**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи № 6 з дисципліни

«Проектування алгоритмів»

„**Пошук в умовах протидії, ігри з повною інформацією, ігри з елементом випадковості, ігри з неповною інформацією**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-12 Боборикін Артем*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.Н.*

Київ 2022

Зміст

[1 Мета лабораторної роботи 3](#_Toc86770239)

[2 Завдання 4](#_Toc86770240)

[3 Виконання 6](#_Toc86770241)

[3.1 Програмна реалізація алгоритму 6](#_Toc86770242)

[3.1.1 Вихідний код 6](#_Toc86770243)

[3.1.2 Приклади роботи 6](#_Toc86770244)

[Висновок 7](#_Toc86770245)

[Критерії оцінювання 8](#_Toc86770246)

# Мета лабораторної роботи

Мета роботи - вивчити основні підходи до формалізації алгоритмів знаходження рішень задач в умовах протидії. Ознайомитися з підходами до програмування алгоритмів штучного інтелекту в іграх з повною інформацією, іграх з елементами випадковості та в іграх з неповною інформацією.

# Завдання

Для ігор з повної інформацією, згідно варіанту (таблиця 2.1) реалізувати візуальний ігровий додаток для гри користувача з комп'ютерним опонентом. Для реалізації стратегії гри комп'ютерного опонента використовувати алгоритм альфа-бета-відсікань. Реалізувати три рівні складності (легкий, середній, складний).

Для ігор з елементами випадковості, згідно варіанту (таблиця 2.1) реалізувати візуальний ігровий додаток, з користувацьким інтерфейсом, не консольним, для гри користувача з комп'ютерним опонентом. Для реалізації стратегії гри комп'ютерного опонента використовувати алгоритм мінімакс.

Для карткових ігор, згідно варіанту (таблиця 2.1), реалізувати візуальний ігровий додаток, з користувацьким інтерфейсом, не консольним, для гри користувача з комп'ютерним опонентом. Потрібно реалізувати стратегію комп'ютерного опонента, і звести гру до гри з повною інформацією (див. Лекцію), далі реалізувати стратегію гри комп'ютерного опонента за допомогою алгоритму мінімаксу або альфа-бета-відсікань.

Реалізувати анімацію процесу жеребкування (+1 бал) або реалізувати анімацію ігрових процесів (роздачі карт, анімацію ходів тощо) (+1 бал).

Реалізувати варто тільки одне з бонусних завдань.

Зробити узагальнений висновок лабораторної роботи.

Таблиця 2.1 – Варіанти

|  |  |  |
| --- | --- | --- |
| **№** | **Варіант** | **Тип гри** |
| 5 | Тринадцять http://www.rules.net.ru/kost.php?id=16 | З елементами випадковості |

# Виконання

## Програмна реалізація алгоритму

### Вихідний код

**main.js –** Головний файл, що створює Desktop App

// main.js  
  
// Modules to control application life and create native browser window  
const { ***app***, BrowserWindow } = ***require***('electron')  
const path = ***require***('path')  
  
const createWindow = () => {  
 // Create the browser window.  
  
 const mainWindow = new BrowserWindow({  
 width: 500,  
 height: 500,  
 fullscreen: true,  
 })  
  
 // and load the index.html of the app.  
 mainWindow.loadFile('index.html')  
  
 // Open the DevTools.  
 mainWindow.webContents.openDevTools()  
}  
  
// This method will be called when Electron has finished  
// initialization and is ready to create browser windows.  
// Некоторые интерфейсы API могут использоваться только после возникновения этого события.  
***app***.whenReady().then(() => {  
 createWindow()  
  
 ***app***.on('activate', () => {  
 // On macOS it's common to re-create a window in the app when the  
 // dock icon is clicked and there are no other windows open.  
 if (BrowserWindow.*getAllWindows*().length === 0) createWindow()  
 })  
})  
  
// Quit when all windows are closed, except on macOS. There, it's common  
// for applications and their menu bar to stay active until the user quits  
// explicitly with Cmd + Q.  
***app***.on('window-all-closed', () => {  
 if (***process***.platform !== 'darwin') ***app***.quit()  
})  
  
// In this file you can include the rest of your app's specific main process  
// code. Можно также поместить их в отдельные файлы и применить к ним require.

