Alkuperäiset tekijät (2015): Matti Tahvanainen ja Tero Keinänen

---

# Rämpytysralli

**Mitä projektipohja sisältää ja miten avaat sen**
Alfaversiossa oleva Masan ja Kenkun tekemä Libgdx-pelikehys (nimi-ideoita otetaan vastaan), minkä avulla on tarkoitus pystyä aloittamaan kätevästi 2D-projektit.

Pohja löytyy osoitteesta:
http://www.cs.helsinki.fi/group/linkki/materiaali/kerho/ohjkevat_2015/rampytysralli.zip Pura zip latauksen jälkeen (oikea-korva + “Extract here”).

Avaa NetBeans ja File-valikosta “Open Project…”. Valitse projektiksi purkamastasi zip:stä löytyvä rampytysralli -tyhj-/Rampytysralli ja klikkaa “Open Project”. Näkymän Projects-välilehdelle ilmestyy “Ralli Parent”. Saat projektin käyttövalmiiksi kaksoisklikkaamalla “Modules”-kohdasta löytyvää “Rampytysralli”:a.

Edellisen kohdan ohjeiden avulla sinulle pitäisi rakentua “Projects”-näkymään uusi, todellinen “Rampytysralli”-projekti. Klikkaa sitä vielä hiiren oikealla korvalla ja etsi “Clean and Build”, jotta projekti kasaa vielä kokoon sen tarvitsemat muualta löytyvät kirjastot. Tämä voi kestää hetken.

**Pelikehys sisältää**
GameObject-luokan, Scene-luokan, SceneManager-luokan, InputListener-luokan, StaticImage-luokan

**Pelikehyksen idea**
Kaikki pelin hahmot ja kuvat ovat GameObjecteja. Kaikki pelin kentät/tasot/valikot yms. ovat Scenejä (suom. kohtaus/tapahtumapaikka). SceneManager tallentaa kaikki Scenet listaan, jolloin Scenestä voi mennä toiseen vaivattomasti.

## Valikon teko

Aloitetaan pelin teko luomalla siihen alkuvalikko. Pelin kaikki kentät, tasot tai valikot laajentavat luokan Scene toiminnallisuutta.

1. Avaa luokka MenuScene. (Löytyy kohdasta Rampytysralli > Source Packages > rampytysralli > ) Luokka laajentaa (extends) luokkaa Scene.

2. Lisätään MenuScenelle taustakuva initialize()-metodissa eli alustettaessa. Kaikki paikallaan jököttävät muuttumattomat kuvat ovat tyypiltään StaticImage, joka löytyy valmiina pelikehyksestä. Uuden kuvan tekeminen onnistuu seuraavalla tavalla:

```
// Tallennetaan taustakuva muuttujaan
Sprite backgroundImg = new Sprite(new Texture("assets/bg.png"));
// Tehdään StaticImage, joka saa koordinaatiston pisteen
// parametrina, sekä yllä luodun kuvan
StaticImage background = new StaticImage(kuvan_x-koordinaatti,
kuvan_y-koordinaatti, backgroundImg);
```

   Libgdx:ssä koordinaatiston origo (kohta (0,0)) sijaitsee ikkunan vasemmassa alakulmassa. Kaikki hahmot myös piirretään alkaen vasemmasta alakulmasta. Täytä kohdat kuvan_x-koordinaatti, sekä kuvan_y-koordinaatti oikeilla arvoilla.

3. Jos Sprite ja/tai StaticImage alleviivautuu punaisella niin klikkaa rivin alussa näkyvää hehkulamppua ja lisää ehdotettu kirjasto: "Add import for …"

4. Seuraavaksi laitamme kuvan näkyviin peli-ikkunaan. Scene-luokasta löytyy valmiina lista kaikille yhden Scenen sisältämille hahmoille. Listaan voi lisätä hahmon kutsumalla MenuScenessä initialize()-metodissa

```
addDrawable(background);
```

   Koska Scene-luokka huolehtii hahmojen piirtämisestä, taustakuvan pitäisi ilmestyä peli-ikkunaan kuin taikaiskusta kun klikkaat vihreästä play-napista ohjelman käyntiin! Jos NetBeans kysyy minkä pääluokan haluat ajaa niin valitse rampytysralli.GameInitializer .

