

```
function bes = beschleunigung (spiel , farbe )
```

```
function initialisieren
    % Wenn meine Farbe rot ist ,
    if strcmp (farbe , 'rot')
        % dann bin ich der rote Spaceball
        ich = spiel .rot;
        % und mein Gegner ist blau .
        gegner = spiel .blau ;
        % Wenn meine Farbe nicht rot ist ,
    else
        % dann bin ich der blaue Spaceball
        ich = spiel .blau ;
        % und mein Gegner ist rot
        gegner = spiel .rot ;
    end

    bes = [0, 0];
    bande_aus = false;
end
```

```
initialisieren
```

```
%%%%%%%%%%%%%
%%%%%%%%%%% wichtige Berechnungen %%%%%%
%%%%%%%%%%%%%
```

```
%Betrag meiner Geschwindigkeit
norm_ichges = norm (ich.ges);

%Bremsweg
bremsweg = norm_ichges^2 / (2*spiel.bes);

%Radius des Spaceballs plus Sicherheit
ich_radius_safe = spiel.spaceball_radius + 0.002;
minen_radius_safe = spiel.mine_radius + ich_radius_safe + 0.003 ;
```

```
%%%%%%%%%%%%%
%%%%%%%%%%% Hilfsfunktionen %%%%%%
%%%%%%%%%%%%%
```

```
%Projektion a auf b
function a_auf_b = projektion (a, b)
    a_mal_b = a*b.';
    betrag_b = norm (b);
    a_auf_b = a_mal_b/(betrag_b)^2*b;
end

%Kosinus zwischen zwei Vektoren
function cos_a_b = Kosinus (a, b)
    cos_a_b = (a*b.')/(norm(a)*norm(b));
end

%senkrechter Vektor
function senk_zu_a = senkrecht(a)
    senk_zu_a = [-a(2), a(1)];
end

% Schnitt von zwei bewegten Kreisen

function mittelpunkte = schnitt_zwei_kreise(m1, ges1, r1, m2, ges2, r2)

    % Berechnet nach Buchholz Doku S. 73f.
    % Inputs: Mittelpunkte beider Kreise zu Beginn, Radien beider Kreise,
    % Geschwindigkeiten beider Kreise.
    % Output: beide möglichen Zeitvariablen, zu denen eine Berührung
    % stattfindet

    e = m2 - m1;
    f = ges2 - ges1;
    r12 = r1 + r2;
    al = f*f.';
    be = e*f.';
    ga = (e*e.')-r12^2;
    de = be^2 - al*ga;
    if de > 0
        la12 = [(-be+sqrt(de))/al; (-be-sqrt(de))/al];
        mittelpunkte = sortrows(la12);
    else
        la12 = [0; 0];
        mittelpunkte = la12;
    end
end
```

```

%Tangentenpunkte von Punkt an Kreis
function [tp1, tp2] = tangentenpunkte(a, b, r)

    % a ist Ortsvektor des Punktes, b das Mittelpunktes, r der Radius

    d = b-a;                                %Mathematische Hilfsrechnungen
    d1 = d(1);
    d2 = d(2);
    D = d*d.';

    radikand = (r^4*d1^2/D^2) - (r^4-r^2*d2^2)/D; % Komplexes Ergebnis vermeiden
    if radikand >= 0
        v1(1) = r^2*d1/D + sqrt(radikand);      % V-Werte sind Komponenten des Vektors vom Tangentenpunkt nach b
    else
        v1(1) = 0;
    end
    v1_2a = +sqrt(r^2 - v1(1)^2);            % 2 Y-Komponenten, da +/- sqrt(...). Welcher Wert gilt, wird unten bestimmt
    v1_2b = -sqrt(r^2 - v1(1)^2);

    if radikand >= 0
        v2(1) = r^2*d1/D - sqrt(radikand);      % Gleiche Berechnungen für TP2
    else
        v2(1) = 0;
    end
    v2_2a = -sqrt(r^2 - v2(1)^2);
    v2_2b = sqrt(r^2 - v2(1)^2);

    v1a = [v1(1), v1_2a];
    v1b = [v1(1), v1_2b];
    p1a = b - v1a;                          % Aufstellung der Vektoren (jeweils a und b, wegen der Wurzel oben
    p1b = b - v1b;                          % Berechnung des Ortsvektors des TP

    v2a = [v2(1), v2_2a];
    v2b = [v2(1), v2_2b];
    p2a = b - v2a;
    p2b = b - v2b;

    if abs((p1a-a)*(p1a-b).') < 10e-5          % Prüfung, welcher Wert gilt über Skalarprodukt
        tp1 = p1a;
    else
        tp1 = p1b;
    end
    if abs((p2a-a)*(p2a-b).') < 10e-5
        tp2 = p2a;
    else
        tp2 = p2b;
    end
end

% Strecken zum beschleunigen/bremsen auf eine bestimmte geschwindigkeit
% function [d1, d2] = bremsen_auf_zielgeschwindigkeit(v_1, v_2, d)
%     if v_2 > v_1
%         d1 = d/2 + sqrt((d/2)^2 - 1/4 * (((v_2-v_1)^2)/(2*0.1) - d)^2);    %d1: längre der strecke, auf der beschleunigt wird
%     else
%         d1 = d/2 - sqrt((d/2)^2 - 1/4 * (((v_2-v_1)^2)/(2*0.1) - d)^2);
%     end
%     d2 = d - d1;                      % d2: längre der strecke, auf der gebremst wird
% end

function pak = punkt_auf_kreis(p, v_k, p_1)
    r = v_k^2 / 0.1;
    r = norm(ich.ges)^2 / 0.1;
    m = p_1 + projektion(p - p_1, senkrecht(ich.ges)) * 1/norm(projektion(p-p_1, senkrecht(ich.ges))) * r;
    m = ich.pos + projektion(p-ich.pos, senkrecht(ich.ges)) * 1/0.1 * r;
    d = norm(p - m);
    if norm(d-r) < (spiel.spaceball_radius + spiel.tanke_radius)
        pak = 'auf_kreis';
    elseif d < r
        pak = 'innerhalb';
    else
        pak = 'ausserhalb';
    end
end

function pak = punkt_auf_kreis_2(p)
    r = v_k^2 / 0.1;
    r = norm(ich.ges)^2 / 0.1;
    m = p_1 + projektion(p - p_1, senkrecht(ich.ges)) * 1/abs(projektion(p-p_1, senkrecht(ich.ges))) * r;
    m = ich.pos + projektion(p-ich.pos, senkrecht(ich.ges)) * 1/norm(projektion(p-ich.pos, senkrecht(ich.ges))) * r;
    d = norm(p - m);
    if norm(d-r) < (spiel.spaceball_radius + spiel.tanke_radius)
        pak = 'auf_kreis';
    elseif d < r
        pak = 'innerhalb';
    else
        pak = 'ausserhalb';
    end
end

%%%%%%% persistente Variablen %%%%%%

