Judul Games: Meteor Destroyer

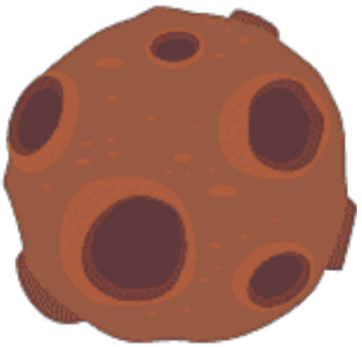
Anggota: 1. Leonardo Nickholas Andrianto/ C14210206
2. Alfons Pramudita Kurnaiwan Cahyono/ C14210237
3. Estavanie Audrey Tjahyono/ C14210148

Pembagian Pekerjaan:

1. Leonardo Nicholas Andrianto/ C14210206:
   - Membuat Class GameScreen, TryAgainScreen, dan Obstacle (serta aplikasi polymorphnya ke class WeakMeteor dan StrongMeteor)
   - Pencarian assets
2. Alfons Pramudita Kurniawan Cahyono/ C14210237
   - Membuat class MainMenuScreen dan membantu dalam coding di bagian GameScreen
   - Membuat Design mainMenu dan tryagain screen
3. Estavanie Audrey Tjahyono/ C14210148
   - Membuat class Meteor dan membantu coding di bagian GameScreen
   - Pencarian assets

1. DESKRIPSI GAMES:

   Untuk gamesnya sendiri diawali dengan screen MainMenu terlebih dahulu kemudian user diminta untuk mengklik mana saja bagian layar game, maka game akan terstar. Untuk gamenya, user nantinya akan berperan sebagai pesawat dimana pada games tersebut, pesawatnya memiliki health dimana jika healthnya == 0, maka akan game over. Untuk mencetak skor sendiri pesawat harus menghindari neon-neon yang ada dan melakukan tembakan ke meteor yang ada. Untuk meteor sendiri di game tersebut terdapat 2 jenis, yaitu:



Perbedaan meteor tersebut terdapat pada kekuatannya. Jika meteor yang berapi, maka user harus menembaknya 2 kali. Sedangkan yang tanpa api, user hanya perlu menembaknya sekali.

