UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENŢĂ

Căpitan Sonar

implementarea pe calculator a unui board game

propusă de

Ileana Popa

Sesiunea: *iulie, 2018*

Coordonator științific

Lect. Dr. Cosmin Vârlan

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Căpitan Sonar

implementarea pe calculator a unui board game

Ileana Popa

Sesiunea: iulie, 2018

Coordonator științific

Lect. Dr. Cosmin Vârlan

Îndrumător Lucrare de Licență,

Lect. Dr. Cosmin Vârlan

Data Semnătura

DECLARAȚIE privind originalitatea conținutului lucrării de licență

Subsemnata POPA ILEANA, cu domiciliul în Iași, născută la data de 05.06.1996, identificată prin CNP 2960605226721, absolventă a Universității „Alexandru Ioan Cuza" din Iași, Facultatea de Informatică, specializarea Informatică, promoția 2015-2018, declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul: ***Căpitan Sonar – implementarea pe calculator a unui board game,*** elaborată sub îndrumarea dl. Asist. Drd. Cosmin Vârlan, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime. De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop. Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diploma sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată, ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, Semnătură student

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul ***Căpitan Sonar – implementarea pe calculator a unui board game***, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică. De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași,

Absolvent Ileana Popa

_____

Cuprins

Introducere

Captain Sonar este un board game lansat pe piață în 2016 de Roberto Fraga și Yohan Lemonnier și este considerat un upgrade complicat la jocul clasic "Battleships". Este jucat de două echipe care reprezintă echipajul câte unui submarin în încercarea de a cuceri adâncurile oceanului. Fiecare submarin are scopul de a-l vâna și a-l distruge pe celălalt. În fiecare echipă există patru roluri, comunicarea dintre ele fiind cheia pentru câștig.

Diferențele semnificative în experiența utilizatorului când joacă un board game sau un joc pe calculator pun în balanță anumite avantaje si dezavantaje. Jocul la masă alături de prieteni aduce experiența petrecerii timpului de calitate și comunicarea umană esențială care nu poate fi niciodată substituită de calculator. Dincolo de asta însă, spațiul virtual propune o varietate infinită de posibilități grafice, de sunet și interacțiuni. De exemplu, în implementarea mea, simularea unei explozii pe calculator este mult mai spectaculoasă și implică o succesiune de desene diferite afișate pe ecran la intervale diferite de timp. În privința graficelor resursele sunt nelimitate și nu este nevoie de materiale cum ar fi tablele de joc care se strică în timp sau foi si markere consumabile. Alte avantaje de necontestat ale unui joc pe calculator în defavoarea unui board game sunt disponibilitatea celor implicați, care se pot conecta la un server sau accesa un site prin internet din orice locație și oricând, fără pierderea sau deteriorarea componentelor.

Capitolul 1 prezintă setul complex de reguli ale jocului care se împart în patru roluri pentru fiecare echipă: Căpitanul, Operatorul Radio, Secundul și Inginerul. Rolurile sunt bine definite și strâns legate între ele, iar cu cât este comunicarea mai eficientă cu atât se obțin rezultate mai bune. În a doua parte a primului capitol am prezentat opiniile unor fani despre varianta board game mult apreciată, păreri printre care am găsit o propunere pentru implementarea unei variante pe calculator însoțită de nevoia rezolvării unor probleme care pot apărea în jocul de societate cum ar fi nerespectarea regulilor, de obicei din grabă sau neatenție.

În capitolul 2 am început cu o ilustrare a unei table complete de joc pentru un utilizator, care încorporează toate cele patru roluri. Pentru început, abordarea era crearea unei ferestre diferite pentru fiecare rol, ceea ce ar fi însemnat 8 ferestre pentru a ilustra un joc complet. Ulterior am schimbat dimensiunile ferestrelor deja implementate pentru încadrarea celor patru roluri într-o singură tablă, care are dimensiunile potrivite pentru a se observa ușor interacțiunea între doi

utilizatori pe același ecran. Elaborarea tuturor desenelor și elementelor grafice a fost o provocare din cauza numărului lor mare și a detaliilor numeroase, fiind nevoie de cele mai multe ori de schimbări la nivelul câtorva pixeli. Provocările întâlnite au presupus pe lângă desenarea lor, afișarea acestora la coordonate exacte și schimbarea lor în stările jocului, uneori cu elemente identice de altă culoare pentru a semnala încărcarea unui sistem sau pierderea unei vieți. De exemplu, după afișarea hărții este nevoie de afișarea unei matrice de puncte în care e posibil să se afle submarinul la un moment dat. Distanțele între puncte sunt calculate așa încât să respecte limitele hărții, iar un alt element grafic, cel care reprezintă calea urmată, este condiționat de această distanță, pentru a apărea în limite potrivite.

Dezvoltarea jocului a însemnat rularea lui într-o buclă infinită și afisarea pe ecran a tuturor elementelor potrivite la un anumit moment în funcție de evoluția variabilelor jocului, schimbate de trecerea timpului sau interacțiunea cu utilizatorul. Inițial au apărut ușoare dificultăți în detectarea erorilor și verificarea funcționalităților într-un script în care codul nu stătea niciodată pe loc. Astfel în capitolul 3 am pus accentul pe mediul de lucru și am descris succint bibliotecile implicate. Esențială pentru primirea comenzilor utilizatorului a fost în cele mai multe cazuri detectarea mouse-ului pe ecran. Printr-un apel de bibliotecă am obținut imediat valorile coordonatelor pe ecran, dar în unele cazuri acestea au trebuit prelucrate pentru a afișa interacțiunea așa cum trebuie. Exemplul principal este alegerea punctul de start care apare numai în puncte fixe, nu urmărește cursorul la toate coordonatele. Pentru a obține acest efect, am calculat punctul cel mai apropiat de cursor la un moment dat și am afișat pătratul roșu de start corespunzător, așa încât un punct este aprins oriunde am fi pe hartă și totodată acesta apare numai în locațiile valide.

Am dedicat capitolul 4 pentru a explica procesarea hărții pentru descoperirea automată a locațiilor posibile prin detectarea culorilor potrivite. Astfel, zonele de pământ sunt ocolite și orice hartă încărcată este procesată corect și gata pentru scufundarea submarinului și crearea unui nou traseu. Pas cu pas, abordările în implementare au devenit din ce în ce mai eficiente, pornind de la harcodare continuând cu îmbunătățiri precum schimbarea formatului de citire a culorilor din BGR în HSV, eliminarea zgomotului din imagine și extinderea zonelor scanate.

În capitolul 5 am descris comunicarea în rețea, care a fost foarte mult simplificată prin folosirea bibliotecii open-source PodSixNet, creată special pentru dezvoltarea jocurilor în python prin pygame așa încât un apel Send trimis de client către server sau invers are forma unui dicționar

python în care cheia “action” conține numele specific al acelei acțiuni, care declanșează execuția unei funcții din partea opusă care are același cuvânt cheie în nume. Un client nou este introdus într-o coadă de așteptare și este asociat cu următorul venit, după care coada se golește. Astfel, clienții sunt împerecheați doi câte doi și li se atașează un id al jocului pentru ca serverul sa poată răspunde corect tuturor cererilor.

