Σολομωνίδη Ελευθερία

up1092837

«Γραφικά και Εικονική Πραγματικότητα»

Project : **Pop – Up World**

***Figure 1**: Κόσμος παιχνιδιού*

Εισαγωγή

Η ιδέα της εργασίας είναι η δημιουργία ενός παιχνιδιού τύπου platform game level με ένα κεντρικό avatar ως παίκτη, την κίνηση του οποίου ελέγχει ο χρήστης. Σε αυτό το παιχνίδι ο παίκτης είναι μία γάτα που προχωρά σε ένα δάσος με στόχο να μπορέσει να φτάσει στο σπιτάκι που βρίσκεται στο τέλος της πλατφόρμας. Υπάρχει η δυνατότητα ο χρήστης να μεταβεί από την αρχική 3D σκηνή σε 2D και αντίστροφα κάνοντας την κάμερα να φαίνεται σαν να εκτελεί μια απότομη αλλά ομαλή περιστροφή. Το χρώματα του παιχνιδιού μπορούν να μεταβληθούν και πιο συγκεκριμένα να κβαντιστούν σε συγκεκριμένες τιμές ώστε η αισθητική του να προσεγγίζει περισσότερο αυτή ενός cartoon – like παιχνιδιού.

3D Σκηνή και Περιβάλλον του παιχνιδιού

Αρχικά για να δημιουργηθεί η πίστα και να φαίνεται αισθητικά όμορφο το παιχνίδι προστέθηκαν objects με τα κατάλληλα textures και ένα skybox που περικλείει την σκηνή ώστε να δημιουργηθεί η ψευδαίσθηση ότι το περιβάλλον του παιχνιδιού δεν έχει όρια (fig.2).

***Figure 2**: Skybox*

Ορισμένα αντικείμενα όπως τα σύννεφα και κάποια από τα μανιτάρια τοποθετήθηκαν σε τυχαίες θέσεις ώστε η σκηνή να φαίνεται πιο φυσική με λιγότερα μοτίβα επανάληψης. Για να γίνει αυτό χρησιμοποιήθηκε η συνάρτηση rand() και οριοθετήθηκε στα πλαίσια της σκηνής.

Όσον αφορά το skybox έγινε χρήση ενός cubemap δηλαδή μιας τεχνικής που περιλαμβάνει 6 διαφορετικά 2D textures κάθε ένα από τα οποία αναπαριστά μια διαφορετική όψη ενός κύβου (fig.3). Έτσι η σκηνή φαίνεται να περιβάλλεται από ένα δυναμικό ουρανό που δίνει την αίσθηση βάθους. Για το skybox χρησιμοποιήθηκαν διαφορετικοί shaders (συγκεκριμένα τα αρχεία: *Skybox.fragmentshader* και *Skybox.vertexshader*) σε σχέση με το υπόλοιπο πρόγραμμα και αυτό γιατί θέλουμε να μένει ανεπηρέαστο από τους φωτισμούς και σκιές που χρησιμοποιούνται στην υπόλοιπη σκηνή.

