

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ, ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ
ΥΠΟΛΟΓΙΣΤΩΝ

Εργασία: Pokemon

Ευάγγελος Λάμπρου
UP1066519

ΠΕΡΙΕΧΟΜΕΝΑ

1 ΕΙΣΑΓΩΓΗ

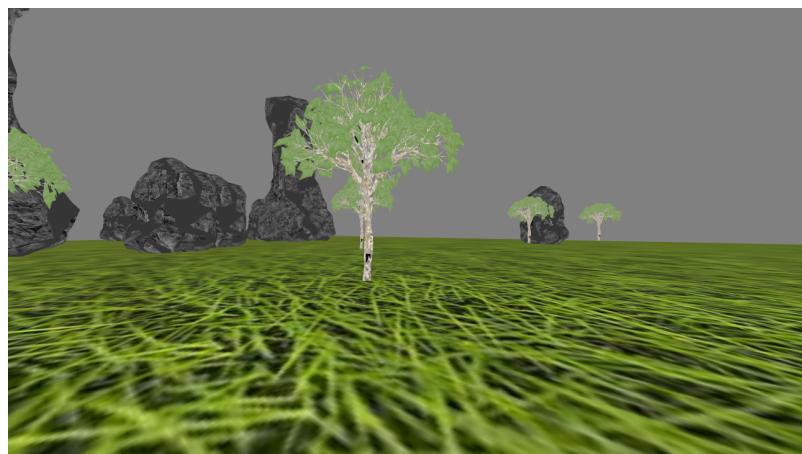
2 ΕΡΩΤΗΜΑΤΑ ΕΡΓΑΣΙΑΣ

2.1 Μέρος Α

2.1.1 Δημιουργία σκηνής κόσμου.

Ο κόσμος τη εργασίας είναι ένα απλό τοπίο αποτελούμενο από δέντρα και βράχους. Σε κάθε εκτέλεση της εφαρμογής δημιουργείται ένα νέο τυχαίο τοπίο, με τα αντικείμενα της σκηνής να τοποθετούνται σε τυχαίες θέσεις.

Σχήμα 2.1: Το τοπίο.

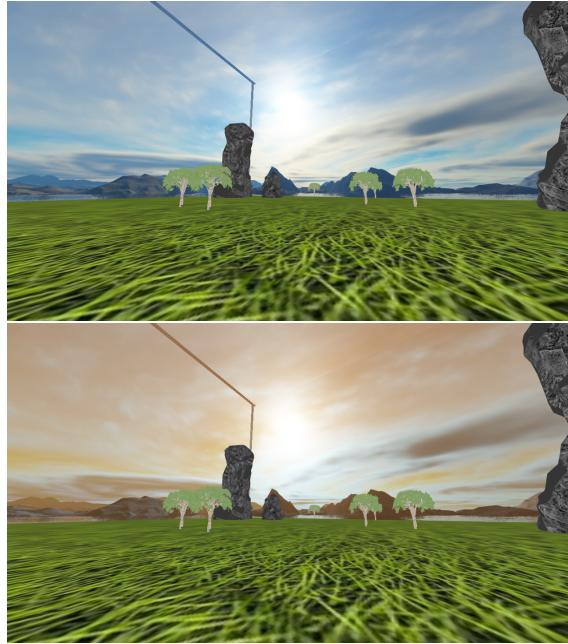


Για τις πέτρες, εφαρμόσαμε normal mapping ώστε να δωθεί καλύτερη αίσθηση υφής σε κάθε βράχο όταν τελικά προσθέσουμε και φωτισμό. Το εφέ, βέβαια είναι αρκετά διαχριτικό.

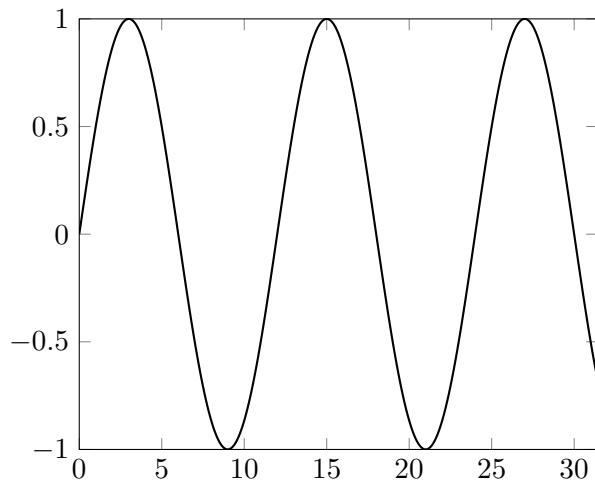
Για το skybox, ακολούθησαμε την τεχνική όπου τοποθετούμε στον ορίζοντα το μοντέλο ενός κύβου το οποίο στις πλευρές του έχει σαν texture την κάθε μία από τις όψεις του ουρανού.

