Материалы к занятию.

```
import os

import arcade

SCREEN_WIDTH = 1000
SCREEN_HEIGHT = 650
SCREEN_TITLE = "Platformer"

CHARACTER_SCALING = 1
TILE_SCALING = 0.5
COIN_SCALING = 0.5
SPRITE_PIXEL_SIZE = 128
GRID_PIXEL_SIZE = SPRITE_PIXEL_SIZE * TILE_SCALING

PLAYER_MOVEMENT_SPEED = 7
GRAVITY = 1.5
PLAYER_JUMP_SPEED = 30

PLAYER_START_X = 64
PLAYER_START_Y = 256

# Layer Names from our TileMap
LAYER_NAME_MOVING_PLATFORMS = "Moving Platforms"
LAYER_NAME_PLATFORMS = "Platforms"
LAYER_NAME_COINS = "Coins"
LAYER_NAME_BACKGROUND = "Background"
LAYER_NAME_LADDERS = "Ladders"


class MyGame(arcade.Window):

   def __init__(self):

       super().__init__(SCREEN_WIDTH, SCREEN_HEIGHT,
SCREEN_TITLE)

       file_path = os.path.dirname(os.path.abspath(__file__))
       os.chdir(file_path)

       self.tile_map = None

       self.scene = None

       self.player_sprite = None

       self.physics_engine = None

       self.camera = None

       self.gui_camera = None
```

```
        self.end_of_map = 0

        self.score = 0

        self.collect_coin_sound =
arcade.load_sound(":resources:sounds/coin1.wav")
        self.jump_sound =
arcade.load_sound(":resources:sounds/jump1.wav")
        self.game_over =
arcade.load_sound(":resources:sounds/gameover1.wav")

    def setup(self):

        self.camera = arcade.Camera(self.width, self.height)
        self.gui_camera = arcade.Camera(self.width, self.height)

        map_name = ":resources:tiled_maps/map_with_ladders.json"

        layer_options = {
            LAYER_NAME_PLATFORMS: {
                "use_spatial_hash": True,
            },
            LAYER_NAME_MOVING_PLATFORMS: {
                "use_spatial_hash": False,
            },
            LAYER_NAME_LADDERS: {
                "use_spatial_hash": True,
            },
            LAYER_NAME_COINS: {
                "use_spatial_hash": True,
            },
        }

        self.tile_map = arcade.load_tilemap(map_name,
TILE_SCALING, layer_options)

        self.scene = arcade.Scene.from_tilemap(self.tile_map)

        self.score = 0

        image_source =
":resources:images/animated_characters/female_adventurer/femaleAd
venturer_idle.png"
        self.player_sprite = arcade.Sprite(image_source,
CHARACTER_SCALING)
        self.player_sprite.center_x = PLAYER_START_X
        self.player_sprite.center_y = PLAYER_START_Y
        self.scene.add_sprite("Player", self.player_sprite)

        self.end_of_map = self.tile_map.width * GRID_PIXEL_SIZE

        if self.tile_map.background_color:

arcade.set_background_color(self.tile_map.background_color)
```

```
        self.physics_engine = arcade.PhysicsEnginePlatformer(
            self.player_sprite,
            platforms=self.scene[LAYER_NAME_MOVING_PLATFORMS],
            gravity_constant=GRAVITY,
            ladders=self.scene[LAYER_NAME_LADDERS],
            walls=self.scene[LAYER_NAME_PLATFORMS]
        )

    def on_draw(self):

        self.clear()

        self.camera.use()

        self.scene.draw()

        self.gui_camera.use()

        score_text = f"Score: {self.score}"
        arcade.draw_text(
            score_text,
            10,
            10,
            arcade.csscolor.BLACK,
            18,
        )

    def on_key_press(self, key, modifiers):

        if key == arcade.key.UP or key == arcade.key.W:
            if self.physics_engine.is_on_ladder():
                self.player_sprite.change_y =
PLAYER_MOVEMENT_SPEED
            elif self.physics_engine.can_jump():
                self.player_sprite.change_y = PLAYER_JUMP_SPEED
                arcade.play_sound(self.jump_sound)
        elif key == arcade.key.DOWN or key == arcade.key.S:
            if self.physics_engine.is_on_ladder():
                self.player_sprite.change_y =
-PLAYER_MOVEMENT_SPEED
        elif key == arcade.key.LEFT or key == arcade.key.A:
            self.player_sprite.change_x = -PLAYER_MOVEMENT_SPEED
        elif key == arcade.key.RIGHT or key == arcade.key.D:
            self.player_sprite.change_x = PLAYER_MOVEMENT_SPEED

    def on_key_release(self, key, modifiers):

        if key == arcade.key.UP or key == arcade.key.W:
            if self.physics_engine.is_on_ladder():
                self.player_sprite.change_y = 0
        elif key == arcade.key.DOWN or key == arcade.key.S:
            if self.physics_engine.is_on_ladder():
                self.player_sprite.change_y = 0
        elif key == arcade.key.LEFT or key == arcade.key.A:
            self.player_sprite.change_x = 0
```

```
        elif key == arcade.key.RIGHT or key == arcade.key.D:
            self.player_sprite.change_x = 0

    def center_camera_to_player(self):
        screen_center_x = self.player_sprite.center_x -
(self.camera.viewport_width / 2)
        screen_center_y = self.player_sprite.center_y - (
            self.camera.viewport_height / 2
        )
        if screen_center_x < 0:
            screen_center_x = 0
        if screen_center_y < 0:
            screen_center_y = 0
        player_centered = screen_center_x, screen_center_y

        self.camera.move_to(player_centered, 0.2)

    def update(self, delta_time):
        self.physics_engine.update()

        self.scene.update_animation(
            delta_time, [LAYER_NAME_COINS, LAYER_NAME_BACKGROUND]
        )

        self.scene.update([LAYER_NAME_MOVING_PLATFORMS])

        coin_hit_list = arcade.check_for_collision_with_list(
            self.player_sprite, self.scene[LAYER_NAME_COINS]
        )

        for coin in coin_hit_list:

            if "Points" not in coin.properties:
                print("Warning, collected a coin without a Points
property.")
            else:
                points = int(coin.properties["Points"])
                self.score += points

            coin.remove_from_sprite_lists()
            arcade.play_sound(self.collect_coin_sound)

        self.center_camera_to_player()


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


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

Теперь давайте модифицируем наш код и добавим анимацию нашему персонажу.
Для начала добавим переменные, которые будут указывать в какую сторону смотрит наш персонаж

