**LAPORAN FINAL PROJECT**

**GRAFIKA KOMPUTER**

**Untuk memenuhi mata kuliah Grafika Komputer**

**“Animasi Kapal Berlayar”**



**Disusun Oleh:**

**Balya Ahmad Waffa 21104410052**

**Moh. Hafid Nur Firmansyah 21104410062**

**Niko Riant Cahyono 21104410083**

**PRODI TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**UNIVERSITAS ISLAM BALITAR**

**2023**

1. **Latar Belakang**

Banyaknya ditemukan adanya pembuatan animasi atau gambar bergerak yang mendorong untuk mencari tau bagaimana membuatnya,kenapa dia menjadi semenarik itu dengan memahami aplikasi yang mungkin digunakan.

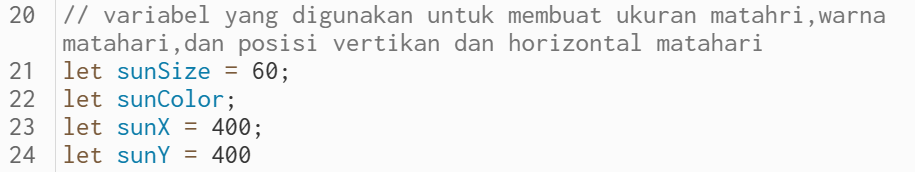
P5.js adalah javascipt library yang digunakan untuk membuat sketch, gambar, desain, di web canvas. P5.js merupakan implementasi dari processing khusus untuk web. Processing sendiri adalah software sekaligus pemograman untuk visual art, prototype, dan animasi untuk pembelajaran.

Tema “Perahu Berlayar” ini diambil karena untuk memberikan edukasi dan motivasi untuk berkarnya dengan membuat animasi sederhana. Selain Itu animasi untuk mengigatkan kita bahwa negara Indonesia merupakan Negara kepulauan dan maritime yang wilayahnya mayoritas adalah lautan.

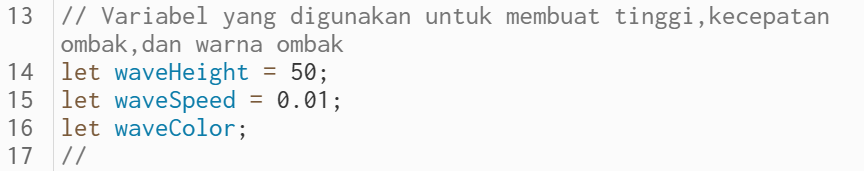
1. **Tujuan**

Pembuatan animasi Ini bertujuan untuk meningkatkan minat membuat animasi dan menjadi media yang efektif dalam berkarya

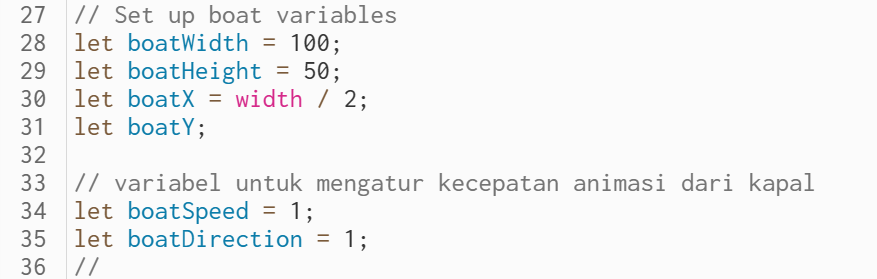
1. **Manfaat**
2. Menginpirasi dan menghibur
3. Hiburan dan daya Tarik visual
4. Ekspresi Kreatif dan imajinatif
5. **Animasi**
6. **Sourcecode Komponen setiap objek**
7. Matahari (Sun) - Representasi matahari pada animasi:
   * Posisi Vertikal dan Horizontal (X dan Y) - (sunX, sunY)
   * Ukuran (Diameter) - sunSize
   * Warna (RGB) – sunColor



1. Ombak (Waves) - Representasi ombak pada animasi:
   * Tinggi Ombak - waveHeight
   * Kecepatan Ombak - waveSpeed
   * Warna Ombak (RGB) – waveColor

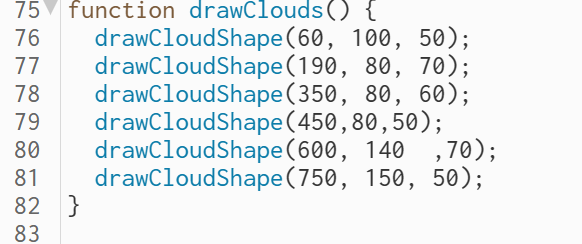


1. Kapal (Boat) - Representasi kapal pada animasi:
   * Posisi Vertikal dan Horizontal (X dan Y) - (boatX, boatY)
   * Lebar dan Tinggi Kapal - boatWidth, boatHeight
   * Kecepatan Animasi Kapal - boatSpeed dan boatDirection (arah pergerakan)