2. ASSETS

ASSET PLANE

ASSET STRONG METEOR

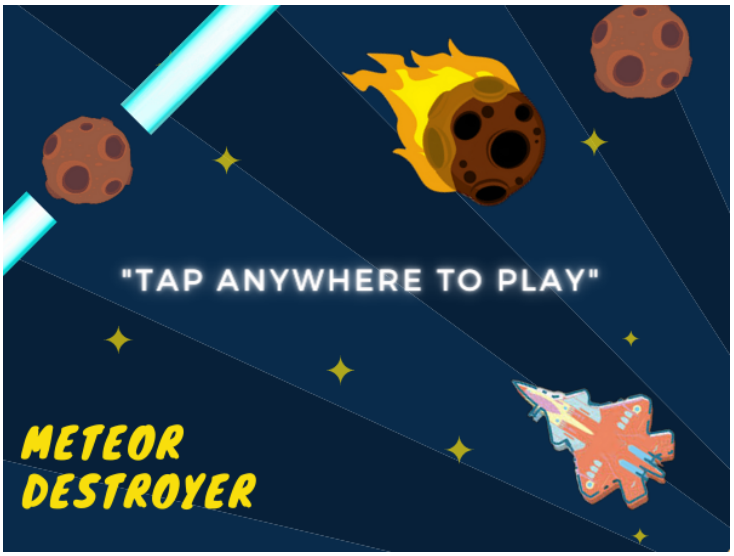
ASSET WEAK METEOR

ASSET PELURU PLANE

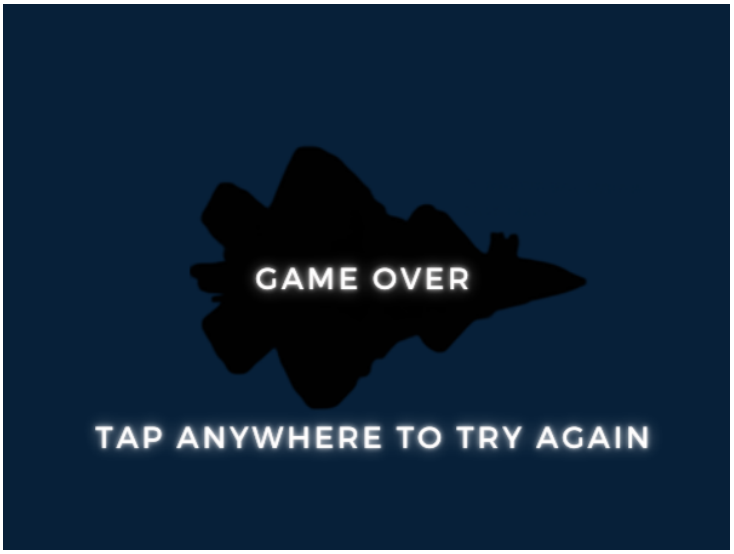
ASSET NEON BIRU

ASSET BATAS KIRI KANAN

ASSET BACKGROUND ON GAME



ASSET MAIN MENU



ASSET GAME OVER

ASSET SOUND DAN MUSIC ADA DI FILE ZIP

PENJELASAN PROGRAM

```java
package com.mygdx.game;

import com.badlogic.gdx.backends.lwjgl3.Lwjgl3Application;
import com.badlogic.gdx.backends.lwjgl3.Lwjgl3ApplicationConfiguration;

// Please note that on macOS your application needs to be started with the -XstartOnFirstThread JVM argument
public class DesktopLauncher {
    public static void main (String[] arg) {
        Lwjgl3ApplicationConfiguration config = new Lwjgl3ApplicationConfiguration();
        config.setForegroundFPS(60);
        config.setWindowedMode( width: 600, height: 450);
        config.setTitle("Meteor Destroyer");
        new Lwjgl3Application(new Meteor(), config);
    }
}
```

Hal pertama yang akan kami jelaskan adalah tentang class DesktopLauncher dimana class ini adalah class main tempat program dijalankan. Seperti setup awal kita memperlukan class Lwjgl3ApplicationConfiguration untuk mengeset FPS, ukuran layar, dan mengeset judul gamenya. Untuk foreground FPSnya sendiri kita menggunakan 60, dan untuk ukurannya width nya 600 dan heightnya 450. Kemudian untuk title gamenya sendiri adalah Meteor destroyer. Kemudian kita masuk ke class perrtama yang akan dijalankan yaitu class Meteor.

```java
import com.badlogic.gdx.Game;
import com.badlogic.gdx.graphics.g2d.BitmapFont;
import com.badlogic.gdx.graphics.g2d.SpriteBatch;

public class Meteor extends Game {
    public SpriteBatch batch;
    MainMenuScreen mainmenu;
    BitmapFont font;
    GameScreen gameScreen;
    TryAgainScreen againScreen;
```

Berikut adalah class Meteor dimana class tersebut adalah inheritance dari class Game. dan attribute yang dibuat dalam class tersebut adalah objek SpriteBatch, objek MainMenuScreen, objek BitmapFont, objek GameScreen, dan objek TryAgainScreen.

Dan kita melakukan pemimporan library-library yang dibutuhkan, yaitu dari library
com.badlogic.gdx.Game,
com.badlogic.gdx.graphics.g2d.BitmapFont,
com.badlogic.gdx.graphics.g2d.SpriteBatch

```
@Override
public void create () {
    batch= new SpriteBatch();
    font= new BitmapFont();
    mainmenu= new MainMenuScreen( game: this);
    gameScreen= new GameScreen(this);
    againScreen= new TryAgainScreen( meteor: this, Score: 0);
    this.setScreen(mainmenu);
}
```

Setelah itu ada method create dimana method tersebut adalah Override dari method create yang ada di class Game. Kemudian kita melakukan inisialisasi atau new pada objek batch, font, mainmenu, gamescreen, dan TryAgainScreen (0 maksudnya score yang nantinya akan ditampilkan di tryAgainScreen). Kemudian, karena awal-awal game kita akan masuk pada mainmenu screen maka kita mengeset screennya menjadi mainmenu

```
@Override
public void dispose () {
    super.dispose();
    batch.dispose();
    font.dispose();
    mainmenu.dispose();
    gameScreen.dispose();
    againScreen.dispose();
}
```

Kemudian, ada method dispose dimana bentuk override dari method create yang ada di class Game. disini kita melakukan dispose attribute-attribute yang sudah kita buat tadi yaitu batch, font, mainmenu, gamescreen, dan TryAgainScreen.

```
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.Screen;
import com.badlogic.gdx.graphics.OrthographicCamera;
import com.badlogic.gdx.graphics.Texture;



public class MainMenuScreen implements Screen {
    final Meteor game;
    OrthographicCamera camera;
    Texture mainMenu;
```

