Tema 1 - Romburi vs hexagoane

- **Responsabili:** Andrei Voicu, Mihnea Tudor, Anca Cristea, Cristian Lambru
- **Lansare:** 30 octombrie 2023
- **Termen de predare:** 19 noiembrie 2023, ora 23:59
- **Regulament:** Regulament general
- **Notă: Orice informație ce nu a fost acoperită în acest document este la latitudinea voastră!**

În cadrul temei 1, veți avea de implementat un joc similar cu "Plants vs Zombies" [https://youtu.be/XENla8M3910?feature=shared&t=314], dar adaptat pentru a conține o geometrie mult mai simplă :) . Puteți viziona un mic demo construit pe baza framework-ului de laborator, care acoperă cerințele și are un aspect vizual minimal.

De asemenea, puteți descărca demo-ul aici pentru a-l testa și a înțelege mai clar comportamentul de joc.

Implementarea din demo-ul pus la dispoziție nu identifică corect poziția cursorului, în situația în care se modifică dimensiunea ferestrei. Aceasta rămâne de implementat la latitudinea voastră și este considerată cerință bonus :) .

Pe scurt, jocul se va desfășura în cadrul a 3 linii, cu 3 coloane fiecare. Din partea dreaptă, din afara ecranului, apar la intervale aleatoare de timp inamici "răi" de tip hexagon ce parcurg o linie până în partea stângă a ecranului. Odată ce un hexagon parcurge toată linia, jucătorul pierde o viață :(. La 3 vieți pierdute, jocul este pierdut.

Pentru a combate hexagoanele rele :(, jucătorul are posibilitatea de a plasa într-una din cele 3 celule de pe fiecare linie, un romb salvator care lansează proiectile ce se deplasează în partea dreaptă, de-alungul liniei în care se află rombul ce a lansat proiectilul. Odată întâlnit un hexagon, proiectilul îi creează daune și la 3 proiectile atinse, hexagonul dispare și situația este salvată pentru moment <3.

Cu toate acestea, jucătorul are 2 inconveniente :(. El nu poate plasa oricâte romburi dorește, deoarece fiecare romb plasat costă un anumit număr de steluțe. Acestea apar la intervale aleatoare în scenă și jucătorul trebuie să le strângă. În plus, fiecare romb poate răni doar un anumit tip de hexagon, pe baza culorii pe care o are fiecare.

Sunteți gata să ajutați jucătorul să învingă hexagoanele "rele" :) ?

Reguli generale de joc

Plasare romb în scena de joc

Jocul are un comportament bazat pe culori. Sunt 4 tipuri de triplete romb-proiectil-inamic, fiecare triplet de aceeași culoare de bază. În exemplul nostru, am ales culorile portocaliu, albastru, galben și mov. Voi puteți să alegeți alte culori.

Vom numi romburi, din acest moment până la final, elementele plasate de jucător.

În interfața grafică cu utilizatorul, sau GUI, în partea stânga-sus a ecranului, se regăsesc 4 chenare, cu câte un tip de romb în fiecare. Prin procesul de drag & drop, descris mai jos, se selectează un anumit tip de romb și se plasează într-una din celulele de joc valide, în care nu se regăsește niciun alt romb.

Jucatorului i se va permite selecția unui tip de romb, doar dacă acesta are resurse suficiente pentru plasarea tipului respectiv. Numărul de resurse stabilit de noi este:

- O steluță pentru rombul de culoare portocalie
- 2 steluțe pentru rombul de culoare albastră
- 2 steluțe pentru rombul de culoare galbenă
- 3 steluțe pentru rombul de culoare mov

Steluțele apar în scenă la intervale aleatoare. Noi am ales să apară câte 3 odată. Jucătorul poate selecta o steluță când cursorul se află în interiorul ei și apasă butonul stânga de la mouse. Odată selectată o steluță, i se adaugă la numărul de steluțe pe care îl are strâns până în acel moment.

Comportament romburi și inamici

Din partea dreaptă, la intervale aleatoare de timp, se deplasează un inamic de-alungul unei linii alese aleator. Inamicul face parte din unul din cele 4 tipuri posibile. În situația în care un romb de același tip se regăsește într-una din cele 3 celule ale liniei, acesta începe să lanseze proiectile la intervale regulate. În situația în care un romb **nu are un inamic de același tip cu el pe linie**, rombul respectiv **nu lansează proiectile**, chiar dacă pe linie sunt inamici de alte tipuri.