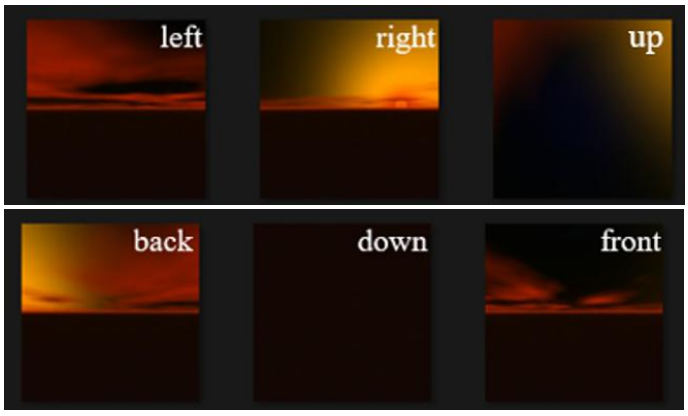

***Figure 3**: 6 jpgs για κάθε όψη του κύβου*

Κίνηση avatar και κάμερας

Το avatar υλοποιεί δύο κινήσεις, συγκεκριμένα μπορεί να προχωράει ευθεία και να κάνει άλμα. Αυτά υλοποιούνται με τα πλήκτρα W και Space αντίστοιχα όπως φαίνεται και στο βοηθητικό παράθυρο πάνω αριστερά στην εικόνα (fig.1). Για να μπορεί το avatar να κινείται πάνω στις πλατφόρμες έχουν γίνει οι κατάλληλοι έλεγχοι στις διαστάσεις κάθε πλατφόρμας ανάλογα με το scaling που έχουν υποστεί στους άξονες y,z και την θέση τους στο χώρο. Αν για κάποιο λόγο το avatar βρεθεί εκτός πλατφόρμας κατευθείαν πέφτει αφού δέχεται την δύναμη της βαρύτητας.

Επίσης το avatar κινείται πάντα σε σχέση με την κάμερα. Και τα δύο βρίσκονται στο ίδιο επίπεδο κατά μήκος του άξονα x (x = 0), αλλά έχουν διαφορετικές συντεταγμένες στους άξονες y και z. Συγκεκριμένα, η διαφορά τους στον άξονα y είναι σταθερή και ίση με h, ενώ στον άξονα z απέχουν απόσταση ρ. Αυτές οι αποστάσεις παραμένουν σταθερές κατά την κίνησή τους. Έτσι, η θέση του avatar μπορεί να θεωρηθεί ως ένα translation της θέσης της κάμερας.

Περιστροφή κάμερας 3D ⇔ 2D views

Υποθέτουμε ότι βλέπουμε την σκηνή από ψηλά (fig.4).

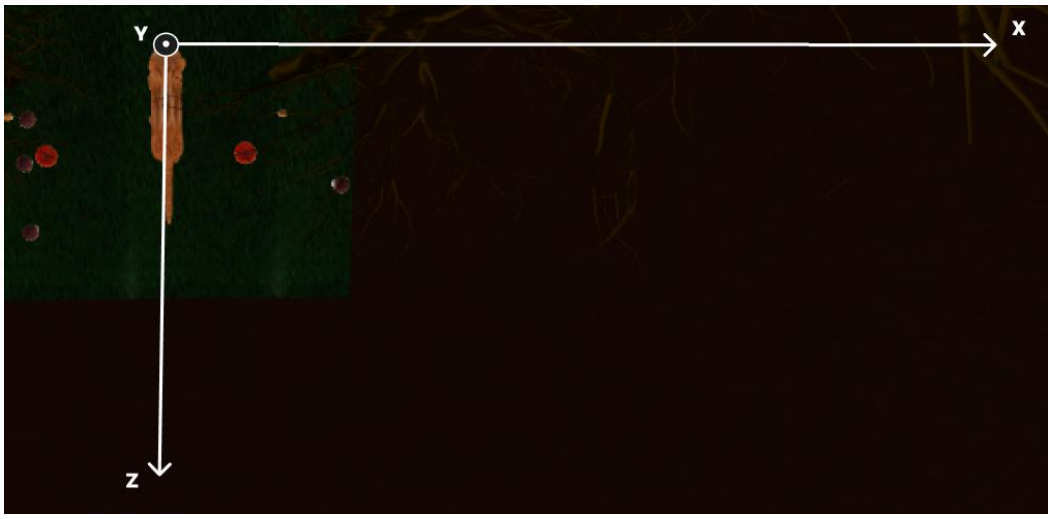

***Figure 4**: Απεικόνιση σκηνής από ψηλά*

Για να μπορέσει να μεταβεί η κάμερα από **3D** σε **2D** προβολή αρκεί να περιστρέψουμε την κάμερα κατά 90° γύρω από τον y άξονα σταδιακά με την πάροδο του χρόνου ώστε να φαίνεται συνεχής η περιστροφή. Η κάμερα θέλουμε δηλαδή να διαγράψει μια ελλειπτική τροχιά στο επίπεδο xz (fig.5). Μετά την περιστροφή, θέλουμε η κάμερα και το avatar να ευθυγραμμιστούν κατά μήκος του άξονα x. Επίσης πρέπει η κάμερα να βρίσκεται στο ύψος y = 0 και να έχει μετακινηθεί στον άξονα z, διανύοντας μια απόσταση που είναι ανάλογη της αρχικής σταθερής απόστασης ρ (α: ακέραιος).

