Материалы к занятию

С подгрузкой звуков мы уже знакомились. В ней нет ничего сложного. Давайте лучше сегодня мы с вами попробуем написать обычную игру пасьянс. Для начала создадим стартовый шаблон

```
import arcade

SCREEN_WIDTH = 1024
SCREEN_HEIGHT = 768
SCREEN_TITLE = "Drag and Drop Cards"


class MyGame(arcade.Window):

   def __init__(self):
       super().__init__(SCREEN_WIDTH, SCREEN_HEIGHT,
SCREEN_TITLE)

       arcade.set_background_color(arcade.color.AMAZON)

   def setup(self):
       pass

   def on_draw(self):
       self.clear()

   def on_mouse_press(self, x, y, button, key_modifiers):
       pass

   def on_mouse_release(self, x: float, y: float, button: int,
                        modifiers: int):
       pass

   def on_mouse_motion(self, x: float, y: float, dx: float, dy:
float):
       pass


def main():
   window = MyGame()
   window.setup()
   arcade.run()


if __name__ == "__main__":
   main()
```

Следующий шаг - создать кучу спрайтов, по одному для каждой карты.

Создадим некоторые константы, используемые при позиционировании карт, и определении, какая карта какая.

```
CARD_SCALE = 0.6

CARD_WIDTH = 140 * CARD_SCALE
CARD_HEIGHT = 190 * CARD_SCALE

MAT_PERCENT_OVERSIZE = 1.25
MAT_HEIGHT = int(CARD_HEIGHT * MAT_PERCENT_OVERSIZE)
MAT_WIDTH = int(CARD_WIDTH * MAT_PERCENT_OVERSIZE)

VERTICAL_MARGIN_PERCENT = 0.10
HORIZONTAL_MARGIN_PERCENT = 0.10

BOTTOM_Y = MAT_HEIGHT / 2 + MAT_HEIGHT * VERTICAL_MARGIN_PERCENT

START_X = MAT_WIDTH / 2 + MAT_WIDTH * HORIZONTAL_MARGIN_PERCENT

CARD_VALUES = ["A", "2", "3", "4", "5", "6", "7", "8", "9", "10",
"J", "Q", "K"]
CARD_SUITS = ["Clubs", "Hearts", "Spades", "Diamonds"]
```

Далее создадим класс карт. Класс карты является подклассом arcade.Sprite. Он будет иметь атрибуты для масти и ценности карты, и автоматически загружать изображение для карты на основе этого.
Мы будем использовать все изображение в качестве хит-бокса, поэтому нам не нужно проходить через трудоемкий расчет хит-бокса. Поэтому мы отключаем его. В противном случае загрузка спрайтов заняла бы много времени.

```
class Card(arcade.Sprite):
   def __init__(self, suit, value, scale=1):
       self.suit = suit
       self.value = value
       self.image_file_name =
f":resources:images/cards/card{self.suit}{self.value}.png"
       super().__init__(self.image_file_name, scale,
hit_box_algorithm="None")
```

Создадим переменную для хранения всех карт

```
self.card_list = None
```

В методе setup укажем создание наших карт с помощью цикла

```
def setup(self):
   self.card_list = arcade.SpriteList()

   for card_suit in CARD_SUITS:
       for card_value in CARD_VALUES:
           card = Card(card_suit, card_value, CARD_SCALE)
           card.position = START_X, BOTTOM_Y
           self.card_list.append(card)
```

Отрисуем наши карты

```
def on_draw(self):
   self.clear()

   self.card_list.draw()
```

Далее добавим возможность поднимать, перетаскивать и опускать карты.

Во-первых, давайте добавим атрибуты, чтобы отслеживать, какие карты мы перемещаем. Поскольку мы можем перемещать несколько карт, мы сохраним это в виде списка. Если пользователь положит карту в неправильное место, нам нужно будет сбросить карту до исходного положения.
__init__

```
self.held_cards = None
self.held_cards_original_position = None
```

setup

```
def setup(self):
   self.held_cards = []

   self.held_cards_original_position = []
   …
```

Когда мы нажимаем на карту, мы хотим, чтобы она была последней выпавшей картой, чтобы она отображалась поверх всех других карт. В противном случае мы могли бы перетащить карту под другую карту, что выглядело бы странно. Добавим метод

```
def pull_to_top(self, card: arcade.Sprite):
   self.card_list.remove(card)
   self.card_list.append(card)
```

