# EIA2 – Endabgabe – Konzept – Marc Siegfried

Gemüsegarten-Simulator - Anforderungen:

* Desktop und Smartphone Kompatibilität
* Mindestens 40 Felder vorhanden
* Mindestens 5 verschiedene Gemüse-Seeds vorhanden (Corn, Potato, Carrot, Tomato, Mushroom)
* Zustand der Pflanzen werden grafisch oder in Zahlen angezeigt
* Gemüse wächst und kann geerntet werden (verkauft werden)
* Geld verdienen (Gemüse verkaufen)
* Auswahlmöglichkeiten für auszuführende Aktionen (pflanzen, gießen, düngen, ernten, Schädling bekämpfen)
* Ausgewählte Aktion wird durch Interaktion mit dem Gartenfeld (der Pflanze) ausgelöst
* Pflanzen haben unterschiedliche Wachstumszeiten und Bedarfe bezüglich Wasser und Dünger
* Anzeige von den Marktpreisen für das geerntete Gemüse, für Setzlinge, Dünger, Pestizide etc., sowie das zur Verfügung stehende Kapital für die Pflege des Gartens.
* Produkte wie Seeds, Wasser, Fertilizer oder Pestizide können mit dem verdienten Geld gekauft werden
* Schädlinge zerstören die Ernte
* Simulationszeit wird deutlich beschleunigt (Pest nimmt in Realzeit zu)

Zusatzfunktionen:

* Energieleiste, welche einzelne Tätigkeiten begrenzen soll. Jede Tätigkeit (wie bspw. pflanzen, bewässern, usw…) hat seinen eigenen Arbeitsaufwand und somit einen individuellen Energiekonsum je nach Aufwand der Tätigkeit.
* Wetter, welches Einfluss auf die Ernte, also auf die einzelnen Parameter der Pflanzen hat und auch visuell dargestellt wird.
* Tote Pflanzen durch fehlende oder falsche Pflege, zu starkem Schädlingsfraß, werden in diesem Zustand angezeigt.

Texturen - Pflanzen und Aktionen:



Potato

Carrot

Mushroom

Tomato

Corn



Pflanzenzustände Corn:



Pflanzenzustände Potato:



Pflanzenzustände Carrot:



Pflanzenzustände Tomato:



Pflanzenzustände Mushroom:



Aktionen:

Water

Fertilize

Pesticide

Harvest

Weed

Plow



Wetter: Cursor:



Weeds: Ground:



Pflanzen und Aktionen - Info:



* Kaufpreis: 1
* Energieverbrauch: 10 (Niedrig)
* Verkaufspreis: 0-6







* Kaufpreis: 2
* Energieverbrauch: 15 (Mittel)
* Verkaufspreis: 0-9







* Kaufpreis: 2
* Energieverbrauch: 15 (Mittel)
* Verkaufspreis: 0-8







* Kaufpreis: 2
* Energieverbrauch: 12 (Mittel)
* Verkaufspreis: 0-7







* Kaufpreis: 1
* Energieverbrauch: 10 (Niedrig)
* Verkaufspreis: 0-6









Plowing

* Energieverbrauch: 20 (Hoch)
* Funktion: Plow and make ridges



Weeding

* Energieverbrauch: 15 (Mittel)
* Funktion: Weeding weeds





Harvesting

* Energieverbrauch: 15 (Mittel)
* Funktion: Harvest and sell crops





Watering

* Energieverbrauch: 5 (Niedrig)
* Funktion: Water soil







**Fertilizing**

* **Energieverbrauch: 5 (Niedrig)**
* **Funktion: Helps crops grow**
* **Kaufpreis: 3**





Pesticiding

* Energieverbrauch: 10 (Niedrig)
* Funktion: Keeps harvest safe from parasites
* Kaufpreis: 10



Simulator Ablauf - Wachstum:

Tote Pflanze

Wachstum = 2.0

Bepflanztes

Feld (Potato)

Wachstum = 0.0

Wachstum

0.25

Wachstum

1.0

Wachstum

0.75

Wachstum

0.5

Leeres Feld



Erntreife

Potatoes

Leeres Feld

Bepflanztes Feld



Normales Wachstum (Zufällige Qualität)

low humidity **↓**

0 humidity



Keine Aktion



Tote

Pflanze

low fertility **↓**



high pest **↑**

100 pest

Keine Aktion

Keine Aktion



0 fertility

Erntreife Potatoes

(low quality)

Erntreife Potatoes

(high quality)

100 fertility



high fertility **↓**



Fertilizing



Use-Case:

**FARMER**



UI-Scribble - Settings:

<h1>

<input

type=range

name=startPrice

min=”5” max=”200”

step=”5”>

Gemüsegarten-Simulator - Settings

<h1>

**Startkapital**:

**1000**

**2000**

**5**

\\

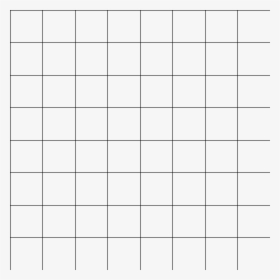
<fieldset>

c

**Preisschwankung**:

\\

<p>

c

<p>

<img src=”toolicon”>

Water

Weed

Select

Fertilize

Pesticide

Harvest

Mushroom

Tomato

Carrot

Potato

Corn

Plow

Working Power:

Money:

Weather cond.:

Sunny

Position:

Object:

Humidity:

Pesticide:

Fertility:

Pest:

Object growth:

Object quality:

Object value:

<img

src=”credit.png”>

<div id=”stage”>

overlay.addChild(new Button(img, name)...

