Labquest- A glimpse into biology and chemistry Educational soft

DACIAN TITUS, COZMA AUTOR-1
COLEGIUL NAȚIONAL"VASILE LUCACIU" BAIA MARE
Specializarea: Mate-Info Bilingv engleza cls XII-a A
Email:dacian.cozma@lucaciu.ro

MIHAI BĂLĂNEAN CRISTIAN, AUTOR-2
COLEGIUL NAȚIONAL"VASILE LUCACIU" BAIA MARE
Specializarea: Mate-Info Bilingv engleza cls XII-a A
Email:mihai.balanean@lucaciu.ro

Profesori coordonatori:
PROF. RADU MARGARETA- profesor chimie, Colegiul Național"Vasile Lucaciu" Baia Mare;
PROF. MATIAȘ CARMEN NICOLETA- profesor biologie, Colegiul Național"Vasile Lucaciu" Baia Mare;
PROF. FILIP ADELA VALERIA OANA- profesor informatică, Colegiul Național"Vasile Lucaciu" Baia Mare;

Abstract

The educational software was designed to be used both during chemistry and biology classes and outside of them by high school students, to assess their knowledge of the material taught by the teachers, as well as their gaming abilities. The game was designed to support and enhance the effectiveness of active and interactive knowledge acquisition, training, and development of various competencies and skills. The information has been structured in a logical manner, corresponding to the designed learning strategy. The language employed aligns with the student's level of understanding and is suitable for the content

Keywords: *educational software, chemestry, biology .*

Introducere

Chimia și biologia sunt considerate de mulți elevi ca fiind subiecte „dificile", deoarece majoritatea dintre ei au adesea dificultăți în înțelegerea acestor concepte. Nu toți elevii pot înțelege mental anumite situații chimice sau termeni de biologie, iar atunci rezultatele lor sunt modeste.

Ideea de a concepe un joc care să permită și să stimuleze exploatarea multilaterală a potențialului intelectual, care să creeze premisele dezvoltării independenței, a spiritului de inițiativă și a stilului propriu de gândire, a venit din dorința de a îmbina subiectul pe care l-am îndrăgit amândoi încă de mici, și anume informatica, cu materii mai putin plăcute nouă: chimia și biologia.

Plecând de la ideea „Vrem să facem un joc care…" ne-am dat frâu liber imaginației și ne-am gândit la: ce vor face jucătorii pentru a câștiga? (se vor lupta cu roboți, își vor încărca viața răspunzând la întrebări de chimie și biologie), care va fi povestea jocului? („luptă" plus „vrei să fii miliardar?"), care vor fi personajele? („om de știință nebun", soldati, roboți), cum să realizăm grafica?, apoi am petrecut sute de ore să programăm codurile astfel încât că jocul să prinde viață.

Secțiunea preliminară

Aplicația: Labquest a glimpse into biology and chemistry își procură toate obiectele necesare din interiorul fișierelor de joc acestea conținând o multitudine obiecte necesare pentru rularea cu succes a jocului precum:

Caracter:

- Folderul care conține toate animațiile caracterului si toate variațiile de culoare si de armură ale acestuia.

- Fiecare acțiune are un număr specific de animații, spre exemplu animația de alergare are 6 animații, în timp ce ceea în care tragi are 7 animații.

Iteme:

- Folderul principal care conține toate obiectele cu care caracterul se va întâlni în timpul jocului
- Fiecare item este notat cu un număr în interiorul fișierului, fiecare obiect fiindu-i atribut ulterior o anumită acțiune în timpul codului

-Fiecare item are deasemenea o descriere care poate fi găsită în interiorul fișierului "DescriptieObiecteInventar"

Robot:

- Folderul care conține toți inamici si toate variațiile de culoare ale acestora și de asemenea ca și în cazul caracterului fiecare acțiune are un număr specific de animații, spre exemplu atacul melee al robotului are 4 animații.

LevelSprite

-Folder care conține un sprite al texturilor care va fi prelucrat din interiorul codului si un PNG de culori care va fi interpretat în interiorul codului.

Texturi

Un folder care conține tot restul texturi care vor fi utilizate în timpul nivelului cum ar fi backgroundul arenei de combat sau backgroundul din timpul quizului, bara de viața, energie a caracterului si a inamicilor.

Carte

- Fișierul care va conține informația care se află în interiorul cărților din joc.
- Conține subfoldere cu informația care se află în fiecare carte aceasta fiecare pagina fiind notată cu un număr.