**js/intro.js** – файл, що містить у собі анімацію запуску

function animation({timing, draw, duration}) {  
  
 let start = ***performance***.now();  
  
 requestAnimationFrame(function animate(time) {  
 // timeFraction изменяется от 0 до 1  
 let timeFraction = (time - start) / duration;  
 if (timeFraction > 1) timeFraction = 1;  
  
 // вычисление текущего состояния анимации  
 let progress = timing(timeFraction);  
  
 draw(progress); // отрисовать её  
  
 if (timeFraction < 1) {  
 requestAnimationFrame(animate);  
 }  
  
 });  
}  
function makeEaseOut(timing) {  
 return function(timeFraction) {  
 return 1 - timing(1 - timeFraction);  
 }  
}  
function quad(timeFraction) {  
 return ***Math***.pow(timeFraction, 2)  
}  
let ***circOut*** = makeEaseOut(quad)  
  
async function main(){  
 await sleep(1000)  
 ***document***.querySelector('.intro').style.opacity = '1'  
 await sleep(1000)  
  
 ***document***.querySelector('.play-table').style.opacity = '1'  
 await sleep(1000)  
  
 animation({  
 duration: 800,  
 timing: quad,  
 draw(progress){  
 ***document***.querySelector('.middle13').style.width = progress \* 100 + '%'  
 }  
 })  
 await sleep(800)  
 ***document***.querySelector('.intro').style.opacity='0.5'  
  
 await sleep(400)  
 ***document***.querySelector('.play-table \*').style.zIndex = '1'  
}  
main()  
  
function sleep (ms){  
 return new Promise((res, rej)=>{  
 setTimeout(res, ms)  
 })  
}

**js/opponent.js –** Файл, що визначає поведінку бота

class Node {  
 constructor(points, finPoints, step, parent = null, current\_rand= null) {  
  
 this.parent = parent  
 this.finPoints = finPoints  
 this.points = points  
 this.step = step  
  
 this.arr = []  
 if(current\_rand){  
 for(let i=0; i<11; i++){  
 if(current\_rand-3===i){  
 this.arr.push({  
 add: null,  
 subtract: null,  
 x2: null,  
 x3: null,  
 x4: null,  
 d2: null,  
 d3: null,  
 d4: null  
 })  
 }  
 else{  
 this.arr.push(null)  
 }  
 }  
 }  
 else {  
 for(let i=0; i<11; i++){  
 this.arr.push({  
 add: null,  
 subtract: null,  
 x2: null,  
 x3: null,  
 x4: null,  
 d2: null,  
 d3: null,  
 d4: null  
 })  
 }  
 }  
  
 }  
}  
  
class Tree{  
 constructor(points, finPoints, step, parent = null, current\_rand = null, last\_step=3) {  
 this.root = new Node(points, finPoints, step, parent, current\_rand)  
 this.last\_step = last\_step  
 }  
  
 init(node){  
 for(let i=0; i<11; i++){  
 if(node.arr[i]===null){  
 continue  
 }  
  
 let points = node.points+i+3  
 node.arr[i].add = new Node (points, points%13===0?node.finPoints+points/13:node.finPoints, node.step+1, node)  
  
 points = node.points-(i+3)  
 if(points>=0){  
 node.arr[i].subtract = new Node (points, points%13===0?node.finPoints+points/13:node.finPoints, node.step+1, node)  
 }  
 else{  
 node.arr[i].subtract = null  
 }  
  
 points = node.points + i\*2  
 node.arr[i].x2 = new Node (points, points%13===0?node.finPoints+points/13:node.finPoints, node.step+1, node)  
  
 points = node.points + i\*3  
 node.arr[i].x3 = new Node (points, points%13===0?node.finPoints+points/13:node.finPoints, node.step+1, node)  
  
 points = node.points + i\*4  
 node.arr[i].x4 = new Node (points, points%13===0?node.finPoints+points/13:node.finPoints, node.step+1, node)  
  
