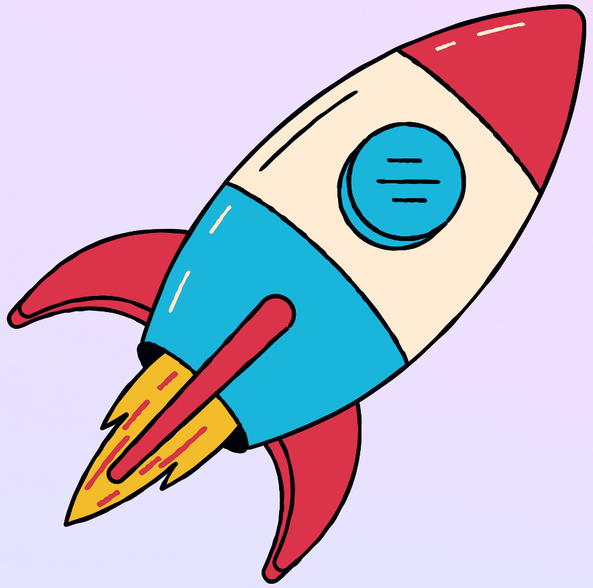
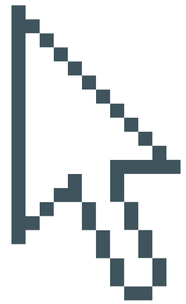


ROCKET MUMBUL

FINAL PROJECT

GRAFIKA KOMPUTER

KELAS F



BY KELOMPOK 4

MEET THE TEAM

KURNIA CAHYA FEBRYANTO

5025201073

MUHAMMAD ISMAIL

5025201223

AHMAD FERDIANSYAH RAMADHANI

5025201218

RERE ARGA DEWANATA

5025201078

ROCKET MUMBUL

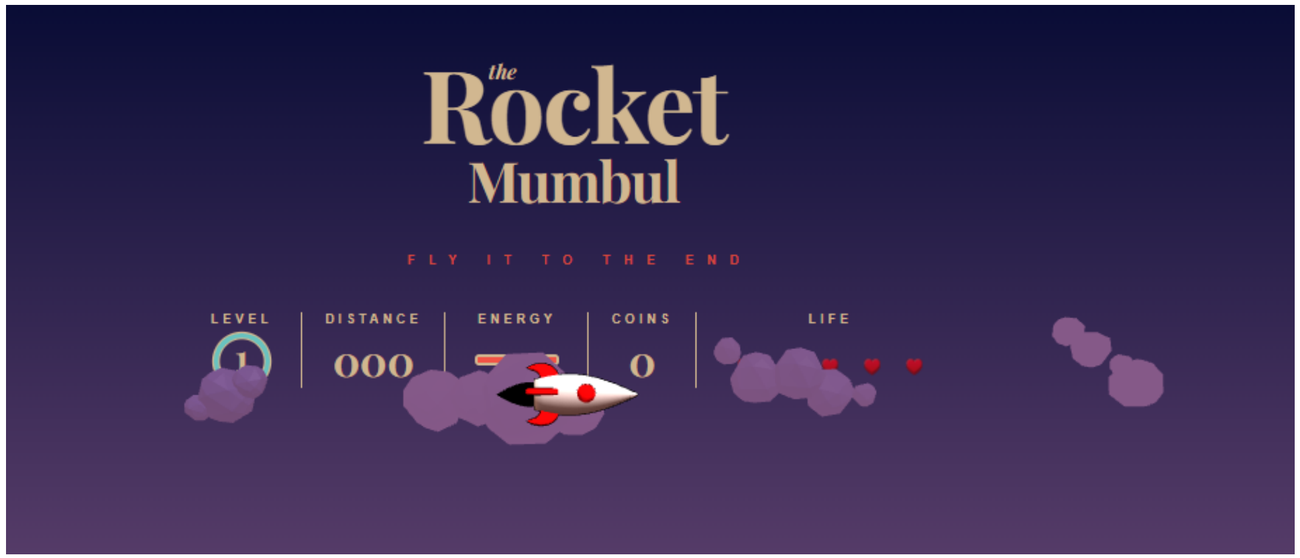
Implementasi Permainan Web Aplikasi Rocket Mumbul dengan Collision, Viewing, dan Fitur Lainnya

BY KELOMPOK 4

ROCKET MUMBUL

LATAR BELAKANG

Game "Rocket Mumbul" dibuat dengan tujuan untuk relaksasi dan hiburan yang dapat diakses oleh semua orang karena berjalan di web browser. Permainan ini dimainkan dengna menjalankan sebuah roket untuk menghindari rintangan yang diberikan dan mengumpulkan poin sebanyak-banyaknya.



TUJUAN PEMBUATAN

Adapun tujuan kami membuat permainan web aplikasi “Roket Mumbul” adalah sebagai berikut:

Implementasi Mata Kuliah

Menerapkan dan mengimplementasikan materi pembelajaran mata kuliah Grafika Komputer

