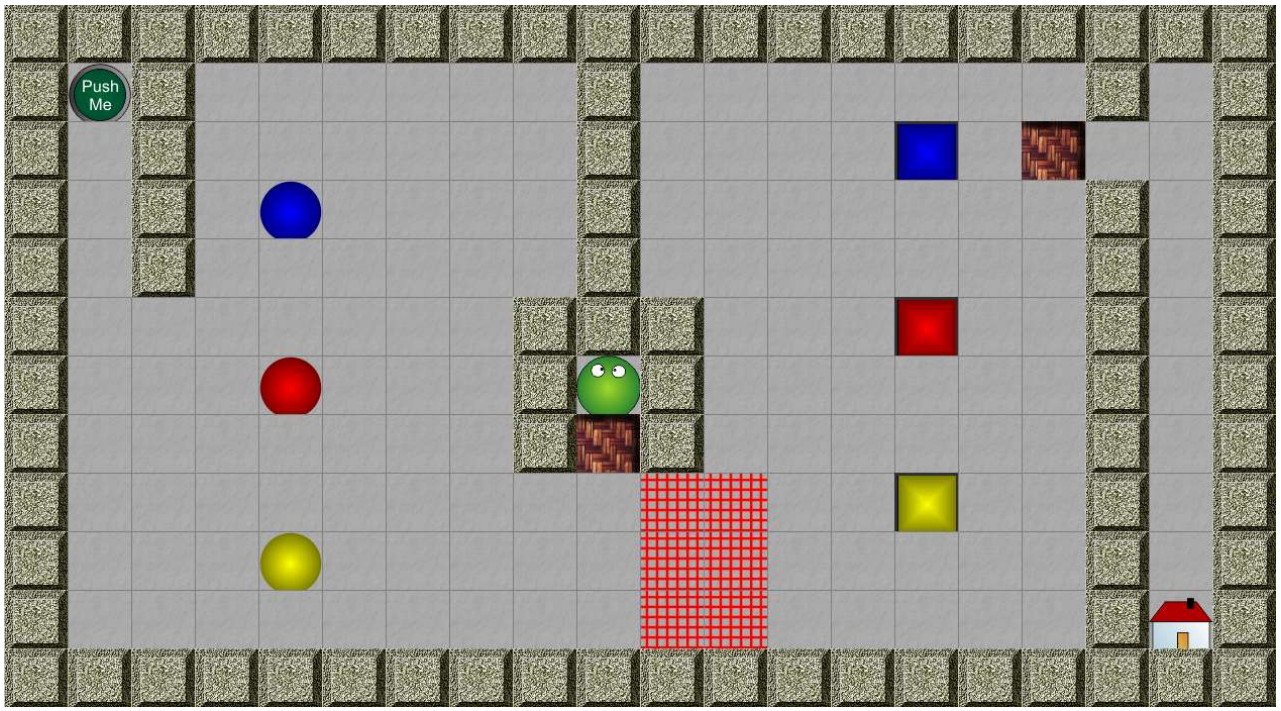
Jannis Görlinger und Sören Seeger
BG - Technik
12BG01 – Q2
Mai - Juni 2018



PUSHY-REVIVAL

SPIELEENTWICKLUNG – BG TECHNIK

Entwickelt von Jannis Görlinger und Sören Seeger im Rahmen der Technologie Projektarbeit „Entwicklung eines einfachen Computerspiels mit Java“.

INHALTSVERZEICHNIS

Inhaltsverzeichnis

Projektvorfeld

EIGENE KENNTNISSE

Allgemeine Erfahrung mit/ im Umgang mit der Programmiersprache Java

Kenntnisse im Bereich der einfachen GUI-Programmierung (Java AWT/Swing Objekte, ActionListener)

Wissen über Exceptions, Algorithmen, Vererbung, Klassen & Objekten etc.

VORAUSSETZUNGEN

Kenntnisse und Analyse der originalen Spiellogik.

BESCHREIBUNG DES ORIGINAL SPIELES

Pushy ist ein Spiel, welches in dem pädagogischen Programm „Lernwerkstatt“ enthalten ist. Es ist dafür konzipiert, das logische und analytische Denken zu trainieren und zu fordern. Pushy ist ein 2D-Spiel und wurde von Ralf zur Linde (Bild) 2002 konzipiert. Das Ziel bei Pushy ist es, mit einer Spielfigur (Pushy) in ein Ziel (Pushys Haus) zu kommen. Dazu müssen vorher Rätsel gelöst werden. Die Rätsel sind vielfältig und teilweise auch sehr anspruchsvoll.

