POLITECNICO DI TORINO

Dipartimento di Automatica e Informatica

Tesi di Laurea Magistrale in Game Design

Implementazione di un sistema di meteo physics-based NPR

Relatore

Prof. Marco MAZZAGLIA

Correlatore

Prof. Alessandro PEZZOLI

Candidato

Lorenzo SEMERARO

Aprile 2023

Ai miei amici.

Abstract

Uno dei più significativi aspetti di realismo nell'ambito della creazione di mondi virtuali è la presenza di un sistema di simulazione meteo convincente. I sistemi di meteo dinamici più apprezzati realizzati finora sono basati sull'analisi di fenomeni atmosferici reali, pertanto anche la resa grafica tende il più possibile al fotorealismo. Tuttavia, quest'approccio potrebbe ostacolare il senso di immersione se il resto dell'applicazione interattiva presenta uno stile non photorealistic (NPR) evidente, e dunque rendere meno efficace un sistema di meteo dinamico ben realizzato. In questo lavoro di tesi verrà presentata un'implementazione di un sistema di simulazione real-time delle condizioni meteo, associato a una serie di shader procedurali che ne permettono una resa grafica stilizzata di tipo NPR. Il sistema sarà integrato all'interno di un motore di gioco come Unity 3D per permettere agli sviluppatori di applicazioni interattive di abbreviare i tempi di lavoro. La grafica stilizzata, combinata con algoritmi physics-based, apre le porte a nuove potenzialità in termini di ottimizzazione per il real-time: si intende dunque esplorare anche questo aspetto per garantire prestazioni adeguate al tipo di applicativi in esame.

Indice

Lista delle figure

Capitolo 1

Introduzione

1.1 Immersione e presenza in simulazioni open-world

L'evoluzione dei game engine, strettamente legata a quella tecnologica, permette l'esecuzione di algoritmi sempre più complessi su macchine di livello consumer. Per questo motivo, vengono anche impiegati per sviluppare applicativi che esulano dall'intrattenimento, attraverso simulazioni di mondi realistici e interattivi finalizzati a diversi scopi, tra cui la formazione tecnica o psicoterapia.

L'efficacia di questi strumenti è senza dubbio proporzionale al senso di immersione e di presenza che questi riescono a garantire; si possono definire, rispettivamente, come il livello di estraniamento dal contesto reale e la sensazione di appartenenza al mondo virtuale. Il primo dipende inevitabilmente dal tipo di interfaccia utilizzata per fruire dell'applicazione, mentre il secondo è funzione sia dell'immersione, sia del livello di interattività con il mondo virtuale.

Garantire una buona presenza risulta un requisito importante per le applicazioni virtuali interattive, e riuscirci è tanto più complesso quanto più gli elementi del mondo virtuale hanno un aspetto organico e naturale. In questo scenario, è chiaro che gli sviluppatori dei videogiochi che presentano la meccanica "open-world", ovvero implementano un mondo virtuale vasto e liberamente esplorabile, si troveranno ad affrontare questa sfida. In particolare, uno degli elementi che contribuiscono notevolmente al senso di immersione è la volta celeste, che in aree all'aperto occupa metà del campo visivo durante la maggior parte dell'esperienza. Per questo motivo, lo sviluppo di un sistema meteorologico curato può influire sensibilmente sulla qualità complessiva dell'applicazione.

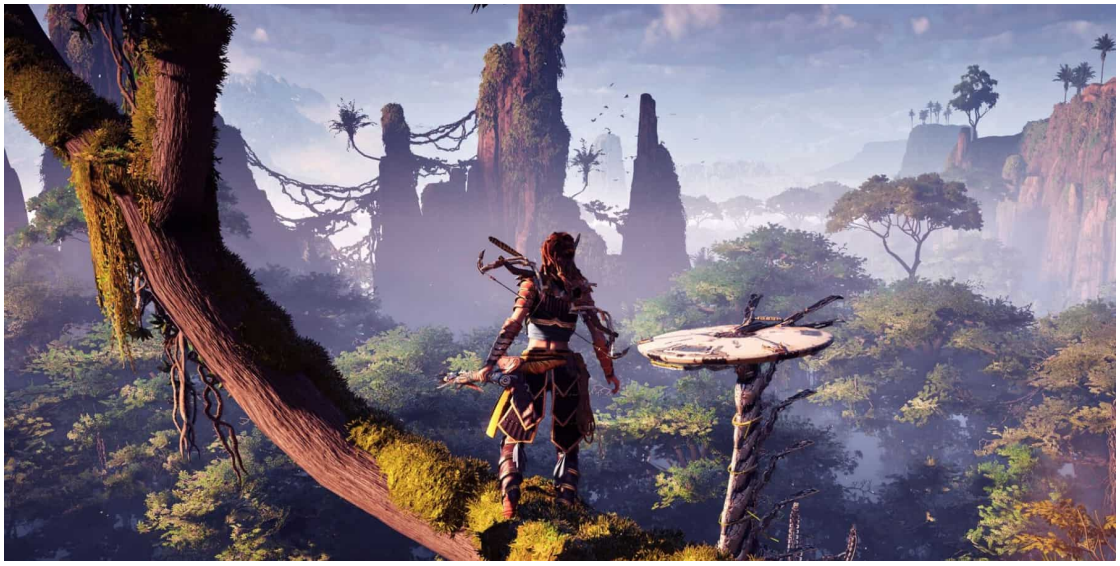

Figura 1.1: Horizon Zero Dawn: videogioco open-world.

1.2 Limitazioni attuali

Nello stato dell'arte si possono includere diversi videogiochi attuali, che spingono la tecnologia al limite per ottenere buone prestazioni e al contempo una resa visiva piacevole e dettagliata. Tuttavia, la quantità di informazioni tecniche condivise dagli sviluppatori, specialmente se riguardano tematiche così specifiche, non è sufficiente per stabilire con precisione quali sono le tecniche e i parametri utilizzati per ottenere il risultato.
Per questa ragione, gran parte delle informazioni sui sistemi meteo dinamici saranno evinte attraverso l'analisi di una serie di applicazioni selezionate con criterio, nonché integrate e confrontate con osservazioni pubblicate dai giocatori stessi.

In generale, è osservabile che videogiochi dalla grafica foto-realistica presentano sistemi meteo dinamici piuttosto immersivi. Gli sviluppatori creano dunque un modello algoritmico che rappresenta alcuni fenomeni fisici reali, talvolta affiancato da elementi casuali col fine di creare la giusta dose di imprevedibilità; oppure vengono utilizzate informazioni reali di eventi meteorologici già avvenuti.

Al contrario, la tendenza è di adoperare diverse approssimazioni quando si tratta di videogiochi dall'aspetto stilizzato (Non-Photorealistic Rendering o NPR): si limita il sistema ad una serie di fenomeni casuali, o in casi più evidenti si lega una specifica condizione meteorologica ad un'area del mondo virtuale. Probabilmente, l'obiettivo che spinge gli sviluppatori ad adoperare quest'ultima scelta risiede nel

controllo artistico della scena, che ricopre un ruolo di maggiore rilievo nei prodotti video-ludici con particolare identità estetica, o nel contenimento dell'overhead del software.
D'altra parte, l'eccessiva semplificazione e staticità del sistema meteo potrebbe compromettere il senso di immersione del giocatore all'interno del mondo virtuale, che potrebbe già essere indebolito dalla resa visiva stilizzata. Per comprendere a fondo le lacune e le potenzialità dei sistemi di meteo dinamici sviluppati per le applicazioni interattive, si analizzano nel prossimo capitolo 5 videogiochi dalle caratteristiche variegate per tecnologie utilizzate, stile grafico, politiche, transizioni tra condizioni meteorologiche e impatto delle stesse sul game-play.

Capitolo 2

Stato dell'arte

2.1 Shenmue

Shenmue è stato uno dei primi videogiochi a porre particolare attenzione alla generazione pseudo casuale, implementando i sistemi di Magic Rooms e Magic Weather, rispettivamente per la generazione di ambienti chiusi virtuali e di condizioni meteorologiche.

Questi sistemi sono due tra gli elementi più importanti per la creazione dell'universo del gioco.

Funzionamento generale. Per ogni giorno in cui il giocatore progredisce, Magic Weather genera condizioni meteo casuali che includono pioggia, neve, cielo coperto, soleggiato e molte altre varianti che possono alternarsi durante un giorno virtuale.

Le condizioni generate riflettono la stagione in cui si svolgono.

Politica di decisione. Viene fornita una spiegazione dell'elaborazione meteorologica, con un diagramma di flusso di accompagnamento.

Nella prima uscita del videogioco, la scelta delle condizioni meteo dipendeva esclusivamente dalla data corrente e la posizione della camera virtuale. Ogni volta che è trascorso il periodo di tempo prescritto (ad esempio, un'ora) dalla precedente elaborazione meteorologica, la condizione meteorologica viene determinata da una Look-Up Table, in base alla posizione e all'ora correnti della telecamera. La tabella meteo è descritta come contenente voci per ogni area di gioco o posizione designata nel gioco. Contiene impostazioni dettagliate tra cui la data, l'ora del giorno, l'aspetto del cielo di sfondo, le condizioni meteorologiche (ad esempio: nuvoloso / nebbioso

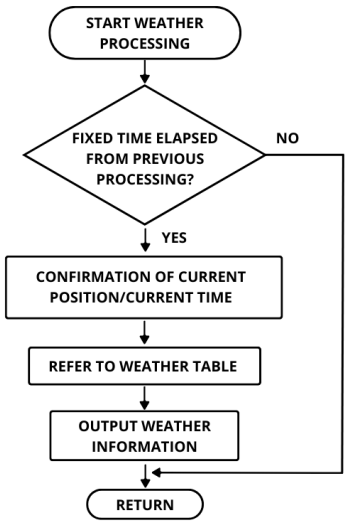

Figura 2.1: Diagramma di flusso della politica di decisione del meteo in Shenmue.

DATA	TIME	SKY (BACKGROUND)	WEATHER	DEGREE OF WEATHER
JANUARY 1	00:00	NIGHT/CLOUDY	CLOUDY	-
	01:00	NIGHT/CLOUDY	CLOUDY	-
	02:00	NIGHT/CLOUDY	RAIN	LIGHT
	03:00	NIGHT/CLOUDY	RAIN	LIGHT
	04:00	NIGHT/CLOUDY	RAIN	MEDIUM
	05:00	NIGHT/CLOUDY	SLEET	MEDIUM
	06:00	DAWN/CLOUDY	SNOW	MEDIUM
	07:00	DAWN/CLOUDY	SNOW	MEDIUM
	08:00	NOON/CLOUDY	SNOW	HEAVY
	09:00	NOON/CLOUDY	SNOW	HEAVY
	10:00	NOON/CLOUDY	SNOW	HEAVY
	11:00	NOON/CLOUDY	SLEET	HEAVY
	12:00	NOON/CLOUDY	RAIN	HEAVY
	13:00	NOON/CLOUDY	RAIN	HEAVY
	14:00	NOON/CLOUDY	RAIN	HEAVY
	15:00	NOON/CLOUDY	RAIN	MEDIUM
	16:00	EVENING/CLOUDY	RAIN	MEDIUM
	17:00	EVENING/CLOUDY	RAIN	LIGHT
	18:00	NIGHT/CLOUDY	RAIN	LIGHT
	19:00	NIGHT/CLOUDY	CLOUDY	-
	20:00	NIGHT/CLOUDY	CLOUDY	-
	21:00	NIGHT/CLOUDY	CLOUDY	-
	22:00	NIGHT/CLEAR	CLEAR	-
	23:00	NIGHT/CLEAR	CLEAR	-
JANUARY 2	00:00	NIGHT/CLEAR	CLEAR	-
	01:00	NIGHT/CLEAR	CLEAR	-
	02:00	NIGHT/CLEAR	CLEAR	-
	03:00	NIGHT/CLEAR	FOGGY	LIGHT
	04:00	NIGHT/CLEAR	FOGGY	LIGHT
	05:00	NIGHT/CLEAR	FOGGY	DENSE
	06:00	DAWN/CLOUDY	FOGGY	DENSE
	07:00	DAWN/CLOUDY	FOGGY	LIGHT
	08:00	NOON/CLOUDY	FOGGY	LIGHT
	09:00	NOON/CLOUDY	CLEAR	-
	10:00	NOON/CLOUDY	CLEAR	-
	11:00	NOON/CLOUDY	CLEAR	-
	12:00	NOON/CLOUDY	CLEAR	SUNNY
	13:00	NOON/CLOUDY	CLEAR	SUNNY
	14:00	NOON/CLOUDY	CLEAR	SUNNY

Figura 2.2: Look-up table di Magic Weather.

/ pioggia / nevischio / neve / chiaro) e l'intensità delle condizioni meteorologiche.

Le condizioni soleggiate e nuvolose sono rappresentate alterando la luminosità in post-processing, e questa luminosità viene ulteriormente modificata in base all'ora del giorno. Il vento (compresa la direzione e la forza) può essere aggiunto a queste informazioni meteorologiche.

Con il buon livello di controllo possibile, ci si potrebbe aspettare che ciò richieda una grande quantità di dati da archiviare, ma non è stato così. I dati meteorologici potrebbero essere memorizzati in un formato compatto, con ogni voce nella tabella meteorologica in grado di essere espressa come un singolo numero a 16 bit (un intervallo da 0x0000 a 0xFFFF in esadecimale).

Nonostante la sua forma compatta, ogni voce sarebbe in grado di indirizzare 24 diverse skybox, 24 effetti meteorologici e 28 livelli di intensità.

Essendo di dimensioni compatte, il gioco carica semplicemente l'intera tabella

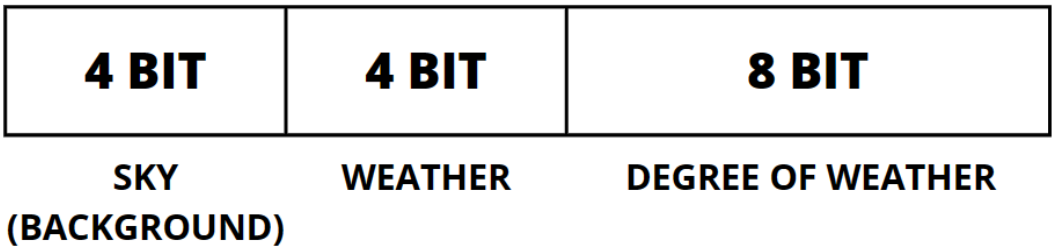

Figura 2.3: Formato di rappresentazione di Magic Weather.

meteorologica e la tiene in memoria centrale.

Con Shenmue II, l'approccio della tavola meteorologica completamente definito è semplificato definendo solo i limiti superiori del tempo in un determinato intervallo di tempo: viene quindi avviato un algoritmo di pseudo generazione di numeri casuali (PRNG). Inoltre, le registrazioni delle condizioni meteorologiche effettive dell'area di Yokosuka durante il 1986/1987 (luogo e periodo storico in cui si svolgono le vicende del videogioco) sono state implementate, dando ai giocatori la possibilità di sperimentare queste condizioni meteorologiche oltre a quelle generate casualmente. Questo meccanismo potrebbe risultare interessante per applicazioni di realtà virtuale, nel caso in cui si volesse creare un mondo virtuale il cui meteo rispecchi le reali condizioni del mondo fisico.

In questo caso, sarebbe necessario definire uno standard per la rappresentazione di condizioni meteorologiche attraverso diversi parametri, la cui quantità risulta proporzionale alla precisione che il sistema deve garantire.

Transizioni. Mentre il giocatore si muove tra le aree, le condizioni meteorologiche possono cambiare. Questa situazione viene gestita tramite interpolazione per garantire una transizione graduale delle condizioni meteorologiche tra le aree.

Questa transizione graduale viene implementata con combinazioni di curve di interpolazione B-spline, ognuna delle quali rappresenta uno dei tipi di tempo esistenti. Ciò significa che, ad esempio, la neve può trasformarsi dolcemente in pioggia - o qualsiasi altra transizione meteorologica immaginabile.

Impatto sul game-play. Il tempo ha un impatto diretto su come appare il gioco: in una giornata piovosa, le persone camminano con gli ombrelli e, in una

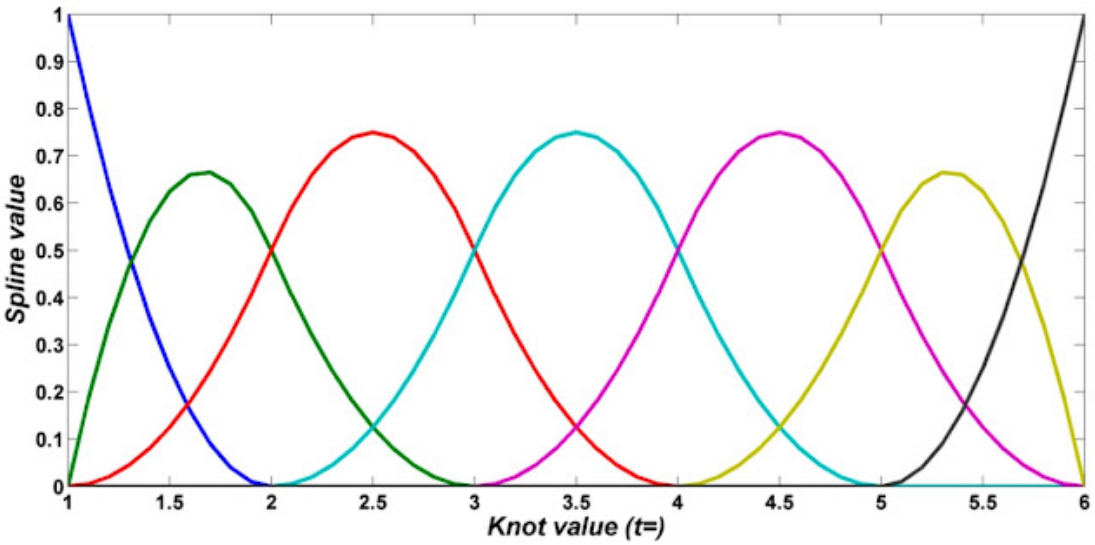

Figura 2.4: B-spline per le transizioni di Magic Weather.

giornata nevosa, la strada è coperta di neve.
La discussione sul sistema Magic Weather si conclude sottolineando il miglioramento rispetto agli altri sistemi meteorologici più semplici fino ad oggi: "Pertanto, rispetto alle rappresentazioni convenzionali delle condizioni meteorologiche che cambiano semplicemente il cielo (sfondo) dopo un tempo prescritto, è possibile visualizzare realisticamente le condizioni meteorologiche dettagliate. Questo fornisce un sufficiente senso di realismo e aumenta l'interesse per ogni rispettiva scena del gioco di ruolo".

2.2 GTA V

Il sistema di questo videogioco è stato selezionato come caso di studio per la sua capacità di condizionare diversi aspetti dell'ambiente e del game-play.

Funzionamento generale. Gli eventi meteorologici possono durare tra 120 secondi reali e un'ora reale. La pioggia notturna nella modalità online è disabilitata, probabilmente per questioni di ottimizzazione: per rappresentare realisticamente superfici bagnate in una scena cittadina notturna (e dunque densa di diverse luci artificiali) è necessario fare un uso intensivo di shader riflettenti, e quindi di illuminazione indiretta, che utilizza algoritmi computazionalmente onerosi.

Inoltre, sarebbe necessario sincronizzare diversi avvenimenti, tra cui la formazione di pozzanghere e la posizione dei fulmini.

Figura 2.5: La resa visiva di una tempesta in GTA V.

Politica di decisione. Questo titolo propone una politica di decisione pseudo-randomica: la sequenza di condizioni meteorologiche è fissata; questa staticità viene compensata dalla lunghezza della sequenza. Infatti, il ciclo meteorologico in GTA si ripete ogni 12,8 ore di vita reale, e cambia 55 volte in questo intervallo – o meglio, 54 volte, perché il ciclo inizia e finisce con lo stesso tipo di tempo, "Parzialmente nuvoloso".

Tuttavia, alcune scene di gioco possono avere il loro tempo predefinito per ottenere un effetto drammatico o, in altri casi, una difficoltà aumentata o ridotta.

Il sistema è quindi assimilabile alla gestione di una struttura dati di tipo FIFO. Segue una possibile implementazione in pseudo-codice.

Transizioni. Le precipitazioni possono variare a seconda di quanto sono limpidi i cieli. Dunque, si può supporre la presenza di parametri per determinare la densità di nubi nel cielo, che modula un altro parametro responsabile di regolare l'intensità della pioggia.
Inoltre, il passaggio tra diverse condizioni non risulta brusco.

