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)

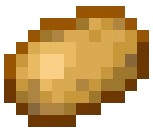
Zusatzfunktionen:

- ✓ 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.

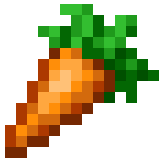
Texturen - Pflanzen und Aktionen:

Corn

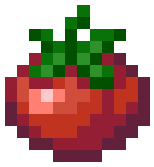
Potato



Carrot

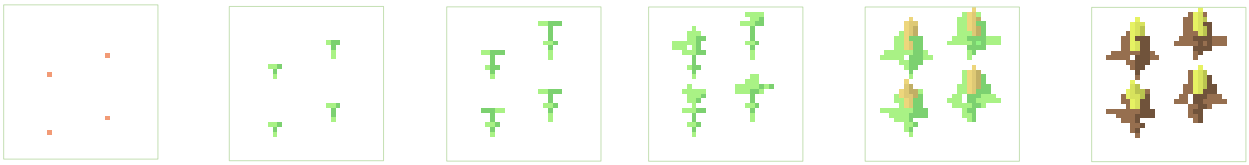


Tomato

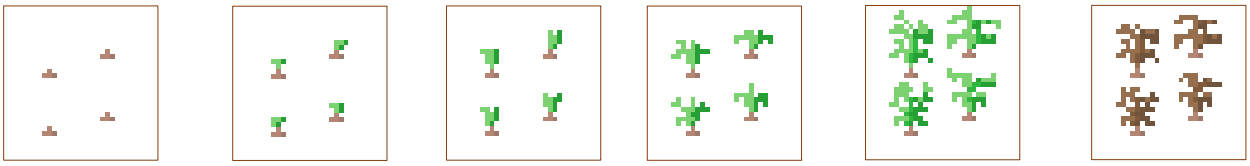


Mushroom

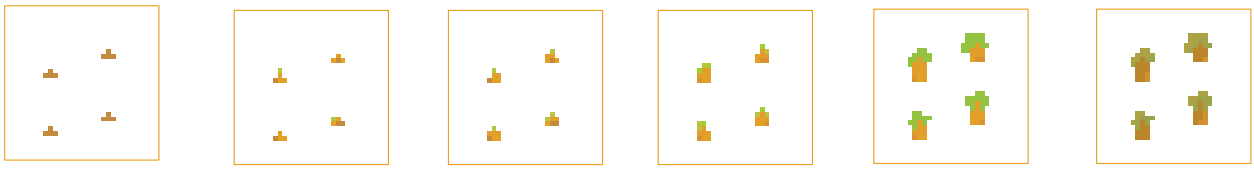
Pflanzenzustände Corn:



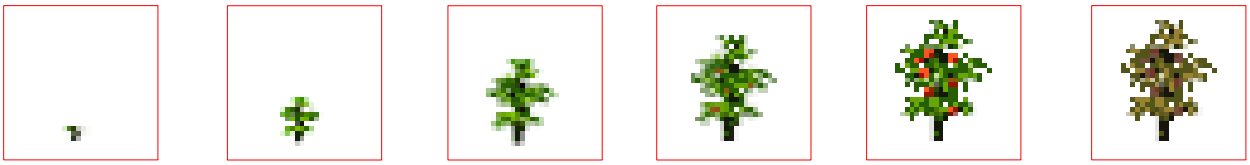
Pflanzenzustände Potato:



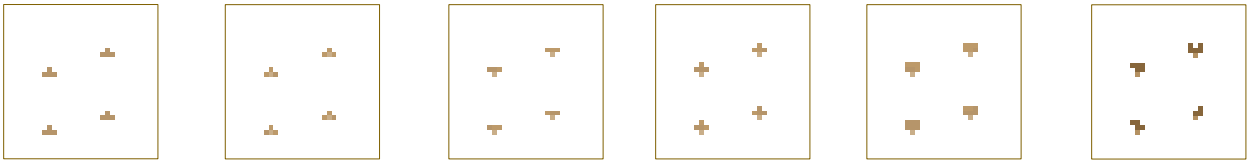
Pflanzenzustände Carrot:



Pflanzenzustände Tomato:



Pflanzenzustände Mushroom:

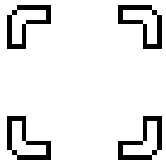


Aktionen:

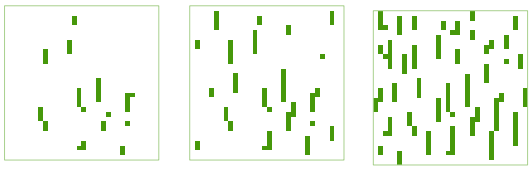
Plow Water Weed Harvest Fertilize Pesticide

Wetter:

Cursor:



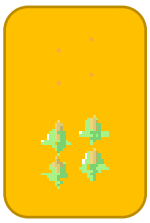
Weeds:



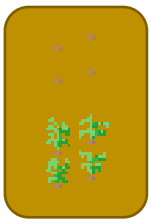
Ground:

Pflanzen und Aktionen - Info:

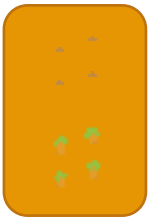
- Kaufpreis: 1
- Energieverbrauch: 10 (Niedrig)
- Verkaufspreis: 0-6



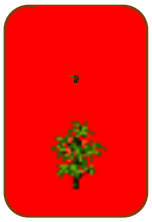
- Kaufpreis: 2
- Energieverbrauch: 15 (Mittel)
- Verkaufspreis: 0-9



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



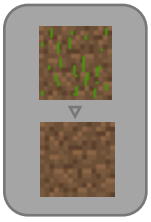
- Kaufpreis: 2
- Energieverbrauch: 12 (Mittel)
- Verkaufspreis: 0-7



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

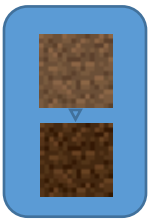
Plowing
- Energieverbrauch: 20 (Hoch)
- Funktion: Plow and make ridges

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



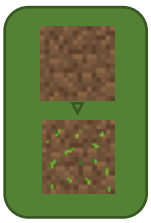
Harvesting
- Energieverbrauch: 15 (Mittel)
- Funktion: Harvest and sell crops

Watering
- Energieverbrauch: 5 (Niedrig)
- Funktion: Water soil

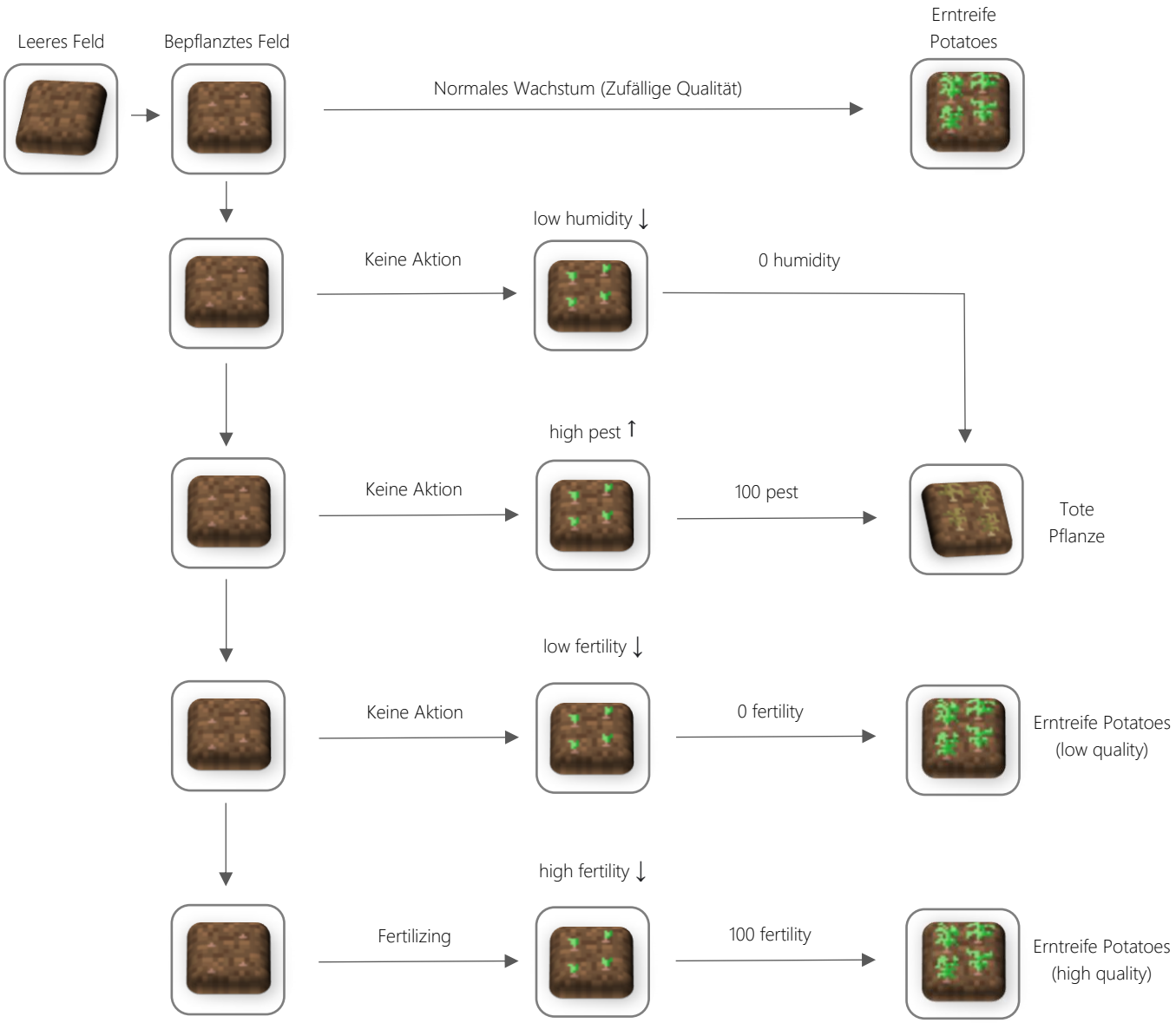


Fertilizing
- **Energieverbrauch: 5 (Niedrig)**
- **Funktion: Helps crops grow**
- **Kaufpreis: 3**

