Around the world

Stati Andreea

1210A

ETAPA 1

Povestea jocului:

Cristian este un adolescent care duce o viață aparent liniștită, într-un cartier liniștit, alături de familia sa și de un cerc restrâns de prieteni. Cea mai mare dorință a sa este să aibă parte de o călătorie în jurul lumii, în care să viziteze diverse forme de relief, de vegetație, floră și faună diversă. Plictisit de monotonia și rutina propriei vieți, de ziua sa de naștere și-a pus cea mai mare dorință a sa. Ce nu știa el însă este faptul că dorința de la 16 ani a oricărui adolescent se îndeplinește. Astfel, Cristian s-a trezit în următoarea zi într-o junglă ciudată, cu animale și plante vorbitoare, cu forme și dimensiuni atipice.

Speriat, dar în același timp entuziasmat că are parte de ceva palpitant în viața sa, acesta începe să exploreze împrejurimile. Ulterior, conștientizează faptul că lumea în care a ajuns e doar o iluzie, el fiind singura ființă umană în acea junglă plină de pericole. Începuse să-și regrete dorința avută și să-i lipsească tot mai mult viața liniștită pe care o avea până atunci și siguranța pe care aceasta i-o oferea.

În timp ce căuta o cale prin care ar putea să se întoarcă la realitate, aude o discuție între doi tucani despre faptul că zonele din jur sunt pline de inamici primejdioși și despre un grup de maimuțe care își pierduse bananele pe potecile pădurii cu o zi înainte. Atunci Cristian a înțeles că acestea reprezintă hrana care îl va ține în viață și îi va da energie în drumul său și că uciderea inamicilor este calea sa de evadare din junglă. Drumul este presărat cu obstacole pe care băiatul trebuie să le evite pentru a rămâne în viață și a se putea întoarce la realitate.

După un drum lung și primejdios, Cristian ajunge la realitate după uciderea inamicilor.

Mecanica jocului:

- **Personajele jocului**:

  Cristian este singurul jucător. Rolul lui este de a rămâne în viață și de a aduna comorile pe care le găsește în drumul său. El este personajul principal din propria poveste și reușește să lupte cu obstacolele care îi ies în cale dând dovadă de stăpânire de sine.

  Inamicii (florile carnivore, țestoasele, mumiile) se deplasează doar stânga-dreapta și pot să-l atace. La contactul cu acestea, personajul pierde din viață un anumit număr, în funcție de tipul inamicului. Atunci când personajul se află în raza lor vizuală vor începe să-l urmărească, schimbându-și sensul de mers în funcție de cel al personajului. Modalitatea prin care aceștia pot fi uciți este prin intermediul click-ului.

- **Obiectele:**

  Cuferele sunt statice: jucătorul trebuie să le spargă prin intermediul click-ului, la deschiderea acestora ieșind banane sau ananași.

  Fierăstraiele sunt componente periculoase, iar la contactul dintre jucător și acestea, protagonistul își pierde viața.

  Bananele și ananașii sunt statice, însă colectarea lor ajută la atingerea unui nivel al vieții, repsectiv al energiei mai mare.

  Tunurile sesizează jucătorul de la o distanță cuprinsă între 1 și 5 tile-uri. Cu fiecare minge încasată de acesta, jucătorul pierde din viață, tocmai de aceea ar fi bine să încerce evitarea mingilor.

- **Componente active:**

  Tabla de joc este compusă din platforme de diferite lungimi și forme pe care jucătorul poate merge/sări. Firăstraiele influențează interacțiunea personajului, rezultând uciderea lui.

- **Modul în care pornește jocul:**

  Se deschide jocul și se pune în acțiune prin intermediul butonului ”START” după care va începe primul nivel al jocului.

- **Cum se joacă?**

  Jocul este unul de tip platformă 2D, surprins dintr-o perspectivă side view, cu un singur jucător ce este controlat prin intermediul tastelor direcționale (A- stânga, D-dreapta, W-sus) fiind afectat de gravitate.

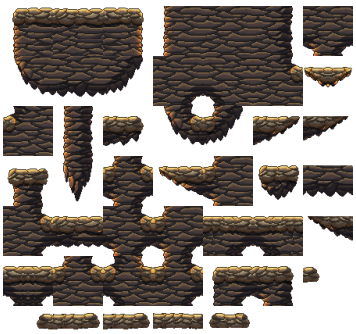
Obiectivul jocului este de a învinge inamicii care apar pe parcurs și să colecteze cât mai multe bonusuri.

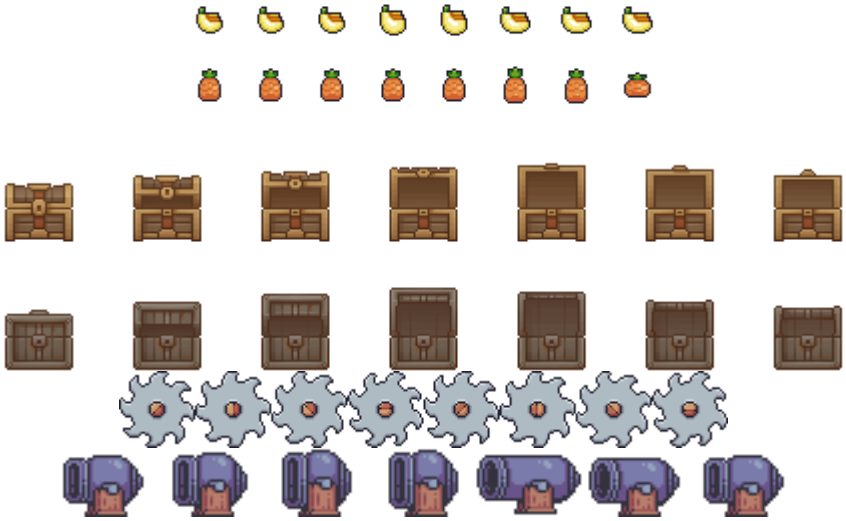
- **Cum se completează nivelurile?**

  Pentru a trece de fiecare nivel, jucătorul trebuie să reușească să treacă de toate obstacolele și de inamici.

Definirea caracterelor:

Game sprite-uri:





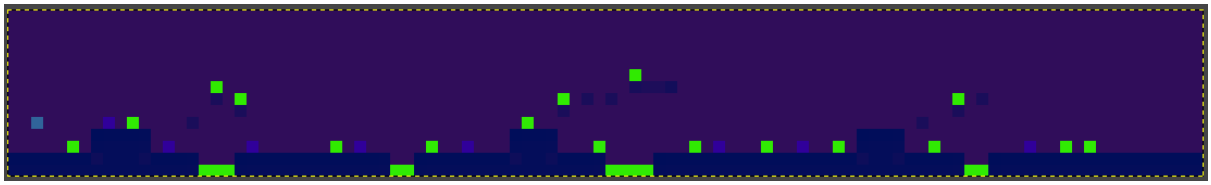
Atacarea inamicilor:

Principalul mod de atacare al inamicilor este atacarea acestora prin click. În cazul în care personajul principal se ciocnește cu inamicii pe orizontală, acesta își pierde viața până când nivelul se resetează.

Proiectarea nivelurilor

Definirea hărții:

Harta jocului este o reprezentare grafică sau textuală a terenului, care arată obstacolele, resursele, locațiile și alte elemente importante.