```

```

persistent tankenzahl best best_2 best_3
persistent neuberechnung neuberechnet herabstufen herabstufen_2
persistent zielpunkt zielpunkt_2 T n verteidigungsmodus
persistent tankmodus angriffsmodus kol_1 kol_2
persistent zwei drei lenker bremsen fuenffuenf
persistent ich_zu_tanke_2 ich_zu_tanke_3 tanke_zu_tanke_2
persistent gegner_bes_alt
persistent ich_zu_gegner cos_gegner_ich bande_angriff

if spiel.i_t == 5
    random_zielpunkt = [rand(1), rand(1)];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Strategie %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

%Tanken wenn tanken da.
if gegner.getankt >= 5
    if ich.getankt == 5 % extremer sonderfall: beide 5 tanken
        bes = -ich.ges;
        fuenffuenf = true;
    else
        fuenffuenf = false;
        bes = verteidigung;
        verteidigungsmodus = true;
        angriffsmodus = false;
        tankmodus = false;
    end
elseif spiel.n_tanke > 0
    bes = tanken;
    tankmodus = true;
    angriffsmodus = false;
    verteidigungsmodus = false;
    fuenffuenf = false;
else
    bes = angriff;
    angriffsmodus = true;
    tankmodus = false;
    verteidigungsmodus = false;
    fuenffuenf = false;
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% TANKEN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function tnk = tanken

```

```

if spiel.i_t < 3
    neuberechnung = true;
    neuberechnet = false;
    tankenzahl = spiel.n_tanke + 1;
end
% mine_aktiv
% mine_aktiv_vorher
%neuberechnung, wenn Gegner Tanke vor mir erreicht
if spiel.i_t > 3
    T_h = T(:, 6); %Vektor mit Herabstufungsparameter heraussuchen
    t_nh = find(~T_h);

    gegner_zu_best = zielpunkt-gegner.pos; %Abstand des Gegners zur besten Tanke
    cos_best_gegner = Kosinus(gegner.ges, gegner_zu_best); %Kosinus des Gegners zur besten Tanke
    ich_zu_best = zielpunkt-ich.pos; %Abstand von mir zur besten Tanke
    if (cos_best_gegner > 0.95 && norm(gegner_zu_best) < 1.2 * norm(ich_zu_best) && norm(gegner_zu_best) < 0.5) || ...
        norm(zielpunkt-gegner.pos) < 0.1 && cos_best_gegner > 0.8 && ich.getankt < 5) && ~isempty(t_nh)
        %disp aktiv
        D_g = zeros (spiel.n_mine, 1);
        v_gegner_best = norm(projektion(gegner.ges, gegner_zu_best));
        v_ich_best = norm(projektion(ich.ges, ich_zu_best));
        for x_4 = 1:spiel.n_mine
            %suche nach Minen zwischen Gegner und (zweit)bester Tanke
            gegner_zu_mine = spiel.mine(x_4).pos-gegner.pos; %Vektor von Gegner zur Mine
            cos_mine_gegner = Kosinus(gegner_zu_mine, gegner_zu_best); %Kosinus zwischen dem Weg zur Tanke und zur Mine
            if cos_mine_gegner > 0.5 && norm(gegner_zu_mine) < norm(gegner_zu_best)
                d_g = norm(projektion(gegner_zu_mine, senkrecht(gegner_zu_best))); %Abstand des Minenmittelpunktes vom weg zur Tanke
                D_g(x_4, 1) = d_g;
            else
                D_g(x_4, 1) = 1;
            end
        end
        k_g = find(D_g < 0.05, 1); %Suche nach Abstand kleiner als der Minenradius
        t_g = -10*v_gegner_best+sqrt(100*v_gegner_best^2+20*norm(gegner_zu_best));
        t_i = -10*v_ich_best+sqrt(100*v_ich_best^2+20*norm(ich_zu_best));
        if t_g < t_i && isempty(k_g) %wenn Gegner schneller als ich bei Tanke und keine Mine im Weg Tanke herabstufen
            herabstufen = true;
            %disp('gegner vor mir')
        else
            herabstufen = false;
        end
    else
        herabstufen = false;
    end
else
    herabstufen = false;
end
T_h = T(:, 6); %Vektor mit Herabstufungsparameter heraussuchen

```

```

t_nh = find(~T_h);      %nicht herabgestufte Tanken suchen
if length(t_nh) > 1 && spiel.n_tanke > 2    %wenn mehr als eine nicht herabgestuft wird
    gegner_zu_best_2 = zielpunkt_2-gegner.pos;           %Abstand des Gegners zur zweitbesten Tanke
    cos_best_2_gegner = Kosinus(gegner.ges, gegner_zu_best_2); %Kosinus des Gegners zu zweitbesten Tanke
    ich_zu_best_2 = zielpunkt_2 -ich.pos;                 %Abstand von mir zur zweitbesten Tanke
    if cos_best_2_gegner > 0.95 || ...%&& norm(gegner_zu_best_2) < norm(ich_zu_best_2)
        norm(zielpunkt_2-gegner.pos) < 0.1 && cos_best_2_gegner > 0.8 && ich.getankt < 5
        D_g_2 = zeros(spiel.n_mine, 1);
        v_gegner_best_2 = norm(projektion(gegner.ges, gegner_zu_best_2));
        v_ich_best_2 = norm(projektion(ich.ges, ich_zu_best_2));
        for x_4 = 1:spiel.n_mine                         %suche nach Minen zwischen Gegner und (zweit)besten Tanke
            gegner_zu_mine = spiel.mine(x_4).pos-gegner.pos;   %Vektor von Gegner zur Mine
            cos_mine_gegner_2 = Kosinus(gegner_zu_mine, gegner_zu_best_2);
            if cos_mine_gegner_2 > 0.5 && norm(gegner_zu_mine) < norm(gegner_zu_best_2)
                d_g_2 = norm(projektion(gegner_zu_mine, senkrecht(gegner_zu_best_2)));
                D_g_2(x_4, 1) = d_g_2;
            else
                D_g_2(x_4, 1) = 1;
            end
        end
        k_g_2 = find(D_g_2 < 0.05, 1); %Suche nach Abstand kleiner als der Minenradius
        t_g_2 = -10*v_gegner_best_2+sqrt(100*v_gegner_best_2^2+20*norm(gegner_zu_best_2));
        t_i_2 = -10*v_ich_best_2+sqrt(100*v_ich_best_2^2+20*norm(ich_zu_best_2));
        if t_g_2 < t_i_2 && isempty(k_g_2) && norm(gegner_zu_best_2) < 0.5
            herabstufen_2 = true;
        else
            herabstufen_2 = false;
        end
    else
        herabstufen_2 = false;
    end
end
% herabstufen
% herabstufen_2

% festlegen ob neuberechnet wird:

if tankenzahl ~= spiel.n_tanke || spiel.i_t > 3 && ...
    norm(zielpunkt - ich.pos) < 1.1*bremsweg && ~neuberechnet ||...
    spiel.i_t > 3 && herabstufen_2 && spiel.n_tanke > 1 ||...
    spiel.i_t > 3 && herabstufen_2 && spiel.n_tanke > 2 %||...
%    spiel.i_t > 3 && mine_aktiv_vorher && ~mine_aktiv
    neuberechnung=true;
    neuberechnet = true;
elseif spiel.i_t < 3 %am Anfang neuberechnen
    neuberechnung = true;
else
    neuberechnung=false;
end

if spiel.i_t > 3 && norm(zielpunkt - ich.pos) < 1.1*bremsweg || spiel.n_tanke == 1 %wenn Bedigung dauerhaft erfüllt nur 1x neuberechnen
    neuberechnet = true; %hierzu neuberechnet auf true setzen
else
    neuberechnet = false;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Tankenmatrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if neuberechnung
    T = zeros (spiel.n_tanke, 9);
    %disp('neuberechnet')

    for x = 1: spiel.n_tanke
        ich_zu_tanke = (spiel.tanke(x).pos-ich.pos);
        gegner_zu_tanke = (spiel.tanke(x).pos - gegner.pos);
        T(x, 1) = x;                                %Index der Tanke
        T(x, 2) = -0.7*norm(ich_zu_tanke)+1;          %Parameter der eigenen Distanz zur Tanke
        T(x, 3) = Kosinus(ich.ges, (ich_zu_tanke)); %=Parameter des eigenen Winkels zur Tanke
        T(x, 7) = -0.7*norm(gegner_zu_tanke)+1;        %Parameter der Distanz des Gegners zur Tanke
        if spiel.n_mine > 0
            D = zeros (spiel.n_mine, 1);
            for x_1 = 1:spiel.n_mine
                ich_zu_mine = spiel.mine(x_1).pos-ich.pos; % Vektor von mir zur Mine
                cos_mine = Kosinus(ich_zu_mine, ich_zu_tanke); %Kosinus zwischen dem Weg zur Tanke und zur Mine
                if cos_mine > 0.5 && norm(ich_zu_mine) < norm(ich_zu_tanke) %wenn Mine vor mir und näher als Tanke
                    d = norm(projektion(ich_zu_mine, senkrecht(ich_zu_tanke))); %Abstand des Minenmittelpunktes zu Weg zur Tanke
                    D(x_1, 1) = d; %Matrix mit den Abständen
                else
                    D(x_1, 1) = 1; %Wenn keine Mine auf dem Weg Abstand auf 1 setzen
                end
            end
            k = find(D <0.07); %Suche nach Abständen kleiner als der Minenradius -> Mine geschnitten

            if length(k) > 1 %Wenn mehr als eine Mine geschnitten
                D_geschnitten = zeros(length(k), 3);
                for x_2 = 1:length(k)
                    D_geschnitten(x_2, 1) = k(x_2); %Index der Mine
                    D_geschnitten(x_2, 2) = D(k(x_2)); %Abstand Ihres Mittelpunktes zu meinem Weg
                    D_geschnitten(x_2, 3) = norm(spiel.mine(k(x_2)).pos - ich.pos); %Abstand von mir zur Mine
                end
                D_sort = sortrows(D_geschnitten, 2); %nach Abstand sortieren
                T(x, 4) = 14.2*D_sort(1, 2); %nächste Mine als Kriterium zur Tanken Bewertung nehmen
            end
        end
    end
end