Impatto sul game-play. Il sistema di meteo considerato ha un visibile riscontro su NPC nelle città: in caso di maltempo, la loro densità cala rapidamente e ognuno

Algorithm 1 Pseudo-codice che descrive la selezione del meteo di GTA V.

```
1:  procedure WeatherSelection
2:      ▷ Initialization
3:      List weatherList[] ← [...]                          ▷ 55 elementi
4:      int weatherTimer ← weatherList[0].duration
5:      int weatherIndex ← 0
6:      ▷ Execution
7:      while weatherTimer > 0 do
8:          if scene not in dramaticScenes then
9:              PickNextWeather(weatherIndex)
10:         end if
11:     end while
12: end procedure
13:
14: procedure PickNextWeather(int i)
15:     SetWeather(weatherList[i])
16:     if i < 54 then
17:         i++
18:     else
19:         i ← 0
20:     end if
21: end procedure
22:
23: procedure SetWeather(Weather w)
24:     w.VFX.Play();
25:     weatherTimer ← w.duration
26: end procedure
```

di loro presenta comportamenti reattivi e diversificati. Durante le forti tempeste, la luce gialla sui semafori lampeggerà continuamente, indicando che l'alimentazione è stata spenta.

Inoltre, le precipitazioni incidono sulle prestazioni di molti veicoli stradali e possono degradare la maneggevolezza e la potenza frenante a causa di strade scivolose in maniera proporzionale all'intensità delle stesse.

Il vento ha un impatto fisico sugli aerei aumentando la turbolenza e spingendoli in direzioni casuali.

Queste ultime due analisi suggeriscono che il sistema meteo interagisce anche

con il motore di simulazione fisica del videogioco, applicando delle forze o modificando i parametri di comportamento di diversi game object.

In Grand Theft Auto V anche la nebbia e le tempeste di sabbia sono implementate, ed hanno l'effetto di limitare la visibilità dell'ambiente virtuale.

Da un punto di vista tecnico, questa operazione potrebbe permettere agli sviluppatori di risparmiare sulla complessità computazionale avvicinando il piano di clipping posteriore e scartando quindi diversi elementi grafici dalla pipeline di rendering.

2.3 Genshin Impact

Genshin Impact è un recente videogioco MMORPG di successo basato soprattutto sull'esplorazione del mondo di gioco open-world.

Il meteo nel mondo virtuale di Tevyat varia a seconda della posizione e del tempo. Sono implementate molte condizioni meteorologiche diverse, che vanno dal cielo soleggiato ai temporali alla nebbia fitta.

Funzionamento generale. Le condizioni meteorologiche sono legate alle aree della mappa, quindi il giocatore può spesso teletrasportarsi in una posizione diversa per sfuggire a determinate condizioni meteorologiche.

Inoltre, alcune situazioni sono caratterizzate da un evento atmosferico fisso, nelle restanti invece il sistema è dinamico.

Lo stile grafico del videogioco è fortemente stilizzato e fa utilizzo della tecnica di cel-shading che limita le tonalità dei colori in modo da avere aree a tinta unita e imitare le immagini di albi a fumetti. Tuttavia, alcuni effetti grafici non rispecchiano fedelmente questo stile. Un esempio è dato dalle nubi molto basse, che vengono renderizzate utilizzando un approccio volumetrico per permettere ai giocatori di attraversarle. Queste nubi hanno un aspetto che tende al realismo fotografico e si discosta molto dalla resa visiva delle nubi più alte e distanti, che presentano al contrario uno stile simile ad un dipinto a mano.

Politica di decisione. Come già accennato, questo sistema meteo è dinamico e presenta dunque una selezione del meteo casuale che varia nel tempo, fatta eccezione per determinati luoghi o necessità narrative che potrebbero sovrascrivere le condizioni attuali.

Figura 2.6: Confronto tra le nubi basse volumetriche e le nubi più alte di Genshin Impact.

Alcune condizioni climatiche dette "condizioni speciali" possono verificarsi in maniera limitata a zone predefinite del mondo. Ad esempio, ne fanno parte la neve e la nebbia.

Altre aree sono soggette inoltre a cambiamenti ambientali unici che non fanno distinzione tra giorno e notte. Nel caso in cui questi cambiamenti siano pilotati dalla trama di gioco, il sistema diventa locale alla macchina che sta eseguendo il videogioco, mentre la versione "online" continuerà la sua esecuzione per tutti gli altri giocatori.

Nel sistema meteo descritto è presente dunque una limitazione al realismo delle occorrenze meteorologiche: nonostante le condizioni siano generate casualmente nel tempo, la maggior parte dei cambi avranno luogo nel momento in cui si passa da un'area di gioco ad un altra, anche viaggiando a bassa velocità.

In generale, le politiche di decisione delle condizioni di Genshin Impact sembrano avere l'obiettivo di garantire una base di realismo, data dalla generazione casuale e dal legame tra area e pool di possibili condizioni; legame che dipende dalla temperatura media o dai luoghi reali a cui le aree sono ispirate. In più, il

sistema è diretto artisticamente per offrire al giocatore un'esperienza più completa, forzando le condizioni per aderire ad esigenze di trama. Si tratta dunque di un buon trade-off tra realismo e direzione artistica.

Transizioni. Per quanto riguarda le transizioni tra due condizioni meteo all'interno della stessa area (quindi determinata dal sistema di generazione casuale), è possibile notare cambiamenti graduali e realistici.

Come già visto, però, la maggior parte dei cambi di condizioni meteorologiche si possono osservare nel passaggio tra due aree di gioco.
Anche in questo caso, la transizione è abbastanza graduale, ma diversi giocatori hanno fatto notare che muoversi tra due zone in maniera sufficientemente rapida e alternata causa frequenti cambi di condizioni altrettanto rapide, rendendo il sistema di meteo un po' meno convincente.

Questo fenomeno può aver luogo soprattutto nelle aree in cui il meteo è impostato in maniera predefinita.

Inoltre, non è possibile avvistare a distanza condizioni meteo diverse da quella attuale. Anche se ci si trova a pochi metri da una zona che presenta condizioni differenti, solo il meteo alla posizione corrente viene renderizzato.

Impatto sul game-play. Il sistema meteo di Genshin Impact ha svariati effetti chiari e importanti sul game-play. Ogni condizione meteorologica, infatti, applica una serie di proprietà al mondo virtuale e al personaggio che ne influenzano il comportamento ma che possono anche rappresentare vantaggi e svantaggi. Le proprietà sono inoltre combinabili e possono portare ad ulteriori stati.

Durante la pioggia, la proprietà di bagnato si applica a tutti i personaggi, tra cui anche NPC. I falò e le torce si spengono, e alcuni oggetti normalmente infiammabili non possono prendere fuoco. Anche la probabilità di incontrare animali selvatici è azzerata.

Se la pioggia è temporalesca, sono implementati fulmini nuvola-terra che possono colpire personaggi e NPC infliggendo loro danni. Gli altri tipi di fulmini sono solo rappresentati da luci intense di breve durata.

Anche durante le nevicate, i personaggi possono subire danni da freddo se non soddisfano determinati requisiti di gioco.

La nebbia ha l'effetto di ridurre la visibilità, ed è anche un effetto utilizzato

come espediente per limitare fisicamente il mondo di gioco. La limitazione dipende anche dal progresso del giocatore; in questi casi l'effetto risulta essere solo locale.

2.4 Ashes of creation

Ashes of creation è un MMORPG in imminente uscita il cui mondo di gioco fantasy è dinamico e fortemente correlato alle azioni dei giocatori, che lo plasmano attraverso esplorazioni, commercio e costruzione. Uno dei fattori che rendono questo videogioco variabile è proprio il sistema di meteo, che implementa una vastissima varietà di condizioni ed effetti sul game-play.

Funzionamento generale. Il mondo di gioco è organizzato in Biomi. Una parte di essi è condizionata da forti cambiamenti visuali causati dal ciclo stagionali; la restante parte è invece caratterizzata da condizioni statiche.
Ogni stagione dura approssimativamente una o due settimane reali, alterando il paesaggio naturale dell'ambientazione e la sua fauna.
Il ciclo diurno si ripete circa ogni due ore, dunque il tempo è accelerato di 12 volte. Durante la notte le costellazioni e la luna ruotano; questo è un segno di un algoritmo non affatto banale per la rappresentazione della volta celeste. Tra le condizioni meteorologiche implementate vi sono vento, pioggia, neve, fulmini, nebbia.

Politica di decisione. è importante notare che in questo videogioco i cambi meteorologici e delle stagioni non sono solo legati al tempo ma anche ad eventi specifici che accadono sul server a causa dei giocatori. Questi possono essere sia locali, quindi legati al game-play del singolo giocatore se emergono necessità artistiche e narrative, ma anche regionali e globali.

Un esempio del primo caso è il seguente: un personaggio del videogioco dotato di poteri particolari blocca una specifica stagione; il ciclo viene interrotto fino a che quel personaggio non verrà sconfitto.

Per quanto riguarda i cambiamenti più vasti, tra le cause vi sono piaghe, eruzioni vulcaniche e altri disastri naturali.

Transizioni. In Ashes of Creation la transizione tra diverse condizioni è graduale, sia per quanto riguarda gli eventi meteorologici, sia per quanto riguarda le stagioni. Anche l'aspetto di alcuni animali varia con il meteo, e anche queste variazioni risultano lente e non improvvise.

Tuttavia, la transizione diventa prevedibile e leggermente forzata nel momento in cui si accede ad una zona caratterizzata da un meteo statico. Per arginare questa problematica, l'accesso alla regione presenta un aspetto intermedio abbastanza vasto da non avere un cambiamento improvviso delle condizioni meteorologiche. Dunque viene introdotto lo stesso approccio di "blending" già visto nelle applicazioni precedenti, ma questa volta anche spaziale e non solo temporale.

Impatto sul game-play. Una delle core mechanics di questo videogioco è proprio l'influenza bidirezionale tra gli eventi di gioco e gli eventi del giocatore: il meteo gioca un ruolo molto importante in questo contesto. Di seguito si elencano alcuni degli effetti che il clima può avere sul game-play:

1. Determinate abilità dei giocatori possono subire un potenziamento o un indebolimento in base all'entità della condizione meteorologica; anche la resistenza dei giocatori ne viene intaccata.
2. Le piante crescono solo in stagioni a loro favorevoli: di conseguenza, il valore di queste risorse potrebbe variare in base alla disponibilità.
3. Alcuni percorsi potrebbero essere bloccati a causa della neve o a causa di alberi caduti, questo influenza a sua volta la possibilità di accedere alle aree di gioco ma anche il tempo necessario per il trasporto delle risorse.
4. Il vento influenza la velocità e la direzione di viaggio nel mare aperto; i giocatori ne devono tenere conto usando strumenti di navigazione.

Come è intuibile, Ashes of Creation presenta la maggiore varietà di interazioni con il meteo del mondo virtuale, proprio perché questo aspetto fa parte delle meccaniche su cui è basato.

2.5 The Legend of Zelda: Breath of the Wild

The Legend of Zelda: Breath of the Wild è un action game basato sull'avventura e l'esplorazione di un open-world, ed è considerato un capolavoro della storia video-ludica.

Funzionamento generale. Il meteo nel mondo virtuale di Hyrule è fortemente condizionato dalla regione in cui ci si trova. Infatti, sono implementate 20 zone climatiche; ognuna di queste presenta caratteristiche ambientali diverse e quindi diverse probabilità per ogni occorrenza meteorologiche. Inoltre, il videogioco tiene traccia della temperatura dell'aria, e anche questo parametro può influenzare le condizioni.

Politica di decisione. Le condizioni meteorologiche cambiano in cicli di 4 minuti reali (4 ore in gioco). Per ogni zona climatica sono presenti 5 valori fissati di probabilità che un determinato evento atmosferico avvenga.

Nonostante la poca quantità di informazioni tecniche reperibili sull'argomento, diversi giocatori hanno analizzato il funzionamento del sistema, e ne hanno riportato i dati. Tra questi, una tabella che riporta le probabilità per ogni zona, nonché la mappa di Hyrule divisa per zone climatiche.

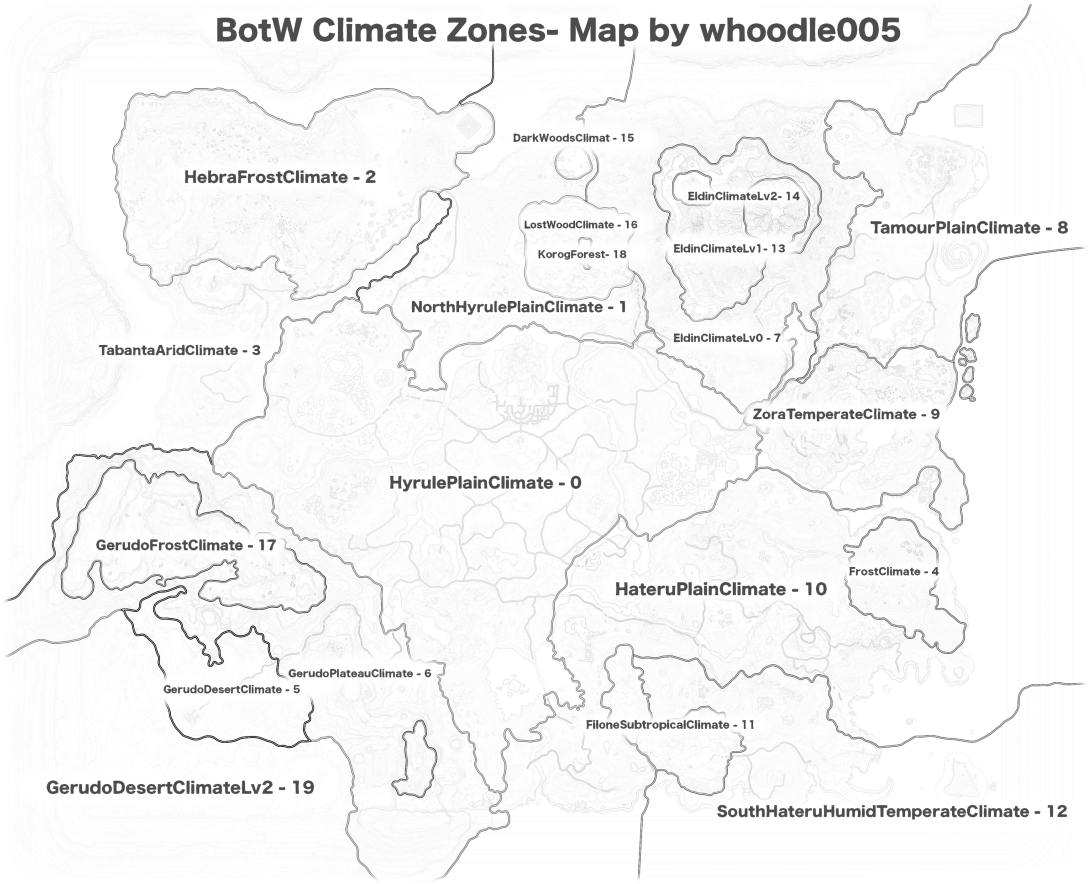

Figura 2.7: Mappatura delle zone climatiche in BOTW.

Le probabilità per ogni zona sembrano inoltre impostate con criterio scientifico: un giocatore ha provato a calcolare le dimensioni del mondo in metri del mondo per poi individuare le zone di alta e bassa pressione basandosi sull'altitudine e sulla temperatura media. Partendo da questa "mappa termica", ha individuato i punti della mappa in cui le probabilità di precipitazione erano più alte. I dati ottenuti

in questo modo coincidono approssimativamente con i dati sperimentali raccolti durante il gioco e quelli riportati nella tabella, confermando la realizzazione ben curata e precisa di questo sistema di meteo.

	Bluesky	Cloudy	Rain	Heavyrain	Storm
HyrulePlainClimate - 0	60	20	10	5	5
NorthHyrulePlainClimate - 1	40	30	15	5	10
HebraFrostClimate - 2	20	10	35	35	0
TabantaAridClimate - 3	94	4	2	0	0
FrostClimate - 4	10	20	20	50	0
GerudoDesertClimate - 5	100	0	0	0	0
GerudoPlateauClimate - 6	95	5	0	0	0
EldinClimateLv0 - 7	100	0	0	0	0
TamourPlainClimate - 8	35	40	10	5	10
ZoraTemperateClimate - 9	50	20	20	10	0
HateruPlainClimate - 10	50	35	10	5	0
FiloneSubtropicalClimate - 11	35	35	10	10	10
SouthHateruHumidTemperateClimate - 12	65	5	15	15	0
EldinClimateLv1- 13	100	0	0	0	0
EldinClimateLv2- 14	100	0	0	0	0
DarkWoodsClimat - 15	40	30	15	5	10
LostWoodClimate - 16	100	0	0	0	0
GerudoFrostClimate - 17	5	20	40	35	0
KorogForest- 18	100	0	0	0	0
GerudoDesertClimateLv2 - 19	100	0	0	0	0

Figura 2.8: Look-up table del sistema meteo di BOTW.

Inoltre, come è possibile notare dalla tabella, ci sono alcune zone che presentano un clima fisso, ovvero il 100 percento di probabilità di una delle occorrenze meteorologiche. Questo accade per motivi artistici, come nel caso delle Dark Woods, o per modellare zone desertiche, come nel Gerudo Desert.

Anche la neve, che non è presente nella tabella, è implementata e coincide con la pioggia: l'entità delle precipitazioni dipende dalla temperatura. A sua volta, la temperatura è funzione dell'ora del giorno, dell'altitudine e della zona climatica.

I fulmini sono generati in punti casuali durante le forti precipitazioni. Se nei dintorni del giocatore sono presenti oggetti metallici, questi attireranno i fulmini su di loro. Come vedremo, questa meccanica è interessante in quanto controllabile da un giocatore consapevole.

Ad ogni zona è inoltre associata una potenza del vento, che potrebbe essere frutto di uno studio sulle differenze di pressione della mappa.

Si noti che questo sistema è studiato ad-hoc per la mappa di Hyrule, quindi non sarebbe automaticamente adattabile ad un altro mondo virtuale. In altre parole, per realizzarne uno simile bisognerebbe conoscere a priori la struttura e la natura del mondo di gioco.

Transizioni. Anche in questo caso, le transizioni sono implementate come blending dei vari parametri atmosferici.

Data la presenza di zone a clima fisso, a volte questo evento può risultare forzato durante l'accesso a determinate aree.
La problematica viene alleviata diluendo molto i tempi di transizione: diversi giocatori hanno notato che l'indicatore meteo dell'interfaccia non corrispondesse all'effettiva evento atmosferico in corso per qualche minuto, segno del fatto che i cambiamenti climatici occupano un lasso di tempo non molto breve.

Impatto sul game-play. In questo videogioco, ogni condizione climatica diversa da "Bluesky", ovvero il cielo sereno, presenta una o più conseguenze sul mondo di gioco che a sua volta influiscono sul game-play. Si analizzano dunque di seguito:

1. La pioggia rende le superfici bagnate e spegne i fuochi, se esposti. Mentre piove, il giocatore troverà difficoltà nell'arrampicarsi sulle pareti, scivolando e consumando più stamina.

2. La neve rende i pavimenti scivolosi, permettendo al giocatore di far rotolare oggetti o rendere più efficaci alcune azioni, tra cui usare uno scudo come tavola da surf.

3. Durante le tempeste, i fulmini possono colpire il giocatore casualmente o se indossa indumenti di metallo. I fulmini incendiano l'erba, se presente, e l'incendio crea a sua volta una corrente ascensionale che può interagire con il giocatore.

4. Anche la temperatura rappresenta un parametro importante durante l'esperienza di gioco. Infatti, il calore o il freddo estremo possono danneggiare il giocatore a meno dell'utilizzo di oggetti specifici. In alcuni luoghi, gli oggetti di legno prendono fuoco spontaneamente.

5. Raffiche di vento casuali possono spingere il giocatore mentre è in volo, avvantaggiandolo o svantaggiandolo a seconda della direzione di viaggio.

2.6 Considerazioni sullo stato dell'arte

In figura 2.9 sono riassunte le caratteristiche principali dei sistemi meteo analizzati.

	POLITICA DI DECISIONE	DOMINIO DI SIMULAZIONE	STILE GRAFICO	IMPATTO SUL GAMEPLAY
SHEMUE	Casuale / Dataset	Ciclo temporale	Skybox (PR)	Comportamento NPC
GTA	FIFO statica	Ciclo temporale	Skybox (PR)	Comportamento NPC, sistema fisico, visibilità
GENSHIN IMPACT	Casuale	Area	Skybox (NPR), Volumetrics (PR)	Meccaniche
ASHES OF CREATION	Event-based	Area	Volumetrics (PR)	Comportamento NPC, mondo di gioco, meccaniche
THE LEGEND OF ZELDA: BOTW	Physics-based	Ciclo temporale, Area	Skybox procedurale (NPR)	Comportamento NPC, sistema fisico, meccaniche

Figura 2.9: Riepilogo delle caratteristiche principali dei sistemi meteo analizzati.

