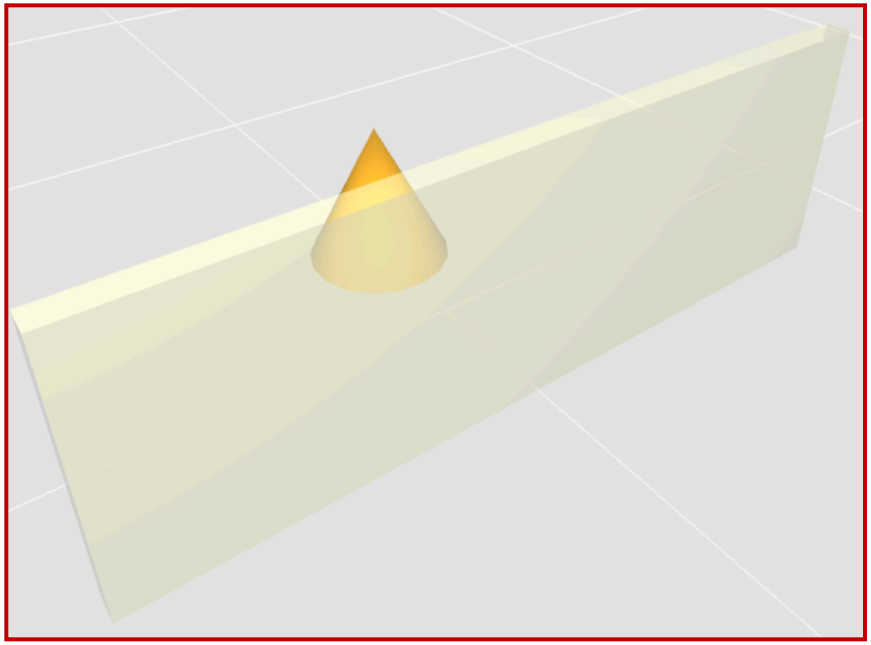
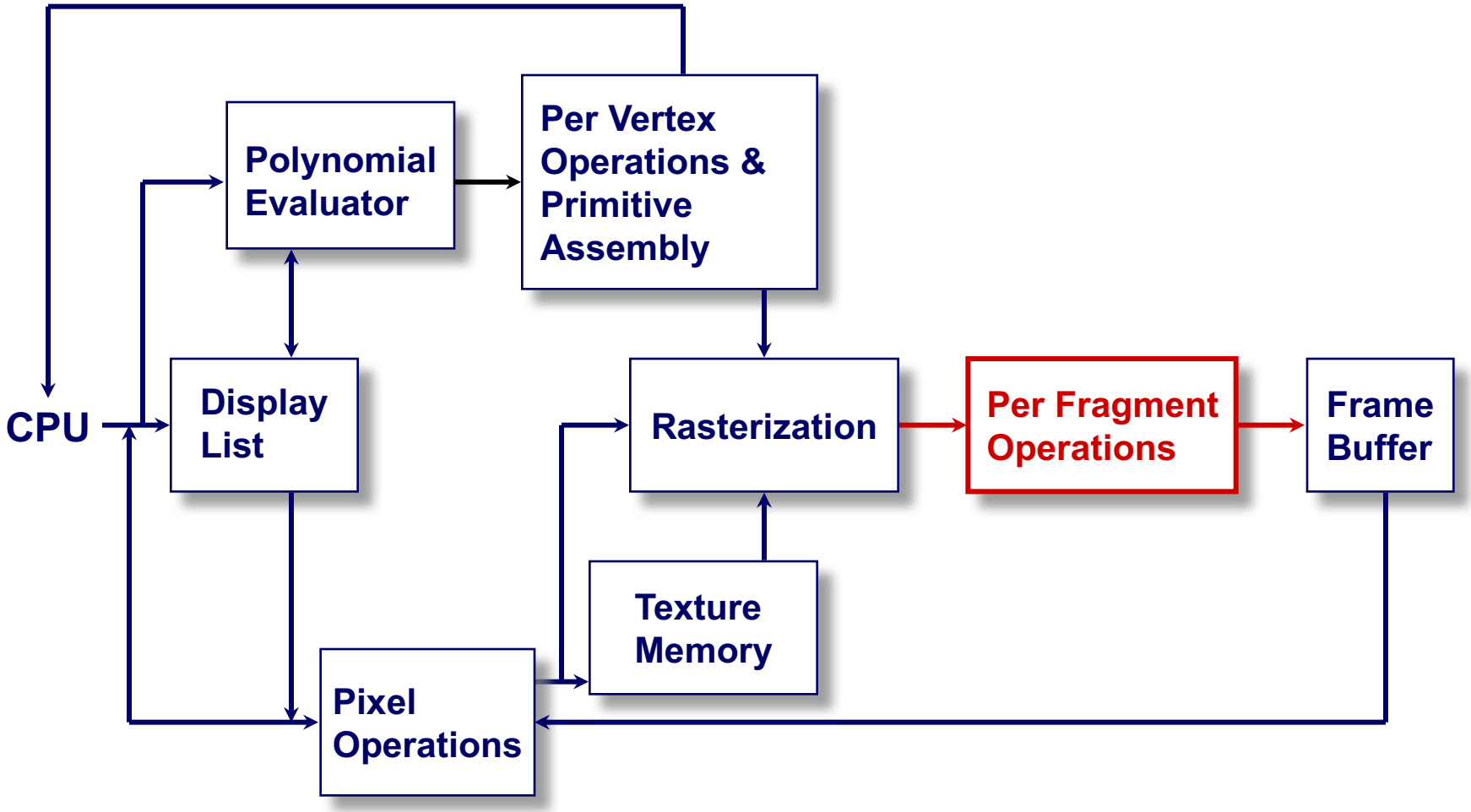
Rendering Avanzato in WebGL