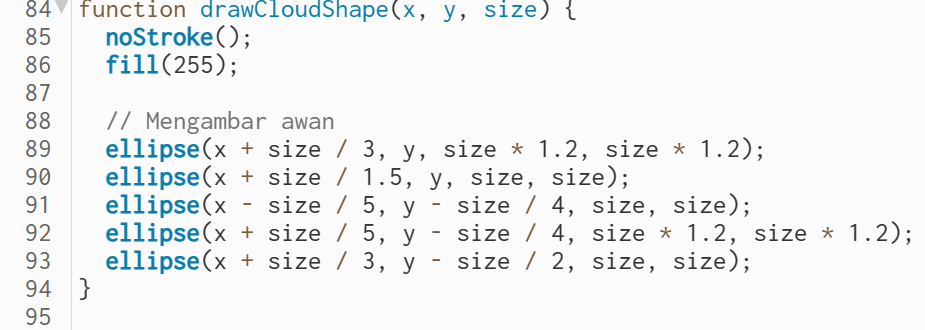


Ketiga objek tersebut diimplementasikan sebagai instance dari class-class yang telah didefinisikan dalam kode. Setiap objek memiliki constructor yang menerima parameter dan memiliki method display() untuk menggambar objek tersebut di layar.

1. **Sourcecode Fungsi-fungsi yang digunakan**
2. **drawClouds()**: Fungsi ini digunakan untuk menggambar awan pada animasi. Fungsi ini memanggil fungsi **drawCloudShape()** untuk menggambar bentuk-bentuk awan.



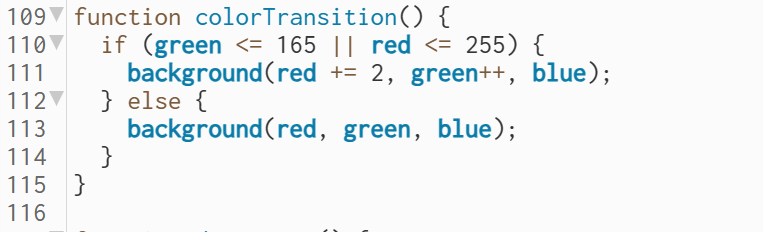
1. **drawCloudShape(x, y, size)**: Fungsi ini digunakan untuk menggambar satu bentuk awan dengan posisi dan ukuran tertentu.



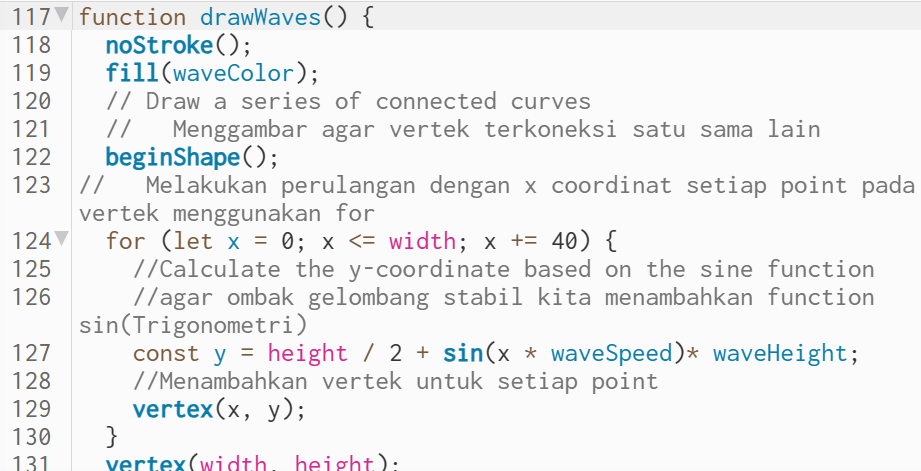
1. **drawSun()**: Fungsi ini digunakan untuk menggambar matahari. Posisi matahari berubah secara vertikal hingga mencapai posisi akhirnya.



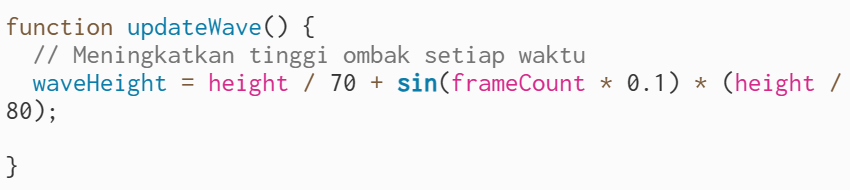
1. **colorTransition()**: Fungsi ini digunakan untuk mengatur transisi warna latar belakang dari hitam ke kuning pada saat matahari terbit.



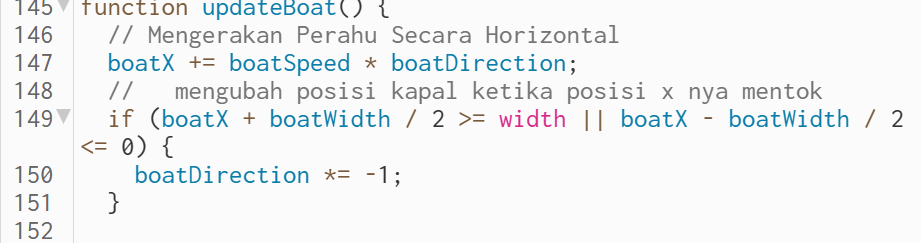
1. **drawWaves()**: Fungsi ini digunakan untuk menggambar gelombang ombak pada animasi. Gelombang dibuat dengan menggunakan fungsi sin(x) untuk menggerakkan titik-titik verteks.



