Developer Guide

This guide provides information for developers who want to contribute to the project.

Quickstart

Clone the repository

```
git clone git@github.com:Konnektoren/konnektoren-web-game.git
```

then compile and serve the page with

```
trunk serve
```

then visit http://localhost:8080

Repository Structure

The repository is structured as follows:

- `docs/` contains the developer guide
- `src/` contains the source code
- `src/assets/challenges/` contains the challenge assets
- `src/assets/i18n/` contains the internationalization assets
- `styles/` contains the scss styles
- `Cargo.toml` contains the package metadata
- `README.adoc` contains the project documentation

Contributing

To contribute to the project, follow these steps:

1. Fork the repository
2. Create a new branch
3. Make your changes
4. Push your changes to your fork
5. Open a pull request

Level Design

The levels are defined in the `src/assets/challenges/levels.yml` file. Each level has a unique identifier and contains a list of challenges. The challenges are defined in separate YAML files in the `src/assets/challenges/` directory.

```yaml
id: "level-a1"
name: "Level A1"
challenges:
  - id: "articles-1"
    name: "Artikel 1"
    description: "Choose the correct article for the following nouns"
    challenge: "articles-info"
    variant: "informative-markdown"
    tasks: 1
    unlock_points: 0
    position: [2, 2]
  - id: "articles-2"
    name: "Artikel 2"
    description: "Choose the correct article for the following nouns"
    challenge: "articles"
    variant: "multiple-choice-circle"
    tasks: 12
    unlock_points: 10
    position: [4, 3]
  - id: "articles-3"
```

This structure allows for a modular and flexible level design. The main levels file references individual challenge files, making it easy to add, remove, or modify challenges without affecting the overall level structure.

Tasks

Each challenge consists of a task that the user must complete. The task can be a text-based question, a multiple-choice question, or a custom challenge. The challenge configuration defines the type, content, and parameters of the task.

```yaml
challenges:
  - id: "articles-1"
    name: "Artikel 1"
    description: "Choose the correct article for the following nouns"
    challenge: "articles-info"
    variant: "informative-markdown"
    tasks: 1
    unlock_points: 0
    position: [2, 2]
```

Key components of a challenge configuration:

1. `id`: A unique identifier for the challenge.
2. `name`: The display name of the challenge.
3. `description`: A brief explanation of what the user needs to do.
4. `challenge`: Specifies the challenge type or references the challenge data file.
5. `variant`: Defines a specific variant or presentation style for the challenge.
6. `tasks`: Indicates the number of tasks or questions in this challenge.
7. `unlock_points`: The number of points required to unlock this challenge.
8. `position`: The challenge's position in the game's visual layout.

The `tasks` field can use the TaskPattern format, allowing for flexible task selection: - Exact number: `tasks: 10` - Range: `tasks: "5..15"` - Random selection: `tasks: "10:random"` or `tasks: "10:1..20"`

This configuration system enables the creation of diverse and engaging challenges while maintaining a consistent structure for easy management and expansion of the game content.

Create a New Challenge Dataset

To create a new challenge dataset, follow these steps:

- Create a new file in `src/assets/challenges/` for the challenge data like `src/assets/challenges/articles.yml`

```yaml
!multiple-choice
id: "articles"
name: "Artikel"
options:
  - id: 0
    name: "der"
  - id: 1
    name: "die"
  - id: 2
    name: "das"

questions:
  - question: "Haus"
    help: "Ich habe ein neues Haus gekauft."
    option: 2
    image: "fa-regular fa-house-chimney-window"
  - question: "Hund"
    help: "Mein Hund ist sehr lieb."
    option: 0
    image: "fa-regular fa-dog"
```

- Add the challenge data to the level file like `src/assets/challenges/levels.yml`

```
id: "level-a1"
name: "Level A1"
challenges:
  - id: "articles-1"
    name: "Artikel 1"
    description: "Choose the correct article for the following nouns"
    challenge: "articles-info"
    variant: "informative-markdown"
    tasks: 1
    unlock_points: 0
    position: [2, 2]
```