Bildquelle: https://de.wikipedia.org/wiki/Ralf_zur_Linde#/media/File:Ralf_zur_Linde.jpg

Vorläufige Projektziele

Inhaltliche Ziele

Neuauflage des bekannten Lernwerkstatt-Spieles „Pushy“.

Unser Ziel: Komplett durch „eigenen Aufwand“ ein eigenständiges Spiel zu entwickeln.

Bedeutet, dass wir uns vornehmen jeglichen Teil des Programms selbst zu kreieren.

Darunter fallen sowohl die Programmlogik, Aufbau, Funktionen als auch das Design und alle grafischen Elemente!

Als vorläufiges Projektziel sehen wir die Implementierung der Grundlogik, der grafischen Benutzeroberfläche des Spielfeldes, das Erschaffen eigener grafischer Elemente, die Steuerung der Spielfigur über die Tastatureigabe und das integrieren von mindestens einer Art von Hindernis im Spielfeld.

Persönliche Ziele

Erweiterung der eigenen Kenntnisse im Bereich der GUI-Programmierung in Java als auch die Vertiefung von „Grundwissen“ (z.B Switch Case, Algorithmen, Objekte, Exeptions, Zugriffsrechte etc.).

Außerdem sehen wir die Möglichkeit neue Klassen zu verwenden, Dinge „auszuprobieren“ und somit das Projekt zu verbessern.

Realisierungsphase/ Dokumentation

STRUKTUR DES PROGRAMMES

Klassen im Package „game“

Run → Ruft den Mainscreen auf

Mainscreen → Willkommensfenster, Level Auswahl / Level Fortschritt

Playground → Spielfläche und Spiellogik

Succsess → Optionsfenster nach Bestehen des Levels [Thread]

ConvertImage2Pane → Konvertiert die jeweiligen Grafiken zu Panels um sie zum Layout hinzuzufügen

Levelmanager → Verwaltung/ Abbildung der einzelnen Level-Sitemaps

PlayerStatistic → Speichert und Lädt den Spielvortschritt in/aus einem Textdokument

Sound → Spielt zufällig einen von fünf Sounds beim Beenden des Levels ab [Thread]

HelpScreen → Gibt Hilfestellungen bei Verständnisproblemen

Die Resourcen befinden sich in einem separaten Resourcen-Ordner (Bild-Resourcen) und direkt im Package (Sound-Resourcen).

Threads

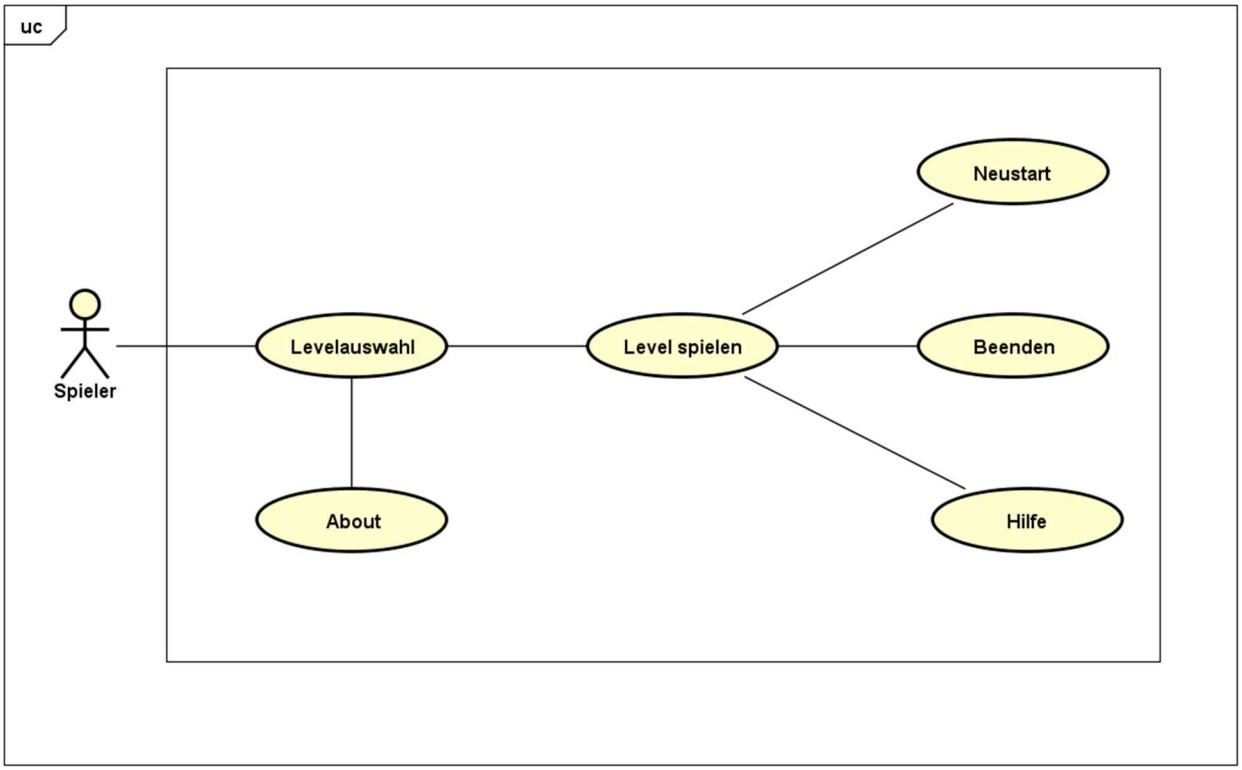
Alle Klassen, welche in der obigen Darstellung mit [Thread] deklariert sind sind als solche implementiert.