 points = node.points + ***Math***.round(i/2)  
 node.arr[i].d2 = new Node (points, points%13===0?node.finPoints+points/13:node.finPoints, node.step+1, node)  
  
 points = node.points + ***Math***.round(i/3)  
 node.arr[i].d3 = new Node (points, points%13===0?node.finPoints+points/13:node.finPoints, node.step+1, node)  
  
 points = node.points + ***Math***.round(i/4)  
 node.arr[i].d4 = new Node (points, points%13===0?node.finPoints+points/13:node.finPoints, node.step+1, node)  
 }  
 }  
  
 full\_init(node){  
  
 this.init(node)  
  
 if(node.step === this.last\_step){  
 return  
 }  
  
 for(let one of node.arr){  
 if(one === null){  
 continue  
 }  
 for(let two in one){  
 if(one[two]!==null){  
 this.full\_init( one[two] )  
 }  
 }  
 }  
 }  
  
 count(node){  
 if(node === null){  
 return 0  
 }  
 let counter = node.finPoints  
  
 for(let one of node.arr){  
 if(one !== null){  
 for(let two in one){  
 if(one[two] !== null){  
 counter += this.count(one[two])  
 }  
 }  
 }  
 }  
 return counter  
 }  
  
 moveFull(node, callback) {  
 if (node != null) {  
 callback(node)  
 return this.moveFull(node.left, callback) || this.moveFull(node.top, callback) || this.moveFull(node.right, callback) || this.moveFull(node.bottom, callback);  
 }  
 }  
}  
  
function find\_the\_best\_way(points, finP, current\_rand, last\_step=3){  
 let tree = new Tree(points, finP, 1, null, current\_rand, last\_step)  
 tree.full\_init(tree.root)  
  
 let obj = {}  
  
 for(let one in tree.root.arr[current\_rand-3]){  
 obj[one] = tree.count(tree.root.arr[current\_rand-3][one])  
 }  
  
 let fin = 'add'  
 let fin\_value = 0  
 ***console***.log(obj)  
 for(let one in obj){  
 if(obj[one]>fin\_value){  
 fin = one  
 fin\_value = obj[one]  
 }  
 }  
 ***console***.log(fin)  
  
 return fin  
}  
  
function count\_next\_step(prev, rand){  
 let obj = {  
 add: prev + rand,  
 subtract: (prev - rand)>=0?prev-rand:null,  
 x2: prev + rand\*2,  
 x3: prev + rand\*3,  
 x4: prev + rand\*4,  
 d2: prev + ***Math***.round(rand/2),  
 d3: prev + ***Math***.round(rand/3),  
 d4: prev + ***Math***.round(rand/4),  
 }  
  
 let fin = null  
 let finV = 0  
  
 for(let one in obj){  
 if(obj[one]%13===0&&finV<obj[one]/13){  
 fin = one  
 finV = obj[one]/13  
 }  
 }  
 return fin  
  
}

**js/main.js –** Файл, що надає функціональність кнопкам та містить логіку самої програми

let ***active*** = false  
let ***switcher*** = 'player'  
  
let ***rounds*** = 0  
  
***document***.querySelector('.restart-button').onclick = ()=>{  
 ***document***.querySelector('.dice').style.display = 'none'  
 ***document***.querySelector('#opponentPoints').innerHTML = '0'  
 ***document***.querySelector('#playerPoints').innerHTML = '0'  
 ***document***.querySelector('#OP\_Full').innerHTML = '0'  
 ***document***.querySelector('#PP\_Full').innerHTML = '0'  
  
 ***active*** = false  
 ***rounds*** = 0  
}  
  
***document***.querySelector('#playButton').onclick = ()=>{  
  
 if(***active***){  
 return false  
 }  
 ***active*** = true  
  
 ***document***.querySelector('.dice').style.display = 'flex'  
  
 let first\_num = throw\_dice(1)  
 let second\_num = throw\_dice(2)  
 second\_num++  
 let full\_num = first\_num + second\_num  
  
 setTimeout(()=>{  
 ***document***.querySelector('.menu').addEventListener('click', menu\_click)  
 }, 5000)  
  
 let menu\_click = (e)=>{  
 let tg = e.target.innerHTML  
 ***console***.log(tg)  
  
