Документација за играта

Loose Ladybug

По предметот - Напреден Веб Дизајн

Професори: Изработилa:

Гоце Арменски Бојана Драгановска 183199

Бобан Јоксимоски

Скопје, 2023

Содржина

1. Вовед................................................................................................................3

2. Користени технологии....................................................................................5

3. Имплементација..............................................................................................5

4. Заклучок..........................................................................................................22

5. Референци.......................................................................................................23

\* Линк до проектот: <https://github.com/Bojana-Draganovska/Loose-LadyBug>

Вовед

Од најраниот почеток па се до ден денес има преголема лепеза на можности на интерактивни игри, кои луѓето можат да ги искористат за да се забавуваат или пак да научат нешто. Игрите датираат многу оддамна бидејќи една од најкорисните карактеристики на луѓето е креативноста. Првата видео игра е направена во 1958 година која многу наликува на култната игра „Pong“.

Поради големиот избор на игри низ целиот Интернет и нивната лесна достапност, дојдено е до можноста еден корисникот да има избор на игра по негова желба, категорија, тип на игра итн. Со правењето на забавни и интерактивни игри, на корисниците им се овозможува да уживаат и да се забавуваат низ целиот Интернет.

Веруваме дека скоро поголемиот дел од игрите се доста слични меѓу себе бидејќи имаат слична цел, на пример да одат кон одредена дестинација, или кој треба да е победникот доколку се натпреваруваат двајца. Меѓутоа и покрај сите сличности сепак игрите имаат разлики. Разликите се прво во нивниот дизајн, како се направени самите игри, можеме да кажеме дека имаат различни бои, различни карактери, различна позадинска слика, исто така нивните функционалности им се различни како на пример нивното движење, односно дали он ќе скока или ќе се движи лево или десно, позадиснки звуци и така натаму.

Играта LoseLadyBug претставува забавна и лесна игра која секој корисник би сакал да ја проба. Инспирацијата за оваа игра беше добро познатата игра „Flappy Bird“. Направена е по предметот „Напреден Веб Дизајн“ на Факултет за информатички науки и компјутерско инженерство (ФИНКИ), Скопје.

LoseLadyBug e дводеминзионална, интеркативна игра каде што целта на играта е играчот да освои што е можно повеќе поени односно да има што е можно поголем резултат (score). Играчот има целосна контрола врз бубамарата и може да ја движи со користење на копчето „Arrow Up“ или со копчето „Space“. Бубамарата се движи во една насока односно по Y оска. Доколку играчот успее да ја удри бубамарата или од горната или од долната цевка, автоматски ја губи партијата (game over). Резултатот се пресметува во зависност од тоа колку цевки има играчот поминато.

Можеме да кажеме дека на Интернет најчесто се сретннуваме со игри кои се дизајнирани дводимензионално. Поради тоа, нивните графики, анимации не се толку напредни за разлика од игрите кои се направени во три димензии. Кога станува збор за видео игри во 3D, знаеме дека игрите се доста детални бидејќи тука спаѓа моделирање на самиот карактер така што карактерот може да биде почист и пореалистичен, моделирање на околината, реалистични бои и слично.

Без разлика дали се работи за носталгијата на класичните аркадни наслови или креативноста на инди-програмерите кои ги поместуваат границите на дводимензионалните игри, 2D игрите нудат разновиден опсег на искуства. Тие ја прикажуваат трајната моќ на ангажираното раскажување приказни, иновативната механика и волшебните уметнички стилови без потреба од сложена 3D графика.

Користени технологии

LoseLadyBug e дводеминзионална игра која е имплементирана во Vanilla JavaScript, со HTML и CSS. Причината зашто ја одбрав оваа технологија е бидејќи JavaScript е доста флексибилен програмски јазик. Исто така знаеме дека HTML и CSS се основните работи за да ја направиме структурата на самата веб страница додека со помош на JavaScript ja „оживуваме“ истата.

JavaScript е еден од најпопуларните програмски јазици во светот кој се користи за разновидни апликации од веб страници до веб апликации за мобилни уреди, игри и слично. JavaScript континуирано напредува со нови функционалности и новитети.