<div id=”gstatuspanel”>

<img

src=”titlel.png”>

<div id=”buttonInfo”>

\\

<button

id=”start”>

<input

type=range

name=priceChange

min=”1” max=”3”

step=”1”>

START

20x20 Felder

ClassDiagramm:

FarmGame

xs: any;

ys: any;

cells: string = [ ];

workingPower: number = 100;

cash: number = startMoney;

weather: number = 0;

time: number = 0.5;

frameCount: number = 0;

Cell

weedRoots: number = 0.5;

plowed: boolean = false;

crop: any = null;

humidity: number = 0.5;

pesticide: number = 0;

potatoPest: number = 0;

fertility: number = 0.5;

serialize( ): any;

deserialize(data: any): void

plow( ): boolean;

seeds( ): boolean;

harvest( ): boolean;

water( ): boolean;

weeding( ): boolean;

pesticide( ): boolean;

fertilize( ): boolean;

Cell

weedRoots: number = 0.5;

plowed: boolean = false;

crop: any = null;

humidity: number = 0.5;

pesticide: number = 0;

potatoPest: number = 0;

fertility: number = 0.5;

...

Potato

grow(cell: any, growth: any): void

eval( ): number

getQuality( ): number

type: string;

workCost: number = 15;

moneyCost: number = 2;

plantValue: number;

Corn

grow(cell: any, growth: any): void

eval( ): number

getQuality( ): number

type: string;

workCost: number = 10;

moneyCost: number = 1;

plantValue: number;

Tomato

grow(cell: any, growth: any): void

eval( ): number

getQuality( ): number

type: string;

workCost: number = 12;

moneyCost: number = 2;

plantValue: number;

Carrot

grow(cell: any, growth: any): void

eval( ): number

getQuality( ): number

type: string;

workCost: number = 15;

moneyCost: number = 2;

plantValue: number;

Mushroom

grow(cell: any, growth: any): void

eval( ): number

getQuality( ): number

type: string;

workCost: number = 10;

moneyCost: number = 1;

plantValue: number;

RandomSequence

nexti( ): number

next( ): number

z: number;

w: number;

Crop

serialize( ); void

deserialize(data: any): void

grow (cell: any, growth: any): void

getQuality( ): number

eval( ): void

type: string = “ “;

amount: number = 0;

quality: number = 1;

AD – Main:

**FarmGame**

\_xs: number, \_ys: number

**update**

\_deltaTime: number

var frameTime = 10;

this.time += deltaTime;

while

(frameTime <

this.time)

this.updateInternal( );

this.time -= frameTime;

this.xs = xs;

this.ys = ys;

cells = [ ];

workingPower = 100;

cash = startMoney;

weather = 0;

time = 0;

frameCount = 0;



**RandomSequence**

\_seed: any

this.z = Math.random();

this.w = Math.random();

**updateInternal**

humidityGrowth

sunlightGrowth

getGrowth



**harvest**

\_cell: any

var workCost = 15;

**getGrowth**

\_timeslice: number

add velocity & \_timeslice to position

ret +=

getter(this.cells[x - 1][y]);

if (x + 1

< this.xs)

ret += getter(this.cells

[x + 1][y]);

if 0 <= y - 1)

ret += getter(this.cells[x][y - 1]);

if (y + 1

< this.xs)

return ret \* humidityGrowth(cell) \* sunlightGrowth(this,cell);

if this.workingpower < workCost



if 0 <= x - 1)



return false;

if

(cell.harvest())



this.workingPower -= workCost;

this.cash += cell.crop.eval();

cell.crop = null;

cell.pesticide = 0;



return true;



**water**

\_cell: any

var workCost = 5;

if this.workingpower < workCost

return false;

if

(cell.water())

this.workingPower -= workCost; return true;



**fertilize**

\_cell: any

var workCost = 5;

var moneyCost = 3;

if this.workingPower <

workCost || this.cash < moneyCost

return false;

if

(cell. fertilize ())

this.workingPower -= workCost; this.cash -= moneyCost;

return true;

**pesticideing**

\_cell: any

var workCost = 10;

var moneyCost = 10;

if this.workingPower <

workCost || this.cash < moneyCost

return false;

if

(cell. pesticideing ())

this.workingPower -= workCost; this.cash -= moneyCost;

return true;



**weeding**

\_cell: any

var workCost = 15;

if this.workingpower < workCost

return false;

if

(cell.weeding())

this.workingPower -= workCost; return true;

**plow**

\_cell: any

var workCost = 20;

if this.workingpower < workCost

return false;

if

(cell.plow())

this.workingPower -= workCost; return true;



**humidityGrowth**

\_cell: any

var h = (0.5 - cell.humidity) / 0.5;

return ((1. - h \* h) + 0.25) / 1.25;

for (var j = 0; j <= 1; j++){

var rng = new RandomSequence (Math.floor(i / period) + j);

var value = rng.next(rng);

sum += value \* value \* (j ? i % period : period - i % period) / period;}

**Smoothnoise**

\_i: number

var seed = 123;

var period = 600;

var sum = 0.;

**sunlightGrowth**

\_game: any, \_cell: any

return (1. - game.weather + 0.25) / 1.25;

AD – Sim:



handleLoad

load

**onload**

width = 640;

height = 640;

renderer =

graf.autoDetectRenderer(width, height);

document.getElementById("stage").appendChild(renderer.view);height);

stage = new graf.Stage;

init( );

