ORIZONTURI CENUSII

DOCUMENTAȚIE PENTRU CONCURSUL NAȚIONAL DE SOFT „GRIGORE MOISIL”

LUGOJ 2015

CONȚINUTUL DOCUMENTAȚIEI

PREZENTAREA JOCULUI

„**Orizonturi cenușii**” prezintă o realitate crudă, dată de existența perpetuă a războaielor de-a lungul istoriei întregii omeniri. Intenția acestui joc nu este de a promova războiul, chiar din contră. Prin intermediul efectelor vizuale, a situației prezentate în joc și a deducțiilor jucătorului se realizează o atitudine negativă față de ororile produse de către război.

Jocul este unul bidimensional, dar ce folosește grafică modernă, sunete reale și diverse alte elemente pentru a oferi o notă de realism. În mod normal, astfel de jocuri se pot limita la o anumită grilă de poziționare, ce împiedică fluiditatea în mișcare sau a efectelor grafice. „*Orizonturi cenușii*" folosește o grilă la nivel de pixel, ce permite o fluiditate maximală în cele două dimensiuni.

GENEZA

„*Orizonturi cenușii*” a început drept un joc anonim ce își propunea să reproducă un joc de arcadă extrem de popular în anii ’80, intitulat „*Battle City*”. Acesta urma să fie o demonstrație practică a implementării programării orientate pe obiecte într-un mod cât mai pur, și ușor de interpretat.

În timp ce proiectul prindea amploare, s-a ajuns în mod natural la concluzia că ar fi mult mai indicat extinderea respectivei idei promovate de jocul de arcadă și aducerea lui la standarde moderne.

Astfel s-a născut „*Orizonturi cenușii*”, joc ce avea să fie supus unui număr extrem de mare de modificări pe parcursul dezvoltării, și evoluării naturale, a acestuia.

GAMEPLAY

Scopul jocului este dat de către misiuni, ce constituie firul poveștii al acestui joc. Fiecare misiune este formată din diverse obiective, ce antrenează jucătorul și îl pune în situații ce necesită concentrare, experiență și o strategie bine definită.

La începutul fiecărui nivel, jucătorul este reprezentat de un soldat. Acesta poate fi controlat folosind tastele standard (WASD, săgeți), rotirea realizându-se cu ajutorul mouse-ului; jucătorul are posibilitatea de a se folosi de anumite vehicule precum mașini (un bun exemplu este mașina ce poate plasa mine), tancuri (principalul element al acestui joc), etc.

Scopul jocului este acela de a duce la bun sfârșit obiectivele impuse de către fiecare misiune la care participă jucătorul.

ELEMENTE TEHNICE CARACTERISTICE

Câmpul de luptă (denumit ulterior „hartă”) permite o dimensiune care se poate extinde în afara rezoluției jucătorului. Harta poate avea, teoretic, o dimensiune infinită (limitată doar de hardware). Harta poate conține un număr nelimitat de obiecte (precum mașini, tancuri, soldați, ziduri, etc.), atât timp cât procesorul (actualizarea stării obiectelor poate fi intensivă) și memoria permit.

Orice obiect aparținând hărții poate avea o dimensiune arbitrară, independentă de celelalte. De asemenea, poziționarea se face astfel încât jucătorul are parte de o experiență cât mai fină. Exista, de asemenea, și posibilitatea unui obiect de a se roti liber, de la 0° până la 360° în planul bidimensional în care se află.

TEHNOLOGIA FOLOSITĂ

Pentru a fi posibilă realizarea acestui joc, au fost folosite un număr de tehnologii. Urmează o listă a acestora, împreună cu o scurtă descriere a lor:

I. **Limbajul de programare** — „Orizonturi cenușii" a fost realizat în limbajul de programare **Microsoft C# 5.0**; acesta este un puternic și bine-cunoscut limbaj ce permite utilizarea paradigmei programării orientate pe obiecte (engl. OOP) implementată cu succes în dezvoltarea jocului. Limbajul C# se bazează pe tehnologia **Microsoft .NET Framework**, concepută cu scopul de a oferi posibilitate de a crea software la un nivel înalt (cât mai abstract de implementarea propriu-zisă) și o mare portabilitate între sisteme de operare și diverse sisteme de operare.

