Phoenixia

Gorie Ioana

1207A

Povestea jocului: Jocul se petrece în tărâmul roșu, având-o în prim plan pe Prințesa Phoenixia, fiica Împăratului Foc, care este preschimbată într-o pasăre Phoenix în urma unui blestem al unei vrăjitoare. Vrăjitoarea este sora mai mică a prințesei, care mereu a simțit că trăiește în umbra acesteia, pentru că niciodată nu a reușit să se facă la fel de iubită de împărat, de slujitori și de popor. Ceea ce a făcut-o specială pe lângă sora ei a fost faptul că s-a născut cu puteri magice, dar pe care alege să le folosească în scopuri malefice, în încercarea de a scăpa de cea pe care o vede ca pe rivala ei, într-o competiție imaginată de ea spre câștigarea inimii întregii împărății. Ca orice blestem din basme, transformarea Phoenixiei în pasăre Phoenix vine la pachet cu o soluție de revocare a vrăjii.

Așadar, singura cale de a-și recăpăta forma umană este de a zbura spre Insula de Azur unde va găsi Cascada Magică, din care, dacă se va adăpa, blestemul vrăjitoarei se va anula și ea își va recăpăta forma umană. Problema este că drumul spre cascadă prin tărâmul roșu este unul anevoios, deoarece vrăjitoarea o va pune în dificultate pe Phoenixia, confruntând-o atât direct (atacând-o dintr-un elicopter cu flăcări), cât și indirect, presărând toată suprafața oceanului ce separă tărâmul roșu de Insula de Azur cu bucăți plutitoare de pământ arzător și alte elemente naturale fierbinți (pomi, brazi) printre care eroina noastră va trebui să se strecoare ca să nu se ardă și să piară.

Prezentare joc: Jocul prevede un singur jucător care trebuie să se ferească de obstacolele care apar (flăcările propulsate din elicopterul vrăjitoarei în cadrul primului nivel sau bucățile de pământ, pomii și brazii plutitori, pe parcursul nivelului al doilea) și să colecteze bănuți plasați în locuri periculoase.

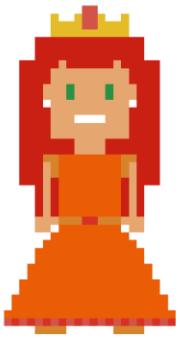
Reguli joc: Fiind întruchipată într-o pasăre Phoenix despre care se spune în mitologie că renaște din propria cenușă, eroina va putea reveni la viață chiar dacă nu reușește să se ferească de obstacole și piere, deoarece îi sunt alocate 3 vieți pentru fiecare nivel. Odată irosindu-le, jocul se va încheia cu totul, iar jucătorul va fi declarat pierzător, chiar dacă a reușit să

treacă cu brio de primul nivel. O viață se va pierde în cazul în care eroina va fi lovită de o flacără propulsată din elicopterul vrăjitoarei (nivelul 1) sau când nu va reuși să ocolească un element arzător de deasupra oceanului (nivelul 2), intrând în contact cu aceasta. Dacă pasărea Phoenix va ajunge la cascadă încadrându-se în viețile alocate pentru fiecare nivel, atunci ea își va recăpăta forma umană și jocul va fi câștigat. În plus, pe parcursul celor două niveluri va putea colecta diverși bănuți. La primul nivel, elicopterul vrăjitoarei are o defecțiune, propulsând, pe lângă flăcările menite a o răpune pe sora ei, bănuți de aur. La cel de-al doilea nivel, printre elementele cadrului natural vrăjite de către personajul negativ să îngreuneze drumul prințesei spre dezlegarea blestemului, se află și bănuți. Pe parcursul jocului, dacă jucătorul va obține o sumă de bănuți prestabilită de dezvoltatorul aplicației, îi va putea folosi să se întoarcă înapoi acasă.

În consecință, există două scenarii de câștig a jocului care au la bază faptul că prințesa reușește să ajungă la cascadă și să rupă blestemul preschimbându-se din pasăre în ființă umană, însă diferența o face suma de bani strânsă pe parcursul celor două niveluri, care stabilește dacă prințesa se va putea întoarce înapoi acasă la tatăl ei împăratul Foc, sau dacă va rămâne blocată pe Insula de Azur. Pentru primul caz jucătorul va fi declarat “Super Câștigător” iar pentru cel de-al doilea doar “Câștigător”.

Personajele jocului

- **Prințesa Phoenixia:** este protagonista și jucătorul-personaj. Ea este fiica Împăratlui Foc care a înmuiat inima tuturor oamenilor din împărăție, fiind o întruchipare a expresiei “frumoasă atât la interior cât și la exterior”. Pe parcursul jocului apare în două ipostaza: cea de ființă umană și cea de pasăre Phoenix, întruchipare obținută în urma blestemului vrăjitoarei.



- **Vrăjitoarea :** este personajul adversar, negativ. Ea este cealaltă fiică a Împăratului Foc care a simțit mereu că trăiește în umbra surorii ei mai mari, Phoenixia, pentru că nu a reușit niciodată să se facă la fel de iubită ca ea. Spre deosebire de sora ei, aceasta a fost înzestrată la naștere cu puteri magice pe care alege să le folosească în mod negativ asupra Phoenixiei, din dorința de a deveni ea prințesa cea mai iubită din împărăție. Pe parcursul jocului aceasta poate fi identificată atât ca în imaginea din stânga jos, cât și ca elicopterul mov (în care se presupune că se află) care o va ataca pe Phoenixia în nivelul 1.

- **Împăratul Foc :** este personajul episodic, care nu influențează în nici într-un mod desfășurarea evenimentelor. El este tatăl Phoenixiei și al vrăjitoarei, care este neputincios în fața forței supranaturale ale fiicei lui mai mici.

