PYGAME.

ЦЕЛЬ: **Изучить принцип реализации слежения камеры за персонажем игры**

ПРИМЕЧАНИЕ: **имеется ввиду вид в игре. Во многих играх (в которых игровой мир больше чем границы экрана) мы видим на экране вид на персонажа от третьего лица, можно представить, что его, как будто снимает камера и передает изображение нам на экран. И куда бы не передвигался персонаж, камера следует за ним, а окружающий мир уходит за границы экрана.**

ПЛАНИРОВАНИЕ

1. Реализация принципа камеры, привязанной к спрайту

Чтобы сделать так, чтобы в игре мир (карта, локация) был больше по размерам чем экран монитора?

Поможет нам в этом объект в библиотеке pygame – Surface и его метод – blit(). Мы уже знакомы с этим объектом и его методом.

Вспомним:

Surface – это виртуальная прямоугольная поверхность, на которой можно отображать (рисовать) спрайты и геометрические фигуры. Мы можем создавать их в программе хоть сколько, и они будут храниться в оперативной памяти. Это аналог листа бумаги, на котором мы можем рисовать или приклеивать какие-нибудь рисунки.

В pygame – есть главная поверхность, это та, которая выводится на экран монитора. И мы до этого момента пользовались только ей. Вспомним код, который создает ее:

w = pygame.display.set_mode ((1279, 700))

Видим, что метод set_mode() у модуля display (а **display** – это модуль), создает объект - поверхность (Surface). И она будет главной и всегда выводиться на экран монитора с помощью метода **update()** модуля **display**:

pygame.display.update()

Но мы можем создать и обычную поверхность:

Surf1 = pygame.Surface ((1280, 2000))

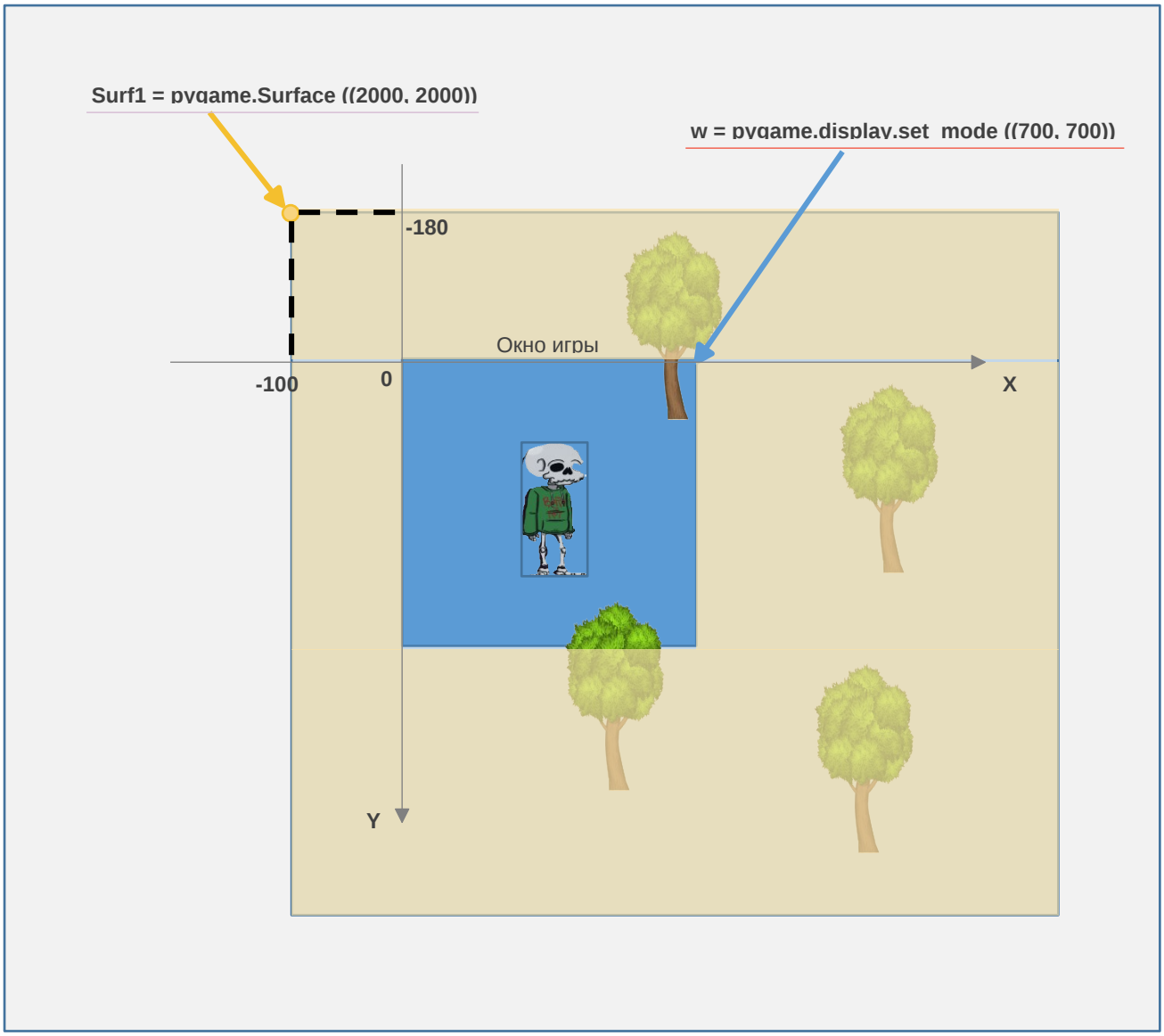
А теперь вспомним метод blit() у объекта Surface. Этот метод отображает на поверхности объект Image в заданных координатах:

