**Praktiskais darbs priekšmetā**

**Programmatūras Inženierija**

**„Spēles *Balloon popping*”**

**projektējuma apraksts**

Izstrādāja:

ak-it-p-ns Kārkliņš Ēriks

Rīga

**Saturs**

[Ievads 3](#_Toc151291838)

[1.1. Nolūks 3](#_Toc151291839)

[1.2. Darbības sfēra 3](#_Toc151291840)

[2. Saistība ar citiem dokumentiem 4](#_Toc151291841)

[3. Dekompozīcijas apraksts 5](#_Toc151291842)

[3.1. Moduļu dekompozīcija 5](#_Toc151291843)

[3.2. Vienlaicīgo procesu dekompozīcija 7](#_Toc151291844)

[3.3. Datu dekompozīcija 8](#_Toc151291845)

[4. Atkarību apraksts 10](#_Toc151291846)

[4.1. Moduļu struktūra un atkarības 10](#_Toc151291847)

[4.2. Starpprocesu atkarības 11](#_Toc151291848)

[4.3. Datu atkarības 11](#_Toc151291849)

[5. Moduļu saskarnes apraksts 13](#_Toc151291850)

[5.1. Moduļu saskarne 13](#_Toc151291851)

[6. Detalizētais projektējums 15](#_Toc151291852)

[6.1. Moduļu detalizētais projektējums 15](#_Toc151291853)

[6.1.1. Lēmumu tabula 15](#_Toc151291854)

[6.1.2. Pseidokods 15](#_Toc151291855)

[6.2. Datu detalizētais projektējums 16](#_Toc151291856)

[7. Atsauces 18](#_Toc151291857)

# Ievads

## Nolūks

Spēle *"Balloon Popping"* (turpmāk - Spēle )ir aizraujoša un vienkārša interaktīva izklaide, kurā spēlētājiem būs iespēja izbaudīt balonu spridzināšanu gan kā izklaides veidu, gan kā izaicinošs uzdevums spēlētājiem, kas vēlas pārbaudīt savas reakcijas spējas un sasniegt maksimālo rezultātu.

## Darbības sfēra

Spēle iederas izklaides un izglītības programmatūras jomā

# Saistība ar citiem dokumentiem

Spēles *Balloon popping* projektējuma apraksta pamatā ir Spēles *Balloon popping* prasību dokuments, uz kuru balstoties tika sastādītas visas struktūras, diagrammas, apraksi un detalizēts projektējums.

# Dekompozīcijas apraksts

## Moduļu dekompozīcija

Tālāk dots apraksts katram spēles modulim – tā nosaukums, nolūks un funkcijas un uzdevumi.

1. Spēles sākums un inicializācija:

* Apraksts: Šī daļa ietver spēles sākuma un inicializācijas procesus, tostarp spēles loga un laika (*timer*) inicializāciju, fona uzstādīšanu un spēles sākuma stāvokļa iestatīšanu.
* Funkcijas vai uzdevumi:
  + - *MainWindow* konstruktors
    - *Timer* inicializācija
    - Fona uzstādīšana
    - Spēles sākuma stāvokļa iestatīšana

1. Galveno spēles funkciju vadība:

* Apraksts: Šī daļa ietver galveno spēles loģiku un funkcionalitāti, piemēram, balonu parādīšanos, to kustību un spēlētāja darbību apstrādi.
* Funkcijas vai uzdevumi:
  + - *startGame funkcija*
    - *ResetGame funkcija*
    - *gameEngine funkcija*
    - *PoPBalloons funkcija*

1. Balonu ģenerēšana un kustība:

* Apraksts: Šī daļa apakšvadībā atbild par balonu dinamisko parādīšanos un to kustību pa ekrānu.
* Funkcijas vai uzdevumi:
  + - *generateBalloon* funkcija
    - Balonu izvēles un attēlošanas loģika

1. Skaņas efekti:

* Apraksts: Šī daļa nodrošina skaņas efektu atskaņošanu, piemēram, skaņu, kad lietotājs noklikšķina uz balona.
* Funkcijas vai uzdevumi:
  + - *playSound* funkcija

1. Rezultātu un stāvokļa atjaunināšana:

* Apraksts: Šī daļa ir atbildīga par spēles rezultātu un stāvokļa atjaunināšanu, kā arī par ātruma palielināšanos sasniegtā rezultāta gadījumā.
* Funkcijas vai uzdevumi:
  + - *Rezultāta un stāvokļa atjaunināšanas loģika*

Spēles sistēmas projektēšanā tika izmantota metode „Objektu sadalīšanas metode” uz komponentu orientētas programmēšanas bāzes, 3.1.attēlā ir redzama sistēmas hierarhija.

