Kinga Chodorowska 184549

Object Oriented Programming

Project 2. Python

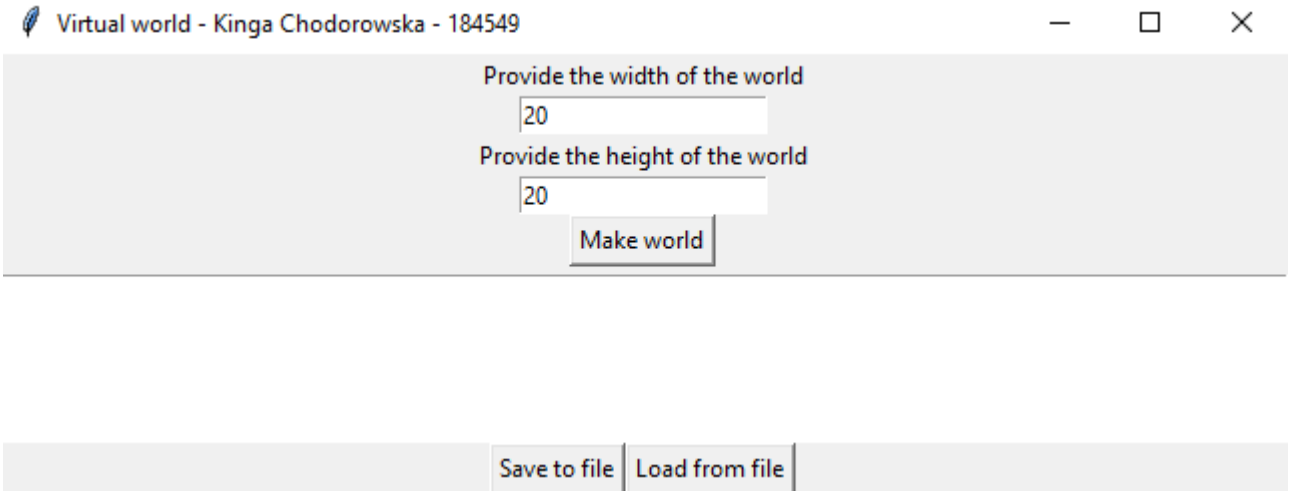
I completed the task for a total score of 4.

My project include:

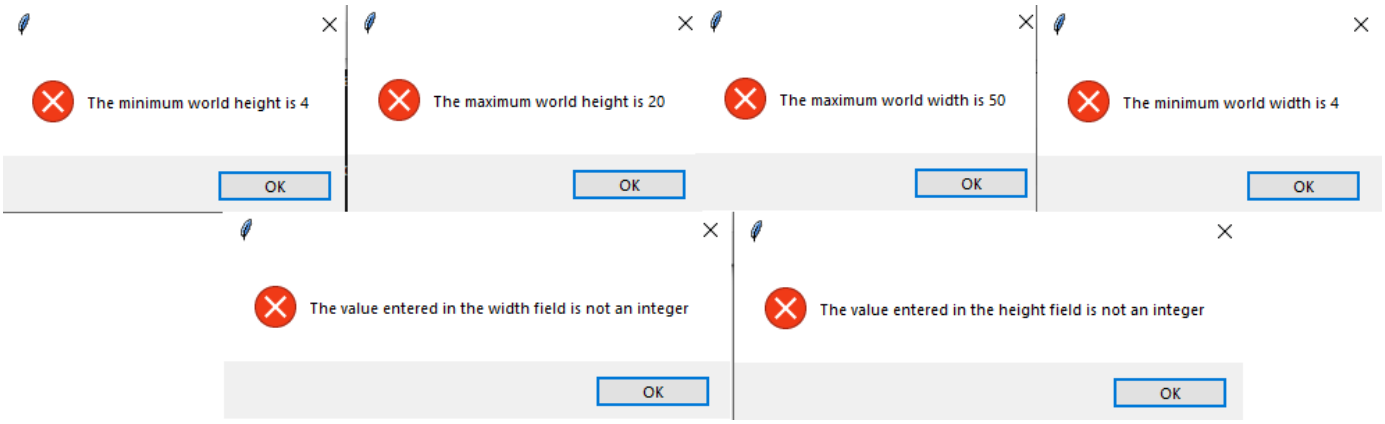
1. **implementation of the world and its visualization,**
2. **implementation of all required animals with breeding,**
3. **implementation of all plants with sawing,**
4. **implementation of a Human which can be controlled by arrow keys,**
5. **implementation of Human's special ability,**
6. **implementation of saving and loading state of the world to file and from file,**
7. **implementation of adding a new organism to the word by clicking on a free map cell.**