1 Descrierea jocului

1.1 Captain Sonar Board game

Jocul de societate cu numele „Captain Sonar” s-a bucurat de o popularitate uriașă încă de la lansare, indiferent de vârstă. Precum cunoscutul “Battleship”, aventura și diferitele situații create în echipă cresc adrenalina și spiritual competitiv.

1.1.1 Roluri în echipă și organizare

Toți membrii echipei sunt așezați pe o parte a unei mese, iar fiecare își are rolul stabilit. Munca este divizată în funcție de câți jucători sunt în fiecare echipă. Numărul maxim de jucători este opt, câte patru pentru fiecare parte. Se poate juca în două moduri: pe rând sau timp-real. Modul rând după rând este mai liniștit, fiecare echipă așteaptă ca cealaltă să termine de făcut mutarea și abia apoi acționează, iar modul timp-real avantajează echipa cea mai rapidă, nu există pauze și presiunea și adrenalina cresc.

În ce privește componentele fizice ale board game-ului, acesta se bazează pe scrierea cu un marker negru pe câte o filă specializată pentru fiecare rol. Între cele două echipe există o barieră fizică din carton, așa încât comunicarea și colaborarea între membri este protejată.

Căpitanul are la dispoziție harta, ia cele mai importante decizii și este responsabil cu alegerea secretă a locului unde va scufunda submarinul. În timpul jocului, el trebuie să decidă direcția de înaintare și să anunțe detalii în legătură cu mutarea aleasă.

Operatorul Radio are acces la o versiune identică a hărții si o folie transparentă pe care marchează traseul urmat de submarinul inamic. Glisând folia pe hartă, poate să estimeze locația aproximativă a acestuia între insule, fiind ajutat de instrumentele încărcate de Secund.

Secundul este responsabil de șase sisteme pe care le încarcă la fiecare rând. O dată ce unul dintre sisteme este încărcat complet, acesta poate fi folosit împotriva inamicului (mine si torpile), pentru a estima mai bine poziția acestuia (drones, sonar), sau pentru a ascunde o parte din traseul făcut de propriul submarin (salience).

Inginerul este obligat să distrugă unul dintre sistemele submarinului și trebuie să aibă grijă că ele funcționează corect în cazul în care Secundul le-a încărcat complet și Căpitanul decide să le folosească.

1.1.2 Popularitate și îmbunătățiri

Este nevoie de opt jucători pentru ca jocul să fie cel mai interesant. Deși regulile par complicate, fiecare din cei patru trebuie să știe foarte bine care este rolul lui și cum să interacționeze cu ceilalți, iar jocul va oferi una dintre cele mai bune experiențe.

Premii si nominalizări:

- 2017 SXSW Tabletop Game of the Year Nominee
- 2017 Kennerspiel des Jahres Recommended
- 2016 Golden Geek Most Innovative Board Game Winner
- 2016 Golden Geek Most Innovative Board Game Nominee
- 2016 Golden Geek Board Game of the Year Nominee
- 2016 Golden Geek Best Thematic Board Game Nominee
- 2016 Golden Geek Best Party Game Nominee
- 2016 Cardboard Republic Socializer Laurel Winner
- 2016 Cardboard Republic Socializer Laurel Nominee

"În scris sună complicat, dar nu e greu de jucat, iar motivul principal este că fiecare este foarte focusat pe ce are de făcut [...] Dacă poți aduna 7 prieteni, sau să-ți faci noi prieteni cu ocazia asta, ca o echipă mare de oameni să ia parte împreună la această experiență este o operă de artă"[1]

Presiunea și intensitatea jocului i-au adus popularitatea, dar din aceleași motive nu este apreciat de persoane care-l consideră prea stresant sau prea gălăgios. Într-un joc real, coechipierii colaborează cu pasiune între ei, căpitanul trebuie să țipe ca să fie înțeles și adrenalina crește pentru că echipa cea mai rapidă care ia deciziile cele mai bune câștigă în final. Tocmai în situațiile în care se creează haos, un "ordin" înțeles greșit poate răsturna clasamentul.

"Una dintre problemele jocului este că este ușor ca o mișcare greșită făcută de una dintre echipe să o dezavantajeze pe cealaltă, de exemplu atunci când căpitanul anunță o mutare, dar înregistrează o alta pe hartă. Când echipa mea câștigă, am dubii inconfortabile că am fi putut câștiga dintr-o greșeală. Noi abordăm problema având doi oameni operând ca observatori care să verifice mutările făcute de o echipă, dar nici această rezolvare nu elimină în totalitate problema. Adăugarea de software jocului e posibil să ajute."[2]

[1] **Iain A.** http://www.thegamingreview.com/10779/2017/11/02/board-game-review-captain-sonar/.
[2] **Robert O'Callahan.** https://robert.ocallahan.org/2018/01/captain-sonar.html

2 Design-ul și grafica

În Fig. 1 observăm interfața completă a jocului pentru un utilizator, care încapsulează toate funcționalitățile. De la stânga sus, în sens invers acelor de ceasornic, un singur jucător are acces asupra rolurilor de Căpitan, Secund, Inginer și operator Radio. De fiecare dată când este rândul său, jucătorul trebuie să treacă pe rând prin fiecare mini-tablă de joc pentru o mutare completă.

