PROJECT UAS OOP

Game Pesawat Tempur Menggunakan Greenfoot

Achmad Haykal Riziq (2211102441170)

Indah Romadansyah (2211102441183)

Nazwa Zakiah (2211102441052)

Muhammad Rendi Herdasaputra (2211102441250)

Teknik InformatikaFakultasSains&Teknologi

Universitas Muhammadiyah Kalimantan Timur

Samarinda, 2023

TEMA : PESAWAT TEMPUR

Pesawat tempur adalah game aksi seru yang mengusung tema pertempuran antara pesawat luar angkasa dan aliean di galaksi yang penuh bahaya. Pemain akan mengendalikan pesawat luar angkasa canggih yang harus melawan laser laser alien jahat yang mengancam perdamaian seluruh galaksi. “pesawat tempur” menawarkan pengalaman gaming yang mendebarkan, diisi dengan aksi luar angkasa, senjata canggih, dan tantangan menarik. Siapkan diri Anda untuk melibatkan diri dalam pertempuran epic melawan alien di galaksi yang luas!

fitur utama

1. **Pertempuran**
    - Pemain akan memasuki medan perang Antariksa yang penuh dengan alien musuh yang agresif.
2. **Senjata**
    - Pesawat dilengkapi dengan senjata peluru plasma
    - Kemampuan khusus??
3. **Intelegensi Buatan Alien yang Cerdas**
    - Alien dilengkapi dengan kecerdasan buatan untuk membuat permainan lebih menantang.
    - Alien dapat menyesuaikan strategi mereka berdasarkan respon pemain, membuat setiap pertemuran menjadi unik.
4. **Sistem Poin dan Prestasi**
    - Pemain dapat mengumpulkan poin selama pertempuran.
    - Ketika peluru plasma pesawat menembaki alien maka skor bertambah sebanyak 20.

CARA BERMAIN

1. Pada laman pertama atau home screen terdapat opsi untuk memilih bermain menggunakan keyboard atau mouse.
2. Setelah memilih ingin bermain menggunakan mouse atau keyboard pemain akan mengendalikan pesawat luar angkasa dengan menggerakkan perangkat atau menekan tombol-tombol pada keyboard. Tujuannya adalah menghindari serangan alien sambil menembaki mereka.
3. Alien akan merespon dengan menembakkan peluru jika pesawat tepat berada di depan alien. Ketika peluru alien mengenai pesawat maka “game over”.
4. Pemain harus menembakkan peluru plasma lebih dulu dari alien agar dapat membunuh alien. Pemain harus menghindari peluru tersebut dan menciptakan strategi untuk mengalahkan peluru-peluru alien. Pemain akan mendapatkan skor banyak 20 jika peluru plasma pesawat berhasil mengenai bagian depan ufo alien

Berikut penjelesan Greenfoootnya.

KELAS WORLD

1. **HomeScreen**
   Kelas “HomeScreen” ini digunakan untuk membuat dunia awal permainan dengan satu tombol “StartButton” di Tengah-tengah layar.
   import greenfoot.*;

   public class HomeScreen extends World
   {
      public HomeScreen()
      {
         super(854, 480, 1);
         prepare();
      }

      private void prepare()
      {
         addObject(new StartButton(), getWidth() / 2, getHeight() / 2);
      }
   }

   Berikut adalah penjelasan singkat dari struktur kode:
   1. **Konstruktor `HomeScreen`:**
      - Konstruktor digunakan untuk membuat dunia HomeScreen.
      - Memanggil konstruktor kelas World dengan parameter lebar (854), tinggi (480), dan jumlah lapisan (1).
      - Memanggil metode `prepare()` untuk menyiapkan objek di dalam dunia.
   2. **Metode `prepare()`:**
      - Metode ini bertanggung jawab untuk menyiapkan objek-objek di dalam dunia.
      - Menambahkan objek baru ke dunia menggunakan `addObject()`.
      - Objek yang ditambahkan adalah instance dari kelas `StartButton` yang ditempatkan di tengah-tengah dunia (`getWidth() / 2, getHeight() / 2`).
   3. **Instance `StartButton`:**
      - `StartButton` adalah kelas yang mungkin telah didefinisikan di tempat lain dalam proyek. Ini mungkin adalah tombol atau objek yang digunakan untuk memulai permainan atau melakukan tindakan lain pada layar awal.