1. Implementation of the world and its visualization

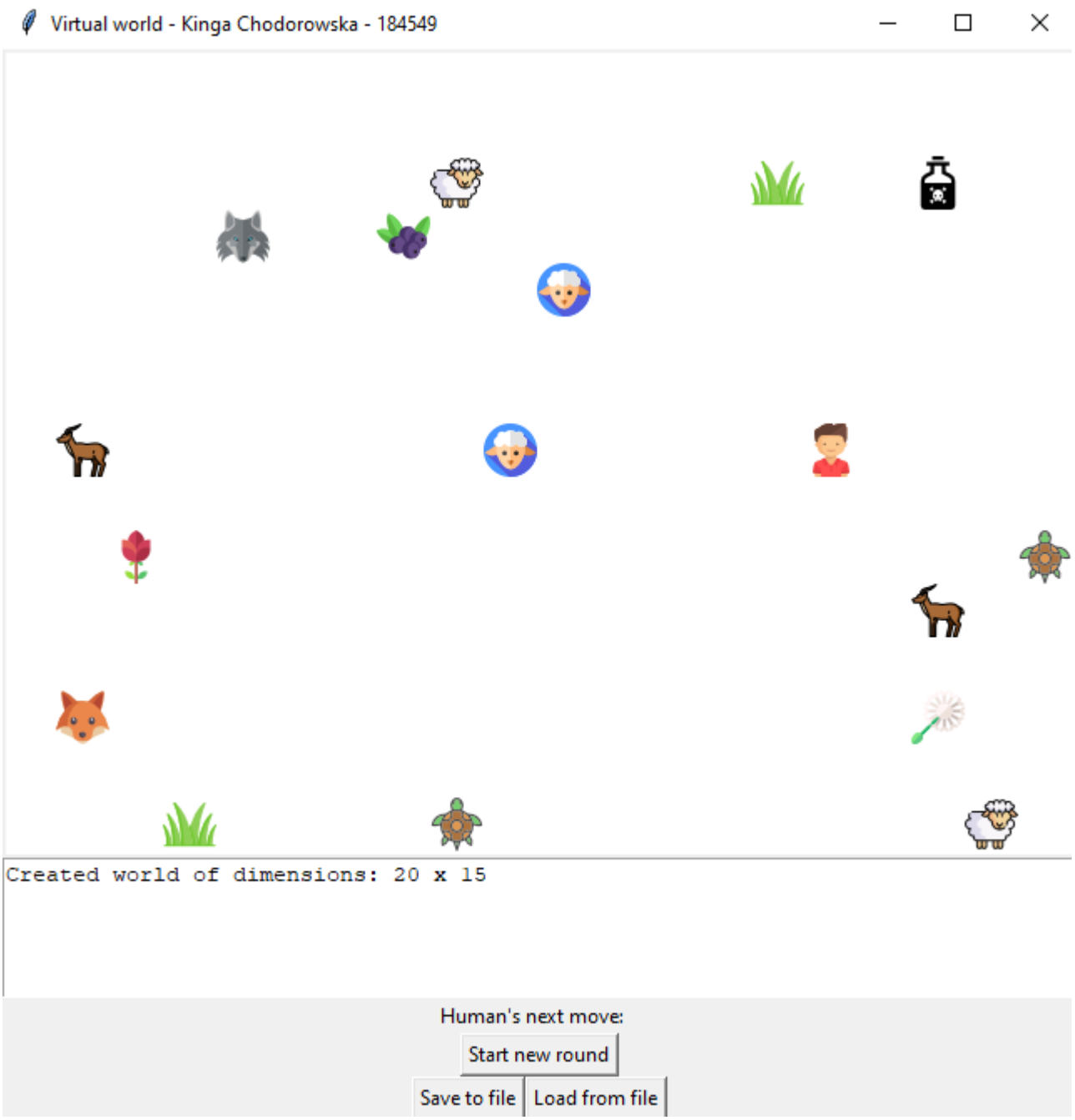
After running the program, the player can start a new game by entering the board size or load a game from a file. It can also save game to a file, but the file will be empty. The bar at the top shows the name of the game, the author's name and index.



The minimum width and height is 4 and maximum is 50 and 20. Organisms are created in such a way that at least one of each kind always appears on the board. In case the player enters an incorrect board dimension, one of the messages will appear on the screen.



When the user enters the correct dimensions or enters the correct file the board will appear in the window with a space for information about the round and information about the direction of human movement. This view will be visible until the game is over but each round the game will be updated.



The constructor of the World class is responsible for the appearance of the window. Each round is handled by the NextRound world method.

```
def __init__(self):                                                                    ⚠1
    self.root = Tk()
    self.root.wm_title("Virtual world - Kinga Chodorowska - 184549")
    self.nextHumanMoveText = StringVar()
    self.logField = Text(self.root, height=5)
    self.board = Canvas(self.root, bg="white")
    self.widthEntry = Entry(self.root)
    self.widthEntry.insert(END, '20')
    self.heightEntry = Entry(self.root)
    self.heightEntry.insert(END, '20')
    self.board.config(width=0, height=0)
    self.widthLabel = Label(self.root, text="Provide the width of the world")
    self.heightLabel = Label(self.root, text="Provide the height of the world")
    self.createWorldButton = Button(text='Make world', command=lambda: self.SetWorldSize(self.heightEntry.get(),
                                                                                          self.widthEntry.get(),
                                                                                          True))
    self.frame = Frame(self.root)
    self.buttonsFrame = Frame(self.root)
    self.nextRoundButton = Button(text='Start new round', command=self.NextRound)
    self.saveButton = Button(self.buttonsFrame, text='Save to file', command=self.Save)
    self.loadButton = Button(self.buttonsFrame, text='Load from file', command=self.Load)
    self.specialAbility = Label(self.root, text="Magical potion is enabled")
    Label(self.frame, text="Human's next move: ").pack(side=LEFT)
    Label(self.frame, textvariable=self.nextHumanMoveText).pack(side=RIGHT)
    self.widthLabel.pack()
    self.widthEntry.pack()
    self.heightLabel.pack()
    self.heightEntry.pack()
    self.createWorldButton.pack()
    self.board.pack()
    self.logField.pack()
    self.saveButton.pack(side=LEFT)
    self.loadButton.pack(side=LEFT)
    self.buttonsFrame.pack()
    self.root.bind_all('<Key>', self.KeyPressed)
    self.board.bind("<Button-1>", self.Callback)
    self.addedManuallyX = -1
    self.addedManuallyY = -1
    self.addManualIndex = 0

    self.roundNumber = 0
    self.organisms = []
    self.worldMap = []
    self.width = 0
    self.height = 0

    mainloop()
```

```
def NextRound(self):
    if self.nextHumanMoveText.get() == "":
        tkinter.messagebox.showinfo(message="Determine the next move of the human")
        return

    for organism in self.organisms:
        organism.actionMade = False

    nextOrganism = self.organisms[0]
    while nextOrganism is not None:
        nextOrganism = None
        for organism in self.organisms:
            if not organism.actionMade:
                if nextOrganism is None:
                    nextOrganism = organism
                elif organism.initiative > nextOrganism.initiative:
                    nextOrganism = organism
                elif organism.initiative == nextOrganism.initiative:
                    if organism.age > nextOrganism.age:
                        nextOrganism = organism
        if nextOrganism is not None:
            nextOrganism.TakeAction()

    self.AddLogInfo("Passed round number " + str(self.roundNumber + 1))
    self.roundNumber += 1
    self.nextHumanMoveText.set("")
    self.DrawWorld()
```

The game ends when the user decides. Death of a player does not end the game.