5. Lisää ylläolevan esimerkin mukaisesti MenuSceneen otsikko kuvana. Huomaa, että sinun täytyy nyt muuttaa luotavan Sprite-muuttujan ja StaticImage-muuttujan nimi, ettei otsikon kuva tallennu taustakuvan tilalle. Päätä myös mihin kohtaan ikkunaa haluat otsikon ilmestyvän. Otsikko löytyy tiedostosta *assets/topic.png*

## Valikon napit

Valikko tarvitsee nappeja, koska ilman nappeja se on vain kuva. MenuScenestä löytyy valmiina lista napeille:

```
private ArrayList<MenuButton> buttonList;
```

Tähän listaan haluamme lisätä valikkomme napit, eli Start-napin ja Quit-napin. Nappeja voi luoda seuraavalla tavalla:

```
MenuButton exampleButton = new MenuButton(x-koordinaatti,
y-koordinaatti, napin_kuva_vakio, napin_kuva_aktiivinen, "napin_teksti");
```

Napin vakio kuva piirretään silloin, kun nappia ei osoiteta hiirellä. Vastaavasti napin aktiivinen kuva piirretään silloin, kun nappia osoitetaan. MenuButtonille annettavien kuvien tulee olla Spritejä. **Huom! Pelipohjassa tulee valmiina MenuButton-luokka. Jos haluat harjoituksen vuoksi toteuttaa osan luokasta itse, kysy ohjaajalta apua!**

Nappeja voi lisätä buttonListiin seuraavalla tavalla:

```
buttonList.add(exampleButton);
```

1. Luo MenuScenen initialize()-metodiin Start-nappi ja Quit-nappi ylläolevan esimerkin mukaan. Jos et halua piirtää omia kuvia napeille, voit käyttää projektin mukana tulevia valmiita kuvia. Valmiit kuvat ovat:
   - assets/buttonStart.png (start-napin vakiokuva)
   - assets/buttonStartHover.png (start-napin aktiivikuva)
   - assets/buttonQuit.png (quit-napin vakiokuva) sekä
   - assets/buttonQuitHover.png (quit-napin vakiokuva).

   a. Luo napin kuvista Sprite-hahmot
   b. Luo MenuButton, huom! aseta napin_teksti:ksi “START” tai “QUIT”, jotta napit myöhemmin toimivat

2. Lisää luomasi napit buttonListiin ylläolevan esimerkin mukaisesti.

3. Anna lopuksi Scene-luokalle tieto piirrettävistä napeista kutsumalla initialize()-metodin lopussa
   `addButtons(buttonList);`

4. Napit eivät vielä tunnista hiiren kosketusta tai klikkausta. Poista MenuScenen checkInputs()-metodista rivien alusta kommentit. Kommentit näyttävät tältä: `//`
   Emme tässä pelissä toteuta itse nappien klikkaamista, koska toteutus on pelikehyksessä vielä keskeneräinen. Tutki ja koita ymmärtää kuitenkin samalla, kun poistat kommentteja, miten nappien painaminen on toteutettu.

## Ralli alkaa

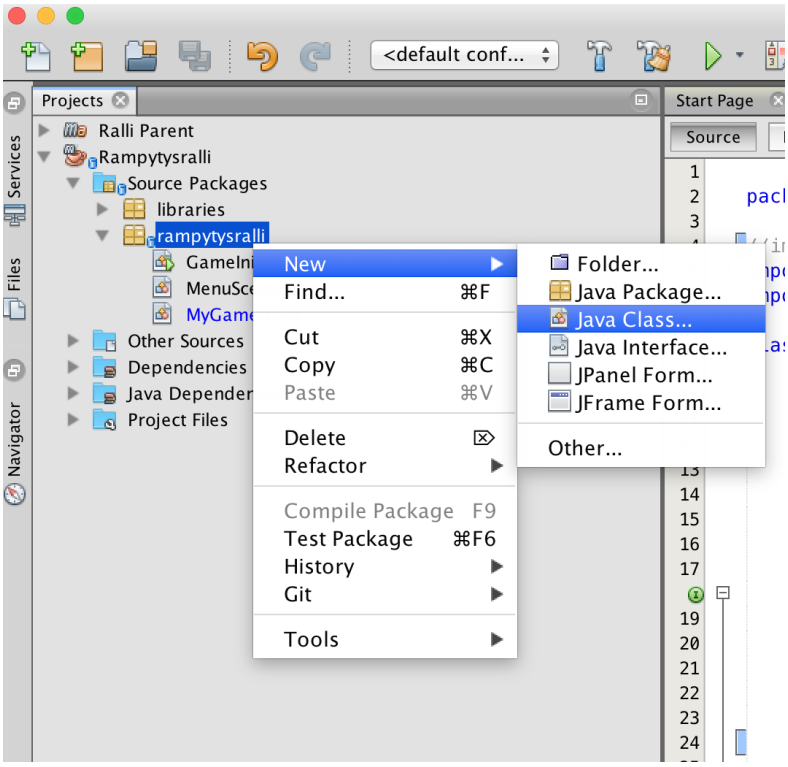
Luodaan seuraavaksi itse pelikenttä ja laitetaan autot liikkumaan käskystä!