Tra le simulazioni meteorologiche che garantiscono più accuratezza e, quindi, più immersione, spicca l'approccio di The Legend Of Zelda: Breath Of the Wild, che permette anche ottime prestazioni se si considera che ogni cambio consiste nell'accesso ad una tabella di dimensioni relativamente piccole e la generazione di un numero pseudo-casuale. Gli unici difetti di questo sistema risiedono nella necessità di suddividere il mondo di gioco in aree, che costituiscono inoltre un limite alla sua dinamicità a causa di transizioni forzate. Inoltre, il mondo deve essere noto a priori in modo da adattare al meglio la suddivisione.
Nello stesso contesto, risultano altrettanto interessanti le tecniche di Shenmue, che può attingere da dati del mondo reale, e Ashes of Creation, che potenzia l'interattività della simulazione rendendo il sistema sensibile alle azioni dei giocatori.

La limitazione del dominio ad aree e cicli temporali non permette nella maggior parte dei casi di poter avvistare a distanza condizioni meteo diverse da quelle correnti, oltre a rendere le transizioni prevedibili, per quanto graduali le si possano renderizzare.

Per quanto riguarda lo stile grafico, si può notare che nella maggior parte dei casi è coeso e coerente con quello del resto dell'esperienza. Tuttavia, le nuvole della volta celeste non risultano sempre visivamente variabili nei casi in cui si utilizzano texture 2D scorrevoli come skybox.

Infine, ognuno dei videogiochi analizzati presenta alcuni effetti diretti delle condizioni meteorologiche sull'esperienza di gioco. Degno di nota è lo sforzo degli sviluppatori di Ashes of Creation, che aggiungono ulteriore senso di immersione introducendo variazioni innovative, che a loro volta si ripercuotono sul sistema economico interno al videogioco.

Capitolo 3

Implementazione

3.1 Ciclo diurno

Per realizzare un sistema di meteo dinamico, risulta fondamentale e propedeutica l'implementazione di un modello del ciclo diurno, che fornirà dati importanti quali la fase dei moti di rotazione e rivoluzione, da cui ricavare, per esempio, l'incidenza dei raggi solari sulla superficie del mondo di gioco.
Per modellare il ciclo diurno, sono ovviamente necessari i dati delle coordinate sul globo del centro del mondo virtuale, nonché informazioni sulla data della simulazione: in breve, è necessario stabilire le coordinate spazio-temporali del mondo di gioco utilizzando i sistemi convenzionali terrestri.
Per farlo, la presente implementazione espone dei campi di input sull'editor di Unity, che permettono di:

1. Impostare la data della simulazione (anno, giorno, ora) secondo il fuso orario convenzionale UTC;

2. Impostare le coordinate geografiche della simulazione (longitudine, latitudine);

3. Impostare un moltiplicatore per lo scorrere del tempo, in modo da avere controllo sulla velocità della simulazione e testare facilmente le diverse politiche algoritmiche su una finestra temporale maggiore.

Di seguito, si elencano inoltre gli obiettivi del modulo del ciclo diurno:

1. Calcolare la posizione del Sole e della Luna;

2. Considerare effetti ottici di curvatura dei raggi luminosi, tra cui l'effetto di parallasse parallasse e la rifrazione atmosferica.

3. Garantire un buon compromesso tra accuratezza e prestazioni.

I corpi celesti sono modellati da Unity come luci direzionali, ovvero fonti di luce a distanza infinita, i cui raggi paralleli raggiungono il mondo di gioco sempre con la stessa inclinazione. L'approssimazione è opportuna, se si pensa che anche i mondi virtuali più vasti hanno dimensioni trascurabili se rapportati a quelle del globo. Dunque, il calcolo della posizione di queste fonti di luce si traduce nel calcolo della loro direzione.

Fasi del moto terrestre. Per individuare la posizione del Sole, è necessario definire alcuni concetti e parametri caratteristici.

1. I meridiani sono curve chiuse immaginarie che si ottengono dall'intersezione tra la superficie terrestre e piani passati per i due poli, mentre i paralleli sono ottenute intersecando la superficie terrestre con i piani perpendicolari all'asse di rotazione terrestre. Il parallelo equidistante dai due poli è detto equatore.

2. La latitudine ϕ è l'angolo formato dal piano dell'equatore e dalla congiungente il punto di osservazione con il centro della terra. Il range dell'angolo è [-90, 90] in gradi dal polo sud al polo nord, l'angolo nullo corrisponde al parallelo dell'equatore.
   Questo angolo è fornito in input dall'utente.

3. L'orbita della Terra intorno al Sole descrive un'ellisse: il suo asse di rotazione rimane sempre parallelo a sè stesso durante questo processo, ed è inclinato di 23.45 gradi rispetto alla verticale della traiettoria.

4. La declinazione solare δ viene definita in base alla Norma UNI 8477-1 come l'angolo che la retta tracciata dal centro della Terra al Sole forma con il piano equatoriale nel piano contenente gli assi dei due corpi celesti. Rappresenta l'informazione della fase del moto di rivoluzione. Può essere vista come la latitudine proiettata sul Sole.

5. L'angolo orario ω è la distanza angolare tra un meridiano di riferimento e il meridiano passante per il Sole. Rappresenta l'informazione della fase del moto rotazione della Terra intorno al proprio asse.

Per calcolare l'angolo di declinazione solare esiste una relazione empirica approssimata basata sul lavoro di P.I. Cooper, 1969. [1]

$$\delta[\text{deg}] = 23{,}45^\circ \sin 2\pi \frac{284 + n}{365} \tag{3.1}$$

Dove:

1. n è il giorno dell'anno, ed è facilmente ottenibile a partire dai dati di input esposti all'utente.
2. 23,45° è l'inclinazione dell'asse terrestre.

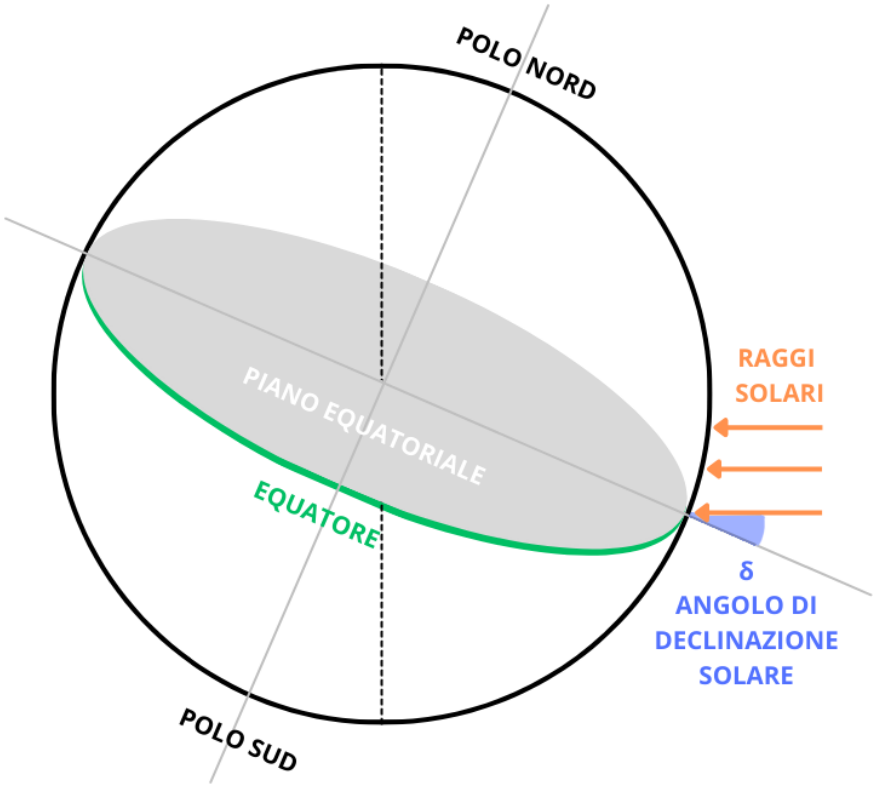

Figura 3.1: Angolo di declinazione solare.

Grazie a questo dato, è possibile stabilire la fase del moto della Terra attorno al Sole.
Calcolare l'angolo orario richiede di fare riferimento all'ora solare, ricavabile a partire dall'ora standard UTC, riferita al meridiano fondamentale che passa per l'osservatorio di Greenwitch a Londra. Si applica dunque una correzione per tenere conto della differenza di longitudine dal meridiano di riferimento a quello fondamentale pari a 4 minuti per ogni grado di longitudine (1 ora ogni 15 gradi).

Il parametro della longitudine è memorizzato come offset rispetto alla longitudine 0 del meridiano fondamentale, quindi basta eseguire:

$$\omega[\text{deg}] = UTC[h] * 15 + longitude[\text{deg}] \tag{3.2}$$

Inoltre, esiste una seconda correzione che considera le perturbazioni dell'orbita terrestre e della velocità di rotazione. Questo termine è noto come equazione del tempo ed è la somma di due curve approssimabile come:

$$\begin{aligned} ET[h] = 0.0072\cos J - 0.0528\cos 2J - 0.0012\cos 3J + \\ -0.1229\sin J - 0.1565\sin 2J - 0.0041\sin 3J \end{aligned} \tag{3.3}$$

Dove:

1. $J = 2\pi(n/365)$ è l'inclinazione dell'asse terrestre.
2. n è il giorno dell'anno

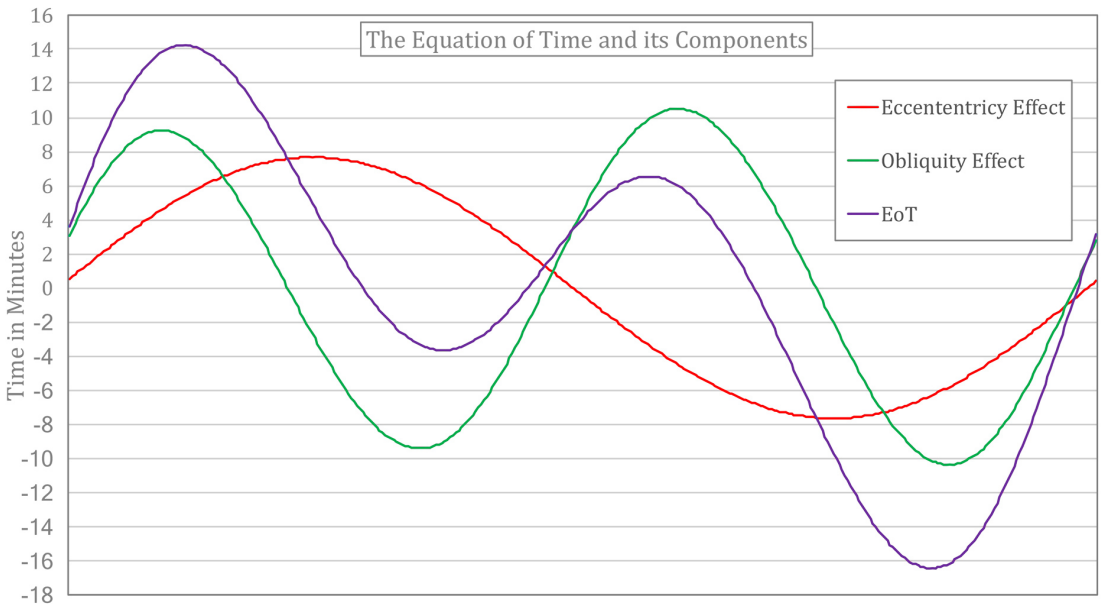

Figura 3.2: Componenti dell'equazione del tempo.

L'angolo orario risulta dunque:

$$\omega[\text{deg}] = (UTC[h] + ET[h]) * 15 + longitude[\text{deg}] \tag{3.4}$$

Calcolo della posizione del Sole. Dopo aver ottenuto l'angolo di declinazione solare e l'angolo orario, è possibile ricavare la posizione del Sole rispetto ad un determinato punto sulla Terra, e lo si fa descrivendola con due ulteriori angoli, proprio come per ogni altro corpo celeste.

1. L'angolo di elevazione solare: angolo tra il vettore diretto verso il Sole e il piano orizzontale (il tutto riferito alla posizione di osservazione). Questo angolo è massimo quando il Sole è allo "zenith" (ovvero giace sulla verticale dell'osservatore), ed è importante perchè determina con quale inclinazione i raggi solari raggiungono la superficie terrestre.

2. L'angolo azimutale: angolo orizzontale tra il piano verticale passante per il Sole e la direzione nord. Al contrario, questo angolo non incide sull'inclinazione con la quale i raggi solari raggiungono la superficie terrestre.

"Algoritmi Astronomici" [2] fornisce dei metodi algoritmici per il calcolo delle coordinate solari. Utilizzando queste formule, semplificate opportunamente, si ha che:

$$elevationAngle = \arcsin(\sin\delta * latitude + \cos\delta * \omega * \cos latitude) \tag{3.5}$$

e

$$azimuthAngle = \arcsin(\cos\delta * \sin\omega / \cos elevationAngle) \tag{3.6}$$

In Unity, associando l'asse z positivo con la direzione del nord e l'asse x positivo con l'est, basta assegnare l'angolo di elevazione come rotazione rispetto all'asse x e l'angolo di azimuth rispetto all'asse y.

Calcolo della posizione della Luna. Utilizzando ancora le formule di [2], è possibile ricavare alcuni termini periodici necessari per calcolare la posizione accurata della Luna in un dato istante. Sarebbero necessarie in realtà centinaia di termini

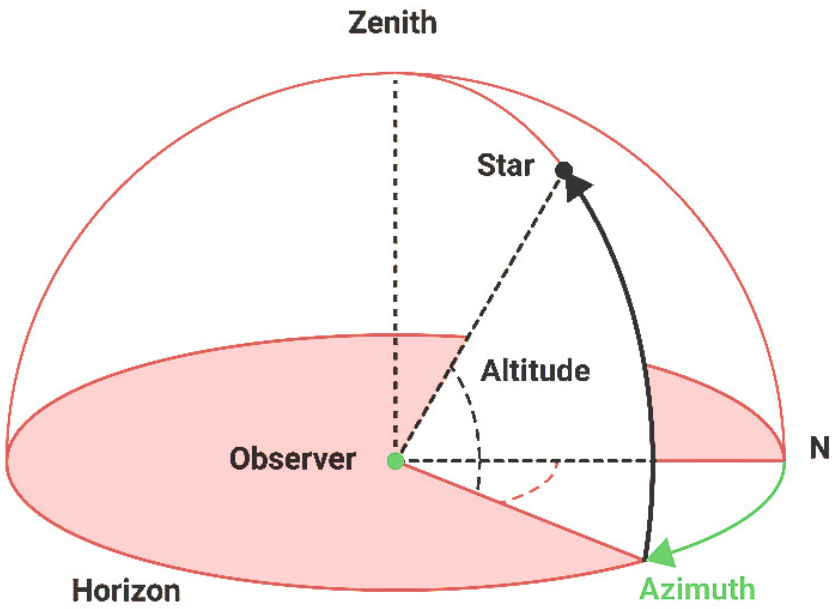

Figura 3.3: Angoli di elevazione e azimutale.

periodici, ma limitare la selezione ai pochi più determinanti permetterà di ottenere delle prestazioni compatibili con un'applicazione real-time ed un'accuratezza di 10" in longitudine e 4" in latitudine. Vi sono ulteriori semplificazioni, ad esempio relative al calcolo dell'età della Luna che è basato sulla durata media del mese sinodico (tempo medio tra due fasi lunari), che abbassano l'accuratezza del calcolo fino ad un errore massimo di circa due ore. Seguono dunque una serie di formule standard per ricavare le coordinate della Luna:

1. Formula standard per il calcolo dell'anno Giuliano, ovvero un contatore di giorni a partire dall'anno -4712. Se il mese è precedente a Febbraio compreso, il parametro relativo al mese viene incrementato di 12 e l'anno diminuito di 1.

$$JD = ((365.25 * year) + (30.6001 * (month + 1)) - 15 + 1720996.5 + \\ + dayOfYear + TimeOfDay.TotalHours/24.0) \quad (3.7)$$

2. Calcolo della fase lunare basato sulla durata del mese sinodico e convertita in radianti:

$$moonAge = ((JD - 2451550.1)/29.530588853) * 2\pi \quad (3.8)$$

3. Formula empirica per ricavare la distanza della Luna in unità di raggio terrestre. É basata sulle osservazioni della posizione della Luna ed è stata rifinita negli anni da astronomi e scienziati:

$$dp = 2\pi * (JD - 2451562.2)/27.55454988 \tag{3.9}$$

$$moonDistance = 60.4 - 3.3 * \cos dp - 0.6* \\ \cos(2 * moonAge - dp) - 0.5 * \cos 2 * moonAge \tag{3.10}$$

4. Formule per il calcolo della latitudine dell'eclittica lunare, basata sull'inclinazione dell'orbita lunare rispetto a quella della Terra; e per il calcolo della longitudine dell'eclittica lunare, basata sulla posizione relativa della Luna rispetto all'equinozio di primavera. Hanno un'accuratezza di circa 2 gradi:

$$np = 2\pi * (JD - 2451565.2)/27.212220817; \tag{3.11}$$

$$moonLatitude = 5.1 * \sin np \tag{3.12}$$

$$rp = (JD - 2451555.8)/27.321582241 \tag{3.13}$$

$$moonLongitude = 360 * rp + 6.3 * \sin dp + 1.3 \\ * \sin(2 * moonAge - dp) + 0.7 * \sin(2 * moonAge) \tag{3.14}$$

5. Conversione dell'anno giuliano in una scala temporale nota come "Terrestrial Time", che è un sistema di tempo basato sulla rotazione della Terra ed è spesso utilizzato in astronomia e geodesia. Il valore di 2451545.0 nella formula rappresenta il Julian Day corrispondente al 1 ° gennaio 2000 a mezzanotte nello standard UTC, che è la data di riferimento utilizzata come punto di partenza per la scala di tempo TT. Si divide quindi per il numero di giorni in un secolo tropico.

$$T = (JD - 2451545.0)/36525 \tag{3.15}$$

6. Si ricava poi l'obliquità dell'eclittica terrestre basandosi sul modello della precessione di Newcomb, che è un modello matematico dell'oscillazione dell'asse terrestre nel tempo.

$$eps = 23.0 + 26.0/60.0 + 21.448/3600.0+ \\ - (46.8150 * T + 0.00059 * T^2 - 0.001813 * T^3)/3600; \tag{3.16}$$

7. In funzione delle coordinate eclettiche e dell'obliquità dell'eclettica terrestre, è possibile calcolare la distanza dal polo nord celeste alla posizione della Luna r e la distanza angolare della Luna dall'equatore celeste z:

$$z = \sin eps * \cos moonLatitude * \sin moonLongitude + \\ + \cos eps * \sin moonLatitude; \tag{3.17}$$

$$r = \sqrt{1 - z^2} \tag{3.18}$$

L'arcotangente di z/r è il valore dell'angolo opposto al lato z di un triangolo rettangolo in cui l'altro lato r è l'ipotenusa. L'angolo calcolato rappresenta l'angolo di declinazione della Luna rispetto all'equatore celeste.

$$moonDeclinationAngle = \arctan \frac{z}{r} \tag{3.19}$$

A questo punto è necessario l'angolo orario della Luna per ottenere la rotazione complessiva da assegnare alla luce direzionale in Unity. Se l'osservatore si trova meridiano fondamentale a mezzogiorno, l'angolo orario vale:

$$moonRightAscension = \arctan((\sin(moonLongitude) * \cos(eps) - \\ \tan(moonLatitude) * \sin(eps))/\cos(moonLongitude)) \tag{3.20}$$

ed è detto anche Ascensione Retta della Luna. Per correggere il valore tenendo conto della longitudine dell'osservatore e dell'ora del giorno bisogna calcolare la rotazione della Terra intorno al suo asse rispetto alla posizione del Sole in equinozio medio, espressa come angolo rispetto al meridiano di riferimento.

$$thetaLocal = \frac{\pi}{180} * 280.46061837 + 360.98564736629 * (julianDay - 2451545.0) + \\ + 0.000387933 * T^2 - T^3/38710000.0) + longitude \tag{3.21}$$

L'angolo orario diventa dunque:

$$moonHourAngle = thetaLocal - moonRightAscension \tag{3.22}$$

Utilizzando 3.5 e 3.6 è possibile infine ottenere gli angoli di rotazione x e y della Luna.

3.2 Nubi

Le nubi rappresentano un fondamentale contributo sia all'aspetto estetico di una scena all'aperto, sia alla contestualizzazione delle occorrenze metereologiche: il loro

comportamento può permettere al giocatore non solo di comprendere le attuali condizioni, ma anche di prevederne l'evoluzione.

L'obiettivo di questa sezione è raccogliere informazioni per selezionare una tecnica di rendering in grado di soddisfare i seguenti requisiti principali:

1. buone prestazioni per il real-time (> 30 fotogrammi per secondo),
2. supporto per un sistema di meteo dinamico,
3. un'esperienza immersiva e realistica,
4. una resa visiva regolabile grazie ad alcuni parametri per adattarsi a stili grafici diversi.