Grund dafür ist die nötige Nebenläufigkeit dieser Funktionen.

Der Sound und das Succsess-Fenster könnten sonst nicht während der Animation gespielt/ angezeigt werden.

UML-DIAGRAMME

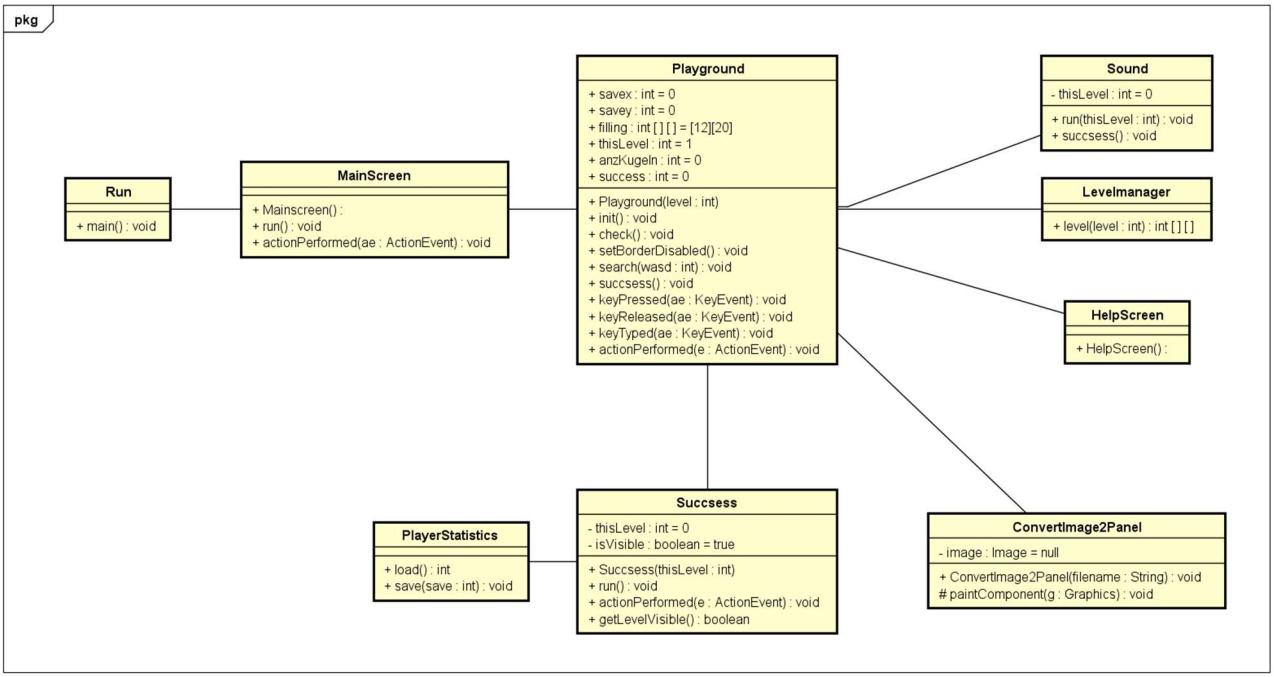
Anwendungsfall-Diagramm für das Spiel



Der Spieler kommt über die Levelauswahl zum eig. Spiel. Zusätzlich hat er die Möglichkeit eine „About-Seite“/ Informationsseite aufzurufen.

Außerdem kann der Spieler das Level neustarten, es beenden und zusätzlich noch eine Hilfe-Seite öffnen.

