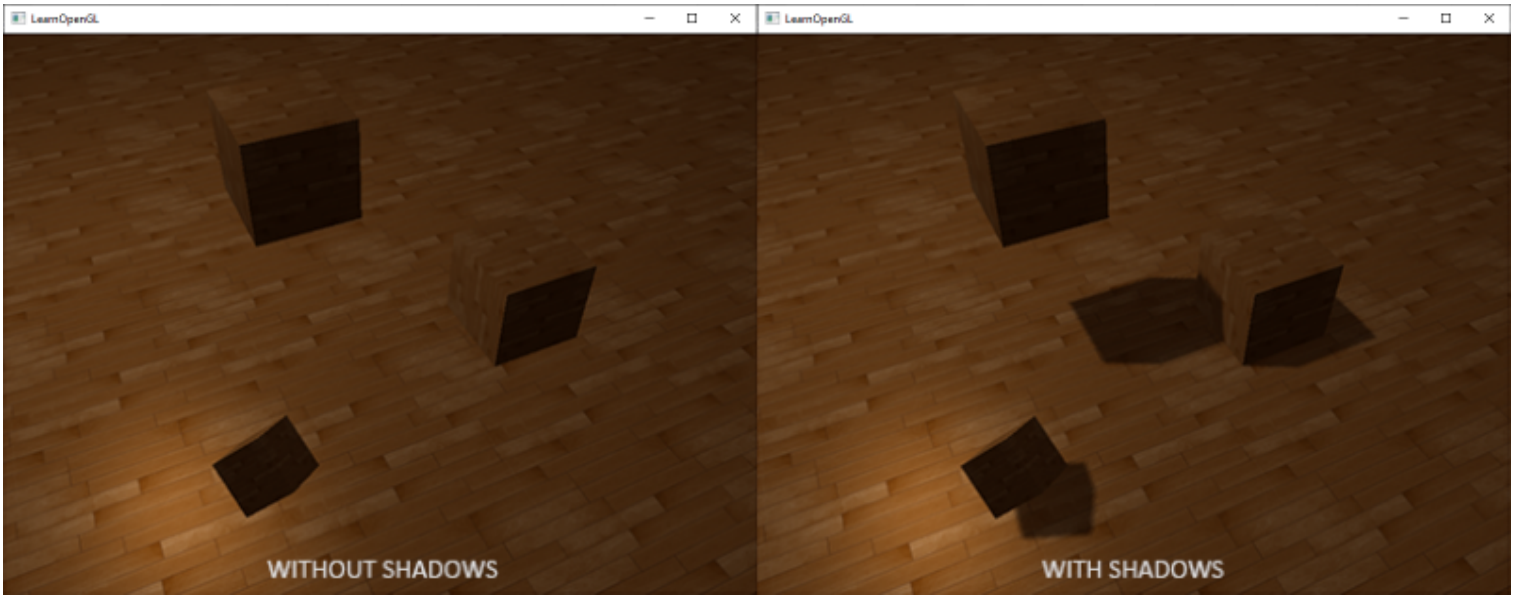
Programmazione Grafica

Esercitazione 2 - Shadow Mapping

Giacomo Mirandola - VR475440

Università degli Studi di Verona
Laurea Magistrale in Ingegneria e Scienze Informatiche

Le ombre sono il risultato dell'assenza di luce dovuta all'occlusione.
Quando io posiziono una fonte di luce e i suoi raggi luminosi non colpiscono un oggetto perché viene sovrapposto / oscurato da un altro oggetto, l'oggetto risultante si troverà totalmente o parzialmente in ombra.
In un film, video, immagine o gioco le ombre giocano un ruolo fondamentale, portando un notevole grado di realismo a una scena illuminata e rendono più facile per lo spettatore osservare le relazioni spaziali tra gli oggetti, quindi distanze e prospettive.
Forniscono una maggiore sensazione di profondità alla nostra scena e agli oggetti.



