| Universitatea Tehnică Gheorghe Asachi din Iași  Facultatea de Automatică și Calculatoare  Departamentul de Automatică și Informatică Aplicată | **Proiect SBC** |
| --- | --- |

Echipă: Baltă Laurențiu Alexandru, Hîji Otilia Cristina

Grupa: 1405A

Dezvoltarea unui sistem decizional de reguli pentru rezolvarea unui cub Rubik

**Rezumat**

Aplicația pentru rezolvarea cubului Rubik este proiectată pentru a oferi o experiență interactivă și captivantă utilizatorilor pasionați de rezolvarea unui cub rubik. Prin intermediul mediului CLIPS, este definită structura bazei de fapte, faptele ajutătoare, dar și regulile ce ajută la realizarea fiecărui pas pentru a ajunge la scopul propus.

Sistemul expert conține reguli pentru realizarea mișcării necesare curente și reguli pentru alegerea mișcărilor următoare. În mediul Python sunt folosite două librării pentru realizarea și buna funcționare a interfeței, și anume: clipspy si pygame. Clipspy realizează interacțiunea dintre python și sistemul expert. Pygame este folosită pentru afișarea cubului într-o interfață GUI și informarea utilizatorului cu mișcările următoare pentru realizarea nivelului.

Interfața conține o zonă principala si una secundară. Zona principală ajută utilizatorii să vizualizeze cele 6 fețe ale cubului rubik. Cealaltă zonă conține la rândul ei o cutie text cu mesaj pentru utilizator, care ajută la ghidarea pas cu pas pentru rezolvare și un buton pentru realizarea pasului următor.

Aplicația utilizează un algoritm ușor de înțeles și are o interfață prietenoasă oferind utilizatorilor posibilitatea de a-și îmbunătăți abilitățile în rezolvarea cubului Rubik. Prin urmare, aplicația este o resursă valoroasă pentru pasionații de cub Rubik, oferindu-le o modalitate distractivă și eficientă de a-și dezvolta abilitățile în rezolvarea acestui puzzle complex.

# I. Enunț problemă.

Dezvoltarea unui sistem decizional de reguli pentru rezolvarea unui cub rubik.

Urmăm setul de pași în funcție de poziția curentă în următoarea ordine:

* Cruce piese albe cu culorile laterale corecte.
* Rezolvare baza, colțuri corecte (fata alba și culori fețe alăturate corecte)
* Rezolvare nivel de mijloc cu aducerea pe fiecare față a pieselor corecte cu fețele laterale potrivite
* Rezolvare cruce piese galbene cu culorile alăturate corecte
* Aducere colțuri în pozițiile corecte
* Intoarcerea colțurilor pentru a avea culoarea galbenă sus

# Obiectiv.

Rezolvarea unui cub rubik prin cunoașterea inițială a tuturor fețelor și a pozițiilor pieselor.

# Ipoteze:

* Se poate aduce o singură piesă la locul dorit în funcție de poziția sa.

# II. Descriere universul de discurs

## Concepte:

Fata, culoare , piesa, straturi(baza, mijloc, varf), colt, cruce, miscare

## Proprietăți:

Sus, jos, stanga, dreapta, fața, spate sunt fețele cubului.

Alb, galben, roșu, verde, albastru și portocaliu sunt culorile fetelor.

## Relații:

Stanga are culorile: alb, alb, alb, roșu, galben, roșu, albastru, albastru și albastru.

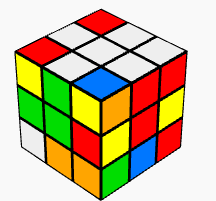
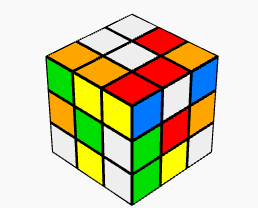
## Acțiuni:

Învârte stanga în sensul acelor de ceasornic. Învârte dreapta în sens invers acelor de ceasornic.

# III. Scenarii considerate și strategii de rezolvare

1. *Instanta 1*

Cubul are piesele asezate random si nu este creata crucea pe fața albă.  
Scopul este de a forma o cruce cu piesele albe și lateral piesele potrivite fețelor alăturate.