 let PointObj = ***document***.querySelector('#playerPoints')  
 let point = +PointObj.innerHTML  
  
 switch (tg) {  
 case 'ADD': PointObj.innerHTML = '' + (point + full\_num); break;  
 case 'SUBTRACT': {  
 if(full\_num>point){  
 alert('Unable')  
 return  
 }  
 PointObj.innerHTML = '' + (point - full\_num);  
 break;  
 }  
 case 'x2': PointObj.innerHTML = point + full\_num\*2 + ''; break;  
 case 'x3': PointObj.innerHTML = point + full\_num\*3 + ''; break;  
 case 'x4': PointObj.innerHTML = point + full\_num\*4 + ''; break;  
 case '/2': PointObj.innerHTML = ***Math***.round(point + full\_num/2) + ''; break;  
 case '/3': PointObj.innerHTML = ***Math***.round(point + full\_num/3) + ''; break;  
 case '/4': PointObj.innerHTML = ***Math***.round(point + full\_num/4) + ''; break;  
 }  
 let poi = +***document***.querySelector('#playerPoints').innerHTML  
 if(poi%13===0){  
 ***document***.querySelector('#PP\_Full').innerHTML = +***document***.querySelector('#PP\_Full').innerHTML + poi/13 + ''  
 }  
  
 if(+***document***.querySelector('#PP\_Full').innerHTML >= 10){  
 alert('YOU win!)')  
 ***active*** = true  
 return;  
 }  
  
 if(***rounds***===4){  
 ***document***.querySelector('#turn').innerHTML = ***switcher***!=='player'?'Your turn':"Opponent's turn"  
 ***rounds*** = 0  
 opponentStep()  
 }  
 else {  
 ***active*** = false  
 ***rounds***++  
 }  
 ***document***.querySelector('.menu').removeEventListener('click', menu\_click)  
 }  
}  
  
async function opponentStep(){  
 let points = +***document***.querySelector('#opponentPoints').innerHTML  
 let pointsF = +***document***.querySelector('#OP\_Full').innerHTML  
  
 await sleep(1000)  
  
 for(let i=0; i<5; i++){  
 let first\_num = throw\_dice(1)  
 let second\_num = throw\_dice(2)  
 second\_num++  
 let dice\_num = first\_num + second\_num  
  
 let step  
 switch (i) {  
 case 0: step=3; break;  
 case 1: step=3; break;  
 case 2: step=3; break;  
 case 3: step=2; break;  
 case 4: step=1; break;  
 }  
  
 await sleep(6000)  
  
 let button\_name = count\_next\_step(points, dice\_num, step)  
 if(!button\_name){  
 button\_name = find\_the\_best\_way(points, pointsF, dice\_num)  
 }  
  
 switch (button\_name){  
 case "add": {  
 points+=dice\_num  
 points%13===0?pointsF+=points/13:null  
 break  
 }  
 case "subtract": {  
 points-=dice\_num  
 points%13===0?pointsF+=points/13:null  
 break  
 }  
 case "x2": {  
 points+=dice\_num\*2  
 points%13===0?pointsF+=points/13:null  
 break  
 }  
 case "x3": {  
 points+=dice\_num\*3  
 points%13===0?pointsF+=points/13:null  
 break  
 }  
 case "x4": {  
 points+=dice\_num\*4  
 points%13===0?pointsF+=points/13:null  
 break  
 }  
 case "d2": {  
 points+=***Math***.round(dice\_num/2)  
 points%13===0?pointsF+=points/13:null  
 break  
 }  
 case "d3": {  
 points+=***Math***.round(dice\_num/3)  
 points%13===0?pointsF+=points/13:null  
 break  
 }  
 case "d4": {  
 points+=***Math***.round(dice\_num/4)  
 points%13===0?pointsF+=points/13:null  
 break  
 }  
 }  
  
 ***document***.querySelector('#'+'b-'+button\_name).classList.add('hover')  
 ***document***.querySelector('#opponentPoints').innerHTML = points + ''  
 ***document***.querySelector('#OP\_Full').innerHTML = pointsF + ''  
  
 await sleep(1000)  
 ***document***.querySelector('#'+'b-'+button\_name).classList.remove('hover')  
  
 if(pointsF >= 10){  
 alert('Opponent win!(')  
 ***active*** = true  
 return  
 }  
 }  
  