2. Implementation of all required animals with breeding

Symbols of animals:

Wolf	Sheep	Cybersheep	Fox	Turtle	Antelope	Human
						

Animals are created as soon as the board dimensions are entered. Animals are kept in the list - organisms, which is an element of the World class. Each turn, animals can move one box in one of four directions (up, down, left, right). Exceptions are: fox (doesn't move to the field where a stronger opponent is), turtle (has 75% chance to stay in its place), antelope (moves two fields) and cybersheep (moves towards the closes hogweed and tries to eat it. If there are no Sosnowsky's hogweeds, it behaves like a normal sheep).

```
def TakeAction(self):
    possibleMoves = self.FindPlaceToMove()
    move = random.randint(0, len(possibleMoves) - 1)
    collisionResult = 0

    for organism in self.world.organisms:
        if organism.x == possibleMoves[move][0] and organism.y == possibleMoves[move][1]:
            collisionResult = organism.Collision(self)
            break

    if collisionResult == CollisionResults.AttackedReflects or collisionResult == CollisionResults.AnimalCreation:
        self.age += 1
        self.actionMade = True
    if collisionResult == CollisionResults.AttackedDies or collisionResult == CollisionResults.AttackedEscapes or \
            collisionResult == CollisionResults.PlantIsEaten or collisionResult == 0:
        self.age += 1
        self.x = possibleMoves[move][0]
        self.y = possibleMoves[move][1]
        self.actionMade = True
```

Action() for Animal

```
def TakeAction(self):
    stay = random.randint(0, 3)
    if stay == 0:
        Animal.TakeAction(self)
    self.age += 1
    self.actionMade = True
```

Action() for Turtle

```
def TakeAction(self):
    possibleMoves = self.FindPlaceToMove()
    move = random.randint(0, len(possibleMoves) - 1)
    collisionResult = 0

    for organism in self.world.organisms:
        if organism.x == possibleMoves[move][0] and organism.y == possibleMoves[move][1]:
            collisionResult = organism.Collision(self)
            break

    if collisionResult == CollisionResults.AttackedReflects or collisionResult == CollisionResults.AnimalCreation:
        self.age += 1
        self.actionMade = True
    if collisionResult == CollisionResults.AttackedDies or collisionResult == CollisionResults.AttackedEscapes or \
            collisionResult == CollisionResults.PlantIsEaten or collisionResult == 0:
        self.age += 1
        self.x = possibleMoves[move][0]
        self.y = possibleMoves[move][1]
        self.actionMade = True
```

Action() for Fox

```python
def TakeAction(self):
    possibleMoves = self.FindPlaceToMove()
    move = random.randint(0, len(possibleMoves) - 1)
    collisionResult = 0

    for organism in self.world.organisms:
        if organism.x == possibleMoves[move][0] and organism.y == possibleMoves[move][1]:
            collisionResult = organism.Collision(self)
            break

    if collisionResult == CollisionResults.AttackedReflects or collisionResult == CollisionResults.AnimalCreation:
        self.actionMade = True
    if collisionResult == CollisionResults.AttackerEscapes:
        freePlaces = self.FindFreePlace()
        move = random.randint(0, len(freePlaces) - 1)
        self.x = freePlaces[move][0]
        self.y = freePlaces[move][1]
        self.age += 1
        self.actionMade = True
    if collisionResult == CollisionResults.AttackedDies or collisionResult == 0 or \
            collisionResult == CollisionResults.PlantIsEaten:
        self.x = possibleMoves[move][0]
        self.y = possibleMoves[move][1]
        self.age += 1
        self.actionMade = True
```

Action() for Antelope

```python
def TakeAction(self):
    possibleMoves = self.FindPlaceToMove()
    move = random.randint(0, len(possibleMoves) - 1)
    collisionResult = 0

    for organism in self.world.organisms:
        if organism.x == possibleMoves[move][0] and organism.y == possibleMoves[move][1]:
            collisionResult = organism.Collision(self)
            break

    if collisionResult == CollisionResults.AttackedReflects or collisionResult == CollisionResults.AnimalCreation:
        self.age += 1
        self.actionMade = True
    if collisionResult == CollisionResults.AttackedDies or collisionResult == CollisionResults.AttackedEscapes \
            or collisionResult == CollisionResults.PlantIsEaten or collisionResult == 0:
        self.age += 1
        self.x = possibleMoves[move][0]
        self.y = possibleMoves[move][1]
        self.actionMade = True
```

Action() for Cybersheep

Method FindPlaceToMove is different for Antelope, Cybersheep and Fox.

```python
def FindPlaceToMove(self):
    possibleMoves = []
    if self.y > 0:
        possibleMoves.append([self.x, self.y-1])
    if self.x > 0:
        possibleMoves.append([self.x-1, self.y])
    if self.y < self.world.height:
        possibleMoves.append([self.x, self.y+1])
    if self.x < self.world.width:
        possibleMoves.append([self.x+1, self.y])

    return possibleMoves
```

FindPlaceToMove() for Animal

```python
def FindPlaceToMove(self):
    possibleMoves = []
    if self.y > 0 and self.IsPositionSafe(self.x, self.y - 1):
        possibleMoves.append([self.x, self.y - 1])
    if self.x > 0 and self.IsPositionSafe(self.x - 1, self.y):
        possibleMoves.append([self.x - 1, self.y])
    if self.y < self.world.height and self.IsPositionSafe(self.x, self.y + 1):
        possibleMoves.append([self.x, self.y + 1])
    if self.x < self.world.width and self.IsPositionSafe(self.x + 1, self.y):
        possibleMoves.append([self.x + 1, self.y])
```