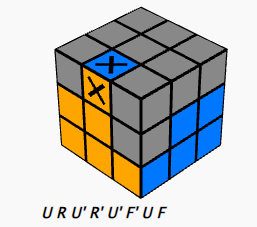
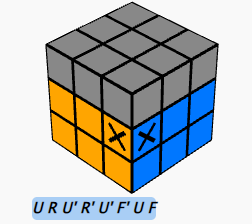
1. *Instanta 2*

Cubul are fețele cu următoarele culori:

* Jos are culorile: alb, alb, alb, alb, alb, alb, alb, alb și alb.
* Spatele are culorile: galben, verde, galben, portocaliu, rosu, albastru, roșu, roșu și roșu.
* Fața are culorile: verde, portocaliu, albastru, portocaliu, portocaliu, galben, portocaliu, portocaliu și portocaliu.
* Stanga are culorile: verde, rosu, rosu, rosu, verde, galben, verde, verde și verde.
* Dreapta are culorile: roșu, albastru, albastru, roșu, albastru, verde, albastru, albastru și albastru.
* Sus are culorile: portocaliu, galben, portocaliu, verde, galben, galben, galben, albastru și galben.

Pentru a aduce piesa din poziția sa inițială în cea dorită realizam urmatoarele mișcări:

* Învârte sus în sensul acelor de ceasornic.
* Învârte dreapta în sensul acelor de ceasornic.
* Învârte sus în sens invers acelor de ceasornic.
* Învârte dreapta în sens invers acelor de ceasornic.
* Învârte sus în sens invers acelor de ceasornic.
* Învârte fața în sens invers acelor de ceasornic.
* Învârte sus în sensul acelor de ceasornic.
* Învârte fața în sensul acelor de ceasornic.

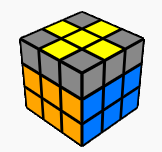
 

1. *Instanta 3*

Cubul are primele două layere complete și nu este formată crucea de pe fata de sus.

Scopul este de a realiza urmatoarele miscari pană se creează crucea de pe fața galbena:

* Învârte fața în sensul acelor de ceasornic.
* Învârte dreapta în sensul acelor de ceasornic.
* Învârte sus în sensul acelor de ceasornic..
* Învârte dreapta în sens invers acelor de ceasornic.
* Învârte sus în sens invers acelor de ceasornic.
* Învârte fața în sens invers acelor de ceasornic.

# IV. Definirea bazei de fapte

1. **Tipare de fapte**

(are <fata> <culoare-pozitie-1-1> <culoare-pozitie-1-2> <culoare-pozitie-1-3> <culoare-pozitie-2-1> <culoare-pozitie-2-2> <culoare-pozitie-2-3> <culoare-pozitie-3-1> <culoare-pozitie-3-2> <culoare-pozitie-3-3>)

1. **Descriere prin fapte nestructurate**

***Instanta 2:***

(are jos alb alb alb alb alb alb alb alb alb)

(are fata verde portocaliu albastru portocaliu portocaliu galben portocaliu portocaliu portocaliu)

(are stânga verde roșu roșu roșu verde galben verde verde verde)

(are dreapta roșu albastru albastru roșu albastru verde albastru albastru albastru)

(are sus portocaliu galben portocaliu verde galben galben galben albastru galben)

(are spate galben verde galben portocaliu roșu albastru rosu rosu rosu)

# V. Definirea bazei de reguli

Pentru a putea rezolva cubul folosind regulile de mai jos trebuie sa pornim cu primii doi pași deja rezolvati, asta însemnând sa avem fata alba rezolvata si primul nivel așezat corect.

Am împărțit regulile în două categorii:

* **reguli pentru alegerea următoarelor mișcări**

**yellow\_cross\_moves** -> această regulă se activează atunci când există un fapt în baza de cunoștințe că mutarea **yellow-cross** trebuie verificată și există în baza de fapte **"?has"** care reprezintă faptul că partea de sus a cubului conține anumite culori, inclusiv culoarea galbenă pe marginea superioară.După aceea, regulă folosește o serie de condiționale "if-then-else" pentru a verifica ce mutări trebuie făcute pentru a forma crucea galbenă.Dacă marginile superioare formează deja o cruce galbenă, atunci regula adaugă un nou fapt, și anume că crucea galbenă a fost completată, și afișează un mesaj pentru a indica acest lucru.Dacă marginile superioare nu formează o cruce galbenă, regulă verifică care dintre cele patru margini nu este încă pe locul său și determină ce mutări trebuie făcute pentru a poziționa marginea corectă. În funcție de caz, se pot realiza una sau mai multe mișcări F R U R' U' F' . Regula adaugă mutările necesare și afișează un mesaj pentru a indica aceste mișcări.