Δώσαμε ακόμα στον κόσμο έναν κύκλο ημέρας-νύκτας, αλλάζοντα το χρώμα του skybox σταδιακά. Συγκεκριμένα, εφαρμόζουμε interpolation ‘ανταλλάζοντας’ τα red και blue channels με βάση μία ημιτονοειδή συνάρτηση.

Σχήμα 2.2: To skybox με χρώματα ημέρας και νύκτας.



$$color = mix(color_{texture}.bgr, color_{texture}.rgb, \frac{\sin(time) + 1}{2}) \quad (2.1)$$



Σχήμα 2.4: Η συνάρτηση ημιτόνου.

Ως πηγή φωτισμού θεωρούμε τον ήλιο. Συνεπώς ερφαρμόζουμε την τεχνική directional lighting όπου κάθε vertex των αντικειμένων της σκηνής μας φωτίζονται από την ίδια κατεύθυνση ανεξαρτήτως της θέσης τους. Πρωτικά θεωρούμε πως η πηγή φωτός απέχει μία άπειρη απόσταση από τα υπόλοιπα αντικείμενα της σκηνής μας. Έτσι, τοποθετήσαμε προσεγγιστικά τον ήλιο σε ένα ‘ψηλό’ σημείο.

Σχήμα 2.5: Το τοπίο με φωτισμό.

/τεξτωιδτη/τεξτωιδτη

2.1.2 Μέθοδος σκίασης

Η σκίαση της σκηνής αποτέλεσε ένα από τα δυσκολότερα μέρη της εργασίας. Ωστόσο, το τελικό αποτέλεσμα είναι ικανοποιητικό.

Η μέθοδος που χρησιμοποιήσαμε είναι αυτή του shadow mapping. Ζωγραφίζουμε την σκηνή από την πλευρά της πηγής φωτός (του ήλιου) σε ένα texture στο οποίο αποτυπώνεται η απόσταση του κάθε αντικειμένου από το φως. Έτσι, μπορούμε να συμπεράνουμε εάν ένα σημείο σκιάζεται.

Για να δωθεί περαιτέρω η αίσθηση για το πέρασμα του χρόνου, έχουμε την φωτεινή πηγή να κινείται γύρω από τη σκηνή για να προσομοιώσει την κίνηση του ήλιου. Για αυτό το λόγο στη σκηνή μας περιοριζόμαστε σε dynamic shadows.

Σχήμα 2.6: Το τοπίο με σκίαση.



Όπως φαίνεται στην εικόνα 2.6, ο παίκτης μπορεί να δει τη σκιά του μέσα στη σκηνή (φαίνεται η σκιά από το μοντέλο ενός γορίλλα στην κορυφή του βράχου).

2.1.3 Περιήγηση στη σκηνή και φυσική αλληλεπίδραση

Για την περιήγηση του χρήστη στη σκηνή, δημιοργήσαμε μία οντότητα κάμερας η οποία είναι υπεύθυνη για τον υπολογισμό των view και projection πινάκων με βάση μία δεδομένη θέση και κατεύθυνση όρασης.

Για να προσθέσουμε φυσικές ιδιότητες στα αντικείμενα της σκηνής, τους προσθέσαμε ιδιότητες στερεού σώματος (rigid body). Συγκεκριμένα, σε κάθε χαρέ της εφαρμογής υπολογίζουμε τη

νέα θέση και ταχύτητα του κάθε αντικειμένου με βάση το διάνυσμα κατάστασής του. Αυτό το διάνυσμα περιέχει πληροφορίες όσων αφορά την τωρινή την θέση, ορμή, γωνιακή ταχύτητα, στροφορμή και περιστροφή.

