Εργασία 3

στο μάθημα "Γραφικά με Υπολογιστές"

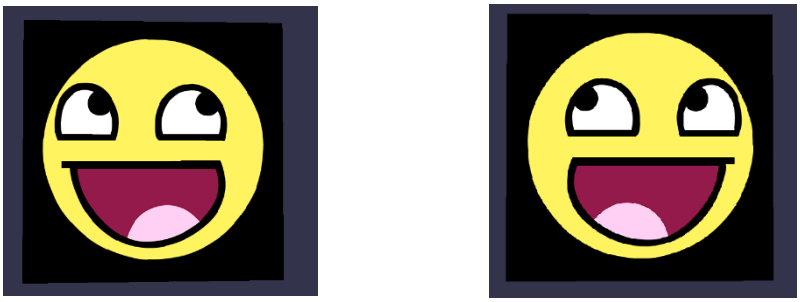
Καταληκτική ημερομηνία παράδοσης : **Κυριακή 09 Μαΐου 2021, ώρα 23:55**

Άσκηση 1

Να δημιουργηθεί αρχείο κώδικα (.cpp) με shaders κατά το οποίο θα παράγεται ένα παράθυρο OpenGL στο οποίο θα εμφανίζονται τα εξής:

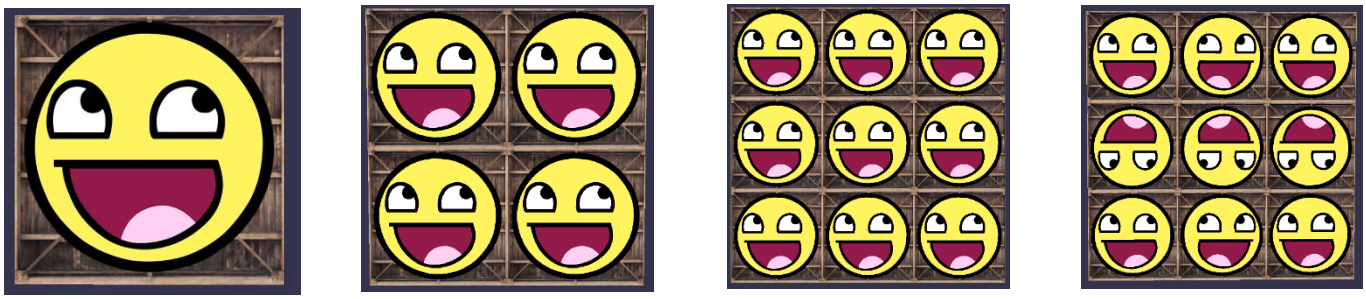
Μία κατευθυντική φωτεινή πηγή με κατεύθυνση (0.0f, 0.0f, -1.0f) και θέση (0.0f, 0.0f, 5.0f) και **τρεις** κύβοι με διαφορετικά textures όπως περιγράφονται στη συνέχεια :

Ο **'ΚΥΒΟΣ 1'** να βρίσκεται στη θέση (-2.0f, 0.0f, 0.0f), χωρίς κάποια περιστροφή. Επάνω σε αυτόν να εφαρμοστεί το 'awesameface.png' texture και με το πάτημα του **πλήκτρου '1'** να αλλάζει η κατεύθυνση που κοιτάει το πρόσωπο κάνοντας την κατάλληλη αλλαγή στα *textureCoordinates* στον fragment shader (βλ. Σχήμα 1).



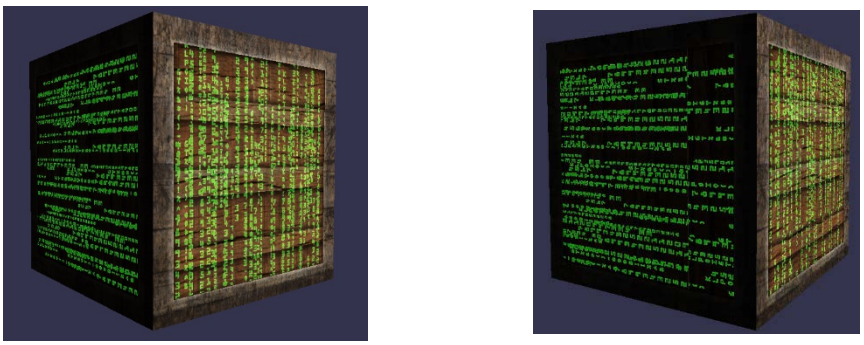
Σχήμα 1: Αναμενόμενο αποτέλεσμα του σχεδίου εφαρμογής για το αντικείμενο 'ΚΥΒΟΣ 1'

Ο **'ΚΥΒΟΣ 2'** να βρίσκεται στη θέση (0.0f, 0.0f, 0.0f) χωρίς κάποια περιστροφή. Επάνω σε αυτόν να εφαρμοστεί η μίξη του 'container.jpg' με το 'awesameface.png' με τέτοιο τρόπο ώστε το 'awesameface.png' να βρίσκεται πάνω στο 'container.jpg'. Για κάθε πάτημα του **πλήκτρου '2'** να εμφανίζεται μία νέα γραμμή και μία νέα στήλη από πρόσωπα. Για κάθε πάτημα του **πλήκτου '3'** να γίνεται το αντίθετο δηλαδή να αφαιρείται μία στήλη και μία γραμμή από πρόσωπα (για να γίνει αυτό θα πρέπει οι παράμετροι GL_TEXTURE_WRAP_S και GL_TEXTURE_WRAP_T να έχουν τιμή GL_REPEAT). Τέλος με το πάτημα του **πλήκτρου '4'** να εναλλάσσεται το mode από GL_REPEAT σε GL_MIRRORED_REPEAT και για το GL_TEXTURE_WRAP_S και για το GL_TEXTURE_WRAP_T (βλ. Σχήμα 2).



