A Comparison Study of GPU-Based Displacement Mapping Methods for Real-Time Terrain Visualization

César González Segura
Mariano Pérez Martínez
Juan M. Orduña Huertas

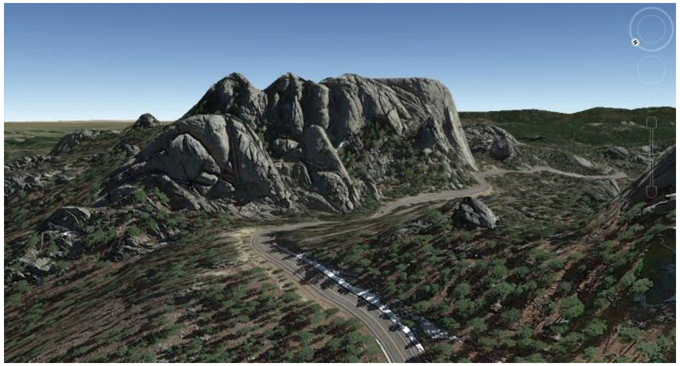
Dpto. de Informática, Universidad de Valencia
Juan.orduna@uv.es

GOBIERNO DE ESPAÑA
MINISTERIO DE ECONOMÍA, INDUSTRIA Y COMPETITIVIDAD

Escola Tècnica Superior d'Enginyeria

Introduction

- Real-time terrain visualization is a key research field in computer graphics.
- It is widely used in Geographic Information Systems (GIS), computer games and civil or military simulators[1],[2],[3].
- These applications must display a high visual quality terrain with interactive frame rates.
  - Terrain datasets usually exceed the capabilities of available hardware.
  - Applications must adjust the quality and complexity in a view-dependent manner.



GOBIERNO DE ESPAÑA MINISTERIO DE ECONOMÍA, INDUSTRIA Y COMPETITIVIDAD

Escola Tècnica Superior d'Enginyeria

Level of detail management

- Traditional techniques used to manage the level of detail employed CPU algorithms[4].
  - Modern GPU architectures feature a more efficient method using GPU based algorithms[5],[6],[7].

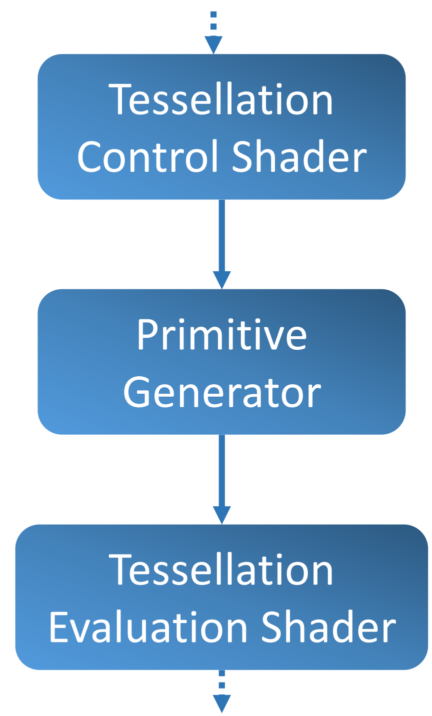
- One way to reduce geometry complexity is through applying displacement mapping on the GPU[8], using either:
  - *Per-vertex displacement mapping*, displacing vertices of a tessellated mesh[9].
  - *Per-pixel displacement mapping*, displacing the texture coordinates of the base mesh.

Parallax mapping

- Parallax mapping uses the elevation information stored in a height map to modify the texture coordinates of the base mesh, approximating them to the real texture.
- It was designed to enhance surfaces with small differences in elevation (floors, walls, etc).
    - When used it is used on a terrain surface the results are not correct, due to the high variability in terrain elevation.