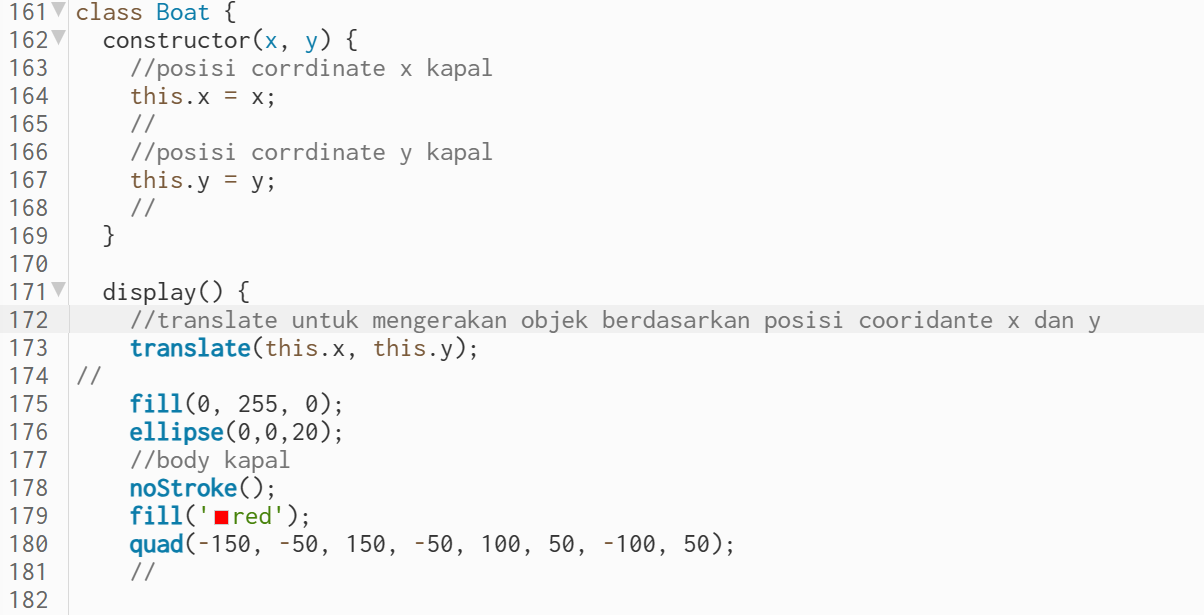
1. **updateWave()**: Fungsi ini digunakan untuk memperbarui tinggi ombak setiap kali animasi diperbarui.



1. **updateBoat()**: Fungsi ini digunakan untuk memperbarui posisi horizontal kapal pada animasi. Kapal bergerak dari sisi kiri ke sisi kanan layar dan kembali lagi dengan kecepatan yang telah ditentukan.



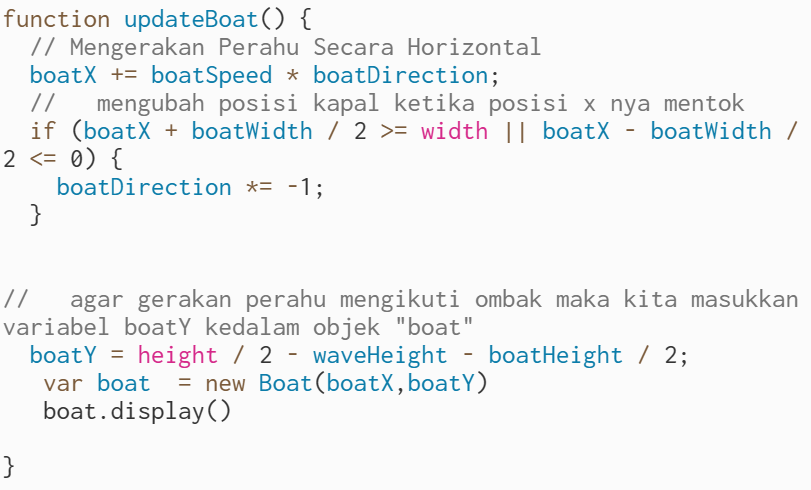
Selain fungsi-fungsi tersebut, terdapat juga class **Boat** yang digunakan untuk mendefinisikan objek kapal. Class ini memiliki constructor dan method display() untuk menggambar kapal di layar.



