Multimédia nappali gyakorlat

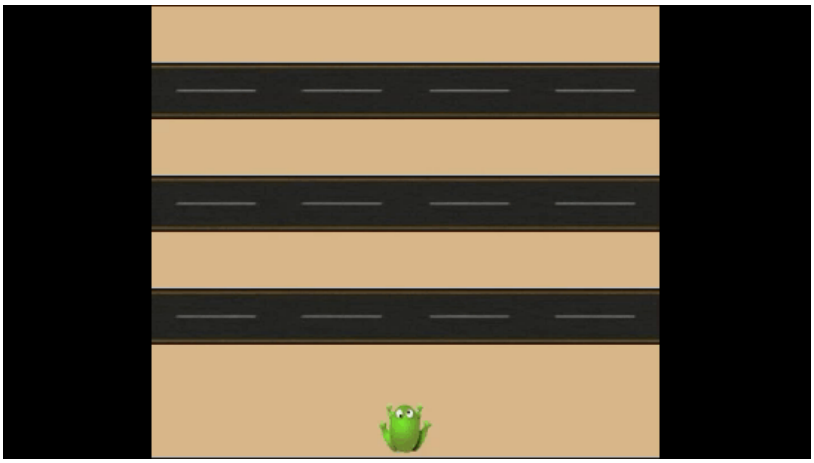
Egyszerű frogger játék

A tananyaghoz készült videó az alábbi linken érhető el: https://www.youtube.com/watch?v=-ugzRhxDkME

Utoljára módosítva
2020. április 29.

Észrevételek, javaslatok
✉ mkatona@inf.u-szeged.hu

Egyszerű frogger játék jQuery használatával



Megoldás: megoldas.zip

Megoldás menete

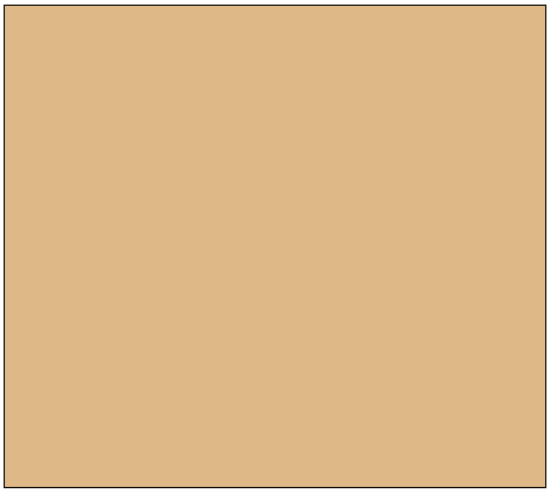
Indítsuk el a PhpStorm-ot vagy Webstorm-ot és hozzunk létre egy új projektet, amiben legyen egy `index.html` is. Először a játék alap felületét készítjük el. Létrehozzuk a `body`-ban a játékterületet és a `game_area` ID-vel látjuk el és ehhez kapcsolódóan különböző CSS tulajdonságokat rendelünk hozzá. Ahogy már korábbi példákban is, most középre pozícionáljuk a játékteret a `margin` és `position` tulajdonságok megfelelő értékadásával. Beállítunk egy alap háttérszínt, szegélyt, illetve a szélességet és magasságot is.

Amennyiben sikerült átjutnia a békának az akadálypályán, úgy szeretnénk majd megjeleníteni a WIN feliratot a játéktéren jól láthatóan, ezért erre vonatkozóan is beállítunk néhány tulajdonságot, pl. a pozícióját, színét.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Frogger</title>
  <style>
    #game_area {
      margin: 0 auto;
      border: 1px solid;
      position: relative;
      width: 450px;
      height: 400px;
      background-color: burlywood;
    }
    #wintext {
      font-size: 200px;
      font-weight: bolder;
      color: firebrick;
      top: 70px;
      left: 12px;
      position: absolute;
    }
```