Σχήμα 2: Αναμενόμενο αποτέλεσμα του σχεδίου εφαρμογής για το αντικείμενο 'ΚΥΒΟΣ 2'

Ο 'ΚΥΒΟΣ 3' να βρίσκεται στη θέση (2.0f, 0.0f, 0.0f) χωρίς κάποια περιστροφή. Επάνω σε αυτόν να εφαρμοστεί το 'container2.png' για το ambient και diffuse Color και το 'container2_specular.png' για το specular Color. Επίσης να εφαρμοστεί και το 'matrix.jpg' (φάκελος Task3Images στο e-class) το οποίο δεν θα επηρεάζετε από τον φωτισμό (θα φαίνεται στο σκοτάδι) και να σχεδιαστεί πάνω στο ξύλο αλλά όχι στην μεταλική επιφάνεια (μόνο εκεί όπου το 'container2_specular.jpg' είναι μαύρο, δηλαδή οι r,g,b, τιμές είναι μηδέν). Με το **πλήκτρο '5'** το 'matrix.jpg' θα πρέπει να ξεκινάει να κινείται στον άξονα t των υφών και αν πατηθεί ξανά το ίδιο πλήκτρο να σταματάει να κινείται. Όταν το texture ξεκινάει να κινείται θα πρέπει να συνεχίζει από το σημείο που ήταν πριν (βλ. Σχήμα 3).



Σχήμα 3: Άναμενόμενο αποτέλεσμα του σχεδίου εφαρμογής για το αντικείμενο 'ΚΥΒΟΣ 3'

Υποσημείωση 1: Όλες οι είσοδοι από το πληκτρολόγιο θα πρέπει να είναι toggle on και off, δηλαδή αν ένα από τα πλήκτρα συνεχίζει να είναι πατημένο δεν θα δίνεται είσοδος σε κάθε frame.

Υποσημείωση 2: Μπορείτε να δημιουργήσετε όσους shaders και όσα shader programs κρίνεται εσείς ότι χρειάζονται.

Το αρχείο κώδικα της άσκησης 1 θα έχει όνομα "E3_A1_AM.cpp", όπου AM θα είναι ο αριθμός μητρώου σας. Ο/Οι vertex shader/s θα έχει/ουν όνομα "E3_A1_VertexShaderX.txt" και ο/οι fragment shader/s θα έχει/ουν όνομα "E3_A1_FragmentShaderX.txt" (όπου X ο αριθμός του shader, 1 για τον πρώτο, 2 για τον δεύτερο κ.τ.λ.π.).

Άσκηση 2

Να δημιουργηθεί αρχείο κώδικα (.cpp) με shaders κατά το οποίο θα παράγεται ένα παράθυρο OpenGL στο οποίο θα εμφανίζονται τα εξής:

Η σκηνή όπως ακριβώς είναι στο 7ο εργαστήριο με κάποιες όμως αλλαγές. Να χρησιμοποιηθεί το 'Water.jpg' και το 'Water_normal.jpg' (φάκελος Task3Images στο e-class). Να υπάρχει η δυνατότητα να ενεργοποιείται και να απενεργοποιείται το normal mapping με την χρήση του **πλήκτρου '1'**. Επίσης και στα δύο textures να εφαρμόζετε μία εξίσωση για την δημιουργία κυμάτων. Για την επίτευξη αυτού του σκοπού να χρησιμοποιηθεί ο παρακάτω κώδικας στην main() του fragment shader :

```
vec2 cPos = -1.0 + 2.0 * gl_FragCoord.xy / resolution.xy;
float cLength = length(cPos);

vec2 uv = gl_FragCoord.xy/resolution.xy+(cPos/cLength)*cos(cLength*12.0-time*4.0)*0.01;
```

Η μεταβλητή ‘resolution’ είναι μία uniform vec2 μεταβλητή που παίρνει την τιμή glm::vec2(800.0f, 600.0f). Η μεταβλητή ‘time’ είναι μία uniform float και παίρνει την τιμή glfwGetTime(). Στη συνέχεια να αντικαταστήσετε τις εντολές :

- vec3 normal = texture(normalMap, fragmentInput.textureCoordinates).rgb;
- vec3 color = texture(diffuseMap, fragmentInput.textureCoordinates).rgb;

με τις εντολές :

- vec3 normal = texture(normalMap, vec2(uv.s, uv.t)).rgb;
- vec3 color = texture(diffuseMap, vec2(uv.s, uv.t)).rgb;

Το αρχείο κώδικα της άσκησης 2 θα έχει όνομα "E3_A2_AM.cpp", όπου ΑΜ θα είναι ο αριθμός μητρώου σας. Ο vertex shader θα έχει όνομα "E3_A2_VertexShader.txt" και ο fragment shader θα έχει όνομα "E3_A2_FragmentShader.txt".

Οι shaders θα πρέπει να τοποθετηθούν σε ένα φάκελο με όνομα "Shaders_AM", όπου ΑΜ ο αριθμός μητρώου σας. Τα αρχεία κώδικα και ο φάκελος των shaders να τοποθετηθούν σε έναν φάκελο με όνομα "Ergasia3_AM", ο οποίος θα συμπιεσθεί σε αρχείο (.rar, .zip) και θα υποβληθεί στο e-class. Φροντίστε ώστε τα ονόματα των path των shaders στα αρχεία κώδικα να είναι έτσι ώστε το πρόγραμμα να τρέχει απευθείας όταν φορτώσουμε το αρχείο κώδικα της κάθε άσκησης και το φάκελο "Shaders_AM" στο project που έχουμε δημιουργήσει.