1. **Sourcecode Kontrol yang digunakan**

Untuk mengontrol pergerakan animasi berikut adalah beberpa fungsi yang kami gunakan

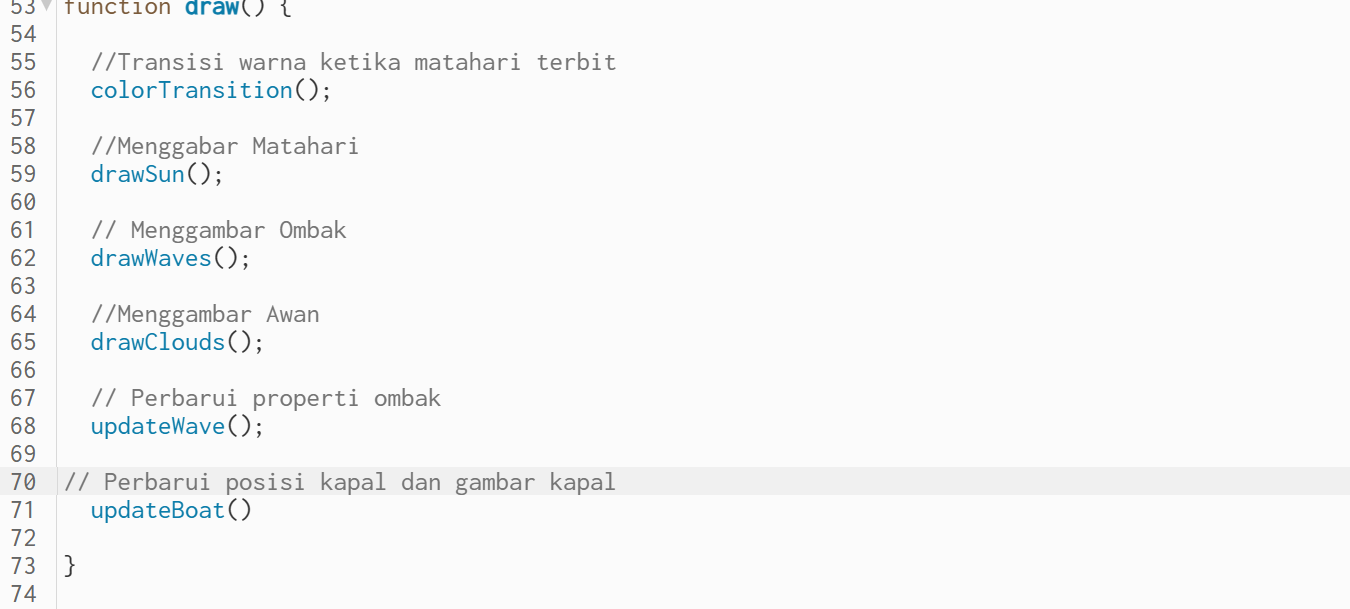
1. Mengontrol pergerkan Kapal



Jika posisi horizontal kapal (boatX) mencapai batas kanan atau kiri layar (lebih besar dari lebar layar atau kurang dari 0), variabel boatDirection akan dikalikan dengan -1, sehingga mengubah arah gerakan kapal. Hal ini menyebabkan kapal bergerak bolak-balik antara batas kiri dan kanan layar.

Posisi vertikal kapal (boatY) diatur sedemikian rupa agar kapal selalu berada di atas ombak. Variabel boatY dihitung dengan mengambil setengah tinggi layar (height / 2), dikurangi dengan tinggi ombak (waveHeight), dan dikurangi dengan setengah tinggi kapal (boatHeight / 2).

1. **Sourcecode Fungsi animasi bergerknya**
   * + 1. fungsi draw() berfungsi untuk melakukan perulangan yang berjalan terus menerus dan menggambar elemen-elemen animasi pada setiap frame.Penjelasan sudah diatas

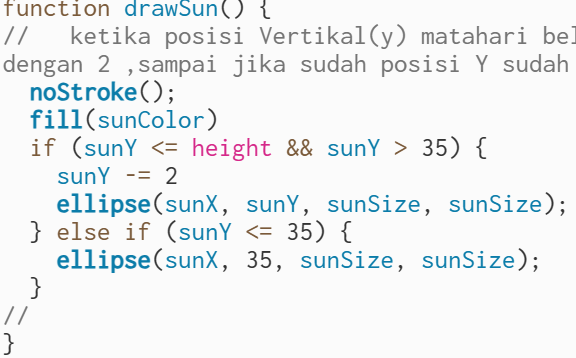


1. **Sourcecode Durasi Animasi**
2. waveSpeed: Variabel waveSpeed digunakan untuk mengatur kecepatan perubahan gelombang ombak pada fungsi updateWave(). Semakin besar nilai waveSpeed, semakin cepat perubahan ombak, sehingga menghasilkan animasi yang lebih cepat.





