**Planspiele – Zahlenkombinationsspiel**

**Funktionsweise**

Das Spiel besteht aus vier Zahnrädern, die mit der richtigen Zahlenkombination eine Tür öffnen. Auf zwei Schildern werden dem Spieler Hinweise gegeben, die zum Lösen des Rätsels nötig sind. Drei Zahlen kann der Spieler mithilfe dieser Hinweistexte selbst ermitteln. Die vierte Zahl bekommt der Spieler als Hinweis nach dem Lösen des vorangegangenen Spiels gegeben. Der Spieler ändert die Zahl auf dem Zahnrad, indem er auf dieses klickt. Jedes Zahnrad zeigt die Zahlen 1 bis 9 und sie drehen sich nur in eine Richtung. Wenn der Spieler die richtige Zahlenkombination eingibt, öffnet sich die Tür und er kann zum nächsten Abschnitt des Spiels gelangen.

**Hinweistext auf den beiden Schildern**

Schild 1: Für die ersten beiden Zahnräder, rechne dir die Anzahl an Möglichkeiten aus, die du bei drei Zahnrädern hast. Rad 1 entspricht der ersten Stelle deines Ergebnisses, das Rad 2 entspricht der letzten Stelle des Ergebnisses. Hinweis: Wie viele Möglichkeiten hat man bei einem Zahnrad? Wie erhält man dann die Anzahl an Möglichkeiten bei drei Rädern?

Lösung: 9 \* 9 \* 9 = 729 => Rad 1: 7, Rad 2: 9

Schild 2: Die Kombination von Rad 3 entspricht der Wahrscheinlichkeit hier die richtige Zahl zu erhalten. Multipliziere diese Wahrscheinlichkeit mit 18, um die richtige Kombination zu erhalten. Hinweis: Wie viele Zahlen des Rads sind die richtige Lösung?

Lösung: 1/9 \* 18 = 2

**Implementierung**

Jedes Rad besitzt eine eigene Rad-Logik, welche das Rad rotiert, die Zahl erhöht, wenn es gedreht wurde und die aktuelle Zahl an die übergeordnete Logik weitergibt. Die Zahlen sind dabei nicht an dem Objekt orientiert, sondern nach der Anzahl der Rotationen. Wenn ein Rad gedreht wird, ist klar, dass auch die Zahl sich erhöht hat. Wenn man die 9 erreicht hat und bei einem weiteren Drehen wieder die 1 erscheint, wird die Zahl zurückgesetzt. Jedes Rad ist dabei eine Repräsentation der aktuellen Zahl in einem Array. Die übergeordnete Logik überprüft die Kombination mit den Zahlen jedes Rads, ob diese mit der Lösung des Rätsels übereinstimmen. Bei der richtigen Kombination öffnet sie die Tür und setzt dafür auch die nötigen Töne. Über Singletons werden die einzelnen Werte der Räder und auch jeder weitere Wert oder Boolean gespeichert, die für eine korrekte Funktionsweise nötig sind. So kann auch von außerhalb jede Funktion auf die Werte zugreifen.

Bei der Vorgehensweise bin ich zuerst von einem einzelnen Rad ausgegangen und habe mir überlegt, was ich tun muss damit sich das Rad dreht und wie ich die Zahlen dabei setzen kann. Denn ich wollte das Spiel so entwickeln, dass man beliebig viele Zahnräder hinzufügen oder entfernen kann. So konnten wir die Zahnräder auch wieder beim letzten Rätsel wiederverwenden und mussten nur ein wenig die Logik verändern, damit es für die neuen Anforderungen funktioniert hat. Danach habe ich die Logik um die Zahnräder herum implementiert, da ich wusste, wie viele Räder ich habe und wie ich auf die Zahlen zugreifen kann. Eine einfach if-Abfrage konnte dann überprüfen, ob der Spieler die richtige Zahl ausgewählt hat.

**Probleme**

Die einzigen Probleme, die meine Implementierung verursacht hat war, dass ich die Rotationen zuerst mit EulerAngles umgesetzt habe. Da diese aber nicht mit einer einfachen Abfrage überprüft werden können, da sie eine andere Rotation verursachen, als eine einfache Drehung des Objekts anhand von vorgegebenen Gradzahlen, drehte sich das Rad entweder zu weit oder zu wenig und hat außerdem unterschiedliche Drehgeschwindigkeiten erhalten. Als ich dann das gesamte Object mit der Quaternion Drehung gedreht habe, konnte ich abfragen, um wie viel Grad sich das Rad gedreht hat. Also konnte ich die Räder an exakte Positionen drehen und auch die Zahlen updaten. Danach gab es keine weiteren nennenswerten Probleme.