Hiburan

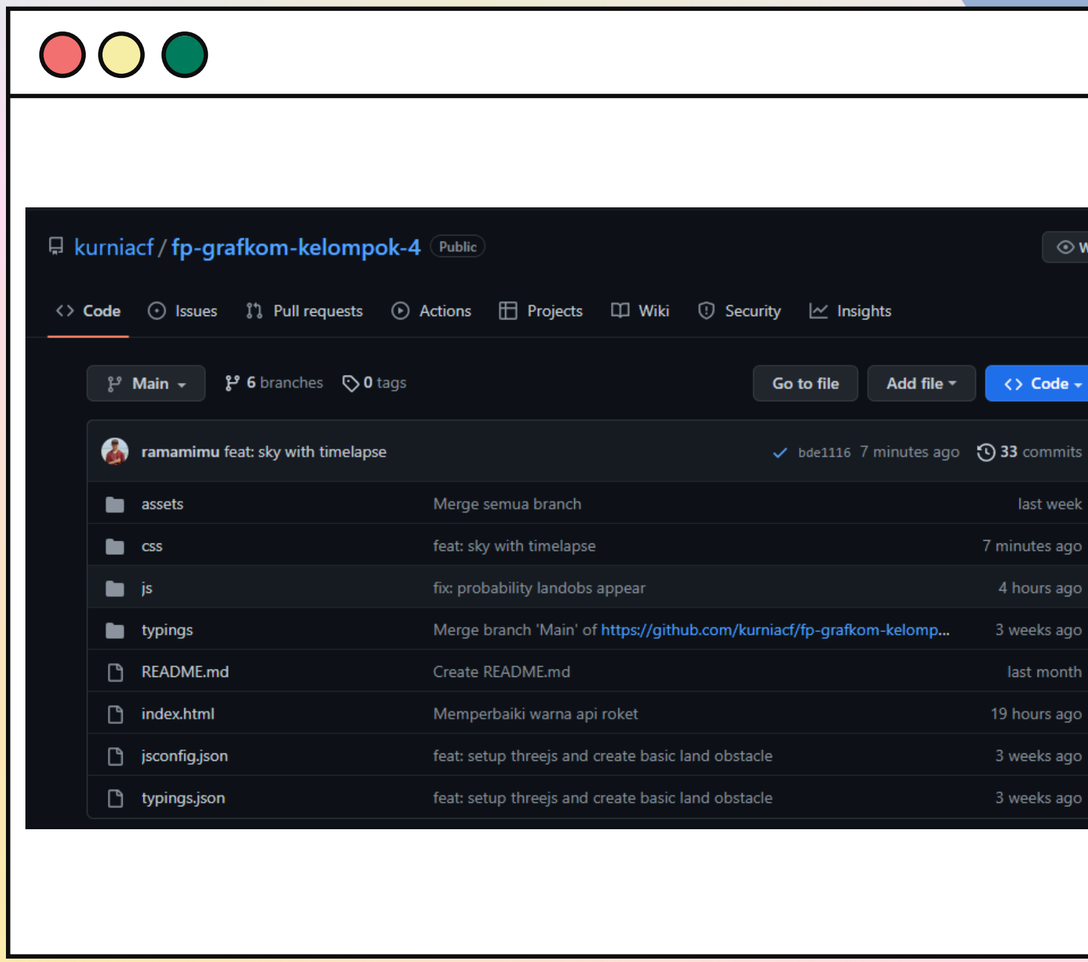
Permainan yang dibuat diharapkan sebagai sarana relaksasi dan hiburan

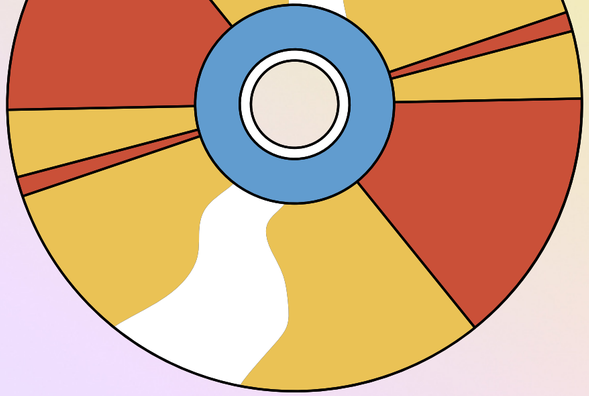
Experience the thrill of space exploration with Rocket Mumbul

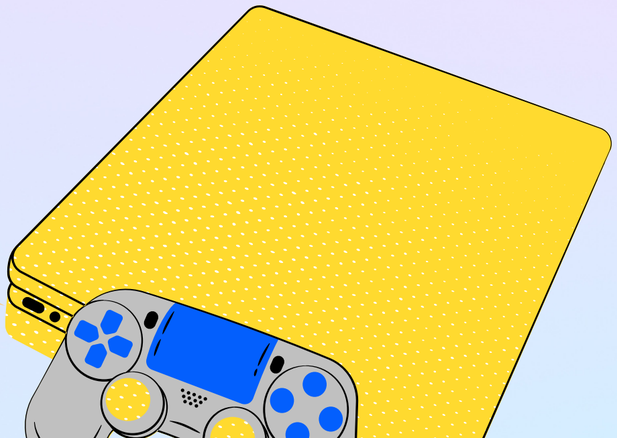
ROCKET MUMBUL

TEKNOLOGI YANG DIGUNAKAN

- **Text Editor:**
  Visual Studio Code
- **Bahasa Pemrograman:**
  HTML, CSS, Javascript
- **Manajemen Proyek:**
  Github
- **Library:**
  Three JS



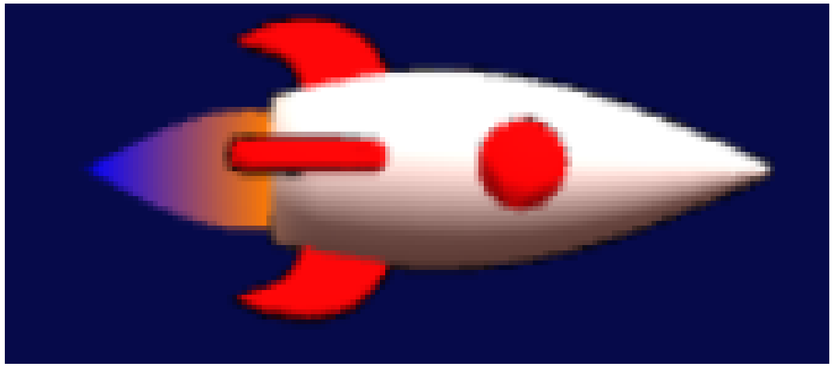




FITUR

ROCKET MUMBUL

3D MODELLING



3D modeling dapat dilakukan dengan cara membuat objek 3D menggunakan kelas-kelas seperti BoxGeometry, SphereGeometry, CylinderGeometry, dan sebagainya.