Најголемата причина зошто е избрана оваа технологија за да се изработи играта е „заштита од иднината“, односно бидејќи јазикот е подржуван од секој прелистувач, голема е веројатноста дека JavaScript како технологија нема да умре никогаш, па со тоа ни дава доза безбедност.

Имплементација

Главниот јазик во кој е направена играта е Vanilla JavaScript, со HTML и CSS. Во овој дел од документацијата ќе ви објаснам за додавање слики на canvas, овозможување на бубамарата да се движи, генерирање на цевки и движење на истите низ екранот, детектирање на колизија и слично.

Структурата на HTML дрвото е целиот HTML документ, вклучувајќи ги таговите за глава (the head tag) и тело (the body tag).

Во <head> елементот само треба да напоменам дека се импортира фонт кој се користи низ целата игра, а тоа е фонтот VT323. Начинот на кој е импортиран фонтот е преку тагот <link> кој е HTML елемент и овој начин на импортирање на фонтови се вика „поврзување на фонтот“. Елементот <link> се користи за поврзување и други ресурси како што се CSS стилови, JavaScript датотеки и слично.

Содржината на телото (the body tag) е дел од HTML дрвото, но не е целото дрво. Делот на <body> елементот се нарекува содржина на <body> тагот. Тоа е дел од HTML документот што ја содржи видливата содржина на страницата, како што се текст, слики и видеа. Содржината на елементот е затворена со ознаките <body> и </body>.

Најбитниот дел од <body> таг е <canvas> елементот. Canvas елементот се користи за цртање графика преку JavaScript. Всушност <canvas> елементот е само „кутија“ за графиката. Мора да користиме JavaScript за да можеме да цртаме. Елементот има неколку методи за цртање патеки, квадрати, кругови, текст и додавање на слики. Му доделуваме id селектор бидејќи сакаме да го пристапиме во JavaScript документот.

Во Javascript документот кој е именуван script.js најпрво се дефинирани потребните променливи кои ќе ми бидат потребни за целата игра. Со цел да не ја претрупувам документацијата со дефинирање на променливи, истите ќе ги изоставам. За да го пристапиме <canvas> елементот го користам методот document.getElementById(). Методот document.getElementById() во JavaScript се користи за да се добие елемент според неговиот ID уникатен селектор. Методот document.getElementById() го зема ID на елементот како аргумент и го враќа објектот на елементот.

let canvas = document.getElementById("canvas");

let canvasWidth = 900;

let canvasHeight = 500;

canvas.height = canvasHeight;

canvas.width = canvasWidth;

Причината зошто canvasWidth и canvasHeight се со вредности 900 и 500 соодветно, е бидејќи таа е големината на главата слика која се користи како позадина на canvas елементот. Кога сме веќе кај позадината на canvas елементот, таа е импортирана преку css:

background-image: url('assets/images/day.png');

Следно што треба да разгледаме е променливата ctx.

let ctx = canvas.getContext("2d");

Методот HTMLCanvasElement.getContext() враќа објект кој ни овозможува цртање врз „платното“ (canvas). Контекстот за рендерирање е објект кој обезбедува методи за цртање форми, текст и слики на платно. Методот getContext() зема string како аргумент, кој го одредува типот на контекстот за рендерирање што треба да се добие. Најчестиот тип на контекст на рендерирање е 2d, кој се користи за 2D цртање.

Следно што треба да направиме е да специфицираме вредности за бубамарата со цел да ја позиционираме каде сакаме да биде нејзината почетна точка и колкава ќе биде бубамарата. За да можеме да и дадеме почетна точка потребни ни се 2 параметри односно X позицијата, Y позицијата, и за да дефинираме колкава ќе биде бубамарата потребни ни се 2 параметри односно ширината и висината на бубамарата.

let ladybugWidth = 34;

let ladybugHeight = 24;

let ladybugX = canvasWidth / 10;

let ladybugY = canvasHeight / 2;

Вредноста на променливата ladybugX ја добиваме така што ширината на „платното“ ја делиме со 10. Резултатот во овој случај е 90px. Истото го правиме и за вредноста на променливата ladybugY, такашто висината на „платното“ ја делиме со 2 со цел да можеме да ја позиционираме бубамарата на средина на платното и 90px одалечена од левата страна.

