Vidzemes augstskola

Inženierzinātņu fakultāte

**Spēle “Pong”**

KURSA DARBS

Autors: Rihards Zīverts

Darba vadītājs: Andris Fjodorovs

Valmiera 2019

**Saturs**

[Ievads 3](#_Toc8817889)

[1. SFML bibliotēka 4](#_Toc8817890)

[1.1. SFML bibliotēkas uzstādīšana 4](#_Toc8817891)

[2. Spēle “Pong” 7](#_Toc8817892)

[2.1. Player.h 7](#_Toc8817893)

[2.2. Player.cpp 8](#_Toc8817894)

[2.3. Ball.h 9](#_Toc8817895)

[2.5. Main.cpp 11](#_Toc8817896)

[3. Izmantotās bibliotēkas 13](#_Toc8817897)

[3.1. Bibliotēku pielietošana programmā 13](#_Toc8817898)

[3.1.1. SFML/Graphics.hpp 13](#_Toc8817899)

[3.1.2. sstream 13](#_Toc8817900)

[4. Programmas darbības prezentācija 15](#_Toc8817901)

# Ievads

Programmēšanas kursa C++ ietvaros tika izveidota spēle “Pong”, kurā izmantotas līdz šim kursā neaplūkotas bibliotēkas sstream un SFML. Kursa darba izveidē izmantotas kā teorētiskās studenta zināšanas, tā arī internetā pieejamā literatūra.

# SFML bibliotēka

Lai izveidotu spēli “Pong”, tika izmantota bezmaksas atvērtā koda SFML (Simple and Fast Multimedia Library) bibliotēka, kas paredzēta, lai nodrošinātu lietotnes lietotāja saskarni (API) dažādiem multivides komponentiem datorā. Bibliotēka veidota C++ programmēšanas valodā, taču tā ir pieejama arī tādās programmēšanas valodās, kā C, Java, .NET, Python, Ruby, kā arī daudzās citās.

## 1.1. SFML bibliotēkas uzstādīšana

Kursa darba izveidei tika izmantota programmatūra “Studio Code 2017”. Lai tajā integrētu SFML bibliotēku, jāveic daži uzdevumi:

* Jāatrod piemērota SFML bibliotēka

Tā kā tiek izmantota “Studio Code” 2017. gada versija, jāizmanto arī atbilstošā gada SFML bibliotēkas versija, kas šajā gadījumā ir “Visual C++ 15” 32 bitu versija. 32 bitu versija tiek izmantota, lai spēle veiksmīgi darbotos arī uz zemākas izšķirtspējas datoriem.

* Iestatījumu konfigurēšana

Kad bibliotēka sekmīgi izveidota, tai jāizveido mape, kurā uzglabāsies visi bibliotēkas faili. Izveidojot jauno projektu, jāatver projekta uzstadījumi, kurā jāveic sistēmas konfigurēšana, lai bibliotēka tiktu apvienota ar konkrēto projektu. *(Skatīt Attēls 1.)*