```
  </style>
  <script src="jquery-3.4.1.min.js"></script>
  <script src="script.js"></script>
</head>
<body>
<div id="game_area"></div>
</body>
</html>
```

Amennyiben minden tulajdonságot hozzárendeltünk a játéktérhez, az alábbi kimenetet láthatjuk:



Hozzunk létre egy külön JS állományt, amibe a szkriptet fogjuk írni. Ezt adjuk hozzá a weboldalhoz és a jQuery függvénykönyvtárat is. A `script.js` állományban hozzunk létre néhány változót, melyeket használni fogunk. A `game_area` változóban fogjuk tárolni a játékterületet megtestesítő DIV-et. A békát tartalmazó képet is változóba mentjük majd, illetve létrehozunk változót az aktuális x, y pozícióknak is. A béka mozgásának mértékét 50 pixelben határozzuk meg és a játéktér méreteit is definiáljuk.

```
let game_area;

let frog;
let frog_x, frog_y;
let move_offset = 50;
let game_area_width = 450, game_area_height = 400;
```

Amint betöltődött a HTML oldal, inicializáljuk a játékteret. Ezt egy külön `init()` függvényben tesszük meg.

```
$(document).ready(function () {
  game_area = $('#game_area');
  init();
});
```

Tekintsük meg az `init()` függvény tartalmát. Itt hívjuk meg azt a két függvényt, mely a statikus pozícióval rendelkező utakat rajzolja ki és azt, amelyik a békát, amit majd mozgatni fogunk a játék során.

```
function init() {
  add_frog();
  add_road();
}
```

Először a békát helyezzük el a játéktéren. Erre már sok példát láthattunk korábban. Szükséges megadni, hogy hol szeretnénk elhelyezni és beállítani a `position` értékét is, majd végezetül hozzáadni a játéktérhez. Természetesen, ez bármelyik lépésben megtörténhet.

```
function add_frog() {
  frog = $('<img src="frog.png">');
  frog.css({
    top: game_area_height - move_offset,
    left: 200,
    position: 'absolute'
  });
  game_area.append(frog);
}
```

Adjuk hozzá az utakat is a játéktérhez. Ebben több is lesz a képen. Definiálunk egy kiindulási y pozíciót, melyet az `s_pos` változóban tárolunk és egy offset-et, hogy milyen távolságban legyenek egymástól az utak. Három utat fogunk kirajzolni és fontos, hogy minden esetben egy-egy kép példányunk legyen, hisz nem egyet akarunk ide-oda pakolászni. Ezeknek állítsuk be dinamikusan az y pozícióját és adjuk hozzá a játéktérhez is. Annak érdekében, hogy könnyebben azonosíthatóak legyenek az utak, a `road` + az aktuális út számát osztálycímkeként adjuk hozzá a létrehozott elemhez.

```
function add_road() {
  let s_pos = 50;
  let offset = 100;

  for (let i = 0; i < 3; i++) {
    let road = $('<div><img src="road.png"></div>');
    road.css({
      left: 0,
      top: s_pos + i * offset,
      position: 'absolute'
    });
    road.addClass('road' + i);
    game_area.append(road)
  }
}
```

Ezt követően az ablakhoz adjuk egy eseményfigyelőt annak érdekében, hogy mozgatni tudjuk a békát. A kurzormozgató billentyűk segítségével fogjuk tudni egyik pozícióról a másikra vinni a békát, ezért a `keydown` esemény lesz az, aminek a következtében meghívjuk a `move_frog` függvényt.

```
$(window).on('keydown', move_frog);
```

Számos példát láthattunk már arra, hogy miként tudunk megfigyelni egy billentyűeseményt és adott gombok lenyomásához utasítást rendelni. Tehát, először is szükséges lekérni a lenyomott billentyűt. Most figyelembe vesszük a fel-le nyilakat is. A béka mozgatásához lekérjük annak x, y pozícióját, melyet a `position()` függvényhívás segítségítségével teszünk meg. A mozgatáshoz a lekért pozícióhoz hozzáadjuk vagy épp kivonjuk a korábban definiált elmozgási értéket. Ne felejtsük el lekezelni a játékt