Setelah objek 3D dibuat atau dimuat, kita dapat mengubah posisi, rotasi, dan skala objek tersebut sesuai kebutuhan dengan mengubah properti yang tersedia pada instance dari kelas Object3D

3D MODELLING

OUTLINE SHADER & ROCKET HEAD

```
var Rocket = function(){
  var OutlineShader = {
    uniforms: {
      offset: { type: 'f', value: 0.3 },
      color: { type: 'c', value: new THREE.Color('#000000') },
      alpha: { type: 'f', value: 1.0 },
    },

    vertexShader:`
      uniform float offset;
      void main() {
        vec4 pos = modelViewMatrix * vec4( position + normal * offset, 1.0 );
        gl_Position = projectionMatrix * pos;
      }
    `,

    fragmentShader:`
      uniform vec3 color;
      uniform float alpha;
      void main() {
        gl_FragColor = vec4( color, alpha );
      }
    `
  };
  ...
}
```

```
var Rocket = function(){

  ...
   var rocketGeo = new THREE.LatheGeometry( points, 32 );

  var rocketMat = new THREE.MeshPhongMaterial({
    color: 0xcccccc,
    shininess: 1
  });

  var rocketOutlineMat = new THREE.ShaderMaterial({
    uniforms: THREE.UniformsUtils.clone( OutlineShader.uniforms ),
    vertexShader: OutlineShader.vertexShader,
    fragmentShader: OutlineShader.fragmentShader,
    side: THREE.BackSide,
  });

  var rocketObj = THREE.SceneUtils.createMultiMaterialObject(
    rocketGeo, [ rocketMat, rocketOutlineMat ]
  );
  rocketObj.scale.setScalar( 0.1 );
  rocket.add( rocketObj );
}
```

3D MODELLING

WINDOW & ROCKET BODY

```
var Rocket = function(){

  ...
   var portalGeo = new THREE.CylinderBufferGeometry( 0.26, 0.26, 1.6, 32 );

  var portalMat = new THREE.MeshPhongMaterial({ color: 0x90dcff });

  var portalOutlineMat = rocketOutlineMat.clone();
  portalOutlineMat.uniforms.offset.value = 0.03;
  var portal = THREE.SceneUtils.createMultiMaterialObject(
    portalGeo, [ portalMat, portalOutlineMat ]
  );
  portal.position.y = 2;
  portal.rotation.x = Math.PI / 2;
  rocket.add( portal );
}
```

```
var Rocket = function(){

  ...
  var circle = new THREE.Shape();
  circle.absarc( 0, 0, 3.5, 0, Math.PI * 2 );

  var hole = new THREE.Path();
  hole.absarc( 0, 0, 3, 0, Math.PI * 2 );
  circle.holes.push( hole );

  var tubeExtrudeSettings = {
    amount: 17,
    steps: 1,
    bevelEnabled: false
  };
  var tubeGeo = new THREE.ExtrudeGeometry( circle, tubeExtrudeSettings
  tubeGeo.computeVertexNormals();
  tubeGeo.center();

  var tubeMat = new THREE.MeshPhongMaterial({
    color: 0xff0000,
    shininess: 1
  });
  var tubeOutlineMat = rocketOutlineMat.clone();
  tubeOutlineMat.uniforms.offset.value = 0.2;
  var tube = THREE.SceneUtils.createMultiMaterialObject(
    tubeGeo, [ tubeMat, tubeOutlineMat ]
  );
  tube.position.y = 2;
  tube.scale.setScalar( 0.1 );
  rocket.add( tube );

}
```

3D MODELLING

ROCKET WINGS

```
var Rocket = function(){

  ...
  var shape = new THREE.Shape();

  shape.moveTo( 3, 0 );
  shape.quadraticCurveTo( 25, -8, 15, -37 );
  shape.quadraticCurveTo( 13, -21, 0, -20 );
  shape.lineTo( 3, 0 );

  var extrudeSettings = {
    steps: 1,
    amount: 4,
    bevelEnabled: true,
    bevelThickness: 2,
    bevelSize: 2,
    bevelSegments: 5
  };

  var wingGroup = new THREE.Group();
  rocket.add( wingGroup );

  var wingGeo = new THREE.ExtrudeGeometry( shape, extrudeSettings );
  wingGeo.computeVertexNormals();

  var wingMat = new THREE.MeshPhongMaterial({
    color: 0xff0000,
    shininess: 1,
  });

}
```

```
var Rocket = function(){

  ...
  var wingOutlineMat = rocketOutlineMat.clone();
  wingOutlineMat.uniforms.offset.value = 1;
  var wing = THREE.SceneUtils.createMultiMaterialObject(
    wingGeo, [ wingMat, wingOutlineMat ]
  );
  wing.scale.setScalar( 0.03 );
  wing.position.set( .6, 0.9, 0 );
  wingGroup.add( wing );

  var wing2 = wingGroup.clone();
  wing2.rotation.y = Math.PI;
  rocket.add( wing2 );

  var wing3 = wingGroup.clone();
  wing3.rotation.y = Math.PI / 2;
  rocket.add( wing3 );

  var wing4 = wingGroup.clone();
  wing4.rotation.y = - Math.PI / 2;
  rocket.add( wing4 );
}
```