 ***document***.querySelector('#turn').innerHTML = 'Your turn'  
 ***active*** = false  
  
}  
  
// active=false

**js/script.js –** Відповідає за 3d графіку кубиків

const ***scene*** = new THREE.Scene();  
const ***camera*** = new THREE.PerspectiveCamera( 75, ***window***.innerWidth / ***window***.innerHeight, 0.1, 1000 );  
  
const ***renderer*** = new THREE.WebGLRenderer();  
***renderer***.setClearColor( 0xffffff, 0);  
***renderer***.setSize( ***window***.innerWidth, ***window***.innerHeight );  
***document***.querySelector('.dice').appendChild( ***renderer***.domElement );  
  
const ***directionalLight*** = new THREE.DirectionalLight( 0xffffff);  
***directionalLight***.position.set(0, 0, 10)  
***scene***.add( ***directionalLight*** );  
  
const ***textureLoader*** = new THREE.TextureLoader();  
let ***textureCube*** = [  
 new THREE.MeshStandardMaterial({  
 map: ***textureLoader***.load("./imgs/dice/dice-six-faces-one.png")  
 }),  
 new THREE.MeshStandardMaterial({  
 map: ***textureLoader***.load("./imgs/dice/dice-six-faces-two.png")  
 }),  
 new THREE.MeshStandardMaterial({  
 map: ***textureLoader***.load("./imgs/dice/dice-six-faces-three.png")  
 }),  
 new THREE.MeshStandardMaterial({  
 map: ***textureLoader***.load("./imgs/dice/dice-six-faces-four.png")  
 }),  
 new THREE.MeshStandardMaterial({  
 map: ***textureLoader***.load("./imgs/dice/dice-six-faces-five.png")  
 }),  
 new THREE.MeshStandardMaterial({  
 map: ***textureLoader***.load("./imgs/dice/dice-six-faces-six.png")  
 }),  
]  
let ***textureCube2*** = [  
 new THREE.MeshStandardMaterial({  
 map: ***textureLoader***.load("./imgs/dice/dice-six-faces-two.png")  
 }),  
 new THREE.MeshStandardMaterial({  
 map: ***textureLoader***.load("./imgs/dice/dice-six-faces-three.png")  
 }),  
 new THREE.MeshStandardMaterial({  
 map: ***textureLoader***.load("./imgs/dice/dice-six-faces-four.png")  
 }),  
 new THREE.MeshStandardMaterial({  
 map: ***textureLoader***.load("./imgs/dice/dice-six-faces-five.png")  
 }),  
 new THREE.MeshStandardMaterial({  
 map: ***textureLoader***.load("./imgs/dice/dice-six-faces-six.png")  
 }),  
 new THREE.MeshStandardMaterial({  
 map: ***textureLoader***.load("./imgs/dice/dice-six-faces-seven.png")  
 }),  
]  
  
const ***geometry*** = new THREE.BoxGeometry( 1, 1, 1 );  
const ***cube*** = new THREE.Mesh( ***geometry***, ***textureCube*** );  
const ***cube2*** = new THREE.Mesh( ***geometry***, ***textureCube2*** );  
***cube***.position.x = -1  
***cube2***.position.x = 1  
***scene***.add( ***cube*** );  
***scene***.add( ***cube2*** );  
  
