웹 프로그래밍 강의노트

11. 게임 프로젝트

2023년 2학기

캔버스canvas

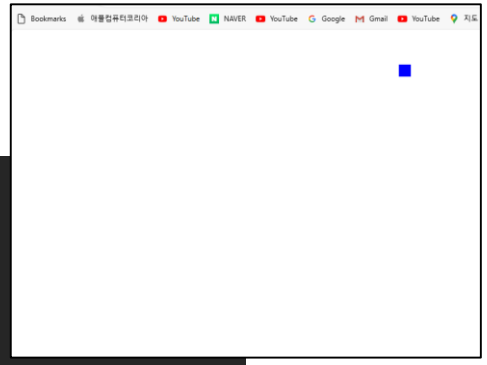
- HTML 〈canvas〉
  - JavaScript를 이용한 그리기를 가능하게 하는 요소
  - 콘테이너 객체로 동작
- 핵심적 기능
  - Drawing Surface
    - 그리기 동작의 결과를 보여준다
  - JavaScript API
    - 그리기 동작을 가능하게 하는 기능을 JavaScript API로 제공한다.

간단한 캔버스 예제

- 캔버스 준비

```
<body>
    <canvas id="myCanvas" width="400" height="200" style="border:1px solid #000;"></canvas>
    <script>
        // Get the canvas element
        const canvas = document.getElementById("myCanvas");
        // Get the rendering context (usually 2d for 2D graphics)
        const ctx = canvas.getContext("2d");
        // Draw a red rectangle ctx.fillStyle = "#FF0000";
        ctx.fillRect(50, 50, 100, 50);
    </script>
</body>
```

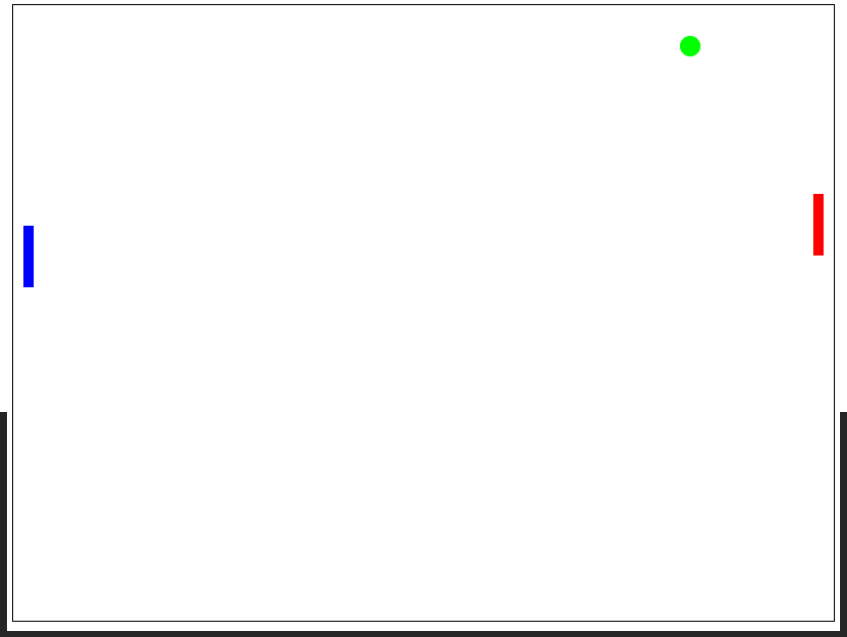
게임 – 루프



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

    <script>
        // Get the canvas element and its context
        const canvas = document.getElementById('gameCanvas');
        const ctx = canvas.getContext('2d');

        // Player object
        const player = {
            x: 50,
            y: 50,
            width: 20,
            height: 20,
            color: '#00F'
        };

        // Update function
        function update() {
            player.x += 2;
            if (player.x > canvas.width) {
                player.x = 0;
            }
        }
```

```
        // Render function
        function render() {
            ctx.clearRect(0, 0,
                          canvas.width, canvas.height);
            ctx.fillStyle = player.color;
            ctx.fillRect(player.x, player.y,
                         player.width, player.height);
        }

        // Game loop
        function gameLoop() {
            update();
            render();
            requestAnimationFrame(gameLoop);
        }

        // Start the game loop
        gameLoop();
    </script>
</body>
```

게임 인터랙션 – 객체 데이터



- 플레이어, 컴퓨터, 공

```
const canvas = document.getElementById('gameCanvas');
const ctx = canvas.getContext('2d');

// Paddle objects
const paddleWidth = 10;
const paddleHeight = 60;
const playerPaddle = { x: 10, y: canvas.height / 2 - paddleHeight / 2,
                       width: paddleWidth, height: paddleHeight, color: '#00F' };
const computerPaddle = { x: canvas.width - 20, y: canvas.height / 2 - paddleHeight / 2,
                         width: paddleWidth, height: paddleHeight, color: '#F00' };

// Ball object
const ball = { x: canvas.width / 2, y: canvas.height / 2, radius: 10, speedX: 2, speedY: 2, color: '#0F0' };
```

게임 인터랙션 – 이벤트 처리

