ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ

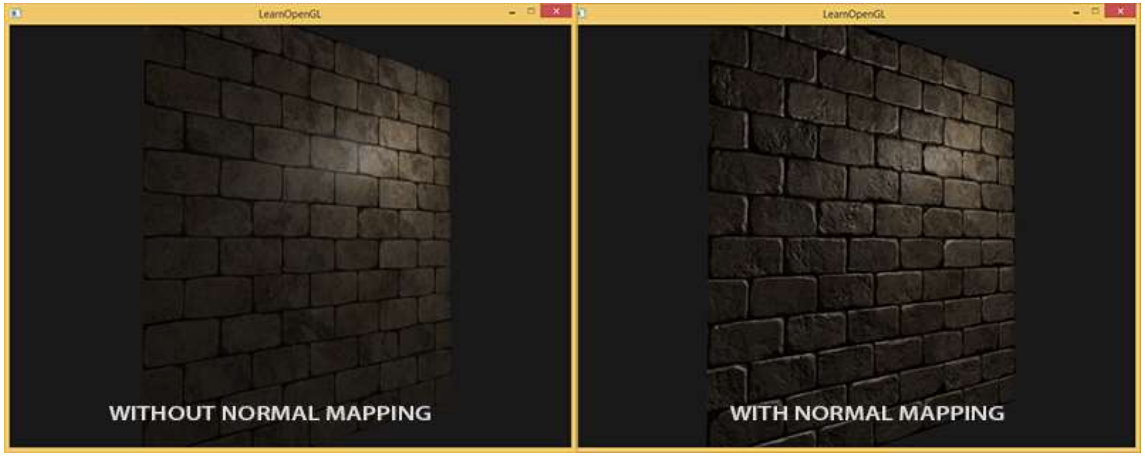
**ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**

# Εισαγωγή στα Γραφικά Υπολογιστών

Εργαστήριο 7

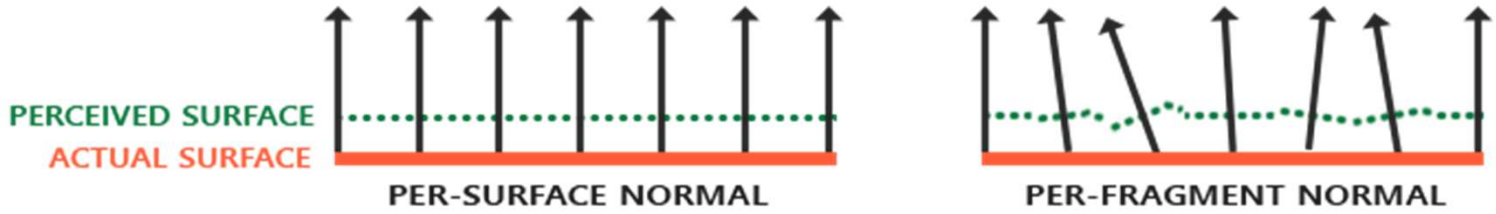
# Χάρτες κανονικών διανυσμάτων

- 3Δ αντικείμενα → πολλά επίπεδα τρίγωνα
- Οι χάρτες φωτισμού προσδίδουν χρωματική λεπτομέρεια, οι επιφάνειες παραμένουν επίπεδες
- Χάρτες κανονικών διανυσμάτων (**normal maps**) → κανονικά διανύσματα ανά fragment



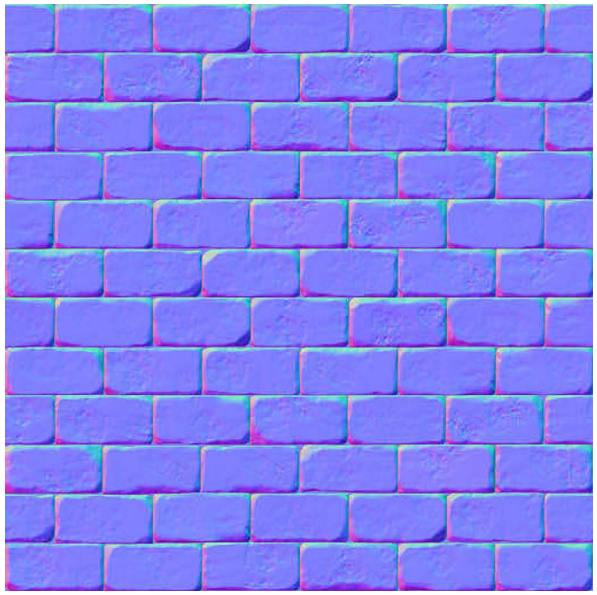
# Χάρτες κανονικών διανυσμάτων

- **Normal mapping** (ή **bump mapping**): χρήση ενός κανονικού διανύσματος ανά fragment αντί για ένα κανονικό διάνυσμα ανά επιφάνεια



# Χάρτες κανονικών διανυσμάτων

- Τα κανονικά διανύσματα αποθηκεύονται σε μία εικόνα υφής:



- Μετασχηματισμός από συντεταγμένες κανονικού διανύσματος σε (R, G, B) χρώμα

# Χάρτες κανονικών διανυσμάτων

- Fragment shader:

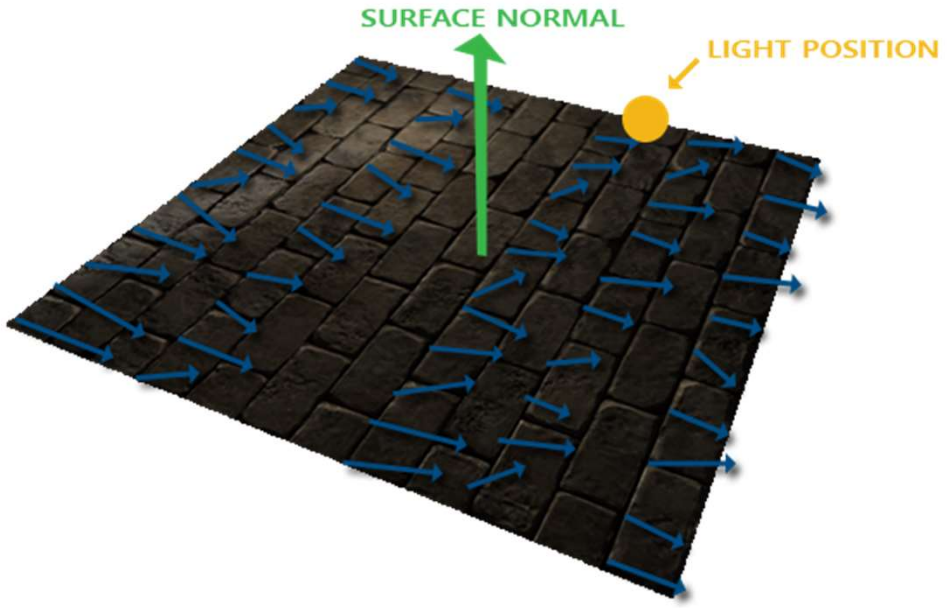
```
uniform sampler2D normalMap;

// obtain normal from normal map in range (0,1)
vec3 normal = texture2D(normalMap, coord).rgb;
// transform normal vector to range (-1,1)
vec3 norm = normalize(normal * 2.0 - 1.0);
//lighting…
```

# Χάρτες κανονικών διανυσμάτων

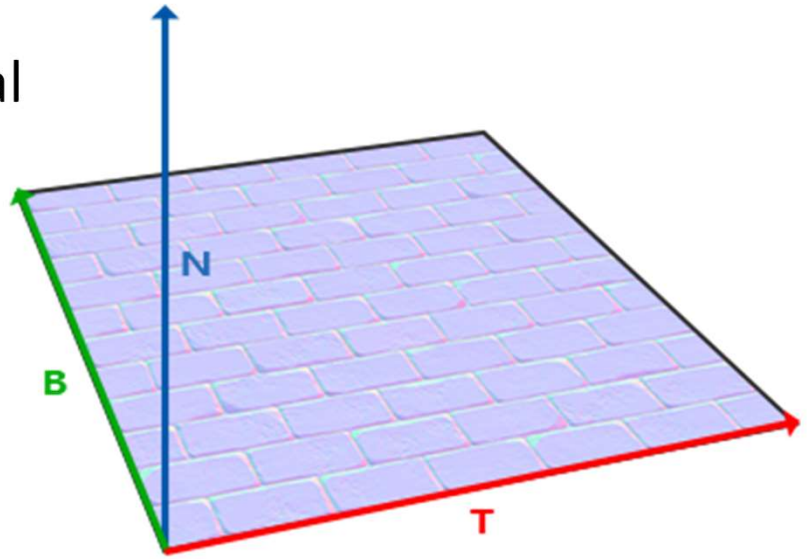
- Αν περιστρέψουμε το επίπεδο:

model = glm::rotate(model, glm::radians(-89.0f), glm::vec3(1.0f, 0.0f, 0.0f));



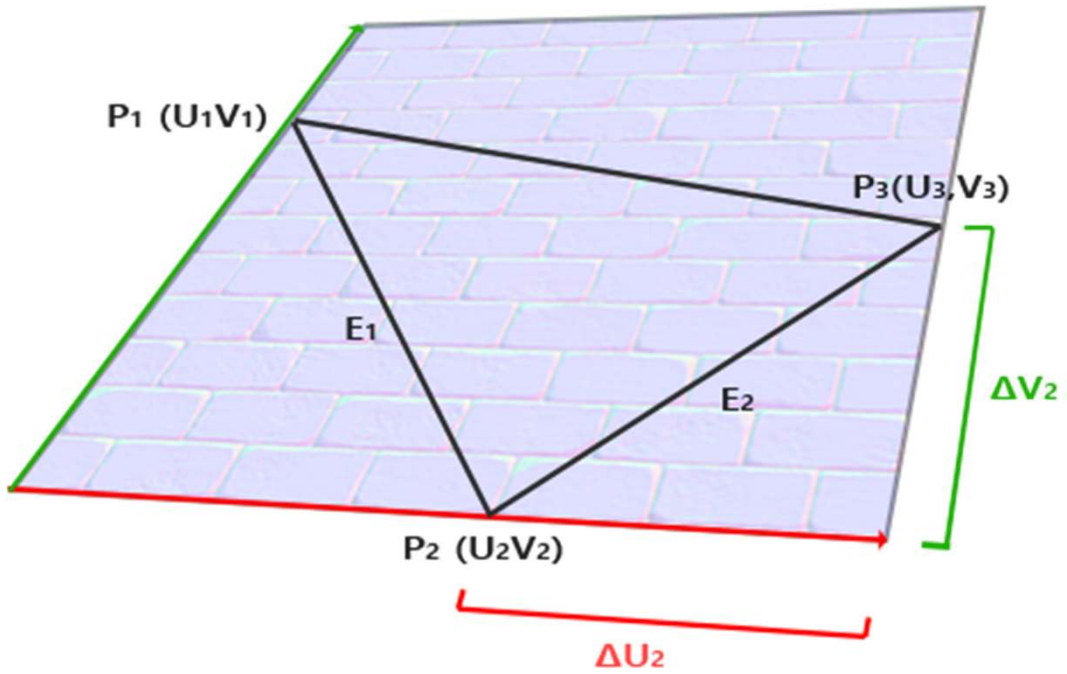
# Χάρτες κανονικών διανυσμάτων

- Λύση: φωτισμός σε καινούριο σύστημα συντεταγμένων (**tangent space**)
- **TBN matrix** (tangent, bitangent, normal vectors)
- Μετασχηματισμός από το τοπικό σ.σ. της υφής στο σ.σ. του κόσμου



# Χάρτες κανονικών διανυσμάτων

$$E_1 = \Delta U_1 T + \Delta V_1 B$$
$$E_2 = \Delta U_2 T + \Delta V_2 B$$



# Χάρτες κανονικών διανυσμάτων

$$E_1 = \Delta U_1 T + \Delta V_1 B$$
$$E_2 = \Delta U_2 T + \Delta V_2 B$$

Μπορεί να γραφεί ως :

$$(E_{1x}, E_{1y}, E_{1z}) = \Delta U_1 (T_x, T_y, T_z) + \Delta V_1 (B_x, B_y, B_z)$$
$$(E_{2x}, E_{2y}, E_{2z}) = \Delta U_2 (T_x, T_y, T_z) + \Delta V_2 (B_x, B_y, B_z)$$

Με μορφή πολλαπλασιασμού πινάκων :

$$\begin{bmatrix} E_{1x} & E_{1y} & E_{1z} \\ E_{2x} & E_{2y} & E_{2z} \end{bmatrix} = \begin{bmatrix} \Delta U_1 & \Delta V_1 \\ \Delta U_2 & \Delta V_2 \end{bmatrix} \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix}$$

# Χάρτες κανονικών διανυσμάτων

Πολλαπλασιασμός με τον αντίστροφο πίνακα ΔUΔV (λύση ως προς Τ και Β):

$$\begin{bmatrix} \Delta U_1 & \Delta V_1 \\ \Delta U_2 & \Delta V_2 \end{bmatrix}^{-1} \begin{bmatrix} E_{1x} & E_{1y} & E_{1z} \\ E_{2x} & E_{2y} & E_{2z} \end{bmatrix} = \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix}$$

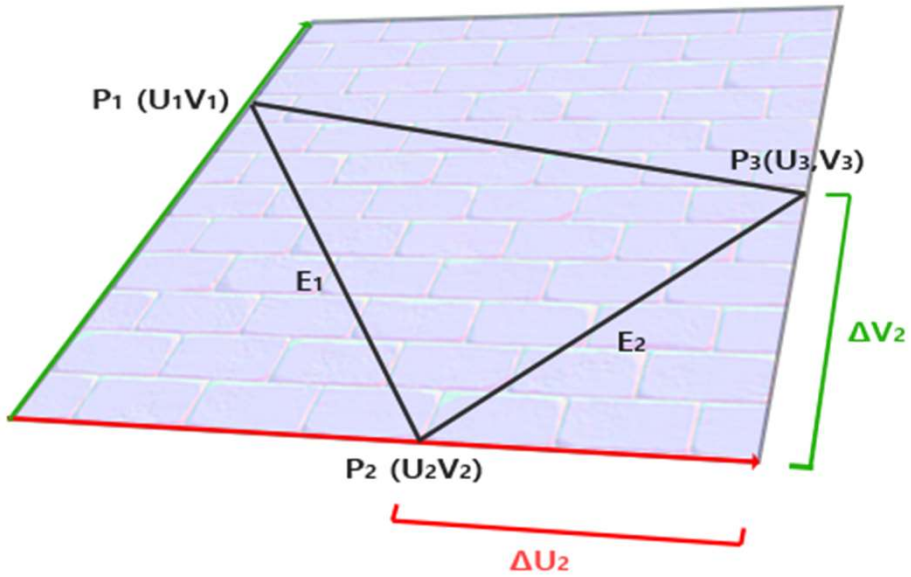
Υπολογισμός του αντιστρόφου πίνακα ΔUΔV.

$$\begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix} = \frac{1}{\Delta U_1 \Delta V_2 - \Delta U_2 \Delta V_1} \begin{bmatrix} \Delta V_2 & -\Delta V_1 \\ -\Delta U_2 & \Delta U_1 \end{bmatrix} \begin{bmatrix} E_{1x} & E_{1y} & E_{1z} \\ E_{2x} & E_{2y} & E_{2z} \end{bmatrix}$$

# Χάρτες κανονικών διανυσμάτων

```
//calculate E1, E2
glm::vec3 edge1 = pos2 - pos1;
glm::vec3 edge2 = pos3 - pos1;

//calculate DUV1, DUV2
glm::vec2 deltaUV1 = uv2 - uv1;
glm::vec2 deltaUV2 = uv3 - uv1;
```



# Χάρτες κανονικών διανυσμάτων

```
// positions
glm::vec3 pos1(-1.0,  1.0, 0.0);
glm::vec3 pos2(-1.0, -1.0, 0.0);
glm::vec3 pos3( 1.0, -1.0, 0.0);
glm::vec3 pos4( 1.0,  1.0, 0.0);
// texture coordinates
glm::vec2 uv1(0.0, 1.0);
glm::vec2 uv2(0.0, 0.0);
glm::vec2 uv3(1.0, 0.0);
glm::vec2 uv4(1.0, 1.0);
// normal vector
glm::vec3 nm(0.0, 0.0, 1.0);
```

# Χάρτες κανονικών διανυσμάτων

```
float f = 1.0f / (deltaUV1.x * deltaUV2.y - deltaUV2.x * deltaUV1.y);
//calculate tangent
glm::vec3 tangent1;
tangent1.x = f * (deltaUV2.y * edge1.x - deltaUV1.y * edge2.x);
tangent1.y = f * (deltaUV2.y * edge1.y - deltaUV1.y * edge2.y);
tangent1.z = f * (deltaUV2.y * edge1.z - deltaUV1.y * edge2.z);
tangent1 = glm::normalize(tangent1);
//calculate bitangent
glm::vec3 bitangent1;
bitangent1.x = f * (-deltaUV2.x * edge1.x + deltaUV1.x * edge2.x);
bitangent1.y = f * (-deltaUV2.x * edge1.y + deltaUV1.x * edge2.y);
bitangent1.z = f * (-deltaUV2.x * edge1.z + deltaUV1.x * edge2.z);
bitangent1 = glm::normalize(bitangent1);
```

# Χάρτες κανονικών διανυσμάτων

```
float vertices[] = {
 // positions          // normal          // tex coords  // tangent                          // bitangent
pos1.x, pos1.y, pos1.z, nm.x, nm.y, nm.z, uv1.x, uv1.y, tangent1.x, tangent1.y, tangent1.z, bitangent1.x, bitangent1.y, bitangent1.z,
pos2.x, pos2.y, pos2.z, nm.x, nm.y, nm.z, uv2.x, uv2.y, tangent1.x, tangent1.y, tangent1.z, bitangent1.x, bitangent1.y, bitangent1.z,
pos3.x, pos3.y, pos3.z, nm.x, nm.y, nm.z, uv3.x, uv3.y, tangent1.x, tangent1.y, tangent1.z, bitangent1.x, bitangent1.y, bitangent1.z,

pos1.x, pos1.y, pos1.z, nm.x, nm.y, nm.z, uv1.x, uv1.y, tangent2.x, tangent2.y, tangent2.z, bitangent2.x, bitangent2.y, bitangent2.z,
pos3.x, pos3.y, pos3.z, nm.x, nm.y, nm.z, uv3.x, uv3.y, tangent2.x, tangent2.y, tangent2.z, bitangent2.x, bitangent2.y, bitangent2.z,
pos4.x, pos4.y, pos4.z, nm.x, nm.y, nm.z, uv4.x, uv4.y, tangent2.x, tangent2.y, tangent2.z, bitangent2.x, bitangent2.y, bitangent2.z
};
```

# Χάρτες κανονικών διανυσμάτων

- Vertex shader:

```
…
layout(location = 3) in vec3 in_tangent;
layout(location = 4) In vec3 in_bitangent;

void main()
{
  //…
  vec3 T = normalize(vec3(model * vec4(in_tangent,   0.0)));
  vec3 B = normalize(vec3(model * vec4(in_bitangent, 0.0)));
  vec3 N = normalize(vec3(model * vec4(in_normal,    0.0)));
  mat3 TBN = mat3(T, B, N);
}
```

# Χάρτες κανονικών διανυσμάτων

- Πολλοί τρόποι υπολογισμού του φωτισμού:

1) Μεταφορά του πίνακα TBN στον fragm. shader → μετασχ. του normal της υφής στο σ.σ. του κόσμου

2) Υπολογισμός του αντίστροφου TBN και μεταφορά στον fragm. shader → μετασχ. των υπόλοιπων μεταβλητών φωτισμού στο σ.σ. Tangent

3) Υπολογισμός του αντίστροφου TBN, υπολογισμός των υπόλοιπων μεταβλητών φωτισμού στο σ.σ. Tangent → μεταφορά των μεταβλητών στον fragm. shader

# Χάρτες κανονικών διανυσμάτων

- Vertex shader:

```
mat3 TBN = transpose(mat3(T, B, N));
TangentLightPos = TBN * lightPos;
TangentViewPos  = TBN * viewPos;
TangentFragPos  = TBN * vec3(model * vec4(in_position, 1.0));
```

# Χάρτες κανονικών διανυσμάτων

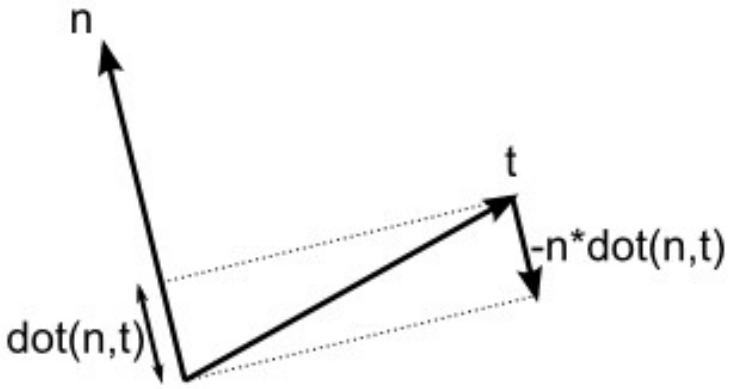
- Καλύτερο αποτέλεσμα:

```
//we only need T, N → we can calculate B
vec3 T = normalize(normal * in_tangent);
vec3 N = normalize(normal * in_normal);

// re-orthogonalize T with respect to N
T = normalize(T - dot(T, N) * N);

//the vector B results from the cross product of T and N
vec3 B = cross(N, T);

mat3 TBN = mat3(T, B, N)
```



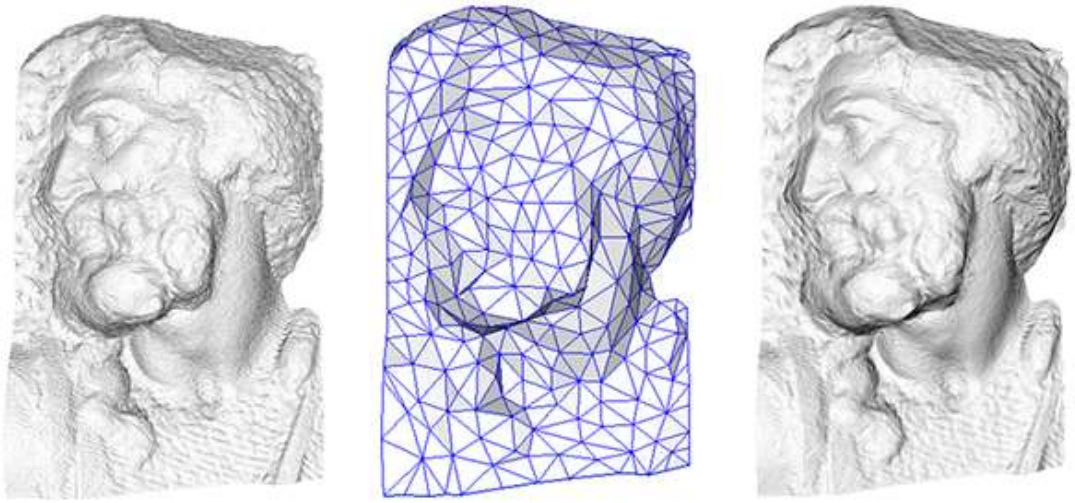
# Άσκηση

- Περιστροφή του επίπεδου στο χρόνο

# Χάρτες κανονικών διανυσμάτων

- Γιατί χρησιμοποιούμε χάρτες κανονικών διανυσμάτων:

Πολλές λεπτομέρειες με μικρό μέγεθος γεωμετρικής πληροφορίας



# Πηγές

Για πιο αναλυτική περιγραφή των εξισώσεων για την δημιουργία των Tangent και Bitangent:

http://ogldev.atspace.co.uk/www/tutorial26/tutorial26.html