Tabla de joc

Structura tablei de joc:

Există două table diferite pentru fiecare nivel care au în comun afișarea celor 3 vieți în partea de stânga sus, chenarul de lângă ele care ține evidența sumei de bănuți câștigate pe parcursul jocului, butonul de părăsire a jocului din colțul din dreapta de sus, care îl va întoarce pe

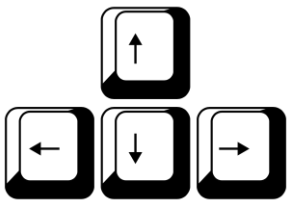
jucător înapoi la meniul principal și, bineînțeles, caracterul jucător, pasărea Phoenix. Ambele niveluri au la bază fundaluri realizate în Adobe Illustrator. Elementele din fundaluri nu intră în contact cu personajele și nu au nicio relevanță în desfășurarea jocului. Elementele active prezente pe ambele table de joc sunt bănuții, care odată colectați, valoarea lor va fi adăugată la comoara Phoenixiei.

Cele două nivele se deosebesc prin:

-**nivel 1** : prezența elicopterului, care nu interacționează în mod direct cu eroina jocului, dar elementele propulsate de către acesta (flăcările și bănuții) au un efect asupra desfășurării jocului în momentul în care intră în contact cu pasărea Phoenix : flacăra îi răpește una dintre vieți Phoenixiei iar bănuțul crește suma contorizată în chenarul din stânga sus. Așadar, nivelul constă în ferirea eroinei noastre de flăcările venite din elicopter și alternativ, colectarea bănuților care îi pot aduce un câștig mai mare jucătorului la final.

- **nivel 2**: prezența brazilor, pomilor și bucăților de pământ de desupra oceanului (elementele active), toate arzătoare, de care pasărea trebuie să se ferească, pentru că de fiecare dată când se va atinge de ele se va arde și își va pierde o viață. Capcana este că deasupra acestor elemente se pot afla bănuți , iar jucătorul trebuie să fie foarte vigilent în a-i colecta, fără a se apropia prea mult de obstacol. Deci, al doilea nivel prevede strecurarea personajului principal printre elementele plutitoare nocive pentru el, și din nou, ca și în cazul primului nivel, colectarea bănuților pentru un avantaj la finalul jocului în cazul atingerii sumei target.

Mecanica jocului



Jucătorul se poate deplasa în sus folosind tasta up arrow, în jos folosind tasta down arrow, la stânga folosind tasta left arrow și la dreapta folosind tasta right arrow.

Pe parcursul jocului propriu zis, utilizatorul va folosi doar arrow keys,

mouse-ul fiindu-i de folos doar dacă vrea să apese butonul EXIT din dreapta sus a ecranului. Tot de mouse va avea nevoie și în meniul principal, unde va trebui să dea click pe butonul aferent acțiunii pe care vrea să o desfășoare în cadrul programului.

Game Sprite

Descriere Meniu

Meniul îi va da opțiunea utilizatorului să înceapă jocul propriu-zis prin apăsarea butonului PLAY, să afle povestea Phoenixiei, instrucțiunile și regulile jocului printr-un click pe butonul ABOUT, și să părăsească aplicația/programul prin acționarea butonului QUIT. În plus, va avea posibilitatea să salveze progresul prin apăsarea butonului SAVE și să continue jocul de unde a rămas prin accesarea paginii cu înregistrări, făcând click pe butonul LOAD.

Descriere Niveluri

Descrierile celor două niveluri au fost realizate atât pe parcursul **Poveștii Jocului** și a **Regulilor de Joc** în care s-a menționat povestea fiecăruia în parte și esența lor, cât și în cadrul topicului **Tabla de joc**, unde s-au prezentat tablele de joc pentru cele două niveluri și s-a explicat rolul în desfășurarea jocului a fiecărui element regăsit în ele.