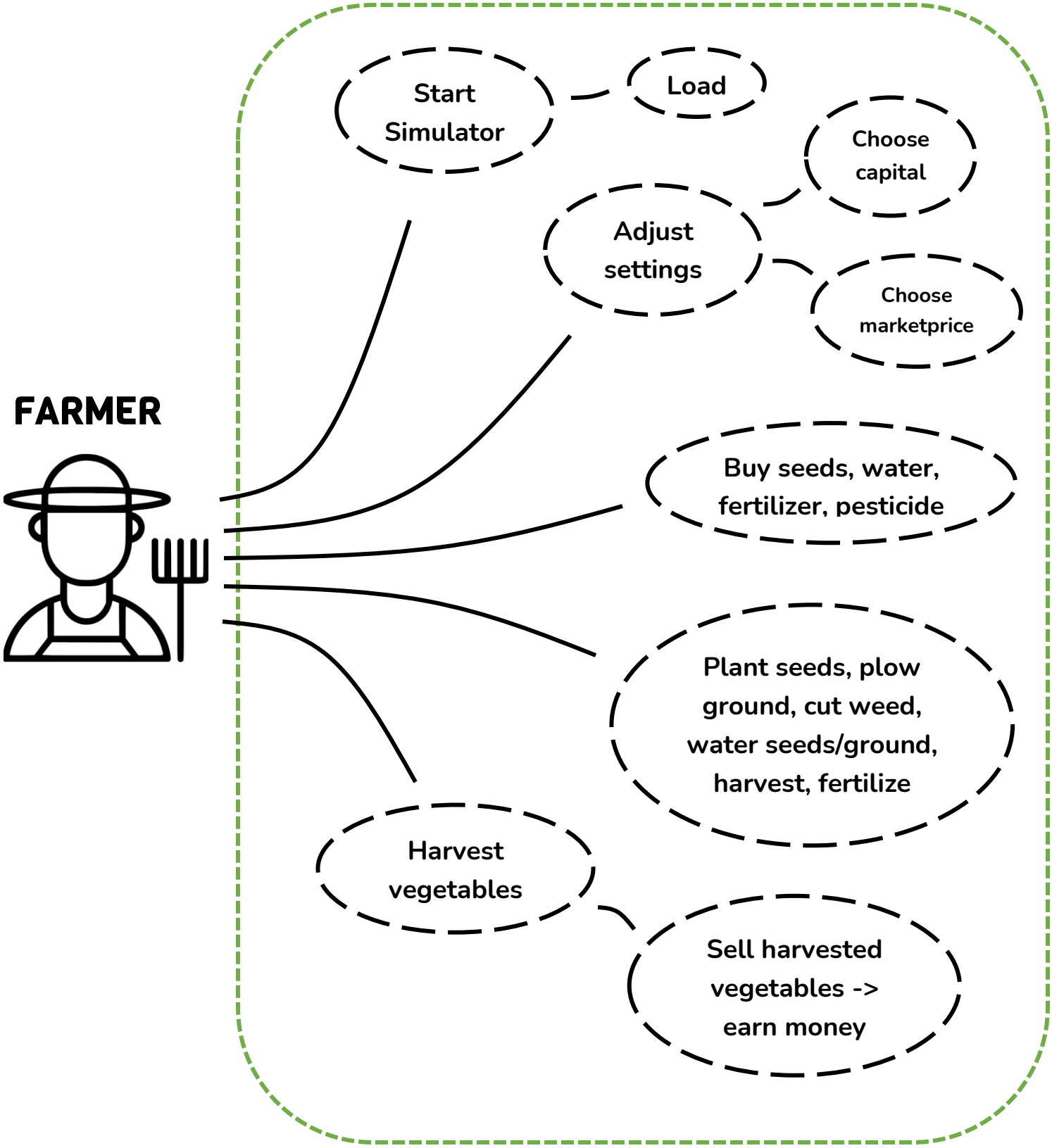
Pesticiding
- Energieverbrauch: 10 (Niedrig)
- Funktion: Keeps harvest safe from parasites
- Kaufpreis: 10



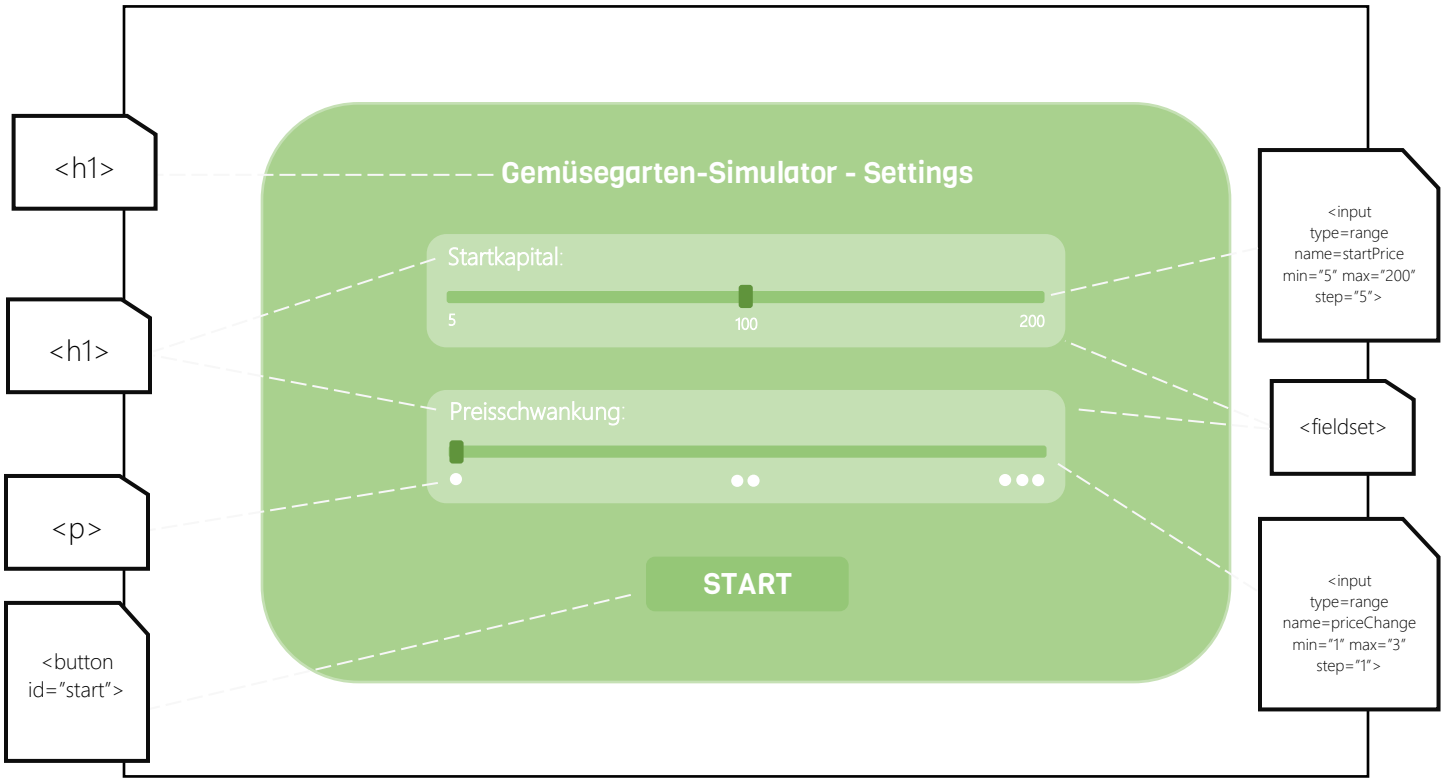
Simulator Ablauf - Wachstum:

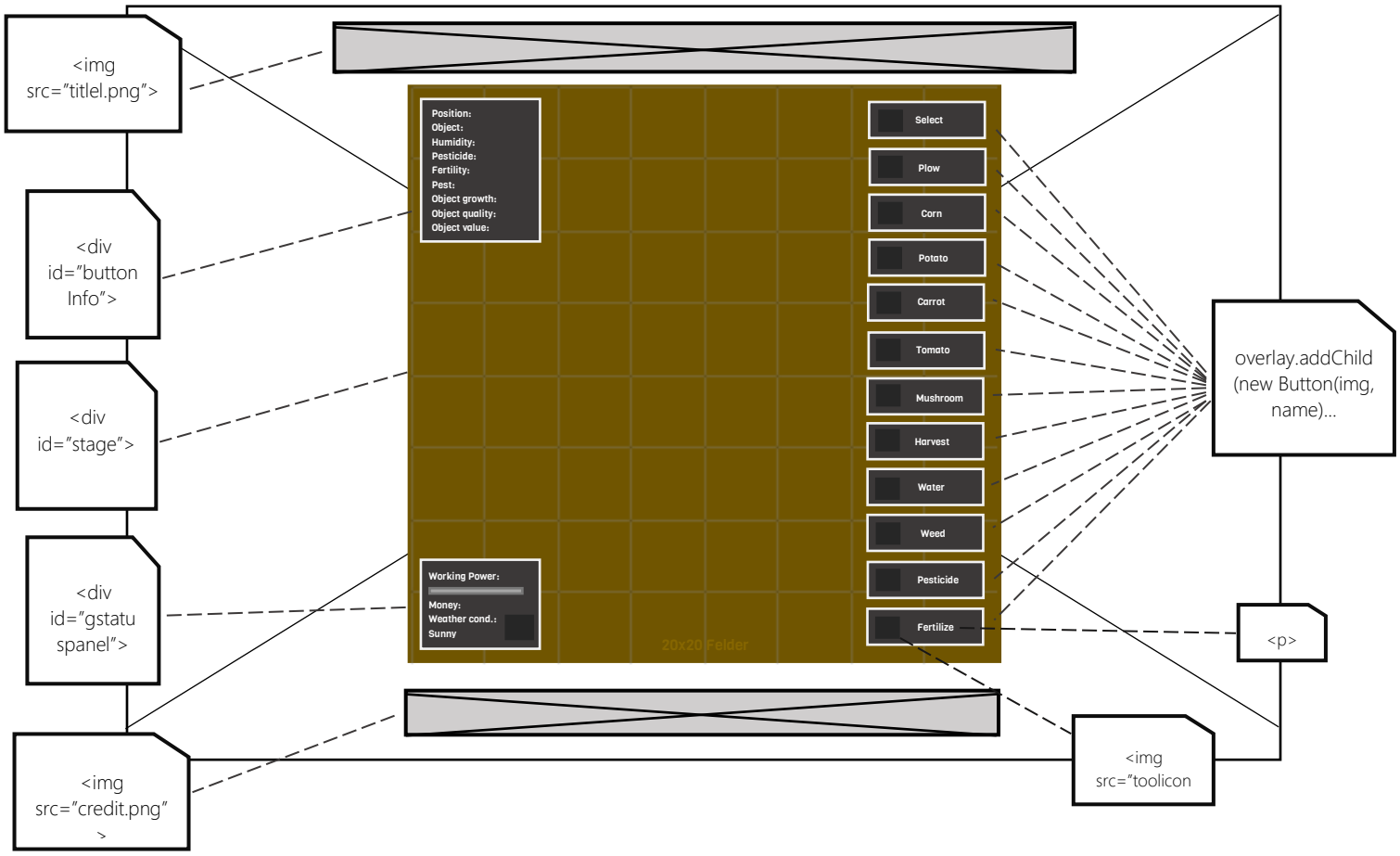


Use-Case:

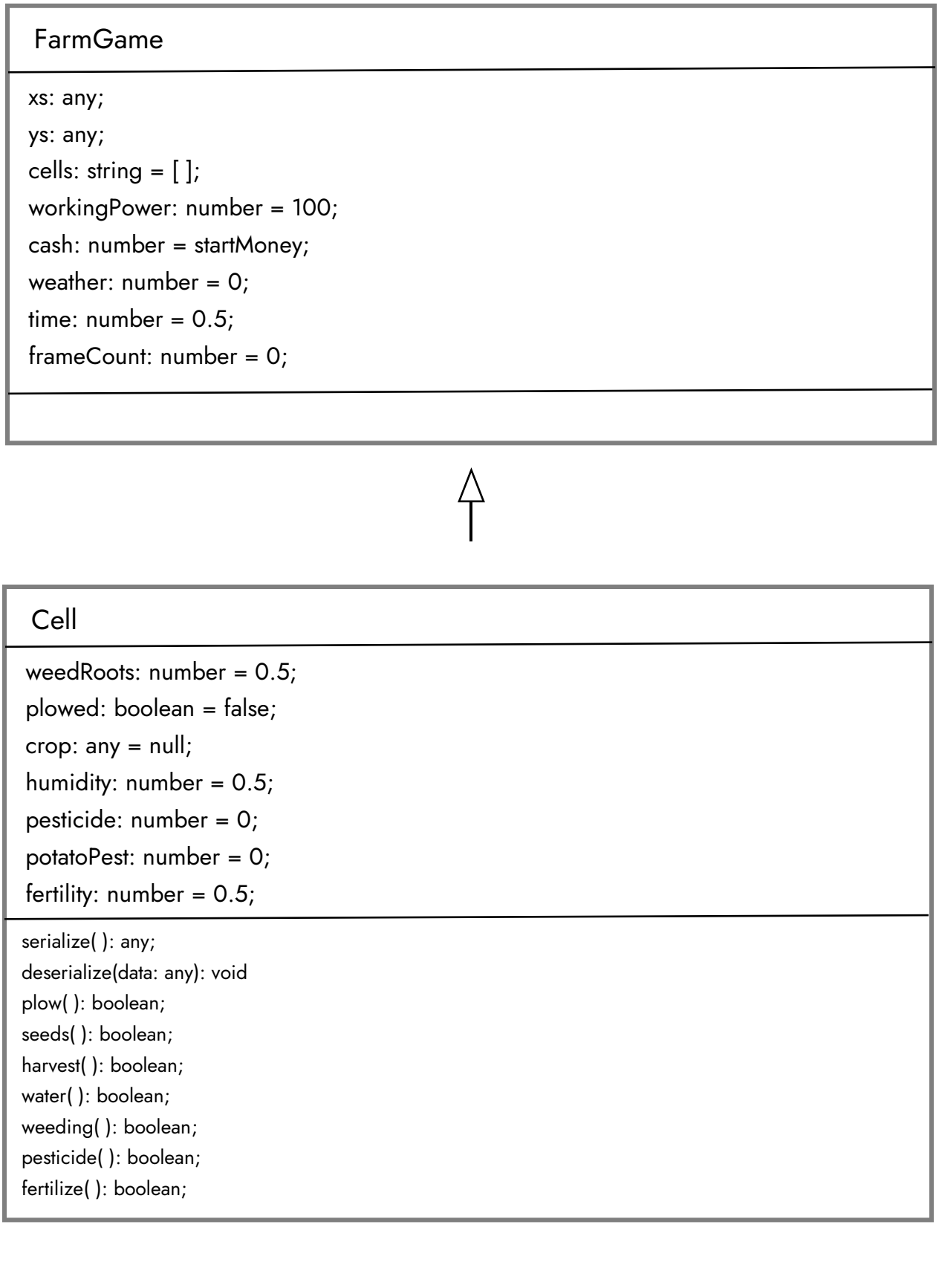

Start Simulator
Load
Choose capital
Adjust settings
Choose marketprice
FARMER
Buy seeds, water, fertilizer, pesticide
Plant seeds, plow ground, cut weed, water seeds/ground, harvest, fertilize
Harvest vegetables
Sell harvested vegetables -> earn money

