

Level Editor für 2D Platformer

Tool- und Pluginprogrammierung

Benedikt Pischinger, MatrNr: 960025

Inhaltsverzeichnis

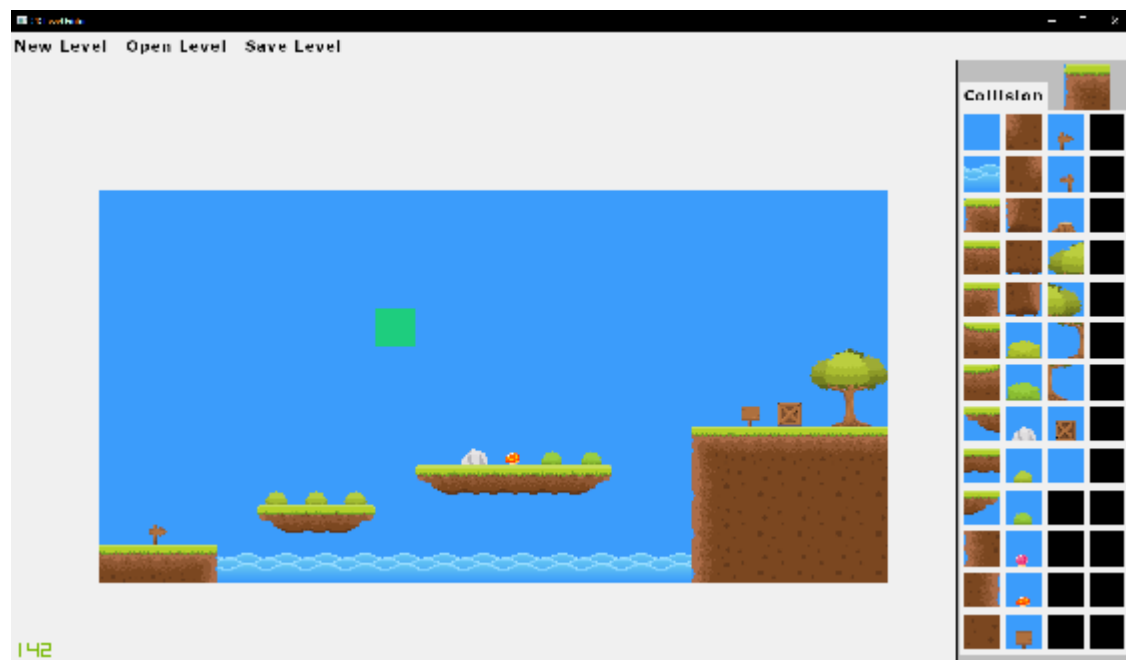
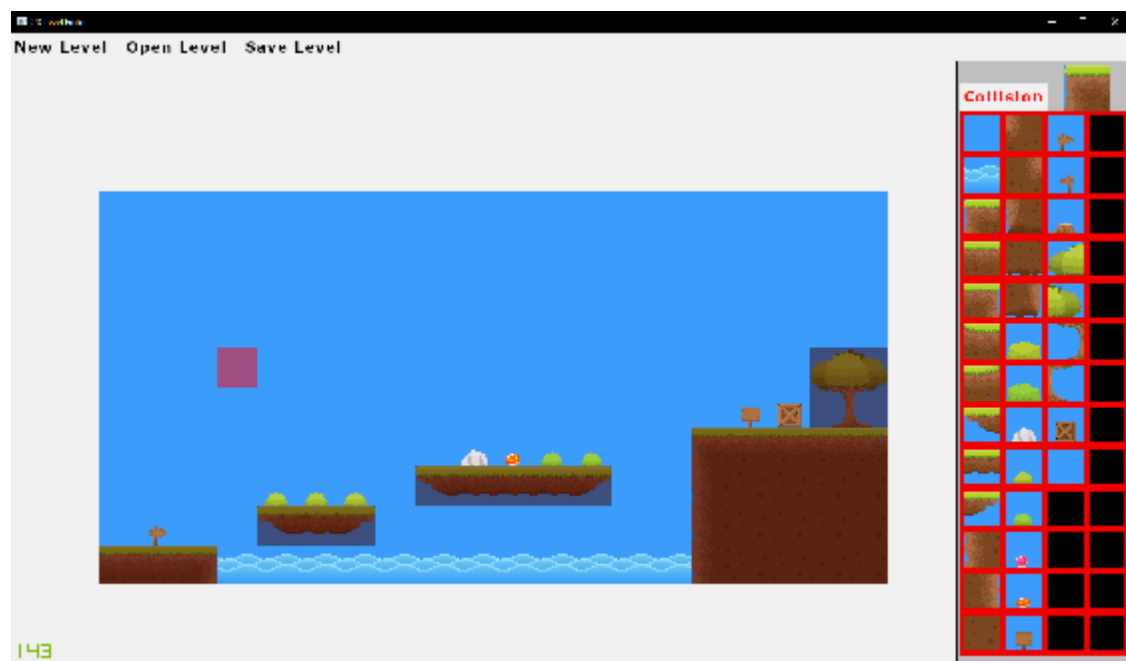
| | | |
|----------|--------------------------------|----------|
| 1 | Motivation | 1 |
| 2 | Benutzung | 3 |
| 3 | Technische Aspekte..... | 5 |