Descrierea scheletului proiectului

Diagrama de clase

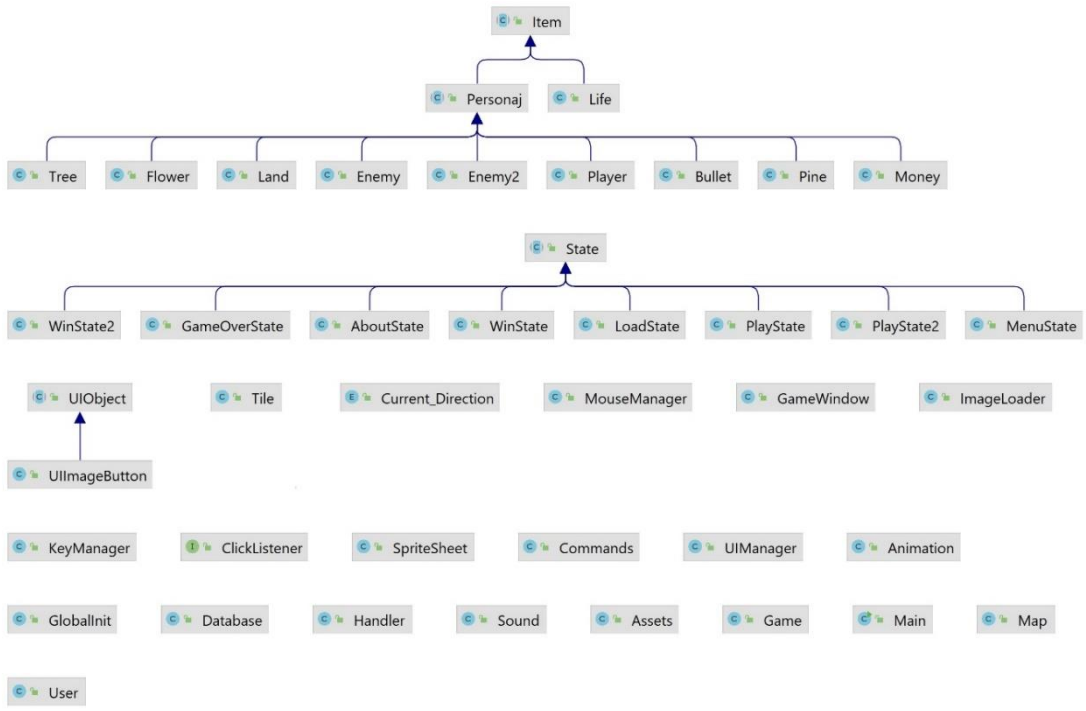
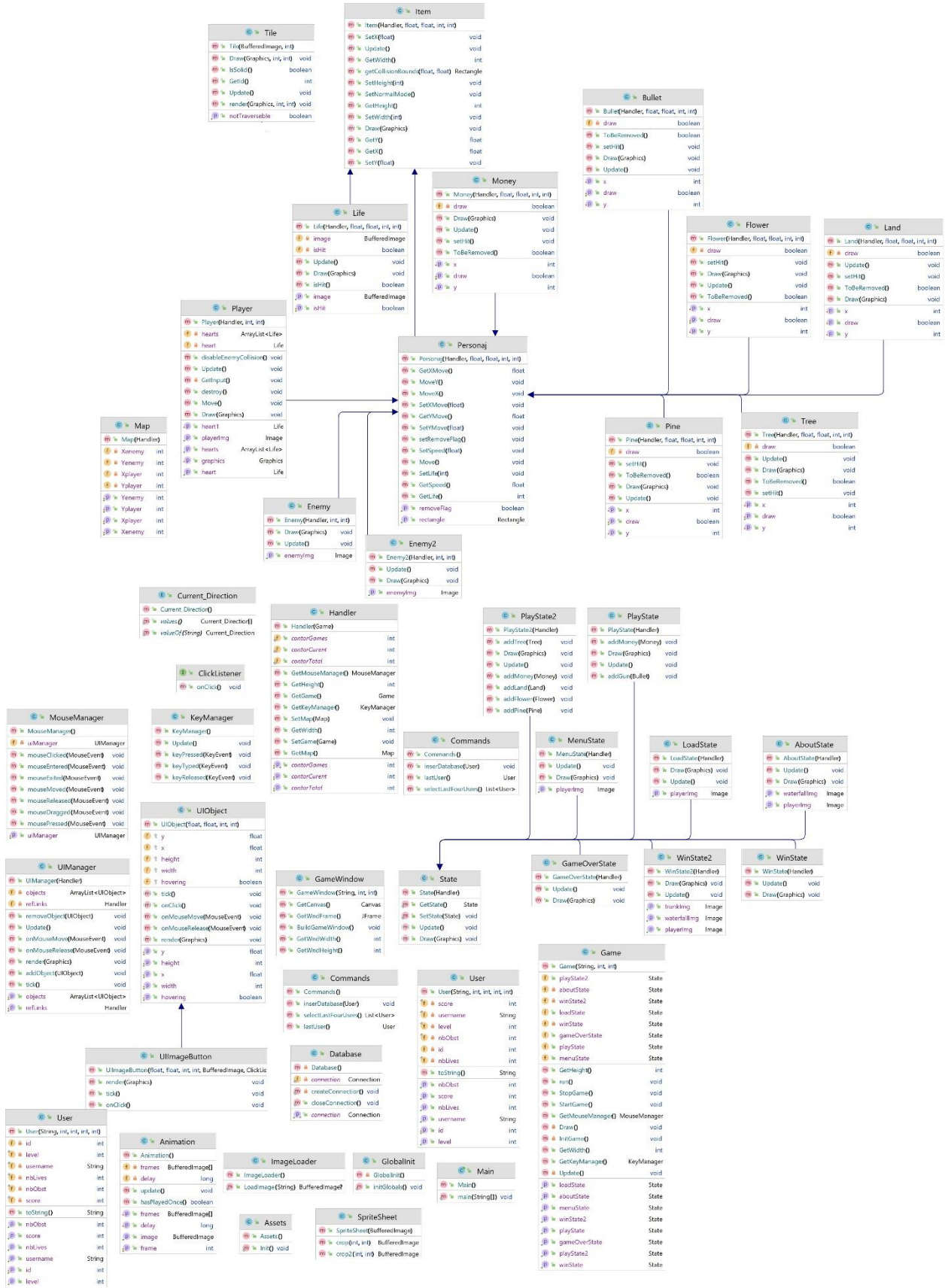


Diagrama proiectului



În cele ce urmează va fi prezentată o scurtă descriere a pachetelor din proiect cu o importanță majoră pentru funcționalitatea jocului, pe baza diagramei acestora.

Diagrama PaooGame

Acest pachet reunește 3 clase esențiale pentru lansarea în execuție a proiectului și stocarea unor “obiecte” utile în program (short-cuturi) cu celelalte pachete ale jocului.