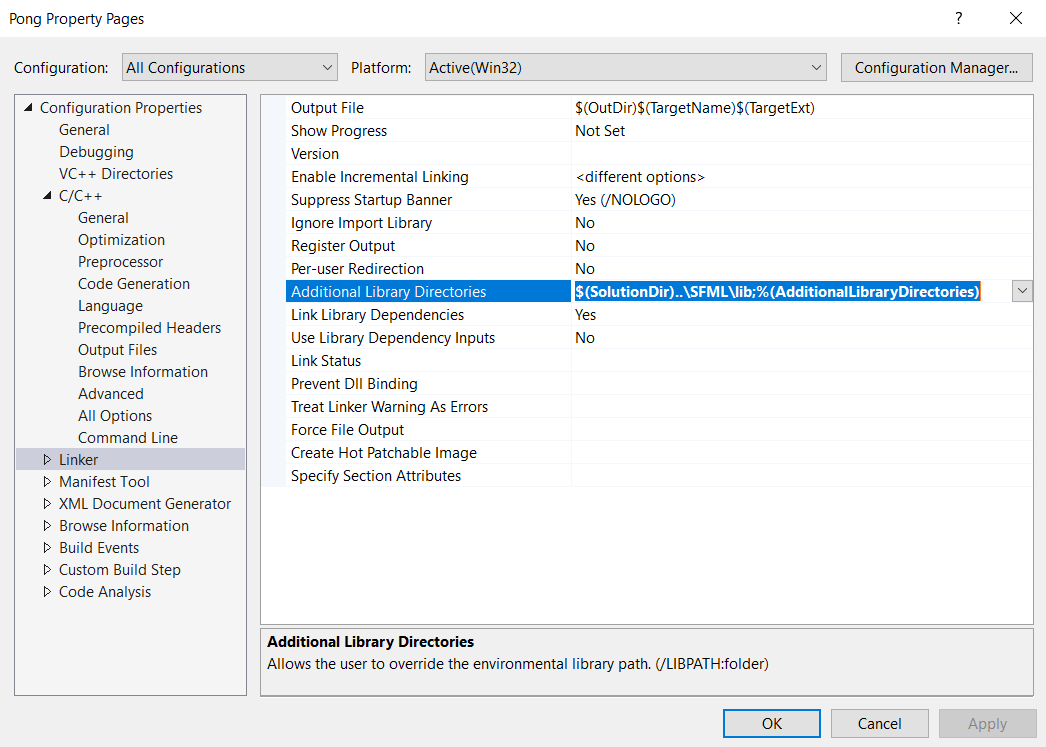
A screenshot of a social media post

Description automatically generated

Attēls 1

Iestatījumos pie konfigurācijas jānorāda “All Configurations”, tālāk sadaļā “C/C++” jāizvēlas “General”, kur savukārt iekš "Additional include Directories" jānorāda ceļš uz bibliotēkas mapi “include” - *$(SolutionDir)..\SFML\include*.

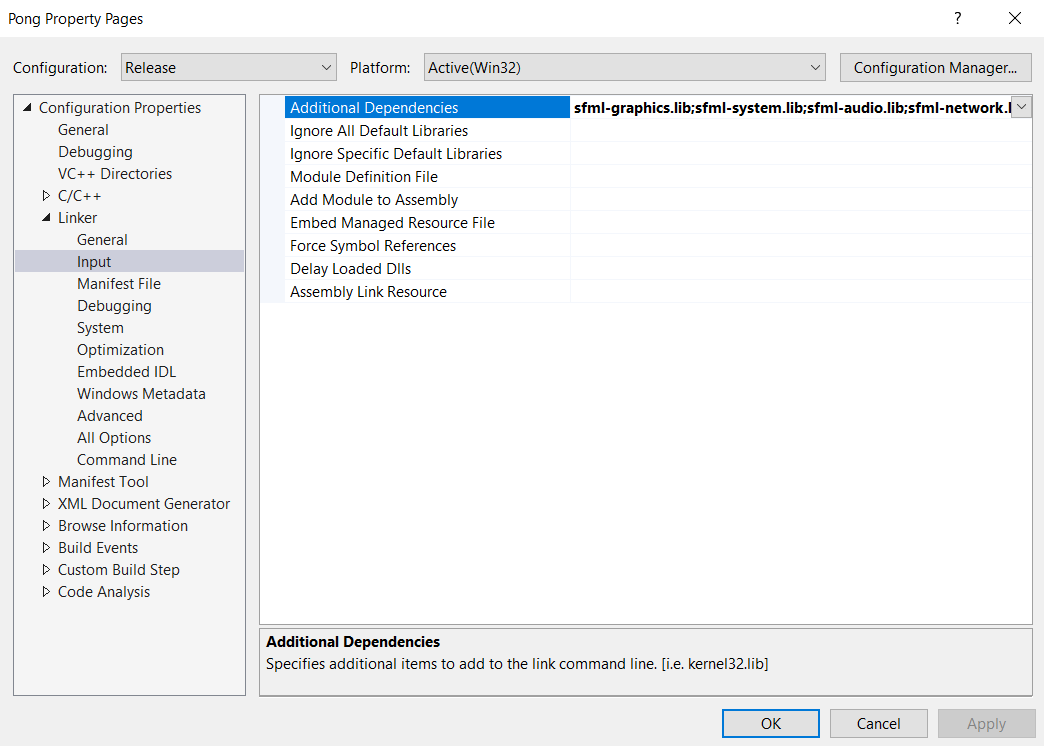
Tālāk jāizvēlas sadaļa “Linker”, kur tālāk jādara tie paši soļi, kas “C/C++” sadaļā, taču “include” vietā jābūt ceļam uz “lib” mapi - *$(SolutionDir)..\SFML\lib*. *(Skatīt attēls 2.)*



