GPU Programming

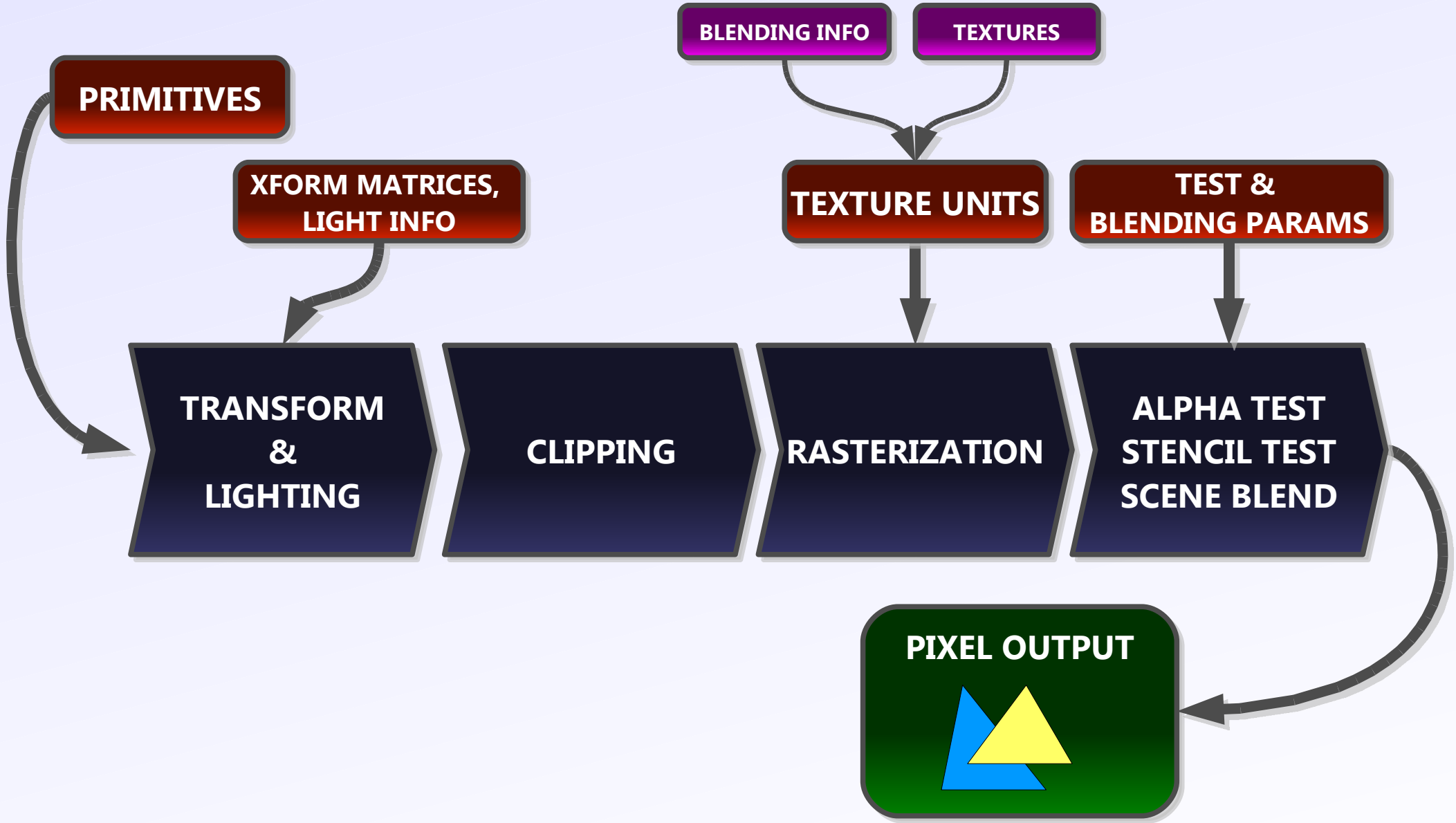
Alessandro Febretti

Politecnico di Milano – Argomenti Avanzati di Sintesi di Immagini A.A. 06/07

Sommario

- Fixed Function Pipelines
- Programmable Pipelines
- Struttura degli Shader
- Esempi
  - Per-vertex lighting & texturing.
  - Per-pixel lighting.
  - Esempi avanzati.

Fixed-Function Pipelines

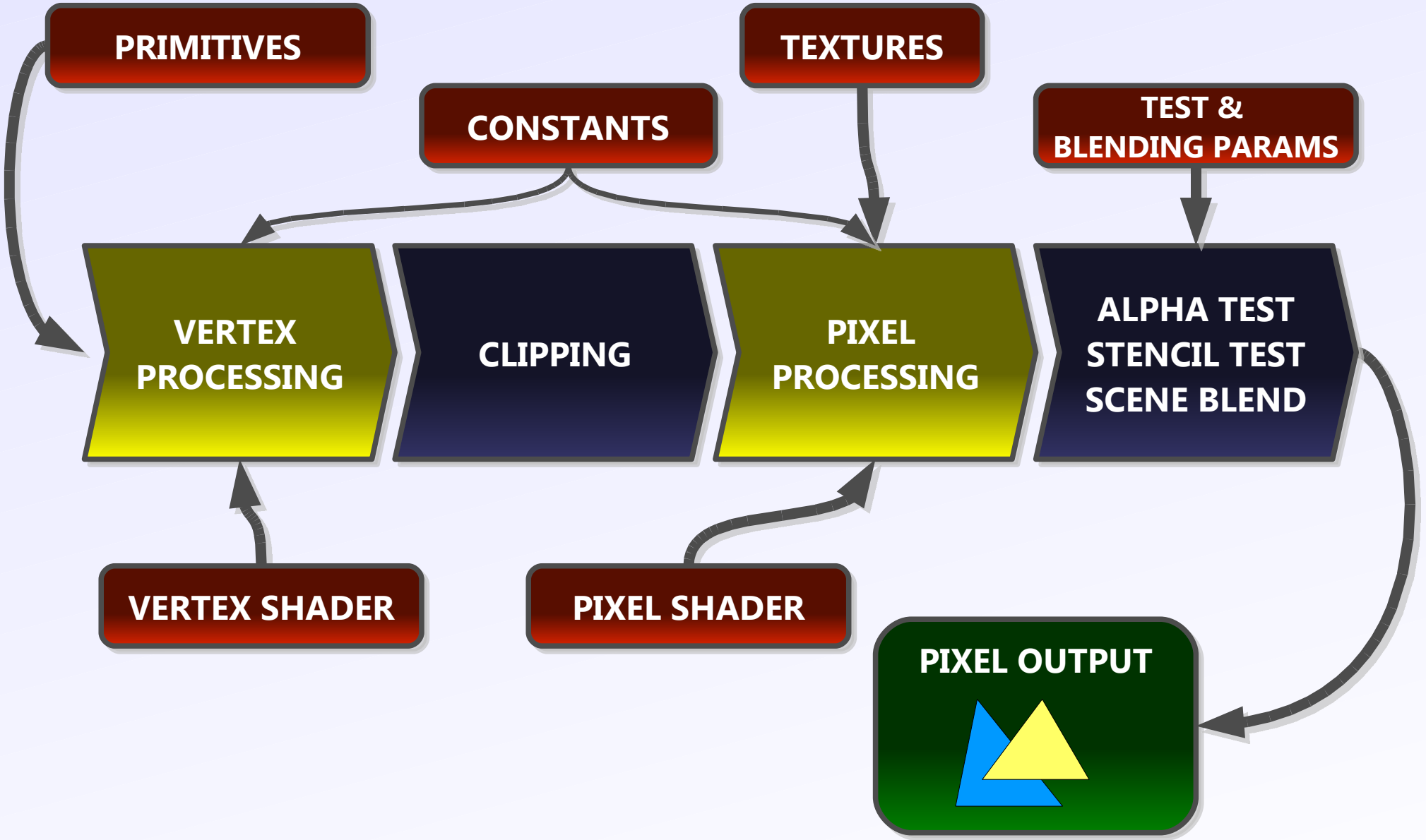


Fixed Function Pipeline

- Presente nella prima generazione di schede grafiche.

- Parametrizzata, ma non completamente flessibile
  - Per-pixel lighting impossibile.
  - Per superare limitazioni ogni vendor implementava feature addizionali differenti, difficili da sfruttare.
  - Content producer conoscono già il concetto di **shader program**

Programmable Pipelines



Programmable Pipeline

- Flessibilità:
  - I due stage più importanti della pipeline sono completamente programmabili (più o meno..)
- Compatibilità:
  - Vendor non devono aggiungere funzionalità nuove ad HW delle schede, ma solo supportare **Shader Models**

Vertex Shaders (aka. Vertex Programs)

- **INPUT**
  - Posizione vertice (model coords.)
  - Info aggiuntive vertice (normale, colore, UV coords, ...)
  - Matrici di trasformazione (di solito una...)
  - Costanti (ie posizione & colore luci)
- **OUTPUT**
  - Posizione vertice (screen coords.)
  - Dati aggiuntivi (ie colore, UV coords)

Pixel Shaders (aka. Fragment Programs)

- **INPUT**
  - Dati emessi da vertex shader **interpolati** (colore, UV coords...)
  - Textures.
  - Costanti.
- **OUTPUT**
  - Colore pixel.
  - Profondità pixel (Z value).

Shading Languages

- Prime versioni: ASM-like.
- Oggi: High Level Shading Languages.
  - Sintassi simile al C.

```
output_type nome_funz(parametri)
{
   [codice...]
}
```

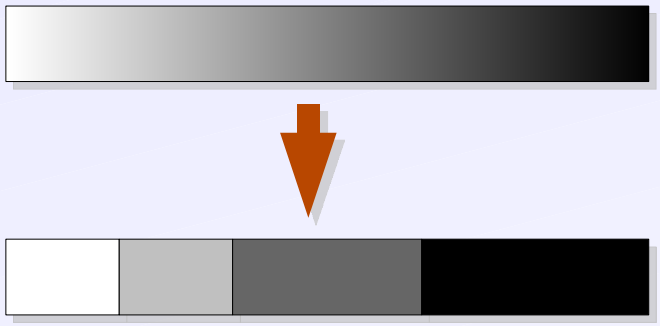
Shaders

- Il programma di uno shader può contenere:
  - Istruzioni artimetiche (anche vettoriali).
    - `float A = B * C;`
    - `float3 V2 = mul(V1, M1);`
    - `float A = dot(V1, V2);`
  - Istruzioni di controllo di flusso (limitate)
  - Funzioni di accesso alle texture
    - `float4 color = tex2D(texture, coords);`
  - Funzioni di utilità
    - `lerp, min, max, lit, ...`

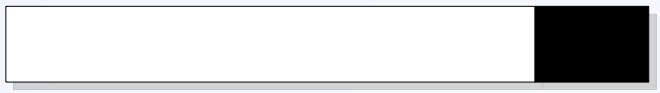
Esempi

Tecniche avanzate: Cel Shading

- Scopo: rendering cartoon-like.
  - Discretizzo intensità dei colori, attraverso lookup texture.



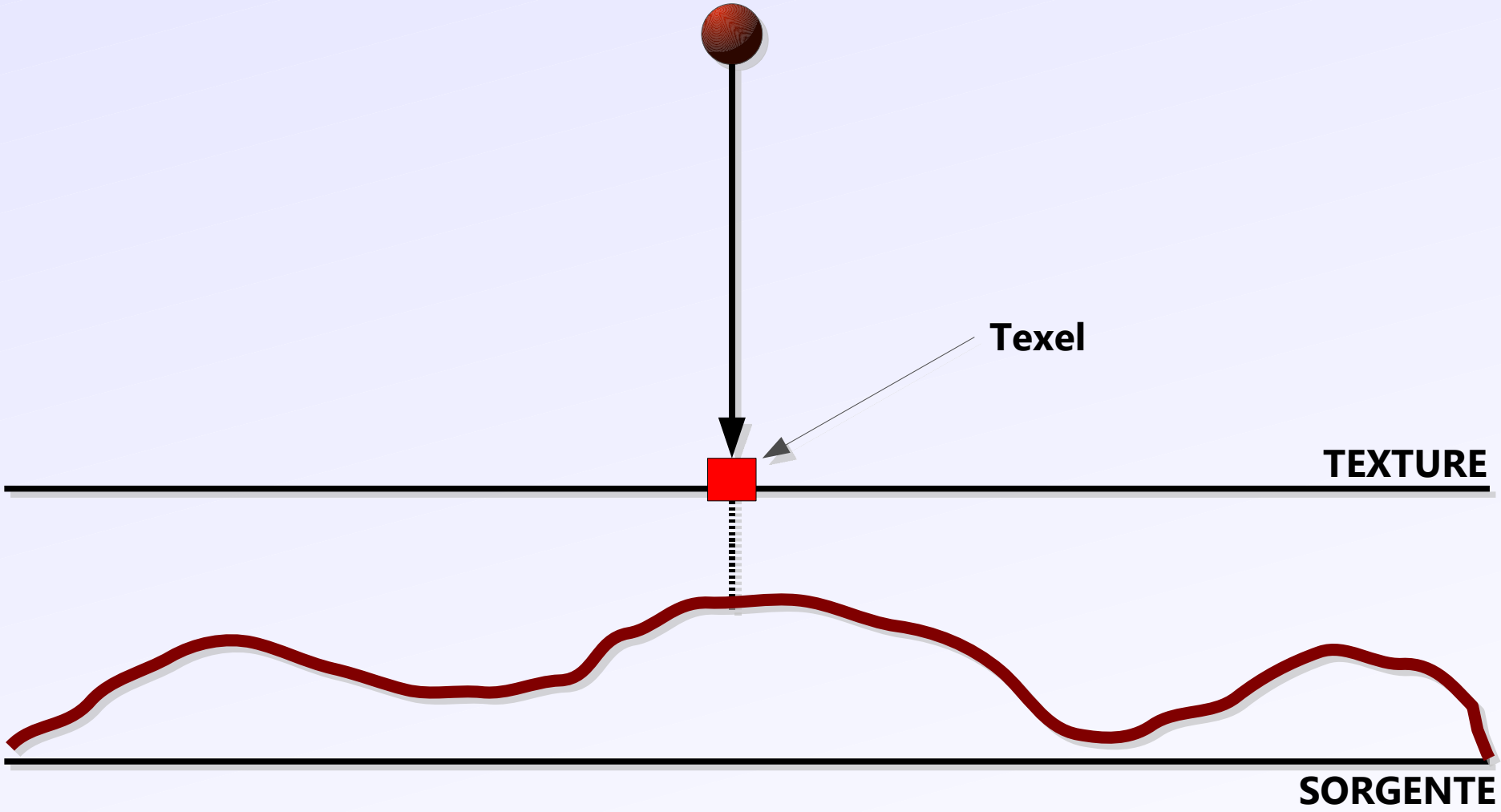
  - Per le outline: discretizzo normale rispetto a eye position.



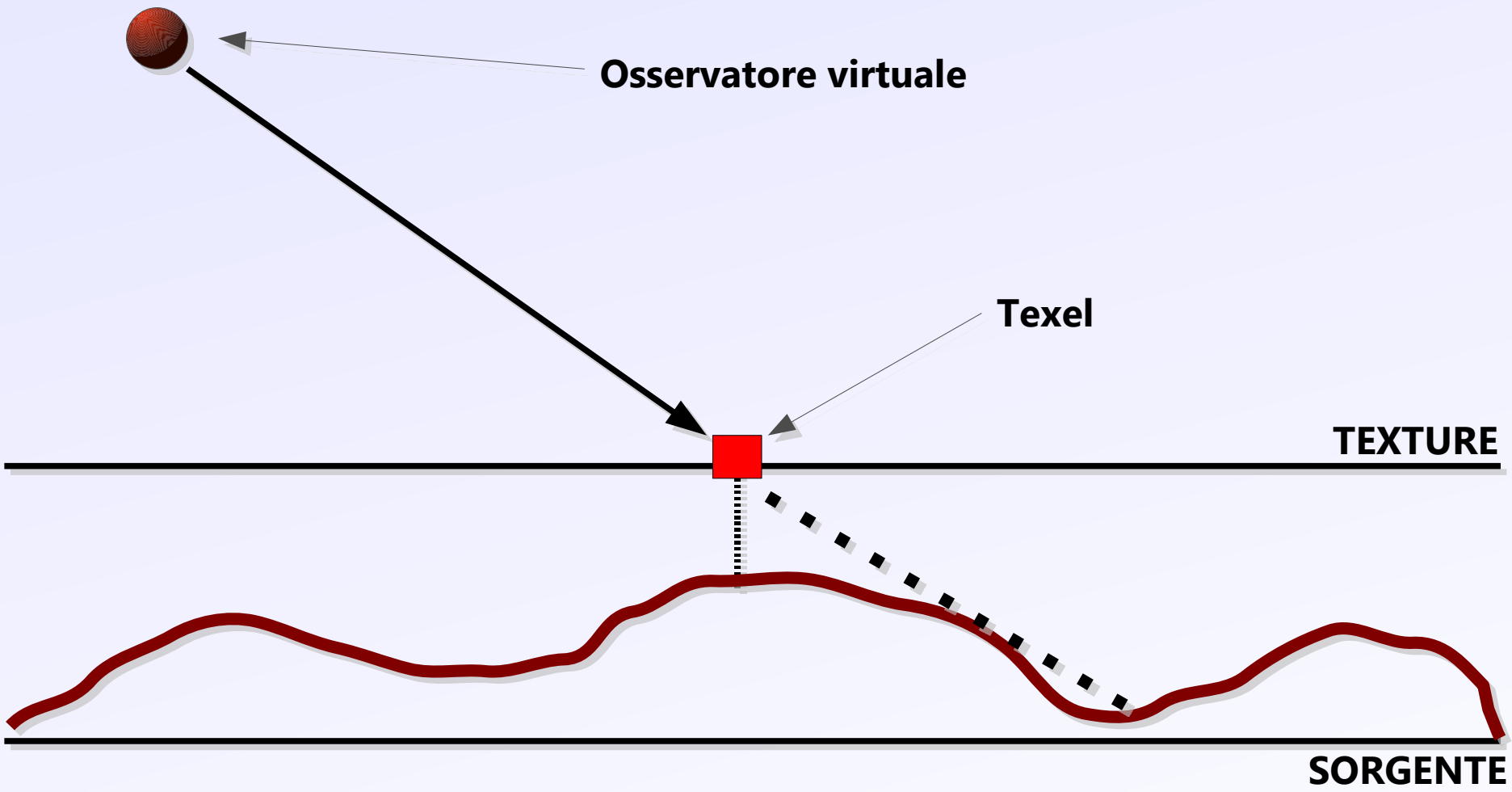
Tecniche avanzate: Offset Mapping

- Aumento effetto fotorealistico per texture non planari.
- Basato sulla correzione delle coordinate di texturing

Offset Mapping



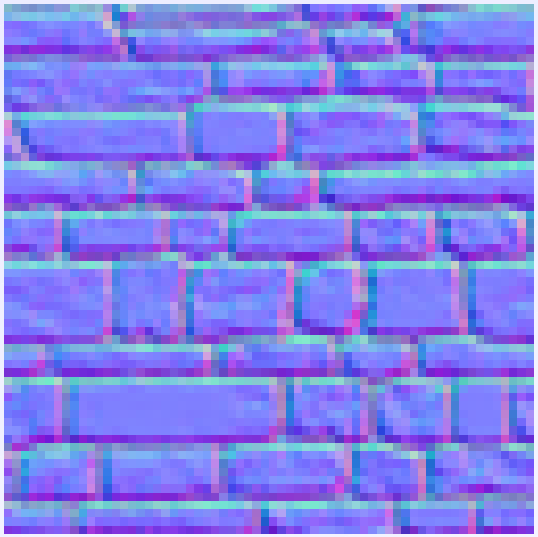
Offset Mapping

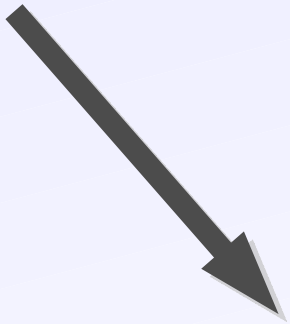


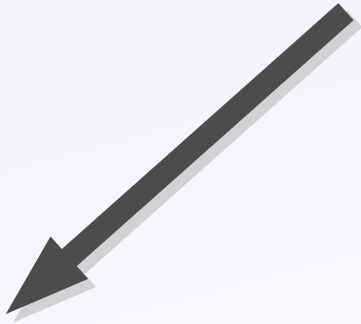
Offset + Normal mapping

RGB: Normal map,
correggo illuminazione.

Alpha: Height map,
correggo texture coords.

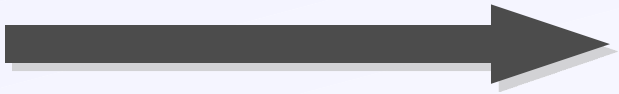


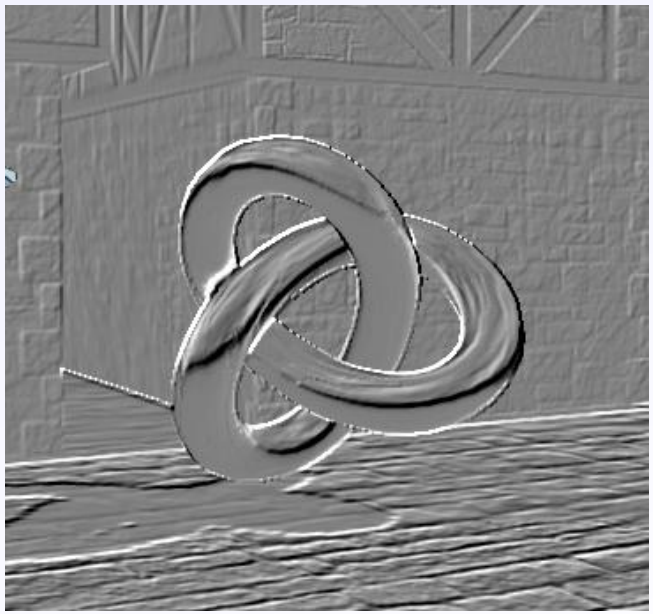




Tecniche avanzate: Postprocessing

- **Idea: utilizzo pixel shader come filtri di postprocessing in real-time.**
  - Fase1: Rendering della scena su texture temporanea.
  - Fase2: Rendering da texture a frame buffer utilizzando pixel shader per postprocessing





Conclusioni

- Pipeline programmabili hanno notevoli potenzialità
  - Non solo grafica!
- Futuro: shader model 4.0
  - Geometry shaders.
  - Supporto completo a istruzioni di controllo di flusso.
  - GPU general purpose.

That's all folks!

:: DOMANDE?