ESAT
INNOVATION SCHOOL
ESCUELA SUPERIOR DE ARTE Y TECNOLOGÍA

GRAPHICS ENGINES:

Utlop

Graphics engines programming
L5
Jose Luis Hidalgo
2021/2022
Javier Segura Forés

1.- Engine features

1.- ECS based.

The engine is ECD based which means it uses an Entity-Component-System, being capable of adding entities and components to those entities. Then, all the systems run on the entities that contain their components.

```
enum ComponentID {
  kLocalTRComp = 1,
  kCameraComp = 2,
  kHeritageComp = 4,
  kLightComp = 8,
  kTypeLightComp = 16,
  kRenderComp = 32
};

enum ComponentPos {
  kLocalTRCompPos = 0,
  kCameraCompPos = 1,
  kHeritageCompPos = 2,
  kLightCompPos = 3,
  kTypeLightCompPos = 4,
  kRenderCompPos = 5
};
```

2.- Display List

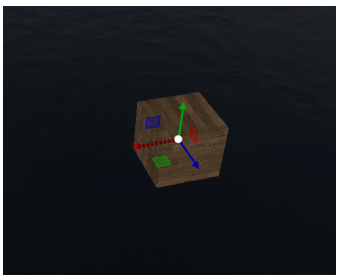
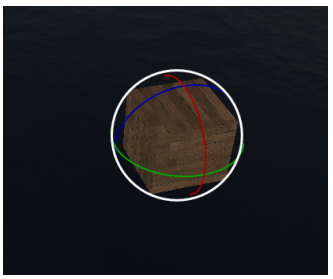
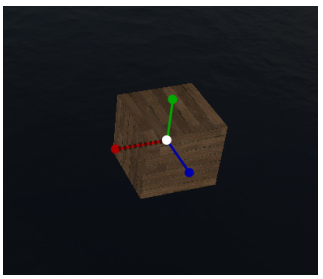
The engine works with a Display list, which means all the GPU operations are called at same time and without logical operations between. This is essential for multi-threading since the GPU can go on the main thread while the logical operations can go on other threads. It simply calls “addCommand” and queues it in the displaylist.

```
struct Command {
  virtual void executeOnGPU() = 0;
};
```

All commands heredate from this struct and overrides the executeOnGPU function to run their corresponding tasks.

3.- Transforms

Any entity that inherits a local component can be translated, rotated or scaled as desired.

4.- Parent / child hierarchy

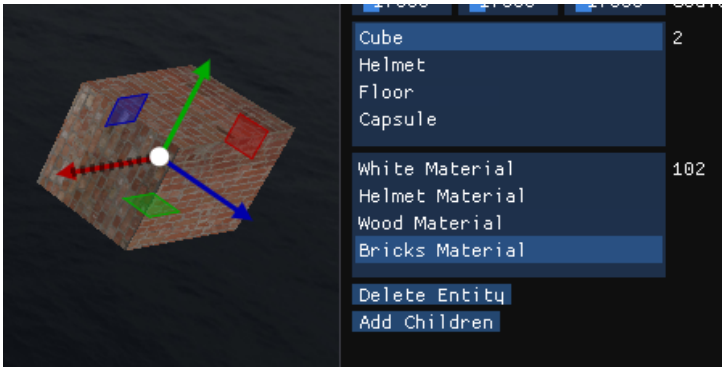
Game Objects can have parents, which means they get their model and use it to get a world space different from local one, translating when parent does, rotation and scaling (on purpose).

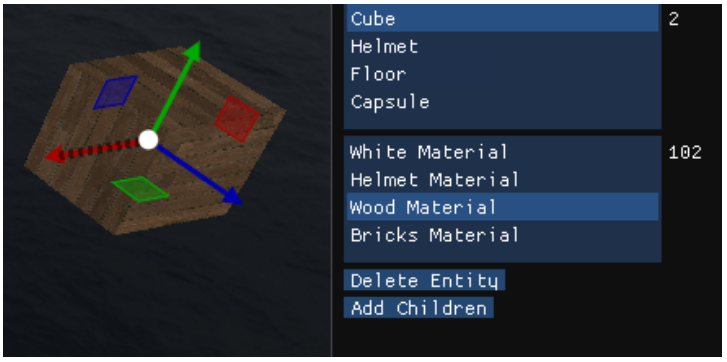
5.- Add / Delete entities at runtime, and add children too.

