Gamesmacher-Prototyp: Dokumentation

von Arne Sibilis
Unity-Version 2020.3.11f1
Stand: 8. Juli 2021

1. Einstellungen

In der Datei „tasks.json“ im Unterordner *„StreamingAssets“** können diverse Einstellungen vorgenommen werden. Vor allem wird dort die gesamte Progression des Spiels als *„tasks“* abgelegt. In der folgenden Tabelle werden alle Einstellungsmöglichkeiten kurz erklärt.

**Zu finden in Windows-Builds unter „Gamesmacher_Data“, in MacOSX-Builds unter „Contents/Resouces/Data“.*

Hinweis: Enthält die Datei „tasks.json“ Syntaxfehler oder kann nicht gefunden werden, wird eine Fehlermeldung angezeigt, die auf das Problem hinweist. Das Spiel startet in diesem Fall nicht.

Parameter	Datentyp	Beschreibung
version	String	Versionsnummer der Datei zur Übersicht (optional).
gameMode	String	Der verwendete Spielmodus. Hier eine Übersicht der verfügbaren Modi. *default*: Standardmodus. Textboxen müssen manuell geschlossen werden. *auto*: Alternativmodus. Textboxen werden nach kurzer Zeit automatisch geschlossen und Pixl kann sich bewegen, während Textboxen aktiv sind. *quick*: Testmodus. Textboxen werden geschlossen, sobald der Text vollständig angezeigt wurde und Pixl hat von Anfang an alle Fähigkeiten.
startIndex	Integer	Der Aufgabenindex, bei dem das Spiel beginnt. Alle Kommandos von vorhergehenden Aufgaben werden im Schnelldurchlauf bearbeitet. **Siehe: 2. Definieren von Aufgaben**
textColor	String	Die Standardtextfarbe in Textboxen im Hex-Format *(#RRGGBBAA).*
textBoxFont	String	Die Schriftart, in der Inhalte von Textboxen dargestellt werden. Derzeit kann zwischen *Helvetica* (Helvetica Neue) und *Roboto* (Roboto Slab) gewählt werden.
textBoxColor	String	Die Hintergrundfarbe von Textboxen im Hex-Format *(#RRGGBBAA).*
textBoxSpeed	Float	Die Geschwindigkeit, mit der der Text in Textboxen angezeigt wird in Zeichen pro Sekunde.
textBoxSize	Float	Die Skalierung von Textboxen.
textBoxPosition	String	Die Position, an der Textboxen angezeigt werden. Möglich sind die Standardeinstellung *„center“* sowie *„bottom“*, bei der die Textboxen im Boden angezeigt werden (hierfür sollte die *textBoxSize* nicht zu groß gewählt werden).

Parameter	Datentyp	Beschreibung
tasks	TaskData-Array	Ein Array aus TaskData-Objekten. Wird chronologisch abgearbeitet und gibt die Progression vor. **Siehe: 2. Definieren von Aufgaben**
soundLibrary	SoundData-Array	Ein Array aus SoundData-Objekten zum Überschreiben vorhandener Soundeffekte, bestehend aus *„name“* (zur Identifikation des Effekts), *„file“* (zur Zuordnung einer *.wav-Datei und *„volume“* (zum Festlegen der relativen Lautstärke). **Siehe: 5. Ersetzen von Soundeffekten**

2. Definieren von Aufgaben

Alle im Spiel auftauchenden Texte und Aufgaben können unter *„tasks“* als *„TaskData“-Objekte* definiert werden. Texteinblendungen können hierbei ebenfalls als Aufgaben verstanden werden. TaskData-Objekte enthalten mehrere Parameter, von denen jedoch, je nach Aufgabentyp, immer nur eine kleine Auswahl relevant ist. Die folgende Tabelle enthält eine Übersicht aller Aufgabentypen und beschreibt, welche Parameter in welchen Fällen angegeben werden sollten.

Bei Spielstart wird die Liste an Aufgaben chronologisch abgearbeitet. Wird eine Aufgabe erfüllt, startet die nächste. Wurde ein *startIndex* > *0* gewählt, springt das Spiel beim Start zu der Aufgabe mit dem entsprechenden Index und verarbeitet dabei alle Kommandos übersprungener Aufgaben.

