# Игра “Танки” написанная на Python с использованием библиотеки Pygame

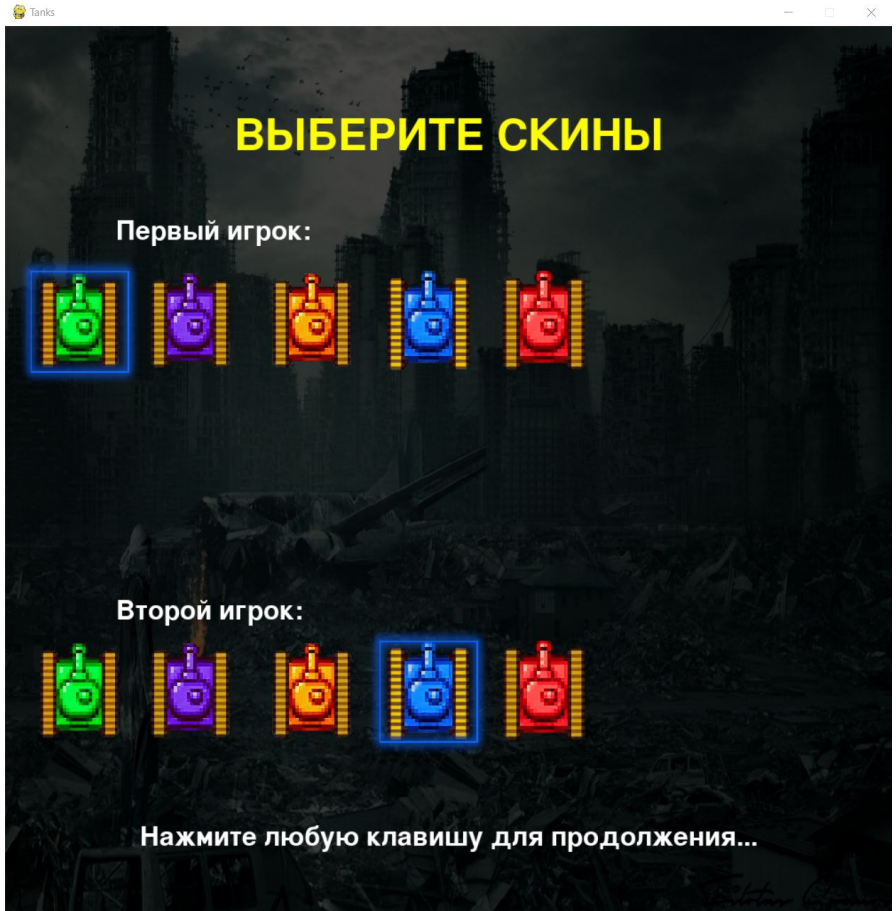
# Суть игры “Танки”:

- Данная игра реализует:
- Выбор скинов в начале игры
- Управление танками на соответствующие кнопки
- Возможность разрушать объекты (но не все, зависит от объекта)
- Возможность прятаться в кустах
- Возможность уничтожение вражеского танка и респавн