Kemudian kita akan menjelaskan class MainMenuScreen dimana class tersebut merupakan implement dari interface Screen. Untuk attribute yang ada yaitu objek dari class Meteor yang diberi nama game, objek dari class OrthographicCamera yang diberi nama camera, dan Texture yang merupakan design gambar dari MainMenu yang kita buat. Kemudian untuk class yang kira import sendiri ada
`com.badlogic.gdx.Gdx`
`com.badlogic.gdx.Screen`
`com.badlogic.gdx.graphics.OrthographicCamera`
`com.badlogic.gdx.utils.ScreenUtils`
`com.badlogic.gdx.graphics.Texture;`

```
public MainMenuScreen(Meteor game){
    this.game=game;
    camera= new OrthographicCamera();

    camera.setToOrtho( yDown: false, viewportWidth: 600, viewportHeight: 450);
    mainMenu= new Texture(Gdx.files.internal( path: "mainMenu.png"));
}
```

Kemudian kita membuat constructor dengan parameternya adalah Meteor. Kemudian kita set game dengan nilai yang ada di nilai parameter. Kemudian kita insialisasi camera. Kemudian kita set posisi kamera kita yaitu sesuai dengan Width dan Height game kita. Kemudian, kita set untuk texture nya, diambil dari files internal.

```
@Override
public void render(float delta) {
    camera.update();
    game.batch.setProjectionMatrix(camera.combined);
    game.batch.begin();
    game.batch.draw(mainMenu, x: 0, y: 0, width: 600, height: 450);
    game.batch.end();

    if(Gdx.input.isTouched()){
        game.setScreen(game.gameScreen);
        dispose();
    }
}
```

Pada awalnya, kita melakukan pengudpatean camera dan setprojectionmatrix. Kemudian kita begin batch yang ada di class Game tadi yang kita buat di kelas MainMenuScreen.Kemudian kita lanjutkan untuk menggambar texture mainMenu tadi dengan x dan y 0, dan width heightnya sesuai dengan ukuran layar yaitu 600 dan 450. Kemudian kita membuat if dimana if tersebut akan dilakukan ketika kita mengtouch dimanapun layarnya, sesuai implementasi "Tap anywhere to begin". Jika dipencet, maka kita akan setscreen gamenya menjadi gameScreen.

Sebelum membahas gameScreen, kita akan membahas terlebih dahulu class-class yang kita buat dan akan dipakai pada gameScreen.

```
interface interfacePlane{
    void left();
    void right();
}
```

(PENGGUNAAN KONSEP PBO INTERFACE, OVERRIDE)

Yang pertama kita buat adalah interfacePlane dimana digunakan untuk menulis method gerakan pesawat yang hanya bisa bergerak ke kiri menggunakan method left dan ke kanan menggunakan method right. Kemudian, interface yang kita buat akan diimplementasikan ke dalam class Plane.

```
public class Plane implements interfacePlane {
    private Rectangle plane;
    int health;

    public Plane(Rectangle x, int health){
        setPlane(x);
        this.health=health;
    }
    @Override
    public void left() {
        getPlane().x-= 200 * Gdx.graphics.getDeltaTime();
    }

    @Override
    public void right() {
        getPlane().x+= 200 * Gdx.graphics.getDeltaTime();
    }

    public void setHealth(int health) { this.health = health; }

    public int getHealth() { return health; }

    public void setPlane(Rectangle plane) { this.plane = plane; }

    public Rectangle getPlane() { return plane; }
```

Attribute yang ada di class Plane, kita membuat attribute rectangle yang akan menyimpan posisi x dan y yang ada nantinya dan menyimpan health dari plane yang kita buat sebagai parameter kalah dalam game. Kita akan membuat constructor terlebih dahulu dimana parameter yang diminta adalah rectangle dan integer kemudian di dalam constructor kita akan mengeset rectangle plane kita dengan parameter yang kita minta dan juga healthnya juga sesuai dengan parameter yang diminta. Setelah itu, kita melakukan override pada method yang ada di interfacePlanee tadi. Yang pertama ada method left dimana gerakan yang dilakukan adalah bergerak ke kiri atau posisi x berkurang sebanyak 200 kemudian dikalikan dengan detlatime. Kemudian ada method right dimana gerakan yang dilakukan adalah bergerak ke kanan atau posisi x bertambah sebanyak 200 kemudian dikalikan dengan deltatime. Kemudian kita juga membuat getter setter. Pada class ini kita menggunakan konsep pbo yaitu interface dan overriding dimana kita memanfaatkan interface untuk membuat sebuah method kemudian diimplementasikan dan meng override method yang ada di interface.

```java
public class Obstacle {
    private Rectangle obstacle;
    private int strength;

    public Obstacle(){

    }

    public Obstacle(Rectangle input, int strength){
        setObstacle(input);
        setStrength(strength);
    }

    public Rectangle getObstacle() {
        return obstacle;
    }

    public void setObstacle(Rectangle obstacle) {
        this.obstacle = obstacle;
    }

    public int getStrength() {
        return strength;
    }
```

