Par Olivier BAKOU et Hamadou DAO

**Master Jeux Video et big data**

DOSSIER DE REALISATION

**PROJET : mISE EN PLACE D’UNE HORLOGE 3D AVEC WEBGL**

SOMMAIRE

[**A-** **Présentation de l’environnement de travail** 2](#_Toc285865)

[1- Equipe de réalisation 2](#_Toc285866)

[2- Durée globale de réalisation et délais 2](#_Toc285867)

[3- Lieu de réalisation 2](#_Toc285868)

[4- Outils 2](#_Toc285869)

[5- Langages utilisés 2](#_Toc285870)

[**B-** **Plateforme d’exécution des rendus** 2](#_Toc285871)

[**C-** **Artefacts des activités réalisées** 3](#_Toc285872)

[1- Outils, Plugin ou add-on 3](#_Toc285873)

[2- Description du code et des procédures 3](#_Toc285874)

[3- Vue du code JavaScript imagé 3](#_Toc285875)

[4- Vue du code à intégrer 5](#_Toc285876)

[**ANNEXES** 10](#_Toc285877)

# **Présentation de l’environnement de travail**

## **Equipe de réalisation**

Afin d’être efficace dans la production des artefacts attendus, ci-dessous la répartition des tâches.

|  |  |  |
| --- | --- | --- |
| Fonction | Ressources principales | Ressources secondes |
| Etudes du projet | Olivier BAKOU | Hamadou DAO |
| Développeurs | Hamadou DAO | Olivier BAKOU |
| Traitement des images ou des textures | Olivier BAKOU | Hamadou DAO |
| Intégration des textures | Hamadou DAO | Olivier BAKOU |

