Class Header

Table of Contents

Class Name: Figure.m Call: name = Figure(GameStates,GameParameter)

Zweck: In dieser Klasse befinden sich alle Methoden zum mit dem User zu Interagieren Dies beinhalten zeichnen der Menues sowie allen spielelemente, sowie die Eventhandler Dies ist die Umfangreichste Klasse des Projektes Empfehlung für Workaround: Parent Class für Anzeigeinstanzen Child für Menu und Gamescreen Wieter sollte der Powerbar als eigene Klasse implementiert werden

Changelog

- Version 00.01.03 11.12.15 Raphael Waltenspül Neu Erstellen der Handle Classes Figure
- Version 00.01.01 12.12.15 Raphael Waltenspül Menu zur eingabe von Parametern in dieser Klasse implementiert
- Version 00.01.05 20.12.15 Raphael Waltenspül Implementieren der Funktione Element Zeichnen, Intersection implementiert
- Version 00.01.09 31.12.15 Raphael Waltenspül Implementieren der Funktione Powerbar und Eventhandler in Figure,
- Version 00.01.10 01.01.16 Raphael Waltenspül Neu Entwickeln der Buttons / Game Mode Taktik in Figure
- 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:    >> GameStates,GameParameter
%                 << Figure Instanz
% Precondition:
% GameStates,GameParameter sind dem Konstruktor übergeben
%
% Postcondition:
% Erzeugt ein Grafikfenster in dem das Spiel Abläuft, gibt eine Figure
% Instanz zurücck
%
% Variables:
% Für Instanzvariabeln siehe Properties
%
```

Implementierte Methoden

```
% this = Figure(GameStates,GameParameter)
% [] = drawMenue(this)
% [] = drawGamescreen(this)
% [] = drawActualPlayer(this, GameState, color)
% [] = drawPlayerPoints(this, GameParameter, Player)
% [] = drawGameRound(this, GameParameter, GameState)
% [] = drawGameButtons(this)
% [] = drawPowerBar(this)
% [] = updatePowerBar(this,power)
% [p] = drawInScreen(this,terrain)
% [p] = updateInScreen(this, terrain)
% [p] = drawElement(this, shape)
% [p] = drawElementCol(this,shape,color)
% [] = deleteElement(this,p)
% [] = updateElement(this,p, shape)
% [] = updateElementCol(this,p,shape,color)
% [] = drawShockwave(this, impact)
% [newTerrain] = drawImpactCircle(this, terrain, impact)
% [intersections] = getOuterIntersections(this, x1arr, y1arr, centerX,
 centerY,r)
% [] = updateState(this,GameStates)
% [] = updateParameters(this,GameParameter)       ^
% [GameParameter] = getParameters(this)
% [fig] = getFig(this)
% [power] = getPower(this)
% [angel] = getAngle(this)
% btnPlayerCountClick(this,source,eventdata)
% btnGameModeClick(this,source,eventdata)
% btnWindClick(this,source,eventdata);
% btnMountainClick(this,source,eventdata)
% btnPlanetClick(this,source,eventdata)nd
% sldRoundsChange(this,source,eventdata)
% btnStartClick(this,source,eventdata)
% [] = btnFireClick(this,source,eventdata)
% btnAngleClick(this,source,eventdata)
% btnPowerClick(this,source,eventdata)
% myMouseDownCallBack(this,hObject,~)
% myMouseUpCallBack(this,hObject,~)
```

```
%
```

Buglist TODO

Empfehlung für Workaround:

Parent Class für Anzeigeinstanzen Child für Menu und Gamescreen Powerbar als eigene Klasse implementieren

instanzVariabeln

```
classdef Figure < handle
    properties
        screenSize = get(0,'ScreenSize'); %Variable für die grösse des
 aktuellen Bildshirms
        fireEvent  = 0; % -- variable für Programmsteuerung {0= Schuss
 nicht erfolgt 1= Schuss erfolgt}
        mousedown;      % - variable für Programmsteuerung {0= Maus
 nicht gedrückt1= Maus gedrückt}
        mouseX;         % -- Xpoition der Maus auf dem Bildschirm
        mouseY;         % -- Ypoition der Maus auf dem Bildschirm
    end

    properties (Access = private)
        fig;                % Figure Object
        title;              % Titeltext als String
        gameParameter;      % instanz des Parameter Klasse aus
 Konstruktor
        gameStates;         % instanz des Status Klasse aus
 Konstruktor
        terrainhandler;     % speichern des Terrainhandler zum löschet
 etc
        btnPlayerCount;     % Menubutton Anzahl Spieler
        btnMode;            % Menubutton Modewahl
        btnWind;            % Menubutton whal Windstärke
        btnMountain;        % Menubutton Wahl Berghöhe
        btnPlanet;          % Menubutton Wahl Planet
        txtRounds;          % Menuetext Ausgabe Runde
        sldRounds;          % Menuslider Wahl Runde
        btnStart;           % Menubutton start spiel
        player;             % Gametext Aktueller Spieler
        fire;               % Gamebutton Fire
        angleText;          % Gametext Gewählter Winkel
        angleSlider;        % Gameslider Wahl Winkel
        powerText;          % Gametext gewählte Power
        powerSlider;        % Gamslider Wahl Power
        playerPoints;       % Gametext Punkte Array
        gameRound;          % Gametext AktuelleRunde
    end

    methods
```

Konstruktor

```
% Zweck:
%
% Input:
%
% Output:
%
function this = Figure(GameStates,GameParameter)
   this.fig = figure;
   this.fig.Visible = 'off';
   this.gameStates = GameStates;
   this.gameParameter = GameParameter;
end
```

Zeichnet Menue

Zweck: Zeichnet das Spielmenue mit allen Tasten.