(PENGGUNAAN KONSEP POLYMORPHISM DAN INHERITANCE)

Kemudian kita,membuat class Obstacle. Dimana kita menyimpan attribute berupa Rectangle yang akan menyimpan posisi x dan y obstacle yang kita buat nanti dan juga ada strength untuk menyimpan kekuatan dari obstacle yang kita buat nanti. Kita juga membuat constructor kosong dan juga constructor dengan parameter yang akan mengset semua attribute. Kita juga membuat getter dan setter tiap attribute yang ada.

```java
public class WeakMeteor extends Obstacle {

    public WeakMeteor(Rectangle input){
        super(input, strength: 50);
    }
}
```

Kemudian kita membuat class WeakMeteor yang merupakan turunan dari class Obstacle yang kita buat tadi. Kemudian kita perlu membuat Constructor untuk mengset rectangle dan strength dari weakmeteor sebesar 50.

```java
public class StrongMeteor extends Obstacle{

    public StrongMeteor(Rectangle input){
        super(input, strength: 100);
    }
}
```

Kemudian, kita juga membuat class Strongmeteor yang juga merupakan turunan dari class Obstacle. Kemudian juga ada constructor yang akan mengset rectangle dan strength sebesar 100.

```java
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.Input;
import com.badlogic.gdx.Screen;
import com.badlogic.gdx.audio.Music;
import com.badlogic.gdx.audio.Sound;
import com.badlogic.gdx.graphics.OrthographicCamera;
import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.math.MathUtils;
import com.badlogic.gdx.utils.Array;
import com.badlogic.gdx.utils.ScreenUtils;
import com.badlogic.gdx.utils.TimeUtils;
import java.awt.*;

public class GameScreen implements Screen {
    Texture planeImg;
    Texture leftLaserImg;
    Texture rightLaserImg;
    Texture peluruPlaneImg;
    Texture blueNeonImg;
    Texture meteorImg;
    Texture meteor2Img;
    Texture backgrounImg;
```

Kemudian, kita membuat class GameScreen yang merupakan implements dari interface Screen. Untuk library yang kita import ada:

```
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.Input;
import com.badlogic.gdx.Screen;
import com.badlogic.gdx.audio.Music;
import com.badlogic.gdx.audio.Sound;
import com.badlogic.gdx.graphics.OrthographicCamera;
import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.math.MathUtils;
import com.badlogic.gdx.utils.Array;
```

```
import com.badlogic.gdx.utils.ScreenUtils;
import com.badlogic.gdx.utils.TimeUtils;
import java.awt.*;
```

Untuk attribute yang dibuat ada Texture yang merupakan gambar yang akan digambar menggunakan batch. Yang pertama ada planeImg yang merupakan gambar pesawat. Kemudian ada leftLaserImg yang akan menyimpan batas kiri dari permainan nantinya. Kemudian rightLaserImg yang akan menyimpan batas kanan dari permainan nantinya. Kemudian, kita juga membuat peluruPlaneImg yang merupakan gambar peluru laser yang nantinya akan ditembakkan ke pesawat. Kemudian, kita juga membuat blueNeonImg yang merupakan gambar rintangan yang akan dilewati. Lalu, kita juga membuat meteorImg yang merupakan gambar dari StrongMeteor kemudian juga membuat meteor2Img yang merupakan gambar dari WeakMeteor. Kemudian kita juga membuat backgroundImg yang merupakan background dari game nantinya.

```
Rectangle plane;
Rectangle leftNeon;
Rectangle rightNeon;
Array<Rectangle> peluruPlane;
Array<Rectangle>blueNeon;
Rectangle meteor;
Rectangle background;

Music planeMusic;
Sound laserSound;
Sound meteorDestroyedSound;
Sound collisionWithLaserSound;
Sound planeMeteor;

OrthographicCamera camera;
```

Kemudian kita juga membuat Rectangle dari tiap texture yang ada yaitu plane, leftneon, rightneon, peluruPlane, blueNeon, meteor, background. Kemudian kita juga mempunyai attribute Music yang merupakan background Music yang nantinya akan di looping. Kemudian kita juga ada Sound, laserSound yang akan keplay ketika laser ditembakkan dan tidak mengenai meteor. Kemudian ada meteorDestroyedSound yang akan keplay ketika meteor hancur. Kemudian ada collisionWithLaserSound yang akan keplay ketika plane collision dengan laser atau neon yang ada. Kemudian, ada planeMeteor yang akan keplay ketika plane

bertabrakan dengan meteor. Kemudian, kita juga membuat OrthographicCamera.

```
int index;
long lastTime;
int randomMeteor;
int countCollision;
int score;

//Class sendiri
Plane input;
Obstacle obstacle;
final Meteor game;
```