Attēls 2

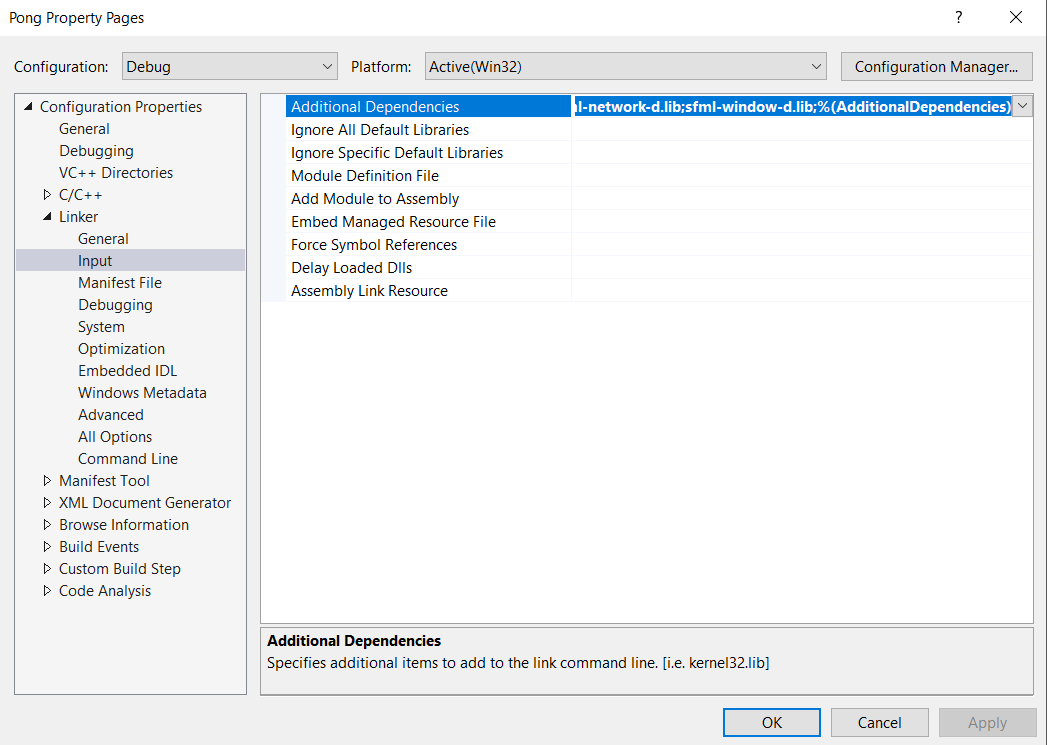
Kad iepriekš minētie soļi izdarīti, jānomaina konfigurācija uz “Release”, kur arī jāizvēlas “Linker” sadaļa, taču tālāk “General” vietā jāizvēlas “Input”, kur iekš “Additional Dependencies” savukārt jānorāda sekojošie faili *(skatīt Attēls 3)*:

* sfml-graphics.lib
* sfml-system.lib
* sfml-network.lib
* sfml-window.lib
* sfml-audio.lib



Attēls 3

Pēdējais solis ir ļoti līdzīgs iepriekšējam solim. Konfigurācijā jānorāda “Debug”, tāpat jāizvēlas “Linker” sadaļas “Input” un iekš “Additional Dependencies” jānorāda iepriekš norādītais saraksts, katra faila beigās pievienojot “-d”. *(Skatīt attēls 4.)*



Attēls 4

# Spēle “Pong”

## 2.1. Player.h

Sākotnēji tiek izveidots *header* fails “Player”, kurš tiek apvienots ar SFML bibliotēkas Graphics.hpp failu. Player.h deklarē spēlētāja vērtības:

* Vector2f pos

Vector2f ir SFML bibliotēkas klase, kas definē vektoru ar divām koordinātām (x un y). To var izmantot, lai attēlot jebko, kam ir divas dimensijas – izmērs, punkts, ātrums, utt.