Lietotāja saskarne (UI)

*gameTimer*

*MyCanvas*

*Balloon Logic*

*Sound Logic*

*Sound & stats*

3.1. att. Sistēmas struktūras diagramma

## Vienlaicīgo procesu dekompozīcija

Vienlaicīgo procesu dekompozīcija ir process, kurā tiek sadalīts izpildes process vairākos vienlaicīgos procesos jeb pavedienos (thread). Šāda pieeja ļauj dažādiem uzdevumiem vai darbībām izpildīties vienlaicīgi, tādējādi uzlabojot spēles veiktspēju un reaktivitāti.

Spēles vienlaicīgo procesu dekompozīciju definē sekojoši:

1. Laiks:

* Apraksts: *DispatcherTimer* ir objekts, kas palīdz veikt noteiktas darbības regulāri un vienlaicīgi. Tas tiek izmantots kā taimeris, kas regulāri izsauc spēles dzinēja *(gameEngine)* metodi ar noteiktu intervālu.
* Vienlaicīgais process: *DispatcherTimer* nodrošina veidu, kā regulāri izpildīt kodu noteiktā taimera ritmā, tādējādi radot vienlaicīgu darbību ar spēles *gameEngine*.

1. Balonu kustība:

* Apraksts: Baloni tiek pārvietoti pa ekrānu, un šī darbība notiek vienlaicīgi. Katrs balons darbojas neatkarīgi, kustoties uz augšu un pa labi vai pa kreisi.
* Vienlaicīgais process: Katrs balons darbojas kā atsevišķs pavediens, kustoties vienlaicīgi ar citiem baloniem, kurus varētu būt ekrānā vienlaicīgi.

Vienlaicīgais procesu modelis šajā spēlē ļaus saskaņot dažādas spēles darbības, piemēram, laika atjaunināšanu, balonu kustību un lietotāja ievadi, tādējādi nodrošinot dinamisku un reaktīvu spēles pieredzi.

## Datu dekompozīcija

Datu dekompozīcija ir process, kurā datus sadala un organizē, lai tos būtu viegli pārvaldīt un izmantot programmā. Dotajā spēlē dati jāiestrādā vairākos mainīgajos, kas atbildēs par dažādām spēles aspektiem. Paredzamie datu dekompozīcijas elementi šajā kodā:

1. Galvenie dati:

* *speed, intervals, rand:* Mainīgie, kas nosaka balonu kustības ātrumu, parādīšanās intervālu un gadījuma skaitļu ģeneratoru.
* *balloonSkins, i, missedBalloons, score:* Mainīgie, kas uztur informāciju par balonu izskatu, kustību, nokavētajiem baloniem un rezultātu.

1. Grafiskie dati:

* *background: ImageBrush* objekts, kas nosaka fona attēlu.

1. Grafikas elementu dzēšana:

* *itemRemover: List<Rectangle>* objekts, kas tiek izmantots, lai uzglabātu *Rectangle (balona)* objektus, kuri tiek dzēsti no ekrāna.

1. Skaņas dati:

* *player: MediaPlayer* objekts, kas tiek izmantots skaņas efekta atskaņošanai.

1. Spēles laika dati:

* *gameTimer: DispatcherTimer* objekts, kas nosaka spēles laika periodu.

1. Grafiskās sastāvdaļas:

* *MyCanvas: Canvas* objekts, uz kura attēlojas spēles komponentes.

Datu dekompozīcijai spēlē jānodrošina skaidru un organizētu veidu, kā pārvaldīt dažādus aspektus un funkcijas. Katrs datiem paredzētais mainīgais vai objekts ir atbildīgs par noteiktu spēles aspektu, kas atvieglo izpratni un uzturēšanu. Tas arī palīdz pievienot vai modificēt spēles funkcionalitāti, nemainot citus datu aspektus, kas ir svarīgi modulārai un viegli uztveramai programmai.

# Atkarību apraksts

## Moduļu struktūra un atkarības

Lai labāk izprastu sadarbības stratēģiju starp projektējuma entītijām, apskatīsim moduļu saistības un moduļu saķēdēšanas tipus.