Kemudian kita mempunyai attribute index yang akan menjadi index dari array of recatngle dari peluruPlane. Kemudian kita mempunyai attribute lastTime yang akan menyimpan lastime yang digunakan untuk menghitung waktu untuk spawn blue neon dan meteor. Kemudian kita juga menghitung score yang didapat player.
Kemudian kita membuat objek dari class Plane, dari class Obstacle dan juga dari class Meteor.

```
public void createPeluru(){
    Rectangle temp= new Rectangle();
    temp.x= plane.x + 25;
    temp.y=80;
    temp.width=5;
    temp.height=200;
    peluruPlane.add(temp);
}
```

Kemudian kita juga membuat method createPeluru. Dimana langkah pertama yang dilakukan adalah membuat rectangle temp. Kemudian set posisi x nya yaitu posisi x plane ditambah dengan 25 agar pas di tengah pesawat. Kemudian, untuk posisi y nya adalah 80. Untuk widthnya sebesar 5 dan heightnya 200. Kemudian kita add rectangle yang dibuat tadi ke array peluruPlane.

```
public void spawnBlueNeonAndMeteor(){
    Rectangle left=new Rectangle();
    Rectangle right= new Rectangle();
    Rectangle meteor= new Rectangle();
    left.x=  0;
    left.y=500;
    left.width=MathUtils.random(100,450);
    left.height=20;

    right.x=left.x+ left.width+100;
    right.y=500;
    right.width=600-right.x;
    right.height=20;
    blueNeon.set(0,left);
    blueNeon.set(1,right);

    meteor.x=left.x+left.width+30;
    meteor.y=495;
    meteor.width=40;
    meteor.height=40;
    this.meteor=meteor;
    lastTime= TimeUtils.nanoTime();
}
```

Kemudian ada method spawnBlueNeonAndMeteor. Langkah pertama yang dilakukan adalah membuat rectangle left, right, dan meteor. Kemudian kita set untuk posisi x dari left ini yaitu 0, untuk y nya 500. Dan untuk widthnya dirandom antara 100 sampai 450 dan heightnya sebesar 20. Kemudian untuk rectangle right, posisi x nya sendiri sebesar posisi x left ditambah dengan left widthnya dan ditambah 100. Kemudian untuk rectangle meteor, posisi meteornya yaitu left x ditambah left width ditambah 30 sehingga posisi meteor sendiri berada di antara right dan left. Kemudian posisi y nya yaitu 495. Untuk width dan heightnya yaitu 40. Kemudian diset meteornya. Kemudian kita menyimpan last time.

```
public GameScreen(Meteor game){
    this.game=game;
    score=0;
    planeImg= new Texture(Gdx.files.internal( path: "pesawat3.png"));
    leftLaserImg= new Texture(Gdx.files.internal( path: "neonKiri.png"));
    rightLaserImg= new Texture(Gdx.files.internal( path: "neonKanan.png"));
    peluruPlaneImg= new Texture(Gdx.files.internal( path: "peluruRed.png"));
    blueNeonImg= new Texture(Gdx.files.internal( path: "blueNeon.png"));
    meteorImg= new Texture(Gdx.files.internal( path: "meteor.png"));
    meteor2Img= new Texture(Gdx.files.internal( path: "meteorLemah.png"));
    backgrounImg= new Texture(Gdx.files.internal( path: "bintang2.png"));

    planeMusic= Gdx.audio.newMusic(Gdx.files.internal( path: "PlaneSound.wav"));
    planeMusic.setLooping(true);
    planeMusic.play();

    laserSound= Gdx.audio.newSound(Gdx.files.internal( path: "laser.mp3"));
    meteorDestroyedSound= Gdx.audio.newSound(Gdx.files.internal( path: "meteorDestroyed.wav"));
    collisionWithLaserSound= Gdx.audio.newSound(Gdx.files.internal( path: "collisionWithLaser.mp3"));
    planeMeteor= Gdx.audio.newSound(Gdx.files.internal( path: "planeMeteor.wav"));
```