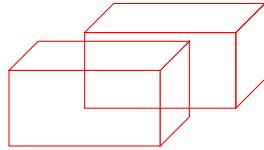
Για να υπολογίσουμε την κάθε επόμενη κατάσταση χρησιμοποιούμε την μέθοδο αριθμητικής ολοκλήρωσης Runge-Kuta 4ης τάξης.

$$s(t) = \begin{bmatrix} x(t) \\ R(t) \\ P(t) \\ L(t) \end{bmatrix} \quad (2.2)$$

Σχήμα 2.7: Το διάνυσμα κατάστασης.

Έτσι, έχουμε αντικείμενα τα οποία μπορούν να υπακούν στο νόμο της βαρύτητας και στα οποία μπορούμε να εφαρμόζουμε αυθαίρετες δυνάμεις.

Για να υπάρχει, ωστόσο, αλληλεπίδραση μεταξύ των αντικειμένων ορίσαμε ένα σύστημα ανίχνευσης συγκρούσεων (collision detection). Σε κάθε αντικείμενο της σκηνής προσθέσαμε ένα AABB (axis-aligned bounding box). Δηλαδή, έναν κύβο του οποίου οι οκτώ (8) κορυφές βρίσκονται στα ακραία vertices του αντικειμένου. Σε περίπτωση όπου στο αντικείμενο εφαρμοστούν transformations, αυτές επηρεάζουν και το αντίστοιχό του AABB. Τελικά, ο έλεγχος συγκρούσεων γίνεται απλά μία υπόθεση σύγκρισης των συντεταγμένων των κορυφών μεταξύ δύο AABBs.



Σχήμα 2.8: Παράδειγμα ελέγχου σύγκρουσης με δύο AABBs.

Με το πάτημα του πλήκτρου T ο χαρακτήρας μας εκτοξεύει μία σφαίρα προς την κατεύθυνση που κοιτάει. Αυτή η σφαίρα αλληλεπιδρά με τις οντότητες του κόσμου της σκηνής αλλά και με άλλες σφαίρες. Ο μαθηματικός τύπος ο οποίος θα μας δίνει τη νέα ταχύτητα της σφαίρας μετά από μία σύγκρουση πέρασε από διάφορες αλλαγές.

Μία επιλογή είναι ο προσεγγιστικός τύπος της ελαστικής κρούσης:

$$\vec{v}' = \vec{v} - 2 \cdot (\vec{v} \cdot \vec{n}) \cdot \vec{n} \quad (2.3)$$

Ο οποίος δίνει ρεαλιστικά αποτελέσματα αλλά απαιτεί διορθώσεις λόγω του τρόπου με τον οποίο υπολογίζουμε το διάνυσμα σύγκρουσης \vec{n} .

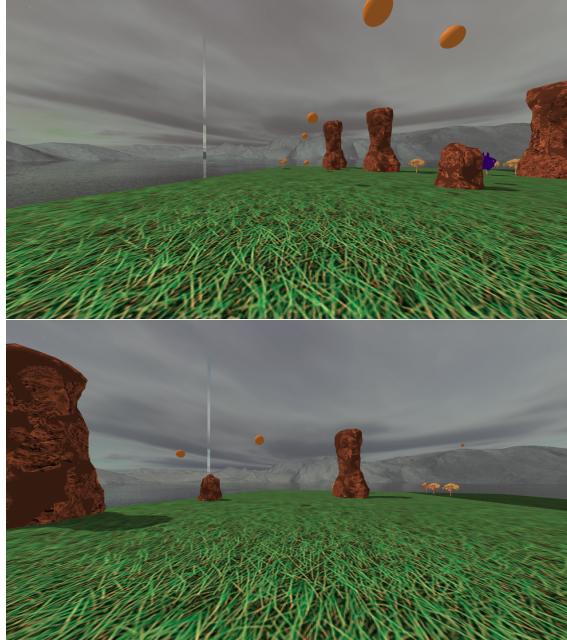
Μία άλλη επιλογή ήταν ο απλοϊκός τύπος

$$\vec{v}' = -\vec{v} \quad (2.4)$$

Ο οποίος κάνει της συγχρούσεις των σφαιρών με το περιβάλλον να μοιάζουν με σκηνή ‘χαρτούν’. Αυτό, φυσικά δεν είναι κάτι ανεπιθύμητο, αλλά έγινε προσπάθεια για πιο ρεαλιστικές συγχρούσεις.

Τέλος, δώσαμε και στον παίκτη φυσικές ιδιότητες, δίνοντάς του ιδιότητες ενός rigid body. Ο χρήστης μπορεί να ανεβαίνει πάνω στα αντικείμενα της σκηνής, να πηδάει και να πετάει.