Klassendiagramme und deren Verknüpfung



Info: Auf die Zugriffsberechtigungen wurde bewusst verzichtet, da alle Klassen, alle Objekte und alle Methoden als „public“ implementiert wurden.

PHASEN DER ENTWICKLUNG

Stufe 1 – GUI/ Layout

In der ersten Phase der Entwicklung ging es darum ein Konzept für die grafische Darstellung zu schaffen.

Ziel war es das Level Fenster zu gestalten.

Zur Auswahl standen mehrere GUI Layouts, welche sich als unterschiedlich effektiv/ leicht zu handhaben herausgestellt haben.

Ein Layout, welches zu unserer ersten Vorstellung passte (Gridlayout mit x-y Koordinaten/jederzeit kann in einer Zelle der Inhalt geändert werden) gab es nicht.

Wir legten uns auf das GridLayout fest und wählten als Darstellung der Level-Map ein 2d-Array.

Anschließend entwickelten wir einen Algorithmus, welcher das 2D Array in eine 1D-Kette konvertiert, da das Gridlayout nur linear befüllt werden kann:

```
for (int i = 0; i < 240; i++) {
        x= i/20;
        y= i%20;

switch(filling[x][y])
```

Eine Schleife geht die 1D Kette des 2D Arrays durch (12*20 = 240).
Zum befüllen der Zellen 0-239 im GridLayout wird jeweils nachgesehen, welche Ziffer(Grafik-Information) an dieser Stelle steht *(siehe Grafik 1D-2D Zuordnung)*.
Um für eine Zelle 0-239 die passende x-Stelle im 2D Array zu finden rechnet man:
Zelle/20
(Die Zeilen sind 20 lang → durch das Teilen durch diese 20 Kästchen erhält man auch für alle Werte über 20 den richtigen x-Wert)

Um für eine Zelle 0-239 die passende y-Stelle im 2D Array zu finden rechnet man:
Zelle%20 (-modulo)
(Der Rest (modulo) der Division aus der x-Wert Berechnung, gibt die Information, wie oft der Wert (0-239) in die 20 Kästchen reinpasst → somit den y-Wert).

2D Array in [x][y] Form

0|0	0|1	0|2	0|3	0|4	0|5	0|6	0|7	0|8	0|9	0|10	0|11	0|12	0|13	0|14	0|15	0|16	0|17	0|18	0|19
1|0	1|1	1|2	1|3	1|4	1|5	1|6	1|7	1|8	1|9	1|10	1|11	1|12	1|13	1|14	1|15	1|16	1|17	1|18	1|19
2|0	2|1	2|2	2|3	2|4	2|5	2|6	2|7	2|8	2|9	2|10	2|11	2|12	2|13	2|14	2|15	2|16	2|17	2|18	2|19
3|0	3|1	3|2	3|3	3|4	3|5	3|6	3|7	3|8	3|9	3|10	3|11	3|12	3|13	3|14	3|15	3|16	3|17	3|18	3|19
4|0	4|1	4|2	4|3	4|4	4|5	4|6	4|7	4|8	4|9	4|10	4|11	4|12	4|13	4|14	4|15	4|16	4|17	4|18	4|19
5|0	5|1	5|2	5|3	5|4	5|5	5|6	5|7	5|8	5|9	5|10	5|11	5|12	5|13	5|14	5|15	5|16	5|17	5|18	5|19
6|0	6|1	6|2	6|3	6|4	6|5	6|6	6|7	6|8	6|9	6|10	6|11	6|12	6|13	6|14	6|15	6|16	6|17	6|18	6|19
7|0	7|1	7|2	7|3	7|4	7|5	7|6	7|7	7|8	7|9	7|10	7|11	7|12	7|13	7|14	7|15	7|16	7|17	7|18	7|19
8|0	8|1	8|2	8|3	8|4	8|5	8|6	8|7	8|8	8|9	8|10	8|11	8|12	8|13	8|14	8|15	8|16	8|17	8|18	8|19
9|0	9|1	9|2	9|3	9|4	9|5	9|6	9|7	9|8	9|9	9|10	9|11	9|12	9|13	9|14	9|15	9|16	9|17	9|18	9|19
10|0	10|1	10|2	10|3	10|4	10|5	10|6	10|7	10|8	10|9	10|10	10|11	10|12	10|13	10|14	10|15	10|16	10|17	10|18	10|19
11|0	11|1	11|2	11|3	11|4	11|5	11|6	11|7	11|8	11|9	11|10	11|11	11|12	11|13	11|14	11|15	11|16	11|17	11|18	11|19