II. Versiunea de *Microsoft .NET Framework* utilizată aici este varianta portabilă menționată anterior, **Mono 4.0**. Implementarea multiplatformă *Mono* este compatibilă cu cea canonică, rulând doar pe Windows, a Microsoft-ului. Acest fapt, împreună cu practicile de portabilitate adoptate, i-ar permite teoretic jocului să ruleze pe o varietate de dispozitive precum Android, Windows 8 (Modern UI), Windows Phone, iOS, Xbox, OUYA, etc.

III. Drept platformă de desenare grafică a fost aleasă biblioteca **MonoGame**; aceasta este o reimplementare open-source și portabilă a *Microsoft Xna Framework*. MonoGame facilitează desenarea grafică atât în plan bidimensional, cât și tridimensional, oferind de asemenea și capabilități de intrare/ieșire. Principala funcționalitate de desenare folosită aici este cea pur 2-dimensională, bazată pe imagini-componente (engl. sprites), dată de clasa `SpriteBatch`.

IV. Pentru administrarea automată a desenării diverselor stări ale jocului (meniu principal, setări, jocul propriu-zis, interfața Heads-Up Display) s-a folosit `GameStateManager`; un set de trei clase create de către Microsoft pentru a servi drept un exemplu complet funcțional a stărilor unui joc.

V. Deoarece geometria implicată în poziționarea absolută, a dimensiunilor arbitrare și a rotirii libere a obiectelor aflate pe câmpul de luptă este suficient de dificilă, a fost folosită o bibliotecă pentru `MonoGame` denumită `RotatedRectangle`; aceasta implementează clasele ce reprezintă un dreptunghi rotit în radiani, împreună cu metoda de verificare a intersecției între două astfel de dreptunghiuri.

VI. Din punct de vedere al inteligenței artificiale, determinarea celei mai potrivite căi pentru a ajunge la un punct oarecare pe hartă devine de o complexitate ridicată. În acest caz, cea mai eficientă metodă este de a folosi poligoane de navigare (engl. Navigation Meshes). Această tehnologie este de obicei folosită în medii tridimensionale, dar poate funcționa și în medii bidimensionale (coordonata Z este zero). Biblioteca ce permite un astfel de comportament se intitulează `SharpNav`.

STRUCTURA INTERNĂ A PROIECTULUI

„SOLUȚIA” ȘI PROIECTELE AFERENTE

Jocul, ca întreg, este reprezentat de către un set de proiecte C# (engl. solution). Proiectele aferente acestui set sunt:

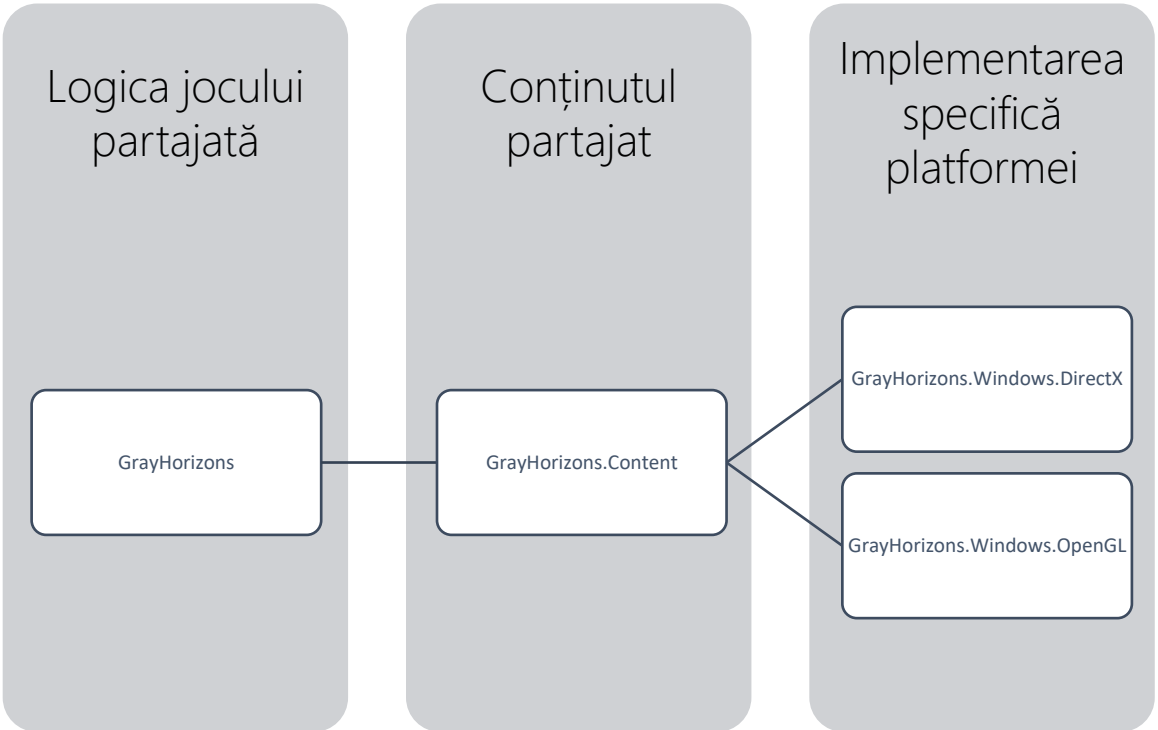
1. Proiectul partajat „`GreyHorizons`" (engl. Shared Project) se include în toate proiectelor derivate specifice unei platforme; acesta constituie practic întreaga logică și funcționalitate a jocului;