- Add the challenge file `src/assets/challenges/articles.yml` to the level file `src/assets/challenges/level_a1.yml` to the level loader `src/model/level_loader.rs`

```
}

impl LevelLoader<ChallengeFactory> for ChallengeFactory {
    fn level_a1() -> Result<ChallengeFactory, LoaderError> {
                personalpronouns_info,
                connectives,
                connectives_info,
            ],
        })
    }

    fn level_a2() -> Result<ChallengeFactory, LoaderError> {
        let articles: ChallengeType =
            serde_yaml::from_str(include_str!("../assets/challenges/articles.yml"))?;
        let articles_info: ChallengeType =

serde_yaml::from_str(include_str!("../assets/challenges/articles_info.yml"))?;
        let negation: ChallengeType =
```

Create new Custom Challenge Design

To create a new custom challenge design, follow these steps:

- Create a new html file in `src/assets/challenges/html` for the challenge design like `src/assets/challenges/custom_articles.html`

```
<!doctype html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>German Articles</title>
```

```
        <link rel="stylesheet" href="../css/custom_articles.css" />
    </head>
    <body>
        <div class="custom-articles-challenge">
            <h2 class="custom-articles-challenge__title">
                German Article Exercise
            </h2>

            <div class="custom-articles-challenge__question-container">
                <p class="custom-articles-challenge__question-text">
                    Question will appear here
                </p>
                <p class="custom-articles-challenge__help-text">
                    Hint will appear here
                </p>
                <i class="custom-articles-challenge__question-image"></i>
            </div>

            <ul class="custom-articles-challenge__options-list">
                <!-- Options will be dynamically added here -->
            </ul>

            <div class="custom-articles-challenge__feedback"></div>

            <div
                class="custom-articles-challenge__completion"
                style="display: none"
            >
                Challenge Finished! Great Job!
            </div>

            <button
                class="custom-articles-challenge__button custom-articles-
challenge__button--finish"
            >
                Finish Exercise
            </button>
        </div>
    </body>
</html>
```