2. **bg**
   kelas “bg” mengatur logika dan elemen-elemen utama dari permainan, termasuk tampilan, suara, dan pengelolaan skor.
   import greenfoot.*;

   public class bg extends World
   {

```
    Counter counter = new Counter("Skor: ");
    private GreenfootSound mulai = new GreenfootSound("ms.mp3");
    private GreenfootSound endSound = new GreenfootSound("end.mp3");
    private int restartDelay = 50;
    private int restartTimer = 0;

    public bg()
    {
        super(854, 480, 1);
        setPaintOrder(skor.class, api.class, bintang.class, ps.class, ufo.class, Counter.class,
peluru1.class, peluru2.class);
        addObject(new bintang(), 150, 50);
        addObject(new ps(), 69, 215);
        addObject(new ufo(), 790, 320);
        addObject(counter, 120, 450);
        showHomeScreen();
    }



    public void act(){
        if(Greenfoot.getRandomNumber(1000) < 4){
            addObject(new bintang(), 853, Greenfoot.getRandomNumber(70));
            addObject(new bintang(), 953, Greenfoot.getRandomNumber(70));
        }

        if(Greenfoot.getRandomNumber(100) < 1){
            addObject(new ufo(), 853, Greenfoot.getRandomNumber(479));
        }


        if (Greenfoot.isKeyDown("R") && restartTimer <= 0) {
            stopEndSound();
            Greenfoot.setWorld(new bg());
            restartTimer = restartDelay;
        }


        if (restartTimer > 0) {
            restartTimer--;
        }
        }
    public void showObjects(boolean show){

        if(!show){
            removeObjects(getObjects(null));
        } else{
```

```
            addObject(new bintang(), 690, 20);
            addObject(new ps(), 69, 215);
            addObject(new ufo(), 790, 320);
            addObject(counter, 120, 450);
        }
    }

    private void stopEndSound() {
        if (endSound != null && endSound.isPlaying()) {
            endSound.stop();
        }
    }

    public void tambah(){
        counter.add(20);
    }

    public void selesai()
    {
        addObject(new skor(counter.getValue()), getWidth()/2, getHeight()/2);
        showReplayMessage();
        endSound.setVolume(40);
        endSound.play();
    }

    private void showReplayMessage() {
        showText("Press 'R' to replay", getWidth() / 2, getHeight() / 2 + 50);
    }

    private void showHomeScreen()
    {
        Greenfoot.setWorld(new HomeScreen());
    }
}
```

Berikut adalah penjelasan singkat dari struktur dan fungsionalitas kelas ini:

1. **Konstruktor `bg`:**
   - Konstruktor digunakan untuk membuat dunia baru (`bg`) dengan lebar 854, tinggi 480, dan satu lapisan.
   - Metode `setPaintOrder()` digunakan untuk menentukan urutan penggambaran objek-objek di dunia.
   - Menambahkan objek `bintang`, `ps` (pesawat pemain), dan `ufo` ke dunia.
   - Menambahkan objek `counter` (objek kounter skor) ke dunia pada posisi tertentu.
   - Memanggil metode `showHomeScreen()` untuk menampilkan layar awal permainan.
2. **Metode `act()`:**
   - Menambahkan objek `bintang` dan `ufo` secara acak ke dunia.
   - Memproses input pemain, seperti menekan tombol "R" untuk merestart permainan.

- Mengatur timer dan kondisi untuk memungkinkan restart setelah permainan selesai.

3. **Metode `showObjects(boolean show)`:**
   - Menampilkan atau menyembunyikan objek-objek di dunia berdasarkan parameter `show`.
   - Jika `show` adalah `false`, maka semua objek dihapus dari dunia.
   - Jika `show` adalah `true`, objek-objek tertentu ditambahkan kembali ke dunia.
4. **Metode `stopEndSound()`:**
   - Memastikan bahwa suara `endSound` dihentikan jika sedang diputar.

5. **Metode `tambah()`:**
   - Menambahkan nilai sejumlah tertentu ke objek kounter skor (`counter`).
6. **Metode `selesai()`:**
   - Menambahkan objek `skor` ke dunia dengan nilai skor saat ini.
   - Memanggil metode `showReplayMessage()` untuk menampilkan pesan replay.
   - Memutar suara `endSound` dengan pesan audio yang sesuai.

