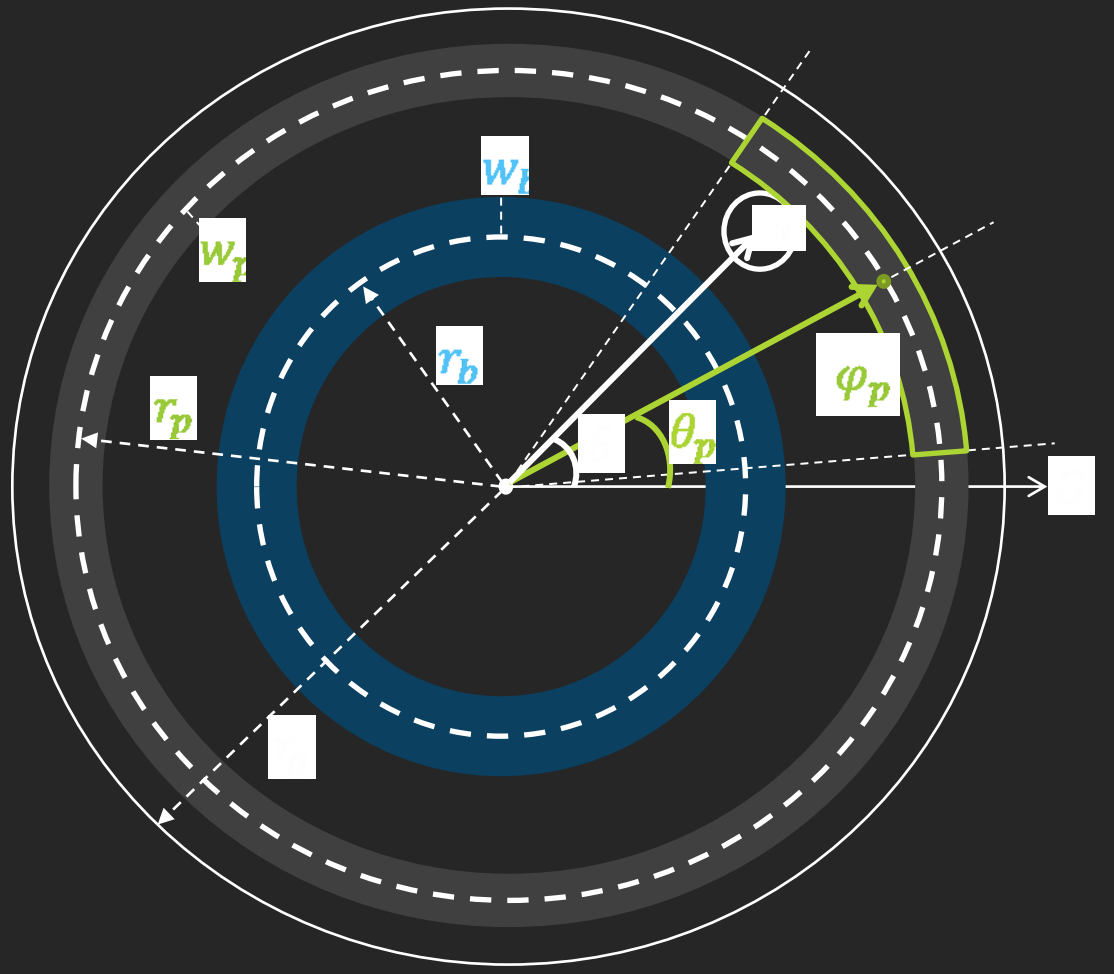
Collision detection and response in the assignment

