

Game Development

Aufbau und Evolution von Spielsystemen

im Bereich Frontend Web Development

Ein Abschlussprojekt von Martin Meurer an der ReDi School



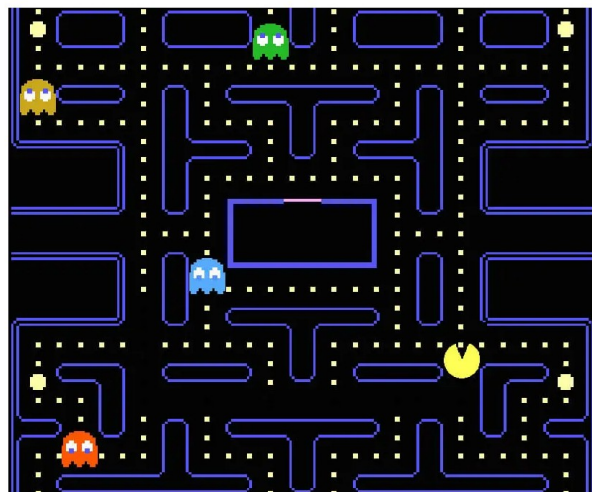
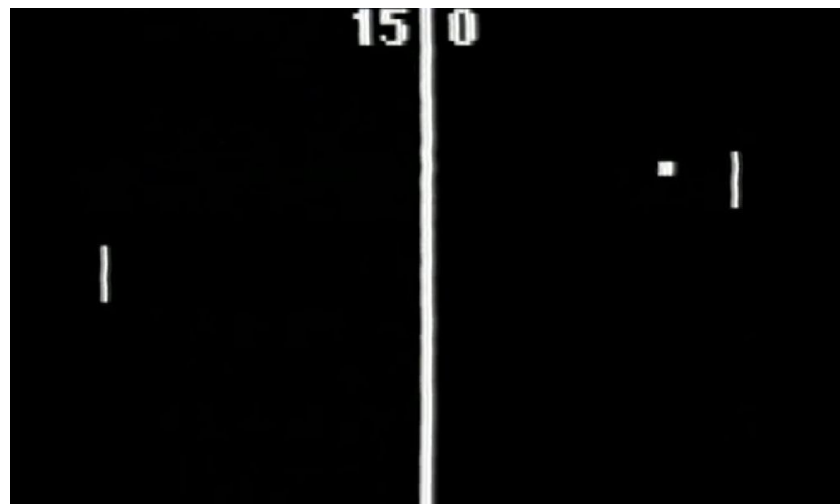
Game Engine

