Developpez

Le Mag

Édition de avril - mai 2013.
Numéro 45.
Magazine en ligne gratuit.

Réalisation : Alexandre Pottiez
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

Sommaire

Éditorial

Ce mois-ci, un mini magazine avec un condensé d'informations sur toutes les technologies.

Profitez-en bien !

La rédaction

Article (X)HTML

Obtenir des images-types pour tester vos design

Nous allons voir comment utiliser Fake Images Please pour générer des images-types pour nos pages Web.

par Sébastien Germez
Page 11

Article 2D/3D/Jeux

Comment calculer la position et la normale dans le vertex shader avec OpenGL et Direct3D

Explication en détail de ces calculs.

par JeGX
Page 36

Les dernières news

Mise à jour 10.8.3 pour OS X Mountain Lion enfin disponible

Après de nombreuses bêta envoyées aux développeurs, Apple a enfin rendu publique la mise à jour de son dernier félin Mountain Lion.
Cette mise à jour était attendue depuis plusieurs mois par les utilisateurs afin de corriger les différents problèmes de stabilité et de sécurité.
Cette version apporte une mise à jour de Safari, mais surtout un meilleur support de Windows 8 et BootCamp.

Voici la liste des correctifs de cette mise à jour :

- la possibilité de convertir le code de cartes cadeaux iTunes dans le Mac App Store à l'aide de la caméra intégrée à votre Mac ;
- la prise en charge par Boot Camp de l'installation de Windows 8 ;
- la prise en charge par Boot Camp des Mac possédant un disque dur de 3 To ;
- la résolution d'un problème de fermeture inopinée des applications probablement à cause de l'URL d'un fichier ;
- la résolution d'un problème probablement à l'origine du blocage de Logic Pro lorsque certains modules sont utilisés ;
- la résolution d'un problème probablement à l'origine d'une mauvaise qualité audio sur les iMac 2011 ;
- l'intégration de Safari 6.0.3.

La mise à jour est directement téléchargeable depuis le Mac App Store : Lien 01.

N'hésitez pas à nous faire remonter vos remarques sur cette mise à jour.

Commentez la news d'Aurélien Gaymay en ligne : Lien 02

Tutoriels iOS

Cet article regroupe une série de tutoriels OpenGL ES 2.0 pour iPhone. En lisant ceux-ci, vous apprendrez à charger des modèles, appliquer un bump mapping, appliquer des effets de lumière et plus encore.

1. Introduction

Dans cette page, vous trouverez une liste de tutoriels sur OpenGL ES 2.0. Vous devez connaître le C++ et avoir de solides bases en OpenGL/GLSL (transformations, VBO, VAO, FBO et l'écriture de shaders). Tous les tutoriels utilisent le framework commun. Ce framework se trouve dans le dossier « engine » du fichier zip. Il est tiré de mon propre moteur appelé Virtual Vision : Lien 03.
Virtual Vision fut initialement écrit pour PC et a été porté plus tard sur iOS. J'ai décidé ensuite de faire quelque chose de plus intéressant avec le moteur. J'ai donc commencé à écrire ma première application pour iPhone qui est maintenant disponible sur l'Apple Store (Lien 04) nommée Camera Art Effects (Lien 05) qui permet d'appliquer des effets de posttraitement en temps réel sur l'image de la caméra. J'ai ensuite développé un jeu avec ce même moteur qui est aussi disponible sur l'Apple Store, Jumper Buggy (Lien 06), un jeu de simulation et d'action en 3D. Je vous recommande de lire cet article si vous souhaitez écrire votre propre moteur : Lien 07.

Restez à l'écoute des futurs tutoriels. Ci-dessous la feuille de route (par ordre de priorité) :

- mapping de l'environnement (FAIT) ;
- éclairage par pixel avancé (FAIT) ;
- mapping d'ombres (EN COURS) ;
- sprite 2D ;
- rendu différé (alors qu'il est toujours possible d'effectuer un rendu sur le G-buffer avec la version 2.0 d'OpenGL ES, ce qui nécessite plusieurs passes pour stocker la géométrie, les matériaux et les normales, cela pourrait être réuni dans un seul passage sur les appareils compatibles avec la future version 3.0 d'OpenGL ES [grâce au rendu sur cibles multiples uniquement. La prochaine version permettra aux pixel shaders d'écrire dans plusieurs tampons]). Les futurs jeux sur appareils mobiles devraient donc remplacer leurs modules de rendu par des plus modernes ;
- occlusion culling (uniquement avec les appareils compatibles avec la version 3.0 d'OpenGL ES).

Tous les tutoriels ont le même point d'entrée situé dans les fichiers Tutorial.h/Tutorial.cpp. Il y a deux fonctions dans cette classe, la première nommée « Deploy » qui charge les shaders et les ressources du disque pour le tutoriel et la seconde nommée Frame() qui affiche les images à l'écran à une fréquence de 60 images/seconde. Cette fonction est appelée par le viewControler de l'application.
Les tutoriels ne couvrent pas tout le code source, ils traitent seulement des parties principales, il vous reste à regarder en détail le code source. Le code est documenté et facile à parcourir si vous connaissez bien le C++.
Pour les questions ou rapports de bogues, retrouvez-moi ici :

- e-mail : **abdallah.dib(replace_with_at)virtual-vision.net**
- Twitter : **@abdallah_dib**

2. Téléchargement

Le framework et les exemples de code sont distribués sous **licence LGPL**.
Cliquez ici pour télécharger le code source pour le framework et les exemples de code : Lien 08.

3. Tutoriel 1. Importer des assets

Ce tutoriel montre comment utiliser Assimp pour importer et effectuer le rendu d'assets. Assimp est une bibliothèque open source capable de charger une variété de modèles 3D (3ds, obj, X, modèles quake et doom). En plus des positions des vertices, des normales, des coordonnées de texture, des tangentes, des os et leur poids (utilisés pour l'animation de personnages dans un prochain tutoriel), les matériaux et textures diffuses sont extraits du modèle et stockés dans un CMeshBuffer. Le CMeshBuffer contient tous les groupes et matériaux du mesh. Les groupes peuvent partager des matériaux. Lors du rendu du modèle, nous parcourons chaque groupe du buffer mesh, trouvons le matériau correspondant à chaque groupe, nous associons les propriétés du matériau au shader et dessinons finalement le groupe.

3.1. Code de la procédure

Nous utilisons la classe CAssimpMesh pour charger le mesh dans le tableau OpenGL comme expliqué dans la section précédente. Pour faire le rendu du CMeshBuffer, nous parcourons tous les groupes du buffer mesh, obtenons le matériau correspondant et envoyons les bonnes informations du matériau au shader et dessinons le groupe. Le code ci-dessous montre la façon dont le CMeshEntity effectue le rendu de son buffer mesh.

```
// Récupère le buffer mesh
CMeshBuffer* buffer = m_pMesh->GetMeshBuffer();

// Parcours de tous les groupes du buffer mesh
for(uint32 g = 0; g < buffer->GroupsCount(); ++g)
{
   // Rendu de chaque groupe
   CMeshGroup* grp = buffer->GroupAtIndex(g);

   // Récupération du matériau correspondant à chaque
groupe
```

```
    CMaterial* material = buffer->MaterialForGroup(grp);

    // Association des propriétés du matériau au shader (nous associons seulement la texture diffuse car il n'y a pas d'éclairage)
    if(material != NULL )
    {
        if( material->diffuseTexture != NULL && shader->texture0 != -1)
        {
            material->diffuseTexture->ActivateAndBind(GL_TEXTURE0);
            glUniform1i(shader->texture0, 0);
        }
    }

    // Activation du Vertex Array Object (VAO)
    grp->MapToGPU(0);

    // Dessin de chaque groupe
    glDrawElements(grp->GetDrawingMode(), grp->GetIndices().size(), GL_UNSIGNED_SHORT, 0);
}
```

4. Tutoriel 2. Placage de relief

Ce tutoriel montre comment effectuer un placage de relief simple et efficace sur iOS. Cette technique est largement utilisée dans les jeux pour ajouter des détails aux modèles sans avoir besoin d'un modèle complexe ou comportant un nombre élevé de polygones. Nous avons besoin d'une texture contenant les normales en plus de la diffuse. La position de l'éclairage est envoyée au shader dans le système de coordonnées espace monde puis convertie en espace tangent pour effectuer les calculs de lumière. Dans le pixel shader, nous récupérons la normale depuis la normal map pour chaque pixel, menant ainsi à un calcul non uniforme de la lumière sur la surface du modèle. Sans la normal map nous aurions un éclairage uniforme sur une surface plane. La capture d'écran ci-dessous montre le résultat de cette technique appliquée à une simple caisse. (Modèle 3D de turbosquid.com : Lien 09.)

4.1. Code de la procédure

Le shader a besoin d'une texture normale définissant le vecteur normal pour chaque pixel. Nous fournissons aussi la position de la lumière dans l'espace modèle, ce qui peut être fait en multipliant la position de la lumière par l'inverse de la matrice de transformation du modèle comme suit :

```
    // Cette partie sert à animer et déplacer la lumière sur une trajectoire circulaire
    static float y = 0.f;
    static float x = 0.f;
    x = -80.f * sinf(time);
    y = -80.f * cosf(time);
    // Conversion de la position de la lumière dans l'espace modèle, de sorte que le calcul de l'éclairage puisse être effectué dans l'espace modèle au sein du shader
    // Pour ce faire, nous multiplions la position de la lumière par l'inverse de la matrice de transformation du modèle
    vec4f lightPos = modelBumpMeshMat.inverse() * vec4f(x, y, 0, 1.f);

// Envoi de la position de la lumière au shader
    m_pShaderBump->SetUniform3fv("LightPosModel", 1, &lightPos.x);
```

Après, dans le vertex shader, nous construisons la matrice Tangent Bitangent Normal (TBN). Cette matrice effectue le passage du système de coordonnées modèle au système tangent. Nous multiplions ainsi le vecteur direction de la lumière par cette matrice pour le convertir dans l'espace tangent. En fait nous avons besoin d'un système de coordonnées commun pour effectuer le calcul de l'éclairage. Les normales et tangentes étant données dans l'espace tangent, nous devons faire passer le vecteur lumière dans le même système de coordonnées pour que cela ait un sens.

```
   // Création de la matrice d'espace tangent
    mat3 tangentSpaceXform = mat3(tangent, bitangent, normal);


    // Conversion du vecteur direction de la lumière depuis l'espace modèle vers l'espace tangent
    v_lightVec = lightDirection * tangentSpaceXform;
```

Dans le vexter shadex, nous créons la normale par pixel à partir de la texture normale, puis calculons l'intensité de la lumière en faisant le produit scalaire entre le vecteur de la normale et le vecteur direction de la lumière.

```
    // Normale par pixel
    vec3 normal = texture2D(textureBump, v_texCoord).rgb * 2.0 - 1.0;
    // L'intensité lumineuse est le produit scalaire du vecteur direction de la lumière par la normale par pixel
    float lightIntensity = dot(v_lightVec, normal);
```

5. Tutoriel 3. Animation de personnages (skinning) sur GPU

Ce tutoriel montre comment réaliser du skinning sur le GPU en utilisant le vertex du pipeline programmable pour

soulager le CPU. Nous utilisons Assimp pour importer le modèle animé. Un modèle animé est composé d'un maillage lié à un squelette articulé. Le squelette est composé d'une hiérarchie d'os où chaque os est représenté par une matrice de transformation. Chaque position de vertex est affectée par N os. Les indices des os influencés, ainsi que leur poids respectif, sont stockés dans les attributs du vertex. Dans le vertex shader, nous calculons la position finale du vertex en additionnant les combinaisons des différentes contributions des os. Par exemple : un vertex 'vert' est influencé par l'os 'bone0' avec le poids 'weight0' et l'os 'bone1' avec le poids 'weight1'. La position finale de 'vert' est calculée comme suit :

```
vert.finalpos = weight0 * (bone0.matrix * vert.pos) + weight1 * (bone1.matrix * vert.pos);
```

5.1. Code de la procédure

Dans le vertex shader, nous trouvons les matrices de transformation des os influencés pour le vertex actuel avec leur poids respectif comme suit :

```
 // skinningMatrix contient les matrices de transformation de tous les os. bones.x/y/z/w contient les indices des os qui affectent le vertex
    vec4 p0 = skinningMatrix[ int(bones.x) ] * position;
    vec4 p1 = skinningMatrix[ int(bones.y) ] * position;
    vec4 p2 = skinningMatrix[ int(bones.z) ] * position;
    vec4 p3 = skinningMatrix[ int(bones.w) ] * position;
```

La position finale du vertex est la somme des quatre vertices, chacun multiplié par le poids de l'os (la somme des poids d'un os doit être égale à 1) :

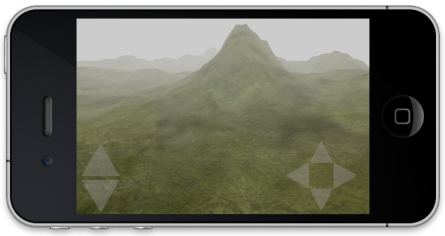
```
    // Position interpolée
    vec4 interpolatedPosition = p0 * weights.x + p1 * weights.y + p2 * weights.z + p3 * weights.w;
```

6. Tutoriel 4. Mouvement de la caméra quaternion

Ce tutoriel implémente une simple caméra quaternion que nous pouvons déplacer dans un monde 3D. La classe Camera n'est pas optimisée, donc si vous prévoyez de l'utiliser dans une application en temps réel, envisagez de faire quelques optimisations. J'ai ajouté un brouillard exponentiel au pixel shader. Pour une application en temps réel, il est recommandé de calculer le facteur du brouillard pour chaque vertex et d'écrire le résultat dans une variable varying qui peut être utilisée par le pixel shader. Le tutoriel emploie le multitexturing pour ajouter quelques détails au terrain.
Dans la fonction UpdateControls(const CControlPad& pad) du fichier Tutorial.cpp, pour faire pivoter la camera, nous obtenons premièrement la représentation actuelle du quaternion et la multiplions par la rotation quaternion adéquate selon le type de contrôle reçu (rotation gauche/droite ou haut/bas). Pour faire bouger la caméra, nous prenons sa position actuelle et le vecteur forward et nous changeons la position de la caméra suivant le vecteur forward.

```
const float step = 0.6f;// Mouvement et vitesse de la vue

    // Obtention de la caméra quaternion
    const Quatf& q = m_pCamera->GetQuaternionRepresentation();

    if(pad.keyUp) // Touche haut
    {
        Quatf rot = Quatf::fromAxisRot(vec3i(1, 0, 0), -step);
        Quatf qf = rot * q; // Cela va faire tourner la caméra quaternion autour de l'axe x (regarder vers le haut)
        m_pCamera->SetQuaternion(qf);// Définit la nouvelle caméra quaternion
    }

    if(pad.keyDown)
    {
        Quatf rot = Quatf::fromAxisRot(vec3i(1, 0, 0), step);
        Quatf qf  = rot * q  ; // Cela va faire tourner la caméra quaternion autour de l'axe x (regarder vers le bas)
        m_pCamera->SetQuaternion(qf);
    }

    if(pad.keyLeft)
    {
        Quatf rotX = Quatf::fromAxisRot(vec3i(0, 0, 1), -step);
        Quatf qf  = q * rotX;// Rotation de la caméra autour de l'axe des z, axe vertical
        m_pCamera->SetQuaternion(qf);
    }

    if(pad.keyRight)
    {
        Quatf rotX = Quatf::fromAxisRot(vec3i(0, 0, 1), step);
        Quatf qf  = q * rotX;// Rotation de la caméra autour de l'axe des z, axe vertical
        m_pCamera->SetQuaternion(qf);
    }

    if(pad.keyMoveUp)
    {
        vec3f pos = m_pCamera->GetPosition();// Position de la caméra
        vec3f forward = m_pCamera->GetForwardVector();// Vecteur forward de la caméra
        pos += forward * step;// Déplacement de la caméra selon le vecteur forward
        m_pCamera->SetPosition(pos);// Application de la nouvelle position

    }

    else if(pad.keyMoveDown)
    {
```

```
        vec3f pos = m_pCamera->GetPosition();// Position de la caméra
        vec3f forward = m_pCamera->GetForwardVector();// Vecteur forward de la caméra
        pos -= forward * step;// Déplacement de la caméra selon le vecteur forward
        m_pCamera->SetPosition(pos);// Application de la nouvelle position
    }
```



7. Tutoriel 5. SkyBox

Ce tutoriel montre comment charger et afficher une skybox en utilisant une texture cube map dans OpenGL. La texture cube map est composée de six textures 2D, chacune représentant une face du cube map. Pour allouer les six textures, nous appelons la fonction glTexImage2D six fois pour chaque face. Le paramètre *target* de la fonction renseignera le programme sur la face à allouer parmi les six. Par exemple, pour transmettre les données depuis la mémoire centrale à la texture cube map sur l'axe X positif :

```
glTexImage2D (GL_TEXTURE_CUBE_MAP_POSITIVE_X,
                          mip_level,
                          internal_format,
                          width,
                          height,
                          border,
                          host_data_format,
                          host_data_type,
                          host_data_xp)
```

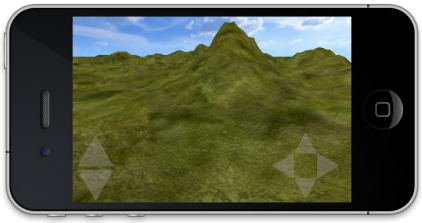
Où le premier paramètre indique à OpenGL que nous voulons envoyer les données à l'axe X positif du cube map. Les deux derniers paramètres spécifient le type et l'adresse mémoire des données de l'hôte à transférer au GPU.
Le CCacheResourceManager situé dans le framework charge et retourne la texture cube map prête à l'emploi. Notez que les six faces de la texture cube map doivent être nommées ainsi
(Nom de base + underscore + vecteur direction) :

- skyboxName_xn.jpg (ou png) ;
- skyboxName_xp.jpg ;
- skyboxName_yn.jpg ;
- skyboxName_yp.jpg ;
- skyboxName_zn.jpg ;
- skyboxName_zp.jpg.