Si definisce nube una qualsiasi idrometeora di particelle di vapore acqueo e cristalli di ghiaccio sospeso nell'atmosfera di un corpo celeste che ne è dotato.
Alcuni tipi di nube possono causare precipitazioni: la loro rappresentazione dovrà essere strettamente legata al meccanismo di occorrenza delle condizioni metereologiche.

In generale, l'aspetto di una nube ha una forma caratterizzata da auto similarità; il suo colore è risultato dei fenomeni ottici che avvengono al suo interno.

Prima di discutere le tecniche di rendering, è necessario analizzare le caratteristiche di tutti i tipi di nubi e come queste specificano il loro aspetto e il loro comportamento.

Le nubi sono presenti nell'atmosfera terrestre dal suolo fino a circa 14.000 metri di altitudine: una classificazione prevede di distinguerle in base alla distanza dal suolo.
Un'ulteriore classificazione si basa invece sullo spessore e la formazione dell'idrometeora. Esistono tuttavia due tipi di nubi a sviluppo verticale che verranno inserite in un altra categoria.

Nubi alte, nessuna precipitazione. Si tratta delle nubi più fredde e ad alta concentrazione di cristalli di ghiaccio, che le rende traslucide e di un colore generalmente chiaro, a meno di fenomeni esterni. A seconda della forma, possono essere:

1. Cirri, filamenti che anticipano un fronte caldo. Possono evolversi in altostrati.
2. Cirrocumuli, con forma più globulare. Indicano stabilità.

3. Cirrostrati, stratiformi. Preannunciano instabilità.

Nubi medie, precipitazioni deboli. Si distinguono in:

1. Altocumuli, con forma globulare. Si formano con umidità elevata e venti deboli.
2. Altostrati, nubi sottili che filtrano la luce e sono dunque di colore grigio o azzurro. Sono più frequenti ai poli e possono provocare precipitazioni deboli.
3. Nembostrati, nubi grigie a forte estensione orizzontale che coprono il cielo e possono portare a lunghe precipitazioni di pioggia o neve.

Nubi basse, precipitazioni deboli o intense. Sono composte prevalentemente da particelle d'acqua. Quando entrano in contatto con il suolo, si parla di nebbia. Le nubi basse sono ulteriormente classificate in:

1. Cumuli, causati dalle correnti convettive. Hanno una sommità leggermente convessa e una forma "a cavolfiore". Possono portare anche a manifestazioni temporalesche.
2. Stratocumuli, simili agli strati ma caratterizzata da addensamenti tondeggianti. Le precipitazioni sono possibili ma ancora in forma leggera.

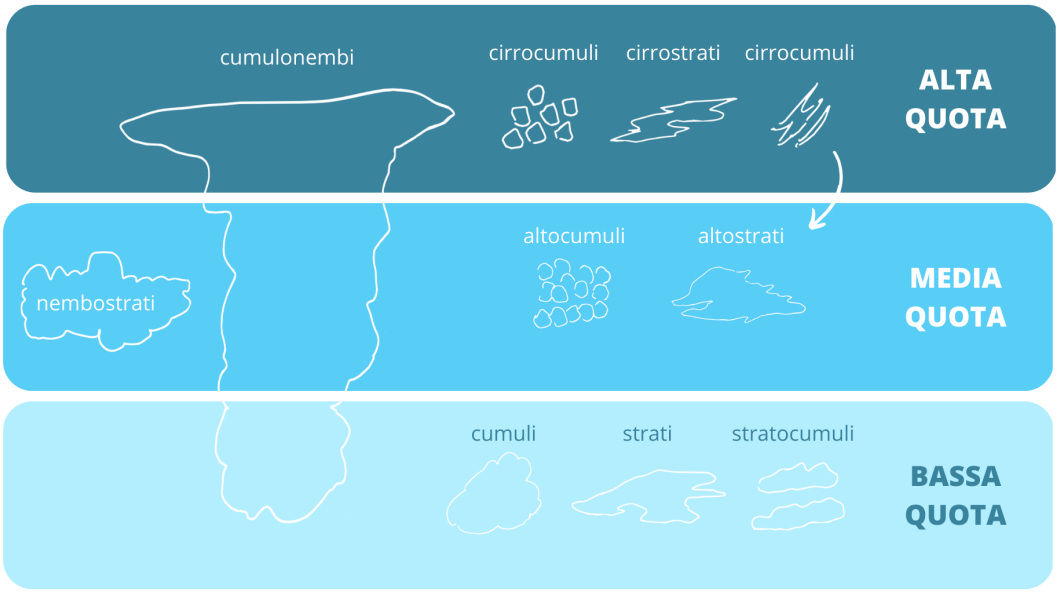

Figura 3.4: Tipi di nubi.

3. Strati, a forte sviluppo orizzontale. Possono dar luogo a precipitazioni generalmente molto deboli.

Nubi a sviluppo verticale, precipitazioni intense. Possono raggiungere dimensioni di ordine di grandezza molto maggiore dei tipi precedenti di nube. Sono generalmente associate a violente precipitazioni e temporali. La loro forma più grande è definita cumulonembo, e si forma quando una corrente ascensionale alimenta continuamente un cumulo. Presentano una forma a incudine sulla sommità, ma dal suolo solo la base della nube è visibile: se non ne si considera il comportamento, è visivamente indistinguibile da un nembostrato.

3.3 Rendering delle nubi

Dopo aver analizzato e classificato le nubi, si hanno molte più informazioni per decidere come, dove e quando rappresentarle in un ambiente virtuale.
Esistono infatti tre tecniche principali, elencate di seguito, per renderizzare dei gas:

1. Texture 2D. Si utilizzano texture mappate su una superficie, tipicamente una semisfera che racchiude il mondo di gioco. In Unity sono implementate dal sistema di "skybox".

   (a) Pro:

      i. Semplicità di implementazione. Come già accennato, i software di authoring mettono a disposizione degli strumenti per realizzare un cielo partendo da una o più texture ambientali.

      ii. Semplicità computazionale.

   (b) Contra:

      i. Resa grafica piatta.

      ii. Impossibilità di illuminare le nubi in maniera tridimensionale.

2. Mesh. Consiste nella realizzazione di un modello 3D standard la cui forma somigli alla superficie di una nube.

   (a) Pro:

      i. Semplicità di implementazione. É possibile utilizzare un modello di illuminazione standard delle superfici e i fenomeni di scattering della luce sono ampiamente stati discussi e implementati per le superfici.

      ii. Resa grafica stilizzata. Molti stili grafici NPR prevedono la rappresentazione di nubi come forme nette piene di colore, ovvero senza trasparenze o bordi sfumati.

(b) Contra:

i. Complessità computazionale. Per generare nubi con un livello di dettaglio sufficiente sono necessarie tantissime mesh molto dense. Una possibile ottimizzazione consisterebbe nell'utilizzo intensivo di mappe delle normali.

ii. Resa grafica non adattabile. Non sarebbe possibile rappresentare nubi semi-trasparenti o dai bordi sfumati, come i cirri.

3. Volumetrics. Si basa sul ray marching: un algoritmo che permette di visualizzare texture tridimensionali con canale alpha nello spazio.

(a) Pro:

i. Resa grafica tridimensionale e versatile. É possibile rappresentare nubi dai bordi sfumati, ma anche forme dal contorno definito.

ii. Possibilità di costruire uno shader physics-based. Il volume permette di eseguire calcoli più accurati tenendo conto degli effetti ottici che avvengono all'interno della nube, garantendo un'esperienza più immersiva.

iii. Possibilità di attraversare le nubi. Anche se la camera è all'interno della nube, sono note tecniche per renderizzare profondità intermedie all'interno del volume.

(b) Contra:

i. Complessità computazionale. La renderizzazione di volumi è computazionalmente assimilabile alla renderizzazione di un vettore di piani semi-trasparenti. L'illuminazione, inoltre, aumenta esponenzialmente il numero di calcoli necessari a causa di un ciclo annidato.

ii. Complessità di implementazione. A differenza degli shader per le superfici (che sono ampiamente diffusi), sarebbe necessario implementare da zero uno shader volumetrico per ragioni di ottimizzazione e stilizzazione.

Scelte di implementazione. Per la presente implementazione, è stata selezionata la tecnica dei volumi per rappresentare tutti i tipi di nube eccetto quelle ad alta quota, per cui verranno invece utilizzate delle Texture 2D (Skybox) generate in maniera procedurale.

La prima scelta permetterà di raggiungere tutti gli obiettivi del sottosistema citati precedentemente, a costo delle prestazioni. Tuttavia, verranno discusse diverse ottimizzazioni, tra cui alcune esclusivamente sfruttabili da una grafica NPR, che porteranno il sistema ad un funzionamento real-time.

La seconda scelta è mirata all'ottimizzazione: le nubi alte sono abbastanza distanti dal suolo da poter essere approssimate a forme piatte. L'utilizzo di texture procedurali permetterà comunque versatilità grafica e di comportamento. Inoltre, queste nubi non generano precipitazioni e dunque condivideranno una quantità di dati minore con il sistema meteo.

Volumetric ray marching. Questa sezione descriverà l'implementazione di uno shader volumetrico tramite ray marching. Il riferimento principale utilizzato è una presentazione del sistema di rendering delle nubi del video-game Horizon Zero Dawn.
Seguiranno modifiche e ottimizzazioni apportate all'approccio originale volte a migliorarne le prestazioni e rendere lo stile grafico adattabile grazie a diversi parametri.

Prima di ragionare sull'algoritmo di shading, è necessario tenere a mente alcuni concetti che modellano il comportamento e la forma delle nubi. Essi sono riassumibili nei seguenti punti chiave:

1. La densità aumenta a basse temperature
2. La temperatura diminuisce con l'altitudine
3. Maggiore densità aumenta la probabilità di precipitazioni
4. La direzione del vento varia con l'altitudine
5. Le nubi si innalzano dal suolo con il calore
6. Le regioni più dense hanno una forma più tonda
7. La luce si diffonde all'interno delle nubi
8. La turbolenza atmosferica distorce ulteriormente le nubi

Questo modello sarà utile più volte per rendere lo shader physics-based e rappresentare nubi più realistiche.

L'approccio del ray marching consiste nel campionare una texture tridimensionale in scala di grigi nello spazio. Ogni campione rappresenta la densità del volume.

Si consideri una mesh 3D semplice, come un parallelepipedo.
Per ogni punto sulla superficie del solido, si considera la retta che parte dall'origine della telecamera e passa per il punto considerato. L'algoritmo campiona la texture nel punto considerato e la memorizza in una variabile density, avanza lungo la

direzione della retta di una quantità stepSize e accumula la densità campionata all'interno della stessa variabile. Questa operazione si ripete in un loop che si interrompe quando il punto corrente esce fuori dal solido.
Il seguente pseudocodice riassume dunque il funzionamento base di un algoritmo di ray marching: Questo algoritmo permette di visualizzare una texture 3D in termini

Algorithm 2 Pseudocodice del ray marching.

```
1: procedure RayMarching
2:     ▷ Initialization
3:     float density ← 0
4:     bool insideBox ← true
5:     float3 point ← position
6:     ▷ Execution
7:     while insideBox do
8:         density ← density + sample(3Dtexture, point)
9:         point ← direction * stepSize
10:        for int coord ← 0; coord < 3; coord++ do
11:            insideBox ← insideBox AND ABS(point[coord]) < bounds[coord]
12:        end for
13:    end while
14: end procedure
```

di densità in un ambiente virtuale. Tuttavia, la rappresentazione non reagisce ancora alla luce, in quanto non si stanno ancora tenendo in considerazione i dati ad essa associati.

Illuminazione delle nubi. L'illuminazione di volumi è un'area di ricerca abbastanza attiva nella computer grafica, e i migliori risultati sono quasi sempre dovuti a un alto numero di campioni.
I tre effetti grafici più importanti per rappresentare le nubi sono:

1. Diffusione direzionale
2. Silver Lining, dovuto alla anisotropia della diffusione
3. Bordi scuri, o "effetto zucchero a velo"

Quando un raggio luminoso entra in una nube, subisce molteplici effetti di rifrazione o combinazione con altri raggi rifratti prima di uscirne e raggiungere il nostro occhio. Questo effetto può essere riprodotto fedelmente tramite simulazioni, ma per applicazioni real-time è necessario applicare diverse approssimazioni.
La prima di queste è detta Legge di Beer, che permette di determinare la quantità

di luce che raggiunge un punto data lo spessore del mezzo in cui viaggia.

$$T = e^{-t} \tag{3.23}$$

Sostituendo l'energia con la trasmittanza e considerando lo spessore come la profondità del punto nella nube, si ottiene una base per il modello di illuminazione delle nubi.
Si noti come l'energia decrementi esponenzialmente con l'aumento della profondità.

Inoltre, c'è una percorso "privilegiato" dei raggi luminosi all'interno di una nube, e coincide con la direzione originale di incidenza del raggio. In altri termini, la probabilità che il raggio continui a diffondersi lungo la sua direzione di partenza è maggiore rispetto alla probabilità che cambi direzione.

Figura 3.5: Effetto della diffusione anisotropa della luce, o Silver Lining.

Questo fenomeno è chiamato diffusione anisotropa, e ne è stato sviluppato un modello matematico nel 1941 come supporto per calcoli di luce su scala astronomica. Il modello non è altro che una funzione di fase, detta Henyey-Greenstein, espressa nella forma seguente:

$$HG = \frac{1}{4\Pi} \ * (1 - g^2)/(1 + g^2 - 2g\mu)^{3/2} \tag{3.24}$$

Dove:

1. g esprime la dominanza statistica della direzione del raggio. Per g>0 si rappresenta la diffusione preferenziale in avanti, per g<0 si rappresenta la diffusione preferenziale nel verso opposto, mentre per g = 0 si ha una distribuzione non anisotropa, quindi isotropa.

2. $\mu = cos\theta = viewDir \cdot sunDir$
è il coseno dell'angolo compreso tra il vettore di vista e l'angolo di incidenza dei raggi solari.

Si moltiplica dunque ogni campione di energia per la questa funzione.

$$T = e^{-t} * HG \tag{3.25}$$

Questo fattore renderà le nubi più luminose vicino alla fonte di luce, il Sole, permettendo di rappresentare il Silver Lining.

L'ultimo effetto da rappresentare è quello dei bordi scuri delle nubi, che è molto più apprezzabile nelle regioni più dense e tonde.
In alcuni casi le zone concave delle nubi sono addirittura più luminose di quelle sporgenti, in quanto ricevono più luce diffusa dall'interno. Questo effetto è molto simile a ciò che avviene ad una pila di zucchero a velo, come mostrato in figura 3.6.

Figura 3.6: Pila di zucchero a velo, in cui si nota l'effetto dei bordi scuri.

Figura 3.7: Effetto dei bordi scuri su una nuvola.

Si può considerare questo effetto come una probabilità statistica basata sulla profondità nella nube. Più interno è il punto considerato, più aumenta la probabilità che in quel punto si verifichi diffusione luminosa. Tale comportamento può essere descritto matematicamente dalla seguente equazione:

$$D = 1 - e^{-d*2} \tag{3.26}$$

Anche questo fattore, come il precedente, dovrebbe essere view-dependent, in particolare è più visibile quanto più la direzione di vista è simile alla direzione di incidenza dei raggi solari.
L'equazione diventa dunque

$$D = (1 - e^{-d*2}) * \mu \tag{3.27}$$

L'energia finale sarà

$$E = e^{-k*t} * HG + D * \mu \tag{3.28}$$

dove k è un ulteriore parametro che può essere variato per ottenere nubi più scure, associate alle precipitazioni: rappresenta, di fatto, il coefficiente di assorbimento della luce della nube. Può essere utilizzato per "fingere" nubi a sviluppo verticale, che essendo molto più spesse assorbono più luce, senza dover effettivamente prolungare le iterazioni del ray marching.

Campioni di densità. Dato un modello di illuminazione delle nubi, sono necessari i dati di densità del gas su cui applicare questi calcoli e ottenere la quantità di luce che raggiunge un punto.
Dunque, è necessaria una matrice tridimensionale, o texture 3D, che contiene i valori di densità per ogni coordinata spaziale in cui sono presenti delle nubi.

L'approccio più utilizzato in quest'ambito è l'utilizzo del concetto di moto browniano, ovvero il moto disordinato di particelle nell'ordine del micrometro sottoposte a una forza di gravità trascurabile, presenti anche in sospensioni gassose.
La sua rappresentazione matematica, modellata dal fisico Albert Einstein, è usata per descrivere l'andamento temporale di una classe molto ampia di fenomeni casuali.

Aggiungendo diverse iterazioni di rumore (ottave), in cui si incrementano successivamente le frequenze a passi regolari (lacunarità) e si diminuisce l'ampiezza (guadagno) del rumore, si può ottenere una granularità più fine nel rumore quindi

più dettagli. Questa tecnica è chiamata "moto browniano frattale" (fBM), o semplicemente "rumore frattale", e presenta la caratteristica di autosimilarità, importante per modellare delle nubi realistiche.

Tuttavia, il rumore frattale non è in grado di rappresentare l'aspetto tondeggiante tipico delle regioni più dense delle nubi; per questo motivo è necessario combinarlo con una seconda tipologia di texture tridimensionale.

La seconda tipologia di rumore utilizzato è il rumore cellulare o di Worley, impiegato ampiamente nella computer grafica per simulare trame nella pietra, effetti caustici dell'acqua o la disposizione di cellule biologiche per via della sua forma a corpuscoli. L'idea alla base della generazione di questo tipo di rumore è considerare dei punti casuali nello spazio, e quindi per ogni posizione memorizzare la distanze d_n al n-esimo punto più vicino (ad esempio il secondo punto più vicino).

Una possibile ottimizzazione implementativa consiste nella suddivisione dello spazio in una griglia uniforme: ogni cella della griglia conterrà un punto in una posizione casuale. In questo modo, durante la generazione della texture, la ricerca del punto più vicino si restringe alla cella in cui è contenuto e le celle ad essa adiacenti.

Inoltre, è necessario che questa trama sia "seamless", ovvero possa essere ripetuta e accostata senza mostrare discontinuità o artefatti ai bordi. Per ottenere una texture con questa proprietà è possibile circondare la griglia iniziale con otto sue copie su ogni lato, e poi eseguire l'algoritmo di generazione considerando anche le celle circostanti.

Posizionamento delle nuvole. Dunque, dati un modello di illuminazione e delle texture 3D di densità, è possibile renderizzare delle nubi volumetriche. Le nuvole basse e medie possono trovarsi da un'altezza di circa 1500 metri ad un'altezza massima di 4000 metri.
Per l'implementazione in Unity, è stata utilizzata la mesh di una semisfera di raggio rn e spessore w, dove:

1. r_n è pari al raggio della Terra $r_n + 1500$,
2. w è la differenza di raggio tra le due superfici sferiche che racchiudono le nubi volumetriche, ovvero 4000 - 1500 = 2500.

Se supponiamo che nel sistema di riferimento il punto (0,0,0) si trovi sulla superficie terrestre, questa semisfera deve essere centrata nel punto (0, $-r_t$, 0). Le normali della mesh dovrebbero essere rivolte verso il centro della semisfera, in modo che le nuvole siano visibili dal suo interno.

L'algoritmo di ray-marching viene lanciato per ogni pixel visibile della faccia inferiore del container delle nubi, limitando il numero di campioni prelevati dalla texture in modo che non vengano considerati punti fuori dalle altezze stabilite, quindi controllando che la lunghezza del vettore posizione (di campionamento) sia inferiore ad r_n.

Nel caso in cui la camera virtuale si trovi all'interno del container, si noti che la faccia della mesh renderizzata è quella posteriore ad altezza 4000 metri.
Le posizioni di campionamento partirebbero dunque da quella superficie, e si fermerebbero subito perché la lunghezza del vettore posizione sarebbe sempre maggiore o uguale ad rn. Quindi, è necessario impostare la posizione iniziale di campionamento alla posizione della camera, in modo da includere nell'esecuzione tutti i campioni interni al contenitore.

Quest'ultimo accorgimento permette all'utente di attraversare realisticamente le nubi, effetto non ottenibile se non si utilizzasse uno shader volumetrico. In quest'ultimo caso, infatti, le nubi apparirebbero "cave" all'interno.