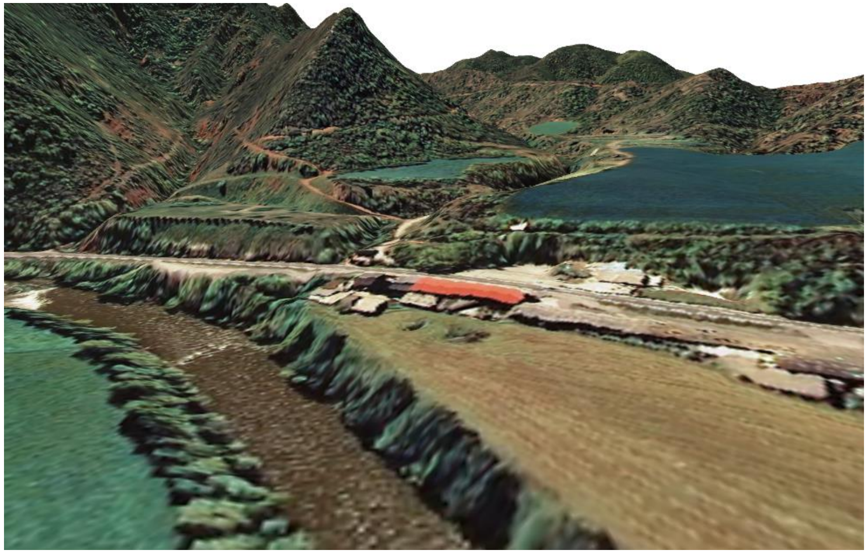
Per-vertex displacement mapping: tessellation

- Hardware tessellation adds geometry to a coarse mesh to form a smooth view-dependent level-of-detail representation.
- Starting from a patch primitive, the rendering pipeline generates the terrain geometry using the following steps:
  - The tessellation control shader determines how the patch must be divided and guarantees continuity between neighboring patches.
  - The primitive generator creates the geometry using the previous parameters.
  - The tessellation evaluation shader transforms the generated vertices to their final position.



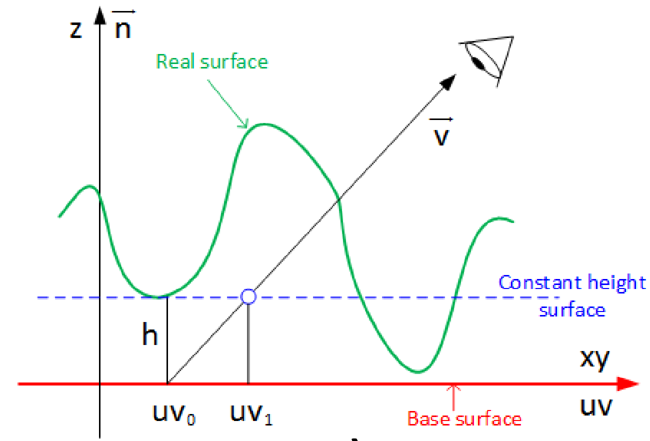
Per-vertex displacement mapping: tessellation

- Left: example terrain rendered with the maximum possible level of detail.
- Right: example terrain rendered with the minimum level of detail.

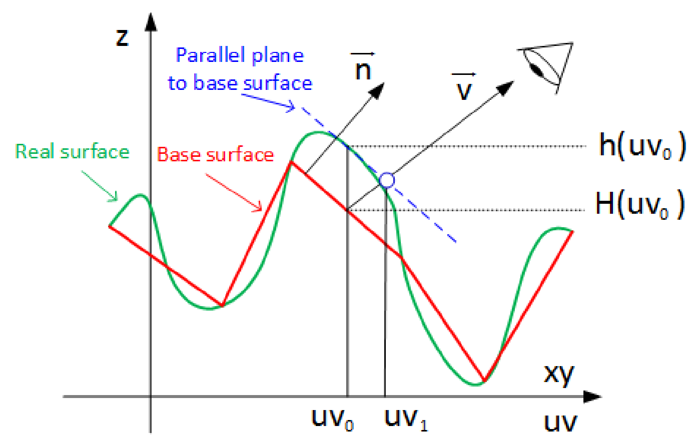


Parallax mapping and terrain rendering

- Parallax mapping corrects the texture coordinates using a horizontal plane as the base, as in figure *a)*[10].
- This method is not-well suited for terrain rendering, yielding an incorrect estimation specially when the camera is close to the surface.
- Our method consists of changing the base surface to the mesh created through tessellation, as in figure *b)*.
    - This modification allows parallax mapping to be used with tessellation for terrain rendering.



a)

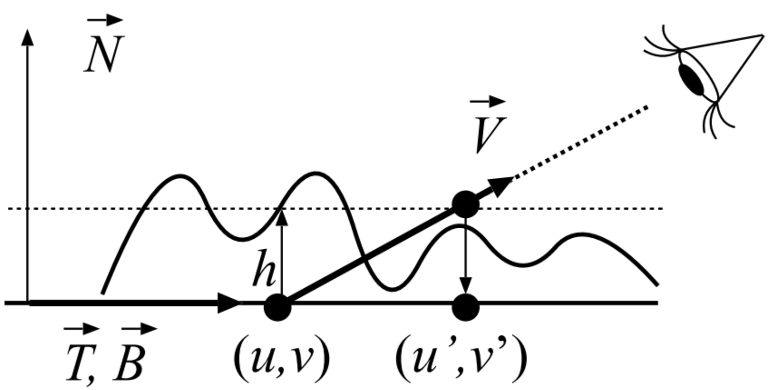


b)

