be selected with a use of the right mouse button.
Examples of spawning 8 organisms around human by clicking:







Code example of the functions responsible for spawning:

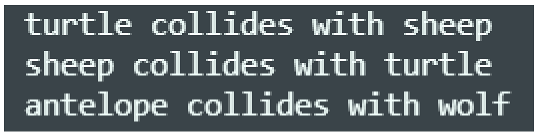


```python
def find_pos(self, coordinate):
    positions = []
    for i in range(self.gr.rows):
        positions.append(i * 45)

    positions.append(720)
    for i in range(len(positions)):
        if positions[i] <= coordinate < positions[i + 1]:
            return positions[i]


def spawnEntity(self):
    mouse_pos = pygame.mouse.get_pos()
    new_x_pos = self.find_pos(mouse_pos[0])
    new_y_pos = self.find_pos(mouse_pos[1])
    if self.gr.grid[new_x_pos // 45][new_y_pos // 45] != 0:
        return
    clas = globals()[self.entity_selected]
    self.create_entity(clas(new_x_pos, new_y_pos, self.g))
```

find_pos function creates an integer list with all of the rectangle centers that the organism can be spawned in. 720 is a “ghost” cell and is appended separately to make sure that the program doesn’t crash when the positions[i] <= coordinate < positions[i + 1] in the following lines is being run. It takes coordinate as a parameter which is a mouse position from a tuple gained from pygame.mouse.get_pos() function. spawnEntity() function uses find_pos function in order to find where to position the sprite so that he is in the center of the cell. The if statement in the spawnEntity() function works as a restriction that organisms can not be spawned in a cell that is not free. At the end the class is being figured out by converting a currently selected name of an organism of a string type to an appropriate class type by using a globals() function. And then it is just spawned in the found coordinates as regular.

**8.Additional information:**

All of the animal collisions are printed to the console:



```
turtle collides with sheep
sheep collides with turtle
antelope collides with wolf
```