Come si nota in questo esempio, le ombre fanno diventare molto più evidente come gli oggetti sono correlati tra loro. Ad esempio, il fatto che uno dei cubi stia fluttuando sopra gli altri è notabile solo quando abbiamo le ombre.

Nel corso degli anni sono state sviluppate alcune tecniche per l'approssimazione delle ombre, ma cè da sottolineare che non sono ancora tecniche perfette e presentano anche alcuni "bug".

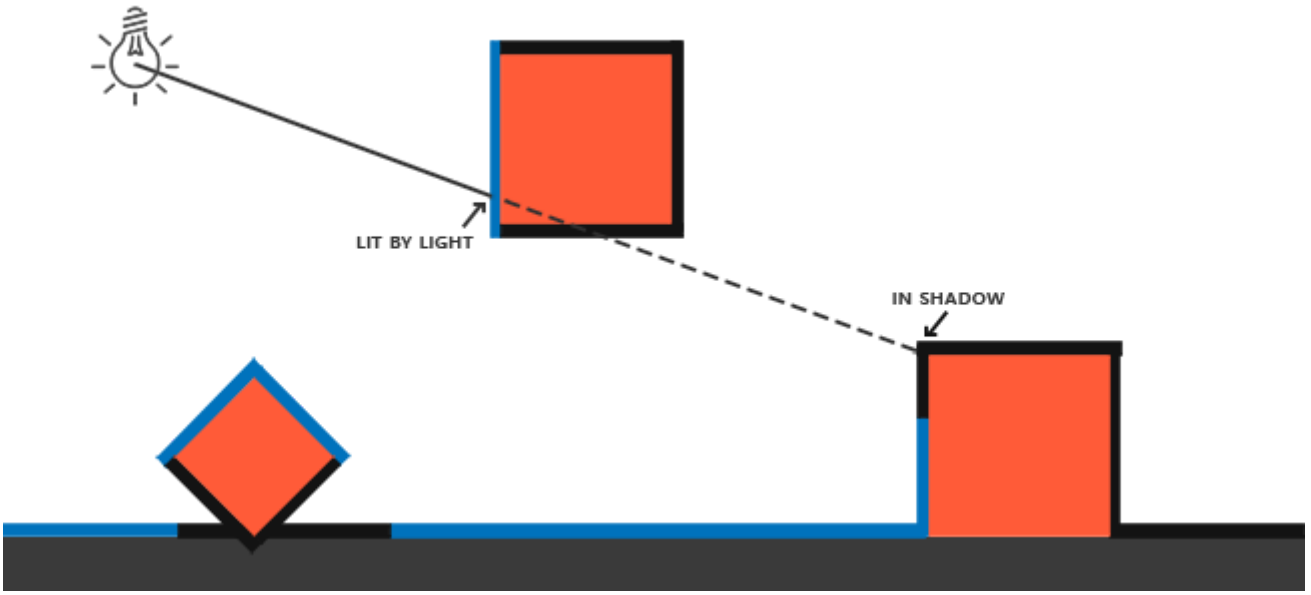
Ci sono diverse buone tecniche di approssimazione delle ombre, ma tutte hanno le loro piccole particolarità e fastidi che dobbiamo tenere in considerazione.

Una tecnica utilizzata dalla maggior parte dei videogiochi che offre risultati decenti ed è relativamente facile da implementare è il "shadow mapping" che è abbastanza facilmente estendibile in algoritmi più avanzati (come le "Omnidirectional Shadow Maps" e le "Cascaded Shadow Maps").