FindPlaceToMove() for Fox

```python
def FindPlaceToMove(self):
    numberOfMoves = 0
    nearestHogweed = None
    for organism in self.world.organisms:
        if organism.name == "Sosnowsky's Hogweed":
            verticalNumber = abs(organism.y - self.y)
            horizontalNumber = abs(organism.x - self.x)
            if numberOfMoves > (horizontalNumber + verticalNumber) or numberOfMoves == 0:
                numberOfMoves = horizontalNumber + verticalNumber
                nearestHogweed = organism
    possibleMoves = []
    if nearestHogweed is None:
        if self.y > 0:
            possibleMoves.append([self.x, self.y - 1])
        if self.x > 0:
            possibleMoves.append([self.x - 1, self.y])
        if self.y < self.world.height:
            possibleMoves.append([self.x, self.y + 1])
        if self.x < self.world.width:
            possibleMoves.append([self.x + 1, self.y])
    else:
        if nearestHogweed.y > self.y:
            possibleMoves.append([self.x, self.y + 1])
        if nearestHogweed.y < self.y:
            possibleMoves.append([self.x, self.y - 1])
        if nearestHogweed.x > self.x:
            possibleMoves.append([self.x + 1, self.y])
        if nearestHogweed.x < self.x:
            possibleMoves.append([self.x - 1, self.y])

    return possibleMoves
```

FindPlaceToMove() for Cybersheep

```
def FindPlaceToMove(self):
    possibleMoves = []
    if self.y > 0:
        possibleMoves.append([self.x, self.y - 1])
    if self.x > 0:
        possibleMoves.append([self.x - 1, self.y])
    if self.y < self.world.height:
        possibleMoves.append([self.x, self.y + 1])
    if self.x < self.world.width:
        possibleMoves.append([self.x + 1, self.y])
    if self.y > 1:
        possibleMoves.append([self.x, self.y - 2])
    if self.x > 1:
        possibleMoves.append([self.x - 2, self.y])
    if self.y < self.world.height - 1:
        possibleMoves.append([self.x, self.y + 2])
    if self.x < self.world.width - 1:
        possibleMoves.append([self.x + 2, self.y])

    return possibleMoves
```

FindPlaceToMove() for Antelope

A collision occurs when an animal enters another animal's field. When both animals are the same species, another animal of the same species will be created on the free field next to them. And the attacker will return to its previous location. When the attacker and attacked are of different species the animal with the greater strength wins and kills the weaker animal (if they are the same strength attacker wins). Only a turtle can reflect an attack of an animal with strength less than 5(then the attacker will return to the previous cell) and antelope has a 50% chance to escape from the fight(in such case it moves to a free neighboring cell).

```
def Collision(self, attacker):
    if type(attacker) == type(self):
        freeSpace = self.FindFreePlace() + attacker.FindFreePlace()

        if len(freeSpace) != 0:
            self.world.AddLogInfo("Pair of " + self.name + "s" + " made a baby")
            newOrganismPosition = random.randint(0, len(freeSpace) - 1)
            x = freeSpace[newOrganismPosition][0]
            y = freeSpace[newOrganismPosition][1]
            self.world.organisms.append(type(attacker)(x, y, self.world))
        return CollisionResults.AnimalCreation

    else:
        if attacker.IsAttackRepealed():
            self.world.AddLogInfo(attacker.name + " tried to attack " + self.name + ", but attacker escaped")
            return CollisionResults.AttackerEscapes
        if attacker.strength >= self.strength:
            self.world.AddLogInfo(attacker.name + " has just murdered " + self.name)
            self.world.organisms.remove(self)
            return CollisionResults.AttackedDies
        else:
            self.world.AddLogInfo(self.name + " has just murdered " + attacker.name)
            self.world.organisms.remove(attacker)
            return CollisionResults.AttackerDies
```