Posisi vertikal matahari (sunY) berubah dari atas ke bawah untuk menggambarkan terbitnya matahari. Posisi matahari diupdate setiap frame menggunakan kondisi terkait (sunY -= 2) di dalam fungsi drawSun(). Durasi animasi tergantung pada seberapa cepat posisi matahari berubah menuju posisi akhirnya



1. **Progress Pengerjaan**

Progres Minggu Pertama(Animasi Matahari)

1. Progres Minggu Pertama(Animasi Matahari)
   1. Tampilan Program

****

****

* 1. **Code Program**

let red=0;

let green=0;

let blue=0;

let sunColor;

let sun=400;

function setup(){

createCanvas(1000, 400);

}

function draw(){

background(red+=3, green, blue);

drawSun();

}

function drawSun(){

noStroke();

fill(255, 255,0,100);

if(sun>100){

circle(500, sun--, 90);

}

else if(sun==100){

circle(500, sun, 90);

}

noStroke();

fill(255, 255, 0, 100);

if(sun>144){

circle(500, sun--, 90);

}

else if(sun==144){

circle(500, sun, 90);

}

}

**Penjelasan Program**

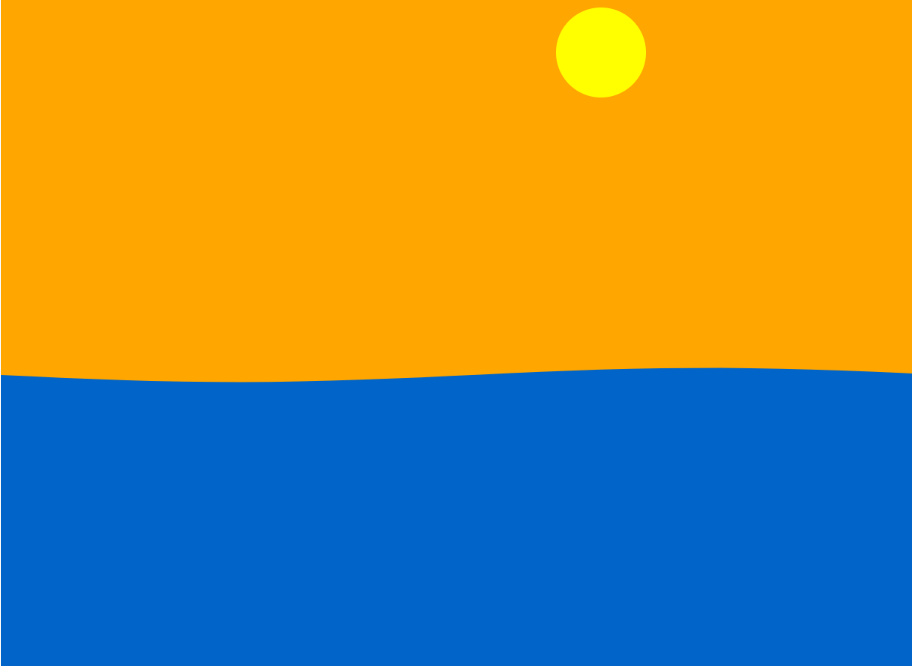
Program diatas membuat sebuah tampilan canvas dengan ukuran 1000

piksel lebar dan 400 piksel tinggi. Kemudian terdapat variabel `red`, `green`, dan `blue` digunakan untuk mengatur warna latar belakang canvas. Pada setiap iterasi `draw()`, nilai variabel `red` bertambah 3 sedangkan `green` dan `blue` tetap.Variabel `sun` memiliki nilai awal 400.

Untuk menggambar matahari disini kami membuat suatu fungsi `drawSun()` yang digunakan untuk menggambar matahari di tengah kanvas dengan koordinat x tetap pada 500, sementara nilai koordinat y dari matahari dikurangi setiap kali fungsi ini dipanggil. Fungsi `drawSun()` menggambar dua lingkaran. Lingkaran pertama memiliki warna fill kuning dengan transparansi 100, dan ukuran yang berkurang setiap kali fungsi ini dipanggil (nilai `sun` dikurangi). Lingkaran kedua memiliki warna fill yang sama dan ukuran tetap, dan akan digambar ketika nilai `sun` mencapai 100 dan 144.Secara keseluruhan, program diatas akan menghasilkan animasi latar belakang canvas yang berubah warna dan menggambar matahari yang semakin menurun posisinya di tengah canvas.

1. **Progres Minggu Kedua (Animasi Laut)**
   1. **Tampilan Program**

****

****

* 1. **Code Program**

const width = 800;

const height = 500;

// Background Color

let red=0;

let green=0;

let blue=0;

// Set up wave variables

let waveHeight = 50;

let waveSpeed = 0.01;

let waveColor;

// Set up sun variables

let sunSize = 60;

let sunColor;

let sunX = 400;

// let sunY = height / 8;

let sunY = 400

// Set up boat variables

let boatWidth = 100;

let boatHeight = 50;

let boatX = width / 2;

let boatY;

// Set up boat animation variables

let boatSpeed = 1;

let boatDirection = 1;

function setup() {

createCanvas(width, height);

waveColor = color(0, 100, 200); // Set wave color to blue

sunColor = color(255, 255, 0); // Set sun color to yellow

boatY = height / 2 - waveHeight - boatHeight / 2; // Position the boat on top of the waves

}

function draw() {

//color Transision

colorTransisiom()

// Draw sun

drawSun();

// Draw waves

drawWaves();