Σχήμα 2.9: Η σκηνή με μπάλες να αναπηδάνε και με τις δύο εξισώσεις χρούσης.



2.2 Μέρος Β

2.2.1 Εμφάνιση τέρατος

Εφόσων μία σφαίρα αναπηδήσει τρεις φορές στο έδαφος, εμφανίζεται στη σκηνή ένα τέρας στη θέση της. Το εφέ που χρησιμοποιήσαμε είναι ένα εφέ καπνού. Ζωγραφίζουμε με το σύστημα σωματιδίων (particle system) έναν μεγάλο αφιθμό από γκρίζους κύβους οι οποίοι κινούνται προς τα πάνω. Με το πέρασμα του χρόνου, μέσα από τον shader ο οποίος χειρίζεται αυτά τα ‘σωματίδια’ τα μεγενθύνουμε και μειώνουμε την τιμή του alpha, κάνοντάς τα μετά από πέντε (5) δευτερόλεπτα διάφανα.

Τπάρχει ένας εύκολος τρόπος να μεγνηθύνουμε ομογενώς ένα αντικείμενο μέσα από τον vertex shader όπως φαίνεται στον κώδικα (1).

Listing 1: Μεγέθυνση αντικειμένου από τον vertex shader

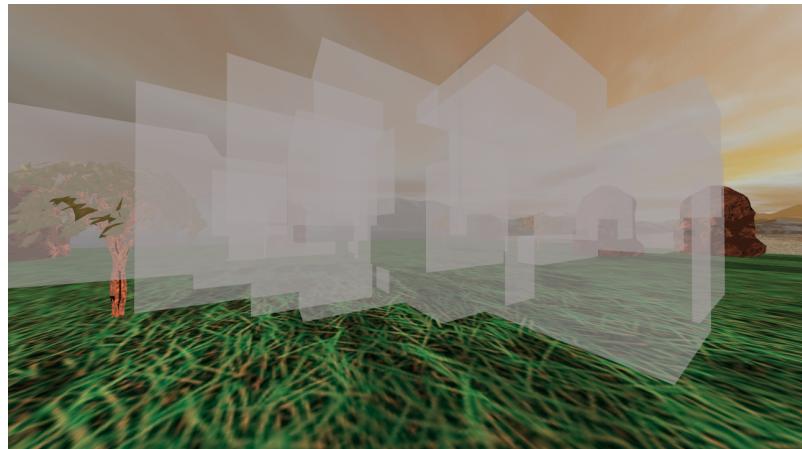
```

1 // scale the object by a factor of c
2 vec4 pos = vec4(vertex_position * c, 1.0);
3 gl_Position = mvp * pos;

```

Ακόμα, για λόγους απόδοσης περιοριστήκαμε σε απλά μοντέλα για το κάθε σωματίδιο (χύβοι και σφαίρες) ώστε να έχουμε χαμηλό αριθμό από vertices. Εφαρμόσαμε επίσης την τεχνική του instancing κατά το οποίο ζωγραφίζουμε στην ουθόνη πολλά αντικείμενα τα οποία μοιράζονται το ίδιο VAO σε ένα (1) μόνο draw call στην κάρτα γραφικών.

Σχήμα 2.11: Ο καπνός πριν εμφανιστεί το τέρας.



Για τα μοντέλα των τεράτων χρησιμοποιήσαμε αρχεία obj's από διάφορους ιστιότοπους.

Σχήμα 2.12: Τα τρία τέρατα.



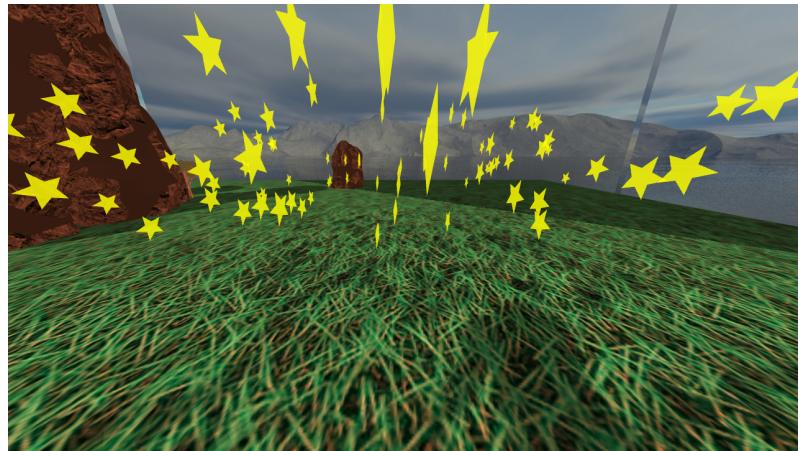
2.2.2 Παγίδευση τέρατος

Πατώντας το πλήκτρο C, ο χρήστης μπορεί να επιλέξει μεταξύ των δύο ειδών σφαίρας οι οποίες είναι:

1. Μπάλες που θα εμφανίσουν ένα τέρας όταν συγχρουστούν με το έδαφος.
2. Μπάλες που θα παγιδεύσουν ένα τέρας όταν συγχρουστούν με αυτό.

Αν ο χρήστης επιλέξει το δεύτερο είδος σφαίρας μπορεί να σημαδεύσει σε ένα τέρας και έτσι να το παγιδεύσει. Η σφαίρα αυτή έχει διαφορεικό χρώμα από την πρώτη. Όταν το παγιδεύει έχουμε ένα απλό εφέ αστεριών.

Σχήμα 2.13: Το εφέ όταν παγιδεύσουμε ένα τέρας.



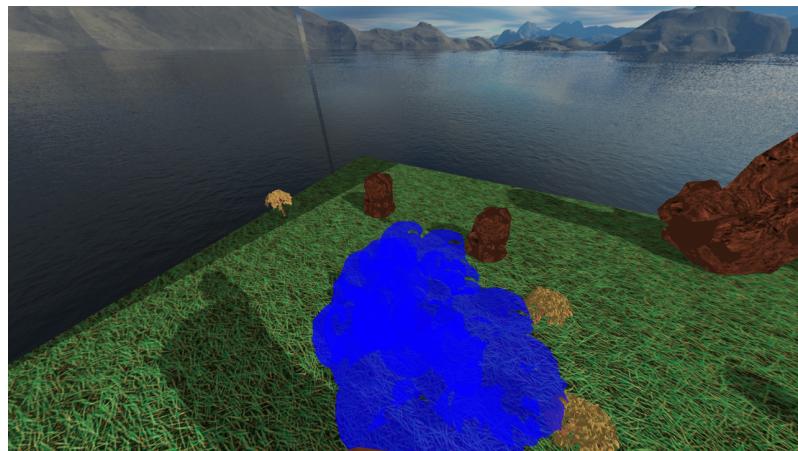
2.2.3 Επίθεση τέρατος

Συνολικά έχουμε τρία (3) διαφορετικά τέρατα. Ένα τέρας φωτιάς, ένα τέρας τοξικής ομίχλης και ένα τέρος ηλεκτρισμού.

Το καθένα έχει μία διαφορετική επίθεση. Η κάθε μία υλοποιείται με ένα διαφορετικό σύστημα σωματιδίων.

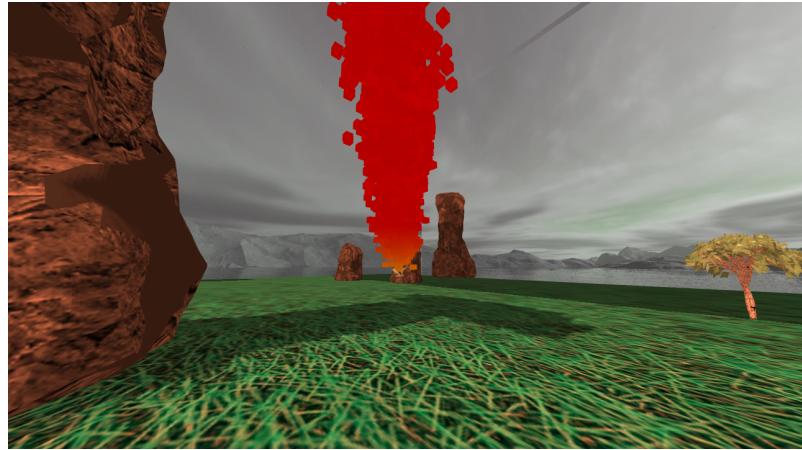
Το τέρας τοξικής ομίχλης εμφανίζει σφαίρες των οποίων τα vertices τα έχουμε μεταβάλει στον vertex shader 'σπρόχνωντάς' τα προς την κατεύθυνση των normals του μοντέλου. Την ποσότητα κατά την οποία το κάθε vertex μετατοπίζεται προς τα έξω ταλαντεύεται ημιτονικά με μία τυχαία μετατόπιση φάσης. Έτσι, έχουμε ένα τελικό αποτέλεσμα που μοιάζει με σύννεφο.