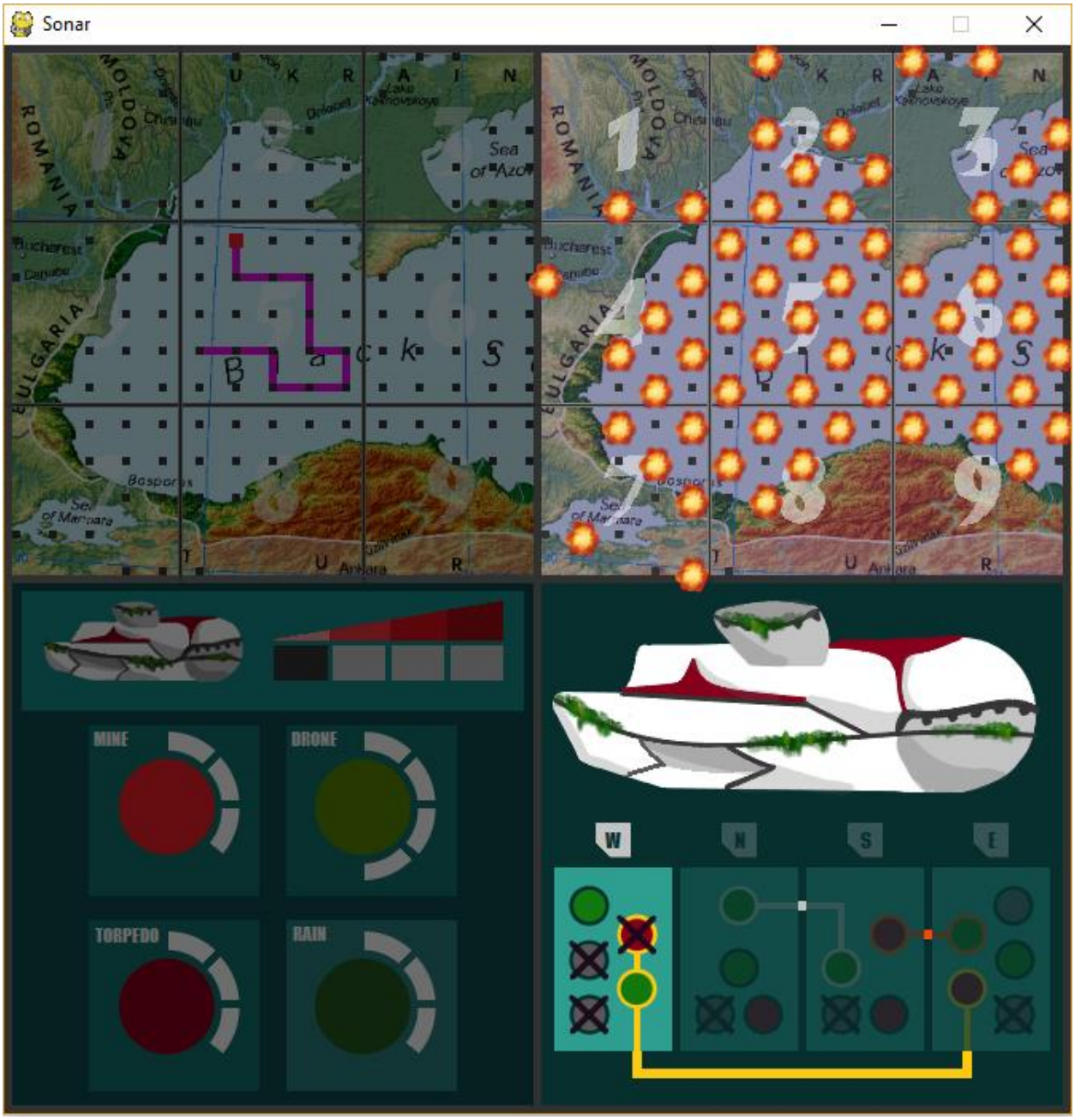

Fig. 1 Interfața completă a jocului

În folder-ul “graphics” am păstrat toate elementele grafice care apar în Fig. 1. Ele sunt încărcate la inițializarea clasei SonarGame și afișate pe ecran la coordonatele potrivite în fiecare buclă la momentele potrivite. Colțul din stânga sus a ecranului are coordonatele [0, 0]. Harta completă a jocului este obținută prin afișarea elementelor unele peste altele începând cu harta, apoi cu numerele și grid-ul care separă cele 9 secțiuni. Pentru afișarea locațiilor am folosi un singur pătrat de 5x5 pixeli care este desenat în două bucle for pe toată suprafața harții acolo unde locațiile sunt accesibile pentru submarin.

Pentru cadranul din stânga jos a fost nevoie de folosirea desenelor cu fundal transparent de doua culori, alb și gri pentru reprezentarea încărcării sistemelor. La fiecare afișare este verificată lista și sunt desenate pe ecran numărul de segmente gri care au fost alese în turele anterioare.

2.1 Interacțiunea cu jucătorul

Interacțiunea cu jucătorul se face prin hover cu mouse-ul sau prin apăsarea săgeților, iar elementele grafice joacă un rol important în asigurarea unei experiențe plăcute și intuitive a utilizatorului răspunzând la comenzile acestuia. Pentru alegerea unui loc de start în cadranul 1 (stânga sus) jucătorul face hover cu mouse-ul peste tabla de joc și este urmărit de punctul roșu de start, care nu este afișat în coordonatele exact ale mouse-ului, ci apare numai în locația cea mai apropiată. Mai apoi, când locația de start este decisă, este aleasă prin click.

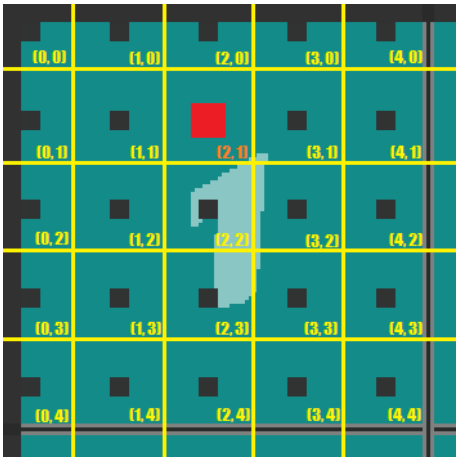

Fig. 2 Coordonatele locațiilor

În figura alăturată am reprezentat zonele corespunzătoare fiecărei locații. Este suficient să se facă click pe zona cea mai apropiată de punct și nu neapărat pe locația însăși, așa încât oriunde am da click pe hartă, va fi ales unul dintre puncte, semnalat înainte prin afișare la mișcarea mouse-ului pe ecran.

Pentru obținerea acestui efect sunt accesate coordonatele de la hover cu apelul pygame.mouse.get_pos(), după care sunt transformate în indici ai locațiilor.

O abordare similară a fost folosită pentr cadranul 3 la rolul Inginerului, unde la fiecare tură trebuie aleas un sistem care devine indisponibil. Acolo, deoarece locațiile detectate sunt puține și fixe am putut păstra coordonatele lor într-o listă, iar atunci când coordonatele mouse-ului sunt într-un interval stabilit, suficient de aproape de un anumit cerc, am afișat crucea care propune eliminarea ei. Astfel, când facem hover peste toate locațiile, crucea apare peste fiecare, iar când alegem una dintre ele cu click se consideră terminat rolul inginerului pentru tura respectivă, iar rândul merge la jucătorul oponent, timp în care îi putem urmări mișcările pe harta operatorului radio.

2.2 Localizarea și atacul oponentului

În ceea ce privește glisarea traseului pe ecran, o calitate pe care o oferă jocul pe calculator care este imposibil de obținut într-o variantă board game este decizia automată dacă traseul este viabil sau nu prin colorarea transparentă a fundalului cu roșu (Fig. 3), dacă trece peste insule sau iese din ecran și cu verde (Fig. 4) altfel. Totodată, în varianta board game este recomandată începerea desenării din mijlocul filtrului, pentru a evita ieșirea din limitele foii, problemă neîntâlnită aici.

Fiecare mutare a oponentului este trimisă prin server și înregistrată de codul fiecărui client. Celelalte trei roluri sunt activate succesiv la fiecare rând, dar acesta rămâne mereu disponibil pentru estimarea poziției submarinului inamic. Dacă s-au făcut multe mutări, iar terenul este suficient de specific devine posibil un atac mai precis care crește șansele de câștig.

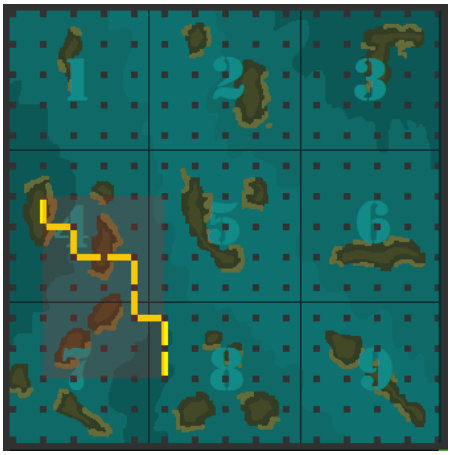

Fig. 3 Operatorul Radio - drum imposibil

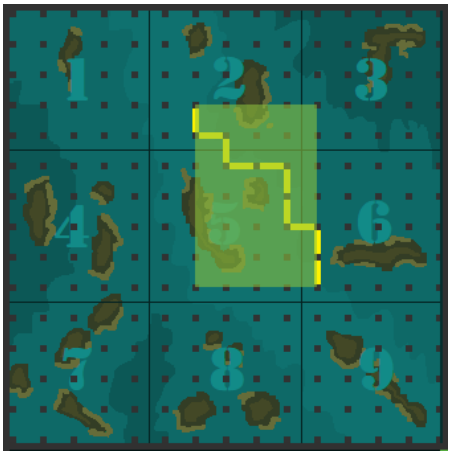

Fig. 4 Operatorul Radio - drum posibil

În implementare am calculat lățimea și înălțimea traseului și am afișat pe ecran o matrice de pătrate transparente de dimensiune p x p (p – distanța între două locații posibile). Astfel au trebuit păstrate în folder-ul cu imagini doar câte un pătrat din fiecare culoare pentru acoperirea oricăror dimensiuni posibile.

