

Опираясь на код, написанный на занятии (см. Иллюстрации ниже), необходимо найти звуки в интернете для следующих действий:

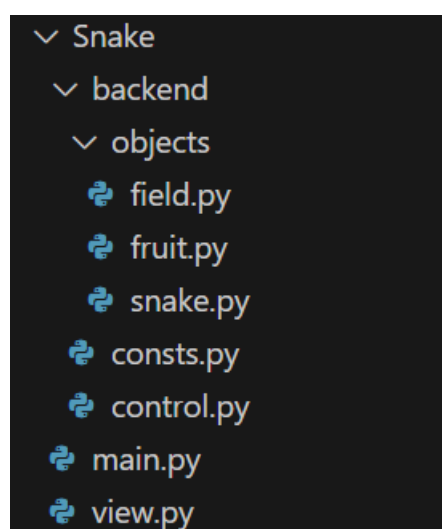
1. Поедание змеей фрукта.
2. Звук проигрыша.
3. Тихая фоновая музыка.

Необходимо подставить эти звуки в программу.

Для нахождения подходящего звука можно попробовать этот сайт - <https://mixkit.co/free-sound-effects/game/>

Код программы с занятия:

Программа имеет следующую файловую структуру:



Main.py:

```
1 import pygame
2 import sys
3 import backend.consts as const
4 from backend.objects.field import Field
5 from backend.objects.snake import Snake
6 from backend.objects.fruit import Fruit
7 from backend.control import Control
8 from view import View
9
10 class Main:
11     def __init__(self):
12
13         pygame.init()
14         self.WINDOW_SIZE = const.WINDOW_SIZE
15         self.screen = pygame.display.set_mode(self.WINDOW_SIZE)
16         self.view = View(self.screen)
17         self.snake = Snake()
18         self.gameField = Field(400,760,40,40)
19         self.fruit = Fruit ()
20         self.control = Control(self.view,self.snake,self.fruit,self.gameField)
21
22     def gameLoop(self):
23         game = True
24         while game:
25             for event in pygame.event.get(): # получаем действия от игрока
26                 if event.type == pygame.QUIT: # если был нажат крестик
27                     pygame.quit() # выйти из библиотеки
28                     sys.exit() # завершить работу программы
29
30             game=self.control.backProcess()
31             self.control.frontProcess()
32             pygame.display.flip()
33             pygame.time.Clock().tick(7)
34
35 if __name__ == '__main__':
36     Main().gameLoop()
```

View.py:

```
1 from pygame import draw
2
3 class View:
4     def __init__(self,screen):
5         self.screen = screen
6
7     def printRectList(self,rectList,color,width_borders = 0):
8         for rect in rectList:
9             draw.rect(self.screen, color, rect,width_borders)
```

Consts.py:

```
1 YELLOW = (255,255,0)
2 WHITE = (255,255,255)
3 BLACK = (0,0,0)
4 DARK = (128,128,128)
5
6 RECT_SIZE = 40
7
8 WINDOW_SIZE = (400, 760)
```

Controls.py:

```
1 from pygame import key
2 from pygame.locals import *
3 from .consts import *
4
5 # проверка вхождения одного списка в другой
6 def check_unification(list1,list2):
7     for element in list1:
8         if element in list2:
9             return True
10    else:
11        return False
12
13 class Control:
14     def __init__(self,view,snake,fruit,gameField):
15         self.view=view
16         self.snake=snake
17         self.fruit = fruit
18         self.gameField = gameField
19         self.returnValue = True
20
21     def backProcess(self):
22
23         keys = key.get_pressed()
24         action = None
25
26         if keys[K_LEFT]: action = 'LEFT'
27         if keys[K_RIGHT]: action = 'RIGHT'
28         if keys[K_UP]: action = 'UP'
29         if keys[K_DOWN]: action = 'DOWN'
30
31         self.snake.setDirection(action)
32         self.snake.moving()
33
34         if check_unification([self.fruit.getRect()],self.snake.getListRects())==True:
35             self.fruit.changePosition()
36             self.snake.grow()
37
38         self.returnValue = self.checkBorders()
39
40         return self.returnValue
41
42     def frontProcess(self):
43         self.view.screen.fill(DARK)
44         self.view.printRectList(self.gameField.getListRects(), WHITE,1)
45         self.view.printRectList([self.fruit.getRect()],self.fruit.color)
46         self.view.printRectList(self.snake.getListRects(),self.snake.color)
47
48     def checkBorders(self):
49         returnValue=True
50         for rect in self.snake.getListRects():
51             if ((rect.x+RECT_SIZE) > WINDOW_SIZE[0]) or rect.x < 0:
52                 returnValue = False
53             elif ((rect.y+RECT_SIZE) > WINDOW_SIZE[1]) or rect.y < 0:
54                 returnValue = False
55         return returnValue
```

field.py

```
1 from pygame import Rect
2
3 class Field:
4     def __init__(self>window_x>window_y*width*height):
5
6         self._rects = []
7
8         for x in range(0>window_x+1*width):
9             for y in range(0>window_y+1*height):
10                 self._rects.append(Rect(x, y, width, height))
11
12     def getListRects(self):
13         return self._rects
```

Fruit.py

```
1 from ..consts import RECT_SIZE,YELLOW
2 from pygame import Rect
3 from random import randint
4
5 class Fruit:
6     def __init__(self):
7         self._rect=Rect(randint(0,9)*RECT_SIZE,randint(0,19)*RECT_SIZE,RECT_SIZE,RECT_SIZE)
8         self.color = YELLOW
9
10    def changePosition(self):
11        self._rect.x = randint(0,9) * RECT_SIZE
12        self._rect.y = randint(0,19) * RECT_SIZE
13
14    def getRect(self):
15        return self._rect
```

Snake.py

```
1
2 from ..consts import *
3 from pygame import Rect
4
5 class Snake:
6     def __init__(self):
7         self._rects = [Rect(4*RECT_SIZE, 8*RECT_SIZE, RECT_SIZE, RECT_SIZE)]
8         self.color = BLACK
9         self.direction = 'RIGHT' # Начальное направление
10
11    def setDirection(self, action):
12        if action in ['LEFT', 'RIGHT', 'UP', 'DOWN']:
13            self.direction = action
14
15    def moving(self):
16        head = self._rects[0].copy() # Копируем голову змеи
17
18
19        if self.direction == 'LEFT': head.x -= RECT_SIZE
20        if self.direction == 'RIGHT': head.x += RECT_SIZE
21        if self.direction == 'UP': head.y -= RECT_SIZE
22        if self.direction == 'DOWN': head.y += RECT_SIZE
23
24        self._rects.insert(0, head) # Добавляем новую голову
25        self._rects.pop(-1) # Удаляем последний сегмент
26
27    def grow(self):
28        # Увеличиваем змейку, добавляя сегмент в конец
29        rect = self._rects[-1].copy()
30        self._rects.append(rect)
31
32    def getListRects(self):
33        return self._rects
```