Tworzenie Aplikacji Internetowych

Laboratorium 8

Wykorzystanie bibliotek JavaScript – gra 3D na przykładzie biblioteki p5.js i klas JS

Celem zadania jest stworzenie gry 3d typu „Flappy bird". Zadanie polega na rozwinięciu prostego przykładu w którym zaimplementowano jedynie fizykę obiektu i system kolizji.

Zadanie 1

- Pobrać bibliotekę p5.js i uruchomić kod z drugiej i trzeciej strony tej instrukcji. Zapoznać się z kodem źródłowym (w szczególności z klasą Flappy, Pipe oraz funkcją rysującą draw().
- Zmodyfikować metodę proces klasy Pipe, tak aby zmniejszała pozycję x obiektu o wartość 2. Przetestować kod.
- Napisać metodę dla klasy Pipe o nazwie czyPoza(), która zwraca wartość 1 (lub true) jeżeli element Pipe znajduje się z lewej strony poza ekranem i wartość 0 (lub false) w innym wypadku. Element znajduje się poza ekranem jeżeli jego pozycja x jest mniejsza niż -WIDTH/2 – 120.
- W pętli rysującej **całą grafikę** (funkcja draw()) dodać na końcu warunek sprawdzający czy pipe1 **lub** pipe2 znajdują się poza ekranem, jeżeli tak, przypisać nowe obiekty do zmiennych pipe1 i pipe2 (o innych parametrach – konstruktor Pipe przyjmuje trzy parametry: a – początek rury, b – koniec rury, x – położenie rury – domyślnie WIDTH/2)
- Uaktualnić powyższy przykład tak aby początek pierwszej rury wynosił 0.0, a jej koniec wynosił wartość losową od 0.0 do 0.6, początek drugiej rury powinien wynosić koniec pierwszej + 0.4, a jej koniec 1.0.
- Dodać system punktów – w momencie gdy rury znikają liczba punktów powinna się zwiększać, w momencie kolizji liczba punktów powinna się zerować. Punkty wyświetlać w dowolnym miejscu na stronie lub przez komunikat alert w przypadku wystąpienia kolizji.

Zadanie 2 (dodatkowe)

- Zmodyfikować grę tak, aby wraz z upływem czasu zwiększała się prędkość przesuwania rur oraz zmniejszał pusty dystans pomiędzy nimi.

Zadanie 3 (dodatkowe)

- Dodać dwa dodatkowe obiekty pipe (pipe3 oraz pipe4), których położenie początkowe powinno zostać ustalone na wartość 0. Uwzględnić pipe3 oraz pip4 w całej logice gry (kolizje, pojawianie się nowych obiektów).

Zadanie 4 (dodatkowe)

- Zmodyfikować kod gry w taki sposób, aby wszystkie obiekty (flappy, pipe1, pipe2, pipe3 i pipe4) przechowywane były w tablicy. Dostosować resztę funkcjonalności kodu do korzystania z tej tablicy (np. wywoływać draw() oraz process() iterując tablicę).

Zadanie 5 (dodatkowe)

- Wczytać i wyświetlić wybrany przez siebie model 3D dla obiektu Flappy. Będzie to działało poprawnie dla obiektów umieszczonych bezpośrednio na stronach (podanie ścieżki do pliku 3d z dysku może nie działać). Rozwiązaniem może być uruchomienie serwera lokalnego (np. XAMMP) i uruchomienie strony przez http://localhost/ (należy wykonać to samodzielnie).

```
<!DOCTYPE html>
<html>
    <head>
        <title>FLAPPY</title>
        <script src="p5.min.js" defer></script>
    </head>
    <body>
        <main></main>
        <script>
            // stałe
            const WIDTH = 640                           // screen resx
            const HEIGHT = 480                          // screen resy
            const FLW = 30                              // flappy width
            const FLH = 20                              // flappy height
            const PPW = 30                              // pipe radious
            const ZSP = -100                            // z-bufor offset

            // flappy class
            class Flappy {
                // flappy constructor
                constructor() {
                    this.x = -WIDTH / 2                 // x position
                    this.y = -HEIGHT / 4                // y position
                    this.a = 0.0                        // acceleration
                    this.v = 0.0                        // velocity
                    this.j = 0.0                        // jump power
                }
                // uaktualnienie pozycji
                process() {
                    this.a = 0.10                       // acceleration
                    this.v += this.a - this.j           // velocity
                    this.y += this.v                    // position
                    if (this.v > 13.0) this.v = 13.0    // max down speed
                    if (this.v < -13.0) this.v = -13.0  // max up speed
                    if (this.j > 0.0) this.j -= 0.4     // jump power decrease
                }
                // ustawienie siły skoku
                jump() {
                    this.j = 3.0
                    this.v /= 2.0
                }
                // sprawdzenie kolizji
                collision(pipe) {
                    // brak kolizji po x
                    if (Math.abs(this.x - pipe.x) > (FLW + PPW)) {
                        return false
                    }
                    // kolizja po y
                    let pipes = 2 * pipe.y - pipe.h
                    let pipee = 2 * pipe.y + pipe.h
                    if ((this.y > pipes) && (this.y < pipee)) {
                        return true
                    }
                    // brak kolizji po y
                    return false
                }
                // narysowanie obiektu
                draw() {
                    push()
                    normalMaterial()
                    scale(1.0, 0.5, 1.0)
                    translate(this.x, this.y, ZSP)
                    rotateX(frameCount * 0.1)
                    cylinder(FLW, FLH)
                    pop()
                }
            }
```

```
            class Pipe {
                // pipe constructor
                constructor(start, end, x) {
                    this.x = x
                    this.h = (end - start) * HEIGHT * 1.5
                    this.y = -HEIGHT * 1.5 / 2 + start * HEIGHT * 1.5 + this.h / 2
                }
                // uaktualnienie pozycji
                process() {
                    // TODO
                }
                // narysowanie obiektu
                draw() {
                    push()
                    noStroke()
                    ambientMaterial(10, 230, 50)
                    translate(this.x, this.y, ZSP)
                    cylinder(PPW, this.h)
                    pop()
                }
            }

            // zmienne globalne (rysowane obiekty)
            let flappy = new Flappy()
            let pipe1 = new Pipe(0.00, 0.25, WIDTH / 2)
            let pipe2 = new Pipe(0.75, 1.00, WIDTH / 2)

            // p5.js setup
            function setup() {
                createCanvas(WIDTH, HEIGHT, WEBGL)
            }

            // funkcja rysująca
            function draw() {
                // tło i światła
                background(190, 210, 230)
                ambientLight(60, 60, 60)
                pointLight(255, 255, 255, 0, 0, 100)
                // sprawdzenie kolizji
                if (flappy.collision(pipe1) || flappy.collision(pipe2)) {
                    background(80, 0, 0)
                }
                // elementy gry
                flappy.process()
                flappy.draw()
                pipe1.process()
                pipe1.draw()
                pipe2.process()
                pipe2.draw()
                // TODO - sprawdzenie pozycji pipe i ewentualna zamiana na nowe
            }

            // key press event
            function keyPressed() {
                flappy.jump()
            }

            // mouse click event
            function mouseClicked() {
                flappy.jump()
            }
        </script>
    </body>
</html>
```