Marek Trtík
PA199

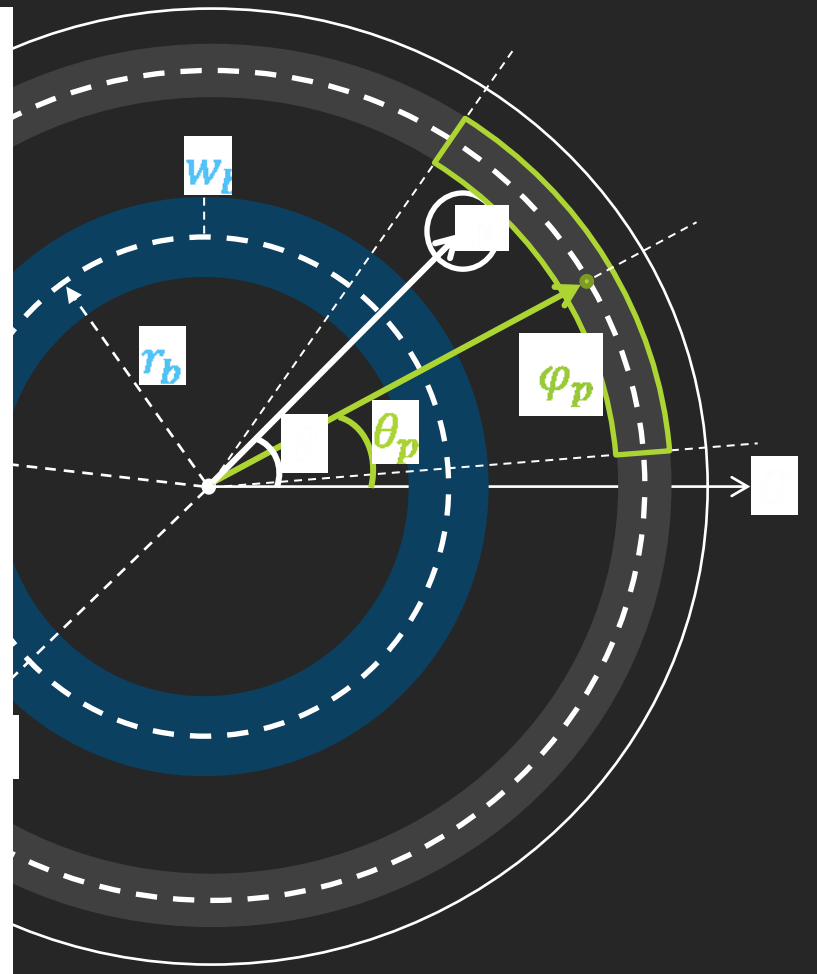
Collision detection: Broad phase

- GAME OVER
- r_p w_p paddles
- r_b w_b bricks
- no collision



Collision detection: Broad phase

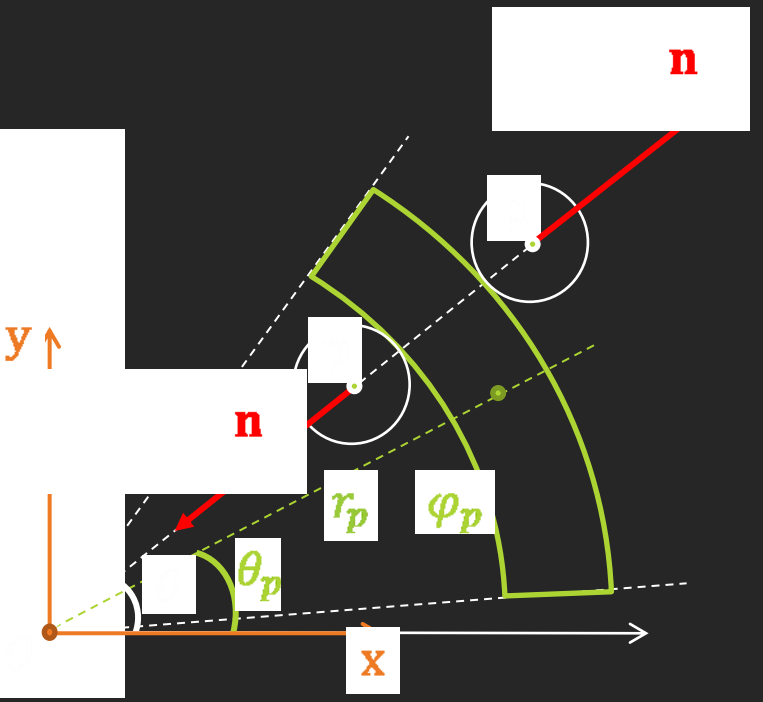
```
# Colliding with Paddles (brick wall case is similar)
def broad_phase(positions, w_p, φ_p):
    r_p, θ_p = positions[0]
    for r'_p, θ'_p in positions[1:]:
        if min_difference(θ'_p, θ_p) < min_difference(θ_p, θ_p):
            r_p, θ_p = r'_p, θ'_p
    if min_difference(θ_p, θ_p) <= φ_p:
        return narrow_phase_case_1(r_p, r_p)
    else:
        return narrow_phase_case_2(r_p, θ_p, w_p, φ_p)
```



Collision detection: Narrow phase

- Case 1: "min_difference(θ_p) < φ_p"