Η κάμερα ξεκινά από τη θέση **CamPos** και καταλήγει στη θέση **CamPos'**, ενώ το **AvatarPos** αντιπροσωπεύει τη σταθερή θέση του άβαταρ στον χώρο. Η γωνία θ μεταβάλλεται με τον χρόνο στο διάστημα: $\theta \in [0,\pi/2]$ καθορίζοντας την καμπύλη κίνηση της κάμερας. Το σημείο **MiddleCamPos** θα αποτελεί το μεταβλητό με τη

γωνία (και άρα με τον χρόνο) διάνυσμα της θέσης της κάμερας. Το διάνυσμα **direction** υπολογίζεται ως **AvatarPos – MiddleCamPos.**

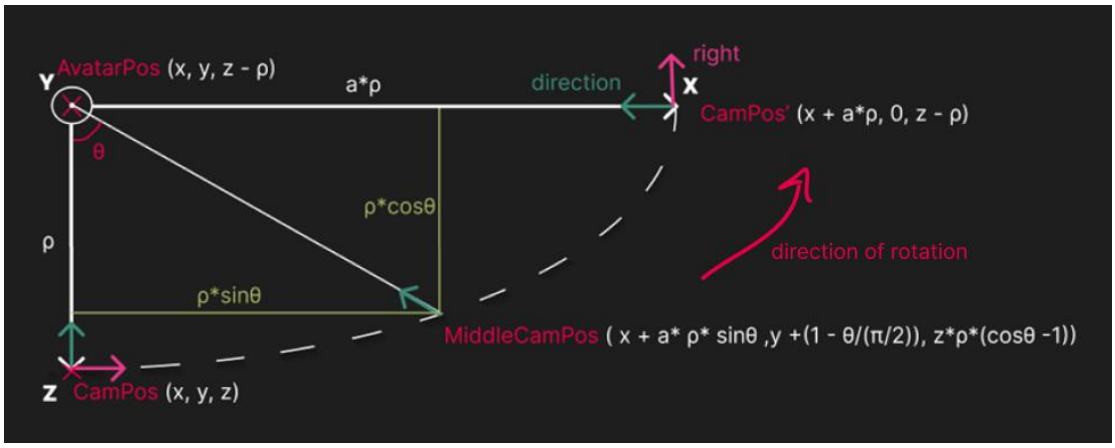

***Figure 5**: Γεωμετρική αναπαράσταση τροχιάς της κάμερας*

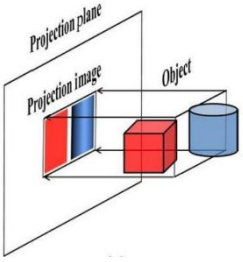

***Figure 6**: Ορθογραφική Προβολή*

Για να φαίνεται όμως πραγματικά σαν μια 2D προβολή χρησιμοποιήθηκε ορθογραφική προβολή για την κάμερα έτσι ώστε τα αντικείμενα να φαίνονται όλα ότι βρίσκονται στο ίδιο βάθος. Το μέγεθος ενός αντικειμένου στην εικόνα που αποδίδεται παραμένει σταθερό ανεξάρτητα από την απόστασή του από την κάμερα (fig.6).

Θεωρώντας την πρώτη εικόνα (fig.1) ως την 3D προβολή, παρακάτω (fig.7) φαίνεται η πίστα αφού έχει γίνει η περιστροφή της κάμερας δηλαδή όταν πλέον είμαστε σε 2D προβολή.