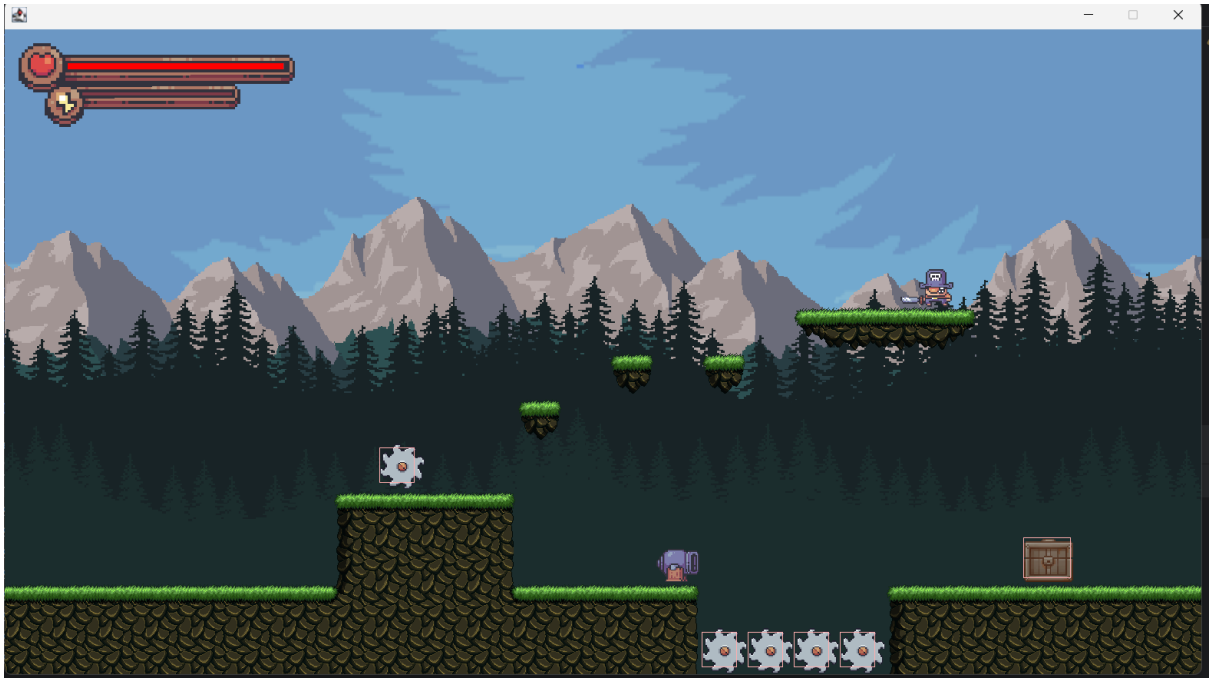
Modul de construire a hărților este prin o imagine colorată de acest fel care are 14 pixeli în înălțime și minim 24 în lățime. Fiecare pixel reprezintă o culoare RGB. Valoarea culorii roșu ne dă tile-urile. În cazul jocului, atlas-ul pentru fiecare nivel are 49 tile-uri, fiecare având câte un indice de la 0 la 48 în ordinea parcurgerii matricii. Valoarea 48 pentru roșu reprezintă absența unui tile, iar această valoare este folosită pentru a reprezenta celelalte elemente ale jocului sau fundalul curat.

Valoarea culorii albastru ne dă entitățile jocului : 100 = locul din care pornește playerul, 0 = florile, 1=țestoasele și 2=mumiile.

Valoarea culorii albastru ne dă obiectele jocului: 0=banane, 1=ananași, 2 și 3 = cuferele, 4 = fierăstraiele, 5=tun îndreptat spre stânga și 6 = tun îndreptat spre dreapta.

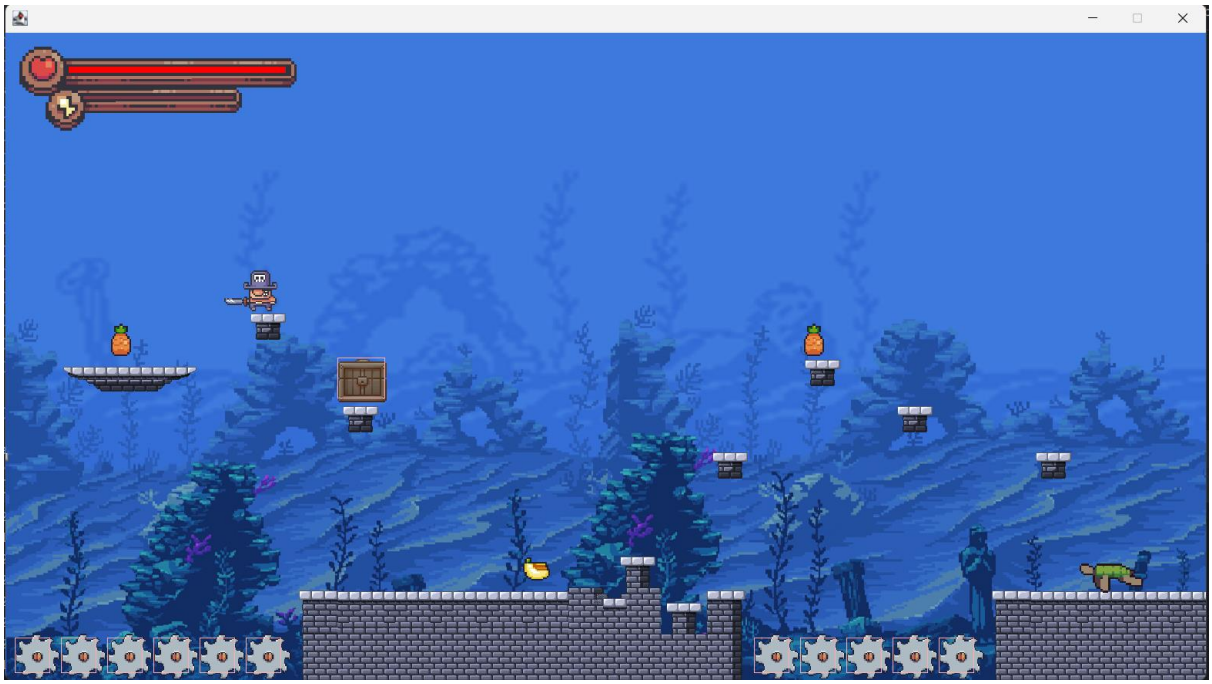
Descrierea nivel urilor

Nivel 1: Personajul se confruntă cu florile ucigașe. Acțiunea se petrece într-un mediu forestier, de junglă sau pădure. Pentru a trece la nivelul următor e necesară uciderea florilor.

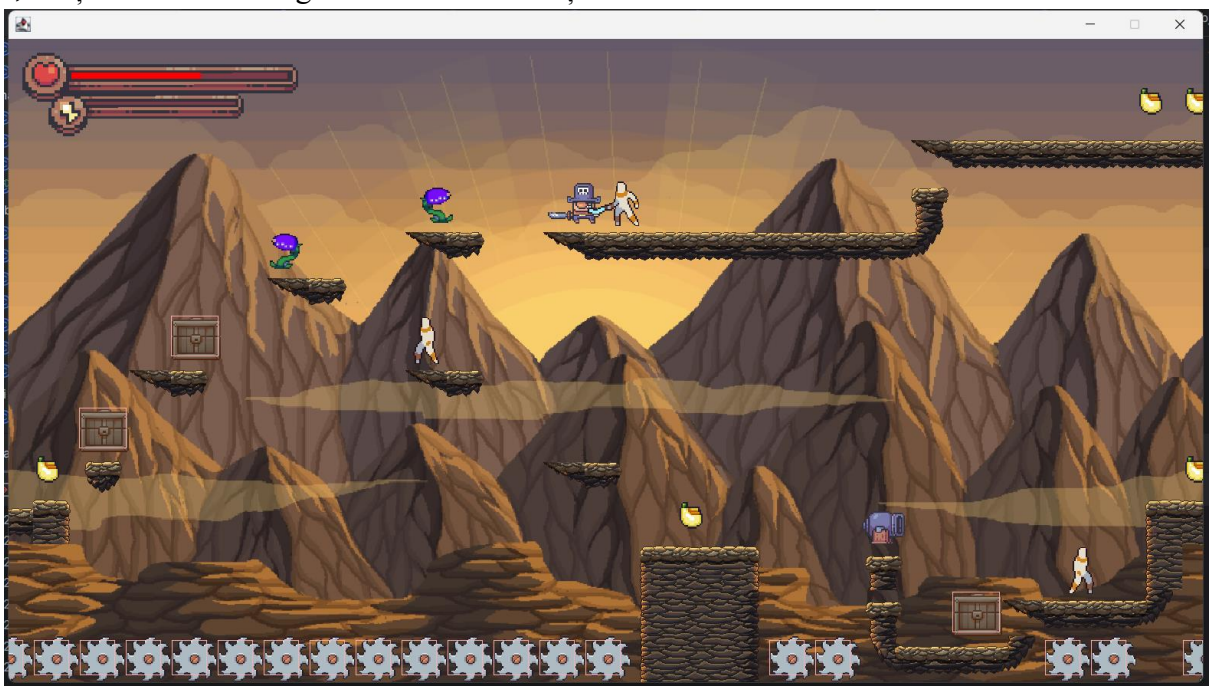


Nivel 2: Cadrul spațial din acest nivel se schimbă, acțiunea desfășurându-

se într-un mediu subacvatic. De această dată, inamicii sunt florile carnivore și țestoasele. Acestea au același comportament ca și inamicii din nivelul precedent, iar pentru a-i elimina e necesară atacarea. Pentru a aduna avea un nivel al vieții cât mai trebuie colectate bonusurile.