2. Proiectul „`GreyHorizons.Content`", care este de fapt un proiect ce nu necesită compilare; acesta conține toate fișierele de conținut aferente jocului (texturi, sunete, fonturi, etc.), aflate atât în original, cât și într-o formă ce poate fi citită de către MonoGame (compilate în .xnb și alte extensii de către MonoGame Content Pipeline).

3. Proiectul „`GrayHorizons.Windows.DirectX`” reprezintă implementarea propriu-zisă a jocului pe platforma Windows, folosind biblioteca grafică specifică — DirectX (folosind SharpDX).

4. Proiectul „`GrayHorizons.Windows.OpenGL`” reprezintă varianta OpenGL a jocului pentru platforma Windows.

Modularitatea conferită de această structură a proiectelor permite portarea cu ușurință a acestui joc pe un număr mare de alte platforme (Xbox, Windows 8, etc.).



PROIECTUL PARTAJAT – LOGICA

Întreaga funcționalitate a jocului este dată de către proiectul partajat intitulat `GreyHorizons`; acesta are o structură cvasi-complexă, conform CodeMap-ului generat de Microsoft Visual Studio 2013, sunt prezente un număr de 131 de noduri și 669 de legături între acestea. Namespace-urile (grupările ce conțin toate clasele) sunt următoarele:

`GrayHorizons.Logic`	În jurul acestui namespace orbitează toate celelalte, având drept rol fundația de obiecte folosite de restul namespace-urilor; aici sunt definite harta, jucătorul, configurația, obiectele abstracte, etc.
`GrayHorizons.Input`	Definește sistemul de control al jocului prin intermediul dispozitivelor de intrare (sunt implementate cele pentru tastatură și mouse, se pot implementa și pentru joystick, touch screen, etc.).
`GrayHorizons.Actions`	Face legătura dintre controlul oferit de `GrayHorizons.Input` și implementarea propriu-zisă a diverselor acțiuni (câteva exemple sunt: deplasarea tancului, controlul meniurilor, ...).
`GrayHorizons.AI`	Se ocupă de inteligența artificială ce se află în spatele tancurilor inamice, acesta le determină comportamentul într-un anumit interval de timp.
`GrayHorizons.Attributes`	Acesta este un namespace intern, ce definește un număr de atribute (meta-informații atașate claselor și a câmpurilor) ce sunt folosite pentru facilitarea procesului de dezvoltare și mentenabilitate.
`GrayHorizons.Entities`	Aici sunt definite toate entitățile (obiecte ce se pot afla în mișcare și se pot actualiza automat), precum soldații, tancurile și mașini.
`GrayHorizons.StaticObjects`	Obiectele statice ce pot fi incluse în hartă, se află definite aici: ziduri și alte obstacole, explozii, etc.
`GrayHorizons.Maps`	Totalitatea hărților folosite de către misiunile jocului sunt definite aici.
`GrayHorizons.Objectives`	Sistemul de obiective (obiective standard, lista lor, etc.) este implementat aici.
`GrayHorizons.Screens`	Aici sunt aflate toate ecranele ce definesc o anumită stare a jocului (în meniu, în timpul jocului, setări, HUD, etc.).
`GrayHorizons.Sound`	Totalitatea sunetelor ce sunt folosite în joc se află aici.
`GrayHorizons.ThirdParty`	Biblioteca `GameStateManagement`, împreună cu `RotatedRectangle` se află în acest namespace.

