Plan:

1. **Add More Player Actions**: Right now, the player can move left, right, jump, attack, and switch weapons. Consider adding more actions such as crouching or sprinting. This can make the game more engaging.
2. **Improve Monster AI**: Currently, your monsters move towards the player at a constant speed. You could make the monsters smarter. For example, the flying monster could try to stay above the player and dive down to attack. The walking monster could jump if the player is above it.
3. **Add More Weapons**: You have a sword and a bow right now. Consider adding more weapons with different characteristics. For example, a magic staff that shoots projectiles in all directions but does less damage.
4. **Add Power-ups**: Power-ups can add a lot of fun to the game. They can be temporary boosts (like invincibility or double damage) or permanent upgrades (like increased health or speed).
5. **Add Levels or Waves**: Right now, the game has one set of monsters. Consider adding levels or waves of monsters. Each wave could be harder than the last, with more monsters, faster monsters, or monsters with more health.
6. **Add a Score System**: Right now, the game doesn't have a clear goal. Adding a score system can give players a goal to aim for. The score could increase for each monster defeated, with bonus points for defeating a wave or not taking damage.
7. **Improve the Game Over State**: When the game is over, you could display a screen that shows the player's score and offers the option to restart the game.

Remember, coding is a process of continuous learning and improvement. Don't worry if you don't get everything right the first time. Keep experimenting, keep learning, and most importantly, keep having fun! You're doing great, and I'm here to help you every step of the way.

Foreløpge Javakoder:

// Get a reference to the canvas var canvas = document.getElementById('gameCanvas'); var ctx = canvas.getContext('2d'); var gameOver = false; // Reference to health display element var healthDisplay = document.getElementById('health'); // Variable to store font-size for the health text var healthFontSize = 20; // Define arrows var arrows = []; // Player controls var keys = { left: false, right: false, up: false, space: false, // Attack key w: false // Switch weapon key }; // Keep track of current level var level = 1; // Listen for keydown events window.addEventListener('keydown', function(event) { switch (event.key) { case 'ArrowLeft': keys.left = true; break; case 'ArrowRight': keys.right = true; break; case 'ArrowUp': keys.up = true; break; case ' ': keys.space = true; break; case 'w': keys.w = true; break; } }); // Listen for keyup events window.addEventListener('keyup', function(event) { switch (event.key) { case 'ArrowLeft': keys.left = false; break; case 'ArrowRight': keys.right = false; break; case 'ArrowUp': keys.up = false; break; case ' ': keys.space = false; break; case 'w': keys.w = false; break; } }); // Define the weapons var weapons = [ { name: 'Sword', type: 'melee', damage: 10, range: 50 }, { name: 'Bow', type: 'ranged', damage: 5, range: 100 } ]; // Define our player var player = { x: 0, y: 0, vy: 0, // Vertical velocity health: 100, weapon: weapons[0], // Start with the first weapon attack: function() { if (this.weapon.name === 'Sword') { // Check for collision with monsters for (var i = 0; i < monsters.length; i++) { var monster = monsters[i]; var dx = monster.x - player.x; var dy = monster.y - player.y; var distance = Math.sqrt(dx \* dx + dy \* dy); if (distance < this.weapon.range) { // If the player and monster are within sword range monster.health -= this.weapon.damage; if (monster.health <= 0) { monsters.splice(i, 1); // Remove the monster from the array console.log('Monster ' + i + ' died!'); } } } } else if (this.weapon.name === 'Bow') { // Create a new arrow object and add it to the arrows array arrows.push({ x: this.x, y: this.y, vx: 5, damage: this.weapon.damage }); } }, switchWeapon: function() { var currentWeaponIndex = weapons.indexOf(this.weapon); var nextWeaponIndex = (currentWeaponIndex + 1) % weapons.length; this.weapon = weapons[nextWeaponIndex]; document.getElementById('weapon').textContent = "Current Weapon: " + this.weapon.name; } }; // Define our platform var platform = { x: 0, y: 300, width: canvas.width, height: 10 }; // Define our monsters var monsters = [ { x: 300, y: platform.y - 50, health: 50, damage: 1, speed: 2, type: 'walking' }, // Walking Monster { x: 400, y: platform.y - 50, health: 50, damage: 1, speed: 2, type: 'flying', vy: 2 } // Flying monster ]; // Function to create monster function createMonsters() { monsters = []; for (var i = 0; i < level; i++) { var walkingMonsterHealth = 50 + level \* 10; var walkingMonsterSpeed = 2 + level \* 0.5; monsters.push({ x: 300 + i \* 50, y: platform.y - 50, health: walkingMonsterHealth, damage: 1, speed: walkingMonsterSpeed, type: 'walking' }); var flyingMonsterHealth = 50 + level \* 10; var flyingMonsterSpeed = 2 + level \* 0.5; monsters.push({ x: 400 + i \* 50, y: platform.y - 50, health: flyingMonsterHealth, damage: 1, speed: flyingMonsterSpeed, type: 'flying', vy: 2 }); } } // Gravity constant var gravity = 1; // Update player position based on keys function updatePlayer() { if (keys.left) { player.x -= 5; } if (keys.right) { player.x += 5; } if (keys.up && player.y === platform.y - 50) { // Only allow the player to jump if they're standing on the platform player.vy = -10; } // Apply gravity player.vy += gravity; // Move player vertically player.y += player.vy; // Collision detection with platform if (player.y > platform.y - 50) { player.y = platform.y - 50; player.vy = 0; } // Attack if (keys.space) { player.attack(); } // Switch weapon if (keys.w) { player.switchWeapon(); } // Check for collision with monsters for (var i = 0; i < monsters.length; i++) { var monster = monsters[i]; var dx = monster.x - player.x; var dy = monster.y - player.y; var distance = Math.sqrt(dx \* dx + dy \* dy); if (distance < 50) { // If the player and monster are touching player.health -= monster.damage; healthDisplay.textContent = "Health: " + player.health; // Update health display if (player.health <= 0) { player.health = 0; // Ensure doesn't go below 0 gameOver = true; // Game over console.log("Game Over"); return; } } } } // Update monsters function updateMonsters() { for (var i = 0; i < monsters.length; i++) { var monster = monsters[i]; // Move monster towards player if (monster.x < player.x) { monster.x += monster.speed; } else if (monster.x > player.x) { monster.x -= monster.speed; } // If the monster is flying, make it move up and down if (monster.type === 'flying') { // If the player is below the monster, make it dive down if (player.y > monster.y) { monster.vy = 2; } else { monster.vy = -2; } monster.y += monster.vy; } // If the monster is walking and the player is above it, make it jump if (monster.type === 'walking' && player.y < monster.y) { monster.vy = -10; // This value may need to be adjusted } // Apply gravity to the monster monster.vy += gravity; // Move monster vertically monster.y += monster.vy; // Collision detection with platform if (monster.y > platform.y - 50) { monster.y = platform.y - 50; monster.vy = 0; } } } // Render player function renderPlayer() { ctx.fillStyle = '#00f'; // Blue ctx.fillRect(player.x, player.y, 50, 50); } // Render platform function renderPlatform() { ctx.fillStyle = '#0f0'; // Green ctx.fillRect(platform.x, platform.y, platform.width, platform.height); } // Render monsters function renderMonsters() { ctx.fillStyle = '#f00'; // Red for (var i = 0; i < monsters.length; i++) { var monster = monsters[i]; ctx.fillRect(monster.x, monster.y, 50, 50); // Draw the health text above the monster ctx.fillStyle = '#fff'; // White ctx.font = healthFontSize + 'px Arial'; ctx.fillText(monster.health, monster.x + 15, monster.y - 10); } } // Update arrows function updateArrows() { for (var i = 0; i < arrows.length; i++) { var arrow = arrows[i]; arrow.x += arrow.vx; // Check for collision with monsters for (var j = 0; j < monsters.length; j++) { var monster = monsters[j]; // Check if the arrow's x and y coordinates are within the monster's bounding box if (arrow.x >= monster.x && arrow.x <= monster.x + 50 && arrow.y >= monster.y && arrow.y <= monster.y + 50) { // If a collision is detected, reduce the monster's health by the arrow's damage value monster.health -= arrow.damage; // Log the collision and the monster's health console.log('Collision detected! Monster ' + j + ' health: ' + monster.health); // Remove the arrow from the array arrows.splice(i, 1); // Check if the monster's health is zero or below if (monster.health <= 0) { // Remove the monster from the array monsters.splice(j, 1); // Log the monster's death console.log('Monster ' + j + ' died!'); } // Break out of the inner loop break; } } } } // Render arrows function renderArrows() { ctx.fillStyle = '#ff0'; // Yellow for (var i = 0; i < arrows.length; i++) { var arrow = arrows[i]; ctx.fillRect(arrow.x, arrow.y, 10, 2); // Check if the arrow goes off-screen if (arrow.x > canvas.width) { // Remove the arrow from the array arrows.splice(i, 1); } } } // Game loop function gameLoop() { // Check if game is over if (gameOver) { ctx.clearRect(0, 0, canvas.width, canvas.height); // Clear the canvas ctx.font = '50px Arial'; // Set the font for the game over text ctx.fillStyle = '#f00'; // Set the color for the game over text ctx.fillText('Game Over', canvas.width / 2, canvas.height / 2); // Display the game over text // Reset game state after a short delay setTimeout(function() { // Reset player state player.x = 0; player.y = 0; player.vy = 0; player.health = 100; player.weapon = weapons[0]; // Start with the first weapon healthDisplay.textContent = "Health: " + player.health; // Update health display // Reset level level = 1; // Create a new set of monsters createMonsters(); // Reset game over flag gameOver = false; // Request next frame requestAnimationFrame(gameLoop); }, 2000); // 2 second delay return; // Stop the game loop temporarily } // Clear canvas ctx.clearRect(0, 0, canvas.width, canvas.height); // Update game state updatePlayer(); updateMonsters(); updateArrows(); // Check if all monsters have been defeated if (monsters.length === 0) { level++; // Increase the level createMonsters(); // Spawn a new set of monsters } // Render game state renderPlayer(); renderPlatform(); renderMonsters(); renderArrows(); // Request next frame requestAnimationFrame(gameLoop); } // Function to create a new set of monsters function createMonsters() { monsters = []; for (var i = 0; i < level; i++) { monsters.push({ x: 300 + i \* 50, y: platform.y - 50, health: 50 + level \* 10, damage: 1, speed: 2 + level \* 0.5, type: 'walking' }); // Walking Monster monsters.push({ x: 400 + i \* 50, y: platform.y - 50, health: 50 + level \* 10, damage: 1, speed: 2 + level \* 0.5, type: 'flying', vy: 2 }); // Flying monster } } // Start the game loop gameLoop();