6.- Change dynamically between meshes (geometries and materials) freely.

Materials are abstracted so any material can be set to any model.

In the demo there are a few but implementing new ones takes less than 1 minute.





7.- Lightning

From the light source you can select a PointLight, Direction Light or SpotLight.

Textures receive that lightning and show more or less color depending on normals, specular maps, texture color and light color.

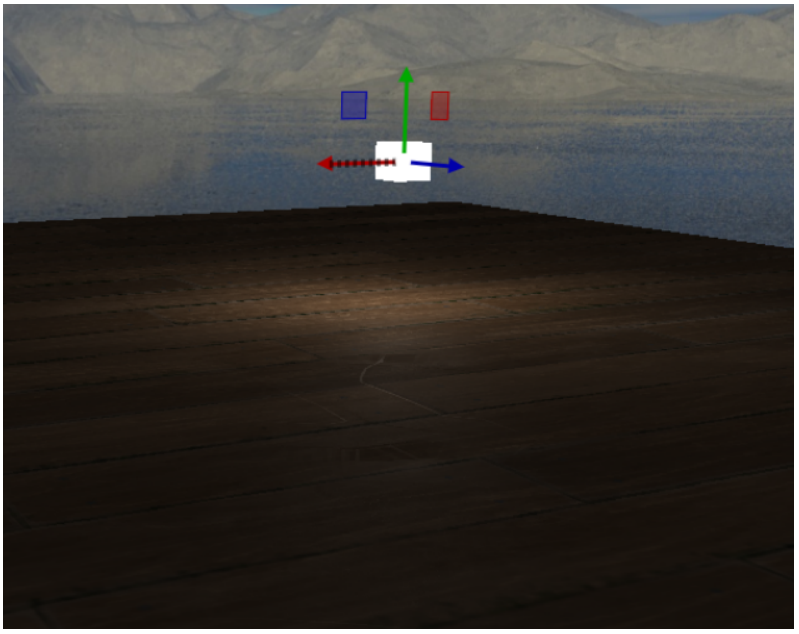
It only affects if the Game Objects does have a LightComponent.