-Conține fișiere cu animațiile de dare a paginii pentru cărțiile respective

Fișiere importante

- fisiere din care să citim informația necesară pentru rularea programului precum ar fi:
- fișierul cu descrierea itemelor din inventar
- fișierul cu descrierea abilităților din skill tree

-fișierul de conexiuni ale abilitătiilor

Alcăturiea aplicației

Aplicația este impărțită in mai multe pachete pentru a pastra ordinea în interiorul codului, printre aceste pachete se numără:

- "actiune_joc" de unde schimbăm starea în care jocul se află spre exemplu, inițial jucătorul se afle in meniu, dar dacă apasam butonul de play atunci el va trece în stadiul de playing si va putea juca jocul. Pachetul conține următoarele clase:
  1. "Carte" clasă din interiorul căreia caracterul va putea citi teoria necesară pentruraspunderea la întrebări. Informația din cărți este citită din fișiere fiind memorată in Vectori de Imagini pentru eficientizarea din punct de vedere a timpului.
  2. Playing" clasa principală de unde caracterul poate accesa toate celelalte clase, clasă care conține inamicii si pozitiile acestora, obiectele, checkpointurile,etc.
  3. "Inventar" clasă unde prin intermediul unui vector de frecventă sunt reținute obiectele pe care caracterul deja le-a colectat in timpul nivelului.
  4. "Intrebari" clasă care poate fi activată doar în combat și unde jucătorul va putea răspunde corect la întrebări.

5. SkillTree" clasă care conține abilitătiile jucatorului

- "main_game", pachet care conține inamicii, caracterul, frameul, threadul principal de unde să putem edita updateurile pe secunda si frameurile pe secundă, bările principale de viață și de energie ale caracterului, checkpointurile de unde caracterul să poată salva si bara de viață a inamicului
- "nivel", pachet care conține un interpretator al fiecărui bloc de 64x 64 din folderul de spriteuri si pachet de unde de asemenea desenăm nivelul și initializăm atât inamicii din nivel cât si obiectele si caracterul.
- "stats", pachet care conține atributele curente ale caracterului și ale inamicului.
- "menu", pachet care conține toate meniurile care pot fi întalnite în timpul jocului precum:
  1. Meniul de start de unde jucătorul porneste jocul
  2. Meniu de pauză care conține posibilitatea de reîncepere de la ultimul checkpoint și de asemenea butoane care îți permit să revii la meniul inițial sau să părăsești aplicația
  3. Un sistem de salvare care reține intr-un fisier doar statisticile importante ale caracterelui
  4. Un sistem de încărcare care reinitializează jocul la momentul salvării
- Un pachet de utilitare care să ne permită să pastram ordinea codului, aceasta conține:
  - Constantele obiectelor din inventar;
  - O funcție care prelucreaza obiectele din inventar;
  - Constantele animatiilor caracterului și inamicului;
  - O functie care returneaza numărul de animații și viteza acestora;
  - Constante abilitătiilor din skilltree;
  - O funcție care prelucrează toate abilitatiile;

Aplicația are mai multe stadii de joc:

1.Stadiul Playing

Este stadiul principal din care toate celelalte stadii pot fi accesate de jucător: nivelul, animațiile caracterului, animațiile roboților,obiectele care se află în joc.

CheckpPoint- Acestea sunt inițializate din interiorul funcției "add_check_points", un checkpoint având aceeași structură ca și cea a unui "Rectangle", mai precis coordonatele x,y si lungimea și lățimea.

Clasa CheckPoint mai are de asemenea incă o funcție de verficare în range aceasta primind ca și parametrii Rectangle cu hitboxul caracterului. Dacă caracterul se intersectează cu unul dintre checkpointuri vom crea un nou custom save file care va reține poziția caracterului și a inamicilor, obiectele rămase, inamicii rămași, culoarea armurii caracterului, statisticile caracterului

Procesul de salvare și de reîncărcare a claselor se numește serializarea clasei, respectiv deserializarea clasei.

Obiecte

O clasă ajutătoare care conține coordonatele fiecărui obiect cu care caracterul poate interacționa și de asemenea tipul obiectului

La fel ca și în cazul clasei CheckPoint conține o funcție de verificare în range care primește ca și parametrii Rectangle cu hitboxul caracterului.

