Multitexturing

Un dels plugins que et proporcionem és **multitex.zip**. El que fa aquest plugin és, bàsicament:

- Al mètode **onPluginLoad**, crea un VS i un FS senzills per usar textures. El VS calcula de forma automàtica coordenades de textura (s,t) a partir de les coordenades dels vèrtexs i del radi d'una esfera contenidora del model (el radi el rep amb un uniform). El FS simplement assigna al color del fragment el color mig de dues textures (sampler0 i sampler1). El mètode també té el codi necessari per carregar dues imatges (obrirà un diàleg per que trieu les imatges) i definir un parell de textures. La classe QImage de Qt simplifica molt aquesta tasca. Observa que el codi està carregant la primera textura a la texture unit 0, i la segona a la texture unit 1 (crides **glActiveTexture**). Els identificadors OpenGL de les textures (textureId0,textureId1) són atributs de la classe, ja que més endavant seran necessaris per activar aquestes textures. La funció més rellevant és **glTexImage2D**, la qual defineix una textura a partir d'un buffer que conté les dades de la imatge que hem carregat:

```
glTexImage2D( GL_TEXTURE_2D, 0, GL_RGB, image.width(), image.height(), 0,
              GL_RGBA, GL_UNSIGNED_BYTE, image.bits());
```

- Al mètode **preFrame**, s'activen el VS i el FS, i es defineixen els uniforms que contenen els texture units de cada sampler:

```
program->setUniformValue("sampler0", 0); // texture unit del primer sampler
program->setUniformValue("sampler1", 1);  // texture unit del segon  sampler
```

També s'activen les textures corresponents:

```
glActiveTexture(GL_TEXTURE0);  // texture unit 0
glBindTexture(GL_TEXTURE_2D, textureId0);
glActiveTexture(GL_TEXTURE1); // texture unit 1
glBindTexture(GL_TEXTURE_2D, textureId1);
```

En aquest exercici només et demanem que descomprimeixis el zip corresponent, el compilis, i el provis amb el viewer. Hauries d'obtenir imatges similars a aquestes (depenent de les textures que carregueu):

Texture Splatting (plugin+shaders)

Escriu un **plugin** (a partir del multitex) que generi un efecte similar al que us demanàvem a l'exercici **Texture Splatting**.

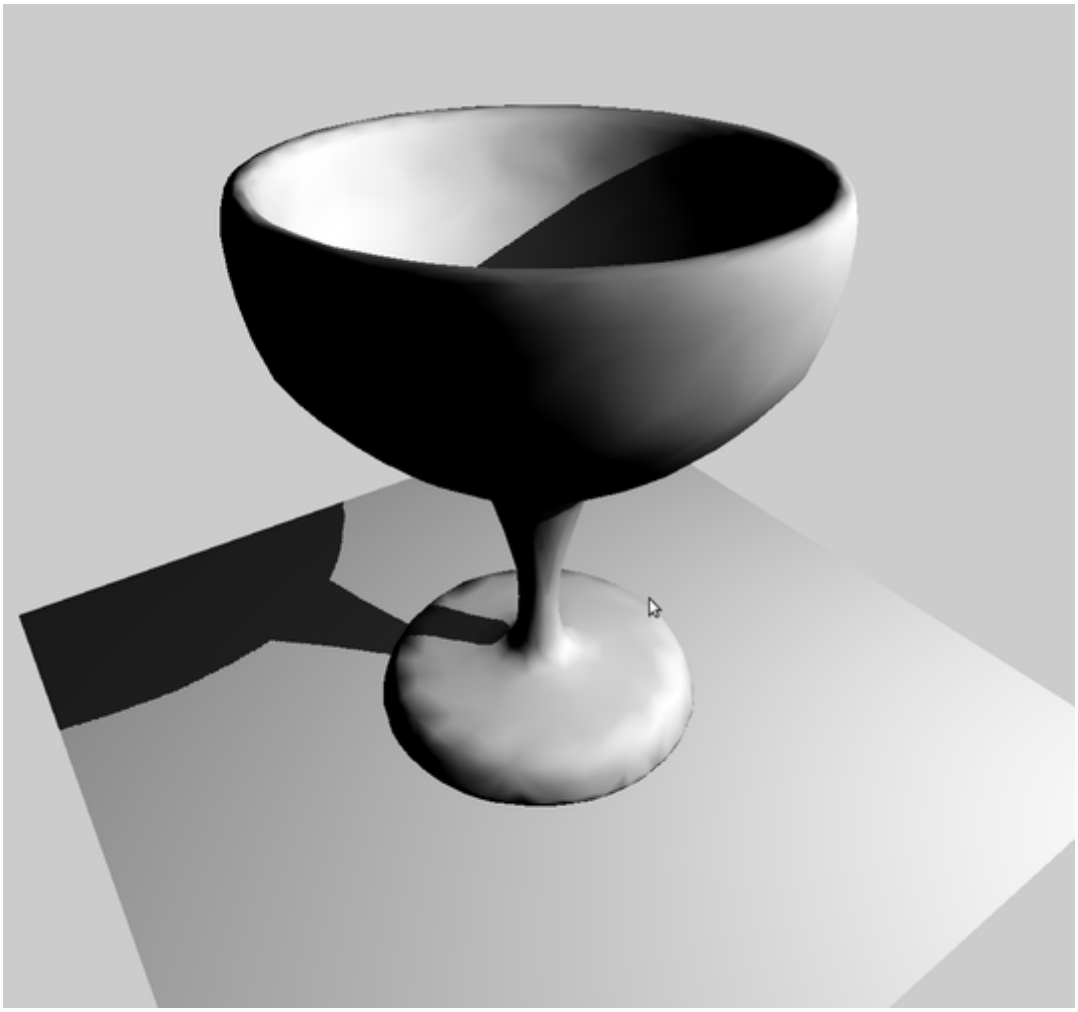
Observa que el VS haurà de calcular coordenades de textura d'alguna manera, ja que els plugins del viewer en principi no envien a OpenGL coordenades de textura (podeu mantenir el VS del multitex). A diferència del multitex, ara caldrà carregar **tres textures diferents**.

