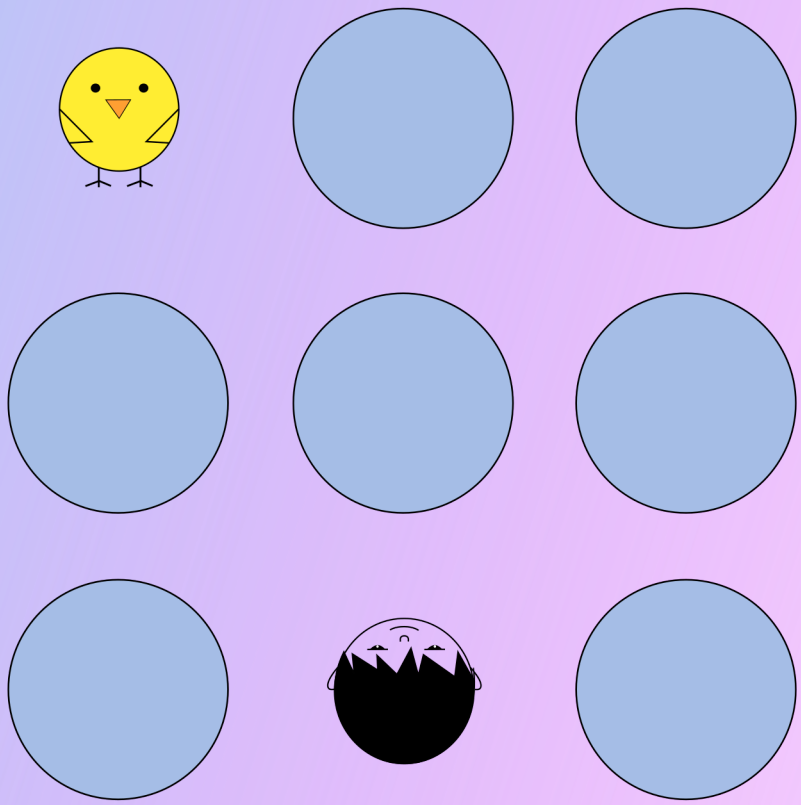
Technische Dokumentation

Christian Schröder MatrNr.: 2323133

Malte Saupe MatrNr.: 2315546

13. Januar 2019



1 Einleitung

Unser Projekt Piep ist eine spielerische Anwendung, mit der man sein räumliches Hörvermögen testen kann.

Ein Küken versteckt sich unter einem der neun Felder und piept. Dieses Piepen wird abhängig von der Position des Spielers und der Position des Kükens akustisch simuliert. Der Spieler hat mehrere Versuche, um das Küken zu finden. Durchläuft er alle Runden, werden die erreichten Prozentpunkte ausgeben und das Spiel kann wieder von vorne gestartet werden.

Die Anwendung wurde in HTML, CSS und JavaScript geschrieben. Für das Berechnen des 3D Audio benutzen wir die Bibliothek Resonance Audio. Die verwendeten Geräusche wurden mit Audacity erstellt. Die Grafiken für den Hintergrund, die Schaltknöpfe, das Küken und den Spieler wurden in Inkscape gezeichnet. Die Spiellogik ist mit dem Entwurfsmuster State umgesetzt. Für die Übersichtlichkeit ist jede Klasse in einer eigenen Datei und die einzelnen zusammenhängende Teile des Codes in separaten Dateien gespeichert:

- main.js - Aufruf des Spiels und responsive Darstellung der Webseite
- game.js - Spiellogik und Darstellung der Spielfiguren
- 3dAudio.js - Berechnung und Ausgabe des 3D Sounds

2 Spielfeld

Das Spielfeld besteht aus neun Feldern. Diese neun DIV-Container sind im GridLayout angeordnet und werden zu Beginn mit der Methode createGameField() erzeugt und mit eine Positions ID und dem Klassennamen fieldButton versehen. Die Koordinaten der Felder sind in der DIV-Container ID gespeichert. Die x-Position steht an erster Stelle gefolgt von der y-Position. Der untransformierte Koordiatenursprung ist das linke obere Feld.

Bsp.: ID = 12 bedeutet x = 1 und y = 2.