***Figure 7**: 2D προβολή*

Υλοποιήθηκε και η αντίστροφη διαδικασία δηλαδή βρισκόμενοι σε 2D προβολή να πάμε σε 3D. Η λογική είναι ίδια μόνο που σε αυτή την περίπτωση η γωνία θ θα μειώνεται μέχρι να προσεγγίσει το 0. Η κάμερα θα επιστρέψει στην αρχική της θέσης δηλαδή στο x = 0 και σε σταθερή απόσταση στους y, z από το avatar. Κατά την περιστροφή η κάμερα θα κινείται προς τα θετικά z και αρνητικά x. Το διάνυσμα θέσης της κάμερας κατά την περιστροφή θα δίνεται από τον τύπο **MiddleCamPos : (x - a*ρ*(1 - sinθ), y + (1 – θ / (π/2)) , z + ρ*cosθ).**

Cartoon-Like λειτουργία

Αρχικά επειδή χρησιμοποιήθηκε διαφορετικός shader για το skybox η διαδικασία που έγινε για την αλλοίωση των χρωμάτων της σκηνής μέσα στον fragment shader εφαρμόστηκε και στον fragment shader για το skybox. Γνωρίζουμε ότι το fragmentColor που δίνει την τιμή των χρωμάτων (rgb & alpha) για κάθε συνιστώσα του διανύσματος παίρνει οποιαδήποτε τιμή μεταξύ των floats 0.0 και 1.0. Για να υλοποιηθεί η cartoon-like λειτουργία χωρίζω αυτό το διάστημα σε 4 ίσα υποδιαστήματα [0.0, 0.25], [0.25, 0.5], [0.5, 0.75] και [0.75, 1.0] και κβαντίζω τις τιμές, δηλαδή όλες οι τιμές που ανήκουν στο ίδιο υποδιάστημα αντιστοιχίζονται σε μια κοινή, σταθερή τιμή. Οι τιμές που έχουν οριστεί για τα διαστήματα είναι οι 0.0, 0.3, 0.6, 1.0 αντίστοιχα. Οι ακραίες τιμές (0.0, 1.0) επιλέχθηκαν επειδή είναι καλύτερη η χρωματική αντίθεση. Η λειτουργία αυτή ενεργοποιείται αυτόματα όταν ο χρήστης πατάει το πλήκτρο 2 στο πληκτρολόγιο του για να μεταβεί στην 2D λειτουργία αλλά μπορεί και να ενεργοποιεί ή απενεργοποιεί την λειτουργία πατώντας το πλήκτρο 7, όπως φαίνεται και στο Helper Window (fig.1). Οι παρακάτω εικόνες δείχνουν την αλλοίωση των χρωμάτων σε 3D και 2D προβολές (fig.8, fig.9).