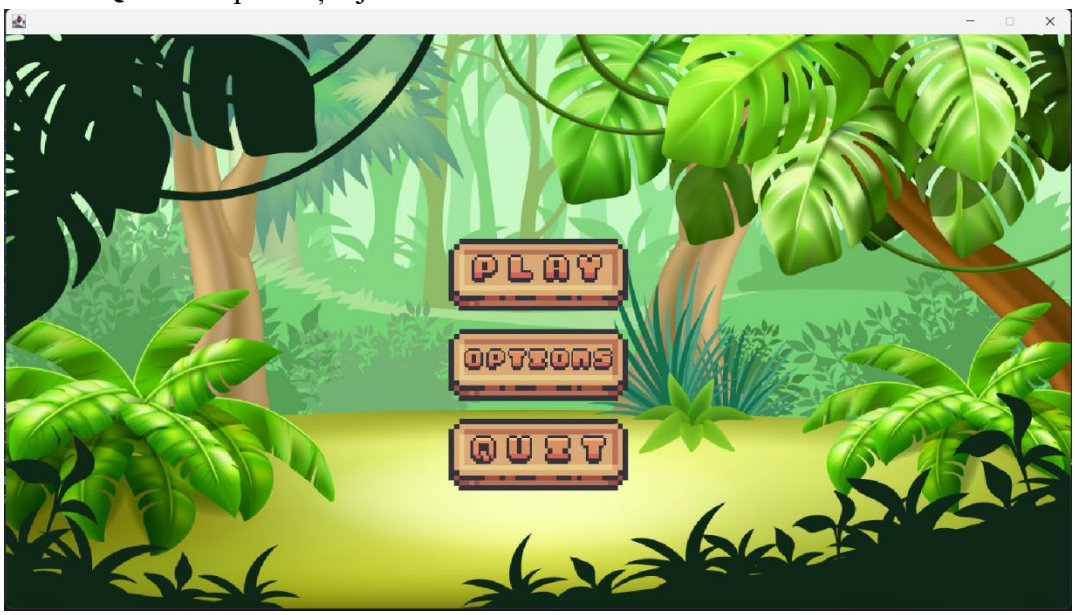


Nivel 3: În acest nivel, acțiunea se desfășoară în deșert. Inamicii sunt mumiile, aceștia fiind cel mai greu de ucis dintre toți.



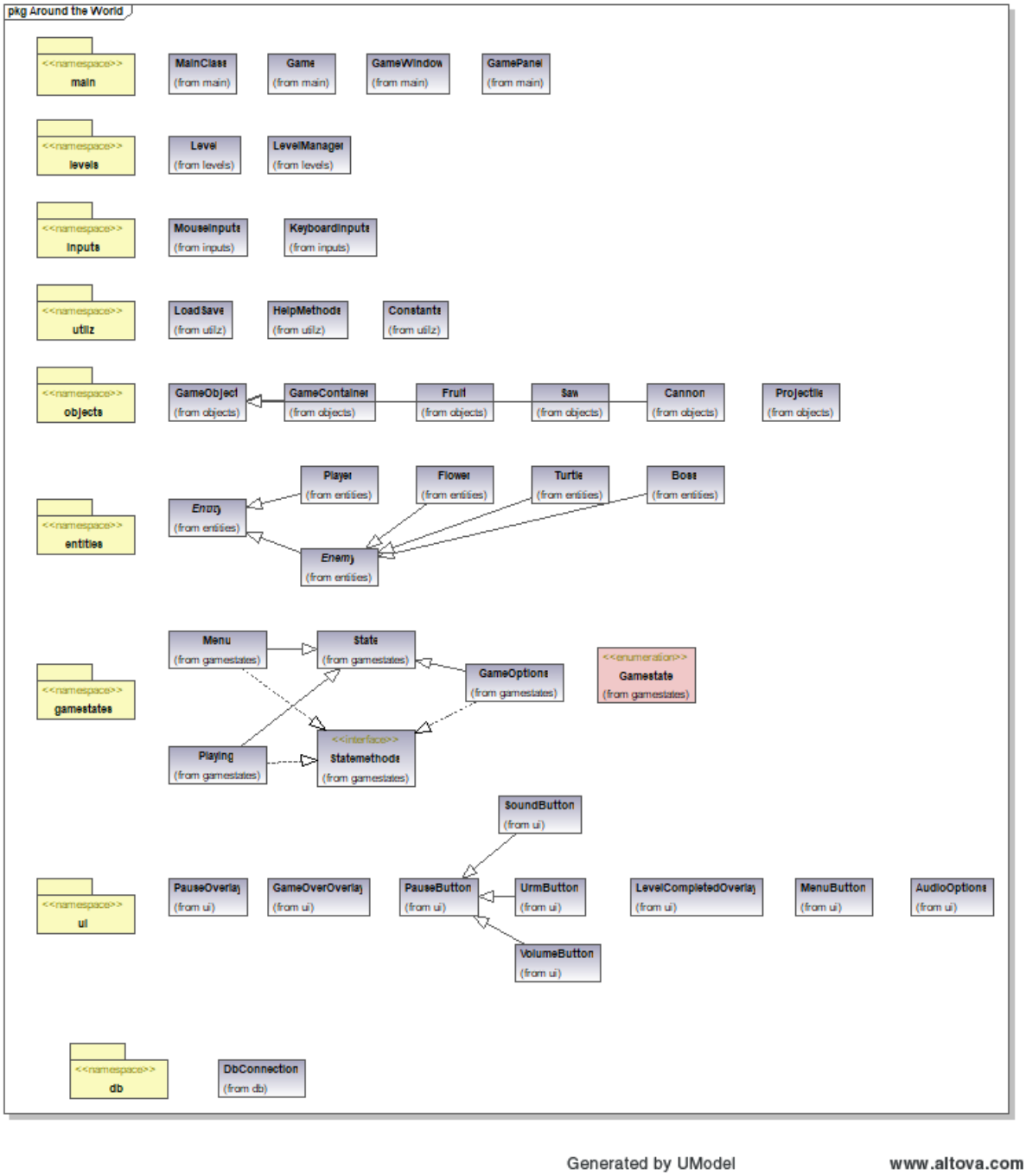
Definirea interfeței

- **PLAY**: se începe jocul de la nivelul 1
- **OPTIONS**: pentru a modifica sunetul
- **QUIT**: se părăsește jocul



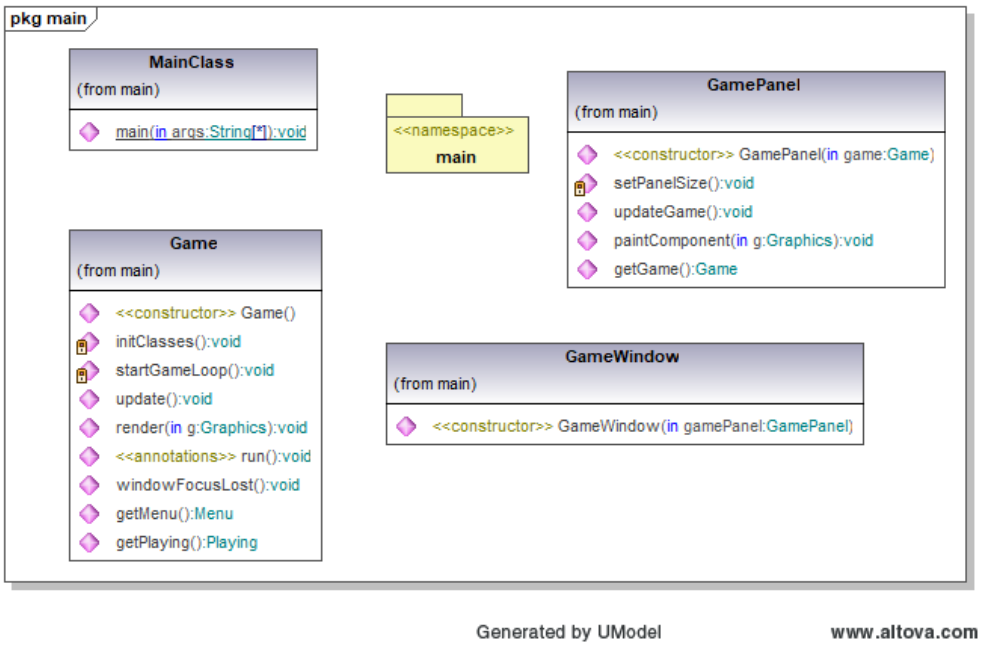
Etapa 2

Diagrame UML



Proiectul este structurat în mai multe package-uri din care fac parte mai multe clasa.