```
% Pre: Instanz erstellt
% Post: Menue ist gezeichnet und Angezeigt
%
% Input:  this Figure Instant, Instanzvariabeln
%
% Output: void
%
function [] = drawMenue(this)
```

Figure erstellen und Parametrieren

```
this.fig.Name = 'Artillery Menue';
this.fig.MenuBar = 'none';
this.fig.ToolBar = 'none';
this.fig.NumberTitle = 'off';
this.fig.Position = this.gameStates.MENUE_POSITION;
this.fig.Color = this.gameStates.BLACK;
```

Die Objekte im fig erstellen

der Titel erstellen und parametrieren

```
this.title = uicontrol;
this.title.Style = 'text';
this.title.String = 'Welcome to Artillery';
this.title.Position =
[0,this.gameStates.MENUE_HIGH-65,this.gameStates.MENUE_WIDTH,35];
this.title.ForegroundColor = this.gameStates.TITLE_COLOR;
this.title.BackgroundColor = this.gameStates.BLACK;
this.title.FontName = this.gameStates.FONT;
this.title.FontSize = this.gameStates.TITLE_SIZE;
```

Auswahl btn Spieleranzahl der Titel erstellen und parametrieren

Eventhandler erstellen

```
            this.btnPlayerCount = uicontrol;
            this.btnPlayerCount.Style = 'pushbutton';
            this.btnPlayerCount.String = ['N off Players >> ',
 num2str(this.gameParameter.playerQuantety)];
            this.btnPlayerCount.Position =
 [0,this.gameStates.MENUE_HIGH-115,this.gameStates.MENUE_WIDTH,25];
            this.btnPlayerCount.ForegroundColor =
 this.gameStates.GREEN;
            this.btnPlayerCount.BackgroundColor =
 this.gameStates.BLACK;
            this.btnPlayerCount.FontName = this.gameStates.FONT;
            this.btnPlayerCount.FontSize = this.gameStates.TEXT_SIZE;
            this.btnPlayerCount.Callback = @this.btnPlayerCountClick;
```

Auswahl btn Spielmodi erstellen und parametrieren

Eventhandler erstellen

```
            this.btnMode = uicontrol;
            this.btnMode.Style = 'pushbutton';
            this.btnMode.String = [this.gameParameter.gameMode];
            this.btnMode.Position =
 [0,this.gameStates.MENUE_HIGH-160,this.gameStates.MENUE_WIDTH,25];
            this.btnMode.ForegroundColor = this.gameStates.GREEN;
            this.btnMode.BackgroundColor = this.gameStates.BLACK;
            this.btnMode.FontName = this.gameStates.FONT;
            this.btnMode.FontSize = this.gameStates.TEXT_SIZE;
            this.btnMode.Callback = @this.btnGameModeClick;
```

Auswahl btn Wetter / Wind erstellen und parametrieren

Eventhandler erstellen

```
            this.btnWind = uicontrol;
            this.btnWind.Style = 'pushbutton';
            this.btnWind.String = [this.gameParameter.wind];
            this.btnWind.Position =
 [0,this.gameStates.MENUE_HIGH-205,this.gameStates.MENUE_WIDTH,25];
            this.btnWind.ForegroundColor = this.gameStates.GREEN;
            this.btnWind.BackgroundColor = this.gameStates.BLACK;
            this.btnWind.FontName = this.gameStates.FONT;
            this.btnWind.FontSize = this.gameStates.TEXT_SIZE;
```

```
            this.btnWind.Callback = @this.btnWindClick;
```

Auswahl btn Berge erstellen und parametrieren

Eventhandler erstellen

```
            this.btnMountain = uicontrol;
            this.btnMountain.Style = 'pushbutton';
            this.btnMountain.String = [this.gameParameter.mountain];
            this.btnMountain.Position =
 [0,this.gameStates.MENUE_HIGH-250,this.gameStates.MENUE_WIDTH,25];
            this.btnMountain.ForegroundColor = this.gameStates.GREEN;
            this.btnMountain.BackgroundColor = this.gameStates.BLACK;
            this.btnMountain.FontName = this.gameStates.FONT;
            this.btnMountain.FontSize = this.gameStates.TEXT_SIZE;
            this.btnMountain.Callback = @this.btnMountainClick;
```

Auswahl btn Planet erstellen und parametrieren

Eventhandler erstellen

```
            this.btnPlanet = uicontrol;
            this.btnPlanet.Style = 'pushbutton';
            this.btnPlanet.String = [this.gameParameter.planet];
            this.btnPlanet.Position =
 [0,this.gameStates.MENUE_HIGH-385,this.gameStates.MENUE_WIDTH,25];
            this.btnPlanet.ForegroundColor = this.gameStates.GREEN;
            this.btnPlanet.BackgroundColor = this.gameStates.BLACK;
            this.btnPlanet.FontName = this.gameStates.FONT;
            this.btnPlanet.FontSize = this.gameStates.TEXT_SIZE;
            this.btnPlanet.Callback = @this.btnPlanetClick;
```

Anzahl Runden erstellen und parametrieren

```
            this.txtRounds = uicontrol;
            this.txtRounds.Style = 'text';
            this.txtRounds.String = ['Rounds >> ',
 num2str(this.gameParameter.numberRounds)];
            this.txtRounds.Position =
 [0,this.gameStates.MENUE_HIGH-295,this.gameStates.MENUE_WIDTH,25];
            this.txtRounds.ForegroundColor = this.gameStates.GREEN;
            this.txtRounds.BackgroundColor = this.gameStates.BLACK;
            this.txtRounds.FontName = this.gameStates.FONT;
            this.txtRounds.FontSize = this.gameStates.TEXT_SIZE;
```

Anzahl Rundene Slider rstellen und parametrieren

Eventhandler erstellen

```
            this.sldRounds = uicontrol;
            this.sldRounds.Style = 'slider';
            this.sldRounds.String = 'Rounds';
            this.sldRounds.Position =
[0,this.gameStates.MENUE_HIGH-340,this.gameStates.MENUE_WIDTH,25];;
            this.sldRounds.ForegroundColor = this.gameStates.RED;
            this.sldRounds.BackgroundColor = this.gameStates.BLACK;
            this.sldRounds.FontName = this.gameStates.FONT;
            this.sldRounds.FontSize = this.gameStates.TEXT_SIZE;
            this.sldRounds.Callback = @this.sldRoundsChange;
```

Startbutton erstellen und parametrieren

Eventhandler erstellen