Shadow mapping

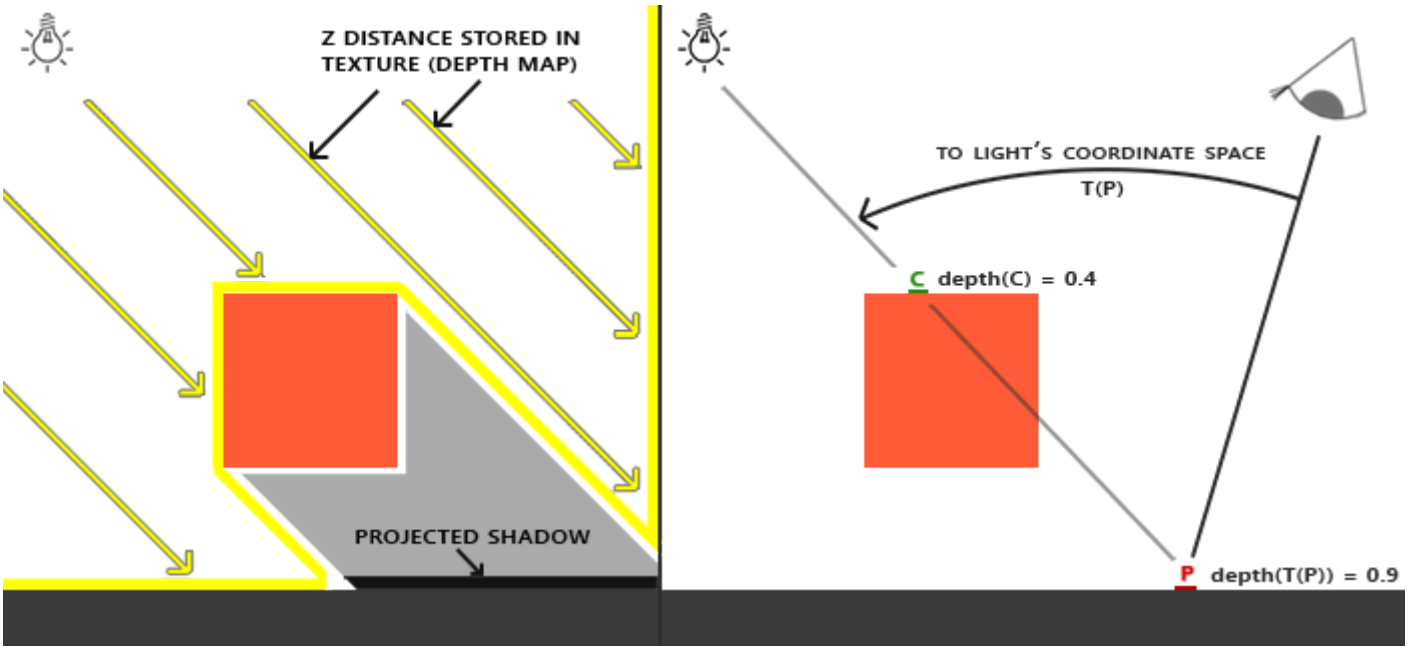
Una tecnica utilizzata dalla maggior parte dei videogiochi che offre discreti risultati e al contempo è relativamente facile da implementare è lo "shadow mapping".

L'idea alla base dello "shadow mapping" è la seguente: facciamo il render della scena dal punto di vista della luce e tutto ciò che vediamo dalla prospettiva della luce viene illuminato, mentre tutto ciò che non possiamo vedere deve essere in ombra.



Qui tutte le linee blu rappresentano i frammenti che la fonte di luce può vedere. I frammenti oscurati sono mostrati come linee nere: questi vengono renderizzati come se fossero in ombra. Se dovessimo disegnare una linea o un raggio dalla fonte di luce a un frammento sulla scatola più a destra, vediamo che il raggio colpisce prima il contenitore fluttuante prima di colpire il contenitore più a destra. Di conseguenza, il frammento del contenitore fluttuante è illuminato e il frammento del contenitore più a destra non è illuminato e quindi in ombra.