Pos šajā gadījumā satur x un y koordinātas.

* Konstruktors Player

Konstruktors satur tādu pašu nosaukumu kā klase. Tas pieņem divus parametrus startX un startY, kas raksturo spēlētāja atrašanās vietas koordinātes.

* RectangleShape ply

RectangleShape ir klase, kas raksturo taisnstūri jeb šajā gadījumā – spēlētāju. Šī klase pārmanto visas funkcijas no SFML bibliotēkas Transformable klases kā arī no Shape klases.

Ply ir ekrānā redzamais spēlētāja objekts – taisnstūris.

* FloatRect getPos()

Funkcija, kas atgriež četrus punktus, kuri raksturo spēlētāja taisnstūri.

* void moveUp()

Funkcija, kas neko neatgriež. Paredzēta, lai, turot taustiņu, spēlētājs varētu virzīties uz augšu.

* void moveDown()

Funkcija, kas neko neatgriež. Paredzēta, lai, turot taustiņu, spēlētājs varētu virzīties uz leju.

* void update()

Funkcija, kas neko neatgriež. Paredzēta, lai atjaunotu informāciju par spēlētāja atrašanās vietu.

## 2.2. Player.cpp

Šajā failā tiek definētas funkcijas, kas atrodas Player.h failā.

* Plyer::Player(float startX, float startY)

Konstruktors, kas tiek izsaukts, kad tiek izveidots objekts. Mainīgie startX un startY norāda uz objekta koordinātām – attiecīgi uz x un y ass.

* ply.setSize(sf::Vector2f(75, 1))

Redzamajam spēlētāja objektam “ply” piešķir izmērus “75” un “1”, izmantojot Vector2f, kas, kā iepriekš minēts, definē vektoru ar divām koordinātām (x un y).

* Ply.setPosition(pos)

Redzamajam spēlētāja objektam “ply” norāda pozīciju “pos”, izmantojot funkciju setPosition().

* FloatRect Player::getPos()

Definējot funkciju getPos(), atgriežamā vērtība ir ply.getGlobalBounds(), kas atgriež spēlētāja vadītā objekta “ply” četru stūru koordinātes, izmantojot funkciju getGlobalBounds().

* RectangleShape Player::getPly()

Definējot funkciju getPly(), atgriežamā vērtība ir spēlētāja objekts “ply” jeb taisnstūris.

* void Player::moveLeft()

Funkcija, kas, lietotājam, nospiežot klaviatūras taustiņu, ļauj pārvietoties pa kreisi.

* void Player::moveRight()

Funkcija, kas lietotājam, nospiežot klaviatūras taustiņu, ļauj pārvietoties pa labi.

* void Player::update()

Funkcija, kas atjaunina spēlētāja objekta “ply” pozīciju “pos”, izmantojot funkciju setPosition().

## 2.3. Ball.h

Veidojot “Ball” klasi, sākotnēji tiek izveidots “header” fails, kurš, tāpat kā Player.h, satur mainīgos un funkcijas, no kā sastāv konkrētā klase.

* Vector2f pos

Vector2f ir SFML bibliotēkas klase, kas definē vektoru ar divām koordinātām (x un y). To var izmantot, lai attēlot jebko, kam ir divas dimensijas – izmērs, punkts, ātrums, utt.

Pos šajā gadījumā satur x un y koordinātas.

* RectangleShape bal

RectangleShape raksturo bumbas objektu “bal”.

* Float xSpd = .8f

Bumbas ātrums pa x asi.

* Float ySpd =.8f

Bumbas ātrums pa y asi.

* Konstruktors Ball

Konstruktors satur tādu pašu nosaukumu kā klase. Tas pieņem divus parametrus startX un startY, kas raksturo bumbas atrašanās vietas koordinātes.