Файл с кодом **main.py**

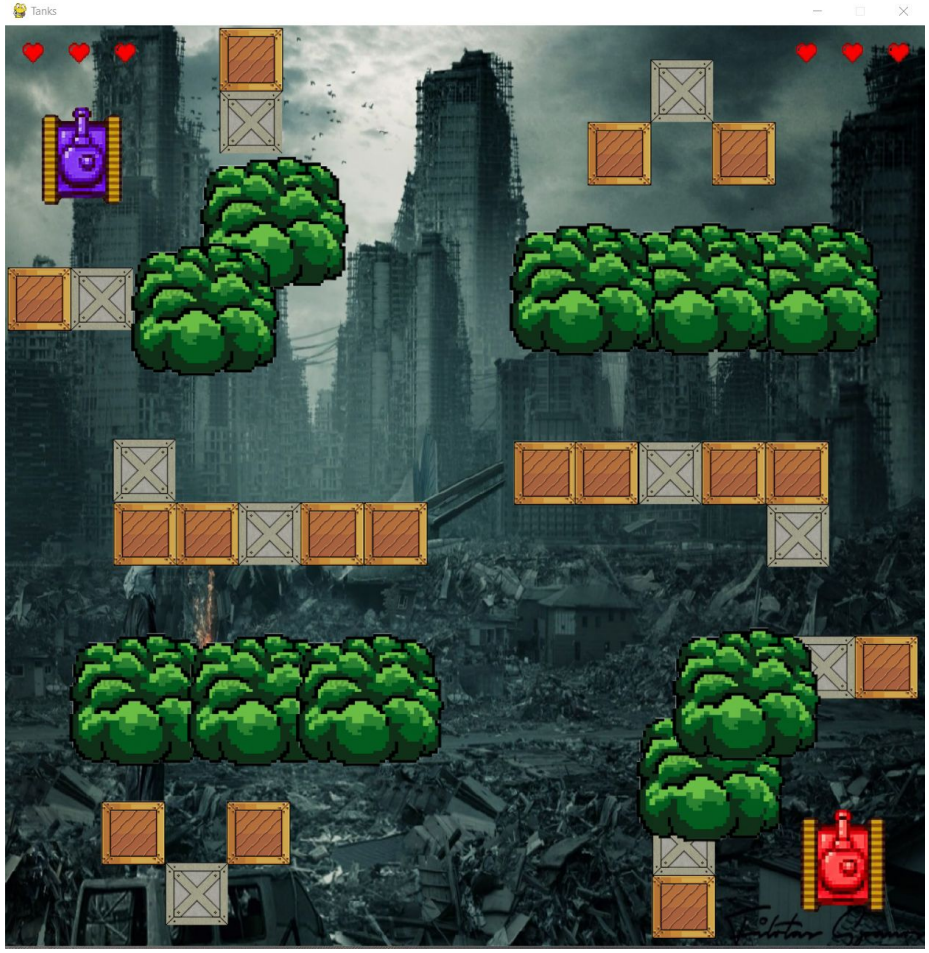
# Игра с точки зрения пользователя

# Начальный экран: возможность выбора скинов

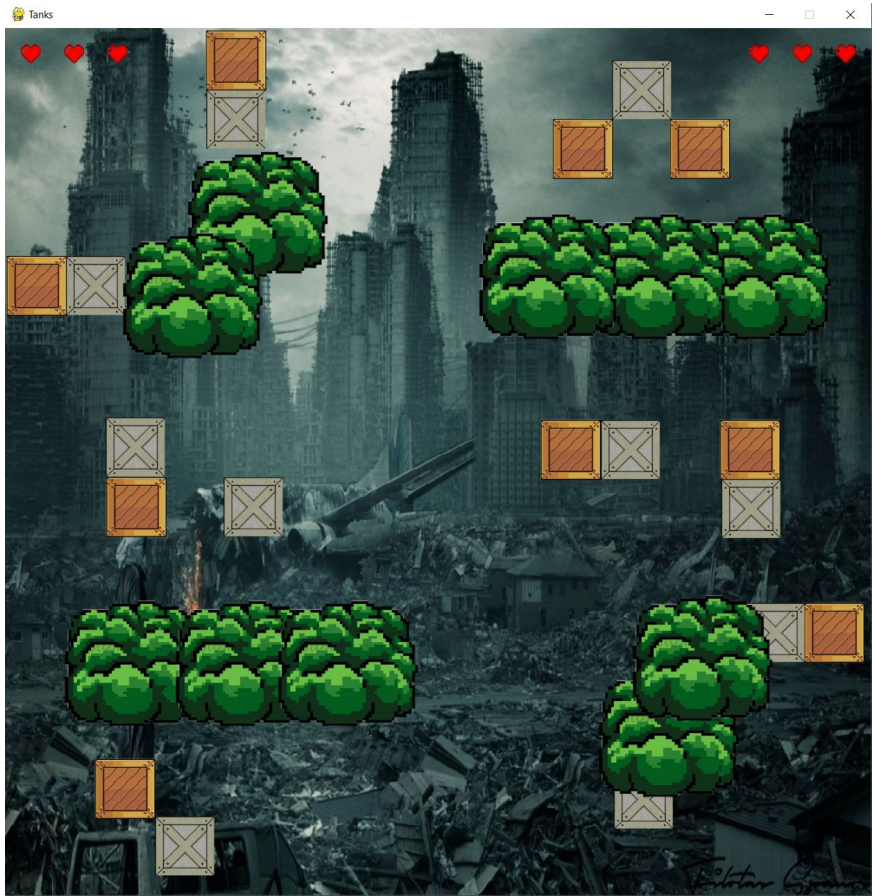