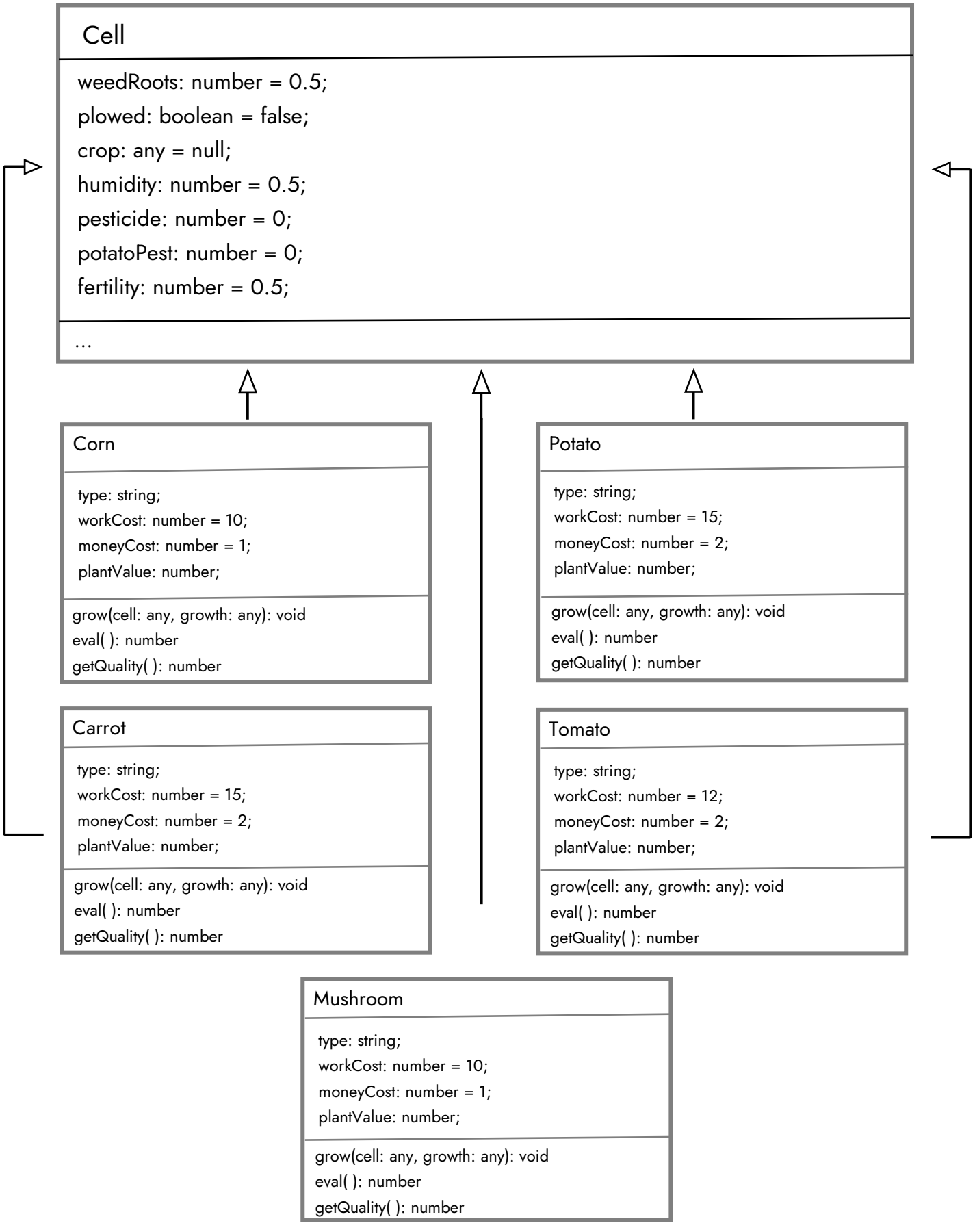
UI-Scribble - Settings:

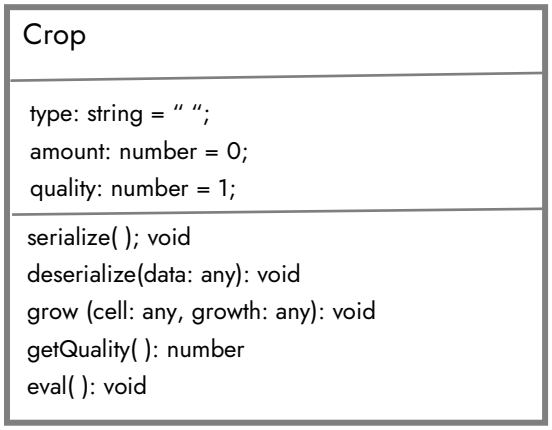


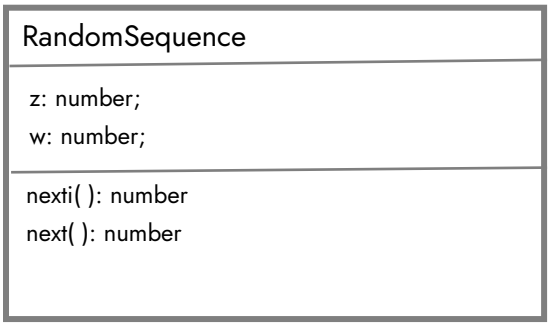


ClassDiagramm:

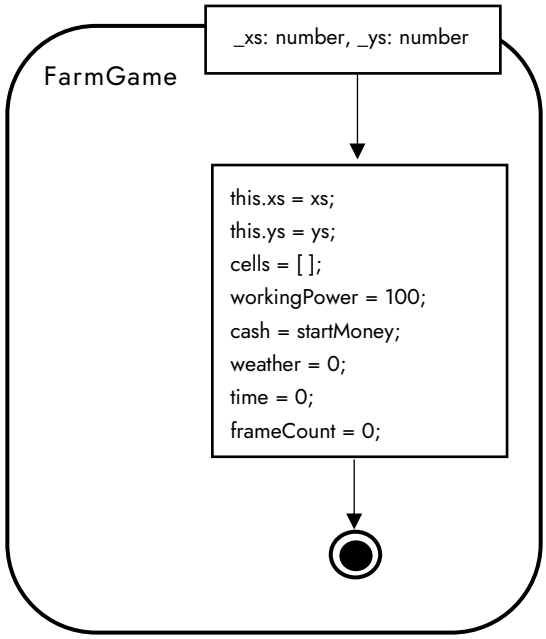


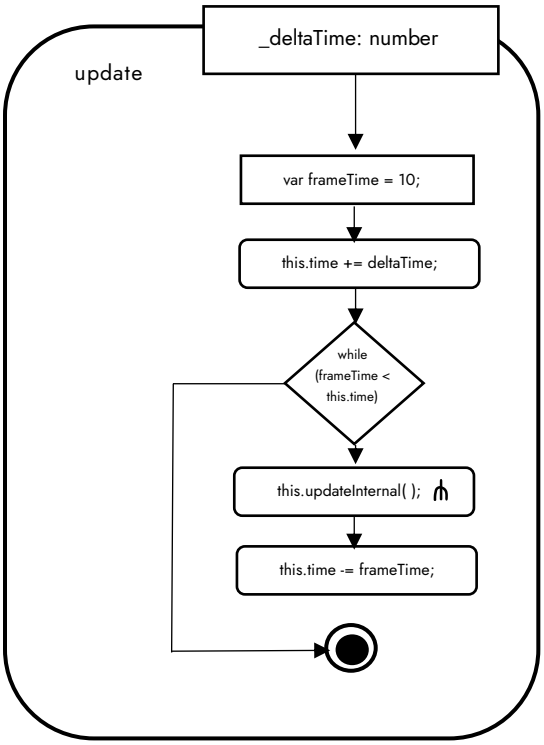

Cell
weedRoots: number = 0.5;
plowed: boolean = false;
crop: any = null;
humidity: number = 0.5;
pesticide: number = 0;
potatoPest: number = 0;
fertility: number = 0.5;
...
Corn
type: string;
workCost: number = 10;
moneyCost: number = 1;
plantValue: number;
grow(cell: any, growth: any): void
eval( ): number
getQuality( ): number
Potato
type: string;
workCost: number = 15;
moneyCost: number = 2;
plantValue: number;
grow(cell: any, growth: any): void
eval( ): number
getQuality( ): number
Carrot
type: string;
workCost: number = 15;
moneyCost: number = 2;
plantValue: number;
grow(cell: any, growth: any): void
eval( ): number
getQuality( ): number
Tomato
type: string;
workCost: number = 12;
moneyCost: number = 2;
plantValue: number;
grow(cell: any, growth: any): void
eval( ): number
getQuality( ): number
Mushroom
type: string;
workCost: number = 10;
moneyCost: number = 1;
plantValue: number;
grow(cell: any, growth: any): void
eval( ): number
getQuality( ): number

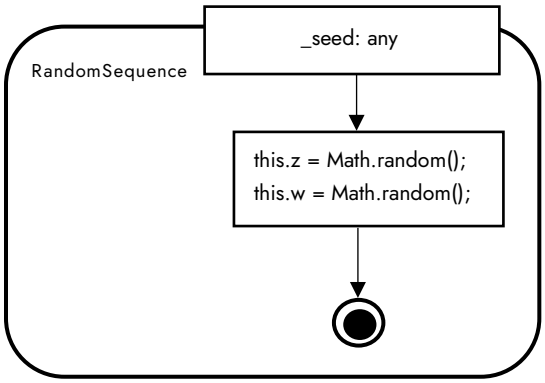

Crop
type: string = " ";
amount: number = 0;
quality: number = 1;
serialize( ); void
deserialize(data: any): void
grow (cell: any, growth: any): void
getQuality( ): number
eval( ): void