Post-processing. Alcuni effetti di post-processing del colore hanno inoltre permesso di ottenere stili diversi.
Tra gli stili di visualizzazione più utilizzati di recente vi è il cel-shading, tecnica finalizzata ad imitare la grafica dei fumetti o di disegno a mano libera, caratterizzata da colorazioni a tinte unite.
Per renderizzare le nuvole in questo stile si è adottata la tecnica della posterizzazione, che trasforma gradienti di colore in variazioni nette, effettuando una sorta di quantizzazione visiva e quindi limitando le tonalità di colore visualizzate. La posterizzazione fa uso dell'arrotondamento per difetto del valore in ingresso, ed è in grado di ritornare un valore in funzione del numero massimo di tonalità desiderate. Segue lo pseudocodice per implementare un semplice effetto di posterizzazione: Questo effetto ha buoni risultati visivi solo nelle zone in cui la nube è densa, ma

Algorithm 3 Effetto di posterizzazione.

```
1: procedure Posterization(In, Steps)
2:     return floor(In / (1 / Steps)) * (1 / Steps)
3: end procedure
```

non dove risulta più rarefatta.
Per far fronte a questo limite, vengono sottoposti a posterizzazione solo i canali RGB, mentre il canale alpha rimane invariato. Inoltre, si può utilizzare il canale alpha per pesare i passi di posterizzazione in base alla trasparenza e far sì che

l'effetto di cel-shading diminuisca proporzionalmente alla densità visualizzata.

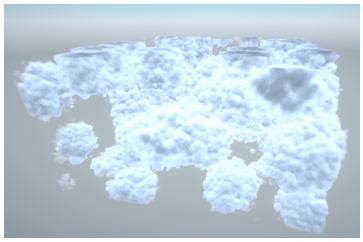

Figura 3.8: Stile fotorealistico.

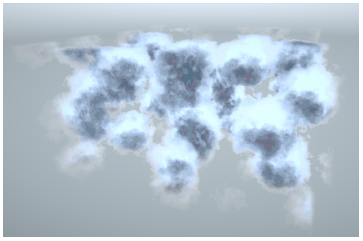

Figura 3.9: Stile dipinto a mano.

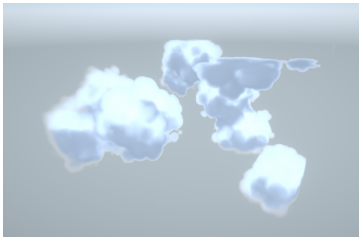

Figura 3.10: Stile cartoon.

Ottimizzazione. La resa grafica stilizzata per le nubi permette di aumentare di molto il passo di campionamento e quindi le prestazioni, senza compromettere l'aspetto delle nubi, ma anzi supportando la grafica cel-shading.
Nel caso in cui si ricerca uno stile con sfumature di colore, aumentare il passo di campionamento potrebbe creare un artefatto di "banding". Per attenuarlo, si può utilizzare del rumore bianco come offset del passo di campionamento: questa operazione sostituirà il banding con una sfocatura, che risulta meno artificiale e si confonde bene con il resto della nube.
In combinazione a queste due tecniche, l'implementazione include un'ottimizzazione del passo di campionamento che dipende dalla distanza. In particolare, si sfrutta il fenomeno di fusione spaziale dei colori dell'occhio umano, per cui i dettagli più lontani risultano sempre meno distinguibili: è possibile incrementare il passo di campionamento in maniera direttamente proporzionale alla distanza dalla camera. In questo modo, si concentra la maggior parte della potenza computazionale sui dettagli vicini all'utente.

Un'altra importante ottimizzazione consiste nel fermare il campionamento su un raggio quando il pixel da colorare ha raggiunto una densità prossima a quella massima, in modo da saltare l'acquisizione di campioni che avrebbero un apporto trascurabile sulla resa finale.

Infine, è possibile ridurre la scala di rendering, e quindi il numero di pixel da campionare, esclusivamente per il render layer delle nuvole. Si utilizza poi un semplice filtro di interpolazione bilineare per calcolare i pixel intermedi e riportare il layer alla risoluzione di rendering precedente. Questa operazione risulta particolarmente efficace per gli stili sfumati, e permette di regolare facilmente il trade-off

tra livello di dettaglio e prestazioni.

3.4 Fulmini

Una situazione di forte instabilità meteorologica all'interno della simulazione è resa molto efficace dalla presenza di fulmini. Anche nella vita reale, questi fenomeni possono avere un forte impatto sull'attività umana, sia direttamente che indirettamente [3]. Inoltre, sono usati come elemento di dinamicità e ingrediente del game-play di molti videogiochi tra quelli analizzati nel capitolo 2.
Una buona modellazione di questo effetto grafico migliorerà dunque il senso di immersione per le condizioni instabili, e potrebbe veicolare ulteriori scelte di game-design che introducono nuove interazioni a supporto del parametro di presenza.

Obiettivi del modello. Si elencano dunque gli obiettivi del modello per la presente implementazione:

1. Prestazioni adeguate ad un applicativo real-time.
2. Rappresentazione accurata ma regolabile per permettere di creare fulmini artisticamente interessanti o stilizzati.
3. Controllo e disponibilità su CPU delle coordinate dell'impatto, in modo da poter usare l'informazione per ulteriori effetti grafici o meccaniche di gioco ad esso collegate.

Anche in questo caso, è bene analizzare le caratteristiche e la classificazione dei fulmini per dedurne l'aspetto e il comportamento da modellare.

Classificazione dei fulmini. Si definisce fulmine un'intensa scarica elettrica di grandi dimensioni, instaurata tra due oggetti ad alta differenza di potenziale. La maggior parte di questi fenomeni ha luogo in una cellula temporalesca, che ne genera circa 2 al minuto. In particolare, i fulmini possono essere classificati in base a quali particelle sono coinvolte nella sua generazione. Segue una lista in ordine crescente di rarità:

1. Fulmini nuvola-nuvola. Sono i più frequenti, e si instaurano tra le cariche positive dei cristalli di ghiaccio in alta quota e le cariche negative nelle gocce di pioggia che si formano in bassa quota.
2. Fulmini nuvola-terra. Rappresentano il 10% delle scariche elettriche, e in questo contesto sono i più importanti da rappresentare perché interagiscono con

l'attività umana. Dall'acqua nebulizzata sulla punta delle piante provengono cariche positive con una differenza di potenziale rispetto a quelle negative nelle gocce di pioggia.

3. Fulmini a secco. Osservabili durante tempeste di sabbia, bufere di neve e nelle nubi di cenere vulcanica e incendi.

4. Fulmini globulari. Tutt'ora sotto fase di studio, sono molto rari e hanno una forma sferica della dimensione di pochi centimetri.

Analisi del fenomeno. Per modellare i fulmini, è anche importante capire cosa innesca la differenza di potenziale e come viene generata la luce (lampo) che raggiunge i nostri occhi e ci fa percepire il fenomeno.
L'aria dell'atmosfera è generalmente isolante, in quanto neutra. Per permettere il passaggio di corrente elettrica, dev'essere ionizzata, ovvero una quantità di energia dev'essere spesa per strappare alcuni elettroni e creare un' eccesso di carica. Il processo di formazione di un fulmine è detto "scarica a valanga", perché la sua energia ionizzerà ulteriori particelle d'aria dopo l'innesco [4].
Si analizzano dunque le fasi di una scarica elettrica nuvola-terra più dettagliatamente.

1. Alcuni elettroni, mossi dal forte campo elettrico, scendono dalla nube a scatti rettilinei di 30-50 metri coperti in circa 0.1 s, intervallati da decine di millisecondi. Lungo il percorso, che per ogni tratto può presentare una direzione diversa, si crea un canale ionizzato che può essere lungo anche chilometri.

2. Allo stesso tempo, le cariche positive salgono verso la nube dal punto più alto del terreno.

3. Le cariche possono incontrarsi a circa 30-50 metri creando il canale ionizzato, della larghezza di pochi centimetri: si instaura una forte corrente elettrica al suo interno. Il canale che parte dal terreno potrebbe raggiungere direttamente la nube, generando ciò che è chiamato fulmine ascendente. Il canale potrebbe anche non crearsi; in questo caso nessun fenomeno ha luogo.

4. Se si crea un canale ionizzato, una potente scarica di ritorno al suo interno porta la corrente dal suolo alla nube alla velocità di 130 milioni di metri al secondo. A questo punto, il canale può essere utilizzato nuovamente da altri fulmini o potrebbe generare ulteriori canali secondari, creando un effetto di luce ramificato.

Il lampo è dunque una luce che attraversa il canale principale e i secondari. Il colore varia dal bianco al blu in base all'intensità ma può presentarsi anche giallo o rosso se l'atmosfera ha un alto livello di inquinamento.

3.5 Rendering dei fulmini

In questa sezione si proporranno tre tecniche per modellare e renderizzare i fulmini, se ne descriveranno i vantaggi e svantaggi, e si descriverà come entrambe vengono utilizzate nell'implementazione proposta.

Come appena visto, la percezione visiva del fulmine è un percorso luminoso composto da una serie di segmenti rettilinei con diversa direzione e possibilità di ramificazione. Come visibile in figura 3.11 e 3.12, la luminosità del fulmine è maggiore alle estremità delle ramificazioni, e raggiunge il picco massimo durante la scarica di ritorno, tanto da nascondere ai nostri occhi e ai sensori ottici, che percepiscono la luce in maniera relativa, i rami secondari. Data l'imprevedibilità

Figura 3.11: Fulmine fotografato mentre crea diversi canali ionizzati.

Figura 3.12: Fotografia della scarica di ritorno dello stesso fulmine.

dei cambi di direzione, sarà sicuramente necessario un algoritmo di generazione di numeri casuali, o una generazione di texture procedurale utilizzando la GPU.
Come già accennato, un aspetto importante è il punto di impatto: per gestire eventuali effetti del fulmine all'interno dell'applicazione, come esplosioni o meccaniche di gioco, bisogna essere a conoscenza del punto in cui colpirà.

Tecniche di rendering. Per la presente implementazione, sono proposte tre tecniche di rendering dei fulmini. Di seguito se ne analizzano i vantaggi e gli

svantaggi:

1. Texture 2D. Si utilizzano texture mappate su un piano.
   (a) Pro:
      i. Semplicità di implementazione. Unity permette l'utilizzo di shader preinstallati che offrono sia l'alpha clipping, sia l'emissione di luce.
      ii. Semplicità computazionale.
      iii. Conoscenza del punto di impatto a priori. Data la texture, il punto di impatto è definito. Richiede però la forzatura di impostare a mano le coordinate relative per ogni texture.
   (b) Contra:
      i. Resa grafica piatta.
      ii. Impossibilità di visualizzare percorsi secondari diramati su 3 dimensioni. Questo limite potrebbe essere risolto utilizzando più piani ruotati sul proprio asse per ogni fulmine.
      iii. Impossibilità di visualizzare la formazione graduale del percorso senza intaccare la credibilità dell'effetto.
2. Sistema particellare. In questo caso il lampo segue il percorso di una particella invisibile che si sposta in diverse direzioni.
   (a) Pro:
      i. Resa grafica tridimensionale e procedurale. É possibile generare ricorsivamente particelle a partire da una singola per renderizzare percorsi secondari.
   (b) Contra:
      i. Impossibilità di prevedere il punto in cui il fulmine colpirà. Sarebbe necessario coinvolgere il sistema di collisioni, che potrebbe non essere affidabile o compromettere le prestazioni nei casi in cui la particella ha altissima velocità, come questo.
      ii. Limitazione delle ottave del rumore. Il sistema particellare si aggiorna ad una frequenza massima fissata dal numero di frame al secondo. Questo pone un limite importante al dettaglio del rumore utilizzabile per rappresentare i cambi di direzione.
3. Mesh. Consiste nella realizzazione di un modello 3D standard la cui forma somigli alla ramificazione di un fulmine.
   (a) Pro:

i. Complessità di implementazione. É necessario sviluppare un sistema che genera delle mesh sulla base di diversi parametri e le animi automaticamente per visualizzare il percorso del fulmine.
ii. Resa grafica accurata e tridimensionale. La mesh può presentare ramificazioni in tutte e tre le dimensioni e il sistema implementato potrebbe condividere informazioni con lo shader (come ad esempio la distanza dall'estremità di ogni ramo), permettendo un ottimo controllo della forma.
iii. Il fulmine può essere generato in precedenza per avere totale controllo del livello di dettaglio del rumore.
iv. Conoscenza del punto di impatto a priori. Si potrebbe utilizzare come pivot della mesh il punto di impatto, per semplificare nettamente i calcoli nella fase di istanziazione da parte del sistema di meteo.

(b) Contra:

i. Complessità computazionale. Il modello risultante potrebbe avere un numero di vertici elevato; tuttavia, i fulmini nuvola-terra sono fenomeni abbastanza rari e si consumano in brevissimo tempo.
ii. Dinamicità limitata. Dovendo generare le mesh a priori (o "baked"), c'è un limite alla variabilità di fulmini visualizzabili, che dipende dalla quantità di memoria che si è disposti ad utilizzare per immagazzinare le mesh animate.

Scelte di implementazione. Per la presente implementazione, sono state selezionate più di una tecnica per rappresentare i fulmini. In particolare, è utilizzato sia l'approccio particellare, per i fulmini più distanti e quelli nuvola-nuvola, sia l'approccio mesh, per modellare quelli prossimi al giocatore.
Questa scelta adotta la metodologia LOD (Level of Detail), ovvero la tendenza nelle applicazioni real-time di visualizzare tutto ciò che è più importante e vicino in maniera dettagliata e quindi impiegare più risorse per gli elementi che saranno maggiormente apprezzati dall'utente, a scapito della precisione dei game object secondari.

Fulmini come mesh. Per modellare i fulmini è stato utilizzata la funzionalità Geometry Nodes del software di modellazione 3D Blender. Geometry Nodes permette di costruire una mesh in maniera procedurale; la variazione dei parametri esposti dal sistema costruito permette poi di animarla. É bene notare che, in questo modo, la mesh di output potrebbe cambiare il numero di vertici durante l'animazione: per questo motivo verrà esportata utilizzando un formato di rappresentazione particolare, compatibile con Unity, che supporta per questa caratteristica. Il formato usato è infatti l'Alembic, ed è ampiamente adottato con questo specifico scopo da

molti programmi di modellazione 3D.
In seguito alla modellazione, un numero limitato di modelli è stato esportato in Alembic (.abc) e importato in Unity come asset.

Si descrive ora l'implementazione dell'algoritmo di generazione della mesh animata.
Si definisce parametro spline la distanza del punto di controllo dall'inizio della curva rimappata su un'intervallo da 0 a 1. L'input del sistema è una curva parametrica di Bézier, generata con Blender utilizzando pochi punti di controllo. La curva viene ricampionata per aumentare la sua risoluzione ed avere una distorsione più dettagliata; una texture tridimensionale di rumore (Perlin noise frattale) è poi utilizzata come offset alla posizione dei singoli campioni. Per non muovere il punto di generazione e il punto di impatto del fulmine, si modula il rumore in base al parametro spline: è necessario moltiplicare per 0 in corrispondenza dell'inizio e della fine, quindi viene utilizzata una funzione simile ad una gaussiana.
Per animare il percorso del fulmine, si rimuovono e aggiungono sezioni della curva dalla punta in base a un parametro esposto (trim), che sarà poi legato al tempo di simulazione.
Le ramificazioni sono create come istanze sui campioni: si istanziano ulteriori curve, la cui rotazione è impostata da una texture tridimensionale di rumore bianco mappata sull'intervallo $[0, 2\pi]$. Uno dei canali del rumore bianco può essere utilizzato per selezionare i campioni che istanzieranno dei rami, in modo da rendere la mesh più variabile. Anche i rami sono ricampionati e disorti dalla prima texture di rumore Perlin, ma per animarli è necessario usare lo spline parameter catturato dopo il trimming della curva principale, in modo che la loro crescita non sia contemporanea e innaturale, ma proporzionale alla distanza dalla punta della curva tagliata. La logica dell'istanziamento dei rami può essere ripetuta in maniera iterativa per avere più livelli di ramificazione. Ad ogni iterazione si diminuisce la lunghezza dei rami per un effetto più naturale. Si nota che queste scelte stilistiche sono state adottate facendo riferimento ad acquisizioni di fulmini reali, come quelle in figura 3.11 e 3.12.
In seguito, la curva è trasformata in una mesh cilindrica che ne segue il profilo. Il raggio del cilindro viene reso proporzionale alla lunghezza della curva: questo garantirà una gerarchia di dimensioni alla ramificazione.

Shader dei fulmini. Come visto in precedenza, le fasi visibili del fenomeno sono la ramificazione e la successiva scarica di ritorno, che nasconde i rami secondari e rimane impressa per qualche istante nei sensori e si dissolve gradualmente. Durante la dissolvenza, la luce può lampeggiare ancora se altri fulmini utilizzano lo stesso canale ionizzato di quello originario.
Inoltre, si noti come le estremità delle ramificazioni siano appaiano più luminose

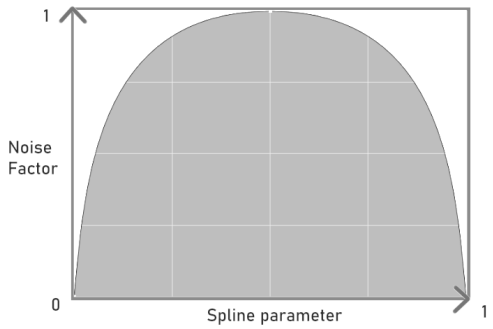

Figura 3.13: Funzione che modula il rumore.

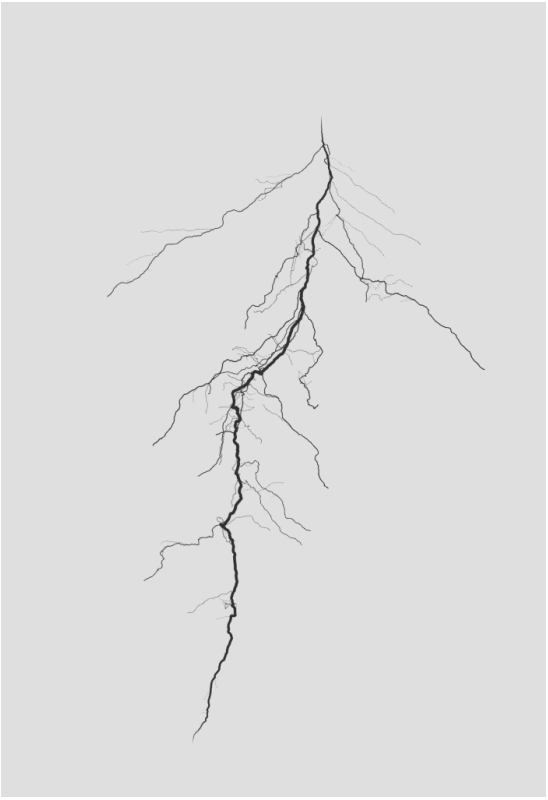

Figura 3.14: Esempio di ramificazione ottenuta con Geometry Nodes.

del resto del canale, si cercherà di riprodurre anche questo effetto.

Quasi la totalità degli obiettivi appena descritti necessitano dell'informazione sulla fase di scarica del fulmine. L'animazione è controllata dal trimming della curva originale esposto sotto forma di parametro "Time"; quindi parte da time = 0 e inizia a ramificarsi fino all'estensione massima in corrispondenza di time = 1. Superato il valore di 1, la curva resterà uguale, ma è possibile sfruttare i valori successivi per gestire l'animazione dopo che il fulmine ha colpito il terreno. Ad esempio, è possibile nascondere i rami se time > 1. Per rendere invece lo shader più luminoso in corrispondenza delle punte dei rami, bisogna impostare un attributo ai vertici della mesh in modo che la GPU possa accedervi in fase di shading. In particolare, si assegna la somma degli spline parameter di tutti i rami: è possibile visualizzare questo attributo in figura 3.15. Quando il valore di time supera 1, è possibile far dissolvere il fulmine moltiplicando la somma degli spline parameter per un fattore che varia da 1 a 0 mentre time varia da 1 a 2 (3.17). Questo fattore

può a sua volta essere moltiplicato per un valore molto alto per creare il lampo della scarica di ritorno (3.16). Adesso si può usare lo stesso fattore per regolare

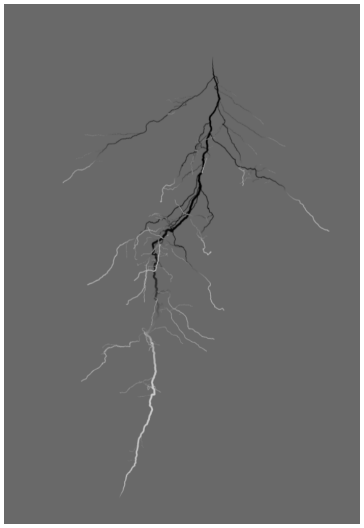

Figura 3.15: Visualizzazione della somma degli spline parameter di tutti i rami un frame prima dell'impatto.

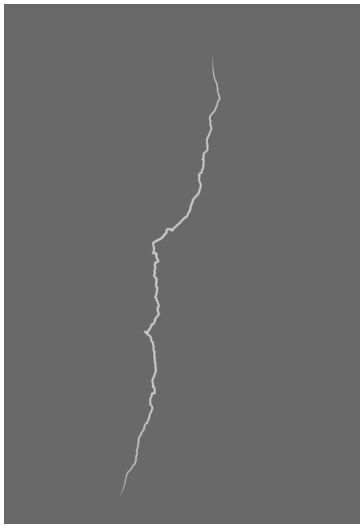

Figura 3.16: Visualizzazione della somma degli spline parameter di tutti i rami nel frame dell'impatto.