**Nombre total des membres d’équipe : 2**

## **Durée globale de réalisation et délais**

**La durée globale du projet est de :** 1 jour

**Heure début :** 9h30

**Heure de fin :** 13H

**Délais de livraison :** -----

## **Lieu de réalisation**

* ESP

## **Outils**

* Sublime Text
* WebGL

## **Langages utilisés**

* JavaScript
* OpenGL

# **Plateforme d’exécution des rendus**

* Google Chrome 
* FireFox 

# **Artefacts des activités réalisées**

## **Outils, Plugin ou add-on**

* **OpenGL**

## **Description du code et des procédures**

1. Déclaration du **vertex Shader** pour la récupération des positions ou des coordonnées

***Type :*** *Voir le code*

1. Déclaration du **fragment shader** pour la pixalisation ou le rendu des coordonnées

***Type :*** *voir le code*

1. Edition de la fonction **init\_wgl** : fonction opengl pour l’initialisation des rendus

* ***Entrée :*** *--*
* ***Sortie :*** *--*

1. Edition de la fonction **draw\_wgl** : fonction opengl pour le maillage des données initialisées pour les shaders

* ***Entrée :*** *--*
* ***Sortie :*** *--*

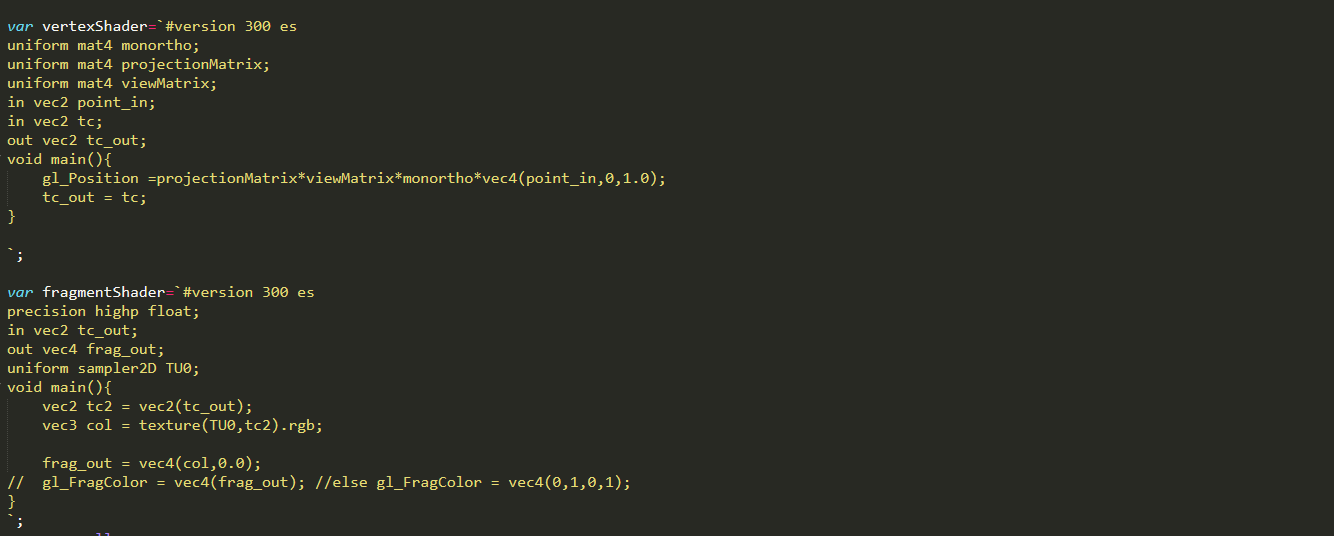
1. Edition de la **fonction** **f\_time** : pour la récupération de l’heure machine

* ***Entrée :*** *type d’heure*
* ***Sortie :*** *degré du cadran*

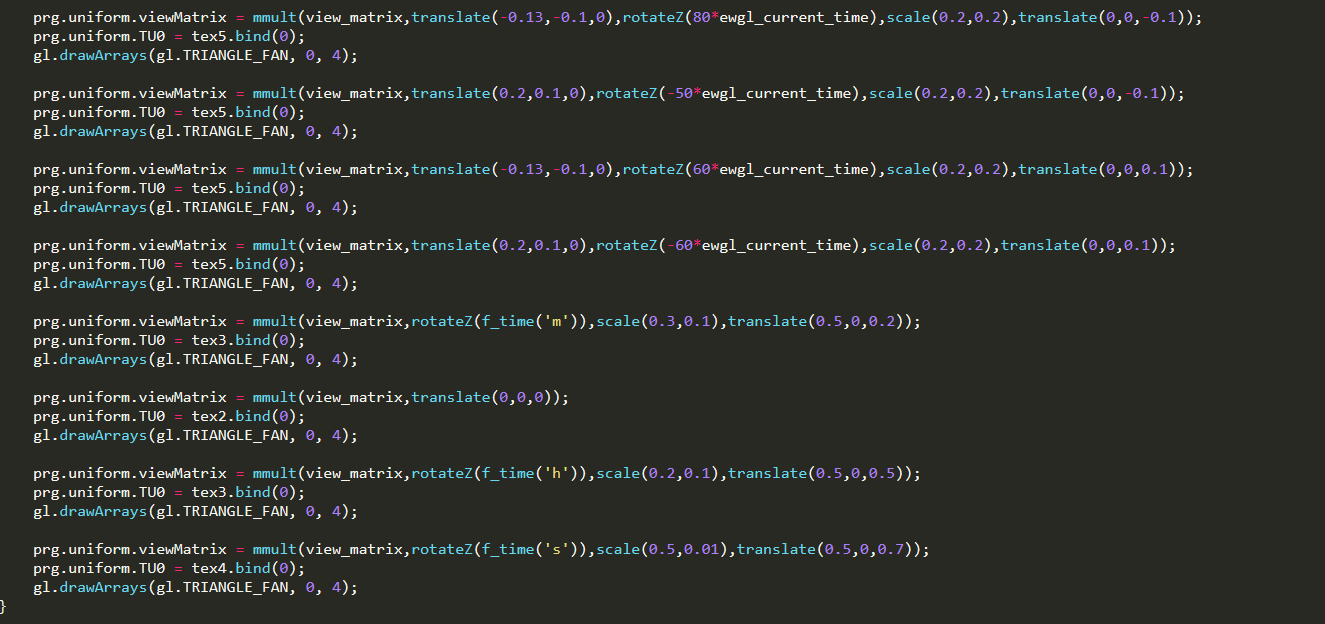
1. Edition de la **fonction** **launch\_3d** : pour le lancement ou l’exécution du code