Поработаем с нажатием мыши.Когда пользователь нажмет кнопку мыши, требуется:

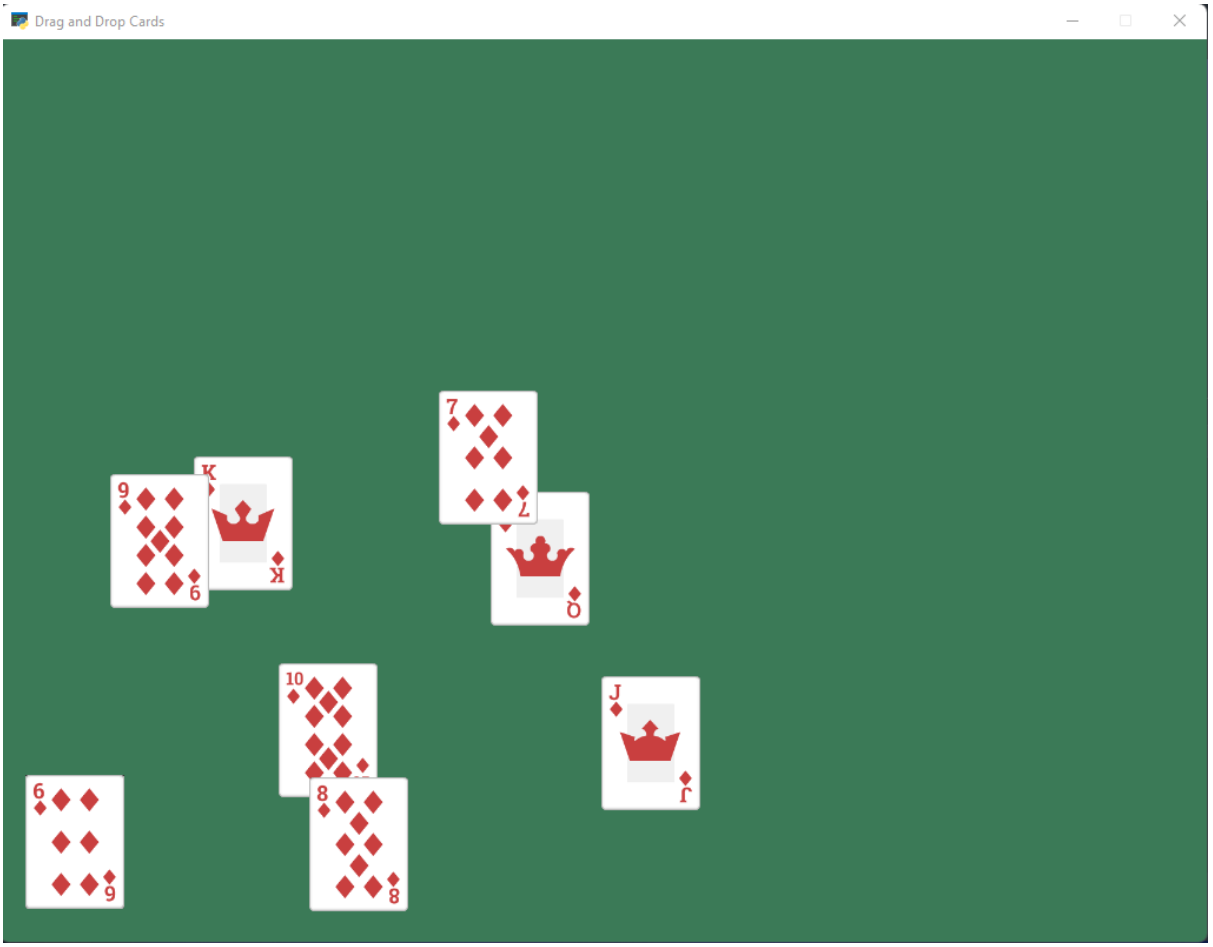
- Посмотреть, нажал ли он на карту
- Если это так, поместите эту карту в наш список удерживаемых карт.
- Сохранить исходную позицию карты

```
def on_mouse_press(self, x, y, button, key_modifiers):

   cards = arcade.get_sprites_at_point((x, y), self.card_list)

   if len(cards) > 0:
       primary_card = cards[-1]

       self.held_cards = [primary_card]
       self.held_cards_original_position =
[self.held_cards[0].position]
       self.pull_to_top(self.held_cards[0])
```

```
def on_mouse_motion(self, x: float, y: float, dx: float, dy:
float):
   for card in self.held_cards:
       card.center_x += dx
       card.center_y += dy
```

```
def on_mouse_release(self, x: float, y: float, button: int,
                     modifiers: int):
   if len(self.held_cards) == 0:
       return

   self.held_cards = []
```

Протестируем нашу программу



Далее мы создадим спрайты, которые будут выступать в качестве ориентиров для того, где стопки карт в нашей игре. Мы создадим их как спрайты, чтобы мы могли использовать обнаружение столкновений и выяснить, опускаем ли мы на них карту или нет.