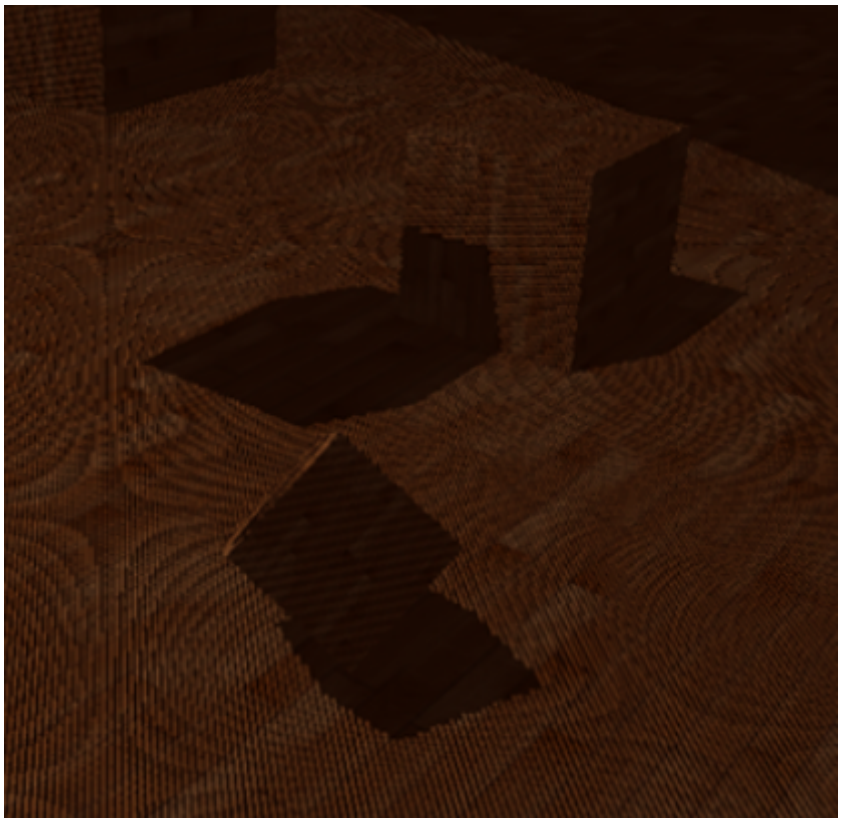
Quello che vogliamo è trovare il punto sul raggio di luce dove colpisce il primo oggetto e confrontare questo closest point con altri punti sul raggio di luce.
Per fare ciò è possibile vedere se la posizione di un punto di test è più lontana nel raggio rispetto al closest point e se è così, il punto di test deve essere in ombra.
Iterare su migliaia di raggi di luce che partono da una sorgente è estremamente inefficiente, quindi invece usiamo un concetto familiare, il depth buffer.
Ricordiamo che un valore all'interno del depth buffer corrisponde ad un frammento troncato a [0,1] dal punto di vista della camera, quindi in questo caso i valori di profondità mostrano il primo frammento visibile dalla prospettiva della luce. Memorizziamo tutti questi valori in una texture che chiamiamo depth map o shadow map.



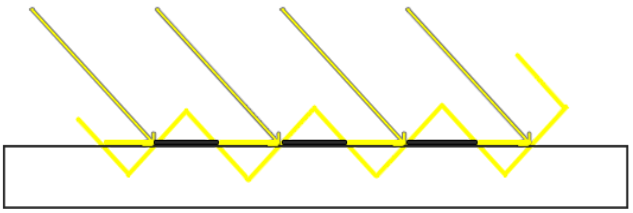
L'immagine di sinistra mostra una sorgente di luce direzionale che proietta un'ombra sulla superficie sotto il cubo. Utilizzando i valori memorizzati nella depth map troviamo il closest point e lo utilizziamo per determinare i frammenti in ombra. Creiamo la depth map effettuando il render della scena (dalla prospettiva della luce) utilizzando view matrix e proiezione specifiche per quella sorgente di luce. Queste matrici insieme formano una trasformata T che trasforma qualsiasi posizione 3D in light coordinate space.
Nell'immagine di destra vediamo la stessa luce direzionale e l'osservatore. Effettuiamo il render di un frammento al punto P per il quale dobbiamo determinare se è in ombra. Per farlo, prima trasformiamo il punto P in coordinate dello spazio luce utilizzando T. Dato che il punto P è adesso nella prospettiva della luce, la sua coordinata z corrisponde alla sua profondità, che in questo esempio è di 0.9.
Utilizzando il punto P possiamo anche indicizzare la depth/shadow map per ottenere la più vicina profondità visibile dalla prospettiva della luce, che corrisponde al punto C con una profondità di 0.4.
Dato che indicizzare la depth map ritorna un valore di profondità minore della profondità al punto P, possiamo concludere che il punto P è in ombra.