Package main:



Clasa MainClass

În această clasa se creează metoda principală main prin intermediul căreia se rulează programul. În cadrul acesteia se creează o instanță de tipul Game.

Clasa Game

Constructorul Game():

Inițializează obiectele necesare pentru afișarea jocului și începe bucla principală a jocului prin apelul metodei startGameLoop().

Metoda initClasses():

Inițializează obiectele menu și playing care reprezintă stările de joc din meniu și jocul propriu-zis.

Metoda startGameLoop():

Creează și pornește un nou fir de execuție (Thread) care va gestiona bucla de joc.

Metoda update():

Actualizează starea jocului în funcție de starea curentă a jocului (Gamestate.state). Dacă jocul se află în starea de meniu, se actualizează meniul, iar dacă se află în starea de joc, se actualizează jocul în sine.

Metoda render(Graphics g):

Desenează starea curentă a jocului în funcție de starea curentă a jocului (Gamestate.state). Dacă jocul se află în starea de meniu, se desenează meniul, iar dacă se află în starea de joc, se desenează jocul propriu-zis.

Metoda run() (implementată din interfața Runnable):

Reprezintă bucla principală a jocului. Calculează și menține un număr constant de cadre pe secundă (FPS) și actualizări pe secundă (UPS) pentru a menține jocul la o viteză constantă.

Metoda windowFocusLost():

Este apelată atunci când fereastra jocului își pierde focusul. În această metodă, dacă jocul se află în starea de joc (Gamestate.PLAYING), se resetează direcțiile jucătorului.

Metoda getMenu():

Returnează obiectul menu, care reprezintă starea de meniu a jocului.

Metoda getPlaying():

Returnează obiectul playing, care reprezintă starea de joc a jocului propriu-zis.

Clasa Game Window:

Constructorul GameWindow(GamePanel gamePanel):

Primește un obiect GamePanel și creează o fereastră (JFrame) pentru joc.

Setează operația implicită la închiderea ferestrei pentru a închide aplicația atunci când fereastra este închisă.

Adaugă panoul de joc (GamePanel) la fereastră.

Dezactivează redimensionarea ferestrei pentru a menține dimensiunile fixe ale jocului.

Redimensionează fereastra pentru a se potrivi conținutului (panoului de joc).

Centrează fereastra pe ecran și o face vizibilă.

Clasa internă WindowFocusListener:

Implementează interfața WindowFocusListener pentru a asculta evenimentele legate de focusul ferestrei.

În metoda windowLostFocus(WindowEvent e), când fereastra își pierde focusul, se apelează metoda windowFocusLost() a obiectului GamePanel, care la rândul său apelează metoda corespunzătoare din clasa Game pentru a trata pierderea focusului în contextul jocului.

Metoda windowGainedFocus(WindowEvent e) este lăsată necompletată, deoarece momentan nu este implementată nicio logică pentru când fereastra își recâștigă focusul.

Această clasă facilitează crearea și gestionarea ferestrei jocului și a interacțiunii utilizatorului cu aceasta. Este responsabilă și pentru comunicarea între interacțiunile utilizatorului și logica jocului în cazul evenimentelor legate de focusul ferestrei.

Clasa GamePanel:

Constructorul GamePanel(Game game):

Inițializează obiectele necesare pentru gestionarea evenimentelor de tastatură și mouse, și primește o referință la obiectul Game pentru a putea comunica cu acesta.

Apelează metoda setPanelSize() pentru a seta dimensiunea panoului de afișare.

Metoda setPanelSize():

Stabilește dimensiunea panoului de afișare conform dimensiunilor definite în clasa Game (GAME_WIDTH și GAME_HEIGHT).

Metoda updateGame():

Momentan nu implementează nicio funcționalitate, dar poate fi utilizată pentru actualizarea jocului în viitor, dacă este necesar.

Metoda paintComponent(Graphics g):

Suprascrie metoda paintComponent() pentru a desena grafica jocului utilizând obiectul Graphics primit ca argument.

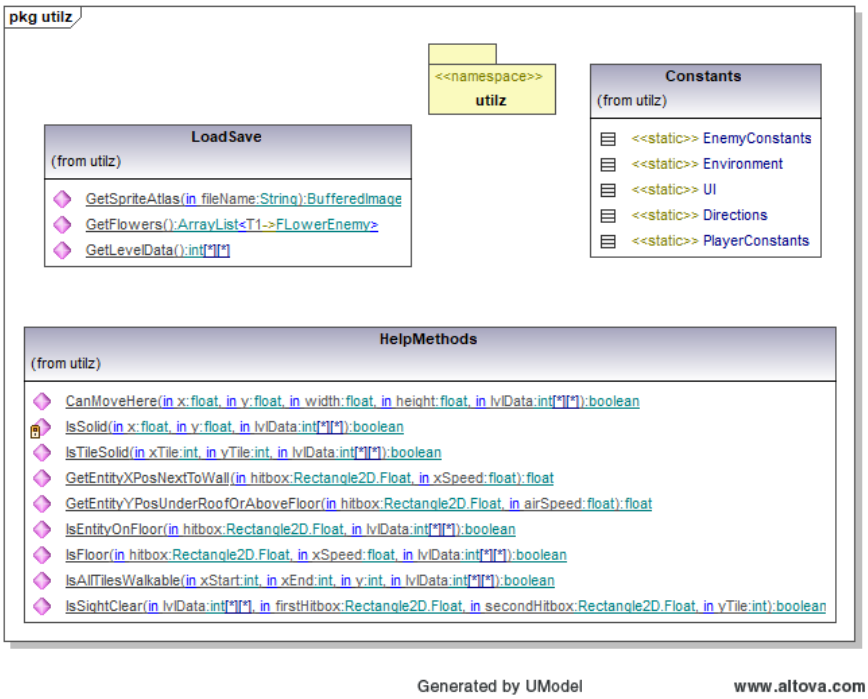
Apelează metoda render(Graphics g) a obiectului Game pentru a desena starea curentă a jocului.

Metoda getGame():

Returnează obiectul game, care reprezintă instanța clasei Game asociată cu acest panou de joc. Această referință este utilă pentru a accesa metodele și datele din clasa Game.

Această clasă joacă un rol important în interfața grafică a jocului și în comunicarea între interacțiunea utilizatorului și logica jocului.

Package utilz:



Clasa LoadSave

Clasa gestionează încărcarea și salvarea resurselor jocului. Iată o descriere a fiecărei metode și a rolului său:

Metoda GetSpriteAtlas(String fileName):

Primește numele fișierului și încearcă să încarce imaginea corespunzătoare din resursele jocului.

Returnează imaginea încărcată sub formă de obiect BufferedImage.

Metoda GetFlowers():

Încarcă imaginea LEVEL_ONE_DATA și parcurge fiecare pixel.

Dacă găsește un pixel cu culoarea corespunzătoare florei (FLOWER), adaugă o nouă floare la acea poziție în listă.

Returnează lista de flori.

Metoda GetLevelData():

Încarcă imaginea LEVEL_ONE_DATA și parcurge fiecare pixel.

Extrage informațiile despre nivel din culorile pixelilor, folosind canalele de culoare ale acestora.

Returnează o matrice de dimensiuni corespunzătoare imaginii, reprezentând datele nivelului.

Această clasă este esențială pentru încărcarea și interpretarea datelor necesare pentru desenarea nivelului și a altor elemente grafice din joc. Ea ajută la organizarea și gestionarea resurselor grafice și de date din joc.

Clasa HelpMethods

Această clasă, HelpMethods, conține metode utile pentru diverse operații în joc.

Metoda CanMoveHere(float x, float y, float width, float height, int[][] lvlData):

Verifică dacă entitatea poate să se miște la poziția specificată fără să intre în coliziune cu tile-urile solide din nivel (lvlData).

Folosește metoda IsSolid() pentru a verifica dacă fiecare colț al hitbox-ului entității este solid.

Metoda privată IsSolid(float x, float y, int[][] lvlData):

Verifică dacă punctul specificat este solid (adică reprezintă o coliziune) pe baza datelor de nivel (lvlData).

Verifică, de asemenea, dacă punctul este în afara marginilor nivelului.