```
            this.btnStart = uicontrol;
            this.btnStart.Style = 'pushbutton';
            this.btnStart.String = '>> Start the best Game >>';
            this.btnStart.Position = [0,
0,this.gameStates.MENUE_WIDTH,25];
            this.btnStart.ForegroundColor = this.gameStates.RED;
            this.btnStart.BackgroundColor = this.gameStates.BLACK;
            this.btnStart.FontName = this.gameStates.FONT;
            this.btnStart.FontSize = this.gameStates.TEXT_SIZE;
            this.btnStart.Callback = @this.btnStartClick;
```

Menu Anzeigen

```
            this.fig.Visible = 'on';

        end
```

Zeichnet Spielfeld im Fullscreen Modus

```
        % Zweck: Zeichnet das Spielfeld
        % Pre: Instanz erstellt
        % Post: Menue ist gezeichnet und Angezeigt
        %
        % Input:  this Figure Instant, Instanzvariabeln
        %
        % Output: void
        %
        function [] = drawGamescreen(this)
```

Laden der Konstanten

```
            FIGURE_COLOR = this.gameStates.BACK_BLACK;
            AXIS_COLOR = this.gameStates.SKY;
            FONT = this.gameStates.FONT;
            LARGE_TEXT = this.gameStates.TITLE_SIZE;
            TITLE_COLOR = this.gameStates.TITLE_COLOR;
```

Spielfeld Parametrieren

```
this.fig.Units = 'normalized';
this.fig.Name = 'Artillery';
this.fig.MenuBar = 'none';
this.fig.ToolBar = 'none';
this.fig.NumberTitle = 'off';
```

Spielfeld plazieren

```
this.fig.Position = this.gameStates.GAME_POSITION;
this.fig.Color = this.gameStates.BLACK;
```

Fullscreen erstellen

```
% Quelle
% http://stackoverflow.com/questions/15286458/
automatically-maximize-figure-in-matlab
% http://www.mathworks.com/matlabcentral/answers/98331-
is-it-possible-to-maximize-minimize-or-get-the-state-of-my-figure-
programmatically-in-matlab
% vom 28.12.2015
 this.fig.Visible = 'on';
 pause(0.1)
 jFrame = get(handle(this.fig), 'JavaFrame');
 jFrame.setMaximized(1);
 pause(0.1)
 this.fig.Resize = 'off';
```

Eigener Mousepointer entwerfen

Inspiriert durch http://www.mathworks.com/matlabcentral/newsreader/view_thread/58453

```
pointer = NaN(16, 16);
pointer(8, 1:16) = 1;
pointer(1:16, 8) = 1;
pointer(8, 8) = 2;
this.fig.Pointer = 'Custom';
this.fig.PointerShapeHotSpot = [4, 4];
this.fig.PointerShapeCData = pointer;
this.fig.PointerShapeCData = pointer;

% Axishandler erstellen
mainAxis = axes();
axis(this.gameParameter.axisArray);
%Resize der Achsen verhindern
axis manual;

% Farbe für Spielfeld erstellen.
% Himmelblau machen
```

```
            set(mainAxis, 'color', AXIS_COLOR, 'YTick', [], 'XTick',
[]);
            % Titel erstellen
            axisTitle = title('Artillery');
            set(axisTitle, 'FontName', FONT,'FontSize', LARGE_TEXT);
            set(axisTitle, 'Color', TITLE_COLOR);

            %FarbSchema erstellen Joel Koch 4.1.16: (hier) nicht
notwendig.
            %colormap(0.4*summer+0.4*flipud(pink)+0.1*flipud(winter));

            % Verhindern Das der Bildschirm verändert wird
            this.fig.Resize = 'off';
            hold on;

        end
```

Zeichnen Spieler welcher am zug ist auf das Spielfeld

```
        % Zweck: Zeichnen eines Textes der Angibt, welcher Spieler am
Zug
        % ist
        % Pre: Instanz erstellt, Spielfeld erstellt
        % Post: test ist gezeichnet
        %
        % Input:    this Figure Instant, Instanzvariabeln
        % GameState -- Status Instanz
        % color -- farbe des Spieler
        %
        % Output: void
        %
        function [] = drawActualPlayer(this, GameState, color)

            % Plaziervariabeln
            % hier werden die Plaziervariabeln berechnet, wo komt der
Text
            % hin
            axes = this.gameParameter.axisArray;
            leftBorder = (this.gameStates.SCREEN_WIDTH -
(axes(1,2)+100)) / 2 ;
            fromleftBorder = 10;
            fromTop = this.gameStates.SCREEN_HIGH - 160;
```

Player Text wird erstellt und parametriert

```
            this.player = uicontrol;
            this.player.Style = 'text';
            this.player.String =  ['Player >> ',
num2str(GameState.getActualPlayer)];
            this.player.Position = [fromleftBorder, fromTop,
leftBorder , 50];
```

```
            this.player.ForegroundColor = color;
            this.player.BackgroundColor = this.gameStates.BLACK;
            this.player.FontName = this.gameStates.FONT;
            this.player.FontSize = this.gameStates.TEXT_SIZE ;

        end
```

Zeichnen Spieler welcher am zug ist auf das Spielfeld

```
        % Zweck: Zeichnen eines Textes für jeden Spieler, der Angibt,
        % welcher Spieler wieviele Siegespunkte hat.
        %
        % Pre: Instanz erstellt, Spielfeld erstellt, Spieler erstellt
        % Post: test ist gezeichnet
        %
        % Input:     this Figure Instant, Instanzvariabeln
        % GameParameter -- Parameter Instanz
        % Player -- Array mit Spieler instanzen
        %
        % Output: void
        %
        function [] = drawPlayerPoints(this, GameParameter, Player)
```

dies sind die Plaziervariabeln um die Texte auf dem Spielfeld zu plazieren