***camera***.position.z = 10;  
  
function show() {  
 requestAnimationFrame( show );  
 ***renderer***.render( ***scene***, ***camera*** );  
}  
show();  
  
function throw\_dice(one\_dice){  
 const num = ***Math***.round(***Math***.random()\*5+1)  
  
 let xf = gradus\_to\_radian(1080)  
 let yf = gradus\_to\_radian(1080)  
  
 switch (num){  
 case 3: xf += gradus\_to\_radian(90); break;  
 case 4: xf += gradus\_to\_radian(270); break;  
 case 2: yf += gradus\_to\_radian(90); break;  
 case 1: yf += gradus\_to\_radian(270); break;  
 case 5: break;  
 case 6: xf += gradus\_to\_radian(180); break;  
 }  
  
 animation({  
 duration: 5000,  
 timing: ***circOut***,  
 draw(progress){  
 if(one\_dice===1){  
 ***cube***.rotation.x = xf \* progress  
 ***cube***.rotation.y = yf \* progress  
  
 if(progress<=0.1){  
 ***camera***.position.z = 10 - 50 \* progress  
 }  
 if(progress>=0.95){  
 ***camera***.position.z = 10 - 100 \* (1-progress)  
 }  
 }  
 else {  
 ***cube2***.rotation.x = xf \* progress  
 ***cube2***.rotation.y = yf \* progress  
 }  
 }  
 })  
  
 return num  
}  
  
function gradus\_to\_radian(grad){  
 const PI = 3.1415926535897932384626433832795  
 return grad\*PI/180  
}

**js/tree.js –** npmмодуль, що відповідає за 3Д графіку

**index.html** – html розмітка (дизайн) програми

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Document</title>  
  
 <link rel="stylesheet" type="text/css" href="style/style.css">  
  
 <script src="js/three.js"></script>  
</head>  
<body>  
  
<div class="play-table">  
  
 <div class="navigation">  
 <div id="playButton" class="button"><p>PLAY</p></div>  
 <div><p id="turn" style="font-style: italic; font-weight: normal; font-size: 40px; font-family: sans-serif; margin-top: 20px">Your turn</p></div>  
 </div>  
   
 <div class="restart-button">  
 <img src="imgs/restart.png" style="height: 100%; width: 100%" alt="restart">  
 </div>  
  
 <div class="fields">  
 <div class="field1">  
 <p id="playerName">Player</p>  
 <hr style="width: 50%; margin: 5px 0; border: 1px solid white"/>  
 <p>Points: <span id="playerPoints">0</span> - <span id="PP\_Full">0</span></p>  
 </div>  
 <div class="field2">  
 <p>Opponent</p>  
 <hr style="width: 50%; margin: 5px 0; border: 1px solid white"/>  
 <p>Points: <span id="opponentPoints">0</span> - <span id="OP\_Full">0</span></p>  
 </div>  
 </div>  
  
 <div class="dice" style="display: none"></div>  
  
  
 <div class="menu">  
 <div id="b-add" class="button">ADD</div>  
 <div id="b-subtract" class="button"><p>SUBTRACT</p></div>  
 <div id="b-x2" class="button">x2</div>  
 <div id="b-x3" class="button">x3</div>  
 <div id="b-x4" class="button">x4</div>  
 <div id="b-d2" class="button">/2</div>  
 <div id="b-d3" class="button">/3</div>  
 <div id="b-d4" class="button">/4</div>  
 </div>  
</div>  
  
<div class="intro">  
 <div class="text-container">  
 <p style="font-size: 100px">1</p>  
 <div class="middle13">  
 <p style="font-style: italic">laboratory work</p>  
 </div>  
 <p style="font-size: 100px">3</p>  
 </div>  
</div>  
  
<script src="js/intro.js"></script>  
<script src="js/opponent.js"></script>  
<script src="js/script.js"></script>  
<script src="js/main.js"></script>  
</body>  
</html>

**style/style.css –** стилі html документа

\*{  
 box-sizing: border-box;  
}  
p{  
 margin: 0;  
}  
  
body{  
 display: flex;  
 justify-content: center;  
 align-items: center;  
 width: 100vw;  
 height: 100vh;  
 padding: 0;  
 margin: 0;  
 background-color: lightgrey;  
}  
  
.play-table{  
 display: flex;  
 justify-content: center;  
 align-items: center;  
 position: relative;  
 width: 80%;  
 height: 80%;  
 background-color: mediumseagreen;  
 border: 20px solid saddlebrown;  
 opacity: 0;  
 transition: opacity 1s;  
 border-radius: 80px;  
}  
  