За да можеме полесно да манипулираме со параметрите кои ги создадовме за бубамарата правиме објект.

let ladybug = {

x: ladybugX,

y: ladybugY,

width: ladybugWidth,

height: ladybugHeight,

};

Бидејќи ги имаме сите потребни параметри за бубамарата, следно што треба да направиме е да ја исцртаме на „платното“.

Цртањето на карактерот во овој случај бубамарата, го правиме во window.onload event handler функцијата. window.onload е event handler во JavaScript што се користи за извршување на функција кога целата веб-страница ќе заврши со вчитување, вклучувајќи ја целата нејзина содржина (слики, листови со стилови, скрипти итн.). Обезбедува вашиот JavaScript код да работи само откако целата веб-страница е подготвена за интеракција. За таа цел цртањето на бубамарата и цртањето на сите останати потребни фигури за играта, ги правиме во овој дел.

ladybugImg = document.createElement('img');

ladybugImg.src = "assets/images/ladybug.png";

ladybugImg.onload = function () {

ctx.drawImage(

ladybugImg,

ladybug.x,

ladybug.y,

ladybug.width,

ladybug.height

);

};

Во JavaScript, document.createElement('img') е метод кој се користи за динамичко креирање на нов HTML елемент од наведениот тип, во овој случај, елемент <img>. Овој метод е особено корисен кога сакаме да креираме и манипулираме со HTML-елементи користејќи JavaScript, или да ги додадеме во DOM (Document Object Model) или да извршиме различни операции на нив.

ladybugImg = document.createElement('img');

Оваа линија код создава нов елемент <img> и го доделува на променливата ladybugImg. Во овој момент, елементот <img> е создаден, но сè уште не е дел од структурата на документот.

Потоа можеме да поставиме различни атрибути и својства на овој новосоздаден <img> елемент, како што се src, alt, ширина и висина, се со цел да го конфигурираме неговиот изглед и однесување.

ladybugImg.src = "assets/images/ladybug.png";

Бидејќи сеуште не е видлив нашиот карактер, ако сакаме да прикажеме слика или HTML содржина во <canvas>, ќе треба да ги икористиме дел од функционалностите кои ги нуди <canvas> елементот, за рачно да ја нацртаме содржината на „платното“.

Kористиме element.onload во овој случај елементот <img>, за да ги извршиме сите дејствата откако сликата целосно ќе се вчита.

На пример, можете да нацртате слика на платно користејќи го методот drawImage:

ladybugImg.onload = function () {

ctx.drawImage(

ladybugImg,

ladybug.x,

ladybug.y,

ladybug.width,

ladybug.height

);

Методот ctx.drawImage() во JavaScript е дел од HTML5 <canvas> API-то, конкретно контекстот на 2D рендерирање (често се нарекува ctx), и се користи за цртање слики на платно. Овој метод ни овозможува да поставиме слики на „платно“, што може да биде корисно за креирање графики, игри, манипулација со слики и многу повеќе.

Еве што претставува секој параметар:

ladybugImg: Ова е изворната слика што сакаме да ја нацртаме на платното.

ladybug.x: Ова е X-координатата на платното кoja jа одредува хоризонталната положба.

ladybug.y: Ова е Y-координатата на платното која ја одредува вертикалната положба.

ladybug.width: Овој параметар ја одредува ширината на нацртаната слика на платното. Можеме да го користиме за да ја зголемиме сликата. Ако ladybug.width е различна од ширината на изворната слика, сликата ќе биде соодветно растегната или компресирана.

ladybug.height: Овој параметар ја одредува висината на нацртаната слика на платното. Ако ladybug.height е различна од висината на изворната слика, сликата соодветно ќе се протега или компресира.