De fiecare dată când din panoul Secundului se activează un sistem are loc interacțiunea cu utilizatorul pentru lansarea armelor la locul dorit. În cazul lansării unei mine, aceasta poate fi montată în împrejurimea locației curente a submarinului, dar nu într-o zonă de teren sau pe traseu.

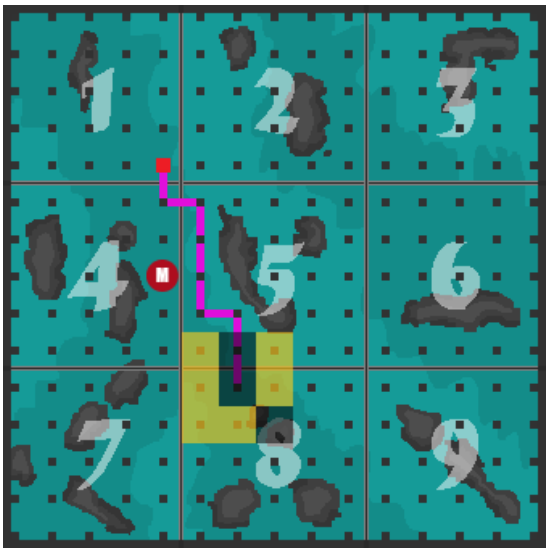

Fig. 5 Lansarea unei mine

Fiecare locație din matricea 3x3 afișată pe ecran poate avea una dintre valorile “on”, “off” sau “hover”, marcate pe hartă prin galben, gri și portocaliu. Coordonatele mouse-ului pe ecran sunt calculate la fiecare rulare și dacă sunt în intervalul potrivit, lista de pătrate își schimbă valorile, sunt afișate pe ecran, iar la click lista de locații este updatată cu valoarea “mine” în punctul ales, unde se afișează simbolul roșu.

Jucătorul care tocmai a plantat mina trimite un mesaj celuilalt pentru actualizarea listei lui de locații, dar pentru oponent acest schimb este invizibil, iar pe harta lui nu apare niciun semn care să-i indice pericolul, dar la intrarea în locația minată are loc explozia și pierderea vieților.

Minele pot fi activate și de jucătorul care le plantează, caz în care acestea sunt activate și au efect asupra tuturor submarinelor din raza lor de acțiune. O explozie declanșată distruge două vieți în centru și doar una la distanță unu în jur.

Pentru declanșarea unei explozii am folosit două imagini care apar alternative pe ecran. De fiecare dată sunt afișate trei imagini, una cu explozia din centru, apoi cu cea completă, apoi revenirea la prima imagine pentru ca efectul obținut să fie mai realist. Cele două intervale de timp între imagini sunt calculate prin numărarea unui număr de rulări ales așa încât rezultatul să fie plăcut din punct de vedere vizual.

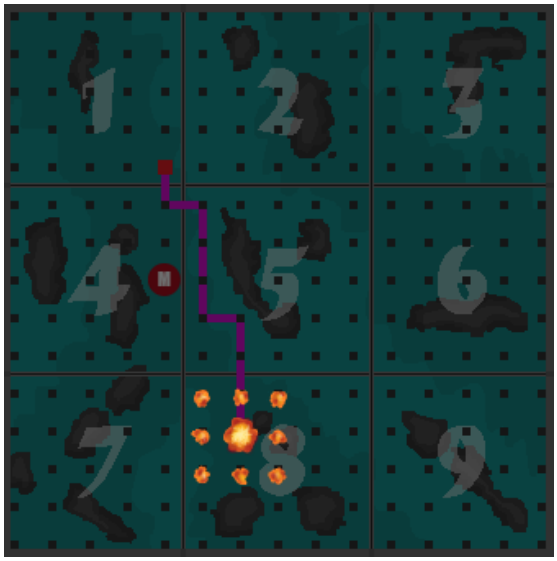

Fig. 6 Declanșarea unei explozii

O torpilă poate fi lansată la patru locații distanță sus, jos, stânga sau dreapta. În cazul minei am folosit hover pentru a alege poziția, dar aici am folosit săgețile. Odată aleasă o direcție de lansare, este imposibilă întoarcerea, iar încărcarea distanței se face prin apăsări repetate în direcția aleasă si apăsarea tastei Enter când poziția este decisă.

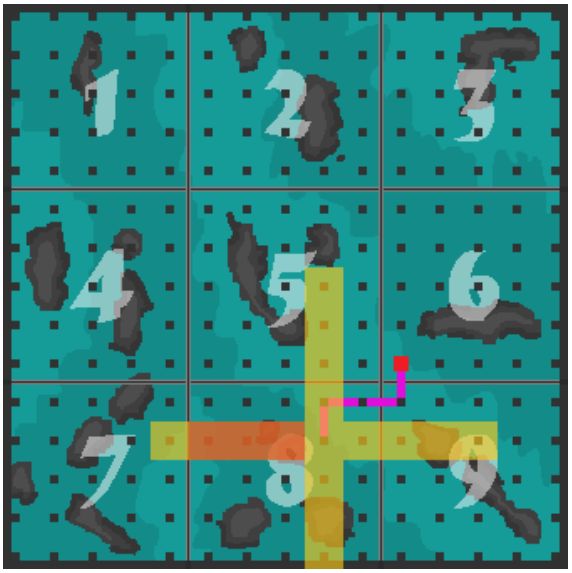

Fig. 7 Lansarea unei torpile

Crucea galbenă este formată din aceleași imagini folosite pentru alegerea minei, adică dintr-o serie de pătrate afișate numai dacă distanțele calculate nu sunt “out of bounds”. Pentru a nu ieși cu graficele din harta propriu-zisa în zone cu alte funcționalități am schimbat planul inițial în care foloseam un singur element galben pentru afișare.

Spre deosebire de mină unde declanșarea este amânată până când unul dintre jucători intră în locația respectivă, în cazul torpilei explozia este imediată.

Sistemul de atac nou care nu e prezent în varianta board game vizează toate locațiile în forma pătratelor de șah, alegându-se între cele două variante așa încât să se ocolească poziția curentă a submarinului care a lansat atacul. Dacă inamicul se află într-una dintre locațiile atacate, pierde o viață.