The Big Picture: Software Architecture

Die Verschmelzung aller Fachrichtungen und Diszipline

Managment komplexer Systeme aus der Interaktion einfacher Bestandteile

Testumgebung für vielerlei dynamische Prozesse

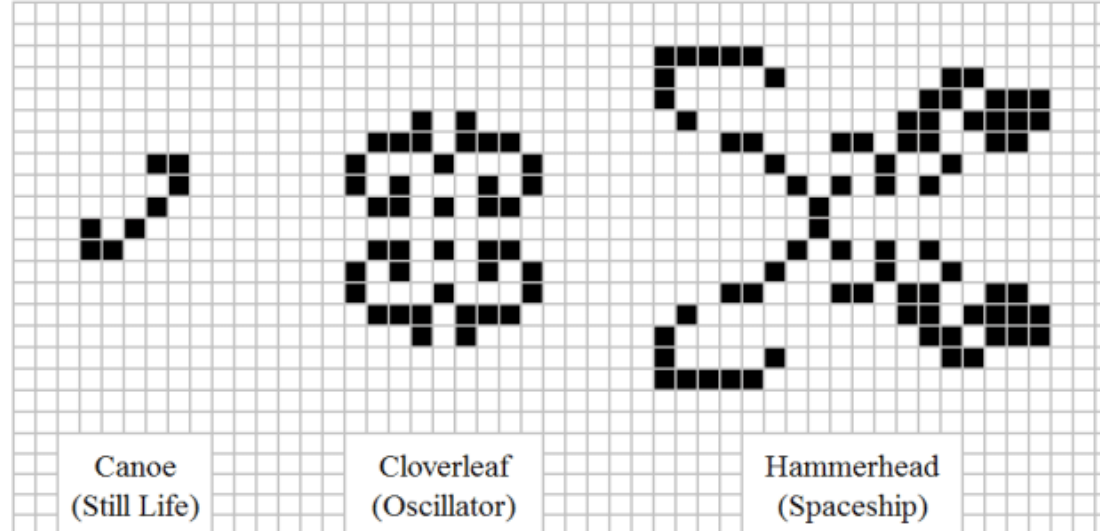


Game of Life

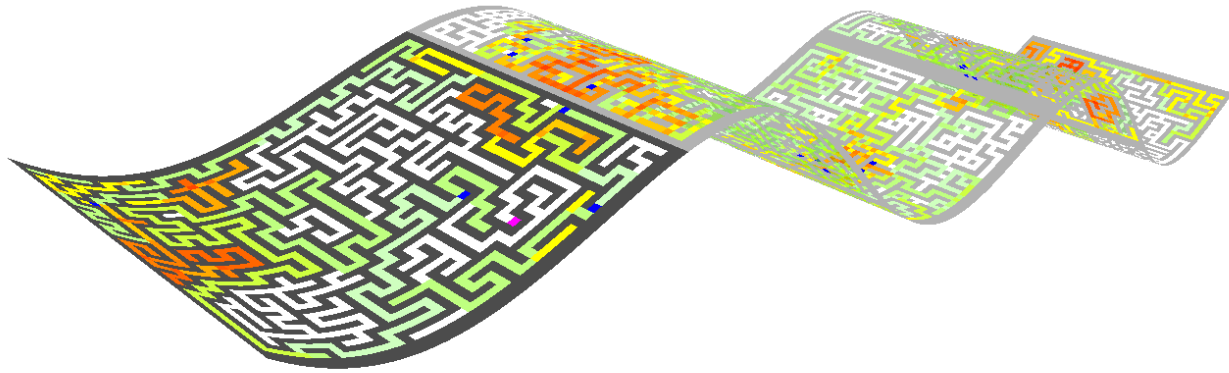
Produktion muss nicht immer aufwendig und kostspielig sein

Eine gute Idee, einfache Spielregeln, einfache Technik

Versteckte Tiefe, endlose Faszination



Realtime Demo



World Building


Das Klassische Labyrinth

Symbolische Representation der Daten

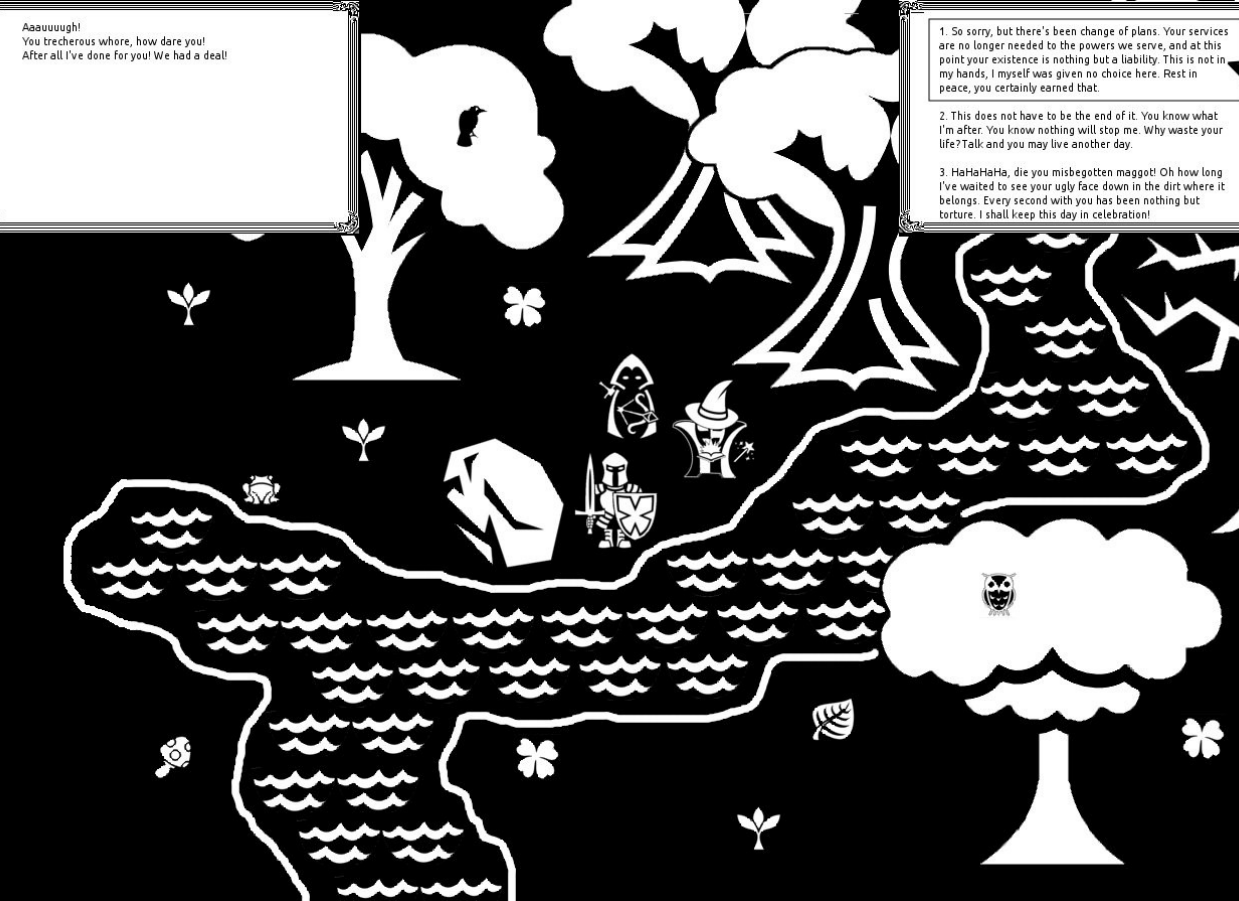
Datenfluss durch Übersetzung Schicht für Schicht