```

```

elseif k > 0 %wenn nur eine Mine auf dem Weg diese als Kriterium verwenden
    T(x, 4) = 14.2*D(k);
    %%disp(D)
    %%disp('mine im weg')
else
    T(x, 4) = 1; %wenn keine Mine geschnitten wird Bewertungsparameter auf 1 setzen
end

if spiel.i_t > 3 && herabstufen || spiel.i_t > 3 && herabstufen_2
    cos_gegner_tanke = Kosinus(gegner.ges, gegner_zu_tanke);
    v_gegner_tanke = norm(projektion(gegner.ges, gegner_zu_tanke));
    v_ich_tanke = norm(projektion(ich.ges, ich_zu_tanke));
    if cos_gegner_tanke > 0.95 || norm(gegner_zu_tanke) < 0.1 && cos_gegner_tanke > 0.8
        t_g = -10*v_gegner_tanke+sqrt(100*v_gegner_tanke^2+20*norm(gegner_zu_tanke));
        t_i = -10*v_ich_tanke+sqrt(100*v_ich_tanke^2+20*norm(ich_zu_tanke));
        if t_g < t_i & (spiel.tanke(x).pos == zielpunkt) & norm(gegner_zu_tanke) < 0.5
            T(x, 6) = 1.5;
            %disp gegnervormir
        elseif t_g < t_i && norm(gegner_zu_tanke) < 0.5
            T(x, 6) = 1;
            %disp('gegner vor mir')
        else
            T(x, 6) = 0;
        end
    end
else
    T(x, 6) = 0;
end

T_dist = zeros(spiel.n_tanke, 2);

if T(x, 6) == 0
    for x_3 = 1:spiel.n_tanke
        if x_3 ~= x && T(x_3, 6) == 0 %für alle Tanken den Abstand zur aktuellen berechnen
            T_dist(x_3, 1) = x_3; %und den dazugehörigen Index in die Matrix eintragen
            T_dist(x_3, 2) = norm(spiel.tanke(x_3).pos - spiel.tanke(x).pos);
        end
    end
end
%%disp(T_dist)
T_clust = find(T_dist > 0 & T_dist < 0.3); %Abstände kleiner 0.3 herausuchen
T(x, 9) = length(T_clust); %Anzahl der nahen Tanken in Matrix einfügen
if T(x, 9) ~= 0
    for x_5 = 1:T(x, 9)
        i = T_clust(x_5)-spiel.n_tanke;
        T(x, 9+x_5) = i; %Indices der Tanken im Cluster in Tankenmatrix eintragen
    end
end
end

%%%%%%%%%%%%% Tankenmatrix Ende %%%%%%%

%Clustersuche
T_clust_sort = sortrows(T, -9); %Tankenmatrix nach anzahl der Tanken pro Cluster sortieren
clust_vec = T_clust_sort(:, 9); %Vektor mit dieser Anzahl herausnehmen
clust_max = T_clust_sort(1, 9); %größtes Cluster herausfinden
gleiche_clust = find(clust_vec==clust_max); %schauen ob es mehrere größte Cluster gibt

if ~isempty(clust_vec) %Wenn es Cluster gibt
    for x_6 = 1:length(gleiche_clust) %für alle größten Cluster
        n_clust = length (T_clust_sort(x_6,:)); %Anzahl der Tanken im Cluster
        i_m = T_clust_sort(x_6, 1); %Index der "mittleren" Clustertanke
        if ich.getankt == 3 & clust_max == 1 %wenn 2 zum Sieg fehlen, stärker bevorzugen
            T(i_m, 8) = 2;
        else
            T(i_m, 8) = 1; %Clusterparameter für diese eintragen
        end
        for x_7 = 10:n_clust %für alle Tanken im Cluster
            i_c = T_clust_sort(x_6, x_7); %Indices herausfinden
            if ich.getankt == 3 & clust_max == 1 %wenn 2 zum Sieg fehlen, stärker bevorzugen
                T(i_c, 8) = 2;
            else
                T(i_c, 8) = 1; %Clusterparameter für diese eintragen
            end
        end
    end
    if clust_max > 1 %wenn mehr als 2 im größten Cluster
        for x_12 = 1:spiel.n_tanke
            if T(x_12, 9) == clust_max-1 && T(x_12, 8) == 0 && T(x_12, 6) == 0 && ...
                spiel.n_tanke > 5
                T(x_12, 8) = 0.5;
            end
        end
    end
end
end

%Gesamtparamter jeder Tanke nach Clustersuche bestimmen
for x_8 = 1:spiel.n_tanke
    T(x_8, 5) = 0.6*T(x_8, 2)+10*norm_ichges^2*T(x_8, 3)+0.1*T(x_8, 4)-T(x_8, 6)-0.1*T(x, 7)+0.1*T(x_8, 8);
end