Σχήμα 2.14: Το εφέ δηλητηριώδους καπνού.



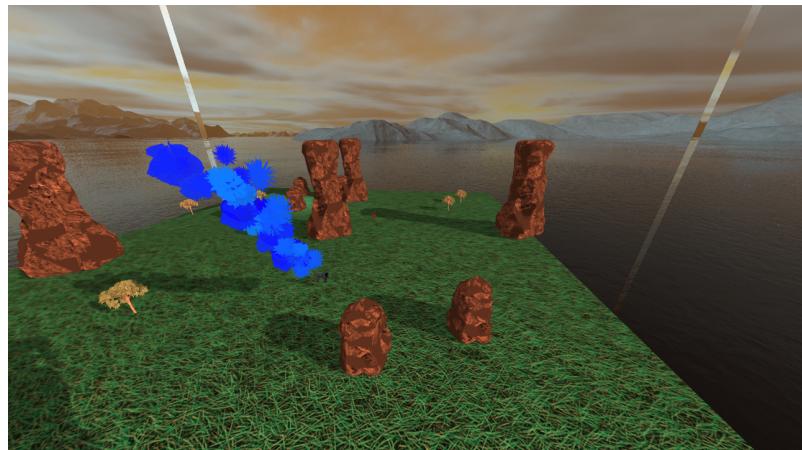
Το τέρας φωτιάς ‘φτήνει’ κύβους οι οποίοι εμφανίζονται σε μορφή νερού συντριβανιού. Αλλάζουμε το χρώμα των κύβων φωτιάς ανάλογα με το ύψος τους στο screen space για να δώσουμε εφέ φλώγας.

Σχήμα 2.15: Το εφέ φωτιάς.



Το τέρας ηλεκτρισμού εμφανίζει σφαίρες οι οποίες ανασβονήνουν έντονα το χρώμα τους και αυξομοιώνουν ‘βίαια’ το μέγεθός τους και τη θέση των vertices τους, πάλι προς την κατεύθυνση των normals τους κάτα μία τυχαία ποσότητα σε κάθε frame.

Σχήμα 2.16: Το εφέ ηλεκτρισμού.



2.3 Bonus

2.3.1 Εξέλιξη τέρατος

Η υλοποίηση μιας εξέλιξης τέρατος έγινε δημιουργώντας αρχικά ένα νέο vertex attribute array στο οποίο θα υπάρχουν στην πρώτη θέση τα vertices του ενός μοντέλου και στη δεύτερη θέση

τα vertices του δεύτερου.

Στη συνέχεια, στον vertex shader κάνουμε γραμμική παρεμβολή μεταξύ των vertices των δύο μοντέλών.