Prozess des Aufbaus großer Umgebungen Schritt für Schritt

First Game Project



Aaauuuugh!
You trecherous whore, how dare you!
After all I've done for you! We had a deal!











1. So sorry, but there's been change of plans. Your services are no longer needed to the powers we serve, and at this point your existence is nothing but a liability. This is not in my hands, I myself was given no choice here. Rest in peace, you certainly earned that.
2. This does not have to be the end of it. You know what I'm after. You know nothing will stop me. Why waste your life? Talk and you may live another day.
3. HaHaHaHa, die you misbegotten maggots! Oh how long I've waited to see your ugly face down in the dirt where it belongs. Every second with you has been nothing but torture. I shall keep this day in celebration!




Name: Malek
Race: Human
Class: Illusionist / Wizard
Strength: 7
Agility: 10
Intelligence: 19 (17+2)
Item: Book of Madness
Ability: Phantom Menace
Life: 29/60
Regeneration: +0.5/s
Mana: 150/150 (110+40)
Regeneration: +2.0/s (1.0+1.0)
Defense: 5 (4+1)
Spell Reflection: 10%
Damage: 5
Rating: 15%
Speed: 3.0
Attack: 1.0
Level: 9/18
Experience: 187323/320000




Juliet equips Dagger.
 Juliet uses Backstab of Dagger on Malek.
 Malek takes 25 damage of Juliet's Backstab.
 Malek is poisoned by Juliet's Pot of Poison.
 Malek takes 2 damage of Juliet's Pot of Poison.
 Malek takes 2 damage of Juliet's Pot of Poison.
 Malek takes 2 damage of Juliet's Pot of Poison.
 Malek takes 2 damage of Juliet's Pot of Poison.
 Malek speaks to Juliet.




Name: Juliet
Race: Elf
Class: Assassin / Thief
Strength: 10 (9+1)
Agility: 18 (15+3)
Intelligence: 14 (12+2)
Item: Dagger
Ability: Backstab
Life: 100/100 (90+10)
Regeneration: +5.0/s (3.5+1.5)
Mana: 40/50 (40+10)
Regeneration: +2.5/s (1.5+1.0)
Defense: 5 (4+1)
Evasion: 10% (5+5)
Damage: 25 (20+5)
Rating: 15% (10+5)
Speed: 5 (3.0+2.0)
Attack: 2.0 (1.0+1.0)
Level: 4/18
Experience: 2563/5000