GUI

Jocul are o interfață grafică pentru utilizator ce conține, pe lângă chenarele cu care interacționează pentru plasarea romburilor, și informații suplimentare, menite să îl ajute pe jucator. Anume, în această interfață se regăsesc:

- Numărul de puncte de viață rămase. Inițial, jocul pleacă de la 3
- Numărul de steluțe strânse până la momentul respectiv de joc
- Sub chenarul fiecărui tip de romb, sunt desenate mai multe steluțe, specific numărului necesar pentru plasarea tipului respectiv de romb

Dacă doriți indicații pentru implementarea anumitor mecanici de joc, mai multe detalii se pot găsi în secțiunea dedicată, sub barem.

Barem

În cadrul temei, puteți implementa o formă simplă de joc, doar prin realizarea cerințelor de bază. Mecanismul de plasare a unui romb poate fi implementat prin identificarea selecției cu butonul stânga de la mouse a unei celule valide și plasarea unui romb în interiorul celulei. În această situație, nu este necesar să se țină cont de numărul de resurse strânse până la un moment dat, ceea ce înseamnă că se permite jucătorului să plaseze oricâte romburi dorește în celulele disponibile :) .

Funcționalități de bază (150 puncte)

- Construcție elemente scenă (60p total)
  - Desenare geometrie scenă de joc 20p
  - Desenare geometrie romb 10p
  - Desenare geometrie inamic 10p
  - Desenare geometrie steluță 20p
- Animații (40p total)
  - Deplasare inamic pe linie 5p
  - Deplasare și rotație proiectil 15p
  - Animație dispariție romb 10p
  - Animație dispariție inamic 10p
- Comportament de joc (50p in total)
  - Detecția selecției unei celule de joc prin apăsarea butonului stânga al mouse-ului și plasarea unui romb în celulă 10p
  - Apariție inamici la intervale aleatoare 5p
  - Detecția faptului că un inamic a traversat în totalitate o linie 5p
  - Apariție proiectil de lânga un romb în momentul în care există inamic pe linie 10p
  - Detecție coliziune proiectil-inamic 10p
  - Detecție coliziune romb-inamic 10p

Funcționalități avansate (75p total)

- Comportament drag & drop (25p in total)
  - Desenare în GUI a 4 chenare ce conțin dreptunghiuri încadratoare și cele 4 tipuri de romburi 5p
  - Detecția selecției și afișarea rombului pe ecran la poziția cursorului, în perioada în care butonul stânga de la mouse este apăsat 15p
  - Detecția faptului că nu s-a mai apăsat butonul stânga de la mouse, în momentul în care poziția mouse-ului este în interiorul unei celule valide 5p
- Cuantificarea numărului de vieți rămase, inițial sunt 3, și afișarea lor in GUI 10p
- Detecția selecției de ștergere a unui romb, prin apăsarea butonului dreapta de la mouse în celula în care se află deja un romb 5p
- Gestionare resurse (20p in total)
  - Apariție steluțe pe ecran, la anumite intervale 2.5p
  - Detecția selecției unei steluțe prin apăsarea butonului stânga al mouse-ului și dispariția ei de pe ecran 10p
  - Desenarea în GUI, sub fiecare tip de romb, a numărului de steluțe necesar pentru plasarea tipului respectiv 2.5p
  - Permiterea plasării unui romb, doar când numărul de steluțe este suficient 2.5p
  - Cuantificarea steluțelor strânse și consumate, desenarea în GUI a numărului de steluțe existente la momentul desenării 2.5p
- Gestionare comportament de joc bazat pe culori (15p in total)
  - Desenarea în GUI a 4 tipuri de romburi, diferențiate printr-o culoare specifică pentru fiecare 2.5p
  - Apariția la anumite intervale a 4 tipuri diferite de inamici, diferențiați printr-o culoare specifică pentru fiecare 2.5p
  - Apariția proiectilului de lânga un romb ce are o anumită culoare, doar în momentul în care există pe linia lui un inamic de aceeași culoare 5p

- Apariție proiectil cu aceeași culoare ca cea a rombului de lânga care a apărut 2.5p
- Detecția coliziunii proiectil-inamic, doar în situația în care proiectilul are aceeași culoare ca cea a inamicului 2.5p