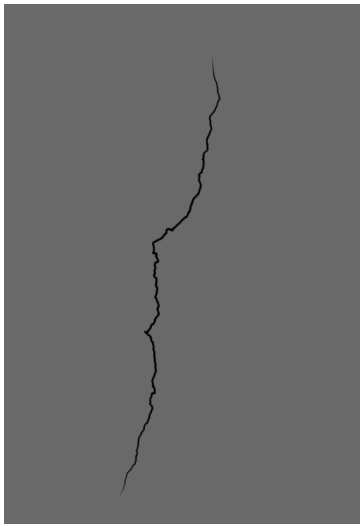

Figura 3.17: Visualizzazione della somma degli spline parameter di tutti i rami dopo l'esaurimento dell'energia del fulmine.

l'intensità di uno shader emissivo. La dissolvenza è resa meno uniforme utilizzando del rumore cellulare tridimensionale come soglia per la trasparenza del fulmine (3.20).
Inoltre, moltiplicando l'intensità dopo l'impatto per una funzione seno valutata su time, si può ottenere il lampeggio che rappresenta l'utilizzo del canale ionizzato da ulteriori fulmini. Per variare la frequenza della funzione, si distorce il segnale aggiungendo del rumore.

Importazione. Lo stile del fulmine può essere ora alterato a piacimento. I controlli includono:

1. Forma generale, attraverso le maniglie della curva Bézier in input.
2. Colore.
3. Spessore.
4. Luminosità.

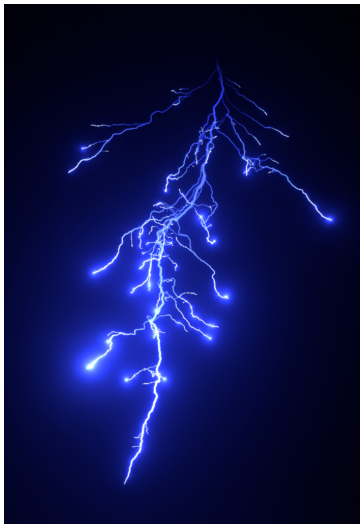

Figura 3.18: Render del fulmine prima dell'impatto (Time = 0,99).

Figura 3.19: Render del fulmine durante l'impatto (Time = 1).

Figura 3.20: Render del fulmine dopo l'impatto (Time = 1,8).

5. Quantità dei rami, sia prima dell'impatto sia dopo l'impatto.
6. Divergenza dei rami.
7. Differenza di luminosità tra le estremità dei rami e la radice.
8. Velocità dell'animazione.
9. Seed dei rumori. Permette di ottenere infinite variazioni.

Si esportano quindi in formato Alembic un numero limitato di fulmini, ognuno dal peso di circa 10 MB. Si utilizza il package Alembic di Unity per leggere i file .abc, che mette a disposizione uno script che funge da player delle animazioni. É importante disabilitare le funzioni aggiuntive dell'importer, come il calcolo delle tangenti, che potrebbero rallentare l'esecuzione dell'animazione e non sono di utilità in questo contesto. I fulmini potranno essere istanziati come GameObject di Unity dal sistema di meteo dinamico, come spiegato nei capitoli successivi.

Fulmini come sistemi particellari Per questo sistema, si è utilizzato il modulo VFX graph di Unity, che permette di generare e gestire le particelle tramite moduli e avere elevato controllo sulle fasi della sua vita, ovvero la fase di inizializzazione, dell'update e della sua distruzione.

Concettualmente, l'implementazione presenta diverse similitudini rispetto alla precedente, ma è naturalmente orientata alle particelle. Una particella madre viene generata con velocità verticale negativa. La sua posizione viene aggiornata campionando una texture di rumore Perlin tridimensionale ad ogni frame e sommando il suo valore con quello del rumore. Come già spiegato, questo limita parecchio il dettaglio del fulmine, motivo per cui verrà utilizzato come effetto visuale distante dalla camera.
In ogni frame, la particella genera inoltre una scia di colore emissivo tramite il modulo "Initialize Particle Strip" di VFX Graph, formando il canale ionizzato mentre scende. Per realizzare la scarica di ritorno, si è creata una curva personalizzata da cui valutare la luminosità in base alla vita della particella madre.
Inoltre, la particella madre genera una particella figlia con un rate costante, che avrà il suo stesso comportamento e una componente di velocità aggiuntiva che la spinge verso l'esterno, parzialmente casuale e esposta ad un controllo dell'utente. Questa generazione può essere reiterata diverse volte per renderizzare una ramificazione. Anche in questa implementazione, la dimensione delle scie è inversamente proporzionale al livello di ramificazione: la particella madre avrà la scia più grande. Come osservabile nelle figure 3.21,3.22 e 3.23, il dettaglio del rumore risulta essere decisamente minore, ma accettabile se l'effetto è visualizzato a grandi distanze.

Figura 3.21: Esempio di fulmine implementato usando VFX Graph di Unity.

Figura 3.22: Esempio di fulmine implementato usando VFX Graph di Unity.

Figura 3.23: Esempio di fulmine implementato usando VFX Graph di Unity.

3.6 Politica physics-based

In questa sezione si descriverà la tecnica adottata per implementare una simulazione dinamica del meteo il più possibile accurata.
La tecnica dovrebbe permettere di memorizzare e gestire separatamente diverse variabili meteorologiche con una precisione bidimensionale (dall'alto):

1. Pressione atmosferica
2. Temperatura
3. Umidità
4. Vento

Verranno dunque utilizzate matrici per salvare le variabili in maniera discreta, con risoluzione regolabile.
Per modellare il comportamento e l'evoluzione di queste proprietà, si utilizza la teoria della fluido-dinamica. Naturalmente è necessario che l'algoritmo funzioni in tempo reale: per questa ragione, durante la trattazione verranno effettuate diverse approssimazioni e semplificazioni rispetto alla reale evoluzione dei fenomeni meteorologici.

Simulazione dei fluidi. Alla base delle simulazioni di fluidi incomprimibili vi sono le equazioni di Navier-Stokes.

$$\nabla \cdot \mu = 0 \tag{3.29}$$

$$\rho \times u' = -\nabla p + \mu \nabla^2 u + \rho F \tag{3.30}$$

1. u è la velocità del fluido
2. $\rho \times u'$ è l'accelerazione del fluido
3. ∇p è il gradiente di pressione
4. μ è la viscosità
5. ρF rappresenta le forze esterne

La prima equazione stabilisce che la divergenza della velocità del fluido è uguale a 0. Ciò significa che lo scorrimento del fluido non può divergere, perché sarebbe come se venisse creato dal nulla; e non può convergere, perché significherebbe distruggere del fluido. In pratica, questa equazione fa in modo che la massa sia conservata e che il fluido non venga compresso o dilatato durante la simulazione.

La seconda equazione stabilisce che l'accelerazione del fluido (derivata della velocità) dipende dalle sue forze interne, ovvero il gradiente di pressione e le forze dovute alla viscosità del fluido (che determina l'attrito tra le particelle del fluido), e dalle forze esterne, come ad esempio la gravità e elementi solidi che interferiscono.
Come si vedrà in seguito, questa equazione può essere soddisfatta modellando due comportamenti del fluido: la diffusione e l'avvezione.

Ci sono due principali tecniche per simulare i fluidi. Infatti, è possibile modellare il fluido usando delle particelle che hanno delle proprietà e si muovono nello spazio oppure una griglia di celle stazionarie in cui sono memorizzate le proprietà del fluido.
La seguente implementazione farà uso della seconda tecnica, in quanto le prestazioni sono facilmente gestibili modificando la risoluzione della griglia. Questo può potenzialmente essere fatto in maniera localizzata, e mette dunque delle basi sicure per eventuali future ottimizzazioni.

Diffusione. La diffusione è il fenomeno per cui i fluidi, anche in assenza di moto, si diffondono progressivamente per occupare tutto lo spazio a disposizione. Ciò accade per ogni proprietà che il fluido possiede: nei successivi esempi si prenderà come esempio la diffusione della pressione, ma lo stesso principio vale per tutte le altre proprietà.

Per modellare la diffusione si vuole che ogni valore nella griglia in posizione (x,y) diventi gradualmente uguale alla media dei 4 valori adiacenti.
Di seguito, dunque, vi è una semplice interpolazione lineare che descrive questa situazione:

$$d_i + 1 = d_i + k(s_i - d_i) \tag{3.31}$$

Questa equazione presenta un problema: quando k è > 1, il valore target si verrà superato e la simulazione risulterà instabile.
Limitare il valore di k a 1 renderà invece la simulazione poco realistica e non permetterà di avere controllo sulla sua velocità.
Dunque, invece di usare i valori correnti per calcolare quelli alla prossima iterazione,

si tenta di trovare i valori che porterebbero a quelli correnti dopo un'iterazione. L'equazione diventa:

$$d_i = d_{i+1} - k(s_{i+1} - d_{i+1}) \tag{3.32}$$

Se si porta d_{i+1} al primo membro:

$$d_{i+1} = d_i + ks_i + \frac{1}{1+k} \tag{3.33}$$

É possibile notare che questa relazione non è più lineare ma iperbolica. Adesso, qualunque sia il valore di k, l'interpolazione sarà stabile e non eccederà il valore target.
Tuttavia questo metodo presenta un secondo problema: è necessario conoscere la media dei valori adiacenti alla prossima iterazione s_{i+1}, dato di cui ancora non si dispone.

Si noti che s_{i+1} dipende da d_{i+1} nelle posizioni adiacenti. In pratica è un sistema di equazioni, che può essere facilmente svolto da un umano usando procedure diverse in base alla situazione; è tuttavia necessario definire un algoritmo che una macchina possa eseguire per risolvere un sistema di equazioni usando sempre lo stesso metodo. Per far ciò, si usa un solutore iterativo per approssimare le soluzioni, ovvero le d_{i+1} delle celle adiacenti.
Il solutore proposto, noto come metodo di Gauss-Seidel, consiste nell'assegnare soluzioni casuali alle incognite (ad esempio 0) per poi risolvere le equazioni singolarmente ed sostituire le soluzioni errate ottenute all'interno delle equazioni di partenza. Continuando a eseguire questo ciclo, le soluzioni errate convergono a quelle esatte. Questo metodo richiede che la matrice associata al sistema di equazioni sia a diagonale dominante in senso stretto, deve cioè valere:

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}| \tag{3.34}$$

Questa condizione assicura che le soluzioni errate calcolate iterativamente convergano ad un valore.
É dunque evidente che la formula considerata soddisfa già questa condizione, in quanto il numeratore ha valori che vanno da 0 a k, mentre il denominatore è sempre maggiore di k (1+k).

Avvezione. L'avvezione è il trasferimento delle proprietà del fluido in una direzione causato dal movimento fisico delle particelle; nel caso della meteorologia, si

parla di correnti (di fatto, il vento). Si possono rappresentare le correnti come un campo vettoriale. Si associa dunque un vettore bidimensionale ad ogni cella della griglia che rappresenta la direzione e forza del vento in quel punto.

Vi sono due problematiche nell'implementazione di questo fenomeno:

1. ogni vettore non punta esattamente nel centro di un'altra cella, e quindi avrà effetto su molteplici celle in maniera opportunamente pesata.
2. i contributi su una singola cella potrebbero provenire da diversi punti della griglia, quindi non è possibile eseguire un singolo calcolo per cella.

Anche in questo caso, la soluzione risiede in un approccio a ritroso: si trova il punto nella griglia dalla quale le proprietà provengono e poi si pesano i contributi per interpolazione bilineare delle quattro celle più vicine a quel punto.

Per eseguire la prima operazione, è possibile sottrarre il vettore nella cella di riferimento moltiplicato per il passo temporale Δt. Una volta trovato il punto preciso, si rappresentano le sue coordinate in maniera relativa al centro delle quattro celle, e si procede con un'interpolazione bilineare, ovvero un'interpolazione lineare sull'asse verticale seguita da un'interpolazione lineare sull'asse orizzontale.

Proiezione. In un campo vettoriale, si parla di "rotore" e "divergenza" per indicare rispettivamente le componenti del campo che formano pattern circolari in senso orario o antiorario, e quelle che formano pattern radiali convergenti o divergenti.
Come già spiegato, la prima equazione di Navier-Stokes suggerisce che la componente divergente vada azzerata per evitare situazioni di allungamento/compressione o generazione/distruzione del fluido.
Utilizzando i concetti di diffusione e avvezione, si calcola un nuovo campo vettoriale che contiene sia una componente di rotore sia di divergenza; è necessario sottrarre quest'ultima. In pratica, il campo vettoriale calcolato può essere espresso come una somma di due campi vettoriali: uno irrotazionale, ovvero con rotore uguale a zero in qualsiasi cella, e uno solenoidale, ovvero con divergenza nulla ovunque. Questo processo è anche detto decomposizione di Helmholtz.
L'obiettivo è calcolare il secondo, quello solenoidale; tuttavia, dato che calcolare la componente irrotazionale risulta più diretto e semplice, è possibile sottrarlo dal campo vettoriale iniziale per ottenere un campo privo di divergenza.

Per farlo, si definisce la divergenza del campo vettoriale in una cella come:

$$\nabla \cdot v(x,y) = \frac{v_x(x+1,y) - v_x(x-1,y) + v_y(x,y+1) - v_y(x,y-1)}{2} \tag{3.35}$$

Questa equazione include una differenza orizzontale e una differenza verticale del vettore velocità, divise per il numero di celle considerate, ovvero 2. Per visualizzare questo concetto, è possibile descriverlo su una dimensione j. Se in una posizione j_1 minore si ha una velocità minore rispetto ad una posizione maggiore j_2, significa che la quantità di fluido in uscita è maggiore di quello in entrata (figura 3.24) . Viceversa, se nella posizione j_1 si ha una velocità maggiore, la quantità di fluido in ingresso è maggiore (figura 3.25). Per trovare la componente divergente dal

Figura 3.24: Divergenza negativa su una dimensione.

Figura 3.25: Divergenza positiva su una dimensione.

campo vettoriale, si utilizzano tecniche basate sulla soluzione di un'equazione di Poisson che coinvolge il potenziale di pressione p, un valore scalare che descrive la distribuzione della pressione. L'equazione di Poisson ci dice che la divergenza del gradiente del potenziale di pressione è proporzionale alla divergenza del campo di velocità del fluido, dove la costante di proporzionalità è la densità del fluido. Questo significa che il potenziale di pressione può essere utilizzato per correggere la divergenza del campo di velocità in una simulazione di fluidi.

$$[p(x-1,y) - p(x+1,y) + p(x,y-1) - p(x,y+1)] - 4p(x,y) = \nabla \cdot v(x,y) \quad (3.36)$$

Si risolve dunque il sistema lineare per p utilizzando il metodo di Gauss-Seidel introdotto in precedenza.

$$p(x,y) = \frac{[p(x-1,y) - p(x+1,y) + p(x,y-1) - p(x,y+1)] - \nabla \cdot v(x,y)}{4} \quad (3.37)$$

Infine si calcola la componente irrotazionale come gradiente del campo di potenziale di pressione:

$$\nabla p(x,y) = (\frac{p(x+1,y) - p(x-1,y)}{2}, \frac{p(x,y+1) - p(x,y-1)}{2}) \quad (3.38)$$

Il rotore di questo campo vettoriale è sempre zero. Sottraendola al campo di

velocità originale si può infine eliminare la divergenza:

$$v_p(x, y) = v(x, y) - \nabla p(x, y) \tag{3.39}$$

L'operazione di proiezione è la più onerosa in termini computazionali, ma è necessario eseguirla più volte per garantire una buona stabilità della simulazione.
É possibile trovare un buon trade-off tra stabilità e prestazioni dimensionando il parametro k di iterazioni per risolvere tutti i sistemi lineari visti in precedenza; $k = 20$ è un valore che garantisce un compromesso ragionevole, ma il parametro è in ogni caso esposto nell'editor di Unity, di modo che possa essere gestito dagli utilizzatori del plug-in o anche fatto variare a runtime per un'ulteriore controllo sull'ottimizzazione.
Quanto detto è anche applicabile alla risoluzione della mappa meteorologica, che si traduce nella dimensione delle matrici rappresentanti i vari campi scalari e vettoriali. Alcuni valori di riferimento sono 32, 64, 128; è bene notare che è necessaria una risoluzione maggiore per avere una precisione costante su terreni più grandi.

Inizializzazione. Parte dei parametri necessari per la simulazione meteorologica evolveranno dunque secondo i concetti di diffusione ed avvezione, ognuno con la propria viscosità; la restante parte è invece calcolata di conseguenza. L'approssimazione più importante, ma necessaria per disporre di buone prestazioni, è la relazione lineare tra temperatura e pressione: di fatto, si assume che l'aria si comporti come un gas ideale. Pertanto, è possibile applicare la legge dei gas perfetti:

$$pV = nRT \tag{3.40}$$

Conoscendo dunque la costante universale dei gas R e il numero di moli dell'aria in un metro cubo, calcolato considerando la sua composizione media, è possibile ricavare il fattore di conversione tra pressione in Pascal e la temperatura in Kelvin:

$$TtoP = R * n = 8.3145 \frac{J}{mol * K} * 44.6429 \frac{mol}{m^3} = 371{,}1816 \frac{Pa}{K} \tag{3.41}$$

In questo modo, è possibile mantenere in memoria solo la mappa di uno dei due parametri e calcolare l'altro all'occorrenza utilizzando la relazione lineare. Inoltre, la simulazione di fluidi potrà essere avviata solo su una delle due matrici.
In particolare, la presente implementazione esegue la simulazione sulla temperatura, sul vento, e sulle nuvole. Prima di iniziare la simulazione, è necessario inizializzare

i valori delle celle con criterio.
La temperatura è inizializzata come funzione dell'altitudine, dell'incidenza dei raggi solari (a sua volta funzione della stagione), e della latitudine. La formula è sperimentale, e fornisce una stima accettabile della temperatura media:

$$baseTemp = 24 + 10 * Clamp(\frac{altitude}{1000}, 1, 2) * sunRatio - \frac{altitude}{100} - |\frac{latitude}{2}| \quad (3.42)$$

Come è chiaro, la temperatura diminuisce con l'aumentare dell'altitudine e del valore assoluto della latitudine (che rappresenta la distanza dall'equatore) e cresce con l'aumentare del parametro *sunRatio*, che è il seno dell'angolo tra la direzione dei raggi solari e un piano tangente alla superficie terrestre rimappato tra -1 e 1. L'altitudine è campionata usando la risoluzione scelta dal GameObject Terrain di Unity, quindi memorizzata in un'apposita matrice (Figura 3.27).
Il vento è invece inizializzato come gradiente di pressione (e quindi di temperatura)

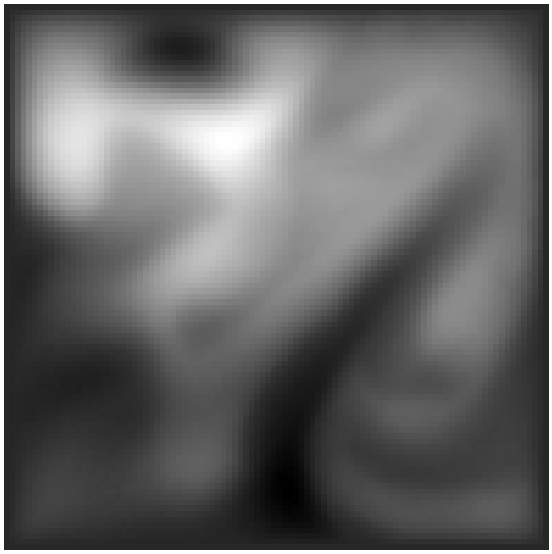

Figura 3.26: Esempio di una simulazione di fluidi bidimensionale in esecuzione.

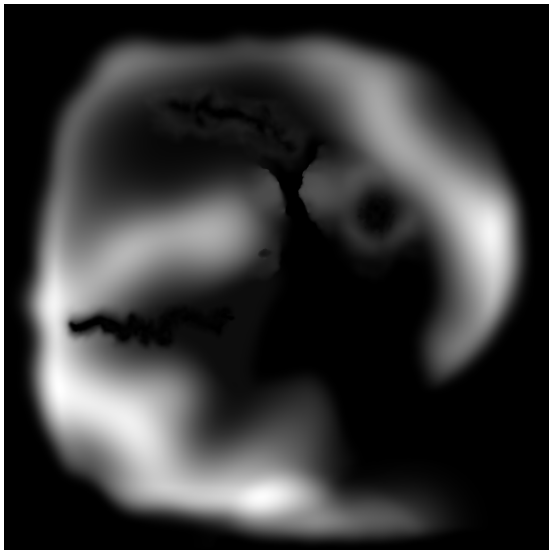

Figura 3.27: Rappresentazione in scala di grigi della matrice di altitudine di un mondo di gioco.

su entrambi gli assi delle due dimensioni, in quanto l'aria tende generalmente a spostarsi dalle zone ad alta pressione verso quelle a bassa pressione.
Infine, l'umidità è inizializzata facendo riferimento al valore medio dell'umidità $H = 1.6652 \frac{moli}{m^3}$.