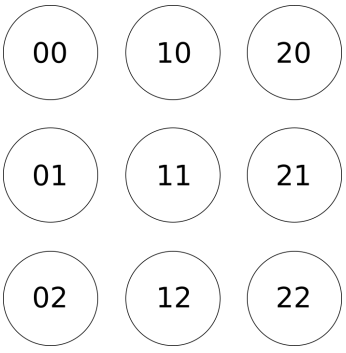


Abbildung 1: Spielfeld

3 Spiellogik

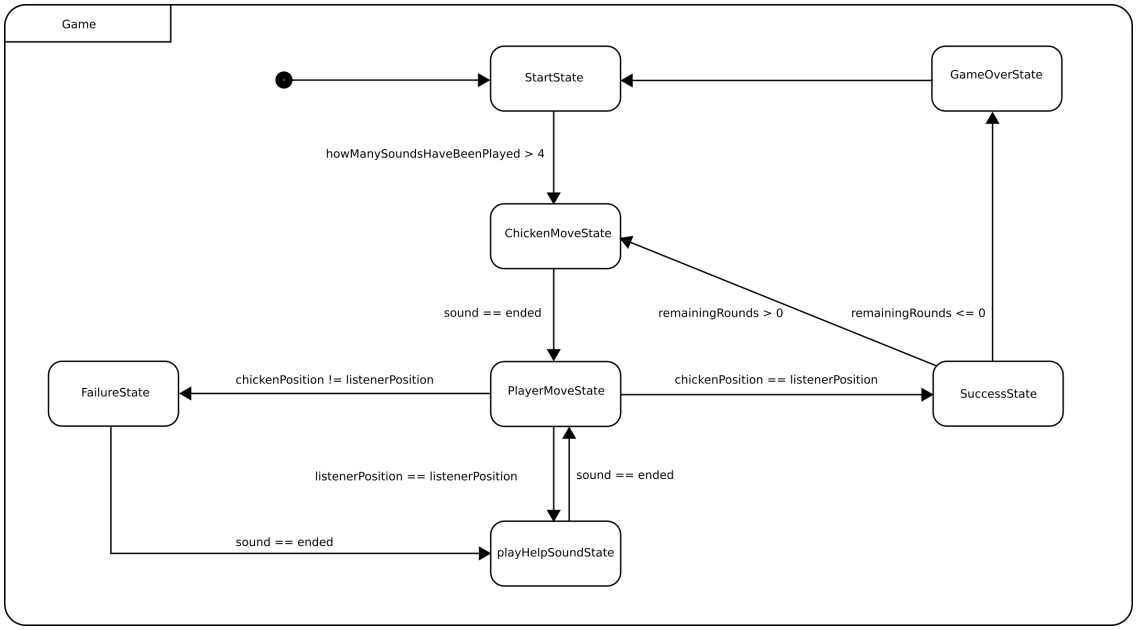


Abbildung 2: Zustandsautomat

Die Spiellogik wurde mit dem Zustands Entwurfsmuster umgesetzt. Jeder Zustand ist in einer eigenen Klasse gekapselt, in der das Verhalten für den jeweiligen Zustand definiert ist. Die Klasse Game und die Zustandsklassen implementieren die Zustands Methoden run(), startPressed(), fieldPressed(), soundPlayingStopped() und nextState().