DESCRIEREA SUMARĂ A FUNCȚIONALITĂȚII

REPREZENTAREA OBIECTELOR

Jocul este format dintr-un număr de obiecte ce pot fi plasate pe câmpul de luptă. Fiecare obiect are stocată poziția acestuia, sub forma unui dreptunghi cu posibilitate de rotire în planul bidimensional al hărții.

Aceste obiecte posedă un nivel al „vieții" (integritatea acestora), iar cele două clase de bază ale acestora implementează parțial funcționalitatea de bază (verificarea coliziunii, deplasarea, etc.).

OBIECTE STATICE

(`GrayHorizons.Logic.StaticObject`)

Nu au nici o funcționalitate propriu-zisă, servind doar drept obstacole; câteva exemple sunt:

(1) Zidurile (`StaticObjects.Wall`), barierele antitanc (`AntitankBarrier`), precum și alte obstacole; acestea servesc doar scopul de obstacole.

(2) Exploziile (`StaticObjects.Explosion`), elemente ce sunt generate la comandă de către tancuri la tragere (`Vehicle.Shoot`) și distrugeri ale obiectelor; acestea își actualizează automat animația (având 10+ cadre diferite) și se autodistruge după timpul alocat.

(3) Entitățile ce au fost distruse sunt înlocuite cu „epave" (`StaticObjects.Wreck`), reprezentând o formă degenerată a acestuia.

(4) Orice alt element static al hărții, precum clădiri sunt reprezentate astfel.

ENTITĂȚI

(`GrayHorizons.Logic.Entity`)

Acestea, spre deosebire de obiectele statice, au capacitatea de a se deplasa și pot avea un comportament personalizabil:

(1) Soldații (`Entities.Soldier`) sau civilii (`Entities.Civilian`), ce au posibilitatea de a trage, de a conduce diverse vehicule și de a folosi obiecte din mediul înconjurător.

(2) Tancurile (ce derivă din `Logic.TankBase`, aflându-se în `Entities.Tank`, derivând la rândul lui din `Vehicle`) au posibilitatea de a trage (muniție limitată). Acestea sunt controlate de computer cu ajutorul inteligenței artificiale. De asemenea, tancurile dispun de o turelă ce se poate roti independent față de tanc.

(3) Alte vehicule (precum `Entities.Cars.MinelayerPickup` – plasatorul de mine) pot implementa propriul lor comportament.

(4) Proiectilele (`Entities.Projectile`) produse de către soldați, tancuri, etc. și își actualizează autonom traiectoria.

DESENAREA ELEMENTELOR GRAFICE

Precum a fost menționat anterior, sistemul de desenare este asigurat de către biblioteca **MonoGame** (variantă open-source a *Microsoft Xna Framework*). Această bibliotecă permite desenarea atât în plan bidimensional, cât și tridimensional, împreună cu anumite facilități precum citirea statutului dispozitivelor de intrare.

Deoarece sistemul de desenare este unul liniar, ce nu permite suprapunerea elementelor grafice decât în funcție de ordinea în care au fost inițial desenate, s-a recurs la folosirea unei biblioteci suplimentare. Această bibliotecă se numește `GameStateManager` și permite crearea de „ecrane" – unități grafice ce dispun de metode de desenare și actualizare independente.

Această metodă de desenare este aplicată în contextul ecranului principal, a meniurilor, al setărilor, etc. Câmpul de luptă este reprezentat de un singur ecran ce se ocupă singur de desenarea obiectelor. Atunci când obiectele din câmpul de luptă necesită desenare, modulul de desenare a elementelor apelează metoda `Render` a fiecărui obiect în parte, permițând modularitate.