Per avere maggiore dinamicità e simulare fenomeni provenienti dall'esterno del mondo di gioco, il rumore di Perlin è anche assegnato ai bordi della matrice, come variazione rispetto al valore più vicino interno al campo.

Evoluzione e fenomeni atmosferici. Si analizza ora l'evoluzione di ogni parametro atmosferico e come questa evoluzione può generare diverse condizioni metereologiche nel tempo.

1. La temperatura evolve in parte utilizzando la simulazione dei fluidi, in parte rimane costante alla temperatura di base calcolata durante l'inizializzazione. La temperatura finale è dunque calcolata come interpolazione lineare tra le due matrici, per non lasciare che diverga troppo dalla media.

2. Come per la temperatura, anche il vento è un'interpolazione tra due matrici, una delle quali evolve usando la simulazione dei fluidi e l'altra viene calcolata come gradiente di pressione.

3. L'umidità cresce nel tempo per evaporazione dell'acqua, quando la temperatura complessiva è maggiore della temperatura di base e l'umidità relativa è minore del 100%. Per la presente implementazione si assume che le aree in cui è presente acqua corrispondano alle aree ad altitudine = 0, ma è possibile sostituire questa informazione con una mappa personalizzata. Per fornire variabilità, la quantità di umidità sommata è poi combinata con una texture procedurale di rumore di Perlin. In seguito anche la matrice dell'umidità evolve lanciando l'algoritmo di simulazione dei fluidi.

4. La mappa della copertura delle nuvole è calcolata come differenza tra la temperatura e il punto di rugiada. Quest'ultimo rappresenta la temperatura alla quale l'aria dev'essere raffreddata per essere satura di vapore acqueo.

$$Cloud = DewPoint - temperature \tag{3.43}$$

Per ricavare il punto di rugiada, si utilizza un'approssimazione empirica, la formula di Magnus-Tetens [5], con un'incertezza di 0.35°C per temperature da -45°C a 60°C.

$$DewPoint = (b * \alpha(T, RH))/(a - \alpha(T, RH)) \tag{3.44}$$

Dove:

- T è la temperatura in gradi Celsius
- RH è l'umidità relativa
- a e b sono i coefficenti Magnus. Come suggerito nel lavoro di Alduchov e Eskridge [6], si possono utilizzare, rispettivamente, i valori costanti di 17.625 e 243.04 °C.
- $\alpha(T, RH) = ln(RH/100) + aT/(b + T)$

L'umidità relativa è definita a tutte le temperature e pressioni come il rapporto tra la pressione del vapore acqueo e la saturazione di pressione del vapore acqueo, ovvero la pressione alla quale si ha un cambiamento di fase (liquido a vapore) ad una temperatura determinata.

$$RH = vapourPressure/saturationVapourPressure \tag{3.45}$$

La legge di Raoult consente di calcolare la pressione di vapore di un gas in una miscela in funzione della pressione totale della miscela e della frazione molare del gas considerato. La frazione molare può essere calcolata dividendo l'umidità (in mol/m^3) per il numero di moli dell'aria in 1 metro cubo. In questa fase, si assume che le due soluzioni (vapore e aria) siano ideali, ovvero le loro interazioni sono della stessa entità delle interazioni interne a ciascuna soluzione.

$$vapourPressure = pressure * humidity/moles \tag{3.46}$$

Per quanto riguarda la pressione di vapore saturo, si fa uso invece dell'equazione di Antoine semplificata, che coinvolge la temperatura e diverse costanti dipendenti dalla natura del vapore.

$$saturationVapourPressure = 6.1164 * 10^{7.5914*T/(T+240.7263)} \tag{3.47}$$

A questo punto è possibile avere una misura della densità nuvolosa in ogni cella della matrice. Tramite interpolazione, si passa dalla copertura corrente (inizialmente nulla) a quella calcolata partendo dall'umidità e dalla temperatura, e si usa la matrice per creare una texture bidimensionale da passare allo shader di raymarching. Quando la densità nuvolosa eccede una soglia in una cella, un'ulteriore matrice che memorizza le precipitazioni viene incrementata

di un passo nella stessa cella, e l'intensità delle precipitazioni nella cella in cui è posizionata la camera virtuale viene assegnata al sistema particellare che ne gestisce la resa grafica.
Durante le precipitazioni, la temperatura diminuisce creando quella che in meteorologia viene definita "cold pool"; inoltre, la densità della nuvola in quel punto diminuisce. Per evitare che questo comporti un effetto "ping-pong" tra l'assenza e la presenza della precipitazione, si introduce una seconda soglia inferiore che è minore della soglia precedente.
La violenza della precipitazione cresce quindi nel tempo se la densità non diminuisce, e può trasformarsi quindi in una cellula temporalesca nel caso in cui la densità della nuvola è tale da far continuare le precipitazioni per tanto tempo. In questo caso, vengono casualmente istanziati dei fulmini intorno alla posizione della camera: come già accennato, si utilizzano i fulmini in formato Alembic nelle immediate vicinanze, e il sistema particellare di Unity per i fenomeni distanti. La probabilità aumenta con la violenza della precipitazione ed è in ogni caso regolabile grazie ad un parametro esposto nell'editor per permettere il controllo artistico della scena.
Inoltre, l'evoluzione di una cellula temporalesca è fortemente legata al vento in quota, detto shear. Per questo parametro si utilizza un vettore bidimensionale dal comportamento casuale basato su rumore di Perlin, che altera il campo

Figura 3.28: Esempio di tempesta. In basso a sinistra sono rappresentate in scala di grigi le mappe di copertura e precipitazione.

scalare delle nubi secondo l'algoritmo di simulazione dei fluidi. In questo modo, i fenomeni possono traslare nel mondo di gioco e creare ulteriore dinamicità e realismo.

Complessivamente, la simulazione dei fluidi è dunque avviata per tre parametri: la temperatura, l'umidità, e le nubi.

3.7 Altre politiche

Politica casuale. La seconda politica meteo scelta è quella casuale. Come già visto nel capitolo 2, la tecnica viene ancora ampiamente utilizzata sia da sola sia solo come supporto a meccanismi più complessi, e permette di avere una buona dinamicità.
La maggior parte delle politiche di questo tipo prevedono un ciclo di aggiornamento per area di gioco: alla fine di ogni ciclo viene selezionata casualmente una nuova condizione meteorologica basata su delle probabilità pre-calcolate a seconda del tipo di area. Questo approccio riduce al minimo il costo computazionale: l'elaborazione per la scelta delle condizioni si limita a specifici momenti dell'esperienza (e.g. passaggio da un ciclo al prossimo).
Tuttavia, questo meccanismo limita il dinamismo all'interno di una determinata area, forzando un cambio ogni qualvolta ci si sposti e rendendo l'esperienza innaturale nei casi in cui questo avvenga più volte in un breve lasso di tempo. Inoltre, tutto ciò è spesso associato ad un rendering locale, che non lascia all'utente nessun modo per avvistare a distanza i cambiamenti imminenti.

Per risolvere queste problematiche, la presente implementazione basa la politica casuale sull'utilizzo del rumore procedurale: vengono usate le stesse mappe introdotte nel capitolo precedente, ma i valori nelle celle sono calcolati come variazione rispetto a quelli standard basati sulle coordinate spazio-temporali impostate. Ad esempio, la temperatura base fa uso della stessa formulazione vista in precedenza; del rumore è aggiunto al campo scalare in modo da ottenere variazioni positive o negative. Inoltre è possibile intervenire sul meteo utilizzando funzioni base di post-processing - ad esempio è possibile coprire i cieli di nubi agendo sulla luminosità della texture associata.
La densità nuvolosa è dunque anch'essa risultato di un rumore procedurale: in questo modo, è possibile avvistare condizioni meteorologiche a distanza rispetto alla posizione dell'utente. Le texture procedurali scorrono a seconda del vento, che a sua volta è una texture procedurale che scorre incrementando nel tempo una dimensione di campionamento.

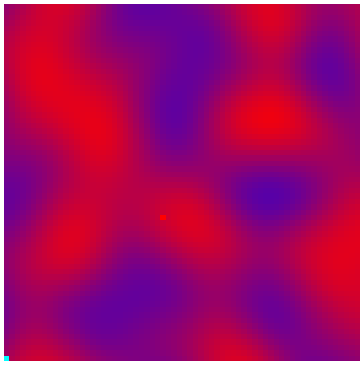

Figura 3.29: Perlin noise per la temperatura.

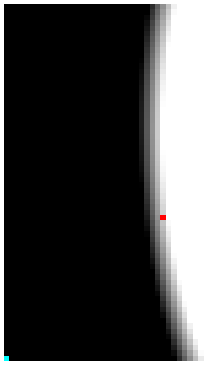

Figura 3.30: Perlin noise per le nuvole. In questo caso, la scala è maggiore per permettere alle condizioni meteo di persistere per un tempo maggiore.

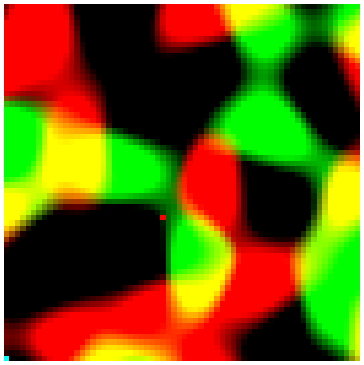

Figura 3.31: Perlin noise per il vento. I canali RGB sono utilizzati per mostrare la direzione del vettore bidimensionale.

Politica dei dati storici. La terza politica consiste nell'utilizzare dati meteoreologici reali, registrati o previsti da servizi esterni specializzati. Nel video-game Shen Mue, i dati erano registrati in forma compatta in una tabella caricata in memoria centrale.
Tuttavia, per garantire al sistema meteo implementato un'alta adattabilità, i dati verranno scaricati in real-time da un servizio web. L'API scelta è Open-Meteo, un servizio open-source e gratuito per uso non commerciale, che li fornisce in formato json con risoluzione temporale oraria e risoluzione spaziale di 1km.
I parametri disponibili per ogni richiesta web sono 45, ma, come mostrato in figura 3.32, solo 7 sono necessari per la presente implementazione. L'URL della richiesta web è poi così costruito:

Algorithm 4 Costruzione dell'URL.

```
1: string uri = "https://archive-api.open-meteo.com/v1/archive?"+
2:              "latitude=" + latitude +
3:              "&longitude=" + longitude +
4:              "&start_date=" + date.ToString("yyyy-MM-dd") +
5:              "&end_date=" + date.ToString("yyyy-MM-dd") +
6:              "&hourly=temperature_2m,rain,snowfall,cloudcover_low,
7:                       cloudcover_mid,windspeed_10m,
8:                       winddirection_10m&windspeed_unit=ms";
```

Figura 3.32: Variabili dell'API disponibili. I parametri richiesti dall'implementazione sono selezionati.

In questo modo, sarà possibile ricevere dati sugli ultimi 60 anni, in tempo reale e previsti, caricando in memoria solamente quelli relativi al giorno specificato in input.
Il json di risposta dal servizio avrà la seguente struttura:


```
{
    "latitude": 52.52,
    "longitude": 13.419998,
    "generationtime_ms": 0.8379220962524414,
    "utc_offset_seconds": 0,
    "timezone": "GMT",
    "timezone_abbreviation": "GMT",
    "elevation": 38.0,
    "hourly_units": {
        "time": "iso8601",
        "temperature_2m": "°C",
        "rain": "mm",
        "snowfall": "cm",
        "cloudcover_low": "%",
        "cloudcover_mid": "%",
        "windspeed_10m": "km/h",
        "winddirection_10m": "°"
    },
```

```
"hourly":
    "time": [...],
    "temperature_2m": [...],
    "rain": [...],
    "snowfall": [...],
    "cloudcover_low": [...],
    "cloudcover_mid": [...],
    "windspeed_10m": [...]
    "winddirection_10m": [...]
```

Al nome del parametro (in "hourly") segue una lista di 24 valori, una per fascia oraria, che possono essere utilizzati per configurare il sistema meteo in base all'ora corrente.

Politica controllata da un parametro. Quest'ultima politica è pensata per semplificare l'interfaccia con il sistema e permettere facilmente agli sviluppatori di ottenere una condizione meteo artisticamente controllabile.
In particolare, viene esposto nell'editor di Unity un parametro di "instabilità" del sistema:

- a valori bassi corrispondono delle condizioni stabili,
- a valori medi corrispondono maggiore nuvolosità e vento,
- a valori alti corrispondono precipitazioni e temporali.

Questa semplificazione risulta efficace soprattutto in specifici casi d'uso del sistema che saranno affrontati in seguito.

Capitolo 4

Casi d'uso

In questo capitolo si analizzano le potenzialità di impiego dell'implementazione proposta.
A tal proposito, si esploreranno vari casi d'uso, tra cui alcuni non convenzionali, con l'obiettivo di individuare diverse modalità di condizionamento dell'utente finale. In pratica, si vuole determinare in che misura può il sistema meteo proposto migliorare l'esperienza di specifici tipi di applicazione interattiva, integrando nuove meccaniche e funzioni.

Segue una lista dei casi d'uso proposti che verranno dettagliati più avanti:

1. Integrazione in un video-game.
2. Impiego nella VGT (video-game therapy), utilizzando sensori EEG.
3. Integrazione in un serious game educativo.

Il sistema meteo proposto, come già visto, si adatta a diversi stili grafici e permette di selezionare una politica di decisione del meteo tra 4 basate sui video giochi di successo analizzati.
Il deployment del sistema su un'applicazione è facilitato grazie a strumenti pensati appositamente: il sistema a eventi, che semplifica la programmazione degli effetti causati dalle condizioni meteo sugli elementi del mondo di gioco, e un'interfaccia che permette la modifica dei parametri della simulazione.
L'obiettivo è rendere l'integrazione del sistema in un video-game intuitiva, e fornire un adeguato controllo per aderire a necessità artistiche e di gameplay variegate.

Sistema a eventi. Il sistema ad eventi abilita gli sviluppatori ad associare una serie di funzioni del video-game a cambiamenti nelle condizioni meteo. Per evitare meccanismi di polling inefficienti, il sistema implementa l'interazione asincrona

attraverso un pattern architetturale di tipo Publish/Subscribe.
In particolare, vi è una classe EventManager che segue il pattern Singleton: può esistere solo un'istanza della classe, il cui riferimento è assegnato a un attributo statico della classe stessa e può essere ottenuto grazie ad una funzione getter. Gli altri attributi sono invece di tipo Action; ognuno di loro corrisponde ad un evento del sistema meteo (ad esempio event Action SunriseEvent), e viene invocato da un'apposita funzione (void StartSunriseEvent()).
Data questa impostazione, il sistema meteo ottiene un riferimento all'unica istanza di EventManager tramite la funzione getter, e lancia gli eventi quando questi si verificano. Ad esempio, quando l'elevazione solare supera l'orizzonte, lancia la funzione EventManager.StartSunriseEvent(). In questo modo, tutte le funzioni del video-gioco sottoscritte all'evento saranno lanciate in cascata.
Per gli sviluppatori, sottoscrivere una funzione ad un evento è semplice: basta utilizzare l'operatore += preceduto dall'attributo Action della classe EventManager e seguito dal nome della funzione desiderata.

Interfaccia. Gli script implementati forniscono delle funzioni per controllarne il comportamento, che coinvolgono, oltre il sistema di meteo, anche i sistemi di gestione della data e dell'ora e quello del ciclo diurno.
Per usarli è sufficiente ottenere una reference a uno dei sistemi sopracitati tramite

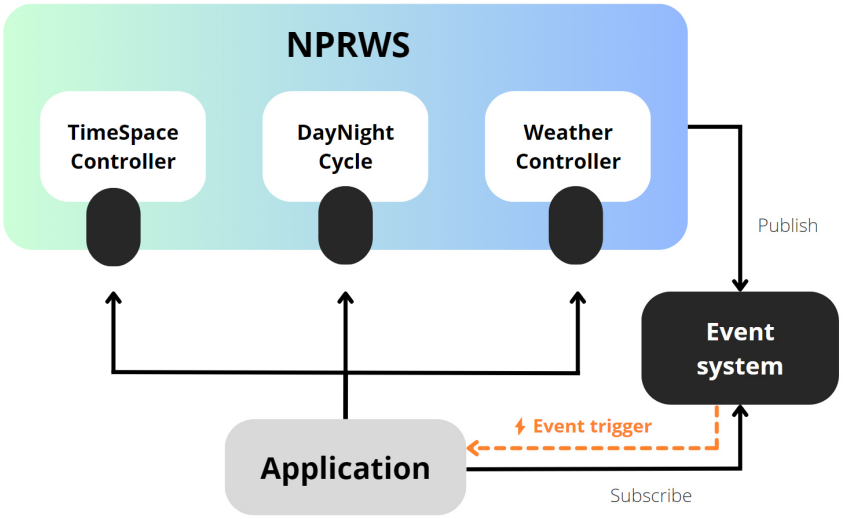

Figura 4.1: Schema dell'interazione tra il plug-in e un'applicazione interattiva sviluppata in Unity.

la funzione FindObjectOfType<T> e utilizzare le funzioni pubbliche esposte. Ad esempio, è possibile generare un fulmine nelle coordinate specificate, qualunque sia la condizione meteo attuale, chiamando il metodo pubblico SpawnLightning(Vector2 position, float height, float scale) dell'istanza di tipo WeatherController.

4.1 Integrazione in un video-game.

Per questo caso di studio, si è integrato il sistema di meteo in una scena di un prototipo di video-game chiamato "A fool's fate". Si tratta di un semplice gioco di ruolo strutturato in puzzle ambientali; in particolare, la scena utilizzata è ambientata sulla cima di una torre, sulla quale il protagonista controllato dal giocatore dovrà affrontare il "boss" del livello utilizzando diverse abilità.
Nel gioco originale, questo momento del gioco prevede un meteo instabile che drammatizza l'esperienza; inoltre, il meteo è un elemento da sfruttare per vincere il livello. Il giocatore deve infatti utilizzare un'abilità sull'armatura del boss in modo da attrarre dei fulmini su di esso per tre volte quando il suo scudo non è attivo. Dunque l'interazione con il sistema di meteo è bidirezionale: esso deve reagire generando un fulmine quando il giocatore utilizza correttamente l'abilità, e il gioco deve reagire quando un fulmine colpisce il boss e il suo scudo non è attivo.
É dunque possibile utilizzare entrambi gli approcci introdotti in precedenza. All'utilizzo dell'abilità, lo script relativo al comportamento del boss ottiene una

Figura 4.2: Scena del prototipo "A fool's fate".

reference al sistema di meteo e chiama la funzione weatherSystem.SpawnLightning (gameObject.transform.position, 0, 0.5f) per generare un fulmine alla sua posizione. Inoltre, lo stesso script può effettuare il Subscribe delle sue funzioni all'evento StartLightningEvent dopo aver ottenuto una reference all'istanza corrente dell'EventManager usando il suo getter. La funzione con cui sarà effettuato il Subscribe esegue il codice responsabile del danneggiamento del boss e il passaggio alla fase successiva.
L'eleganza di questa soluzione è particolarmente evidente se si considera un video gioco più completo in cui esiste più di una meccanica per attrarre fulmini sul nemico: la generazione del fulmine e le sue conseguenze sono disaccoppiate, e questo permette al giocatore di sperimentare nuove soluzioni creative, anche se non previste dagli sviluppatori.

Per variare la difficoltà dell'esperienza di gioco con le condizioni meteo, è stata pensata una regola aggiuntiva. In particolare, un meteo fortemente instabile genererà fulmini senza l'intervento del giocatore: questi possono danneggiare il pavimento limitando l'area percorribile dal protagonista. Anche lo script che gestisce questa regola fa uso del sistema ad eventi.

Infine, è importante notare che l'aspetto visuale del gioco risulta fortemente NPR con elementi in cel-shading. I parametri degli shader sono stati dunque impostati per aderire artisticamente al resto del video gioco. Gli altri parametri, relativi all'ora del giorno e al meteo, permettono di trasmettere facilmente diversi stati d'animo, come mostrato nelle figure 4.3, 4.4 e 4.5.