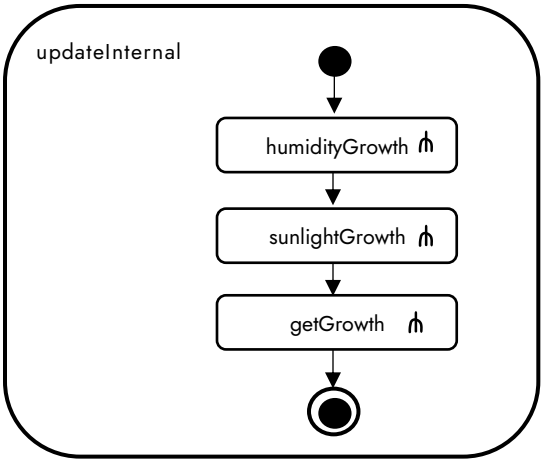

RandomSequence
z: number;
w: number;
nexti( ): number
next( ): number

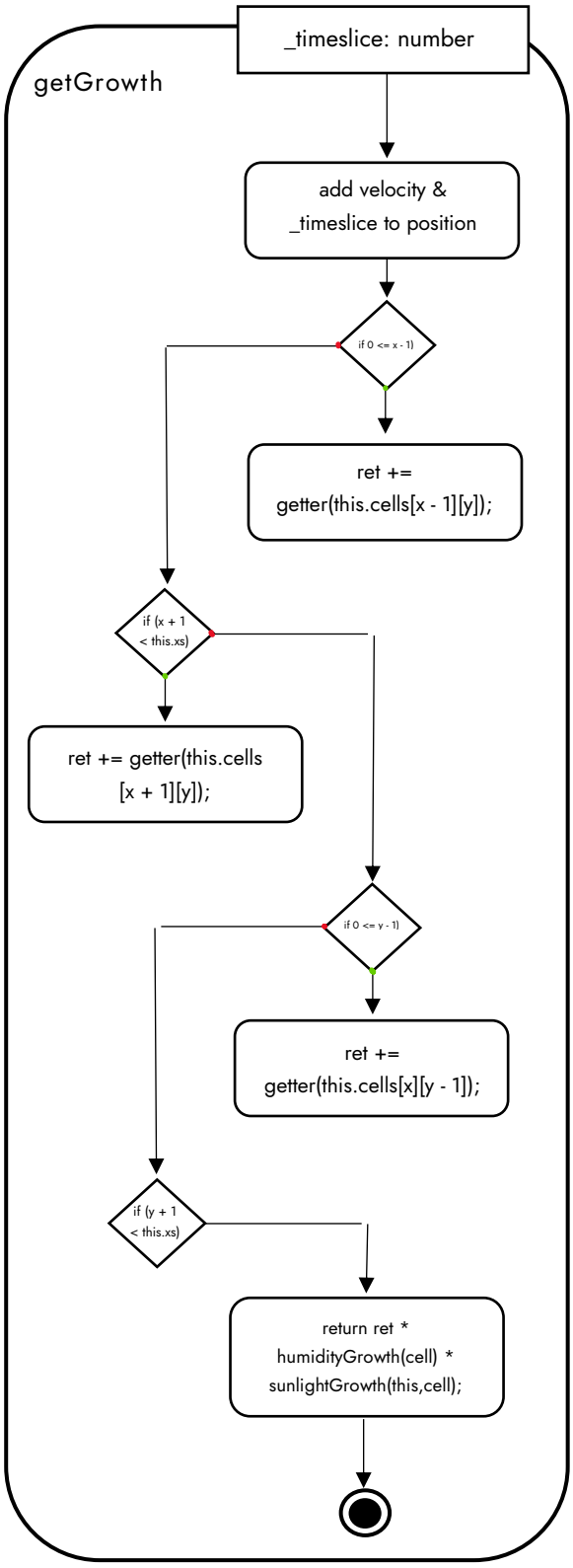
AD – Main:

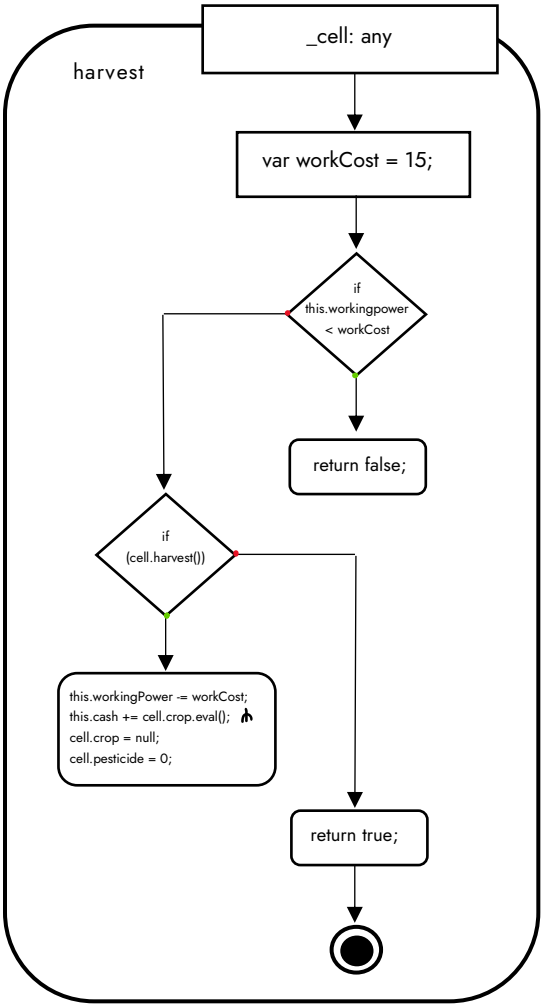

FarmGame
_xs: number, _ys: number
this.xs = xs;
this.ys = ys;
cells = [ ];
workingPower = 100;
cash = startMoney;
weather = 0;
time = 0;
frameCount = 0;

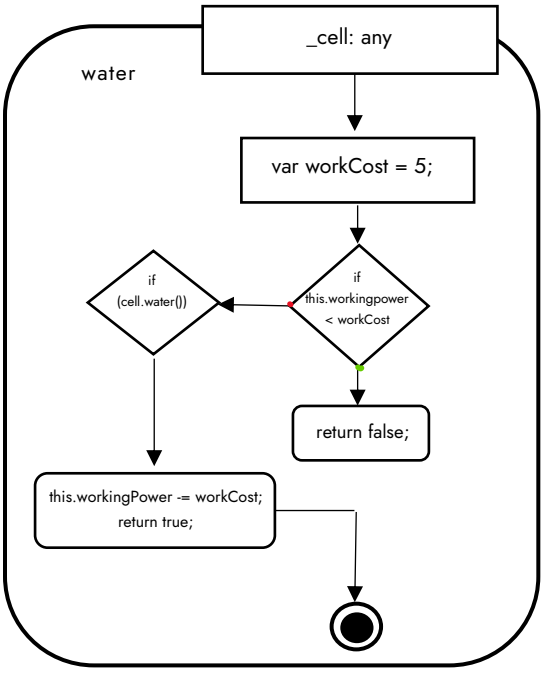

update
_deltaTime: number
var frameTime = 10;
this.time += deltaTime;
while (frameTime < this.time)
this.updateInternal( );
this.time -= frameTime;

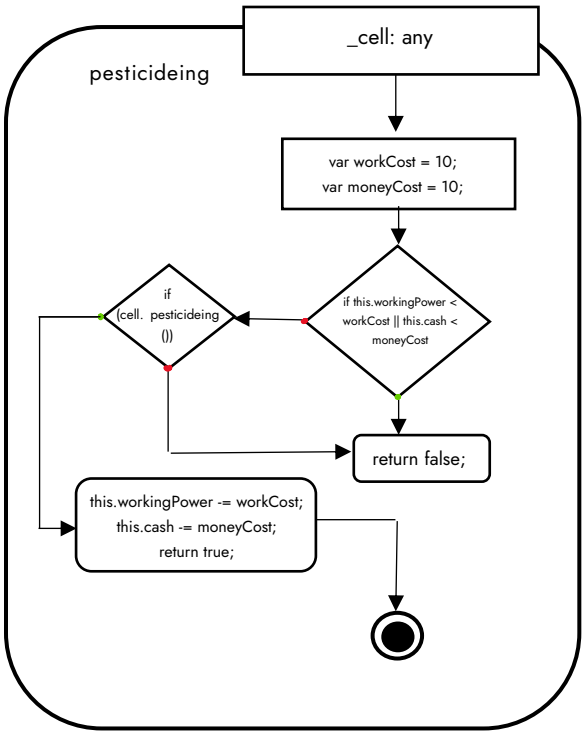

RandomSequence
_seed: any
this.z = Math.random();
this.w = Math.random();