Daca caracterul se intersectează cu unul dintre obiecte acesta este eliminat din vectorul de obiecte colectabile și este adăugat în inventarul jocului

Caracter

1.Clasă care gestionează toate acțiunile caracterului. In această clasa sunt încărcate animațiile caracterului, folosind o mapă vom reține numele fiecărui folder al animațiilor pentru a putea inițializa toate animațiile caracterului

Din interiorul funției „setSpriteAnimation” care primește ca și parametrii indicele culorii armurii, path_file si matricea de animații în care sunt incărcate fiecare animație în matricea de animație care este de forma PlayerAnim[culoare][animație][număr].

Funcția setSpriteAnimation” de asemenea modifică culoarea armurii prin intermediul unei funcții care analizează fiecare pixel al armurii, găsește un set de pixeli custom, iar deoarece nu fiecare pixel din imagine care alcătuiesc armura caracterului va verifica de asemenea pentru pixeli care diferă foarte puțin din punct de vedere ar RGB, maxim diferențe de 20 în modul între valoarea inițială a oricărei componetă fie aceasta rosu, galben sau albastru și ii va inlocui cu o valoare prestabilită.

Funcția UpdateTick:-updatează animația curentă a caracterului, aceasta fiind păstrată în variabila player_action. Daca animația este JUMP atunci funcția nu va fi apelata, anmatia de sărit depinzând de viteza curentă a caracterului pe axa de coordonate y. funcția conține o variabilă numită „tick” care se updatează de un număr egal cu updateurile pe secundă. dacă variabila este mai mare decât viteza curentă a actiunii, se mărește numarul animației. daca numarul de animații curente este mai mare decat numărul de animatii ale acțiunii.,atunci numărul de animații curente se va reseta la 1.

Funcția CanMoveHere- verifică dacă hitboxul caracterului se intersectează cu una dintre texturi

Hitboxul caracterului este centrat și nu este mai mare decât un bloc de 64x64 pixeli. Acesta verifică daca caracterul intersectează una dintre texturi într-un loc și în caz afirmativ returneaza false

Texturile jocului au în general 64 x 64 pixeli, cu câteva excepții minore acestea fiind luate din funcția GetTileHitbox, fiecare având un număr ca și identificator general

Deoarece unele blocuri nu sunt solide si poți trece prin ele vom apela o altă funcție care să verifice dacă unul dintre acele blocuri sunt solide sau nu, în caz negativ acele texturi nu au hitbox.

Funcția updatePos- verifica initial daca asupra caracterului se intâmplă vreo actiune, in caz negativ apelul funcției se termină. Dacă caracterul se mișcă stânga sau dreapta si poziția respectiva nu îl face să se intersecteze cu vreun obiect caracterul se va deplasa. În cazul unei coliziuni cu alte texturi vom folosi funcția “CanMoveHere” din pachetul utilz, clasa Help_Methods. Dacă caracterul execută o săritură atunci ii vom atribui o viteza de săritură care va crește gradual, pînă la apariția forțelelor gravitaționale destul de mari care impun coborârea la sol. Dacă playerul este în aer și are o viteză care să îi împună căderea atunci acesta va coborâ până ce se va intersecta cu o textură. La finalul funcției de updatePos vom seta din nou animația playerului în funcție de acțiunea pe care o face în acel moment, in cazul în care acțiunea trecută nu coincide cu cea prezentă, vom reseta animațiile caracterului

CheckCloseToBorder- Funcție care bazată pe poziția caracterului mută nivelul vizibil pe ecranul jucătorului desenând texturile care ar depăși lungimea și latimea unui ecran.

Daca poziția caracterului pe axa x sau y depășește o bornă prestabilita (in cazul nostru 60% din dimensiunile nivelului, atunci vom muta ecranul jocul la stânga sau în sus), iar dacă este mai mic decât 40% din nivel vom muta nivelul spre dreapta sau în jos.

Enemies

1. Clasă care gestionează toate acțiunile inamicului.

2. La fel ca și în cazul caracterului folosim o mapă ca să incărcăm animațiile din foldere și le initializăm intr-un vector numit „EnemiesAnim”.

3. Constructorul clasei are ca și parametrii coordonatele x si y, tipul inamicului, numărul de inamici, și clasa principala.

4. La fel ca și în cazul caracterului are o funcție care ii updatează animația, care verificai daca poate ajunge in poziția respectiva și daca plutește și trebuie adus la sol.

5.Adițional mai are o funcție care se ocupă cu verificarea faptului că inamicul este in raza de acțiune a caracterului, iar în caz afirmativ initializează stadiul Combat.