# Следующий экран: инструкция по управлению

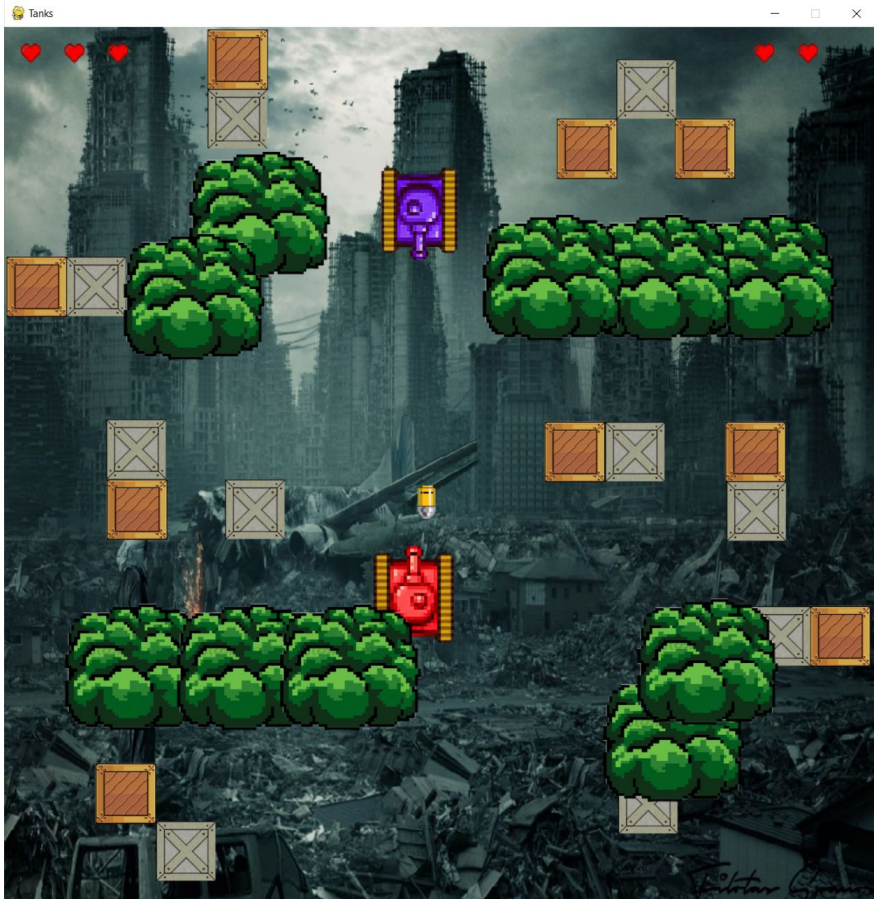
# Начало игры: танки, объекты, здоровье



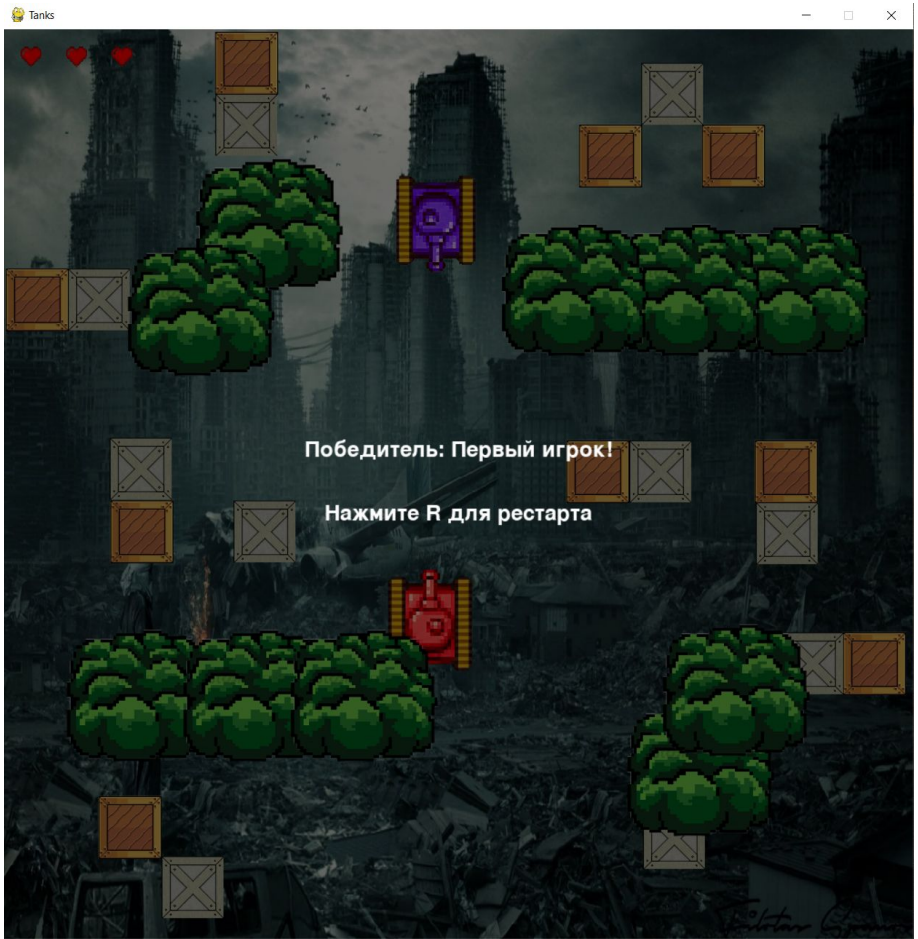
# Процесс игры: возможность разрушать объекты и прятаться в кустах