Истата постапка ја имаме и за цевките.

topPipeImg = document.createElement('img');

topPipeImg.src = "assets/images/topPipe.png";

bottomPipeImg = document.createElement('img');

bottomPipeImg.src = "assets/images/bottomPipe.png";

Следно нешто што треба да се рзгледа во функцијата window.onload е следново парче код:

function startGame() {

if (gameStarted == false) {

startBtn.addEventListener("click", function () {

startBtn.style.display = "none";

gameStarted = true;

document.addEventListener("keydown", moveLadybug);

setInterval(placePipes, 1500);

});

}

}

startGame();

Функцијата startGame() ни овозможува играта да не започне пред да се стисне копчето за старт. Од кога ќе се притисне копчето за старт, на истото му доделуваме “display: none”. Што тоа значи? Тоа значи дека ќога ќе го поставите својството за прикажување на елементот на ништо односно “display: none”, елементот целосно се отстранува од страницата и нема никакво влијание врз изгледот.

Ова исто така имплицира дека читачите на екранот, кои ги прават веб-страниците достапни за слепите, нема да можат да го користат елементот. Дали тоа значи дека елементот е целосно отстранет од DOM дрвото? - Факт е дека не го отстранува елементот од DOM дрвото. Елементот всушност не е само во дрвото за рендерирање.

На тој начин го “криеме” копчето за старт на играта.

document.addEventListener("keydown", moveLadybug);

Оваа линија на кодот му додава event listener на документот. Го слуша настанот „keydown“, кој се активира кога ќе се притисне копче на тастатурата. Кога ќе се случи овој настан, тој ја повикува функцијата наречена moveLadybug.

document. Ова го претставува целиот HTML документ, а event listener-от е прикачен на целиот документ, што значи дека ќе ги таргетира сите настани од тастатурата каде било на страницата.

"keydown": Ова е типот на настан што го слушаме, особено кога копче е притиснато надолу.

moveLadybug: Ова е функцијата што ќе се изврши кога ќе се случи настанот „keydown“. Тоа е функција што се справува со внесување на тастатурата за да контролира некој аспект од програмата, односно во оваа функција ни се крие логиката за движење на бубамарата.

setInterval(placePipes, 1500);

Оваа линија на код поставува тајмер за повторување користејќи ја функцијата setInterval(). Целта е да се изврши функцијата placePipes во редовни интервали.

placePipes: Ова е името на функцијата што ќе се повикува постојано за ажурирање на цевките која ќе ја разгледаме подетално подоцна во документацијата.

1500: Овој број го претставува интервалот во милисекунди. Во овој случај, placePipes ќе се повикува на секои 1500 милисекунди (или 1,5 секунди).

Најбитно, а и за крај последното парче код од нашата window.onload функција е requestAnimationFrame(update);

Методот window.requestAnimationFrame() му кажува на прелистувачот дека сакате да извршите анимација и бара прелистувачот да повика одредена функција за ажурирање на анимација непосредно пред следното пребојување. Методот зема повратен повик како аргумент што треба да се повика пред повторното сликање. Најчесто се користи за создавање мазни и ефикасни анимации или за извршување задачи што треба да се синхронизираат со процесот на рендерирање на прелистувачот.

Треба да го повикате овој метод секогаш кога сте подготвени да ја ажурирате вашата анимација на екранот. Ова ќе бара да се повика вашата функција за анимација пред прелистувачот да го изврши следното пребојување.

Параметри:

callback

Функција за повикување кога е време да ја ажурирате вашата анимација за следното повторно боење.

Во нашиот случај нашата callback функција е функцијата update().

function update() {

requestAnimationFrame(update);

if (gameOver) {

return;

}

ctx.clearRect(0, 0, canvas.width, canvas.height);

… }

Најглавно од се, функцијата update() ни служи за ажурирање на рамките ( frames ) на „платното“, односно ќе го прецртува „платното“ одново и одново. Секој пат кога ја ажурираме тековната рамка, сакаме да ја пребришеме претходната.

ctx.clearRect(0, 0, canvas.width, canvas.height);

Ова ќе ни го пребрише целиот canvas, почнувајќи од 0, 0 кој што го претставува горниот лев ќош од целото платно и опсегот ќе биде canvas.width, canvas.height. Ако не ја пребришеме претходната рамка, сите рамки ќе бидат една врз друга.