Pentru a eficientiza procesul de încărcare a texturilor bidimensionale (`Texture2D`), se folosește un sistem propriu de preîncărcare a lor. Odată apelat când sesiunea de joc propriu-zisă începe, acest sistem se folosește de abilitatea de reflecție a limbajului pentru a căuta meta-informații legate de ce texturi ar trebui încărcate. Așadar, centralizând sistemul de încărcare al texturilor nu vor apărea momente de îngreunare cauzate de limitările de intrare/ieșire.

COLIZIUNEA ÎNTRE OBIECTE

Detectarea coliziunii între două obiecte este folosită pentru a preveni obiectele să se suprapună atunci când nu este cazul. Spre exemplu, un tanc trebuie să nu mai poată înainta atunci când se află în fața unui obstacol, sau un proiectil trebuie să detecteze impactul acestuia cu un alt obiect.

Astfel, pentru coliziune se folosește un sistem rudimentar: cele două obiecte sunt reprezentate în interiorul jocului drept două dreptunghiuri ce au poziționarea la nivel de pixel și rotire arbitrară. A fost menționat anterior faptul că pentru rotire arbitrară s-a folosit biblioteca `RotatedRectangle`. Testul de coliziune este simpla verificare dacă cele două dreptunghiuri se intersectează într-un punct oarecare, dacă da, atunci cele două obiecte se află în coliziune.

HARTĂ DERULABILĂ ȘI SISTEMUL DE SCALARE

O altă caracteristică a acestui joc este faptul că harta are capabilitatea de a depăși dimensiunile ecranului. Astfel, jucătorul poate naviga printr-o hartă ce se derulează odată cu entitatea ce este controlată de către jucător. Poziția obiectelor este relativă față de colțul stânga-sus al hărții.

De asemenea, câmpul de vizualizare poate fi supus scalării, fiind posibilă scalarea acesteia. Sunt implementate funcții de a micșora elementele grafice astfel încât să se poată vedea harta într-o perspectivă mai largă.

Pentru translatarea coordonatelor relative față de hartă la cele relative față de fereastra de desenare și aplicarea scalării, se folosește o metodă `CalculateViewportCoordinates` ce aparține clasei hărții (`Map`).

SISTEMUL DE PROPAGARE A DATELOR ESENȚIALE

Deoarece clasele în interiorul jocului sunt separate între ele, neputând-se accesa datele dintre acestea în mod implicit, a fost implementat o clasă intitulată `GameData` (date ale jocului).

Clasa `GrayHorizons.Logic.GameData` conține diverse proprietăți ce conțin referințe către informații esențiale ce trebuie propagate către celelalte clase. Printre aceste informații se enumeră:

- O listă a tuturor jucătorilor din sesiunea actuală de joc („Orizonturi cenușii" are infrastructura completă pentru a permite mai mulți jucători simultan – *Multiplayer*, dar care nu a fost finalizată în timp util până la termenul limită).
- Lista referințelor texturilor bidimensionale ce sunt preîncărcate pentru a spori eficiența prin încărcarea centralizată a lor.
- Câmpul de lupta (harta), împreună cu lista de obiective specifice acestei sesiuni de joc.
- Instanțe ale claselor utilizate specific pentru desenarea elementelor (`GraphicsDevice`, `GraphicsDeviceManager` și `ScreenManager`).

Atunci când este pornit jocul se creează o instanță a acestei clase care va fi folosită pe tot timpul funcționării lui. Locația din memorie (pointerului) a clasei purtătoare de date este transferată apoi tuturor claselor ce necesită accesul la date importante (printre care harta, toate obiectele, sistemul de inteligență artificială, etc.).

MINI-HARTA

Utilizatorului îi este oferită posibilitatea de a vedea câmpul de luptă în întregime, la o dimensiune redusă în colțul din dreapta-sus (poziția hărții poate fi schimbată folosind tasta F7). Acest lucru permite o vedere de ansamblu asupra poziției inamicilor, permițând dezvoltarea unei strategii de luptă.

Terenul (textura hărții) este miniaturizată, iar obiectele importante sunt marcate pe această mini-hartă folosind mici pătrate. Culoarea pătratelor indică tipul acestora (alb – jucătorul, galben – vehicule, roșu – tancuri, gri – obiecte statice, etc.) și este definită drept proprietate în clasa respectivului obiect.