Aquí teniu un exemple:

ShadowMapping

Un dels plugins que et proporcionem és **shadowmap.zip**.

En aquest exercici només et demanem que descomprimeixis el zip corresponent, el compilis, i el provis amb el viewer. Hauries d'obtenir imatges similars a aquestes:



Glowing

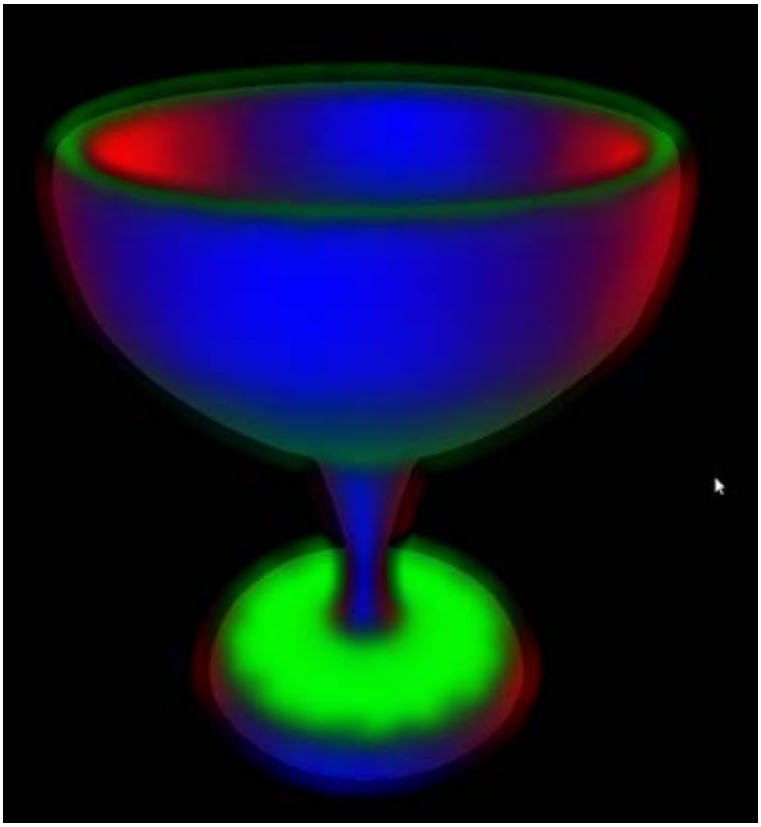
Un altre **plugin** que et proporcionem és **glowing**. Aquest plugin és un exemple de tècnica que requereix **múltiples passos de rendering**. En aquest cas, el mètode **paintGL** que implementa el plugin fa bàsicament aquests passos:

1. Pinta l'escena normalment (glClear, drawScene...), i **guarda el contingut del buffer de color en una textura** que hem inicialitzat prèviament. D'aquesta manera la textura té una còpia de la imatge corresponent al frame actual.

2. Activa un VS i un FS que implementen un efecte de glowing (el color de cada punt es veu afectat per la brillantor dels veïns), i dibuixa un rectangle que omple tot el viewport (amb vèrtexs en clip space de -1 a 1). Per a cada fragment, el FS consultarà diferents texels de la textura per produir aquest efecte.