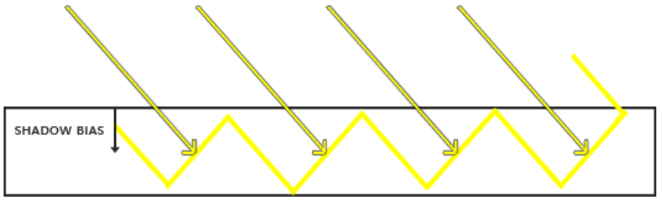
Problematiche delle Shadow Maps

- Shadow acne → Se si analizza attentamente la soluzione realizzata si potrà notare quello che viene chiamato Moiré-like pattern:



Possiamo vedere che parte del piano del pavimento viene renderizzato con chiare linee nere in un modo alternato, questo problema delle mappatura delle ombre è chiamato "shadow acne".



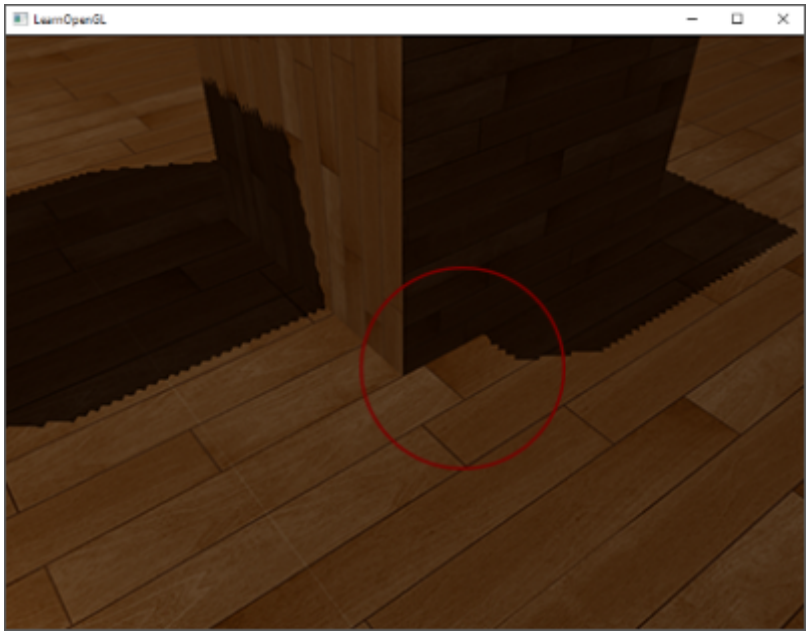


Poiché la shadow map è limitata dalla risoluzione, più fragment possono campionare lo stesso valore dalla mappa di profondità quando sono relativamente lontani dalla sorgente di luce. L'immagine mostra il pavimento in cui ogni pannello giallo inclinato rappresenta un singolo texel della mappa di profondità. Come puoi vedere, diversi frammenti campionano lo stesso valore di profondità.

Sebbene questo sia in generale accettabile, diventa un problema quando la sorgente di luce guarda la superficie da un angolo, poiché in questo caso la depth map viene anch'essa renderizzata da un angolo. In questo caso, diversi frammenti accedono allo stesso tilted depth texel, mentre alcuni sono sopra e alcuni sotto il pavimento; ottenendo una discrepanza nell'ombra. A causa di ciò, alcuni frammenti vengono considerati in ombra e alcuni no, causando la generazione di strisce nell'immagine.