WebGL Architecture

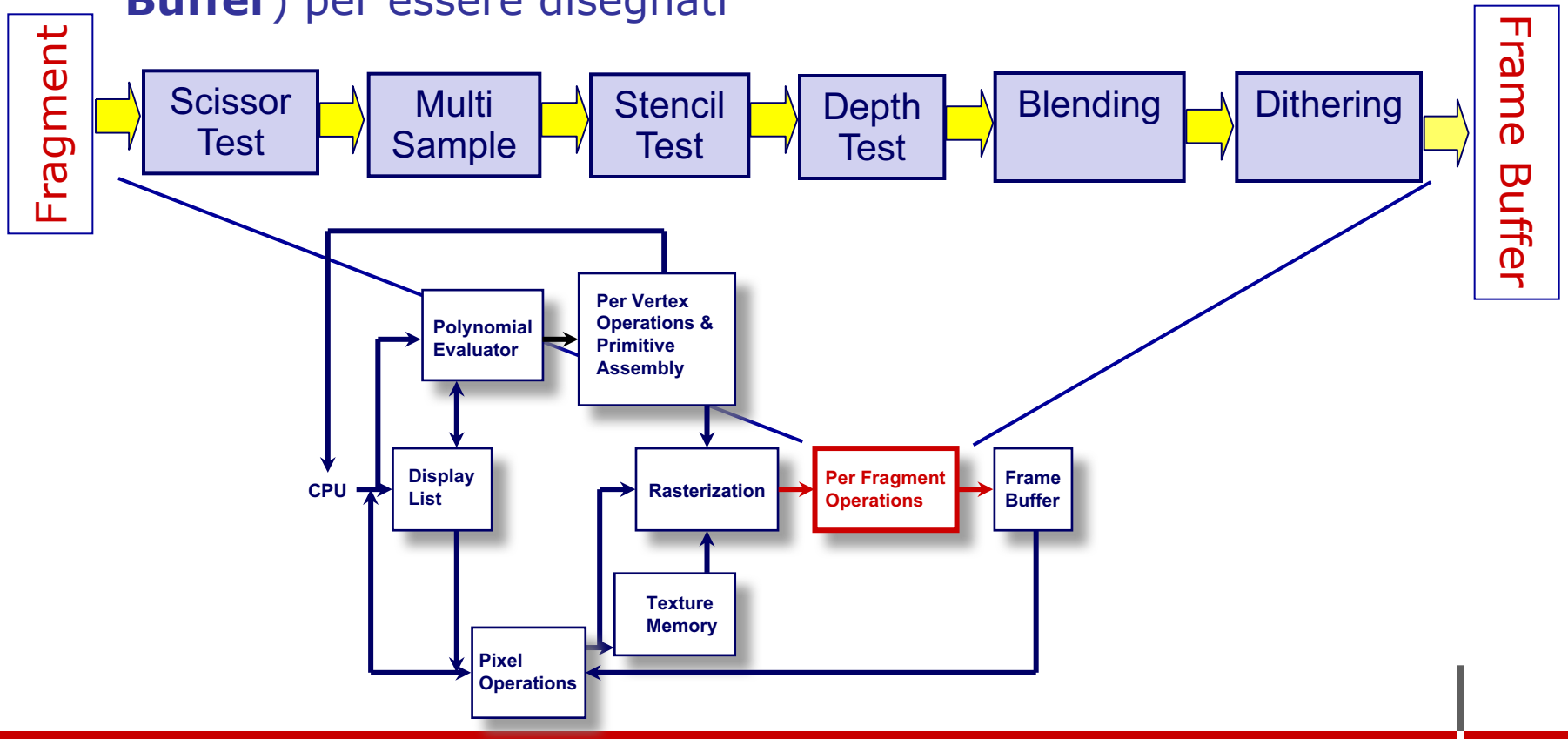


Conoscendo il funzionamento della pipeline grafica è possibile sfruttarne i meccanismi per implementare tecniche avanzate di rendering

Per-Fragment Operation

Fase significativa della pipeline per progettare tecniche di rendering

- Prevede l'utilizzo di **buffer**
- Prevede una **serie di test**: i fragment/pixel che sopravvivono a tutti i test vengono scritti nel buffer definitivo (**Color/Frame Buffer**) per essere disegnati

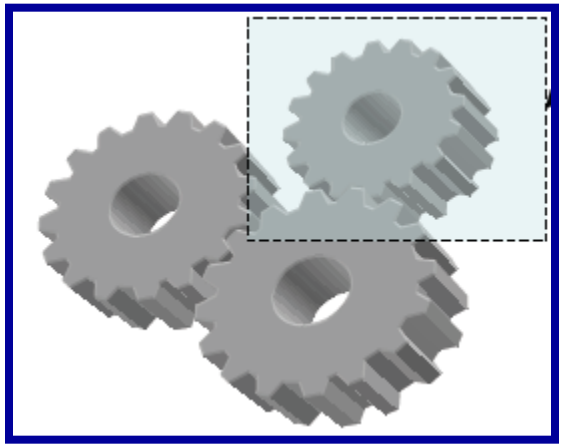


Frame Buffer

- Il frame buffer o Color Buffer memorizza i pixel che verranno accesi sullo schermo
- Un applicazione può usare uno o più Frame Buffer

Scissor Test

- Il test più semplice è lo "*scissor test*"
  - Permette di restringere il disegno ad una porzione rettangolare della viewport
  - Per abilitarlo:
    gl.enable(gl.SCISSOR_TEST);
  - Per definirlo:
    gl.scissor(*x*, *y*, *width*, *height*);
    i parametri sono come quelli della gl.Viewport



MultiSample Anti-Aliasing (MSAA)

Una tecnica molto semplice per fare Full Screen Anti-Aliasing (FSAA) è la SuperSampling Anti-Aliasing che consiste nel fare il rendering della scena ad una risoluzione maggiore rispetto a quella richiesta, quindi nel fare down-sampling alla risoluzione della viewport;

questa tecnica ha come inconveniente che è molto costosa; una sua semplificazione è nota come MultiSample Anti-Aliasing.

WebGL implementa il MSAA ed è attivo di default; per disattivarlo all'inizio si deve settare :

```
var context = canvas.getContext('webgl',{antialias:false});
```

WebGL Buffer

- I buffer WebGL memorizzano/conservano un certo numero di informazioni per ogni fragment/pixel

- Tipi di buffer:
  - Color/Frame buffer
  - Alpha buffer
  - Depth buffer
  - Stencil buffer

- Ogni buffer ha una ben precisa funzione; comunque si possono usare in un qualunque modo.
- Alla base delle tecniche di rendering avanzato c'è la manipolazione dei buffer.

WebGL ContextAttributes

Per essere usato, un buffer deve essere allocato

```
canvas.getContext(contextType, contextAttributes);
```

Il parametro WebGL ContextAttributes è opzionale. Questo parametro definisce alcuni stati dei buffer di WebGL;

Esempio:

```
var canvas = document.getElementById('canvas');
var context = canvas.getContext('webgl',{antialias:false, stencil:true});
```

ContextAttributes

- antialias. Se il suo valore è true, si ha un buffer di disegno che realizza anti-aliasing. Il default è true.

