Transfer Textures for Fast Precomputed Radiance Transfer

Sirikonda Dhawal, Aakash KT, P.J. Narayanan

CVIT, KCIS, IIIT-Hyderabad, Telangana, India

Abstract

Precomputed Radiance Transfer (PRT) can achieve high-quality renders of glossy materials at real-time framerates. PRT involves precomputing a k-dimensional transfer vector or a $(k \times k)$-matrix of Spherical Harmonic (SH) coefficients at specific points for a scene depending on whether the material is diffuse or glossy respectively. Most prior art precomputes values at vertices of the mesh and interpolates color for interior points. They require finer mesh tessellations for high-quality renders. In this work, we introduce transfer textures for decoupling mesh resolution from transfer storage and sampling specifically benefiting the glossy renders. Dense sampling of the transfer is possible on the fragment shader while rendering with the use of transfer textures for both diffuse as well as glossy materials, even with a low tessellation. This simultaneously provides high render quality and frame rates.

RELATED WORK

Precomputed Radiance Transfer (PRT) offloads expensive computations of ray tracing to a pre-computation step, after which the stored data can be utilized for real-time photorealistic rendering. The core PRT framework as proposed by [SKS02] precomputes the transfer function and stores it in SH basis at the vertices of the scene. Prior works like [McK10] and PRT of D3D9 leverage the continuous texture space to store transfer but are limited to diffuse reflection, due to their choice of the formulation [SKS02], and extending their work directly will lead to heavy texture storage.