***Figure 8**:Cartoon-like σε 3D προβολη*

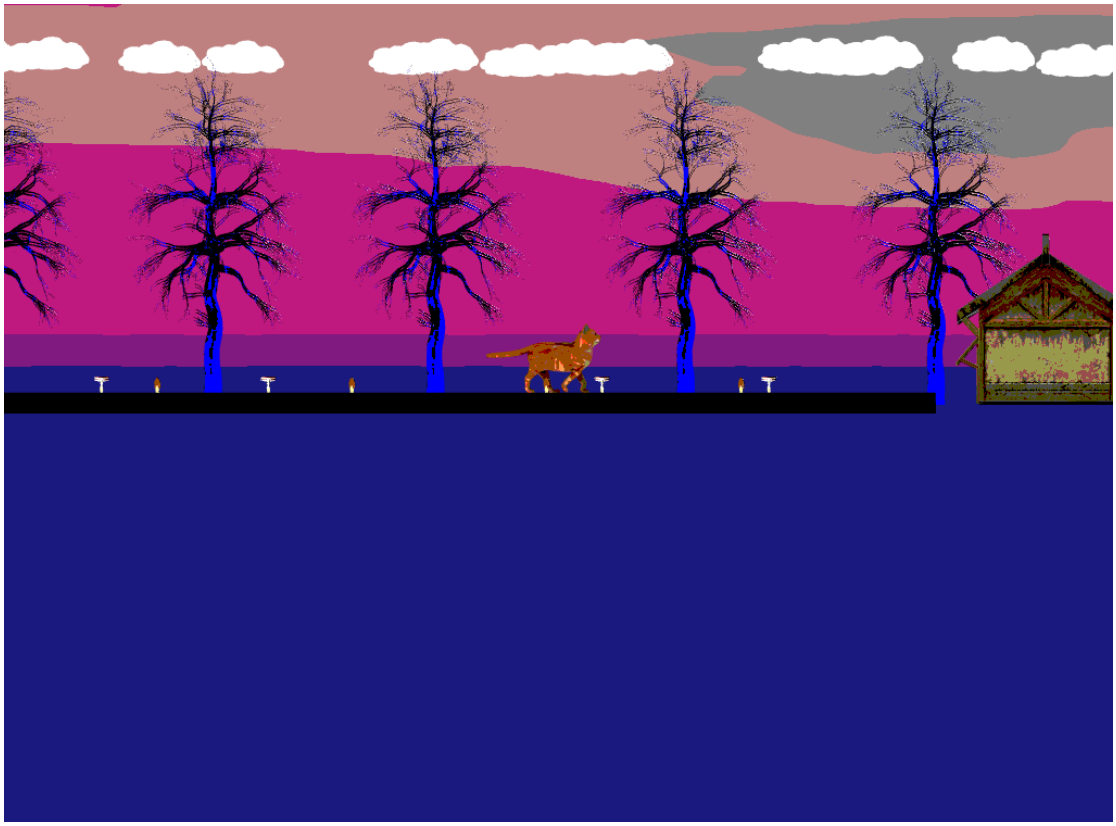

***Figure 9**:Cartoon-like σε 2D προβολη*

Μοντέλο Φωτισμού

Στο παιχνίδι χρησιμοποιήθηκε το μοντέλο φωτισμού Phong με ένα directional light. Πιο συγκεκριμένα το μοντέλο αυτό χρησιμοποιείται για να προσομοιώσει τον τρόπο με τον οποίο το φως αλληλοεπιδρά με τις επιφάνειες. Το συνολικό χρώμα ενός pixel στο μοντέλο του Phong δίνεται από τον τύπο: I=Ia+Id+Is όπου Ia, Id, Is οι 3 συνιστώσες του φωτός και δίνονται από τους τύπους (I), (II), (III). Η Ia είναι η ambient συνιστώσα και αντιπροσωπεύει το φως που διαχέεται στη σκηνή και φωτίζει όλες τις επιφάνειες εξίσου. Η Id είναι η diffuse συνιστώσα και προσομοιώνει το φως που χτυπά κατευθείαν σε μια επιφάνεια και διαχέεται προς όλες τις κατευθύνσεις. Όσο πιο κάθετα πέφτουν οι ακτίνες του φωτός τόσο πιο φωτεινό φαίνεται το αντικείμενο. Τέλος Is είναι η specular συνιστώσα και είναι αυτή που ευθύνεται για τις