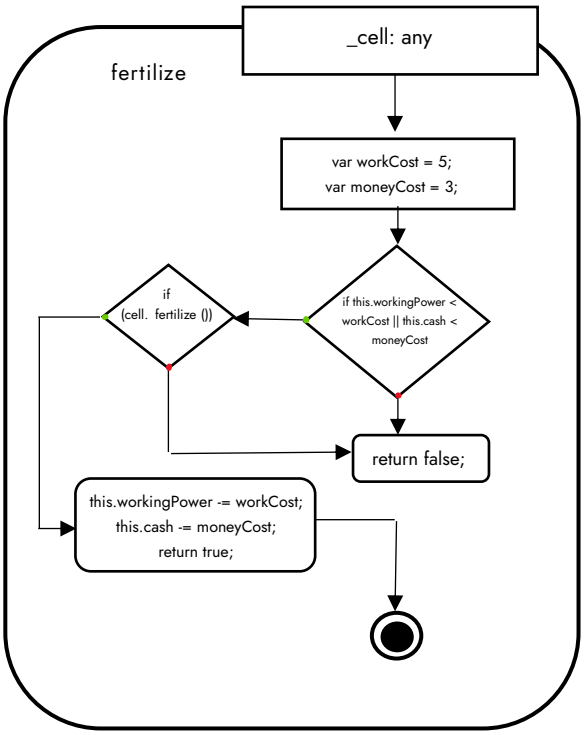

updateInternal
humidityGrowth
sunlightGrowth
getGrowth

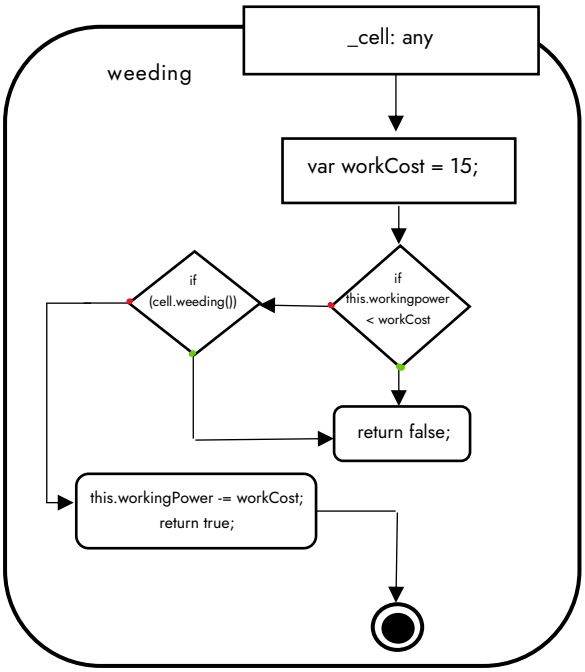

getGrowth
_timeslice: number
add velocity &
_timeslice to position
if 0 <= x - 1)
ret +=
getter(this.cells[x - 1][y]);
if (x + 1
< this.xs)
ret += getter(this.cells
[x + 1][y]);
if 0 <= y - 1)
ret +=
getter(this.cells[x][y - 1]);
if (y + 1
< this.xs)
return ret *
humidityGrowth(cell) *
sunlightGrowth(this,cell);

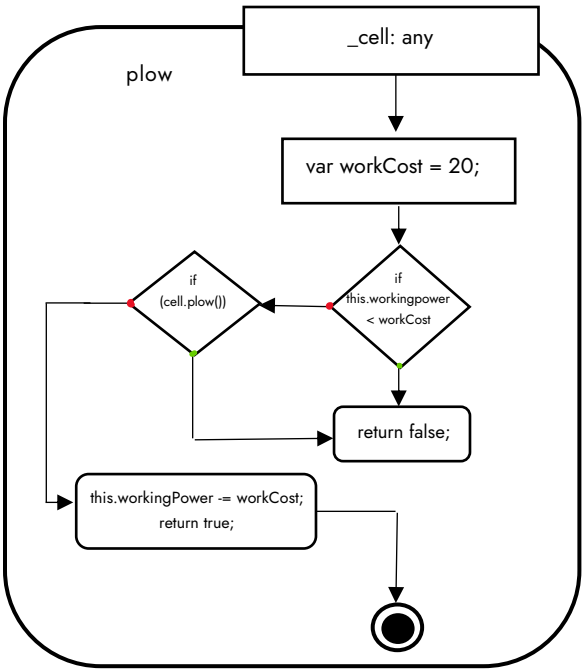

harvest
_cell: any
var workCost = 15;
if
this.workingpower
< workCost
return false;
if
(cell.harvest())
this.workingPower -= workCost;
this.cash += cell.crop.eval();
cell.crop = null;
cell.pesticide = 0;
return true;

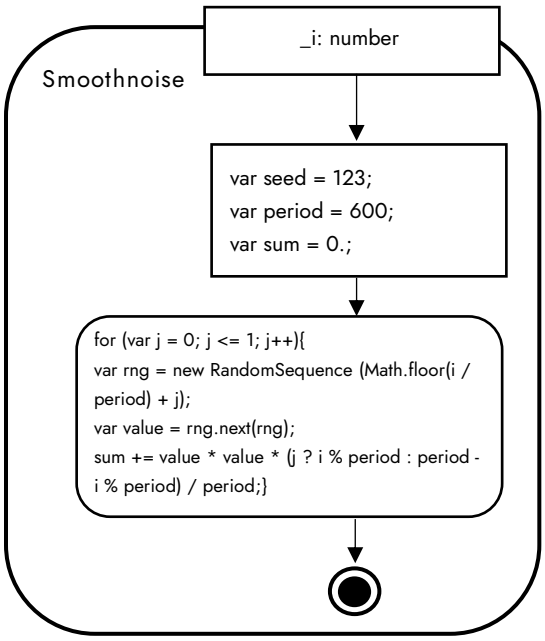

water
_cell: any
var workCost = 5;
if
(cell.water())
if
this.workingpower
< workCost
return false;
this.workingPower -= workCost;
return true;

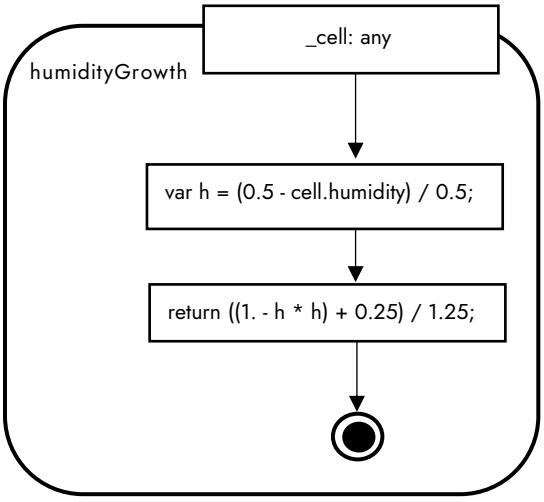

pesticideing
_cell: any
var workCost = 10;
var moneyCost = 10;
if this.workingPower < workCost || this.cash < moneyCost
if (cell. pesticideing ())
return false;
this.workingPower -= workCost;
this.cash -= moneyCost;
return true;

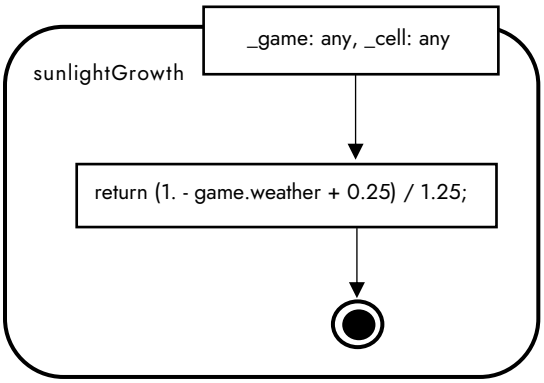

fertilize
_cell: any
var workCost = 5;
var moneyCost = 3;
if this.workingPower < workCost || this.cash < moneyCost
if (cell. fertilize ())
return false;
this.workingPower -= workCost;
this.cash -= moneyCost;
return true;


weeding
_cell: any
var workCost = 15;
if this.workingpower < workCost
if (cell.weeding())
return false;
this.workingPower -= workCost;
return true;

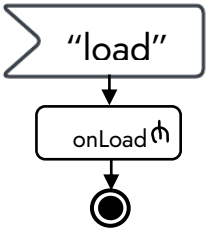

plow
_cell: any
var workCost = 20;
if this.workingpower < workCost
if (cell.plow())
return false;
this.workingPower -= workCost;
return true;

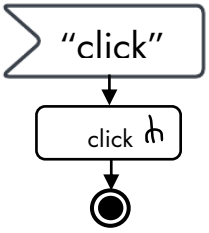

Smoothnoise
_i: number
var seed = 123;
var period = 600;
var sum = 0.;
for (var j = 0; j <= 1; j++){
var rng = new RandomSequence (Math.floor(i /
period) + j);
var value = rng.next(rng);
sum += value * value * (j ? i % period : period -
i % period) / period;}

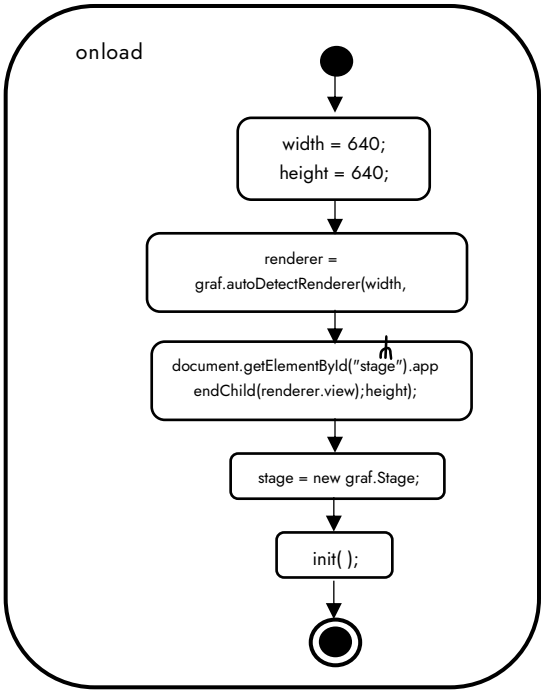

humidityGrowth
_cell: any
var h = (0.5 - cell.humidity) / 0.5;
return ((1. - h * h) + 0.25) / 1.25;