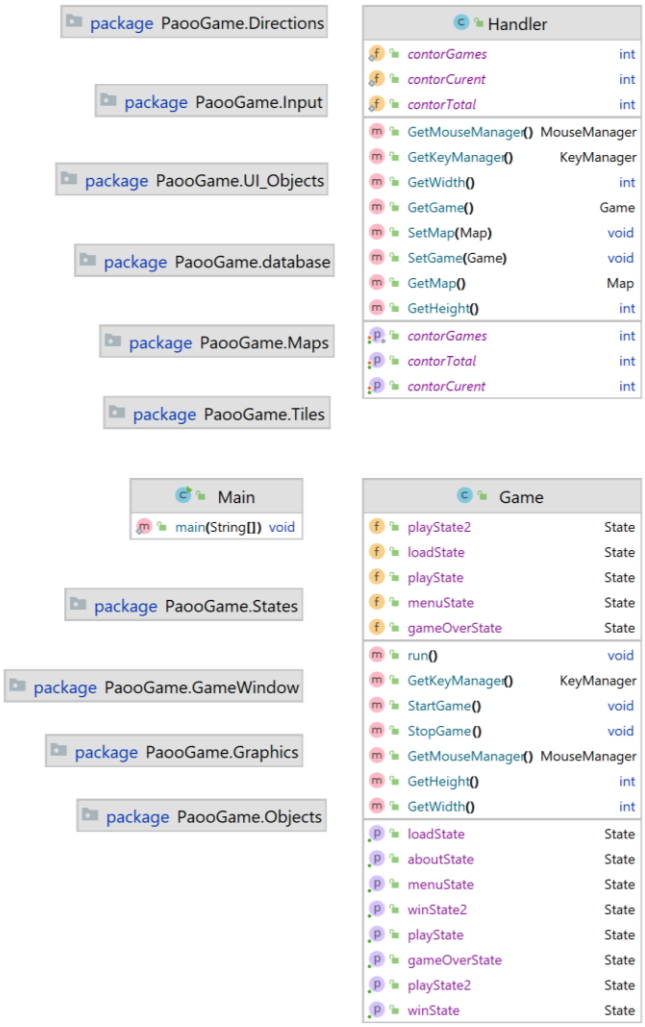


Diagrama Graphics

Reunește clasele care se ocupă de partea grafică a jocului, realizând asocierea obiectelor din joc cu anumite imagini decupate din spridesheet-uri și încărcarea pe ecran a unei imagini din memorie.

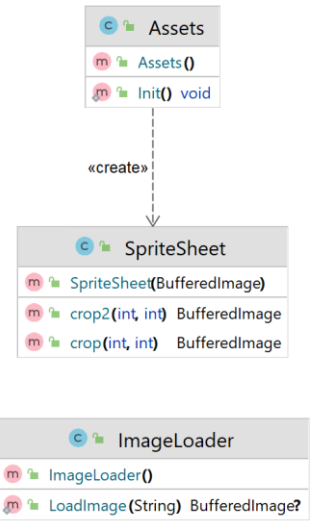


Diagrama UI_Objects

Pachetul UI_Objects realizeaza reuniunea claselor ce se ocupă de partea de Interfață Umană.

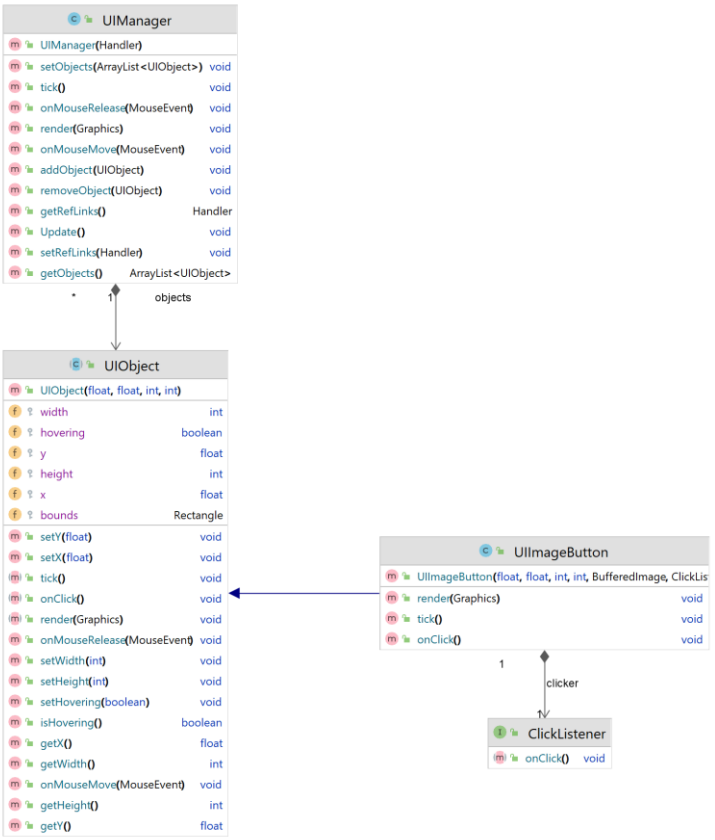
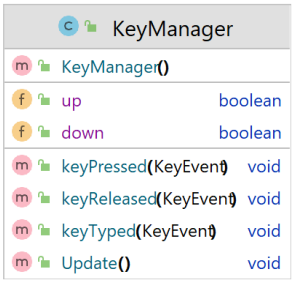


Diagrama Input

Conține cele două managere KeyManager și MouseManager care se ocupă de gestionarea input-urilor de la tastatură și de la mouse.



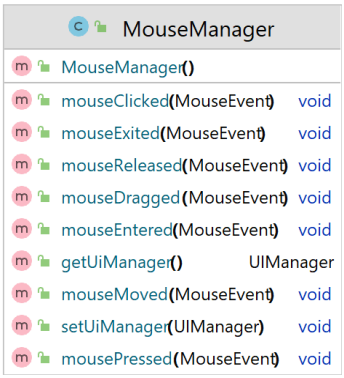


Diagrama Database

Reunește clasele ce se ocupă de gestionarea bazei de date și a elementelor ei, de conectarea acesteia cu proiectul și de inițializarea globală a membrilor pe care îi conține.

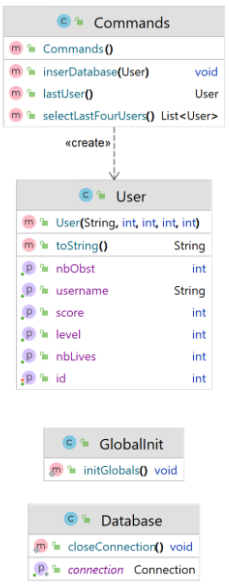


Diagrama Objects

Pachetul Objects reunește clasele ce implementează noțiunea abstractă de entitate activă din joc, element cu care se poate interacționa: personaje, obstacole etc. Motiv și pentru care există o clasă abstractă de bază , Item, pe care o vor extinde toate celelalte clase fie direct (Clasa Life) fie prin intermediul clasei abstracte Personaj, care îi adaugă unui obiect de tipul item niște caracteristici în plus precum viteza sau numărul de vieți.