Detalii de implementare

Construcție elemente vizuale

Aveți libertate totală asupra felului în care se construiesc elementele vizuale necesare pentru joc, ce vor fi și descrise în continuare. Mai exact:

- Puteți construi geometria unui element dintr-o singură rețea de triunghiuri, model 2D, care să descrie toată forma, prin vârfuri și indici. Pentru a atribui mai ușor coordonate vârfurilor, vă recomandăm să utilizați platforma https://www.geogebra.org/calculator [https://www.geogebra.org/calculator].
- Puteți construi geometria unui element vizual prin afișarea mai multor figuri geometrice de bază, pătrat, triunghi, pentru care să aplicați transformări de translație, scalare sau rotație.

În vederea creării animațiilor, vă recomandăm să fiți atenți asupra formei de construcție. O stea sau un hexagon cu centrul în originea axelor de coordonate este mai ușor de prelucrat :) .

Scena de joc

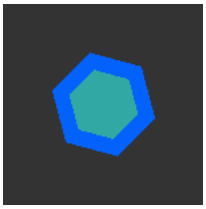
Scena de joc este compusă din mai multe pătrate amplasate pe o grilă cu 3 linii și 3 coloane. Aceasta poate fi vizualizată în imaginea de mai jos. Culoarea aleasă de noi pentru fiecare pătrat ce reprezintă o celulă din grilă, este verde, dar voi puteți alege orice altă culoare, atâta timp cât poate fi distinsă față de culoarea de fundal. În partea stânga a grilei, am adăugat un dreptunghi roșu, pentru a marca zona în care se consideră că inamicii iau un punct de viață :(în situația în care ajung la ea.

Romb

Geometria elementelor ce sunt plasate în celulele scenei de joc, descrisă mai sus, este formată dintr-un romb și un dreptunghi în partea din dreapta a rombului. O imagine individuală a acestui element poate fi văzută mai jos.

Inamic

Geometria inamicului este reprezentată de 2 hexagoane de dimensiuni diferite, ce au centrul în același punct. Hexagonul mic trebuie să fie desenat în fața hexagonului mare. Culorile celor 2 hexagoane trebuie să fie diferite. O imagine cu un astfel de rezultat poate fi vizualizată mai jos.



Pentru a desena o formă geometrică în fața altei forme, puteți folosi componenta z a coordonatelor vârfurilor. Pe ecran rămâne desenată geometria ce are componenta z cea mai mare.

Steluța

Geometria proiectilelor, a resurselor și a unui set de elemente din GUI este sub forma unor steluțe cu 5 colțuri, de culori diferite. Imaginea de mai jos prezintă geometria steluței.

GUI

În partea de sus este afișată o interfață grafică pentru utilizator. Interacțiunea cu această interfață, prin mecanismul de drag & drop, este explicată mai jos. Elementele vizuale ale interfeței sunt:

- 4 chenare așezate în linie, ce conțin geometria romburilor, cu o culoare spefică tipului. Fiecare chenar are un dreptunghi încadrator. Sub fiecare chenar se afișează mai multe steluțe, specific numărului de resurse steluță ce trebuie consumate pentru a plasa tipul de romb specific chenarului.
- Punctele de viață rămase pentru jucator la un moment dat, reprezentate de pătrate de culoare roșie, așezate în linie.
- Steluțele colectate, afișate în linie.

Interfața grafică pentru utilizator poate fi văzută în imaginea de mai jos.

Animații

Pentru toate animațiile din joc, utilizați parametrul `deltaTime` pentru a crea animații independente de numărul de cadre desenate pe secundă.

Deplasare de-alungul liniilor de joc

Pentru deplasarea proiectilelor și a inamicilor de-alungul unei linii de joc, se utilizează o transformare de translație. Suplimentar, proiectilul cu formă de steluță va avea o transformare de rotație, **în sens orar**, față de centru, în același timp cu deplasarea.

Dispariție romb din joc

În momentul în care un romb dintr-o celulă de joc, plasat anterior de către jucător, trebuie să dispară din scenă, rombul trebuie să aibă o animație de micșorare față de centru, până ajunge la scara (0, 0). Acest efect se poate obține printr-o transformare de scalare.