LevelManager

Clasă de unde sunt desenate și randate graficele. Fiecare instanță creată a unei clase primește ca parametrii un pachet de texturi, reprezentat prin minitexturi de 64x64 pixelil, acesteac fiind de tipurile solide sau prin care poți trece. Pentru construirea nivelului vom folosi un PNG pe care îl vom importa din fișierele jocului, din nivelul pe care dorim să il construim și pe care îl vom manipula convenabil în funcție de valorile culorilor rosu,verde si albastru din modelul RGB. Pentru a interpreta ce texturi dorim să desenăm într-o poziție x,y vom segmenta jocul sub forma unor tile-uri de 64 x 64 de pixeli carora le vom atribui o cheie de identificare de forma (i,j) si o valoare care să corespundă texturi pe care dorim să o desenăm și randăm în locul respectiv.Pentru manipularea texturilor ne vom folosi de valoarea roșu din modelul RGB, valoarea acesteia determinând ce textură vom folosi în tile-ul caruia îi corespunde cheia (i,j). Pentru interpretarea inamicilor vom folosi o valoare convenabilă pentru valoarea verde și ne vom folosi de valoarea lui albastră ca să stabilim tipul obiectului. Același procedeu îl vom folosi și în cazul obiectelor pentru a le interpreta și a le putea adăuga in vectorii de obiecte și ai inamicilor.

2. Stadiul Combat

Stadiul principal unde se petrece combatul din joc, fiind impărțit în mai multe clase:

Abilități- Clasa de unde pot fi activate abilitățile caracterului

În momentul în care jucătorul apasă click în interiorul unei abilități jocul va verifica dacă caracterul are energia necesară pentru utilizarea abilității și dacă abilitatea este in cooldown sau nu. Pentru verificarea tuturor acestor proprietăți vom apela funcția “based_on_ability_pressed” care primește ca parametrii indicele butonului apăsat și butonul respectiv, luând numele abilității și verificând condițiile în fiecare caz

În momentul în care caracterul intră în interiorul butonului unei abilități pe ecran îi va apărea textul cu descrierea abilității. Aceste lucru se realizează din interiorul funcției “setButtonText” care are ca parametrii un string care reprezintă numele butonului respectiv

Objects_Combat- Clasa ajutătoare care are rolul de a fi pasată tuturor claselor care gestionează acțiunile referitoare la combat

Clasa este alcătuită dintr-o multitudine de constante care au scopul de a putea fi accesate și de clasele care moștenesc această clasa, precum constantele caracterului și a robotului din timpul combatului, imaginile proiectilelor din combat, coordonatele finale ale locului de lansare a gloantelor, pozițiile finale ale caracterului și ale robotului, etc

Fight_Enemy- Clasă care gestionează toate acțiunile robotului

Atacuri roboților sunt împărțite în mai multe funcții

Robotul are 2 atacuri principale:

Atacul 1- Are ca parametrii de intrare clasa principală de combat, caracterul și inamicul care atacă momentan. Este un atac de la apropiere astfel are nevoie de un punct de oprire

Punctul de oprire este gestionat de funcția „update_enemy_poz” din pachetul “utilz”, Clasa „Help_Methods” care are ca parametrii poziția inamicului, pozitia caracterului și vitezele pe axele x si y are robotului și care aproprie inamicul de hitboxul caracterului

În momentul în care robotul și inamicul se intersectează începe animația de atac a robotului, stadiu care se termină în momentul în care animația de atac se resetează.

După ce atacul se finalizează robotul caracterul primeste un efect de knock_back, iar robotul revine la poziția inițială, atacul sfârșindu-se în momentul în care inamicul și caracterul se află în pozițiile lor inițiale, toate aceste evenimente fiind gestionate de funcția „turn_back” din interiorul clasei.

Atacul 2 - Reprezintă un atac de la distanță, glontul fiind diferit depinzând de poziția inamicului. În funcție de atacator, glontul are de asemenea viteze diferite, acesta traverând aerul până în momentul în care se intersectează cu hitboxul caracterului

Dacă caracterul nu este mort la finalul atacului, acesta își ia knock_back, iar în momentul în care se află în poziția inițială combatul continua cu urmatorul atacator.