**green\_second\_layer\_moves ->** această regulă este responsabilă de rezolvarea celui de-al doilea strat al cubului R, în care trebuie să se potrivească piesele verzi cu cele albe de pe stratul de sus. Regula începe prin verificarea dacă primul strat albastru și colțurile albe sunt deja completate. Dacă nu este așa, atunci regula nu se va declanșa și va aștepta ca alte reguli să finalizeze aceste etape.Regula verifică apoi câteva condiții în stratul de mijloc, inclusiv dacă piesa de mijloc din față și cele două piese din mijlocul stratului din dreapta și stânga sunt verzi. Dacă aceste condiții sunt îndeplinite, atunci regula va marca etapa ca fiind finalizată și va afișa un mesaj corespunzător.În continuare, regula verifică câteva configurații posibile ale fațetei verzi și poziției pieselor de pe stratul de sus, și adaugă mișcările necesare pentru a poziționa corect piesele verzi. Aceste mișcări se vor executa ulterior de către programul care interpretează regulile și executa acțiunile necesare pentru a finaliza cubul.

**orange\_second\_layer\_moves ->**verifică dacă fața portocalie este plasată corect și, dacă nu, efectuează un set de mișcări pentru a o plasa corect. În mod similar, verifică dacă fețele stângă și dreaptă sunt plasate corect și, dacă nu, efectuează un alt set de mișcări pentru a le plasa corect. Apoi verifică bucățile rămase de pe fața portocalie pentru a determina dacă sunt necesare și alte mișcări.De asemenea, afișează un mesaj care indică care mișcări trebuie efectuate, dacă este cazul.

**red\_second\_layer\_moves->**verifică dacă fața roșie este poziționată corect și, dacă nu, efectuează un set de mișcări pentru a o plasa corect. În mod similar, verifică dacă laturile stânga și dreapta sunt poziționate corect și, dacă nu, efectuează un alt set de mișcări pentru a le plasa corect. Verifică apoi piesele rămase de pe fața roșie pentru a determina dacă sunt necesare mișcări suplimentare.De asemenea, afișează un mesaj indicând ce mișcări trebuie făcute, dacă este cazul.

**blue\_second\_layer\_moves->** acesta este un alt set de reguli de producție în care se verifică dacă nivelul doi este finalizat pentru fața albastră și, dacă nu, se efectuează o serie de mișcări pentru a-l finaliza. Regulile sunt similare cu cele pentru fața roșie și se concentrează pe verificarea dacă celelalte piese de pe fața albastră sunt în poziția corectă. Dacă nu, se efectuează mișcări suplimentare pentru a ajunge la starea finală.După ce toate mișcările necesare sunt efectuate, se afirmă faptul că nivelul doi este finalizat, iar un mesaj este afișat pentru a confirma acest lucru și pentru a indica care mișcări au fost necesare pentru a ajunge la această stare.

**second\_layer\_check->** verifica dacă al doilea nivel a fost completat corect. Aceasta face verificari pentru a vedea dacă fata din fata, fata din dreapta, fata din stanga și baza au fost orientate corect in al doilea strat. Daca condiția este îndeplinită, regula afirma un fapt **second-layer completed** și afișează un mesaj care indică faptul ca al doilea nivel a fost rezolvat.

**yellow\_edge\_moves ->** această regulă verifică starea marginilor galbene ale cubului. Regula prima dată verifică mișcările marginilor galbene. Apoi, verifică starea marginilor cubului pe fiecare față a acestuia.În cazul în care marginile galbene sunt completate, trebuie să verifice mișcările marginilor galbene și se afișează un mesaj că au fost rezolvate marginile galbene.Dacă marginile galbene nu sunt completate, sunt verificate anumite combinații de culori ale marginilor și se realizează anumite mișcări pentru a le muta în poziția corectă și se afișează un mesaj corespunzător. Dacă marginile nu sunt într-o poziție corectă, se realizează o anumită mișcare pentru a le muta în poziția corectă.