- Create a new css file in `src/assets/challenges/css` for the challenge design like `src/assets/challenges/css/custom_articles.css`

```
.custom-articles-challenge {
    text-align: center;
    padding: 1.25rem;
    font-family: "Arial", sans-serif;
    background-color: #f9f9f9;
    border-radius: 0.625rem;
```

```
    width: 90%;
    max-width: 37.5rem;
    margin: auto;
    box-shadow: 0 0.25rem 0.75rem rgba(0, 0, 0, 0.15);
    transition: transform 0.3s ease;
}

.custom-articles-challenge:hover {
    transform: translateY(-0.625rem);
    box-shadow: 0 0.375rem 1.25rem rgba(0, 0, 0, 0.2);
}

.custom-articles-challenge__title {
    font-size: 1.75rem;
    margin-bottom: 1.25rem;
    letter-spacing: 0.125rem;
    color: #444;
}

.custom-articles-challenge__question-container {
    margin-bottom: 1.25rem;
    padding: 0.625rem;
    background-color: #f1f1f1;
    border-radius: 0.5rem;
    box-shadow: 0 0.125rem 0.3125rem rgba(0, 0, 0, 0.1);
    transition: background-color 0.3s ease;
}

.custom-articles-challenge__question-container:hover {
    background-color: #e9e9e9;
}

.custom-articles-challenge__question-text {
    font-size: 1.625rem;
    font-weight: bold;
    color: #333;
    margin-bottom: 0.625rem;
    animation: custom-articles-challenge__fadeIn 1s ease-in;
}

.custom-articles-challenge__help-text {
    font-size: 1rem;
    color: #888;
    margin-bottom: 0.625rem;
    font-style: italic;
    animation: custom-articles-challenge__fadeIn 1.2s ease-in;
}

.custom-articles-challenge__question-image {
    font-size: 3.75rem;
    margin: 0.9375rem 0;
```

```
    color: #555;
    animation: custom-articles-challenge__popIn 1s ease-in-out;
}

.custom-articles-challenge__options-list {
    list-style-type: none;
    padding: 0;
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    gap: 0.625rem;
}

.custom-articles-challenge__option {
    padding: 0.75rem 1.5rem;
    background-color: #f0f0f0;
    cursor: pointer;
    border-radius: 0.625rem;
    font-size: 1.125rem;
    transition: all 0.3s ease;
    box-shadow: 0 0.125rem 0.3125rem rgba(0, 0, 0, 0.1);
    transform: scale(1);
}

.custom-articles-challenge__option:hover {
    background-color: #d0d0d0;
    transform: scale(1.1);
    box-shadow: 0 0.25rem 0.625rem rgba(0, 0, 0, 0.2);
}

.custom-articles-challenge__button {
    padding: 0.75rem 1.875rem;
    font-size: 1rem;
    margin-top: 1.25rem;
    cursor: pointer;
    border: none;
    border-radius: 0.3125rem;
    background-color: #007bff;
    color: white;
    transition:
        background-color 0.3s ease,
        transform 0.3s ease;
    box-shadow: 0 0.1875rem 0.5rem rgba(0, 0, 0, 0.15);
}

.custom-articles-challenge__button:hover {
    background-color: #0056b3;
    transform: translateY(-0.3125rem);
}

.custom-articles-challenge__button--finish {
```

```
    background-color: #28a745;
}

.custom-articles-challenge__button--finish:hover {
    background-color: #218838;
}

.custom-articles-challenge__feedback {
    margin-top: 1.25rem;
    font-size: 1.25rem;
    font-weight: bold;
    padding: 0.5rem 1rem;
    border-radius: 0.3125rem;
    display: none; /* Change from 'inline-block' to 'none' */
    opacity: 1; /* Change from 0 to 1 */
}

.custom-articles-challenge__feedback--correct {
    color: white;
    background-color: #28a745;
}

.custom-articles-challenge__feedback--incorrect {
    color: white;
    background-color: #dc3545;
}

.custom-articles-challenge__completion {
    margin-top: 1.875rem;
    font-size: 1.375rem;
    font-weight: bold;
    color: green;
    opacity: 0;
    animation: custom-articles-challenge__fadeIn 1.5s forwards;
}

/* Results page specific styles */
.custom-articles-challenge__performance-wrapper {
    font-size: 1.2rem;
    margin-bottom: 1rem;
}

.custom-articles-challenge__performance {
    font-weight: bold;
    color: #007bff;
}

.custom-articles-challenge__results-container {
    background-color: #f8f9fa;
    border-radius: 0.5rem;
    padding: 1rem;
```

```
    margin-top: 1rem;
    text-align: left;
}

.custom-articles-challenge__results-list {
    list-style-type: none;
    padding: 0;
}

.custom-articles-challenge__result-item {
    margin-bottom: 1rem;
    padding-bottom: 1rem;
    border-bottom: 1px solid #dee2e6;
}

.custom-articles-challenge__result-item:last-child {
    border-bottom: none;
}

.custom-articles-challenge__result-question,
.custom-articles-challenge__result-answer,
.custom-articles-challenge__result-correct,
.custom-articles-challenge__result-help {
    margin: 0.5rem 0;
}

.custom-articles-challenge__result-question {
    font-weight: bold;
}

.custom-articles-challenge__result-answer,
.custom-articles-challenge__result-correct {
    padding-left: 1rem;
}

.custom-articles-challenge__result-help {
    font-style: italic;
    color: #666;
}

.custom-articles-challenge__overview {
    background-color: #f0f0f0;
    border-radius: 0.5rem;
    padding: 1rem;
    margin-bottom: 1rem;
}

.custom-articles-challenge__overview-title {
    font-size: 1.25rem;
    margin-bottom: 0.5rem;
}
```

```
.custom-articles-challenge__overview-item {
    font-size: 1rem;
    margin: 0.25rem 0;
}

.custom-articles-challenge__overview-item--correct {
    color: #28a745;
}

.custom-articles-challenge__overview-item--incorrect {
    color: #dc3545;
}

.custom-articles-challenge__results-list {
    margin-top: 1rem;
}

/* Animations */
@keyframes custom-articles-challenge__fadeIn {
    0% {
        opacity: 0;
    }
    100% {
        opacity: 1;
    }
}

@keyframes custom-articles-challenge__popIn {
    0% {
        transform: scale(0);
        opacity: 0;
    }
    100% {
        transform: scale(1);
        opacity: 1;
    }
}

@keyframes custom-articles-challenge__slideIn {
    0% {
        transform: translateY(1.875rem);
        opacity: 0;
    }
    100% {
        transform: translateY(0);
        opacity: 1;
    }
}

@keyframes custom-articles-challenge__bounceIn {
```

```
    0%,
    100% {
        transform: translateY(0);
    }
    50% {
        transform: translateY(-0.625rem);
    }
}

@keyframes custom-articles-challenge__shake {
    0%,
    100% {
        transform: translateX(0);
    }
    25% {
        transform: translateX(-0.3125rem);
    }
    75% {
        transform: translateX(0.3125rem);
    }
}

/* Responsive adjustments */
@media (max-width: 48rem) {
    .custom-articles-challenge {
        width: 95%;
        padding: 1rem;
    }

    .custom-articles-challenge__title {
        font-size: 1.5rem;
    }

    .custom-articles-challenge__question-text {
        font-size: 1.375rem;
    }

    .custom-articles-challenge__help-text {
        font-size: 0.875rem;
    }

    .custom-articles-challenge__question-image {
        font-size: 3rem;
    }

    .custom-articles-challenge__option {
        padding: 0.625rem 1.25rem;
        font-size: 1rem;
    }

    .custom-articles-challenge__button {
```

```
        padding: 0.625rem 1.5625rem;
        font-size: 0.875rem;
    }

    .custom-articles-challenge__feedback {
        font-size: 1.125rem;
    }

    .custom-articles-challenge__completion {
        font-size: 1.25rem;
    }

    .custom-articles-challenge__performance-wrapper {
        font-size: 1rem;
    }

    .custom-articles-challenge__results-container {
        padding: 0.75rem;
    }

    .custom-articles-challenge__result-item {
        margin-bottom: 0.75rem;
        padding-bottom: 0.75rem;
    }
}
```