Simple parallax mapping

- The simple parallax mapping method corrects the texture coordinates (u,v) by projecting the intersection point of the view vector ($\vec{V}$) with the real surface ($\bar{h}$) on the base surface ($\overline{TB}$).
- The shader code has been adapted from the original parallax mapping method to reflect the changes explained in the previous slide.
- However, simple parallax mapping can yield an incorrect estimation, specially if the eye vector is close to being parallel to the base surface.

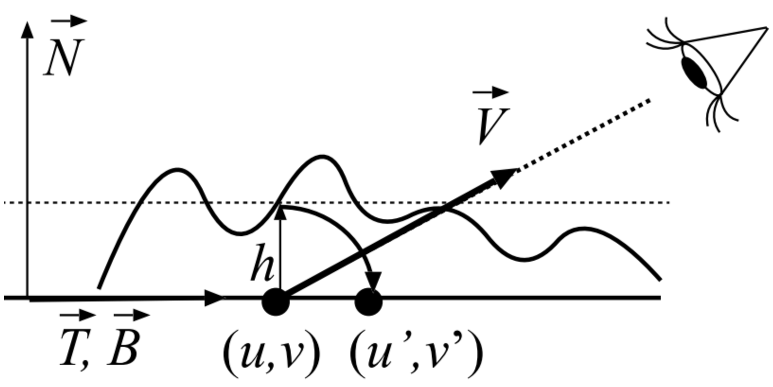
```
float h = texture(uHeightNormalmap,teTexCoor).r;
vec3 v = normalize(teEyeVector);
vec3 n = normalize(teNormal);
newTexCoords = (teTexCoords+(h-teHBase)*n.z*
                 v.xy/max(dot(v,n),0.5));
```



Offset limited parallax mapping

- Using simple parallax mapping, the target texture coordinates can grow excessively (even towards infinity) if the view vector ($\vec{V}$) is parallel to the ground plane ($\overline{TB}$).
- A solution to this problem is applying a maximum offset to the texture coordinate correction.
- Using the Welsh approximation, the offset is limited to one and the calculations are further simplified.

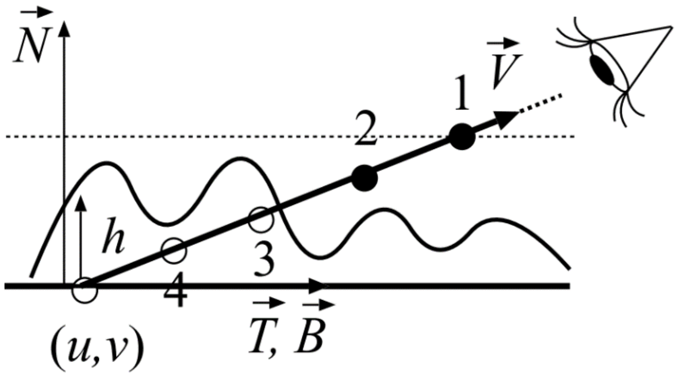
```
newTexCoords = teTexCoords+(h−teHBase)∗n.z∗v.xy;
```



Iterative parallax mapping

- It is unlikely that the correct texture coordinates are calculated in a single attempt.
- Accuracy can be improved by applying the simple parallax mapping method iteratively a few times[8].
- The number of iterations can be modified, getting better results as the number of iterations increases.

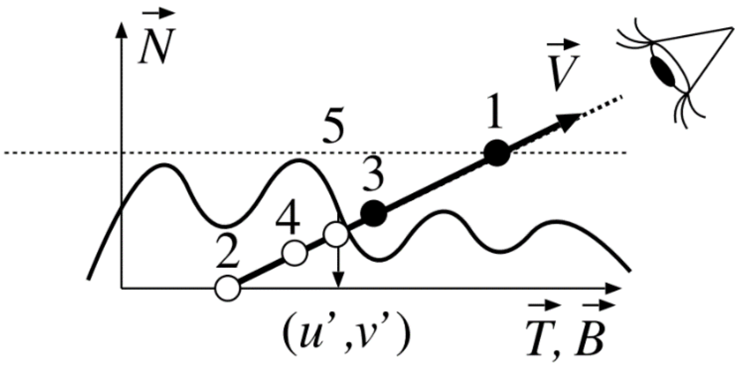
```
vec3 v = normalize(teEyeVector);
vec3 n = normalize(teNormal);
vec2 uv = teTexCoords;
for ( int i = 0; i < PAR_ITER; i ++ ) {
   float h = texture(uHeightNormalmap, uv).r;
   uv += (h - teHBase) * n.z * v.xy;
}
newTexCoords = uv;
```