```
            axes = this.gameParameter.axisArray;
            leftBorder = (this.gameStates.SCREEN_WIDTH -
(axes(1,2)+100)) / 2 ;
            fromleftBorder = 10; % plaziern von der linken kante
entfernt
            fromTop = this.gameStates.SCREEN_HIGH -
200; %startposition von oben
            high = 50;  % höhe der texte
            space = 25; % Abstand zwischen den Texten
```

Player Points

in dieser Schleife wird für jeden Spieler ein uicontroll text mit der aktuellen Punktzahl erstellt

```
            for pk = 1 : 1 : GameParameter.playerQuantety
                playerPointsloc(pk) = uicontrol;
                playerPointsloc(pk).Style = 'text';
                playerPointsloc(pk).String =  ['Player ',
num2str(pk), ' >> ', num2str(Player(pk).score)];
                playerPointsloc(pk).Position = [fromleftBorder,
fromTop - space * pk, leftBorder , high];
                playerPointsloc(pk).ForegroundColor =
Player(pk).getTankColor;
```

```
                playerPointsloc(pk).BackgroundColor =
this.gameStates.BLACK;
                playerPointsloc(pk).FontName = this.gameStates.FONT;
                playerPointsloc(pk).FontSize =
this.gameStates.TEXT_SIZE_TINY ;
            end
                % Speichern in Instanzvariabel
                this.playerPoints = playerPointsloc;

        end
```

Zeichnen Aktuelle Spielrunde auf das Spielfeld

Zweck: Zeichnen eines Textes für die Spielrunde, der angibt, welche Spielrunde aktuell am laufen ist

```
        % Pre: Instanz erstellt, Spielfeld erstellt, Spieler erstellt,
        % Runde gestartet
        % Post: text ist gezeichnet
        %
        % Input:    this Figure Instant, Instanzvariabeln
        % GameParameter -- Parameter Instanz
        % GameState -- Status instanz
        %
        % Output: void
        %
        function [] = drawGameRound(this, GameParameter, GameState)
```

dies sind die Plaziervariabeln um die Texte auf dem Spielfeld zu plazieren

```
            axes = this.gameParameter.axisArray;
            width = (this.gameStates.SCREEN_WIDTH - (axes(1,2)+100));
            fromleftBorder = 10;
            fromTop = this.gameStates.SCREEN_HIGH - 110;
            high = 30;
```

Spielrunde Zeichnen

```
            for pk = 1 : 1 : GameParameter.playerQuantety
                this.gameRound = uicontrol;
                this.gameRound.Style = 'text';
                this.gameRound.String =  ['Round
>> ', num2str(GameState.getGameRound), ' of ',
num2str(GameParameter.numberRounds)];
                this.gameRound.Position = [fromleftBorder, fromTop,
width , high];
                this.gameRound.ForegroundColor =
GameState.TITLE_COLOR;
                this.gameRound.BackgroundColor =
this.gameStates.BLACK;
```

```
                this.gameRound.FontName = this.gameStates.FONT;
                this.gameRound.FontSize =
this.gameStates.TEXT_SIZE_SMALL;
            end

        end
```

Zeichnen der Gamebutton für den Taktik Mode auf das Spielfeld

Zweck: Zeichnen aller für den Taktikmode benötigten Spielbuttons auf das Spielfeld erstllt die jewiligen Eventhandler zu ein ausgabe

```
        % Pre: Instanz erstellt, Spielfeld erstellt, taktik mode,
        % Runde gestartet
        % Post: alle Buttons sind gezeichnet ist gezeichnet, alle
        % Eventhandler sind erstellt
        %
        % Input:    this Figure Instant, Instanzvariabeln
        %
        % Output: void
        %
        function [] = drawGameButtons(this)
```

dies sind die Plaziervariabeln um die Texte auf dem Spielfeld zu plazieren

```
            buttonWidth = 150;
            buttonHigh = 50;
            fromleftBorder = (this.gameStates.SCREEN_WIDTH -
buttonWidth)/2 ;
            fromBot = 10;
```

Feuer Befehl button erstellen und parametrieren

eventhandler erstellen

```
            this.fire = uicontrol;
            this.fire.Style = 'pushbutton';
            this.fire.String = '!!!FIRE!!!';
            this.fire.Position = [fromleftBorder, fromBot ,
buttonWidth , buttonHigh];
            this.fire.ForegroundColor = this.gameStates.RED;
            this.fire.BackgroundColor = this.gameStates.BLACK;
            this.fire.FontName = this.gameStates.FONT;
            this.fire.FontSize = this.gameStates.TEXT_SIZE;
            this.fire.Callback = @this.btnFireClick;
```

WinkelText erstellen und parametrieren

```
            this.angleText = uicontrol;
            this.angleText.Style = 'text';
            this.angleText.String = 'Angle >> ';
            this.angleText.Position = [(fromleftBorder - buttonWidth -
30), fromBot + buttonHigh, buttonWidth , buttonHigh];
            this.angleText.ForegroundColor = this.gameStates.YELLOW;
            this.angleText.BackgroundColor = this.gameStates.BLACK;
            this.angleText.FontName = this.gameStates.FONT;
            this.angleText.FontSize = this.gameStates.TEXT_SIZE ;
```

Winkel Slider erstellen und parametrieren

eventhandler erstellen

```
            this.angleSlider = uicontrol;
            this.angleSlider.Style = 'slider';
            this.angleSlider.String = 'ANGLE';
            this.angleSlider.Position = [(fromleftBorder - buttonWidth
- 30), fromBot , buttonWidth , buttonHigh];
            this.angleSlider.ForegroundColor = this.gameStates.GREEN;
            this.angleSlider.BackgroundColor = this.gameStates.BLACK;
            this.angleSlider.FontName = this.gameStates.FONT;
            this.angleSlider.FontSize = this.gameStates.TEXT_SIZE;
            this.angleSlider.Callback = @this.btnAngleClick;
```

Power Text erstellen und parametrieren