Kemudian, kita juga membuat Constructor yang meminta parameter Meteor untuk mengesit objek dari class Meteor nantinya. Langkah yang dilakukan langsung set objek dari meteor. Kemudian, score kita buat jadi 0. Kemudian kita melakukan pengimporan tiap texture dari file internal dan juga melakukan pengimporan music. Untuk planemusic sendiri kita setloopingnya menjadi true.

```
plane= new Rectangle();
plane.x= 600/2-55/2;
plane.y=70;
plane.width=55;
plane.height=55;

leftNeon= new Rectangle();
leftNeon.x=0;
leftNeon.y=-60;
leftNeon.width=64;
leftNeon.height=700;

rightNeon= new Rectangle();
rightNeon.x=536;
rightNeon.y=-60;
rightNeon.width=64;
rightNeon.height=700;

background= new Rectangle();
background.x=0;
background.y=0;
background.width=600;
background.height=450;
```

Kemudian kita atur untuk posisi awal planenya dengan x sebesar 600/2 - 55/2. Kemudian posisi y nya 70 dan widht dan heightnya sebesar 55. Dan untuk posisi leftneon dengan x nya 0 dan y nya -60 dan width 64 dan heightnya 700. Kemudian, untuk rightneon

kita set posisi x nya adalah 536 dan posisi y nya -60, widthnya 64 dan heightnya 700. Kemudian, untuk backgroudnya sendiri posisi x dan y nya 0 dan width dan heightnya sama dengan besarnya layar yaitu 600 dan 450.

```
peluruPlane= new Array<>();
createPeluru();

meteor= new Rectangle();

blueNeon= new Array<>();
blueNeon.add(null);
blueNeon.add(null);
randomMeteor= MathUtils.random(0,1);
spawnBlueNeonAndMeteor();

if(randomMeteor==0){
    obstacle= new StrongMeteor(meteor);
}else {
    obstacle= new WeakMeteor(meteor);
}
index=1;
camera=new OrthographicCamera();
camera.setToOrtho( yDown: false, viewportWidth: 600, viewportHeight: 450);

//class sendiri
input= new Plane(plane);
```

Kemudian kita melakukan inisialisasi pada Array peluruPlane, kemudian panggil method createPeluru yang kita buat tadi. Kemudian kita melakukan insialisasi pada Rectangle meteor. Kemudian lanjut inisialisasi Array blue neon dan pemberian nilai awal pada index 0 dan 1 adalah null.

(POLYMORPHISM) Kemudian kita melakukan randomMeteor antara 0 atau 1. Jikaa 1, sistem game akan menghasilkan weakmeteor. Jika 0, sistem game akan menghasilkan StrongMeteor. Kemudian, kita set index peluruPlane menjadi 1. Kemudian kita melakukan set kamera dan melakukan inisialisasi pada objek plane yang bernama input dan set healthnya menjadi 150.

```
@Override
public void render(float delta) {
    ScreenUtils.clear( r: 0, g: 0, b: 0,  a: 1);
    camera.update();
    game.batch.setProjectionMatrix(camera.combined);
    game.batch.begin();
    game.batch.draw(backgrounImg, background.x, background.y, background.width, background.height);
    game.font.draw(game.batch,  str: "SCORE: " + score, x: 40, y: 430 );
    game.batch.draw(planeImg,plane.x,plane.y,plane.width,plane.height);
    game.batch.draw(leftLaserImg, leftNeon.x, leftNeon.y, leftNeon.width,leftNeon.height);
    game.batch.draw(rightLaserImg, rightNeon.x, rightNeon.y, rightNeon.width, rightNeon.height);
```

Kemudian pada method render. Awalnya mengclearkan screennya tanpa warna atau hitam karena kita akan menggunakan background. Terus melakukan update camera dan setProjectionMatrix. Kemudian kita memulai batch. Yang paling awal kita gambar adalah background. Kemudian menggambar tulisan score pada posisi x 40 dan y 430. Kemudian menggambar gambar pesawat dan mengambar batas kiri dan batas kanan laser.

```
if(Gdx.input.isKeyJustPressed(Input.Keys.SPACE)) {
    createPeluru();
    Rectangle y= peluruPlane.get(index);
    if(peluruPlane.get(index).intersects(meteor)){
        obstacle.setStrength(obstacle.getStrength()-50);
        if(obstacle.getStrength()<=0){
            meteorDestroyedSound.setVolume( soundId: 0, volume: 20);
            meteorDestroyedSound.play();
            meteor.y=-50;
            if(obstacle instanceof WeakMeteor){
            score+=50;
            }else if(obstacle instanceof StrongMeteor){
            score+=100;
            }
        }
    }else {
        laserSound.play();
    }
    index++;
    game.batch.draw(peluruPlaneImg, y.x, y.y, y.width, y.height);

}
```