**yellow\_corners\_moves ->** verifică colțurile cubului care sunt galbene. Dacă colțurile galbene sunt completate, se activează faptul **yellow-corners completed** și se afișează un mesaj. În caz contrar, se verifică ce combinație de colțuri galbene poate fi realizată și se afișează mesajul corespunzător.

**orrient\_yellow\_corners\_moves ->**verifică starea cubului pentru a determina dacă colțurile galbene sunt orientate corect. Dacă sunt orientate corect, regula va șterge faptul curent și va afișa un mesaj care indică că această parte a rezolvării a fost finalizată. În caz contrar, regula va determina mișcările necesare pentru a orienta colțurile galbene și va afișa un mesaj care indică aceste mișcări.

**complete\_cube\_moves ->** aceasta are ca scop verificarea dacă cubul a fost complet rezolvat.Regula verifică dacă toate cele 6 fețe ale cubului au aceeași culoare, deci fiecare față trebuie să fie complet formată din piese cu aceeași culoare. În cazul în care toate fețele au aceeași culoare, regula **șterge** faptul **check-moves** **complete-cube** și adaugă un nou fapt **cube completed**, semnificând că cubul a fost complet rezolvat.În cazul în care cubul nu este încă complet rezolvat, regula adaugă o mișcare U, pentru a începe următoarea fază a rezolvării cubului Rubik.

* **reguli pentru realizarea fiecărei mutări**

**miscare\_R** -> pentru a efectua o mișcare R, trebuie să verifice mai întâi dacă există o astfel de mișcare în lista de mișcări. Apoi, verifică starea cubului pentru a vedea poziția pieselor față de celelalte piese înainte și după mișcare.Dacă toate condițiile sunt îndeplinite, atunci regula va retrage toate aceste fapte și va înlocui cu noile fapte, reflectând noua stare a cubului după mișcarea R

**miscare\_L ->** pentru a efectua o mișcare L, trebuie să verifice mai întâi dacă există o astfel de mișcare în lista de mișcări. Apoi, verifică starea cubului pentru a vedea poziția pieselor față de celelalte piese înainte și după mișcare.Dacă toate condițiile sunt îndeplinite, atunci regula va retrage toate aceste fapte și va înlocui cu noile fapte, reflectând noua stare a cubului după mișcarea L

**miscare\_U ->** pentru a efectua o mișcare U, trebuie să verifice mai întâi dacă există o astfel de mișcare în lista de mișcări. Apoi, verifică starea cubului pentru a vedea poziția pieselor față de celelalte piese înainte și după mișcare.Dacă toate condițiile sunt îndeplinite, atunci regula va retrage toate aceste fapte și va înlocui cu noile fapte, reflectând noua stare a cubului după mișcarea U

**miscare\_D->** pentru a efectua o mișcare D, trebuie să verifice mai întâi dacă există o astfel de mișcare în lista de mișcări. Apoi, verifică starea cubului pentru a vedea poziția pieselor față de celelalte piese înainte și după mișcare.Dacă toate condițiile sunt îndeplinite, atunci regula va retrage toate aceste fapte și va înlocui cu noile fapte, reflectând noua stare a cubului după mișcarea D.

**miscare\_F->** pentru a efectua o mișcare F, trebuie să verifice mai întâi dacă există o astfel de mișcare în lista de mișcări. Apoi, verifică starea cubului pentru a vedea poziția pieselor față de celelalte piese înainte și după mișcare.Dacă toate condițiile sunt îndeplinite, atunci regula va retrage toate aceste fapte și va înlocui cu noile fapte, reflectând noua stare a cubului după mișcarea F.

**miscare\_B->** pentru a efectua o mișcare B, trebuie să verifice mai întâi dacă există o astfel de mișcare în lista de mișcări. Apoi, verifică starea cubului pentru a vedea poziția pieselor față de celelalte piese înainte și după mișcare.Dacă toate condițiile sunt îndeplinite, atunci regula va retrage toate aceste fapte și va înlocui cu noile fapte, reflectând noua stare a cubului după mișcarea B.

**miscare\_R\_prim** -> pentru a efectua o mișcare R’, trebuie să verifice mai întâi dacă există o astfel de mișcare în lista de mișcări. Apoi, verifică starea cubului pentru a vedea poziția pieselor față de celelalte piese înainte și după mișcare.Dacă toate condițiile sunt îndeplinite, atunci regula va retrage toate aceste fapte și va înlocui cu noile fapte, reflectând noua stare a cubului după mișcarea R’.