Во-первых, мы создадим константы для среднего ряда из семи строк и для верхнего ряда из четырех строк. Мы также создадим константу того, как далеко друг от друга должна находиться каждая колода.

```
TOP_Y = SCREEN_HEIGHT - MAT_HEIGHT / 2 - MAT_HEIGHT *
VERTICAL_MARGIN_PERCENT

MIDDLE_Y = TOP_Y - MAT_HEIGHT - MAT_HEIGHT *
VERTICAL_MARGIN_PERCENT

X_SPACING = MAT_WIDTH + MAT_WIDTH * HORIZONTAL_MARGIN_PERCENT
```

Создадим переменную

```
self.pile_mat_list = None
```

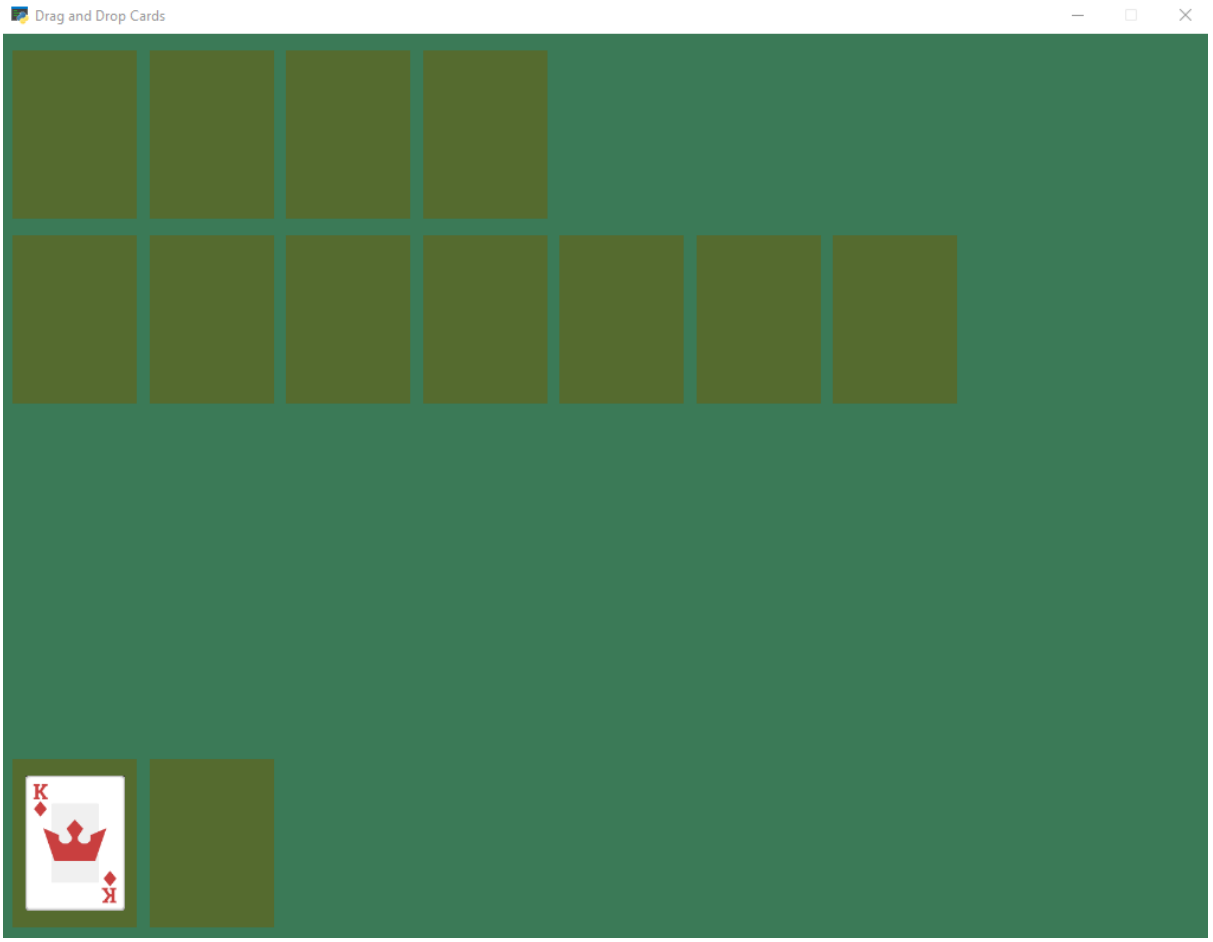
Создадим спрайты в методе setup

```
self.pile_mat_list: arcade.SpriteList = arcade.SpriteList()

pile = arcade.SpriteSolidColor(MAT_WIDTH, MAT_HEIGHT,
arcade.csscolor.DARK_OLIVE_GREEN)
pile.position = START_X, BOTTOM_Y
self.pile_mat_list.append(pile)

pile = arcade.SpriteSolidColor(MAT_WIDTH, MAT_HEIGHT,
arcade.csscolor.DARK_OLIVE_GREEN)
pile.position = START_X + X_SPACING, BOTTOM_Y
self.pile_mat_list.append(pile)

for i in range(7):
   pile = arcade.SpriteSolidColor(MAT_WIDTH, MAT_HEIGHT,
arcade.csscolor.DARK_OLIVE_GREEN)
   pile.position = START_X + i * X_SPACING, MIDDLE_Y
   self.pile_mat_list.append(pile)

for i in range(4):
   pile = arcade.SpriteSolidColor(MAT_WIDTH, MAT_HEIGHT,
arcade.csscolor.DARK_OLIVE_GREEN)
   pile.position = START_X + i * X_SPACING, TOP_Y
   self.pile_mat_list.append(pile)
```