Motivation

Da ich schon immer ein Fan von 2D Plattformern war und in meiner Freizeit an dem ein oder anderen Prototyp gearbeitet habe, dachte ich mir es wäre eine gute Idee für diesen Kurs einen einfachen Level Editor für 2D Platformer zu entwickeln. Mit dem für diesen Kurs entstandenen Level Editor kann nun ein beliebiges Tilesset als Image File eingelesen werden. Die Tiles des Files können anschließend mit Hilfe eines minimalistischen Interfaces benutzt werden, um ein Level aufzubauen. Weiterhin können Tiles mit Collision Boxes markiert werden, um zu definieren bei welchen Tiles es sich um feste Gegenstände handelt und welche quasi Teil des Hintergrunds sind.

Das entstandene Level kann anschließend als ein *.lvl* File exportiert werden. Dieses File speichert das verwendete Tilesset, die platzierten Tiles und deren Positionen, so wie die Positionen der Collision Boxes. Das *.lvl* kann anschließend in einen 2D Platformer importiert werden.

Weiterhin können schon existierende Level in den Level Editor importiert werden, um Änderungen an den Leveln vorzunehmen und diese anschließend zu speichern.

Abb. 1.1. Level Editor im *Edit-Mode*Abb. 1.2. Level Editor im *Collision-Mode*


Benutzung

Nach dem Starten des Programms sieht man zunächst das Interface mit dem leerem Level und den leeren Tiles.

Mit den Buttons am oberen Rand des Programms kann entweder ein neues Level kreiert, oder ein existierendes Level geladen werden, sowie das entstandene Level gespeichert werden.

Da es sich um eine Console Application handelt, werden Inputs für das Programm über die Console entgegen genommen. Wählt man nun *New Level* aus, wird man dazu aufgefordert zunächst den Namen des gewünschten Tilesets in die Console einzugeben. Als nächsten folgen die Breite und die Höhe der einzelnen Tiles, sowie der Abstand der Tiles innerhalb des Image Files. Alle diese Eingaben werden in Pixeln gemessen. Anschließend wird die Breite und Höhe des Levels eingegeben. Diese beiden Angaben werden in Tiles gemessen und somit ist die tatsächliche Größe des Levels relativ zu den zuvor eingegeben Spezifikationen des Tilesets 2.1. Das mitgelieferte Tileset heißt *tileset1.png* (bei dieser Eingabe wird die File Extension gebraucht, da auch JPEGs und andere Image Files eingelesen werden können). Die Breite und Höhe sind jeweils 128 Pixel und der Abstand zwischen Tiles beträgt 5 Pixel.

