Skateboard Game Qt

Iosif Daniel-Ionel

16 ianuarie 2022

Rezumat

Jocul a fost creat ca proiect in cadrul laboratorului de "Tehnici avasate de programare" de la Universitatea de Medicină, Farmacie, Științe și Tehnologie „George Emil Palade" din Târgu Mureș și are ca scop învățarea frameworkului Qt.

Cuprins

1 Introducere

Jocul constă în evitarea obstacolelor ce apar in fața playerului, acesta fiind nevoit să sară pentru a le evita, fiecare obstacol trecut înseamnă un punct acumulat. Pentru a începe jocul fiecare jucător trebuie să își creeze un cont care va fi salvat intr-o mini Bază de date, având asociat cel mai bun scor al său de la crearea contului, de asemenea jocul dispune de un Leaderboard pentru vizualizarea unui clasament cu cei mai buni jucatori si cu cel mai bun scor al lor. În continuare vor fi explicate fiecare clasă și fiecare metodă a clasei respective.

2 Clasa Clouds

Clasa **Clouds** este o clasa tipică Qt deoarece moștenește clasa **QObject** pentru a permite lucrul cu semnale și sloturi(eng. signals and slots) , dar și clasa **QGraphicsItemGroup** care este defapt un container ce permite stocarea mai multor iteme ca unul singur.
Scopul clasei este de a anima norii în fundalul jocului.

2.1 Constructorul

Constructorul clasei **Clouds** inițializează cele trei atribute principale ale clasei(cloud1, cloud2, cloud3) cu o imaginii ai unui nori, acestea fiind de tipul QPixmap , iar cu ajutorul funcției de generare a unui număr din Qt (QRandomGenerator) decide ordinea norilor care urmează a fi adăugați in container. Tot în interiorul constructorului este inițializată și pornită animația (QPropertyAnimation) care are scopul de a mișca grupul de nori in fundalul jocului de la o valoare setată la altă valoare setată într-un interval de timp stabilit. In plus animația va vi ștearsă din scenă, iar pointerul său va fi dealocat în momentul in care animația s-a terminat.

2.2 Metodele x() și setX(qreal x)

Aceste metode sunt getterul si setterul(slot) atributului `m_x` care sunt folosite de către macroul `Q_PROPERTY` pentru a misca grupul de nori.

2.3 Metoda freezeClouds()

Este defapt un slot , unde este semnalizată oprirea animației, daca aceasta este activă.

3 Clasa Obstacle

Aceasta este o clasă care moștenește **QObject** pentru a permite lucrul cu semnale și sloturi, dar și clasa **QGraphicsItemGroup** care este defapt un container ce permite stocarea mai multor iteme ca unul singur. Scopul clasei este de a mișca un grup de 2 obstacole in scena jocului pe care playerul este nevoit să le evite pentru a trece mai departe.

3.1 Constructorul

Constructorul acestei clase inițializează câmpurile obstacle1, obstacle2 cu o imagine a unei obstacol, și decide distanța dintre cele două, apoi le adaugă într-un grup. În plus tot aici se inițializează atributele pastObst1 si pastObst2 care semnifică daca playerul a trecut de obstacole. Pe langă asta, tot aici se creează xAnimation(QPropertyAnimation) care va mișca grupul de obstacole în scenă

de la o valoare setată la o valoare setată , într-un timp setat, iar când animația s-a terminat va fi ștearsă din scenă și este dealocat pointerul grupului(itemului).

3.2 Metoda collidesWithSkate()

Această metodă salvează cele 2 obstacole într-o listă, iar apoi parcurge lista pentru a verifica coliziunea dintre obstacol si skateItem.

3.3 Metoda freezeObstacle()

Aceasta metoda este defapt un slot care oprește animația.