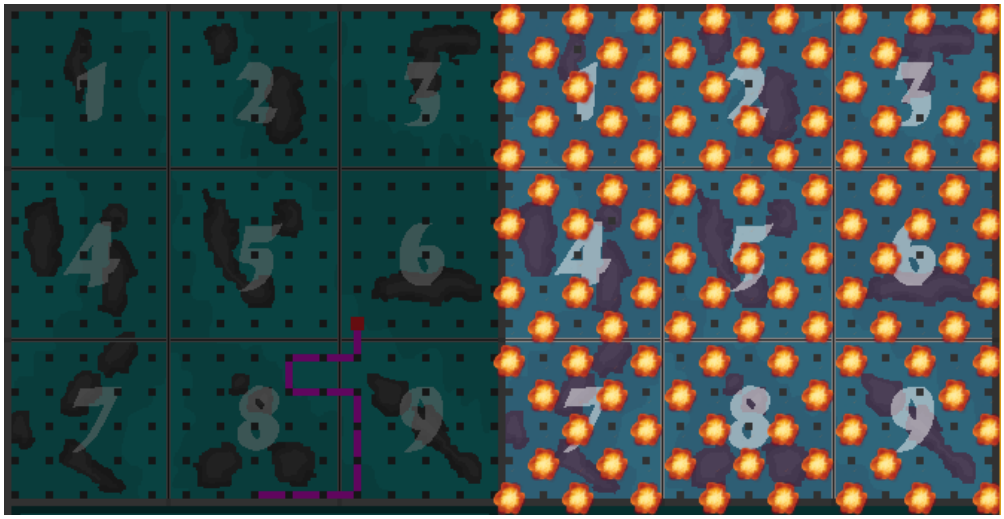

Fig. 8 Declanșarea sistemului de atac

3 Dezvoltare

Pygame oferă funcții care creează o interfață cu utilizatorul la rulare, permite detectarea coordonatelor mouse-ului pe ecran în timpul jocului și răspunde la comenzile utilizatorului (hover, click, buttons-pressed)

Jocul se bazează pe rularea într-o buclă infinită care se oprește din execuție când o stare este întreruptă (un utilizator renunță și închide fereastra de joc), sau când se ajunge într-o stare finală. O buclă reprezintă trecerea succesivă prin trei stări:

- Răspunderea la acțiunile utilizatorului (Handling events)
- Schimbarea variabilelor și stării jocului
- Desenarea noii stări pe ecran

Căpitanul are la dispoziție harta jocului pentru a desena cursul urmat de submarin. Harta jocului este cea din Fig. 10. Punctele albe de pe porțiunea cu apă reprezintă locațiile în care submarinul ar putea ajunge la un moment dat.

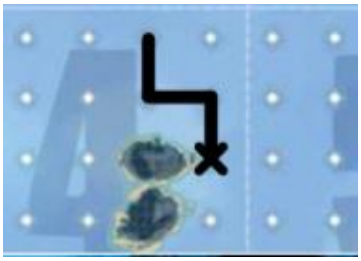

Fig. 9 Alegerea locului de start

La începerea jocului, căpitanul trebuie să marcheze cu X locul de scufundare al submarinului, iar apoi să dea ordine în funcție de direcția în care se îndreaptă submarinul echipei sale.
Cele patru direcții posibile sunt Nord, Sud, Est, Vest, iar ceilalți coechipieri și parcursul jocului depind de aceste decizii.
Ordinul este de forma: "Către Nord!", în varianta board game acesta fiind strigat tare așa încât atât coechipierii, cât și cei din echipa adverse să facă mutările potrivite.

În harta identică pentru jocul pe calculator din Fig. 11 alegerea locului de start se face prin hover peste harta căpitanului. Pătratul roșu apare în locațiile posibile și la click este fixat ca loc de start, de unde se începe mutarea submarinului prin apăsarea săgeților: UP/Nord, DOWN/Sud, RIGHT/Est, LEFT/Vest. Cursul submarinului nu poate trece prin insule sau să încalce propriul traseu. În varianta pentru calculator a fost eliminată posibilitatea încălcării regulilor de acest gen prin structurile de date verificate în spatele mutărilor, evitându-se erorile care s-ar fi putut produce din neatenție în varianta board game.

Fig. 10 Harta Căpitanului (board game)

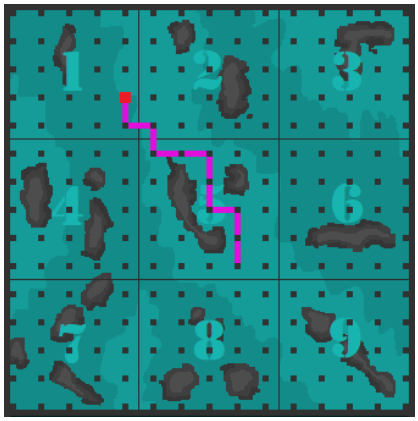

Fig. 11 Harta Căpitanului

Rolul operatorului radio este în strânsă legătură cu cel al căpitanului inamic. El are la dispoziție o harta identică pe care poate urmări traseul submarinului oponent pentru a-i estima cât mai bine poziția. Operatorul radio reprezintă "ochii și urechile echipei" și ascultă cu atenție ordinele date de căpitanul celeilalte echipe. Deasupra hărții jocului, el are un filtru transparent pe care desenează cu markerul succesiunea de ordine auzite de la oponent.

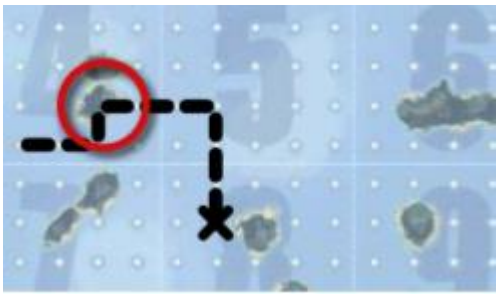

Fig. 12 Operatorul Radio - Glisarea traseului

Suprapunând filtrul transparent peste hartă, operatorul radio poate decide care sunt locațiile în care ar putea fi submarinul inamic, știind că este imposibilă trecerea peste insule. El poate începe desenarea traseului de oriunde, deoarece nu știe poziția de start a oponentului.

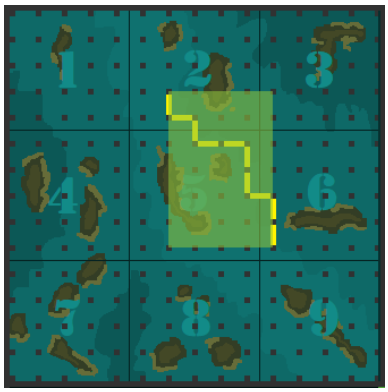

Fig. 13 Operatorul Radio - drum posibil

Deși similară ca scop, varianta pe calculator aduce o suită de avantaje jucătorului în ceea ce privește acuratețea traseului și urmărirea vizuală mai ușoară. Ajungerea la marginea foii transparente nu mai poate fi o problemă.

Greșelile din anunțarea sau receptarea traseului nu se mai pot strecura, pentru că lista mutărilor căpitanului oponent este transmisă și desenată automat pe harta operatorul radio, așa încât acuratețea traseului este garantată.

Secundul are rolul de a avansa sistemele submarinului de fiecare dată când căpitanul face o mutare prin umplerea unui spațiu alb din dreptul fiecăruia. Când un sistem este încărcat complet, secundul trebuie să anunțe echipa că este gata de utilizat. În partea de sus a paginii se marchează unul dintre spațiile albe de fiecare dată când echipa pierde o viață. După patru vieți pierdute, echipa adversă câștigă jocul. În Fig. 15 am păstrat armele cele mai folosite.