Metoda IsTileSolid(int xTile, int yTile, int[][] lvlData):

Verifică dacă un tile specific este solid sau nu pe baza datelor de nivel (lvlData).

Metodele GetEntityXPosNextToWall() și GetEntityYPosUnderRoofOrAboveFloor():

Determină poziția pe axa X și, respectiv, pe axa Y a unei entități în raport cu un perete sau podea în funcție de viteza acesteia.

Metoda IsEntityOnFloor(Rectangle2D.Float hitbox, int[][] lvlData):

Verifică dacă entitatea se află pe podea.

Metoda IsFloor(Rectangle2D.Float hitbox, float xSpeed, int[][] lvlData):

Verifică dacă entitatea se află pe podea, luând în considerare viteza acesteia pe axa X.

Metoda IsAllTilesWalkable(int xStart, int xEnd, int y, int[][] lvlData):

Verifică dacă toate tile-urile între două coordonate X și o coordonată Y sunt părți accesibile ale nivelului.

Metoda IsSightClear(int[][] lvlData, Rectangle2D.Float firstHitbox, Rectangle2D.Float secondHitbox, int yTile):

Verifică dacă liniile de vizibilitate între două hitbox-uri nu sunt blocate de obiecte solide din nivel.

Aceste metode sunt esențiale pentru gestionarea mișcării și coliziunilor în joc și sunt utile pentru determinarea poziției și interacțiunilor entităților cu mediul din jurul lor.

Clasa Constants

Clasa Constants conține constante și metode utile pentru diferite aspecte ale jocului.

Clasa internă EnemyConstants:

Conține constante referitoare la inamici, cum ar fi tipul de inamic (în acest caz, FLOWER), stări ale inamicilor (idle, running, attack, hurt, dead), dimensiunile și alte detalii specifice inamicilor.

Metode precum GetSpriteAmount, GetMaxHealth și GetEnemyDamage oferă valori specifice inamicilor în funcție de tipul acestora.

Clasa internă Environment:

Conține constante pentru elementele de mediu, cum ar fi norii mari și mici, precum și dimensiunile lor.

Clasa internă UI:

Conține subclase pentru butoanele UI, butoanele de pauză și alte elemente de interfață utilizator.

Clasa internă Directions:

Conține constante pentru direcțiile posibile ale mișcării (stânga, dreapta, sus, jos).

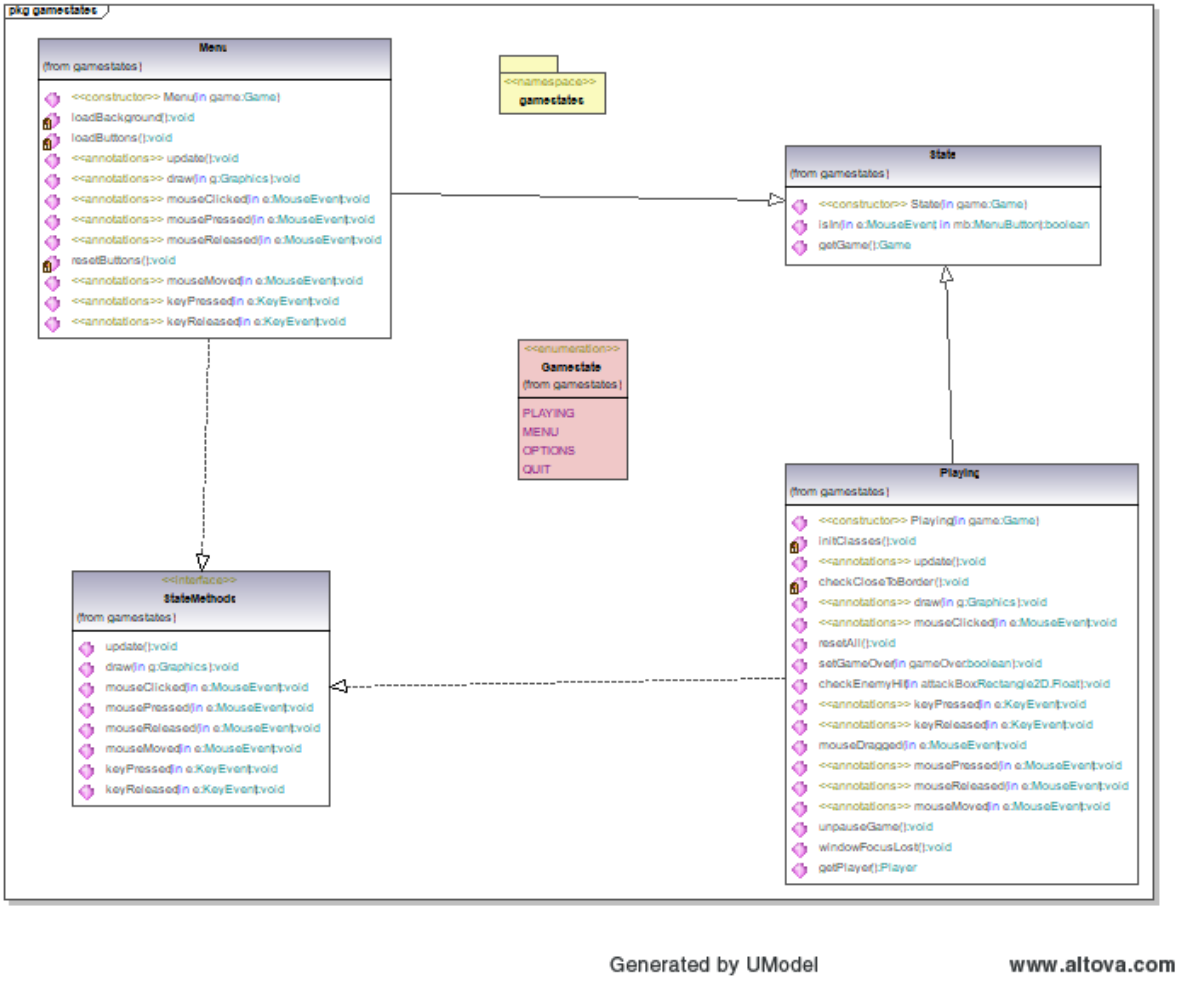
Clasa internă PlayerConstants:

Conține constante pentru stările și acțiunile jucătorului, cum ar fi staționarea, alergarea, săritura, căderea, atacul și altele.

Metoda GetSpriteAmount oferă numărul de cadre de animație necesare pentru fiecare acțiune a jucătorului.

Aceste constante și metode sunt utilizate pentru a face codul mai ușor de înțeles și de întreținut și pentru a asigura coerența în întregul proiect. Ele sunt utile pentru a defini și gestiona diferite aspecte ale jocului, precum inamicii, elementele de mediu, interfața utilizatorului și acțiunile jucătorului.

Package gamestates



Enum Gamestates

Fisierul reprezintă stările posibile ale jocului.

PLAYING: Starea în care jocul este în desfășurare, iar jucătorul controlează personajul sau interacționează cu mediul de joc.

MENU: Starea în care jocul afișează meniul principal, oferind opțiuni cum ar fi începerea jocului, accesul la opțiuni sau ieșirea din joc.

OPTIONS: Starea în care jucătorul poate modifica setările jocului, cum ar fi sunetul, grafica sau controalele.

QUIT: Starea în care jocul se pregătește să se închidă.

Variabila statică state este folosită pentru a reține starea curentă a jocului și este inițializată cu valoarea MENU, indicând că jocul începe cu afișarea meniului principal. Această variabilă poate fi accesată și modificată de la orice loc în cod, permițând controlul și schimbarea stării jocului în diferitele sale etape.

Clasa Menu

Clasa gestionează starea meniului din joc. Iată o scurtă descriere a metodelor și funcționalităților sale:

Constructorul Menu(Game game):

Inițializează meniul, încărcând butoanele și fundalul.

Metoda update():

Actualizează starea butoanelor din meniu.

Metoda draw(Graphics g):

Desenează fundalul și butoanele meniului.

Metodele pentru gestionarea evenimentelor de mouse (mousePressed, mouseReleased, mouseMoved, mouseClicked):

Detectează evenimentele de mouse și reacționează în consecință.

Verifică dacă butoanele sunt apăsate, eliberate sau mișcate peste și activează starea corespunzătoare a jocului în funcție de butonul apăsat.

Metoda keyPressed(KeyEvent e):

Detectează apăsările de taste și, în cazul apăsării tastei Enter, schimbă starea jocului la PLAYING.