Pour charger la skybox, donnez seulement le nom de base au gestionnaire de ressources :

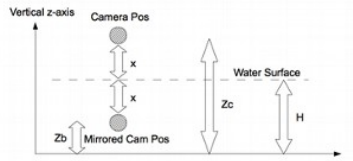
```
CTextureCubeMap *textureCubeMap = crm.LoadTextureCube("skyboxName.jpg");
```

La fonction LoadTextureCube charge et transfère chaque image à la face correspondante de la texture cube map. La fonction retourne un objet texture cube map qui peut être utilisé et échantillonné par le pixel shader.



8. Tutoriel 6. Réflexion de l'eau

Ce tutoriel montre comment on peut implémenter la réflexion de surface en temps réel. La partie la plus importante est le rendu de la texture de réflexion basé sur la position de la caméra. Nous utilisons un Frame Buffer Object (FBO) pour effectuer un rendu offscreen du reflet de la scène dans une texture. Cette texture est passée à un shader pour simuler la surface de l'eau.



Pour trouver la position de la caméra miroir, nous avons à partir du schéma ci-dessus :

```
Zc = Zb + 2 * x Eq 1)
where x = H - Zb
```

En substituant x dans l'équation (Eq 1), la hauteur finale de la position du reflet est égale à :

```
Zb = 2H - Zc
```

8.1. Code de la procédure

Dans CwaterEntity, nous créons une cible de rendu offscreen avec une texture couleur et un buffer de rendu de profondeur. La fonction PrepareReflectionPass active le frame buffer object pour le rendu offscreen, et nous retrouvons les positions de la caméra miroir et de la cible comme expliqué avant :

```
    // Activation du FBO pour le rendu offscreen
    m_pOffscreenRT->Enable();

    // Envoi de la hauteur de l'eau au shader et activation du rejet, pour couper tous les objets sous l'eau
    shader->SetUniform1i("enableDiscardScene",1) ;
    shader->SetUniform1f("waterHeight", m_fWaterHeight);
```

```
    // Effacement du contexte
    glClear(GL_DEPTH_BUFFER_BIT |
GL_COLOR_BUFFER_BIT);

    vec3f camPos = camera->GetPosition(); //
Position de la caméra
    vec3f camTarget = camPos + camera-
>GetForwardVector();// Cible

    // Si la caméra est au-dessus de l'eau,
inverser sa position
    if(camPos.z >= m_fWaterHeight)
    {

        // Position de la caméra miroir
        camPos.z = 2.0f * m_fWaterHeight -
camPos.z;
        camTarget.z = 2.0f * m_fWaterHeight -
camTarget.z;

        vec3f forwardVector = camTarget - camPos;
// Trouver le vecteur forward à partir des
nouvelles positions
        vec3f sideVector = camera-
>GetRigthVector(); // Obtention du vecteur right
        vec3f reflectionCamUp =
sideVector.crossProduct(forwardVector);
//Obtention du vecteur up

        // Mise en place de la matrice vue miroir
        mat4f m;
        m = mat4f::createLookAt(camPos,
camTarget, reflectionCamUp);
        m_mLookAt =  m_mMirror * m;
    }
```

Maintenant que nous avons la texture de réflexion, la prochaine étape est de l'utiliser pour effectuer le rendu de la surface de l'eau. La surface de l'eau est composée d'un simple rectangle auquel nous appliquons la texture de la réflexion. Pour troubler la texture de la réflexion, nous utilisons une texture normale sur différentes directions u,v. Dans le pixel shader, nous utilisons deux coordonnées de texture avec un facteur de répétition pour créer la texture normale. Les coordonnées de la texture résultante sont combinées avec les coordonnées du fragment pour créer la texture miroir :

```
   // Définition du nombre de tuiles de la
surface d'eau (répétition de la texture)
    const float tile_factor = 20.0;
    const float noise_factor = 0.03;

    vec2 texCoordNormal0 = v_texCoord *
tile_factor;
    texCoordNormal0 += time ;

    vec2 texCoordNormal1 = v_texCoord *
tile_factor;
    texCoordNormal1.s -= time ;
    texCoordNormal1.t += time ;

// Coordonnées tex utilisées pour troubler le
vecteur texCoordReflection
    vec3 normal0 = texture2D(normal,
texCoordNormal0).rgb * 2.0 - 1.0;
    vec3 normal1 = texture2D(normal,
texCoordNormal1).rgb * 2.0 - 1.0;
    vec3 normal = normalize(normal0 + normal1);

    // Ajustement des coordonnées de la texture
selon la taille de l'écran
    vec2 texCoordReflection = vec2(gl_FragCoord.x
* screenWidth, gl_FragCoord.y * screenHeight);

    // Les coordonnées de la texture finale sont
troublées par la normale multipliée par un
facteur de bruit
    vec2 texCoordFinal = texCoordReflection.xy +
noise_factor * normal.xy;

    gl_FragColor =  texture2D( texture0,
texCoordFinal);
```

Dans ce tutoriel nous ne traitons pas le cas où la caméra se situe en dessous de la surface, situation dans laquelle nous avons besoin d'une passe supplémentaire pour faire la capture de la texture de réfraction. En réalité, sous l'eau nous pouvons toujours voir le monde réfracté, cette texture additionnelle est utilisée pour simuler l'effet de réfraction. La couleur finale est obtenue par le mélange des textures de réflexion et réfraction selon la formule de Fresnel.
Le but du tutoriel était de présenter une simulation simple et rapide de réflexion d'eau sur mobile. Vous pourrez trouver des simulations de surfaces d'eau plus avancées sur le net : Lien 10.
Pour optimiser, nous pouvons réduire la taille de la texture de réflexion en réduisant la taille du frame buffer object de sortie dans le CwaterEntity. La taille actuelle est fixée à 256, essayez différentes valeurs pour entrevoir l'impact sur la qualité/performance.

```
m_pOffscreenRT->Initialize(kCONTEXT_COLOR |
kCONTEXT_DEPTH,  256);
```

Une autre optimisation consiste à remplacer l'opération de rejet du pixel shader lors du rendu du terrain. Cette opération est utilisée pour ne pas prendre en compte les pixels situés au-dessus de l'eau lors de la capture de la texture de réflexion. Ce qui pourrait être effectué en ajustant les plans de coupe de la matrice de projection.
Voir le SDK de Imagination Technologies (exemple de réflexion d'eau) : Lien 11.

9. Tutoriel 7. Éclairage avec relief

Ce tutoriel suppose que vous connaissiez les rudiments d'éclairage en OpenGL, ce que lumière ambiante, diffuse et spéculaire veulent dire et les mathématiques derrière. Il ne couvre pas la théorie de l'éclairage Phong, ni comment les facteurs ambiant, diffus et spéculaire sont calculés. Il y a un grand nombre d'articles et de documentations sur le Net, suivez ce lien pour plus de détails : Lien 12.
Ce tutoriel montre comment réaliser un éclairage par point sur un shader avec les reliefs. Le bump mapping ajoute des détails à la scène en perturbant les normales de la surface des objets, que l'on utilise ensuite dans le pixel shader pour le calcul de l'éclairage. Il en résulte une surface irrégulière. La capture d'écran ci-dessous montre la même scène

rendue avec et sans bumps (dans l'exemple de code fourni, vous pouvez basculer entre les deux scènes en utilisant le bouton switch).

9.1. Code de la procédure

Du point de vue CPU, les positions de la lumière et de la caméra sont envoyées au shader dans le système de coordonnées monde.
Tout d'abord, nous obtenons l'inverse de la matrice modèle de l'objet comme suit :

```
    //Obtention de la matrice modèle
    mat4f &modelBumpMeshMat =  mesh-
>GetTransfromationMatrix();
    mat4f modelInv = modelBumpMeshMat.inverse();
```

Puis nous multiplions la position de la lumière par l'inverse de la matrice modèle pour obtenir sa position dans l'espace modèle :

```
    //Position de la lumière dans l'espace monde
    vec4f lightPosition = modelInv * lightPos;

    // Envoi des uniformes au shader
    m_pActiveShader-
>SetUniform3fv("lightPosModel", 1,
&lightPosition.x);
```

En ce qui concerne la position de la caméra, elle se situe toujours à l'origine (0, 0, 0). La matrice de vue transforme chaque objet depuis l'espace monde vers l'espace caméra, donc si nous multiplions la position de la caméra située à (0, 0, 0, 1) par l'inverse de la matrice de vue, nous obtenons la position de la caméra dans l'espace monde. Puis nous multiplions cette position par l'inverse de la matrice modèle pour obtenir la position de la caméra dans l'espace objet :

```
    vec4f cameraPos = inv( modelMatrix ) *
(inv(ViewMatrix) * vec4(0, 0, 0, 1)
```

Pour une expression compacte :

```
vec4f cameraPos= modelInv * m_pCamera-
>GetPosition();
```

m_pCamera->GetPosition() calcule l'inverse de la matrice de vue, que nous multiplions par le vecteur homogène vec4f(0, 0, 0, 1.f). Notez qu'effectuer la multiplication d'une matrice « M » 4x4 par un vecteur « v » homogène 4x1 extrait la partie translation de la matrice (M[12] M[13] M[14]).
Maintenant que nous avons la position de la caméra dans l'espace monde, nous l'envoyons au shader.

```
m_pActiveShader->SetUniform3fv("camPosModel", 1,
&cameraPos.x);
```

De plus, nous devons envoyer les paramètres de la lumière et les propriétés du matériau définis par les vecteurs ambiant, diffus et spéculaire.
Pour les paramètres de la lumière :

```
    m_pActiveShader-
>SetUniform4fv("lightColorAmbient", 1,
&lightAmbient.x);
    m_pActiveShader-
>SetUniform4fv("lightColorDiffuse", 1,
&lightDiffuse.x);
    m_pActiveShader-
>SetUniform4fv("lightColorSpecular", 1,
&lightSpecular.x);
```

Pour les propriétés du matériau, l'envoi se fait dans la fonction Render de l'objet CMeshEntity comme suit :

```
            // Envoi du paramètre ambiant
            if(shader->matColorAmbient != -1)
            {
                glUniform4fv(shader-
>matColorAmbient, 1, material->ambient);
            }

            // Envoi du paramètre diffus
            if(shader->matColorDiffuse != -1)
            {
                glUniform4fv(shader-
>matColorDiffuse, 1, material->diffuse);
            }

            // Envoi du paramètre spéculaire
            if(shader->matColorSpecular != -1)
            {
                glUniform4fv(shader-
>matColorSpecular, 1, material->specular);
            }

             // Envoi du paramètre de brillance
            if(shader->matShininess != -1)
            {
                glUniform1f(shader->matShininess,
material->shininess);
            }
```

Passons ensuite au vertex shader. Nous commençons par construire la matrice tangente utilisée pour passer tous les vecteurs dans le même système de coordonnées tangent (espace tangent). Le vecteur direction de la lumière et demi-angle sont convertis vers l'espace tangent.

```
    // Calcul du vecteur bitangent  (ceci peut
être fait sur le CPU)
    vec3 bitangent = cross(normal, tangent);
    // Création de la matrice de l'espace tangent
    mat3 tangentSpace = mat3(tangent, bitangent,
normal);

    // Obtention du vecteur direction de la
lumière du vertex actuel
    v_lightVector =  lightPosModel - position.xyz
;
    // Conversion du vecteur direction de la
lumière vers l'espace tangent
    v_lightVector = v_lightVector * tangentSpace;
    // Normalisation
    v_lightVector = normalize(v_lightVector);

    // Obtention de la direction de la caméra du
vertex actuel
    v_halfVector = camPosModel - position.xyz ;
    // Conversion vers l'espace tangent
    v_halfVector = v_halfVector * tangentSpace;
    // Normalisation
    v_halfVector = normalize(v_halfVector);
    // Calcul du vecteur demi-angle
    v_halfVector = (v_halfVector + v_lightVector)
```

```
/2.0;
    // Normalisation
    v_halfVector = normalize(v_halfVector) ;
```

Dans le pixel shader, nous créons une normale par pixel, perturbée à partir de la normal map. Puis nous calculons la couleur finale du pixel par la somme des contributions ambiante, diffuse et spéculaire de la lumière.

```
    // Obtention de la couleur diffuse par vertex
    vec4 color  = texture2D(texture0, texCoord) ;

    // Obtention de la normale par pixel
    vec3 bump = texture2D(textureBump,
texCoord).rgb * 2.0 - 1.0;
    // Inversion de la normale pour les faces
arrière (si le face culling est activé, cette
étape peut être supprimée)
    if (!gl_FrontFacing)
        bump = - bump;

    // Calcul de la contribution de la lumière
    // 1- lamber ou facteur diffus
    float lamber = max(0.0,
dot(normalize(v_lightVector), bump) );

    //2- facteur spéculaire
    float specular = 0.0;
    if (dot(bump, v_lightVector) < 0.0)
        specular = 0.0;
    else
        specular = max(0.0,
pow(dot(normalize(v_halfVector), bump),
matShininess)) ;

    // Obtention des couleurs finales ambiantes,
diffuses et spéculaires
    vec4 finalAmbientContrib  = lightColorAmbient
* color /** matColorAmbient.xyz*/;
    vec4 finalDiffuseContrib  = lightColorDiffuse
* color *  lamber * matColorDiffuse;
    vec4 finalSpecularContrib =
lightColorSpecular  * specular *
matColorSpecular;

    // La couleur finale est la somme des
composantes ambiante, diffuse et spéculaire
    gl_FragColor = finalAmbientContrib +
(finalDiffuseContrib + finalSpecularContrib)  ;
```

Scène éclairée sans relief

Scène éclairée avec relief

10. Tutoriel 8. Réflexion par cube mapping

Aussi connu sous le nom d'environment mapping, ce tutoriel montre comment réaliser la réflexion d'un environnement sur un objet. Cette technique est utilisée pour simuler une surface réfléchissante dans une scène. Nous utilisons une texture cube map comme texture de réflexion. Cette texture est découpée en fragments dans le pixel shader pour simuler cette réflexion sur l'objet.

10.1. Code de la procédure

Pour commencer, nous chargeons la texture cube map qui sera utilisée comme texture de réflexion :

```
CTextureCubeMap *textureCubeMap =
crm.LoadTextureCube("cm.jpg");
```

Puis nous envoyons la position de la caméra au shader dans l'espace modèle, position qui sera utilisée pour déterminer le vecteur de réflexion sur la surface normale.

```
/* Pour trouver la position de la caméra dans
l'espace modèle :

la caméra (ou œil) est située en v0(0, 0, 0),
nous multiplions v0 par l'inverse de la matrice
de vue (V) puis par l'inverse de la matrice
modèle (M). camPos = inv(M) * inv(V) *
vec4(0, 0, 0, 1); */

vec4f camera_pos = ( m_pCamera->GetViewMatrix() *
model ).inverse() * vec4f(0, 0, 0, 1.f);

// Envoi de la position de la caméra au shader
m_pShaderMesh->SetUniform4fv("camPosModel", 1,
&camera_pos.x);
```

Le vecteur de réflexion calculé dans le vexter shader doit être multiplié par le vecteur de rotation de l'objet pour obtenir une réflexion correcte. Donc nous devons envoyer la partie rotation de la matrice modèle au vexter shader pour reprojeter le vecteur réflexion dans l'espace monde.

```
// Extraction de la matrice de rotation
mat3f model_mat3 = mat3f(model);

// Envoi au shader
m_pShaderMesh-
>SetUniformMatrix3x3fv("matModelToWorld", 1,
GL_FALSE, &model_mat3[0]);
```

En ce qui concerne le GPU, nous calculons au sein du vertex shader la direction de la caméra dans l'espace modèle en soustrayant la position de la caméra à la position du vertex.

```
// Obtention de la direction de la caméra dans
l'espace modèle
vec4 eyeDir = normalize(position - camPosModel);
```

Nous calculons ensuite le vecteur de réflexion :

```
reflect(eyeDir.xyz, normal);
```

Nous avons besoin de ramener le vecteur de réflexion vers l'espace monde :

```
 /* Retrouver le vecteur réflexion entre la
direction de la caméra et la normale du vertex,
multiplier ce vecteur par la matrice de rotation
du modèle (conversion vers l'espace monde) */
v_reflectionVector = matModelToWorld *
reflect(eyeDir.xyz, normal);
```

Finalement, au sein du pixel shader, nous échantillonnons la texture cube map en utilisant le vecteur réflexion interpolé renvoyé par le vertex shader.

```
gl_FragColor = textureCube(textureCubeMap,
v_reflectionVector);
```

La capture d'écran ci-dessous montre la réflexion sur une théière. Pour effectuer le rendu du skybox, décommentez la ligne suivante du fichier Tutorial.cpp :

```
//#define RENDER_SKYBOX
```

Retrouvez l'article d'Abdallah Dib traduit par Jérôme Marsaguet en ligne : Lien 13

Obtenir des images-types pour tester vos design

Dans cet article, nous allons voir comment utiliser Fake Images Please (Lien 14) pour générer des images-types pour nos pages Web.

1. Traduction

Cet article est la traduction la plus fidèle possible de l'article original de Paul Underwood, Fake Images Please : Lien 15.