* RectangleShape getBall()

Atgriež ekrānā redzamo objektu – bumbu.

* FloatRect getPos()

Funkcija, kas kontrolē bumbas atrašanās vietu.

* float getXSpd()

Atgriež bumbas ātrumu pa x asi.

* void rebsOnTop()

Funkcija, kura tiek izsaukta ik reizi, kad bumba atsitas pret loga augšu.

* void rebs()

Funkcija, kura tiek izsaukta ik reizi, kad bumba atsitas pret loga sāniem.

* void rebsOnBot()

Funkcija, kura tiek izsaukta, kad bumba sasniedz loga apakšu.

* void update()

Funkcija, kas atgriež bumbas pašreizējo atrašanās vietu.

**2.4. ball.cpp**

* Konstruktors Ball

Konstruktors satur tādu pašu nosaukumu kā klase. Tas pieņem divus parametrus startX un startY, kas raksturo bumbas atrašanās vietas koordinātes.

* pos.x = startX

Bumbas sākuma atrašanās vieta uz x ass.

* pos.y = startY

Bumbas sākuma atrašanās vieta uz y ass.

* bal.setSize(sf::Vector2f(6, 6));

Redzamajam spēlētāja objektam “bal” piešķir izmērus “6” un “6”, izmantojot Vector2f, kas, kā iepriekš minēts, definē vektoru ar divām koordinātām (x un y).

* bal.setPosition(pos)

Redzamajam spēlētāja objektam “ply” norāda pozīciju “pos”, izmantojot funkciju setPosition().

Pārējā ball.cpp daļā tiek definētas Ball.h esošās funkcijas.

## 2.5. Main.cpp

Main ir galvenā klase, kurā atrodas funkcija “main”, ar kuru sākas spēles kompilēšana. Šajā klasē jau reāli tiek izmantotas iepriekš definētās funkcijas un argumenti.

* int windowWidth = 1440

Spēles loga platums.

* Int windowHeight() = 900

Spēles loga augstums.

* RenderWindow window(VideoMode(windowWidth, windowHeight), "Kursa darbs", sf::Style::Titlebar | sf::Style::Close)

RenderWindow ir galvenā klase grafiskajā modulī. Tā definē operētājsistēmai logu, kuru var zīmēt, izmantojot citas grafiskā moduļa klases.

Konkrētajā koda rindā tiek izveidots spēles logs ar iepriekš minētajiem izmēriem – 1440 x 900 pikseļu lielumā. Loga nosaukums ir “Kursa darbs”.

* Font font

Tiek izveidots fonts tekstam, kurš tiks izmantots koda vēlākajā daļā.

* font.loadFromFile("Technology.ttf")

No mapes, kurā atrodas programmas faili, tiek ielādēts fonta izskats “Technology.ttf”.

* Text teksts;

Tiek izveidots Text objekts “teksts”, kurš izmantos iepriekš minēto fontu, izmantojot funkciju setFont(font). Tāpat tekstam ar funkciju setPosition(510.f, 850.f) tiek norādīta atrašanās vieta, setCharacterSize(30) norāda teksta izmēru, setFillColor(sf:Color::White) norāda teksta krāsu.

* int res = 0

Tiek izveidots int tipa mainīgais, kurš uzglabās spēlētāja rezultātu, kamēr tas spēlē spēli.

* Int prevRes = 0

Izveidots int tipa mainīgais prevRes, kurš saglabās spēlētāja pēdējo rezultātu. Izejot no spēles, šis mainīgais atkal būs 0.

* while (window.isOpen())

Izveidots cikls, kurš darbojas, kamēr spēles logs ir atvērts, uz ko norāda iekavās esošā funkcija window.isOpen(). Šis cikls sastāv no vairākiem notikumiem, kas saistīti ar to, kur spēlē atrodas bumba un kādu taustiņu nospiež spēlētājs.

* std::stringstream rezultats

Tiek izveidots stringstream objekts “rezultats”, kas tiek izmantots, lai kopā varētu apvienot skaitliskās vērtības ar rakstzīmēm, kas galā izveido vienu kopīgu virkni. Tas ļauj dinamiski parādīt spēlētāja rezultātu.

