OpenGL

OpenGL : The origin

In the 80's, developing software for a wide range of graphic hardware was a pain. There wasn't any APIs.

In the early 90's, Silicon Graphics Inc. (SGI) was a leader in 3D graphics for workstations, that work with its own API, *IRIS GL*, easy to use and able to draw in immediate mode.

OpenGL : The origin

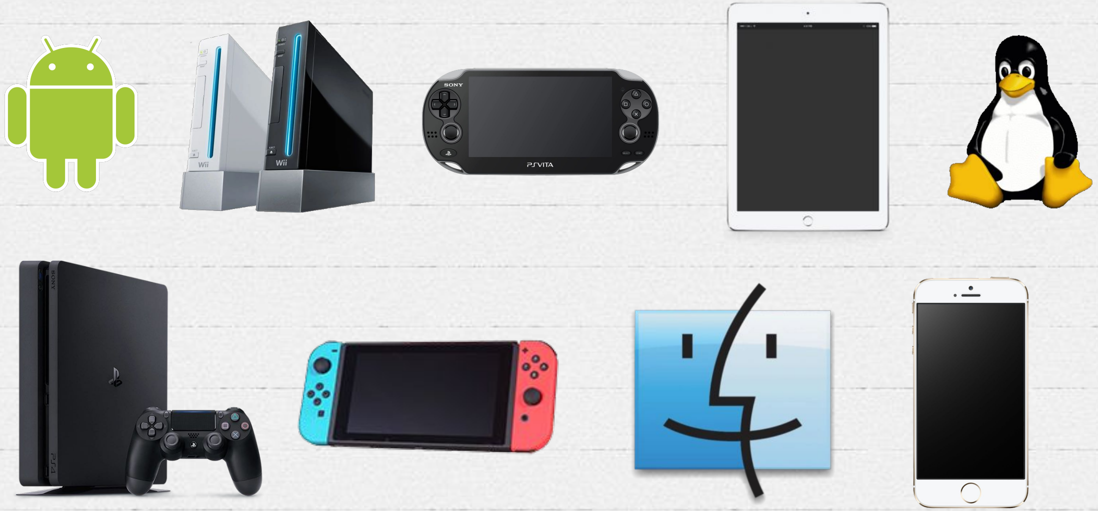
However, the 3D hardware market became very competitive.

To turn the tide and influence the market, SGI decided to turn IRIS GL into an open standard.

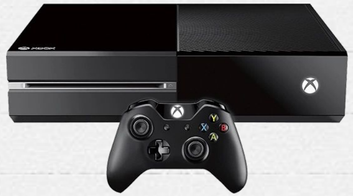
An Open GL ARB (Architectural Review Board) was created to establish and maintain its software specifications.

Today, Khronos group is responsible of OpenGL maintenance.

OpenGL support



✓



✗

New (3D) perspective

- ✗ We still will use SDL, but now within a OpenGL context
- ✗ Immediate Mode will be used for drawing. Buffers and shaders, for…

Video game engines !!

- ✗ Render module will adapt OpenGL (see *ModuleRenderer3D* class)
- ✗ A Camera module will be necessary (see *ModuleCamera3D* class)
- ✗ The racing game will be formed by simple primitives (see *Primitives* class)
- ✗ The math library will include more functionality

Our goal

YOUR TURN !

TODO No 1

```
// TODO 1: Create a Plane primitive. This uses the plane formula
// so you have to express the normal of the plane to create
// a plane centered around 0,0. Make that it draw the axis for reference
```

Theoretically, a plane is infinite. But the primitive will draw 200 units of width and height

TODO № 2

```
// TODO 2: Place the camera one unit up in Y and one unit to the right
// experiment with different camera placements, then use LookAt()
// to make it look at the center
```

Try to position the camera at different places in order to understand how it works. Use *LookAt()* to rotate the camera and point to a different position.

TODO Nº 3

```
// TODO 3: Make the camera go up/down when pressing R (up) F(down)
```

- ✗ Use KEY_REPEAT enum to add small increments to the camera position in order to reproduce a up & down movement
- ✗ **Position** and **Reference** must be update as well

TODO No 4

```
// TODO 4: Make the camera go forward (w) and backward with (s)
// Note that the vectors X/Y/Z contain the current axis of the camera
// you can read them to modify Position
```

- ✗ Here, you need to access the vectors (X,Y and Z) that define the camera axis in order to do a forward / backward movement
- ✗ **Position** and **Reference** must be update as well

TODO Nº 5

```
// TODO 5: Make the camera go left (a) and right with (d)
// Note that the vectors X/Y/Z contain the current axis of the camera
// you can read them to modify Position
```

- ✗ Here, you need to access the vectors (X,Y and Z) that define the camera axis in order to do a lateral movement
- ✗ **Position** and **Reference** must be update as well

TODO Nº 6

```
// TODO 6: Draw a sphere of 0.5f radius around the center
// Draw somewhere else a cube and a cylinder in wireframe
```

- ✗ Try to draw others primitives (even in *wireframe* render).
- ✗ Try to rotate them

Homework

```
// TODO (Homework): Rotate the camera with the mouse
```

There are different ways to do that. You can use X,Y,Z camera axis or work LookAt( ) method. A math function in glmath.h will be useful:

```
// “u” is the axis of rotation
// “angle” is the value to rotate
// “v” is the vector to rotate
vec3 rotate(const vec3 &u, float angle, const vec3 &v)
{
   return *(vec3*)&(rotate(angle, v) * vec4(u, 1.0f));
}
```

Next Week . . .

Bullet Integration