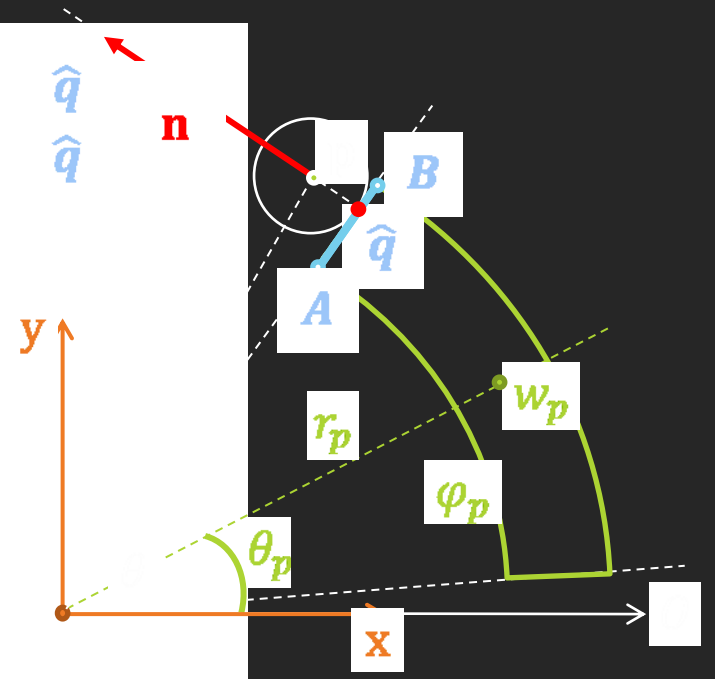
```
def narrow_phase_case_1(p, r_p):
    n = p / |p|
    return n if |p| < r_p else n
```



Collision detection: Narrow phase

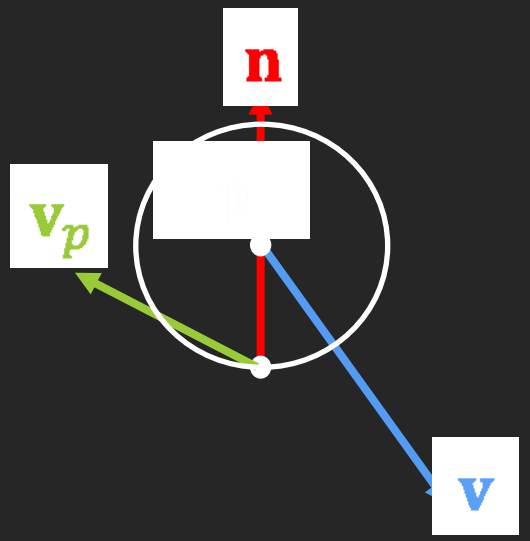
- Case 2: "min_difference(θ_p, φ_p) > φ_p"

```
def narrow_phase_case_2(p, r_p, θ_p, w_p, φ_p):
    sign = 1 if on_left(θ_p, φ_p) else -1
    A = to_cartesian(r_p, w_p, θ_p + sign*φ_p)
    B = to_cartesian(r_p, w_p, θ_p + sign*φ_p)
    q̂ = closest_point_on_line(AB, p)
    return (p - q̂) / |p - q̂| if |p - q̂| ∈ (0, r) else None
```