Parameter	Datentyp	Beschreibung
type	String	Der Typ der Aufgabe. Mögliche Typen sind: message, collect, goto, wait, enter, perform. Je nach Typ müssen Bedingungen erfüllt werden, damit die Aufgabe als erledigt gilt. **Siehe: 3. Aufgabentypen**
target	String	Das Ziel der Aufgabe (nur notwendig bei Aufgaben der Typen *collect* und *perform*). Mögliche Ziele sind: right, left, jump, enter, star, key. Beim Typ *collect* entspricht das *target* den Objekten, die gesammelt werden sollen (Skillboxen, Sterne oder Schlüssel), beim Typ *perform* steht es für die erforderliche Aktion (hier wird nur *jump* unterstützt).
text	String	Der Text, der bei Aufgaben des Typs *message* angezeigt wird.
mobileText	String	Der alternative Text, der auf Mobilgeräten angezeigt wird. Wird dieser nicht angegeben, wird der Eintrag bei *„text“* verwendet.
amount	Integer	Bei Aufgaben des Typs *collect*: Die Anzahl an Gegenständen, die gesammelt werden muss. Dabei zählt die Gesamtanzahl inklusive bereits erhaltener Gegenstände. Bei Aufgaben des Typs *wait*: Die Zeit in Sekunden, die gewartet wird, bis es weitergeht.
x	Integer	Die X-Koordinate der Zielkachel (nur notwendig bei Aufgaben des Typs *goto*).
y	Integer	Die Y-Koordinate der Zielkachel (nur notwendig bei Aufgaben des Typs *goto*).
commands	String-Array	Ein optionales Array von Kommandos, die beim Erhalt der Aufgabe ausgeführt werden. Kann mit allen Aufgabentypen verwendet werden. **Siehe: 4. Kommandos**

3. Aufgabentypen

Jeder Aufgabentyp erfordert das Erfüllen einer bestimmten Bedingung, um abgeschlossen zu werden. Welche Bedingung dies ist und welche Parameter wichtig sind, um diese klar zu definieren, kann folgender Tabelle entnommen werden.

Typ	Beschreibung
message	Zeigt eine Textbox mit dem unter „text“ bzw. „mobileText“ angegebenen Inhalt an. Gilt als erfüllt, wenn die Textbox geschlossen wird.
collect	Erfordert das Sammeln eines Gegenstands vom bei „target“ angegebenen Typs. Soll mehr als ein Exemplar gesammelt werden, muss die Anzahl bei „amount“ spezifiziert werden. Dabei werden die Gegenstände global erfasst, d.h. wenn eine Aufgabe das Sammeln von 4 Sternen erfordert und zuvor bereits 3 gesammelt wurden, ist die Aufgabe nach dem Einsammeln eines weiteren Sterns erfüllt. Gilt als erfüllt, wenn genügend Exemplare des verlangten Typs im Inventar registriert wurden. Ist der zu sammelnde Gegenstand eine Fähigkeit, gilt die Aufgabe nach dem Erlernen dieser Fähigkeit als erfüllt (die Angabe bei „amount“ spielt keine Rolle).
goto	Erfordert das Betreten bzw. „Berühren“ der Kachel mit den bei „x“ und „y“ angegebenen Koordinaten. Gilt als erfüllt, sobald Pixl die definierte Kachel betritt.
wait	Die Zeit, die hier gewartet wird, entspricht der Angabe bei „amount“ in Sekunden. In dieser Zeit kann sich Pixl frei bewegen. Sie kann z.B. genutzt werden, um Objekte in der Spielwelt erscheinen zu lassen. Gilt nach Ablauf der angegebenen Zeit automatisch als erfüllt.
enter	Erfordert das Betreten einer (beliebigen) Tür. Wurde zuvor eine Tür durchquert, ohne dass dies von einer Aufgabe verlangt wurde, wird sich dies gemerkt und die Aufgabe kann nachträglich als erfüllt gewertet werden. Gilt als erfüllt, wenn Pixl eine Tür betritt.
perform	Erfordert das Ausführen einer Aktion, die bei „target“ spezifiziert werden muss. Derzeit wird ausschließlich das Springen erkannt, weshalb dieser Aufgabentyp nur mit dem „target“: „jump“ verwendet werden sollte. Gilt als erfüllt, wenn Pixl die definierte Aktion ausgeführt hat.