```

```

%am Anfang nächste Tanke bevorzugen?
if spiel.i_t < 200
    T_nah = sortrows(T, -2);
    T_mine = T(:, 4);
    k_m = find(T_mine > 0.95);
    if T_nah(1, 4) > 0.9 || isempty(k_m)
        T(T_nah(1, 1), 5) = T(T_nah(1, 1), 5)+0.1;
        %disp('aufgewertet')
    end
end

T_sort = sortrows(T, -5); %sortieren nach Gesamtparameter

%Tanken in Reihe mit bester Tanke genauso wie Tanken in Cluster bewerten
if spiel.n_tanke > 2 && ich.getankt > 1
    for x_11 = 2:spiel.n_tanke
        w_r = T_sort(1, 3)-T_sort(x_11, 3);
        if abs(w_r) < 0.01 && T_sort(x_11, 8) == 0 && T_sort(x_11, 4) > 0.9 &&...
            T_sort(x_11, 6) == 0
            i_r = T_sort(x_11, 1);
            T(i_r, 8) = 1;
            T(i_r, 5) = T(i_r, 5) + 0.1*T(i_r, 8);
        end
    end
end

T_sort = sortrows(T, -5);

%mittlere Tanke bevorzugen sinnvoll? (noch in Arbeit)

if spiel.tanke(1).pos(1)==0.5 && norm(spiel.tanke(1).pos-ich.pos) < norm(spiel.tanke(T_sort(1, 1)).pos - ich.pos) && ...
    norm_ichges < 0.15 && T(1, 6) == 0 && Kosinus(ich.ges, (spiel.tanke(1).pos-ich.pos)) > 0.7 && T(1, 9) ~= 0
    T(1, 5) = T(1, 5)+0.05;
    T_sort = sortrows(T, -5);
    %disp mitte
end

%auf dem Weg liegende einsammeln
if spiel.n_tanke > 1
    for x_13 = 2:spiel.n_tanke
        if T_sort(x_13, 2) > 1.05 * T_sort(1, 2) && T_sort(x_13, 3) > 0.9*T_sort(1, 3)
            T_sort(x_13, 5) = T_sort(x_13, 5) + 0.05;
        end
    end
end

T_sort = sortrows(T_sort, -5);

%Tankenauswahl:
best = T_sort(1,1);
best = T_sort(1,1);
if spiel.n_tanke > 2
    best_2 = T_sort(2,1);
    best_3 = T_sort(3,1);
    zielpunkt_2= spiel.tanke(best_2).pos;
elseif spiel.n_tanke > 1
    best_2 = T_sort(2,1);
    zielpunkt_2= spiel.tanke(best_2).pos;
end

%berechnen ob zwei oder drei Tanken in einer Reihe liegen
if spiel.n_tanke > 2
    tanke_zu_tanke_2 = spiel.tanke(best_2).pos-spiel.tanke(best).pos;
    cos_tanke_ich_2 = Kosinus(ich.ges, tanke_zu_tanke_2);
    ich_zu_tanke_2 = spiel.tanke(best_2).pos-ich.pos;

    tanke_2_zu_tanke_3 = (spiel.tanke(best_3).pos-spiel.tanke(best_2).pos);
    cos_tanke_2_tanke_3 = Kosinus(tanke_zu_tanke_2, tanke_2_zu_tanke_3);
    ich_zu_tanke_3 = spiel.tanke(best_3).pos-ich.pos;
elseif spiel.n_tanke > 1
    tanke_zu_tanke_2 = (spiel.tanke(best_2).pos-spiel.tanke(best).pos);
    cos_tanke_ich_2 = Kosinus(ich.ges, tanke_zu_tanke_2);
    ich_zu_tanke_2 = spiel.tanke(best_2).pos-ich.pos;

    ich_zu_tanke_3 = 1;
    cos_tanke_2_tanke_3 = 0;
else
    cos_tanke_ich_2 = 0;
    ich_zu_tanke_2 = 1;

    ich_zu_tanke_3 = 1;
    cos_tanke_2_tanke_3 = 0;
end

if spiel.i_t > 1 && cos_tanke_ich_2 > 0.97 && cos_tanke_2_tanke_3 > 0.97 && ...
    T(best_2, 4) > 0.9 && T(best_3, 4) > 0.85
    drei = true;
    %disp('drei')
elseif cos_tanke_ich_2 > 0.97 && T(best_2, 4) > 0.9
    zwei = true;
    drei = false;
    %disp('zwei')
else
    zwei = false;
    drei = false;
end

```

```

if spiel.i_t==1
    n=0;
end

n=n+1;
% %
% disp(n)
% disp(T_sort)

zielpunkt = spiel.tanke(best).pos;
if spiel.n_tanke > 1
    zielpunkt_2 = spiel.tanke(best_2).pos;
end
end
%%%%%%%%%%%%% Ende der Neuberechnung %%%%%%%%%%%%%%

```

```

tnk = tanke_anfliegen;

function tnk = tanke_anfliegen
%Kosinus zur nächsten Tanke
cos_tanke_ich = Kosinus((spiel.tanke(best).pos-ich.pos),ich.ges);

%auf tanke zufliegen
ich_zu_tanke = spiel.tanke(best).pos - ich.pos;
ich_zu_tanke_senkrecht = senkrecht(ich_zu_tanke);
ges_senkrecht = senkrecht(ich.ges);
gegner_zu_best = spiel.tanke(best).pos - gegner.pos;
cos_gegner_best = Kosinus(gegner_zu_best, gegner.ges);

%wenn Winkel zu tanke groß, lenken
if norm(ich.ges) > 0.03 && 0 < cos_tanke_ich
    a = (sqrt(1-cos_tanke_ich^2)/cos_tanke_ich) * norm(ich_zu_tanke); %Gegenkathete berechnen
    if a > 0.015
        lenken = true;
    else
        lenken = false;
    end
else
    lenken = false;
end

if spiel.n_tanke > 2
    ich_zu_tanke_2 = spiel.tanke(best_2).pos-ich.pos;
    ich_zu_tanke_3 = spiel.tanke(best_3).pos-ich.pos;
    cos_tanke_ich_2 = Kosinus(ich.ges, ich_zu_tanke_2);
    %cos_tanke_tanke_2 = Kosinus(ich_zu_tanke, tanke_zu_tanke_2);
elseif spiel.n_tanke > 1
    ich_zu_tanke_2 = spiel.tanke(best_2).pos-ich.pos;
    cos_tanke_ich_2 = Kosinus(ich.ges, ich_zu_tanke_2);
    %cos_tanke_tanke_2 = Kosinus(ich_zu_tanke, tanke_zu_tanke_2);
end

%alternative Bremswegberechnung(test)

b = (sqrt(1-cos_gegner_best^2)/cos_gegner_best) * norm(gegner_zu_best);
v_0 = norm_ichges;
a = -spiel.bes;
if spiel.n_tanke > 1 && cos_tanke_ich_2 > 0 && norm(ich_zu_tanke) < norm(ich_zu_tanke_2)
    tt = spiel.tanke(best_2).pos - spiel.tanke(best).pos;
    tt_norm = norm(tt);
    t_senkrecht = norm(projektion(tt, senkrecht(ich.ges)));
    verhaeltnis = t_senkrecht/tt_norm;
    v = -0.3*verhaeltnis + 0.3;
elseif spiel.n_tanke == 1 && ich.getankt == 4 && (Kosinus(gegner.ges, gegner_zu_best) > 0.95 || abs(b) < 0.02)
    v = v_0;
else
    v = 0;
end

t_g = -10*norm(gegner.ges)+sqrt(100*norm(gegner.ges)^2+20*norm(gegner_zu_best));
t_i = -10*norm_ichges+sqrt(100*norm_ichges^2+20*norm(ich_zu_tanke));

%Gas geben wenn wir auf die selbe Tanke wie der Gegner zufliegen
if abs(b) < 0.02 && t_g >= t_i && 0.9*norm(ich_zu_tanke) < norm(gegner_zu_best) && norm(gegner_zu_best) < 1.1*norm(ich_zu_tanke)
    v = v_0;
    %disp vollgas
end

if v > 0 && cos_tanke_ich > 0.9 && spiel.n_tanke > 1
    kreis = punkt_auf_kreis(spiel.tanke(best_2).pos, v, spiel.tanke(best).pos);
    if strcmp(kreis, 'innerhalb') || strcmp(kreis, 'auf_kreis')
        v_t = v;
        while strcmp(kreis, 'innerhalb') || strcmp(kreis, 'auf_kreis')
            v_t = v_t-0.01;
            kreis = punkt_auf_kreis(spiel.tanke(best_2).pos, v_t, spiel.tanke(best).pos);
        end
        v = v_t;
    end
end

v_mine = mine_nach_tanke(zielpunkt, 0.5);

if v > v_mine
    v = v_mine;