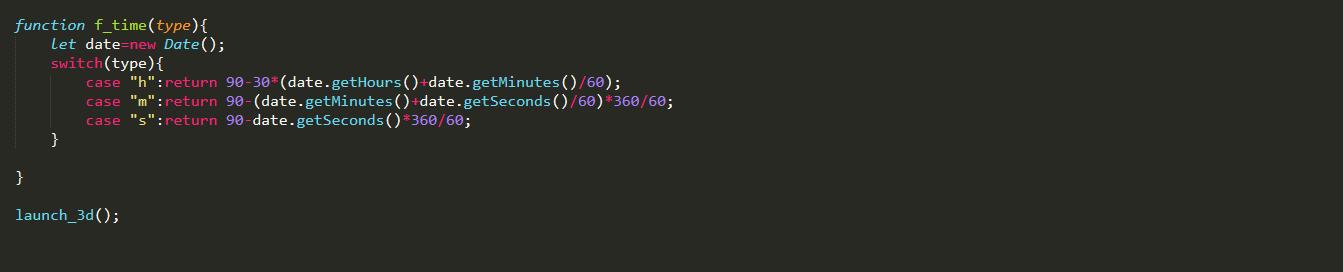
* ***Entrée :*** *--*
* ***Sortie :*** *--*

## Vue du code JavaScript imagé









## Vue du code à intégrer

var vertexShader=`#version 300 es

uniform mat4 monortho;

uniform mat4 projectionMatrix;

uniform mat4 viewMatrix;

in vec2 point\_in;

in vec2 tc;

out vec2 tc\_out;

void main(){

gl\_Position =projectionMatrix\*viewMatrix\*monortho\*vec4(point\_in,0,1.0);

tc\_out = tc;

}

`;

var fragmentShader=`#version 300 es

precision highp float;

in vec2 tc\_out;

out vec4 frag\_out;

uniform sampler2D TU0;

void main(){

vec2 tc2 = vec2(tc\_out);

vec3 col = texture(TU0,tc2).rgb;

frag\_out = vec4(col,0.0);

// gl\_FragColor = vec4(frag\_out); //else gl\_FragColor = vec4(0,1,0,1);

}

`;

var prg=null;

var vao=null;

var tex2=null;

var tex3=null;

var tex4=null;

var tex5=null;

function init\_wgl(){

prg=ShaderProgram(vertexShader,fragmentShader,"Texture");

let vbo=VBO([-1,-1,1,-1,1,1,-1,1],2);

let vbo\_tc=VBO([0,1, 1,1, 1,0, 0,0], 2);

vao=VAO([prg.in.point\_in,vbo],[prg.in.tc,vbo\_tc]);

scene\_camera.set\_scene\_radius(3);

scene\_camera.set\_scene\_center(Vec3(0,0,0));

tex2 = Texture2d([gl.TEXTURE\_WRAP\_S,gl.REPEAT],[gl.TEXTURE\_WRAP\_T,gl.REPEAT]);

tex3 = Texture2d([gl.TEXTURE\_WRAP\_S,gl.REPEAT],[gl.TEXTURE\_WRAP\_T,gl.REPEAT]);

tex4 = Texture2d([gl.TEXTURE\_WRAP\_S,gl.REPEAT],[gl.TEXTURE\_WRAP\_T,gl.REPEAT]);

tex5 = Texture2d([gl.TEXTURE\_WRAP\_S,gl.REPEAT],[gl.TEXTURE\_WRAP\_T,gl.REPEAT]);