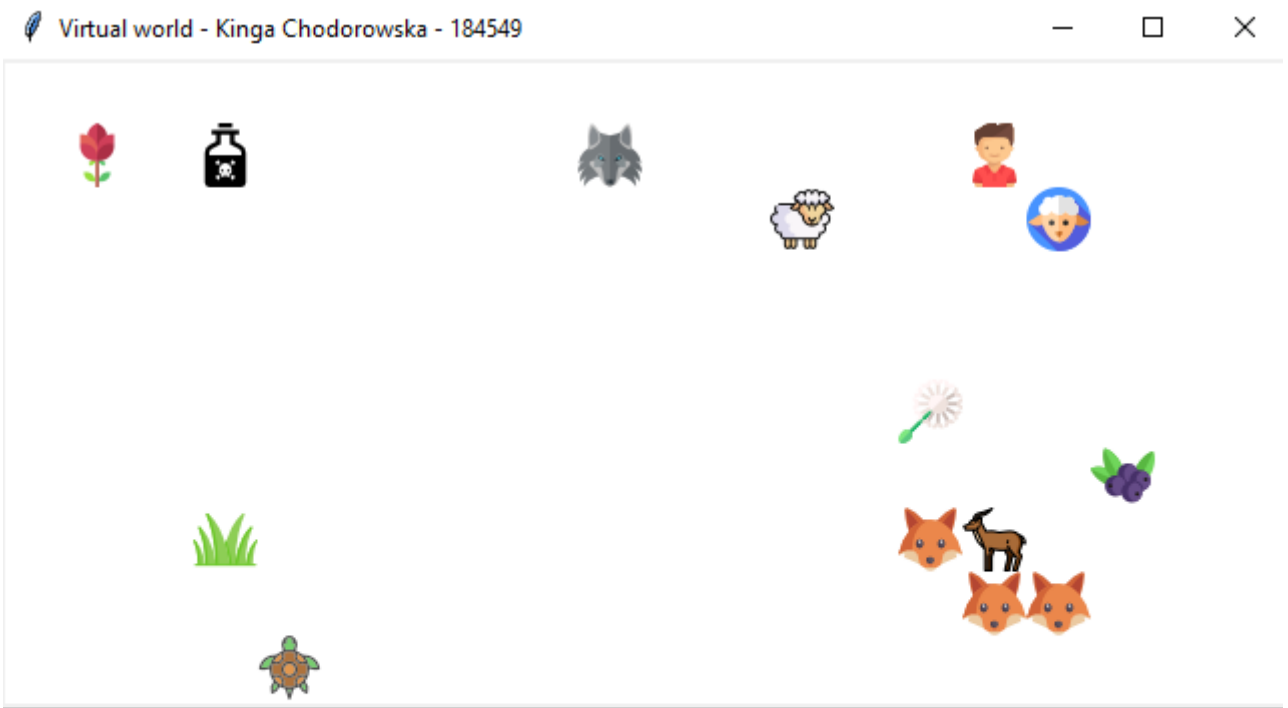
Collide() for Animal

```
def Collision(self, attacker):
    if type(attacker) == type(self):
        freeSpace = self.FindFreePlace() + attacker.FindFreePlace()

        if len(freeSpace) != 0:
            self.world.AddLogInfo("Pair of " + self.name + "s" + " made a baby")
            newOrganismPosition = random.randint(0, len(freeSpace) - 1)
            x = freeSpace[newOrganismPosition][0]
            y = freeSpace[newOrganismPosition][1]
            self.world.organisms.append(attacker(x, y, self.world))

        return CollisionResults.AnimalCreation
    else:
        if self.IsAttackReflected(attacker):
            return CollisionResults.AttackedReflects
        if attacker.IsAttackRepealed():
            self.world.AddLogInfo(attacker.name + " tried to attack " + self.name + ", but attacker escaped")
            return CollisionResults.AttackerEscapes
        if attacker.strength >= self.strength:
            self.world.AddLogInfo(attacker.name + " has just murdered " + self.name)
            self.world.organisms.remove(self)
            return CollisionResults.AttackedDies
        else:
            self.world.AddLogInfo(self.name + " has just murdered " + attacker.name)
            self.world.organisms.remove(attacker)
            return CollisionResults.AttackerDies
```

Collide() for Turtle

```python
def Collision(self, attacker):
    if type(attacker) == type(self):
        freeSpace = self.FindFreePlace() + attacker.FindFreePlace()
        if len(freeSpace) != 0:
            self.world.AddLogInfo("Pair of " + self.name + "s" + " made a baby")
            newOrganismPosition = random.randint(0, len(freeSpace) - 1)
            x = freeSpace[newOrganismPosition][0]
            y = freeSpace[newOrganismPosition][1]
            self.world.organisms.append(type(attacker)(x, y, self.world))
        return CollisionResults.AnimalCreation
    else:
        if self.IsAttackRepealed():
            self.world.AddLogInfo(attacker.name + " tried to attack " + self.name + ", but it escaped")
            freePlaces = self.FindFreePlace()
            move = random.randint(0, len(freePlaces) - 1)
            self.x = freePlaces[move][0]
            self.y = freePlaces[move][1]
            return CollisionResults.AttackedEscapes
        if attacker.strength >= self.strength:
            self.world.AddLogInfo(attacker.name + " has just murdered " + self.name)
            self.world.organisms.remove(self)
            return CollisionResults.AttackedDies
        else:
            self.world.AddLogInfo(self.name + " has just murdered " + attacker.name)
            self.world.organisms.remove(attacker)
            return CollisionResults.AttackerDies
```

Collide() for Antelope



Example of breeding and collision.

3. Implementation of all plants with sawing

Symbols of plants:

Grass	Sow thistle	Guarana	Belladonna	Sosnowsky's hogweed
				

Animals are created as soon as the board dimensions are entered. Animals are kept in the list - organisms, which is an element of the World class.
Plants also are created as soon as the board dimensions are entered. Plants also are stored in list. Each turn, plants can sow a new plant on a random free neighboring field with a probability of 15% and only sow thistle performs 3 attempts at spreading in each turn. Also Sosnowsky's hogweed has extra action which causes the death of every animal in its neighbourhood except cybersheep.

```
def TakeAction(self):
    freePlaces = []
    sow = random.randint(0, 99)  #15%
    if sow < 15:
        freePlaces = self.FindFreePlace()
        if len(freePlaces) != 0:
            newOrganismPosition = random.randint(0, len(freePlaces) - 1)
            x = freePlaces[newOrganismPosition][0]
            y = freePlaces[newOrganismPosition][1]
            self.world.organisms.append(type(self)(x, y, self.world))
            self.world.AddLogInfo(self.name + " spread 1 time")
    self.age += 1
    self.actionMade = True
```

Action() for plants

```python
def TakeAction(self):
    n = 0
    freePlaces = []
    for i in range(0, 2):
        sow = random.randint(0, 99)
        if sow < 15:
            freePlaces = self.FindFreePlace()
            if len(freePlaces) != 0:
                newOrganismPosition = random.randint(0, len(freePlaces) - 1)
                x = freePlaces[newOrganismPosition][0]
                y = freePlaces[newOrganismPosition][1]
                self.world.organisms.append(type(self)(x, y, self.world))
                n = n + 1
            else:
                break
    if n == 1:
        self.world.AddLogInfo(self.name + " spread 1 time")
    if n > 1:
        self.world.AddLogInfo(self.name + " spread " + str(n) + " time(s)")

    self.age += 1
    self.actionMade = True
```

