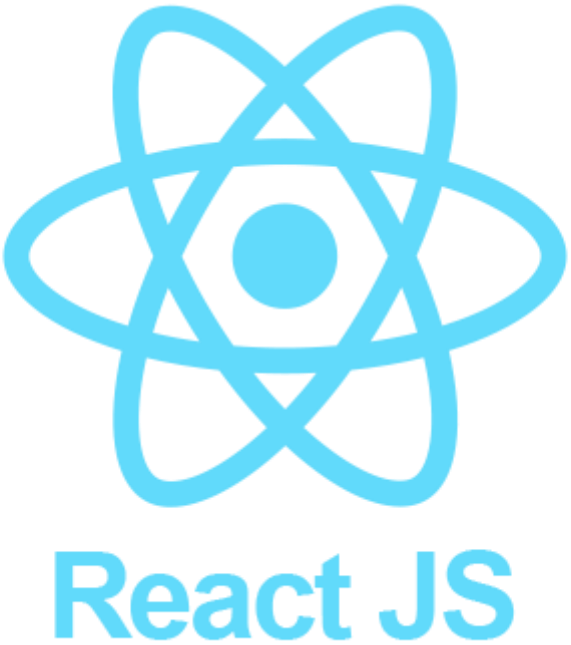


ASSIGNMENT-2

NAME : VIGNESHWAR K R

ROLL NO: 20I159

REACT JS:

React JS is a library of javascript, it is not a framework. There are various libraries like this. Example node js. React is a component based architecture. While HTML and CSS felt bored with dynamic web pages Java Scripts is developed by Brendan Eich. These webpages and its elements run on a browser and the first browser applied in Mozilla Firefox where Brendan Eich was working. Later it is applied in all the browsers. Every Browser has separate javascript engines. Not only in browsers, JS can be used in all displays. After this revolution ECMA script was created in1997.ECMAScript, also known as JavaScript, is a programming language adopted by the European Computer Manufacturer's Association as a standard for performing computations in Web applications

SPACE SHOOTER:

The game starts with space jet with 100% health, stop the meteor with the space jet which has ammo loaded in it. Initially it looks easy but as the score increases the difficulty will rise.

INSTRUCTIONS:

- Shoot the meteors
- Missing a meteor will deduct health
- Crashing space jet with meteor will also detuct health
- Moving beyond the frame will end the game.

PLATFORM:

The entire code of the game in coded in a online javascript compiler called “CodeSandBox” which is a live platform where it lively updates in web browser.

SOURCE FOLDER:

This folder contains our javascript files and its components files. A separate folder is created for components to avoid confusions and make a proper file structure. Other files are stored and linked with components. These file are rendered to display on the webpage.

COMPONENTS:

component is encapsulation of DOM elements. In this game i have created 3 components ‘player.js’,’bullet.js’ and ‘meteor.js’. The main use of javascript is to make changes in a particular element without making any changes to the entire page. Those particular elements are the components. In this game we are only going to make changes with these so the components. Here the use of 3 files is one for storage of images and operations whereas other one is for rendering into Apps.js.

PALYER.JS:

This player folder will first import the image of the space jet, sets the position of space jet, Health of space jet, speed, score. The health and scores are created in function as there is change in in these values. Assigning the keys to the movement of jet.

```
import img from './photos/player.png'
export class Player{
    dead = false;
    health = 100;
    ammo = 100;
    score = 0;
    speed = 25;
    firebullets = [];
    lastFireAt = Date.now();

    constructor(posX,posY){
        this.posX = 550;
        this.posY = 550;
    }

    deductHealth = () => {
        this.health -= 10
    }

    increaseScore = () =>{
        this.score += 10
    }

    update = (firecb) =>{
        document.onkeydown = (e) =>{
            if(e.keyCode === 39){
                this.posX += this.speed
            }
            if(e.keyCode === 37){
                this.posX -= this.speed
            }
            if(e.keyCode === 38){
                this.posY -= this.speed
            }
            if(e.keyCode === 40){
                this.posY += this.speed
            }
        document.addEventListener("keypress",(e) => {
            if(e.keyCode === 32){
                if(Date.now() - this.lastFireAt > 250){
                    firecb(this.posX + 32,this.posY);
                    this.lastFireAt = Date.now();
                }
            }
        })
        }
        if(this.posX < -10 || this.posX > 1450){
            this.dead = true
            gameOver(this.score);
            gameOver(this.score);
        }
        if (this.health <= 0) {
            this.dead = true;
            gameOver(this.score);
        }
        if (this.score === 50){
            this.speed = 35;
            this.score += 10;
        }
    }

    draw = (ctx) => {
        const image = new Image();
        image.src = img;
        ctx.drawImage(image,this.posX,this.posY,65,90);

        ctx.font = '16px Arial';
        ctx.fillStyle = "yellow";
        ctx.fillText(`Health: ${this.health}`, 950 - -450, 550 - -125);

        ctx.font = '16px Arial';
        ctx.fillStyle = "lightgreen";
        ctx.fillText(`Score: ${this.score}`, 15, 25);
    }
}


function gameOver(score) {
    document.body.innerHTML = `
    <center>
    <br/>
    <h2>Game Over!</h2>
    <p>Your Score: ${score}</p>
    <button class="btn btn-danger mt-2" onClick="location.reload()">Again</button>
    </center>
    `
}

export default Player;
```

BULLET.JS:

This bullet.js file contains speed of bullet movements, position of launching phase, size of bullets and shape of bullets.

```
import Player from "./Player";

export class Bullet{
    dead = false;
    speed = 10;

    constructor(xPos,yPos){
        this.xPos = xPos
        this.yPos = yPos
    }

    update = () => {
        this.yPos -= this.speed;

        if(this.yPos < 0 || this.yPos > 680){
            this.dead = true
        }
        if (Player.score >=100){
            this.speed = 15;
        }
    }

    draw = (ctx) => {
        ctx.beginPath();
        ctx.arc(this.xPos,this.yPos,5,0,2*Math.PI);
        ctx.fillStyle = "#FF1E1E";
        ctx.fill();
        ctx.lineWidth = 1;
        ctx.stroke()
    }
}

export default Bullet;
```

METEOR.JS:

This meteor.js will also contains same properties of bullet file like size, shape, speed, falling position etc.

