Table of Contents

Class Header

```
%   Class Name: Landscape.m
%   Call: name = Landscape(GameParameter)
%
%   Zweck: In der Instanz dieser Klasse werdird die Landschaft erzeugt
 und gespeichert
%
```

Changelog

- Version 00.00.04 16.10.15 Joel Koch Erstes Gui entworfen
- Version 00.00.05 17.10.15 Joel Koch Lanschaft Algorithmus implementiert
- Version 00.00.11 28.10.15 Joel Koch Code Aufgeräumt
- 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 00.01.11 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:    >> GameParameter
%                   << Figure Instanz
```

```
%   Precondition:  GameParameter ist dem Konstruktor übergeben
%
%   Postcondition: Ein Landscape Instanz ist erstellt
%
% Variables:
%       Für Instanzvariabeln siehe Properties
%
```

Implementierte Methoden

```
%       this = Landscape(GameParameter)
%       this = genLandscape(this)
%       terrainArray = getLandscape(this)
```

Buglist TODO / this

Properties

```
classdef Landscape < handle
    properties
        gameParameter;  % Instanz der Gameparameter
        terrainArray;   % variable des Terrainarrays
    end

    methods
```

Landscape Kostruktor

Zweck: Erstellt eine Landscape Instanz und speicher die Gamparameter

```
        % Pre: Instanz Gameparameter ist erstellt
        %
        % Post: Instanz von Landscape ist erstellt
        %
        % Input: GameParameter
        %
        % Output: void
        %
        function this = Landscape(GameParameter)
            this.gameParameter = GameParameter;
        end
```

Landscape genLandscape

Zweck: Erstellt ein zufälliges Landscape Array

```
        % Pre: Instanz Landscape ist erstellt
        %
        % Post: Landscape Array ist berechnet und in der
 Instanzvariabel
        % terrainArray gespeichert
```

```
%
% Input: GameParameter
%
% Output: void
%
% Modifizierte InstanzVraiabeln: terrainArray
%
function this = genLandscape(this)
```

Limits für max_iterations durchsetzten

```
max_iterations = floor(this.gameParameter.max_iterations);
if (max_iterations < 1)
    max_iterations  = 1;
end
if (max_iterations > 9)
    max_iterations  = 9;
end  % 1+2^9 Punkte reichen aus!
```

Vektor initialisieren

```
terrain = zeros(max_iterations,2^max_iterations+1);
```

Start und Ende und mittelpunkt setzen

```
            terrain (1,2) = rand*1.5*this.gameParameter.JITTER +
 this.gameParameter.BERGOFFSET; % Mittenwert (der Berg)
            terrain (1,1) = rand*50;   % Startwert (Linker Rand)
            terrain (1,3) = rand*50;   % Endwert (rechter Rand)

            JITTER = this.gameParameter.JITTER;
            DAEMPFUNG = this.gameParameter.DAEMPFUNG;
            JITTERBALANCE = this.gameParameter.JITTERBALANCE;
            PLATFORMOFFSET = this.gameParameter.PLATFORMOFFSET;

        % die Terrainpunkte werden in einer Matrix erstellt. ähnlich wie bei
         der
        % erstellung des pascalschen Dreiecks. Zu Beginn sind nur in der
         ersten
        % Zeile die Werte von Links,JITTER Rechts und der Mitte (Berg). Jede
         Iteration
        % erzeugt die Werte in eine nächste Zeile und fügt dort die neuen
         Punkte
        % ein. start bei 2, weil die erste bereits gesetzt (die 3
         Startpunkte).

            for rowindex=2:1:this.gameParameter.max_iterations   %für
         jede iteration gibts eine neue Zeile  in der Matrix
               for colindex=1:1:2^rowindex+1   %Jede Zeile hat mehr
         Werte als die letzte
                    if mod(colindex, 2) > 0    %ungerade zeilen
         übernehmen bestehende werte (verschoben)
```

```
                        terrain(rowindex,colindex)=
 terrain(rowindex-1, (colindex+1)/2);
                    else    %gerade Zeilen berechnen einen neuen
 mittelwert +- random
                        left= terrain(rowindex-1, (colindex)/2);
                        right= terrain(rowindex-1, (colindex+2)/2);

% Die Dämpfung wichtig: Einerseits soll die Gesamtlandschaft
% nicht zu flach sein, andernseits müssen die
% Zufallshöhenunterschiede bei fortschreitenden Detailgraden
% immer kleiner werden. Um zu Beginn wenig zu dämpfen und
% später sehr stark, wird die DAEMPFUNG^ITERATION verwendet.
% die Korrektur an der Iteration (rowidex-1.8) stellt quasi den
% Arbeitspunkt der Dämpfung ein.
                        terrain(rowindex,colindex)= (left+right)/2
 +...
                            (rand*JITTER-(JITTER*(1-JITTERBALANCE)))/
DAEMPFUNG^((rowindex-2.4)*DAEMPFUNG^2.2);
                    end
                end