3D MODELLING

ROCKET FLAME

```
var Rocket = function(){

  ...
   var fireGeo = new THREE.LatheGeometry( firePoints, 32 );
  var fireMat = new THREE.ShaderMaterial({
    uniforms: {
      color1: { type: 'c', value: new THREE.Color('blue') },
      color2: { type: 'c', value: new THREE.Color(0xff7b00)}
    },
    vertexShader: `
      varying vec2 vUv;
      void main() {
        vUv = uv;
        gl_Position = projectionMatrix * modelViewMatrix * vec4(position,1.0);
      }
    `,
    fragmentShader: `
      uniform vec3 color1;
      uniform vec3 color2;
      varying vec2 vUv;
      void main() {
        gl_FragColor = vec4(mix(color1, color2, vUv.y), 1.0);
      }
    `,
  });
}
```

```
var Rocket = function(){

  ...
  this.fire = new THREE.Mesh(fireGeo, fireMat);
  this.fire.scale.setScalar( 0.06 );
  rocket.add( this.fire );

  var fireLight = new THREE.PointLight( 0xff7b00, 1, 9 );
  fireLight.position.set( 0, -1, 0 );
  rocket.add( fireLight );
}
```

ROCKET MUMBUL

KOIN DAN SKORING

Koin yang melayang di udara dapat diambil oleh user untuk mengisi gauge energy. Apabila gauge energy telah penuh maka boost akan aktif. Jumlah koin yang dikumpulkan akan tampil pada kolom "Coins"

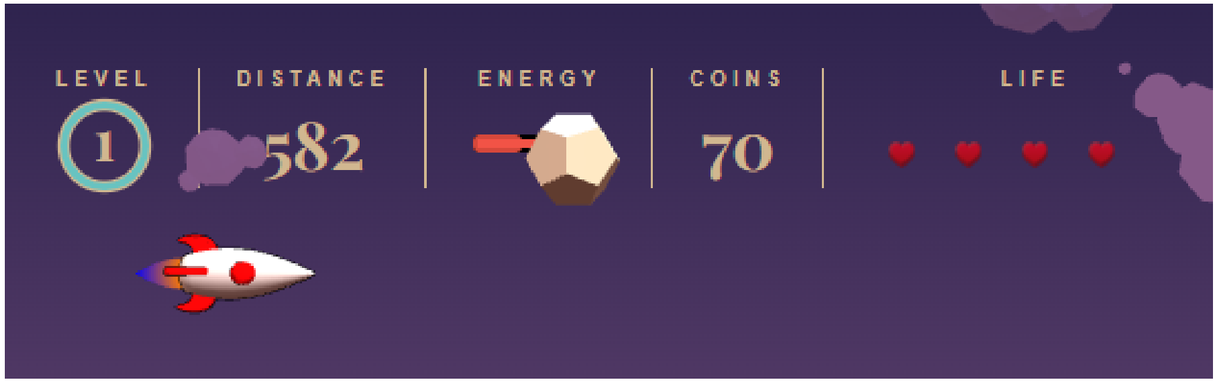
KOIN

```
Coin = function(){
  let geom = new THREE.CylinderGeometry(4, 4, 1, 10)

  let mat = new THREE.MeshPhongMaterial({
    color: 0xFFD700 ,
    shininess: 1,
    specular: 0xffffff,
    flatShading: true,

    shading:THREE.FlatShading
  });

  this.mesh = new THREE.Mesh(geom,mat);
  this.mesh.castShadow = true;
  this.angle = 0;
  this.dist = 0;
}
```

SCORING

```
function addEnergy(){
  ...

  game.score += game.coinValue;
  ...
  coinsCounter.innerHTML =  game.score;

  ...
}
```

LIFE



Life atau nyawa adalah fitur yang melambangkan nyawa yang dimiliki oleh user. Nyawa dapat berkurang apabila mengenai obstacle. Apabila nyawa telah habis, maka akan terjadi game over. Nyawa dapat ditambahkan apabila mengumpulkan koin. Pada awalnya user akan memiliki 5 nyawa.

INIT

```
function resetGame(){
  game = {
    ...
    heart:5,
    heart_point:0,
    ...
  };

  for(var i=0; i<game.heart; i++){
    if(!heart_init){
      var heart_icon = document.createElement('img');
      heart_icon.src = "./assets/photo/heart_icon.png"
      heart_icon.setAttribute("alt", "heart");
      heart_icon.classList.add('heart')
      heart_icon.classList.add('visible')
      healthCounter.appendChild(heart_icon)
    } else{
      healthCounter.childNodes[i].classList.remove('invisible')
      healthCounter.childNodes[i].classList.add('visible')
    }
  }
}
```

UPDATE

```
function updateEnergy(){
  game.heart = Math.max(0, game.heart);

  if (game.heart <1){
    game.status = "gameover";
  }
}
```

ADD AND REMOVE ENERGY

```
function addEnergy(){
  if (game.status == 'gameover') return;

  game.heart = Math.min(game.heart, 6);

  game.score += game.coinValue;
  game.heart_point += 2;
  coinsCounter.innerHTML =  game.score;

  let invisible = healthCounter.getElementsByClassName('invisible')
  if(game.heart_point >=10 && game.heart < 5){
    healthCounter.childNodes[healthCounter.childNodes.length -
invisible.length].classList.add('visible')
    healthCounter.childNodes[healthCounter.childNodes.length -
invisible.length].classList.remove('invisible')
    game.heart +=1;
    game.heart_point = 0;
  }

  // Play pickup sound
  var asd = pickupSound.play();
}
```

```
function removeEnergy(){
  if (game.status == 'gameover') return;
  game.heart -= 1;
  game.heart = Math.max(0, game.heart)

  let visible = healthCounter.getElementsByClassName('visible')

  if(visible.length){
    healthCounter.childNodes[visible.length-1].classList.add('invisible')
    healthCounter.childNodes[visible.length-1].classList.remove('visible')
  }
  // Play hurt sound
  hurtSound.play();
}
```

OBSTACLE

Obstacle adalah sebuah rintangan yang dapat menghadang pemain selama permainan berlangsung. Apabila pemain mengenai obstacle tersebut, pemain akan kehilangan nyawanya dan apabila nyawa tersebut telah habis, maka akan terjadi "Game Over".