Grundlegende Struktur einer Engine

Denken in Systemen und Schnittstellen

Modulen und Komponenten

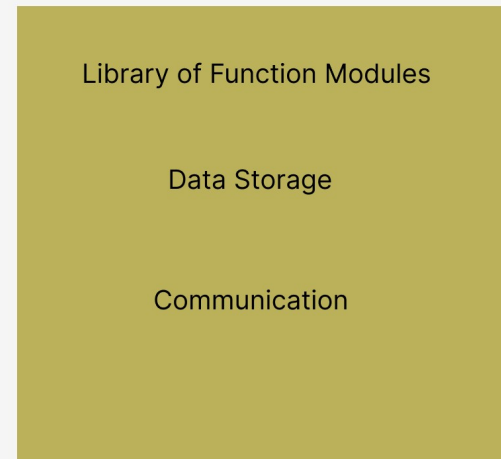
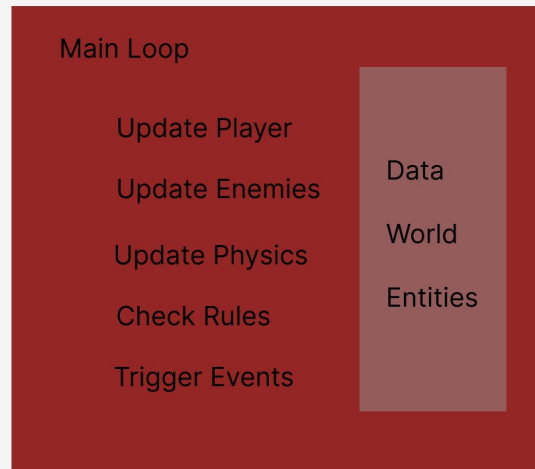
Definitionen und Strukturen