- Create a new js file in `src/assets/challenges/js` for the challenge design like `src/assets/challenges/js/custom_articles.js`

```
console.log(window.konnektoren);

const data = window.konnektoren
  ? window.konnektoren.challenge.data
  : new Map([
      [
        "options",
        [
          new Map([
            ["id", 0],
            ["name", "der"],
          ]),
          new Map([
            ["id", 1],
            ["name", "die"],
          ]),
          new Map([
            ["id", 2],
            ["name", "das"],
          ]),
        ],
```

```
      ],
      [
        "questions",
        [
          new Map([
            ["question", "What is the article for 'Haus'?"],
            ["option", 2], // 'das' is correct
            ["help", "The article for 'Haus' is 'das'."],
          ]),
          new Map([
            ["question", "What is the article for 'Tisch'?"],
            ["option", 0], // 'der' is correct
            ["help", "The article for 'Tisch' is 'der'."],
          ]),
          new Map([
            ["question", "What is the article for 'Buch'?"],
            ["option", 2], // 'das' is correct
            ["help", "The article for 'Buch' is 'das'."],
          ]),
        ],
      ],
    ]);

console.log(data);

let currentQuestionIndex = 0;
let correctAnswers = 0;
let userAnswers = [];

// Translate static text on page load
function translateStaticText() {
  const title = document.querySelector(".custom-articles-challenge__title");
  const finishButton = document.querySelector(
    ".custom-articles-challenge__button--finish",
  );

  if (title) {
    title.textContent = window.konnektoren.tr("German Article Exercise");
  }

  if (finishButton) {
    finishButton.textContent = window.konnektoren.tr("Finish Exercise");
  }
}

function finishChallenge() {
  if (window.konnektoren && window.konnektoren.sendEvent) {
    const event = {
      type: "Finish",
      result: {
        id: window.konnektoren.challenge.id,
```

```
        performance: calculatePerformance(),
        data: {
          answers: userAnswers,
        },
      },
    };
    window.konnektoren.sendEvent(event);
  } else {
    // For testing purposes, display the results
    console.log(
      "Quiz Finished! Your performance: " + calculatePerformance() * 100 + "%",
    );
    console.log("User Answers:", userAnswers);
    alert(
      "Quiz Finished! Your performance: " + calculatePerformance() * 100 + "%",
    );
  }
}

// Load the current question
function loadQuestion() {
  const currentQuestion = data.get("questions")[currentQuestionIndex];
  const questionText = document.querySelector(
    ".custom-articles-challenge__question-text",
  );

  if (questionText === null) {
    return;
  }

  const helpText = document.querySelector(
    ".custom-articles-challenge__help-text",
  );
  const questionImage = document.querySelector(
    ".custom-articles-challenge__question-image",
  );
  const optionsList = document.querySelector(
    ".custom-articles-challenge__options-list",
  );
  const feedback = document.querySelector(
    ".custom-articles-challenge__feedback",
  );

  // Reset feedback on new question
  feedback.textContent = "";
  feedback.style.display = "none";

  // Update question and help text
  questionText.textContent = currentQuestion.get("question");
  helpText.textContent = currentQuestion.get("help");
```

```
  // Update question image if available
  if (currentQuestion.get("image")) {
    questionImage.className =
      "custom-articles-challenge__question-image fas " +
      currentQuestion.get("image");
  } else {
    questionImage.className = "custom-articles-challenge__question-image";
  }

  // Clear previous options
  optionsList.innerHTML = "";

  // Create and display options
  data.get("options").forEach((option) => {
    const li = document.createElement("li");
    li.className = "custom-articles-challenge__option";
    li.textContent = option.get("name");
    li.dataset.id = option.get("id");

    // Handle option click
    li.addEventListener("click", () => {
      checkAnswer(option.get("id"));
    });

    optionsList.appendChild(li);
  });
}

function nextQuestion() {
  currentQuestionIndex++;
  loadQuestion();
}

// Check if the selected answer is correct
function checkAnswer(selectedOption) {
  const currentQuestion = data.get("questions")[currentQuestionIndex];
  const feedback = document.querySelector(
    ".custom-articles-challenge__feedback",
  );
  let isCorrect = false;

  // Show feedback based on whether the answer is correct or not
  if (selectedOption === currentQuestion.get("option")) {
    correctAnswers++;
    isCorrect = true;
    feedback.textContent = window.konnektoren.tr("Correct!");
    feedback.classList.add("custom-articles-challenge__feedback--correct");
    feedback.classList.remove("custom-articles-challenge__feedback--incorrect");
  } else {
    feedback.textContent = window.konnektoren.tr("Incorrect!");
    feedback.classList.add("custom-articles-challenge__feedback--incorrect");
```

```
    feedback.classList.remove("custom-articles-challenge__feedback--correct");
  }

  // Record user's answer
  userAnswers.push({
    questionId: currentQuestion.id,
    selectedOption: selectedOption,
    correctOption: currentQuestion.option,
    isCorrect: isCorrect,
  });

  feedback.style.display = "inline-block"; // This line now directly shows the
feedback

  if (currentQuestionIndex === data.get("questions").length - 1) {
    finishChallenge();
    return;
  } else {
    setTimeout(() => {
      feedback.style.display = "none"; // Hide feedback before moving to next question
      nextQuestion();
    }, 1000);