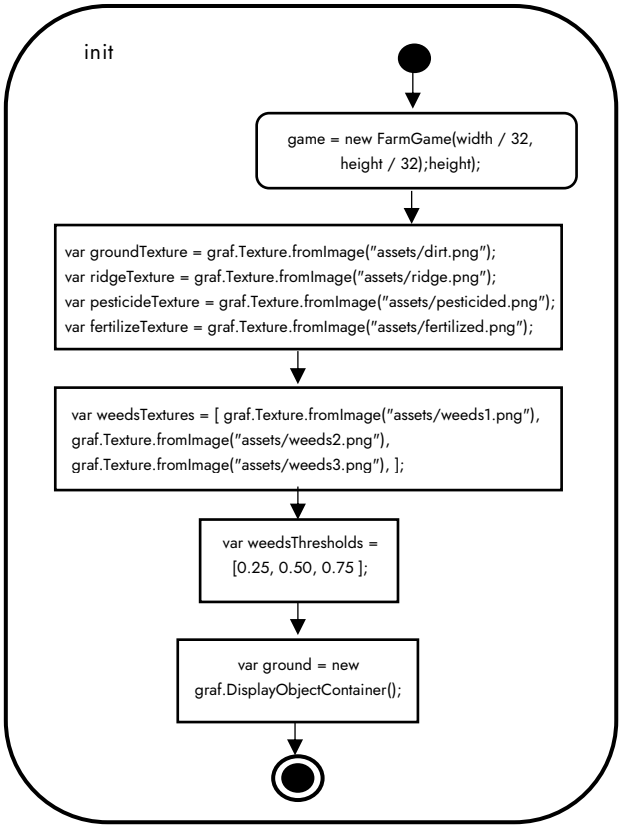

sunlightGrowth
_game: any, _cell: any
return (1. - game.weather + 0.25) / 1.25;

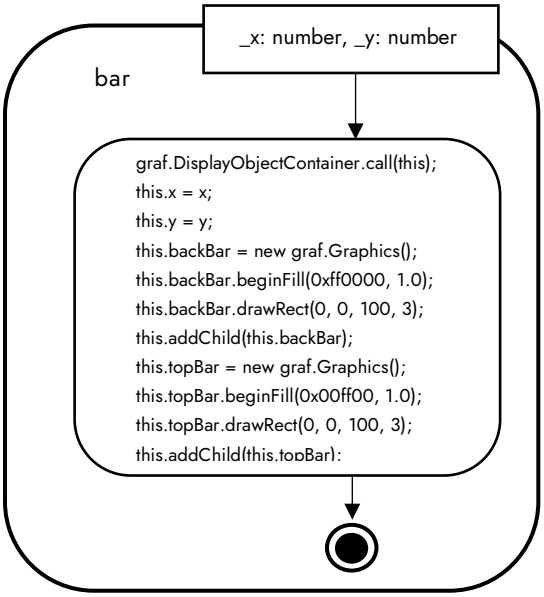
AD – Sim:

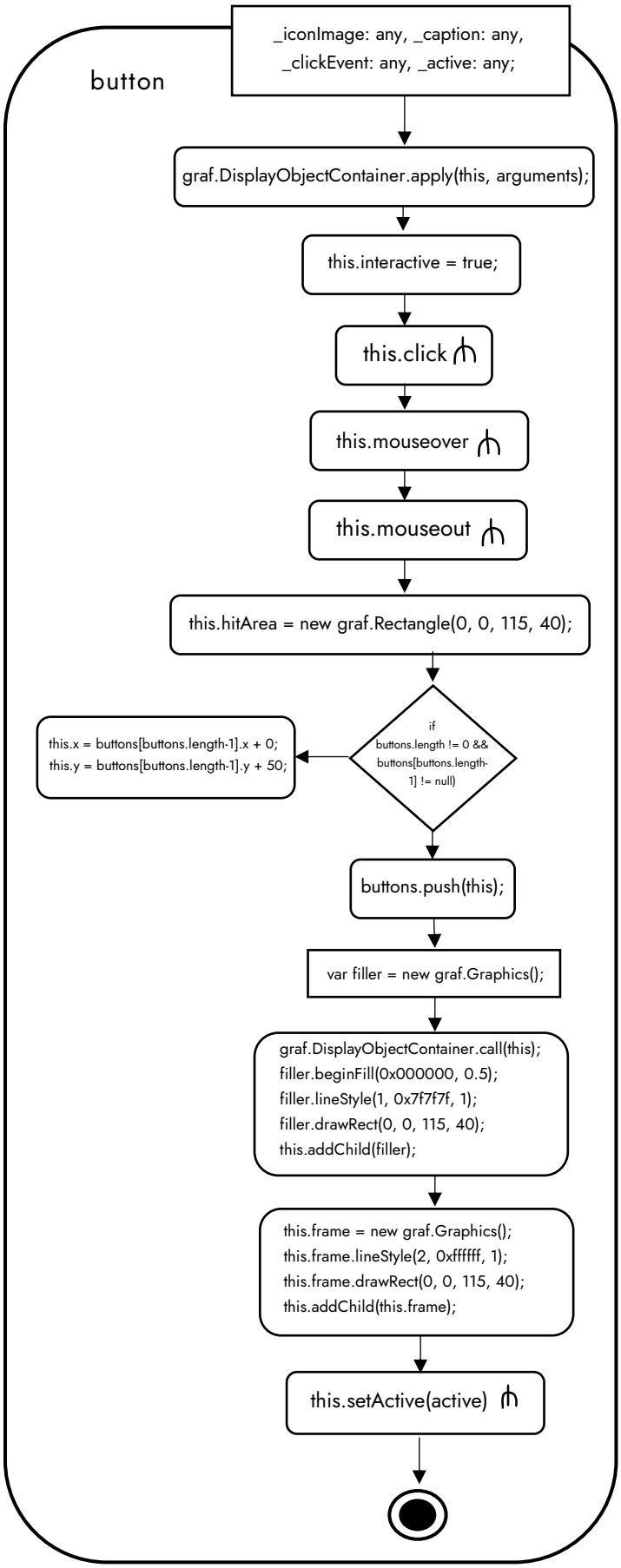

"load"
onLoad

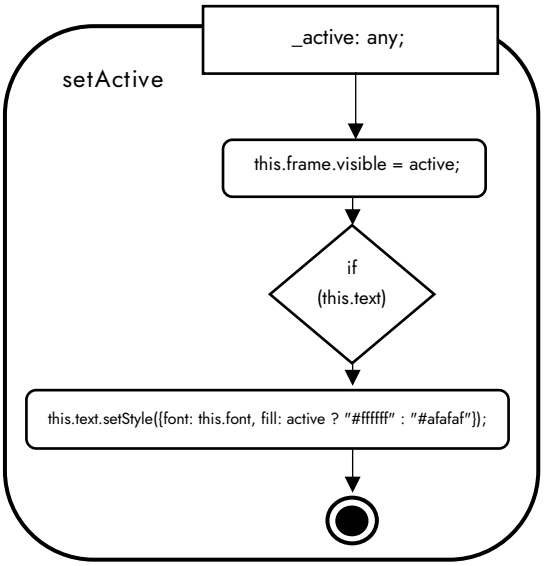

"click"
click

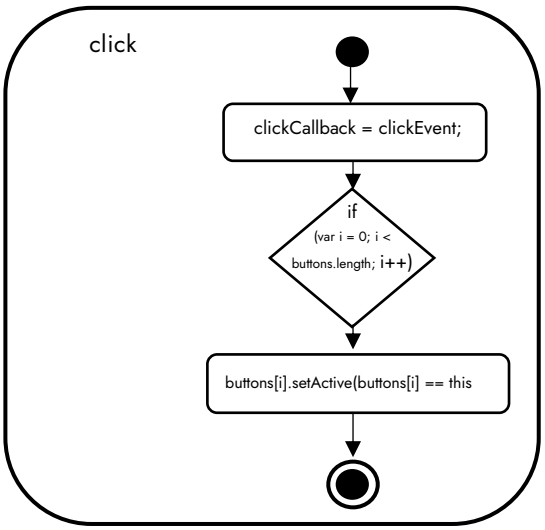

onload
width = 640;
height = 640;
renderer =
graf.autoDetectRenderer(width,
document.getElementById("stage").app
endChild(renderer.view);height);
stage = new graf.Stage;
init( );