7. **Metode `showReplayMessage()`:**
   - Menampilkan pesan di tengah layar untuk memberitahu pemain bahwa mereka dapat menekan tombol "R" untuk replay.
8. **Metode `showHomeScreen()`:**
   - Memanggil metode `showHomeScreen()` untuk menampilkan layar awal permainan.

KELAS AKTOR

1. **Counter**

Kelas counter merupakan acktor untuk menangani dan menampilkan nilai skor dalam permainan. Kelas ini bertanggung jawab untuk memanajemen dan menampilkan nilai skor di layar. Ketika nilai skor berubah, actor ini secara otomatis memperbarui gambarnya. Suara “coin.mp3” akan diputar saat nilai skor bertambah, memberikan umpan balik audio kepada pemain.

```
import greenfoot.*;

public class Counter extends Actor
{
   private int value = 0;
   private int target = 0;
   private String text;
   private int stringLength;

   public Counter()
   {
      this("");
   }

   public Counter(String prefix)
   {
      text = prefix;
      stringLength = (text.length() + 2) * 16;
```

```
        setImage(new GreenfootImage(stringLength, 24));
        GreenfootImage image = getImage();
        Font font = image.getFont();
        image.setFont(font.deriveFont(24.0F));

        updateImage();
    }

    public void act() {
        if(value < target) {
            value++;
            updateImage();
        }
        else if(value > target) {
            value--;
            updateImage();
        }
    }

    public void add(int score)
    {
         Greenfoot.playSound("coin.mp3");
        target += score;
    }

    public void subtract(int score)
    {
        target -= score;
    }

    public int getValue()
    {
        return value;
    }

    /**
     * Make the image
     */
    private void updateImage()
    {
        GreenfootImage image = getImage();
        image.clear();
        image.drawString(text + value, 1, 18);
        image.setColor(Color.WHITE);
    }
}
```

Berikut adalah penjelasan singkat dari struktur dan fungsionalitas kelas ini:

1. **Variabel dan Konstruktor:**
   - `value`: Menyimpan nilai skor saat ini.
   - `target`: Menyimpan nilai skor target yang ingin dicapai.
   - `text`: Menyimpan teks atau awalan yang akan ditampilkan sebelum nilai skor.
   - `stringLength`: Menyimpan panjang teks dan menyertakan ruang tambahan.
   - Konstruktor `Counter(String prefix)`: Inisialisasi nilai-nilai dan membangun gambar untuk actor.

2. Metode `act()`:
   - Diaktifkan secara terus menerus selama permainan.
   - Jika nilai skor saat ini kurang dari target, nilai skor bertambah satu dan gambar diperbarui.
   - Jika nilai skor saat ini lebih dari target, nilai skor berkurang satu dan gambar diperbarui.

3. **Metode `add(int score)`:**
   - Menambahkan nilai skor target dengan jumlah tertentu.
   - Memutar suara "coin.mp3" menggunakan `Greenfoot.playSound()` saat nilai skor bertambah.

4**. Metode `subtract(int score)`:**
   - Mengurangkan nilai skor target dengan jumlah tertentu.

5**. Metode `getValue()`:**
   - Mengembalikan nilai skor saat ini.

6. **Metode `updateImage()`:**
   - Mengupdate gambar actor dengan teks yang mencakup nilai skor dan teks awalan.
   - Menggunakan objek `GreenfootImage` untuk memanipulasi tampilan gambar.

2. StarButton

Kelas StarButton merupakan actor yang akan memulai permainan Ketika dklik. Kelas ini bertindak sebagai pemulai permainan dan memastikan bahwa dunia permainan diganti Ketika tombol start diklik. Selain itu, Ketika permainan dimulai, music latar belakang juga dapat diputar untuk menambah atmosfer permainan.

```
import greenfoot.*;

public class StartButton extends Actor
{
   private GreenfootSound backgroundMusic = new GreenfootSound("ms.mp3");

   public StartButton()
   {

      setImage(new GreenfootImage("start_button.png"));
   }

   public void act()
   {
```

```
        if (Greenfoot.mouseClicked(this))
        {
            startGame();
        }
    }

    private void startGame()
    {
        Greenfoot.setWorld(new bg());
        if (!backgroundMusic.isPlaying()) {
            backgroundMusic.setVolume(30);
            backgroundMusic.play();
        }
    }
}
```

Berikut adalah penjelasan singkat dari struktur dan fungsionalitas kelas ini:

1. **Variabel dan Konstruktor:**
   - `backgroundMusic`: Objek `GreenfootSound` untuk memainkan musik latar belakang.
   - Konstruktor `StartButton()`: Mengatur gambar untuk actor menggunakan file gambar "start_button.png".

