# Assets

La cartella env ha tutte le cubemap.

Labirths ha i JSON della struttura del labirinto.

Models ha i modelli di torcia+mano, scimmia e piedistallo.

## Shaders:

character:

nel vertex shader usiamo MVP per determinare GL position e passiamo normali in world space + surface to view.

Nel fragment calcoliamo eyeDIr, i colori con mix, usiamo Lambert (abbiamo sia componente principale che un vettore fornito dal modelli per regolare la diffuse) come diffuse e Blinn come specular.

envDayNight

gestiamo la skybox mettendo i vertici direttamente in clip space

nel fs utilizziamo la inversa VP per trovare la direzione in cui mappiamo la cubemap.

Models

Lo utilizzano la scimmia e il piedistallo, è simile a quello del character ma li abbiamo divisi perché i models hanno anche il contributo di spotlight, mentre è impossibile per il character.

La luce della spotlight calcola theta e epsilon utilizzando le due misure in coseni che utilizziamo, poi clampiamo (otteniamo una specie di point light tagliata al di fuori del cono), poi calcoliamo distanze e attenuazione con coefficienti costanti, lineari e quadratici.

Pbr: è utilizzato per pareti e pavimento, utilizza le texture di pbrfree

Nel vertex cose normali (le uv vanno da 00 a 11 in ogni quadrato).

Nel fragment casino

Il pbr si basa sulle microfacet e conservazione di energia.

Usa cook torrance brdf, divise per contribuzione di direct e spot.

Come diffuse usa il lambertian che divide albedo per pigreco, ed è moltiplicato per il coefficient kd (percentuale di luce che viene refratto), come specular usiamo cook torrance.

A Numerator abbiamo D, F, G.

D è la normal distribution, che approssima l’area relativa di microfacets allineate con l’halfway vector ( a metà tra luce e vista). Utilizza alpha che si basa sulle roughness. In codice abbiamo tradotto la Trowbridge-Reitz GGX normal distribution.

G è la geometry function che serve a descrivere il self shadowing.

Usa SchlickGGX con Smith Method (non conosciamo i dettagli matematici ma abbiamo tradotto le formule in codice).

F è la Fresnel (si pronuncia Freh-nel), descrive il rapporto di luce riflesso rispetto a quello refratto, dipendente dall’angolo dal quale stiamo guardando la superficie.

Usiamo la Fresnelk-Schick approxximation, dove F0 rappresenta la reflettivita della superficie ( si basa sulla metalness).

Utilizziamo delle texture prese dal sito.

Albedo rappresenta il colore principale, e andiamo a trasporre dalla texture ad un vec3 rgb.

Normal è la normal map della texture, utilizziamo una funzione che ricava un vec3 dalla texture utilizzando derivate.

Metallic lo traduciamo su un float prelevando solo r channel e lo utilizzeremo per regolare il mix di F0 e altri rapport di riflessivita (un materiale completamente metallico riflette totalmente).

La rougness è un valore float che stabilisce la casualità di microfacet orientation.

AO è ambient occlusion e rappresenta un extra shadowing per l’ambient light.

Alla fine utilizziamo una gamma correction che serve a dare un effetto realistico.

# Texture

I paccheti di texture per il pbr.

# Components

LabyrinthModel è la classe che rappresenta un modello (Suzanne, Pedestal).

Usa delle funzioni nel costruttore per parsare i file .obj e mtl, otendendo dal primo vertici, indici, normali e uv (se presenti), dall’mtl otteniamo proprietà del materia (emission, specular,…).

Quando carichiamo i vao e quando disegnamo è importante usare il ciclo per ogni geometry, che serve nel caso ci siano gruppi di mesh in un solo modello.

La classe ereditando la PbrElement è nello Scene Graph come figlio del labirinto.

Pbr floor/wall

Cambia solo il calcolo compute 3d walls/ floor.

Creano vao, poi usano solo funzioni ereditate.

Pedestal / Suzanne

Estendo labyrinthModel, settano i loro parametri fondamentali e usano position model computando la local matrix (poi grazie alla struttura dello scene graph verrano mossi alla loro posizione all’interno di quello effettiva del labirinto).

Suzanne si attiva a fine livello con una animazione.

Utilizza interpolazione lineare, macendo un movimento graduale ad ogni istante fino a raggiungere il character.

Camera

Setta posizioni, direzioni,.. caratteristiche di ogni livello.

Va anche a costruire perspective e view.

L’angle invece viene definito dalla computazione del labirinto.

Gestisce poi i suoi parametri in base ai comandi di movimento che riceve dal movement handler.