// Update wave properties

updateWave();

}

function drawCloud(){

noStroke();

fill()

}

function drawSun() {

noStroke();

fill(sunColor)

if(sunY <= height && sunY >35 ){

sunY -=2

ellipse(sunX, sunY, sunSize, sunSize);

}else if(sunY <= 35) {

ellipse(sunX, 35, sunSize, sunSize);

}

}

function colorTransisiom(){

if(green <=165 || red <=255){

background(red+=2,green++, blue);

}else {

background(red,green, blue);

}

}

function drawWaves() {

noStroke();

fill(waveColor);

// Draw a series of connected curves

beginShape();

// Iterate over the x-coordinates of each point

for (let x = 0; x <= width; x += 20) {

// Calculate the y-coordinate based on the sine function

const y = height / 2 + sin(x \* waveSpeed) \* waveHeight;

// Add a vertex for each point

vertex(x,y);

}

// Add the last two vertices to complete the shape

vertex(width, height);

vertex(0, height);

endShape(CLOSE);

}

function updateWave() {

// Adjust the wave height over time

waveHeight = height / 70 + sin(frameCount \* 0.1) \* (height / 80);

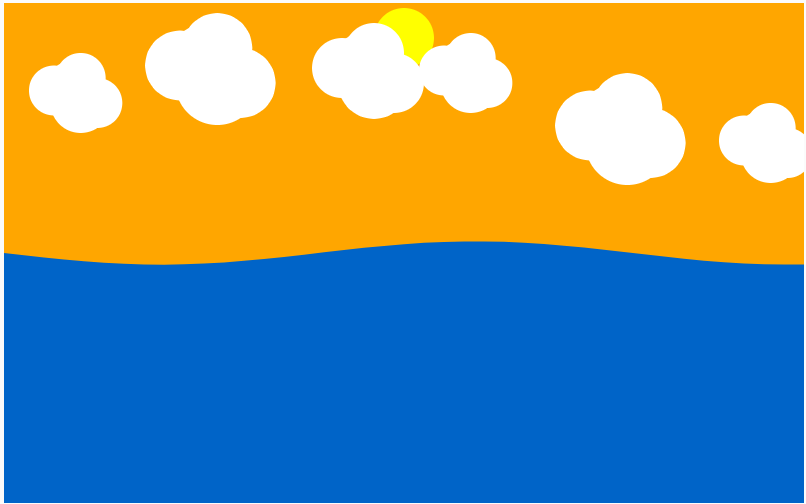
}

* 1. **Penjelasan Program**

Dalam membuat ombak kami menambahkan fungsi drawWaves yang berfungsi membuat gelombang dan fungsi updateWaves untuk untuk menentukan tinggi gelombang.dalam fungsi drawWaves noStroke() digunakan untuk menghilangkan garis tepi pada gambar gelombang.kemudian terdapat fill(waveColor) mengatur warna isian gelombang sesuai dengan nilai yang disimpan dalam variabel waveColor.beginShape() digunakan untuk memulai pembentukan poligon yang akan digambar.Di dalam loop for, kita melakukan iterasi terhadap nilai x yang dimulai dari 0 hingga mencapai nilai width, dengan kenaikan sebesar 20 pada setiap iterasi (x += 20).

Di dalam loop, kita menghitung nilai y (koordinat y) berdasarkan fungsi sinus dengan menggunakan rumus height / 2 + sin(x \* waveSpeed) \* waveHeight. Fungsi sinus digunakan untuk menciptakan efek gelombang yang berubah-ubah. Nilai x dikalikan dengan waveSpeed untuk mengatur kecepatan perubahan gelombang, dan hasilnya dikalikan dengan waveHeight untuk mengatur tinggi gelombang.Selanjutnya, kita menambahkan sebuah vertex (titik) pada poligon dengan menggunakan fungsi vertex(x, y). Setiap titik dalam loop for akan ditambahkan ke poligon.Setelah loop selesai, kita tambahkan dua vertex terakhir untuk melengkapi bentuk poligon. Vertex pertama adalah (width, height), yang berada di sudut kanan bawah kanvas. Vertex kedua adalah (0, height), yang berada di sudut kiri bawah kanvas.

1. **Proses Minggu ke-3 (Animasi Awan)**
   * + - 1. **Tampilan Program**

****

* + - * 1. **Source Code:**

const width = 800;

const height = 500;

// Background Color

let red=0;

let green=0;

let blue=0;

// Set up wave variables

let waveHeight = 50;

let waveSpeed = 0.01;

let waveColor;

// Set up sun variables

let sunSize = 60;

let sunColor;

let sunX = 400;

// let sunY = height / 8;

let sunY = 400

// Set up boat variables

let boatWidth = 100;

let boatHeight = 50;

let boatX = width / 2;

let boatY;

// Set up boat animation variables

let boatSpeed = 1;

let boatDirection = 1;

function setup() {

createCanvas(width, height);

waveColor = color(0, 100, 200); // Set wave color to blue

sunColor = color(255, 255, 0); // Set sun color to yellow

boatY = height / 2 - waveHeight - boatHeight / 2; // Position the boat on top of the waves