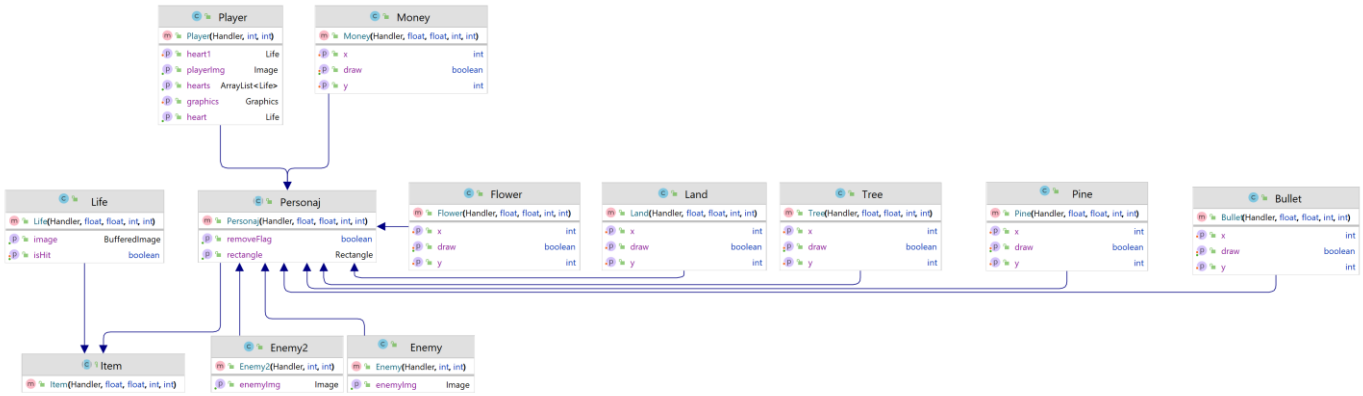
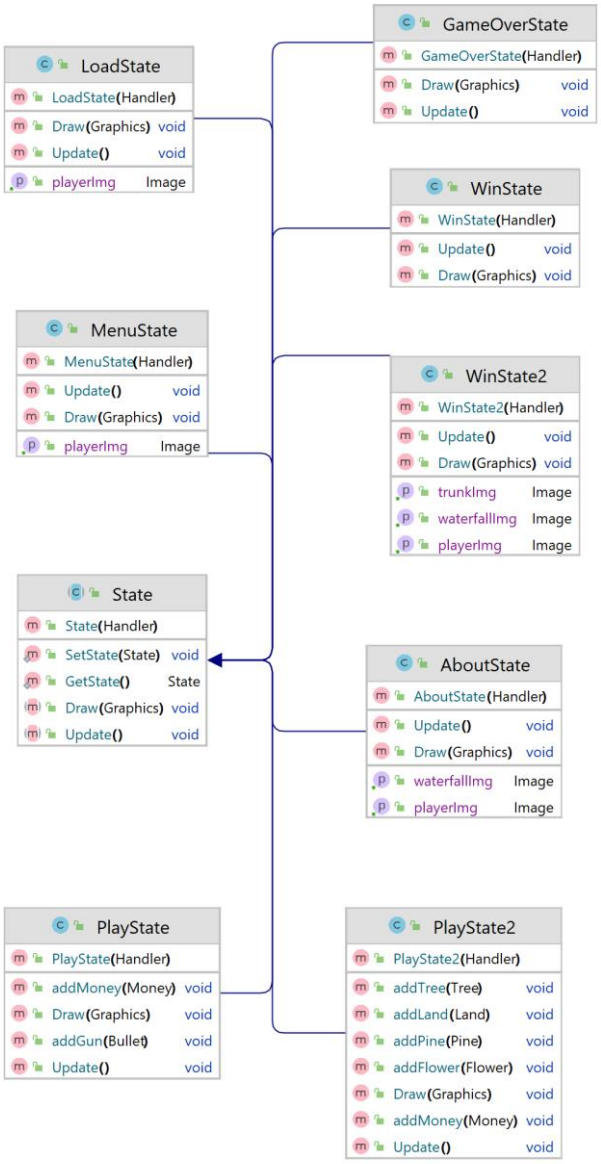


Diagrama States

Pachetul States are la bază clasa abstractă State ce implementează noțiunea abstractă de stare a jocului, din care derivă 8 stări ce se preocupă de meniul jocului (MenuState), pagina de informații (AboutState), pagina cu salvări a progreselor utilizatorilor (LoadState), nivelul 1 / nivelul 2 (PlayState / PlayState2) și nu în ultimul rând, cele 3 stări destinate verdictului/statusul jucătorului: pierzător (GameOverState), câștigătorul nivelului 1 (WinState) și al întregului joc, care implică, bineînțeles și terminarea cu succes a nivelului 2 (WinState2).



Descriere clase proiect

Main

După cum se poate observa din diagrama prezentată mai sus, această clasă creează o instanță de tipul Game, iar mai apoi pornește fluxul activităților prin apelul metodei StartGame() din Game.

Game

Rolul acestei clase este de a face inițializările (de exemplu de a crea instanțe pentru jucători/inamici, pentru a inițializa harta, etc.), dar și de a menține interfața grafică la zi cu cea ce se întâmplă în joc. Această clasă implementează interfața Runnable pentru a avea comportamentul unui fir de execuție (thread). În momentul apelului metodei StartGame() se instanțiază un obiect de tip Thread pe baza instanței curente a clasei Game. Orice obiect de tip Thread trebuie să implementeze metoda run() care este apelată atunci când firul de execuție este pornit (start()). Această metodă run() inițializează jocul prin crearea unei instanțe GameView, iar mai apoi controlează numărul de cadre pe secundă printr-o buclă while și "pregătește" noua scenă (Update()) pe care o va desena pe interfața grafică (Draw()). Metoda Update() actualizează starea jocului (de exemplu: modifica poziția jucătorilor pe baza tastelor apăsate, schimbă poziția inamicilor folosind chiar tehnici de inteligență artificială, crează diferite tile-uri (dale), etc. Dacă ne referim la un joc 2D care are harta pătrată, aceasta este împărțită în celule de aceeași dimensiune(ca o tablă de sha), iar aceste celule sunt tile-uri care pot avea texturi diferite, pot reprezenta diverse obiecte (de exemplu proiectilele unui tanc, acesta poate fi un tile ca și restul elementelor de pe hartă dar care are fundalul transparent). Metoda Draw() va desena pe interfața grafică modificările făcute de metoda Update(). Interfața grafică este un canvas, făcând o analogie cu realitatea, poate fi considerată o pânză pentru desen, pe care sunt desenate diverse obiecte.