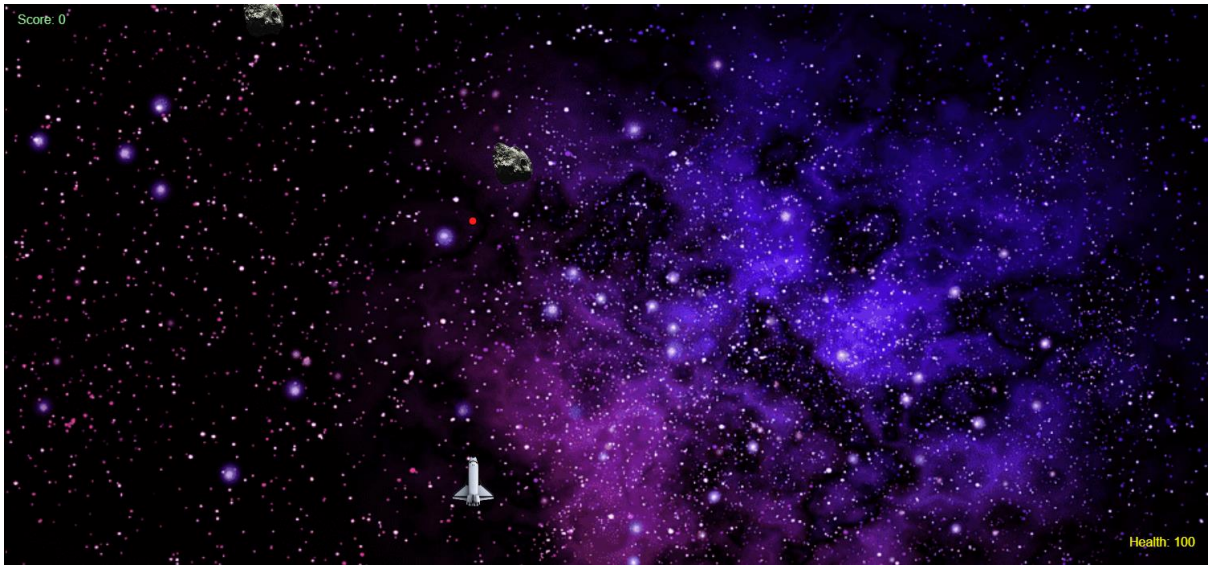
```
import img from './photos/meteors.png'
export class Meteor {
    speed = 4;
    dead = false;

    constructor(xPos,yPos){
        this.xPos = xPos;
        this.yPos = yPos;
    }

    isDead = () =>{
        if(this.yPos > 650){
            return true;
        }
    }

    update = (player,bullets) => {
        if (this.dead) return;
        this.yPos += this.speed;

        if(!this.dead && this.isDead()){
            this.dead = true;
            player.deductHealth();
        }
        if (player.score >= 100){
            this.speed = 8;
        }

        if(!this.dead){
            bullets.forEach(bullet => {
                if(Math.abs(bullet.xPos - this.xPos) < 75 && Math.abs(bullet.yPos - this.yPos) < 100){
                    player.increaseScore();
                    this.dead = true;
                    bullet.dead = true;
                }
            });

            if(!this.dead){
            if(Math.abs(player.posX - this.xPos) < 65 && Math.abs(player.posY - this.yPos) < 90){
                this.dead = true;
                player.deductHealth();
                }
            }
        }

    }

    draw = (ctx) =>{
        const image = new Image();
        image.src = img;
        ctx.drawImage(image,this.xPos,this.yPos,75,100);
    }
}

export default Meteor;
```

APP.JS:

```
import React, { useEffect } from 'react';
import { Player } from './Player';
import bg from './photos/space2.png';
import { Meteor } from './Meteor'
import { Bullet } from './Bullet';

function App() {
  let canvas;
  let ctx;
  let maxMeteorCount = 10;
  let lastMeteorSpawnAt = Date.now();

  const player = new Player(950 / 2,550 / 1.5)
  const randomNumber = (min,max) => Math.random() * max + min;

  useEffect(() => {
    // eslint-disable-next-line react-hooks/exhaustive-deps
    canvas = document.getElementById("myCanvas");

    let meteors = []
    let bullets = []
    const fireBulletcb = (xpos,ypos) => bullets.push(new Bullet(xpos,ypos));

    setInterval(() => {
      // eslint-disable-next-line react-hooks/exhaustive-deps
      ctx = canvas.getContext("2d");
      ctx.clearRect(0,0,1500,700);

      player.update(fireBulletcb);
      player.draw(ctx);

      const random = randomNumber(0,1300);
      if(meteors.length < maxMeteorCount && (Date.now() - lastMeteorSpawnAt) > 1500){
        meteors.push(new Meteor(random,-200));
        lastMeteorSpawnAt = Date.now();
      }
```

```
      }

      meteors = meteors.filter((enemy) => !enemy.dead);
      meteors.forEach(meteor => {
        meteor.update(player,bullets);
        meteor.draw(ctx);
      });

      bullets = bullets.filter((bullet) => !bullet.dead);
      bullets.forEach(bullet => {
        bullet.update();
        bullet.draw(ctx);
      });

    }, 1000 / 30);
  })
  return (
    <div style={{
    display:'flex',justifyContent:'center',alignItems:'center',height:'100%',flexDirection:'row'
    }}>
        <canvas id="myCanvas" width="1500" height="700" style={{backgroundImage: `url(${bg})`,backgroundSize:"cover" ,border:'2px solid #000000',marginTop:'
    </div>

  );
}

export default App;
```

OUTPUT:


Score: 0
Health: 100