3.1 Game

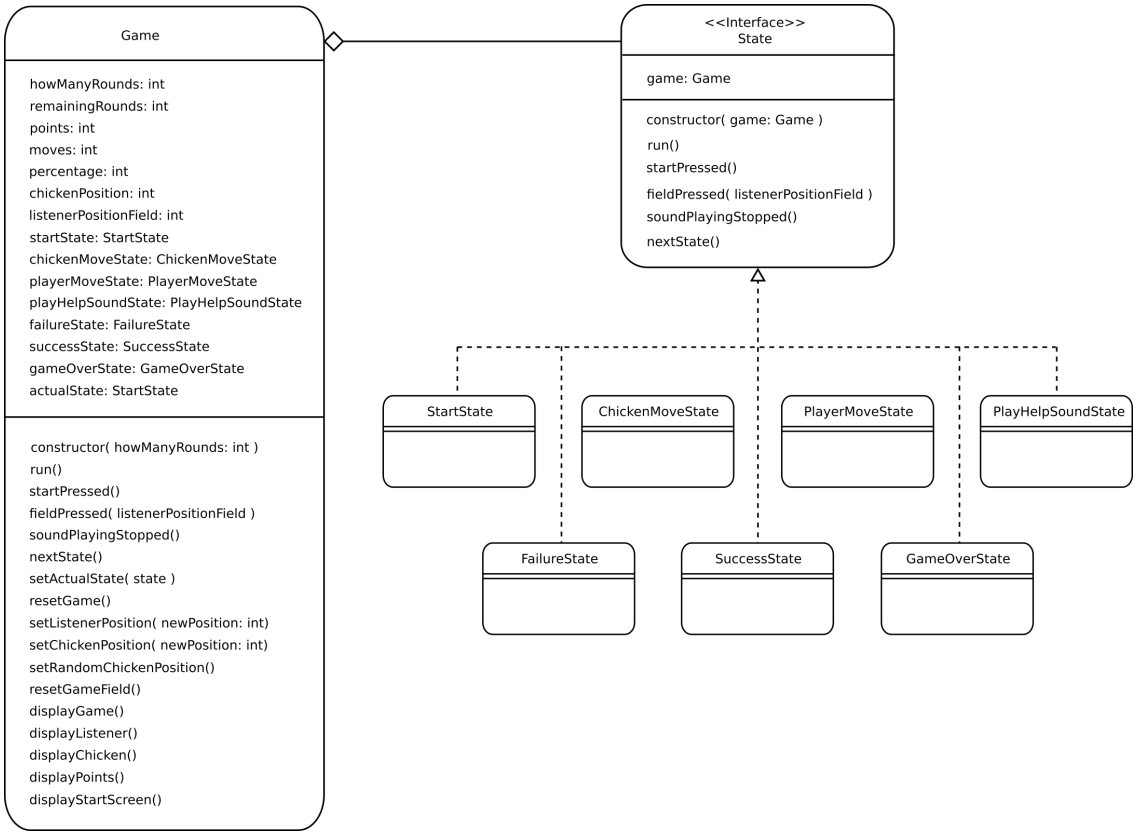


Abbildung 3: Klassendiagramm

Game hat von jedem Zustand ein Objekt und über die Instanzvariable actualState eine Referenz auf das aktuelle Zustandsobjekt. Von außen betrachtet werden die Methoden von Game aufgerufen und Game übergibt den Aufruf an den jeweiligen aktuellen Zustand.

Die externe Steuerung geschieht durch mehrere ActionListener. Die Methode startPressed() wird durch einen ActionListener auf den DIV-Container mit der ID overlayStartButton ausgelöst, die Methode fieldPressed() durch ActionListener auf den DIV-Containern der Klasse fieldButton. Die Methode soundPlayingStopped() wird durch einen ActionListener auf das Ende des Abspielens der Sound Dateien getriggert.

Die Methoden run() und nextState() werden nur intern in der Klasse Game verwendet. Der Aufruf der Methode run() dient dazu, Aktionen in einem Status auszuführen, die nicht von außen gesteuert werden wie z.B. das Anzeigen der Spielfiguren und das Abspielen von Geräuschen. Die Methode nextState() definiert den folgenden aktiven Zustand.

Die Spiellogik ist so umgesetzt das nur der StartState auf das Klicken des Startbuttons reagiert und nur der Zustand PlayerMove auf das Klicken eines Feldes reagiert.

- resetGame() - setzt das Spiel auf den Ausgangszustand zurück
- setRandomChickenPosition() - erstellt eine zufällige Position für das Küken auf dem Spielfeld, die nicht der aktuellen Position und der des Spielers entspricht
- resetGameField() - setzt für alle DIV-Container der Klasse fieldButton die Standardgrafik

3.2 StartState

Reagiert auf das Anklicken des Startbuttons. Das Spiel wird zurückgesetzt, anschließend wird fünfmal das Küken an zufälligen Positionen angezeigt und der entsprechende 3D Sound abgespielt.

3.3 ChickenMoveState

Nur der Spieler wird angezeigt. Das Küken wird an eine zufällige Position gesetzt und der entsprechende 3D Sound abgespielt

3.4 PlayerMoveState

Reagiert auf das Anklicken eines der Felder. Hierbei wird die ID des Feldes übergeben. Stimmt diese mit der ID des Spielers überein, so wird der PlayHelpSoundState aufgerufen, wenn nicht wird überprüft ob sie mit der ID des Kükens übereinstimmt. Ist das der Fall, wird der SuccessState aufgerufen ansonsten der FailureState.

3.5 PlayHelpSoundState

Das Piepen wird abgespielt und der PlayerMoveState aufgerufen

3.6 FailureState

Der Klang für den Fehler wird abgespielt und PlayHelpSoundState aufgerufen.