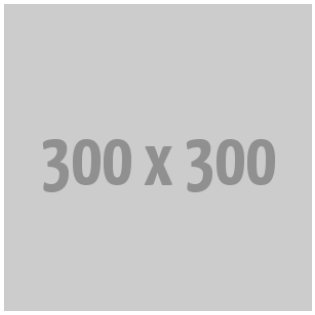
2. Introduction

Lorsque vous créez un nouveau site Web, vous passez obligatoirement par quelques étapes avant de finaliser le design.
Tout d'abord, obtenir les besoins (cahier des charges par exemple) du client ; à la suite de quoi vous pouvez créer une maquette que vous pourrez montrer au client afin d'être sûr que votre production correspond bien à ses attentes.
Grâce à cette maquette, vous aurez un aperçu visuel basique de ce à quoi doit ressembler votre site Web, mais ce n'est pas encore très abouti. C'est à partir de là que vous commencez à créer les prototypes de vos pages. Et pour ce faire vous avez besoin de contenu-type (texte) et d'images-types.
Pour le texte rien de plus simple : il suffit d'insérer le très connu **Lorem Ipsum**. Pour les images en revanche le travail est plus fastidieux. Vous devez les créer de toutes pièces et si vous avez besoin de modifier leurs tailles par la suite vous êtes obligés de repasser par l'étape d'édition via un éditeur d'images quelconque.
Une solution possible est d'utiliser un service de « placeholder » comme Fake Images Please (Lien 14). Ce service vous permet de créer des images-types de la taille que vous souhaitez en utilisant leur API, ce qui signifie que vous pouvez simplement utiliser ces images en les insérant en HTML.

3. Insérer une image de 300x300

La méthode la plus simple pour créer une image est de lui passer un paramètre pour la taille. Cet exemple va créer une image de 300 pixels de large et 300 pixels de haut.

```
<img src="http://fakeimg.pl/300/">
```



4. Insérer une image de 250x100

Vous pouvez également passer en paramètre deux tailles. Votre image sera alors créée avec la taille suivante : *largeurxhauteur*.

```
<img src="http://fakeimg.pl/250x100/">
```

5. Insérer une image de 250x100 avec un arrière-plan rouge

Vous pouvez aussi préciser la couleur de l'arrière-plan. Pour ce faire il vous suffit d'ajouter un deuxième paramètre dans l'URL de votre image.

```
<img src="http://fakeimg.pl/250x100/ff0000/">
```

6. Insérer une image de 350x200 avec un arrière-plan rouge et un texte noir

Vous pouvez passer un troisième paramètre afin de préciser une couleur pour le texte à l'intérieur de l'image. Pour créer une image de 350x200, avec un arrière-plan rouge et un texte noir, utilisez le code suivant :

```
<img src="http://fakeimg.pl/350x200/ff0000/000">
```

7. Insérer une image de 350x200 avec un texte personnalisé

Vous pouvez modifier le texte contenu dans l'image en passant en paramètre une chaîne de caractères « text » dont la valeur sera le nouveau contenu texte de votre image.

```
<img src="http://fakeimg.pl/350x200/?
text=Paulund">
```

8. Insérer une image de 350x200 avec une police personnalisée pour le texte

Non seulement vous pouvez modifier le texte contenu dans l'image, mais vous pouvez également changer la police utilisée pour ce texte. Pour cela il suffit d'ajouter le paramètre « font » dont la valeur sera une police Google fonts.

```
<img src="http://fakeimg.pl/350x200/?
text=World&font=lobster">
```

9. Conclusion

Nous avons vu dans cet article comment utiliser très simplement des images-types pour nos prototypes de pages Web. Le tout sans avoir besoin d'un éditeur d'images. Cette méthode permet un gain de temps considérable !

Retrouvez l'article de Paul Underwood traduit par Sébastien Germez en ligne : Lien 16

[Android] Avancée du support d'Android dans Qt 5.1

Le portage de Qt sur Android n'est pas neuf : en janvier 2010, un système graphique fonctionnel était rendu public par BogDan Vatra.

Lien 17

Un peu plus d'un an plus tard, en février 2011, Necessitas sortait au grand jour, avec la première version de Qt utilisable sur Android : Lien 18. Fin 2012, le projet Necessitas quittait le giron KDE pour passer au Qt Project, avec intégration dans Qt 5 prévue : Lien 19.

Début 2013, une branche de développement était créée sur le dépôt Git de Qt, qui vient d'être intégrée au tronc commun : le support d'Android sera bel et bien disponible avec Qt 5.1.

Cette première version s'oriente plus vers les développeurs, pour qu'ils lancent et testent de manière aussi facile que possible leurs applications sur Android. Un support plus finalisé sera disponible pour Qt 5.2, avec plus de possibilités pour le déploiement et le support de plus d'API Android.

En pratique ? Une première démo Qt 5 sur un Nexus 4, un Asus Transformer Pad TF300T et un Nexus 7 ; elle montre un grand nombre d'effets graphiques de Qt Quick 2, dont les *shaders*, les particules, etc., à soixante images par seconde.

Qt 5 Cinematic Experience demo on Android : Lien 20

L'API de lecteur multimédia de Qt Multimedia est aussi supportée en QML, ici avec un *shader* par-dessus.

Qt 5 media player running on Android : Lien 21

Cette version de Qt supporte aussi le multitouch, notamment utile pour du dessin à la main. Cette vidéo montre aussi les menus natifs.

Qt 5 multitouch demo running on Android : Lien 22

Qt 5 utilise l'API Android en version 10 (soit Android 2.3.3), c'est-à-dire qu'il est utilisable sur un très grand nombre de périphériques (selon certaines statistiques, cela concerne à peu près tous les périphériques : Lien 23). Ici, sur un Huawei Y100 :

Qt 5 multitouch image viewer running on Android : Lien 24

Qu'est-ce qui est actuellement supporté ? Les applications Qt Widgets et Qt Quick, le lecteur multimédia QML de Qt Multimedia, les capteurs les plus utilisés dans Qt Sensors, les fonctionnalités multiplateformes de Qt (comme les contrôles Qt Quick), le développement et le déploiement d'applications dans Qt Creator 2.7.

D'autres choses sont prévues, comme la distribution de Qt par Ministro (afin de partager les bibliothèques dynamiques entre les applications), ce qui sera la méthode la plus recommandée pour Qt 5.1. Tous les détails sont disponibles sur le wiki du projet (Lien 25).

Source : Lien 26.

Commentez la news de Thibaut Cuvelier en ligne : Lien 27.

Sortie de Qt Creator 2.7.0

Peu de temps après la sortie de Qt Creator 2.6.2 (Lien 28), voici la préversion de la prochaine version majeure de l'EDI C++, qui donne quelques impressions sur les nouvelles fonctionnalités et améliorations apportées. La version finale est prévue pour la fin mars.

Avec l'*open governance*, pas moins de mille trois cents commits ont été réalisés par soixante-trois développeurs, ce qui montre la vivacité de l'environnement.

Nouveautés côté C++

Le support de C++11 s'améliore encore, mais n'est toujours pas parfait. Les mots-clés alignof, alignas et noexcept sont maintenant gérés, ainsi que >> dans les définitions de templates. Le support des lambdas a aussi été amélioré. La plus grosse amélioration, aux yeux des développeurs, est le support de l'initialisation uniforme, avec des accolades.

Lorsque le contexte ne fournit pas assez d'éléments, l'environnement passe maintenant par défaut en mode C++11 et non plus en mode C++03.

Il faut aussi noter que ce n'est pas dans cette version que l'on verra l'utilisation de Clang pour l'analyse syntaxique du code, pour des raisons de performances.

Nouveautés côté QML

Avec la sortie de Qt 5, il était fort probable que le support de Qt Quick 2 allait recevoir beaucoup d'attention – et cela a été le cas. Le designer graphique devrait à présent fonctionner bien mieux avec cette version de l'environnement de programmation déclarative d'interfaces graphiques.

Cependant, les binaires actuellement disponibles sont compilés avec Qt 4.8, ce qui signifie que le rendu ne peut pas encore utiliser cette version de Qt Quick. Il faudra, pour en profiter, soit compiler soi-même Qt Creator, soit

attendre la sortie de Qt 5.1 (cette version de Qt devrait également inclure les composants Qt Quick en tant que module essentiel, ce qui ne manquera pas de donner un plus grand intérêt à la plateforme).

Nouveautés côté kits

La grande nouveauté de Qt Creator 2.6 était la présence des kits, une solution très flexible pour gérer les chaînes de compilation utilisées (ce qui est très utile dans un contexte multiplateforme avec de la compilation croisée, de plus en plus fréquente avec la montée en puissance des plateformes mobiles).

Il sera maintenant possible d'utiliser certaines chaînes personnalisées sans devoir créer son propre plug-in.

Cette flexibilité était venue avec quelques zones plus sombres, une interface pas aussi claire qu'elle aurait dû. Désormais, une grande partie de ces défauts de jeunesse est corrigée, l'avis de la communauté sera d'une très grande aide pour peaufiner encore cette partie.

Divers

Le probable prochain système de compilation, QBS, qui devrait remplacer QMake cette année (vers l'été ?), est supporté de manière expérimentale par l'EDI.
Des templates pour la création d'applications BlackBerry 10 sont disponibles.

Les chaînes à traduire seront gelées la semaine prochaine (jeudi), le vrai travail de traduction pourra commencer à ce moment.
Une première version Release Candidate est attendue deux semaines plus tard.

Source : Lien 29
Téléchargement des binaires : Lien 30

Commentez la news de Thibaut Cuvelier en ligne : Lien 31.

[iOS] Avancée du support d'iOS dans Qt 5.1

Tout de go, Digia annonce que Qt 5.2 supportera iOS, soit vers la fin 2013. Cependant, tous les détails ne sont pas encore décidés (comme ce qui concerne les restrictions sur l'App Store ou le support de parties plus anciennes du code de Qt). Qt 5.1, en tout cas, contiendra une première version de ce support, le code a été intégré vendredi dernier.

Tout le développement et le déploiement passent par Xcode. Actuellement, la manière de procéder est la création d'un fichier de projet .pro, qui est exporté en un fichier de projet Xcode par QMake (et réexporté à chaque modification), ce dernier étant alors utilisé dans Xcode. Toute l'édition du code peut évidemment se faire en dehors de cet EDI.

Le style Mac de Qt pour plateformes desktop utilise l'API HITheme d'OS X pour l'affichage d'éléments natifs à l'écran. Cependant, cette API n'a pas d'équivalent sur iOS, créer une classe QiOSStyle comme l'actuelle QMacStyle n'est donc pas possible. Les styles multiplateformes tels que le nouveau Fusion sont cependant disponibles. Tous les efforts sur le style des applications seront portés sur les contrôles intégrés dans Qt Quick 2.

Apple limite fortement les possibilités des applications sur iOS. Notamment, il n'est pas possible d'utiliser un compilateur de type JIT, tel que celui utilisé dans V8 – ce qui signifie qu'il n'y aura pas de Qt Quick 2 sur iOS pour le moment. Ce problème est bien connu et une solution est à l'étude.

Actuellement, déjà quelques modules fonctionnent correctement : les widgets, QGraphicsView, Qt Quick 1, OpenGL, les éléments tactiles et d'orientation.

Pour tester cette première ébauche, il est nécessaire d'avoir Xcode installé (avec les certificats et le périphérique configurés). Ensuite, il faut cloner qtbase

```
git clone git://gitorious.org/qt/qtbase.git qtbase-ios
cd qtbase-ios/
```

et le compiler (soit pour le périphérique, soit pour le simulateur).

```
./configure -xplatform unsupported/macx-ios-clang -developer-build -nomake examples -nomake tests -release [-sdk iphonesimulator]
make
```

Récupérer une démo et l'ouvrir dans Xcode :

```
git clone git://github.com/msorvig/qt-ios-demo.git
cd qt-ios-demo
../qtbase-ios/bin/qmake
open qt-ios-demo.xcodeproj
```

Commentez la news de Thibaut Cuvelier en ligne : Lien 32.

« apprendre-a-taper-au-clavier » - Comment se former à écrire rapidement sur le clavier

Mon intention n'est pas de vous fournir la « formule magique », ni même la meilleure méthodologie, mais de vous proposer des outils qui vous permettront de progresser dans ce domaine et de mettre en valeur cette formation.

1. Introduction

Toujours en quête d'optimisation de son temps, que l'on soit blogueur, « forumeur », programmeur, etc., on se doit d'apprendre à taper avec ses dix doigts, tout cela, bien entendu, sans regarder son clavier.
Certains diront : facile ? Et d'autres : non… et certains penseront savoir !
Afin d'évaluer votre maîtrise, vous pouvez vous tester sur le site : 10-fast-fingers.com (Lien 33), qui va vous indiquer votre « nombre de mots tapés par minute ».
La disposition des touches varie en fonction des pays, voici une liste des types de claviers standards qui existent : « Disposition des touches des claviers informatiques » (Lien 34).

2. Pourquoi ?

En effet en maîtrisant ces méthodes vous pourrez désormais :

- lire ce que vous écrivez au moment de la frappe et cela est d'autant plus utile quand vous discutez sur un forum, sur un chat, etc. ;
- corriger vos fautes de frappe au fur et à mesure de l'écriture ;
- vous concentrer sur ce que vous écrivez au lieu de contrôler les touches du clavier ;
- vous ménager le dos, car quand on ne sait pas taper au clavier, la tête fait des mouvements entre le clavier et l'écran, ce qui entraîne une fatigue et une mauvaise position ;
- etc.

Je laisse à chacun le soin de la compléter avec ses idées…

Deux sites pour vous faire croire que vous tapez hyper vite sur votre clavier : essaytyper.com (Lien 35) et hackertyper.net (Lien 36).

Les bases pour bien commencer sont de s'habituer à positionner ses doigts sur le clavier :

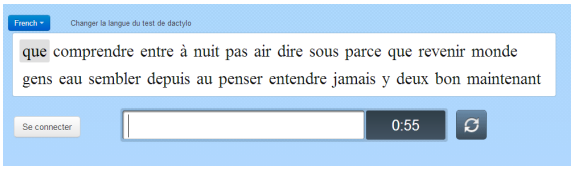
- les index doivent se trouver sur les lettres « F » et « J », ces lettres sont identifiables tactilement, car elles possèdent un tiret en relief ;
- les pouces sont sur la barre d'espace ;
- et, surtout au début, s'habituer à ne pas regarder son clavier pour avancer dans la formation.

3. Les solutions gratuites

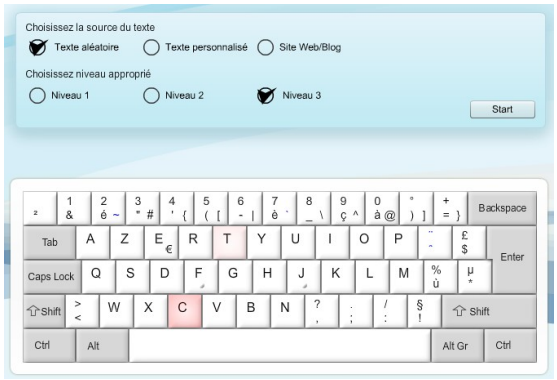
3.1. Les sites Web

Sur Internet, nous trouvons une multitude de sites proposant des méthodologies différentes, à chacun de trouver la sienne.
Je vais pour ma part vous présenter quelques sites qui vont vous permettre d'améliorer votre technique.

10FastFingers : propose de saisir le plus de mots possible en une minute. À chaque fin de tableau, cela vous informe sur vos performances dactylographiques : nombre de caractères tapés à la minute, etc. (Lien 33)



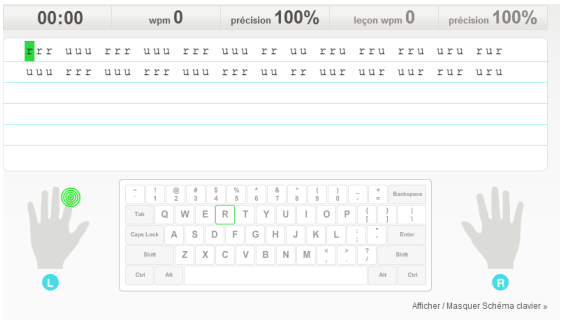
keybr : propose une série de paragraphes aléatoires ou personnalisés à enchaîner avec trois niveaux. Le système enregistre votre avancement et analyse vos erreurs pour vous aider à les éviter. À la fin, cela affiche un graphique de vos performances : vitesse de frappe, erreurs, etc. (Lien 37)



sense-lang : propose une quinzaine de leçons qui vont du niveau facile au plus compliqué, une fois passé ce test vous serez en mesure d'être un bon tapeur de texte. Tout au long de l'exercice, vous avez des compteurs qui vous indiquent vos erreurs, le temps passé, etc. (Lien 38)

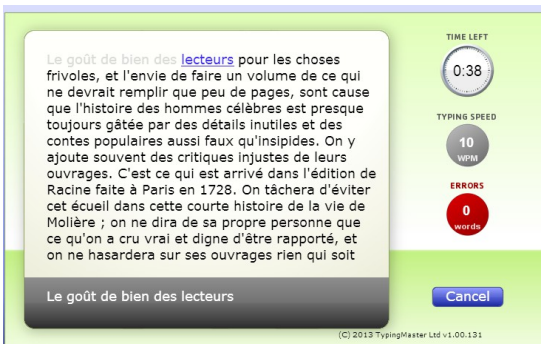
Typingweb : il propose une variété de didacticiels interactifs pour apprendre rapidement la dactylographie pour tous les niveaux (débutant, avancé, spécial). Il y a aussi un entraînement pour les chiffres. Tout au long de l'exercice, vous avez des compteurs qui vous indiquent vos erreurs, le temps passé, etc. (Lien 39)

Le site est en anglais, mais il suffit de changer la langue pour l'avoir en français.