Situațiile în care un romb este eliminat din joc sunt următoarele:

- În momentul în care un inamic s-a intersectat cu un romb, rombul dispare prin această animație, iar inamicul își continuă deplasarea spre stânga.
- În momentul în care utilizatorul apasă butonul dreapta de la mouse și cursorul este deasupra unei celule în care se află un romb, rombul dispare prin această animație din celulă.

Dispariție inamic din joc

În momentul în care un proiectil se intersectează cu un inamic, acesta din urmă dispare din scenă printr-o animație de micșorare față de centrul comun al celor două hexagoane care formează geometria inamicului. Această animație poate fi realizată, similar cu cea de micșorare a rombului, printr-o transformare de scalare. Proiectilul dispare și el la intersecția lui cu inamicul, doar că nu realizează nicio animație, doar nu se mai afișează de la cadrul următor după detecția intersecției.

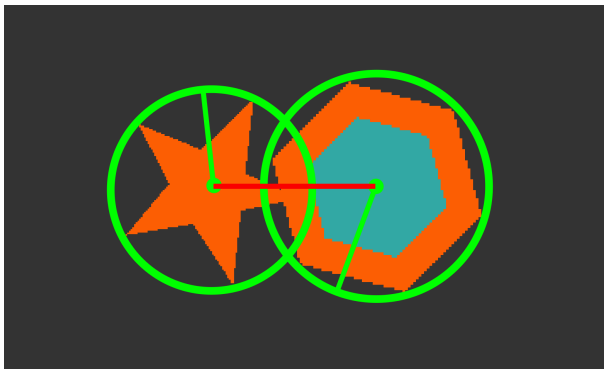
Detecție coliziuni

Mecanismul de detecție a coliziunilor între diverse obiecte din scenă reprezintă un feature esențial în cam orice joc la care vă puteți gândi. În Pong [https://www.youtube.com/watch?v=fiShX2pTz9A] trebuie verificată coliziunea mingii, atât cu ecranul jocului, cât și cu "paleta" fiecărui jucător. În jocurile Mario [https://www.youtube.com/watch?v=QLF0FXcW25E], trebuie verificată coliziunea dintre jucător și toate obiectele înconjurătoare: cuburi, inamici, power-ups etc. În jocul nostru, avem nevoie de o implementare puțin mai simplificată a coliziunilor.

Există două interacțiuni esențiale între obiecte, care necesită verificarea coliziunilor:

- între hexagon și romb - când hexagonul avansează la stânga până întâlnește un romb, moment în care rombul este distrus
- între proiectil și hexagon - când un romb lansează la dreapta un proiectil și se intersectează, pe drum, cu un hexagon

Deoarece formele obiectelor sunt mai complexe, iar implementarea unor coliziuni care să coincidă cu forma obiectului este mult prea complicată, vom folosi cercuri (mai multe detalii se pot găsi aici [https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection#circle_collision]). Pe scurt, aceste cercuri oferă o aproximare a formei unui obiect: un disc al cărui centru coincide cu centrul obiectului. Această abordare se implementează ușor și necesită mai puțină putere de procesare. O imagine care demonstrează funcționarea unei coliziuni între cercuri este următoarea:



Astfel, se va defini această formă de coliziune pentru proiectile, hexagoane și romburi. Cercurile **nu** vor fi desenate pe ecran, ci vor fi folosite doar pentru a crea logica jocului. Pentru a detecta coliziunea între 2 obiecte, trebuie să urmăriți pașii:

1. Aflarea coordonatelor centrelor celor două obiecte
2. Stabilirea razei fiecărui cerc
3. Aflarea distanței dintre cele două centre (hint: glm::distance() [https://registry.khronos.org/OpenGL-Refpages/gl4/html/distance.xhtml])
4. Verificarea dacă distanța este mai mică decât suma razelor cercurilor

Aveți grijă cum definiți formele geometrice: proiectil, romb, hexagon. Dacă ați definit obiectul original astfel încât centrul formei geometrice să se afle în punctul (0, 0), atunci centrul cercului va coincide cu centrul formei. Altfel, este posibil să intervină niște deplasări care trebuie luate în calcul. De asemenea, fiți atenți la eventuale rotații și translații ale obiectelor în scenă.