După încărcarea completă a unui sistem, acesta poate fi lansat. Minele si Torpilele colorate cu roșu din partea dreaptă sunt folosite pentru atacul direct al inamicului, iar cele verzi, drona si sonarul, pentru a estima poziția lui pe hartă.

Fig. 14 Secundul (board game)

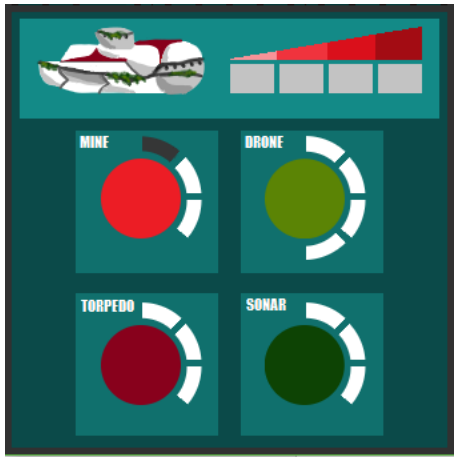

Fig. 15 Secundul

Inginerul este nevoit să distrugă unul dintre sisteme la fiecare mutare, în funcție de direcția de deplasare. De exemplu, în Fig. 17, inginerul poate alege doar un sistem din secțiunea Sud, pentru că acesta a fost cursul ales de căpitan în tura respectivă. Un sistem care este distrus la un moment trebuie reparat pentru a fi activat.

Colaborarea între roluri este esențială pentru minimizarea pierderilor. Dacă se dorește activarea imediată a unei mine, secundul trebuie sa-și folosească mutările doar pentru a încărca mina, inginerul trebuie să evite pe cât posibil distrugerea sistemelor roșii care ar dezactiva minele si torpilele, dar trebuie susținut de căpitan, care se asigură ca următorul traseu ales nu pierde una

dintre vieți și permite distrugerea temporară a altor locații. Inginerul poate informa căpitanul când anumite trasee pot fi periculoase pentru submarin.

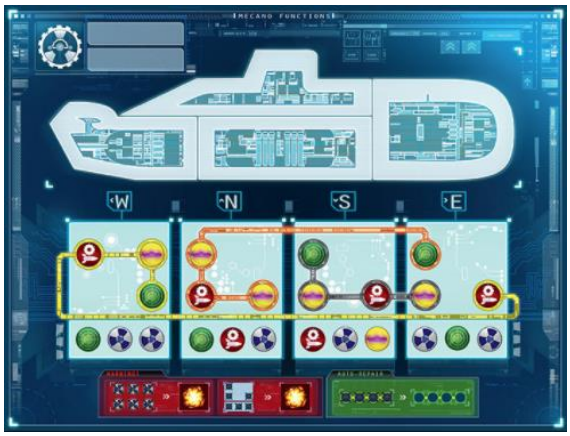

Fig. 16 Inginerul (board game)

Fig. 17 Inginerul

Repararea se face automat fără pierderi dacă toate sistemele legate între ele sunt distruse (gri, galben, portocaliu), dar dacă toate simbolurile albastre sau toate cele dintr-o secțiune V/N/S/E au fost distruse, se pierde o viață.

Se pot evita pierderile atunci când căpitanul ia în considerare posibilele distrugeri atunci când decide următoarea mutare.

4 Recunoașterea hărții

Jocul încarcă imaginile necesare dintr-un folder adițional pentru a le afișa în forma dorită în fiecare iterație în funcție de modificarea parametrilor rezultată din interacțiunea cu jucătorii. Dintre aceste elemente grafice, în Fig. 18 putem vedea harta inițială, identică cu una dintre hărțile principale ale board game-ului.

4.1 Abordări inițiale

Punctele prezente în Fig. 19 reprezintă locațiile valide în care s-ar putea găsi submarinul la un moment dat. Ele sunt păstrate sub forma de listă cu una din următoarele valori: “water”, “land”, “mine” sau “hidden_mine”.

Pentru harta clasică din Fig. 18 coordonatele locațiilor “land” au fost inițial hardcodate. Pentru încărcarea mai multor hărți această abordare este greu de pus în aplicare din mai multe motive, precum dificultatea în implementare prin efortul mare de a identifica manual toate locațiile și păstrarea seturilor de informații asociate fiecărei imagini.

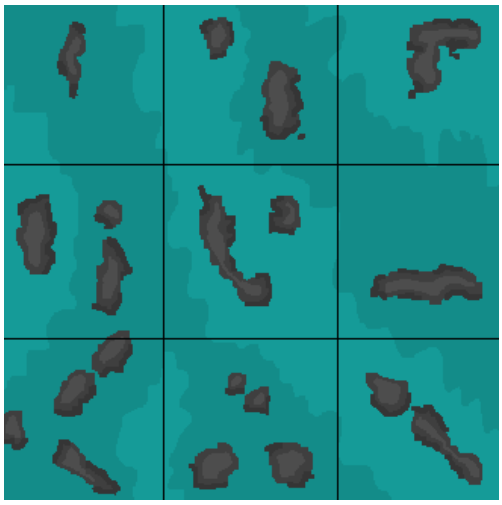

Fig. 18 Harta încărcată în format jpg

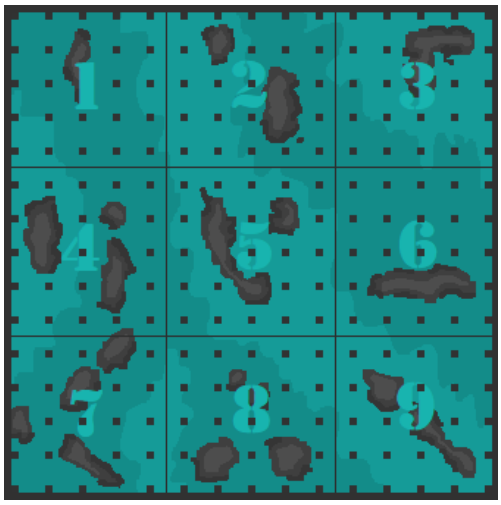

Fig. 19 Harta completă

De la imaginea inițială din Fig. 18 se ajunge la harta jocului din Fig. 19 prin identificarea coordonatelor punctelor de pe hartă. Având acces la fiecare punct este posibilă identificarea culorilor de pe hartă pentru a decide dacă submarinul se poate scufunda în acel loc, așa încât

punctele sunt adăugate în memorie și pe ecran doar dacă a fost recunoscută culoarea albastră, deci dacă locația curentă este apă.

4.2 Scanarea hărții

Cu biblioteca openCV pentru python am putut obține culoarea unui pixel de la coordonatele (x, y) în format BGR[3]. În cazul din Fig. 18 diferențele erau foarte clare între nuanțele de albastru si cele de gri, valorile pentru roșu, verde și albastru fiind considerabil mai mari pentru albastru, deci în acest caz a fost suficientă stabilirea unui prag pentru valorile BGR sub care pixelii erau considerați gri și peste care erau considerați albaștri, ceea ce e posibil de folosit pentru o gamă limitată de culori în care diferențele se observă ușor din acest format. O nuanță tipică de gri folosită în harta din Fig. 18 are valorile [77, 77, 77], iar albastrul pentru apă [152, 155, 21], deci pentru ca algoritmul să producă rezultatul corect pentru harta cu culori limitate a fost suficient să consider că o locație este pământ dacă are toate valorile pentru albastru, verde și roșu mai mici ca 100. Mai târziu am schimbat abordarea cu una mai potrivită prin citirea culorilor în alt format.