Cum numărul de inamici este mai mic decât 4, dacă tura este egală cu 4 atunci va rezulta că tura următoare este a caracterului

Fight_Player- Similar cu clasa care gestionează lupta inamicului aceasta este împărțită în mai multe atacuri, toate fiind similare cu atacul 2 al robotului, diferența majoră fiind în cazul atacului cu grenada care atacă toți inamicii

La fel ca și în cazul atacului 2 al robotului, proiectilul traversează aerul până intr-un punct prestabilit acesta fiind în cazul atacurilor cu laser până când se intersectează cu hitboxul inamicului atacat, respectiv în cazul grenadei până ce ajunge în centrul inamicilor.

Funcția update_hitbox_glont face ca glonțul sa traverseze ecranul până în momentul în care ajunge la coordonatele finale, fie acestea x_final și y_final. Dacă glonțul ar depăși aceste coordonate acesta ar lua exact acele 2 coordonate și ar dispărea.

Odată ce proiectilul se intersectează și toate caracterele sunt în pozițiile lor inițiale jocul verifică dacă mai există vreun inamic încă în viață, în caz afirmativ tura se mută pe acesta, altfel combatul se sfârșește.

Phase_1 - Fază de tranziție a jocului, în această faza inamicii și caracterul sunt tractați în OZN, acesta dispărând doar în momentul în care atât adversarul cât și caracterul au fost tractați.

Tractarea se petrece doar când ozn intră în raza de acțiune a unuia dintre cei doi, lucru care se poate petrece doar după ce OZN ajunge la ultima animație disponibilă

După ce atât inamicul cât și caracterul au fost tractați OZN-ul dispare de pe ecran, iar în locul lui apare un loading screen care încărcă lumea pentru modul: combat.

Odată ajunși în această lume începe procesul de așezare a acestora în pozițiile lor finale, care deja au fost prestabilite înainte de combat .

Funcția care gestionează tractarea caracterului la sol este reprezentata de “update_carcater_in_poz” care are prioritate față “update_enemies_in_poz” și “update_ozn_in_poz” care mută toate aceste obiecte în pozițiile deja prestabilite .Odată ce caracterul se află în poziție începe procesul de așezare a OZN în poziție, iar mai apoi a inamicilor, care spre deosebire de primele 2 se despart în trei poziții finale, la fel prestabilite.

Odată ce toate elementele se află în pozițiile corespunzătoare, abilitățile sunt activate, iar combatul poate începe, acesta fiind gestionat din interiorul clasei „Phase_Combat”.

Phase_Combat- Clasa principală de unde sunt gestionate toate atacurile și acțiunile din timpul combatului. De aici sunt desenate toate elementele combatului, abilitățile pe ecran, inamicii, arena de luptă. În funcție de inamicul și de atacul ales vor fi desenate pe ecran acțiunile corespunzătoare. Fiecare inamic are câte un vector de atacuri generat aleatoriu în momentul începerii combatului, acestea fiind generate de funcțiile din interiorul clasei Random în interiorul constructorului inamicului.

Fiecare abilitate care poate fi folosită are o energie proprie, acestea putând fi utilizate doar dacă caracterul mai dispune de energia necesară care să le folosească.

Atacul simplu este atacul de bază unde proiectilul utilizat are forma unui bloc de gheață, atacul concentrat dă impresia unui țurțure alungit din gheață, atacul grenadă, în momentul exploziei lansează un nor de pumni care simulează animațiile utilizate în desenele pentru copii.

Butonul de așteptare care sare peste tura caracterului are rolul de a reîmprospăta energia.

Butoanele de poțiuni care pot fi colectate în timpul jocului și care au abilitatea de a regenera vitalitatea și energia caracterului mai pot fi numite și regenerare instantanee.

Butonul de recharge tranziționează jocul la stadiul de întrebări de unde jucătorul în urma unui răspuns bun poate să își reîncarce parțial bateria .

Atacurile inamicilor sunt reprezentate de un atac de apropiere care diferă în funcție de inamic, combinat cu un atac de la distanță sub forma unei mase gelatinoase.