```

```

end

kein_kreis = false;

if norm_ichges > 0 && cos_tanke_ich > 0
    kreis_2 = punkt_auf_kreis_2(spiel.tanke(best).pos);
    if strcmp(kreis_2, 'innerhalb') || strcmp(kreis_2, 'auf_kreis')
        %disp bremsen
        kein_kreis = true;
    end
end

if spiel.n_tanke > 0 &&spiel.i_t > 1
    s_b = (v^2-v_0^2)/(2*a);
end

%Bremsen wenn:
if spiel.i_t > 3 && norm(ich_zu_tanke) < s_b ||... %Abstand zu Tanke < sb
    spiel.i_t > 3 && ~drei && zwei && norm(ich_zu_tanke_2) < bremsweg ||... %eine weitere Tanke hintendran & Abstand zu dieser < bw
    spiel.i_t > 3 && drei && norm(ich_zu_tanke_3) < bremsweg |||... %zwei weitere Tanken hinter angeflogener & Abstand zu dritter < bw
    %spiel.i_t > 3 && norm_ichges > 0.02 && abs(cos_tanke_ich)< 0.2 |||... %um Kreisen um Tanke zu vermeiden
    %spiel.i_t > 3 && norm_ichges > 0.25 %wenn die Geschwindigkeit zu groß wird
    bremsen = true;
    %disp('bremsen Tanke')
else
    bremsen = false;
    %disp('nicht bremsen')
end

%Beschleunigung festlegen
if ~lenken && ~bremsen
    tnk = ich_zu_tanke;
    %disp('draufzufliegen')
elseif lenken && ~bremsen
    if norm_ichges > 0.05 && cos_tanke_ich < 0.5 || kein_kreis
        tnk = -projektion(ich.ges, ich_zu_tanke_senkrecht);
        %disp alt
    else
        tnk = projektion(ich_zu_tanke, ges_senkrecht);
    end
    %disp('lenken')
elseif ~lenken && bremsen
    tnk = -ich.ges;
    %disp('bremsen')
else
    tnk = - projektion(ich.ges, ich_zu_tanke_senkrecht) - 0.1*ich.ges;
    %disp('beides')
end
end

```

```
end
```

```
tankenzahl = spiel.n_tanke;
```

```
%%%%%%%%%%%%%%%
%%%%% Angriff %%%%%%
%%%%%%%%%%%%%%%

function ang = angriff

    s_g = gegner.pos + gegner.ges - ich.pos;
    s_g_norm = norm(s_g); %Strecke bis zum Gegner
    v_i_norm = norm(projektion(ich.ges, s_g));
    v_g = gegner.ges;
    t_g = -10*v_i_norm+sqrt(100*v_i_norm^2+20*s_g_norm); %Zeit bis Gegner berechnen
    gegner_pos_neu = 0.5*gegner.bes*t_g^2+v_g*t_g+gegner.pos;

    % s_g = norm(gegner_pos_neu - ich.pos); %strecke neu berechnen für genauere Zeit
    % t_g = -10*v_i_norm+sqrt(100*v_i_norm^2+20*s_g); %neue Zeit bis Gegner berechnen
    % gegner_pos_neu = 0.5*gegner.bes*t_g^2+v_g*t_g+gegner.pos;

    if ~isempty(gegner.bes_alt) && Kosinus(gegner.bes, gegner.bes_alt) < 0.5
        %disp flackern
        gegner_pos_neu = v_g*t_g + gegner.pos;
    end

    bande_angriff = false;
    %etwas weiter weg von der Bande parallel zum Gegner fliegen um bei
    %Bandenwechsel schneller da zu sein wenn Gegner parallel zur Bande
    %verteidigt
    if norm(gegner.pos-ich.pos) > 0.1
        if abs(gegner.ges(2)) < 0.03 && abs(gegner.ges(1)) > 0.1 && gegner.pos(2) < 0.15
            gegner_pos_neu = [gegner_pos_neu(1), (gegner_pos_neu(2) + 0.06)]; %unten
            bande_angriff =true;
        elseif abs(gegner.ges(2)) < 0.03 && abs(gegner.ges(1)) > 0.1 && gegner.pos(2) > 0.85
            gegner_pos_neu = [gegner_pos_neu(1), (gegner_pos_neu(2) - 0.06)]; %oben
            bande_angriff =true;
        elseif abs(gegner.ges(1)) < 0.03 && abs(gegner.ges(2)) > 0.1 && gegner.pos(1) < 0.15
            gegner_pos_neu = [(gegner_pos_neu(1) + 0.06), gegner_pos_neu(2)]; %links
        end
    end