Kemudian membuat if ketika tombol space dipencet maka peluruPlane akan dicreate. Kemudian posisi yang dihasilkan itu ditampung pada rectangle y ini. Kemudian membuat percabangan kalo semisal peluru laser yang ditembakkan mengenai meteor maka strength dari meteor itu akan berkurang 50. Kemudian dicek lagi jika strengthnya <= 0, maka play sound meteordestroyed, kemudian hilangka dari layar meteor yang ada dengan set y nya menjadi -50. **(POLYMORPHISM)** Disini kita menggunakan konsep polymorphism dimana jika obstaclenya itu adalah weakmeteor maka scorenya ditambah 50 dan jika strongMeteor maka scorenya ditambah 100. Jika laser itu tidak mengenai apapun maka laserSound itu akan diplay. Kemudian menambah index dan menggambarkan gambar peluru laser.

```
for (Rectangle blue: blueNeon){
    game.batch.draw(blueNeonImg, blue.x,blue.y, blue.width, blue.height);
}
if(randomMeteor==0) {
    game.batch.draw(meteorImg, meteor.x, meteor.y, meteor.width, meteor.height);
}else if(randomMeteor==1){
   game.batch.draw(meteor2Img, meteor.x, meteor.y, meteor.width,meteor.height);
}
game.batch.end();
```

Kemudian dilanjutkan dengan menggambarkan blue neon yang menjadi rintangan planenya. Kemudian lanjut menggambar meteor jika randommeteornya 0 maka gambar strong meteor dan jika 1 maka gambar weakmeteor. Kemudian kita end batchnya.

```
if(Gdx.input.isKeyPressed(Input.Keys.LEFT)){
    input.left();
}

if(Gdx.input.isKeyPressed(Input.Keys.RIGHT)){
    input.right();
}

if(plane.x<=35){
    plane.x=35;
    collisionWithLaserSound.play();
    score--;
    System.out.println(score);
}
if(plane.x >=510){
    plane.x=510;
    collisionWithLaserSound.play();
    score--;
    System.out.println(score);
}
```

Kemudian jika tombol left dipencet maka kita panggil method left yang ada di input itu tadi dimana input adalah objek dari Plane. Kemudian jika dipencet right maka memanggil method right yang ada di inpt. Kemudian kan nanti di gamenya ada batesan neon kiri kanan maka jika posisi x planenya itu <=35 atau mengenai batas neon kiri maka tetapkan posisi x planenya di 35 dan kita play sound collision with lasernya kemudian score nya kita -1 untuk tiap framenya dan juga health pesawarnya. Kemudian sama halnya dengan batas neon kanan maka tetapkan posisi x pada 510 dan play sound collisionwithlaser dan score nya kita -1 tiap frame dan juga health pesawatnya.

```
if(TimeUtils.nanoTime() - lastTime > 1999999999){
    randomMeteor= MathUtils.random(0,1);
    spawnBlueNeonAndMeteor();
    if(randomMeteor==0){
        obstacle= new StrongMeteor(meteor);
    }else {
        obstacle= new WeakMeteor(meteor);
    }
}

blueNeon.get(0).y-=250 * Gdx.graphics.getDeltaTime();
blueNeon.get(1).y-=250 * Gdx.graphics.getDeltaTime();
meteor.y-=250*Gdx.graphics.getDeltaTime();
```

Maksud dari if ini adalah ketika sejak meteor dan blue neon dispawn sudah lebih 1.99999 detik maka kita memulai spawn blue neon dan meteor dengan merandom meteor. Kemudian dispawn. Jika random meteornya 0 hasilnya adalah strong meteor jika 1 maka Weakmeteor. Kemudian blue neon yang sudah dibuat tadi kita buat bergerak dengan mengurangi y nya sebesar 250 dikali deltatime begitu pula dengan meteornya.

```
if(blueNeon.get(0).intersects(plane) || blueNeon.get(1).intersects(plane)){
    collisionWithLaserSound.play();
    score--;
}

if(plane.intersects(meteor)){
    planeMeteor.play();
    score--;
}

if(score<=0){
    score=0;
}
```

Kemudian jika planenya itu intersects sama blue neon maka kita play sound collisionnya dan scorenya dimin 1 perframenya dan dimin 1 juga untuk health pesawatnya. Kemudian jika planenya bertabrakan dengan meteor maka kita play sound planeMeteor dan scorenya diminus 1 per framenya dan dimin 1 juga untuk health pesawatnya. Kemudian, karena score minimumnya itu 0 maka kita buat if score <= 0 maka tetapkan nilai scorenya di 0.

```
if(input.health<=0){
    TryAgainScreen againScreen = new TryAgainScreen(game,score);
    game.setScreen(againScreen);
}
```