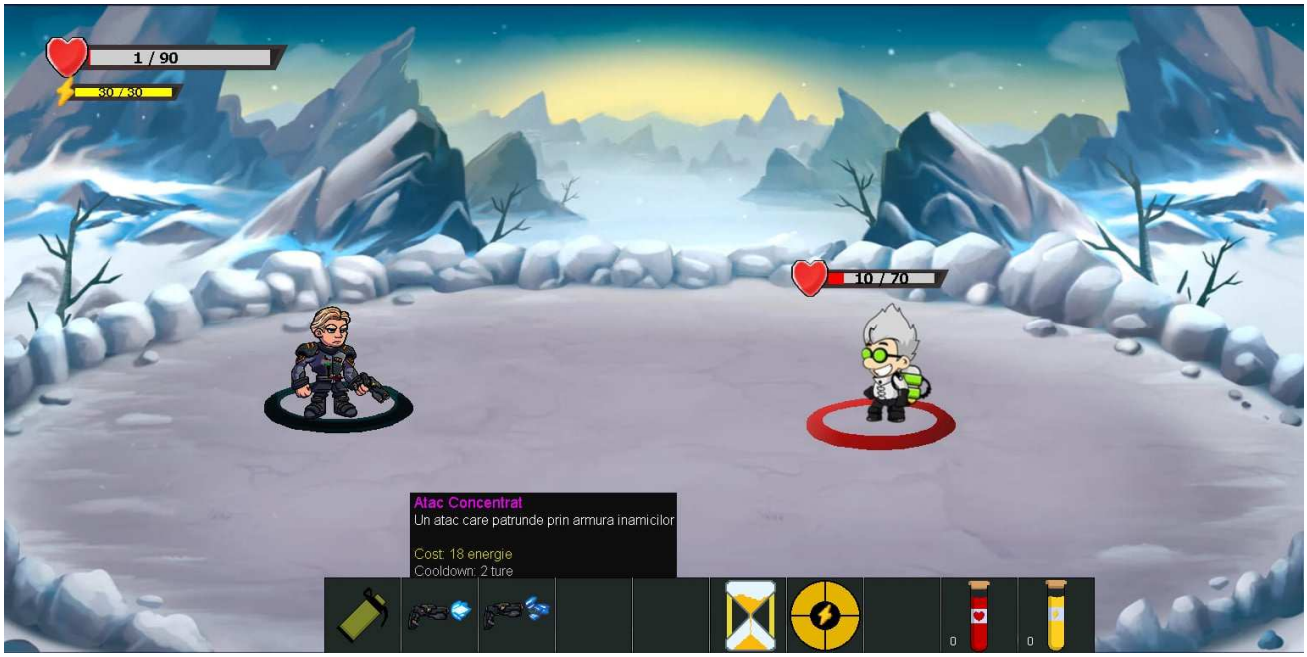

Fig 1 Interfața combat din jocul: Labquest a glimpse into biology and chemistry
Cozma Dacian si Mihai Bălănean

Stadiul Întrebări- Stadiu care poate fi accesat doar din interiorul combatului, caracterul având oportunitatea să își refacă energia dacă răspunde corect la o întrebare de chimie sau biologie.

Întrebările sunt încărcate dintr-un fișier de tip ".txt" având forma Întrebare și 4 răspunsuri.

Pentru a determina dacă un răspuns este corect vom folosi indicatorul * pentru a delimita un răspuns corect. Dacă * se află la începutul întrebării atunci acesta este unul corect.

Pentru a verifica dacă un răspuns este corect vom folosi o structură de tip queue (LinkedList) care să ne permită să reținem răspunsurile selectate. În momentul în care jucătorul apasă în interiorul unui răspuns atunci, dacă acesta a fost deja apăsat îl vom deselecta, astfel eliminându-l din queue, altfel îl vom selecta, astfel adăugîndu-l in queue.

Odată ce jucătorul apasă pe butonul de send răspunsurile lui din queue sunt verificate, iar dacă toate sunt corecte atunci energia acestuia va fi reîncărcată.

Pentru desenarea lor pe ecran vom proceda similar cu procedura urmată în cazul textului obiectelor, singura diferență fiind reprezentată de alinierea care în cazul nostru este de la centru. Pentru a putea alinia un text la centru vom calcula pentru fiecare rând numărul de cuvinte care încap pe rând și spațiul pe care îl ocupă, iar mai apoi în funcție de spațiul ocupat vom calcula diferența dintre lungimea rândului și spațiul ocupat, iar apoi această valoare o vom împărți la 2 pentru a afla exact poziția de start de la care ar trebui să înceapă primul cuvânt de pe fiecare rând. Cum textul va putea avea maxim trei linii vom calcula de asemenea poziția de start a coordonatei y în funcție de numărul de linii pe care îl vom avea.

După apasarea butonului de trimitere, jucătorului i se dă posibilitatea de a vedea explicații suplimentare la fiecare răspuns în parte, astfel încât să înțeleagă pe deplin dacă a existat o greșeală sau nu în raționamentul său.

Explicațiile suplimentare sunt încărcate individual pentru fiecare întrebare în parte din fișiere de tip text, și sunt încărcate în joc într-un ArrayList numit "Explicații".