Interne Abstraktion -> Sinnvolle Representation

Input



Timer



Render Data Output

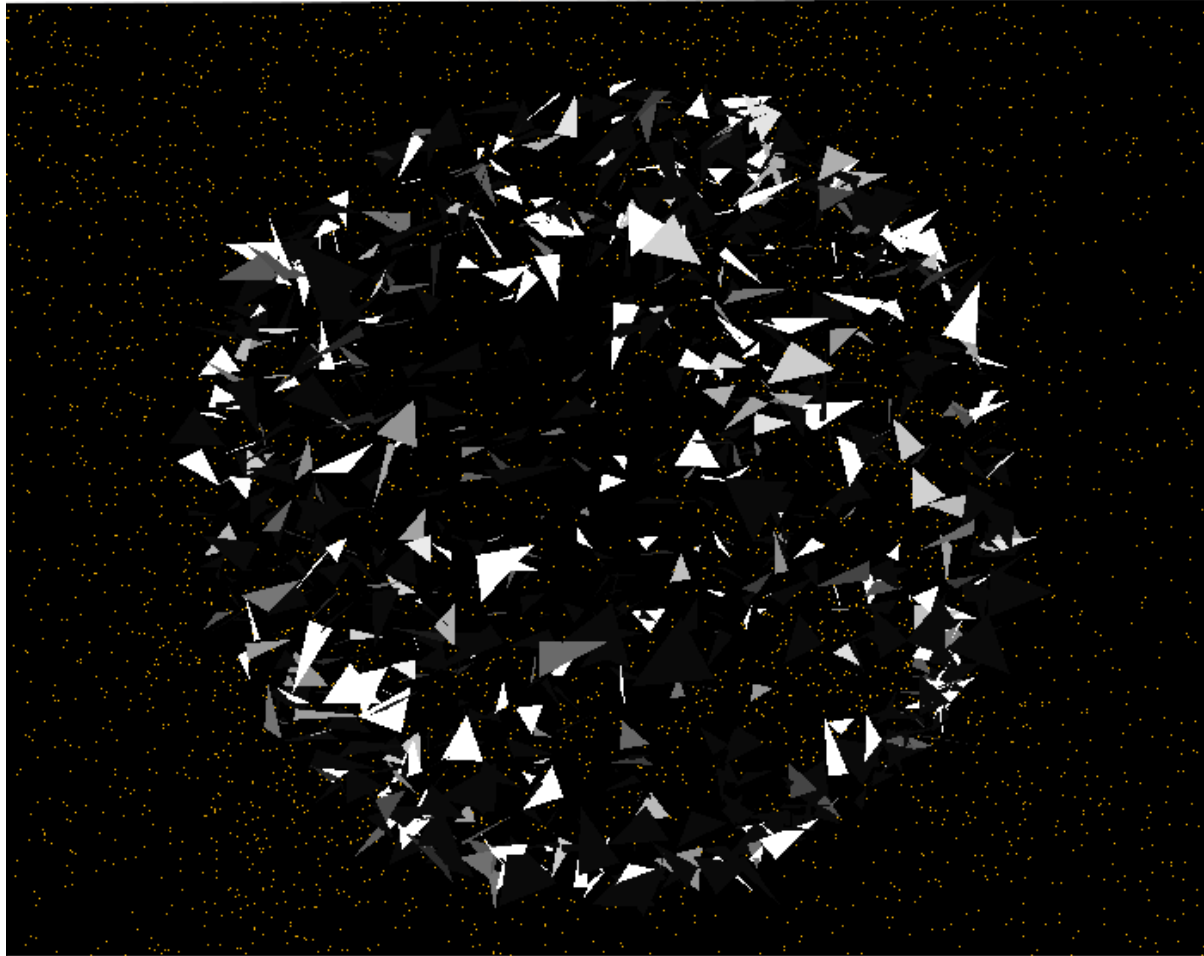
Game Life Cycle

Eingabe, Timer → Update Zyklus der Systeme

Verarbeitung, Bewegung und Übersetzung

Ausgabe einer übersetzten Wirklichkeit