```
            this.powerText = uicontrol;
            this.powerText.Style = 'text';
            this.powerText.String = 'POWER >> ';
            this.powerText.Position = [fromleftBorder + buttonWidth +
30, fromBot + buttonHigh, buttonWidth , buttonHigh];
            this.powerText.ForegroundColor = this.gameStates.ORANGE;
            this.powerText.BackgroundColor = this.gameStates.BLACK;
            this.powerText.FontName = this.gameStates.FONT;
            this.powerText.FontSize = this.gameStates.TEXT_SIZE;
```

Power Slider erstellen und parametrieren

eventhandler erstellen

```
            this.powerSlider = uicontrol;
            this.powerSlider.Style = 'slider';
            this.powerSlider.String = 'POWER';
            this.powerSlider.Position = [fromleftBorder + buttonWidth
+ 30, fromBot ,buttonWidth,buttonHigh];
            this.powerSlider.ForegroundColor = this.gameStates.GREEN;
            this.powerSlider.BackgroundColor = this.gameStates.BLACK;
            this.powerSlider.FontName = this.gameStates.FONT;
```

```
            this.powerSlider.FontSize = this.gameStates.TEXT_SIZE;
            this.powerSlider.Callback = @this.btnPowerClick;

        end
```

Zeichnen der Powerbar für den Geschiklichkeitsmodus

Zweck: Zeichnen aller für den Geschiklichkeitsmodus benötigten Spielelemente auf das Spielfeld erstllt die jewiligen Eventhandler zu ein / ausgabe

Pre: Instanz erstellt, Spielfeld erstellt, Geschiklichkeitsmodus, Runde gestartet Post: der powerbar ist gezeichnet, alle Eventhandler sind erstellt

Input: this Figure Instant, Instanzvariabeln

Output: void

```
        function [] = drawPowerBar(this)

            this.updatePowerBar(0); % ruft update mit wert null = leer
 auf
```

eventhandler werden erstellt

```
            set(this.fig,'WindowButtonDownFcn',
 @this.myMouseDownCallBack)
            set(this.fig,'WindowButtonUpFcn', @this.myMouseUpCallBack)

        end
        function [] = updatePowerBar(this,power)
```

zeichnet die powerbar

```
            blueX = [350,650,650,350];
            blueY = [700,700,720,720];
            pbarX = [350, 350+300*min(power,1), 350+300*min(power,1),
 350];
            pbarY = blueY;
            patch(blueX,blueY,[0.6 0.9 1]); % Hellblau Himmel;
            patch(pbarX,pbarY,'R');

        end
```

Zeichnen der Landscape im Spielfeld

Zweck: Zeichnen des Landschaftsarrays im spielfeld

```
        % Pre: Instanz erstellt, Spielfeld erstellt,
        %
        % Post: Die Landschaft ist gezeichnet
```

```
        %
        % Input:  this Figure Instant, Instanzvariabeln
        % terrain -- array mit den aktuellen
        % Landschaftskoordinaten
        %
        % Output: void
        %
        function [p] = drawInScreen(this,terrain)
            shapeX = terrain(1,:); % Xwerte des Arrays
            shapeY = terrain(2,:); % Ywerte des Array
            shapeC = terrain(2,:); % Cwerte des Array
            p = patch(shapeX,shapeY,
 shapeC,'EdgeColor','interp','MarkerFaceColor','flat'); % Zeichnen
            colormap(0.4*summer
+0.4*flipud(pink)+0.1*flipud(winter)); % Farbe
            axis(this.gameParameter.axisArray); % Achsen definieren
            this.terrainhandler = p; % Terrainhandler speichern
            uistack(this.terrainhandler, 'bottom') % Landscahft in den
 Hintergrund setzen
        end
        function [p] = updateInScreen(this, terrain)
            this.terrainhandler.delete; % Update Funktion löscht
 zusätlich die Alte Landschaft
             p = this.drawInScreen(terrain);
        end
```

Zeichnen eines [x,y] Arrays im bildschirm

Zweck: Zeichnen eines beliebigen [x,y] Arrays im Spielfeld (Bsp.: Tank, Wtterpfeil)

```
        % Pre: Instanz erstellt, Spielfeld erstellt, shape erstellt
        %
        % Post: das shape ist gezeichnet, ein shapehndeler wurde
        % zurückgegeben
        % Das shape wird in Schwarz gezeichnet
        %
        % Input:    this Figure Instant, Instanzvariabeln
        % shape -- [x,y] Koordinaten
        %
        % Output: shapehandler
        %
        function [p] = drawElement(this, shape)
                color = [0 0 0]; %Schwarze farbe definieren
                p = this.drawElementCol(shape, color);
        end
```

Wie drawElement, mit zusätzlicher farbe definieren

```
        function [p] = drawElementCol(this,shape,color)
             polygonX = shape(1,:);
             polygonY = shape(2,:);
```

```
            p = patch(polygonX, polygonY, color);% Shape wird
gezeichnet
        end
```

Löschen eines Shape

Zweck: Löscht ein bereits erstelltes shape

```
        % Pre: Instanz erstellt, Spielfeld erstellt, shape gezeichnet,
        % Shapehandler vorhanden
        %
        % Post: das shape ist gelöschtt
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        % p -- shapehandler
        %
        % Output: void
        %
        function [] = deleteElement(this, p)
             p.delete;
        end
```

Update eines Shape

Zweck: Löscht ein bereits erstelltes shape und zeichet dieses neu

```
        % Pre: Instanz erstellt, Spielfeld erstellt, shape gezeichnet,
        % Shapehandler vorhanden
        %
        % Post: das shape ist gelöscht und neu mit der selben Farbe
        % gezeichnet
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        % shape -- [x,y] Koordinaten
        % p -- shapehandler
        %
        % Output: void
        %
        function [p] = updateElement(this, p, shape)
            color = p.FaceColor; % Shapefarbe Speichern
            this.deleteElement(p); % Shape Löschen
            p = this.drawElementCol(shape,color); % Shape Zeichnen
        end
```

wie updateElement, jedoch mit zusätlicher Farbvariabel