Action() for sow thistle

```python
def TakeAction(self):
    immediateNeighbor = 0
    possibleNeighbor=[]
    if self.y > 0:
        possibleNeighbor.append([self.x, self.y - 1])
    if self.x > 0:
        possibleNeighbor.append([self.x - 1, self.y])
    if self.y < self.world.height:
        possibleNeighbor.append([self.x, self.y + 1])
    if self.x < self.world.width:
        possibleNeighbor.append([self.x + 1, self.y])
    for neighbor in possibleNeighbor:
        for organism in self.world.organisms:
            if organism.x == neighbor[0] and organism.y == neighbor[1]:
                if type(organism) != type(self) and organism.name != "Cybersheep":
                    self.world.AddLogInfo(self.name + " poisoned " + organism.name)
                    self.world.organisms.remove(organism)
                break

    freePlaces = []
    sow = random.randint(0, 99)  #15%
    if sow < 15:
        freePlaces = self.FindFreePlace()
        if len(freePlaces) != 0:
            newOrganismPosition = random.randint(0, len(freePlaces) - 1)
            x = freePlaces[newOrganismPosition][0]
            y = freePlaces[newOrganismPosition][1]
            self.world.organisms.append(type(self)(x, y, self.world))
            self.world.AddLogInfo(self.name + " spread 1 time")
    self.age += 1
    self.actionMade = True
```

Action() for Sosnowsky's hogweed

Plants do not move, so a collision only happens when an animal enters the plant's field then the plant gets eaten. However, when an animal eats guarana its strength increases by three and belladonna and Sosnowsky's hogweed kills all animals that eat it.

```
def Collision(self, attacker):
    self.world.AddLogInfo(attacker.name + " has just eaten " + self.name)
    self.world.organisms.remove(self)
    return CollisionResults.PlantIsEaten
```

Collide() for plants

```
def Collision(self, attacker):
    attacker.strength += 3
    self.world.AddLogInfo(attacker.name + " has just eaten " + self.name)
    self.world.organisms.remove(self)
    return CollisionResults.PlantIsEaten
```

Collide() for guarana

```
def Collision(self, attacker):
    self.world.AddLogInfo(attacker.name + " has just eaten " + self.name)
    if (attacker.name == "Cybersheep"):
        self.world.organisms.remove(self)
        return CollisionResults.PlantIsEaten
    else:
        self.world.AddLogInfo("and " + self.name + " poisoned " + attacker.name)
        self.world.organisms.remove(attacker)
        return CollisionResults.PlantKills
```

Collide() for Sosnowsky’s hogweed

```
def Collision(self, attacker):
    self.world.AddLogInfo(attacker.name + " has just eaten " + self.name + " and died")
    self.world.organisms.remove(self)
    self.world.organisms.remove(attacker)
    return CollisionResults.PlantKills
```

Collide() for Belladonna

Example of spreading of Sow Thistle and killing by Hogweed

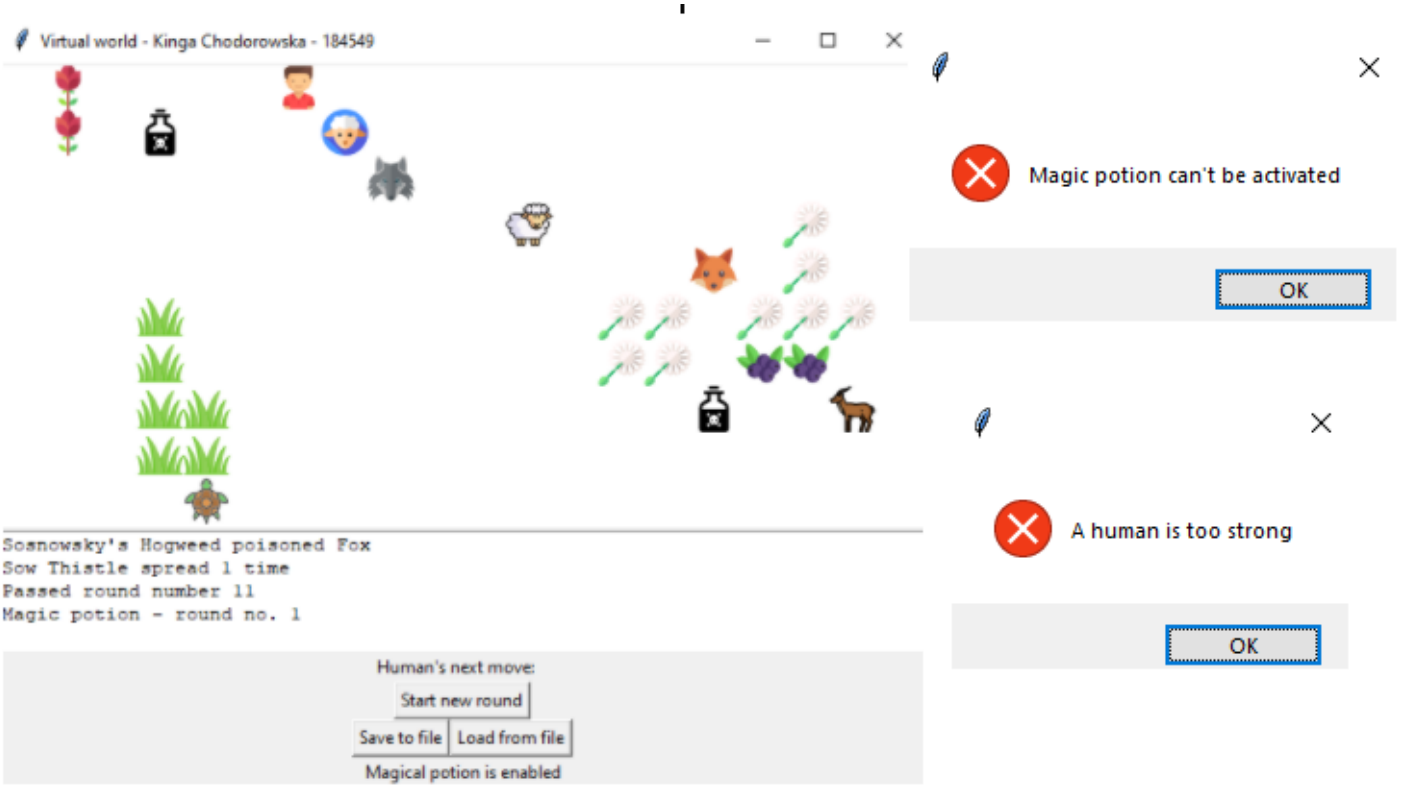
4. Implementation of a Human which can be controlled by arrow keys and implementation of Human's special ability