2. **Metode `act()`:**
   - Metode ini dipanggil secara terus menerus selama permainan.
   - Jika tombol start (`StartButton`) diklik, maka metode `startGame()` dipanggil.

3. **Metode `startGame()`:**
   - Mengganti dunia permainan (`bg`) menggunakan `Greenfoot.setWorld(new bg())`.
   - Memainkan musik latar belakang (`ms.mp3`) hanya jika musik tidak sedang diputar.

3. Api

Kelas api merupakan actor yang akan mebuat efek animasi ledakan api. Kelas api bertanggung jawab untuk membuat efek animasi ledakan api. Ketika objek ini dimunculkan dalam dunia/world, ia akan memainkan suara ledakan dan secara bertahap mengubah ukuran dan gambar untuk menciptakan efek ledakan yang visual. Setelah animasi selesai, objek ini dihapus dari dunia.

```
import greenfoot.*;

public class api extends Actor
{
    private final static int IMAGE_COUNT= 8;
    private static GreenfootImage[] images;
    private int size=0;
    private int increment=1;

    public api() {
        initialiseImages();
        setImage(images[0]);
        Greenfoot.playSound("fire.wav");
```

```
    }

    /**
     * Create the images for explosion.
     */
    public synchronized static void initialiseImages() {
        if(images == null) {
            GreenfootImage baseImage = new GreenfootImage("api.png");
            int maxSize = baseImage.getWidth()*2;
            int delta = maxSize / (IMAGE_COUNT+1);
            int size = 0;
            images = new GreenfootImage[IMAGE_COUNT];
            for(int i=0; i < IMAGE_COUNT; i++) {
                size = size + delta;
                images[i] = new GreenfootImage(baseImage);
                images[i].scale(size, size);
            }
        }
    }
    public void act()
    {

        setImage(images[size]);

        size += increment;
        if(size>=IMAGE_COUNT) {
            increment = -increment;
            size += increment;
        }

        if(size <= 0) {
            getWorld().removeObject(this);
        }
    }
}
```

Berikut adalah penjelasan singkat dari struktur dan fungsionalitas kelas ini:

1. **Variabel:**
   - `IMAGE_COUNT`: Konstanta yang menyimpan jumlah gambar animasi ledakan.
   - `images`: Array dari objek `GreenfootImage` yang menyimpan gambar-gambar animasi ledakan.
   - `size`: Variabel yang menunjukkan ukuran saat ini dari animasi ledakan.
   - `increment`: Variabel yang menunjukkan penambahan atau pengurangan ukuran animasi ledakan.

2. **Konstruktor `api()`:**
   - Menginisialisasi gambar animasi ledakan menggunakan metode `initialiseImages()`.
   - Menetapkan gambar pertama dari animasi ledakan sebagai gambar utama.

- Memainkan suara ledakan menggunakan `Greenfoot.playSound("fire.wav")`.

3. **Metode `initialiseImages()`:**
   - Metode yang digunakan untuk membuat gambar-gambar animasi ledakan.
   - Menggunakan gambar dasar (`api.png`) dan membuat gambar dengan skala yang berbeda untuk setiap frame animasi.

4. **Metode `act()`:**
   - Dipanggil secara terus menerus selama permainan.
   - Mengatur gambar actor sesuai dengan ukuran saat ini dari animasi ledakan.
   - Memodifikasi ukuran animasi ledakan dengan menambah atau mengurangkan nilai `size`.
   - Jika ukuran mencapai batas tertentu, animasi ledakan dihapus dari dunia.

4. Bintang

Kelas `bintang`, merupakan actor yang jatuh dari atas layar ke bawah. Kelas `bintang` bertanggung jawab untuk membuat efek visual bintang yang jatuh dari atas ke bawah layar. Ketika bintang mencapai atau melampaui batas bawah layar, objek ini dihapus dari dunia.

```
import greenfoot.*;

public class bintang extends Actor
{
   /**
    * Act - do whatever the bintang wants to do. This method is called whenever
    * the 'Act' or 'Run' button gets pressed in the environment.
    */
   public void act()
   {
      int x = getX();
      int y = getY();

      // Decrease the Y-coordinate to make the star fall
      setLocation(x, y + 1);

      // Remove the star when it reaches the bottom of the screen
      if (y >= getWorld().getHeight() - 1) {
         getWorld().removeObject(this);
      }
   }
}
```

Berikut adalah penjelasan singkat dari struktur dan fungsionalitas kelas ini:

1. **Metode `act()`:**
   - Dipanggil secara terus menerus selama permainan.
   - Mendapatkan posisi saat ini (`x` dan `y`) dari objek.
   - Menurunkan nilai `y` untuk membuat bintang jatuh ke bawah.
   - Mengatur posisi objek baru menggunakan `setLocation(x, y + 1)`.

2. **Kondisi `if (y >= getWorld().getHeight() - 1)`:**
   - Memeriksa apakah bintang telah mencapai atau melampaui batas bawah layar.

- Jika iya, objek bintang dihapus dari dunia menggunakan `getWorld().removeObject(this)`.

5. **peluru1**

Kelas `peluru1`, merupakan actor untuk menangani pergerakan dan interaksi peluru yang ditembakkan oleh pemain. Kelas `peluru1` bertanggung jawab untuk mengatur pergerakan peluru yang ditembakkan oleh pemain dan menangani tabrakan dengan objek UFO. Jika peluru mencapai batas kanan layar atau menabrak UFO, efek ledakan (`api`) ditampilkan, skor ditingkatkan, dan objek UFO serta peluru dihapus dari dunia.

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

public class peluru1 extends Actor
{

   public void act()
   {
      int x=getX();
      int y=getY();
      setLocation(x+29,y);
      if(x == 853){
         getWorld().removeObject(this);
         return;
      }
      Actor a = getOneObjectAtOffset(0 ,0, ufo.class);
      if(a != null){
         getWorld().addObject(new api(),getX(),getY());
         ((bg)getWorld()).tambah();
         getWorld().removeObject(a);
         getWorld().removeObject(this);
         return;
      }
   }
}
```

Berikut adalah penjelasan singkat dari struktur dan fungsionalitas kelas ini:

1. **Metode `act()`:**
   - Dipanggil secara terus menerus selama permainan.
   - Mendapatkan posisi saat ini (`x` dan `y`) dari objek.
   - Menggeser posisi peluru ke kanan (`setLocation(x + 29, y)`).
   - Memeriksa apakah peluru telah mencapai batas kanan layar (x == 853).
   - Jika ya, peluru dihapus dari dunia menggunakan `getWorld().removeObject(this)` dan keluar dari metode.
   - Memeriksa apakah peluru menabrak objek UFO (`ufo`) menggunakan `getOneObjectAtOffset`.
   - Jika ya, menambahkan objek `api` (efek ledakan) ke dunia pada posisi peluru.
   - Menambahkan skor dengan memanggil metode `tambah()` pada objek `bg`.
   - Menghapus objek UFO dan peluru dari dunia.
   - Keluar dari metode setelah menangani tabrakan.

6. **peluru2**

Kelas `peluru2`, merupakan actor untuk menangani pergerakan dan interaksi peluru yang ditembakkan oleh musuh. Kelas `peluru2` bertanggung jawab untuk mengatur pergerakan peluru yang ditembakkan oleh musuh dan menangani tabrakan dengan objek pesawat pemain. Jika peluru mencapai batas kiri layar atau menabrak pesawat pemain, efek ledakan (`api`) ditampilkan, permainan diakhiri, dan objek pesawat pemain serta peluru dihapus dari dunia.

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

public class peluru2 extends Actor
{

   public void act()
   {
      int x=getX();
      int y=getY();
      setLocation(x-17,y);
      if(x==0){
         getWorld().removeObject(this);
         return;
      }
      Actor a = getOneIntersectingObject(ps.class);
      if(a != null){
         getWorld().addObject(new api(),getX(),getY());
         ((bg)getWorld()).selesai();
         getWorld().removeObject(a);
         getWorld().removeObject(this);
         return;
      }
   }
}
```

Berikut adalah penjelasan singkat dari struktur dan fungsionalitas kelas ini:

1. **Metode `act()`:**
   - Dipanggil secara terus menerus selama permainan.
   - Mendapatkan posisi saat ini (`x` dan `y`) dari objek.
   - Menggeser posisi peluru ke kiri (`setLocation(x - 17, y)`).
   - Memeriksa apakah peluru telah mencapai batas kiri layar (x == 0).
   - Jika ya, peluru dihapus dari dunia menggunakan `getWorld().removeObject(this)` dan keluar dari metode.
   - Memeriksa apakah peluru menabrak objek pesawat pemain (`ps`) menggunakan `getOneIntersectingObject`.
   - Jika ya, menambahkan objek `api` (efek ledakan) ke dunia pada posisi peluru.
   - Memanggil metode `selesai()` pada objek `bg` untuk menyelesaikan permainan.
   - Menghapus objek pesawat pemain dan peluru dari dunia.
   - Keluar dari metode setelah menangani tabrakan.

7. **ps**

Kelas `ps` (pesawat pemain), mengatur perilaku pesawat pemain dalam permainan. Kelas `ps` bertanggung jawab untuk mengatur perilaku pesawat pemain, termasuk pergerakan dan penembakan. Selain itu, kelas ini juga menangani interaksi dengan objek UFO dan mengakhiri permainan jika terjadi tabrakan.

```
import greenfoot.*;

public class ps extends Actor
{
   private bg pl;
   private int shootDelay = 30;
   private int shootTimer = 0;
   private int movementSpeed = 5;

   public void addedToWorld(World bg) {
      pl = (bg) bg;
   }

   public void act()
   {
      if (Greenfoot.mouseMoved(null)) {
         MouseInfo mouse = Greenfoot.getMouseInfo();
         setLocation(mouse.getX(), mouse.getY());
      }


      if (Greenfoot.mouseClicked(null) && shootTimer <= 0) {
         pl.addObject(new peluru1(), getX() + getImage().getHeight(), getY());
         Greenfoot.playSound("p1.wav");
         shootTimer = shootDelay;
      }


      if (shootTimer > 0) {
         shootTimer--;
      }


      if (Greenfoot.isKeyDown("W")) {
         setLocation(getX(), getY() - movementSpeed);
      }
      if (Greenfoot.isKeyDown("S")) {
         setLocation(getX(), getY() + movementSpeed);
      }
      if (Greenfoot.isKeyDown("A")) {
         setLocation(getX() - movementSpeed, getY());
      }
```

```
        if (Greenfoot.isKeyDown("D")) {
            setLocation(getX() + movementSpeed, getY());
        }
        if (Greenfoot.isKeyDown("Space")) {
            pl.addObject(new peluru1(), getX() + getImage().getHeight(), getY());
            shootTimer = shootDelay;
        }



        Actor a = getOneObjectAtOffset(0, 0, ufo.class);
        if (a != null) {
            getWorld().addObject(new api(), getX(), getY());
            ((bg) getWorld()).selesai();
            getWorld().removeObject(a);
            getWorld().removeObject(this);
        }
    }
}
```

Berikut adalah penjelasan singkat dari struktur dan fungsionalitas kelas ini:

1. **Variabel:**
   - `pl`: Objek dari kelas `bg` (dunia permainan).
   - `shootDelay`: Waktu jeda antara penembakan.
   - `shootTimer`: Waktu tersisa sebelum dapat melakukan penembakan lagi.
   - `movementSpeed`: Kecepatan pergerakan pesawat pemain.

2. **Metode `addedToWorld(World bg)`:**
   - Dipanggil ketika objek `ps` ditambahkan ke dunia.
   - Menginisialisasi variabel `pl` dengan objek dunia saat ini.

3. **Metode `act()`:**
   - Dipanggil secara terus menerus selama permainan.
   - Menggerakkan pesawat pemain sesuai dengan input mouse dan keyboard.
   - Mengecek apakah terjadi klik mouse dan waktu penembakan (`shootTimer`) sudah habis, jika ya, menembakkan peluru (`peluru1`).
   - Memainkan suara penembakan.
   - Mengurangi waktu jeda penembakan (`shootTimer`).
   - Mengecek input keyboard untuk pergerakan atas, bawah, kiri, dan kanan serta untuk melakukan penembakan (`Space`).
   - Mengecek apakah pesawat pemain bertabrakan dengan objek UFO (`ufo`). Jika ya, menampilkan efek ledakan (`api`), mengakhiri permainan, dan menghapus objek UFO dan pesawat pemain dari dunia.