```
        function [p] = updateElementCol(this,p,shape,color)
            this.deleteElement(p);
            p = this.drawElementCol(shape,color);
        end
```

Zeichnen einer Shockwelle

Zweck: Zeichnet eine shockwelle an einer stelle X/Y

```
        % Pre: Instanz erstellt, Spielfeld erstellt, shuss berechnet,
        % einschlagkoordinaten vorhanden
        %
        % Post: auf dem Spielfeld wurde eine Shockwelle angezeigt
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        % impact -- [x,y] Koordinaten des einschlages
        %
        % Output: void
        %
        function [] = drawShockwave(this, impact)

            r = this.gameParameter.detonationRadius; % Definition des
Explosionsradius

            % rechnet den explosionsradius und die explosion, macht
einen fadeout
            % löscht sie wieder, bevor die funktion zurückkehrt
            alpha = linspace( 0,2*pi,100);          % Intervall
            shockX=3*r*cos(alpha)+impact(1,1);      % Kreis
            shockY=3*r*sin(alpha)+impact(2,1);      % Kreis
            blast2 = patch(shockX,shockY,'w');      % Kreis zeichnen,
handler=blast2
            blast2.LineStyle='none';                % Kreislinie nicht
zeichnen
            uistack(this.terrainhandler, 'top')     % terrain nach
ganz vorne bringen, Blast Radius ist nur in der Luft.

            shockX=0.99*r*cos(alpha)+impact(1,1);
            shockY=0.99*r*sin(alpha)+impact(2,1);
            blast0=patch(shockX,shockY,[0.6 0.9 1]);% Hellblau Himmel;
 % hintergrund "patchen" mit hellblau, das richtige loch wird erst
später reingegerechnet.
            blast1=patch(shockX,shockY,'r');        % roter
Explosionsradius darüber zeichnen
            blast0.LineStyle='none';
            blast1.LineStyle='none';

            ptime = 0.015;                          % pause zeit
zwischen den animationsschritten
            for fadesteps = 0.5:-0.03:0             % animation
deckkraft von 0.5 bis 0
                pause(ptime);
                blast1.FaceAlpha=max(0,fadesteps*8-3);% der
Explosionsradius rot klingt schneller ab
                blast2.FaceAlpha=fadesteps;
            end
            delete(blast2);                         % Die Animation
ist fertig, alle
```

```
            delete(blast1);                          % benutzten
Elemente löschen
            delete(blast0);

            uistack(this.terrainhandler, 'bottom')  % Terrain wieder
nach hinten bringen
        end
```

Zeichnen und animieren eines neuen Terrains

Zweck: Zeichnet an einer Einschlagstelle einen Krater

```
        % Pre: Instanz erstellt, Spielfeld erstellt, schuss berechnet,
        % einschlagkoordinaten vorhanden, terrain vorhanden
        %
        % Post: auf dem Spielfeld wurde ein Krater gezeichnet das neue
        % Terrain wurde zurückgegeben
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        % impact -- [x,y] Koordinaten des einschlages
        % terrain -- [x,y] KoordinatenArray des Terrains
        %
        % Output: newTerrain -- [x,y] KoordinatenArray des neu
moddelierten Terrains
        %
        function [newTerrain] = drawImpactCircle(this, terrain,
impact)
            % Array speichern
            x1arr = terrain(1,:);
            y1arr = terrain(2,:);
            centerX = impact(1,1);
            centerY = impact(2,1);

            % RadiusVariabel des einschalg Kraters
            r = this.gameParameter.detonationRadius;

            % Landschaftspunkte ausserhalb des Radius holen:
            intersections = this.getOuterIntersections(x1arr, y1arr,
centerX, centerY, r);

            % Für die Verständlichkeit namen vergeben:
            outerLeftX  =   x1arr(intersections(1));
            outerLeftY  =   y1arr(intersections(1));
            outerRightX =   x1arr(intersections(2));
            outerRightY =   y1arr(intersections(2));
            craterSteps =   intersections(2)-intersections(1)-1;
            craterSteps =   20;   % Anzahl Koordinatenpaare für den
Bogen

            % Kreissegment Start und Endpunkte rechnen in Bogenmass
            % mit complex-zahlen machen, sonst gibts noch
            % QuadrantenFallunterscheidung! Dazu muss aber der
Einschlagpunkt die
```

```
            % Koordinate 0+0i haben
            z= outerLeftX - centerX + (outerLeftY-centerY) * i;
            circleStart=angle(z);

            z= outerRightX - centerX + (outerRightY-centerY) * i;
            circleEnd=angle(z);

            % der Bogen muss zwingend im Gegenuhrzeigersinn erfolgen,
sonst gibts
            % Hügel und andere Fehler.
            if circleEnd<circleStart
                circleEnd=circleEnd+2*pi;
            end

            % Schockwelle zeichnen (Animation, dauert einen Moment)
            this.drawShockwave(impact)

            % rechnet den Explosions-Bogen in n=craterSteps Schritten
            phi=linspace(circleStart, circleEnd, craterSteps);
            arcX=r*cos(phi);
            arcY=r*sin(phi);
            x =arcX + centerX;
            y =arcY + centerY;

            % Terrain-Matrizen anpassen, ein Stück davon ersetzen
            partXbefore = x1arr(1:intersections(1));
            partXafter = x1arr(intersections(2):end);
            partYbefore =y1arr(1:intersections(1));
            partYafter = y1arr(intersections(2):end);
            terrainshapeX = [partXbefore, x, partXafter];
            terrainshapeY = [partYbefore, y, partYafter];
            newTerrain = [terrainshapeX; terrainshapeY];
        end
```

Rechnen der Outer Intersections

Zweck: Rechnet die Outerintersections