With Light component

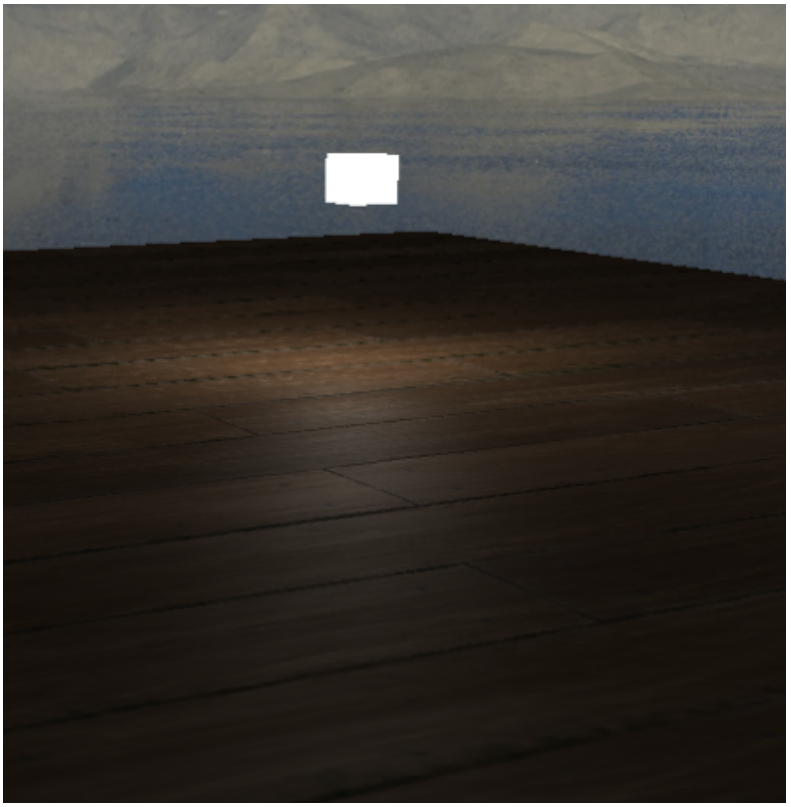
Without Light component

There is phong rendering. Textures' light changes get affected by specular maps that determine which area should get more affected by light.