Obstacle terdiri dari 2, yaitu:

1. Sky Obstacle = obstacle yang berada di udara
2. Land Obstacle = obstacle yang berada di darat

SKY OBSTACLE

```
// Fungsi Utama
function createObstacles() {
  for (let i = 0; i < 10; i++) {
    let obstacle = new Obstacle();
    obstaclesPool.push(obstacle);
  }
  obstaclesHolder = new ObstaclesHolder();
  scene.add(obstaclesHolder.mesh);
}
```

```
// Fungsi pembuatan objek obstacle
Obstacle = function () {
  let geom = new THREE.DodecahedronGeometry(8, 0);
  let mat = new THREE.MeshPhongMaterial({
    color: new THREE.Color(0xc7bca1),
    shininess: 0,
    specular: 0x252525,
    shading: THREE.FlatShading,
  });
  this.mesh = new THREE.Mesh(geom, mat);
  this.mesh.castShadow = true;
  this.angle = 0.25;
  this.dist = 0;
};
```

SKY OBSTACLE

```
// Fungsi spawn obstacle
ObstaclesHolder.prototype.spawnObstacles = function () {
  let nObstacles = game.level;

  for (let i = 0; i < nObstacles; i++) {
    let obstacle;
    if (obstaclesPool.length) {
      obstacle = obstaclesPool.pop();
    } else {
      obstacle = new Obstacle();
    }

    obstacle.angle = -(i * 0.1);
    obstacle.distance =
      game.waterRadius +
      game.planeDefaultHeight +
      (-1 + Math.random() * 2) * (game.planeAmpHeight - 20);
    obstacle.mesh.position.y = -game.waterRadius + Math.sin(obstacle.angle) * obstacle.distance;
    obstacle.mesh.position.x = Math.cos(obstacle.angle) * obstacle.distance;

    this.mesh.add(obstacle.mesh);
    this.obstaclesInUse.push(obstacle);
  }
};
```

```
// Fungsi rotate obstacle
ObstaclesHolder.prototype.rotateObstacles = function () {
  for (let i = 0; i < this.obstaclesInUse.length; i++) {
    let obstacle = this.obstaclesInUse[i];
    obstacle.angle += game.speed * deltaTime * game.obstaclesSpeed;

    if (obstacle.angle > Math.PI * 2) obstacle.angle -= Math.PI * 2;

    obstacle.mesh.position.y =
      -game.waterRadius + Math.sin(obstacle.angle) * obstacle.distance + 30;
    obstacle.mesh.position.x = Math.cos(obstacle.angle) * obstacle.distance;
    obstacle.mesh.rotation.z += Math.random() * 0.1;
    obstacle.mesh.rotation.y += Math.random() * 0.1;

    var diffPos = airplane.rocketGroup.position.clone().sub(obstacle.mesh.position.clone());
    var d = diffPos.length();
    if (d < game.obstacleDistanceTolerance) {
      obstaclesPool.unshift(this.obstaclesInUse.splice(i, 1)[0]);
      this.mesh.remove(obstacle.mesh);
      game.planeCollisionSpeedX = (100 * diffPos.x) / d;
      game.planeCollisionSpeedY = (100 * diffPos.y) / d;
      ambientLight.intensity = 2;

      removeEnergy();
      i--;
    } else if (obstacle.angle > Math.PI) {
      obstaclesPool.unshift(this.obstaclesInUse.splice(i, 1)[0]);
      this.mesh.remove(obstacle.mesh);
      i--;
    }
  }
};
```

LAND OBSTACLE

```
// Fungsi utama
function createLandObstacles() {
  for (let i = 0; i < 10; i++) {
    let landObstacle = new LandObstacle();
    landObstaclesPool.push(landObstacle);
  }
  landObstaclesHolder = new LandObstacleHolder();
  scene.add(landObstaclesHolder.mesh);
}
```

```
// Fungsi pembuatan objek land obstacle
LandObstacle = function () {
  let geom = new THREE.BoxGeometry(20, 70, 20);
  let mat = new THREE.MeshPhongMaterial({
    color: new THREE.Color(0x8b7e74),
    shininess: 0,
    specular: 0x252525,
    shading: THREE.FlatShading,
  });
  this.mesh = new THREE.Mesh(geom, mat);
  this.mesh.castShadow = true;
  this.angle = 0.25;
  this.dist = 0;
};
```