Game Window

După cum discutam mai sus, metoda run() a clasei Game crează o instanță a clasei GameWindow. Această clasă este responsabilă cu fereastra în care vor fi desenate obiectele pe un canvas (se poate observa ca avem o metoda GetCanvas care întoarce canvasul). De asemenea, avem un obiect JFrame care permite desenarea de butoane, controale, textbox-uri, etc, dar poate conține și un canvas în care pot fi desenate diverse obiecte folosind texturi. Dacă de exemplu dorim să avem un meniu pentru joc, atunci sunt necesare mai multe obiecte JFrame, unul pentru meniu în care sunt desenate butoanele, etc și unul pentru acțiunea jocului,

iar în funcție de interacțiunea utilizatorului acestea vor fi schimbate între ele.

Assets

Conține câte un membru static de tip BufferedImage pentru fiecare tile/dală, chiar și pentru jucători/inamici. In acești membri sunt stocate imaginile, adică texturile acestora care vor fi desenate prin apelarea metodelor Draw() din fiecare clasa tile apelate din metoda Draw() din clasa Game. În acest exemplu este doar o imagine care conține toate texturile folosite, așadar pentru a putea instanța acest obiecte statice trebuie să decupăm fiecare textura din acea imagine. Pentru a face asta se folosește o instanță a clasei SpriteSheet în metoda Init(). Metoda folosită din SpriteSheet este crop(x, y).

Spritesheet

Constructorul acestei clase primește imaginea, iar clasa conține 2 membri constanți (final in Java) pentru înălțimea, respectiv lățimea texturilor (trebuie sa aibă toate aceleași dimensiuni). Metoda crop(x, y) primeste doi parametrii, x specifică pe ce coloană se află textura pe care dorim să o decupăm în imaginea cu toate texturile, iar y specifică linia. Pentru a afla efectiv poziția în imagine se determină pixelii de unde încep texturile de interes prin înmulțirea liniei/coloanei cu înălțimea, respectiv lățimea texturilor. Astfel se obține colțul din stânga sus al texturii, iar pentru a afla celelalte colțuri se folosesc înălțimea, respectiv lățimea.

Image Loader

Înainte de a extrage fiecare textură, imaginea cu toate texturile trebuie să fie citită din memorie, iar pentru asta se folosește clasa ImageLoader cu metoda statică LoadImage(path) care primește ca parametru calea către imagine în memoria calculatorului.

Handler

Clasă ce reține o serie de referințe/shortcuturi ale unor elemente pentru a fi la îndemâna dezvoltatorului aplicației, ce are ca scop evitarea îngreunării programului cu funcțiile respective prin seria lor numeroasă de parametrii. O metodă seminficativă de menționat din această clasă, este metoda GetGame() care întoarce referința către obiectul Game, sau metoda SetGame() care primește referința obiectului Game ca parametru și seteaza referința către un obiect Game.

KeyManager

Extinde clasa KeyListener și are ca scop gestiunea intrării (input-ului) de la tastatura.
Clasa verifică dacă a fost apasată o tasta, stabilitește care dintre ele a fost cea în cauză și setează corespunzator un flag. Flagul respectiv va lua valoarea true dacă tasta a fost apăsată, iar pentru cazul contrar, valoarea lui va fi false.
Membrii acestei clase vor fi în primul rând cele două flaguri destinate tastelor de interes din acest program (sus și jos) și nu în ultimul rând vectorul de flaguri pentru toate tastele. Pe lângă metoda Update() care se ocupă de actualizarea managerului de taste, două funcții de interes ale clasei sunt keyPressed(KeyEvent e), care este apelată atunci cand un un eveniment de tastă apasată este generat reținând in vectorul de flaguri acest lucru , și keyReleased(KeyEvent e) ce va fi apelată atunci când un un eveniment de tastă eliberată este generat, reținând în vectorul de flaguri că respectiva tastă nu mai este apăsată.

MouseManager

Extinde clasa MouseAdapter și are ca scop gestiunea intrării (input-ului) de la mouse.
Clasa verifică dacă a fost apasat un click și setează flagul aferent. Flagul respectiv va lua valoarea true dacă click-ul a fost apăsat, iar pentru cazul contrar, în care nu a fost apăsat, valoarea lui va fi false.
Membrii acestei clase sunt flagul pentru contorizarea apasarii clickului stang, flag pentru contorizarea apasarii clickului drept, locația pe axa X a mouse-ului, locația pe axa Y a mouse-ului, referința către managerul UIManager. Pe lângă metoda mouseMoved(MouseEvent e) care se ocupă cu memorarea locației efective a cursorului pe ecran, două funcții de interes ale clasei sunt mousePressed(MouseEvent e), ce va fi apelata atunci cand este apasat un click pe mouse, fără să contexe care dintre ele (drep/stâng) și mouseReleased(MouseEvent e) ce se apelează atunci când click-ul este eliberat, indiferent de care dintre ele. Aceste două funcții vor realiza setarea flagurilor aferente clik-ului stâng și/sau celui drept, asociindu-le valorea true dacă acestea au fost apăsate, sau false dacă au fost eliberate.

Item

Implementează noțiunea abstractă de entitatea activă din joc, element cu care se poate interacționa: obstacole, jucători etc.