Without specular map



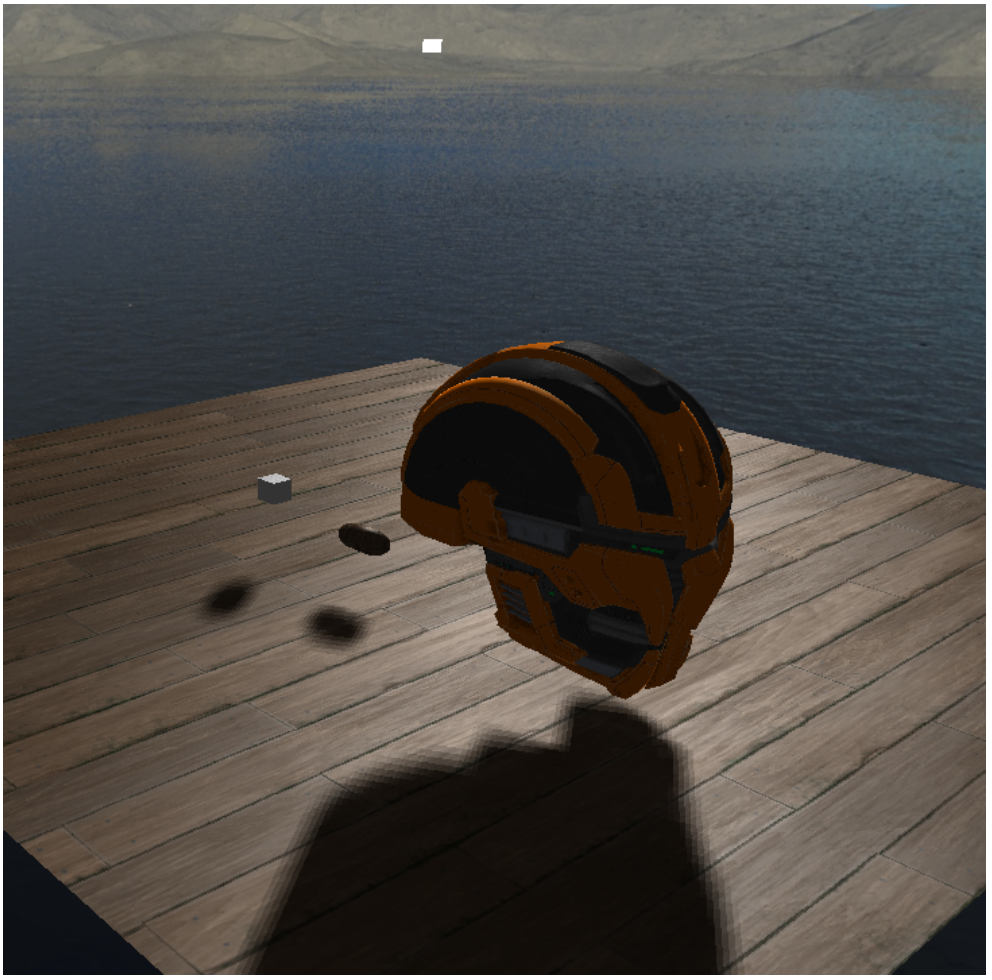
With Specular map



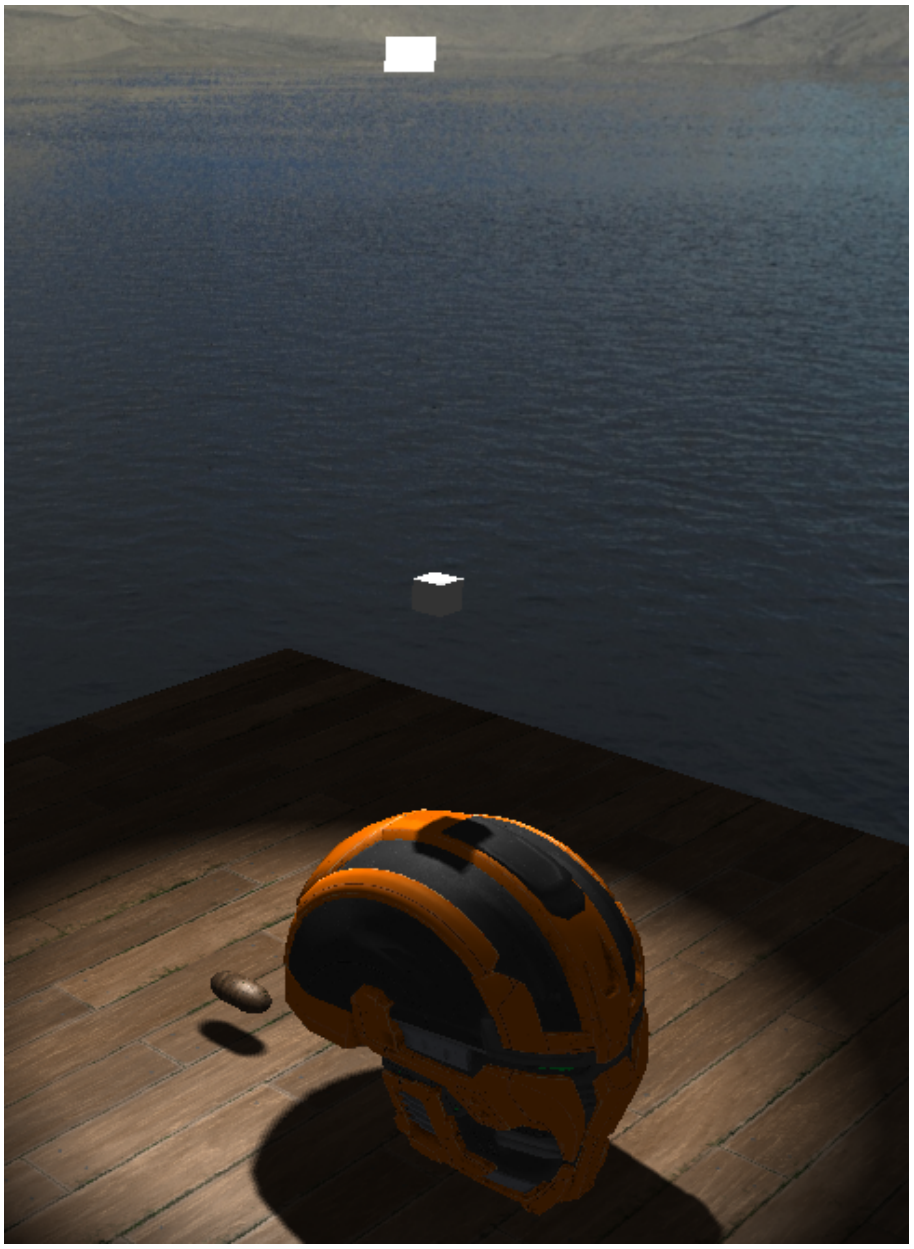
8.- Shadow mapping

Directional and Spotlight lightning projects shadows through a shadow map frame buffer.