# Процесс игры: при попадании во вражеский танк отнимаются жизни



# При уничтожении танка противника, появляется данное сообщение



# Структура кода

# 1. Константы и глобальные переменные

```
3   FPS = 60
4   W, H = 1000, 1000
5   BG = (100, 170, 220)
6   speed = 3
7   bullet_speed = 6
8   HEART_SIZE = 40
9   MAX_LIVES = 3
10  HEART_SPACING = 10
11  waiting_for_choose = True
```

# 2. Класс танков (Tank)

```
15    class Tank:  3 usages
16        def __init__(self, player1, player2):...
42
43        def flip1_player1(self):...
49
50        def flip2_player1(self):...
56
57        def flip3_player1(self):...
63
64        def flip4_player1(self):...
70
71        def flip1_player2(self):...
77
78        def flip2_player2(self):...
84
85        def flip3_player2(self):...
91
92        def flip4_player2(self):...
98
99        def move_1(self, dx=0, dy=0, wood_boxes=None, metal_boxes=None):...
136
137       def move_2(self, dx=0, dy=0, wood_boxes=None, metal_boxes=None):...
174
175       def draw(self, screen):...
179
180       def can_shoot(self):...
182
183       def shoot(self):...
185
186       def get_mask1(self):...
188
189       def get_mask2(self):...
```

# Основные функции класса Tank:

flip-функции - отвечают за поворот танков на 4 стороны

move-функции - отвечают за движение танка, а также за невозможность проехать сквозь блоки

draw(screen) - отрисовка танков на экране

can_shoot() и shoot() - отвечают за выстрелы

функции get_mask() - возвращают маски для двух танков соответственно

А также ***конструктор __init__***, в котором указываются скины по умолчанию и загрузка и масштабирование спрайтов танков

# 3. Класс пулей (Bullet)

```
193    class Bullet:  2 usages
194        def __init__(self, tank_rect, direction, tank_type=1):...
232
233        def is_active(self):...
235
236        def fly(self):...
253
254        def draw(self, screen):...
```

# Функции класса Bullet:

is_active() - возвращает значение флага, который проверяет активность пули

fly() - отвечает за движение пули

draw() - отрисовка пуль

А также ***конструктор __init__***, в котором загружается и масштабируется спрайт пули, а еще меняет свое направление в зависимости от направления танка

# 4. Класс жизней (HeartsDisplay)

```
258  class HeartsDisplay:  2 usages
259      def __init__(self, tank_type="player1"):...
273
274      def draw(self, screen):...
278
279      def lose_life(self):...
282
283      def is_alive(self):...
285
286      def reset(self):...
288
```

# Функции класса HeartsDisplay:

draw() - отрисовка жизней на экране

lose_life() - отвечает за потерю жизней

is_alive() - возвращает кол-во жизней в данный момент; если оно равно нулю, то танк, у которого оно равно нулю, проигрывает

reset() - сбрасывает кол-во жизней у обоих танков

А также ***конструктор __init__***, в который загружается и масштабируется спрайт сердец

# 5. Класс деревянных коробок (WoodBoxes)

```
290  class WoodBoxes:  2 usages
291      def __init__(self):...
312
313      def update_rects_and_masks(self):...
319
320      def get_all_obstacle_rects(self):...
322
323      def get_obstacle_mask(self, obs_rect):...
326
327      def draw_wood_boxes(self, screen):...
330
```

# Функции класса WoodBoxes:

update_rects_and_masks() - обновляет ректы и маски для этих коробок

get_all_obstacle_rects() - возвращает все ректы

get_obstacle_mask() - возвращает все маски

draw_wood_boxes() - отрисовка всех коробок на экране

А также ***конструктор __init__***, где загружается и масштабируется спрайт коробки, а еще определяются координаты для каждой коробки