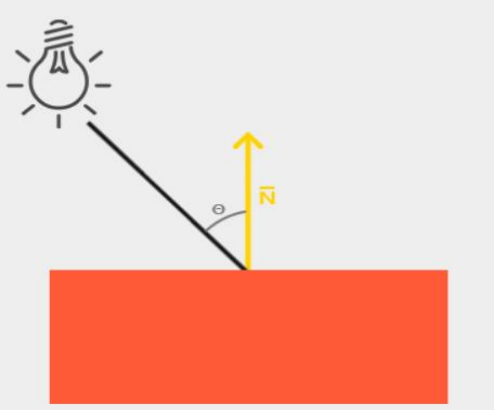

***Figure 10**: Διάχυτος φωτισμός*

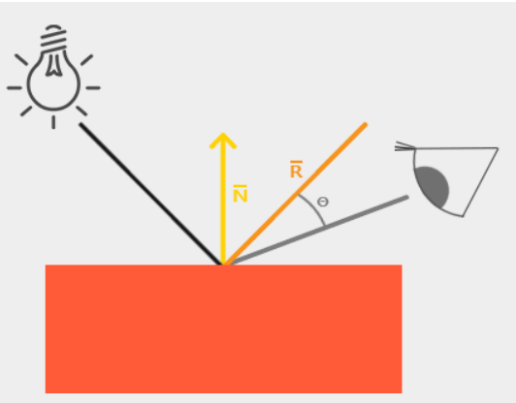

***Figure 11**: Κατοπτρικός φωτισμός*

γυαλάδες πάνω στα αντικείμενα κάνοντας τις επιφάνειες να φαίνονται πιο ρεαλιστικές.

$Ia = LaKa$ ***(I)***

$Id = LdKd \cos\theta$ ***(II)***

$Is = LsKs \cos\alpha^{Ns}$ ***(III)***

Όπου Ka, Kd, Ks, Ns οι συντελεστές των υλικών και La, Ld, Ls οι συντελεστές έντασης του φωτός

Η πηγή φωτισμού αναπαρίσταται ως ένας κίτρινος ήλιος ψηλά στα σύννεφα (fig.1) φωτίζοντας έτσι όλη τη σκηνή. Επειδή το φως βρίσκεται μέσα στην κίτρινη σφαίρα για να φαίνεται και η σφαίρα ότι φωτίζεται πρέπει να γίνει αντριστροφή των normal της σφαίρας.

First Person View

Για να μπορέσουμε να μεταβάλλουμε την κάμερα από την αρχική 3D προβολή σε first person view έστω ότι εξετάζουμε την κίνηση από το πλάι (2D προβολή) για να μπορέσουμε να δούμε τις αποστάσεις που πρέπει να καλύψει η κάμερα στους άξονες y, z (fig.12).

***Figure 12**: 2D προβολή για υλοποίηση First Person View*

Αρχικά η κάμερα έστω ότι βρίσκεται στην θέση **CamPos** και θέλουμε να καταλήξει στην θέση **CamPos'** (x,y1,z1) δηλαδή ακριβώς στα μάτια του avatar. Η κάμερα ακολουθεί ασυμπτωτικά με την πάροδο του χρόνου της τροχιά της εικόνας. Κατά τη διάρκεια αυτής της κίνησης η κάμερα κατεβαίνει κατακόρυφα κατά απόσταση h, μεταβαίνοντας από το y στο y1=y−h. Παράλληλα, μετατοπίζεται προς τα εμπρός κατά μήκος του άξονα z κατά απόσταση ρ, από το z στο z1=z−ρ.

Imgui

Χρησιμοποιήθηκε η βιβλιοθήκη imgui για να μπορέσει να δημιουργηθεί το Helper Window (fig.1). Ουσιαστικά είναι ένα βοηθητικό παράθυρο το οποίο εμφανίζεται στην αρχή του παιχνιδιού για μερικά δευτερόλεπτα και έπειτα εξαφανίζεται. Σκοπός του είναι να ενημερώσει τον παίκτη για τις λειτουργίες του παιχνιδιού δείχνοντας του

τα κατάλληλα κουμπιά που πρέπει να πατήσει για να τις ενεργοποιήσει. Τέλος όταν το avatar φτάσει στον τελικό προορισμό, σπιτάκι, εμφανίζεται ξανά ένα νέο βοηθητικό παράθυρο που ενημερώνει τον χρήστη ότι το παιχνίδι τελείωσε.

***Figure 13**: Game over!*

Source code:

https://github.com/eleftheriaa/pop-up-world.git

Χρήσιμα links :

[1] https://learnopengl.com/Advanced-OpenGL/Cubemaps

[2] https://learnopengl.com/Lighting/Basic-Lighting