1. Ensin tarvitaan uusi Scene, johon siirrytään kun päävalikosta on napsautettu aloitusnappia. MenuScene löytyi valmiina, mutta tehdään tällä kertaa luokka itse!

   Jos osaat jo, tee näin: luo rampytysralli-pakettiin uusi luokka nimeltä RallyScene ja laita se perimään/laajentamaan (extends) luokkaa Scene

   Jos et niin katso eteenpäin ohjeita:

a. Uusi luokka luodaan klikkaamalla rampytysralli-pakettia oikealla napilla ja valitsemalla valikosta New -> Java Class




b. Vaihda luokan nimeksi (*Class Name*) RallyScene ja paina *Finish*.

c. Laitetaan RallyScene perimään luokka Scene. Kun luokka perii jonkun toisen luokan, saa se perimänsä luokan ominaisuudet. Scene osaa tehdä paljon valmiiksi, esimerkiksi piirtää sille annetut GameObjectit. Lisää siis RallyScene-luokan määrittelyyn `extends Scene`. Mallia voit katsoa esimerkiksi MenuScene-luokan määrittelystä.

d. Nyt Scene-sana on alleviivattu punaisella virheen merkiksi. Koodi ei käänny, koska Java ei tiedä mistä etsiä Scene-luokkaa! Kerrotaan Javalle mistä Scene löytyy: lisää tiedoston alkuun (ennen `public class RallyScene` -alkuista riviä) rivi `import libraries.Scene;` Toinen tapa lisätä puuttuva import on klikata virheellisen rivin vasemmalla puolella olevaa hehkulampun kuvaa ja valita “Add import for libraries.Scene”

e. Nyt Scene löytyy, mutta ei tämä vieläkään kelpaa Javalle. Scene vaatii, että siitä perivä luokka toteuttaa muutaman metodin. Klikkaa rivin vasemmalla puolella olevaa hehkulamppua ja valitse “Implement all abstract methods”.

f. Poista vielä metodien sisältä, eli aaltosulkeiden { ja } välistä automaattisesti luotu sisältö (rivit alkavat `throw new …`)

2. SceneManager tarvitsee tiedon uudesta RallyScenestämme, koska muuten ohjelma ei ymmärrä, että siihen lisättyä uutta luokkaa pitäisi myös käyttää jossain. Etsi MyGame-luokasta metodi create(). Siellä pitäisi olla seuraavanlaiset rivit:

```
sceneManager = new SceneManager();
MenuScene menu = new MenuScene();
menu.initialize();
sceneManager.addScene(menu);
```

   Ensimmäisellä rivillä alustamme sceneManagerin. Toisella rivillä käskemme luomaan MenuScene-olion (MenuSceneen kirjoitettujen määritelmien mukaisesti). Kolmannella rivillä kutsumme menun initialize()-metodia, jonka toteutimme aiemmin. Lopuksi lisäämme sceneManagerille tiedon menusta. Toteuta näiden rivien alle vastaavalla tavalla RallyScenen alustus, kutsu luodun olion initialize()-metodia, sekä lisää sceneManagerille tieto uudesta Scenestä.

3. Lisätään taustakuva myös peliruutuun
   a. Tämä tapahtuu täysin samalla tavalla kuin taustan lisääminen valikkoon. Vilkaise Valikon teko -osion kohtia 2 - 4!

4. Luodaan Player -luokka
   a. Tämäkin tapahtuu kuten aikaisemmin RallyScene-luokan luominen. Katso siis ylempää vinkkiä kohdista 1 a ja 1 b! Luokan nimeksi tulee siis Player
   b. Player ei kuitenkaan peri (extends) Scene-luokkaa, vaan GameObject -luokan. GameObject osaa mm. piirtää sille annetun kuvan automaattisesti oikeaan paikkaan. Voit kerrata ylempää miten periminen tapahtuu (1c).
   c. Playerin tapauksessa perimisestä aiheutuu vähän erilaisia seurauksia, sillä Player-luokka tarvitseekin *konstruktorin*, joka suoritetaan aina kun uusi pelaaja luodaan. Playerille riittää automaattisesti luotu konstruktori, jonka saa tehtyä klikkaamalla hehkulamppua ja valitsemalla “Add Constructor”