Surf1.blit(Image, (x, y))

Оказывается, что с помощью этого метода можно отображать на поверхности не только объекты класса Image, но и объекты класса Surface, иначе говоря, на поверхности можно отображать другие поверхности.

Значит мы можем все рисовать на нами созданной поверхности Surf1, а потом отображать ее на главной поверхности – **w**, с помощью метода – **blit()** в таких координатах, чтобы наш персонаж всегда попадал на середину главной поверхности, то есть отображался на экране монитора всегда в центре окна игры.

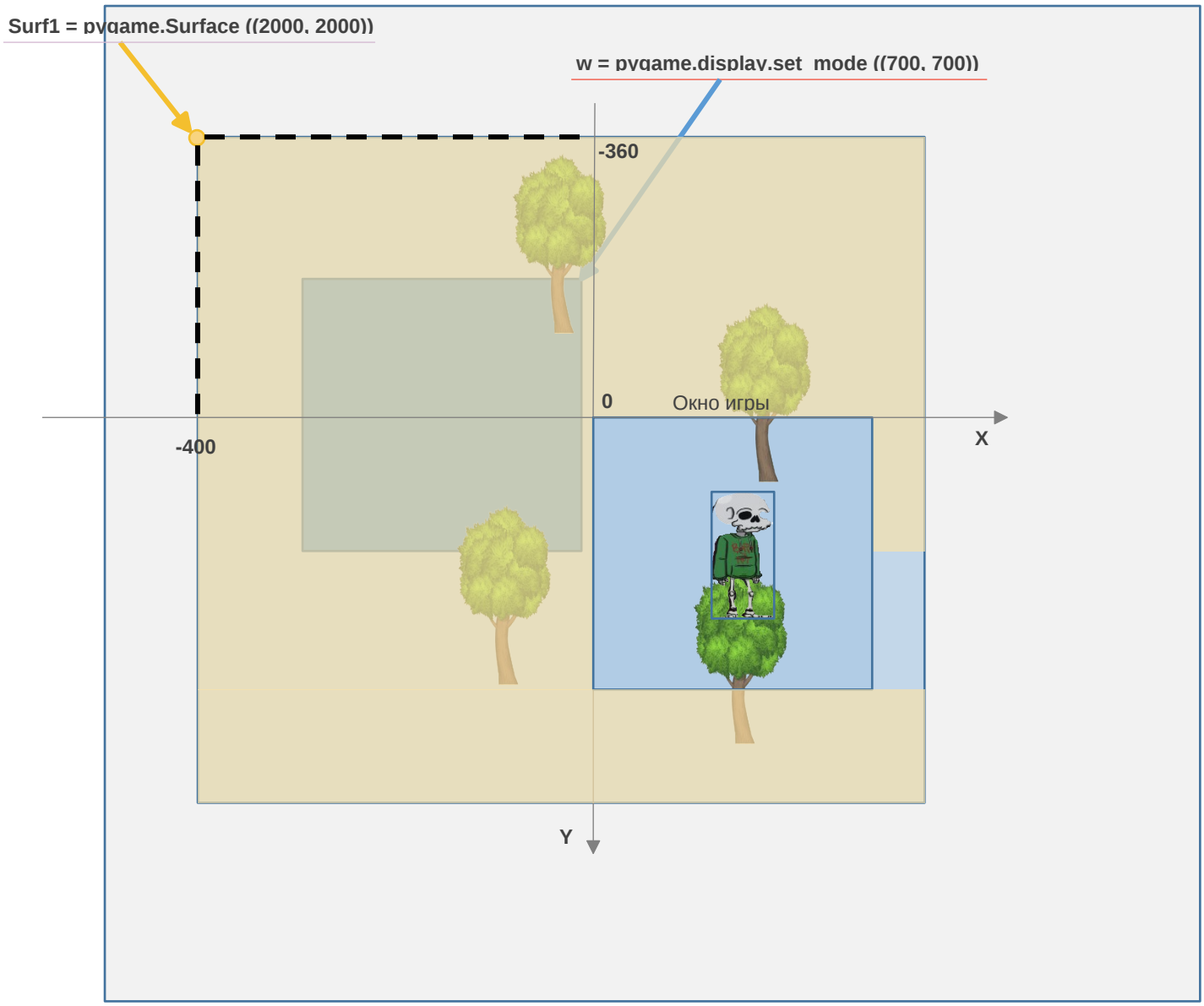
Схематически это выглядит так:



Как видно, чтобы наш персонаж попал в синюю область, необходимо поверхность Surf1, на которой отображен спрайт персонажа, отобразить на поверхности – **w** в координатах: x = - 100 y = -180 (числа приблизительные исходя из масштабов схемы):

w.blit(Surf1, (-100, -180))

Если спрайт перешел на поверхности Surf1 в другое место:



То отображаем поверхность Surf1 на поверхности – **w** уже в других координатах:

w.blit(Surf1, (-360, -400))

! Проблемная задача: Выразить координаты на поверхности игрового окна (в нашем случае это – w), в которых необходимо отображать поверхность Surf1 (на которой рисуются все спрайты) – через координаты спрайта на поверхности Surf1

Решение:

x = w.width / 2 - sprite.x

y = w.height / 2 - sprite.y

Значит нам необходимо хранить эти координаты, для этого будет логично создать класс, который можно назвать **Камера.** Ведь экран как камера следит за спрайтом.

```
class Camera():
    def __init__(self, rect, halfwidth, halfheight):
        self.x = halfwidth - rect.x
        self.y = halfheight - rect.y
    def update(self, rect):
        self.__init__(rect)
```

Создадим нашу камеру – объект класса **Camera**

И после того, как у всех спрайтов и групп спрайтов будут обновлены их координаты и изображения, будем обновлять положение камеры:

```
……
cam = Camera(Player.rect)
……
……
cam.update(Player.rect)
……
```

! Проблемная задача: внедрить принцип камеры в нашу игру.

2. Определение границ игрового мира, настройка камеры.

В нашей игре, как мы видим, персонаж может скрываться за границы своей поверхности, то есть, за границы той поверхности, на которой он отображается. Поэтому нужно установить ограничения, заблокировать изменение координат, когда персонаж доходит то какого-нибудь края экрана. Установим эти ограничения для передвижения по горизонтали. Для этого в методе update() в классе Player там, где мы меняем координаты при событиях нажатия клавиш : стрелка вправо и стрелка влево, установим проверки:

```
if self.rect.x + self.rect.width + self.speedx < WorldWidth:
        self.rect.x += self.speedx
if self.rect.x + self.speedx > 0:
        self.rect.x += self.speedx
```

И помимо персонажа, нам нужно настроить камеру. Ведь она тоже убегает за экран.

Нам нужно сделать так, чтобы камера доходила только до края:

Добавим проверки в код класса Camera:

```
class Camera():
    def __init__(self, sprite):
        self.x = 0
        self.y = -800
    def update(self, sprite):
       if sprite.rect.x < WorldWidth - ScreenWidth/2 and sprite.rect.x > ScreenWidth/2:
            self.x = int(ScreenWidth/2) - sprite.rect.x
        if sprite.rect.y < WorldHeight - ScreenHeight/2 and sprite.rect.y > ScreenHeight/2:
            self.y = ScreenHeight/2 - sprite.rect.y
```

3. Рефлексия

- Сегодня мы реализовали принцип слежения камеры за спрайтом игрока

- Познакомились с классом Surface и его методами