Această clasă permite navigarea și interacțiunea utilizatorului cu meniul jocului, inclusiv schimbarea stării jocului și reacționarea la evenimente de mouse și tastatură.

Clasa Playing

Clasa este responsabilă de gestionarea stării jocului în timpul desfășurării acestuia.

Constructorul Playing(Game game):

Inițializează obiectele necesare pentru desfășurarea jocului, cum ar fi jucătorul, nivelul și managerul de inamici.

Metoda update():

Actualizează starea jocului și a obiectelor din el în funcție de stadiul jocului (în pauză, nivel completat sau joc terminat).

Metoda draw(Graphics g):

Desenează fundalul, nivelul, jucătorul și inamicii în funcție de stadiul jocului.

Metodele pentru gestionarea evenimentelor de mouse (mousePressed, mouseReleased, mouseMoved, mouseClicked, mouseDragged):

Detectează evenimentele de mouse și reacționează în consecință, cum ar fi atacul jucătorului sau interacțiunea cu meniul de pauză.

Metodele pentru gestionarea evenimentelor de tastatură (keyPressed, keyReleased):

Detectează apăsările și eliberările de taste și controlează mișcarea și acțiunile jucătorului.

Metoda checkEnemyHit(Rectangle2D.Float attackBox):

Verifică dacă atacul jucătorului lovește vreun inamic și, în caz afirmativ, actualizează starea inamicului.

Această clasă gestionează în mod activ interacțiunea utilizatorului cu jocul și actualizează starea acestuia în funcție de acțiunile jucătorului și de evenimentele din joc.

Clasa State

Această clasă servește ca bază pentru alte stări din joc, precum meniul sau jocul propriu-zis.

Constructorul State(Game game):

Inițializează obiectul de joc asociat acestei stări.

Metoda isIn(MouseEvent e, MenuButton mb):

Verifică dacă coordonatele evenimentului de mouse (e) se află în interiorul dreptunghiului de coliziune al unui buton de meniu (mb).

Este folosită pentru detectarea interacțiunilor cu butoanele de pe ecran.

Metoda getGame():

Furnizează acces la obiectul de joc asociat stării curente.

Această clasă oferă o funcționalitate de bază comună pentru stările din joc și facilitează interacțiunea cu obiectul de joc și elementele grafice ale acestuia.

Clasa StateMethods

Această interfață, StateMethods, definește metodele necesare pentru gestionarea stărilor din joc. Aceste metode sunt implementate de către clasele care reprezintă diferite stări ale jocului, cum ar fi meniul sau jocul propriu-zis. Iată o scurtă descriere a fiecărei metode:

update(): Actualizează starea stării jocului.

draw(Graphics g): Desenează elementele stării jocului pe ecran.

mouseClicked(MouseEvent e): Reacționează la un clic al mouse-ului.

mousePressed(MouseEvent e): Reacționează la apăsarea unei taste de mouse.

mouseReleased(MouseEvent e): Reacționează la eliberarea unei taste de mouse.

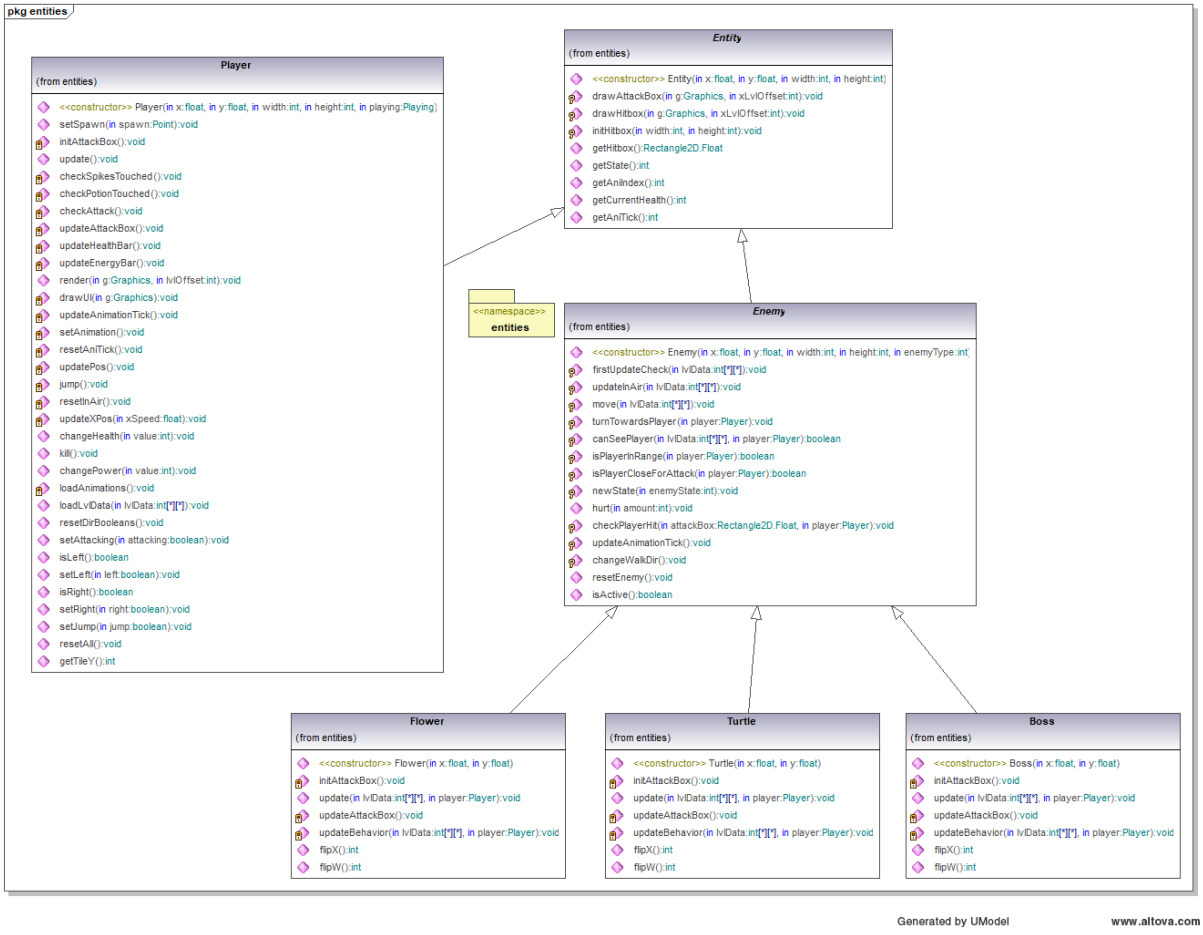
mouseMoved(MouseEvent e): Reacționează la mișcarea mouse-ului.

keyPressed(KeyEvent e): Reacționează la apăsarea unei taste de pe tastatură.

keyReleased(KeyEvent e): Reacționează la eliberarea unei taste de pe tastatură.

Această interfață definește un set standard de metode care trebuie implementate de către toate stările din joc pentru a gestiona corect interacțiunea cu utilizatorul și actualizarea stării jocului.

Package entities



Clasa Entity

Această clasă abstractă, Entity, servește ca o bază pentru toate entitățile din joc, cum ar fi jucătorul și inamicul.

x, y: Coordonatele poziției entității în planul de joc.
width, height: Dimensiunile entității în planul de joc.
hitbox: Un obiect Rectangle2D.Float care reprezintă zona de coliziune a entității.
Metodele și caracteristicile acestei clase sunt concepute pentru a oferi funcționalitate de bază necesară tuturor entităților din joc:

Constructorul Entity(float x, float y, int width, int height) inițializează poziția și dimensiunile entității.

Metoda drawHitbox(Graphics g, int xLvlOffset) desenează hitbox-ul entității în scopuri de depanare.

Metoda initHitbox(float x, float y, int width, int height) inițializează hitbox-ul entității la poziția și dimensiunile date.

Metoda getHitbox() furnizează hitbox-ul entității.

Această clasă servește ca o bază pentru alte clase de entități din joc, care vor extinde această clasă și își vor implementa comportamentul specific.

Clasa Player

Această clasă reprezintă entitatea jucătorului din joc și gestionează toate acțiunile și proprietățile asociate cu jucătorul.