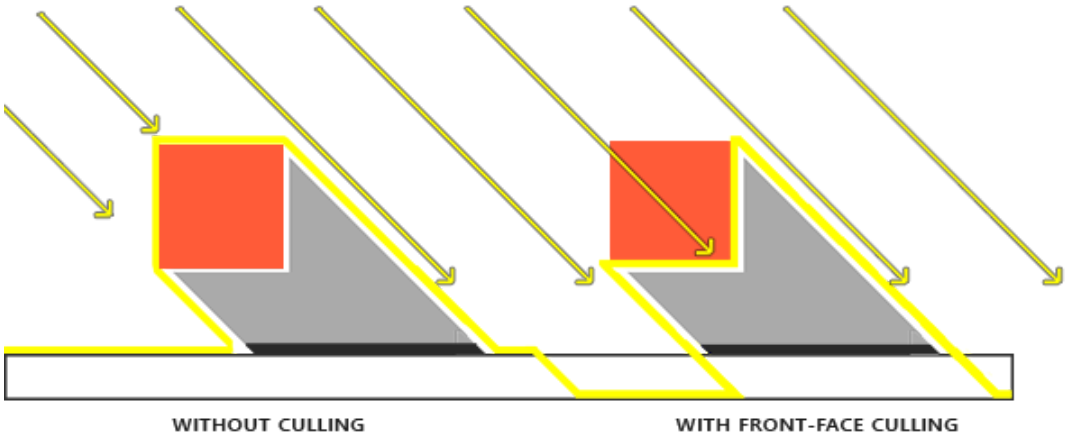
Possiamo risolvere questo problema con un workaround chiamato "shadow bias", in cui semplicemente spostiamo la profondità della superficie (o della shadow map) di una piccola quantità di bias in modo che i fragment non vengano considerati erroneamente al di sopra della superficie.

- Peter Panning → C'è però da considerare che nell'usare il bias per le ombre, viene applicato un offset alla profondità degli oggetti. Di conseguenza, il bias potrebbe diventare sufficientemente grande da causare un offset visibile delle ombre rispetto alle posizioni effettive degli oggetti.