```
        % Pre: Instanz erstellt, Spielfeld erstellt, schuss berechnet,
        % einschlagkoordinaten vorhanden, terrain vorhanden
        %
        % Post:     die Intersections sind berechnet
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        % x1arr -- X KoordinatenArray des Terrains
        % y1arr -- Y KoordinatenArray des Terrains
        % centerX -- X Koordinaten des Einschlags
        % centerY -- Y Koordinaten des Einschlags
        % r -- Radius des kraters
        % Output: intersections --
        %
        function [intersections] = getOuterIntersections(this, x1arr,
y1arr, centerX, centerY, r)
```

```
            % Erstelle Array mit distanzen zum Kreismittelpunkt
            % normaler pythagoras
            distance=(((x1arr - centerX).^2 + (y1arr-
centerY).^2).^0.5);

            %finde den ersten Punkt *vor* dem Radius (Krater soll
 nicht
            %eingzackt sein, sondern eher gegen aussen gerissen.
            withinRadius=distance<r;
            isect1=max(1,find(withinRadius,1,'first')-1);
            %finde den letzten Punkt *nach* dem Radius
            isect2=min(size(x1arr,2),find(withinRadius,1,'last')+1);
            intersections = [isect1, isect2];
        end
```

Speichern / Update des Statusobjektes

Zweck: Speicher ein neues Statusobjekt

```
        % Pre: Instanz erstellt, Statusobjekte vorhanden
        %
        % Post:     das neue Statusobjekt wurde gespeichert
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        % GameStates -- Status Instanz
        % Output: void
        %
        function [] = updateState(this,GameStates)
           this.gameState = GameStates;
        end
```

Speichern / Update des Prameterobjektes

Zweck: Speicher ein neues Prameterobjekt

```
        % Pre: Instanz erstellt, Prameterobjekt vorhanden
        %
        % Post:     das neue Prameterobjekt wurde gespeichert
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        % GameParameter -- Status Instanz
        % Output: void
        %
        function [] = updateParameters(this,GameParameter)
           this.gameParameter = GameParameter;
        end
```

gibt das Prameterobjektes aus

Zweck: Erhalten eines durch das Menue modifizierte Prameterobjektes

```
        % Pre: Instanz erstellt, Prameterobjekt vorhanden
        %
```

```
        % Post:     das Parameterobjekurde wurde ausgegeben
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
        % Output:   GameParameter -- Instanz der GameParameter Klasse
        %
        function [GameParameter] = getParameters(this)
            GameParameter = this.gameParameter;
        end
```

Gibt den figurehandler der aktuellen Figure Instanz zurück

Zweck: erhalten eines figurehandlers zum Manipulieren (löschen etc,)

```
        % Pre: Instanz erstellt, fig handler vorhanden
        %
        % Post:     das Parameterobjekurde wurde ausgegeben
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
        % Output:   fig -- figurehandler
        %
        function [fig] = getFig(this)
            fig = this.fig;
        end
```

Rückgabe aktuellen Wert des Powerslider

Zweck: Auslesen des wertes des powersliders und Rückgabe des Wertes

```
        % Pre: Instanz erstellt, Gamebuttons sind, erstellt, Taktik
 Mode,
        %
        % Post:     der Powerwert wurde zurückgegeben
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
        % Output:   power -- Wert für power (double)
        %
        function [power] = getPower(this)
            power = this.powerSlider.Value *
 this.gameParameter.maxPower;
        end
```

Rückgabe aktuellen Wert des AngleSlider

Zweck: Auslesen des wertes des AngleSlider und Rückgabe des Wertes

```
        % Pre: Instanz erstellt, Gamebuttons sind, erstellt, Taktik
 Mode,
```

```
        %
        % Post:     der Anglewert wurde zurückgegeben in Grad
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
        % Output:   angle -- Wert für angle in Grad (double)
        %
        function [angel] = getAngle(this)
            angel = this.angleSlider.Value *
 this.gameParameter.maxAngle;
        end
```

Eventhandler btnPlayerCountClick

Zweck: Manipulation der Spieleranzah

```
        % Pre: Instanz erstellt, fig handler vorhanden, Gamebuttons
 sind
        % erstellt, Enthandler ist definiert, Menue gezeicnet
        %
        % Post:     Die Spieleranzahl ist in den Parametern
 manipuliert
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
        % Output:   void
        %
        function btnPlayerCountClick(this,source,eventdata)
            if this.gameParameter.playerQuantety <
 this.gameParameter.maxPlayerQuantety
            this.gameParameter =
 this.gameParameter.setPlayerQuantety(this.gameParameter.playerQuantety
+1);
            else
            this.gameParameter =
 this.gameParameter.setPlayerQuantety(2);
            end
            this.btnPlayerCount.String = ['N off Players >> ',
 num2str(this.gameParameter.playerQuantety)];
        end
```

Eventhandler btnGameModeClick

Zweck: Manipulation der Gamemodus

```
        % Pre: Instanz erstellt, fig handler vorhanden, Gamebuttons
 sind
        % erstellt, Enthandler ist definiert, Menue gezeicnet,
        %
        % Post:     Die Gamemodus ist in den Parametern manipuliert
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
```

```
        % Output:   void
        %
        function btnGameModeClick(this,source,eventdata)
            this.gameParameter = this.gameParameter.nextMode;
            this.btnMode.String = this.gameParameter.gameMode;
        end
```

Eventhandler btnWindClick

Zweck: Manipulation der Windstärke

```
        % Pre: Instanz erstellt, fig handler vorhanden, Gamebuttons
 sind
        % erstellt, Enthandler ist definiert, Menue gezeichnet,
        %
        % Post:     Die Windstärke ist in den Parametern manipuliert
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
        % Output:   void
        %
        function btnWindClick(this,source,eventdata)
            this.gameParameter = this.gameParameter.nextWind;
            this.btnWind.String = this.gameParameter.wind;
        end
```

Eventhandler btnMountainClick

Zweck: Manipulation der Berghöhe