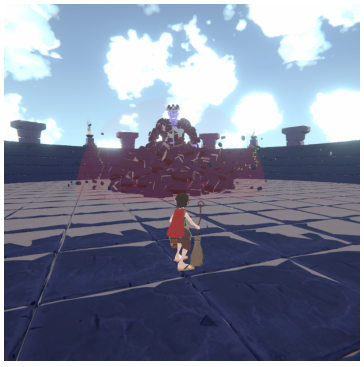

Figura 4.3: Scena del prototipo "A fool's fate" durante il giorno.

Figura 4.4: Scena del prototipo "A fool's fate" durante un tramonto.

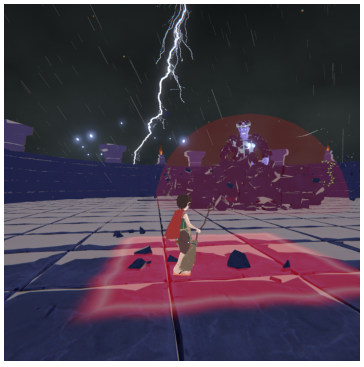

Figura 4.5: Scena del prototipo "A fool's fate" durante una tempesta notturna.

4.2 Impiego nella VGT

La video-game therapy è una forma di terapia che utilizza i video-giochi come strumento per aiutare le persone a superare problemi psicologici o fisici. L'idea alla base della video-game therapy è che i video-giochi possano fornire un ambiente sicuro e controllato per affrontare problemi emotivi o fisici, fornendo al contempo un'esperienza coinvolgente e divertente.

Questo approccio può essere utilizzato per una vasta gamma di problemi, tra cui la depressione, l'ansia, il disturbo da stress post-traumatico, la disabilità fisica, la dipendenza da sostanze e altro ancora. I videogiochi utilizzati in terapia possono essere di diversi tipi, da quelli basati sulla realtà virtuale a quelli basati sulla narrativa.

Molti studi hanno dimostrato l'efficacia della video-game therapy per aiutare le persone a superare problemi emotivi o fisici. Ad esempio, la terapia basata sui videogiochi è stata utilizzata con successo per aiutare i veterani militari a superare il disturbo da stress post-traumatico e per aiutare le persone con paralisi cerebrale a migliorare la loro forza e coordinazione.

Meteoropatia. Il disturbo psicologico di riferimento per questo caso d'uso è detto meteoropatia. Si tratta di una reazione psico-fisica di stress associato a cambi graduali o improvvisi del tempo atmosferico. In pratica, il corpo dei soggetti affetti da questa condizione fa fatica a compensare l'alterazione delle condizioni meteorologiche e avverte dunque un malessere generale: mal di testa, cambi d'umore, cali di pressione, e altre difficoltà psico-fisiche. Questa condizione può essere:

1. Primaria, se non è associata a malattia pre-esistenti. In questo caso è in genere meno grave.
2. Secondaria, se l'alterazione del meteo peggiorano le manifestazioni di altre malattie pre-esistenti. Di solito, gli anziani e gli individui che hanno subito traumi muscolo-scheletrici, cardiaci, o malattie croniche e degenerative sono più inclini a presentare questa problematica.

Quello che accade in questi soggetti può attribuirsi all'ipotalamo, che impartisce ordini sbagliati alle ghiandole neuroendocrine, o al sistema nervoso centrale che manifesta una reazione eccessiva: in quest'ultimo caso, i sintomi principali sono ansia e depressione, e possono essere trattati mediante una tecnica psicologica di esposizione.

Terapia di esposizione. Questa tecnica consiste nell'esporre il soggetto alla situazione che gli genera ansia in modo da metterla nelle condizioni di sperimentare

un cambiamento e mettere alla prova nuove reazioni. L'obiettivo è abituare la mente del paziente a far fronte a questi stimoli.

In questo contesto, la strategia per utilizzare il sistema implementato sarebbe l'esposizione graduata: si elabora una gerarchia di situazioni ansiogene, in ordine incrementale di intensità e si induce il paziente ad affrontare la situazione meno intensa; si passa alla successiva quando non gli causerà più una reazione negativa.

L'implementazione proposta si presta a quest'applicazione; risulta utile, in particolare, la politica basata su un parametro di instabilità. Per generare la gerarchia incrementale basta fissare una serie di valori tra 0 e 1 e poi immergere il soggetto all'interno del mondo virtuale utilizzando un Head-Mounted Display per la realtà virtuale. Il vantaggio di quest'approccio risiede nella possibilità di tenere il paziente in un ambiente controllato, in modo da poter reagire ad eventuali problemi in maniera sicura e tempestiva.

Inoltre, il sistema di interazione bidirezionale fornisce agli sviluppatori un controllo ulteriore che potrebbe essere successivamente esposto al terapeuta responsabile dell'esperienza.

Utilizzo di sensori EEG. Infine, l'utilizzo di sensori potrebbe automatizzare la risposta del sistema all'attività cerebrale dell'utente finale. In particolare, si fa riferimento a dispositivi in grado di rilevare variazioni nell'attività cerebrale conseguenti a stimoli visivi e sonori attraverso l'elettroencefalografia.

Questi dispositivi hanno un numero variabile di elettrodi. Ognuno di essi produce un segnale tempo variante associato all'intensità di corrente dell'area di scalpo ad esso sottesa.

Analizzando i segnali in frequenza, ad esempio usando una trasformata DCT, si distinguono diverse componenti delle onde cerebrali:

1. Delta (da 1 a 3 Hz), legate al sonno profondo.

2. Theta (da 3,5 a 8 Hz), associate all'immaginazione e emozioni profonde. I picchi alti possono essere segno di un disturbo depressivo, mentre i picchi bassi si verificano con l'ansia e lo stress.

3. Alfa (da 8 a 13 Hz), rappresentano lo stato di agitazione del soggetto. Anche in questo caso, un basso livello arriva con ansia e stress.

Figura 4.6: Neurosky Mindwave, un dispositivo EEG differenziale di livello consumer con due canali, di cui uno di riferimento. Frequenza di campionamento: 512Hz.

4. Beta (da 12 a 33 Hz). Le frequenze più alte sono legate a fenomeni più interessanti ma complessi: queste frequenze si presentano in attività comuni che richiedono tutta l'attenzione del soggetto.

5. Gamma (da 25 a 100 Hz). Sono legate a un'alta elaborazione cognitiva, come accade durante l'assimilazione di nuove informazioni o durante i picchi di felicità.

In questo contesto, è possibile monitorare le frequenze da 3,5 a 13 Hz (Theta e Alfa) e rilevare eventuali picchi di massimo e minimo, che forniscono informazioni sui due sintomi principali della meteoropatia nei soggetti in cui il sistema nervoso

presenta risposte eccessive agli stimoli: ansia per i livelli bassi, e depressione per i livelli alti.
Queste informazioni possono sia essere fornite in output per il terapeuta, sia essere associate ad una risposta automatica del sistema. Ad esempio, se l'ansia misurata supera una soglia predefinita, il sistema può tornare lentamente allo stadio precedente della gerarchia per compensare la problematica e favorire uno stato di rilassamento.

4.3 Integrazione in un serious game educativo.

Un serious game è un videogioco progettato con l'obiettivo di formare o sensibilizzare i giocatori su un determinato argomento. A differenza dei videogiochi tradizionali, che mirano principalmente a intrattenere, i serious game utilizzano l'esperienza di gioco per raggiungere obiettivi educativi o sociali.

I serious game possono essere utilizzati in molti contesti diversi, come la formazione aziendale, la prevenzione della salute, la simulazione di situazioni reali o la promozione di comportamenti sostenibili. Possono anche essere utilizzati per aiutare le persone a sviluppare competenze specifiche, come la risoluzione dei problemi, la collaborazione, la gestione del tempo e la leadership.

In particolare, l'implementazione proposta potrebbe essere impiegata nelle seguenti modalità:

Simulazione di situazioni reali. L'obiettivo di questo caso d'uso sarebbe insegnare agli utenti semplici linee guida per prevedere condizioni meteo avverse imminenti e come i fenomeni meteorologici influenzano l'ambiente e la vita quotidiana. Questa formazione potrebbe essere veicolata da un gioco di sopravvivenza in cui i giocatori devono affrontare situazioni meteorologiche variabili.

Videogioco gestionale. Il sistema potrebbe essere utilizzato per simulare le conseguenze dei cambiamenti climatici in un gioco educativo che vuole sensibilizzare i giocatori sulle tematiche ambientali. Ad esempio, la simulazione potrebbe mostrare gli effetti di un aumento delle temperature sulla vegetazione e sugli animali, o sulle infrastrutture e sulla vita delle persone. Un altra problematica ambientale che acquisisce sempre più rilevanza è rappresentata dagli incendi: risulterebbe interessante provare a rendere i cittadini comuni consci della relazione tra il clima e il rischio incendi. Infatti, esistono metodi matematici per calcolare un indice di pericolo per incendi boschivi sulla base di parametri disponibili nel sistema di meteo implementato [7], tra cui l'umidità relativa, la temperatura e il vento.

Simulazioni di volo. Il sistema di meteo potrebbe anche essere utilizzato per creare un'esperienza di gioco più immersiva e realistica in giochi di simulazione, come ad esempio i simulatori di volo o di guida. In questo modo, i giocatori potrebbero avere una migliore comprensione di come il clima influisce sulle loro attività e su come affrontare le diverse situazioni meteorologiche. Per avere la massima accuratezza sull'evoluzione degli eventi atmosferici è preferibile utilizzare la politica basata su dati reali: in questo modo, sarebbe anche possibile far esercitare gli utenti su un fenomeno meteorologico particolare realmente accaduto.

Capitolo 5

Conclusioni e sviluppi futuri

5.1 Conclusioni

In questa sezione si riassumerà l'analisi dello stato dell'arte e l'implementazione proposta, evidenziando in che misura è in grado di colmare le carenze dei titoli analizzati come riferimento. Una necessità aggiuntiva del sistema proposto è l'adattabilità e versatilità, e deriva dal fatto che è implementata in forma di plug-in per un game engine come Unity in modo da poter essere utilizzata dagli sviluppatori di diverse applicazioni per accelerare i tempi di lavoro.

Si rammenta che l'analisi dei vantaggi e svantaggi è riferita al senso di presenza (senso di appartenenza al mondo virtuale) che il sistema è in grado di garantire, attraverso il realismo degli eventi e la coerenza artistica, nonché il livello di controllo sul sistema stesso.

Le caratteristiche migliori individuate sono riassumibili nei seguenti punti:

Realismo degli eventi meteorologici. Fare in modo che le condizioni meteorologiche siano plausibili concorre a rendere il sistema più realistico. Uno dei fattori che migliora il senso di presenza è infatti l'illusione di plausibilità, che si verifica quando gli eventi simulati sono percepiti da una persona il più vicino possibile alla realtà [8].
Evitare i cambiamenti improvvisi è un altro elemento da tenere in considerazione quando si progetta un sistema di meteo. Nella realtà, tutte le transizioni meteo sono distribuite nel tempo. Inoltre, non dipendono necessariamente da aree spaziali definite: è anche preferibile evitare di limitare la simulazione in aree ben delineate o le transizioni appariranno prevedibili e innaturali, nonostante permettano di ottimizzare molto l'elaborazione.

La variabilità deve anche essere visiva. Ad esempio, è impossibile nella realtà osservare due nubi identiche, e portare questa caratteristica all'interno dell'applicazione migliorerà ulteriormente la sua credibilità.

Coerenza grafica e controllo artistico. Questi due elementi sono necessari per migliorare il coinvolgimento dell'utente, ovvero catturano la sua attenzione rendendo il contenuto multimediale interessante. Il coinvolgimento rende senza dubbio più facile l'estraniamento dal contesto reale favorendo l'immersione.

Interazione bidirezionale. Anche il livello di interattività, come già spiegato nel capitolo 1, incide direttamente sul parametro della presenza. Risulta particolarmente efficace introdurre nell'esperienza diversi effetti delle condizioni meteo sul giocatore o sul mondo di gioco e viceversa. In quest'ultimo caso, la scelta risulta piuttosto situazionale: il condizionamento del meteo non può che essere causato da meccaniche di gioco a larga scala o appartenenti al genere fantasy.

Adattamento al mondo di gioco. Questa caratteristica non è evinta dall'analisi dello stato dell'arte, ma è una necessità dell'implementazione descritta, che si propone di essere impiegata e personalizzata dagli sviluppatori in forma di plug-in. Anche per questo motivo, non è stato favorito l'approccio di suddivisione del mondo di gioco in aree predefinite.

É stato dunque sviluppato un sistema meteo dinamico che copre tutte queste caratteristiche. Le funzionalità principali includono il rendering di nubi volumetriche NPR con illuminazione physics-based, la politica di decisione del meteo physics-based, e un sistema di interfacce per la realizzazione di interazioni bidirezionali.

In particolare, il primo modulo realizzato permette di simulare il ciclo diurno, calcolando la posizione del Sole e della Luna rispetto coordinate spazio-temporali fornite in input. Il modulo fa uso di algoritmi astronomici approssimati ma aventi un margine di errore sufficientemente basso.

Segue l'implementazione di shader ed effetti visuali per rappresentare graficamente le condizioni meteorologiche. Quelli descritti riguardano il rendering delle nubi, che permette un illuminazione realistica e una resa finale regolabile per più stili; e il rendering dei fulmini, che grazie all'utilizzo del formato Alembic possono raggiungere un alto numero di ottave di rumore.

In seguito, si implementa la politica di decisione del meteo physics-based, che

utilizza concetti di fluido-dinamica per evolvere parametri meteorologici inizializzati facendo riferimento alla struttura morfologica del mondo virtuale. Per garantire la sua dinamicità, si utilizzano variazioni casuali ai limiti della mappa di gioco.
Vengono anche esplorate altre politiche di meteo basate sullo stato dell'arte ma che presentano delle migliorie. É stata realizzata una politica casuale, che mantiene l'intera mappa come dominio di simulazione, dando la possibilità all'utente di prevedere le condizioni imminenti; una politica basata su dati reali, che tramite richieste web ad un database remoto permette il massimo livello di realismo; e una politica basata su un solo parametro di instabilità.

Infine, si è reso l'utilizzo del plug-in intuitivo utilizzando un approccio ad eventi di tipo Publish/Subscribe e delle interfacce per configurarne i parametri dall'esterno. Grazie a questi moduli aggiuntivi, si è integrato il sistema all'interno di una scena prototipale di un videogioco, "A fool's fate", mostrando l'immediatezza di sviluppo di logiche interattive bidirezionali, nonché l'adattamento grafico e la capacità di ottenere un atmosfera diversa a seconda del meteo e dell'ora del giorno.

Il sistema potrebbe inoltre essere impiegato in diversi casi d'uso proposti: nella VGT, per esempio come trattamento di disturbi psicologici tra cui la meteoropatia, e all'interno di videogiochi educativi.
Nel primo caso si propone una terapia di esposizione, in cui il paziente è tenuto ad affrontare la causa del proprio stress in un ambiente controllato e sicuro; dei sensori EEG potrebbero supportare il percorso registrando l'attività cerebrale dell'utente.
Nel secondo caso, l'obiettivo dell'applicazione potrebbe spaziare dall'insegnamento di metodi pratici per prevedere condizioni meteo avverse alla sensibilizzazione del cittadino comune riguardo l'effetto dell'aumento delle temperature sulla natura e le attività umane. La simulazione in VR rappresenta un'altra potenziale modalità di impiego del sistema meteo, per esempio per esercitare autisti di veicoli o aeromobili a far fronte a diverse condizioni meteo.

5.2 Sviluppi futuri

Nonostante il presente lavoro di tesi proponga, come già visto, diverse tecniche per far fronte alle limitazioni dello stato dell'arte, lascia spazio per annettere nuove funzionalità che ne migliorino ulteriormente il realismo e la versatilità.
Infatti, la meteorologia è una materia molto complessa che coinvolge una quantità di variabili elevatissima: per questo motivo, anche i fenomeni atmosferici possono essere sensibilmente vari e unici.
Inoltre, risulta di importanza centrale migliorare le prestazioni del sistema per l'esecuzione dello stesso in real-time su un ampio range di macchine con diverse

specifiche tecniche.

Alla luce di queste considerazioni, si analizzeranno di seguito due macro-categorie di possibili migliorie da implementare nei lavori futuri previsti per il sistema realizzato.

Funzionalità aggiuntive.

1. Integrazione dei calcoli e shader relativi alla fase lunare.
2. Annessione di altri fenomeni atmosferici: tempeste di sabbia, grandine, aurora boreale, arcobaleni.
3. Implementazione di uno shader physics-based per il cielo che modelli lo scattering atmosferico.
4. Annessione di altre politiche di decisione del meteo.
5. Annessione di interazioni standard del sistema meteo con il mondo virtuale, ad esempio la posa della neve, o la formazione di pozzanghere.
6. Integrazione del vento con il sistema fisico di Unity.

Funzionalità migliorate.

1. Ottimizzare l'esecuzione dei moduli del ciclo diurno e del sistema di meteo. In particolare, l'aggiornamento della posizione del Sole e della Luna e della simulazione di meteo avviene con una frequenza regolabile. Per migliorare le prestazioni, è possibile diminuire di molto questa frequenza e calcolare le condizioni intermedie per interpolazione. Distribuendo l'elaborazione sui frame tra due aggiornamenti con le co-routine di Unity si riduce drasticamente il carico computazionale, permettendo l'implementazione di ulteriori funzioni per aumentare il realismo generale.
2. Annessione di altre simulazioni di fluidi corrispondenti a più livelli di altitudine. Modellare accuratamente le dinamiche verticali della formazione delle nubi è la chiave per portare il sistema al livello successivo di realismo.
3. Permettere agli sviluppatori di utilizzare più politiche nello stesso momento, e assegnare dinamicamente dei pesi a ciascuna di esse per avere il maggior controllo possibile.
4. Fornire agli sviluppatori un maggiore controllo del dominio di simulazione, ad esempio permettendo di limitarne l'estensione o utilizzare delle griglie non uniformi, in modo da ottimizzare la precisione del sistema dove è necessario.

Inoltre, si potrebbero fornire strumenti per la suddivisione del mondo virtuale in aree, e sviluppare una politica con più domini di simulazione di forma e risoluzione arbitraria.

Bibliografia

[1] John A. Duffie e William A. Beckman. «Fundamentals.» In: Solar Engineering of Thermal Processes (2013), pp. 1–25 (cit. a p. 21).

[2] Meeus Jean. «Astronomical Algorithms.» In: (1998) (cit. a p. 24).

[3] *Istituto Superiore di Sanità, EpiCentro - L'epidemiologia per la sanità pubblica.* URL: `https://www.epicentro.iss.it/fulmini/` (cit. a p. 40).

[4] *Tuoni e fulmini! I fenomeni elettrici dell'atmosfera.* URL: `http://www.centrometeo.com/articoli-reportage-approfondimenti/fisica-atmosferica/4298-tuoni-fulmini-atmosfera` (cit. a p. 41).

[5] Mark G. Lawrence. «The Relationship between Relative Humidity and the Dewpoint Temperature in Moist Air: A Simple Conversion and Applications.» In: (feb. 2005) (cit. a p. 57).

[6] Oleg A. Alduchov e Robert E. Eskridge. «Improved Magnus Form Approximation of Saturation Vapor Pressure.» In: (apr. 1996) (cit. a p. 58).

[7] T.L. Pickett C.E. Van Wagner. «Equation and FORTRAN Program for the Canadian Forest Fire Weather Index System.» In: (1985) (cit. a p. 71).

[8] *Immersione e mondi immersivi: La sfida del presente.* URL: `https://www.cadschroer.it/novita/articoli/immersione-e-mondi-immersivi-la-sfida-del-presente/` (cit. a p. 73).

[9] Kalogirou Soteris. «Environmental Characteristics.» In: Solar Energy Engineering: Processes and Systems (2014), pp. 49–71.

[10] William B. Stine e Michael Geyer. «Power From The Sun.» In: Solar Geometry (2001).

[11] Montenbruck Oliver e Pfleger Thomas. «Astronomy on the Personal Computer.» In: (2004).

[12] Fredrik Haggstrom. «Real-time rendering of volumetric clouds». Tesi di laurea mag. Cambridge: Umeå Universitet, 2018.

[13] Patricio Gonzalez Vivo e Jen Lowe. «The Book of Shaders.» In: Fractal Brownian Motion (2015).

[14] Alexander Majercik, Cyril Crassin, Peter Shirley, Morgan McGuire, NVIDIA e Williams College. «Journal of Computer Graphics Techniques.» In: 7. A Ray-Box Intersection Algorithm and Efficient Dynamic Voxel Rendering (2018).

[15] D.Sonnatag. «The history of formulations and measurements of saturation water vapour pressure.» In: Papers and abstracts from the Third International Symposium of Humidity and Moisture, vol.1, Teddington (1998).

[16] B.Hardy. «Formulations for vapur pressure, frostpoint, temperature, dewpoint temperature, and enhancement factors in range -100°C to 100°C.» In: Papers and abstracts from the Third International Symposium of Humidity and Moisture, vol.1, Teddington (1998).