```
// User input handling
upPressed = false;
downPressed = false;

document.addEventListener('keydown', keyDownHandler);
document.addEventListener('keyup', keyUpHandler);

function keyDownHandler(e) {
    if (e.key === 'ArrowUp') {
        upPressed = true;
    } else if (e.key === 'ArrowDown') {
        downPressed = true;
    }
}

function keyUpHandler(e) {
    if (e.key === 'ArrowUp') {
        upPressed = false;
    } else if (e.key === 'ArrowDown') {
        downPressed = false;
    }
}
```

게임 인터랙션 – 이벤트 처리에 따른 상태 갱신

```
// Move computer paddle
if (ball.y < computerPaddle.y + computerPaddle.height / 2) {    computerPaddle.y -= 1; }
else {   computerPaddle.y += 1; }

// Move the ball
ball.x += ball.speedX;
ball.y += ball.speedY;

// Check for collisions with walls
if (ball.y + ball.radius > canvas.height || ball.y - ball.radius < 0) { ball.speedY = -ball.speedY;  }

// Check for collisions with paddles
if (
    (ball.x - ball.radius < playerPaddle.x + playerPaddle.width && ball.y > playerPaddle.y && ball.y < playerPaddle.y + playerPaddle.height) ||
    (ball.x + ball.radius > computerPaddle.x && ball.y > computerPaddle.y && ball.y < computerPaddle.y + computerPaddle.height)   )
{
    ball.speedX = -ball.speedX;
}

// Game Reset
if (ball.x - ball.radius < 0 || ball.x + ball.radius > canvas.width)
{
    // Reset the ball position
    ball.x = canvas.width / 2;
    ball.y = canvas.height / 2;
}
```

게임 화면 그리기

```
// Render function
function render() {
    // Clear the canvas
    ctx.fillStyle = '#0AA'
    ctx.fillRect(0, 0, canvas.width, canvas.height);

    // Render paddles
    ctx.fillStyle = playerPaddle.color;
    ctx.fillRect(playerPaddle.x, playerPaddle.y, playerPaddle.width, playerPaddle.height);

    ctx.fillStyle = computerPaddle.color;
    ctx.fillRect(computerPaddle.x, computerPaddle.y,
                 computerPaddle.width, computerPaddle.height);

    // Render the ball
    ctx.beginPath();
    ctx.arc(ball.x, ball.y, ball.radius, 0, Math.PI * 2);
    ctx.fillStyle = ball.color;
    ctx.fill();
    ctx.closePath();
}
```

게임의 실행

```
// Game loop
function gameLoop() {
    // Update the game logic
    update();

    // Render the game
    render();

    // Request the next frame
    requestAnimationFrame(gameLoop);
}

// Start the game loop
gameLoop();
```

슈팅 게임 – 객체

```
<body>
  <canvas id="gameCanvas" width="800" height="400"></canvas>

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

    let player = {
      x: 50,
      y: canvas.height / 2,
      width: 50,
      height: 50,
      color: "#00F"
    };

    let bullets = [];
```

슈팅 게임 – 객체 그리기

```
function drawPlayer() {
  ctx.fillStyle = player.color;
  ctx.fillRect(player.x, player.y, player.width, player.height);
}
function drawBullets() {
  ctx.fillStyle = "#F00";
  for (let bullet of bullets) {
    ctx.fillRect(bullet.x, bullet.y, 10, 5);
  }
}
```

```
function updateBullets() {
  for (let bullet of bullets) {
    bullet.x += 5;

    // Check if bullet hits the right edge
of the canvas
    if (bullet.x > canvas.width) {
      bullets.splice(bullets.indexOf(bullet
), 1);
    }
  }
}
function gameLoop() {
  ctx.clearRect(0, 0, canvas.width,
canvas.height);
  drawPlayer();
  drawBullets();
  updateBullets();
  requestAnimationFrame(gameLoop);
}
```

슈팅 게임 – 객체 상태 변경

```
function updateBullets() {
  for (let bullet of bullets) {
    bullet.x += 5;

    // Check if bullet hits the right edge of the canvas
    if (bullet.x > canvas.width) {
      bullets.splice(bullets.indexOf(bullet), 1);
    }
  }
}

function gameLoop() {
  ctx.clearRect(0, 0, canvas.width, canvas.height);
  drawPlayer();
  drawBullets();
  updateBullets();
  requestAnimationFrame(gameLoop);
}
```

javascript Copy code

```
array.splice(start, deleteCount, item1, item2, ...);
```

- `start`: The index at which to start changing the array. If negative, it indicates an offset from the end of the array.
- `deleteCount`: The number of elements to remove from the array.
- `item1, item2, ...`: The elements to add to the array, starting at the `start` index.

이벤트 처리

```
function shoot() {
  bullets.push({ x: player.x + player.width, y: player.y + player.height / 2 });
}

window.addEventListener("keydown", function (event) {
  if (event.code == "Space") {
    shoot();
  }
  if (event.code == "ArrowUp") {
    player.y -= 5
  }
  if (event.code == "ArrowDown") {
    player.y += 5
  }
});

gameLoop();
```