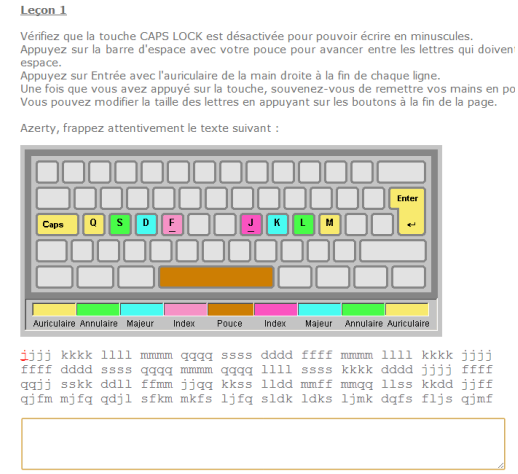


Typingtest : il propose une durée de test ainsi que différents exercices. Il vous suffit ensuite de saisir le texte qui apparaît à l'écran. Tout au long de l'exercice, vous avez des compteurs qui vous indiquent vos erreurs, le temps passé, etc. (Lien 40)

Le site est en anglais, mais il suffit de changer la langue pour l'avoir en français.



Dactylocours : propose dans la méthode gratuite vingt-sept leçons interactives et guidées. Chacun des exercices est là pour vous apprendre à positionner correctement vos doigts. Pour valider un cours et passer au suivant, vos statistiques doivent atteindre un certain niveau. (Lien 41)



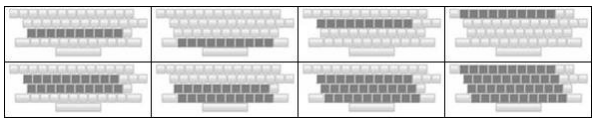
Typinglessons-online : il propose trois exercices :

- des caractères imposés, qui permettent d'apprendre progressivement la position des lettres. Cet exercice a huit niveaux de difficulté de A à H où l'on commence rangée par rangée. Puis, dans chaque niveau, on a une sélection de plusieurs caractères à apprendre, voir *Image 1*
- un texte choisi par le site lui-même ;
- la possibilité de choisir un texte soi-même et de le taper.

(Lien 42)

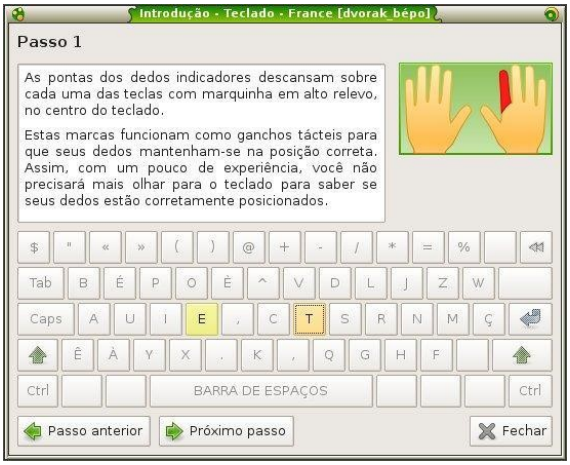
Le site est en anglais, mais l'option du clavier AZERTY est présente.

Image 1 :



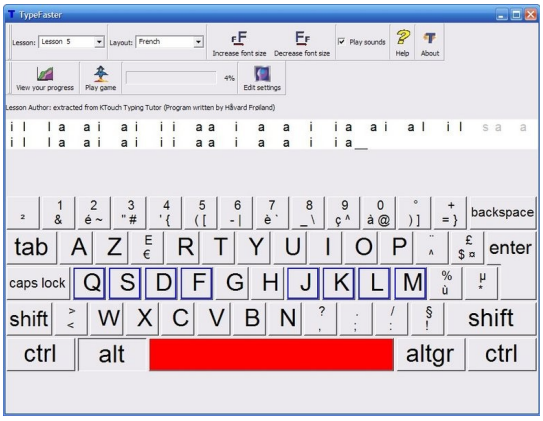
3.2. Les softwares

Klavaro : logiciel d'apprentissage dactylographique, le programme prend en charge les claviers « azerty », « qwerty », etc.
L'apprentissage commence par le positionnement des doigts sur le clavier. Ensuite, au fil des leçons, vous apprendrez à taper de plus en plus vite sur votre clavier. Des statistiques vous permettront de suivre votre progression. (Lien 43)



RapidTyping Typing Tutor : apprendre à taper rapidement au clavier et le positionnement correct des doigts, facilement et gratuitement, avec l'excellent RapidTyping Typing Tutor ! (Lien 44)

TypeFaster : taper rapidement au clavier sans le regarder. Il vous propose d'améliorer votre frappe grâce à un ensemble d'exercices consistant à saisir des séries de plus en plus complexes, allant de deux lettres jusqu'à des mots complets dans les dernières leçons. (Lien 45)



Tux Typing : propose divers petits jeux animés par Tux. Pour chaque activité, plusieurs niveaux de difficulté sont disponibles. Les exercices sont de taper des noms avant qu'ils n'atteignent le bas de l'écran, ou bien de saisir des lettres avant qu'elles ne soient avalées par des poissons, etc. (Lien 46)

4. Quelques astuces !

Afin que les mouvements des mains suivent le rythme de la pensée, voici quelques astuces.
Comme chacun sait, deux doigts ne sont pas suffisants. Il faut se servir des dix doigts. Pour cela, il faut apprendre à positionner les mains et les doigts sur le clavier et pratiquer régulièrement des exercices pour pouvoir taper rapidement.

4.1. Comment placer les doigts ?

Dans le cas d'un clavier « azerty », la technique est simple, et l'on dira que c'est la position initiale :

- il faut poser l'auriculaire gauche sur la touche « Q » et le droit sur « M » ;
- l'annulaire gauche sur la touche « S » et le droit sur la touche « L » ;
- le majeur gauche est placé sur le « D » et le droit sur le « K » ;
- l'index gauche sur le « F » et le droit sur le « J ».

Exemples :

- lorsqu'on a besoin de taper la lettre « G », il faut juste glisser l'index gauche vers la droite et taper ;
- lorsqu'on veut obtenir la lettre « H », on procède de la même façon de glissage, mais cette fois-ci, c'est vers la gauche.

Il est conseillé, dans un premier temps, d'apprendre à taper ces lettres sans regarder le clavier. On peut par exemple commencer de la gauche vers la droite, puis en sens inverse. Ensuite, taper les lettres une à une en intervertissant les tours.

4.2. Comment acquérir une vitesse de frappe ?

- La vitesse ne peut s'acquérir que par une activité régulière d'exercices. L'essentiel étant d'avoir de bonnes habitudes et d'éviter de regarder le clavier.
- Une fois la ligne du milieu maîtrisée, l'étape suivante consistera à essayer de taper les lettres de la ligne supérieure. Pour taper le « A », il suffit de remonter l'auriculaire gauche et taper, et surtout de bien revenir dans sa position initiale « Q ».
- Procéder ainsi pour tous les autres doigts. Pour taper la lettre « T », remonter l'index gauche, puis glisser un peu vers la droite avant de taper.

Revenir ensuite à la position initiale « F ».

- Ensuite passer à la ligne inférieure, les exercices seront les mêmes que pour la ligne supérieure, mais cette fois-ci on glisse le doigt vers le bas, en revenant toujours à la position initiale.
- Comme exercice, on peut copier des textes de journaux chaque jour pendant une durée de quinze minutes. Le fait de lire le texte permet d'attirer le regard, donc évite de regarder le clavier.

Un progrès ne sera visible qu'au bout de quinze jours de pratique régulière.

5. Conclusion

La liste fournie n'est pas exhaustive, car vous vous doutez bien qu'il existe une multitude de sites et de programmes qui permettent d'apprendre à saisir.
Je ne fais aucun jugement sur un site, car un site peut correspondre à une personne et pas à une autre. Donc je vous laisse faire le choix du site qui vous convient le mieux.
Il est même recommandé d'essayer plusieurs solutions (deux ou trois), car chacune développe des méthodologies et des caractéristiques différentes.
Il existe aussi des variantes pour des dispositions de clavier nécessitant l'installation d'un programme : Bépo (Lien 47), Dvorak (Lien 48), etc.

Retrouvez l'article de Vincent Viale et Lana Bauer en ligne : Lien 49

La FAQ OpenOffice et LibreOffice

Qu'appelle-t-on OpenOffice (AOO) et LibreOffice (LibO) ?

Ce sont deux suites bureautiques libres, AOO et LibO sont constituées :

Type	Nom
D'un traitement de texte (.odt)	Traitement de texte
D'un tableur (.ods)	Tableur
D'un module de présentations (.odp)	Présentation
D'un module de dessins (.odg)	Dessin
D'un module pour les bases de données (.odb)	Base de données
D'un éditeur d'équations mathématiques	Formule

Les deux suites sont multiplateformes

Est-il possible de lire les fichiers AOO et LibO si l'application n'est pas installée sur le poste ?

Il existe une visionneuse pour lire les documents issus du Traitement de texte (formats .sxw et .odt) si une des deux suites n'est pas installée sur le poste.
Il s'agit de Visioo-Writer : Lien 50.

Sinon, il existe des versions portables des suites bureautiques, vous pouvez en trouver ici PortableApps (Lien 51), le site est en anglais, mais les suites bureautiques sont en français.
Cela consiste à décompresser un fichier sur le répertoire, qui installera l'application.
Vous pouvez aussi le faire sur une clé USB ou un disque dur externe.

Comment modifier le format par défaut pour l'enregistrement des documents ?

Il faut pour cela aller :

1. Dans le menu « Outils » ;
2. Sélectionnez « Options » ;
3. Sélectionnez « Chargement » et « Enregistrement » dans le menu de gauche ;
4. Puis « Général » ;
5. Dans le champ « Format de fichier par défaut » vous pouvez spécifier le format d'enregistrement pour chaque type de document.

Comment convertir en masse les fichiers de la version 1.x ou des documents Microsoft Office vers la version la plus récente ?

L'option de conversion des anciens fichiers (Version 1.x) n'est pas disponible sous LibO.
Nota : LibO peut ouvrir ces fichiers.

La conversion des documents Microsoft Office fonctionne sur les deux suites, et le mode opératoire est identique à celui des anciens fichiers.

Dans la version 1.x, les documents texte étaient créés au format « .sxw » et les classeurs au format « .sxc ». Ces formats sont toujours utilisables dans la nouvelle version, mais vous pouvez aussi les convertir aux formats « .odt » et « .ods » qui sont les nouveaux standards OpenDocument :

- **Texte formaté** (.odt) application/vnd.oasis.opendocument.text ;
- **Tableur** (.ods) application/vnd.oasis.opendocument.spreadsheet ;
- **Présentation** (.odp) application/vnd.oasis.opendocument.presentation ;
- **Dessin** (.odg) application/vnd.oasis.opendocument.graphics ;
- **Base de données** (.odb)

application/vnd.oasis.opendocument.database ;

Pour effectuer la conversion d'un ou plusieurs documents :

- menu « Fichier » ;
- « Assistants » ;
- « Convertisseur de documents ;
- sélectionnez les types de documents à convertir ;
- cochez l'option « StarOffice » ou « Microsoft Office », puis le ou les types de fichiers à convertir ;
- cliquez sur le bouton « Suivant » ;
- décochez/cochez les options qui vous intéressent dans la nouvelle fenêtre et paramétrez les différentes zones d'information ;
- le champ « Import depuis : » correspond au dossier contenant les fichiers à convertir ;
- le champ « enregistrer dans : » correspond au dossier de destination pour les nouveaux fichiers convertis ;
- cliquez sur le bouton « Suivant » ;
- La fenêtre de paramétrage s'affiche une nouvelle fois si vous avez choisi de convertir plusieurs types de fichiers (Documents texte et Classeurs) ;
- cliquez sur le bouton « Suivant » ;
- cliquez sur le bouton « Convertir » pour lancer l'opération ;
- cliquer sur le bouton « Fermer » dans la boite de dialogue.

Ouvrez le répertoire de destination que vous avez précédemment indiqué afin de visualiser le résultat de la conversion.

Comment convertir des chiffres en lettres ?

Cela n'existe pas par défaut dans les suites bureautiques, mais il existe une macro qui le fait. Elle se trouve ici : Lien 52, il suffit de l'installer comme une extension.

Une fois installée, vous avez une nouvelle barre d'outils qui s'est créée avec deux icônes :

- le premier permet le paramétrage ;
- et le second la conversion.

Ces deux commandes sont aussi accessibles dans le menu « Outils » et « Add-ons ».

Il faut paramétrer, pour cela il faut lancer Tableur et aller dans « Réglages chiffres en lettre ».
Dans les cases à cocher, il faut choisir « France, Canada… », « Demande choix », « À droite » et « Demande de confirmation ».
Ensuite, il ne reste plus qu'à faire des essais, écrivez « 25 » dans la cellule A1, et cliquer sur l'icône **1un**, cela va écrire « vingt-cinq » dans la cellule B1.

Comment utiliser des jockers dans des formules de calcul ?

Par exemple vous voulez savoir combien de cellules commencent par une expression.
Prenons le cas où nous souhaitons connaître le nombre de cellules commençant par « CLIENT ».

La formule devient alors :

```
=NB.SI(A1:A1000;"CLIENT.*")
```

Pour information : « . » correspond au remplacement de n'importe quel caractère, et « * » correspond à la répétition du précédent caractère (ici « . ») un nombre indéterminé de fois.

Comment analyser la structure d'une formule ?

Sélectionnez la cellule qui contient la formule et cliquez sur le bouton « Assistant fonctions » (qui se trouve dans la barre de formule) :

- dans la fenêtre qui s'ouvre, affichez l'onglet « Structure » ;
- la formule apparaît dans une arborescente. Chacun de ses éléments peut-être précédé d'un signe +. Il suffit de cliquer dessus pour voir le contenu de cet élément. Le champ « Résultat » donne le résultat de la fonction analysée ;
- en analysant de cette façon les différentes fonctions de la formule, vous trouverez sans difficulté celle qui pose un problème.

Comment comparer deux versions d'un document ?

Lorsque vous disposez de deux versions d'un même fichier, il est parfois utile de comparer la copie et l'original.

AOO/LibO dispose d'un outil permettant :

- de regrouper les deux documents dans le fichier original ;
- d'identifier les modifications ;
- d'accepter ou de refuser chaque différence.

Ouvrez votre document original, puis sélectionnez le menu « Édition » et « Comparer des documents »
Sélectionnez la copie du document et cliquez sur le bouton « Insérer ».

LibO/AOO combine les deux documents dans votre document initial. Tous les passages de texte qui s'affichent dans votre document, mais pas dans la copie, sont identifiés comme des insertions, et tous les passages de texte manquants dans votre document initial sont identifiés comme des suppressions.

Vous pouvez alors accepter les « insertions », auquel cas le texte correspondant reste dans sa forme d'origine, ou accepter les « suppressions », auquel cas le texte marqué (contenu dans la copie) n'est pas inséré dans le document.

Nota : La fonction de révision est disponible pour les documents texte et les classeurs.

Comment changer le type de guillemets dans les documents ?

Utilisez la procédure suivante pour utiliser les guillemets :

```
" à la place de << et >>
```

- Menu « Outils ».
- « AutoCorrection ».
- Onglets « Guillemets typographiques ».
- Décochez l'option « Remplacer » dans la zone « Guillemets doubles ».
- Cliquez sur le bouton « OK » pour valider.

Cette option est applicable pour les différents types de fichiers LibO/AOO.

Vous pouvez aussi définir des caractères personnalisés en cliquant sur les boutons à droite des champs « En début de mot » et « En fin de mot ». Ces boutons permettent d'afficher la table de caractères.

Comment installer un correcteur grammatical ?

Même s'il y a dans le menu « Orthographe et grammaire » dans le Traitement de texte, il n'y a pas de correcteur grammatical intégré, juste un correcteur orthographique.

Mais il existe deux solutions gratuites :

- Grammalecte : Lien 53 ;
- LanguageTool : Lien 54.

Il existe aussi deux solutions payantes :

- Antidote : Lien 55 ;
- Cordial : Lien 56.

Retrouvez la FAQ OpenOffice et LibreOffice en ligne : Lien 57

Red Hat continuera à maintenir JDK 6 dans le cadre du projet OpenJDK 6 suite à l'annonce de la fin du support de plate-forme par Oracle

Repoussée déjà deux fois, Oracle publie, enfin, la dernière mise à jour du JDK 6 qui met fin à son support public pour le produit. La mise à jour vient corriger une faille majeure dans le plugin Java des navigateurs. Oracle recommande aux utilisateurs de migrer pour la version 7 du JDK.

De son côté, Red Hat Entreprise, l'un des supports actifs de la communauté de l'OpenJDK, annonce qu'il continuera à maintenir le JDK 6 sous la forme OpenJDK 6, malgré l'annonce d'Oracle. Red Hat a une présence effective dans le monde de Java depuis son acquisition de Jboss.

La vision de Red Hat pour le JDK 6 est orientée amélioration des performances et Cloud Computing. Pour l'instant, aucune annonce officielle n'a été faite sur le futur du JDK 6.

Cependant, d'après le site web de l'OpenJDK 6, les mises à jour du JDK 7 qui ne lui seront pas exclusives seront compatibles avec le JDK 6. Ce qui veut dire que les mises à jour du JDK 7 d'Oracle seront intégrées à l'OpenJDK 6 avec le concours de Red Hat.

Par ailleurs, il existe une alternative pour les utilisateurs qui ne voudront pas utiliser l'OpenJDK 6 maintenu par Red Hat. En effet, Oracle propose un support premium moyennant des frais pour le téléchargement des mises à jour du JDK 6. Ce support pourrait durer deux ans voire plus.

La direction du projet OpenJDK6 a été attribuée au développeur Andrew Haley de Red Hat.

La page du projet OpenJDK 6 : Lien 58

Commentez la news de Hinault Romaric en ligne : Lien 59.

Les derniers tutoriels et articles

Chapitre 2 : La programmation orientée objet

Le but de cet article est de présenter de façon précise et concise les notions du langage Java relatives à la programmation orientée objet. En clair, nous parlerons de l'encapsulation, l'héritage, la surcharge des méthodes, la conversion des types, l'instanciation. Sans oublier le couplage et la cohésion.