Preču katalogam ir komunikatīvā saistība, jo tas sevī ietver vairākus procesus ar kopēju datu struktūru (preces dati), moduļa strukturālā diagramma redzama 4.1.1 attēlā. Un 4.1.2 attēlā.

*gameEngine* modulis

(Spēles loģika)

Lietotāja saskarne (UI) modulis

Skaņas modulis

4.1.1. att. Spēles moduļa vienkāršotā strukturētā diagramma

Lietotāja saskarne (UI)

*UI Handling*

*Visualisation*

*gameEngine* modulis

(Spēles loģika)

Skaņas modulis

Spēles *state* un loģika

*GameTimer*

*Balloon* loģika

Skaņas loģika

Skaņas atskaņošana

4.1.2. att. Spēles moduļa paplašinātā strukturētā diagramma

## Starpprocesu atkarības

Spēles sistēma sastāv no vairākiem procesiem, to atkarības un atbildības aprakstītas zemāk:

1. Lietotāja saskarnes modulis (UI):

* Atbildība: Nodrošina lietotāja saskarni un vizuālo reprezentāciju.
* Atkarības: Sākas un pārvalda galveno logrīku, kas saistīts ar lietotāja interakciju un vizuālo attēlojumu.

1. Spēles loģikas modulis (*gameEngine*):

* Atbildība: Nodrošina galveno spēles loģiku, tai skaitā balonu parādīšanu, kustību un rezultātu apstrādi.
* Atkarības: Sazinās ar lietotāja saskarnes moduli, lai attēlotu rezultātus un vadītu spēles laiku.

1. Skaņas modulis (*MediaPlayer*):

* Atbildība: Pārvalda skaņas efektu atskaņošanu, īpaši, kad balons tiek noklikšķināts.
* Atkarības: Sazinās ar spēles loģikas moduli, lai reaģētu uz notikumiem un atskaņotu skaņas efektu.

## Datu atkarības

1. Lietotāja saskarnes un spēles loģikas datu atkarība:

* Lietotāja saskarnes modulis UI (MainWindow) ir saistīts ar spēles loģikas moduli (gameEngine). Lietotāja darbības, piemēram, pogas noklikšķināšana, ietekmē spēles loģiku un var izraisīt spēles stāvokļa izmaiņas.

1. Skaņas un spēles loģikas datu atkarība:

* Skaņas modulis (*MediaPlayer*) ir saistīts ar spēles loģikas moduli (*gameEngine*). Skaņas efekti tiek atskaņoti, kad spēles loģika to norāda, piemēram, balons tiek noklikšķināts un sprāgst.

1. Grafiskās lietotāja saskarnes un spēles loģikas datu atkarība:

* Grafiskā lietotāja saskarnes modulis (MainWindow) saņem informāciju par spēles stāvokli no spēles loģikas moduļa (gameEngine). Tas attiecas uz rezultātu attēlošanu un spēles stāvokļa atjaunināšanu ekrānā.

# Moduļu saskarnes apraksts

## Moduļu saskarne

Dotajā spēlē ir trīs galvenie moduļi: Lietotāja saskarnes modulis UI (MainWindow), Spēles loģikas modulis (gameEngine), un Skaņas modulis (MediaPlayer). Zemāk šo moduļu saskarnes apraksts:

1. Lietotāja saskarnes modulis UI (MainWindow):

* Atbildība:
  + Nodrošina grafisko lietotāja saskarni spēlei.
  + Reaģē uz lietotāja darbībām, piemēram, pogas noklikšķināšana.
* *Interface:*
  + *canvasKeyIsDown(object sender, KeyEventArgs e):* Metode, kas apstrādā taustiņu nospiešanu un nodrošina spēles restartu.
  + *PoPBalloons(object sender, MouseButtonEventArgs e):* Metode, kas apstrādā peles klikšķi un reaģē uz balonu sprādzieniem.

1. Spēles loģikas modulis (gameEngine):

* Atbildība:
  + Pārvalda galveno spēles loģiku un stāvokli.
  + Nosaka balonu ģenerēšanas un kustības loģiku.
  + Izlemj par spēles notikumu izraisīšanu un rezultātu atjaunināšanu.
* Interface:
  + *startGame():* Metode, kas sāk jaunu spēli un inicializē spēles sākuma stāvokli.
  + *ResetGame():* Metode, kas izsaucas, lai atjaunotu spēli pēc tās beigām vai restartētu.
  + *gameEngine(object sender, EventArgs e):* Metode, kas tiek izsaukta regulāri, lai atjauninātu spēles stāvokli.