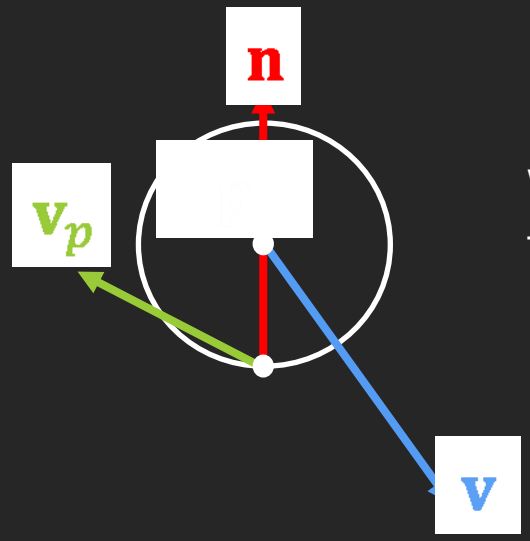


Collision response

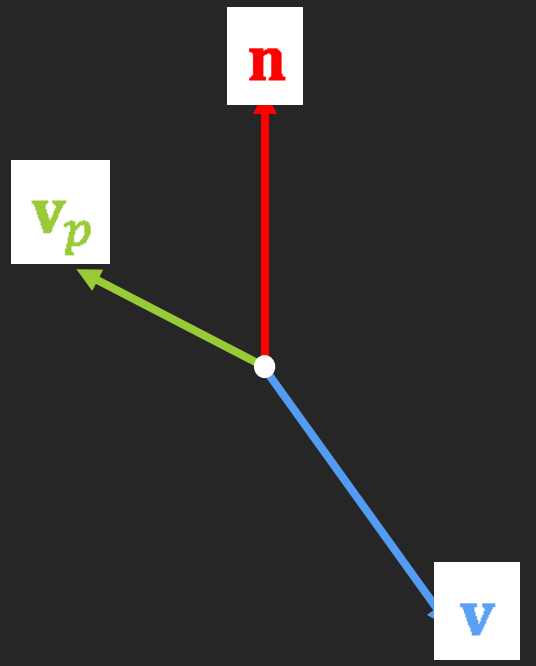
- Ball's velocity: $\mathbf{v} = (v_x\ 0\ v_z)^T$, $|\mathbf{v}| = v_0$ where v_0 is the fixed speed.
- We have the unit collision normal $\mathbf{n} = (n_x\ n_y\ 0)$, $|\mathbf{n}| = 1$ from the collision detection.
- Velocity of a paddle is $\mathbf{v}_p$