**miscare\_L \_prim->** pentru a efectua o mișcare L’, trebuie să verifice mai întâi dacă există o astfel de mișcare în lista de mișcări. Apoi, verifică starea cubului pentru a vedea poziția pieselor față de celelalte piese înainte și după mișcare.Dacă toate condițiile sunt îndeplinite, atunci regula va retrage toate aceste fapte și va înlocui cu noile fapte, reflectând noua stare a cubului după mișcarea L’.

**miscare\_U\_prim ->** pentru a efectua o mișcare U’, trebuie să verifice mai întâi dacă există o astfel de mișcare în lista de mișcări. Apoi, verifică starea cubului pentru a vedea poziția pieselor față de celelalte piese înainte și după mișcare.Dacă toate condițiile sunt îndeplinite, atunci regula va retrage toate aceste fapte și va înlocui cu noile fapte, reflectând noua stare a cubului după mișcarea U’.

**miscare\_D\_prim->** pentru a efectua o mișcare D’, trebuie să verifice mai întâi dacă există o astfel de mișcare în lista de mișcări. Apoi, verifică starea cubului pentru a vedea poziția pieselor față de celelalte piese înainte și după mișcare.Dacă toate condițiile sunt îndeplinite, atunci regula va retrage toate aceste fapte și va înlocui cu noile fapte, reflectând noua stare a cubului după mișcarea D’.

**miscare\_F\_prim->** pentru a efectua o mișcare F’, trebuie să verifice mai întâi dacă există o astfel de mișcare în lista de mișcări. Apoi, verifică starea cubului pentru a vedea poziția pieselor față de celelalte piese înainte și după mișcare.Dacă toate condițiile sunt îndeplinite, atunci regula va retrage toate aceste fapte și va înlocui cu noile fapte, reflectând noua stare a cubului după mișcarea F’.

**miscare\_B\_prim->** pentru a efectua o mișcare B’, trebuie să verifice mai întâi dacă există o astfel de mișcare în lista de mișcări. Apoi, verifică starea cubului pentru a vedea poziția pieselor față de celelalte piese înainte și după mișcare.Dacă toate condițiile sunt îndeplinite, atunci regula va retrage toate aceste fapte și va înlocui cu noile fapte, reflectând noua stare a cubului după mișcarea B’.