Kemudian, jika healthnya <=0, maka kita set screennya menjadi TryAgainScreen.

```
@Override
public void dispose() {
    leftLaserImg.dispose();
    planeImg.dispose();
    rightLaserImg.dispose();
    peluruPlaneImg.dispose();
    blueNeonImg.dispose();
    meteorImg.dispose();
    meteor2Img.dispose();
    backgrounImg.dispose();
    planeMusic.dispose();
    laserSound.dispose();
    meteorDestroyedSound.dispose();
    collisionWithLaserSound.dispose();
    planeMeteor.dispose();

}
```

Kemudian langkah akhirnya kita dispose attribute yang bisa didispose.

```
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.Screen;
import com.badlogic.gdx.graphics.OrthographicCamera;
import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.utils.ScreenUtils;


public class TryAgainScreen implements Screen {
    Texture againTexture;
    final Meteor game;
    OrthographicCamera camera;
    GameScreen game2;
    int score;
```

Kemudian, class terakhir yang kita buat adalah TryAgainScreen yang meimplementasi interface Screen. Untuk atribut yang dimiliki adalah texture yang merupakan design dari TryAgainScreen kita, kemudian objek dari meteor, Orthographic camera, Gamescreen, dan juga score yang merupakan score akhir yang diperoleh player. Untuk library yang diimport adalah:

```
com.badlogic.gdx.Gdx;
com.badlogic.gdx.Screen;
com.badlogic.gdx.graphics.OrthographicCamera;
com.badlogic.gdx.graphics.Texture;
com.badlogic.gdx.utils.ScreenUtils;
```

```
public TryAgainScreen(Meteor meteor, int Score) {
    game=meteor;
    againTexture= new Texture(Gdx.files.internal( path: "gameOver.png"));
    camera=new OrthographicCamera();
    camera.setToOrtho( yDown: false, viewportWidth: 600, viewportHeight: 450);
    this.score=Score;
}
```

Kemudian, untuk Constructornya parameter yang diminta adalah Meteor dan juga integer score. Kemudian kita set gamenya sesuai nilai parameter. Set texturenya dari files internal. Kemudian, kita new Orthographic cameranya kemudian ser ortonya dan set scorenya sesuai parameter.

```
@Override
public void render(float delta) {
    ScreenUtils.clear( r: 0, g: 0, b: 0, a: 1);
    camera.update();
    game.batch.setProjectionMatrix(camera.combined);
    game.batch.begin();

    game.batch.draw(againTexture, x: 0, y: 0, width: 600, height: 450);
    game.font.draw(game.batch,  str: "SCORE: " + score,  x: 210, y: 200);
    game.batch.end();

    if(Gdx.input.isTouched()){
        game2= new GameScreen(game);
        game.setScreen(game2);
        dispose();
    }
}
```

Kemudian, untuk method rendernya sendiri, langkah awal yang dilakukan adalah mengclear screen menjadi warna hitam. Kemudian update cameranya dan setProjectionMatrix. Kemudian kita begin batchnya. Texture yang digambar pertama adalah gambar design dari Try again Screen. Kemudian, kita draw scorenya pada x 210 dan y 200 kemudian kita end untuk 200. Kemudian kita beri IF jika layar ditap dimana saja, maka kita set screenya lagi untuk kembali ke GameScreen.

3. Petunjuk/Peraturan Permainan dan Penjelasan

MAIN MENU:

→ Yang dilakukan user untuk memulai permainan adalah mengtap layar agar game start

ON GAME:

→ **CONTROLLER:**

1. **LEFT BUTTOn=** Menggerakkan plane ke kiri
2. **RIGHT BUTTON=** Menggerakkan plane ke kanan
3. **SPACE BUTTON=** menembakkan peluru laser

→ **TUTORIAL:**

Jadi, yang dilakukan user adalah melewati lubang yang pada awalnya terdapat meteor. Cara menghancurkan meteornya adalah dengan menembakkan peluru laser ke meteor yang ada (SPACE). Perlu diingat, ada 2 jenis meteor, dimana meteor yang tidak berapi one hit akan langsung hancur. Sedangkan, meteor berapi hancur ketika ditembak 2 kali. Ketika meteor hancur, player akan mendapatkan score. Selama permainan, player harus tetap memperhatikan plane healthnya karena jika plane healthnya 0 maka akan game over.

GAME OVER:

Ketika plane healthnya 0, maka akan game over. Untuk bermain lagi / try again, player bisa mengtap layar untuk memulai gamenya kembali.