Game Controllers

❮ Previous Next ❯

Push the buttons to move the red square:

UP

LEFT RIGHT

DOWN

Get in Control

Now we want to control the red square.

Add four buttons, up, down, left, and right.

Write a function for each button to move the component in the selected direction.

Make two new properties in the `component` constructor, and call them `speedX` and `speedY`. These properties are being used as speed indicators.

Add a function in the `component` constructor, called `newPos()`, which uses the `speedX` and `speedY` properties to change the component's position.

The newpos function is called from the updateGameArea function before drawing the component:

Example

```
<script>
function component(width, height, color, x, y) {
  this.width = width;
  this.height = height;
  this.speedX = 0;
  this.speedY = 0;
  this.x = x;
  this.y = y;
  this.update = function() {
    ctx = myGameArea.context;
    ctx.fillStyle = color;
    ctx.fillRect(this.x, this.y, this.width, this.height);
  }
  this.newPos = function() {
    this.x += this.speedX;
    this.y += this.speedY;
  }
}

function updateGameArea() {
  myGameArea.clear();
  myGamePiece.newPos();
  myGamePiece.update();
}

function moveup() {
  myGamePiece.speedY -= 1;
}

function movedown() {
  myGamePiece.speedY += 1;
}

function moveleft() {
  myGamePiece.speedX -= 1;
}

function moveright() {
  myGamePiece.speedX += 1;
}
</script>

<button onclick="moveup()">UP</button>
<button onclick="movedown()">DOWN</button>
<button onclick="moveleft()">LEFT</button>
<button onclick="moveright()">RIGHT</button>
```

Try it Yourself »

Stop Moving

If you want, you can make the red square stop when you release a button.

Add a function that will set the speed indicators to 0.

To deal with both normal screens and touch screens, we will add code for both devices:

Example

```
function stopMove() {
  myGamePiece.speedX = 0;
  myGamePiece.speedY = 0;
}
</script>

<button onmousedown="moveup()" onmouseup="stopMove()" ontouchstart="moveup()">UP</button>
<button onmousedown="movedown()" onmouseup="stopMove()" ontouchstart="movedown()">DOWN</button>
<button onmousedown="moveleft()" onmouseup="stopMove()" ontouchstart="moveleft()">LEFT</button>
<button onmousedown="moveright()" onmouseup="stopMove()" ontouchstart="moveright()">RIGHT</button>
```

Try it Yourself »

Keyboard as Controller

We can also control the red square by using the arrow keys on the keyboard.

Create a method that checks if a key is pressed, and set the `key` property of the `myGameArea` object to the key code. When the key is released, set the `key` property to `false`:

Example

```
var myGameArea = {
  canvas : document.createElement("canvas"),
  start : function() {
    this.canvas.width = 480;
    this.canvas.height = 270;
    this.context = this.canvas.getContext("2d");
    document.body.insertBefore(this.canvas, document.body.childNodes[0]);
    this.interval = setInterval(updateGameArea, 20);
    window.addEventListener('keydown', function (e) {
      myGameArea.key = e.keyCode;
    })
    window.addEventListener('keyup', function (e) {
      myGameArea.key = false;
    })
  },
  clear : function(){
    this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
  }
}
```

Then we can move the red square if one of the arrow keys are pressed:

Example

```
function updateGameArea() {
  myGameArea.clear();
  myGamePiece.speedX = 0;
  myGamePiece.speedY = 0;
  if (myGameArea.key && myGameArea.key == 37) {myGamePiece.speedX = -1; }
  if (myGameArea.key && myGameArea.key == 39) {myGamePiece.speedX = 1; }
  if (myGameArea.key && myGameArea.key == 38) {myGamePiece.speedY = -1; }
  if (myGameArea.key && myGameArea.key == 40) {myGamePiece.speedY = 1; }
  myGamePiece.newPos();
  myGamePiece.update();
}
```

Try it Yourself »

Multiple Keys Pressed

What if more than one key is pressed at the same time?

In the example above, the component can only move horizontally or vertically. Now we want the component to also move diagonally.

Create a `keys` *array* for the `myGameArea` object, and insert one element for each key that is pressed, and give it the value `true`, the value remains true untill the key is no longer pressed, the value becomes `false` in the *keyup* event listener function:

```
var myGameArea = {
  canvas : document.createElement("canvas"),
  start : function() {
    this.canvas.width = 480;
    this.canvas.height = 270;
    this.context = this.canvas.getContext("2d");
    document.body.insertBefore(this.canvas, document.body.childNodes[0]);
    this.interval = setInterval(updateGameArea, 20);
    window.addEventListener('keydown', function (e) {
      myGameArea.keys = (myGameArea.keys || []);
      myGameArea.keys[e.keyCode] = true;
    })
    window.addEventListener('keyup', function (e) {
      myGameArea.keys[e.keyCode] = false;
    })
  },
  clear : function(){
    this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
  }
}

function updateGameArea() {
  myGameArea.clear();
  myGamePiece.speedX = 0;
  myGamePiece.speedY = 0;
  if (myGameArea.keys && myGameArea.keys[37]) {myGamePiece.speedX = -1; }
  if (myGameArea.keys && myGameArea.keys[39]) {myGamePiece.speedX = 1; }
  if (myGameArea.keys && myGameArea.keys[38]) {myGamePiece.speedY = -1; }
  if (myGameArea.keys && myGameArea.keys[40]) {myGamePiece.speedY = 1; }
  myGamePiece.newPos();
  myGamePiece.update();
}
```

Try it Yourself »

Using The Mouse Cursor as a Controller

If you want to control the red square by using the mouse cursor, add a method in `myGameArea` object that updates the x and y coordinates of the mouse cursor:.