```
function move_frog(e) {
  let key = e.key;
  frog_x = frog.position().left;
  frog_y = frog.position().top;

  if (key === 'ArrowLeft') {
    if (frog_x - move_offset >= 0) {
      frog.animate({
        left: frog_x - move_offset
      }, 1);
    }
  } else if (key === 'ArrowRight') {
    if (frog_x + move_offset < game_area_width) {
      frog.animate({
        left: frog_x + move_offset
      }, 1);
    }
  } else if (key === 'ArrowUp') {
    if (frog_y - move_offset >= 0) {
      frog.animate({
        top: frog_y - move_offset
      }, 1);
    }
  } else if (key === 'ArrowDown') {
    if (frog_y + move_offset < game_area_height) {
      frog.animate({
        top: frog_y + move_offset
      }, 1);
    }
  }
}
```

A következő lépésben az autókat helyezzük el a játéktéren és adunk hozzájuk folytonos mozgást. Kétféle irányból érkezhetnek majd, ezért két orientáltságú autót tartalmazó képet találhatunk a megoldásban. Ezeknek hozzunk létre egy-egy változót. Adjuk hozzájuk a `car` class-t is.

```
let car_left, car_right;

car_left = $('<img src="car_left.png" class="car">');
car_right = $('<img src="car_right.png" class="car">');
```

Az autók folyamatos mozgatásához a `setInterval` időzítés használjuk ebben a megoldásban. 1000 ms-onként meghívjuk az `add_cars` függvényt, melyben az autók mozgását fogjuk definiálni. Az időzítés eseményét változóban eltároljuk, mert később fel fogjuk majd használni.

```
let car_birth;

car_birth = setInterval(add_cars, 1000);
```

Három utat adtunk hozzá a játéktérhez, de csak kétféle autó van. Az első és harmadik úton jobbra közlekedő, míg a középső szakaszon balra menő autókat fogunk elhelyezni. Ezeket úgy adjuk meg, hogy nagyobb valószínűséggel jelenik meg egy jobbra közlekedő autó, mint a másik. Ennek az az egyszerű oka, hogy azok két úton is megjelenhetnek. 65% eséllyel generálhatunk jobbra haladó autót és 35%-kal balra menőt.

Tekintsük először a jobbra menő esetet. Ahhoz, hogy több autót is meg tudjunk jeleníteni, szükséges, hogy azokból több példány is rendelkezésre álljon. Ezért a korábban megadott `car_right` változó tartalmát duplikáljuk a `clone()` függvény segítségével. 50% eséllyel első vagy a harmadik útra helyezzük-e el az új autót. Létrehozzuk erre a célra az `y_coord` változót. Ezután beállítjuk a megfelelő x és y pozíciókat és hozzáadjuk a játéktérhez az autót. Annak érdekében, hogy könnyebb legyen az ütközésdetektálás, hozzáadjuk az elemhez a `car_r` osztálycímkét is. A mozgatást az `amnimate()` függvény segítségével valósítjuk meg. Amint az autó balról jobbra elhaladt, levesszük azt a játéktérről a `remove()` hívással.

A balra haladó autók esete egyszerűbb, csak készíteni kell egy másolatot a korábban megadott autóról, pozícionálni és a mozgást hozzárendelni. Fontos, hogy itt nem a `car_r` hanem a `car_l` osztályt adjuk hozzá a HTML elemhez. Mivel a két autó szélessége nem azonos, így fontos lesz majd a későbbiekben, hogy milyen típusú autóval ütközhet a béka.

```
function add_cars() {
  if (Math.random() > 0.35) {
    let car = car_right.clone();
    let y_coord;

    if (Math.random() > 0.5) {
      y_coord = 250;
    } else {
      y_coord = 50;
    }

    car.css({
      position: 'absolute',
      left: 0,
      top: y_coord
    });
    game_area.append(car);
    car.addClass('car_r');

    car.animate({
      left: game_area_width - 74,
    }, 1000, function () {
      car.remove()
    });

  } else {
    let car = car_left.clone();
    let y_coord = 150;

    car.css({
      position: 'absolute',
      left: game_area_width - 56,
      top: y_coord
    });
    game_area.append(car);
    car.addClass('car_l');
```

```
37
38      car.animate({
39        left: 0
40      }, 1000, function () {
41        car.remove()
42      });
43    }
44  }
```

Most már mozgatható/mozognak a játék kulcselemei, tehát már csak az hiányzik, hogy detektáljuk az ütközést vagy azt, ha sikeresen átjut a béka az utakon. Ehhez létrehozzuk az alábbi két változót.