Nachdem das Level erfolgreich kreiert wurde, können auf der rechten Seite des Interfaces Tiles ausgewählt werden. Die aktuelle Wahl wird dabei rechts oben angezeigt. Mit den WASD Tasten kann das Level verschoben und mit dem Mausrad herein und heraus gezoomt werden. Mit der linken Maustaste können Tiles platziert und mit der rechten Maustaste wieder gelöscht werden. Durch Gedrückhalten der Maustasten können Tiles innerhalb einer ganzen Fläche platziert oder gelöscht werden. In den Collision-Mode kann entweder mit Drücken der Leertaste, oder des *Collision* Buttons auf der rechten Seite über den Tiles, gewechselt werden. Das Platzieren der Collision Tiles funktioniert parallel zum Platzieren der normalen Tiles. Mit dem *Open Level* Button kann wiederum ein existierendes Level geladen werden. Im Projekt-Ordner sind drei Beispiellevel mit den Namen *example1*, *example2* und *example3* hinterlegt. Die Eingabe erfordert hier keine File Extension, diese wird automatisch angehängt. Beim Speichern des Levels wird nur ein Name gebraucht, welcher ebenfalls ohne File Extension in die Console eingegeben wird.



```
Select D:\User\Documents\GitHub\LevelEditor\Win32\Debug\LevelEditor.exe
New Level
Included tileset is named: "tileset1.png"
Name of the tileset file (including file extension): tileset1.png
Included tileset width: 128
Tile width in pixels: 128
Included tileset height: 128
Tile height in pixels: 128
Included tileset gap: 5
Gap between tiles in pixels: 5
Recommended level width: 10 - 20
Level width in tiles: 15
Recommended level height: 10 - 20
Level height in tiles: 10
Level created successfully.
```

Abb. 2.1. Console Input für den Level Editor

Technische Aspekte

Für die Umsetzung wurde eine Windows Console Application mit Hilfe von C++ und der Simple and Fast Multimedia Library (SFML) entwickelt.

- Die *Button* Klasse definiert die einzelnen Buttons zur Navigation des Programms.
- Die *Tile* Klasse dient zur Anzeige der einzelnen Tiles am rechten Bildschirmrand.
- Die *Interface* Klasse baut das Interface auf, welches aus dem *Interface View* und dem *Level View* besteht. Außerdem lädt diese Klasse die einzelnen Tiles aus dem Image File, basierend auf den Eingaben des Benutzers und übergibt diese an die *Tile* Klasse für die Darstellung der Tiles.
- Mit der *Level* Klasse wird das Level in der Mitte des Bildschirms angezeigt. Hier wird zunächst ein Level basierend auf der eingegebenen Höhe und Breite kreiert und mit der ersten Tile im Tilesset gefüllt. Weiterhin dient diese Klasse zum Lesen und Schreiben der importierten und exportierten Level.
- Als letztes gibt es noch die *LevelEditor* Klasse, welche alles zusammenbindet und das Programm initialisiert. Weiterhin ist diese Klasse auch für die Abarbeitung von Maus und Tastatur Inputs zuständig.

Relevante Methoden sind unter anderem die *LoadTilesset()* Methode in der *Interface* Klasse. Hier wird das eingelesene Image File in 52 individuelle Tiles, basierend auf Breite, Höhe und Tileabstand verarbeitet.

Mit der *ReadFromFile()* Methode in der *Level* Klasse wird ein Level File eingelesen. Die in dem File gespeicherten Informationen werden in Variablen gespeichert und das Level in den *Level View* geladen.

Die *WriteToFile()* Methode wiederum exportiert die Daten für ein Level und speichert diese in einem *.lvl* File, welches dann von einem passenden 2D Platformer geladen werden kann.