Binary search parallax mapping

- A better result can be obtained by performing a binary search on each iteration, instead of applying the simple parallax mapping method[8].
- Endpoints are placed on the intersection point of the view vector ($\vec{V}$) with the base surface and are updated proportionally to the triangle size. The number of iterations is limited, as in the iterative method.
- This implementation yields better results than a simple iterative method.

```
vec3 v = normalize(teEyeVector);
vec3 UVH1 = vec3(teTexCoords, teHBase)+tsize*v;
vec3 UVH2 = vec3(teTexCoords, teHBase)-tsize*v;
for ( int i = 0; i < PAR_ITER; i ++ ) {
  vec3 UVHm = (STH1 + STH2) * 0.5;
  float h = texture(uHeightNormalmap, UVHm).r;
  if (h < UVHm.z) {d1 = h-UVHm.z;UVH1 = UVHm;}
  else            {d2 = h-UVHm.z;UVH2 = UVHm;}
}
newTexCoords = vec2(UVH2+(UVH1-UVH2)*d2/(d2-d1))
```



Performance evaluation

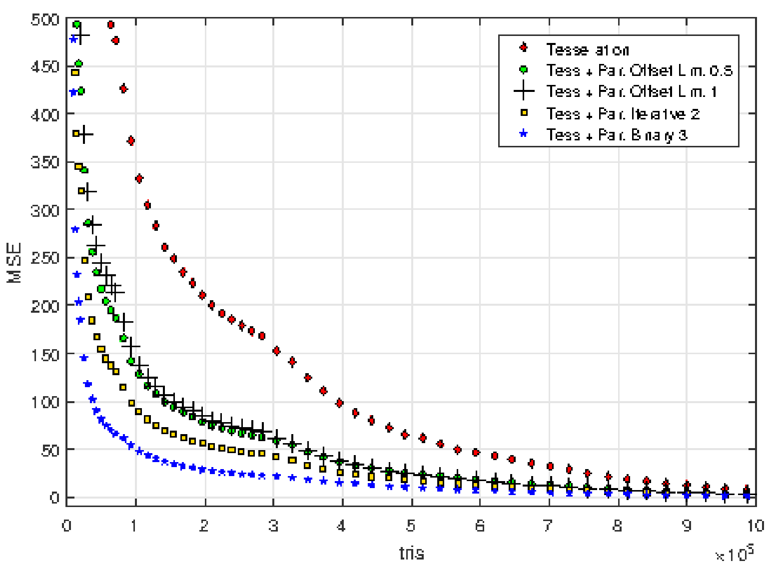
- We have implemented a terrain visualization application which performs a flight simulation over the terrain model using different visualization parameters.
- The dataset used for the evaluation is a terrain model of the Pyrenees, with an extension of 15360x25600 meters and a total size of around 400 MB.
- We have measured two parameters:
  - The Mean Square Error (MSE) between the reference rendered image and the corresponding technique.
  - The Frames per Second (FPS) obtained while executing the rendering.

Analyzed techniques
Tessellation
Tesselation + parallax (0.5 offset)
Tesselation + parallax (1.0 offset)
Tessellation + iterative parallax (2 iter)
Tessellation + binary search parallax (3 iter)