Следно што имаме во функцијата за ажурирање на нашиот canvas e цртањето на нашиот карактер:

if (gameStarted) {

movingY += gravity;

ladybug.y = Math.max(ladybug.y + movingY, 0);

ctx.drawImage(

ladybugImg,

ladybug.x,

ladybug.y,

ladybug.width,

ladybug.height

);

}

Ова го правиме тука бидејќи сакаме да ја имаме бубамарата во секој frame. Делот за движење на карактерот, ќе го разгледаме подоле.

Треба да поставиме неколку услови како крај за игра, како што ќе ги сретнуваме редоследно во кодот, така ќе ја објаснуваме нивната логика. Еден од условите за крај на играта може да го разгледаме сега:

if (ladybug.y > canvas.height) {

gameOver = true;

}

Бидејќи вредноста на ladybug.y кога се движи по У оската надоле се зголемува односно вредноста расте. Во нашиов случај canvas.height e 500px и ако играчот ја остави бубамарата да паѓа, евентуално вредноста на ladybug.у ќе биде поголема од висината на самото платно. Така што тоа е еден од излезите од самата игра, односно тоа означува губење на тековната партија.

Стигнавме до делот каде што треба конечно да ги исцртаме цевките во играта. Низ целата игра ќе имаме потреба да генерираме нови цевки па затоа имаме потреба од низа. На почетокот на кодот дефиниравме празна низа со името pipeArray која се исполнува бидејќи почнавме да ја повикуваме на секоја 1.5 секудна. Исто така на почетокот имаме дефинирано променлива со името movingX со почетна вредност -2 која ќе ни служи за движење на цевките хоризонтално. Како потсетник, вредностите на цевките ги сетиравме на почетокот на кодот.

for (let i = 0; i < pipeArray.length; i++) {

let pipe = pipeArray[i];

pipe.x += movingX;

ctx.drawImage(pipe.img, pipe.x, pipe.y, pipe.width, pipe.height);

if (!pipe.passed && ladybug.x > pipe.x + pipe.width) {

score += 0.5; // 0.5 because there are 2 pipes!

getScore.innerHTML = score;

pipe.passed = true;

if (score > 3) {

gameGreeting.style.backgroundColor = "#191e35";

gameGreeting.style.color = "#8d6641";

canvas.style.backgroundImage = "url('assets/images/night.png')";

gameResults.style.backgroundColor = "#191e35";

gameResults.style.color = "#8d6641";

}

}

if (collisonDetect(ladybug, pipe)) {

gameOver = true;

} }

Пред секоја, индивидуална цевка, ја ажурираме вредноста на pipe.x односно позицијата на цевката по Х оската.

pipe.x += movingX;

Со ова парче код, ќе ги движиме хоризонтално цевките за 2px, од десно кон лево.

Сеуште сме во for циклусот, па затоа следно нешто што ќе разгледаме е ова парче код.

if (!pipe.passed && ladybug.x > pipe.x + pipe.width) {

score += 0.5; // 0.5 because there are 2 pipes!

getScore.innerHTML = score;

pipe.passed = true;

if (score > 3) {

gameGreeting.style.backgroundColor = "#191e35";

gameGreeting.style.color = "#8d6641";

canvas.style.backgroundImage = "url('assets/images/night.png')";

gameResults.style.backgroundColor = "#191e35";

gameResults.style.color = "#8d6641";

}

}

Во кратки црти, овде е логиката за броење на колку цевки се изминати воедно и да следиме играчот колку цевки има изминато.

pipe.passed е својство на објектот pipe, и се користи за да се следи дали бубамарата ја поминала оваа конкретна цевка.

ladybug.x ја претставува хоризонталната положба на бубамарата.

pipe.x ја претставува хоризонталната положба на тековната цевка.

pipe.width е ширината на цевката.

Условот проверува дали:

pipe.passed е false (т.е. бубамарата не ја поминала оваа цевка).

Дали вредноста на ladybug.x е поголема од збирот на pipe.x и нејзината ширина pipe.width. Ако се исполнети двата услови, тоа значи дека бубамарата успешно ја поминала цевката.

score += 0.5; // 0.5 because there are 2 pipes!