```
LAYER_NAME_PLAYER = "Player"
RIGHT_FACING = 0
LEFT_FACING = 1
```

Добавим функцию загрузки пары текстур, одна из них будет зеркальным отражением другой. Важно! Пишем не в классе нашей игры

```
def load_texture_pair(filename):
   return [
       arcade.load_texture(filename),
       arcade.load_texture(filename, flipped_horizontally=True),
   ]
```

Для персонажа мы создадим отдельный класс, в основном классе игры оставим только создание экземпляра класса

```
class PlayerCharacter(arcade.Sprite):
   def __init__(self):

       super().__init__()

       self.character_face_direction = RIGHT_FACING

       self.cur_texture = 0
       self.scale = CHARACTER_SCALING

       self.jumping = False
       self.climbing = False
       self.is_on_ladder = False

       main_path =
":resources:images/animated_characters/male_person/malePerson"

       self.idle_texture_pair =
load_texture_pair(f"{main_path}_idle.png")
       self.jump_texture_pair =
load_texture_pair(f"{main_path}_jump.png")
       self.fall_texture_pair =
load_texture_pair(f"{main_path}_fall.png")
```

```
        self.walk_textures = []
        for i in range(8):
            texture =
load_texture_pair(f"{main_path}_walk{i}.png")
            self.walk_textures.append(texture)

        self.climbing_textures = []
        texture = arcade.load_texture(f"{main_path}_climb0.png")
        self.climbing_textures.append(texture)
        texture = arcade.load_texture(f"{main_path}_climb1.png")
        self.climbing_textures.append(texture)

        self.texture = self.idle_texture_pair[0]

        self.hit_box = self.texture.hit_box_points
```

Класс персонажа наследуется от класса arcade.Sprite. В методе инициализации мы подгружаем списки текстур для прыжка, состояния покоя, движения и проигрыша. Также мы определяем булевы переменные: находимся ли мы в прыжке и т.п . Теперь для работы анимации нам необходим метод update_animation для нашего нового класса, который будет отвечать за анимацию. В нем мы пропишем условия, благодаря которым будет происходить переключение анимации.

```
def update_animation(self, delta_time: float = 1 / 60):
    if self.change_x < 0 and self.character_face_direction ==
RIGHT_FACING:
        self.character_face_direction = LEFT_FACING
    elif self.change_x > 0 and self.character_face_direction ==
LEFT_FACING:
        self.character_face_direction = RIGHT_FACING

    if self.is_on_ladder:
        self.climbing = True
    if not self.is_on_ladder and self.climbing:
        self.climbing = False
    if self.climbing and abs(self.change_y) > 1:
        self.cur_texture += 1
        if self.cur_texture > 7:
            self.cur_texture = 0
    if self.climbing:
        self.texture = self.climbing_textures[self.cur_texture //
4]
        return

    if self.change_y > 0 and not self.is_on_ladder:
        self.texture =
self.jump_texture_pair[self.character_face_direction]
        return
    elif self.change_y < 0 and not self.is_on_ladder:
```