- alpha. Se il suo valore è true, fornisce un buffer alpha all'area di disegno. Il default è true.

- depth. Se il suo valore è true, si ha un buffer di disegno che contiene un depth buffer di almeno 16 bit. Il default è true.

- stencil. Se il suo valore è true, si ha un buffer di disegno che contiene uno stencil buffer di almeno 8 bit. Il default è false.

- premultipliedAlpha. Se il suo valore è true, si ha un buffer di disegno che contiene colori con valore alpha pre-moltiplicato. Il default è true.

- preserveDrawingBuffer. Se il suo valore è true, i buffer non verranno cancellati e conserveranno i loro valori fino a quando non saranno cancellati o sovrascritti dall'utente. Il default è false.

WebGL Test

- Collegati ai buffer ci sono i test:
  - Scissor test
  - Alpha test
  - Depth test
  - Stencil test
- Si tratta di un'espressione booleana valutata per ogni fragment
  - Se il test risulta True, il test passa e il fragment continua la pipeline;
  - Se il test fallisce, il fragment viene scartato;
  - Tutti i test vengono eseguiti nella fase di "fragment-processing" della pipeline.
- Per avere effetto, ogni test deve essere abilitato: gl.enable( )

Buffer & Test

- I buffer e i test sono associati

  Abbiamo già visto un esempio di questo nel depth buffer che implementa l'algoritmo Z-buffer
- Ricorda:
  - Allocate buffer
  - Enable test

Buffer	Corresponding Test
--	Scissor Test
Color/Frame Buffers	Alpha Test
Depth Buffer	Depth Test
Stencil Buffer	Stencil Test

Clearing

- I Buffer vengono inizializzati (clear) con gl.clear

```
gl.clearColor( );
gl.clearDepth( );
gl.clearStencil( );
```

Esempio: Depth Buffering

- Alloca il depth buffer (è allocato di default)

- Abilita il depth test

```
gl.enable( gl.DEPTH_TEST );
```

- Inizializza (clear) il color e il depth buffer

```
gl.clear( [1,1,1,1]);
gl.clearDepth(1.0)
```

- Si rende la scena, quindi …

Esempio: Depth Buffering

gl.depthFunc(func);

func può essere uno dei seguenti parametri:

- gl.NEVER
- gl.LESS default
- gl.LEQUAL
- gl.EQUAL
- gl.NOTEQUAL
- gl.GEQUAL
- gL.GREATER
- gl.ALWAYS

RGB-Alpha e Blending

A (di RGBA) sta per Alpha ed è una quarta componente colore ("buffer" Alpha)

Misura l'opacità del pixel a cui è associato:

valori da 0 (trasparente) a 1 (completamente opaco)

– Simula gli oggetti traslucidi: vetro, acqua,…

– Composizione (sovrapposizione) di immagini

– Antialiasing di primitive geometriche

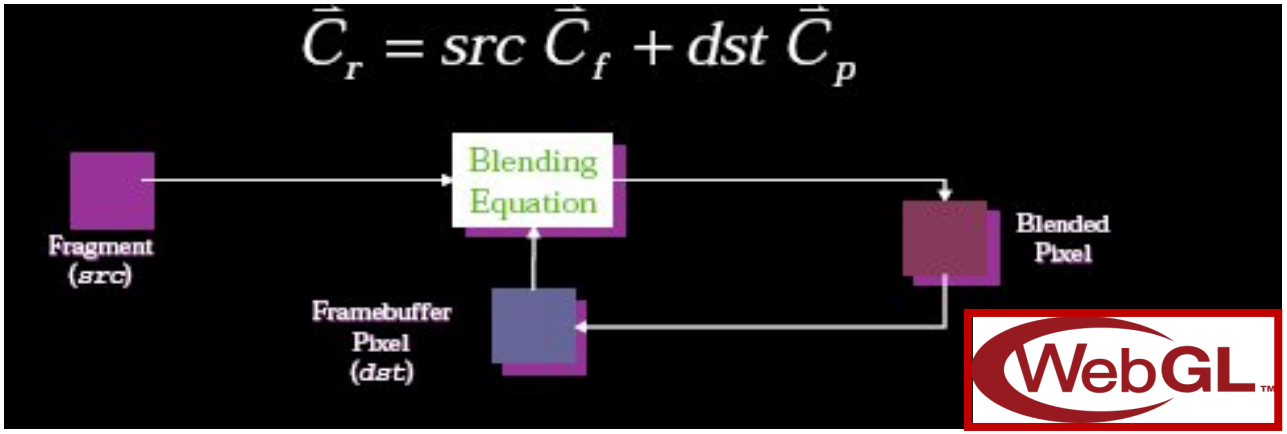
il "buffer" Alpha è allocato di default

ma A è ignorata se non è abilitato il blending

gl.enable(gl.BLEND)