Ако бубамарата ја помине цевката, резултатот се зголемува за 0,5. Овој коментар сугерира дека има две цевки, а резултатот се зголемува за 0,5 за секоја помината цевка. Значи, ако бубамарата ги помине двете цевки, резултатот ќе се зголеми за 1.

getScore.innerHTML = score;

Оваа линија ја ажурира содржината на HTML елементот со id getScore за да се прикаже ажурираниот резултат. Променливата score го задржува резултатот на играта и оваа линија го ажурира прикажаниот резултат на веб-страницата.

pipe.passed = true;

Откако бубамарата ќе ја помине цевката, својството pipe.passed се поставува на true, што покажува дека оваа конкретна цевка е помината. Ова спречува иста цевка да се брои повеќе пати.

Накратко, овој код проверува дали бубамарата поминала цевка, ја зголемува оценката, го ажурира прикажаниот резултат на веб-страницата и ја означува цевката како помината за да избегне повторно броење. Вообичаен механизам во игрите е да се следи напредокот на играчот и соодветно да се ажурира резултатот.

if (score > 3) {

gameGreeting.style.backgroundColor = "#191e35";

gameGreeting.style.color = "#8d6641";

canvas.style.backgroundImage = "url('assets/images/night.png')";

gameResults.style.backgroundColor = "#191e35";

gameResults.style.color = "#8d6641";

}

}

Оваа парче код проверува дали променливата score има вредност поголема од 3. Ако резултатот е навистина поголем од 3, се извршува кодниот блок во кадравите загради {}. Оваа условна изјава се користи за активирање на одредени дејства или промени во стилот кога резултатот ќе достигне одреден праг.

gameGreeting.style.backgroundColor = "#191e35";

Ако резултатот е поголем од 3, оваа линија ја поставува бојата на позадината на елементот HTML со id gameGreeting на „#191e35“. Ова го менува визуелниот изглед на елементот, односно позадината ја прави потемна.

gameGreeting.style.color = "#8d6641";

Слично, оваа линија ја поставува бојата на текстот на истиот елемент на „#8d6641“. Ја менува бојата на текстот во елементот во конкретната боја.

canvas.style.backgroundImage = "url('assets/images/night.png')";

Оваа линија ја поставува задната слика на HTML <canvas> елемент на URL. Ја менува позадината на платното во слика лоцирана на „assets/images/night.png“. Ова може да се користи за префрлање на визуелната тема на платното, како на пример од дневна на ноќна сцена.

Слично како и за елементот „gameGreeting“ така и за елементот „gameResults“ ги сменивме бојата на текстот и бојата на позадината.

Накратко, овој код ги прилагодува стиловите на одредени HTML-елементи врз основа на вредноста на променливата за резултат. Кога резултатот е поголем од 3, применува одредени бои на позадината и бои на текстот на елементите со одренедени ID, дури пак и ја менуваме позадината на платното. Ова често се прави за да се обезбеди визуелна повратна информација или да се промени изгледот на играта како што напредува резултатот на играчот.

if (collisonDetect(ladybug, pipe)) {

gameOver = true;

}

На крај на for циклусот имаме повик на функција која ќе ја разгледаме подоле.

Пред да излеземе од функцијата update() имаме уште еден дел да разгледаме, а тоа е делот за ресетирање на играта со цел повторно да се игра.

if (gameOver) {

restartBtn.style.display = "block";

gameGreeting.textContent = "Game Over!";

restartBtn.addEventListener("click", function () {

restartBtn.style.display = "none";

pipeArray = [];

gameGreeting.textContent = "Good Luck!";

ladybug.y = ladybugY;

getScore.innerHTML = 0;

score = 0;

gameOver = false;

gameGreeting.style.backgroundColor = "#ff356f";

gameGreeting.style.color = "#60284d";

canvas.style.backgroundImage = "url('assets/images/day.png')";

gameResults.style.backgroundColor = "#ff356f";

gameResults.style.color = "#60284d";

});

}

Поената на овој дел од кодот е кога играчот ќе ја изгуби партијата на некоја од можните начини, сите променливи што ги користиме ги сетираме на 0 односно ги пребришуваме се со цел да може одново да се игра играта,