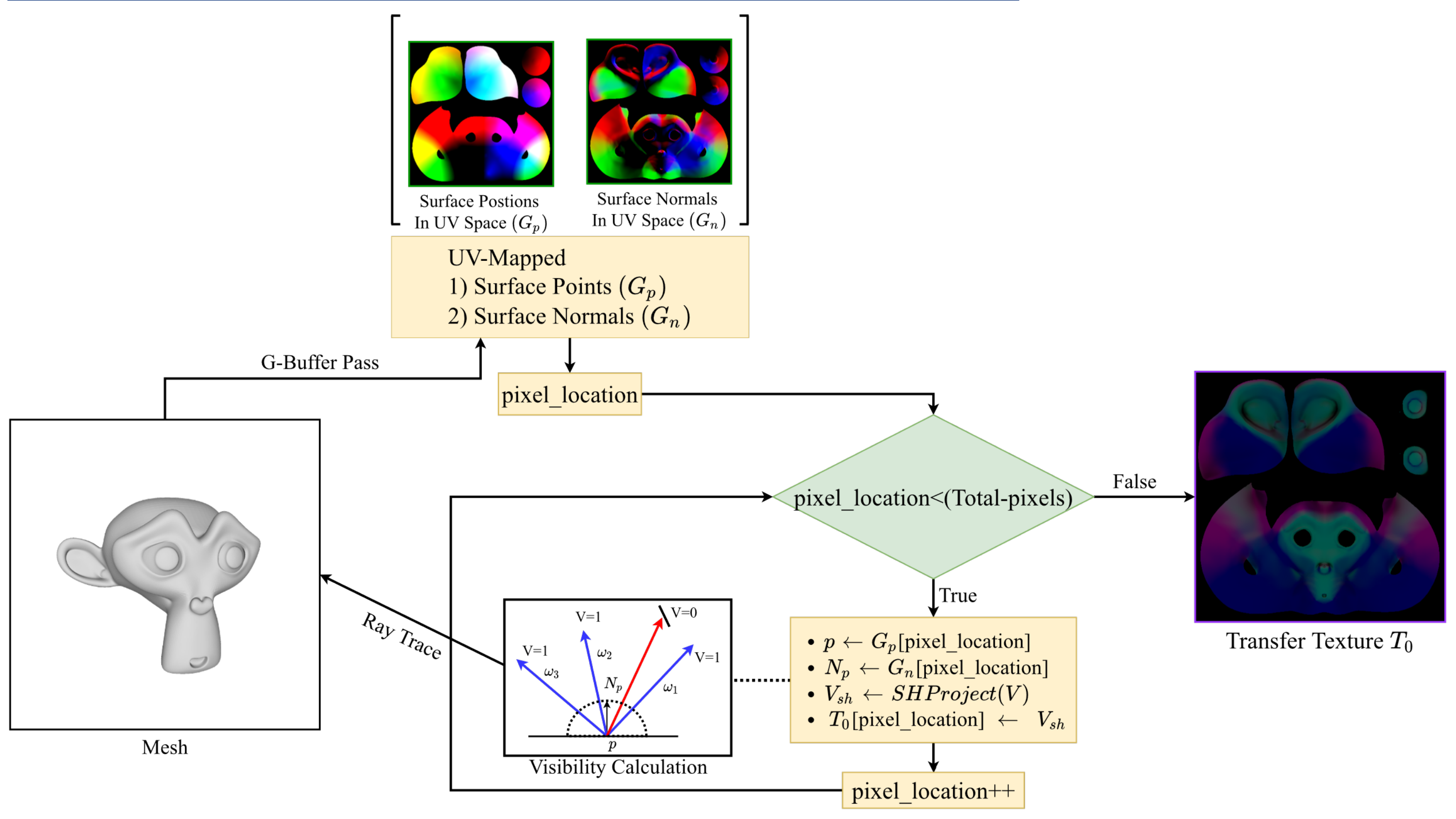
OVERVIEW

Usage of vertex-based methods to obtain accurate renderings necessitates dense tessellation, despite having minimal to no change in geometry, eg. plain walls or tabletops. To address this issue we leverage the texture space and propose transfer textures. We augment transfer textures with Triple Product Formulation(TPF) [NRH04] for the evaluation of color at run-time in the fragment shader.

The choice of TPF helps reduce the memory requirement, as a k-dimensional transfer vector is sufficient to produce glossy renders. While the original formulation of Sloan et al. requires a $(k \times k)$-transfer matrix. The storage of matrix for each texel location makes it in-feasible in real-time scenarios. Refer to the document for more details.