RGB-Alpha e Blending

Combina i fragment con il valore dei pixel che sono già nel frame buffer secondo la seguente formula:



gl.blendFunc(src, dst)

src e dst possono essere: gl.ZERO, gl.ONE, gl.SRC_COLOR, gl.DST_COLOR, gl.SRC_ALPHA,gl.DST_ALPHA, gl.CONSTANT_COLOR,gl.CONSTANT_ALPHA, gl.ONE_MINUS_SRC_ALPHA, gl.ONE_MINUS_DST_ALPHA, gl.ONE_MINUS_SRC_COLOR, gl.ONE_MINUS_DST_COLOR, gl.ONE_MINUS_CONSTANT_COLOR, gl.MINUS_CONSTANT_ALPHA, gl.ALPHA_SATURATE

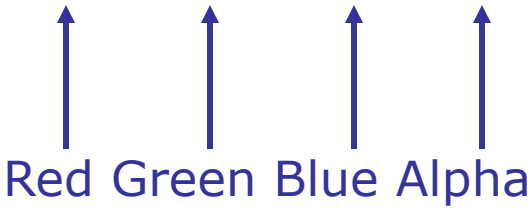
Esempio

codice transparency.html

Masking

- Le maschere (Mask) determinano se un buffer (o parte di un buffer) deve essere scritto
- Per esempio,

```
gl.colorMask(false, true, true, true);
```

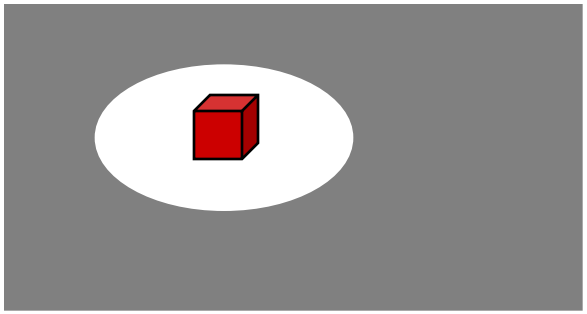


Significa che la componente Red del Color buffer non dovrà essere cambiata, le altre sì

- Un Mask influisce su tutti i comandi che dovrebbero cambiare il buffer, anche gl.clear

Stencil Buffer

- Viene usato per controllare il disegno basato su valori
  - I fragment che falliscono lo stencil test non sono disegnati
  - Esempio: si crea un Mask nello stencil buffer e si disegnano solo gli oggetti non nella Mask area



Controllare lo Stencil Buffer

gl.stencilFunc(*func, ref, mask*);

- Confronta i valori nel buffer con ref usando func
- Si applica solo per i pixel nel mask che sono 1
- func è una delle funzioni standard di confronto (gl.EVER, gl.ALWAYS, gl.LESS, gl.LEQUAL, gl.[NOT]EQUAL, gl.GREATER, gl.GEQUAL)

gl.stencilOp(*fail, zfail, zpass*);

- Permette modifiche nello stencil buffer basate sul passaggio o meno dello stencil test e depth test: gl.KEEP, gl.INCR, ecc.

Creare un Mask

```
gl.enable( gl.STENCIL_TEST );
gl.clearStencil( 0x0 );

gl.stencilFunc( gl.ALWAYS, 0x1, 0x1 );
gl.stencilOp( gl.REPLACE, gl.REPLACE, gl.REPLACE );
```

Disegna il mask

Usare lo Stencil Mask

1. Costruisce uno stencil buffer solo in lettura (read-only)

   gl.stencilOp(gl.KEEP, gl.KEEP, gl.KEEP);

2. Disegna oggetti dove lo stencil vale 1

   gl.stencilFunc(*gl.EQUAL, 0x1, 0x1*);

3. Disegna oggetti dove lo stencil ha valori diversi da 1

   gl.stencilFunc(gl.NOT_EQUAL, 0x1, 0x1);