* window.clear()

Notīra visu spēles logu pirms tiek attēlots nākamais kadrs.

* window.draw(player.getPly())

Tiek zīmēts spēlētāja objekts.

* window.draw(ball.getBall())

Tiek zīmēts bumbas objekts.

* window.draw(teksts)

Tiek zīmēts teksta objekts.

* window.display()

Attēlo uz ekrāna to, kas uzzīmēts.

# Izmantotās bibliotēkas

Kursa darbā izmantotas sekojošās bibliotēkas:

* <SFML/Graphics.hpp>;
* <sstream>;

## 3.1. Bibliotēku pielietošana programmā

### 3.1.1. SFML/Graphics.hpp



Attēls 5

Player.h izmantotā *Vector2f* klase, kā jau iepriekš minēts, ir vienkārša klase, kas definē matemātisko vektoru ar divām koordinātēm (x un y). Savukārt *RectangleShape* ir speciāla klases forma, kas, pārmantojot citu klašu īpašības, attēlo taisnstūri.



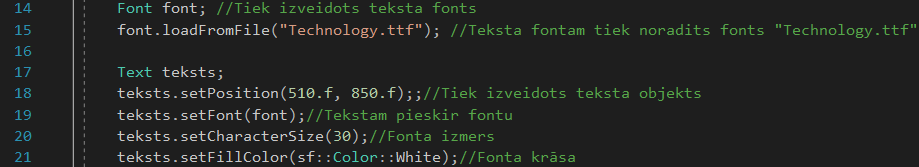
Attēls 6

*Rect* ir klase, ko izmanto, lai izlīdzinātu objekta pozīciju. *Rect* ir šablons, kuru var izmantot ar jebkuru skaitļu tipu. Šajā gadījumā tas ir Float, kā rezultātā veidojas *FloatRect* tipa funkcija getPos().



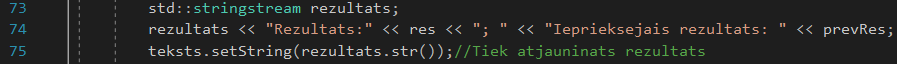
Attēls 7

*RenderWindow* klase tiek izmantota, lai izveidotu spēles logu, bet *VideoMode* klase nosaka spēles loga parametrus.



Klases *Font* un *Text* attiecīgi veido to, par ko noraāda klašu nosaukumi – fonta objektu un teksta objektu, kurš izmanto fontu.

### 3.1.2. sstream



Stringstream ir klase, kas darbojas ar virknēm. Šīs klases objekti izmanto virknes buferi, kas sastāv no secīgām rakstzīmēm. Rakstzīmes var ievietot vai dinamiski iegūt, izmantojot jebkuru darbību, kas atļauta gan ieejas, gan izejas datu plūsmās.

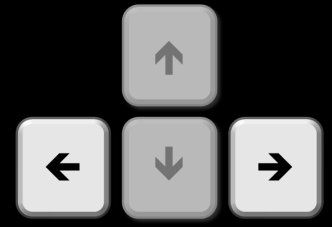
# Programmas darbības prezentācija

Atverot programmu, atveras 1440x900 pikseļu liels spēles logs, kura izmērus nav iespējams mainīt. Logā redzams spēlētāja objekts līnijas formā virs teksta “Rezultats:0 Ieprieksejais rezultats:0”. *(Skatīt attēls 8)*, kā arī krītoša bumba, kura spēlētājam jāatsit.



Attēls 8

Spēlē iespējams izmantot trīs taustiņus. Izmantojot *Attēls 9* izceltos klaviatūras taustiņus, lietotājs var pārvietoties. Izmantojot taustiņu “Esc”, lietotājs iziet no spēles, taču citi taustiņi nepilda funkcijas.



Attēls 9

Spēles loga lejasdaļā redzams esošais rezultāts un iepriekšējais rezultāts jeb rezultāts, kāds spēlētājam bijis, pirms tas zaudējis. *(Skatīt attēls 3.)*



Attēls 10