  }
}

function calculatePerformance() {
  const totalQuestions = data.get("questions").length;
  const performance = correctAnswers / totalQuestions;
  return performance;
}

// Automatically load the first question on page load
translateStaticText();
loadQuestion();

const finishButton = document.querySelector(
  ".custom-articles-challenge__button--finish",
);

if (finishButton) {
  finishButton.addEventListener("click", function () {
    finishChallenge();
  });
}

function displayResults() {
  // Check if window.konnektoren and result data are available
  const result = window.konnektoren && window.konnektoren.result;

  if (!result) {
    console.warn("No result data available in window.konnektoren.result.");
```

```
    return; // Exit the function if no result data
  }

  // Access the challenge data (questions and options)
  const challengeData =
    window.konnektoren &&
    window.konnektoren.challenge &&
    window.konnektoren.challenge.data;

  if (!challengeData) {
    console.warn(
      "No challenge data available in window.konnektoren.challenge.data.",
    );
    return; // Exit the function if no challenge data
  }

  // Get the performance and answers
  const performance = result.performance;
  const answersMap = result.data.get("answers");

  console.log("answers", answersMap);

  // Try to get the performance wrapper element
  const performanceWrapper = document.querySelector(
    ".custom-articles-challenge__performance-wrapper",
  );

  if (performanceWrapper) {
    // Update the text content with the translated version
    performanceWrapper.innerHTML = `
      ${window.konnektoren.tr("Your performance")}:
      <span class="custom-articles-challenge__performance"></span>%
    `;

    // Get the performance span element
    const performanceElement = performanceWrapper.querySelector(
      ".custom-articles-challenge__performance",
    );

    if (performanceElement) {
      // Display performance percentage
      performanceElement.textContent = (performance * 100).toFixed(2);
    } else {
      console.warn(
        'Element with class "custom-articles-challenge__performance" not found.',
      );
    }
  } else {
    console.warn(
      'Element with class "custom-articles-challenge__performance-wrapper" not found.
',
```

```
    );
  }

  // Try to get the results container element
  const resultsContainer = document.querySelector(
    ".custom-articles-challenge__results-container",
  );

  if (resultsContainer) {
    let correctCount = 0;
    let incorrectCount = 0;

    // Create overview section
    const overviewSection = document.createElement("div");
    overviewSection.className = "custom-articles-challenge__overview";

    // Create detailed results list
    const resultsList = document.createElement("ul");
    resultsList.className = "custom-articles-challenge__results-list";

    // Extract answers, questions, and options
    const answers = Array.from(answersMap);
    const questions = Array.from(challengeData.get("questions"));
    const options = Array.from(challengeData.get("options"));

    // Iterate through answers using index
    answers.forEach((answer, index) => {
      // Get the corresponding question from the array using the index
      const question = questions[index]; // Use index to access the question

      // Check if the question is found
      if (question) {
        const selectedOption = options.find(
          (o) => o.get("id") === answer.get("selectedOption"),
        );
        const correctOption = options.find(
          (o) => o.get("id") === question.get("option"),
        );

        if (answer.get("isCorrect")) {
          correctCount++;
        } else {
          incorrectCount++;
        }

        const listItem = document.createElement("li");
        listItem.className = "custom-articles-challenge__result-item";

        listItem.innerHTML = `
          <p class="custom-articles-challenge__result-question">
<strong>${window.konnektoren.tr("Question")}:</strong> ${question.get("question")}</p>
```

```
          <p class="custom-articles-challenge__result-answer">
<strong>${window.konnektoren.tr("Your answer")}:</strong> ${selectedOption ?
selectedOption.get("name") : "Option not found"} ${answer.get("isCorrect") ? "" :
""}</p>
          <p class="custom-articles-challenge__result-correct">
<strong>${window.konnektoren.tr("Correct answer")}:</strong> ${correctOption ?
correctOption.get("name") : "Option not found"}</p>
          <p class="custom-articles-challenge__result-help">
<strong>${window.konnektoren.tr("Help")}:</strong> ${question.get("help")}</p>
        `;

        resultsList.appendChild(listItem);
      } else {
        console.warn("Question not found at index:", index);
      }
    });

    // Add overview to the overview section
    overviewSection.innerHTML = `
      <h3 class="custom-articles-challenge__overview-title"
>${window.konnektoren.tr("Overview")}</h3>
      <p class="custom-articles-challenge__overview-item custom-articles-
challenge__overview-item--correct">${window.konnektoren.tr("Correct Answers")}:
${correctCount}</p>
      <p class="custom-articles-challenge__overview-item custom-articles-
challenge__overview-item--incorrect">${window.konnektoren.tr("Incorrect Answers")}:
${incorrectCount}</p>
    `;

    // Append overview and results list to the results container
    resultsContainer.appendChild(overviewSection);
    resultsContainer.appendChild(resultsList);
  } else {
    console.warn(
      'Element with class "custom-articles-challenge__results-container" not found.',
    );
  }
}

// Call the function to display results
displayResults();
```