4. Kommandos

Mit Kommandos kann direkt auf die Spielwelt sowie auf das User Interface Einfluss genommen werden. Auf diese Weise können z.B. in einem geeigneten Moment neue Gegenstände oder Plattformen erschaffen werden.

Wenn Kommandos angegeben werden, muss dies als String-Array geschehen. Dabei muss jedes Kommando als separater String im Array abgelegt werden:
commands: [„spawn character 3 1“, „spawn star 5 3 4 0.5“]

Hinweis: Die Reihenfolge der Wörter innerhalb von Kommandos ist nicht entscheidend, die der numerischen Werte jedoch schon. Werden Werte nicht angegeben, werden sie durch Standardwerte ersetzt (in der Regel 0). Nullen am Ende von Kommandos sind daher meist optional.

Kommando	Beschreibung
set tile [type] [x1] [y1] [x2] [y2] [xDelay] [yDelay] [delay]	Ändert den Typ* aller Kacheln, die im Rechteck zwischen den Koordinaten (x1,y1) und (x2,y2) liegen nach einer Verzögerung von [delay] Sekunden. Dabei wird die Statusänderung jeder Kachel um weitere [xDelay] Sekunden je Spalte und [yDelay] Sekunden je Reihe (von oben zählend) verzögert. *verfügbare Typen: empty, ground Beispiel: set tile empty 0 0 7 5 0.125 0.25 0 Setzt die Kacheln zwischen (0,0) und (7,5) – also alle – auf „empty“ und dreht sie dabei mit einer Verzögerung von 0.125s je Spalte und 0.25s je Reihe um.
Set outlines alpha [alpha] [duration] [delay]	Verändert den Alpha-Wert aller Outlines nach [delay] Sekunden über [duration] Sekunden zu [alpha]. Beispiel: set outlines alpha 0 1 0.5 Blendet die Outlines nach 0.5s Verzögerung über die Dauer einer Sekunde vollständig aus.
draw horizontal outlines [duration] [xyDelay] [delay] draw vertical outlines [duration] [xyDelay] [delay]	Zeichnet nach [delay] Sekunden horizontale bzw. vertikale Outlines über [duration] Sekunden (je Kachel!) mit einer Verzögerung von [xyDelay] Sekunden je Spalte und Reihe. Beispiel: draw horizontal outlines 0.125 0.125 0 Zeichnet mit einer Geschwindigkeit von 0.125s je Kachel und einer Verzögerung von 0.125s je Spalte und Reihe, also gleichmäßig, Outlines um die Kacheln.
spawn character [x] [y] [delay]	Erzeugt Pixl nach [delay] Sekunden an den Koordinaten (x,y). Beispiel: spawn character 3 1 0.25 Erschafft Pixl nach 0.25s an den Koordinaten (3,1).
spawn [type] skill [x] [y] [speed] [delay]	Erzeugt nach [delay] Sekunden eine Skill-Box des angegebenen Typs* an den Koordinaten (x,y) und lässt sie mit [speed] Blocks pro Sekunde nach unten fallen, bis sie den Boden oder eine Plattform erreicht. *verfügbare Typen: moveleft, moveright, jump, enter Beispiel: spawn jump skill 5 3 4 Erschafft ohne Verzögerung (Nullen am Ende sind optional) eine Skill-Box, die Pixl den Sprung beibringt, an den Koordinaten (5,3) und lässt sie mit 4 Blocks pro Sekunde zu Boden fallen.
spawn star [x] [y] [speed] [delay]	Erzeugt nach [delay] Sekunden einen Stern an den Koordinaten (x,y) und lässt ihn mit [speed] Blocks pro Sekunde zu Boden fallen. Statt der X-Koordinate kann auch der Begriff „random“ verwendet werden, um einen Zufallswert zwischen 0 und 7 zu generieren. Beispiel: spawn star 2 4 1.5 0.25 Erschafft nach 0.25s einen Stern an den Koordinaten (2,4) und lässt ihn mit 1.5 Blocks pro Sekunde zu Boden gleiten.