Directional Light Shadows

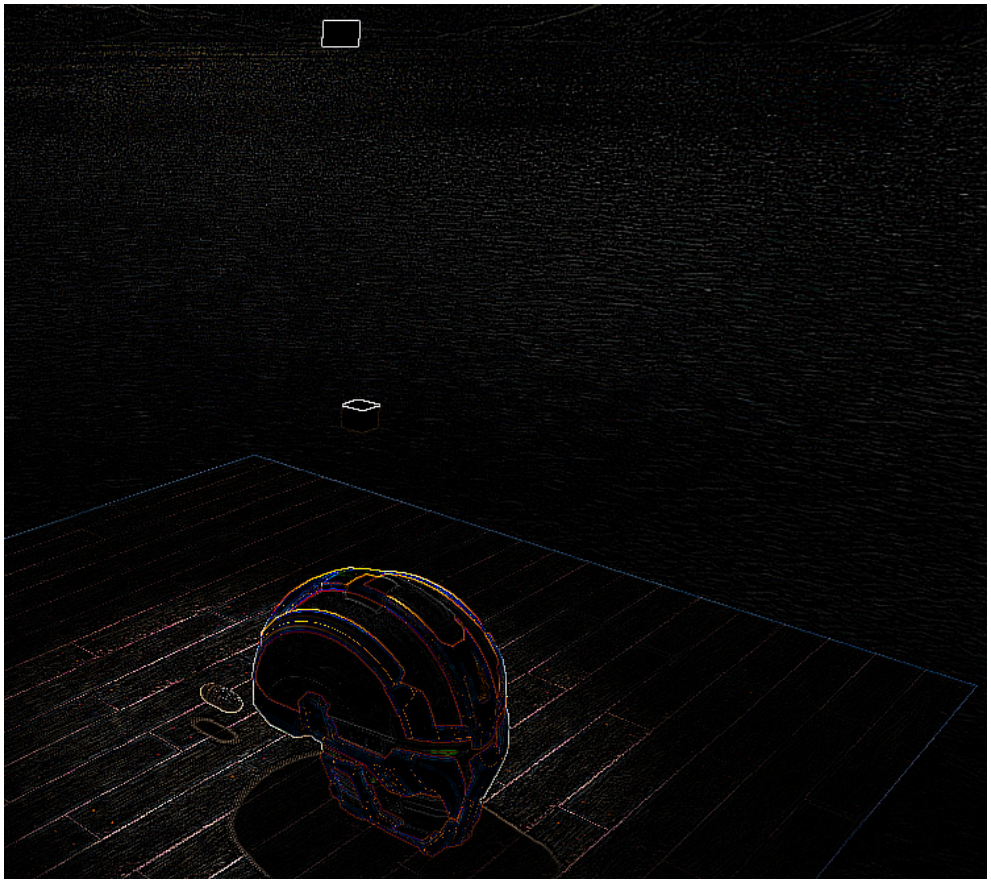


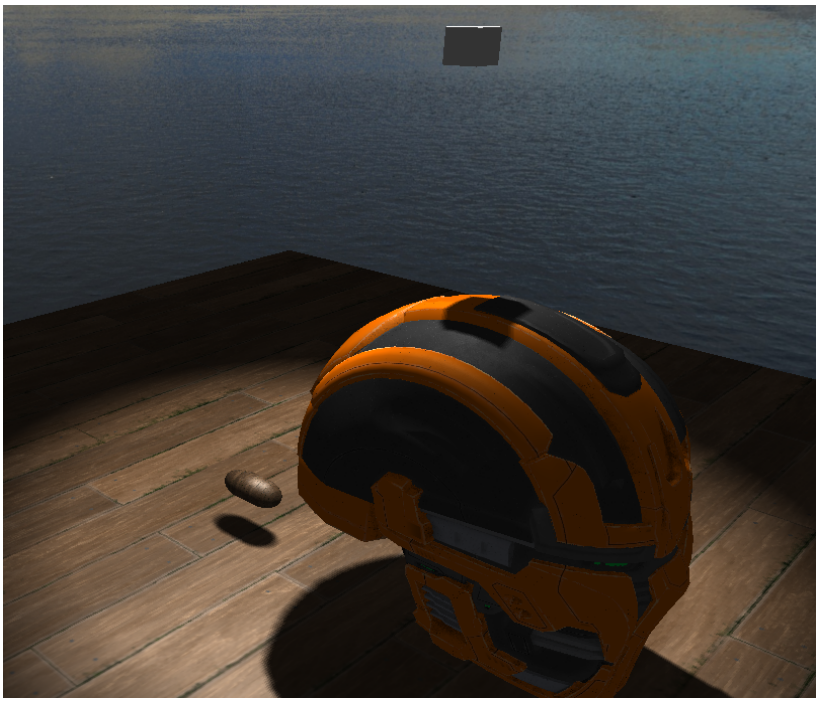
Spot Light shadows



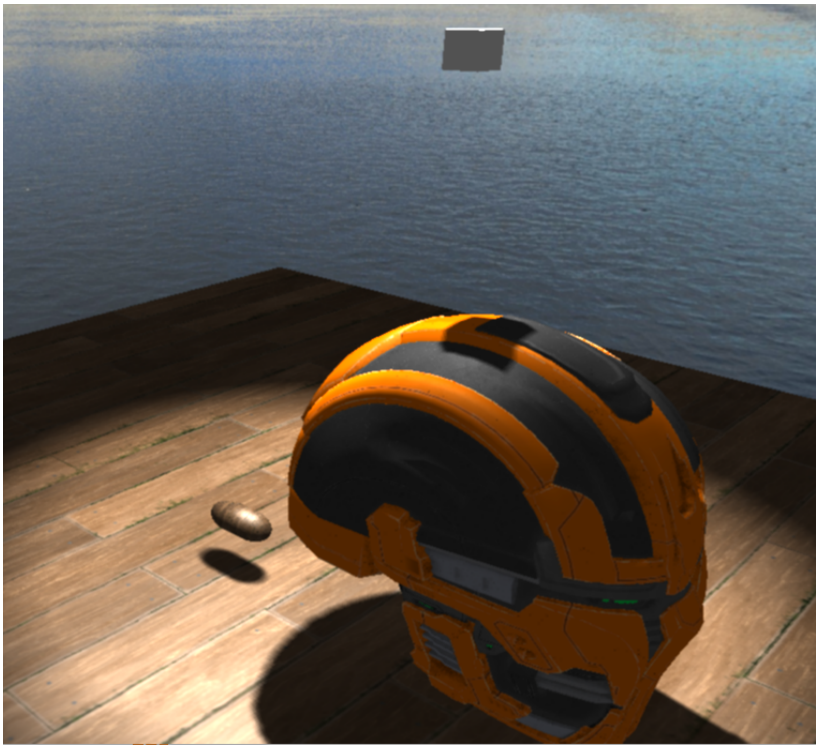
9.- Post-Processing

Utlop Engine uses a framebuffer texture to properly modify adjacent pixels in fragment shader, allowing post processes such as blur.





Without blur



With blur

10.- Loading scenes from database

Utlop engine can load from a database sql (.db) using sqlite3. It only has some variables stored but it can be easily extended.

Tabla: SAVED_SCENE

Filtrar en cualquier colu...

	ID	ID_COMPONENTS	LOCATION_X	LOCATION_Y	LOCATION_Z	MESH	MATERIAL	PARENT_IDX
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	1	49	0	0	0	0	0	-1
2	2	41	0	0	-20	1	1	-1

This is what loads the table above.

As an important note, the basic entity drawable has 41 as id components. Children ones do have 45 (41 + 4 as HeritageComponent is added). The first one is highly recommended to be 49 because it adds the TypeLightComponent (Light source).

2.- Implemented techniques

1.- Render to Texture.

In order to do post-processing, Utlop engine implements framebuffer. The framebuffer stores the drawed scene as a texture, and thanks to that the fragment shader can access any other pixel to make very different effects. It also allows you to shadow map the scene but we will talk about that later.