```

```

bande_angriff =true;
elseif abs(gegner.ges(1)) < 0.03 && abs(gegner.ges(2)) > 0.1 && gegner.pos(1) > 0.85
    gegner_pos_neu = [(gegner_pos_neu(1) - 0.06), gegner_pos_neu(2)];      %rechts
    bande_angriff =true;
end
end

%liegt der Zielpunkt außerhalb des Spielfelds wird er nach innen
%versetzt
if gegner_pos_neu(1)>1 %rechte Bande
    gegner_pos_neu(1) = 1 - 1.5*spiel.spaceball_radius;
end
if gegner_pos_neu(1)<0 %linke Bande
    gegner_pos_neu(1) = 1.5*spiel.spaceball_radius;
end
if gegner_pos_neu(2)>1 %obere Bande
    gegner_pos_neu(2) = 1 - 1.5*spiel.spaceball_radius;
end
if gegner_pos_neu(2)<0 %untere Bande
    gegner_pos_neu(2) = 1.5*spiel.spaceball_radius;
end

if norm(gegner.ges) > 0.05
    einheit_gegner_ges = gegner.ges/norm(gegner.ges);
    gegner_pos_neu = gegner_pos_neu + einheit_gegner_ges * 0.015;
end

ich_zu_gegner = gegner_pos_neu - ich.pos;
ich_zu_gegner_direkt = gegner.pos-ich.pos;
ich_zu_gegner_senkrecht = senkrecht(ich_zu_gegner);
cos_gegner_ich = Kosinus (ich_zu_gegner, ich.ges);

if Kosinus(ich_zu_gegner, ich_zu_gegner_direkt) < -0.7 %wenn Gegner hinter uns aber zukünftige Position des Gegners vor uns: auf Gegner zufliegen
    ich_zu_gegner = ich_zu_gegner_direkt;
    gegner_pos_neu = gegner.pos;
    %disp neu
end

if 0 < cos_gegner_ich && cos_gegner_ich < 0.999
    if norm_ichges > 0.05 && cos_gegner_ich < 0.5
        ang = -projektion(ich.ges, ich_zu_gegner_senkrecht);
        %disp alt
    else
        ang = projektion(ich_zu_gegner, senkrecht(ich.ges));
    end
else
    ang = ich_zu_gegner;
end

if norm(gegner.ges) < 0.03 && spiel.i_t < 5000 && norm(ich_zu_gegner_direkt) < 0.7 %wenn noch genug Zeit ist: Geschwindigkeit reduzieren wenn Gegner sehr schnell reagiert
    v_max = 0.15;
    if norm_ichges > v_max
        ang = -ich.ges;
    end
end

i_1 = ich.pos(1);
i_2 = ich.pos(2);
v_i_1 = ich.ges(1);
v_i_2 = ich.ges(2);
g_1 = gegner.pos(1);
g_2 = gegner.pos(2);
v_g_1 = gegner.ges(1);
v_g_2 = gegner.ges(2);

beta = (i_2*v_i_1+g_1*v_i_2-i_1*v_i_2-g_2*v_i_1)/(v_g_2*v_i_1-v_g_1*v_i_2);
alpha = (g_1+beta*v_g_1-i_1)/v_i_1;
s_p = ich.ges*alpha+ich.pos;
alpha_2 = -10*norm_ichges+sqrt(100*norm_ichges^2+20*(norm(s_p-ich.pos)));
gegner_pos_s = 0.5*gegner.bes*alpha_2^2+gegner.ges*alpha_2+gegner.pos;
s_d = 0.5*norm(gegner.bes)*alpha_2^2+norm(gegner.ges)*alpha_2;

d_s = norm(s_p-gegner_pos_s);

if d_s < 0.02 && Kosinus(ich.ges, gegner.ges) > 0 && alpha > 0 && ...
    s_d < 0.02 && beta > 0
    %disp direkt
    ang = ich.ges;
    bande_aus = true;
else
    bande_aus = false;
end

zielpunkt = gegner_pos_neu;
gegner_bes_alt = gegner.bes;
end

```

```

%%%%%%%%%%%%% VERTEIDIGUNG %%%%%%
%%%%%%%%%%%%% VERTEIDIGUNG %%%%%%
%%%%%%%%%%%%% VERTEIDIGUNG %%%%%%

function def = verteidigung
notbremse_aus = 0;
def = -(gegner.pos-ich.pos)+0.1*ich.ges/norm_ichges;

ich_radius=spiel.spaceball_radius;

%
ich_links = ich.pos(1) - ich_radius;
ich_rechts = 1 - ich.pos(1) - ich_radius;
ich_oben = 1 - ich.pos(2) - ich_radius;
ich_unten = ich.pos(2) - ich_radius;

gegner_links = gegner.pos(1)-1*ich_radius;
gegner_rechts = 1 - gegner.pos(1) - ich_radius;
gegner_oben = 1 - gegner.pos(2) - ich_radius;
gegner_unten = gegner.pos(2) - ich_radius;

abstand_default = 0.05;
gegner_abstand = 1.5;
norm_ich_zum_gegner = norm(gegner.pos - ich.pos);
if norm_ich_zum_gegner < gegner_abstand
    gegner_nah = true;
else
    gegner_nah = false;
end

if ich_oben < abstand_default
    oben = true;
else
    oben = false;
end
if ich_rechts < abstand_default
    rechts = true;
else
    rechts = false;
end
if ich_unten < abstand_default
    unten = true;
else
    unten = false;
end
if ich_links < abstand_default
    links = true;
else
    links = false;
end

if oben
    if gegner_links >= gegner_rechts
        def = [-1, 0];
        zielpunkt = [0.025, 0.975];
    else
        def = [1, 0];
        zielpunkt = [0.975, 0.975];
    end

elseif rechts
    if gegner_oben >= gegner_unten
        def = [0, 1];
        zielpunkt = [0.975, 0.975];
    else
        def =[0, -1];
        zielpunkt = [0.975, 0.025];
    end

elseif unten
    if gegner_links >= gegner_rechts
        def = [-1, 0];
        zielpunkt = [0.025, 0.025];
    else
        def = [1, 0];
        zielpunkt = [0.975, 0.025];
    end

elseif links
    if gegner_oben >= gegner_unten
        def = [0, 1];
        zielpunkt = [0.025, 0.975];
    else
        def = [0, -1];
        zielpunkt = [0.025, 0.975];
    end
end

%
% ecken

% unten links
if unten && links && gegner_nah
    if gegner_unten >= gegner_links
        def = [1,0];

```

```

else
    def = [0,1];
end

% unten rechts
elseif unten && rechts && gegner_nah
if gegner_unten >= gegner_rechts
    def = [-1,0];
else
    def = [0,1];
end

% links oben
elseif oben && links && gegner_nah
if gegner_obern >= gegner_links
    def = [1,0];
else
    def = [0,-1];
end

% rechts oben
elseif oben && rechts && gegner_nah
if gegner_obern >= gegner_rechts
    def = [-1,0];
else
    def = [0,-1];
end
end
end

%%%%%%%%%%%%%
%%%%% Kollisionsvermeidung während Tanken/Verteidigung %%%%
%%%%%%%%%%%%%