Pentru un rezultat corect este necesară verificarea unei zone mai mari de un pixel pentru a decide dacă împrejurimile sunt într-adevăr o zonă cu apă. De exemplu, în Fig. 20 punctul de interes este localizat între cele două insule înconjurate cu roșu, caz în care decizia trebuie să se bazeze pe o zonă mai mare care împrejmuiește locația.

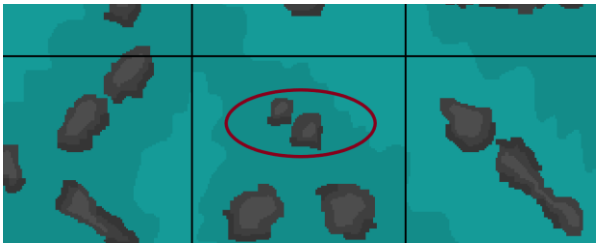

Fig. 20 Insulă nedetectată

Am decis că suprafața verificată trebuie extinsă ca în Fig. 2 (zona delimitată cu galben) pentru a considera toată partea din hartă reprezentativă pentru un punct fix, astfel încât toate porțiunile din hartă sunt implicate în luarea unei decizii.

În continuare am vrut să ofer posibilitatea detectării hărților reale cu același algoritm, dar formatul BGR nu a mai fost potrivit pentru detectarea unei game mult mai largi de nuanțe. Setarea unui prag pentru roșu, verde și albastru devine imposibilă când valorile implicate sunt variate și

[3] BGR – Similar modelului RGB, cu diferența că roșul ocupă byte-ul cel mai puțin semnificativ
Blue [0-255], Green [0-255], Red [0-255]

imprevizibile. În schimb, am folosit modelul de culoare HSV[4], care în schimbul descompunerii în roșu, galben și albastru folosește doar valoarea H pentru stabilirea culorii dominante percepute de observator (exprimată de la 0 la 360 de grade), diferențiind culorile apropiate prin saturație, care decide cât de luminoasă sau întunecată e o culoare prin cantitatea de alb adăugată și intensitate, care apropie culoarea de negru când are o valoare mică și de culoarea însăși la valori mari. (valori de la 0 la 100)

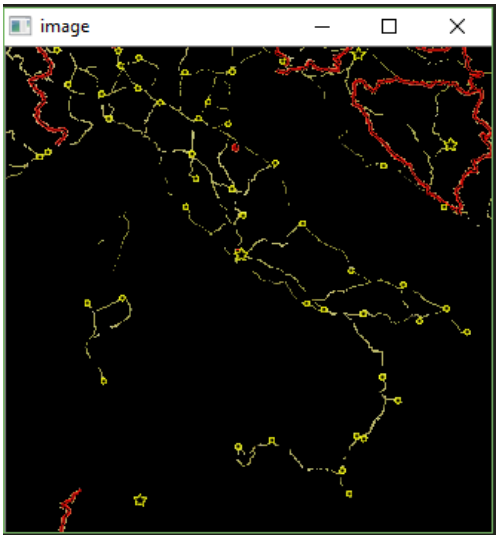

Fig. 21 Extragerea culorilor

Schimbarea formatului folosit pentru citirea imaginii a permis extragerea oricăror culori din hartă prin obținerea unei măști care creează o nouă imagine care culorile selectate și le înlocuiește pe celelalte cu negru.

În Fig. 21 de alături am combinat două măști, una pentru extragerea galbenului și una pentru roșu, deci vom avea de comparat culorile extrase cu negru în imaginea rezultat, în locul unor valori mai apropiate.

Apelul funcției primește imaginea originală și două valori care reprezintă limitele unui interval de culoare (ex. lower_blue = np.array([110,50,50]) și upper_blue = np.array([130,255,255])).

```
# AQUA/TEAL
lower_blue = np.array([70,50,50])
upper_blue = np.array([110,255,255])
mask_aqua_teal = cv2.inRange(img_hsv, lower_blue, upper_blue)

# BLUE/NAVY
lower_blue = np.array([110,50,50])
upper_blue = np.array([130,255,255])
mask_blue_navy = cv2.inRange(img_hsv, lower_blue, upper_blue)

mask = mask_aqua_teal + mask_blue_navy
```

Fig. 22 Detectarea tuturor nuanțelor de albastru

Pentru extragerea tuturor nuanțelor de albastru este nevoie de două măști: una pentru valorile de Bleu-Turcoaz (Aqua-Teal) și alta pentru valorile de Albastru-Bleumarin (Blue-Navy) ca în Fig. 22

[4] HSV – Hue [0, 179], Saturation [0, 255], Value [0, 255]

Fig. 23 Extragerea nuanțelor de albastru

Fig. 24 Eliminarea zgomotului din imagine

Rezultatul extragerii tuturor nuanțelor de albastru este în Fig. 23 în care se observă ocolirea diverselor titluri care pot apărea pe hărți în mijlocul apelor sau pe marginea porțiunilor de teren, fapt ce-ar putea îngreuna detectarea locațiilor în acele zone. De asemenea, lacurile și râurile nu au fost ocolite.

Pentru a ameliora problema descrisă mai sus, am putut elimina zgomotul din imagine, întrucât râurile și lacurile mici au fost eliminate complet, iar titlurile marilor si oceanelor au devenit suficient de difuze încât se confundă cu zonele de apă, așa cum vedem în Fig. 24.

Fig. 25 Recunoașterea locațiilor

Figura alăturată reprezintă harta așa cum apare în joc după rularea scriptului image_search în varianta îmbunătățită descrisă mai sus.
Zona marcată cu roșu este cea scanată pentru a decide dacă punctul din centru corespunzător ei este o locație care se află în apă sau pe uscat, iar decizia se ia prin calcularea numărului de pixeli albaștri sau negri din masca creată după citirea imaginii în format HSV.

Distanța între două puncte și dimensiunea hărții sunt singurii parametri folosiți în calcularea coordonatelor x si y ale centrului unui punct pe hartă. Pentru a delimita zona care urmează a fi scanată am calculat coordonatele colțului din stânga sus și ale celui din dreapta jos a zonei roșii din figură la fiecare pas pentru fiecare punct în parte.

Pentru punctele de pe margine și din colțuri suprafața pentru care se ia decizia se modifică așa cum observăm în colțul din dreapta din Fig. 25. În cazul lor am schimbat coordonatele colțurilor și aria suprafeței scanate.

După procesarea unei zone de centru, margine sau colț am putut calcula procentul de pixeli care reprezintă suprafața de pământ (cei negri din mască), având la dispoziție aria suprafeței și numărul de pixeli negri. Dacă procentul calculat este mai mare de 50% locația este marcată cu “land” și va fi accesibilă pentru submarin.