Realtime Demo



Vorteil Browser Game

Plattformübergreifendes Publikum

Schnelles Prototyping und Feedback

Nahtlose Integration in großes Ökosystem von online Services

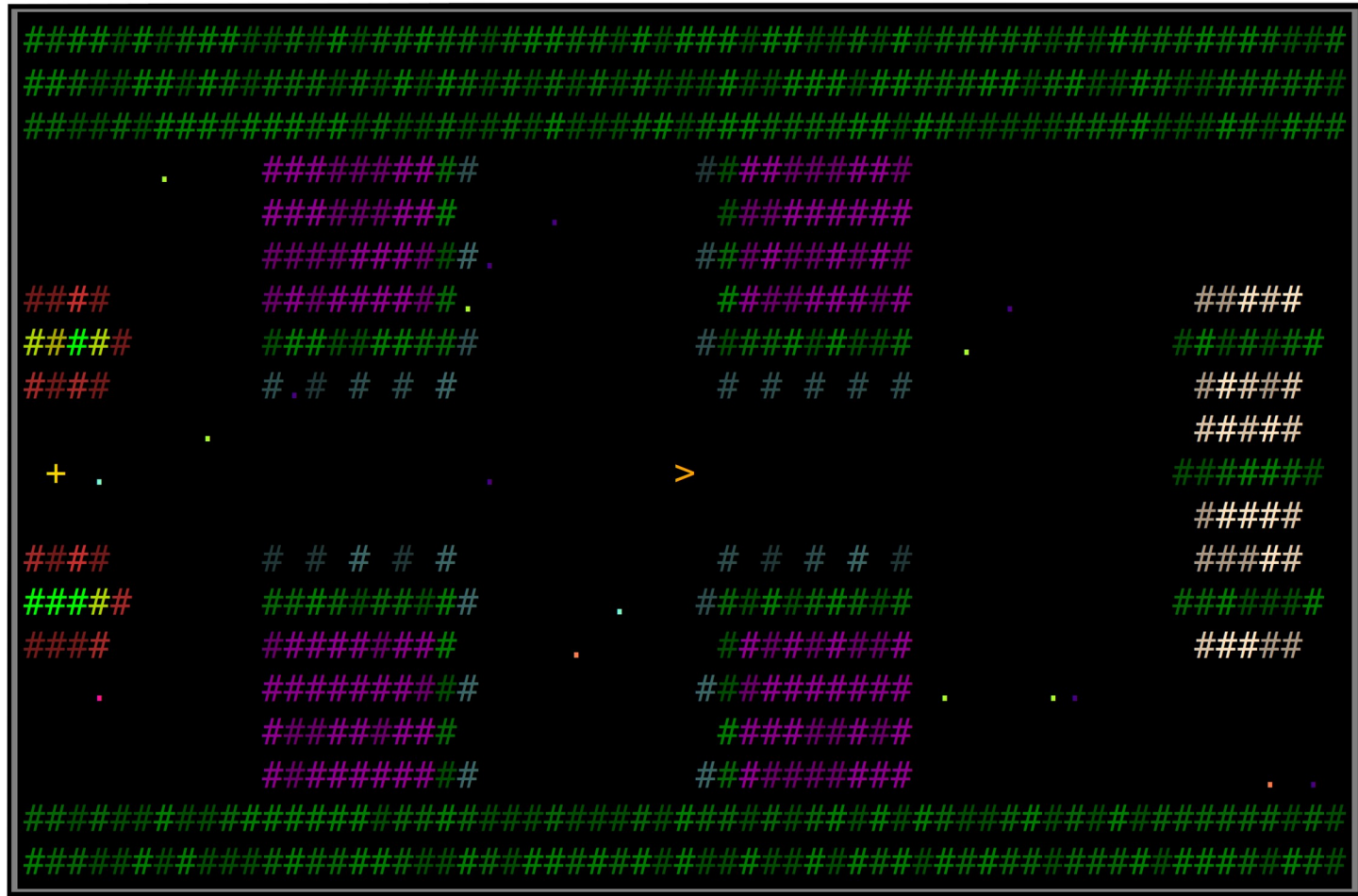
Scriptsprachen sind inzwischen schnell genug und mächtig