3.4 Metoda x() si metoda setX(qreal x)

Metoda **x()** este un getter al macroului `Q_PROPERTY` care este folosit pentru mișcarea animației.
Metoda **setX(qreal x)** este un slot tot al macroului `Q_PROPERTY` și este folosită pentru mișcarea animației în scenă. În această metodă se verifică dacă playerul a trecut de obstacol și se incrementează scorul, in cazul în care afirmația este adevărată. Pe lângă asta tot în această metodă în cazul coliziunii cu playerul a unuia dintre obstacole se emite un semnal (**collideJump()**) pentru a semnaliza oprirea jocului.

4 Clasa Road

Este clasa care simulează mișcarea drumului prin intermediul unei animații. La fel ca clasele prezentate mai sus, aceasta moștenește clasa QObject și QGraphicsItemGroup.

4.1 Constructorul

Constructorul acestei clase este locul unde atributul **road** de tipul **QPixmap** este inițializat cu imaginea unui drum și este adăugat la grup, iar prin intermediul unei animații acesta străbate scena dintr-un punct ales până in alt punct ales, într-o perioadă stabiltă astfel simulând parcurgerea drumului de către player. În momentull in care animația s-a terminat aceasta este ștearsă din scenă, iar pointerul acesteia este dealocat.

4.2 Metoda x() și metoda setX()

La fel ca la celelalte clase prezentate anterior, aceaste metode sunt specifice macroului `Q_PROPERTY` care este folosit pentru a mișca grupul drumului.

4.3 freezeRoad()

Aceasta metoda este un slot care oprește animația.

5 Clasa SkateItem

Aceatsă clasă reprezintă playerul propriuzis care este de altfel un **QGraphics-PixmapItem** , dar moștenește și clasa **QObject** pentru a permite utilizarea semnalelor și a sloturilor.

5.1 Metoda jumping() si metoda updateJumpPixmap()

Metoda **jumping()** este metoda care simulează (prin intermediului metodei **updateJumpPixmap()**) saltul playerului. Această primă metodă inițializează atributul **jumpPosition**(de tip enum) cu o valoare primă de început care reprezintă o imagine a palyerului, iar pe urmă pornește un timer care apelează metoda **updateJumpPixmap()**) la un interval de timp setat. Această ultimă metodă verifică de fiecare dată când este aplelată prin atributul **jumping()** dacă trebuie sa simuleze saltul playerului prin schimbarea imaginii și a poziției acesteia în scenă(pornind în același timp si un sunet pentru salt), este folosit atributul **jumpPosition** care este actualizat la fiecare apel al metodei, acesta știind care imagine trebuie aleasă pentru a simula jumpul. In final este oprit sunetul si timerul pentru jump, este setat atrbutul **jumping()** pe false(adica animația pentru săritură a luat sfârșit) și este apelată metoda pentru skate.

5.2 Metoda skating() si metodaupdatePixmap()

Metoda **skating()** este metoda care simuleaza mișcarea playerului(prin intermediul metodei **updatePixmap()**) . Acesta inițializează atributul **skatingPosition**(de tip enum) cu o primă valoare care indică imaginea care trebuie aleasă pentru player, apoi este apelat metoda **updatePixmap()** prin intermediul unui timer, la un interval setat de timp. Metoda **updatePixmap()**) schimbă imaginea playerului in funcție de atributul **skatingPosition**, care indică ce imagine trebuie alesă(acesta fiind actualizat la fiecare apel al metodei), astfel realizandu-se animația.

5.3 Metoda fall()

Această metodă schimbă imaginea playerului(skateItem-ului).

5.4 getStateJumping()

Această metodă returnează starea playerlui(daca simuleaza saltul), fiind folosită pentru fixarea bugului in cazul in care se incearcă să se spameze saltul.

5.5 setStateJumpingTrue()

Această metodă seteaza starea playerului, prin atributul **isJumping** , insemnând că acesta sare.

5.6 freezeSkate()

Această metodă oprește timerul care simulează animația de jump sau de skating daca unul dintre acestea este activ.

5.7 Constructorul

Aici este pornită animația de skating prin apelul metodei **skating()**, dar și sunetul jocului.