1. Introduction

Plusieurs parmi nous (je suis le premier) ont appris le langage Java en rassemblant des bouts de code dans le but de réaliser un logiciel (ce que l'on a coutume d'appeler formation sur le tas). Quoique cela a toujours fini par donner un résultat, on ne saurait cependant nier son inconvénient majeur : à la fin on ne sait jamais comment les choses fonctionnent au fond. Ce qui peut avoir pour conséquences : la mauvaise utilisation des types d'objets et de la mémoire qui parfois occasionne des pertes en temps énormes suite à une mauvaise approche lors du débogage.
À la suite du premier chapitre qui traitait des énumérations, des classes, des interfaces, des règles sur les identifiants et du standard JavaBean, ce chapitre (qui est le deuxième de la suite le mémo du certifier Java SE 6 : Lien 60) va s'attarder sur le concept de programmation orientée objet en prenant le soin d'élucider les mécanismes qui gravitent tout au tour. Comme mécanismes, nous parlerons de : l'encapsulation, l'héritage, la surcharge des méthodes, la conversion des types, l'instanciation, du couplage et de la cohésion. À l'issue de cette présentation, cinq exercices corrigés sont mis à la disposition du lecteur pour des besoins d'évaluation personnelle. Je conseille vivement que ce test soit effectué avant et après la lecture de l'article.

2. Encapsulation

L'encapsulation est un mécanisme permettant de masquer l'implémentation des objets tout en les rendant accessibles à travers un certain nombre de services ou interfaces. Du fait qu'elle permet de définir les interfaces de manipulation de données au sein d'un objet, l'encapsulation constitue un gage pour l'intégrité des données.

- L'encapsulation favorise la maintenabilité, la flexibilité et l'extensibilité des applications. Pour la mettre en œuvre, il faut :
- Grâce aux méthodes d'accès (les getters et les setters), il vous sera possible de modifier votre code sans avoir besoin de faire des ajustements supplémentaires sur les classes qui l'utilisent.
- Les variables locales résident dans le stack.

i. Faire précéder les variables d'instance du mot-clé ***protected*** ou ***private*** ;
ii. Définir des accesseurs ***public*** et obliger les objets extérieurs à modifier ou à accéder aux variables d'instance uniquement à travers ses accesseurs. Il ne faut pas que les objets extérieurs puissent directement accéder aux variables d'instance ;
iii. Pour les méthodes d'accès aux variables d'instance, il faut utiliser les conventions de nommage JavaBeans (setXxx and getXxx).

3. L'héritage , Is-A, Has-A

- ***instanceof*** retourne ***true*** si la variable de référence à tester est du même type que la classe utilisée pour faire la comparaison.

```
Test t = new Test(); // t est la variable de
référence
if(t instanceof Object) //Object est la classe
pour la comparaison
  System.out.println("OK");
//le résultat est OK
```

- Toutes les classes Java sont des sous-classes de la classe Object. Excepté la classe Object elle-même. C'est pour cette raison que toutes les classes en Java contiendront toujours les méthodes equals(), clone(), notify(), wait()…
- L'utilisation de l'héritage favorise :
  i. la réutilisation du code ;
  ii. la mise en œuvre du polymorphisme.
- Dans le monde de l'Orienté Objet (OO), le concept de "Is-A" est basé sur l'héritage ou des classes ou l'implémentation des interfaces. Et cela est matérialisé par les mots-clés : ***extends*** pour l'héritage des classes et ***implements*** pour l'implémentation des interfaces.
- La relation "Has-A" est basée sur l'utilisation plutôt que l'héritage. Une classe A "Has-A" B si la classe A possède une référence vers une instance de la classe B. Dans l'exemple ci-dessous, on dira que Voiture "Has-A" Roue et Rav4 "Is-A" Voiture.

```
public class Voiture {
   Roue roue;
}

class Roue {
}
class Rav4 extends Voiture {
}
```

- La relation "Has-A" permet d'élaborer des classes en suivant de bonnes pratiques de l'orienté objet. Car cela évite d'avoir des classes monolithiques qui contiennent tout. Cela facilite aussi la réutilisation du code.

4. La réécriture et la surcharge des méthodes

4.1. La réécriture des méthodes

Lorsqu'on hérite d'une classe, on hérite aussi de son comportement et de ses caractéristiques. Il peut donc arriver pour une raison ou une autre de vouloir changer le mode opératoire d'une méthode. Alors, pour remplacer l'implémentation de la méthode héritée, on va procéder à une réécriture.

- À chaque fois que vous avez une classe qui étend une autre, vous avez la possibilité de réécrire des méthodes, à moins que ces méthodes soient ***final***, ***private*** ou ***static***.
- La réécriture d'une méthode ne doit pas avoir un modificateur d'accès plus restrictif que celui de la méthode réécrite.

```
public class Toyota extends Voiture{

    //erreur de compilation le modificateur doit
être public
    protected void rouler(){}

    public void accelerer(){}//OK

    //erreur de compilation le modificateur doit
être public, default ou protected
    private void freiner(){}

    protected void virer(){}//OK
}

class Voiture {
    public void rouler(){}
    private void accelerer(){}
    void freiner(){}
    void virer(){}
}
```

- Les règles concernant la réécriture des méthodes :
  i. La liste des arguments de la réécriture doit être exactement la même que celle de la méthode réécrite. Sinon il s'agira d'une surcharge au lieu d'une réécriture ;
  ii. Le type de retour doit être le même ou un sous-type du type déclaré dans la méthode réécrite ;

```
class Fabriquant {
  public Voiture fabriquer(){ return new
Voiture();}
}
class Toyota extends Fabriquant{
  public Rav4 fabriquer(){ return new Rav4(); }
}
class Mazda extends Fabriquant {
 public Runner fabriquer(){ return new
Runner(); }
}
class Voiture{}
class Rav4 extends Voiture {}
class Runner extends Voiture {}
```

-
  i. Le modificateur d'accès ne doit pas être plus restrictif ;
  ii. On parle de réécriture de méthode uniquement lorsqu'il y a héritage ;
  iii. La réécriture d'une méthode peut lever de nouvelles exceptions du type "Runtime Exception" même si la méthode réécrite ne l'a pas déclarée ;

```
class Fabriquant {
  public Voiture fabriquer(){
    return new Voiture();
  }
}
class Toyota extends Fabriquant{
  public Voiture fabriquer() throws
RuntimeException{
    return new Voiture();
  }
```

```
  //une erreur de compilation se produit si on
décommente la méthode
  //public Voiture fabriquer() throws
Exception{ return new Voiture();}
}
class Voiture{}
```

- i. La réécriture d'une méthode ne doit pas lever une exception du type “checked exception” qui ne soit pas déclarée par la méthode réécrite ou qui n'ait pas de parent parmi les exceptions déclarées au niveau de la méthode réécrite ;*

```
class Toyota extends Voiture{
  public void rouler() throws
ArithmeticException,
     NullPointerException{} //OK exception du
type RuntimeException

 /*
   erreur de compilation, exception du type
checked
   exception non déclarée dans la super-méthode
 */
 public void accelerer() throws
ClassNotFoundException { }

  // OK, sous-classe de la classe Exception
 public void virer() throws
ClassNotFoundException{}
}

class Voiture{
  public void rouler(){}
  public void accelerer(){}
  public void virer() throws Exception{}
 }
```

- i. La réécriture d'une méthode peut lever moins d'exceptions que la super-méthode. Ceci s'explique par le fait qu'il est possible de traiter dans la réécriture des exceptions levées au niveau de la super-méthode ;

```
class Toyota extends Voiture{
   public void rouler() {} //ok
}

Public class Voiture{
  public void rouler() throws IOException,
ParseException{}
}
```

- i. On ne peut pas réécrire une méthode ***final*** ;
- ii. On ne peut pas réécrire une méthode ***static*** ;
- iii. Si une méthode ne peut pas être héritée (par exemple les méthodes ***private***), elle ne peut non plus être réécrite.

```
class Animal{
  public void eat()
{System.out.println("Animal");}
}
class Horse extends Animal {
  public void eat(){
    super.eat();
    System.out.println("Horse");
  }
}
```

- En Java, il n'existe pas de réécriture de variable. Ainsi, il est possible de déclarer des variables de même nom dans une sous-classe et dans la classe parent sans qu'il y ait interférence.

```
class Animal{
  int age = 6;
}
class Horse extends Animal {
  int age = 10;
  public void eat(){
    Horse h = new Horse();
    Animal a = new Horse();
    System.out.println("ageH : "+h.age+" ageA :
"+a.age);
  }
}
//ageH : 10 ageA : 6
```

4.2. La surcharge des méthodes

Il s'agit d'un mécanisme permettant d'avoir au sein d'une classe, des méthodes/constructeurs ayant le même nom avec des arguments de type différent. Ainsi, on dira qu'il y a surcharge de méthodes/constructeurs dans une classe, lorsque deux méthodes au moins posséderont le même nom avec chacune des particularités au niveau du type d'argument pris en paramètre.

```
class Horse {

  public void eat(){ }
  public void eat(String food){ }
  public void eat(int age){ }
}
```

- Les règles concernant la surcharge des méthodes :
  - i. les méthodes qui participent à la surcharge ne doivent pas avoir les mêmes listes d'arguments ;

```
class Horse {
  public void eat(){ }
  public void eat(String food){ }
  public void eat(String water){ }//erreur à la
compilation
  public void eat(String food, int age){ }
}
```

- i. les méthodes d'une surcharge peuvent avoir des types retour distincts ;
- ii. les méthodes d'une surcharge peuvent avoir des modificateurs d'accès différents ;
- iii. les méthodes d'une surcharge peuvent déclarer des exceptions de façon

indépendante ;

iv. une méthode peut être surchargée dans la même classe ou dans une sous-classe. Ceci dit, deux méthodes de même nom, mais contenues dans différentes classes peuvent toujours être considérées comme surchargées.

```
class Animal{
  public void eat(){}
}
class Horse extends Animal {

  protected Horse eat(int age) throws
Exception{ }
  public int eat(String food){ }
  private Animal eat(String food, int age) throws
RuntimeException{ }
}
```

4.3. Appel des méthodes surchargées

- L'appel d'une méthode surchargée est fonction du type de la référence et non du type de l'objet.

```
class Animal{} class Horse extends Animal {}
class TheAnimals {
  void eat(Animal a)
{System.out.printl("Animal");}
  void eat(Horse h){System.out.printl("Horse");}
  public static void main(String[] args){
    TheAnimals us = new TheAnimals();
    Animal an = new Animal();
    Animal anref = new Horse();
    Horse hor = new Horse();
    us.eat(an);//affiche Animal
    us.eat(anref);//affiche Animal
    us.eat(hor);//affiche Horse
  }
}
```

- Même si à l'exécution anref est de type Horse, le choix de la méthode surchargée à appeler est déterminé à la compilation en fonction du type de référence qui est Animal pour le cas d'espèce.
- De manière générale, l'appel d'une version de méthodes réécrite est déterminé à l'exécution de l'application en fonction du type des objets. Tandis que, l'appel d'une version de méthodes surchargée est déterminé à la compilation en fonction du type de référence des paramètres.
- Quelques différences entre la réécriture et la surcharge :

	Méthode surchargée	Méthode réécrite
La liste des arguments	Doit changer	Ne doit pas changer
Le type de retour	Peut changer	Doit être le même ou un sous-type du type déclaré dans la méthode à réécrire
Exceptions	Peuvent changer	Peuvent réduire ou éliminer les exceptions. Ne doivent pas lever de nouvelles exceptions de type « checked»
Modificateur d'accès	Peut changer	Il ne doit pas être plus restrictif que celui de la méthode à réécrire
Invocation	Elle s'appuie sur le type de référence à la compilation	Elle s'appuie sur le type d'objet à l'exécution

5. Conversion et accès aux variables d'une classe

5.1. Conversion des types de variables (casting)

```
class Animal {}
class Dog extends Animal {
    void playDead(){}
}
If you tried :
   Animal a = new Dog();
   a.playDead(); //you will get a compile error :
cannot find symbol
But if you do something like :
   Dog d = (Dog)animal; //this code compiles!
when you try to run it we'll get an exception
ClassCastException
   Animal a = new Animal();
   String s = (String)animal;//animal can never
be a String we'll get compile error like
inconvertible types.
```

5.2. Méthodes et variables statiques

- La variable ***static*** d'une classe est partagée par toutes les instances de cette classe.

```
public class Animal {

  static int speed;

  public static void main(String[] args){
     Animal an1 = new Animal();
     an1.speed++;
     Animal an2 = new Animal();
     an2.speed++;
     Animal an3 = new Animal();
     an3.speed++;
     System.out.println("speed : "+speed);//le
resultat est 3
  }
}
```

- Une méthode ***static*** ne peut pas directement accéder aux variables ou aux méthodes non static.
- On peut accéder aux objets (variables et méthodes) ***static*** d'une classe en faisant : «nom_de_la_classe ».«nom_objet_static» ou «référence_objet».«nom_objet_static».

```
public class Animal {

  static int speed;
  public static void main(String[] args){
     //ces deux appellations sont identiques
```

```
System.out.println(Animal.speed);//nom_de_la_classe.nom_objet_static
     Animal an = new Animal();

System.out.println(an.speed);//référence_objet.nom_objet_static
  }
}
```

- Une méthode ***static*** ne peut pas être réécrite dans une sous-classe, mais elle peut être redéfinie.

```
public class Animal {

  public static void manger(){
     System.out.println("Animal");
  }
  public static void main(String[] args){
     Animal a = new Animal();
     a.manger();//imprime Animal
     a = new Lion();
     a.manger();//imprime Animal au lieu de Lion
     Lion l = new Lion();
     l.manger();//imprime Lion
  }
}

class Lion extends Animal {
  public static void manger(){
     System.out.println("Lion");
  }
}
```

6. Implémentation des interfaces

- Afin d'implémenter une interface, une classe non doit :
    i. fournir une implémentation concrète pour toutes les méthodes déclarées dans l'interface ;
    ii. respecter toutes les règles de la réécriture des méthodes.
- Une classe non ***abstract*** peut aussi implémenter une interface. Dans ce cas, la classe n'est pas obligée de fournir une implémentation concrète des méthodes de l'interface.
- Une classe peut implémenter plus d'une interface mais ne peut étendre qu'une seule classe.
- Une interface peut étendre une autre interface, mais ne peut rien implémenter d'autre.

```
interface Voiture {

  public void rouler();
}

interface Toyota extends Voiture {}

abstract class Rav4 implements Toyota {}

public class Rav4NewDesign extends Rav4 {
  public void rouler(){  }
}
```

7. Type retour des méthodes

7.1. Déclaration des types de retour

- Pour une méthode surchargée, il n'y a pas de restriction sur le type de retour. Rappelez-vous que pour surcharger une méthode, vous avez seulement besoin de changer les arguments qu'elle prend en paramètre. Ainsi, une méthode surchargée est complètement différente de ses consœurs.
- Pour une méthode réécrite, il est possible de changer le type de retour tant que le nouveau type est un sous-type du type retourné par la méthode réécrite.

7.2. Le retour des valeurs

- Il y a principalement cinq règles à respecter pour le retour des valeurs :
    i. On ne peut retourner ***null*** que pour des méthodes qui retournent des types d'objet (à l'opposé des primitives) ;

```
class Voiture {

  public int vitesse(){ //elle retourne une primitive
     return null;//erreur de compilation
  }

  public Integer kilometrage(){//elle retourne un objet (un wrapper)
     return null;//ok
  }
}
```

-
    i. Il est légal de retourner des tableaux d'objets ou de primitives ;
    ii. Pour une méthode qui retourne une primitive, il est possible de retourner une valeur ou une variable qui peut implicitement être convertie au type de retour déclaré. Dans le cas contraire, il faudra d'abord convertir la valeur à retourner ;

```
class Voiture {
  public int vitesse(){
     char a = 'a';
     return a;
  }

  public int vitesse(double speed){
     return (int) speed;
  }
}
```

-
    i. On ne doit rien retourner pour une méthode qui retourne le type ***void*** ;

```
class Voiture {

//les deux méthodes sont correctes
```

```
  public void vitesse(){
     return;
  }
  public void vitesse(double speed){}
}
```

- i. Pour une méthode qui retourne un objet, il est possible de retourner tout objet qui peut implicitement est converti au type de retour déclaré ;

```
interface Voiture {}

class Toyota implements Voiture{}

class Mazda implements Voiture{}

public class Fabrication {

  public Voiture fabriqueToyota(){
     Toyota t = new Toyota();
     return t;
  }

  public Voiture fabriqueMazda(){
     Mazda m = new Mazda();
     return m;
  }
}
```

8. Les constructeurs

En Programmation Orientée Objet (POO), un constructeur est une méthode « spéciale » d'une classe qui permet de créer une instance d'un objet (instancier), d'allouer la mémoire nécessaire à l'objet et d'initialiser ses attributs.