Ну и отрисуем

```
self.pile_mat_list.draw()
```

Проверяем



Прямо сейчас мы можем перетащить карты куда угодно. Добавим код в on_key_release, который закрепляет карту на стопку. Если карта не попала на стопу, сбросим обратно в исходное место.

```
if arcade.check_for_collision(self.held_cards[0], pile):

    for i, dropped_card in enumerate(self.held_cards):
        dropped_card.position = pile.center_x, pile.center_y

    reset_position = False

if reset_position:
    for pile_index, card in enumerate(self.held_cards):
        card.position =
self.held_cards_original_position[pile_index]
```

Добавим перетасовку карт в методе setup

```
for pos1 in range(len(self.card_list)):
```

```
   pos2 = random.randrange(len(self.card_list))
   self.card_list.swap(pos1, pos2)
```

Мы будем вести отдельный список для каждой стопки карт. Когда мы перемещаем карту, нам нужно переместить позицию и указать в каком списке она находится.

```
CARD_VERTICAL_OFFSET = CARD_HEIGHT * CARD_SCALE * 0.3

PILE_COUNT = 13
BOTTOM_FACE_DOWN_PILE = 0
BOTTOM_FACE_UP_PILE = 1
PLAY_PILE_1 = 2
PLAY_PILE_2 = 3
PLAY_PILE_3 = 4
PLAY_PILE_4 = 5
PLAY_PILE_5 = 6
PLAY_PILE_6 = 7
PLAY_PILE_7 = 8
TOP_PILE_1 = 9
TOP_PILE_2 = 10
TOP_PILE_3 = 11
TOP_PILE_4 = 12
```

Создадим переменную

```
self.piles = None
```

В методе setup создадим список для каждой стопки.

```
self.piles = [[] for _ in range(PILE_COUNT)]

for card in self.card_list:
   self.piles[BOTTOM_FACE_DOWN_PILE].append(card)
```

Получив карту, вернем индекс, к какой стопке принадлежит эта карта

```
   def get_pile_for_card(self, card):
        for index, pile in enumerate(self.piles):
            if card in pile:
                return index
```

Затем извлечем карту из стопки, в которой она оказалась.

```
def remove_card_from_pile(self, card):
   for pile in self.piles:
       if card in pile:
           pile.remove(card)
           break
```

Наконец, переместим карту из одной стопки в другую.

```
def move_card_to_new_pile(self, card, pile_index):
   self.remove_card_from_pile(card)
   self.piles[pile_index].append(card)
```

Таким образом мы реализовали простой прототип игры пасьянса, правда без полной логики