6 Clasa Scene

Această clasă moștenește clasa **QGraphicsScene** și o instanță a acesteia urmează a fi setată ca scenă in **graphicsView** din **widget.cpp**, aici fiind locul unde vor fi adăugate toate celelalte iteme și tot aici realizandu-se acțiunea jocului.

6.1 Metoda setupCloudsTimer()

Aceasta este metoda unde este creat timerul pentru nori și unde se adaugă la o perioadă de timp o instanță a clasei **Clouds** in scenă(căreia i se specifică si prioritatea in plan).

6.2 Metoda setupObstaclesTimer()

Aceasta este metoda unde este creat timerul pentru obstacole și unde la o perioadă specificată de timp se creează un obiect al clasei **Obstacle**, căruia i se setează prioritatea in plan și care este adaugat in scenă. Tot în expresia lambda care se apelează în momentul in care timpul timerului a expirat, se verifică dacă obstacolul respectiv a emis un semnal care sa anunțe coliziunea cu playerul. În acest caz se apelează metoda **fall()** a obiectului **skateItem**, se emite un semnal pentru afișarea meniului, se afișează grafica de game over și se oprește muzica.

6.3 Metoda setupRoadTimer()

Aceasta este metoda unde este creat timerul pentru drum și unde se adaugă la o perioadă de timp o instanță a clasei **Road** in scenă(căreia i se specifică si prioritatea in plan).

6.4 Metoda setupRoad()

Această metodă este folosită pentru a crea o instanță a unui drum, căruia i se setează prioritatea in plan și care este adăugat in scenă.

6.5 Metoda setupClouds()

Această metodă este folosită pentru a crea o instanță a unui nor, căruia i se setează prioritatea in plan și care este adăugat in scenă.

6.6 Metoda setupSkate()

Această metodă este folosită pentru a crea o instanță a unui skateItem, căruia i se setează prioritatea in plan și care este adăugat in scenă.

6.7 Metoda gameOver()

Această metodă este apelată la terminarea unui joc, iar aici se verifică dacă scorul maxim obținut in timpul jocului curent este mai mare decât scorul salvat în baza de date, in cazul in care afirmația precedentă este adevărată se actualizează scorul jucatorului în baza de date. Tot aici se îngheață toate itemele care există in scenă și se opresc timerele obiectelor din scenă.

6.8 Metoda cleanScene()

Această metodă curăță scena, ștergând din ea toate itemele și dealocând memoria pointerilor itemelor respective.

6.9 Metoda showGameOverGraphics()

Această metodă afișează grafica de game over.

6.10 Metoda hideGameOverGraphics()

Această metodă ascunde grafica de game over dealocând memoria pointerului **gameOverPix**, care este o imagine.

6.11 Metoda showScoreGame()

Această metodă afișează scorul sub forma de text html.

6.12 Metoda hideScoreGame()

Această metodă șterge din scenă grafica scorului.

6.13 Metoda updateUser(QString *cuser)

Această metodă actualizează numele userului curent și extrage din baza de date scorul maxim al acestuia.

6.14 Metoda setScoreGame()

Această metodă actualizeaza scorul.

6.15 Metoda setBackgroundMusic()

Această metodă setează muzica de pe fundal.

6.16 Metoda getGameOn()

Returnează valoarea atributului **gameOn**.

6.17 Metoda setGameOn(bool value)

Setează valoarea atributului **gameOn**.

6.18 Metoda incrementScore()

Această metodă crește scorul curent și verifică dacă acesta este mai mare ca scorul obținut in sesiunea curentă de joacă, actualizând scorul maxim din sesiunea curentă daca este nevoie. În plus se actualizează si grafica scorului , iar la fiecare număr divizibil cu 15 se repornește muzica.

6.19 Metoda keyPressEvent(QKeyEvent *event)

Acesta este un key listener care la apăsarea tastei "space" va verifica dacă nu cumva playerul sare deja și dupa va modifica valoarea câmpului care ne indică daca playerul este deja in animația de jump, pornește animația de salt și oprește timerul pentru skating al playerlui.