8.1. Les constructeurs et l'instanciation

- Toute classe, y compris les classes ***abstract*** doivent avoir au moins un constructeur.
- Un constructeur n'a aucun type retour et son nom doit exactement être le même que celui de la classe qui l'abrite.

```
public class Animal {

   public Animal(){}//ok
   public Animal(Object obj){}//ok
   public animal(){}//non,il faut respecter la
casse
   public void Animal(){}//c'est une méthode
légale pas un constructeur
}
```

- Les constructeurs sont généralement utilisés pour initialiser les variables d'instance.
- Un constructeur est invoqué à l'exécution lorsqu'on utilise le mot-clé ***new*** pour l'instance d'un objet : Animal a **= new** Animal**();.**
- Le déroulement du processus de création d'une instance.
- L'instanciation d'un objet en Java suit plusieurs étapes. Pour comprendre ce processus, considérons la création d'un objet Bmwv de l'exemple suivant :

```
class Voiture {
   public Voiture(){
      System.out.print("-Voiture-");
   }
}
public class Bmwv extends Voiture{

   public Bmwv(){
      System.out.print("-Bmwv-");
   }

   public static void main(String… args){
       new Bmwv();
   }
}
//resultat : -Voiture--Bmwv-
```

- i. le constructeur Bmwv**()** est invoqué ;
- ii. avant que la méthode print() ne soit exécutée, le constructeur Voiture**()** est invoqué. Ici, il est important de noter que, tout constructeur invoque toujours le constructeur de la super-classe de façon explicite à travers la méthode super() ou de façon implicite, à moins que le constructeur n'invoque un constructeur surchargé de la même classe. Dans notre cas, Voiture est la super-classe de Bmwv ;
- iii. le constructeur de la classe Object est invoqué car, la classe Object est la super-classe de toutes les classes ;
- iv. les variables d'instance de la classe Object sont initialisées ;
- v. le constructeur de la classe Object est entièrement exécuté ;
- vi. les variables d'instance de la classe Voiture sont initialisées (assignation des valeurs par défaut) ;
- vii. le constructeur de la classe Voiture est entièrement exécuté. Dans notre cas, c'est ici que sera imprimé le texte «-Voiture-» ;
- viii. les variables d'instance de la classe Bmwv sont initialisées (assignation des valeurs par défaut) ;
- ix. enfin, le constructeur de la classe Bmwv est entièrement exécuté. Et c'est ici que sera imprimé le texte «-Bmwv-». D'où le résultat.

- i. un constructeur peut avoir n'importe quel modificateur d'accès, y compris ***private*** (sauf que, dans le cas d'un modificateur d'accès ***private***, la classe doit fournir une méthode ***static*** ou une variable qui permettra l'accès à l'instance créée au sein de la classe) ;
- ii. le nom du constructeur doit être le même que celui de la classe qui l'abrite ;
- iii. un constructeur ne doit pas avoir de type retour ;
- iv. il est possible d'avoir une méthode ayant

le même nom que la classe, mais cela ne fait pas de cette méthode un constructeur.

```
class Voiture {
   public Voiture(){
      System.out.print("-Voiture-");
      super();//erreur de compilation, la méthode
super doit venir en premier
   }

   public Voiture(int i){
   // la méthode super(); sera insérée à la
compilation à ce niveau
      System.out.print("-Voiture-");
   }
   public Voiture(long i){
     this(2);//OK, appel du constructeur qui
prend un int en paramètre
   }
}
```

- La méthode super() peut être appelée sans argument, mais il est aussi possible de lui passer des arguments en fonction des constructeurs définis dans la super-classe.
- Un constructeur sans argument n'est pas forcément un constructeur par défaut (constructeur généré par le compilateur).
- Il est impossible d'accéder à une méthode de classe ou une variable d'instance avant l'appel du super-constructeur.

```
class Voiture {
   public Voiture(int i)  }
}
public class Bmwv extends Voiture{
   int speed;
   static int value;

   public Bmwv(){
      super(speed);//erreur de compilation, speed
est une variable d'instance
      //super(value); OK car value est une
variable static
   }
}
```

- Uniquement des variables static et des méthodes static peuvent être passées en paramètre des appels à super() ou this().
- Les classes abstraites possèdent aussi des constructeurs et ces constructeurs sont toujours appelés lors du processus d'instanciation des sous-classes.
- Une interface ne possède pas de constructeur.
- L'unique façon de faire appel à un constructeur dans un autre c'est d'utiliser la méthode this(). En d'autres termes, il n'est pas possible d'appeler un constructeur comme suit :

```
class Voiture {
   public Voiture(){}
   public Voiture(int i){
      Voiture();//erreur de compilation
    //il faut faire : this();
   }
}
```

8.2. Conditions de création d'un constructeur par défaut

- Le compilateur crée le constructeur par défaut dans une classe lorsque ladite classe ne comporte aucun constructeur à la compilation.
- Les caractéristiques d'un constructeur par défaut :
  i. il possède le même modificateur d'accès que la classe qui le contient ;
  ii. il n'a aucun argument ;
  iii. il fait appel au super-constructeur sans argument (super()).

```
public class Animal {
   protected Animal(){//ce n'est pas un
constructeur par défaut
      super();
   }
}
public class Animal {
   public Animal(){//c'est un constructeur par
défaut
      super();
   }
}
```

- Si une super-classe ne contient que des constructeurs avec argument, vous êtes obligé de faire appel à la méthode super() en lui passant les arguments nécessaires dans tous les constructeurs des sous-classes.

```
class Voiture {
   public Voiture(int speed){}
}
public class Toyota extends Voiture {
   public Toyota(){} //erreur de compilation
   public Toyota(int speed){
      super(0);//OK
   }
}
```

- Si une super-classe ne contient que des constructeurs avec arguments, vous êtes obligé de définir des constructeurs dans toutes ses sous-classes et chaque constructeur devra faire appel à la méthode super() en lui passant les arguments nécessaires.

```
class Voiture {
   public Voiture(int speed){}
}
public class Toyota extends Voiture {
   //erreur de compilation
}
```

8.3. La surcharge des constructeurs

- Surcharger un constructeur signifie créer plusieurs versions de constructeurs avec des arguments différents.

```
public class Voiture {
   public Voiture(){}
   public Voiture(short sieges){}
   public Voiture(int speed){}
}
```

- Si vous voulez appeler un constructeur dans un autre, vous devez utiliser le mot-clé ***this.***

```
class Animal {
String name;
Animal(String name){this.name = name}
Animal(){ this("My name");}//appel du premier
constructeur
}
```

- La première instruction dans un constructeur doit être l'appel de la méthode super() ou this(). Si aucune de ces instructions n'est insérée, le compilateur va insérer la méthode super() sans argument.
- Un constructeur ne peut pas avoir à la fois un appel à super() et un appel à this(), car chacune de ses méthodes doit être la première instruction dans un constructeur.
- Un compilateur ne peut pas insérer la méthode super() dans un constructeur qui contient déjà la méthode this().

9. Le couplage et la cohésion

L'objectif d'une bonne conception est de faciliter la création, la maintenance et l'évolution d'une application. Pour ce faire, elle devra favoriser un couplage lâche entre les entités et une forte cohésion de chacune des entités.

9.1. Le couplage

- Le couplage ici permet d'exprimer le niveau de visibilité qu'une classe a sur une autre. Ainsi, deux classes seront fortement couplées si le changement d'une variable dans l'une nécessite impérativement un changement dans l'autre.
- Si deux classes s'échangent des données uniquement à travers des interfaces alors les deux classes sont dites faiblement couplées. Ce qui est une bonne chose. Dans le cas d'espèce, la classe Voiture et la classe Roue sont faiblement couplées. Ainsi, il devient possible de modifier l'attribut diamètre ou poids dans la classe Roue sans toutefois perturber la relation Voiture-Roue.

```
class Voiture {
    Roue roue;
    public void montage(){
        roue = new Roue();
        roue.setDiametre(12.5);
        roue.setPoids(19);
    }
}

class Roue {
   private double poids;
   private double diametre;

    public double getDiametre() {
        return diametre;
    }

    public void setDiametre(double diametre) {
        this.diametre = diametre;
    }

    public double getPoids() {
        return poids;
    }

    public void setPoids(double poids) {
        this.poids = poids;
    }
}
```

- Si une classe A accède aux données d'une classe B sans passer par les interfaces de la classe B alors les classes A et B sont dites fortement couplées. Ce qui n'est pas une bonne chose pour une application.

```
class Voiture {
    Roue roue;
    public void montage(){
        roue = new Roue();
        roue.diametre = 12.5;
        roue.poids = 19;
    }
}

class Roue {
   public double poids;
   public double diametre;
}
```

9.2. La cohésion

- Alors que le couplage concerne les interactions entre les classes, la cohésion se focalise sur la conception interne d'une classe.
- La cohésion permet d'exprimer le niveau de conception d'une classe en tant qu'entité indépendante. Cela nécessite l'encapsulation des données et le masquage des informations.
- Une classe ayant une forte cohésion est facilement maintenable et réutilisable.

10. Exercices

10.1. Corrigez les erreurs et trouvez le résultat

```
class Animale {
  public Animale(String nom)
{ System.out.print("B"); }
}
public class Chien implements Animale {
  public Chien(String nom)
{ System.out.print("A"); }
  public static void main(String … args) {
    new Chien("C");
    System.out.println(" "); }}
```

10.2. Corrigez les erreurs et trouvez le résultat

```
class Animale {
  private final void manger()
{ System.out.println("Animale"); }
}
public class Chien extends Animale {
  public final void manger()
{ System.out.println("Chien");  }
  public static void main(String [] args) {
    new Chien().manger();
```

```
  }
}
```

10.3. Corrigez les erreurs et trouvez le résultat

```
class Animale { public void manger() {
    System.out.println("Animale");} }

 class Chien extends Animale {
    void manger() {System.out.println("Chien");}
    void aboie() {System.out.println("Chien-
aboie");}
    }

 class Courrir {
   public static void main(String [] args) {
     Animale a1 = new Chien();
     Animale a2 = new Chien();
     Chien c = new Chien();
        a1.manger();
        a1.manger();
      a1.aboie();
        c.manger();
      c.aboie();
 } }
```

10.4. Corrigez les erreurs et trouvez le résultat

```
public class Chien extends Animale {
   public static String manger() { return
"viande"; }
   public static void main(String args[]) {
     Chien c1 = new Chien();
     Animale a1 = new Chien();
     Animale a2 = new Animale();
     System.out.println(c1.manger() + " " +
a1.manger() + " " + a2.manger());
   }
 }
 class Animale { public static String manger()
{ return "nourriture"; } }
```

10.5. Corrigez les erreurs et trouvez le résultat

```
class Animale { }
class Chien implements Animale { }
public class Foret {
   String s = "-";
   public static void main(String[] args) {
     Animale[] a = new Animale[2];
     Chien[] c = new Chien[2];
     chercher(a);
     chercher(c);
     chercher(7);
     System.out.println(s);
   }
   static void chercher(Animale[]… a)    { s +=
"1"; }
   static void chercher(Chien[]… b)    { s +=
"2"; }
   static void chercher(Chien[] b)       { s +=
"3"; }
   static void chercher(Object o)     { s += "4";
}
 }
```

11. Correction des exercices

11.1. Exercice 1

a. **Erreurs**

On implémente une interface et on étend une classe. Il faut donc utiliser le mot-clé ***extends*** entre Chien et Animale

Lorsqu'une classe mère n'a pas de constructeur sans argument, la classe fille doit explicitement faire appel à la méthode *super()* dans tous ses constructeurs en lui passant les paramètres nécessaires. Dans notre cas, nous allons appeler la méthode super() avant l'instruction System.out.print("A"); en lui passant le nom de l'animal comme paramètre.

a. **Solution**

Le résultat après correction des anomalies est : BA. Car, à l'instanciation de la classe Chien, le constructeur de la classe mère (Animale) sera entièrement exécuté avant celui de la classe fille (Chien).

11.2. Exercice 2

a. **Erreurs**

Il n'y a aucune erreur. Par contre, si la méthode manger () de la classe Animale était ***public, default*** ou ***protected***, un problème allait se poser parce que l'on ne peut réécrire une méthode ***final***.

a. **Solution**

La solution est : Chien

11.3. Exercice 3

a. **Erreurs**

La réécriture d'une méthode ne doit pas avoir un modificateur d'accès plus restrictif que celui de la méthode réécrite. Ici, la méthode manger() de la classe Chien doit avoir ***public*** comme modificateur d'accès .

L'appel d'une version de méthode réécrite est déterminé à l'exécution de l'application en fonction du type des objets et non du type de référence. Dans le cas présent, pour la variable a1 le type d'objet est Chien et le type de référence est Animale. Or la classe animale ne possède pas de méthode aboie(). D'où il faudra commenter l'instruction a1.aboie**();**

a. **Solution**

La solution est :
Chien
Chien
Chien
Chien - aboie

11.4. Exercice 4

a. **Erreurs**

Il n'y a aucune erreur.

a. **Solution**

La solution sera : viande nourriture nourriture. Il faut se rappeler que l'on ne peut pas réécrire une méthode ***static***. Ceci dit, l'instruction a1.manger() fait appel à la méthode manger() de la classe Animale au lieu de la classe Chien qui est le type de l'objet.

11.5. Exercice 5

a. **Erreurs**

On implémente une interface et on étend une classe. Il

faut donc utiliser le mot-clé ***extends*** entre Chien et Animale.

On ne peut accéder aux variables non-static dans une méthode static. Il faut donc déclarer la variable « s » comme ***static***.

a. **Solution**

Le résultat est : -434

12. Conclusion

Le prochain article traitera principalement des variables. On verra entre autres, la création, l'initialisation, le transtypage, le passage en paramètre et la suppression des variables.

Retrouvez l'article de Armel Fabrice Ndjobo en ligne : Lien 61

Google s'attaque aux applications antipub et éjecte AdAway, AdFree et AdBlock de Google Play

Après s'en être pris à l'application AdBlock, Google frappe un plus grand coup. Pour rappel, AdBlock est une application antipublicité qui configure automatiquement le serveur proxy utilisé pour détourner le trafic web.

Désormais toutes les applications antipub sont dans le collimateur du géant.

La firme parle du non-respect de la section 4.4 de la convention du développeur sur Google Play. Elle stipule entre autres que « Vous vous engagez à ne pas vous livrer à aucune activité dans le marché […] qui perturbera, désorganisera, endommagera, ou accèdera de manière non autorisée à des dispositifs, des serveurs, des réseaux, ou d'autres biens ou services d'une tierce partie, y compris, mais sans s'y limiter, les utilisateurs d'Android, de Google ou de tout autre opérateur de réseau mobile… »

La firme perçoit la fonction antipub comme une altération d'un service fourni par un tiers. Elle s'est contentée d'envoyer des notifications aux bloqueurs de publicité comme AdAway ou encore AdFree pour leur signifier que leurs applications avaient été retirées du store en raison d'une violation des conditions de distribution.

Probablement une façon pour le géant de la recherche de protéger ses revenus ?

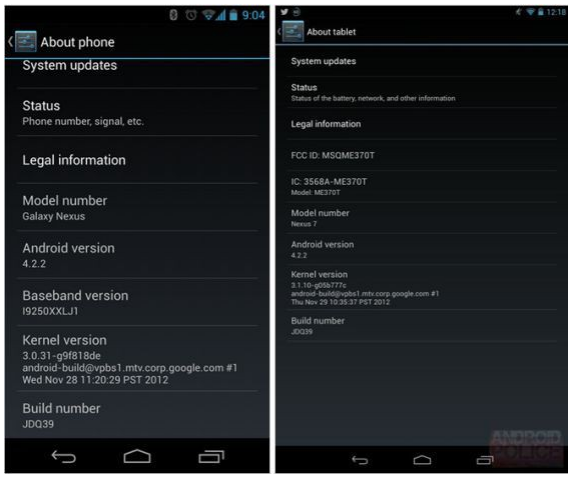
Commentez la news de Stephane Le Calme en ligne : Lien 62

Google déploie Android 4.2.2 une mise à jour qui améliore la stabilité, les performances et apporte quelques correctifs de bogues

Google a débuté le déploiement d'une mise à jour mineure de son système d'exploitation mobile Android.

Android 4.2.2 n'apporte pas de nouveautés majeures, en dehors des améliorations de performances, de la stabilité, des corrections de bogues, notamment un bogue lié à la lecture de la musique via une connexion Bluetooth.

Les premiers dispositifs à recevoir cette mise à jour sont les smartphones Galaxy Nexus, les tablettes Nexus 7 et Nexus 10. Le smartphone Nexus 4 suivra, ainsi que les autres types de terminaux sous l'OS.



Rien de vraiment surprenant, cette mise à jour n'apporte pas de nouvelles fonctionnalités. Elles sont certainement réservées à la prochaine version majeure de l'OS mobile (Android 5.0) qui, selon le calendrier de Google, pourrait être dévoilée en mai prochain lors de l'événement Google I/O.

Selon des rumeurs, Android 5.0, dont le nom de code serait « Key Lime Pie », pourrait apporter comme nouveautés : un widget Google, un meilleur support des comptes utilisateurs, la sélection multiple dans les contacts, une application native pour la visioconférence et l'intégration en natif des réseaux sociaux populaires.

Commentez la news de Hinault Romaric en ligne : Lien 63

ViewPager sous Android - Comment slider d'un Fragment à un autre

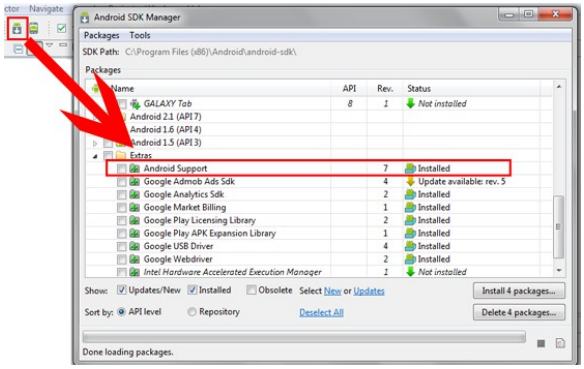
Nous revoici pour un nouveau tutoriel, le premier à aborder la notion de Fragment. Les Fragments ont été introduits dans la version Android 3.0 (également utilisable avec des versions antérieures) avec pour objectif de permettre une plus grande flexibilité pour les écrans larges tels que les tablettes tactiles (ce que nous verrons dans un prochain tutoriel). Notre objectif aujourd'hui est simplement de créer un joli effet « slide » entre des pages.

1. Qu'est-ce qu'un Fragment ?

C'est certainement la question que vous devez vous poser. On peut voir un Fragment comme une « miniActivity » ou une portion d'une activité qui ne peut pas vivre en dehors d'une Activity. Un des aspects pratiques du Fragment est qu'il peut être facilement réutilisé d'une Activity à l'autre.