}

function draw() {

//color Transition

colorTransition();

// Draw sun

drawSun();

// Draw waves

drawWaves();

// Draw clouds

drawClouds();

// Update wave properties

updateWave();

}

function drawClouds() {

drawCloudShape(60, 100, 50);

drawCloudShape(190, 80, 70);

drawCloudShape(350, 80, 60);

drawCloudShape(450,80,50);

drawCloudShape(600, 140 ,70);

drawCloudShape(750, 150, 50);

}

function drawCloudShape(x, y, size) {

noStroke();

fill(255); // Set the cloud color to white

// Draw cloud shape using ellipses

//ellipse(x, y, size, size);

ellipse(x + size / 3, y, size \* 1.2, size \* 1.2);

ellipse(x + size / 1.5, y, size, size);

ellipse(x - size / 5, y - size / 4, size, size);

ellipse(x + size / 5, y - size / 4, size \* 1.2, size \* 1.2);

ellipse(x + size / 3, y - size / 2, size, size);

}

function drawSun() {

noStroke();

fill(sunColor)

if (sunY <= height && sunY > 35) {

sunY -= 2

ellipse(sunX, sunY, sunSize, sunSize);

} else if (sunY <= 35) {

ellipse(sunX, 35, sunSize, sunSize);

}

}

function colorTransition() {

if (green <= 165 || red <= 255) {

background(red += 2, green++, blue);

} else {

background(red, green, blue);

}

}

function drawWaves() {

noStroke();

fill(waveColor);

// Draw a series of connected curves

beginShape();

// Iterate over the x-coordinates of each point

for (let x = 0; x <= width; x += 20) {

// Calculate the y-coordinate based on the sine function

const y = height / 2 + sin(x \* waveSpeed) \* waveHeight;

// Add a vertex for each point

vertex(x, y);

}

// Add the last two vertices to complete the shape

vertex(width, height);

vertex(0, height);

endShape(CLOSE);

}

function updateWave() {

// Adjust the wave height over time

waveHeight = height / 70 + sin(frameCount \* 0.1) \* (height / 80);

}

**c. Penjelasan Program**

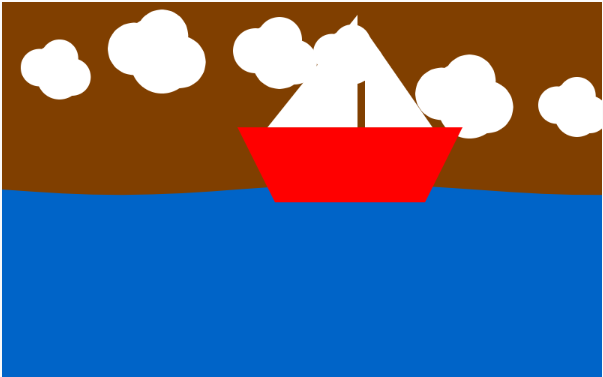
Fungsi drawClouds(): Fungsi ini dipanggil di dalam fungsi draw() untuk menggambar awan-awan. Di dalam fungsi ini, kita memanggil fungsi drawCloudShape() beberapa kali dengan parameter yang berbeda untuk menggambar beberapa awan dengan ukuran dan posisi yang berbeda.

Fungsi drawCloudShape(): Fungsi ini digunakan untuk menggambar bentuk awan menggunakan elips-ellips. Setiap elips mewakili bagian dari awan. Dalam fungsi ini, kita menggunakan fungsi ellipse() untuk menggambar elips dengan ukuran dan posisi yang ditentukan. Dalam contoh di atas, enam elips digunakan untuk membentuk bentuk awan, tetapi Anda dapat menyesuaikan jumlah dan posisi elips sesuai keinginan Anda.

Setiap elips dalam drawCloudShape() diberi warna putih dengan menggunakan fungsi fill(255). Anda juga dapat mengubah warna awan sesuai keinginan dengan mengubah nilai warna dalam fungsi fill().

Anda dapat menambahkan atau mengubah posisi dan ukuran elips dalam drawCloudShape() untuk menciptakan bentuk awan yang berbeda. Misalnya, dengan menyesuaikan koordinat dan ukuran elips, Anda dapat membuat awan yang lebih besar atau lebih kecil, dengan bentuk yang lebih kompleks, atau menggabungkan warna lain untuk memberikan efek shading pada awan.