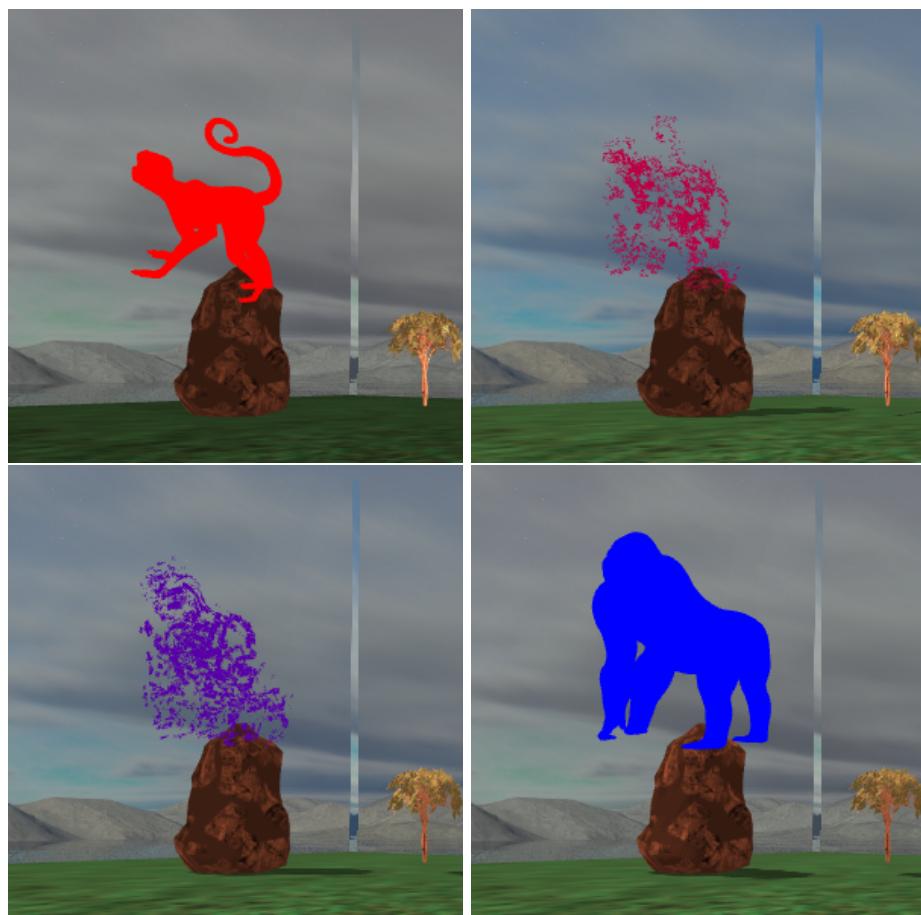
Αυτή η μέθοδος λειτουργεί ιδιαίτερα όταν τα δύο μοντέλα έχουν τον ίδιο αριθμό vertices. Με σκοπό την κάλυψη όλων των περιπτώσεων (το πρώτο μοντέλο να έχει περισσότερα/λιγότερα vertices από το δεύτερο κλπ.) καλούμε τη συνάρτηση `glDrawArrays` με βάση την πρόοδο της μεταμόρφωσης.

Listing 2: Κλήση συνάρτησης με βάση την πρόοδο της μεταμόρφωσης.

```
1  glDrawArrays(GL_TRIANGLES, 0,  
2                  morph_factor < 0.5f ?  
3                  model_first.vertices.size() : model_second.vertices.size());
```

Έτσι, όταν η μεταμόρφωση βρίσκεται νωρίτερα της μέσης ζωγραφίζουμε όσα vertices έχει το πρώτο μοντέλο, αλλιώς προσπαθούμε να ζωγραφίσουμε τον αριθμό των vertices του δεύτερου μοντέλου.

Σχήμα 2.17: Η διαδικασία της μεταμόρφωσης.



3 ΕΠΙΛΟΓΟΣ

Στην εργασία αυτή έγινε προσπάθεια να δημιουργηθεί μία εφαρμογή η οποία παρουσιάζει διάφορικες τεχνικές προγραμματισμού γραφικών.

Στο πρόγραμμα έγινε, ακόμα προσπάθεια υλοποίησης ενός εύχρηστου API, με το οποίο θα έχουμε τη δυνατότητα να επεκτείνουμε εύκολα το παιχνίδι με νέα αντικείμενα και συμπεριφορές. Αυτό μέσω ενός συστήματος διαχείρησης αντικειμέων όπου μπορούμε στη σκηνή να προσθέσουμε στατικά/μη στατικά αντικείμενα σκηνής, particle emitters κ.α. μέσω ενός μοναδικού interface (Engine::addObject). Παρόμοια τεχνική χρησιμοποιήσαμε για τη διαχείριση των assets όπως shaders, models και textures.

Ακόμα, ορίσαμε μερικές αφαιρέσεις για δομές οι οποίες σχετίζονται με την OpenGL όπως μοντέλα και shaders. Στόχος ήταν η απλοποίηση του κώδικα, αν και οι πολλές στρώσεις αφαιρετικότητας μπορεί να οδηγήσουν σε δυσνόητο και πιο αργό κώδικα.

4 ΒΙΒΛΙΟΓΡΑΦΙΑ

- <https://learnopengl.com/>
- <https://tympanus.net/codrops/2019/03/26/exploding-3d-objects-with-three.js/>
- <https://relativity.net.au/gaming/java/SimpleSquareVertexManipulation.html>
- <https://learnopengl.com/Lighting/Light-casters>
- https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_collision_detection_AABB_Collisions
- <https://forum.defold.com/t/morph-one-3d-model-to-another-using-vertex-shader-solved/1914>

Assets που χρησιμοποιήθηκαν βρίσκονται στο αρχείο README.md