Collision response

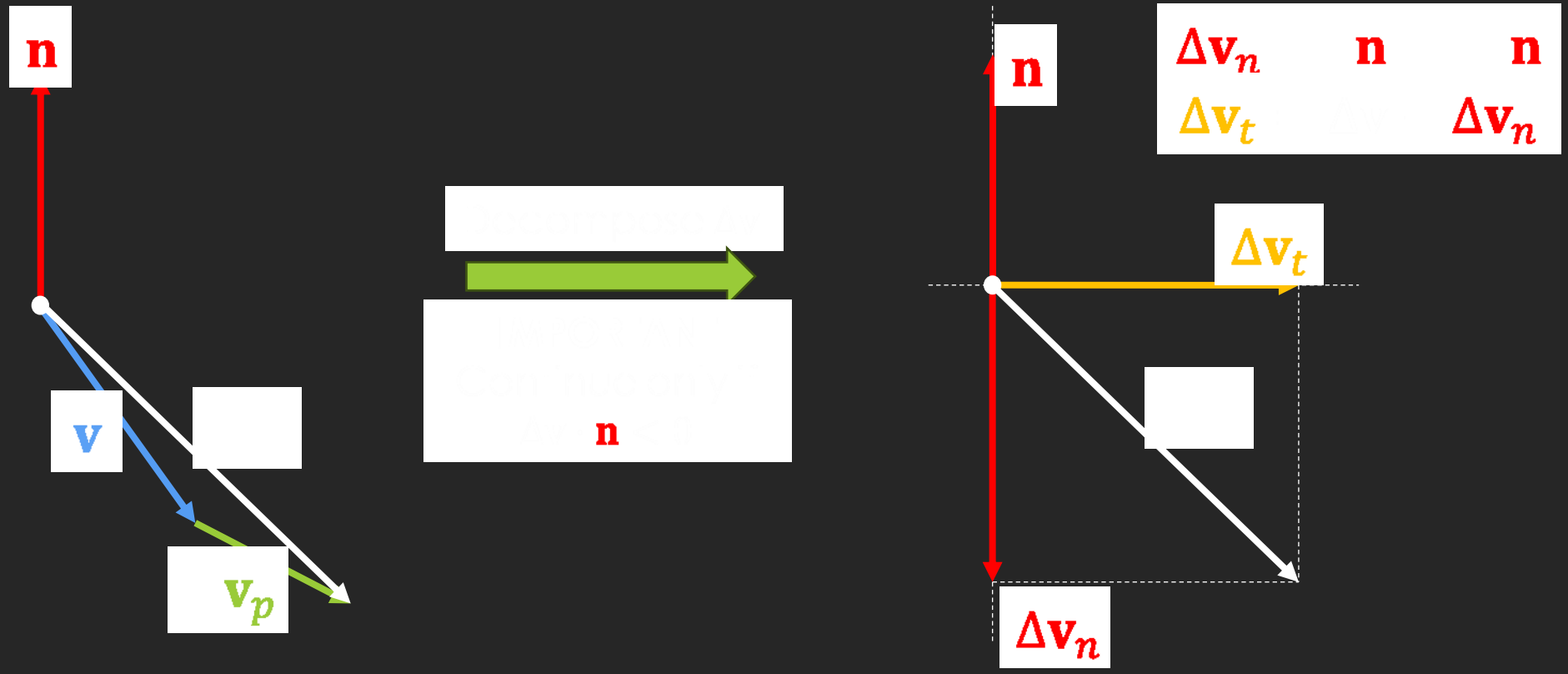


We can model the situation as



Compute the relative velocity

Collision response



Collision response

- "Bounce off the paddle" velocity change:

$$\Delta \mathbf{v}_n' = -\Delta \mathbf{v}_n$$

- "Match paddle's velocity" velocity change:

$$\Delta \mathbf{v}_t' = -\mu_p \min\left(|\Delta \mathbf{v}_n|, |\Delta \mathbf{v}_t|\right) \frac{\Delta \mathbf{v}_t}{|\Delta \mathbf{v}_t|} \quad \text{if } |\Delta \mathbf{v}_t| > 0,$$

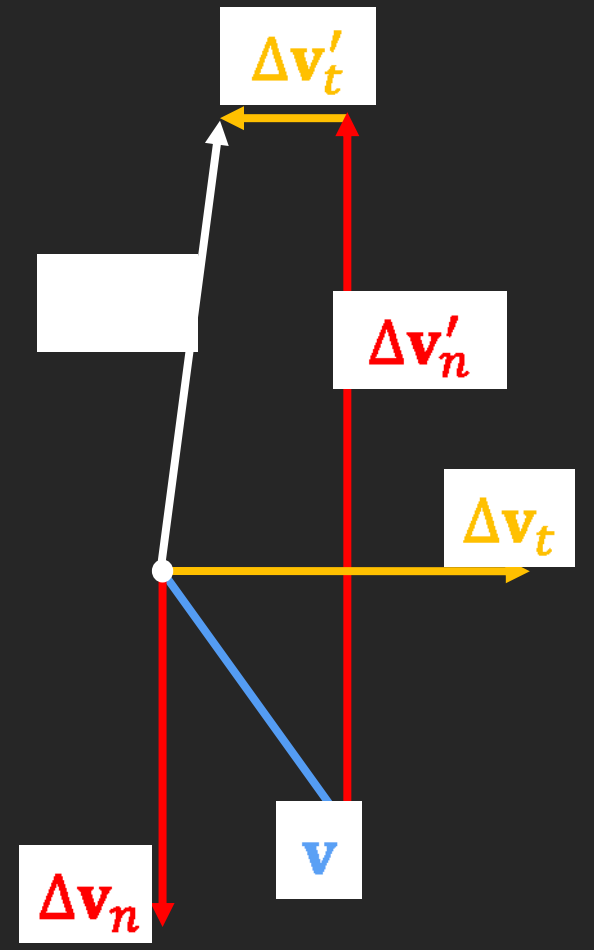
where $0 \leq \mu_p \leq 1$ is a "friction" coefficient.

- So, the collision response velocity is:

$$\mathbf{v}_{res} = \mathbf{v} + \Delta \mathbf{v}_n' + \Delta \mathbf{v}_t'$$

- The final velocity is then:

$$\mathbf{v} = v_0 \frac{\mathbf{v}_{res}}{|\mathbf{v}_{res}|}, \quad \text{NOTE: } |\mathbf{v}_{res}| > 0.$$



Implementation notes

- Polar coordinates:
  - Always normalize the angles to the range $(0, 2\pi)$ before comparison.
  - Consider using normalization directly in:
    - Conversion from the Cartesian to polar coordinates.
    - Operators for addition and subtraction of angles.
      - Alternatively, in comparison operators.
  - Otherwise, assert angles are normalized before comparisons.
  - When implementing angle comparison algorithm, keep in mind the case of passing the polar axis (in CW or CCW direction).

Implementation notes

- Recommendations:
  - Build **tests** and **test scenes** for collision detection and response algorithms.

    => Do **not** build the complete scene of the game (all paddles all wall bricks).

    => Test function “closest_point_on_line” is different situations (configurations of line’s points and the reference point).

    => Test all phases of the collision detection in separate test scenes.

    => Test collison response in separate test scenes (for different velocities of the ball and the paddle).