Kommando	Beschreibung
spawn upper platform [x] [y] [duration] [delay] spawn lower platform [x] [y] [duration] [delay]	Erzeugt nach [delay] Sekunden über eine Dauer von [duration] Sekunden eine Plattform an den Koordinaten (x,y). Die Plattform ist zwei Blocks breit und belegt somit auch die Koordinaten (x+1,y). Sie befindet sich je nach Kommando in der oberen oder unteren Kachelhälfte. Beispiel: spawn lower platform 0 3 1 0.25 Erschafft nach 0.25s eine Plattform in der unteren Hälfte der Kacheln (0,3) und (1,3), der Vorgang dauert dabei eine Sekunde.
spawn door [x] [y] [exitX] [exitY] [delay]	Erzeugt nach [delay] Sekunden eine Tür an den Koordinaten (x,y), die Pixl beim Betreten an die Koordinaten (exitX,exitY) befördert. Beispiel: spawn door 2 1 7 5 Erschafft ohne Verzögerung eine Tür an den Koordinaten (2,1), die Pixl nach Betreten in die rechte obere Ecke (7,5) teleportiert.
spawn locked [color] door [x] [y] [exitX] [exitY] [delay]	Funktioniert wie das Kommando „spawn door“, versieht die Tür aber mit einem Schloss in der angegebenen Farbe*. Die Farbe des Schlüssels muss aktuell nicht mit dem des Schlosses übereinstimmen. *verfügbare Farben: red, orange, yellow, green, blue, violet, grey Beispiel: spawn locked blue door 6 4 0 1 0.5 Erschafft nach 0.5s eine Tür mit einem blauen Schloss an den Koordinaten (6,4), die zu den Koordinaten (0,1) führt.
spawn [color] key [x] [y] [speed] [delay]	Erzeugt nach [delay] Sekunden einen Schlüssel der angegebenen Farbe* an den Koordinaten (x,y) und lässt ihn mit [speed] Blocks pro Sekunde fallen. *verfügbare Farben: red, orange, yellow, green, blue, violet, grey Beispiel: spawn yellow key 4 5 3 Erschafft einen gelben Schlüssel an den Koordinaten (4,5) und lässt ihn mit 3 Blocks pro Sekunde zu Boden fallen.
show star [x] [duration] [delay] hide star [x] [duration] [delay]	Zeigt oder versteckt den Stern-Container mit dem Index [x] (gültige Werte: 0 bis 5) im User Interface. Der Vorgang dauert [duration] Sekunden und wird um [delay] Sekunden verzögert. Beispiel: show star 3 1 0.25
show equals [duration] [delay] hide equals [duration] [delay]	Zeigt oder versteckt nach [delay] Sekunden das Gleichheitszeichen im User Interface – fest an den Koordinaten (6,5) – über [duration] Sekunden. Beispiel: hide equals 1 0.5
show reward [duration] [delay] hide reward [duration] [delay]	Zeigt oder versteckt nach [delay] Sekunden die Belohnungsbox im User Interface – fest an den Koordinaten (7,5) – über [duration] Sekunden. Beispiel: show reward 1 0.5

Kommando	Beschreibung
set reward [type]	Legt den Typ der Fähigkeit* fest, die sich in der Belohnungsbox befindet. *verfügbare Typen: moveleft, moveright, jump, enter Beispiel: set reward jump
show stick [duration] show button [duration]	Blendet – sofern das Spiel auf einem Mobilgerät ausgeführt wird – den virtuellen Analogstick bzw. Button über [duration] Sekunden ein, um darauf hinzuweisen. Das Element bleibt aktiv, bis es einmal verwendet wurde. Beispiel: show stick 1
set character [x] [y]	Teleportiert Pixl an die Koordinaten (x,y). Beispiel: set character 7 5
set move speed [s]	Ändert Pixls Bewegungsgeschwindigkeit auf [s] Blocks pro Sekunde. Beispiel: set move speed 3
set jump height [h]	Ändert Pixls Sprunghöhe in Blocks. Beispiel: set jump height 1.75
fade in [duration] fade out [duration]	Öffnet („fade in“) oder schließt („fade out“) die Blende, die auch eingesetzt wird, wenn Pixl durch eine Tür geht, über [duration] Sekunden. Beispiel: fade out 2
remove platforms [duration] [delay] remove doors [delay]	Entfernt nach [delay] Sekunden alle vorhanden Plattformen oder Türen. Bei ersteren muss die Dauer des Verschwindens angegeben werden. Beispiele: remove platforms 2 0.5 remove doors 0.5 Entfernt nach 0.5s alle Plattformen und Türen, wobei die Plattformen über 2s ausgeblendet werden.
load [sceneIndex]	Lädt die Szene mit dem Index [sceneIndex]. Da es aktuell nur eine Szene mit dem Index 0 gibt, genügt das Kommando „load“, um das Spiel neu zu starten. Hinweis: Dieses Kommando wird auch ausgeführt, wenn Escape gedrückt wird. Dabei werden die Datei tasks.json sowie alle Soundeffekte neu eingelesen.