Nutzung von Web development für User Interfaces in Spielen

Einfaches modding von Spielen wegen weitverbreiteten Standards

Frameworks und Laufzeitumgebungen für Stand-Alone Application

Realtime Demo



Building System Layers

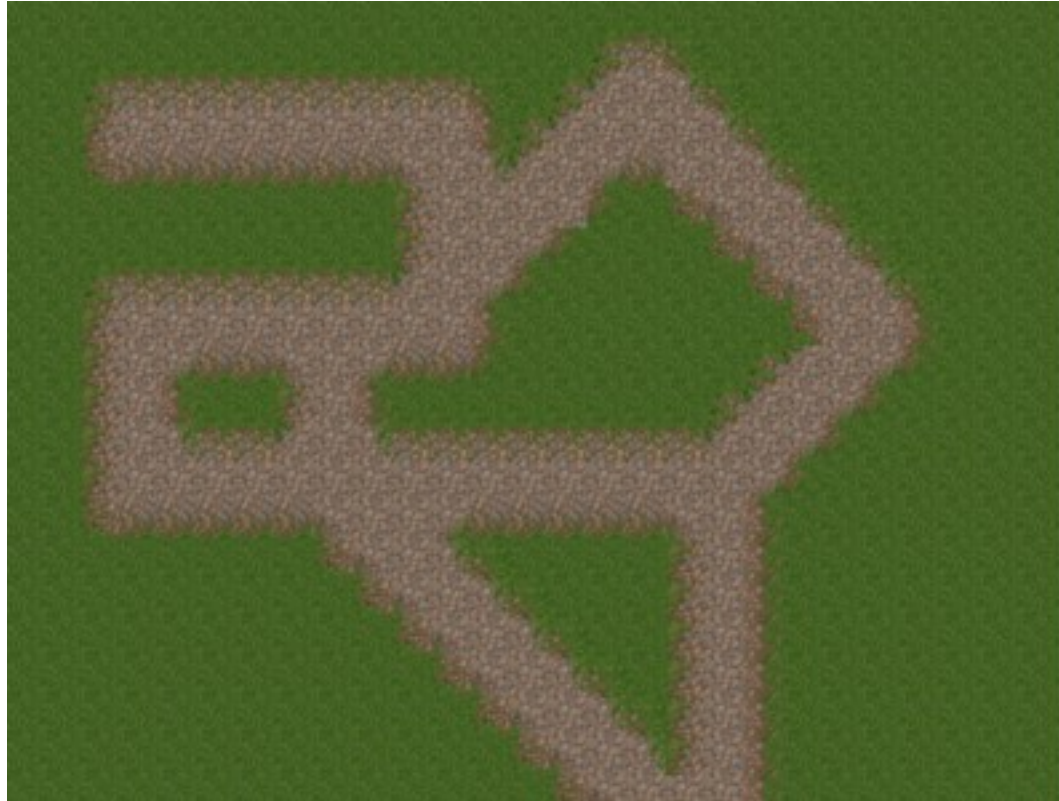
Übersetzung von Datenschichten

In höherwertige Representation

Übersetzung von Logikschichten

In höherwertige Funktion

Realtime Demo



```

594 path27:
595 [
596     "11111      55555      555555",
597     "111111     5555555555555555  ",
598     "11111111    5555555555555555  ",
599     "1111111111   5555      4",
600     "1111111111   444",
601     "11111      4444444",
602     "      6666      444444",
603     "      6666666666      4444",
604     "      6666666666666666",
605     "      222      666666",
606     "22222222     6666666      33",
607     "2222222     66666666      3333",
608     "222222     666666666      33333",
609     "22222     6666666666      333333",
610     "2222     6666666      333",
611 ],
612
613
614 path28:
615 [
616     "3333333316      6133333333",
617     "333222331      133222333",
618     "3332223316      6133222333",
619     "333333331      133333333",
620     "1111111116      611111111",
621     "6 6 6 6 6      6 6 6 6 6",
622     "      ",
623     "      ",
624     "      ",
625     "6 6 6 6 6      6 6 6 6 6",
626     "1111111116      611111111",
627     "333333331      133333333",
628     "3332223316      6133222333",
629     "333222331      133222333",
630     "3333333316      613333333",
631 ],
632 path29:
633 [
634     "      ",
635     "      ",
636     "      ",
637     "      11111",
638     "      12221",
639     "      12221",
640     "1111111122211111111",
641     "1222222222222222221",
642     "1111111122211111111",
643     "      12221",
644     "      12221",
645     "      11111",
646     "      ",
647     "      ",
648     "      ",
649 ],

```

Map Tile Data


```

1968
1969
1970 function UpdateInput() {
1971     let u = unit[player1];
1972
1973     if (key["ArrowUp"] || key["KeyW"]) {
1974         if (u.position.y > 0 && GetMap(u.position.x, u.position.y - 1) == type_empty && (u.position.d == look_up || (key.ArrowDown || key.ArrowLeft || key.ArrowRight))) {
1975             SetMap(u.x, u.y, type_empty);
1976             SetMapColor(u.x, u.y, color.empty);
1977             u.position.y--;
1978             SetMap(u.x, u.y, type_player);
1979             SetMapColor(u.x, u.y, color.player);
1980
1981             screenPosY--;
1982         }
1983         if (!key.Space && !key.ControlLeft && !key.ControlRight && !key.ShiftLeft && !key.AltLeft && !key.ArrowLeft && !key.ArrowRight) {
1984             u.graphic.current = u.graphic.up
1985             u.position.d = look_up
1986         }
1987     }
1988
1989     if (key["ArrowDown"] || key["KeyS"]) {
1990         if (u.position.y < mapRows - 2 && GetMap(u.position.x, u.position.y + 1) == type_empty && (u.position.d == look_down || (key.ArrowUp || key.ArrowLeft || key.ArrowRight))) {
1991             SetMap(u.x, u.y, type_empty);
1992             SetMapColor(u.x, u.y, color.empty);
1993             u.position.y++;
1994             SetMap(u.x, u.y, type_player);
1995             SetMapColor(u.x, u.y, color.player);
1996
1997             screenPosY++;
1998         }
1999         if (!key.Space && !key.ControlLeft && !key.ControlRight && !key.ShiftLeft && !key.AltLeft && !key.ArrowLeft && !key.ArrowRight) {
2000             u.graphic.current = u.graphic.down
2001             u.position.d = look_down
2002         }
2003     }
2004
2005     if (key["ArrowLeft"] || key["KeyA"]) {
2006         if (u.position.x > 0 && GetMap(u.position.x - 1, u.position.y) == type_empty && (u.position.d == look_left || (key.ArrowDown || key.ArrowUp || key.ArrowRight))) {
2007             SetMap(u.x, u.y, type_empty);
2008             SetMapColor(u.x, u.y, color.empty);
2009             u.position.x--;
2010             SetMap(u.x, u.y, type_player);
2011             SetMapColor(u.x, u.y, color.player);
2012
2013             screenPosX--;
2014         }
2015         if (!key.Space && !key.ControlLeft && !key.ControlRight && !key.ShiftLeft && !key.AltLeft && !key.ArrowUp && !key.ArrowDown) {
2016             u.graphic.current = u.graphic.left
2017             u.position.d = look_left
2018         }
2019     }
2020
2021     if (key["ArrowRight"] || key["KeyD"]) {
2022         if (u.position.x < mapColumns - 2 && GetMap(u.position.x + 1, u.position.y) == type_empty && (u.position.d == look_right || (key.ArrowDown || key.ArrowLeft || key.ArrowUp))) {
2023             SetMap(u.x, u.y, type_empty);
2024             SetMapColor(u.x, u.y, color.empty);
2025             u.position.x++;
2026             SetMap(u.x, u.y, type_player);
2027             SetMapColor(u.x, u.y, color.player);
2028
2029             screenPosX++;
2030         }
2031     }
2032 }