# VI. Rezolvarea conflictelor

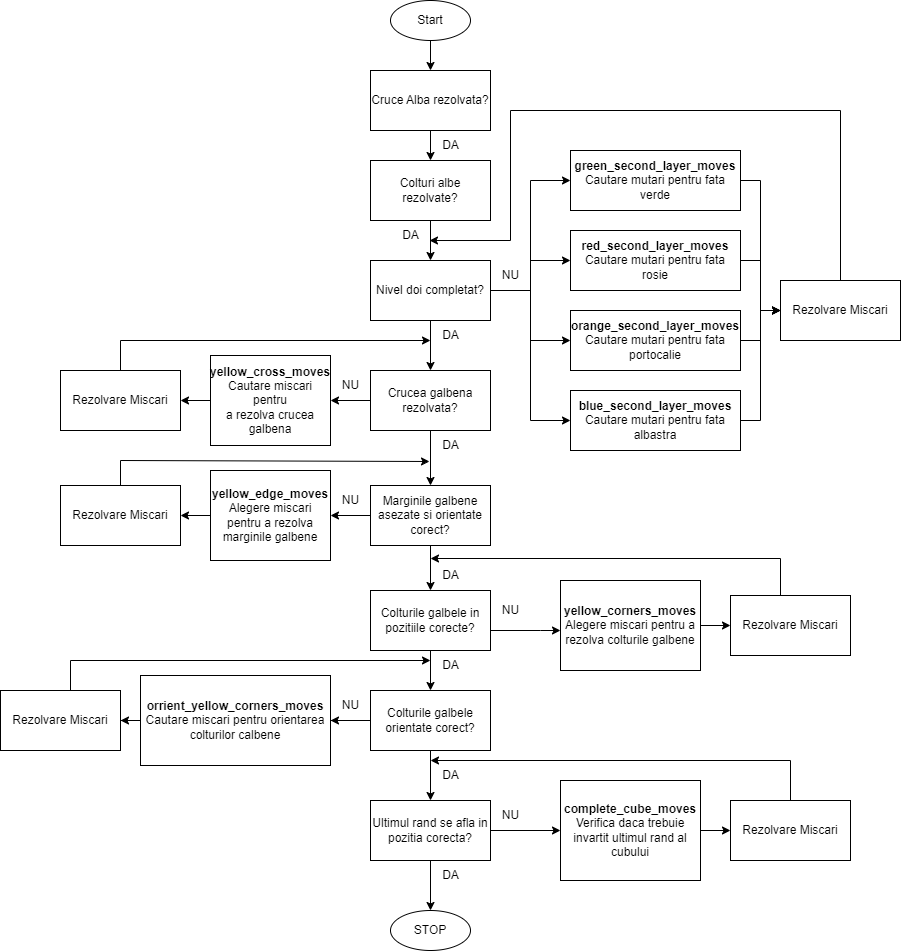
***1. Cazuri conflictuale***

1. Putem întâlni situația în care mișcările necesare pentru aducerea unei piese în poziția dorită strica alte fețe ale cubului.
2. Multe conflicte pot apărea și în timpul rezolvării unui nivel care duce la deformarea pe parcurs a unui nivel trecut.
3. La rezolvarea celui de-al doilea nivel unele piese puteau să se afle deja în poziția corectă, dar întoarse.

***2. Strategii de rezolvare conflicte***

1. Pentru a evita problemele apărute după stricarea unui layer vechi pentru a rezolva unul nou au fost introduse fapte suplimentare și necesare pentru alegerea mișcărilor viitoare.  
   Ex: (white-corners completed) (second-layer completed) (yellow-cross completed)
2. Pentru a evita realizarea unor mișcări care ar putea afecta rezolvarea piesei curente a fost adaugat un fapt suplimentar cu mișcările dorite iar dacă acel fapt încă ale elemente rămase, alte mișcări nu pot fi realizate. Mișcările sunt selectate din acest fapt de la stanga la dreapta.  
   Ex: (moves L U R L’ U’ R’)
3. Pentru toate regulile care realizeaza mișcările au fost adăugate și priorități mai mari pentru a le realiza înaintea verificării pentru mișcări noi.

