Developing an HTML5 Angel Game with AI Assistance
A Practical Example of AI-Supported Educational Game Development

Abstract

This project demonstrates how an HTML5 browser game can be developed through iterative collaboration with an AI assistant. The game features an angel character that avoids clouds and birds while the difficulty gradually increases. During the development process, functionality such as image graphics, keyboard controls, collision detection, score tracking, game restart mechanisms, mobile support, and progressive difficulty scaling were added step by step.

The project illustrates how educators and non-programmers can use AI as a coding partner to create playable educational games.

Initial Game Concept

The original goal was:

Create an HTML5 browser game in which an angel avoids clouds and different birds.

The first implementation used:

- HTML5 Canvas
- JavaScript animation loop
- Simple rectangular shapes
- Keyboard controls
- Collision detection
- Score counting

The angel moved vertically while obstacles approached from the right side of the screen.

Replacing Shapes with Graphics

The next development step was replacing geometric shapes with image graphics.

Three images were introduced:

<img id="enkeliKuva"

src="**https://lastenkirkko.fi/wp-content/uploads/2023/05/enkeli.png**"

style="display:none;">

<img id="lintuKuva"

src="**https://lastenkirkko.fi/wp-content/uploads/2023/05/lintu.png**"

style="display:none;">

<img id="pilviKuva"

src="https://lastenkirkko.fi/wp-content/uploads/2023/05/Pilvi-1.png"

style="display:none;">

Drawing was performed using:

ctx.drawImage(...)

This significantly improved the visual quality of the game.

Keyboard Controls

The first control system used the Space key.

Later, the controls were changed to:

Key	**Action**
W	Move up
S	Move down
A	Move left
D	Move right
Arrow Keys	Alternative movement
Enter	Restart game

Example:

```
if (e.key === 'ArrowUp' || e.key === 'w' || e.key === 'W') {
    uusiY -= 30;
}
```

This made the game familiar to players accustomed to standard gaming controls.

Preventing Movement Outside the Canvas

A common issue in beginner game development is allowing the player character to leave the visible play area.

Boundary checking was added:

```
if (uusiX >= 0 &&
   uusiX <= canvas.width - enkeliWidth) {
   enkeliX = uusiX;
}
if (uusiY >= 0 &&
   uusiY <= canvas.height - enkeliHeight) {
   enkeliY = uusiY;
}
```

This keeps the angel fully visible at all times.

Collision Detection

The game uses rectangular collision detection.

Example:

```
if (
   enkeliX < pilvi.x + pilviLeveys &&
   enkeliX + enkeliWidth > pilvi.x &&
   enkeliY < pilvi.y + pilviKorkeus &&
   enkeliY + enkeliHeight > pilvi.y
) {
   return true;
}
```

When a collision occurs:

1. The game stops.
2. The score is shown.

3. The player can restart.

Progressive Difficulty

To keep gameplay engaging, obstacle speed increases as the score grows.

Initial values:

```
let pilviNopeus = 2;

let lintuNopeus = 4;
```

Speed increase:

```
if (pisteet % nopeudenKasvu === 0) {
  pilviNopeus += 0.5;
  lintuNopeus += 0.5;
}
```

This creates a gradual increase in challenge.

Score Tracking

The score increases continuously during gameplay.

```
pisteet++;
```

Displayed on screen:

```
ctx.fillText("Score: " + pisteet, 20, 40);
```

This provides immediate feedback to the player.

Restart Functionality

Several restart methods were explored:

Automatic Restart

```
setTimeout(aloitaPeli, 1000);
```

Enter Key Restart

```
if (e.key === 'Enter') {
  aloitaPeli();
}
```

Button Restart

```
<button id="uudelleenLatausPainike">
  Restart Game
</button>
uudelleenLatausPainike.addEventListener(
  'click',
  () => {
    aloitaPeli();
  }
);
```

Mobile Device Support

To support smartphones and tablets, touch controls were added.

Touch events:

touchstart

touchmove

touchend

Main function:

function liikutaEnkeliaKosketuksella(e)

The angel follows the player's finger position on the screen.

This transformed the game from a desktop-only application into a cross-platform HTML5 game.