Human is created right after the rest of the organisms are created. A human also is stored in list organisms. The user can move the human by selecting one of the arrows or activate the super power by pressing ‘S’ then an arrow to move the human and activate the power. Human’s super power is magical potion. Human strength rises to 10 in the first turn and decreases by "1" per round until it returns to original value and if his strength is bigger than 9 users can't activate super power. When a special activity is activated, the screen shows that information. The user may reactivate the skill 10 rounds after activating it. If the human strength is greater than 9 pover is not activated.

```python
def TakeAction(self):
    self.NextSpecialAbilityRound()

    move = [self.x, self.y]
    if self.world.nextHumanMoveText.get() == "Right" and self.x < self.world.width:
        move = [self.x + 1, self.y]
    if self.world.nextHumanMoveText.get() == "Left" and self.x > 0:
        move = [self.x - 1, self.y]
    if self.world.nextHumanMoveText.get() == "Up" and self.y > 0:
        move = [self.x, self.y - 1]
    if self.world.nextHumanMoveText.get() == "Down" and self.y < self.world.height:
        move = [self.x, self.y + 1]
    if move == [self.x, self.y]:
        self.actionMade = True
        return

    collisionResult = 0

    for organism in self.world.organisms:
        if organism.x == move[0] and organism.y == move[1]:
            collisionResult = organism.Collision(self)
            break

    if collisionResult == CollisionResults.AttackedReflects:
        self.age += 1
        self.actionMade = True
    if collisionResult == CollisionResults.AttackedDies or collisionResult == CollisionResults.AttackedEscapes \
            or collisionResult == CollisionResults.PlantIsEaten or collisionResult == 0:
        self.age += 1
        self.x = move[0]
        self.y = move[1]
        self.actionMade = True
```

Action() for human

Menu after activating super ability



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

The user can save the state of their game to a file or load the game from a file at any time. This file contains in sequence : the dimensions of the board separated by a comma and line by line the names of the organisms with positions, strength, age, initiative. Human additionally includes specialAbilityRound.

```
10,20
Antelope,4,4,12,7,0
Cybersheep,11,4,2,4,0
Fox,3,7,17,7,0
Sheep,4,4,19,6,0
Wolf,9,5,8,1,0
Turtle,2,1,15,7,0
Sosnowsky's Hogweed,10,0,17,5,0
Guarana,0,0,5,9,0
Sow Thistle,0,0,7,4,0
Grass,0,0,5,1,0
Belladonna,99,0,17,2,0
Human,5,4,1,6,0,0
```

```python
def Load(self):
    filename = askopenfilename()
    f = open(filename, "r")
    lines = f.readlines()
    h = lines[0].split(",")[0]
    w = lines[0].split(",")[1]
    self.SetWorldSize(h, w, False)
    for i in range(1, len(lines)):
        splited = lines[i].split(",")
        organismType = splited[0]
        if organismType == "Antelope":
            self.organisms.append(Antelope(int(splited[3]), int(splited[4]), self))
        elif organismType == "Cybersheep":
            self.organisms.append(Cybersheep(int(splited[3]), int(splited[4]), self))
        elif organismType == "Human":
            self.organisms.append(Human(int(splited[3]), int(splited[4]), self))
        elif organismType == "Fox":
            self.organisms.append(Fox(int(splited[3]), int(splited[4]), self))
        elif organismType == "Sheep":
            self.organisms.append(Sheep(int(splited[3]), int(splited[4]), self))
        elif organismType == "Wolf":
            self.organisms.append(Wolf(int(splited[3]), int(splited[4]), self))
        elif organismType == "Turtle":
            self.organisms.append(Turtle(int(splited[3]), int(splited[4]), self))
        elif organismType == "Sosnowsky's Hogweed":
            self.organisms.append(SosnowskysHogweed(int(splited[3]), int(splited[4]), self))
        elif organismType == "Guarana":
            self.organisms.append(Guarana(int(splited[3]), int(splited[4]), self))
        elif organismType == "Sow Thistle":
            self.organisms.append(SowThistle(int(splited[3]), int(splited[4]), self))
        elif organismType == "Grass":
            self.organisms.append(Grass(int(splited[3]), int(splited[4]), self))
        elif organismType == "Belladonna":
            self.organisms.append(Belladonna(int(splited[3]), int(splited[4]), self))
        organism = self.organisms[len(self.organisms) - 1]
        organism.strength = int(splited[1])
        organism.initiative = int(splited[2])
        organism.age = int(splited[5])
        if type(organism) is Human:
            organism.specialAbilityRound = int(splited[6])
            if organism.specialAbilityRound > 0:
                self.specialAbility.pack()

    self.DrawWorld()
    tkinter.messagebox.showinfo(message="Loaded from file")
```

Load()