LAND OBSTACLE

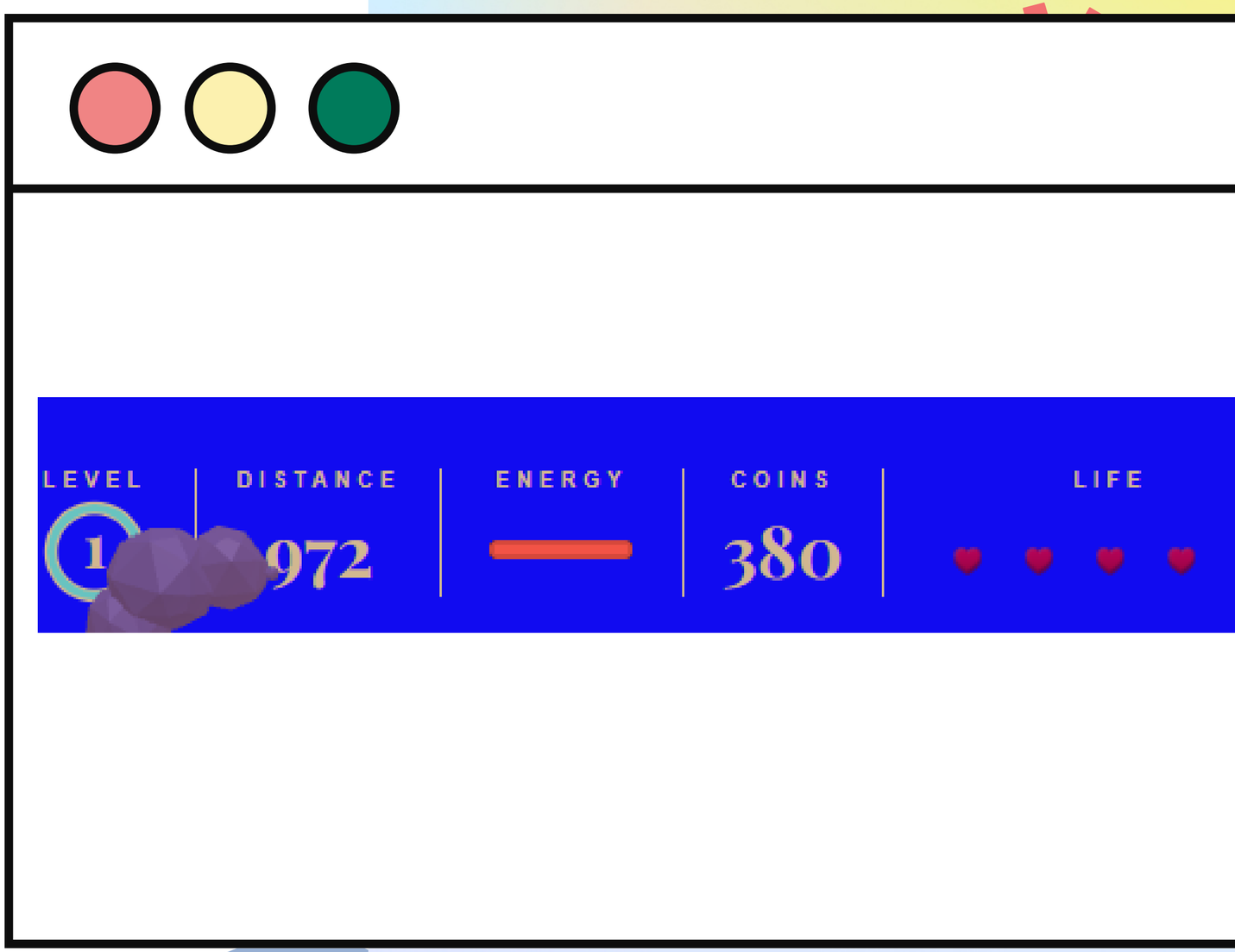
```
// Fungsi spawnObstacle untuk land obstacle
LandObstacleHolder.prototype.spawnObstacles = function () {
  let nObstacles = game.level;

  for (let i = 0; i < nObstacles; i++) {
    let landObstacle;
    if (landObstaclesPool.length) {
      console.log("Posisi Land Obstacle y = " + landObstaclesPool[0].mesh.position.y);
      landObstacle = landObstaclesPool.pop();
    } else {
      landObstacle = new LandObstacle();
    }

    landObstacle.angle = -(i * 0.8);
    landObstacle.distance =
      game.waterRadius + game.planeDefaultHeight + (-2 + Math.random()) * (game.planeAmpHeight -
    landObstacle.mesh.position.y =
      -game.waterRadius + Math.sin(landObstacle.angle) * (landObstacle.distance - 2);
    landObstacle.mesh.position.x = Math.cos(landObstacle.angle) * landObstacle.distance;

    this.mesh.add(landObstacle.mesh);
    this.landObstaclesInUse.push(landObstacle);
  }
};
```

```
// Fungsi rotateObstacle bagi untuk land obstacle
LandObstacleHolder.prototype.rotateObstacles = function () {
  for (let i = 0; i < this.landObstaclesInUse.length; i++) {
    let obstacle = this.landObstaclesInUse[i];
    obstacle.angle += game.speed * deltaTime * game.obstaclesSpeed;

    if (obstacle.angle > Math.PI * 2) obstacle.angle -= Math.PI * 2;

    obstacle.mesh.position.y =
      -game.waterRadius + Math.sin(obstacle.angle) * (obstacle.distance - 2) + 10;
    obstacle.mesh.position.x = Math.cos(obstacle.angle) * obstacle.distance;

    var diffPos = airplane.rocketGroup.position.clone().sub(obstacle.mesh.position.clone());
    var d = diffPos.length();
    if (d < game.landObstacleDistanceTolerance) {
      landObstaclesPool.unshift(this.landObstaclesInUse.splice(i, 1)[0]);
      game.planeCollisionSpeedX = (100 * diffPos.x) / d;
      game.planeCollisionSpeedY = (100 * diffPos.y) / d;
      this.mesh.remove(obstacle.mesh);
      ambientLight.intensity = 2;
      removeEnergy();
      i--;
    } else if (obstacle.angle > Math.PI) {
      landObstaclesPool.unshift(this.landObstaclesInUse.splice(i, 1)[0]);
      this.mesh.remove(obstacle.mesh);
      i--;
    }
  }
};
```

ROCKET MUMBUL

SKILL BOOST

User dapat mengaktifkan skill boost apabila gauge energy telah penuh. Dengan aktifnya skill ini, pesawat akan bergerak jauh lebih cepat dibandingkan sebelumnya



SKILL BOOST

```
// Apabila boost aktif
if (BoostStatus == 1) {
  game.speed = game.baseSpeed * game.planeBoostSpeed;
  boostSound.pause();
} else if (BoostStatus == 0) {
  boostSound.play();
}

// Boost akan aktif apabila energy bar telah penuh
  if (game.energy < 100) {
  game.energy += game.ratioSpeedEnergy;
  game.energy > 100 ? (game.energy = 100) : (game.energy =
  game.energy);
  BoostStatus = 1;
    if (game.energy >= 100) {
      BoostStatus = 0;
      setTimeout(function () {
        game.energy = 10;
        BoostStatus = 1;
        boostSound.pause();
      }, 3000);
    }
 }
```