Educational Significance

This project demonstrates several important concepts:

Programming Concepts

- Variables
- Functions
- Arrays
- Event handling
- Animation loops

- Collision detection
- Random object generation

Game Development Concepts

- Player control
- Obstacle spawning
- Difficulty scaling
- Scoring systems
- Game states
- Restart mechanics

Educational Technology Concepts

- AI-assisted coding
- Rapid prototyping
- Iterative development
- Creative game design
- Digital storytelling

Role of Artificial Intelligence

The development process followed a conversational model:

1. Define an idea.
2. Generate initial code.
3. Test.
4. Request improvements.
5. Refine functionality.
6. Repeat.

Examples of user requests included:

- Replace shapes with graphics.
- Add keyboard controls.
- Add mobile support.

- Increase difficulty over time.
- Prevent leaving the canvas.
- Restart after collision.
- Add reload buttons.

The AI acted as:

- Programmer
- Tutor
- Debugging assistant
- Design consultant

This demonstrates how AI can lower the barrier to entry for educational game development.

Final Outcome

The final product is a complete HTML5 game featuring:

✅ Angel character graphics
✅ Bird and cloud obstacles
✅ Keyboard controls (WASD and Arrow Keys)
✅ Mobile touch controls
✅ Collision detection
✅ Score tracking
✅ Increasing difficulty
✅ Restart functionality
✅ Browser compatibility
✅ Educational programming value

Conclusion

This project illustrates how modern AI tools can support educators, students, and content creators in developing interactive learning experiences. Through an iterative dialogue process, a simple game concept evolved into a fully functional HTML5 application suitable for demonstrations, workshops, and educational technology conferences.

Keywords: HTML5, JavaScript, Canvas, Educational Games, Artificial Intelligence, Game Development, Mobile Learning, Educational Technology, AI-Assisted Programming.