La funzione di idle viene continuamente chiamanta, computa i parametri e aggiorna le matrici

Compute position utilizza anche il collision handler, prevendo un movimento se sta per collidere con una parete.

Character

Draw e Vao funzionano simili a Suzanne, non lo abbiamo però voluto inserire nel scene graph (in futuro potrebbe servire trasformare il labirinto senza che il personaggio sia coinvolto.

La compute light usa coordinate sferiche per determinare il punto da cui parte la luce della torcia, mentre la direzione della luce è determinata con un offset statico.

Level

Contiene tutti gli oggeti di un livello (comodo per la gestione essendo multilivello)

Permette la draw di tutti gli oggeti.

PbrLabyrinth

Gestisce lo scene graph del labirinto, gestondo inizializzazione e disegno dei figli.

PbrLabyrinthElemtn

Padre di Floor e Wall, gestisce l’update della world matrix ricorsivamente, e la draw utilizzando gli sshader di PBR

PbrTexture

Modella un pacchetto texture pBR (Albedo, ao…) e ne gestisce il caricamento e dei parametri di utilizzo (gl:TEXTURE per ognuna).

Skybox

Gestisce la skybox con i suoi due pacchetti di texture (giorno, notte), passando ciò che serve allo shader.

Le vert Posizion sono passate in clip space.

SkyboxTexture

Un pacchetto dei due di skybox (o giorno o notte), gestisce caricamento di immagini e creazione di CubeMap.

# Handlers

Collision Handler

Ricevendo la struttura del labirinto usa una funzione di utilità per estrapolare intrappolando la camera in una matrice 2D (ricava i muri, ne crea un bordo con un offset e capisce se la prossima mossa della camera è valida).

Controlla in maniera particolare gli angoli del muro che necessitavano un trattamento univoco.

Bisogna anche tenere in considerazione i bordi laterali in vista di una possibile rotazione (che è sempre concessa).

Interaction Handler

Setta dei listenere per interazione di tasti e mouse

Level Handler

Controlla lo stato di avanzamento di livello

Movement Handler

Associa alle interazioni con tasti e mouse una funzione di movimento della camera o di altre azioni di gioco (accendimento della torcia)

Pbr loading handler

Si occupa del caricamento di tutti gli elementi di gioco, referenziando i json, complindo gli shader, passando programmi e dati ai costruttori,.. fino ad avere costruito un livello.

Si occupa anche del caricamento di un livello successivo a quello attuale.

Scene Handler

Gestisce la draw scene, bindata alla request animation frame, chiamando quindi idle degli handler e la draw del livello.

A fine livello abilita la funzione di animate che impedisce i movimenti e gestisce traslazione del personaggio nella posizione dove arriva la scimmia.

Quando accade questo si passa per qualche istante da un gioco ad una specie di video, cedendo poi il controllo al main.

# Utils

Labyrinth Utils

Contiene parametri di debug per la generazione della struttura e delle enumeration.

Le funzioni chiamabili sono compute3dFloor/Wall che dalla struttura 2D va a generare una struttura 3d con vertici, indici, normali, uv e, se richiesto, anche colori.

L’idea è che da ogni cella, guardi il tutte le direzioni per capire se hai un muro o no, settando vertici se è presente.

Il labirinto 3D viene centrato nello start point del labirinto2d

Il generate 2d labirinth, prende in ingresso numero di righe, colonne e la probabilità di rumore (muri vengono rotti introducendo potenziali loop).

Determina una start position, poi dalle 4 potenziali direzioni da cui partire rompe muri e genera path, poi ripete ricorsivamente, per l’algoritmo se siamo in una cella e dall’altro lato del muro c’è un altro path, il muro non dovrebbe essere rotto, il rumore introduce appunto una probabilità che questo muro venga comunque rotto.

Infine va a computare la finale position dove posizionare Suzanne, che utilizza BFS per trovare il punto più lontano.

Le altre funzioni sono utilitarie per interfacciare labirinto 2d con quello 3D, per esempio il collision handler ne utilizza alcune, anche traducendo posizioni e inclinazioni nel mondo 3d

Utils sono quelle del prof, sono state aggiunte funzioni di lerp, lettura da json di enum, parsing di material e object e getGeometriesExtents, utilizzando per determinare i limiti di posizione di un model, utile per trovare il suo centro nelle 3 dimensioni.

Script

È il main del programma, crea gli handler, contesto gl e cede il controllo allo scene handler

Settings

Devinisce ordine del level ordinando i loro json

Setting Specifici

Tutte le informazioni del livello (posizioni, texture, skybox, shaders, struttura)