1. Skaņas modulis (*MediaPlayer*):

* Atbildība:
  + Pārvalda skaņas efektu atskaņošanu.
  + Reaģē uz spēles notikumiem, piemēram, balona sprādzieniem.
* *Interface:*
  + Nav īpašas saskarnes metodes, bet tiek aktivizēts ar citiem moduļiem, kad ir nepieciešams atskaņot skaņas efektus.

# Detalizētais projektējums

## Moduļu detalizētais projektējums

Moduļu projektēšanā izmantošu gan lēmumu tabulas gan arī pseidokodu.

### Lēmumu tabula

Nākamajās tabulā tiks attēlota spēles lēmumi

6.1.tabula

Lēmumu tabula

|  |  |  |  |
| --- | --- | --- | --- |
| *Space* taustiņš nospiests | Peles kreisais taustiņš nospiests | Spēle aktīva *(Game active)* | Darbība |
| nē | - | nē | Nedari neko |
| jā | - | nē | Atjaunot spēli |
| nē | jā | jā | Uzspridzināt balonu |
| jā | jā | jā | Uzspridzināt balonu un spēlēt skaņu |

### Pseidokods

Tālāk aprakstīšu moduļus ar pseidokoda palīdzību.

Inicializē spēles mainīgos un objektus

Iestatījumi:

ātrums = 2

intervāls = 90

skaits = 0

i = 0

neizšautiebaloni = 0

spēlesAktīva = false

Uzstādi fona attēlu un sāc spēli:

Izsauc funkciju: IzveidoJaunuSpēli()

Funkcija: IzveidoJaunuSpēli()

Notīra ekrānu no esošajiem objektiem

Uzstāda sākuma vērtības mainīgajiem

Sāk spēles taimeri

Uzstāda spēles aktīvumu uz true

Funkcija: AtjauninaSpēli(katra 20 milisekundēs)

Attēlo rezultātu ekrānā

Atjauno balonu pozīcijas un ātrumu

Pārbauda, vai ir laiks ģenerēt jaunu balonu

Ja jā, izveido jaunu balonu un uzstāda ģenerēšanas intervālu

Pārvieto esošos balonus uz augšu un sānos

Pārbauda, vai balons ir ārpus ekrāna

Ja jā, palielina neuzsprāgušo balonu skaitu

Funkcija: UzliktSkaņu(balonu sprādzienā)

Atskaņo "pop\_sound.mp3"

Lietotāja darbību apstrāde:

Ja taustiņš "Space" nospiests un spēle nav aktīva

Izsauc funkciju: IzveidoJaunuSpēli()

Ja peli noklikšķina un spēle ir aktīva

Pārbauda, vai klikšķis trāpījis uz balona

Ja jā, izsauc funkciju: UzliktSkaņu(balonu sprādzienā)

un noņem balonu no ekrāna, palielina rezultātu

Pamatprogrammas cilpa:

Katrā ciklā izsauc funkciju: AtjauninaSpēli()

Pārtrauc cilpu, ja ir pārsniegts maksimālais neuzsprāgušo balonu skaits

vai sasniegts noteikts rezultāts

## Datu detalizētais projektējums

Datu detalizētajā projektējumā tiek parādīts mainīgo apraksts.

6.4. tabula

Autorizācijas moduļa mainīgo apraksts

|  |  |  |
| --- | --- | --- |
| Nosaukums | Datu tips | Nolūks |
| Score | int | Spēlētāja rezultāts. |
| IncreaseScore | Metode | Palielina spēlētāja rezultātu par norādītu punktu skaitu. |
| ResetScore | Metode | Atiestata spēlētāja rezultātu uz nulli. |
| BalloonID | int | Unikāls identifikators balonam |
| BalloonType | int | Veida identifikators balonam (1-5). |
| PositionX | double | X koordināta ekrānā. |
| PositionY | double | Y koordināta ekrānā. |
| IsPopped | bool | Norāda, vai balons ir saplīsis. |
| Pop | Metode | Atzīmē balonu kā saplīsis. |
| Move | Metode | Pārvieto balonu uz jaunām koordinātēm ekrānā. |

# Atsauces

1. Alberta Koledža, Programmatūras inženierija, lekciju materiāli.