INTERFAȚA HEADS-UP DISPLAY

În termeni de *gaming*, **Heads-Up Display** reprezintă acele elemente grafice care sunt suprapuse peste grafica propriu-zisă a jocului cu scopul de a oferi informații suplimentare ce sunt de importanță jucătorului.

Aici, aceste tipuri de elemente sunt reprezentate de către:

- Obiectivul actual al jucătorului în cadrul misiunii curente;
- Mini-harta prezentată anterior;
- Indicatorul de „viață" și muniție rămase a jucătorului.

FUNDAMENTE GEOMETRICE UTILIZATE

În cadrul jocului „Orizonturi cenușii”, datorită naturii acestuia, a fost necesară adoptarea unor noțiuni elementare de geometrie în concordanță cu logica software. Urmează a fi prezentate în continuare câteva dintre cele care au fost folosite:

TRANSFORMAREA DIN GRADE ÎN RADIANI ȘI VICE-VERSA

De o mare importanță pentru joc este capabilitatea de a putea transforma rapid și facil rotația din grade în radiani:

$$\alpha_{radiani} = \frac{\pi}{180°} * \alpha_{grade}$$

Prin extensie, pentru a converti din grade în radiani:

$$\alpha_{grade} = \frac{180°}{pi} * \alpha_{radiani}$$

DETERMINAREA DEPLASĂRII FRONTALE A UNUI PUNCT OARECARE

Atunci când o entitate necesită a fi deplasată cu o oarecare măsură în față, se poate folosi formula:

$$\begin{cases} \Delta_x = \sin\alpha * \beta \\ \Delta y = -\cos\alpha * \beta \end{cases}$$

unde α este unghiul de rotație al obiectului, iar β este factorul de deplasare.

Dacă se dorește deplasarea inversă, spre exemplu, în marșarier a tancului:

$$\begin{cases} \Delta_x = -\sin\alpha * \beta \\ \Delta y = \cos\alpha * \beta \end{cases}$$

DEPLASAREA A UNUI DREPTUNGHI AFLAT SUB ROTAȚIE CU X ȘI Y

Una dintre problemele fundamentale întâlnite în acest joc este cea de a putea deplasa coordonatele efective (x și y, aflate sub influența rotației) a unui dreptunghi ce permite rotație luând în considerare coordonatele neutre (fiind la 0°).

Deplasarea coordonatei neutre x în funcție de rotire este determinată de deplasarea neutră a dreptunghiului prin punctele x' și y'.

$\begin{cases} x' = \cos r * x \\ y' = \sin r * x \end{cases}$, unde r este unghiul de rotire a dreptunghiului exprimat în radiani.

Pentru deplasarea coordonatei y se folosește:

$$\begin{cases} x' = \sin r * -y \\ y' = \cos r * y \end{cases}$$

Câteva exemple de utilizare sunt reprezentate de determinarea punctului de origine a proiectilelor atunci când sunt trase, a poziției turelei relativ față de tanc și a deplasării proiectilelor.

DETERMINAREA UNGHIULUI DE ROTIRE A LUI A PENTRU A SE ÎNDREPTA SPRE *B*

Luând două dreptunghiuri aflate într-un plan (cu posibilitate de rotire), A și B, este necesară determinarea unghiului de rotire față de centru a lui A astfel încât acesta să se afle față în față cu dreptunghiul B.

$$\alpha = \left(\operatorname{atan}\frac{y}{x}\right)° + 180°$$

Această formulă are o utilitate extrem de mare în determinarea rotației turelei unui tanc deținut de jucător astfel încât aceasta să urmeze poziția mouse-ului. De asemenea, aceasta este folosită și în cadrul inteligenței artificiale pentru a îndrepta turela către un alt obiect.

DETERMINAREA DISTANȚEI A DOUĂ PUNCTE A ȘI B

Fiind date două puncte A și B, distanța dintre cele două puncte este determinată printr-o variantă a Teoremei lui Pitagora:

$$\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

Formula de calculare a distanței este implementată în clasa hărții pentru a facilita determinarea distanței a celei mai apropiate ținte pentru inteligența artificială.