Table of Contents

Class Header

```
%   Class Name: FlightPath.m
%   Call: name = FlightPath()
%
%   Zweck: In der Instanz dieser klasse findet die berechnung der
 Flugbahn
%   statt. Wieter stellen Instanzen dieser klasse die methode zur
%   trefferermittlung zu verfügung
%
```

Changelog

- Version 00.00.01 07.10.15 Raphael Waltenspül Erstellt des Main Objekts, noch nicht Objekt Orientiert.
- Version 00.00.07 19.10.15 Joel Koch Einschlag implementier
- Version 00.00.08 21.10.15 Raphael Waltenspül Flugparabel berechnung erster entwurf
- Version 00.00.09 22.10.15 Raphael Waltenspül Koordinaten der Flugparabel berechnen und ausgeben
- Version 00.01.00 22.11.15 Raphael Waltenspül Umbau in Objektoriert erfogt
- Version 00.01.04 11.12.15 Raphael Waltenspül Implementieren der Funktionen des Landscapes in Handle Class Landscape
- Version 00.01.06 28.12.15 Raphael Waltenspül Trefferermittlung
- Version 00.01.08 31.12.15 Raphael Waltenspül Trefferermittlung
- Version 00.01.11 02.01.16 Raphael Waltenspül Aufräumen fertigstellen Gameablauf
- Version 01.00.00b 03.01.16 Raphael Waltenspül Buglist Testen Kommentieren Dokumentieren

Input und Output:

für Methoden, siehe Methoden

```
%   Konstruktor:   void
%   Precondition:
%
%   Postcondition: Instanz von FlightPath ist erstellt
%
% Variables:
%       Für Instanzvariabeln siehe Properties
%
```

Implementierte Methoden

```
% [this]= FlightPath()
% [retCoordinates] = calcCoordinates(this, energie, winkel, Wether,
 Landscape, Player)
% [percentDamage] = isHit(this, PlayerArray, playerNr)
%
```

Buglist TODO / this

Berechnen der Flugbahn

```
classdef  FlightPath < handle
    properties
        impact; % Einschlagkoordinatn [x;y]
    end

    methods
```

FlightPath Konstruktor

Zweck: Instanz von GameStates ist erzeugt

```
% Pre:
%
% Post: FlightPath ist erstellt
%
% Input: void
%
% Output: Instanz FlightPath
%
% Modifizierte Instanzvariable
        function [this]= FlightPath()

        end
```

FlightPath calcCoordinates

Zweck: Berechnet die Koordinaten der Flugbahn numerisch Dies unter berücksichtigung von Wind und Lufztwiederstand

```
% Pre: instanz FlightPath it erstellt
```

```
% energie ist zwischen 0 und 100000
% winkel ist zwischen 0 und 180 Grad
% Instanzen Wether, Landscape, Player sind erstellt
%
% Post: Flugbahnkoordinaten sind berechnet
%
% Input: instanz FlightPath
% energie
% winkel
% Wether
% Landscape
% Player
%
% Output: retCoordinates [x;y] -- Array mit den
% Flugbahnkoordinaten
%
        function [retCoordinates] = calcCoordinates(this, energie,
 winkel, Wether, Landscape, Player)
```

Parameter

```
% Der Winkel in Grad und Rad
%
% $$ang_{rad} = \pi  \frac{ang_{deg}}{180}$$
%
            angle = winkel;
            angRad = pi() * angle/180;
```

Berechnung der Abbschussgeschwindigkeit

```
% $$Energie Projektil: E_{prj} \ Geschwindigkeit Projektil: v_{prj} \
 Msse Projektil: m_{Projektil}$$
%
% $$Wirkungsgrad Kanon: n_{can} Energie Treibladung $$
%
% $$E_{prj} = \frac{m_{prj}}{2} v_{prj}^{2}$$
%
% $$E_{prj} = E_{trbl} * n_{can}$$
%
% $$v_{prj} = \sqrt{ \frac{2 E_{prj}}{m_{prj}}}$$

            masseProjektil = 1; % Variable erst in späteren Versionen
 veränderbar
            energieTreibladung = energie; % Variable erst in späteren
 Versionen veränderbar
            wirkungsGradKanone = 1; % Variable erst in späteren
 Versionen veränderbar
            masseKanone = 10000; % Variable erst in späteren Versionen
 verwendet
            g = 9.81; % Variable erst in späteren Versionen
 veränderbar (Wenn Planten implementiert)
```

```
            % Die Energie wird auf das Projektil gerechnet
            vStart = sqrt( 2 * energieTreibladung *
wirkungsGradKanone / masseProjektil);
            % Die Startgeschwindigkeiten werden berechnet
            vxStart = cos(angRad) * vStart; %Startgeschwindigkeit in X
Richtung
            vyStart = sin(angRad) * vStart; %Startgeschwindigkeit in Y
Richtung

            % Die Maximale zeit für die Berechnung
            tmax = 60;

            dichteMedium = 1.2 *10^(-3); % Dichte der Athmosphäre
Variable erst in späteren Versionen veränderbar (Wenn Planten
implementiert)
            koeffzient = 1; % koeffizient des Geschosses

            %deltaT = 0.001; % Schrittweite für die Numerische
Berechnung
            deltaT = 0.002; % Schrittweite für die Numerische
Berechnung geändert Joel Koch 4.1.16: Sonst reichts nicht beim
vertikalen Schuss zurück auf den boden.

            vx = (vxStart); % Startgeschwindigkeit in X Richtung
speichern
            vy = (vyStart); % Startgeschwindigkeit in Y Richtung
speichern
            n=1; % Schleifenzähler auf eins

            playerStartPos = Player.positionXY; % Startposition
speichern
            x(n) = playerStartPos(1,1);  % Schuss X Startposition
festlegen
            y(n) = playerStartPos(2,1)+10; % Schuss Y Startposition
festlegen
```