Method

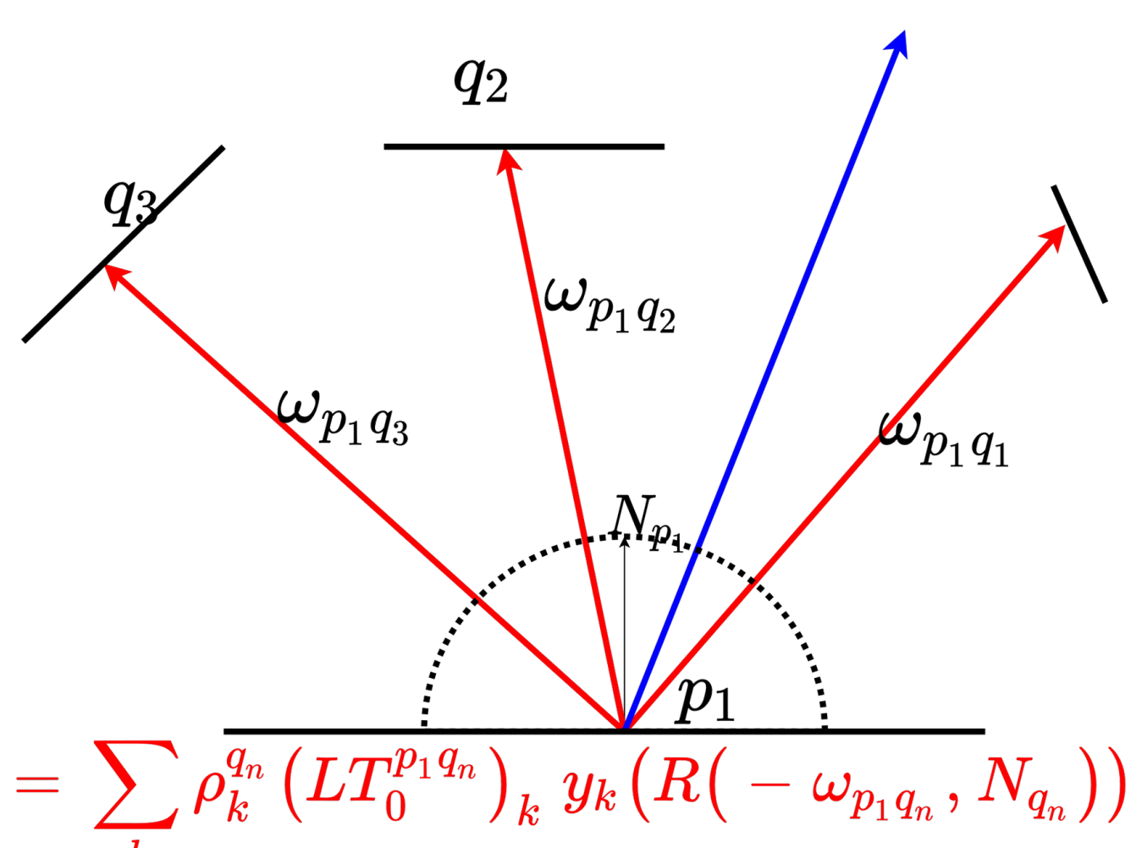
Computing Transfer Textures


Surface Postions In UV Space (G_p)
Surface Normals In UV Space (G_n)
UV-Mapped
1) Surface Points (G_p)
2) Surface Normals (G_n)
G-Buffer Pass
pixel_location
pixel_location<(Total-pixels)
False
True
Transfer Texture T_0
Mesh
Ray Trace
V=1
V=0
ω_1
ω_2
ω_3
N_p
p
Visibility Calculation
• $p \leftarrow G_p$[pixel_location]
• $N_p \leftarrow G_n$[pixel_location]
• $V_{sh} \leftarrow SHProject(V)$
• T_0[pixel_location] $\leftarrow V_{sh}$
pixel_location++

Given a mesh, we obtained UV-mapped surface positions and surface normal in texture space via G-buffer passes. For every pixel location, we obtain a surface position *(p)* and surface normal *(N_p)* we ray-trace to obtain visibility of *(p)* and project it to SH-basis. We do this to every pixel location to obtain the Transfer-Texture *(T_0)*.

We use this Transfer-Texture in the fragment shader, coupling with Triple-Product formulation to obtain final renders.

Interreflections



$$B^{p_1 q_n} = \sum_k \rho_k^{q_n} \left(LT_0^{p_1 q_n}\right)_k y_k\left(R\left(-\omega_{p_1 q_n}, N_{q_n}\right)\right)$$

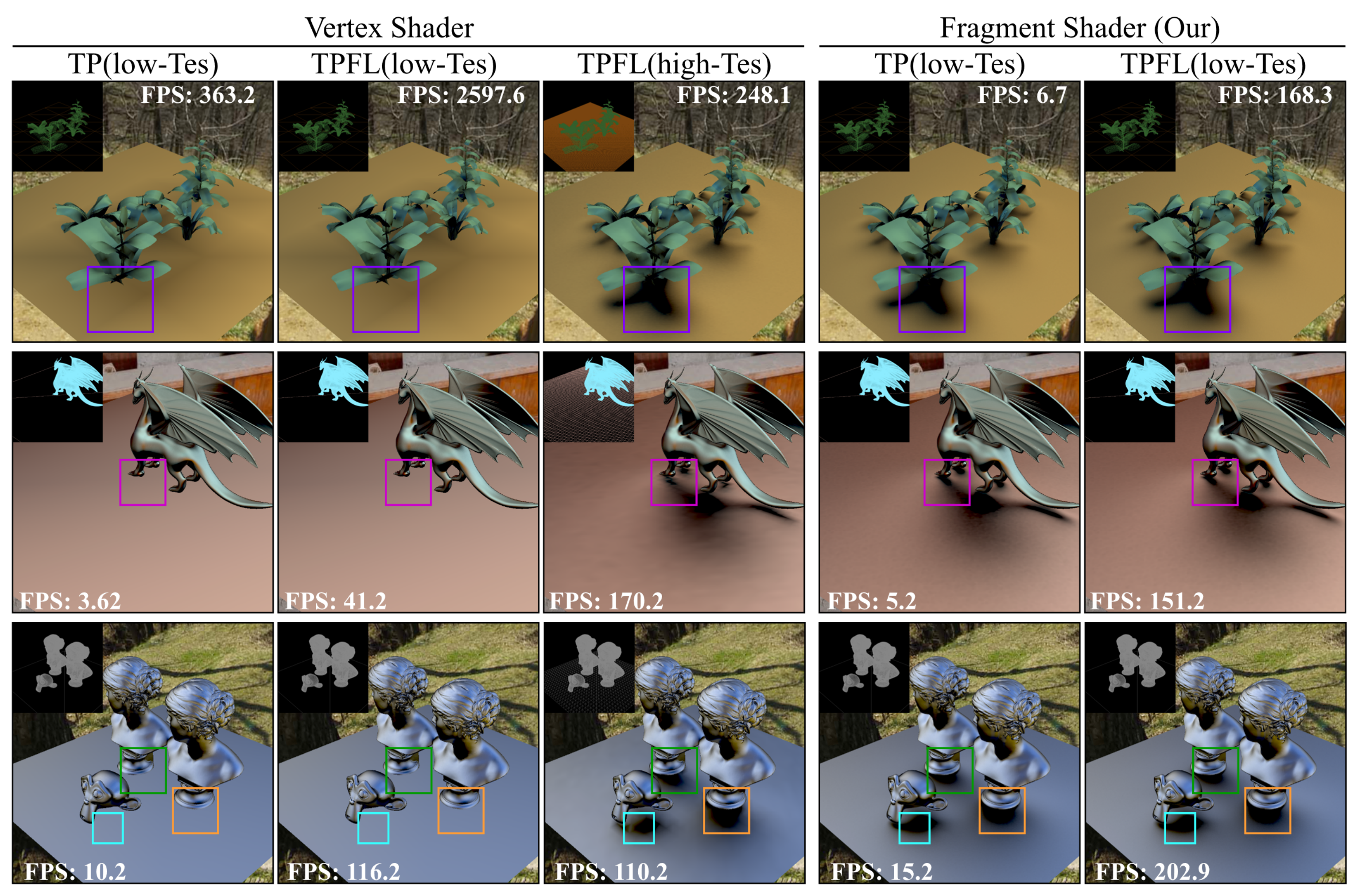
We compute interreflection by finding hit-points about a given surface position *(p_1)* obtained from the UV-mapped Surface points *(G_p)*. For each hit-points *(q_i)* we calculate radiance contribution at *(p_1)* *[we fetch (T_{qi}) from previously calculated Transfer-texture(T_0)]*. This is repeated for all such hit-points *(q_i)* to obtain an indirect radiance map.

We obtain an Interreflection Texture *(T_1)*, which is used at run-time to incorporate the *inter-reflections.*

RESULTS

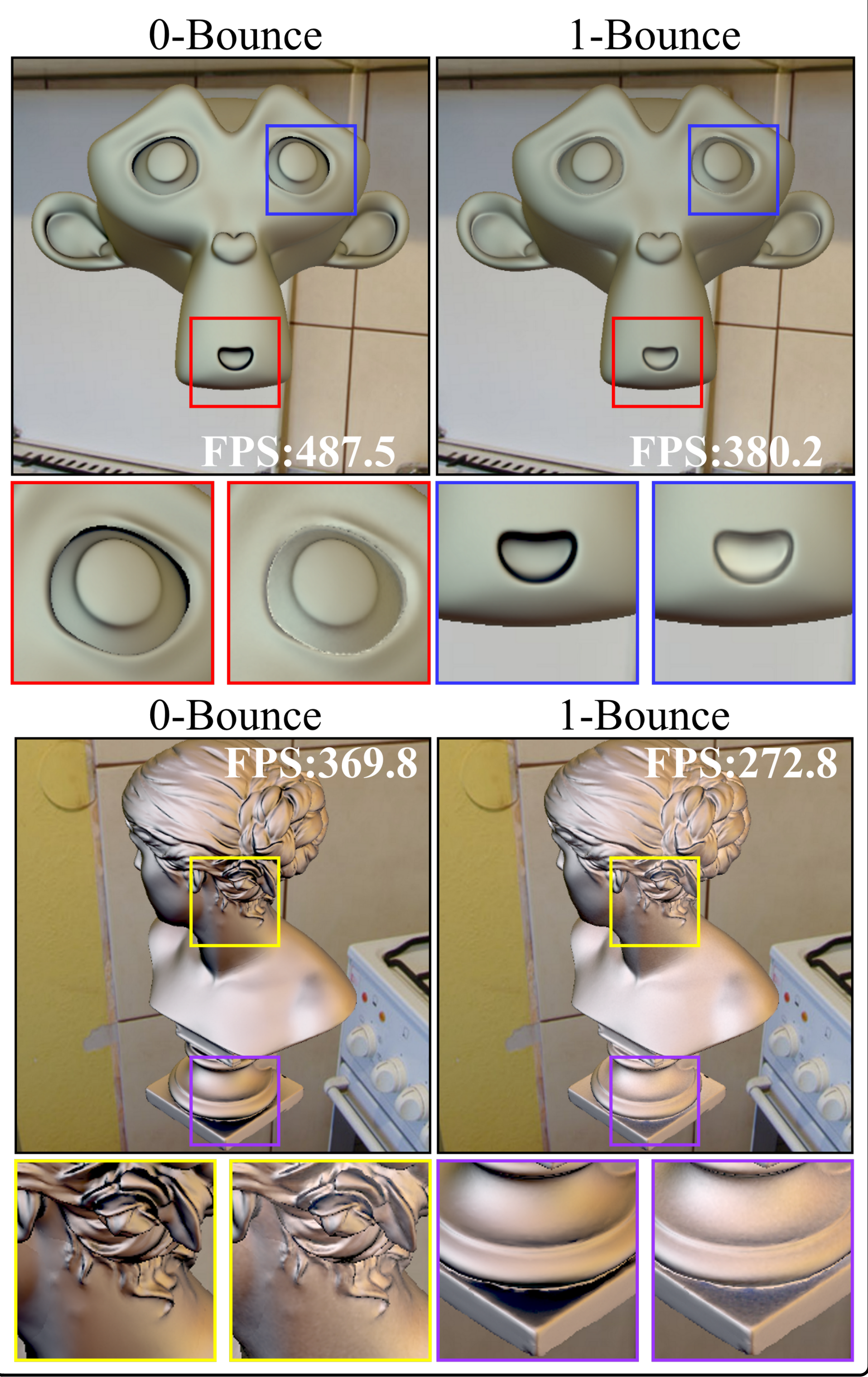
Glossy Results

We show results using TP(Triple Product) and TPFL(Triple Product Fixed Light) for various scenes. Renderings obtained by *vertex-shader-based* methods(*left*) with low tessellation, fail to produce the shadows on the ground plane. The *fragment-shader-based* rendering(*right*) with the same tessellation using our transfer textures, produce accurate renderings. The *vertex-shader-based* methods approximately approach our rendering quality with the addition of redundant vertices (*middle*-highTes). The degree of tessellation used is shown in the top-left inset of each scene along with the corresponding FPS.



Inter-reflection Results

We show *0-bounce*(left) and *1-bounce*(right), Insets show the effect of inter-reflections.



REFERENCES

[McK10] MCKENZIE CHAPTER, HARRISON LEE. "Textured Hierarchical Precomputed Radiance Transfer". (2010)

[NRH04] NG, REN, RAMAMOORTHI, RAVI, and HANRAHAN, PAT. "Triple Product Wavelet Integrals for All-Frequency Relighting". ACM Trans. Graph. (Aug. 2004).

[SKS02] SLOAN, PETER-PIKE, KAUTZ, JAN, and SNYDER, JOHN. "Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments". ACM Trans. Graph. 21.3 (July 2002).

IIIT HYDERABAD

EG'22 TheEvent