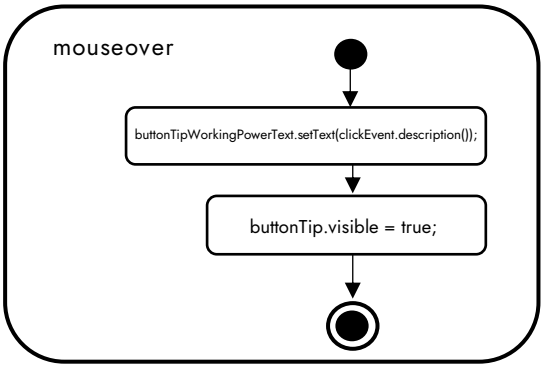

init
game = new FarmGame(width / 32,
height / 32);height);
var groundTexture = graf.Texture.fromImage("assets/dirt.png");
var ridgeTexture = graf.Texture.fromImage("assets/ridge.png");
var pesticideTexture = graf.Texture.fromImage("assets/pesticided.png");
var fertilizeTexture = graf.Texture.fromImage("assets/fertilized.png");
var weedsTextures = [graf.Texture.fromImage("assets/weeds1.png"),
graf.Texture.fromImage("assets/weeds2.png"),
graf.Texture.fromImage("assets/weeds3.png"),];
var weedsThresholds =
[0.25, 0.50, 0.75];
var ground = new
graf.DisplayObjectContainer();

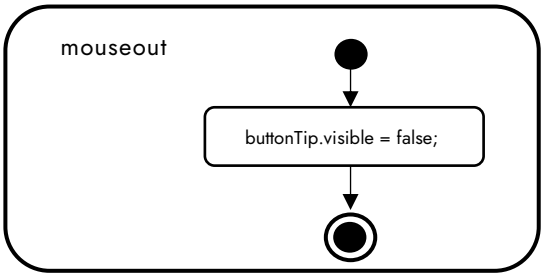

_x: number, _y: number
bar
graf.DisplayObjectContainer.call(this);
this.x = x;
this.y = y;
this.backBar = new graf.Graphics();
this.backBar.beginFill(0xff0000, 1.0);
this.backBar.drawRect(0, 0, 100, 3);
this.addChild(this.backBar);
this.topBar = new graf.Graphics();
this.topBar.beginFill(0x00ff00, 1.0);
this.topBar.drawRect(0, 0, 100, 3);
this.addChild(this.topBar);

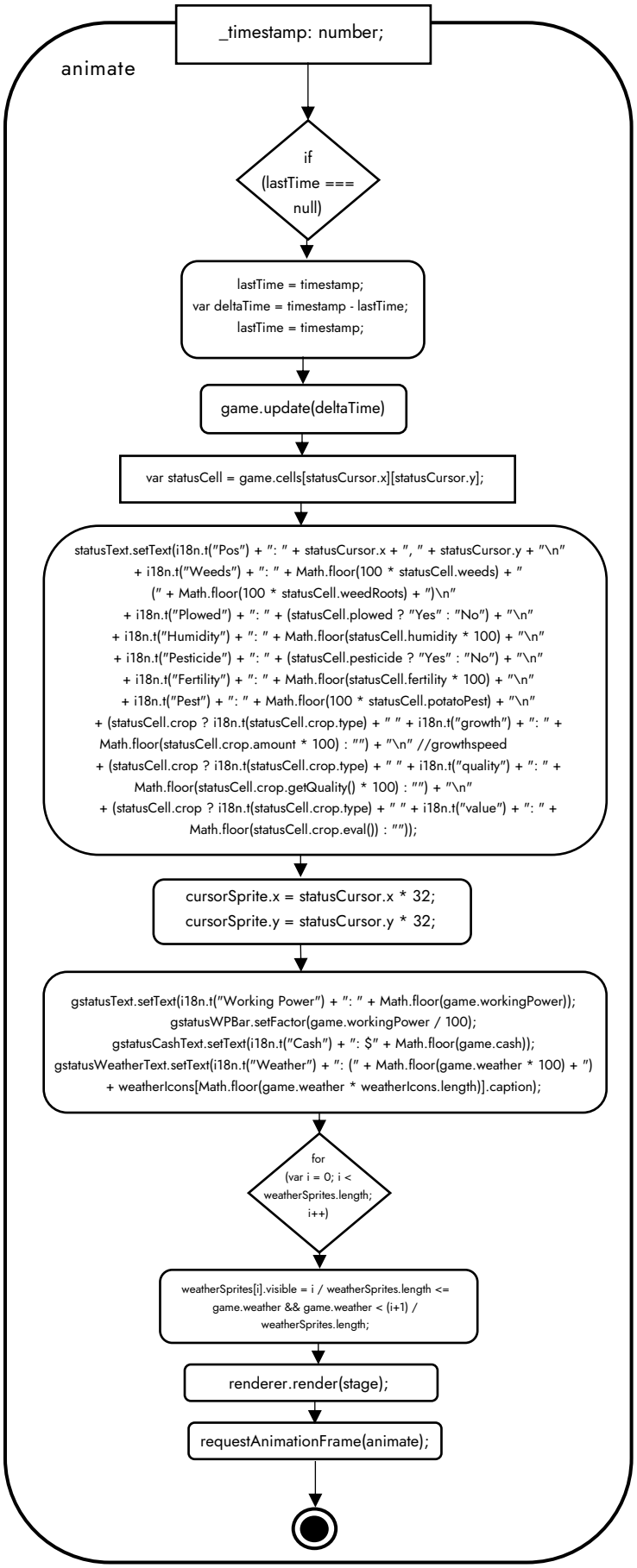

button
_iconImage: any, _caption: any,
_clickEvent: any, _active: any;
graf.DisplayObjectContainer.apply(this, arguments);
this.interactive = true;
this.click
this.mouseover
this.mouseout
this.hitArea = new graf.Rectangle(0, 0, 115, 40);
if
buttons.length != 0 &&
buttons[buttons.length-
1] != null)
this.x = buttons[buttons.length-1].x + 0;
this.y = buttons[buttons.length-1].y + 50;
buttons.push(this);
var filler = new graf.Graphics();
graf.DisplayObjectContainer.call(this);
filler.beginFill(0x000000, 0.5);
filler.lineStyle(1, 0x7f7f7f, 1);
filler.drawRect(0, 0, 115, 40);
this.addChild(filler);
this.frame = new graf.Graphics();
this.frame.lineStyle(2, 0xffffff, 1);
this.frame.drawRect(0, 0, 115, 40);
this.addChild(this.frame);
this.setActive(active)


setActive
_active: any;
this.frame.visible = active;
if
(this.text)
this.text.setStyle({font: this.font, fill: active ? "#ffffff" : "#afafaf"});


click
clickCallback = clickEvent;
if
(var i = 0; i <
buttons.length; i++)
buttons[i].setActive(buttons[i] == this


mouseover
buttonTipWorkingPowerText.setText(clickEvent.description());
buttonTip.visible = true;


mouseout
buttonTip.visible = false;


_timestamp: number;
animate
if
(lastTime ===
null)
lastTime = timestamp;
var deltaTime = timestamp - lastTime;
lastTime = timestamp;
game.update(deltaTime)
var statusCell = game.cells[statusCursor.x][statusCursor.y];
statusText.setText(i18n.t("Pos") + ": " + statusCursor.x + ", " + statusCursor.y + "\n"
+ i18n.t("Weeds") + ": " + Math.floor(100 * statusCell.weeds) + "
(" + Math.floor(100 * statusCell.weedRoots) + ")\n"
+ i18n.t("Plowed") + ": " + (statusCell.plowed ? "Yes" : "No") + "\n"
+ i18n.t("Humidity") + ": " + Math.floor(statusCell.humidity * 100) + "\n"
+ i18n.t("Pesticide") + ": " + (statusCell.pesticide ? "Yes" : "No") + "\n"
+ i18n.t("Fertility") + ": " + Math.floor(statusCell.fertility * 100) + "\n"
+ i18n.t("Pest") + ": " + Math.floor(100 * statusCell.potatoPest) + "\n"
+ (statusCell.crop ? i18n.t(statusCell.crop.type) + " " + i18n.t("growth") + ": " +
Math.floor(statusCell.crop.amount * 100) : "") + "\n" //growthspeed
+ (statusCell.crop ? i18n.t(statusCell.crop.type) + " " + i18n.t("quality") + ": " +
Math.floor(statusCell.crop.getQuality() * 100) : "") + "\n"
+ (statusCell.crop ? i18n.t(statusCell.crop.type) + " " + i18n.t("value") + ": " +
Math.floor(statusCell.crop.eval()) : ""));
cursorSprite.x = statusCursor.x * 32;
cursorSprite.y = statusCursor.y * 32;
gstatusText.setText(i18n.t("Working Power") + ": " + Math.floor(game.workingPower));
gstatusWPBar.setFactor(game.workingPower / 100);
gstatusCashText.setText(i18n.t("Cash") + ": $" + Math.floor(game.cash));
gstatusWeatherText.setText(i18n.t("Weather") + ": (" + Math.floor(game.weather * 100) + ")
+ weatherIcons[Math.floor(game.weather * weatherIcons.length)].caption);
for
(var i = 0; i <
weatherSprites.length;
i++)
weatherSprites[i].visible = i / weatherSprites.length <=
game.weather && game.weather < (i+1) /
weatherSprites.length;
renderer.render(stage);
requestAnimationFrame(animate);