5 Comunicarea în rețea

Comunicarea în rețea a fost implementată prin folosirea bibliotecii open-source PodSixNet, creată special pentru dezvoltarea jocurilor în python prin pygame așa încât un apel Send trimis de client către server sau invers are forma unui dicționar python în care cheia “action” conține numele specific al acelei acțiuni, care declanșează execuția unei funcții din partea opusă care are același cuvânt cheie în nume. Un client nou este introdus într-o coadă de așteptare și este asociat cu următorul venit, după care coada se golește. Astfel, clienții sunt împerecheați doi câte doi și li se atașează un id al jocului pentru ca serverul sa poată răspunde corect tuturor cererilor.

```
print "planted mine at location: ", mine_location
self.Send({"action": "mine", "location": mine_location, "player": self.player, "gameID": self.gameID})
```

Fig. 26 Trimiterea mesajului din Client (Jucător)

```
def Network_mine(self, data):
    mine_location = data["location"]
    player = data["player"]
    gameID = data["gameID"]
    self._server.add_mine(mine_location, player, gameID)
```

Fig. 27 Receptarea mesajului la server

În Fig. 26 unul dintre jucători adaugă o mină la o anumită locație, care trebuie să fie trimisă și celuilalt jucător pentru a putea fi declanșată dacă ajunge în locația respectivă. Orice astfel de apel este compus din trei categorii de chei: acțiunea, adică scopul cererii, un număr de date care se doresc a fi comunicate și ID-urile jucătorului și al jocului. În cazul de mai sus, în Fig. 27, în clasa ClientChannel este apelată funcția Network_mine la trimiterea mesajului, care extrage datele din dicționarul trimis de jucător și apelează funcția corespunzătoare din GameServer.

Pentru a apela o funcție Network_myaction, datele primite de la client trebuie să aibă “myaction” ca valoare pentru cheia “action”.

```
def add_mine(self, mine_location, player, gameID):
    g = self.games[gameID]
    for i in range(0, len(g.player_channels)):
        if not i == player:
            g.player_channels[i].Send({"action": "mine", "location": mine_location, "player": player})
```

Fig. 28 Funcția din server care trimite mesajul clientului

Funcția add_mine din Fig. 28 găsește jocul după gameID și iterează prin lista de jucători asociați pentru a le trimite mesajul. În acest caz, jucătorul diferit de cel care a trimis cererea la server va fi notificat de plantarea unei mine.

```
def Network_mine(self, data):
    x, y = data["location"]
    self.locations[x][y] = "hidden_mine"
```

Fig. 29 Primirea răspunsului de la server

La rândul lui, clientul care trebuie să fie notificat primește un dicționar similar care are unul dintre câmpuri “action”: “mine”, deci apelează funcția Network_mine, din care poate extrage datele referitoare la locație, și apoi o poate marca cu “hidden_mine”.Acum mina va exploda dacă acesta va intra în locația respectivă.

Concluzii

Implementarea unui joc atât de complex precum “Captain Sonar” a presupus dezvoltare software în direcții diverse precum: adaptarea și punerea în aplicare a unui design, procesarea de imagine, comunicarea în rețea, implementarea logică a funcționalităților jocului și crearea unei experiențe plăcute pentru utilizator.

Design-ul în forma finală a fost conceput după mai multe abordări. Pentru început a fost implementată tabla de joc a Căpitanului cu funcționalitatea deplasării pe ecran și ocolirii insulelor. Operatorul Radio a fost creat într-o fereastră diferită cu aceeași imagine a hărții pe fundal pentru a putea înregistra mutările celuilalt jucător. Ulterior am îmbinat toate funcționalitățile unei echipe într-un singur utilizator complet care are acces la toate rolurile printr-un ecran compact și le poate gestiona alternativ.

Pocesarea hărții a fost îmbunătățită pas cu pas până la adăugarea automată a locațiilor într-o hartă reală. Inițial punctele au fost hard-codate, apoi culorile de la coordonatele lor au fost recunoscute cu biblioteca openCV. Trecerea de la o hartă a jocului cu culori limitate la una reală a fost posibilă prin schimbarea formatului în care este citită imaginea și crearea unei măști care extrage toate nuanțele de albastru. Rezultatele au fost îmbunătățite prin scoaterea zgomotului din imagine și extinderea zonei verificate. Pentru fiecare locație a fost calculat procentul de pixeli negri ai măștii din totalul de pixeli scanați, iar dacă acesta este mai mare de 50%, locația în cauză este considerată pământ și nu îi este adăugat un punct pentru ca submarinul să nu aibă acces acolo.

Comunicarea în rețea s-a realizat folosind biblioteca open-source PodSixNet creată pentru dezvoltarea jocurilor în python. Trimiterea unui mesaj este un apel Send trimis de client către server sau invers are forma unui dicționar python în care cheia “action” conține numele specific al acelei acțiuni, care declanșează execuția unei funcții din partea opusă care are același cuvânt cheie în nume.

Implementarea funcționalităților jocului a presupus detectarea coordonatelor pe ecran ale utilizatorului și declanșarea comenzilor în funcție de variabilele actualizate la fiecare stare. Elementele grafice implicate au de multe ori părți transparente sau cu opacitate scăzută pentru a putea fi suprapuse unele peste altele. Prin încărcarea unor desene similare de culori diferite am putut reprezenta completarea câmpurilor. Plantarea minelor, declanșarea torpilelor și a armelor conține animații ale exploziilor: imagini alternative afișate pe ecran pentru unități diferite de timp.

O abordare folosită des în îmbinarea funcționalității cu grafica este refolosirea elementelor și afișarea lor la coordonate diferite. De exemplu, pentru determinarea unei locații valide este afișat un pătrat gri de 5X5 pixeli în toate punctele fixe. Pentru declanșarea unei torpile, crucea de alegere care apare pe ecran este formată din pătrate transparente galbene, portocalii sau gri, folosite si pentru plantarea minelor. Culorile au fost folosite pentru efecte vizuale intuitive, de exemplu o locație este marcată cu gri dacă reprezintă o zonă de teren și nu poate fi aleasă, iar în cazul căutării traseului pe hartă fundalul este marcat printr-un filtru transparent verde sau roșu dacă este valid sau nu.

În final, am transpus cu succes funcționalitățile principale ale jocului de societate pe calculator, prin crearea unui design nou, compact și ușor de folosit, am profitat de avantajele oferite de păstrarea consistentă a datelor pentru a garanta un mod de joc fairplay și l-am extins printr-o armă nouă și posibilitatea detecției unei hărți reale.

Bibliografie

1. **Sweigart, Al.** *Making Games with Python and PyGame.* 2012.

2. **PodSixNet lightweight network layer. [Interactiv] https://github.com/chr15m/PodSixNet.**

3. (fan al jocului Captain Sonar), Iain A. [Interactiv] http://www.thegamingreview.com/10779/2017/11/02/board-game-review-captain-sonar/.

4. (fan al jocului Captain Sonar), Robert O'Callahan. [Interactiv] https://robert.ocallahan.org/2018/01/captain-sonar.html.

5. PodSixNet mailing list. [Interactiv] https://groups.google.com/forum/#!forum/podsixnet.

6. Tabel cu tipuri de culori și date pentru detectarea lor. [Interactiv] http://i.stack.imgur.com/gCNJp.jpg.

7. Modelul de culoare HSV. [Interactiv] https://www.lifewire.com/what-is-hsv-in-design-1078068.