2. Notre projet

Comme d'habitude, je vous invite à créer un projet dédié à ce tutoriel, que j'ai baptisé TutoFragment . Je l'ai volontairement créé pour la version Android 2.2 pour y intégrer le package de compatibilité, puisqu'à la base, les Fragments sont une fonctionnalité de la version 3.0.
Nous allons donc premièrement télécharger la bibliothèque complémentaire. Pour cela, allez dans le SDK Manager en cliquant sur l'icône suivante et téléchargez « Android Support ».



Ajoutons ensuite cette bibliothèque au projet. Pour cela, sélectionnez votre projet sous Eclipse, clic droit/Android tools/Add support library. La bibliothèque s'intègre automatiquement au projet, ce qui nous permet d'utiliser la classe Fragment !

3. Les fichiers XML

Le layout principal, que j'ai renommé viewpager.xml , se présente de la manière suivante :

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.v4.view.ViewPager

xmlns:android="http://schemas.android.com/apk/res
/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/viewpager">
</android.support.v4.view.ViewPager>
```

La balise du composant utilisé ici a une syntaxe particulière puisqu'elle fait référence à un composant de la bibliothèque ajoutée précédemment.
Les trois autres layouts seront les pages affichées successivement lors du slide et possèdent un code similaire. Voici le code de page_gauche_layout.xml , je vous laisse créer les deux autres par vous-même (page_milieu_layout et page_droite_layout).

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >


    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="20dp"
        android:text="Page de gauche" />

</LinearLayout>
```

4. Le code Java

J'ai renommé la classe principale en FragmentsSliderActivity . Celle-ci dérive de la classe FragmentActivity qui est simplement une Activity permettant de gérer les Fragments. Le code est commenté, je vous laisse regarder.

```
import java.util.List;
import java.util.Vector;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentActivity;
import android.support.v4.view.PagerAdapter;
import android.support.v4.view.ViewPager;

public class FragmentsSliderActivity extends
FragmentActivity {

        private PagerAdapter mPagerAdapter;

        @Override
        protected void onCreate(Bundle
savedInstanceState) {

super.onCreate(savedInstanceState);
```

```
super.setContentView(R.layout.viewpager);

                // Création de la liste de
Fragments que fera défiler le PagerAdapter
                List fragments = new Vector();

                // Ajout des Fragments dans la
liste

fragments.add(Fragment.instantiate(this,PageGauch
eFragment.class.getName()));

fragments.add(Fragment.instantiate(this,PageMilie
uFragment.class.getName()));

fragments.add(Fragment.instantiate(this,PageDroit
eFragment.class.getName()));

                // Création de l'adapter qui
s'occupera de l'affichage de la liste de
                // Fragments
                this.mPagerAdapter = new
MyPagerAdapter(super.getSupportFragmentManager(),
fragments);

                ViewPager pager = (ViewPager)
super.findViewById(R.id.viewpager);
                // Affectation de l'adapter au
ViewPager

pager.setAdapter(this.mPagerAdapter);
        }
}
```

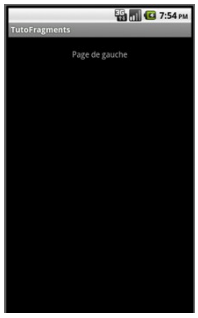
Pour passer d'une page à une autre, nous avons besoin d'un adapter (à la manière des ListViews). L'adapter, appelé MyPagerAdapter dérive de FragmentPagerAdapter. Rien de compliqué, il fonctionne sur le même principe que les Adapters de ListViews.

```
import java.util.List;

import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import
android.support.v4.app.FragmentPagerAdapter;

public class MyPagerAdapter extends
FragmentPagerAdapter {

        private final List fragments;

        //On fournit à l'adapter la liste des
fragments à afficher
        public MyPagerAdapter(FragmentManager fm,
List fragments) {
                super(fm);
                this.fragments = fragments;
        }

        @Override
        public Fragment getItem(int position) {
                return
this.fragments.get(position);
        }

        @Override
        public int getCount() {
                return this.fragments.size();
        }
}
```

Enfin, la classe PageGaucheFragment , qui dérive de la classe Fragment possède des méthodes différentes de la classe Activity.
Ici, nous allons surcharger la méthode onCreateView . Cette méthode retourne un type View , on va donc lui indiquer notre layout par le biais de l'inflater. Les classes PageMilieuFragment et PageDroiteFragment sont quasi identiques, je vous laisse vous en occuper.

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class PageGaucheFragment extends Fragment
{

        @Override
        public View onCreateView(LayoutInflater
inflater, ViewGroup container,
                        Bundle
savedInstanceState) {
                return
inflater.inflate(R.layout.page_gauche_layout,
container, false);
        }
}
```

Rendu final :



5. Conclusion

Ainsi se termine notre premier tutoriel sur les Fragments, en espérant que tout soit clair. Vous pouvez trouver le code source ici : Lien 64.

Retrouvez l'article de Maxim Benbourahla en ligne : Lien 65

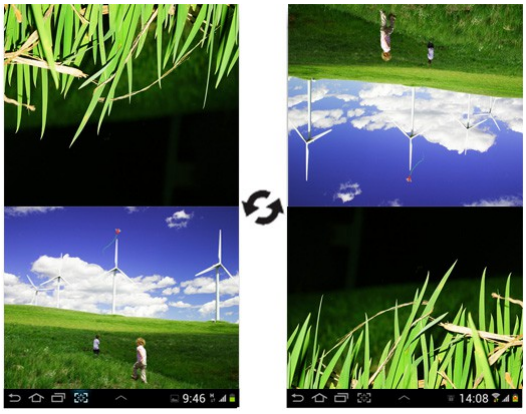
Multi-touch et fragments - Geste de « rotation »

Bonjour à tous ! Ce nouveau tutoriel orienté algorithmique va tenter d'expliquer la manière de gérer le geste de rotation multi-touch. Peu de code Java dans ce tutoriel, juste de quoi illustrer mes propos et donner des pistes de réflexion. La fonction principale sera détaillée, mais exceptionnellement, je ne mettrai pas de projet en téléchargement par manque de temps pour l'implémenter proprement.

Pour les gestes simple-touch, l'implémentation est très facile grâce au Gesture Builder (tutoriel en anglais ici : Lien 66), mais pour le multi-touch, c'est un peu plus compliqué car aucune bibliothèque n'existe. Si on veut pouvoir utiliser des gestes cool dans son application, il faut donc les implémenter.

1. Principe global

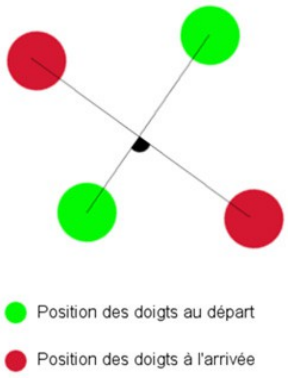
On souhaite reconnaître un mouvement de rotation effectué avec deux doigts pour par exemple inverser deux fragments positionnés l'un en dessous de l'autre comme indiqué sur l'image ci-dessous.



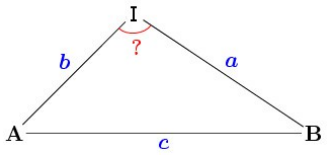
Concrètement, nous allons enregistrer les coordonnées de départ et d'arrivée de nos deux doigts puis calculer l'angle formé par ceux-ci, comme indiqué dans le schéma ci-dessous. Dans notre cas, si l'angle mesuré est supérieur à 8° (par exemple), la rotation s'effectue.

2. Algorithme

On détecte lorsque plusieurs doigts sont posés sur l'écran puis on sauvegarde les coordonnées des points touchés. Lorsque les doigts sont enlevés, on enregistre également leurs coordonnées.



On calcule ensuite le point d'intersection I de la droite formée par les deux doigts au départ et de celle formée à l'arrivée. Appelons A le point de départ du doigt 1 et B son point d'arrivée. L'étape suivante consiste à calculer les distances des côtés du triangle ABI. Comme on possède les coordonnées de ces points, il ne reste qu'à utiliser la formule :
AB= (xA - xB)² + (yB - y1)² , où A (xA, yA) et B (xB, yB)



Enfin, pour calculer l'angle en radian, on utilise le théorème d'Al Kashi (ou théorème de Pythagore généralisé) dont la formule est la suivante :

$$\gamma = \arccos \frac{a^2 + b^2 - c^2}{2ab}.$$

3. Implémentation Java : quelques pistes

Premièrement, il faut placer un listener sur votre layout principal, celui qui fait la taille de la page et qui contient tous les éléments. C'est lui qui captera l'événement.

```
mainLayout.setOnTouchListener(new
View.OnTouchListener() {

        @Override
        public boolean onTouch(View view,
MotionEvent motionEvent) {

                int pointerIndex =
((motionEvent.getAction() &
MotionEvent.ACTION_POINTER_ID_MASK) >>
MotionEvent.ACTION_POINTER_ID_SHIFT);
                int action =
(motionEvent.getAction() &
MotionEvent.ACTION_MASK);
                int pointCnt =
motionEvent.getPointerCount();
                try {
                        // On effectue un
traitement seulement lorsque deux doigts touchent
l'écran
                        if ((pointCnt == 2)&&
(pointerIndex <=1)){
                                switch (action) {
                                        case
MotionEvent.ACTION_POINTER_DOWN:
                                                /
/ Sauvegarde des coordonnées de départ des deux
doigts

for (int i = 0; i < pointCnt; i++) {
```

```
int id = motionEvent.getPointerId(i);

//simple exemple de comment récupérer les
coordonnées des points. Celles-ci doivent être
traitées en fonction de ce que l'on veut faire

float x= motionEvent.getX(i)

float y= motionEvent.getY(i));
                                                }


break;

                                        case
MotionEvent.ACTION_POINTER_UP:
                                                /
/ Sauvegarde des coordonnées des doigts lorsque
l'utilisateur les enlève de l'écran

for (int i = 0; i < pointCnt; i++) {

int id = motionEvent.getPointerId(i);

//simple exemple de comment récupérer les
coordonnées des points. Celles-ci doivent être
traitées en fonction de ce que l'on veut faire

float x= motionEvent.getX(i)

float y= motionEvent.getY(i));
```

```
                                                }
                                                /
/calculer ici la valeur de l'angle formé, comme
expliqué dans le paragraphe précédent

break;

                                        default:

break;

                                    }
                                }
                            } catch (Exception e) {
                                    Log.e("error",
e.getMessage());
                            }
                            return true;
                    }
});
```

4. Conclusion

En espérant que ce tutoriel vous soit utile pour la compréhension du multitouch. Vous avez maintenant toutes les cartes en main pour l'implémenter !

Retrouvez l'article de Maxim Benbourahla en ligne : Lien 67

Comment calculer la position et la normale dans le vertex shader avec OpenGL (GLSL) et Direct3D (HLSL) - première partie

La première étape à effectuer dans un vertex shader est le calcul de la position et si besoin de la normale. Cet article vous explique en détail ce calcul.

1. Explications

Cet article est valide pour toutes les versions d'**OpenGL** (2, 3 et 4) ainsi que toutes les versions de **Direct3D** (9, 10 et 11). Je donne les exemples de code en OpenGL 2 et en Direct3D 9. Les codes sont compatibles avec OpenGL 3/4 et Direct3D 10/11 et peuvent être facilement déduits des versions OpenGL 2/D3D9.
Pour calculer la position transformée (gl_Position en GLSL, dans l'espace de clipping), vous avez besoin de trois matrices : **projection**, **view** (vue) et **model** (modèle). Les matrices de projection et de vue sont les matrices correspondant à la caméra et la matrice de modèle est la matrice de transformation de l'objet actuellement affiché.
Les trois matrices sont passées par l'application hôte au vertex shader comme variables d'entrée (déclarées uniform en GLSL).
Dans le code des vertex shaders suivants, P est la position XYZ du vertex dans l'espace local (espace du point de vue de l'objet), c'est-à-dire la position sans aucune transformation. Il en est de même pour N, la normale du vertex dans l'espace local, sans aucune transformation.

2. Shader OpenGL (GLSL)

```
uniform mat4 projectionMatrix;
uniform mat4 viewMatrix;
uniform mat4 modelMatrix;

varying vec3 normal;

void main()
{
    vec3 P = gl_Vertex.xyz;
    vec3 N = gl_Normal.xyz;

    mat4 modelView = viewMatrix * modelMatrix;
    mat4 modelViewProjection = projectionMatrix *
modelView;
    gl_Position = modelViewProjection * vec4(P,
1.0);

    normal = modelView * vec4(N, 0.0);
}
```

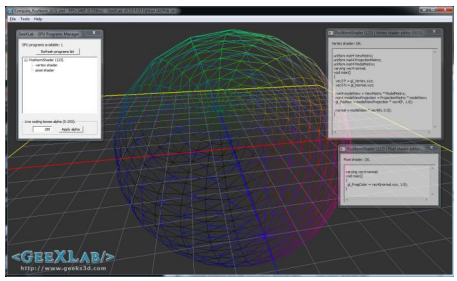
3. Shader Direct3D (HLSL)

```
float4x4 projectionMatrix;
float4x4 viewMatrix;
float4x4 modelMatrix;

struct VS_OUTPUT
{
    float4 Position : POSITION;
    float3 Normal : TEXCOORD0;
};

VS_OUTPUT VertexShaderD3D(float4 P : POSITION,
float3 N : NORMAL)
{
    VS_OUTPUT o;
    float4x4 modelView = mul(modelMatrix,
viewMatrix);
    float4x4 modelViewProjection = mul(modelView,
projectionMatrix);
    o.Position = mul(float4(P.xyz, 1.0),
modelViewProjection);
    o.Normal = mul(N, (float3x3)modelView);
    return o;
}
```

4. Conclusion

La convention pour les matrices dans OpenGL est **column-major** alors que les matrices Direct3D sont **row-major**. Cela explique pourquoi les multiplications de matrices ne sont pas effectuées dans le même ordre entre OpenGL et Direct3D.

Si vous souhaitez jouer avec le shader GLSL (programmation en live), vous pouvez le trouver dans le pack de code (Lien 68) de GeeXlab dans le dossier GLSL_ComputePosNorm/ de l'archive. La dernière version de GeeXlab (Lien 69) est disponible sur le site de Geeks3D (Lien 70).



Retrouvez l'article de JeGX en ligne : Lien 71

Comment calculer la position et la normale dans le vertex shader avec OpenGL (GLSL) et Direct3D (HLSL) - deuxième partie

Suite du précédent tutoriel où l'on apprenait comment calculer la position et la normale dans le vertex shader. Cet article revient sur ce calcul pour en analyser la performance.

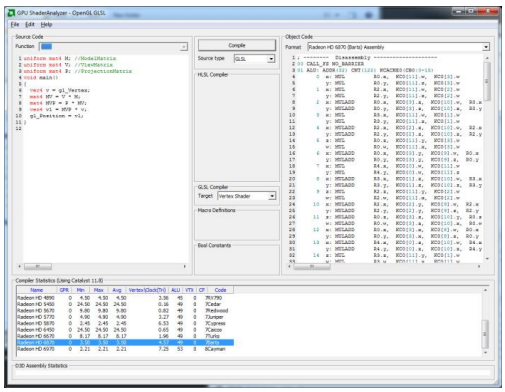
1. Optimisation du vertex shader

Précédemment, j'ai publié un simple code d'exemple pour calculer la position et la normale dans le vertex shader en **OpenGL** (GLSL) et **Direct3D** (HLSL). Le but était de montrer le fonctionnement des différentes matrices : projection, view (vue) et model (modèle) pour déterminer la variable **gl_Position** et notamment l'ordre des multiplications des matrices avec OpenGL et Direct3D. Voici le vertex shader GLSL (nommé VS_A) :

Vertex shader GLSL : VS_A

```
uniform mat4 M; // Matrice du modèle
uniform mat4 V; // Matrice de vue
uniform mat4 P; // Matrice de projection

void main()
{
    vec4 v = gl_Vertex;
    mat4 MV = V * M;
    mat4 MVP = P * MV;
    vec4 v1 = MVP * v;
    gl_Position = v1;
}
```

Cette méthode construit la matrice de transformation finale (MVP ou ModelViewProjection) et la multiplie avec la position du vertex dans l'espace local de l'objet (gl_Vertex). Cette méthode fonctionne, mais génère un grand nombre d'instructions GPU.
J'ai démarré l'outil GPU Shader Analyzer et compilé le vertex shader précédent :



Compiler Statistics (Using Catalyst 11.8)

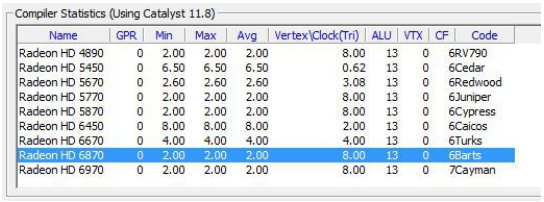
Name	GPR	Min	Max	Avg	Vertex\Clock(Tri)	ALU	VTX	CF	Code
Radeon HD 4890	0	4.50	4.50	4.50	3.56	45	0		7RV790
Radeon HD 5450	0	24.50	24.50	24.50	0.16	49	0		7Cedar
Radeon HD 5670	0	9.80	9.80	9.80	0.82	49	0		7Redwood
Radeon HD 5770	0	4.90	4.90	4.90	3.27	49	0		7Juniper
Radeon HD 5870	0	2.45	2.45	2.45	6.53	49	0		7Cypress
Radeon HD 6450	0	24.50	24.50	24.50	0.65	49	0		7Caicos
Radeon HD 6670	0	8.17	8.17	8.17	1.96	49	0		7Turks
Radeon HD 6870	0	3.50	3.50	3.50	4.57	49	0		7Barts
Radeon HD 6970	0	2.21	2.21	2.21	7.25	53	0		8Cayman

Une fois compilé, le vertex shader VS_A génère **49 instructions UAL** (Note de traduction : unité arithmétique et logique (en anglais, *Arithmetic Logic Unit*, soit ALU). Il s'agit du composant effectuant les opérations arithmétiques et logiques dans un processeur. Dans le contexte de l'article, cela correspond aux cœurs du processeur de la carte graphique exécutant les shaders).
Daniel Rakos m'a montré une solution plus rapide :

Vertex shader GLSL : VS_B