Riflessioni planari

Il dinosauro viene riflesso dal pavimento.
Si disegna dino due volte, la seconda volta lo si "scala" di (1,-1,1) per generare il simmetrico rispetto al pavimento (il riflesso attraverso il pavimento)

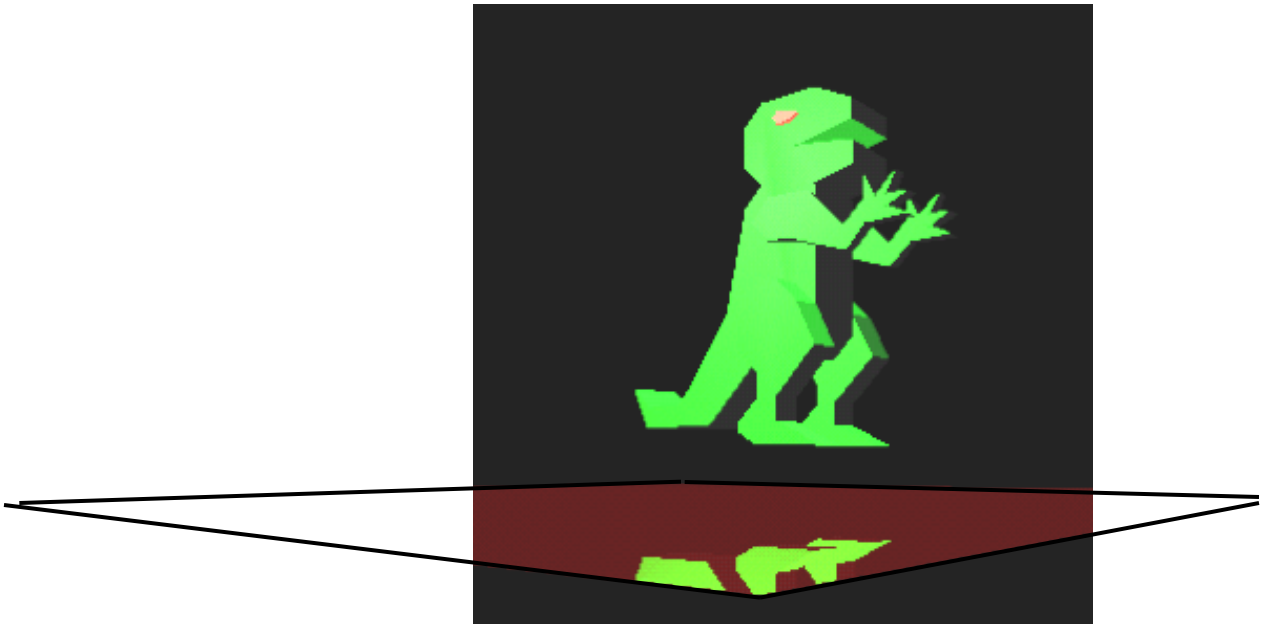
Confronta le due versioni

Bene

Male

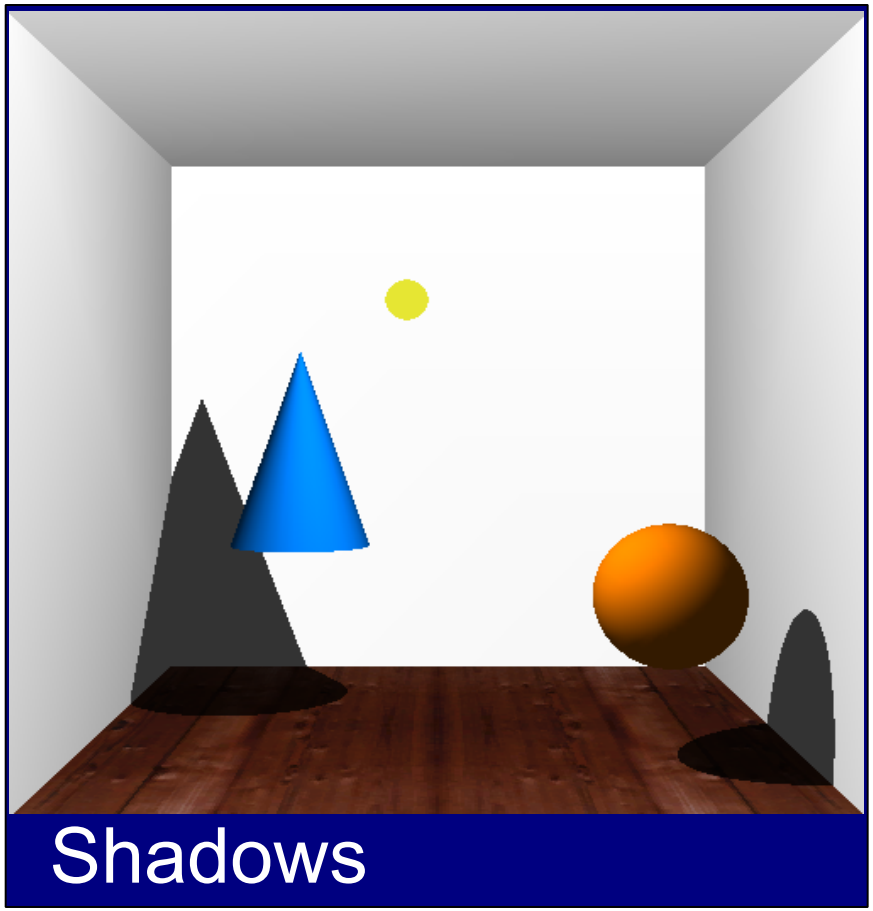
Si noti che l'immagine riflessa di destra esce dal pavimento!

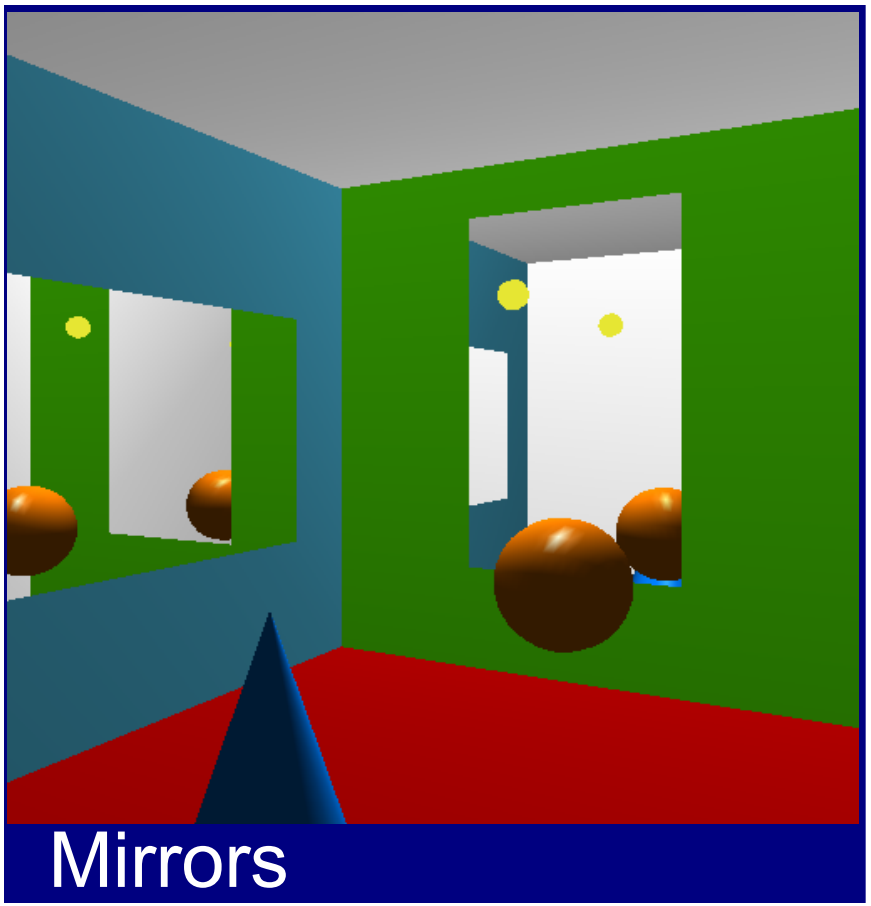
Lo Stencil mantiene il pavimento



- Inizializza lo stencil a zero.
- Disegna il poligono del pavimento nello stencil con valore 1
- Disegna il secondo Dino (la riflessione) solo dove lo stencil è 1

Stencil Buffer: esempi





Applications

Real World Applications using WebGL

https://manu.ninja/25-real-world-applications-using-webgl/

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Giulio Casciola

Dip. di Matematica

giulio.casciola at unibo.it