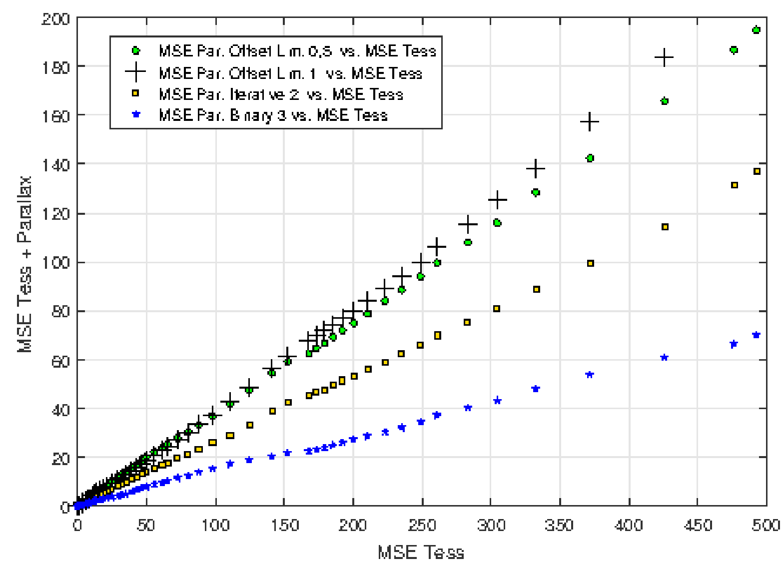
Test Machine
Intel Core i7 4790K
12 GB RAM
Nvidia GTX 590 / 650 / 970
Windows 7 x64

Performance evaluation: MSE

- Figure *a)* shows the MSE as a function of the number of triangles drawn in the scene (lower is better).
- Figure *b)* shows the ratio between the MSE with only tessellation and MSE with each of the applied techniques (lower is better).
- Results show how binary search parallax mapping achieves the best results, with an improvement of around 610% in terms of MSE.



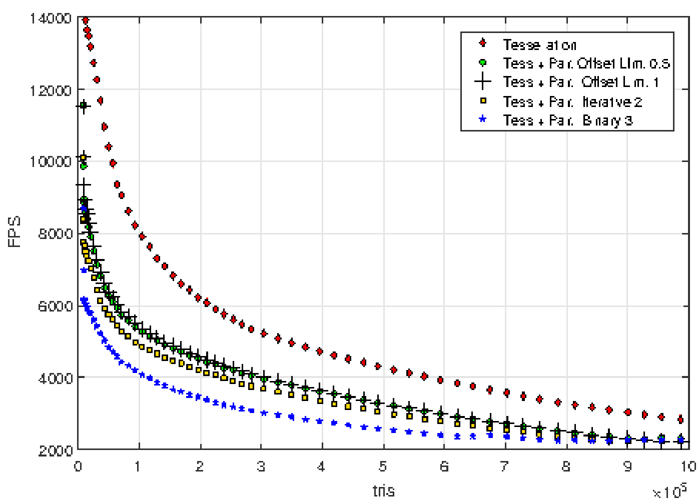
a)



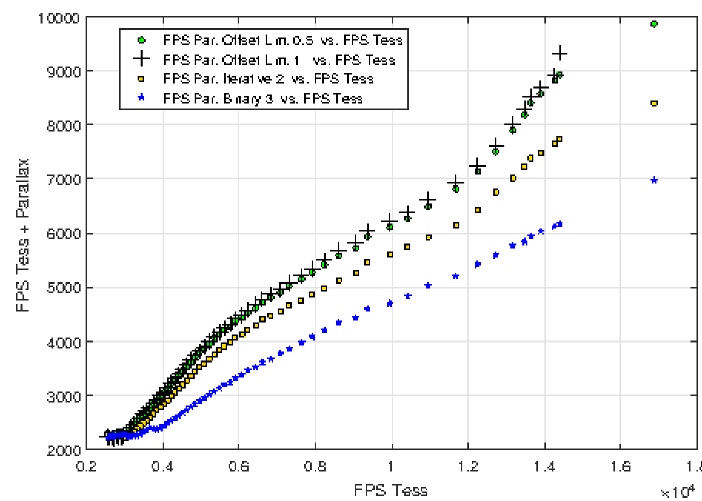
b)

Performance evaluation: FPS

- Figure *a)* shows the FPS as a function of the number of triangles drawn in the scene (higher is better).
- Figure *b)* shows the ratio between the FPS with only tessellation and FPS with each of the applied techniques (lower is better).
- Regarding FPS, the best results are obtained by offset limiting parallax mapping.
  - However, even if the FPS of the binary search method are the lowest, they are an order of magnitude over the required FPS for interactive applications.