Proprietăți ale jucătorului: Acestea includ coordonatele poziției, dimensiunile, hitbox-ul și altele.
Animații: Jucătorul are o varietate de animații pentru diferite acțiuni, cum ar fi mersul, săritul și atacul. Aceste animații sunt încărcate și gestionate în metoda loadAnimations().
Mișcare și sărit: Jucătorul poate mișca spre stânga sau dreapta și poate sări. Logica pentru actualizarea poziției jucătorului și pentru gestionarea săritului este gestionată în metoda updatePos().
Atac: Atunci când jucătorul atacă, se verifică dacă lovitura atinge un inamic. Această verificare se face în metoda checkAttack().
UI: Jucătorul afișează o bara de stare și o bara de sănătate în interfața utilizatorului (UI). Aceste elemente UI sunt desenate în metoda drawUI().
Interacțiune cu nivelul de joc: Jucătorul interacționează cu nivelul de joc și verifica coliziunile cu pereții sau podelele nivelului.
Resetarea stării jucătorului: Există o metodă resetAll() care resetează toate proprietățile jucătorului la valorile inițiale.

Aceasta este o clasă centrală pentru gestionarea jucătorului în joc, și include logica pentru comportamentul jucătorului, interacțiunea cu mediul jocului și afișarea elementelor UI asociate cu jucătorul.

Clasa Enemy

Această clasă abstractă servește drept șablon pentru toți inamicii din joc

Inițializare și caracteristici generale: Clasa inițializează hitbox-ul și alte caracteristici comune pentru toți inamicii, cum ar fi sănătatea maximă și curentă și starea lor activă.

Actualizare și desenare: Metodele update() și draw() sunt folosite pentru a actualiza starea și animațiile inamicilor și pentru a-i desena pe ecran.

Mișcare și detecție: Clasa conține metode pentru gestionarea mișcării inamicilor și pentru detectarea și reacționarea la prezența jucătorului în apropiere.

Animatii și stări: Inamicii pot fi în diferite stări, cum ar fi "atac", "rănit" sau "mort", iar aceste stări sunt gestionate în clasă prin intermediul unui index de animație.

Interacțiune cu jucătorul: Există metode pentru verificarea și gestionarea coliziunilor dintre inamici și jucător, inclusiv cauzarea de daune jucătorului.

Resetare și gestionare a stării: Inamicii pot fi readuși la starea inițială și gestionarea stării lor active este luată în considerare pentru a controla când sunt actualizați sau desenați.

Clasa EnemyManager

Această clasă gestionează inamicii din joc.

Inițializare și încărcare: Clasa încarcă imaginile inamicilor dintr-un fișier de atlas și inițializează lista de inamici.

Actualizare și desenare: Metoda update() este responsabilă pentru actualizarea stării fiecărui inamic, în timp ce metoda draw() se ocupă de desenarea lor pe ecran.

Verificare coliziuni: Clasa are o metodă checkEnemyHit(), care verifică dacă lovitura jucătorului intersectează hitbox-ul unui inamic și, în caz afirmativ, îl afectează corespunzător.

Resetare inamici: Atunci când jocul este resetat, toți inamicii sunt readuși la starea inițială prin apelul metodei resetAllEnemies().

Interacțiune cu jocul: Inamicii sunt actualizați și desenați în contextul clasei Playing, care reprezintă starea de joc actuală.

Această clasă servește drept manager pentru toți inamicii din joc și se asigură că aceștia sunt actualizați și desenați corect în timpul jocului. De asemenea, gestionează coliziunile și alte interacțiuni între inamici și alte elemente ale jocului.

Clasa Flower

Clasa Flower definește comportamentul unei entități inamice specifice din joc, reprezentând o floare. Aceasta gestionează mișcarea și atacul florei, în timp ce reacționează la prezența jucătorului în apropiere. De asemenea, controlează animațiile și starea sănătății entității, având în vedere logica specifică pentru atac și evitarea jucătorului.

Clasa Turtle

Clasa Turtle reglementează comportamentul unei alte entități inamice, și anume o broască țestoasă, în joc. Aceasta gestionează mișcarea și atacul broaștei țestoase, în timp ce detectează și răspunde la prezența jucătorului în apropiere. Totodată, controlează animațiile și sănătatea entității, având în vedere strategia specifică de atac și interacțiunea cu mediul.

Clasa Boss

Clasa Boss este responsabilă pentru definirea comportamentului unui inamic de tip șef, care este un adversar puternic și provocator în joc. Ea gestionează mișcarea și atacul șefului, precum și reacția acestuia la prezența jucătorului în apropiere. De asemenea, controlează animațiile și starea sănătății șefului, având în vedere strategia sa unică de atac și modul în care interacționează cu mediul și cu jucătorul.

Package inputs

Clasele MouseInputs si KeyboardInputs

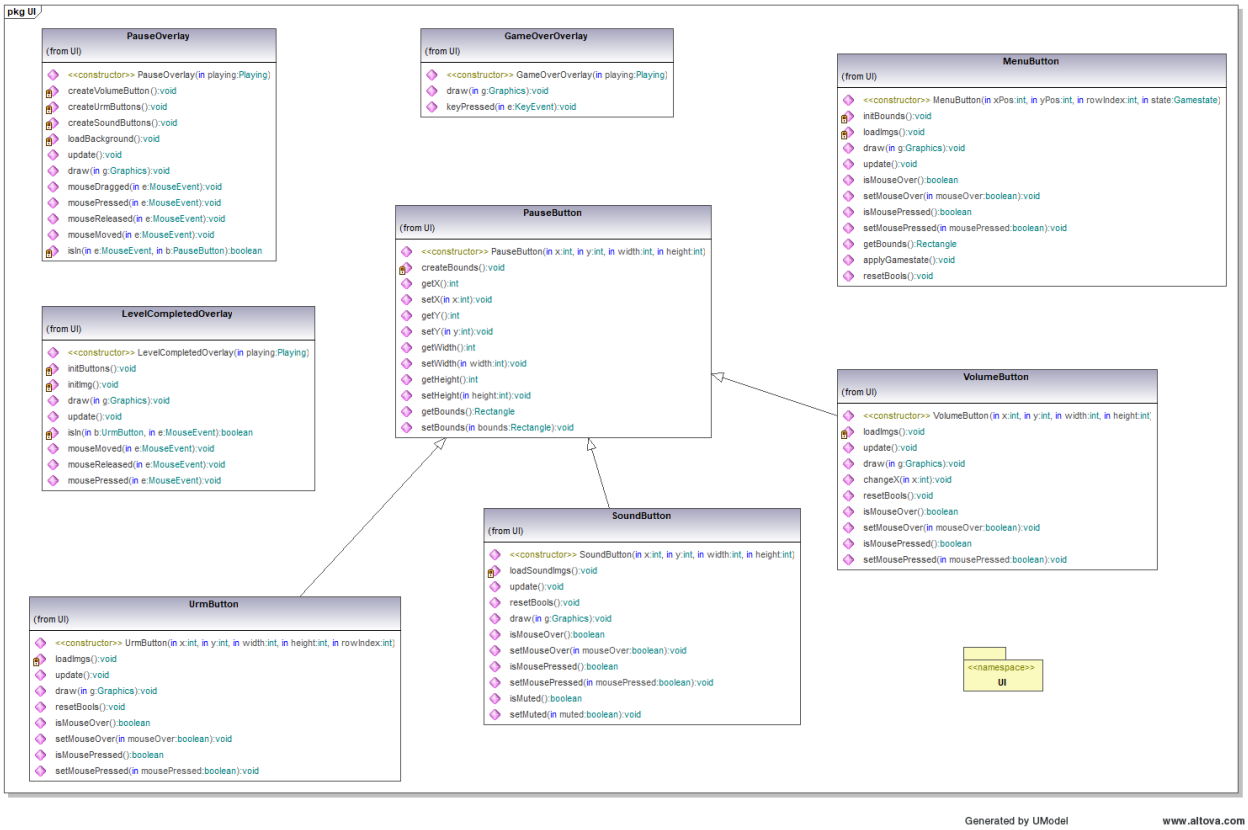
Aceste clase gestionează evenimentele mouse și tastatură în joc. Clasa MouseInputs se ocupă de evenimentele mouse-ului, iar KeyboardInputs de evenimentele tastaturii. Ambele clase verifică starea curentă a jocului și trimit evenimentele către metodele corespunzătoare din meniu sau din jocul propriu-zis. Astfel, se realizează o separare eficientă a logicii de input de logica jocului.

Package levels

Clasele Level si LevelManager

Clasa Level stochează și oferă acces la datele nivelului, inclusiv indexul sprite-ului pentru o anumită poziție pe hartă. LevelManager se ocupă de gestionarea nivelelor și afișarea acestora pe ecran. La inițializare, încarcă sprite-urile nivelului și datele acestuia. Metoda draw desenează nivelul pe ecran, utilizând sprite-urile încărcate și datele nivelului.