3.7 SuccessState

Die erreichten Punkte für diesen Zug werden berechnet, die Züge zurückgesetzt und der Klang für den Erfolg wird abgespielt. Ist dieser beendet so wird das Küken aufgedeckt und überprüft ob es noch verbleibende Spielrunden gibt. Wenn ja, wird in den ChickenMoveState gewechselt ansonsten in den GameOverState.

3.8 GameOverState

Die erreichten Prozentpunkte werden berechnet und der Startscreen angezeigt. Dann wird in den StartState gewechselt.

4 3D Audio

Die drei benötigten Klänge für das Spiel wurden mit Audacity erzeugt. Das Piepen des Kükens mit dem Generator Zirpen und die Klänge für Erfolg und Fehler mit dem Generator Klang. Dabei haben wir darauf geachtet, dass die Geräusche sanft ein- und ausgeblendet werden.

Das räumliche Soundfeld wird mit der Bibliothek Resonance Audio berechnet. Hierfür ist es im einfachsten Fall nur notwendig die Positionen von Quelle und Hörer zu übergeben. Es gibt die Möglichkeit ein Raummodell anzugeben und in diesem für die unterschiedlichen Raumseiten Materialien festzulegen.

Die ersten Versuche haben wir mit einem Raummodell unternommen. Um eine natürlichen Klang zu erhalten haben wir das Material des Bodens auf Gras und das der Wände auf transparent gestellt. Subjektiv ergab das für uns kein natürliches Klangfeld. Gerade im Bereich der Wände kam es weiterhin zu Reflexionen, die die Ortung des Kükens erschwerten. Nach mehreren

Versuchen mit weiteren Materialien haben wir uns gegen die Verwendung eines Raummodelles entschieden. Das führte zu einer natürlicheren Klangwiedergabe.

Wir haben drei verschiedene Klänge, die wir absspielen. Die Klänge für Fehler und Erfolg sind dabei ortsunabhängig, das Piepen des Kükens ist ortsabhängig. Das Abspielen dieser zwei Varianten wird mit Methode play3DSound() ermöglicht. In ihr wird die Art des Klangs, die Position der Quelle und die Position des Hörers übergeben. Bei ortsunabhängigen Soundquellen sind die Position der Quelle und des Hörers identisch.

Mit den Methoden setSourceAndListenerPosition() und transformCoordinatesToAudioField() wird die ID aufgesplitted und die Koordinaten transfomiert, so dass der Abstand zwischen den Feldern dem definierten Audioabstand entspricht und der Koordinatenursprung in der Mitte liegt. Damit enspricht das Layout der Vorgabe von Resonance Audio: die X-Achse zeigt nach rechts, die Y-Achse nach oben und die Z-Achse auf den Hörer.

5 Spieloptimierung

Beim Usertest der fertigen Anwendung stellte sich heraus, dass ein Spielfeld bestehend aus einem Raster von fünf mal fünf Feldern, für den Großteil der Anwender zu schwierig íst. Eine Änderung des räumlichen Abstands der Felder konnte daran nichts ändern. Daher haben wir uns entschlossen das endgültige Spielfeld als drei mal drei Raster auszulegen.

Die Ortung der Geräusche ist in unserem Fall hauptsächlich von der Orientierung der Quelle und des Hörers zueinander und des Abstands zwischen den Feldern abhängig. Für die Ausrichtung der Quelle gibt es drei sinvolle Möglichkeiten:

- die Quelle strahlt in Richtung Hörer - positive Z-Achse
- die Quelle strahlt in die entgegengesetzte Richtung zum Hörer - negative Z-Achse
- die Quelle strahlt nach oben - positive Y-Achse

Für die drei Varianten gab es keinen hörbaren Unterschied. Daher haben wir eine Testmatrix erstellt und mit dieser gleichzeitig versucht, den optimalen Abstand herauszufinden. Jede Variante haben wir mehrmals getestet und den Durchschnitt ermittelt.

Richtung Quelle, Abstand	0,5 [m]	1 [m]	2 [m]
positive Z-Achse	83 %	91 %	79 %
negative Z-Achse	75 %	84 %	72 %
positive Y-Achse	67 %	73 %	63 %

Auf Grundlage des Tests haben wir uns für die Variante Quelle strahlt in die positve Z-Achse mit einem Feldabstand von einem Meter entschieden.