```
1  let coll_check;
2  let hit_thresh;
```

Az előzőleg deklarált `coll_check` változóhoz az ütközésdetektálást rendeljük, melyet 1 miliszekundumonként fogunk meghívni.

```
1  coll_check = setInterval(check_collision, 1);
```

A béka és az aktuálisan a játéktéren lévő autók távolságát fogjuk monitorozni. Ehhez a `car` osztállyal ellátott elemeket összegyűjtjük és egyenként megvizsgáljuk az aktuális pozíciójukat. Korábban nem csak a `car` class-t adtuk hozzá az autót tartalmazó elemekhez, hanem azt az osztályt is, amelyik azonosítja, hogy az egy balra vagy egy jobbra tartó autó-e. Ehhez lekérjük az aktuális elem nevét a `this.splitName` használatával. A `split()` függvény segítségével szét tudjuk választani a sztringeket egy elválasztó elem mentén és az eredményt tömbben kapjuk vissza. Ha megvizsgáljuk az előbb említett `this.splitName` eredményét, akkor az visszakapjuk az alábbit: `"car car_r"` vagy `"car car_l"`. Ahhoz, hogy megkapjuk az aktuális autó irányát jelző címkét, szétválasztjuk ezt a sztringet a " " mentén, így a következőt kapjuk: ["car", "car_r"] vagy ["car", "car_l"]. Ezt láthatjuk a 11. sorban.

Az ütközésdetektálás a korábban megismert módon fogjuk most is vizsgálni. Attól függően, hogy melyik irányból érkezik az autó, aszerint határozzuk meg, hogy mekkora lesz az ütközés meghatározásához megadott küszöbszám. Amennyiben a béka ütközik egy autóval, akkor újra meghívjuk az `init()` függvényt, ezáltal kezdőállapotba tesszük a békát. Azt is folyamatosan vizsgáljuk az ütközés mellett, hogy átjutott-e a béka az úton.

```
1  function check_collision() {
2
3    $('.car').each(function () {
4
5      let act_c_x = $(this).position().left;
6      let act_c_y = $(this).position().top;
7
8      let frog_x = frog.position().left;
9      let frog_y = frog.position().top;
10
11     let car_type = this.className.split(' ')[1];
12     if (car_type === 'car_r' && frog_y === act_c_y) {
13       hit_tresh = 74;
14     } else {
15       hit_tresh = 50;
16     }
17
18     if (distance({x: act_c_x, y: act_c_y}, {x: frog_x, y: frog_y}) <= hit_tresh) {
19       init();
20     }
21   });
22
23   win_check(frog_y);
24 }
```

Ahogyan említettem, a korábban már megismert, Euklideszi távolságot számítjuk ki két pont között.

```
1  function distance(a, b) {
2    let dx = a.x - b.x;
3    let dy = a.y - b.y;
4
5    return Math.sqrt(dx * dx + dy * dy)
6  }
```

A győzelem vizsgálatánál azt fogjuk megnézni, hogy a béka y pozíciója kisebb-e az általunk megadottnál. Amennyiben igen, akkor nem fogunk több autót létrehozni, illetve nem detektáljuk tovább az ütközést. Beállítunk egy új időzítést, mely 200 miliszekundomot vár és utána hajtódik végre.

```
1  function win_check(f_y) {
2    if (f_y <= 50) {
3      clearInterval(car_birth);
4      clearInterval(coll_check);
5      setTimeout(clearContents, 200);
6    }
7  }
```

Az `empty()` függvény segítségével az összes elemet eltávolítjuk abból a HTML elemből, melyre azt meghívjuk, jelen esetben a játéktérről. Azután pedig hozzáadjuk azt a DIV-et, ami a win szót fogja tartalmazni és a `#wintext` tulajdonságauit fogja felvenni.

```
1  function clearContents() {
2    game_area.empty();
3    game_area.append('<div id="wintext">WIN</div style>');
4  }
```