8. Ufo

Kelas `ufo`, yang bertanggung jawab untuk mengatur perilaku objek UFO dalam permainan. Kelas `ufo` bertanggung jawab untuk mengatur pergerakan UFO ke kiri dan menanggapi pergerakan mouse dengan menembakkan peluru (`peluru2`) jika posisi mouse berada dalam kisaran vertikal UFO.

```
import greenfoot.*;

public class ufo extends Actor
{
   /**
    * Act - do whatever the ufo wants to do. This method is called whenever
    * the 'Act' or 'Run' button gets pressed in the environment.
    */

   public void act()
   {

      int x=getX();
      int y=getY();
      setLocation(x-2,y);
      if(x==0){
         getWorld().removeObject(this);
      }
      if(Greenfoot.mouseMoved(null)){
         MouseInfo mouse=Greenfoot.getMouseInfo();
         if(mouse.getY()>(y-7) && mouse.getY()<(y+7)){
            getWorld().addObject(new peluru2(),getX(),getY());
            Greenfoot.playSound("p2.wav");
         }
      }
   }
}
```

Berikut adalah penjelasan singkat dari struktur dan fungsionalitas kelas ini:

1. **Metode `act()`:**
   - Dipanggil secara terus menerus selama permainan.
   - Mendapatkan posisi saat ini (`x` dan `y`) dari objek UFO.
   - Menggeser posisi UFO ke kiri (`setLocation(x - 2, y)`).
   - Memeriksa apakah UFO telah mencapai batas kiri layar (x == 0).
   - Jika ya, UFO dihapus dari dunia menggunakan `getWorld().removeObject(this)`.
   - Mengecek apakah terjadi pergerakan mouse.
   - Jika ya, mendapatkan informasi mouse.
   - Jika posisi mouse berada dalam kisaran vertikal UFO (±7 dari posisi `y` UFO), menembakkan peluru (`peluru2`).
   - Memainkan suara penembakan (`p2.wav`).

9. skor

Kelas `ufo`, bertanggung jawab untuk mengatur perilaku objek UFO dalam permainan. Kelas `ufo` bertanggung jawab untuk mengatur pergerakan UFO ke kiri dan menanggapi pergerakan mouse dengan menembakkan peluru (`peluru2`) jika posisi mouse berada dalam kisaran vertikal UFO.

```
import greenfoot.*;

import java.util.Calendar;

public class skor extends Actor
{

   public static final float FONT_SIZE = 48.0f;
   public static final int WIDTH = 400;
   public static final int HEIGHT = 300;

   /**
    * Create a score board with dummy result for testing.
    */
   public skor(){
      this(100);
   }

   /**
    * Create a score board for the final result.
    */
   public skor(int score){
      makeImage("Game Over", "Skor: ", score);
   }

   /**
    * Make the score board image.
    */
   private void makeImage(String title, String prefix, int score){
      GreenfootImage image = new GreenfootImage(WIDTH, HEIGHT);

      image.setColor(new Color(0, 0, 0, 160));
      image.fillRect(0, 0, WIDTH, HEIGHT);
      image.setColor(new Color(255, 255, 255, 100));
      image.fillRect(5, 5, WIDTH-10, HEIGHT-10);
      Font font = image.getFont();
      font = font.deriveFont(FONT_SIZE);
      image.setFont(font);
      image.setColor(Color.WHITE);
      image.drawString(title, 60, 100);
      image.drawString(prefix + score, 60, 200);
      setImage(image);
   }
   public void act()
   {

   }
```

}
Berikut adalah penjelasan singkat dari struktur dan fungsionalitas kelas ini:
1. **Metode `act()`:**
   - Dipanggil secara terus menerus selama permainan.
   - Mendapatkan posisi saat ini (`x` dan `y`) dari objek UFO.
   - Menggeser posisi UFO ke kiri (`setLocation(x - 2, y)`).
   - Memeriksa apakah UFO telah mencapai batas kiri layar (x == 0).
   - Jika ya, UFO dihapus dari dunia menggunakan `getWorld().removeObject(this)`.
   - Mengecek apakah terjadi pergerakan mouse.
   - Jika ya, mendapatkan informasi mouse.
   - Jika posisi mouse berada dalam kisaran vertikal UFO (±7 dari posisi `y` UFO), menembakkan peluru (`peluru2`).
   - Memainkan suara penembakan (`p2.wav`).