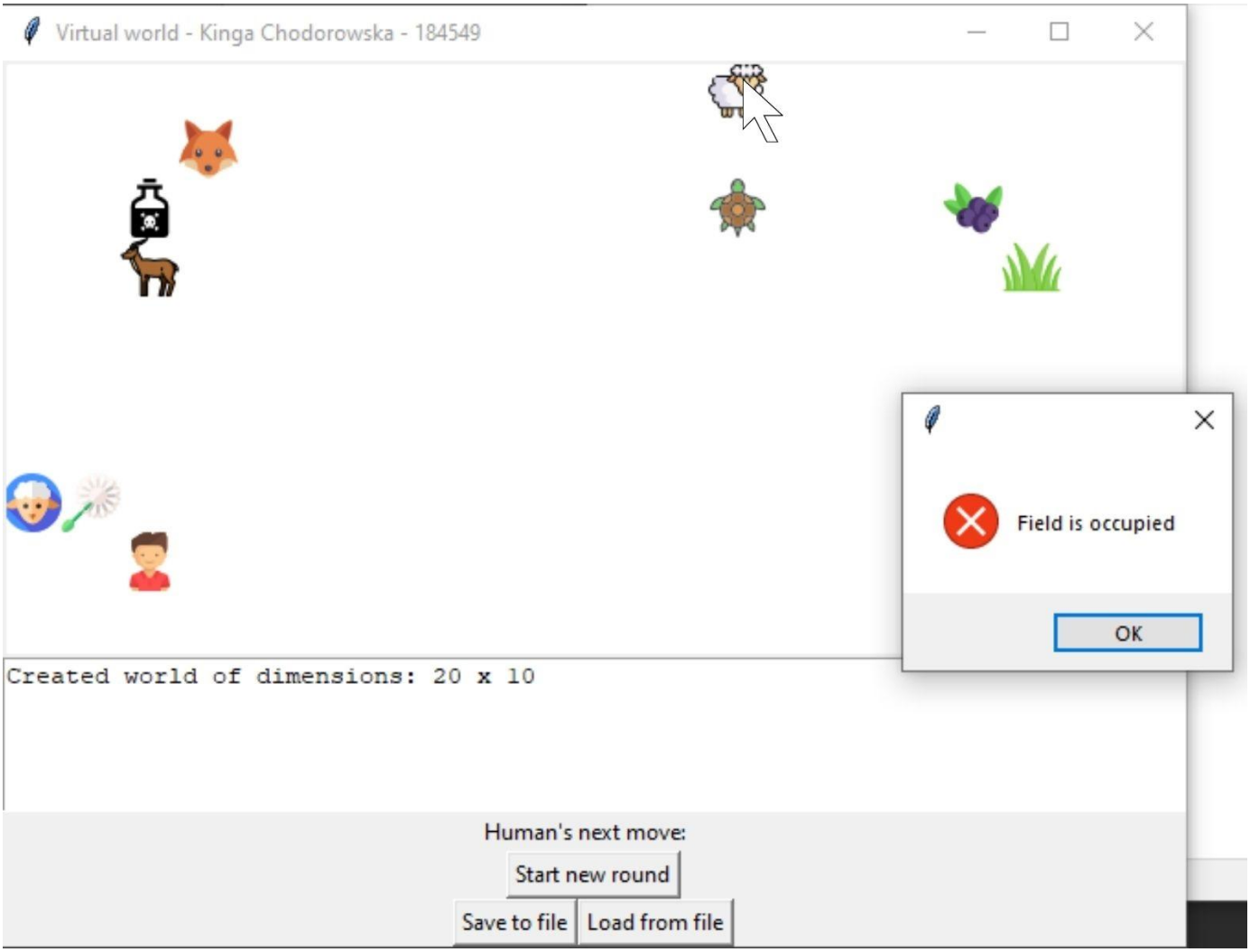
Save()

```
def Save(self):
    filename = asksaveasfilename()
    f = open(filename, "w")
    f.write(str(self.height + 1) + "," + str(self.width + 1) + "\n")
    for organism in self.organisms:
        if type(organism) is Human:
            f.write(organism.name + "," + str(organism.strength) + "," + str(organism.initiative) + "," + str(
                organism.x) + "," + str(organism.y) + "," + str(organism.age) + "," + str(
                organism.specialAbilityRound) + "\n")
        else:
            f.write(organism.name + "," + str(organism.strength) + "," + str(organism.initiative) + "," + str(
                organism.x) + "," + str(organism.y) + "," + str(organism.age) + "\n")
    f.close()
    tkinter.messagebox.showinfo(message="Saved to file")
```

6. Implementation of adding a new organism to the word by clicking on a free map cell

Callback method of the World class is used to create an organism at the user's desired location. Depending on how many times the user presses the selected place in this area, all the organisms (except for humans) are shown in turn. Adding an organism is not possible in an occupied space.

```
def Callback(self, event):
    x = (event.x - event.x % 32) / 32
    y = (event.y - event.y % 32) / 32
    empty = self.IsFieldEmpty(x, y)
    if x == self.addedManuallyX and y == self.addedManuallyY:
        self.organisms.pop()
        if self.addManualIndex == 0:
            self.organisms.append(Cybersheep(x, y, self))
        if self.addManualIndex == 1:
            self.organisms.append(Fox(x, y, self))
        if self.addManualIndex == 2:
            self.organisms.append(Sheep(x, y, self))
        if self.addManualIndex == 3:
            self.organisms.append(Wolf(x, y, self))
        if self.addManualIndex == 4:
            self.organisms.append(Turtle(x, y, self))
        if self.addManualIndex == 5:
            self.organisms.append(SosnowskysHogweed(x, y, self))
        if self.addManualIndex == 6:
            self.organisms.append(Guarana(x, y, self))
        if self.addManualIndex == 7:
            self.organisms.append(SowThistle(x, y, self))
        if self.addManualIndex == 8:
            self.organisms.append(Grass(x, y, self))
        if self.addManualIndex == 9:
            self.organisms.append(Belladonna(x, y, self))
        if self.addManualIndex == 10:
            self.addManualIndex = -1
            self.addedManuallyX = -1
            self.addedManuallyY = -1

        self.addManualIndex += 1
        self.DrawWorld()
        return
    if not empty:
        tkinter.messagebox.showerror(message="Field is occupied")
    else:
        self.organisms.append(Antelope(x, y, self))
        self.addedManuallyY = y
        self.addedManuallyX = x
        self.DrawWorld()
```


Virtual world - Kinga Chodorowska - 184549
Field is occupied
OK
Created world of dimensions: 20 x 10
Human's next move:
Start new round
Save to file
Load from file