ROCKET MUMBUL

LIGHTING & SHADOW

Lighting membantu untuk menciptakan ilusi ruang di dalam sebuah scene 3D, menambahkan kesan kedalaman dan detail ke objek-objek yang ada di dalamnya. Sedangkan Shadow membantu untuk menciptakan ilusi objek yang terpapar sinar cahaya dan memiliki bayangan di permukaan lain

LIGHTING & SHADOW

```
function createLights() {
  hemisphereLight = new THREE.HemisphereLight(0xaaaaaa, 0x000000, 0.9);
  ambientLight = new THREE.AmbientLight(0xdc8874, 0.5);
  shadowLight = new THREE.DirectionalLight(0xffffff, 0.9);
  shadowLight.position.set(150, 350, 350);
  shadowLight.castShadow = true;
  shadowLight.shadow.camera.left = -400;
  shadowLight.shadow.camera.right = 400;
  shadowLight.shadow.camera.top = 400;
  shadowLight.shadow.camera.bottom = -400;
  shadowLight.shadow.camera.near = 1;
  shadowLight.shadow.camera.far = 1000;
  shadowLight.shadow.mapSize.width = 4096;
  shadowLight.shadow.mapSize.height = 4096;
  var ch = new THREE.CameraHelper(shadowLight.shadow.camera);

  scene.add(hemisphereLight);
  scene.add(shadowLight);
  scene.add(ambientLight);
}
```

ROCKET MUMBUL

3D VIEWING

Adanya pilihan bagi user untuk memainkan game secara 3D dengan 3rd person view dari belakang roket. Pada mode ini, bentuk obstacle baik di udara maupun di darat di sesuaikan agar lebih mudah bagi pemain

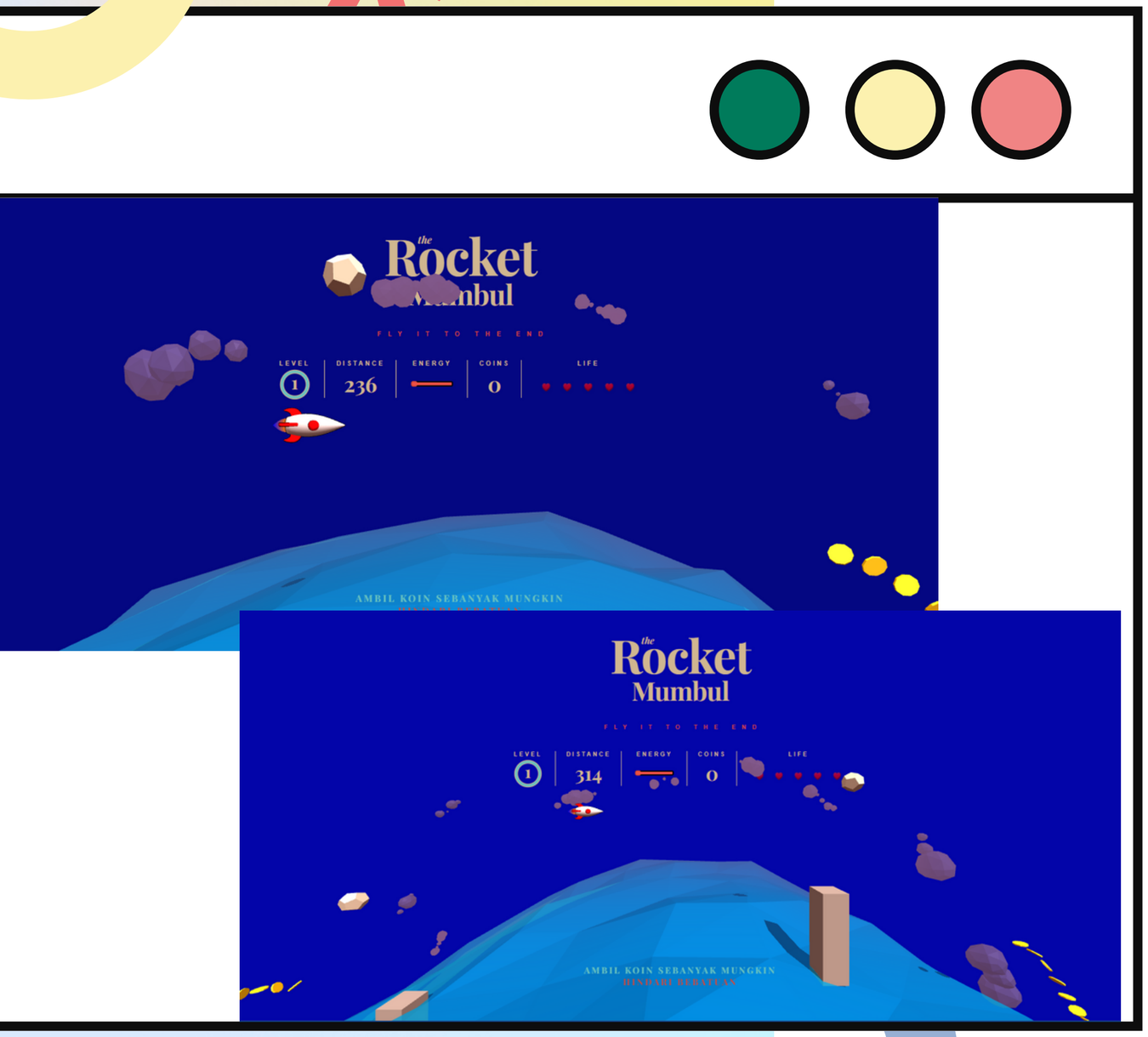
3D VIEWING

```
document.addEventListener("keydown", handleKeyDown, false);

function handleKeyDown(event) {
  switch (event.keyCode) {
    // jika menekan 2
    case 50:
      if (game.status != "playing") {
        is3D = false;
      }
      break;
    // jika menekan 3
    case 51:
      if (game.status != "playing") {
        is3D = true;
      }
      break;
  }
}
```