5. Ersetzen von Soundeffekten

Alle im Spiel enthaltenen Soundeffekte können ersetzt werden, indem in der *„soundLibrary“* in der *tasks.json* Objekte mit Informationen zu *„name“, „file“* und *„volume“* (optional) hinzugefügt werden. Der *„name“* identifiziert dabei den Soundeffekt und kann der unterstehenden Tabelle entnommen werden. Bei *„file“* muss der Name der Datei im *.wav-Format (inklusive Dateiendung) angegeben werden. Die Datei selbst sollte im Ordner „*StreamingAssets/audio“* abgelegt werden.

Das Einlesen erfolgt beim Spielstart. Wird eine angegebene Datei nicht gefunden, wird der Standardsound verwendet. Durch das Zuweisen der im Ordner enthaltenen Datei *„silence.wav“* können einzelne Soundeffekte auch deaktiviert werden. Zudem kann die Lautstärke von Standardsounds verändert werden, indem keine *„file“* angegeben und bei *„volume“* ein Wert zwischen 0 und 1 gesetzt wird.

Beispieleinträge:

```
{
    "name": "pixl_jump",
    "file": "pixl_jump.wav",
    "volume": 0.5
},
{
    "name": „skill_collect",
    "file": "silence.wav"
},
{
    "name": „text_show“,
    "volume": 0.25
}
```

(Überschreibt den Sprung-Sound mit der auf halber Lautstärke abgespielten Datei „pixl_jump.wav“, entfernt den Soundeffekt beim Einsammeln einer Skill-Box und setzt die Lautstärke des Effekts beim Einblenden von Text auf ein Viertel.)

Kategorie	Name des Soundeffekts	ggf. Beschreibung
Pixl	pixl_eat	Auffuttern eines Sterns
	pixl_jump	
	pixl_land	Aufkommen auf dem Boden
	pixl_spawn	
Sterne	star_collect	Einsammeln eines Sterns
	star_sparkle	Beim Herunterfallen und nach dem Einsammeln des ersten gefundenen Sterns
	star_spawn	
	star_touch_ground	
Fähigkeiten	skill_collect	Einsammeln einer Skill-Box
	skill_spawn	
	skill_touch_ground	
Schlüssel	key_collect	

Kategorie	Name des Soundeffekts	ggf. Beschreibung
	key_spawn	
	key_touch_ground	
Plattformen	platform_spawn	
Türen	door_close	
	door_open	
	door_spawn	
	door_unlock	Beim Aufschließen mit Schlüssel
Schlösser	lock_shatter	Nach dem Aufschließen mit Schlüssel
Textboxen	text_show	Beim Erscheinen einzelner Buchstaben
Kacheln	tile_reveal	Bei der Veränderung des Typs einer Kachel
	draw_horizontal_outlines	(Einmalig) beim Zeichnen horizontaler Outlines
	draw_vertical_outlines	(Einmalig) beim Zeichnen vertikaler Outlines
User Interface	ui_container_show	Einblenden von UI-Elementen am oberen Rand
	ui_equals_full	Gleichheitszeichen ploppt nach dem Einsammeln von sechs Sternen auf
	ui_reward_open	Belohnungsbox wird geöffnet
	ui_star_set_full	Gesammelter Stern taucht im UI auf
Blendeffekte	aperture_close	Schließen der Blende beim Betreten einer Tür
	aperture_open	Öffnen der Blende nach dem Betreten einer Tür