- Add a new challenge data to the challenges folder like `src/assets/challenges/custom_articles.yml`

```
!custom
id: "custom_articles"
name: "Artikel"
description: "Learn the articles of the German language"
html: "/assets/challenges/html/custom_articles.html"
```

```yaml
results_html: "/assets/challenges/html/custom_articles_results.html"
css: "/assets/challenges/css/custom_articles.css"
js: "/assets/challenges/js/custom_articles.js"
i18n: "/assets/challenges/i18n/custom_articles.yml"
data:
  options:
    - id: 0
      name: "der"
    - id: 1
      name: "die"
    - id: 2
      name: "das"
  questions:
    - question: "Haus"
      help: "Ich habe ein neues Haus gekauft."
```

- Update the level loader `src/model/level_loader.rs` to include the new challenge design

```
        let custom_interrogative_particles: ChallengeType =
serde_yaml::from_str(include_str!(
            "../assets/challenges/custom_interrogative_particles.yml"
        ))?;
        let custom_modalverben: ChallengeType =

serde_yaml::from_str(include_str!("../assets/challenges/custom_modalverben.yml"))?;

        Ok(ChallengeFactory {
            challenge_types: vec![
                custom_past_tenses,
                custom_verbs_prepositions,
                custom_interrogative_particles,
                custom_modalverben,
            ],
        })
    }
```