3D VIEWING

```
if (is3D) camera.position.set(-200, 100, 20);
else camera.position.set(0, 400, game.planeDefaultHeight + 100);
```

```
if (is3D) {
  camera.lookAt(
    new THREE.Vector3(
      airplane.rocketGroup.position.x,
      airplane.rocketGroup.position.y,
      airplane.rocketGroup.position.z
    )
  );
} else {
  camera.lookAt(
    new THREE.Vector3(0, airplane.rocketGroup.position.y, airplane.rocketGroup.position.z)
  );
}
```



ZOOM PERSPECTIVE

Memungkinkan pengguna untuk melakukan zoom in dan zoom out dengna mengarahkan mouse ke kiri dan ke kanan

ZOOM PERSPECTIVE

```
document.addEventListener("touchmove", handleTouchMove, false);

function handleTouchMove(event) {
  event.preventDefault();
  var tx = -1 + (event.touches[0].pageX / WIDTH) * 2;
  var ty = 1 - (event.touches[0].pageY / HEIGHT) * 2;
  mousePos = { x: tx, y: ty };
}
```

ZOOM PERSPECTIVE

```
function updatePlane() {
  ...
  let targetY = normalize(
    mousePos.y,
    -0.75,
    0.75,
    game.planeDefaultHeight - game.planeAmpHeight,
    game.planeDefaultHeight + game.planeAmpHeight
  );

  airplane.rocketGroup.position.y +=
    (targetY - airplane.rocketGroup.position.y) * deltaTime * game.planeMoveSensivity;

  airplane.rocketGroup.rotation.x =
    (airplane.rocketGroup.position.y - targetY) * deltaTime * game.planeRotZSensivity;

  camera.position.y +=
    (airplane.rocketGroup.position.y - camera.position.y) * deltaTime * game.cameraSensivity;
  ...
}
```

ROCKET MUMBUL

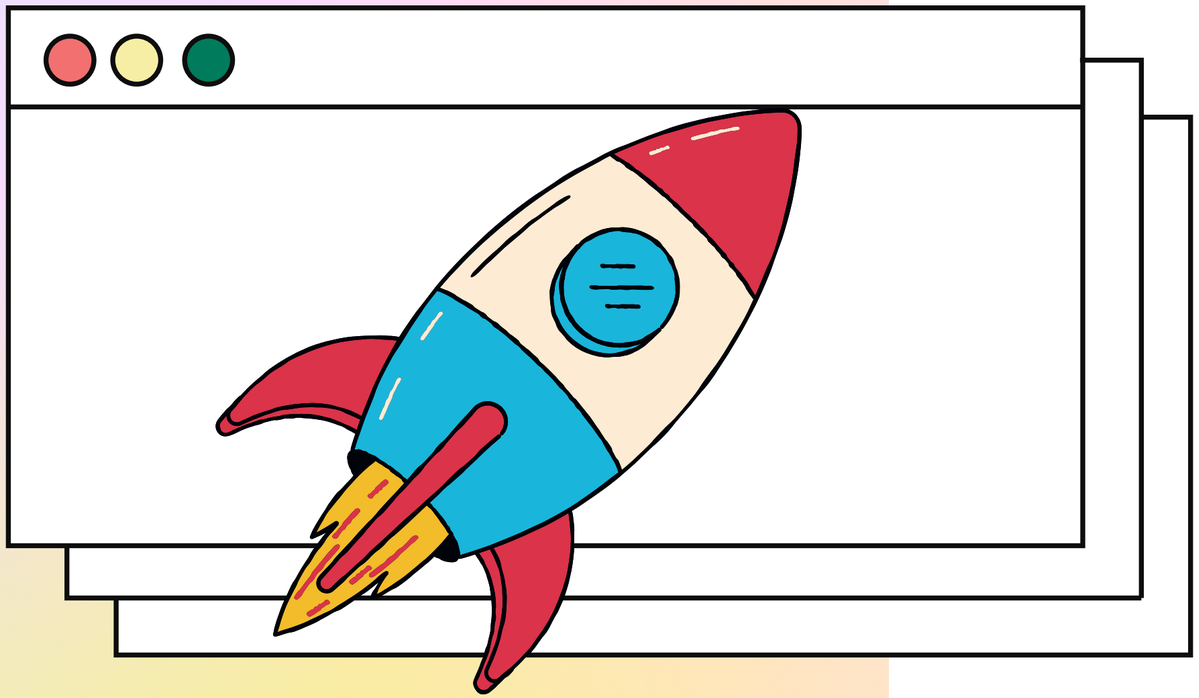
DYNAMIC SKIES

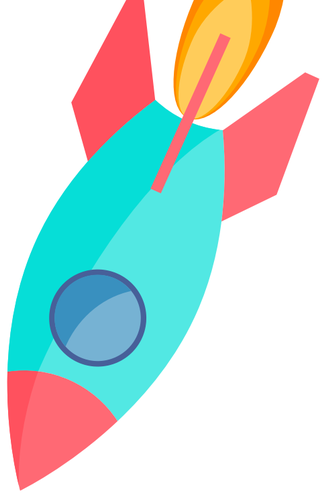
Seiring game berjalan, latar belakang langit akan berubah dari pagi hingga ke malam hari

DYNAMIC SKIES

```css
.game-holder {
    ...
    animation: 50s ease-in-out 0.5s bg-animation infinite alternate;
}

@keyframes bg-animation {
  0% {
    background: #070b34;
  }
  50% {
    background: blue;
  }
  100% {
    background: #855988;
  }
}
```

ROCKET MUMBUL





LINK PENTING

PROJECT

https://rocket-mumbul.vercel.app/

GITHUB

https://github.com/kurniacf/fp-grafkom-kelompok-4

VIDEO DEMO

https://youtu.be/W5ZV0EmXI98.

REFERENSI

https://codepen.io/carrot-e/pen/WGZJBe

THANK YOU