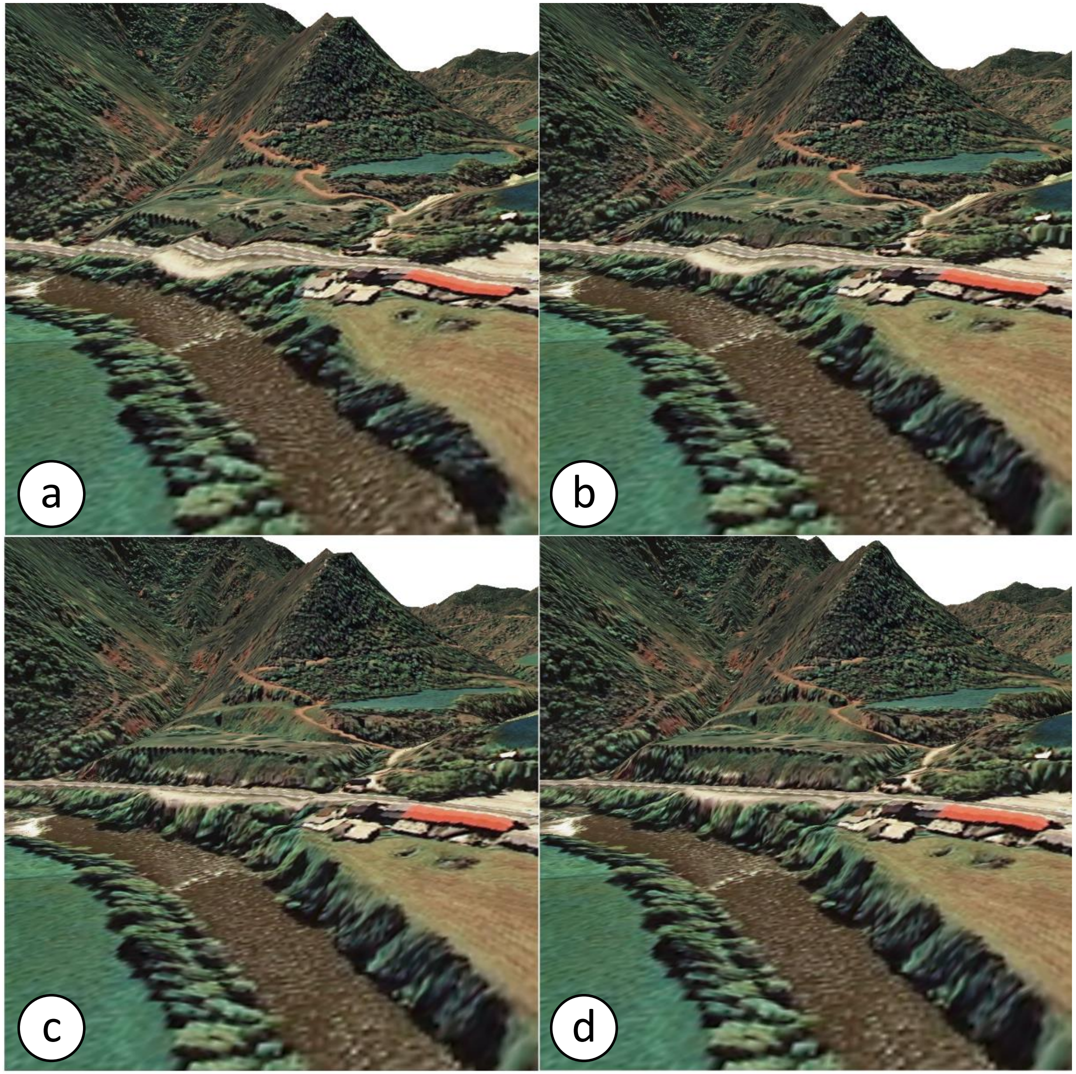
a)



b)

Performance evaluation: visual fidelity

- We have compared the visual results of each method:
  - Figure *a)* shows tessellation with the min. resolution (ca. 2000 tris.).
  - Figure *b)* shows offset limiting parallax mapping applied to the previous case.
  - Figure *c)* shows binary search parallax mapping applied to the *b*) case.
  - Figure *d)* shows tessellation with the max. resolution (ca. 130000 tris.).



Conclusions

- We have proposed a technique which makes parallax mapping compatible with terrain rendering and hardware tessellation.
- We have performed a comparative study of different parallax mapping implementations. The performance evaluation results show how all the parallax mapping methods improve the visual quality of the picture.
- The binary search method seems the most appropriate method for current GPUs, yielding the best relationship between visual quality and frame rate.

A Comparison Study of GPU-Based Displacement Mapping Methods for Real-Time Terrain Visualization

César González Segura
Mariano Pérez Martínez
Juan M. Orduña Huertas

Dpto. de Informática, Universidad de Valencia
Juan.orduna@uv.es

GOBIERNO DE ESPAÑA
MINISTERIO DE ECONOMÍA, INDUSTRIA Y COMPETITIVIDAD

Escola Tècnica Superior d'Enginyeria