5. Olemme luoneet reseptin miten tehdä yksi pelaaja. Yhden reseptin avulla voi toki luoda useita kopioita ja luodaankin nyt kaksi pelaajaa RallyScene-luokkaan

   a. Ensin esitellään muuttujat, joihin Player-luokasta tehdyt oliot laitetaan muistiin. Haluamme että muuttujiin pääsee käsiksi koko RallyScene-luokassa, joten muuttujat esitellään luokan määrittelyn jälkeen. Muuttujien esittelyn pitäisi

näyttää suunnilleen tältä:

```
import com.badlogic.gdx.graphics.g2d.Sprite;
import libraries.Scene;
import libraries.StaticImage;

class RallyScene extends Scene {

    private Player player1;
    private Player player2;

    @Override
    public void initialize() {
        StaticImage background = new StaticImage(0, 0, new Spri
        addDrawable(background);
```

b. Seuraavaksi luodaan oliot, jotka sijoitetaan player1- ja player2-muuttujiin. Lisää initialize-metodin loppuun koodi, jolla luodaan player1:
`player1 = new Player(20, 520, car1Image);`

c. car1Image antaa nyt virheen. Korvataan se kuvanlatauskoodilla. Kuva ladataan samalla tavalla kuin aiemminkin tehtiin Spritejä. Maaginen koodi on muotoa `new Sprite(new Texture("assets/ferrari.png"))`. Laita se car1Imagen paikalle ja virheen pitäisi hävitä. Näyttää aika samanlaiselta kuin `new Sprite(new Texture("assets/bg.png"))`!

d. Seuraavaksi vielä kerrotaan Scenelle, että ensimmäinen pelaaja pitäisi piirtää. Tämäkin tapahtuu samalla tavalla kuin taustan kanssa, eli addDrawable-metodia kutsuen! Voit kerrata tämän Valikon teko -otsikon alta kohdasta 3.

e. Toisen pelaajan luominen tapahtuu lähes samalla tavalla. Toista äsken tehty koodi, mutta vaihda pelaaja1 muotoon pelaaja2. Kannattaa myös vaihtaa toisen pelaajan y-koordinaattia, muuten pelaajat piirretään päällekkäin. Y-koordinaatti on esimerkissä 520, kokeile itse millä saat toisen pelaajan oikeaan kohtaan. Muista myös lisätä player2 piirrettävien hahmojen joukkoon!

## Pelaajat liikkumaan

Haluamme muokata hieman aiemmin tekemäämme Player-luokkaa. Lisätään Playerille näppäimet: (lisättävä teksti on mustalla)

```
package rampytysralli;

import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.graphics.g2d.Sprite;
import libraries.GameObject;
```

```
public class Player extends GameObject{
    private int leftButton;
    private int rightButton;
    private boolean leftPressed = false;

    public Player(float x, float y, Sprite img, int leftButton, int
rightButton) {
        super(x, y, img);
        this.leftButton = leftButton;
        this.rightButton = rightButton;
    }
    @Override
    public void move() {
    }
}
```

Näppäimen painallusta voi tarkistaa käyttämällä Libgdx:n tarjoamaa painalluksen tarkistusta:

```
if(Gdx.input.isKeyPressed(leftButton)){
            //tänne kooodia
}
```

1. Muuta luotavaa Player-hahmoa siten, että se tietää oman vasemman ja oikean näppäimensä. Lisää RallyScene-luokassa uuden pelaajan luonnin yhteyteen (initialize()-metodi) pelaajalle näppäimet esimerkiksi näin:

   ```
   Player first = new Player(x-koordinaatti, y-koordinaatti,
   pelaajanKuva, Keys.A, Keys.D);
   ```

   NetBeans alleviivaa Keys-sanat, klikkaa hehkulamppua rivin alussa ja “Add import for com.badlogic.gdx.Input.Keys”

2. Lisää myös toiselle Player-hahmolle näppäimet.

3. Lisää Player-luokan move()-metodiin tarkistus, onko Playerin vasenta näppäintä painettu (tähän oli esimerkki äsken). Jos näin on, aseta leftPressed arvoon *true*.

4. Lisää Player-luokan move()-metodiin tarkistus edellisen tarkistuksen jälkeen (ei sisäkkäin), onko Playerin oikeaa näppäintä painettu JA (*ehto* && *ehto*) onko leftPressed arvossa true (leftPressed==true) JA (&&) vasenta näppäintä **ei** (!*ehto*) ole painettu. Jos näin on, muuta pelaajan x-sijaintia seuraavalla tavalla:

   ```
   this.setX(this.getX() + 10);
   ```

   Aseta tämän jälkeen leftPressed arvoon false. Kysy ohjaajalta apua, jos tämä kohta tuntuu vaikealta.

5. Nyt Player osaa liikkua jos sitä siltä pyydetään. Mene vielä RallySceneen ja lisää updateScene()-metodiin kummallekin autolle komennot testata pitäisikö niiden liikkua:

```
player.move();
```

## Voittoruutu

Lisätään peliin vielä uusi Scene, johon siirrytään kun peli loppuu eli toinen pelaajista pääsee oikeaan reunaan.

1. Luodaan taas uusi Scene. Tämä tapahtuu samalla tavalla kuin RallyScenen luonti, mutta valitaan nimeksi VictoryScene. Ohjeet tähän löytyvät Ralli alkaa -otsikon alta. Kuvana voit käyttää tätä: “assets/victory.png”