6.20 Metoda startGame()

Această metodă verifică daca jocul este pornit pentru prima dată și in caz afirmativ va seta drumul, va porni timerul pentru drum, va seta nori, va porni timerul pentru nori, va inițializa playerul, va porni timerul pentru obstacole, va porni muzica de pe fundal și va seta câmpul ce reține daca jocul este pornit pentru prima dată pe true și va inițializa scorul . În cazul in care atributul **gameOn** ne indică că jocul a fost pornit cel puțin odată, atunci va curăța scena și va face exact aceleași lucruri prezentate mai sus.

6.21 Constructorul

În constructot se inițializează userul curent cu "" și se setează muzica de pe fundal.

7 Clasa SqlLiteHandler

Este o clasă utilitară, care conține metode pentru interacținea cu baza de date.

7.1 Metoda createAccount(QString password,QString username, int scor

Această metodă creează un account, iar in cazul in care baza de date nu există creează si baza de date.

7.2 selectUsername(QString username)

Această metodă selectează userul și în cazul în care există returnează userul, iar în caz contrar returnează "".

7.3 selectPassword(QString username)

Această metodă selectează parola și în cazul în care există returnează parola, iar în caz contrar returnează "".

7.4 updateScore(QString username, int scor)

Această metodă modifică scorul playerului în baza de date.

7.5 getScore(QString username)

Această metodă returnează scorul, iar în cazul în care nu îl găsește returnează 0.

8 Clasa Widget

Această clasă moștenește QWidget și este o clasă de bază in care care adaugăm un atribut graphicsView în care setăm scena cu o instanță a clasei Scene. Aici avem toate butoanele, labeluri și lineEdituri.

8.1 Constructorul

În constructor setăm dimensiunile widgetului, adăugăm scena, intițialăm userul curent cu "", ascundem butoanele de creare, inițializăm toate icoanele pentru butoane și setăm timerul pentru butoane.

8.2 Metoda hideButtons()

Această metodă ascunde butoanele.

8.3 Metoda showButtons()

Această metodă afișează butoanele.

8.4 Metoda setupButtonsTimer()

Aceasta este o metodă care conține un timer, care tot la o perioadă de timp verifică dacă scena nu a emis un semnal pentru afișsarea butoanelor, iar in caz afirmativ afișează butoanele.

8.5 `Metoda on_startButton_clicked()`

Această metodă este un button listener care pornește jocul, în cazul în care există un user conectat, altfel afișează un mesaj într-un label, care dispare după o perioadă de timp.

8.6 `Metoda on_startButton_clicked()`

Această metodă este un button listener care închide jocul.

8.7 `Metoda on_clasamentButton_clicked`

Această metodă este un button listener care afișează clasamentul.

8.8 `Metoda on_CreateButton_clicked()`

Această metodă este un button listener care crează un cont dacă datele introduse sunt valide.

8.9 Metoda hiddeCreateButtons()

Această metodă ascunde butoanele de creare a contului.

8.10 `Metoda on_Cancel_clicked()`

Această metodă ascunde butoanele de creare a contului și afișează celelalte butoane.

8.11 `Metoda on_Create_clicked()`

Această metodă este un button listener care ne creează un cont dacă datele introduse sunt valide și dacă usernamul este unic.

8.12 `Metoda on_LoginButton_clicked()`

Această metodă este un button listener care ne conectează la cont dacă contul există.

8.13 `Metoda on_Logout_clicked()`

Această metodă ne deconectează de la cont.

8.14 Metoda getTopList())

Această metodă se conectează la baza date și ne adaugă toate datele într-un list widget.

8.15 `Metoda on_closeTopScore_clicked()`

Această metodă este un button listener care ascunde tabela cu clasamentul.

9 Clasa main

Aceasta este clasa principală în care threadul acesteia ne ține deshis jocul, apelând în buclă pană la închiderea programului jocul.