The final Code

```
<!DOCTYPE html>
<html lang="fi">
<head>
<meta charset="UTF-8">
<title>Enkelipeli</title>

<style>
body {
margin: 0;
text-align: center;
font-family: Arial, sans-serif;
background-color: #e6f7ff;
}

h1 {
margin-top: 20px;
}

canvas {
display: block;
margin: 20px auto;
background-color: skyblue;
border: 3px solid white;
max-width: 100%;
touch-action: none;
}

button {
padding: 10px 20px;
font-size: 16px;
margin-bottom: 20px;
cursor: pointer;
}
</style>
</head>

<body>

<h1>Enkelipeli</h1>
<p>Käytä nuolinäppäimiä tai W, A, S, D -näppäimiä. Mobiilissa liikuta enkeliä sormella.</p>

<canvas id="enkelipeli" width="800" height="600"></canvas>
```

```
<button id="uudelleenLatausPainike">Aloita alusta</button>

<img id="enkeliKuva" src="https://lastenkirkko.fi/wp-content/uploads/2023/05/enkeli.png"
style="display:none;">
<img id="lintuKuva" src="https://lastenkirkko.fi/wp-content/uploads/2023/05/lintu.png"
style="display:none;">
<img id="pilviKuva" src="https://lastenkirkko.fi/wp-content/uploads/2023/05/Pilvi-1.png"
style="display:none;">

<script>
const canvas = document.getElementById('enkelipeli');
const ctx = canvas.getContext('2d');

const enkeliKuva = document.getElementById('enkeliKuva');
const lintuKuva = document.getElementById('lintuKuva');
const pilviKuva = document.getElementById('pilviKuva');

const uudelleenLatausPainike = document.getElementById('uudelleenLatausPainike');

let enkeliX;
let enkeliY;

const enkeliLeveys = 100;
const enkeliKorkeus = 100;

const pilviLeveys = 120;
const pilviKorkeus = 80;

const lintuLeveys = 80;
const lintuKorkeus = 80;

let pilviNopeus;
let lintuNopeus;

let pilvet = [];
let linnut = [];

let pisteet = 0;
let peliKaynnissa = true;

const nopeudenKasvu = 300;

function aloitaPeli() {
```

```
enkeliX = canvas.width / 4;
enkeliY = canvas.height / 2;

pilvet = [];
linnut = [];

pisteet = 0;
pilviNopeus = 2;
lintuNopeus = 4;

peliKaynnissa = true;

requestAnimationFrame(paivitaPeli);
}

function piirraEnkeli() {
ctx.drawImage(
enkeliKuva,
enkeliX,
enkeliY,
enkeliLeveys,
enkeliKorkeus
);
}

function piirraPilvet() {
for (const pilvi of pilvet) {
ctx.drawImage(
pilviKuva,
pilvi.x,
pilvi.y,
pilviLeveys,
pilviKorkeus
);
}
}

function piirraLinnut() {
for (const lintu of linnut) {
ctx.drawImage(
lintuKuva,
lintu.x,
lintu.y,
```

```
lintuLeveys,
lintuKorkeus
);
}
}

function piirraPisteet() {
ctx.fillStyle = "white";
ctx.font = "28px Arial";
ctx.fillText("Pisteet: " + pisteet, 20, 40);
}

function paivitaPeli() {
if (!peliKaynnissa) return;

ctx.clearRect(0, 0, canvas.width, canvas.height);

piirraEnkeli();
piirraPilvet();
piirraLinnut();
piirraPisteet();

for (const pilvi of pilvet) {
pilvi.x -= pilviNopeus;
}

for (const lintu of linnut) {
lintu.x -= lintuNopeus;
}

if (Math.random() < 0.015) {
pilvet.push({
x: canvas.width,
y: Math.random() * (canvas.height - pilviKorkeus)
});
}

if (Math.random() < 0.008) {
linnut.push({
x: canvas.width,
y: Math.random() * (canvas.height - lintuKorkeus)
});
}
```

```
pilvet = pilvet.filter(pilvi => pilvi.x + pilviLeveys > 0);
linnut = linnut.filter(lintu => lintu.x + lintuLeveys > 0);

pisteet++;

if (pisteet % nopeudenKasvu === 0) {
pilviNopeus += 0.5;
lintuNopeus += 0.5;
}

if (tormays()) {
peliKaynnissa = false;

setTimeout(() => {
if (confirm("Peli päättyi! Sait " + pisteet + " pistettä. Aloitetaanko uudelleen?")) {
aloitaPeli();
}
}, 100);

return;
}

requestAnimationFrame(paivitaPeli);
}

function tormays() {
for (const pilvi of pilvet) {
if (
enkeliX < pilvi.x + pilviLeveys &&
enkeliX + enkeliLeveys > pilvi.x &&
enkeliY < pilvi.y + pilviKorkeus &&
enkeliY + enkeliKorkeus > pilvi.y
) {
return true;
}
}

for (const lintu of linnut) {
if (
enkeliX < lintu.x + lintuLeveys &&
enkeliX + enkeliLeveys > lintu.x &&
enkeliY < lintu.y + lintuKorkeus &&
```

```
enkeliY + enkeliKorkeus > lintu.y
) {
return true;
}
}

return false;
}

function liikutaEnkelia(e) {
let uusiX = enkeliX;
let uusiY = enkeliY;

if (e.key === 'ArrowUp' || e.key === 'w' || e.key === 'W') {
uusiY -= 30;
} else if (e.key === 'ArrowDown' || e.key === 's' || e.key === 'S') {
uusiY += 30;
} else if (e.key === 'ArrowLeft' || e.key === 'a' || e.key === 'A') {
uusiX -= 30;
} else if (e.key === 'ArrowRight' || e.key === 'd' || e.key === 'D') {
uusiX += 30;
} else if (e.key === 'Enter') {
aloitaPeli();
}

if (uusiX >= 0 && uusiX <= canvas.width - enkeliLeveys) {
enkeliX = uusiX;
}

if (uusiY >= 0 && uusiY <= canvas.height - enkeliKorkeus) {
enkeliY = uusiY;
}
}

function liikutaEnkeliaKosketuksella(e) {
e.preventDefault();

const rect = canvas.getBoundingClientRect();
const kosketuspiste = e.touches[0];

const scaleX = canvas.width / rect.width;
const scaleY = canvas.height / rect.height;
```

```
let uusiX = (kosketuspiste.clientX - rect.left) * scaleX - enkeliLeveys / 2;
let uusiY = (kosketuspiste.clientY - rect.top) * scaleY - enkeliKorkeus / 2;

uusiX = Math.max(0, Math.min(canvas.width - enkeliLeveys, uusiX));
uusiY = Math.max(0, Math.min(canvas.height - enkeliKorkeus, uusiY));

enkeliX = uusiX;
enkeliY = uusiY;
}

document.addEventListener('keydown', liikutaEnkelia);

canvas.addEventListener('touchstart', liikutaEnkeliaKosketuksella, { passive: false });
canvas.addEventListener('touchmove', liikutaEnkeliaKosketuksella, { passive: false });

uudelleenLatausPainike.addEventListener('click', () => {
aloitaPeli();
});

aloitaPeli();
</script>

</body>
</html>
```

Images