1. **Proses Minggu ke-4 (Finishinng dan pembuatan Perahu)**
   1. **Tampilan**

****

* 1. **Source Code**

// mengatur tinggi dan lebar canvas

const width = 800;

const height = 500;

// =====================//

// Background Color

let red=0;

let green=0;

let blue=0;

//

// Variabel yang digunakan untuk membuat tinggi,kecepatan ombak,dan warna ombak

let waveHeight = 50;

let waveSpeed = 0.01;

let waveColor;

//

// Variabel yang digunakan untuk membuat tinggi,kecepatan ombak,dan warna ombak

let waveHeight = 50;

let waveSpeed = 0.01;

let waveColor;

//

// variabel yang digunakan untuk membuat ukuran matahri,warna matahari,dan posisi vertikan dan horizontal matahari

let sunSize = 60;

let sunColor;

let sunX = 400;

let sunY = 400

//

// Set up boat variables

let boatWidth = 100;

let boatHeight = 50;

let boatX = width / 2;

let boatY;

// variabel untuk mengatur kecepatan animasi dari kapal

let boatSpeed = 1;

let boatDirection = 1;

//

function setup() {

createCanvas(width, height);

// Mengatur Warna Ombaknya biru

waveColor = color(0, 100, 200);

//

// Mengatur Warna Matahari

sunColor = color(255, 255, 0);

//

//Memberikan Posisi agar Kapal berada diatas Ombak

boatY = height / 2 - waveHeight - boatHeight / 2;

//

}

function draw() {

//Transisi warna ketika matahari terbit

colorTransition();

//Menggabar Matahari

drawSun();

// Menggambar Ombak

drawWaves();

//Menggambar Awan

drawClouds();

// Perbarui properti ombak

updateWave();

// Perbarui posisi kapal dan gambar kapal

updateBoat()

}

// Menggambar Ombak

drawWaves();

//Menggambar Awan

drawClouds();

// Perbarui properti ombak

updateWave();

// Perbarui posisi kapal dan gambar kapal

updateBoat()

}

function drawClouds() {

drawCloudShape(60, 100, 50);

drawCloudShape(190, 80, 70);

drawCloudShape(350, 80, 60);

drawCloudShape(450,80,50);

drawCloudShape(600, 140 ,70);

drawCloudShape(750, 150, 50);

}

function drawCloudShape(x, y, size) {

noStroke();

fill(255);

// Mengambar awan

ellipse(x + size / 3, y, size \* 1.2, size \* 1.2);

ellipse(x + size / 1.5, y, size, size);

ellipse(x - size / 5, y - size / 4, size, size);

ellipse(x + size / 5, y - size / 4, size \* 1.2, size \* 1.2);

ellipse(x + size / 3, y - size / 2, size, size);

}

function drawSun() {

// ketika posisi Vertikal(y) matahari belum mencapai 35 maka akan terus terus dikurangi dengan 2 ,sampai jika sudah posisi Y sudah 35 maka animasi akan berhenti

noStroke();

fill(sunColor)

if (sunY <= height && sunY > 35) {

sunY -= 2

ellipse(sunX, sunY, sunSize, sunSize);

} else if (sunY <= 35) {

ellipse(sunX, 35, sunSize, sunSize);

}

//

}

function colorTransition() {

if (green <= 165 || red <= 255) {

background(red += 2, green++, blue);

} else {

background(red, green, blue);

}

}

function drawWaves() {

noStroke();

fill(waveColor);

// Draw a series of connected curves

// Menggambar agar vertek terkoneksi satu sama lain

beginShape();

// Melakukan perulangan dengan x coordinat setiap point pada vertek menggunakan for

for (let x = 0; x <= width; x += 40) {

//Calculate the y-coordinate based on the sine function

//agar ombak gelombang stabil kita menambahkan function sin(Trigonometri)

const y = height / 2 + sin(x \* waveSpeed)\* waveHeight;

//Menambahkan vertek untuk setiap point

vertex(x, y);

}

vertex(width, height);

vertex(0, height);

endShape(CLOSE);

}

function updateWave() {

// Meningkatkan tinggi ombak setiap waktu

waveHeight = height / 70 + sin(frameCount \* 0.1) \* (height / 80);

}

// function agar kapal bergerak

function updateBoat() {

// Mengerakan Perahu Secara Horizontal

boatX += boatSpeed \* boatDirection;

// mengubah posisi kapal ketika posisi x nya mentok

if (boatX + boatWidth / 2 >= width || boatX - boatWidth / 2 <= 0) {

boatDirection \*= -1;

}

// agar gerakan perahu mengikuti ombak maka kita masukkan variabel boatY kedalam objek "boat"

boatY = height / 2 - waveHeight - boatHeight / 2;

var boat = new Boat(boatX,boatY)

boat.display()

}

class Boat {

constructor(x, y) {

//posisi corrdinate x kapal

this.x = x;

//

//posisi corrdinate y kapal

this.y = y;

//

}

boat.display()

}

class Boat {

constructor(x, y) {

//posisi corrdinate x kapal

this.x = x;

//

//posisi corrdinate y kapal

this.y = y;

//

}

display() {

//translate untuk mengerakan objek berdasarkan posisi cooridante x dan y

translate(this.x, this.y);

//

fill(0, 255, 0);

ellipse(0,0,20);

//body kapal

noStroke();

fill('red');

quad(-150, -50, 150, -50, 100, 50, -100, 50);

//

//layer kapal 1

noStroke();

fill('white');

triangle(-110, -50, 10, -50, 10, -200);

//

noStroke();

fill('white');

triangle(110, -50, 20, -50, 20, -180);

}

}