# VII. Schema (logică) de funcționare

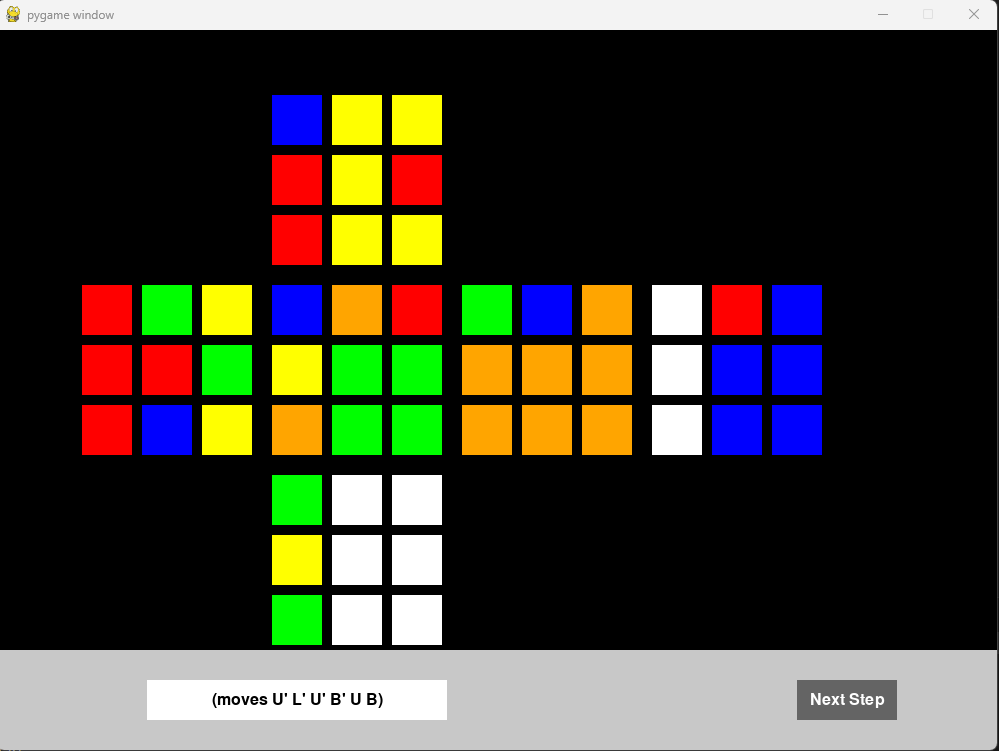


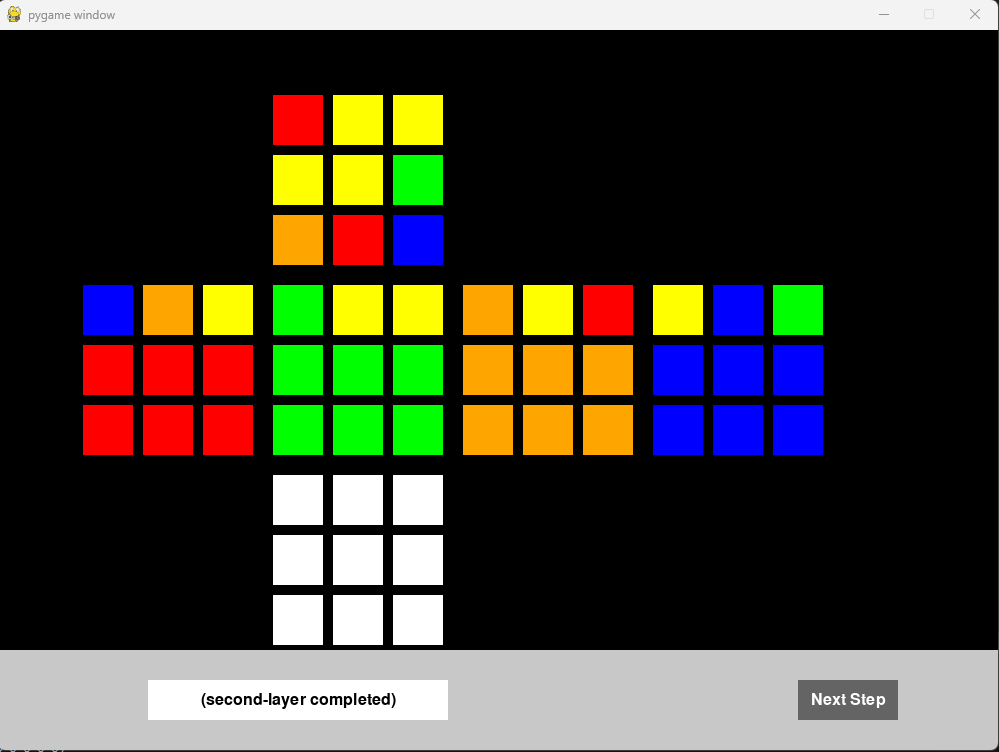
# VIII. Utilizare Clips în altă aplicație

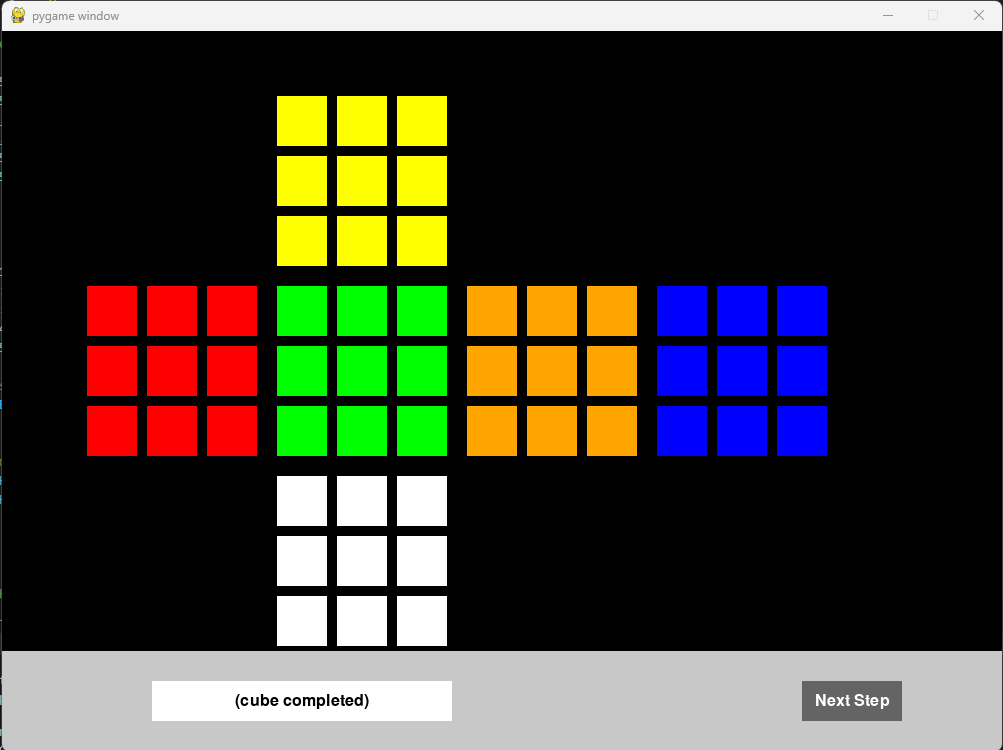
Proiectul deține o interfata realizata în mediul de dezvoltare Python implementată folosind librăria **pygame**. În fereastra aplicației sunt afișate cele 6 fețe ale cubului rubik si o bară de meniu care conține un buton ce verifică pașii necesari urmând să îi realizeze.