```
        self.texture =
self.fall_texture_pair[self.character_face_direction]
        return

    if self.change_x == 0:
        self.texture =
self.idle_texture_pair[self.character_face_direction]
        return

    self.cur_texture += 1
    if self.cur_texture > 7:
        self.cur_texture = 0
    self.texture = self.walk_textures[self.cur_texture][
        self.character_face_direction
    ]
```

Класс персонажа готов. Теперь давайте перейдем к нашему основному классу и перепишем его.

Во первых добавим переменные в метод __init__

```
self.left_pressed = False
self.right_pressed = False
self.up_pressed = False
self.down_pressed = False
self.jump_needs_reset = False
```

В setup создадим экземпляр игрока и укажем координаты

```
self.player_sprite = PlayerCharacter()
self.player_sprite.center_x = PLAYER_START_X
self.player_sprite.center_y = PLAYER_START_Y
self.scene.add_sprite(LAYER_NAME_PLAYER, self.player_sprite)
```

Вынесем для удобства условия управления в отдельный метод

```
def process_keychange(self):
    if self.up_pressed and not self.down_pressed:
        if self.physics_engine.is_on_ladder():
            self.player_sprite.change_y = PLAYER_MOVEMENT_SPEED
        elif (
            self.physics_engine.can_jump(y_distance=10)
            and not self.jump_needs_reset
        ):
            self.player_sprite.change_y = PLAYER_JUMP_SPEED
```

```
            self.jump_needs_reset = True
            arcade.play_sound(self.jump_sound)
    elif self.down_pressed and not self.up_pressed:
        if self.physics_engine.is_on_ladder():
            self.player_sprite.change_y = -PLAYER_MOVEMENT_SPEED

    if self.physics_engine.is_on_ladder():
        if not self.up_pressed and not self.down_pressed:
            self.player_sprite.change_y = 0
        elif self.up_pressed and self.down_pressed:
            self.player_sprite.change_y = 0

    if self.right_pressed and not self.left_pressed:
        self.player_sprite.change_x = PLAYER_MOVEMENT_SPEED
    elif self.left_pressed and not self.right_pressed:
        self.player_sprite.change_x = -PLAYER_MOVEMENT_SPEED
    else:
        self.player_sprite.change_x = 0
```

Также перепишем стандартные методы управления

```
def on_key_press(self, key, modifiers):

    if key == arcade.key.UP or key == arcade.key.W:
        self.up_pressed = True
    elif key == arcade.key.DOWN or key == arcade.key.S:
        self.down_pressed = True
    elif key == arcade.key.LEFT or key == arcade.key.A:
        self.left_pressed = True
    elif key == arcade.key.RIGHT or key == arcade.key.D:
        self.right_pressed = True

    self.process_keychange()

def on_key_release(self, key, modifiers):

    if key == arcade.key.UP or key == arcade.key.W:
        self.up_pressed = False
        self.jump_needs_reset = False
    elif key == arcade.key.DOWN or key == arcade.key.S:
        self.down_pressed = False
    elif key == arcade.key.LEFT or key == arcade.key.A:
        self.left_pressed = False
    elif key == arcade.key.RIGHT or key == arcade.key.D:
        self.right_pressed = False
    self.process_keychange()
```

В методе on_update добавим обновление нашей анимации

```
if self.physics_engine.can_jump():
   self.player_sprite.can_jump = False
else:
   self.player_sprite.can_jump = True

if self.physics_engine.is_on_ladder() and not
self.physics_engine.can_jump():
   self.player_sprite.is_on_ladder = True
   self.process_keychange()
else:
   self.player_sprite.is_on_ladder = False
   self.process_keychange()

self.scene.update_animation(
   delta_time, [LAYER_NAME_COINS, LAYER_NAME_BACKGROUND,
LAYER_NAME_PLAYER]
)
```

Создадим стартовый шаблон

```
import arcade

SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600
SCREEN_TITLE = "Игра"


class Game(arcade.Window):
   def __init__(self, width, height, title):
       super().__init__(width, height, title)

   def setup(self):
       pass

   def on_draw(self):
       self.clear()

   def on_update(self, delta_time):
       pass

window = Game(SCREEN_WIDTH, SCREEN_HEIGHT, SCREEN_TITLE)
window.setup()
arcade.run()
```

Добавим фоновое изображение

__init__

```
self.bg =
arcade.load_texture(':resources:images/backgrounds/abstract_2.jpg
')
```

on_draw

```
arcade.draw_texture_rectangle(SCREEN_WIDTH / 2, SCREEN_HEIGHT /
2, SCREEN_WIDTH, SCREEN_HEIGHT, self.bg)
```

Наш фон отрисовался. Теперь создадим класс нашего игрока