Package UI



Fiecare clasă se ocupă de afișarea sau interacțiunea cu diferite ecrane și butoane din joc.

GameOverOverlay desenează un ecran care apare atunci când jocul se încheie și permite jucătorului să revină la meniu.

LevelCompletedOverlay afișează un ecran atunci când nivelul este completat și oferă butoane pentru a merge la următorul nivel sau la meniu.

MenuButton este o clasă pentru butoanele din meniu, care reacționează la evenimentele de suprapunere a cursorului și clic.

PauseButton: Această clasă definește un buton simplu care poate fi folosit în diverse locații ale interfeței utilizatorului. Ea include metode pentru a seta și a obține poziția, dimensiunea și limitele butonului.

PauseOverlay: Această clasă este responsabilă pentru afișarea ecranului de pauză în joc. Ea desenează fundalul ecranului de pauză, butoanele pentru controlul muzicii și efectelor sonore, precum și butoanele pentru meniu, reluare și deblocarea pauzei. De asemenea, gestionează interacțiunile cu mouse-ul, cum ar fi clicurile și mișcările.

Package Objects

Clasa GameObject servește ca șablon pentru obiectele din joc, oferind funcționalități comune și caracteristici de bază, precum poziționarea în joc, gestionarea animațiilor și hitbox-urile asociate. Aceasta este extinsă de clasele specifice obiectelor din joc, precum fructele, containerele, tunurile și alte elemente.

Clasa ObjectManager administrează toate obiectele din joc, inclusiv fructele, containerele, tunurile și proiectilele. Ea coordonează interacțiunile între aceste obiecte și alte entități, cum ar fi jucătorul și inamicii. De asemenea, se ocupă de logica de detectare a coliziunilor și de actualizarea stării și poziției obiectelor.

Clasa GameContainer reprezintă un tip specific de obiect din joc, cum ar fi cutiile sau butoaiele, care pot fi interactate. Ea implementează funcționalități specifice pentru aceste obiecte, cum ar fi gestionarea animațiilor și hitbox-urile asociate.

Clasa Fruit definește comportamentul și caracteristicile fructelor din joc. Ea gestionează animațiile și mișcarea fructelor, precum și efectele asociate atunci când jucătorul interacționează cu acestea.

Clasa Cannon reprezintă tunurile din joc și implementează logica asociată cu acestea, inclusiv gestionarea animațiilor și a proiectilelor trase de tunuri.

Clasa Projectile reprezintă proiectilele trase de tunuri și implementează logica asociată cu acestea, inclusiv actualizarea poziției și detectarea coliziunilor cu alte entități din joc.

Clasa Saw definește comportamentul și caracteristicile ferăstrăului din joc. Ea gestionează animațiile și mișcarea ferăstrăului, fiind responsabilă pentru interacțiunea cu alte entități și detectarea coliziunilor.

Algoritmul de coliziune:

Clasa HelpMethods este o clasă utilitară care conține diverse metode pentru gestionarea coliziunilor și detecția acestora într-un joc.

Algoritmul se bazează pe o matrice de date a nivelului jocului (lvlData) care reprezintă harta în formă de grid, unde fiecare celulă din matrice poate fi solidă sau liberă.

Iată o descriere a metodelor din clasa HelpMethods:

CanMoveHere(float x, float y, float width, float height, int[][] lvlData): Metoda verifică dacă un obiect poate fi mutat într-o anumită poziție. Aceasta verifică dacă punctele din colțurile dreptunghiului descris de coordonatele (x, y) și dimensiunile (width, height) nu se ciocnesc de obiecte solide din harta (lvlData).

IsSolid(float x, float y, int[][] lvlData): Metoda verifică dacă un punct specificat (x, y) se află pe o celulă solidă din harta (lvlData). Punctul este convertit în coordonate de celule (tile) și se verifică valoarea corespunzătoare din matricea lvlData. Dacă valoarea este mai mică decât 40 și mai mare sau egală cu 0, înseamnă că celula este solidă.

IsTileSolid(int tileX, int tileY, int[][] lvlData): Metoda verifică dacă o celulă specificată prin coordonatele de celulă (tileX, tileY) din matricea lvlData este solidă. Verifică valoarea corespunzătoare din matrice și returnează true dacă este mai mică decât 40 și mai mare sau egală cu 0.

GetEntityPosNextToWall(Rectangle2D.Float hitBox, float xspeed): Metoda returnează poziția pe axa X a entității după ce a întâlnit un perete. Se calculează poziția pe

axa X a entității și offset-ul pentru a se plasa în mod corespunzător lângă perete, luând în considerare viteza (xspeed) și dimensiunea entității (hitBox).

GetEntityPosUnderRoofOrAboveFloor(Rectangle2D.Float hitBox, float airSpeed): Metoda returnează poziția pe axa Y a entității după ce a întâlnit un acoperiș sau un podea. Se calculează poziția pe axa Y a entității și offset-ul pentru a se plasa în mod corespunzător sub acoperiș sau deasupra podelei, luând în considerare viteza în aer (airSpeed) și dimensiunea entității (hitBox).

IsEntityOnFloor(Rectangle2D.Float hitBox, int[][] lvlData): Metoda verifică dacă o entitate se află pe podea. Verifică dacă nu există obiecte solide sub dreptunghiul de coliziune

Algoritmul de deplasare inamici:

Modul în care inamicul se mișcă în joc este gestionat de mai multe metode și variabile din clasele Enemy și EnemyManager. Voi detalia pas cu pas modul în care inamicul se mișcă folosind codul dat.

Inițializarea și configurarea inamicului:

La crearea unui obiect Enemy se inițializează variabilele și se stabilește dimensiunea hitbox-ului și a cutiei de atac a inamicului. Constructorul clasei Enemy primește coordonatele de poziție (x, y) și inițializează hitbox-ul și cutia de atac cu aceste valori.

Actualizarea mișcării și stării inamicului:

Metoda update(int[][] lvlData, Player player) este responsabilă de actualizarea mișcării și stării inamicului în funcție de datele nivelului și poziția jucătorului. Inițial, se verifică dacă este prima actualizare a inamicului (variabila firstUpdate) și dacă inamicul se află în aer (inAir). Dacă inamicul se află în aer, se actualizează poziția acestuia pe verticală folosind viteza de cădere (fallSpeed) și gravitația (gravity). În caz contrar, se verifică starea curentă a inamicului (enemyState) și se efectuează acțiunile corespunzătoare: Dacă starea este IDLE (inamicul este în repaus), starea este schimbată în WALK (inamicul începe să meargă). În starea WALK, inamicul poate să se miște pe orizontală verificând coliziunile cu pereții și podeaua și schimbând direcția de mers dacă întâlnește o coliziune.

După actualizarea mișcării, se verifică starea inamicului pentru alte acțiuni specifice, cum ar fi atacul (ATTACK) sau rănirea (HURT).

Actualizarea animației:

Metoda updateAnimationTick() este responsabilă de actualizarea indexului de animație al inamicului în funcție de viteza de animație (animationSpeed). Se incrementează animationTick și se verifică dacă a depășit viteza de animație. Dacă animationIndex depășește numărul de imagini de animație disponibile pentru starea și tipul de inamic, se revine la prima imagine și se gestionează schimbarea stării inamicului în cazul în care este necesar.

Schimbarea direcției de mers:

Metoda changeDir() este folosită pentru a schimba direcția de mers a inamicului atunci când întâlnește o coliziune. Dacă inamicul se mișcă în dreapta (right) și întâlnește o coliziune, direcția de mers devine left. Dacă inamicul se mișcă în stânga (left) și întâlnește o coliziune, direcția de mers devine right. Aceasta este o descriere detaliată a modului în care inamicul se mișcă în joc, conform codului furnizat.

Principalele acțiuni sunt gestionate în metoda update(int[][] lvlData, Player player), care actualizează poziția inamicului și starea acestuia în funcție de datele nivelului și interacțiunea cu jucătorul.

Bibliografie

https://sanderfrenken.github.io/Universal-LPC-Spritesheet-Character-Generator/#?body=Body_color_light&head=Human_male_light

https://www.spriters-resource.com/search/?q=jungle

https://www.piskelapp.com/

Top free game assets - itch.io

https://github.com/2DGD-F0TH/2DGD_F0TH