Întrebările pot fi editate printr-o aplicație software, astfel încât profesorul să poată introduce întrebări noi, adaptate nivelului de cunoștințe a clasei la care predă, întrebăriloe fiind transformate prin intermediul acestei aplicații în întrebări care să respecte formatul de importare al acestora în joc.

Editorul de întrebări este realizat în Windows Forms, permitând un acces ușor la elementele de interfață. Din punct de vedere al funcționalității, editorul poate accesa un fișier de întrebări odată. Prin fișier de întrebări, ne referim la modul de stocare al întrebărilor de tip grilă. Când programul este deschis, va folosi primul argument dat pentru a determina calea fișierului de întrebări și le va încărca în editor. Fiecare întrebare are un text corespunzător întrebării, 4 texte ce corespund fiecărei variante de răspuns și 4 casete check box, ce semnalează validitatea fiecărei variante. Toate acestea pot fi editate prin text box-urile și check box-urile aferente. Editorul mai are funcționalitatea de a adăuga sau elimina întrebări, pe lângă o funcție de salvare a întrebărilor, urmată de posibilitatea de a vedea un preview al cum s-ar vedea întrebarea în joc.

Stadiul Inventar- Locul unde obiectele colectate în timpul stadiului de playing sunt reținute

Obiectele sunt incărcate din fisierele jocului fiecare având câte o cheie identificator care să ne permită să îi gestionăm mult mai ușor acțiunile, incărcarea din foldere realizându-se în funcția numită „ObjectIndex”. Pentru a adăuga un obiect în inventar vom crea o matrice care să rețină obiectul pe care vrem să îl adăugăm indexarea obiectelor începând de la 0, această matrice având numele de lvlData.

Pentru a desena inventarul ne vom folosi de funcția din pachetul „java.awt.Graphics” care să ne permită să desenăm dreptunghiurile de 80x80 în care vor apărea obiectele, fiecărui dreptunghi fiindu-i atribuit un indice de coloană și rând. De asemenea vom desena caracterul cu armura pe care o are în acel moment, backgroundul inventarului și obiectele dacă există.

Deoarece fiecare obiect are câte o descriere, pe această o vom importa din fișierele jocului și o vom desena într-un dreptunghi folosind funcția drawsString și asigurându-ne ca nu depășește perimetrul asociat folosindu-ne de o funcție “getFontMetrics” care ne va returna numărul de pixeli pe care îl va ocupa pe ecran.

Dacă cuvintele depășesc lungimea liniei dreptunghiului în care vrem să le desenăm vom trece pe o nouă linie mărind coordonata de pe axa y cu lungimea fontului.

În această funcție care inițial setează culoarea fontului și fontul, apoi împarte cuvântul în Stringuri mai mici în funcție de existența separatorului spațiu între ele, iar mai apoi pentru fiecare cuvânt îi calculează lungimea în pixeli, iar mai apoi dacă depășește lungimea maximă admisă îl desenează pe rândul de mai jos, altfel îl adaugă lungimea variabilei „distantax” și continua să deseneze pe rândul respectiv.

Pentru a gestiona efectul fiecărui obiect din inventar am creat o funcție în pachetul “utilz” in clasa “Constante” numită “GetAction” care pentru fiecare obiect din inventor îî aplică efectul dorit. Pentru a reține ce obiecte am colectat vom folosi un vector de frecvențe care să rețină numărul de apariții în inventar al fiecărui lucru cules de caracter. Pentru a adăuga un obiect vom apela funcția Add_Items care are ca și parametru de intrare cheia obiectului pe care vrem să îl adăugăm, aceasta având scopul de a verifica dacă obiectul este deja în inventar, în caz afirmativ mărim frecvența lui, altfel apelăm funția “AddObject” care adaugă obiectul în primul loc din inventor disponibil.

Pentru eliminarea unui obiect odată ce a fost consumat ne vom folosi de funcția removeObject care are ca parametrii indicii de coloană ai obiectului din inventor. Acesta verifică dacă obiectul apare o singură dată în vectorul de frecvența, iar dacă da permută toate obiectele cu o coloană, iar dacă este cazul chiar cu un rând și o coloană, în cazul în care avem mai mult de un rând de obiecte. Un alt caz pe care trebuie să il gestionăm este dacă obiectul este sau nu unul care poate fi consumat, deoarece spre exemplu nu ne-am dori ca armura sau cărțile să ne dispară din inventar astfel dacă este unul dintre acele obiecte funcția nu se va executa.