Aceasta are ca membrii unele caracteristici ale imaginii entității precum poziția pe axa X de joc a, poziția pe axa Y de joc, lățimea, lungimea , dar și diferite ipostaze ale dreptunghiului de coliziune: dreptunghiul curent de coliziune, dreptunghiul de coliziune aferent starii obisnuite (spațiul ocupat de entitate în mod normal), dreptunghiul de coliziune aferent starii de atac. Referința către obiectul de tip Handler nu lipsește nici ea dintre membrii clasei Item, fiind un obiect de tip “shortcut” către colecția de referințe utile în program.
Dintre metodele clasei amintim funcția Update() a cărui rol este de a actaliza poziția item-ului, funcția Draw() ce randează/desenează item-ul în noua poziție, și o serie numerioasă de setters și getters destinate setării și returnării membrilor clasei Item (GetX(), SetX(float x), GetWidth(), SetWidth(int width) , etc.).

Personaj

Extinde clasa Item, implementând noțiunea abstractă de personaj/caracter din joc.
Atributele care fac obiectul de tip Personaj să fie mai mult decât un simplu Item, sunt numărul de vieți, viteza dar și un flag denumit *remove* ce va fi setat la valoarea true în momentul în care obiectul de tip Personaj trebuie șters. Pe lângă seria numeroasă de setters și getters destinate setării și returnării membrilor clasei Personaj (GetLife(), SetLife(int life), GetSpeed(), etc.), mai apare nou față de clasa Item metoda *Rectangle getCollisionBounds(float xOffset, float yOffset)* ce se ocupă cu returnarea unui nou obiect de tipul Rectangle obținut prin prelucrarea valorilor primite ca parametru.

Player

Extinde clasa Personaj și implementează noțiunea de player (personajul controlat de jucător).
Elementele suplimentare pe care le aduce față de clasa de baza sunt: imaginea de tip BufferedImage, obiectul referință către clasa Graphics care se ocupa cu grafica jocului, viața personajului, vectorul de inimi care reprezintă starea curentă a vieților personajului și flagul coliziunii cu alte elemente din joc. Clasa Personaj implementează metodele Update() , Draw(), Move() din clasa de bază , iar în plus, pe lângă funcțiile setters și getters aferente membrilor proprii clasei, se remarcă metoda destroy() care se ocupă cu preschimbarea imaginii caracterului într-una care simbolizează distrugerea acestuia (fie ea nulă/invizibilă sau o explozie).

Enemy și Enemy 2

Extind clasa Personaj și implementează noțiunea de inamic/adversar al player-ului pentru cele două niveluri ale jocului (Enemy – nivel 1, Enemy 2 – nivel 2).

Nu sunt niște clase bogate în metode, întrucât implementează doar noțiunea de Update() și Draw() din clasa de bază , având nouă doar funcția getEnemyImg() care se ocupă cu returnarea imaginii inamicului a cărei extensie este gif prin utilizarea funcției getDefaultToolkit() regăsită în librăria java.awt . În metoda Update() se realizează mișcarea automată a caracterului de sus în jos între niște limite spațiale setate de către dezvoltatorul aplicației, dar și propulsarea “gloanțelor” în cazul primului nivel, a obstacolelor în nivelul 2 și a bănuților pentru ambele ipostaze ale jocului.

Bullets, Flower, Tree, Land și Pine

Extind clasa Personaj și implementează noțiunea de gloanțe pentru primul nivel prin intermediul clasei *Bullets*, și de obstacole pentru nivelul 2 prin clasele *Flower, Tree, Land și Pine.*

Gloanțele și obstacolele din joc sunt considerate a fi Personaje, întrucât ele reprezintă o abstractizare a notiunii de inamic, deoarece coliziunea lor cu player-ul are un impact asupra acestuia. O metodă remarcabilă implementată de aceste clase se numește setHit() și se ocupă cu setarea flagului “lovit” care indică dacă un obiect de tipul celor menționate în titlu a avut o coliziune cu Player-ul.

Money

Obiectele de tip Money sunt tratate identic cu cele de tip Bullets, Flower, Tree, Land și Pine, doar că în momentul producerii unei coliziuni între o instanță a clasei Money și Player, efectul lor va fi considerat pozitiv, incrementându-se contorul care se ocupă cu memorarea numărului de bănuți colectați de jucător pe parcursul nivelului.

Life

Life are ca și clasă de bază Item și se ocupă cu implementarea noțiunii abstracte de viață a jucătorului. Este singura clasă care instanțiază obiecte de tip Item care nu sunt și Personaje, deoarece acestea nu au contact direct cu playerul, ci sunt doar niște obiecte statice care apare în colțul din

stânga sus a ecranului în permanență, informându-l pe utilizator cu privire la numărul de vieți (prevăzute sub forma unor inimoare) care i-au rămas pentru nivelul curent. În funcție de un flag denumit isHit care se va seta cu valoarea true in momentul in care player-ul atinge un glont sau un obstacol, programul va “crăpa” câte una dintre viețile acestuia.

State

Implementeaza notiunea abstracta de stare a jocului/programului. Programul este structurat sub forma unui "pachet" cu mai multe stari, a caror scop este de a-l ghida pe utilizator de la meniul de baza spre jocul propriu-zis sau spre o "pagina" cu informatii despre functionalitatea si regulile jocului.

Aceste stari ajuta la organizarea jocului in mai multe etape (niveluri) dar si la creearea slide-urilor intermediare dintre nivelul 1 si 2, care ii ofera informatii utilizatorului despre suma acumulata de banuti, cat si a scenariilor de finalitate a jocului, fie ea cu un status de castigator sau pierzator (WinState2, GameOverState).
Pe lângă metodele specifice majorității claselor din acest program care se ocupă cu actalizarea stării curente (Update()) și cu desenarea ei (Draw()), clasa State conține și un setter ce se ocupă cu salvarea stării anterioare și a stării curente, dar și un getter ce returnează starea curentă a jocului.

MenuState

Implementeaza noțiunea de meniu a jocului și se ocupă cu afișarea unui “panou” de navigare prin program, care prin intermediul a 5 butoane îi permite utilizatorului să întreprindă diverse acțiuni:

Butonul NewGame: pornește un joc nou.

Butonul Save: salvează progresul jucătorului.