Schleife zur Numerischen berechnung der Flugparabel

```
            for t = 0 : deltaT : tmax
                ve = [vx, vy] / sqrt(vx^2 + vy^2); % Einsvektor für
die Kraft des Flugluftwiederstandes
                fVector = ve * (sqrt(vx^2 + vy^2)^2 * koeffzient *
dichteMedium); % Kraft des Flugluftwiederstandes berechnen [Fx;Fy]
                % änderung v in X richtung berechnen inklusive der
                % Windrichtung
                vx = vx - fVector(1,1)*deltaT - (Wether.wind + vx) *
abs(Wether.wind + vx) * koeffzient * dichteMedium * deltaT;
                % änderung v in Y richtung berechnen inklusive der
                vy = vy - g * deltaT - fVector(1,2)*deltaT;

                % Speichern der Parameter in einem Array
```

```
                x(n+1) = x(n)+vx * deltaT;
                y(n+1) = y(n)+vy * deltaT;
                n = n+1;
            end

            coordinates = [x;y]; % Speichern der Koordinaten

            landArray = Landscape.getLandscape; % Speichern des
Landscapes Array

            %finalLength = 13000; % Die Finale Länge des Arrays (JOKO:
welches Array?? coordinates wäre ja 300000 lang)
            finalLength = 30000;

            for ak = 1 : 1 : length(coordinates)
                xcor = round(coordinates(1,ak)); % gerundter x Wert an
stelle ak im Array
                ycor = coordinates(2,ak); % y Wert an stelle ak im
Array

                % Wenn nun an der Stelle X der Y Wert der Flugbahn
kleiner ist als der Y Wert der Landscape so ist dies ein einschlag
                % Weil die Array-Grösse des LAndscape-Polygons
variabel ist
                % (Krater), muss di Prüfung mit "inpoygon" erfolgen:

                % alter Code:
                %if  ycor < landArray(2, xcor)
                %    this.impact = [xcor; ycor];
                %    finalLength = ak; % Die länge der Flugbahn wird
begrenzt
                %    break
                %end

                in = inpolygon(xcor,ycor,landArray(1,:),
landArray(2,:));
                if in ~= 0
                    this.impact = [xcor; ycor];
                    finalLength = ak; % Die länge der Flugbahn wird
begrenzt
                    fprintf('   => ak limit =  %d' , ak)
                    break
                end



                % Wenn nun die Stelle X grösser ist als 1000 wurde aus
dem
                % bild gefeuert
                if xcor > 999
                    this.impact = [1000; ycor];
                    finalLength = ak;
                    break
                end
```

```
                % Wenn nun die Stelle X grösser ist als 4 wurde aus
 dem
                % bild gefeuert
                if xcor < 4
                    this.impact = [1; ycor];
                    finalLength = ak;
                    break
                end
            end
            retCoordinates = coordinates(:,1:finalLength); % Die
 Koordinaten werden zurückgegeben

        end
```

FlightPath isHit

Zweck: Prüfen ob ein Speiler getroffen wurde

```
% Pre: instanz FlightPath it erstellt,
%  playerNr ist bekannt und übergeben
%
% Post: Ermittelt ob der Spieler mit der playerNr getroffen wurde
% und der Schaden wurde zurückgegeben
%
% Input: instanz FlightPath
% PlayerArray --
% playerNr --
%
% Output: percentDamage-- Schaden am Spieler
%
        function [percentDamage] = isHit(this, PlayerArray, playerNr)

            tempPsXY = PlayerArray(playerNr).tankArray; % Spieler wird
 gespeicher
            % Prüfen ob impaktkoordinaten iauf dem Spieler liegen
            if this.impact(1,1) > min(tempPsXY(1,:)) &&
 this.impact(1,1) < max(tempPsXY(1,:))
                percentDamage = 100; % 100 prozent Schaden wenn
 getroffen
            else
                percentDamage = 0; % 0 prozent Schaden wenn nicht
 getroffen
            end
        end

    end

end
```

Published with MATLAB® R2015b