Example

```
var myGameArea = {
  canvas : document.createElement("canvas"),
  start : function() {
    this.canvas.width = 480;
    this.canvas.height = 270;
    this.canvas.style.cursor = "none"; //hide the original cursor
    this.context = this.canvas.getContext("2d");
    document.body.insertBefore(this.canvas, document.body.childNodes[0]);
    this.interval = setInterval(updateGameArea, 20);
    window.addEventListener('mousemove', function (e) {
      myGameArea.x = e.pageX;
      myGameArea.y = e.pageY;
    })
  },
  clear : function(){
    this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
  }
}
```

Then we can move the red square using the mouse cursor:

Example

```
function updateGameArea() {
  myGameArea.clear();
  if (myGameArea.x && myGameArea.y) {
    myGamePiece.x = myGameArea.x;
    myGamePiece.y = myGameArea.y;
  }
  myGamePiece.update();
}
```

Try it Yourself »

Touch The Screen to Control The Game

We can also control the red square on a touch screen.

Add a method in the `myGameArea` object that uses the x and y coordinates of where the screen is touched:

Example

```
var myGameArea = {
  canvas : document.createElement("canvas"),
  start : function() {
    this.canvas.width = 480;
    this.canvas.height = 270;
    this.context = this.canvas.getContext("2d");
    document.body.insertBefore(this.canvas, document.body.childNodes[0]);
    this.interval = setInterval(updateGameArea, 20);
    window.addEventListener('touchmove', function (e) {
      myGameArea.x = e.touches[0].screenX;
      myGameArea.y = e.touches[0].screenY;
    })
  },
  clear : function(){
    this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
  }
}
```

Then we can move the red square if the user touches the screen, by using the same code as we did for the mouse cursor:

Example

```
function updateGameArea() {
  myGameArea.clear();
  if (myGameArea.x && myGameArea.y) {
    myGamePiece.x = myGameArea.x;
    myGamePiece.y = myGameArea.y;
  }
  myGamePiece.update();
}
```

Try it Yourself »

Controllers on The Canvas

We can also draw our own buttons on the canvas, and use them as controllers:

Example

```
function startGame() {
  myGamePiece = new component(30, 30, "red", 10, 120);
  myUpBtn = new component(30, 30, "blue", 50, 10);
  myDownBtn = new component(30, 30, "blue", 50, 70);
  myLeftBtn = new component(30, 30, "blue", 20, 40);
  myRightBtn = new component(30, 30, "blue", 80, 40);
  myGameArea.start();
}
```

Add a new function that figures out if a component, in this case a button, is clicked.

Start by adding event listeners to check if a mouse button is clicked (`mousedown` and `mouseup`). To deal with touch screens, also add event listeners to check if the screen is clicked on (`touchstart` and `touchend`):

Example

```
var myGameArea = {
  canvas : document.createElement("canvas"),
  start : function() {
    this.canvas.width = 480;
    this.canvas.height = 270;
    this.context = this.canvas.getContext("2d");
    document.body.insertBefore(this.canvas, document.body.childNodes[0]);
    this.interval = setInterval(updateGameArea, 20);
    window.addEventListener('mousedown', function (e) {
      myGameArea.x = e.pageX;
      myGameArea.y = e.pageY;
    })
    window.addEventListener('mouseup', function (e) {
      myGameArea.x = false;
      myGameArea.y = false;
    })
    window.addEventListener('touchstart', function (e) {
      myGameArea.x = e.pageX;
      myGameArea.y = e.pageY;
    })
    window.addEventListener('touchend', function (e) {
      myGameArea.x = false;
      myGameArea.y = false;
    })
  },
  clear : function(){
    this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
  }
}
```

In the `updateGameArea` function, we take the neccessarry actions if one of the blue buttons is clicked:

Example

```
function component(width, height, color, x, y) {
  this.width = width;
  this.height = height;
  this.speedX = 0;
  this.speedY = 0;
  this.x = x;
  this.y = y;
  this.update = function() {
    ctx = myGameArea.context;
    ctx.fillStyle = color;
    ctx.fillRect(this.x, this.y, this.width, this.height);
  }
  this.clicked = function() {
    var myleft = this.x;
    var myright = this.x + (this.width);
    var mytop = this.y;
    var mybottom = this.y + (this.height);
    var clicked = true;
    if ((mybottom < myGameArea.y) || (mytop > myGameArea.y) || (myright < myGameArea.x) || (myleft > myGameArea.x)) {
      clicked = false;
    }
    return clicked;
  }
}

function updateGameArea() {
  myGameArea.clear();
  if (myGameArea.x && myGameArea.y) {
    if (myUpBtn.clicked()) {
      myGamePiece.y -= 1;
    }
    if (myDownBtn.clicked()) {
      myGamePiece.y += 1;
    }
    if (myLeftBtn.clicked()) {
      myGamePiece.x += -1;
    }
    if (myRightBtn.clicked()) {
      myGamePiece.x += 1;
    }
  }
  myUpBtn.update();
  myDownBtn.update();
  myLeftBtn.update();
  myRightBtn.update();
  myGamePiece.update();
}
```

Try it Yourself »

❮ Previous

Log in to track progress

Next ❯