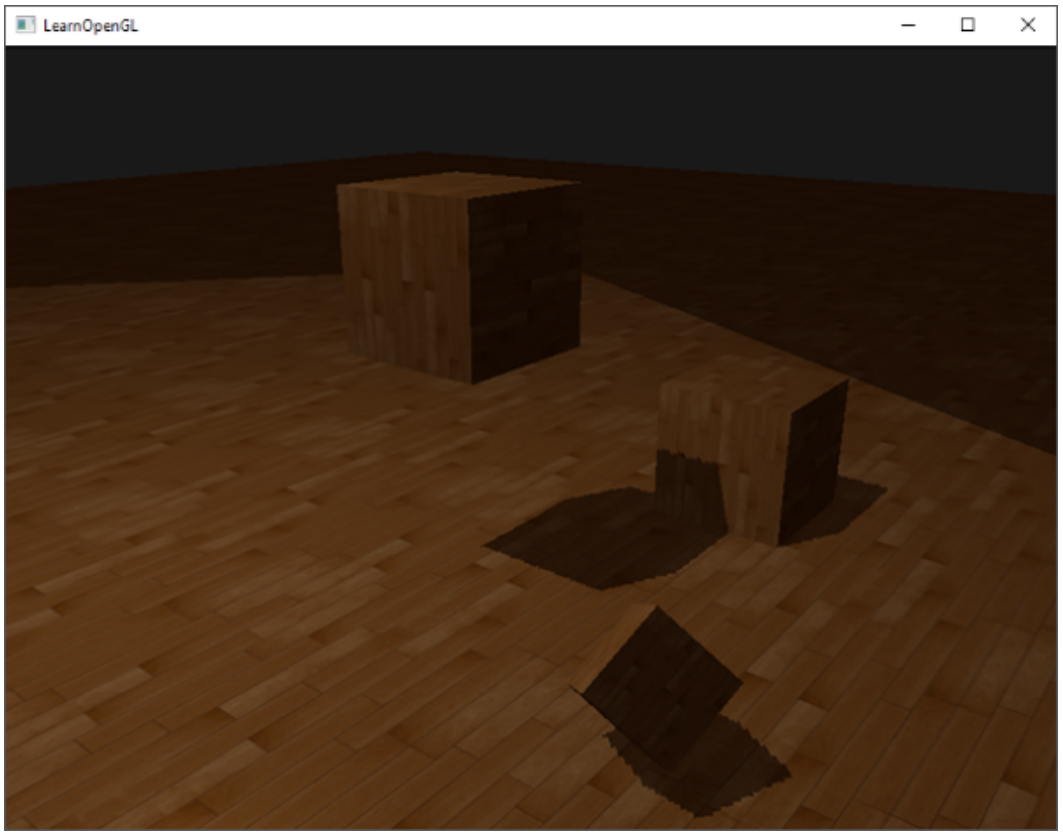


Questo problema è conosciuto come Peter Panning, dove essenzialmente gli oggetti sembrano leggermente staccati dalle loro ombre. Per ovviare a questa anomalia, è possibile utilizzare un piccolo trucco per risolvere la maggior parte dei problema dovuti al Peter Panning: il "front face culling" durante il rendering della mappa di profondità.

Poiché abbiamo bisogno solo dei depth value per la shadow map, non dovrebbe fare differenza per gli oggetti solidi se prendiamo la profondità delle loro facce frontali o delle facce posteriori. Utilizzare le profondità delle facce posteriori non fornisce risultati errati poiché non ha importanza se abbiamo ombre all'interno degli oggetti; comunque non possiamo vederle.



- Over sampling → Un'altra discrepanza visiva è che le regioni al di fuori del frustum visibile della luce vengono considerate in ombra, anche se di solito non lo sono. Questo accade perché le coordinate proiettate al di fuori del frustum della luce sono superiori a 1.0 e quindi campioneranno la texture di profondità al di fuori del suo intervallo predefinito [0,1]. In base al metodo di "wrapping" della texture, otterremo risultati di profondità incorretti che non si basano sui veri valori di profondità dalla sorgente di luce.



Nell'immagine si può vedere che c'è una sorta di regione immaginaria di luce, e una grande parte al di fuori di questa area che è in ombra; questa area rappresenta la dimensione della depth map proiettata sul pavimento.
Il motivo di questo effetto è che è stato impostata l'opzione di "wrapping" della depth map su GL_REPEAT.
Quello che bisognerebbe ottenere è che tutte le coordinate al di fuori dell'intervallo della mappa di profondità abbiano una profondità di 1.0, il che significa che queste coordinate

non saranno mai in ombra (poiché nessun oggetto avrà una profondità superiore a 1.0). Per arrivare a questo risultato sarà necessario configurare un colore di bordo per la texture e impostando le opzioni di wrapping della texture della depth map su GL_CLAMP_TO_BORDER e forzando la shadow value a 0.0 ogni volta che la coordinata z è più grande di 1.0.
Come risultato abbiamo che ci sono ombre solo dove le coordinate proiettate dei frammenti si trovano all'interno dell'intervallo della mappa di profondità, quindi tutto ciò che si trova al di fuori del frustum della luce non avrà ombre visibili.

- PCF → C'è però da risolvere un altro problema visivo, la "frastagliatura" delle ombre allo zoom:

Poiché la depth map ha una risoluzione fissa, la profondità spesso copre più di un frammento per texture element.
Di conseguenza, diversi frammenti campionano lo stesso valore di profondità dalla depth map e giungono ai stessi valori di conclusioni sull'ombra, il che produce questi bordi frastagliati e squadrati.
Una possibile soluzione per ovviare a questi bordi frastagliati è chiamata PCF, o "percentage-closer filtering", il quale offre diverse funzioni di filtraggio che producono ombre più morbide, rendendole meno frastagliate e omogenee.
L'idea è campionare più di una volta dalla depth map, ogni volta con coordinate di texture leggermente diverse. Per ciascun campionamento individuale verifica se è in ombra o meno. Tutti i risultati parziali vengono quindi combinati e mediati, ottenendo un'ombra dall'aspetto più piacevole