The Far Lands

Game Design Document

Matei Alexandru Antonie
Proiect licenta Unibuc FMI, prof. Coordonator Popescu S.

1. Conceptul jocului

Titlu joc: **The Far Lands**
Platforma: **PC (Windows)**
Engine folosit: **Unity**
Gen: **First-person Survival / 3D Sandbox**
Inspiratii: **Minecraft, Survivalcraft**

1.1. Descriere scurta

The Far Lands este un joc 3D interactiv de tip sandbox voxel-based, axat pe explorarea si interactiunea cu o harta infinita generata procedural in timp real formata din blocuri. Player-ul poate interactiona cu harta prin a sparge si a plasa blocuri unde doreste, modificand astfel aspectul hartii.
Atfel, lumea este dinamica si formata dintr-o grila 3D de blocuri/voxeli. Din punct de vedere tehnic prin acest proiect se prezinta algoritmica din spatele acestui gameplay, ce consta in optimizare spatiala si grafica (Culling, Meshing), management-ul eficient al memoriei (impartirea lumii in chunk-uri, care se incarca dinamic in memorie dupa un anumit rendering distance), demonstrand capacitatea de a afisa milioane de elemente 3D dinamice la o performanta buna.

2. Mecanici de baza de gameplay

Voi detalia pe scurt actiunile principale pe care le poate face jucatorul si modul in care jocul reactioneaza la input-urile acestuia.

2.1. Perspectiva si deplasarea

Jocul este de tip first-person, jucatorul vazand lumea din ochii personajului principal, pentru a oferi o imersiune maxima si o precizie ridicata in manipularea blocurilor individuale.

Deplasarea 3D se face cu sistemul clasic de controale W, A, S, D pentru miscare, mouse pentru rotatia camerei, Space pentru saritura.

2.2. Interactiunea cu lumea

Player-ul poate traversa harta cu sistemul de movement mentionat anterior, blocurile avand sistem de coliziune. Jucatorul poate interactiona cu aceste blocuri prin 3 mari metode:

- Selectie - tinand cursorul (crosshair-ul) pe un bloc suficient de apropiat de player, blocul va avea un outline ce indica posibilitatea de a-l sparge/distrugere sau de a plasa un alt bloc peste el
- Distrugere bloc - la comanda jucatorului (left click) blocul selectat este distrus si inlocuit cu “aer”, actualizand automat geometria zonei in care se afla pentru a reflecta schimbarile in timp real
- Plasare bloc - la right click se plaseaza peste fata selectata a blocului alt bloc, de asemenea updatat in timp real

2.3. MVP cu core gameplay loop

Fiind un joc sandbox gameplay-ul se desfasoara intr-un ciclu continuu format din cateva activitati principale:

- Explorarea hartii, navigarea prin terenul generat procedural, descoperirea noilor forme de relief, observarea peisajelor
- Modificarea mediului prin eliminarea obstacolelor, plasarea drumurilor, podurilor
- Constructia structurilor personalizate - case, platforme - la imaginatia jucatorului

2.4. Mecanici suplimentare

Ce am evidentiat pana acum se incadreaza in MVP, un prototip cu mecanicile de baza. In plus, in limita timpului disponibil, urmaresc implementarea urmatoarelor elemente:

Obiective tehnice:

- Generale procedurala 3D - sistem de pesteri subterane - extinderea algoritmului de generare a hartii de la 2D la 3D fara un impact prea mare asupra performantei

Obiective de gameplay:

- Ciclu dinamic zi/noapte - simularea trecerii timpului prin manipularea surselor globale de lumina si parametrilor iluminarii mediului
- Supravietuirea: player-ul are HP, exista pe harta surse de damage si de regenerare, in plus player-ul are un sistem de foame/sete, in care trebuie sa isi gestioneze resursele corespunzator si sa gaseasca oaze de apa si surse de mancare

Polish vizual/estetica:

- O paleta de blocuri din care jucatorul poate selecta un bloc curent pentru a construi cu el, astfel se extinde cu mult posibilitatea constructiilor personalizate si modificarea reliefului generat procedural dupa plac
- Sunete pentru un gameplay mai imersiv, muzica de fundal a jocului

3. Arhitectura tehnica

In aceasta sectiune voi propune principalele provocari tehnice si solutiile la care ma gandesc pentru a ajunge la obiectivele mentionate si la varianta finala a jocului.

3.1. Reprezentarea blocurilor

Jocul va avea o “decuplare” intre datele logice ale lumii si reprezentarea vizuala. Lumea va fi impartita in Chunk-uri de dimemnsiune fixa - subregiuni care contin o matrice 3D de elemente propriu-zise. Fiecare element va avea un element identificator numeric (ex. 0 - aer, 1 - pamant, 2 - piatra), mentinand o performanta ridicata si o amprenta redusa de memorie.

3.2. Generarea procedurala a reliefului

Lumea nu va fi pre-generata, ci va fi creata in momentul gameplay-ului in timp real, treptat prin instantierea fiecarui nou chunk. Se vor folosi functii de zgomot coerent - nu noise random - ci ceva similar cu Perlin Noise/Simplex Noise, evitand generare pur-random care ar rezulta intr-o lume haotica. Acest noise va fi transformat intr-o valoare de inaltime pe axa Y si matricea chunk-ului va fi populata cu blocuri solide de la baza lumii pana la acest Y pentru X si Z-urile respective. De asemenea, ca “Stretch Goal” mi-am propus extinderea la Noise 3D pentru a excava spatii subterane goale sub nivelul solului, formand retele de caverne.