Butonul Load: deschide o pagină în care utilizatorul poate continua un joc pornit anterior.

Butonul About: deschide o pagină de informații legate de controllers, povestea jocului și reguli.

Butonul Quit: oprește programul din execuție.

AboutState

Implementeaza notiunea de manual de instructiuni/despre joc. Contine informatii despre controllers, povestea jocului pentru intelegerea contextului jocului si informatii despre regulament. Este prevăzut sub

forma unei galerii de pagini, care pot fi accesate prin butoanele Next și Back.

LoadState

Clasa ce se ocupă de starea programului in care sunt afisate progresele în joc ale utilizatorilor, reprezentând totodată un portal către momentul de joc la care au ramas in trecut jucatorii. Este prevăzută sub forma unei galerii cu 4 chenare, care se umplu cu datele utilizatorului în momentul în care acesta apasă butonul Save din meniu.

PlayState și PlayState2

Sunt clasele ce implementeaza/controleaza cele doua niveluri ale jocului, avand ca membrii toate instantele claselor de care va fi nevoie pentru a pune in scena cele doua etape de joc.

Obiectul de tip jucător (Player) si inamic (Enemy/Enemy 2) sunt nelipsite, întrucat lupta in ambele niveluri se duce intre cele doua caractere de tip Personaj. De o importanta majora sunt si obiectele de tip Bullet (nivel 1) și Flower, Pine, Tree și Land (nivel 2) care interacționează în mod direct cu Playerul, fiind propulsate din obiectul de tip Enemy/Enemy 2, care are contact prin intermediul lor cu jucătorul. Atât obiectele de tip Life care informează utilizatorul cu privire la starea curentă a vieților care i-au mai rămas în nivelul curent cât și obiectele de tip Money care trebuie colectati in numar de minim 10 de catre jucator pentru a castiga statutul de SUPERWINNER si nu de simplu WINNER, sunt esentiale pentru implementarea celor doua staturi mentionate in subtitlu.

Toata actiunea jocului este implementata in functia Update() a celor doua clase, iar desenarea prin functia Draw(). Metodele care adaugă noi instanțe ce întruchipează gloanțele/obstacolele (addGun(Bullet gun) , addFlower(Flower floare)) sunt esențiale pentru asigurarea continuitatii jocului dar si a finalitatii acestuia, stabilindu-se in cadrul ei in functie de variabila contorGun de tip int (care memoreaza cate gloante au mai ramas inamicului , decrementandu-se in momentul in care metoda este apelata) daca inamicul a ramas fara munitie si jucatorul poate fi declarat castigator al nivelului.

GameOverState, WinState și WinState2

Implementează noțiunie de sfârșit de nivel/joc.

Reprezinta stările care informează utilizatorul cu privire la statusul sau dupa incheierea unui nivel in cazul in care acesta castiga (WinState pentru

nivel 1 si WinState2 pentru nivel 2), sau chiar pe parcursul acestuia in cazul in care si-a piedut toate vietile si jocul se considera a fi pierdut și încheiat (GameOverState).
WinState realizează tranziția dintre cele două niveluri dacă nivelul 1 a fost dus cu succes la final prin intermediul butonului Play, dar rolul acestei stări este și să-i aducă la cunoștință utilizatorului că a trecut cu succes de prima etapă a jocului.
WinState2 este starea care i se dezvăluie utilizatorului doar în momentul în care câștigă și cel de-al doilea nivel al jocului. În cadrul ei, se va afișa prima dată pe ecran faptul că utilizatorul a finalizat cu succes și această etapă, urmată de o animație cu Phoenixia preschimbându-se în om când ajunge la destinație. Următoarul lucru care se va afișa va fi suma totală de bănuți colectată pe parcursul celor două niveluri. Se va specifica dacă jucătorul a strâns numărul suficient de bani ca să capete statusul de SUPERWINNER, caz în care se va afișa o animație care o întruchipează pe Phoenixia mergand spre Palat. În cazul contrar, statusul jucatorului va fi de simplu WINNER si pe ecran se va afisa o imagine in care Phoenixia rămâne blocată pe Insula de Azur.

Șabloane de proiectare utilizate

1. **State**
   Modelul State permite unui obiect să își schimbe comportamentul când starea sa interna se modifică. Schimbarea are loc prin folosirea moștenirii, clasele derivatate reprezentând stările și funcționalitatea programului.
   În acest sens am creat pachetul *States* care are la bază clasa abstractă *State* din care derivă cele 8 clase ce vor implementa noțiunea de meniu, pagină cu înregistrări, pagină cu informații/reguli, nivel 1/ nivel 2, ipostaza de joc pierdut, ipostaza de nivel 1 câstigat și ipostaza de joc câștigat care o implică și pe cea de nivel 2 câștigat.

2. **Singleton**
   Scopul modelului Singleton este de a asigura faptul că o clasă are o singură instanță și de a furniza un singur punct de acces global catre această instanță.
   Am utilizat acest model în crearea claselor Database pentru a asigura realizarea unei singure conexiuni între baza de date și program, dar și în formarea clasei GlobalInit(), pentru a permite un singur tip de inițializare globală.

Bibliografie

Butoanele și fundalurile nivelurilor au fost realizate în aplicația Adobe Illustrator , iar toate celelalte elemente din Game Sprite au fost creeate pe aplicația online https://www.piskelapp.com/p/create/sprite .

Fundalul și titlul jocului din meniu au fost realizat cu ajutorul site-ului https://www.canva.com/

Desenele în pixeli au fost inspirate din imagini găsite pe internet, majoritatea fiind adaptate de propria mea imaginație pentru a fi cât mai unicate. Unele dintre site-urile pe care am găsit imagini în pixeli pe care le-am folosit ca inspirație sunt:

https://www.dreamstime.com/

https://www.istockphoto.com/ro/vector/vector-pixel-art-izolat-elicopter-dron%C4%83-gm1211336953-351252024

https://vector-images.com/clipart/clp4154361/

https://www.pinterest.cl/pin/607352699737701741/