Погоре во документацијата се сретнавме со една функција која нагласивме дека подоцна ќе ја разгледаме. Таа е функцијата за полнење на низата pipeArray. Името на функцијата е placePipes().

function placePipes() {

if (gameOver) {

return;

}

let randomPipeY = pipeY - pipeHeigt / 4 - Math.random() \* (pipeHeigt / 2);

let opening = canvas.height / 3;

…}

Првиот услов е краток и јасен, односно ако променливата gameOver e true, нема да се изврши ништо од наведеното.

let randomPipeY = pipeY - pipeHeigt / 4 - Math.random() \* (pipeHeigt / 2);

Оваа линија на код пресметува случајна вертикална положба (randomPipeX) за цевка. Еве што прави секој дел од пресметката:

pipeY: Jа претставува основната вертикална положба на цевката. Ова е местото каде што дното на цевката ќе биде првично поставено.

pipeHeigt / 4: Овој израз ја дели висината на цевката (pipeHeight) со 4. Jа поместува цевката нагоре за една четвртина од нејзината висина.

Math.random()\*(pipeHeigt / 2): Овој дел генерира случаен број помеѓу 0 и pipeHeight / 2. Определува колку позицијата на цевката може да варира вертикално.

Целокупната пресметка ги одзема овие вредности од pipeY, што значи дека ја поместува почетната положба на цевката (pipeY) нагоре за одредена сума, создавајќи случајна вертикална положба за цевката. Оваа случајност е корисна за создавање варијации во висините на цевките во играта.

let opening = canvas.height / 3;

Оваа линија на код пресметува променлива наречена opening, која ја претставува вертикалната положба на отворот помеѓу две цевки. Еве што прави:

canvas.height: Ова ја претставува висината на платното на кое се црта играта.

/ 3: Ја дели висината на платното за 3. Oваа поделба го дели платното на три приближно еднакви вертикални делови.

Значи, отворот е поставен на една третина од висината на платното, што сугерира дека оваа вредност ја претставува вертикалната положба каде што треба да се наоѓа отворот или јазот помеѓу цевките. Со тоа го одредуваме просторот помеѓу цевките, каде што играчот треба да се движи.

let topPipe = {

img: topPipeImg,

x: pipeX,

y: randomPipeY,

width: pipeWidth,

height: pipeHeigt,

passed: false,

};

pipeArray.push(topPipe);

let bottomPipe = {

img: bottomPipeImg,

x: pipeX,

y: randomPipeY + pipeHeigt + opening,

width: pipeWidth,

height: pipeHeigt,

passed: false,

};

pipeArray.push(bottomPipe);

Во останатиот дел од функцијата дефинираме два објекти со името topPipe и bottomPipe.

Сега ќе ја разгледаме функцијата за движење на нашиот карактер, што ќе се изврши кога ќе се случи настанот „keydown“.

function moveLadybug(e) {

if (e.code == "Space" || e.code == "ArrowUp") {

// jump

movingY = -6;

}

}

function moveLadybug(e): Оваа линија дефинира JavaScript функција наречена moveLadybug која зема настан како свој параметар. Функцијата е дизајнирана да биде event handler, што значи дека ќе биде повикана како одговор на интеракцијата на корисникот со страницата, односно на притискање на копче.

if (e.code == "Space" || e.code == "ArrowUp"): Во рамките на функцијата, постои изјава if која ја проверува карактеристиката на кодот на објектот на настанот e. Својството на кодот го претставува клучниот код на клучот што го активирал настанот.

e.code == "Space": Овој услов проверува дали притиснатото кошче е еднакво на копчето „Space“. Оваа состојба е точно кога корисникот ќе го притисне копчето Spacebar на тастатурата.

e.code == "ArrowUp": Оваа состојба проверува дали притиснатото кошче е еднакво на копчето „ArrowUp“. Вистина е кога корисникот ќе го притисне копчето со стрелка нагоре на тастатурата.

movingY = -6: Ако се исполнети условите во наредбата if (т.е. се притисне spacebar или копчето со стрелка нагоре), оваа линија на код ја менува променливата со име moveY. Го поставува moveY на -6. Ова претставува промена во вертикалната положба на бубамарата, предизвикувајќи таа да се движи нагоре. Вредноста -6 го претставува бројот на пиксели што карактерот ги движи нагоре при секој скок.