3.3. Optimizari

Randand fiecare bloc complet ar crea probleme mari de performanta, asa ca optimizarea principala consta in urmatoarele puncte:

- Constructia dinamica a geometriei - se foloseste un singur mesh pentru un chunk intreg
- Culling - se vor randa doar fetele vizibile ale blocurilor (care au vecin aer)
- La orice interactiune a jucatorului cu mediul se va re-calcula si re-afisa doar chunk-ul in care s-a produc modificarea, asigurand timp de raspuns insesizabil
- Hash Maps pentru stocarea chunk-urilor active pentru timp de accesare O(1)
- Lumea se va genera automat intr-un perimetru circular in jurul jucatorului - se vor instantia chunk-uri noi intr-o raza pre-definita (render distance)
- Chunk-urile care ies din aceasta raza vizuala sunt dezactivate controlat pentru a elibera resurse

3.4. Flux de utilizare al aplicatiei

Aplicatia va avea un flux de utilizare format din 3 stari:

- Utilizatorul se afla pe meniul principal (title screen) - unde poate intra in joc, poate face ajustari la controale/setari si poate incheia sesiunea din aplicatie
- Utilizatorul se afla mid-gameplay
- Utilizatorul este in meniul de pauza (gameplay-ul se opreste temporar pana iese din meniu)

3.5. Interfata grafica UI/HUD (Heads-up Display)

Interfata UI/HUD-ul din gameplay va fi unul minimalist ce afiseaza informatiile esentiale:

- Crosshair in mijlocul ecranului pentru a facilita selectarea si manipularea blocurilor din lume
- Hotbar, HP, informatii vitale, indicatori de stare

3.6. Sistemul de input

Sistemul de input va fi proiectat pentru periferice standard de PC (mouse si tastatura) si va functiona pe baza captarii si procesarii continue a evenimentelor de intrare:

- Translatie si rotatie - navigarea hartii prin tastele W, A, S, D, miscarea mouse-ului pentru rotatia camerei si sarit cu Space
- Interactiunea cu mediul - click stanga pentru spargere bloc, click dreapta pentru plasare bloc, rotita/numere pentru a selecta blocul curent echipat pentru plasare

4. Modul de lucru

Pentru a mentine un flux de lucru profesionist, ciclul de dezvoltare va respecta standardele si bunele practici din industrie:

- Version control - intregul cod sursa si asset-urile vor fi stocate intr-un repository GitHub, garantand siguranta datelor, istoricul detaliat al modificarilor si posibilitatea izolarii noilor functionalitati pe branch-uri separate inainte de a fi integrate in proiectul principal
- Organizarea task-urilor in sistem Kanban - cu GitHub projects/Jira - descompunerea goal-urilor in tichete individuale si mutate prin stari specifice (To-Do, In Progress, Done) si planificare in sprinturi pentru o monitorizare eficienta a evolutiei proiectului si pentru un workflow cat mai productiv si organizat.
- Arhitectura modulara - proiectul Unity va fi organizat intr-o ierarhie stricta a directoarelor pentru a separa resursele (Script-uri, Materiale, Prefab-uri, UI, Sunete) - de asemenea, codul va respecta bunele practici si fiecare element va fi in locul sau corespunzator, evitand conflictele si dependentele care nu sunt necesare. Astfel, si debugging-ul/QA-ul va fi mult mai simplificat.

5. Roadmap

Procesul de dezvoltare a jocului va trece prin urmatoarele mari etape:

- **1.** Fundatia tehnica: definirea structurilor de date pentru stocarea hartii si functionalitatile de afisare a acestor structuri in timp real pe ecran. **Scop livrabil:** un sistem capabil sa instantieze blocuri si sa le randeze la un framerate optim, generand doar geometria exterioara
- **2.** Generarea procedurala si interactiunea: integrarea algoritmilor de zgomot pentru modelarea reliefului, dezvoltarea unui manager al lumii pentru incarcarea/descarcarea chunk-urilor, raycasting si sistemul de adaugare si eliminare de blocuri, integrarea controlului First-Person (fizica si coliziuni). **Scop livrabil: MVP complet functional**
- **3.** Extinderea functionalitatii: implementarea meniurilor, generare de pesteri, ciclu zi-noapte, mecanici de stare. **Scop livrabil:** un joc mai interactiv si mai complet decat un “tech demo”
- **4.** QA si documentatie: **Feature-freeze** - schimbarea focusului pe bug-hunting, optimizari si organizari finale + redactarea tezei finale de licenta si a concluziilor

Fiecare etapa este planuita sa aiba loc de-a lungul a **3 saptamani** sub forma unui sprint bine organizat si documentat. Prin design etapa 3 poate sa fie “sacrificata” in cazul in care celelalte etape necesita mai mult timp, ele constituind parti obligatorii in procesul dezvoltarii jocului, pe cand etapa 3 intra in categoria “Nice-to Have”.

Redactarea tezei de licenta va fi facuta continuu de-a lungul procesului dezvoltarii aplicatiei pentru a avea contextul proaspat in minte, la final facand doar reformulari, modificari si proofreading la conceptele deja detaliate.