Interacțiune jucător

Un alt aspect esențial este interacțiunea jucătorului cu obiectele din scenă, în acest caz, prin intermediul mouse-ului. Avem două astfel de interacțiuni:

- colectarea de resurse, prin apăsarea unui buton pe mouse
- plasarea romburilor în celule, prin intermediul drag & drop
- ștergerea romburilor din celule

Modul de funcționare este similar cu cel de la coliziunea obiect-obiect, fiind nevoie să determinăm marginile fiecărui obiect. În schimb, în acest caz, trebuie determinată poziția curentă a cursorului pe ecran, după care verificăm dacă se află în interiorul marginilor obiectului.

În funcțiile callback care sunt apelate la mișcări sau apăsări de buton ale mouse-ului ('OnMouseMove()', 'OnMouseBtnPress()'), coordonatele cursorului sunt date ca parametri, astfel că poziția cursorului este (mouseX, mouseY). Cele două valori sunt numere **întregi** ce reprezintă poziția cursorului în spațiul de vizualizare (pe fereastra jocului) pe orizontală, respectiv pe verticală.

Poziția cursorului pe fereastra jocului are ca punct de origine **colțul din stânga sus**, în timp ce spațiul de joc/logic are ca punct de origine **colțul din stânga jos**. Din acest motiv, pentru a folosi coordonata y a cursorului pentru obiectele din scena de joc, trebuie să utilizăm următorul calcul: *y_scena_joc = 720 - y_cursor*. Valoarea 720 reprezintă înălțimea ferestrei.

Pentru detecția situației în care poziția cursorului se află în interiorul unui element vizual din joc, vom folosi dreptunghiuri încadratoare aliniate cu axele (axis-aligned bounding boxes, mai multe detalii aici [https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection#axis-aligned_bounding_box]) pentru obiecte. Pe scurt, ne vom imagina un dreptunghi (în locul cercului de la coliziunile obiect-obiect) cu laturile paralele cu axele Ox și Oy, și cu centrul în cel al obiectului.

Pentru a verifica apăsarea unui buton de la mouse, când cursorul se află în interiorul unui obiect, trebuie să:

1. Transformăm coordonatele cursorului de pe ecran în coordonatele scenei de joc

2. Calculăm marginile dreptunghiului, în urma eventualelor aplicări de translații și scalări
3. Verificăm dacă poziția cursorului este între marginile dreptunghiului

Drag & drop

Pentru a plasa un romb pe una dintre celule, acesta trebuie selectat cu mouse-ul din bara romburilor disponibile, după care trebuie tras deasupra unei celule **libere**. O celulă liberă este o celulă care nu este deja ocupată de un romb.

Pentru a crea acest mecanism, trebuie realizați următorii pași:

1. Detecția apăsării butonului stâng al mouse-ului
2. Detecția poziției cursorului într-una dintre căsuțele cu romburi disponibile (pentru care avem suficiente resurse)
3. Randarea rombului la poziția cursorului, în fiecare frame, cât timp butonul rămâne apăsat
4. Detecția eliberării (release) al aceluiași buton de mouse (cel stâng)
5. Detecția poziției cursorului într-una dintre celulele libere
6. Plasarea rombului în celulă
7. Reducerea resurselor disponibile, în funcție de costul rombului

În cazul în care utilizatorul apasă pe un romb pentru care nu are suficiente resurse, algoritmul se oprește la pasul 2. De asemenea, dacă butonul mouse-ului este eliberat într-o poziție invalidă (oriunde în afara unei celule libere), desenarea rombului la poziția cursorului este oprită, iar resursele **nu** sunt consumate (practic, plasarea rombului într-o pozitie invalidă nu are niciun efect).

Exemple de funcționalități bonus

Orice funcționalitate suplimentară implementată (care nu este inclusă în cerințele obligatorii) poate fi considerată ca punctaj bonus dacă este suficient de complexă. Funcționlitățile bonus se iau în considerare doar dacă funcționalitățile obligatorii au fost realizate.