- Add the new challenge design to the level file `src/assets/challenges/level_c1.yml`

```yaml
  - id: "custom-articles-1"
    name: "Artikel"
    description: "Learn the articles of the German language"
    challenge: "custom_articles"
    variant: "custom"
    tasks: 10
    unlock_points: 0
    position: [7, 2]
```

JavaScript Integration with konnektoren.js

The project uses a `window.konnektoren` object in JavaScript to handle challenge data and event handling for custom challenges. This section provides an overview of how this integration works.

Overview of `window.konnektoren`

When the application loads, `window.konnektoren` is initialized as a global JavaScript object. It contains two key components:

1. `challenge`: Stores the challenge data, which can be accessed as a plain JavaScript object.
2. `sendEvent`: A function that can be called to send challenge-related events from JavaScript to Rust.

Challenge Data

The `challenge` object holds various properties of the custom challenge. Here is an example of how the `challenge` object is structured:

```
{
  "id": "custom_articles",
  "name": "Artikel",
  "description": "Learn the articles of the German language",
  "html": "/assets/challenges/html/custom_articles.html",
  "css": "/assets/challenges/css/custom_articles.css",
  "js": "/assets/challenges/js/custom_articles.js",
  "i18n": "/assets/challenges/i18n/custom_articles.yml",
  "data": {
    "options": [
      { "id": 0, "name": "der" },
      { "id": 1, "name": "die" },
      { "id": 2, "name": "das" }
    ],
    "questions": [
      {
        "question": "Haus",
        "help": "Ich habe ein neues Haus gekauft.",
        "option": 2,
        "image": "fa-regular fa-house-chimney-window"
      }
    ]
  }
}
```

You can access specific properties of the `challenge` object using the `.get()` method if `data` is stored as a `Map`. For example, to access the `options` and `questions`, use the following:

```
console.log(window.konnektoren.challenge.data);
```

```
const data = window.konnektoren.challenge.data;

// Accessing options and questions using .get() for Map
const firstOption = data.get("options")[0];
console.log(firstOption.name); // Output: "der"

const firstQuestion = data.get("questions")[0];
console.log(firstQuestion.question); // Output: "Haus"
```