```

Input Controls

```

function UpdateShots() {
    let d = 0, x = 0, y = 0, r = 0;
    let u = unit[player1], start = u.shots;

    for (let i = start; i < u.shot.length; i++) {
        next = u.shot[u.shots];
        s = u.shot[i];
        if (s.d != look_none) {
            if (s.d == look_up) {
                s.y--;
                let tile = GetMap(s.x, s.y);
                if (tile != type_empty || s.y < 0 || s.r <= 0) {
                    if (tile != type_block) {
                        SetMap(s.x, s.y, type_empty);
                        SetMapColor(s.x, s.y, color.empty)
                    }
                    s.d = look_none;

                    d = next.d;
                    x = next.x;
                    y = next.y;
                    r = next.r;
                    next.d = s.d;
                    next.x = s.x;
                    next.y = s.y;
                    next.r = s.r;
                    s.d = d;
                    s.x = x;
                    s.y = y;
                    s.r = r;

                    u.shots++;

                    // console.log("Shots Recover:" + String(u.s
                } else {
                    s.r--
                }
            } else if (s.d == look_down) {
                s.y++;
                let tile = GetMap(s.x, s.y);
                if (tile != type_empty || s.y >= mapRows || s.r
                    if (tile != type_block) {

```

Projektil Update

```

2318 function GetColor(color) {
2319     return color[0]
2320 }
2321
2322 function GetColorVariant(color_object) {
2323     return color_object[RandomInt(0, color_object.length)]
2324 }
2325
2326
2327 function RandomInt(min, max) {
2328     return min + Math.floor(Math.random() * (max - min))
2329 };
2330
2331 function SetRandomMap(type, color) {
2332     let x = Math.floor(Math.random() * mapColumns);
2333     let y = Math.floor(Math.random() * mapRows);
2334     let u = unit[player1].position;
2335     if ((x > 0 && x < mapColumns && y > 0 && y < mapRows) && !(x == u.x && y == u.y) && (GetMap(x, y) == type_empty)) {
2336         SetMap(x, y, type);
2337         SetMapColor(x, y, color);
2338     }
2339 }
2340
2341 function SetRandomGround(type, color) {
2342     let x = Math.floor(Math.random() * mapColumns);
2343     let y = Math.floor(Math.random() * mapRows);
2344     let u = unit[player1].position;
2345     if ((x > 0 && x < mapColumns && y > 0 && y < mapRows) && !(x == u.x && y == u.y) && (GetMap(x, y) == type_empty)) {
2346         SetGround(x, y, type);
2347         SetGroundColor(x, y, color);
2348     }
2349 }
2350
2351 function InitMaze() {
2352     for (let y = 0; y < miniMap.length; y++) {
2353         for (let x = 0; x < miniMap[y].length; x++) {
2354             SetMaze(x, y, GetMini(x, y))
2355         }
2356     }
2357 }
2358
2359
2360 function GetRandomTilePath() {
2361     if (Math.random() > 0.50) {
2362         return pathSet[0]
2363     } else {
2364         return pathSet[RandomInt(0, pathSet.length)]
2365     }
2366 }
2367
2368 function GetRandomTileColor() {
2369     return tileColor[RandomInt(0, tileColor.length)];
2370 }
2371
2372 function BuildMaze(type, color) {
2373     let index = 0;
2374     let mazeTile = " ";
2375     let mapTile = " ";
2376     let pathTile = pathSet[0];
2377     let pathColor1 = tileColor[0],
2378         pathColor2 = tileColor[1]

```

Common Use Functions

Newest Game Project



Vergiss nicht wie weit du in den letzten 3 Monaten
gekommen bist.

Hättest du dir jemals vorstellen können was du jetzt
kannst?

Stell dir vor was du in 3 Jahren sein kannst!