Promise.all([tex2.load("Traitée2.png")],[tex3.load("TraitéeAiguille2.png")],

[tex4.load("aiguille2.png")],[tex5.load("roue.png")]).then(update\_wgl);

ewgl\_continuous\_update = true;

}

function draw\_wgl(){

gl.clearColor(0,0,0,0);

gl.enable(gl.DEPTH\_TEST);

gl.clear(gl.COLOR\_BUFFER\_BIT | gl.DEPTH\_BUFFER\_BIT);

gl.pixelStorei(gl.UNPACK\_PREMULTIPLY\_ALPHA\_WEBGL, true);

gl.blendFuncSeparate (gl.SRC\_ALPHA, gl.ONE\_MINUS\_SRC\_ALPHA, gl.ONE, gl.ONE\_MINUS\_SRC\_ALPHA);

const projection\_matrix = scene\_camera.get\_projection\_matrix();

const view\_matrix = scene\_camera.get\_view\_matrix();

prg.uniform.projectionMatrix=projection\_matrix;

prg.uniform.viewMatrix=view\_matrix;

prg.bind();

vao.bind();

prg.uniform.monortho=ortho2D;

prg.uniform.viewMatrix = mmult(view\_matrix,translate(-0.13,-0.1,0),rotateZ(80\*ewgl\_current\_time),scale(0.2,0.2),translate(0,0,-0.1));

prg.uniform.TU0 = tex5.bind(0);

gl.drawArrays(gl.TRIANGLE\_FAN, 0, 4);

prg.uniform.viewMatrix = mmult(view\_matrix,translate(0.2,0.1,0),rotateZ(-50\*ewgl\_current\_time),scale(0.2,0.2),translate(0,0,-0.1));

prg.uniform.TU0 = tex5.bind(0);

gl.drawArrays(gl.TRIANGLE\_FAN, 0, 4);

prg.uniform.viewMatrix = mmult(view\_matrix,translate(-0.13,-0.1,0),rotateZ(60\*ewgl\_current\_time),scale(0.2,0.2),translate(0,0,0.1));

prg.uniform.TU0 = tex5.bind(0);

gl.drawArrays(gl.TRIANGLE\_FAN, 0, 4);

prg.uniform.viewMatrix = mmult(view\_matrix,translate(0.2,0.1,0),rotateZ(-60\*ewgl\_current\_time),scale(0.2,0.2),translate(0,0,0.1));

prg.uniform.TU0 = tex5.bind(0);

gl.drawArrays(gl.TRIANGLE\_FAN, 0, 4);

prg.uniform.viewMatrix = mmult(view\_matrix,rotateZ(f\_time('m')),scale(0.3,0.1),translate(0.5,0,0.2));

prg.uniform.TU0 = tex3.bind(0);

gl.drawArrays(gl.TRIANGLE\_FAN, 0, 4);

prg.uniform.viewMatrix = mmult(view\_matrix,translate(0,0,0));

prg.uniform.TU0 = tex2.bind(0);

gl.drawArrays(gl.TRIANGLE\_FAN, 0, 4);

prg.uniform.viewMatrix = mmult(view\_matrix,rotateZ(f\_time('h')),scale(0.2,0.1),translate(0.5,0,0.5));

prg.uniform.TU0 = tex3.bind(0);

gl.drawArrays(gl.TRIANGLE\_FAN, 0, 4);

prg.uniform.viewMatrix = mmult(view\_matrix,rotateZ(f\_time('s')),scale(0.5,0.01),translate(0.5,0,0.7));

prg.uniform.TU0 = tex4.bind(0);

gl.drawArrays(gl.TRIANGLE\_FAN, 0, 4);

}

function f\_time(type){

let date=new Date();

switch(type){

case "h":return 90-30\*(date.getHours()+date.getMinutes()/60);

case "m":return 90-(date.getMinutes()+date.getSeconds()/60)\*360/60;

case "s":return 90-date.getSeconds()\*360/60;

}

}

launch\_3d();

# **ANNEXES**