Legatura dintre aplicația python și sistemul expert este realizata folosind librăria clipspy. Aceasta pune la dispoziție o multitudine de metode ce ajuta la efectuarea funcțiilor din CLIPS cum ar fi generarea de fapte noi, creearea unor template-uri sau adaugarea unor reguli.

Help it perform clips functions.  
  
În aplicația prezentată sunt folosite metodele ***environment()*** și ***load()*** pentru a genera mediul clips și a popula baza de fapte și baza de reguli folosind un fișier extern. Mai departe este folosită funcția ***reset()*** pentru reinițializare. Pentru a executa câte o regulă este folosită funcția ***run(1)*** la fiecare apăsare a utilizatorului pe butonul Next Step din interfața prezentată.

**

**

******

# IX. Concluzii

Acest proiect are o mulțime de avantaje care îl pot face potrivit pentru rezolvarea problemei propuse. Printre avantajele oferite unul poate fi cel că procesul de rezolvare al cubului urmează algoritmul începătorului [3] care trece prin urmatoarele nivele: crucea alba, colturile albe, marginile fețelor, crucea galbenă, marginile galbene, colțurile galbene și orientarea lor. Un al doilea avantaj este cel oferit de interfață care informează utilizatorul, într-o cutie de text ușor de urmat, când fiecare din acele nivele este realizat și ce mișcări trebuie aplicate pentru a termina nivelul curent. Alt avantaj este și faptul că aplicația afișează statusul fiecărei fețe după realizarea oricărei mișcări astfel utilizatorul poate urmări cum evoluează cubul și poate compara propriul cub cu cel din interfață.

În ciuda avantajelor, se pot aduce și îmbunătățiri pentru a ușura procesul de rezolvare a cubului rubik. În starea actuală a proiectului nu pot fi realizate crucea albă și colțurile albe din lipsa de reguli ale sistemului expert. O primă îmbunătățirea ar fi adăugarea respectivelor reguli și optimizarea lor pentru a permite utilizatorului o experiență completă. Altă îmbunătățire bine-venită este implementarea unei opțiuni care permite utilizatorului să adauge un cub rubik sau stările actuale ale fețelor acestui cub direct din interfață iar rezolvarea să continue de la starea oferită.

În concluzie, proiectul “Dezvoltarea unui sistem decizional de reguli pentru rezolvarea unui cub Rubik” a constat în realizarea unei aplicații complexe care vine în ajutorul unui utilizator care dorește sa invete pașii necesari pentru rezolvarea fiecărui nivel al unui cub rubik. Aplicația contine o interfata generata în Python, ușor de utilizat, care face legatura dintre utilizator și sistemul expert realizat în CLIPS ce oferă o multitudine de avantaje însă poate fi îmbunătățită și adusă într-un stadiu mai performant.

**X. Bibliografie**

[1] Panescu D., Pascal C., Programare bazată pe reguli, Îndrumar de laborator, Editura Conspress, București, 2013, ISBN 978-973-100-258-3.  
[2] https://www.pygame.org/  
[3] <https://ruwix.com/>

[4] https://rubikscu.be/solver/