Note: If the `data` is stored as a plain object, use dot notation instead:

```
const data = window.konnektoren.challenge.data;
const firstOption = data.options[0];
console.log(firstOption.name); // Output: "der"
```

Event Handling with `sendEvent`

`window.konnektoren` also exposes a `sendEvent` function that allows sending events from JavaScript to Rust. The possible events that can be sent include:

- `NextTask(index)`: Triggered when the user moves to the next task.
- `PreviousTask(index)`: Triggered when the user moves to the previous task.
- `SolvedCorrect(index)`: Triggered when the user answers a task correctly.
- `SolvedIncorrect(index)`: Triggered when the user answers a task incorrectly.
- `Finish(ChallengeResult)`: Triggered when the challenge is completed.

Example of sending an event from JavaScript:

```
window.konnektoren.sendEvent({
  type: "NextTask",
  index: 1
});
window.konnektoren.sendEvent({
    type: "Finish",
    result: {
        correct: 5,
        incorrect: 2,
        total: 7
    }
})
```

Translation with 'tr'

The `window.konnektoren` object also provides a `tr` function that can be used to translate text in the custom challenge. The `tr` function takes a key as an argument and returns the translated text. The challenge data should contain a line for i18n that points to the translation file.

```
i18n:
  "German Article Exercise":
    "de": "Deutsche Artikel Übung"
    "en": "German Article Exercise"
    "es": "Ejercicio de artículos en alemán"
    "cn": "□□□□□□"
    "ua": "Німецька стаття вправа"
    "tr": "Almanca Artikel Egzersizi"
    "ar": "□□□□□ □□□□□ □□□□□□□□□"
  "Finish Exercise":
    "de": "Übung beenden"
    "en": "Finish Exercise"
    "es": "Terminar ejercicio"
    "cn": "□□□□"
    "ua": "Завершити вправу"
    "tr": "Egzersizi bitir"
    "ar": "□□□□□ □□□□□□□"
```

Achievements

Achievements are a way to reward players for their progress and engagement with the game. They are defined in a YAML file and evaluated based on various statistics tracked during gameplay.

Adding Achievements

To add new achievements, edit the `src/assets/achievements.yml` file. Each achievement is defined with the following structure:

```
achievements:
  - id: xp_master
    name: XP Master
    description: Earn 1000 XP
    icon: □
    condition: "total_xp > 1000"
  - id: challenge_champion
    name: Challenge Champion
    description: Complete 50 challenges
    icon: □
    condition: "total_challenges >= 50"
```

Key components of an achievement definition:

1. `id`: A unique identifier for the achievement.
2. `name`: The display name of the achievement.
3. `description`: A brief explanation of what the achievement represents.

4. `icon`: An emoji, Font Awesome icon class (prefixed with "fa-"), or image URL to represent the achievement visually.
5. `condition`: A string that defines the condition for unlocking the achievement.

Trackable Statistics

The following statistics can be used in achievement conditions:

1. `total_challenges`: The total number of challenges completed.
   - Usage: `"total_challenges >= 50"`
2. `average_performance`: The average performance across all challenges (as a percentage).
   - Usage: `"average_performance >= 90"`
3. `total_xp`: The total experience points earned.
   - Usage: `"total_xp > 1000"`
4. `completed_game_paths`: The number of game paths (levels) completed.
   - Usage: `"completed_game_paths >= 3"`
5. `perfect_challenges`: The number of challenges completed with a perfect score.
   - Usage: `"perfect_challenges >= 10"`
6. `different_challenge_types_completed`: The number of unique challenge types completed.
   - Usage: `"different_challenge_types_completed >= 5"`

Condition Syntax

The condition string uses a simple comparison syntax:

- Supported operators: `>`, `>=`, `<`, `⇐`, `==`
- The left side of the condition should be one of the trackable statistics.
- The right side should be a numeric value.

Examples:

```
condition: "total_xp > 1000"
condition: "average_performance >= 95"
condition: "completed_game_paths == 5"
```