Konvertierung in 1D Kette:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Kette (Zeile für Zeile) im GridLayout:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239

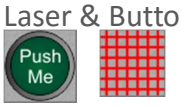
Stufe 2 – Grafiken und deren Logik

In der 2. Stufe ging es darum, die Hindernisse für die Level zu gestalten und deren Funktion zu implementieren. Wir haben uns auf drei Arten von Hindernissen festgelegt, welche selbst in CorelDRAW gestaltet wurden (Projektdateien im Anhang):

Holzblock:

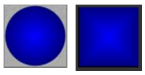
Der Holzblock ist das einfachste Hindernis. Er kann von Pushy geschoben werden, jedoch nur, wenn vor dem Holzblock frei ist. Der Holzblock kann dazu genutzt werden, um Eingänge zu versperren oder Begrenzungen darzustellen

Laser & Button:



Der Laser stellt eine Art Grenze dar, die vorerst nicht überwunden werden kann. Nur durch Drücken des Buttons mit der Aufschrift „Push Me“ kann der Laser ausgeschaltet und somit überquert werden.

Bälle & Ziele:



Die Bälle in den Farben Blau, Rot und Gelb müssen in das jeweilig farbig passende Ziel gebracht werden. Bevor dies nicht passiert ist, kann Pushy sein Haus nicht betreten. Mit der Methode check() werden die Kugeln gezählt und bei versenken im Ziel wird eine Variable dezimiert. Wenn diese Variable 0 ist, kann Pushy in sein Haus.

Stufe 3 - Benutzerfreundlichkeit

In der 3. Stufe war es unser Ziel, die benutzerfreundliche Oberflächengestaltung zu realisieren. Darunter fallen zum Beispiel der Startbildschirm mit Levelauswahl, die Menubar im Spielfenster und das Fenster, welches nach Gewinnen des Levels erscheint. Probleme gab es bei der Implementierung der Soundabspielung, da wir dies noch nicht im Unterricht behandelt hatten und uns die Kenntnisse selbst aneignen mussten.

Stufe 4 - Optimierung

Stufe 4 war größtenteils geprägt von Optimierung und Verbesserungen des Programmcodes. Es gab kleinere und größere Probleme, die den Spielablauf behinderten und gelöst werden mussten. Auch die Implementierung eines GIFs stellte uns vor eine Herausforderung, da der ursprünglich bevorzugte Lösungsweg nicht realisierbar war.

Testphase

Die Testphasen wurden nicht explizit geplant, sondern fanden schon während des Programmierens statt. Dabei wurden häufig kleinere Fehler gefunden, welche aber meist schnell behoben werden konnten.

Vor besondere Herausforderungen stellten uns Probleme, wie Zugriffsrechte und Erbbeziehungen. Diese konnten aber auch gelöst werden. Da am Ende ein Art GIF und ein Sound gleichzeitig abgespielt werden sollten, mussten wir uns noch mit Threads beschäftigen, um dies zu ermöglichen. Auch das Einsetzen eines Images in ein Panel konnte realisiert und umgesetzt werden.

Als Testphase kann der Abschnitt Phasen der Entwicklung herangezogen werden.

Projekterfolg/ Zusammenfassung

Trotz einzelner Veränderungen konnte das Projektziel erfüllt werden und sogar noch um zusätzliche Funktionen erweitert werden(Soundausgabe, Level Auswahl, etc.). Die Implementierung stellte uns vor Herausforderungen, die jedoch gelöst werden konnten.

Insgesamt sind wir mit dem Projektverlauf sehr zufrieden und denken, dass so eine frei gestaltete Unterrichtseinheit auch für kommende Klassen eine gute Abwechslung zum normalen Unterricht ist.

Der gesamte Zeitaufwand beläuft sich auf über 25 Stunden.

Ausblick

In Zukunft könnte das Programm noch um einige Funktionen ergänzt werden.

Vorgestellt haben wir uns einen Multiplayer (ggf. über das Netzwerk), einen grafischen Level-Designer und die Ergänzung um weitere Spielobjekte(Teleporter, Farbeimer, Laser, Wasser, Aktionsfelder uvm.).

Für uns war wichtig, dass das „Grundkonzept“ steht, welches durch leichtes Modifizieren/ Hinzufügen erweitert werden kann!

Informationen

SÖREN SEEGER 12BG01	JANNIS GÖRLINGER 12BG01
	
soeren.seeger@gmx.de	jannisgoerlinger@web.de

Berufliches Gymnasium
Eschwege - Technik
Am Südring 12
37269 Eschwege

Berufliches Gymnasium
Technik