```
class Player(arcade.Sprite):
   pass
```

Теперь создадим экземпляр класса и укажем текстуру и координаты

```
class Game(arcade.Window):
   def __init__(self, width, height, title):
       super().__init__(width, height, title)
       self.bg =
arcade.load_texture(':resources:images/backgrounds/abstract_2.jpg
')
       self.player = None

   def setup(self):
       self.player =
Player(":resources:images/animated_characters/male_adventurer/mal
eAdventurer_walk0.png", 1)
       self.player.center_x = 170
       self.player.center_y = 155
```

Отрисуем нашего персонажа в on_draw

```
self.player.draw()
```

Отлично, но наш персонаж висит в воздухе. Давайте как прежде мы создадим землю

```
self.wall_list = None
```

```
self.wall_list = arcade.SpriteList(use_spatial_hash=True)
for x in range(0, 1250, 64):
   wall = arcade.Sprite(":resources:images/tiles/grassMid.png",
1)
   wall.center_x = x
   wall.center_y = 32
   self.wall_list.append(wall)
```

Отрисуем нашу землю

```
self.wall_list.draw()
```

Давайте теперь добавим анимацию нашему персонажу. Так как он всегда будет бежать и не будет других состояний, то мы просто добавим дополнительные текстуры для нашего персонажа, чтобы они постоянно переключались между собой

```
for i in range(8):
   self.player.append_texture(
       arcade.load_texture(

f":resources:images/animated_characters/male_adventurer/maleAdven
turer_walk{i}.png"))
```

Теперь модифицируем класс игрока и добавим метод update_animation

```
class Player(arcade.Sprite):
   i = 0

   def update_animation(self, delta_time):
       if self.i == len(self.textures) - 1:
           self.i = 0
       else:
           self.i += 1
           self.set_texture(self.i)
```

и конечно же укажем это в методе on_update

```
self.player.update_animation(delta_time)
```

Теперь добавим управление для нашего персонажа. Из управления будет только прыжок. Только сделаем мы это другим способом.

Изменим класс нашего персонажа, а также введем значение гравитации

```
GRAVITY = 0.5
class Player(arcade.Sprite):
   i = 0
   jump = False
   def update(self):
       self.center_y += self.change_y
       self.change_y -= GRAVITY
       if self.center_y < 155:
           self.center_y = 155
           self.jump = False
```

Добавим управление

```
def on_key_press(self, key, modifiers):
   if key == arcade.key.SPACE and not self.player.jump:
       self.player.change_y = 12
       self.player.jump = True
```

и вызовем update для нашего персонажа

```
def on_update(self, delta_time):
   self.player.update_animation(delta_time)
   self.player.update()
```

Давайте теперь добавим в нашу игру препятствие, счет, победу и проигрыш

Начнем с препятствия

```
class Cactus(arcade.Sprite):
   pass
```

```
self.cactus = Cactus(":resources:images/tiles/cactus.png", 1)
self.cactus.center_x = SCREEN_WIDTH
self.cactus.center_y = 155
```

```
self.cactus.draw()
```

```

```

Отлично кактус отрисовался. Теперь давайте добавим ему движения.

```
class Cactus(arcade.Sprite):
   def update(self):
       self.center_x -= self.change_x
```

```
self.cactus.change_x = 5
```

```
def on_update(self, delta_time):
   self.player.update_animation(delta_time)
   self.player.update()
   self.cactus.update()
```

Отлично кактус двигается, но давайте укажем чтобы он постоянно появлялся с правой стороны экрана, после того как скроется слева. Для этого пропишем условие в классе кактуса, а также воспользуемся random для более случайного появления нашего кактуса

```
class Cactus(arcade.Sprite):
   def update(self):
       self.center_x -= self.change_x
       if self.right < 0:
           self.left = SCREEN_WIDTH + random.randint(0,
SCREEN_WIDTH)
```

Теперь нам требуется обработать столкновение кактуса с персонажем. Для этого мы введем булеву переменную в __init__, которая будет отвечать за статус игры

```
self.game = True
```

и теперь добавим условие, чтобы update и управление работало только если данная переменная True

```
def on_update(self, delta_time):
   if self.game:
       self.player.update_animation(delta_time)
       self.player.update()
       self.cactus.update()

def on_key_press(self, key, modifiers):
   if self.game:
       if key == arcade.key.SPACE and not self.player.jump:
           self.player.change_y = 12
           self.player.jump = True
```

Отлично теперь обработаем само столкновение в on_update и если оно произошло то будем менять значение статуса игры

```
if arcade.check_for_collision(self.player, self.cactus):
   self.game = False
```