```
        % Pre: Instanz erstellt, fig handler vorhanden, Gamebuttons
 sind
        % erstellt, Enthandler ist definiert, Menue gezeichnet,
        %
        % Post:     Die Berghöhe ist in den Parametern manipuliert
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
        % Output:   void
        %
        function btnMountainClick(this,source,eventdata)
            this.gameParameter = this.gameParameter.nextMountain;
            this.btnMountain.String = this.gameParameter.mountain;
        end
```

Eventhandler btnPlanetClick

Zweck: Manipulation der Planteeinstellung

```
        % Pre: Instanz erstellt, fig handler vorhanden, Gamebuttons
 sind
        % erstellt, Enthandler ist definiert, Menue gezeichnet,
```

```
        %
        % Post:     Die Planteeinstellung ist in den Parametern
manipuliert
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
        % Output:   void
        %
        function btnPlanetClick(this,source,eventdata)
            this.gameParameter = this.gameParameter.nextPlanet;
            this.btnPlanet.String = this.gameParameter.planet;
        end
```

Eventhandler sldRoundsChange

Zweck: Manipulation der Anzahl Runden

```
        % Pre: Instanz erstellt, fig handler vorhanden, Gamebuttons
sind
        % erstellt, Enthandler ist definiert, Menue gezeichnet,
        %
        % Post:     Die Anzahl Runden ist in den Parametern
manipuliert
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
        % Output:   void
        %
        function sldRoundsChange(this,source,eventdata)
            this.gameParameter.numberRounds =
round(this.sldRounds.Value * 99)+1;
            this.txtRounds.String = ['Rounds >> ',
num2str(this.gameParameter.numberRounds)];
        end
```

Eventhandler btnStartClick

Zweck: Menue Schliessen und Starten des Spiels

```
        % Pre: Instanz erstellt, fig handler vorhanden, Gamebuttons
sind
        % erstellt, Enthandler ist definiert, Menue gezeichnet,
        %
        % Post:     Das Menue ist geschlossen
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
        % Output:   void
        %
        function btnStartClick(this,source,eventdata)
            this.gameStates.setMenueProccessed(1);
            close(this.fig)
        end
```

Eventhandler btnFireClick

Zweck: Feuerbefehl des Taktik Modus ausgeben

```
        % Pre: Instanz erstellt, Gamebuttons sind erstellt, Enthandler
 ist
        % definiert, Speilfeld gezeichnet,
        %
        % Post:     Der Fireevent wurde gesetzt
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
        % Output:   void
        %
        function [] = btnFireClick(this,source,eventdata)
            this.fireEvent = 1;
        end
```

Eventhandler btnAngleClick

Zweck: Einstellen des Winkelwertes und ausgeben auf Spielfeld

```
        % Pre: Instanz erstellt, Gamebuttons sind erstellt, Enthandler
 ist
        % definiert, Speilfeld gezeichnet,
        %
        % Post:     Der Winkel wurde verändert und der neue wert
 ausgegeben
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
        % Output:   void
        %
        function btnAngleClick(this,source,eventdata)
            value = this.angleSlider.Value *
 this.gameParameter.maxAngle;
            this.angleText.String = ['Angle >>   ', num2str(value)];
        end
```

Eventhandler btnPowerClick

Zweck: Einstellen des Powerwertes und ausgeben auf Spielfeld

```
        % Pre: Instanz erstellt, Gamebuttons sind erstellt, Enthandler
 ist
        % definiert, Speilfeld gezeichnet,
        %
        % Post:     Die power wurde verändert und der neue Wert
 ausgegeben
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
```

```
        % Output:   void
        %
        function btnPowerClick(this,source,eventdata)
            value = this.powerSlider.Value *
 this.gameParameter.maxPower;
            this.powerText.String = ['Power >>   ', num2str(value)];
        end
```

Eventhandler myMouseDownCallBack

Zweck: Zeigt an das die Muas gedrückt wird. Speicher die Koordinaten des Mauszeigers

```
        % Pre: Instanz erstellt, Gamebuttons sind erstellt, Enthandler
 ist
        % definiert, Speilfeld gezeichnet,
        %
        % Post:     Die Variable mousedown wird auf eins gesetzt
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
        % Output:   void
        %
        function myMouseDownCallBack(this,hObject,~)
            % quelle: http://stackoverflow.com/questions/2769249/
matlab-how-to-get-the-current-mouse-position-on-a-click-by-using-
callbacks
            % set(f,'WindowButtonDownFcn',@mytestcallback)
            % function mytestcallback(hObject,~)
            % pos=get(hObject,'CurrentPoint');
            % disp(['You clicked X:',num2str(pos(1)),',
 Y:',num2str(pos(2))]);
            % end
            % Quelle :http://stackoverflow.com/questions/14684577/
matlab-how-to-get-mouse-click-coordinates
            % set(imageHandle,'ButtonDownFcn',@ImageClickCallback);
            % function ImageClickCallback ( objectHandle , eventData )
            % axesHandle  = get(objectHandle,'Parent');
            % coordinates = get(axesHandle,'CurrentPoint');
            % coordinates = coordinates(1,1:2);
            % message     = sprintf('x: %.1f , y: %.1f',coordinates
 (1) ,coordinates (2));
            % helpdlg(message);
            % end
                mouseposition = get(gca, 'CurrentPoint');
                this.mouseX   = mouseposition(1,1);
                this.mouseY  = mouseposition(1,2);
                this.mousedown = 1;
        end
```

Eventhandler myMouseUpCallBack

Zweck: Zeigt an das die Muas losgelassenwird. Speicher die Koordinaten des Mauszeigers

```
        % Pre: Instanz erstellt, Gamebuttons sind erstellt, Enthandler
 ist
        % definiert, Speilfeld gezeichnet,
        %
        % Post:     Die Variable mousedown wird auf 0 gesetzt
        %
        % Input:    this Figure Instanz, Instanzvariabeln
        %
        % Output:   void
        %
        function myMouseUpCallBack(this,hObject,~)
                mouseposition = get(gca, 'CurrentPoint');
                this.mouseX  = mouseposition(1,1);
                this.mouseY  = mouseposition(1,2);
                this.mousedown = 0;
        end

    end
end
```

Published with MATLAB® R2015b