Stadiul Carte- Locul de unde caracterul poate accesa informația din joc legată de datele chimice.

Cărțile pot fi colectate pe parcursul jocului doar în stadiul Playing. Informația din cărți este stocată in vectori de imagini care diferă în funcție de fiecare carte. Animația cărți este prelucrată din poza cu extensia PNG "Carti" fiecare carte având aceeași dimensiune prestabilită.

Cititorul are posibilitatea să dea paginile înainte și înapoi acestea, odată ce sunt încărcate din Folderul cărți, având o animație de dare a paginii.

Pe ecran sunt desenate maximum doua pagini la orice timp. Pentru a da pagina la dreapta animația cărții va începe de la prima animație în timp ce pentru a da pagina la stânga animația va începe de la ultima pagină Pentru a face mai veridic efectul de animație vom desena în continuare în timpul animației un număr de pagini bazat în funcție de indicele de redare al animației cărți.

Pentru ca profesorul să poată edita cărțile, am creat o funcție, folosindu-ne de libraria "apache.poi" care se ocupă de interacțiunea cu fișierele de tip powerpoint prin care să transformăm slide-urile unei prezentări în poze pe care le vom salva în fișierele jocului și de unde le vom încărca ca și informația din cărți.

Concluzii

Proiectarea unui aplicații de tip soft educațional depinde de o serie de factori, precum ar fi: grafica, elementele de pedagogie, animațiile incluse, lucruri care pot influenţa experienţa utilizatorilor într-un mod plăcut sau nu.

Prin modul în care este realizat acest soft, profesorul poate modifica dinamic softul educaţional, adaptându-l continuu, în funcţie de capacităţile cognitive ale grupei de nivel şi de evoluţia colectivului de elev, lucru care este de real ajutor oricărui cadru didactic implicat, care dorește să schimbe ceva în modul de predare.

Prin intermediul jocului propus, elevul se autoevaluează continuu şi ia decizia traseului de instruire cel mai potrivit, fiind astfel responsabil pentru propria învăţare, prin intermediul jocului el poate aprofunda informații suplimentare care îl ajută să înțeleagă mai bine explicațiile de la răspunsurile bune date.

În cazul în care răspunsurile date nu sunt cele bune, greşelile nu sunt considerate eşecuri ci încercări şi nu sunt penalizate, iar decizia schimbării parcursului instruirii este la liberul arbitru.

Rezultatele obţinute în urma utilizării aplicației propuse de noi, ne îndreptăţesc să afirmăm că utilizarea acestui software educaţional pentru instruirea diferenţiată individuală a elevilor, poate și va avea un impact pozitiv semnificativ faţă de strategiile clasice de predare-învăţare-evaluare. Vorbind din proprie experiență acest soft poate determina un progres semnificativ al elevilor, în ceea ce priveşte eficienţa activităţii de achiziţie activă şi interactivă a cunoştinţelor, a formării şi dezvoltării unor competenţe și abilități , dar și a interesului elevilor pentru studiu chimiei și biologiei, două materii care nu sunt neapărat îndrăgite de toți.

Bibliografie

MouseListener

[1] https://docs.oracle.com/javase%2F7%2Fdocs%2Fapi%2F%2F/java/awt/Component.html#addMouseListener(java.awt.event.MouseListener) accesat la data de 13.01.2024;

Class Panel https://docs.oracle.com/javase%2F7%2Fdocs%2Fapi%2F%2F/java/awt/Panel.html accesat

[2] la data de 12.04.2024

Interface Serializable https://docs.oracle.com/javase/8/docs/api/java/io/Serializable.html accesat la data [3] de 3.04.2024;

Inputs Platformer Game Tutorial *Java* Accesat: la data de 12.03.2024 HYPERLINK

[4] "https://www.youtube.com/watch?v=6Tj6XYGWfko&list=PL4rzdwizLaxYmltJQRjq18a9gsSyEQQ-0&index=2" https://www.youtube.com/watch?v=6Tj6XYGWfko&list=PL4rzdwizLaxYmltJQRjq18a9gsSyEQQ-0&index=2

[5] ***Game Loop Platformer Game Tutorial*** *Java* Accesat: la data de 19.11.2023 HYPERLINK https://www.youtube.com/watch?v=aFS9Whsoecc&list=PL4rzdwizLaxYmltJQRjq18a9gsSyEQQ-0&index=3