# 6. Класс металлических коробок (MetalBoxes)

```
332  class MetalBoxes:  2 usages
333      def __init__(self):...
356
357      def update_rects_and_masks(self):...
363
364      def get_all_obstacle_rects(self):...
366
367      def get_obstacle_mask(self, obs_rect):...
370
371      def draw_metal_boxes(self, screen):...
374
```

# Функции класса MetalBoxes:

update_rects_and_masks() - обновляет ректы и маски для этих коробок

get_all_obstacle_rects() - возвращает все ректы

get_obstacle_mask() - возвращает все маски

draw_metal_boxes() - отрисовка всех коробок на экране

А также ***конструктор __init__***, где загружается и масштабируется спрайт коробки, а еще определяются координаты для каждой коробки

# 7. Класс кустов (Tree)

```
376    class Tree:  # класс для куста. позволяет танку стать невидимым  1 usage
377        def __init__(self):...
394
395        def draw_trees(self, screen):...
398
```

# Функции класса Tree:

draw_trees() - отрисовка всех кустов

А также ***конструктор __init__***, в котором загружается и масштабируется спрайт куста, а еще определяются его координаты

# 8. Вспомогательные функции

```
399  def collisions_bullets_with_tanks(tank, bullets_player1, bullets_player2, player1_hearts, player2_hearts):...
425
426
427  def collisions_bullets_with_blocks(bullets_player1, bullets_player2, wood_boxes, metal_boxes):...
497
498
499  def show_start_screen():...
536
537
538  def show_screen_for_choose_skins():...
695
696
697  def reset_game():...
713
```

collisions_bullets_with_tanks() - обработка столкновений пуль с танками

collisions_bullet_with_blocks() - обработка столкновений пуль с блоками

show_screen_for_choose_skins() - показ экрана для выбора скинов

show_start_screen() - показ предстартового экрана с инструкцией по управлению

reset_game() - отвечает за перезапуск игры (при выигрыше одного из игроков)

# 9. Главный игровой цикл:

Его можно разбить на 3 части:

1) **Обработка управления**

2) **Отрисовка объектов**

3) **Отображение экрана конца игры** (и отображение победителя)

# 1) Обработка движения:

```
764     if not game_over:
765         keys = pg.key.get_pressed()
766
767         # прописываем движение и выстрел для 1-го игрока
768         if keys[pg.K_a]:...
771         elif keys[pg.K_d]:...
774         elif keys[pg.K_w]:...
777         elif keys[pg.K_s]:...
780
781         if keys[pg.K_SPACE] and tank.can_shoot():...
785
786         # прописываем движение и выстрел для 2-го игрока
787         if keys[pg.K_LEFT]:...
790         elif keys[pg.K_RIGHT]:...
793         elif keys[pg.K_UP]:...
796         elif keys[pg.K_DOWN]:...
799
800         if keys[pg.K_RCTRL] and tank.can_shoot():...
804
```

# 2) Отрисовка объектов

```
813     # отрисовываем все коробки
814 >   for w_box in wood_boxes:...
816 >   for m_box in metal_boxes:...
818
819     # отрисовка сердец и танков
820     player1_hearts.draw(screen)
821     player2_hearts.draw(screen)
822     tank.draw(screen)
823
824     # отрисовываем все пули
825 >   for bullet in bullets_player1:...
828
829 >   for bullet in bullets_player2:...
832
833     # отрисовка кустов
834 >   for tree in trees:...
```

# 3) Отображение конечного экрана и победителя

```
840     if not game_over:
841         if not player1_hearts.is_alive():
842             game_over = True
843             winner = "Второй игрок"
844         elif not player2_hearts.is_alive():
845             game_over = True
846             winner = "Первый игрок"
847 
848     # отображаем экран конца игры
849     if game_over:
850         overlay = pg.Surface( size: (W, H), pg.SRCALPHA)
851         overlay.fill((0, 0, 0, 128))
852         screen.blit(overlay,  dest: (0, 0))
853 
854         # отображаем победителя
855         winner_text = win_font.render( text: f"Победитель: {winner}!",  antialias: True,  color: (255, 255, 255))
856         restart_text = win_font.render( text: "Нажмите R для рестарта",  antialias: True,  color: (255, 255, 255))
857 
858         screen.blit(winner_text,  dest: (W // 2 - winner_text.get_width() // 2, H // 2 - 50))
859         screen.blit(restart_text,  dest: (W // 2 - restart_text.get_width() // 2, H // 2 + 20))
```