.intro{  
 position: absolute;  
 display: flex;  
 justify-content: center;  
 align-items: center;  
 width: 100%;  
 height: 100%;  
 font-size: 50px;  
 line-height: 50px;  
 font-weight: bold;  
 font-family: Arial, sans-serif;  
 color: white;  
 transition: opacity 1s;  
 opacity: 0;  
}  
  
.intro p{  
 margin: 0;  
 padding: 0;  
}  
  
.text-container{  
 display: flex;  
 flex-direction: row;  
 align-items: center;  
 justify-content: center;  
}  
  
.middle13{  
 width: 0;  
 display: flex;  
 justify-content: center;  
 border-top: 3px solid white;  
 border-bottom: 3px solid white;  
 padding: 10px 0;  
 margin: 0 15px;  
 white-space: nowrap;  
 overflow: hidden;  
}  
  
.fields{  
 display: flex;  
 flex-direction: row;  
 width: 100%;  
 height: 100%;  
 font-size: 50px;  
 color: white;  
 font-style: italic;  
 font-weight: bold;  
 text-align: center;  
}  
.fields p{  
 margin: 0;  
}  
.fields div{  
 display: flex;  
 flex-direction: column;  
 justify-content: end;  
 align-items: center;  
 flex: 1;  
}  
  
.navigation{  
 top: 0;  
 position: absolute;  
 display: flex;  
 flex-direction: column;  
 justify-content: center;  
 align-items: center;  
 width: 100%;  
 font-family: Arial, sans-serif;  
 font-weight: bold;  
 color: white;  
 font-size: 25px;  
}  
  
.button{  
 padding: 5px 10px;  
 border: 2px solid white;  
 margin: 5px;  
 border-radius: 10px;  
 transition: all 0.5s;  
 cursor: pointer;  
}  
.button:hover{  
 transform: scale(0.9);  
 opacity: 0.7;  
}  
.button.hover{  
 transform: scale(0.9);  
 opacity: 0.7;  
 background-color: lightgreen;  
}  
  
.dice{  
 display: none;  
 justify-content: center;  
 align-items: center;  
 width: 400px;  
 height: 200px;  
 position: absolute;  
 z-index: 1;  
 overflow: hidden;  
}  
  
.menu{  
 padding: 0 20px;  
 width: 100%;  
 position: absolute;  
 top: calc(100% + 25px);  
 display: flex;  
 justify-content: center;  
 align-items: center;  
 font-weight: bold;  
 font-family: Arial, sans-serif;  
 text-align: center;  
}  
.menu .button{  
 flex: 1;  
 z-index: 10;  
}  
  
.restart-button{  
 position: absolute;  
 top: 20px; left: 20px;  
 height: 30px; width: 30px;  
 border: 1px solid black;  
 border-radius: 5px;  
 padding: 5px;  
 background-color: whitesmoke;  
 transition: opacity 0.5s;  
 cursor: pointer;  
 z-index: 1;  
}  
.restart-button:hover{  
 opacity: 0.5;  
}  
  
.button .p{  
 height: 100%;  
 width: 100%;  
}

**imgs/… -** папка з зображеннями

### Приклади роботи

## C:\Users\starp\Desktop\Screenshot_1.png

## 

## 

Висновок

В рамках даної лабораторної роботи була розроблена гра за допомогою модуля Electron.js, 3d графіка була реалізована за допомогою Tree.js. Алгоритм вибору рішення ігровим ботом – мінімакс. Реалізований алгоритм в роботі є сильним прикладом роботи ігрових ботів. У даній роботі був помічений один мінус алгоритму – оскільки алгоритм перебирав усі можливі варіанти розвитку дій – то кількість потрібної оперативної пам’яті зростало в показниковій прогресії після кожного кроку глибини.

**Приклад.** У кожного випадкового числа є 8 варіантів розвитку (додати, відняти, помножити (2,3,4), поділити (2,3,4) ) + на кожному кроці глибини – 11 випадкових чисел. Отже <=88 варіантів на першому кроці (оскільки відняти не завжди можна). На наступному кроці для кожного з 88 варіантів ще 88.

Отже, у випадку гри «Тринадцять» - алгоритм працює, але можна замітити різке зростання потрібних ресурсів для обчислення ходу бота з кожним кроком глибини