Накратко, оваа функција moveLadybug е дизајнирана да се справи со вертикалното движење на бубамарата. Кога корисникот ќе го притисне копчето Spacebar или копчето со стрелка нагоре, ја поставува променливата moveY на -6, иницирајќи вертикален скок за бубамарата.

Последната функција на целиот код, е функцијата за детекција на колизии.

function collisonDetect(ladybug, pipe) {

return (

ladybug.x < pipe.x + pipe.width &&

ladybug.x + ladybug.width > pipe.x &&

ladybug.y < pipe.y + pipe.height &&

ladybug.y + ladybug.height > pipe.y

);

}

Кодот дефинира JavaScript функција наречена collisionDetect која се користи за проверка на судири помеѓу два објекти, особено „бубамарата“ и „цевката“. Ајде да разложиме што прави овој код:

function collisonDetect(ladybug, pipe): Оваа линија претставува функција во JavaScript со име collisionDetect која зема два параметри, бубамара и цевка. Овие параметри ги претставуваат предметите што се проверуваат за судир.

Функцијата пресметува и враќа булова вредност (точно или неточно) врз основа на следните услови:

ladybug.x < pipe.x + pipe.width: Оваа состојба проверува дали левиот раб на бубамарата е лево од десниот раб на цевката. Со други зборови, проверува дали левата страна на бубамарата е лево од десната страна на цевката.

ladybug.x + ladybug.width > pipe.x: Оваа состојба проверува дали десниот раб на бубамарата е десно од левиот раб на цевката. Проверува дали десната страна на бубамарата е десно од левата страна на цевката.

ladybug.y < pipe.y + pipe.height: Оваа состојба проверува дали горниот раб на бубамарата е над долниот раб на цевката. Проверува дали горната страна на бубамарата е над долната страна на цевката.

ladybug.y + ladybug.height > pipe.y: Оваа состојба проверува дали долниот раб на бубамарата е под горниот раб на цевката. Проверува дали долната страна на бубамарата е под горната страна на цевката.

Ако сите овие услови се исполнети истовремено, тоа значи дека кутијата за ограничување на бубамарата и кутијата за ограничување на цевката се вкрстуваат и дека е откриен судир. Во овој случај, функцијата враќа true. Во спротивно, доколку некој од условите не е исполнет, се враќа false, што укажува дека нема судир.

Оваа функција вообичаено се користи во развојот на игри и интерактивни апликации за да се утврди дали два објекти, како што се ликови од играта и пречки, се судираат, што овозможува да се појават дејства или настани кога се случуваат судири.

Заклучок

Моментално играта Loose Ladybug ќе може да се игра само на десктоп уреди, но верувам дека во иднина со мало подобрување на играта ќе овозможам играчите да ја играат играта и на нивните мобилни телефони и таблети. Функционалности што ги објаснив детално за играта, можеби ќе им помогне на доста други луѓе кои што сакаат да направат слична или некоја различна игра така што ќе можат да го превземат кодот односно дел од кодот кој им е потребен за да можат да си ги мплементираат функционалностите за нивната игра и да им се олесни целиот процес.

Дводимензионалните (2D) игри продолжуваат да заземаат посебно место во светот на забавата. Додека модерната технологија воведе нова ера на извонредни 3D искуства, 2D игрите остануваат сакани од играчите од сите возрасти. Нивната едноставност во дизајнот, пристапноста и безвременската привлечност ги прават вреден дел од индустријата за игри. Тие продолжуваат да напредуваат во гејмерски пејзаж во кој доминира 3D технологијата, докажувајќи дека добро изработеното 2D искуство може да биде исто толку привлечно и пријатно како и секој наслов на 3D блокбастер.

Референци

1. <https://developer.mozilla.org/en-US/>
2. <https://www.w3schools.com/js/default.asp>
3. <https://stackoverflow.com/>