- Geometrie mai complexă față de cea din cerința de bază pentru elementele vizuale din joc
- Utilizarea transformării fereastră-poartă pentru detecția corectă a poziției cursorului, la orice dimensiune a ferestrei
- Romburi mai speciale (ex: romburi care generează, la un anumit interval, resurse care se pot culege; romburi care explodează; romburi care atacă mai multe linii de-odata)
- Hexagoane mai speciale (ex: hexagoane care își schimbă culoarea la fiecare 'hit'; hexagoane mai slabe sau puternice, cu indicatori vizuali)

- Creșterea dificultății pe măsură ce avansați în timp
- Implementare de "niveluri" cu durate stabilite, avansarea la următorul nivel în urma terminării celui anterior
- Sistem de "waves" (ca în jocul original): în anumite momente ale jocului, un grup mare de hexagoane este trimis către romburi
- Resurse care cad din cer și se așază undeva în scenă
- Lawnmowers, ca în Plants vs Zombies
- Orice aduce îmbunătățiri vizuale jocului

Dacă vreți să funcționeze corect detecția în cazul redimensionării ferestrei de joc, se va aplica următoarea abordare:

Inițial, dacă începeți cu scheletul din laboratorul 3, spațiul logic coincide 1:1 cu spațiul de vizualizare, adică toate pozițiile obiectelor coincid cu coordonata respectivă de pe ecran. Marginea dreaptă a spațiului logic va coincide cu marginea dreaptă a ferestrei de joc, adică lungimea ferestrei (resolution.x), iar marginea de sus cu înălțimea ferestrei (resolution.y).

Dacă vrem să redimensionăm fereastra, este nevoie să aplicăm o transformare fereastră-poartă din noua rezoluție în cea inițială. Mai exact, daca notăm noua rezoluție cu coordonatele (new_width, new_height), vrem să traducem noul spațiu de coordonate, cuprins între (0, 0) - (new_width, new_height), în cel inițial (corespunzător celui logic), cuprins între (0, 0) - (1280, 720). De notat este faptul că (1280, 720) este rezoluția inițială a ecranului, și este setată în funcția *main()*.

Această transformare trebuie aplicată pentru oricare alt fragment de cod sau logică ce se folosesc de marginile ecranului.

Întrebări și răspunsuri

Pentru întrebări vom folosi forumurile de pe moodle. Orice nu este menționat în temă este la latitudinea fiecărui student!

Notare

Baremul este orientativ. Fiecare asistent are o anumită libertate în evaluarea temelor (de exemplu, să dea punctaj parțial pentru implementarea incompletă a unei funcționalități sau să scadă pentru hard coding). Același lucru este valabil atât pentru funcționalitățile obligatorii, cât și pentru bonusuri.

Tema trebuie încărcată pe moodle. Pentru a fi punctată, tema trebuie prezentată la laborator. Vor exista laboratoare speciale de prezentare a temelor (care vor fi anunțate).

Indicații suplimentare

Tema va fi implementată în OpenGL și C++. Este indicat să folosiți framework-ul și Visual Studio.

Pentru implementarea temei, în folderul **src/lab_m1** puteți crea un nou folder, de exemplu **Tema1**, cu fișierele **Tema1.cpp** și **Tema1.h** (pentru implementare POO, este indicat să aveți și alte fișiere). Pentru a vedea fișierele nou create în Visual Studio în Solution Explorer, apăsați click dreapta pe filtrul **lab_m1** și selectați **Add→New Filter**. După ce creați un nou filtru, de exemplu **Tema1**, dați click dreapta și selectați **Add→Existing Item**. Astfel adăugați toate fișierele din folderul nou creat. În fișierul **lab_list.h** trebuie adăugată și calea către header-ul temei. De exemplu: **#include "lab_m1/Tema1/Tema1.h"**

Arhivarea Proiectului

- În mod normal arhiva trebuie să conțină toate resursele necesare compilării și rulării
- Înainte de a face arhiva asigurați-vă că ați curățat proiectul Visual Studio:
    - Click dreapta pe proiect în **Solution Explorer → Clean Solution**

    - Ștergeți folderul **/build/.vs** (dacă nu îl vedeți, **este posibil să fie ascuns**)
- În cazul în care arhiva tot depășește limita de 50MB (nu ar trebui), puteți să ștergeți și folderul **/deps** sau **/assets** întrucât se pot adăuga la testare. Nu este recomandat să faceți acest lucru întrucât îngreunează mult testarea în cazul în care versiunea curentă a bibliotecilor/resurselor diferă de versiunea utilizată la momentul scrierii temei.

egc/teme/2023/01.txt · Last modified: 2023/11/03 20:25 by andrei.lambru