% *Ein paar Korrekturen für die Positionierung, es geht
% am einfachsten in der 4. Iteration, wenn 17 Punkte gesetzt sind:
                if rowindex == 4 % Beide Spieler  etwas nach unten.
                    terrain(4,2) = max(terrain(4,2) +
 PLATFORMOFFSET,5);
                    terrain(4,16)= max(terrain(4,16) +
 PLATFORMOFFSET,5);
                    %Spielfeld gegen aussen flacher machen, gegen
 innen leicht flacher.
                    terrain(4,1) = terrain(4,1)-((terrain(4,1)-
terrain(4,2))/1.1);
                    terrain(4,3) = terrain(4,3)-((terrain(4,3)-
terrain(4,2))/2);
                    % auch für den anderen Spieler:
                    terrain(4,17) = terrain(4,17)-((terrain(4,17)-
terrain(4,16))/1.1);
                    terrain(4,15) = terrain(4,15)-((terrain(4,15)-
terrain(4,16))/2);
                end

            end


            YLIMITS = this.gameParameter.YLIMITS;
```

Limits

```
            %Versetzen Tiefster Punkt als Refernz auf YLIMITS(1)
            lowestpoint=min(terrain(max_iterations,:));
            terrain=terrain-lowestpoint+YLIMITS(1);
```

```
            %neuen höchsten Punkt suchen, wenn höher als
            %limite, wird das ganze terrain zusammengestaucht
            highestpoint=max(terrain(max_iterations,:));
            if highestpoint > YLIMITS(2)
                terrain=terrain/(highestpoint/YLIMITS(2));
            end

            % auf 1000 punkte aufblasen
            terrain= imresize(terrain,[max_iterations,
 1001], 'Method','bilinear');
```

Glättung

```
            contour_raw=terrain(max_iterations,:);  %relevante letzte
 zeile aus den generierten Terrain Daten kopieren
            contour_soft=contour_raw;               %Diese Version
 wird geglättet
            contour_mix=contour_raw;                %Diese Version
 wird die gemischte
            POSTSMOOTHING = this.gameParameter.YLIMITS;

            for s=1:floor(POSTSMOOTHING/1*2^max_iterations)   % so oft
 durchlaufen, wie konfiguriert ist
                for colindex=2:1:1000 % Letzte Zeile ist relevant ==>
 glätten

 mittelwert=(contour_soft(colindex-1)+contour_soft(colindex+1))/2; %
 Mittelwert von der 2 nachbarpunkte
                        difference = contour_soft(colindex)-
mittelwert; % Abweichung gegenüber dem mittel der 2 Nachbarpunkte
                        contour_soft(colindex)=
 contour_soft(colindex)-0.1*(difference); %Angleichen in kleinen
 Schritten
                end
            end
            FELSUEBERGANG  = this.gameParameter.FELSUEBERGANG;

            %Mix raw und soft anhand der Parameter FELSUEBERGANG(1)
 und (2)
            for colindex=1:1:size(contour_raw,2)
               if  contour_raw(colindex) < FELSUEBERGANG(1)     % Nur
 Berge
                    contour_mix(colindex)=contour_soft(colindex);
                elseif contour_raw(colindex) > FELSUEBERGANG(2) % Nur
 Hügel
                    contour_mix(colindex)=contour_raw(colindex);
                else % Mischung
                    felsanteil= (contour_raw(colindex)-
FELSUEBERGANG(1))/(FELSUEBERGANG(2)-FELSUEBERGANG(1));
                    contour_mix(colindex)=
 felsanteil*contour_raw(colindex) + (1-
felsanteil)*contour_soft(colindex);
                end
```

```
            end

            % make polygon
            terrainshapeY = [0, (contour_mix), 0];
                        % die interssante zeile übernehmen vorne ein
und hinten zwei 0 als y-wert
            terrainshapeX = [0, 0:1:size(terrainshapeY,2)-3,
size(terrainshapeY,2)-3 ];      % die X-werte füllen, am schluss
wieder auf x=0 weil für polygon
            c = terrainshapeY;
            terrain = [terrainshapeX terrainshapeY];
```

Stretch Y

```
            fittingTerrainX=terrainshapeX;
            fittingTerrainY=terrainshapeY.*3.5;

            this.terrainArray=[fittingTerrainX; fittingTerrainY];

        end
```

Landscape getLandscape

Zweck: getter für die LandscapeVaraiable

```
        % Pre: Instanz Landscape ist erstellt
        % LandscapeVaraiable berechnet
        %
        % Post: Landscape Array ist ausgegeben
        %
        % Input: Landsacpe Instanz, Instanzvariabel
        %
        % Output: terrainArray -- Zufälliges Landscape Array
        %
        function terrainArray = getLandscape(this)
            terrainArray = this.terrainArray;
        end

    end

end
```

Published with MATLAB® R2015b