```

```

%%%%%%%%%%%%%
%%%%% Minen %%%%
%%%%%%%%%%%%%

function ges = mine_nach_tanke(tanke1, faktor)
la_vec = 1:spiel.n_mine;
ges = 1;
for z8 = 1:spiel.n_mine
    la_werte = schnitt_zwei_kreise(tanke1, ich.ges, spiel.spaceball_radius, spiel.mine(z8).pos, 0, minen_radius_safe);
    if la_werte > 0
        la_vec(z8, 1) = la_werte(1);
    end
end
mine_rel = min(la_vec(la_vec ~= 0));
if ~isempty(mine_rel)
    ges = sqrt(2*faktor*norm(mine_rel*ich.ges)*0.1);
end
end

%%%%% MINEN ELEGANT %%%%%%
%zielpunkt;
function minen_el
ich_zum_zp = zielpunkt - ich.pos;
if ~verteidigungsmodus
    notbremse_aus = 0;
end
minenmatrix = [0, 0, 0, 0, 0, 0];

for z2 = 1 : spiel.n_mine
    la_werte_z2 = schnitt_zwei_kreise(ich.pos, ich_zum_zp, ich_radius_safe, spiel.mine(z2).pos, 0, spiel.mine_radius); %ich.ges --> ich_zum_zp
    if la_werte_z2(1) > 0 && la_werte_z2(1) < 1 %Kollision mit mine zwischen mir und zielpunkt
        [p1, p2] = tangentialpunkte(ich.pos, spiel.mine(z2).pos, minen_radius_safe);

        ges_senkrecht = senkrecht(ich.ges);

        %berechnung des tp näher am ziel
        p_mat = [1, p1, norm(zielpunkt - p1);2, p2, norm(zielpunkt - p2)];
        p_mat_sorted = sortrows(p_mat, 4);
        p_ent = [p_mat_sorted(1, 2), p_mat_sorted(1,3)];
        if norm_ichges > 0
            M_k = ich.pos + projektion(p_ent - ich.pos, ges_senkrecht) * 1/norm(projektion(p_ent - ich.pos, ges_senkrecht)) * (norm_ichges^2 / 0.1);
            d = norm(spiel.mine(z2).pos - M_k);
            r = (norm_ichges^2 / 0.1)+spiel.mine_radius + spiel.spaceball_radius + 0.005;
            if d > r && norm_ichges < 0.15
                p_mat = [1, p1, norm(zielpunkt - p1);2, p2, norm(zielpunkt - p2)];
                p_mat_sorted = sortrows(p_mat, 4);
            else
                p_mat = [1, p1, Kosinus(p1-ich.pos, ich.ges);2, p2, Kosinus(p2-ich.pos, ich.ges)];
                p_mat_sorted = sortrows(p_mat, -4);
            end
        else
            p_mat = [1, p1, norm(zielpunkt - p1);2, p2, norm(zielpunkt - p2)];
            p_mat_sorted = sortrows(p_mat, 4);
        end
        tp_best = [p_mat_sorted(1, 2), p_mat_sorted(1,3)];
        ich_zu_tp_best = tp_best - ich.pos;
        tp_verschoben = false;
        for z3 = setdiff(1:spiel.n_mine, z2)
            la_werte_z3 = schnitt_zwei_kreise(ich.pos, ich_zu_tp_best, ich_radius_safe, spiel.mine(z3).pos, 0, spiel.mine_radius);
            if la_werte_z3(1) > 0 && la_werte_z3(1) < 1 %Kollision mit mine zwischen mir und tp_nah

```

```

[p1, p2] = tangentenpunkte(ich.pos, spiel.mine(z3).pos, minen_radius_safe);
if norm_ichges > 0
    p_mat = [1, p1, Kosinus(p1-tp_best, ich.ges);2, p2, Kosinus(p2-tp_best, ich.ges)];
    p_mat_sorted = sortrows(p_mat, -4);
    tp_verschoben = z3;
    tp_best = [p_mat_sorted(1, 2), p_mat_sorted(1,3)];
else
    p_mat = [1, p1, norm(zielpunkt - p1);2, p2, norm(zielpunkt - p2)];
    p_mat_sorted = sortrows(p_mat, 4);
    tp_verschoben = z3;
    tp_best = [p_mat_sorted(1, 2), p_mat_sorted(1,3)];
end
end
tp_am_rand =false;
for z5=1:2
    if tp_best(z5) < 0.015
        tp_best(z5) = (spiel.mine(z2).pos(z5) - spiel.mine_radius) /2;
        if tp_best(z5) < 0.015
            tp_best(z5) = 0.015;
            %experimentell, dauerhaft sinnvoll??
        end
        tp_am_rand = true;
    elseif tp_best(z5) > 0.985
        tp_best(z5) = 1 - ((1-spiel.mine(z2).pos(z5)) - spiel.mine_radius) /2;
        if tp_best(z5) > 0.985
            tp_best(z5) = 0.985;
        end
        tp_am_rand = true;
    end
    end
    ich_zu_tp_best = tp_best - ich.pos;
    minenmatrix(z2, 1) = tp_best(1);
    minenmatrix(z2, 2) = tp_best(2);
    minenmatrix(z2, 3) = ich_zu_tp_best(1);
    minenmatrix(z2, 4) = ich_zu_tp_best(2);
    minenmatrix(z2, 5) = norm(ich_zu_tp_best);
    minenmatrix(z2, 6) = tp_am_rand;
    minenmatrix(z2, 7) = tp_verschoben;
end
end
%minenmatrix;
minenmatrix_5 = minenmatrix(:, 5);
a = min(minenmatrix_5((minenmatrix_5~=0)));
i_vec = [];
for z9 = 1: length(minenmatrix_5)
    if abs(minenmatrix_5(z9)- a) < 0.0001
        i_vec(length(i_vec) + 1) = z9;
    end
end
i = 0;
if length(i_vec) > 1
    for z7 = 1:length(i_vec)
        if minenmatrix(i_vec(z7), 7) ~= 0
            if minenmatrix(i_vec(z7), 7) <= length(minenmatrix_5)
                i = minenmatrix(i_vec(z7), 7);
            else
                i = i_vec(z7);
            end
        end
    end
else
    i = i_vec;
end
if i ~= 0
    tp_best = [minenmatrix(i, 1), minenmatrix(i, 2)];
    ich_zu_tp_best = tp_best - ich.pos;
    cos_tp_best_ich = Kosinus(ich_zu_tp_best, ich.ges);
    if tankmodus && minenmatrix(i, 6)
        notbremse_aus(length(notbremse_aus)+1) = i;
        if spiel.i.t < 1000
            bande_aus = true;
        end
    else
        %bande_aus = false;
    end
    ges_senkrecht = senkrecht(ich.ges);

    if norm_ichges > 0
        M_k = ich.pos + projektion(ich_zu_tp_best, ges_senkrecht) * 1/norm(projektion(ich_zu_tp_best, ges_senkrecht)) * (norm_ichges^2 / 0.1);
        d = norm(spiel.mine(i).pos - M_k);
        r = (norm_ichges^2 / 0.1)+spiel.mine_radius + spiel.spaceball_radius + 0.005;
        if d > r && tankmodus
            notbremse_aus(length(notbremse_aus)+1) = i;
        end
    end
    if Kosinus(ich_zum_zp, ich_zu_tp_best) < 1
        z1 = zielpunkt(1);
        z2 = zielpunkt(2);
        t1 = tp_best(1);
        t2 = tp_best(2);
        zt = zielpunkt-tp_best;
        b = [-zt(2), zt(1)];
        b1 = b(1);
        b2 = b(2);
    end
end