En general, per dibuixar quelcom en una textura és més eficient utilitzar Frame Buffer Objects (FBOs). En aquest exemple però, per tal de fer-ho més senzill, hem dibuixat igualment al *on-screen* frame buffer, i després hem utilitzat **glCopyTexSubImage2D** per copiar el contingut del buffer de color a una textura que hem inicialitzat prèviament. Amb el mateix objectiu de fer l'exemple senzill, veureu que com es carrega el plugin, la finestra OpenGL es canvia a una mida potència de dos (512x512, 1024x1024...).

En aquest exercici només et demanem que descomprimeixis el zip corresponent, el compilis, i el provis amb el viewer. Per tal que funcioni, hauràs d'editar el **glowing.cpp**, i canviar la línia amb el path dels fitxers **glowing.vert** i **glowing.frag**. Hauries d'obtenir imatges com aquesta:



Distort

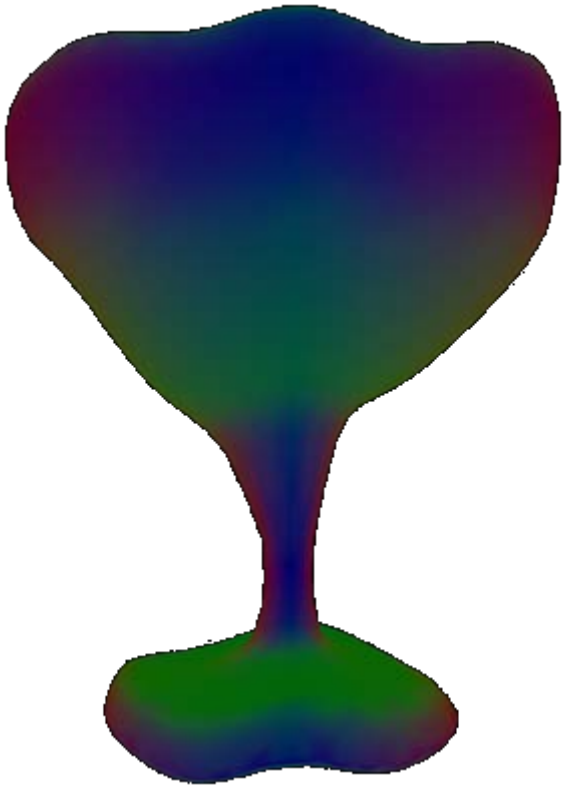
Prenent com a punt de partida el plugin de glowing, escriu un plugin que, amb un VS i un FS apropiats, aconsegueixi un efecte d'ones aplicat a la imatge.

Per aconseguir aquest efecte, caldrà dos passos de rendering.

Al primer pas, es dibuixarà l'escena normalment per obtenir una textura.

Al segon pas, es dibuixarà un rectangle amb un FS que calcularà el color final accedint a la textura amb coordenades de textura (s,t) alterades afegint-hi un offset 0.01*sin(10.0*time + 30.0*s).

Aquí teniu un exemple aproximat del resultat esperat:



Reflection

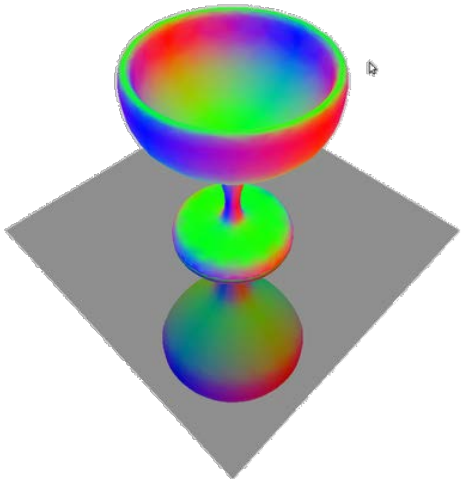
Escriu un plugin (prenent com a base el glowing) que implementi la tècnica de reflexió amb textures dinàmiques. Com a mirall, podeu fer servir una de les cares de la capsa contenidora de l'escena. Caldrà fer tres tasques al paintGL:

1. Dibuixar l'escena reflectida respecte el pla del mirall. Guardar la imatge resultant en una textura i esborrar el framebuffer.

2. Esborrar el framebuffer i dibuixar l'escena en la posició real.

3. Dibuixar el quad del mirall, texturat amb la textura del primer pas.

Aquí teniu alguns exemples:



Observa que l'objecte reflectit només apareix a la porció del pla ocupada pel mirall: