Free Course Create a Game HTML5 Canvas Simple Catcher Game JavaScript

Create a Game HTML5 Canvas Simple Catcher Game JavaScript

https://www.udemy.com/course/javascript-html-game/

Explore how you can create an HTML5 Canvas based game from scratch using JavaScript Step by Step game creation

Do you want to learn how to make Games online ????

Start here - perfect to see how a game can be **created from scratch using HTML5 and JavaScript**

No libraries no shortcuts - straight pure JavaScript to draw on canvas and add game controls and interactions.

This is a one of kind build that you will only get here - custom code from start to finish -

Step by step learning on how to create all the game functionality you need to complete a full functional game.

Quick tutorial in just over 1 hour will give you a fast paced no filler content course that you can learn from.

Professional instruction - taught by a developer with over 20 years experience - having developed over 100 web games.

Instructor is ready to help you learn and answer any questions you may have.

Explore how you can create a quick simple game using JavaScript and HTML5 canvas element

Learn coding with FUN interactive game development - See what you can build with JavaScript!!!

Course covers all the core functions needed in a game

- Basics of drawing on HTML5 canvas
- Clearing and drawing shapes
- Adding key event listeners
- Making objects MOVE with keypresses
- Tracking key presses to make game interaction.
- Collision detection
- Adding buttons that are interactive on the canvas
- Adding game scoring and game controls
- Creating enemies and building unlimited enemies within an array
- Movement and animation of objects

Source code is included so you can build your own version of the game as you go through the lessons

https://www.udemy.com/course/javascript-html-game/

The below code is a JavaScript game that creates a canvas element, sets up player and enemy objects, and listens for user input to control the player object.

The canvas element is created using the document.createElement() method and assigned to the canvas variable. The size of the canvas is then set using the setAttribute() method, and the background color is set using the style property.

The enemies object contains properties for the speed of enemies, an array to store enemy objects, and a total number of enemies to generate.

The game object contains properties to track the state of the game, including whether the game is currently being played (play) and a request ID (req) that is used to cancel the game loop.

The btnPos object contains properties for the position, width, and height of a button that is used to start the game.

A button element is created using the document.createElement() method and assigned to the btn variable. The text content of the button is set using the textContent property, and an event listener is added using the addEventListener() method to start the game when the button is clicked.

An event listener is also added to the canvas element using the addEventListener() method to handle clicks. When the canvas is clicked and the game is not currently being played, the position of the mouse click is calculated relative to the canvas using the getBoundingClientRect() method. If the mouse click is within the bounds of the button, the game starts.

The startScreen() function is called to display the start screen when the page is loaded. The function draws a rectangle on the canvas to represent the button and adds text that prompts the user to click to start the game.

The player object contains properties for the position, speed, size, color, score, and number of lives.

The keyz object contains properties to track whether the arrow keys are currently being pressed.

Event listeners are added to the document using the addEventListener() method to listen for keydown and keyup events. When an arrow key is pressed or released, the corresponding property in the keyz object is updated.

The col() function checks whether two objects are colliding by comparing their positions and dimensions. If a collision is detected, the function logs a message and returns true.

The enemyMaker() function generates a new enemy object with a random position, size, color, and speed. The object is added to the enemies.arr array.

The gameOver() function is called when the game is over. It cancels the game loop using the cancelAnimationFrame() method,

sets the play property to false, and displays the game over screen.

The gameStart() function is called when the game is started. It initializes the game state, generates enemies, and starts the game loop using the requestAnimationFrame() method.

The draw() function is called on each frame of the game loop. It clears the canvas using the clearRect() method, updates the position of the player object based on user input, and updates the position of enemy objects. If an enemy object goes off the bottom of the canvas, it is removed from the enemies.arr array. The player and enemy objects are then drawn on the canvas using the fillRect() method, and the score and number of lives are displayed using the fillText() method. If the player runs out of lives, the game over function is called.

```
const canvas = document.createElement('canvas');
const tile = 25;
const enemies = {
  speed: 1
  , arr: []
  , total: 20
};
const game = {
  play: false
  , req: ''
};
canvas.setAttribute('height', tile * 20);
canvas.setAttribute('width', tile * 25);
canvas.style.backgroundColor = 'black';
const ctx = canvas.getContext('2d');
```

```
const btnPos = {
  x: 10
  , y: canvas.height / 2 - 100
  , width: canvas.width - 20
  , height: 100
};
const btn = document.createElement('button');
btn.textContent = "start game";
document.body.prepend(btn);
btn.addEventListener('click', () => {
  btn.style.display = 'none';
  gameStart();
});
canvas.addEventListener('click', (e) => {
  if (!game.play) {
    const rect = canvas.getBoundingClientRect();
    console.log(rect);
    const mouseObj = {
      x: e.clientX - rect.left
      , y: e.clientY - rect.top
      , width: 5
      , height: 5
    }
    if (col(btnPos, mouseObj)) {
      gameStart();
    }
  }
})

function startScreen() {
  let output = `Click to Start the Game`;
  ctx.beginPath();
```

```
  ctx.fillStyle = 'red';
  ctx.fillRect(btnPos.x, btnPos.y, btnPos.width,
btnPos.height);
  ctx.font = '24px arial';
  ctx.textAlign = 'center';
  ctx.fillStyle = 'white';
  ctx.fillText(output, canvas.width / 2, btnPos.y +
40);
}
document.body.prepend(canvas);
const player = {
  x: canvas.width / 2
  , y: canvas.height - (tile * 3)
  , speed: 5
  , width: tile * 4
  , height: tile * 1
  , color: 'red'
  , score: 0
  , lives: 0
};
const keyz = {
  ArrowLeft: false
  , ArrowRight: false
  , ArrowUp: false
  , ArrowDown: false
};
document.addEventListener('keydown', (e) => {
  if (e.code in keyz) {
    keyz[e.code] = true;
  }
})
document.addEventListener('keyup', (e) => {
```

```
  if (e.code in keyz) {
    keyz[e.code] = false;
  }
})
startScreen();

function col(a, b) {
  let boo = a.x < b.x + b.width && a.x + a.width > b.x
&& a.y < b.y + b.height && a.y + a.height > b.y;
  if (boo) {
    console.log('HIT');
  }
  return boo;
}

function enemyMaker() {
  let xPos = Math.random() * (canvas.width - tile);
  let badValue = Math.random() < 0.2;
  let colorBack = badValue ? 'red' : '#' +
Math.random().toString(16).substr(-6);
  let wid = Math.random() * 20 + 10;
  enemies.arr.push({
    x: xPos
    , y: Math.random() * -1000
    , width: wid * 2
    , height: wid * 2
    , size: wid
    , speed: Math.random() * 2 + 3
    , color: colorBack
    , bad: badValue
    , toggle: true
    , growth: 0
```

```
  })
}

function gameOver() {
  cancelAnimationFrame(game.req);
  game.play = false;
  btn.style.display = 'block';
  ctx.fillStyle = 'black';
  ctx.fillRect(0, 0, canvas.width, canvas.height);
  let output = `GAME OVER Your Score is
${player.score}`;
  ctx.beginPath();
  ctx.fillStyle = '#222';
  ctx.fillRect(10, 10, canvas.width - 20, 50);
  ctx.font = '24px arial';
  ctx.textAlign = 'center';
  ctx.fillStyle = 'white';
  ctx.fillText(output, canvas.width / 2, 40);
  startScreen();
}

function gameStart() {
  game.req = requestAnimationFrame(draw);
  game.play = true;
  enemies.arr = [];
  player.score = 0;
  player.lives = 3;
  player.x = canvas.width / 2;
  player.y = canvas.height - (tile * 3);
}

function draw() {
```

```
  ctx.clearRect(0, 0, canvas.width, canvas.height);
  if (enemies.arr.length < enemies.total) {
    //console.log(enemies);
    enemyMaker();
  }
  if (keyz.ArrowLeft && player.x > 0) {
    player.x -= player.speed;
  }
  if (keyz.ArrowRight && player.x < canvas.width -
player.width) {
    player.x += player.speed;
  }
  if (keyz.ArrowUp && player.y > canvas.height - tile *
8) {
    player.y -= player.speed;
  }
  if (keyz.ArrowDown && player.y < canvas.height -
tile) {
    player.y += player.speed;
  }
  ctx.fillStyle = player.color;
  ctx.fillRect(player.x, player.y, player.width,
player.height);
  enemies.arr.forEach((enemy, index) => {
    enemy.y += enemy.speed;
    if (enemy.y > canvas.height) {
      enemies.arr.splice(index, 1);
    }
    ctx.beginPath();
    ctx.fillStyle = enemy.color;
    if (enemy.toggle && enemy.bad) {
      enemy.growth++;
```

```
      enemy.size += 1;
      if (enemy.growth > 10) {
        enemy.toggle = false;
        enemy.growth = 0;
      }
    }
    else if (enemy.bad) {
      ctx.fillStyle = '#000000';
      enemy.growth++;
      if (enemy.growth > 10) {
        enemy.toggle = true;
        enemy.growth = 0;
      }
      enemy.size -= 1;
    }
    if (game.play) {
      ctx.strokeStyle = 'white';
      ctx.arc(enemy.x + (enemy.width / 2), enemy.y +
(enemy.height / 2), enemy.size, 0, Math.PI * 2);

//ctx.strokeRect(enemy.x,enemy.y,enemy.width,enemy.heig
ht);
      ctx.stroke();
      ctx.fill();
    }
    if (col(player, enemy)) {
      let removed = enemies.arr.splice(index, 1)[0];
      if (removed.bad) {
        player.lives--;
        if (player.lives < 0) {
          gameOver();
          player.lives = '-';
```

```
        }
      }
      else {
        player.score += Math.ceil(removed.size);
      }
      console.log(removed);
    }
  })
  if (game.play) {
    let output = `Live(s) : ${player.lives} Score :
${player.score}  `;
    ctx.beginPath();
    ctx.fillStyle = '#222';
    ctx.fillRect(10, 10, canvas.width - 20, 50);
    ctx.font = '24px arial';
    ctx.textAlign = 'center';
    ctx.fillStyle = 'white';
    ctx.fillText(output, canvas.width / 2, 40);

//ctx.fillRect(enemy.x,enemy.y,enemy.size,enemy.size);
    game.req = requestAnimationFrame(draw);
  }
}
```