All the draw calls are sent to the framebuffer binded. Once the displaylist is done submitting the commands, it gets unbind and draws a “quad” with window height and width that we see like always. The framebuffer offers more functionality, like allowing to store the texture and keep doing operations, for example. Then, you can swap buffers, draw the framebuffer texture while logic operations are running without having to wait for them.

In the fragment shader we have a variable called Kernel which helps us to easily apply different effects.

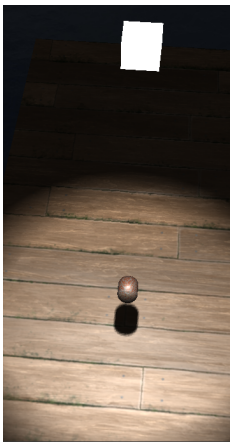
```
float kernelBlur[9] = float[](
    1.0 / 10, 2.0 / 10, 1.0 / 10,
    2.0 / 10, 4.0 / 10, 2.0 / 10,
    1.0 / 10, 2.0 / 10, 1.0 / 10
);
```

This, with an offset, will make cool effects.

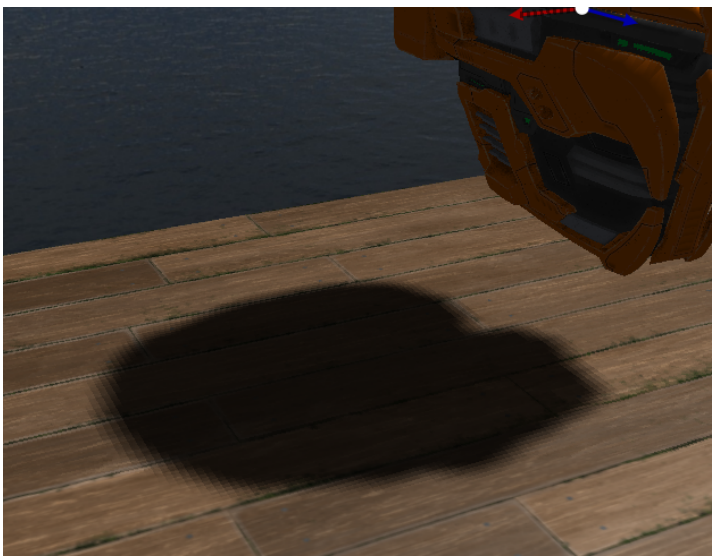
2.- Shadow maps

Is a framebuffer that only cares about depth. It renders to texture the depth of the scene given a light point. Is the absence of light due to occlusion. The closer the object is to the light source, the more shadow it projects to objects behind.

It requires the scene to be “draw” into the depth frame buffer, and then bind to another framebuffer that has to “re-draw” the scene having the shadow map texture already on gpu in order to draw less light on those pixels that are shadowed.



Furthermore, the depth is passed through the alpha channel and in the fragment shader we calculate the quantity of shadowing projected.
We use what is called Bias to reduce shadow acne, a common problem in shadow maps



3.- Strategies and solutions to development problems

The main problem of this development is being alone. The tasks are not divided like other groups so it took a lot of time to implement some things more advanced like shadows and I had to search for a lot of information not only the opengl one, but the display list and the ECS structure by myself.

The good part of this is that I have learned a lot in this process and I know every piece of code implemented, which is a super advantage versus other groups that are not used to working with other programers and other methodologies.

The development was plenty of try-error. I've spent a lot of time in this engine, working on every feature. Opengl has never been my strong point so getting familiar with the libraries and gl functions was a true challenge. I'm pretty happy with the results although it is very improvable. In fact, I'm thinking of developing it during summer to have a nice portfolio for further job interviews.

4.- Building the solution

Inside the folder "genie" there is the .bat file that compiles and creates the build folder with the solution. If it's not set by default, is highly recommended to change the solution to x64 and making the "UtlopTests" as default start project.