IGS Lilienthal

Informatik

JuicySoup Dokumentation

Leon Kommerau, Pascal Jung,
Hauke Schnau, Amro El-Seyed

Lehrer:
Herr Engelbertz

20. Dezember 2019

Inhaltsverzeichnis

Kapitel 1

Unser Spiel

1.1 Die Idee

Juicy Soup ist ein Spiel mit historischem Inhalt. Man spielt einen Zeitreisenden, der in der Vergangenheit gestrandet ist und versucht, wieder in die Gegenwart zu kommen. Dafür muss er Gegenstände aus der jeweiligen Zeit finden, um seine Zeitmaschine zu reparieren.

1.2 Anleitung

Nachdem man ein Level ausgewählt hat, kann man seinen Charakter mit den Tasten A und D nach links und rechts steuern. Mit der Leertaste springt man, mit Shift duckt man sich. Wenn man eine Waffe ausgewählt hat, kann man mit der Maus in Richtung des Mauszeigers schießen. Das Ziel ist auf der rechten Seite.

1.3 Warum serious?

Das Spiel ist serious, weil man beim Spielen auf spielerische Art etwas über die dargestellten Zeitalter lernt. Wir möchten den Spielern mit spaßigen Methoden mehr über die damaligen Helden und Bösewichte näher bringen.

Kapitel 2

Grundlagen der Algorithmik

2.1 Was ist ein Algorithmus?

Ein Algorithmus ist eine Folge von Anweisungen, die eine vorher bestimmte Aufgabe erledigen. Er muss bei denselben Eingabewerten immer die selben Ausgabewerte produzieren.

2.2 Was ist ein Programm?

Ein Programm ist eine Ansammlung von Algorithmen, die zusammenwirken. Der Nutzer/Anwender des Programms kann kann ihm beispielsweise über Kommandozeilenargumente oder eine Schnittstelle wie Maus oder Tastatur Eingabewerte übermitteln.

2.3 Was braucht man zum Programmieren?

Zum Programmieren braucht man einen Computer, auf dem zumindest ein Textbearbeitungsprogramm und ein Compiler oder Interpreter installiert sind. Programmiert man wie wir in JavaScript für den Browser, reicht dafür ein normaler Webbrowser aus. Für Java oder C++ bräuchte man allerdings einen Compiler, der den Code so übersetzt, dass er für den Computer verständlich ist.

2.4 Was ist p5.js?

p5.js ist eine Bibliothek, die auf JavaScript aufbaut und eigene Funktionen zum Zeichnen von geometrischen Formen und Bildern auf ein Canvas implementiert. Sie ist besonders an Künstler, Designer, Lehrer und Anfänger gerichtet.

2.5 Was sind Variablen und Datentypen?

In Variablen kann ein Programm Werte speichern, verändern und abrufen. Die Variablen haben dabei immer einen Datentyp wie z.B. `String`, `boolean`, `int`, `Array` und viele mehr. Bei den meisten Programmiersprachen muss man den Datentyp beim deklarieren der Variable mit angeben. Das ist bei JavaScript jedoch nicht nötig.

2.6 Was sind Kontrollstrukturen?

Kontrollstrukturen beeinflussen den Ablauf eines Programms.

Dazu gehören zum Beispiel Entscheidungen wie `if/else`-Abfragen, die einen Codeblock nur dann ausführen, wenn eine gegebene Bedingung zutrifft. Der Code im optionalen `else`-Block wird dann ausgeführt, wenn die Bedingung **nicht** zutrifft.

Schleifen führen einen Codeblock mehrfach aus. Es gibt sie in Form von `while`- und `for`-Schleifen. Beide führen den Code so lange aus, bis die im Kopf der Schleife gegebene Bedingung **nicht** mehr `true` ist, wobei die `for`-Schleife dies meist durch eine Zählervariable erreicht. Dazu gibt es noch die `for-each` Schleife, die über ein Array iteriert.

2.7 Was sind Funktionen und Objekte?

Funktionen werden genutzt, um Code zu strukturieren und wiederzuverwenden. Ein Ablauf, der oft gleich oder sehr ähnlich ausgeführt wird, kann in eine Funktion abgekapselt werden, um den Code übersichtlicher und leichter lesbar zu machen.

Objekte sind ähnlich wie Funktionen nützlich, um Struktur in den Code zu bringen. Sie werden genutzt, um ähnliche Variablen in einer zu bündeln

und bestehen aus Key/Value paaren. Ein Objekt, das ein Rechteck beschreibt, kann die Attribute `x`, `y`, `breite`, `höhe` und `farbe` haben.

Kapitel 3

Organisation der Gruppenarbeit

3.1 Verlauf der Projektarbeit

Nachdem wir gemeinsam unsere Idee festgelegt hatten, schrieben wir die ersten Zeilen. Der Prototyp, der dabei entstand ähnelte einem Super Mario Spiel. Nachdem erste Kollisionsabfragen und Physik implementiert waren, war es Zeit für einen Map-Editor, damit man die Level nicht von Hand in eine JSON-Datei schreiben muss. Dann wurden Grafiken für das Spiel rausgsucht, Level designt und Items, Gegner und Helfer einprogrammiert. Und natürlich wurden währenddessen ständig auch noch Bugs gefixt.

3.2 Aufgabenverteilung

- Hauke
  - Großteil des Codes geschrieben
  - Kapitel 2 ,,Grundlagen der Algorithmik" verfasst
- Pascal
  - Grafiken rasugesucht
  - Level Design
  - Fazit verfasst

- Leon
  - Musik rausgesucht
  - Teile des Codes geschrieben
  - Kaptitel 1 ,,Unser Spiel" verfasst
- Amro
  - Namensschöpfer
  - Kapitel 3 ,,Organisation der Gruppenarbeit" verfasst
  - mentale Unterstützung

Kapitel 4

Fazit

Wir sind mit unserem Ergebnis sehr zufrieden, auch wenn wir nicht alle Level fertigstellen konnten. Leider muss man zurzeit noch springen, um den Speer werfen zu können.

Unser größtes Problem war, dass wir uns so viel vorgenommen hatten, dass wir nicht alles in dem Zeitraum erledigen konnten. Ein weiteres Hindernis waren die Grafiken, da keiner von uns künstlerisch sehr begabt ist. Der Mangel an Grafiken lag nicht zuletzt daran, dass unsere Arbeitsteilung zumindest zu Beginn miserabel war. Sobald wir uns jedoch jeden Tag online getroffen haben, ging es voran und alle arbeiteten am Projekt. Wir nutzten das Live Share Plugin von Visual Studio Code, damit wir alle gleichzeitig Code schreiben, das Spiel spielen und Assets einfügen konnten. Die größte technische Hürde war die Kollisionserkennung. Sowohl die zwischen zwei achsenparallelen Rechtecken, besonders aber die zwischen rotierten Rechtecken machte uns zu Schaffen.

Wenn wir an dem Spiel weiterarbeiten könnten, würden wir alle weiteren Gegner, NPCs und Level in das Spiel einfügen.

Kapitel 5

Quellenverzeichnis

https://p5js.org https://www.codeproject.com/Articles/15573/2D-Polygon-Collision-D