// Define a spawn rate for monsters

var spawnRate = 100; // The rate at which new enemies are spawned (lower number = more often)

// Function to spawn enemies

function spawnEnemies() {

if (Math.random() < 1 / spawnRate) {

// Spawn a new monster off screen to the right

monsters.push({

x: canvas.width,

y: platform.y - 50,

health: 50,

damage: 1,

speed: 2,

type: 'walking'

});

}

}

// Update monsters

function updateMonsters() {

for (var i = 0; i < monsters.length; i++) {

var monster = monsters[i];

// Move monster towards player

if (monster.x < player.x) {

monster.x += monster.speed;

} else if (monster.x > player.x) {

monster.x -= monster.speed;

}

// If the monster is flying, make it move up and down

if (monster.type === 'flying') {

monster.y += monster.vy;

if (monster.y < platform.y - 200 || monster.y > platform.y - 100) {

// If the monster has reached the top or bottom of its range, reverse its vertical velocity

monster.vy \*= -1;

}

}

// If the monster goes off screen to the left, remove it

if (monster.x < -50) {

monsters.splice(i, 1);

i--; // Decrement i so we don't skip the next monster

}

}

}

// Game loop

function gameLoop() {

// Check if game is over

if (gameOver) {

ctx.clearRect(0, 0, canvas.width, canvas.height); // Clear the canvas

ctx.font = '50px Arial'; // Set the font for the game over text

ctx.fillStyle = '#f00'; // Set the color for the game over text

ctx.fillText('Game Over', canvas.width / 2, canvas.height / 2); // Display the game over text

return; // Stop the game loop

}

// Clear canvas

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

// Update game state

updatePlayer();

spawnEnemies(); // Spawn enemies

updateMonsters();

updateArrows();

// Render game state

renderPlayer();

renderPlatform();

renderMonsters();

renderArrows();

// Request next frame

requestAnimationFrame(gameLoop);

}

// Start the game loop

gameLoop();

Hello, this is Bing. I can help you with your question.😊

To add a picture to the player or monster in the game using HTML, CSS and JavaScript, you need to do the following steps:

- Use the HTML `<img>` tag to embed an image in your web page. The `<img>` tag has two required attributes: `src` - Specifies the path to the image and `alt` - Specifies an alternate text for the image¹.

- Use the CSS `style` attribute or the `width` and `height` attributes to define the size and position of the image¹. You can also use the CSS `border-radius` property to create rounded or circular images².

- Use JavaScript to manipulate the image element according to your game logic. For example, you can use the `document.getElementById()` method to get a reference to the image element, and then use the `style` property to change its attributes such as `left`, `top`, `width`, `height`, etc. You can also use the `addEventListener()` method to attach event handlers to the image element, such as `click`, `mouseover`, `keydown`, etc.

Here is an example of how you can add a picture of a player to your game using HTML, CSS and JavaScript:

HTML:

```html

<!-- Create a canvas element for the game -->

<canvas id="gameCanvas" width="800" height="600"></canvas>

<!-- Create an image element for the player -->

<img id="player" src="player.png" alt="Player" style="position:absolute; left:0; top:0; width:100px; height:100px;">

```

CSS:

```css

/\* Make the image circular \*/

#player {

border-radius: 50%;

}

```

JavaScript:

```javascript

// Get a reference to the canvas

var canvas = document.getElementById('gameCanvas');

var ctx = canvas.getContext('2d');

// Get a reference to the player image

var player = document.getElementById('player');

// Define some variables for the player movement

var playerSpeed = 5;

var playerDirection = 'right';

// Define a function to update the player position

function updatePlayer() {

// Check if the player reaches the edge of the canvas

if (playerDirection === 'right' && player.offsetLeft + player.offsetWidth >= canvas.width) {

// Change direction to left

playerDirection = 'left';

} else if (playerDirection === 'left' && player.offsetLeft <= 0) {

// Change direction to right

playerDirection = 'right';

}

// Move the player according to its direction and speed

if (playerDirection === 'right') {

// Increase left position by speed

player.style.left = (player.offsetLeft + playerSpeed) + 'px';

} else if (playerDirection === 'left') {

// Decrease left position by speed

player.style.left = (player.offsetLeft - playerSpeed) + 'px';

}

}

// Define a function to draw the game on the canvas

function drawGame() {

// Clear the canvas

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

// Draw some background elements

ctx.fillStyle = 'lightblue';

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

ctx.fillStyle = 'green';

ctx.fillRect(0, canvas.height - 100, canvas.width, 100);

// Update and draw the player

updatePlayer();

}

// Call the drawGame function every 20 milliseconds

setInterval(drawGame, 20);

```

I hope this helps you with your game development. If you have any other questions, please let me know.😊

Kilde: Samtale med Bing, 21.5.2023(1) HTML Images - W3Schools. https://www.w3schools.com/html/html\_images.asp Åpnet 21.5.2023.

(2) CSS Styling Images - W3Schools. https://www.w3schools.com/css/css3\_images.asp Åpnet 21.5.2023.

(3) CSS Image Gallery - W3Schools. https://www.w3schools.com/Css/css\_image\_gallery.asp Åpnet 21.5.2023.