```
uniform mat4 M; // Matrice du modèle
uniform mat4 V; // Matrice de vue
uniform mat4 P; // Matrice de projection
void main()
{
  vec4 v = gl_Vertex;
  vec4 v1 = M * v;
  vec4 v2 = V * v1;
  vec4 v3 = P * v2;
  gl_Position = v3;
}
```

Une fois compilé dans GPU ShaderAnalyzer (Lien 72), VS_B génère **13 instructions UAL**.



Compiler Statistics (Using Catalyst 11.8)

Name	GPR	Min	Max	Avg	Vertex\Clock(Tri)	ALU	VTX	CF	Code
Radeon HD 4890	0	2.00	2.00	2.00	8.00	13	0		6RV790
Radeon HD 5450	0	6.50	6.50	6.50	0.62	13	0		6Cedar
Radeon HD 5670	0	2.60	2.60	2.60	3.08	13	0		6Redwood
Radeon HD 5770	0	2.00	2.00	2.00	8.00	13	0		6Juniper
Radeon HD 5870	0	2.00	2.00	2.00	8.00	13	0		6Cypress
Radeon HD 6450	0	8.00	8.00	8.00	2.00	13	0		6Caicos
Radeon HD 6670	0	4.00	4.00	4.00	4.00	13	0		6Turks
Radeon HD 6870	0	2.00	2.00	2.00	8.00	13	0		6Barts
Radeon HD 6970	0	2.00	2.00	2.00	8.00	13	0		7Cayman

J'ai rapidement effectué un test dans GeeXLab avec un objet (sphère) contenant deux millions de faces et un million de vertex :
- VS_A : 425 FPS ;
- VS_B : 425 FPS.

J'ai aussi effectué un test avec une simple démonstration OpenGL (une simple application de test win32) avec de l'instanciation géométrique (10 000 instances, 900 vertex par instance), sans trouver de différence entre les deux vertex shaders. Je m'attendais à une petite différence. Une explication peut être due à la simplicité de la scène. Je suis cependant sûr que nous allons voir une différence dans des shaders ou des scènes 3D plus complexes, car il doit y avoir un gain de performances entre 49 et 13 instructions ALU…

1.1. Mise à jour du 31 octobre 2011

Je pense avoir trouvé une réponse possible : le surcoût provoqué par l'appel du shader et/ou de l'accès aux vertex. En effet, si la charge de travail réelle du vertex shader est trop petite, la majorité du temps est perdu dans d'autres tâches telles que la récupération des vertex ou l'appel au shader ou, plus généralement, toute tâche venant avant ou après l'exécution du vertex shader. En conséquence, une différence d'une trentaine d'instructions ALU n'est pas visible.
Pour valider cette idée, j'ai augmenté le travail du vertex shader et comparé les deux vertex shaders suivants avec

un objet constitué d'un million de vertex :

Vertex shader A (vsA)
```
uniform mat4 M; // Matrice du modèle
uniform mat4 V; // Matrice de vue
uniform mat4 P; // Matrice de projection

void main()
{
    vec4 v = gl_Vertex;
    vec4 pos = vec4(0.0);
    for (int i=0; i<100; i++)
    {
        mat4 MV = V * M;
        mat4 MVP = P * MV;
        vec4 v1 = MVP * v;
        pos += v1;
    }
    gl_Position = pos / 100.0;
}
```

et

Vertex shader B (vsB)
```
uniform mat4 M; // Matrice du modèle
uniform mat4 V; // Matrice de vue
uniform mat4 P; // Matrice de projection

void main()
{
    vec4 v = gl_Vertex;
    vec4 pos = vec4(0.0);
    for (int i=0; i<100; i++)
    {
        vec4 v1 = M * v;
        vec4 v2 = V * v1;
        vec4 v3 = P * v2;
        pos += v3;
    }
    gl_Position = pos / 100.0;
}
```

Avec le vertex shader A, la démonstration tourne à **19 FPS**. Avec le vertex shader B, la démonstration tourne à **64 FPS**.
Ouf, trouvé ! L'intuition était bonne : il y a une importante différence de vitesse entre ces deux vertex shaders.

Retrouvez l'article de JeGX en ligne : Lien 73

Comment calculer la position et la normale dans le vertex shader avec OpenGL (GLSL) et Direct3D (HLSL) - Troisième partie

Troisième et dernière partie consacrée au calcul de la position et de la normale dans le vertex shader. Cet article revient et détaille la théorie des transformations d'espace de coordonnées.

1. Les espaces de coordonnées

Après la publication de mes articles précédents sur le **calcul de la position dans le vertex shader** (partie 1, partie 2), j'ai reçu une explication détaillée de la part de Graham sur les espaces de coordonnées et bien sûr sur le calcul de… la position ! Je pense que c'est un texte à partager avec mes lecteurs. J'ai légèrement modifié le texte original pour correspondre aux besoins de l'article, mais l'explication est inchangée.
Les espaces de coordonnées sont une notion importante qui doit être claire pour tout le monde. Globalement, quatre systèmes de coordonnées majeurs coexistent : **Object** (objet), **World** (monde), **View** (vue) et **Clip**.
Nous avons besoin d'une matrice de transformation entre chacun d'entre eux. La matrice de modèle transforme de l'espace objet à l'espace monde, la matrice de vue transforme de l'espace monde à l'espace vue et la matrice de projection (avec la division homogène implicite) transforme la vue en espace de coordonnées clip.
Les vertex entrants (les entrées du vertex shader) sont généralement dans l'espace de coordonnées objet, sauf si nous avions un objet qui est notre image référence (comme un terrain).
Pour l'éclairage, nous avons besoin soit des positions dans l'espace monde, soit dans l'espace vue, mais pas les deux. L'éclairage dans l'espace de coordonnées monde nécessite de connaître la position de la caméra dans l'espace de coordonnées monde, alors que l'éclairage dans l'espace de coordonnées vue nécessite d'avoir la position de la lumière dans l'espace de coordonnées vue. La première solution est généralement plus simple à gérer (une mise à jour par image plutôt qu'une par lumière).
Toutefois, il sera plus efficace de rassembler les matrices. En OpenGL classique, nous aurions eu *gl_ModelViewMatrix*. Cette matrice rassemble la matrice de modèle avec la matrice de vue et transforme les vertex de l'espace de coordonnées objet à l'espace de coordonnées vue directement. Nous avons aussi les matrices *gl_ProjectionMatrix* et *gl_ModelViewProjectionMatrix*, qui transforment les vertex de l'espace de coordonnées vue à l'espace de coordonnées clip et de l'espace de coordonnées objet à l'espace de coordonnées clip respectivement. Il est recommandé d'utiliser des variables *uniforms* pour celles-ci. Notre vertex shader minimal pourrait donc être aussi simple que :

```
uniform mat4 model_view_projection_matrix;

void main (void)
{
    gl_Position = model_view_projection_matrix *
gl_Vertex;
}
```

Une seule multiplication de matrice est suffisante si nous n'avons pas besoin des résultats intermédiaires.
Si nous avons besoin des coordonnées dans l'espace de coordonnées monde ou dans l'espace de coordonnées clip, nous devons au moins faire une transformation de plus. Les vertex shaders suivants sont équivalents :

```
uniform mat4 model_view_matrix;
uniform mat4 projection_marix;
```

```
void main (void)
{
     vec4 view_space_vertex = model_view_matrix *
gl_Vertex;
     gl_Position = projection_matrix *
view_space_vertex;
}
```

```
uniform mat4 model_matrix;
uniform mat4 view_projection_matrix;

void main (void)
{
     vec4 world_space_vertex = model_matrix *
gl_Vertex;
     gl_Position = view_projection_matrix *
world_space_vertex;
}
```

Les deux nécessitent deux multiplications de matrices - que nous devons effectuer selon l'espace de coordonnées dont nous avons besoin (si nous voulons effectuer l'éclairage dans l'espace de coordonnées monde ou dans l'espace de coordonnées vue).
Par contre, il y a, ici, des dépendances. Le second peut être meilleur, car la variable uniforme *model_matrix* changera pour chaque objet affiché alors que la variable uniforme *view_projection_matrix* ne changera probablement qu'une fois par image. De plus, même si cela ne fera certainement pas une grande différence, la seconde multiplication requiert le résultat de la première. Suivant le contenu de la matrice et de la machine sur laquelle le programme tourne, cela peut provoquer des problèmes. Il serait préférable d'aller de l'espace de coordonnées objet à l'espace dans lequel nous voulons être :

```
uniform mat4 model_matrix;
uniform mat4 model_view_matrix;
uniform mat4 model_view_projection_matrix;

void main (void)
{
    vec4 world_space_vertex = model_matrix *
gl_Vertex;
    vec4 view_space_vertex = model_view_matrix *
gl_Vertex;
    gl_Position = model_view_projection_matrix *
gl_Vertex;
}
```

Ici, les trois calculs sont indépendants et plus précis, car il n'y a pas de valeurs intermédiaires. Nous n'avons certainement besoin que de deux des résultats suivant les nécessités de notre algorithme et la façon dont nous rassemblons les matrices.
Finalement, toutes ces transformations sont linéaires et donc leurs résultats peuvent être interpolés. Par exemple, nous voulons la lumière dans l'espace de coordonnées monde. Nous avons la position de notre caméra dans l'espace de coordonnées monde et nous calculons les coordonnées dans l'espace de coordonnées monde de notre vertex pouvant être tous les deux interpolés et, donc, le delta peut être aussi interpolé :

```
uniform mat4 model_matrix;
uniform mat4 model_view_projection_matrix;
uniform vec3 viewer_position; // Position de la
caméra en coordonnées monde
out vec3 view_vector ;

void main (void)
{
     vec4 world_space_vertex = model_matrix *
gl_Vertex;
     view_vector = world_space_vertex.xyz -
viewer_position;
     gl_Position = model_view_projection_matrix *
gl_Vertex;
}
```

Une nouvelle soustraction est apparue dans le vertex shader, mais évite une variable uniforme *model_view_matrix* pour chaque mise à jour de *model_matrix*. C'est particulièrement important si model_matrix est un énorme tableau et que nous utilisons l'instanciation pour indexer dans celui-ci. La soustraction supplémentaire n'est pas très importante, car les **opérations sur une UAL de GPU sont peu coûteuses par rapport aux mêmes opérations sur le CPU et aux mises à jour des variables uniform** : il est plus économique de mettre à jour deux matrices à chaque image qu'une centaine ou plus. Bien sûr, nous aurons peut-être aussi besoin d'une matrice pour les normales. Celle-ci transforme les normales de l'espace de coordonnées objet à l'espace de coordonnées vue et peut être donc dérivée à partir des matrices de modèle et de vue.

Retrouvez l'article de JeGX en ligne : Lien 74

Liens

Lien 01 : https://itunes.apple.com/fr/app/os-x-mountain-lion/id537386512?mt=12
Lien 02 : http://www.developpez.net/forums/d1320828/mac/mise-a-jour-1083-pour-os-x-mountain-lion-enfin-disponible/
Lien 03 : http://jumperbuggy.virtual-vision.net/3.html
Lien 04 : https://itunes.apple.com/us/app/camera-art-effects/id429811521?mt=8
Lien 05 : http://www.arteffects.virtual-vision.net/
Lien 06 : http://jumperbuggy.virtual-vision.net/
Lien 07 : http://scientificninja.com/blog/write-games-not-engines
Lien 08 : http://jeux.developpez.com/tutoriels/ios-opengl-es-2/fichiers/tutorials.zip
Lien 09 : http://www.turbosquid.com/
Lien 10 : http://habibs.wordpress.com/lake/
Lien 11 : http://www.imgtec.com/powervr/insider/sdkdownloads/
Lien 12 : http://en.wikipedia.org/wiki/Blinn-Phong_shading_model
Lien 13 : http://jeux.developpez.com/tutoriels/ios-opengl-es-2/
Lien 14 : http://fakeimg.pl/
Lien 15 : http://www.paulund.co.uk/fake-images-please#more-6799
Lien 16 : http://paulund.developpez.com/tutoriels/html/images-types/
Lien 17 : http://blip.tv/bogdan-vatra/qt-tooltips-example-on-android-3140576
Lien 18 : https://groups.google.com/forum/#%21msg/android-qt/IJ7e98XO7Io/_ti9asJPEsAJ
Lien 19 : http://blog.qt.digia.com/blog/2012/11/08/necessitas-android-port-contributed-to-the-qt-project/
Lien 20 : http://www.youtube.com/watch?v=TzNWqoJRFsc
Lien 21 : http://www.youtube.com/watch?v=tmy7GVZEoj8
Lien 22 : http://www.youtube.com/watch?v=odiSw-i0_4o
Lien 23 : http://www.developpez.com/actu/52541/Android-4-0-devient-plus-populaire-que-Gingerbread-qui-a-la-peau-dure-Jelly-Bean-enregistre-une-forte-adoption/
Lien 24 : http://www.youtube.com/watch?v=1Y8rFlvtAOA
Lien 25 : http://qt-project.org/wiki/Qt5ForAndroid
Lien 26 : http://blog.qt.digia.com/blog/2013/03/13/preview-of-qt-5-for-android/
Lien 27 : http://www.developpez.net/forums/d1320036/mobiles/android-avancee-du-support-dandroid-dans-qt-51/
Lien 28 : http://qt.developpez.com/actu/51464/Qt-Creator-2-6-apporte-les-kits-cette-seconde-mise-a-jour-corrige-des-blocages/
Lien 29 : http://blog.qt.digia.com/blog/2013/02/07/qt-creator-2-7-0-beta-released/
Lien 30 : http://releases.qt-project.org/qtcreator/
Lien 31 : http://www.developpez.net/forums/d1306881/qt-creator/sortie-qt-creator-270/
Lien 32 : http://www.developpez.net/forums/d1316421/mobiles/ios-avancee-du-support-dios-dans-qt-51/
Lien 33 : http://10fastfingers.com/typing-test/french
Lien 34 : http://fr.wikipedia.org/wiki/Disposition_des_touches_des_claviers_informatiques
Lien 35 : http://www.essaytyper.com/
Lien 36 : http://hackertyper.net/
Lien 37 : http://keybr.com/
Lien 38 : http://vviale.developpez.com/tutoriels/apprendre-a-taper/fichiers/www.sense-lang.org%20
Lien 39 : http://vviale.developpez.com/tutoriels/apprendre-a-taper/fichiers/www.typingweb.com
Lien 40 : http://www.typingtest.com/
Lien 41 : http://vviale.developpez.com/tutoriels/apprendre-a-taper/fichiers/%20www.dactylocours.com
Lien 42 : http://www.typingtutor-online.com/EN/Aspx/Start.aspx
Lien 43 : http://klavaro.sourceforge.net/
Lien 44 : http://www.rapidtyping.com/features.html
Lien 45 : http://www.typefastertypingtutor.com/
Lien 46 : http://tux4kids.alioth.debian.org/tuxtype/index.php
Lien 47 : http://bepo.fr/wiki/Accueil
Lien 48 : http://www.dvorak-keyboard.com/
Lien 49 : http://vviale.developpez.com/tutoriels/apprendre-a-taper/
Lien 50 : http://visioo-writer.tuxfamily.org/FR/index.html
Lien 51 : http://portableapps.com/
Lien 52 : http://www.openoffice.org/fr/Documentation/Outils/ChiffresLettres.sxc
Lien 53 : http://www.dicollecte.org/grammalecte/telecharger.php
Lien 54 : http://www.languagetool.org/
Lien 55 : http://www.druide.com/a_description.html
Lien 56 : http://www.synapse-fr.com/correcteur_orthographe_grammaire.htm
Lien 57 : http://openoffice-libreoffice.developpez.com/faq/
Lien 58 : http://openjdk.java.net/projects/jdk6/
Lien 59 : http://www.developpez.net/forums/d1319421/general-java/red-hat-continuera-a-maintenir-jdk-6-dans-cadre-du-projet-openjdk-6/
Lien 60 : http://armel-ndjobo.developpez.com/memocertifjava6/
Lien 61 : http://armel-ndjobo.developpez.com/tutoriels/java/chap2-programmation-orientee-objet/
Lien 62 : http://www.developpez.net/forums/d1320057/android/google-sattaque-aux-applications-antipub/
Lien 63 : http://www.developpez.net/forums/d1272607-2/android/fuite-fonctionnalites-dandroid-42/
Lien 64 : http://www.tutos-android.com/wp-content/uploads/2012/04/TutoFragments.zip
Lien 65 : http://nbenbourahla.developpez.com/tutoriels/android/view-pager/
Lien 66 : http://www.vogella.com/articles/AndroidGestures/article.html
Lien 67 : http://nbenbourahla.developpez.com/tutoriels/android/multitouch-fragments/
Lien 68 : http://www.geeks3d.com/geexlab/code_samples.php
Lien 69 : http://www.geeks3d.com/geexlab/
Lien 70 : http://www.ozone3d.net/redirect.php?id=601
Lien 71 : http://geeks3d.developpez.com/calcul-vertex-normal-glsl-part1/
Lien 72 : http://jeux.developpez.com/telecharger/detail/id/3188/GPU-ShaderAnalyzer
Lien 73 : http://geeks3d.developpez.com/calcul-vertex-normal-glsl-part2/
Lien 74 : http://geeks3d.developpez.com/calcul-vertex-normal-glsl-part3/