```

```

a = projektion(ich_zum_zp, ich.ges) * 1/norm(projektion(ich_zum_zp, ich.ges));
a1 = a(1);
a2 = a(2);
alpha = (z1*b2 + b1*t2 - z2*b1 - t1*b2)/(a1*b2 - a2*b1);
M = tp_best + alpha * a;
r = norm(M-tp_best);
ges_faktor = 0.8;
v2a = ges_faktor*sqrt(0.1*r);
else
    v2a = 0.25;
end
if Kosinus(ich_zum_zp, ich.ges) > 0.999 || verteidigungsmodus
    v2a = 1;
end

if (norm(ich.ges) > 0.01 && abs(cos_tp_best_ich) < 0.9995) && cos_tp_best_ich > 0.5 || norm_ichges > 0.01 && norm_ichges < 0.03 && abs(cos_tp_best
    bes = projektion(ich_zu_tp_best, ges_senkrecht);
elseif (norm(ich.ges) > 0.03 && abs(cos_tp_best_ich) < 0.9995) && cos_tp_best_ich < 0.5 && cos_tp_best_ich > 0
    bes = -projektion(ich.ges, senkrecht(ich_zu_tp_best));
else
    la_vec = zeros(1, spiel.n_mine);
    for z6 = setdiff(1 : spiel.n_mine, 0)
        la_werte_3 = schnitt_zwei_kreise(tp_best, ich.ges, ich_radius_safe, spiel.mine(z6).pos, 0, spiel.mine_radius);
        if la_werte_3(1) > 0
            la_vec(z6) = la_werte_3(1);
        end
    end
    la_min = min(la_vec((la_vec~=0)));
    if ~isempty(la_min)
        ges_faktor2 = 0.4;
        v2b = sqrt(2*ges_faktor2*norm(la_min*ich.ges)*0.1);
    else
        v2b = 1; % hoch, damit v2a kleiner ist wenn v2b nicht berechnet wird
    end
    if v2a < v2b
        v2 = v2a;
    else
        v2 = v2b;
    end

    s_b2 = (norm_ichges^2 - v2^2)/(2*0.1);
    if (norm(ich_zu_tp_best) - s_b2) < 0.001 && norm_ichges > 0.01
        bes = -ich.ges;
    else
        bes = ich_zu_tp_best;
    end
end
end
zielpunkt;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% NOTBREMSE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function minen_not
    notbremse_aus;
    for z1 = setdiff(1 : spiel.n_mine, notbremse_aus)
        cos_zur_mine = Kosinus(spiel.mine(z1).pos - ich.pos, ich.ges);
        if cos_zur_mine > 0
            la_wert_beide = schnitt_zwei_kreise(ich.pos, ich.ges, ich_radius_safe, spiel.mine(z1).pos, 0, spiel.mine_radius);
            la_wert_kleinster = la_wert_beide(1);
            pos_bei_koll = ich.pos + la_wert_kleinster * ich.ges;
            ich_zu_koll = pos_bei_koll - ich.pos;
            norm_ich_zu_koll = norm(ich_zu_koll);

            if norm(ich_zu_koll) < 1.1*bremsweg && la_wert_kleinster ~= 0 && norm_ichges > 0
                bes = -ich.ges;
                if norm(ich_zu_koll) < bremsweg && ~bande && norm_ich_zu_koll > 0.06 %&& norm_ichges > 0 && la_wert_kleinster ~= 0
                    if norm_ichges > 0.02
                        bes = projektion(ich.ges, senkrecht(spiel.mine(z1).pos - ich.pos));
                    end
                end
            else
                if norm_ich_zu_koll > 0 && norm_ich_zu_koll < 0.06
                    bes = projektion(zielpunkt - ich.pos, senkrecht(spiel.mine(z1).pos - ich.pos));
                end
            end
        end
    end
end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Bandenvermeidung %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Bandenvermeidung
    %Kollision Kreis mit Gerade nach Buchholz Dokumentation
    d_1 = 1; %Abstand der Bande oben und rechts zum Ursprung
    d_2 = 0; %Abstand der Bande unten und links zum Ursprung
    r = 0.014; %default 0.015

    pos = ich.pos;
    n_1 = [0, 1]; %Einheitsnormalenvektor Bande oben
    n_2 = [1, 0]; %Einheitsnormalenvektor Bande rechts
    n_3 = [0, -1]; %Einheitsnormalenvektor Bande unten
    n_4 = [-1, 0]; %Einheitsnormalenvektor Bande links
    v = ich.ges;

    %Zeitvariablen bis Kollision mit:
    la_1 = (d_1-r-(pos*n_1.'))/(v*n_1.');?>

```

```

la_2 = (d_1-r-(pos* n_2.'))/(v* n_2.');?>Bande rechts
la_3 = (d_2-r-(pos* n_3.'))/(v* n_3.');?>Bande unten
la_4 = (d_2-r-(pos* n_4.'))/(v* n_4.');?>Bande links

la = [la_1, la_2, la_3, la_4];
n_b = [n_1; n_2; n_3; n_4];
d_b = [d_1; d_1; d_2; d_2];
bande = false;

kol_1 = false;
kol_2 = false;
for x_2 = 1:4

    if la(x_2) > 0
        pos_kol = pos+la(x_2)*v;

        if norm(pos_kol-pos) < 1.1*bremsweg && tankmodus %im Tankmodus einfach Geschwindigkeit verringern
            bes = -ich.ges;
            kol_1 = false;
            kol_2 = false;
            bande = true;
            %disp bande
        elseif norm(pos_kol-pos) < 1.1*bremsweg && (angriffsmodus || verteidigungsmodus) && n_b(x_2)==0 %im Angriffsmodus nur senkrecht zu Bande bshle
            if norm(gegner.ges) < 0.01 && angriffsmodus || angriffsmodus && ~bande_angriff && Kosinus(gegner.ges, ich.ges) > 0.999 && ...
                Kosinus((gegner.pos-ich.pos), ich.ges) > 0.999 && norm(gegner.pos-ich.pos) < 0.3
                bes = -ich.ges;
                %disp bremsen
            else
                bes(1) = 0;
                bes(2) = -n_b(x_2+4);
            end
            kol_1 = true;
            %disp bande
        elseif norm(pos_kol-pos) < 1.1*bremsweg && (angriffsmodus || verteidigungsmodus) && n_b(x_2+4)==0
            if norm(gegner.ges) < 0.01 && angriffsmodus || angriffsmodus && ~bande_angriff && Kosinus(gegner.ges, ich.ges) > 0.999 && ...
                Kosinus((gegner.pos-ich.pos), ich.ges) > 0.999 && norm(gegner.pos-ich.pos) < 0.3
                bes = -ich.ges;
                %disp bremsen
            else
                bes(1) = -n_b(x_2);
                bes(2) = 0;
            end
            kol_2 = true;
            %disp bande
        end
    end
end

if kol_1 && kol_2
    bes = -ich.ges; %in Ecken bremsen
end

d_s = zeros(4, 1);
for x_10 = 1:4
    n_rel(1)= n_b(x_10); %relevanten Einheitsnormalenvektor raussuchen
    n_rel(2)= n_b(x_10+4);
    d_s(x_10, 1) = d_b(x_10)-(pos* n_rel.');?>
    %Abstand zur Bande berechnen
    if d_s(x_10, 1) < 0.015
        bes = -n_rel; %Wenn man sehr nah dran ist von der Bande weg beschleunigen
        bande = true;
        %disp('Notbremse')
    end
end

d_krit = find(d_s < 0.015);
if length(d_krit)==2
    bes = [0.5, 0.5]-ich.pos;
    bande = true;
    %disp('Ecke')
end
end

%
%%%%%%%%%%%%%%% Lebenserhaltung %%%%%%
%bande_aus = false;

if ~fuenffuenf
% if ~verteidigungsmodus
minen_el
% end

bes_vor_bande = bes;

if ~bande_aus
    Bandenvermeidung

```

```
end

if bes ~= bes_vor_bande
    bande = true;
else
    bande = false;
end

if ~bande_aus && (bande || kol_1 || kol_2)
    notbremse_aus = 0;
end

minen_not
end
```

```
Published with MATLAB® R2017a
```